International Journal of Information Management 40 (2018) 186–189

Contents lists available at ScienceDirect

International Journal of Information Management

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

A case analysis of enabling continuous software deployment through knowledge management

Ricardo Colomo-Palacios[a], Eduardo Fernandes[b], Pedro Soto-Acosta[c,*], Xabier Larrucea[d]

[a] Department of Computer Science, Østfold University College, B R A Veien 4, 1783 Halden, Norway
[b] Meta4 Spain, Centro Europa Empresarial, Edificio Roma, C/. Rozabella, 8, 28290 Las Rozas Madrid, Spain
[c] Department of Management & Finance, University of Murcia, Campus de Espinardo, 30100 Espinardo, Murcia, Spain
[d] Tecnalia, Parque Tecnológico de Bizkaia. Calle Geldo, Edificio 700, E-48160 Derio, Bizkaia, Spain

ARTICLE INFO

Keywords:
Knowledge management
Continuous deployment
Continuous software engineering
DevOps
Case study

ABSTRACT

Continuous software engineering aims to accelerate software development by automating the whole software development process. Knowledge management is a cornerstone for continuous integration between software development and its operational deployment, which must be implemented using sound methodologies and solid tools. In this paper, the authors present and analyse a case study on the adoption of such practices by a software company. Results show that, beyond tools, knowledge management practices are the main enablers of continuous software engineering adoption and success.

1. Introduction

In order to preserve their competitive advantage, software producers need to deliver products and new features to customers as fast as they can. It is generally accepted that important problems in software delivery are rooted, among other aspects, in the disconnections among software development activities, causing delays in software delivery (Fitzgerald & Stol, 2017). This lack of connection lies not only on the technical side, where human aspects and knowledge management facets are some of the main areas to be improved. Continuous software engineering permits software features delivery at rates which a few years ago would have been considered unachieveable (Colomo-Palacios, Fernandes, Soto-Acosta, & Sabbagh, 2011, p. 4; O'Connor, Elger, & Clarke, 2017). This approach is based heavily on applying automation to the overall software development process (including code collaboration tools, verification, version control system, deployment and release management…) by using several tools. These tools act as structures in which different types of knowledge are coded and shared among software practitioners.

Like any other approach, continuous deployment presents benefits but also caveats. On the benefits side, the literature reports: Increased customer satisfaction, shorter time-to-market, higher developer productivity and efficiency, continuous rapid feedback and, finally, higher quality and reliability. With regard to the challenges, researchers found the wide panoply of tools available and their integration, organizational culture to be a hindrance to the transformation process and increased quality assurance efforts.

The continuous approach goes beyond the borders of traditional software development to reach the operational side as well. In this scenario, DevOps stands for a continuous integration between software development (Dev) and its operational deployment (Ops). DevOps efficiently integrates development, delivery, and operations, thus facilitating a lean and fluid connection of these traditionally separated silos (Ebert, Gallardo, Hernantes, & Serrano, 2016). Consequently, DevOps implies a cultural shift toward collaboration between development, quality assurance, and operations (Ebert et al., 2016). The success of DevOps is based on four principles (Humble & Molesky, 2011):

- Culture. Joint responsibility for the delivery of high quality software.
- Automation. Automation in all development and operation steps towards rapid delivery and feedback from users.
- Measurement. All process must be quantified to understand delivery capability and proposals of corrective actions should be formulated for improving the process.
- Sharing. Sharing knowledge enabled by tools is crucial.

Accordingly, knowledge management is one of the pillars of DevOps and must be implemented using sound methodologies and solid tools. The literature has reported specific knowledge management systems

* Corresponding author.
E-mail addresses: ricardo.colomo-palacios@hiof.no (R. Colomo-Palacios), eduardofer@meta4.com (E. Fernandes), psoto@um.es (P. Soto-Acosta), xabier.larrucea@tecnalia.com (X. Larrucea).

https://doi.org/10.1016/j.ijinfomgt.2017.11.005
Received 24 October 2017; Accepted 17 November 2017
Available online 26 November 2017
0268-4012/