Brooks' Law Revisited: A System Dynamics Approach

Pei Hsia Chih-tung Hsu David C. Kung
Computer Science Engineering Department
The University of Texas at Arlington
Arlington, TX 76019-0015, USA
E-mail: {hsia, chsu, kung}@cse.uta.edu

Abstract

The Brooks' Law says that adding manpower to a late software project makes it later. Brooks developed the law through observation of many projects and derived the generalization. His explanation was quite reasonable and convincing. However, it becomes a debilitating statement to any software project manager who is faced with a late project. This paper presents an in-depth study of Brooks' Law using the system dynamics simulation approach. Unlike the first order approximation of Brooks' Law, we found some interesting results that can benefit the practicing project managers. We have found that adding people to a late project will always increase its cost, but the project may not always be late. Only under certain degree of sequential constraints among project tasks will the project be delayed. To investigate the impact of sequential constraint, we simulated numerous task conditions and found that there is a time line T for each project that if enough manpower is added before T, the project still can finish before the scheduled deadline. If manpower adjustment is after T, then the project will definitely be late. This way, the Brooks' Law is revised and turned into a useful guiding principle to benefit software development.

1. Introduction

Despite the recent advances in software development and management technologies, software development continues to suffer schedule delays and budget overruns. When a project is behind schedule, software managers respond by bringing people into the project. The result is, as suggested by the famous Brooks' Law, a further delayed or even collapsed project. The purpose of this study is to answer two questions: (1) What is the impact of adding people into a software project in terms of completion time and cost? (2) What is the best time to add new people and how many should be added? To answer these two questions, we have developed a system dynamics simulation model.

The remainder of this paper is organized as follows. Related studies on Brooks' Law are reviewed in Section 2. Section 3 discusses the dynamic implications of Brooks' Law. The sequential constraint of development tasks is modeled in Section 4. Section 5 summarizes the results of testing the system dynamics model. Our simulated results are presented in Section 6. Section 7 presents our study's conclusions.

2. Background

Brooks' Law has been extensively addressed in the past. Gordon and Lamb studied Brooks' Law and suggested that the best way to recover from a slipping schedule is to add more people than might be expected to be necessary, and to add them early [4]. Three factors are considered in their study: time loss due to new staff learning, time loss due to teaching by experienced staff, and time loss due to group communication. They suggest to add more manpower than you think is necessary as soon as you sense trouble, then do not change anybody's job until the project is finished [4].

Weinberg addressed Brooks' Law from the system dynamics perspective [6]. He argued that the effect of Brooks' Law is caused by an increased coordination and training overhead. More coordination overhead means more work has to be done. The training load on the experienced workers leads to a reduced amount of productive work being done. The effect of Brooks' Law can be made even worse when management takes erroneous actions. For example, when management waits too long to communicate the problem and attempts "big" corrective actions, this usually leads to a project collapse.

Abdel-Hamid and Madnick studied Brooks' Law using their system dynamics model. Two important, but unrealistic, assumptions are made in their study. First, their model assumed that development tasks can be partitioned, but there is no sequential constraint among them. The development production rate depends solely on available manpower, not on sequential constraint. In reality, if tasks have to be done sequentially, then adding more people will not speed up the development process, since there are not enough tasks ready for them to work on. You expend people hours, but get little results [5]. The number of months of a project depends upon its sequential constraints. The maximum number of staff members depends upon the number of independent subtasks.

Another assumption is that project managers will continuously add new people as long as they sense a shortage in manpower. In reality, project managers can only add new people a few times throughout the entire project life cycle. The two unrealistic assumptions lead to their conclusion that "adding more people to a late project always causes it to become more costly but does not always cause it to be completed later" ([1], [2]). The increase in the cost of the project is caused by the increased training and communication overhead, which, in effect, decreases the productivity of the average team member and thus increases the project's man-day requirements. Only when the incurred training and communication overheads outweigh the increased productive manpower will the addition of new staff members translate into a later project completion time.

