CRACKING THE BIG FREEZE!
TRACEABILITY SOLUTIONS FOR HIGH-DEPENDABILITY AGILE PROJECTS

Jane Cleland-Huang, PhD
JaneClelandHuang@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.
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.

Safety is paramount!
Innovation thwarted by the big freeze!
Natural Strengths

- Fast paced, Incremental
- Agility
- Scale up
- Distributed
- Safety Critical

Traditional Planned (Waterfall/RUP)

- Fast paced, Incremental
- Scale up
- Distributed
- Safety Critical

Agile

- Fast paced, Incremental
- Scale up
- Distributed
- Safety Critical

Safety Assurance
# The Agile Manifesto

- **Individuals** and **interactions** over **processes and tools**
- **Working software** over **comprehensive documentation**
- **Customer collaboration** over **contract negotiation**
- **Responding to change** over **following a plan**

That is, while there is value in the items on the right, we value the items on the left more.

<table>
<thead>
<tr>
<th>Processes/Tools</th>
<th>Documentation</th>
<th>Negotiate contract</th>
<th>Follow a plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ensure safety in a fast-paced environment</td>
<td>Auto-generate documentation to construct safety case</td>
<td>Demonstrate compliance to standards. Deliver safe product.</td>
<td>Manage the impact of change on safety.</td>
</tr>
</tbody>
</table>
Can we merge Agility and Safety Critical?

QUMAS
Agile process in regulated domain
Hardened to address safety issues
Regulated

Dronology
Agile development for Cyber-Physical System (CPS) with safety concerns in a regulated space (e.g., USA FAA)
Studies on Agility in Safety Critical Environments

Scaling Agile Methods to Regulated Environments: An Industry Case Study

Brian Fitzgerald*, Klaus-Jan Sot*, Ryan O’Sullivan*, and Donald O’Brien*

Abstract—Agile development methods are growing in popularity with a recent survey reporting that more than 80% of organizations now follow an agile approach. Agile methods are, however, often perceived as being unsuitable for developing safety-critical systems due to a lack of rigor and traceability. Our goal is to show that this perception is unfounded and that agile methods can be successfully applied to regulatory environments. We present a case study to illustrate how an agile approach was successfully implemented for the design and development of a complex safety-critical system.

I. INTRODUCTION

The widespread popularity of agile methods is widely attributed to the perception that 80% of companies believe that agile methods are suited to their development needs [1]. This widespread adoption has been facilitated by the publication of a number of resources and guidelines, such as the Capers Jones’ Agile Manifesto [2], that have articulated the benefits of agile methods to organizations.

In this paper, we provide a case study of the successful application of agile methods to a complex safety-critical system. The case study was conducted in a regulated environment and demonstrates that agile methods can be used to develop safety-critical systems effectively and efficiently.

II. Background

A. Agile Methodology

Agile methodologies emphasize collaboration, iteration, and adaptability. Key characteristics of agile methodologies include:

1. Continuous development and feedback
2. Collaboration and communication
3. Iterative and incremental delivery
4. Flexibility and adaptability

B. Regulated Environment

Regulated environments are characterized by strict requirements, compliance with standards, and rigorous testing. Agile methods can be applied to regulated environments, but modifications may be necessary to meet regulatory requirements.

III. Case Study

A. Project Overview

The case study describes the application of agile methods to the development of a complex safety-critical system. The project was managed using the Scrum framework, with daily stand-ups, sprint reviews, and retrospectives. The development team was composed of experienced software engineers and domain experts.

B. Challenges

The primary challenge was to ensure compliance with regulatory requirements while maintaining the benefits of agile development. This was achieved through the incorporation of compliance checks and regular reviews into the development process.

C. Successes

The project was successfully completed on time and within budget, meeting all regulatory requirements. The agile approach allowed for rapid iteration and adaptation to changing requirements, leading to improved quality and customer satisfaction.

IV. Conclusion

The case study demonstrates that agile methods can be successfully applied to regulated environments, providing a flexible and efficient approach to software development. Further research is needed to explore the scalability and effectiveness of agile methods in other regulated domains.

References


Keywords: Agile, Scrum, regulated environments, safety-critical systems.
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.
Traceability is important for Safety Critical Systems

The ability to **interrelate any uniquely identifiable** software engineering artifact to any other, **maintain** required links over time, and **use the resulting network** to answer questions of both the software product and its development process.

