Continuous Delivery: The Ultimate Challenge for Software Development Managers

By Kevin Aguanno, PMP, MAPM, IPMA-B, Cert.APM, CSM, CSP
and Mike Bowler, CSM

In a recent article titled “Agile Methods and The Need for Speed” (http://pmac-agpc.ca/node/565), the author notes that many people are adopting agile development methods in hopes that they can deliver their project faster. The article goes on to state that, while speed is a possible outcome of agile methods, it is not guaranteed. Agile projects are more likely to achieve other benefits first, such as reduced technical risk, higher quality, greater likelihood of meeting true requirements, and more. Yet there are still those elusive speed-related benefits that people strive to achieve in their first agile projects.

To truly deliver software faster, one must look towards cutting down the timespan of all processes in the software development lifecycle from requirements gathering to deployment. Many who seek faster delivery use agile methods to improve the requirements gathering, design and development processes but are frustrated in their attempts to get a speedier deployment of the new software. These people often see deployment activities as unnecessarily cumbersome and often without much perceived value.

Perhaps the agile community has contributed to this perception. We have talked about concepts such as “continuous deployment” for years as if it were just one of the many agile techniques we can employ on our projects. Yet, this particular technique stands apart from many of the other basic agile techniques such as holding daily stand-up meetings, managing requirements using backlogs, and breaking a project down into iterations which culminate in a demonstration to stakeholders. Continuous deployment is among the most difficult of agile techniques to employ successfully and requires a very high level of agile maturity and discipline in the team.

Generally, to be successful at continuously deploying software into a production environment, a number of preconditions must be met:

- **Automated Builds** – The team must have tooling in place to automate their software builds. In addition, the team must follow consistent coding standards to help facilitate the automated builds. For continuous deployment, there is no time to waste manually integrating source code from various developers into the official build.

- **Automated Regression Tests** – Every time an automated build takes place, there should be software in place to execute a suite of automated regression tests to make sure that the existing deployed software is not impacted negatively by the new code. Some automated build tools can execute "smoke tests" or a partial suite of lightweight regression tests to verify basic functionality. For full regression testing passes, however, you will likely want to take advantage of a separate, dedicated automated testing solution.

- **Automated Compliance Scans** – In some environments, after the software development team determines that a high-quality package of software is ready for deployment to production, the software must still go through external compliance scans. One example of this is found in web applications that capture credit card information as part of a financial transaction. Credit card merchants are now required to have their applications go through a Payment Card Industry (PCI) compliance scan by an independent, external provider. If one of these external compliance scans is required for an application before it can go to production, then the initiation and execution of such scans also should be automated as much as possible. We know of one company with a PCI compliance scanning requirement who is integrating HP Fortify (security) checks into their ongoing builds so that the developers are notified as early as possible about possible security issues. It is not yet clear to the team whether a full HP Fortify scan can be done in less than three or four days so this can be a real bottleneck to rapid deployments. If more rapid deployments are required, there has to be a discussion with the external compliance verification agency of whether PCI compliance can be maintained with only a subset of a full scan.

- **"One-Click" Deployment** – While not achievable in many organizations due to regulatory or policy restrictions, in the ideal world, teams would have the ability to select a software build for deployment to production and with the click of a mouse promote the software. Executing typical governance functions such as user acceptance testing, independent systems integration testing, and scheduled deployment windows all slows down the release of software to production, often forcing days – or even weeks – into the timeline between code completion and live deployment. For truly continuous deployment, teams need to be able to bypass these quality assurance and management processes. Obviously, this requires very high-quality code to consistently come out of the development phase – something many organizations have difficulty achieving.

- **Automated Database Migrations** – Data schema changes may occur during an agile development project. Such changes complicate deploying code to production as database updates also need to be coordinated. In continuous deployment environments, these changes need to be automated. Automated database migration is a concept that became popular through Ruby On Rails several years ago and is now implemented in a wide range of languages such as SQL, Java, or C#. Migrations are typically written in pairs: one to modify the database forward and one to roll back the same change in case of mistakes. A typical project will have a database in several environments that all use the same schema (for example: Development, Staging, UAT, and Production) and automated migration ensures that they are all consistent and reproducible.

- **Automated Software Alerts and Monitoring** – If code and database changes are being deployed at the click of a button, one should monitor the solution closely once it has been deployed to a production environment. Code should be written with monitoring "hooks" for application monitoring tools such as IBM Tivoli Monitoring, HP Application Performance Management, or even Open Source tools such as the Nagios package. Software applications should generate alerts that these monitoring tools can access to identify errors at a granular level. In a perfect world, these monitoring tools would not be necessary as we should be generating perfect, high-quality code from our development phase; however, in the real world, these tools become a necessity. Because extensive (and time-consuming) independent testing has not occurred at the end of the development phase, teams attempting continuous deployment need a way to identify granular issues in real time and monitoring tools are an ideal solution to this problem.

- **Flags to Enable or Disable Features** – Finally, once we automatically deploy software with the click of a button, and our automated monitoring tools sense a problem, we need a way to at least disable the offending features and at most quickly revert code back to a previous version. Externalizing flags to enable or disable features is a simple programming technique that can allow monitoring tools (or administrators) to turn off troublesome areas quickly, keeping a production environment running while the problem is investigated and fixed in a development environment. This also allows us to deploy an incomplete feature into a production environment; with the flag for the incomplete feature turned off, no one will ever see the feature and the code will not execute. Having the ability to do this means that software can be promoted to production *at any time*, not just when features are complete. Similar to feature flags is the concept mentioned earlier of backing out database schema changes if they are found to cause problems. Development managers must take care in maintaining traceability between code changes and data schema changes so that related items can be turned off or rolled back in parallel to avoid causing further issues.

