



Continuous Delivery: Overcoming adoption challenges[☆]

Lianping Chen

Lianping Chen Limited, Dublin, Ireland



ARTICLE INFO

Article history:

Received 29 August 2016

Revised 23 December 2016

Accepted 16 February 2017

Available online 20 February 2017

Keywords:

Agile Software Development

Continuous Delivery

Continuous Deployment

Continuous Software Engineering

DevOps

Adoption

ABSTRACT

Continuous Delivery (CD) is a relatively new software development approach. Companies that have adopted CD have reported significant benefits. Motivated by these benefits, many companies would like to adopt CD. However, adopting CD can be very challenging for a number of reasons, such as obtaining buy-in from a wide range of stakeholders whose goals may seemingly be different from—or even conflict with—our own; gaining sustained support in a dynamic complex enterprise environment; maintaining an application development team's momentum when their application's migration to CD requires an additional strenuous effort over a long period of time; and so on. To help overcome the adoption challenges, I present six strategies: (1) selling CD as a painkiller; (2) establishing a dedicated team with multi-disciplinary members; (3) continuous delivery of continuous delivery; (4) starting with the easy but important applications; (5) visual CD pipeline skeleton; (6) expert drop. These strategies were derived from four years of experience in implementing CD at a multi-billion-euro company. Additionally, our experience led to the identification of eight further challenges for research. The information contributes toward building a body of knowledge for CD adoption.

© 2017 The Author. Published by Elsevier Inc.

This is an open access article under the CC BY license. (<http://creativecommons.org/licenses/by/4.0/>)

1. Introduction

Continuous Delivery (CD) is a software engineering approach in which teams keep producing valuable software in short cycles and ensure that the software can be reliably released at any time (Chen, 2015a).

The CD approach is relatively new. It started gaining wide attention only in 2010, when Humble and Farley published the book titled “Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation” (Humble and Farley, 2010). However, the CD approach has become increasingly popular, as shown by Google search trends (Fig. 1).

Companies that practice CD have reported huge benefits, such as significant improvements in time-to-market, customer satisfaction, product quality, release reliability, productivity and efficiency, and the ability to build the right product through rapid experiments (Chen, 2015a; Leppanen et al., 2015).

These benefits have motivated many companies to adopt CD. According to a recent survey of 600 software developers, managers, and executives in the United States and the United Kingdom, only 3% of the respondents said they had no plans to adopt CD (Perforce Software Inc., 2015).

However, implementing CD can be quite challenging (Chen, 2015a; Leppanen et al., 2015; Claps et al., 2015). Although CD as a goal (a target state) is no longer a new idea and has been well documented (Humble and Farley, 2010), the adoption journey for CD is not yet a smooth path. The journey itself is where the challenges lie, where many companies struggle, and where practitioners need help. Some of the fundamental challenges include the following:

- acquiring buy-in from a wide range of stakeholders whose goals may seemingly be different from—or even conflict with—those of the team driving the CD implementation;
- gaining sustained support in a dynamic complex enterprise environment;
- maintaining an application development team's momentum when their application's migration to CD requires an additional strenuous effort over a long period of time.

To help overcome the adoption challenges, I present six strategies we learned along our journey to CD at a multi-billion-euro company called Paddy Power. Paddy Power has been in the process of implementing CD for the past four years and, consequently, we have encountered and overcome many challenges. However, eventually, we achieved huge benefits (Chen, 2015a).

I hope the strategies described here can help fellow practitioners to overcome similar challenges on their way to achieving CD. The information also contributes toward building a body of knowledge for CD adoption.

[☆] The work was done when the author was at Paddy Power PLC.
E-mail address: lianping.chen@outlook.com

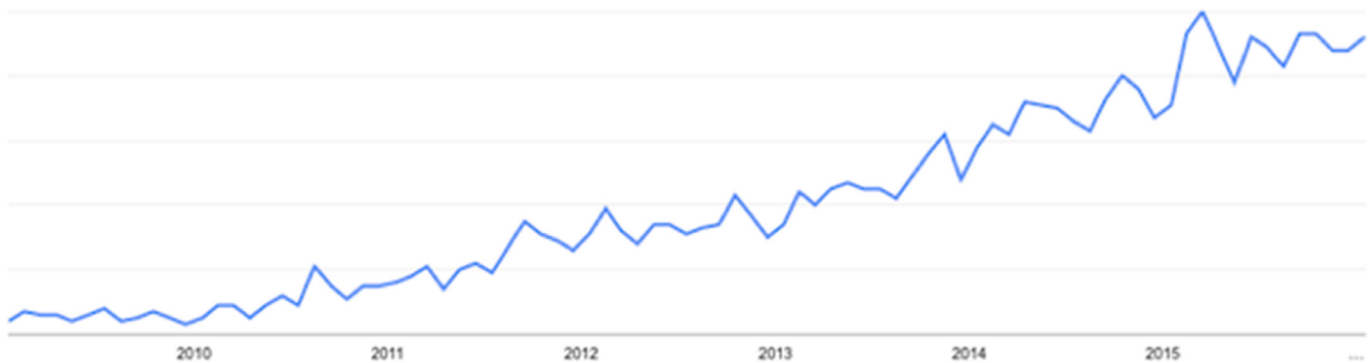


Fig. 1. Five-year Google trend for the search term "Continuous Delivery".

