# Product Development Resource Allocation with Foresight

Nitin R. Joglekar and David N. Ford

**Contact author:** David N. Ford, Department of Civil Engineering, Texas A&M University, College Station, Tx 77843-3136, USA. Voice: (979) 845-3759. Fax: (979) 845-6554. Email: DavidFord@tamu.edu

**Co-author:** Nitin R. Joglekar, Boston University School of Management, 595 Commonwealth Avenue, Boston, MA 02215, USA. Voice: (617) 353 4290. Fax: (617) 353 4098. E-Mail: joglekar@bu.edu

# Product Development Resource Allocation with Foresight

**Nitin R. Joglekar**
Boston University School of Management
595 Commonwealth Avenue
Boston, MA 02215, USA.
Voice: (617) 353 4290.
Fax: (617) 353 4098.
E-Mail: joglekar@bu.edu

**David N. Ford**
Department of Civil Engineering,
Texas A&M University,
College Station, Tx 77843, USA
Voice: (979) 845-3759
Fax: (979) 845-6554
Email: DavidFord@tamu.edu

2002 International System Dynamics Conference

## Abstract

*Shortening project duration is critical to product development project success in many industries. As a primary driver of progress and an effective management tool, resource allocation among development activities can strongly influence project duration. Effective allocation is difficult due to the inherent closed loop flow of development work and the dynamic demand patterns of work backlogs. The Resource Allocation Policy Matrix is proposed as a means of describing resource allocation policies in dynamic systems. A simple system dynamics model and control theoretic model of resource allocation in a product development model are developed. The control theory model is used to specify a foresighted policy, which is tested with the system dynamics model. The benefits of foresight are found to reduce with increasing complexity. Process concurrence is found to potentially reverse the impact of foresight on project duration. The model structure is used to explain these results and future research topics are discussed.*

**Keywords**: product development, resource allocation, policy analysis, control theory, project management, concurrent development

## Introduction

Failing to complete projects by their deadlines can lead to staggering levels of cost escalation and technology obsolescence in defense related product development projects (McNulty, 1998). Meeting the development schedule deadlines has been identified as the most important concern for product development project performance in many fields ranging from construction to software development (Patterson 1993, Meyer 1993, Wheelwright and Clark 1992). Good schedule performance is particularly difficult in development projects due to the iterative nature of the process that creates closed loop flows of work. Two primary approaches to improving schedule performance are process improvements and resource management.

A variety of process improvement approaches to improving schedule performance have been explored including, the use of information technology tools (Joglekar and Whitney 1999), cross functional development teams (Moffatt. 1998), and increased levels of concurrence (Backhouse and Brookes 1996). Increasing concurrence in particular has become a common form of process improvement. Research directed at concurrent development process design has addressed several aspects of iteration, including information evolution (Krishnan 1996, Krishnan, Eppinger, Whitney 1995) and activity order and grouping (Smith and Eppinger 1997a, 1997b). Studies that build formal models of the concurrent development approach have been addressed in previous system dynamics work by the authors and others (Ford and Sterman, 1998a, 1998b, Cooper 1980, 1993, Cooper and Mullen. 1993, Abdel-Hamid and Madnick 1991). The nature of process concurrence relationships is often governed by the product architecture. For example in semiconductor chip development the design phase specifies the locations of components and electronic pathways on the chip. Prototype building requires all of the design to be completed before prototype construction can begin, thereby requiring a sequential concurrence relationship between the two phases. These constraints on process concurrence often leave managers with few degrees of freedom with which to alter the process once the architecture is defined (Joglekar et al., 2001).

Because of the difficulty in reducing project duration through process improvement effective resource management is important to the timely completion of projects. Managers can have a large effect on resource utilization even when the total quantity of resources (e.g. the number of developers) is fixed. Both resource productivity improvement and policies for allocating resources among specific development activities can be used to speed up projects. The current work focuses on resource allocation policies as a means of reducing project duration. In contrast to process improvements, the resource allocation policies associated with the dynamics of product development in general and the rework cycle in specific have not been extensively researched. Within a concurrent development context two resource allocation questions must be addressed to improve policy design and implementation:

1. How should product development project managers allocate resources to shorten project duration?
2. How does concurrence in a product development process impact the effectiveness of resource allocation policies?

