Impact of Requirements Discovery Pattern on Software Project Outcome: Preliminary Results

Rahul Thakurta  
Indian Institute of Management  
Calcutta, Joka, Diamond Harbour  
Road, Kolkata 700104  
fp092004@iimcal.ac.in

Rahul Roy  
Indian Institute of Management  
Calcutta, Joka, Diamond Harbour  
Road, Kolkata 700104  
rahul@iimcal.ac.in

Subir Bhattacharya  
Indian Institute of Management  
Calcutta, Joka, Diamond Harbour  
Road, Kolkata 700104  
subir@iimcal.ac.in

Abstract

Requirements discovery during project development is known to be the most critical risk in any software project, and managing this is paramount to success in software development. Requirements can change in various ways during the course of a project, and the effect of change on the outcome can differ widely. In this paper we study the effect of different patterns of requirements discovery on a software project. Using a validated model of software process we show that the effect on the total effort, the completion time and the workforce deployment patterns can be counter intuitive as different patterns of change impact software project dynamics in different ways. The insight into the relationship between requirements discovery pattern and project outcome can help managers decide appropriate risk mitigation policy and workforce augmentation plan.

1. Introduction

Software developers nowadays have reconciled to the fact that requirements will change during the development of software [5]. It is also known – and, in fact, quite intuitive – that such requirements discovery during development adversely impacts project outcomes like effort, time and residual errors [3, 4]. Recent effort estimation models like COCOMO II tries to factor in expected amount of change in requirement in terms of ‘breakage’ [2]. However, different projects experience different requirement discovery patterns. For instance in some project, rate of requirements discovery may be high towards the start of the project, slowing down subsequently as time passes. Another project might experience a completely opposite behavior, wherein the rate of requirements discovery increases as the project progresses. Will the addition of requirements in these two projects have similar impact on the project outcomes? Again, effort in a software project has various components – development, QA, testing, training etc. Are all the components affected equally with different patterns of requirements discovery? How is the hiring pattern influenced, if at all, with changes in requirements discovery pattern? Answers to these and similar other questions are of paramount importance to the project managers, especially to decide risk mitigation plan and workforce augmentation plan. This paper discusses only the relationship between requirements discovery pattern and project outcomes in the absence of any cost or penalty structure. Extensions that are currently being investigated upon have been discussed in the concluding section.

New requirements, as and when they are discovered, dynamically get added to the project tasks that are already being worked upon. To see the effect of various rates of requirements discovery, we propose to study how each selected possible pattern influences the dynamics of the project. Software engineering research broadly classified under Software Project Dynamics has used simulation models to show that complex dynamics of management action and developers’ rational choice causes software project performance to deviate significantly from those estimated. Our starting point is Abdel-Hamid’s [1] Systems Dynamics (SD) model that integrates all relevant processes of software development. Abdel-Hamid’s model uses a factor ‘Task Underestimation Fraction’ that captures fraction of undiscovered tasks that get added to the project scope. However the model does not provide for investigating the sensitivity of the project outcomes to changes in requirements discovery pattern.

The next section briefly describes the model and outlines the tasks environment. The experimental design and findings are summarized in Section 3. Section 4 analyses the findings. Finally Section 5 concludes the paper with a discussion on future research opportunities.
2. Model Description

System Dynamics (SD) [6] has been used increasingly in software development to model different problems and conduct what-if type analysis to assist various stakeholders [1]. The basic premise in SD is that system behavior results from interaction among its feedback loops. Model building begins with development of a causal loop diagram that consists of a collection causal links, each having a certain polarity. A positive (negative) link implies a reinforcing (balancing) relation where a positive change in the cause results in a positive (negative) change in the effect. A double line intersecting a link represents delays in an effect. A causal loop is formed by a closed sequence of causal links. A negative feedback loop has an odd number of negative polarity links, while a positive loop has an even number of negative links. The causal loop graph can be mapped to a mathematical model consisting of a system of difference equations, which can be simulated under different parametric conditions.

Abdel-Hamid [1] used his simulation model to investigate software cost and schedule estimation, the economics of the quality assurance function, and project staffing. Regarding system boundaries, they confined their model to development, ignoring requirements gathering and analysis, and to the software development organization. The model was constructed based on in-depth interviews with sixteen software project managers at three major corporations and on an extensive review of software engineering
literature. The interviews were used to fill in gaps in the literature. The model parameters were estimated based on a NASA satellite software development project and by reproducing the patterns of the completed software project cost, schedule, and work force loadings.

The key assumptions of Abdel-Hamid’s model are:

a) The project follows waterfall methodology.
b) The software tasks are divisible and can be carried out in parallel.
c) The requirements once specified do not change. Only new requirements can get added in course of the project.
d) There is no upper bound on the project’s schedule. The schedule gets extended based on the amount of effort remaining to be spent in the project.

