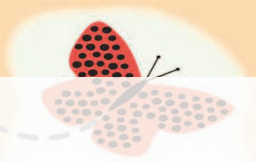


Release Engineering 3.0

Bram Adams, Polytechnique Montreal

Stephany Bellomo, Software Engineering Institute

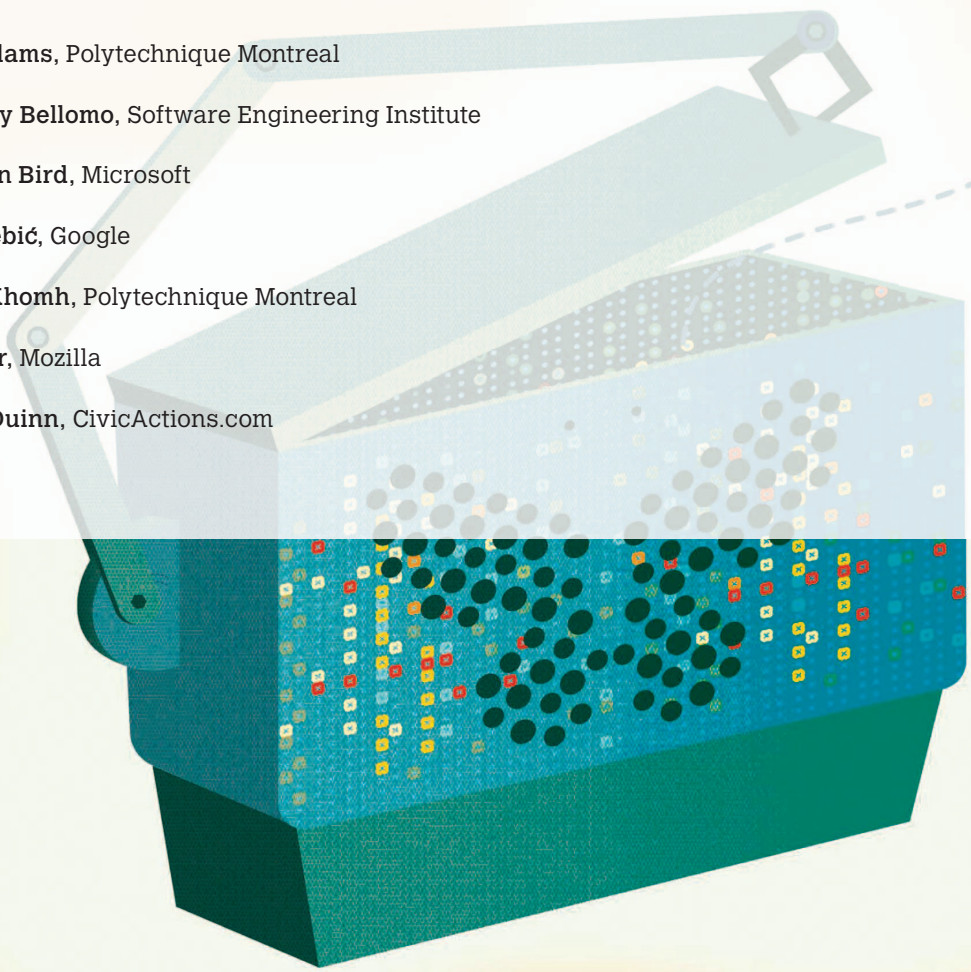
Christian Bird, Microsoft

Boris Debić, Google

Foutse Khomh, Polytechnique Montreal

Kim Moir, Mozilla

John O'Duinn, CivicActions.com



0740-7459/18/$33.00 © 2018 IEEE

WE'RE HAPPY TO introduce you to the Release Engineering 3.0 theme issue of *IEEE Software*. It builds on the successful RELENG workshop series (releng.polymtl.ca) and the March/April 2015 *IEEE Software* issue on Release Engineering. So, what's all this fuzz about release engineering, and what about the "3.0" moniker?

