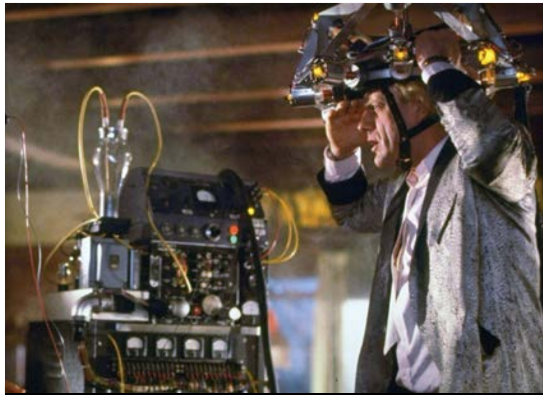
Requirements Engineering Research in a Changing World

International Requirements Engineering Conference
Keynote: August 22nd, 2018
Banff, Canada

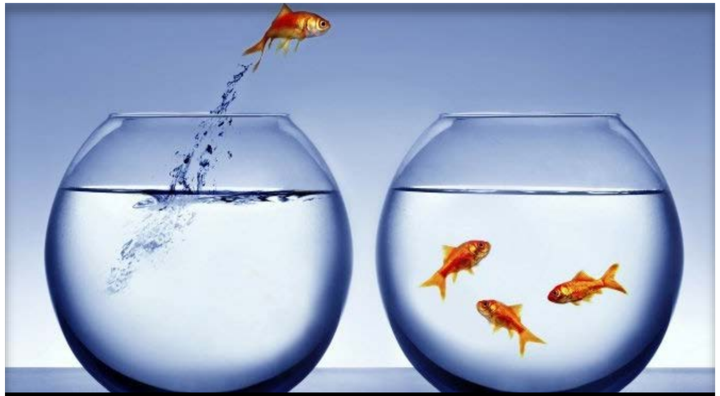
Jane Cleland-Huang, PhD
University of Notre Dame
Dept. of Computer Science and Engineering
http://sarec.nd.edu
JaneClelandHuang@nd.edu

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



Back to the future

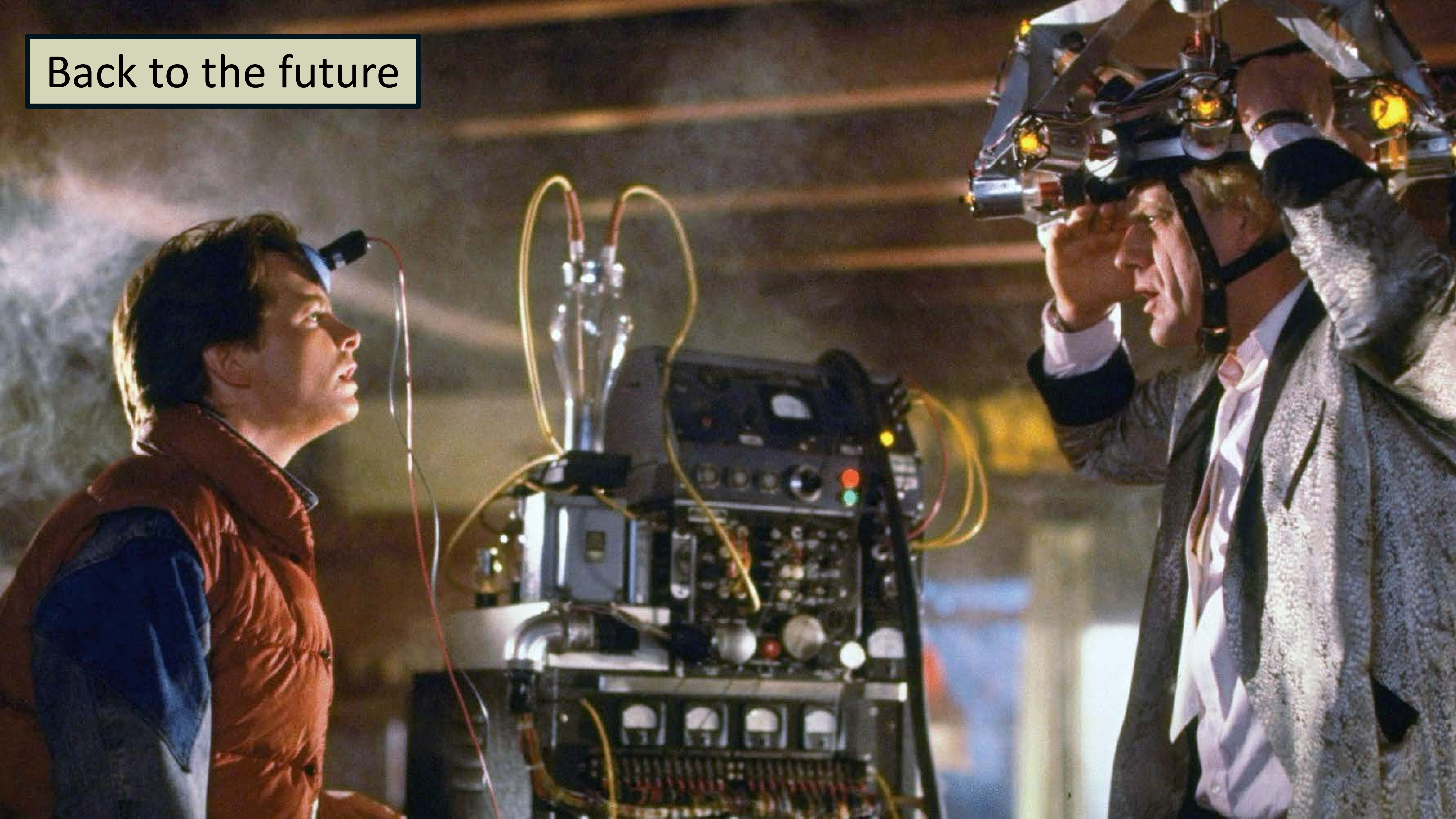


Disruptive change

Our Response

Confessions of a Traceability Researcher!

Back to the future



A Requirements Engineering Road Map (2000)

Requirements Engineering: A Roadmap

Bashar Nuseibeh
Department of Computing
Imperial College
180 Queen's Gate
London SW7 2BZ, U.K.
Email: ban@doc.ic.ac.uk

Steve Easterbrook
Department of Computer Science
University of Toronto
6 King's College Road
Toronto, Ontario M5S 3H5, Canada
Email: sme@cs.toronto.edu

ABSTRACT
This paper presents an overview of the field of software systems requirements engineering (RE). It describes the main areas of RE practice, and highlights some key open research issues for the future.

1 Introduction
The primary measure of success of a software system is the degree to which it meets the purpose for which it was intended. Broadly speaking, *software systems requirements engineering* (RE) is the process of discovering that purpose, by identifying stakeholders and their needs, and documenting these in a form that is amenable to analysis, communication, and subsequent implementation. There are a number of inherent difficulties in this process. Stakeholders (including paying customers, users and developers) may be numerous and distributed. Their goals may vary and conflict, depending on their perspectives of the environment in which they work and the tasks they wish to accomplish. Their goals may not be explicit or may be difficult to articulate, and, inevitably, satisfaction of these goals may be constrained by a variety of factors outside their control.

In this paper we present an overview of current research in RE, presented in terms of the main activities that constitute the field. While these activities are described independently and in a particular order, in practice, they are actually interleaved, iterative, and may span the entire software systems development life cycle. Section 2 outlines the disciplines that provide the foundations for effective RE, while Section 3 briefly describes the context and background needed in order to begin the RE process. Sections 4 through 8 describe the core RE activities:

- *eliciting* requirements,
- *modelling* and *analysing* requirements,
- *communicating* requirements,
- *agreeing* requirements, and
- *evolving* requirements.

Section 9 discusses how these different activities may be integrated into a single development process. We conclude with a summary of the state of the art in RE, and offer our view of the key challenges for future RE research.

2 Foundations
Before discussing RE activities in more detail, it is worth examining the role of RE in software and systems engineering, and the many disciplines upon which it draws. Zave [83] provides one of the clearest definitions of RE:

> "Requirements engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behavior, and to their evolution over time and across software families."

This definition is attractive for a number of reasons. First, it highlights the importance of "real-world goals" that motivate the development of a software system. These represent the 'why' as well as the 'what' of a system. Second, it refers to "precise specifications". These provide the basis for *analysing* requirements, *validating* that they are indeed what stakeholders want, *defining* what designers have to build, and *verifying* that they have done so correctly upon delivery. Finally, the definition refers to specifications' "evolution over time and across software families", emphasising the reality of a changing world and the need to reuse partial specifications, as engineers often do in other branches of engineering.

It has been argued that requirements *engineering* is a misnomer. Typical textbook definitions of engineering refer to the creation of cost-effective solutions to practical problems by applying scientific knowledge [74]. Ther… the use of the term *engineering* in RE serves as… that RE is an important part of an eng… being the part concerned with … activities to a real-world…

Future of Sofware Engineering Limerick Ireland

Requirements engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints … software systems. It is also concerned with the relationship … ors to **<u>precise specifications of software behavi</u>**… …olution over time and across software families. *Zave*

RE is often regarded as a **<u>fron</u>**… …e software systems development process. *Nuseibeh and Easter*…

Yesterday's radical changes are today's norms

Historical F…
- Eli…
- …alyzing Requirements
- …cating Requirements
- …eeing Requirements
- Evolving Requirements
- Integrated Requirements Engineering

Radical Changes of the previous period
- Modeling and analysis within social context
- Shift from modeling information flow and state modeling goals and scenarios.
- Acknowledging that RE must accept inconsistencies, uncertainties, and conflicts.

Roadmap:
New techniques for formally modeling environment, bridging the gap between elicitation based on contextual enquiry and formal modeling, richer models for capturing NFRs & a focus on architectural impact, reuse of requirements models, and multidisciplinary training for practitioners.

Research Directions in Requirements Engineering (2007)

Research Directions in Requirements Engineering

Betty H.C. Cheng
Michigan State University
3115 Engineering Building
East Lansing, Michigan 48824 USA
chengb@cse.msu.edu

Joanne M. Atlee
University of Waterloo
200 University Ave. West
Waterloo, Ontario N2L 3G1 CANADA
jmatlee@uwaterloo.ca

Abstract

In this paper, w... ments engineer- ing (RE) researc... rections suggested by em... the state of the... ered with resp... cific requiren... analysis. Su... of research... tion. Next... extending RE... of future research dire... consider to be the "hot" curre... which aim to address RE needs for eme... future.

seibeh and Easterbrook's paper [127], hereafter referred to as the "2000 RE Roadmap Paper", from the Future of Software Engineering track at ICSE 2000 [64]. Whereas the 2000 RE Roadmap Paper focused on current research in requirements engineering, this paper concentrates on research ...ctions and identifies RE challenges posed by emerging ...software needs. We start, in Section 2, with an ...erent difficulties in requirements engi- ...provide a summary of the state ...arch, and in Section 4, ...advancing the ...ionary ... conclu... research infrasu... the research community ... addressing these problems.

1. Introduction

The success of a software system depends on how well it fits the needs of its users and its environment [127, 130]. *Software requirements* comprise these needs, and *requirements engineering (RE)* is the process by which the requirements are determined. Successful RE involves understanding the needs of users, customers, and other stakeholders; understanding the contexts in which the to-be-developed software will be used; modeling, analyzing, negotiating, and documenting the stakeholders' requirements; validating that the documented requirements match the negotiated requirements; and managing requirements evolution[1].

In this paper, we offer our views on the research directions in requirements engineering. The paper builds on Nu-

[1] In addition, there are a number of software-engineering activities that are based on requirements information, such as cost estimation, project planning, and requirements-based derivations of architectures, designs, code, and test cases. Although these activities are "related" to a system's requirements, they play at most a minor role in determining and agreeing on the system's requirements; as such, we consider them to be outside the scope of requirements engineering.

2. Why Requirements Engineering is Di...

In general, the research challenges faced by the requirements-engineering community are distinct from those faced by the general software-engineering community, because requirements reside primarily in the problem space whereas other software artifacts reside primarily in the solution space. That is, *requirements* descriptions, ideally, are written entirely in terms of the environment, describing how the environment is to be affected by the proposed system. In contrast, other software artifacts focus on the behavior of the proposed system, and are written in terms of internal software entities and properties. Stated another way, requirements engineering is about defining precisely the *problem* that the software is to solve (i.e., defining *what* the software is to do), whereas other SE activities are

Requirements engineering is about defining precisely the problem that the software is to solve. … RE Activities may be more iterative, involve many more players…, requirement more extensive analyses of options, and call for more complicated verification of more diverse components..

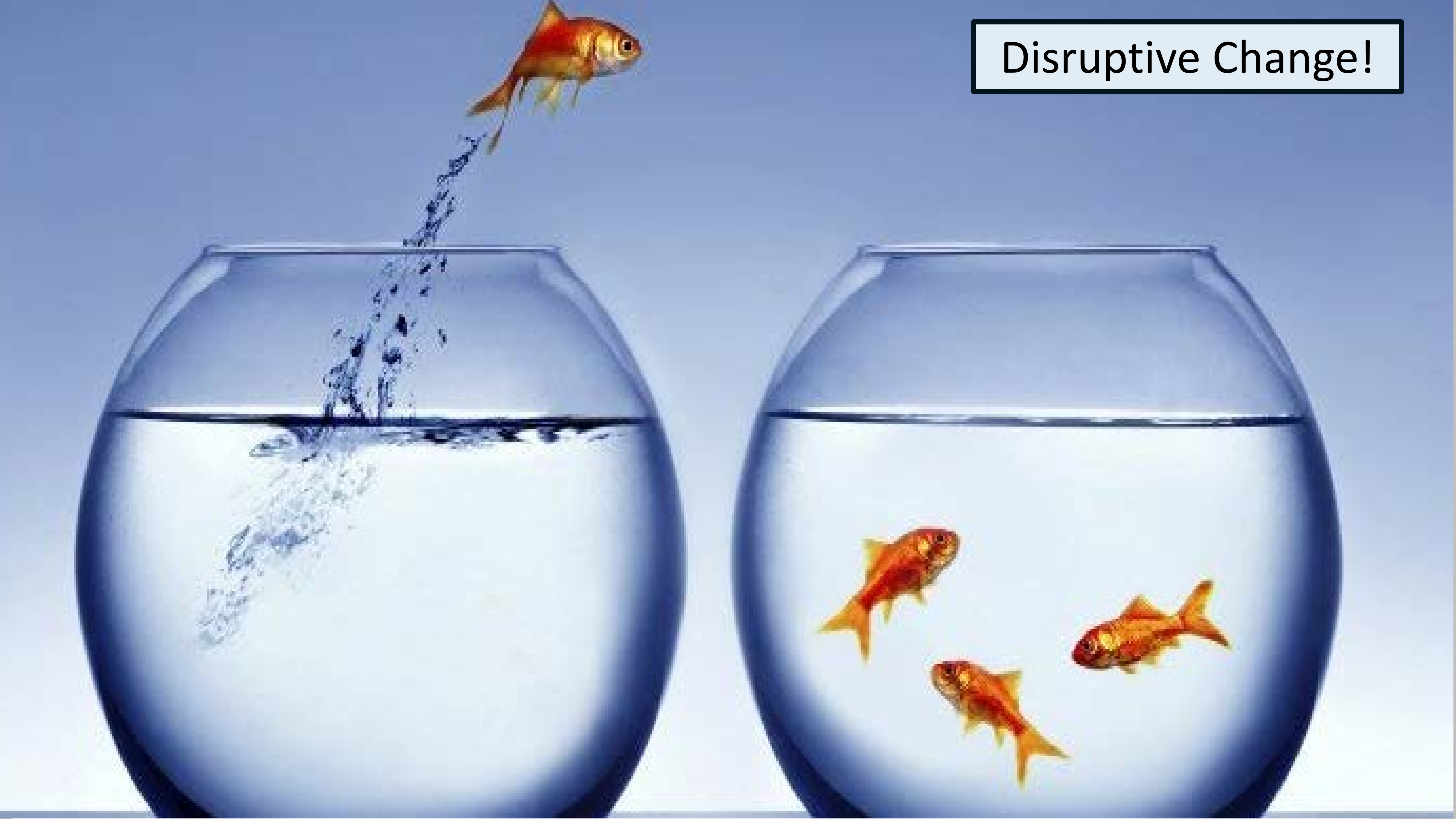
State of the Art
- Elicitation
- Modeling
- ...ments Analysis
- ...erification
- ...ent

Research Strategies
- Paradigm Shift
- Leverage other disciplines
- Leverage technology
- Evolutionary research
- Domain-specific
- Generalization
- ...ation

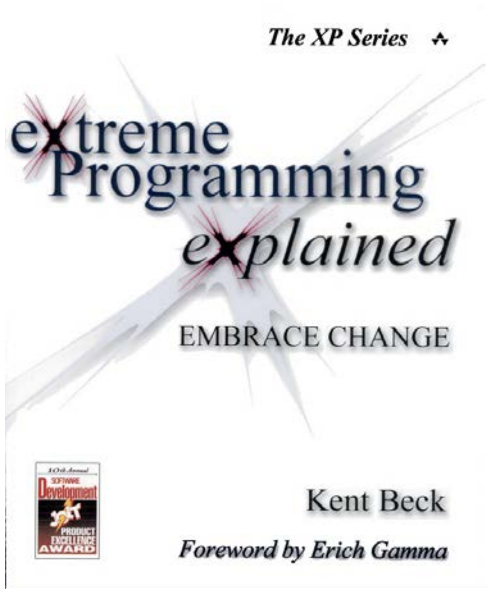
Hot Spots
Scale; security; tolerance; environmental ... globalization; methodologies, patterns, and tools; requirements ... Technologies.

Paradigm shifts are disruptive and transformational

- Paradigm Shift
- Leverage other disciplines


Disruptive Change!

We are bombarded with change on every side…

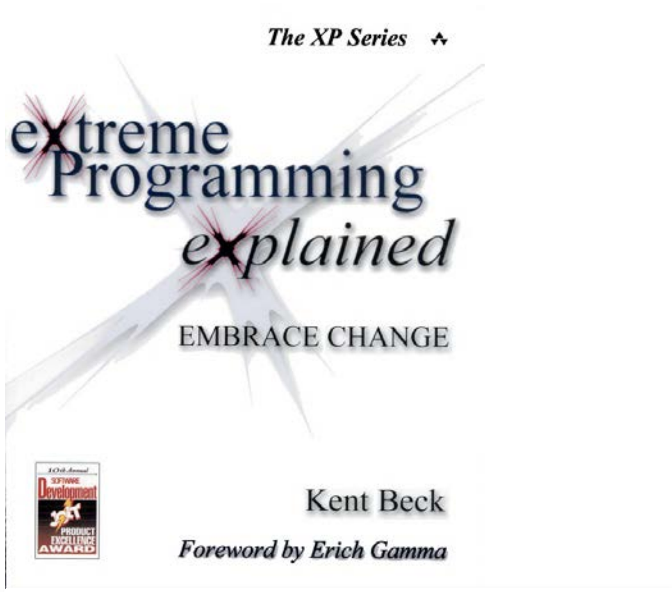


User Stories, Sprints, Scrum!

The year 2002…

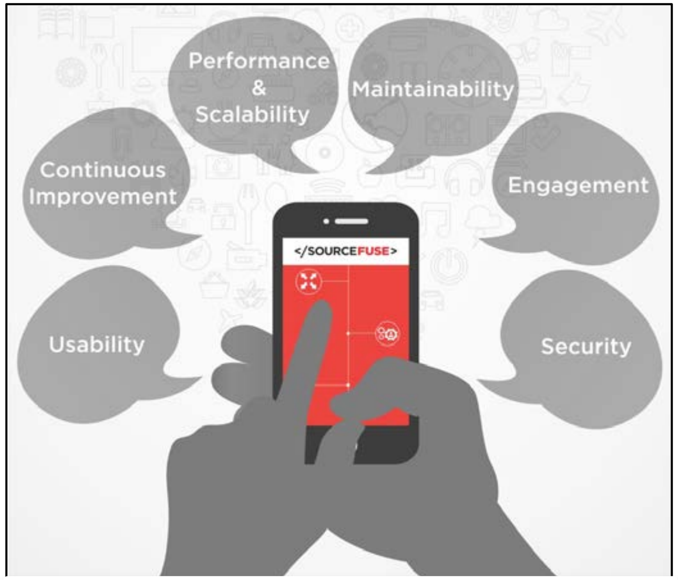
Requirements Engineering Conference, 2002, Essen.
Who believes that Agile is more than a passing trend?

We are bombarded with change on every side…



User Stories, Sprints, Scrum!