All of these prerequisites mean that continuous deployment is very difficult to achieve and to execute well. It is not just a fancy term for debugging code in the production environment – which, although unfortunately common, is clearly a very bad practice. Continuous deployment requires clearly-defined standards, very strict discipline among team members, a high level of skill, a maniacal focus on quality, and unwavering professionalism. In short, it is achievable for mature agile development teams but is certainly out of reach for those new to agile development practices.

Many companies have been able to achieve this gold standard of agile development:

- Etsy, an online marketplace for handmade items, deploys code to production an average of 50 times per day. In order for such a large number of daily production deployments to succeed, their code quality is forced to be exceptionally high.

- Facebook has institutionalized continuous code deployment with no explicit human governance checkpoints. They have incorporated extensive application monitoring, however, to identify when they have introduced an error. Facebook mandates that its developers create automatic rollbacks to address any errors. In some cases, the error-handling routines in the Facebook source code turn off offending features, creating a self-modifying code base.

- Finally, most impressive of all, Google has an amazing continuous build system that runs 75 million automated regression tests per day across their codebase. Google has taken automation to the extreme to manage their 5,500 source code check-ins per day.

As we have shown, faster releases to production are possible. What business sponsors must realize, however, is that to continuously deploy code requires a number of development environment, application design, production monitoring, coding practice prerequisites. While those may seem simple to achieve, the agile process maturity, high level of team discipline, and extreme focus on quality may be harder to achieve. Organizations should start with obtaining some of the agile benefits that are easier to achieve and leave continuous deployment – faster delivery – for later when they have achieved increased maturity.

About the Authors

Kevin Aguanno

With over 20 years of managing complex systems integration and software development projects, **Kevin Aguanno** is known in the industry for his innovative approaches to solving common project management problems. He focuses on three project management specialty areas: agile project management, troubled project recovery, and project methodology consulting. As a well-known keynote speaker, trainer, and executive coach in agile management methods, Aguanno has taught thousands of people how to better manage high-change projects by using techniques from Scrum, Extreme Programming, Feature-Driven Development, OpenUP and other agile methods. He is a frequent presenter at conferences and private corporate events where he delights audiences with practical advice peppered with fascinating stories from his own experiences in the trenches practicing agile project management. He has taught for several years at the University of Waterloo and the University of Toronto where he won the coveted SCS Excellence in Teaching Award, and is a regular guest lecturer in software engineering and project management classes at several other universities.

Kevin Aguanno holds a B.A. from the University of Western Ontario, and a Master's in Project Management from the School of Business and Public Management at George Washington University. He is a PMI-certified Project Management Professional (PMP), and his competency is certified by IBM as a Certified Executive Project Manager and by the International Project Management Association (IPMA) as a Senior Project Manager (IPMA Level B). He is also certified by the Scrum Alliance as both a Certified ScrumMaster (CSM) and Certified Scrum Professional (CSP). He is also certified by the Project Management Association of Canada as a Certified Agile Project Manager (Cert.APM). Aguanno is an active member of the Project Management Institute (U.S.A.) including the Information Systems SIG, the Association for Project Management (U.K.), the Project Management Association of Canada where he is a founding director and the current President, the Agile Alliance, and the Scrum Alliance.

Kevin is accredited by the International Project Management Association (founded in Switzerland) as a project management competency assessor, and he performs IPMA assessments for the ASAPM in the U.S.A. and the PMAC in Canada, for both of which the IPMA awarded him the honorary designation of First Assessor. He is the author of over twenty books, audiobooks, and DVDs in addition to a number of articles published in magazines and journals worldwide. Find out more at http://www.mmpubs.com/catalog/Aguanno--Kevin-i-7.html.

Mike Bowler

Mike Bowler is an experienced agile coach with strong emphasis on both process and technical practices. He works with teams writing software and helps them do that better. He became interested in agile in 2000 and founded XP Toronto that year to get like-minded people together to discuss better ways to ship software. In 2003, after completing the Certified Scrum Master (CSM) designation, he co-founded Scrum Toronto to focus on that methodology. More recently, in 2012, Bowler co-founded Agile Toronto East to better serve the Agile community through Durham Region. Many agile coaches seem to focus only on the process skills behind agile. In addition to that, Bowler is also very strong with the technical skills such as Test Driven Development (TDD), refactoring, pair programming, continuous delivery and others. He regularly coaches teams on the use of these skills using languages such as Java, Ruby and C#. Bowler has demonstrated strong leadership skills by running a development shop in the insurance industry, founding two companies and also organizing and running various user groups such as Agile Toronto East, XP Toronto, The Durham IT Association and Scouts Canada. He has a wide range of industry experience including life insurance, medical, automotive, banking, alarm systems and loyalty programs. Finally, Bowler is an experienced speaker, having given many highly rated presentations at international conferences and local user groups. More at www.GenXus.com/bowler.

Managing Agile Projects, edited by Kevin Aguanno, published in January 2012 by Multi-Media Publications; ISBN: 9781895186116; soft cover, 420 pages – Info about the book at http://www.mmpubs.com/catalog/managing-agile-projects-book-p-1.html.