AGILE STORY PYRAMIDS

Minneapolis Dojo: April 18th 2019

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

Some of the work described in this talk was funded by the US National Science Foundation under Grants CCF-0959924 and CCF-1265178.

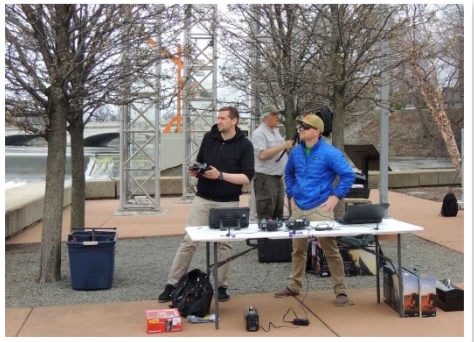
About our Domain

http://sarec.nd.edu/pages/Dronology.html

A platform for coordinating the flight of UAVs for use in emergency response scenarios.

River Rescue Demo with Dronology

Deployment of drones for emergency response introduces safety concerns that we needed to address.



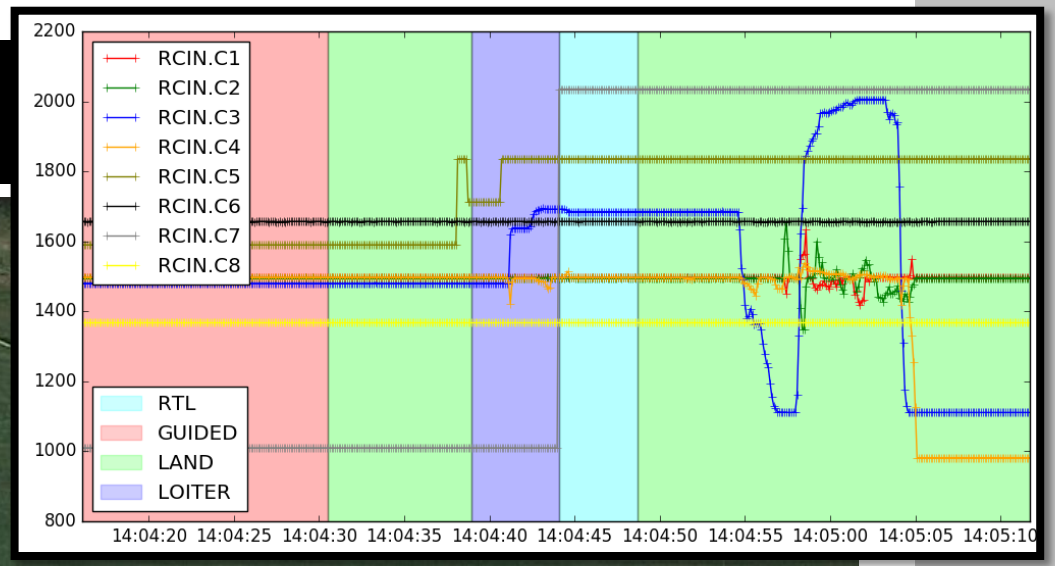
Testing an AED drop

Drones delivering medical supplies must fly far beyond line of sight, circumvent obstacles and changing terrain, fly over urban areas, & deliver heavy payloads in potentially populous regions.

Accidents happen

We focus significant attention on hazard analysis and subsequent safety stories.

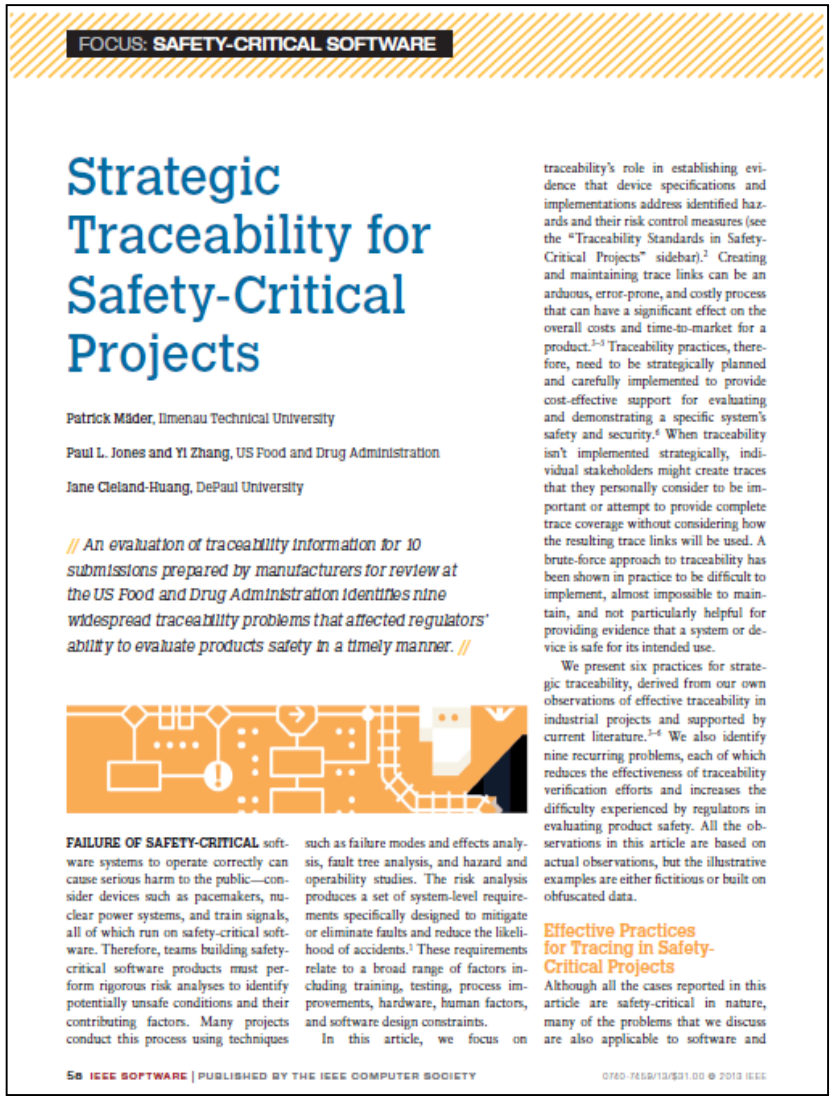
Safety Story (SAF-1):
The GPS coordinates of each UAV must be accurate within three meter at all times.



Our Challenge

- **Continue** to follow the agile approach with all the benefits it brings to our project.
- Imbue the process with **the right amount** and the **right kinds** of safety analysis.
- Merge Safety Assurance into the agile life cycle.

Typically prescribed solutions fall short



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

// An evaluation of traceability information for 10 submissions prepared by manufacturers for review at the US Food and Drug Administration identifies nine widespread traceability problems that affected regulators' ability to evaluate products safety in a timely manner. //

FAILURE OF SAFETY-CRITICAL software systems to operate correctly can cause serious harm to the public—consider devices such as pacemakers, nuclear power systems, and train signals, all of which run on safety-critical software. Therefore, teams building safety-critical software products must perform rigorous risk analyses to identify potentially unsafe conditions and their contributing factors. Many projects conduct this process using techniques such as failure modes and effects analysis, fault tree analysis, and hazard and operability studies. The risk analysis produces a set of system-level requirements specifically designed to mitigate or eliminate faults and reduce the likelihood of accidents.[1] These requirements relate to a broad range of factors including training, testing, process improvements, hardware, human factors, and software design constraints.

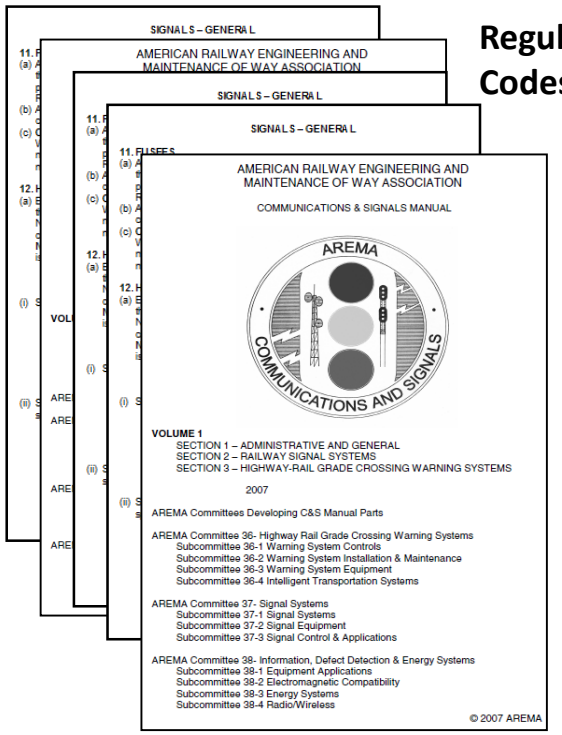
In this article, we focus on traceability's role in establishing evidence that device specifications and implementations address identified hazards and their risk control measures (see the "Traceability Standards in Safety-Critical Projects" sidebar).[2] Creating and maintaining trace links can be an arduous, error-prone, and costly process that can have a significant effect on the overall costs and time-to-market for a product.[3–5] Traceability practices, therefore, need to be strategically planned and carefully implemented to provide cost-effective support for evaluating and demonstrating a specific system's safety and security.[6] When traceability isn't implemented strategically, individual stakeholders might create traces that they personally consider to be important or attempt to provide complete trace coverage without considering how the resulting trace links will be used. A brute-force approach to traceability has been shown in practice to be difficult to implement, almost impossible to maintain, and not particularly helpful for providing evidence that a system or device is safe for its intended use.

