Effective Software & Systems Traceability

Distinguished Lecture
University of Luxembourg
Friday, March 20th

Professor Jane Cleland-Huang
Center of Excellence for Software Traceability
DePaul University

DePaul University

CoEST
Center of Excellence for Software Traceability

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

What this talk is about…

Traceability
– its challenges, successes, and open issues…

Tackling
a hard, multi-faceted research problem.

Teaming
with industrial partners so that we address the right problem and move towards transition to practice.

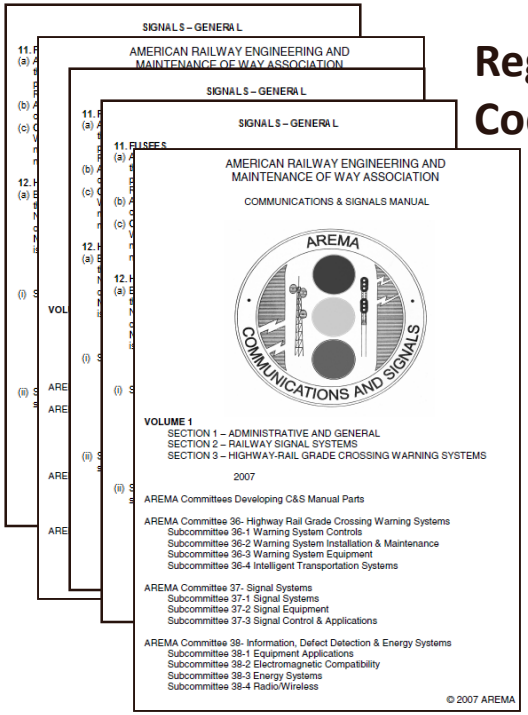
Software traceability is…

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 Compliance
- Change Impact Analysis
- Regression Test selection
- Safety Analysis
- Design Rationales
- Coverage Analysis

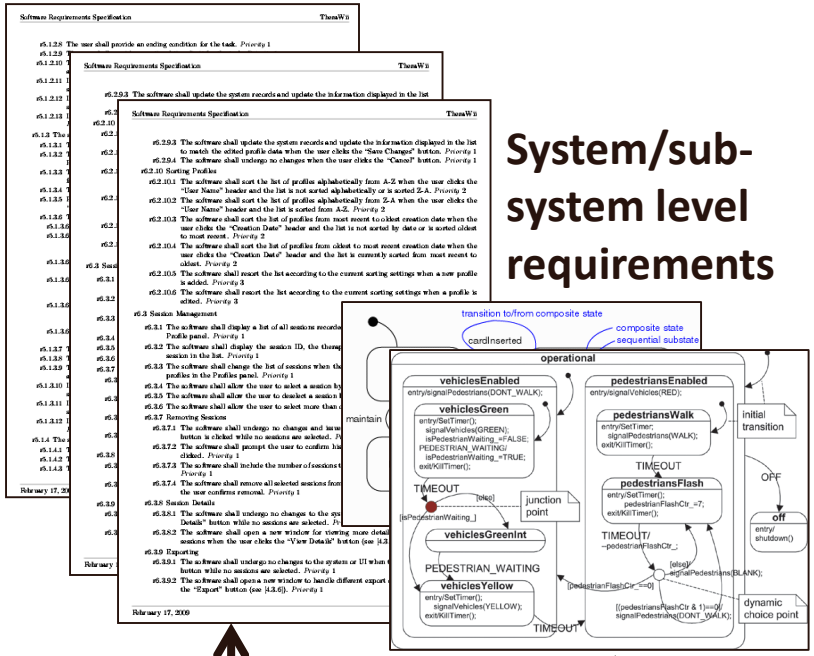
Traceability is of particular concern in safety & mission-critical systems.

Achieve Regulatory Compliance



Regulatory Codes

Satisfies relevant codes



System/sub-system level requirements

Validates

Realizes

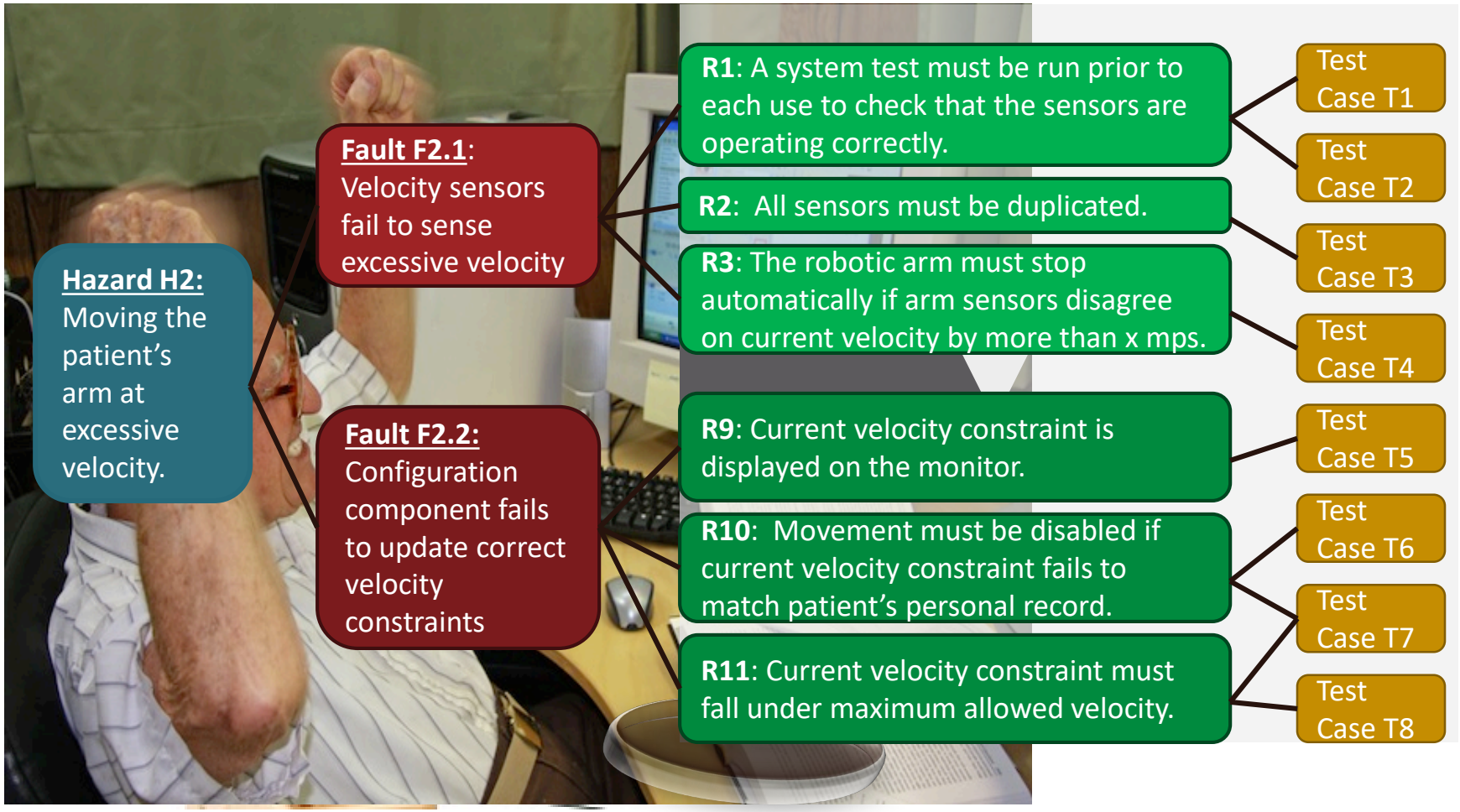
Code

Tests

Test cases (unit, integration, acceptance)

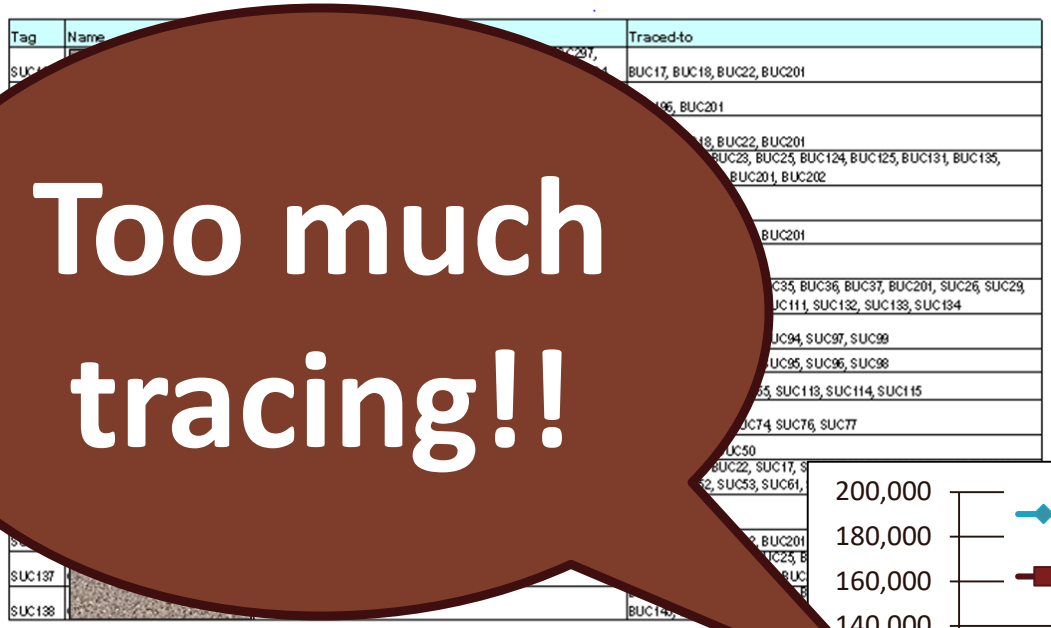
Use Traceability to demonstrate compliance to regulatory codes.

Creating Trace Links



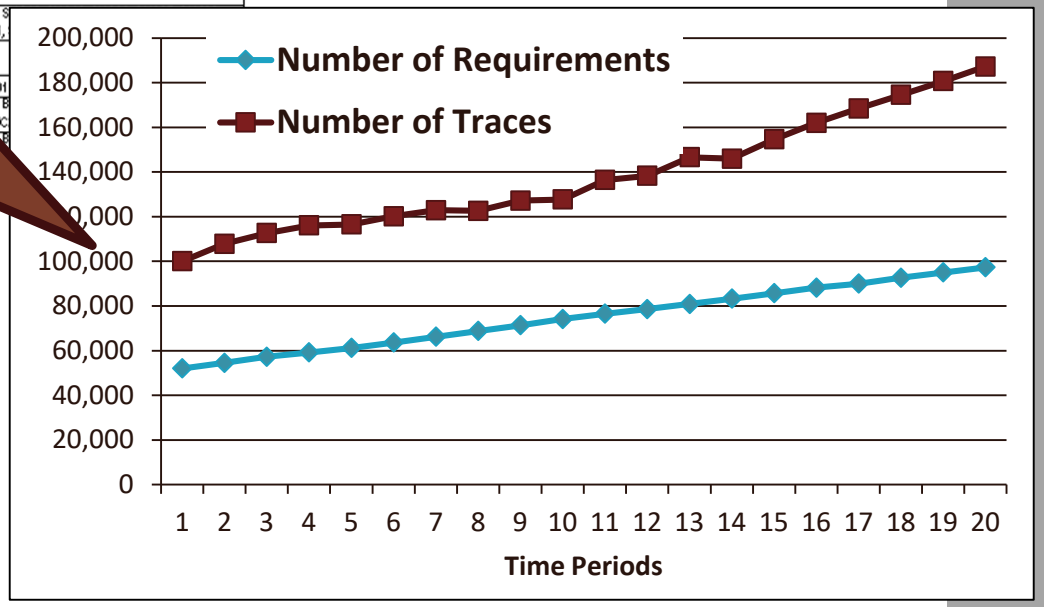
Current practice is primarily manual in nature

…but



Too much tracing!!

The number of requirements and other artifacts grow quickly as the project progresses.



Excessive numbers of traceability links deteriorate into an unwieldy, inaccurate, tangle of relationships which are extremely hard to maintain in an evolving software system.

Available Industrial Tools e.g. DOORS

Formal module '/Passenger Car Development/Requirements/User Requirements' current 2.1 (1998) - ...

File Edit View Insert Link Analysis Table Tools User Help

Standard view | All levels

User requirements for passenger car

3.1.3.1 Speed & Acceleration

3.1.3.1.1 Backwards

The car shall be able to move backwards to a maximum speed of

3.1.3.1.2 Forwards

Users shall be able to travel at speeds up to 200 kilometers per ho

Users shall be able to accelerate from 0 to 100 Kilometers per hou

Users shall be able to travel automatically at predefined speeds.

3.1.3.2 Distance

Users shall be able to travel 2000 kilometers without the need for

Users shall be able to travel 2000 kilometers without the need for

Username: Jim Wagner | Exclusive edit mode

Formal module '/Passenger Car Development/Requirements/System Requirements' current 0.0 ...

File Edit View Insert Link Analysis Table Tools User Help

Standard view | All levels

System requirements for passenger car

2.1 Power car

2.1.1 Move car

2.1.1.1 Move forwards

The car shall be able to move forwards at all speeds from 0 to 200 kilometers per hour on standard flat roads with winds of 0 kilometers per hour, with 180 BHP.

2.1.1.2 Move backwards

The car shall be able to move backwards to a maximum speed of standard flat roads with winds of 0 kilometers per hour, with 180

2.1.2 Accelerate car

The car shall be able to accelerate from 0 to 100 Kilometers per h standard flat roads with winds of 0 kilometers per hour.

The car shall be able to accelerate from 100 to 150 kilometers per

Make Link From Start

Copy object after

Copy object below

Copy and link object after

Copy and link object below

Cancel

Link from original object to current object

Drag-and-drop to link within a document, across documents, or across projects. . . .

Slide used from DOORS Training material

Major Research Advances

The goal is the **total automation** of high-quality trace link creation and maintenance.

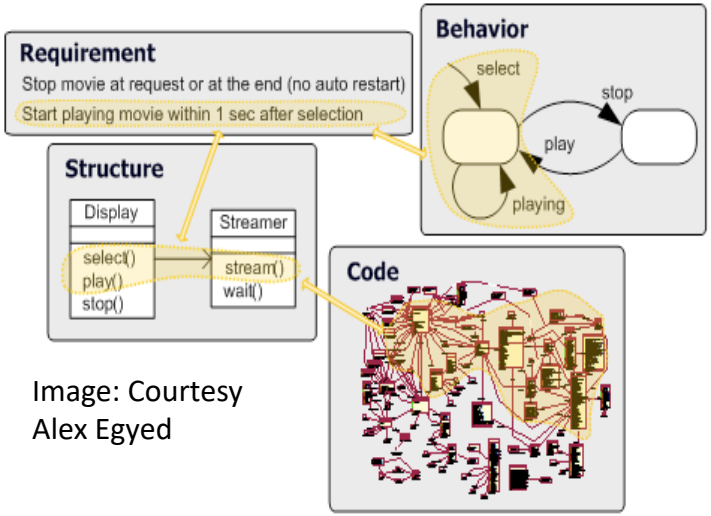


Image: Courtesy Alex Egyed

Leverage the fact that text is found across almost every model and use **information retrieval** methods to identify related artifacts.

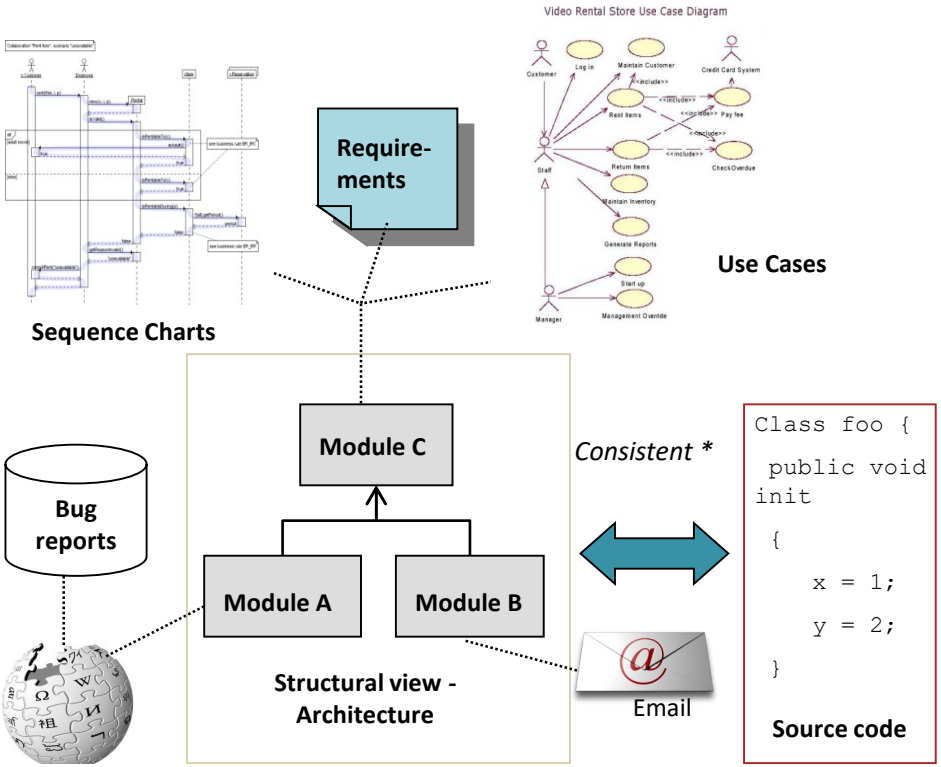


Image: Courtesy Hazel Asuncion.

Monitor the project environment and use **project exhaust** to infer trace links based on sequencing of tasks and version control check-in information.

First Generation: Trace Retrieval Solution

Poirot : *TraceMaker*

Project | Query | Artifacts | Options | Help

Standard Query > Report

Query: Document ID: 6.9.7

Joints and connections: Gasket materials shall be of either neoprene or other similar material resistant to any action of gas. Natural rubber shall not be used.

<<>> GasReqsForPaper

CANDIDATE LINKS | UNLIKELY LINKS | ID : | Find | Save

1 - 10 of 52 Next Last

Document ID:	Document Description:	Confidence Level	Accept
A201	Below Ground Pipe: Buried gas supply pipe shall have non-metallic, flat, ring type, flange gaskets.		☑
A241	Natural Gas Pipe Casings: Neoprene transition end jackets shall be used between pipe under railway tracks and pipe casing.		☐
A215	Above Ground Pipe: Above ground gas supply pipe of NPS larger than 50 mm shall have non-metallic, flat, ring type, flange gaskets.		☑
A240	Natural Gas Pipe Casings: Plastic insulating spacers shall be used between casings and neoprene transition end jackets for pipes under railway tracks.		☐
A199	Below Ground Pipe		☐
A203	Below Ground Pipe: Joints in buried gas supply pipe shall be butt weld connections.		☐
A213	Above Ground Pipe		☐
A292	Input: The Natural Gas Pipeline network shall use commercial grade natural gas from the Gas Utility Company.		☐
A276	Plug type		☐
A210	Above Ground Pipe		☐

1 - 10 of 52 Next Last

Modify Query - Windows Internet Explo...

http://golevka.cstcis.cti.depaul.edu/Poirot/ArtifactDet

Query modification — Enable clouds: ☑

Additional words:
flange

Add word: | Add

Click on the underlined term to filter it out

Query:

- Joints and connections Gasket materials Found in Query be of either neoprene other similar material resistant to any action of gas. Natural rubber shall not be used.

