Applying Emergent Outcome Controls to Mitigate Time Pressure in Agile Software Development

Research-in-Progress

Onkar Malgonde, Rosann Webb Collins, and Alan Hevner
University of South Florida
Tampa, FL 33620
{omalgonde, rwcollins, ahevner}@usf.edu

Abstract

Can agile software development methods handle time pressure effectively? In this research-in-progress paper we examine the sources and remedies for time pressure in an agile software development project. We draw upon research on emergent outcome controls to understand how they can be used effectively to handle time pressure. In particular, we use Extreme Programming (XP) as an agile development exemplar and propose 3 interesting research propositions. Further, we discuss the limitations, practical implications, and future research efforts on how emergent outcome controls can be used to balance aspects of quality, time, and cost in software development.

Keywords

Agile software development, controls, time pressure, Extreme Programming

Introduction

Agility in information systems development (ISD) is defined as "the continual readiness of an ISD method to rapidly or inherently create change, proactively or reactively embrace change, and learn from change while contributing to perceived customer value (economy, quality, and simplicity), through its collective components and relationships with its environment" (Conboy 2009). The agile manifesto (Beck et al. 2001) emphasizes the importance of responding to change, customer centric development, iterative development, and interactions within and outside the development team during software development. These central ideas of agile software development drive development processes and influence decisions throughout the development cycles.

While the literature on agile methods considers time pressure as a constraint in development, how to handle time pressure or changes in time pressure has not been the focus of these studies (Abdel-Hamid 1989; Koushik and Mookerjee 1995). However, time pressure is prevalent throughout software development projects (Nan and Harter 2009), especially when first-to-market is a high priority goal (Baskerville et al. 2011). There is a lack of theoretical understanding about if, and how, agile software development mechanisms can effectively deal with a sudden or incremental increase in time pressure. In this research-in-progress paper we view effective responses to increases in time pressure as development approaches that remain faithful to agile software development as well as continue to exert proper controls over that development. To investigate the problem of increased time pressure in agile development, we employ a theoretical understanding of the controls embodied in agile development mechanisms to determine how those controls can be used to mitigate the impact of changes in time pressure. In order to do so, we focus on extreme programming (XP) as an agile development exemplar to evaluate controls in the presence of increased time pressure.

Time Pressure and its Sources

Multiple approaches have been undertaken to study time pressure in software engineering literature. As noted by Nan and Harter (2009), the impact of increased time pressure on projects is not uniform, both on development processes and outcomes. Abdel-Hamid (1989) provides a system dynamics approach to study staffing policies. These staffing policies are developed under schedule constraints and used to study the dynamics of the system under investigation. Koushik and Mookerjee (1995) develop an analytical model to study the level of coordination required in the software construction phase under given time constraints. Ji et al. (2005) study the optimal policy to develop and debug software code while adhering to time pressures like product release date. Chong et al. (2011) report that time pressure can act as a challenge (positive) or hindrance (negative) for software development. Clearly, time pressure has rightly received considerable attention. However, none of these studies have delved into if and how time pressure impacts the specific case of agile software development. Further, only Austin (2001) specifically investigates how developers deal with time pressure. Using an agency framework, Austin finds that when developers do not have to worry about being singled out for failure to meet deadlines, they can employ a strategy of quality improvement to reduce rework and thus reduce development time.

Sources of time pressure can be both internal and external. Internal time pressure is generated by actions or events within the organization that is developing the software. For example, if a software tool critical to development is made available 1 week late, the software development time frame is reduced, thereby increasing internal time pressure. Similarly, issues like staff vacations or turnover, unanticipated technical issues, or poor execution contribute to increasing internal time pressure for the entire development team.

External time pressure is generated by actions or events outside the organization that is developing the software. For example, consider a software development project that has to be completed in 2 months because of a federal or state mandated regulation. This time pressure is generated by an external source. Similarly, if market trends shift and the development team is required to deliver a software product within two weeks, we deem this time pressure as externally sourced.

