AUTOMATED REQUIREMENTS ENGINEERING CHALLENGES

33rd International Conference on Automated Software Engineering
Montpellier, France
6th September, 2018

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

Work described in this talk is funded by the US National Science Foundation under Grants CCF-0959924, CCF-1265178, and CCF-1741781
What I’m going to talk about today

Dronology: Requirements Challenges across the LifeCycle

Flavors of RE

Open RE Automation Challenges

Pathways to Practice
Requirements engineering is the branch of software engineering concerned with the **real-world goals** for, **functions** of, and **constraints** on software systems.

*Zave – 1982.*

RE Activities may be more iterative, **involve many more players**, ..., require more extensive **analyses of options**, and call for more complicated verification of more diverse components.

* Cheng and Atlee, 2005

RE is often regarded as a **front-end activity** in the software systems development process.

*Nuseibeh and Easterbrook - 2000*
What does RE look like today?

Requirements are ‘crowd sourced’ and discovered from app reviews and feature requests.

Requirements are written informally as user stories.

Projects driven by Natural Language Requirements

Requirements are written using “shall” statements.

Requirements are formally specified using logic or models.
Why do we care?

• Problems with requirements are one of the leading causes of project failures.
  • Unresolvable conflicts
  • Volatile requirements
  • Lack of stakeholder input

• Late emerging architecture breakers
• Thrashing design
• Building the wrong product
Quality requirements (aka NFRs) were not adequately explored
- Vulnerable to security attacks
- Ground to a stand still
- Didn’t address users’ needs adequately.
Infamous Failures: Mars Climate Orbiter

The orbiter was lost during a mission to Mars.

One team used inches & pounds while another used metric measures.

One of the 8 contributing factors:

Problems in verifying and validating certain engineering requirements...
Requirements @ ASE

TESTING STUDIES (WEDNESDAY 5TH, SEPTEMBER - 10:30-12 - AUDITORIUM EINFELD)

PERFORMANCE (WEDNESDAY 5TH, SEPTEMBER - 10:30-12 - ROOM JOFFRE AB)
- On Adopting Mixers to Deal with Performance Concerns in Android Apps. Sara Habchi, Xavier Blanc, Romain Bruguière.

QUALITY ASSURANCE FOR MACHINE LEARNING TECHNIQUES (WEDNESDAY 5TH, SEPTEMBER - 1:30-3 - AUDITORIUM EINFELD)
- Unit Testing for Deep Neural Networks. Yucheng Sun, Min Wu, Wenjie Ruan, Xiaoli Huang, Marta Kwiatkowska, Daniel Kroening.

BUILD AND TEST AUTOMATION (WEDNESDAY 5TH, SEPTEMBER - 1:30-3 - ROOM JOFFRE AB)

MINING AND CROWD SOURCING (WEDNESDAY 5TH, SEPTEMBER - 2:30-5 - AUDITORIUM EINFELD)
- Assessing the Type Annotation of Mining File Histories: Should We? Pedro M. Barata, João J. Monteiro, Tiago S. Elzir, Daniel P. Ferreira.

ARCHITECTURE AND REQUIREMENTS (THURSDAY 6TH, SEPTEMBER - 3:30-5 - ROOM JOFFRE AB)
- Estimating the number of remaining links in traceability recovery. Davide Falesi, Massimiliano Di Penta, Gerardo Canfora, Filippo Mencarelli.

A Selection of Open RE Challenges

1. Mining domain knowledge
2. Requirements Elicitation
3. Requirements Analysis
4. Requirements Traceability
5. Requirements Visualization
6. Requirements @ Runtime
Traditional Requirements

1. Project Blastoff
2. Domain analysis
3. Project goals
4. Discovering and Prioritizing Needs
5. Trawl for Requirements
6. Domain knowledge & Reusable requirements
7. Write Requirements
8. Classifying Requirements
9. Requirements Analysis
10. Quality Gateway
11. Requirements Specification
12. Review Requirements
13. Design & Development
14. Working Product
15. Automated traceability
16. Product use & Evolution
17. Reuse library
18. Missing Requirements
19. Risks and Costs
20. Modified from: http://www.volere.co.uk/masteringrequirementsprocess.htm