In this paper, we study Brooks' Law using more realistic assumptions. The sequential constraint of a software project is considered in our model. We also make an assumption that people are added into the project only once throughout the entire development life cycle, because it is not easy to obtain approvals from upper management to frequently add manpower to any project.

3. The Dynamics of Brooks' Law

The dynamics of Brooks' Law starts with management bringing new staff into a project. Three effects, as illustrated in Figure 1, are: (1) an increase of communication and training overhead, (2) an increase of the amount of work repartitioning, and (3) an increase of the total manpower available for project development. Note that a minus (plus) sign on an arrow means that the two entities connected by the arrow move in the opposite (same) direction.

When new staff are brought in, they require a certain level of training which will take away part of old staff's productive time. Also, more people require more communication. As a result, the total project manpower resource (man-days) also decreases. Less total project manpower means less manpower for development and decreased work rate. This results in project progress being delayed even further and leads to another round of people-hiring feedback loop.

The second effect of bringing in new people midway in the project occurs when work needs to be repartitioned. The work currently being performed by old staff needs to be repartitioned so some of it can be assigned to new staff. Project staff, both new and old, have to adapt to and learn new tasks. The coordination overhead is also increased, especially when the work is not well partitioned.

Another impact of bringing in new people is that more people are available to be assigned to the project. As a result, the work rate, as determined by the total number of project staff and the average staff productivity, also increases. An increase of work rate means work is being done at a faster pace, which eventually will catch up with planned progress. As a result, the degree of schedule slippage is reduced, which reduces the need to bring new people into the project.

As schedule pressure rises, part of the planned QA work might be skipped. As a result, the defects contained within the work product remain undetected, which results in defect amplification. Also, under extreme schedule pressure, project staff are prone to commit more defects than normal. The impact of an increased amount of defects is that part of the planned manpower for development now has to be devoted to defect correction. With less manpower available for development, the project is delayed even further, which causes the schedule pressure to rise and triggers another round of defect amplification "vicious cycle."

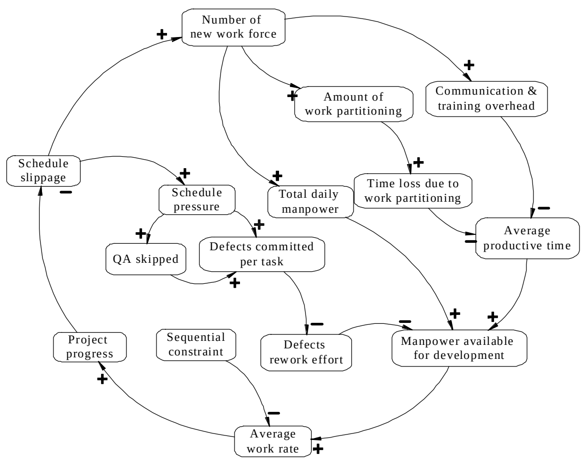


Figure 1. The dynamics of Brooks' Law.

4. Modeling Sequential Constraint

Unlike the Abdel-Hamid and Madnick (AHM) model, we take the sequential constraint of a software project into consideration in our model. Sequential constraint, as modeled as "degree of concurrency (DC)," is defined as the fraction of the number of tasks (including development and testing) that are ready to be worked on and the number of tasks project staff are able to perform. As shown in Figure 2, the number of tasks that project staff can perform is determined by multiplying "the amount of daily manpower allocated" by "staff's average productivity." For example, degree of concurrency = 0.8 means only 80% of the tasks that project staff are able to perform are ready for assignment. To simplify, we assume that the degree of concurrency remains unchanged before and after unplanned tasks are discovered. By changing the values of this parameter, we can examine the impact of different degrees of sequential constraint on project duration and cost.

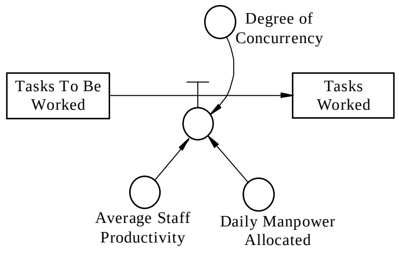


Figure 2. Modeling sequential constraint: work (development and testing) rate depend on sequential constraint.

