Path to Hardware Agility

Maarit Laanti, 01-February, 2024

Maarit Laanti

Ph.D., SAFe Fellow, SPCT 6.0, RTE 6.0

2023 Partner @ wikiflow, Partner @WikiAgile
XP 2023 conference Education & Learning track chair
Trained > 3000 people

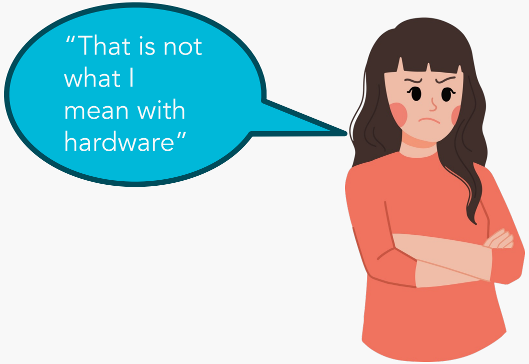
2020 SAFE® Fellow
2019 Nominated to LIA100 – top 100 women contributing Lean & Agile

2016 SAFe SPCT
2014 Contributor to Lean-Agile Budgeting in SAFe 3.0
2013 Founded of Nitor Delta, and brought SAFe to Europe
Ph.D. "Agile Methods in Large-Scale Software Development Organizations Applicability and Model for Adoption"
2012 20 years of product development at Nokia

WikiAgile & wikiflow Partner, SAFe Fellow, SPCT 6.0

My path to Hardware Agile

2009
Started hardware workshops on the Electronics (motherboard, display, and camera)
Research paper Piloting Lean-Agile Hardware Development



2020
1st Agile Hardware training & transformation coaching "Heavy metal"

2021
2nd Agile Hardware Training & transformation coaching "inside chips"

2022 Collaboration with Joe Justice started
3rd Agile Hardware Training

2024
Whitepaper "Accelerating Product Development with Agile Practices in Hardware Design"

Download the whitepaper from wikiagile.com!

wikiagile.com

Recent Publications

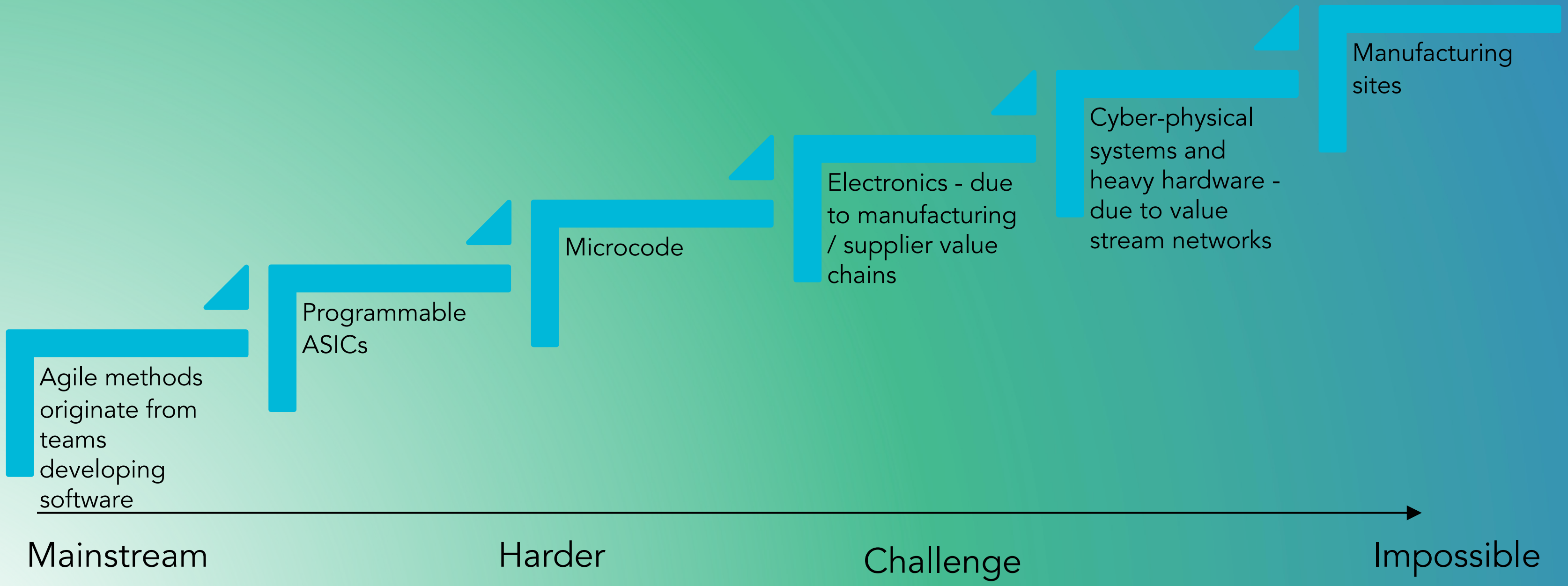
Accelerating Product Development with Agile Practices in Hardware Design