Elicitation vs. invention

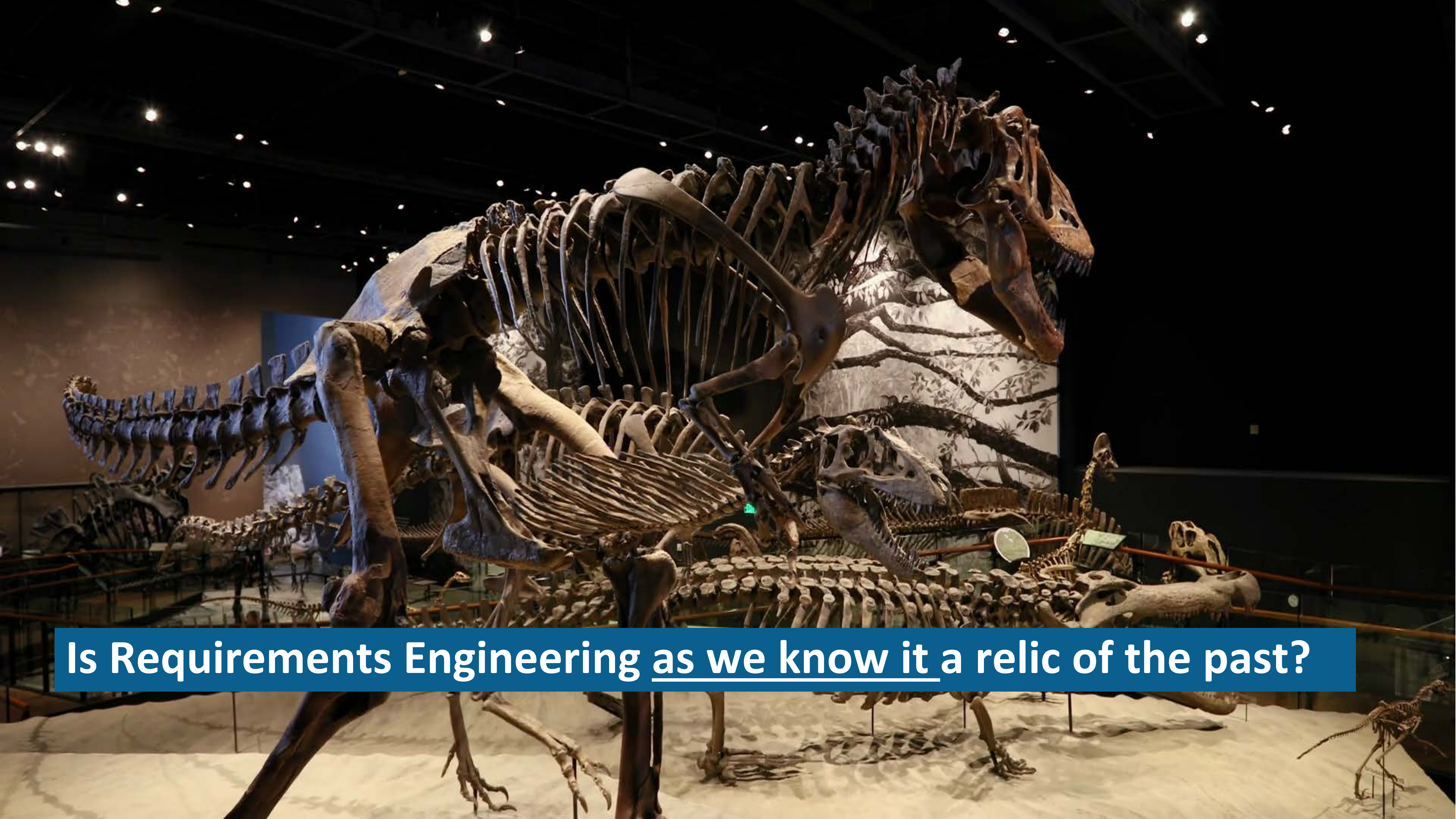


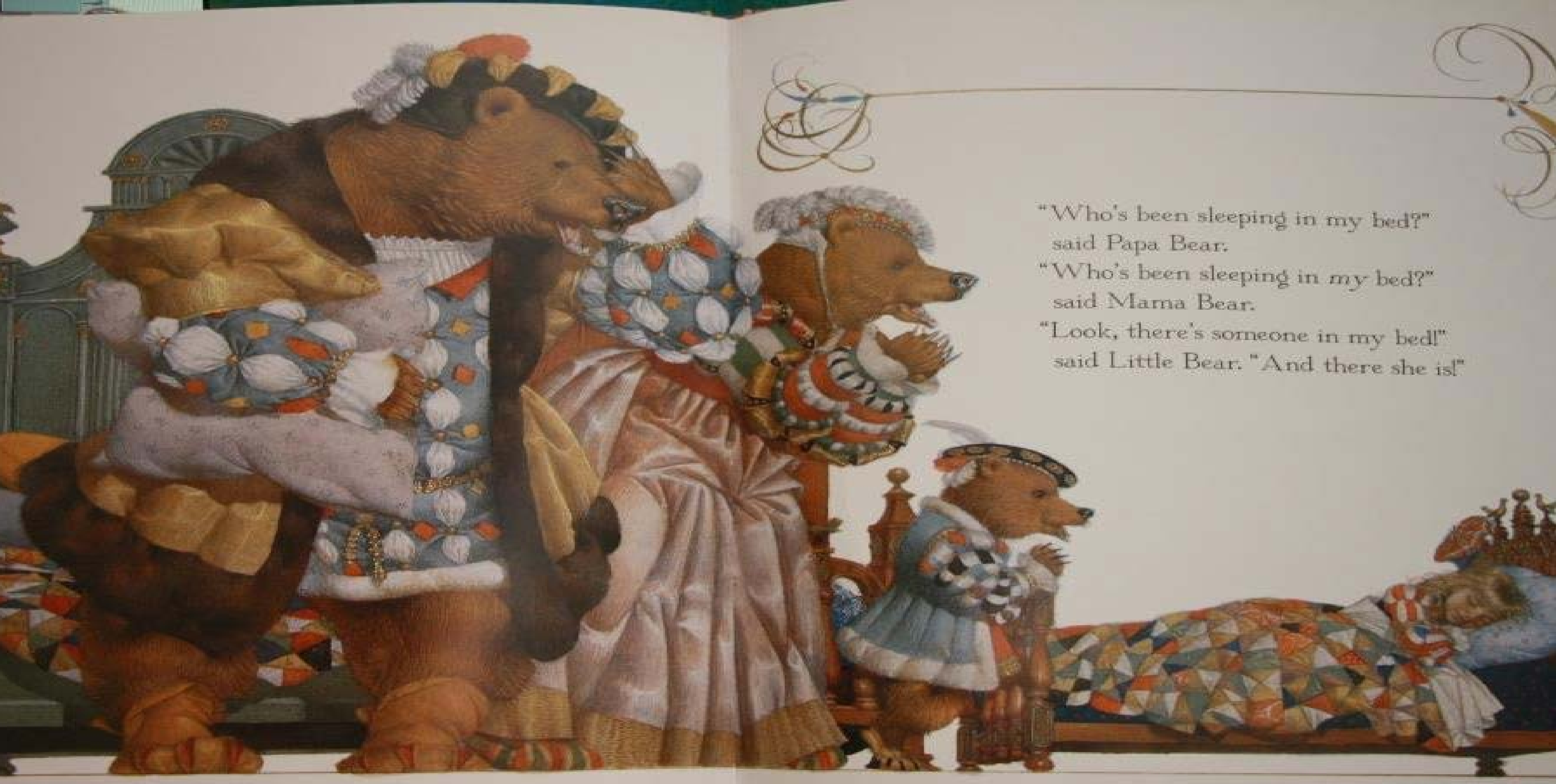
Requirements discovery by means of feature mining and app reviews.

The rise of CPS…

Where requirements are feature requests and bug reports!

Safety Critical projects seek increased agility..

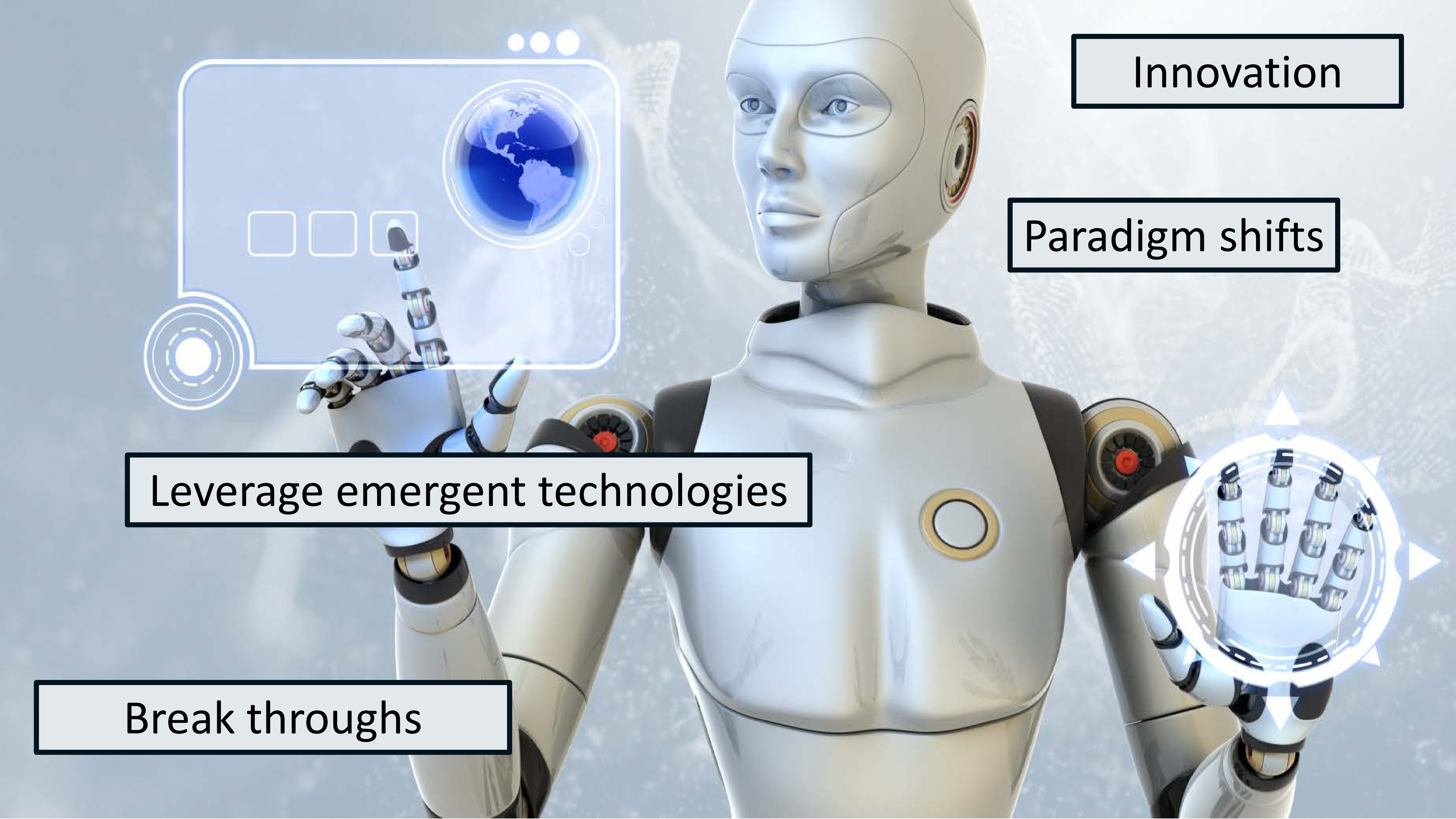

Is Requirements Engineering as we know it a relic of the past?



Or is this a Goldilocks Moment?

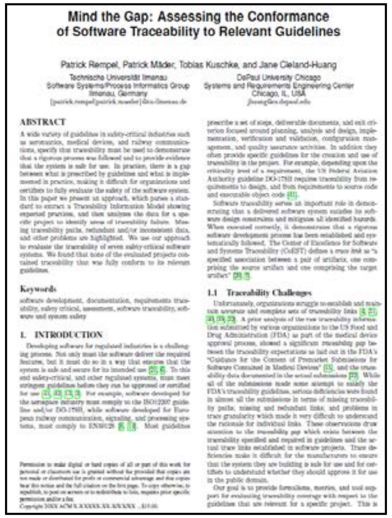
The Goldilocks Principle

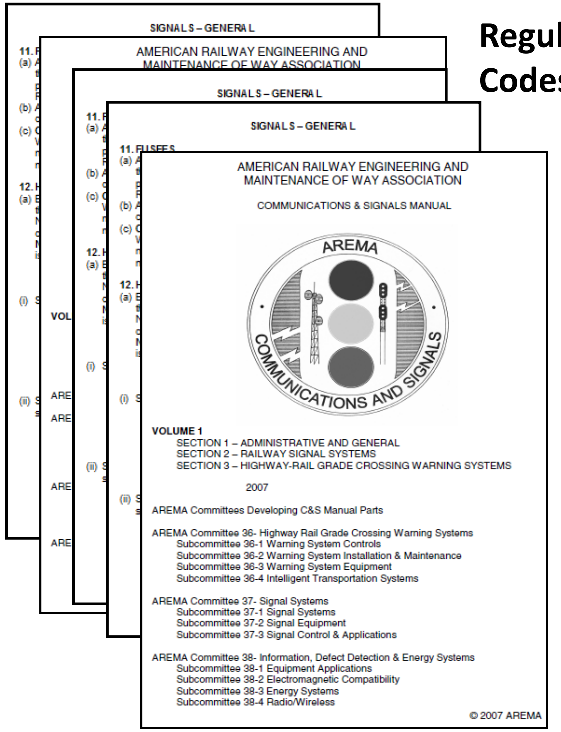
Conditions need to be 'just right' for transformative change.


Innovation
Paradigm shifts
Leverage emergent technologies
Break throughs

Traceability in a nutshell..

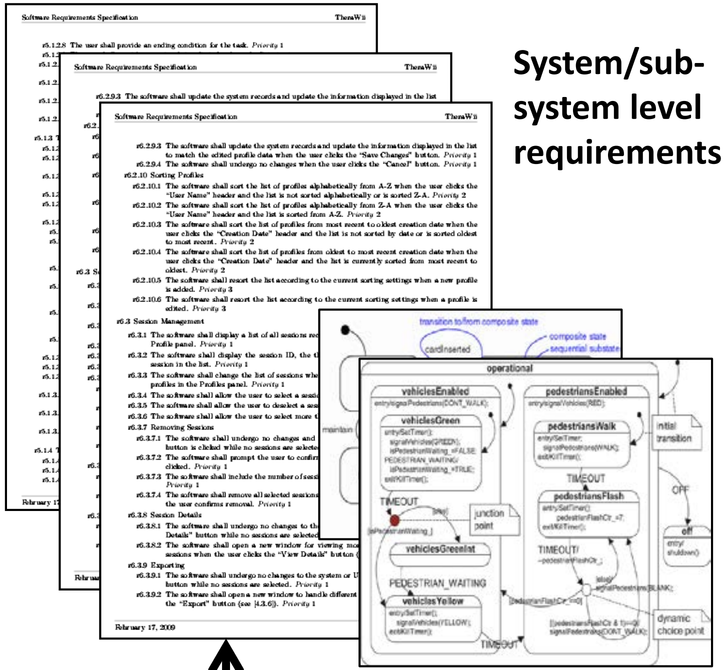
The ability to **interrelate any uniquely identifiable** software engineering artifact to any other, **maintain** required links over time, and **use the resulting network** to answer questions of both the software product and its development process.
- CoEST Definition





Regulatory Codes

Satisfies relevant codes



System/sub-system level requirements

Realizes

Validates

Code

Tests

Test cases (unit, integration, acceptance)

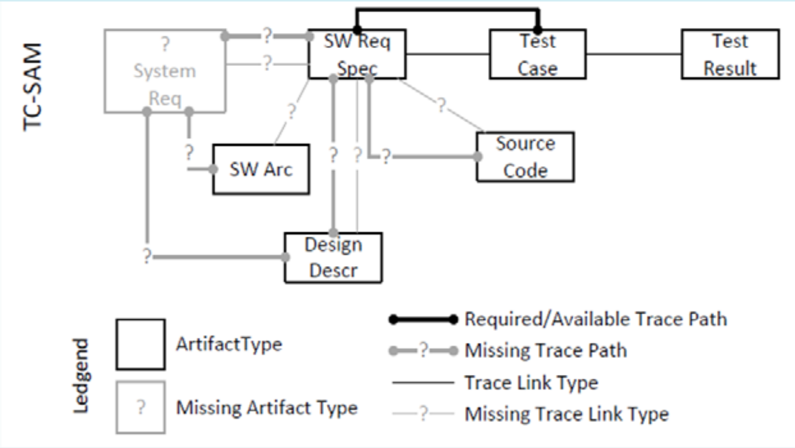
Required by many regulatory bodies and standards..

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**.

Margaret Hamilton's Keynote @ ICSE

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

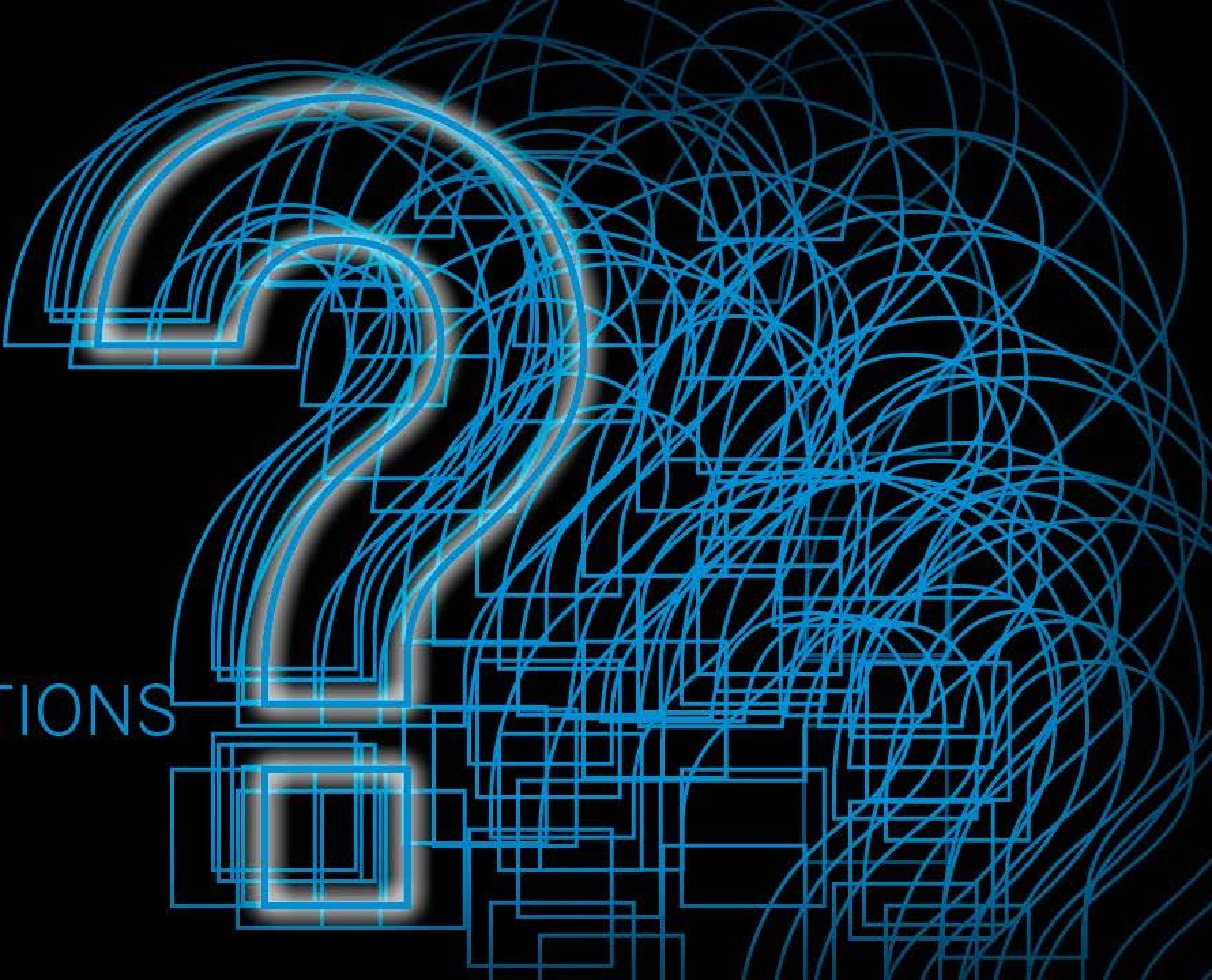
Why still? Not unlike 50 years ago when the field was brand new*. What to do...