5. Model Testing

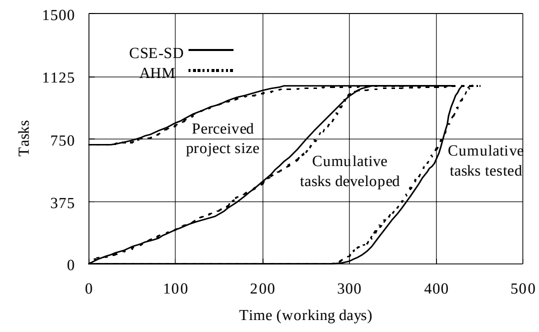
To place faith in simulation model-based analyses and policy recommendations, we have to know the degree to which those analyses might change as reasonable alternative assumptions are built into the model. First, we want to make sure our model (called Concurrent Software Engineering System Dynamics model, or CSE-SD model) produces similar behavior with minor variations in equation formulations and parameter values. Next, we want to know if the CSE-SD model is capable of generating project behavior similar to those reported in the literature. To conduct the test, we calibrate CSE-SD against the data reported in Abdel-Hamid and Madnick [2]. Our purpose is twofold: (1) to use their data and simulated result as a reference and (2) to compare our simulated results with theirs.

We compare seven key project measures, namely, perceived job size, perceived project cost, cumulative units developed, cumulative units tested, scheduled completion date, cumulative project cost, and work force distribution pattern. The comparison of these key project statistics is illustrated in Figures 3(a) to 3(d). Figure 3(a) displays three key project measures: perceived project size, cumulative tasks developed, and cumulative tasks tested. The "perceived project size" curve depicts the pattern of how the project scope was changed over time due to the discovery of unplanned development tasks. The real size of the project is 64 KLOC (1067 tasks), but initially was estimated to be 42.88 KLOC (715 tasks). The "cumulative tasks developed" and "cumulative tasks tested" curves show how the development tasks are completed and tested over time. The patterns of these two curves are very close to that of the AHM, especially in the first half (prior to day 220) of the "cumulative tasks developed" curve. However, after that, the CSE-SD simulated project progress is faster than that of the AHM, although the difference is not significant. The reason that causes the difference stems from the difference in work force. As indicated in Figure 3(d), after day 220, the CSE-SD has a higher work force level than that of the AHM. A larger work force means more tasks can be done within the same period of time.

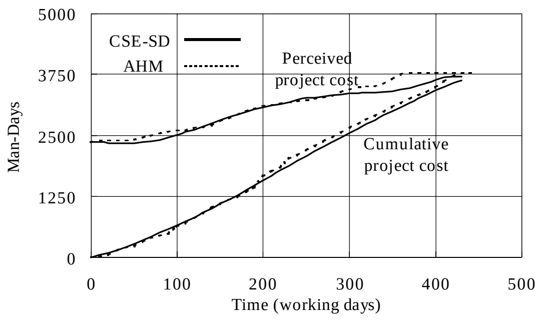
Figure 3(b) shows the cumulative project effort expenditure and the change in estimated project cost. Our simulated "cumulative project cost" curve is almost identical to that of the AHM prior to day 200. AHM produces a higher effort expenditure after day 200 because its work force curve reaches the peak earlier than CSE-SD. In fact, the AHM simulated project has more people on board than that of CSE-SD within the period of day 160 to 220. This explains why the project cost accumulated at a faster pace in AHM than in CSE-SD.

The "perceived project cost" curve shows how management adjust the estimated project cost as a result of the discovery of unplanned development tasks. Overall, the two simulated curves are similar before day 280. After that, the AHM curve displays an immediate uprise that is not seen in the CSE-SD curve. The reason for the difference lies in the difference in the project control mechanism. In CSE-SD, when new tasks are discovered, the adjustment of estimated project cost includes both the development cost and the system testing cost. The estimated project cost is adjusted far before conducting the system test phase. Therefore, we do not see any sharp change in the perception of the project cost right before conducting the system test. The other reason that causes the difference is that CSE-SD has a smaller project cost than that of the AHM (3686 man-days in CSE-SD as opposed to 3795 man-days in AHM).