Filter All | Clear All | Re-run query

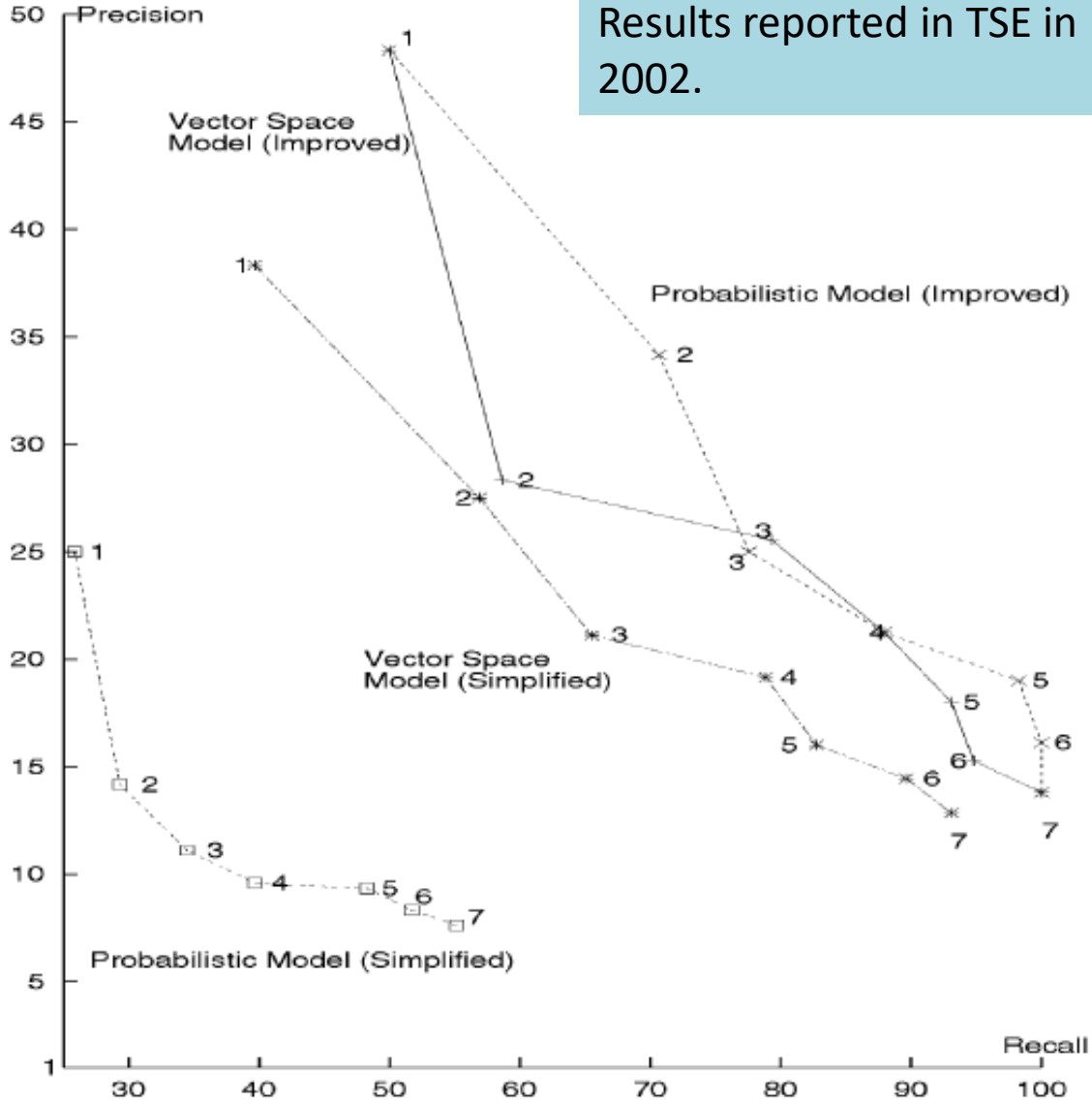
Internet | 100%

Many solutions have emerged from the research community.
Poirot is a service hosted at DePaul.

The Quest to Automate

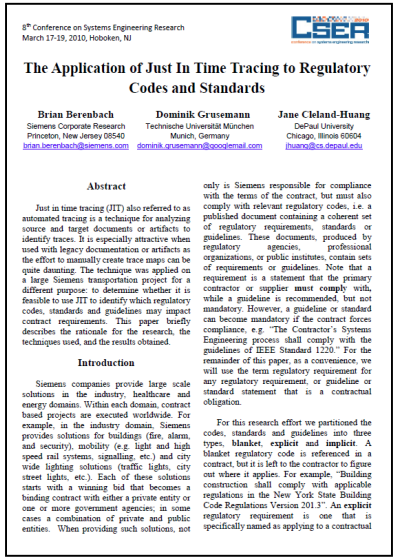
Giulio Antoniol with others…

Seminal work in 2001 launched a new research direction – the quest to automate the traceability process.



Results reported in TSE in 2002.

Pilot Study (Example)



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

CSER

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

Brian Berenbach
Siemens Corporate Research
Princeton, New Jersey 08540
brian.berenbach@siemens.com

Dominik Grusemann
Technische Universität München
Munich, Germany
dominik.grusemann@googlemail.com

Jane Cleland-Huang
DePaul University
Chicago, Illinois 60604
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 2013". An **explicit** regulatory requirement is one that is specifically named as applying to a contractual

Experiment 1: Accuracy of links

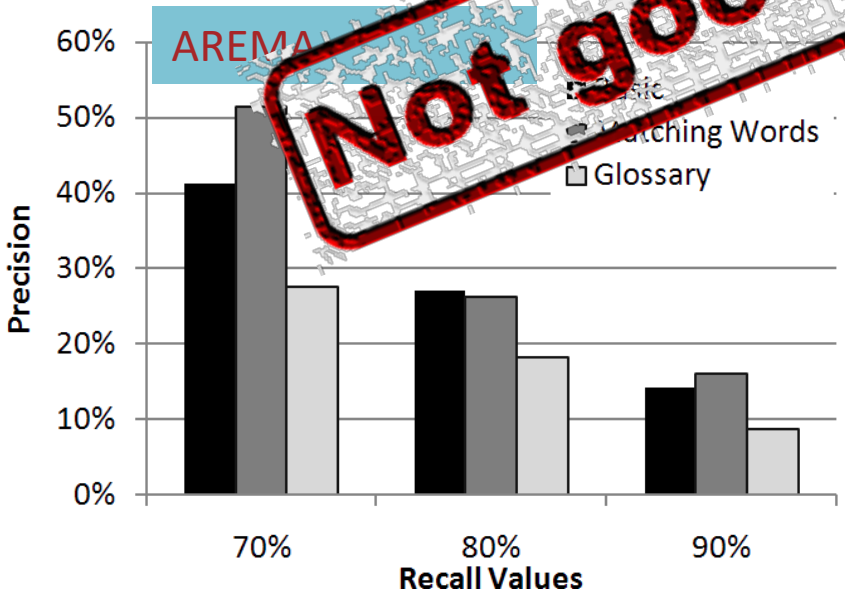
1. Export contractual and component level requirements from DOORS.
2. Develop customized parser and extract AREMA and CAN/ CSA regulatory codes from pdf files.
3. Use Poirot to generate traceability links between individual regulations and requirements.
4. Manually develop & validate an answer - set depicting a correct set of traceability links between regulatory codes and requirements.
5. Evaluate results and compute recall and precision metrics.

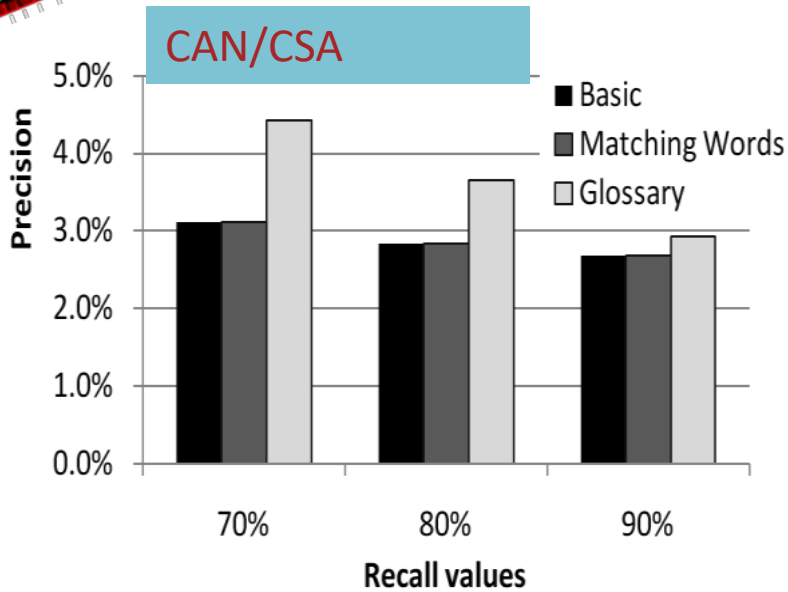
Experiment 2: Identifying applicable segments of requirements and codes

1. Split regulatory codes into segments.
2. Split requirements into segments.
3. Use Poirot to generate traceability links between individual regulations and individual requirements.
4. Compute impact metrics. Generate candidate links [illegible] sorted value.
5. Develop validated answer set depicting a trace matrix between regulatory code and requirement segments.
6. Compare results from Poirot with validated answer set from (5) and compute recall and precision metrics.

About 80% accuracy.

Not good enough!





How this talk is structured

- Quick Overview of Traceability
- FOSE Challenges
- Towards More Intelligent Tracing Solutions
  - An Expert Traceability System
  - Acquiring Domain Knowledge
  - Configuring & Optimizing a Trace Engine
  - TiQi: Naturally Speaking
- Transition to Practice
- Closing Comments

The Traceability Gap

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:

FOCUS: SAFETY-CRITICAL SOFTWARE

Strategic Traceability for Safety-Critical Projects

Patrick Mäder, Ilmenau Technical University

Paul L. Jones and Yi Zhang, US Food and Drug Administration

Jane Cleland-Huang, DePaul University

An evaluation of traceability information for 30 submissions prepared by manufacturers for review at the US Food and Drug Administration identifies nine widespread traceability problems that affected regulators' ability to evaluate products safely in a timely manner.

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

In this article, we focus on traceability's role in establishing evidence that device specifications and design implementations satisfy safety-and-risk-control measures (see the "Traceability Standards in Safety-Critical Projects" sidebar).[2] Creating and maintaining trace links can be an arduous, error-prone, and costly process that can have a significant effect on the cost of developing and maintaining a product.[3] Traceability practices, therefore, need to be strategically planned and carefully implemented to provide cost-effective support for evaluating and demonstrating safety-and-risk-control measures.[4] When traceability isn't implemented strategically, individual stakeholders might create traces that they personally consider to be important or attempt to provide complete trace coverage without considering how the traceability data will be used. The brute-force approach to traceability has been shown in practice to be difficult to implement, almost impossible to maintain, and not particularly helpful for providing evidence that a system or device is safe for its intended use.

This article presents a set of strategic traceability, derived from our own observations of effective traceability in industrial projects and supported by current literature.[5–8] We also identify nine recurring problems, each of which reduces the effectiveness of traceability information and in some cases makes it difficult experienced by regulators in evaluating product safety. All the observations in this article are based on actual observations, but the illustrative examples are either fictitious or built on obfuscated data.

Effective Practices for Tracing in Safety-Critical Projects

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

Mind the Gap: Assessing the Conformance of Software Traceability to Relevant Guidelines

Patrick Rempel, Patrick Mäder, Tobias Kuschke, and Jane Cleland-Huang
Technische Universität Ilmenau
Software Systems/Process Informatics Group
Ilmenau, Germany
{patrick.rempel|patrick.maeder|tito-ilmenau.de
DePaul University Chicago
Systems and Requirements Engineering Center
Chicago, IL, USA
jhuang@cs.depaul.edu

ABSTRACT

A wide variety of guidelines in safety-critical industries such as automotive, medical devices, and railway communications, specify that traceability must be used to demonstrate that a rigorous process was followed and to provide evidence that the system is safe for use. In practice, there is a gap between what is prescribed by guidelines and what is implemented in practice, making it difficult for organizations and certifiers to fully evaluate the safety of the software system. In this paper we present an approach, which parses a standard to extract a Traceability Information Model showing expected practices, and then analyzes the data for a specific project to identify areas of traceability failure. Missing traceability paths, redundant and/or inconsistent data, and other problems are highlighted. We use our approach to evaluate the traceability of seven safety-critical software systems. We found that none of the evaluated projects contained traceability that was fully conform to its relevant guidelines.

Keywords

software development, documentation, requirements traceability, safety critical, assessment, software traceability, software and system safety

I. INTRODUCTION

Developing software for regulated industries is a challenging process. Not only must the software deliver the required features, but it must do so in a way that ensures that the system is safe and secure for its intended use [5], [6]. To this end safety-critical, and other regulated systems, must meet stringent guidelines before they can be approved or certified for use [6], [12], [13], [3]. For example, software developed for the aerospace industry must comply to the RTCA/DO178B guideline and/or DO-178B, while software developed for European railway communication, signaling, and processing systems, must comply to EN50128 [8] [3]. Most guidelines prescribe a set of steps, deliverable documents, and exit criteria focused around planning, analysis and design, implementation, verification and validation, configuration management, and quality assurance activities. In addition they often provide specific guidelines for the creation and use of traceability in the project. For example, depending upon the criticality level of a requirement, the US Federal Aviation Authority guideline DO-178B requires traceability from requirements to design, and from requirements to source code and executable object code [11].

Software traceability serves an important role in demonstrating that a delivered software system satisfies its software design constraints and mitigates all identified hazards. When executed correctly, it demonstrates that a rigorous software development process has been established and systematically followed. The Center of Excellence for Software and Systems Traceability (CoEST) defines a *trace link* as "a specified association between a pair of artifacts, one comprising the source artifact and one comprising the target artifact" [9, 7].

1.1 Traceability Challenges

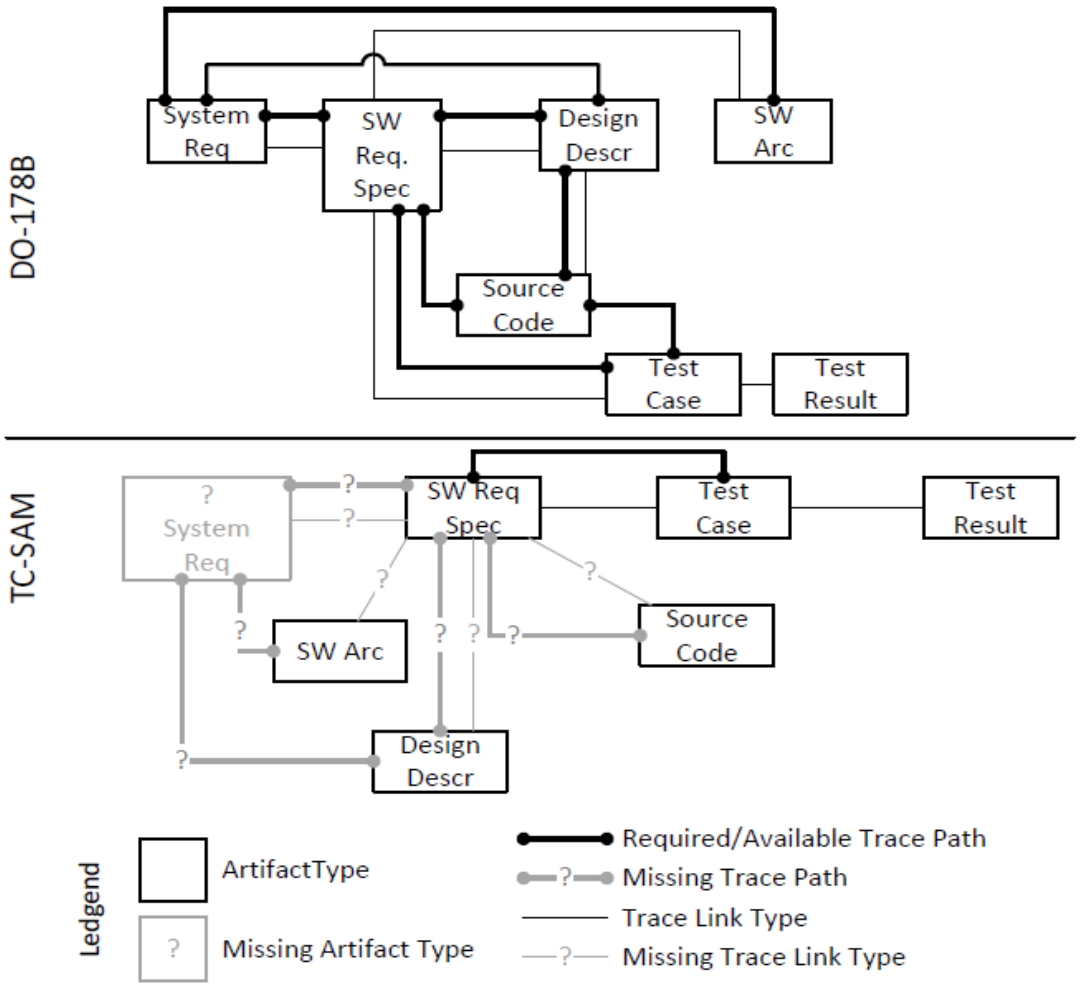
Unfortunately, organizations struggle to establish and maintain accurate and complete sets of traceability links [4, 11] [4, 15, 3]. A prior analysis of the raw traceability information submitted by various organizations to the US Food and Drug Administration (FDA) as part of the medical device approval process, showed a significant *traceability gap* between the traceability expectations as laid out in the FDA's "Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices" [13], and the traceability data documented in the actual submissions [2]. While all of the submissions made some attempt to satisfy the FDA's traceability guidelines, serious deficiencies were found in almost all the submissions in terms of missing traceability paths; missing and redundant links; and problems in trace granularity which made it very difficult to understand the rationale for individual links. These observations draw attention to the traceability gap which exists between the traceability specified and required in guidelines and the actual traceability information provided in practice. The deficiencies make it difficult for the manufacturers to ensure that the system they are building is safe for use and for certifiers to understand whether they should approve it for use in the public domain.

Our goal is to provide formalisms, metrics, and tool support for evaluating traceability coverage with respect to the guidelines that are *relevant* for a specific project. This is

Our study of the traceability components of Medical Device submissions to the FDA showed **incomplete and sometimes entirely missing trace links**, inaccurate, redundant traceability – delivered through a big bang solution.

A **formal comparison of five safety-critical software systems** against prescribed guidelines showed **similar traceability problems**.

The Traceability Gap



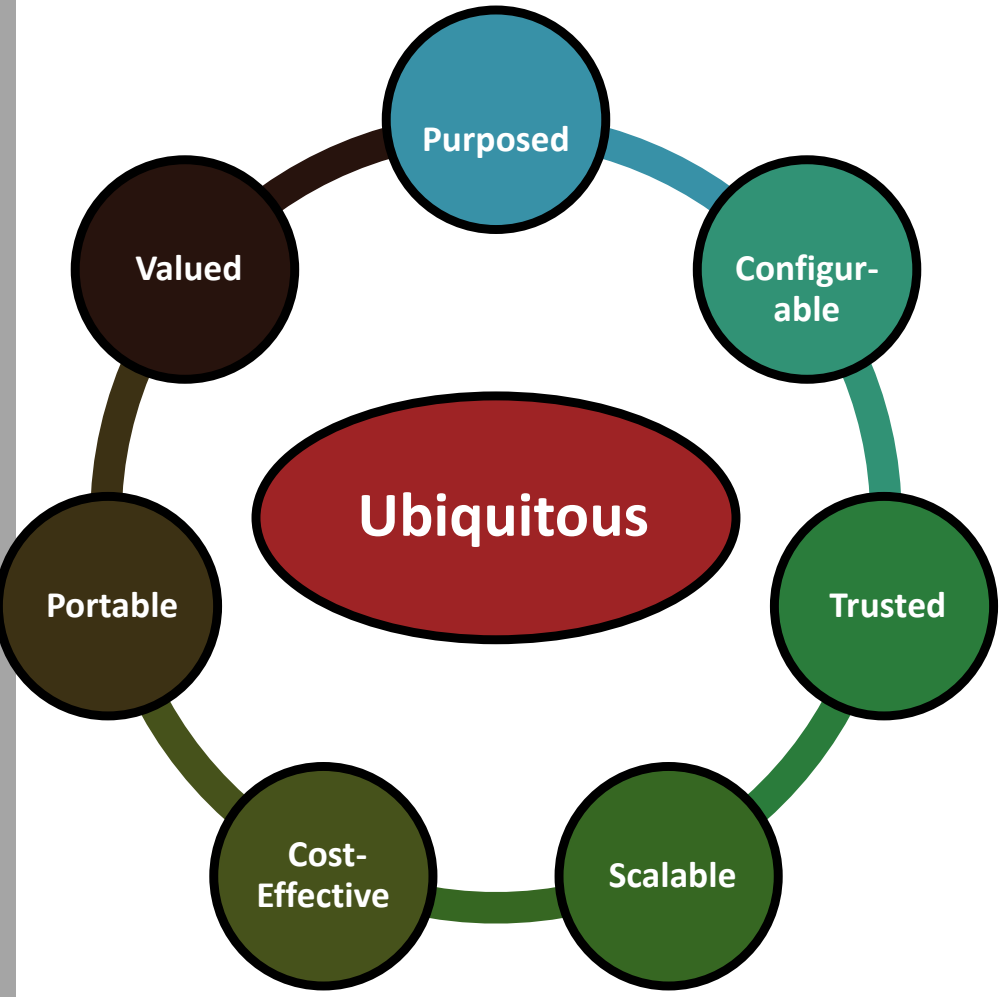
Patrick Maeder

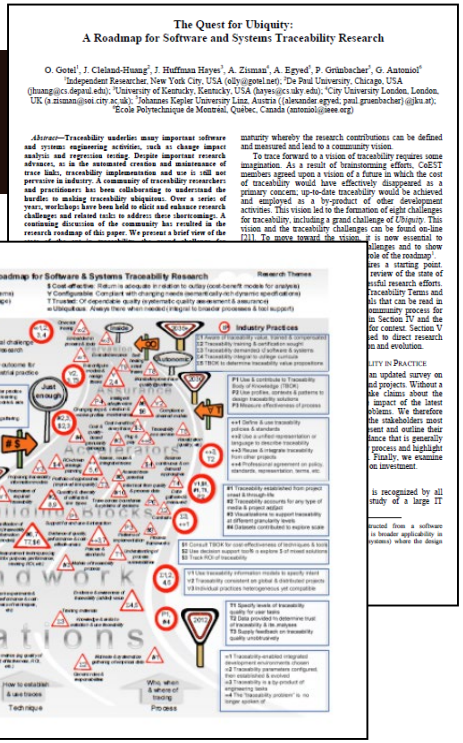
Patrick Rempel

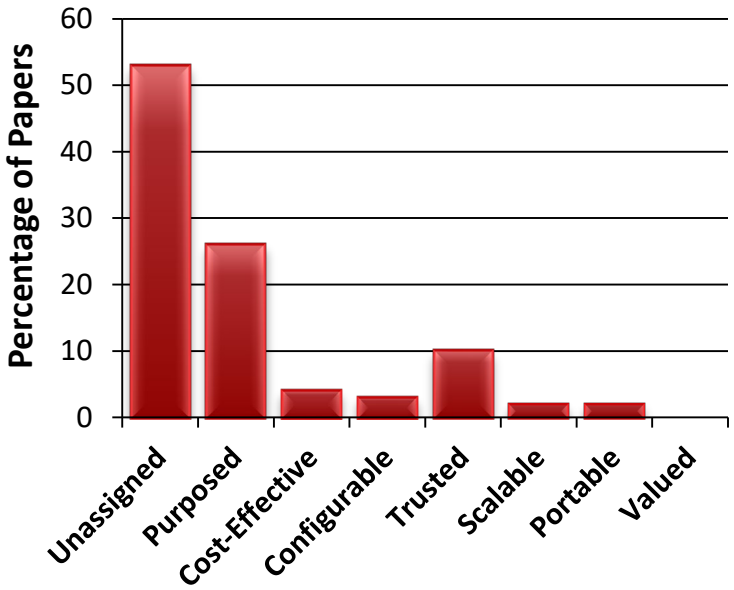
Mind the Gap: Assessing the Conformance of Software Traceability to Relevant Guidelines, Patrick Rempel, Patrick Mäder, Tobias Kuschke (TU Ilmenau), and Jane Cleland-Huang (DePaul), ICSE 2014, Hyderabad, India

Why is the Traceability Problem so hard to solve?

