UNIVERSITY OF MINNESOTA
SENG 5852

Continuous Integration, Delivery, & Deployment: What Does It Mean?

RESEARCH PAPER

LUE XIONG

May 9, 2019

Contents

1 Cultural Shift in the Software Industry

The software industry is transforming at a rapid pace to accommodate the dynamic nature of the market and as a result, it continues to struggle to find process-identity with continuous software engineering.

Software engineering has for two decades, experimented with the concept of distributing software in faster release cycles; endeavoring to do so without sacrificing reliability and security. To achieve such a goal, there has been a widespread movement in the technical community to advocate for using Agile practices, and in particular: continuous integration, delivery, and deployment. The traditional methods of software development no longer meet the need of businesses that – now more than ever – want to proactively engage and retain their customers. The organizational transition to Agile practices demands a large mentality change and require individuals to recognize software as incremental features developed with cross-collaboration of small comprehensive team units, as opposed to large modules developed by siloed units.

2 Born from Agile

The word 'Agile' can be quite a jumbled terminology depending who a person talks to. Software professionals have their own experiences and stories of using it in their day-to-day work practices with varying degrees of opinions and outcomes. The Agile Manifesto is a declaration of the principles of Agile, which has a principle stating that the highest priority is satisfying customers with early and often continuous delivery (Meyer, 2014, p. 50). It is apparent however, that at the heart of Agile is the value of producing software at a quick and efficient pace, without a sacrifice to a team's sustainability. One such software development method that has been existent for over two decades, and fundamentally an expression of the principles of the Agile Manifesto,is Extreme Programming (XP).

Extreme Programming involves a set of practices including test-first development, automated testing, fast release cycles, refactoring, continuous integration, and among other things. Software professionals have embraced these set of practices for its effectiveness, even if arbitrarily and selectively picking from its' set. Bertrand Meyer, a well-respected individual in the software engineering community, considers many of the XP practices to be 'brilliant'; even going as far as to say that the practices of continuous integration and testing are major factors for the success of modern software projects (Meyer, 2014, p. 154). Software professionals and organizations have embodied these ideals to keep up with the demands of the software market, and have effectively put them into action.

Of the ideal software practices come commonly known and used vocab-

ulary: continuous integration, delivery, and deployment. Many software professionals however, have misunderstood the above vocabulary and at times, infuse them as one concept without careful and thoughtful distinction. It would therefore, be wise to define each component for what they represent and why it matters.

- What is Agile?
  - (Bosch, 2014) & (Meyer, 2014)
- What are the core ideas of Agile?
  - (Meyer, 2014)
- How does Agile tie in with CI/CDE/CD?
  - (Shahin, Babar, & Zhu, 2017) & (*Continuous Delivery, Deployment & Integration: 20 Key Differences*, 2018)

2.0.1 Continuous Integration

- What does CI mean?
  - (Shahin et al., 2017) & (*Continuous Delivery, Deployment & Integration: 20 Key Differences*, 2018)

2.0.2 Continuous Delivery

- What does CDE mean?
  - (Shahin et al., 2017) & (*Continuous Delivery, Deployment & Integration: 20 Key Differences*, 2018)

2.0.3 Continuous Deployment

- What does CD mean?
  - (Shahin et al., 2017) & (*Continuous Delivery, Deployment & Integration: 20 Key Differences*, 2018)

2.1 Differences of Interpretation & Implementation

2.1.1 Viewpoint of Software Professionals

- How do software professionals interpret and implement CI/CD/CDE?
  - (Atkinson & Edwards, 2018) & (*Continuous Delivery, Deployment & Integration: 20 Key Differences*, 2018)

2.1.2 Viewpoint of Academic Researchers

- How do academic researchers interpret and believe how CI/CD/CDE should be implemented?
  - (Bosch, 2014), (Shahin et al., 2017), & (Stahl, 2017)

