

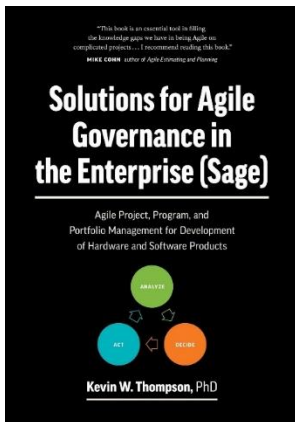
Kevin Thompson
Ph.D. Consulting

When Agile Meets Hardware: Lessons from an Early Agile Transformation

A Thermo Fisher Scientific Case Study



- Dr. Kevin Thompson, Ph.D. (Physics)
 - Has written extensively on Agile topics, including in his book, *Solutions for Agile Governance in the Enterprise* (Sage)
 - Has pioneered the adaptation of Agile processes such as Scrum to the needs of hardware-development organizations.
 - Has conducted more than 100 Agile consulting and training engagements.



Background

- Very few documented examples of such transformations existed in 2016.
- Thermo Fisher Scientific's Life Sciences Mass Spectrometry Division wanted to see how well Scrum would work for mass spectrometer development.
- I had spent 18 months researching how to do Agile hardware development, and made a number of predictions about how Scrum-related practices would have to shift from software norms.
- Research had produced a set of hypotheses.
- Thermo Fisher became the first opportunity to test them on a real hardware product.

Everything I am about to describe contradicted what many Scrum practitioners considered essential to Scrum in 2016—and still do today.

How it Started at Thermo Fisher

- Major biotech company.
- Life Sciences Mass Spectrometry Division makes chromatography, mass spectrometry devices for analytical chemistry applications.
- Benefits observed from Agile techniques in Bremen, Germany led San Jose, CA office to seek our help.
- I described research, recommended Scrum.
- I trained new Team on 'hardware-flavored' Scrum.
- We kicked off development of a new mass spectrometer:
- **We weren't simply introducing Scrum.**
- **We were testing whether Scrum itself could survive contact with hardware reality.**
- And this is where the project-management lessons began...



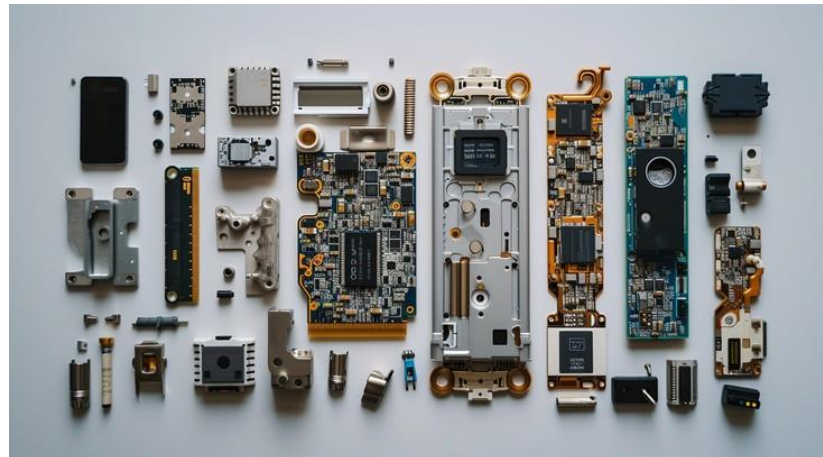
How it Started at Thermo Fisher

- If Scrum were only a software development process, this experiment should have failed.
- Yet one thing surprised me almost immediately. There was very little organizational resistance. The engineers were curious, open-minded, and willing to experiment. The project proceeded with an almost eerie smoothness. The surprises weren't about people—they were about the engineering.



Hardware Changes how Work is Represented

- The way work is represented determines how it can be planned, estimated, prioritized, and tracked.
- Hardware development accreted design elements, not functionality.
- Requirements artifacts were Technical Stories (describing technical deliverables rather than user-facing functionality), not User Stories.
- Team initially confused non-functional requirements which had no deliverables with Stories.
- Scope decomposition was component-oriented, not user experience oriented.

