Software Project Control: An Experimental Investigation of Judgment with Fallible Information

Tarek K. Abdel-Hamid, *Member, IEEE*, Kishore Sengupta, and Daniel Ronan

Abstract—Software project management is becoming an increasingly critical task in many organizations. While the macro-level aspects of project planning and control have been addressed extensively, there is a serious lack of research on the micro-empirical analysis of individual decision making behavior. In this study we investigate the heuristics deployed to cope with the problems of poor estimation and poor visibility that hamper software project planning and control, and present the implications for software project management. The paper presents a laboratory experiment in which subjects managed a simulated software development project. The subjects were given project status information at different stages of the lifecycle, and had to assess software productivity in order to dynamically readjust project plans. A conservative anchoring and adjustment heuristic is shown to explain the subjects' decisions quite well. Implications for software project planning and control are presented.

Index Terms—Anchoring, experimentation, project control, software productivity, software project management.

I. INTRODUCTION

HOW are software projects controlled? This question speaks to the often frustrating task of delivering software systems on budget and on schedule. Yet, it is a task that is becoming increasingly critical in many organizations as computer-based technology continues to play an ever larger role in determining how organizations operate, how they create their products, and indeed in reshaping the products themselves.

While the software engineering literature has been quite extensive in its coverage of the macro-level aspects of project planning and control, there is a serious lack of research on the micro-empirical analysis of individual decision making behavior. Our objective in this paper is to investigate the heuristics deployed to cope with the problems of poor estimation and poor visibility that hamper software project planning and control, and present the implications for software project management.

II. A MODEL OF PROJECT CONTROL

Fundamentally, control is any process that guides activity toward some predetermined goal. The essence of the concept is in determining whether the activity is achieving the desired results. Fig. 1 depicts the "textbook" model of project control that has been adopted in one form or another for many years in the literature [3]. A control system is shown to have four basic

Fig. 1. Project control model.

elements: a measuring device which detects what is happening; a mechanism for comparing what is actually happening with some standard or expectation of what should be happening; a procedure for altering behavior if the need for doing so is indicated; and a means of transmitting feedback information to the control device [2].

Two information inputs are absolutely vital to effective project control. The first is planning information, such as schedule and resource estimates, which provide the standards used for assessing the significance of what is happening. Indeed, it has been stated that the degree of control over a project can be no greater than the extent to which adequate plans have been made for the project [19]. The second is accurate and timely status information.

III. PROJECT CONTROL IN THE SOFTWARE DEVELOPMENT ENVIRONMENT

Throughout the short history of the software industry, both researchers and practitioners alike have agonized over the industry's continuing failure to adequately control software development [18]. That the problem is both serious and very much alive is highlighted in a recent congressional report, in which eight major software projects in the U.S. Department of Defense experienced a combined cost overrun of more than one billion dollars. The report highlighted a wide range of "common" and "chronic" problems: "These problem areas include lack of cost visibility (i.e., knowing the real extent of costs at a particular point in time) and control; schedule delays; and lack of capability at the time promised" [7].

The reason why controlling software projects has been problematic is no mystery. It has proven to be quite difficult to, on the one hand, accurately estimate development costs and schedules and, on the other, to measure a project's progress rate; the two critical information inputs that are absolutely vital to effective project control.

Manuscript received March 10, 1992; revised December 22, 1992. Recommended by Michael Deutsch.

The authors are with the Department of Administrative Sciences, Naval Postgraduate School, Monterey, CA 93943.

IEEE Log Number 9208633.

U.S. Government work not protected by U.S. Copyright

To address the estimation difficulty, a large number of algorithmic estimation models have been developed over the last decade. However, the utility of such models has proven to be quite poor [11]. As a result, many software development organizations do not yet rely on these tools [26].

On the measurement side, managers continue to complain that "it is difficult to measure performance in programming. It is difficult to diagnose trouble in time to prevent it. It is difficult to evaluate the status of intermediate work such as undebugged programs or design specification and their potential value to the complete project" [17]. Such measurement difficulties are caused both by people-type factors as well as product-type factors.

