A Study of Staff Turnover, Acquisition, and Assimilation and Their Impact on Software Development Cost and Schedule

TAREK K. ABDEL-HAMID

TAREK K. ABDEL-HAMID received his B.Sc. in aeronautical engineering from Cairo University, Cairo, Egypt, and his Ph.D. in Management Information Systems from MIT. He is Assistant Professor of Information Sciences at the Dept. of Administrative Sciences of the Naval Postgraduate School in Monterey, CA. Prior to joining NPS, he was a senior consultant at the Stanford Research Institute. His research interests are software project management, system dynamics, expert simulators, and management information systems. He has authored or coauthored more than 20 papers and technical reports; his papers have appeared in *Communications of the ACM*, *Journal of Systems and Software*, *MIS Quarterly*, and *IEEE Transactions on Software Engineering*. He is also coauthor of a forthcoming book on software project management.

ABSTRACT: In this article we investigate how staff turnover, acquisition, and assimilation rates affect software development cost and schedule. A system dynamics model of the software development process is employed as our experimentation vehicle. In addition to permitting less costly and less time-consuming experimentation, simulation-type models can provide useful insights into the causes behind the different behavior patterns observed. Our results indicate that staff turnover, acquisition, and assimilation rates can increase a project's cost and duration by as much as 40 to 60 percent. This suggests that the three staffing variables are indeed critical for the successful development of software systems, as well as for the accurate estimation of software development cost and schedule.

KEY WORDS AND PHRASES: Software development, human resource management, project staffing, software project management, and software development cost and schedule.

Introduction

THE HUMAN RESOURCE DIMENSION of software project management has become an area of substantial research interest in the past few years. There are several reasons for this: personnel costs are skyrocketing relative to hardware costs [8]; information systems staff sizes are growing dramatically with little time for adequate selection and training [7]; chronic problems in software development and implementation are being traced to personnel shortcomings [17]; and information systems personnel continue to exhibit a troublesome turnover rate [21]. While most practicing software managers are willing to concede the idea that they've got more people worries than technical worries, they seldom manage that way:

> The main reason [they] tend to focus on the technical rather than the human side of the work is not because it's more crucial, but because it's easier to do. . . . Human interactions are complicated and never very crisp and clean in their effects, but they matter more than any other aspect of the work. . . . If you find yourself concentrating on the technology rather than the sociology, you're like the vaudeville character who loses his keys on a dark street and looks for them on the adjacent street because, as he explains, "The light is better there."[17]

The objective of this article is to shed some light on the dynamics of three human resource issues, namely, staff turnover, staff acquisition, and staff assimilation. The first issue—turnover—continues to be a troublesome problem in the information systems field. In many organizations the turnover rate has reached 30 percent or higher, causing real alarm. While several studies have investigated this turnover issue, the focus has been mostly limited to delineating the factors causing the high turnover rates. Our objective here is different. It is to understand the impact of staff turnover on the cost and schedule of a software project.

Because of the continuing shortages in a long list of information systems occupations, finding quality software professionals can be a real challenge, requiring long-term planning and the commitment of significant resources. Staff acquisition lead times and their impact on project cost and duration is the second staffing variable we investigate in this study.

The third staffing issue addressed is the *staff assimilation* period and its impacts. The assimilation period is the "orientation period" needed to acquaint newly acquired team members with the mechanics of the project, integrate them into the project team, and train them in the necessary technical areas. The length of this assimilation period can vary depending on such things as the nature of the project and the training resources committed. According to one estimate, it takes 18 months for new software employees to become maximally effective [21].

In software project management, it is remarkably easy to propose hypotheses and remarkably difficult to test them. Controlled experiments have proven to be too costly and time consuming [25]. Even when affordable, the isolation of the effect and the evaluation of the impact of any given practice within a large, complex, and dynamic project environment can be exceedingly difficult [20]. Accordingly, it is useful to seek other methods for testing hypotheses.

Simulation modeling provides a viable laboratory tool for such a task. In addition to permitting less costly and less time-consuming experimentation, simulation-type models make "perfectly" controlled experimentation possible. Indeed:

> The effects of different assumptions and environmental factors can be tested. In the model system, unlike the real systems, the effect of changing one factor can be observed while all other factors are held unchanged. Such experimentation will yield new insights into the characteristics of the system that the model represents. By using a model of a complex system, more can be learned about internal interactions than would ever be possible through

> manipulation of the real system. Internally, the model provides complete control of the system's organizational structure, its policies, and its sensitivities to various events [19].

In the next section we propose a system dynamics-based simulation approach to the study of the software development process in general and the staffing issues in particular. A discussion of the information sources utilized to develop the model is first presented, followed by an overview of the model's structure. In the later sections of the article we present and discuss the model's experimental results pertaining to the impacts of staff turnover, acquisition, and assimilation on software project behavior.

A System Dynamics Model of Software Development

IT IS IMPORTANT TO NOTE THAT OUR RESEARCH WORK on software project staffing is being 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 has been devoted to the development of a comprehensive system dynamics computer model of software development. The model is currently being used in several research capacities, one of which is to serve as a laboratory vehicle for conducting experimentation in the area of project staffing, the topic of this article.

The model was developed on the basis of a field study of software project managers in five organizations. The process involved three information gathering steps. First, we conducted a series of interviews with software development project managers in three organizations. The purpose of this set of interviews was to gain a first-hand account of how software projects are "really" managed. The information collected in this phase, complemented with our own software development experience, was the basis for formulating a skeleton system dynamics model of software project management.

The second step involved an extensive review of the literature. The skeleton model served as a road map in carrying out this literature review. When this exercise was completed, many knowledge gaps were filled, giving rise to a second, much more detailed version of the model.

In the third and final step:

> The model is exposed to criticism, revised, exposed again and so on in an iterative process that continues as it proves to be useful. Just as the model is improved as a result of successive exposures to critics a successively better understanding of the problem is achieved by the people who participated in the process [30].

The setting for this final step was a series of intensive interviews with software project managers at three organizations (only one of which was included in the first group).

