Impact of Scope Creep on Software Project Quality*

Rahul Thakurta
Assistant Professor
Xavier Institute of Management,
Bhubaneswar, India
rahul@ximb.ac.in

Abstract

Generation of software requirements can occur in different ways during the course of the project, affecting the process in a widely different manner and extent. Using system dynamics modeling approach, here we study the impact of scope creep following different patterns on the project quality assurance activity. Results indicate non-uniform deviations across values of certain process parameters like quality assurance effort, error detection, etc under the experimental scenarios. Findings are expected to assist project managers in devising approaches that contribute to better quality of the final delivery.

Keywords: Quality Assurance, Scope Creep, System Dynamics

Introduction

Despite advances in project management methodologies and tools, the chances that a software project is able to achieve its process estimates is about 30% (The Standish Group, 2004). A major reason behind such phenomena is the frequent change of requirements during project progress, brought about by the dynamic nature of development activities (Winters, 2010). This phenomenon where there is a change in the set of requirements due to addition, deletion, and modification during project progress is called requirement volatility. Studies related to software project risks have attributed requirement volatility as one of the primary causes resulting in project failures (Davis, Nurmuliani, Park and Zowghi, 2008; Mathiassen, Saarinen, Tuunanen and Rossi, 2007).

In this study, we narrow down the definition of requirement volatility to focus only on scope creep. Scope creep has been defined as the addition of requirements or functionality to the existing scope during execution of the project. The addition/generation of requirements out of scope creep has been observed to occur following different patterns, for example, exponential decay (Abdel-Hamid and Madnick, 1991), exponential rise (Zowghi and Nurmuliani, 2002), and triangular (Houston, Mackulak and Collofello, 2001). These patterns are the result of generation of change orders/requests by the users or customers while the project development is still on. Now, will these changes following different patterns have similar impact on the project quality? The answer to such a question assumes greater significance in the current context, with the delivery quality considered to be one of the important determinants of project success or failure.

* *Received November 16, 2012; Revised February 11, 2013*

In this paper, we investigate the impact of these various requirement generation patterns out of scope creep on parameters related to the project quality assurance (QA) activity. QA is chosen because of its significant role in upfront detection and correction of errors (Abdel-Hamid, 1988). We introduce the metric QA effectiveness in order to judge the usefulness of the QA process. QA effectiveness is measured as follows:

$$\text{QA effectiveness} = \frac{\text{number of errors detected}}{\text{QA effortexpended}}$$

This metric is different from the Defect Removal Efficiency (DRE) metric which is one of the most important metric for measuring QA efficiency. DRE metric is represented as:

$$\text{DRE (in \%)} = \frac{100*E}{(E+D)}$$

Where, E = number of errors found before delivery
E+D = number of errors found after delivery

We have used QA effectiveness in our study as we focus on how scope creep during project development influences actual effort allocation to the QA process, and this is expected to have tangible influence on the project delivery quality.

The paper is structured as follows: the next section updates on some background work related to this research work. The following section presents the methodology that has been adopted to carry out the study. The study results are presented and discussed subsequently. Finally in conclusion, we summarize the key findings and highlight the future research opportunities.

Relevant Literature

Scope creep has been addressed in the literature mostly in the context of requirement volatility. Studies on scope creep in software projects have looked into its nature, cause and effects, and management strategies. Researchers have tried to ascertain scope creep both in terms of magnitude and pattern of requirement addition (Barry, Kemerer and Slaughter, 2006; Thakurta, Roy and Bhattacharya, 2009). The causes of scope creep mostly relate to problems with comprehension (e.g. conflicting requirements), behaviour (i.e. of project members and users), and actions (i.e. project management decisions) (Davis et al., 2008; Kotonya and Sommerville, 1998; Nurmuliani, Zowghi and Fowell, 2004). The effect of scope creep is found to be pronounced on project quality. Late additions in requirements significantly impact the proportion of high severity defects resulting in deterioration of product quality (Ferreira, Collofello, Shunk and Mackulak, 2009; Zowghi and Nurmuliani, 2002). Finally, the activities concerning managing scope creep have mostly tried to manage occurrence of scope creep with respect to magnitude. The dominant suggestions in this regard include formation of change control boards (Jones, 1998), adoption of specific project development methodologies (Thakurta and Ahlemann, 2010), and adoption of specific approaches, for example, developing requirements jointly with users (Jones, 1998), base lining requirements (Wiegers, 1999), etc.