First, controlling software projects (like most human systems) suffers from the following phenomenon: people are inclined to report favorable information (this is relatively unimportant to management since no corrections are needed) and to withhold unfavorable information (this is the type needed if corrections are to be made). Moreover, the very temperament of the typical programmer, who is highly motivated by technical problems but generally less motivated by management ones, further degrades the measurement process [21].

Second, the software product is relatively intangible during most of the development process, and for which there are no visible milestones to measure progress and quality like a physical product would [6]. "This invisibility is compounded for *large* software, for which logical complexity cannot be maintained in one person's mind, and for which development must be partitioned into a number of tasks assigned to different people" [27].

Having to cope with such impediments to project planning and control is indeed a formidable challenge that has frustrated both software practitioners and researchers alike. Lehman articulated well the common sentiment:

The process of controlling a software engineering project may well be the most talked about and least understood of all the project managers' functions. Many projects have failed because of the managers' inability to adequately define where the project was in the development cycle. Other development efforts have been cancelled not for inadequate programming, or for lack of technology, but from the sheer frustration of all concerned in attempting to determine when, if ever, they would be completed [15].

Sadly, Lehman's observations are as true today as they were more than a decade ago [7].

IV. THE RESEARCH QUESTION

The research question addressed in this paper is the following: How do software project managers cope with poor initial estimates and unreliable measurements in controlling software development throughout the lifecycle? And what are the implications of whatever (compensatory) strategies they opt to use on project dynamics?

More specifically, we will focus on the heuristics that managers use in assessing their teams' software development

productivity. To assess progress on a software project, a manager needs to track work accomplished as well as the resources utilized. Combining both "measures" (by dividing the former by the latter) yields an estimate of the team's productivity. The amount of effort still needed to complete the project may then be estimated as a function of the tasks that remain to be accomplished (total project size minus work already completed) and the estimated productivity. As such, productivity assessment plays a central role in the planning and control process [14].

V. EXPERIMENTAL SETTING

The research question was explored in the context of a role-playing project simulation game. Specifically, 33 graduate students at a U.S. university participated as part of a software engineering management course requirement. The subjects were fifth and sixth quarter masters students studying in the computer systems management curriculum (a six-quarter M.B.A. program with a concentration in information technology).

The simulation game mimics the life of a real software project from the start of the design phase until the end of testing. (A description of the model's structure is provided in the Appendix.) The subject's role was to track the project's progress using status reports on resources used, work accomplished, etc., generated at different stages of the project (by the "simulated" project team), and to estimate the team's average productivity in tasks/person-day. A task is a unit of work, e.g., a software module 50 lines of code long. Using a task as the unit of work for reporting progress and estimating future resources is a common practice [5, p. 615]. For example, Cooper [8] explains how projects are decomposed into "smaller, more comprehensive subtasks until the level is reached where the tasks cannot be subdivided any further. You will end up at the 50/100 lines of code level ... These small tasks also permit the allocation of resources, the loading of manpower, and the subsequent costing (of the project)."

The subjects were told that their productivity estimates would then be used to make the necessary adjustments to the project's staffing level and its schedule. In reality, however, the model's gaming interface was designed to disregard the subjects' estimates. That is, the subjects' individual estimates had no influence upon the simulated project. This ensured that the model provided an identical project lifecycle profile to all subjects. It was possible to maintain such an illusion because, as in real life, productivity in the model is affected by a large number of interacting factors (e.g., changes in staff mix, changes in staff size, learning, changes in lifecycle phases, etc.) that change dynamically throughout the lifecycle. (Post-experimental interviews of the subjects indicated that none of the subjects were at any time in doubt that they were in fact "calling the shots" throughout the project's lifecycle.)

The simulated project used in the experiment was a real project developed at NASA in the early 1980's. The initial estimates of the project's size and cost were 396 tasks and 1111 person-days, respectively. The project ended up growing to 610 tasks in size and it cost 2064 person-days to complete.

Initial Estimate of Project Size: 396 Tasks

Initial Estimate of Overall
Average Productivity: 0.20 Tasks
Man Day

Initial Estimate of
Project Cost: $\frac{396}{0.20} = 1,980$ Man days

Initial Estimate of Project Duration: 320 Days

There is approximately a two month safety
factor applied to the project duration estimate.
(i.e., while any schedule slippage is
undesirable, a slippage of more than
55 days is intolerable.)