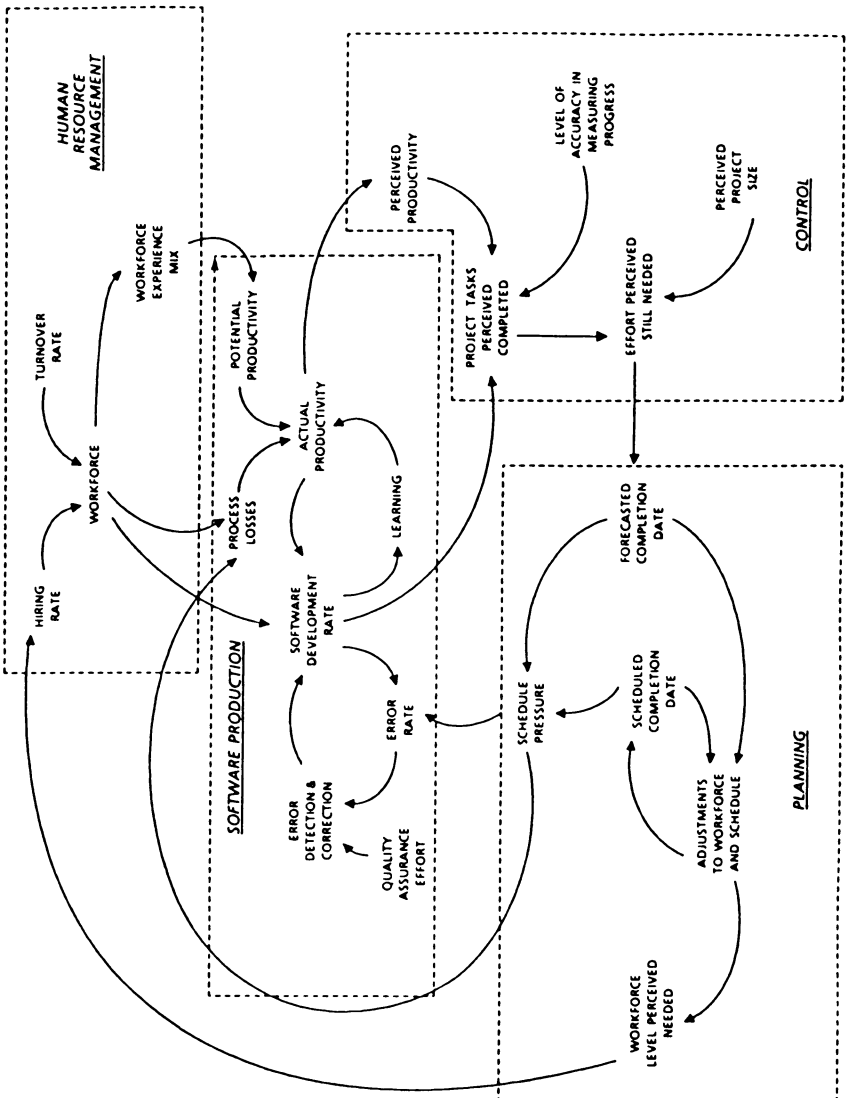


Figure 1. Overview of Model Structure

Figure 1 depicts a highly aggregated view of the model's four subsystems, namely: (1) The Human Resource Management Subsystem; (2) The Software Production Subsystem; (3) The Control Subsystem; and (4) The Planning Subsystem. The figure also illustrates some of the interrelations between the four subsystems. Since the actual model is very detailed, containing over a hundred causal links, only a high level description of the model can be presented in the limited space of this article. For a full discussion of the model's structure, its mathematical formulation, and its validation, the reader is referred to [1, 2, 4].

The Human Resource Management Subsystem

The Human Resource Management Subsystem captures the hiring, training, assimilation, and transfer of the human resource. The project's total workforce is segregated into different types of employees, e.g., "Newly Hired Workforce" and "Experienced Workforce." Segregating the workforce into such categories is necessary for two reasons. First, newly acquired team members are less productive (on the average) than the "old timers" [15]. Secondly, it allows us to capture the training processes involved in assimilating the new members into the project team. This training of newcomers, both technical and social, is usually carried out by the "old-timers" [10, 37]. This can have a significant impact on the progress of a project, because as the old-timer is helping the new employee learn the job, his/her own productivity is reduced.

On deciding upon the total workforce level desired, project managers typically consider a number of factors. One important factor is the project's scheduled completion date. As part of the planning function, management determines the workforce level it believes is necessary to complete the project on schedule. In addition to this factor, consideration is also given to the stability of the workforce. Thus, before adding new project members, management tries to contemplate the project employment time for the new members. In general, the relative weight given to workforce stability versus on-time completion is dynamic, i.e., it will change with the stage of project completion. For example, toward the end of the project there could be considerable reluctance to bring in new people since there just wouldn't be enough time to acquaint the new people with the mechanics of the project, integrate them into the project team, and train them in the necessary technical areas.

Turnover is captured in the model through the "Quit Rate" of "Experienced Workforce." That is, we are assuming no turnover among the "Newly Hired Workforce," since it is quite unlikely for a newly acquired staff member to quit within a few months of joining the project. The "Quit Rate" is formulated as a first-order exponential delay. Such delays are primary building blocks in system dynamics models. In the Appendix we show how a first-order exponential delay looks schematically, how it is formulated mathematically, and how it behaves over time.

The Software Production Subsystem

The Software Production Subsystem models the software development process. The

operation and maintenance phases of the software lifecycle are, thus, not included. The development lifecycle phases incorporated include the designing, coding, and testing phases. Notice that the initial requirements definition phase is also excluded. There are two reasons for this. The primary reason relates to the desire to focus this study on the "indigenous" software development organization—the project managers and the software development professionals—and how their policies, decisions, actions, etc. affect the success/failure of software development. The requirements definition phase was thus excluded, since in many environments the definition of user requirements is not totally within the control of the software development group [22]. Second, "Analysis to determine requirements is distinguished as an activity apart from software development. Technically, the product of analysis is non-procedural (i.e., the focus is functional)" [22].

As software is developed, it is reviewed to detect any errors, e.g., through 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.

A large section of the Software Production Subsystem is devoted to modeling software productivity and its determinants in great detail. The formulation of software productivity takes the following general form:

$$\text{Productivity} = \text{Potential Productivity} - \text{Losses Due to Faulty Process}$$

Potential productivity is defined as "the maximum level of productivity that can occur when an individual or group . . . makes the best possible use of its resources" [34]. It is a function of two sets of factors, the nature of the task (e.g., product complexity, database size) and the group's resources (e.g., personnel capabilities, software tools). Losses due to faulty process refer to the losses in productivity incurred as a result of factors such as communication and coordination overheads and/or low motivation.