- CoEST Definition

Required by many regulatory bodies and standards.

**But hard to achieve in practice.**
Accurate & Complete Traceability is challenging

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

Our study of Medical Device submissions to the FDA showed **incomplete and sometimes entirely missing**, inaccurate, redundant trace links – delivered as a big bang!

A **formal comparison of five safety-critical software systems** which claimed to follow various standards and guidelines showed **similar traceability problems**.
QUMAS
Agile process in regulated domain
Hardened to address safety issues
Regulated

Dronology
Agile development for Cyber-Physical System (CPS) with safety concerns in a regulated space (e.g., USA FAA)
Dronology Project @ Notre Dame

A platform for coordinating the flight of UAVs. Supports research in safety assurance, runtime monitoring, & adaptation.

Developed by the Notre Dame Team: Michael Vierhauser, Jane Wyngaard, Jinghui Cheng, Sean Bayley, Greg Madey, Joshua Huseman, Jane Cleland-Huang, & more...

http://sarec.nd.edu/pages/Dronology.html
River Rescue Demo with Dronology
Drone technology is imperfect.
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.
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

Potential deliverable

Safe?

Yes
No

Sprint

Safety officer

Safety Assurance Case

Working Software

Product backlog

Sprint backlog

FMECA

Preliminary hazard analysis
A1: Discover and Specify System Stories

Work with project stakeholders to elicit and specify an initial set of safety stories, representing the functional requirements of the project.

**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.
A2: Perform Preliminary Hazard Analysis

Conduct a hazard analysis in early phases of the project to identify potential hazards and failure modes in order to drive activities such as architecture design, story specification, safety analysis, and rigorous testing throughout the remainder of the project.

Preliminary Hazard:
UAV’s battery fails unexpectedly and the UAV falls to the ground.

1. The battery level detector on the UAV fails to detect a low battery level.
2. The battery level indicator on the UAV reports incorrect battery level.
3. Dronology software fails to check battery level in time to take responsive actions before the battery fails.
4. Dronology software is aware of low battery warning but fails to take responsive actions.
A3: Specify Safety Stories

Identify, analyze, and specify safety stories that, if satisfied, will prevent the hazard from occurring or reduce the impact of its occurrence. Place safety stories into the product backlog.

Safety Story (SAF-1):
The GPS coordinates of each UAV must be accurate within one meter at all times.
Design a solution to address each of the safety stories. Specify the solution as a set of design definitions prior to scheduling its associated safety-stories into a sprint.

(DD-1): When Dronology is deployed in an urban environment at least two independent means of UAV localization must be used.

(DD-2): If UAV localization mechanisms provide conflicting information, the lowest reported distance between two UAVs shall serve as their current separation distance.

(DD-3): Real-Time Kinematic (RTK) shall be deployed. (Note: This increases the guaranteed accuracy of GPS to 2 centimeters.)
A5: Establish Hazard related Trace Links

Leverage capabilities of tools commonly adopted in agile projects (e.g., Jira and Github) to incrementally construct trace links from safety stories to hazards, design definitions to safety stories, dependent system stories to safety stories, source code to design, and from acceptance tests to safety stories.

More details later in the talk!
A6: Sprint Planning

Track the design status of each safety requirement. Be aware of dependencies between system stories and safety stories and favor schedules in which safety stories and their associated design decisions are implemented as close as possible to their dependent system stories.
A7: Sprint Level Hazard Analysis

Perform an in-depth hazard analysis at the start of each sprint to identify new hazards, failure cases, and mitigations associated with new features and their interactions with other features.

System story (R5): UAVs shall be launched from a boat during river rescue.

Requirement (R6): When the return to launch (RTL) command is issued, the UAV shall return to its original launch coordinates.

Safety Story (R6'): When the return to launch (RTL) command is issued, the UAV shall return to its home coordinates.
A8: Handle Emergent Faults

When new hazards, failure modes, and safety stories emerge as a result of observed faults during testing, the faults are documented, new safety stories specified and design solutions explored. The new stories and design definitions are added to the product backlog, and dependencies are documented as trace links.
A9: Incremental Safety Case

Throughout the sprint, the team must incrementally refine the safety case (SC) and construct an accurate, clear, comprehensive and defensible argument that a system is acceptably safe to operate in a particular context.
Safety Assurance Cases

Claim: Assertion of compliance with key requirements and properties. Must be within a specific context of use.

