IARJSET

ISSN (Online) 2393-8021
ISSN (Print) 2394-1588

International Advanced Research Journal in Science, Engineering and Technology
Vol. 3, Issue 4, April 2016

A System Dynamics Approach towards Software Development Project- A Case Study

Pijush Chandra Das[1], U.R. Dhar[2]

Research Scholar, Business Administration, Gauhati University, Guwahati, India[1]

Retired Professor, Gauhati University, and Professor, Royal School of Business, Guwahati, India[2]

Abstract: Managing production and operations with available resources have become the most challenging job of today's software industry. This involves not only optimizing on available resources but achieving improved productivity with minimum cost of production, controlled overruns and adherence to schedule. The challenge to optimize becomes even more hard-hitting and uncompromising when the requirement changes are too frequent as in case of agile software production system. This dynamic behaviour of agile software production systems can be modelled using system dynamics (SD). The software production process can be modelled and its behaviour predicted along with consequences of managerial policies on the production system. Agile development is one of the solutions to the problem of extremely complicated methods, which is being adopted of late by various software production systems. Hence, the prime objective of this research is to investigate how the application of system dynamics can aid in the performance analysis and improvement of Agile software production systems. The paper puts forth some of the dynamic forces that affect Quality Assurance (QA) activity which are modeled using system dynamics. The feedback loops developed explain how schedule pressure in an ongoing software project, which arises when the project is behind its time schedule, leads to an increased error generation rate. With increasing number of errors committed, a major part of the available resources (man, machine, money etc.) are utilized in error correction and rework instead of development activities. This adversely affects the project's progress rate thereby leading to increase in schedule pressures and adding to greater cost of poor quality.

Keywords: Agile software development, software projects, system dynamics, change request.

I. INTRODUCTION

The software projects have always been facing excess effort overshoot, schedule pressure and increase in rework due to more of defects injected in the system because of Requirement management problem. This is due to the changing scenario of software development where requirements keep on varying between the iterations and within iteration. After functionalities have been developed for iteration the customer on reviewing it may either go for no change or go for changes. Contrary to the plan-driven cascading waterfall method for researchers and practitioner's agile methodology has become an alternative to overcome the cost, effort overrun, and schedule slippage and quality problems [22]. Agile Methods like XP-extreme programming, SCRUM and Dynamic System Development Methodology have been introduced [2][3][4]. These methods have enabled teams in quickly responding to frequent changes in the requirements [5]. The cost of accommodating a change at a later stage in the iterations is more. The cost and project risk minimizes to an extent because of the ability of agile project teams to respond fast to changes [2]. This paper studies how a System dynamic model can be used to understand the impact of requirement volatility in an agile scenario which causes frequent changes in the project and schedule pressure.

II. LITERATURE REVIEW

Rapid advances in technical environment quickly evolving system requirements demand a flexible approach like agile to develop software systems. The agile methodology is now widely in vogue in software projects but the effectiveness and appropriateness is not extensively proved through empirical research. A system dynamic model was developed taking the interdependencies of the various approaches of agile development. An integrated system dynamic model is studying the two most important aspects like refactoring and pair programming of agile methodology. Through this new integrated tool some of factors like customer involvement, refactoring, pair programming, agile planning and control and change management are being investigated thoroughly [23]. Agile approach may not suitable for large-mission critical projects because of low test coverage, focus on quick response and dearth of appropriate architecture planning [7]. The type of project, size of the project, the experience of team members and the domain knowledge level of both project team, client and also commitment of customers are the other constraints for the implementation of agile methodology [19], [20].

D. Philips has studied the impact of agile methods on the project team resources, the project itself and process implemented in the project [6]. Scrum is the most extensively used agile methodology and Lean-Kanban is the fastest growing agile methodology. Cocco et al. has done a comparative analysis of the dynamic behavior of the Kanban and scrum framework versus the traditional waterfall model [21]. Agile methodology is iterative, incremental, adaptive, self-organizing and evolving.

III. OBJECTIVE OF THE STUDY

The objectives of the study are:

1. How requirement Changes impact Agile software development Projects.
2. How interdependencies work in an agile approach and schedule and other associated variables impact.
3. Role of feedback loop in agile software projects.
4. How effective is Causal loop diagram in linking task of agile projects?

