Scaling Agile Software Development Through Lean Governance

Scott W. Ambler
IBM Canada
scott_ambler@ca.ibm.com

Abstract

Agile project teams are potentially easy to govern than traditional project teams due to the greater visibility and opportunities to steer provided to stakeholders. Unfortunately, traditional IT governance strategies prove to be at odds to agile's collaborative, value-focused strategies. As a result a lean approach based on enablement, collaboration, and motivation is required to effectively govern agile teams.

1. Introduction

Governance isn't something that is commonly associated with agile software development projects. Yet agile projects, like all other projects, can and should be governed. Unfortunately the agile literature rarely mentions governance and when it does it has little good to say about the topic, likely a reflection of the current state of maturity of the agile mainstream. Recently I have worked to define a three level agile process maturity model to help distinguish the various agile methodologies so as to make their strengths and weaknesses explicit.

Level 1 includes agile processes such as Scrum, Extreme Programming (XP), and my own Agile Modeling (AM) which address only a portion of the development lifecycle. The focus of Scrum is project leadership and requirements management, XP is on construction, and AM is on modeling and documentation – all important parts of the overall system development lifecycle (SDLC), but nowhere near a complete picture.

Level 2 processes go further by covering the full agile system development lifecycle (SDLC) [1]. This includes disciplined agile software development processes such as Dynamic System Development Method (DSDM), Open Unified Process (OpenUP), and agile instantiations of the IBM Rational Unified Process (RUP) process framework. DSDM is an agile development process often used to developer user interface intensive application; OpenUP combines Scrum, XP, and strategies from RUP for co-located agile teams, and RUP is a comprehensive process framework for iterative software development.

Level 3 addresses disciplined agile development processes applied at scale, which include tailored forms of level 2 processes as well as Enterprise Unified Process (EUP). It is important to recognize that team size is only one of several issues that agile teams face at scale. At IBM we have expanded upon the scaling factors identified by Eckstein [2] which an agile team may face to a lesser or greater extent, to include team size, physical distribution, organizational distribution, regulatory compliance, cultural or organizational entrenchment, system complexity, and enterprise disciplines (such as enterprise architecture, strategic reuse, and portfolio management). Our experience is that your approach to governance is one of several factors which impact your ability to scale agile software development techniques.

2. Agile and Governance

There are two aspects of agile software development that promote superior levels of governance when compared to traditional software development [3]. First is greater stakeholder visibility into the project. The agile strategy of producing working, potentially shippable software on a regular basis provides stakeholders explicitly reveals the actual accomplishments of the project team. Stakeholders don't have to rely on the promises made in traditional status reports, or in comprehensive specifications, but instead can determine if the team is actually delivering value by examining working software. Furthermore, disciplined agile development processes, the level 2 processes, often include explicit decision gates n the form of milestones. Effective milestones are based on reducing project risks, such as coming to stakeholder agreement on project goals or

978-1-4244-3736-8/09/$25.00 © 2009 IEEE

proving the architectural strategy through a working end-to-end skeleton of the system, as opposed to delivery of documentation. In short, the level 1 strategy of delivering working software on a regular basis and the level 2 strategy risk-based milestones greatly increases stakeholder visibility.

The second governance enabler is more opportunities for stakeholders to steer the project, potentially making them accountable for the scope, budget, and even schedule of agile project teams. Stakeholders can be made responsible for the scope because agile teams adopt flexible requirements management techniques which enable stakeholders to modify, add, remove, or reprioritize their requirements at any time throughout the project. Stakeholders can be made responsible for the budget because each iteration agile teams ask them how much they want to invest that iteration, effectively putting them in control of the budget. Stakeholders can be made responsible for the schedule because a side effect of producing potentially shippable working software each iteration, thereby giving them the option to choose when enough functionality exists for deployment.

3. Lean Development Governance

The opportunity to govern agile projects is clearly present, but that begs the question how to do so effectively. In 2006 through 2007 Per Kroll and I developed a development governance framework which is based on the philosophical foundation provided by the seven principles of lean software development [4]. These principles are to eliminate waste, build quality in, create knowledge, defer commitment, deliver fast, respect people, and optimize the whole. Our experience was that these lean principles provided the insight required for effective governance of agile project teams, and potentially traditional project teams as well. Traditional governance is outside the scope of this paper.

The heart of the Lean Development Governance framework [5] is that good governance should motivate, and then enable, IT professionals to do what your organization believes to be the correct behaviors. The framework identifies 18 practices which support this strategy. These practices are: Promote Self-Organizing Teams, Align Team Structure With Architecture, Align HR Policies With IT Values, Align Stakeholder Policies With IT Values, Adapt the Process, Continuous Improvement, Embedded Compliance, Iterative Development, Risk-Based Milestones, Simple and Relevant Metrics, Continuous Project Monitoring, Business-Driven Project Pipeline, Pragmatic Governance Body, Staged Program Delivery, Scenario-Driven Development, Valued Corporate Assets, Flexible Architectures, and Integrated Lifecycle Environment.

4. Lean Governance Enables Agility at Scale

Agile processes, like traditional ones, can breakdown when one or more scaling factors is present. Lean governance practices such as aligning the team structure with the architecture, risk-based milestones, and staged program delivery address complexities inherent in large or distributed teams. Aligning IT values with stakeholder policies and a pragmatic governance body can alleviate cultural and organizational entrenchment. Practices such as continuous project monitoring, integrated lifecycle environment, and embedded compliance help to address the additional complexity of regulated environments. Scenario driven-development, staged program delivery, continuous improvement, and valued corporate assets address the complexities inherent in supporting effective enterprise disciplines. In short, a new governance approach is needed to effectively support a new development paradigm.

5. References

[1] S.W. Ambler, "The Agile System Development Lifecycle (SDLC)", www.ambysoft.com/essays/agileLifecycle.html, last accessed March 1, 2009.

[2] J. Eckstein, "Agile Software Development in the Large - Diving Into the Deep", Dorset House, New York, 2004.

[3] S.W. Ambler, "Governing Agile Software Development", Dr. Dobb's Journal, San Francisco, November 2007.

[4] M. Poppendieck and T. Poppendieck, "Implementing Lean Software Development: From Concept to Cash", Addison Wesley, 2006.

[5] S.W. Ambler and P. Kroll, "Lean Development Governance", www14.software.ibm.com/webapp/iwm/web/preLogin.do?lang=en_US&source=swg-ldg, last accessed March 1, 2009.