Analyzing Natural-Language Requirements:

The Not-too-sexy and Yet Curiously Difficult Research that Industry Needs

Lionel Briand

REFSQ, Feb 28th, 2017

Interdisciplinary Centre for ICT Security, Reliability, and Trust (SnT)
University of Luxembourg, Luxembourg
SVV Lab Overview

• Established in 2012, part of the SnT centre

• Requirements Engineering, Security Analysis, Design Verification, Automated Testing, Runtime Monitoring

• ~ 25 lab members

• Eight partnerships

• Budget 2016: ~2 Meuros
Mode of Collaboration

- Basic and applied research driven by industry needs
- High-impact research
- Develop and evaluate innovative solutions in context
Objectives of the Talk

- Most requirements are stated in some form of natural language, with varying degrees of structure
- What are the challenges?
- How can we exploit such requirements?
- What form of automated support can be provided?
- Objective: Report on experience performing collaborative research with industrial partners
Natural Language Requirements

- Fuzzy
- Hard to analyze
- Not mathematical

But ...
- Easier to write and read
- Usable by most engineers, in most contexts
- Commonplace
Natural Language Processing

- Automated techniques to fruitfully process natural language corpora: Part-of-Speech (PoS) tagging, grammar-based parsing, lexical semantic analysis …

- Many applications: translation, NL understanding, sentiment analysis, text classification, …

- First applications in translating Russian to English (1950s).

“The spirit is willing but the flesh is weak” => Russian => “The vodka is good but the meat is rotten”
**Natural Language Processing**

- **Huge progress has been made**
- Requirements engineering can greatly benefit from it too
- NLP has a long history in RE research
- Traceability, transformation, ambiguity detection
  ...
- **Limited use in RE practice and much room for improvement**
- **Commercial and OS requirements management tools provide no or limited NLP analysis**
Industrial Challenges
Compliance with Templates

- Templates and guidelines address **ambiguity and incompleteness** in NL requirements
- Large number of requirements
- People tend **not to comply** with templates and guidelines, unless they are **checked and enforced**
- **Scalable and accurate automation** is needed
Domain Knowledge

- All requirements depend, more or less explicitly, on domain knowledge
- **Domain-specific concepts and terminology**
- **Not always consistent** among all stakeholders
- Software engineers often have a **superficial understanding of the application domain** they target
- Capturing domain knowledge: **Glossary, domain model**
Traceability

• In many domains various types of traceability are required
• For example, in automotive (ISO 26262), traceability between requirements and system tests: requirements-driven testing
• Many requirements, many tests, therefore many traces . . .
• Automation is required
Change

- Requirements change frequently
- Changes have **side-effects** on other requirements, design decisions, test cases …
- How do we **support such changes** in ways that scale to hundreds of requirements or more?
- **Automated impact analysis**
Configuring Requirements

- Many software systems are part of product families targeting **varying needs among multiple customers**
- Requirements typically need to be tailored or configured for each customer
- Because of interdependencies among such decisions, this is often error-prone and complex
- **How do we support this with natural language requirements?**
Addressing the Challenges
Representative Context

SES

your satellite company
Challenges

• Large projects in satellite domain (e.g., ESA)
• Hundreds of natural language requirements
• Three tiers of requirements
• Many stakeholders
• Requirements capture a contract
• Requirements frequently change
Checking Compliance with Templates
Rupp’s Template

As soon as the visual notification is presented, the SOT Operator shall launch the local S&T application as a separate process.
There must be something existing
RQA (Glossary)
DODT (Ontology)

Requirements Templates
Mitigate ambiguity

Template Conformance?
Large number of requirements
Evolving requirements

NL Requirements
Ambiguity prone
Contractual Basis
Approach

- **Text chunking**: identifies sentence segments (chunks) without performing expensive analysis over the chunks’ internal structure, roles, or relationships
- **Templates**: **RUPP and EARS**, expressed as BNF grammars and then pattern matching rules
- **Practical**: No reliance on glossary, ontology …
- **Scalable**: Hundreds of requirements in a few minutes
Text Chunking

Process of decomposing a sentence into non-overlapping segments.

