Published on *TechWell* (http://www.techwell.com)

Agile Code for Agile Teams

Article by Steve Berczuk [1] | Comments: (0) | Wed, 02/29/2012 - 15:26
Summary:

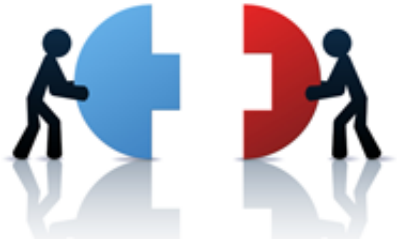
What makes a team agile? Is it in way it plans projects, or how it engineers its products? In this article, Steve Berczuk explains how agile code and technical practices can help a team stay agile across the product lifecycle.

TechWell Originals

What makes a team agile? Is it in way it plans projects, or how it engineers its products? In this article, Steve Berczuk explains how agile code and technical practices can help a team stay agile across the product lifecycle.

There are two closely related aspects to a team's being agile: 1) planning or project management, and 2) execution or engineering practices. When people think about agile projects, the focus is often either on the planning and project management process or the engineering practices. It's important to understand that both aspects work together. This article discusses how the way you execute your engineering practices can help your agile process be effective.

Agile Basics
Agile methods acknowledge uncertainty and manage it with techniques that rely on transparency, inspection, and adaptation, with an emphasis on working software. Agile projects allow teams to adjust goals in response to technical issues or changes in current needs or future technical or market forces.



Some agile methods, such as XP, focus on technical practices. Others, such as Scrum, focus on project management practices. Both planning and execution are necessary for an agile approach to work. Good planning makes it easier to execute in an agile way. Agile plans can only be effective if the team follows good engineering practices that give you the feedback that you need to inspect and evaluate and a codebase that will allow you to adapt.

To understand how engineering practices can drive improvements in the agility of your team, you need to understand the cultural challenges of being agile, and some basics of agile planning.

Cultural Challenges
Agile is about incremental progress and continuous improvement. To become a better agile team, you need to acknowledge uncertainty and the possibility of failure. In many teams, this is hard to do. Getting good feedback requires changes in how you define requirements, develop features, and interact with others in your organization.

Defining requirements with precise-enough definition to show whether you have met the goal, while also maintaining enough simplicity that you can move from specification to development quickly, can be challenging. Developing, maintaining working code, and coding and designing in a way that allows for incremental development and continual feedback require new approaches and skills. Change, learning, and accepting and giving feedback are often difficult.

Agile Planning
"Agile planning" means enough planning to move forward and measure progress. Agile requirements are focused on delivering functionality to users and start with three things:

- A user, who has a business need
- A feature, which is what the system will do
- A goal, which is reason the user wants to do the task

While many teams can develop lists of features, the user and end-goal parts of the story are often missing. The user and his goal are the most difficult parts to express but also the most important.

Having a clear statement of a goal allows you to decide what implementation will satisfy the core need and to evaluate whether the story is even necessary to implement. Since implementing features that do not have a clear use is wasteful, removing items from the backlog can be a major efficiency gain for a team.

Tracking and Defining Done
Tracking progress on a frequent basis is an important way to identify problems. Agile teams measure progress by continually re-evaluating the amount of work remaining, rather than effort spent or "percent complete."

Even when the product owner has a clear vision, teams often struggle with defining what needs to be built in a way that can be evaluated. Tests pass or fail, and builds are successful or not, but it's more difficult to determine if the test is testing the right thing. Deciding whether a team has completed a story may always have a subjective element, but you can make it easier if you have a good sense of what "done" means.

With an agile approach, some degree of consensus on how to evaluate completeness is critical. Without a good understanding of what a complete story is, the team cannot estimate accurately and, thus, cannot set expectations. Not being able to set expectations can cause a breakdown in trust between the team and the product owner. Without trust, it's harder for management to accept the idea of self-organizing teams, thus negating the efficiency benefits this approach provides.

Quite often, even the product owner is unclear about what to ask for. In these cases, it's best to

make decisions and acknowledge that you may be wrong rather than work with difficult-to-evaluate stories.

