The Journal of Systems and Software 000 (2016) 1–14

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

Going with the flow: An activity theory analysis of flow techniques in software development

Denis Dennehy*, Kieran Conboy

J.E. Cairnes School of Business & Economics, National University of Ireland Galway, Upper Newcastle Road, Galway city, Ireland

ARTICLE INFO

Article history:
Received 21 March 2016
Revised 29 September 2016
Accepted 4 October 2016
Available online xxx

Keywords:
Activity theory
Lean
Flow
Kanban
Continuous software development

ABSTRACT

Managing flow is fundamental to continuous development, particularly in knowledge intensive work activities such as software development. However, while numerous articles describe flow tools and practice there is little research on their application in context. This is a significant limitation given that software development is a highly complex and socially embedded activity. This research applies activity theory (AT) to examine the adoption of flow techniques by using the multiple-case method in two companies. AT is particularly pertinent in this study as it identifies contradictions, which manifest themselves as problems such as errors or a breakdown of communication in the organisation and congruencies between flow techniques and the development context and indeed contradictions between components of flow techniques themselves. Rather than view contradictions as a threat to flow or as an argument to abandon, a theoretical contribution of this study is that it shows how contradictions and congruencies can be used to reflect, learn, and identify new ways of structuring and enacting the flow activity. It also provides an immediate practical contribution by identifying a set of lessons drawn from the cases studied that may be applicable in future implementations of flow techniques.

1. Introduction

Flow is a fundamental tenet of contemporary lean thinking, and is seen as the primary method to transition from agile to true continuous software development (Anderson, 2013; Fitzgerald and Stol, 2015; Olsson and Bosch, 2014; Poppendieck, 2002; Reinertsen, 2009; Tichy et al., 2015). Flow is about managing a continuous stream of value creating activities throughout the entire development process (Anderson, 2010; Reinertsen, 2009; Petersen and Wohlin, 2011; Poppendieck, 2002). This approach differs from traditional project management through its emphasis on managing queues rather than project phases and milestones (Power and Conboy, 2015; Anderson, 2013; Anderson et al., 2011; Ikonen et al., 2011). Flow techniques have been well received by those in software development, and there is evidence to suggest that awareness and indeed use of these methods is becoming quite prevalent in software development practice (Anderson, 2013; Nord et al., 2012; Petersen and Wohlin, 2011; Poppendieck and Cusumano, 2012; Power and Conboy, 2015; Reinertsen, 2009). For example, the flow concept is useful in considering continuous software engineering which emphasizes a continuous movement, rather than focusing on agile methods *per se* (Fitzgerald and Stol, 2015; Fitzgerald and Stol, 2014). Flow encourages collaboration between teams, measurement of value, costs, and technical metrics, and knowledge sharing, which are key characteristics of DevOps (Bang et al., 2013).

Initial claims that evidence to support flow techniques remain largely anecdotal in the software development field (Ebert et al., 2012; Sjøberg et al., 2012; Ikonen et al., 2011) are now being addressed. A systematic review by Ahmad et al. (2013) showed nineteen relevant studies to 2011. There are other relevant studies before and since (Nord et al., 2012; e.g. Petersen and Wohlin, 2011; Petersen et al., 2014; Sjøberg et al., 2012; Staron and Meding, 2011; Khurum et al., 2014) where Kanban adoption and usage (Cocco et al., 2011; Shinkle, 2009; Senapathi et al., 2011), the effect of bottleneck detection (Rutherford et al., 2010; Petersen et al., 2014) and visualisation of development on metrics such as velocity (Anderson et al., 2011; Power and Conboy, 2015; Polk, 2011) are examined.

However, the existing body of knowledge is limited by the fact that most studies only focus on specific elements of flow e.g. Kanban boards, bottlenecks or Cumulative Flow Diagrams (CFDs) or tend to focus on the textbook 'vanilla' version of these tools. While they demonstrate the effect of flow they usually do not examine the tensions and contradictions that often arise when multiple elements of these techniques are implemented as part of an overall flow programme. Activities are not isolated elements; they are influenced by other activities and other changes in their environment. For example, Kanban by itself does not guarantee success as

* Corresponding author.
E-mail address: denis.dennehy@nuigalway.ie (D. Dennehy).

http://dx.doi.org/10.1016/j.jss.2016.10.003
0164-1212/

it is a relatively basic control tool that needs to be supported by additional practices (Ikonen et al., 2011).

The depth and complexity of this interplay between activities is exacerbated further given the socially embedded contextual nature of software development. It is well accepted that culture, team dynamics, the development team's 'softer' skills and the development and solution context are critical determinants of software development success and that a method, practice or tool cannot be studied in isolation (Conboy, 2009; Ebert et al., 2012; Fitzgerald et al., 2002; Kitchenham et al., 2002; Lyytinen and Rose, 2006; Petersen and Wohlin, 2009). These issues indicate a need, not just to study flow techniques, but also to study them in the environment within which they must be implemented in and cognisant of in order to be effective.

This research draws on AT as it provides a theoretical lens specifically designed to study (i) multiple activities and practices in a multilevel, stratified manner, and (ii) to identify the contradictions and congruencies that emerge when people interact and use artifacts (e.g. Kanban board, cumulative flow diagrams) in a social context (Kaptelinin, 1996; Karanasios and Allen, 2014; Mursu et al., 2007; Spasser, 2002).