A simple approach is taken here to develop initial but valuable insights on these issues. First, foresight, an omnipresent and potentially powerful form of information processing used by managers, is chosen as a focus. As discussed here, foresight is the use of resource demand forecasts as the basis for allocation. Second, models that are very simple relative to actual product development practice are used to expose the relationships between policy structure and project behavior. For example, total resource quantities and productivity are assumed fixed and concurrence relationships are assumed to be known a-priori and immutable. Projects are described with two important development project characteristics: complexity and concurrence. These assumptions (as well as others made in modeling) allow us to integrate system dynamics and control theory to derive insights that are not available by either alone, but at the cost of limiting the range of applicability of our results. We describe model extensions to partially address these limitations in our conclusions. Here a system dynamics model and control theoretic model of a simple product development rework cycle are used in combination to investigate the effectiveness of foresighted resource allocation policies in reducing project duration.

## The Challenges of Resource Allocation in Product Development Projects

Development project resource management can improve schedule performance by increasing the quantity of resources, productivity, and other means (Clark and Wheelwright, 1993). Total resource quantities and their productivity are often limited and difficult to improve. In contrast, the allocation of scarce resources among development phases and activities is a realistic management opportunity for improving schedule performance. A complete dynamic view of product development project resource management must include delays in making allocation decisions, the desire of the workforce to minimize the number and frequency of reassignments, and delays in productivity ramp-up during the implementation of allocation decisions. Effective resource allocation is made more difficult by the challenges in accurately predicting the amounts of work to be initially completed, inspected or tested to discover change requirements, and reworked, and by the different types of concurrence relationships among these activities.

Resource allocation practice in many projects is based on a simple approach: look at the current backlogs of work to be serviced by the different development activities and allocate resources to each activity in the same proportion that the activity's backlog contributes to the total amount of work waiting to be done. Such a directly proportional policy is an example of how managers use relatively simple information structures and heuristics to bridge the gap between high project complexity and their bounded rationality (see Ford 2002 for discussion and examples). But, as will be shown, the dynamic structure of development causes directly proportional policies to be relatively ineffective in minimizing project duration. As described by Ford and Sterman (2002), that dynamic structure includes important sequencing and concurrence constraints. The development of any given piece of work can only occur in a particular sequential order within a single development phase (e.g. initially complete, inspect, correct errors) and within individual projects (e.g. plan, design, build). These development activities occur over characteristic durations that determine minimum phase durations. This aging chain structure delays the development of backlogs in downstream activities and phases, distorting optimal resource allocations away from those described by current backlogs. In addition, development activities and phases are coupled through closed, conserved rework cycles that change future backlogs in ways that are not reflected in current backlogs. In addition to delaying the evolution of backlogs, development processes create additional work for some activities through rework cycles. Directly proportional policies are myopic in that they do not fully capture future resource needs in current resource allocations.

The delays and closed conserved flows inherent in development and resource allocation suggest that the allocation targets set by managers should be based on future resource needs. This requires that managers forecast the demands for various development activities. We term policies that incorporate future conditions in current allocations "foresighted" and propose them as a potentially improved type of resource allocation policy. Practising managers probably include some foresight into policies intuitively. But the tacit nature of these policies and their design prevent their description, evaluation, and improvement. Foresighted resource allocation policies are also intuitively attractive to system dynamists because they can potentially balance short and long-term objectives. But do foresighted resource allocation policies always improve schedule performance compared to myopic resource allocation policies? How should managers anticipate the impacts of development processes and rework on resource demands and adjust resource allocations accordingly?

## The Resource Allocation Policy Matrix

Describing policies for allocating resources to development activities that are based on the backlogs of work to be processed requires a means of relating each work backlog to each process. Prior research (Ford and Sterman 1998b) has established the backlogs of work to be completed (Wc), work ready for quality assurance after completion (Wqa), and work ready for iteration (Wit) as critical elements in managing product development rework cycles. In our model each backlog is reduced by the development activity with the same name (i.e. Completion, Quality Assurance, or Iteration). We introduce the Resource Allocation Policy Matrix (Figure 1) as a tool to describe how individual work backlogs influence, separately and in combination, the activities that serve those backlogs. Each cell of the Matrix quantifies the relative influence of one work backlog, $W_B$, $B \in \{\text{Completion, Quality Assurance, Iteration}\}$, on one development process, identified in Figure 1 by the backlog that the process services. When these relative influences are combined for each backlog and compared across activities they specify how resource allocations respond over time to work backlogs.

