Analyzing Natural-Language Requirements:

Industrial Needs and Scalable Solutions

UU/SIKS Symposium on Natural Language in Requirements Engineering, 30 November 2017

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

Introduction

- NL requirements come in a variety of forms
- NL requirements won’t go away
- Many and varying industrial needs
- NLP has made a huge leap forward in recent years
- Research leading to practical and scalable solutions
- Context factors, working assumptions

Outline

- Report on a variety of research projects
- Collaborations with industry
- Various objectives and applications
- Examples from automotive and satellite
- Lessons learned

Experience

- Compliance with requirements templates
- Change impact analysis
- Domain knowledge extraction
- Requirements completeness assessment
- Requirements-driven testing
- Product lines and configuration

Checking Compliance with Templates

Representative Context

SES

your satellite company

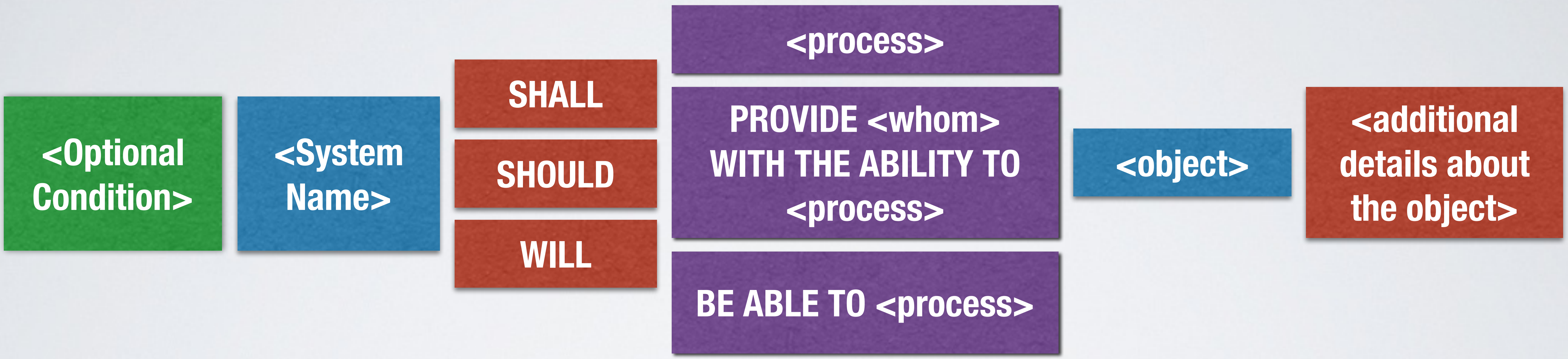
Challenges

- Large projects (e.g., ESA)
- Hundreds of natural language requirements
- Tiers of requirements
- Many stakeholders
- Requirements capture a contract
- Requirements frequently change

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
- Existing tools (DODT, RQA) require **glossary or ontology**

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?

Approach

- **Text chunking:** identifies sentence segments (chunks) without performing expensive analysis
- **NLP parsing** only when needed
- 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

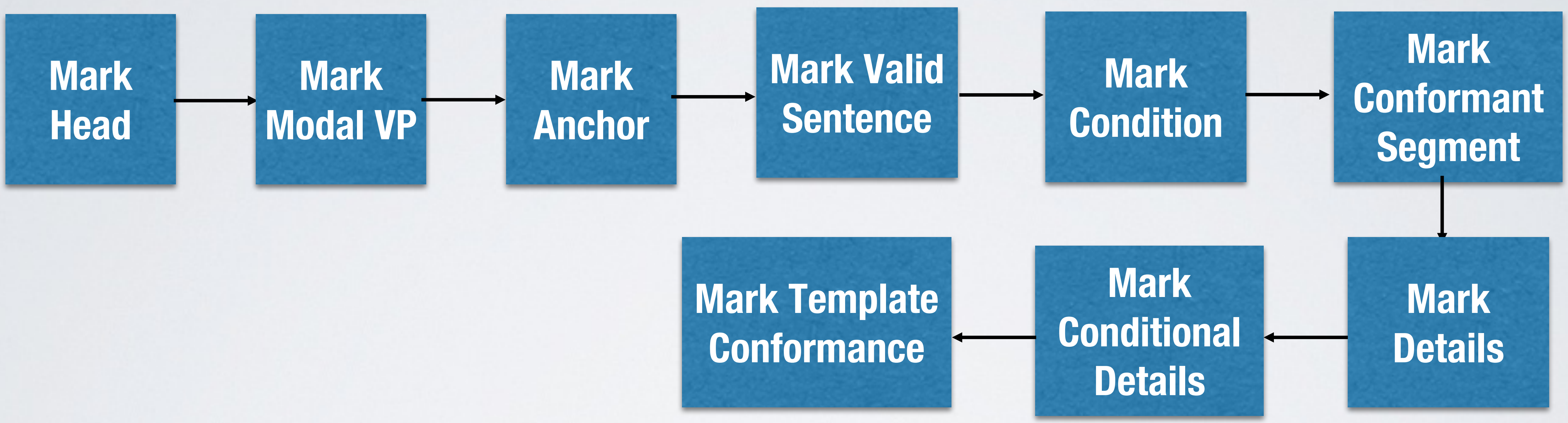
Process of decomposing a sentence into non-overlapping segments.

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

380 Requirements
380 Requirements

1 case study

110 Requirements

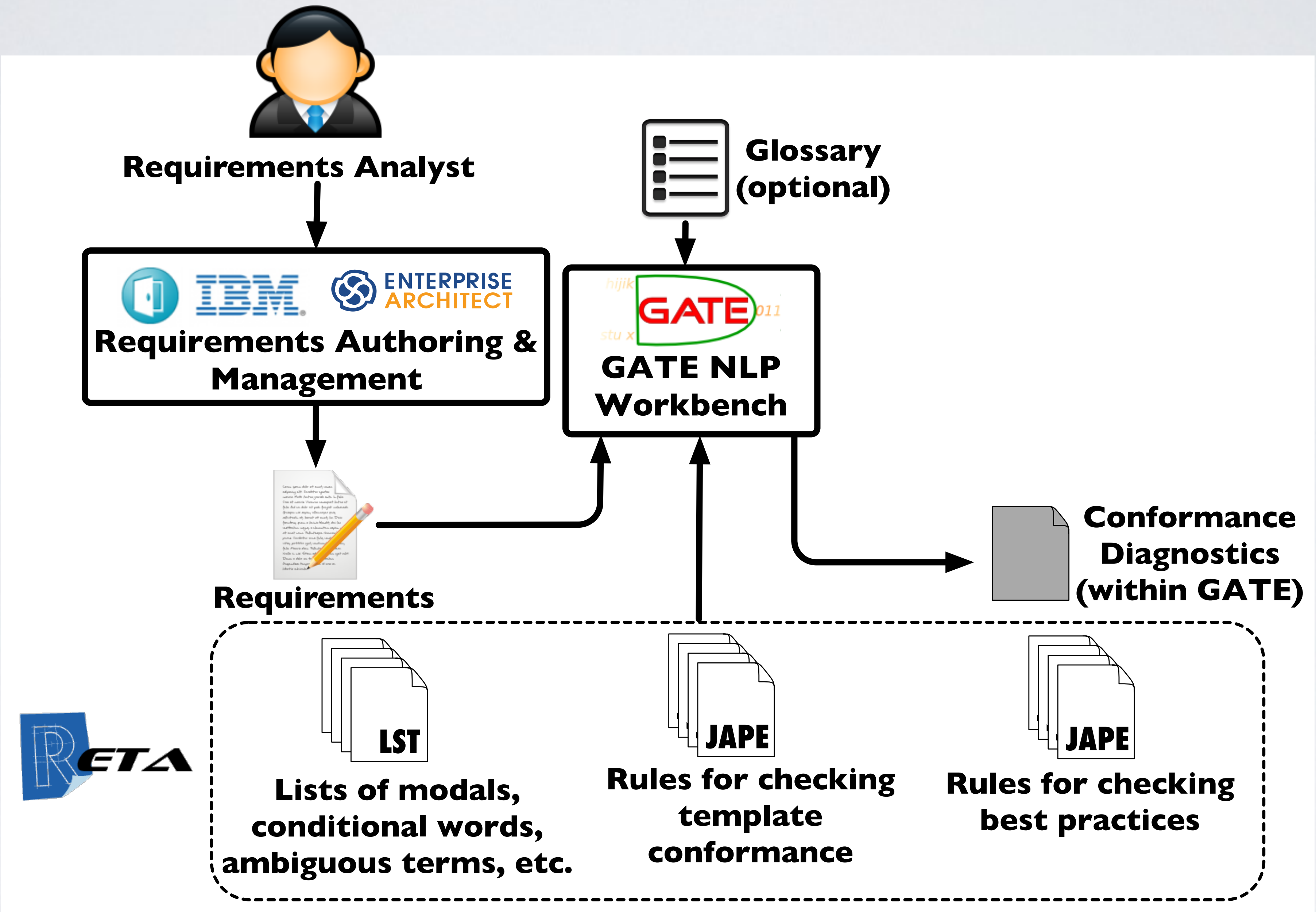
1 case study

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

Supporting 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

Inter-Requirements

Inter-Requirements
Change Impact Analysis

Approach

- 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" (R3)
3: We no longer want to "transmit satellite status reports" to "user help desk" but instead to "user document repository" (only R1)

Solution Characteristics

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

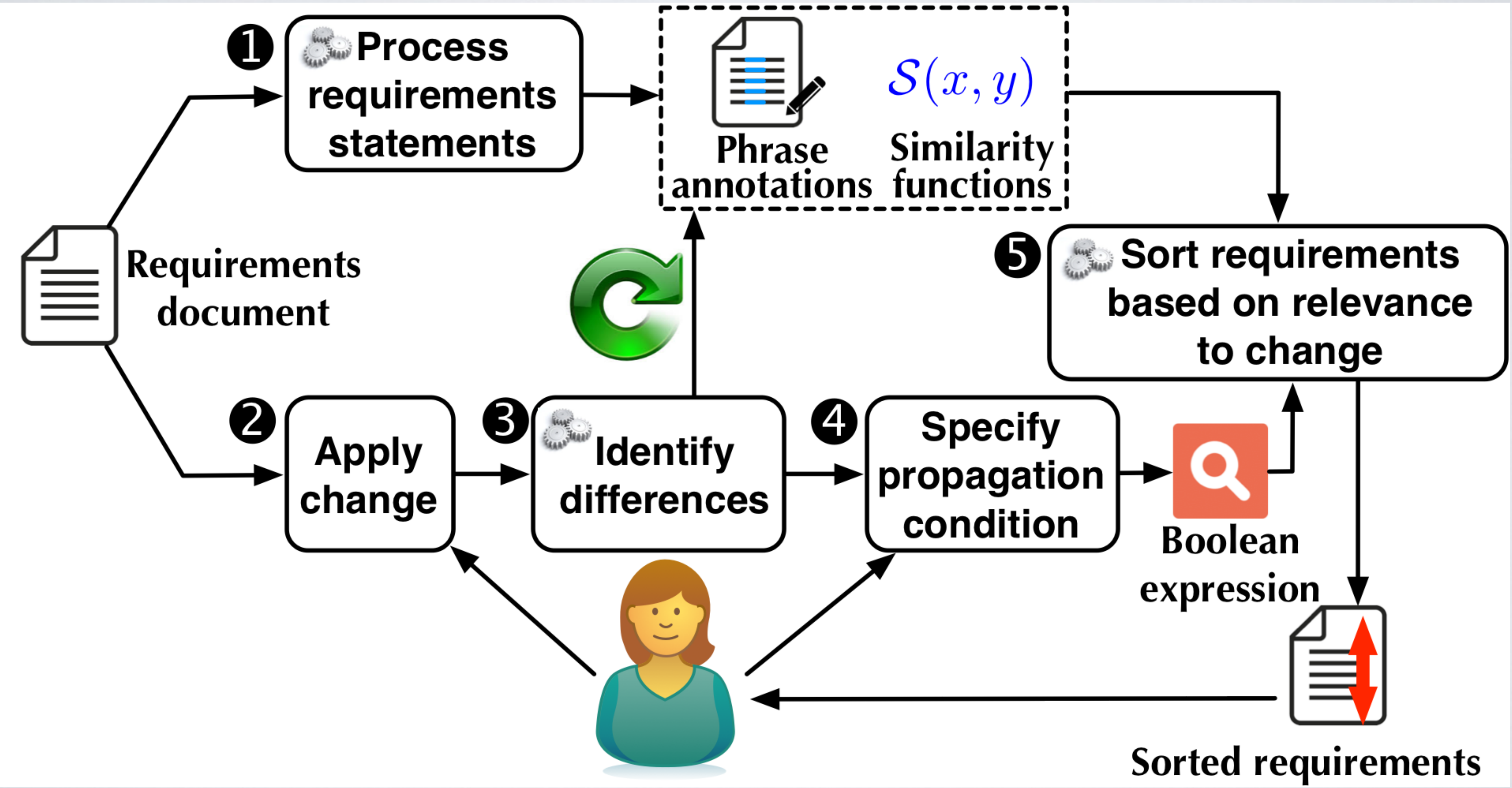
The mission operation controller shall transmit satellite status reports to the user ~~help desk~~ document repository.