A high level view of his model is presented in Figure 1. Each arrow on this diagram represents a cause-effect relationship e.g. Schedule Pressure affects Error Rate. In some cases sequence of arrows give rise to a loop e.g. Adjustments to Workforce and Schedule → Schedule Completion Date → Adjustments to Workforce and Schedule.

Perceived Project Size culminates in Project Tasks Perceived Completed as Workforce at Actual Productivity level work on the project at Software Development Rate. Workforce size changes as a result of Adjustments to Workforce and Schedule decision of the management and the resulting Hiring rate. Actual Productivity of workforce is affected by Potential Productivity, Process Losses and Learning. Rookies in the team affect Workforce Experience Mix unfavorably and lower Actual Productivity. Increase in Workforce size increases Process Losses and deteriorates Actual Productivity. Increase in Forecasted Completion Date increases Schedule Pressure and, in turn, increases Actual Productivity.

In reality the change in dynamics due to change in Perceived Project Size is far more complex because of delays in various cause-effect links. For example, organizations take time to find right people and allocate them to projects. Rookies also take time to get trained and become fully productive. This introduces delay between Adjustments to Workforce and Schedule decision and Workforce. The increase in Effort Perceived Still Needed caused by increase in Perceived Project Size thus takes time to affect increase in Project Tasks Perceived Completed and subsequent downward adjustment of Effort Perceived Still Needed.

3. Experiment Design and Results

As mentioned before, we intend to study the effect of requirements discovery patterns on project outcomes. In a real life software project, the rate of requirements discovery can follow an arbitrary pattern. In terms of the demonstrated variation in the project tasks some of the probable ones can be approximated as:

a) Linear: Constant rate of requirements discovery throughout the project’s duration causes project tasks to grow linearly.
b) Logarithmic: Initial high rate of requirements discovery decreases with time. Close collaboration with users generates high rate of requirements change initially. With time the requirements gradually stabilize and the rate of change declines.
c) Exponential: Rate of requirements change increases with time. Users’ and developers’ learning curves make project tasks grow at an increasing rate.
d) S-Curve: Project tasks grow initially at an increasing rate as developers learn about the domain. But with few requirements left to be discovered the project tasks stabilizes at a diminishing rate.

The model parameters in our study are same as that of the ‘NASA project’ case study reported in Abdel–Hamid [1]. The Initial Specified Job Size is 1067 Function Points (FP) which corresponds to Delivered Source Instruction of 64,000. We take a Task Underestimation Fraction of 0.67 implying that the Initial Specified Job Size can grow by 50%. COCOMO estimates for effort, time and manpower are as follows:

<table>
<thead>
<tr>
<th>Initial Specified Job Size</th>
<th>64000 DSI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effort (assuming 19 working days in a month)</td>
<td>3592Person-days</td>
</tr>
<tr>
<td>Estimated Completion Time (ECT)</td>
<td>348 days</td>
</tr>
<tr>
<td>Full-Time Equivalent Software Professional (FTE)</td>
<td>10.32</td>
</tr>
</tbody>
</table>

**Table 1: Project initial Estimates**

The growth of project tasks under different requirements discovery patterns are shown in Figure 2.
Since the Task Underestimation Fraction is same in all cases, the same amount of tasks always gets delivered at the end. However, different requirements discovery patterns modulate the growth of Perceived Project Size in different ways. In all simulation runs, half of the FTE estimated is taken as the initial Workforce. Unlike Abdel-Hamid’s model, we put a cap on the maximum allowable delay. We stipulate that the Scheduled Completion Date in the model can vary by a maximum of 40% over estimated completion time (ECT). The model tries to compensate for schedule shortfall beyond maximum tolerable limit through hiring. Table 2 shows comparison of project outcomes across different patterns. The ‘Base’ column gives result for the run where the requirement remains fixed at the Initial Specified Job Size (1067 FP).

4. Discussions on the Results

From Table 2 we observe that for an increase of 50% in the Initial Specified Job Size, the total effort increase ranges from 51% (logarithmic pattern) to 73% (linear pattern). However, schedule-wise variations are comparatively low. In the absence of any penalty on time, the Base case (with no addition to initial requirements) stretches the schedule within permissible limits using less FTE, and at consistently lower productivity (graph not shown). The increase in efforts devoted to different components of the project activities vary, the most notable being variations in Training Effort (79% to 154%) and Testing Effort (74% to 118%). Increase in total errors varies between 27% (exponential) and 33% (logarithmic). To explain these variations let us contrast results for the linear and the logarithmic patterns.

