Rapid Continuous Software Engineering – State of the Practice and Open Research Questions

Report on the 6th International Workshop on Rapid Continuous Software Engineering (RCoSE 2020)

Marco Konersmann[1], Brian Fitzgerald[2], Michael Goedicke[3], Helena Holmström Olsson[4], Jan Bosch[5], Stephan Krusche[6]

[1]University of Koblenz-Landau, Germany, konersmann@uni-koblenz.de
[2]Lero, University of Limerick, Ireland, brian.fitzgeralt@ul.ie
[3]University of Duisburg-Essen, Germany, michael.goedicke@paluno.uni-due.de
[4]Malmö University, Sweden, helena.holmstrom.olsson@mau.se
[5]Chalmers | University Gothenburg, Sweden, jan.bosch@chalmers.se
[6]Technische Universität München, Germany, krusche@in.tum.de

ABSTRACT

We need to built software rapidly and with a high quality. These goals seem to be contradictory, but actually, implementing automation in build and deployment procedures as well as quality analysis can improve both the development pace and the resulting quality at the same time. Rapid Continuous Software Engineering describes novel software engineering approaches that focus on short release cycles, continuous deployment, delivery, and continuous improvement through rapid tool-assisted feedback to developers. To realize these approaches there is a need for research and innovation with respect to automation and tooling, and furthermore for research into the organizational changes that support high pace development. This paper reports on the results of the 6th International Workshop on Rapid Continuous Software Engineering (RCoSE 2020), which focuses on the challenges and potential solutions in the area of Rapid Continuous Software Engineering, before reporting on our discussions regarding the state of the practice and open research topics.

Keywords

rapid software engineering, continuous software engineering

1. INTRODUCTION

Systems we build are ultimately evaluated based on the value they deliver to their users and stakeholders. To increase the value, systems are subject to fast-paced evolution of the systems, due to unpredictable markets, complex and changing customer requirements, pressures of shorter time-to-market, and rapidly advancing information technologies. To address this situation, agile practices advocate flexibility, efficiency and speed. **Rapid continuous software engineering** refers to the organizational capability to develop, release and learn from software in **rapid parallel cycles**, typically hours, days or very small numbers of weeks. This includes to determine new functionality to build, evolving and refactoring the architecture, developing the functionality, validating it, and releasing it to customers, and collecting experimental feedback from the customers to inform the next cycle of development. One needs to relate the changes performed on the system with their effect on the metrics of interest, keep the changes with positive effects, and discard the rest. This requires not only agile processes in teams but in the complete research and development organization. Additionally, the technology used in the different development phases, like requirements engineering and system integration, must support the quick development cycles.

The capability to perform all these activities in days or a few weeks requires significant changes in the entire software engineering approach, including parallelising activities, empowering cross functional teams to allow for rapid decision making and light weight coordination across teams. It also requires significant technical advances in the engineering infrastructure, including continuous integration and deployment, collection of post-deployment product usage data, support for running automatic live experiments to evaluate different system alternatives, e.g., A/B testing.

Reaching this goal requires crosscutting research which spans from the area of **process and organizational aspects** in software engineering to **technical aspects** in the individual phases of the software engineering life cycle. Rapidly developing and evolving software systems is important in control-flow oriented as well as data-centric systems, from internet services to cyber-physical systems, and many more. Still, the processes and technology need to respect the differences between these types of systems.

2. RCoSE WORKSHOP

The workshop RCoSE 2020 took place online on July 1st, 2020 using a video conferencing software due to the Corona pandemic. It was part of the similarly virtual ICSE 2020 conference. The workshop had 34 registered participants. Videos of the talks were provided online before the workshop alongside the paper contributions. In the workshop the authors briefly summarized their contribution to create a common base before we started an intense discussion phase. This format proved to be valuable as there was plenty of time for discussion and the other participants could follow the presentations when it fitted well. After a keynote we discussed the workshop papers. The workshop had a total of

