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*

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

Regulatory Codes

System/sub-system level requirements

Satisfies relevant codes

Use Traceability to demonstrate compliance to regulatory codes.

Tests

Validates

Realizes

Code

Test cases (unit, integration, acceptance)
Creating Trace Links

Current practice is primarily manual in nature

Hazard H2: Moving the patient’s arm at excessive velocity.

Fault F2.1: Velocity sensors fail to sense excessive velocity.

Fault F2.2: Configuration component fails to update correct velocity constraints.

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.

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.

Fault F2.1:
Velocity sensors fail to sense excessive velocity.

Fault F2.2:
Configuration component fails to update correct velocity constraints.

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.

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

Current practice is primarily manual in nature
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

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.

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

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

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 Poirer 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 regulatory codes

1. Split regulatory codes into segments.
2. Split requirements into segments.
3. Use Poirer to generate traceability links between individual regulations and requirements.
4. Compute impact metrics & score candidate answer set from evaluated answer set.
5. Determine applicability of each set of traceability links to regulatory segments.

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:

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**.
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?

Three perspectives
A Process Perspective

Creating
- Trace envisaged
- Trace created [use requested]
- Creation feedback
- Trace creation planned [create directed]
- Trace created [elements change]
- Maintenance feedback

Planning and Managing Traceability Strategy
- Requirements for traceability changed
- Trace maintenance planned
- Traceability required
- Trace maintenance [use requested]
- Trace maintenance required [elements change]
- Trace retired

Using
- Use feedback
- Traceability required
- Project archived

Maintaining
- Trace retired

Percentage of papers

Plan & Mgm
Creating
Maintaining
Using

Usage by Subcategory %
- Tools
- Human in the Loop
- General
A Technical Perspective

1. Artifact and trace access layer or repository
   - Artifacts
   - Trace Matrices
   - Core assets
   - Variabilities

2. Traceability Information Model
   - ISO 26262
   - DO-178B
   - Safety-critical standards
   - Guideline-level TIM
   - Product/Productline TIMs

3. Trace Query Layer
   - Safety Hazard
     - Hazard
   - FTA Min cut set
     - faultSet
   - System Requirement
     - id
     - description

4. Workflows

5. Trace engine
   - Trace retrieval and capture engines

6. User Interactions
   - Modify TIM (define artifact and link types)
   - Create, delete, and modify artifacts
   - Safety analyses

Percentage of Papers by Type

- Unassigned
- Planning
- Storage
- Workflows
- Queries
- Trace Algorithms
- Usage
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:
RD-1.2 Evaluate traceability by stakeholders.

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.

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.

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 maintenance.

Jane Cleland-Huang, Orlena Gotel, Jane Huffman Hayes, Patrick Mäder, Andrea Zisman:
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.
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.

Broadcast is similar to transmit.

Automobile controller is part of the automobile.

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

How do I trace this?
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

A1: Any critical failure during the Disengaged Mode will force the OBM to enter the Failed 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
- **Action**: enter
- **Semantic Grps**: Motive
- **Theme**: obm
- **Location**: fail mode
Thematic Roles

**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"

In Linguistics there are hundreds of thematic roles. These are the ones that we have found to be common in the transportation domain.
**Heuristic # 1: Basic Match**

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

A2: The Highway Wayside Segment shall transmit information to the Automobile Segment in the form of WSMs.