The Control Subsystem

Decisions made in any organizational setting are, of course, based on what information is actually available to the decision maker(s). Often, this available information is inaccurate. Apparent conditions may be far removed from the actual or true state, depending on the information flows that are being used and the amount of time lag and distortion in these information flows. Thus, system dynamicists go through great lengths to differentiate in their models between actual and perceived model variables [19].

The true progress rate in a software project is a good example of a variable that is often difficult to assess during the project. Because software is basically an intangible product during most of the development process, "It is difficult to measure performance in programming. . . . It is difficult to evaluate the status of intermediate work such as un-debugged programs or design specification and their potential value to the complete project" [24].

How, then, is progress measured in the project's control system? Our own field study findings corroborate those reported in the literature, namely, that progress, especially in the earlier phases of software development, is typically measured by the rate of expenditure of resources rather than by some count of accomplishments [16]. Baber [6] explains:

> It is essentially impossible for the programmers to estimate the fraction of the program completed. What is 45% of a program? Worse yet, what is 45% of three programs? How is he to guess whether a program is 40% or 50% complete? The easiest way for the programmer to estimate such a figure is to divide the amount of time actually spent on the task to date by the time budgeted for that task. Only when the program is almost finished or when the allocated time budget is almost used up will he be able to recognize that the calculated figure is wrong.

When progress in the early phases is measured solely by the expenditure of budgeted resources, status reporting ends up being nothing more than an echo of the original plan. As the project advances toward its final stages, though, work accomplishments become relatively more visible and project members become increasingly more able to perceive how productive the workforce has actually been.

The Planning Subsystem

In the Planning Subsystem, initial project estimates (e.g., for completion time, man-days) are made at the beginning of the project. These estimates are then revised, as necessary, throughout the project's life. For example, to handle a project that is perceived to be behind schedule, plans can be revised to add more people, extend the schedule, or do a little of both. Such planning decisions are driven by factors that change dynamically throughout the project lifecycle. For example, we explained earlier that while management may respond to a delay in the early stages of a project by increasing staff size, there is often great reluctance to do that later in the lifecycle.

The DE-A Software Project: A Case Study

AS PART OF MODEL VALIDATION, a case study was conducted at NASA to test the model's accuracy in replicating the dynamic behavior of a real software project, namely, NASA's DE-A project. (NASA was not one of the five organizations studied during model development.) The DE-A project was conducted at the Systems Development Section of the Goddard Space Flight Center (GSFC) at Greenbelt, Maryland to design, implement, and test a software system for processing telemetry data and providing attitude determination and control for the DE-A satellite. The development and target operations machines were the IBM S/360-95 and -75. The programming language was FORTRAN.

At the start of the project, the estimates for system size, total development effort,

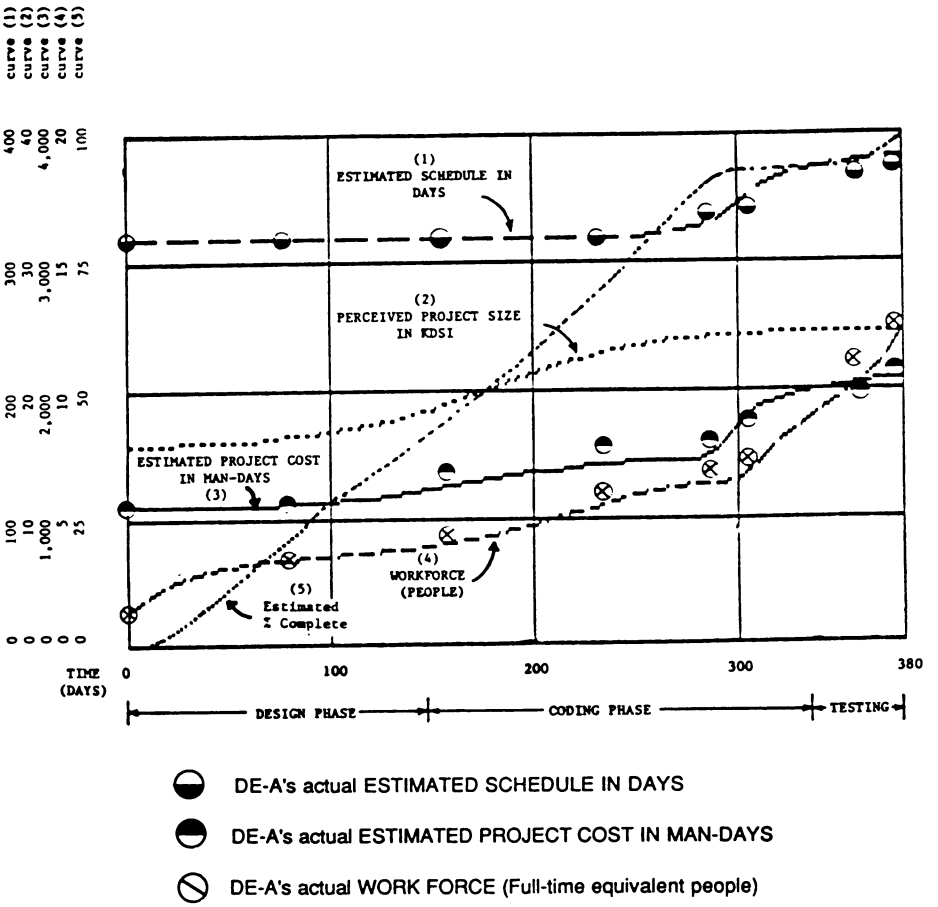


Figure 2. Model Simulation of the DE-A Project

and schedule were 16,000 delivered source instructions (DSI), 1,100 man-days, and 320 (working) days respectively. Upon completion, the DE-A project's actual results were as follows:

project size 24,400 DSI
development cost 2,200 man-days
completion time 380 working days

Figure 2 depicts the model's simulation of the DE-A software project. As the figure indicates, the model's results replicated the project's actual behavior (represented by the 0 points in the figure) quite accurately.

Notice how DE-A's management held on to the project's initial "Estimated Schedule in Days" during most of the development phase of the project, even as the project experienced an increase in its size. Adjustments were instead made to the project's

workforce level. This behavior is not atypical. It arises, according to DeMarco [16], because of political reasons:

> Once an original estimate is made, it's all too tempting to pass up subsequent opportunities to estimate by simply sticking with your previous numbers. This often happens even when you know your old estimates are substantially off. 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.