**Work Backlog (Wb)**

| Development Activity | | Wc | Wqa | Wit |
|---|---|---|---|---|
| | Completion | 1 | 0 | 0 |
| | Quality Assurance | 0 | 1 | 0 |
| | Iteration | 0 | 0 | 1 |

Figure 1: A Resource Allocation Policy Matrix describing a Directly Proportional Policy

Mathematically, the relative influence of a backlog on a development activity is the product of the size of the backlog and the matrix coefficient in the {Development Activity, Work Backlog} cell. The total relative influence of all backlogs on an activity is the sum of the relative influences of individual backlogs. The proportions of the total relative influence on each activity determine the fractions of the resources applied to the activities. A directly proportional policy (Figure 1) illustrates a simple resource allocation policy. A verbal form of this policy is "Allocate the same fraction of the available resources to each activity as the fraction of the current total work backlog that the activity's backlog contributes to the total work backlog." The zeros in the non-diagonal terms of the Resource Allocation Policy Matrix specify that only the backlog reduced by each activity impact the amount of resources received by that activity. The equal values of the diagonal terms specify that no backlog receives more or less than indicated by the size of its backlog. Note that in this and all Resource Allocation Policy Matrices the resource allocation is dynamic even though the coefficients are constant (except for an empty matrix) because the backlog sizes vary dynamically, thereby constantly changing the resource fractions.

A directly proportional resource allocation policy is attractive to managers because it is relatively easy to design and implement. The policy requires very little information processing because each activity is impacted only by it's own backlog. However, a directly proportional allocation policy might be myopic in that it does not adjust for the previously described delays and backlog changes created by processing work. Alternatively, a foresighted policy that accounts for future resource needs is derived using a linear model of a system with closed loop feedback control (based on McDaniel, 1996). Foresighted resource allocation policies use more information in the form of additional work backlogs, to determine allocation fractions. Therefore, for typical development scenarios, foresighted matrices have non-zero off diagonal terms.

Resource Allocation Policy Matrices facilitate the design of improved policies and the explanation of the factors that influence these policies by explicitly and clearly describing the individual relationships between each work backlog: development activity pair. We use Resource Allocation Policy Matrices to describe the two types of policy described above. In a subsequent section we derive a foresighted matrix and show that comparing these matrices can develop an intuitive understanding of the resource allocation choices. This comparison helps explain how the foresighted policy allocates more resources to a bottleneck task identified in previous research (Cooper 1993) than the directly proportional policy allocates.

## Basic System Model

Our basic model of the work in a development project builds on a standard rework cycle model from the system dynamics literature (Ford and Sterman 1998b) as shown in figure 2. In a separate model sector resources are allocated among the Completion, Quality Assurance, and Iteration activities based proportionally on the indicated demand for each. In the system dynamics model these indicated demands are adjusted from the levels indicated purely by the current size of activity backlogs according to the Resource Allocation Policy Matrix to reflect specific managerial policies. Therefore, generally, the demand for an activity is the weighted sum of the demands created by each backlog for that activity. In a directly proportional policy this reduces to the resource demand for each activity (Completion, Quality Assurance, or Iteration) being the backlog served by the activity (Wc, Wqa, or Wit, respectively in Figure 2). In the foresighted policy the resource demand for each activity is the sum of these same three backlogs after each backlog has been multiplied by its relative influence or "weighting" as described in the Resource Allocation Policy Matrix.

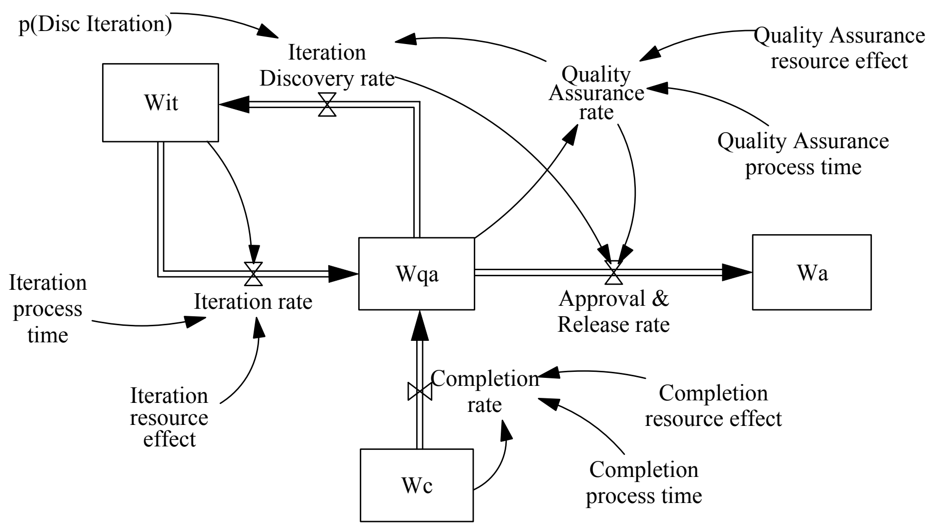

Figure 2: A Capacitated Product Development Project Rework Cycle
(Based on Ford and Sterman1998b)

Formally, the resource impact is modelled as a fraction that reduces the uncapacitated rate of progress of each activity. The fraction is the ratio of the progress based on the amount of resources provided to the maximum rate allowed by resources. The progress based on provided resources is the product of total quantity of resources, resource productivity, and the fraction allocated to the specific activity. It is through the allocated fraction that the resource allocation policy impacts system behaviour. See appendix 2 a listing of the of model equations.