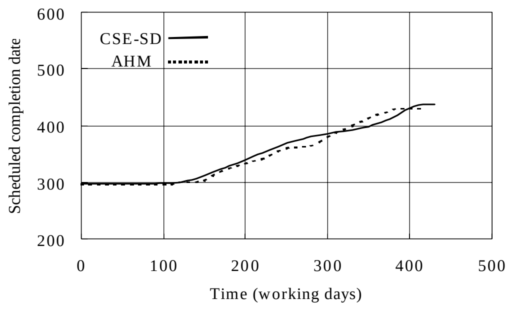
As illustrated in Figure 3(d), CSE-SD produces a Rayleigh-curve work force distribution pattern with a peak work force of around 11 project staff. The shape of the curve is very close to that of the AHM model. However, after around day 220, the CSE-SD curve deviates from that of the AHM. The CSE-SD work force curve reaches its peak at around day 220 and gradually tapers off to around eight staff on board at the end of the project. However, in AHM, the work force curve reaches its peak at around day 190 and gradually tapers off to around seven staff on board at the end of the project.



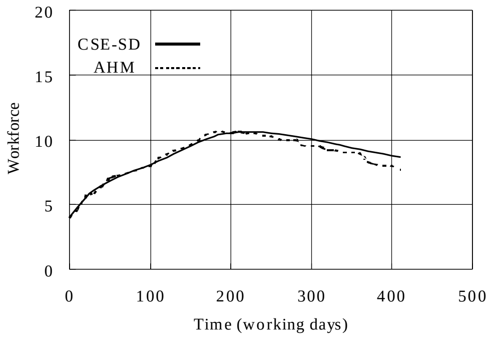
(a) Project progress



(b) Project cost



(c) Project scheduled completion time



(d) Work force distribution

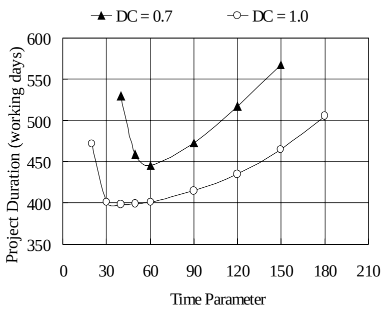
Figure 3. Comparison of key project statistics.

6. Simulation Results

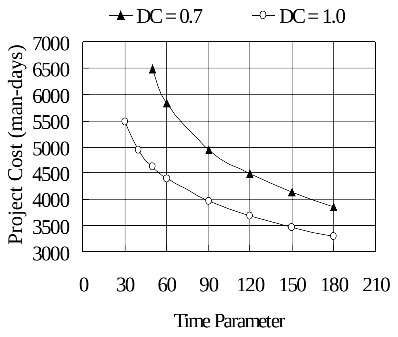
We present our simulation results with a focus on two questions: (1) What is the impact of adding people to a software project, in terms of project completion time and cost? (2) When is the best time to add people into a software project, and how many people should be added? We first address question 1: what is the impact of adding people to a software project, in terms of project completion time and cost? To answer this question and to compare our results with those of AHM, we use the same manpower addition assumption as theirs in this study, however, we add the sequential constraint factor to reveal its effect. We continue to add people as long as there is a shortage in manpower until a preset date. For example, as indicated in Figure 4(a), for a project without sequential constraint (i.e, DC = 1.0), if we continue to add people whenever we sense a shortage in manpower until 36 (i.e., 0.3 x 120) working days remaining in the planned project schedule, then the project is expected to complete at around day 435. However, after 180 working days remaining (i.e., 1.5 x 120), management is not 100% willing to hire enough people as desired. The total cost of the project is 3686 man-days, as shown in Figure 4(b).

As shown in Figure 4, a more aggressive manpower acquisition policy results in a shorter project duration, but increases project cost. We simulated different manpower acquisition policies by changing the value of Time Parameter (TP) and found out 398 working days is the shortest possible schedule one can achieve for this specific project. Time Parameter is defined as the sum of the time to hire new staff (hiring delay) and the time to train and assimilate new hires (assimilation delay). Our results indicate that Brooks' Law holds only when the Time Parameter is fewer than 40 working days for a medium-sized COCOMO organic-mode project. Our result is very similar to that of AHM.

