



**33<sup>rd</sup>** Annual **INCOSE**  
international symposium

hybrid event

Honolulu, HI, USA  
July 15 - 20, 2023



# Contrasting and Comparing Agile Systems Engineering and Agile Software Engineering

**IS23 Panel**

**Monday 17-July, 15:30-17:00**

Rick Dove [dove@parshift.com](mailto:dove@parshift.com)  
Kerry Lunney [kerry.lunney@thalesgroup.com.au](mailto:kerry.lunney@thalesgroup.com.au)  
Duncan Kemp [duncan.kemp735@mod.gov.uk](mailto:duncan.kemp735@mod.gov.uk)  
Robin Yeman [robinyeman@gmail.com](mailto:robinyeman@gmail.com)

**Download this file: [www.parshift.com/t/p1.pdf](http://www.parshift.com/t/p1.pdf)**

# Panel Context

**Agile engineering, of any kind, is a principle-based method for designing, building, sustaining, and evolving purpose-fulfilling creations when knowledge is uncertain and operational environments are dynamic.**

**Principles are abstractions for what needs to be accomplished and why, without constraints or directions on how.**

**How those abstractions manifest operationally depends upon the engineering context.**

**Single-domain software engineering is different than multi-domain systems engineering.**

**While tactical methods necessarily vary among different engineering domains (the how part), strategies for achieving common goals (the what and why parts) are domain independent.**

# Panel

**Panelists will offer their views on selected agility-enabling strategies with considerations for fitting them differently in Agile Systems Engineering and Agile Software Engineering.**

## **Panel Purpose:**

- **Mitigate confused thinking that agile software methods are appropriate for agile Systems Engineering.**
- **Show how SE and SW have a different perspective on methods**