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

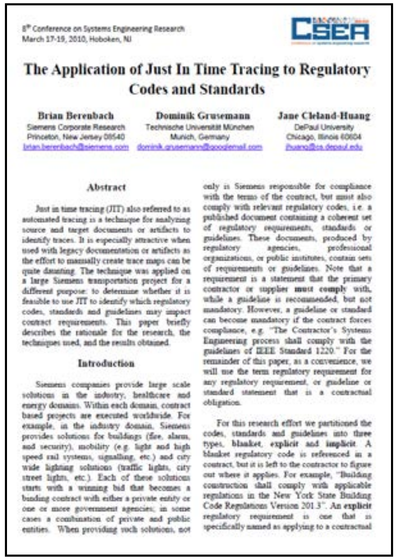
The programmer who wrote much of the code that took Apollo to the moon!

https://futurism.com/margaret-hamilton-the-untold-story-of-the-woman-who-took-us-to-the-moon/

HARDQUESTIONS



Question #1: Am I addressing an important problem?



8th Conference on Systems Engineering Research
March 17-19, 2010, Hoboken, NJ

The Application of Just In Time Tracing to Regulatory Codes and Standards

Brian Berenbach	Dominik Grusemann	Jane Cleland-Huang
Siemens Corporate Research	Technische Universität München	DePaul University
Princeton, New Jersey 08540	Munich, Germany	Chicago, Illinois 60604
brian.berenbach@siemens.com	dominik.grusemann@googlemail.com	jhuang@cs.depaul.edu

Abstract

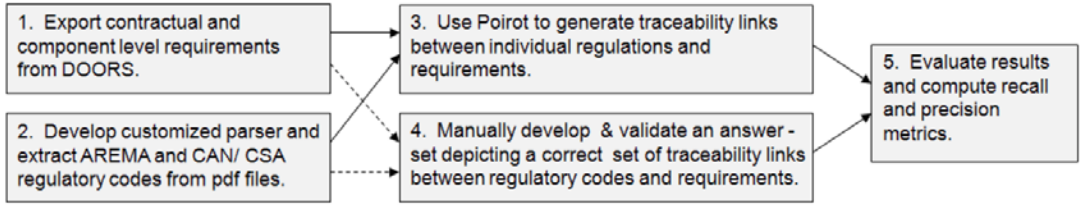
Just in time tracing (JIT) also referred to as automated tracing is a technique for analyzing source and target documents or artifacts to identify traces. It is especially attractive when used with legacy documentation or artifacts as the effort to manually create trace maps can be quite daunting. The technique was applied on a large Siemens transportation project for a different purpose: to determine whether it is feasible to use JIT to identify which regulatory codes, standards and guidelines may impact contract requirements. This paper briefly describes the rationale for the research, the techniques used, and the results obtained.

Introduction

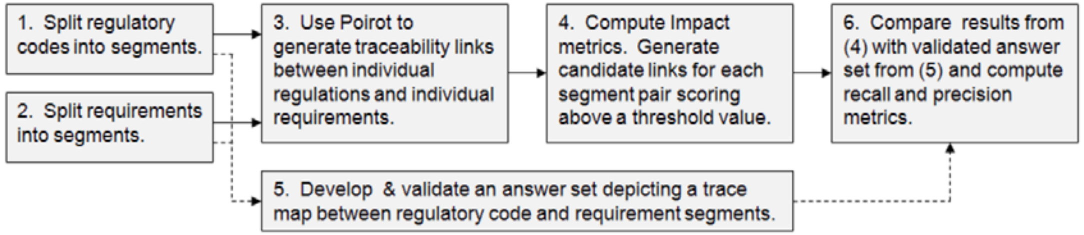
Siemens companies provide large scale solutions in the industry, healthcare and energy domains. Within each domain, contract based projects are executed worldwide. For example, in the industry domain, Siemens provides solutions for buildings (fire, alarm, and security), mobility (e.g. light and high speed rail systems, signalling, etc.) and city wide lighting solutions (traffic lights, city street lights, etc.). Each of these solutions starts with a winning bid that becomes a binding contract with either a private entity or one or more government agencies; in some cases a combination of private and public entities. When providing such solutions, not only is Siemens responsible for compliance with the terms of the contract, but must also comply with relevant regulatory codes, i.e. a published document containing a coherent set of regulatory requirements, standards or guidelines. These documents, produced by regulatory agencies, professional organizations, or public institutes, contain sets of requirements or guidelines. Note that a requirement is a statement that the primary contractor or supplier **must comply** with, while a guideline is recommended, but not mandatory. However, a guideline or standard can become mandatory if the contract forces compliance, e.g. "The Contractor's Systems Engineering process shall comply with the guidelines of IEEE Standard 1220." For the remainder of this paper, as a convenience, we will use the term regulatory requirement for any regulatory requirement, or guideline or standard statement that is a contractual obligation.

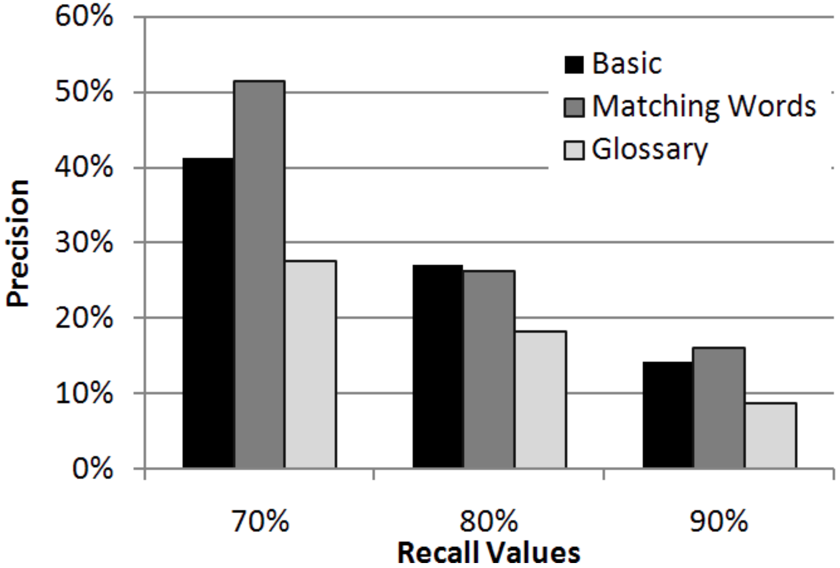
For this research effort we partitioned the codes, standards and guidelines into three types, **blanket**, **explicit** and **implicit**. A blanket regulatory code is referenced in a contract, but it is left to the contractor to figure out where it applies. For example, "Building construction shall comply with applicable regulations in the New York State Building Code Regulations Version 201.3". An **explicit** regulatory requirement is one that is specifically named as applying to a contractual

Experiment 1: Accuracy of links



Experiment 2: Identifying applicable segments of requirements and codes



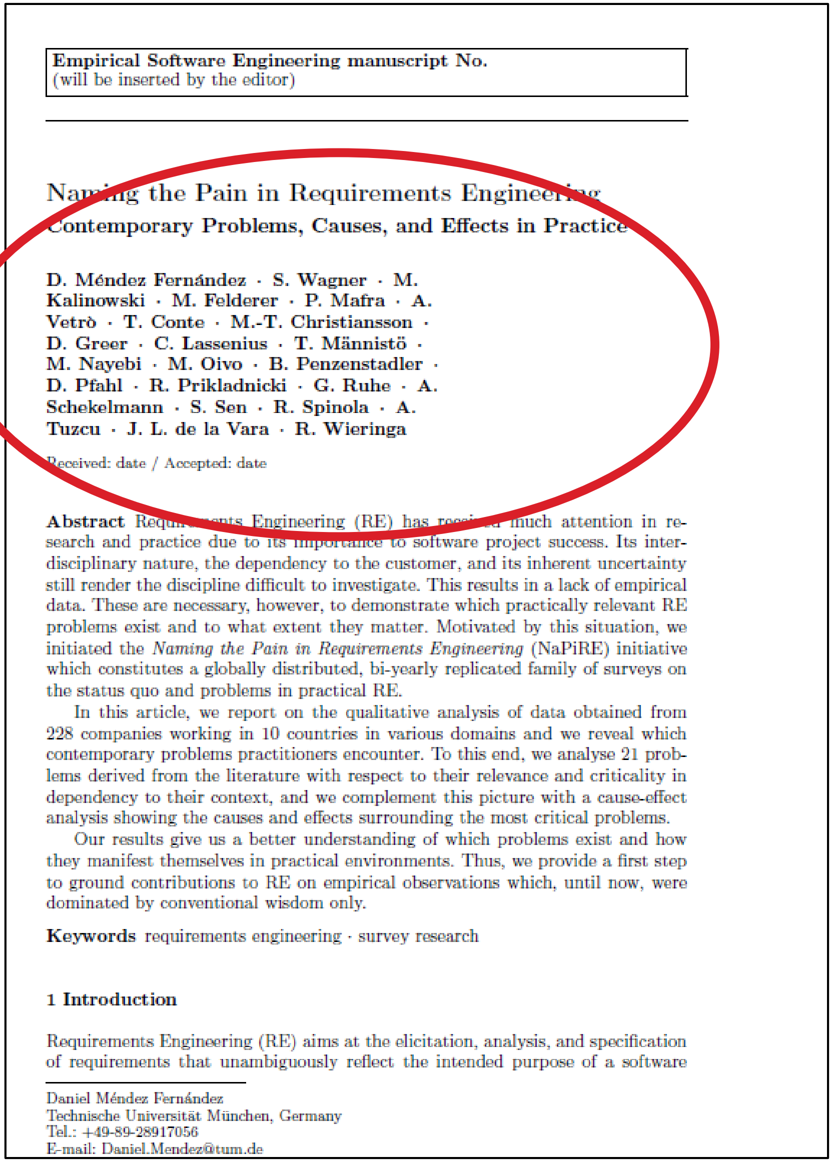


About 80% accuracy.

5 things industry told me when I listened

- Traceability is one of the most pressing problems we face.
- Traceability is a made-up problem!
- You are solving the ‘wrong’ part of the problem.
- Your solutions don’t scale.
- You are the expert. Give us ready-to-use tools, now!

Others have listened too..



Empirical Software Engineering manuscript No.
(will be inserted by the editor)

Naming the Pain in Requirements Engineering

Contemporary Problems, Causes, and Effects in Practice

D. Méndez Fernández · S. Wagner · M. Kalinowski · M. Felderer · P. Mafra · A. Vetrò · T. Conte · M.-T. Christiansson · D. Greer · C. Lassenius · T. Männistö · M. Nayebi · M. Oivo · B. Penzenstadler · D. Pfahl · R. Prikladnicki · G. Ruhe · A. Schekelmann · S. Sen · R. Spinola · A. Tuzcu · J. L. de la Vara · R. Wieringa

Received: date / Accepted: date

Abstract Requirements Engineering (RE) has received much attention in research and practice due to its importance to software project success. Its interdisciplinary nature, the dependency to the customer, and its inherent uncertainty still render the discipline difficult to investigate. This results in a lack of empirical data. These are necessary, however, to demonstrate which practically relevant RE problems exist and to what extent they matter. Motivated by this situation, we initiated the *Naming the Pain in Requirements Engineering* (NaPiRE) initiative which constitutes a globally distributed, bi-yearly replicated family of surveys on the status quo and problems in practical RE.

In this article, we report on the qualitative analysis of data obtained from 228 companies working in 10 countries in various domains and we reveal which contemporary problems practitioners encounter. To this end, we analyse 21 problems derived from the literature with respect to their relevance and criticality in dependency to their context, and we complement this picture with a cause-effect analysis showing the causes and effects surrounding the most critical problems.

Our results give us a better understanding of which problems exist and how they manifest themselves in practical environments. Thus, we provide a first step to ground contributions to RE on empirical observations which, until now, were dominated by conventional wisdom only.

Keywords requirements engineering · survey research

1 Introduction

Requirements Engineering (RE) aims at the elicitation, analysis, and specification of requirements that unambiguously reflect the intended purpose of a software

Daniel Méndez Fernández
Technische Universität München, Germany
Tel.: +49-89-28917056
E-mail: Daniel.Mendez@tum.de

Lessons learned

- Lack of time
- Lack of experience of RE team members
- Weak qualification of RE team members
- Communication flaws between project team and the customer
- Requirements remain too abstract
- Changing business needs
- Customer does not know what he wants
- Missing direct communication to customer
- Language barriers
- Strict time schedule by customer

How should I as a Requirements Engineering Researcher respond to these pain points?

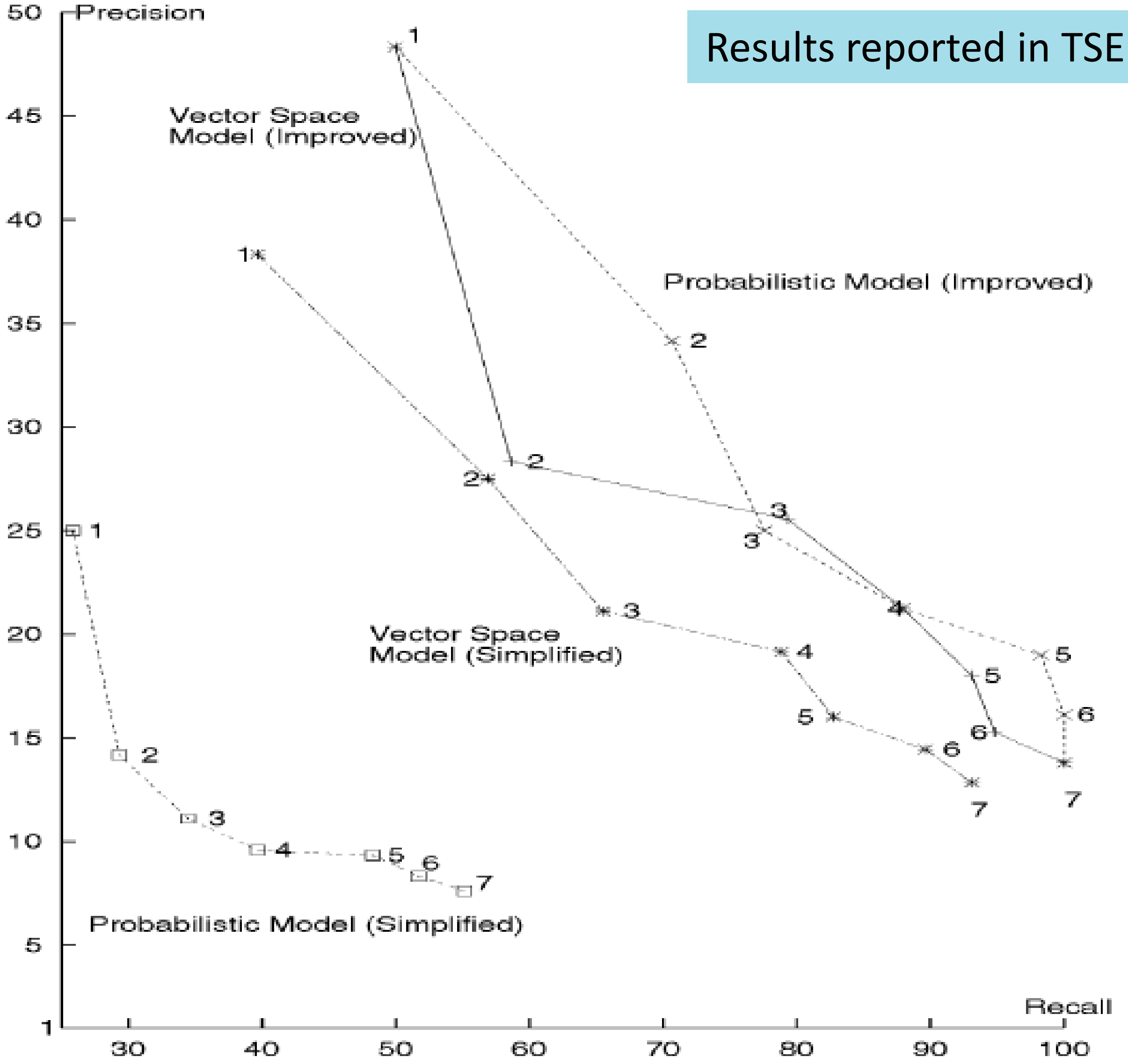
Reference the paper

Claims:

- Provides insights into industrial RE problem trends
- Helps to lay the foundation to steer academic and industrial research in a problem-driven manner where scientic contributions to RE can be put in tune with practically relevant problems.

Question #2: Am I making progress towards the goal?

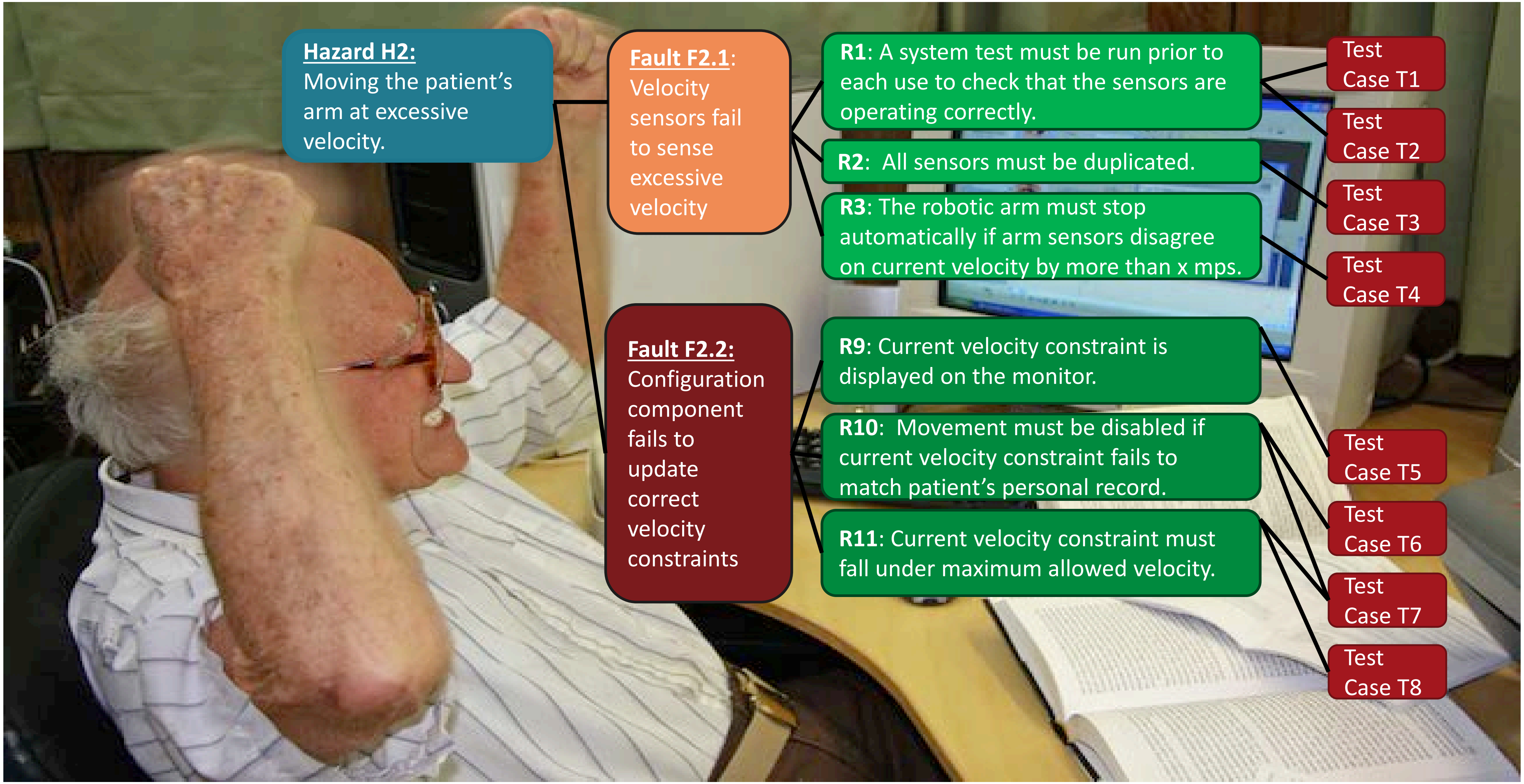
Giulio Antoniol with others…



Results reported in TSE in 2002.

