1999-12

The Impact of Goals on Software Project Management: An Experimental Investigation

Abdel-Hamid, Tarek K.; Sengupta, Kishore; Swett, Clint

MIS Quarterly, Volume 23, No. 4, pp. 531-555, December 1999
http://hdl.handle.net/10945/43630

MIS Quarterly
RESEARCH ARTICLE

THE IMPACT OF GOALS ON SOFTWARE PROJECT MANAGEMENT: AN EXPERIMENTAL INVESTIGATION[1]

By: **Tarek K. Abdel-Hamid**
Department of Systems Management
Naval Postgraduate School
Monterey, CA 93943
U.S.A.
tareka@worldnet.att.net

Kishore Sengupta
Department of Systems Management
Naval Postgraduate School
Monterey, CA 93943
U.S.A.
kishore@nps.navy.mil

Clint Swett
Department of Systems Management
Naval Postgraduate School
Monterey, CA 93943
U.S.A.

Abstract

Over the last three decades, a significant stream of research in organizational behavior has established the importance of goals in regulating human behavior. The precise degree of association between goals and action, however, remains an empirical question since people may, for example, make errors and/or lack the ability to attain their goals. This may be particularly true in dynamically complex task environments, such as the management of software development.

To date, goal setting research in the software engineering field has emphasized the development of tools to identify, structure, and measure software development goals. In contrast, there has been little microempirical analysis of how goals affect managerial decision behavior. The current study attempts to address this research problem. It investigated the impact of different project goals on software project planning and resource allocation decisions and, in turn, on project performance. The research question was explored through a role-playing project simulation game in which subjects played the role of software project managers. Two multigoal structures were tested, one for cost/schedule and the other quality/schedule. The cost/schedule group opted for smaller cost adjustments and was more willing to extend the project completion time. The quality/schedule group, on the other hand, acquired a larger staff level in the later stages of the project and allocated a higher percentage of the larger staff level to quality assurance. A cost/schedule goal led to lower cost, while a quality/schedule goal led to higher quality. These findings suggest that given specific software project goals, managers do make planning and resource allocation choices in such a way that will meet those goals. The implications of the results for project management practice and research are discussed.

Keywords: Goals, software project management, software cost, software quality

ISRL Categories: AC04

Introduction and Motivation

Managers widely rely on goal setting as a motivational technique to regulate task performance,

[1]Robert Zmud was the accepting senior editor for this paper.

for example, "as a component in some broad managerial system such as performance appraisal, quality control circles, or management by objectives" (Wood and Bailey 1985, p. 62). The basic assumption of goal-setting is the proposition, based on introspective evidence, that conscious human behavior is purposeful and is regulated by the individual's goals (Latham and Locke 1991).

Over the last three decades, a significant amount of research in the organizational behavior field has established the important and ubiquitous role that goals play in regulating human behavior. It has also demonstrated that the precise degree of association between goals and action is an empirical question (Locke et al. 1981). That is, no one-to-one correspondence between goals and action can be assumed. This is because people may, for example, make errors and/or lack the ability to attain their goals (Locke et al. 1981). This may be particularly true in dynamically complex task environments such as software development (Rasch and Tosi 1992).

The objective of the current study was to explore the impact of project goals on managerial decision making in software project environments. A laboratory experiment was conducted to investigate how different goals affect managerial planning and staffing decisions, and the impact on project outcome was assessed. The research question was explored in the context of a role-playing project simulation game in which experimental subjects played the role of software project managers. Two multigoal structures were tested, one for cost/schedule and the other quality/schedule.

We begin our discussion by considering the relevant background literature. Next, we present a set of research questions and describe our research setting. Finally, we present the results of our study and discuss the findings.

Conceptual Background

The domain of goal setting theory lies within the realm of purposefully directed action. Specifically, the theory focuses on the question of why some people perform better on work tasks than others. "The theory states that the simplest and most direct motivational explanation of why some people (with similar ability and knowledge) perform better than others is because they have different performance goals" (Latham and Locke 1991, p. 213). A goal is what an individual is trying to accomplish; it is the object or aim of an action.

The effects of goal setting on task performance have been studied on a wide range of tasks, including clerical tasks, prose learning, typing, supervision, and selling. The core finding is that the assignment of challenging, specific goals enhances performance more than no assignment of goals or instructions to "do your best." However, the magnitude of goal effects on performance appears to decrease as the cognitive complexity of the task increases (Locke and Latham 1990, p. 307).

Past research also shows that assigned goals influence performance through two types of mechanisms: those having a direct effect (effort, persistence, and directional attention) on an individual and those having an indirect effect (strategy development) (Earley et al. 1987). The mechanisms of effort, persistence, and direction of attention operate virtually automatically. "Individuals learn from an early age that, to achieve a goal, they must exert effort, persist over time, and pay attention to what they are doing and what they want to achieve" (Locke and Latham 1990, pp. 94–95). As tasks become more complex, these automatized mechanisms become progressively less adequate by themselves to ensure goal achievement, while the development of task-specific strategies becomes progressively more important. "For example, a manager given a specific goal related to increasing her or his department's profitability must develop strategies for increasing productivity and decreasing costs. It is insufficient to simply work harder or longer" (Gilliland and Landis 1992, p. 672).

> An important distinction between these two types of effects is that the direct influence is primarily motivational, that is to say, an allocation of the individual's energy-related resources to task performance, whereas the indirect effect is primarily cognitive, that is, the way in which a plan or strategy is developed to use the mobilized, energy-related resources (Earley et al. 1987, p. 107).

This distinction may explain why assigned goals have weaker effects on complex tasks than they do on simple tasks. In simple activities, dissatisfaction with substandard performance enhances effort and thereby contributes to the power of goal setting. "However, in complex tasks... the negative effect due to self-evaluations can undermine performance by deleteriously affecting required short-term memory functions, biasing recall of previously encoded information, or diverting attention from task-relevant thoughts" (Cervone et al. 1991, p. 264).

Recently, a number of researchers have taken the further step of investigating the boundaries beyond which goal setting will not work or may even be harmful. After conducting a number of laboratory experiments using a stock market prediction task, Earley et al. (1989, p. 25) concluded that the effects of challenging, specific goals on performance disappear or reverse for tasks in which

> (a) performance is primarily a function of strategy rather than of task effort; (b) there are many available strategies; (c) the optimal strategy is neither obvious nor readily identifiable; and (d) little opportunity to test hypotheses retrospectively exists (go back and retry a strategy).

The authors speculated that such tasks include a wide variety of professional and skilled work and are, therefore, both common and important. Software project management is such a complex task (Rasch and Tosi 1992).

Software Project Management

Throughout the short history of the software industry, researchers and practitioners alike have agonized over the industry's continuing failure to manage the complexity of software development and deliver quality software systems on time and on budget (Jones 1995). A major complicating factor has been the inability to estimate software development costs and schedules with acceptable accuracy and consistency. While numerous cost and schedule estimation models have been developed, their accuracy has proven inadequate (Thayer and Fairley 1994). This is caused primarily by the fact that most estimation tools are based on estimates of product size (e.g., lines of code, function points), which are extremely difficult to assess in the planning stage of a software project.

In addition to being key customer-oriented measures of project performance, cost and schedule estimates have a direct influence on project staffing (average staff size is determined by dividing the person-day cost estimate by the schedule). The accurate projection of required staff levels, in turn, has proven to be an absolutely critical function in software development. Overstaffing, for example, may lead to higher communication and coordination overheads, which translates into lower productivity. On the other hand, under-staffing often leads to project delays, volatile priorities, and inadequate testing (Lehder et al. 1988).

Once a project's total staffing requirement is determined (as a function of cost and schedule), it needs to be allocated between development and quality assurance (QA) activities such as testing, structured walkthroughs, inspections, etc. But again, as with initial estimation of product size and staff, the *initial* determination of the proportion of resources to allocate to QA is a difficult one because it must be done at a time when values of many of the factors driving software quality are unknown quantities. This applies both to product factors (e.g., product size and complexity), as well as to organization factors (e.g., staff skill and turnover rate).

The practical implication of the foregoing statements is that initial project plans (for cost, schedule, staffing, QA) do not necessarily constitute the best course of action to take during the project's life cycle; initial plans merely show what was thought of as best when the plans were made. Since actual events on a software project almost always differ from the assumed events that the plans were designed to meet, project managers must react continuously to real world events that actually occur, and not to those that might have occurred had the real world been kind enough to conform to the initial planning assumptions.

Indeed, attempting to conform to initial unrealistic estimates can be quite dysfunctional. For

example, when an initially undersized project falls behind schedule, project managers often resist adjusting the project's scheduled completion date, especially in the early stages of the lifecycle. According to DeMarco (1982, pp. 10–11), the reasons are political.

> There are a few different possible explanations for this effect: It's too early to show slip.... If I re-estimate now, I risk having to do it again later (and looking bad twice).... As you can see, all such reasons are political in nature.

So, rather than adjust an unrealistic schedule, managers often react to schedule delays by increasing staff size. Unfortunately, adding people late in a project can be quite disruptive.

> The people who are added must learn the system, and the people who teach them are the same people who were doing the work. While teaching, no work is done and the project falls further behind.... In addition to the time it takes to learn the system, more people increase the number of communication paths and the complexity of communication throughout a project (Pressman 1992, p. 104).