Stakeholders

Feedback
Agile “Requirements”

- Inputs from Executives, Team, Stakeholders, Customers, Users
  - Extracting user stories
  - Prioritizing
  - Selecting stories
  - Effort estimation
  - Safety analysis
  - Impact analysis
  - Automated traceability

- Daily Scrum Meeting
  - Burndown/up Charts
  - 1-4 Week Sprint
  - Task Breakout

- Sprint Review
- Finished Work
- Sprint Retrospective

- Product Owner
- The Team
- Product Backlog
- Sprint Planning Meeting
DATA AVAILABLE?

- run experiments
- evaluate results
Dronology Project @ Notre Dame

http://sarec.nd.edu/pages/Dronology.html
A Product Line for Managing & Controlling UAVs

An entirely new controllable environment for experimenting with Software Traceability across multiple artifact types and multiple versions.

A system that focuses on an agile, yet safety-conscious, more requirements-centric process.
A platform for coordinating the flight of UAVs. Supports research in requirements engineering, cyber-physical systems, safety assurance, runtime monitoring, & adaptation.
River Rescue Demo with Dronology

Prototyping the use of UAVs as part of a River Rescue with stakeholders from the Fire Department, City, and public.
Testing an AED drop
Testing an AED drop
We started with just a simple idea...
The Learning Curve
Requirements Discovery

- Ethnography
- Brainstorming
- Interviewing
- Workshops
- Prototyping
- Domain Mining
What our industry collaborators told us:

It takes us months of person effort to understand the domain when we enter a new market. A large amount of that time is spent manually analyzing documents we find online.

Can you automate that task?

Mine domain specific artifacts from the internet.

Analyze them

Discover functional requirements

Discover quality requirements

Generate Architectures
## Challenge #1: Mine Domain knowledge

### Communication Segment
- **Wireless Network**
  - *GPS antenna*  
  - *Wi-Fi antenna*  
  - *DCS antenna*
- **antenna**
- **Edge message protocol (EMP)**
- **Base Comm. Package (BCP)**
- **Cluster Controller (CC)**
- **Communication Manage. Unit (CMU)**
- **Front end processor (FEP)**

### Wayside Segment
- **Wayside Interface Unit (WIU)**
  - Hash Message Authentication Code (HMAC)
- **Wayside Detector**
  - Wheel impact load detector
  - Wheel temperature detector (WTD)
  - Hazard detector
  - Automated equipment indent.
  - Output processing
  - Strain gauge-based detector
  - Proximity sensor
  - Current sensor
- **Track Warrant Control (TWC)**
- **Automatic block signal (ABS)**
- **Centralized traffic control (CTC)**
- **Movement authority**
- **Direct traffic control**
- **Wayside Status Relay Service (WSRS)**
- **Logic Controller**

### Locomotive Segment
- **Locomotive Control**
  - Cruise control
  - Direct control
  - Monitor
  - Command messages
  - Location management
  - Speed management
  - Train line
  - Cab control signal
  - Authority management
  - Motive power
  - Event function logger
  - Remote control
  - Bulletin management
  - Remote control locomotive (RCL)
- **Brake**
  - Dynamic Brake Monitor (DBM)
  - Electronic Air Brake (EAB)
  - Electronically Controlled Pneumatic brake (ECP)
  - Running Brake
- **Event Recorder**
  - OEM Identifier
  - Crashworthy Memory Module (CMM)
  - Event Data Acquisition Process (EDAP)
- **Locomotive Network (LNET)**
- **Locomotive Interface Gateway (LIG)**
  - Edge Message Protocol (EMP)
  - Class C message
  - Class D message
  - Cyclic Redundancy Check (CRC)
- **Onboard Unit (OBU)**
  - Central Processing Unit
  - Braking Algorithm
  - Crew interaction
  - Wheel tachometer
- **Computer Display Unit (CDU)**
  - Human Machine Interface (HMI)
  - Crew acknowledge
  - Navigation
  - Energy management
  - Bulleting cancellation
  - Track location
  - Authority change request
  - Mandatory directive
  - New bulletin
  - Enforcement
  - PTC territory
  - Park train
  - Departure test
  - Map display

