

# ISA Working Group 2020 Webinar Series

(Webinar will commence at 11:05 am)

Are you interested in joining the ISA Working Group? Let us know by e-mailing [SEP@theiet.org](mailto:SEP@theiet.org)

## ***Assurance in a Connected World***

**Webinar 3: Procuring Software Intensive Systems – Pitfalls and Recommendations**

### Panellists

Rosanna Butters – Speaker

Stephen Hatton – ISA WG Chair

Pete Hutchison – ISA WG Deputy Chair

John Canning – ISA WG Secretary

Tim Clement – ISA WG Member

Matthew Bassett – ISA WG Member

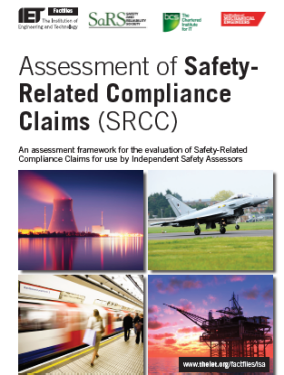
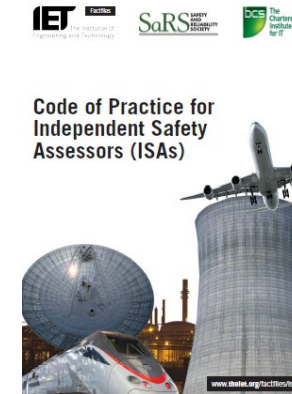
# ISA WG Terms of Reference – Purpose

- Promote the ISA role as a means of providing independent safety assurance of products to the supplier, purchaser and user
- Promote the ISA role of a safety professional in standards
- Support professional development by defining minimum standards, identifying training that meets minimum standards and supporting resources
- Support professional ISAs by developing guidance and providing information that affects their role

# Guidance – Published

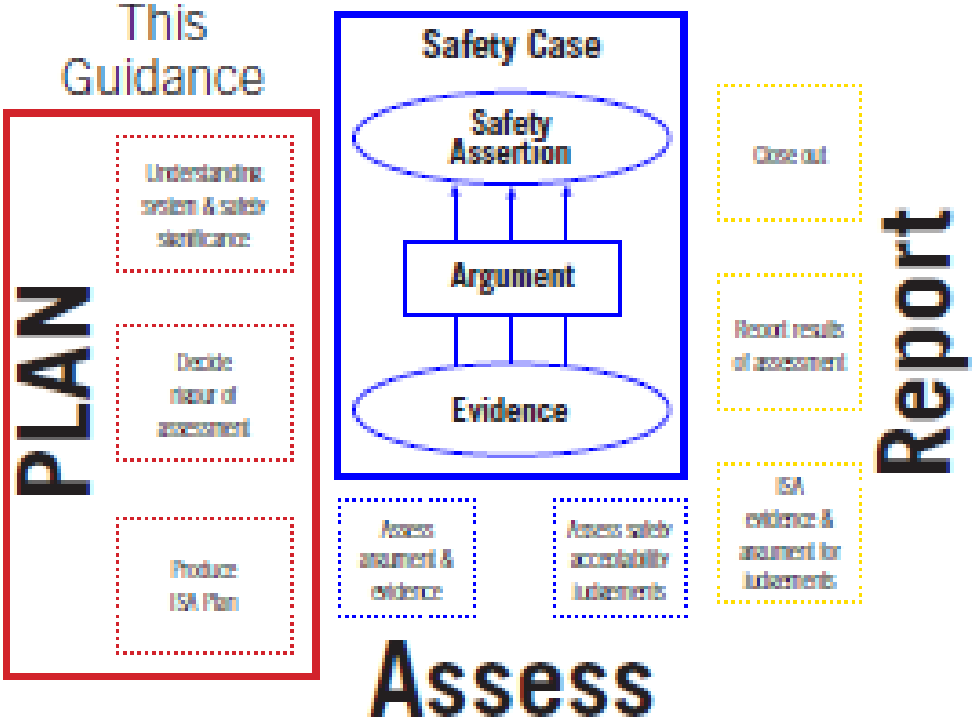
- General
  - ISA Working Group Terms of Reference
  - What is Independent Safety Assessment (ISA)?
- Professional
  - ISA Code of Practice for Independent Safety Assessors (ISAs)
  - Competency Framework for Independent Safety Assessors (ISAs)
- Substantive Guidance
  - Assessment of Safety Related Compliance Claims (SRCC)
  - Guidance on the Procurement of Independent Safety Assessors
- Guidance Notes / Position Papers
  - Guidance on the Use of Accident and Incident Data by ISAs
  - Documents useful to Independent Safety Assurance
  - Position Statement on Security, Safety and ISA

in Review for update



# Assessing a Safety Case Series

- Assessing a Safety Case Series
  - Guidance for Producing an ISA Plan for Assessing a Safety Case published
  - Guidance on Safety Assessment Reports to be published
  - Guidance on Degree of Rigour in progress



# Documents in Development

- Standards Group

- Requirements for independent review/assessment called up in Standards and Industry Guidance
- Environment Assurance and Safety Assurance

- Professionalism Group

- Using Key Performance Indicators with an ISA Contract ready for issue
- Agile Development – FEEDBACK BEING SOUGHT (Please complete our questionnaire!)

# Housekeeping

- Q&A (Zoom Webinar)
  - Use Q&A button to type your question (don't use chat button; don't raise hand)
  - Use 'thumbs up' to vote up or vote down a question (once only)
  - Panellists will select and discuss questions
  - Questions not discussed today will be recorded and commentary provided afterwards
- Feedback
  - Short re-cap article after the event
  - Please read and complete our questionnaire (to be e-mailed to you)
    - Questions organised around the structure of the presentation
    - No need to answer all questions
  - Let us know if you're interested in joining the ISA Working Group

# Procuring Software Intensive Systems – Pitfalls and Recommendations

## **Rosanna Butters**

*Rosanna (Rose) has an MA in Modern History and Spanish from the University of St Andrews and an MSc in Project and Enterprise Management from University College London. Prior to working for Transport for London (TfL) she worked for a gas consultancy and for a publication on Latin American legal matters. She has been working for TfL for five years, on a variety of civils projects, and for the past 18 months has been Project Manager for software on the Four Lines Modernisation (4LM) programme, one of the largest signalling upgrades in Europe. Outside her day job she carries out research into Organisational Learning and last September presented a paper on knowledge management tools at the European Conference on Knowledge Management in Lisbon.*

*A collation of the recommendations from industry specialists on best practice for the procurement and management of software intensive systems*

*Rosanna Butters*

## Procuring software intensive systems – pitfalls and recommendations



**EVERY JOURNEY MATTERS**



### The purpose

Transport for London often procures software intensive systems and the increasing prevalence of automated systems and artificial intelligence indicates that these contracts will become even more common.