IV. METHODOLOGY

A. Model structure

Schedule pressure is a main variable of our system Dynamic Model. Schedule slippage is a very well known phenomenon in the development of application software as well as in software product development [15]. The team members feel the inadequacy of time in an agile methodology project due to the urgency of releases. Since in the beginning we are not very clear regarding the number of task associated with a project, perfect effort estimation remains a challenge. Software development is not deterministic in nature [11],[9].

When Project estimation is done, other than including the highly visible main components, project management has to emphasise on the less visible components also. As more and more task are added on as the project execution progresses the requirement for additional effort and time increases, the schedule pressure increases unless the scheduled is revised or more resource allocation is done.

B. Simulation Model

The extensively tested model of Abdel Hamid [8] is considered in our experimentation. The various enhancement modules of the development project MTR each of 120 days duration has been taken for our study and divided into iterations of multiple parts of Agility level.

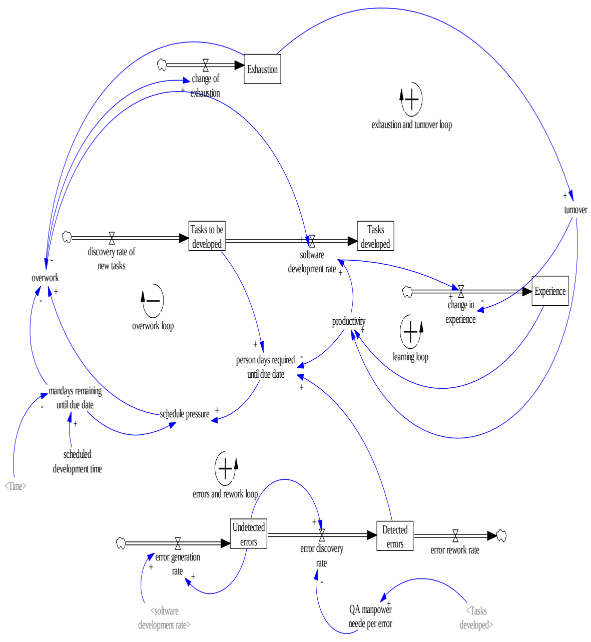

Fig. 1 Source Abdel Hamid et al. [8]

More the levels of agility as the levels of iterations increases. As the number of iterations increases more the reviews and more the number of release due dates. In waterfall life cycle development when the project realizes that they have huge schedule variance there is only one revised and final due date fixed, which leads to overwork. In an agile approach at the end of each iteration there are multiple schedule pressure peaks. Therefore due to overwork, more rework and schedule crunch is spread throughout the project life cycle. The more the effort put by working overtime the rate of software development increases. As we proceed with more iterations in the further stages the discovery of additional task reduces and hence the number remaining task to be completed reduces. Since with increasing overtime and overwork the team learns faster and hence the experience and productivity of the project team enhances [17], [12], [13], [18], [10]. The agile project iterations have multiple sprint due dates in comparison to the waterfall model which disrupts in the learning curve of the project team and also affects productivity [16].

C. COCOMO (Constructive cost model) Estimation for one enhancement of MDOCS Project

The enhancement module of the MDOCS project consists of around 18000 LOC each and the schedule duration of them is 120 days. The COCOMO equations are used to estimate the project effort in Person-days from the number of KLOC.

Planned person-days for project = 2.73*19*(perceived project size in LOC/1000)$^{1.05}$

Perceived Project size = Actual Project size in LOC X uncertainty fraction= 18000*0.65 = 11700 LOC.

Planned person-days for Project = 2.73*19(11700/ 1000)$^{1.05}$ = 687 person-days.

The number of resources is taken as 8.

As the number of task is now finally known 18000 LOC then 1078 person days are required. So with a team of 8 it would take 134 days. Therefore with 8 project team members the schedule pressure will increase and excess effort for overtime work will be required.

V. CONTEXT OF THE CASE STUDY

There are two scenarios that are considered in this case study. One is no change request received from the customer between the iterations and the other is change request received from the customer. Here the duration of the iteration of the various Enhancement modules of MDOCS project is taken as a parameter to probe its influence on schedule. The duration of each Enhancement to be delivered to the customer is 120 days. The Level of Agility (iterations) means the modules are divided into equal sizes of those many parts. In case of no change request, data is simulated from the model and the data for the requirement change (CRs) is from the enhancement module of the Live MDOCS project (primary data).

