# Analyzing Natural-Language Requirements:

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

**Lionel Briand**

**REFSQ, Feb 28th, 2017**

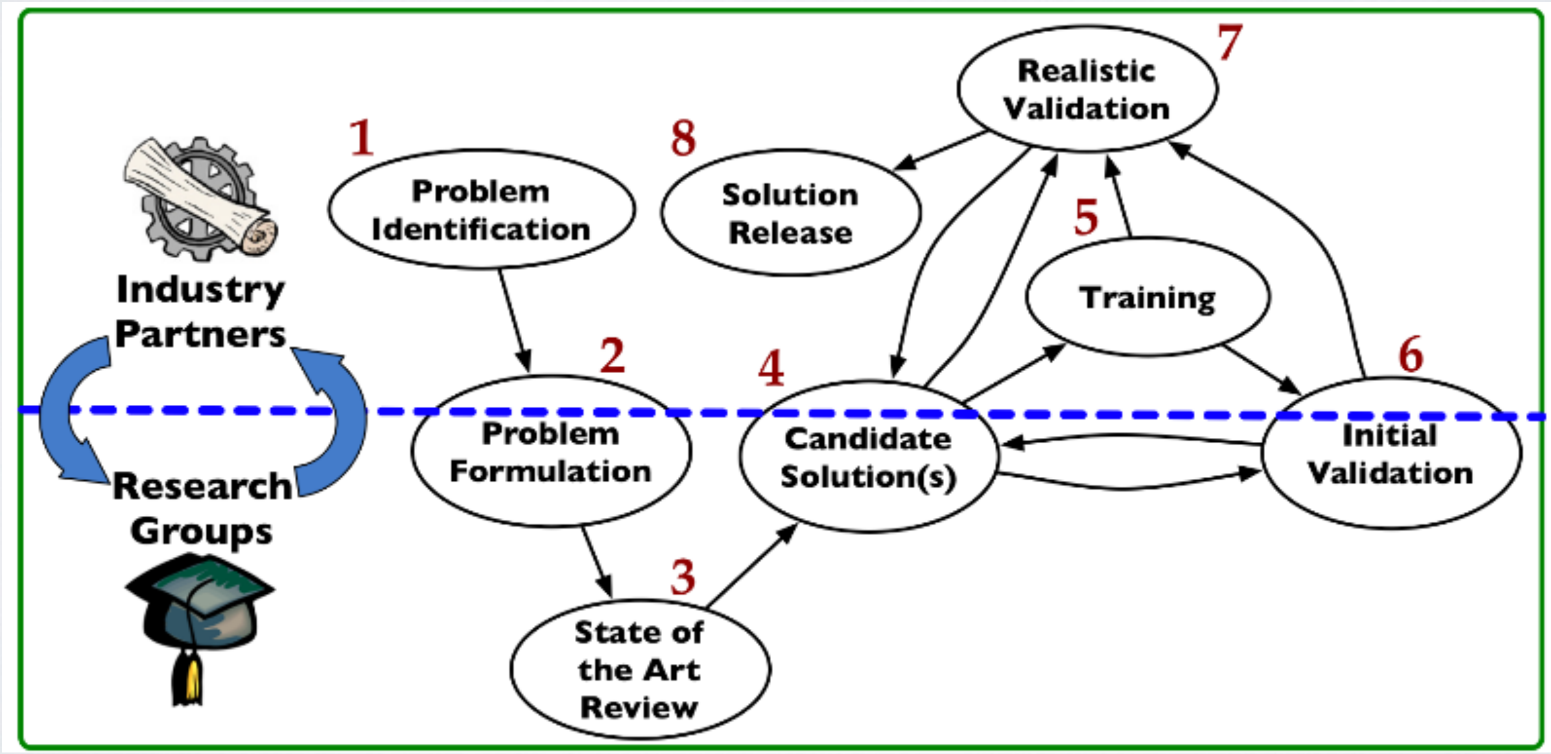
**Interdisciplinary Centre for ICT Security, Reliability, and Trust (SnT)**
**University of Luxembourg, Luxembourg**

# 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

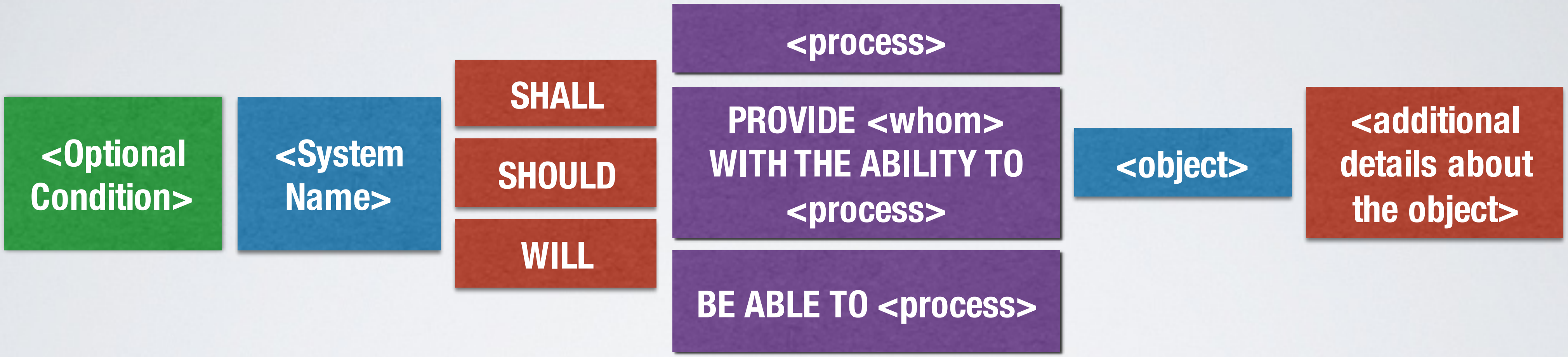
your satellite company

# 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.

Glossary?

# Story Behind

**NL Requirements**

Ambiguity prone

Contractual Basis

**Template Conformance?**

Large number of requirements

Evolving requirements

**Requirements Templates**

Mitigate ambiguity

**There must be something existing**

**RQA (Glossary)**

**DODT (Ontology)**

# 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

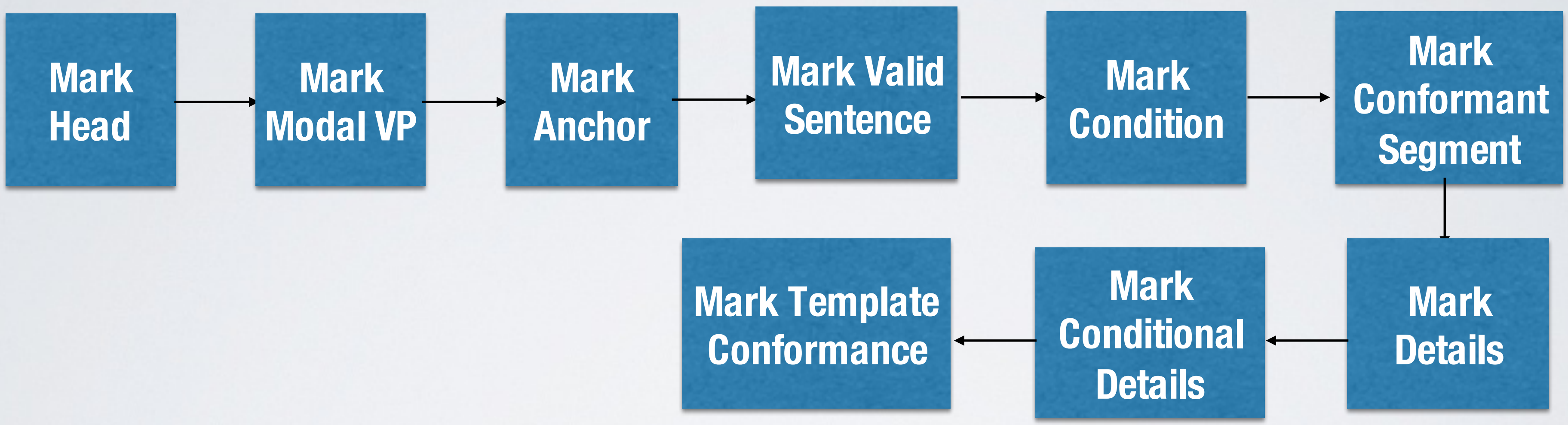
**Process of decomposing a sentence into non-overlapping segments.**

**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) Adverbial Phrase (ADVP)

# Template Conformance Checking




**Valid Sentence**

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

**CONFORMANT**

# Evaluation

SES
your satellite company

OPENCOSS

STUK

2 case studies

1 case study


1 case study


380 Requirements
380 Requirements

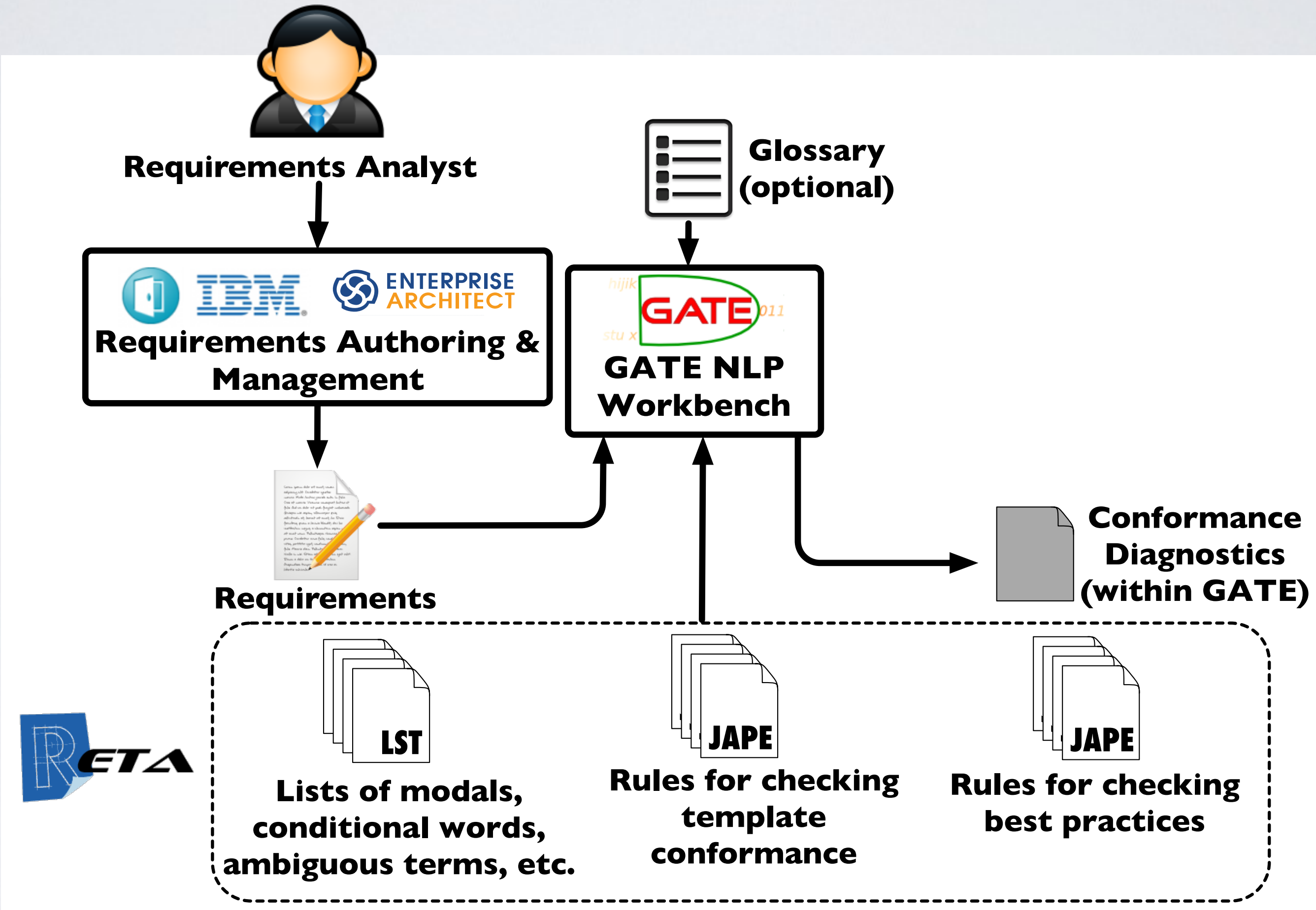
110 Requirements

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




http://sites.google.com/site/retanlp/

# Change Impact Analysis

# Inter-Requirements

Inter-Requirements
Change Impact Analysis

# Story Behind

SES
your satellite company

SW.lu
software verification & validation

- 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.

# Change

- **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

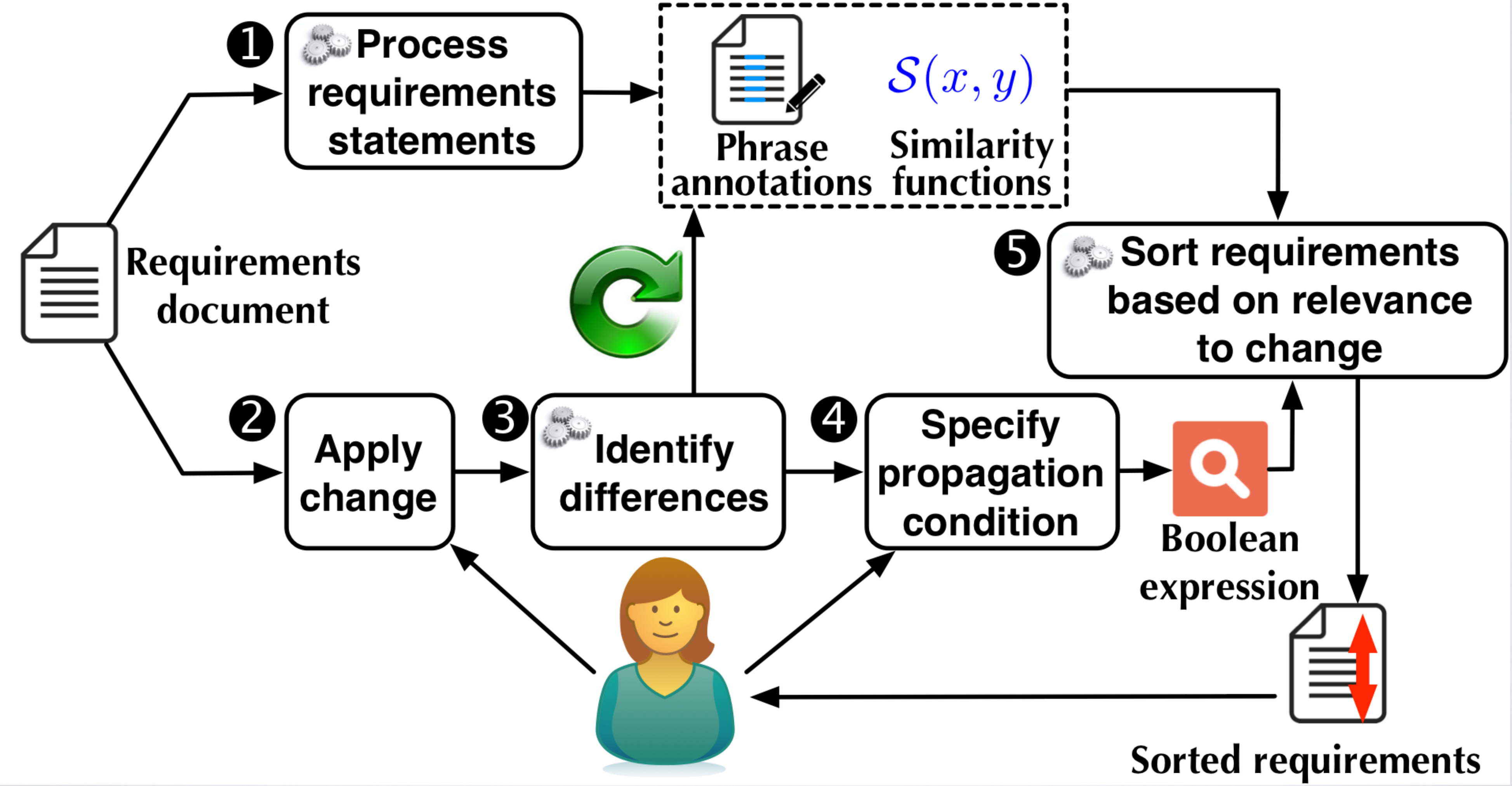
- **Account for the phrasal structure of requirements**

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




https://sites.google.com/site/svvnarcia/

# Narcia in Action

NARCIA

Upload Requirements

Change Requirement

| Requirement | Requirement Text |
| --- | --- |
| 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 via FTP. |
| R3 | The mission operation controller shall transmit any detected anomalies to the user help desk. |
| R4 | The mission operation controller shall transfer all the monitoring failure reports to the user, in the priority list of his help desk. |
| R5 | The mission operation controller shall provide a mechanism for updating user-defined parameters in the configuration database. |
| R6 | The satellite management system shall authorise all updates to the telemetry configuration of a satellite before applying the changes to the satellite telemetry database. |

