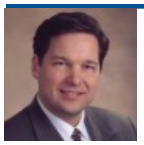
Precision meets Expertise.

Procept ASSOCIATES LTD.



Kevin Aguanno, BA, CSPM(IPMA-B), PMP, PMI-ACP, CSM, CSP, Cert.APM, MCPM, FPMAC, FAPM
Agile Practice Lead

Agile Methods and the Need for Speed

One of the most common reasons executives cite for wanting to adop agile management practices is the desire to deliver faster. The need for speed is understandable, given the fast pace of change in the business world; however, it is perhaps the most difficult agile benefit to achieve and requires high levels of agile maturity to realize. Instead, executives should look towards the other benefits of agile for justification, as they are much easier to achieve at lower levels of agile maturity. This article explores the benefits that can be realized immediately by those just starting their agile transformation and the preconditions to achieving the higher-maturity benefits such as faster delivery.

When asking people why they want to use agile delivery methods, one of the most common reasons we hear is that they want to "deliver faster." It seems that there is a widespread frustration with the way administrative bureaucracy, inefficient development processes, and overburdening governance processes impede project performance. In many cases, an apparently simple, short development project cannot be delivered quickly because of the process and governance overheads that stretch the project out across the calendar and act as a multiplier on the estimated project budget.

Of course project sponsors are frustrated with this situation – I'd be frustrated too. If there is needless red tape slowing down a project, that is an evil that should be rooted out and eradicated within our organizations. The problem, however, is that agile methods are not about delivering faster; rather, their benefits are in other areas:

- **Lower Risk of Building the Wrong Thing** — With frequent demonstrations of the evolving solution, the project sponsor and other stakeholders can see where the project is headed and they can redirect the project team's efforts if there has been a misunderstanding of requirements. Additionally, this redirection can include the addition of new requirements or changes to existing requirements to ensure the project is delivering optimum value. It

is important to note that the business stakeholders' understanding of what they asked for evolves over the course of the project, bringing new insights and new requests.

- **Rapid Reduction in Technical Risk** — Through careful prioritization, the project team can quickly eliminate technical risk in the project by validating the solution design in early iterations. If an assumption proves wrong, or if the solution design does not work, then there will be minimal rework required (and possibly plenty of time remaining on the schedule) to correct the issue. Once the major technical issues have been resolved, the remainder of the project should proceed without further major interruptions.
- **Higher Quality** — By testing throughout the project, defects are found early when there is time left in the project to correct them and relatively low level of rework required to correct the problems. This is very different from the waterfall method wherein testing is performed at the end of the whole project, leaving little time left for correcting issues and a higher likelihood that corrections will require significant updates across many areas of the completed solution. Add in the benefits of regression testing – re-testing past features/deliverables to ensure that recent changes or additions have not impacted them – and we can see that our most important items are tested and then retested over and over to ensure that they are of the highest quality at the end of the project.
- **Reduced Waste** — Agile methods put a very strong emphasis on continuous improvement activities with an aim to improve efficiency of the delivery process, allowing teams to reduce waste. At the end of each iteration, the delivery team holds a retrospective — a "lessons learned" meeting – where team members identify processes that are working effectively, those that did not work well, and what changes can be recommended for the next iteration to improve the teams' productivity and the quality of its deliverables.

Faster Delivery is in the top 5 most common reasons for adopting agile.

– "2014 State of the IT Union Survey," by Scott Ambler for *Dr. Dobbs Journal*.

- **Improved Trust** — Under the agile delivery model, a project team makes smaller, short-term commitments that are easier to estimate and to achieve. As a result, the team develops a reputation for meeting their commitments, which helps build trust with the business sponsor.
- **Open, Transparent Communications** — Agile methods recommend frequent, open communications between team members and also between the project delivery team and the project sponsor. Daily team meetings help team members understand each other's issues, fostering collaboration, mentoring, and early identification of issues to the project manager for resolution or escalation. Daily access to the project sponsor helps with escalation and rapid decision making. Agile status tracking techniques, such as requiring the delivery team to demonstrate completed deliverables at the end of each iteration, also help to improve transparency by revealing the true progress of the project — there should be no last-minute surprises for the sponsor of an agile project.
- **Improved Morale** — With a team able to meet its commitments by delivering completed solution components on a regular basis, the team members feel productive. Combined with the positive results of continuous improvement activities, team members know they are being efficient and see the value they are creating for the business. Shared team goals lead to cross-functional cooperation, which also adds to team productivity. All of these lead to good morale among team members.
- **Lower Risk of Being Late or Over Budget** — The above benefits combine to reduce the risk of the project completing late or over budget. Frequent feedback cycles ensure that misunderstandings are surfaced earlier when there may still be time to correct them without impacting the timeline. Continuous testing finds defects earlier so that they may be fixed earlier, reducing the risk of a major defect being found in the last days of the project. Continuous improvement activities

