

Dynamics of Schedule Pressure in Software Projects

Veerendra K. Rai* & ⁺Biswajit Mahanty

*Tata Consultancy Services
54B, Hadapsar Industrial Estate
Pune- 411 013, India
Phone: 91 20 6871058, Fax: 91 20 6810921
E-mail: vkrai@pune.tcs.co.in

Department of Industrial Engineering and Management
Indian Institute of Technology, Kharagpur- 721302
India

Abstract

This paper studies the dynamics of schedule pressure in software projects. It constructs a system dynamics model of the software development process, which includes the following modules. Flow of Work, Work Being Done, Inspection, Fault Detection and Rework, and Supplementary Variables. It includes the module 'Effect of Schedule Pressure' and integrates it with the rest of the model. The model presents the results of two sets of schedule-related policies of a system dynamics model. The first set of policies that pertain to the base model does not consider the effect of schedule pressure on the staff productivity and errorproneness while the second set of policies pertaining to the revised model does. This study finds that effect of schedule pressure on software project is non-linear and dynamics of schedule pressure is intricately related to decision making process of the project manager, thus, making software projects management complex and counterintuitive.

Key words: *Schedule pressure, software projects, productivity, errorproneness, and decision making*

1. Introduction

Software development is a manpower intensive problem solving process. Elements normally considered important for successful execution of software projects are- human resource, software production, inspection and detection of software with respect to quality objectives and finally, management. The common objective in all these considerations is to produce quality software at optimal cost. Quality here refers not only to lack of errors and bugs, but the extent to which software meets client's requirements. Equally important is the cost incurred on the development of the software. The most important factor that determines the cost of the software is the amount of effort that has gone into producing the software. It is precisely for reducing the effort spent and increasing the productivity that various technical and managerial measures are taken, such as, CASE tools, auto code generation, distributed computing etc.

Software project development environment often incurs the wrath of schedule pressure. Schedule pressure is brought about by a variety of factors pertaining to above-mentioned factors. Loss of experience due to attrition, amount of rework due to error commitment, lagging behind the scheduled project completion time due to low productivity or rework or both, technological changes, discrepancy in estimation of size of the project, effort required and others. This schedule pressure in turn affects productivity, errorproneness, and other behavioral issues affecting a software development team.

These are standard sources of schedule pressure. There are some other sources of schedule pressure, which are often unnoticed and therefore are not recorded formally. One of them is the time spent on quality considerations particularly in the light of ISO (International Standard Organization) and SEI, CMM (Software Engineering Institute's Capability Maturity Model). Organizations adhering to these standards spent lot of time towards these. No less than 20% of project completion time is spent on quality, which is more often not taken into account at the time of effort estimation for the projects. Rarely, project managers allocate additional time for quality considerations. As a result projects lag behind their schedule and schedule pressure starts mounting on them. The other source of schedule pressure is even more unaccounted for, and it is rather called for consideration now. During the software boom in the decade of 1990s software companies expanded and registered phenomenal growth. Under pressure to perform better than their competitors (in terms of revenue generation) software companies started taking projects any which way without any regard to whether or not they can deliver the project on time. In such scenario one of the two had to happen. Either software quality suffered or project got delayed leading to schedule pressure. Schedule pressure is not a fancy word it is a reality.

Studies on impact of schedule pressure can be found in (Mills, 1983; Abdel-Hamid 1989; 1993; Abdel-Hamid and Madnick, 1986; 1991; Putnam, 1978). People under time pressure do not work better they just work faster. In the struggle to deliver any software at all, the first casualty has been the consideration of the quality of the software delivered (DeMarco, 1982). Shneiderman (1980) suggests that schedule pressure increases the 'anxiety levels' of programmers. A high anxiety level interferes with performance, probably by reducing the size of the short-term memory available. When programmers become more anxious with the approach of the deadline, they tend to make more errors.

Thibodeau and Dodson (1980) suggest that schedule pressure often results in the overlapping of activities that would have been accomplished better sequentially and overlapping can significantly increase the chance of errors.

1.1 The Dynamics of Schedule Pressure

Software development is a manpower intensive problem solving process carried out by a group of individuals with a common and sometimes, competing goal (Waterson et al., 1997) and hence involves multiple agents' interaction (Curtis et al., 1987). Out of this interaction among the members of the team working on a software project, emerge certain parameters or variables, pertaining to behavioural and organisational issues, which affect productivity, error commitment rate, motivation, turnover, and overall successful completion of a software project. How these parameters/variables are related to each other and how they dynamically affect each other in the course of software project development, is the issue that must be addressed in modelling the software

development process. Furthermore, software projects are prone to encounter ‘stress situations’ that are brought about by schedule and cost overrun, volatility of customer requirements, loss of experience due to turnover, and rework. This stress situation, in turn, affects the productivity and errorproneness of staff members, and overall software development process. This, missing causal link, from ‘stress’ to productivity and errorproneness, which completes the loop, has been specifically explored in this study. A simplified view of schedule pressure yields the following scenario given in Figure 1.

1.2 The Use of System Dynamics

The system dynamicists most effectively captured the informational representations of software process modeling. A thorough set of investigations of system dynamics modeling of software development process can be found in the works of Abdel-Hamid and his colleagues (Abdel-Hamid and Madnick, 1986; Abdel-Hamid, 1989; Abdel-Hamid, Sengupta, and Hardebeck, 1994; Abdel-Hamid, Sengupta, and Ronan, 1993). Software project management with system dynamics approach can also be found in the works of Mohapatra and Mahanty (1993) and Rai (1998). A platform for modeling multi-agent systems with system dynamics has been articulated by Kim and Juhn (1997).