A. Data Analysis and Result Interpretation

TABLE 1 SIMULATED DATA AND DATA OF MDOCS PROJECT ENHANCEMENTS MODULE OF NO CHANGE AND REQUIREMENT CHANGE (CRS) SCENARIO.

Iterations	Iteration Duration (Days)	Defect leakage (no change)	Effort (Person Days) for NO CHANGE	Actual time (in days)	Defect Leakage (requirement changes)	Effort (Person Days) FOR CHANGES	Actual time (in days)
Iteration 1	120	217	1026	129	220	1045	131
Iteration 2	60	201	998	126	215	1032	129
Iteration 3	40	192	987	125	208	1020	126
Iteration 4	30	180	990	122	194	1014	125
Iteration 5	24	164	988	122	178	1006	125
Iteration 6	20	160	977	121	180	995	122
Iteration 7	17	172	995	123	199	1020	127
Iteration 8	15	207	1000	123	204	1030	130
Iteration 9	13	230	1015	124	254	1035	133
Iteration 10	12	249	1032	125	320	1087	140

B. Trend of Iteration duration on defect count

The defect generation is the least in-case of iteration 6. Hence Schedule pressure will reduce and rework will be less. The exhaustion in the project will decrease, productivity gradually increasing with decreasing turn over [1], [14].

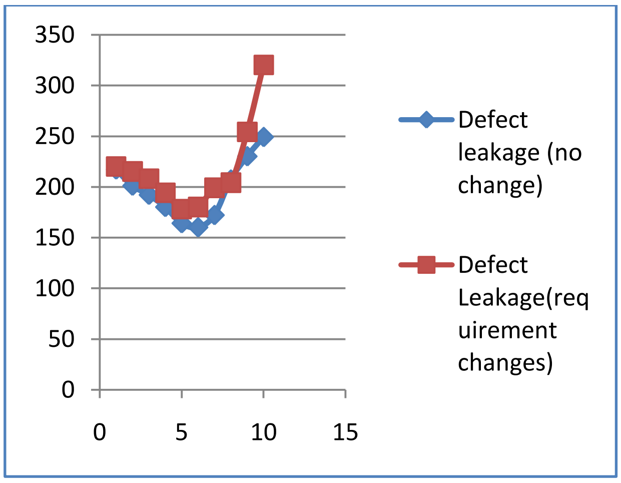

Fig. 2 Effect of iteration duration on defect count (X-axis: No. of iterations, Y-axis: defect count)

C. Trend of iteration duration on actual effort of the project

The effort for both no change context and requirement change is decreasing gradually and it is the minimal in case of iteration or agility level 6. Since for reducing effort, schedule pressure is reducing there are indication that cost of the project will be less and productivity of the team members will improve.

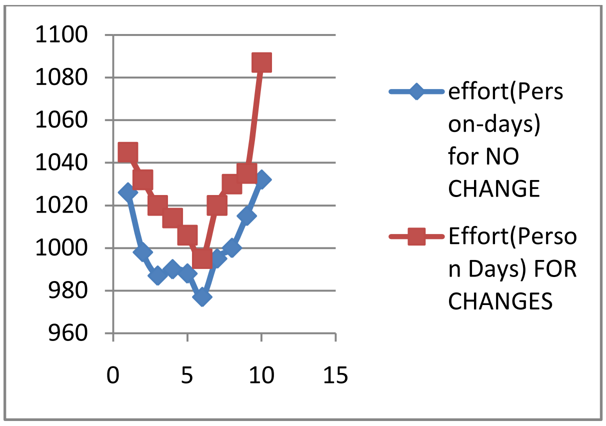

Fig. 3 Effect of iteration duration on actual effort of the project (X-axis: No. of iterations, Y-axis: effort)

VI. DISCUSSION, RECOMMENDATION AND FUTURE RESEARCH WORK

The usage of System dynamic model in case of Agile approach of software project development and finding positive and improved trend shows that there is good scope for SD in Agile development environment. The model in future needs to be implemented in case of multiple projects across multiple domains of agile approach. A new vast area of research will be thrown open if the results are visibly consistent across domains and technology. In addition to this there are also avenues to develop a comprehensive, commercialized software project management tool using system dynamics, having provision for both agile and waterfall lifecycle approach.