Arguments/Strategy: Link evidence to claims via inference rules. Can be deterministic (true/false), probabilistic, or qualitative (i.e. link to regulations).

Evidence: Process and people, testing, reviews, mathematical proofs.

Context: Environmental Assumptions

Examples provided by Jinghui Cheng
Challenges of Safety Assurance Cases

Problems include:
- Building them
- Reviewing them
- **Maintaining** them
- **Reusing** them

It is difficult to understand the impact of a change on the assurance case because:
- Huge volumes of data
- The impact of a change on the assurance structure is complex

T. Scott Ankrum, MITRE (2012)
Incrementally Evolving Safety Case

Given a fully approved/certified version V1, and a modified V2 – can we reuse any of the Safety Assurance Case, and if so, which parts??

Release V1

Release V2
A10: End of Sprint Safety Gateway

Analyze the safety of the system at the end of each sprint.

- If necessary conduct “hardening” sprints to address safety concerns and/or to prepare systems in regulated domains for certification.
- Understand the safety status of the release

**System Story (RX):**
Dronology shall directly control the flights of up to 50 concurrent UAVs.

**Safety Story (SR-3):**
UAVs must maintain a MINIMUM SEPARATION DISTANCE of 3 meters at all times

Resolved through a combination of manual and encoded design decisions
Addressing the Incremental Safety Case Challenge

Instrument the environment with “living traceability” so that trace links are current.

Establish traceability between hazards and artifacts so that you can understand exactly how changes in the system impact the safety case, and evolve it accordingly.
A Safety Assurance Case Argues that a Hazard is mitigated

Hazard: UAV is no longer controllable from Dronology due to communication loss.

Each drone has a human operator assigned to its hand-held GCS, responsible for taking control when directed by the safety officer.

G1: A human operator shall have to ability to safely assume control of a UAV in flight.

G1.1: Upon receiving a takeover command, the central control unit stops issuing directives.

G1.1.1: The system only issues directives in GUIDED mode.

G1.2: The GCS monitors for mode changes from GUIDED to STABILIZE and issues the takeover command to the central control unit.

G1.2.1: All mode changes issued on the hand-held GCS are monitored by the core control unit.

G1.3: The UAV shall hover-in-place upon hand-over until the user issues a command from the GCS.

G1.3.1: Operators trained to check switch positions.

A goal in the safety assurance case is designed to mitigate a specific hazard.

We take a slightly different approach from a traditional safety argument.
Overview of SAFA (Software Artifact Forest Analysis)

1. Design the TIM (Traceability Information Model) and instrument the environment.

2. Identify hazards and organize them hierarchically.

3. Generate a safety tree and an artifact tree for each hazard. Inject warnings and recommendations.

4. Generate a delta tree showing changes between versions and analysis of their impact.
Hazards can be identified through any hazard analysis process such as: brainstorming, checklists, Fault Mode Effect Analysis, or through Fault Tree Analysis.
The tree is automatically generated from links in Jira. Warnings are issued for missing links, and link recommendations made.
# Transform the Tree into a Safety Case

<table>
<thead>
<tr>
<th>Strategy Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Id</strong></td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
</tbody>
</table>

## Assumption Node

| **Id** | **Source** | **Target** | **Condition** | **Argument or Claim** |
| 9 | Prob.of hazard | Assumption | A FMEA assumption node exists | Claim that FMEA supports assumption of hazard’s likelihood |

## Context Nodes

| **Id** | **Source** | **Target** | **Condition** | **Argument or Claim** |
| 10 | Safety Req. | Context | System level hazard analysis context node exist | Claim that system level hazard analysis is a context node for top safety requirement |
| 11 | Safety Req. or Hazard | Context | A context node exists | Claim that defining and explaining ambiguous term in statement such as ”correct”, ”low”, ”sufficient”, ”negligible” is a context node for Safety Req or Hazard |

## Solution Node

| **Id** | **Source** | **Target** | **Condition** | **Argument or Claim** |
| 12 | Prob. of hazard | FTA | FTA gives probability | Claim that FTA evidence is a solution for probability of hazard |
Add SAC nodes to create a Safety Tree
Add Warnings

Warnings:
- Egregious omissions e.g., no source code links
- Nuanced omissions e.g., insufficient diversity of evidence

Recommendations:
- Vet a set of candidate links
- Add missing artifacts
- Explain a specific strategy.

