

Raytheon
An RTX Business

# Increasing the Agility of Agile throughout the Program Lifecycle

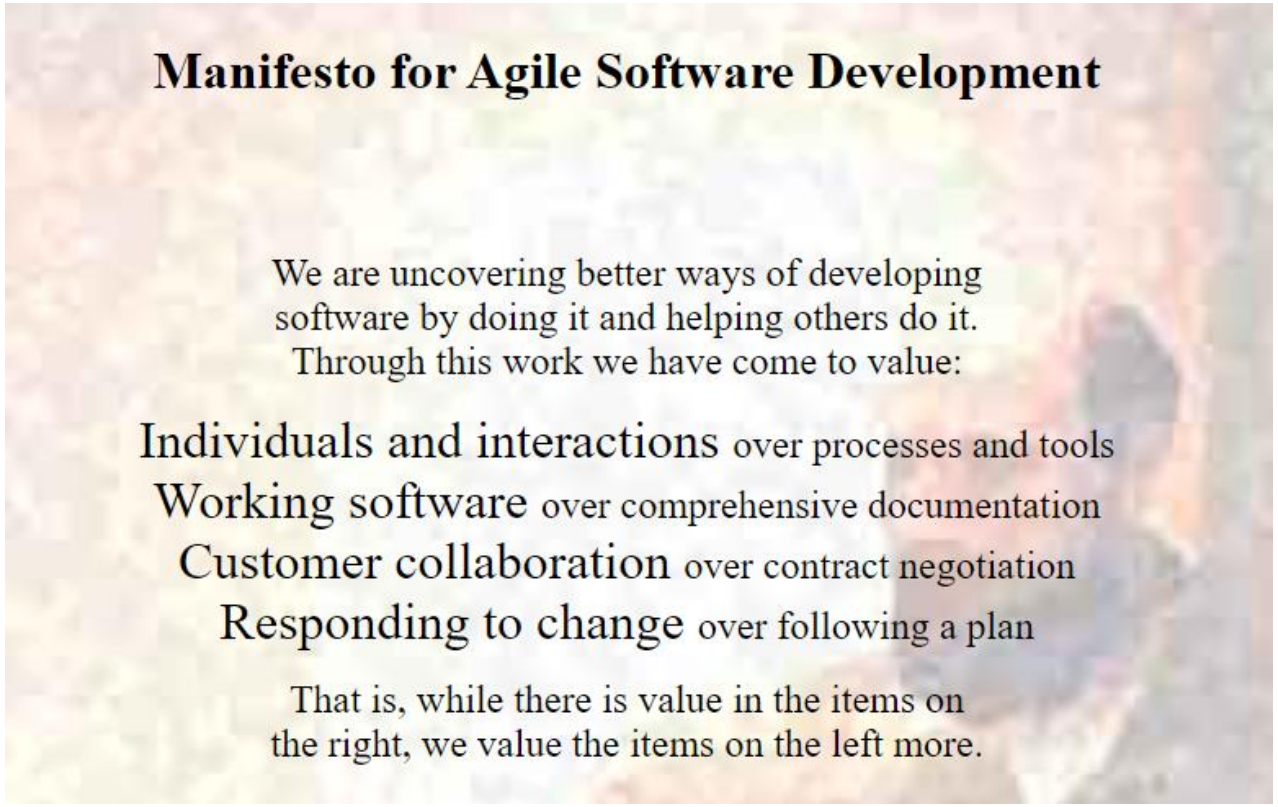
Kurt Mohr, Sr. Fellow

October 28, 2024

# Agenda

- Agile Defined
- Evaluating what's important
  - Program Startup
  - Early Program
  - Core Development
  - Integration and Test
  - System Completion and Delivery
- Agile methodology decision matrix