user help desk, Deleted
user document repository, Added

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

- **Account for change rationale expressed by user**

Narcia



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

Evaluation

SES
your satellite company

1 case study

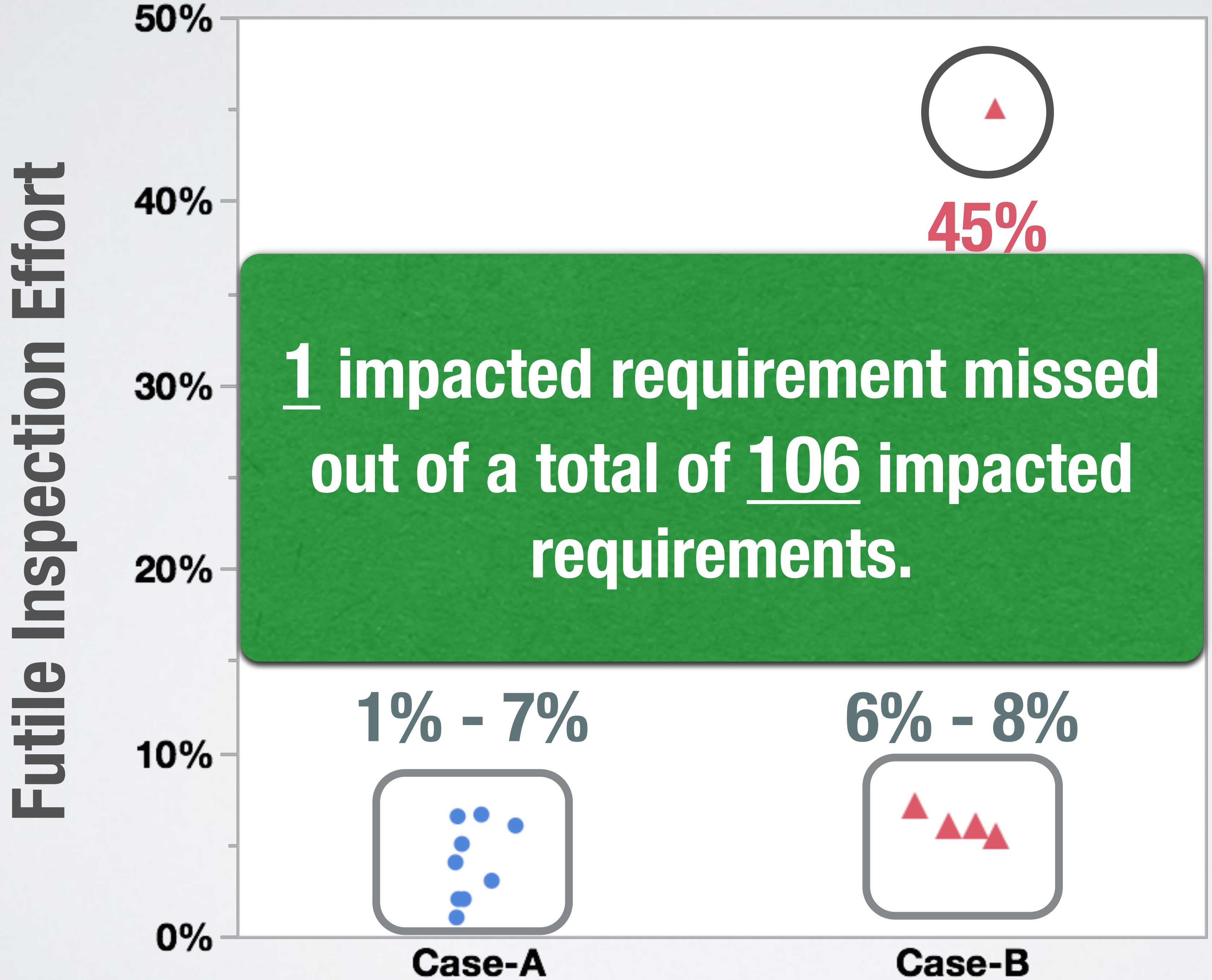
158 Requirements
9 change scenrios

novay

1 case study

72 Requirements
5 Change
Scenarios

Effectiveness of Our Approach



Requirements to Design

Requirements-to-Design
Change Impact Analysis

Motivations

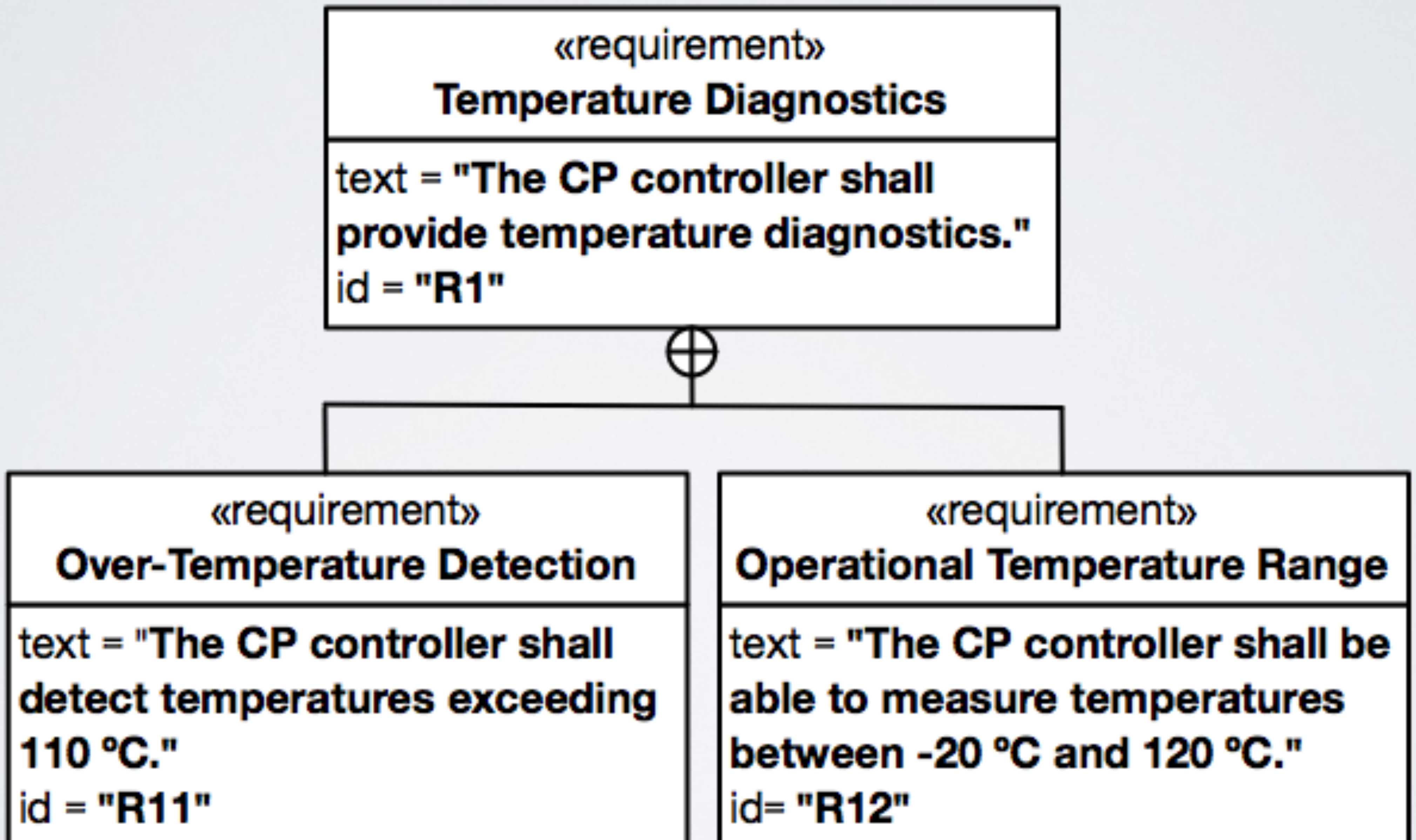
- **Rigorous change management** required by many standards and customers in safety critical systems, and embedded systems in general in many industry sectors

