Agile Goes Critical

Jane Cleland-Huang, PhD
University of Notre Dame
Department of Computer Science and Engineering
JaneClelandHuang@nd.edu
sarec.nd.edu

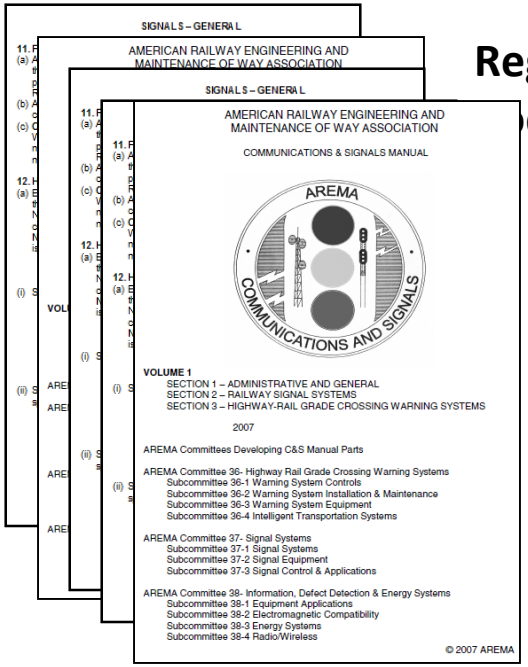
Some of the ideas described in this talk emerged as a result of projects funded by the US National Science Foundation under Grants CCF-0959924 and CCF-1265178.

Safety Critical Software

Software whose failure or malfunction may result in death or serious injury to people or in loss or severe damage to equipment or property.

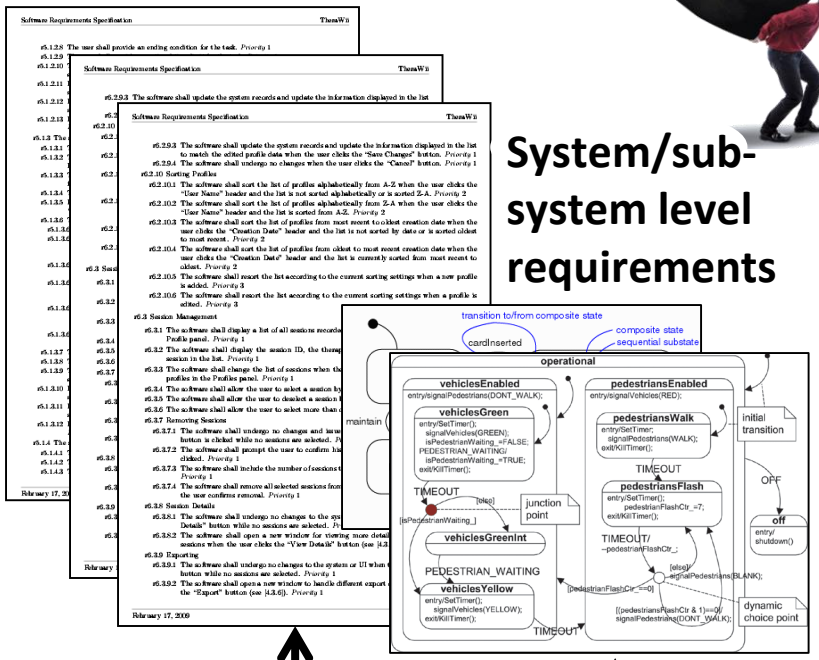
Such systems cannot be fragile!

Demonstrate Compliance/Safety



Regulatory codes

Satisfies relevant codes



System/sub-system level requirements

Realizes

Code

Validates

Tests

Test cases (unit, integration, acceptance)

Certification standards prescribe the use of traceability to demonstrate compliance to regulatory codes and to show that all hazards have been mitigated.

The Traceability Gap

Based on over a decade of traceability engagements in industrial projects we have observed a **<u>traceability gap</u>** between what is prescribed and what is delivered:

FOCUS: SAFETY-CRITICAL SOFTWARE

Strategic Traceability for Safety-Critical Projects

Patrick Mäder, Ilmenau Technical University

Paul L. Jones and Yi Zhang, US Food and Drug Administration

Jane Cleland-Huang, DePaul University

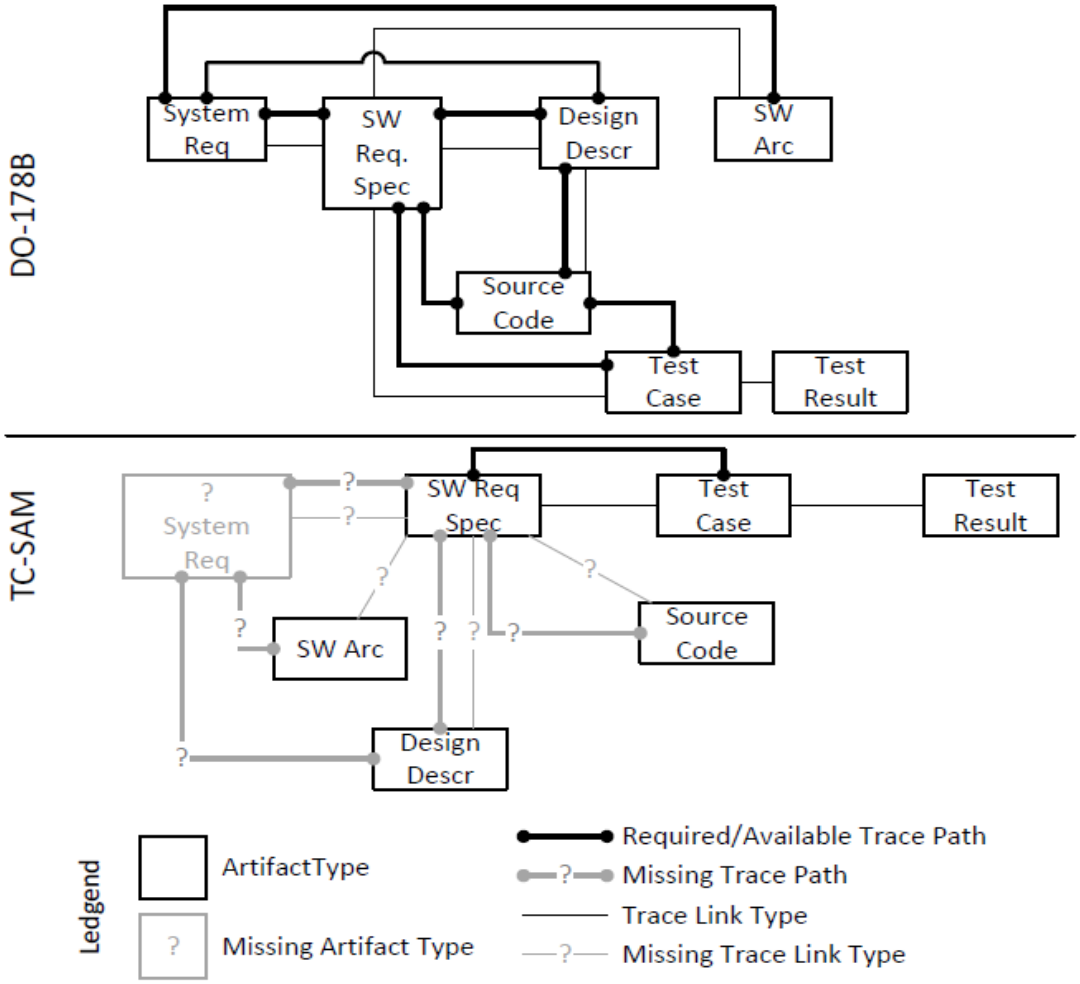
Our study of the traceability components of Medical Device submissions to the FDA showed **<u>incomplete and sometimes entirely missing trace links</u>**, inaccurate, redundant traceability – delivered through a big bang solution.

Mind the Gap: Assessing the Conformance of Software Traceability to Relevant Guidelines

Patrick Rempel, Patrick Mäder, Tobias Kuschke, and Jane Cleland-Huang

A **<u>formal comparison of five safety-critical software systems</u>** against prescribed guidelines showed **<u>similar traceability problems</u>**.

The Traceability Gap



Mind the Gap: Assessing the Conformance of Software Traceability to Relevant Guidelines, Patrick Rempel, Patrick Mäder, Tobias Kuschke (TU Ilmenau), and Jane Cleland-Huang , ICSE 2014, Hyderabad, India

Injecting Safety into Agility

In this talk I'm going to share with you my own experiences and how I've woven safety critical practices into agile..

1. Preliminary Hazard Analysis
2. Continuous safety assessment
3. Continuous traceability analysis
4. Continuous Safety Assurance

Safety Effort: Fit for purpose

DO-178b Criticality Levels:

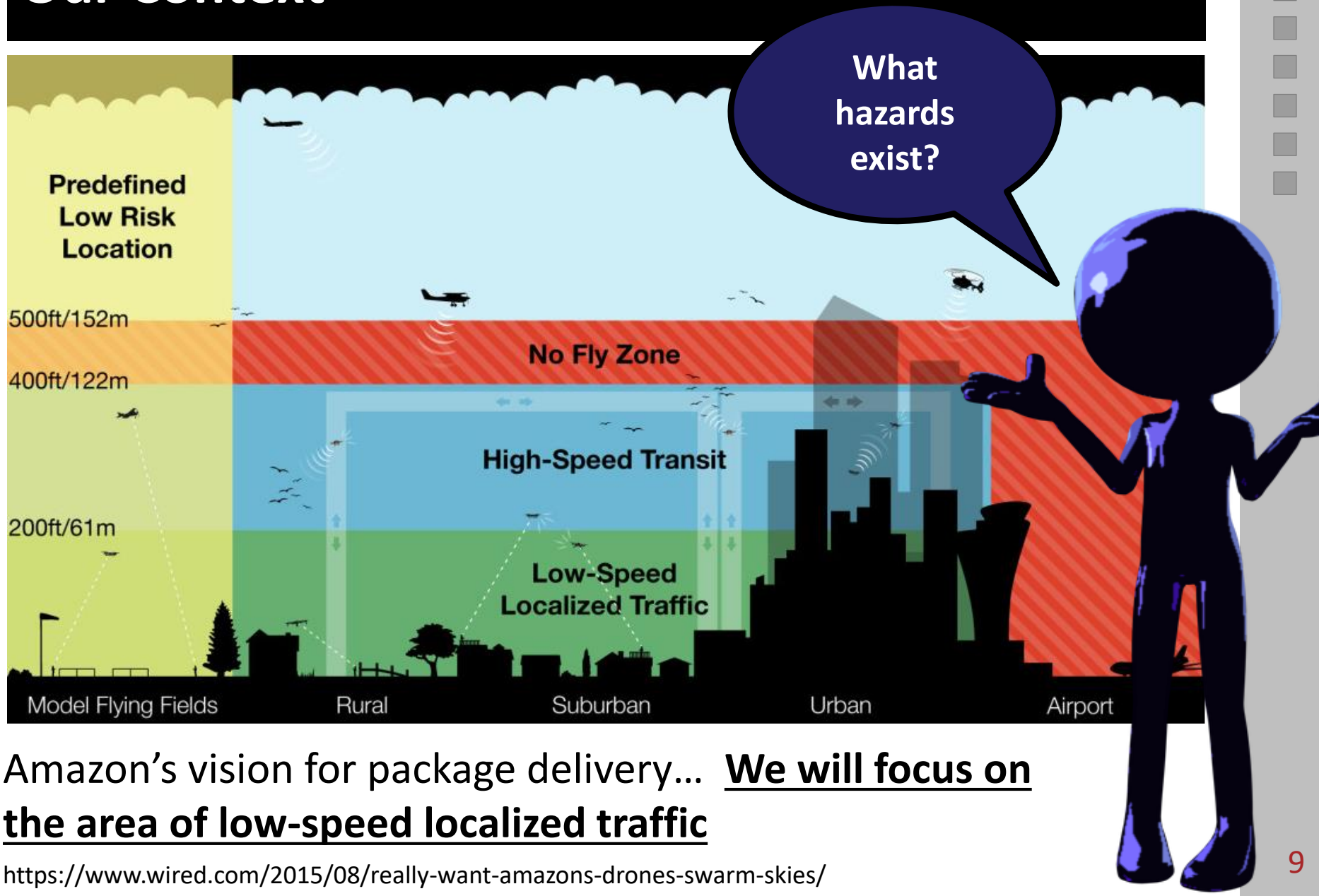
Catastrophic	Failure may cause a crash. Error or loss of critical function required to safely fly and land aircraft.
Hazardous	Failure has a large negative impact on safety or performance, or reduces the ability of the crew to operate the aircraft, or causes serious or fatal injuries among the passengers.
Major	Failure is significant, but has a lesser (for example, leads to passenger discomfort rather than injuries)
Minor	Failure is noticeable
No Effect	Failure has no impact on safety