by Maarit Laanti

This whitepaper presents stepwise guidelines on how to speed up your product development using agile methods in hardware development. The work is based on our decades-long experience in product and process development and coaching in the largest and fastest product development enterprises. The goal is to provide customer value sooner – gaining speed in product development is an essential step in that pursuit.

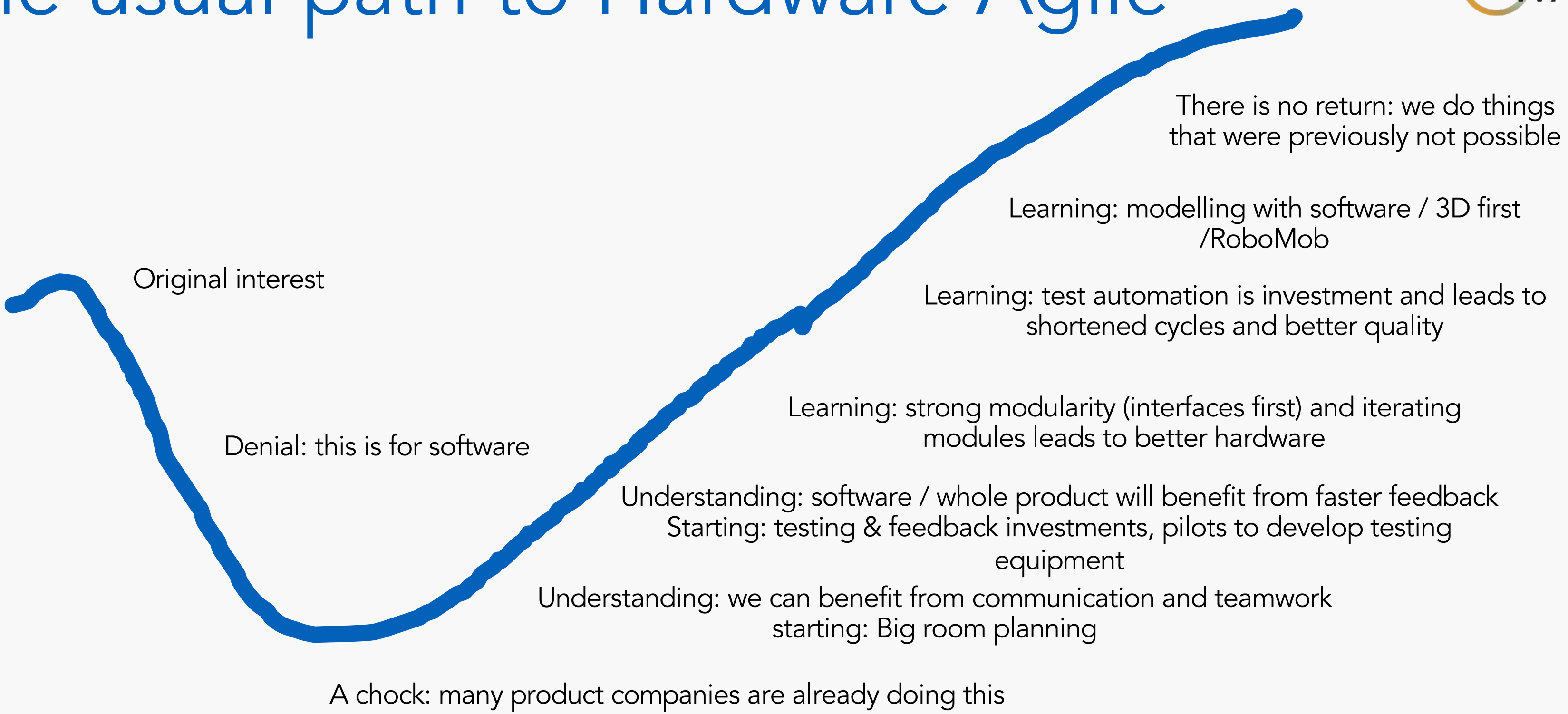
Find out more

How hard we see adoption of agile?