# 90 Minute Plan

**3 mins: Context**

**44 mins: Four different points of view**

**3 mins: One-slide summary**

**40 mins: Audience questions, comments, contributions**

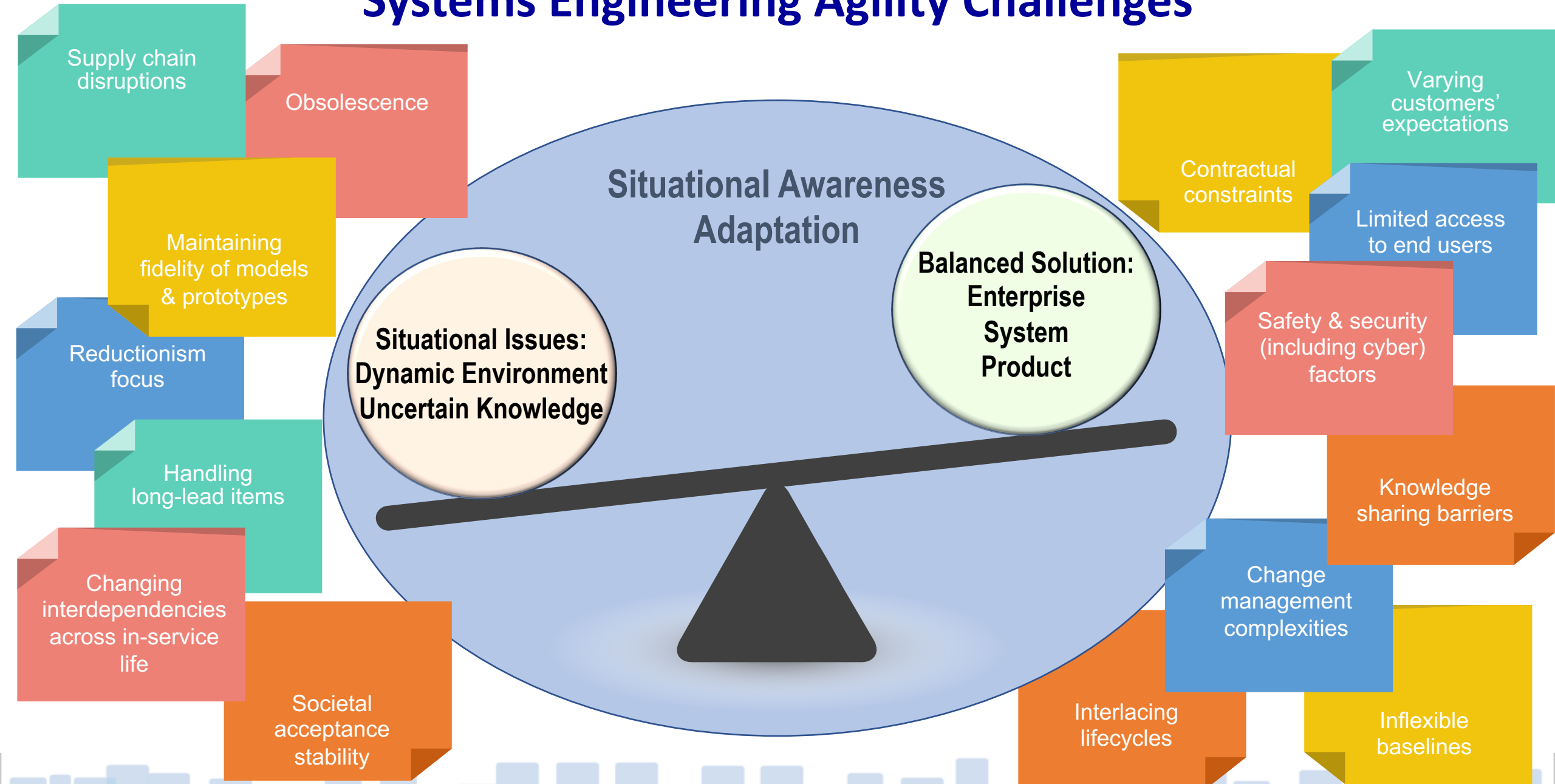


## **Agility is Essential for Situational Awareness Adaptation**

**Kerry Lunney**

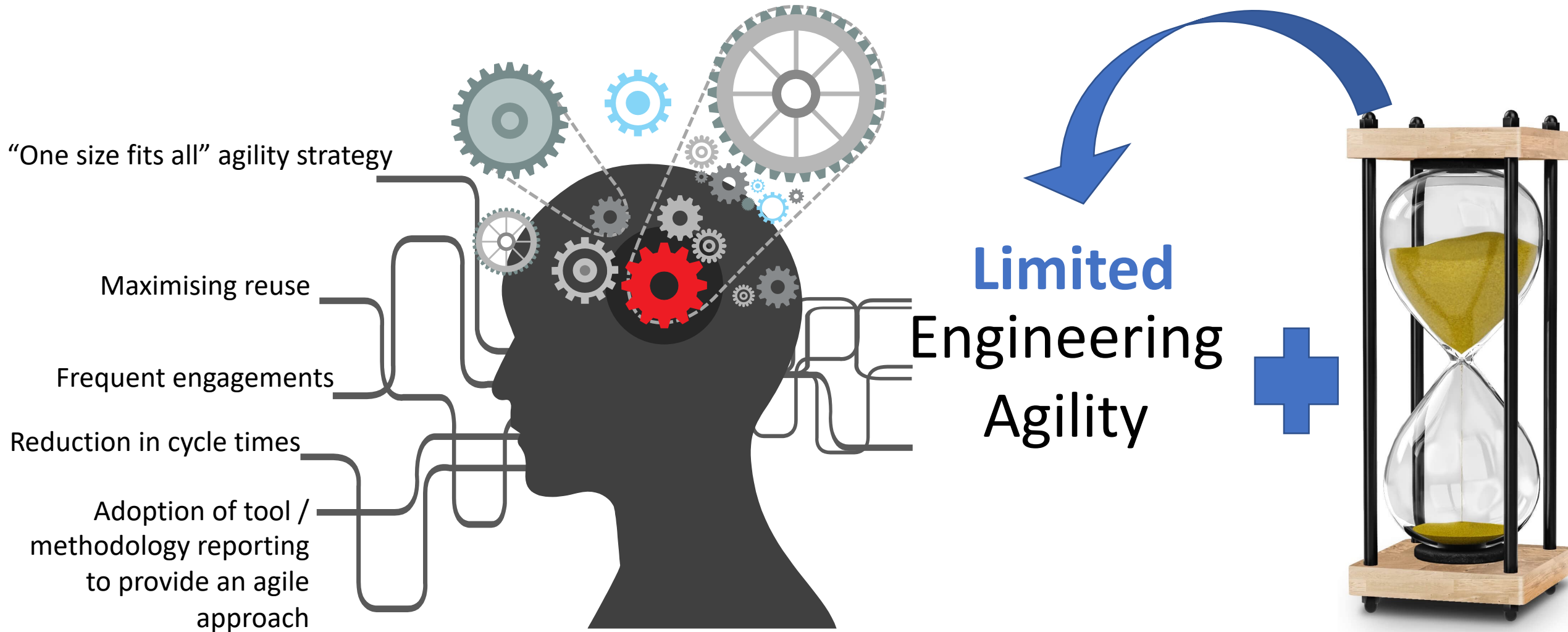
**Thales Australia Country Engineering Director / Chief Engineer**

# Systems Engineering Agility Challenges





# Misconceptions Leading to Limited Engineering Agility



Temporal considerations are key to being agile

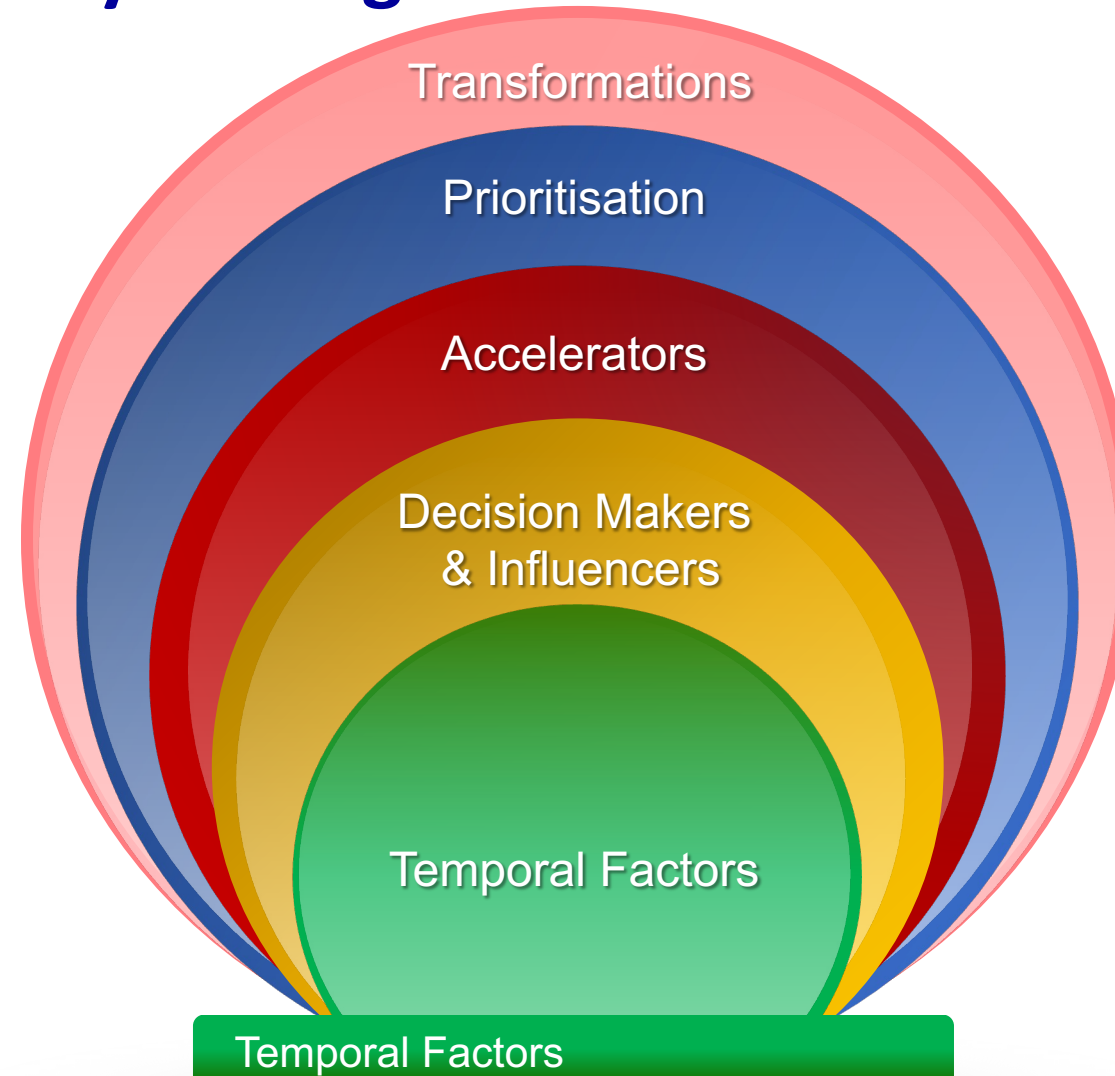
# Engineering Agility Through Situational Awareness Adaptation