- Impact of requirements changes on **design decisions**

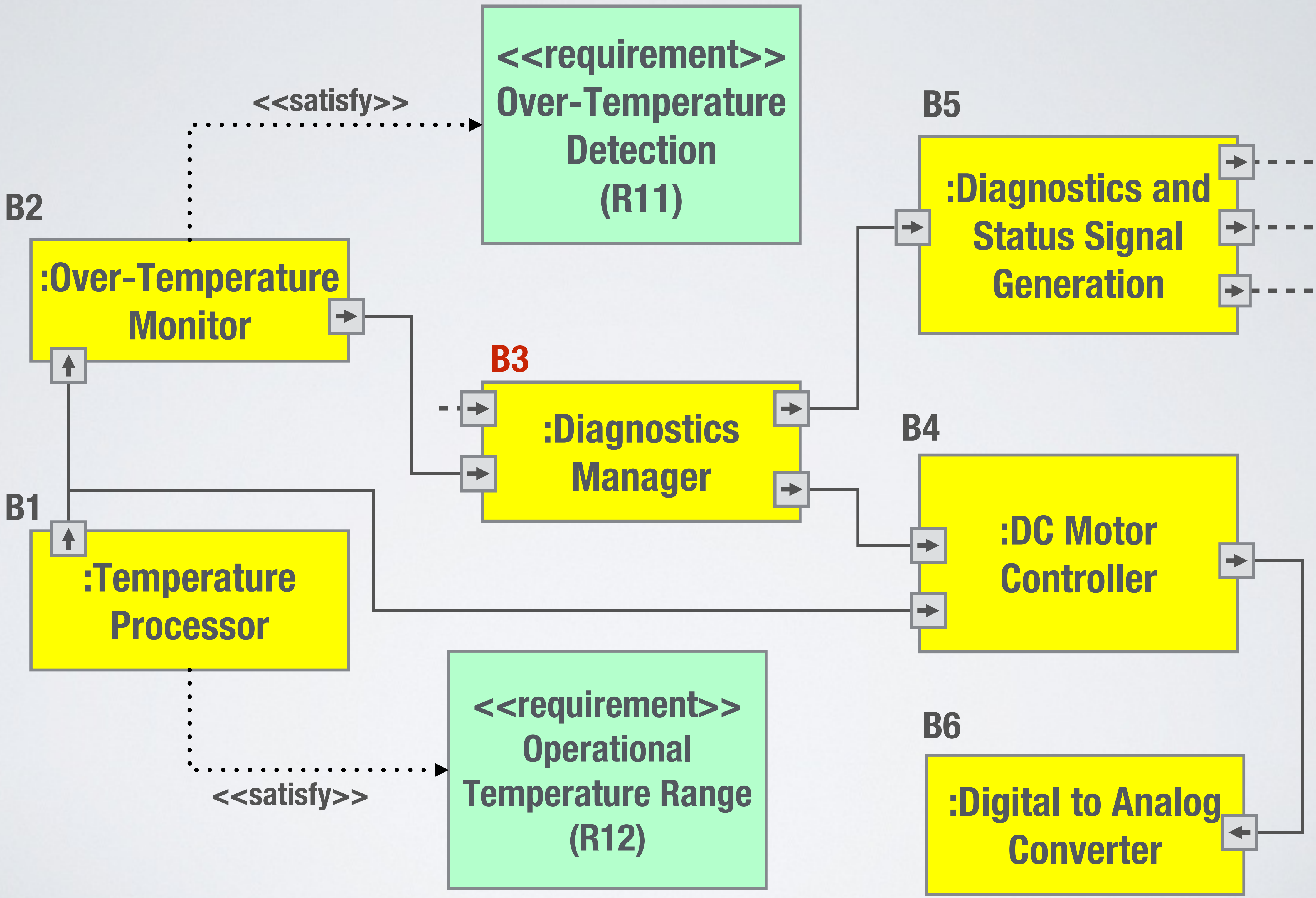
- **Complete and precise design impact set**

- **SysML** commonly used as embedded and cyber-physical system design representation