### Employee-in-Charge Portable Remote Terminal (EIC)
- **Interface with PTC**
  - To Locomotive
  - To Office
- **Communication**
  - *GPS*
  - *WLAN*
- **Safety control**
  - Work Vehicle Enforcement Module (WVEM)
  - Communication-Based Train Control (CBTC)
  - Maintenance of Way Protection (MWP)
  - Work Authorities (WA)

### Office Segment
- **Dispatch System**
  - Bulletin
  - Dispatcher message
  - Dispatcher rules
  - Mandatory directives
  - Consist
- **System Management Public Key (SMPK)**
- **database management**
- **Back office server (BOS)**
  - Track topology information
  - Speed restriction
  - Authorities
  - User authentication and authorization data
- **System Management Agent**
  - Security management
  - Kit rollback
Algorithms & Tool to Discover Requirements

(a) Documents ranked by relevance

<table>
<thead>
<tr>
<th>File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>ED043_PTC implementation plan - NCTD.pdf</td>
</tr>
<tr>
<td>ED043_PTC implementation plan - Northern Indiana.pdf</td>
</tr>
<tr>
<td>ED047_PTC implementation plan - Northern Indiana.pdf</td>
</tr>
<tr>
<td>ED039_PTC implementation plan - Metra.pdf</td>
</tr>
<tr>
<td>ED039_PTC implementation plan - Metra.pdf</td>
</tr>
<tr>
<td>ED105_S-9101 Loco Sys Architecture.v1.5.pdf</td>
</tr>
<tr>
<td>ED105_S-9101 Locomotive System Architecture.v1.5.pdf</td>
</tr>
<tr>
<td>ED205_2010.12.03_Standard_ITC_PTC_Proposal.pdf</td>
</tr>
<tr>
<td>ED205_2010.12.03_Standard_ITC_PTC_Proposal.pdf</td>
</tr>
<tr>
<td>ED038_PTC implementation plan - MBTA.pdf</td>
</tr>
</tbody>
</table>

(b) Summary of relevant parts of the selected file

2.3.1.1 OBU OPERATING MODES

- The OBU will support the following basic operational modes in addition to those necessary during startup and operation:
  - Supervision Mode: The OBU occupies this mode during normal operation. When in this mode, the OBU will continue to...
  - Restricted Mode: The OBU can be made to...
  - Non-PTC Mode: The OBU enters this mode...
  - Movement Authority: While in this mode, the OBU will continue to...
  - Cut-out Mode: OBU can be...

(c) Selected page showing relevant text highlighted.

2.4.2.3.2 FUNCTIONAL DESCRIPTION

The Track Database information provided during train initialization (as described in 2.4.1) will identify PTC territory boundaries and enable the OBU to perform activities necessary for region transitions.

The OBU will continuously monitor train location and speed in order to determine whether a territory boundary is approaching. When the train approaches a region boundary, the OBU will contact the new territory’s BOS and collect track database and track bulletins for the new territory. The process of updating the track database will be the same as for train initialization, with the exception that crew authentication and train consist data will not be...

Table 10: Function Allocation for PTC Territory Transition

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monitor Region</td>
<td>The OBU uses train location information to detect approaching territory boundaries.</td>
</tr>
<tr>
<td>Change Region</td>
<td>When approaching a new territory boundary, the OBU will establish connections to the BOS serving the new region and perform the needed initialization tasks:</td>
</tr>
<tr>
<td></td>
<td>- Provide crew authentication data</td>
</tr>
<tr>
<td></td>
<td>- Provide Locomotive ID</td>
</tr>
<tr>
<td></td>
<td>- Receive Track Database and Track Bulletin updates from the new BOS</td>
</tr>
</tbody>
</table>

Xiaoli Lian, Mona Rahimi, Jane Cleland-Huang, Li Zhang, Remo Ferrai, Michael Smith: Mining Requirements Knowledge from Collections of Domain Documents. RE 2016: 156-165
Trained a classifier to identify keywords associated with various functional requirements. The goal is to identify quality requirements from amongst a set of existing requirements.