Release Engineering 3.0

Release engineering is the discipline of code integration, build, test execution, deployment, and delivery of high-quality software releases to users. First, the code contributions developed independently by an organization's developers must be integrated into a coherent whole. Then, the source code, libraries, and any other resources of the integrated components must be transformed ("built") into a working set of executables to enable testing and (if all is well) deployment into the production environment, such as a cloud, an app store, or some download server. Finally, the deployed product can be released at the right time to the right set of users. In other words, release engineering forms the vital link between software's design and development phases and the finished product's use and maintenance.

Releasing complex software systems on time with the right quality requires considerable process and technical changes.[1] This is especially true when you aim to perform continuous delivery (or even deployment), in which a software product should be shippable (or shipped) after each valid code change, automatically.

For example, continuous-integration builds and tests must be scaled up for increasing volumes of code changes, infrastructure requirements such as virtual machines or library dependencies must be codified using infrastructure-as-code programming languages, and operators should be able to roll back to a safe earlier release at the click of a button. Furthermore, the release-engineering team needs to provide a feedback loop to the development team to flag architectural or design issues that inhibit achieving rapid, robust deployment. For example, for canary deployment or release, two versions of the system must be live at the same moment, which requires compatibility of APIs and database schemas.

Even though the practices of continuous delivery and deployment have become part of developers' vocabulary over the last eight years, the scope and nature of the process and technical changes we mentioned before have been costly to discover, often by trial and error. Even at this point, large software companies such as Google, Facebook, and Netflix, who were at the forefront of modern release-engineering practices, have a long list of open challenges and issues—for example, regarding the long-term viability and deficiencies of practices such as A/B testing or dark launching, or regarding dependency management in their ecosystems of autonomously updated microservices. Furthermore, new deployment platforms such as mobile-app stores still provide major challenges in terms of managing releases without having full control.

This brings us to the "3.0" moniker. Release Engineering 1.0 and 2.0 refer, respectively, to traditional ad hoc release engineering and today's highly automated release engineering for general-purpose cloud and mobile systems. In contrast, Release Engineering 3.0 targets the future iteration of release-engineering processes aimed at supporting small companies, start-ups, civic organizations, government administrations, and safety-critical industries. For example, the software in cars, hospital equipment, or election software needs updates to deliver critical bug fixes and new functionality. However, without proper precautions, innocent lives could be at stake. Similarly, how should we go about continuously delivering software in new domains such as the Internet of Things or swarm robotics, without endangering people?

Ideally, small companies, start-ups, civic organizations, government administrations, and safety-critical industries (healthcare, the automotive industry, and so on) should be able to select and adopt a release-engineering process and tool chain that fit their needs. Yet, such an "out-of-the-box" process and tool chain are far from a reality, and so are textbooks or experience reports with empirically validated best practices for release engineering. Although many blogs and papers and some books discuss release engineering for large cloud applications and (to some extent) mobile apps, no thorough treatment exists of today's challenges and solutions for release engineering of the "other 80 percent" of software systems.

In This Issue

This theme issue aims to get the ball rolling on Release Engineering 3.0 and stimulate both industry practitioners and researchers to reflect on what such release-engineering practices and tools could look like and how they could evolve out of

ABOUT THE AUTHORS

BRAM ADAMS is an associate professor at Polytechnique Montreal, where he heads the Lab on Maintenance, Construction, and Intelligence of Software. His research interests include release engineering in general, as well as software integration, software build systems, and infrastructure as code. Adams obtained his PhD in computer science engineering from Ghent University. He is a steering-committee member of the International Workshop on Release Engineering (RELENG) and coedited the first *IEEE Software* issue on Release Engineering. Contact him at bram.adams@polymtl.ca.

STEPHANY BELLOMO is a member of the technical staff at Carnegie Mellon University's Software Engineering Institute. She focuses on empirical research for improving software delivery and working with US Department of Defense and government practitioners on software-related challenges. Bellomo received an MS in software engineering from George Mason University. She is a steering-committee member of RELENG and coedited the first *IEEE Software* issue on Release Engineering. Contact her at sbellomo@sei.cmu.edu.

CHRISTIAN BIRD is a researcher at Microsoft Research. He has studied many aspects of release engineering, most recently code movement and social dynamics in build teams. Bird received a PhD in computer science from the University of California, Davis. He is a steering-committee member of RELENG and coedited the first *IEEE Software* issue on Release Engineering. Contact him at cbird@microsoft.com.

