Asian Journal of Information Technology 11 (6-12): 186-193, 2012
ISSN: 1682-3915
© Medwell Journals, 2012

Investigating Dynamics of Software Project Quality under Requirement Volatility Patterns

Rahul Thakurta
Department of Information Systems, Xavier Institute of Management,
Xavier Square, 751 013 Bhubaneswar, India

Abstract: The impact of varying magnitude of requirement volatility on project performance parameters like schedule, effort, etc. has been well investigated in the literature. However, there is a lack of evidence of how pattern of change of requirements can influence project management practices and project performance. In order to address the gap, here using system dynamics, researchers investigate the impact of different resource allocation strategies on project quality under two experimental patterns of requirement volatility. Findings indicate variation in quality metrics depending upon the experimental scenario, thereby suggesting the need to adopt pattern-dependent contextual project management practices.

Key words: Requirement volatility, quality assurance, resource management, system dynamics, India

INTRODUCTION

One of the challenges in software project management has been to adhere to the initially established process estimates. Requirement volatility which refers to the change in requirements (in terms of the number of additions, deletions and modifications) during the software development life cycle is considered to be the major reason behind the reported deviations. Literature on software project risk has identified requirement volatility as a significant risk affecting software project outcome (Boehm, 1991; Hoorn *et al*., 2007).

Studies focusing on requirement volatility have given more emphasis to magnitude of requirement changes (Ferreira *et al*., 2009). However, there is evidence that requirement volatility can also take place following different patterns given the same magnitude (Thakurta *et al*., 2009) with patterns referring to the various geometric shapes of requirements generation. It seems plausible then such pattern wise change of requirements will influence the choice of different management approaches used in projects. The available literature is silent on this front.

This study pursues the inquiry with the help of simulations carried out on an established system dynamics Model of Software project management proposed by Abdel-Hamid and Madnick (1991). By considering two requirements volatility patterns, researchers were able to demonstrate the efficacies of the different resource allocation policies as project management tasks on project quality. The study focused on the Quality Assurance (QA) activity with quality being measured with the help of the metric QA effectiveness which is defined as the ratio of number of errors detected and QA effort expended. The selection of QA activity was driven by the fact that error detection and correction at this stage is relatively easy and also less expensive (Abdel-Hamid, 1988). Further, a low value of QA effectiveness will imply either more errors in the final product or service delivered to the users or a higher expenditure of project effort arising out of error detection and correction during the later stages of the project (Abdel-Hamid and Madnick, 1991).

Relevant work: Researchers present a review of the relevant literature pertaining to requirement volatility, resource allocation and quality which the present research touches upon.

Requirement volatility: Software requirement volatility has been well researched upon in terms of understanding its nature and source; its effect and its management strategies. The findings indicate the following:

- Requirement volatility can occur not only in terms of magnitude (Barry *et al*., 2006; Costello and Liu, 1995) but also in terms of pattern of change (Thakurta *et al*., 2009)

- The causes of requirement volatility have been attributed to the presence of inconsistencies or conflicts among requirements; activities carried out during the project like defect fixing, functionality correction (Nurmuliani *et al*., 2004); evolving user/customer knowledge and priorities; technical, schedule or cost related problems, change in research environment (Davis *et al*., 2008; Kontonya and Sommerville, 2002) and process model selection decisions (Madachy, 1994). In this context, a process model (also known as Systems Development Life Cycle Model or SDLC) describes the various stages involved in an information system development project and provides a mechanism to plan for and manage project execution
- Late changes in requirements are found to increase software defects resulting in deterioration of product quality (Zowghi and Nurmuliani, 2002)
- Suggestions to managing projects under varying magnitude of requirement volatility include adoption of specific frameworks (like formation of change control boards (Jones, 1998) and specifying the project execution strategy upfront like selecting the process model for the project (Thakurta and Ahlemann, 2010) and adoption of specific techniques during project development (for example usage of Joint Application Design (JAD) and configuration management (Jones, 1998), base lining requirements (Wiegers, 1999), proper change management planning (Young, 2001), etc.