## Accelerators

Rapid catalysts for adaptations –  
Minimal Viable Product/Capability (MVP/C) strategies / Mental models / Knowledge sharing / Tailoring / “Safe” labs & environments / Flexible Chain of Command / End user accessibility / Pilot strategies / Protected core features / Frequent engagements / Mission thread preservation

## Decision Makers & Influencers

- Over-arching governance
- Decision making empowerment to those closer to the problem/activity
- Multiple levels of delegation & responsibilities
- Re-balance customer expectations



## Temporal Factors

- Time is not constant, but relative
- Situational awareness changes with time
- Adaptations need to be fluid but controlled

## Transformations

Enabling adaptations –  
Digital transformations / MVP/MVC / Models, prototypes, digital twinning / Agile tools & methodologies / Supply chain optimisation / Feedback loops coupled with decision authority / Aids for test & recovery / Reuse & variability / Robust change management

## Prioritisation

- Common shared vision
- Re-alignment of goals
- Playbook
- Enabling pathway from accelerators to transformations



# Take aways

**Engineering Agility is required for situational awareness adaptations**

**The dynamics of time constantly impact agility approaches**

**Temporal Factors + Decision Makers/Influencers + Accelerators +  
Prioritisation + Transformations → ↑ Engineering Agility**

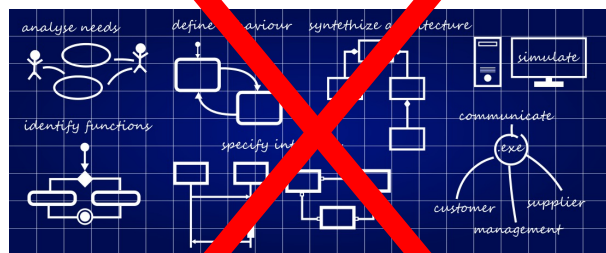
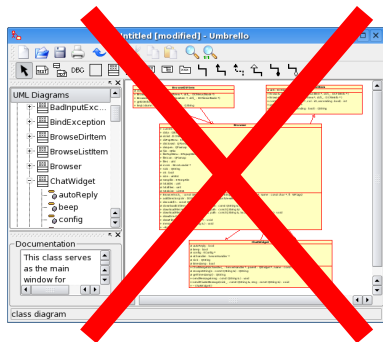
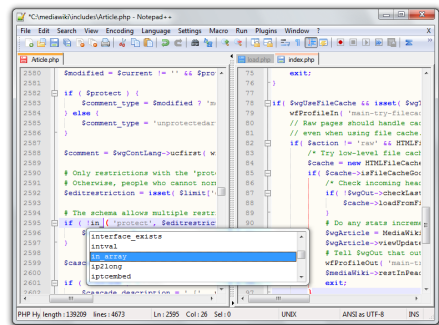


# **Professor Duncan Kemp, CEng, FIET, INCOSE Fellow**

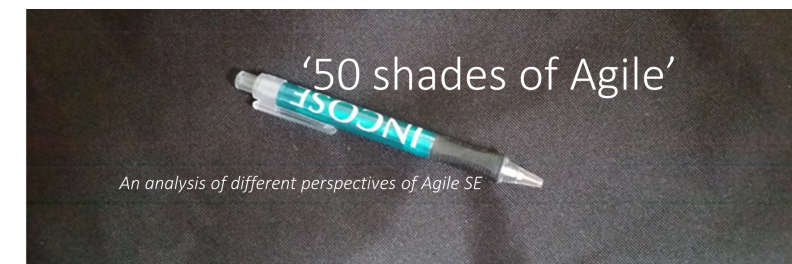
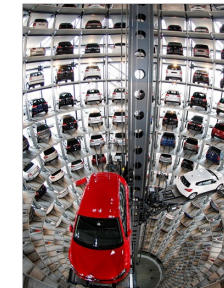
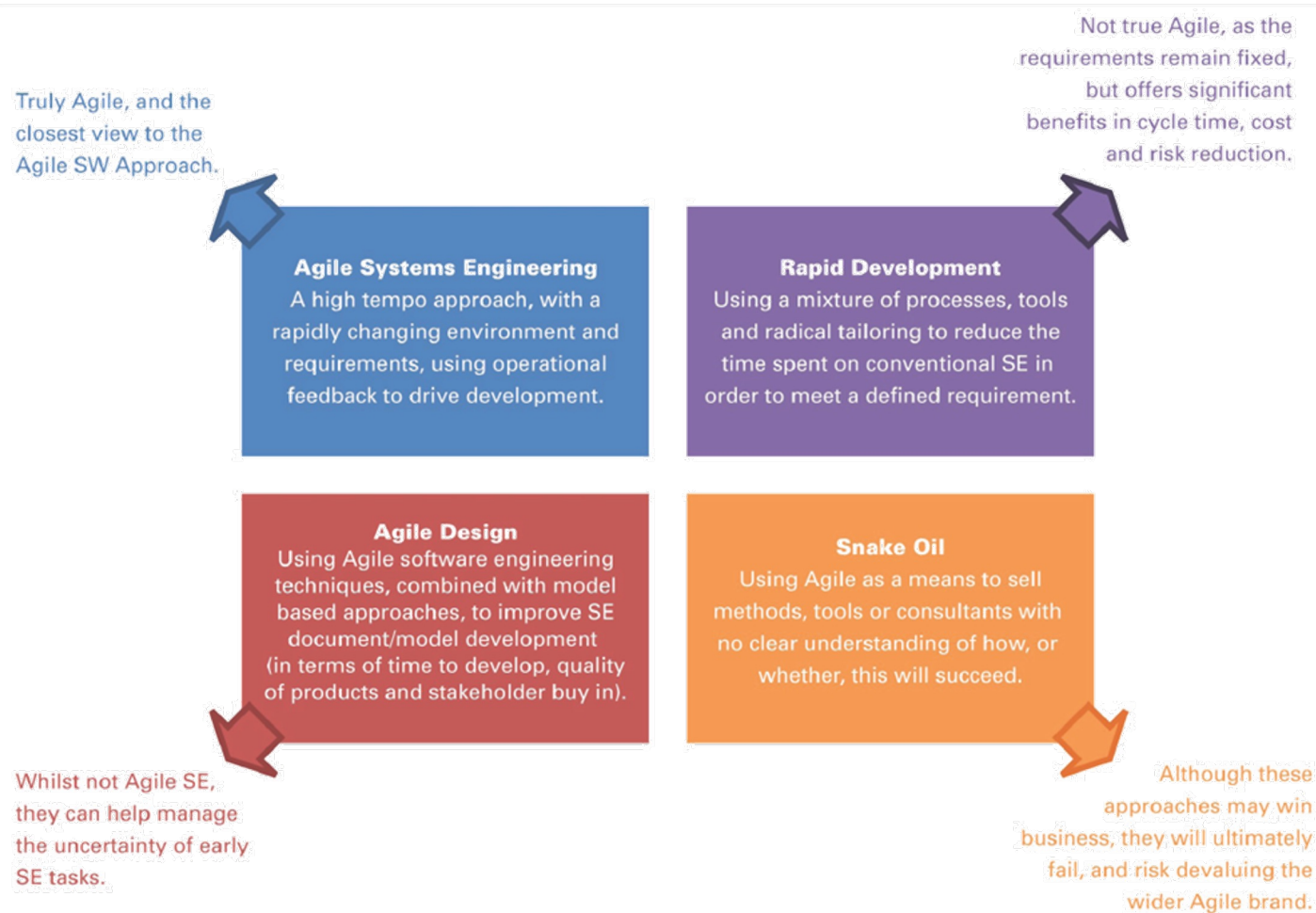
## **UK Ministry of Defence**