Safety Critical?

Drones	E-Health	Environment	Mission-Control
			
Fleets of drones are coordinated to deliver medical supplies in a natural catastrophe and to use aerial reconnaissance for tracking.	A person recovering from a medical issue such as a heart attack, is safely monitored during his/her rehabilitation.	Crowd-sourced pollution detection in case of a chemical spill. (Also for environmental pollution).	Soldiers (or rescue team) on a mission. Monitor position & health of team. Provide instruct-ions and informa-tion via visor.

Examples of safety-critical systems that everyday developers might engage in…

Our Context



Amazon's vision for package delivery… **We will focus on the area of low-speed localized traffic**

https://www.wired.com/2015/08/really-want-amazons-drones-swarm-skies/

Integrating Safety into the Agile Process

Inputs from Stakeholders, Customers, Users, Team

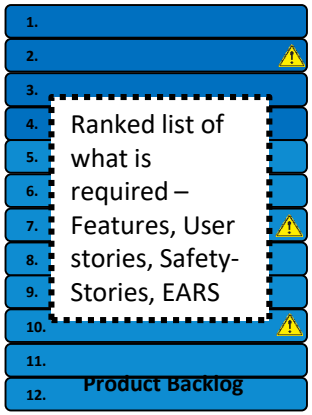
Product Owner

Given a standard SCRUM or Kanban Process, where do safety activities fit?

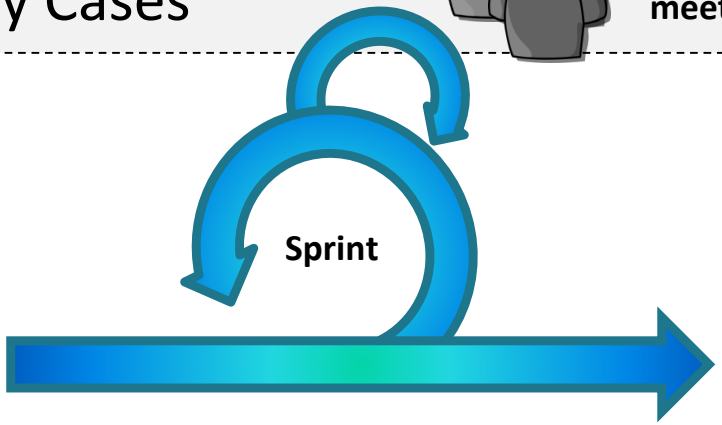
Before answering this question – let's look at several safety aspects:

- FMECAs
- Safety Stories
- Traceability
- Safety Cases

Scrum meeting



Team selects user stories for the sprint



Working Product

FMECA: Data Faults

Preliminary Hazard: Drone's battery fails unexpectedly and the drone crashes.

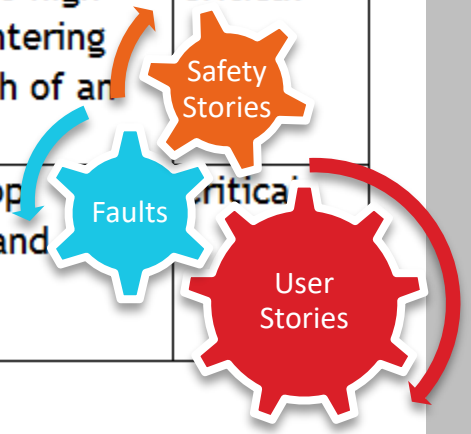
❶ The battery level **detector** fails to detect a low battery level.

❷ Battery level **indicator** fails and an incorrect battery level is reported.

❸ The software fails to check battery level in time to take responsive actions before the battery fails.

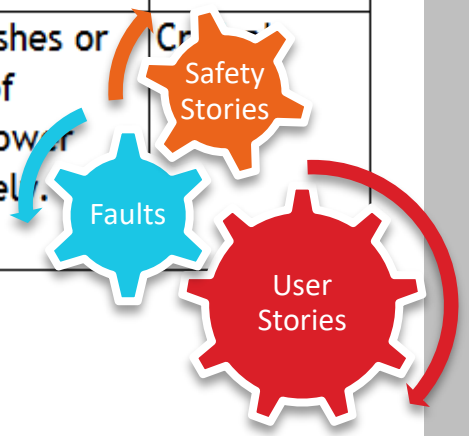
FMECA: Data Faults

ID	Data Item	Data Fault Type	Description		Effect	Criticality
FM-D2	Battery level indicator	Faulty error detection	Low battery level is not detected.	EF-4	Drone runs out of power and lands in an uncontrolled way.	Critical
FM-D3	Battery level	Faulty data	Battery level indicator depicts incorrect power availability.	EF-5	Drone runs out of power and lands in an uncontrolled way.	Critical
FM-D4	Drone health	Missing data	Drone fails to communicate its location	EF-6	Mission control can not accurately track the drone, potentially causing accidents such as drone crashes.	Critical
FM-D5	Altitude level	Faulty error detection	Altitude reading is lower than the actual altitude of the drone.	EF-7	Drone flies too high potentially entering the flight path of an airplane.	Critical
FM-D1	Landed status	Faulty status	On ground status = true even though drone is still in the air.	EF-8	Propellers stop prematurely and crashes	Critical



