



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

Being Agile in Agile



What is "Agile" within this Context?

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools Working software over comprehensive documentation Customer collaboration over contract negotiation Responding to change over following a plan

> That is, while there is value in the items on the right, we value the items on the left more.

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

Ravtheon

Principles behind the Agile Manifesto

We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity--the art of maximizing the amount of work not done--is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

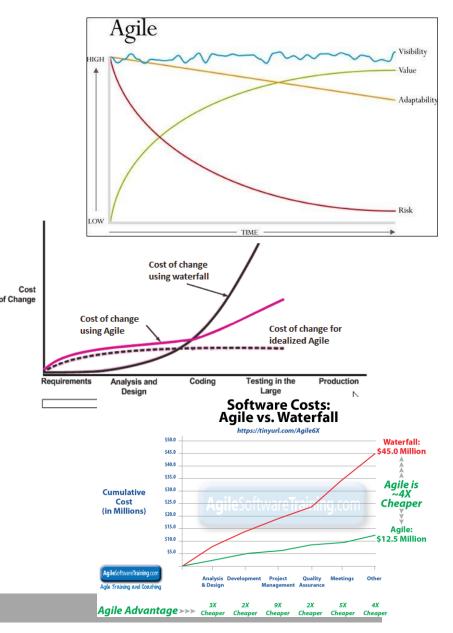
https://agilemanifesto.org/

Raytheon Company - Approved for Public Disclosure

This document does not contain technology or technical data controlled under either the U.S. International Traffic in Arms Regulations or the U.S. Export Administration Regulations.

Agile Helps, but Which Agile?

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



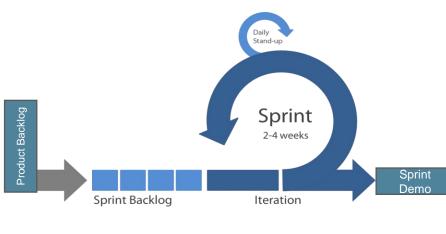


Agile improves responsiveness

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

To Do

Configure

external

topics

Update

Kubernetes

definitions

AWS Cost

integration

alerting

SQS

Update virus

1)

2)

3)

4)

5)

- Team members pick up the next priority item
- Each story is demonstrated upon completion

1)

2)

3)

In Work

Install

Kafka

cluster

DNS

Configure

Zookeeper

migration to

Route53

Done

Install

LDAP

Fiber

update

Kubernetes

configuratio

RabbitMQ

installation

1)

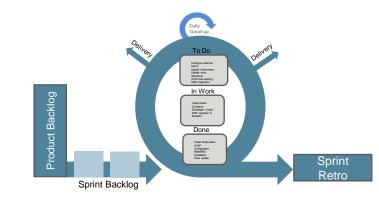
2)

3)

4)

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

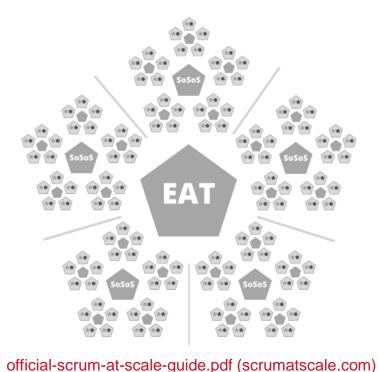


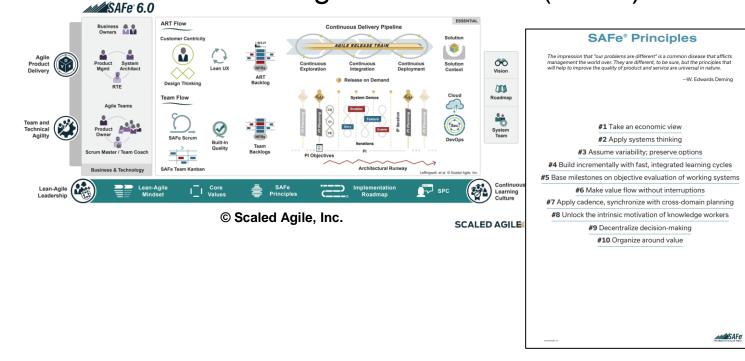


This document does not contain technology or technical data controlled under either the U.S. International Traffic in Arms Regulations or the U.S. Export Administration Regulations.