Having identified the sources of time pressure, it is important to understand how the different sources of time pressure have different impacts on the development team. Following a sudden increase in time pressure that is generated internally, the development team can immediately investigate the problem. Also, when time pressure increases due to internal sources, the development team has a degree of control over the situation. For example, if the required software tool is not available on time, the team can procure a different tool or opt for an open source tool. On the other hand, when increases in time pressure are related to external source, the development team has little to no control over the situation. For example, if the delivery deadline is moved forward due to some governmental regulation or market needs, the development team may have to cut feature-set and provide the bare minimum software that is in working condition. Also, when external time pressure increases, the development team might not be informed about it explicitly by the customer or market.

Given that time pressure is a consistent factor in software development, and that there are multiple sources that may increase this pressure, it is important to understand how software development can be managed in such a dynamic setting. Emergent outcome controls (Harris et al. 2009b) offer a way to develop that understanding, since they focus on how to be flexible in the management of software development while maintaining control of the process.

> *Proposition 1: Source of increasing time pressure dictates the choice of corrective/reactive actions.*

Emergent Outcome Controls

Emergent outcome controls allow software development processes to be flexible but controlled, at the same time (Harris et al. 2009a). Emergent outcome controls are based on dynamic capabilities theory (Teece et al. 1997) and control theory (Ouchi 1977; Ouchi 1979; Ouchi 1980). The initial research identifies two prominent emergent outcome controls: (1) scope boundaries, and (2) ongoing feedback.

Scope boundaries limit the feasible solution so that the development team has the flexibility to explore but is constrained within a boundary (Harris et al. 2009b). Scope boundaries channel the development process, without dictating the outcomes. Boundaries can include overarching goals like a shared vision for the software being developed, feature specifications, as well as more technical constraints like architecture and development tools. For example, consider an agile development team that is starting its planned iteration. Before the execution can begin, the management tightens the scope boundaries by limiting the choice of tools, programming language, and application programming interface (API). Though the developers are free to be creative, they are limited to explore a constrained space patrolled by scope boundaries.

Ongoing feedback is provided within the team or from users or the market, so that the software development is on track and reaches its goals with minimum iterations. Feedback is required for development teams when scope boundaries are not sufficiently tight to resemble a feature-driven approach. For example, consider the above example of scope boundary. If we relax the programming language scope, the development team will need feedback from customers before implementing the software. This is particularly true since the choice of programming language will affect the implementation, maintenance, and integration with other systems. Often the reason for choosing an agile development approach is that there is uncertainty about the exact nature of the software to be developed (Harris et al. 2009b). Ongoing feedback is critical to systematically shape the software to fit the needs of users or market, but obtaining and reacting to feedback is time consuming (Harris et al. 2009b). If time pressures are increased, how can project managers continue to use this important development mechanism?

> *Proposition 2: During increased time pressure, the project managers tend to abandon agile software development mechanisms.*

Extreme Programming

Before we explore how emergent outcome controls can guide project managers in how to effectively mitigate time pressure, we briefly introduce Extreme Programming (XP) as an exemplar of agile software development methodology. Figure 1 (Wells 2000) provides a brief overview of XP.

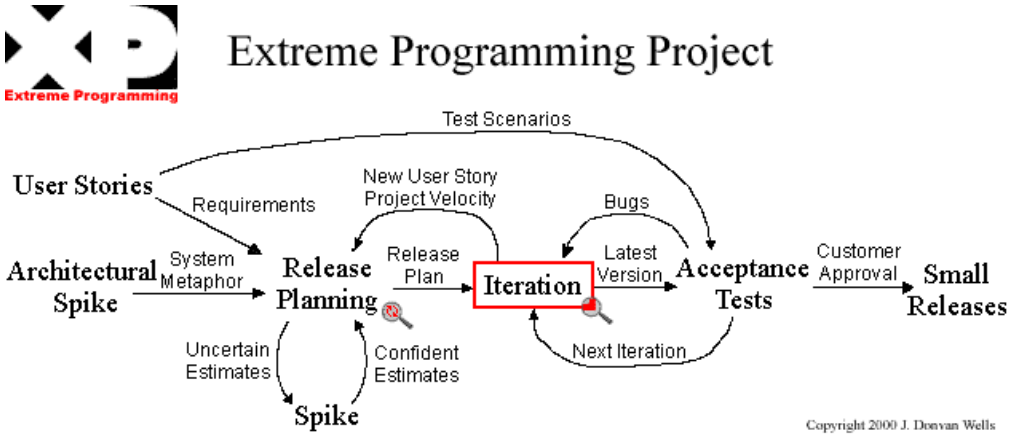

Figure 1. An Extreme Programming Process Model (from Wells (2000))