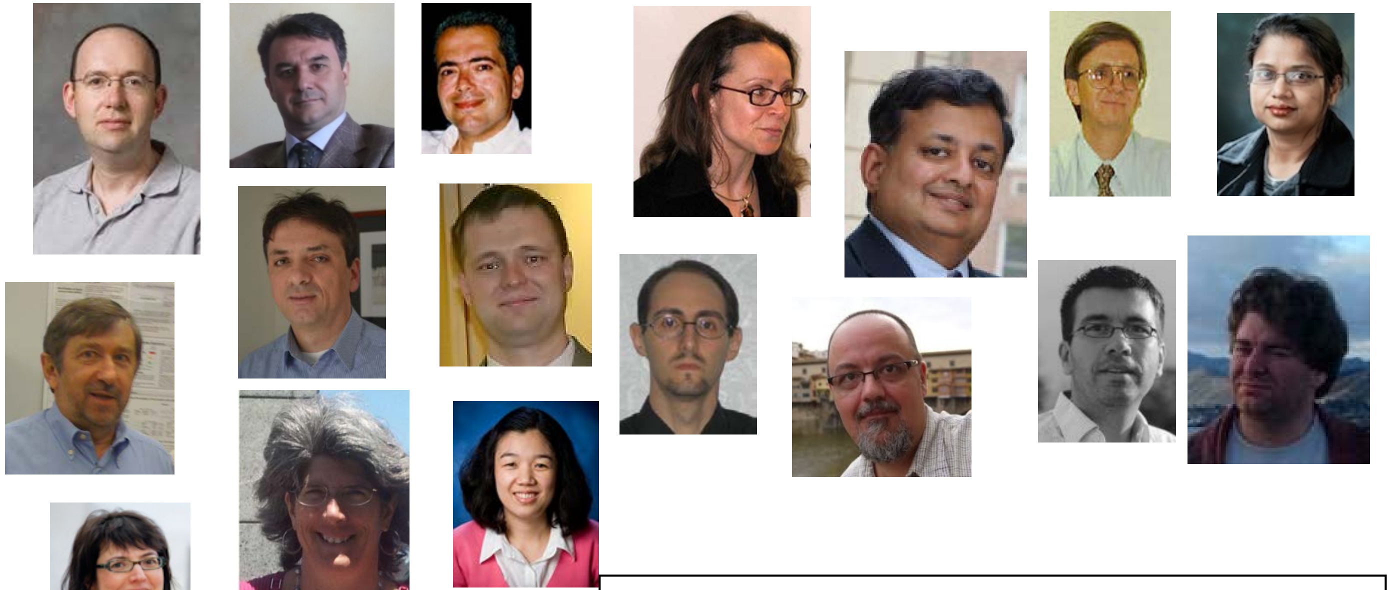
<u>Seminal work</u> in 2001 launched a new research direction – the quest to automate the traceability process – followed shortly thereafter by work at RE by Jane Hayes and Alex Dehktyar

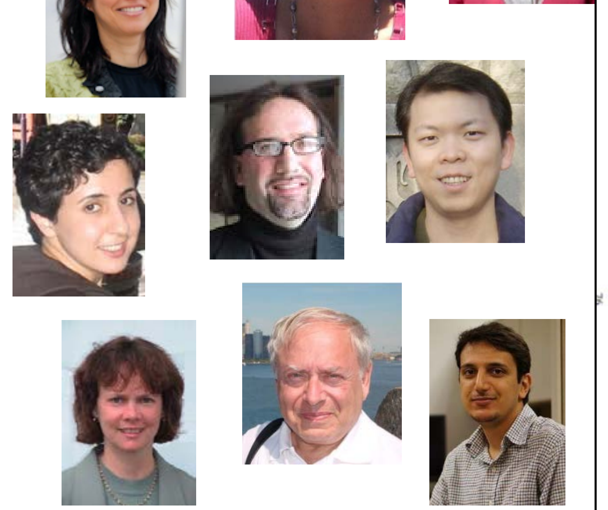
Ask this guy!

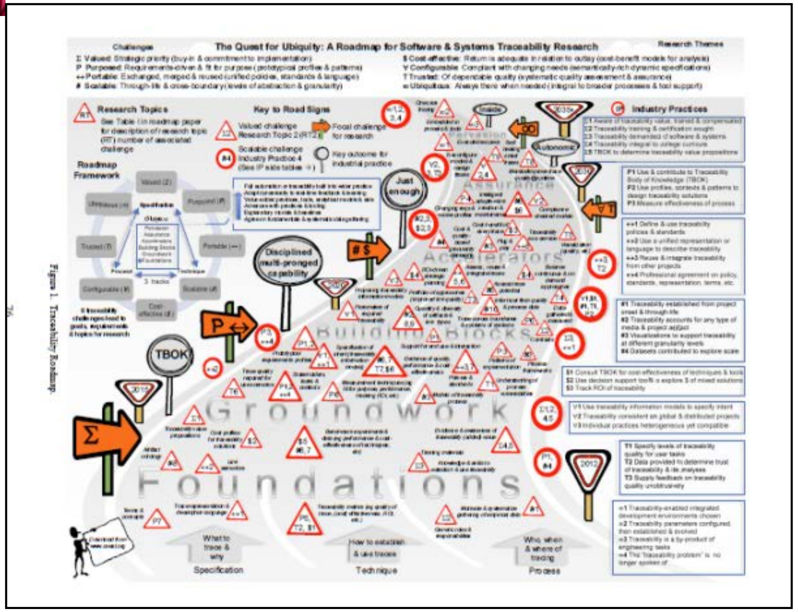


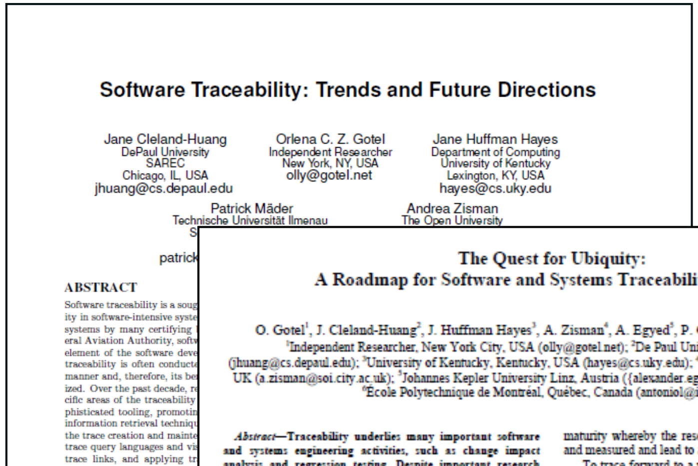
Hazard H2: Moving the patient's arm at excessive velocity.
Fault F2.1: Velocity sensors fail to sense excessive velocity
R1: A system test must be run prior to each use to check that the sensors are operating correctly.
R2: All sensors must be duplicated.
R3: The robotic arm must stop automatically if arm sensors disagree on current velocity by more than x mps.
Fault F2.2: Configuration component fails to update correct velocity constraints
R9: Current velocity constraint is displayed on the monitor.
R10: Movement must be disabled if current velocity constraint fails to match patient's personal record.
R11: Current velocity constraint must fall under maximum allowed velocity.
Test Case T1
Test Case T2
Test Case T3
Test Case T4
Test Case T5
Test Case T6
Test Case T7
Test Case T8

Roadmaps help set directions









Software Traceability: Trends and Future Directions

Jane Cleland-Huang
DePaul University
SAREC
Chicago, IL, USA
jhuang@cs.depaul.edu

Orlena C. Z. Gotel
Independent Researcher
New York, NY, USA
olly@gotel.net

Jane Huffman Hayes
Department of Computing
University of Kentucky
Lexington, KY, USA
hayes@cs.uky.edu

Patrick Mäder
Technische Universität Ilmenau

Andrea Zisman
The Open University

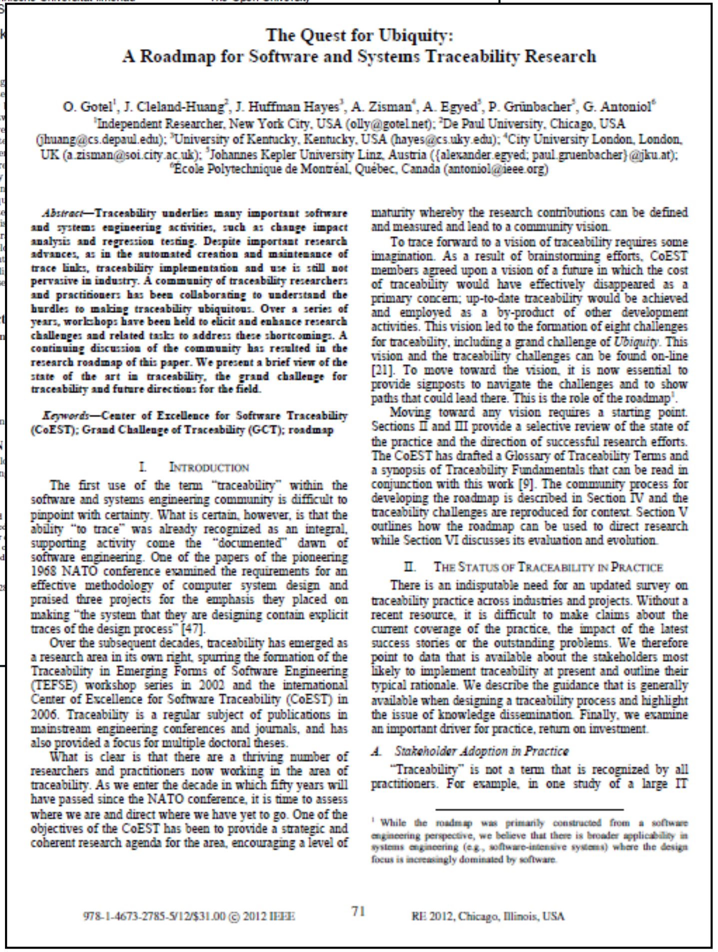
ABSTRACT

Categories and Subject

General Terms
Documentation

Keywords

1. INTRODUCTION



The Quest for Ubiquity:
A Roadmap for Software and Systems Traceability Research

O. Gotel[1], J. Cleland-Huang[2], J. Huffman Hayes[3], A. Zisman[4], A. Egyed[5], P. Grünbacher[5], G. Antoniol[6]
[1]Independent Researcher, New York City, USA (olly@gotel.net); [2]De Paul University, Chicago, USA (jhuang@cs.depaul.edu); [3]University of Kentucky, Kentucky, USA (hayes@cs.uky.edu); [4]City University London, London, UK (a.zisman@soi.city.ac.uk); [5]Johannes Kepler University Linz, Austria ({alexander.egyed; paul.gruenbacher}@jku.at); [6]École Polytechnique de Montréal, Québec, Canada (antoniol@ieee.org)

Abstract—Traceability underlies many important software and systems engineering activities, such as change impact analysis and regression testing. Despite important research advances, as in the automated creation and maintenance of trace links, traceability implementation and use is still not pervasive in industry. A community of traceability researchers and practitioners has been collaborating to understand the hurdles to making traceability ubiquitous. Over a series of years, workshops have been held to elicit and enhance research challenges and related tasks to address these shortcomings. A continuing discussion of the community has resulted in the research roadmap of this paper. We present a brief view of the research roadmap of this paper. We present a brief view of the state of the art in traceability, the grand challenge for traceability and future directions for the field.

Keywords—Center of Excellence for Software Traceability (CoEST); Grand Challenge of Traceability (GCT); roadmap

I. Introduction

The first use of the term "traceability" within the software and systems engineering community is difficult to pinpoint with certainty. What is certain, however, is that the ability "to trace" was already recognized as an integral, supporting activity come the "documented" dawn of software engineering. One of the papers of the pioneering 1968 NATO conference examined the requirements for an effective methodology of computer system design and praised three projects for the emphasis they placed on making "the system that they are designing contain explicit traces of the design process" [47].

Over the subsequent decades, traceability has emerged as a research area in its own right, spurring the formation of the Traceability in Emerging Forms of Software Engineering (TEFSE) workshop series in 2002 and the international Center of Excellence for Software Traceability (CoEST) in 2006. Traceability is a regular subject of publications in mainstream engineering conferences and journals, and has also provided a focus for multiple doctoral theses.

What is clear is that there are a thriving number of researchers and practitioners now working in the area of traceability. As we enter the decade in which fifty years will have passed since the NATO conference, it is time to assess where we are and direct where we have yet to go. One of the objectives of the CoEST has been to provide a strategic and coherent research agenda for the area, encouraging a level of maturity whereby the research contributions can be defined and measured and lead to a community vision.

To trace forward to a vision of traceability requires some imagination. As a result of brainstorming efforts, CoEST members agreed upon a vision of a future in which the cost of traceability would have effectively disappeared as a primary concern; up-to-date traceability would be achieved and employed as a by-product of other development activities. This vision led to the formation of eight challenges for traceability, including a grand challenge of *Ubiquity*. This vision and the traceability challenges can be found on-line [21]. To move toward the vision, it is now essential to provide signposts to navigate the challenges and to show paths that could lead there. This is the role of the roadmap[1].

Moving toward any vision requires a starting point. Sections II and III provide a selective review of the state of the practice and the direction of successful research efforts. The CoEST has drafted a Glossary of Traceability Terms and a synopsis of Traceability Fundamentals that can be read in conjunction with this work [9]. The community process for developing the roadmap is described in Section IV and the traceability challenges are reproduced for context. Section V outlines how the roadmap can be used to direct research while Section VI discusses its evaluation and evolution.

II. The Status of Traceability in Practice

There is an indisputable need for an updated survey on traceability practice across industries and projects. Without a recent resource, it is difficult to make claims about the current coverage of the practice, the impact of the latest success stories or the outstanding problems. We therefore point to data that is available about the stakeholders most likely to implement traceability at present and outline their typical rationale. We describe the guidance that is generally available when designing a traceability process and highlight the issue of knowledge dissemination. Finally, we examine an important driver for practice, return on investment.

A. Stakeholder Adoption in Practice

"Traceability" is not a term that is recognized by all practitioners. For example, in one study of a large IT

[1] While the roadmap was primarily constructed from a software engineering perspective, we believe that there is broader applicability in systems engineering (e.g., software-intensive systems) where the design focus is increasingly dominated by software.

978-1-4673-2785-5/12/$31.00 © 2012 IEEE 71 RE 2012, Chicago, Illinois, USA

Open Research Questions

Planning and Managing

Planning and managing is at the **heart of all other areas of the traceability life-cycle**.

What tasks do people need traceability to support?

What is the role of traceability in each of those tasks?

Research Directions

RD-1.1 Develop prototy
including scenarios of u

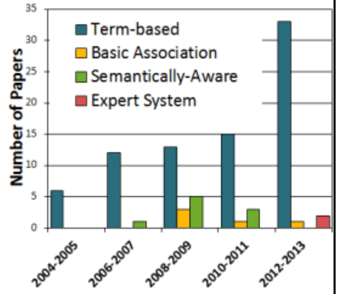
RD-1.2 Empirically valid
by stakeholders.

Planning and Managing

Knowledge Reuse

RD-2.1 Identify ingredients for through-life traceability success in different contexts, from a thorough understanding of industry best and worst practice, and then use this knowledge to establish a proces
standa
(Purpo

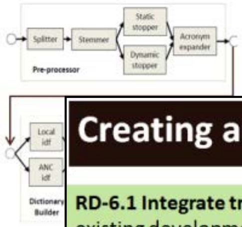
RD- 3.1 More Intelligence.....



Hypothesis: Real advancements, that **make a** **…erence** to the traceability problem, will … be achieved as we transition towards **…e intelligent traceability** solutions.

RD- 3.3 Self Adapting Solutions

Self-aware systems are able to modify their own behavior in an attempt to optimize performance. Such systems can self-diagnose, self-repair, adapt, add or remove software components dynamically.



RD-3.3
projec
trace l

RD- 3.2: Leave no exhaust

Develop techniques that monitor the environment and human activities – and use this information to infer new trace

…utions that are capable of
…ding artifacts and human

Maintaining Trace Links

While traceability is touted for its ability to support change, **trace maintenance actually adds work and can impede change**. Furthermore, trace links are brittle and break easily.

RD-4.1 Understand patterns of change across

Trusting Links

As we cannot guarantee complete and accurate traceability, we should devise techniques for clearly communicating confidence levels to the stakeholders.

RD-5.1 Develop human-centric tools to support link vetting.

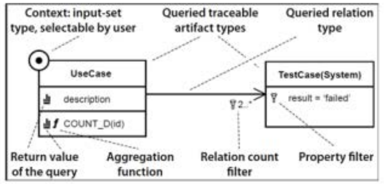
RD-5.2 Develop algorithms and supporting tools for automatically evaluating the correctness of existing trace links, whether created manually or with tool-support.

RD-5.3 Create visual dashboards to visualize the traceability quality of a project.

Creating and Using Trace **Queries**

RD-6.1 Integrate traceability into existing development tools

RD-6.2 Provide intuitive forms of query mechanism including visual languages and natural language interfaces.

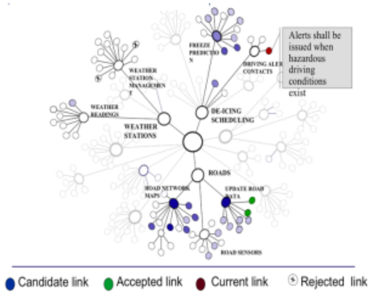


Visualizing and Understanding Results

Enormous advances have been made in popular techniques and tools for information and knowledge visualization.

Visual analytics are now a common form of support for decision-making activities in many fields of endeavor.

Despite some pockets of research, our field has been slow to keep pace, and must re-examine its information-seeking needs and mechanisms.



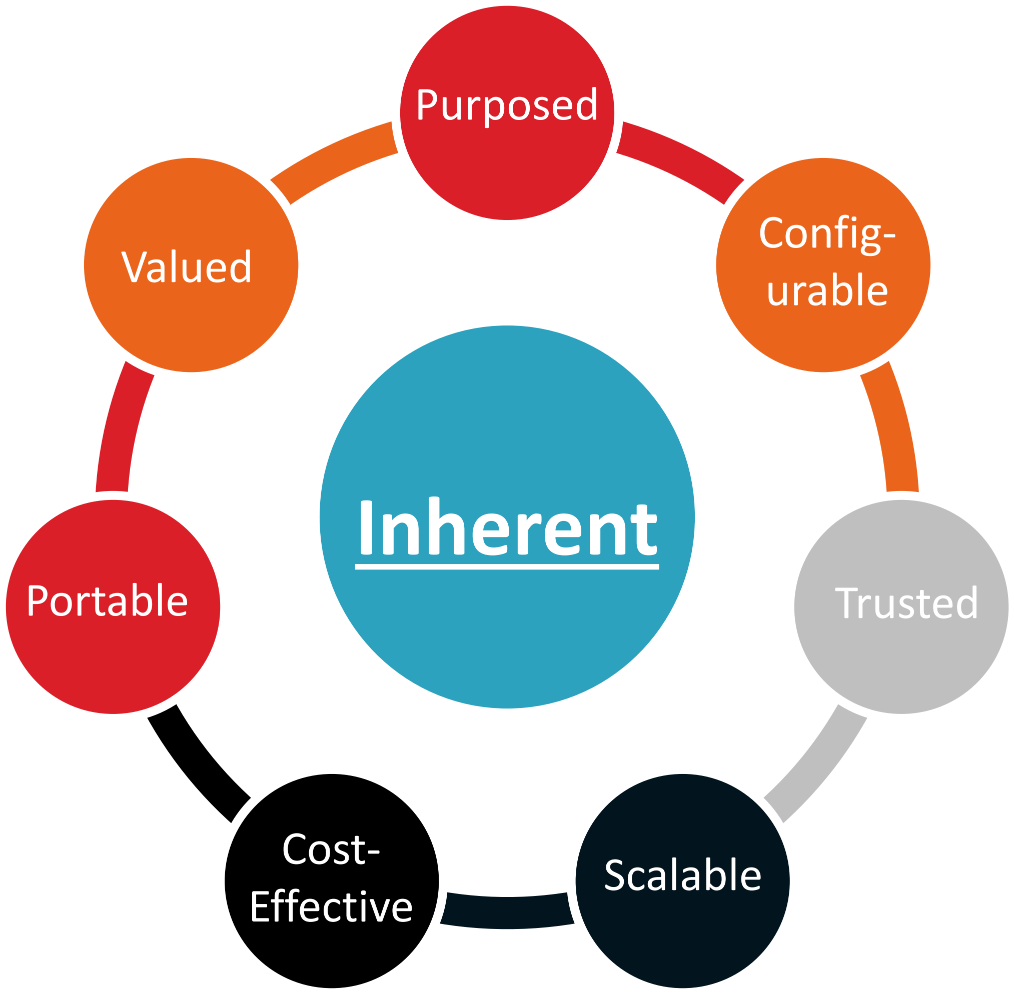
RD-7.1 Construct a taxonomy of available visualizations and fundamental traceability tasks. Bridge these by exploring the basic visualization principles that they either provide or demand.

RD-7.2 Gather and share user-based empirical data to evaluate trace visualizations...

RD-7.3 Perform in-situ user studies to evaluate and understand task-specific needs for trace information.

Jane Cleland-Huang, Orlena Gotel, Jane Huffman Hayes, Patrick Mäder, Andrea Zisman:
Software traceability: trends and future directions. FOSE 2014: 55-69

The Grand Challenge of Traceability



Inherent. Traceability is always there, without having to think about getting it there. Traceability is neither consciously established nor sought; it is built-in and effortless. It has effectively 'disappeared without a trace.'

How do we measure that?