To specify this resource impact we define the allocation fraction as:

$$f_i = \sum_B (W_B * C_{B,i}) / \sum_i (\sum_B (W_B * C_{B,i}))$$

This equation mirrors the impact of the Resource Allocation Policy Matrix in the control theory model (described next). $C_{B,i}$ are the coefficients taken from the Resource Allocation Policy Matrix. Recall that these coefficients form an identity matrix in the case of a directly proportional allocation policy. Next we use a control theoretic model from linear systems theory for the closed loop control of resources to derive the coefficient for the foresighted policies. .

## Linear Optimal Control of Resources

The strength of the close loop linear control model lies in its ability to specify the resource allocation policy associated with an optimal controller. We developed a continuous time (CT) formulation of the system dynamics model described above that included a closed loop optimal controller. This work was based on the discrete time formulation developed by McDaniel (1995). The continuous time formulation includes two types of feedback loops (Figure 3). The small loop in Figure 3 represents the autonomous (without management control) impacts of the systems behaviour on the system to be controlled (a.k.a. the plant in control theory nomenclature). The large feedback loop in Figure 3 represents the use of resource allocation to impact system behaviour. In this formulation the equations in a traditional system dynamics model are replaced by the coefficients in matrices that relate the state variables (the stocks) to the changes in state variables (the rates).

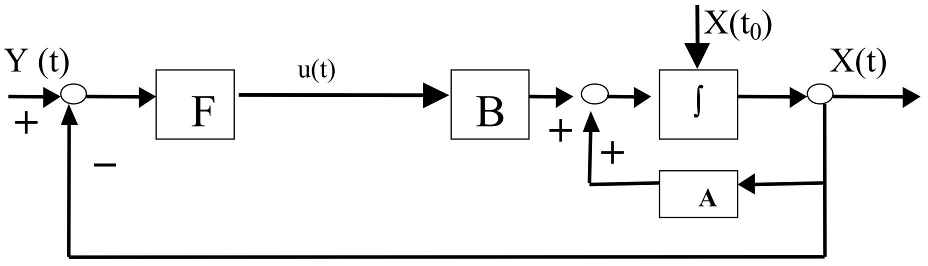

Figure 3: Resource Allocation with Closed Loop Control

In order to set up formal analysis based on the resource allocation control loop, we define the system behaviour as X(t), a vector of the three stocks of work to be processed (Wc(t), Wqa(t), Wit(t)) that describes the sizes of these stocks over time. Y(t) is the corresponding vector of stock values that describes the desired schedule. B is the rate transformation matrix (Figure 4) that computes the various rates. It is mathematically equivalent to the system dynamics model shown in Figure 2 and specified by equations (1) through (9) for uncapacitated conditions. The diagonal terms of this matrix, with negative signs, are the reciprocals of the completion times ($\tau_i$). The off diagonal terms represent the rework and approvals rates.[1] Recall that $\tau_c$, $\tau_{qa}$, and $\tau_{it}$ are the times for completion, and p is the probability of rework.

[1] The discrete time equivalent formulation is called the work transformation matrix (Smith and Eppinger, 1997). We have added the negative sign for algebraic convenience in setting up the state equation.

**Work Backlog (Wb)**

| | | Wc | Wqa | Wit |
|---|---|---|---|---|
| **Development Activity** | **Completion** | $(-1)/T_c$ | 0 | 0 |
| | **Quality Assurance** | $1/T_c$ | $(-1)/T_{qa}$ | $1/T_{it}$ |
| | **Iteration** | 0 | $p/T_{qa}$ | $(-1)/T_{it}$ |

Figure 4: The Rate Transformation Matrix (B)