---

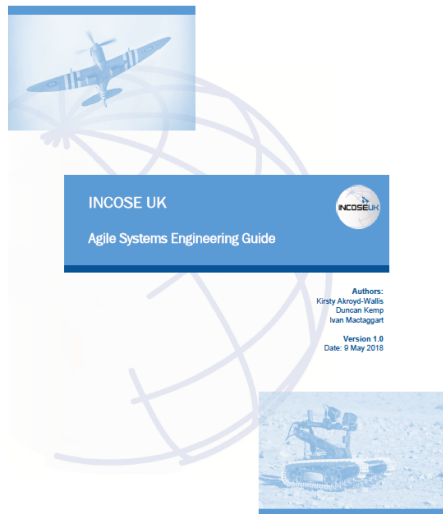
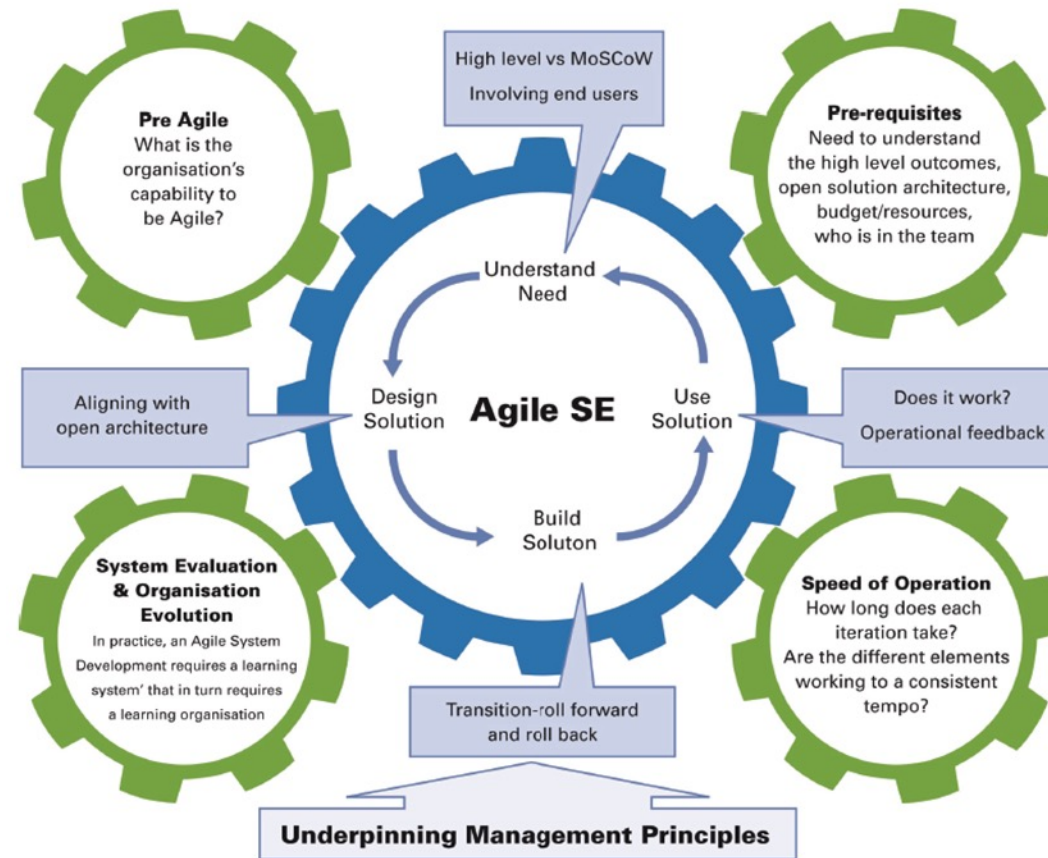
# A long time ago in a galaxy far-far away



