The Journal of Systems and Software 000 (2015) 1–14

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

# Continuous software engineering: A roadmap and agenda

Brian Fitzgerald, Klaas-Jan Stol*

*Lero—the Irish Software Research Centre, University of Limerick, Ireland*

ARTICLE INFO

*Article history:*
Received 11 November 2014
Revised 24 April 2015
Accepted 25 June 2015
Available online xxx

*Keywords:*
Continuous software engineering
Lean software development
DevOps

ABSTRACT

Throughout its short history, software development has been characterized by harmful disconnects between important activities such as planning, development and implementation. The problem is further exacerbated by the episodic and infrequent performance of activities such as planning, testing, integration and releases. Several emerging phenomena reflect attempts to address these problems. For example, Continuous Integration is a practice which has emerged to eliminate discontinuities between development and deployment. In a similar vein, the recent emphasis on DevOps recognizes that the integration between software development and its operational deployment needs to be a continuous one. We argue a similar continuity is required between business strategy and development, *BizDev* being the term we coin for this. These disconnects are even more problematic given the need for reliability and resilience in the complex and data-intensive systems being developed today. We identify a number of continuous activities which together we label as 'Continuous *' (i.e. *Continuous Star*) which we present as part of an overall roadmap for Continuous Software engineering. We argue for a continuous (but not necessarily rapid) software engineering delivery pipeline. We conclude the paper with a research agenda.



## 1. Introduction

Software development has been characterized by harmful disconnects between important activities, such as planning, analysis, design and programming. This is clearly reflected in the traditional waterfall process for software development described (and criticized) by Royce (1987). In the last two decades, there has been a widespread recognition that increasing the frequency of certain critical activities helps to overcome many challenges. Practices such as 'release early, release often' are well established in open source software development (Feller et al., 2005), which offer several benefits in terms of quality and consistency (Michlmayr et al., 2015). The pervasive adoption of agile methods (Kurapati et al., 2012; Papatheocharous and Andreou, 2014) provides ample evidence of the need for flexibility and rapid adaptation in the current software development environment. Very complex and business- and safety-critical software is being developed, often by distributed teams. A tighter connection between development and execution is required to ensure errors are detected and fixed as soon as possible. The quality and resilience of the software is improved as a result. This is manifest in the increasing adoption of continuous integration practices. The popularity of continuous integration is facilitated by the explicit recommendation of the practice in the Extreme Programming agile method (Beck, 2000), and indeed the practice is highly compatible with the frequent iterations of software produced by agile approaches. Also, many open source toolsets such as Jenkins CI[1] are freely available to automate the continuous integration process which makes this practice readily available to potential adopters.

However, a number of recent trends illustrate that a more holistic approach is necessary rather than one which is merely focused on continuous integration of software. For example, the *Enterprise Agile* (Overby et al., 2005) and *Beyond Budgeting* (Bogsnes, 2008) concepts have emerged as a recognition that the benefits of agile software development will be sub-optimal if not complemented by an agile approach in related organizational functions such as finance and HR (Leffingwell, 2007; Overby et al., 2005). In a similar vein, the recent emphasis on *DevOps* recognizes that the integration between software development and its operational deployment needs to be a continuous one (Debois, 2009). Complementing this, we argue that the link between business strategy and software development ought to be continuously assessed and improved, "BizDev" being the term we coin for this process. Several researchers across the management, information systems and software engineering disciplines have provided arguments which reinforce the BizDev concept and we elaborate on this in Section 4 below.

Furthermore, the holistic approach mentioned above should not assume that the process is complete once customers have initially

* Corresponding author. Tel.: +353 61 233737.
*E-mail addresses:* bf@lero.ie (B. Fitzgerald), klaas-jan.stol@lero.ie (K.J. Stol).

[1] https://jenkins-ci.org/

http://dx.doi.org/10.1016/j.jss.2015.06.063
0164-1212/

**Table 1**
Seven additional practices to scale agile to the enterprise level according to Leffingwell (2007).

| Practice | Description |
|---|---|
| Intentional Architecture | For large systems consisting of many subsystems, agile components team fit their components into an intentional architecture, which is component-based and is aligned with the team's core competencies, physical location and distribution. |
| Lean requirements at scale | Non-functional/cross-cutting system requirements such as performance and reliability must be considered and understood by all teams contributing to the system. Define a *vision* stating the overall objectives; a *roadmap* that defines an implementation strategy, and defer specific requirements as long as possible, implement them *just-in-time*. |
| Systems of systems and the agile release train | Create a release schedule with fixed dates that is imposed upon all teams, and leave decisions regarding the delivered features/functionality up to those teams. |
| Managing highly distributed development | Large-scale software development is inherently distributed, as developers are necessarily located in different physical locations, ranging from different rooms, floors, and buildings to different countries and time zones. Additional coordination practices and enterprise-level tool support are necessary. |
| Impact on customers and operations | More frequent delivery has an impact to a variety of stakeholders, including sales and marketing, operations, support, distribution organizations, and ultimately customers. The interaction with each of these stakeholders must be considered and adapted to fit enterprise-level agile adoption. |
| Changing the organization | Transforming the enterprise to an agile one requires a combination of top-down leadership, bottom-up adoption and expansion and empowered managers in the middle, all with a common vision. |
| Measuring business performance | Whereas the key measure of small-scale agile is the presence of working software, enterprise agile requires a number of additional measures to monitor efficiency, value delivery, quality, and agility. |

