About our Domain

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

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

Required by many regulatory bodies.

Hard to achieve in practice \(\rightarrow\) Traceability Gap.
Agility and Safety Critical Process Converged in Dronology

1. SCRUM provided insufficient support for systematic safety.
2. Many safety concerns emerged alongside the functionality. Surveys supported this.
3. Building an arguably safe system (i.e., through a safety case), was incredibly time consuming.
4. Instrumenting and tooling the agile environment was essential to deliver living, ubiquitous traceability.
Safety Scrum (SScrum)

1. Discover and specify system stories
2. Preliminary hazard analysis
3. Specify Safety Stories
4. Design solutions
5. Establish trace links from hazards across system artifacts.
6. Sprint level planning
7. Sprint level hazard analysis
8. Sprint execution
9. Incremental Safety Case
10. End of sprint review

Product backlog

Potential deliverable

Yes

No

Safety Case

Working Software

Safe?

Sprint

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

Structure it

Context —> Hazard —> Fault —> Assumption

Acceptance Tests —> Safety Story —> Code —> Unit Tests

Dependent User Story

Issue ID: GROOVY-5223  Type: Improvement
Summary: [GROOVY-5223] Bytecode optimizations: make use of LDC for class literals
Description: Class literals are currently loaded using generated GetClass methods which increase bytecode size and may prevent some optimizations. In most situations though, we may use the LCD bytecode instruction to load the class literal.
Status: Closed  Created: 30/Dec/11 09:59
Resolution: Fixed  Resolved: 03/Jan/12 02:29
• 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.
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

<table>
<thead>
<tr>
<th>FAA Waiver</th>
<th>Explanation</th>
<th>ND</th>
<th>SB</th>
</tr>
</thead>
<tbody>
<tr>
<td>§ 107.25 – Operation from a Moving Vehicle</td>
<td>Fly a UAS from a moving aircraft in populated areas</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>§ 107.29 – Daylight Operations</td>
<td>Fly a UAS at night</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>§ 107.31 – Visual Line of Sight Aircraft Operation</td>
<td>Fly a UAS beyond your ability to clearly determine its orientation with unaided vision</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>§ 107.33 – Visual Observer</td>
<td>User a visual observer without following all visual observer requirements</td>
<td></td>
<td></td>
</tr>
<tr>
<td>§ 107.35 – Operation of Multiple Small UAS</td>
<td>Fly multiple UAS with only 1 remote pilot</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>§ 107.37(a) – Yielding Right of Way</td>
<td>Fly a UAS without having to give way to other aircraft</td>
<td></td>
<td></td>
</tr>
<tr>
<td>§ 107.39 – Operation Over People</td>
<td>Fly a UAS over a person/people</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>§ 107.51 – Operating limitations for Small UAS</td>
<td>Fly a UAS ... fast, high, with low visibility, or close to clouds</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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  |
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?

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

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

<table>
<thead>
<tr>
<th>ID</th>
<th>Role</th>
<th>Domain</th>
<th>Yrs</th>
<th>SC</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Software Engineer</td>
<td>Aviation &amp; Defense</td>
<td>8</td>
<td>Y</td>
</tr>
<tr>
<td>P2</td>
<td>Developer</td>
<td>Operating Systems</td>
<td>1</td>
<td>N</td>
</tr>
<tr>
<td>P3</td>
<td>Developer</td>
<td>Development</td>
<td>2+</td>
<td>Y</td>
</tr>
<tr>
<td>P4</td>
<td>Developer</td>
<td>Embedded Systems</td>
<td>1</td>
<td>N</td>
</tr>
<tr>
<td>P5</td>
<td>Developer</td>
<td>Embedded Systems</td>
<td>2</td>
<td>Y</td>
</tr>
<tr>
<td>P6</td>
<td>Software Engineer</td>
<td>Software Development</td>
<td>7</td>
<td>Y</td>
</tr>
<tr>
<td>P7</td>
<td>Developer</td>
<td>Information Systems</td>
<td>2</td>
<td>N</td>
</tr>
<tr>
<td>P8</td>
<td>Software Engineer</td>
<td>Unmanned Aerial Systems</td>
<td>1</td>
<td>Y</td>
</tr>
<tr>
<td>P9</td>
<td>Systems Engineer</td>
<td>Embedded Systems</td>
<td>35</td>
<td>Y</td>
</tr>
<tr>
<td>P10</td>
<td>Requirements Engineer</td>
<td>Defense Systems</td>
<td>23</td>
<td>Y</td>
</tr>
</tbody>
</table>

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.
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.
All links must be:

- Strategic
- Have demonstrable ROI
- Automated where possible
Quality
Fault Tolerance

Decision
Redundant Ground Control Stations

Rationale
To have at least two GCS for deployment in urban settings.

Alternate argument:
Redundant GCS not needed if drones are given sufficient autonomy.

Supporting argument:
We need two active GCS to provide fault tolerance during an emergency response.

Stakeholders:
Fire Chief
Risk Manager

Sub-Decision
GCS emits heartbeat with monitoring by health monitor

Sub-Decision
Drones can hot-swap between available GCS.

Model
Sequence Diagram

Code
Unit Tests
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
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.