The project's workforce pattern, on the other hand, does not conform to the staffing pattern typically portrayed in the literature (where the workforce level rises, peaks, and then drops back to lower levels as the project proceeds toward the system testing phase). Because NASA's launch of the DE-A satellite was tied to the completion of the DE-A software, serious schedule slippages were strongly resisted. Specifically, all software was required to be accepted and frozen 90 days before launch. As this "Maximum Tolerable Completion Date" was approached, pressures developed that overrode workforce stability considerations. That is, project management became increasingly willing to "pay any price" necessary to avoid overshooting the 90-day-before-launch date. This translated, as the figure indicates, into a management that was increasingly willing to add more people. (In [2] we investigate whether such a staffing policy does or does not contribute to the project's late completion.)

In the remaining sections, we utilize the model to investigate how changes in the turnover, acquisition, and assimilation variables affect the cost and completion time of the DE-A project.

Experiment 1: Turnover Rate

TURNOVER VARIES ENORMOUSLY FROM ONE ORGANIZATION TO ANOTHER. "We hear of companies with a ten percent turnover, and others in the same business with a hundred percent or higher turnover" [17]. The high turnover rates of information systems personnel have been attributed to both market demand factors (e.g., strong market demand for software professionals, lucrative salaries) as well as to internal organizational factors (e.g., inadequate career paths, poor job design).

At NASA's GSFC, the turnover rate was approximately 20 percent. Using the first-order exponential formulation of the Appendix, we can translate any turnover rate to an equivalent average employment time as follows:

$$L(t) = L(0)*e^{-t/T}$$

where,

L = experienced workforce
t = time (years)
T = average employment time (years)

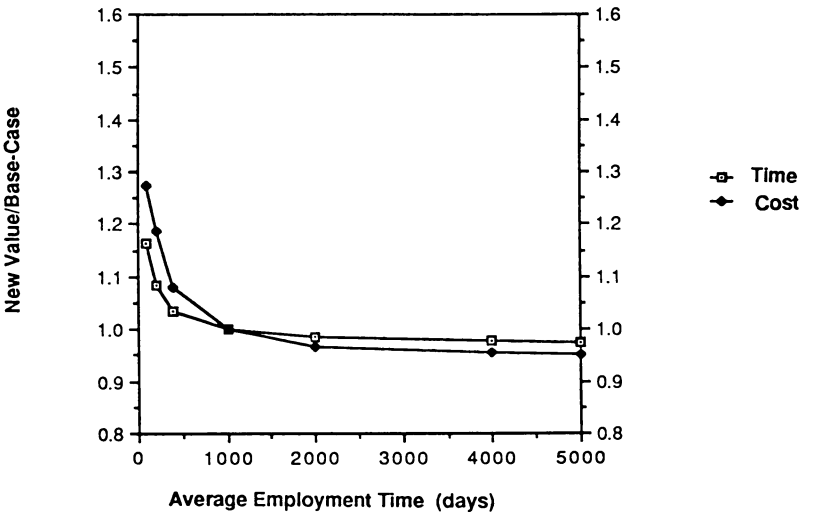


Figure 3. Project Cost and Duration for Different "Average Employment Times"

Thus, for a 20 percent annual turnover rate we get,

$$0.80*L(0) = L(0) * e^{-1/T}$$

This leads to

$$T = 1 / -1\,n\,(0.8)$$
$$= 4.48 \text{ years}$$

In the model, time is defined in terms of working days. One week is five working days, and one year is 48 working weeks (i.e., 240 working days). The average employment time calculated above, thus, translates into 1,075 working days.

To investigate the impact of turnover, we resimulated the DE-A project using values for average employment time that ranged from 100 to 5,000 days. A smaller average employment time signifies a higher rate of turnover on the project, and vice versa. The results are depicted in Figure 3. As expected, higher levels of turnover have a negative impact on both cost and duration. The impact is particularly severe when the average employment time drops below 1,000 days, which is approximately 2.5 times the project's duration of 387 days.

An attractive utility of a simulation-type experimentation tool is that it not only provides answers to "What?" but can, in addition, provide useful insights into the reasons "Why?" In Figure 4a we plot a number of project variables for the base-case simulation (average employment time = 1,075 days), and in 4b when the average employment time is reduced to 100 days. Note the differences in the workforce mix between the two runs. As people quit and new staff are hired to replace them at the higher rate (Figure 4b), the project's "Fraction of the Workforce Experienced" (which is the ratio of the "Experienced Workforce" to the total workforce) drops. This has a negative effect on the team's overall productivity. There are several reasons for this.