The trend for the DC = 0.7 project (i.e., a project with a certain degree of sequential constraint) is similar to that of DC = 1.0 project; project duration continues to decrease when new work force is added. However, management pays the price of increasing project cost. For projects with certain degree of sequential constraint (i.e., DC = 0.7), Brooks' Law holds when the Time Parameter is less than 60 working days–about one month earlier than that of the DC = 1.0 project (40 working days). This implies that sequential constraint does play a role in this situation. If management fails to sense the shortage in manpower and does not make a timely decision to add work force, then the project will be delayed further, especially if there is a certain degree of sequential constraint among development tasks. Project cost continues to rise when new people are added, as shown in Figure 4(b). Project cost increase nonlinearly when Time Parameter is less than 90 working days. This implies that adopting a more aggressive manpower acquisition policy late in the project will pay a higher price.



(a) Project duration



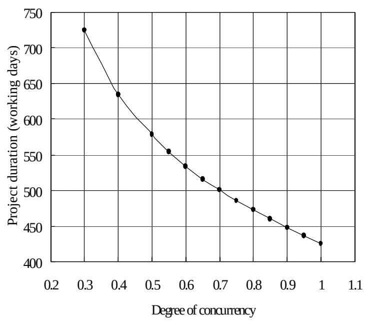
(b) Project cost

Figure 4. The impact of work force stability on project duration and cost.

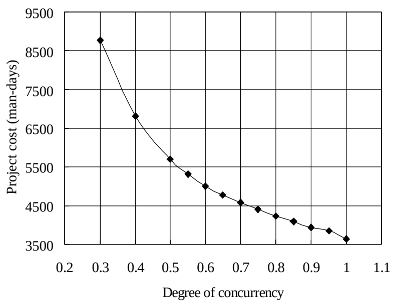
Figure 5 shows the impact of sequential constraint on project duration and cost. As expected, as the degree of sequential constraint increases (degree of concurrency decreases), project duration will increase, and so does the project cost. However, project duration and cost increase nonlinearly when DC is less then 0.5. The result indicates that a tighter sequential constraint has a stronger negative impact on project duration and cost.

We next address research question 2: what is the best time to add people into a software project and how many people should be added? Unlike the AHM model, we take the sequential constraint of a software project into consideration. Besides, to answer the question, we make a more realistic assumption that people are added into the project only once throughout the entire development life cycle. We conducted 24 simulation runs; 12 on projects with perfectly partitionable task (*PPT*) [3] and 12 on projects with a certain degree of sequential constraint. The results are summarized in Figure 6.

At the specified milestone date, the desired work force that is needed to complete the project on time is brought into the project. For example, at day 20 (one month after the project was launched), the desired new work force perceived needed to complete the project on time is 4.16 for *PPT* projects (i.e., degree of concurrency = 1.0). This is because, in the beginning of the project, there are only four engineers on board, while eight engineers are expected (i.e., 50% understaffed). Management will need to bring in the other four people as initially planned plus extra manpower to make up for the delayed work caused by having four engineers doing the work that is expected by eight engineers. In this specific organic-mode project, there is a threshold time *T*–about one-third (i.e., 140/472) of the development life cycle–before which adding people into a software project will not extend project duration. However, after the threshold time, adding people to the project will cause the scheduled completion date to extend.



(a) Project duration

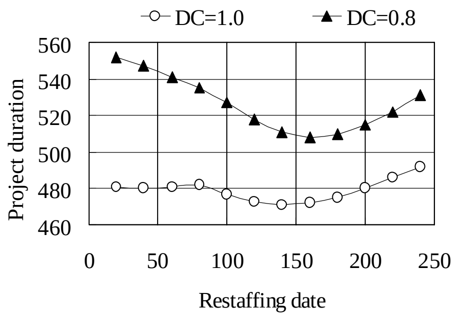


(b) Project cost

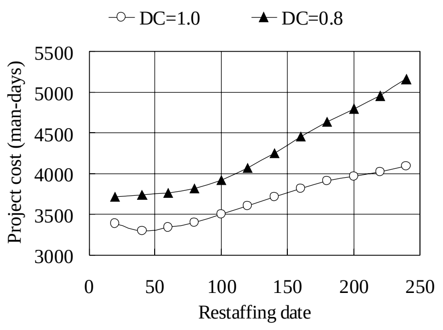
Figure 5. The impact of degree of concurrency on project duration and cost.