Studies on software project QA have primarily focused on the different quality improvement approaches in order to increase acceptance of the project deliverables. Of the recent studies, Liu, Tamai and Nakajima (2009) suggest a way to integrate formal specification, review, and testing activities so as to detect and rectify requirement errors. Wagner, Lochmann, Winter, Goebb and Klaes (2009) update on the usage, techniques, and associated problems related to four quality models used in practice. Li, Shu, Boehm and Wang (2010) investigate the effectiveness of review, process audit, and testing and their overall contribution to return on investment (ROI).

The studies hitherto indicate scope creep to be a major cause of concern in relation to achieving successful project endeavours. Furthermore, the noted emphasis on managing scope creep mostly rests on assuming that scope creep can influence project performance in terms of the magnitude of occurrence. Given the importance of quality in terms of achieving project success (Shenhar, Dvir, Levy and Maltz, 2001), here we try to figure out how scope creep following identified patterns can influence the QA process.

Methodology

Task Environment

We use the system dynamics (SD) (Sterman, 2000) approach in our study which works on the premise that system behavior is an outcome of the interaction among its feedback loops. The vocabulary of SD introduces the causal loop diagram which represents the problem hypothesis. The causal loop diagram is developed by linking cause and effect relationships in a sequence of chains and loops. Each linkage is assigned a plus or minus sign. A plus sign present on a linkage indicates that the increase (decrease) in the independent variable will lead to increase (decrease) in the dependent variable. The minus sign on the linkage indicates the reverse. The causal loop model is subsequently converted into a stock and flow model. Simulation of the stock and flow model allows one to investigate for the desired effects under different run conditions.

Our research setting has been contextualized by considering a familiar in-house medium-sized project implementing the waterfall methodology (Royce, 1987). The choice of waterfall methodology was driven out of its observed predominance even in projects endangered because of scope creep (Thakurta and Ahlemann, 2010). The selection of waterfall methodology was found to be driven by management preferences and business influence. Given the findings, here we opt for Abdel-Hamid's (Abdel-Hamid and Madnick, 1991) SD model which implements the development environment of the waterfall methodology. Scope creep is captured in this model by use of a factor 'task underestimation fraction'. This represents the maximum amount of new tasks that can get added to the project scope over the project duration. In the model, schedule pressure plays a key role in determining the extent of effort allocation to QA. The model further assumes that there is no hard deadline for project completion. The causal loop diagram representing the problem of interest has been shown below in Figure 1. Figure 1 has been derived by drawing out the relevant causal loops from the Abdel-Hamid's SD model related to the

problem that we want to investigate in this study. The causal loop diagram shown in Figure 1 is able to capture successfully both the sources of scope creep and the parameters that will enable one to arrive at measure of QA effectiveness as discussed above. An explanation of the causal behavior arising out of the model structure is included below.