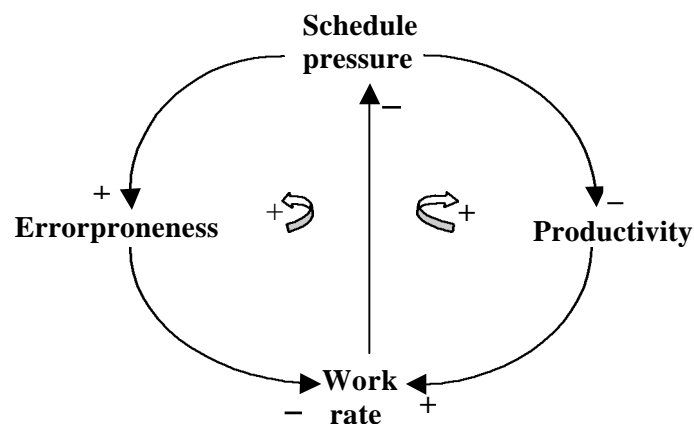


Figure 1: A Simplified View of the Effects of Schedule Pressure

Software environments are complex, opaque and dynamic. In addition to that, software environments are also characterised by information that is unreliable with respect to the initial estimate as well as outcome feedback available over the life of the project (Sengupta and Abdel-Hamid, 1996). The unreliability in initial estimate is due to the inability of extant forecasting techniques to provide accurate projections (Kemerer, 1987). The unreliability in outcome feedback results from difficulty in assessing how much of the work has been completed during a particular time period, especially for intermediate products such as design specifications and undebugged programs (Mills, 1983). This inherent lack of accuracy in measurement is further compounded by the tendency of the project staff to report favourable information (which is relatively unimportant for control purposes) and to withhold unfavourable information (which is critical for taking corrective action). Software environments are autonomous, for, they change autonomously in response to changes in environment (Sengupta and Abdel-

Hamid, 1996). Information feedback, therefore, acquires crucial importance in the decision making process.

1.3 Aim of the Paper

In this paper, we present a system dynamics model in order to obtain the dynamic behavior of a representative software development project. Two versions of the dynamic model are simulated for varying values of the schedule project completion time. The first one is the **base model** that considers constant values of errorproneness and productivity of the software project developers throughout the project. The other one is the **revised model** that takes into account changing values of errorproneness and productivity of the software project developers based on the schedule pressure.

1.4 The Software Project

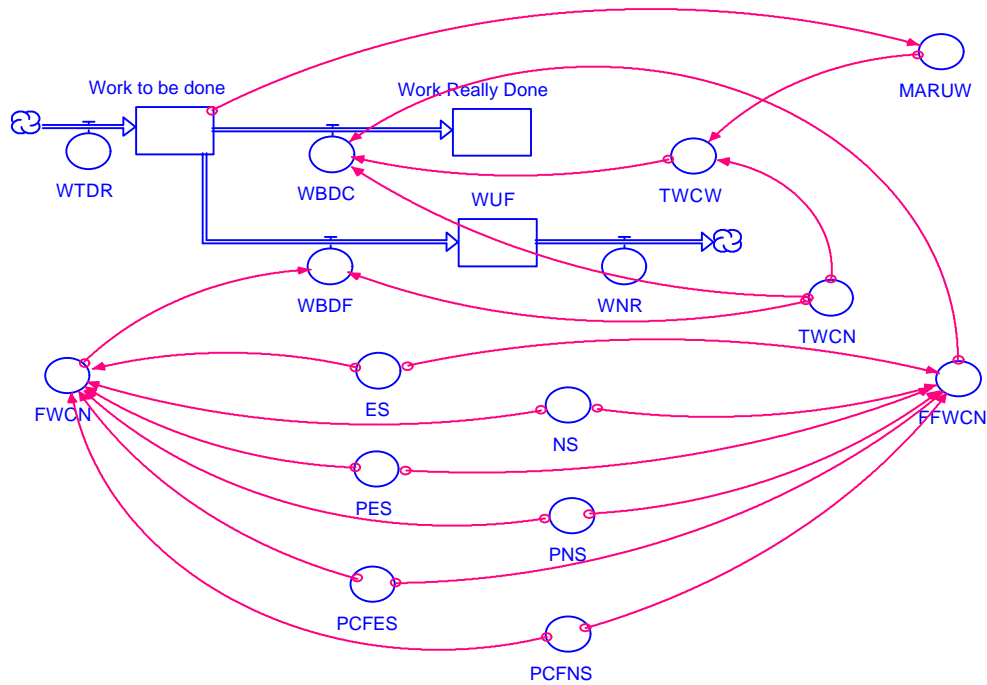
A hypothetical software project involving about 400 tasks (function points) is considered in this paper. The scheduled project completion time (SPCT) is taken as 50 weeks and it is assumed that 2 experienced and 2 new software development persons would carry out the software project. The distinction between new and experienced staff is necessary because new team members are usually less productive and more error-prone compared to experienced ones. It is assumed that the productivity of experienced software personnel will be 3 tasks/person/week while that of new software personnel will be 2 tasks/person/week. The errorproneness values for them will be 20% for experienced personnel and 40% for the new ones.

2.0 Development of the System Dynamics Model

Two versions of a system dynamic model are developed which are simulated for varying values of the schedule project completion time. The first one is the **base model** that considers constant values of errorproneness and productivity of the software project developers throughout the project. The other one is the **revised model** that takes into account changing values of errorproneness and productivity of the software project developers based on the schedule pressure.

Traditionally, a task of a project is a discrete chunk of work. All works done are not correct. Some of them are done correctly (Work Really Done, WRD) and the rest are done with faults (Work with Undetected Faults, WUF). The corresponding rate variables are (1) Work Being Done Correctly (WBDC), and (2) Work Being Done with Faults (WBDF). The faulty works remain undetected till inspection is carried out. Some of the work with undetected faults gets discovered by inspection and it, subsequently, goes for rework identified here as 'Work Needing Rework' (WNR). It is presumed that some of the faulty work will be discovered in the process of inspection, and the resulting rework will add to Work to be done (W). This additional work is represented by the rate: 'Work To be Done due to Rework' (WTDR).