There is still no evidence of how pattern-wise occurrence of requirement volatility can be effectively managed. Researchers expect that the project management approach will be contingent on such pattern-wise variation, thereby affecting project quality.

Resource allocation: Discussion on resource allocation policies in software projects is found to be limited, probably driven by the fact that each software project represents a unique scenario (Otero *et al*., 2009). In software projects, the estimates of tasks' durations are often imprecise and subjective and progress is difficult to estimate (Plekhanova, 1999). The available literature discusses different resource allocation policies with optimal effect on project performance like variation of resource adjustment times (Lee *et al*., 2007) use of proportional and foresighted resource forecasting techniques (Joglekar and Ford, 2005), altering resource allocation order to project tasks (Black and Repenning, 2001), overstaffing the project from the onset (Collofello *et al*., 1998) and keeping the level of workforce constant (Collofello *et al*., 1998). An investigation of the efficacy of each can act as a rule of thumb in facilitating improved resource allocation decisions.

Quality: Researchers focus on project QA activity given its importance as noted above. Studies on software QA have primarily focused on the different quality improvement approaches in order to increase acceptance of the project deliverables. In this regard, Basili and Rombach (1987) provide a five step methodology for software process improvement based on analysis of defect related data. Liu *et al*. (2009) present an approach of integrating formal specification, review and testing activities with a view to remove errors and identify missing requirements. Wagner *et al*. (2009) presents the findings of a survey on quality models in practice conducted among four software companies in order to update on the usage, techniques and associated problems encountered in practice. Li *et al*. (2010) investigate the effectiveness of three types of QA activities (viz. review, process audit and testing) and their overall contribution to QA Return On Investment (ROI). The study investigates how choice of a specific project management task (different resource allocation policies in the case) affects the QA process and hence is different from the above treatments.

MATERIALS AND METHODS

Task environment: Project management is a complex phenomenon involving a dynamic interplay of a wide range of hard and soft factors (Crawford and Pollack, 2004). This prompted us to use the System Dynamics (SD) (Sterman, 2000) approach that uses feedback structures to analyze system behaviour. The first step in model building is to develop a causal loop diagram consisting of a collection of 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. The causal loop graph can be subsequently mapped to a mathematical model consisting of a system of difference equations which can be simulated under different parametric conditions.

The study setting is contextualized to represent 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 requirements volatility (Thakurta and Ahlemann, 2010). The finding prompted us to opt for Abdel-Hamid and Madnick, 1991)'s SD Model based on waterfall methodology. The model effectively integrates all relevant processes of software development like development, quality assurance, testing, rework, etc. It also allows one to investigate for the effect of changes in project human resource, project size and project plan on project progress and in the process appreciate the dynamics involved in software development so as to better manage the changes. Validation of the model was carried out based on case studies conducted by the researchers and supplemented by expert review techniques (Abdel-Hamid and Madnick, 1991). The model further assumes the following:

- The software tasks are divisible and can be carried out in parallel
- The requirements once specified do not change. Only new requirements get added in course of the project
- Quality assurance gets precedence over development activity during the course of the project

The model uses a factor task underestimation fraction that captures fraction of undiscovered tasks that get added to the project scope and is a measure of the magnitude of requirement volatility during project development. Effort allocation to QA gets adjusted based on the project schedule pressure. However, there is no imposed cap on the maximum allowable delay during project development. Figure 1 shows the causal loop diagram of the problem embodied in the model structure. The causal loop diagram was arrived upon by identifying the structure representing the problem of interest from Abdel-Hamid's SD Model and hence it excludes project testing and subsequent activities. The model behaviour can be understood based on how the different feedback loops influence the dynamics. A description of the behavior of the causal loop diagram is provided below with the model parameters shown in italics.