As soon as the visual notification is presented the SOT Operator shall launch the local S&T application as a separate process.

Noun Phrase (NP) Verb Phrase (VP) Subordinate Clause (SBAR)

Prepositional Phrase (PP) Adverbal Phrase (ADVP)
As soon as the visual notification is presented the SOT Operator shall launch the local S&T application as a separate process.

CONFORMANT
Evaluation

2 case studies

380 Requirements
380 Requirements

1 case study

Safety-Critical Software
110 Requirements

1 case study

Radioactive Materials
890 Requirements
Results

- **Absence of glossary** has no significant impact on the accuracy of template conformance checking

- Avg. Recall - **94.3%**

- Avg. Precision - **91.6%**
Tool: RETA

Requirements Analyst

GATE NLP Workbench

- Glossary (optional)
- Conformance Diagnostics (within GATE)

Requirements

- Lists of modals, conditional words, ambiguous terms, etc.
- Rules for checking template conformance
- Rules for checking best practices

http://sites.google.com/site/retanlp/
Change Impact Analysis
Inter-Requirements

Change Impact Analysis
Story Behind

- Large number of requirements
- So many stakeholders
- Consistency needs to be maintained (Contractual basis)

How do you manage all the changes?

MANUALLY
Approach

- A change in requirements may lead to changes in other requirements
- Hundreds of requirements
- No traceability
- We propose an approach based on: (1) Natural Language Processing, (2) Phrase syntactic and semantic similarity measures
- Results: We can accurately pinpoint which requirements should be inspected for potential changes
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.
• **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.

Possible Rationales:

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

• **Account 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**

• **Consider semantically-related phrases** that are not exact matches and **close syntactic variations** across requirements
Narcia

1. Process requirements statements
2. Apply change
3. Identify differences
4. Specify propagation condition
5. Sort requirements based on relevance to change

Requirements document

Phrase annotations

Similarity functions

S(x, y)

Boolean expression

Sorted requirements

https://sites.google.com/site/svvnarcia/
<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>The mission operation controller shall transmit satellite status reports to the user help desk.</td>
</tr>
<tr>
<td>R2</td>
<td>The satellite management system shall provide users with the ability to transfer maintenance and service plans to the user help desk via FTP.</td>
</tr>
<tr>
<td>R3</td>
<td>The mission operation controller shall transmit any detected anomalies to the user help desk.</td>
</tr>
<tr>
<td>R4</td>
<td>The mission operation controller shall transfer all the monitoring failure reports to the user, in the priority list of his help desk.</td>
</tr>
<tr>
<td>R5</td>
<td>The mission operation controller shall provide a mechanism for updating user-defined parameters in the configuration database.</td>
</tr>
<tr>
<td>R6</td>
<td>The satellite management system shall authorise all updates to the telemetry configuration of a satellite before applying the changes to the satellite telemetry database.</td>
</tr>
</tbody>
</table>
Evaluation

158 Requirements
9 change scenarios

72 Requirements
5 Change Scenarios
Effectiveness of Our Approach

1 impacted requirement missed out of a total of 106 impacted requirements.

Futile Inspection Effort

1% - 7%

6% - 8%

Case-A

Case-B
Requirements to Design

Requirements-to-Design Change Impact Analysis
Motivations

- **Rigorous change management** required by many standards and customers in safety critical systems, and embedded systems in general in many industry sectors

- Impact of requirements changes on design decisions

- Complete and precise design impact set

- **SysML** commonly used as system design representation
Requirements Diagram

- **Temperature Diagnostics**
  - text = "The CP controller shall provide temperature diagnostics."
  - id = "R1"

- **Over-Temperature Detection**
  - text = "The CP controller shall detect temperatures exceeding 110 °C."
  - id = "R11"

- **Operational Temperature Range**
  - text = "The CP controller shall be able to measure temperatures between -20 °C and 120 °C."
  - id = "R12"
Behavourial Diagram

Diagnostics Manager

<<Decision>>
Is position valid?