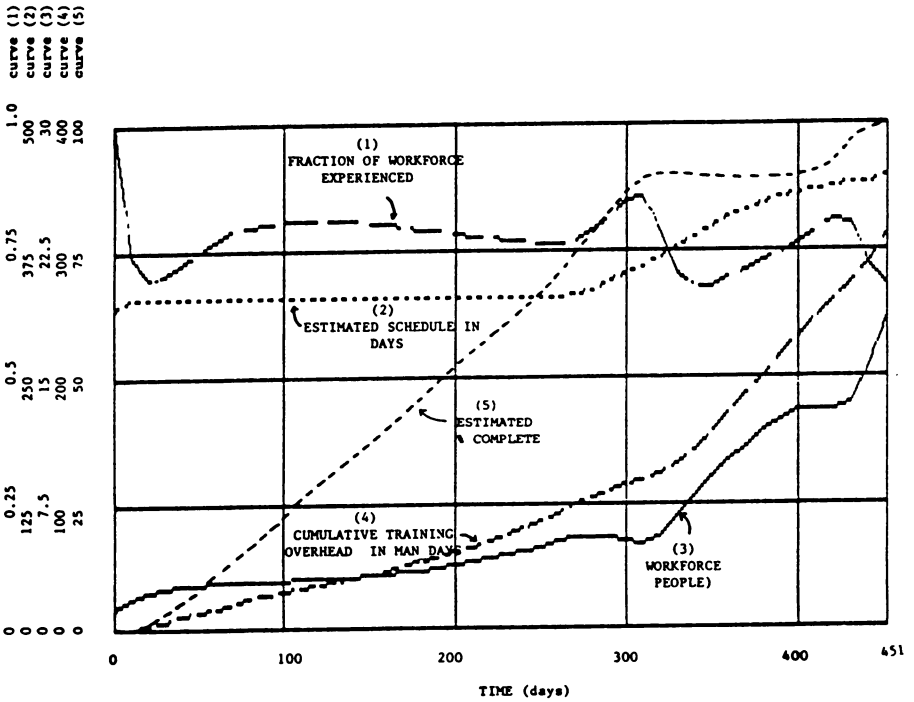


Figure 4 a. Average Employment Time = 100 Days

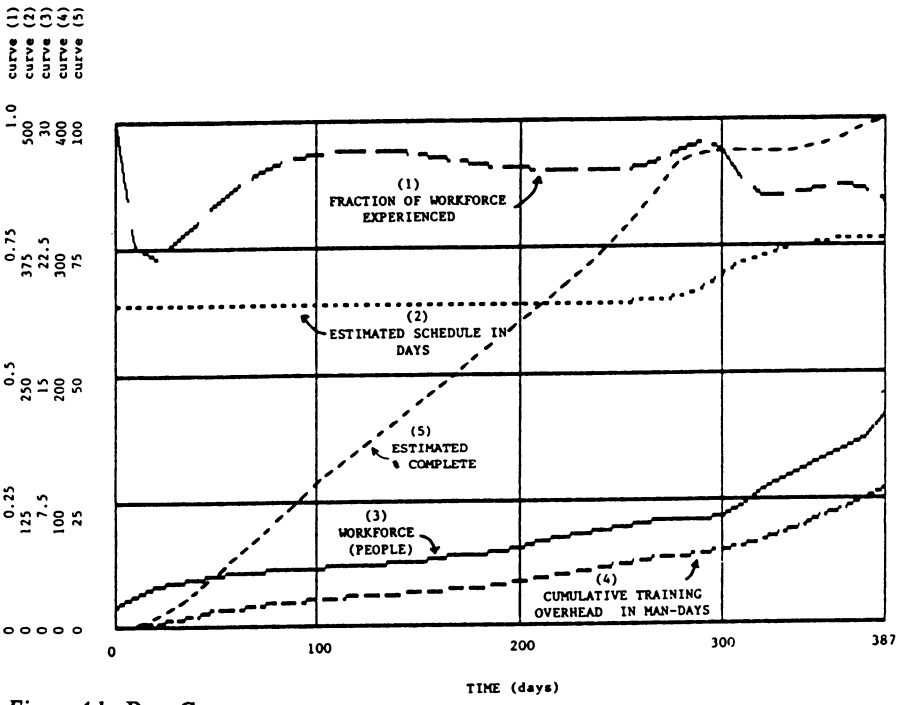


Figure 4 b. Base Case

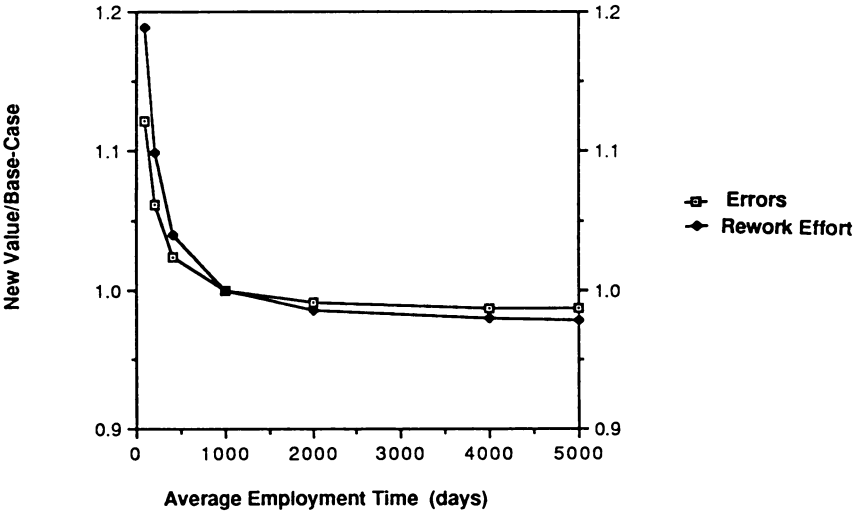


Figure 5. Error and Rework Effort for Different "Average Employment Times"

First, the productivity of newly hired employees is lower during the assimilation period while they strive to learn the project's ground rules and the details of the system. It has been estimated that the productivity of a newly hired staff member is 35 to 50 percent lower than that of an experienced staff member [9, 26]. Second, the higher the influx of new people into a project, the higher will be the training overhead incurred. This is quite evident from the "Cumulative Training Overhead" curves, which represent the cumulative amount of man-days expended to train the new project members. The cumulative training overhead is significantly higher in the high turnover rate scenario of Figure 4b than it is in the base case. Third, newly hired employees are also more error prone than their experienced counterparts [18]. The larger the number of errors committed on a project, the larger will be the drain on the team's productivity as resources are diverted from new project tasks to reworking those errors. Figure 5 shows how the number of errors and the effort expended to rework them increase on the DE-A project as the turnover rate increases.

The drop in productivity caused by the above factors translates into a higher cost and a slower progress rate. Interestingly, though, the progress delays are not readily detectable. Notice that the "Estimated % Complete" curves, which track the progress rate as it is reported on the project, are quite similar in Figures 4a and 4b through the project's first calendar year (240 working days). A combination of programmers' optimism and the less than perfect accuracy in measuring development progress early in the lifecycle act to mask the full extent of the delays [11, 6]. The classic result is the "90 percent completion syndrome" phenomenon, i.e., where estimates of the fraction of work completed increase as originally planned until a level of 80 to 90 percent is reached, and then they increase only very slowly until the project is actually completed. A comparison of the "Estimated Percent Complete" curves of Figures 4a and 4b suggests that while the DE-A project did experience the "90 percent completion syndrome," the problem becomes even more severe as turnover increases.

As visibility improves in the final stages of the development phase, the extent of the backlog becomes evident. And as this happens, management reacts by sharply increasing the workforce level, in an attempt to avoid overshooting the completion-date deadline. Because the team's productivity is lower in the high turnover rate scenario of Figure 4b, the work backlog ends up being larger, which in turn induces a larger increase in the workforce level.

Experiment 2: Staff Acquisition Delay

ADDING STAFF TO A SOFTWARE PROJECT TAKES TIME. While some internal staff may be available within a short period of time, others (especially when management is seeking special skills) will take a much longer time to acquire. On the DE-A project, hiring delay was 30 working days (i.e., 1.5 calendar months). This is somewhat lower than indicated in the literature [23 and 29]. DE-A's low hiring delay can be attributed to two factors. First, the Computer Science Corporation (CSC), had a special arrangement with the Goddard Space Flight Center, under which a pool of CSC software professionals was available for quick transfer to work on NASA projects. Second, NASA's human resource policies encouraged rapid intra-organizational deployment of resources [13].