Requirement volatility during project development leads to augmentation of project size. With increase in

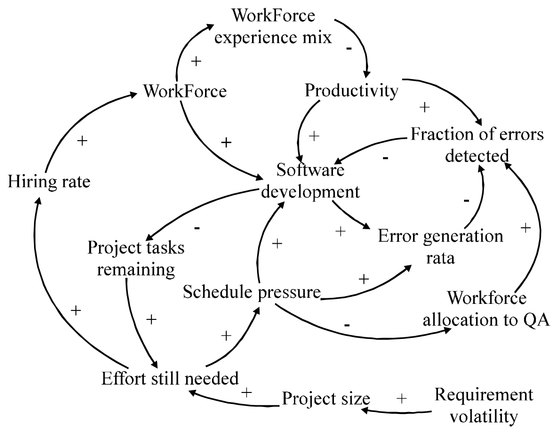

Fig. 1: Model causal loop diagram

project size, the estimate of effort still needed to complete the project which is a function of projectsize (Boehm *et al.*, 1995) also increases. This increased effort requirement positively affects the schedule pressure and leads to generation of more errors because of higher error generation rate. With increase in schedule pressure, some readjustment in the software team engaged in the project is expected to take place. In order to meet the agreed upon delivery schedule and keep the project costs under control, the project managers facing schedule pressure might give more priority to development related activities compared to QA. In the process, they might completely abandon or do some curtailment in the team assigned for QA (Abdel-Hamid and Madnick, 1991). Thus, under the circumstances, some reduction in percentage of workforce allocation to QA takes place. High error generation rate and reduced QA manpower in turn negatively impacts fraction of errors detected, thereby hampering QA effectiveness. The increased effort requirement (effort still needed) arising because of requirement volatility also induces hiring (hiring rate) which increases the project workforce. Presence of a higher workforce boosts up software development resulting in more number of tasks pending for QA. Tasks processing at a higher rate bring down the effort still needed (because of reduction in project tasks remaining) and thus helps to reduce the schedule pressure. The decrease in schedule pressure reduces the error generation rate. Under the circumstances and with availability of a larger workforce, percentage of workforce allocation to QA also increases. The net result is an increase in QA effectiveness.

The dynamics is further completed by the pattern in which change orders are generated during project development (requirement volatility pattern) and the resource allocation policy adopted. The later changes the workforce experience mix (ratio of rookies and experienced professionals in the workforce) and thus affects the software process owing to the fact that rookies are less productive and also more error-prone compared to their experienced counterparts.

The model parameters (Table 1) were set as per the TRW Inc. case study (Abdel-Hamid and Madnick, 1991) which matches the project context. The reported project is medium sized having initial specified job size as 64,000 Delivered Source Instructions (DSI) which corresponds to 1067 Function Points (FP). The initial estimates of effort and schedule were derived using COCOMO (Constructive Cost Model: Boehm *et al.*, 1995) as follows.

The value of project average Full-Time-Equivalent (FTE) professionals was arrived at 10.3 persons; implying ten persons to be working fulltime on the project and one person to be devoting 30% of his/her daily research-hour on the project.

Experiment design: Researchers experimented with the following two change order generation patterns which closely approximates some real project scenarios.

Linear rise: The exponential rise pattern of change order generation rate given in NASA case study (Abdel-Hamid and Madnick, 1991) was approximated using the linear rise pattern in this case. Here, the rate of change order generation increases linearly with time. Users and developers learning curves make project tasks grow at an increasing rate.

Uniform variation: Constant rate of change order generation throughout the project's duration which causes project tasks to grow linearly. Researchers experimented with the following workforce management policies to investigate their effect on the QA activity:

Policy 1: Controlling the level of workforce over the development period. This can be visualized as a project manager trying to maintain the level of workforce at some desired value (Collofello *et al*., 1998). The situation can arise for example when the project is executed in a fixed price contract and the management is unwilling to change the workforce level. Here, researchers considered two scenarios: scenario A with the level of desired workforce estimated based on initial project scope and scenario B with desired workforce level estimated based on the hunch of the size of additional tasks arising out of change order generation.