Work being done depends on staff size, its productivity, and the average quality of task accomplishment. Here, the quality of task accomplishments depends upon the Fault Free Work Completed Normally (FFWCN) as well as Faulty Work Completed Normally (FWCN).

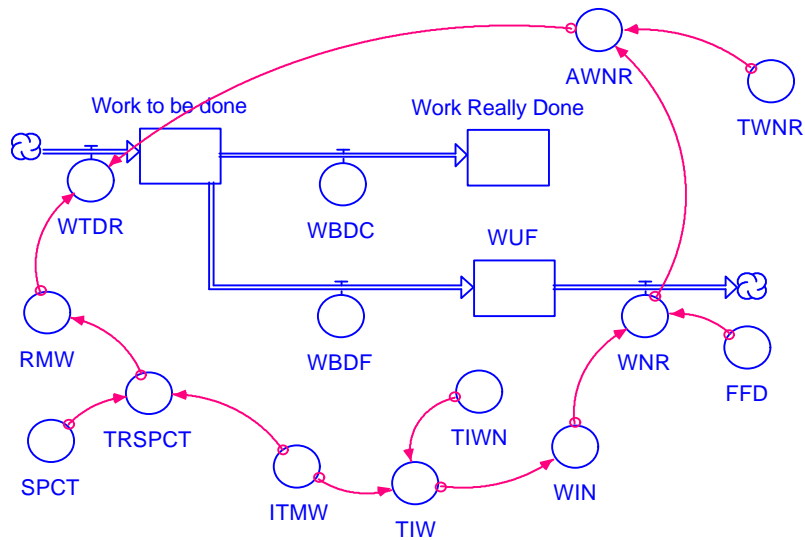


Abbreviations:	
WTDR: Work To be Done due to Rework	WBDC: Work Being Done Correctly
WBDF: Work Being Done with Fault	WUF: Work with Undetected Fault
MARUW: Maximum Allowable Rate of Undertaking Work	
WNR: Work Needing Rework	TWCW: Total Work Completed per Week
TWCN: Total Work Completed Normally	FWCN: Faulty Work Completed Normally
FFWCN: Fault Free Work Completed Normally	ES: Experienced Staff
PES: Productivity of Experienced Staff	NS: New Staff
PCFES: Propensity of Fault Commitment for Experienced Staff	PNS: Productivity of New Staff
PCFNS: Propensity of Fault Commitment for New Staff	

Figure 2: Work Being Done

Figure 3 shows the dynamics of inspection, detection and rework. As software is developed it is periodically reviewed by quality assurance activities such as structured walkthroughs. This review is conducted by the team which developed it as well as quality assurance experts. As the project tasks are carried out not all work is done correctly. Work done is inspected to detect fault, if any. Not all faults, however, are discovered in a single cycle of inspection and detection. Some faults may evade inspection and detection. Tasks with detected faults are to be reworked. A single task to be reworked may actually result in more than one tasks to be redone again. It is also assumed that as time passes and project nears the scheduled project completion time, the amount of additional work to be done actually decreases for a given amount of rework tasks. This additional work is added to 'Work to be done'. The distinction between work and rework is artificial. Once

the nature and extent of rework is determined, the additional work due to rework becomes a part of work to be done.



Abbreviations:	
WTDR: Work To be Done due to Rework	WBDC: Work Being Done Correctly
WBDF: Work Being Done with Fault	WUF: Work with Undetected Fault
WNR: Work Needing Rework	WIN: Work being Inspected
AWN: Average Work Needing Rework	TWNR: Time to smooth WNR
FFD: Fraction of Faults Detected	RMW: Rework Multiplier from Work
ITMW: Inspection Time Multiplier from Work	TIW: Time to Inspect Work
SPCT: Schedule Project Completion Time	TIWN: Time to Inspect Work Normal
TRSPCT: Time Relative to Schedule Project Completion Time	

Figure 3: Inspection, Fault Detection and Rework

Time to Inspect Work (TIW) depends upon the stage and nature of work being inspected. Inspection time is not uniform across all the stages of software development life cycle. If work being inspected is design work, it is assumed to consume more time compared to inspection of coding work. 'Inspection Time Multiplier from Work' (ITMW) is thus a table function of 'Time Relative to Schedule Project Completion Time' (TRSPCT).

TRSPCT	0.0	0.25	0.50	0.75	1.0
ITMW	5.0	4.0	3.0	2.0	1.0

It is pertinent to mention here that inspection, detection and correction of faults are the most perennial problems in the software development process which, more often than not, are present throughout the software life cycle. Once the faults are detected as a result of inspection of work, it takes time to decide the extent of work to be carried out due to rework. This time delay is introduced by smoothing the 'Work Needing Rework' (WNR) to Average Work Needing Rework (AWN):

As explained earlier, every fault does not lead to equal amount of rework generated due to it. It depends upon the nature of fault. If it is a design fault, it will generate more rework than if it is a coding fault. Therefore, decision with respect to 'Work To be Done due to Rework' (WTDR) must take into account the 'Work Needing Rework' (WNR) and 'Rework Multiplier from Work' (RMW):

More additional work due to rework is generated in the initial stages of the project. As the time passes and the project comes closer to schedule project completion time additional work due to rework is less and less. RMW is thus a table function of TRSPCT as given below:

TRSPCT	0.0	0.25	0.50	0.75	1.0
RMW	5.0	4.0	3.0	2.0	1.0

Software projects are opaque and suffer from poor visibility. Estimation of work is the first managerial task in software project management. Estimation is a continuous process and at every stage the estimation of work is updated, rectified and decision is made for the future course of action. Variables such as, 'Extent of Rework' (EOR), 'Extent of Latent Error' (EOL), 'Percentage of Work Really Done' (PWRD), and 'Percentage of Work Perceived Done' (PWPD), are supplementary variables that measure the project performance. They do not influence the dynamics of the project directly, nonetheless, they do influence the decision-making process of the manager which, in turn, influences the course of the project. The perceptions of the manager interact with the dynamics of the project duly affecting each other. Figure 5 shows the derivation of the performance variables.