FMECA: Event Faults

Event	Event	Event Fault Type	Description		Effect	Criticality
FM-E1	Route miscalculated	Faulty Algorithm	Drone's route is miscalculated	EF-8	Drone flies into prohibited airspace	Catastrophic
FM-E2	Minimum separation distance violated	Faulty Algorithm	Minimum separation distance is not preserved between drones during flight.	EF-9	Drones crash in midflight	Catastrophic
FM-E3	Payload exceeds manufacturer's limits	Faulty functionality	Drone's payload is too heavy for sustained flight.	EF-10	Drone crashes or runs out of battery power	Critical
FM-E4	Windspeeds exceed manufacturer's limits.	Faulty functionality	Drone flies in windspeeds greater than specified by the manufacturer.	EF-11	Drone crashes or runs out of battery pow[illegible] prematurel[illegible]	Cr[illegible]



Using EARS to write Safety-Stories

<optional preconditions>
<optional trigger>
the <system name> shall
<system response>

- Simple structure adds rigor & clarity
- System response describes what the system must actually do that is *visible* at the boundary of the system

1. **Ubiquitous**
   Requirement is always active
2. **Event-driven** (keyword When)
   Required response to a triggering event
3. **State-driven** (keyword While)
   Required response in a specified state
4. **Option** (keyword Where)
   Applicable only if feature is included
5. **Hybrid**:
   Use combinations of when, while and where for requirements with complex conditional clauses.

Incremental in synch with Sprints or Kanban phases..

Alistair Mavin, Philip Wilkinson, Adrian Harwood, Mark Novak: Easy Approach to Requirements Syntax (EARS). IEEE International Conference on Requirements Engineering. 2009: 317-322

Using EARS to write Safety-Stories

Ubiquitous

The <*component name*> shall <*response*>

The drone shall maintain a *minimum-separation distance* at all times.

Event Driven.

When <*trigger*> the <*system name*> shall <*system response*>

When the drone is within X centimeters of minimum separation distance from another drone, the collision avoidance system shall provide directives to all drones in the vicinity.

State Driven

While <*in a specific state*> the <*system name*> shall <*system response*>

While in landing mode the drone shall descend vertically until it reaches the ground.

Option

Where <feature is included> the <system name> shall <system response>

Where parachute mode is enabled and a drop is initiated the drone shall scan the dropzone for obstacles.

Unwanted Behavior

If <optional preconditions> <trigger>, then the <system name> shall <system response>

If wind gusts exceed desired wind velocity but are below the maximum wind velocity, the drone shall return to base.

Assumptions

It is essential to understand your assumptions!

Wheels are turning if, and only if, plane is on the runway.

Led to an accident when a plane failed to brake because runway was wet and hydroplaning occurred.

Assumptions and Safety Requirements

Assumption: The Model-X Drone is able to transmit messages at distances up to 1000 meters.

User Story: As a drone dispatcher I will dispatch drones to deliver supplies.

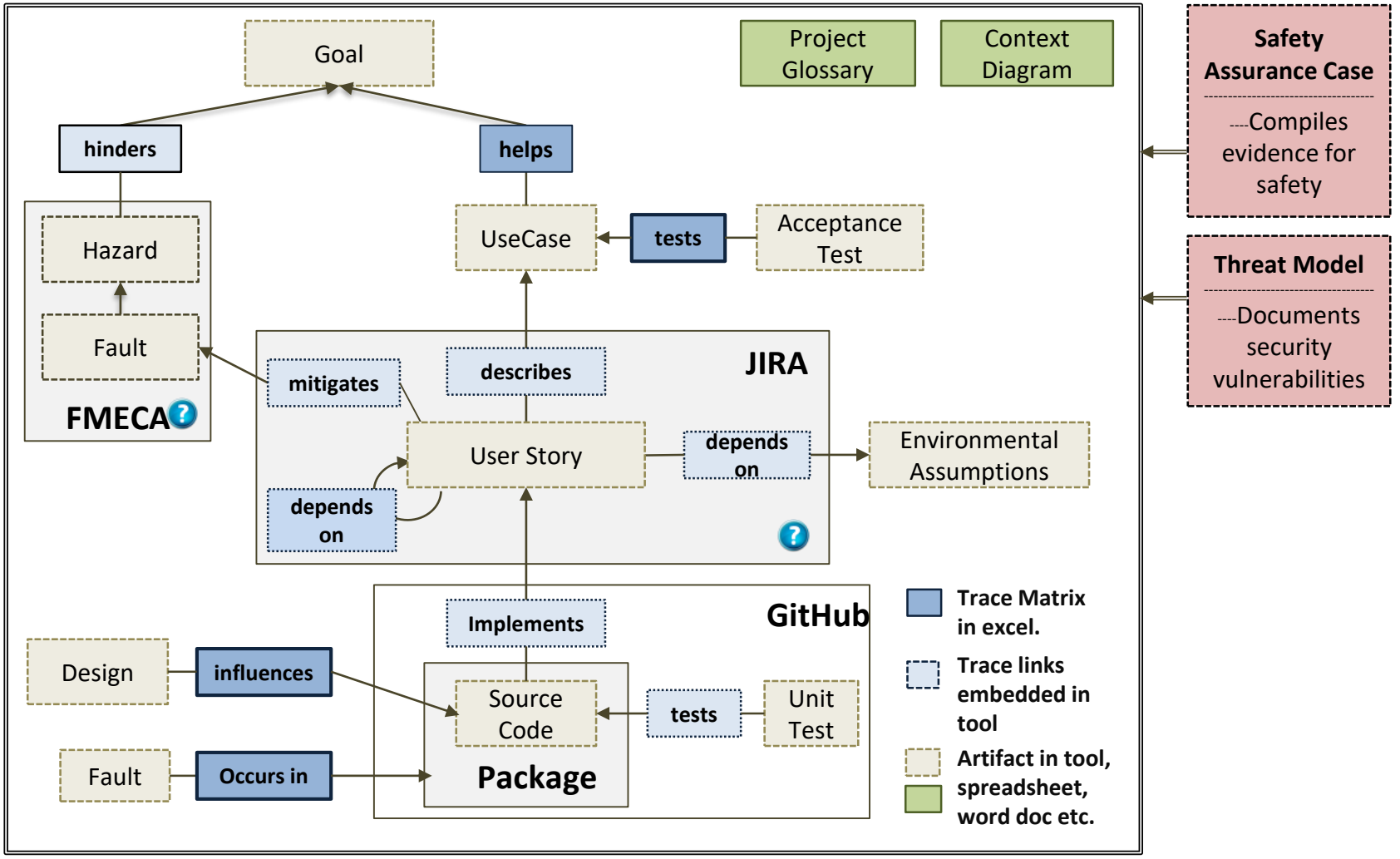
Safety Story: The drone must remain within 900 meters of a ground control station at all times.