# What is “Agile” within this Context?

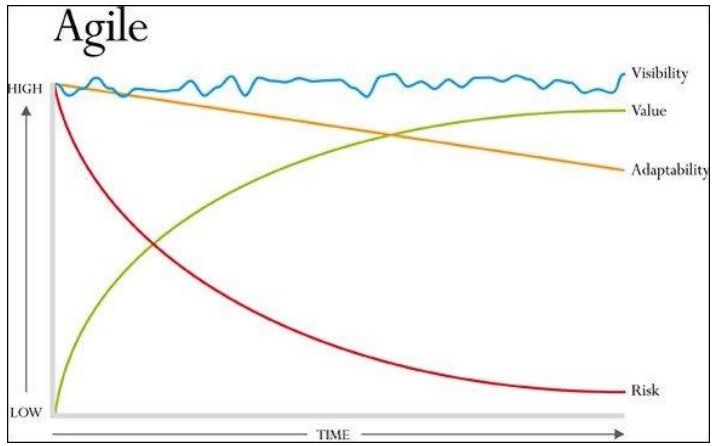


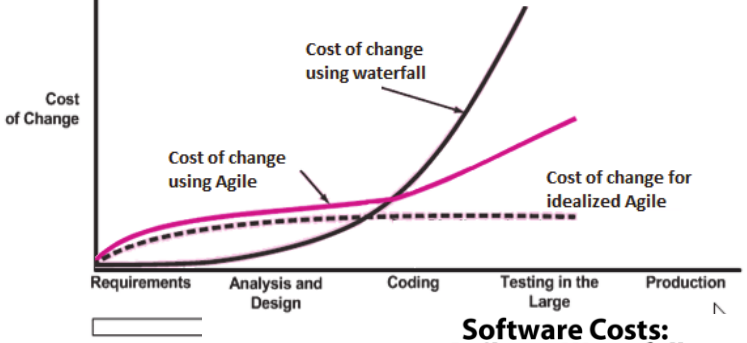

- Working software is the core criteria
- Principles are customer and delivery focused
- Technical excellence and good design
- Simplicity and constant improvement

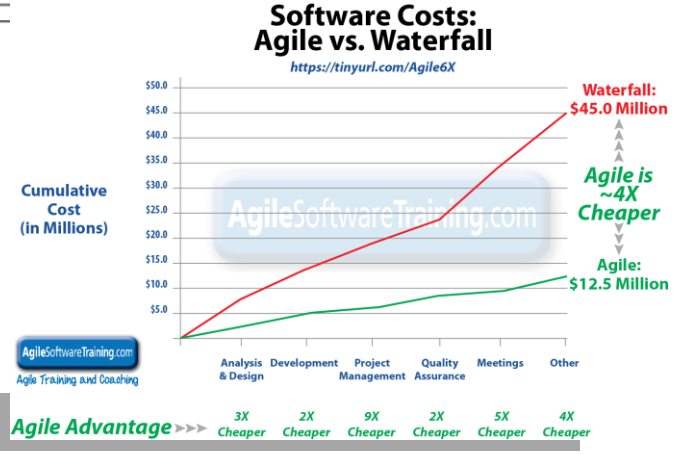

https://agilemanifesto.org/

# Agile Helps, but Which Agile?

- Research studies show the value of agile, but not *which* agile
  - https://www.researchgate.net/publication/249011841_Comparative_Study_on_Agile_software_development_methodologies
- Many agile organizations have data, though each one tends to favor their approach
  - https://www.scrum.org/
  - https://www.scruminc.com/
  - https://www.scaledagileframework.com/team-and-technical-agility/



