Thermo Fisher Scientific: Lesson #4 – Hardware Products Accrete Components, Not Functionality

As the Thermo Fisher Scientific project progressed, I began to notice another fundamental difference between hardware and software development.

Even within a single Sprint, the evolution of the product looked very different from what I was accustomed to seeing in software organizations.

In software development, Scrum teams typically spend most of their time implementing features. At the end of a Sprint, the product usually provides some new capability that was not present at the beginning of the Sprint. The product evolves by gradually accumulating functionality.

A user who examines the product at the end of each Sprint can often see a steady progression of new capabilities being added over time.

The Thermo Fisher project followed a very different pattern.

At the beginning of the project, there was no product in any meaningful sense. There was no device that could be demonstrated to a customer, and there was no user experience to evaluate.

Instead, there was a growing collection of designs, prototypes, breadboards, components, and subsystems. Sprint by Sprint, the team created and refined these elements. Individually, many of them provided little visible value. Collectively, however, they moved the project steadily toward a working product.

Over time, more and more of the required pieces came into existence. Components were designed. Subsystems were tested. Interfaces were defined. Integration activities increased. Eventually, enough of the pieces existed that they could be assembled into a functioning system.

This was very different from the pattern typically seen in software development.

Software products tend to evolve by accumulating functionality. Hardware products tend to evolve by accumulating components and subsystems.

The distinction may seem subtle, but it has important implications.

It influences how progress is measured, how work is planned, how risks are managed, and how stakeholders evaluate the state of the project. Expectations that make perfect sense in a software environment can become misleading when applied directly to hardware development.

One of the recurring themes of the Thermo Fisher engagement was that hardware development is not simply "software development with longer lead times." The underlying dynamics are often quite different.

This observation about product evolution was another example of that principle.

Lesson #4: Software products in development accrete functionality over time, while hardware products in development accrete components and subsystems over time.

In the next post, I will discuss how this difference influences the role of User Stories in hardware development.

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.