Policy 2: Overstaffing the project from the start. Here, the project maintains additional bench strength based on the expectation of requirement volatility during project development. Usage of this overstaffing strategy can be noticed by Collofello *et al*. (1998). Projects which have high business impact or face huge time constraint can employ this strategy. In real life, the extent of overstaffing in projects could be at different degrees depending upon the projects' intended objectives. For experimentation purpose, here researchers implement overstaffing by setting the value of starting workforce equal to twice the average Full Time Equivalent (FTE) as shown in Table 1. This would provide us with an understanding of how this policy influences project dynamics under the described experimental settings.

Policy 3: Appropriately managing the resource allocation delays. Resource allocation delays represent the average time required to hire in extra personnel from outside the organization. Past research has indicated that tuning resource allocation delays to the project characteristics

Table 1: Initial parameter estimates

Parameters	Estimate
Initial specified job size	1067 function point
Initial estimated effort	3594 person-days
Initial schedule estimate	348 days
Project average FTE	10.3 persons

Table 2: Parameter changes for policy implementation

Policy #	Parameter values
1	Scenario A: Desired workforce level = 10.3; Scenario B: Desired workforce level = 15.8
2	Starting workforce level = 20.6 (Twice the average FTE, Table I)
3	Scenario A: Hiring delay = 5 days; Scenario B: Hiring delay = 75 days
4	Forecasted hiring rate ~ Change order generation rate

helps to improve the project performance (Lee *et al*., 2007). Reduction in hiring delays is possible through pre-hiring of desired competency. Two scenarios were considered-the first having resource allocation delay of 5 days (scenario A) and the second having delay of 75 days (scenario B). These two scenarios were considered to be representatives of minimum hiring delay and maximum hiring delay for the example. Evidence of usage of a similar range of hiring delay as proxy of minimum and maximum can also be noticed in Lee *et al*. (2007).

Policy 4: Using resource allocation strategy based on forecasting techniques depending upon requirement change expectations. The assumption researchers take here is that the project managers based on earlier experiences or based on data of earlier occurrences have guessed the pattern in which the requirement is expected to change in their project and have planned resource deployment accordingly. Researchers adopted a proportional forecasting policy where the hiring rate is adjusted in an identical fashion as the rate of change order generation in the project. Evidence of usage of forecast based resource allocation techniques can be noticed by Joglekar and Ford (2005).

Table 2 lists the parameter values relevant to the implementation of the stated policies. The values of other parameters are same as that of the base case (i.e., the behaviour as depicted by the model structure without implementation of any of the resource allocation policies).

In order to carry out the simulation, the cause and effect model shown in Fig. 1 was converted into a Simulation Model also known as a stock and flow diagram (Sterman, 2000) using the iThink (http://www.iseesystems.com/software/Business/ithinkSoftware.aspx) Software. In the Simulation Model, researchers set a quality objective of 75% implying project in concern has high quality requirements which appropriately matches the study objectives. The task underestimation fraction is set at 0.67 implying that the initial project size can grow by 50% during project development because of requirement volatility. The growth of project tasks under linear rise and uniform requirement volatility patterns (Fig. 2) is shown in Fig. 3.