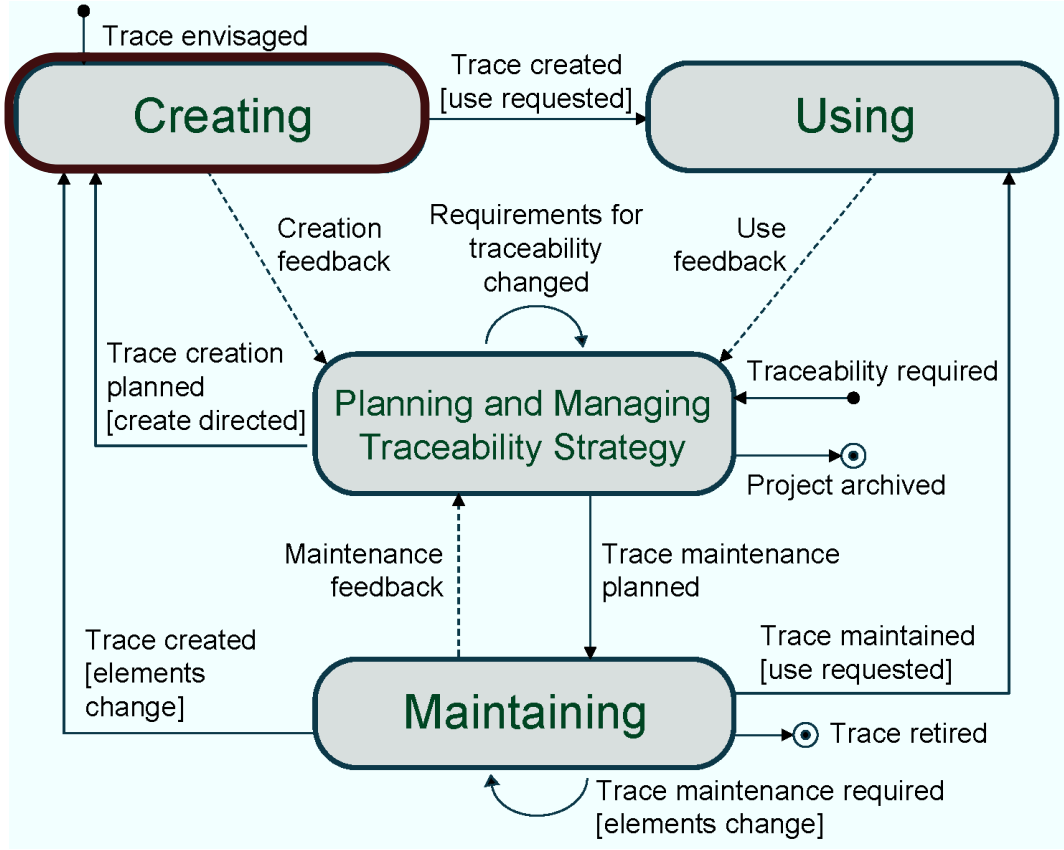
A Goal Oriented Perspective

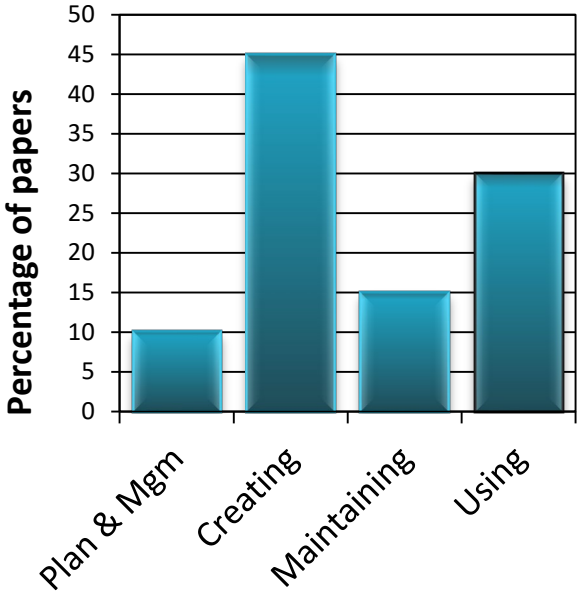


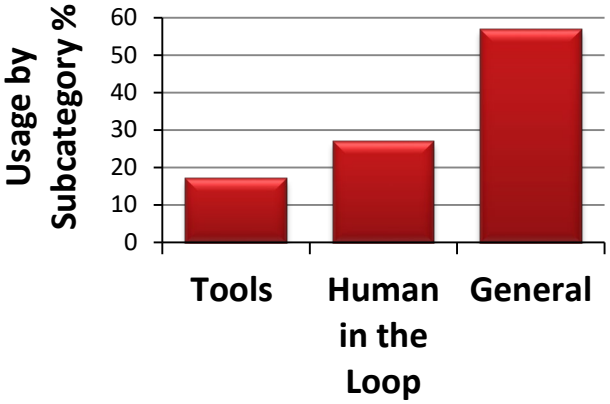

The Quest for Ubiquity:
A Roadmap for Software and Systems Traceability Research



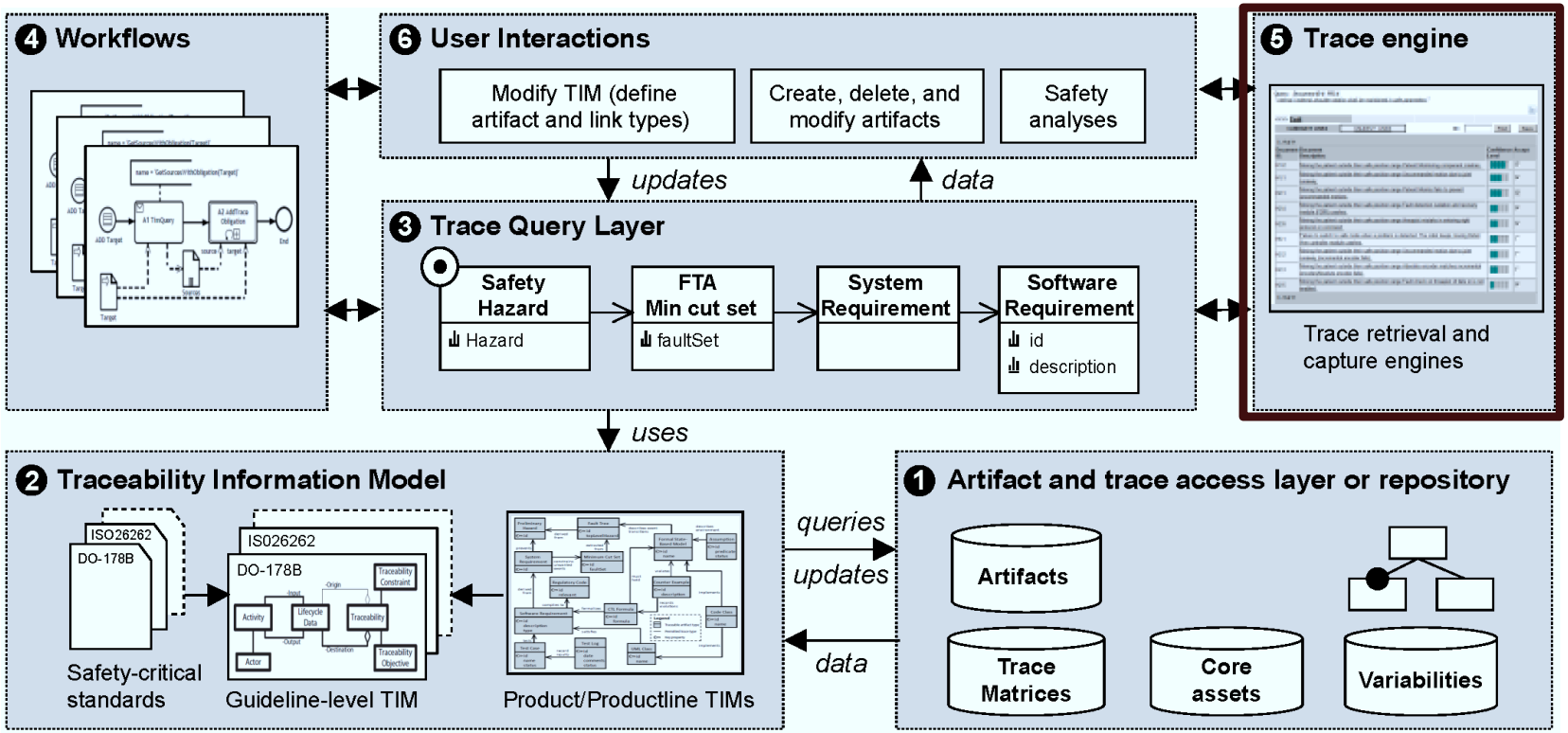
A Process Perspective

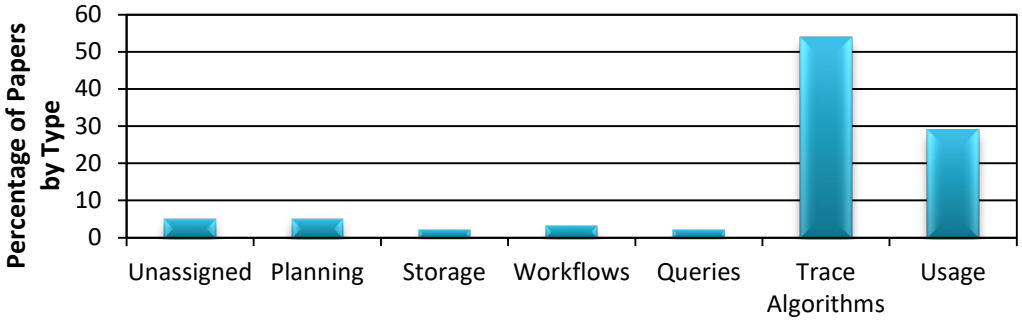






A Technical Perspective





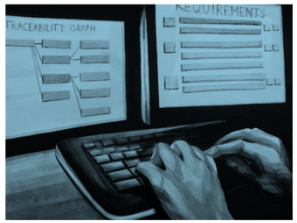
Future of Software Traceability (FOSE)

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 prototypical stakeholder requirements for traceability, including s

RD-1.2 Em by stakeho

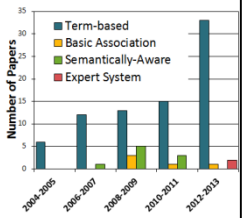
Planning and Managing

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 process framework to guide practitioners, develop standards, inform tools, and enable training. (Purposes: RT4,RT7) Configurable.RT1)

RD-2.2 Prepare a family of standardized TIMs and usage guidance. Adaptable and extensible meta-models need to provide the capability for a project or organization to grow its traceability competence via well-defined pathways (Configurable.RT3)

RD- 3.1 More Intelligence.....



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

RD-3.1 Develop intelligent tracing solutions which are not constrained by

RD- 3.2: Leave no exhaust

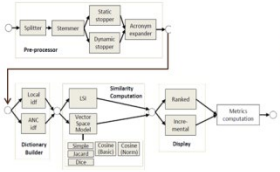
Develop techniques that monitor the environment and human activities – and use this information to infer new trace links and to maintain existing ones:

Techniques include:

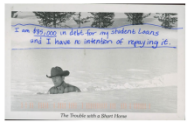
- Eye-trackers
- Email
- Version control logs
- User clicks

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.



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 various artifacts including requirements, design, and code.

RD-4.2 Develop heuristics and probabilistic approaches for evolving trace links as artifacts change.

RD-4.3 Integrate prospective capture with link

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.

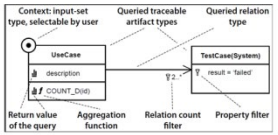
RD-5.2 Develop algorithms and supporting tools for automatically evaluating the correctness of existing trace links, whether created manually or with tool-support.

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.

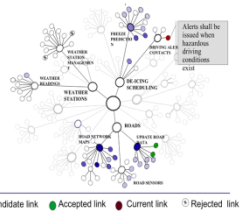


Visualizing and Understanding Results

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.



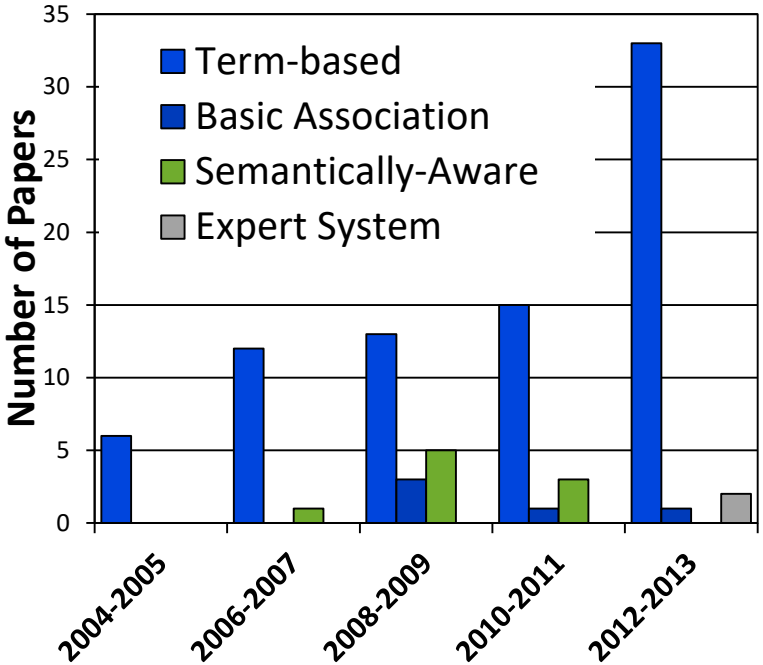
RD-7.1 Construct a taxonomy of available visualizations and fundamental traceability tasks. Bridge these by exploring the basic visualization principles that they either provide or demand.

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

RD- 3.1 More Intelligence…..



Despite at least a decade of effort we are running up against a glass ceiling when it comes to automated traceability. That ceiling is called **"Term Mismatch"**

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.

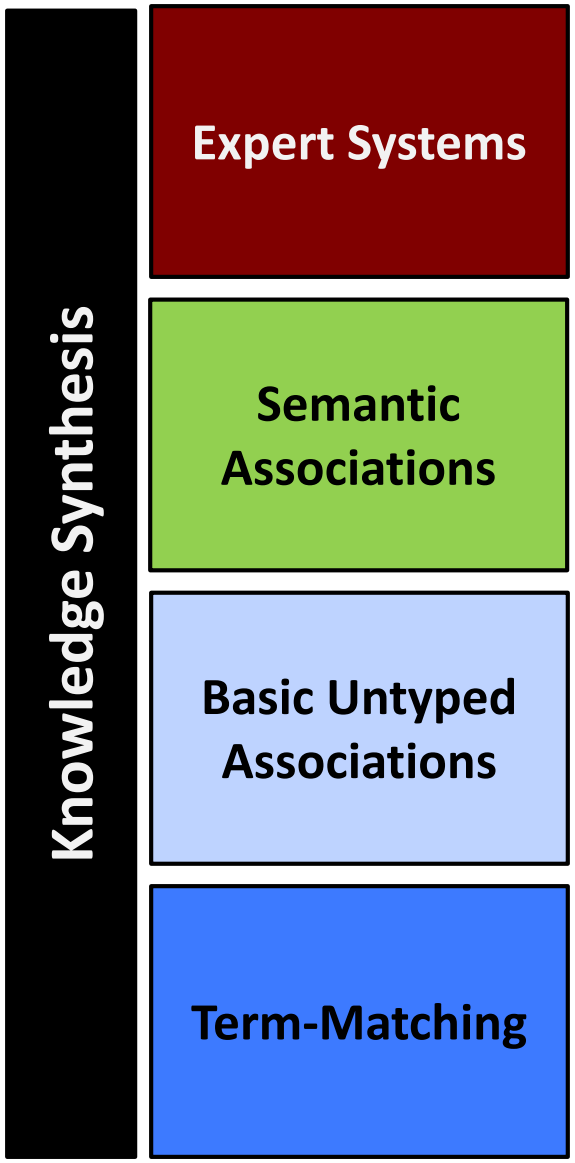
How this talk is structured

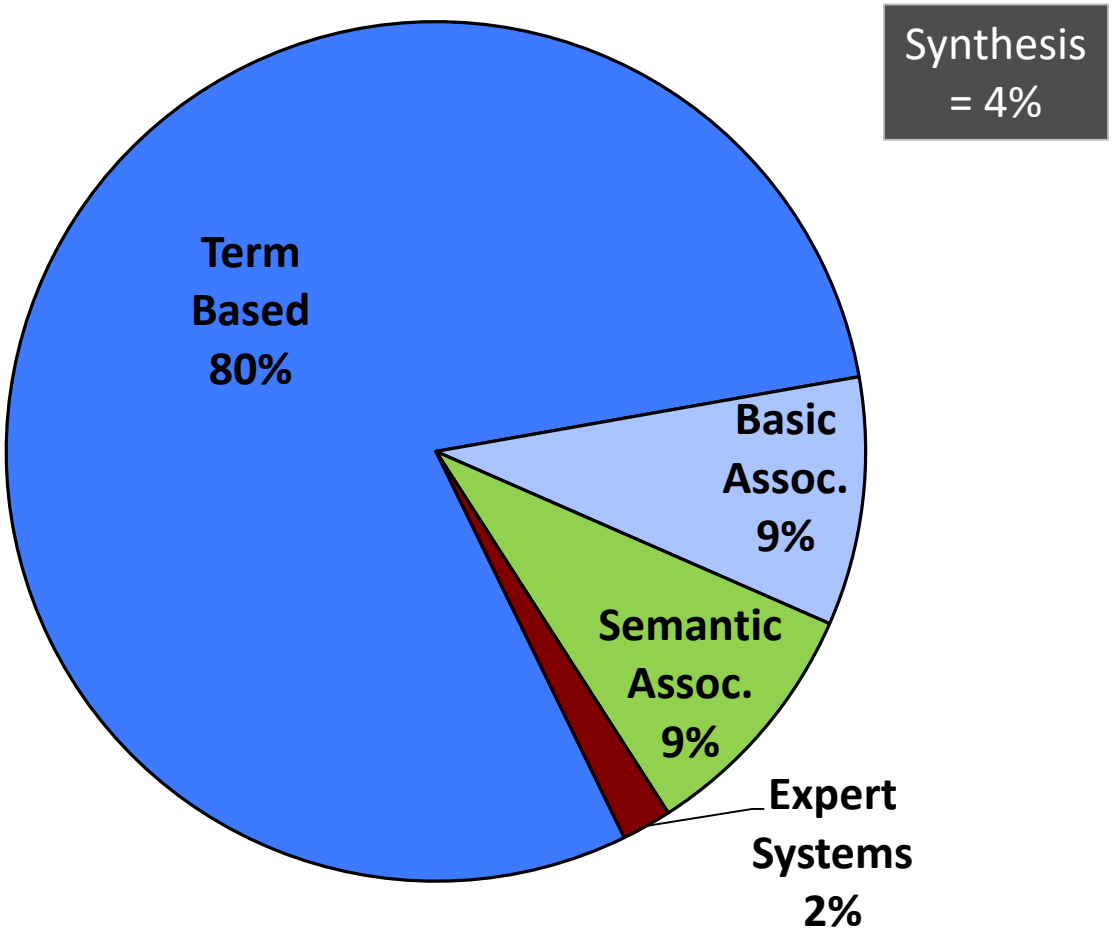
- Quick Overview of Traceability
- Open Challenges
- Towards More Intelligent Tracing Solutions
  - An Expert Traceability System
  - Acquiring Domain Knowledge
  - Configuring & Optimizing a Trace Engine
  - TiQi: Naturally Speaking
- Transition to Practice
- Closing Comments

Can we have more intelligent Traceability?