2.1.3 Collaboration Effort

- What effort is there to bridge the phenomena of non-collaboration between developers and researchers?
  - (Bosch, 2014) & (Stahl, 2017)

2.2 Benefits of Continuous Integration, Delivery, & Deployment

2.2.1 Self-healing Systems

- What are the metrics and tools that software professionals use to mitigate having to manually fix software issues?
  - (Bosch, 2014)
- How do these self-healing systems work?
  - (Bosch, 2014)

2.2.2 Risk Reduction

- How does continuous software engineering reduces risk in systems?
  - (Atkinson & Edwards, 2018), (Bosch, 2014), (*Continuous Delivery, Deployment & Integration: 20 Key Differences*, 2018), & (Stahl, 2017) (*Continuous Delivery, Deployment & Integration: 20 Key Differences*, 2018)

2.2.3 Faster Release Cycles

- How are faster release cycles are achieved?
  - (Atkinson & Edwards, 2018), (Bosch, 2014), (*Continuous Delivery, Deployment & Integration: 20 Key Differences*, 2018), & (Stahl, 2017) (*Continuous Delivery, Deployment & Integration: 20 Key Differences*, 2018)

2.2.4 Overall Cost Reduction

- Why will all of the above will reduce cost?

  - (Atkinson & Edwards, 2018), (Bosch, 2014), (*Continuous Delivery, Deployment & Integration: 20 Key Differences*, 2018), & (Stahl, 2017) (*Continuous Delivery, Deployment & Integration: 20 Key Differences*, 2018)

2.3 Struggles of Traceability

2.3.1 Importance

- What is the importance of traceability for the software engineering community?
  - (Stahl, 2017) & (D. Stahl, Hallen, & Bosch, 2016)

2.3.2 Problem of Mapping

- What is the problem of mapping requirements to implemented code and the converse?
  - (Stahl, 2017) & (D. Stahl et al., 2016)

2.3.3 Eiffel Framework

- What is the proposed solution to address traceability issues in CI/CDE/CD environments?
  - (Stahl, 2017) & (D. Stahl et al., 2016)

2.4 Transition an Agile Environment

2.4.1 The Effect of Organizational Change to Agile

- What are the problems that businesses face in attempt to switch to Agile practices?
  - (Bosch, 2014) & (Meyer, 2014)

2.4.2 Roles in Agile

- What are typical roles that each individual plays in an Agile environment?
  - (Bosch, 2014) & (Meyer, 2014)
- Why do these roles exist?
  - (Bosch, 2014) & (Meyer, 2014)

2.4.3 Paradigm Shift in Leadership

- How has leadership changed as a result of Agile?
  - (Bosch, 2014)

3 Conclusion

3.1 Rephrase Thesis Statement

3.2 Closing Statement

4 Bibliography

References

Atkinson, B., & Edwards, D. (2018). *Generic pipelines using docker: The devops guide to building reusable, platform agnostic ci/cd frameworks*. Apress. doi: 10.1007/978-1-4842-3655-0

Bosch, J. (2014). *Continuous software engineering*. Springer International Publishing. doi: 10.1007/978-3-319-11283-1

Continuous delivery, deployment & integration: 20 key differences. (2018). Retrieved from `https://stackify.com/continuous-delivery-vs-continuous-deployment-vs-continuous-integration/`

Meyer, B. (2014). *Agile! the good, the hype and the ugly*. Springer International Publishing. doi: 10.1007/978-3-319-05155-0

Shahin, M., Babar, M. A., & Zhu, L. (2017). Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices. *IEEE Access*, *5*, 3909–3943. doi: 10.1109/access.2017.2685629

Stahl. (2017). *Large scale continuous integration and delivery: Making great software better and faster*. University of Groningen.

Stahl, D., Hallen, K., & Bosch, J. (2016). Achieving traceability in large scale continuous integration and delivery deployment, usage and validation of the eiffel framework. *Empirical Software Engineering*, *22*(3), 967–995. doi: 10.1007/s10664-016-9457-1