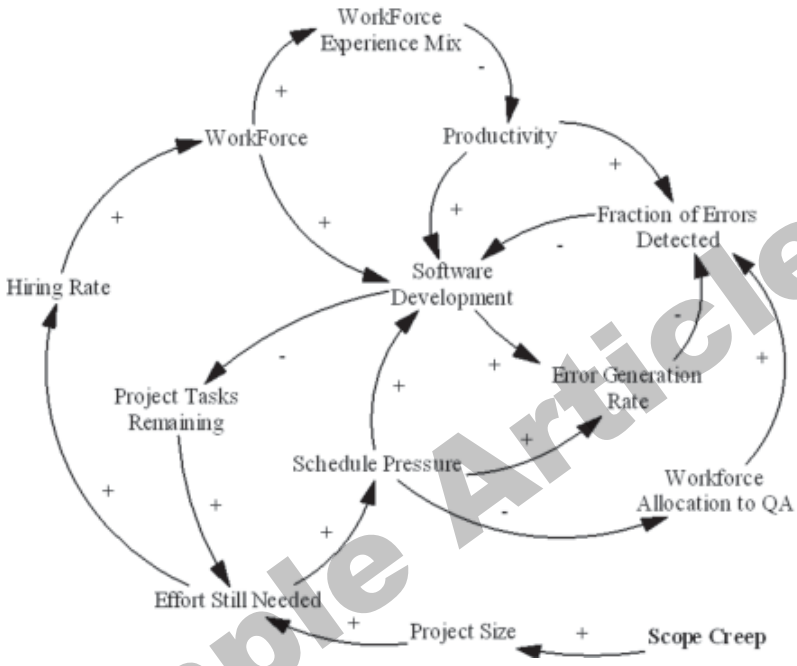

Figure 1: Model Causal Loop Diagram

Occurrence of *scope creep* contributes to an increase in the *project size.* The *schedule pressure* thus increases in the project. The increase in *schedule pressure* contributes to more error generation during project progress. On the other hand, given the scenario, there is a propensity of the project managers to cut down on the *workforce allocation to QA* in order to reduce chances of cost and schedule overruns. The net result is a decrease in the *fraction of errors detected,* thereby affecting QA effectiveness.

On the other side, the increase in the value of *effort still needed* because of *scope creep* necessitated project hiring. With hiring, the project *workforce* also increases. The rate of *software development* gets positively affected in presence of higher *workforce,* which in turn causes more tasks to be waiting for QA. *Schedule pressure* is reduced owing to task processing at a higher rate in presence of the higher *workforce.* The decrease in *schedule pressure* reduces the *error generation rate*. Simultaneously some increase in *workforce allocation to QA* also takes place. All these contribute to increasing the QA effectiveness.

Based on the model structure, we investigate how requirement generation following

different patterns influence the structure, and hence the behaviour. To achieve this, the model was simulated under four different requirement generation patterns given below using the commercially available iThink[1] software.

Experiment Design

In a real life scenario, generation of change orders can appear to be completely random. The basic structures of such patterns are depictions of increasing trend, decreasing trend, and uniform trend, which are described below for the purpose of experimentation. The presence of these basic structures is also evident from the requirement change pattern that has been provided in Stark, Oman, Skillicorn and Ameele (1999, pp 8) in a different context (i.e. for maintenance projects).

Linear Decay: Here the rate of change order generation is high initially, and it decreases linearly with time. The high rate of change order generation occurs because of collaboration with users' early on in the project. The requirements stabilize with time, and the rate of change declines. Projects implementing agile methodology are likely to demonstrate patterns which resemble linear decay pattern of requirement variation.

Linear Rise: This is opposite to that of the linear decay with the rate of requirement generation increases with time. Delayed user involvement in the project can contribute to such a variation of requirements with time. Projects implementing waterfall methodology are likely to demonstrate patterns which resemble linear rise pattern of requirement variation.

Uniform Variation: Constant rate of change order generation throughout the project's duration which causes project tasks to grow linearly. This kind of variation may be observed in projects which implements spiral methodology.

Triangular Variation: This is the combination of linear rise and linear decay pattern with linear rise early on, and linear decay at the later stages of the project. Projects implementing incremental-iterative methodology are likely to demonstrate patterns which resemble series of triangular variations in requirements.

The relevant parameters to simulate the model were based on the data of a real life software project carried out in an IT firm based in India. The project implemented the waterfall methodology, which matched our research setting (indicated above). The project is small sized having initial specified job size as 7572 delivered source instructions (DSI) which corresponds to 126.2 function points (FP). In order to carry to the simulation, it was required to specify the initial values of effort and schedule. COCOMO ('constructive cost model': Boehm, Clark, Horowitz, Christopher, Madachy and Selby, 1995) was used for the purpose, which enables one to arrive at the estimates of effort and schedule given the project size. The initial estimates of effort and schedule, hence derived is given in Table 1.

[1] Available at http://www.iseesystems.com/softwares/Business/ithinkSoftware.aspx

Table 1:Initial Parameter Estimates

Parameter	Estimate
Initial Specified Job Size	126.2 Function Point (FP)
Initial Estimated Effort	630 Man-Days
Initial Schedule Estimate	90 Days
Project Average FTE	7.0 Persons

In the model, we additionally set a quality objective of 75% implying high quality requirements of the project deliverables. We allow the project tasks to grow by 50% during project development which is achieved by setting the parameter task underestimation fraction at 0.67. The growth of project tasks under the four different requirement generation patterns (Figure 2) is shown in Figure 3.