# Agile Systems Engineering



# Agile Systems Engineering



# Take aways

**There is a lot of confusion as to what Agile Systems Engineering is**

**Vanilla Agile Software Engineering techniques will not work on Agile Systems**

**But it is possible, and necessary, to increase the agility of SE**



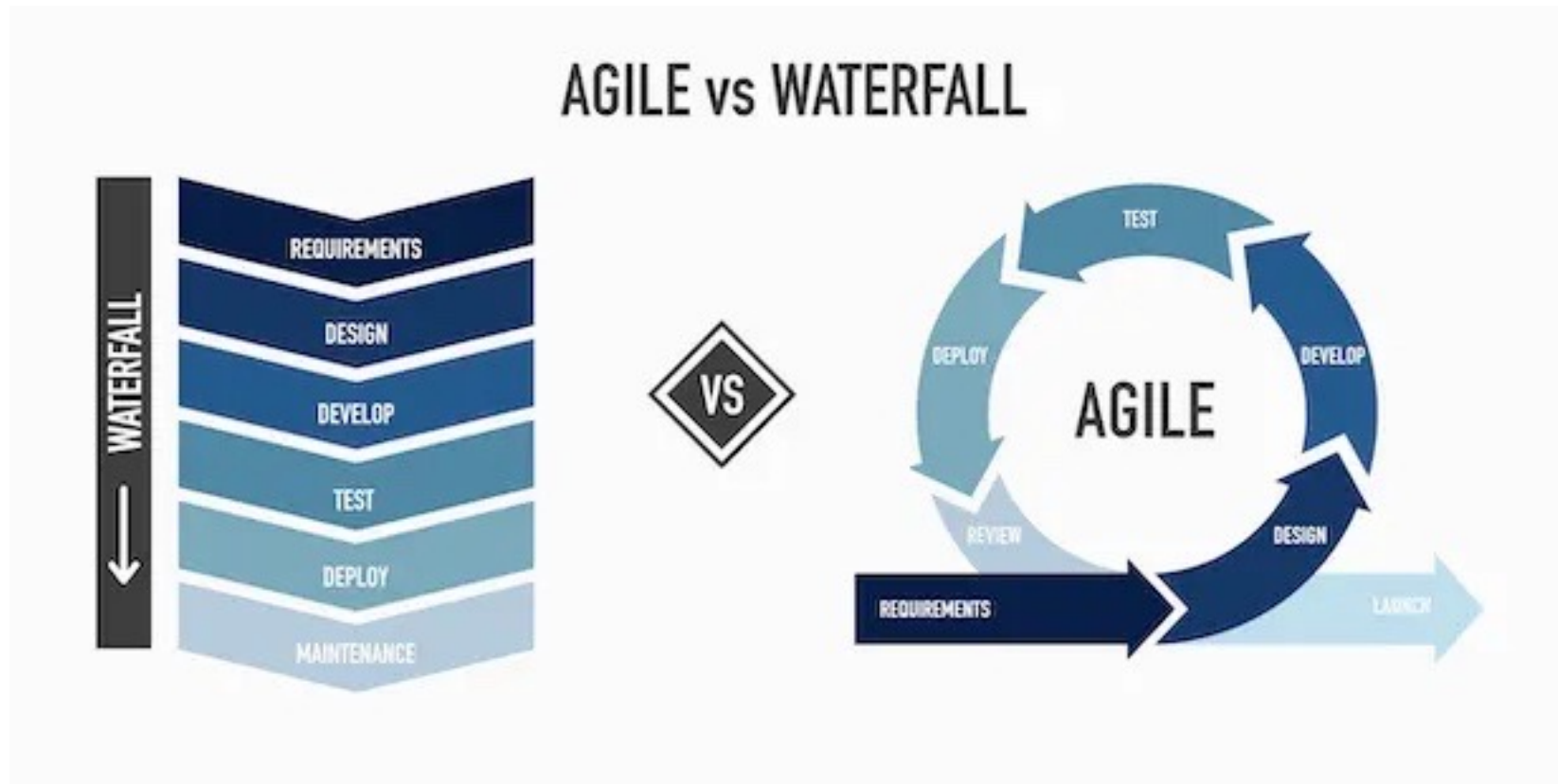


**Robin Yeman, SAFe Fellow / Systems Engineering PHD Candidate**

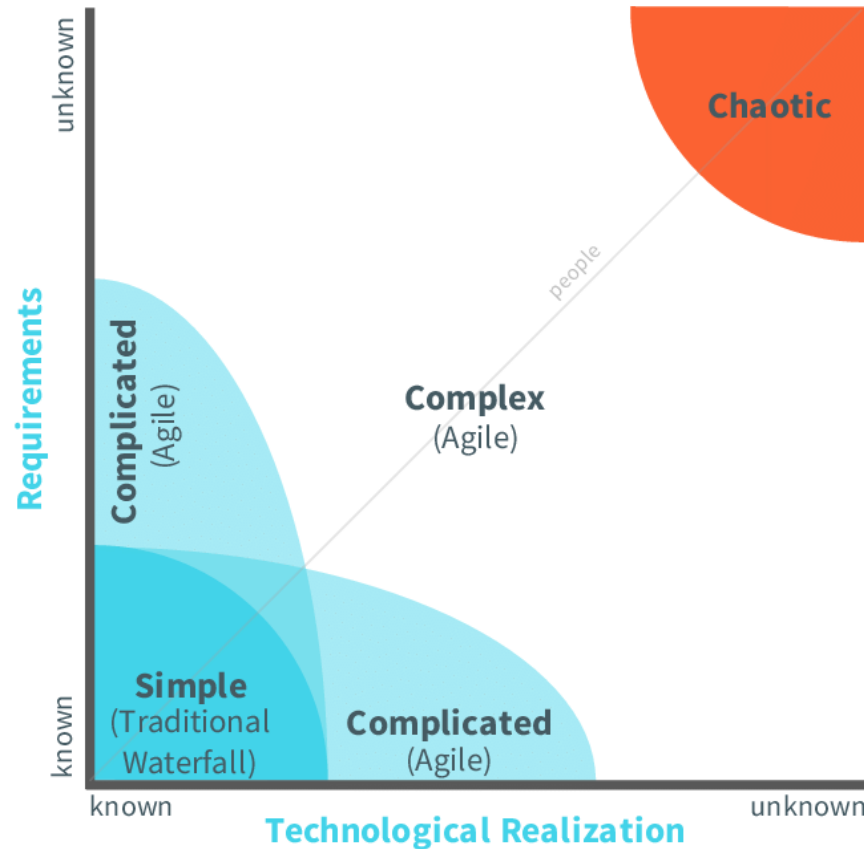
# **Agile is an empirical lifecycle**

# Agile is a Lifecycle

Waterfall is a predictive lifecycle based on phase gates, Agile is an empirical lifecycle based on objective data.