Maximum Tolerable Project Duration: 375 Days

In this organization people are typically assigned to more than
one project. They may spend anywhere from 20 to 80% of their
time on a particular project. FOR CLARITY, THE AVERAGE STAFF
SIZE AND SIMULATION OUTPUT WILL BE GIVEN IN FULL-TIME EQUIVALENT
EMPLOYEES. One full time equivalent employee is equal to one
person who spends 100% of his time on the project or two people
that spend 50% of their time on the project.

Average Staff Size: $\frac{1980}{320} = 6$ Full-time
Equivalent Employees

The Project will start with a full time equivalent core team of
1.5 senior designers, and staff up quickly.

Fig. 2. Initial estimates report.

The actual overall staff productivity was, thus, 0.30 tasks per person-day.

The 33 subjects were randomly assigned to three experimental groups of eleven subjects each. The only difference between the three groups was in the initial project estimates provided to each group. As part of the game's documentation, each subject received an "Initial Estimates Report" like the one shown in Fig. 2. Group 2 received an initial productivity estimate of 0.3 tasks/person-day (the actual for the project). Group 1's initial productivity estimate was 0.20 tasks/person-day, and Group 3's initial productivity estimate was 0.45 tasks/person-day. The subjects were told that the initial productivity estimate was derived from an extensive database of historical project statistics that the simulated (although real) organization has developed and maintained in the past five years. The differences between the groups are summarized below:

	GROUP 1	GROUP 2	GROUP 3
Initial productivity estimate (Tasks/person-day)	0.20	0.30	0.45
Initial estimate of project cost (person-days)	1980	1320	880

Table I summarizes the classroom abilities and work experience of the subjects. The groups exhibited no significant difference with respect to classroom ability ($F(2,25) = 0.01, p > 0.9907$) or work experience ($F(2,25) = 0.25, p > 0.7792$).

Each subject was required to provide three productivity estimates during the life of the project:

- 1) At the end of the design phase (end of the fifth month)
- 2) At the middle of the coding phase (end of tenth month)
- 3) At the end of coding (approximately the end of the fifteenth month)

The project required 19 (simulated) months to complete. At each one of these three decision points, the subjects were provided with a progress report on the project's status. An example status report is shown in Fig. 3. The subjects were advised to give whatever weight they saw fit to these reports. As was mentioned above, the subjects were in their final stages of a masters program in computer systems management. As such they appreciated the real life complexities of the control task. For example, they understood the reasons why "percent reported complete" may not be totally reliable, especially in the

TABLE I
SUBJECT CHARACTERISTICS—MEANS AND STANDARD DEVIATIONS ($n = 28$)

Group		Classroom Ability	Work Experience
		Mean (S.D)	Mean (S.D)
1	Initial estimate = 0.2 ($n = 9$)	3.4222 (0.4381)	3.3333 (0.7071)
2	Initial estimate = 0.3 ($n = 9$)	3.3444 (0.4746)	3.1111 (0.7817)
3	Initial estimate = 0.45 ($n = 10$)	3.4200 (0.3615)	3.1000 (0.8756)

Notes:

- Classroom ability indicates the grade received by the subject on a scale of 1-4. The grade received for the experiment was excluded in the calculation of the overall grade.
- Data on work experience was collected on the following scale:
 - 0-4 years' work experience.
 - 5-8 years' work experience.
 - 9-12 years' work experience.
 - 13-16 years' work experience.

CURRENT INTERVAL STATISTICS:		Elapsed Time =	100
INITIAL ESTIMATES: (These will not change throughout the project)			
Project Size	396	Tasks	
Project Cost	1,980.00	Person-Days	
Project Duration	320	Days	
REPORTED STATISTICS AT TIME =		100	Days
% Project Reported Complete	22.92	Percent	
Updated Size of Project	421	Tasks	
Total Number of Full-time Equiv. Staff	3.5	Staff	
Effort Expenditures to Date:			
Development Activities	292.3	Person-Days	
Design and Coding	152.32	Person-Days	
Rework (i.e., fixing errors)	50.69	Person-Days	
Quality Assurance	89.29	Person-days	
Testing	0.00	Person-days	
Total Person-Days Expended	292.30	Person-Days	
New Est. of Project Duration (Start-End)	320	Days	
Max Tolerable Project Duration	375	Days	

Fig. 3. Status report.

early phases of the project. The students also appreciated the dynamic impacts of factors such as work force mix, learning, schedule pressures, etc., on productivity.