The impacts of changes in the hiring delay on project cost and duration are exhibited in Figure 6. In this case, the impacts on project cost and project duration are qualitatively different. As hiring delay increases, project duration increases, but project cost actually drops.

The drop in project cost can be attributed to the fact that a larger hiring delay constrains the inflow of new people into the project and thus leads to a smaller workforce size. This is particularly evident in the later stages of the project. Recall that in the base case (Figure 4a), a sharp rise in the workforce occurs in the final stages as management struggles to meet the schedule deadline. When the hiring delay is increased, such a rapid workforce buildup becomes infeasible. This is demonstrated in the new simulation run of Figure 7, in which the hiring delay is increased from its base-case value of 30 days to 75 days. The longer hiring delay discourages management from acquiring new people late in the lifecycle, as they realize that there isn't enough time for hiring the new people and for spending the time and effort needed to bring them up to speed.

Thus, the smaller the hiring delay is, the larger the influx of new people into the project, which in turn will lead to a larger workforce size and a lower "Fraction of Workforce Experienced." A larger workforce size increases the communication overhead on the project, while the lower "Fraction of Workforce Experienced" leads to higher rework and training overheads. Both changes translate into a higher project cost.

But if project cost increases as hiring delay decreases, why does project duration decrease? To understand why, we need to distinguish between a new hiree's *net cumulative* contribution to the team's productivity, and his or her impact on the team's *average* productivity. Consider the latter first. When a new person is added to a project,

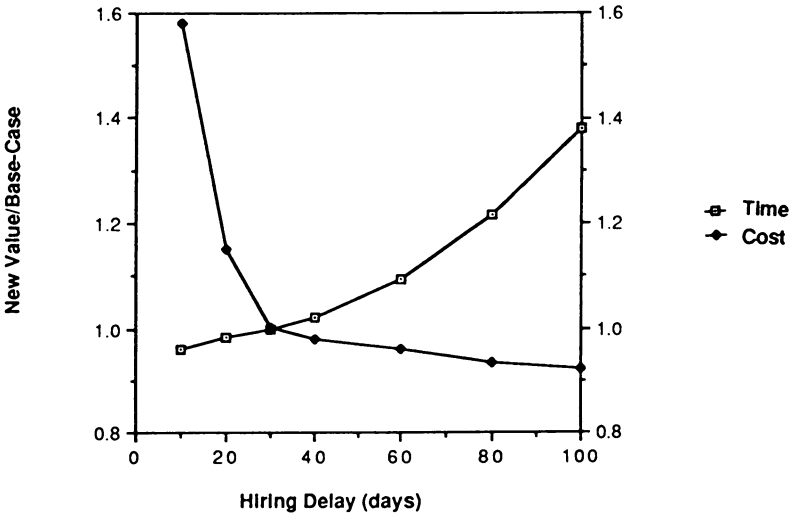


Figure 6. Project Cost and Duration for Different Hiring Times

he/she will communicate with many others on the team. It is widely held that communication overhead increases in proportion to n^2, where n is the size of the team [32, 33]. As communication overhead increases, the team's average productivity decreases. This, in turn, leads to a higher project cost.

For the project's schedule to suffer also, the drop in productivity must be large enough to render an additional person's *net cumulative* contribution to the project team to be, in effect, a negative contribution. 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 man-days from direct project work to the training of and communicating with the new staff member. And we need to calculate the cumulative contribution, because while a new hiree's net contribution might be negative initially, as training takes place and the new hiree's productivity increases, the net contribution becomes less and less negative, and eventually (given enough training on the project) the new person starts contributing positively to the project. Only when the cumulative impact is a negative one will the addition of the new staff member translate into a longer project completion time.

To take a simple example, consider a team of size N and with average productivity P. If a new person is added to the team increasing its size from N to $N + 1$, we can assume that the team's average productivity will drop slightly from P to P' (due to the increase in the communication and training overheads). The progress rate on the project, and hence the schedule, will slow down only if the net contribution of the added person is a negative one, i.e., if $(N + 1)*P'$ is less than $N*P$.

Notice that the results of Figure 6 suggest that Brooks' law, which states that adding manpower to a late project makes it later [11], does not seem to apply to the DE-A project environment. In [2] we discuss this issue in more depth, and explain the reasons why.

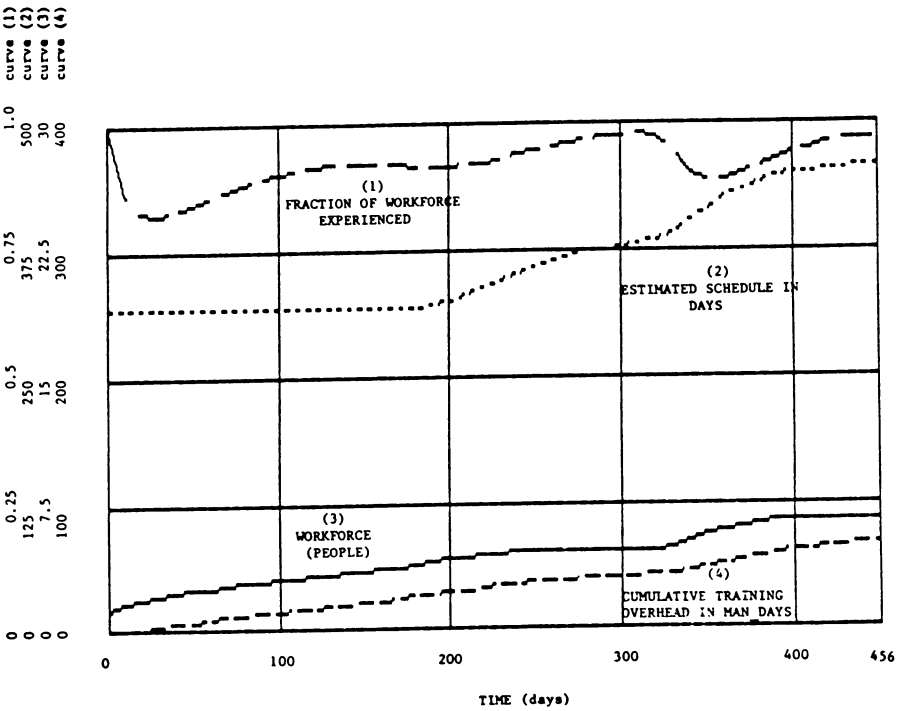


Figure 7. Hiring Delay = 25 Days

Experiment 3: Assimilation Delay

> We all know that a new employee is quite useless on day one or even worse than useless, since someone else's time is required to begin bringing the new person up to speed. By the end of a few months, the new person is doing some useful work; within five months, he or she is at full working capacity. A reasonable assessment of start-up cost is therefore approximately three lost work-months per new hiree [17].