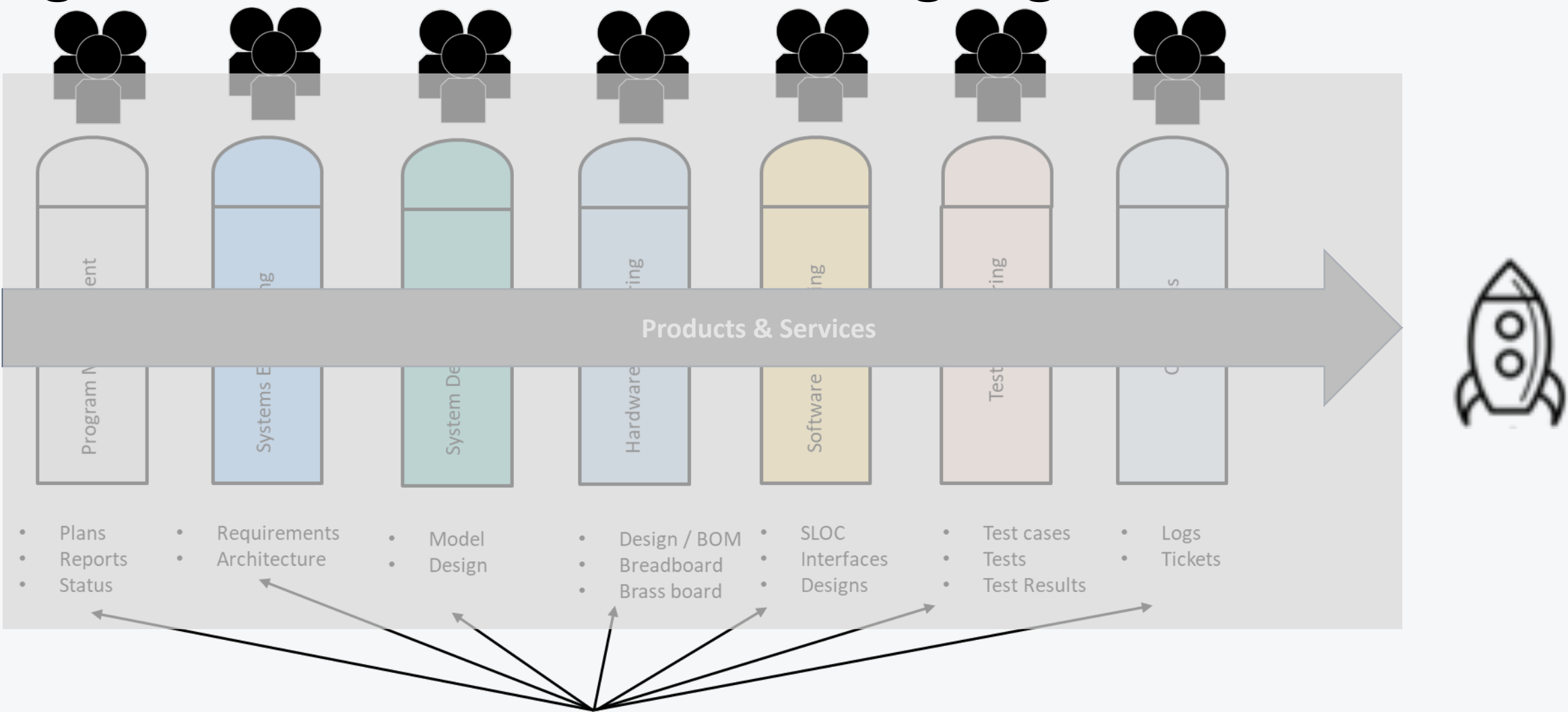


# Begin with Why

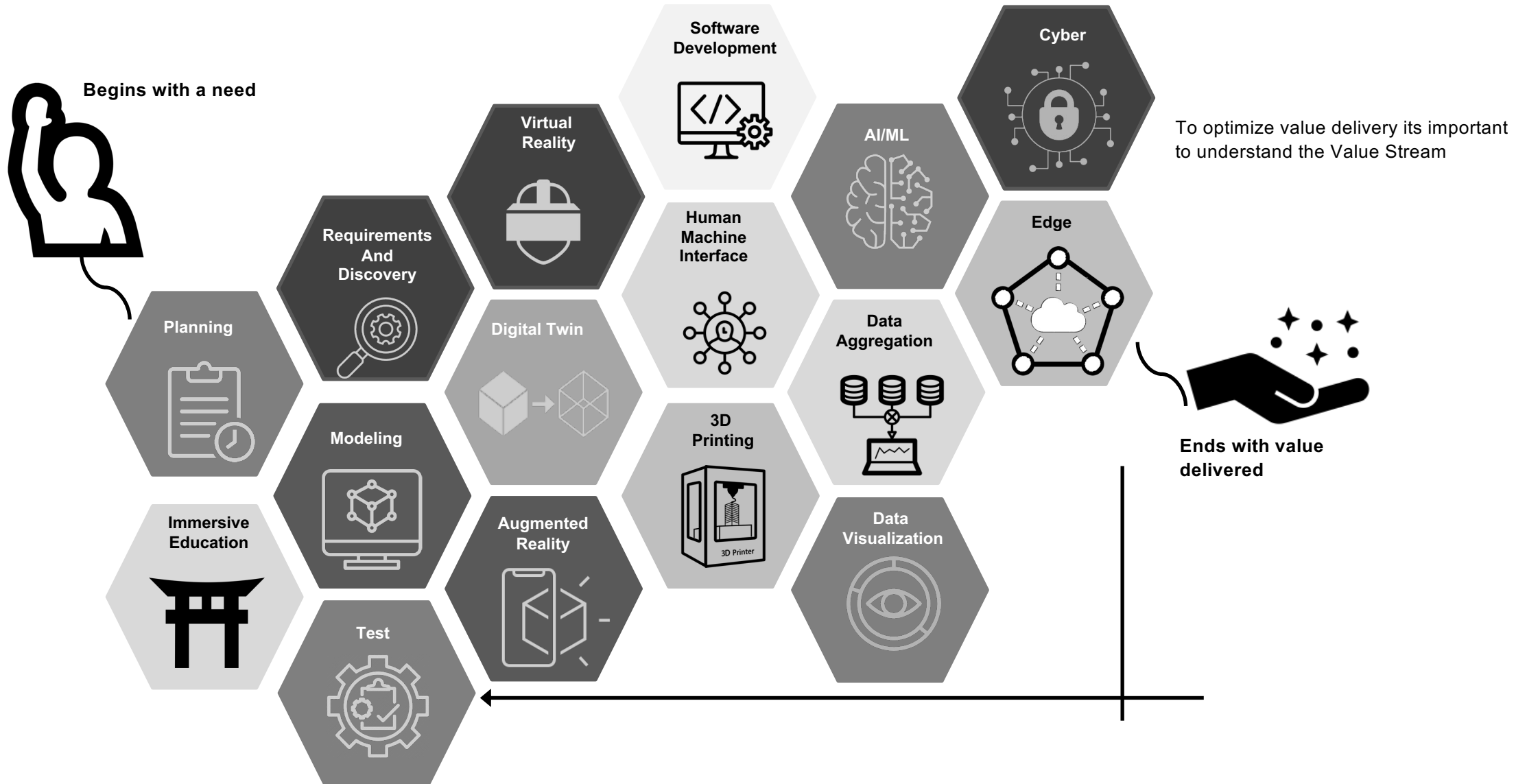


*The motivation to migrate to Agile is a demand for **faster** development, difficulty managing **change**, and increased product **complexity***

# Organization and Common Language



# Use all the tools in Value Stream



# Agile Engineering



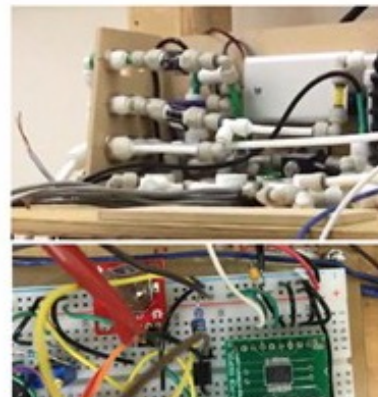
## Plywood Prototype

- Wooden support frame
- Power-on
- Installation features
- Time-based filter replacement



## Hydraulic Prototype

- Functional hydraulics
- In-line sensors installed
- Read sensors
- Flow-based filter expiration



## "Alpha" Near-Final Prototype

- 'Rev 1' hardware
  - PCBA
  - Sheet metal
  - Cables
- Internet connection
- Logging
- Secondary GUI screens



## "Beta" Fully-Functional Verification Ready

- 'Rev A' hardware
- Service screens
- Reliability improvements
- Final GUI improvements
- Bug fixes



<https://www.alten.com/mechanical-electrical-engineering-agile-method/>



# Take aways

**Stop thinking of Agile as a software practice**

**Move from a predictive to empirical lifecycle**

**Use all the tools in your toolbox**

# SE Agility Primer (wip)

## Eight Strategic Aspects

Tuesday  
@ 1:30

### Product Line Architectures

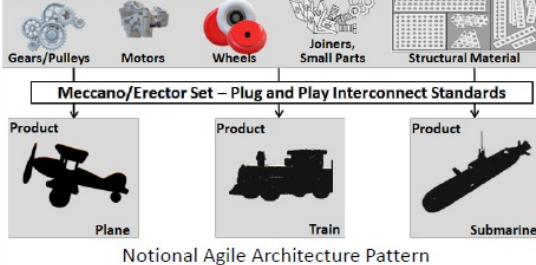
**Needs:** Facilitated product and process experimentation, modification, and evolution.

**Behaviors:** Composable and reconfigurable product and process designs from variations of reusable assets.

**Discussion:** One fixed process approach won't fit all projects, so an appropriate process should be easy to compose and evolve according to context and usage experience. Variations of reusable assets are built over time as features are modified for different contextual usage.

A hallmark of agile systems engineering is iterative incremental development, which modifies work in process as suitability is repetitively evaluated. The agility of the process depends upon the agility of the product – so both process and product can be easily changed.