We present six practices for strategic traceability, derived from our own observations of effective traceability in industrial projects and supported by current literature.[3–6] We also identify nine recurring problems, each of which reduces the effectiveness of traceability verification efforts and increases the difficulty experienced by regulators in evaluating product safety. All the observations in this article are based on actual observations, but the illustrative examples are either fictitious or built on obfuscated data.

Effective Practices for Tracing in Safety-Critical Projects

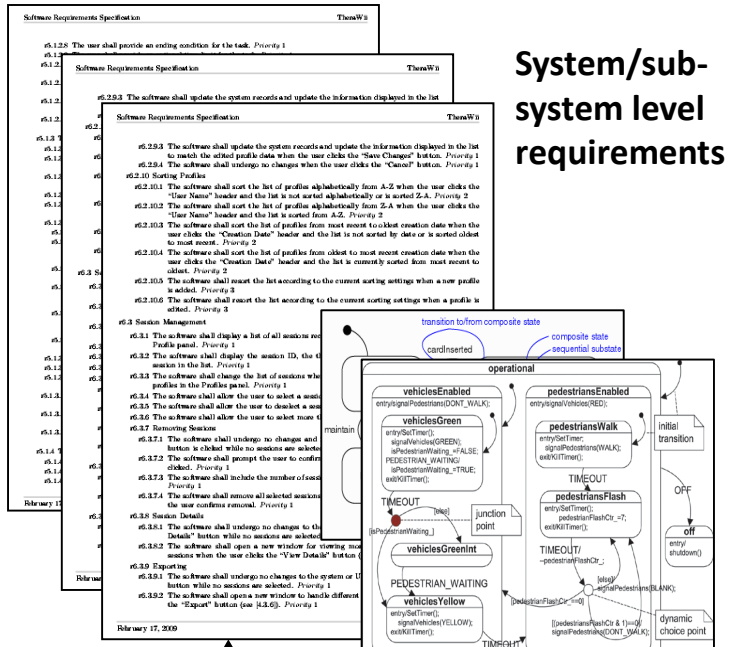
Although all the cases reported in this article are safety-critical in nature, many of the problems that we discuss are also applicable to software and

0740-7459/13/$31.00 © 2013 IEEE



Regulatory Codes

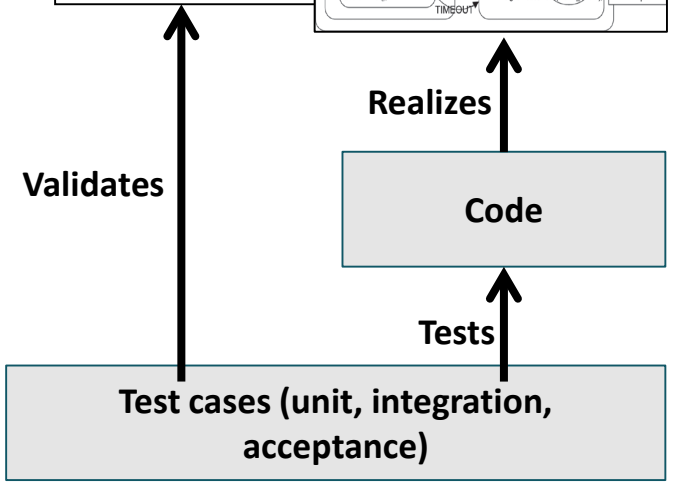
Satisfies relevant codes



System/sub-system level requirements

Required by many regulatory bodies.

Hard to achieve in practice → Traceability Gap.



Agility and Safety Critical Process Converged in Dronology

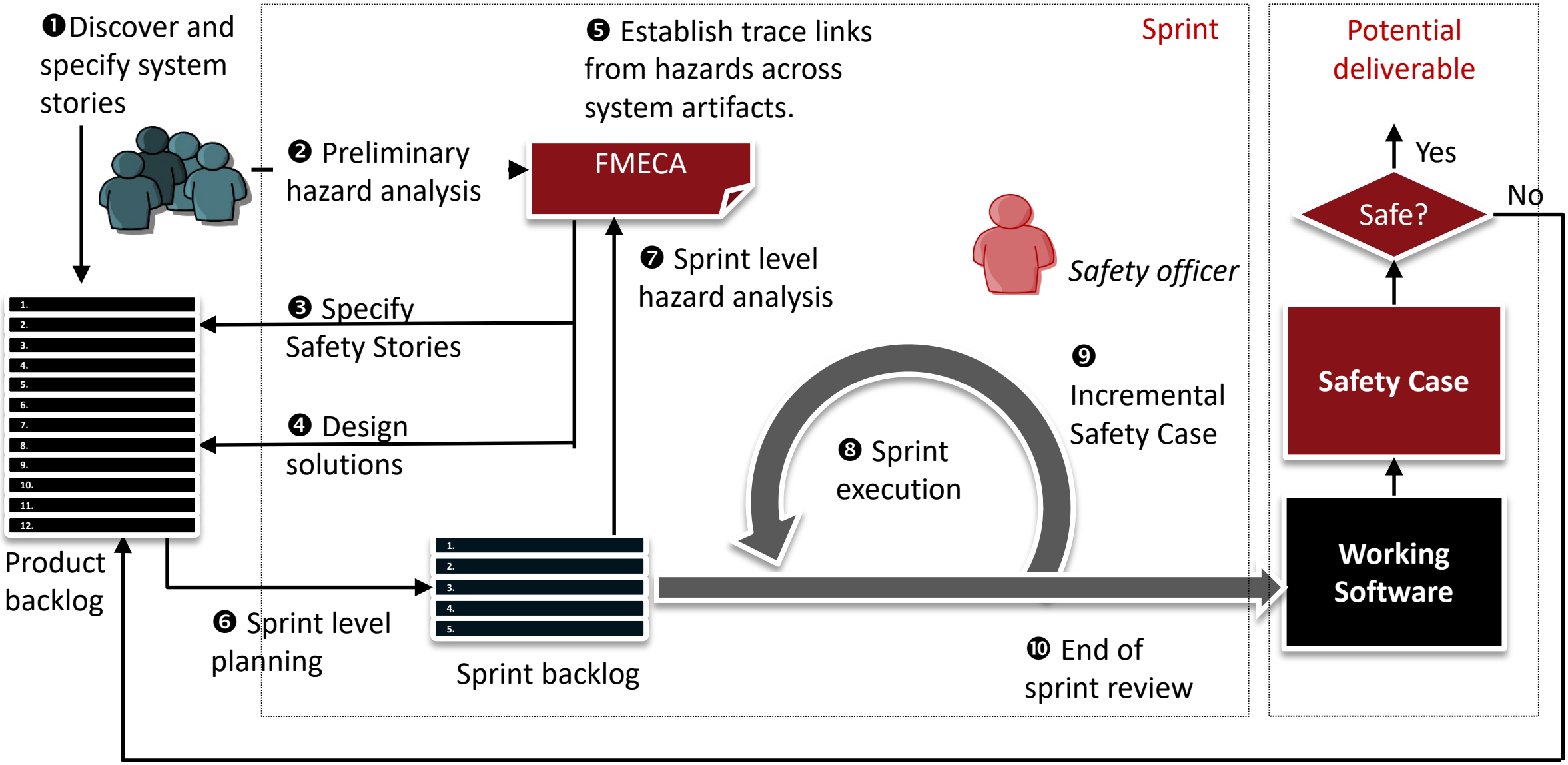
❶ SCRUM provided insufficient support for systematic safety.

❷ Many safety concerns emerged alongside the functionality. Surveys supported this.