The influence of resource capacity constraints and allocation policies are described in a separate matrix, F. This matrix is the Resource Allocation Policy Matrix. This matrix is applied to the difference between the actual behaviour (X(t)) and the desired behaviour (Y(t)) in a manner similar to the application of an adjustment to a gap between desired and actual conditions in many system dynamics models (Sterman 2000). Therefore the control signal sent to the system is given by:

$$u(t) = -F \{ X(t) - Y(t) \}$$

Like in system dynamics, this control theory model uses the state conditions to determine the rates of change. In contrast to many system dynamics models, this particular control theory model uses full-state feedback, meaning that the sizes of all the backlogs (not a subset of the backlogs) are used to determine the control signal. Substituting the notation above into the usual control theory notation, the state equation is:

$$d X(t)/ dt = A X(t) + B u(t)$$

The optimisation seeks a policy that will minimize the cost J over the duration of the project where the cost is defined as:

$$J = \int \{X'QX + u'Ru + 2*X'Nu\} \, dt$$

The three terms within the integrand represent the three contributors to quadratic costs (namely, system conditions, X, the control signal, u, and the interaction between them). Q, R, N are weights on this three types of costs; these are defined in the appendix. The nature of the cost function forces the optimisation process to reduce both X (the stock of work), and u (the effort needed) at once. Therefore, this objective function also reduces the project duration in the most cost-effective manner. This formulation yields the linear time invariant Riccati Equation:

$$SA + A'S - (SB+N)R^{-1}(B'S+N') + Q = 0$$

Where: $F = - R^{-1}B'S$ and S is the solution to the Riccati Equation

This equation was solved numerically using Matlab. The set up parameters for the numerical analysis and the results for p=0.64 are available in the appendix. The model generated the following Resource Allocation Policy Matrix for the time constants shown in the model equations and p = 25%.

| | Wc | Wqa | Wit |
|---|---|---|---|
| **Completion** | 0.21 | -0.01 | 0.00 |
| **Quality Assurance** | 0.13 | 0.15 | 0.13 |
| **Iteration** | 0.07 | 0.06 | 0.27 |

Figure 5: Resource Allocation Matrix derived from the Control Theory Model for p=0.25

Note that the values in Figure 5 are relative influences. Those values are summed across the rows and those sums are proportioned to determine allocation fractions for the three activities. A comparison of the Resource Allocation Policy Matrices in Figures 1 (directly proportional) and 5 (foresighted) provides an intuitive understanding of the impacts of foresight on resource allocation. The foresighted policy increases the influence of iteration on the amount of resources used to process that backlog (0.27 versus the average diagonal value of 0.21) and decreases the influence of the quality assurance backlog on the resources applied to process that backlog. More importantly, the foresighted policies also add important impacts of backlogs on activities that do not process the work in that backlog (off diagonal terms). Notice in particular that resources are shifted away from initial completion and toward quality assurance. This is consistent with the results of previous system dynamics research on means of reducing project durations (Cooper 1993) and the previously described generation of future backlogs by development processes that are not captured in myopic resource allocation policies.

The coefficients in Figure 5 were used in the system dynamics model as weightings to alter the indicated sizes of work backlogs and thereby the relative demands for resources for each activity. In this way the policies described by the Resource Allocation Matrix generated by the control theory model were implemented in the system dynamics model, which simulated project duration.

## Simulation Results

Initially the project is assumed to be as concurrent as the conserved closed loop structure allows. Project duration using directly proportional and foresighted policies for a simple and a complex project were simulated in the system dynamics model. Simple and complex are described with the fraction of inspected work that is discovered to require rework. A simple project is assumed to have 25% rework probability and a complex project is assumed to have a 64% rework probability.

Table I: Foresighted Policies with Full Concurrence

| Resource Alloaction Policy | Type of Concurrence | Simple Task: 25% Rework Probability | | Complex Task: 64% Rework Probability | |
|---|---|---|---|---|---|
| | | Duration* | % Reduction | Duration* | % Reduction |
| With Foresight | Full | 44 | 19% | 94 | 6% |
| Conventional | Full | 55 | | 100 | |

* Results show elaspsed time in weeks for reaching 99% Completion

The results in table I support our basic premise that foresighted policies improve schedule performance with full concurrence. The simple project finishes 19% faster and the complex project finishes 6% faster with foresighted policies than with directly proportional policies. More complex and therefore difficult projects are traditionally expected to benefit most from foresight. However, the simulations show a counterintuitive result: more complex projects appear to benefit *less* from foresighted policy than simple projects.

In the case of full concurrence all the work to be accomplished is assigned to the initial condition of the stock for work to be completed (Wc) and is therefore available for initial completion. We enhance the basic structure of the system dynamics model to allow for partial concurrence as follows. An additional stock (Wna, work to be completed but not available) is added. The outflow from this stock feeds into the stock of work to be completed (Wc). Partial concurrence is described by starting with all the work in Wna and controlling its outflow based on the amount of work approved (Wa) through a look up function. Specification of the partial concurrence relationship is based on the table function described in Ford and Sterman (1998a). The concurrence relationship used consists of two linear relationships as shown by the term $\partial$ (in appendix 2). We assume that the act of releasing the work based on the stock of the work approved does not require any development resources. Hence, neither Wa, nor Wna figure in the derivation of the governing resource allocation matrix.