XP is characterized by principles like pair programming, continuous integration, incremental design, and the 10-minute build. Requirements are incorporated into the development process as stories. These stories are described by the customer to the development team. The development team, then conducts an analysis of these stories to order and estimate their size and priority. Development of the software is conducted in iterations, where each iteration is usually of 2 weeks. The entire team decides on the plan and deliverables of each iteration, at the start of each iteration. For the stories that have been developed

and unit tested, they are integrated and tested for user acceptance. Table 1 illustrates several XP mechanisms that we evaluate as controls for mitigating time pressure.

Extreme Programming Mechanisms	**Emergent Outcome Controls**		**How to Adapt Emergent Controls when Time Pressure Increases**	
	Scope Boundaries	**Ongoing Feedback**	**Scope Boundaries**	**Ongoing Feedback**
Sit Together		Progress observable by team members		Reduce meeting time by having only initial or periodic physical meetings, but maintain visibility through shared workspace that monitors team members' progress (may already be employed to create an *informative workspace*)
Whole Team		Feedback including customer representatives		Reduce number of feedback events by careful selection of which team members and customer representatives to include in each feedback event, while ensuring that all appropriate individuals provide feedback at least at some points
Informative Workspace		Outcomes observable		If a shared workspace that monitors team members' progress is already employed, no change; otherwise create this space
Pair Programming		New ideas are tested with partner		Encourage pairs to adapt technique to work as pairs to share ideas, but allow individual programming work as appropriate to the pair
Stories	Broad statements of intent focus efforts		Use stories to communicate changes in feature set	

1-3 Week Cycle	Limit amount of change that can occur in each iteration	Market feedback every 1-3 weeks	Review existing limits to what can be changed and reduce as appropriate	Consider whether to reduce the feedback by increasing the cycle for market feedback
Quarterly Cycle	Place business constraints as well as market constraints	Review with management. Guard against feature creep	Communicate new business or market constraints to team	Review becomes even more important given new business and/or market constraints
10-Minute Build		Make it easy to demonstrate		Maintain this control
Continuous Integration		Always be ready to demo latest product		Maintain this control
Build test cases first	Develop a detailed goal for each feature		This will be automatically reduced when the increased time pressure has resulted in a smaller feature set	
Incremental Design	Each iteration focuses on only a few things	Each iteration ready to use or demonstrate to market	While still limiting as much as possible, consider increasing what is included in each iteration	Maintain this control

Table 1. Adaption of XP Mechanisms under Time Pressure to Maintain Project Control

Proposition 3: When time pressure increases, project manager maintains control over the project through the use of emergent outcome controls.

Proposed Research Methodology

In order to validate our theoretical conceptualization and test our propositions (see Table 2), we plan to employ a critical incident method (Flanagan 1954). Specifically, we plan to interview software development project managers about incidents in which there was an increase in time pressure, sudden or incremental, and record their actions to mitigate the impacts of increased time pressure. In particular, we will ask project managers about incidents where the sources of increased time pressure include internal and external sources. We will ask the managers to compare their use of agile methods in the project before and after the time pressure increases.

Based on their responses and the outcomes, we can identify and validate the existence of emergent outcome controls. Further, based on the interview transcripts, we can identify additional ways in which emergent outcome controls can be used to prescribe actions to project managers to help mitigate increases in time pressure.

1	Source of increasing time pressure dictates the choice of corrective/reactive actions
2	During increased time pressure, the project managers tend to abandon agile software development mechanisms.
3	When time pressure increases, project manager maintains control over the project through the use of emergent outcome controls.

Table 2. Research Propositions

Discussion

Understanding the impact of time pressure and how it can be effectively mitigated is of practical importance. Agile software development, with its iterative and incremental approach, has become a mainstream software development methodology. In this paper, we have highlighted the issue of time pressure in agile software development. This is of particular concern for agile development projects, since time to completion can take more time than more traditional, plan-driven development (Harris et al, 2009a). We illustrate that by understanding how agile development methods embody project controls, we can adapt and use agile mechanisms to effectively mitigate increased time pressure while maintaining proper software development control. Our conceptualization of this link between time pressure, emergent outcome controls and agile development is based on prior literature. The next step is to collect empirical data on how project managers respond to increases in time pressure in agile development projects, and the relationship between the type of response and (a) the effectiveness of project controls and (b) project outcomes. In addition, we will consider whether the source of time pressure has differential impacts on how project managers respond, how effectively the project is controlled, and project outcomes.