<<Assignment>>
Error = 1

<<Decision>>
Over-Temp detected?

<<Assignment>>
MotorDriveMode = RUN

<<Assignment>>
MotorDriveMode = OFF
Compute Impacted Elements

- Structural Analysis
- Behavioural Analysis
Structural Diagram

<<requirement>>
Over-Temperature Detection (R11)
<<requirement>>
Operational Temperature Range (R12)

<<satisfy>>

B1
:Temperature Processor

B2
:Over-Temperature Monitor

B3
:Diagnostics Manager

B4
:DC Motor Controller

B5
:Diagnostics and Status Signal Generation

B6
:Digital to Analog Converter
Change to R11: Change over temperature detection level to 147 C from 110 C.
Over-Temperature Monitor

<<requirement>>
Over-Temperature Detection (R11)

<<requirement>>
Operational Temperature Range (R12)

Diagnostics Manager

Diagnostics and Status Signal Generation

DC Motor Controller

Digital to Analog Converter

B1

B2

B3

B4

B5

B6

<<satisfy>>
Diagnostics Manager

<<Decision>>
Is position valid?

[yes]  [no]

<<Assignment>>
MotorDriveMode = RUN

<<Assignment>>
MotorDriveMode = OFF

<<Assignment>>
Error = 1

<<Decision>>
Over-Temp detected?

[yes]  [no]
Diagnostics Manager

<<Decision>> Is position valid?

<<Assignment>> MotorDriveMode = OFF

<<Assignment>> MotorDriveMode = RUN

<<Assignment>> Error = 1

Behavioural Diagram

input from B2

output to B4

output to B5
Structural Diagram

<<requirement>>

Over-Temperature Detection (R11)

Over-Temperature Monitor

<<satisfy>>

Operational Temperature Range (R12)

Temperature Processor

<<satisfy>>

Diagnostics and Status Signal Generation

Diagnostics Manager

DC Motor Controller

Digital to Analog Converter

B1

B2

B3

B4

B5

B6
Change to R11: Change over temperature detection level to 147°C from 110°C.

B2, B3, B4, B6

Natural Language Processing Analysis

Ranked according to likelihood of impact
Change Statements

- **Informal inputs** from systems engineers regarding impact of changes

  - **Example:** “Temperature lookup tables and voltage converters need to be adjusted”
Natural Language Processing

- Computing **similarity scores** for model elements by applying NLP techniques to measure similarity **between model elements labels and change statements**.

- **Sorting** the design elements obtained after structural and behavioral analysis based on the similarity scores

- Engineers inspect the **sorted lists** to identify impacted elements
Identifying a Subset to Inspect

- Pick the last significant peak in delta similarity between two successive elements
Approach

Requirements Changes and Informal Change Statements

Process Change Statements

Phrases Similarity Matrix

Build SysML Models

System Requirements

Requirements and Design Models

Compute Impacted Elements

Estimated Impact Set

Sort Elements

Sorted Elements
Evaluation

1 case study

370 elements
16 change scenarios
Effectiveness of Our Approach
Effectiveness of Our Approach

![Box plot showing Futile Inspection Effort (%)]

- **Structural**
- **Behavioural**
Effectiveness of Our Approach

1 impacted element missed out of a total of 81 impacted elements.
Glossary Extraction and Clustering
NL Requirements

- Usually multiple stakeholders, organizations …
- Inconsistent terminology
  - Multiple terms for same concepts
    - element / component / object
  - Multiple representations of same keywords
    - status of Ground Station Interface component
    - Ground Station Interface component’s status
    - Interface component status
Requirements Glossary

- Glossaries help mitigate ambiguities
  - consistent terminology
  - improves communication among stakeholders
Story Behind

Wait, I think we’ve used different variations for the terms. Let me fix these variations in the document.