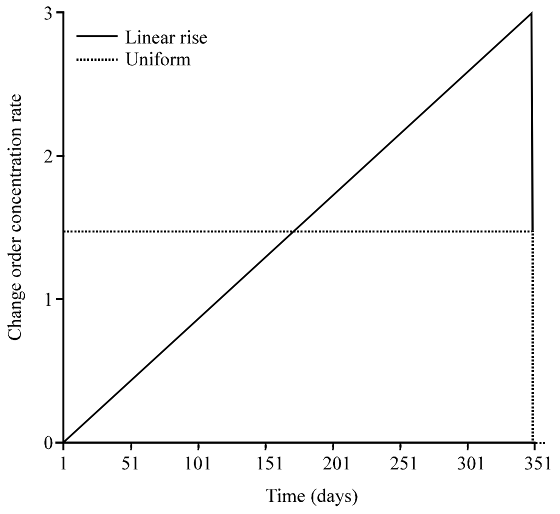

Fig. 2: Change order generation rates

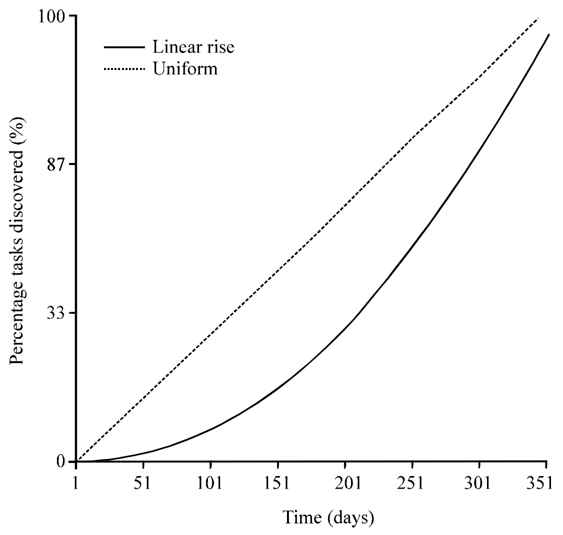

Fig. 3: Growth of project tasks

is same in all cases, the same amount of tasks always gets delivered at the end. However, the two change order generation patterns modulate the growth of project tasks in different ways.

RESULTS AND DISCUSSION

First, let's consider the linear rise pattern of requirement volatility. Table 3 show, comparison of Since, the task underestimation fraction project performance for Base and the different policies as discussed above (Table 2). The values in each cell in Table 3 shows the actual result of simulation and a percentage (%) figure given within brackets. The percentage figure indicates where the values of each parameter stand with respect to the Base (taken as 100%) for each of the different policies. In all cases a total of 1592 tasks were processed (50% above the initial specified as given in Table 1).

In this case, Policy 2 could be found to be the most effective. In order to understand these variations here researchers compare Policy 2 results with Policy 4 (maximum QA effort expenditure). Policy 4 uses the forecasting technique in order to adjust the project workforce depending upon the change order generation rate. The linear rise pattern of change order generation results in a progressive increase of workforce (Fig. 4a). The rookies coming in cause some decrease in productivity during the initial stages (Fig. 4b). With time, there are perceived delays in project progress arising out of productivity losses. This does not lead to hiring since in this scenario hiring is not driven by the project status. In absence of a finite schedule completion limit, the schedule pressure also does not increase. Hence, the QA activity is not curtailed and continues as long as task remains pending for QA. This longer QA duration results in the QA effort expended being higher than base (Table 3). Error detection is affected by both the pool of errors present and the productivity. In absence of late hiring, at the final stages of the project, the workforce productivity gets hampered owing to exhaustion. This

Table 3: Effect of different policies under linear rise

		Policy 1 (%)			Policy 3 (%)		
Effects	Base (%)	Sc A	Sc B	Policy 2 (%)	Sc A	Sc B	Policy 4 (%)
Qa effort (person-days)	15572 (100)	11154 (72)	11632 (75)	9294 (60)	16705 (107)	14691 (94)	17973 (115)
Rework effort (person-days)	1883 (100)	1262 (67)	1403 (75)	1472 (78)	2386 (127)	1713 (91)	2446 (130)
Completion date (days)	789 (100)	1710 (217)	1262 (160)	601 (76)	614 (78)	922 (117)	1015 (129)
FTE manpower (person)	31.5 (100)	10.2 (32)	14.6 (46)	26.5 (84)	45.2 (143)	25.3 (80)	28.9 (92)
No. of errors generated	2053 (100)	1791 (87)	1898 (92)	1824 (89)	2205 (107)	1977 (96)	2079 (101)
No. of errors detected	1662 (100)	1397 (84)	1503 (90)	1406 (85)	1818 (109)	1584 (95)	1669 (100)
QA effectiveness (No. of errors detected/person-days)	0.11	0.13 (118)	0.13 (118)	0.15 (136)	0.11 (100)	0.11 (100)	0.09 (82)