Activity: Localization Related Faults

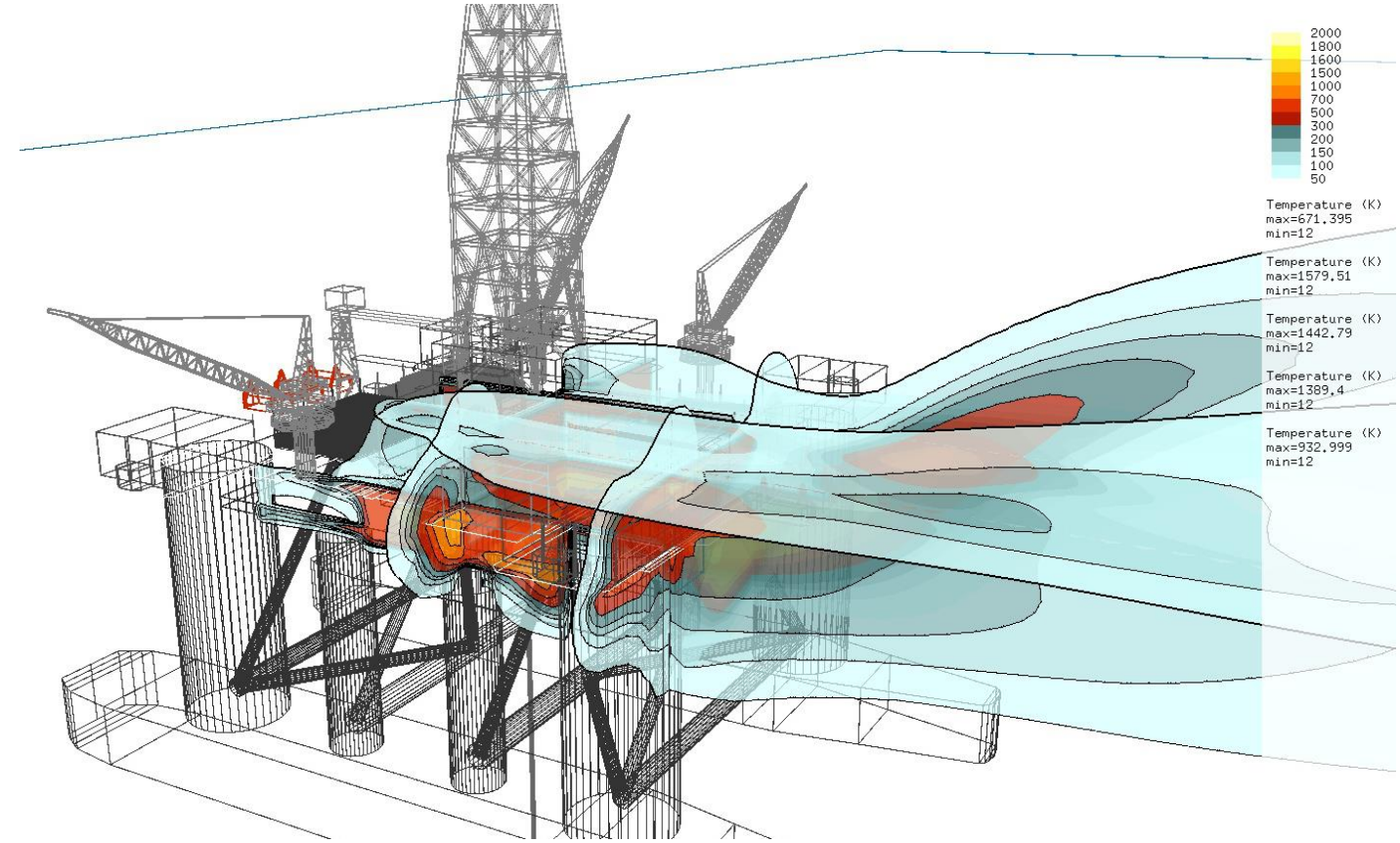
Can you think of Data and Event faults related to localization?

Assump.	Assumption Description	External System
A-6	The operating range (i.e. maximum distance that a drone can communicate with its base station) is 1 Km.	Telemetry
A-13	Communicable distance between the base station and the drone can be significantly reduced by environment factors such as radio interference.	Telemetry
A-20	The maximum flight time of the Iris 3DR drones is 22 minutes.	Iris 3DR
A-12	Iris 3DR GPS accuracy is within 8 meters.	GPS
A-29	The wing span of the 3DR drone is 55 cm.	Iris 3DR
A-21	The maximum airspeed of the Iris 3DR drone is 22 meters per second.	Iris 3DR
A-24	When a lock is acquired on 3 satellites, GPS accuracy will be within 5 meters.	GPS
A-40	The locator beacon can broadcast reliably over a distance of 400 feet.	Loc8tor beacon
A-41	The locator beacon is accurate to within one inch.	Loc8tor beacon

Making connections..