Total automation of trace creation and trace maintenance, with quality and performance levels superior to manual efforts.

Automated Trace Link Creation and Maintenance

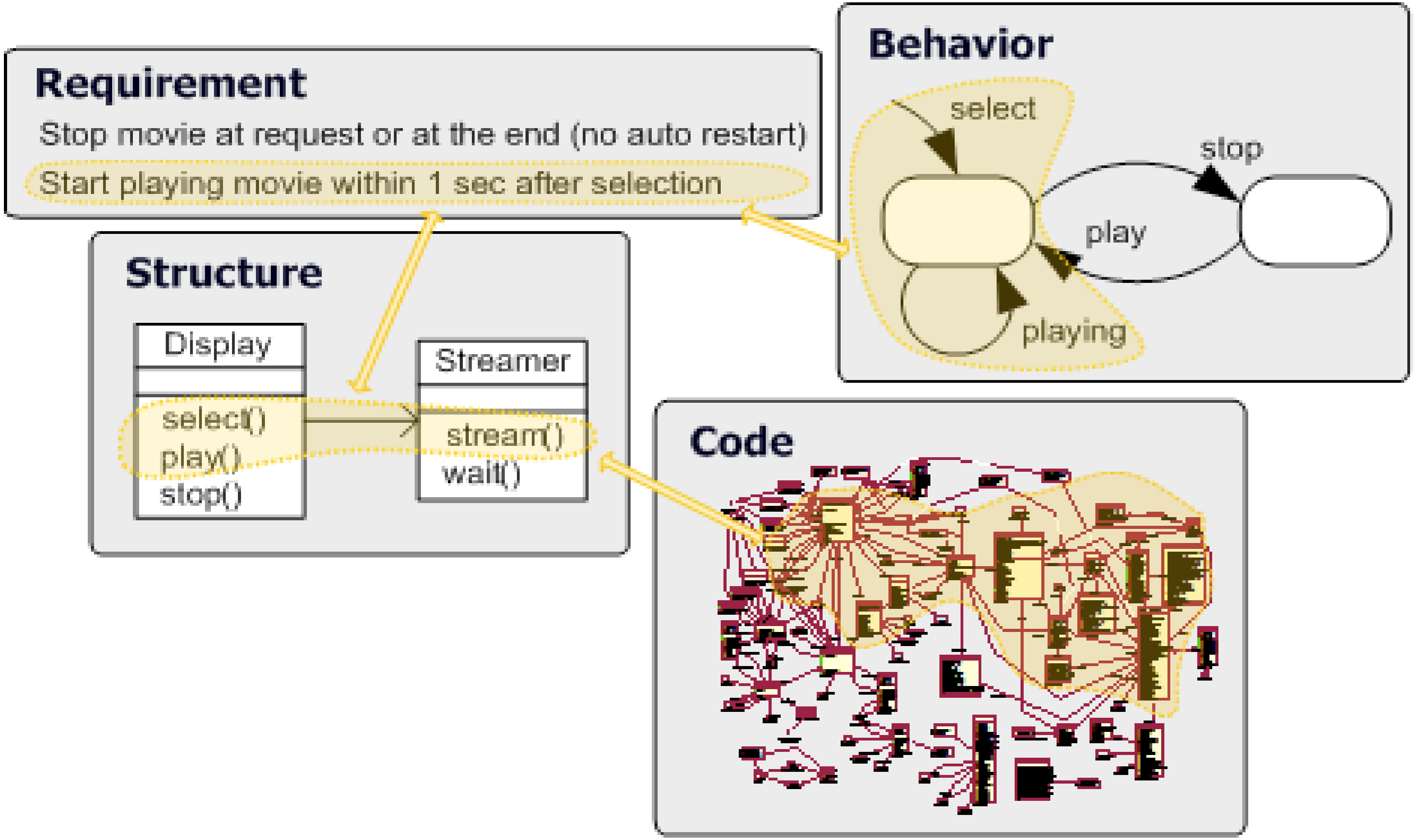


Figure courtesy of Alex Egyed, University of Vienna.

Establish Measurable goals

We need to know where we are going!

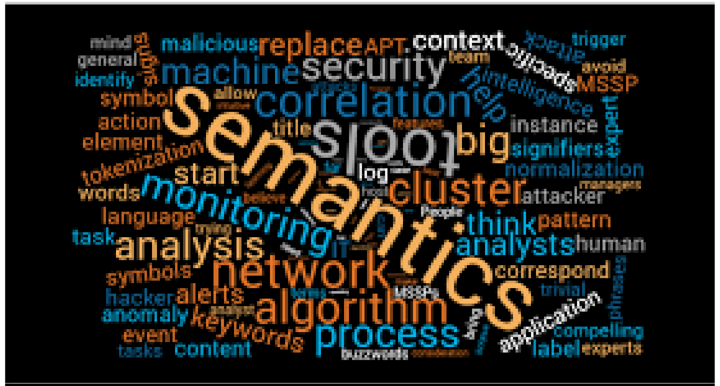
Define meaningful measures!

- RD-3.1 Develop intelligent tracing solutions which are not constrained by the terms in source and target artifacts, but which **understand domain-specific concepts**, and can reason intelligently about relationships between artifacts.
- RD-3.1 Deliver **prospective trace capture solutions** that are capable of monitoring development environments, including artifacts and human activities, to infer trace links.
- RD-3.3 Adopt **self adapting solutions** which are aware of the current project state and **reconfigure** accordingly in order to optimize the quality of trace links.

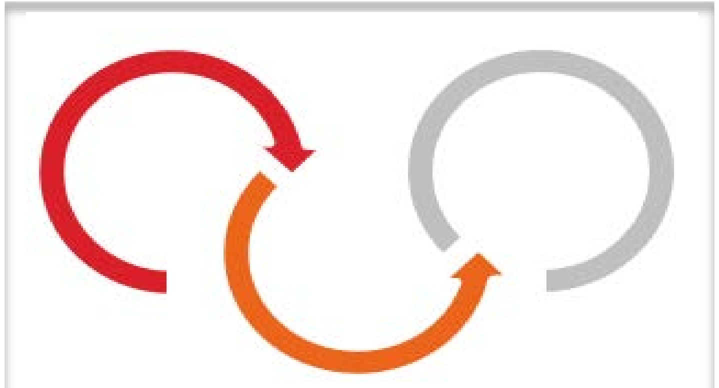
Question #3: Is there a trajectory to a real solution?

Our response

Yes – but we are probably not on it!



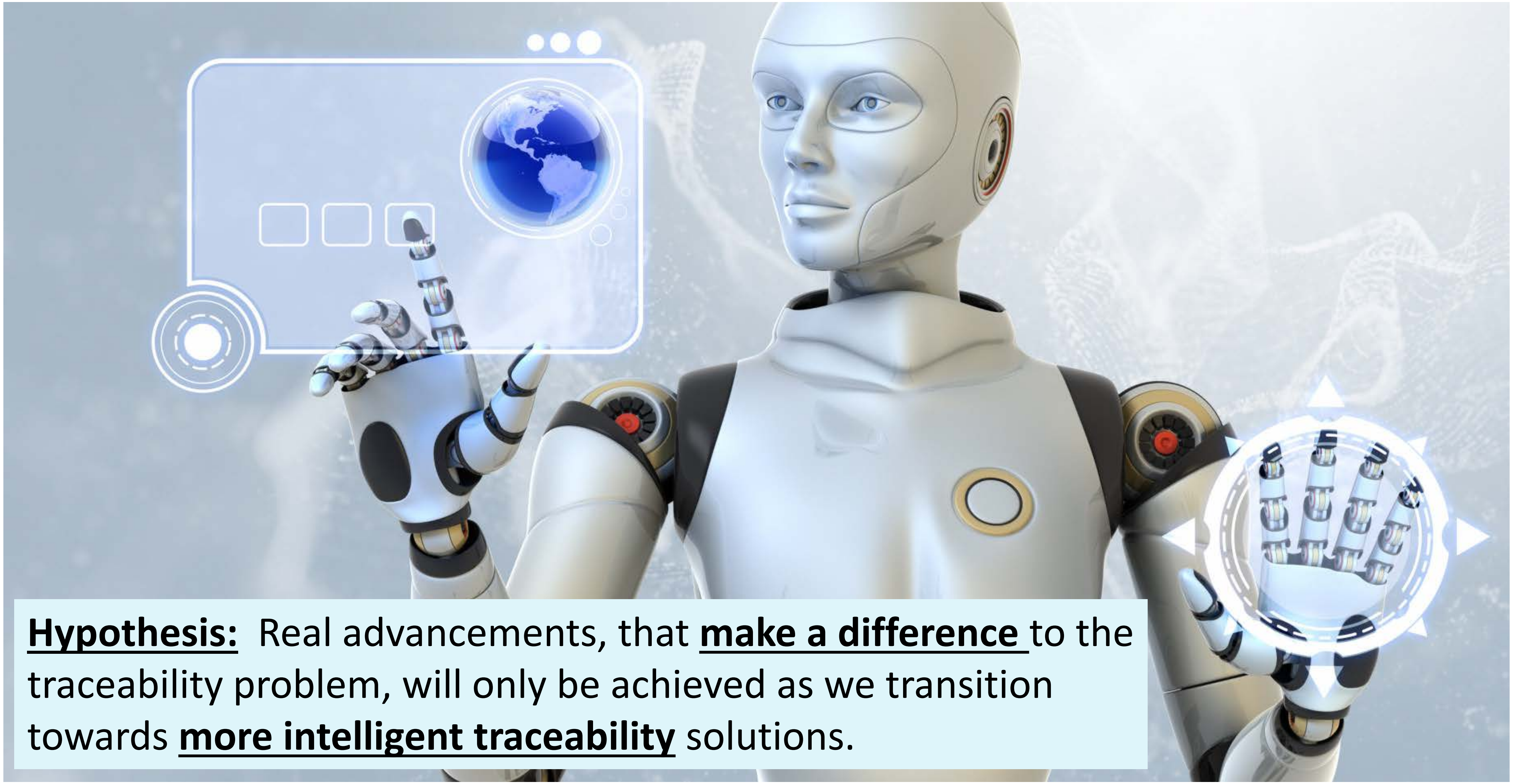
Semantic Solutions



Evolving and Managing Links

Traceability for Safety Critical…

(1) Solution 1: Semantic Traceability



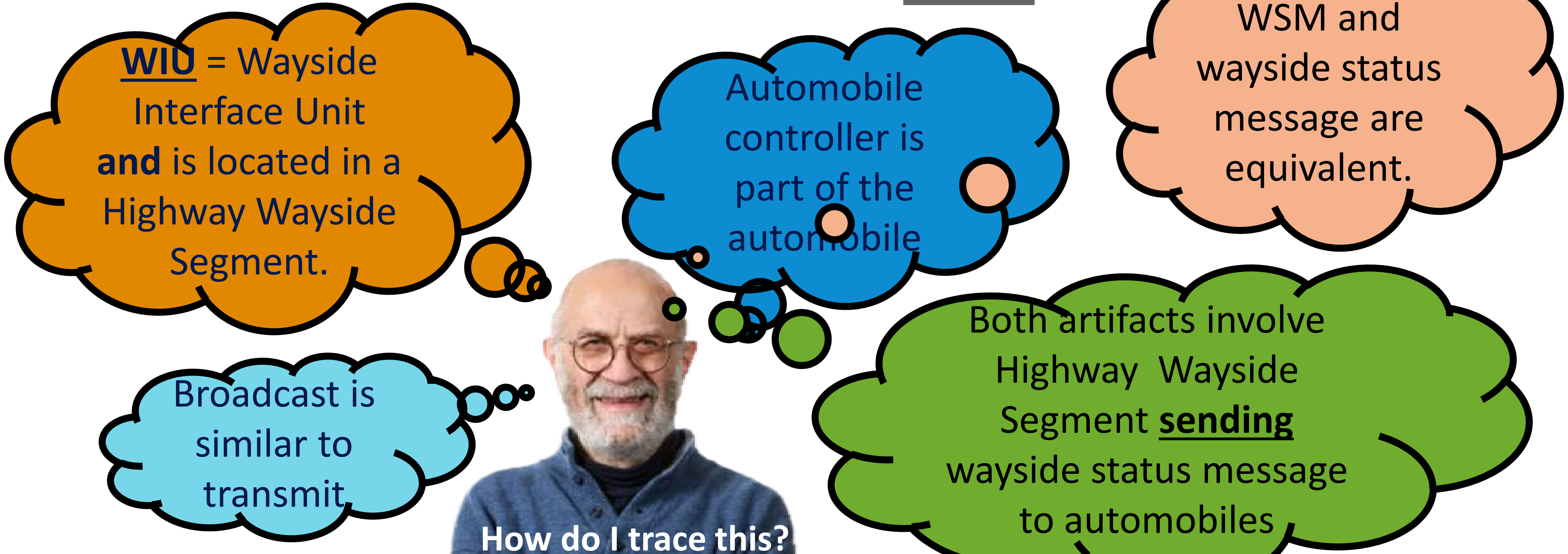
<u>Hypothesis:</u> Real advancements, that **<u>make a difference</u>** to the traceability problem, will only be achieved as we transition towards **<u>more intelligent traceability</u>** solutions.

Why a semantic approach?

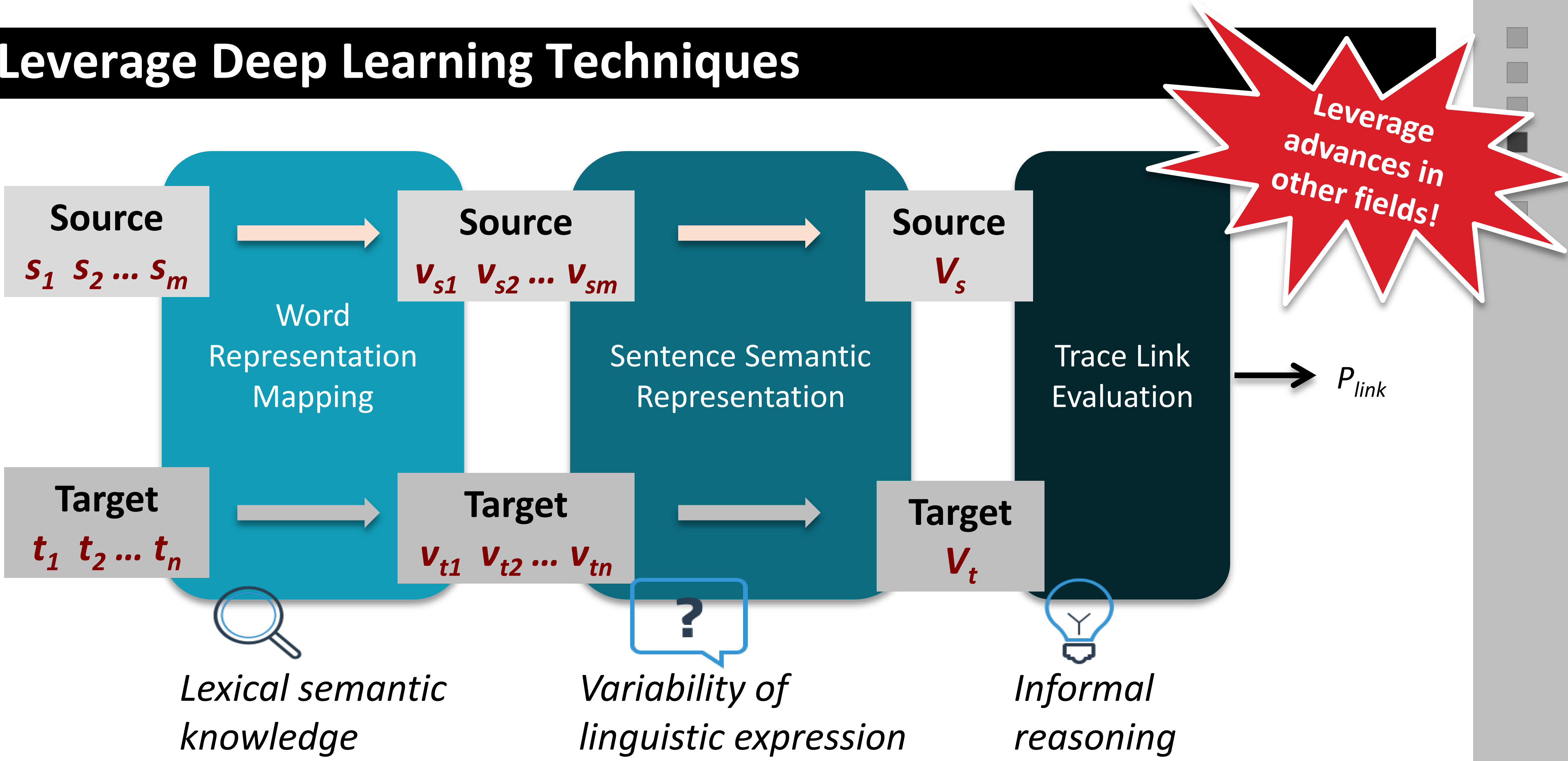
What goes through a domain expert's mind as he or she performs the tracing task?

Status of field elements is consumed by the WIU, which in turn creates a wayside status message and broadcasts that message out to any automobile within range.

The Highway Wayside Segment shall transmit information to the automobile controller in the form of WSMs.

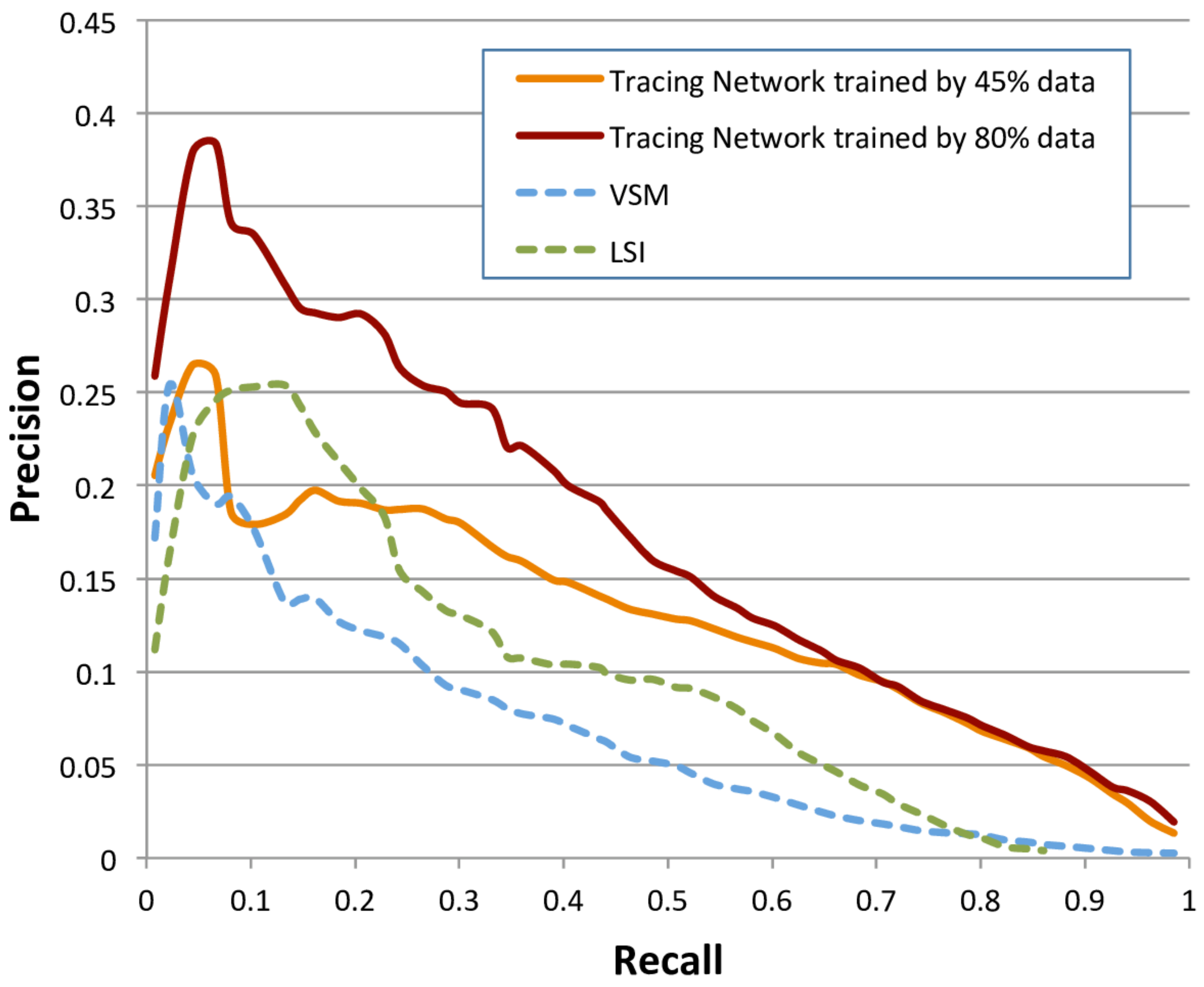


Leverage Deep Learning Techniques



Semantically enhanced Software Traceability using Deep Learning techniques.
Jin Guo, Jinghui Cheng, Jane Cleland-Huang: ICSE 2017: 3-14