Assets with Feature Variations



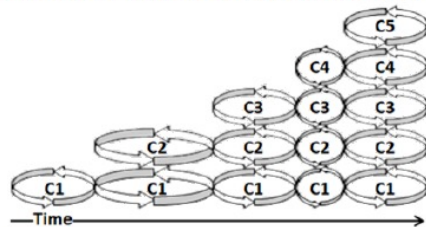
### Iterative Incremental Development

**Needs:** Minimize unexpected rework and maximize quality.

**Behaviors:** Incremental loops of building, evaluating, correcting, and improving capabilities.

**Discussion:** Generally increments *create* capabilities and iterations add and augment features to *improve* capabilities.

- Increment cycles are beneficially timed to coordinate events such as integrated testing and evaluation, capability deployment, experimental deployment, or release to production.
- Increments may have constant or variable cadence to accommodate management standards or operational dynamics.
- Iteration cycles are beneficially timed to minimize rework cost as a project learns experimentally and empirically.



Iterative capability improvements (looping) and incremental capability additions (successive development periods)

### Attentive Situational Awareness

**Needs:** Timely knowledge of emergent risks and opportunities.

**Behaviors:** Active monitoring and evaluation of relevant internal and external operational-environment factors.

**Discussion:** Are you doing things right (internal awareness) and doing the right things (external awareness)? Having the agile capability for timely and cost-effective change does little good if you don't know when that ability should be exercised. Situational awareness can be enhanced with systemic methods and mechanisms.



Alert in-the-moment constant attention

### Attentive Decision Making

**Needs:** Timely corrective and improvement actions.

**Behaviors:** Systemic linkage of situational awareness to decisive action.

**Discussion:** Empower decision making at the point of most knowledge. As a counter example, technical debt (a term for knowing something needs correction or improvement but postponing action) is situational awareness without a causal link to prompt action.