Table 4: Effect of different policies under uniform pattern

Effects	Base (%)	Policy 1 (%) Sc A	Policy 1 (%) Sc B	Policy 2 (%)	Policy 3 (%) Sc A	Policy 3 (%) Sc B	Policy 4 (%)
QA effort (person-days)	15903 (100)	11197 (70)	11939 (75)	11682 (73)	17363 (109)	15086 (95)	13879 (87)
Rework effort (person-days)	1883 (100)	1267 (67)	1403 (75)	1599 (85)	2419 (128)	1721 (91)	1679 (89)
Completion date (days)	796 (100)	1717 (216)	1294 (163)	585 (73)	619 (78)	930 (117)	1203 (151)
FTE manpower (person)	31.8 (100)	10.2 (32)	14.6 (46)	32.9 (103)	46.4 (146)	25.7 (81)	18.3 (58)
No. of errors generated	2046 (100)	1794 (88)	1894 (93)	1842 (90)	2197 (107)	1974 (96)	2035 (99)
No. of errors detected	1655 (100)	1399 (85)	1498 (91)	1429 (86)	1811 (109)	1581 (96)	1642 (99)
QA effectiveness (No. of errors detected/person-days)	0.10 (100)	0.12 (120)	0.13 (130)	0.12 (120)	0.10 (100)	0.10 (100)	0.12 (120)

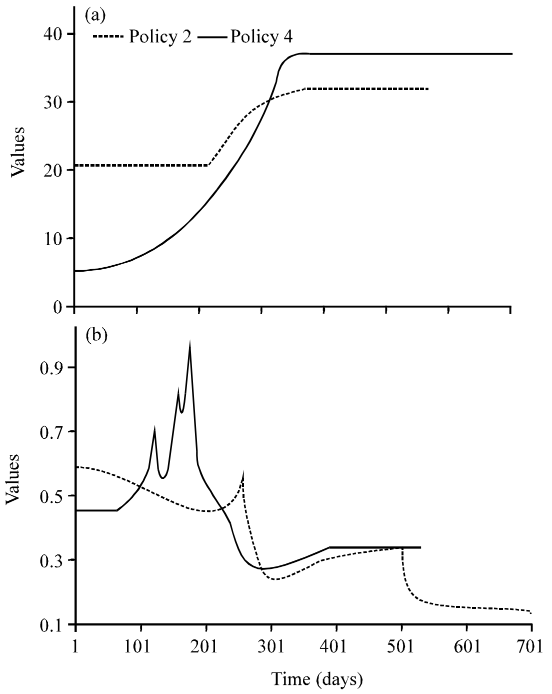

Fig. 4: Model parameter variation subjected to linear rise pattern of change order generation

causes the error detection rate to decline towards the end resulting in identification of about 1669 errors which is equivalent to the base result (Table 3).

Compared to this in case of Policy 2, the project starts with a higher-workforce (Fig. 4a). In absence of upfront hiring needs, the productivity depicts an increasing trend over the initial period, minor variations being caused by communication related losses (Fig. 4b). Because of this, tasks get processed and assigned for QA at a relatively higher rate. High productivity also causes the error generation rate to be low. With progress as project delays become visible, hiring takes place. Hiring augments the workforce size and in turn the development and the QA process, (despite productivity reductions) because of additional training overheads. All these ensure an early completion of the project (Table 3). The shorter duration of the QA phase substantially reduces the QA effort expenditure. Low error generation rate and the assigned quality objective (75%) also lead to number of errors detected to be lower in this case (Table 3). The above results indicate that with change request generation taking place in a linear rise fashion, overstaffing the project from the beginning (Policy 2) contributes towards highest QA effectiveness.

Now will the result be similar when change order generation follows uniform pattern? The simulation results are provided in Table 4 (data representations same as Table 3). In this case, the effectiveness of QA activity could be observed to be the highest under Policy 1: Scenario B (Table 4). The presence of higher workforce (Policy 1: Scenario B) ensured that the schedule pressure was not very high given the uniform pattern of change order generation during the initial stages of the project. This ensured QA process to be executed as planned and error detection was facilitated because of high productivity of the project workforce.