The "Average Assimilation Delay" is the average time it takes to acquaint new recruits with the project's goals, integrate them into the project team, and train them in the necessary technical areas. The "Average Assimilation Delay" on the DE-A project is 20 working days. This is a value that is much lower than those suggested by DeMarco and Lister and others in the literature [17, 21]. The reason, again, has to do with the special arrangement NASA had with the Computer Sciences Corporation. As was mentioned earlier, on many occasions, software professionals were recruited from CSC to work on Goddard projects. This tapped pool of software professionals has, over the years, gained experience with the NASA project environment. And as a result, when recruited on a new project, CSC professionals are assimilated at a faster rate.

The simulation results of Figure 8 indicate that as the "Average Assimilation Delay"

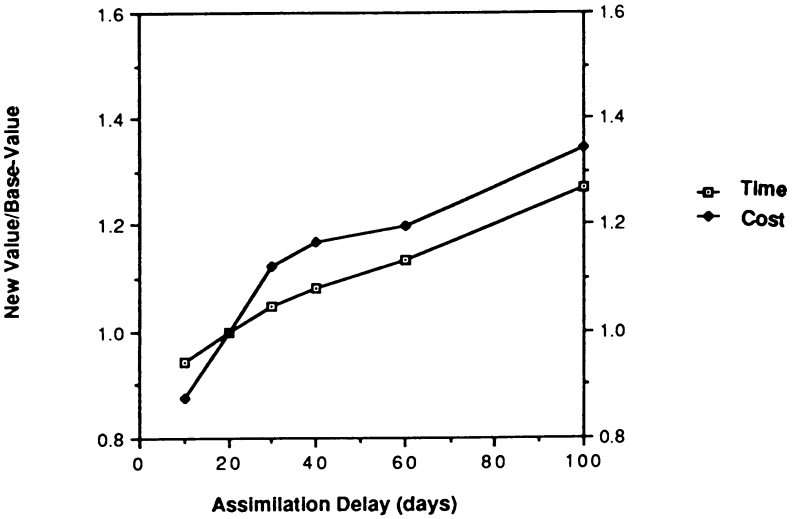


Figure 8. Project Cost and Duration for Different Assimilation Delays

increases, both cost and time increase as well. As has already been explained, "unassimilated" employees are less productive, more error prone, and require a lot of hand-holding. Therefore, the larger the assimilation period, the larger will be the productivity drain on the project, and, as a consequence, the higher the cost and the slower the progress rate.

Figure 9 is a simulation of the DE-A project when the "Average Assimilation Delay" is increased to 75 days (from its base-case value of 20 days). With a larger assimilation period, the "Fraction of Workforce Experienced" dips lower and rises slower during the periods of rapid workforce buildup. This translates into a longer overall apprenticeship period on the project, which of course leads to a much higher training overhead. Also notice that the sharp rise in the workforce level in the final stages of the project is even higher here than in the base case. Because the team's productivity is lower, the work backlog that surfaces at the end of the development phase ends up being a larger backlog, which in turn induces a larger increase in the workforce level.

Conclusion

IN THIS ARTICLE OUR AIM WAS TO UNDERSTAND how staff turnover, staff acquisition, and staff assimilation rates affect software development cost and schedule. Heretofore, the impact of such staffing practices on software development has been difficult to quantify. Controlled experimentation is too costly and time-consuming to be practical in the software project management area. Even when affordable, the isolation of the effect and the evaluation of the impact of any given practice within a large, complex, and dynamic project environment can be exceedingly difficult. The research vehicle proposed in this paper addresses the above difficulties. We developed a comprehensive system dynamics model of the software development process and used it as an experi-

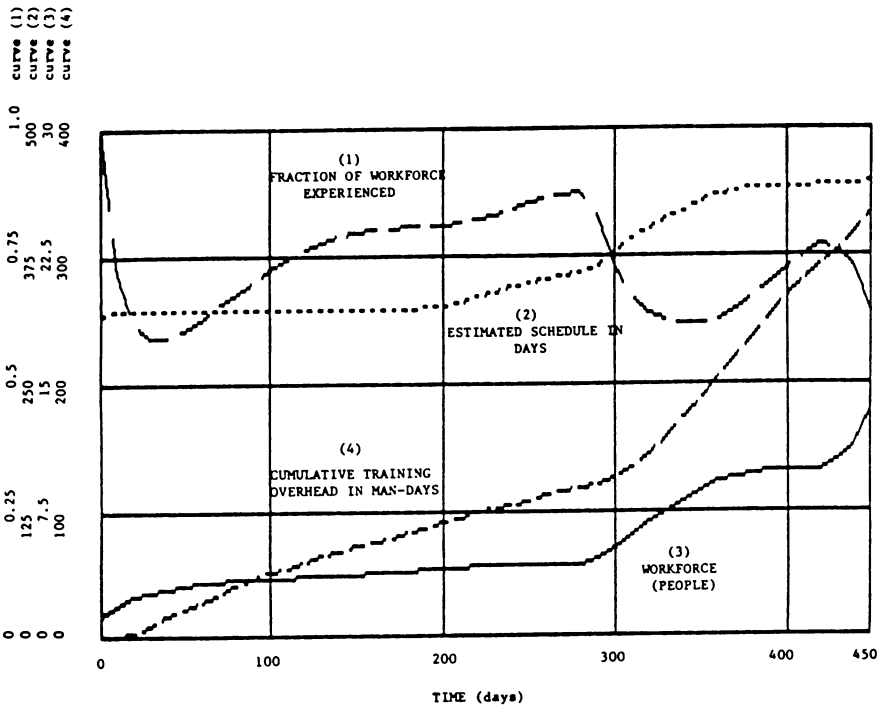


Figure 9. Assimilation Delay = 75 Days

mentation vehicle to study the sensitivity of project cost and duration to variations in the staffing variables. In addition to permitting less costly and less time-consuming experimentation, the simulation model provided useful insights into the causes behind the different behavior patterns observed. Our results indicate that staff turnover, acquisition, and assimilation rates can vary a software project's cost and duration by as much as 40 to 60 percent. With such a significant impact, these three staffing variables can only be viewed as critical to the successful management of software development. It also suggests that such staffing variables need to be explicitly incorporated in software estimation models if we are to increase the accuracy of such models.