John Boyd's OODA loop

### Common-Mission Teaming

**Needs:** Coherent collective pursuit of a common mission.

**Behaviors:** Engaged collaboration, cooperation, and teaming among all relevant stakeholders.

**Discussion:** Collaboration, cooperation, and teaming are not synonymous, and need individual support attention. Collaboration is an act of relevant information exchange among individuals, cooperation is an act of optimal give and take among individuals, and teaming is an act of collective endeavor toward a common purpose.



Tightly integrated coherent operation

### Shared-Knowledge Management

**Needs:** Accelerated mutual learning and single source of truth for internal and external stakeholders.

**Behaviors:** Facilitated communication, collaboration, and knowledge curation.

**Discussion:** There are two kinds of knowledge to consider. Short time frame operational knowledge: what happened, what's happening, what's planned to happen. Long time frame curated knowledge: what do we know of reusable relevance, e.g., digital artifacts, lessons learned, and proven practices.



Depicted books represent information containers of any kind; but typically digital

### Continual Integration & Test

**Needs:** Early revelation of system integration issues.

**Behaviors:** Integrated demonstration and test of work-in-process.

**Discussion:** Discovering integration issues late in development activities can impact cost and schedule with major rework. Synchronizing multiple domain engineering activities via continual integration and test provides faster and clearer insight into potential system integration issues.



SpaWar iteratively evolving unmanned technology integration platform.

### Being Agile: Operations Concept

**Needs:** Attentive operational response to evolving knowledge and dynamic environments.

**Behaviors:** Sensing, responding, evolving.

**Discussion:** Agile systems engineering is not about doing Agile, it is about being agile. Being agile is a behavior, not a procedure – a behavior sensitive to threats and opportunities in the operational environment, decisive when faced with threat or opportunity, and driven to improve these capabilities. Deciding how to implement any of the core aspects, even this one, should be done with sense-respond-evolve principles in mind as aspect objectives.



Three principles that operationalize agility

# Strategic Aspects <sup>1/2</sup>

## Product-Line Architecture

**What:** Composable and reconfigurable designs from variations of reusable assets.

**Why:** Facilitated product & process experimentation, modification, and evolution.

## Iterative Incremental Development (and delivery)

**What:** Incremental loops of building/evaluating/correcting/improving/delivering capabilities.

**Why:** Minimize unexpected rework and maximize quality.

## Attentive Situational Awareness

**What:** Active monitoring/evaluation of internal & external threats/opportunities.

**Why:** Timely knowledge of emergent risks and opportunities.

## Attentive Decision Making

**What:** Systemic linkage of situational awareness to decisive action

**Why:** Timely corrective and improvement actions.



# Strategic Aspects <sup>2/2</sup>

## Common-Mission Teaming

**What:** Engaged collaboration, cooperation, and teaming among stakeholders.

**Why:** Coherent collective pursuit of a common mission.

## Shared-Knowledge Management

**What:** Facilitated communication, collaboration, and knowledge curation.

**Why:** Accelerated mutual learning and single source of truth for all stakeholders.

## Continual Integration and Test

**What:** Integrated demonstration and test of work-in-process.

**Why:** Early revelation of system integration issues.

## Being Agile: Operations Concept

**What:** Sensing, responding, evolving.

**Why:** Attentive response to evolving knowledge and dynamic environments.

# Common Strategies, Different Tactics

Provocative examples: many contrasts can be listed in each row depending upon project context.

Strategic Aspect	Software Engineering	Systems Engineering
Product-Line Architecture	<ul style="list-style-type: none"><li>• Standard interface</li><li>• COTS object-oriented integrated dev environment</li></ul>	<ul style="list-style-type: none"><li>• Proprietary interface</li><li>• MOSA design effort</li></ul>
Iterative Incremental Development	<ul style="list-style-type: none"><li>• Tight</li><li>• Test time stability</li></ul>	<ul style="list-style-type: none"><li>• Loose</li><li>• Test facility conflicts</li></ul>
Attentive Situational Awareness	<ul style="list-style-type: none"><li>• Introspective</li><li>• Frequent user feedback</li></ul>	<ul style="list-style-type: none"><li>• Extrospective</li><li>• Sparse user feedback</li></ul>
Attentive Decision Making	<ul style="list-style-type: none"><li>• Simple</li><li>• Few people</li></ul>	<ul style="list-style-type: none"><li>• Complicated</li><li>• Many people</li></ul>
Common-Mission Teaming	<ul style="list-style-type: none"><li>• Homogeneous</li><li>• Shared language</li></ul>	<ul style="list-style-type: none"><li>• Heterogeneous</li><li>• Different languages</li></ul>
Shared Knowledge Management	<ul style="list-style-type: none"><li>• Code libraries</li><li>• Integrated tools</li></ul>	<ul style="list-style-type: none"><li>• PLM</li><li>• Federated tools</li></ul>
Continual Integration and Test	<ul style="list-style-type: none"><li>• Common platforms</li><li>• Synchronous</li></ul>	<ul style="list-style-type: none"><li>• Proprietary platforms</li><li>• Asynchronous</li></ul>
Being Agile: Operations Concept	<ul style="list-style-type: none"><li>• Do Agile</li><li>• Many COTS Options</li></ul>	<ul style="list-style-type: none"><li>• Be agile</li><li>• Home grown and tailored</li></ul>

# Take aways

**Common goals and strategies (why & what)**

**Context dependent tactics (how)**

**Discussing contrasts can provoke thinking and understanding**



# Last Words

## Kerry

- Engineering Agility is required for situational awareness adaptations
- The dynamics of time constantly impact agility approaches
- Temporal Factors + Decision Makers/Influencers + Accelerators + Prioritisation + Transformations → ↑ Engineering Agility

## Duncan

- There is a lot of confusion as to what Agile Systems Engineering is
- Vanilla Agile Software Engineering techniques will not work on Agile Systems
- But it is possible, and necessary, to increase the agility of SE

## Robin

- Stop thinking of Agile as a software practice
- Move from a predictive to empirical lifecycle
- Use all the tools in your toolbox

## Rick

- Common goals and strategies (why & what)
- Context dependent tactics (how)
- Discussing contrasts can provoke thinking and understanding