Yet the numerous benefits of software intensive systems should be balanced against the potential for serious incidents. Several high profile incidents relating to software have made headlines in recent years, emphasising how critical it is to safely manage the implementation of such systems.

To help TfL safely prepare for a more autonomous future, this research was carried out as a continuous improvement project.

### Contributors

Founder of a critical systems software company

Safety-critical software consultancy

Electronic and automated services conglomerate

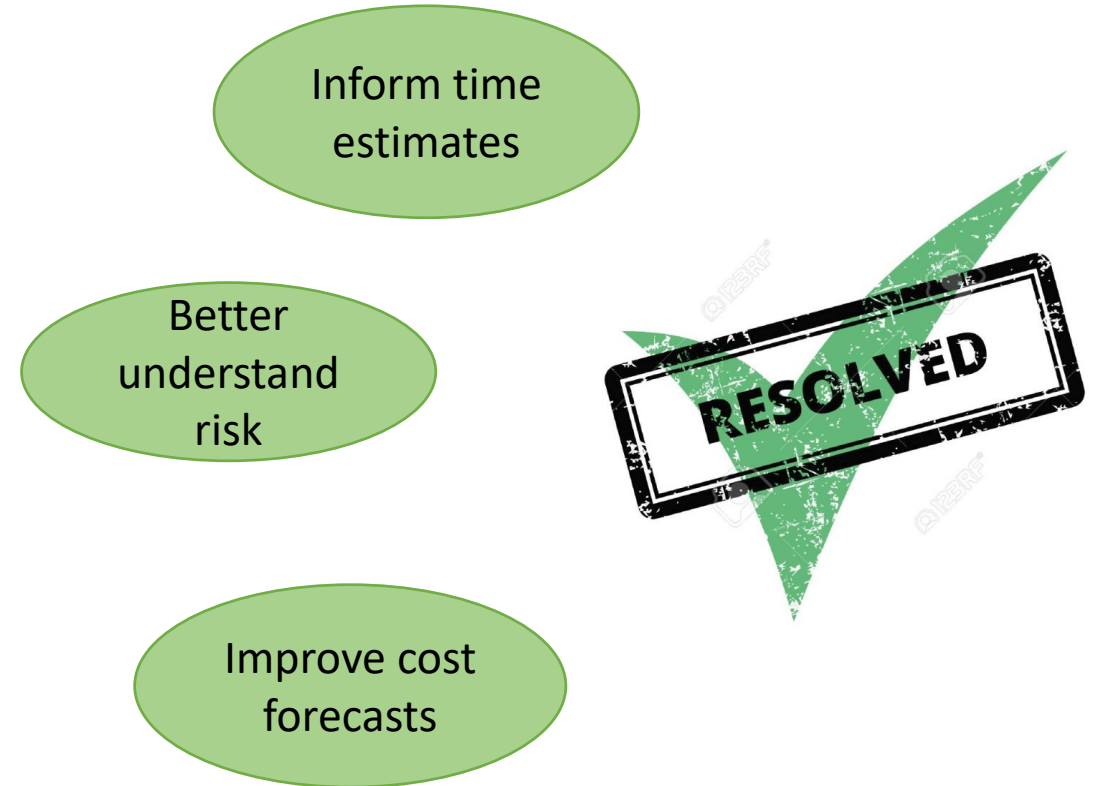
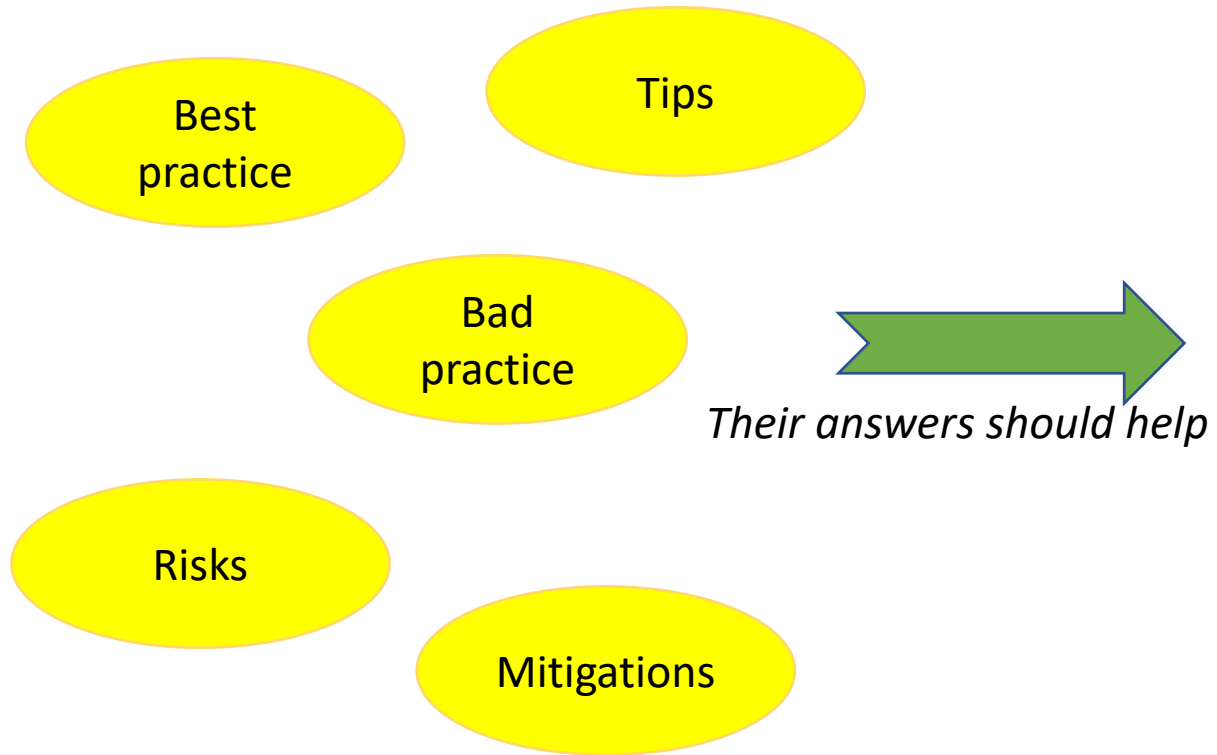
Multinational security and aerospace company