In Abdel-Hamid and Madnick (1991), the impact of staff additions on project completion time was analyzed in some detail. The results indicate that adding people late in the lifecycle lowers the average productivity of the project team because it increases the training and communication overheads. Lower average productivity translates into higher project cost. For a project's schedule to also suffer, though, the drop in productivity must be large enough to render an additional person's net cumulative contribution to (in effect) become negative. We need to calculate the *net* contribution because an additional person's contribution to useful project work must be balanced against the losses incurred as a result of diverting available experienced person-days from direct project work to training and communicating with the new staff member. And we need to calculate the *cumulative* contribution because while a new hire's net contribution might be negative initially, his/her productivity increases as training takes place: given enough experience on the project, s/he eventually starts contributing positively to the project. Only when the cumulative impact is negative will the addition of the new staff member translate into a longer project completion time. Thus, the earlier in the lifecycle that people are added and/or the shorter the training period needed (e.g., due to high quality of staffing pool or low complexity/novelty of project), the more likely that the net cumulative contribution will turn positive.[2]

In practice, assessing the net cumulative contributions of new staff is done on the basis of bare intuition, if at all. The reason for this is that most of the algorithmic resource models in use are *static* tools, unsuited for supporting the dynamic and iterative staff allocation process (Abdel-Hamid 1993a). For example, the COCOMO model's static staff/schedule trade-off relationship applies only to a project's *initial* estimates for total lifecycle cost and schedule (Boehm 1981). But as the above discussion suggests, the staff/schedule trade-off changes as the project proceeds through the lifecycle. That is, while it may be possible to trade X person-months for Y months early in the project, the cost (X person-months) will be higher in the middle of the project; and in the later stages of the lifecycle, it may be impossible to buy project time at any cost.

Unfortunately, human intuition is often unreliable in assessing the dynamic consequences of a managerial intervention such as adjusting staff level on a complex process such as software development. Not surprisingly, then, the results of many such interventions are "surprising" project difficulties (Brooks 1995).

Like many real-world managerial tasks, a further complication facing the software project manager is the pressure to satisfy simultaneously a variety of constituencies such as the users, the development team, and the maintenance organization. Each of these constituencies has its own goals with respect to the software product, e.g., many functions versus a low budget and no overruns, program efficiency versus ease of maintenance,

[2]Note that even if the net cumulative contribution of an added staff member is positive (implying no schedule penalty), the average productivity of the expanded team may still be lower (translating into cost penalty).

etc. These goals create fundamental conflicts when taken together. If the project team concentrates on any one of these, the others are likely to suffer (Boehm and Ross 1989). This led Boehm (1981, p. 205) to conclude that successful software engineering requires

> a continuous process of identifying goals, reconciling and making decisions with respect to conflicting goals, and managing with respect to several simultaneous goals.... Thus, it is extremely important to have a good understanding of the techniques of goal reconciliation and multiple goal decision making.

The Research Questions

This study was designed to investigate the impact of different project goals on project management decisions (planning and resource allocation) and project performance (cost, schedule, and quality). The objectives of the study were operationalized in the form of two research questions.

Project Planning and Resource Allocation

A number of earlier studies have demonstrated that software developers have very high achievement motivation (Cougar 1986; Weinberg and Schulman 1974). If you define good achievement in terms of what you want from the project, programmers will generally work very hard to give you what you asked for. For example, in the Weinberg and Schulman experiment, five teams were given the same programming assignment, but each team was given different directions about what to optimize while doing the job. Each team finished first (or, in one case, second) with respect to the objective they were asked to optimize, and none of the teams performed consistently well on all of the objectives.

Weinberg and Schulman's (1974, p. 76) major conclusions were:

> (1) Programming is such a complex activity that programmers have an almost infinite number of choices in terms of how they will write a program in order to meet certain specifications; and (2) if given specific objectives, programmers can make programming choices in such a way that they will meet those objectives—provided they do not conflict with other specific objectives.

While the focus of the current study on *managerial* planning and resource allocation tasks differs from Weinberg and Schulman's experimental focus on programming, the two are interdependent. Indeed, in the Weinberg and Schulman study, the programmers adjusted their estimates, depending on what goals were assigned, to give themselves more "cushion" for meeting assigned goals.

These results are consistent with multigoal study findings in other complex domains. For example, Locke and Bryan (1969) found that people often need to focus on a subset of the task's attributes or elements. Further, "a person's goals affect...*what* aspects of a task he will focus on, i.e., the relevant amount of attention he will pay to different task parameters" (Locke and Bryan 1969, p. 35).

These findings lead us to ask the following question,

> Do differences in goal sets result in different planning and staff allocation strategies in the management of software projects?

Project Performance

Software project performance is typically defined in terms of three main outcome factors: cost, duration, and quality.

> [T]he the challenge of software development is represented as the "push-pull" attempt to control and manage the three major outcome factors: cost, schedule, and quality.... [These] outcome factors are viewed as "springs" that are dynamically compressed or decompressed during the life of a project (Krasner 1994, p. 5).

Prior research indicates that project cost/schedule estimates and staff allocation decisions do have a significant impact on all three performance dimensions (Abdel-Hamid and Madnick 1991; Humphrey 1989; Lehder et al. 1988). For example, a project's schedule estimate directly

influences work force hiring and firing. A tighter schedule leads to a higher work force and vice versa. The work force level acquired determines the communication and training overheads incurred on the project that, in turn, affects the team's productivity and thus, the ultimate cost of the project (Jeffery 1987).

Project schedules can also affect product quality. As DeMarco (1982) points out, a tight schedule leads to schedule pressures, and people under time pressure do not necessarily work better, they just work faster. As a result, the quality of the software delivered is often the first casualty in the struggle to deliver any software at all.

The possibility that goals affect project planning and staff allocation raises the follow-up question:

> Do differences in project planning and staff allocation strategies result in different levels of project performance (defined in terms of cost, schedule, and quality)?

Method

Task Environment

The research questions were explored in the context of a simulation game in which subjects played the role of project managers running the programming phase of a software project. The simulator is based on a systems dynamics model of the software development process, developed on the basis of a study of software development practices in five organizations, and supplemented by an extensive database of empirical findings from the literature. The simulation game has been used in a number of recently published studies (e.g., Abdel-Hamid et al. 1993, 1994; Sengupta and Abdel-Hamid 1993; 1996). Apart from the overview in the Appendix, the model is completely described in Abdel-Hamid and Madnick (1991).

The simulated project used in the experiment was a real project conducted to develop a software system for processing satellite telemetry data. The initial estimates of the project's size, cost, and duration were 15,860 Delivered Source Instructions (DSI), 944 person-days, and 272 working days respectively. Additional new requirements were "discovered" during the programming phase, causing the system's size to expand to 24,400 DSI.

At the start of the programming phase (day zero), subjects had to make two decisions, namely, the total staff level desired and the percent of staff to allocate to quality assurance (QA) activities. The software ran the simulation for 40-day intervals (two calendar months). At the end of each simulated 40-day interval, the simulation provided status reports and graphs on resources used and work accomplished. From a menu screen such as shown in Figure 1, the subjects could select any of six status reports/graphs to browse and view each for as long as they wished. They could also access a report/graph more than once. (Example status reports and graphs are shown in Figures 2, 3, and 4.) As project managers, the subjects could then update cost and schedule estimates, increase (or decrease) total staff level, and change the percent of staff allocated to QA (i.e., make a total of four decisions per period). The simulation would then proceed to simulate the next 40 days. The process continued until the programming phase was completed.

As in the real project, the hiring of new employees was not instantaneous; there was an average hiring delay of eight weeks (40 working days). New hires passed through an assimilation period of four months, on average, in which the new employee was trained and brought up to speed. During this assimilation period, a new employee was only half as productive as an experienced employee. Furthermore, since new hires were trained by experienced employees, the productivity of experienced employees went down as well. The personnel turnover rate in the project was 30% per year. The costs associated with turnover were implicit in the sense that any temporary decrease in staffing levels would mean hiring at a later date, with all its attendant costs.

Two sets of factors affect the defect generation rate in the model. A nominal defect generation variable captures the overall organizational setting (e.g., an organization's use of structured techniques, the overall quality of the staff, etc.) and project type (e.g., project complexity, system size, programming language, etc.). In addition, the model captures a second set of factors that

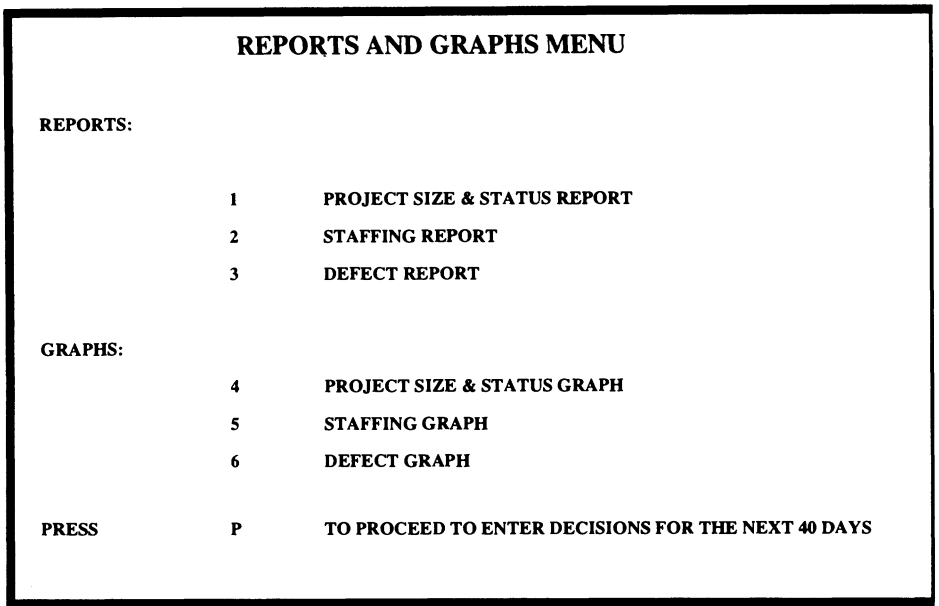


Figure 1. Menu Screen

affect defect generation rates dynamically during the life of a project. These include the work force-mix (newly hired versus experienced) and schedule pressures. It is assumed in the model that a newly hired employee is, on the average, twice as defect-prone as an experienced employee would be. Further, it is assumed that as schedule pressure increases, the defect generation rate also increases (by as much as 50%).

Defects remain undetected until the defective software is reviewed and tested, at which point some of the defects do get detected, and those are then reworked. Not all defects are necessarily detected, though some will "escape" and pass undetected into the subsequent phases of software development, where they might then be caught, albeit at a higher cost.