Each revised productivity estimate had to be entered into the model through a menu screen as well as hand written on a special estimation sheet which was submitted to a lab attendant before the subject was allowed to continue the simulation. Upon completion of the project, each subject was required to briefly specify the method(s) he/she used to generate the estimates.

The simulation was played over approximately a one hour session, with each subject working alone. Ten percent of each student's grade in the course was based on their performance on this exercise. Good performance was defined in terms of "closure to the actual productivity," i.e., on providing the most accurate productivity estimate. As noted by DeMarco [9, p. 21] "Success for the estimator must be defined as a function of convergence of the estimate to the actual, and of nothing else."

VI. EXPERIMENTAL RESULTS AND STATISTICAL ANALYSIS

The experiment elicited three productivity estimates per subject from 33 subjects. Five subjects made significant errors, and were omitted from the analysis, resulting in a sample size of 28. (The five subjects misapplied the definition of productivity as person-days/task instead of tasks/person-day. While the estimates provided by them can still be used in the analysis, the fact that they made a significant conceptual mistake limits the validity of their estimates.)¹

Table II shows summary statistics for the productivity estimates. Plots of the same statistics are given in Fig. 4.

Following the approach suggested by Winer [25], a multivariate analysis of variance for repeated measures was performed. Since the numbers of subjects in the three groups

¹ We conducted a sensitivity analysis in order to determine how the results would be affected had these subjects been included in the original data set. The productivity estimates reported by these subjects were transformed to their reciprocals and added to the estimates of the 28 subjects. The results remain unchanged.

TABLE II
MEANS AND STANDARD DEVIATIONS OF PRODUCTIVITY ESTIMATES OF THREE TREATMENT GROUPS ACROSS THREE TIME PERIODS ($n = 28$)

Group		Time 100	Time 200	Time 300
		Mean (S.D)	Mean (S.D)	Mean (S.D)
1	Initial estimate = 0.2 ($n = 9$)	0.2649 (0.0836)	0.2559 (0.0893)	0.2390 (0.0944)
2	Initial estimate = 0.3 ($n = 9$)	0.3109 (0.0305)	0.3552 (0.0420)	0.3564 (0.0651)
3	Initial estimate = 0.45 ($n = 10$)	0.4260 (0.0853)	0.3830 (0.0910)	0.3790 (0.0795)

Fig. 4. Group productivity estimates.

were not equal (after the discarded subjects the sizes were 9, 9, and 10), the analysis was conducted with the General Linear Models procedure in SAS [22]. Table III contains the MANOVA results. The results show that the productivity estimates of subjects across the three different groups were significantly different ($p < 0.0003$). We, therefore, reject the null hypothesis of no significant differences among groups. In other words, the subjects' productivity estimates were indeed influenced by the initial estimates (anchors) provided to them.

There was no within-subjects effect of time ($p > 0.81$), or any interaction between the initial estimates and time ($p > 0.1$). The lack of a significant time effect indicates that on the whole, subjects' estimates did not change significantly over time, i.e., did not show a (statistically) significant upward or downward trend. The absence of an interaction effect indicates that subjects in different groups did not behave differently over time, i.e., subjects with different initial estimates did not converge to a common point over time. Collectively, therefore, the two results show that a) subjects adhered to their respective anchors in making productivity estimates, and b) did not converge to the actual estimates over time.²

²A univariate test yielded a significant time*initial estimates interaction ($p < 0.04$). However, univariate methods for repeated measures can present difficulties, and frequently result in Type I errors [16]. Moreover, the sphericity test was strongly rejected ($p < 0.001$) for our sample. It is, therefore, inappropriate in this case to use the results of the univariate test.