Safety-critical computer system consultancy



## Project overview

Five organisations specialising in safety critical software were asked about:



The information in this presentation is exclusively from other organisations and is not based on TfL experience or recommendations



## What are the main problems in Safety Critical Software projects?

Multiple interfaces

Inadequate change management

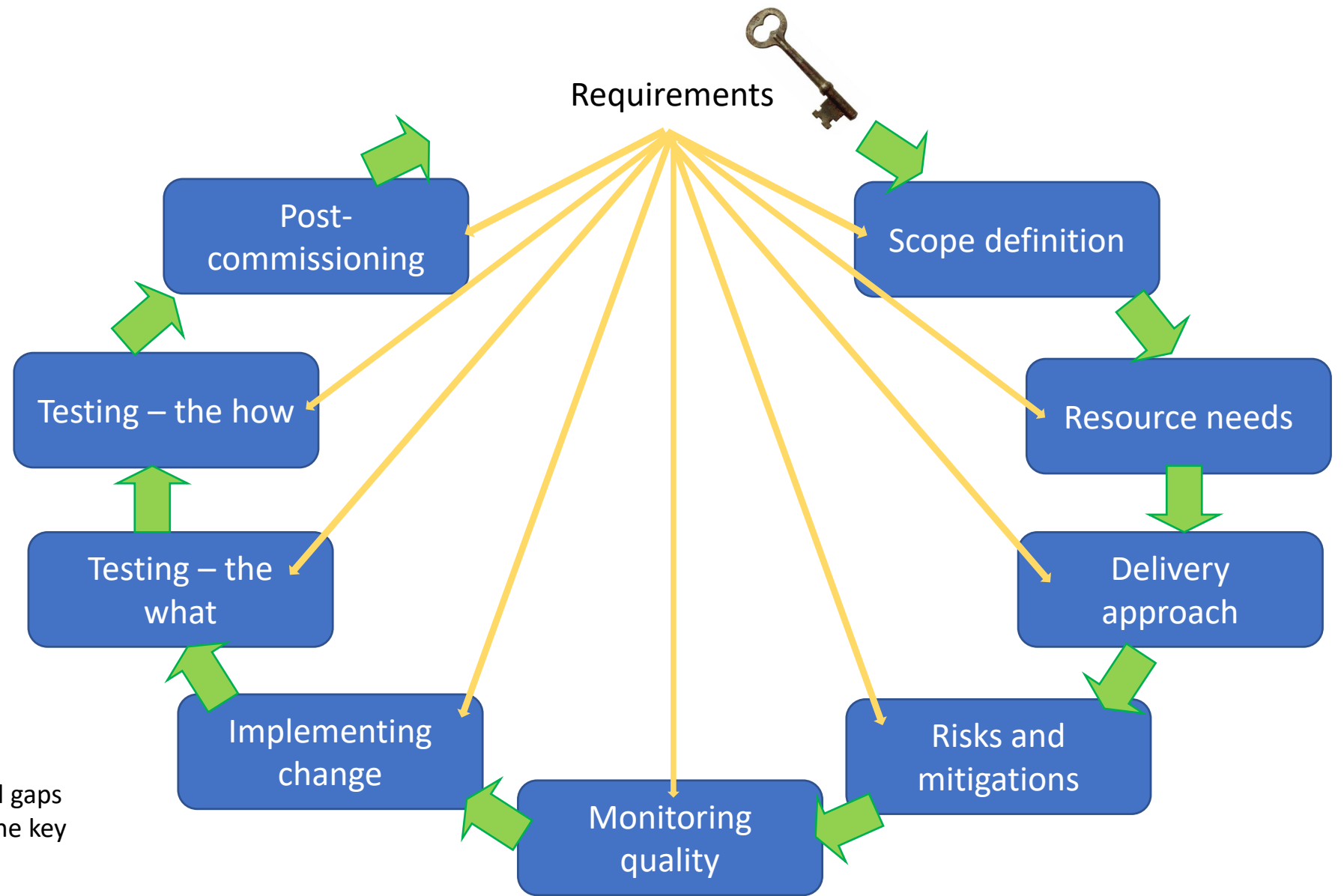
Complexity - and a lack of appreciation of it

Underestimated time and budget

Ambiguous requirements



# Software development project stages:



Eliciting the details, necessity and gaps of requirements at the outset is the key to success at all stages.



# *The Early Lifecycle Stages*



**EVERY JOURNEY MATTERS**

# Establishing the scope

*Minimising complexity is very very important*

Is an upgrade or a whole new system better?