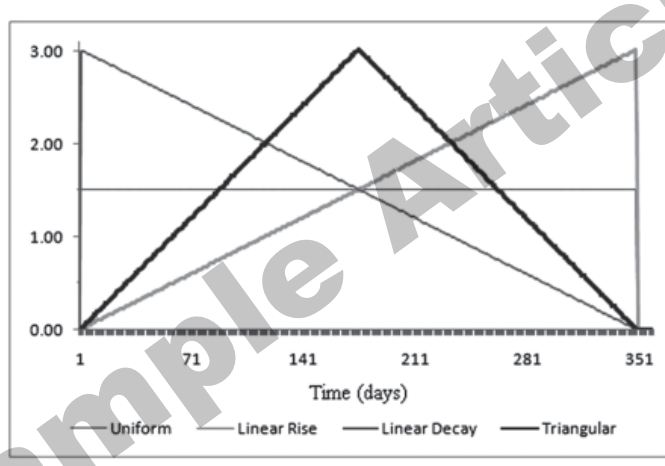

Figure 2. Change Order Generation Rates.

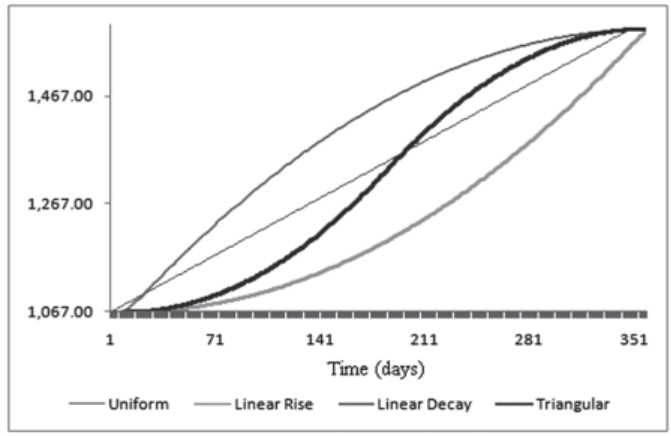

Figure 3. Growth of Project Tasks

The different requirement generation patterns modulate the growth of projects tasks in different ways contributing to variations depicted in Figure 3. The results are discussed in the next section.

Results

Table 2 show comparison of project performance under the four requirement generation patterns considered in this study. In all cases, a total of 189.3 FP of tasks were processed because of addition of new requirements to the project scope during development. The results allow us to make the following observations:

1. QA effort expended could be found to be maximum under uniform, and minimum under linear decay (19% variation).

2. Rework effort (33% variation) and completion date (6% variation), are also maximum under uniform, and minimum under linear decay.

3. QA effectiveness (elaborated above) could be observed to be maximum under linear decay, and minimum under uniform. The extent of variation in this case is about 21%.

Results indicate variations in parameters across the four requirement generation patterns considered in this study. In order to understand the variations, let's compare results for linear decay and uniform patterns.

Under linear decay, the change order generation rate is highest upfront (Figure 2). Due to rapid build-up of requirements initially, hiring gets triggered. This augments the workforce quickly as evident from Figure 4. The higher rate of change order generation in this case also leads to schedule pressure (not shown). The schedule pressure results in some curtailment in resources allocated to the QA activity. Hence, QA effort expenditure is lower in this case as the results in Table 2 shows.

At the final stages, with progressive decrease of change order generation (Figure 2), no further hiring takes place and the workforce settles at a constant level (Figure 4). The communications overhead is also less in this case, and hence the project is able to process the tasks relatively early (Table 2).

In comparison, with uniform pattern, the augmentation of project tasks happens uniformly throughout the project time-span (Figure 2). Under uniform, the perceived schedule pressure (not shown) is not much initially. Thus there is no curtailment in effort allocation to QA, which is found to be the maximum in this case (Table 2). In absence of perceived schedule pressure, the rate of hiring upfront is also comparatively low (Figure 4). The rate of hiring however increases towards the mid-end stages as more visibility is gained in presence of uniform change order generation. The FTE workforce is hence higher in this case compared to the other patterns (Table 2). The late hiring negatively influence project productivity (not shown), resulting in maximum schedule slippage to be encountered in this case (Table 2).

