Traceability Beyond Source Code: An Elusive Target?

Lionel Briand
Interdisciplinary Centre for Security, Reliability and Trust (SnT)
University of Luxembourg

Rennes, December 3, 2015
Acknowledgements

• Shiva Nejati
• Mehrdad Sabetzadeh
• Fabrizio Pastore
• Chetan Arora
• Chunhui Wang
• Ghanem Soltana
• Davide Falessi
Outline

• Introduction

• Overview

• Examples from industrial research projects

• Reflections and conclusions
Traceability

• The ability to follow the life of software artifacts, in both a backward and forward direction, e.g., requirements, design decisions, test cases.

• Requirements traceability: Trace a requirement from its emergence to its fulfillment.

• Motivations:
  • Understand rationale
  • Assess impact of change
  • Certification, auditing, compliance with standards
Motivations

• Traceability research is source-code-centric

• Certification (safety, privacy …)

• Change management: Impact analysis, design rationale, regression testing …

• Change management is a key challenge to certification

• Traceability analysis is a system-level activity
Challenges

- Establishing and maintaining traces is typically expensive
- Automation, in most cases, does not provide the level of accuracy required
- The benefits of exploiting traces are still unclear in many contexts
- Highly contextualized: A great deal of variation in development contexts entails a great deal of variation in traceability solutions
- Targeted analysis of traces drives traceability solutions
Requirements

- Hundreds or thousands of them
- Higher-level requirements (usually from customers) decomposed into lower-level ones (analysts)
- Some more critical than others
- Constantly changing and evolving: A stronger argument for the economic benefits of traceability
Modeling

- In many application domains where traceability is required, system and software modeling is a rising practice

- Provisions in standards lead to modeling

- IEC 61508 (meta-standard), DO-178B (Avionics), EN50129 (Railways), ISO 26262 (Automotive)

- UML, SysML, Simulink, …
Economic Decision

• Not just about trace “accuracy” …

• Economic trade-off
  
  • Cost: Establishing and maintaining traces
  
  • Benefit: More accurate decisions, decrease in human effort

• Decision science

• Makes it hard to study, out of context, as it determines effort and benefits
Overview
Traceability at a Glance

- Regulations
- Requirements
- Test Cases
- Archi. & Design
- Source Code
- Bulk of Research
Requirements-Source Code

- Natural language
- Hundreds or thousands of traces
- Information Retrieval & Natural Language Processing
- Coding conventions
- Level of granularity?
- Minimum accuracy for ensuring practicality? Few human studies …
Traceability at a Glance

- Regulations
- Requirements
- Test Cases
- Change Impact
- Archi. & Design
- Source Code
Requirements

- Mostly natural language
- Sometimes structured (template)
- Hundreds of traces
- Domain terminology, concepts, and their relationships are key to discovering traces among requirements
- Syntactic and semantic similarity measures
Traceability at a Glance

- Regulations
- Requirements
- Test Cases
- Archi. & Design
- Source Code

Compliance with laws, regulations, standards
Standards-Requirements

- Many standards, laws, and regulations
- They must be interpreted in context
- Compliance must be ensured
- Critical systems: Risks and hazards
- Requirements as mitigations
- Subjectivity, residual risks
Traceability at a Glance

- Regulations
- Requirements
- Test Cases

Certification, change management

Archi. & Design
Source Code
Requirements-Design

- Capture the rationale of design decisions
- Support evolution, avoid violating essential design decisions
- Useful for impact analysis based on traces
- What is a rationale? Level of granularity?
- Design representation?
Traceability at a Glance
Requirements-Test Cases

- Requirements “coverage” required by standards
- Normally many test cases per requirement
- Thousands of traces
- Regression testing
- Precise impact analysis requires explicit test strategy and rationale
  - How were test cases derived from requirements?
  - Representation of requirements matters
Traceability at a Glance

- Regulations
- Requirements
- Test Cases
- Archi. & Design
- Source Code

Impact analysis, design conformance
Design-Source Code

- Ideally, code should be generated from design models, e.g., controllers with Simulink
- This would lead to “free” traceability
- In practice, not always that simple …
Example Projects
Requirements

[RE 2015, TSE 2015, ESEM 2014, ESEM 2013]

