Agile Practices in Embedded Systems

Charles E. Matthews
Fifth Generation Systems, Ltd.
Markham, Ontario, Canada
charles.matthews@acm.org

Abstract

In the software industry, the Agile movement has produced much discussion since its inception at the 2001 meeting in Snowbird, Utah. Although Agile practices have been applied quite widely in various software development environments, the field of embedded systems has seen relatively few Agile projects. Because interest in Agile is growing in the embedded world, this paper reviews Agile practices from the perspective of embedded systems development. Some characteristics of embedded system development may influence the applicability of Agile practices to this subfield.

Categories and Subject Descriptors D.2.9 [**Software Engineering**]: Management – Software process models (e.g.e, CMM, ISO, PSP).

General Terms Management.

Keywords Agile methodologies; Embedded systems.

The Agile Manifesto[1] uses four guiding principles to determine which practices are to be followed in Agile projects. These principles illustrate philosophical preferences that Agile projects should follow. In one's work, as in one's life, the choices that you make while performing daily work tasks often reflect decisions of the importance of one value over others. The four guiding principles of the Agile Manifesto state that the following preferences should guide Agile projects.

- Value individuals and interactions over processes and tools
- Value working software over comprehensive documentation
- Value customer collaboration over contract negotiation
- Value responding to change over following a plan

Much of the literature on Agile uses practices from eXtreme Programming (XP) to teach this methodology. The primary practices of the XP methodology[2] are:

- Sit together: The team should all sit together - preferably in a single big room with no barriers between each other.
- Whole team: The team should include members from every stakeholder in the project - including customers, management, and testing.
- Informative workspace: The team location should reflect an informative workspace.
- Pair programming: All programming should be performed as pairs of programmers.
- Energized work: The typical explanation for this practice is to encourage 40 hour weeks over 50-60 hour weeks.
- Stories: The work tasks should revolve around stories of individual features that can be completed in 4-8 hour time frame.
- Weekly iterations: The fundamental release schedule should plan for weekly iterations of a releasable product.
- Quarterly cycle: High level planning for the project should be done on a quarterly cycle to allow for inevitable change.
- Slack: Slack time that occurs during the weekly iterations should be used to improve the existing code base rather than to start programming a new feature.
- Ten-minute build: The time that is necessary to perform a complete product build and release should be on the order to ten minutes.
- Continuous integration: Developers should continuously integrate their source code changes so that the entire product is buildable at any time.
- Test-first Programming: Developers should follow a test-first programming process.
- Incremental design: The team should perform incremental design during the development cycle rather than expend effort creating a detailed, up-front design.

Regardless of whether any specific practice from the XP list is useful or not, the author matched this list against the last 7 embedded projects with which he was associated. The following results identify the number of projects that used a specific practice.

- Sit together: 7-no; all projects involved geographically dispersed personnel
- Whole team: 7-no; most projects involved contractual requirements and were non-negotiable after the project began

SPLASH'11 Workshops, October 22–23, 2011, Portland, Oregon, USA.

- Informative workspace: 7-no
- Energized work: 6-yes; 1-no
- Pair programming: 7-no
- Stories: 7-no
- Weekly iterations: 7-no
- Quarterly cycle: 4-yes; 3-no
- Slack: 7-no; any slack time that occurred was used by selecting the next feature for development rather than spent improving the existing code base
- Ten-minute build: 7-yes
- Continuous integration: 5-yes; 2-no
- Test-first programming: 7-no
- Incremental design: 3-yes; 4-no

Because programming is a complex and technical activity, subconsciously one expects that the literature that proposes a programming methodology (such as, the Agile and XP literature) would follow technical standards for scientific discourse. Arguments for and against the proposal should be given in a neutral manner so that one can apply logical reasoning to reach a defendable conclusion. Unfortunately, much of the literature for Agile and XP programming is written in a style that more closely resembles propaganda than it does scientific discourse.

For example, consider that some of the XP practices are "motherhood" statements that every methodology would strive to achieve instead of being a practice that distinguishes XP from other methodologies. The practice of energized work states that people should work 40 hour weeks instead of scheduled overtime. Seriously, does any methodology advocate working 12 hour days and 7 days a week for extended durations? Wanting a balanced work schedule is not specific to XP or to Agile. Likewise, the practice of including team members from all stakeholders is a good practice that is recognized by all methodologies. Whether or not it is actually followed is usually an executive management decision that the development team has no influence over.

One could argue that at least one XP practice actually violates the first principle of the Agile Manifesto - value individuals over processes. Whether or not pair programming produces good code, it appears to enforce an unnatural process upon the individuals. Such an act violates the first value. Pair programming implies that all programmers conceive their programs while sitting at the keyboard. This is not always the case. Forcing them to do so is unnatural.

The practice of all team members sitting together in a large room may sound beneficial, at first. However, there has been a substantial amount of research on the influence of noise and distractions on a person's ability to perform technical activity. A highly quoted study[3] from The Institute of Psychiatry at King's College in London showed that multitasking and distractions can lower an individual's IQ by 10 points when performing complex tasks. Furthermore, a study[4] that IBM commissioned regarding how to design office space for programmers stated clearly that programmers should have private offices in order to allow them to focus more intently on their tasks. This study became the primary architectural factor when IBM built its Santa Teresa Lab in California.

In the author's opinion, some practices in embedded systems software development may be influenced by the field's closer ties to hardware development than other software areas. As a result, I expect that many embedded systems programmers will be resistant to incremental design. Hardware designers tend to put significant thought into designing for redundancy and thinking through the entire design than software designers. The often given reasoning is that a hardware change is much more expensive to make than a software change. Therefore, one must think through the hardware design much more thoroughly than the software design. Software mistakes cost less to fix once they are discovered.

Unlike some of the Agile practices whose impacts are difficult to measure quantitatively, the practices of continuous integration and test-first programming appear to be practices that can make measurable improvements in the quality and development time for software projects. Therefore, it is likely that they will be readily adopted in the embedded systems field. Whether other Agile practices become widely adopted remains to be seen. Those practices that cannot be quantitatively measured will be emotionally debated. Agile is coming to the embedded world. It remains to be seen which of its practices will be adopted.

References

[1] http://agilemanifesto.org

[2] Shore, J. and Warden, S. *The Art of Agile Development*, O'Reilly Media, Inc., 2007.

[3] http://www.drglennwilson.com/Infomania_experiment_for_HP.doc

[4] McCue, G.M. IBM's Santa Teresa Laboratory - Architectural Design for Program Development. *IBM Systems Journal 17,1* (1978), 4-25.