Conservative Anchor Dragging: According to Kahneman and Tversky [24], anchoring is quite common in *static* decision tasks involving judgement under uncertainty:

In many situations, people make estimates by starting from an initial value that is adjusted to yield the final answer... (The) adjustments are typically insufficient. That is, different starting points yield different estimates, which are biased toward the initial values. We call this phenomenon anchoring.

On the other hand, there has been very little research specifically examining anchoring in a *dynamic* environment. Hogarth [12], reports an experiment in which subjects used anchoring and adjustment effectively in making repeated forecasts in an exponential series. Mackinnon and Wearing [28] report similar results on a similar experimental task. However, the two studies employed relatively simple tasks, and subjects received accurate feedback. No previous study has examined anchoring in complex, dynamic environments characterized by fallible information.

A rationale for why decision makers might resort to anchoring over multiple time periods is provided by Hogarth [12]:

... in continuous environments, the adjustment and anchoring heuristic essentially provides the basic mode of judgment. Consider, for instance, how one forms impressions of strangers through interaction. That is, in discrete incidents a single (possibly inaccurate) judgment is made. In continuous processing, however, a series of adjustment and anchoring responses, all of which may be relatively inaccurate, takes one progressively to the target. (pp. 206–207)

The results of Fig. 4, however, seem to indicate otherwise. While a series of adjustments were indeed made, they did not necessarily take the subjects (e.g., Group 1) "progressively to the target." Rather, the subjects opted for a more conservative route. This conservative anchoring strategy can be expressed as follows: when confronted with two candidate (but tentative) productivity estimates (the anchor and the one implicit in the status report data), the subjects tended to lean toward the lower of the two. In a project situation, a lower (i.e., more conservative) productivity estimate is "safer" since it justifies a higher level of resource requirements and/or time to complete

TABLE III
MULTIVARIATE ANALYSIS OF VARIANCE ($n = 28$)

Source of Variation	S.S	Degrees of freedom	F	Significance of F
<i>Between Subjects</i>				
Initial Estimate	0.292	2	11.37	0.0003
Subjects-within-cells	0.321	25		
<i>Within Subjects</i>				
Time	0.983	2, 24	0.20	0.8161
Time*Initial Estimate	0.732	4, 48	2.02	0.1052

TABLE IV
EVIDENCE OF CONSERVATIVE ANCHOR DRAGGING ($n = 28$)

Group		Anchoring Strategy		Total
		Anchor Dragging	No Anchor Dragging	
1	Initial estimate = 0.2	17	10	27
2	Initial estimate = 0.45	20	10	30
Total:		37	20	
Note: $t = -2.1032$; degrees of freedom = 36.0; $p = 0.0425$				

the project, thereby providing a cushion of safety for the project manager. Such conservatism is consistent with findings in the behavioral decision theory field, where "an abundance of research has shown that human beings are conservative processors of *fallible* information" ([10], emphasis added). We will label this decision strategy "conservative anchor dragging," i.e., continuous anchoring moderated by conservatism.

In order to test the strength of the conservative anchor dragging phenomenon, we conducted an additional test to examine the decision strategies adopted by individual subjects. The decision strategies were inferred from the following test: If the estimate chosen was lower than the midpoint between the anchor and the reported estimate, the subject was considered to have used conservative anchor dragging. Thus, for a subject in Group 1, the candidate estimates at time 100 were the anchor (0.2) and the value implicit in the status report data (0.33). If the subject's estimate was lower than 0.265 (the midpoint between 0.33 and 0.2), the subject was considered to have used anchor dragging.

Table IV shows the results aggregated over three time periods for Groups 1 and 3, i.e., those with initial estimates of 0.2 and 0.45 (Group 2, which was given the perfect estimate of 0.3 was excluded from the analysis). Table IV confirms that subjects in both groups converged toward the lower of the anchor and reported estimate, particularly in time periods 200 and 300. The conservative anchor dragging strategy was exhibited in almost two-thirds of the instances, and significantly more often than otherwise ($t = -2.1032$; $d.f. = 36$; $p < 0.05$).

VII. DISCUSSION

Since any conclusions are contingent on the limitations present, the study's limitations are discussed prior to the implications of the results.