Case A
- 160 Requirements
- 9 change scenarios

Case B
- 72 Requirements
- 5 change scenarios

SES
your satellite company

novay
Example

- **R1**: The mission operation controller shall transmit satellite status reports to the user help desk.
- **R2**: The satellite management system shall provide users with the ability to transfer maintenance and service plans to the user help desk.
- **R3**: The mission operation controller shall transmit any detected anomalies with the user help desk.
Example

- **R1:** The mission operation controller shall transmit satellite status reports to the user help desk document repository.
- **R2:** The satellite management system shall provide users with the ability to transfer maintenance and service plans to the user help desk.
- **R3:** The mission operation controller shall transmit any detected anomalies with the user help desk.
Challenge#1 - Capture Changes Precisely

- R1: The mission operation controller shall transmit satellite status reports to the user help desk document repository.
- R2: The satellite management system shall provide users with the ability to transfer maintenance and service plans to the user help desk.
- R3: The mission operation controller shall transmit any detected anomalies with the user help desk.
Challenge#2 - Capture Change Rationale

- R1: The mission operation controller shall transmit satellite status reports to the **user help desk** document repository.
- R2: The satellite management system shall provide users with the ability to transfer maintenance and service plans to the **user help desk**.
- R3: The mission operation controller shall transmit any detected anomalies with the **user help desk**.
Challenge#2 - Change Rationale

- R1: The mission operation controller shall transmit satellite status reports to the user help desk document repository.
- R2: The satellite management system shall provide users with the ability to transfer maintenance and service plans to the user help desk.
- R3: The mission operation controller shall transmit any detected anomalies with the user help desk.

Rationales:

R1: We want to globally rename “user help desk”
R2: Avoid communication between “mission operation controller” and “user help desk”
R3: We no longer want to “transmit satellite status reports” to “user help desk” but instead to “user document repository”
Solution Characteristics

• Accounts for the phrasal structure of requirements

  The mission operation controller shall transmit satellite status reports to the user help desk document repository.

  user help desk, Deleted

  user document repository, Added

• Account for semantically-related phrases that are not exact matches and close syntactic variations
Rationale:
Avoid communication between mission operation controller and user help desk.

Propagation condition:
mission operation controller AND user help desk AND transmit
RQ1 - Which similarity measures are best suited to our approach?

- Experimented with 10 syntactic, 9 semantic measures, and all their pairwise combinations (109 combinations)
RQ2 - How should analysts use the sorted requirements list produced by our approach?
RQ3 - How effective is our approach?

- **Extra requirements traversed**
- **Case-A** between 1%-7%
- **Case-B** between 6%-8% except one case
- **Number of impacted requirements missed:** 1 out of 106
Requirements-Design


Requirements  Archi. & Design
Context

- Context: Certification of safety-critical monitoring applications (fire and gas detection and emergency and process shutdown) in oil & gas industry
- Certification: Assessing and discussing software requirements, design/architecture and implementation documents
- Typically, many meetings taking place over 6 to 18 months
Observations

- Analyzed 66 distinct certification issues:
  - Issues collected through observing certification meetings at different suppliers of maritime and energy systems
  - Meetings focused on requirements, architecture, and design documents

![Bar chart showing minor, requirement problems, and poor traceability and structure issues, with an annotation indicating expensive to fix.]
Research Objective

• Developing a model-based traceability methodology

• Generate a sound and yet minimal design slice for a given safety requirement, to support safety inspections

• Slices constructed based on traceability links established between safety requirements and design
Research Approach

Traceability Methodology

to relate safety requirements to design

Slicing Algorithm

to extract a design slice relevant to a given safety requirement

Model Driven Engineering (MDE) is the enabler
Modeling

- System Modeling Language (SysML)
  - A subset of UML extended with system engineering diagrams
  - A standard for system engineering
  - Preliminary support for requirement analysis and built-in traceability mechanism
Is SysML enough?