The DC = 0.8 project also has a threshold time at day 160; one month (20 working days) later than that of the DC = 1.0 project. However, it is also at about one-third (i.e., 160/508) of the entire development life cycle. After simulating projects with different degrees of concurrency (from DC = 0.5 to 1.0), we found that the threshold time will shift forward as the degree of concurrency increases. As shown in Figure 6(b), adding people into a software project will, in general, cause the project cost to increase.

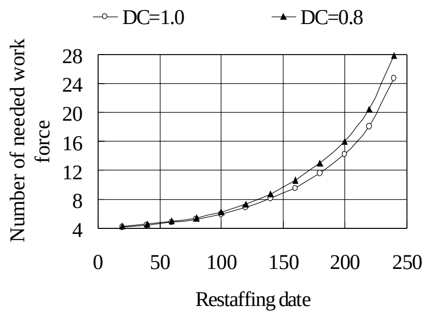
There could be numerous alternatives between the two extreme manpower acquisition policies we use in our simulation runs, namely, continuous manpower acquisition policy and onetime manpower acquisition policy. The outcomes of adopting different manpower acquisition policies are expected to fall between our simulated results. For organic-mode projects, we predict a project schedule-effective time range from one-third of the project to halfway into the project life cycle.



(a) Project duration



(b) Project cost



(c) Number of needed new work force

Figure 6. Impact of restaffing time on project duration, cost, and number of needed work force.

7. Conclusions

We performed an in-depth study of Brooks' Law using a system dynamics simulation approach. The results of the study are based on three sets of simulation runs with different assumptions. First, we use the same assumptions as those of AHM: (1) project tasks can be partitioned, but there is no sequential constraint among them; and (2) management will continuously add new people as long as they sense a shortage in manpower. Under these assumptions, our results are consistent with those of AHM, namely, adding more people to a late project always causes it to become more costly but does not always cause it to be completed later.

Next, we use a more realistic assumption by considering sequential constraint. We found out that continuously adding people to a late project makes it later and more costly. This confirms Brooks' Law. However, these results are not consistent with those of AHM's. This implies that sequential constraint does play a role in project development.

Finally, we add another realistic assumption that people are added to a project only once throughout the entire project life cycle because it is difficult to obtain frequent manpower addition approvals from upper management. We found out that there is an optimal time range for adding people without delaying a project. It ranges from one-third to halfway into the project development. If software project managers cannot make a timely and accurate decision on project restaffing prior to halfway into the project, the project has a high probability of being delayed, especially when task sequential constraints are involved. However, adding people during the project always causes the project cost to increase.

In summary, it is always costly to add people to a late project. When sequential constraint is significant, adding people late in a project will make it later. We have also found an optimal time range for adding people without delaying a project in this study.

8. References

[1] T. K. Abdel-Hamid, "The Dynamics of Software Project Staffing: a System Dynamics Based Simulation Approach," *IEEE Transactions on Software Engineering*, Vol. 15, No. 2, February 1989, pp. 109-119.

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

[3] F. P. Brooks, Jr., *The Mythical Man-Month*, Addison-Wesley, 1995.

[4] R. L. Gordon and J. C. Lamb, "A Close Look at Brooks' Law," *Datamation*, June 977, pp. 81-86.

[5] L. H. Putnam and W. Myers, *Industrial Strength Software: Effective Management Using Measurement*, IEEE Computer Society Press, Los Alamitos, CA, 1997.

[6] G. M. Weinberg, *Quality Software Management: Volume 1, system thinking*, Dorset House Publishing, 1992.