The number of defects detected is a function of how much QA effort is allocated and the average QA effort needed to detect a defect. The latter is a function of organizational-type factors such as the overall quality of the staff, as well as project-specific-type factors such as project complexity and programming language. In addition, the model captures the impact of two dynamic factors: work efficiency and defect density. For example, it is assumed in the model that as QA activities are performed, the more obvious defects are detected first. As these are detected, it then becomes increasingly expensive to uncover the remaining defects, which are more subtle (albeit less pervasive).

Experimental Design

The experiment had one independent variable: *project goal*. Participants were randomly assigned to one of two goal conditions:

- Goal Set 1: minimize overruns in both cost and schedule.
- Goal Set 2: deliver a quality product (i.e., detect as many defects as possible) and minimize any schedule overrun.

Notice that the goal sets were multidimensional, as would be typical in real-life managerial tasks (Gilliland and Landis 1992). It would have been too simplistic to set one-dimensional goals such as, say, minimizing cost overrun. This could have led some subjects to staff the project with a single person to minimize communication and coordination overheads, thereby decreasing cost, but at the expense of an enormous schedule overrun. While such behavior could in some sense be considered "correct," it, however,

PROJECT STATUS REPORT (Time: 160)

INITIAL ESTIMATES: (These will not change during the project)		
System size	15,860	DSI
Programming cost	944	Person-days
Programming phase duration	272	Days
UPDATED ESTIMATES		
New estimate of system size	19,313	DSI
Your last estimate of cost	1,291	Person-days
Your last estimate of duration (start-end)	285	Days
Time remaining	280	Days
REPORTED PROGRESS		
% DSI reported complete	28.49	Percent
Total DSI Reported complete	5,503	DSI
Total person-days expended	598	Person-days
Reported productivity	9	DSI/person-day

PRESS <ENTER> TO RETURN TO MENU

A. Status Report

STAFFING REPORT (Time: 160)

Current Total Staff Size	4.5	People
Staff Allocated to Programming	3.3	People
Staff Allocated to QA	1.2	People
Percentage of workforce that is experienced	77	Percent

PRESS <ENTER> TO RETURN TO MENU

B. Staffing Report

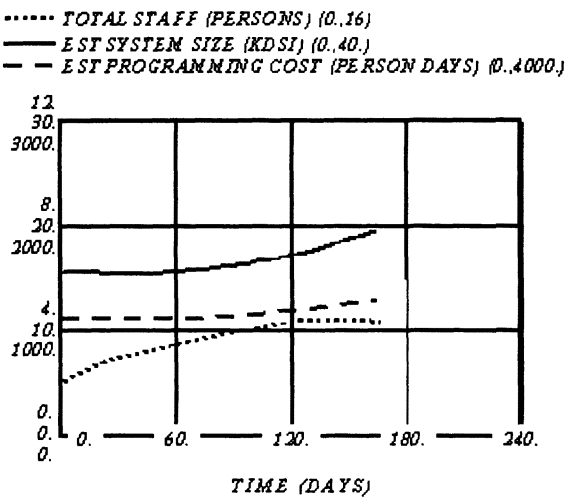
DEFECT REPORT (Time: 160)

Cumulative status from start of programming to day	===>	160
Total person-days expended to date	598	Person-days
Programming person-days expended to date	423	Person-days
QA person-days expended to date	175	Person-days
Total defects detected	132	Defects
Total KDSI completed	5.5	KDSI
Defect density	24.0	Defects/KDSI
Statistics for the last 40-day period only		
QA person-days expended in the last 40 days	48	Person-days
Defects detected in the last 40 days	38	Defects
Density of defects detected in the last 40 days	22.8	Defects/KDSI

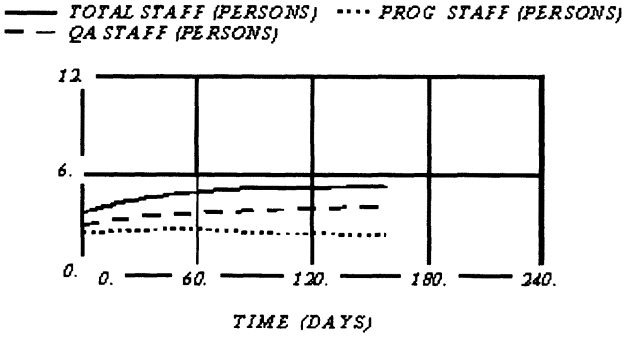
PRESS <ENTER> TO RETURN TO MENU

C. Defect Report

Figure 2. Project Status Reports Available to Subjects



A. Project Size and Status



B. Staffing

Figure 3. Project Status and Staffing

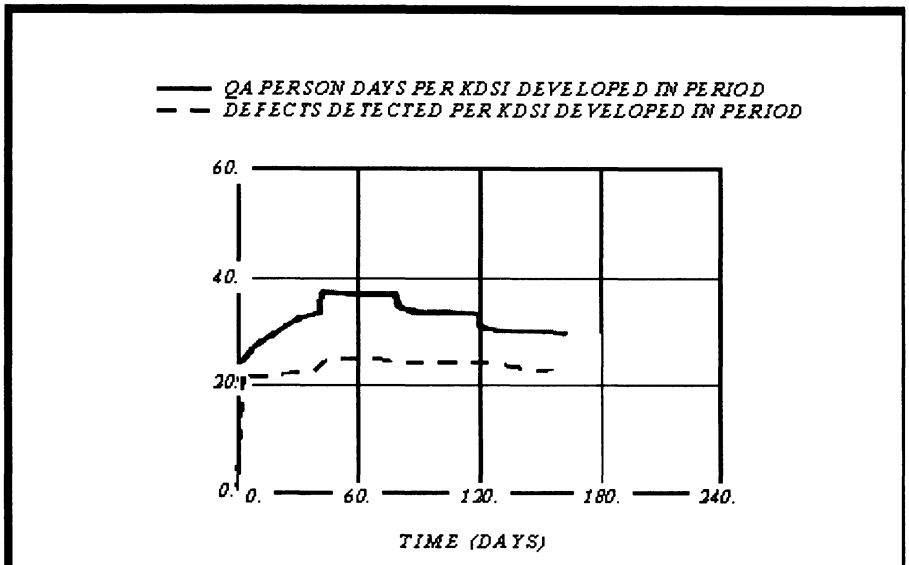


Figure 4. Defect Graph

would be quite unrealistic. In reality, a missed schedule is often quite costly, for it can "reduce market impact, create dissatisfied customers, and raise internal costs by creating additional problems during system integration" (Pressman 1993, p. 275). As a result, most organizations place a premium on limiting schedule overruns. Minimizing schedule overrun was, thus, incorporated in both goal conditions.

We also note that quality is narrowly defined as the number of defects per DSI, which is a commonly used metric for quality in practice, because it is easily and objectively measured (Grady 1992). In theory, however, it is a limiting definition in that it excludes product characteristics such as complexity, understandability, portability, etc.

The experiment's instructions indicated that the project's cost and schedule estimates (944 person-days and 272 days respectively) were derived from an off-the-shelf estimation tool that was recently acquired by the organization (as was the case in the real project). The subjects were also told that the defect density (i.e., number of defects detected during programming divided by the number of KDSI developed) historically ranged between five and 20 defects/KDSI.

Experimental Setting and Participants

We first conducted a pilot study with seven subjects. The purpose of the pilot was to ensure that the software worked as intended and that the instructions were understood by the subjects. On the basis of feedback from the pilot subjects, the instructions on the goal sets were refined.

We planned to conduct the experiment with 26 subjects. On the day of the experiment, one subject fell sick and could not participate. The experiment was thus conducted with 25 subjects: 13 in the cost/schedule goal condition, and 12 in the quality/schedule goal condition.

The subjects were second-year masters' students in an information technology management curriculum at a U.S. university. All subjects were mid-level managers and had an average of 12.6 years work experience. The experiment was part of a course in software engineering. The subjects were told that 20% of their course grade would be determined by their performance in the simulation.

In order to prepare the subjects for the simulation, a one-hour classroom lecture was given to explain and demonstrate the computer interface and explain the task. After this session, all sub-

jects spent a second one-hour session period playing the game (using a practice project). No goals were administered to the subjects for the practice session.

The experiment was conducted two days later in a computer lab. Subjects took between 90 and 120 minutes to complete the experiment. After the experiment, subjects filled up a debriefing questionnaire. The questionnaire provided subjects with an opportunity to express views regarding the simulation. It also contained questions on demographics, the specifics of the subjects' project management experience (if any), the ease of use of the simulator, and the understandability of the instructions.

Dependent Measures

Project performance was operationalized through three dependent measures: final *cost* (in person-days), *completion time* (in days), and remaining undetected *defects* at the end of the programming phase (in number of defects). The number of remaining undetected *defects* indicated the quality of the software product, i.e., fewer remaining undetected defects indicating higher quality software, and vice versa.

In addition, the subjects' four decision variables—*desired total staff level* (in people), *percent of staff allocated to QA* (percent), and the *updated cost and schedule* estimates (in person-days and days respectively)—were captured at every 40-day interval. Following the suggestion in Wofford et al. (1992), the *updated cost and schedule* estimates were operationalized as the delta between the subject's latest estimates and the project's initial estimates.

Experimental Results and Statistical Analysis

The project's initial cost and schedule were underestimated due, in part, to an initial underestimation of product size. The upward adjustments to project size were, of course, visible to the subjects through the status reports (see Figure 2). In addition, the experimental subjects could also compare reported data on staff productivity, defect rates, etc., with the initial project assumptions and continuously adjust their project plans/decisions, as real-life project managers often do. The first research question we sought to address was whether goal-setting influences the manner in which this dynamic project planning and resource allocation process unfolds.