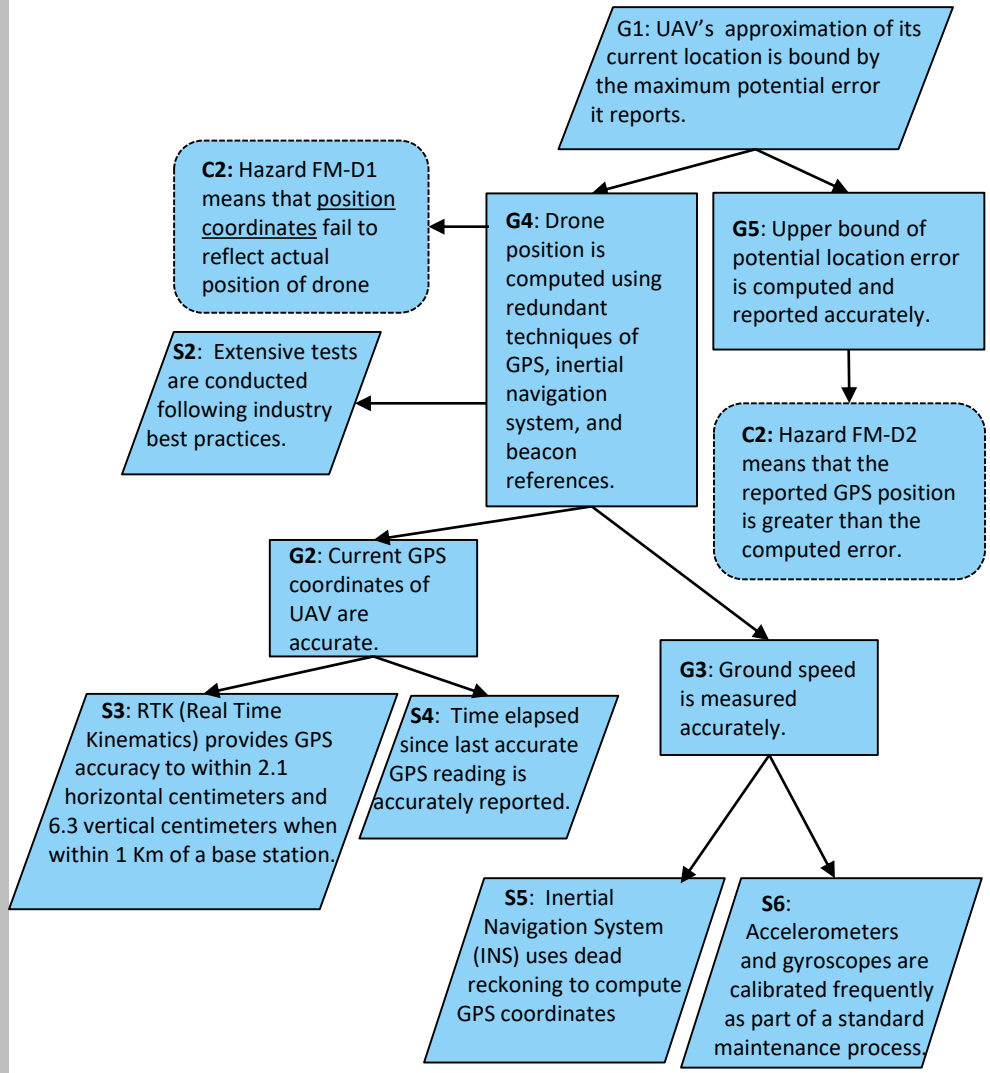
Build a Safety Argument



- Test cases
- Best Practices
- Field tests
- Simulations
- Formal Models
- Strategic safety functions
- Architect-ural tactics
- Regulatory compliance

Construct a systematic convincing argument, supported by evidence, to demonstrate that the system is safe for use.

Goal Structured Notation (GSN)



Fit for purpose?

Catastrophic: Failure may cause a crash.

Hazardous: Failure has a large negative impact on safety or performance, … or causes serious or fatal injuries among the passengers.

Major: Failure is significant, but has a lesser impact such as discomfort.