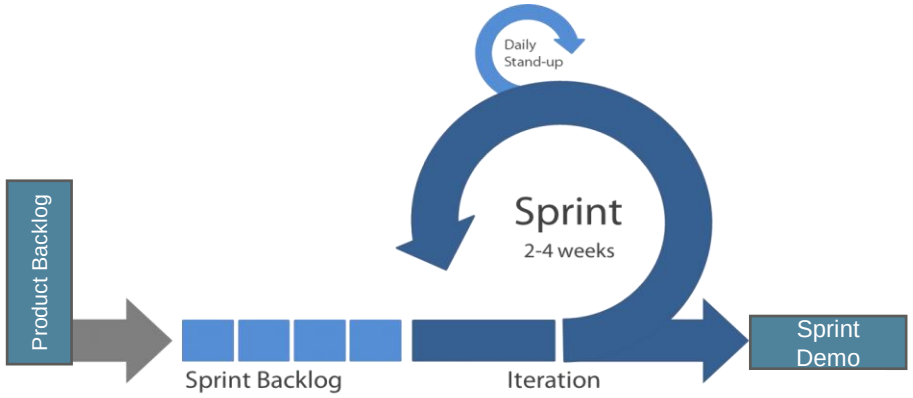




**Agile improves responsiveness**

# Agile Types

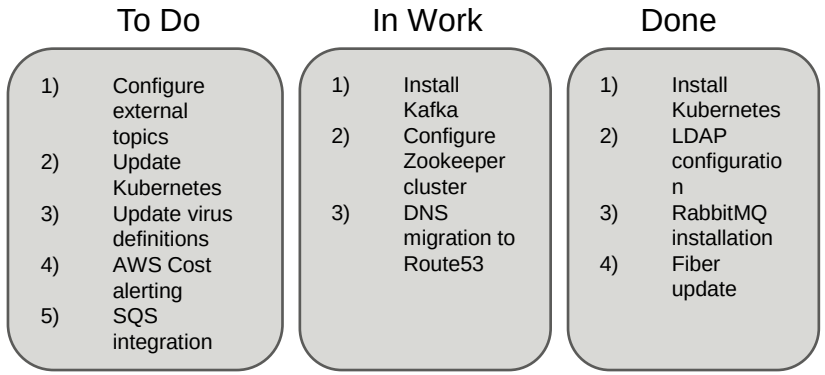
## Scrum

- Short cycle time (1-4 weeks)
- Every sprint is potentially releasable
- Incrementally improving capabilities
- Demonstrable progress
- Customer is engaged in the prioritization
- Regular backlog grooming



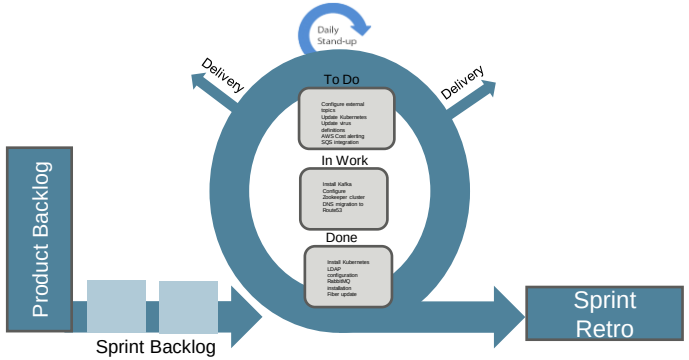

## Kanban

- Limited Work In Progress (WIP)
- Constantly prioritized backlog
- No ‘end point’ because it is continual progress
- Stories are independent
- Team members pick up the next priority item
- Each story is demonstrated upon completion



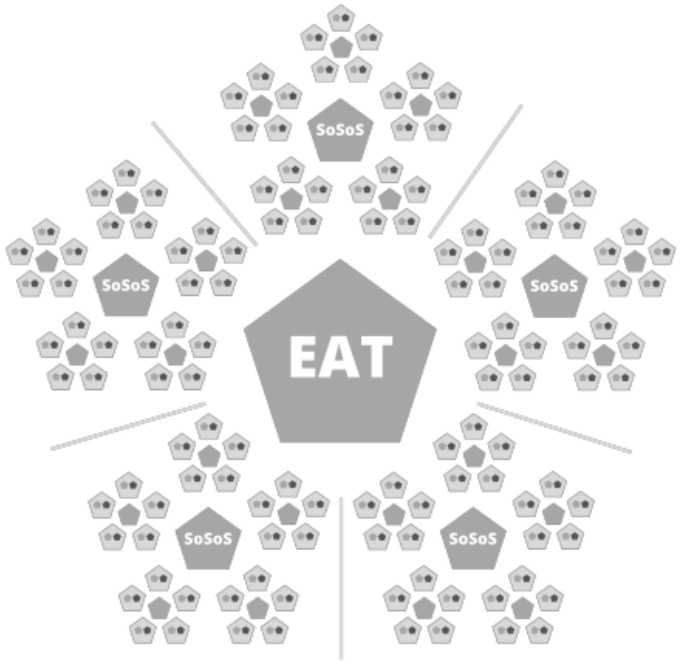

## Scrumban

- Short cycle times
- Limited Work In Progress (WIP)
- Prioritized backlog every sprint
- Team members pick up the next priority item within sprint plan
- Each story is demonstrated upon completion
- Capabilities delivered continually