5

The usual path to Hardware Agile



Warning: There is no such thing as Agile Hardware Development

FROM

- Product development division to hardware and software departments
- Quality verified at the end
- Long design cycle targeting to error-free manufacturing instructions

TO

- Hardware or product being modeled by software or 3D before it gets built; digital twins
- Test-first approaches
- Digitalization leading to disruption: innovation is what matters – concurrent engineering taken to extreme

- It is a holistic Agile Product Development Change

Why Point-Based Design Fails?

Bicycle Factory Example

Typical case: five subsystems; each finding optimal solution on their own

- Suspension
- Gears
- Brakes
- Wheels
- Frames

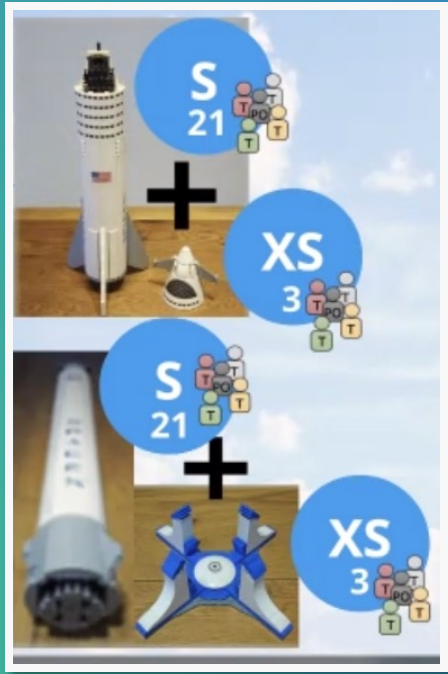
Subsystem design is done so that it will best support the system design

- ✓ 20% change that each subsystem will cause major problem (market miss, project delay, etc.) + 40% change it will not work well together
- ✓ Probability of ach subsystem to success = 0.8; for systems integration = 0.6, Overall success probability 0.2!!!

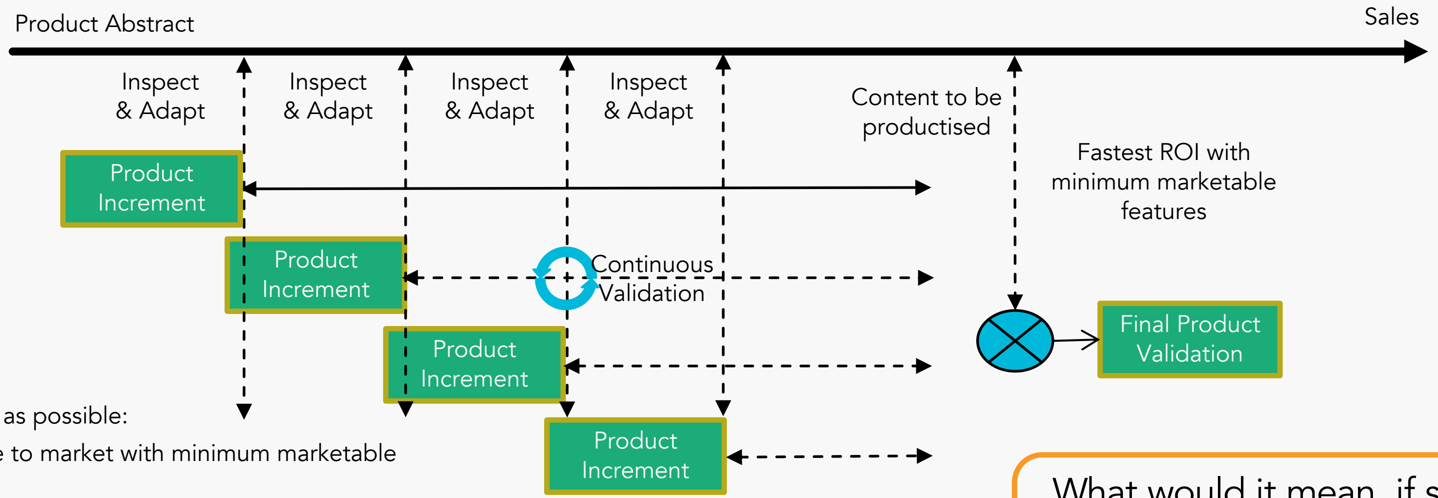
Why Hardware Agility Pays Off?