# Evaluation

SES
your satellite company

1 case study

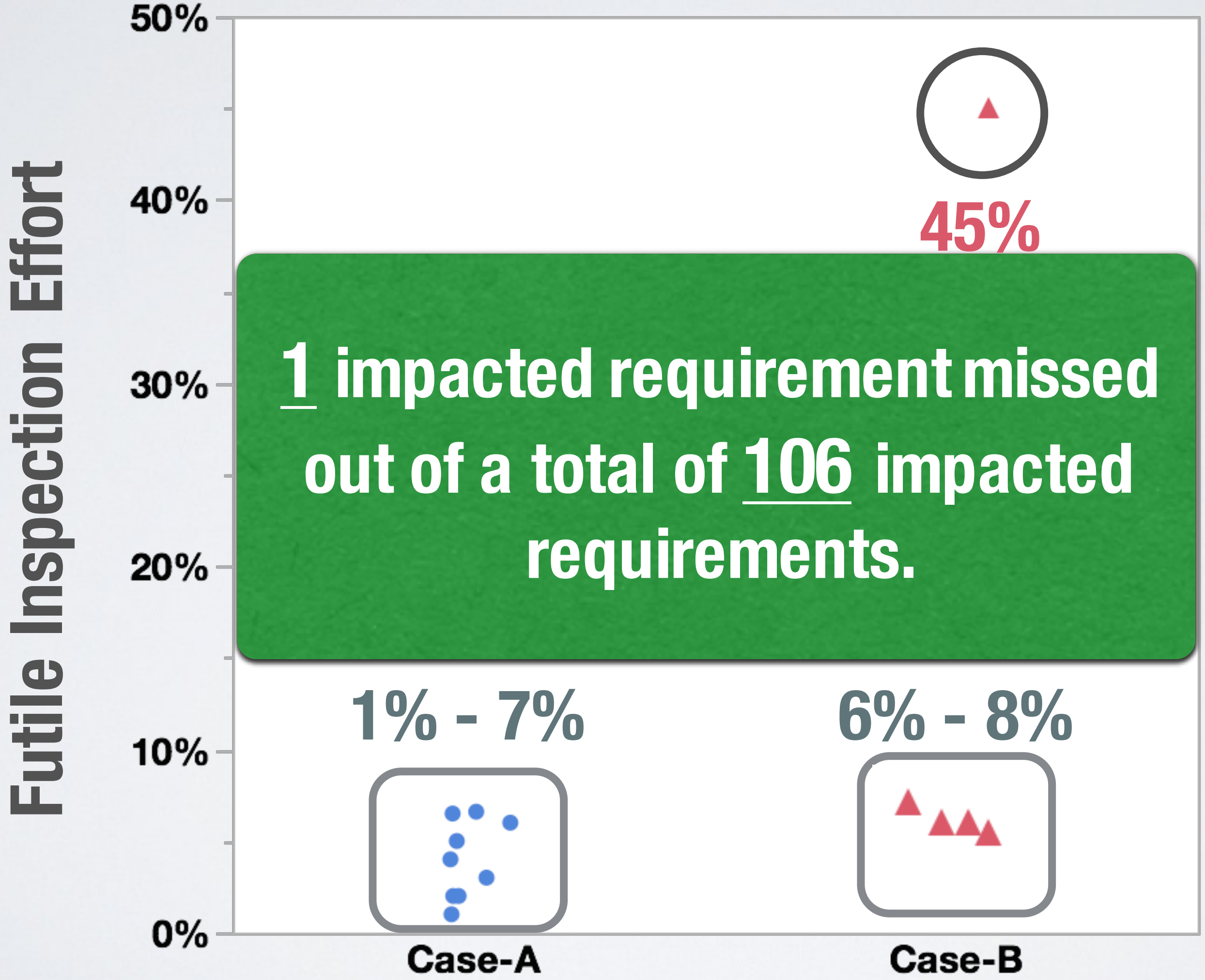
158 Requirements
9 change scenrios

novay

1 case study

72 Requirements
5 Change Scenarios

# Effectiveness of Our Approach



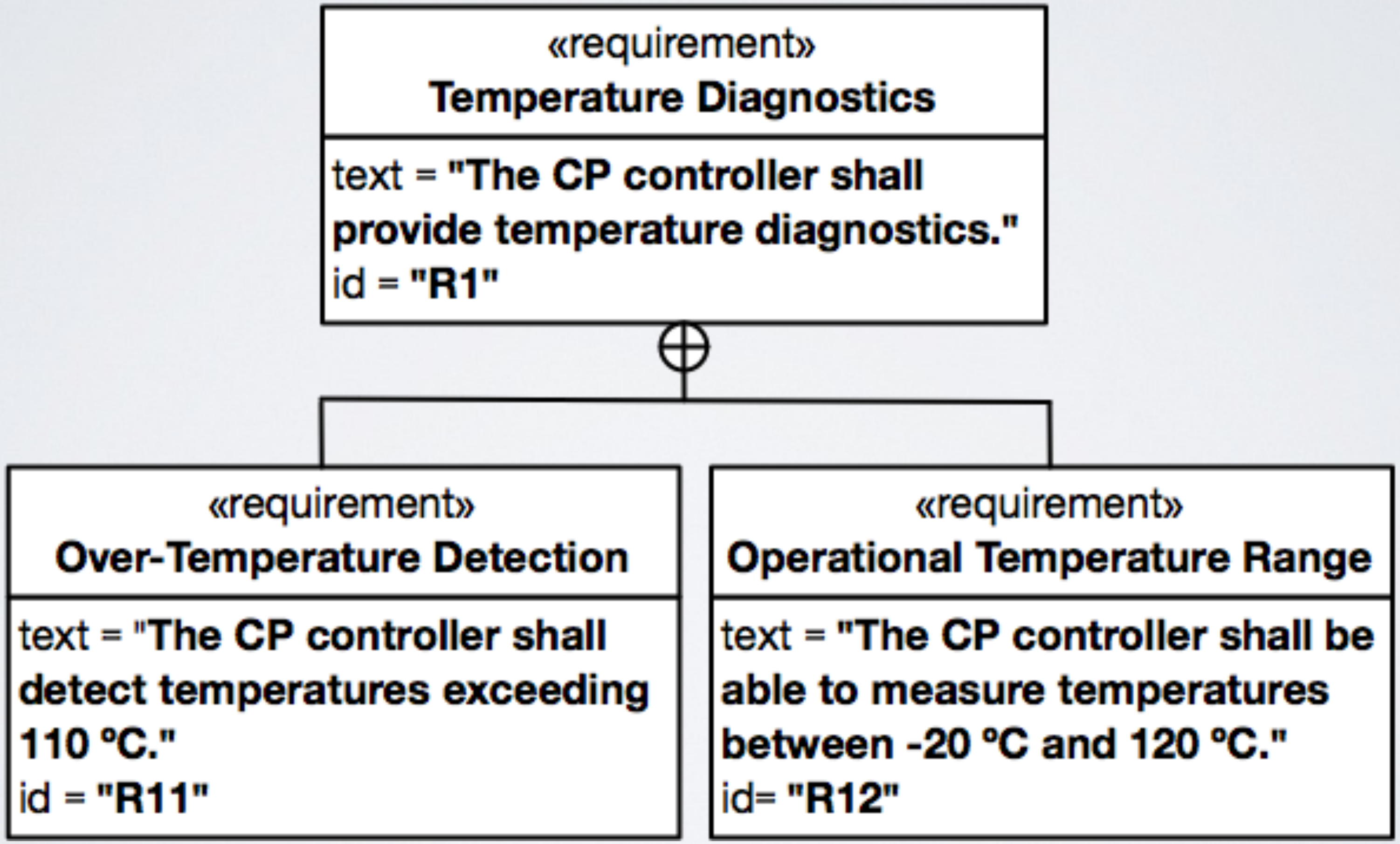
# Requirements to Design

**Requirements-to-Design Change Impact Analysis**

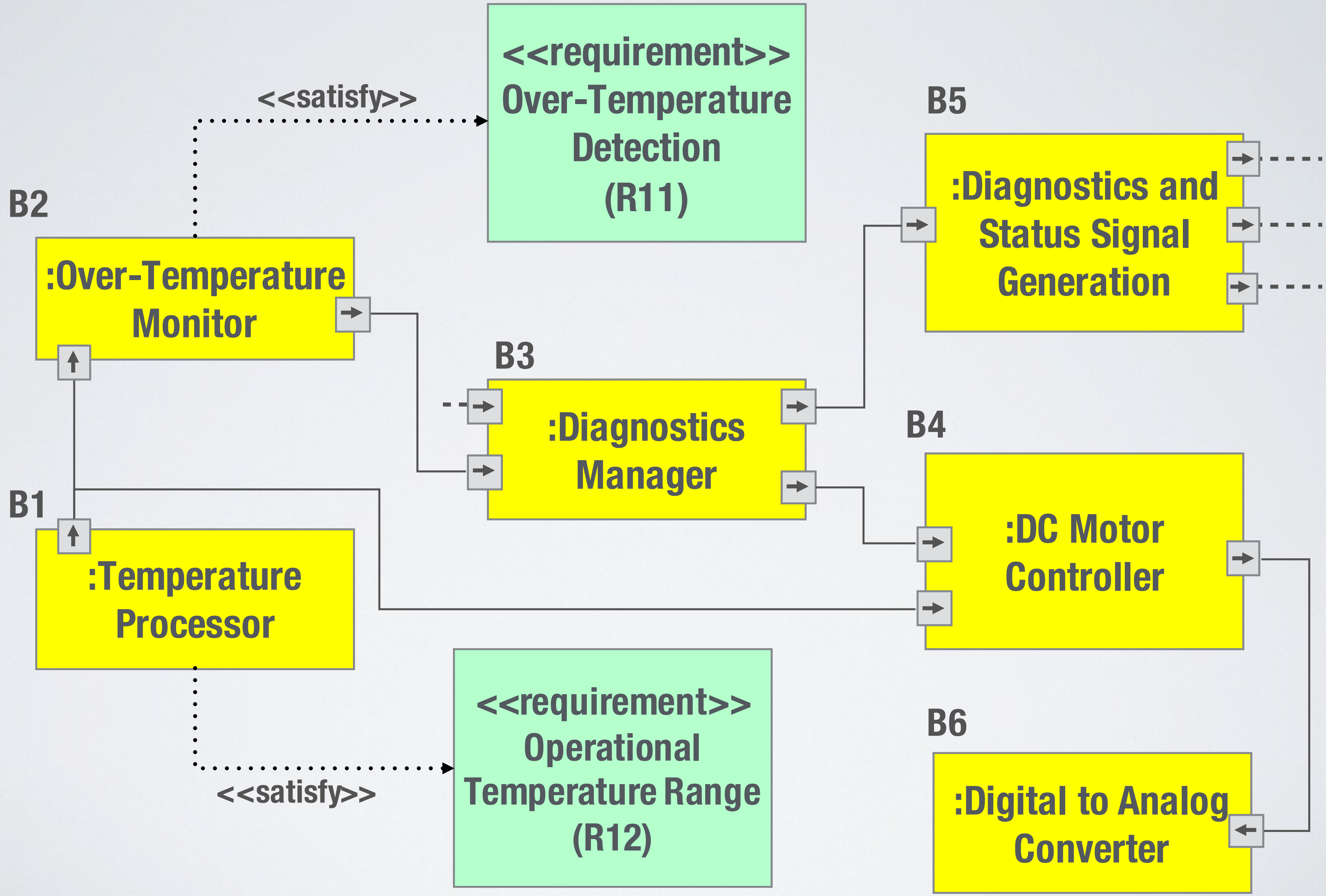
# Motivations

- 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

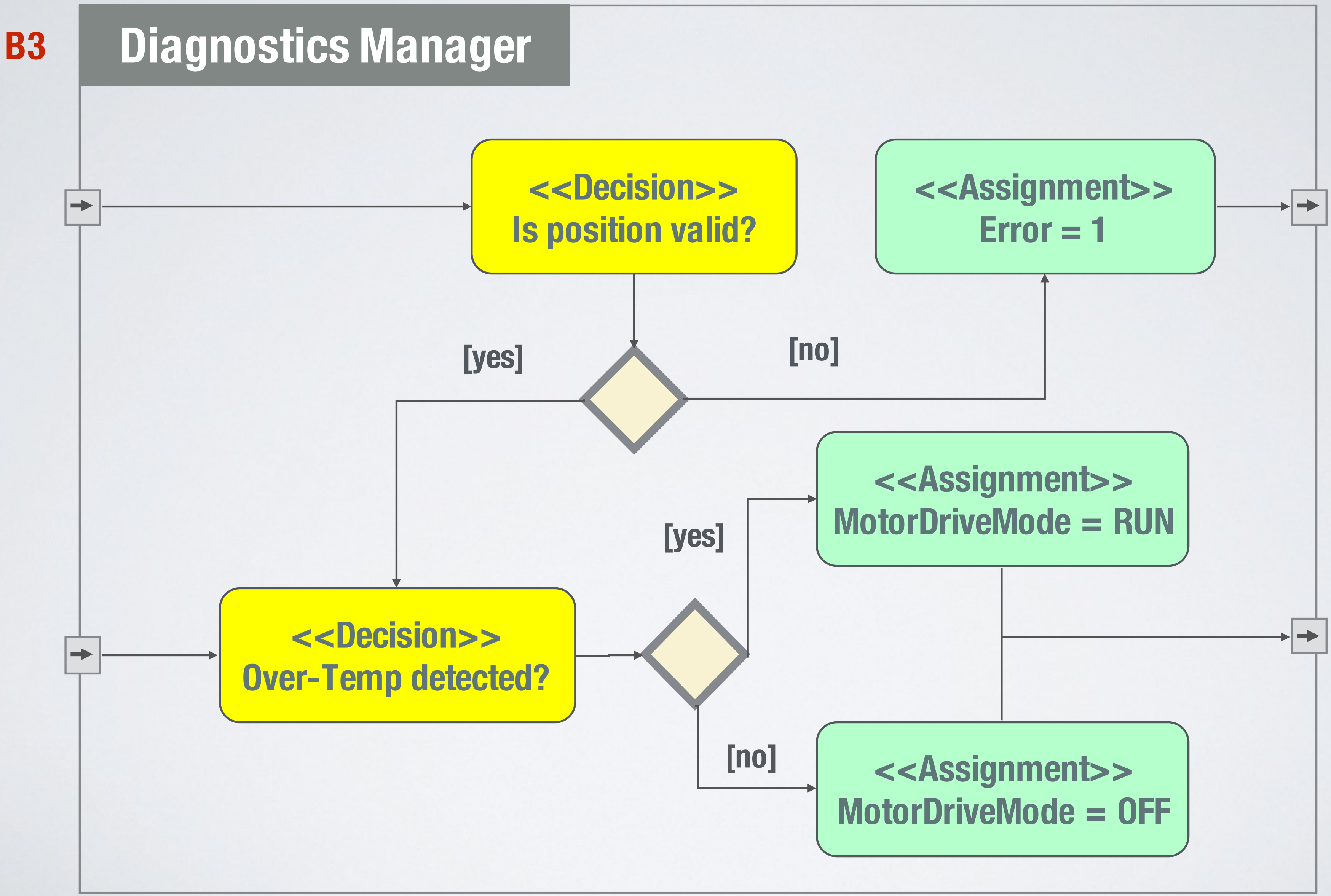
# Requirements Diagram



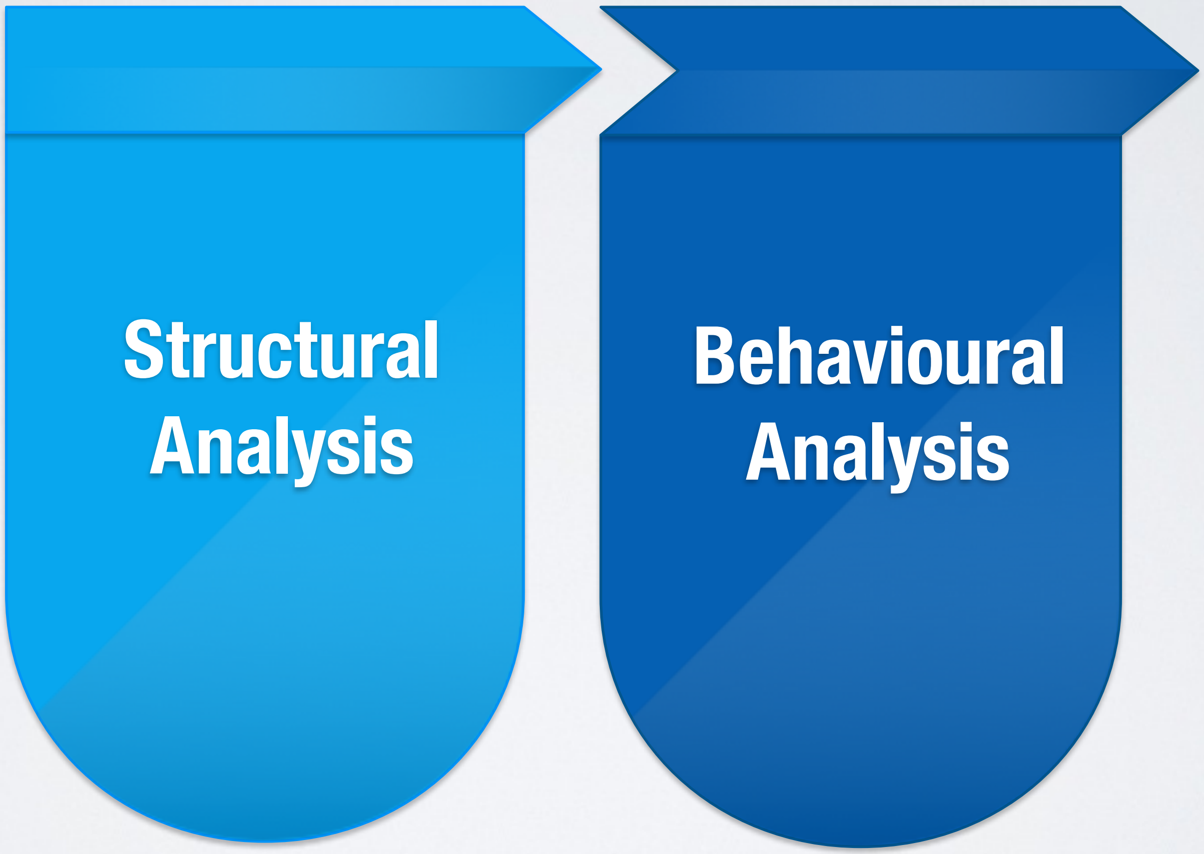
# Structural Diagram



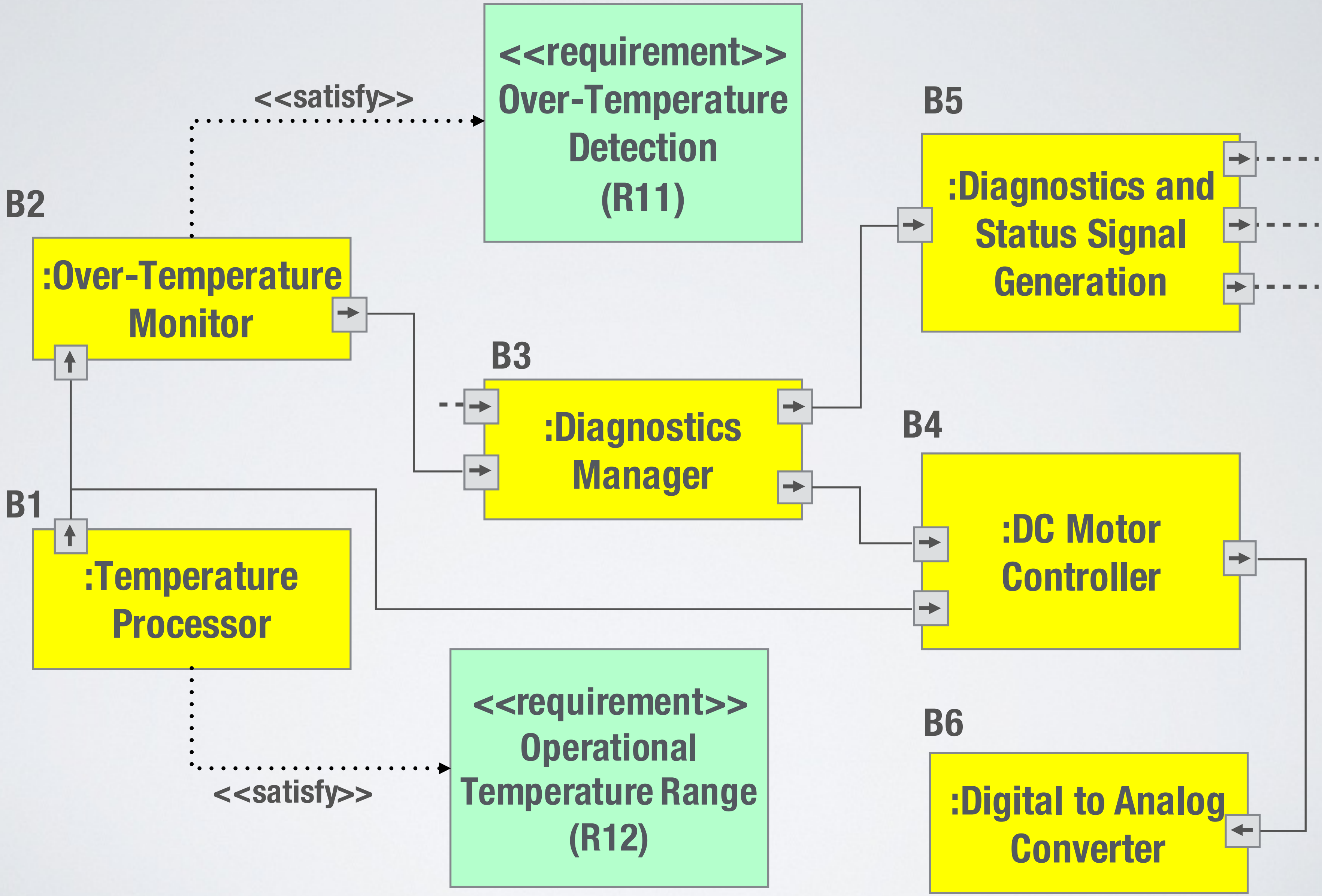
# Behavioural Diagram



# Compute Impacted Elements



# Structural Diagram

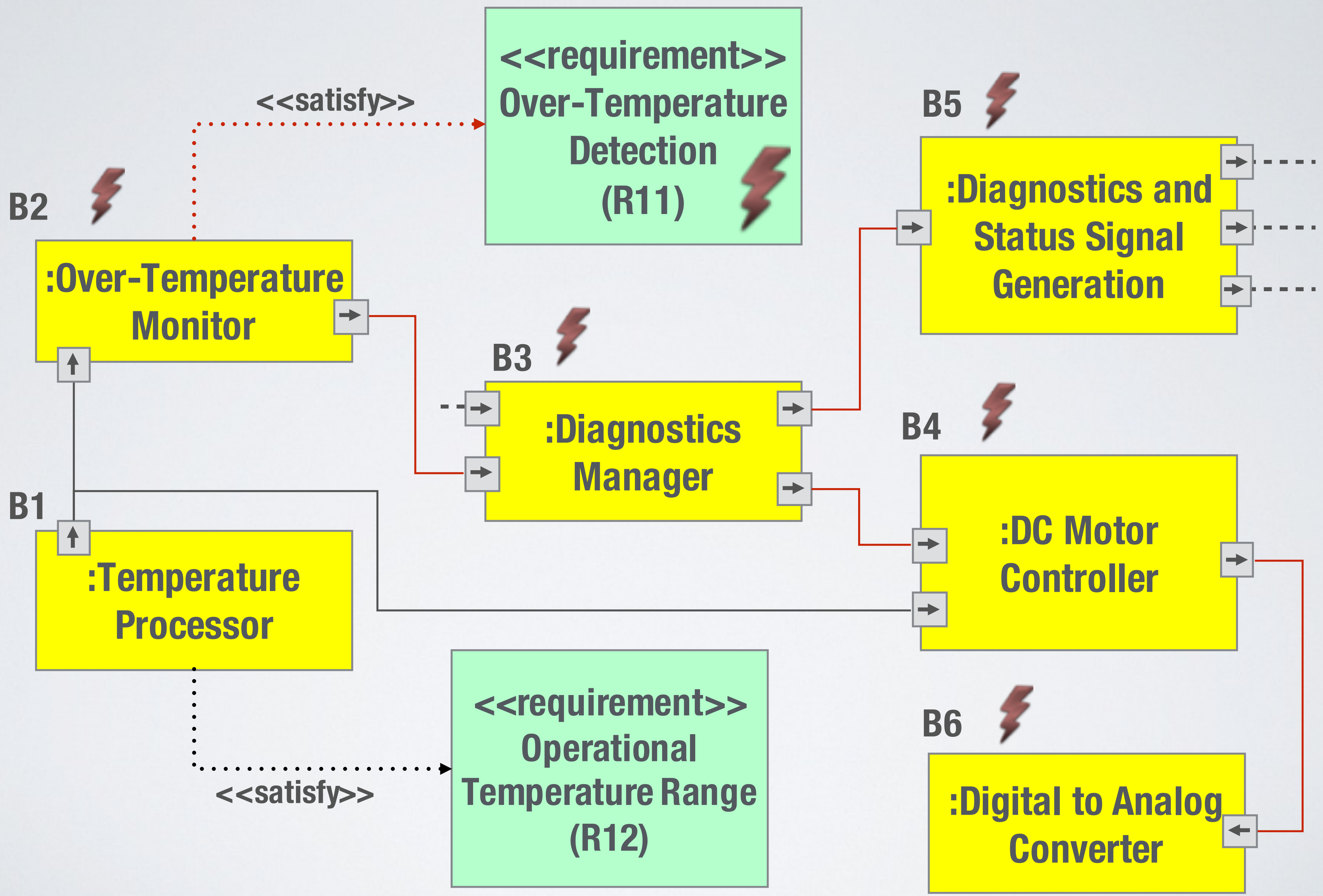