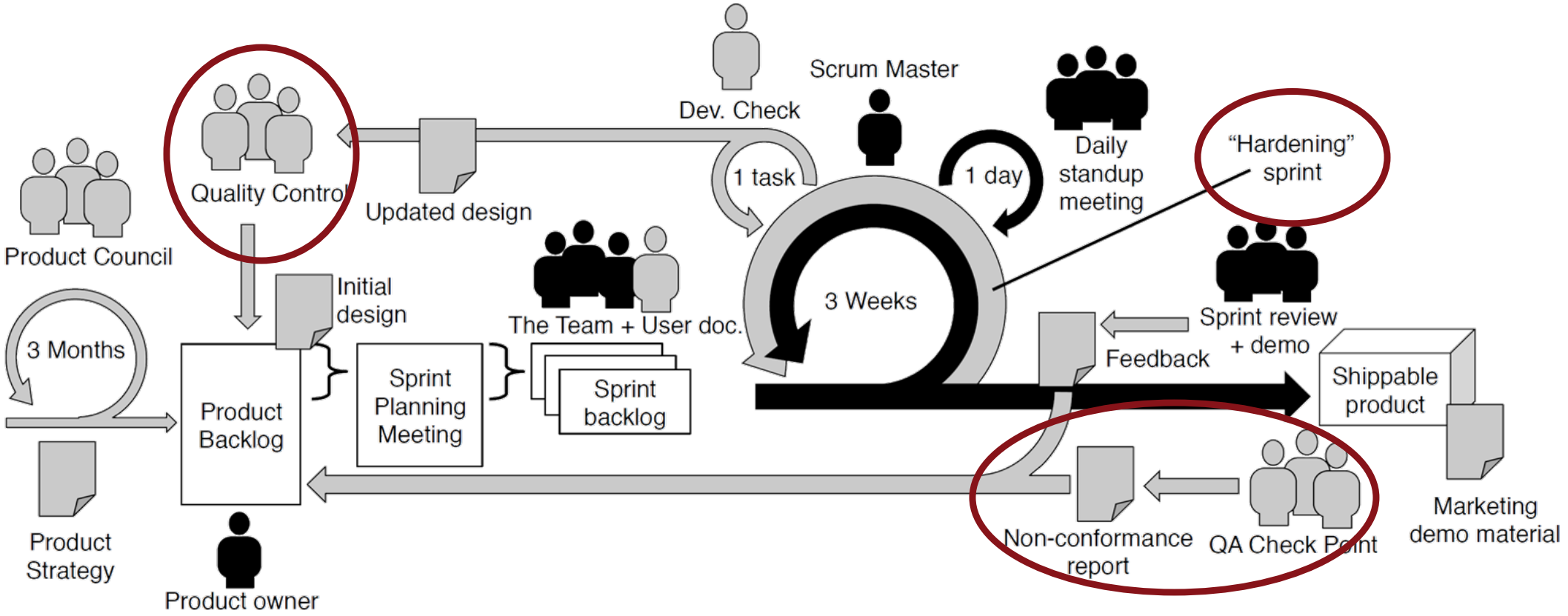
❸ Building an arguably safe system (i.e., through a safety case), was incredibly time consuming.

❹ Instrumenting and tooling the agile environment was essential to deliver living, ubiquitous traceability.

Safety Scrum (SScrum)



QUMAS: RScrum



Strong internal quality management & culture. All development sprints audited by QA.

User stories assigned risk factors, and risk managed proactively.

Living traceability has huge impact esp. on compliance.

Hardening sprint to prep system for release.

Incremental Hazard Analysis

Switch in wrong position

EARS (Easy Requirements Specification)

To write functional and safety stories for Cyber-Physical Systems.

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

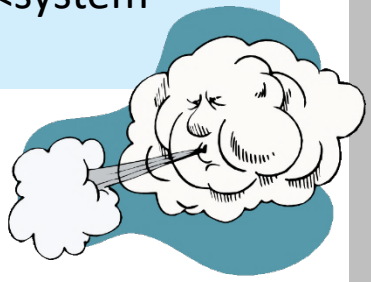
Where <feature is included> the <system name> shall <system response>

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

Unwanted Behavior

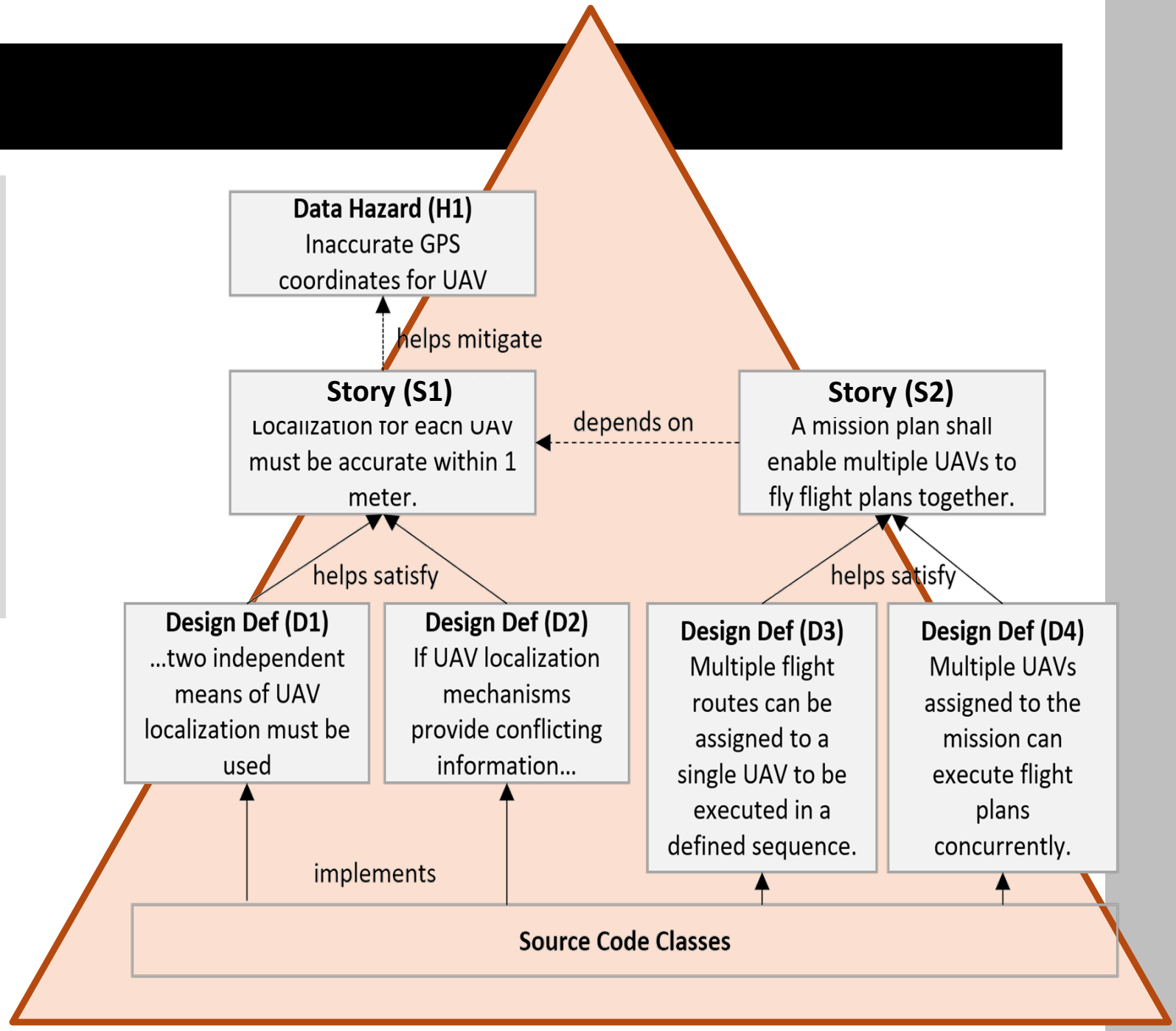
If <optional preconditions> <trigger>, then the <system name> shall <system response>