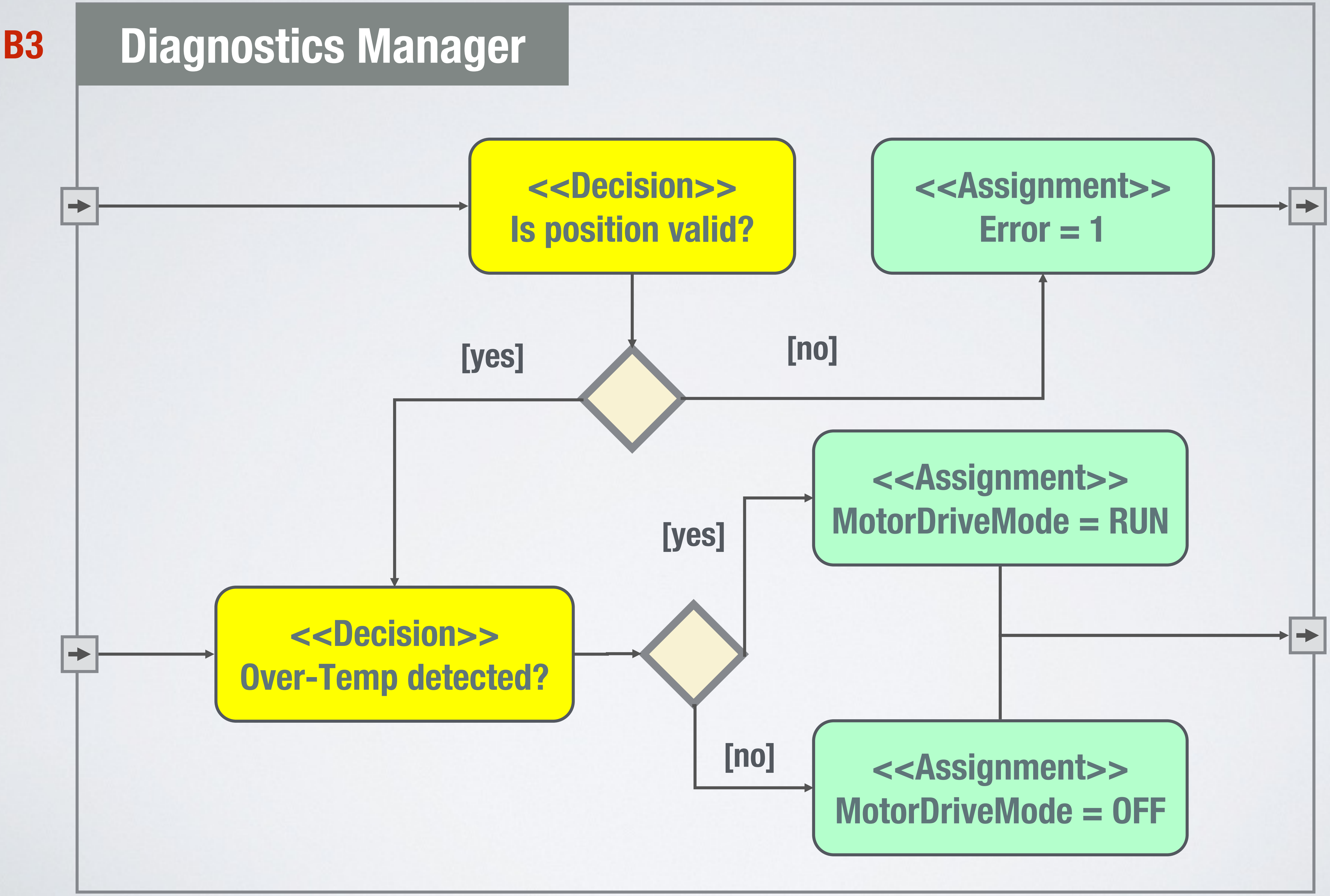
Requirements Diagram



Structural Diagram



Behavioural Diagram

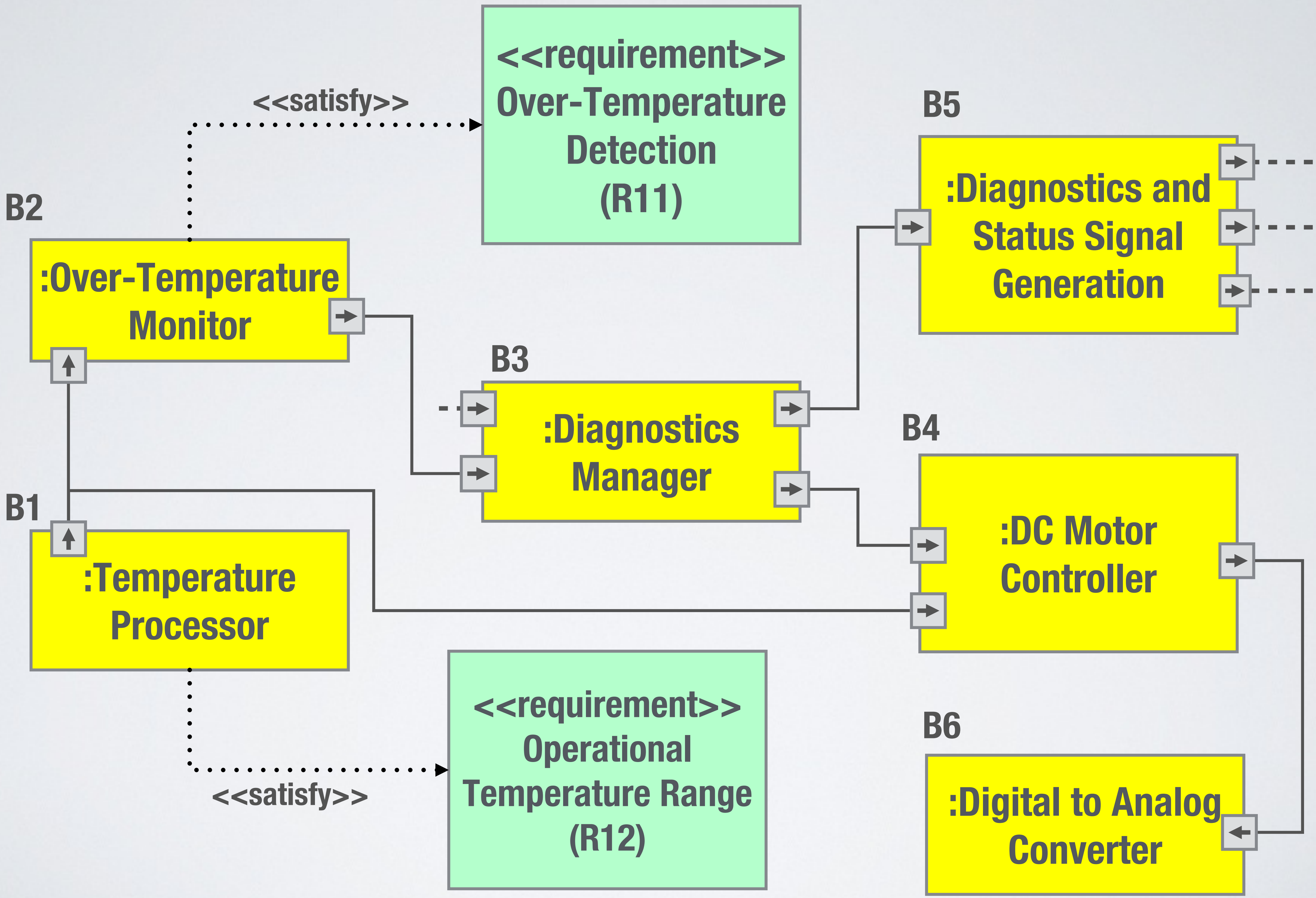


Compute Impacted Elements

Structural Analysis

Behavioural Analysis

Structural Diagram

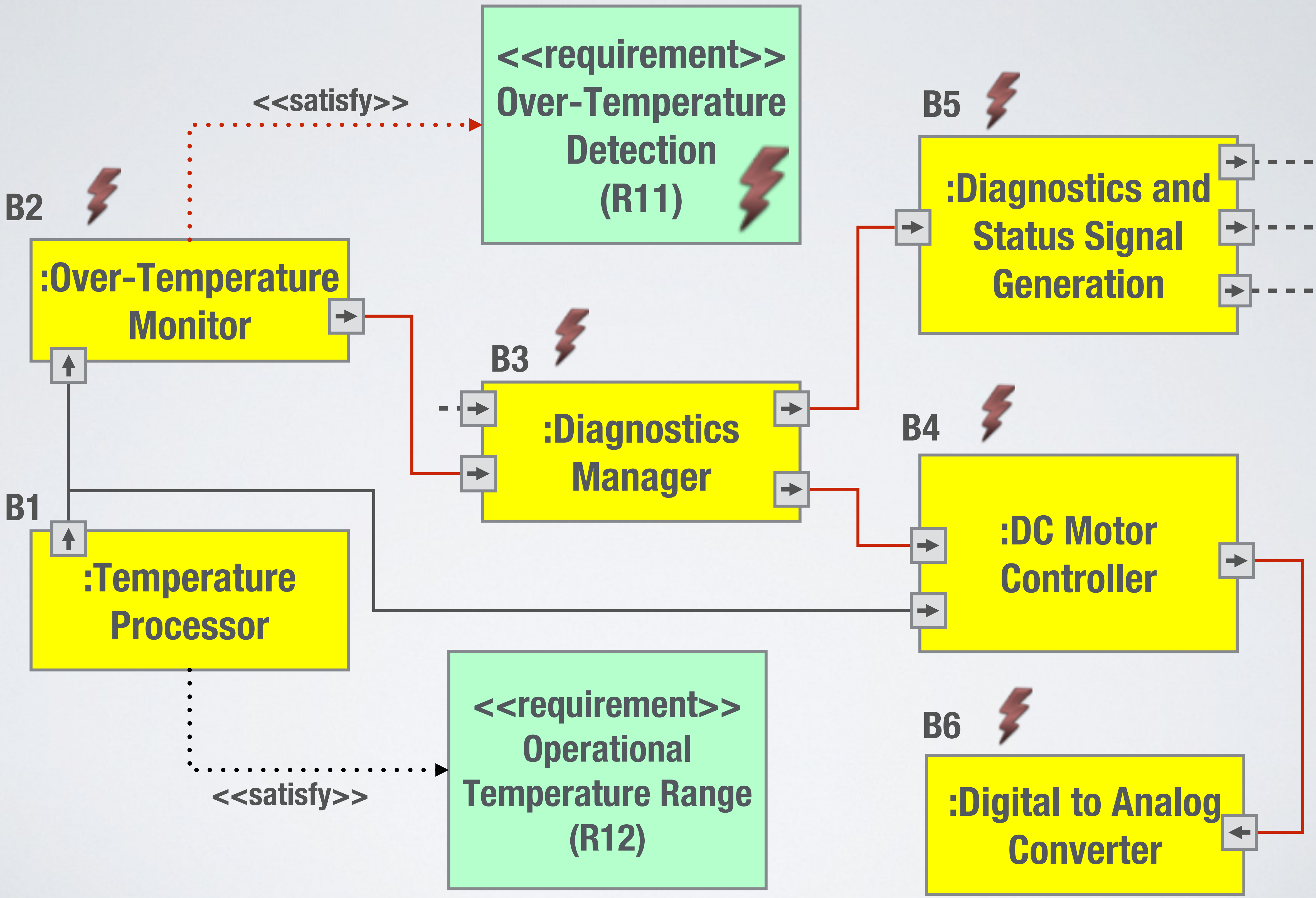


Structural Diagram

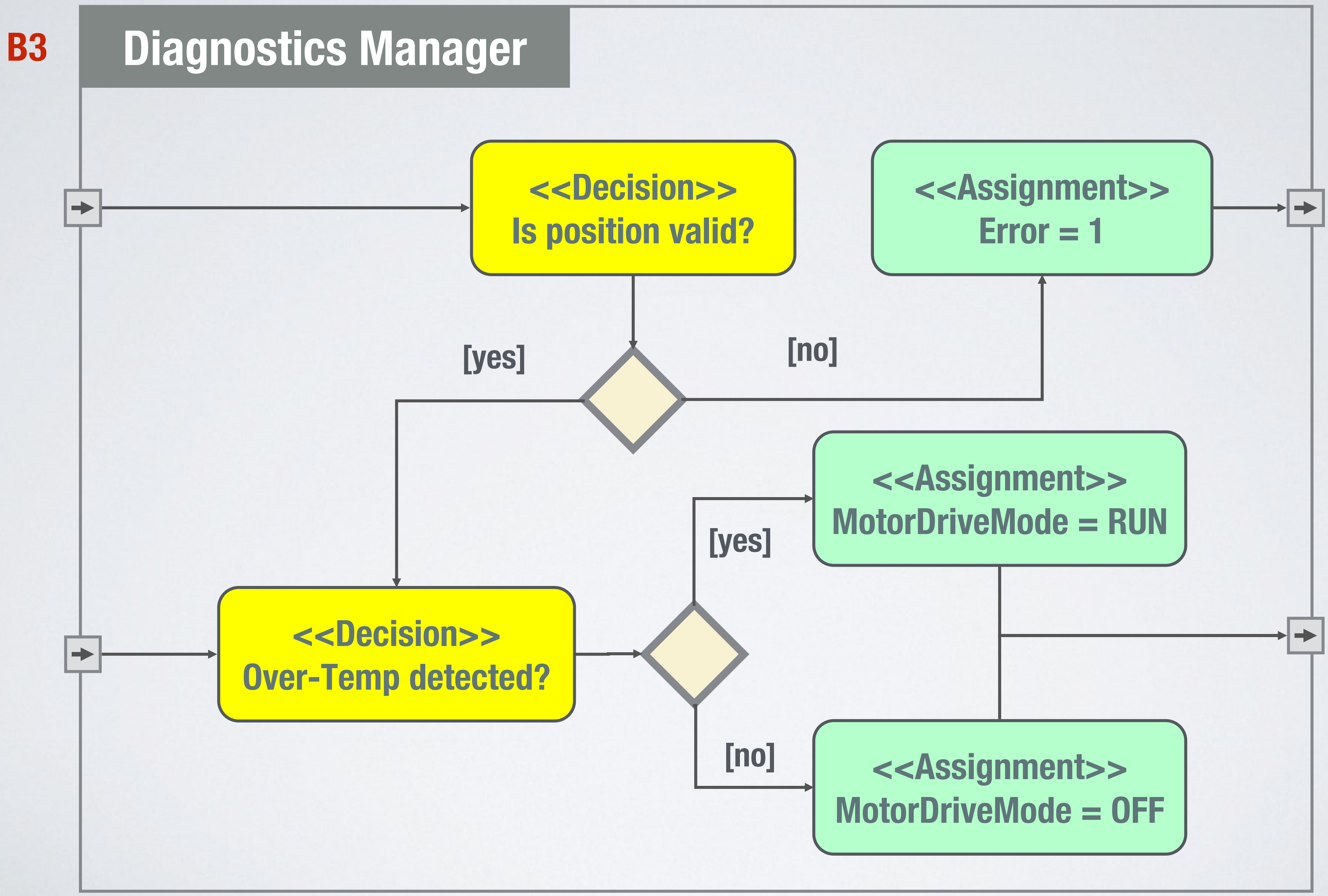
<<requirement>>

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

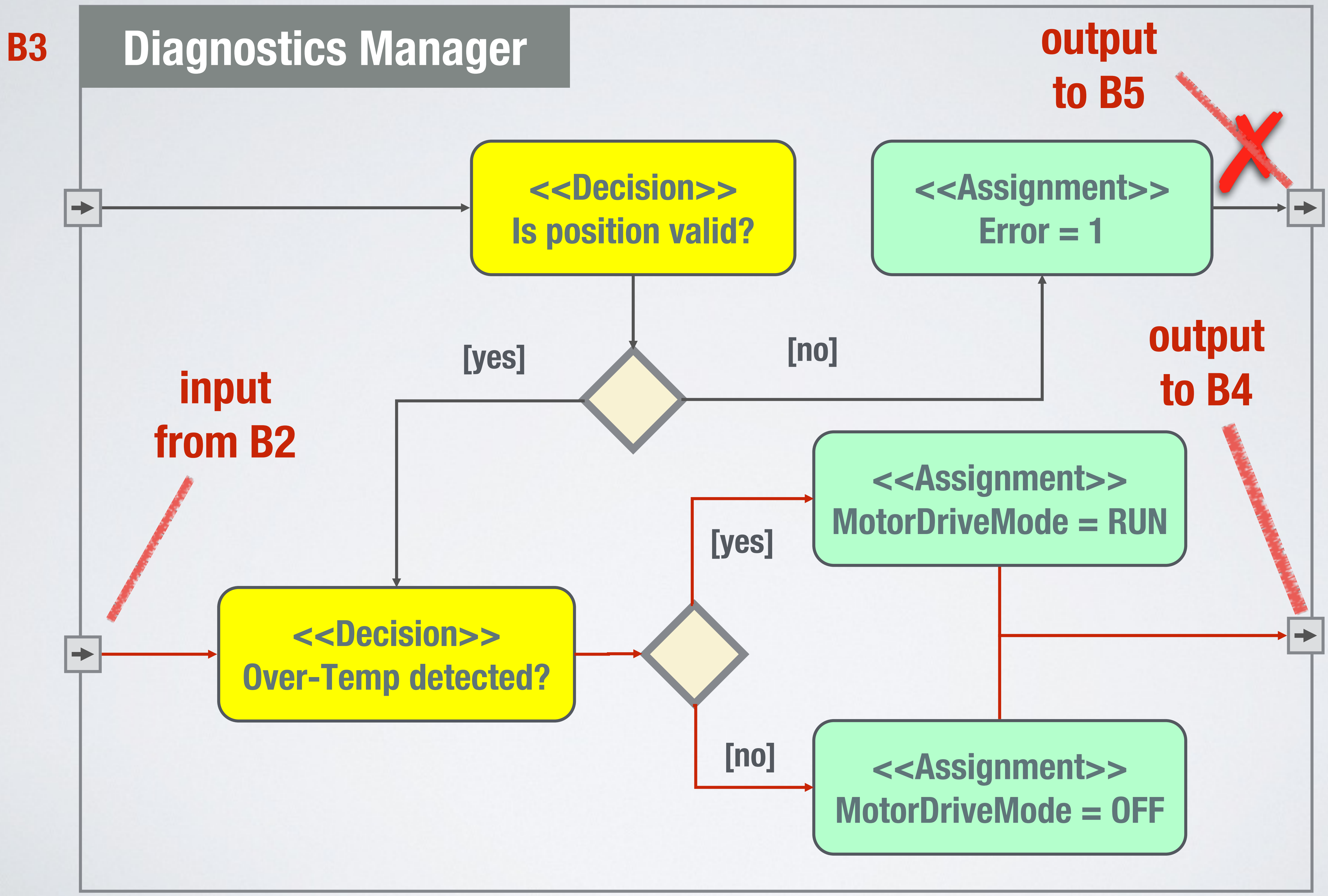
Structural Diagram



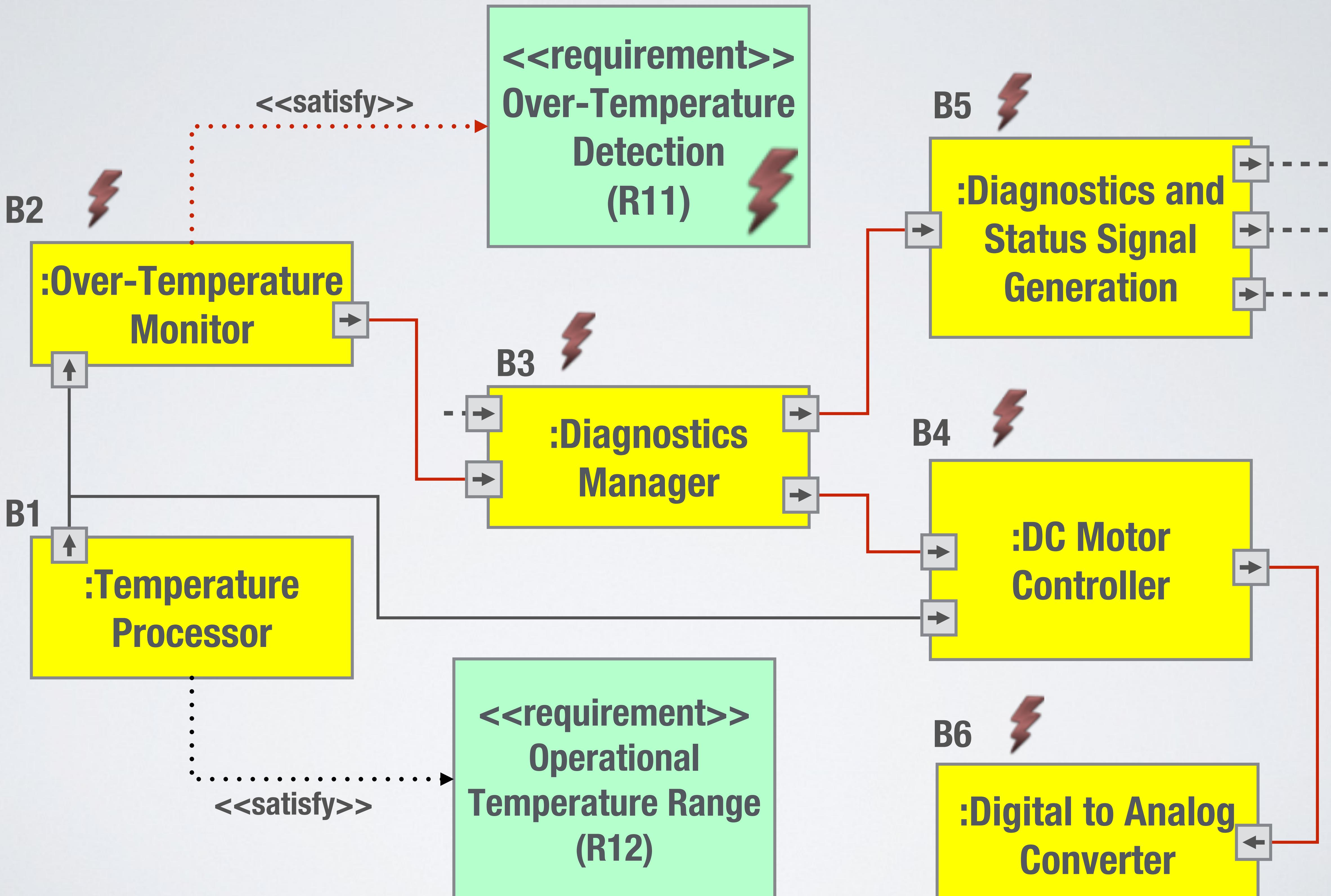
Behavioural Diagram



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