The developers on an agile project can contribute to project success in the face of uncertainty by working towards maintaining agile code.

Agile Code

Understanding the needs of the agile planning process, we can now talk about how engineering practices meet the needs of agile methods and drive them when they are lacking. Regardless of the agility of your product backlog, to be an agile team you need agile code. Agile code is code that you can change while still being able to deliver working software on a regular basis.

Agile code can be maintained by a combination of good design and having practices in place that provide constant feedback on the state of your code so that you can detect problems as soon as they occur. These practices include:

- *Automated unit and integration tests* in combination with *continuous integration*, to provide immediate feedback on the effects of a change to the code
- *Refactoring* and continually improving the structure of code while maintaining functionality
- *Frequent deployments* to a production-like environment to identify issue before they can cause a last-minute emergency and to make the application visible to stakeholders

By maintaining code in a working state and in a state where it is easier to change, you make it possible for the team to implement changes to a product backlog that a product owner might ask for.

While technical practices such as refactoring, design, and testing are essential to a successful agile project, avoid having purely technical tasks on the product backlog. Rather, consider how doing these tasks furthers the progress of the project. When working on backlog items, always consider effort required to refactor, design, and so forth as part of the estimate, since delivering code that can sustain change is essential to agile success.

Delivery and Deployment

Working software is the measure of progress in an agile project, but “working” means more than just “can demo” or even “compiles and passes automated tests.” Software isn’t useful and stakeholders cannot provide useful feedback on it until it can run on the target environment. Also, to be “done,” you need to address differences between a development system and a production-like one. There are two steps to building code in a way that supports an agile approach: developing in vertical slices and deploying early and often to a target environment.

The vertical slices approach is to develop end-to end features, from the user interface (UI) through whatever backend systems are involved. Rather than focusing on the data model or the UI or application tier, building end-to end (though less rich) solutions has advantages. Users can see the application do something. A “data model,” while important, is not easy to demonstrate. The team can validate the interactions between layers and make changes to make work at other layers easier, thus minimizing unnecessary rework. UI implementation can be influenced by decisions made at the data layer, for example, and vice versa. The team and product owners can understand what feature are truly necessary and have a better sense of what to defer if

something is late.

Make your application available on a target system early and verify the deployment and installation process often. This will give you an early opportunity to identify decisions that will simplify the deployment and configuration process.

The Agile Team

To be able to implement in vertical slices and be efficient, agile teams are often composed of generalizing specialists. Generalizing specialists can work on multiple aspects of the system, though they have expertise in a particular area. This means that all work that touches the UI is not blocked if your UI developer is overly busy. It also allows a first pass, end-to-end implementation by a single developer. As a generalizing specialist, you are not abandoning the idea that there are no “experts,” but you are encouraging team members to learn about and work with other aspects of the code. Having such a cross-functional team not only reduces bottlenecks in the development process but can also improve code quality by increasing the number of people who work with—and thus implicitly review—code.

Agile Code at the Center

To be successful at agile, you need to consider the entire product lifecycle, from planning to execution. You also need to be very aware of the challenges that the difference in approach will present to teams that have a different initial mindset. In many ways, implementing agile technical practices may present less resistance than planning practices, and, as long as your organization wants to be more agile, working on the technical practices can help identify the other bottlenecks to agile.

Content Images:

S_Berczuk_Agile_TW-AJ-CMC-SM_200x121.png [2]

Development & Deployment [3] agile [4] Agile Development [5] deployment [6] Project Planning [7]

Development & Deployment agile Agile Development deployment Project Planning

Source URL: http://www.techwell.com/articles/original/agile-code-agile-teams

Links:
[1] http://www.techwell.com/members/berczuk
[2] http://www.techwell.com/sites/default/files/article/images/S_Berczuk_Agile_TW-AJ-CMC-SM_200x121.png
[3] http://www.techwell.com/category/topics/development-deployment
[4] http://www.techwell.com/category/other-keywords/agile
[5] http://www.techwell.com/category/other-keywords/agile-development
[6] http://www.techwell.com/category/other-keywords/deployment
[7] http://www.techwell.com/category/other-keywords/project-planning