Strike a balance between contractor and client risk when choosing between a tailor-made or off-the-shelf system

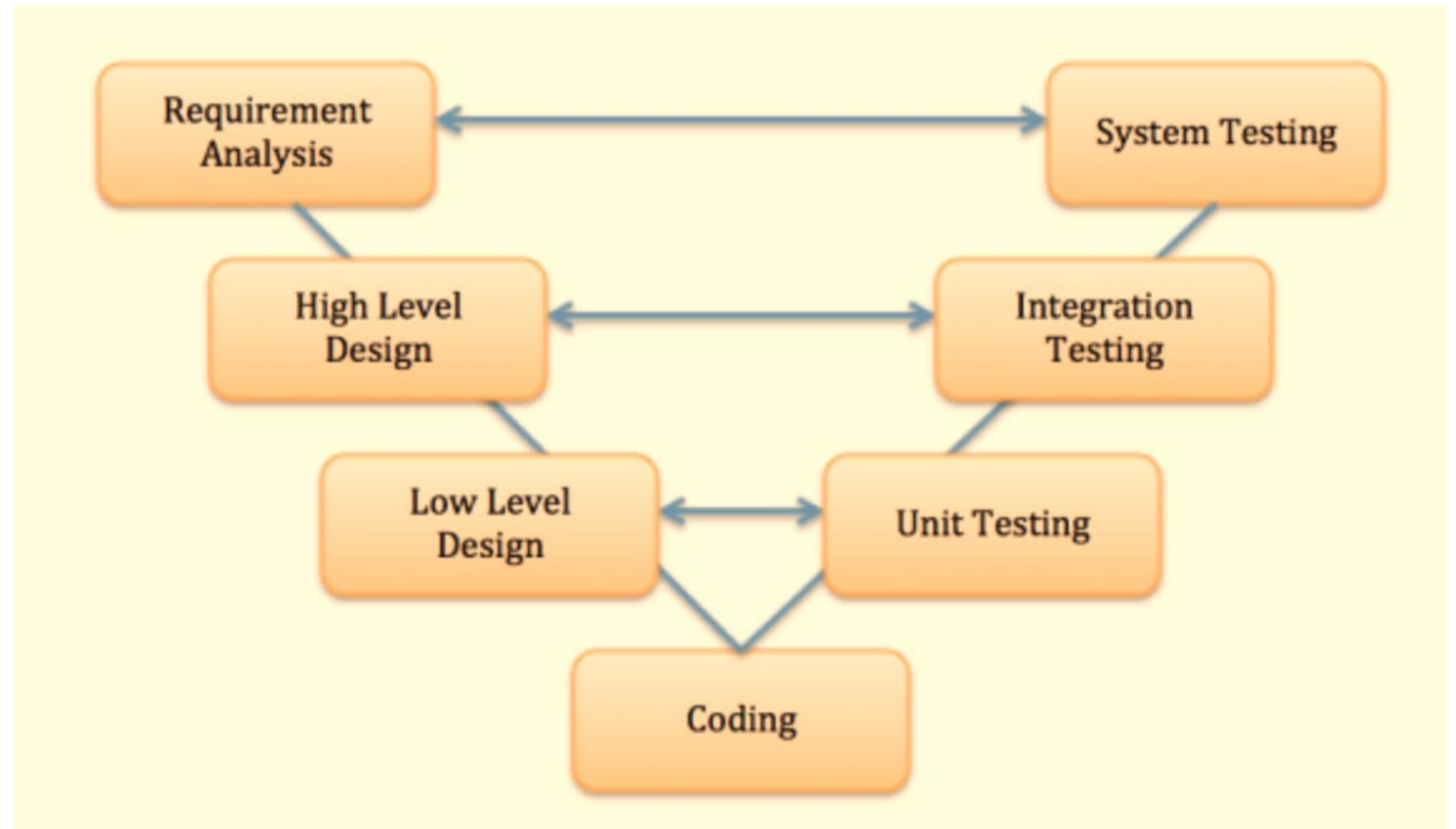


*Customers hate surprises but if you can't specify what you want you will get a surprise*