During the initial stages of project the fraction of correcting wrong work estimation is almost non-existent, since not enough progress has been made to re-evaluate the estimation, but it increases as the project proceeds. Therefore, 'Fraction of Correcting wrong work estimation with Time' (FCT) takes into account the time relative to scheduled project completion time as given by the following table function:

TRSPCT	0.0	0.25	0.50	0.75	1.0
FCT	0.0	0.0	0.50	1.0	1.0

The Estimated Work content of the project (ESTW) at any point of time takes into account the initial estimation of work (ESTWI), the discrepancy between the 'actual Work Content of the project' (WC) and ESTWI, and FCT. The 'actual Work Content of a project' (WC) is the sum of 'Work to be done' (W), 'Work Really Done' (WRD) and 'Work with Undetected Fault' (WUF).

Percentage of Work Really Done (PWRD) takes into account the 'actual Work Content of the project' (WC) and the level of 'Work Really Done' (WRD). 'Work Perceived Done' (WPD), however, is the sum of 'Work with Undetected Fault' (WUF) and 'Work Really Done' (WRD). It is pertinent to mention here that WUF is also perceived as work done by

the staff as it consumes valuable production time and other resources. However, WUF determines the extent of accuracy of the estimation process. Therefore, Percentage of Work Perceived Done (PWPD) can be expressed as a fraction of Work Perceived Done (WPD) and Estimated Work content of the project (ESTW).

The transformation of work to be done into work done is a conserved process. However, at an instant, the 'actual Work Content of the project' (WC) is usually more than the 'Work Initial' (WI)! The excess work is actually a measure of Extent of Rework (EOR).

As explained earlier, the inspection and detection process discovers faults lying in WUF. However, the extent to which such faults are *not* discovered (and thus not reworked) is an important indicator of the software quality. At an instant (also at the end of the project), the ratio of WUF and WI is defined as the Extent of Latent error (EOL).

2.5 Effect of Schedule Pressure

It is interesting to note that the consideration of the effect of schedule pressure on productivity and errorproneness of the software development staff completes a number of control feedback loops. So far, we have considered the effect of productivity and errorproneness on the progress of the software project. It is well known that the progress of the software project, measured against the planned dates, exerts schedule pressure on the software development staff (Smith, Nguyen, and Vidale, 1993; Abdel-Hamid, 1993). Therefore, when the effect of schedule pressure on productivity and errorproneness is considered, the feedback loops are completed, and the resulting system may show certain counter-intuitive behaviour (Sengupta and Abdel-Hamid, 1996).

Schedule pressure, in the revised system dynamics model, has been defined in terms of the 'Time Ratio' (TR). It is assumed that, as time progresses, the project manager would continually calculate the time required completing the remaining work. 'Time Ratio' (TR) can be defined as a ratio of 'scheduled Time Left for the project' (TLEFT) and 'estimated Time Required to complete the project' (TREQ). If, the software project continues beyond 'Scheduled Project Completion Time' (SPCT), TLEFT is taken as zero. TREQ would depend on 'Work to be done' (W) and the estimated rate at which fault free work is completed (FFWCE). Based on the definition of the Time Ratio (TR), the 'Schedule Pressure' (SP) can be defined as given below on a three-point scale:

Condition	$TR > 0.75$	$0.25 < TR < 0.75$	$TR < 0.25$
SP	Low (1)	Medium (2)	High (3)

The effect of schedule pressure on productivity and errorproneness is an important topic of research. We present the following values of productivity and errorproneness as a function of schedule pressure in Table-1 below:

Table -1: Effect of Schedule Pressure on Productivity and Errorproneness of Staff

Productivity and Errorproneness		Low Schedule Pressure (2)	Medium Schedule Pressure (3)	High Schedule Pressure (4)
Productivity (Tasks per week per person)	For Exp. Staff (PES)	High 3.0	Medium to Low 3.0	Medium to Low 2.5
	For New Staff (PNS)	High 2.0	Medium to Low 2.0	Medium to Low 1.5
Errorproneness (Dimensionless)	For Exp. Staff (PCFES)	Low 0.15 or 15%	Medium 0.2 or 20%	Medium 0.2 or 20%
	For New Staff (PCFNS)	Low 0.25 or 25%	Medium 0.4 or 40%	Medium 0.4 or 40%

Two important control feedback loops involving schedule pressure are shown in Figure 4.

