Lean-Agile Development for Large Enterprises: Adding Hardware to the Mix

Harry Koehnemann, Director of Technology at 321 Gang, Inc.
@HarryKoehnemann

Alex Yakyma, Fellow and Principle Consultant at Scaled Agile, Inc.
@AlexYakyma

© 2016 Scaled Agile, Inc. All Rights Reserved.
Agenda

- Organizing around value
- Aligning on a common cadence
- Lowering hardware batch sizes
- Redefining the Systems Engineer role
Organizing around value
Why organize around value?

*Align the organization around projects and product lines.*
—Allen C. Ward

1. Fewer handoffs, faster value delivery
2. Easier to build in quality
3. Built-in alignment between the business and software development
4. Optimizing the system as a whole

Result: Faster delivery, higher quality, higher customer satisfaction
Building teams
Myth #1

People from different engineering disciplines (Software, Firmware, Hardware, Mechanical, etc.) can’t work together
Myth #2

You can just adopt Agile practices here and there and you will be fine!
Realizing Value Streams
Realizing Value Streams

VALUE STREAM

[Diagram showingValue Stream with people and arrows indicating flow]
Realizing Value Streams
Realizing Value Streams
Example 1

- Ignition
- Fuel
- Transmission
- Performance
Example 2: Scaling up

The relationship is not always trivial “one-to-many”
Organizational Paradigms

Things to take into consideration:

- Capabilities, features
- Subsystems, components
- Business demand
- Commonality / Variability (PLE)
- Coupling
- ...

© 2016 Scaled Agile, Inc. All Rights Reserved.
Exercise: Organizing cross-functional teams around value.

- At your table, discuss the relative merits of component/part-centric vs. feature-centric organization with the addition of hardware elements

- Be ready to share with the group
Aligning on a common cadence
Myth #3

*Hardware development cannot be Agile*
Cadence creates the vital pace for the Value Stream

Cadence supports natural Work-In-Process limits that foster learning cycles across the Value Stream

Plan → Do
Act → Check

Program Increment (PI)
4-6 iterations (8-12 weeks)
Plan, execute and measure on the cadence

Planning, execution and demo become predictable events. Make trade-offs to be able to plan after you demo’d the previous one.
Frequent Integration and Testing

- The goal of frequent integration and testing is to provide frequent feedback on newly built functionality
- The goal dictates the approach
- Trade-offs are inevitable in terms of:
  - Frequency of integration
  - Depth of integration
  - Fidelity of feedback
Solution Integration is a set of trade-offs

- Solution integration is realized as a combination of real subsystems and proxies of various complexity
- Segregation of interfaces supports the ability to grow the level of fidelity (SIL → HIL)
- Maturity of infrastructure and practices lowers cost of integration over time
Trade-off example: Fidelity of feedback

The cost of integrating an actual subsystem can be too high while substituting it with a stub may provide no useful information. Every subsystem dictates a unique trade-off.
Proxy Example: FPGA

```vhdl
process (N_RESET, DATA_READ, START_ADDRESS, END_ADDRESS)
begin
  if (N_RESET = '0') then
    ADDR_TMP <= START_ADDRESS;
  elsif (DATA_READ'event and (DATA_READ = '0')) then
    if (ADDR_TMP <= END_ADDRESS) then
      ADDR_TMP <= ADDR_TMP + '10';
      inc_tmp = inc_tmp xor '1';
    end if;
  end if;
end process;
```
Other tools and trade-offs

Integration Sequence
Other tools and trade-offs

Integration Sequence
Other tools and trade-offs

Integration Sequence

![Diagram](image-url)
Other tools and trade-offs

Integration Sequence
Other tools and trade-offs

Integration Sequence
Other tools and trade-offs

Integration Sequence

- Physical assembly
- Subsystem connectivity
- Firmware Deployment
- Software Deployment
- System scenario test
Other tools and trade-offs

Integration Sequence

Scoping within a PDCA

The greedy pay twice!!!
Examples of designing for flow
Exercise: Aligning on a common cadence.

- Discuss how non-software disciplines can participate in traditional agile activities – planning, demos, retrospectives, and system integration and testing.

- Be ready to share with the group
Lowering hardware batch sizes
Decompose large initiatives into small batches

- Grow key capabilities and initiatives over incremental milestones
- Each increment focuses on gaining knowledge and/or demonstrating parts of the solution
- Chief engineer sets vision; teams determine detailed plans
Enable late decision making with smaller batch sizes

Aggressively evaluate alternatives. Converge specifications and solution set.
—Allen Ward, Lean Product and Process Development

- Fast PDCA cycles with small batches can quickly validate many assumptions
- Allows teams to simultaneously explore multiple options over longer period of time
- Suboptimal options dropped as they become inferior or no longer cost effective to pursue
Manufacturing/deployment are part of overall value stream

- Include manufacturing/deployment and their concerns in the development process
- Explore concerns each increment, in flow with rest of work
- HINT: apply same approach for other, “external” concerns (security, quality, safety, etc.)

- *Determine vehicle location options*
- *Understand mounting constraints*

- *Understand changes to assembly process*
- *Know supply chain impact*
Beware obstacles to lower-fidelity solutions

- **Obstacles**
  - Supply-chain/purchasing
  - Quality organizations
  - Corporate standards and procedures
  - Manufacturing availability

- **Guidance**
  - Don’t call anything a “release”
  - Consider above obstacles as part of the overall value stream
  - Train *everyone* in organization on lean principles – flow, small batches, cost of delay
Exercise: Lowering hardware batch sizes.

- Discuss practices that facilitate early feedback on hardware systems. What are the relative cost-benefit tradeoffs?
- How would you determine the minimal units of value to pull through an increment?

Examples
- Custom mockups (3D Printing, dev kits)
- MIL, SIL, HIL
- Parts for existing systems, programs
- Internal, low-overhead manufacturing
Redefining the Systems Engineering Role
The Lean-Agile Systems Engineer

Old habits  ❌

- Defines detailed, up-front specifications
- Drives teams to meet detailed, fixed schedule
- Hands off detailed specifications and plans to teams
- Technical leader, primary decision maker, bottleneck for questions/issues

New habits  ✔

- Supports emergent specifications; adapts based on new knowledge
- Owns vision and roadmap; allows teams to implement solution
- Understands face-to-face is best form of communication
- Understands face-to-face is best form of communication
Supports emergent specifications; adapts based on new knowledge

Assume variability, preserve options

- Provide flexibility to solution builders by minimizing up-front constraints
- Use regular cadence for teams to explore alternatives and converge on a timely decision
- Simultaneously build what we know while we explore what we don’t know
Owns vision and roadmap; allows teams to implement solution

- Define inspiring vision and flexible roadmap with challenging goals
- Facilitate team planning and committing to near-term work
- Use regular cadence to learn about team’s ability to deliver ultimate solution
Understands face-to-face is best form of communication

- Use specifications to record knowledge and meet compliance; not handoff decisions
- Leverage agile intentional, face-to-face interactions (plan, stand-up, demo, retro, specification workshop, scrum-of-scrums, etc.)
Servant-leader focused on team growth and mutual influence

- Provides team knowledge necessary to make localized, time-critical decisions
  - Teams see the big picture, understand economic impact of decisions, etc.
- Creates a team jointly responsible for overall solution success
- Produces a strong vision with challenging goals
- Builds a safe, trusting environment for learning, growth, and mutual influence
QUESTIONS

harry@321gang.com
alex.yakyma@scaledagile.com

@HarryKoehnemann
@AlexYakyma