- Do we have proper guidelines for establishing traceability links between requirements and design?
  - **SysML is only a notation and needs a methodology**

- Are the built-in SysML traceability links capable of addressing certification traceability issues?
  - New traceability links: Source and assumptions of sys. safety reqs.
  - We specialized the semantics of existing ones: Refine, decompose, derive …
  - Explicit and implicit links
Research Approach

Traceability Methodology

to relate safety Requirements to design

Slicing Algorithm

to extract a design slice relevant to a given safety requirement
Modeling Methodology

(I) System Requirement Specification
(1) System Context Diagram
(2) System-Level Requirements
(3) Top-Level Use Cases

(II) System Design
Structural Models
(4) System Decomposition
(5) Communication Interfaces

Behavioural Models
(6) Intra-Block Communication
(7) Inter-Block Communication

Establish Traceability
(8) Decompose System-Level Requirements
(9) Re-express Requirements in terms of Block Operations and States
Traceability Information Model

Structural relations

- OCL Constraint
- Parametrics

- Assumption

- System Context
- Environment Block
- System Block

- Block-Level Requirement
- Mapping

- Block State
- Block Operation

- Activity Partition
- Activity Node
- Activity Edge

Trace
1..* allocate
Traceability Information Model

[Diagram showing traceability links and relationships between OCL Constraint, Assumption, System Context, Environment Block, System Block, Block-Level Requirement, Mapping, Block State, Block Operation, Activity Node, Activity Partition, Activity Edge, and Mapping]
Traceability Information Model

Implicit Links
Requirement to Design Traceability

Avoidance of falling metal blanks

The feed belt conveys a blank to table if the table is in load position

feedbelt.feed_table() causes "feed belt conveys a blank to table"

trace

After executing table.go_load_position(), "table is in load position"

trace

- Mappings are documenting the design rationale!
- Implications relations between phrases and block states and operations
Research Approach

Traceability Methodology

to relate safety Requirements to design

Slicing Algorithm

to extract a design slice relevant to a given safety requirement
Design Slicing

Avoidance of falling metal blanks

The feed belt conveys a blank to table if the table is in load position

*feedbelt.feed_table() causes "feed belt conveys a blank to table"

After executing *table.go_load_position(), "table is in load position"

**FeedBelt**
- running: boolean
- blankAtEnd: boolean
- initialize()
  - add_blank()
  - feed_table()

**Table**
- pos: Table_Position
- loaded: boolean
- initialize()
  - go_load_position()
  - go_unload_position()
Establish traceability. Through the activities under this task, we establish traceability links from the system-level requirements down to the design diagrams adapting and using the SysML traceability links. The traceability links specify which parts of the design contribute to the satisfaction of each requirement. This part of the methodology is the main extension compared to the existing methodologies [13,14,11,10].

Fig. 6. A fragment of design diagrams for PCS: (a) block definition diagram, and (b) an activity diagram consisting of two activity partitions where each corresponds to a block in (a).

Fig. 7. An fragment of the internal block diagram for PCS representing interaction points between software blocks and their interfaces.

S. Nejati et al. / Information and Software Technology 54 (2012) 569–590
Slices

Avoidance of falling metal blanks

decompose

The feed belt conveys a blank to table if the table is in load position

<table>
<thead>
<tr>
<th>feedbelt: FeedBelt</th>
<th>table: Table</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image.png" alt="Diagram" /></td>
<td><img src="image.png" alt="Diagram" /></td>
</tr>
</tbody>
</table>
Slicing Algorithm

• If a requirement holds over a design slice, it should also hold over the original design (soundness)
  - Proven analytically (formal proof)

• If a requirement holds over the original design, then the design slice created for that requirement should conclusively satisfy that requirement (completeness)
  - Evaluated empirically (Case studies and experiments)
Tool Support

Customized traceability links
Case Study: SW/HW Interfaces

Communicates commands and data between control modules and hardware

Goal: Practical guidelines to:
1. Capture the concurrent design of interfaces
2. Reduce the number and criticality of certification issues related to interfaces
Results

- Created design models with traceability to requirements
  - One context diagram (BDD), One architecture diagram (IBD), One detailed structure diagram (BDD), One activity decomposition diagram (BDD), One overall activity diagram, 19 detailed activity diagrams
  - Created 65 traceability links for 30 safety-relevant requirements
  - Modeling effort was approximately 40 person-hours
- Model Slicing
  - Extracted 34 block slices and 31 activity slices
  - Slicing reduced the number of block operations by 70% and the number of activity nodes by 50%
Controlled Experiment

• Question: Do safety slices help find design issues?

