Thermo Fisher Scientific: Lesson #9 – Story Points Do Not Work Well for Hardware Teams

Every Scrum team must estimate work in order to plan Releases and Sprints.

At Thermo Fisher Scientific, we used the same estimation techniques commonly employed by software teams. Stories were estimated during Sprint Planning using Planning Poker, while larger Epics were estimated during Release Planning using Affinity Estimation. (It is important to understand that either estimation technique can be used with any system of units.)

The challenge was not the estimation techniques.

The challenge was the units.

The Thermo Fisher team strongly preferred effort-based estimates from the beginning. Understanding why requires understanding how Story Points work.

Most Scrum literature recommends estimating work using Story Points. Story Points are intentionally abstract. They are not tied directly to time, effort, or any other measurable quantity. Instead, they represent a team's relative assessment of the size or complexity of a piece of work.

The process begins by establishing a reference scale. Team members compare candidate Stories, arrange them from smaller to larger, and assign values such as 1, 2, 3, 5, 8, and 13.

These numbers come from the Fibonacci sequence, in which each number is the sum of the previous two. The sequence has the useful property that the spacing between adjacent values grows as the numbers become larger. This reflects the reality that uncertainty increases as work becomes larger and more complex.

The result is a scale that is fine enough to support planning, but coarse enough to avoid time-wasting discussions about whether a Story should be estimated as a 10 or an 11. As uncertainty grows, the scale intentionally becomes less precise.

Future Stories are then estimated by comparing them to previously sized work.

This approach works surprisingly well for many software teams.

The reason it works, however, is rarely discussed.

For a team to create and use a common sizing scale, team members must possess enough overlapping knowledge to develop a shared intuition about the relative size of the work being estimated.

Software teams often satisfy this condition. Team members may have different specialties, but they frequently function as what Agile practitioners call *generalizing specialists*. They possess a primary area of expertise while still understanding the work performed by their teammates well enough to participate meaningfully in estimation discussions.

The Thermo Fisher team did not fit this model.

Like many hardware-development teams, it was composed of highly specialized experts. Team members possessed deep knowledge in areas such as electronics, optics, firmware, physics, and mechanical engineering.

These specialties were cross-functional from an organizational perspective, but they were not interchangeable.

An optics engineer could not reliably estimate a complex electronics task.

A mechanical engineer could not reliably estimate a sophisticated FPGA design effort.

As a result, the team could not create a common Story Point reference scale that everyone understood and trusted.

This was not a failure of the team.

It was a consequence of the specialized nature of hardware development.

Fortunately, there was another option.

For many years I had taught two possible units of estimation in my Scrum classes: Story Points and Person-Days.

A Person-Day represents eight hours of effort spent implementing and validating a Story. It is a measure of effort rather than calendar duration. A Story estimated at one Person-Day of effort will often require more than one day of elapsed time because engineers divide their attention among meetings, collaboration, reviews, and other activities.

Like Story Points, Person-Days can be estimated using Planning Poker and Affinity Estimation. The difference is that the estimates are expressed in units that engineers naturally understand.

The Thermo Fisher team immediately preferred this approach.

Rather than debating the relative size of work across specialties, engineers could estimate the effort required within their own domains of expertise.

The estimation sessions proceeded smoothly, even though not every team member could estimate every Story. Engineers with relevant expertise provided estimates, while others contributed questions and insights.

After estimating the Stories using Planning Poker, the team decomposed each Story into tasks and estimated the effort for those tasks in Person-Hours. The task estimates provided a useful cross-check on the Story-level estimates and increased confidence in the resulting Sprint Plan.

The outcome was highly successful.

The team found the estimates intuitive, planning discussions were productive, and Sprint Planning proceeded without difficulty.

This experience reinforced another conclusion from my Agile Hardware research.

Lesson #9: Story Points depend on a shared understanding of the work being estimated. Because hardware teams are often composed of highly specialized experts, effort-based estimates such as Person-Days frequently provide a more practical alternative.

The lesson is not that estimation is impossible in hardware development. Rather, it is that hardware teams often require estimation approaches that acknowledge their highly specialized skill sets.

In the next article, I will discuss another Scrum concept that behaved differently in a hardware environment: Velocity.

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.