Let’s automatically identify the glossary terms using text chunking.
R1 - STS shall supply GSI monitoring information (GSI input parameters and GSI output parameters) to the STS subcontractor.

R2 - When GSI component’s status changes, STS shall update the progress of development activities.

- STS
- STS Subcontractor
- development activity
- progress of development activity
- GSI
- GSI input parameter
- GSI output parameter
- GSI component
- GSI component’s status
- GSI monitoring information
Evaluation of Glossary Terms

2 case studies

380 Requirements
138 Requirements

1 case study

110 Requirements
Results

Our Approach

\[ \Delta \text{Recall} > 20\% \]

VS

JATE
TOPIA
TextRank
TermoStat
TermRaider
Results

Our Approach vs Precision ~

JATE
TOPIA
TextRank
TermoStat
TermRaider
Clustering Evaluation

2 case studies

20 clusters each case study

1 case study

27 clusters

• Interview Survey
How useful is our approach?

- I find this cluster helpful for identifying **the related terms** for a glossary term.
  - 89.6% (strongly agreed / agreed)

- As the result of seeing this cluster, I can define a glossary term more precisely than I originally had in mind.
  - 88% (strongly agreed / agreed)

- I find this cluster helpful for identifying **the variations (synonyms)** of a glossary term.
  - 61% (strongly agreed / agreed)
  - 28% (not relevant)
Domain Model Extraction
Motivation

• Representation of important **domain concepts and their relations**
  
  • **Facilitate communication** between stakeholders from different backgrounds
  
  • Help identify **inconsistencies** in terminology, etc.

• In practice, domain models **are not preceding the elicitation and writing of requirements**
A domain model is a representation of conceptual entities or real-world objects in a domain of interest.

Diagram:
- **Satellite**
  - **Satellite Ground Station**
  - **Satellite S&T Station**
  - **Satellite Control Centre**
    - **- location**
  - **transfers user**
  - **requests to**
Context

Requirements Analysts

Specify Requirements

Build Domain Model

NL Requirements Document

Domain Model

Class A

Class B

Class C

Class D

Relation
Problem Definition

- Manually building domain models is **laborious**

- **Automated support is required** for building domain models
State of the Art

- **Multiple approaches exist** for extracting domain models or similar variants from requirements using **extraction rules**
  - Majority assume **specific structure**, e.g., restricted NL
  - Extraction of **direct relations only** but not indirect ones
  - **Limited empirical results** on industrial requirements
Approach

NL Requirements

Process Requirements Statements

Lift Dependencies to Semantic Units

Construct Domain Model

Phrasal Structure Dependencies

Phrase-level Dependencies

Extraction Rules
Approach

1. NL Requirements
2. Process Requirements Statements
3. Lift Dependencies to Semantic Units
4. Construct Domain Model

- Phrasal Structure Dependencies
- Phrase-level Dependencies
- Extraction Rules

Class A
Class B
Class C
Class D
Relation 1
The system operator shall initialize the simulator configuration.
The system operator shall initialize the simulator configuration.
Approach

Process Requirements Statements

Lift Dependencies to Semantic Units

Construct Domain Model

Domain Model

Class A

1

Relation

Class B

Class C

Class D

Extraction Rules

Phrase-level Dependencies

Phrasal Structure Dependencies

NL Requirements
The simulator shall send log messages to the database via the monitoring interface.
How useful is our approach?

- Interview survey with experts
- Correctness and Relevance of each relation
- Missing relations in each requirement

50 Requirements
213 Relations
Results

Correctness - 90% (avg.)

Relevance - 36% (avg.)