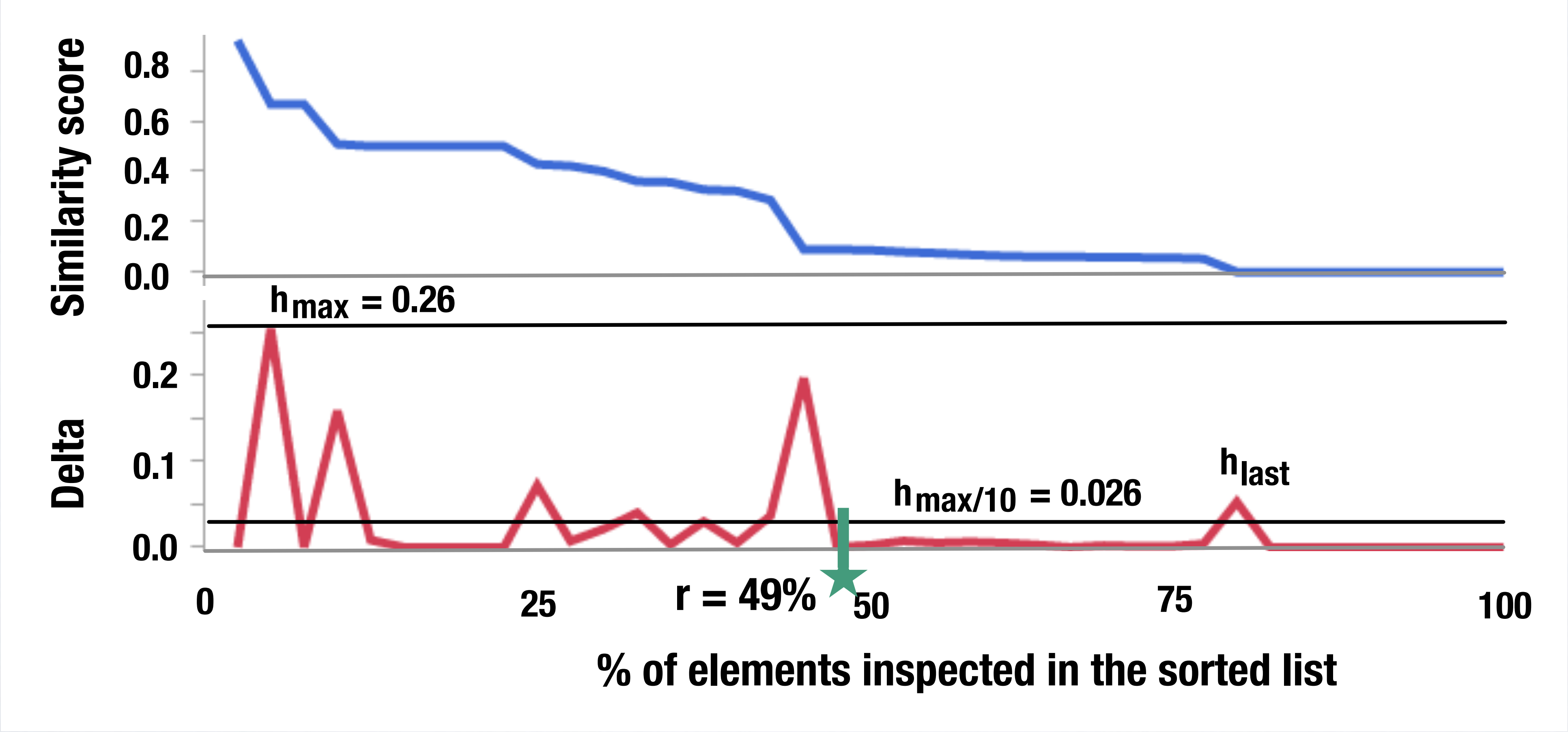
- **Informal inputs** from systems engineers regarding impact of changes
- **Example:** “Temperature lookup tables and voltage converters need to be adjusted”

Natural Language Processing

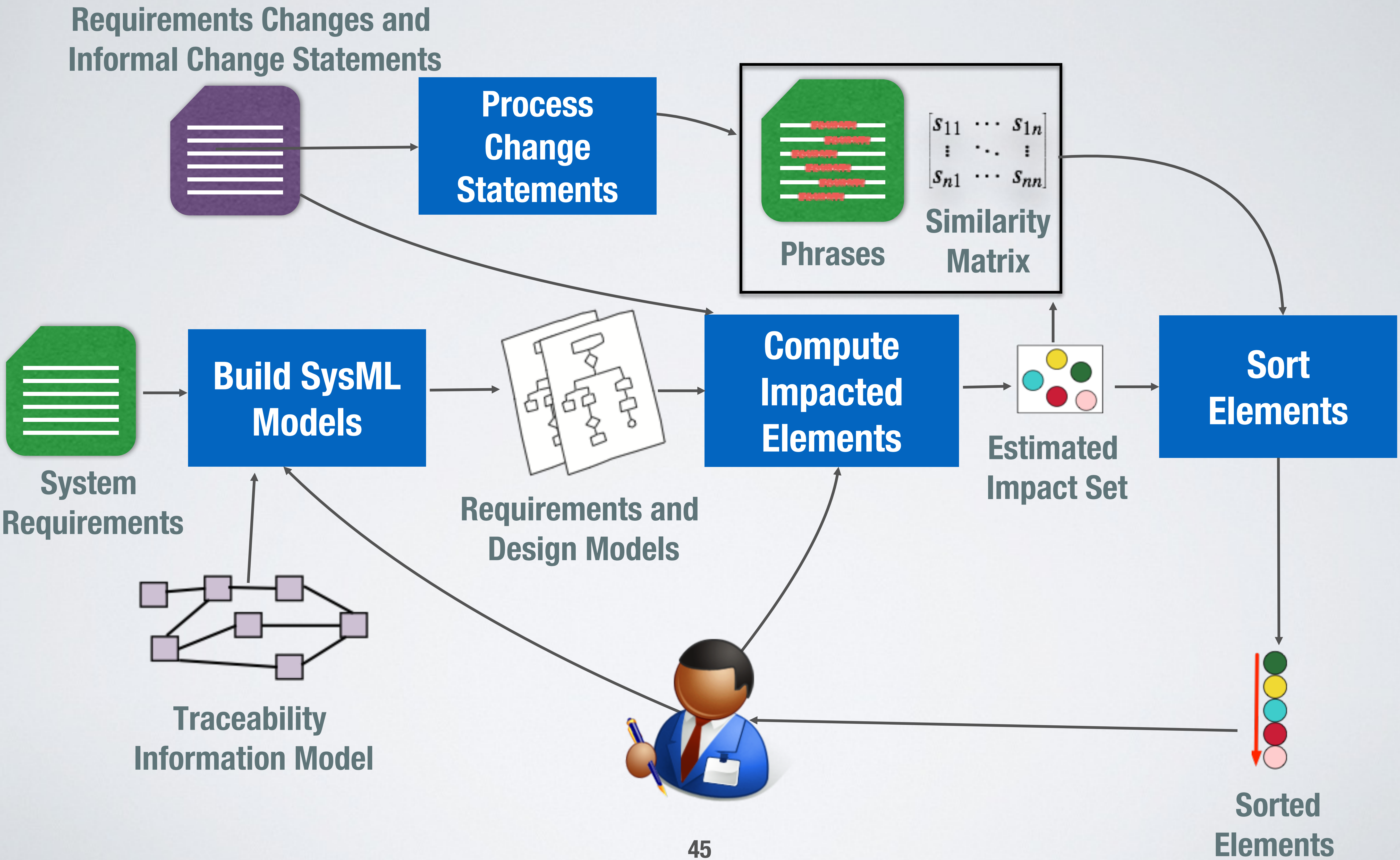
- Computing **similarity scores** for model elements by applying NLP techniques to measure similarity (syntactic and semantic) **between model elements labels and change statements.**
- **Sorting** the design elements obtained after structural and behavioral analysis based on the similarity scores
- 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