# Structural Diagram

<<requirement>>

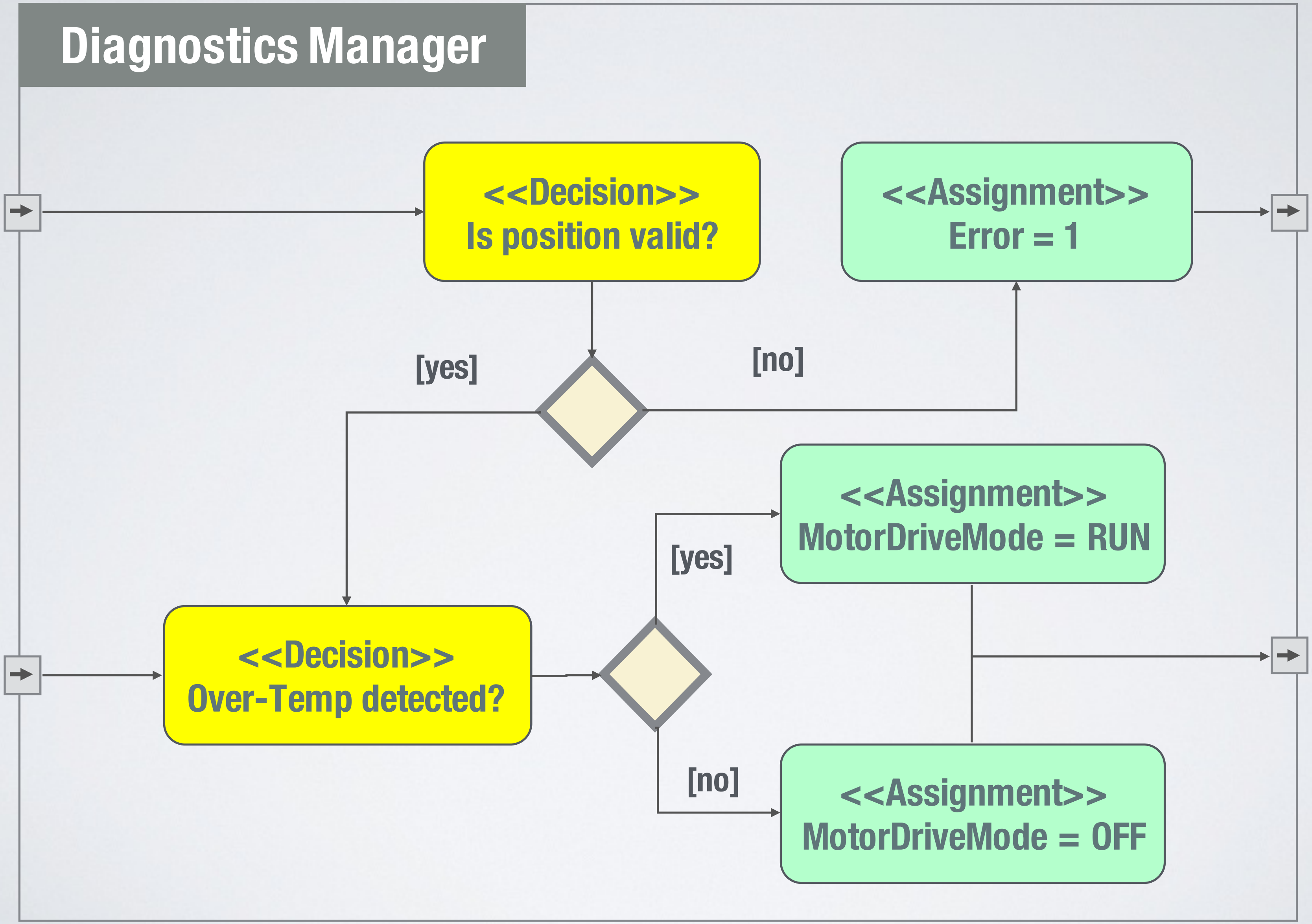
**Change to R11: Change over temperature detection level to 147 C from 110 C.**

# Structural Diagram

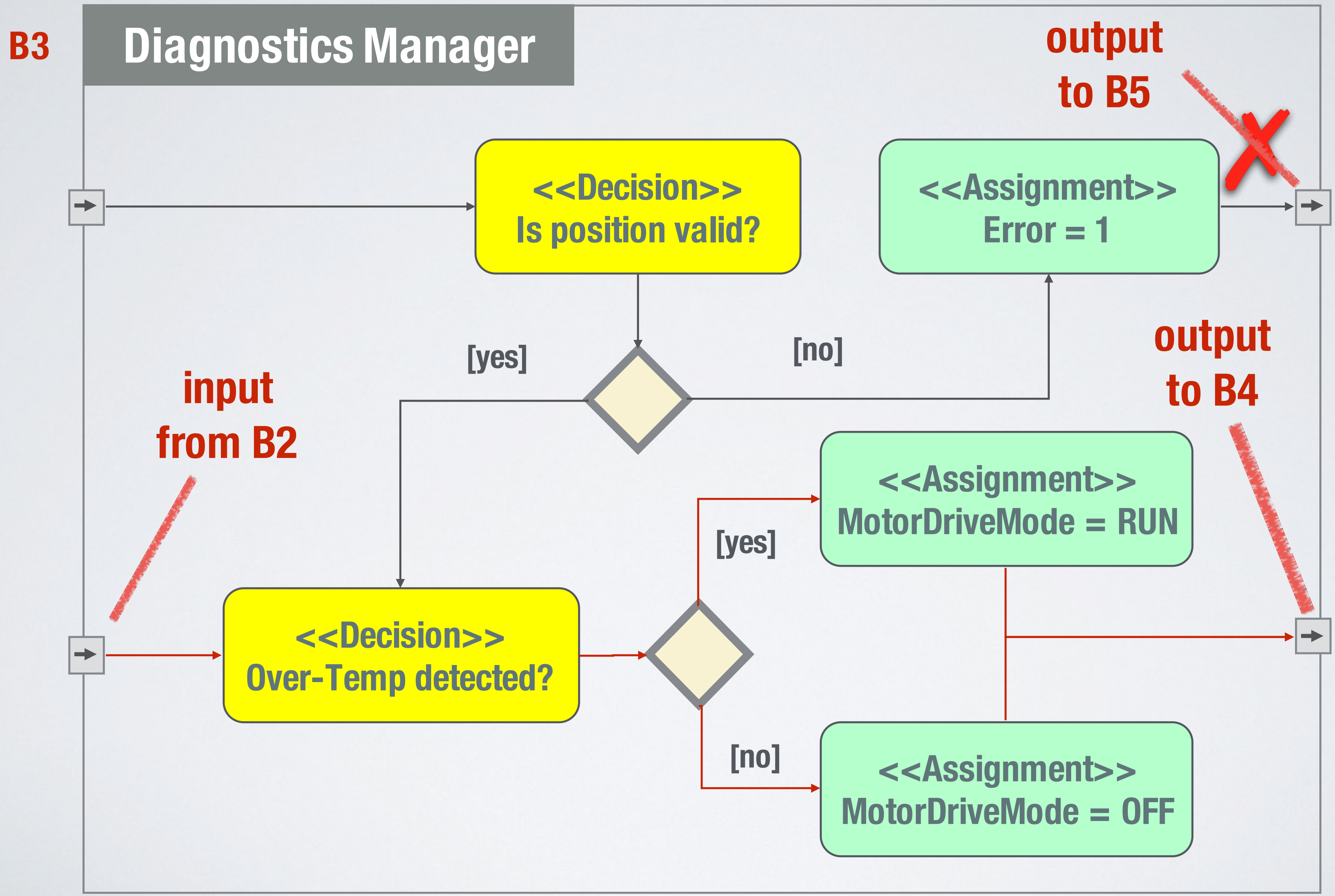


# Behavioural Diagram

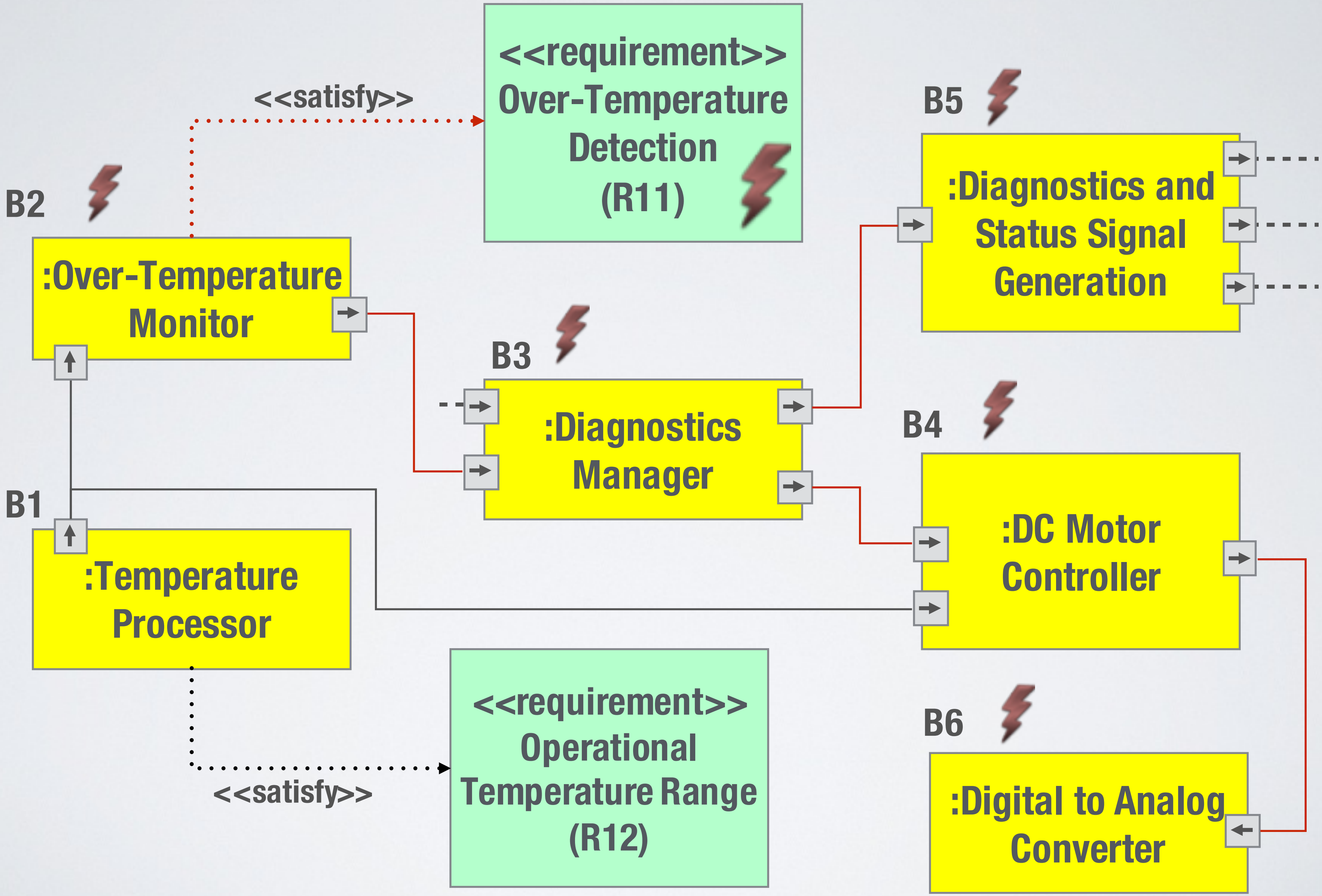
B3



# Behavioural Diagram



# Structural Diagram



# Rank Elements

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

B2, B3, B4, B6

Natural Language Processing Analysis

B2
B6
B3
B4

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

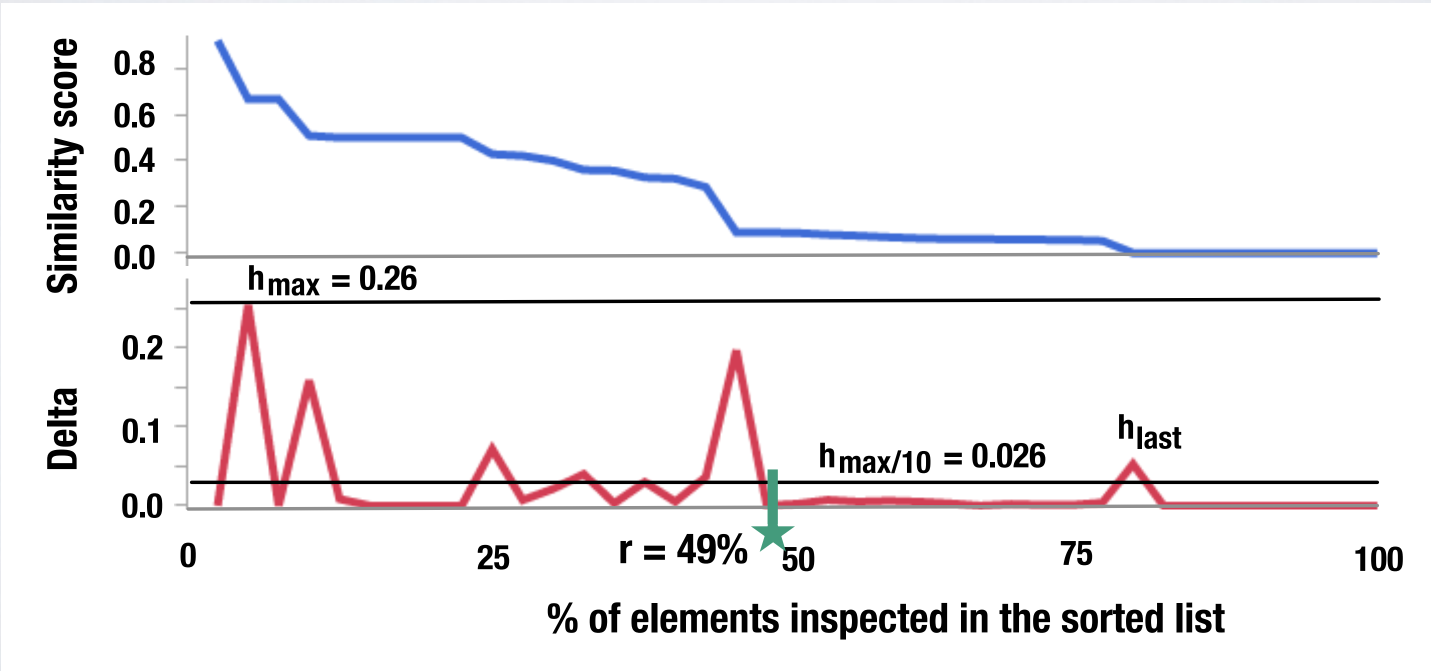
- Computing **similarity scores** for model elements by applying NLP techniques to measure similarity **between model elements labels and change statements.**