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

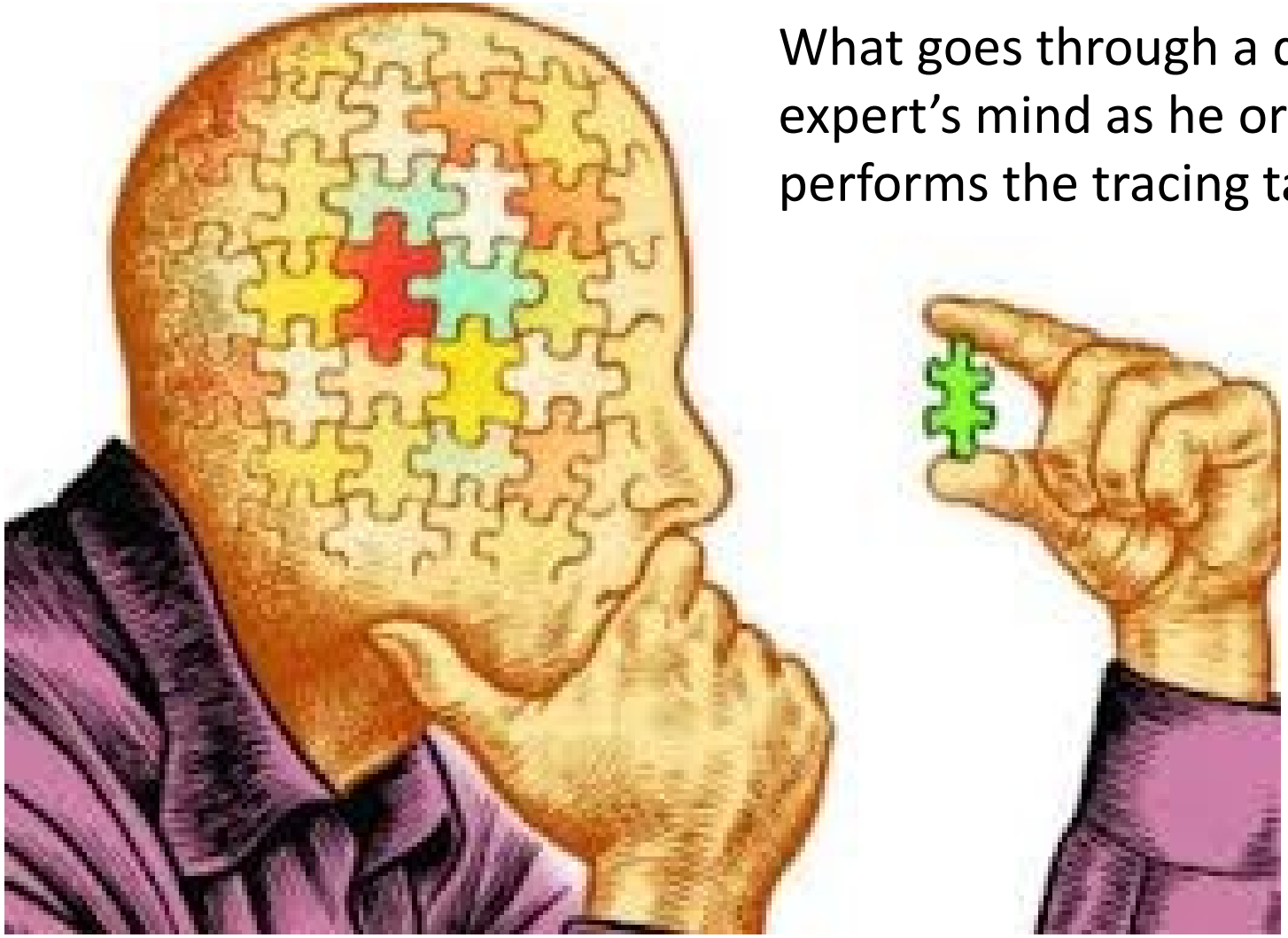
Publication Categories (2004-2013)





Current solutions are quite dumb and focus on term based solutions.

Towards more Intelligent Traceability



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

Human Analysts think about concepts…

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.

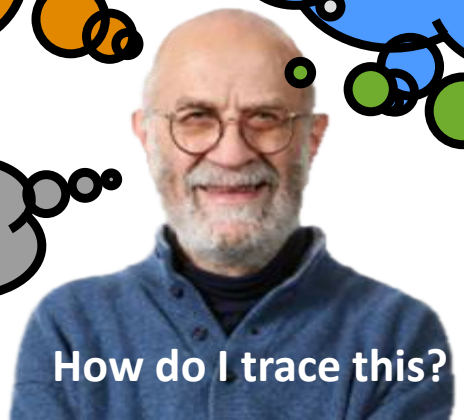
The Highway Wayside Segment shall transmit information to the automobile controller in the form of WSMs.

WIU = Wayside Interface Unit **and** is located in a Highway Wayside Segment.

Automobile controller is part of the automobile

WSM and wayside status message are equivalent.

Broadcast is similar to transmit



Both artifacts involve Highway Wayside Segment **sending** wayside status message to automobiles

Domain Centric Expert Traceability

1. Identify **Verbs**.
2. Categorize each verb by its **usage group**.
3. Identify **nouns and noun phrases** associated with each verb. Assign **thematic roles** to each noun and noun phrase.

4. Identify the **semantic group** of each action unit's verb.
5. Apply **heuristics** to pairs of action units across source and target artifacts to determine whether a trace link exists. **Create trace links accordingly**.

An Action Frame
is normally defined by:

- A **verb**
- A **semantic type**.
- A set of nouns & noun phrases assigned various **thematic roles**.

Jin Guo

Natawut Monaikul

Cody Plepel

Phase 1: Action Frame Extraction Rules

Usage Group: UG3

A1: Any critical failure during the Disengaged Mode will force the OBM to **enter** the Failed Mode.

Action: enter

Semantic Grps: Motive

Theme: obm

Location: fail mode

1. Identify the verb and its usage group. e.g. **enter = UG3**
2. Lookup the semantic group(s) of the verb. **i.e. Motive**
3. Use the Stanford Parser to generate a dependency tree showing POS.
4. Use the action unit extraction rules associated with UG3 to identify relevant parts of speech.
   For example in UG3:

   Subject = theme | Object=location

Thematic Roles

In Linguistics there are hundreds of thematic roles. These are the ones that we have found to be common in the transportation domain.

Agent: the doer of the action, as in
"The DOHS system shall send..."

Theme: the object upon which the action is performed, as in:
"...shall send a message"

Recipient: the object which receives the action, as in:
"...send a message to a subsystem"

Instrument: the object with which the action is performed, as in:
"...send a message through an interface"

Location: the place in which the action is completed, as in
"...send a message within the limits"

Initial Location: the place in which the action is initiated, as in
"...leave the area"

Heuristic # 1: Basic Match

Link

A1: 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.

Action: broadcast

Semantic Groups: Transmisive

Agent: WIU

Recipient: automobile

Theme: message

WSM is one kind of message

A2: The Highway Wayside Segment shall transmit informa obile Segment in the f

WIU is part of the wayside segment

Action: transmit

Semantic Groups: Transmisive

Agent: Wayside Segment

Recipient: Automobile Segment

Theme: WSM

1. **Semantic group** must match ✓
2. Compare **Agents**, **Recipients**, and **Instruments**. If a pair is present, it must exhibit an exact or hierarchical match. If only one side is present, this step is skipped. ✓ ✓
3. Compare **Themes** and **Locations**. ✓
   Either both sides of a pair, or neither side of a pair must be present. If they are both present, they must exhibit an exact or hierarchical match.

Heuristic # 2: Transmissive-Receptive

Link

A1: The OBM shall support reception and decomposition of Wayside Status Messages.
Action: reception
Semantic Groups: Receptive
Recipients : OBM
Themes: wayside status message

A2: The Wayside Se transmit informatio Automobile Segmen of Wayside Status Messages.
Action: transmit
Semantic Groups: Transmissive
Agents: wayside segment
Recipients: automobile segment
Themes: wayside status message

OBM is part of the Automobile segment

1. **Semantic group** in one action group is **Transmissive** and the other is **receptive**. **(Applicable)**
2. Compare **Agents, Recipients**, ✓ **Instruments**, **Locations**. If a pair is present, it must exhibit an exact or hierarchical match. If only one side is present, this step is skipped.
3. Compare **Themes**. ✓ Both sides of a pair must be present and they must exhibit an exact or hierarchical match.

Current Groups and Heuristics

24 Semantic Groups

1. Administrative
2. Affirmative
3. Calculative
4. Causative
5. Cooperative
6. Descriptive
7. Effective
8. Enforcive
9. Evasive
10. Inceptive
11. Inclusive
12. Inspective
13. Interactive
14. Motive
15. Necessitative
16. Permissive
17. Preservative
18. Receptive
19. Regulative
20. Reparative
21. Submissive
22. Supportive
23. Transgressive
24. Transmissive

Heuristics
300 pairs possible
27 pairs defined

Basic Match: 19
Motive-Motive:5
Administrative-Inceptive: 4
Motive-Permissive: 3
Transmissive-Receptive: 3
Calculative-Enforcive: 2
Transmissive-Transmissive: 2
Permissive-Permissive: 2
Regulative-Inclusive: 1
Transmissive-Descriptive: 1
Receptive-Enforcive: 1

Question: How generalizable are these rules across domains?

Domain Centric Intelligent Traceability (DoCIT)

❶ All regulatory codes and requirements are parsed into action units using NLP.

The Wayside Segment shall transmit Wayside Status Messages (WSMs) to the Automobile Segment.

Action	transmit	**Semantic Group**	Transmissive
Properties	Agent	wayside segment	
	Theme	wayside status message	
	Recipient	automobile segment	
Instance of an Action Unit			

❷ Trace links are established between action units using sophisticated trace link heuristics. Potentially 576 different heuristics.

The Wayside Segment shall transmit Wayside Status Messages (WSMs) to the Automobile Segment.

Action	transmit	**Semantic Group**	Transmissive
Properties	Agent	wayside segment	
	Theme	wayside status message	
	Recipient	automobile segment	
Action Unit from Source Artifact			

The approaching automobile can also request the status message from the WIU at any time.

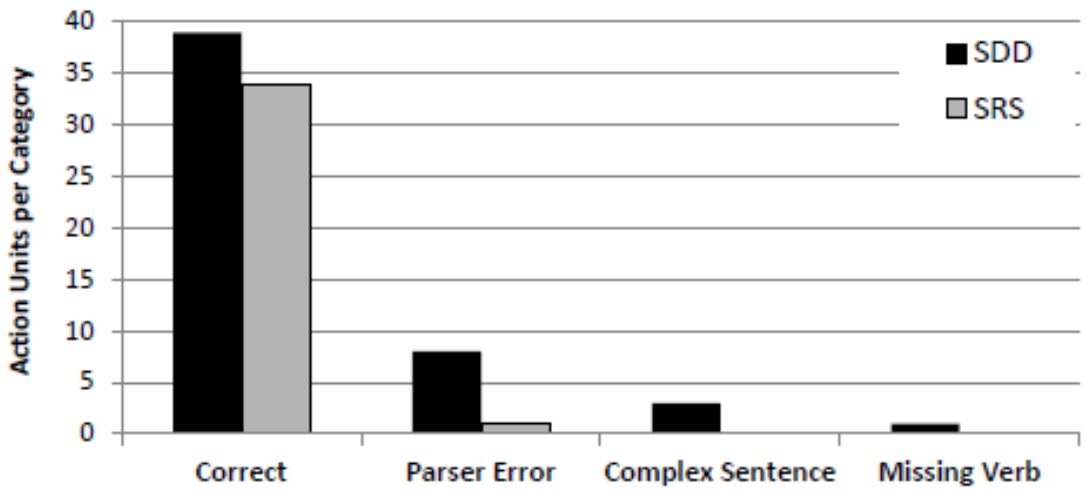
Action	request	**Semantic Group**	Receptive
Properties	Agent	WIU	
	Theme	status message	
	Recipient	automobile	
Action Unit from Target Artifact			

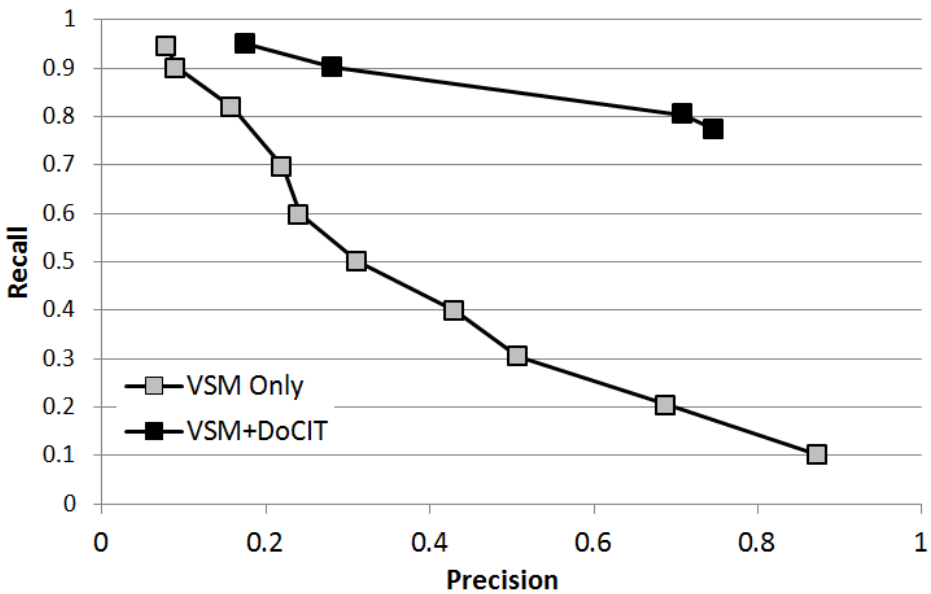
Trace link

❸ Entire process supported by a knowledge base containing domain facts, mapping facts, and link heuristics. This KB needs to be learned continually for each and every domain.

Domain Centric Expert Traceability

97% of action units extracted correctly from SRS
76.5% of action units extracted correctly from SDD





Very significant improvement in both recall and precision when DoCIT is used.

This is just our first prototype. We believe we can do much better.

How this talk is structured

- Quick Overview of Traceability
- Open Challenges
- Towards More Intelligent Tracing Solutions
  - An Expert Traceability System
  - Acquiring Domain Knowledge
  - Configuring & Optimizing a Trace Engine
  - TiQi: Naturally Speaking
- Transition to Practice
- Closing Comments

Knowledge Acquisition

We have shown that DoCIT can deliver quite accurate trace links if it has sufficient domain knowledge.

The next challenge is real-time learning.

Domain facts

Verb facts

Dependency mappings

Link heuristics

Acquiring Knowledge

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.

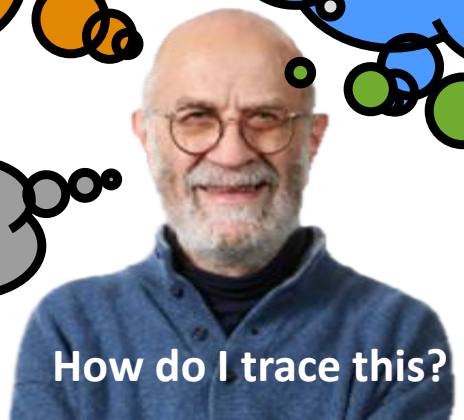
The Highway Wayside Segment shall transmit information to the automobile controller in the form of WSMs.

WIU = Wayside Interface Unit **and** is located in a Highway Wayside Segment.

Automobile controller is part of the automobile

WSM and wayside status message are equivalent.

Broadcast is similar to transmit



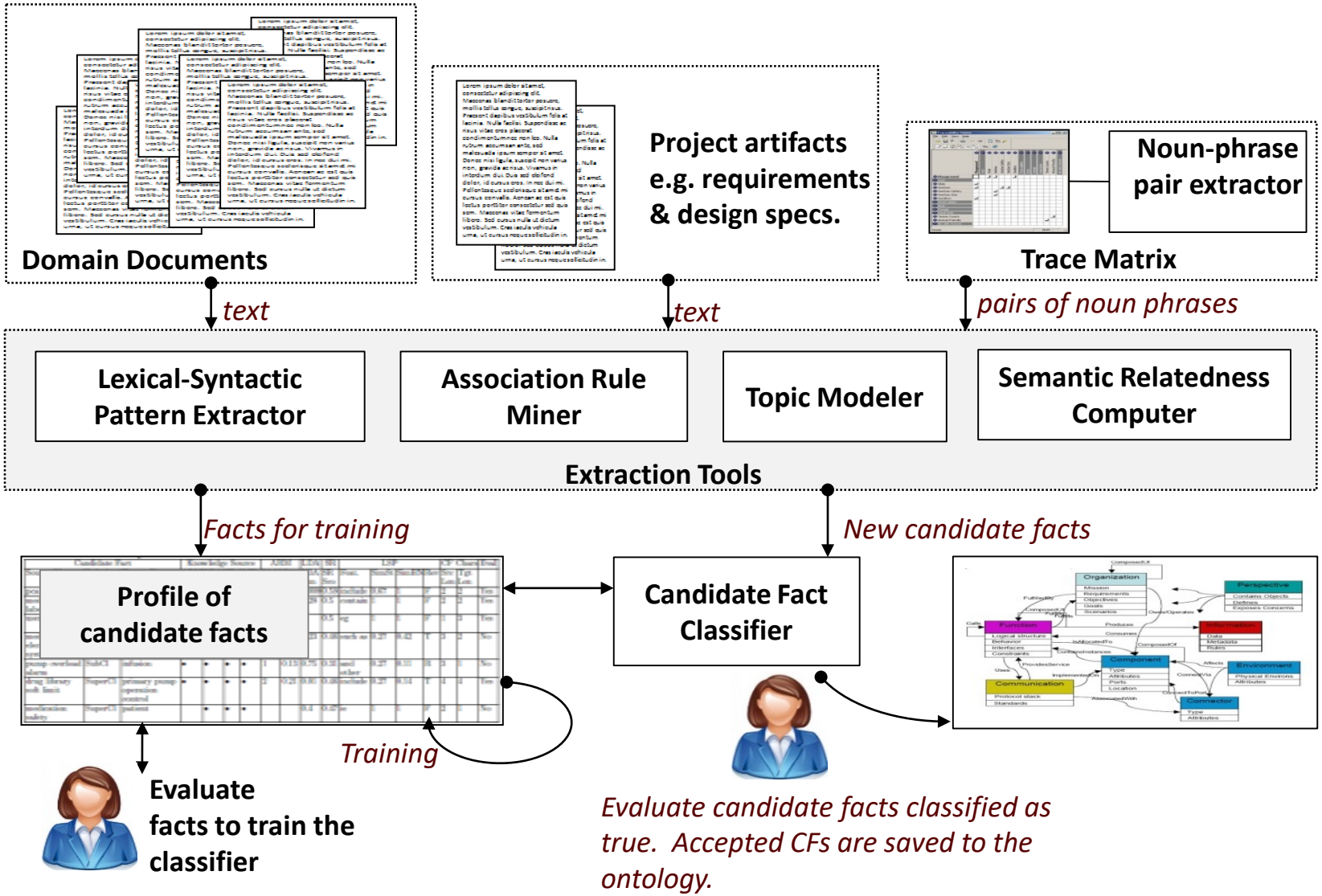
Both artifacts involve Highway Wayside Segment **sending** wayside status message to automobiles

How many domain terms?

	Single Words	Phrases	Overlap	
			Words	Phrases
DVC	13,960	104,026	23%	37%
	1,784	4,173		
EHR	3,118	971	16%	4%
	621	89		
MIP	3,409	18,164	98%	55%
	3,632	10,664		

Counts include only domain-specific terms and phrases – with high domain specificity.

Learning Domain Ontology



Training the Classifier

Candidate Fact			Knowledge Source				ARM		LDA	SR	LSP				Length		
Source Phrase	Relation	Target Phrase	ARM	LDA	SR	LSP	Link Cnt	Cos Sim	LDA Sim	SR Sco	Feat.	Sim St	Sim HN	Rev	Src Len	Tgt Len	Eval
pca pump	superClass	drug library	•	•	•	•	4	0.24	0.008	0.59	include	0.67	1	F	2	2	Yes
medication label	hasPart	bar code		•	•	•			0.29	0.5	contain	1	1	F	2	2	Yes
user	superClass	health care provider			•	•				0.5	eg	1	1	F	1	3	Yes
medication electrical system	superClass	infusion rate	•	•	•	•	1	0.24	0.23	0.46	such as	0.27	0.42	T	3	2	No
pump overload alarm	subClass	infusion	•	•	•	•	1	0.13	0.75	0.31	and other	0.27	0.11	T	3	1	No
drug library soft limit	superClass	primary pump operation control	•	•	•	•	2	0.21	0.01	0.46	include	0.27	0.14	T	4	4	Yes
medication safety	superClass	patient		•	•	•			0.4	0.47	ie	1	1	F	2	1	No

Dataset	Class	Random Forest					Naïve Bayes				
		TP	FP	Prec	Rec	FM	TP	20	Prec	Rec	FM
DVC	Invalid	0.66	0.31	0.68	0.66	0.67	0.83	0.45	0.64	0.83	0.72
	Valid	0.68	0.33	0.67	0.68	0.68	0.54	0.16	0.76	0.54	0.63
EHR	Invalid	0.75	0.26	0.74	0.75	0.74	0.78	0.39	0.66	0.78	0.72
	Valid	0.73	0.24	0.75	0.73	0.74	0.6	0.21	0.73	0.6	0.66
MIP	Invalid	0.77	0.2	0.78	0.77	0.78	0.74	0.39	0.65	0.74	0.69
	Valid	0.79	0.22	0.78	0.79	0.78	0.61	0.26	0.7	0.61	0.65

Putting the Human in the Loop

Ontology Builder

Source:

The scanner shall read and authenticate information from the drug's package label.