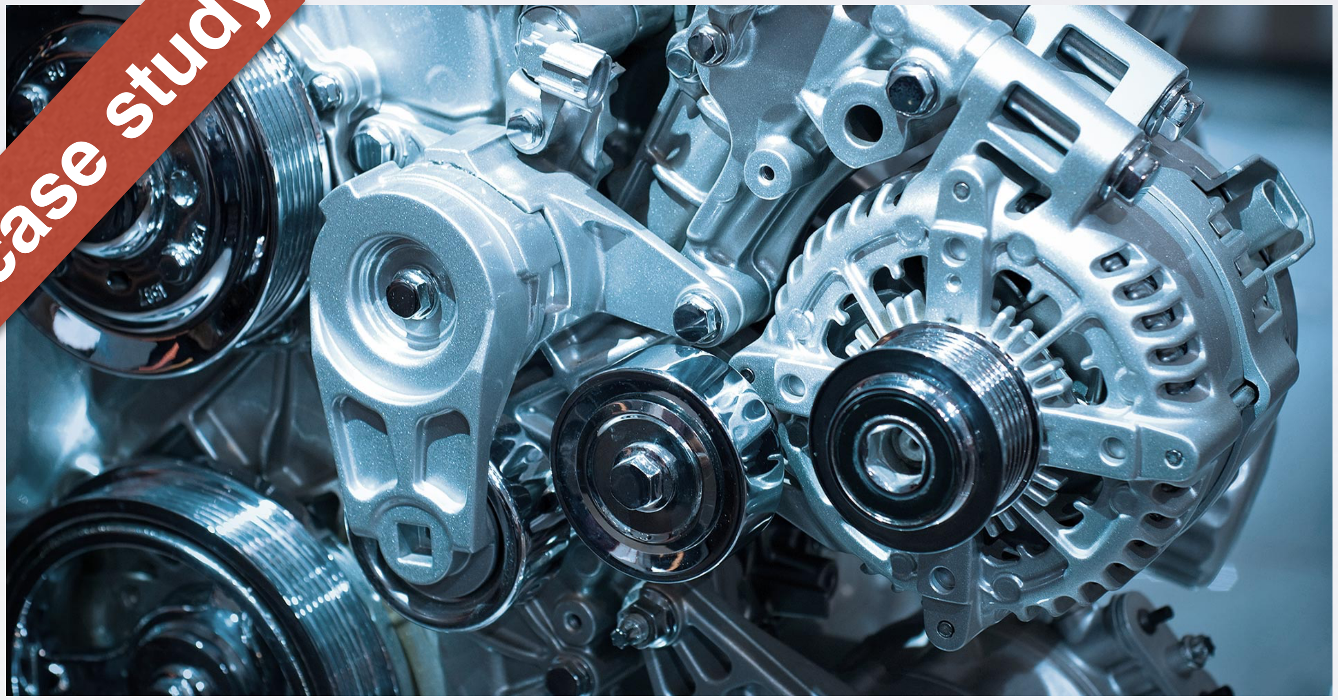
Requirements Changes and Informal Change Statements
Process Change Statements
Phrases
$\begin{bmatrix} s_{11} & \cdots & s_{1n} \\ \vdots & \ddots & \vdots \\ s_{n1} & \cdots & s_{nn} \end{bmatrix}$
Similarity Matrix
System Requirements
Build SysML Models
Requirements and Design Models
Compute Impacted Elements
Estimated Impact Set
Sort Elements
Traceability Information Model
Sorted Elements