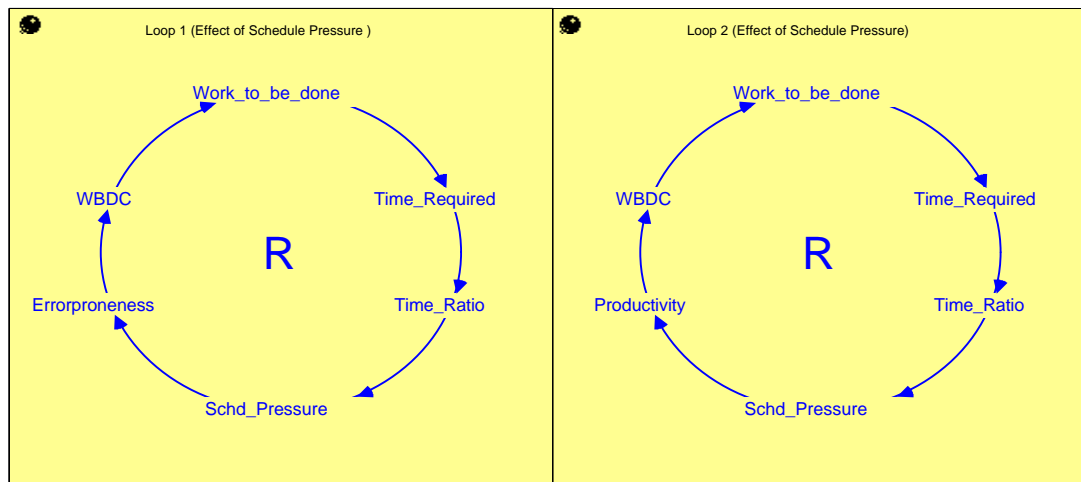


Figure 4: Two Control Feedback Loops of the Effect of Schedule Pressure

It is interesting to note that the feedback loops are not always reinforcing because the relationship between schedule pressure and productivity and also that between schedule pressure and errorproneness are non-linear. This makes the problem of managing the software complex and counterintuitive: and an attempt to control the project may become difficult.

3.0 System Dynamics Simulation Runs

The system dynamics model is simulated with the help of STELLA 5.0 software with a DT value of 0.25. The model was simulated for 100 weeks. The project completion time

was obtained with a consideration of perceived completion of 99.85% of work. Two sets of schedule-related policies of the system dynamics model have been considered. The first set of policies pertaining to the base model does not consider the effect of schedule pressure and hence take constant values of productivity and errorproneeness. The second set of policies for the revised model considers the effect of schedule pressure and thus have varying values of productivity and errorproneeness as presented in Table 1. For each set, three policies are considered with respect to scheduled project completion time of the project. For all the policies, it is assumed that the project is carried out by two experienced and two new software personnel. Nobody leaves or joins the project in between. The policies are:

1. **SPCT = 40 weeks.** Optimistic estimation.
2. **SPCT = 50 weeks.** Normal estimation
3. **SPCT = 60 weeks.** Pessimistic estimation.

3.1 Base Model Policy Runs with Constant Productivity and Errorproneeness

Figure -5 shows the performance variables, namely PWRD, PWPD, EOR, and EOL, for the policy with SPCT = 50 weeks.

As may be seen, 'Percentage of Work Really Done' (PWRD) lags 'Percentage of Work Perceived Done' (PWPD) in the initial period (by about 10 to 14 per cent by the 25th week). The difference reduces afterwards (50th week) to about 3-4 percent, and beyond SPCT (i.e. 50th week) the gap is small (about 2 per cent). In all the plots, EOL rises initially to about 9 to 10 percent by the 25th week, and thereafter it falls to 3-4 percent. Extent of Rework (EOR), a measure of the increase in the volume of work over its initial value, rises rapidly almost up to the SPCT, and then stabilizes.

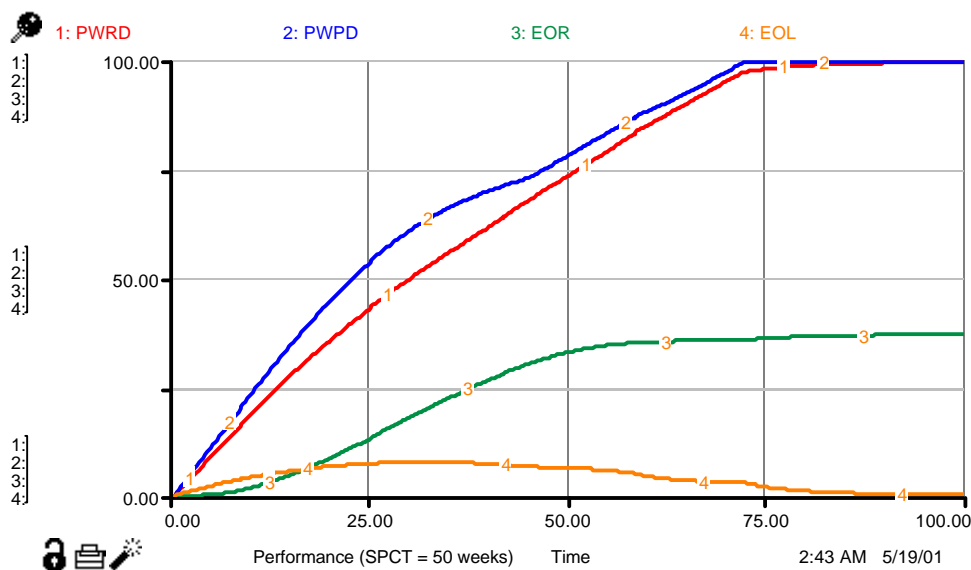


Figure 5: Performance Variables of Base Model (SPCT = 50 weeks)

Figure 6 compares the three policies with respect to work to be done (W) for the base model.

In all the cases, the project could not be completed within the scheduled project completion time. The best results are obtained for the optimistic schedule project completion time, i.e. when SPCT = 40 weeks. Project completion time here is 68 weeks. The worst results are obtained for the pessimistic SPCT, i.e. when SPCT = 60 weeks. The



Figure 6: Comparison of Work to be done for the Base Model Policies

normal policy run (with SPCT = 50 weeks) shows that the project is completed in 74 weeks. Table 2 shows some of the important results of the base model.

Table 2: Schedule Policy Runs at Project Completion Time (PCT)

Schedule Policy	PCT (weeks)	PWRD (%)	PWPD (%)	WC (tasks)	EOR (%)	WUF (tasks)	EOL (%)
SPCT=40	68.00	97.48	99.86	498	24.41	12.25	3.06
SPCT=50	74.00	97.51	99.87	543	35.74	12.80	3.20
SPCT=60	81.00	97.80	99.88	591	47.80	12.28	3.07

Note: WC: Total Work Content of the Project; WUF: Work with Undetected Faults
 WI : Initial Work Content of the Project: 400 Tasks.