- **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

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

<table>
<thead>
<tr>
<th>24 Semantic Groups</th>
<th>Heuristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Administrative</td>
<td>300 pairs possible</td>
</tr>
<tr>
<td>2. Affirmative</td>
<td>27 pairs defined</td>
</tr>
<tr>
<td>3. Calculative</td>
<td>Basic Match: 19</td>
</tr>
<tr>
<td>4. Causative</td>
<td>Motive-Motive: 5</td>
</tr>
<tr>
<td>5. Cooperative</td>
<td>Administrative-Inceptive: 4</td>
</tr>
<tr>
<td>6. Descriptive</td>
<td>Motive-Permissive: 3</td>
</tr>
<tr>
<td>7. Effective</td>
<td>Transmissive-Receptive: 3</td>
</tr>
<tr>
<td>8. Enforcive</td>
<td>Calculative-Enforcive: 2</td>
</tr>
<tr>
<td>9. Evasive</td>
<td>Transmissive-Transmissive: 2</td>
</tr>
<tr>
<td>10. Inceptive</td>
<td>Permissive-Permissive: 2</td>
</tr>
<tr>
<td>11. Inclusive</td>
<td>Regulative-Inclusive: 1</td>
</tr>
<tr>
<td>12. Inspective</td>
<td>Transmissive-Descriptive: 1</td>
</tr>
<tr>
<td>13. Interactive</td>
<td>Receptive-Enforcive: 1</td>
</tr>
<tr>
<td>14. Motive</td>
<td></td>
</tr>
<tr>
<td>15. Necessitative</td>
<td></td>
</tr>
<tr>
<td>16. Permissive</td>
<td></td>
</tr>
<tr>
<td>17. Preservative</td>
<td></td>
</tr>
<tr>
<td>18. Receptive</td>
<td></td>
</tr>
<tr>
<td>19. Regulative</td>
<td></td>
</tr>
<tr>
<td>20. Reparative</td>
<td></td>
</tr>
<tr>
<td>21. Submissive</td>
<td></td>
</tr>
<tr>
<td>22. Supportive</td>
<td></td>
</tr>
<tr>
<td>23. Transgressive</td>
<td></td>
</tr>
<tr>
<td>24. Transmissive</td>
<td></td>
</tr>
</tbody>
</table>

**Question:** How generalizable are these rules across domains?
Domain Centric Intelligent Traceability (DoCIT)

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

![Instance of an Action Unit]

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

3. 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.
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
The **Highway Wayside Segment** shall transmit information to the automobile controller in the form of **WSMs**.

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.

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

Broadcast is similar to transmit.

Automobile controller is part of the automobile

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

WSM and wayside status message are equivalent.

How do I trace this?
## How many domain terms?

<table>
<thead>
<tr>
<th></th>
<th>Single Words</th>
<th>Phrases</th>
<th>Overlap</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Words</td>
</tr>
<tr>
<td><strong>DVC</strong></td>
<td>13,960</td>
<td>104,026</td>
<td>23%</td>
</tr>
<tr>
<td></td>
<td>1,784</td>
<td>4,173</td>
<td></td>
</tr>
<tr>
<td><strong>EHR</strong></td>
<td>3,118</td>
<td>971</td>
<td>16%</td>
</tr>
<tr>
<td></td>
<td>621</td>
<td>89</td>
<td></td>
</tr>
<tr>
<td><strong>MIP</strong></td>
<td>3,409</td>
<td>18,164</td>
<td>98%</td>
</tr>
<tr>
<td></td>
<td>3,632</td>
<td>10,664</td>
<td></td>
</tr>
</tbody>
</table>

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

**Domain Documents**

- Project artifacts e.g. requirements & design specs.

**Extraction Tools**

- Lexical-Syntactic Pattern Extractor
- Association Rule Miner
- Topic Modeler
- Semantic Relatedness Computer

**Trace Matrix**

- Noun-phrase pair extractor

**Profile of candidate facts**

- Facts for training

**Candidate Fact Classifier**

- New candidate facts

**Evaluation**

- Evaluate facts to train the classifier

- Evaluate candidate facts classified as true. Accepted CFs are saved to the ontology.
## Training the Classifier

<table>
<thead>
<tr>
<th>Candidate Fact</th>
<th>Knowledge Source</th>
<th>ARM Link Cnt</th>
<th>Cos Sim</th>
<th>LDA Sim</th>
<th>SR Sco</th>
<th>Feat. Include</th>
<th>Sim St</th>
<th>Sim HN</th>
<th>Rev</th>
<th>Src Len</th>
<th>Tgt Len</th>
<th>Eval</th>
</tr>
</thead>
<tbody>
<tr>
<td>pca pump</td>
<td>superClass</td>
<td>ARM</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>4</td>
<td>0.24</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>medication</td>
<td>hasPart</td>
<td>LDA</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>label</td>
<td></td>
<td>SR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>user</td>
<td>superClass</td>
<td>LSP</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>medication</td>
<td>superClass</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>electrical</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>system</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pump overload</td>
<td>subClass</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>alarm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>drug library</td>
<td>superClass</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>soft limit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>medication</td>
<td>superClass</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>safety</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Class</th>
<th>Random Forest</th>
<th>Naïve Bayes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>TP</td>
<td>FP</td>
</tr>
<tr>
<td>DVC</td>
<td>Invalid</td>
<td>0.66</td>
<td>0.31</td>
</tr>
<tr>
<td></td>
<td>Valid</td>
<td>0.68</td>
<td>0.33</td>
</tr>
<tr>
<td>EHR</td>
<td>Invalid</td>
<td>0.75</td>
<td>0.26</td>
</tr>
<tr>
<td></td>
<td>Valid</td>
<td>0.73</td>
<td>0.24</td>
</tr>
<tr>
<td>MIP</td>
<td>Invalid</td>
<td>0.77</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>Valid</td>
<td>0.79</td>
<td>0.22</td>
</tr>
</tbody>
</table>
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
..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*

**Link Heuristics**

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

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

More permutations than the number of nanoseconds in the universe.