Impact of Goals on Project Planning and Resource Allocation

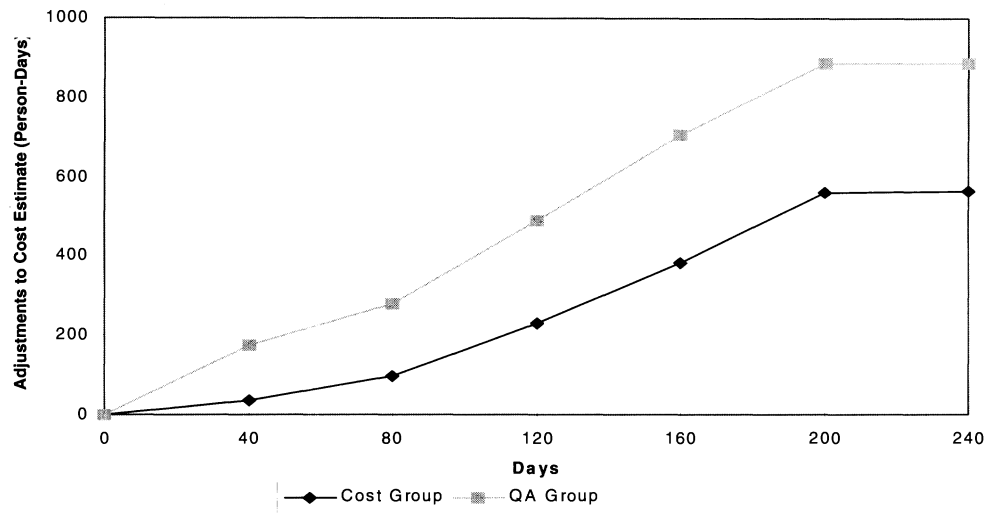
Adjustments to the project's cost and schedule as well as the two staffing decisions were analyzed using a multivariate analysis of variance model for repeated measures (Winer 1971). Because some of the subjects completed their projects as early as day 240, while others finished it later, there were missing values from interval eight (day 280) onward. Therefore, repeated measures analyses were conducted through period seven (day 240). (From day 0 through 240 for the two staffing decisions, and from day 40 through 240 for the cost and schedule adjustments.) Since the number of subjects in the two groups was not equal, the analysis was conducted with the General Linear Models procedure in SAS (SAS 1987).[3]

Adjustments to Project Cost Estimate

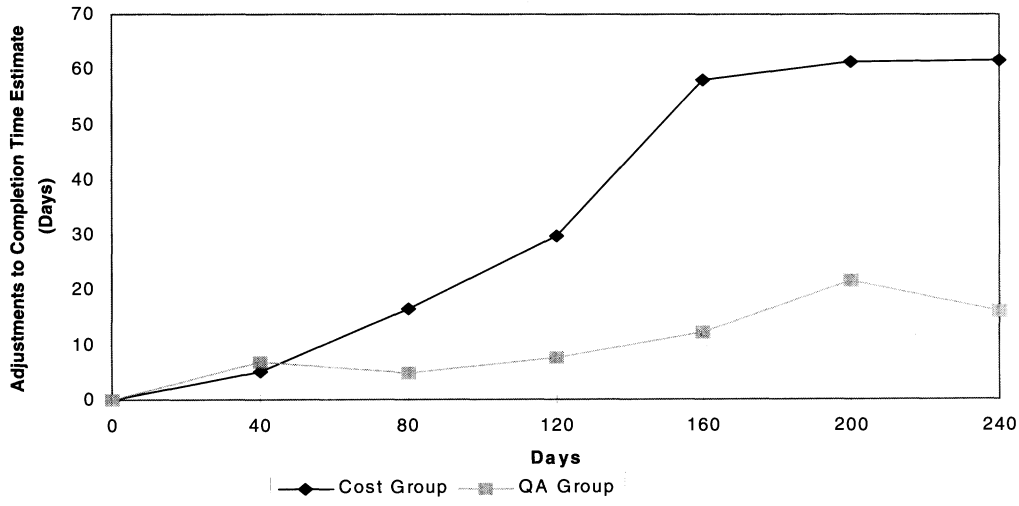
The groups' average adjustments (deltas) to the project's cost estimate are compared in Figure 5a. The plots indicate that the cost group made smaller upward adjustments to the project's cost estimate. This is confirmed by the MANOVA results, which show a significant between subjects effect ($F(1, 23) = 5.47$; $p < 0.0293$). The time effect was also significant ($F(5, 19) = 18.1$; $p < 0.0001$), indicating that the pattern of cost updates changed over time. The time*group interaction,[4] however, was not significant ($F(5, 19) = 0.893$). Thus, the picture that emerges from this analysis is that the cost adjustment strategies of both goal groups were qualitatively similar (similar shape), but that quantitatively, the cost group opted for smaller cost adjustments.

[3]We also analyzed the data after dropping a subject at random from the cost/schedule group to create two groups of equal sizes (12 subjects each). None of the results changed.

[4]Time*group interaction tests for the possibility that different experimental conditions (i.e., different goal sets here) may behave differently over time for a particular dependent variable.



A. Adjustments to Costs



B. Adjustments to Estimates of Completion Time

Figure 5. Planning Decisions

Adjustments to Project Completion Time Estimate

Figure 5b shows the group means for the upward adjustments (deltas) to the project's completion time estimate. The between-subjects effect was significant ($F(1,23) = 3.98$; $p < 0.0581$). The time effect was also significant ($F(5,19) = 5.3785$; $p < 0.003$), indicating that the completion time adjustments changed over time (they are not horizontal). However, the time*group interaction was not significant ($F(5,19) = 1.41$).

These results, therefore, suggest that the schedule adjustment strategies of both groups were qualitatively similar (similar shape), but that quantitatively, the cost group was more willing to extend the project's completion time. While minimizing schedule overruns was a common goal to both groups, the cost group's greater willingness to extend the project's completion time may be attributed to the widely touted "lesson" that tighter schedules often lead to higher project costs (Boehm 1981; Brooks 1995; Londeix 1987).

Total Staff

The between subjects analysis indicates that the staffing levels were not significantly different ($F(1,23) = 2.84$). This is due to the fact that the two groups' staffing decisions were initially quite close, as is shown in Figure 6a. However, notice that after day 80 they do diverge, suggesting a possible time lag effect, i.e., the lag between initiating an action and its completion. Wood and Locke (1990) had found that such time lag effects could develop on complex tasks:

> The lag in the effect of goals may be caused by the fact that effort does not pay off right away on complex tasks.... The task strategies which are developed in response to the goals take time to formulate, to master and to affect outcome measures. Thus, studies of performance on moderately or extremely complex tasks should look for time lag effects (Wood and Locke 1990, p. 99).

This, indeed, appears to be the case in software project staffing. For example, when a decision to add staff is made, people do not become available immediately because of delays in the staff acquisition/hiring process. Furthermore, when the new staff are finally acquired, there are often additional assimilation delays before they become fully productive.

When we conducted the analysis from period three (day 80) onward, we found a significant group effect for total staff ($F(1,23) = 5.27$; $p < 0.032$). The time effect was also significant ($F(4,20) = 3.21$; $p < 0.035$), but not the time*group interaction ($F(4,20) = 1.23$). This suggests that the effect of goals on the staffing level decision is time lagged. That is, after day 80, the cost group opted for significantly lower total staff levels. Because a project's average staffing level is typically determined by dividing the estimated cost in person-days by the estimated completion time in days (Boehm 1981), the staffing results are consistent with the group's overall persistence in maintaining a lower cost estimate while being willing to incur a larger schedule overrun.

Staff Allocation to Quality Assurance

Figure 6b shows that the QA group allocated a higher percentage of their staff to quality assurance in all intervals. This is confirmed by the MANOVA results. The between subjects effect is highly significant ($F(1,23) = 39.83$; $p < 0.0001$). However, both the time effect ($F(6,18) = 1.85$) and the time*group interaction effect ($F(6,18) = 1.00$) were not significant, indicating that the lines in Figure 6b do not change significantly over time.

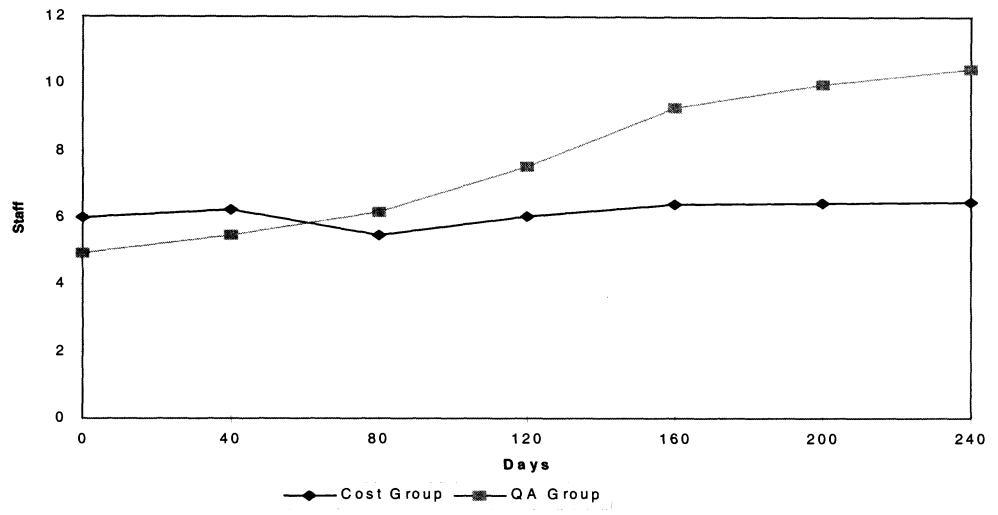
It is important to note that the QA group's willingness to allocate more staff resources to QA is, in fact, more impressive than what the differences in the percentage of staff allocated to QA that are depicted in Figure 6b would suggest. By taking the two groups' staff size decisions into consideration, we find that the QA group allocated a larger percentage of a larger total staff level to QA. The cumulative effect of these two decisions makes the 412 person-days expended (on average) by the QA group on quality assurance to be more than double the 173 person-days expended by the cost group ($F(1,23) = 28.51$; $p < 0.0001$).