Target:

The scanner reads an optical or RFID code on the patient, clinician, and the drug container that is loaded into the reservoir.

Please evaluate the following facts for possible addition to the ontology.

Drug package label information	contains	Drug container RFID code

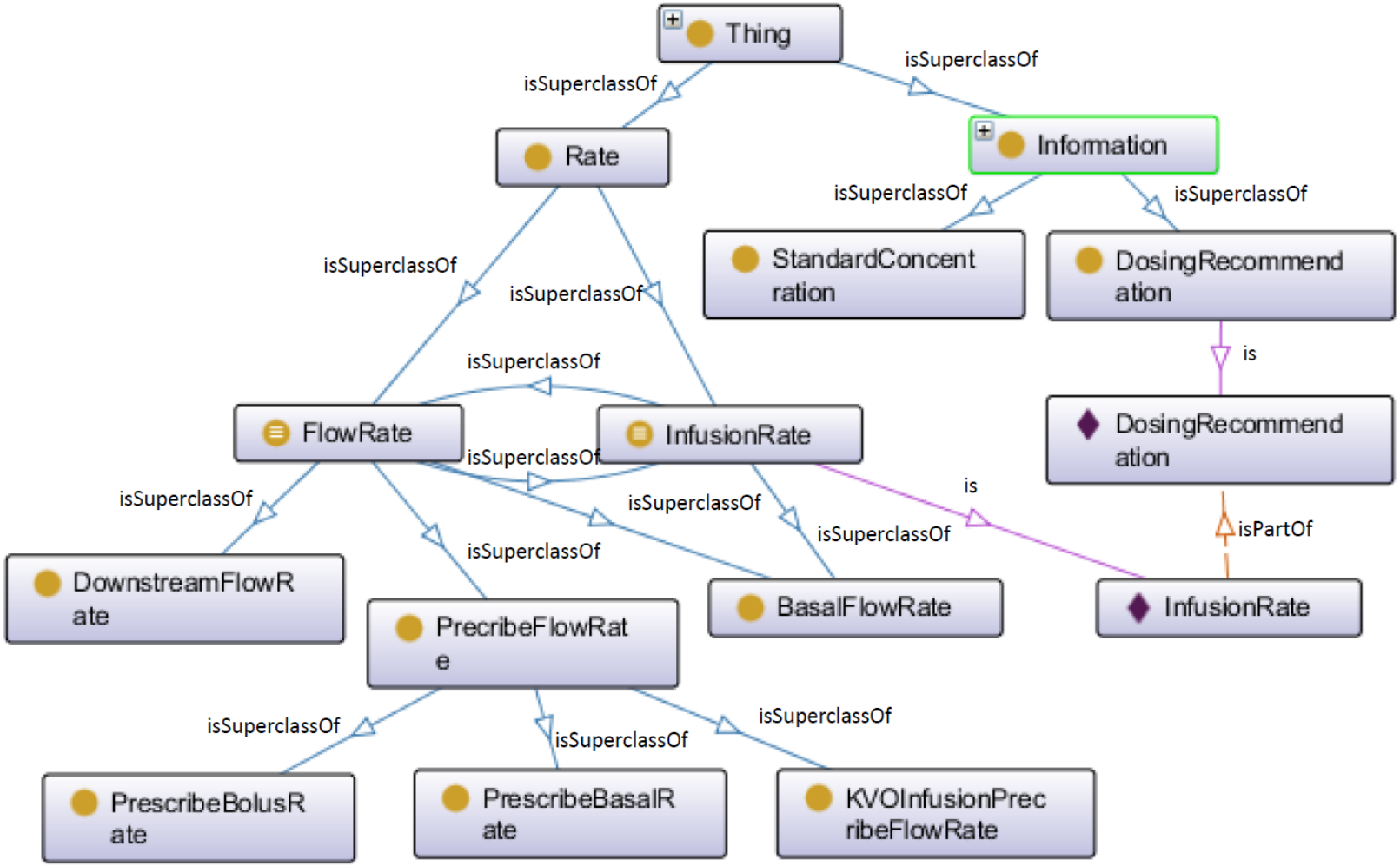
Accept **Reject** **Edit** **Undecided**

Drug package label information	contains	Clinician Optical Code

Accept **Reject** **Edit** **Undecided**

Previous Next Close

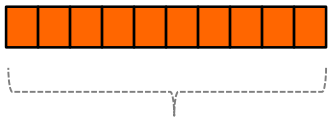
..and finally the Ontology



Accuracy is improving – but Ontology is still incomplete…

Learning Rules…

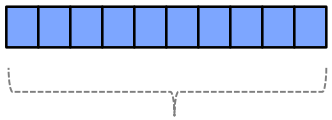
Semantic group per verb



Each verb (v) is assigned a cell. The cell's value is set to 1..n representing the **semantic group** in which the corresponding verb *v* belongs.

Cell count: numVerbs (v)
Cell range: 1..n

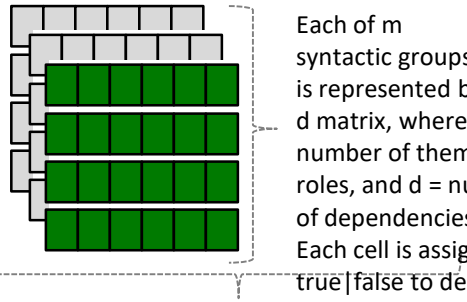
Syntactic group per verb



Each verb (v) is assigned a cell. The cell's value is set to 1..m representing the **syntactic group** in which the corresponding verb *v* belongs.

Cells: numVerbs (v)
Cell range: 1..m

Dependency mappings



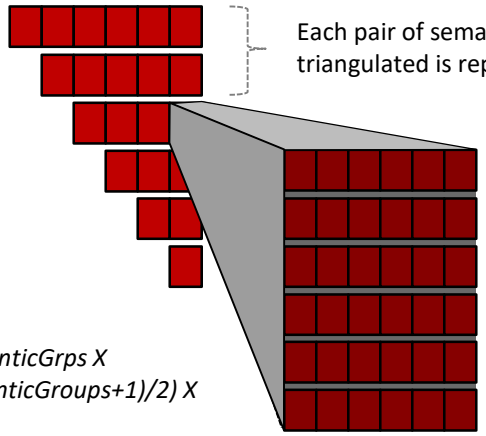
Each of m syntactic groups is represented by a t X d matrix, where t = number of thematic roles, and d = number of dependencies. Each cell is assigned true|false to depict whether the mapping exists or not.

Cells:
NumSyntacticGrps (m)
** numRoles (t)*
Range: 0|1

More permutations than the number of nanoseconds in the universe.

Currently exploring multi-objective solution using NGSA-II.

Link Heuristics



Each pair of semantic groups (m) – upper triangulated is represented by a t X t matrix.

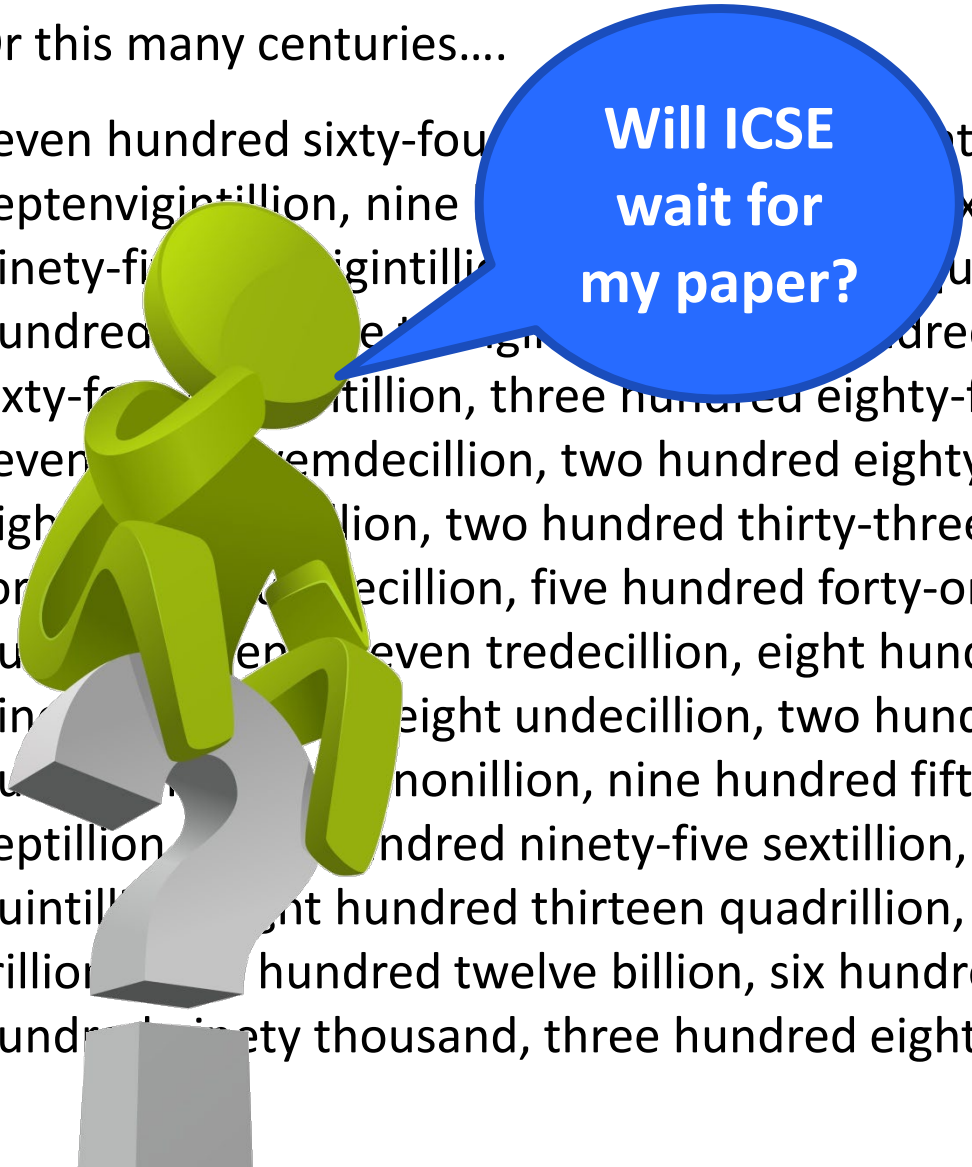
Each row represents the occurrence of a thematic role in Action Unit 1, and each column represents the occurrence of the thematic role in Action Unit 2. Cell values range from -1..4 where -1 = no rule for matching thematic roles, and 0..3 represents the matching rules described in this paper.

Cells:
(NumSemanticGrps X (numSemanticGroups+1)/2) X numRoles^2)
Each cell -1|0..4

We missed the ICSE deadline last Fall!

Or this many centuries….

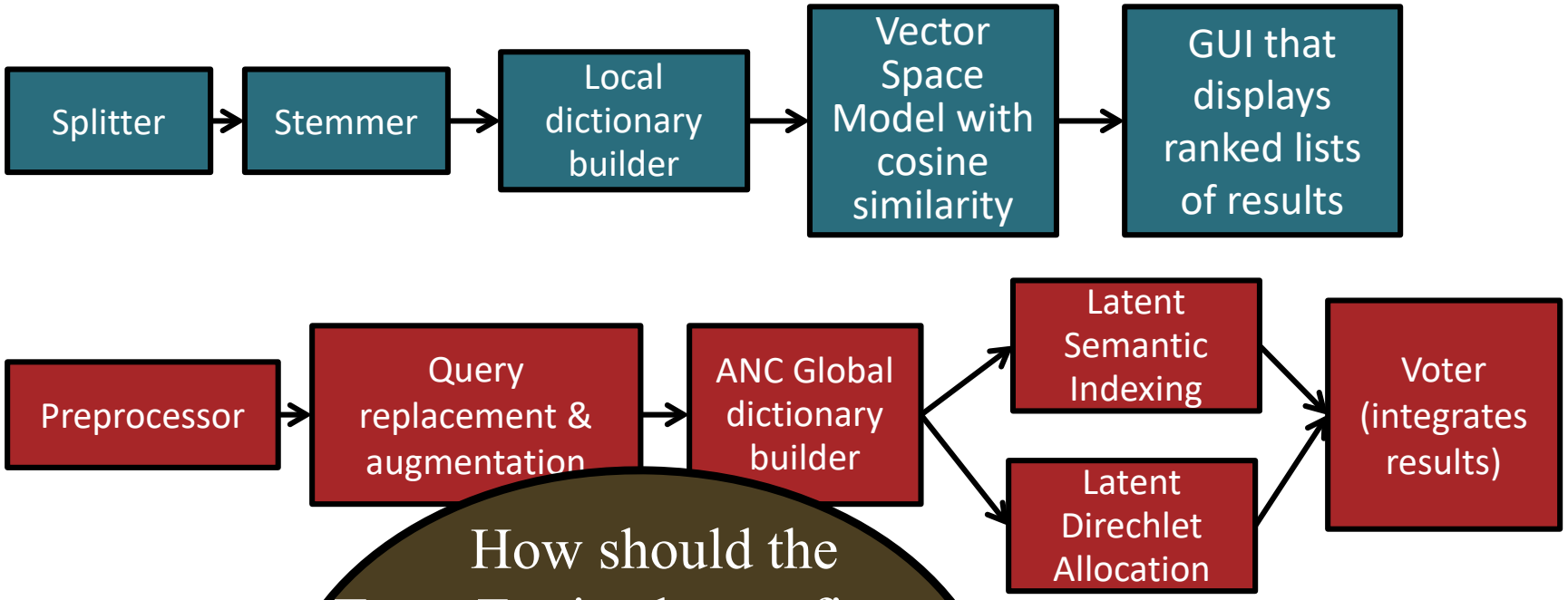
seven hundred sixty-fou t hundred eighty-one
septenvigintillion, nine xvigintillion, one hundred
ninety-fi igintilli uattuorvigintillion, three
hundred dred eighty-one duovigintillion,
sixty-f tillion, three hundred eighty-five vigintillion, nine hundred
seven emdecillion, two hundred eighty-eight octodecillion, twenty-
eigh lion, two hundred thirty-three sexdecillion, one hundred
for ecillion, five hundred forty-one quattuordecillion, five
hu en even tredecillion, eight hundred sixty-four duodecillion,
nin eight undecillion, two hundred nine decillion, nine
hu nonillion, nine hundred fifty octillion, six hundred sixteen
septillion ndred ninety-five sextillion, nine hundred twenty-seven
quintil ht hundred thirteen quadrillion, seven hundred eighty-eight
trillion hundred twelve billion, six hundred twenty-nine million, three
hundr ety thousand, three hundred eighty-seven.

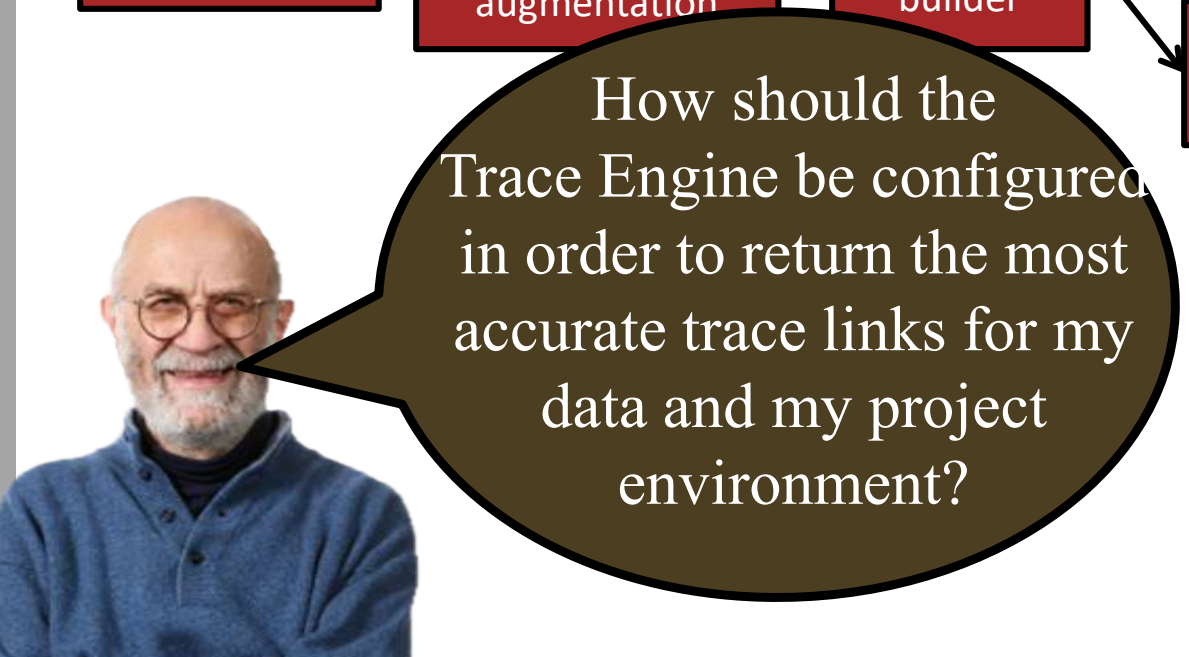


How this talk is structured

- Quick Overview of Traceability
- FOSE Open Challenges
- Towards More Intelligent Tracing Solutions
  - An Expert Traceability System
  - Acquiring Domain Knowledge
  - Configuring & Optimizing a Trace Engine
  - TiQi: Naturally Speaking
- Transition to Practice
- Closing Comments

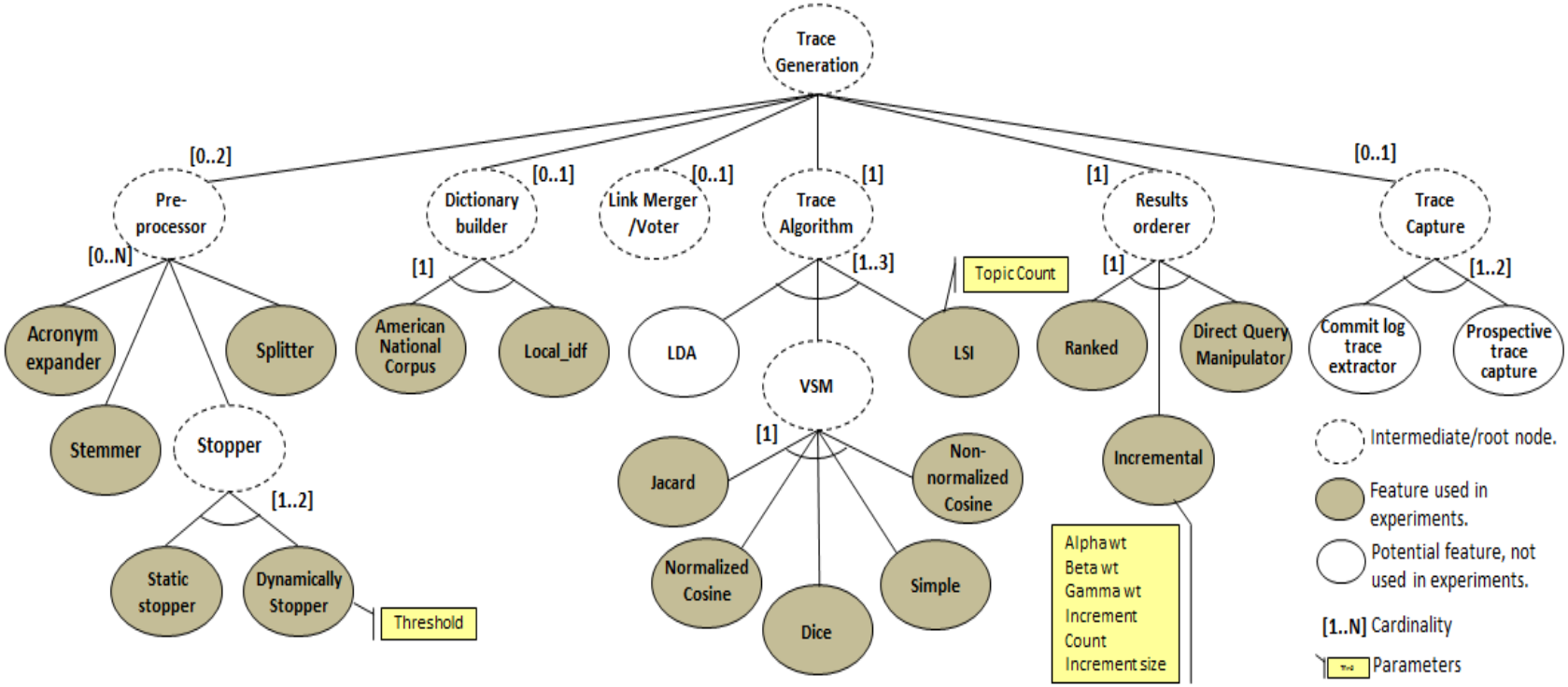
We can configure more intelligently too..





Important for industrial application and for creating research baselines.

Depict as a Feature Model



A feature model depicts commonalities and variabilities, cross-tree constraints, and configuration parameters.