A. Limitations

The main limitations of the study center on the experimental setting and the participants. First, the experimental setting was a computer-based laboratory experiment. Laboratory research entails giving up the richness of context to obtain control. For example, to control for information feedback to the experimental subjects, feedback information was limited to status reports. In reality project managers may rely on more than status reports in determining project status (including conversations with programmers, interviews, etc.). The question is, what price did we pay for such a simplification?

As noted earlier, the evidence indicates that software project visibility continues to be rather poor. This suggests that even with such additional cues status information remains unreliable (especially early in the lifecycle). Therefore, the inclusion of such additional cues would not have altered the fundamental experimental task, namely, weighing two sets of unreliable information (initial estimates and status information) in estimating the project team's software productivity. However, while such additional cues may not necessarily eliminate status uncertainty, they may or may not affect the weights that managers give to the different information sources. Such an issue would be an interesting one to investigate in a future study.

Second, the generality of the current results is tempered by the use of graduate students as managerial surrogates. Surveying over 250 recent studies on the dynamics of small social groups, Bettenhausen [4] concluded that "... findings using student groups in manipulated settings often generalize to organizational settings as well or better than findings from intact groups that are frequently confounded by unique, unmeasured contextual factors." In one particularly relevant study, Remus [20] found no significant differences between

Fig. 5. Productivity impact on project behavior.

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 graduate students are acceptable surrogates in this experimental investigation.

B. Implications for Project Planning and Control

Having shown that conservative anchor dragging can lead to significantly different productivity-assessment profiles, our objective here is to explore the impact such differences may or may not have on project behavior. Conceptually, it is not difficult to see why such impacts may be significant. For example, Fig. 5's causal loop diagram demonstrates how estimated productivity directly influences work force hiring and firing. The work force level acquired determines the communication and training overheads incurred on the project which, in turn, affect the progress rate and the ultimate cost of the project. In addition, estimated productivity can influence project behavior through more subtle, and less direct means. For example, if a project is perceived to be behind schedule (e.g., due to an underestimate of productivity), software developers may feel pressured to work harder in order to bring the project back on schedule.

Going a step beyond a conceptual appreciation of the impacts of estimated productivity on project behavior, we further explored the impacts of the different productivity-assessment profiles on project dynamics. In Fig. 6 we show the results of two simulation runs that incorporate the mean productivity estimates of Group 1 (initial anchor = 0.2) and Group 3 (initial anchor = 0.45). Recall that in the role-playing experiment, the model interface was designed to disregard the subjects' productivity estimates. In the simulation runs of Fig. 6, on the other hand, the groups' mean productivity estimates were incorporated into the simulated project, i.e., they were the values deployed by the simulated project team in managing the project.

Because Group 1's productivity estimate starts low and remains low, a larger person-day estimate is made for the project. As a result, a larger work force is assembled early on, and remains relatively stable until the end-of-project "winding-down" phase. Contrast this with Group 3, where the work force level is significantly lower throughout the first half of the lifecycle. This is caused by the overestimated productivity value which, in effect, underestimates the project's cost in person-days. This (not uncommon) underestimation problem inevitably catches up with management; causing the project to fall behind schedule. This, in turn, induces the sharp upward correction in the work force level around day 200.

As was mentioned earlier, the actual average productivity for the simulated project was 0.30 tasks/person-day. The simulated project costs and completion times for Groups 1 and 3 are summarized below:

	Group 1	Group 3
- project cost (person-days)	2956.5	1995.7
- completion time (days)	324	347
- average productivity (tasks/person-day)	0.21	0.31

Notice that overestimating productivity (Group 3) does not lead to any significant increase over the project's nominal productivity level (0.3 tasks/person-day). On the other hand, underestimation does lead to a significant drop in the team's average productivity (from 0.3 to 0.21). This, we feel, is quite an interesting finding, and of particular relevance to the practicing software manager.

We can offer two reasons to explain this result. Underestimating productivity translates directly into an overestimated project cost in person-days. When this happens, work typically expands to consume the overestimated resources. For example, work expansion could take the form of goldplating, or it could be in the form of an increase in people's slack time activities (such as catching up on the mail, coffee breaks, etc.) [5]. Thus, low productivity estimates can lead to more "wasteful" processes which, in turn, "fulfills the prophecy" of lower development productivity.