Split Large Projects into Small, Independent projects

- ✓ Object-oriented architecture is an agile risk reduction strategy
- ✓ Optimum project size increases as technical excellence and automation improve



Incremental Approach for Product Development

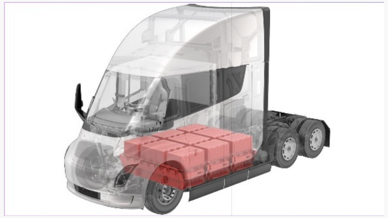


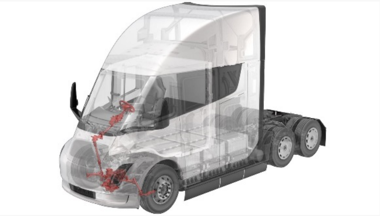
Deliver as fast as possible:

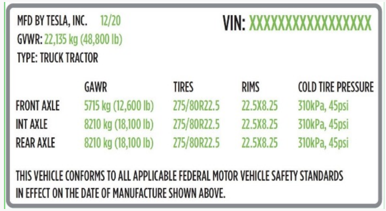
- ✓ Fastest time to market with minimum marketable features
- ✓ Continuous validation ensuring fast enough final product validation
- ✓ Continuous inspect & adapt cycles ensuring fast feedback and adaptation to changes

What would it mean, if some of these increments were software models of hardware?

Pre-requisites for Agile Hardware Development
Understand & define modules

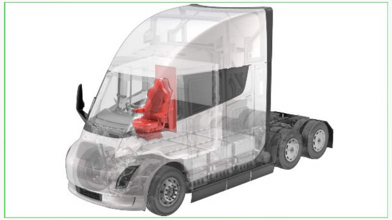




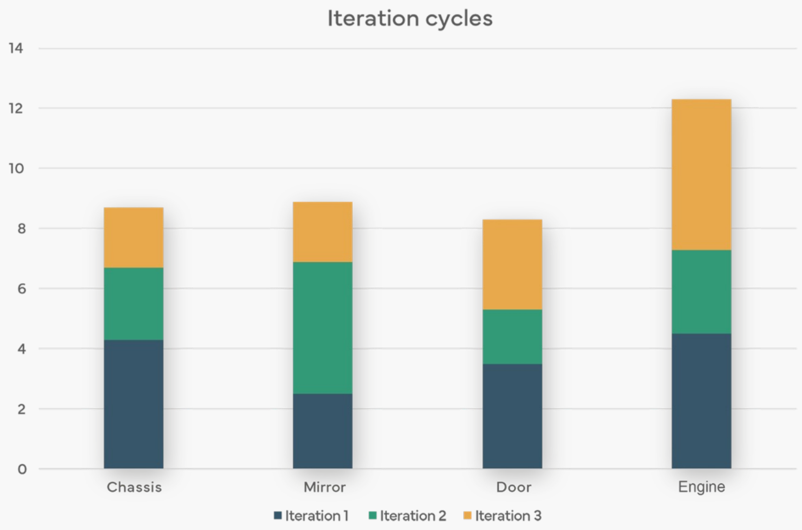



Modular architecture is the key!