1,644,408 configurations. With parameters at reasonable increments - over **seven trillion configurations**.

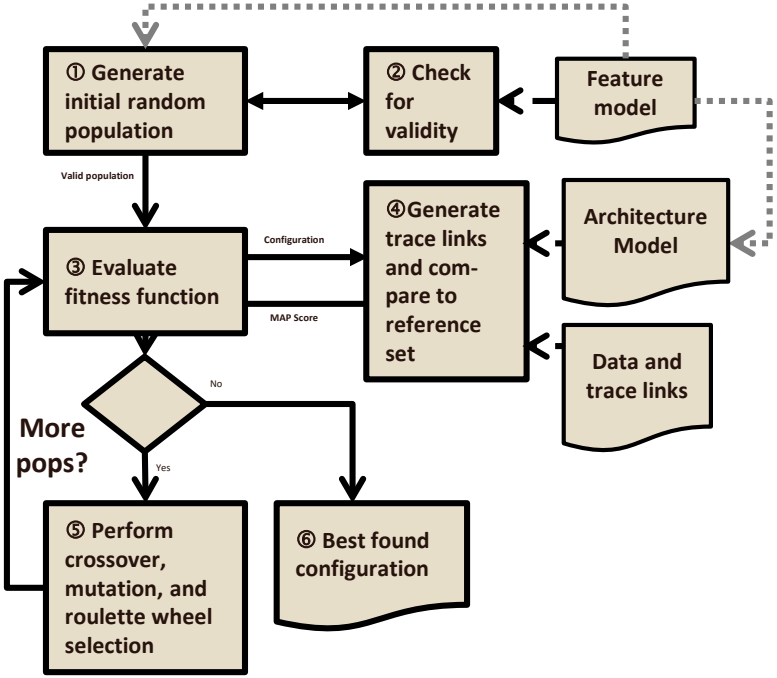
Searching for the Best Configuration

Initial Population

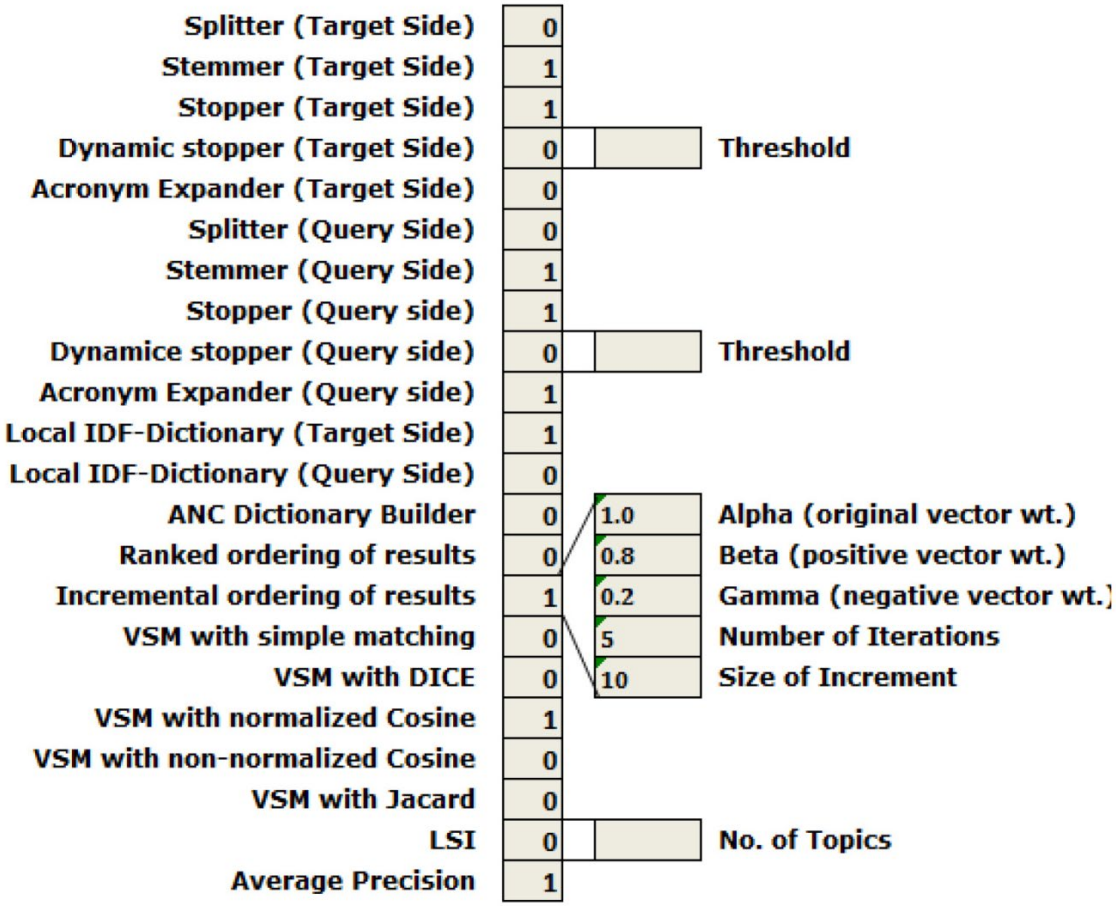
1. Randomly create an initial population of individuals
2. Evaluate the fitness of each individual in that population

Next Generation

3. Select individuals for **reproduction**
4. Breed new individuals through crossover and mutation to produce new offspring
5. Evaluate the individual **fitness** of new individuals
6. Repeat steps 3-5 until termination (time limit, sufficient fitness achieved, etc.)



Represent a valid configuration

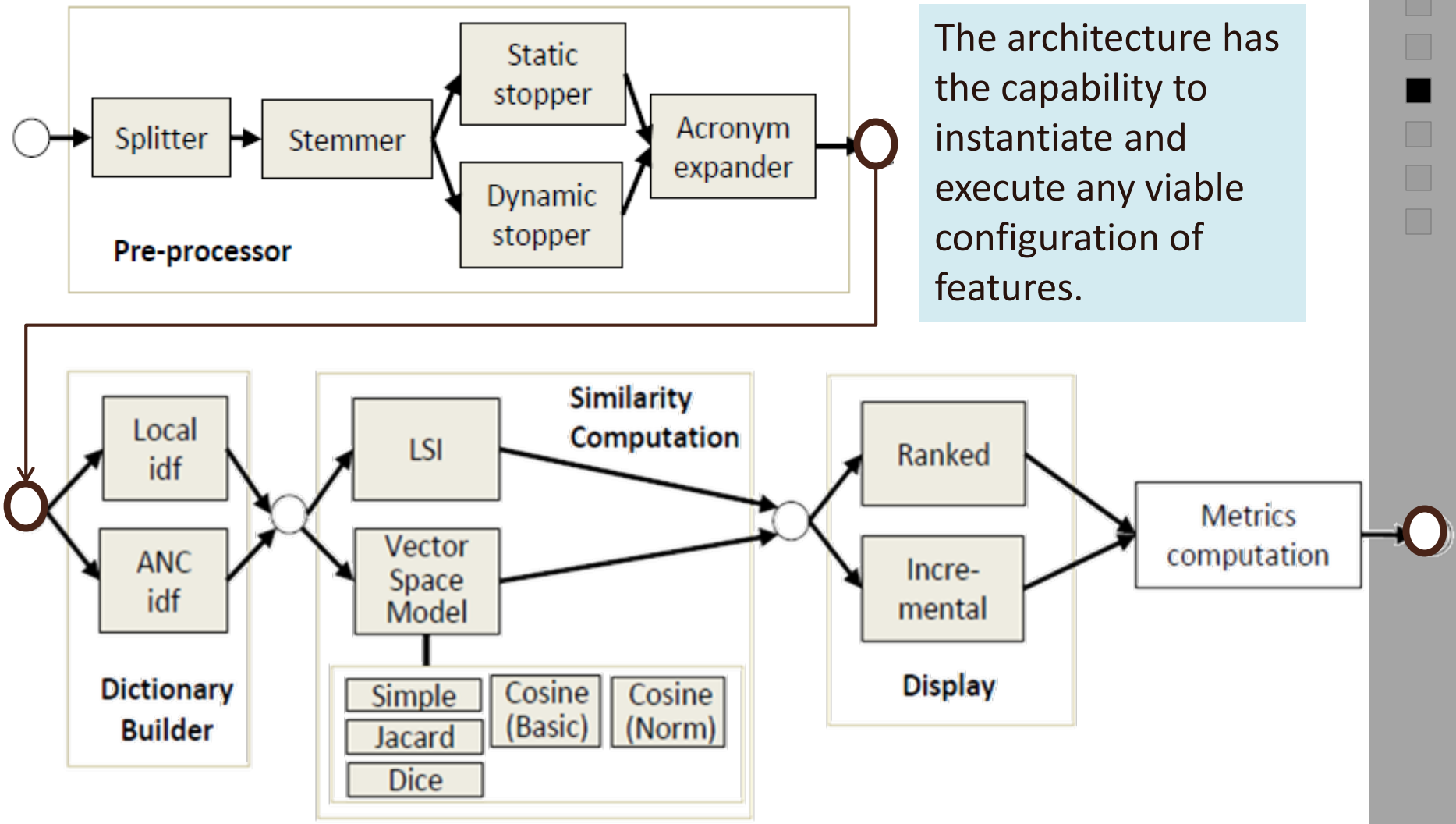


Each potential solution is represented as a **chromosome**.

Parameters are represented in the sub-hierarchy.

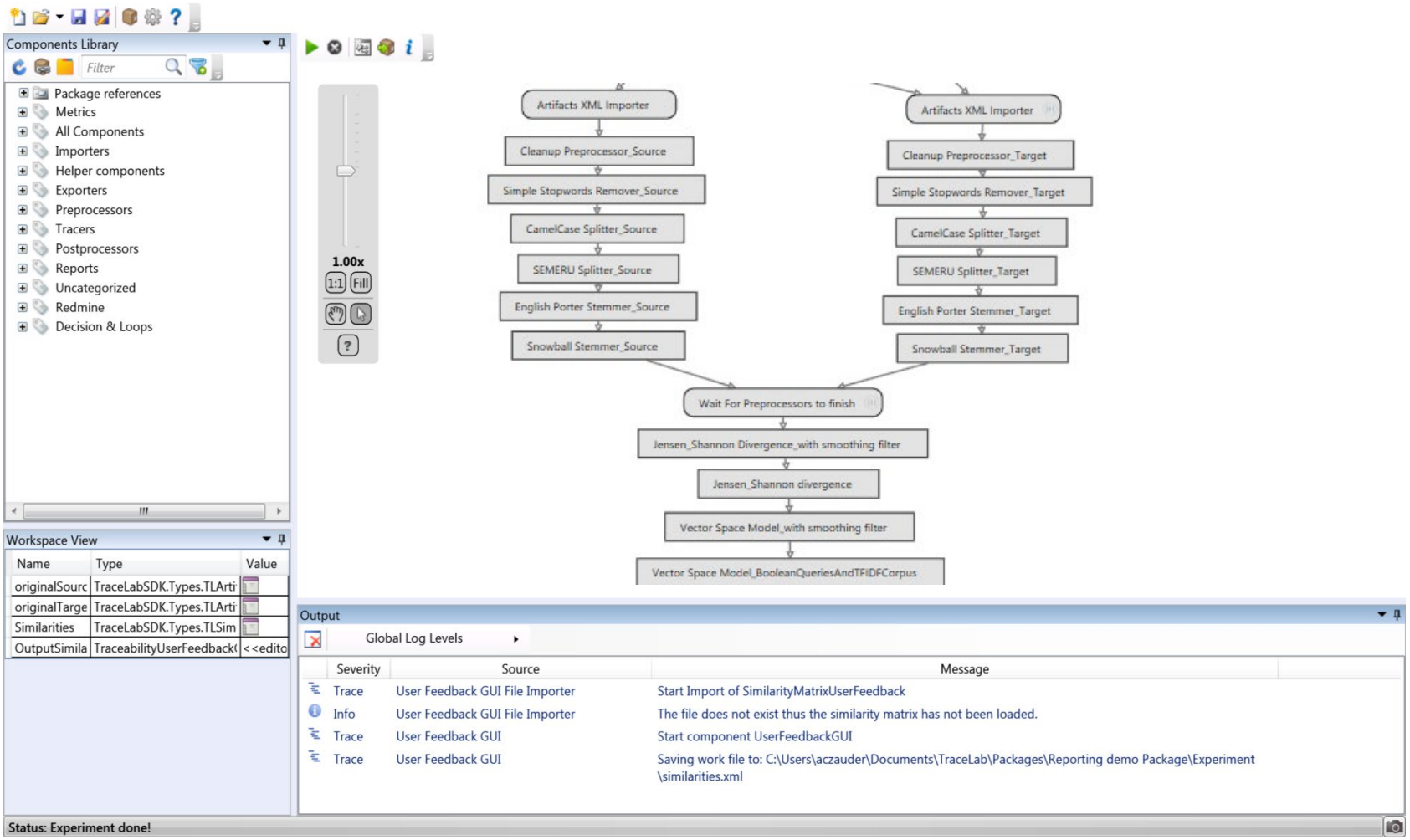
Each configuration is validated against the Feature Model

Instantiate the Configuration (Architecture)



The architecture has the capability to instantiate and execute any viable configuration of features.

Architecture Implemented in TraceLab

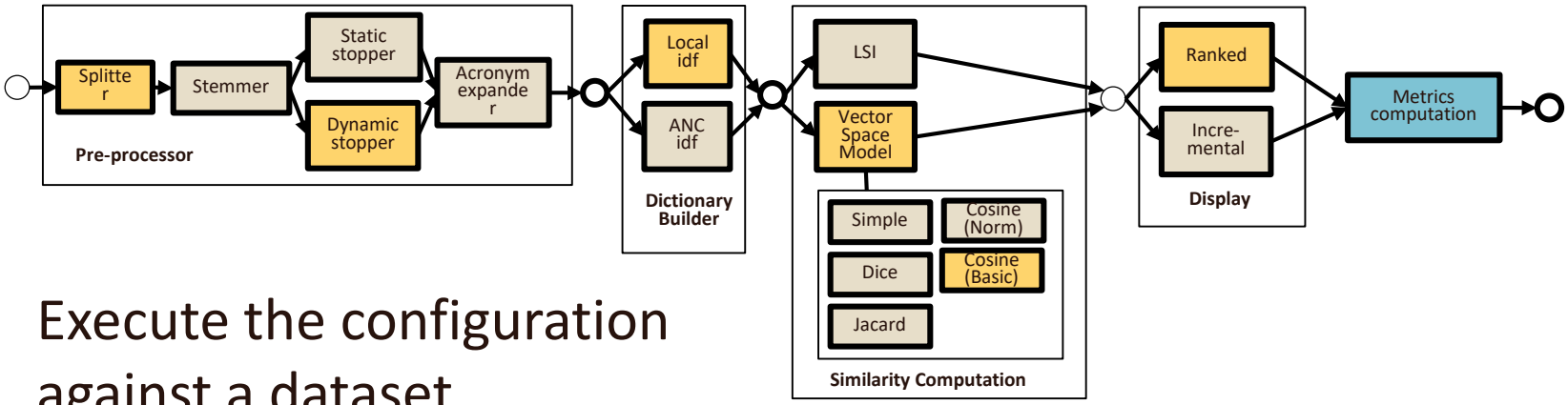


TraceLab is available for download from Coest.org

Next release of fully-functioning Multi-platform version in November, 2013.

Compute the Fitness Function

1. Instantiate the configuration



2. Execute the configuration against a dataset.
3. Compare generated links against the "answer set" of links.
4. Compute the Mean Average Precision

$$AP = \frac{\sum_{r=1}^{N}(P(r) \times isRelevant(r))}{|RelevantDocuments|}$$

$$MAP = \frac{\sum_{q=1}^{Q} AveragePrecision(q)}{Q}$$

MAP evaluates the extent to which good links are placed at the top of a ranked list of returned links. It computes average precision at a recall of 100%

5 Research Questions

Data Set	Domain	Source	Target	Links
E-Clinic	HealthCare	Use Cases (30)	Test Cases (47)	63
CM-1	NASA-Space	Requirements (22)	Design (46)	46
CCHIT	HealthCare	Regulatory Codes (453)	Requirements (958)	534
i-Trust	HealthCare	Requirements (131)	Code (332)	535
Industry 1	Transportation	Requirements (442)	Design (3104)	6961
Industry 2	Transportation	Requirements (224)	Design (945)	700

Basic Evaluation

- **RQ1**: Does each dataset of source and target artifacts have a distinct optimal trace configuration?
- **RQ2**: Is there a single configuration which performs well on all datasets?
- **RQ3**: Does each pair of artifact types have a distinct top performing configuration?

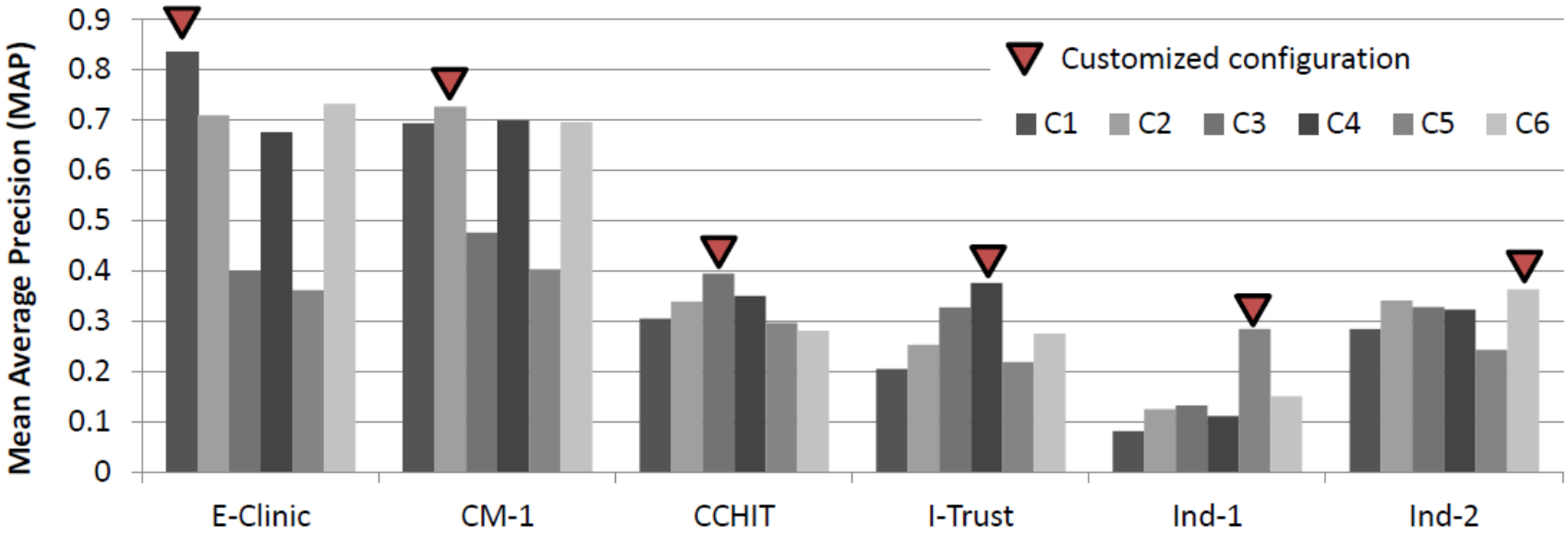
DTC in Practice

- **RQ4**: Are configurations stable over time?
- **RQ5**: Do reconfigurations of the trace infrastructure lead to better trace quality in future traces?

Evaluation: RQ2

Is there a single configuration which performs well on all datasets?

Evaluate each of the six datasets using the six previously discovered top-performing configurations.



None of the tested configurations performed well across all datasets.

How this talk is structured

- Quick Overview of Traceability
- Open Challenges
- Towards More Intelligent Tracing Solutions
  - An Expert Traceability System
  - Acquiring Domain Knowledge
  - Configuring & Optimizing a Trace Engine
  - TiQi: Naturally Speaking
- Transition to Practice
- Closing Comments

TiQi

One of the primary barriers to Traceability in practice is the difficulty of **using** previously created trace links.

Not enough tools to construct trace queries.

Complex trace queries must often be created using SQL.