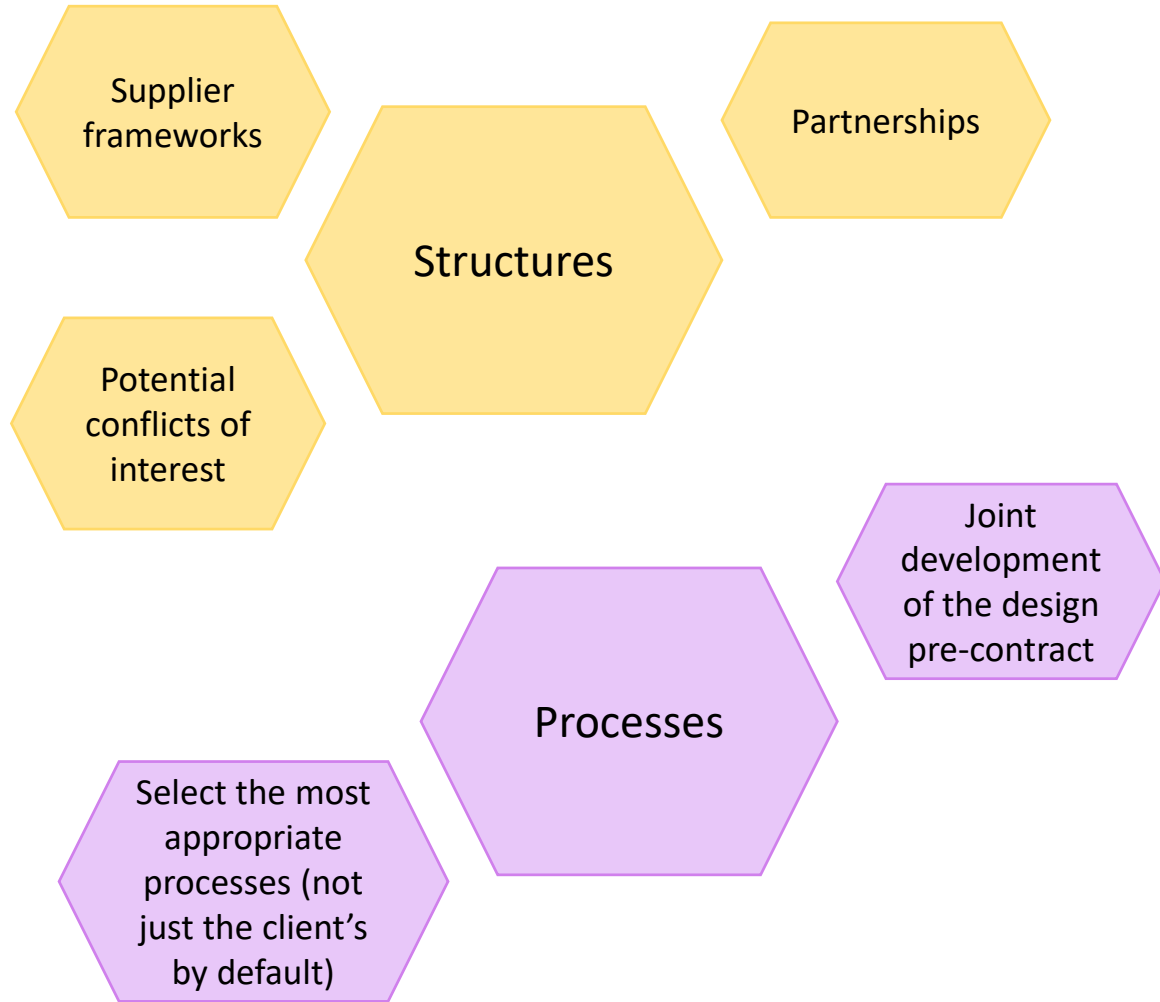


## Key tips on eliciting and specifying requirements

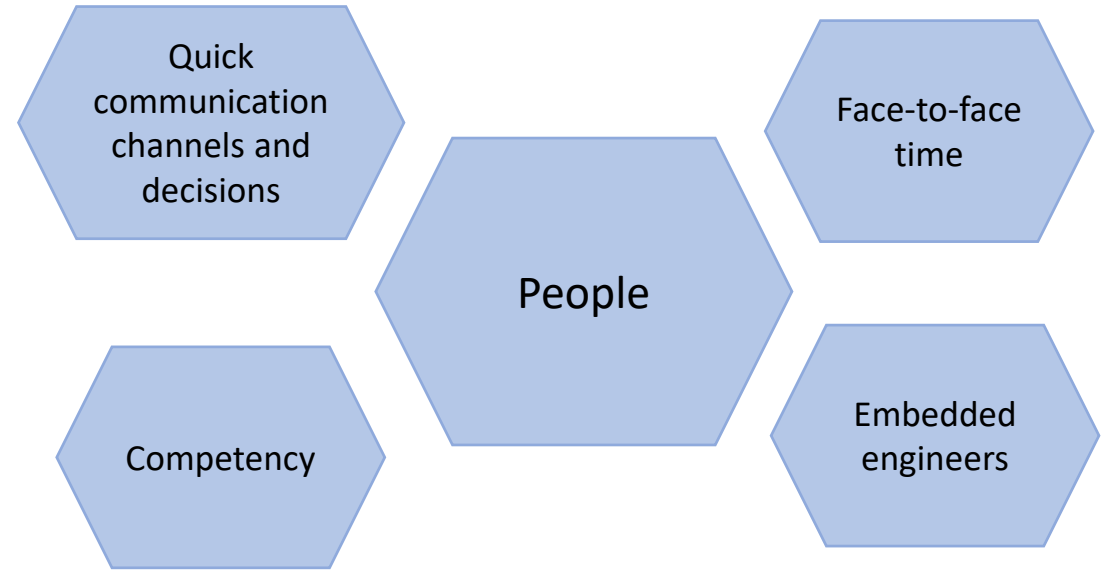
- Include as much information as possible about the target environment of the system
- Identify areas of greatest uncertainty at the start of the project
- Use joint interface specifications and joint tests through all the different stages of the V-model to minimize the need for late changes.



# Supplier Relationships

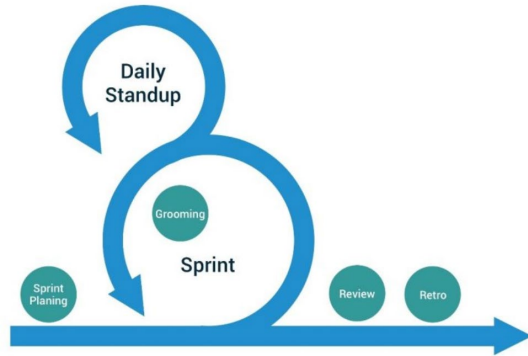


The relationship should exceed a mere delivery-contract, and be more like a sustainable long-term partnership.





## An agile approach?



### When does agile work best?

Agile is best suited to changes to an existing system, where the fundamental architecture and design does not need to change for the new features to be added. If this is not the case, the time and cost of the development is likely to increase drastically, or reliability and maintainability is likely to suffer.

### When is agile not a suitable approach?

Highly regulated environments, such as rail and aerospace, have assurance requirements that hamper the agile development model as:

- They require assurance loops that are not suitable for scrums, as scrums are designed to accommodate changes in requirements.
- It is not easy to provide the strong evidence of the system level properties of safety, security and reliability, using the tests in agile process



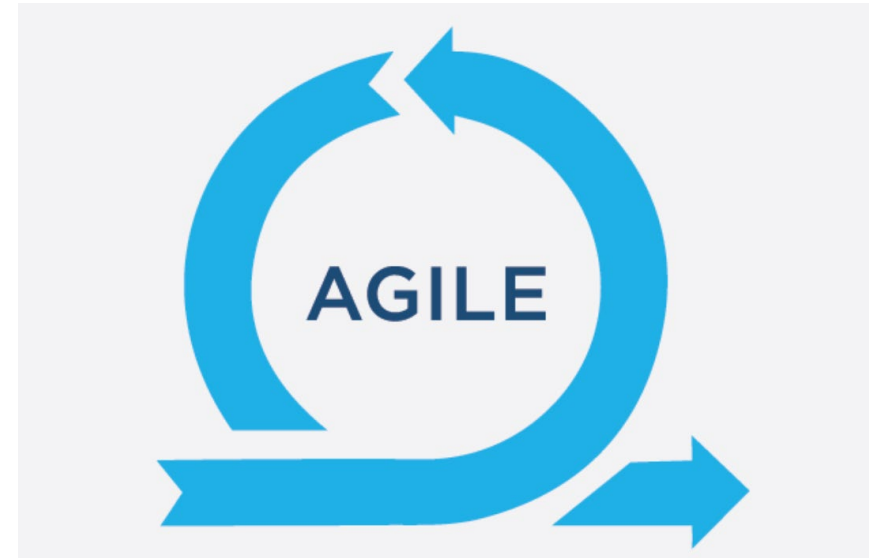
# An agile approach?

