**Spring**

2012

ARINC 838 Agile Experiment

Team Information Overload V2

Mike Deats, Scott Griffin, Ryan Neal, Brandon Sutherlin, and Liron Yahdav

08

**Fall**

Table of Contents

ARINC 838 Agile Experiment 3

The Project 3

The Team 3

Setup 4

Requirements 4

What Worked Well 4

Challenges 4

Takeaway 5

Development, Coding, Testing and Continuous Integration 6

What Worked Well 6

Challenges 6

Takeaway 6

Conclusion 6

Works Cited 8

# ARINC 838 Agile Experiment

He who rejects change is the architect of decay. The only human institution which rejects progress, is the cemetery. ~Harold Wilson

Agile software development techniques are not considered ideal for mission-critical software (Boehm and Turner p27). However, the ultimate goal of Agile Development, according to the Agile Manifesto, is to “uncover better ways of developing software.” It is in this spirit of process improvement that this document seeks to identify if Agile techniques can add value to the ARINC 838 Loadable Software Part Specification (ARINC838) proof of concept.

# The Project

Carnegie Mellon University Silicon Valley (CMUSV) selected the ARINC838 proposal as a practicum project for the MS Software Engineering Technical Track (MSSE) for the Spring 2012 semester. The project was sponsored by the Boeing Company and has the following objectives:

* Mature the standards or the ARINC 838 Specification
* Identify shortcomings in the draft standard
* Simplify the loadable software installation process
* Demonstrate that relatively simple target loader code can reliably identify, verify and install loadable software to the LRU
* Evaluate whether this project, and subsequently other projects in the aerospace industry, can be completed using Agile methodologies

The project would also result in a reference implementation of ARINC 838. An actual implementation must adhere to the constraints of DO-178B Level D. The constraints required by DO-178B Level D are beyond the scope of this document.

# The Team

Team Information Overload V2 (IO2) was comprised of five members. In addition to their roles as part-time MSSE students, each participant has a full-time career as a professional software engineer. Four of the members have experience in defense, aerospace and contracts for the government. None of the members have experience in the particular domain covered by ARINC 838. The level of experience ranges from four to eight years. The participants have excellent team chemistry from previous academic projects and requested the opportunity to leverage that chemistry on a practicum project. All members of the team have been exposed to both Agile and traditional development modalities in both concept and practice.

# Setup

IO2 selected Extreme Programming (XP) to guide implementation of the project. The team expected to tailor XP as needed. IO2 anticipated being able to minimize pair programming for rote tasks. They employed the following tools for development:

|  |  |  |
| --- | --- | --- |
| **Tool** |  | **Agile Practice** |
| Pivotal Tracker |  | Velocity  User Stories  Iterative Development |
| Planning Poker |  | Story Estimation |
| Mikogo |  | Pair Programming |
| TestNG |  | TDD |
| TeamCity |  | Continuous Integration |
| GitHub & Pivotal Tracker |  | Informative Workspace |

The full description of the configuration of the development environment is available in the engineering notebook at the following URL:

<https://github.com/squirrely/arinc_838/wiki/_pages>

The rationale for the employing XP as well as the tools selected is available in the file [SPG2012\_Practicum\_TeamIO2\_RoadMap.docx](https://github.com/squirrely/arinc_838/blob/master/docs/Project/SPG2012_Practicum_TeamIO2_RoadMap.docx).

# Requirements

Agile development does not seek to define all the requirements as a precondition to coding. Hence, it is ideal for projects with dynamic or as yet unrefined requirements. One of the key objectives of the project was to mature the ARINC 838 standard, which among other characteristics makes it ideal for an Agile approach.

## What Worked Well

The customer was familiar with XP and provided a rough outline of stories to be implemented. This saved a great deal of time over more conventional requirement derivation techniques such as personal narratives and personas. IO2 was able to refine and estimate the stories and then work with the customer to prioritize the stories for iteration planning. The whole team focused on the work for the current iteration and providing immediate and demonstrable value. This allowed the team to begin developing earlier, which became a mechanism to better understand the problem and identify gaps in the requirements.

## Challenges

Many Agile practices are predicated on the concept of *Just Enough.* It can be extremely daunting to know how much is just enough. The issue is exacerbated if the team is inexperienced with the problem domain. This particular challenged surfaced in at least two instances.

At the end of the first release, a significant refactor became necessary. This activity was motivated in part by a lack of understanding by IO2 and ARINC 838 gaps identified by IO2. The refactor cost just over a week of schedule. In addition to schedule slippage the team suffered some loss of momentum.

At the end of the second release, and effectively the project, there was a dependency between the final stories. This dependency left the team with a few options. The team decided to have each pair work in their respective branch and then resolve the conflicts during the merge. It is not surprising that the merge was cumbersome.