In real life, the pattern of change order generation can also approximate other geometric shapes for example, linear decay (initial high rate of change order generation decreases linearly with time), triangular (the rate of change order generation increases upto some point then decreases to zero), etc. Now with the results suggesting the need to adopt different resource management policies contingent on the pattern of change (overstaffing under linear rise pattern, constant workforce under the uniform pattern), new insights are expected to be obtained when carrying out similar experimentation with other patterns. Future research can look to address this.

CONCLUSION

The simulation results suggest that the effectiveness of the QA process is contingent upon the pattern of change order generation and the resource allocation policies adopted. In the current context, overstaffing led to the best results under linear rise pattern but it was not as effective under uniform pattern. However, the findings need to be interpreted in the light of its inherent limitations. The extent of variations in project parameters across the policy choices was not very significant given that the project did not have any imposed schedule penalty. Also, usage of other resource management policies not considered in this research might contribute towards greater QA effectiveness. The results are also expected to vary depending upon the project characteristics like project size, project development methodology, etc. These limitations don't undermine but rather emphasize the need to adopt contextual management approaches depending upon the expectation of change order generation in projects. Failure in this regard is likely to contribute towards user dissatisfaction related to the quality of the final deliverable or excess cost to the organization in terms of increased testing efforts at the latter stages of the project.

The study is targeted at both researchers and practitioners. The study differs from several published studies on requirement volatility by taking a pattern oriented viewpoint of the phenomenon. From a practical context, the results showcase the need to adopt contextual project management practices depending upon the project settings. If based on prior experience or supported by data of similar projects, a project manager is able to anticipate the pattern in which requirements are expected to change during project development; the study results serve as guidance to what resource allocation policy to be adopted. Efficacies of any existing policy can also be judged through simulation and corrective measures taken so as to meet the project objectives.

Researchers expect that future research will address the limitations highlighted above. Additional research can also investigate the effect of resource management policy on the total effort expended as this ultimately translates as cost to the project organization. Impacts of project development constraints like competency of available workforce, cost penalty, schedule penalty etc on the QA process can be investigated. Further, sensitivity analysis with policy parameters would provide us with additional insights on the nature of response of these levers to evolving project environment. Researchers expect that the research will stimulate interest and pave way for further dialogues that contribute towards project objectives in a meaningful way.

REFERENCES

Abdel-Hamid, T.K. and S.E. Madnick, 1991. Software Project Dynamics: An Integrated Approach. Prentice Hall, Englewood Cliffs, NJ, USA.

Abdel-Hamid, T.K., 1988. The economics of software quality assurance: A simulation-based case study. MIS Q., 12: 395-411.

Barry, E.J., C.F. Kemerer and S. Slaughter, 2006. Environmental volatility, development decisions and software volatility: A longitudinal analysis. Manage. Sci., 52: 448-464.

Basili, V.R. and H.D. Rombach, 1987. Tailoring the software process to project goals and environments. Proceedings of the 9th International Conference on Software Engineering, (ICSE'87), IEEE Computer Society Press Los Alamitos, CA, USA., pp: 345-357.

Black, L.J. and N.P. Repenning, 2001. Why firefighting is never enough: Preserving high-quality product development. Syst. Dyn. Rev., 17: 33-62.

Boehm, B.W., 1991. Software risk management: Principles and practices. IEEE Software, 8: 32-41.

Boehm, C.B., B. Boehm, B. Clark, E. Horowitz, C. Westl, R. Madachy and R. Selby, 1995. Cost models for future software life cycle processes: COCOMO2.0. Ann. Software Eng., 1: 57-94.

Collofello, J., I. Rus, A. Chauhan, D. Houston, D.M. Sycamore and D. Smith-Daniels, 1998. A system dynamics software process simulator for staffing policies decision support. Proceedings of the 31st Hawaii International Conference on System Sciences, Jan 6-9, 1998, Kohala Coast, HI, pp: 103-111.

Costello, R.J. and D.B. Liu, 1995. Metrics for requirements engineering. J. Syst. Software, 29: 39-63.