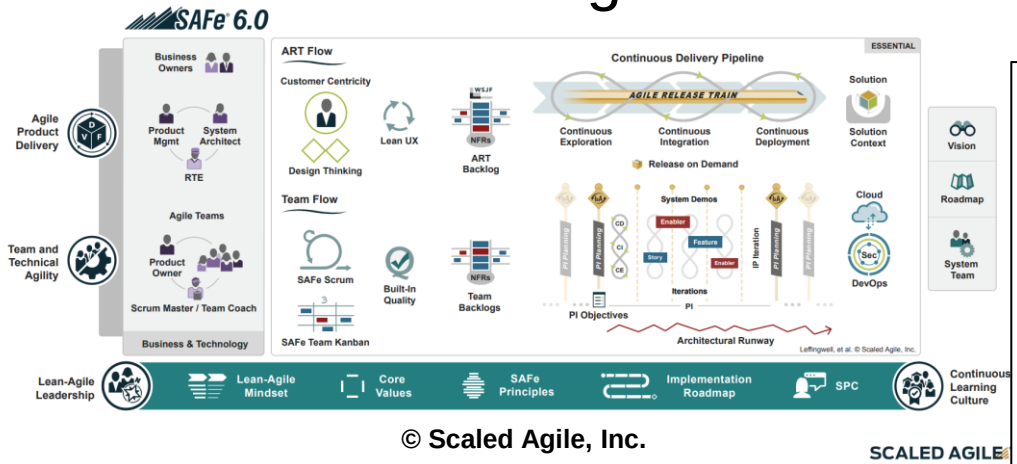
# Scaled Agile Approaches

- Scrum at Scale (S@S)



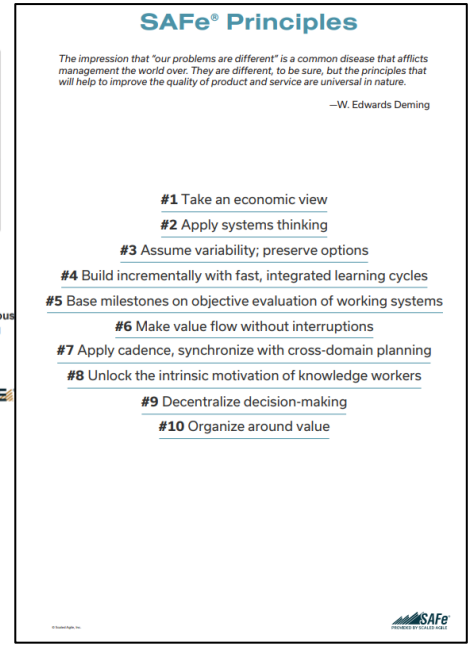

official-scrum-at-scale-guide.pdf (scrumatscale.com)

- Scaled Agile Framework (SAFe)




© Scaled Agile, Inc.

SCALED AGILE




SAFe 6.0 (scaledagileframework.com)

**So many choices, how do we decide?**

# Program Startup

- What's Important?
  - Understanding what the work is
  - Knowing what is important when
  - Knowing the risks
  - Finding out who the stakeholders are
  - Determining what 'good' looks like
  - Finding our Minimal Viable Product
  - Getting people on the same page
  - Finding the unknown-unknowns
  - Developing the staff
- What does that tells us?
  - Backlog content needs to be established
  - Prototyping to refine the value
  - Experimentation to buy down risks
  - Establishing strong communication channels
  - Defining relative importance of capabilities
  - Letting teams storm and form
  - Getting all perspectives on the table
  - Finding skill and knowledge gaps early

**Getting off on the right foot can never be overlooked**

# Program Startup – Agile Selection

## Scrum

- Backlog content needs to be established
- Prototypin
- Experime risks
- Establishing strong communication channels
- Defining relative importance of capabilities
- Letting teams storm and form
- Getting all perspectives on the table
- Finding sk gaps early

Scrum works small portions of the backlog rather than performing a full backlog elaboration

Backlog grooming tends to be shorter sessions, populating enough for a few sprints

## Kanban and Scrumban

- Backlog content needs to be established
- Prototyping to
- Experimentation to buy down risks
- Establishing strong communication channels
- Defining relative i capabilities
- Letting teams storm and form
- Getting all perspectives on the table
- Finding skill and knowledge gaps early

Kanban is about working off a prioritized backlog, not creating an initial backlog

Fewer ceremonies and external customer collaborations

More about velocity and closure rates than initial information gathering

## Scaled

- nt needs to be
- Prototyping to refine the value
- Experimentation to buy down risks
- Establish communication channels
- Defining rel capabilities
- Letting teams storm and form
- Getting a table
- Finding skill and knowledge gaps early

Increments/iterations are longer duration, so prototypes don't get immediate feedback