Future Research Directions

Future research in the context of agile methods also has the potential to identify additional emergent outcome controls, since the very nature of agile methods is that they embrace a dynamic development environment. Such additional controls that are identified may also be those that help mitigate time pressure. We believe that there can be a portfolio of emergent outcome controls that can be used to effectively control agile software development, while maintaining its flexibility in the face of change. Also, the effect of emergent outcome controls on other constraints like cost, and quality, can be examined. This is particularly interesting because the effects of time pressure, cost, and quality are interdependent (Figure 2).

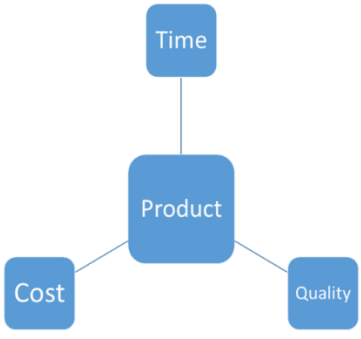

Figure 2. The Software Development Golden Triangle

An increase in time pressure affects the quality and cost of the project simultaneously. Similarly, an increase in quality requirements implies increase in time pressure (or an extended due date), and increased costs. Thus the constraints of time pressure and increases in time pressure during development become a lever for investigating these relationships and how project managers maintain control in such dynamic settings.

References

Abdel-Hamid, T. 1989. "The Dynamics of Software Project Staffing: A System Dynamics Based Simulation Approach " *IEEE Transactions on Software Engineering* (15:2), pp. 109-119.

Baskerville, R., Heje-Pries, J., and Madsen, S. 2011. "Post-Agility: What Follows a Decade of Agility?," *Information and Software Technology* (53:5), pp. 543-555.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J., and Thomas, D. 2001. "Manifesto for Agile Software Development." Retrieved 02/13/2014, from www.agilemanifesto.org

Chong, D., Van Eerde, W., Chai, K.H., and Rutte, C.G. 2011. "A Double-Edged Sword: The Effects of Challenge and Hindrance Time Pressure on New Product Development Teams," *IEEE Transactions on Engineering Management* (58:1), pp. 71-86.

Conboy, K. 2009. "Agility from First Principles: Reconstructing the Concept of Agility in Information Systems Development," *Information Systems Research* (20:3), pp. 329-354.

Flanagan, J. 1954. "The Critical Incident Technique," *Psychological Bulletin* (51:4), pp. 327-358.

Harris, M.L., Collins, R.W., and Hevner, A.R. 2009a. "Agile Methods: Fast-Paced, but How Fast?," *AMCIS 2009 Proceedings*).

Harris, M.L., Collins, R.W., and Hevner, A.R. 2009b. "Control of Flexible Software Development under Uncertainty," *Information Systems Research* (20:3), pp. 400-419.

Ji, Y., Mookerjee, V., and Sethi, S. 2005. "Optimal Software Development: A Control Theoretic Approach," *Information Systems Research* (16:3), pp. 292-306.

Koushik, M., and Mookerjee, V. 1995. "Modeling Coordination in Software Construction: An Analytical Approach," *Information Systems Research* (6:3), pp. 220-254.

Ouchi, W. 1977. "The Relationship between Organizational Structure and Organizational Control," *Administrative Science Quarterly* (22:1), pp. 95-113.

Ouchi, W. 1979. "A Conceptual Framework for the Design of Organizational Control Mechanisms," *Management Science* (25:9), pp. 833-848.

Ouchi, W. 1980. "Markets, Bureaucracies, and Clans," *Administrative Science Quarterly* (25:1), pp. 129-141.

Teece, D., Pisano, G., and Shuen, A. 1997. "Dynamic Capabilities and Strategic Management," *Strategic Management Journal* (18:7), pp. 509-533.

Wells, D. 2000. "Extreme Programming Project."