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

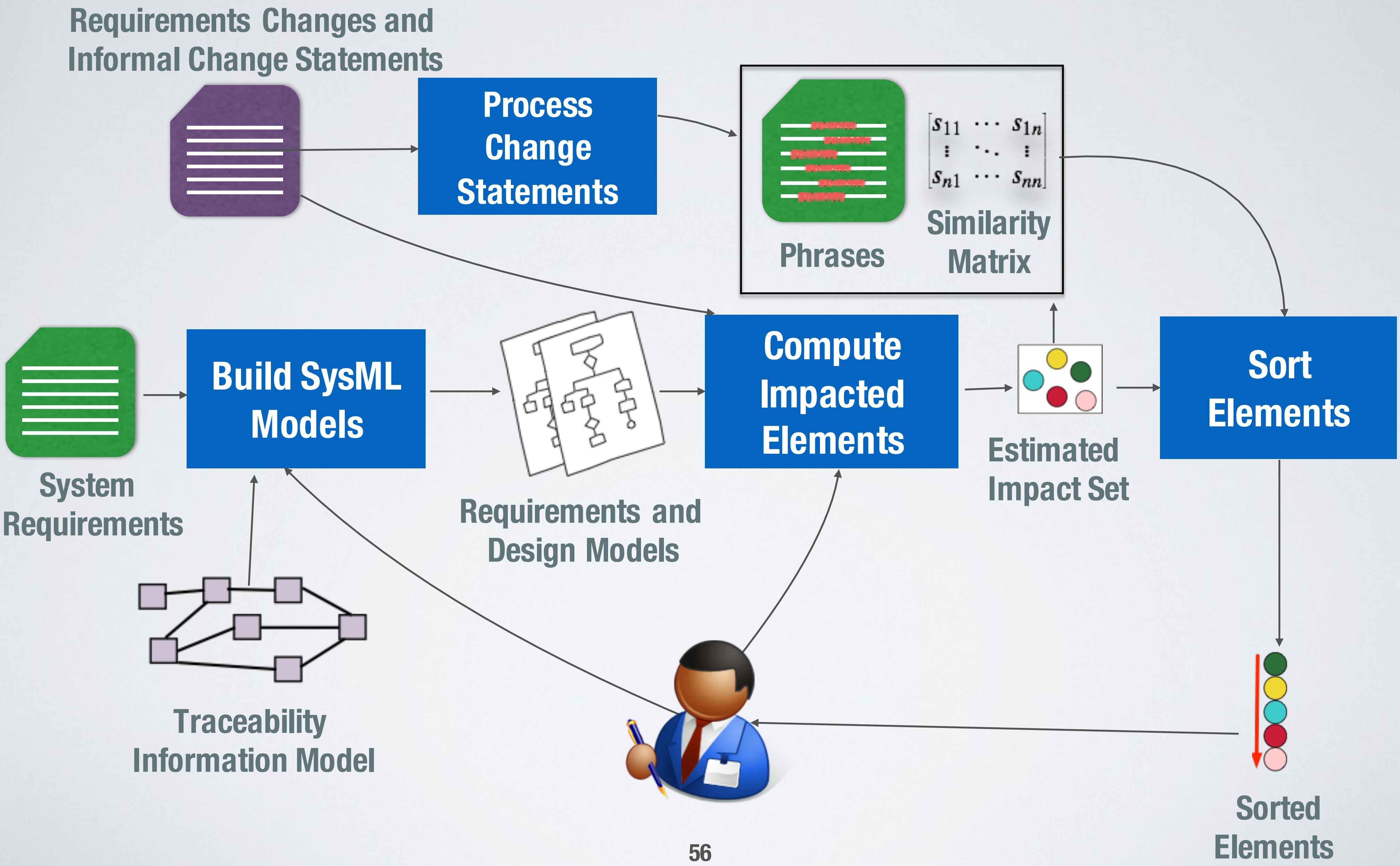
- Engineers inspect the **sorted lists** to identify impacted elements

# Identifying a Subset to Inspect

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



# Approach



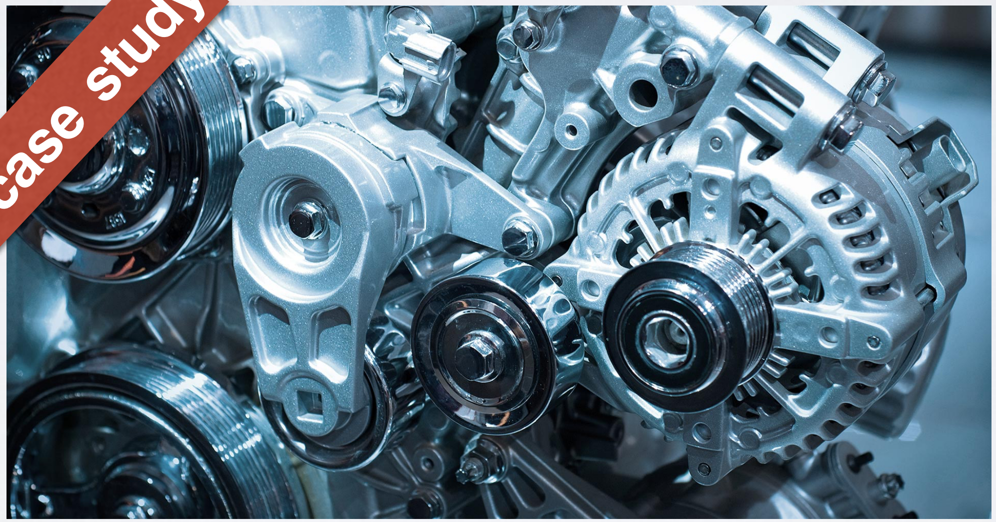
Requirements Changes and
Informal Change Statements
Process Change Statements
Phrases
Similarity Matrix
System Requirements
Build SysML Models
Requirements and Design Models
Compute Impacted Elements
Estimated Impact Set
Sort Elements
Traceability Information Model
Sorted Elements

# Evaluation

DELPHI

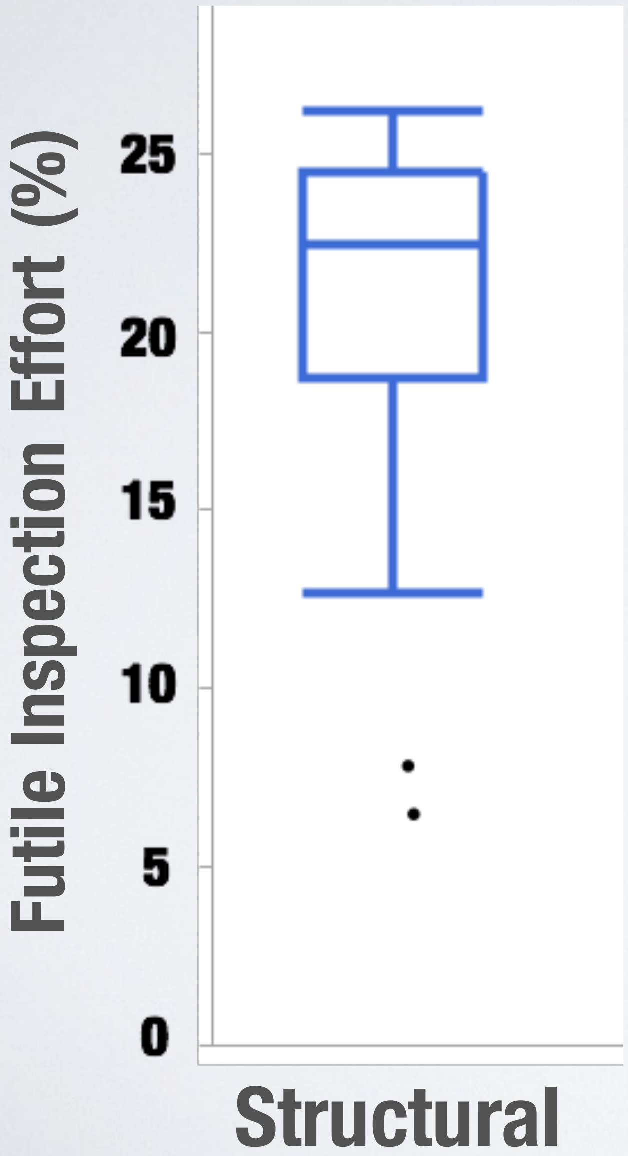
Innovation for the Real World

1 case study

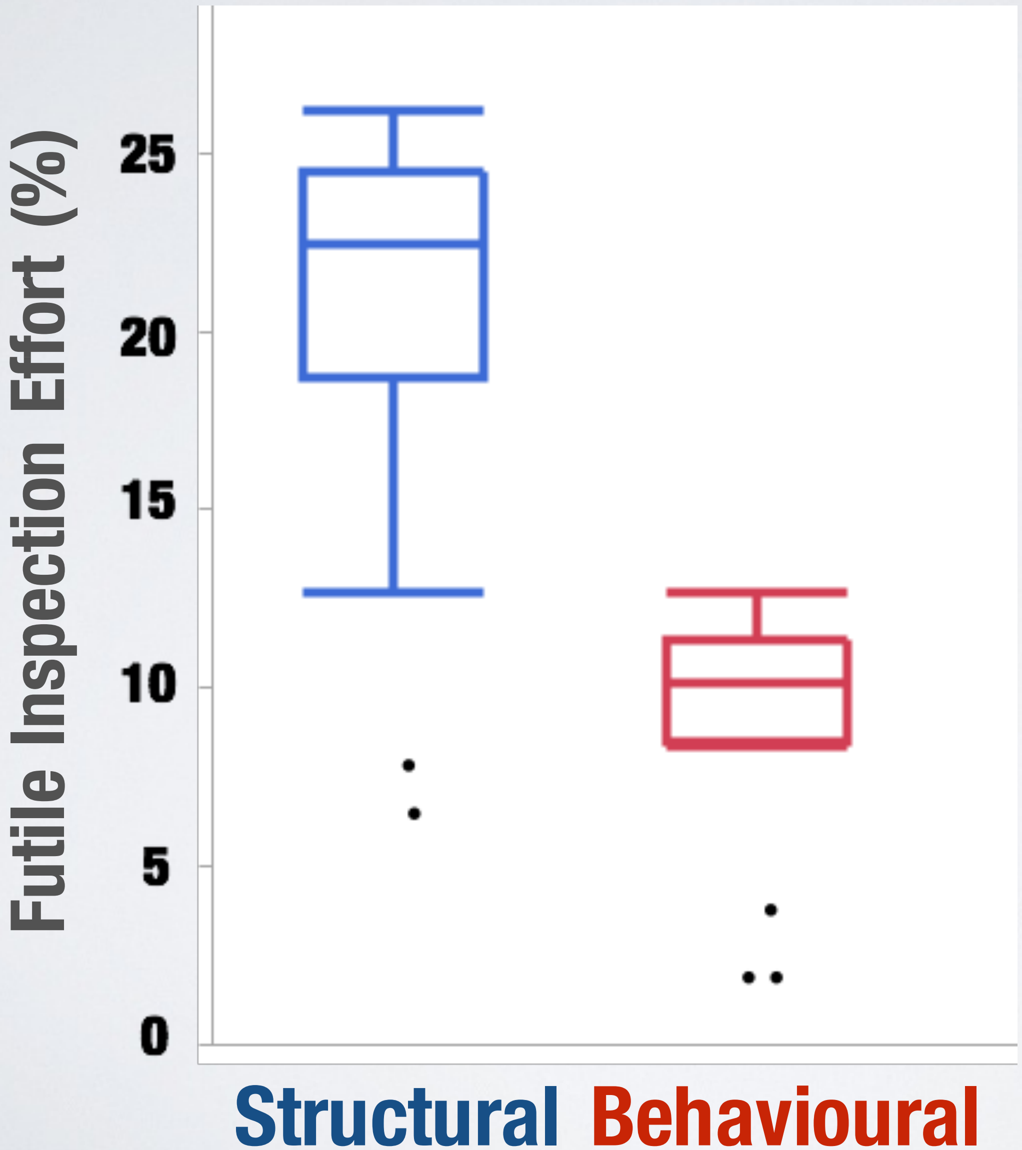


370 elements
16 change scenarios

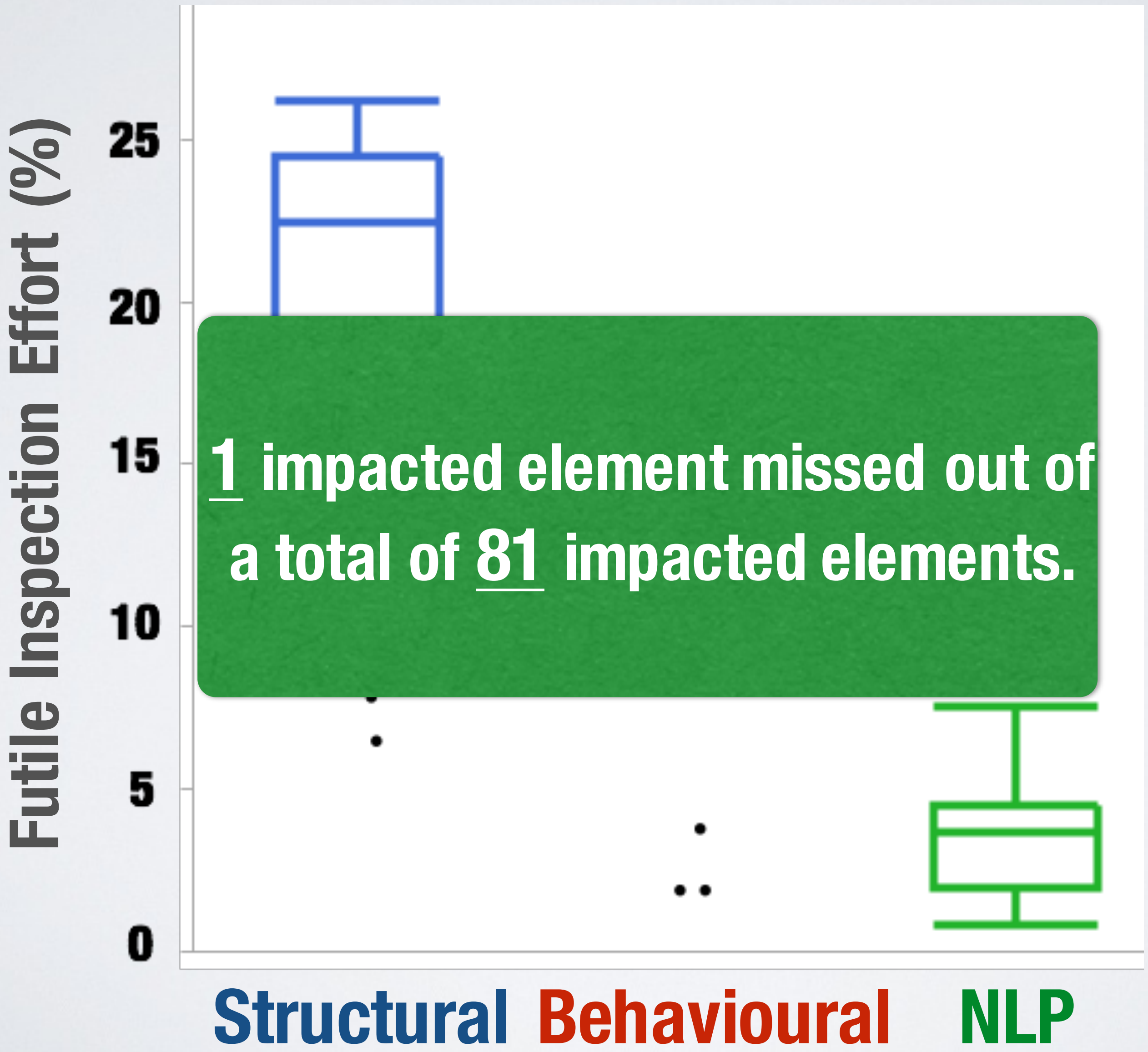
# Effectiveness of Our Approach



# Effectiveness of Our Approach



# 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

## Terms Clustering

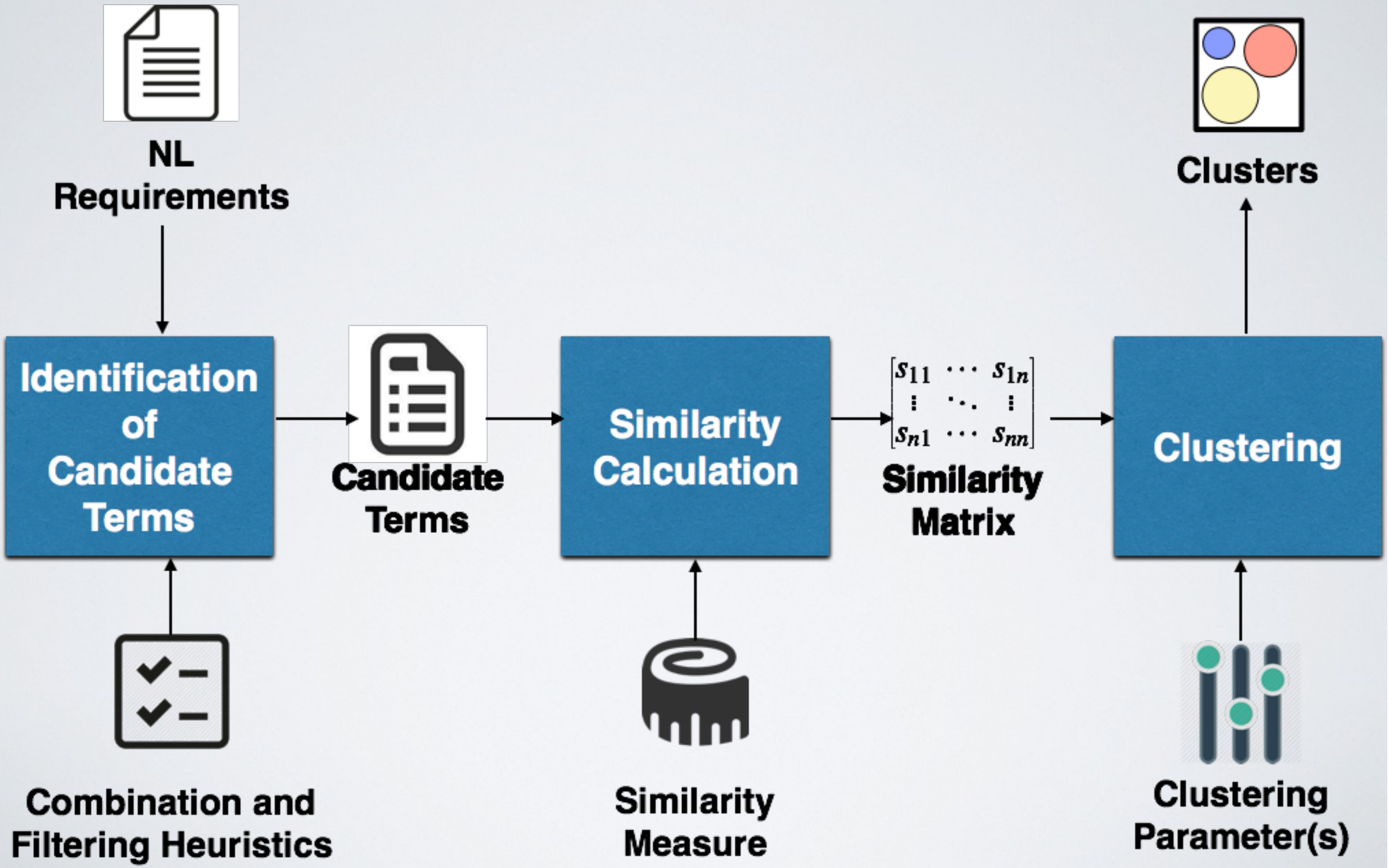
SES
your satellite company

SW.lu
software verification & validation

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.

# Approach


NL
Requirements
Identification of Candidate Terms
Candidate Terms
Similarity Calculation
$\begin{bmatrix} s_{11} & \cdots & s_{1n} \\ \vdots & \ddots & \vdots \\ s_{n1} & \cdots & s_{nn} \end{bmatrix}$
Similarity Matrix
Clustering
Clusters
Combination and
Filtering Heuristics
Similarity
Measure
Clustering
Parameter(s)

Identification of Candidate Terms → Similarity Calculation → Clustering

R1 - STS shall supply GSI monitoring information (GSI input parameters and GSI output parameters) to the STS subcontractor.

