The Journal of Systems and Software 113 (2016) 418–429

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Aligning codependent Scrum teams to enable fast business value delivery: A governance framework and set of intervention actions

Jan Vlietland[a], Rini van Solingen[b], Hans van Vliet[c,*]

[a] Search4Solutions B.V., Professional Services, Utrecht, The Netherlands
[b] Department of Electronics, Mathematics and Computer Science, Delft University of Technology, Delft, The Netherlands
[c] Department of Computer Science, VU University Amsterdam, Amsterdam, The Netherlands

ARTICLE INFO

Article history:
Received 27 September 2014
Revised 6 October 2015
Accepted 7 November 2015
Available online 19 November 2015

Keywords:
Agile
Chain codependencies
Collaboration
Coordination
Alignment

ABSTRACT

Many enterprises that adopt Agile/Scrum suffer from collaboration issues between Scrum teams that depend on one another to deliver end-to-end functionality. These dependencies delay delivery and as a result deteriorate the business value delivered in such value chains. The objective of our study is to support enterprises that suffer from such dependencies with a governance framework that helps them mitigate collaboration issues between sets of codependent Scrum teams. We first identify a set of intervention actions that aim to mitigate the collaboration issues between codependent Scrum teams. Second, we validate the effectiveness of these intervention actions in a large confirmatory industrial case study. This study was held in a large multi-national financial institute that worked with a large number of codependent Scrum teams. Third, we triangulate the findings in three focus groups. We finally package the intervention actions in a governance framework. The intervention actions led to a delivery time reduction from 29 days to 10 days. The participants in the focus groups confirmed the causality between the intervention actions and the observed delivery improvement. The empirical results show that the intervention actions, packaged in the governance framework, enable codependent sets of Scrum teams to deliver faster.

1. Introduction

Large companies operating in information intensive industries experience rapid changing business demands, requiring swift adaption of front to back (business) value chains. Since these value chains are automated with IT services, the rapid changing business demands require flexible IT services. The IT services that enable these front to back value chains are delivered by a portfolio of interdependent applications. That application portfolio is typically delivered by multiple codependent IT service providers (ISP). IT service changes therefore often require software development staff of multiple ISPs (Plugge & Janssen, 2009; TFSC, 2011). These ISPs have to jointly execute a fast paced software development process (Moniruzzaman & Hossain, 2013; Pikkarainen et al., 2005).

To achieve a fast paced software development process, many internal IT development centers increasingly transfer to Agile methods. The most common Agile method used in industry is the Scrum software development method (VersionOne, 2013). Scrum is an incremental method that uses low boundary cross-functional collaboration in software development teams that work toward a set team goal (Sutherland & Schwaber, 2013). Scrum works with fixed iterations of less than one calendar month that deliver working and tested increments, resulting in faster delivery.

Scrum teams can be mapped in different ways onto the (interdependent) application portfolio. Some prefer a single Scrum team for all interdependent applications that support the front to back value chain (Sutherland, 2005). Two constraints make such coverage difficult. Firstly, the number of involved IT staff (typically from different ISPs) then easily exceeds the generally agreed upon maximum Scrum development team size of 9 members. Secondly, changes typically require highly specialized skills (due to a complex IT landscape with multiple commercial-off-the-shelf items) that cannot be easily shared within a single team. The solution chosen in companies is to set up dedicated Scrum teams. Each Scrum team then develops one or more applications in the portfolio that automates a part of the front to back value chain (Vlietland & van Vliet, 2015). Together, the applications developed by multiple Scrum teams result in value-adding features. Features are defined as 'intentional distinguishing characteristics of the application landscape that can be used by a business user' (IEEE, 2008), e.g. a mortgage registration feature.

* Corresponding author.
E-mail addresses: j.vlietland@Search4Solutions.nl (J. Vlietland), d.m.vansolingen@tudelft.nl (R. van Solingen), hans@cs.vu.nl (H. van Vliet).

http://dx.doi.org/10.1016/j.jss.2015.11.010
0164-1212/© 2015 Elsevier Inc. All rights reserved.

Since features are the result of codependent software development activities by multiple Scrum teams, collaboration between the teams is needed. Particularly the high frequency of deliveries common in Scrum settings makes collaboration a performance factor (Dorairaj et al., 2012). Yet, due to the nature of Scrum teams, such collaboration might not happen naturally. A Scrum team has specific characteristics, such as a maximum of 9 members, a multidisciplinary setup, allocated IT applications, high-frequency deliveries and focus on a single product backlog (Sutherland & Schwaber, 2013). These characteristics typically limit the focus of a Scrum team, resulting in collaboration issues (Vlietland & van Vliet, 2015).

Vlietland and van Vliet (2015) identified six blocking issues in sets of codependent Scrum teams. In the current study, the next step is taken. A set of intervention actions (IAs) for sets of codependent Scrum teams to support a front to back value chain is developed. The IAs aim to mitigate the blocking issues, reducing the delivery time of new features by the set of Scrum teams. The IAs are solidly embedded in organizational change and performance improvement intervention literature.