Summary of Results

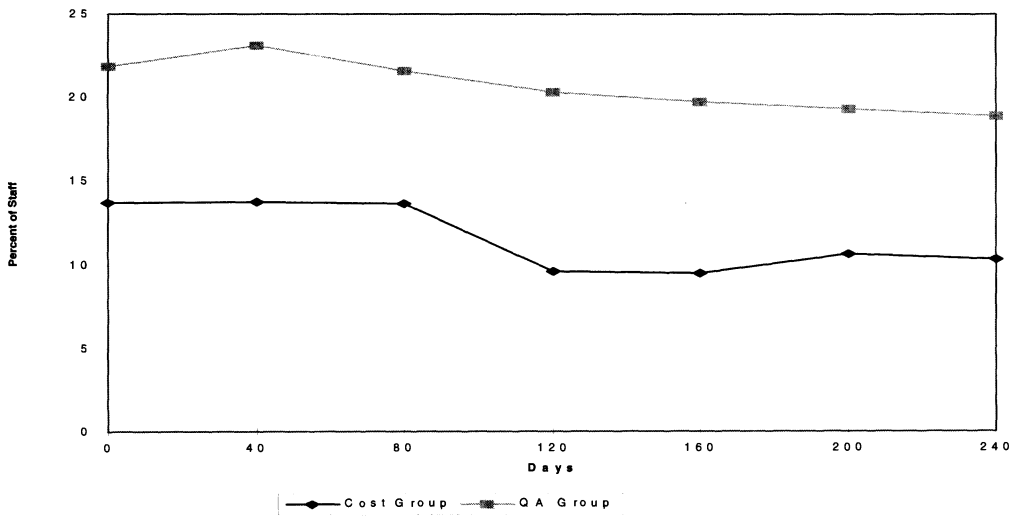
The above results reveal that goals do influence project planning and resource allocation decisions on software projects. Specifically, the cost group opted for smaller cost adjustments and was more willing to extend the project completion time (weak support). On the other hand, the quality group acquired a larger staff level in the later stages of the project, and allocated a larger percentage of the larger staff level to quality assurance.

Impact of Goals on Project Outcome

Given that goals were found to influence project planning and staff allocation, we sought next to investigate whether goals affected project outcome in terms of cost, duration, and quality.



A. Staffing Decisions



B. Allocation of Staff to Quality Assurance

Figure 6. Resource Allocation Decisions

Table 1 shows summary statistics (means and standard deviations) for the groups' final cost, completion time, and remaining defects. A MANOVA shows that the two goal conditions were significantly different ($F(4,18) = 9.0993$; $p < 0.0003$). The univariate analysis of variance results indicate that the cost group had a significantly lower cost ($F(1,23) = 16.39$; $p < 0.0005$) and a significantly higher number of remaining defects ($F(1,23) = 12.81$; $p < 0.0016$). The differ-

Table 1. Means and Standard Deviations for Project Outcome Variables

	Cost {Person-Days} Mean (Std. Dev.)	Completion Time {Days} Mean (Std. Dev.)	Defects {Defects} Mean (Std. Dev.)
Goal (1): Cost/Schedule	1500 (165)	297 (45)	1,591 (805)
Goal (2) Quality/Schedule	1963 (375)	319 (55)	742 (166)

ence in project completion time between the two groups, however, was not significant ($F(1,23) = 1.28$).

These findings suggest that given specific software project goals, managers do make planning and resource allocation choices in such a way that will meet those goals. Specifically, a cost/schedule goal led to lower cost, while a quality/schedule goal led to higher quality.

These results are consistent with multigoal study findings in other complex domains. For example, Locke and Bryan found that, in complex tasks, people often need to focus on a subset of the task's attributes or elements. Further, since a person's goals affect the relevant amount of attention s/he will pay to different task parameters (Locke and Bryan 1969), this often leads to poorer performance on aspects of the task that are not relevant to the goals. "Goals, in effect, give the individual 'tunnel vision'. This can be advantageous if one wants to stay in the tunnel, but it may not be if other outcomes are desired as well" (Locke and Latham 1990, p. 95).

Discussion/Conclusion

Limitations

The generalizability of the findings is constrained by three potential limitations. First, the experimental setting was computer-based laboratory experimentation. While laboratory research naturally entails giving up the richness of context to obtain control, it should not be automatically assumed to be non-generalizable. A review of the organizational behavior/personnel literature has found remarkable similarities between research findings obtained in laboratory and field settings (Locke 1986).

Simulation-based laboratory experimentation is particularly advantageous in the study of complex dynamic decision-making environments such as software project management. In such environments, much of the complexity and dynamism in tasks is a result of the flow of information, action, and outcomes during the performance of the task, making the complete preprogramming of stimuli in a laboratory setting impossible. On the other hand,

> games involving experimental simulations can capture some of the dynamic intertemporal aspects of tasks which are characteristic of natural work settings but absent from the preprogrammed stimuli in a controlled laboratory experiment. It is the temporal interdependencies between stimuli, actions and outcomes that make a research game an ideal method... in the study of goal effects" (Wood and Bailey 1985, p. 67).

Second, the use of students as subjects raises the question of the generalizability of the results. Studies examining the use of students in decision making studies have generally concluded that the formal properties of a task are much more important determinants of decision making than subject profiles (Ashton and Kramer 1980; Brehmer and Brehmer 1988). In this study, the principal difference between the subjects and actual project managers was in their epistemic competence (cf. Brehmer 1992). The question, then, is whether experience in a task substantively alters the decision-making processes of a sub-

ject. Evidence from the clinical and medical domains (Camerer and Johnson 1991; Garb 1989) shows this not to be the case. In the domain of managerial tasks, Remus (1986) found no significant differences between students and managers in making production scheduling decisions. Although software project management decisions are somewhat different from production scheduling decisions, they are similar enough to apply his findings and assume that software engineering graduates are acceptable surrogates in this experimental investigation.

A third potentially limiting factor concerns the nature of the experimental task. It is not claimed that the above results generalize to all types of project situations. In this experiment, the simulated project was a medium-sized organic project, i.e., a project developed in a familiar in-house environment in which the team has extensive experience in working with related systems within the organization (Boehm 1981).

Key Findings

Like many real-world managerial tasks, software project management is a dynamic process involving repeated cycles of planning, task performance, time delays, and feedback. Because actual events on a software project almost always differ from the assumed events that project plans were designed to meet, software project managers must continuously adjust their staffing levels, estimates, resources, etc. to an evolving project and organizational environment. Our experimental results suggest that project goals can have a significant impact on how this dynamic and adaptive project planning and control process unfolds. And this, in turn, can significantly influence final project outcome.

The choice differences between the two goal groups were manifest, for example, in how they played the cost/schedule trade-off. With an explicit project goal to minimize cost overruns, the cost group demonstrated greater commitment to the initial cost estimate by exhibiting less willingness to change/abandon the original cost estimate, while it was more willing to extend the project's completion time. As explained earlier, this may be attributed to the widely shared view that tighter schedules often lead to higher project costs. To achieve a lower project cost at the expense of longer project duration, the cost subjects maintained smaller and more stable staff levels.

Conversely, the QA group subjects exhibited greater unwillingness to change/abandon the original completion time target, while they were more willing to raise the project's cost. As a result, they tended to acquire larger staff sizes, and to continue hiring until late into the lifecycle. (This, incidentally, is quite similar to what actually happened on the real project.) This led to higher cost. On the other hand, by allocating a higher fraction of a larger staffing level to quality assurance activities, the QA group successfully achieved higher quality levels, as evidenced by the group's fewer latent defects at the end of development (742 on average, versus 1,591 for the cost group). The larger number of remaining defects observed in the cost group's performance can be attributed to two factors. First, a lower allocation of person-days to quality assurance led to the detection of fewer defects. Second, the larger number of defects that escaped detection did *not* remain stable. Instead, these undetected defects multiplied and propagated through succeeding development tasks (cf. Pressman 1992). The earlier in the lifecycle a defect is introduced, the more generations of defects it will reproduce, and thus the more costly it will end up being. That is why,

> The value of early defect detection and correction cannot be overemphasized. Statistics gathered on software projects from several major corporations establish that the cost of defect correction increases dramatically the further from their source defects are detected" (Ebenau and Strauss 1994, p. 21).

Thus, the cost group's policy of persistently under-investing in quality assurance (relative to the QA group) not only led to a lower defect detection rate (15% on average, versus 37% for the QA group) but, in addition, instigated the generation of more total defects (1,879 on average, versus 1,170 defects for the QA group). The compounding impact on quality is depicted in Figure 7: the cost group's undetected defects at the end of development (1,591) was more than double that of the QA group (742).

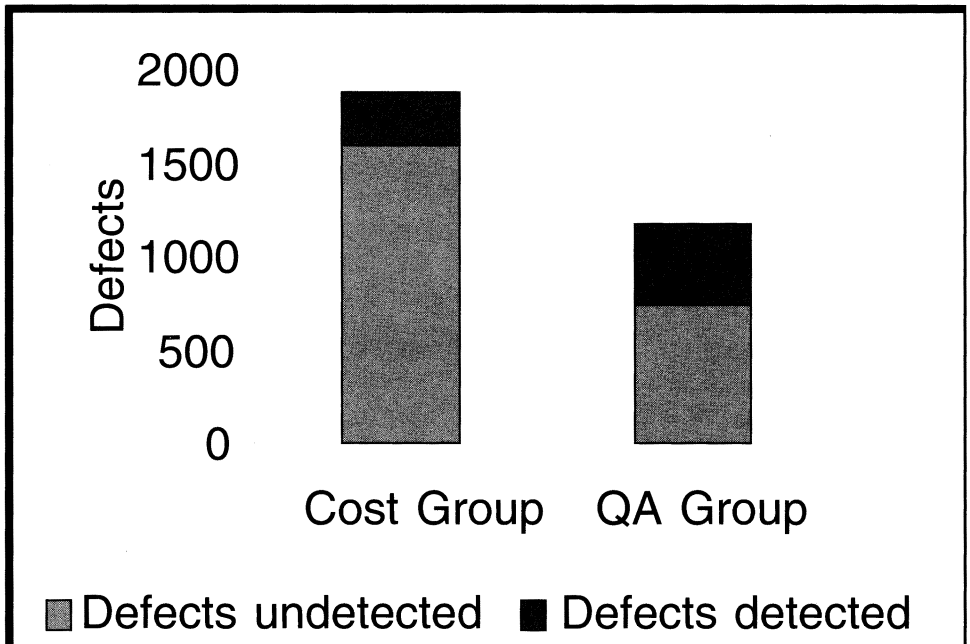


Figure 7. Detected Versus Undetected Defects

In summary, our experimental results demonstrate that in software development *different* project goals do, in a real sense, create *different* projects. This is because, like most dynamic systems, the software development environment is a *reactive* system that changes over time both autonomously as well as in reaction to a manager's decisions. Decision making in such reactive environments is like chasing a target that not only moves but also reacts to the actions of the pursuer (Sengupta and Abdel-Hamid 1996). In the experiment's project environment, for example, the project's size increased autonomously and similarly for all subjects, while the defect rate was substantially impacted by the different staff allocation decisions adopted under the different goal scenarios.