It is interesting to note that the EOR values are widely different for the different policies. It is 24.41 per cent for the optimistic policy, 35.74 per cent for the normal policy, and as high as 47.80 per cent for the pessimistic policy. The actual Work Content of the project (WC) is also different for the different policy runs. It is 498 tasks for the optimistic policy, 543 tasks for the normal policy, and as high as 591 tasks for the pessimistic policy.

What could be the reasons for the wide differences between the policy runs? It is interesting to note that the base model considers both ITMW (Inspection Time Multiplier from Work) and RMW (Rework Multiplier from Work) as a function of TRSPCT (Time Relative to SPCT). Thus when SPCT is lower, inspection is faster, and the extent of rework is lower (as per the assumed table function values). The fundamental assumption here is that if schedule pressure is increased, the project progresses faster. The simulation results compare well with this assumption. Lower the SPCT, better is the performance: less project completion time and less rework (hence lower WC values). These results, although sound logical, are actually flawed as will be discovered later.

The base model policy runs produce expected results. The project completion time increases with increase in SPCT (i.e. 68 weeks to 81 weeks). The extent of rework also increases with the increase in SPCT, varying from 24 % to 48%. However, the extent of latent error remains around 3 percent for all the three policy runs. Also, the discrepancy between work perceived done and work really done is little over 2% for all instances of schedule policy. The base model is a simplistic one. The results obtained here are expected, but not realistic.

3.2 Revised Model Policy Runs with Varying Values of Productivity and Errorproneness

This section discusses the results of the revised model that considers the impact of schedule pressure on productivity and errorproneness of staff. The same policies, as have been taken for implementation in the base model, are considered for the revised model as well. Figure 7 shows, for the policy with SPCT = 50 weeks, the schedule pressure as it rises from low (SP = 1) to medium (SP = 2) to high (SP = 3) with the Time Ratio (TR) falling steadily from a high value near 1 to the 0 value. The schedule pressure rises somewhere between the 30th and the 45th week. As expected, the productivity and the errorproneness values would also change in accordance with the changes in the schedule pressure (Refer Table 1).

Figure 8 shows the performance variables namely PWRD, PWPD, EOR, and EOL of the revised model with normal values of the SPCT (SPCT = 50 weeks).

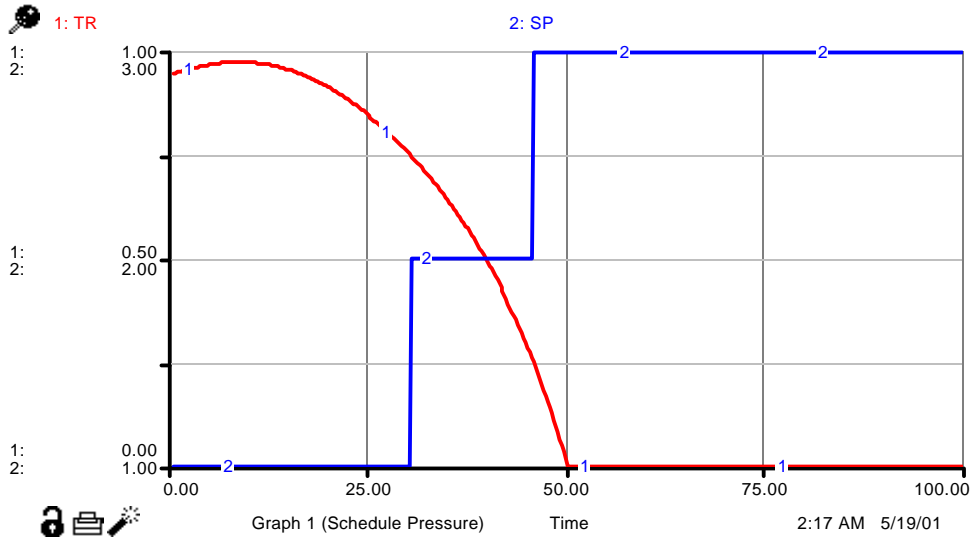


Figure 7: Changes in Schedule Pressure with the Time Ratio

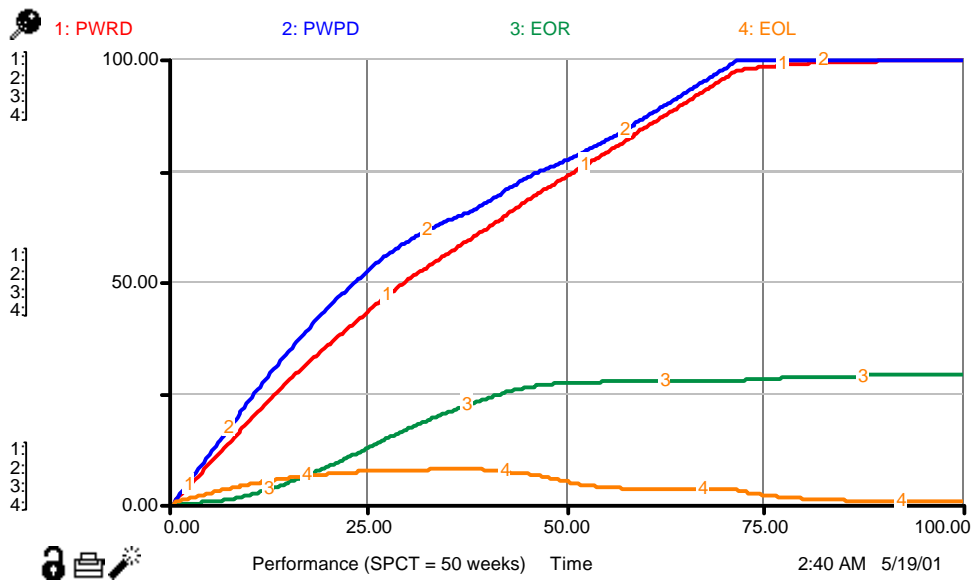


Figure 8: Performance Variables of Revised Model (SPCT = 50 weeks)

Figure 9 compares the three policies with respect to Work to be done (W).