Second, since a lower productivity estimate translates into an overestimated project cost, it can also lead to hiring a larger work force. This, in turn, can cause higher communication and training overheads on the project, which eventually leads to a decrease in team productivity, as is captured by the feedback loop of Fig. 5.

VIII. CONCLUSION

Past studies have demonstrated anchoring as a pervasive phenomenon in how individuals make decisions in static situations. This paper contributes to research in decision making and software engineering by demonstrating the dynamic nature of anchoring in the software project environment. The software project control task is somewhat unique in that the anchor (initial project estimates) as well as the feedback information (status reports) are both unreliable. (Although, as the project

Fig. 6. Simulation results.

proceeds the feedback information becomes relatively more reliable.)

Our results suggest that anchoring persists over time and is not necessarily mitigated by feedback provided in the intervening periods. Specifically, the experimental subjects adopted a "conservative anchor-dragging strategy," i.e., when faced with two estimates of productivity, they accorded greater weight to the lower (i.e., more conservative) figure. The "advantage" to the project manager of adopting a more conservative estimate is that it justifies a higher level of resource requirements and/or time to complete the project, thereby providing a cushion of safety. The potential pitfall to the organization, however, is that underestimating productivity can become self-fulfilling.

APPENDIX OVERVIEW OF MODEL STRUCTURE

Fig. 7 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 [1].

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 errors, e.g., using quality assurance activities such as structured walkthroughs. Errors detected through such activities are reworked. Not all software errors 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 [23]. Loss from faulty processes are losses in productivity from things like communication and coordination overhead 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.

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 towards its final stages, though, 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

Fig. 7.

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 the training requirements and the work force's stability. 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 organi-

zations, 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. All case studies were conducted in organizations other than the five organizations studied during model development.

REFERENCES

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

- [2] R. N. Anthony and J. Dearden, *Management Control Systems*. Chicago, IL: Richard D. Irwin, Inc., 1980.
- [3] A. O. Awani, *Data Processing Project Management*. Princeton, NJ: Petrocelli, 1986.
- [4] K. L. Bettenhausen, "Five years of group research: What we have learned and what needs to be addressed," *J. Management*, vol. 17, no. 2, pp. 345-381 1991.
- [5] B. W. Boehm, *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice-Hall, 1981.
- [6] F. P. Brooks, "The Silver Bullet: Essence and accidents of software engineering," *IEEE Comput. Mag.*, pp. 10-19, Apr. 1987.
- [7] Committee on Government Operations, "DoD automated information systems experience runaway costs and years of schedule delays while providing little capability." U.S. House of Representatives, Rep. 101-382, Nov. 20, 1989.
- [8] J. Cooper, "Software development management planning," *IEEE Trans. Software Eng.*, vol. SE-10, no. 1, pp. 22-26 Jan. 1984.
- [9] T. DeMarco, *Controlling Software Projects*. New York: Yourdon, 1982.
- [10] W. Edwards, "Conservatism in human information processing," in *Formal Representation of Human Judgment*, B. Kleinmuntz, Ed. New York, Wiley, 1968.
- [11] M. Genuchten and H. Koolen, "On the use of software cost models," *Inform. and Management*, vol. 21, pp. 37-44, 1991.
- [12] R. Hogarth, "Beyond discrete biases: Functional and dysfunctional aspects of judgmental heuristics," *Psychological Bull.*, vol. 90, no. 2, pp. 197-217, 1981.
- [13] B. Kitchenham and N. Taylor, "Software project development cost estimation," *J. Syst. and Software*, vol. 5, pp. 267-278 1985.
- [14] N. R. Howes, "Managing software development projects for maximum productivity," *IEEE Trans. Software Eng.*, vol. SE-10, no. 1, pp. 27-35, Jan. 1984.
- [15] J. H. Lehman, "How software projects are really managed," *Datamation*, pp. 119-129, Jan. 1979.
- [16] S. A. LaTour and P. W. Miniard, "The misuse of repeated measures analysis in marketing research," *J. Marketing Res.*, vol. XX, pp. 45-57, Feb. 1983.
- [17] H. D. Mills, *Software Productivity*. Canada: Little, Brown & Co., 1983.
- [18] H. D. Mills, "Software engineering: Retrospect and prospect," in *Proc. Twelfth Annu. Int. Comput. Software and Appl. Con. (COMPSAC)*, Oct. 5-7, 1988, pp. 89-96.
- [19] U. W. Pooch and P. F. Gehring, "Toward a management philosophy for software development," in *Advances in Computer Programming Management*, T. A. Rullo, Ed. Philadelphia, PA: Heyden, 1980.
- [20] W. E. Remus, "Graduate students as surrogates for managers in experiments on business decision making," *J. Business Res.*, vol. 14, pp. 19-25, 1986.
- [21] P. Rook, "Controlling software projects," *Software Eng. J.*, Jan. 1986.
- [22] SAS/STAT, *Guide for Personal Computers*, Version 6. Cary, NC: SAS 1987.
- [23] I. D. Steiner, *Group Process and Productivity*. New York: Academic, 1972.
- [24] A. Tversky and D. Kahneman, "Judgement under uncertainty: Heuristics and biases," *Science*, vol. 185, no. 27, pp. 1124-1131, Sept. 1974.
- [25] B. J. Winer, *Statistical Principles in Experimental Design*. New York: McGraw-Hill, 1971.
- [26] C. Wrigley and A. Dexter, "Software development estimation models: A review and critique," in *Proc. ASAC Conf.*, Univ. Toronto, 1987.
- [27] R. W. Zmud, "Management of large software development efforts," *MIS Quarterly*, vol. 4, no. 2, pp. 45-56, June 1980.
- [28] A. J. Mackinnon and A. J. Wearing, "Feedback and the forecasting of exponential change," *Acta Psychologica*, vol. 76, pp. 177-191, 1991.