I also outline eight further challenges for research. Although not all of them are new, I provide elaboration on why these challenges are important for the industry, which on its own right can be useful input to researchers.

The remainder of this paper is organized as follows: Section 2 describes the context from which the strategies described here were derived. Section 3 describes the six strategies. I discuss related work in Section 4 and outline further research challenges in Section 5. Section 6 summarizes the paper.

2. The context

2.1. Paddy Power

Paddy Power is a rapidly growing company in the bookmaking industry. It offers its services in regulated markets, through betting shops, phones, and the Internet. It has an annual turnover of approximately €7 billion and employs approximately 5000 people. Paddy Power recently merged with Betfair, making it the world's largest public online betting and gaming company.

The company relies heavily on an increasingly large number of custom software applications that include websites, mobile apps, trading and pricing systems, live-betting-data distribution systems, and software used in betting shops. We developed these applications using a wide range of technology stacks, including Java, Ruby, PHP, and .NET. To run the applications, the company has an IT infrastructure consisting of thousands of servers in different locations. The applications are developed and maintained by the Technology Department, which employs approximately 500 people.

2.2. Continuous Delivery

At Paddy Power, we view Continuous Delivery as a software engineering approach in which teams keep producing valuable software in short cycles and ensure that the software can be reliably released at any time (Chen, 2015a). The following sections highlight the characteristics of CD that are important to us.

2.2.1. Valuable software

Developing valuable software is a goal that has long been on the Agile manifesto (Beck et al., 2001). However, it is not an easy goal to achieve. Before adopting CD, some of our teams had been using an Agile method called Kanban (Anderson, 2010); however, due to delivery problems, we still had situations where a team had completed a feature but could not deliver it to production to obtain users' feedback. Consequently, they built additional functionalities on top of that feature, simply assuming it was useful. Unfortunately, when they finally delivered the software to the users, they found out that the feature was not what the users needed. Even worse, by that point, significant effort had been spent on the

feature and the additional functionalities. An important objective of our CD implementation was to alleviate this problem. We want teams to build valuable software rather than spend time on features that users do not need.

2.2.2. Short cycles

We use "cycle time" to refer to the time from the conception of a user story to its production. Our cycle time used to be multi-month. Because of this long cycle time, the user stories that were completed earlier in the cycle had to wait for a long time. The value they could otherwise have generated was lost. Moreover, we were unable to obtain early user feedback. Therefore, shortening the cycle time is important.

2.2.3. Releasable at any time

Before implementing CD, applications were only releasable at the end of a long release cycle. This constraint caused problems. For example, business people could not obtain a release to users on demand in the middle of this long cycle, no matter how desperately they needed it; users were simply forced to wait until the next big release to obtain some important features—although those features might have been developed early, at the beginning of the cycle. To eliminate these problems, we want applications to be releasable at any time.

2.2.4. Reliable releases

Our emphasis on reliable releases stem directly from bad release experiences. Each time we were about to release, we had little confidence in the release reliability. Many times, these releases would be followed by P1 (priority 1) incidents (Rob, 2007), meaning that release activity was always full of uncertainty, failures, and stress. Reliable releases are a big point of distinction between our old practices and CD.

A term very similar to CD exists: Continuous Deployment. In academia, people tend to use it interchangeably with Continuous Delivery (Rodríguez et al., 2016).

However, many practitioners tend to clearly distinguish these two terms. The distinction is that under Continuous Deployment we deploy any change to production that passes a series of tests. In contrast, under Continuous Delivery, we ensure that the software can be reliably released at any time, but it is up to a human to decide when to release. In both Continuous Deployment and Continuous Delivery, deployment to production itself is automated. The difference lies in the trigger for making the deployment. One is triggered automatically, while the other is triggered by a human. According to this distinction, Continuous Delivery is compatible with a wide range of scenarios, but Continuous Deployment is suitable only under special conditions (O'Dell and Skelton, 2016). As do most companies, we mainly use Continuous Delivery.

Download English Version:

<https://daneshyari.com/en/article/4956479>

Download Persian Version:

<https://daneshyari.com/article/4956479>

[Daneshyari.com](https://daneshyari.com)