Step 1 – Identify current iteration lengths on hardware

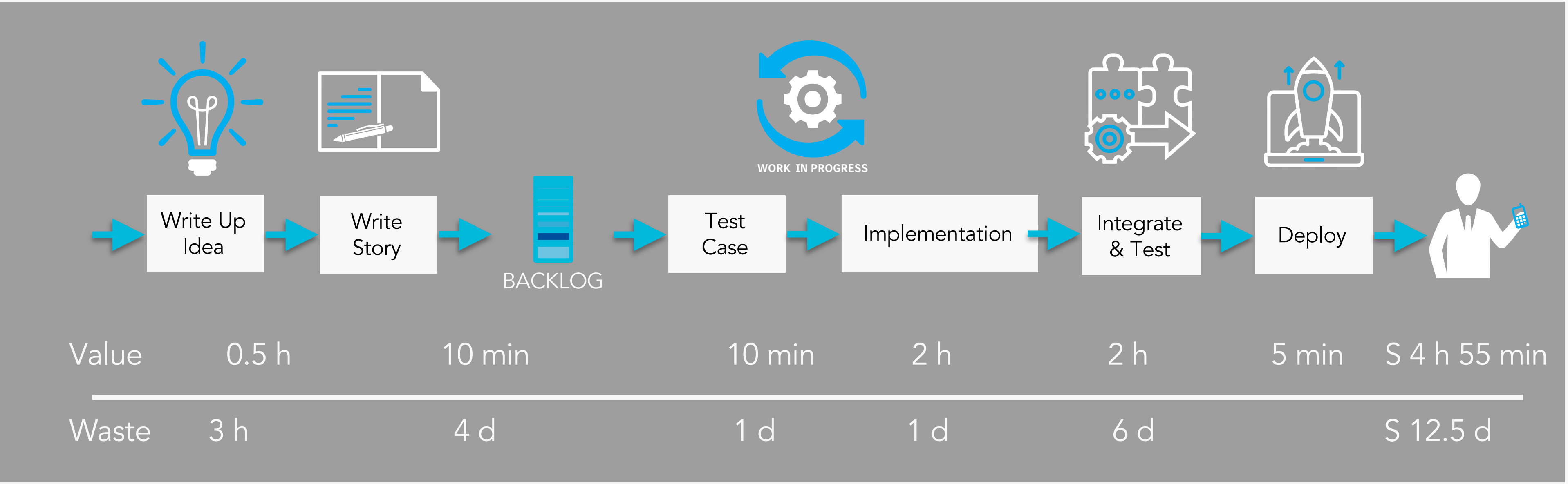


Option to speed up: Consider modularization of the part that has the longest iteration cycles. This could give several hundred percent improvements in overall time-to-market, resulting in significant business benefits.

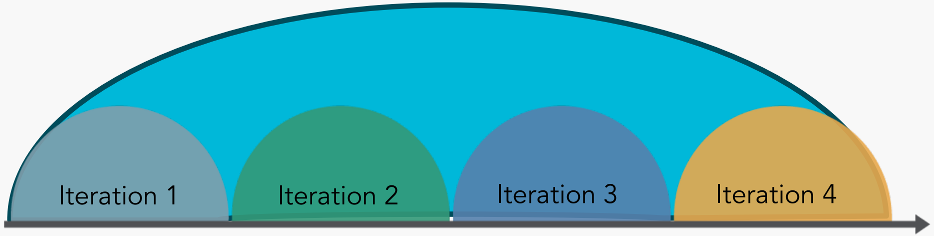
- Different hardware modules have different iteration lengths, and the amount of time needed
- Iteration cycle length may vary a lot depending on what type of hardware you are working on
- The cycle starts from design or specifications and ends when you test it
- The overall cycle time within that specific value stream is a sum of all iterations
- Note that a same module can be used in multiple value streams

Step 2: Apply value stream thinking to development value streams

- Understand which teams contribute to overall design together
- Create a set of connected Kanbans to manage the work in Value streams



Step 3: Implement Rolling planning



Rolling planning involves two-level approach, consisting of:

1. A high-level overall schedule, and
2. A more detailed schedule for the next iteration level

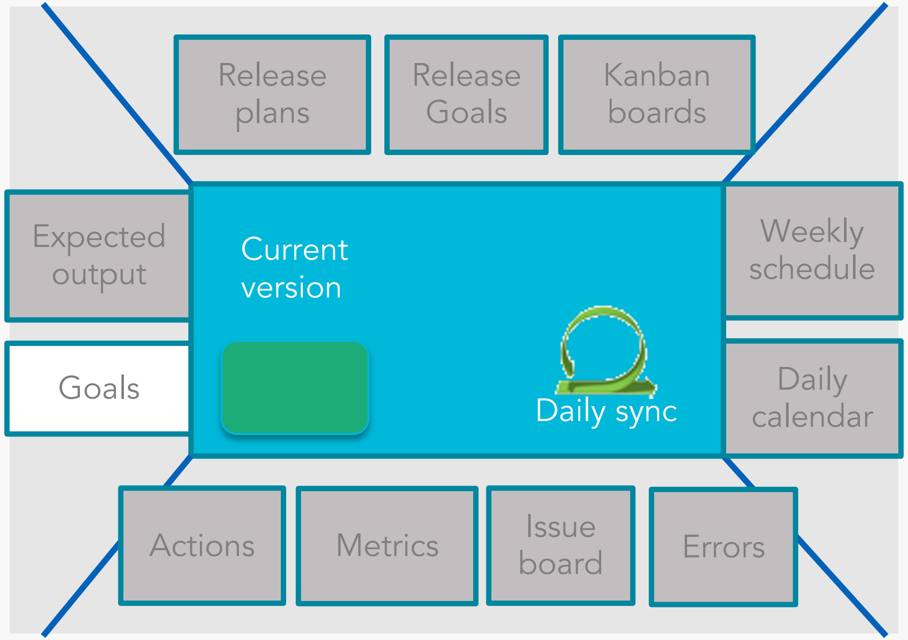
- Rolling planning is needed because of "single" delivery
- As we execute each iteration, we reflect the learning to the higher-level plan and adjust this long-term plan as we see necessary