Total effort is product of Workforce and Schedule Completion Date. Average workforce (FTE in Table 2) in the logarithmic case is minimal and the Scheduled Completion Date is earliest. In linear case, in contrast, it is just the opposite. It is important to note that the Workforce in a project gets distributed among different types of activities like development, training, testing, QA etc. Hence effort spent in training eats into effort for development and delays the project. Figure 3 shows

![Figure 2: Perceived Project Size under Different Requirements Discovery Pattern](image-url)

Since the Task Underestimation Fraction is same in all cases, the same amount of tasks always gets delivered at the end. However, different requirements discovery patterns modulate the growth of Perceived Project Size in different ways. In all simulation runs, half of the FTE estimated is taken as the initial Workforce. Unlike Abdel-Hamid’s model, we put a cap on the maximum allowable delay. We stipulate that the Scheduled Completion Date in the model can vary by a maximum of 40% over estimated completion time (ECT). The model tries to compensate for schedule shortfall beyond maximum tolerable limit through hiring. Table 2 shows comparison of project outcomes across different patterns. The ‘Base’ column gives result for the run where the requirement remains fixed at the Initial Specified Job Size (1067 FP).

![Table 2: Comparison of Project Parameters Under Different Requirements Discovery Pattern](image-url)
that logarithmic case uses larger and steady workforce up to period 350. Due to rapid build-up of requirement initially, Hiring Rate gets triggered early. This augments the Workforce quickly and they become fully productive quite early in the project. Overall this results in lower peak in Workforce in the logarithmic case and reduced training effort. A higher level of workforce from the beginning enables higher Software Development Rate as is evident in Figure 4. Schedule Pressure also builds up later in logarithmic case (not shown).

Figure 3: Workforce Use

Development effort remains almost same for all patterns since the functionality delivered at the end is identical in all the cases. QA and consequent rework efforts are also dependent only on delivered task and hence hardly different across patterns.

But how come the total errors generated is more in case of logarithmic than in case of linear pattern? The CLD of figure 1 shows that Error Rate is affected by Software Development Rate and the Schedule Pressure, implying that increase in any one increases the Error Rate. As can be seen from Figure 4, Software Development Rate for the logarithmic case is more for a substantial period of time than the linear case. In Abdel-Hamid’s model, the Error Rate increases stock of Active Undetected Errors – higher the Error Rate, speedier the build-up. Active Undetected Errors regenerates other errors in the software during development. Increase in Active Undetected Errors therefore sets in motion a reinforcing cycle causing more

Active Undetected Errors to be created. This cycle can only be nullified by testing effort that brings down the stock of Active Undetected Errors. Figure 5 plots the Error Regeneration Rates for three patterns.

Figure 4: Development Rates

Then why is the testing effort less in case of logarithmic pattern compared to that of linear? As per Abdel-Hamid’s model, testing starts only when 80% of the Perceived Job Size has been worked upon. In the linear case, because of requirements trickling in, the target of 80% required to trigger testing keeps moving to the right. Workforce augmentation is also slow compared to the logarithmic case (figure 3). Owing to delay in hiring and training, by the time the augmented workforce become fully productive, only the testing tasks remain, and the entire workforce is devoted to this activity at a productivity level lower than that in the case of logarithmic pattern (not shown). At this point a few words about the exponential pattern would be in order. Initially, with few addition of requirements, work progresses like the Base case albeit with slightly higher productivity. As new requirements pile up towards the end, workforce gets augmented, and, owing to schedule limit, additional tasks are worked upon at high productivity level.
5. Conclusion

Based on the simulation results we can therefore conclude that the requirements discovery pattern can significantly impact project outcomes. The impacts on various components of effort are also not equal or proportional. Some of the findings are apparently counter-intuitive, and warrant insight into the project dynamics for proper explanation. Change in requirements is the norm. However, though the intention of the project manager would be to gather as much of requirements as possible in the initial stage of the project, the additional tasks may unveil following a totally different pattern during development. This type of simulations can help the manager to assess the risk of schedule and cost overruns. Manager can even extrapolate rate of growth of requirements experienced till any point of time during development, and take pre-emptive measures to mitigate the risk. Needless to say, it would involve augmentation of Abdel-Hamid’s model with cost and penalty implications of managerial decisions. Similar experimentations with other process models like the iterative-incremental, spiral and agile methodology will enable as to assess their applicability under requirements volatility. Work is on the anvil to check sensitivity of the results in the presence of cost and schedule overrun penalties. Trade-off between people-on-bench and apprehended penalties due to added task is also being investigated upon. It would be interesting to see if this method can lead to design of an appropriate change management process.

6. References