Crawford, L. and J. Pollack, 2004. Hard and soft projects: A framework for analysis. Int. J. Project Manage., 22: 645-653.

Davis, A.M., N. Nurmuliani, S. Park and D. Zowghi, 2008. Requirements change: What's the alternative? Proceedings of the 32nd Annual IEEE International Computer Software and Applications, July 28-August 1, 2008, Turku, Finland, pp: 635-638.

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

Hoorn, J.F., E.A. Konijn, H. van Vliet and G. van der Veer, 2007. Requirements change: Fears dictate the must haves; desires the won't haves. J. Syst. Software, 3: 328-355.

Joglekar, N.R. and D.N. Ford, 2005. Product development resource allocation with foresight. Eur. J. Oper. Res., 160: 72-87.

Jones, C., 1998. Estimating Software Costs. McGraw Hill, London, UK.

Kontonya, G. and I. Sommerville, 2002. Requirements Engineering Process and Techniques. Wiley Publications, UK.

Lee, Z.W., D.N. Ford and N. Joglekar, 2007. Effects of resource allocation policies for reducing project durations: A systems modeling approach. J. Syst. Res. Behav. Sci., 24: 551-566.

Li, Q., F. Shu, B. Boehm and Q. Wang, 2010. Improving the ROI of software quality assurance activities: An empirical study. ProceeOctober 31, 2012dings of the 2010 International Conference on New Modeling Concepts for Today's Software Processes: Software Process, July 8-9, 2010, Paderborn, Germany, pp: 357-368.

Liu, S., T. Tamai and S. Nakajima, 2009. Integration of formal specification, review and testing for software component quality assurance. Proceedings of the 2009 ACM Symposium on Applied Computing, March 8-12, 2009, Honolulu, HI, USA., pp: 415-421.

Madachy, R.J., 1994. A software project dynamics model for process cost, schedule and risk assessment. Ph.D. Thesis, Department of Industrial and Systems Engineering, University of Southern, California, Los Angeles.

Nurmuliani, N., D. Zowghi and S. Fowell, 2004. Analysis of requirements volatility during software development life cycle. Proceedings of the 2004 Australian Software Engineering Conference, April 13-16, 2004, Innsbruck, Austria, pp: 28-37.

Plekhanova, V., 1999. Capability and compatibility measurement in software process improvement. Proceedings of the 2nd European Software Measurement Conference, October 4-8, 1999, Amsterdam, Netherlands.

Royce, W.W., 1987. Managing the development of large software systems: Concepts and techniques. Proceedings of the IEEE 9th International Conference on Software Engineering, March 30- April 2, 1987. California, USA., pp: 328-338.

Sterman, J.D., 2000. Business Dynamics: Systems Thinking and Modeling for a Complex World. McGraw-Hill, Irwin, USA.

Thakurta, R. and F. Ahlemann, 2010. Understanding requirements volatility in software projects-an empirical investigation of volatility awareness, management approaches and their applicability. Proceedings of the 43rd Hawaii International Conference on System Sciences, January 5-8, 2010, Honolulu, HI, pp: 1-10.

Thakurta, R., R. Roy and S. Bhattacharya, 2009. Impact of requirements discovery pattern on software project outcome: preliminary results. Proceedings of the 42nd Annual Hawaii International Conference on System Sciences, January 5-8, 2009, Big Island, HI, pp: 1-6.

Wagner, S., K. Lochmann, S. Winter, A. Goeb and M. Klaes, 2009. Quality models in practice: A preliminary analysis. Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement, Ocotober 15-16, 2009, Washington, DC, USA.

Wiegers, K., 1999. Software Requirements. Microsoft Press, Redmond, USA.

Young, R.R., 2001. Effective Requirements Practices. Addison-Wesley, Boston.

Zowghi, D. and N. Nurmuliani, 2002. A study on the impact of requirements volatility on software project performance. Proceedings of the 9th Asia Pacific Software Engineering Conference, December 4-6, 2002, Queensland, Australia, pp: 3-11.