Subject: traceability query

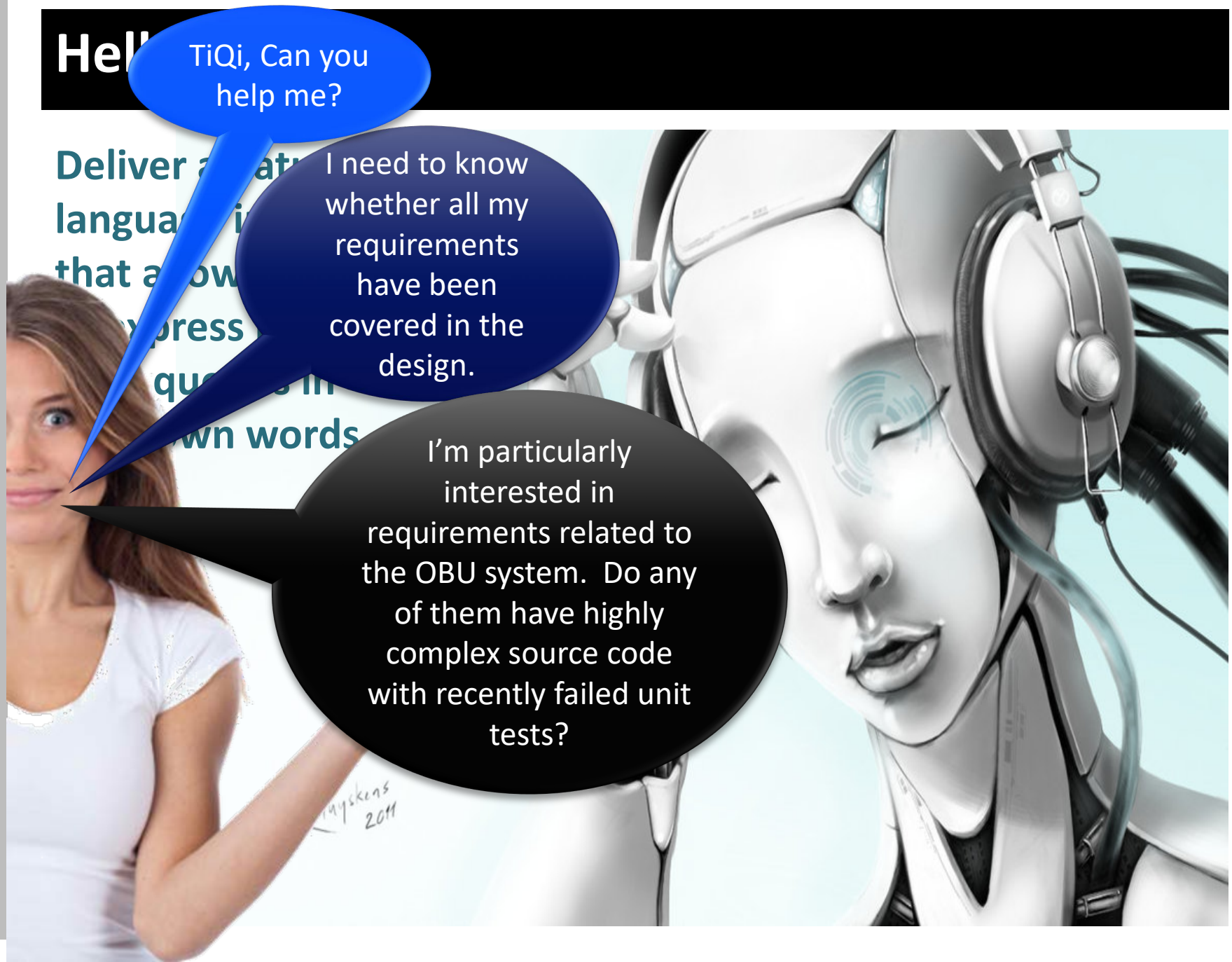
Sep 29, 2009 15:54:29 GMT

Hey all I need a super query writer...I have the following query:

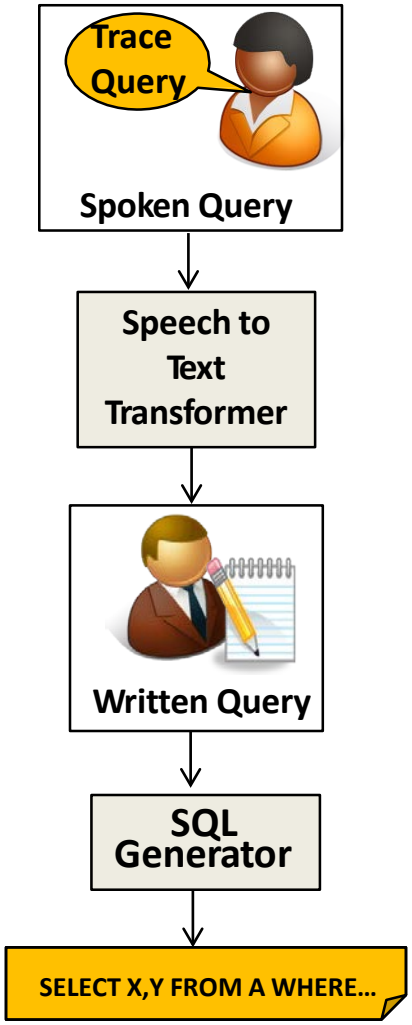
```
SELECT req.rq_user_17 AS "Requirement Type",REQ.RQ_REQ_ID as
"Requirement ID", REQ.RQ_USER_08 as "Requirement
Class",REQ.RQ_REQ_NAME as "Requirement Name" , count
(REQ_COVER.RC_ITEM_ID) AS "Coverage"
FROM REQ, req_cover
WHERE REQ.RQ_REQ_ID in (select REQ_COVER.RC_REQ_ID from
req_cover)
and req.rq_user_17='WS'
GROUP BY req.rq_user_17, REQ.RQ_REQ_ID,
REQ.RQ_USER_08,REQ.RQ_REQ_NAME
union
SELECT req.rq_user_17 AS "Requirement Type",REQ.RQ_REQ_ID as
"Requirement ID", REQ.RQ_USER_08 as "Requirement
Class",REQ.RQ_REQ_NAME as "Requirement Name" , count
(REQ.rq_attachment) AS "Coverage"
FROM REQ, req_cover
WHERE REQ.RQ_REQ_ID not in (select REQ_COVER.RC_REQ_ID from
req_cover)
and req.rq_user_17='WS'
GROUP BY req.rq_user_17, REQ.RQ_REQ_ID,
REQ.RQ_USER_08,REQ.RQ_REQ_NAME
order by req.rq_user_17
```

it gives me the aggregate number of testcase in req_cover and it puts this number to each requirement. But what I need is the aggregate number of testcase linked to each requirement. can anyone help?

Hel


TiQi, Can you help me?
Deliver a at
langua i
that a ow
press
qu s in
wn words
I need to know whether all my requirements have been covered in the design.
I'm particularly interested in requirements related to the OBU system. Do any of them have highly complex source code with recently failed unit tests?

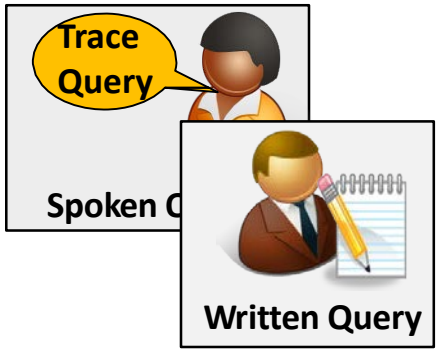
TiQi: As it is today



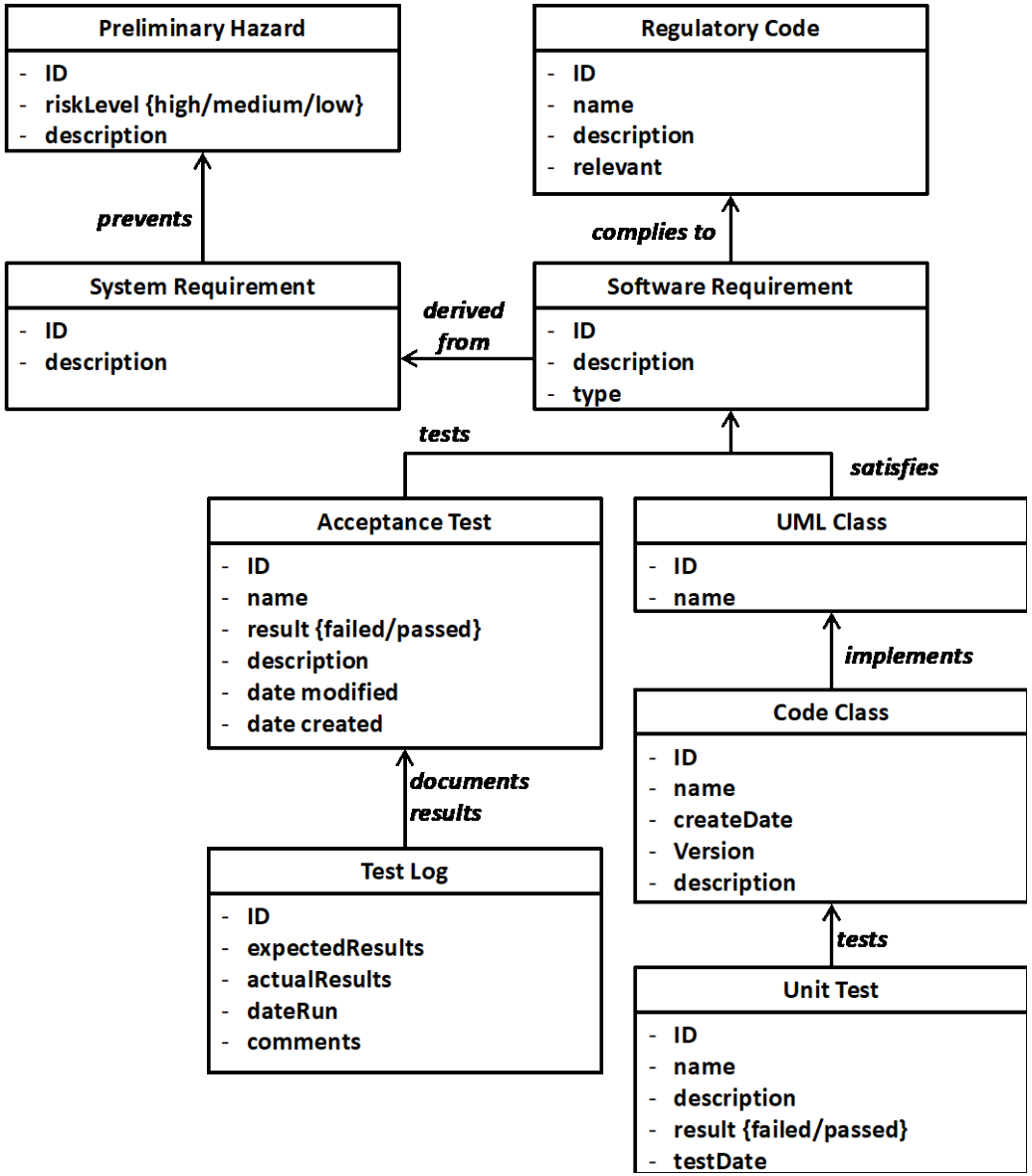
TiQi allows a user to express a trace query in spoken or written natural language and then transforms it into executable SQL statements.

- Are there any hazards with no identified contributing faults?
- List all tests which have recently failed and which are associated with high severity faults.
- List all requirements related to heat sensor faults.
- Is the system safe for use?
- What's up dude?

TiQi:

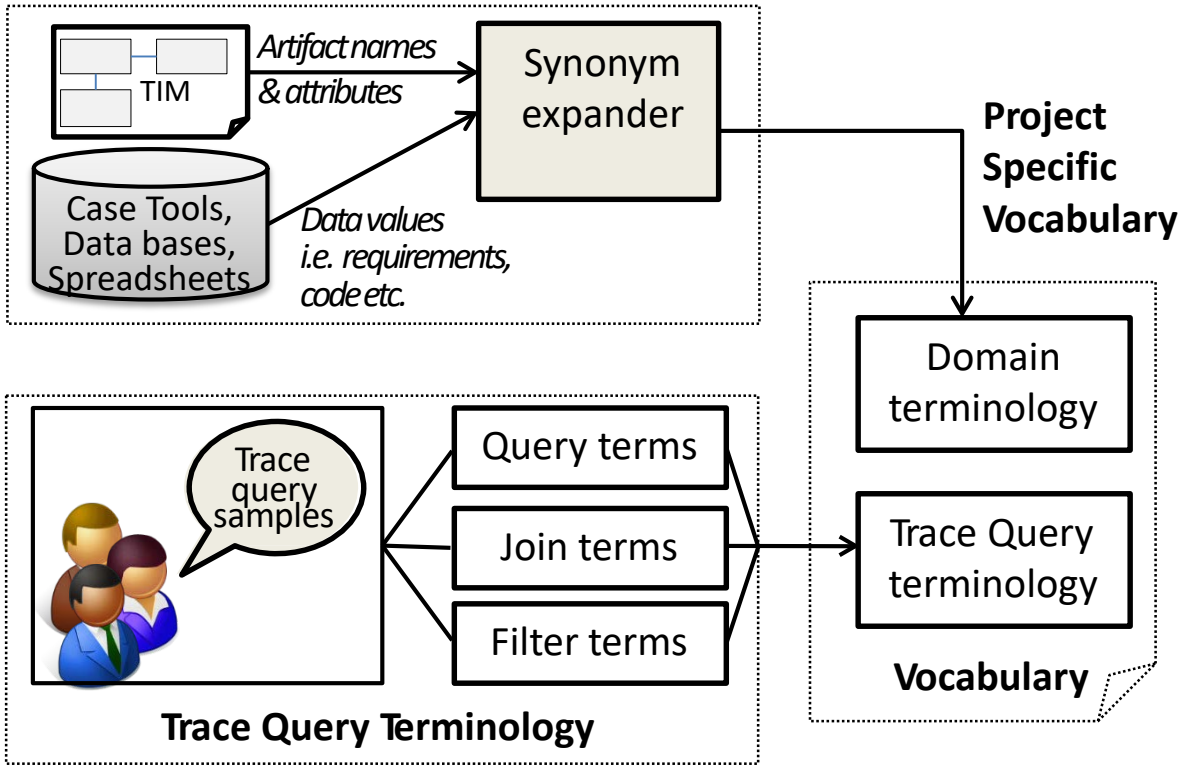


Users are prompted for queries by the Traceability Information Model. (TIM)



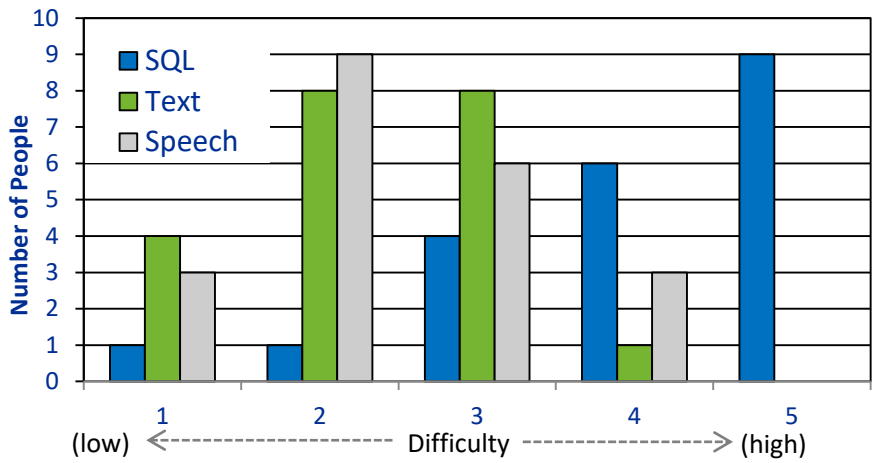
TiQi:

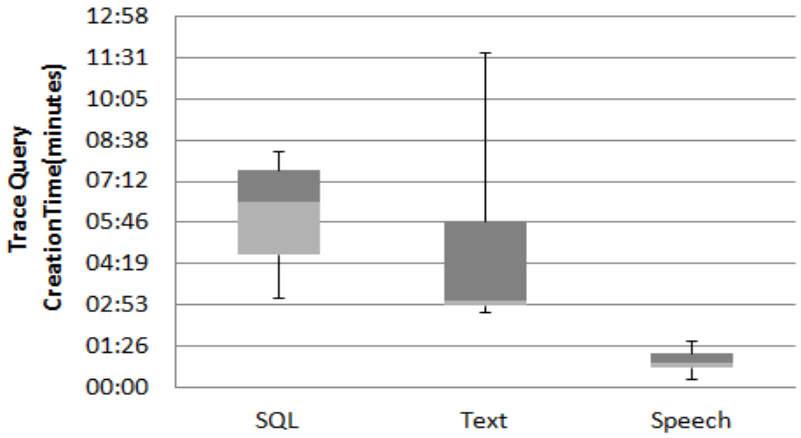
What kind of words and phrases do people use when they express natural language trace queries?



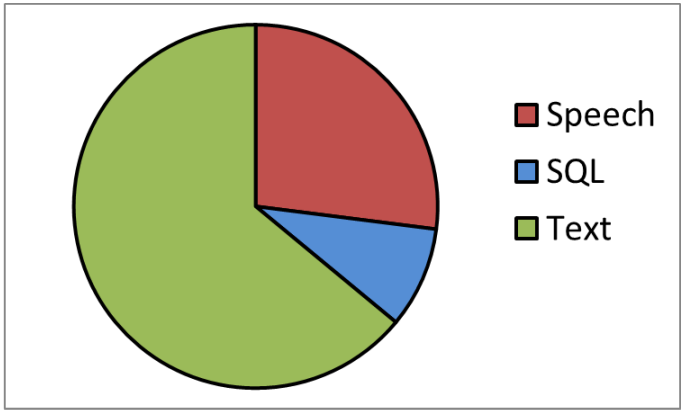
Domain specific vocabulary is extracted from the TIM and associated data, while traceability jargon is identified in advance and reused across projects.

Early Results

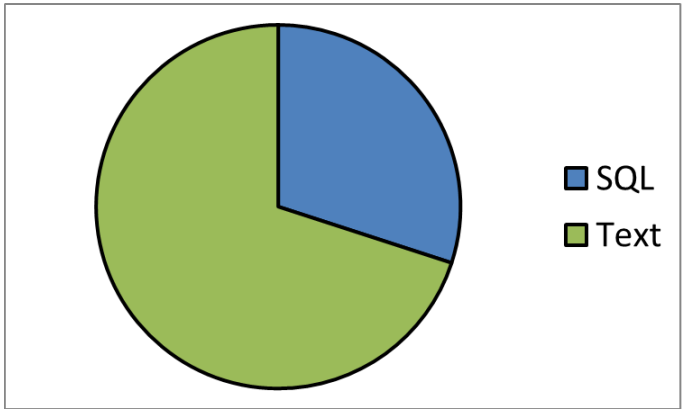




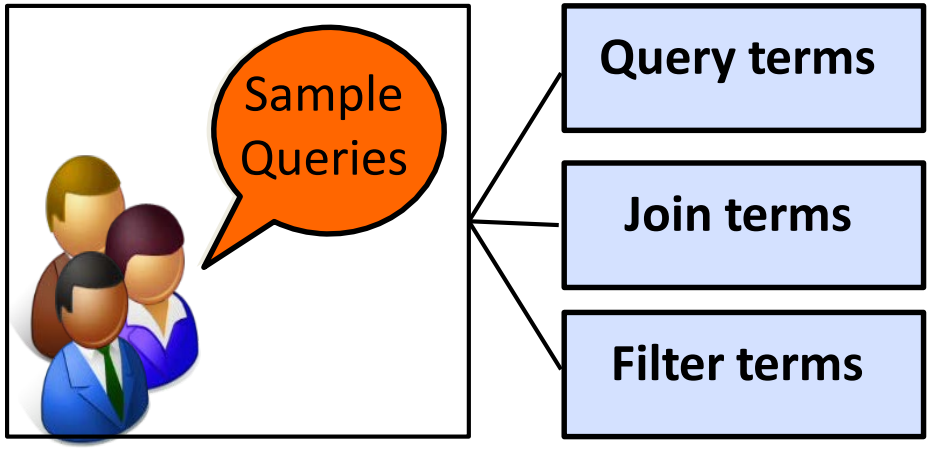
Traceability Experts



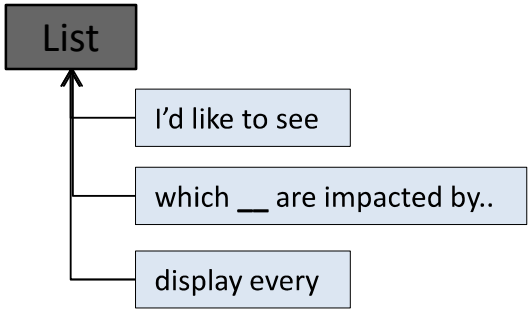
Non Experts

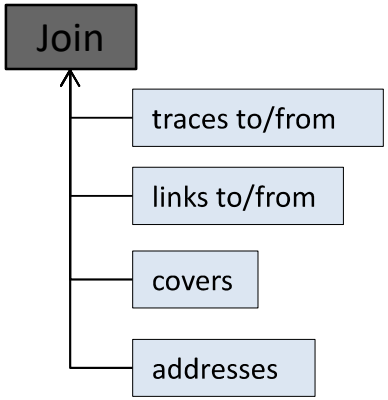


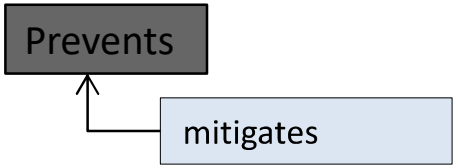
We constructed a vocabulary

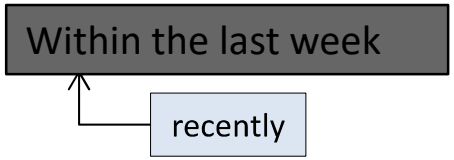


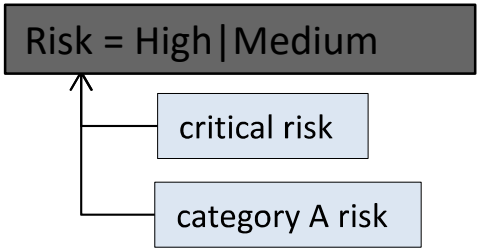
Trace Query Terminology

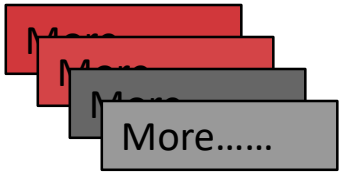




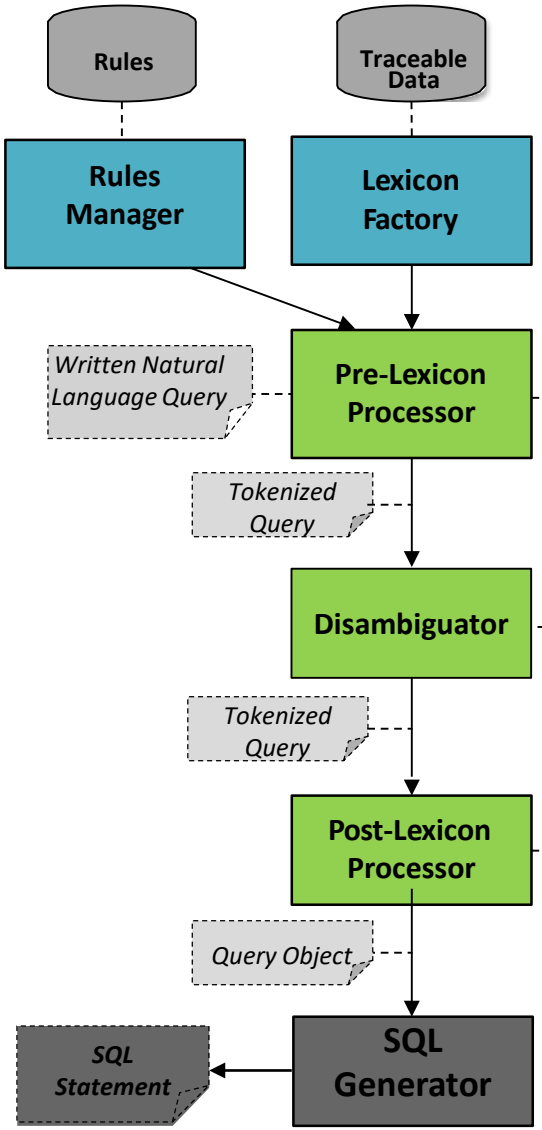








Current TiQi Flow



Pre-Lexicon Processor

Recognize synonyms
Recognize group-by
Recognize negation Number Identifiers
Preprocess Yes/No queries
Preprocess summation queries Stop word removal
Remove unprocessed protected words