Table 2:Impact of Different Requirement Generation Patterns on Project Parameters

	Linear Decay	Linear Rise	Triangular	Uniform
QA Effort (Man-Days)	171	189	179	203
Rework Effort (Man-Days)	9	12	11	12
Completion Date (Days)	114	118	117	121
FTE Manpower (Person)	19.8	22.1	21.7	21.3
Errors Generated	58	61	65	57
Errors Detected	48	51	49	47
Percent Error Detected	82.76%	83.61%	75.38%	82.46%
QA Effectiveness	0.28	0.27	0.27	0.23

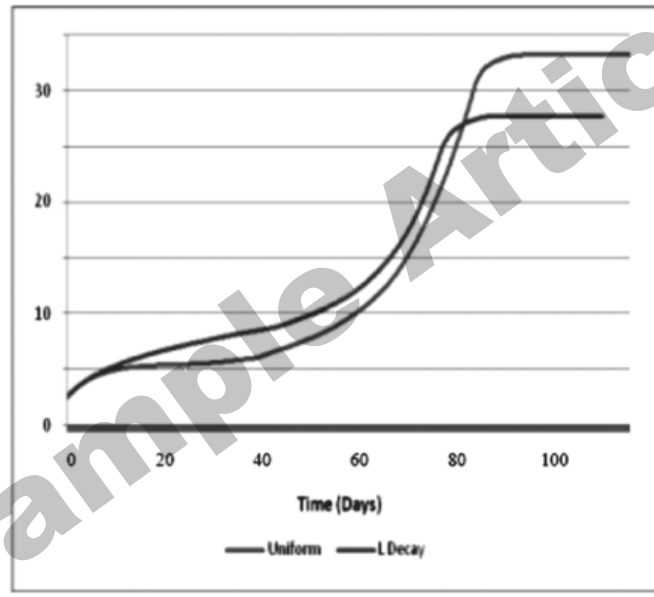

Figure 4. Variation in Workforce subjected to Uniform and Linear Decay Requirement Generation Patterns

Error generation under uniform change order generation is affected by the presence of rookies, and the schedule pressure and is found to be nearly identical to that under the linear decay pattern. In presence of the assigned quality objective of 75%, nearly same number of errors also gets detected under the two patterns. The presence of a lower value of QA effort expended hence resulted in QA effectiveness to be the largest under linear decay.

The results show how the pattern of change order generation influences the dynamics by which tasks are generated and processed during software development. Further, QA effectiveness values under the linear rise and triangular patterns are found to be intermediate (Table 2). Under the linear rise patterns, the change order generation

rate is lower during the initial stages (Figure 2). The schedule pressure hence remains lower, and there is not much curtailment in the QA activity. In comparison to the linear rise case, the curtailment in QA effort allocation is more under the triangular pattern as the gradient of increase in change order generation is higher in this case (Figure 2).

Conclusion

The results permit us to conclude that the effectiveness of the QA process is indeed influenced by the pattern of change order generation during project progress. Given the experimental scenarios, QA effectiveness was found to be the maximum under linear decay, while results under the uniform pattern indicated it to be the minimum. The results suggest that project managers should device some approaches of acquiring and releasing the requirements in a linear decay fashion, which in turn is also expected to contribute positively towards increasing the project quality.

The level of variation in the results under the different simulation runs is not very high given the facts that the experimentation was conducted based on data of small scale project, and the project did not have any imposed schedule penalty. Further, different results can also be obtained depending upon other project characteristics like project development methodology etc. This further suggests that the project manager needs to adjust their project management style based on the expectation of change order generation in projects. Project managers can use the simulation bench as a decision support framework in order to arrive at approaches that best meet the agreed performance criteria. Research can also look at efficacies of previously explored approaches towards multiple goal satisfaction (i.e. cost, schedule, quality adherence). We expect that the results will pave the way for more similar studies benefitting the software project management discipline.

References

Abdel-Hamid, T.K. (1988). The economics of software quality assurance: a simulation-based case study. MIS Quarterly, 12 (3).

Abdel-Hamid, T.K., & Madnick, S.E. (1991). Software project dynamics: an integrated approach. Englewood Cliffs, NJ: Prentice Hall.