A Proof of Concept

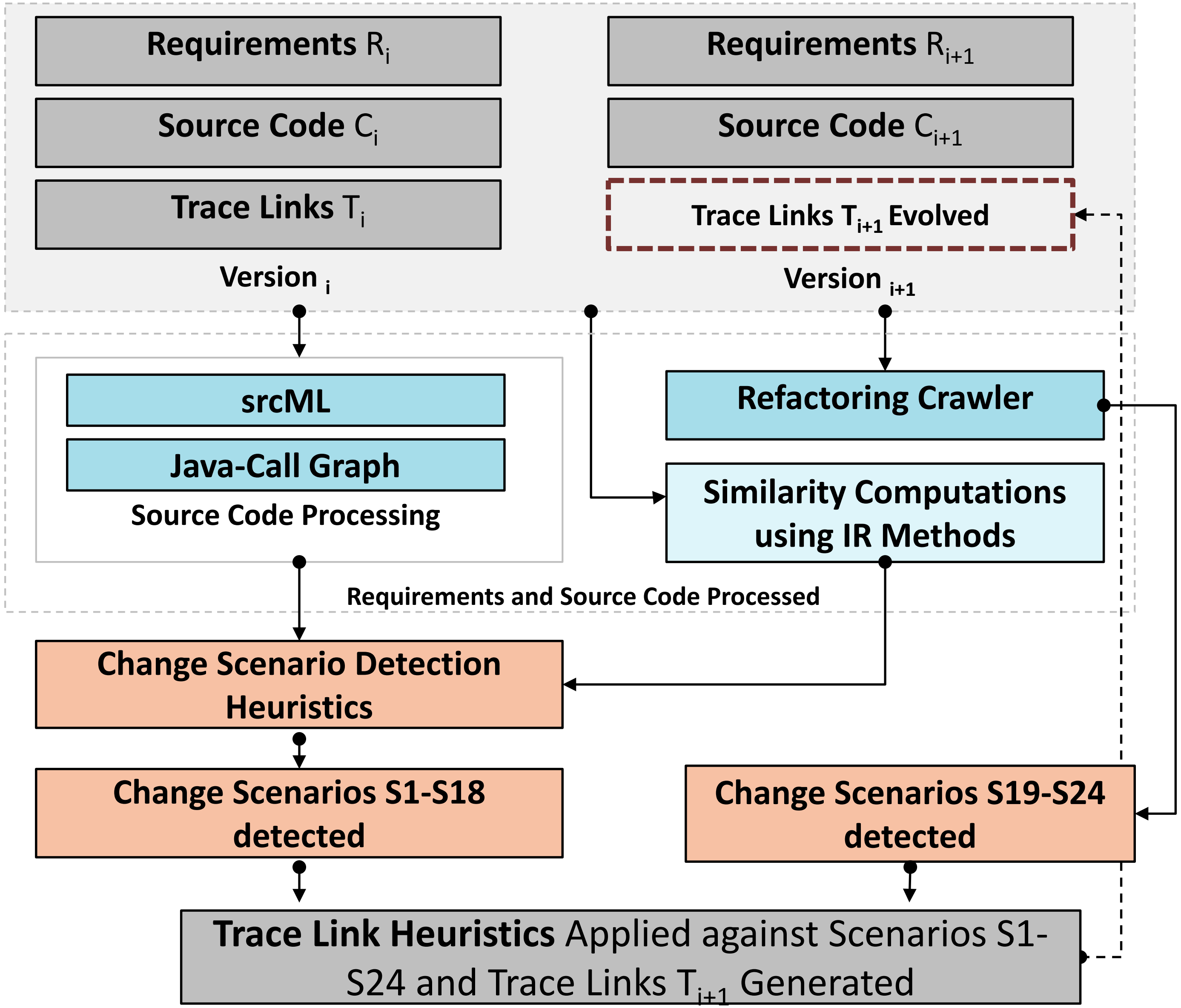


Requirements to Design trace links for Positive Train Control System

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

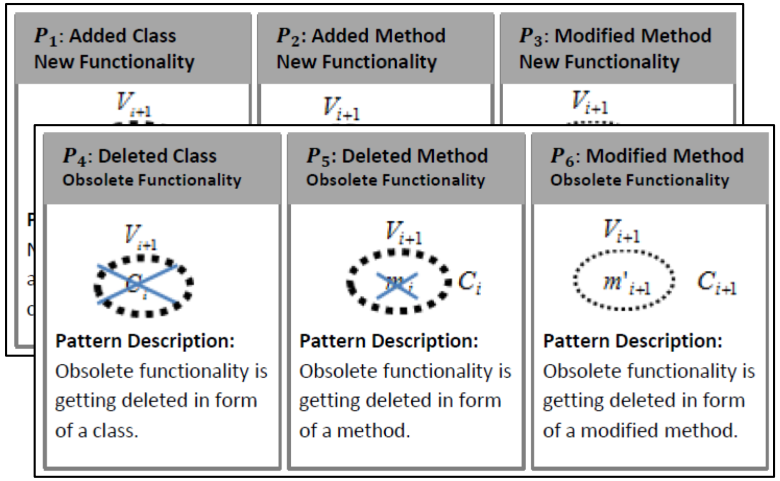


❶ Software artifacts changed across versions

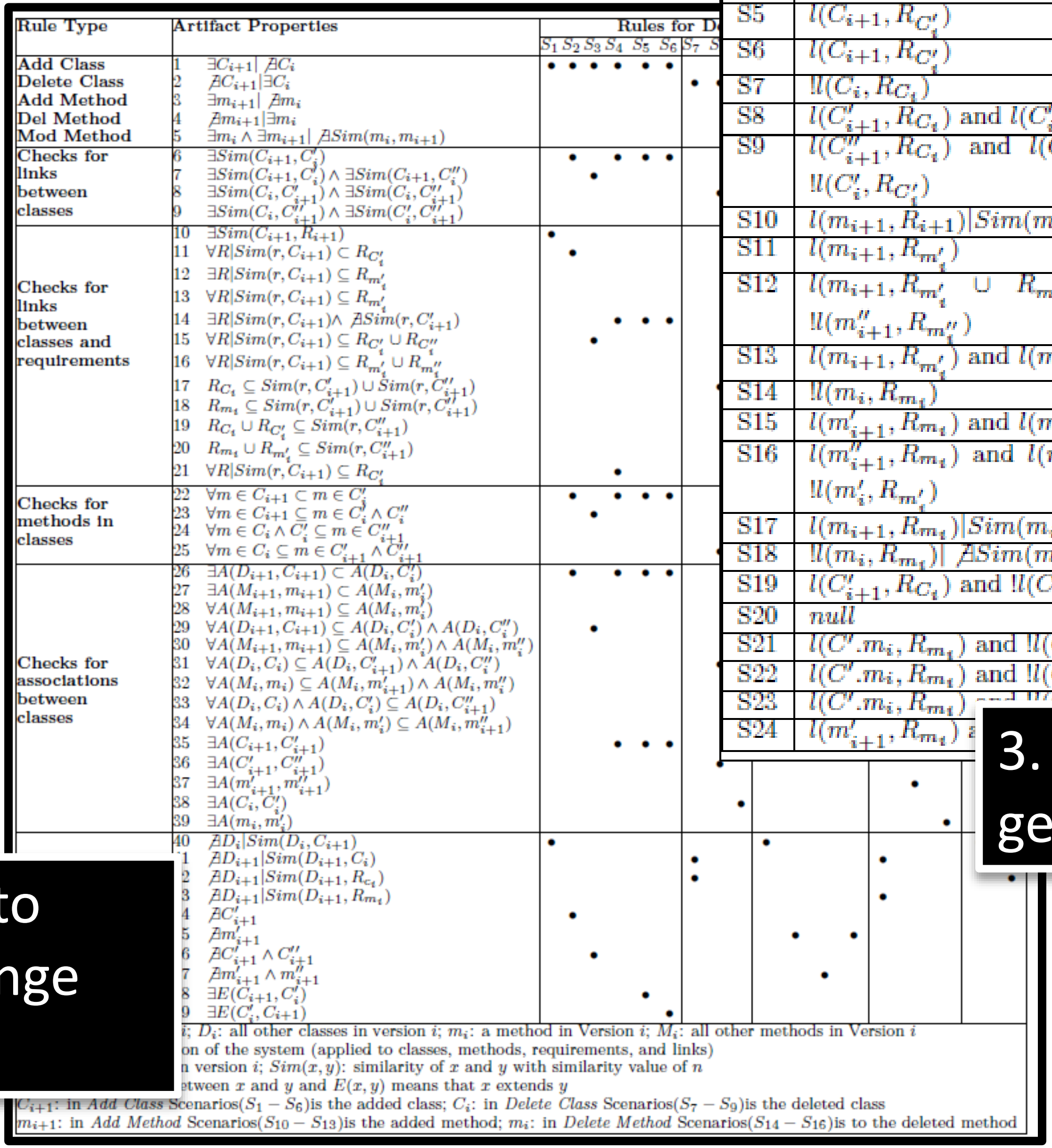
❷ Tools for detecting changes in code and requirements.

❸ 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.

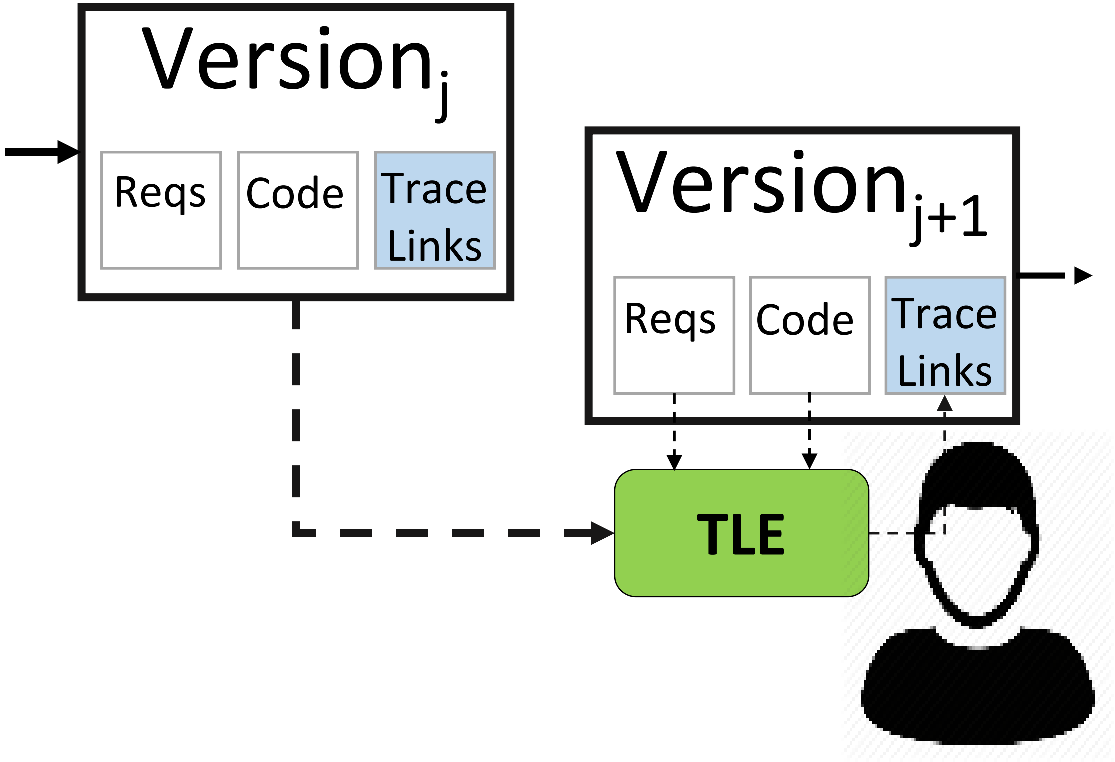


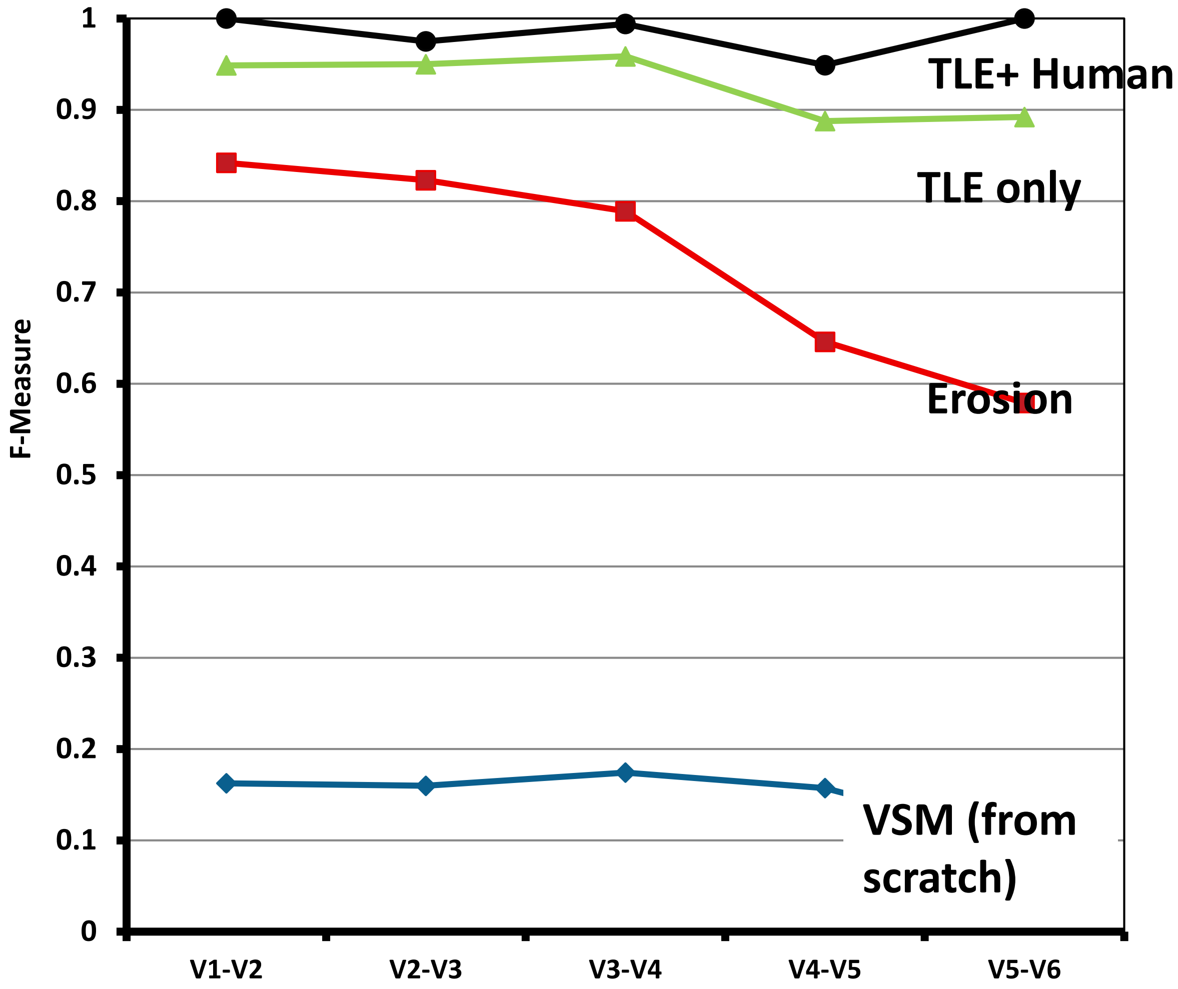
2. Define properties to detect when the change has occurred.

	Link Evolution Actions
S1	$l(C_{i+1}, R_{i+1}) \| Sim(C_{i+1}, R_{i+1})$
S2	$l(C_{i+1}, R_{C'_i})$
S3	$l(C_{i+1}, R_{C'_i} \cup R_{C''_i})$ and $!l(C'_i, R_{C'_i})$ and $!l(C''_i, R_{C'_i})$
S4	$l(C_{i+1}, R_{m_i})$ and $!l(C'_{i+1}, R_{m_i})$
S5	$l(C_{i+1}, R_{C'_i})$
S6	$l(C_{i+1}, R_{C'_i})$
S7	$!l(C_i, R_{C_i})$
S8	$l(C'_{i+1}, R_{C_i})$ and $l(C''_{i+1}, R_{C_i})$ and $!l(C_i, R_{C_i})$
S9	$l(C''_{i+1}, R_{C_i})$ and $l(C''_{i+1}, R_{C'_i})$ and $!l(C_i, R_{C_i})$ and $!l(C'_i, R_{C'_i})$
S10	$l(m_{i+1}, R_{i+1}) \| Sim(m_{i+1}, R_{i+1})$
S11	$l(m_{i+1}, R_{m'_i})$
S12	$l(m_{i+1}, R_{m'_i} \cup R_{m''_i})$ and $!l(m'_{i+1}, R_{m'_i})$ and $!l(m''_{i+1}, R_{m''_i})$
S13	$l(m_{i+1}, R_{m'_i})$ and $l(m''_{i+1}, R_{m'_i})$ and $!l(m'_i, R_{m'_i})$
S14	$!l(m_i, R_{m_i})$
S15	$l(m'_{i+1}, R_{m_i})$ and $l(m''_{i+1}, R_{m_i})$ and $!l(m_i, R_{m_i})$
S16	$l(m''_{i+1}, R_{m_i})$ and $l(m''_{i+1}, R_{m'_i})$ and $!l(m_i, R_{m_i})$ and $!l(m'_i, R_{m'_i})$
S17	$l(m_{i+1}, R_{m_i}) \| Sim(m_{i+1}, R_{m_i})$
S18	$!l(m_i, R_{m_i}) \| \nexists Sim(m_{i+1}, R_{m_i})$
S19	$l(C'_{i+1}, R_{C_i})$ and $!l(C_{i+1}, R_{C_i})$
S20	$null$
S21	$l(C'.m_i, R_{m_i})$ and $!l(C.m_i, R_{m_i})$
S22	$l(C'.m_i, R_{m_i})$ and $!l(C.m_i, R_{m_i})$
S23	$l(C'.m_i, R_{m_i})$ [illegible]
S24	$l(m'_{i+1}, R_{m_i})$ [illegible]

3. Define trace link generation heuristics

Integrate the user in the loop

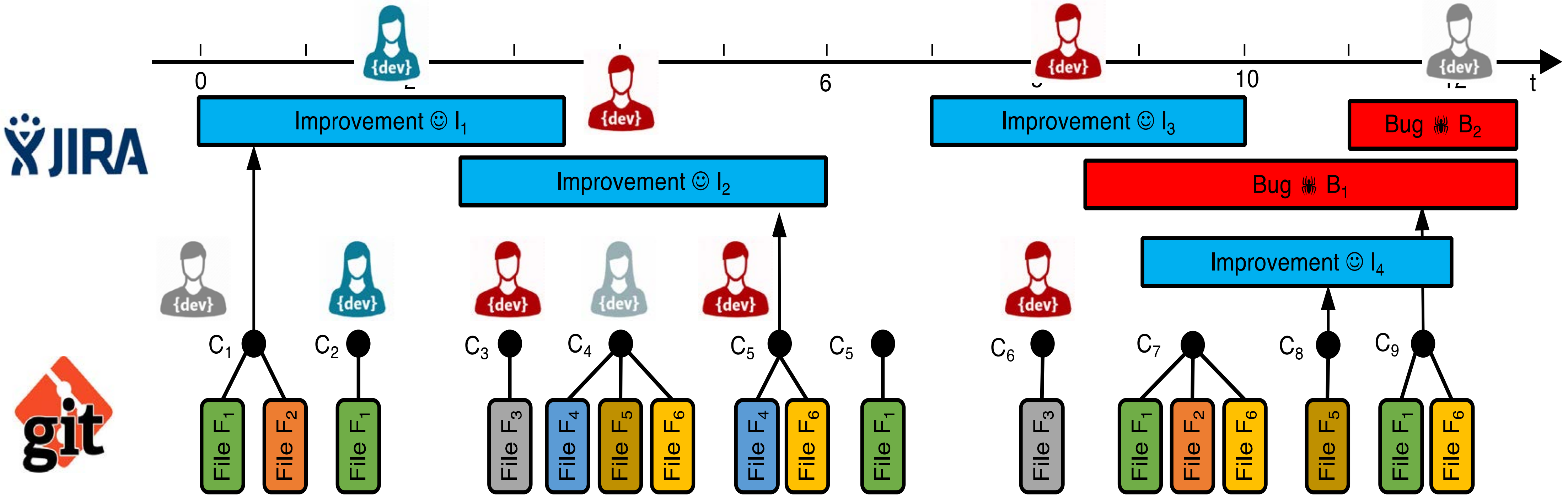




In our experiments the effort required by humans to confirm or deny TLE links was minimal – with few decision points per day.