Currently exploring multi-objective solution using NGSA-II.
We missed the ICSE deadline last Fall!

Or this many centuries....

seven hundred sixty-four octovigintillion, eight hundred eighty-one septenvigintillion, nine hundred forty-eight sexvigintillion, one hundred ninety-five quinvigintillion, two hundred six quattuorvigintillion, three hundred fifty-three trevigintillion, four hundred eighty-one duovigintillion, sixty-four unvigintillion, three hundred eighty-five vigintillion, nine hundred seventeen novemdecillion, two hundred eighty-eight octodecillion, twenty-eight septendecillion, two hundred thirty-three sexdecillion, one hundred forty-seven quindecillion, five hundred forty-one quattuordecillion, five hundred seventy-seven tredecillion, eight hundred sixty-four duodecillion, nine hundred forty-eight undecillion, two hundred nine decillion, nine hundred forty-nine nonillion, nine hundred fifty octillion, six hundred sixteen septillion, three hundred ninety-five sextillion, nine hundred twenty-seven quintillion, eight hundred thirteen quadrillion, seven hundred eighty-eight trillion, eight hundred twelve billion, six hundred twenty-nine million, three hundred ninety thousand, three hundred eighty-seven.

Will ICSE wait for my paper?
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..

How should the Trace Engine be configured in order to return the most accurate trace links for my data and my project environment?

Important for industrial application and for creating research baselines.
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
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

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

\[
MAP = \frac{\sum_{q=1}^{Q} \text{AveragePrecision}(q)}{Q}
\]

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

**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?

---

**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
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
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.
Deliver a natural language interface that allows users to express their queries in their own words.

Hello TiQi!

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, Can you help me?
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?
Users are prompted for queries by the Traceability Information Model (TIM).
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

Sample Queries

Query terms
- Prevents
- mitigates

Join terms
- Within the last week
- recently

Filter terms
- Risk = High | Medium
  - critical risk
  - category A risk

List
- I’d like to see
- which __ are impacted by..
- display every

Join
- traces to/from
- links to/from
- covers
- addresses

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

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

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`.'TargetID' AND
`UnitTest`.'ID' = `LINKCodeClass2UMLClass`.'SourceID' AND
`UnitTest`.'testDate' >= "03/01/2014" AND `PreliminaryHazard`.'Description' LIKE '%arm movement%';
TiQi: Experimental Results

### All Queries

<table>
<thead>
<tr>
<th></th>
<th>Correct</th>
<th>Incorrect</th>
<th>% Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Isolette</td>
<td>49</td>
<td>21</td>
<td>70.0%</td>
</tr>
<tr>
<td>Easy Clinic</td>
<td>25</td>
<td>15</td>
<td>62.5%</td>
</tr>
</tbody>
</table>

### Supported Queries

<table>
<thead>
<tr>
<th></th>
<th>Correct</th>
<th>Incorrect</th>
<th>% Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Isolette</td>
<td>49</td>
<td>7</td>
<td>87.5%</td>
</tr>
<tr>
<td>Easy Clinic</td>
<td>25</td>
<td>9</td>
<td>73.53%</td>
</tr>
</tbody>
</table>
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
Fast paced agile/business oriented projects

Isn’t it some heavyweight thing?

I think it is a made-up problem

Slower-paced, carefully planned safety-critical projects

We need traceability for certification!

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

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.
Tackle cutting edge problems in software traceability.

Developing automated tools to minimize the cost and effort of traceability.

Center of Excellence for Software Traceability

The vision of the Center of Excellence for Traceability (CoEST) is to provide leadership for traceability research, education, and practice; promoting the pursuit of excellence from research idea to practice, based on a foundation of innovative, ethical, collaborative work.

Build a supportive community of researchers.
Community Infrastructure

TraceLab now released on GitHub

Community datasets available.

Community wiki.
Distinguished Lecture
University of Luxembourg
Friday, March 20th

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

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