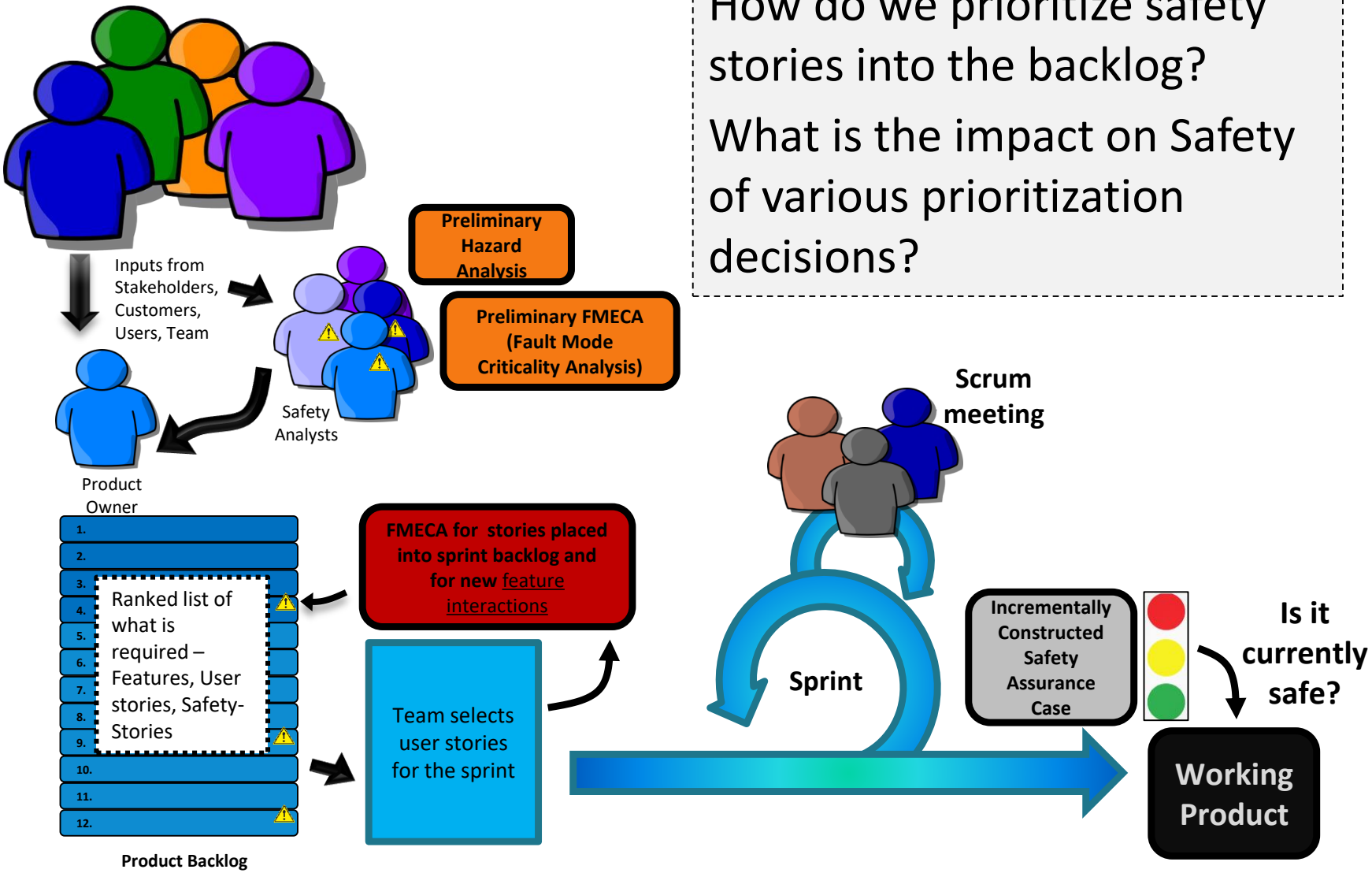
Minor: Failure is noticeable but not dangerous.

No Effect: Failure has no impact on safety

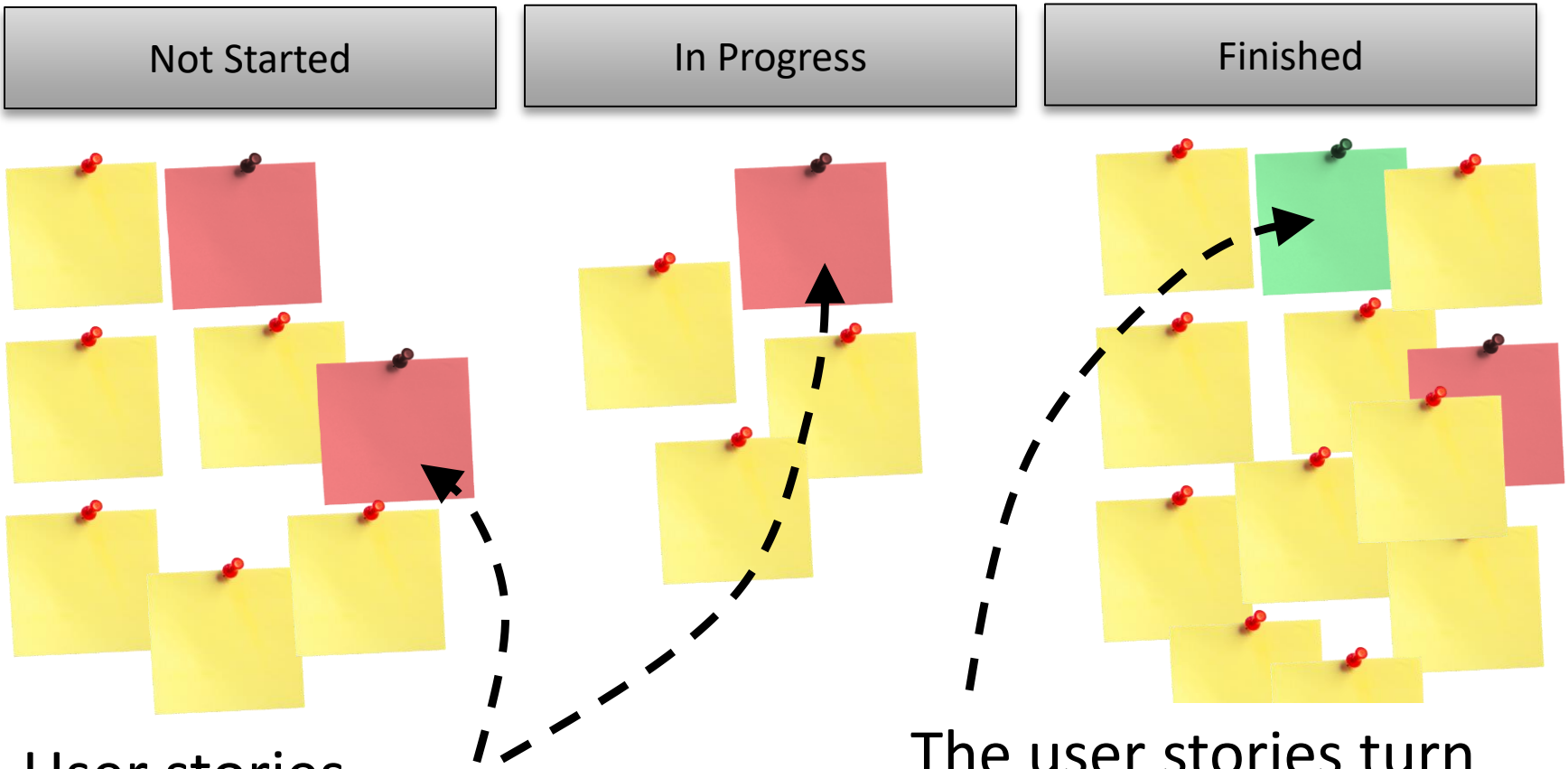
Construct the FMECA incrementally

How do we prioritize safety stories into the backlog?