0.85

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

SES
your satellite company

2 case studies

380 Requirements
138 Requirements

OPENCOSS

1 case study


110 Requirements

# Results

ΔRecall > 20%

Our Approach

VS

JATE

TOPIA

TextRank

TermoStat

TermRaider

# Results

Precision ~

Our Approach

VS

JATE

TOPIA

TextRank

TermoStat

TermRaider

# Clustering Evaluation

SES your satellite company

OPENCOSS

2 case studies

1 case study


- **Interview Survey**

**20 clusters each case study**

**27 clusters**

# 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)

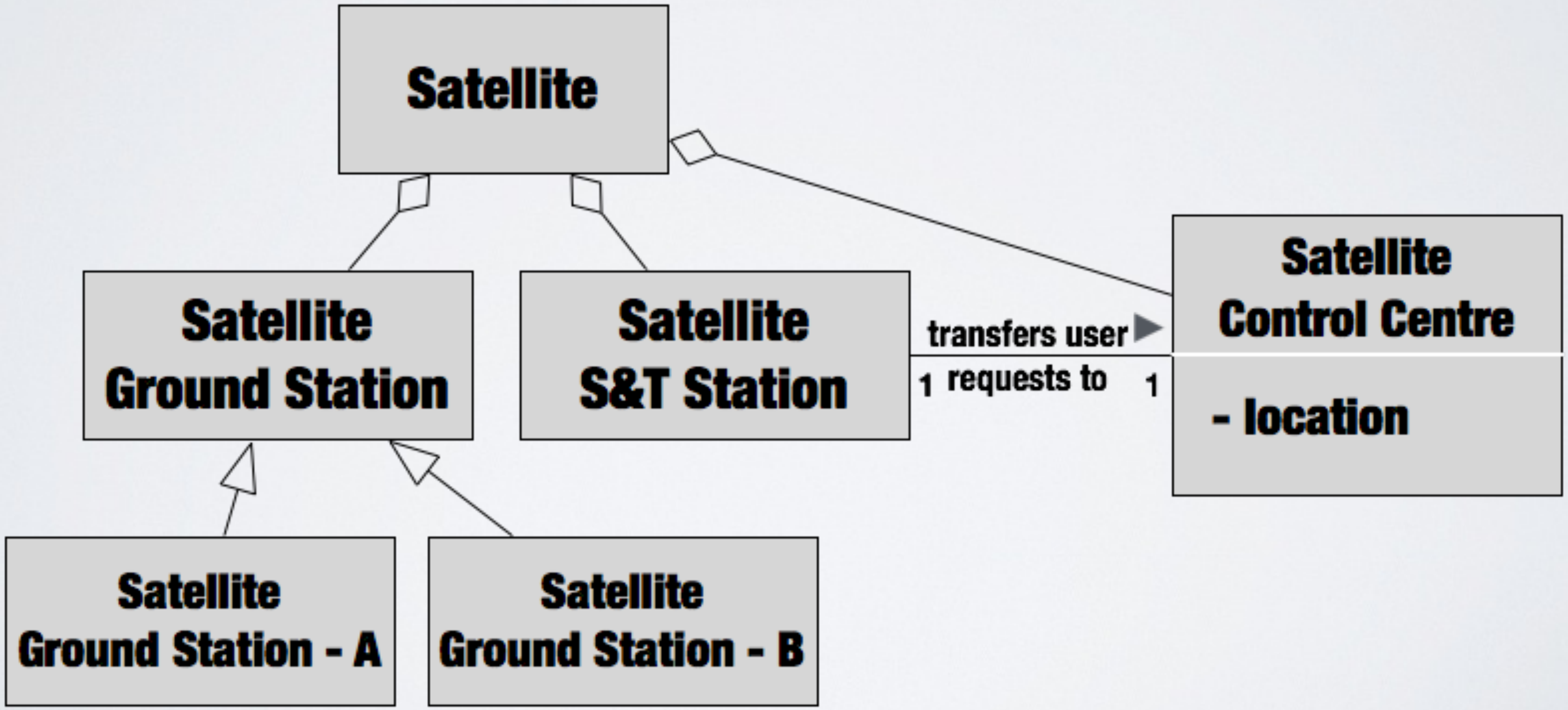
# Domain Model Extraction

# Motivation

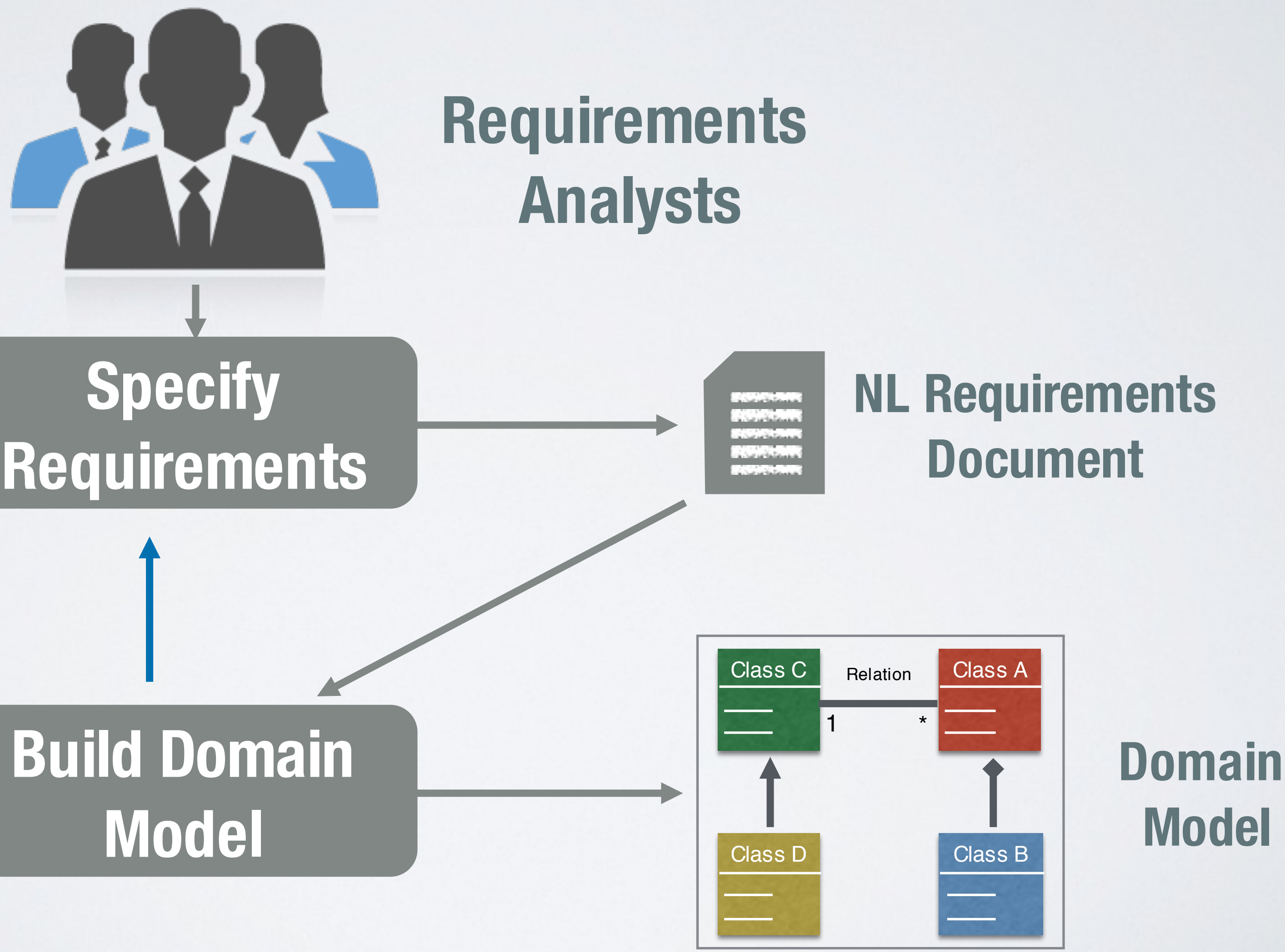
- Representation of important domain concepts and their relations
  - Facilitate communication between stakeholders from different backgrounds
  - Help identify inconsistencies in terminology, etc.
- In practice, domain models are not preceding the elicitation and writing of requirements

# Domain Models

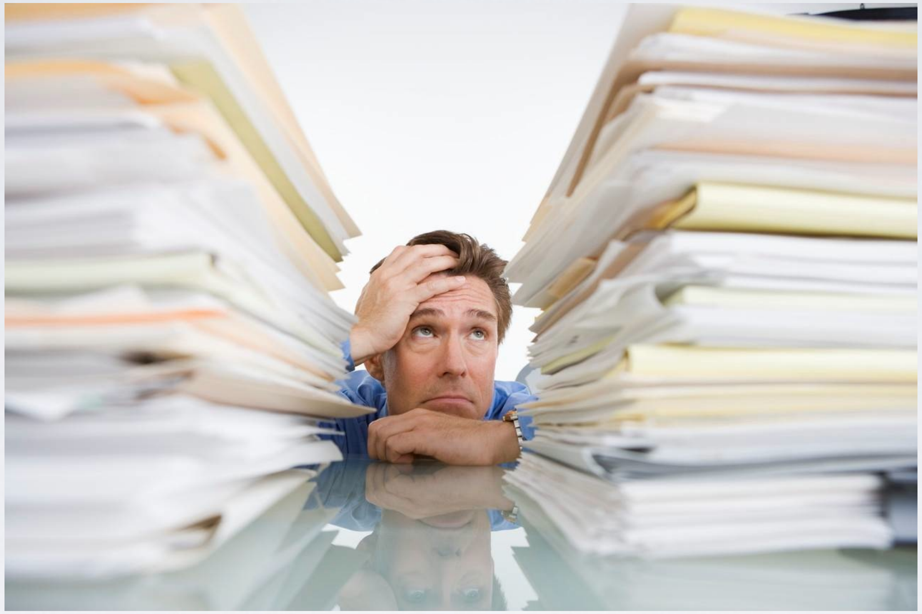
A domain model is a representation of conceptual entities or real-world objects in a domain of interest.



# Context


Requirements
Analysts
Specify
Requirements
NL Requirements
Document
Build Domain
Model
Class C
Relation
Class A
1
*
Class D
Class B
Domain
Model

# Problem Definition

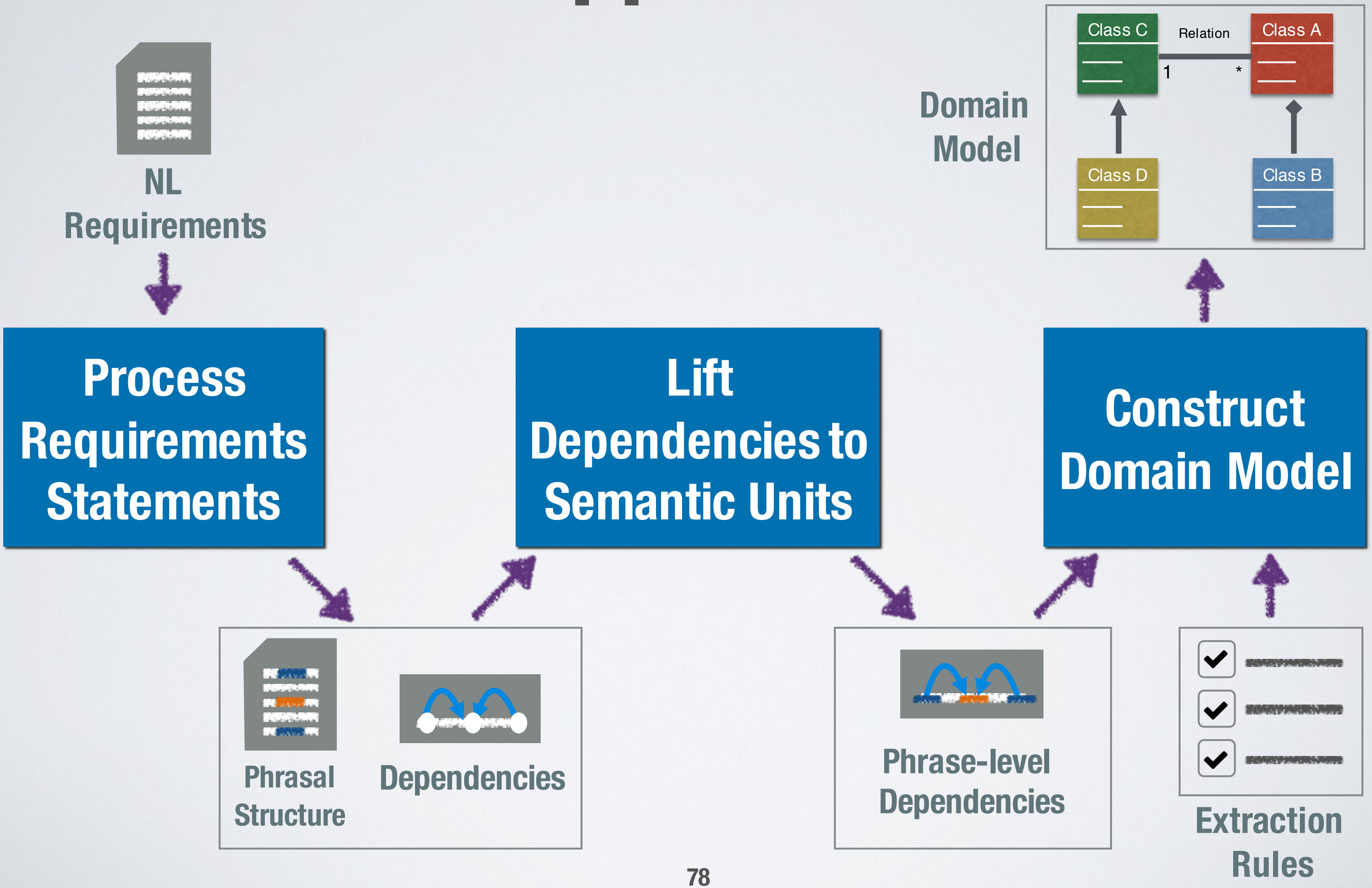


- Manually building domain models is laborious
- Automated support is required for building domain models

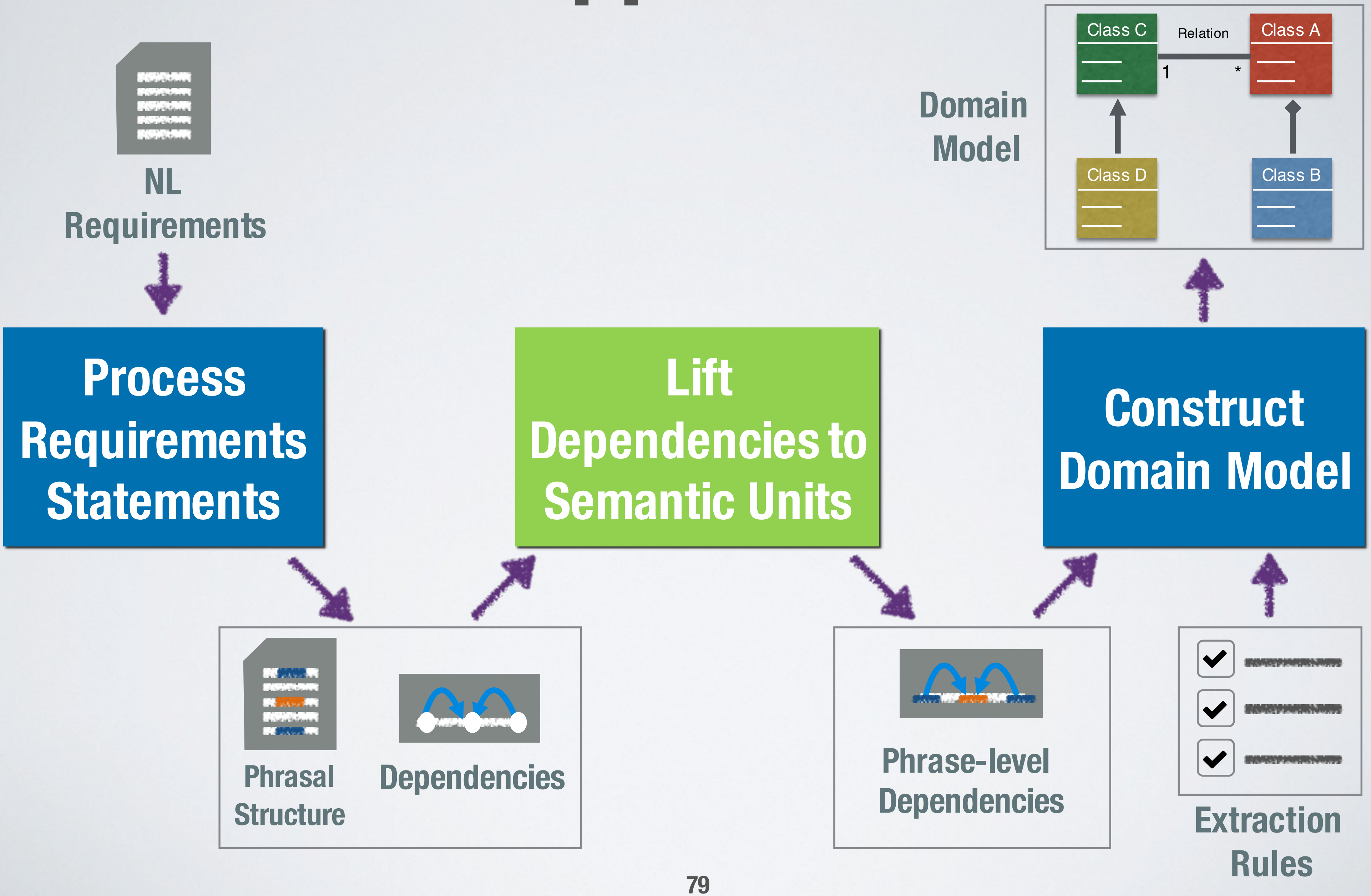
# State of the Art

- **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

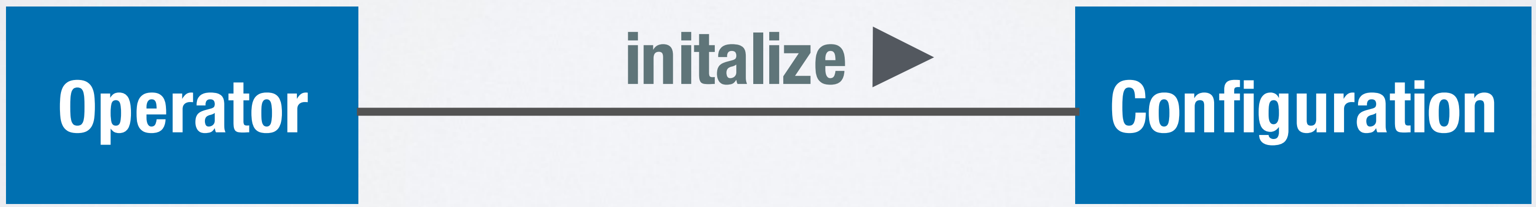
# Approach


NL Requirements
Process Requirements Statements
Phrasal Structure
Dependencies
Lift Dependencies to Semantic Units
Phrase-level Dependencies
Construct Domain Model
Extraction Rules
Domain Model
Class C
Relation
Class A
1
*
Class D
Class B

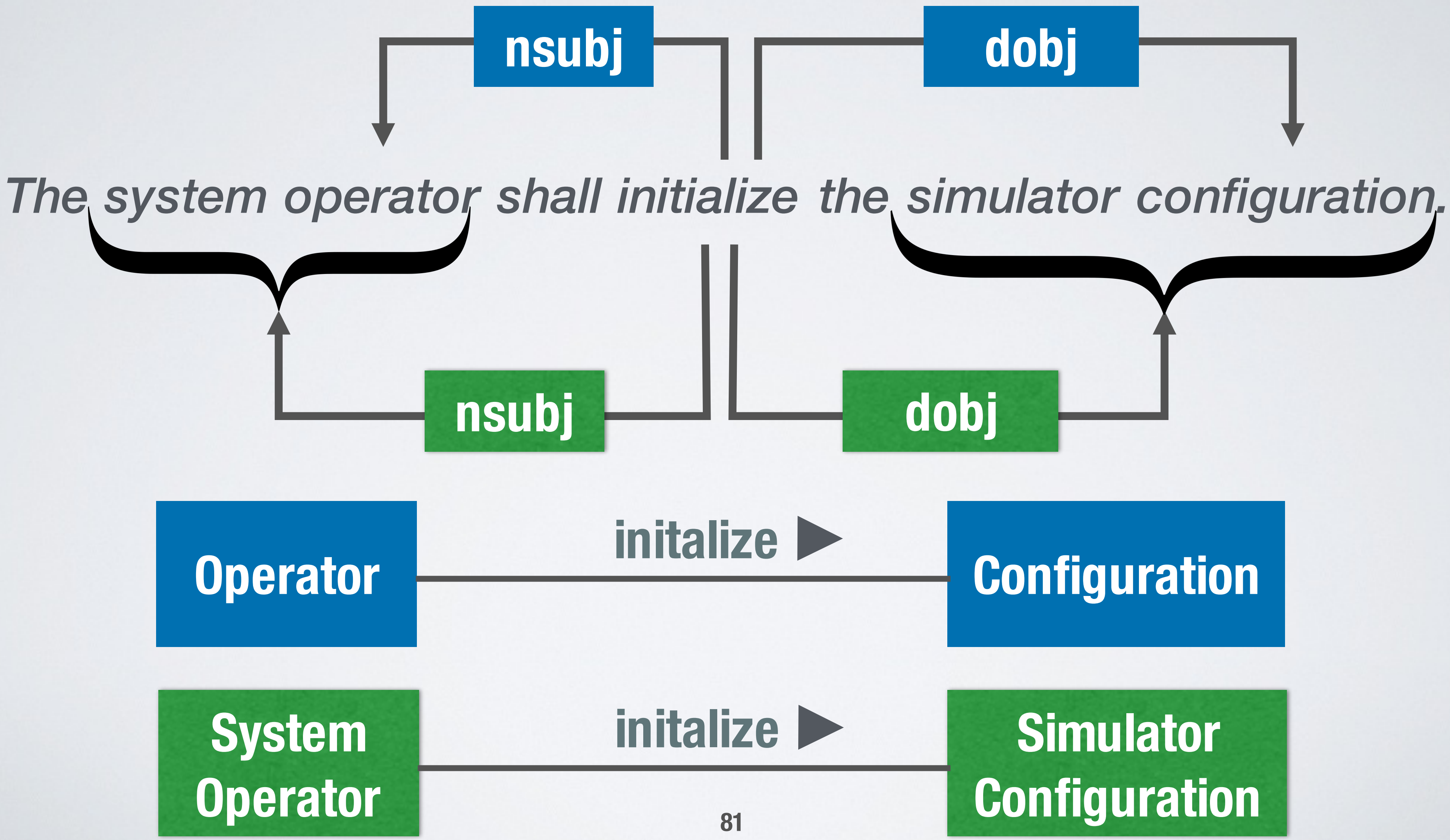
# Approach


NL Requirements
Domain Model
Class C
Relation
Class A
1
*
Class D
Class B
Process Requirements Statements
Lift Dependencies to Semantic Units
Construct Domain Model
Phrasal Structure
Dependencies
Phrase-level Dependencies
Extraction Rules