Implications for Managing Software Projects

The results of the study point to two specific implications for practice. First, the difference in how the two goal groups acted adds weight to the argument to set up quality assurance as an organization independent from the development organization. It has been argued in the software engineering literature that such organizational independence is necessary to maintain an adequate check and balance on development activities (Bryan and Siegel 1988; Myers 1979). Our results suggest that this separation may also be useful to insulate QA from the impacts of cost and schedule pressures. As we found in our laboratory experiment, quality assurance activities were relaxed when cost pressure mounted. It is not difficult to see why this might indeed occur on real-life projects. Since the objective of quality assurance activity is to detect defects, and since undetected defects are by their very nature invisible, it is almost impossible to tell whether or not an adequate job was done in quality assurance (except much later in the lifecycle). Under such circumstances, it is easy to rationalize both to oneself and to management that the quality assurance job that was "convenient" to do was not insufficient. Further, because line managers under cost/schedule pressures are not likely to listen sympathetically to the QA group's reports of inadequate test plans, human factors prob-

lems, or documentation errors, we further recommend that the independent QA group report to a management level higher than line management. This way, the quality assurance activity would have a better chance of influencing priorities and obtaining/preserving the resources needed to maintain quality control.

Second, given the problematic state of software project estimation, the initial estimates for a project are typically flawed. Therefore, conformance to initial project estimates should *not* be the sole standard against which managerial performance is measured. Rather than drive managers to stubbornly pursue possibly unrealistic project goals, they should, instead, be asked to formulate explicit project goals and be judged on how effective their decisions are in achieving these goals.

Implications for Research

The core finding of the study—that goals do matter in software projects—leads us to propose three issues that could be addressed in future research: what are the goals that should be set for effective management of software projects, how the goals should be set, and what types of information should be provided to decision makers.

What Are the Goals That Should be Set?

The explicit goals that guide a project should be decoupled from the (unreliable) initial estimates. Instead, goals should be set with a view to affecting the *strategy* that the manager chooses to follow (Chesney and Locke 1991). In practical terms, this entails setting the appropriate behavioral metric to guide the manager's decisions. It would be a worthwhile topic for future research to identify goals that can help shape strategy and thus free managers from having to persevere with minimizing overruns from the initial cost estimate.

A plausible candidate for investigation would be a goal to *maximize productivity*.[5] When provided with such a goal, a manager would no longer have to adhere to a cost estimate that was unreliable to begin with. Instead, s/he could then select staffing strategies that are more appropriate for managing software projects, such as seeking to attain high productivity as soon as possible and sustain it across the life of the project. Furthermore, as Chesney and Locke (1991, p. 420) suggest, "high levels of appropriate strategies used tend to moderate the goal-performance relationship." Therefore, the selection of suitable strategies through setting a goal of maximizing productivity holds the promise of improving overall performance on the project.

How Should Goals Be Set?

Goal commitment refers to one's determination to reach a goal.

> It is virtually axiomatic that a goal that a person is not *really* trying for is not *really* a goal and therefore cannot have much effect on subsequent action. Only an individual who is genuinely trying for a goal can be described as being committed to that goal (Locke and Latham 1990, p. 124).

Thus, there is a need to attain an understanding of the factors that affect goal commitment in software project management.

One factor that can affect goal commitment is the manner in which goals are set. Generally speaking, goals can be assigned to an individual, or set participatively (Latham et al. 1988). The use of participation is posited to engender greater commitment to the goals (Erez et al. 1985; Locke et al. 1988). Moreover, in complex tasks, a higher degree of goal commitment is associated with superior performance (Erez et al. 1985; Locke et al. 1988). Future research can usefully shed light on these questions in the context of software projects, i.e., (1) whether a software project manager is more likely to be committed to a goal when s/he has a voice in setting it, and (2) whether a higher degree of goal commitment is indeed associated with superior performance in software projects.

What Feedback Should Be Provided?

In dynamic tasks, feedback plays a crucial role in affecting decision strategies as well as performance (Sengupta and Abdel-Hamid 1993). Thus, another implication of the current study concerns the role of feedback in goal setting and performance; more specifically, whether the type of feedback (outcome or process feedback) provided affects the goal-performance relationship (cf. Earley et al. 1990). For feedback to be effective in

[5]Productivity in software projects can be measured as the number of DSI produced divided by the number of person-days expended.

an individual's behavior, it should identify not only the need to adjust action, but also provide specific information concerning *how* to adjust. Such adjustment information is particularly important in performing complex or unstructured tasks in which the relation of behaviors to performance outcomes may be uncertain (Campbell 1988). While outcome feedback does provide information on the need to adjust, it does not contain information suggesting *what* should be adjusted, i.e., it lacks strategy-shaping properties. This may often result in inappropriate adjustments on the part of the decision maker (Earley et al. 1990). For effective performance in complex environments, subjects also need access to process feedback (such as cognitive feedback). Thus, a profitable avenue for future research would be to examine the relationship of goals, type(s) of feedback, and performance in managing software projects.

Conclusion

> It would be convenient if we could set up a single overall goal for software engineering such that, if we satisfied this goal, we would satisfy all of the...challenges in the process.... Unfortunately—or, on second thought, fortunately—the world of software engineering is not so simple (Boehm 1981, pp. 20–21).

To date, goal-setting research in the software engineering field has tended to focus on the development of practical tools to support the processes of identifying software development goals and reconciling/managing with respect to several simultaneous project goals. For example, Boehm proposed GOALS (Goal-Oriented Approach to Life-cycle Software) to hierarchically structure software goals for human factors, resource engineering, and program engineering, and use it as a form of *management by objectives* to guide the specification, development, and maintenance of software development (Boehm 1981, p. 23). In another effort, Basili (1994) devised the Goal-Question-Metric (GQM) paradigm for guiding measurement activities in software quality improvement efforts. It involves defining goals, constructing a comprehensive set of questions to help answer whether the goals are being achieved, and defining a set of quantitative measures to answer the questions objectively.

On the other hand, there has been a serious lack of microempirical analysis of the impact of software project goals on managerial decision making behavior. Our study sought to address this research problem. This work also contributes to the heretofore limited number of goal setting studies involving dynamically complex tasks. Most prior research on goal setting has emphasized simple, single trial, single goal tasks that display high levels of stability (Cervone et al. 1991). In contrast, the software project planning and resource allocation task, like many real-world managerial tasks, is a complex dynamic process involving repeated cycles of planning, task performance, time delays, and feedback. Furthermore, the present study's focus on the planning and resource allocation task is worth noting:

> Despite its importance, resource-allocation behavior is not nearly as well understood as choice behavior because there has been almost no research on this topic. There is a huge body of research on choice behavior, where subjects make a selection from two or more already defined alternatives, events, or lotteries. Choice decisions are common in real life, but so are resource-allocation decisions (Langholtz et al. 1994, p. 28).

Our experimental results suggest that project goals can have a significant impact on the dynamic and adaptive project planning and control process, and, in turn, on project outcome. Specifically, a goal to minimize cost and schedule overruns led to lower project cost, while a quality/schedule goal led to higher product quality. Neither group, thus, did better on all three project performance dimensions (cost, duration, and quality). The results have significant implications for the management of software projects, as well as for future research on goal setting in software projects.

References

Abdel-Hamid, T. K. "Adapting, Correcting, and Perfecting Software Estimates: A maintenance Metaphor," *Computer,* March 1993a.

Abdel-Hamid, T. K. "A Multiproject Perspective of Single-Project Dynamics," *Journal of Systems Software* (22:3), September 1993b, pp. 151–166.

Abdel-Hamid, T. K., and Madnick, S.E. *Software Project Dynamics: An Integrated Approach,* Prentice Hall, Englewood Cliffs, NJ, 1991.

Abdel-Hamid, T. K., Sengupta, K., and Hardebeck, M. "The Effect of Reward Structures on Allocating Shared Staff Resources Among Interdependent Software Projects: An Experimental Investigation," *IEEE Transactions on Engineering Management* (41:2), May 1994.

Abdel-Hamid, T. K., Sengupta, K., and Ronan, D. "Software Project Control: An Experimental Investigation of Judgment with Fallible Information," *IEEE Transactions on Software Engineering* (19:6), June, 1993, pp. 603–612.

Ashton, R., and Kramer, S. "Students as Surrogates in Behavioral Accounting Research: Some Evidence," *Journal of Accounting Research* (18), pp. 1–15.

Basili, V. R. "Goal Question Metric Paradigm," in *Encyclopedia of Software Engineering, Volume I.,* by J. J. Marciniak (ed.), John Wiley & Sons, Inc., New York, 1994.

Boehm, B. W. *Software Engineering Economics.* Prentice Hall, Inc., Englewood Cliffs, NJ, 1981.