Step 4: Synchronize the iterations of the hardware modules

- With hardware, we cannot follow/ adjust to just any cadence – we need to build the frame understanding our own development
- Synchronization allows the creation of common testing points
- If the 4-week cycle syncs every 6 weeks we can test every six weeks – with no sync every 12 weeks

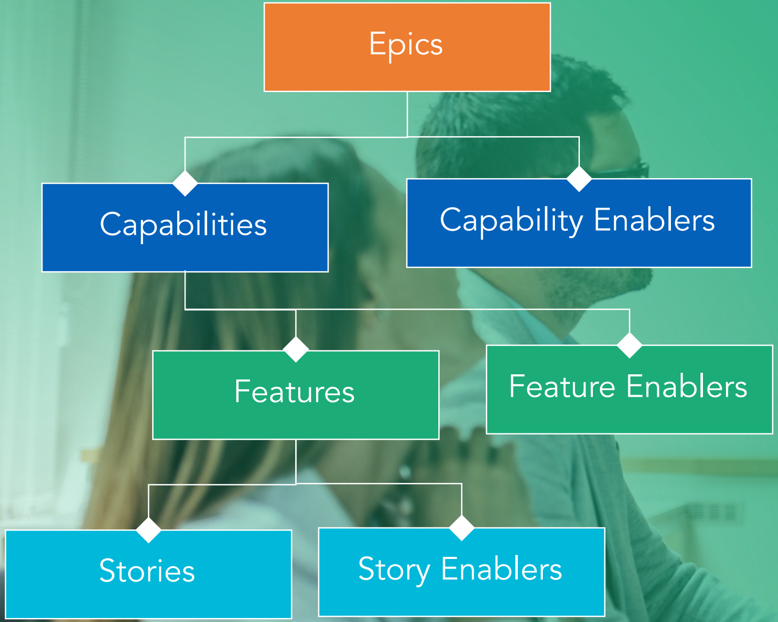
- Quite often we see hardware organizations working with a 3-month cadence
- What are the options of speeding the feedback?

Step 5: Apply Big room planning for managing dependencies and introducing cadence



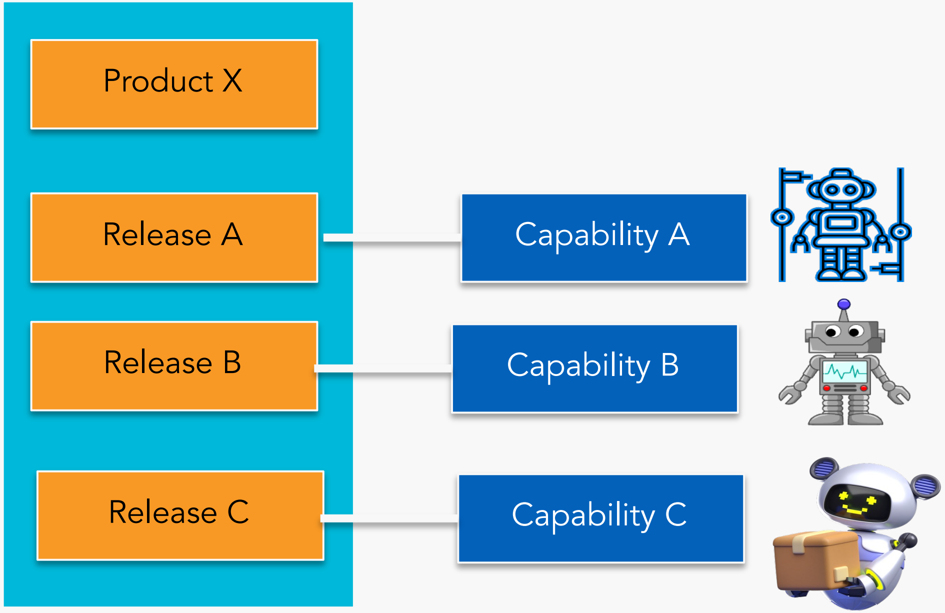
- Agile hardware development starts usually as a design & process change, but in the second phase a tools change is needed
- Visual planning is easy practice" to start with → just improves communication, and does not require anything specific to work
- A modern, digital "war-room" with up-to-date information helps to keep everyone on the same page; motivated and aligned