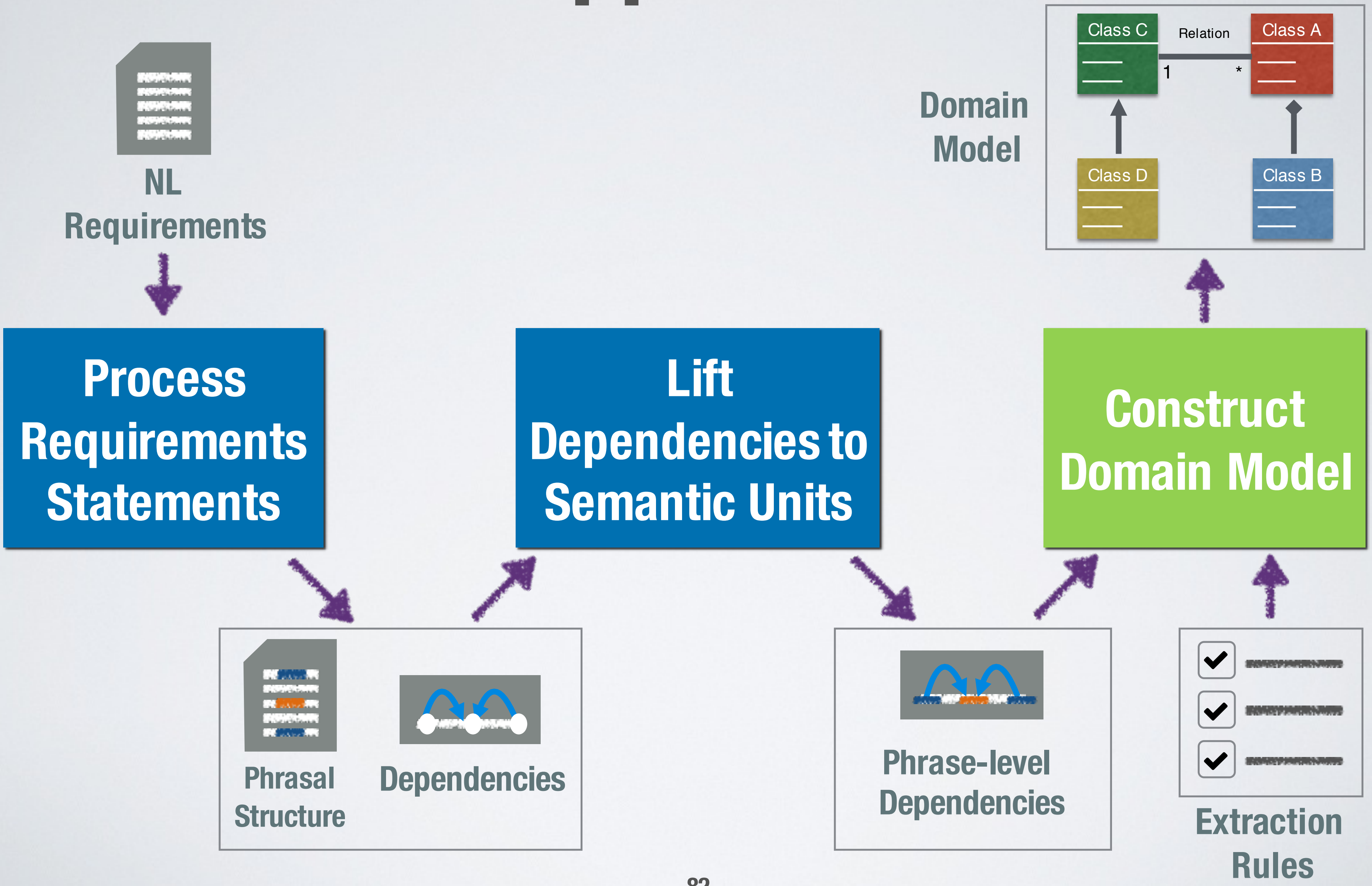
# Grammatical Dependencies

# Lift Dependencies to Semantic Units

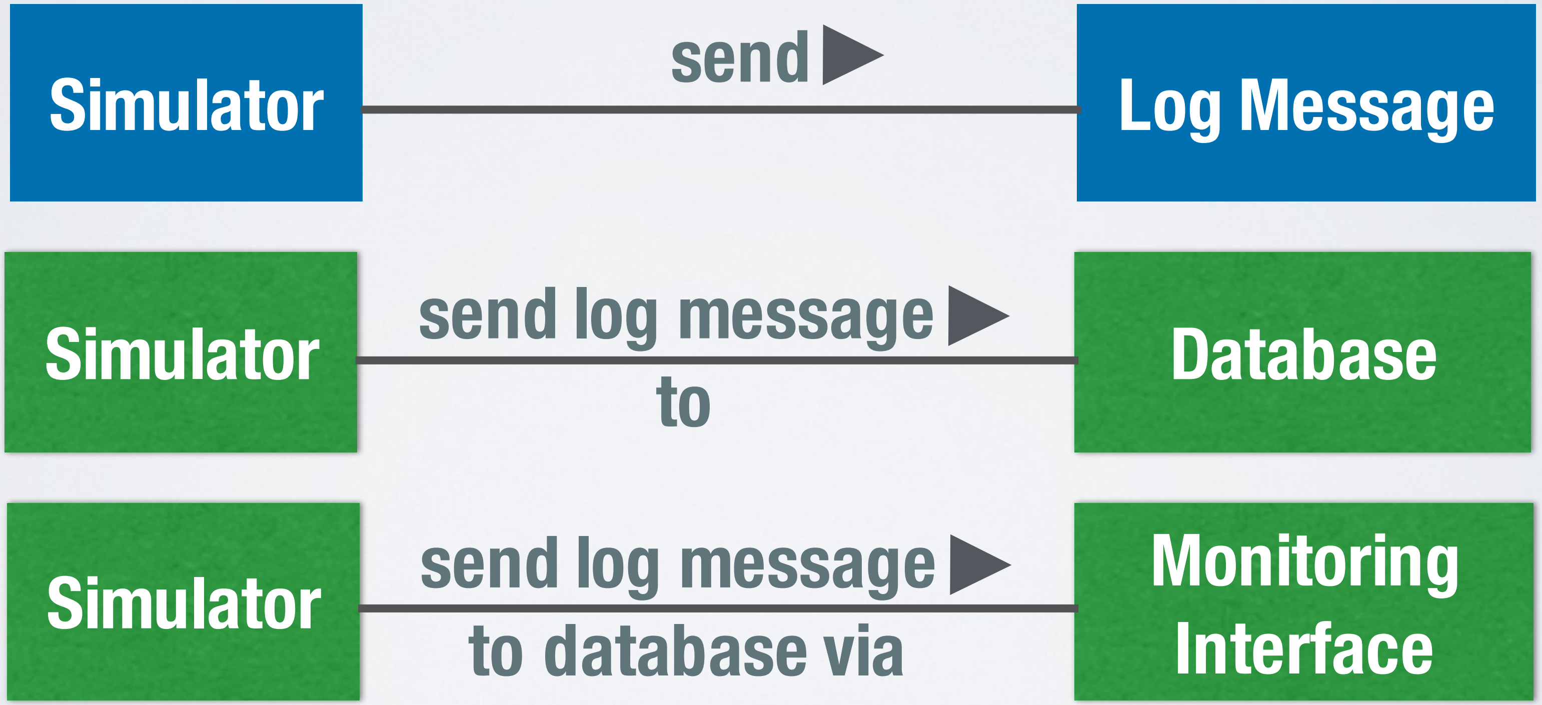


# Approach


NL Requirements
Process Requirements Statements
Phrasal Structure
Dependencies
Lift Dependencies to Semantic Units
Phrase-level Dependencies
Construct Domain Model
Extraction Rules
Domain Model
Class C
Relation
Class A
1
*
Class D
Class B

# Link Paths

***The simulator shall send log messages to the database via the monitoring interface.***



# How useful is our approach?

1 case study

50 Requirements
213 Relations

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

# Results

Correctness- 90% (avg.)

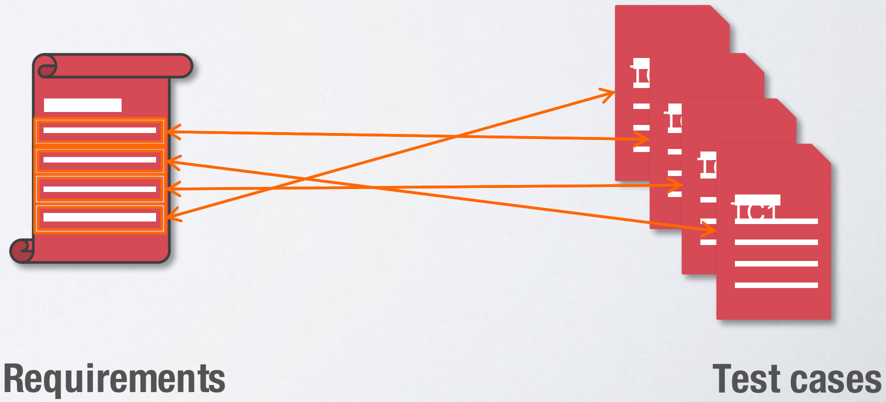
Relevance- 36% (avg.)

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**



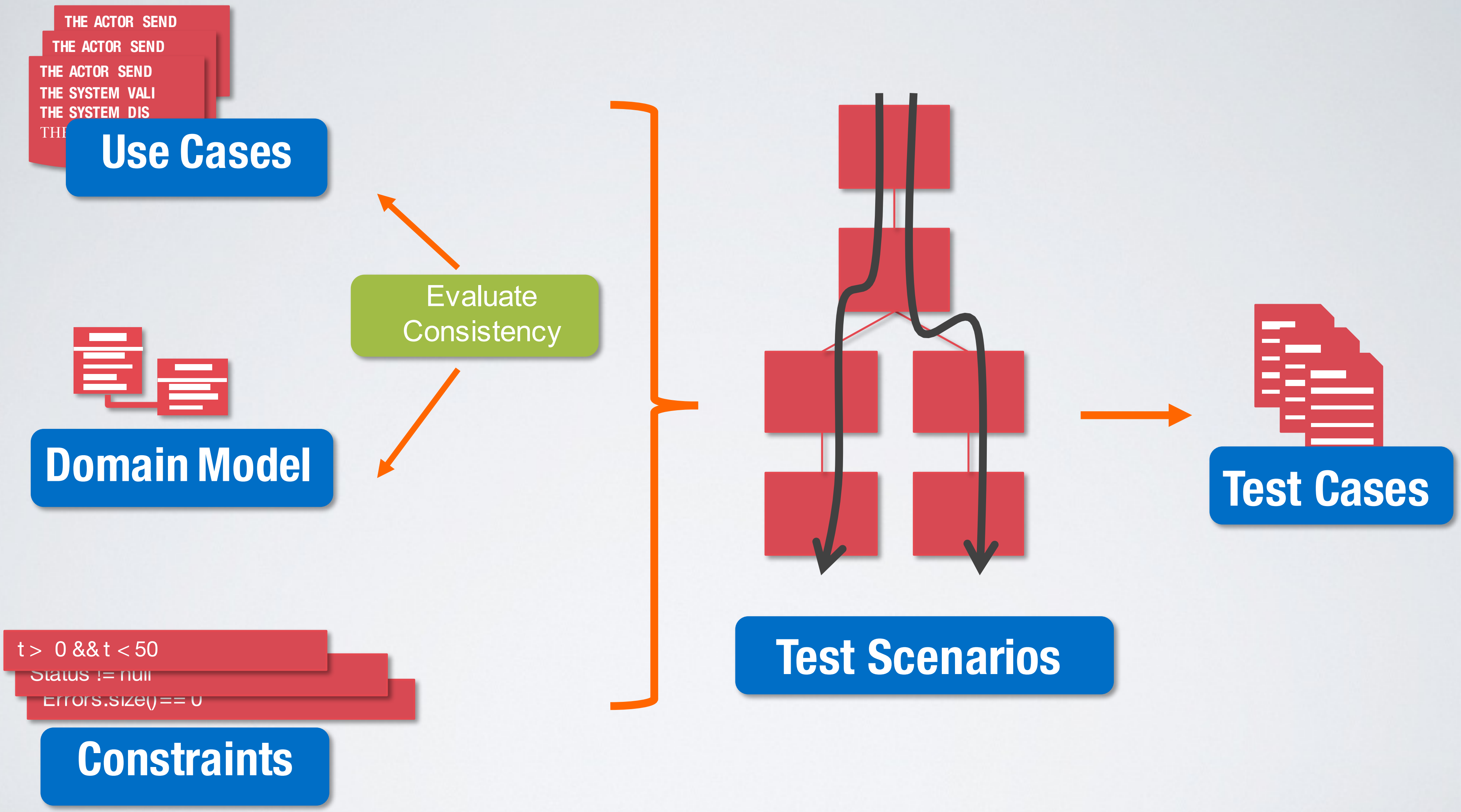
# Objectives

- 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

- **Analyzable** use case specifications
- Automatically extract test model from the use case specifications (**Natural Language Processing**)
- **Minimize modeling**, domain modeling only
- **No behavioral modeling**

# UMTG


THE ACTOR SEND
THE ACTOR SEND
THE ACTOR SEND
THE SYSTEM VALI
THE SYSTEM DIS
Use Cases
Evaluate Consistency
Domain Model
t > 0 && t < 50
Status != null
Errors.size() == 0
Constraints
Test Scenarios
Test Cases

# RUCM

[Yue et al. TOSEM'13]

**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.

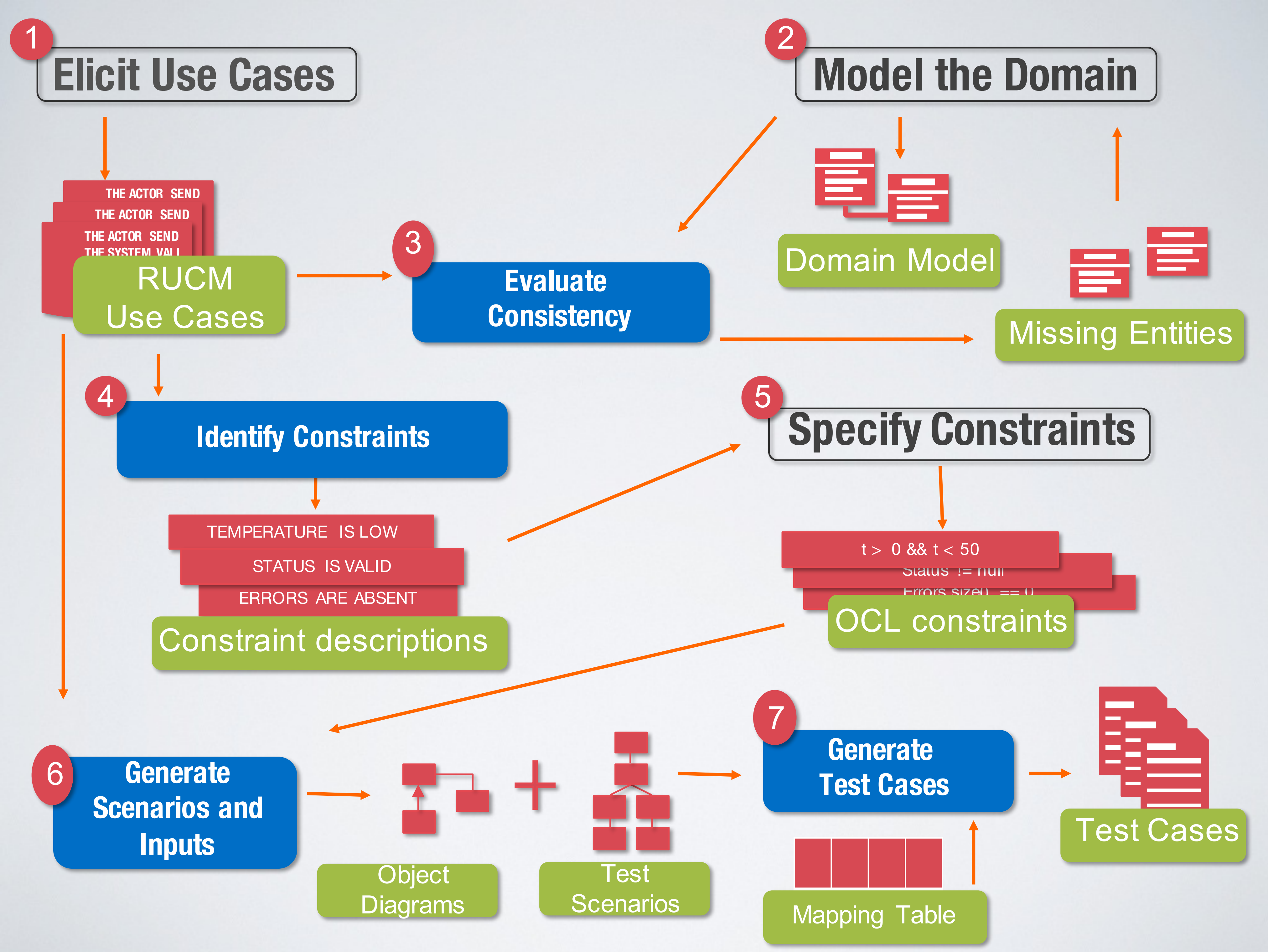
2. INCLUDE USE CASE Classify occupancy status.
3. The system VALIDATES THAT the occupant class for airbag control is valid.
4. The system SENDS the occupant class for airbag control TO AirbagControlUnit.