Xiaoli Lian, Jane Cleland-Huang, Li Zhang: Mining Associations Between Quality Concerns and Functional Requirements. RE 2017: 292-301
Mapping Requirements

Lots of automation challenges and activity…

Is it helpful? Scalable? Time saving?

One barrier to answering this question is user accessibility.

Xiaoli Lian, Ahmed Fakhry, Li Zhang, Jane Cleland-Huang: Leveraging Traceability to Reveal the Tapestry of Quality Concerns in Source Code. SST@ICSE 2015: 50-56
Where does Automation Make Sense?

- More efficient!
- More complete results!
- More accurate results!
- More fun!
- Trustworthy
- Timely (response time)
- Integrated (user friendly)

Human intensive part of the Software Development Life Cycle
The Path to Impact

Idea

Proof of Concept

User Tests

Adoption

Academia Inspired

Industry Inspired

Practitioners

Students

Research in Automated Software Engineering must include more user testing in the research life-cycle.
Dronology as a Resource

**Collision Avoidance**
- Backend message
- Guidance Commands
- Avoidance Strategy

**Coordinate**
- Exceptions
- Fleet
- Monitoring

**Vehicle**
- Commands
- Internal
- Managed drone internal
- Proxy
- Flight
- Internal
- Flight zone

**Ground Station Connector**
- Connect
- Messages
- Dispatch
- Service
- Monitoring
- Service
- Tree
- Simple Checker

**Core**

**Dronology Research Incubator provides a more complete set of resources than are typically made available in industrial projects.**

**Hazard (UAV-931):** Throttle position is in incorrect position when UAV is handed over to RPIC causing the UAV to crash into the ground.

**Requirement: (UAV-967):** UAV hovers in place when user requests control until further directives are received from the hand-held controller.

**Design Definition (UAV-927):** Human operators are trained to handle diverse UAV failures.

**Design Definition (UAV-968):** Check that hand-held throttle position is in neutral position prior to takeoff.
And the challenge is..

Discover requirements for a domain, organize them within an architectural decomposition, and specify them in terms of functional and non-functional requirements in a “standard” format.

Perform this ‘from scratch’ or by ‘filling in the gaps’ of an initial specification.

A collection of existing requirements specifications are provided by Didar Zowghi at: http://research.it.uts.edu.au/re/cgi-bin/resources_srs.cgi
Another Challenge: Requirements Traceability
Why real projects?

- Strengthens the believability of Dronology as an ‘industrial strength’ project – helping us to achieve our original goal of developing a ‘research incubator’.
- Opens up *new* and interesting research questions. Advances state of the art in small UAV applications.
- Addresses a humanitarian need.

Safety Critical, Cyber Physical System.
Dealing with a black box controller...

- Bent prop drone
- Trajectory challenged, upside down drone
- Broken leg drone
- Rescued Drone
- Drowned and missing drone
A Near Catastrophe
When we ask developers who are looking for a better way to develop software and we ask them what their most pressing issues are:

- Integration too late if at all
- Lack of traceability, flexibility and evolvability
- Reuse methods ad hoc and error prone
- Software unreliable even with extensive testing
- Costs too much. Takes too long

Why still? Not unlike 50 years ago when the field was brand new*. What to do...

* M. Hamilton, “What the Errors Tell Us”, IEEE Software • Special issue "50 years of Software Engineering"

The programmer who wrote much of the code that took Apollo to the moon!

And the challenge is..

Given a set of requirements, automatically generate links back to their origins and forward to their downstream artifacts.

Perform this task by leveraging any natural language text, process information, and/or existing trace links available to the project.

A collection of existing requirements specifications are provided by CoEST (Center of Excellence for Software Engineering): http://coest.org/datasets
Traceability Slices

Identify, analyze, and specify safety requirements that, if fully realized, will prevent the hazard from occurring or reduce the impact of its occurrence.

Is the safety requirement fully decomposed?

Are all trace links intact?

Is the safety argument sufficient?
Automated Trace Link Creation and Maintenance

Figure courtesy of Alex Egyed, Johannes Kepler University
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 $v_{t1} v_{t2} \ldots v_{tn}$

Trace Link Evaluation

$p_{\text{link}}$

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.
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.