What is the impact on Safety of various prioritization decisions?



Carefully Track Safety Dependencies

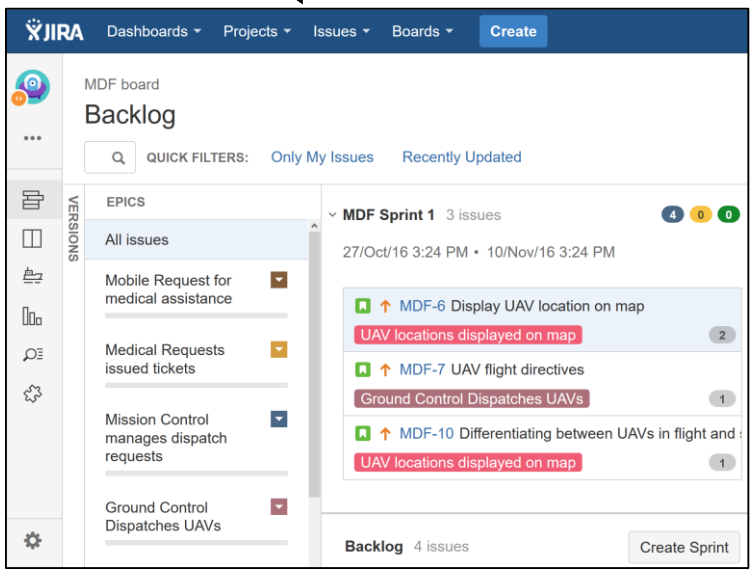


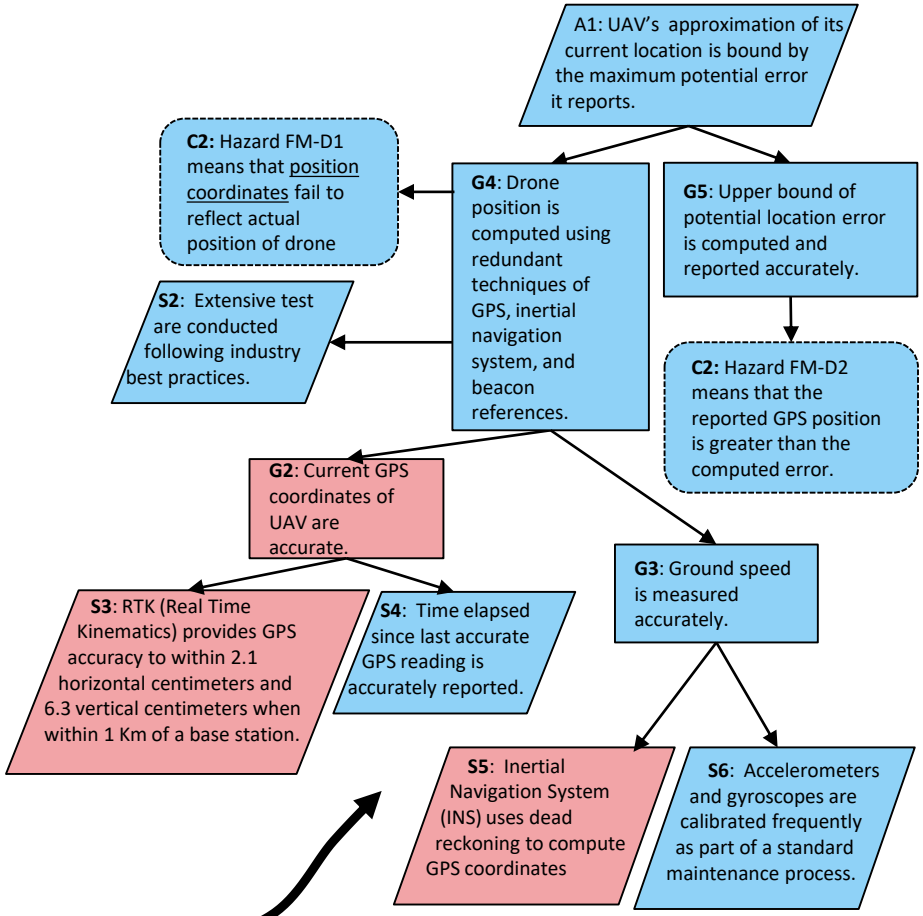
User stories with safety dependencies are tracked as they move through the system.

The user stories turn “green” once all safety-dependencies are completed.

Goal – to build a demonstrably safe system

ID	Fault	Description	Consequence	Level
FM-D1	Faulty data	GPS coordinates do not reflect actual position of drone.	Drones fail to maintain minimum separation distance, crash in midair, and debris falls to the ground.	Critical
FM-E1	Faulty Algo-rithm	Drone's route is miscalculated	Drone flies into prohibited airspace	Catas-trophic





In Closing

Drones	E-Health	Environment	Mission-Control
			

- Safety Critical Systems are different.
- Think about the degree of criticality.
- Incrementally identify potential hazards and faults – and define safety stories to mitigate them.
- Track user stories with safety dependencies through the system so that you always understand the degree to which safety has been achieved in each end-of-sprint deployment.

Agile Goes Critical

Jane Cleland-Huang, PhD
University of Notre Dame
Department of Computer Science and Engineering
JaneClelandHuang@nd.edu
sarec.nd.edu

Some of the ideas described in this talk emerged as a result of projects funded by the US National Science Foundation under Grants CCF-0959924 and CCF-1265178.