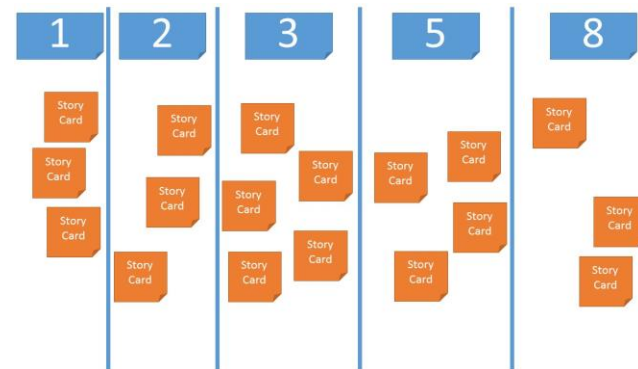


Hardware Changes Scrum Roles

- Roles have to reflect the information required to make good decisions.
- Hardware Product Owners often need greater technical depth than is required of Software Product Owners.
- This hardware Product Owner was a half-time Team member, because the role for hardware required deep technical mastery into the internals of the mass spectrometer.
- In software, Product Owners write User Stories and Team members write Technical Stories.
- For this hardware project, the prevalence of Technical Stories meant that Team members shared the load of writing Stories along with the Product Owner. The result was that the Product Owner wrote fewer Stories than is typical for software teams.

Hardware Changes How We Plan

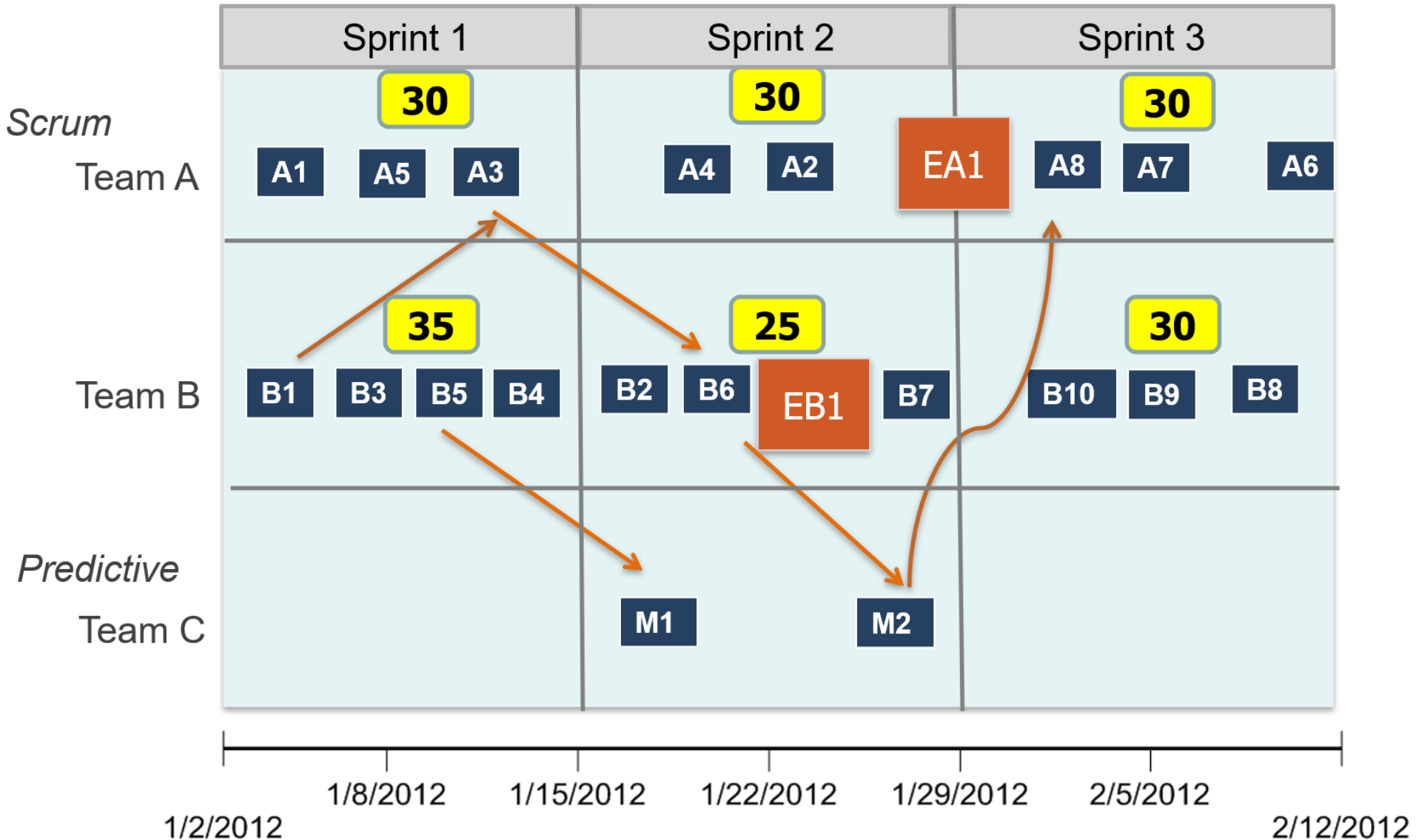
- Planning must reflect the constraints of the work—not the conventions of the methodology.
- High specialization prevented the team from developing a shared sizing norm, making Story Points impractical.
- The Team instead used Person-Days, where one Person-Day is eight hours of time spent working on a deliverable.
 - Planning Poker was still used to estimate Stories.
- The need to address long lead-time items (parts to order, partial prototypes, completed prototype, etc.) drove Release planning.
 - The Affinity Estimation technique was used to estimate Stories and Epics for the Release plan.