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.
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
My Research Group’s Path to Impact?

<table>
<thead>
<tr>
<th>Idea</th>
<th>Proof of Concept</th>
<th>User Tests</th>
<th>Adoption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Academia Inspired</td>
<td><img src="Diagram.png" alt="Diagram" /></td>
<td><img src="Practitioners.png" alt="Practitioners" /></td>
<td></td>
</tr>
<tr>
<td><img src="Diagram.png" alt="Diagram" /></td>
<td><img src="Practitioners.png" alt="Practitioners" /></td>
<td><img src="Students.png" alt="Students" /></td>
<td></td>
</tr>
</tbody>
</table>

Research in Automated Software Engineering must include more user engagement in the research life-cycle.
In our experiments the effort required by humans to confirm or deny TLE links was minimal – with few decision points per day.

I would kill to have Tool X in my workplace.

Integrate real users into automation experiments.

Another User Study

Safety Critical Trace..

I find myself implicitly trusting the tool – so if it fails, I will miss something important.
Research in Automated Software Engineering must include more user testing in the research life-cycle.
A New Section in our Automation Papers?

If we are serious about impact...

Pathway to Practice
The Eyrie Research Incubator in collaboration with Robyn Lutz

<table>
<thead>
<tr>
<th>CISE Users</th>
<th>Eyrie Research Incubator Projects &amp; Resources</th>
</tr>
</thead>
</table>
| **Researcher** clones one of the incubator projects and uses the runtime environment to support and/or evaluate their own research agenda. E.g., self-adaptation algorithms. | **Dronology**
- Software Artifacts
- Physical Elements
- Runtime Environment
- Development Environ. |
| **Researcher** uses prepared bundle of static artifacts to address an open research challenge. E.g., evolution of safety assurance cases across versions of a product. | **SafeWalk**
- Software Artifacts
- Physical Elements
- Runtime Environment
- Development Environ. |
| **Instructor** creates an assignment using a challenge as an exercise. | **Static Artifact Challenges**
- Automatically evolve trace links across safety-related software artifacts. |
| | **Runtime Challenges**
- Evaluate a new algorithm for supporting self-adaptation. |
| | - Student exercise to create and evaluate collision avoidance algorithms. |
| | - Present runtime data in ways that support human decision making. |

**Challenges**

**SafeWalk**

- Formally specify requirements for CPS User Interface.
- Model safety and check safety properties associated with a UI.
An entirely new controllable environment for experimenting with Software Traceability across multiple artifact types and multiple versions.

A system that focuses on an agile, yet safety-conscious, more requirements-centric process.

https://dronology.info/datasets/
Benefits: Runtime Environments
Benefits: Immersive Software Engineering Environments

More realistic user studies prior to full-blown industrial ones..
The Eyrie Research Incubator

CISE Users

Researcher clones one of the incubator projects and uses the runtime environment to support and/or evaluate their own research agenda. e.g., self-adaptation algorithms.

Researcher uses prepared bundle of static artifacts to address research challenges, e.g., evolution of safety assurance cases across versions of a product.

Instructor creates an assignment using a challenge as an exercise.

Eyrie Research Incubator Projects & Resources

Projects

Dronology
- Software Artifacts
- Physical Elements
- Runtime Environment

SafeWalk
- Software Artifacts
- Physical Elements
- Runtime Environment

Challenges

Static Artifact Challenges
- Automatically evolve trace links across safety-related software artifacts.
- Formally specify requirements for CPS User Interface.
- Model safety and check safety properties associated with a UI.

Runtime Challenges
- Evaluate a new algorithm for supporting self-adaptation.
- Student exercise to create and evaluate collision avoidance algorithms.
- Present runtime data in ways that support human decision making.

Open for collaborators!
The Eyrie Project

Open RE Challenges
- Mining domain knowledge
- Requirements Analysis
- Requirements Traceability
- Requirements Visualization

Transfer to Practice that includes User Studies

Research in Automated Software Engineering must include more user testing in the research life-cycle.

ANY QUESTIONS?