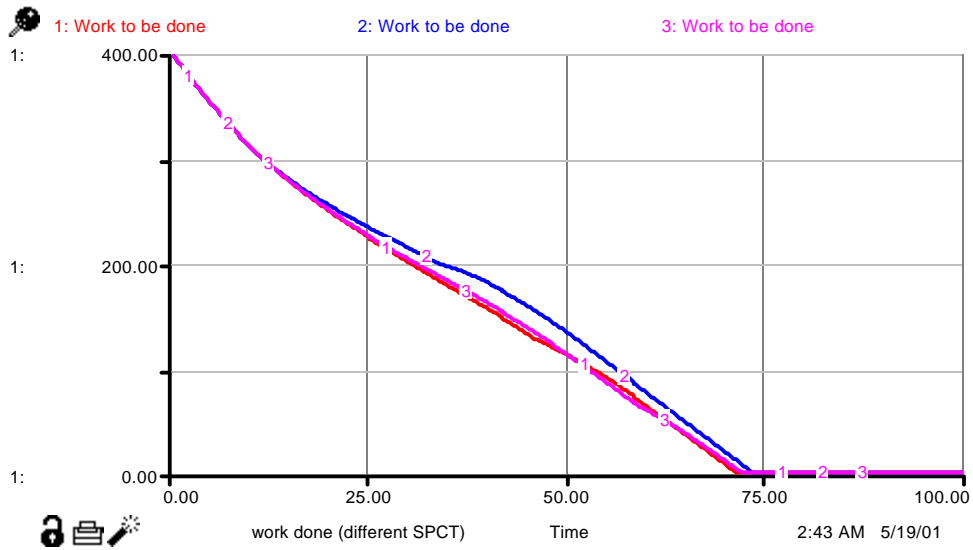


Figure 9: Comparison of Work to be Done for the Revised Model Policies

In all the cases, the project could not be completed within the scheduled project completion time. The results for all the three policies are more or less similar. The best results are obtained, contrary to the expectation, for the normal scheduled project completion time, i.e. when SPCT = 50 weeks! Project completion time here is 72 weeks. For the optimistic SPCT i.e. SPCT = 40 weeks, the project is completed in 74 weeks. It is interesting to note that the project is completed in 73 weeks for the pessimistic SPCT, i.e. when SPCT = 60 weeks. It must be mentioned here that, with changes in parametric values (such as productivity and error-proneness), the preference orders may vary to some extent.

Table 4 shows some of the important results of the revised model.

Table 4: Schedule Policy Runs at Project Completion Time

Schedule Policy	PCT (weeks)	PWRD (%)	PWPD (%)	WC (tasks)	EOR (%)	WUF (tasks)	EOL (%)
SPCT=40	74.00	97.82	99.89	489	22.29	10.09	2.52
SPCT=50	72.00	97.91	99.89	511	27.70	10.10	2.53
SPCT=60	73.00	98.05	99.89	544	35.98	10.01	2.50

In all the cases, the ‘Percentage of Work Really Done’ (PWRD) lags the ‘Percentage of Work Perceived Done’ (PWPD) in the initial period (by about 10 per cent by the 25th week). The difference reduces afterwards (50th week) to about 3-4 percent, and beyond SPCT (i.e. 50th week) the gap is small (about 2 per cent). In all the plots, EOL rises initially to about 7 to 8 percent by the 25th week, and thereafter it continues to fall to about 2-3 percent at the end of the project. Extent of Rework (EOR), a measure of the increase in the volume of work over its initial value, rises rapidly almost up to the SPCT (to about 27 per cent), and then rises rather slowly. It is interesting to note that the stabilised EOR values are widely different for the different policies. It is 27.70 per cent for the normal policies, 22.29 per cent for the optimistic policy, and as high as 35.98 per cent for the pessimistic policy. The total actual Work Content of the project (WC) is also different for the different policy runs. It is 511 tasks for the normal policies, 489 tasks for the optimistic policy, and as high as 544 tasks for the pessimistic policy. The reasons for the wide differences among the policies are the same as explained in the base system dynamics model discussions. When SPCT is lower, inspection is faster, and the extent of rework is lower.

There is a significant departure of model behavior in the revised system dynamics model from the base model. The software project is completed earlier in the case of higher SPCT (in 73 weeks). This is because of comparatively lower schedule pressure and hence higher productivity and lower errorprone values in the policy with SPCT = 60 weeks.

3.3 Comparison of Revised Model Results with that of the Base Model

As explained earlier, the base model assumes constant values of schedule pressure, productivity and errorprone of the staff. The revised model, on the other hand, considers, the effect of schedule pressure on the productivity and errorprone of the staff. This leads to significant differences in the model behaviour between the base and revised system dynamics models. Table 6 compares the base and the revised system dynamics model results for the three policies at the respective values of the project completion time.