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

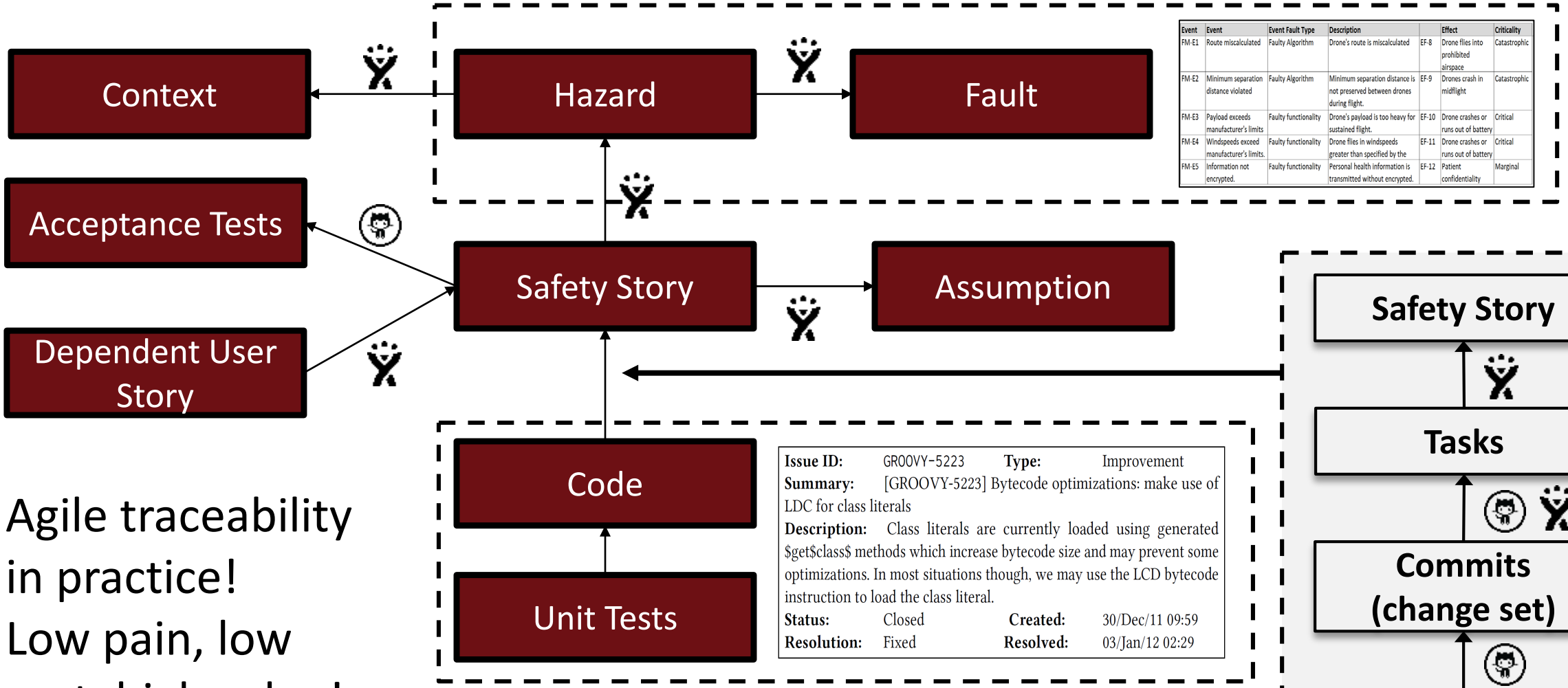


Build a Pinnacle

Start with the pinnacle and add artifacts to show how it was addressed.



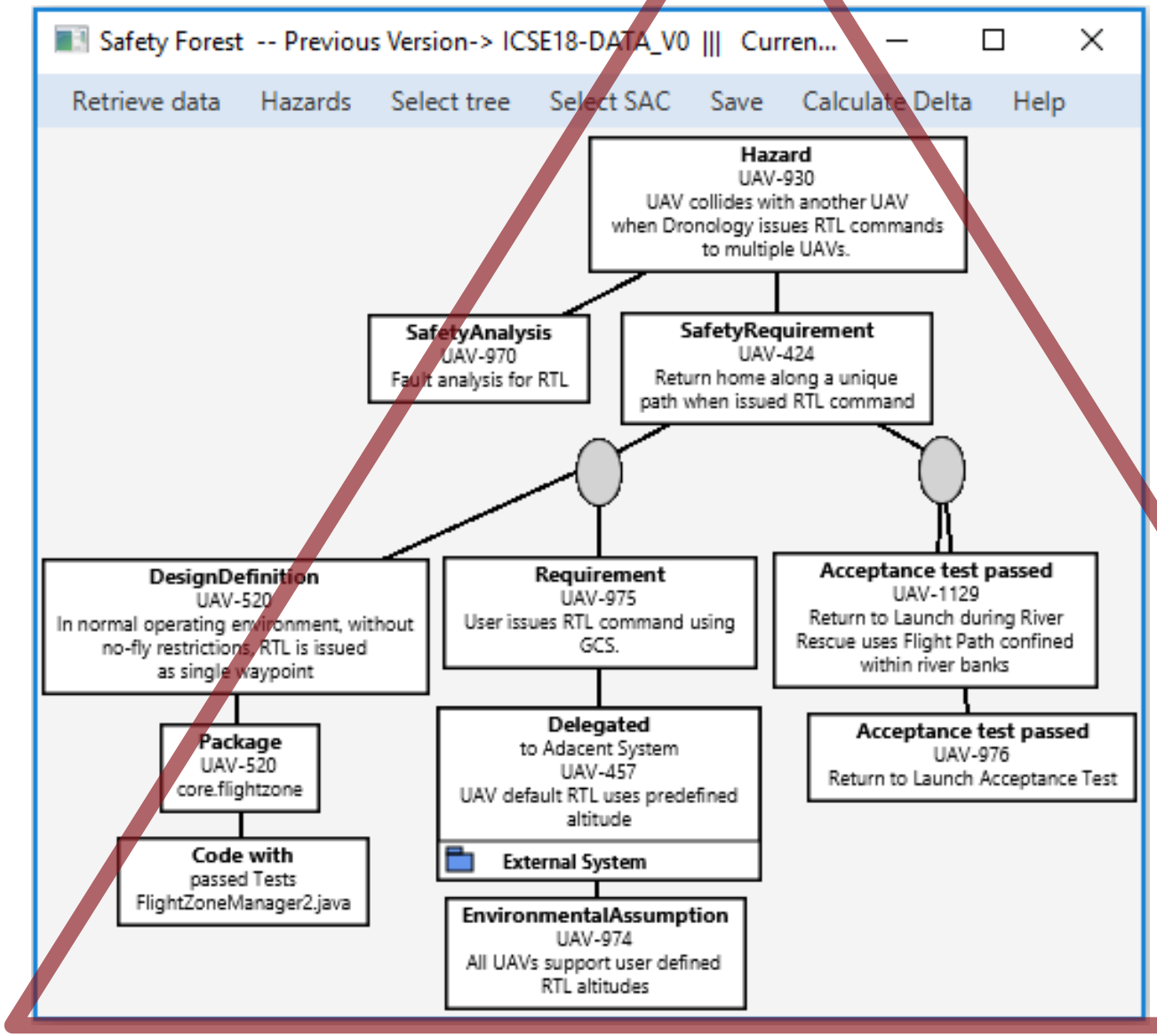
Structure it



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	Critical
FM-E4	Windspeeds exceed manufacturer's limits.	Faulty functionality	Drone flies in windspeeds greater than specified by the	EF-11	Drone crashes or runs out of battery	Critical
FM-E5	Information not encrypted.	Faulty functionality	Personal health information is transmitted without encrypted.	EF-12	Patient confidentiality	Marginal

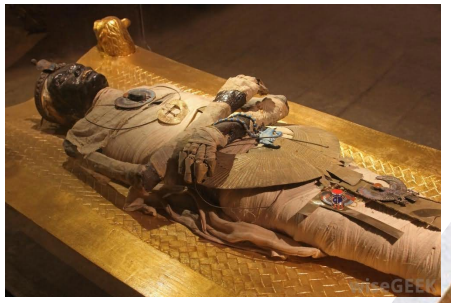
Agile traceability in practice!
Low pain, low cost, high value!

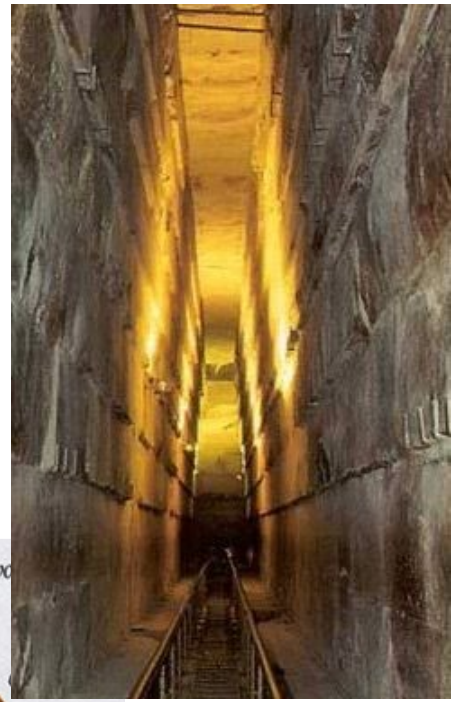
Generate it!



- The pyramid can be generated automatically from Jira, GitHub and other repos.