Some artifacts have been removed to simplify the example.
<table>
<thead>
<tr>
<th>Change</th>
<th>Warning</th>
<th>Secondary Condition</th>
<th>Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1: Feature is added or deleted</td>
<td>A new/deleted feature [FEATURE] potentially impacts the following Safety Trees: [LIST OF SAFETY TREES].</td>
<td>FMEA or FTA exists for feature.</td>
<td>Check that hazard analysis is complete for [FEATURE NAME]. If necessary create new Safety Trees associated with new hazards.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>At least one existing feature exists.</td>
<td>Perform a feature interaction analysis for this feature with other existing features.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>At least one Safety Tree added or removed for the feature.</td>
<td>Perform a complete check of each Safety Tree associated with hazards for [FEATURE NAME] including all strategies, claims, and contexts.</td>
</tr>
<tr>
<td>W2: Requirement is added, deleted or modified</td>
<td>Requirement [REQUIREMENT ID] has been [Add., Del., Mod.]</td>
<td>Requirement has been added or modified.</td>
<td>Check the claim that requirement [REQUIREMENT ID] mitigates hazard [SAFETY TREE HAZARD ROOT] and is fully realized through the design, implemented in the code, with diverse and sufficient evidence provided.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Requirement has been deleted or modified.</td>
<td>Check the claim that [REQUIREMENT PARENT] remains satisfied even though [REQUIREMENT ID] has been [DELETED—MODIFIED].</td>
</tr>
<tr>
<td>W3: Source code is changed</td>
<td>Source code in [PACKAGE] has been modified by [MODS]</td>
<td>All</td>
<td>Check that all code is covered by passed unit tests.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SAFA recommends trace link maintenance</td>
<td>Check the [LIST OF RECOMMENDED TLE ACTIONS] to confirm additions and deletions of trace links.</td>
</tr>
<tr>
<td>W4: Context and/or assumptions have changed</td>
<td>Context/Assumption associated with [CONTEXT NODE] has changed.</td>
<td>Environmental assumption has changed</td>
<td>Check all goals, claims, strategies, and solutions that are influenced by [ASSUMPTION].</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Probability has changed</td>
<td>Check that all goals, claims, strategies, and solutions that are influenced by [PROBABILITY] are supported within the new probability context.</td>
</tr>
<tr>
<td>W5: Evidence has been modified or deleted</td>
<td>Evidence has been modified or removed.</td>
<td>Solution has been deleted or modified</td>
<td>Solution that [goal] has been satisfied is challenged by elimination or change of [EVIDENCE TYPE]. Check that [GOAL] is sufficiently satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Diversity of evidence has been reduced.</td>
<td>Diversity of evidence that [GOAL] has been satisfied is reduced by elimination of [EVIDENCE TYPE]. Check that [GOAL] is sufficiently satisfied.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Context node using this evidence has changed.</td>
<td>Check all goals, claims, strategies, and solutions that are influenced by [CONTEXT].</td>
</tr>
</tbody>
</table>
Compare Two Trees

What has changed and how do those changes impact safety?

General Notation used in the Safety Trees

- **Strategy**: Typically used to argue that a high level element is fully satisfied or addressed by its child nodes.
- **Context**: Describes a context in which the system is expected to operate.
- **Assumption**: Describes an assumption of the safety argument or the environment.
- **Solution**: Represents a raw artifact retrieved from the project repository, e.g., hazard requirements, source code, or test case.
- **Delegated Solution**: A responsibility delegated to an adjacent system.
- **Warning!!**: Warns that a type of element is completely missing.

**Delta Tree Notation**
- **No change across versions**.
- **Added to V2**.
- **Existed in V1 but deleted from V2**.
- **Existed in V1 and modified in V2**.

Right click node for recommendations.
Current state of the art traceability techniques show great promise for supporting industrial projects. They can be used to recommend missing links and to support trace link evolution.
**Empirical Study**

**RQ1:** What types of changes impacted hazard mitigations between versions v1 and v2, and is SAFA able to detect and visualize them?