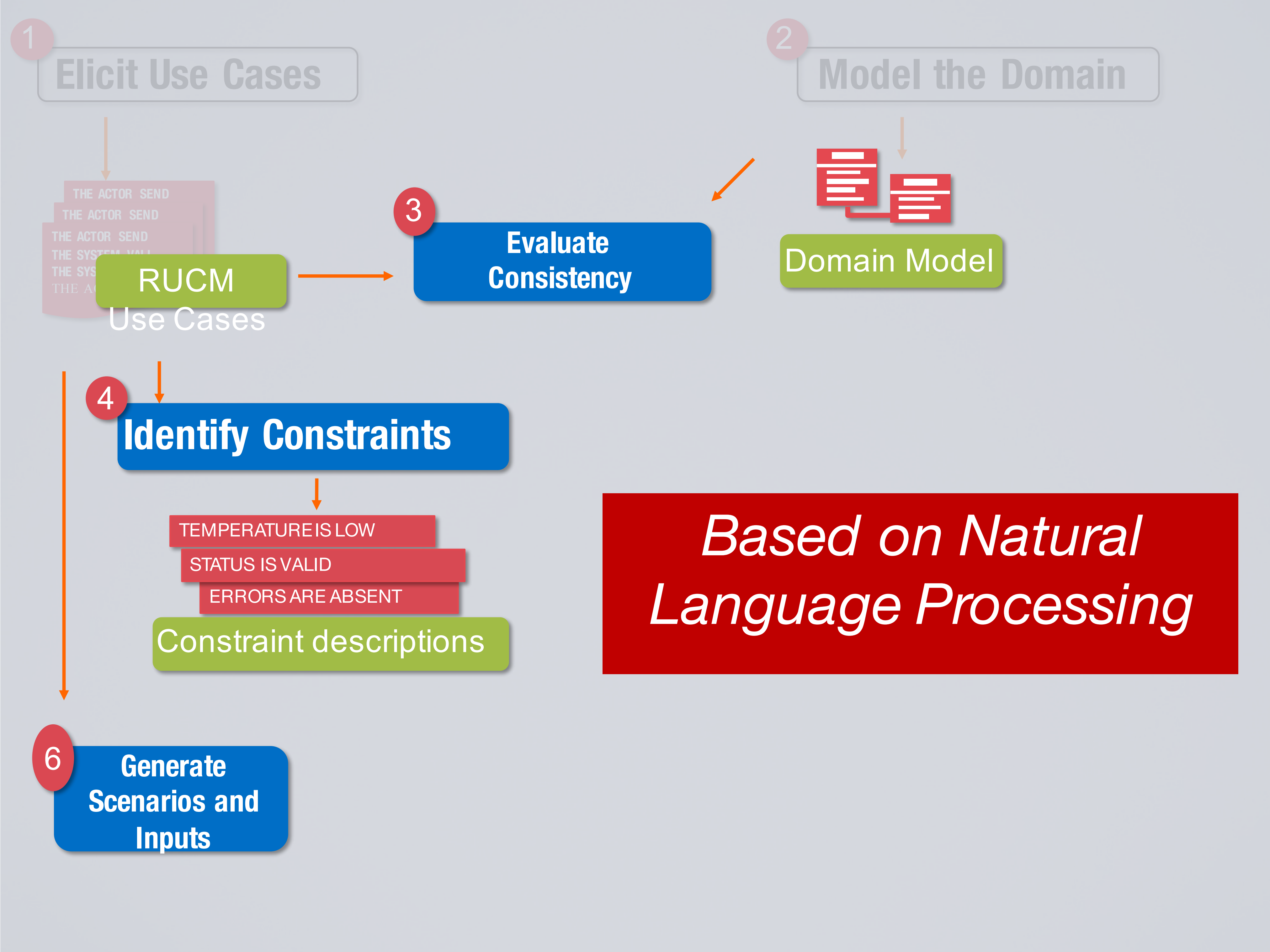
## Specific Alternative Flow

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


1
Elicit Use Cases
THE ACTOR SEND
THE ACTOR SEND
THE ACTOR SEND
RUCM Use Cases
2
Model the Domain
Domain Model
Missing Entities
3
Evaluate Consistency
4
Identify Constraints
TEMPERATURE IS LOW
STATUS IS VALID
ERRORS ARE ABSENT
Constraint descriptions
5
Specify Constraints
t > 0 && t < 50
OCL constraints
6
Generate Scenarios and Inputs
Object Diagrams
Test Scenarios
7
Generate Test Cases
Mapping Table
Test Cases



1
Elicit Use Cases
2
Model the Domain
THE ACTOR SEND
THE ACTOR SEND
THE ACTOR SEND
RUCM
Use Cases
3
Evaluate Consistency
Domain Model
4
Identify Constraints
TEMPERATURE IS LOW
STATUS IS VALID
ERRORS ARE ABSENT
Constraint descriptions
Based on Natural Language Processing
6
Generate Scenarios and Inputs

Basic Flow

1. The seat **SENDS** occupancy status **TO** the system. → INPUT STEP
   DOMAIN ENTITY

2. **INCLUDE USE CASE** Classify occupancy status. → INCLUDE STEP

3. The system **VALIDATES THAT** → CONDITIONAL STEP

   the occupant class for airbag control is valid and
   CONSTRAINT

   the occupant class for seat belt reminder is valid.
   CONSTRAINT

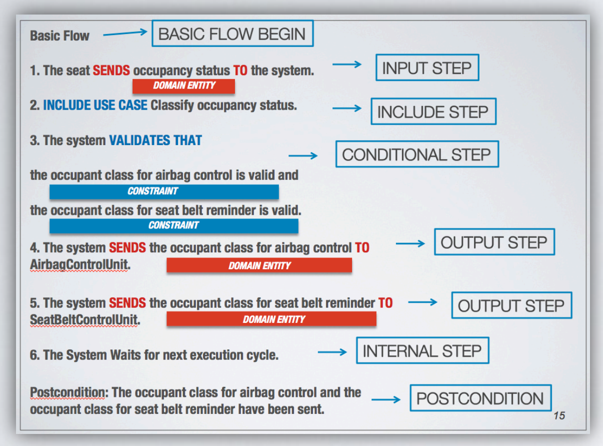
4. The system **SENDS** the occupant class for airbag control **TO** AirbagControlUnit. → OUTPUT STEP
   DOMAIN ENTITY

5. The system **SENDS** the occupant class for seat belt reminder **TO** SeatBeltControlUnit. → OUTPUT STEP
   DOMAIN ENTITY

6. The System Waits for next execution cycle. → INTERNAL STEP

Postcondition: The occupant class for airbag control and the occupant class for seat belt reminder have been sent. → POSTCONDITION

# Evaluate Model Consistency



Basic Flow → BASIC FLOW BEGIN

1. The seat SENDS occupancy status TO the system. → INPUT STEP
DOMAIN ENTITY

2. INCLUDE USE CASE Classify occupancy status. → INCLUDE STEP

3. The system VALIDATES THAT → CONDITIONAL STEP

the occupant class for airbag control is valid and
CONSTRAINT

the occupant class for seat belt reminder is valid.
CONSTRAINT

4. The system SENDS the occupant class for airbag control TO AirbagControlUnit. → OUTPUT STEP
DOMAIN ENTITY

5. The system SENDS the occupant class for seat belt reminder TO SeatBeltControlUnit. → OUTPUT STEP
DOMAIN ENTITY

6. The System Waits for next execution cycle. → INTERNAL STEP

Postcondition: The occupant class for airbag control and the occupant class for seat belt reminder have been sent. → POSTCONDITION

15

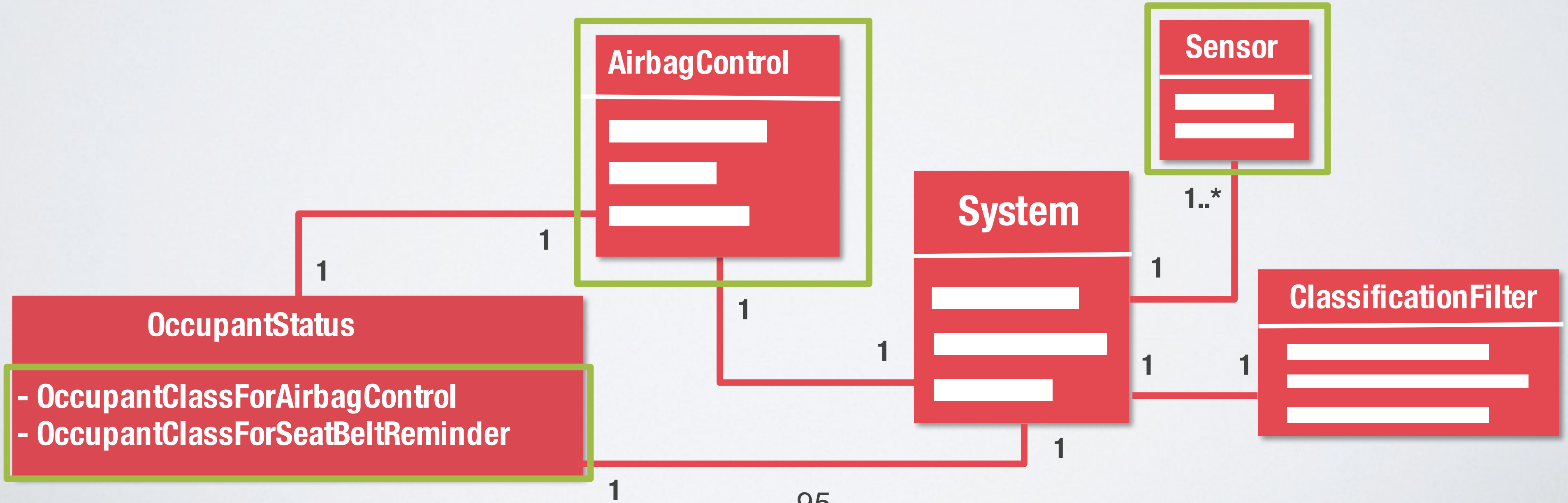
**Tagged Use Case**

Airbag Control ✓ | Classification Filter ? | Sensor ✓

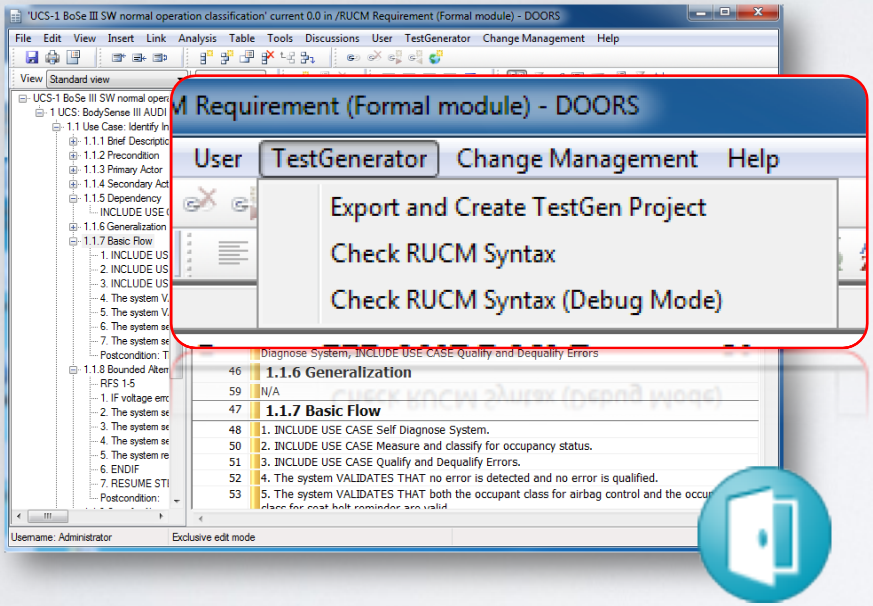
Occupant Class for Airbag Control ✓

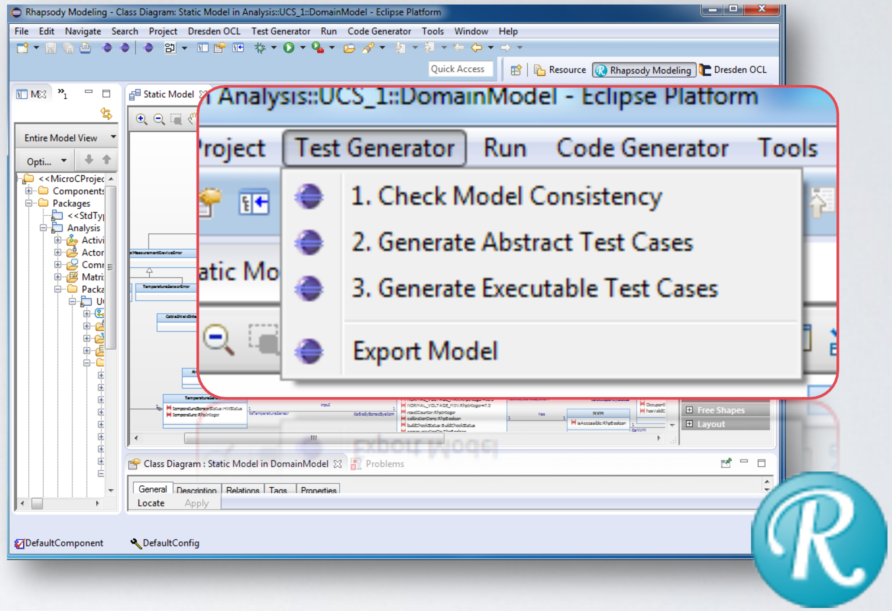
Occupant Class for Seat Belt Reminder ✓

**Domain Entities**



# 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



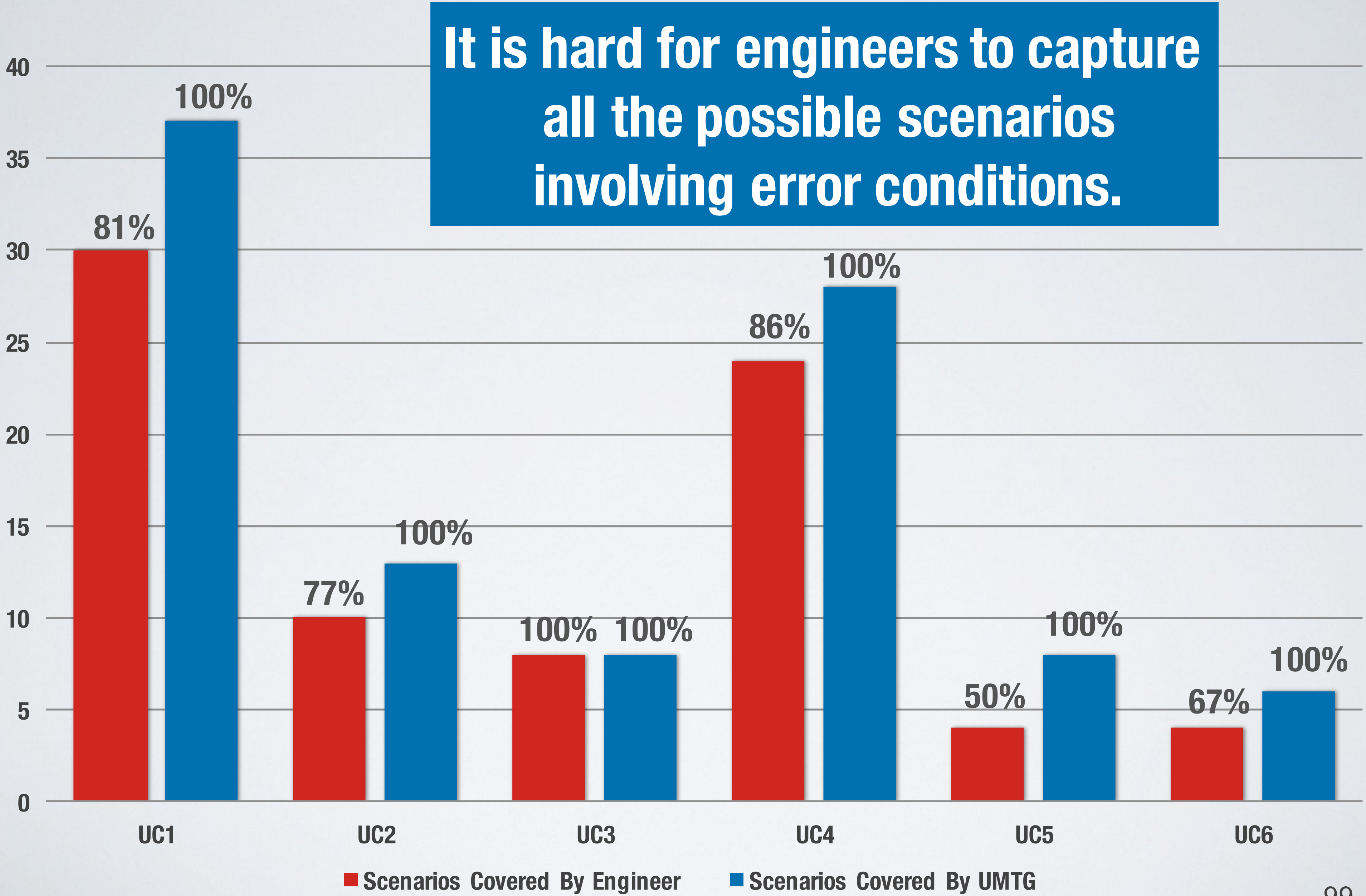
- Evaluation:
  - Cost of additional modelling
  - Effectiveness in terms of covered scenarios compared to current practice at IEE
  - Keep in mind changes and repeated testing

# Costs of Additional Modeling

| Use Case | Steps | Use Case Flows | OCL Constraints |
|---|---|---|---|
| UC1 | 50 | 8 | 9 |
| UC2 | 44 | 13 | 7 |
| UC3 | 35 | 8 | 8 |
| UC4 | 59 | 11 | 12 |
| UC5 | 30 | 8 | 5 |
| UC6 | 25 | 6 | 12 |