Missed Relations - 8%
Requirements-Driven Testing
Context

- **Context**: Automotive, sensor systems

- **Traceability** between system requirements and test cases

- Mandatory when software must comply with ISO 26262

- Customers also require such compliance

- **Use-case-centric development**
Objectives

• Automatically generate test cases from requirements

• Capture and create traceability information between test cases and requirements

  • Requirements are captured through use cases

  • Use cases are used to communicate with customers and the system test team

  • Complete and precise behavioral models are not an option: too difficult and expensive (Model-based testing)
Strategy

- **Analyzeable** use case specifications
- Automatically extract test model from the use case specifications (*Natural Language Processing*)
- **Minimize modeling**, domain modeling only
- **No behavioral modeling**
Errors.size == 0

Status != null
Errors.size() == 0

THE ACTOR SEND
THE ACTOR SEND
THE ACTOR SEND
THE SYSTEM VALI
THE SYSTEM DIS
THE

Use Cases

Domain Model

Evaluate Consistency

Constraints

Test Scenarios

Test Cases

UMTG
Use Case Name: Identify Occupancy Status

Actors: AirbagControlUnit

Precondition: The system has been initialized ...

Basic Flow

1. The seat SENDS occupancy status TO the system.
   Postcondition: The occupant class for airbag control has been sent.
2. INCLUDE USE CASE Classify occupancy status.
3. The system VALIDATES THAT the occupant class for airbag control is valid.
4. The system SENDS the occupant class for airbag control TO AirbagControlUnit.

Specific Alternative Flow

RFS 3
Postcondition: The previous occupant class for airbag control has been sent.
1. IF the occupant class for airbag control is not valid THEN
2. The system SENDS the previous occupant class for airbag control TO ...
The system validates the actor's send. Errors are absent. Temperature is low. Status is valid.

**Identify Constraints**
- Temperature is low
- Status is valid
- Errors are absent

**Specify Constraints**
- $t > 0 \&\& t < 50$
- Status != null
- Errors.size() == 0

**Generate Test Cases**
- Test Scenarios
- Mapping Table
- Test Cases

**Elicit Use Cases**
- RUCM Use Cases

**Model the Domain**
- Domain Model
- Missing Entities

**Generate Scenarios and Inputs**
- Object Diagrams

**Evaluate Consistency**
- Constraint descriptions
1. Elicit Use Cases

2. Model the Domain

3. Evaluate Consistency

4. Identify Constraints

5. Domain Model

6. Generate Scenarios and Inputs

Based on Natural Language Processing
Basic Flow

1. The seat **Sends** occupancy status **to** the system.

2. **Include Use Case** Classify occupancy status.

3. The system **Validates That** the occupant class for airbag control is valid and the occupant class for seat belt reminder is valid.

4. The system **Sends** the occupant class for airbag control **to** AirbagControlUnit.

5. The system **Sends** the occupant class for seat belt reminder **to** SeatBeltControlUnit.

6. The System Waits for next execution cycle.

Postcondition: The occupant class for airbag control and the occupant class for seat belt reminder have been sent.
Evaluate Model Consistency

Tagged Use Case

Domain Entities

Airbag Control
Classification Filter
Sensor

Occupant Class for Airbag Control
Occupant Class for Seat Belt Reminder

AirbagControl
System
Sensor
ClassificationFilter

OccupantStatus
- OccupantClassForAirbagControl
- OccupantClassForSeatBeltReminder
Toolset integrated with IBM DOORS and Rhapsody

https://sites.google.com/site/umtgTestGen/
Case Study

- BodySense, embedded system for detecting occupancy status in a car

Evaluation:

- Cost of additional modelling
- Effectiveness in terms of covered scenarios compared to current practice at IEE
- Keep in mind changes and repeated testing
## Costs of Additional Modeling