Hardware Changes How We Plan

- Ranking (sequencing by value) was still important, but not as dominant a driver as in software development.
- Skill specialization also drove selection of Stories to be planned into a Sprint.
- Velocity, the ceiling for work to be taken on in the Sprint, did matter, but was not sufficient for Sprint planning.
- The Sprint could not simply be filled to its Velocity ceiling with Stories. Skill specialization meant that availability of individual Team members to do work in the Sprint had to be utilized during Sprint Planning to align work with each member's capacity.

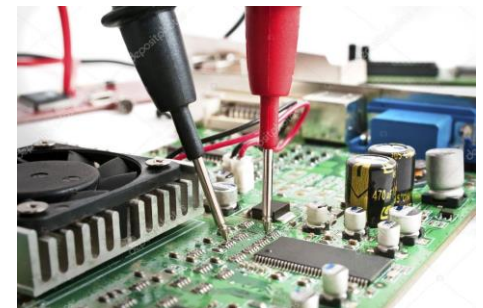
Sample Release Plan



Hardware Changes Execution

- The Scrum Team had no analog to QA personnel, which do not exist at the Team level in hardware development because there is no QA skillset that can test Story-level deliverables.
- Engineers had to test their own Stories' deliverables because only they had the necessary skills.
- Swarming maximizes the number of engineers working on each Story to speed its completion as much as possible. This largely did not work in the hardware environment, because skill specialization in the cross-functional team meant that only one or two people could work on any given Story.
- Maximizing swarming would require organizing around single-discipline teams rather than cross-functional teams, but...
- There is no universally optimal team structure. Optimize for the constraints of the work.

The value of having a cross-functional team outweighs the value of swarming.



Looking Back

- Surprises
 - The complete absence of User Stories surprised me, although the dominance of Technical Stories in general did not.
 - I was surprised at the initial difficulty Team members had with writing Technical Stories, as they wrote a number that reflected non-functional requirements that had no deliverables.
- Predictions Confirmed
 - The need for Release Planning
 - The need for Capacity planning in Sprint Planning
 - The Component orientation
- Every Agile hardware transformation I've worked on since has reinforced these observations.
- The biggest lesson was not that hardware is different. It was that Agile assumptions about how work "should" be managed were often software-derived.

Key Takeaways

1. Adapt the method to the work.
 - Don't confuse a methodology's implementation conventions with its underlying principles.
2. Make work visible in the form the work actually takes.
 - Hardware work is often organized around components, interfaces and technical deliverables rather than user-facing functionality.
3. Plan around real constraints.
 - Specialized skills, capacity, long lead times and physical dependencies matter.
4. Preserve the principles while changing the practices.
 - The goal is not to make hardware development look like software development.
5. Good project management is ultimately about making good decisions.

The Larger Lesson

- Agile is not a collection of practices to be copied regardless of the kind of work being done.
 - It is a set of principles that must be implemented in ways appropriate to the nature of the work.
- The mistake in applying Agile to hardware is not failing to follow the Agile playbook.
 - The mistake is assuming there is only one playbook.

Questions & Comments

Email: kevin@kevinthompsonphd.com

Website: <https://kevinthompsonphd.com>

LinkedIn: <https://www.linkedin.com/in/kevinthompsonphd>