Agile in Physical Product Development
WHY?
Agility in Software Dev is 20 years further along than in Physical Product Dev
greater complexity in Physical Product Dev is NOT an argument AGAINST agility!
Gartner model - a possible path from product strategy to marketable product (by colleague agilists)
10 specific challenges for agility in Physical Product Dev & what to do...
1. Definition of potentially shippable product increments?

→ define MVP/MVF (minimal viable product/features)
→ define prototypes/testable results
2. Definition of Done

- adapt the DoD continuously
- focus on integrative Acceptance Criteria
3. necessary parallelization

→ Simultaneous Engineering (sensorical, mechanical, electrical, electronical)
→ share data
→ Peer Know How Flow
4. prototyping

→ make use of Rapid Prototyping
→ construct proper environment for development (& integration)
5. testing

→ early User Feedback
5. testing

→ Automated Testing
→ Virtual Testing/Simulations
5. testing

→ Automated Testing
→ Virtual Testing/Simulations
5. testing

→ Automated Testing
→ Virtual Testing/Simulations
6. Change Management

→ define sensible modules & components
→ keep cost of change low
→ welcome change even late in dev...
7. dependencies (internal/external)

→ visualize and integrate processes in planning early on (supplier integration, partnership dev...)
8. constraints

→ visualize and consider physical limitations in product design early on & continuously
9. regulations, rules & norms

→ visualize and integrate necessary processes and limitations early on & continuously
10. continuous delivery & integration

→ define as concrete goal of product dev
→ achieve through steps of continuous improvement
Trumpf
TruLaser Center 7030 - only 2D laser cutter with fully integrated automated process

2 years of development
100 team members
3 core innovations
  = major success!
  (Agile Coach: Heinz Erretkamps)
Wikispeed

“From **Lean software design** we take the concept of using less stuff wherever responsible. This is based on the common-sense mandate to “use less stuff,” which is then defined in a clear, applicable way by the contemporary software team.

From **Extreme Programming (XP)** we take the practices of pairing and swarming. These practices date back at least as far as the apprentice model but have been carefully defined to replace the need for most types of training and process documentation.

From **Agile software development** we take the principle of reducing cost to make change—changes in team, materials, machinery, and even goals.

From **Scrum software development** we take clearly defined team roles and responsibilities, which allows us to spend more time rapidly developing product with no nonworking (management only) roles and only two meetings.

From **Test-Driven Development** we start with failing tests and then develop solutions. This allows us to quickly identify if current work is not targeted to pass a test or is causing problems elsewhere in the system, which avoids waste.

From **Object-Oriented Programming** we take Contract-First Development, which enables the modularity of the WIKISPEED car and all of our solutions.”
Start - before you’re ready!

And don’t forget: Agile has no brain – use your own!
Katrin Wellmann, Dipl. Des.
Agile Coach & Business Change Consultant @ Prowareness WeON Germany @Scrum Day Europe 2018

Questions & feedback welcome!