BORIS DEBIĆ is a Google engineer. With support from NASA's Ames Research Center, he also directs the Mars Society's NorCal Rover project. His research interests are release engineering, machine learning (classification), and privacy. Debić received an MSc in physics from the University of Zagreb. He's a steering-committee member of RELENG. Contact him at releng@debic.net.

FOUTSE KHOMH is an associate professor at Polytechnique Montreal, where he leads the Software Analytics and Technology Lab. His research interests are software maintenance and evolution, cloud engineering, service-centric software engineering, empirical software engineering, and software analytics. Khomh received a PhD in software engineering from the University of Montreal. He is a steering-committee member of RELENG and co-edited the first *IEEE Software* issue on Release Engineering. Contact him at foutse.khomh@polymtl.ca.

KIM MOIR is a staff release engineer at Mozilla. Her interests lie in build optimization, scaling large infrastructure, and writing about the complexities of open source release engineering. Kim received a bachelor of business administration from Acadia University. She is a steering-committee member of RELENG and co-edited the first *IEEE Software* issue on Release Engineering. Contact her at kmoir@mozilla.com; kimmoir.blog.

JOHN O'DUINN is a senior strategist at CivicActions.com and an advisor and a mentor to geodistributed organizations. His research interests include how release engineering reframes the business of software development while at the same time improving the lives of developers and the wider public depending on these systems. O'Duinn received an MSc in computer science from Dublin City University. He's a steering-committee member of RELENG. Contact him at john@oduinn.com.

more-advanced forms of Release Engineering 2.0. With these goals in mind, after at least three experts from both academia and industry rigorously reviewed each submission, *IEEE Software* accepted the following four.

In "Continuous Experimentation: Challenges, Implementation Techniques, and Current Research," Gerald Schermann and his colleagues review the state of the art of, and issues with, using the release-engineering pipeline for quick, data-driven experiments of new product features or tweaks. Such experiments target user subsets with a new release for a limited time, to obtain actual usage feedback (in production) that can help with decisions about future releases. Continuous experiments will likely be an essential component of Release Engineering 3.0.

In "Correct, Efficient, and Tailored: The Future of Build Systems," Guillaume Maudoux and Kim Mens examine build systems, a critical ingredient of release-engineering pipelines. They survey various improvements made by different build system technologies and identify features and optimization mechanisms build systems could implement to make themselves suitable for Release Engineering 3.0, thereby improving the new pipeline's efficiency and correctness.

In "Continuous Delivery: Building Trust in a Large-Scale, Complex Government Organization," Rodrigo Siqueira and his colleagues discuss organizational and technical challenges related to the introduction of continuous-delivery practices in government administrations, which typically aren't accustomed to release-engineering practices. The authors report their experience working with a government institution used to waterfall-like processes, and provide advice on how to implement continuous delivery. This article is one example of the application of Release Engineering 3.0 in a civic setting.

Finally, in "Over-the-Air Updates for Robotic Swarms," Vivek Varadharajan and his colleagues consider another new domain for Release Engineering 3.0: robotic swarms. Such swarms are essential for supporting humanitarian missions in rough circumstances, where robots or drones must autonomously explore and interact with a disaster scene without global network connectivity, under time pressure. The authors outline the challenges of performing over-the-air software updates in this context and present an initial gossip-based approach that spreads updates in peer-to-peer fashion.

As is clear from the previous summaries, each article highlights a different facet of what Release Engineering 3.0 could look like, with two articles focusing on fundamental technologies and practices and two articles discussing new application domains. Of course, many unexamined facets remain; thus, we see this special issue as an initial milestone toward defining and elaborating Release Engineering 3.0.

Finally, we thank all the people who did a great job writing or reviewing the high-quality submissions for this theme issue. We also thank Editor in Chief Diomidis Spinellis and his *IEEE Software* crew for their guidance and support.

Happy reading!

Reference

1. J. Humble and D. Farley, *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*, 1st ed., Addison-Wesley, 2010.