Experimentation can be planned in, but may not result in immediate action

PI objectives with too many unknown unknowns is hard

We may not know the skills we need yet

**Initial formation and execution will have bumps**

# Early Program Execution

- What's Important?
  - Uncertainties exist about the system and the context
  - Risks that need to be bought down as well as identifying new risks
  - Experimentation and prototyping heavily encouraged
  - Learning lessons about system and operational use
  - Teams forming and storming so they may not have the optimal velocity
- What does that tells us?
  - Experimentation is needed to fail fast and find the best path
  - Prototyping will be useful to get everyone on the same page
  - Definition of Done may be changing as lessons are learned
  - Team composition and cadence may need refining
  - Changing directions periodically will help incorporate changes

**Developing concept, learning constraints, and getting aligned**

# Early Program Execution – Agile Selection

## Scrum

- Experimentation is needed to fail fast and find the best path
- Prototyping will be useful to get everyone on the same page
- Definition of Done may be changing as lessons learned
- Team composition and cadence may need refining
- Changing directions periodically will help incorporate changes

Changing teams impacts velocity measures

Changes limited to sprint boundaries

## Kanban and Scrumban

- Experimentation is needed to fail fast and find the best path
- Prototyping will be useful to get everyone on the same page
- Definition of Done may be changing as lessons learned
- Team composition and cadence may need refining
- Changing directions periodically will help incorporate changes

Teams not expected to be static so nothing to 'improve'

## Scaled

- Experimentation is needed to fail fast and find the best path
- Prototyping will be useful to get everyone on page
- Definition of Done may be changing as lessons are learned
- Team composition and cadence may need refining
- Changing directions periodically will help incorporate changes

Experimentation not aligned with PI planning

Changing DoD undermines confidence vote

Changing direction undermines increment plan

**Level of change and uncertainty still requires short term planning**

# Core Development

- What's Important?
  - Teams norming
  - Backlog is maturing, but not complete
  - Major risks reduced but some still coming up
  - Teams haven't yet hit 'performing' phase with maximum velocity
  - Beginning to look at system integration
  - Need to minimize defects introduced during this phase
  - Identifying defects early to reduce correction costs and schedule impact
- What does that tells us?
  - Established cadence and expectations
  - Fewer in-cycle changes required
  - More predictability in team production
  - Team communication has stabilized
  - Stakeholders have visibility into priorities of the teams and programs
  - More time spent on early integration and preventing defect escapes
  - Deployment/promotion approach is mature

**Getting into a cadence helps maintain pace longer term**

# Core Development – Agile Selection

## Scrum

- Established cadence and expectations
- Fewer in-cycle changes required
- More predictable productivity
- Team communication has stabilized
- Stakeholders have visibility into priorities of the teams and programs
- More time sp… integration and preventing defect escapes
- Deployment/promotion approach is mature

Primarily short-term visibility

## Kanban and Scrumban

- Established cadence and expectations
- Fewer in-cycle changes rec…
- More predictable productivity
- Team communication has stabilized
- Stakeholders … priorities of th… programs
- More time spent on early integration and preventing defect escapes
- Deployment/promotion approach is mature

Cadence is WIP limit, no expectation of feature completion

Velocity yes, productivity on capabilities less so

Team communications are tight loop, not strategic

## Scaled

- Established cadence and …ectations
- …er in-cycle changes required
- More predictable productivity
- …m communication has …ilized
- Stakeholders have visibility into priorities of the teams and programs
- More time spent on early integration and preventing defect escapes
- Deployment/promotion approach is mature

**Less uncertainty and more predictability lend themselves to more structure**

# Integration and Test

- What's Important?
  - Shift to test focus
  - Rapid turnaround of defects and change requests
  - Disconnects and design gaps become readily apparent
  - Unable to wait weeks for fixes (peeling the onion slowly)
  - Needing to report progress and status regularly on a rapid cycle
- What does that tells us?
  - Cycle time needs to be short and decreasing
  - Team priorities will change rapidly based on test results
  - Capability gaps will need short term solutions to hold schedule
  - Team communication will need to be rapid and effective
  - Internal and external visibility will be important

**Accelerating to the finish line**

# Integration and Test – Agile Selection

## Scrum

- Cycle time needs to be short and decreasing
- Team priorities will change rapidly based on test results
- Capability gaps will need short term solutions to hold schedule
- Team communication will need to be rapid and effective
- Internal and external visibility will be important