APPENDIX

An appendix treating first order exponential delay can be found on p. 40.

REFERENCES

1. Abdel-Hamid, T. K. The Dynamics of Software Development Project Management: An Integrative System Dynamics Perspective. Unpublished Ph.D. dissertation, Sloan School of Management, MIT, January, 1984.

2. Abdel-Hamid, T. K. The dynamics of software project staffing: a system dynamics

based simulation approach. *IEEE Transactions on Software Engineering*, 14, 2 (February 1989).

3. Abdel-Hamid, T. K., and Madnick, S. E. Impact of schedule estimation on software project behavior. *IEEE Software*, 3, 5 (July 1986), 70–75.

4. Abdel-Hamid, T. K., and Madnick, S. E. *Software Project Management*. Englewood Cliffs, NJ: Prentice Hall, to be published in 1988a.

5. Abdel-Hamid, T. K., and Madnick, S. E. Modeling the dynamics of software project management. (In its second review for publication in the *Communications of the ACM* [1988b].)

6. Baber, R. L. *Software Reflected*. New York: North Holland, 1982.

7. Bartol, K. M., and Martin, D. C. Managing information system personnel: a review of the literature and managerial implications. *MIS Quarterly*, 6, 4 (December, 1982), 49–70.

8. Birnbaum, J. Computer: A survey of trends and limitations. *Science*, 215, 4534 (February 12, 1982), 760–765.

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

10. Bott, H. S. The personnel crunch. In *Perspectives on Information Management.*, J. B. Rochester, ed. New York: John Wiley & Sons, 1982.

11. Brooks, Jr., F. P. *The Mythical Man-Month*. Reading, MA: Addison-Wesley, 1978.

12. Canning, R. G. Managing staff retention and turnover. *EDP Analyzer*, 15, 8 (August, 1977), 1–13.

13. Chapman, R. *Project management in NASA*. National Aeronautics and Space Administration, Washington, DC, 1973.

14. Corbato, F. J., and Clingen, C. T. A managerial view of the multics system development. In *Research Directions in Software Technology*, P. Wegner, ed. Cambridge, MA: MIT Press, 1979.

15. Cougar, J. D., and Zawacki, R. A. *Motivating and Managing Computer Personnel*. New York: John Wiley & Sons, 1980.

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

17. DeMarco, T., and Lister, T. *Peopleware: Productive Projects and Teams*. New York: Dorset House, 1987.

18. Endres, A. B. An Analysis of Errors and Their Causes in System Programs. *IEEE Transactions on Software Engineering*, 1, 2 (June 1975), 140–149.

19. Forrester, J. W. *Industrial Dynamics*. Cambridge, MA: MIT Press, 1961.

20. Glass, R. L. *Modern Programming Practices: A Report from Industry*. Englewood Cliffs, NJ: Prentice Hall, 1982.

21. Laudon, K. C., and Laudon, J. P. *Management Information Systems*. New York: Macmillan, 1988.

22. McGowan, C. L., and McHenry, R. C. Software Management. In *Research Directions in Software Technology*, P. Wegner (ed.). Cambridge, MA: MIT Press, 1980.

23. McLaughlin, R. A. That old bugaboo, turnover. *Datamation*, 25, 11 (October 1979), 97–101.

24. Mills, H. D. *Software Productivity*. Toronto: Little, Brown, 1983.

25. Myers, G. J. *Software Reliability: Principles and Practices*. New York: John Wiley & Sons, 1976.

26. Okada, M. Software development effort estimation study—a model for CAD/CAM system development experiences. The Sixth International Computer Software and Applications Conference (COMPSAC), Chicago, November 1982.

27. Pressman, R. S. *Software Engineering: A Practitioner's Approach*. New York: McGraw-Hill, 1982.

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

29. Rifkin, G. Finding and keeping DP/MIS professionals. *Computer World* (June 3, 1985).

30. Roberts, E. B. (ed.). *Managerial Applications of System Dynamics*. Cambridge, MA: MIT Press, 1981.

31. Schein, E. H. *Organizational Psychology*. 3rd ed. Englewood Cliffs, NJ: Prentice Hall, 1980.

32. Scott, R. F., and Simmons, D. B. Predicting Programming Group Productivity—A Communications Model. *IEEE Transactions on Software Engineering*, 1, 4 (December 1975), 411–414.

33. Shooman, M. L. *Software Engineering—Design, Reliability, and Management*. New York: McGraw-Hill, 1983.

34. Steiner, I. D. *Group Process and Productivity*. New York: Academic Press, 1972.

35. Tanniru, M. R., et al. Causes of turnover among DP professionals. *Proceedings of the Eighth Annual Computer Personnel Research Conference*, Miami, FL (June 1981).

36. Thayer, R. H. and Lehman, J. H. Software Engineering Project Management: A Survey Concerning U.S. Aerospace Industry Management of Software Development Projects. Sacramento Air Logistics Center, McClellan Air Force Base, CA (November 1977).

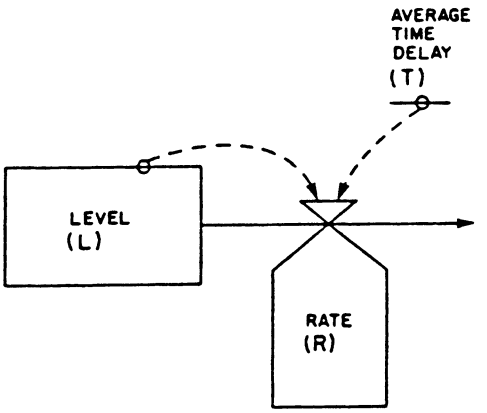
37. Winrow, Bruce. Acquiring Entry-Level Programmers. In *Computer Programming Management*, J. Hannan, ed. Pennsauken, NJ: Auerbach, 1982.

38. Zelkowitz, M. V. Perspectives on software engineering. *Computing Surveys,* 10, 2 (June 1978), 197–216.

APPENDIX

First Order Exponential Delay

A. Schematic



B. Mathematical information

At any time (t),

$$R(t) = L(t)/T$$

Also,

$$\frac{d}{dt} L(t) = -R(t) = -L(t)/T$$

Separating variables and integrating both sides yields

$$L(t) = L(0)\, e^{-t/T}$$

And it can be shown that the average time spent in the delay = T.

C. Behavior