Table II: Foresighted Policies with Partial Concurrence

| Resource Alloaction Policy | Type of Concurrence | Simple Task: 25% Rework Probability | | Complex Task: 64% Rework Probability | |
|---|---|---|---|---|---|
| | | Duration* | % Reduction | Duration* | % Reduction |
| With Foresight | Partial | 287 | -1% | 678 | -17% |
| Conventional | Partial | 285 | | 581 | |

* Results show elaspsed time in weeks for reaching 99% Completion

A comparison of simulation results in table I and II indicates that a closed loop foresighted policy, which effectively improves schedule performance in a concurrent project, can degrade schedule performance in a partially concurrent project. These results will further discussed in the next section.

## Discussion

Consistent with the system dynamics tradition (Forrester 1961, Sterman 2000), our goal is neither to show how optimisation can improve existing modelling practice nor to present the best optimisation techniques for a set of resource allocation problems. Instead, we use the optimal results as a benchmark to discuss how conventional managerial intuition might be informed through policy changes and some related issues. We first explain the counter intuitive results presented in table 1 and 2. We then discuss one challenge to successfully implementing foresighted resource allocation policies to reduce project duration, the "yanking" of labor. We use this discussion to identify some of the limitations of our control theory model and suggest opportunities for over coming these limitations by developing more sophisticated system dynamics models.

The foresighted Resource Allocation Policy Matrix is used to explain the counterintuitive result for full concurrence as follows. If the stocks Wc, Wqa, and Wit remained constant and equal the directly proportional and two foresighted policies allocate resources in the percents shown in columns 2-5 of Table III. Columns 4 and 5 show the bias of the foresighted policies when compared with the directly proportional policy:

| **Development Activity** | **Conventional (Proportional) Policy** | **Policy with Foresight** | | **Foresight Bias Over Proportional** | |
|---|---|---|---|---|---|
| | | *P=0.25* | *P=0.64* | *P=0.25* | *P=0.64* |
| *Completion* | 33% | 20% | 9% | -14% | -25% |
| *Quality Assurance* | 33% | 41% | 36% | 7% | 3% |
| *Iteration* | 33% | 39% | 55% | 6% | 22% |
| Total | 100% | 100% | 100% | 0% | 0% |

Table III: Allocation Fractions with Equal Backlogs

For the parameters used in this analysis and with low complexity (p=0.25), the quality assurance activity requires the largest fraction of resources (41%) and is therefore the bottleneck. However, with increased complexity, i.e. when the probability of rework increases from 0.25 to 0.64, the iteration activity requires the most resources (55%) and becomes the bottleneck. The more complex project increases the allocation 7% compared to the directly proportional case where as the simple project increases the allocation only 3%.

The structure of the development processes provides an explanation for this reduction in the marginal improvement in schedules with increasing complexity. The marginal benefit of foresight shrinks with increasing complexity because both foresight and complexity increase the fraction of total effort (and therefore resources) needed away from initial completion and toward quality assurance and iteration. Complexity does this by increasing the fraction of work "trapped" in the closed flow of quality assurance and iteration. This increases the size of the quality assurance and iteration backlogs, thereby increasing the resource allocations to those activities. Project durations of more complex projects are still longer than simpler ones because increasing complexity also increases the total work effort required to complete the project. Foresight shifts resources toward quality assurance and iteration by taking the future of the project into account and shifting resources to accommodate future quality assurance and iteration backlogs. Therefore, increasing project complexity and iteration have a similar effect on resource allocation and more complex projects are less able to take advantage of foresighted policies.

The Resource Allocation Policy Matrices also help explain the degradation of schedule performance under partially concurrent conditions. As modelled here, foresighted policies cannot account for work that becomes available for initial completion only after the beginning of the project. The policy therefore is based on the work available at the project start. For the concurrence modelled here, and many experienced in practice (see Ford and Sterman 1998a) this represents only a small fraction of the total scope. This is a limitation of the current control theory model formulation. We speculate that partial concurrence, increases in work available but not completed (Wc) through exogenous task creation, or both are best modelled with a Kalman-Busy type "look ahead" controller that dynamically predicts work loads. This could effectively generate coefficients in our Resource Allocation Policy Matrix that evolve dynamically during projects. Such an analysis lends itself to system dynamics modelling (Anderson, 2001), but the implementation lies beyond the scope of this paper.

The Resource Allocation Policy Matrix and our combined use of a control theory and system dynamics models can provide insight into other important resource management issues. One example is "yanking" in which workers are moved frequently from one activity to another. Workers involved in product development setting dislike frequent reassignment, which results in a feeling of being “yanked around” across activities or being given different work priorities. The switching of assignments across activities also affects the organisational learning and thus work force productivity. We measured yanking with the cumulative absolute value of changes in the resource allocation fraction for each development activity.