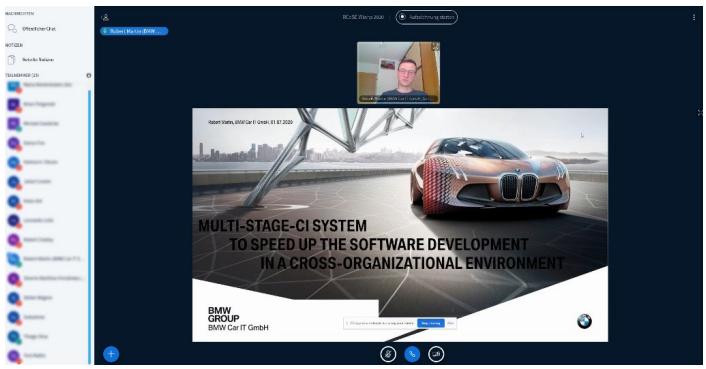

Figure 1: Snapshot from the keynote at the virtual RCoSE @ ICSE 2020

6 submissions of which 3 were accepted. The workshop had an extensive discussion session in which we discussed the state of the art of rapid and continuous software engineering in different application domains and open research topics.

2.1 Keynote

The workshop started with a keynote from Robert Martin of the BMW Group in Germany, who presented a multi-stage CI system to speed up the software development in a cross-organizational environment (see Figure 1). He discussed different formats for continuous deployment with multiple organizations in multiple stages and heterogeneous development environments. The goal of the system is to reduce the software delivery hang time, which is the time for a commit to get to the product. The CI system is highly automated to test as much as possible as fast as possible. A key factor is to block errors as early as possible to save time and resources. Challenges include the testing in hardware. Also, while ideally a car would run automatically for hundreds of kilometers for every commit, this is obviously too expensive. Still, continuous integration at this point opens doors for continuous improvement.

Introducing a Multi-Stage-CI system technically can be expensive, based on the cost of the technology to be used, e.g., when using expensive compilers, that produce costs per build. The organizational and behavioral changes of the stakeholders highly depend on the organizational culture: there can be a snowball effect when stakeholders can see the benefits. Certification is an issue, and that is not integrated into the pipeline as of today. Certification bodies and producers should think about continuous certification. While research regarding this point already exists, this has yet to find its place in practice.

2.2 Workshop Contributions

The industry abstract "Automating Continuous Planning in SAFe" [1] of Darius Foo, Jonah Dela Cruz, Subashree Sekar, and Asankhaya Sharma is motivated by quarterly face-to-face PI planning sessions, that are expensive and difficult to manage. The authors presented how they replaced the meetings with continuous planning, i.e., the plan is always refined and always up-to-date. They introduced Sapling, a novel tool for collaboratively managing and visualizing continuous planning, that integrates with Jira. The tool can be used to continuously optimize the work plan of multiple teams regarding hard and soft constraints. The research prototype has been evaluated with 3 teams concurrently using the tool.

The industry abstract "Challenges and Benefits from Using Software Analytics in Softeam" [2] of Alessandra Bagnato, Antonin Abhervé, Silverio Martínez-Fernández, and Xavier Franch describe the use of Modelio at Softeam. Modelio is a case study for Q-Rapids. It collects data of software engineering tools like Jenkins, Mantis, Sonarqube, etc. for data-analytics with metrics and strategic indicators. By integrating their processes with the tool Softeam now can do real time updates of a quality model. The tool centralizes metrics and strategic indicators in one platform and automates the quality management process, by automatically triggering alarms and automatically entering entries into a backlog. It also allows for simulation of strategic changes and monitoring of the quality requirement resolution process. The integration into further organizational processes is still an issue to be resolved.

In the research paper "Platform Teams: The Leading Edge Organizational Structure for Continuous Delivery" [3] of Leonardo Alexandre Ferreira Leite, Gustavo Pinto, Fabio Kon, and Paulo Meirelles use the Grounded Theory approach to build a taxonomy of organizational structures for continuous SE and evaluate existing structures for their effectiveness using interviews. They presents platform teams as well-suited organizational structure and compare them to collaborating siloed departments, cross-functional teams, and devops teams regarding their performance. Platform teams are product teams that have the platform as a product and the developers of other product teams as internal customers. As a key characteristic of platform teams the authors identify, that the platform team is an infrastructure team, that provides highly-automated infrastructure services to empower product teams.