• Conducted in a laboratory setting with master students

• Overall design

  • Seeded faults into the design
    • Incorrect behavior and structure

  • Divided the subjects into two groups
    • One group gets the design without slices
    • One group gets the design plus the relevant slices
Experiment Results

- Slices show strong benefits in terms of:
  - Increasing the correctness of inspection decisions
  - Decreasing the proportion of uncertain decisions
  - Reducing the effort of inspections
Recent Similar Experiment

- Do developers benefit from requirements traceability when evolving and maintaining a software system? Patrick Mäder, Alexander Egyed
- Empirical Software Engineering (Springer), 2015
- Focus on program comprehension and maintenance
- Tasks with and without traceability
- Traceability led to 24% speed improvement and 50% better correctness
Impact Analysis

- Automotive system (Delphi)
- Similar SysML modeling methodology
- Use models to support requirements change impact analysis
Our Objective

- Given a change in a requirement, our goal is to compute a set of (potentially) impacted elements that
  - (high recall) Includes all the actually impacted elements, and
  - (high precision) Includes very few non-impacted elements (false positives)
The eVCP controller shall switch off the power stages as soon as the measured temperature exceeds 142degC.

**Requirement:**
Over Temperature Detection

**Software Block:**
Over Temperature Diagnostics

**Software Block:**
Diagnostics Manager

**Software Block:**
Diagnostics

**Software Block:**
Position and Diagnostics Signal Generation

Text: "The eVCP controller shall switch off the power stages as soon as the measured temperature exceeds 142degC."

Id: "R11"
Behavioural Analysis
(Forward Static Slicing)
Research Question (1)

• How much our Behavioral and Structural analysis can help in identifying actually impacted elements?

• The model size: 400

• Average impact set size after structural analysis: 80, and after combined structural and behavioral analysis: 38

• Recall for both structural and combined approaches: 100%
Analysis based on Natural Language Processing

- Two textual descriptions provided with a change request
  - A textual description of a change:
    
    E.g., Change to R12: Temperature range should be extended to -40/150 °C from -20/120 °C

- A preliminary and early analysis of impact
  
  E.g., impacts voltage divider (hardware) and lookup tables (software)
Our NLP-Based Analysis

- We identify noun phrases in change/impact descriptions (text chunking technique)

- We compare the similarity degree between these noun phrases and design element labels
  - Semantic NLP similarity measures
  - Syntactic NLP similarity measures

- We sort the design elements obtained after structural and behavioral analysis based on these similarity measures

- Engineers inspect the sorted lists to identify impacted elements
Research Question (2)

- Which NLP Similarity Measures Perform Best?

<table>
<thead>
<tr>
<th>Syntactic Measures</th>
<th>Semantic Measures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block Distance</td>
<td>HSO</td>
</tr>
<tr>
<td>Cosine Similarity</td>
<td>LCH</td>
</tr>
<tr>
<td>Dice’s coefficient</td>
<td>JCN</td>
</tr>
<tr>
<td>Euclidean</td>
<td>LESK</td>
</tr>
<tr>
<td>Jaccard</td>
<td>LESK_TANIM</td>
</tr>
<tr>
<td>SOFTTFIDF</td>
<td>LIN</td>
</tr>
<tr>
<td>Levenstein</td>
<td>PATH</td>
</tr>
<tr>
<td>Monge Elkan</td>
<td>RES</td>
</tr>
</tbody>
</table>
Research Question (3)

• How much improvement in Precision does our NLP technique bring about?

• More than 13.2 % improvement in Precision after applying NLP

• Recall remains at 100%

• Engineers need to prune roughly two thirds of elements from the generated impact set
Requirements-Test Cases

BodySense

[ISSTA 2015]
Context

- Context: Automotive, sensor systems
- Traceability between system requirements and test cases
- Mandatory when software must undergo a certification process (e.g. ISO 26262)
- Customers require such compliance
- Use-case-centric development
Automated Test Generation

- Restricted use case specifications: Structure, templates, restricted natural language (RUCM)
- Domain modeling
- Constraints
- Automation combines Natural Language Processing and constraint solving
- Automated test generation comes with traceability between use case flows and system test cases
1. Elicit Use Cases

2. Model the Domain

3. Evaluate Completeness

4. Identify Constraints

5. Specify Constraints

6. Generate Abstract Test Cases

7. Generate Platform Specific Test Cases

- TEMPERATURE IS LOW
- STATUS IS VALID
- ERRORS ARE ABSENT

- t > 0 and t < 50
- Status <> null
- Errors.size() = 0

- Domain Model
- Missing Entities

- List of Constraint descriptions
- OCL constraints

- RUCM Use Cases
- Abstract Test Cases
- Mapping Table
- Platform Test Cases
Case Study Results