Disambiguators:

Max-flow
Basic priority conflict resolver
Resolve table names
Resolve column names
Resolve data Value names

Post-Lexicon Processor

Group by
Pattern Check -cross-table aggregates
YesNoQuery Processor
Summation Query processor
Find selector
Comparison Operator Resolver
Negation processor
Combine DB Entities into Queries

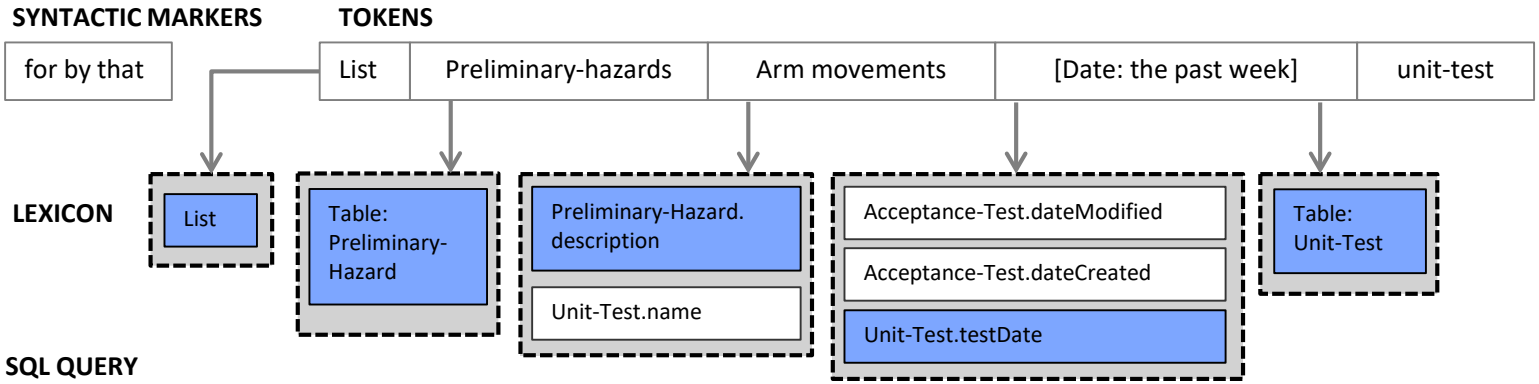
Transforming NL to SQL

QUERY

I'd like to see a list of all preliminary hazards for arm movements which are tested by recent unit tests.

PRE-PROCESSED QUERY

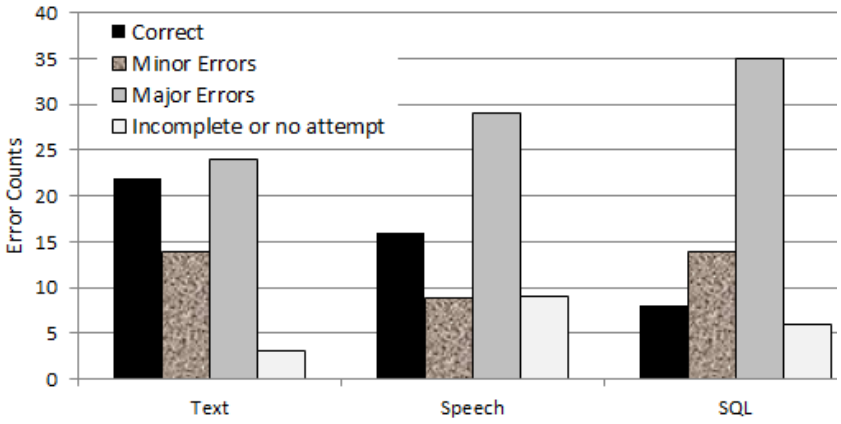
[List] preliminary-hazard for arm movement [Join] tested by [Date: the past week] unit-test.

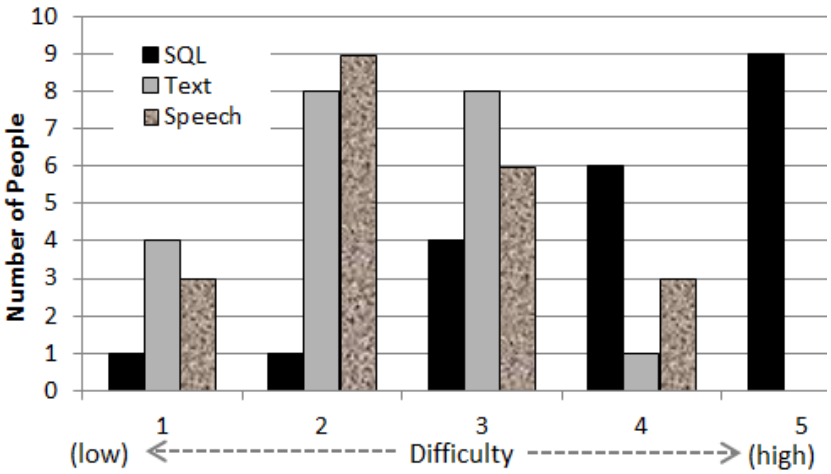


SQL QUERY

```
SELECT   `PreliminaryHazard`.*
FROM     `PreliminaryHazard`, `LINKSystemRequirement2PreliminaryHazard`, `SystemRequirement`,
         `LINKSoftwareRequirement2SystemRequirement`, `SoftwareRequirement`, `LINKUMLClass2SoftwareRequirement`,
         `UMLClass`, `LINKCodeClass2UMLClass`, `UMLCode`, `LINKUnitTest2CodeClass`, `UnitTest`
WHERE    `PreliminaryHazard`.`ID` = `LINKSystemRequirement2PreliminaryHazard`.`TargetID` AND
         `SystemRequirement`.`ID` =  `LINKSystemRequirement2PreliminaryHazard`.`SourceID` AND
         `SystemRequirementID`.`ID` = `LINKSoftwareRequirement2SystemRequirement`.`TargetID` AND
         `SoftwareRequirement`.`ID` = `LINKSoftwareRequirement2SystemRequirement`.`SourceID` AND `SoftwareRequirement`.`ID` =
         `LINKUMLClass2SoftwareRequirement`.`TargetID` AND `UMLClass`.`ID` = `LINKUMLClass2SoftwareRequirement`.`SourceID` AND
         `UMLClass`.`ID` = `LINKCodeClass2UMLClass`.`TargetID` AND `UMLCode`.`ID` = `LINKCodeClass2UMLClass`.`SourceID` AND
         `UMLCode`.`ID` = `LINKUnitTest2CodeClass`.`TargetID` AND `UnitTest`.`ID` = `LINKUnitTest2CodeClass`.`SourceID` AND
         `UnitTest`.`testDate` >= "03/01/2014" AND `PreliminaryHazard`.`Description` LIKE '%arm movement%';
```

TiQi: Experimental Results





All Queries	Correct	Incorrect	% Correct
Isolette	49	21	70.0%
Easy Clinic	25	15	62.5%

Supported Queries	Correct	Incorrect	% Correct
Isolette	49	7	87.5%
Easy Clinic	25	9	73.53%

How this talk is structured

- Quick Overview of Traceability
- Open Challenges
- Towards More Intelligent Tracing Solutions
  - An Expert Traceability System
  - Acquiring Domain Knowledge
  - Configuring & Optimizing a Trace Engine
  - TiQi: Naturally Speaking
- Transition to Practice
- Closing Comments

What Practitioners Say…

Isn't it some heavyweight thing?

I think it is a made-up problem

We need traceability for certification!

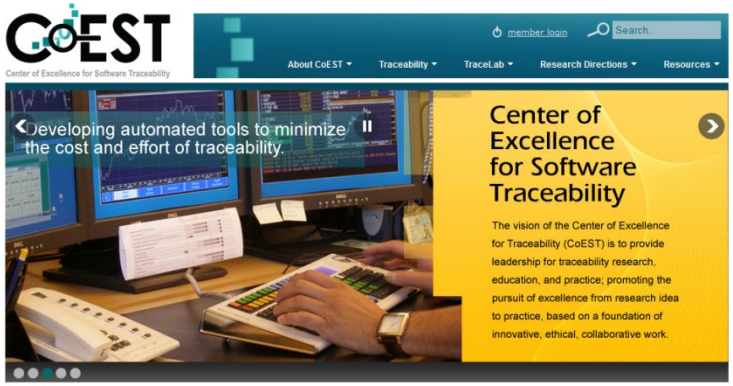
We often construct all of the links after development is over.

Fast paced agile/ business oriented projects

Slower-paced, carefully planned safety-critical projects

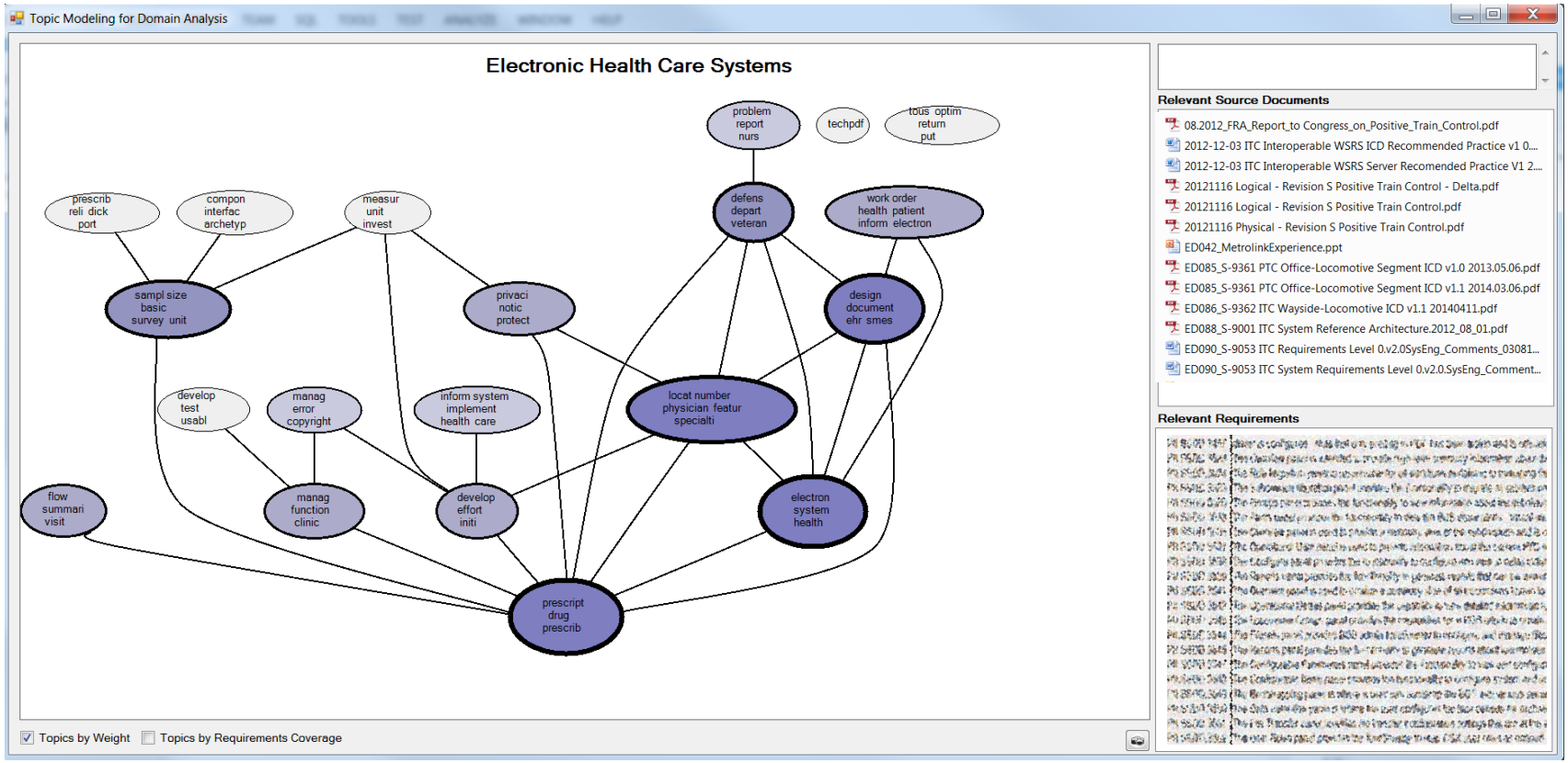
Benefits extend to non-safety critical projects

Support for safety-critical agile-projects and/or for continual certification



Much of our research is motivated by the needs of safety-critical systems.

What our Industry Collaborators Asked For



Automated traceability to fill in the gaps they've missed.

Click-through between requirements and "cloud" documents.

Queries that provide support for abstraction.

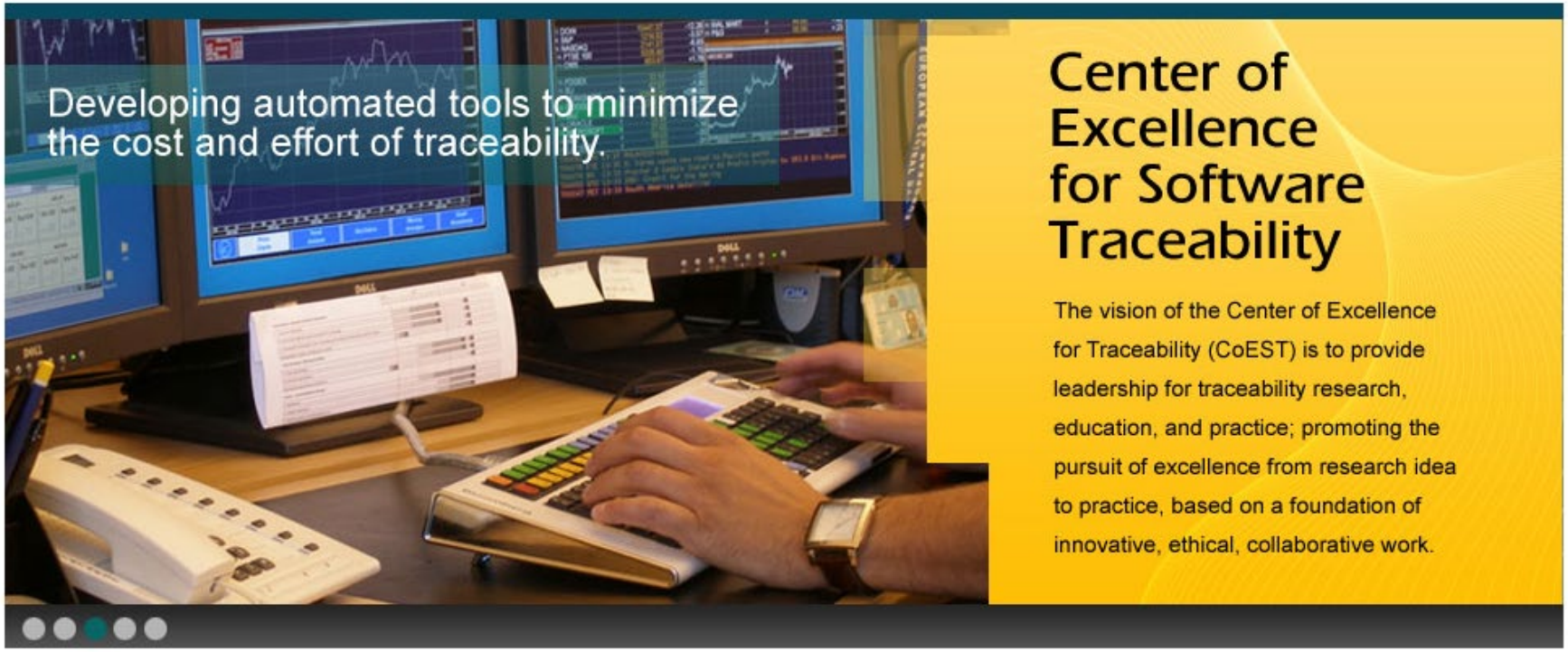
CoEST

Center of Excellence for Software Traceability

member login

Search...

About CoEST | Traceability | TraceLab | Research Directions | Resources



FOCUS: SAFETY-CRITICAL SOFTWARE

Strategic Traceability for Safety-Critical Projects

Patrick Mäder, Ilmenau Technical University

Paul L. Jones and Yi Zhang, US Food and Drug Administration

Jane Cleland-Huang, DePaul University

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

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

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

We present six practices for strategic traceability, derived from our own observations of effective traceability in industrial projects and supported by current literature.[7-9] We also identify nine recurring problems, each of which reduces the effectiveness of traceability verification efforts and increases the difficulty experienced by regulators in evaluating product safety. All the observations in this article are based on actual observations, but the illustrative examples are either fictitious or built on obfuscated data.

Effective Practices for Tracing in Safety-Critical Projects

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

Tackle cutting edge problems in software traceability.

Build a supportive community of researchers.

Community Infrastructure

TraceLab now released on GitHub

Standard VSM : C:\p4root\RELab\dev\Data\challenge1_HIPPA\StandardVSM.gml

Components Library

- Algorithms
- ControlLogic
  - Empty decision
- IO
  - Exporters
  - Importers
- Preprocessors
  - Cleanup Preprocessor
  - English Porter Stemmer
  - Java Porter Stemmer
  - Preprocessor
  - Splitter
  - Stopwords Remover
- Results
  - Computation
    - Results Loader
    - Results Metric Computation
  - Display
- Uncategorized
- Utilities

Workspace View

Name	Type	Val
answerMatrix	TraceLabSDI	Trac
originalSourceArtifacts	TraceLabSDI	Trac
originalTargetArtifacts	TraceLabSDI	Trac
sourceArtifacts	TraceLabSDI	Trac
targetArtifacts	TraceLabSDI	Trac
dictionaryIndex	TraceLabSDI	Trac

Start → Target Artifacts Importer and Preprocessor; Source Artifacts Importer and Preprocessor; AnswerMatrixImporter

TFIDF Dictionary Index Builder → Tracer Component → Results Metric Computation → Results Charts → End

1.00x

1:1

Fill

TFIDF Dictionary Index Builder

Input/Output

Input	Mapped to	Type
listOfArtifacts	targetArtifacts	TraceLabSDK.Types.TLArtifactsC

Output	Output as	Type
dictionaryIndex	dictionaryIndex	TraceLabSDK.Types.TLDictionary

Configuration

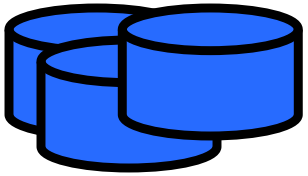
Component Info

Output

Global Log Levels

Severity	Source	Message
Trace	Tracer Component	Completed component TracerComponent
Trace	Results Metric Computation	Start component ResultsMetricsComponent
Trace	Results Metric Computation	Completed component ResultsMetricsComponent

Status: Experiment done!



Community datasets available.

Community wiki.

Effective Software & Systems Traceability

Distinguished Lecture
University of Luxembourg
Friday, March 20th

Professor Jane Cleland-Huang
Center of Excellence for Software Traceability
DePaul University

DePaul University

CoEST
Center of Excellence for Software Traceability

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