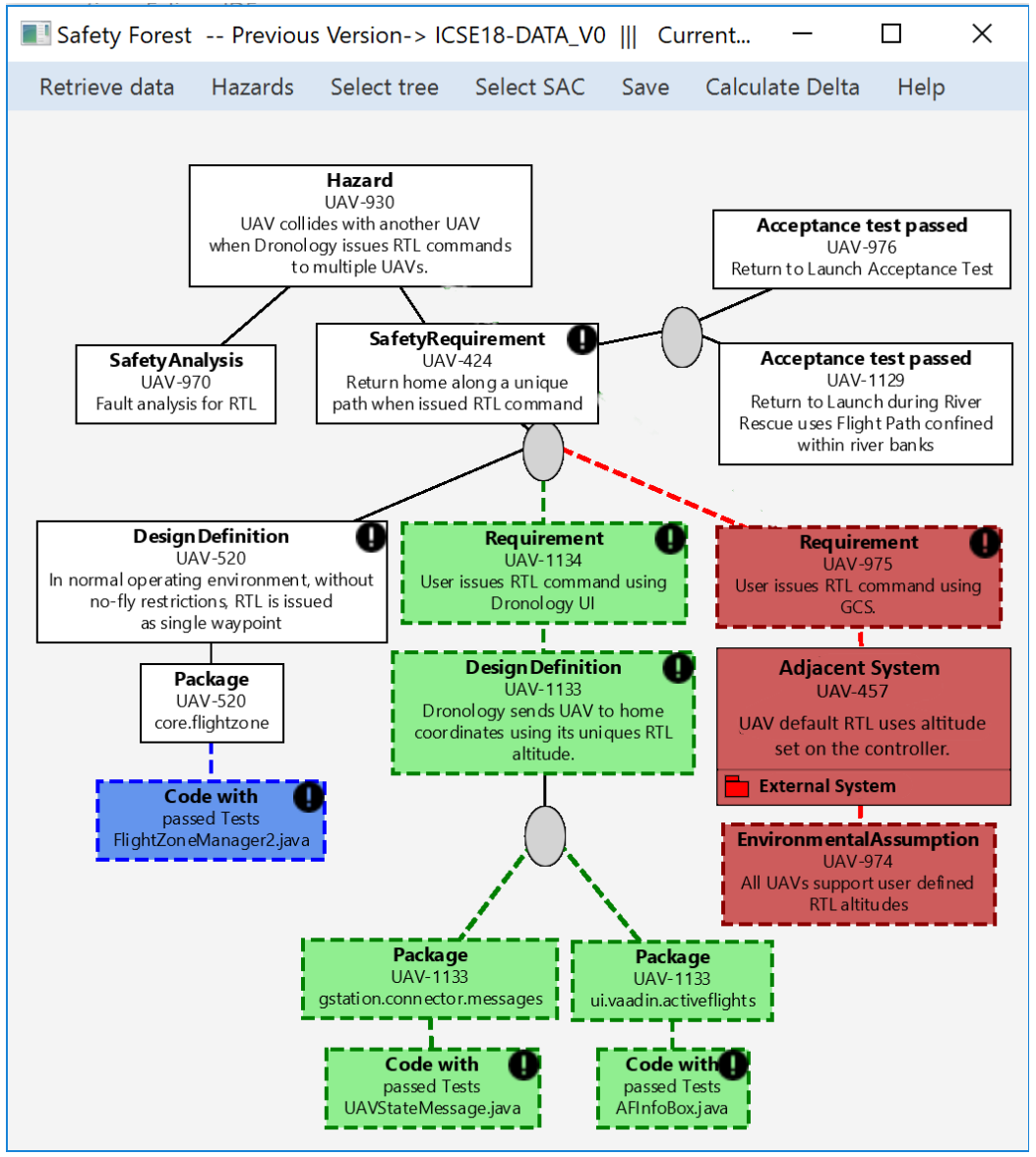
The Pyramid Metaphor





- Pyramids convey legends.
- Let's add legends to our agile toolkit.
- What value do they bring to your project?

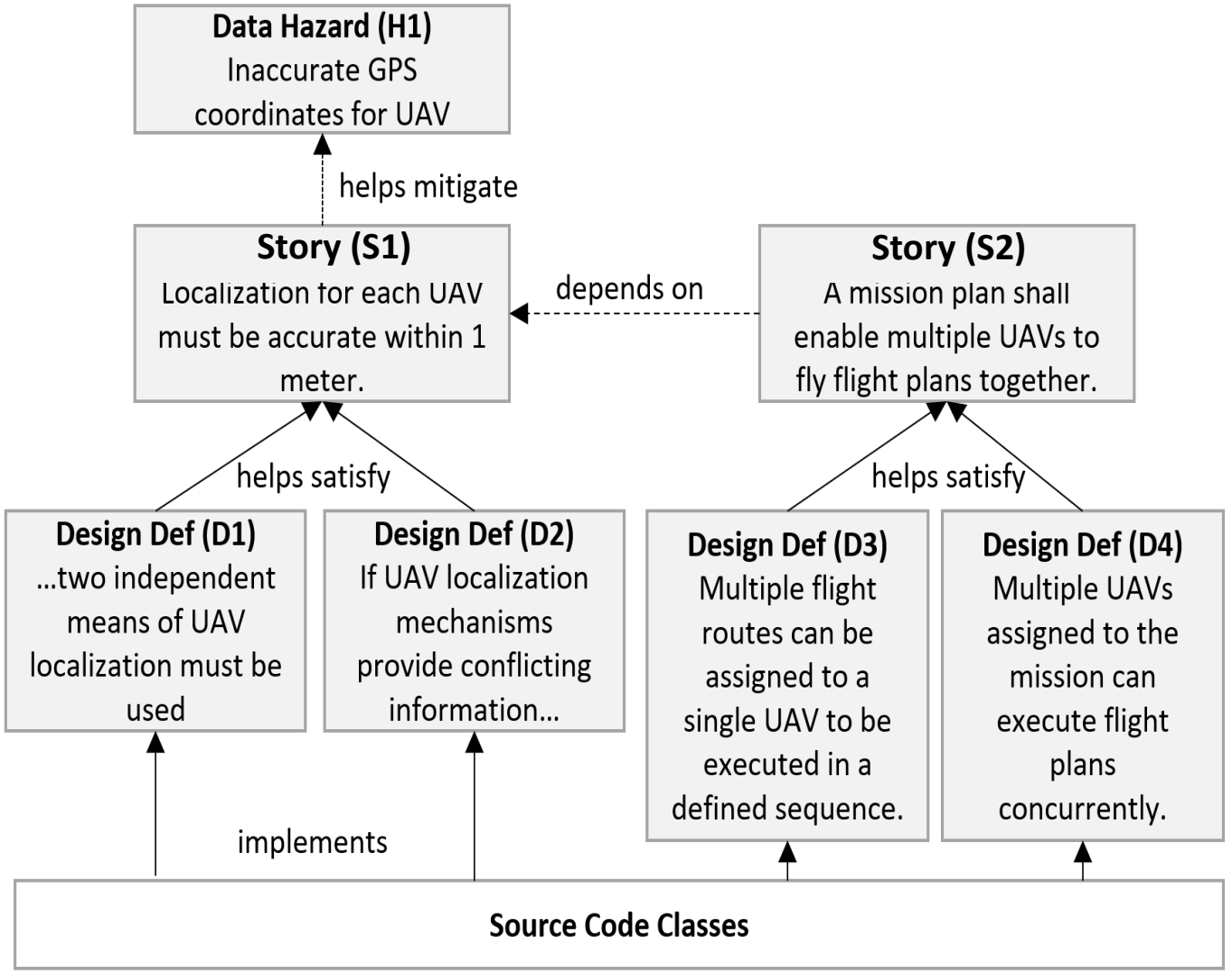
1. Delta Analysis



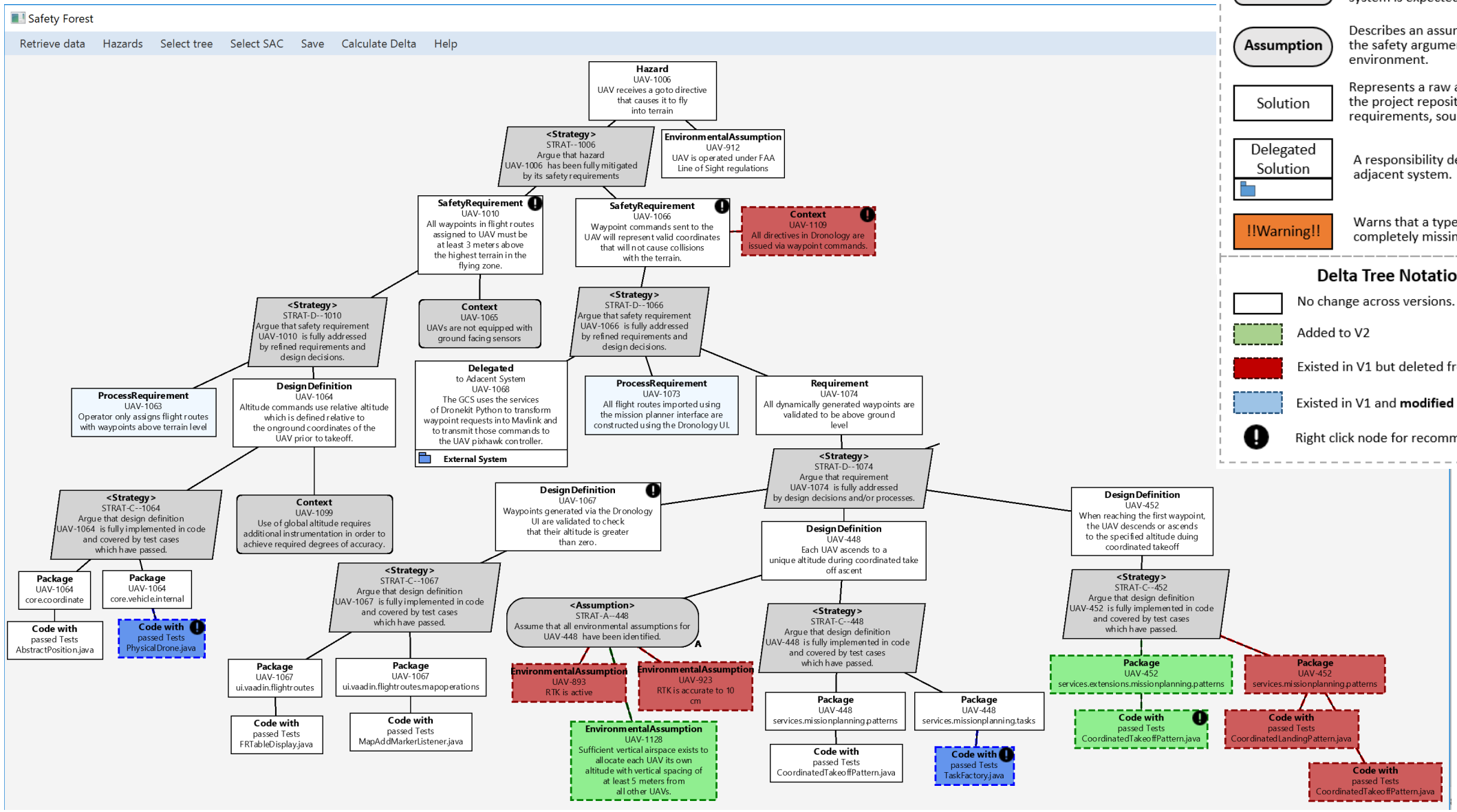
- We can compare two versions of the pyramid to instantly see what has changed over time.
  - Inform developers.
  - Build ‘safety’ cases.