Table 6: Comparison of Base and Revised Model Results at PCT

Schedule d Policy	PCT (weeks)		EOR (%)		EOL (%)	
	Base Model	Revised Model	Base Model	Revised Model	Base Model	Revised Model
SPCT=40	68.00	74.00	24.41	22.29	3.06	2.52
SPCT=50	74.00	72.00	35.74	27.70	3.20	2.53
SPCT=60	81.00	73.00	47.80	35.98	3.07	2.50

Conclusion

This study has taken two variants of system dynamics model- a base model and a revised one. While base model does not consider the effect of schedule pressure on projects the revised model does. The revised model produces realistic results in accordance with the considerations incorporated into it. The base system dynamics model shows increase in the project completion time (PCT) with increase in schedule project completion time (SPCT), the revised model shows substantially different results. In this case, PCT actually changes little with increase in SPCT (i.e. from 74 weeks to 72-73 weeks as SPCT increases from 40 weeks to 60 weeks). This is because the schedule pressure is expected to be more for the project with low SPCT, the productivity and errorproneness are severely affected, and, therefore, the project takes more time to complete. The extent of rework increases from 22% to 36% as SPCT increases from 40 weeks to 60 weeks. The extent of latent errors, the measure of software quality, remains around 2.5% for all cases.

Result of modeling and simulation of this problem has shown that there are two sets of models. One is the actual model of software development process constructed with actual project variables and parameters. The other is the mental model of the decision maker, which is perceptual in nature and constructed with the help of perform variables as perceived by him/her. Both of these models duly influence each other. Software project environment being autonomous in nature reacts to any change in the environment as well as to the decisions made by the decision-maker. This is evident in case of underestimation as well overestimation of scheduled project completion time of the project. In both the cases project dynamics changes in an unanticipated way and there is a risk of falling into the trap of self-fulfilling prophesies as demonstrated by the results of system dynamics model simulation. For instance, an unreasonably short estimate of project completion time introduces schedule pressure quite early in the software project environment. This schedule pressure, very low and low in the beginning keeps the productivity normal or may even increase it, during the first few weeks of project development. The outcome feedback received by the manager based on the reported good productivity, leads him to believe that shortening the scheduled project completion time helps completing the project early. However, what he appears to ignore is that schedule pressure, which was apparently increasing the staff productivity initially, would start yielding just opposite results when project reaches in the middle and schedule pressure becomes higher and higher. Moreover, the high schedule pressure not only adversely affects productivity, but also increases errorproneness of staff.

References

- Abdel-Hamid, T.K. (1989), The dynamics of software project staffing: A system dynamics based simulation approach, *IEEE Transactions on Software Engineering*, Vol. **15**, pp. 109-119.
- Abdel-Hamid, T.K. (1993), Thinking in circles, *American Programmer*, pp. 3-9, May.
- Abdel-Hamid, T.K. and S.E. Madnick (1986), Impact of schedule estimation on software project behavior, *IEEE Transactions on Software Engineering*, Vol. **12**, pp. 70-75.
- Abdel-Hamid, T.K. and S.E. Madnick (1991), *Software Project Dynamics: An Integrated Approach*, Prentice Hall, Englewood Cliffs NJ.

Abdel-Hamid, T.K., K. Sengupta and M. Hardebeck (1994), The impact of reward structures on staff allocation in a multi-project software development environment, *IEEE Transactions on Engineering Management* Vol. **41**, pp. 115-125.

Abdel-Hamid, T.K., K. Sengupta and D. Ronan (1993), Software project control: An experimental investigation of judgement with fallible information, *IEEE Transactions on Software Engineering*, Vol. **19**, pp. 603-612.

Boehm, B.W. (1981), *Software engineering economics*. Englewood Cliffs, NJ, Prentice Hall, 1981

Curtis, B., Krasner, H., Shen, V., and Iscoe, N. (1987), On building software process models under lamppost, *Proceedings of the Ninth International Conference on Software Engineering* pp. 96-103.

DeMarco, T. (1982), *Controlling software projects*, New York, NY: Yourdon Press, Inc.

Kemerer, C. (1987), An empirical validation of software cost estimation models. *Communications of the ACM*, Vol. **30**, pp. 416-429.

Kim, D.H. and J.H. Juhn (1997), System dynamics as a modeling platform for multi-agent systems, *Proceedings of the 15th International System Dynamics Conference*, August 19-22, Istanbul, Turkey, pp. 31-38.

Mills, H. (1983), *Software productivity*, Toronto: Little, Brown.

Mohapatra, P.K.J. and B. Mahanty (1993), *Software project management: A system dynamics approach*, Unpublished lecture notes, Department of Industrial Engineering and Management, Indian Institute of Technology, Kharagpur, India.

Putnam, L.H. (1978). A General Empirical Solution to the Macro Software Sizing and Estimating Problem, *IEEE Transactions on Software Engineering*, Vol. SE-4, No. 4, July, pp.345-361.

Rai, V.K. (1998), *Studies on Behavioral and Organizational Issues in Software Projects*, Unpublished Ph.D. thesis, Department of Industrial Engineering and Management, Indian Institute of Technology, Kharagpur, India.

Richardson, G.P. (1991), *Feedback thought in social science and systems theory*. Philadelphia: University of Pennsylvania Press, P.84.

Sengupta, K. and T.K. Abdel-Hamid (1996). The impact of unreliable information on the management of software projects: A dynamic decision perspective. *IEEE Transactions on Systems, Man and Cybernetics-part A: Systems and Humans*. Vol. 26. no. 2 .

Shneiderman, B.(1980), *Software psychology- Human factors in computer and information systems*, Cambridge, MA: Winthrop Publishers, Inc.

Smith, B. J., N. Nguyen and R.F. Vidale (1993), Death of a software manager: How to avoid career suicide through dynamic software process modeling, *American Programmer*, pp. 10-17, May.

Waterson, P.E., C.W. Clegg and C.M. Axtell (1997), The dynamics of work organization, knowledge and technology during software development. *International Journal of Human-Computer Studies*, Vol. 46, pp. 79-101.

Thibodeau, R. and E.N. Dodson (1980), .Life cycle phase interrelationship, *Journal of systems and software*, Vol. 1, pp. 203-211.