The IAs are validated in a large confirmatory case study with a set of codependent Scrum teams at a multinational financial institute, during a period of 9 months. After deploying the IAs the delivery time was reduced from 29 days to 10 days. The effect of the IAs, measured in this case-study, was triangulated in focus groups consisting of the members of the codependent Scrum teams. These focus group sessions confirmed that the observed reduction was a direct result of the IAs. The results indicate the effectiveness of the IAs. The IAs were subsequently packaged into the Scrum Value chain Framework (SVF) to support practice in deploying the IAs.

The remainder of this article is organized as follows. Section 2 covers the related work for developing the IAs and the related work regarding (Agile) governance frameworks. Section 3 develops the intervention actions (IAs). Section 4 provides an overview of the empirical results. Section 5 discusses the results and presents the Scrum Value chain Framework (SVF). Section 6 summarizes the threats to validity. Section 7 concludes the study, deduces implications and suggests future research avenues.

2. Related work

Three areas of related work are studied. First, an overview of organizational change literature is given, to position the research design and the IAs in Section 3. Second, the Agile IA literature related to intended performance improvements is studied, to extract the IAs from the Agile literature in Section 3. The section closes with related work on Agile governance frameworks to develop the Scrum Value chain Framework (SVF) and validate the completeness of the IAs in Section 5.2.

2.1. Organizational change literature

Three perspectives on change in the organizational change literature are identified: (1) the tempo of change, (2) planned versus spontaneous change and (3) top-down versus bottom-up change. After introducing these three perspectives, a deeper analysis is performed on a combination that fits this case study, while introducing learning theory as catalyst for organizational change. The section closes with a summary of the change design for this case study.

Tempo of change: One perspective on organizational change is the tempo of change (Weick & Quinn, 1999). At one end of the spectrum is evolutionary change, which involves a relatively long stream of small changes in reaction to the changing environment, as first modeled by Darwin. Evolutionary change in organizations progresses continuously. Revolutionary change at the other end of the spectrum happens in short bursts (Hannan & Freeman, 1984). One theory in the area of revolutionary change is the theory of inertia and punctuated equilibrium (Romanelli & Tushman, 1994). In case an organization does not evolutionary follow the changing environment, the organization gets disconnected from the environment and tends to an inert equilibrium state (Gersick, 1991). In such a state it is hard to change the organization. After some time, strategic reorientation is required to realign the organization with the environment, resulting in a revolutionary change. For such revolutionary change the inert equilibrium needs to be punctuated. After the inertia is broken, the organization experiences a turbulent change to find a new equilibrium, closer aligned with the environment.

Planned versus spontaneous change: A related perspective to evolutionary and revolutionary change is planned versus spontaneous change. Spontaneous change occurs without a set purpose. Each individual actor interacts with other actors and the system changes through evolution (Stacey, 1995). At the other end there is planned change. The actors together aim to achieve a planned state.

Top-down versus bottom up: A perspective related to planned change is top-down versus bottom-up change. Yamakami (2013) analyzed organizational change initiatives in the IT industry and identifies three types of initiatives (1) top-down, in which top management takes initiative, (2) bottom-up, in which the work floor staff takes own initiatives to realize change, and (3) a hybrid approach.

Deeper analysis: Cummings and Worley (2014) elaborate on planned change as a way to change organizations. They identify two planned change strategies (1) a positivistic approach with an unfreezing, moving and freezing phase and an (2) interpretivistic approach with iterations and feedback loops (Jrad et al., 2014). Positivistic based change paradigms have long dominated the IT industry, such as CMMI (Team, 2010) and ISO 9000 (Hoyle, 2001). The positivist paradigm uses a machine metaphor in which input is transformed to output (Ilgen et al., 2005; Stelzer & Mellis, 1998). The paradigm stimulated the use of detailed prescribed work processes which can be quantitatively measured, analyzed and controlled (Unterkalmsteiner et al., 2012). A positivistic approach works in areas of high predictability. The intrinsic human intensive activity of software development with high levels of unpredictability and uncertainty however seems a misfit with such a positivistic paradigm (Clarke & O'Connor, 2013). That misfit was answered at the beginning of this century when the interpretivistic based Agile paradigm got momentum (Akbar et al., 2011). The Agile paradigm uses a bottom-up, continuous change paradigm to utilize human capital in the software development industry (Van Tiem, et al., 2006). Agile is supported with iterations and feedback loops to increase the evolutionary change capability (Qumer & Henderson-Sellers, 2008). Baskerville and Wood-Harper (1996) and Baskerville (1999) specify cyclical action research based on the description of Susman and Evered (1978). Their research design consists of five phases: diagnosing, action planning, action taking, evaluating and specifying learning. The five phases are repeatedly executed to allow adaptation of the change strategy during each cycle.

Learning as catalyst: Experience-based learning can be seen as catalyst for organizational change in Agile environments. Kolb (1984) uses three models of experiential learning for developing a model that combines experience, perception, cognition and behavior. His experience learning model consists of four phases: (1) concrete experience, (2) reflective observation, (3) abstract conceptualization and (4) active experimentation. The capacity to reflect on past experience is one of the key principles for continuous learning in Agile environments (Holz & Melnik, 2004; Salo & Abrahamsson, 2005). Such reflective practice exists in different development disciplines on individual, team and organizational level. For instance a Scrum team conducts a demo and notices that the Product owner repeatedly struggles with a drag and drop action. Such observation allows the team to rethink the functionality and experiment another solution. Qumer and Henderson-Sellers (2008) similarly argue that