AT has inspired a number of theoretical reflections on software development (Allen et al., 2013; Bertelsen and Bødker, 2000; Chen et al., 2013; Hasan, 1998; Korpela et al., 2001; Kuutti and Molin-Juustila, 1998; Vermeulen et al., 2016). Ultimately, contradictions and congruencies interrupt or enable the fluent flow of work (Helle, 2000) and so AT is therefore particularly pertinent in the context of this study. Contradictions "indicate a misfit within elements, between them, between different activities or different development phases of a same activity" (Kuutti, 1995, p. 28). The resolution of contradictions within and between activities acts as a driver of change and may result in several levels of congruency which shape the success, normalisation and agreement between the use of the tool and the work activity (Hasan et al., 2010; Karanasios and Allen, 2014). This study examines contradictions and congruencies between flow techniques and the development context or indeed contradictions between the flow techniques themselves.

When we see how AT is constructively used in other fields (Foot, 2001; Helle, 2000; Kuutti, 1995), rather than view contradictions as a threat to flow techniques or as an argument to abandon them, contradictions can be viewed as an opportunity to reflect, to learn, and to identify new ways of structuring and enacting the flow activity. The lens of contradictions and congruencies within AT enable us to generate two interlinked research questions in the context of flow and its use in practice, namely:

(1) *What is the motivation for adopting flow techniques?*
(2) *What are the major contradictions in the implementation and use of flow techniques?*
(3) *What are the major congruencies in the implementation and use of flow techniques?*

The next sections of the paper summarize the pertinent flow and AT literature and describe the theoretical basis and research approach adopted in this study. The findings from two 'revelatory' case studies are then presented and discussed, and through further analysis, the emerging set of (i) contradictions and (ii) congruencies in the implementation and use of flow techniques are presented. The paper concludes with a discussion of the implications and limitations of the study and possible avenues for future research.

1.1. Background to the concept of lean

Rooted in lean manufacturing, leanness has largely focused on cost reduction (Ohno, 1988), "the elimination of waste" (Naylor et al., 1999; Ohno, 1988; Womack et al., 1990), and "doing more with less" (Towill and Christopher, 2002). However, the concept of lean has evolved over time with subsequent emphasis on the value and more recently the flow of work. Lean strives to deliver maximum value to the customer by reducing waste, controlling variability, maximizing the flow of information, focusing on the whole process, and not on local improvements (Anderson et al., 2011; Poppendieck, 2002). Lean is a mind-set, a mental model of how the world works (Poppendieck and Poppendieck, 2013). Lean thinking is guided by five interlinked concepts: value; value stream, flow, pull, and perfection (Wang et al., 2012).

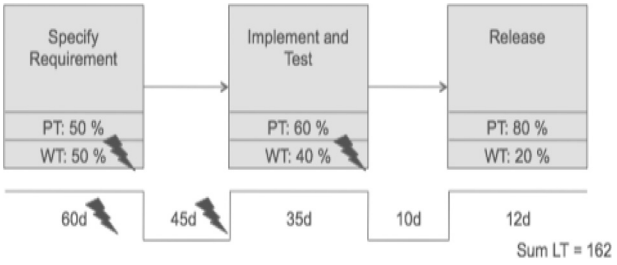

Fig. 1. Value stream mapping (source: Khurum et al., 2014, p. 6).

1. Value: It is defined by the customer and it is paramount to have a clear understanding of what that is.
2. Value stream: A map that identifies every step in the process and categorises each step in terms of the value it adds
3. Flow: It is important that the production process flows continuously.
4. Pull: Customer orders pull product, ensuring nothing is built before it is needed.
5. Perfection: Striving for perfection in the process by continuously identifying and removing waste.

1.2. Principles of lean software development

Lean software development is a relatively new addition to the agile method family but is increasingly being adopted by software teams (Anderson et al., 2011). Poppendieck and Poppendieck (2003) published the first book that adopted lean principles from manufacturing and applied them to software development was adapted from lean production which consists of seven principles: 1) eliminate waste, 2) amplify learning, 3) decide as late as possible, 4) deliver as fast as possible, 5) empower the team, 6) build integrity, and 7) see the whole. These principles were later refined and are listed in Table 1 below.

Flow in lean product development is defined as "the progressive achievement of tasks along the value stream so that a product proceeds from design to launch, order to delivery, and raw materials into the hands of the customer with no stoppages, scrap, or backflows" (Womack and Jones, 2010, p. 306). Managing a continuous and smooth flow that delivers value to the customer is a key aspect of lean principles (Anderson, 2010; Petersen and Wohlin, 2011; Reinertsen, 2009). This results in production not being based on forecasts as commitment is delayed until demand is present to indicate what the customer really wants (Poppendieck, 2002).

1.3. Related work

There are five commonly known techniques that are used to identify flow bottlenecks: (i) value stream maps, (ii) Kanban board, (iii) cumulative flow diagrams (CFDs), (iv) burn-down charts, and (v) line of balance status charts (Petersen et al., 2014).

Value stream maps: Value stream maps (see Fig. 1) are used to follow a specified item of work through the process in order to establish value added in each processing step (Petersen et al., 2014).