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

### Version 2 (v2)
- LOC: 73,591
- Java Classes: 646
- Requirements: 185
- Design definitions: 283

### Legend:
- SR(Safety Req)
- PR(Process)
- DD(Design Def.)
- AJ(Adjacent system)
- CX(Context)
- AS(Assumption)
- TS(Tests)
- SC(Code)

### Results
All changes were identified by SAFA. Changes from our study covered 8 different artifact types.
User Study

**RQ2:** 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

(Q1) With respect to this hazard, has the system changed in a way that potentially affects its safety? If so please explain your answer.  **SAFA 100%, Paired view 80.6%**

(Q2) Was the information provided to you for each of the two methods sufficient for assessing the impact of change upon system safety?  **Yes but SAFA provided more information about changed nodes.**

(Q3) Which of the two methods did you prefer using? Why?  **SAFA 9, Paired view 1**

(Q4) Can you recommend any changes for SAFA?
### User Study

<table>
<thead>
<tr>
<th>Q</th>
<th>Theme</th>
<th>Description</th>
<th>Cnt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q3</td>
<td>Visualization</td>
<td>The extent to which color coding and other visualizations highlight issues</td>
<td>6</td>
</tr>
<tr>
<td>Q3</td>
<td>Speed</td>
<td>The extent to which problems can be identified quickly</td>
<td>6</td>
</tr>
<tr>
<td>Q3</td>
<td>Informative</td>
<td>The extent to which the provided information supports safety analysis</td>
<td>5</td>
</tr>
<tr>
<td>Q3</td>
<td>Process</td>
<td>The ease of the analysis process</td>
<td>5</td>
</tr>
<tr>
<td>Q3</td>
<td>Trust</td>
<td>The trust that a user places in SAFA to identify problems</td>
<td>1</td>
</tr>
<tr>
<td>Q4</td>
<td>Code insight</td>
<td>The ability of SAFA to identify and display impactful code changes</td>
<td>3</td>
</tr>
<tr>
<td>Q4</td>
<td>Rationale</td>
<td>Rationales explaining additions, deletions, or modifications of artifacts.</td>
<td>2</td>
</tr>
</tbody>
</table>

Many helpful suggestions for AI features that SAFA could support in the next iterations!

- 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?
Conclusion

The ability to interrelate any uniquely identifiable software engineering artifact to any other, maintain required links over time, and use the resulting network to answer questions of both the software product and its development process.

- CoEST Definition

Traceability is important for safety critical projects

- Required by many regulatory bodies and standards.
- But hard to achieve in practice.

We need to build safety cases incrementally

- Breaking the big freeze is highly feasible
CRACKING THE BIG FREEZE!
TRACEABILITY SOLUTIONS FOR HIGH-DEPENDABILITY AGILE PROJECTS

Jane Cleland-Huang, PhD
JaneClelandHuang@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.
Leverage Deep Learning Techniques

Source $s_1 s_2 \ldots s_m$

Word Representation Mapping

Target $t_1 t_2 \ldots t_n$

Source $v_{s1} v_{s2} \ldots v_{sm}$

Sentence Semantic Representation

Target $v_{t1} v_{t2} \ldots v_{tn}$

Source $V_s$

Trace Link Evaluation

Target $V_t$

$p_{link}$

Lexical semantic knowledge

Variability of linguistic expression

Informal reasoning

Semantically enhanced Software Traceability using Deep Learning techniques.
Jin Guo, Jinghui Cheng, Jane Cleland-Huang: ICSE 2017: 3-14
Automated approaches that generate trace links from scratch, return imprecise results.

They are useful for supporting tasks such as Impact analysis, but not currently sufficiently reliable on their own.
Solution 2: Evolving and Discovering Links

1. Software artifacts changed across versions
2. Tools for detecting changes in code and requirements.
3. TLE tools and algorithms for recognizing change patterns and evolving trace links.
Evolving Links

1. Identify types of changes that could invalidate existing trace links.

2. Define properties to detect when the change has occurred.

3. Define trace link generation heuristics
Integrate the user in the loop

In our experiments the effort required by humans to confirm or deny TLE links was minimal – with few decision points per day.

Leverage the Project Environment

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