Table IV: Yanking for Fully Concurrent Complex Project

| **Development Activity** | **Amount of Yanking** | |
|---|---|---|
| | *Directly Proportional Policy* | *Foresighted Policy (p=64%)* |
| *Completion* | 0.94 | 0.19 |
| *Quality Assurance* | 0.41 | 0.04 |
| *Iteration* | 0.53 | 0.15 |

As shown in Table IV, the foresighted policy reduces yanking dramatically, by 80%, 90%, and 72% for the completion, quality assurance, and iteration activities, respectively. This suggests that, in addition to reducing durations in fully concurrent projects, foresighted policies provide an added benefit of less yanking, which can improve labor productivity.

But reducing yanking may not occur even though it would improve project performance. Although managers may want to reduce yanking, they are also pressured by their bosses to allocate resources based on the existing and very visible backlogs of unfinished work. Because the sizes of future backlogs (especially rework) that any given set of development processes, project characteristics, and managerial policies will create is difficult to forecast a priori, managers may tend to shy away from foresighted policies or keep their policy design processes tacit and intuitive.

## Conclusions

In this work we integrate a traditional control theory based derivation of optimal resource allocation and a system dynamics model. We use the control theory model to derive an optimal allocation policy, which we describe with a Resource Allocation Policy Matrix. The Matrix is useful in explaining differences in project performance and developing and intuitive understanding of the characteristics and impacts of different allocation policies. Resource allocation policies were then used in a system dynamics model of the system to test performance. **Our results show that and how full concurrence foresighted policies can improve schedule performance (i.e. reduce cycle time), without increasing the total amount of resources.** While preliminary, our results could have far reaching impacts on resource management through allocation policies because the schedule improvements are essentially free, requiring no additional resources. However, counter-intuitively, gains in performance are less for complex projects than for simple projects. These results suggest that an improved understanding and heuristics for the design of foresighted resource allocation policies can improve product development project schedule performance. In contrast, our results suggest that with partial concurrence conventional managerial wisdom prevails over the “foresighted” policies. An improved model and understanding of the impacts of process concurrence is required to adequately develop and test such policies. Therefore, additional modelling and analysis of the relationships investigated here is needed to advance this application of system dynamics to product development project planning and management.

## References

Abdel-Hamid, Tarek and Madnick, Stuart (1991). *Software Project Dynamics, An Integrated Approach.* Prentice-Hall. Englewood Cliffs, NJ.

Backhouse, C. J. and N. J. Brookes (1996). *Concurrent Engineering, What's Working Where*. The Design Council. Gower. Brookfield, VT.

Cooper, K. G. (1980). Naval Ship Production: A Claim Settled and a Framework Built. *Interfaces* 10(6) 20-36.

Cooper, Kenneth, G. (1993) The Rework Cycle: Benchmarks for the Project Manager. *Project Management Journal*. 24:1:17-22

Cooper, K. G., T. W Mullen. (1993). Swords and Plowshares: The Rework Cycle of Defense and Commercial Software Development Projects. *American Programmer.* 6(5).

Ford, D. (2002). "Achieving Multiple Project Objectives through Contingency Management" *ASCE Journal of Construction Engineering and Management*. 128(1):1-10. February, 2002.

Ford, D. and Sterman, J. (1998a), "Expert Knowledge Elicitation for Improving Formal and Mental Models," *System Dynamics Review*. 14(4):309–340.

Ford, D. and Sterman, J. (1998b). "Modeling Dynamic Development Processes," *System Dynamics Review*. 14(1): 31-68.

Ford, D. and Sterman, J. (2002) "Iteration Management for Reduced Cycle Time in Concurrent Development Projects" working paper available from the authors.

Forrester, J.W. (1961). *Industrial Dynamics.* Cambridge, MA. Productivity Press.

Joglekar N.R. and D.E. Whitney. (1999). "Automation Usage Pattern during Complex Electro Mechanical Product Development." MIT Center for Technology Policy and Industrial Development Report, prepared under contract number F33615-94-C-4429 from the US Air Force Research Laboratory (US-AFRL).

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

Krishnan, V. (1996). Managing the Simultaneous Execution of Coupled Phases in Concurrent Product Development. *IEEE Transactions on Engineering Management*. 43(2) 210-217.

Krishnan, V., S. D. Eppinger, D. E. Whitney. (1995). A Model-Based Framework to Overlap Product Development Activities. *Management Science*. 43 437-451.