Evaluation

DELPHI

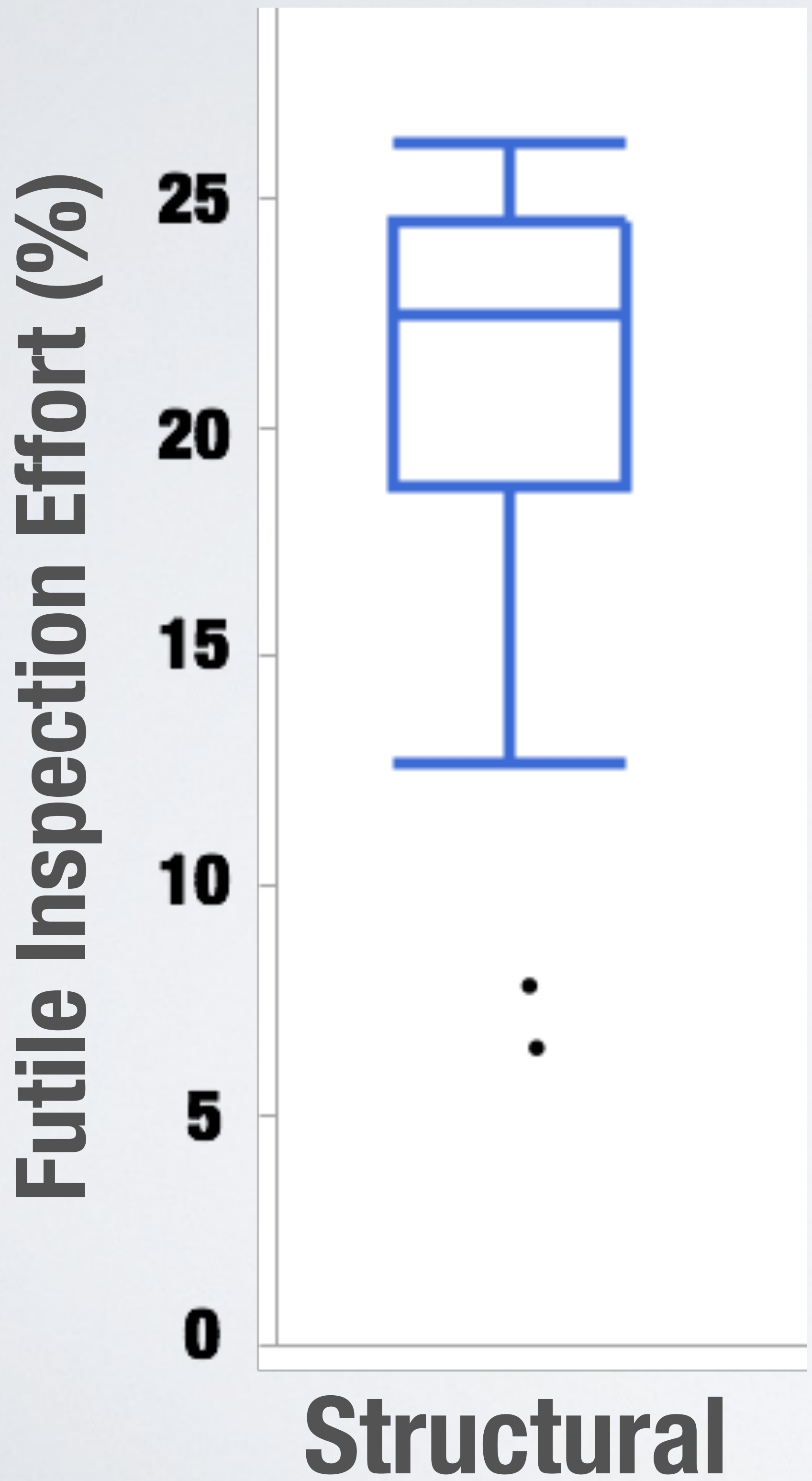
Innovation for the Real World

1 case study

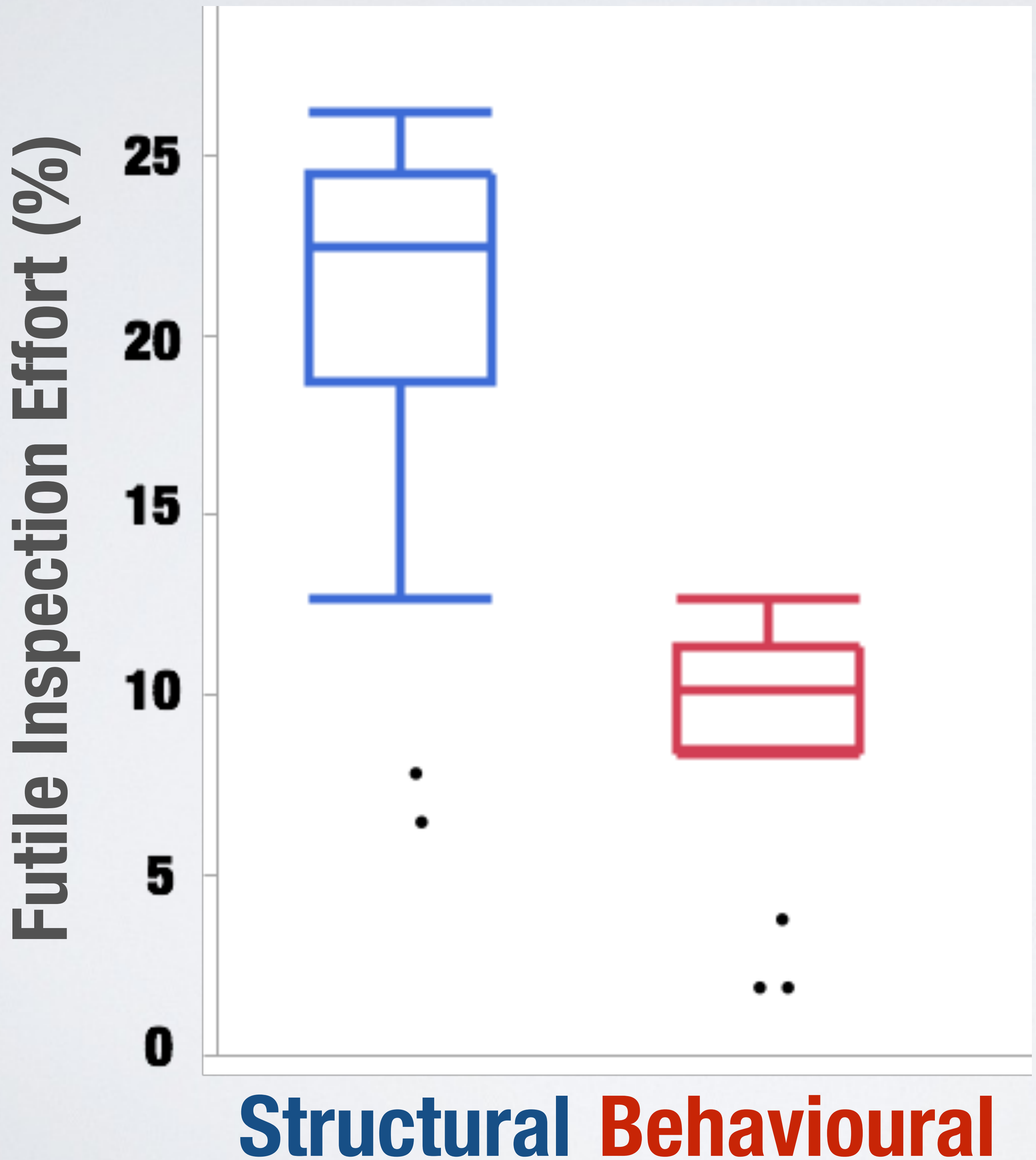


370 elements
16 change scenarios

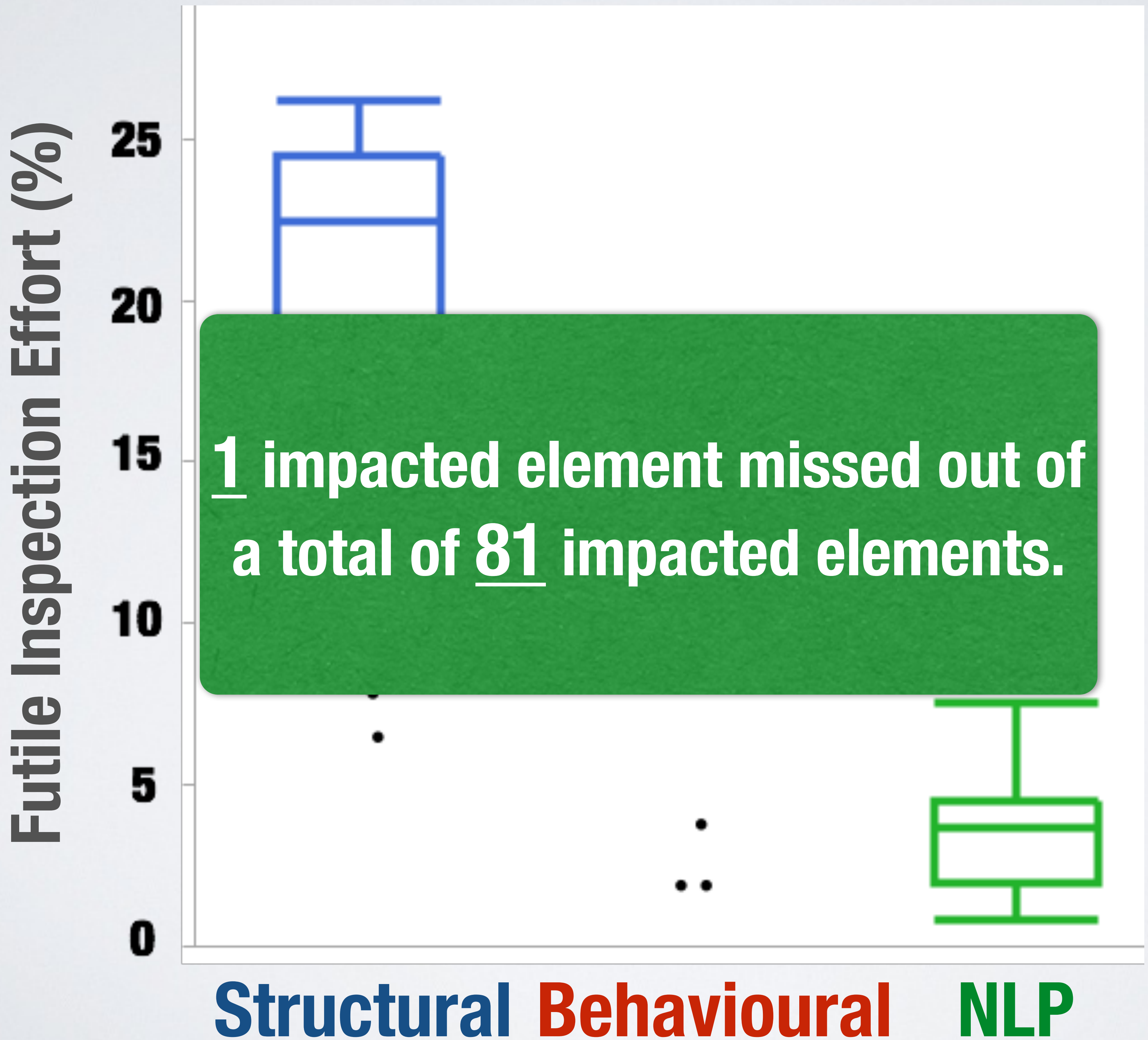
Effectiveness of Our Approach



Effectiveness of Our Approach



Effectiveness of Our Approach



Extracting Domain Knowledge

Domain Knowledge

- All requirements depend, more or less explicitly, on domain knowledge
- Domain-specific concepts and terminology
- In practice: 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

Glossary Extraction and Clustering

Terminology

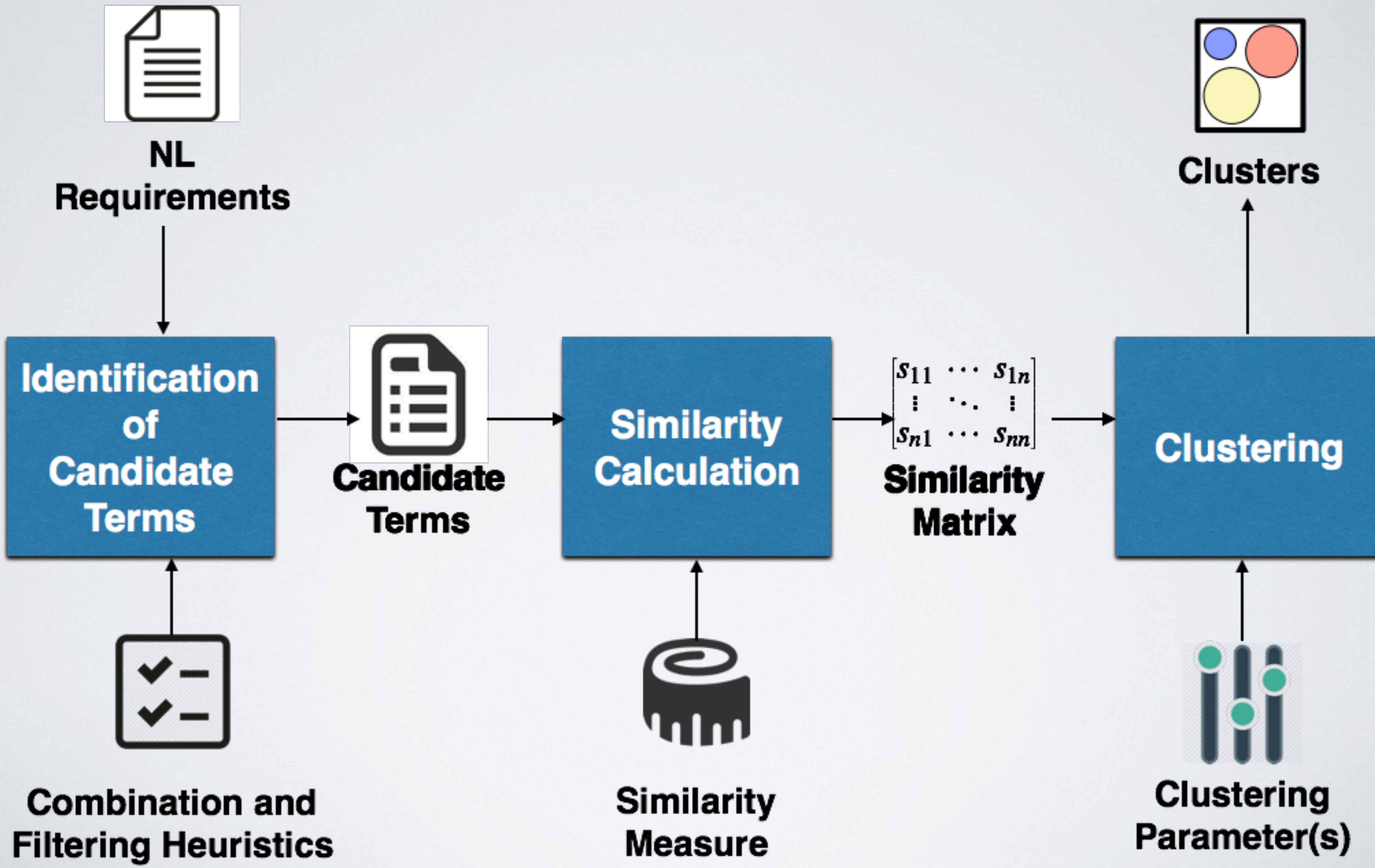
- Usually multiple stakeholders, organizations …
- Inconsistent terminology
  - Multiple terms for same concepts
    - element / component / object
  - Multiple representations of same keywords
    - status of Ground Station Interface component
    - Ground Station Interface component's status
    -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

Requirements Glossary

- Glossaries help **mitigate ambiguities**
    - consistent terminology
    - improves communication among stakeholders

- **Glossaries are, <u>in practice</u>, rarely (fully) defined before requirements are written**

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

OPENCOSS

2 case studies

380 Requirements
138 Requirements

1 case study

110 Requirements

Results

ΔRecall > 20%

Our Approach

JATE

TOPIA

TextRank

TermoStat

TermRaider

No clustering

Results

Precision ~

Our Approach

VS

JATE

TOPIA

TextRank

TermoStat

TermRaider

Clustering Evaluation

SES
your satellite company

OPENCOSS

2 case studies

1 case study

20 clusters
each 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)