Applicability

• Rewrote 6 use case specifications of BodySense
• 48 constraints to specify

Effectiveness

• Automatically generated test cases for 6 use cases
• Specific test strategy (Rationale)

• Approach covers more scenarios than manual testing: 100 versus 86
• Automated testing covers alternative flows not covered by manual testing
Discussion

- Modeling effort reasonable after initial training
- Main challenge is writing OCL constraints.
- Test generation time took about 12 min per test case, mostly due to constraint solving
- Engineers miss important test scenarios because
  - Path analysis across multiple use case specifications is difficult
- Regular use case specifications are less precise than RUCM
Regulations - Requirements

- New tax system
- Customs and excise: complex European laws
- Systems need to be compliant with the law and remain so over time

[RE 2014, MODELS 2014]
Solution Overview

Actual software system

Test cases

Generates

Analyzable interpretation of the law (models)

Traces to

Simulates

Impact of legal changes

Simulated result

Results match?

Yes

No
Art. 105bis [...] The commuting expenses deduction (FD) is defined as a function over the distance between the principal town of the municipality on whose territory the taxpayer's home is located and the place of taxpayer's work. The distance is measured in units of distance expressing the kilometric distance between [principal] towns. A ministerial regulation provides these distances.
The amount of the deduction is calculated as follows:

If the distance exceeds **4 units** but is less than **30 units**, the deduction is **€ 99** per unit of distance. The first **4 units** does not trigger any deduction and the deduction for a distance exceeding **30 units** is limited to **€ 2,574**.
Discussion

- We addressed the gap between legal experts and IT specialists
- Models understandable by both legal experts and IT specialists
- Modeling effort was considered reasonable given the life span of such eGovernment systems
- Traceability to the law was considered a significant asset given frequent and complex changes in the law
Conclusions
Conclusions

- From an economic standpoint,
  - the accuracy of trace recovery techniques cannot be interpreted out of context
  - what traceability information to capture is a trade-off
  - benefits depend on context

- More human studies are required to assess cost-benefits

- Design of such studies is not easy: baseline of comparison, comparable tasks, training, comparable skills …
Conclusions

• Change impact analysis among requirements was surprisingly accurate

• Change rationale needed to be captured

• But this is expected to depend on requirements writing practice, e.g., precision and consistency

• Accurate inter-requirements traces may require capturing tacit dependencies between domain concepts, e.g., domain model

• What type of domain model do we need? Ontologies?

• Can accuracy be improved through the use of NL templates?
Conclusions

- Requirements-design traces require a precise design methodology, including practical mechanisms to capture design rationale and link it to requirements.

- Documenting design rationale cannot be automated, but can be facilitated.

- Questions, in each new context:
  - What is the right Modeling methodology?
  - What is the right trace granularity?
  - What information do traces need to carry?

- Change impact analysis: Models are expensive, tradeoff between modeling requirements and accuracy, combining model analysis and NLP can be effective.
Rationale Matters!
Traceability is an economic decision
Context Matters!
Natural Language Requirements

• [RE 2015] C. Arora et al., Change Impact Analysis for Natural Language Requirements: An NLP Approach

• [TSE 2015] C. Arora et al., Automated Checking of Conformance to Requirements Templates using Natural Language Processing

• [ESEM 2014] C. Arora et al., Improving Requirements Glossary Construction via Clustering

• [ESEM 2013] C. Arora et al., Automatic Checking of Conformance to Requirements Boilerplates via Text Chunking

Requirements-Driven Testing

• [ISSTA 2015] C. Wang et al., Automatic Generation of System Test Cases from Use Case Specifications
SysML Traceability and Safety Analysis

- [TOSEM 2014] L. Briand et al., Traceability and SysML Design Slices to Support Safety Inspections: A Controlled Experiment

Legal Modeling and Analysis

- [MODELS 2014] G. Soltana et al., UML for Modeling Procedural Legal Rule
General Literature on Traceability


Traceability Beyond Source Code: An Elusive Target?

Lionel Briand

Interdisciplinary Centre for Security, Reliability and Trust (SnT)
University of Luxembourg

Rennes, Dec 3, 2015