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

McNulty, R. (1998). Reducing DOD Product Development Cycle Time: The Role of Schedule in Development Process, Unpublished Doctoral Dissertation, (TMP/Aero), MIT Cambridge, MA.

Meyer, C. (1993). *Fast Cycle Time, How to Align Purpose, Strategy, and Structure for Speed*. The Free Press. New York. .

Moffatt, L. K. (1998). Tools and teams: competing models of integrated product development project performance. Journal of Engineering Technology and Management. 15:55-85.

Nevins, J. L. and D. Whitney (1989). *Concurrent Design of Products & Processes, A Strategy for the Next Generation in Manufacturing*. McGraw-Hill. New York.

Patterson, M. L. (1993). *Accelerating Innovation, Improving the Process of Product Development.* Van Nostrand Reinhold. New York. .

Smith, R. P. and S. D. Eppinger. (1997a). A Predictive Model of Sequential Iteration in Engineering Design. *Management Science*. 43(8) 1104-1120..

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

Sterman, J. S. (2000). *Business Dynamics, Systems Thinking and Modeling for a Complex World*. New York. Irwin McGraw-Hill.

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

## Appendix 1: The solution to the Control Theory Model

For numerical analysis, we have set up the following parameters
Time Constants: $\tau_c = 2$ days $\tau_{qa} = 1$ day $\tau_{it} = 2$ days
System Matrices: A = Q= R = I ; N = Null
For p= 64% these parameters yield the following resource allocation matrix:

| | Wc | Wqa | Wit |
|---|---|---|---|
| Completion | 0.09 | 0.00 | 0.00 |
| Quality Assurance | 0.12 | 0.13 | 0.12 |
| Iteration | 0.15 | 0.15 | 0.24 |

## Appendix 2: System Dynamics Model Equations

**Stocks**

$(d/dt)\, W_{na} = - r$ (1)

$(d/dt)\, W_{c} = r - c$ (2)

$(d/dt)\, W_{qa} = c + it - d - a$ (3)

$(d/dt)\, W_{a} = a$ (4)

$(d/dt)\, W_{it} = d - it$ (5)

where: $W_{na}$ – Work not available for initial completion
$W_c$- Work needing Initial Completion
$W_a$ - Work Approved
$W_{qa}$ - Work needing Quality Assurance
$W_{it}$ - Work known to need Iteration (changes)
r – Release work for initial completion
c – Initial completion rate
it - Iteration rate

d - Discovery of need for Iteration rate
a - Approval rate

**Flows and Auxiliaries**

$P_i = (w_i / \tau_i) * (R_i / R'_i)$ (6)

$R_i = R_T * f_i * p_i$ (7)

$d = qa * p(d)$ (8)

$a = q – d$ (9)

$r = \mathrm{Max}\,(W_{na}, (w_c *S) - (S – (W_a + W_{qa} + W_{it}))) / \tau_{na}$ (10)

$w_c = f\partial(W_a /S)$ (11)

where for activity i and $i \in \{c, qa, it\}$:

- $P_i$ – Progress of activity i
- $w_i$ – Work available activity i
- $\tau_i$ – Time required to perform activity i on a work package
- $\tau_{na}$ - Time required to release a work package for initial completion
- $R_i$ - Rate of activity i allowed by resources
- $R'_i$- Maximum Rate of activity i allowed by resources
- $R_T$ – Total quantity of resources
- $f_{i*}$ - fraction of total resources applied to activity i
- $p_i$ – productivity of resources applied to activity i
- S - scope of work
- p(d) - probability discovering work requiring Iteration
- $\partial$ - concurrence relationship

**Concurrence Relationships**

Full concurrent:$w_c = 1$ (11a)

Partial concurrence: Equation (11b) approximates an "S" shaped concurrence graph with three lines

```
w_c =  IF (W_a /S)<alpha                                                  (11b)
    THEN (U1+(U2-U1)*X/alpha)
    ELSE (IF ((W_a /S) < beta)
        THEN (U2+(U3-U2)* ((Wa/S) - alpha)/(beta-alpha))
        ELSE (U3+(1-U3)*((Wa/S) - beta)/(1-beta)) ) )
```

where:

| | |
|---|---|
| alpha | Fraction of scope approved at which the rate of concurrence with respect to the fraction approved increases |
| beta | Fraction of scope approved at which the rate of concurrence with respect to the fraction approved decreases |
| U1 | Minimum concurrence (at start of project) |
| U2 | Concurrence at progress described by alpha |
| U3 | concurrence at progress described by beta |

For the simulations in the work the following values were used:

alpha = 0.2;  
Beta = 0.9  
U1 = 0.05  
U2=  0.25  
U3= 0.95;

Back to the Top