Evolving software trace links between requirements and source code. Mona Rahimi, Jane Cleland-Huang: Empirical Software Engineering 23(4): 2198-2231 (2018)

Leverage the Project Environment



Mining project repositories

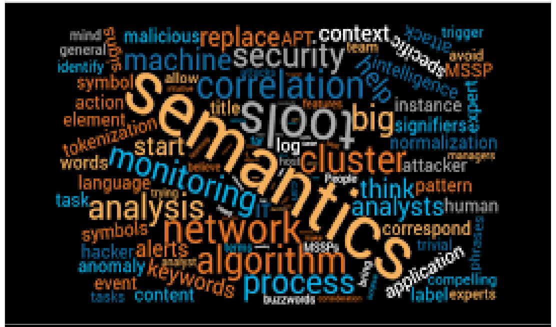
Temporal properties

Machine learning

Project exhaust!

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

New Trajectories bring new challenges



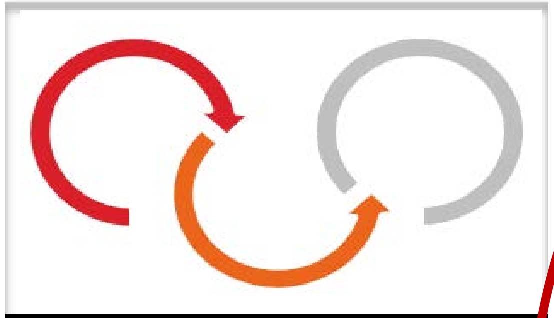
Semantic Solutions

Addresses real problem ✓

Leverages Goldilocks ✓

Well defined benchmarks ✓

Adequate data sets X



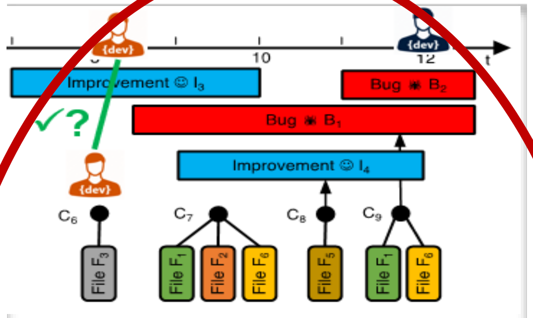
Evolving Links

Addresses real problem ✓

Leverages Goldilocks ✓

Well defined benchmarks ✓

Adequate data sets X



Mine Links

Addresses real problem ✓

Leverages Goldilocks ✓

Well defined benchmarks ✓

Adequate data sets ✓

Safety Critical Trace..

Addresses real problem ✓

Leverages Goldilocks ✓

Well defined benchmarks X

Adequate data sets X

Obvious solution is to build industry collaborations…

Dronology Project @ Notre Dame

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

Dronology Project @ Notre Dame

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

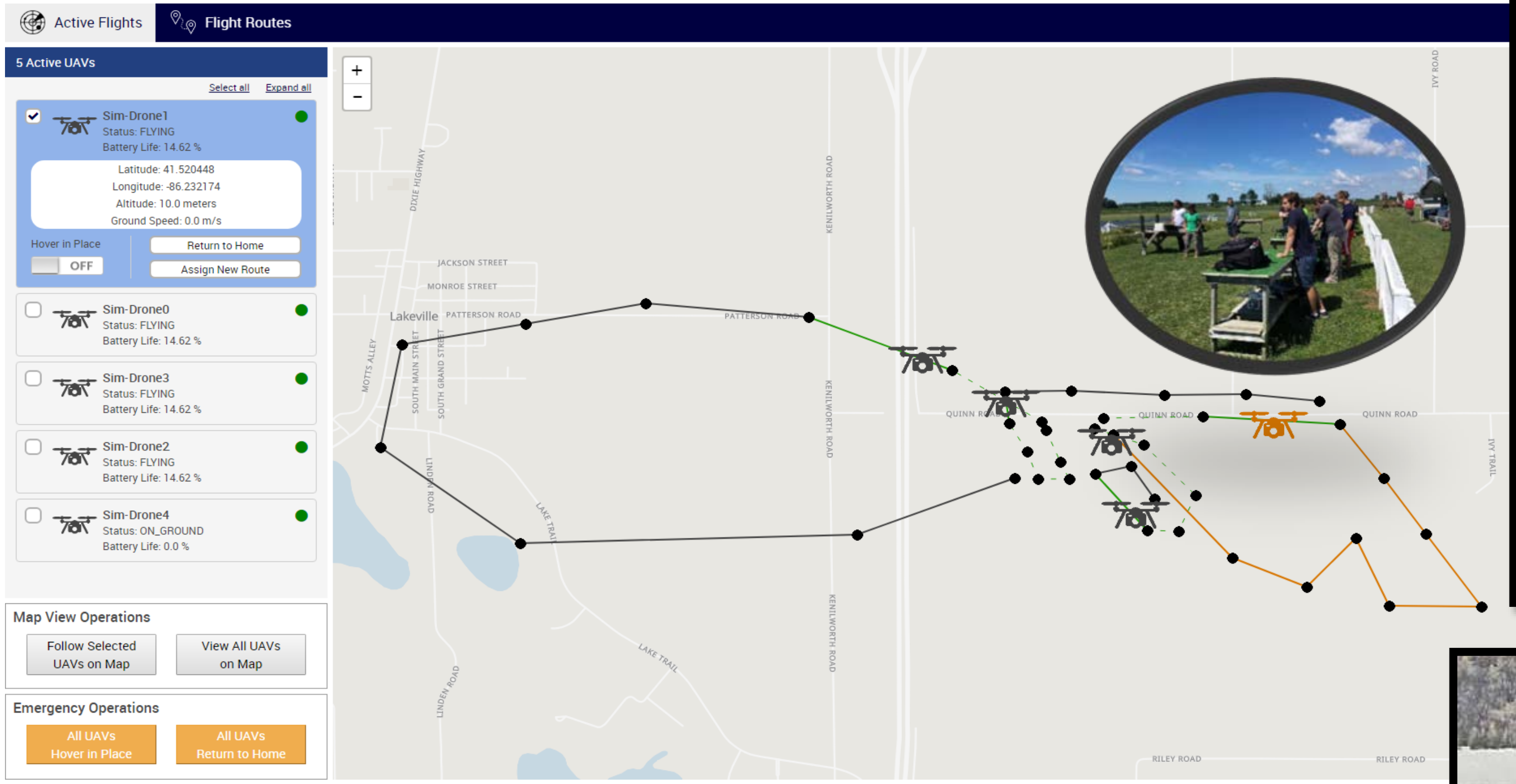
Safety Critical,
Cyber Physical
System.

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.

Dronology Project @ Notre Dame

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



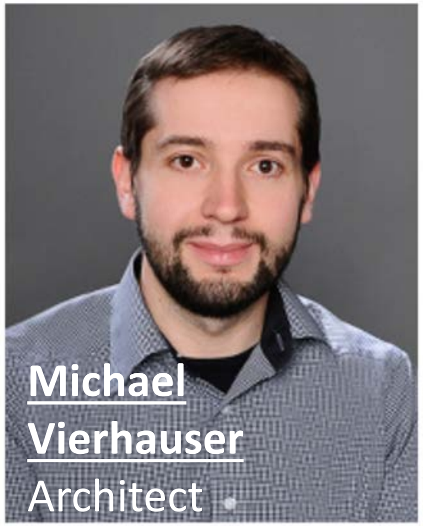
River rescue

Highway Chemical Spill

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…

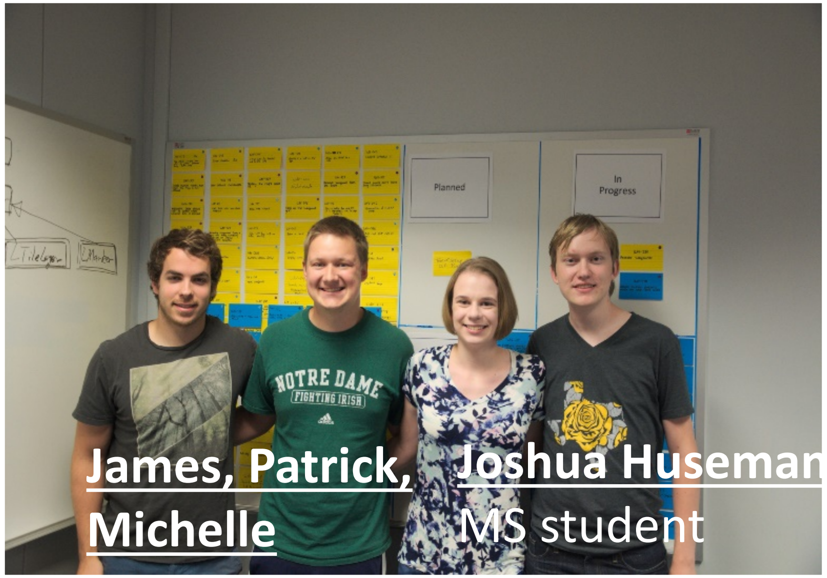
Team Work



Michael Vierhauser
Architect

Jane Wyngaard
Hardware

Jinghui Cheng
Post Doc / UI



James, Patrick, Michelle

Joshua Huseman
MS student

Sean Bayley
PhD Student

Alex Madey
Undergraduate

Quinlan McMillan
Undergraduate

Greg Madey
Frank

Dronology Stakeholders

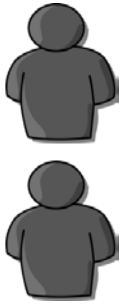
http:/Dronology.info




You?

Requirements Engineering researchers interested in using Dronology to support research into traceability, forensic requirements, goal modeling, runtime adaptation….

End users interested in deploying Dronology to support specific scenarios of use.



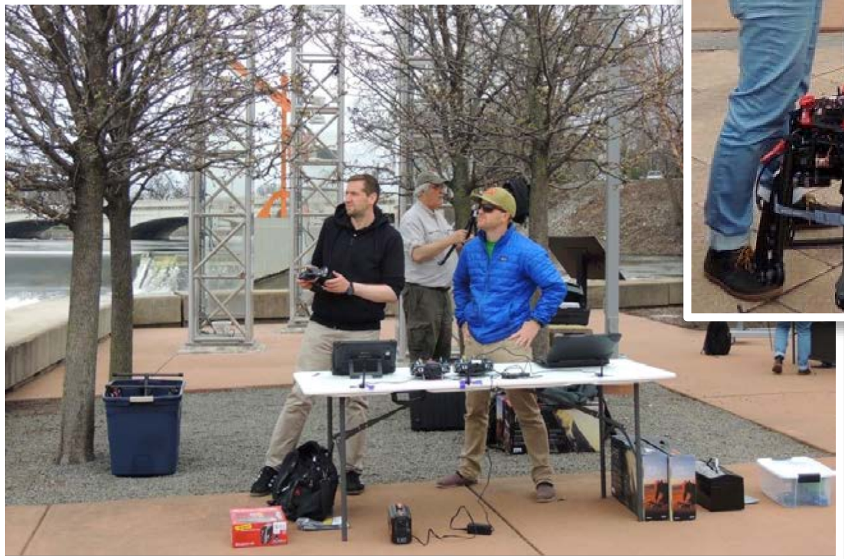
Students

General public

Funders

Stakeholders' needs ultimately drive feature prioritization and variability points.

River Rescue Demo with Dronology

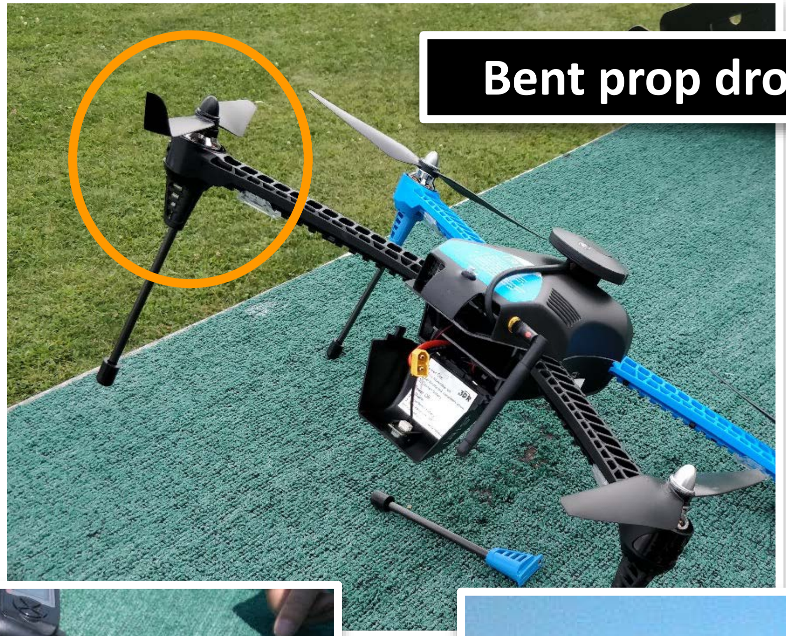


Testing an AED drop

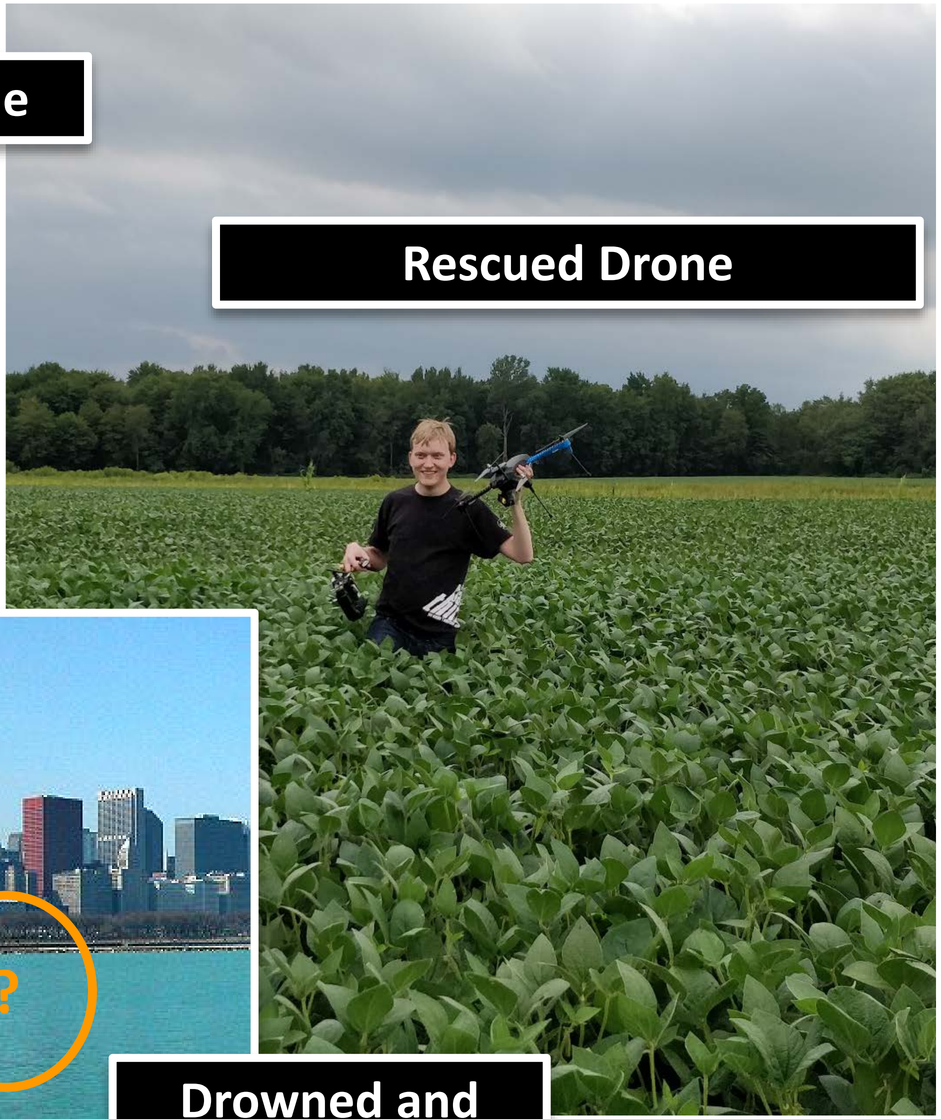
Testing an AED drop

Dronology: Our Crashes

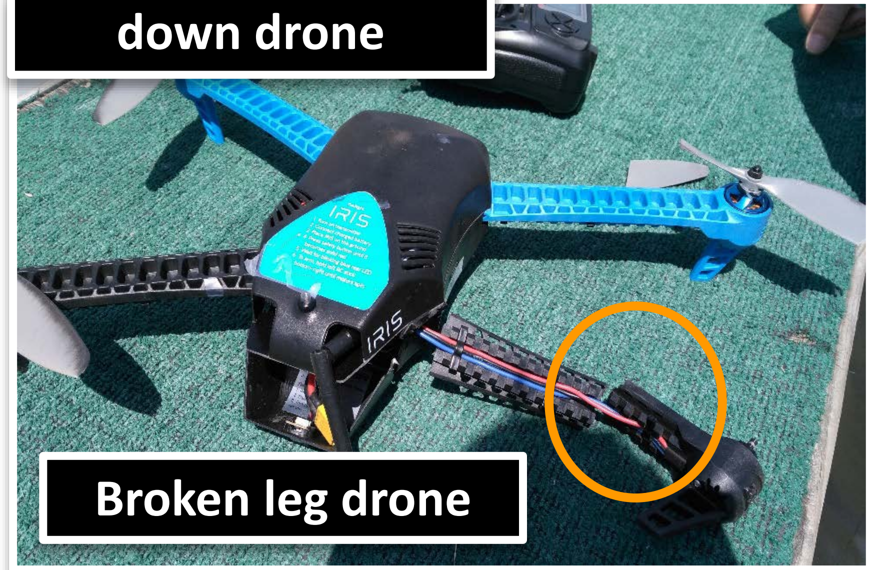
Trajectory challenged, upside down drone