**5 to 10 minutes to write each constraints**
**=> A maximum of 10 hours in total**

# Effectiveness: scenarios covered



# 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

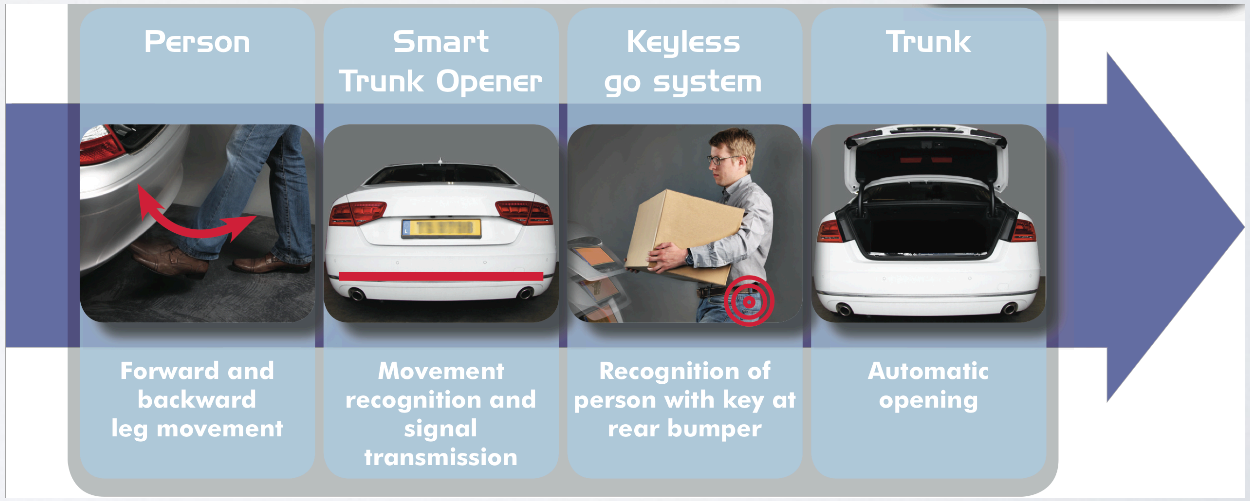
28 February 2017

# Context

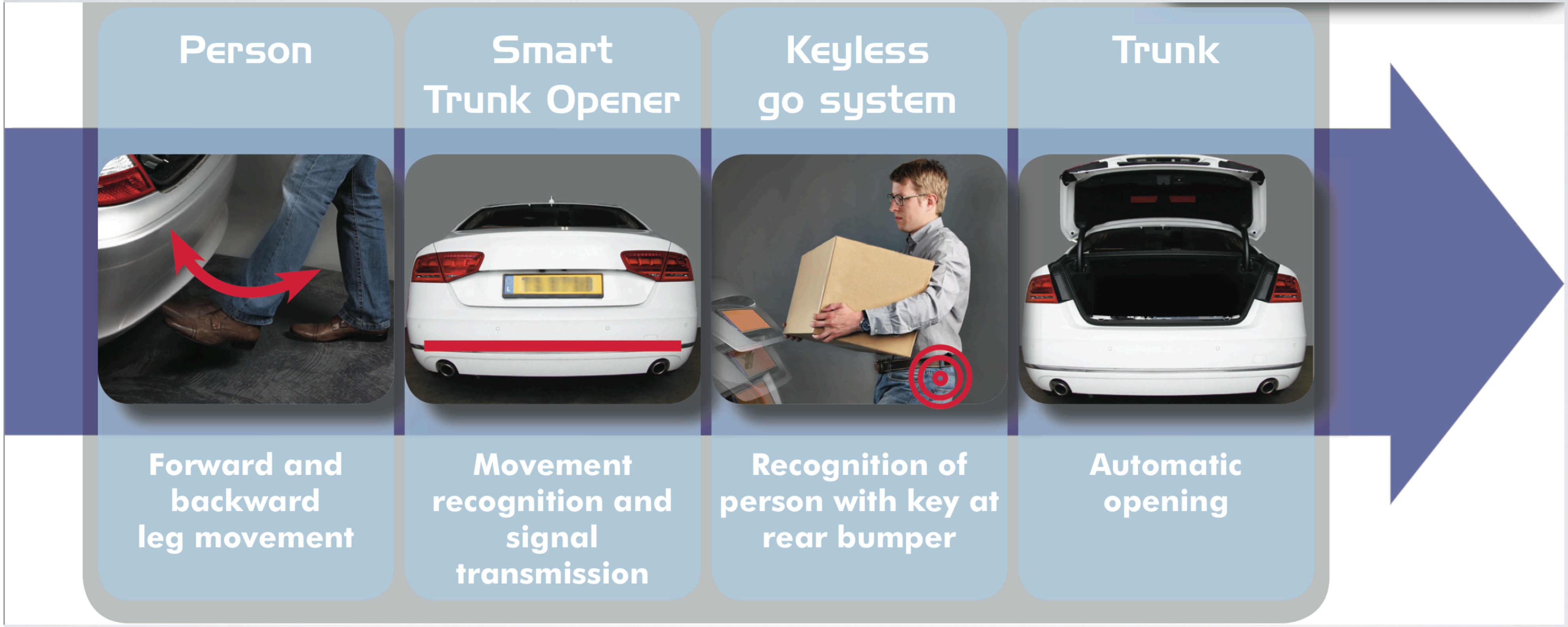
IEE develops real-time embedded systems:

- Automotive **safety sensing systems**
- Automotive **comfort & convenience** systems, e.g., Smart Trunk Opener

International Electronics & Engineering (IEE)



# Smart Trunk Opener (STO)

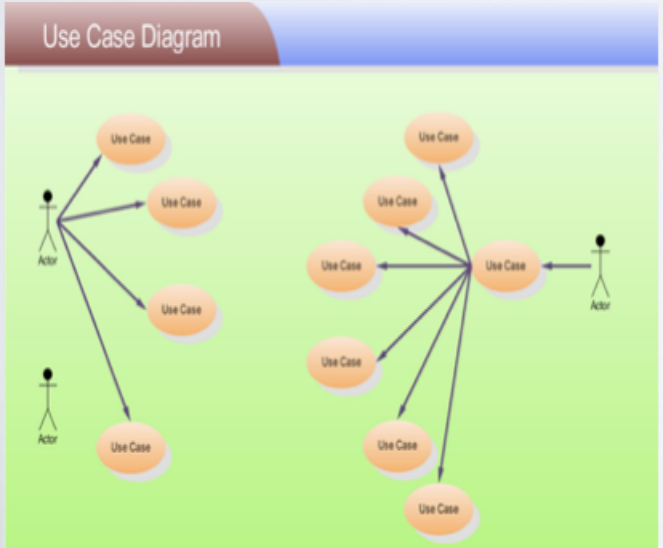



**STO Provides automatic and hands-free access to a vehicle's trunk (based on a keyless entry system)**

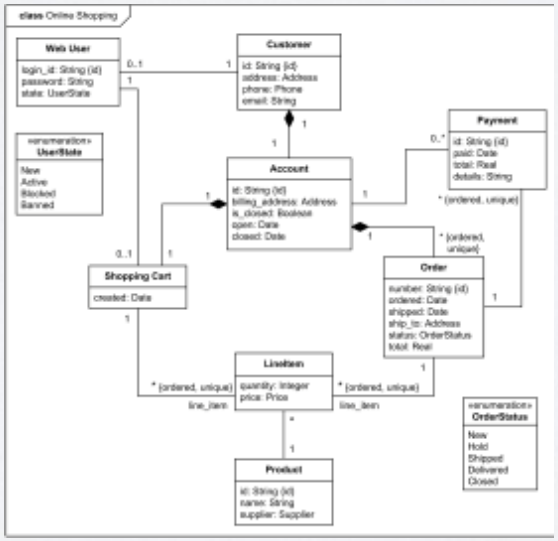
# IEE Requirements Engineering

**Use Case Driven Development**

## Use Case Diagram



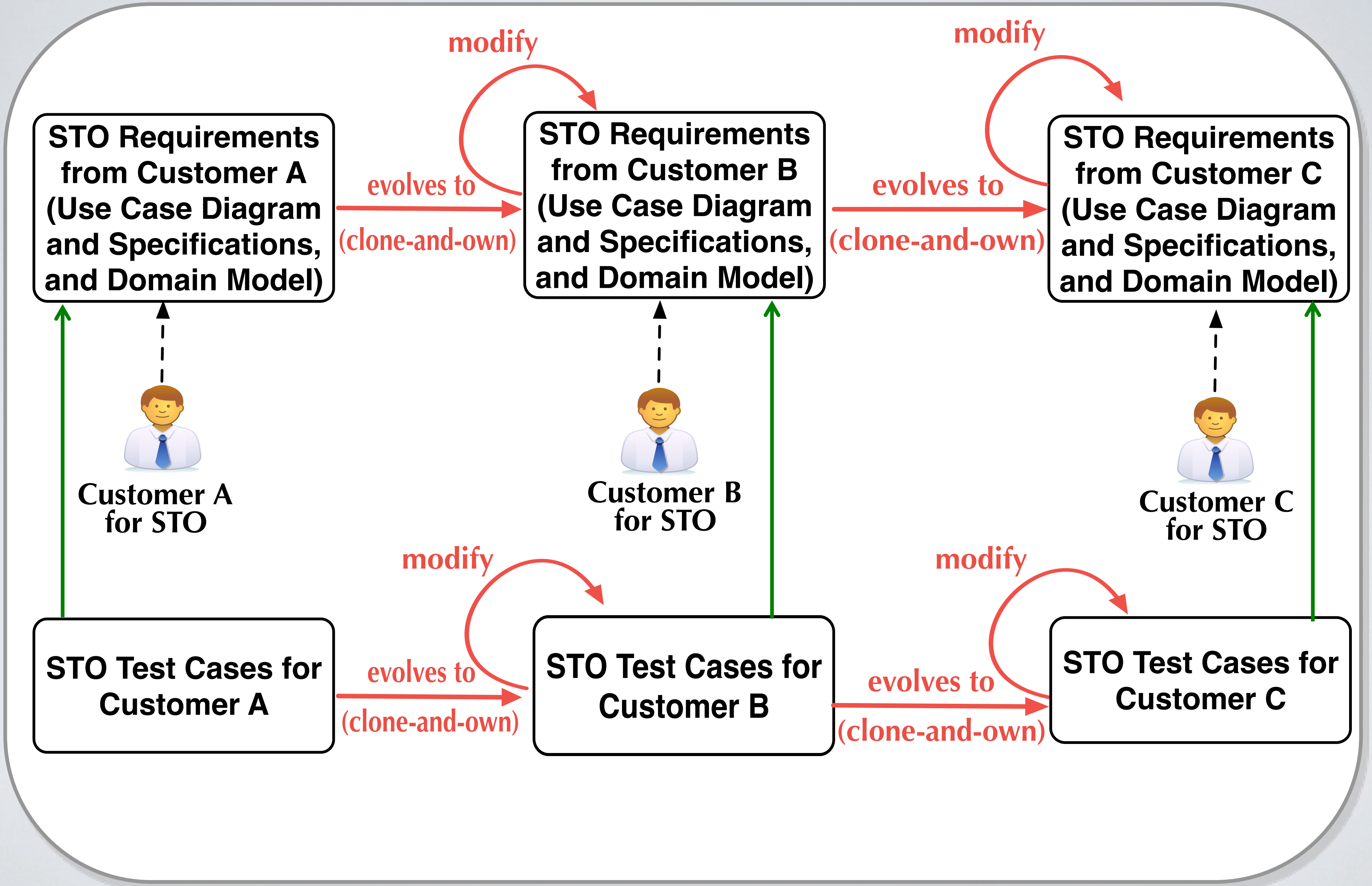

## Domain Model




## Use Case Specifications

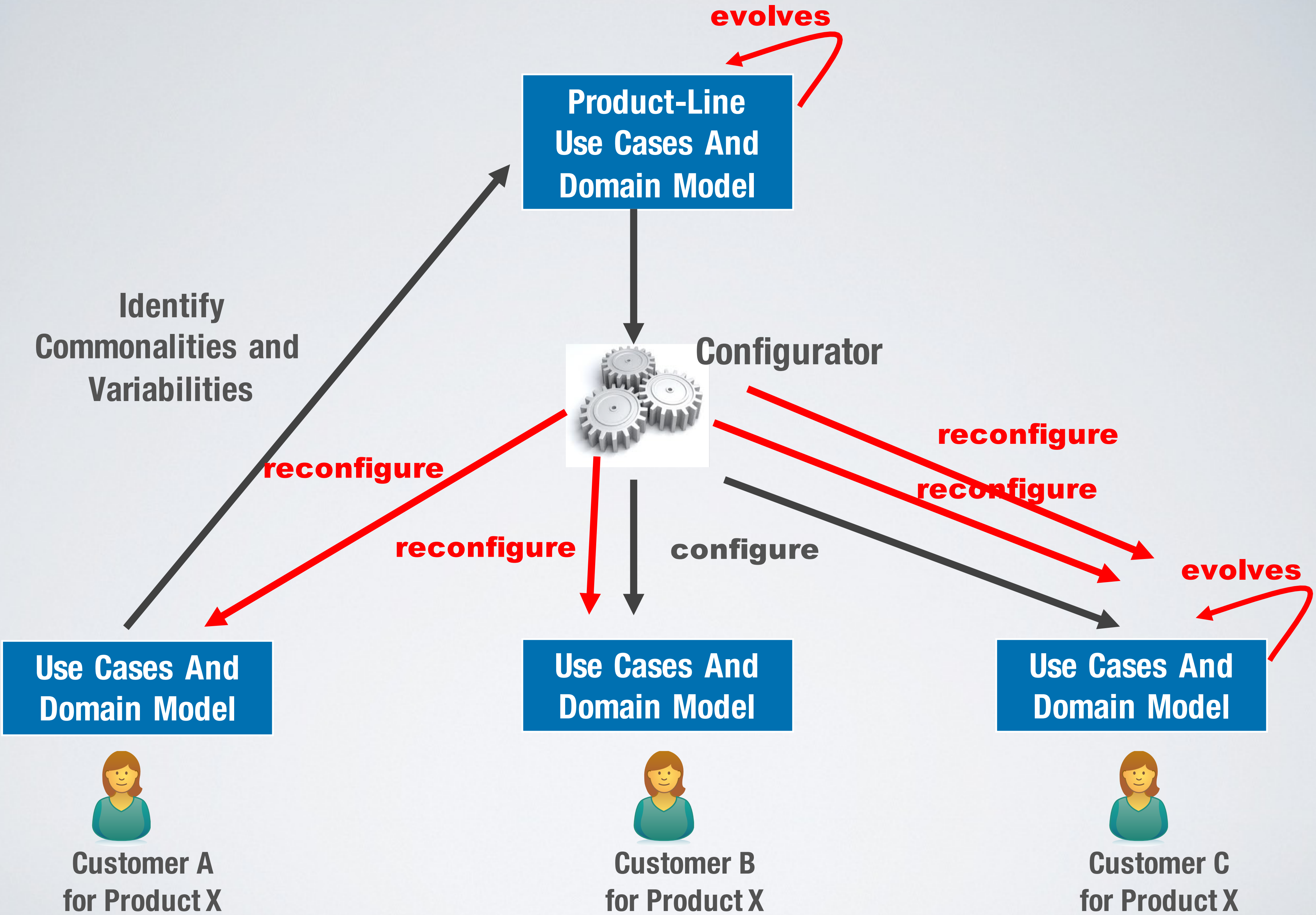
| | |
|---|---|
| **Use Case ID:** | 3 |
| **Use Case Name:** | Deposit check |
| **Actors:** | Customer |
| **Description:** | Deposit cash without using ATM card by using E-Card system. |
| **Preconditions:** | 1- The Customer has an activated E-Bank username and password.<br>2- The agreement should be signed by the customer.<br>3- The check must be valid. |
| **Postconditions:** | 1- Customer account balance is increased by the amount of the deposit check. |
| **Normal Flow:** | 1- Open the application.<br>2- The application shows welcome screen.<br>3- Log in to the application.<br>4- Choose the account.<br>5- Choose the transaction then deposit check service.<br>6- Enter the amount of money of the check and submit it.<br>7- Receiving the barcode.<br>8- Scan the barcode.<br>9- Insert the check into the ATM machine.<br>10- Receive notification.<br>11- 11- Log out of the application. |
| **Alternative Flows:** | 7a. if the customer didn't receive the barcode :<br>4- Customer will click on the get barcode bottom.<br>5- Bank sends a new barcode.<br>6- Use case resumes on step 8 of normal flow.<br><br>9a. if the ATM didn't accept the check:<br>3- Reenter the check into the ATM.<br>4- Use case resumes on step 9 of normal flow. |
| **Exceptions:** | 8a. In step 8 of the normal flow, if the customer cannot scan the barcode<br>4- Transaction is disapproved<br>5- Customer rescan the barcode correctly<br>6- Use Case resumes on step 9 of normal flow. |

# Dealing with Multiple Customers

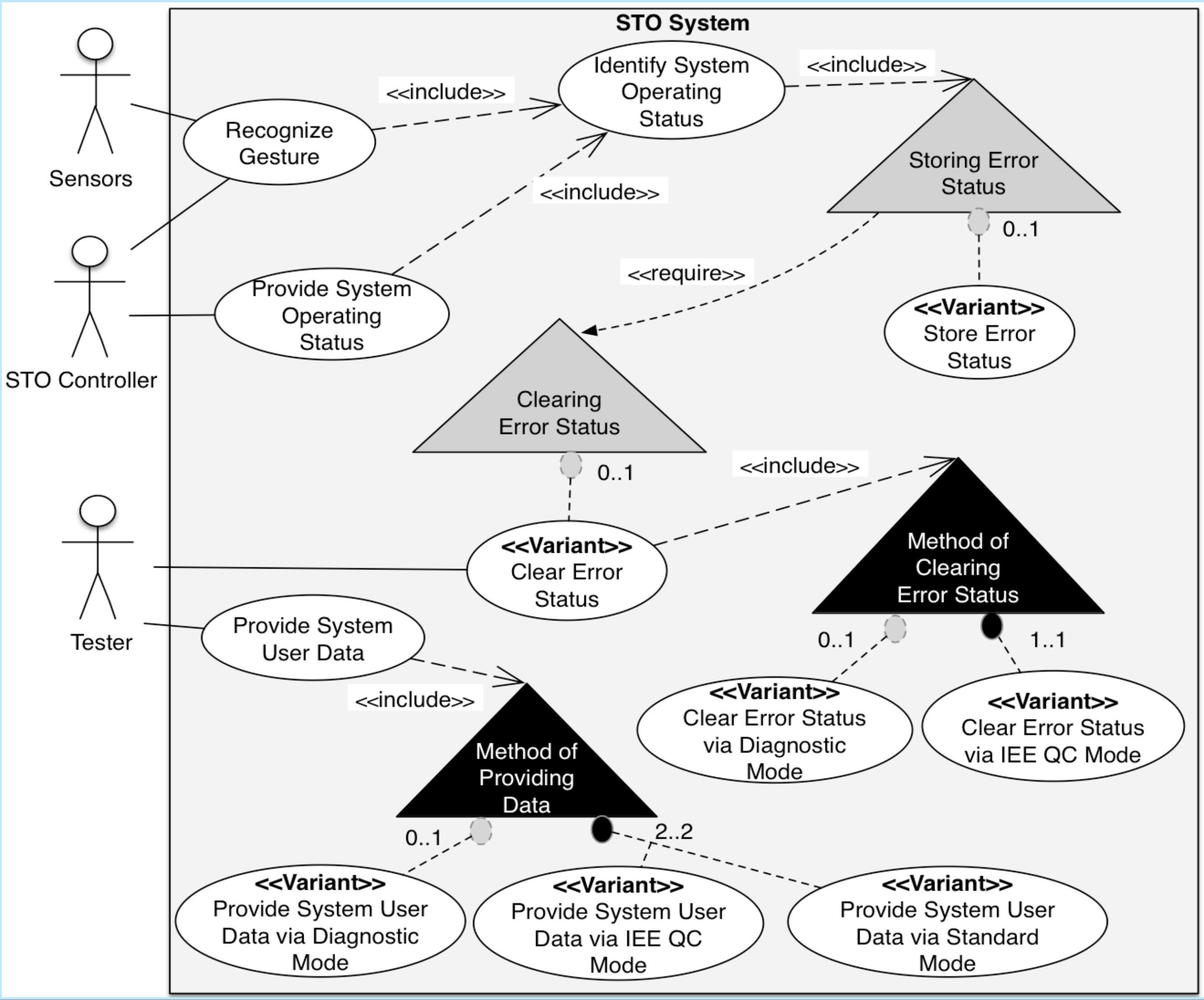


# Product Line Approach

- 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


evolves
Product-Line Use Cases And Domain Model
Identify Commonalities and Variabilities
Configurator
reconfigure
reconfigure
reconfigure
reconfigure
configure
evolves
Use Cases And Domain Model
Use Cases And Domain Model
Use Cases And Domain Model
Customer A for Product X
Customer B for Product X
Customer C for Product X

# 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**

| **Basic Flow** |
|---|
| 1. INCLUDE USE CASE Identify System Operating Status. |
| 2. The system VALIDATES THAT the operating status is OK. |
| 3. The system REQUESTS the move capacitance FROM the UpperSensor. |
| 4. The system REQUESTS the move capacitance FROM the LowerSensor. |
| 5. The system VALIDATES THAT the movement is a valid kick. |
| 6. The system VALIDATES THAT the overuse protection feature is enabled. |
| 7. The system VALIDATES THAT the Overuse protection status is inactive. |
| 8. The system SENDS the valid kick status TO the STOController. |
| **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:

| |
|---|
| **Use Case: Identify System Operating Status** |
| **Basic Flow** |
| 1. The system VALIDATES THAT the watchdog reset is valid. |
| 2. The system VALIDATES THAT the RAM is valid. |
| 3. The system VALIDATES THAT the sensors are valid. |
| 4. The system VALIDATES THAT there is no error detected. |
| **Specific Alternative Flow** |
| RFS 4 |
| 1. **INCLUDE VARIATION POINT: Storing Error Status.** |
| 2. ABORT. |

# 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
- **[ICST 2017]** C. Wang et al., System Testing of Timing Requirements based on Use Cases and Timed Automata

# Product Families and Configuration

- **[MODELS 2015]** I. Hajri et al., Applying Product Line Use Case Modeling in an Industrial Automotive Embedded System: Lessons Learned and a Refined Approach

- **[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