2. Smart iteration planning

During development we track safety story dependencies so that we can prioritize user stories correctly in each release.



3. Safety Case Generation



4. Regulatory Support

FAA Waiver	Explanation	ND	SB
§ 107.25 – Operation from a Moving Vehicle ..	Fly a UAS from a moving aircraft .. in populated areas		
§ 107.29 – Daylight Operations	Fly a UAS at night		1 2
§ 107.31 – Visual Line of Sight Aircraft Operation	Fly a UAS beyond your ability to clearly determine its orientation with unaided vision	4	
§ 107.33 – Visual Observer	User a visual observer without following all visual observer requirements		
§ 107.35 – Operation of Multiple Small UAS	Fly multiple UAS with only 1 remote pilot	3	
§ 107.37(a) – Yielding Right of Way	Fly a UAS without having to give way to other aircraft		
§ 107.39 – Operation Over People	Fly a UAS over a person/people		2
§ 107.51 – Operating limitations for Small UAS	Fly a UAS … fast, high, with low visibility, or close to clouds		
Jurisdictional COA to operate in Class C airspace at or below altitudes depicted in the UASFM for the Class C Surface Area in the vicinity South Bend International Airport ATCT (SBN)			2

Granted **Applied & currently under review** **Under preparation with lawyer**

Practitioners like it

To what extent does SAFA's Delta View support an analyst in identifying changes which potentially impact the safety of a new version of the system?

Version 1 (v1)
49,400 LOC
418 Java Classes
146 Requirements
224 Design definitions

Version 2 (v2)
73,591 LOC
646 Java classes
185 Requirements
283 Design definitions

Each participant was given six hazards to evaluate using two treatments.

T1: View artifact trees for v1 and v2
T2: View delta tree

ID	Role	Domain	Yrs	SC
P1	Software Engineer	Aviation & Defense	8	Y
P2	Developer	Operating Systems	1	N
P3	Developer	Development	2+	Y
P4	Developer	Embedded Systems	1	N
P5	Developer	Embedded Systems	2	Y
P6	Software Engineer	Software Development	7	Y
P7	Developer	Information Systems	2	N
P8	Software Engineer	Unmanned Aerial Systems	1	Y
P9	Systems Engineer	Embedded Systems	35	Y
P10	Requirements Engineer	Defense Systems	23	Y

I would kill to have SAFA in my workplace

Give me back the delta view!

I find myself implicitly trusting the tool. Is the tool certified?

What else can we use Pyramids for?

Safety
Building a safety case

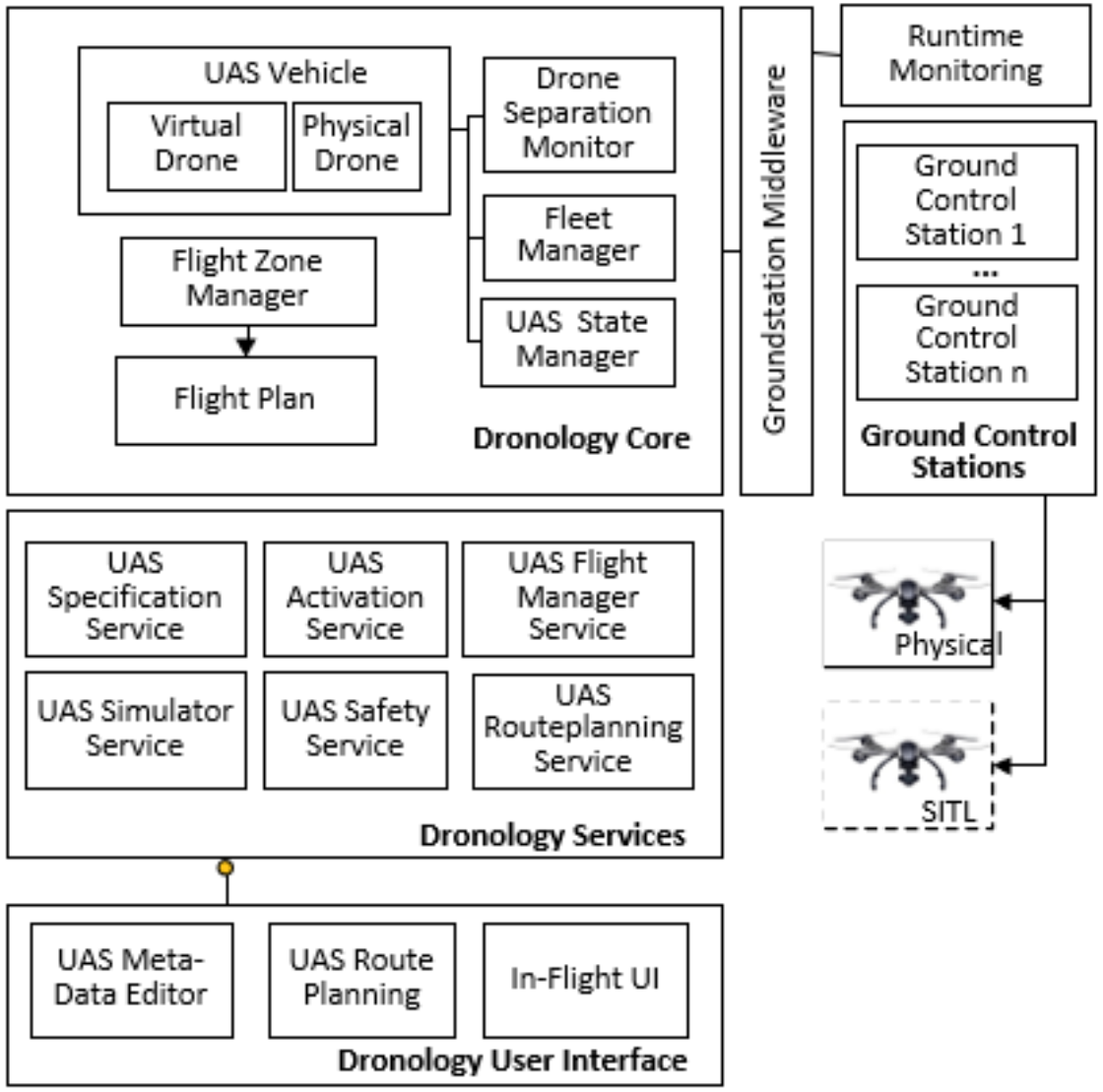
Architectural Preservation
Your system is maintained by hundreds+ developers and you need to preserve design knowledge.

Regulatory Compliance
Any external regulations you need to comply to.

Privacy

Technical Debt

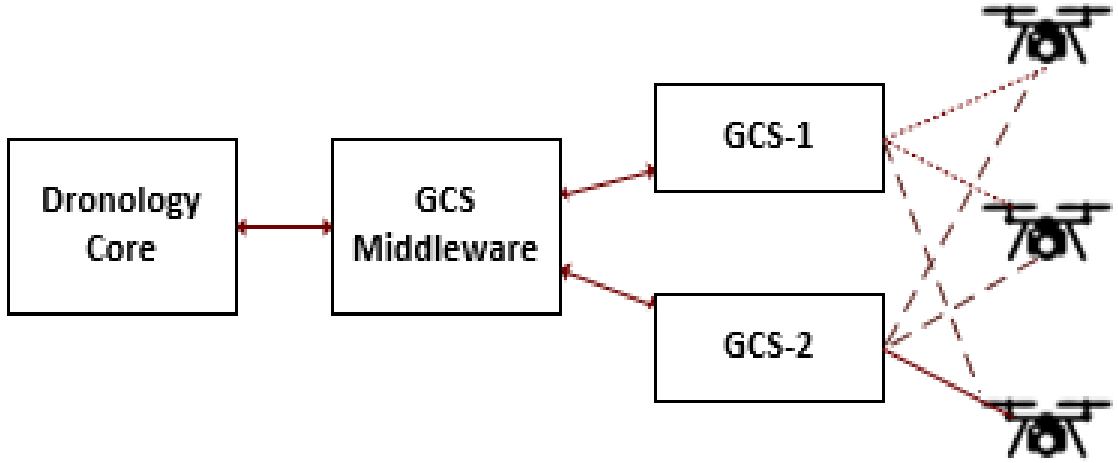
Another Example



- Architectural decisions need to be preserved in order to maintain system qualities.
- Any changes to underlying architectural decisions should be well informed.