(retrospectives) ensure that the project focuses on delivering efficiently. Transparent communications help to reduce misunderstandings that may lead to rework or other project delays. Many of our agile practices contribute towards reducing the risk of the project exceeding schedule and budget constraints.

The previous items are the most common (and substantial) benefits that organizations achieve through the use of agile methods and none of them related to speed. There are a couple of other benefits, however, that do tie in to the need for faster delivery:

- **Earlier Delivery of Business Value** — By breaking the project down into segments that are delivered incrementally in regular intervals, the project creates the opportunity for the sponsor to make some use of a completed portion of the solution partway through the project. The sponsor may decide to use the partially-completed-but-still-functioning solution to beat competitors to the market and capture valuable market share, demonstrate to business partners what will be forthcoming when the project is complete so that they can begin aligning their own business offerings with the new solution, begin training users of the solution before the entire solution is complete, or any number of other beneficial scenarios. To capture early benefits in this way, the project sponsor needs to carefully prioritize project requirements and work with the project team to build a release plan that aligns with the business' strategy. Each release to a production state requires a significant investment in acceptance testing, independent quality assurance, data migration, training support staff, and other production-readiness activities. Business sponsors should carefully consider these activities and costs when determining how often they should promote work-in-progress into production, as too frequent promotions can negatively impact the business case.

- **Possibility of Completing the Project Early** — Something that rarely happens under the waterfall method but which becomes feasible using an agile approach is the early completion of the project, saving significant time and money. If a waterfall project is terminated three quarters of the way through, the project team is probably still building the solution or is just entering the testing phase. At this point, the solution is untested and likely has many defects preventing its use by the project sponsor to achieve the expected business value. With the delivery of the solution in fully-completed segments throughout the project, with the ordering of those segments in order of business priority, and with early and continuous testing, there is the opportunity with agile methods for the business to decide to end the project early, when the remaining work is of lower priority not being worth the extra investment to build those pieces.

These last two items, when implemented, can lead to faster delivery. Project sponsors should be very conscious, however, that the early promotion of a partial solution to production incurs significant extra costs for acceptance, production readiness verification, and other transition activities. The option to end the project early also comes with the cost of cutting out some of the lower-priority project scope.

In both of these cases, increased speed of delivery comes with a significant cost that needs to be carefully considered. Sponsors citing "the need for speed" as their primary reason for pushing for an agile approach may not understand the other benefits that may be easier to obtain and may more meaningfully impact their bottom line. Single-mindedly pushing for speed may jeopardize the attainment of the project business case without a sound understanding of the activities required to transition deliverables into a production state. Project sponsors should engage in broader discussions with the delivery team to understand the required transition activities and to prepare a delivery strategy that maximizes the delivery of business value considering all relevant costs. As always, increased communication is vital for project success.

Pre-Conditions for Faster Delvery

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:

1. 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.

2. 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.

3. 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.

Kevin Aguanno is the Agile Practice Lead for Procept Associates Ltd., one of PMI's first Registered Education Providers, specializing in training and project and programme strategy consulting. Author of over 30 books, audiobooks, and DVDs on project management topics, Aguanno teaches agile methods at several universities and at conferences around the world. As one of the pioneers in the agile movement, Aguanno invented the AgilePM methodology and was among the first to push more structured, disciplined approaches to agile. Currently, he spends most of his time helping large, complex organizations integrate agile project management methods into their governance frameworks. Find out more at **www.Procept.com/team/kevin-aguanno**