Tarek K. Abdel-Hamid (S'80-M'83) received the B.S. degree in aeronautical engineering from Cairo University, Cairo, Egypt, in 1972, and the Ph.D. degree in management information systems from MIT in 1984.

He is Associate Professor of information systems in the Department of Administrative Sciences, Naval Postgraduate School, Monterey, CA. Prior to joining NPS, he spent two and a half years at the Stanford Research Institute as a senior consultant in the information technology area. He has been an advisor to NASA's Jet Propulsion Lab., since 1986, on the development of computer-based tools for software project management. His research interests focus on software project management, system dynamics, expert simulators, and management information systems. He is the co-author of *Software Project Dynamics: An Integrated Approach* (Englewood Cliffs, NJ: Prentice-Hall, 1991). In addition to his book, he has authored or co-authored more than 20 papers, published in journals such as the *Communications of the ACM*, *Journal of Systems and Software*, *MIS Quarterly*, *IEEE SOFTWARE*, and *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*.

Dr. Abdel-Hamid is a member of the Association for Computing Machinery, SIM, the IEEE Computer Society, and the Systems Dynamics Society.

Kishore Sengupta received the Ph.D. degree in information systems from Case Western Reserve University, Cleveland, OH, in 1990.

He is currently Assistant Professor of Information Systems at the Department of Administrative Services, Naval Postgraduate School, Monterey, CA. Previously, he was with AT&T's Network Software Center and Arthur Young (now Ernst and Young). His research interests are in decision support for dynamic tasks, computer supported cooperative work, learning, and intelligent tutoring. His research appears in journals such as *Management Science* and *MIS Quarterly*, conference proceedings, and book chapters. He is currently working on a book on cognitive aspects of decision support in distributed environments.

Dr. Sengupta is a member of the American Association for Artificial Intelligence and the Institute of Management Sciences.

Daniel Ronan received the B.S. degree in mathematical sciences from the Coast Guard Academy, New London, CT, and the M.S. degree in computer systems management from the Naval Postgraduate School, Monterey, CA.

He is Chief of the Business Systems Section at the Coast Guard's Electronics Engineering Center, Wildwood, NJ. He is currently managing three independent software development projects dealing with financial management, electronics inventory, and message management that will be distributed throughout the Coast Guard.