adopted a software product. Digital natives, the term for those who have been born in the technology era (Vodanovich et al., 2010) have high expectations of software and are not put off by the high switching costs associated with moving to alternatives. Frequently, third-party opinions and word-of-mouth can cause customers to switch, and software providers must be more proactive in such a market-place. Also, privacy and trust issues loom much larger in the data-intensive systems being used today. Run-time adaptation is increasingly a factor as software is expected to exhibit some degree of autonomy to respond to evolving run-time conditions.

We believe that rather than focusing on agile methods *per se*, a useful concept from the lean approach, namely that of 'flow' (Reinertsen, 2009) is useful in considering continuous software engineering. Rather than a sequence of discrete activities, performed by clearly distinct teams or departments, the argument for continuous software engineering is to establish a continuous movement, which, we argue, closely resembles the concept of flow found in lean manufacturing and product development (Morgan and Liker, 2006), a school of thought that is called 'Lean Thinking' (Womack and Jones, 2003). In recent years, there has been much interest in lean software development (Conboy and Fitzgerald, 2004; Fitzgerald et al., 2014; Maglyas et al., 2012; Petersen, 2011; Wang et al., 2012), however, there has been a frequent tendency to adopt a somewhat narrow view on the topic by closely linking it to agile practices only.

In this paper we review a number of initiatives that are termed 'continuous.' This paper extends our preliminary view on this topic (Fitzgerald and Stol, 2014). The goal of this paper is to sketch a holistic view of these initiatives and position them within the continuous software engineering context, and illustrate how Lean Thinking is a useful and relevant lens to view continuous software engineering. Furthermore, we look beyond software development, and consider issues such as continuous use, continuous trust, etc. and coin the term 'Continuous *' (pronounced as "Continuous Star") to refer to this holistic view.

The various developments are by and large at different levels of maturity—continuous integration is a concept and practice that has gained widespread currency, probably in large part due to it being a formal practice in XP. However, recent research shows that different organizations implement this practice in different ways (Ståhl and Bosch, 2013). In contrast, continuous delivery is an idea that has not widely been established in the software industry. Thus, we consider our holistic overview as a conceptual research agenda, and we envisage future research to operationalize each of the concepts identified, much as Ståhl and Bosch have initiated for continuous integration.

This paper proceeds as follows. Section 2 describes a number of developments that have attempted to scale up the agile paradigm, such as Enterprise Agile, Beyond Budgeting and DevOps. Section 3 reviews a number of key concepts from the school of Lean Thinking, of which the concept of 'flow' is the most important as it can be used as a suitable foundation for the holistic concept of 'Continuous *' that we propose in this paper. Section 4 presents the activities which comprise Continuous * in more detail, and we observe how the various concepts of lean thinking can be identified within these activities. Section 5 discusses a number of directions for future research and implications for practice.

## 2. Trends in the software engineering landscape

A number of recent trends have focused on the need to transform organizations to deliver software more rapidly. We discuss these in turn.

### 2.1. Enterprise agile

Agile methods were initially considered to be only suitable for small teams, and research on agile methods has long focused quite narrowly on the software development function only. In recent years, several authors have identified the need to scale the agile concept to the enterprise level (Kettunen and Laanti, 2008; Reifer et al., 2003). Leffingwell (2007), for example, documented a set of seven practices to complement the practices that are common to agile methods such as Scrum, XP and DSDM (see Table 1). These seven principles address several dimensions in which agile approaches should scale, such as the link to other functions of the organization (e.g., marketing, operations), the product (e.g., architecture, requirements), and the development process (e.g., distributed development). Leffingwell also initiated the Scaled Agile Framework (SAFe).[2] The SAFe is based on experiences of organizations that have adopted agile at enterprise scale and describes practices and activities, roles, and artifacts.

Overby et al. (2005) defined 'enterprise agility' as *"the ability of firms to sense environmental change and respond appropriately,"* hence identifying the two main components of 'sensing' and 'responding' appropriately. As organizations may possess different capabilities to sense and respond, these can be seen respectively as two dimensions along which organizations may be positioned. For instance, an organization may have well-developed capabilities to *sense* new market

[2] http://www.scaledagileframework.com/