Boehm, B. W., and Ross, R. "Theory-W Software Project management: Principles and Examples," *IEEE Transactions on Software Engineering* (15:7), July 1989, pp. 902–916.

Brehmer, B. "Dynamic Decision Making: The Control of Complex Systems," *Acta Psychologica* (81), 1992, pp. 211–241.

Brehmer, A., and Brehmer, B. "What Have We Learned about Human Judgment from Thirty Years of Policy Capturing?" in *Human Judgment: The SJT View,* B. Brehmer and C.R.B. Joyce (eds), North-Holland Elsevier, Amsterdam, 1988.

Brooks, F. P. *The Mythical Man Month: Anniversary Edition,* Addison Wesley Publishing Co., Reading, MA, 1995.

Bryan, W. L., and Siegel, S. G. *Software Product Assurance: Techniques for Reducing Software Risk,* Elsevier, New York, 1988.

Camerer, C., and Johnson, E. "The Process-performance Paradox in Expert Judgment," in *Toward a General Theory of Expertise,* K. Ericsson and J. Smith (eds.), Cambridge University Press, Cambridge, England, 1991.

Campbell, D. "Task Complexity and Strategy Development: A Review and Conceptual Analysis," *Academy of Management Review* (13), 1988, pp. 40–52.

Cervone, D., Jiwani, N., and Wood, R. "Goal Setting and the Differential Influence of Self-Regulatory Processes on Complex Decision-Making Performance," *Journal of Personality and Social Psychology* (61:2), 1991, pp. 257–266.

Chesney, A., and Locke, E. "Relationships Among Goal Difficulty, Business Strategies, and Performance on a Complex Management Simulation Task," *Academy of Management Journal* (34), pp. 400–424.

Cougar, J. "Motivation Norms for Software Engineers versus those for Programmer Analysts," *Proceedings of the Seventh International Conference on Information Systems,* L. Maggi, R. Zmud, and J. Wetherbe (eds.), December 15–17, 1986, San Diego, CA, pp. 214–223.

DeMarco, T. *Controlling Software Projects,* Yourdon Press, Inc., New York, 1982.

Earley, P. C., Connolly, T., and Ekegren, G. "Goals, Strategy Development, and Task Performance: Some Limits on the Efficacy of Goal Setting," *Journal of Applied Psychology* (74:1), 1989, pp. 24–33.

Earley, P. C., Northcraft, G. B., Lee, C., and Lituchy, T. R. "Impact of Process and Outcome Feedback on the Relation of Goal Setting to Task Performance," *Academy of Management Journal* (33:1), 1990, pp. 87–105.

Earley, P. C., Wojnaroski, P., and Prest, W. "Task Planning and Energy Expended: Exploration of How Goals Influence Performance," *Journal of Applied Psychology* (72:1), 1987, pp. 107–114.

Erez, M., Earley, P., and Hulin, C. "The Impact of Participation on Goal Acceptance and Performance: A Two-step Model," *Academy of Management Journal* (28) 1985, pp. 50–66.

Garb, H. "Clinical Judgment, Clinical Training, and Professional Experience," *Psychological Bulletin,* (105), 1989, pp. 387–396.

Gilliland, S. W., and Landis, R. S. "Quality and Quantity Goals in a Complex Decision Task: Strategies and Outcomes," *Journal of Applied Psychology* (77:5), 1992, pp. 672–681.

Glickman, S., and Kopcho, J. "Bellcore's Experiences Using Abdel-Hamid's Systems Dynamics Model," *1995 COCOMO*

Conference, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 1995.

Grady, R. B. *Practical Software Metrics for Project Management and Process Improvement,* Prentice Hall, Englewood Cliffs, NJ, 1992.

Humphrey, W. *Managing the Software Process,* Addison-Wesley Publishing Company, Reading, MA, 1989.

Jeffery, D. "The Relationship Between Team Size, Experience, and Attitudes and Software Development Productivity," *The Eleventh Annual International Computer Software Applications Conference (COMPSAC),* Tokyo, Japan, October, 1987.

Jones, C. "Patterns of Large Software Systems: Failure and Success," *Computer,* March 1995, pp. 86–87.

Krasner, H. "The Payoff for Software Process Improvement (SPI): What It Is and How to Get It," *Software Process Newsletter* (1), September 1994, pp. 3–8.

Langholtz, H., Gettys, C., and Foote, B. "Allocating Resources over Time in Benign and Harsh Environments," *Organizational Behavior and Human Decision Processes* (58), 1994, pp. 28–50.

Latham, G. P., and Locke, E. A. "Self-Regulation Through Goal Setting," *Organizational Behavior and Human Decision Processes* (50), 1991, pp. 212–247.

Latham, G. P., Erez, M., and Locke, E. A. "Resolving Scientific Disputes by the Joint Design of Crucial Experiments by the Antagonists: Application to the Erez-Latham Dispute Regarding Participation in Goal Setting," *Journal of Applied Psychology* (73), pp. 753–772.

Lehder, W., Smith, P., and Yu, W. "Software Estimation Technology," *AT&T Technical Journal,* July/August 1988, pp. 10–18.

Locke, E. A. *Generalizing from Laboratory to Field Settings: Research Findings from Organizational Behavior and Human Resource Management,* Heath Lexington, Lexington, MA, 1986.

Locke, E. A., and Bryan, J. F. "The Directing Function of Goals in Task Performance," *Organizational Behavior and Human Performance* (4), 1969, pp. 35–42.

Locke, E. A., and Latham, G. P. *A Theory of Goal Setting and Task Performance,* Prentice Hall, Englewood Cliffs, NJ, 1990.

Locke, E. A., Latham, G. P., and Erez, M. "The Determinants of Goal Commitment," *Academy of Management Review* (13), pp. 23–39.

Locke, E. A., Shaw, K. N., Saari, L. M., and Latham, G. P. "Goal Setting and Task Performance: 1969–1980," *Psychological Bulletin* (90:1), 1981, pp. 125–152.

Londeix, B. *Cost Estimation for Software Development,* Addison-Wesley Publishing Company, Wokingham, England, 1987.

Myers, G. *The Art of Software Testing,* Wiley, New York, 1979.

Powell, F. D. "Study of a Software Development Process Dynamic Model," Technical Report MTR 10314, Mitre Corporation, December 1987.

Pressman, R. S. *Software Engineering: A Practitioner's Approach,* 3rd ed., McGraw-Hill, Inc., New York, 1992.

Pressman, R. S. *A Manager's Guide to Software Engineering,* McGraw-Hill, Inc., New York, 1993.

Rasch, R. H., and Tosi, H. L. "Factors Affecting Software Developers' Performance: An Integrated Approach," *MIS Quarterly,* September 1992, pp. 395–413.

Remus, W. E. "Graduate Students as Surrogates for Managers in Experiments on Business Decision Making," *Journal of Business Research* (14), 1986, 19–25.

SAS/STAT Guide for Personal Computers, Version 6 ed., SAS, Cary, NC, 1987.

Sengupta, K., and Abdel-Hamid, T. K. "Alternative Conceptions of Feedback in Dynamic Decision Environments: An Experimental Investigation," *Management Science* (39), 1993, pp. 411–428.

Sengupta, K., and Abdel-Hamid, T. K. "The Impact of Unreliable Information on the Management of Software Projects: A Dynamic Decision Perspective," *IEEE Transactions on Systems, Man, and Cybernetics* (26), 1996, pp. 177–189.

Steiner, J. D., *Group Process and Productivity,* Academic Press, New York, 1972.

Thayer, R. H., and Fairley, R. "Project Management," in *Encyclopedia of Software*

Engineering Volume II, John J. Marciniak (ed.), John Wiley, New York, 1994, pp. 900–923.

Weinberg, G. M., and Schulman, E. L. "Goals and Performance in Computer Programming," *Human Factors* (16:1), 1974, pp. 70–77.

Winer, B. J. *Statistical Principles in Experimental Design,* McGraw-Hill, New York, 1971, pp. 630–633.

Wofford, J. C., Goodwin, V. L., and Premack, S. "Meta-Analysis of the Antecedents of Personal Goal Level and of the Antecedents and Consequences of Goal Commitment," *Journal of Management* (18:3), 1992, pp. 595–615.

Wood, R. E., and Bailey, T.C. "Some Unanswered Questions about Goal Effects: A Recommended Change in Research Methods," *Australian Journal of Management* (10), 1985, pp. 61–73.

Wood, R. E., and Locke, E. A. "Goal Setting and Strategy Effects on Complex Tasks," in *Research in Organizational Behavior,* by B. M. Staw and L. L. Cummings (eds.), JAI Press, Inc., Greenwich, CT, 1990.

About the Authors

Tarek K. Abdel-Hamid has been a professor of information systems in the Department of Systems Management at the Naval Postgraduate School from 1986 to 1999. In 1997, he took a "re-tooling" sabbatical and enrolled in the Master's program at Stanford's Engineering Economic Systems Department, where he focused on decision analysis and medical informatics. Upon gradutation in 1999, he started PhysioCraft, a medical informatics company. Prior to joining NPS, he spent two-and-a-half years at the Stanford Research Institute. Dr. Abdel-Hamid received his Ph.D. in management information systems from MIT in 1984. His current research interests at PhysioCraft focus on the application of system dynamics to the modeling of human physiology. He is the coauthor of *Software Project Dynamics: An Intergrated Approach* (Prentice Hall, 1991), for which he was awarded the 1994 Jay Wright Forrester Award. In addition to his book, he has authored or coauthored more than 30 papers.

Kishore Sengupta is an associate professor of information systems, Naval Postgraduate School, Monterey, California. In 1996-1997, he was a visiting scholar at the Hong Kong University of Science and Technology. Dr. Sengupta's research interests are in software management, knowledge management, decision making, and electronic commerce. He has published in leading journals in management and information systems, including *Management Science, MIS Quarterly, IEEE Transactions on Engineering Management, Omega, Decision Support Systems, Accounting, Management, and Information Technology,* and *Concurrent Engineering Research and Applications.*

Clint Sweet graduated from the Naval Postgraduate School with an M.S. in information technology management.