Barry, E.J., Kemerer, C.F., & Slaughter, S. (2006). Environmental volatility, development decisions, and software volatility: a longitudinal analysis. Management Science, 52 (3), 448-464.

Boehm, B., Clark, B., Horowitz, E., Christopher, J., Madachy, R., & Selby, R. (1995). Cost models for future software life cycle processes: COCOMO 2.0. Annals of Software Engineering, 1, 57–94.

Davis, A.M., Nurmuliani, N., Park, S., & Zowghi, D. (2008). Requirements change: what's the alternative? Proceedings of the 32nd Annual IEEE International Computer Software and Applications Conference, 635-638.

Ferreira, S., Collofello, J., Shunk, D., & Mackulak, G. (2009). Understanding the effects of requirements volatility in software engineering by using analytical modeling and software process simulation. Journal of Systems and Software, 82 (10), 1568-1577.

Houston, D.X., Mackulak, G.T., & Collofello, J. (2001). Stochastic simulation of risk factor potential effects for software development risk management. Journal of Systems and Software, 59 (3).

Jones, C. (1998). Estimating software costs. New York: McGraw Hill.

Kotonya, G., & Sommerville, I. (1998). Requirements engineering: processes and techniques. Chichester: John Wiley and Sons.

Li, Q., Shu, F., Boehm, B., & Wang, Q. (2010). Improving the ROI of software quality assurance activities: an empirical study. Proceedings of the 2010 International Conference on New Modeling Concepts for Today's Software Processes: Software Process (ICSP'10), Berlin, Heidelberg.

Liu, S., Tamai, T., & Nakajima, S. (2009). Integration of formal specification, review, and testing for software component quality assurance. Proceedings of the 2009 ACM symposium on Applied Computing (SAC '09), New York, USA.

Mathiassen, L., Saarinen, T., Tuunanen, T., & Rossi, M. (2007). A contingency model for requirements development. Journal of the Association for Information Systems, 8 (11), 570-598.

Nurmuliani, N., Zowghi, D., & Fowell, S. (2004). Analysis of requirements volatility during software development life cycle. Proceedings of the 2004 Australian Software Engineering Conference, Innsbruck, Austria.

Ropponen, J., & Lyytinen, K. (2000). Components of software development risk: how to address them? A project manager survey. IEEE Transactions on Software Engineering, 26 (2), 98-112.

Royce, W.W. (1987). Managing the development of large software systems. Proceedings of the 9th International Conference on Software Engineering, Los Alamitos, CA, USA.

Shenhar, A.J., Dvir, D., Levy, O., & Maltz, A.C. (2001). Project success: a multidimensional strategic concept. Long Range Planning, 34 (6).

Stark, G.E., Oman, P., Skillicorn, A., & Ameele, A. (1999). An examination of the effects of requirements changes on software maintenance releases. Journal of Software Maintenance, 11 (5), 293-309.

Sterman, J.D. (2000). Business dynamics: systems thinking and modeling for a complex world. New York: Irwin/McGraw-Hill.

Thakurta, R., & Ahlemann, F. (2010). Understanding requirements volatility in software projects – an empirical investigation of volatility awareness, management approaches and their applicability. Proceedings of 43rd Hawaii International Conference on System Sciences, Hawaii, USA.

Thakurta, R., Roy, R., & Bhattacharya, S. (2009). Impact of requirements discovery pattern on software project outcome: preliminary results. Proceedings of the 42nd Annual Hawaii International Conference on System Sciences, Hawaii, USA.

The Standish Group. (2004). Chaos report. Retrieved from http://www.standish-group.com

Tiwana, A., & Keil, M. (2004). The 1- minute risk assessment tool. Communications of the ACM, 47 (11), 73-77.

Wagner, S., Lochmann, K., Winter, S., Goebb, A., & Klaes, M. (2009). Quality models in practice: a preliminary analysis. Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM '09), Washington, DC, USA.

Winters, F. (2010). The top ten reasons projects fail (part 7). Retrieved from http://www.gantthead.com/article.cfm?ID=187449

Zowghi, D., & Nurmuliani, N. (2002). A study on the impact of requirements volatility on software project performance. Proceedings of Ninth Asia Pacific Software Engineering Conference, Queensland, Australia.