Thermo Fisher Scientific: Lesson #3 – Scope Decomposition Is Not Driven by User Experience

One of the first challenges facing the Thermo Fisher Scientific team was defining the scope of the product in a way that would support planning and execution.

The team understood the product concept very well. They knew what they wanted to build and why it mattered. What was not yet clear was how to organize that understanding into a structure that could support Release Planning, Story writing, and Sprint execution.

In software development, I would normally begin by identifying the major areas of user-facing functionality. From there, we would define the supporting infrastructure and other technical capabilities needed to deliver those features. The result would be a collection of Epics that collectively described the product as it was currently understood.

This approach had worked well for many software organizations.

The problem was that this was not a software product.

I was not an expert in mass spectrometers or analytical equipment, and I had no preconceived notion of how the team should organize its understanding of the product. Rather than imposing a structure, I decided to start with a simple question.

"How do you think about the major pieces of this device? How do you organize your understanding of the design of the product you want to build?"

Then I sat back and listened.

The discussion immediately focused on the physical components of the product. Team members talked about subsystems, assemblies, and other elements of the device's architecture. Very little attention was paid to user experience.

This was strikingly different from what I typically observed in software development environments. Software teams usually begin by discussing what the product does. This team was focused primarily on what the product is.

At first, I was uncertain whether this was the right approach.

Was the team overlooking something important? Were they thinking about the problem incorrectly? Or was I observing a fundamental difference between hardware and software development?

To test the idea, I periodically steered the discussion back toward user experience and user-facing capabilities.

The team was perfectly willing to discuss those topics. However, the conversations rarely produced additional clarity about the work that needed to be performed. Before long, the discussion would naturally return to the product's components and subsystems.

Over time, it became clear that the team understood exactly where its attention needed to be.

Unlike software products, which are often organized around user-facing capabilities, hardware products are organized around the physical systems that must be designed, built, integrated, tested, and ultimately manufactured.

Once I accepted that reality and followed the team's lead, the scope-definition effort progressed quickly. The resulting structure provided a clear foundation for writing Epics, planning Releases, and organizing development work.

The lesson was an important one.

Lesson #3: Scope decomposition for hardware products is typically driven by components and subsystems rather than by user experience.

This does not mean that user needs are unimportant. The product still exists to solve customer problems. However, when planning and organizing development work, hardware teams naturally think in terms of the physical elements that must be designed and manufactured to deliver those outcomes.

For someone coming from a software background, the distinction is subtle but important. It influences how requirements are organized, how work is decomposed, how Releases are planned, and ultimately how Agile practices are applied in a hardware environment.

In the next post, I will discuss another major difference between hardware and software development: the very different ways in which products evolve during their development lifecycles.

For those interested in the full case study, the complete Thermo Fisher Scientific story is available on my website at https://kevinthompsonphd.com/storage/papers/LessonsLearned-AgileHardware-v6.pdf.