# Enhancing Continuous Delivery through Release Candidates

**Daniel D. R. Barros, Flávio Horita**

*Federal University of ABC (UFABC). Av. dos Estados, 5001 – Santo André – São Paulo – Brazil – 09210-580*

{*daniel.barros, flavio.horita*}*@ufabc.edu.br*

**Abstract:** Agile practices have been decreasing the spent time on software customers used to wait, however, they are not enough. Modern and digital customers request quick and assertive final products, demanding Continuous Delivery (CD) from software companies. Adopting CD is not straightforward, but a Release Engineering Pipeline (REP) can support it allowing improvements. Therefore, the interest in CD and REP has been increasing. This paper proposes enhancing CD through Release Candidates (RC), an important markup in REP showing that software will be live soon. The contextualization and methodology are provided, as well as the conclusion and next steps this study aims at exploring.

## 1. Introduction

The digital customers increasing demand for fast and zero bug products required software companies the adoption of agile best practices and CD. Although agile methodology was initially faced as a liability, software companies are increasingly used to it, adopting them as the main software deliverable policy [3]. CD generated interest attempting to satisfy end-users needs [1]. The recent advances in REP were motivated by industry, but there is a lot of space for researching too [1]. The **motivation** of this paper is enhancing CD, supported by REP, through one of releases best practices called RC. Adding RC in the main pipeline is the main **contribution**, as this can decrease the workload and role around delivering software.

CD was motivated by the influence of agile development, and the need to receive software assertively and quicker [2]. There is a general consensus around CD definition in the literature, however, some authors tend to use interchangeably the concept of Continuous Deployment and CD, which they are clearly different [3]. While Continuous Deployment is about automation, understood by the ability to deliver software frequently learning from customers usage [4], CD deals with the whole value chain proposing a logical progression to automate software delivery [3]. This study uses the Krusche and Alperowitz CD definition [5], where they see CD as a set of theory and implementation to release software quicker and more often.

REP is defined by an established release engineering process, represented by pipeline phases. Adams and McIntosh proposed a pipeline [1] describing individual roles and practices for every 6 major phases cyclically: (1) Integration; (2) Continuous Integration; (3) Build System; (4) Infrastructure-as-Code; (5) Deployment; (6) Release.

Release is the REP final stage in a software CD strategy [1]. Releases can use labels in order to identify maturity level and purpose (such as Alpha, Beta, Candidate or Final), and these labels can mean something to business and technical teams [6]. RC are software versions with all conditions to be a final product, unless a critical issue appears before the production upgrade [6].

## 2. Methodology

The methodology suggested involves taking RC to enhance CD supported by REP [1]. REP supporting CD already presumes agile practices, however, local improvements can be made to maximize each milestone (i.e. Release) decreasing delivering time, increasing software reliability and boosting user experience. Fig. 1 shows RC inserted into REP, and how it can help the decision of sending the production version to customers or not.

RC as a decision-making artifact can contribute in many ways to enhance CD in REP process: (i) it digitizes the release managers' handwork decreasing role complexity (more supervision instead of hands-on tasks), releasing the massive workload around the job. In other words, release managers can assume more roles, as part of their manual tasks of go or no go to live are partially delegate to a decision making process; (ii) the process of releasing a version to production should be quicker, as partially there is no more manual decision making; (iii) adding machine learning

algorithms to predict RC will increase the feedback over software availability to stakeholders, as they will be able to guess when a version will be upgraded to production.

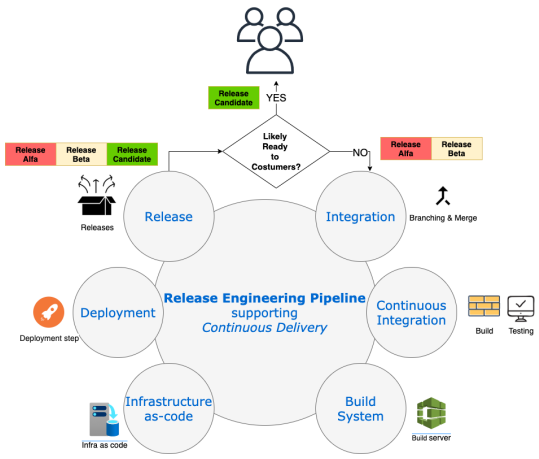


Fig. 1. REP supporting CD [1], and how it is enhanced through RC.

## 3. Conclusion

This paper proposes enhancing CD through RC. CD can be boosted using REP, but digital customers always demand more. RC is an option to increase release delivery and, therefore, improve CD significantly. Predicting RC is not something new, as Barros et al. [7] proposed a way to discover DevOps trends from repositories. Thus, adding RC in CD, supported by REP, can definitely enhance software delivery and make a better user experience.

Future lines of research should expand the theory and practice of RC being used to enhance CD. This approach can be tested in Open Source Software projects hosted at social coding sites, like GitHub, where CD data and REP are public and available for the community usage and analysis.

## 4. References


[1] B. Adams and S. McIntosh, "Modern Release Engineering in a Nutshell – Why Researchers Should Care," in *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, Vol. 5 of 2016, pp. 78-90.

[2] J. Humble and D. Farley, "Continuous delivery: reliable software releases through build, test, and deployment automation,", Pearson Education, 2010

[3] P. Rodríguez, A. Haghighatkhah, L. E. Lwakatare, S. Teppola, T. Suomalainen, J. Eskeli, T. Karvonen, P. Kuvaja, J. M. Verner and M. Oivo, "Continuous deployment of software intensive products and services: A systematic mapping study," in *Journal of Systems and Software*, Vol. 123 of 2017, pp. 263-291.

[4] H. H. Olsson and H. Alahyari and J. Bosch, "Climbing the "Stairway to Heaven" – A Mulitiple-Case Study Exploring Barriers in the Transition from Agile Development towards Continuous Deployment of Software," in *2012 38th Euromicro Conference on Software Engineering and Advanced Applications*, 2012, pp. 392-399.

[5] S. Krusche and L. Alperowitz, "Introduction of Continuous Delivery in Multi-Customer Project Courses," in *Companion Proceedings of the 36th International Conference on Software Engineering*, 2014, pp. 335–343.

[6] J. R. Erenkrantz, "Release management within open source projects," in *Proceedings of the 3rd Workshop on Open Source Software Engineering*, 2003, pp. 51-55.

[7] D. D. R. Barros, F. Horita and D. G. Fantinato, "Data mining tool to discover DevOps trends from public repositories: Predicting Release Candidates with gthbmining.rc," in *Proceedings of the 34th Brazilian Symposium on Software Engineering (SBES '20)*, 2020.