## Takeaway

In both cases, additional planning could have prevented the schedule slippage specifically associated with the activity. One may be tempted to think in terms of the adage about an ounce of prevention, however the danger is that it is difficult to measure exactly an ounce, and where that ounce needs to be applied. The following questions need to be considered:

* *How much additional upfront planning, requirements analysis, and design would have been necessary to prevent the issues*.
* *Is the required time reasonable considering the scope and duration of the project*? *Is that less time than each refactor?*
* *Which approach yields the most value to the customer*?
* *What is the cost of change if the additional upfront effort failed to prevent the issues?*

For these particular incidences, IO2 believes that the Agile approach provides more value for the customer. Furthermore, IO2 feels that *Just Enough* increases in accuracy directly proportional to experience in both XP and ARINC 838.

A final consideration with regard to requirements is documentation. Agile values *working software over comprehensive documentation (Agile Manifesto)*. Ed Katz Ph. D of CMUSV warns that this notion is frequently misperceived to imply that Agile means no documentation. Katz clarifies that it means *Just Enough* documentation. The same questions apply to the ambiguity of *Just Enough* for documentation. However, in this case there are guidelines and criterion provided by DO-178B Level D. Hence, just enough documentation must include fulfilling these criteria. IO2 created a requirements traceability matrix and accompanying requirements document toward this end. No facet of XP prevents a team from generating documentation. In fact, IO2 found the effort necessary to create the traditional documentation required by DO-178B Level D to be trivial.

# Development, Coding, Testing and Continuous Integration

Pair programming is a core practice of XP. This practice is easily one of the most challenging concepts for management. It is often misconstrued as two engineers doing the work of one. If the technique is employed optimally where one partner writes the test and the other partner writes the code to fix the test, then the project will have distributed knowledge, automated test suites with a high-level of test coverage and real time code review. Moreover, like test-driven development and continuous integration, there is not anything prohibiting traditional development from implementing these tenets of XP.

## What Worked Well

It is difficult to overstate the value of test-driven development. It guarantees that all code will be tested, provides real-time and immediate regression tests, and helps to shape *Just Enough* design. Moreover, the resulting test suite rolls up nicely into integration tests.

IO2 rotated pairing partners. Every team member enjoyed opportunities to work together. The team leveraged its tacit knowledge and diversity to rapidly evolve to the ideal solution and improve each other’s knowledge and skills. In addition, at least one other developer reviewed all code and no participants served as a bottleneck or single point of failure.

Continuous Integration (CI) is another practice for which it is challenging to overstate the value. When combined with automated testing suites this practice allows issues to be detected earlier when they are cheaper to fix. CI also helps keep the workspace open such that all team members directly observe how individual and team activities impact the build.

## Challenges

In an ideal implementation of XP, no code is written before a test for that code. This approach can become tedious for boilerplate code. In addition to the tedium, refactors and maintenance are more expensive because both the tests and code must be refactored. The final layer of tedium IO2 observed was the setup of mock data structures to test.

IO2 anticipated tailoring XP such that pair programming was limited to complex tasks. In practice the team found value in pairing the majority of the time. Due to the lack of domain experience, the pairing was especially useful for spreading the knowledge and resolving design issues.

## Takeaway

Pair programming, TDD, and CI are core practices of XP. There is nothing about traditional development that prohibits the use of these practices. If they are employed, they can add immediate value. These practices are consistent with, and even exceed, some DO-178B Level D criteria by providing both test coverage and code review.

# Conclusion

The primary goal of Agile development is to uncover better ways to develop software. IO2 employed XP to determine how Agile practices could add value to ARINC838 aerospace software constrained to DO-178B Level D compliance. IO2 successfully employed XP to implement ARINC838. The cost of the dynamic requirement was no more expensive than a traditional approach. It is likely that the cost of change was less expensive in the Agile approach. Many Agile core practices including pair programming, test driven development, and Continuous Integration add immediate value by providing real-time code reviews, distributed knowledge, and automated testing suites among other benefits. These practices can be infused seamlessly into traditional development.

# 

# Works Cited

Manifesto for Agile Software Development. 2001. Kent Beck et al. 28 Apr. 2012

< http://agilemanifesto.org/>.

Boehm, Barry W., and Richard Turner. *Balancing Agility and Discipline: A Guide for the Perplexed*. Boston, MA: Addison-Wesley, 2009. Print.

"BibMe: Fast & Easy Bibliography Maker - MLA, APA, Chicago, Turabian - Free." *BibMe: Fast & Easy Bibliography Maker - MLA, APA, Chicago, Turabian - Free*. N.p., n.d. Web. 3 May 2012. <http://www.bibme.org/>.