Bent prop drone



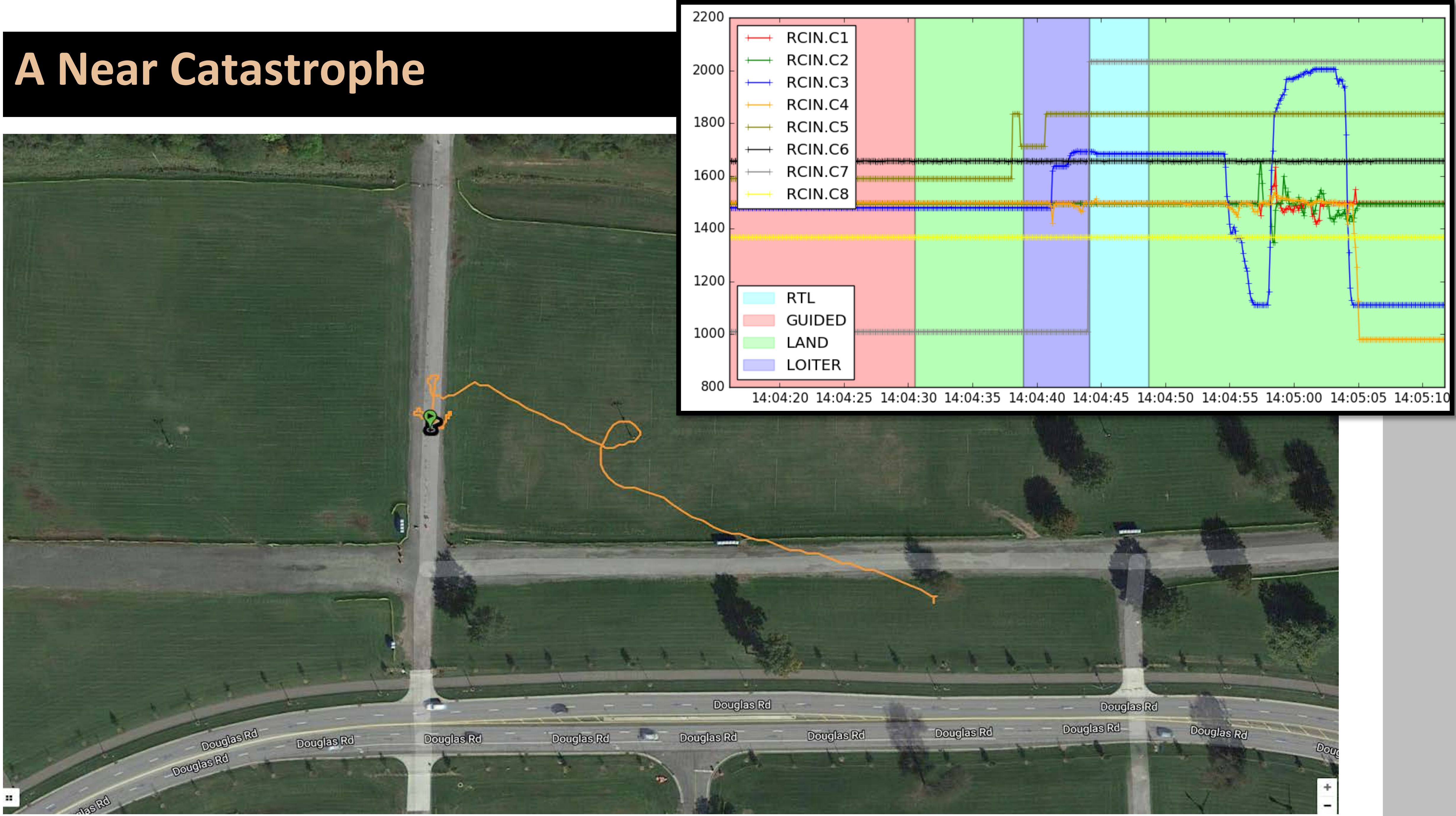
Rescued Drone



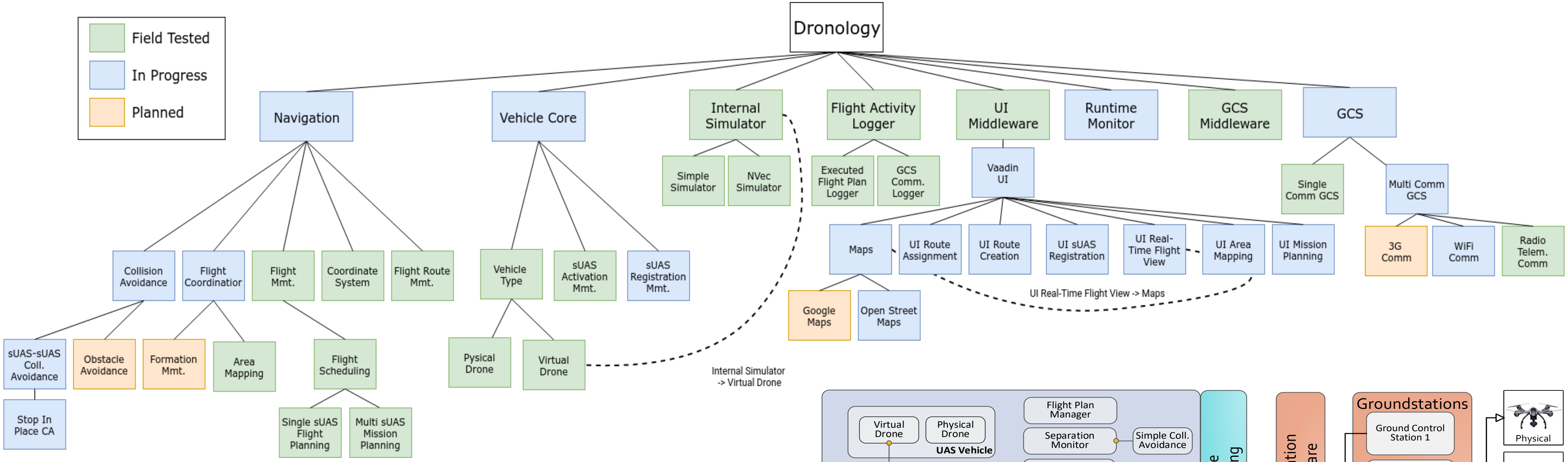
Broken leg drone

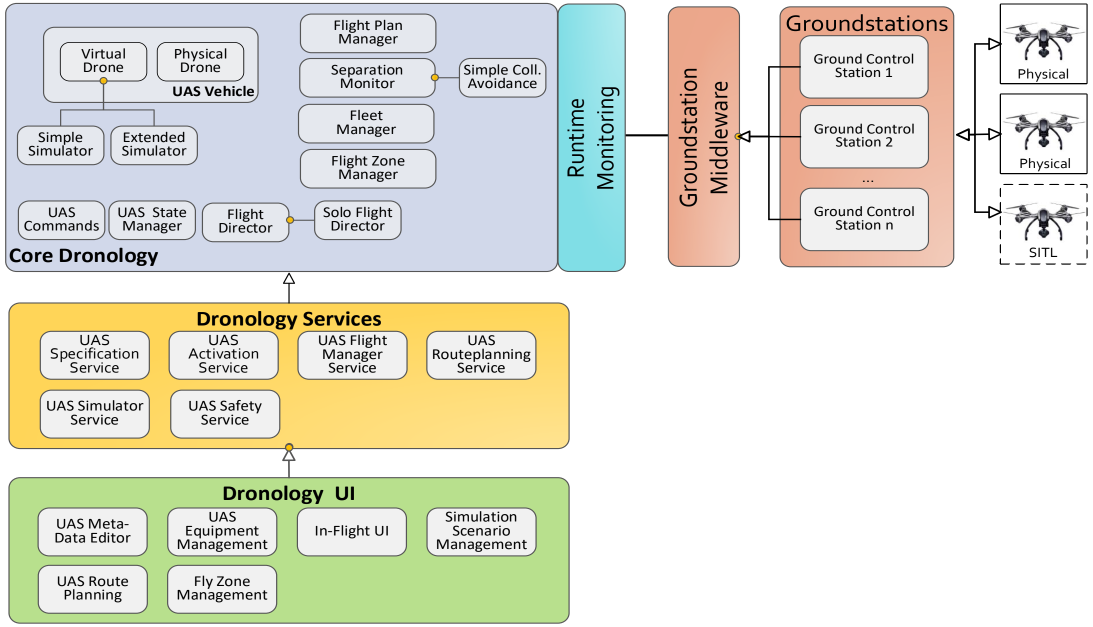
Drowned and missing drone

A Near Catastrophe



What do we gain?

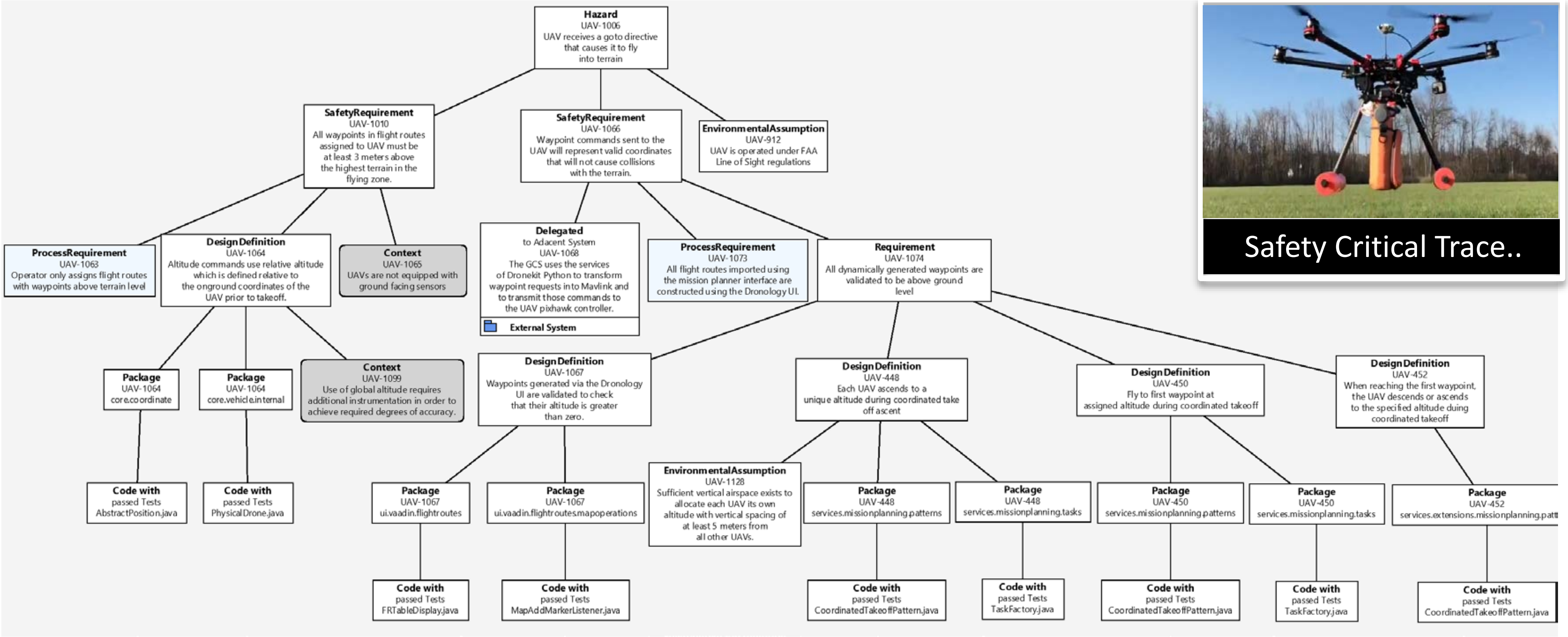




An entirely new controllable environment for experimenting with Software Traceability across multiple artifact types and multiple versions.

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

An Ecosystem of Traceability Artifacts



Safety Critical Trace..

Artifact tree generated automatically from Jira and Github showing traceability from hazard to code.

Immersive Requirements Engineering

Discovering, Analyzing, and Managing Safety Stories in Agile Projects

Jane Cleland-Huang and Michael Vierhauser
Department of Computer Science and Engineering
University of Notre Dame
Indiana, USA
JaneClelandHuang@nd.edu, mvierhau@nd.edu

***Abstract*—Traditionally, safety-critical projects have been developed using the waterfall process. However, this makes it costly and challenging to incrementally introduce new features and to certify the modified product for use. As a result, there has been increasing interest in adopting agile development paradigms within the safety-critical domain. This in turn introduces numerous challenges. In this paper we address the specific problems of discovering, analyzing, specifying, and managing safety requirements within the agile Scrum process. We propose *SafetyScrum*, a methodology that augments the Scrum lifecycle with incrementally applied safety-related activities and introduces the notion of "safety debt" for incrementally tracking the current safety status of a project. We demonstrate the viability of *SafetyScrum* for managing safety stories in an agile development environment by applying it to a project in which our existing Unmanned Aerial Vehicle system is enhanced to support a River-Rescue scenario.**

I. Introduction

Systems operating in safety-critical domains, where failures can cause harm or injury, must not only deliver prescribed functionality, but must do so in a way that ensures that the system is safe and secure for its intended use [28]. To this end, safety-critical systems must meet stringent guidelines in order to receive approval or certification [53, 19, 7, 14]. The strict requirements of the certification process as well as constraints introduced by the rigid timelines imposed by hardware components have led many organizations to follow a traditional waterfall approach – often resulting in the phenomenon referred to as the *big freeze* [42]. The significant cost and effort of changing and then recertifying a product makes it difficult to introduce change, thus hampering the ability to provide new features or to respond to customer needs.

Agile techniques have traditionally been deemed unsuited to safety-critical development [9]; however, recently the idea of leveraging agility has gained considerable traction. For example, the European Open-DO initiative [42] is exploring techniques for integrating agility into the safety-critical development process, and there are numerous accounts reporting its experimental and effective deployment [34, 31]. Doss and Kelly [18] reported results from a recent practitioner survey with a total of 31 participants, 87% of whom had experience in safety-critical systems development, and 77% with practical experience using a broad range of agile methods. Their survey produced several insights of particular interest to the requirements process. Respondents strongly supported the notion that eliciting safety requirements, performing hazard analysis, and developing safety assurance cases must be performed iteratively, with 50% reporting that safety problems were not always identified early in the lifecycle during the upfront hazard analysis. In other words, they acknowledged the need for a more incremental development process.

On the other hand, applying agile processes in safety-critical projects introduces multiple challenges – each of which must be carefully explored in order to develop appropriate solutions and practices. In this paper we provide concrete examples derived from our experiences in using an agile process to develop the Dronology system for controlling Unmanned Aerial Vehicles (UAVs) and we describe the agile safety process we adopted as a result of those experiences. The characteristics of our project, including its initially unknown requirements, a steep technical learning curve associated with entering a new domain, and team members' prior experience with agile methods, indicated a good fit for applying an agile, Scrum-based approach [60]. However, in early phases of our project, we realized that the project's safety concerns were non-trivial and could not be adequately addressed without carefully augmenting the Scrum process.

Initially, in early phases of our project, we addressed safety issues through conducting a series of brainstorming sessions in which we identified hazards and their contributing faults, and then proposed safety requirements and design constraints that would prevent or mitigate the occurrence of the hazard. However, we found that identifying *all* hazards and their contributing failures at the start of the project was particularly challenging given the emergent nature of the UAV domain, including its novel end-user applications and rapidly changing technologies. Many new hazards and faults were discovered incrementally as we conducted field tests with the UAV hardware, met with stakeholders to explore their emerging requirements, and brainstormed design solutions. Our early observations aligned closely with those made by participants in Doss' survey [18] and highlighted the potentially competing goals of agile processes versus safety-critical development.

Agile development practices are founded upon four core principles laid out in the agile manifesto [6]. Two of these principles of "responding to change over following a plan" and delivering "working software over comprehensive documentation" are particularly challenging to achieve in safety-critical

What is a Research Incubator?

An incubator enables researchers to experiment with a theory or hypothesis in a controlled environment, and to progressively develop the idea until it is ready for testing and deployment in a fully industrial environment.

- Software and Systems Requirements
- Safety Assurance
- Product Lines
- Software & Systems traceability
- Runtime monitoring
- Human studies

The Eyrie Research Incubator in collaboration with Robyn Lutz

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 an open research challenge. 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
- Development Environ.

SafeWalk

- Software Artifacts
- Physical Elements
- Runtime Environment
- Development Environ.

Eyrie Website

Advisory Board

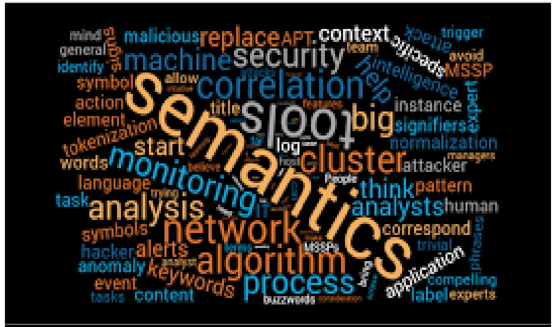
Future Incubator Projects

Challenges

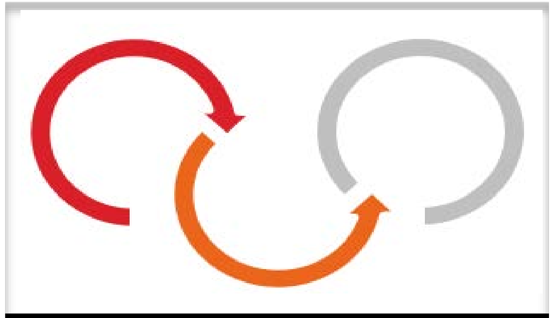
Static Artifact Challenges
- Automatically evolve trace links across safety-related software artifacts.
- Formally specify requirements for CPS User Interface.
- Model safety and check safety properties associated with a UI.

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

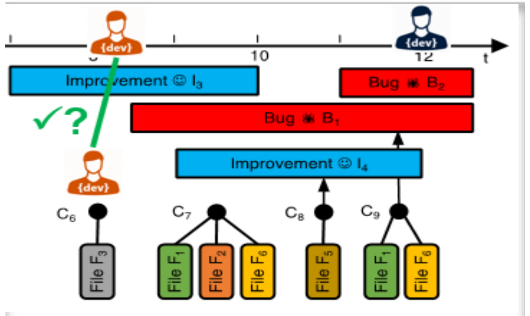
New Trajectories bring new challenges



Semantic Solutions



Evolving Links



Mine Links

Safety Critical Trace..

Semantic Solutions	Evolving Links	Mine Links	Safety Critical Trace..
Addresses real problem ✓	Addresses real problem ✓	Addresses real problem ✓	Addresses real problem ✓
Leverages Goldilocks ✓	Leverages Goldilocks ✓	Leverages Goldilocks ✓	Leverages Goldilocks ✓
Well defined benchmarks ✓	Well defined benchmarks ✓	Well defined benchmarks ✓	Well defined benchmarks X
Adequate data sets X	Adequate data sets X	Adequate data sets ✓	Adequate data sets X

This is a starting point. It is a long "game"…
We *need* strong collaborations between industry and academic research.

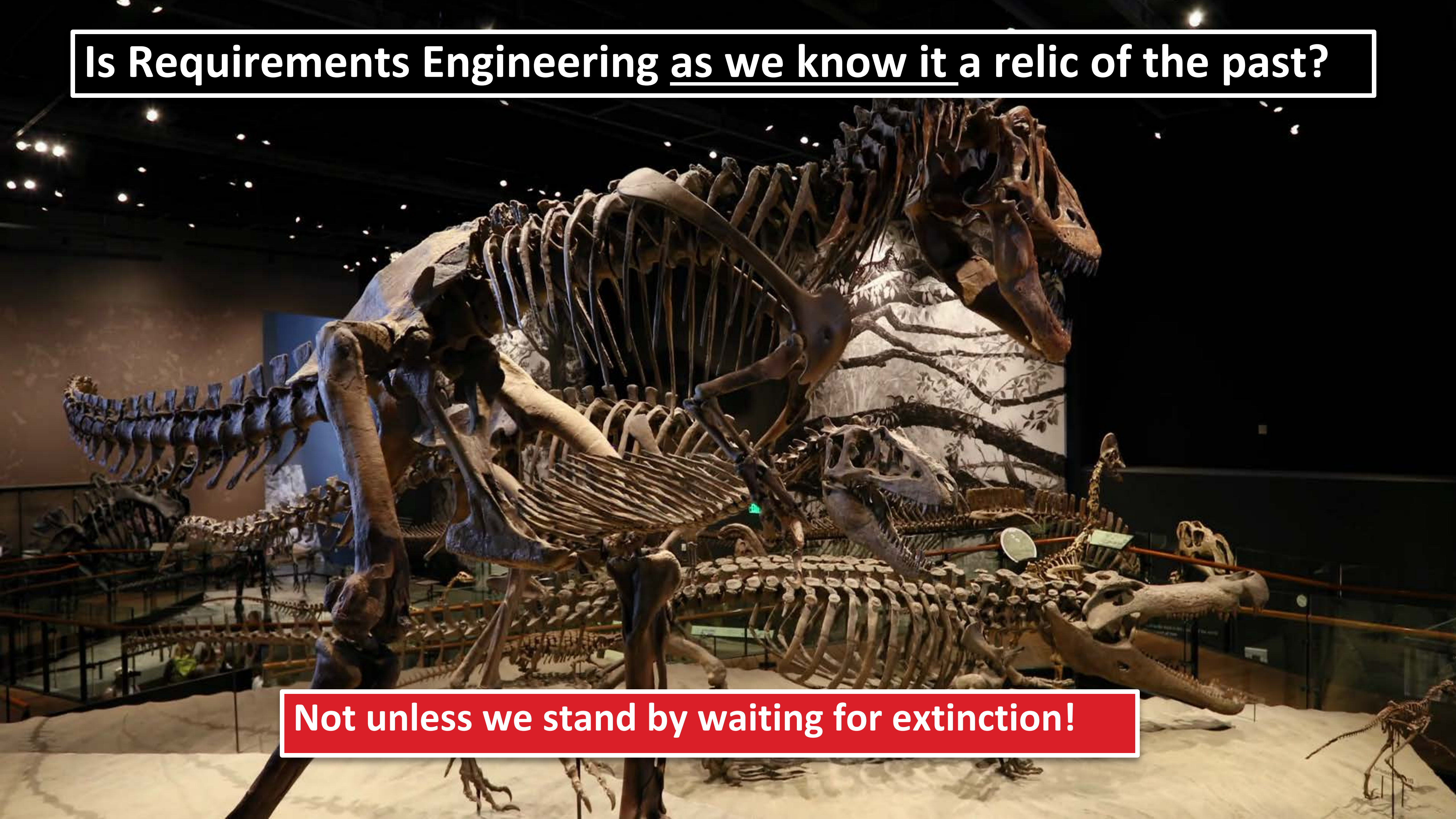
As a community of practitioners and researchers:

Individually:

- Be visionary
- Be courageous -- to ask the hard questions.
- Constantly evaluate progress.
- Tackle important questions with potential for real impact.

As a community:

- Set vision!
- Be open-minded to innovative work
- Nurture collaborations.
- Tackle inhibitors head on.
- Maintain open communication channels between industry and academia.


Is Requirements Engineering as we know it a relic of the past?
Not unless we stand by waiting for extinction!

Requirements Engineering Research in a Changing World

International Requirements Engineering Conference
Keynote: August 19th, 2018
Banff, Canada

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