APPENDIX

Overview of Model Structure

The current study was conducted within the context of a much broader research effort to study, gain insight into, and make predictions about the dynamics of the entire software development process. A major part of this effort was devoted to the development of a comprehensive system dynamics computer model of software development. The model is currently being used in several research capacities. Through modeling and simulation, the model is used to study/predict the dynamic behavior of the software development process and of the implications of managerial policies and procedures pertaining to the development of software. In our own research, four areas have so far been studied: (1) scheduling; (2) control; (3) quality assurance; and (4) staffing. The exercise produced three kinds of results: uncovered dysfunctional consequences of some currently adopted policies (e.g., in the scheduling area); provided guidelines for managerial policy (e.g., on the allocation of the quality assurance effort); and provided new insights into software project phenomena (e.g., Brooks' Law) (Abdel-Hamid and Madnick 1991).

In addition, the model is being used as an "experimentation microworld" to study dynamic decision-making behavior in the software management domain, e.g., project planning and control (Abdel-Hamid et al. 1993), and the impact of feedback (Sengupta and Abdel-Hamid 1993), unreliable information (Sengupta and Abdel-Hamid 1996), and reward structures (Abdel-Hamid et al. 1994) on performance.

The model was developed on the basis of field interviews of software project managers in five organizations, complemented by an extensive database of empirical findings from the literature. The model integrates the multiple functions of the software development process, including both the management-type functions (e.g., planning, controlling, and staffing) as well as the software production-type activities (e.g., designing, coding, reviewing, and testing).

Figure A1 shows a high-level view of the model's four subsystems: human-resource management, software production, control, and planning, and some of the relations between them. The actual model is very detailed and contains more than 100 causal links; a full description of the model's structure and its mathematical formulation is published in (Abdel-Hamid and Madnick 1991).

Human-Resource Management

This subsystem captures the hiring, assimilation, and transfer of people. We segregate the project's work force into employee types (newly hired and experienced, for example). We make this distinction because new team members are usually less productive than veterans.

This segregation also lets us capture the training process to assimilate new members. The veterans usually train the newcomers, both technically and socially. This is important, because this training can significantly affect a project's progress by reducing the veteran's productivity.

In deciding how big a work force they need, project managers typically consider several factors. One, of course, is the project's scheduled completion date. Another is the work force's stability, so managers try to predict project employment time for new members before they are hired. In general, the relative weight managers give to stability versus completion date changes as the project progresses.

Software Production

This subsystem models development; it does not include the operation and maintenance phases. The development phases included are designing, coding, and testing.

As software is developed, it is reviewed to detect any defects, e.g., using quality assurance activities such as structured walkthroughs. Errors detected through such activities are reworked. Not all software defects are detected during development, however; some escape detection until the testing phase.

The software-production subsystem models productivity and its determinants in great detail. Productivity is defined as potential productivity minus the loss from faulty processes. Potential productivity is the level of productivity that can occur when an individual or group makes the best possible use of its resources, and is a function of the nature of the task and the group's resources (Steiner 1972). Loss from faulty processes are losses in productivity from factors such as communication/coordination overheads and low motivation.

Control Subsystem

As progress is made, it is reported. A comparison of the degree of project progress to the planned schedule is captured within the control subsystem.

In all organizations, decisions are based on the information available to the decision maker. Often, this information is inaccurate. Apparent conditions may be far removed from those actually encountered, depending on information flow, time lag, and distortion.

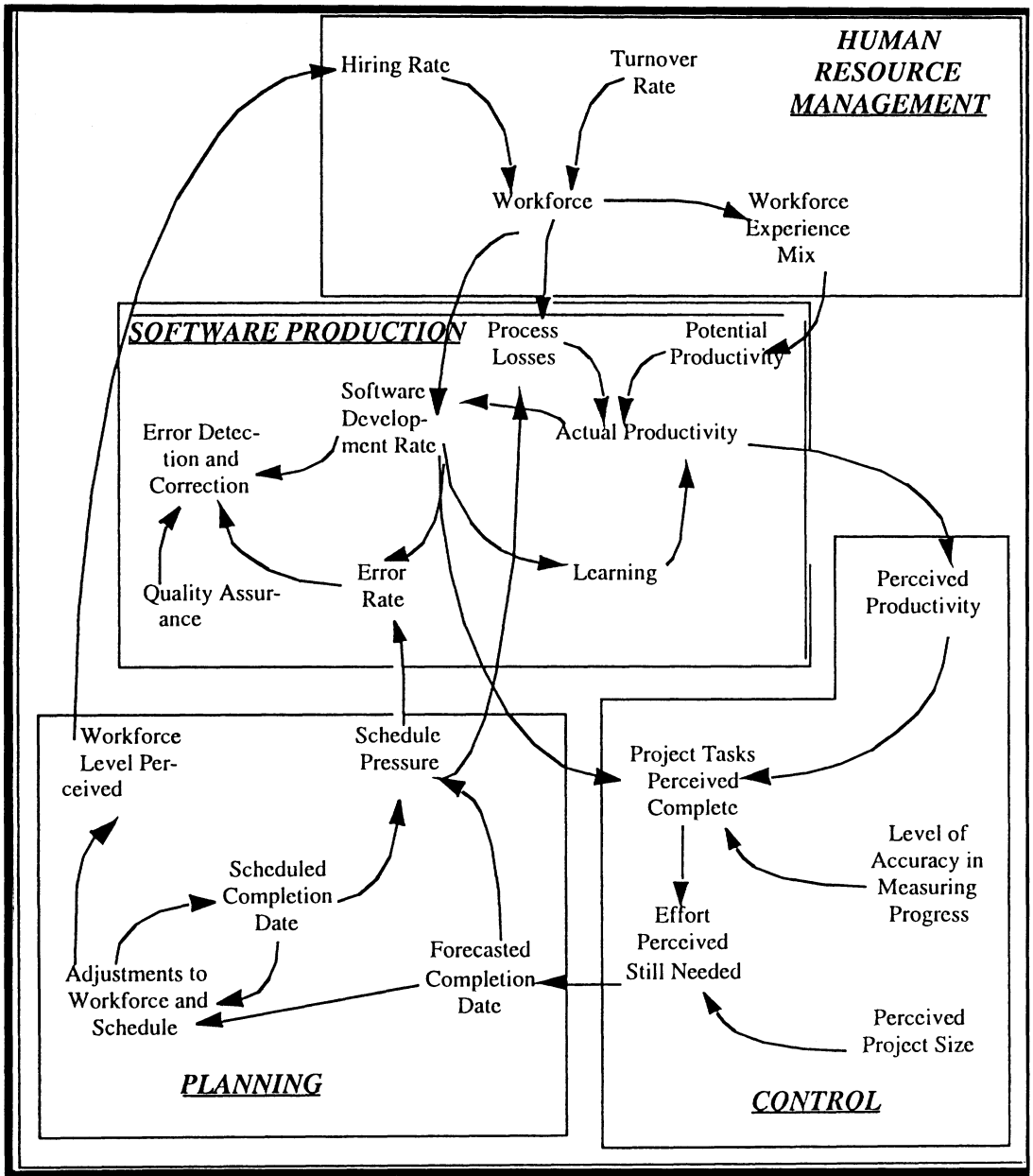


Figure A1. Model Structure

Progress rate is a good example of a variable that is difficult to assess during the project. Because software is basically an intangible product during most of the development, it is difficult to measure things like programming performance and intermediate work. In the earlier phases of development, progress is typically measured by the rate of resource expenditure rather than accomplishments. But as the project advances toward its final stages, work accomplishments become relatively more visible and project members better perceive how productive the work force has actually been.

Progress rate is a good example of a variable that is difficult to assess during the project. Because software is basically an intangible product during most of the development, it is difficult to measure things like programming performance and intermediate work. In the earlier phases of development, progress is typically measured by the rate of resource expenditure rather than accomplishments. But as the project advances toward its final stages, work accomplishments become relatively more visible and project members better perceive how productive the work force has actually been.

Planning Subsystem

In the planning subsystem, project estimates are made and revised as the project progresses. For example, when a project is behind schedule, the plan may be revised to hire more people, extend the schedule, or both.

By dividing the value of person-days remaining at any point in the project by the time remaining, a manager can determine the indicated work force level, which is the work force needed to complete the project on time. However, hiring decisions are not made solely on the basis of scheduling requirements. Managers also consider training requirements and the stability of the work force. Thus, before adding new project members, management assesses the project employment time for the new members. In general, the relative weighting between the desire for work force stability and the desire to complete the project on time is not static; it changes throughout the project's life.

Although management determines the work-force level needed to complete the project, this level does not necessarily translate into the actual hiring goal. The hiring goal is constrained by the ceiling on new hires. This ceiling represents the highest work-force level management believes can be adequately handled by its experienced project members.

Thus, three factors—scheduled completion time, work-force stability, and training requirements—affect the work-force level.

Model Validation

The model was developed on the basis of field interviews of software project managers in five organizations, complemented by an extensive database of empirical findings from the literature. The following set of tests were conducted to validate the model:

- Face validity test. To test the fit between the rate/level/feedback structure of the model and the essential characteristics of real project environments. This fit was confirmed by the software project managers involved in the study.
- Replication of reference modes. To test whether the model can endogenously reproduce the various reference behavior modes characterizing real environments. Reference modes reproduced by the model included a diverse set of behavior patterns both observed in the organizations studied as well as reported in the literature (e.g., the "90% syndrome," diminishing returns of QA effort, the deadline effect, etc.).
- Case studies. Five case studies (two by the first author, and three conducted independently by three separate organizations) were conducted after the model was completely developed.[6] All case studies were conducted in organizations other than the five organizations studied during model development.

[6]The two case studies conducted by the first author are published in Abdel-Hamid and Madnick (1991) and Abdel-Hamid (1993b). Case studies were conducted independently at Bellcore (Glickman and Kopcho 1995), Mitre (Powell 1987), and Bell Laboratories (not published).