RedMonk
+1.866.RED.MONK
www.redmonk.com

What Does Agile Smell Like?

A RedMonk Briefing Document

Overview

This guide helps you sniff test how Agile an organization is. A "sniff test" is a quick way to establish a gut-feel about something. It helps you determine what to do next.

What is Agile?

While there is an Agile Manifesto, there is no single source that one can point to and say "that is Agile." Instead, Agile is a combination of product management, project management, and development practices.

Agile practices follow an iterative process for creating software, relying on rapid feedback loops to plan. The result is a method of developing software that values leaving options open until the right, business-advantageous time instead of locking down decisions early. Two of the more popular "Agile implementations" are Extreme Programming (or "XP") and Scrum; see the resources at the end for more pointers.

Agile Planning

Iterative Loops

In a sense, Agile development follows something akin to the water-fall cycle of software development, but tightens the loop to as short as a week or as long as 4-6 weeks. Compare this to the traditional time-span of of months or years. Each of these smaller loops is called an "iteration." The organization can develop incrementally, planning a small, but self-standing part of the project, developing that part, and then collectively asking "is this what we want?"

This brings us to the first, most important sniff test:

An Agile organization develops iteratively, verifying that the software is going in the right direction at the end of each iteration.

Flat Communication and Decision Paths

In Agile software development, all parties involved work on a level playing field. Different roles may be given the final say on a decision, but collaboration is expected from every role and every person. This brings us to the next sniff test:

An Agile organization has very flat and open communications paths.

If the organization is following a very command-and-control, hierarchical decision making process, something is smelly.

The Work Queue

The artifacts of Agile planning usually reduce to prioritized work queues containing stories which describe discrete, user-centric features. These light-weight mechanisms let an organization keep track of the work to be done and specify the desired results. The related sniff test is:

An Agile organization arranges it's work to be done in prioritized queues, allowing the ordering and content to change over time as required by the business needs.

The prioritized work queue, or "backlog" in Scrum, is one of the key planning artifacts. At any given time, a team can look at the backlog and know what to work on next, "pulling" their work rather than waiting for direction from management.

Each item in the work queue is a stand-alone feature. This means that management can choose to ship the software earlier rather than later if it's economically beneficial: it may not be feature complete, but the software will work. This brings us to the next sniff-test:

An Agile organization values putting off a decision for as long as possible, but, still makes many decisions.

Being able to put off a decision until all of the facts are in is incredibly valuable and often looked over by teams under pressure to deliver against set dates. Agile establishes the framework to make this possible and profitable.

Practices and Tools

Small Checkins

The Agile spirit of valuing short feedback loops reaches down to the day-to-day life of the developers:

Agile developers tend to check in smaller, but still operational, batches of code rather than larger, feature complete batches.

The team can then integrate the entire code base sooner rather than later, delivering working software at any given time, even though that software may not be as feature complete as desired. Supporting this practice is a very disciplined approach to quality management.

Testing

At any given time the code-base should contain as few bugs as possible. Bugs slow down development, making changes more difficult and costly. To avoid this:

An Agile organization runs a large amount of tests on a daily, if not hourly basis.

Unit tests help enable the above. A unit test is a small piece of code that can be easily and quickly run to verify that changes to a given piece of code have not created new bugs. While any given unit test may exercise a small part of the over-all project, successfully running all of the unit tests together verifies the that the project is in "a good state."

If a developer accidentally introduced a bug, after running the unit tests, they find the bug immediately and fix the bug before committing their code. This practice avoids exposing everyone else to that bug, which slows down the entire project.

Continuous Builds

Agile teams often run automatic builds, or, "continuous builds." A continuous build compiles and packages the software, then runs the automated tests, reporting any failures. If the build is broken, the team immediately fixes it. Thus:

An Agile organization builds and tests it's software at least daily, fixing broken builds rapidly.

A build that is constantly broken is a sign of a less than Agile organization. Realistically, very few organizations will have a 100% working build 100% of the time. What matters is how responsive the teams is to fixing the build.

Collaboration

Agile teams are very collaborative, sometimes seemingly painfully so. Team members in all roles should have a high degree of interaction with each other:

An Agile organization values frequent communication over comprehensive documentation.

This communication is not always spoken: IM, email, wikis, blogs, and other "quiet mediums" are used as much, if not more, than face-to-face communication.

Applying the Sniff Tests

These sniff tests cannot be used to "rate" an organization or team. But, they can give you an intuitive feel for how Agile an organization is. The end result should help guide your decisions about the organization.

If you're still at a loss, RedMonk recommends two courses of action:

- Ask the team members what they think is working and what's not working. If something is working, keep doing and amplify it. If it's not working, stop doing it and try something new.
- Bring in an expert to evaluate your organization and advice you. One organization's method of *becoming* Agile is rarely re-usable

across organization. Becoming Agile often involves changing the nature of an organization and outside help is often the best catalyst for change. Beware of "permanent consultants" however: a genuine Agile consultant's final goal it to make themselves unneeded.

Trust and Respect Your People

The last sniff test is establishing how much you trust each person in your organization. Agile is a software development process that relies on each individual being empowered and, more importantly, trained and informed to make the right decisions day-to-day, thus:

An Agile organization operates on trust, respect, and delegation instead of command and control.

Far from being vaporous, feel good claims, Agile is not just snappy mottoes and slogans. Rather, Agile is made up of tactics and practices that create and maintain organization that can deliver working code on schedule.

More Resources

The following are additional resources and links for more background. The current state of Agile is best documented in books, though weblogs, mailing lists, and wikis are quickly taking over:

- http://www.agilemanifesto.org/ - the Agile Manifesto whose 4 principals guide Agile-think.
- *Agile Software Development*, Alastair Cockburn - a great overview of Agile thinking along with actionable advice.
- *Agile Software Development with SCRUM*, Ken Schwaber, Mike Beedle - the Scrum manual.
- *Extreme Programming Explained*, Kent Beck and Cynthia Andres -the XP book.
- *Lean Software Development*, Mary Poppendieck and Tom Poppendieck - applying lean thinking to software development.
- http://groups.yahoo.com/group/scrumdevelopment/ - home to much practitioner discussion on Agile and Scrum.
- http://www.redmonk.com/cote/archives/agile/ - further discussion on Agile from RedMonk's Michael Coté.

About the Creative Commons License

About RedMonk

RedMonk is a research and advisory services firm that assists enterprises, vendors, systems integrators and corporate finance analysts in the decision making process around today's enterprise software stacks. We cover the industry by looking at integrated software stacks, focusing on business and operational context rather than speeds and feeds and feature tick-lists.

Founded by James Governor and Stephen O'Grady, and headquartered in Denver, Colorado, RedMonk is on the web at www.redmonk.com. If you would like to discuss this report email Michael Coté, cote@redmonk.com.