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Steps</th>
<th>Use Case Flows</th>
<th>OCL Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC1</td>
<td>50</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>UC2</td>
<td>44</td>
<td>13</td>
<td>7</td>
</tr>
<tr>
<td>UC3</td>
<td>35</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>UC4</td>
<td>59</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>UC5</td>
<td>30</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>UC6</td>
<td>25</td>
<td>6</td>
<td>12</td>
</tr>
</tbody>
</table>

5 to 10 minutes to write each constraints

=> A maximum of 10 hours in total
Effectiveness: scenarios covered

It is hard for engineers to capture all the possible scenarios involving error conditions.
Recently: Extension of the approach for testing timeliness requirements based on use cases and timed automata
Supporting Product Lines and Requirements Configuration in Use-Case Driven Development
Incremental Reconfiguration of Product Specific Use Case Models for Evolving Configuration Decisions

Ines Hajri
joint work with Arda Goknil, Lionel Briand, Thierry Stephany

SnT Center, University of Luxembourg
IEE, Luxembourg

28 February 2017
Context

IEE develops real-time embedded systems:
• Automotive safety sensing systems
• Automotive comfort & convenience systems, e.g., Smart Trunk Opener
Smart Trunk Opener (STO)

STO Provides **automatic** and **hands-free access** to a vehicle’s trunk (based on a **keyless entry system**).
Use Case Driven Development

Domain Model

Use Case Specifications

Use Case Diagram
Dealing with Multiple Customers

STO Requirements from Customer A (Use Case Diagram and Specifications, and Domain Model)

modify

evolves to (clone-and-own)

Customer A for STO

STO Test Cases for Customer A

modify

evolves to (clone-and-own)

STO Requirements from Customer B (Use Case Diagram and Specifications, and Domain Model)

Customer B for STO

STO Test Cases for Customer B

modify

evolves to (clone-and-own)

STO Requirements from Customer C (Use Case Diagram and Specifications, and Domain Model)

Customer C for STO

STO Test Cases for Customer C
Product Line Approach

- A Product Line approach was clearly needed
- Restricted and analyzable use case specifications (NLP)
- Variability modeling in use case diagrams and specifications
- Automated configuration guidance for configuring requirements with each customer
- Automated generation of product-specific use case models based on decisions
Use Cases And Domain Model

Customer A for Product X

Customer B for Product X

Customer C for Product X

Product-Line Use Cases And Domain Model

Identify Commonalities and Variabilities

reConfigure

reConfigure

reConfigure

Configure

evolves
Product Line Use Case Diagram for STO (Partial)
Restricted Use Case Modeling: RUCM

- RUCM is based on a (1) **template**, (2) **restriction rules**, and (3) specific **keywords** constraining the use of natural language in use case specifications.

- RUCM **reduces ambiguity** and **facilitates automated analysis of use cases**.
RUCM

- Flow of events is described in restricted natural language

<table>
<thead>
<tr>
<th>Basic Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. INCLUDE USE CASE Identify System Operating Status.</td>
</tr>
<tr>
<td>2. The system <strong>VALIDATES THAT</strong> the operating status is OK.</td>
</tr>
<tr>
<td>3. The system <strong>REQUESTS</strong> the move capacitance <strong>FROM</strong> the UpperSensor.</td>
</tr>
<tr>
<td>4. The system <strong>REQUESTS</strong> the move capacitance <strong>FROM</strong> the LowerSensor.</td>
</tr>
<tr>
<td>5. The system <strong>VALIDATES THAT</strong> the movement is a valid kick.</td>
</tr>
<tr>
<td>6. The system <strong>VALIDATES THAT</strong> the overuse protection feature is enabled.</td>
</tr>
<tr>
<td>7. The system <strong>VALIDATES THAT</strong> the Overuse protection status is inactive.</td>
</tr>
<tr>
<td>8. The system <strong>SEND</strong>s the valid kick status <strong>TO</strong> the STOController.</td>
</tr>
</tbody>
</table>

**Post condition:** The gesture has been recognised and the STO Controller has been informed.
Example Variability Extension

• **Keyword:** INCLUDE VARIATION POINT: ...