Only at sprint boundaries

External visibility in backlog grooming and sprint demos

## Kanban and Scrumban

- Cycle time needs to be short and decreasing
- Team priorities will change rapidly based on test results
- Capability gaps will need short term solutions to hold schedule
- Team communication will need to be rapid and effective
- Internal and external visibility will be important

Kanban might struggle with Definition of Done

## Scaled

- Cycle time needs to be short and decreasing
- Team priorities will change rapidly based on test results
- Capability gaps will need short term solutions to hold schedule
- Team communication will need to be rapid and effective
- Internal and external visibility will be important

Times are defined in process

Increment priorities are fairly firm

External visibility mainly on boundaries

**Agility on shorter timelines for rapid adjustment**

# System Completion and Delivery

- What's Important?
  - Moving from development to sustainment
  - Increasing rigor on reviews and change control
  - Less agile innovation because the system is stabilized
  - Experimentation is on the side for future integration (not time critical)
  - Potential for rapid interruptions to support operations (less predictability in plan execution)
- What does that tells us?
  - Less of a set backlog of development activities to prioritize
  - Capabilities mostly complete
  - Operational defects may be sporadic and high priority
  - Fewer needs for prototyping because operational need is solidified
  - Additional maintenance activities on a regular cadence (e.g. patching)
  - Standard customer communication cadence

**Moving to sustainment of operations instead of 'development'**

# System Completion – Agile Selection

## Scrum

- Less of a set backlog of development activities to prioritize
- Capabilities mostly complete
- Operational defects may be sporadic and high priority
- Fewer needs for prototyping because operational need is solidified
- Additional maintenance activities on a regular cadence (e.g. patching)
- Standard customer communication cadence

Sprint interruptions challenge velocity

Innovation cycles not needed

## Kanban and Scrumban

- Less of a set backlog of development activities to prioritize
- Capabilities mostly complete
- Operational defects may be sporadic and high priority
- Fewer needs for prototyping because operational need is solidified
- Additional maintenance activities on a regular cadence (e.g. patching)
- Standard customer communication cadence

Large planning events lower value

Increment objectives harder to determine

Interruptions would be lost capacity

Kanban and Scrumban are WIP based, not time based

## Scaled

- Less of a set backlog of development activities to prioritize
- Capabilities mostly complete
- Operational defects may be sporadic and high priority
- Fewer needs for prototyping because operational need is solidified
- Additional maintenance activities on a regular cadence (e.g. patching)
- Standard customer communication cadence

**Combining cadence and uncertain priority injection**

# Decision Matrix – Which Agile is Right When?

- These are general decision criteria
  - Easiest to rule out methodologies that don't fit well

| | Establishing Backlog | Uncertainty | Experimenting | Status Reporting | Rapid Adjustments | Established Teams | Skill Gaps | Predictable Cadence | Rate of Change | Integration Activities | Cycle Time | Intra-Team Communication |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Scrum** | ✓ | Medium | ✓ | ✓ | | ✓ | ✓ | ✓ | Medium | ✓ | 2-4 Week | Periodic |
| **Kanban** | | High | ✓ | | ✓ | | | | High | | Immediate | Rare |
| **Scrumban** | | High | ✓ | | ✓ | | ✓ | | High | | 2 Week | Periodic |
| **Scaled** | ✓ | Low | | ✓ | | ✓ | | ✓ | Low | ✓ | 4 Week+ | Weekly |

**Needs change, so should your agility**

# Decision Time

- Program Startup
  - Short sprint duration (2 weeks or less) Scrum supports early refinement of plan
- Early Program
  - Scrumban may provide benefits for rapid prototyping and risk reduction
- Core Development
  - Scaled approach if size is large, otherwise Scrum
- Integration and Test
  - Kanban allows rapid turnaround of critical defects from integration
- System Completion and Delivery
  - Scrumban provides a cadence for normalcy while allowing rapid adjustments

**Every situation is unique, but Agile is about adapting and improving, not locking into one method**

# About the Author

## Kurt Mohr

Kurt Mohr is a Raytheon Senior Engineering Fellow with over 20 years in Software Development, Systems Engineering and Architecture spanning both commercial and multiple defense contractors.  He has a strong background in multiple Agile development methodologies including Scrum, Kanban, and Scaled Agile.  Kurt is a Software Technical Director for Raytheon Engineering supporting program execution, independent reviews, productivity improvements, and architecture training and development.  Kurt has architectural experience with multiple forms of software systems including cloud, bare metal, and embedded as well as highly scalable distributed computing.