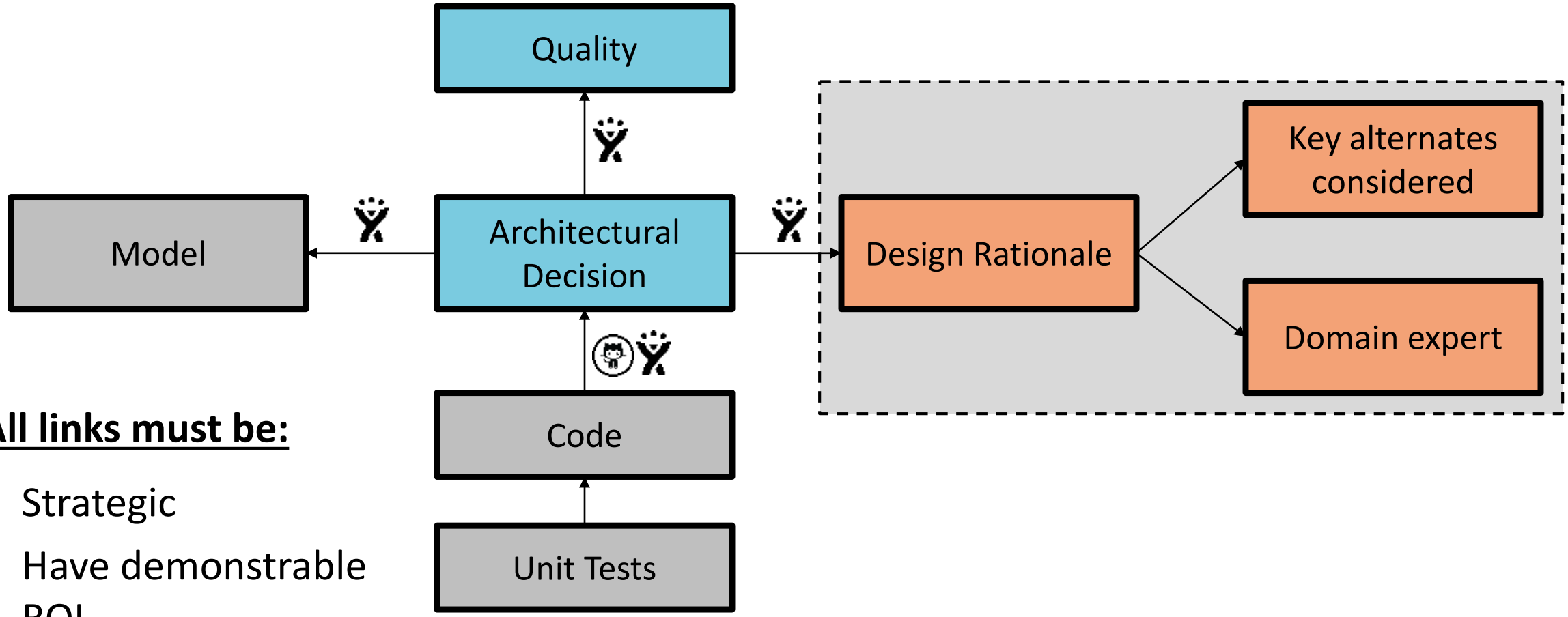
Architectural Decision

When a UAV is registered with Dronology and in active flight, at least one Ground Control Station must be available 99% of the time



- Redundant GCS
- GCS emits heartbeat.
- Monitored by Dronology
- UAV swaps GCS in flight.
- Flights suspended when < 2 GCS are active.
- Cold-replacement of GCS.

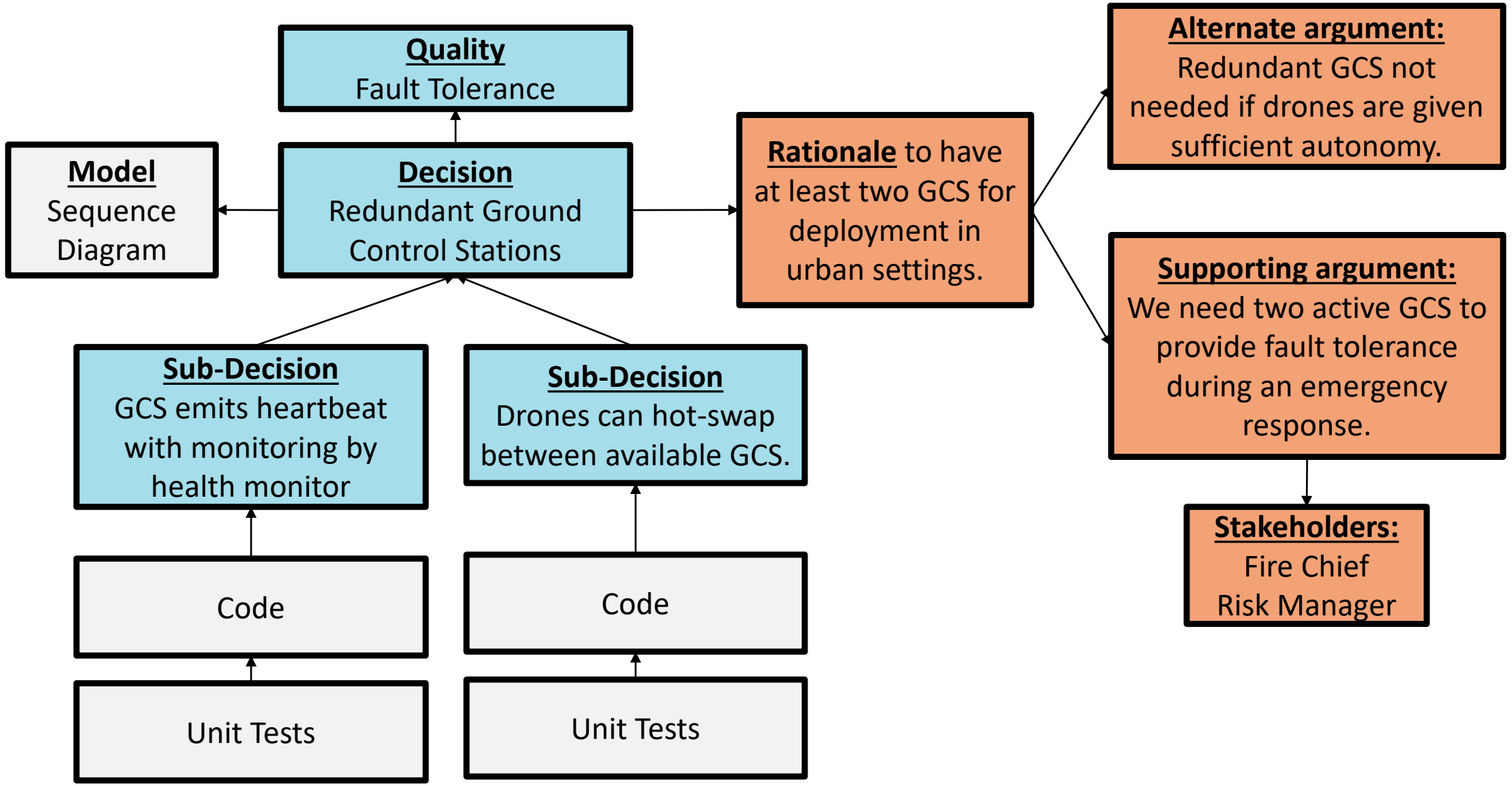
Pyramid Structure



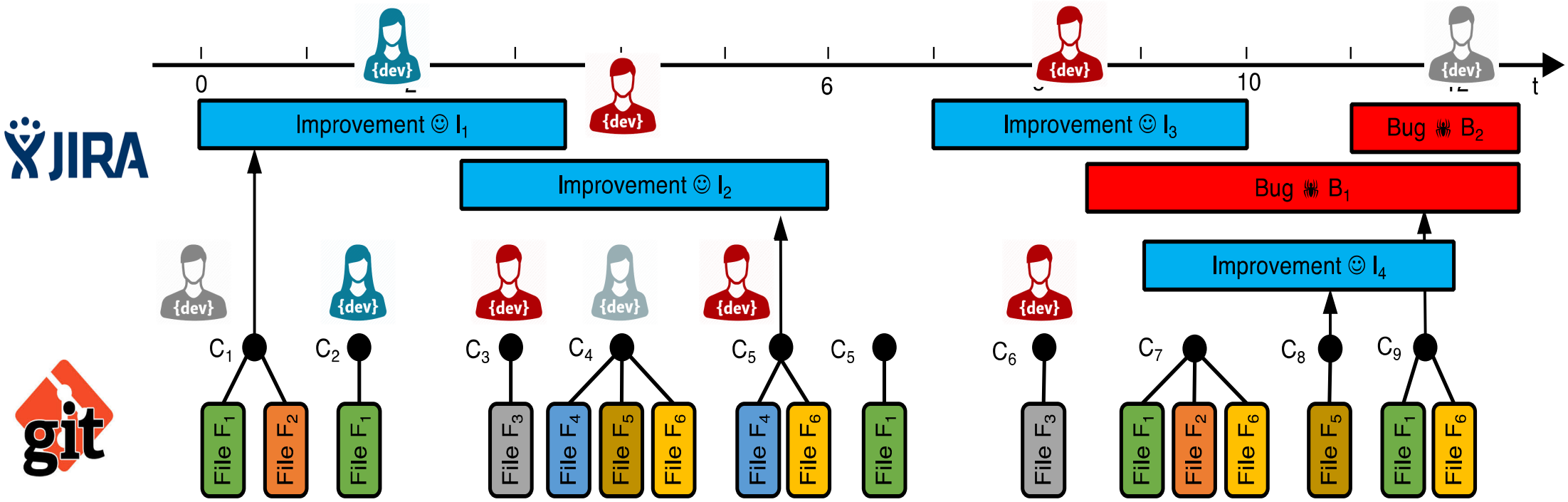
All links must be:

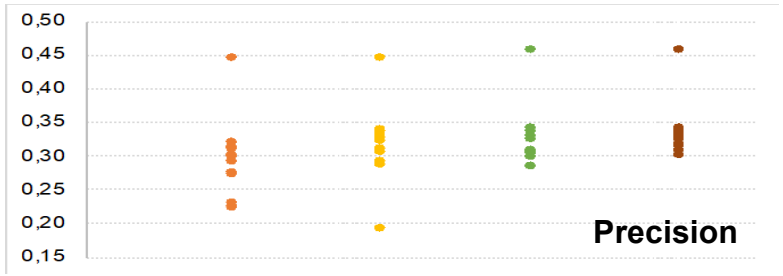
- Strategic
- Have demonstrable ROI
- Automated where possible

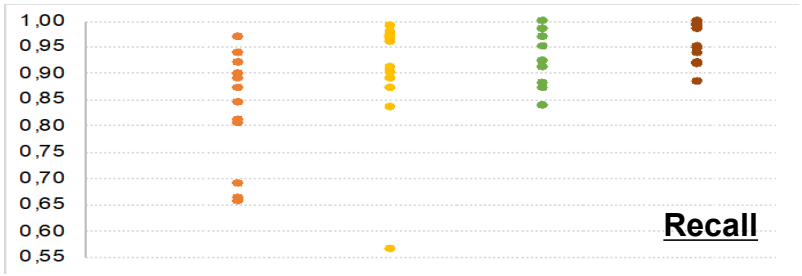
Instantiated…



Can we really automate the links?



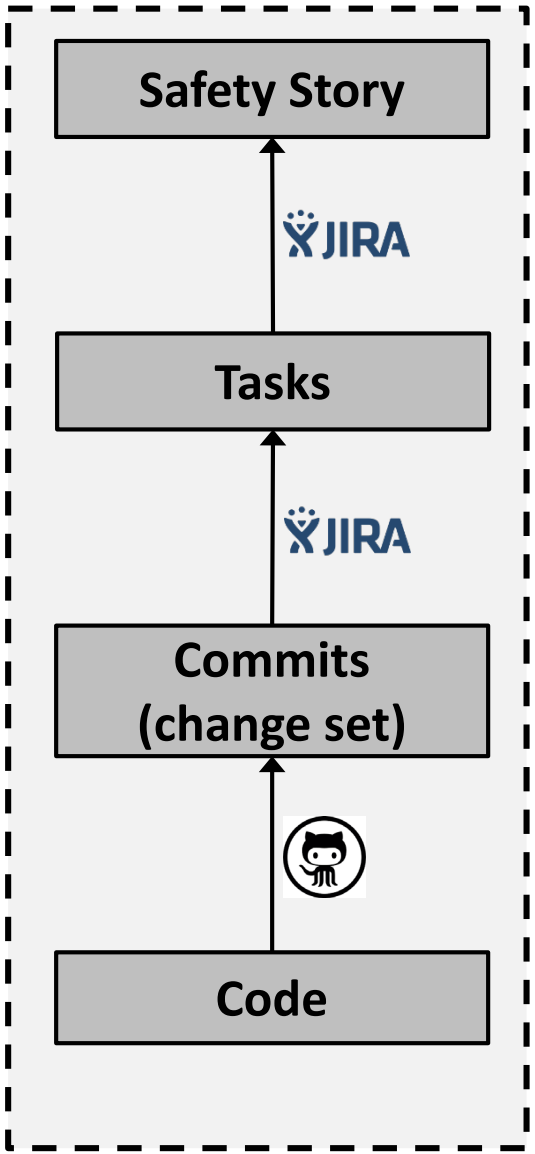


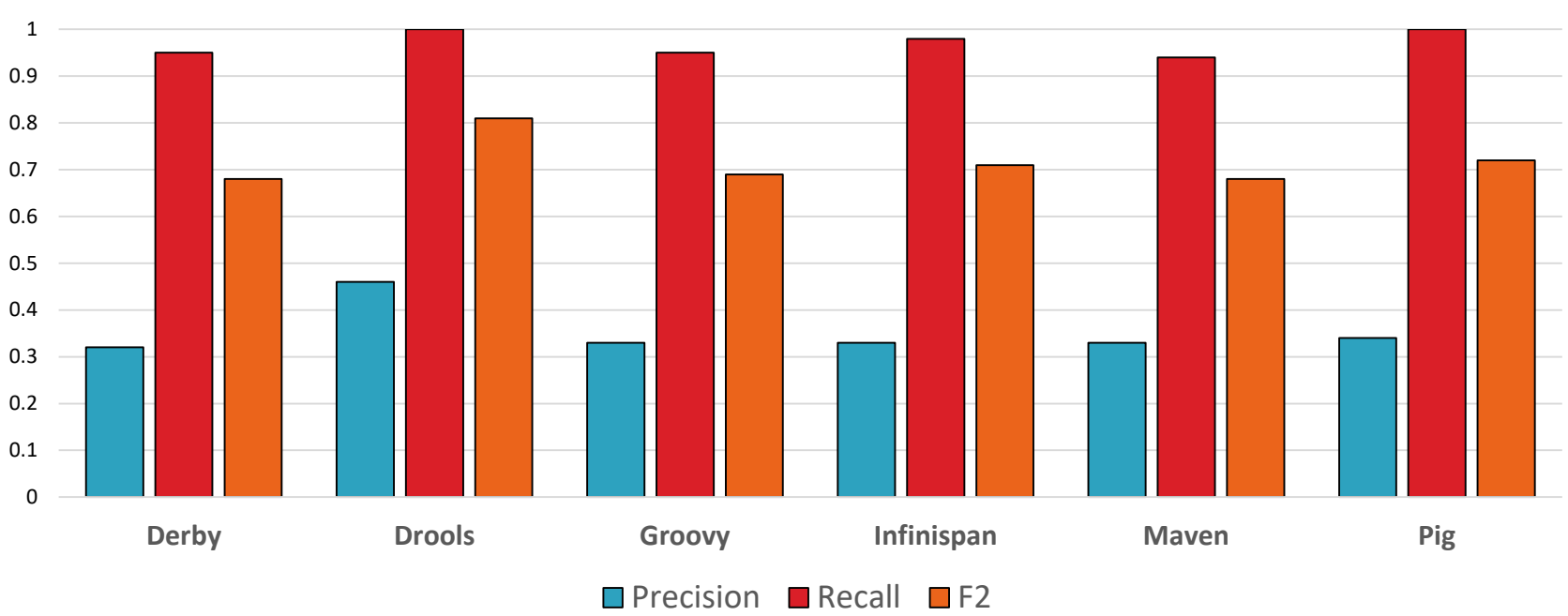


Traceability in the wild: automatically augmenting incomplete trace links.
Michael Rath, Jacob Rendall, Jin L. C. Guo, Jane Cleland-Huang, Patrick Mäder:
ICSE 2018: 834-845

- Mining project repositories
- Temporal properties
- Machine learning
- Project exhaust!

Jira-Git Commit Link Inferencing





		Project					
Link Type	Profile	De	Dr	Gr	In	Ma	Pi
1:1	Bug	1093	1040	1609	1714	609	1066
	Imp.	399	61	459	393	222	313
1:n	Bug	273	236	422	254	99	87
	Imp.	225	28	179	96	46	48
Non	Bug	1272	605	1667	994	769	944
	Imp.	730	42	392	157	313	251

Pyramid Building

1. Decide what type of pyramid you might be interested in.
   - Why might it be important to you?
2. Sketch out the details of the pyramid structure.
   - How much effort would it take to build vs. benefits?
   - How would the links be created/maintained?
3. Build your pyramid as a byproduct of your development effort.


Pyramids tell us
stories from the past
….. legends

Thoughts?

Safety
Building a safety case

Architectural Preservation
Your system is maintained by hundreds+ developers and you need to preserve design knowledge.

Regulatory Compliance
Any external regulations you need to comply to.

Privacy

Technical Debt

PYRAMIDS: THE SOURCE OF LEGENDS

Minneapolis Dojo: April 18th 2019

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

Some of the work described in this talk was funded by the US National Science Foundation under Grants CCF-0959924 and CCF-1265178.