Step 6: Define hardware modules as Capabilities



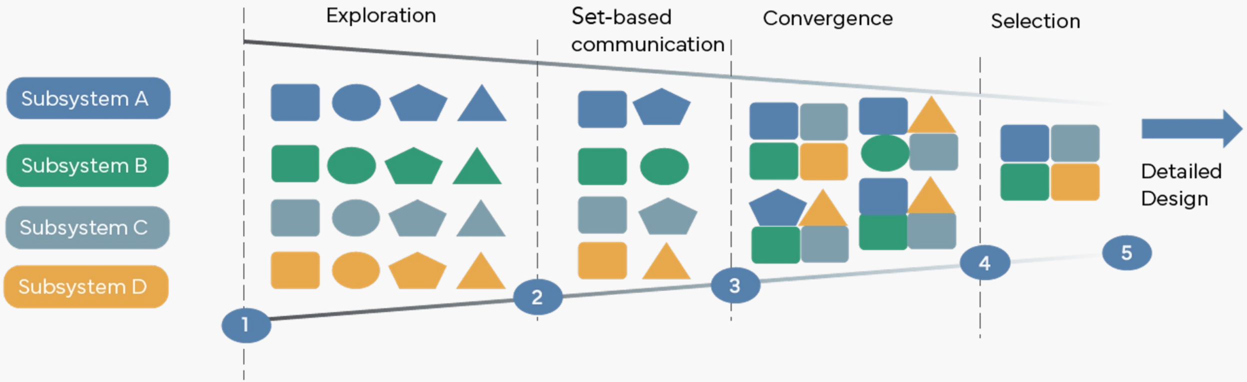
- A common information model allows modelling the requirements for the entire product – software, and hardware
  - A Story may or may not have a parenting Feature
  - A Feature may or may not have a parenting Capability
  - All Capabilities, Features, and Stories will have an identifier for traceability – and for fulfilment of any compliance requirements

They may be several versions of Capabilities that go to different releases



- This is supporting modular iterations of hardware components

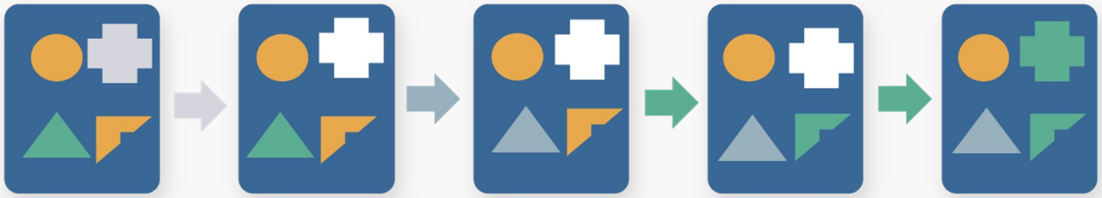
Step 7: Apply Set-based design for managing risk and product lifecycle



- Find design constraints, and find solutions that fit into that space
- Integrate and explore the best solutions
- Limit the final selection at the very end

- Use set-based design, especially in areas that contain the highest reward and highest risk
- This allows design to co-evolve in the future