## Benefits

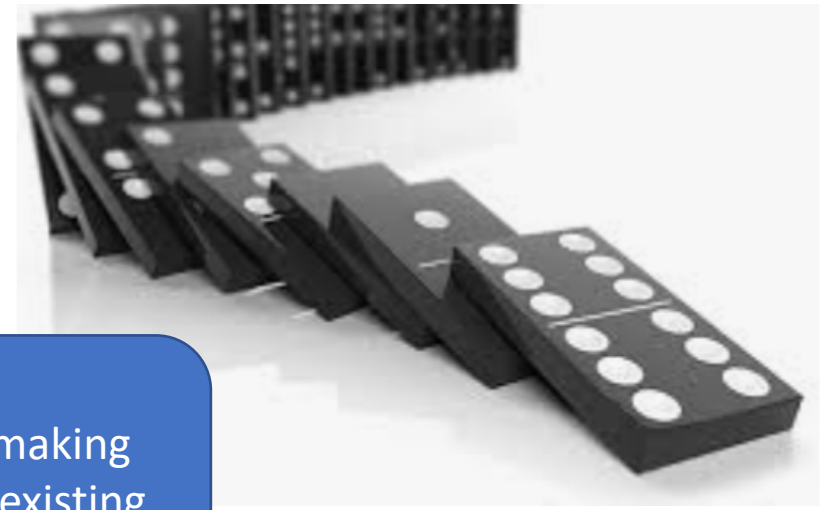
- Prevents wasted effort
- Helps deal with complexity
- More flexible
- Reduces costs

## Dangers

- Not good for systems that have not been created before
- Doesn't provide adequate documentation for systems with a long service lifetime
- Increases the risk of deviation from requirements
- Not appropriate for highly regulated environments



## Key considerations for writing the contract



Level of risk being taken on by each party



Impact of making changes to existing systems

The requirements should be:

set at the sub-system (not the functional) level

Set **before** design starts

Sufficient time and budget allocated for testing and rework



# *Managing Progress Throughout the Lifecycle*



**EVERY JOURNEY MATTERS**

## Warning sign

## Mitigation



Copying buggy code and incorporating off-the-shelf components



Root cause analysis  
Automated testing



Increasing levels of uncertainty



Define and conquer risk.



Lots of people on the project → less efficiency → shortcuts



Understand the interfaces and be realistic about the schedule and budget.



Lots of non-critical defects could indicate critical defects.



Pyramid assessment  
Out of the box questions



## Warning sign



Claims about the system not substantiated by evidence



Lack of transparency



Combining high integrity systems



Manual coding



Code not being checked by senior/independent reviewers

## Mitigation



Robust qualitative and quantitative evidence



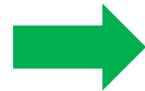
Non-disclosure agreements



Master-slave system  
Rigorous testing  
Staged integration



Automate coding

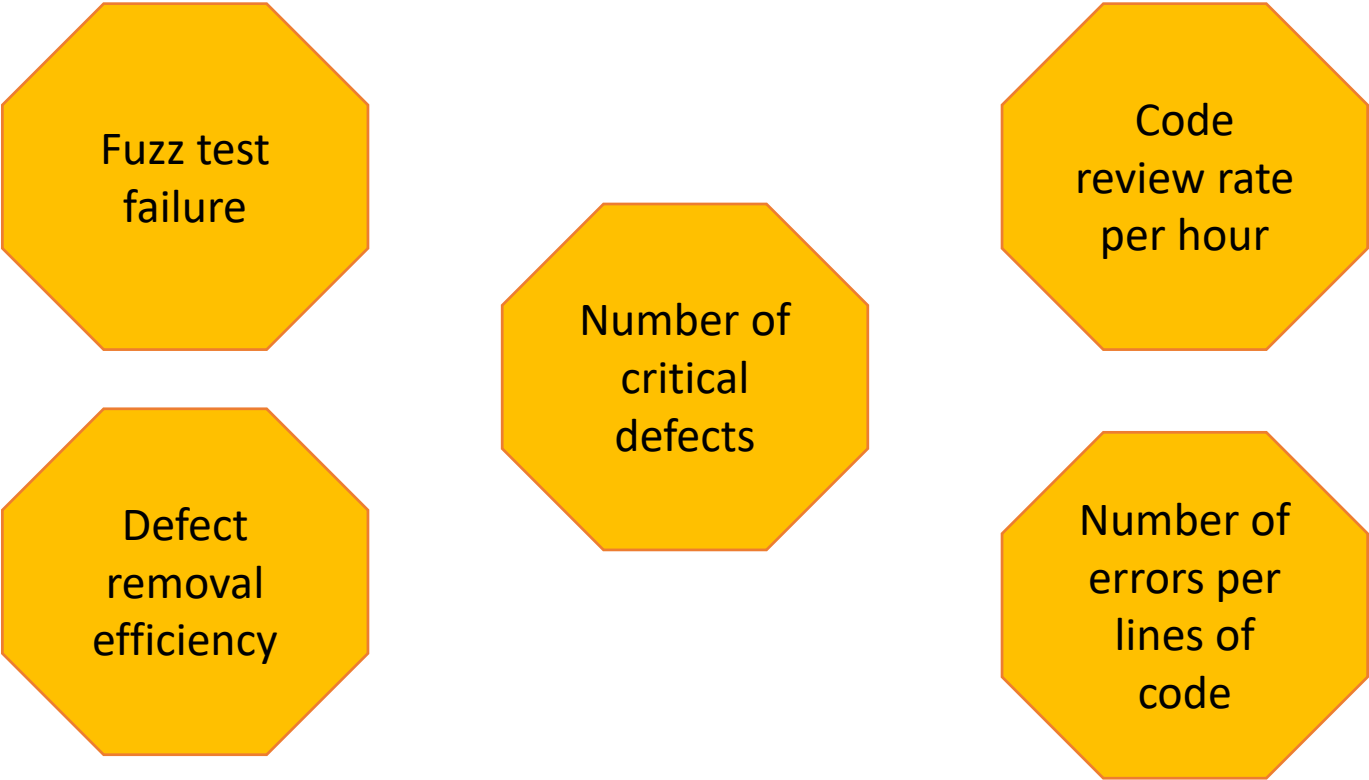


Enforce approved reviewal structure



# Mechanisms for monitoring the health of software development

Changes in the metrics below can indicate a change in the quality of the software, prompting mitigation actions or further investigation:



## Audits

Audits can be constructive ways of understanding current progress, gauging quality and identifying improvements. Some tips are:

Identify where in the process the error occurred

Conduct audits in person at their office

Audit hardware and software

In-depth reviews better than frequent ones

Lead by example





## Cyber security

When managing cybersecurity, remember the following:



**Consider cybersecurity and safety throughout the entire project lifecycle, from procurement to de-commissioning**

Manage cybersecurity with the **same processes and attention as safety**

**If it's not secure, it's not safe** – any claims made about how safe a system is are not valid unless they are also informed by security

If an error message appears about a system failure, **don't just assume the message is correct** – lots of cybersecurity issues start with this problem

Detailed guidance on protecting assets, developing a security strategy and adopting the right mindset when thinking about cybersecurity can be found on the Centre for Protection of National Infrastructure [website](#)



## Change management

The quantity of changes to requirements can overwhelm the project so change management should:

Often people assume it's just a case of "adding a couple of lines of code"

Be managed jointly with suppliers

Adopt a system-level view

Collate smaller, less urgent, changes to review in batches

Acknowledge time and cost impact



# *Testing the Software*



**EVERY JOURNEY MATTERS**

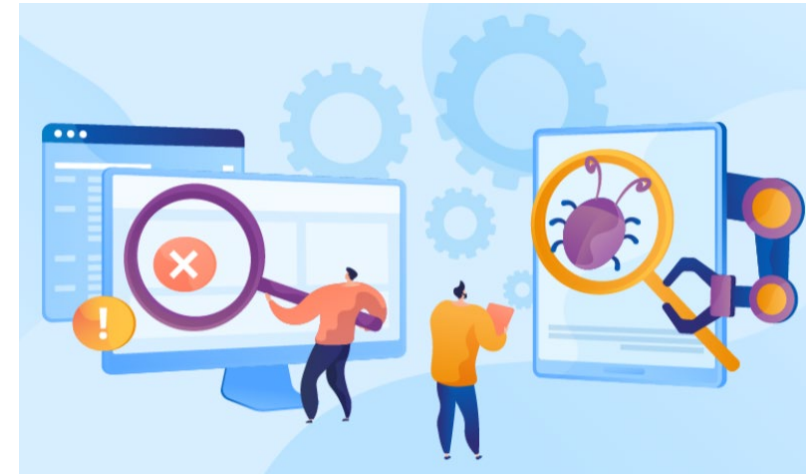
## Testing requirements

When deciding what tests to use, think about:

- Are you using tests to build confidence or to get rid of bugs?
- Different tests can be used to target critical areas
- Is the required confidence level defined in the contract?
- What are your assumptions?
- What is the purpose of the system?
- What went wrong in failures in similar projects?



*If elements are outsourced,  
do the developers still  
understand the context?*

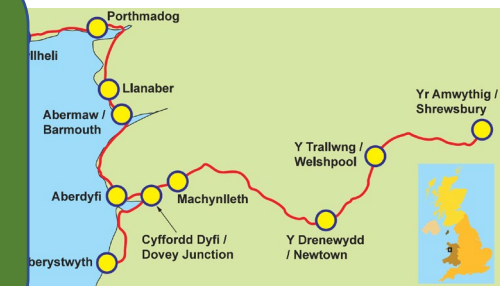


## *Illustrative examples*

The Boeing 737 Max crashes illustrated why operators need to understand the system and the danger of single-points of failure



The Cambrian Temporary Speed Restriction incident shows the importance of a robust checking process, particularly when part of a non-standard process



1: Geographical area controlled by Machynlleth signalling centre

The Ariane 5 flight failure shows the criticality of complete simulations and understanding restrictions



# Loss of Safety Critical Signalling Data on the Cambrian Coast Line - 2017

[Link to report](#)

## What happened?

- Temporary speed restriction (TSR) of 30kph exceeded by 50kph

## Causes

- Software was single point of failure for TSR data and signalling display
- No independent check of TSR data upload
- System safety justification in a non-standard format, meaning changes to design not identified and lack of clarity of design not noticed

## Learning points

- Technical solution should **remove need for humans to check** automatically updated speed restrictions
- Train drivers should **report inconsistencies** in data provided to them
- Independent Safety Assessors should **understand scope of checks** undertaken by other bodies and apply **extra vigilance if documents are part of a non-standard process**
- The **specified level of safety must be achieved** when implementing TSRs
- **Clients must undertake client role** when procuring high-integrity software

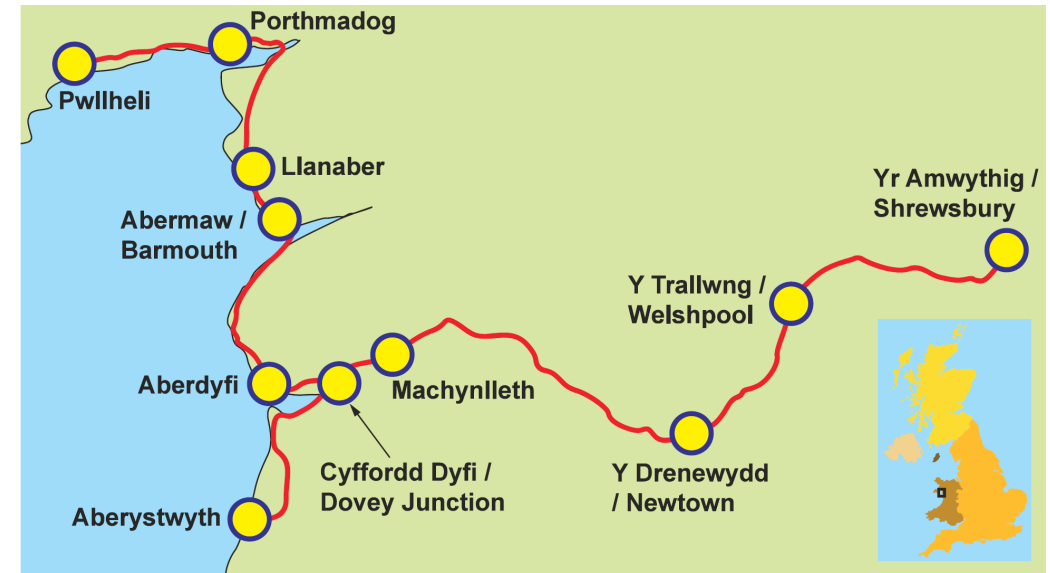


Figure 1: Geographical area controlled by Machynlleth signalling centre



# Ariane 5 Flight 501 failure - 1996

[Link to report](#)