REFERENCES

[1] Moore, Jo E., One road to turnover: an examination of work exhaustion in technology professionals, MIS Quarterly. 24:1:141-168. 2000.
[2] K. Beck, Extreme programming explained. Addison- Wesley. 2000.
[3] J. Coplien and J. Ostergaard, SCRUM: It’s all about common sense, Scrum Training Institute, Scrum Alliance, Inc. 2009.
[4] J. Stapleton, DSDM – Dynamic System Development Method. Addison-Wesley. 1995.
[5] Paetsch, F., A. Eberlein, and F. Maurer, “Requirements Engineering and Agile Software Development,” Proceedings of the 12th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2000, pp. 308 – 313.
[6] D. Phillips, The Software Project Manager’s Handbook: Principles that work at Work, IEEE Computer Society Press. 1998.
[7] Boehm, B., Get Ready for Agile Methods, with Care, Computer, pp. 64-69. 2002.
[8] Abdel-Hamid, Tarek K., and Stuart E. Madnick, Software Project Dynamics – an integrated approach. Prentice Hall, Englewood Cliffs, New Jersey. 1991.
[9] Boehm B. and Barry W. , Software Engineering Economics. Englewood Cliffs, New Jersey: Prentice-Hall, Inc. 1981.
Boh, Wai Fong, Sandra A. Slaughter, and J. Alberto Espinosa,

"Learning from Experience in Software Development: A Multilevel Analysis," Management Science, 53(8):1315-1331.2007.

[10] Brooks Jr. and Fred P., The Mythical Man-Month, Essays on Software Engineering, University of North Carolina, Chapel Hill, Addison-Wesley Publishing Company. 1979.
Ford, David N., and John D. Sterman, "Dynamic modeling of product development processes", System Dynamics Review, 14:1:31-68.1998.

[11] Kessler, Eric H. and Paul E. Bierly, "Is faster really better? An empirical test of the implications of innovation speed," IEEE Transactions on Engineering Management.49:1:2-12. 2002.

[12] Oliva, Rogelio, and John D. Sterman, "Cutting Corners and Working Overtime: Quality Erosion in the Service Industry," Management Science. 47:7:894-914. 2001. Perlow, Leslie A., "The Time Famine: Toward a Sociology of Work Time," Administrative Science Quarterly, 44:57-81. 1999. Seshadri, Sridhar, and Zur Shapira, "Managerial Allocation of Time and Effort: The Effects of Interruptions," Management Science, 47:5:647-662. 2000.

[13] Sterman, John D., Business Dynamics - Systems Thinking and Modeling for a Complex World, Irwin McGraw-Hill, Boston. 2000.
Wiersma, Eelke, "Conditions that Shape the Learning Curve: Factors that Increase the Ability and Opportunity to Learn," Management Science, 53: 12:1903-1915. 2007.

[14] Erickson,J., Lyytinen,K. and Siau, K., "Agile modeling, agile software development and extreme programming: The state of research," Journal of Database Management, 16, 88–99. 2005.

[15] Fitzgerald,B., Hartnett,G., and Conboy, K., "Customizing agile methods to software practices at Intel Shannon," European Journal of Information Systems, 15, 200–213. 2006.

[16] L. Cocco, K. Mannaro, G. Concas, and M. Marchesi (2011), Simulating Kanban and Scrum vs. Waterfall with System Dynamics, Chapter :Agile Processes in Software Engineering and Extreme Programming, Volume 77 of the series Lecture Notes in Business Information Processing, Springer, pp 117-131. 2011.

[17] K.van Oorschot, K. Sengupta and Luk van Wassenhove, "Dynamics of Agile Software Development," Proceedings of the International Conference of the System Dynamics Society. 2009.

[18] L. Cao, B. Ramesh, and T. Abdel-Hamid (2010), "Modeling dynamics in agile software development," Journal ACM on Management Information Systems (TMIS), Article No. 5, Volume 1 Issue 1. 2010.