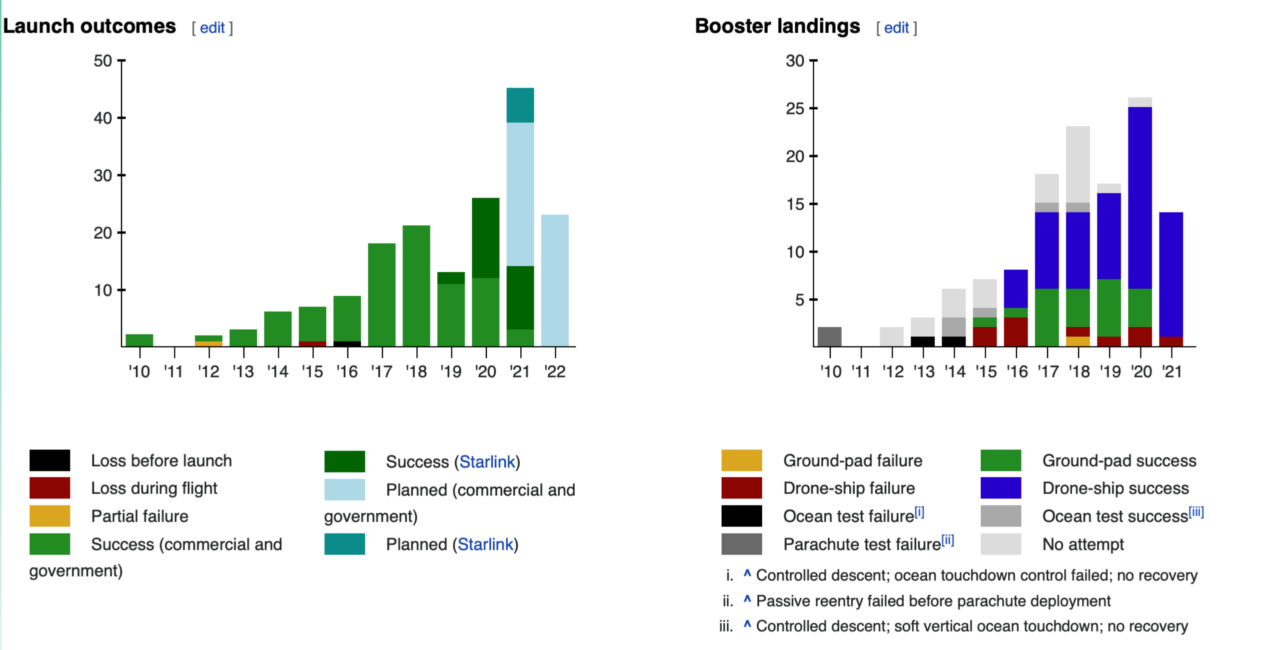
The future: Products as a Service Business?



- Agile has enabled continuous releases & transformed vendors from product to service business
- Modules having different life cycles are designed to be exchanged in the future. In this sequence of product upgrades one module at a time is exchanged for a newer version. This extends the whole product lifecycle significantly.

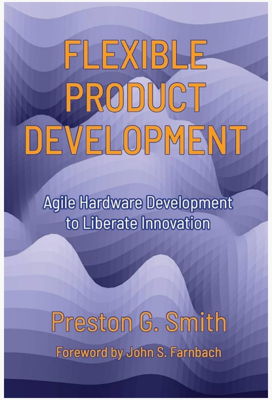
- The United Nations, together with the European Union, has set ambitious goals for Sustainable Development Actions (SDGS). Goal number 12 is about Responsible Production and Consumption.
- One of the goals listed for this initiative is to "Support developing countries to strengthen their scientific and technological capacity to move towards more sustainable patterns of consumption and production".

Yes – it is all possible! See SpaceX



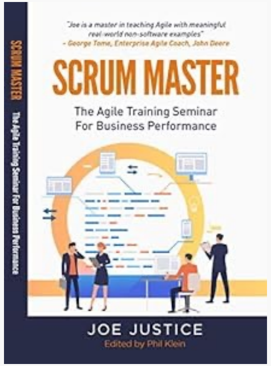
Source: Wikipedia, https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches

More Information on



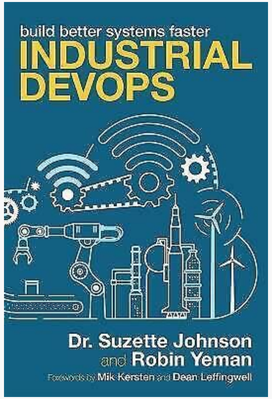
2018

2019



2021

2023



2023

Thank you

Maarit Laanti

maarit@wikiagile.com
www.linkedin.com/in/mlaanti

WikiAgile
BUILDING AGILE FUTURES TOGETHER