## What happened?

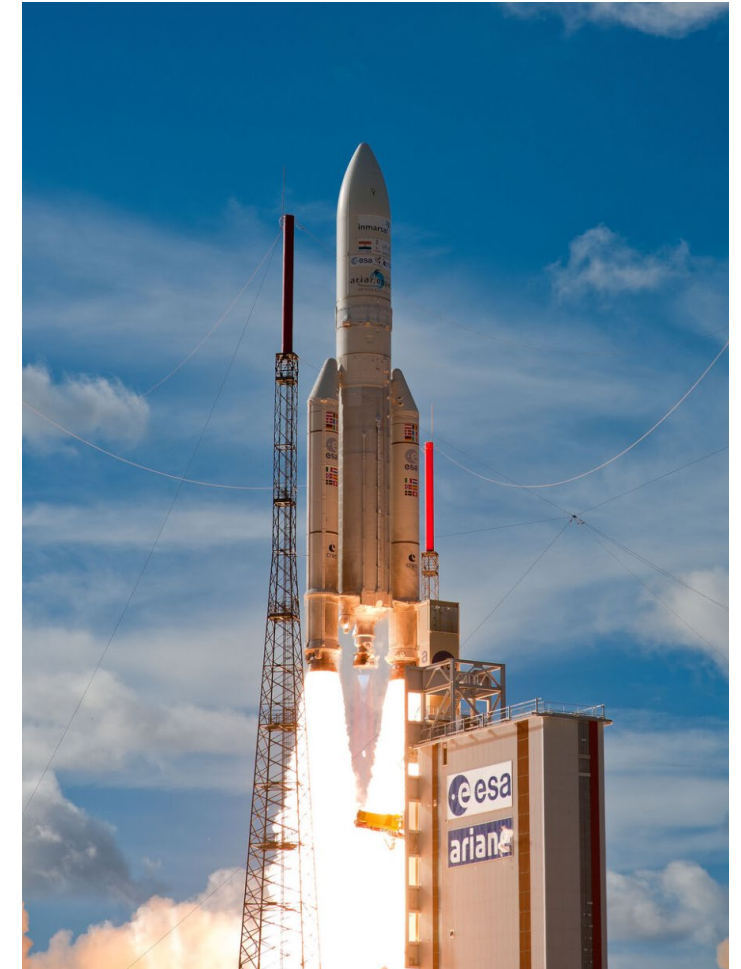
- At an altitude of about 3700 m, the launcher veered off its flight path, broke up and exploded.

## Causes

- Ariane 5 had a higher a horizontal velocity than Ariane 4 but used the same software
- The design change to not protect the inertial system computer was not properly understood
- The specification of the inertial reference system and tests performed did not specifically include the Ariane 5 trajectory data
- It was decided to use the simulated output of the inertial reference system, not the system itself or its detailed simulation

## Learning points

- Prepare a **test facility including as much real equipment** as technically feasible, inject **realistic input data**, and perform complete, closed-loop, system testing.
- **Complete simulations** must take place before any mission.
- All **restrictions on use of the equipment shall be made explicit** for the Review Board.
- Make all critical software a Configuration Controlled Item (CCI).
- Include **external participants when reviewing specifications, code and justification documents**.
- Give justification documents the same attention as code
- Close engineering cooperation, with **clear cut authority and responsibility**.



## Boeing 737 MAX (Lion Air flight) - 2018

[Link to report](#)

### What happened?

- 346 people died in crashes resulting from nosedives caused by the MCAS, a new automated flight control which Boeing omitted from crew manuals

### Causes

- Boeing assumed that:
  - the MCAS function is automatic;
  - the responses for it are the same as for existing procedures;
  - and that crews were not expected to encounter MCAS in normal operation.
- Boeing did not provide information and additional training requirements for the 737-8 (MAX) since the condition was considered similar to previous 737 models.
- Human error was not included in the probability analysis, even though the flight crew is often used as a means to mitigate a failure condition.
- The design of MCAS relied on input from a single sensor, making it susceptible to a single point of failure.

### Learning points

- **Failure Mode and Effects Analysis** would have been able to **identify single-point and latent failures** which have significant effects as in the case of MCAS design.
- Consider the **effect of all possible flight deck alerts on flight crew response**
- **Closely scrutinize the development and certification process for systems** whose malfunction can **lead to loss of control** of the airplane.
- Provide flight crew with information and alerts to **help them understand the system** and know how to resolve potential issues






*What are the key recommendations?*

It's likely that much of the content in this presentation is familiar to you, but **our peers in industry are telling us that these are areas where software intensive systems often fall down, so key things we should remember are...**




**EVERY JOURNEY MATTERS**

## Key lessons: day-to-day



Ensure there's face-to-face contact and direct communication channels

Monitor the metrics, warning signs and processes to address problems at their cause



Be realistic about the functional, cost and time impact of changes – and whether they are needed at all



Clarify requirements and areas of uncertainty early and monitor progress against them



## Key lessons: longer term



Invest in long-term supplier relationships

Establish a stronger client-contractor relationship internally within organisations, reducing the temptation to change requirements

Allow a larger proportion of budgets and programmes to be allocated to rework and testing



# Next Events in the Series

## Webinar 4: Sufficient Assurance?

(Wednesday 2 December at 11:00)

## Webinar 5: AI and Functional Safety

(Thursday 17 December at 11:00)

(Register at IET Events)

Are you interested in joining the ISA Working Group?

Let us know by e-mailing [SEP@theiet.org](mailto:SEP@theiet.org)