Scaled Agile Approaches

Scrum at Scale (S@S)





• Scaled Agile Framework (SAFe)

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 Scrum works small portions of the backlog rather than performing a Experime full backlog elaboration risks
- Establishing strong communication channels
- Defining relative importance of capabilities
- Letting teams storm and form
- Getting all perspectives on the table

shorter sessions, populating

enough for a few sprints

Backlog grooming tends to be Finding sl gaps early

Kanban and Scrumban

 Backlog content needs to be established ^{*}

Kanban is about working off a Prototyping to prioritized backlog, not creating an initial backlog Experimentatio risks

- Establishing strong communication_channels
- Defining relative ir capabilities

Fewer ceremonies and external customer collaborations

information gathering

- Letting teams storm and form
- Getting all perspectives on the table
- Finding skill and knowledge More about velocity and gaps early closure rates than initial

nt needs to be longer duration, so prototypes don't get immediate feedback

- Prototyping to refine the value
- Experimentation to buy down
- risks

Scaled

Increments/iterations are

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

- Defining rel capabilities
 - PI objectives with too many unknown unknowns is hard
- Letting teams storm and form
- Getting a We may not know the skills we need yet table
- Finding skill and knowledge gaps early

Raytheon Company - Approved for Public Disclosure

Initial formation and execution will have bumps

This document does not contain technology or technical data controlled under either the U.S. International Traffic in Arms Regulations or the U.S. Export Administration Regulations.

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 les learned

Changing teams impacts velocity measures

- Team composition and cadence may need refining
- Changing directions periodically will help incorporate changes

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 Teams not expected to be static so nothing to learned
- Team composition and cadence may need refining
- Changing directions periodically will help incorporate changes

Scaled

'improve'

Experimentation not aligned with PI planning

- Experimentation is needed to fail fast and find the best path
- Prototyping will be useful to get everyone on page

Changing DoD undermines confidence vote

- Definition of Done may be changing as lessons are learned
- Team composition and cadence may need refining
- Changing directions periodically will help incorporate changes

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

Kanban and Scrumban

- Established cadence and expectations
- Fewer in-cycle changes rec
- More predictable productivity
- Team communication has stabilized
- Stakeholders priorities of the programs
 Team communications are tight loop, not strategic
- More time spent on early integration and preventing defect escapes
- Deployment/promotion approach is mature

Scaled

completion

Velocity yes, productivity

on capabilities less so

- Established cadence and
- Cadence is WIP limit, no ctations
 - er in-cycle changes required
 - More predictable productivity

n communication has

- 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 Kanban and Scrumban Scaled Times are defined in process Cycle time needs Cycle time needs to be short Cvcle time page Only at sprint and decreasing and decreasing and decreasing boundaries - Team priorities will shange - Team priorities Team priorities will change Kanban might struggle with rapidly based o rapidly based on rapidly based on test results t results **Definition of Done** Capability gaps will need Capability gaps will need - Capability gaps Increment priorities short term solutions to hold short term solutions to hold short term soluti are fairly firm schedule schedule schedule Team communication will Team communication will Team communication will _ External visibility in need to be rapid ar need to be rapid and e rapid and External visibility backlog grooming and mainly on boundaries effective effective sprint demos - Internal and external visibility Internal and external visibility Internal and external visibility will be important will be important will be important

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
 Sprint interru challenge versionen
- Capabilities mostly compared
- Operational defects may be sporadic and high priority
- Fewer needs for prototyping because operational need is solidified

 Additional main needed activities on a regular cadence (e.g. patching)

 Standard customer communication cadence

Ka	nban and Scrumban	Scaled
_	 Less of a set backlog development activities 	 Less of a set backlog of development activities to
ptions elocity	prioritize Capabilities mostly cor harder to determine	prioritize Capabilities mostly complete
_	 Operational defects may be sporadic and high priorit Interruptions would 	 Operational defects may be
	- Fewer needs for prototyping because operations Kanban and Scrumban are WIP based, not time based	Fewer needs for prototyping because operational need is solidified
t _	Additional mainter ince activities on a regular cadence (e.g. patching)	 Additional maintenance activities on a regular cadence (e.g. patching)
-	 Standard customer communication cadence 	 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 Bac	Uncertainth Hog	Speriment.	Statis Reporting	Repieracijustine	¹⁵ ¹² 11 ¹⁵ ¹⁶⁴ ¹⁶⁴ ¹⁶⁴¹¹	Stall Caps	Redictable Caten	Rate of Charge	PITES STORACINITIC	Gole Time	Intra Ream Commu	Incation
Scrum	\checkmark	Medium	✓	\checkmark		\checkmark	\checkmark	\checkmark	Medium	\checkmark	2-4 Week	Periodic
Kanban		High	\checkmark		\checkmark				High		Immediate	Rare
Scrumban		High	\checkmark		\checkmark		\checkmark		High		2 Week	Periodic
Scaled	\checkmark	Low		\checkmark		\checkmark		\checkmark	Low	\checkmark	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.