• Inclusion of variation points in basic or alternative flows of use cases:

<table>
<thead>
<tr>
<th>Use Case: Identify System Operating Status</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Basic Flow</strong></td>
</tr>
<tr>
<td>1. The system VALIDATES THAT the watchdog reset is valid.</td>
</tr>
<tr>
<td>2. The system VALIDATES THAT the RAM is valid.</td>
</tr>
<tr>
<td>3. The system VALIDATES THAT the sensors are valid.</td>
</tr>
<tr>
<td>4. The system VALIDATES THAT there is no error detected.</td>
</tr>
<tr>
<td><strong>Specific Alternative Flow</strong></td>
</tr>
<tr>
<td>RFS 4</td>
</tr>
<tr>
<td>1. <strong>INCLUDE VARIATION POINT:</strong> Storing Error Status.</td>
</tr>
<tr>
<td>2. ABORT.</td>
</tr>
</tbody>
</table>
Results

- Tool Support (PUMConf): https://sites.google.com/site/pumconf/

- Positive feedback from engineers, both about the modeling approach and configuration tool

- They confirmed they benefited from:
  - **Understanding the commonalities** and differences across product requirements
  - **Automated guidance** in a configuration that is often complex, i.e., many (interdependent) decisions
Discussion
Many Applications

• Requirements to support a shared understanding among many stakeholders in large projects
• Requirements to support communication between software engineers and domain experts
• Requirements as contract with customers
• Requirements to support compliance with standards, e.g., traceability to tests
• Requirements to support quality assurance, e.g., testing
• Requirements to support change control
But automation is required to justify the cost of rigorous requirements engineering and to achieve its full potential.
Varying Forms of Requirements

- **Natural language** statements, complying or not with templates
- **Use case specifications**, possibly structured and restricted
- **(Formal) models**, e.g., class and activity diagrams
The best form of requirements depends on context, but in most cases significant information is captured in natural language.
Contextual Factors

- Regulatory compliance, e.g., standards
- Project size, team distribution, and number of stakeholders
- Background of stakeholders and communication challenges
- Domain complexity
- Presence of product lines with multiple customers
- Importance of early contractual agreement
- Frequency and consequences of changes in requirements
Choosing an adequate way to capture requirements is essentially a trade-off between RE cost & flexibility and precision & automation
Conclusions

• Many challenges related to Natural Language requirements:
  (1) Ambiguity
  (2) Domain knowledge extraction
  (3) Change impact and management
  (4) Requirements-driven testing

• NLP technology now provides many opportunities for automation and lowering documentation overhead
• But more attention to NL requirements analysis is needed in research
• We need much more (reported) industrial experience
Acknowledgements

- Mehrdad Sabetzadeh
- Chetan Arora
- Fabrizio Pastore
- Chunhui Wang
- Arda Goknil
- Ines Hajri
- Shiva Nejati
Analyzing Natural-Language Requirements:
The Not-too-sexy and Yet Curiously Difficult Research that Industry Needs

Lionel Briand

REFSQ, Feb 28th, 2017

Interdisciplinary Centre for ICT Security, Reliability, and Trust (SnT)
University of Luxembourg, Luxembourg
Natural Language Requirements

- [TSE 2017] C. Arora et al., Automated Extraction and Clustering of Requirements Glossary Terms

- [MODELS 2016] C. Arora et al., Extracting Domain Models from Natural-Language Requirements: Approach and Industrial Evaluation

- [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

Requirements-Driven Testing

- [ISSTA 2015] C. Wang et al., Automatic Generation of System Test Cases from Use Case Specifications

Product Families and Configuration


- [SoSYM 2016] I. Hajri et al., A Requirements Configuration Approach and Tool for Use Case-Driven Development

Impact Analysis

- [FSE 2016] S. Nejati et al., Automated Change Impact Analysis between SysML Models of Requirements and Design

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