3. STATE OF THE PRACTICE

Since the advent of the RCoSE workshop in 2014 things have changed. We noticed that early birds of RCoSE often overstated the frequency of changes. More realistically, the very frequent changes are very small changes. More significant changes are less frequent. It might be a good idea to differentiate between these. Also, while RCoSE is beneficial in practice, as has been shown in the State of DevOps reports [4], and many organizations benefit from it once implemented, many organizations do not benefit because they don't jump the bar to implement such a process for their core systems. Currently, no common, tailorable process exists to show the benefits of RCoSE to individual organizations.

4. OPEN RESEARCH TOPICS

We discussed we discussed **the role of traceability in RCoSE** as important open research topic The participants discussed traceability as means to trace failures to root causes and tracing releases to deployments. Tracing business goals to production is important to answer questions like "How much value does my feature generate?". In the latter it was noted that A/B testing seems to be a natural choice for answering such questions, but while this works well with, e.g., UI changes, it is not desirable for security design. Traceability as a legal requirement is commonplace. The participants discussed the potential and limitations for automating traceability for legal purposes. Discussing the challenges for traceability in RCoSE, we identified that the robustness of tools for traceability is often shaky today and that a holistic traceability would certainly be good, but maintaining traces does not scale well. We further discussed continuous certification and traceability for data-driven Software Engineering. Hard questions to answer are: What should be traced, how much tracing do we need and how much tracing is too much.

5. CONCLUSIONS

As a summary, we can state that RCoSE is beneficial in SE in practice. This is often shown in the domain of web-based services, but it can also be beneficial in other domains like embedded systems.

A main take-away of the workshop is that the key factors for benefiting from rapid and continuous software engineering seem to be technology to enable RCoSE and team topologies to operate them. As a major open research topic we identified traceability, as it is important for RCoSE in multiple aspects, but hard questions remain unanswered until now.

6. ACKNOWLEDGMENTS

We would like to thank Robert Martin for his inspiring keynote. We would like to thank all participants for their contributions in the forms of papers, talks and discussions in the breakout groups. We would like to thank the organization team of the ICSE for providing the frame for our workshop, the ICSE 2020 Workshops Chairs Miryung Kim and Lori Pollock for helping us hosting our workshop virtually in the challenging times of the Corona pandemic. Finally, we want to thank the RCoSE Program Committee comprised of Jan Bosch, Brian Fitzgerald, Wolfgang Gehringng, Michael Goedicke, Jan Ole Johanßen, Marco Konersmann, Stephan Krusche, Casper Lassenius, Jürgen Münch, Helena Holmström Olsson, Karen Smiley, Klaas-Jan Stol, Matthias Tichy, and Stefan Wagner.

This research was supported by the Science Foundation Ireland grant 13/RC/2094.

7. REFERENCES

[1] Darius Foo, Jonah Dela Cruz, Subashree Sekar, and Asankhaya Sharma. 2020. Automating Continuous Planning in SAFe. In Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW'20). Association for Computing Machinery, New York, NY, USA, 504. DOI:10.1145/3387940.3391536

[2] Alessandra Bagnato, Antonin Abhervé, Silverio Martínez-Fernández, and Xavier Franch. 2020. Challenges and Benefits from Using Software Analytics in Softeam. In Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW'20). Association for Computing Machinery, New York, NY, USA, 512. DOI:10.1145/3387940.3391537

[3] Leonardo Leite, Fabio Kon, Gustavo Pinto, and Paulo Meirelles. 2020. Platform Teams: An Organizational Structure for Continuous Delivery. In Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW'20). Association for Computing Machinery, New York, NY, USA, 505–511. DOI:10.1145/3387940.3391455

[4] N. Forsgren, D. Smith, J. Humble, and J. Frazelle, "2019 AccelerateState of DevOps Report," Google, Tech. Rep., 2019. [Online]. Available: https://services.google.com/fh/files/misc/state-of-devops-2019.pdf