Notes on Agile 101: Execution

Steve Berczuk

Essential Agile

Agile teams do "enough" planning, and design, with a mindset that the goals will change, and that demonstrating working software is the best measure of progress. Evaluate agile practices in the context of the value and principles of the Agile Manifesto.

Agile methods, such as Scrum, involve the following elements:

- A Product Backlog which is a prioritized list of things that you want to build
- A Sprint (or iteration) backlog which is the items from Product Backlog that you are building "next."
- A development iteration (or sprint) where the team develops items from the backlog while meeting frequently to address roadblocks.
- A review to compare what the team did to the sprint backlog.

Agile execution and planning interact:

- Agile methods provide for frequent feedback on the state of the project, as reflected in the working software.
- Execution and planning go together. Good planning makes it easier to execute in an agile way.
- Your engineering practices can facilitate or of interfere with your goal of being agile.

Challenges

Getting good feedback has challenges in the areas of:

- Analysis: It's a challenge to define requirements in a way that includes a precise-enough definition of done that you can decide if you met the goal.
- Execution: maintaining working code, and coding and designing in a way that allows for incremental development requires new approaches and skills.
- Culture: Acknowledging incremental failure and changing direction is difficult in some organizations.

Agile Requirements

Agile requirements are focused on delivering functionality to users, and start with 3 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.

A template to follow is *as a <**user**> I want to <**do something**> so that <**goal**>.* These three elements allow you to validate the requirements and find the simplest implementation that meets the goal. Often people tend to focus on *what they want to* do and leave the *user* and *goal* unspecified.

Agile Code

To be an agile team, you need agile code. Agile code is code that can change while still allowing you to deliver working software on a regular basis. Engineering practices that help your code be more agile include:

- Unit Testing.
- Refactoring.
- Continuous Integration
- Frequent Deployments with a devops mindset, involving the operations team.
- Appropriate use of branching

Even as refactoring and setting up infrastructure are essential to be agile, resist the temptation to phrase backlog items in terms of them directly, but attach a business value. For example, if refactoring will make implementing a feature easier, include that in the estimate. Agile code is a means to delivering business value, not an end.

Agile Delivery and Architecture

One of the best ways to ensure that your team is on track is to deliver functionality in vertical slices through the architecture so that you can demonstrate functionality to the product owners at a review. It's often better to build simplified versions of a feature end to end rather than concentrate on either UI or a data model. Doing this offers the following 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 re-work. UI implementation can be influenced by decisions made at the data layer, for example, and vis-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.

Agile Teams

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.

To Learn More

Effective agile execution is a vast topic, and this overview only skims the surface of what you need to know. Here are some resources that may be useful:

1. The Agile Manifesto: http://www.agilemanifesto.org/
2. Agile Architecture:Coplein, J. (2010). *Lean Architecture: for Agile Software Development*, Wiley.
3. Agile SCM: Berczuk, S. P. and B. Appleton (2003). *Software Configuration Management Patterns : Effective Teamwork, Practical Integration*. Addison-Wesley.
4. Generalizing Specialists: Generalizing Specialists: Improving Your IT Career Skills by Scott Ambler. http://www.agilemodeling.com/essays/generalizingSpecialists.htm
5. Scrum: Schwaber, K. (2004). *Agile Project Management with Scrum*. Redmond, Wash., Microsoft Press.
6. XP: Beck, K. and C. Andres (2004). *extreme Programming Explained : Embrace Change*. Addison-Wesley.
7. User Stories: Cohn, M. (2004). *User Stories Applied*. Boston, MA, Addison-Wesley.

The following are good resources for learning about current ideas in agile software development:

1. Techwell: www.techwell.com
2. Agile New England: www.agilenewengland.com

Steve Berczuk is a software engineer at Humedica, author of the book *Software Configuration Management Patterns*, and an expert in agile SCM and a Certified Scrum Master with over 20 years experience helping teams work effectively. For more information visit Steve's website (www.berczuk.com), follow him on twitter(@sberczuk) or read his blog (http://steveberczuk.blogspot.com/) or find him on Linked In.