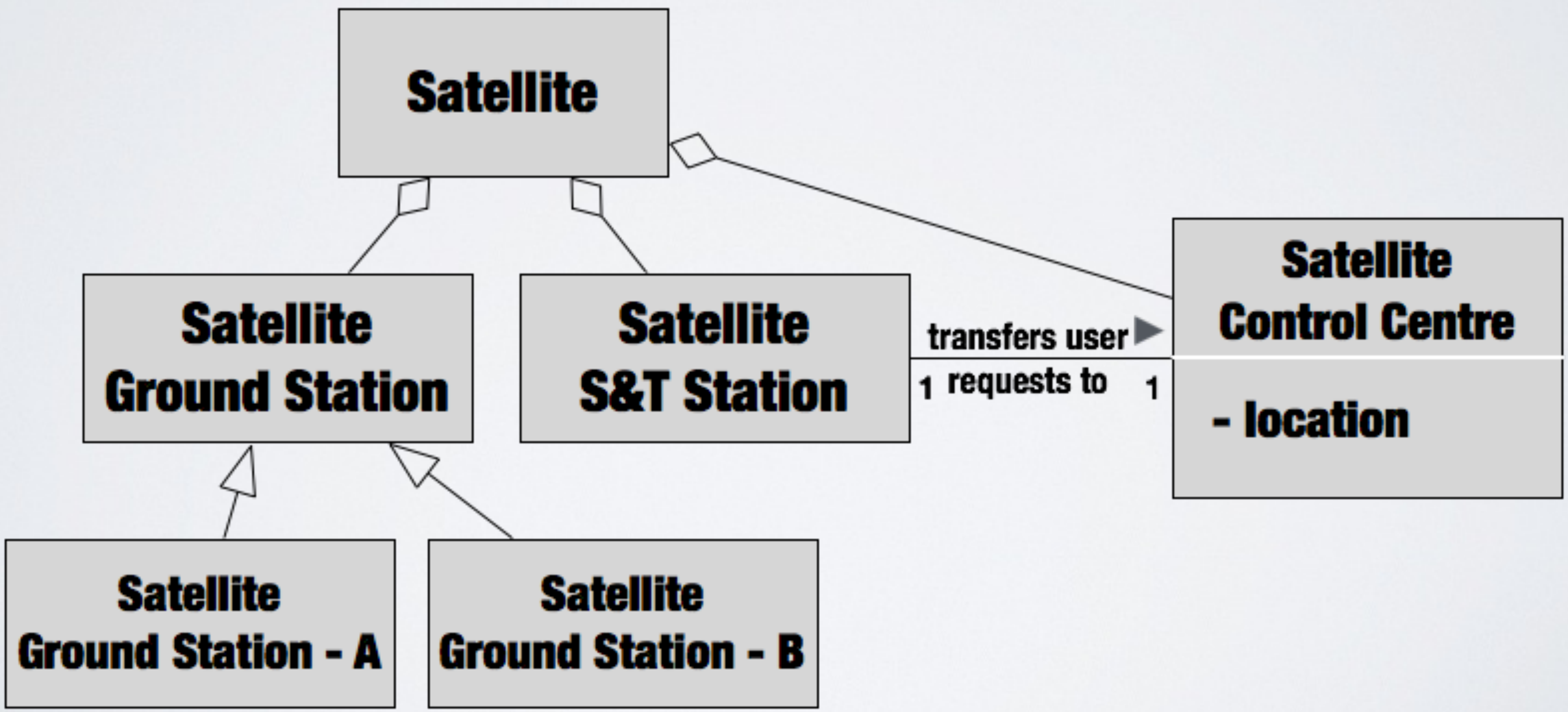
Domain Model Extraction

Motivation

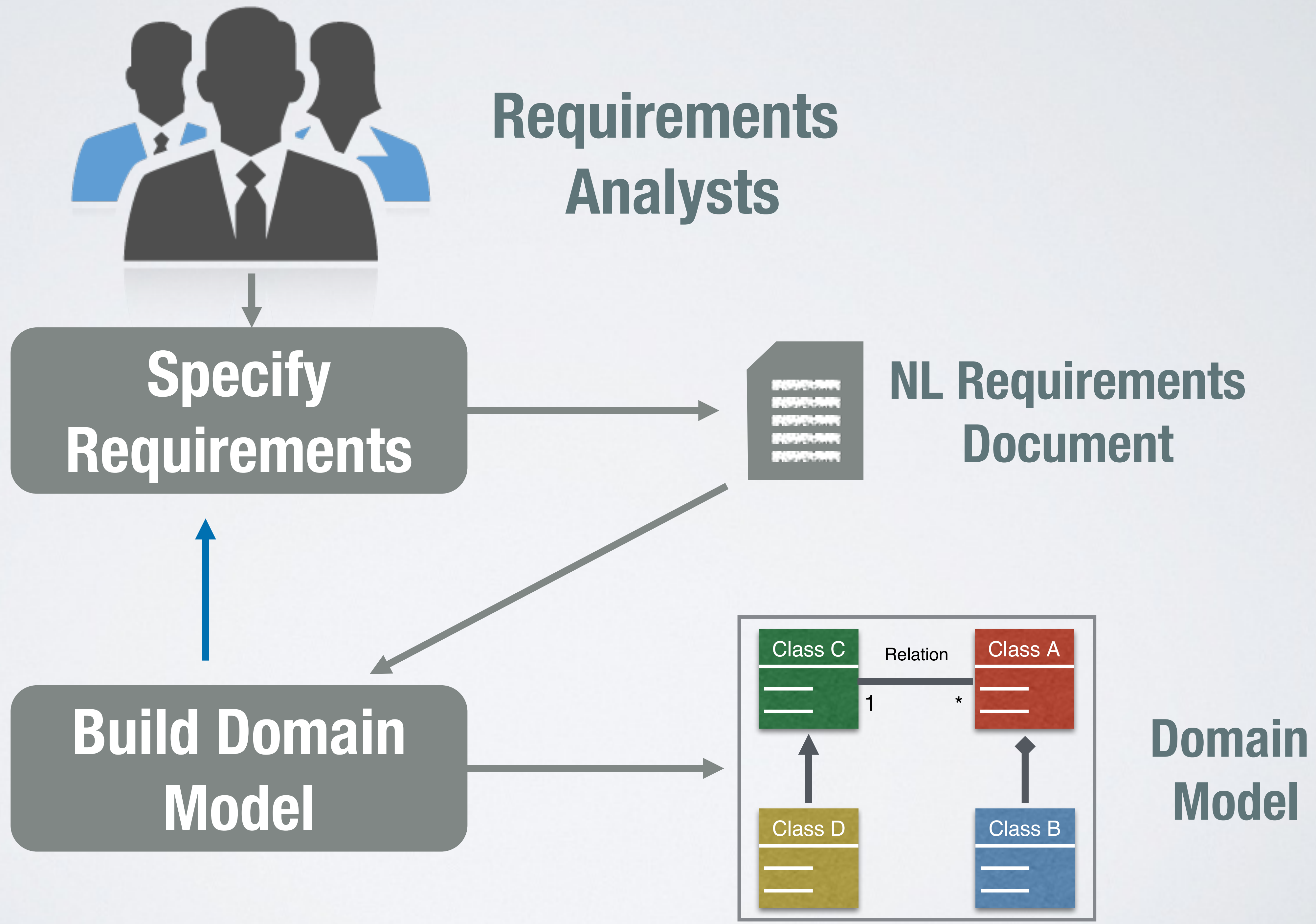
- Representation of important domain concepts and their relations
  - Facilitate communication between stakeholders from different backgrounds
  - Help identify inconsistencies in terminology, etc.
  - Helps assess completeness of requirements
- In practice, domain models are not preceding the elicitation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

Domain Models

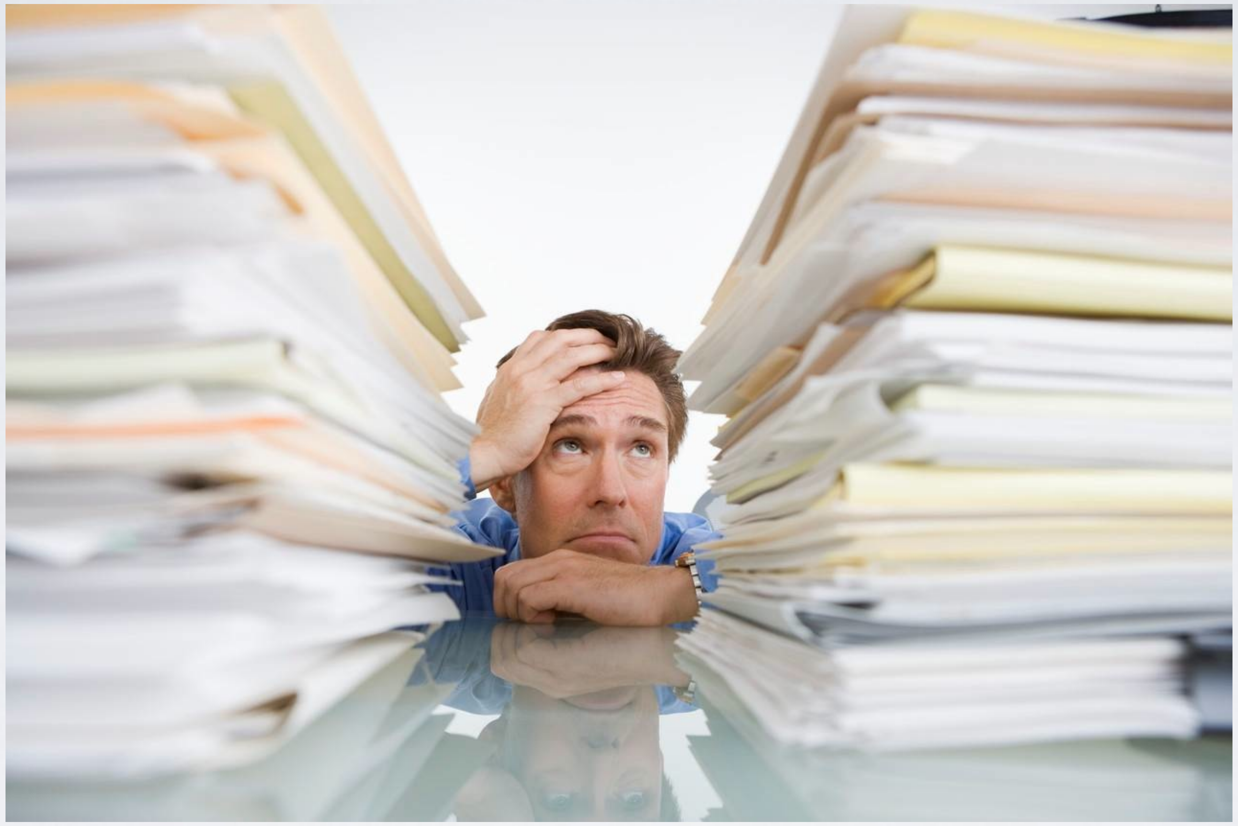
A domain model is a representation of conceptual entities or real-world objects in a domain of interest.



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

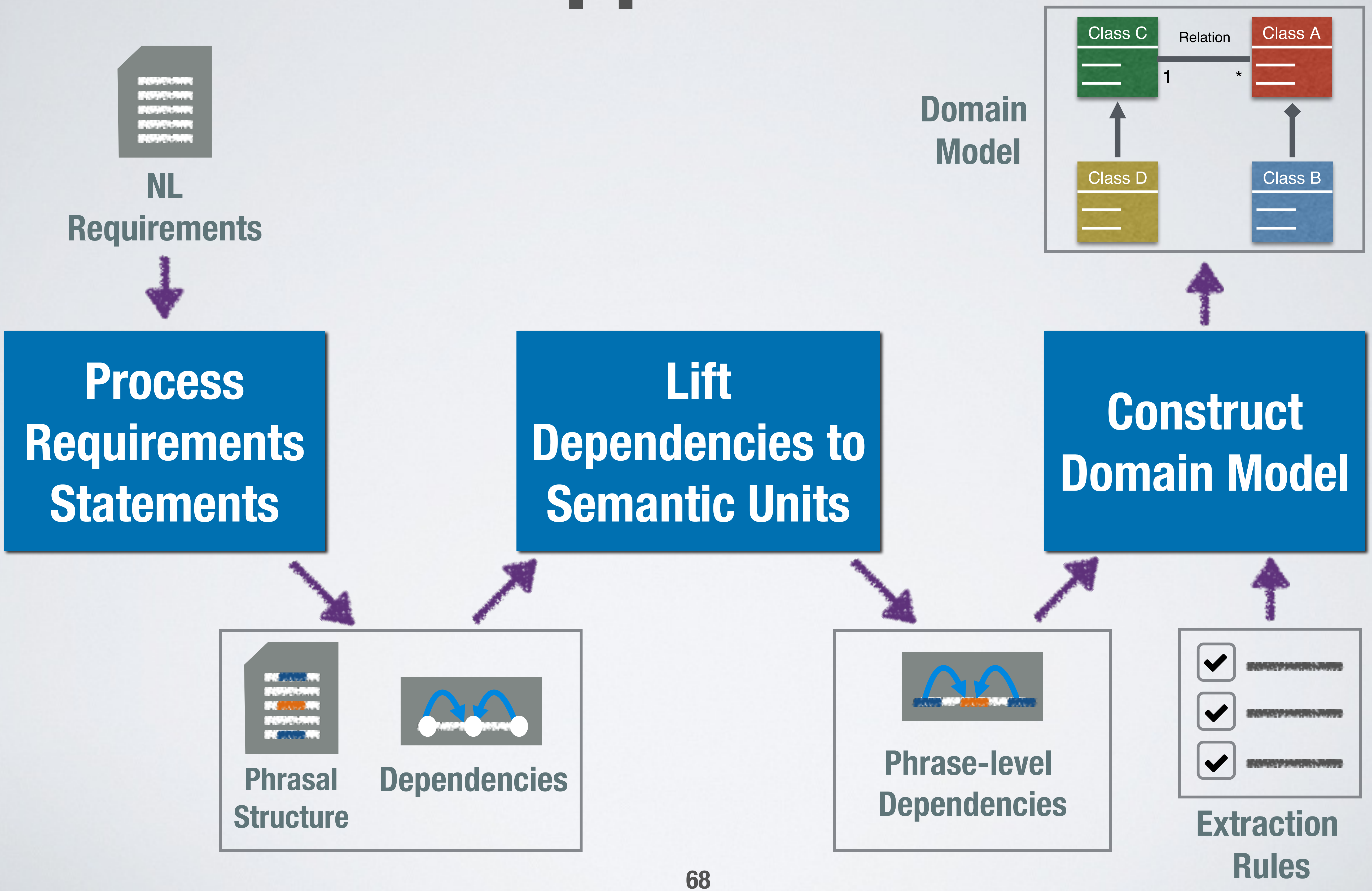


- Manually building domain models is laborious
- Automated support is required for building domain models

State of the Art

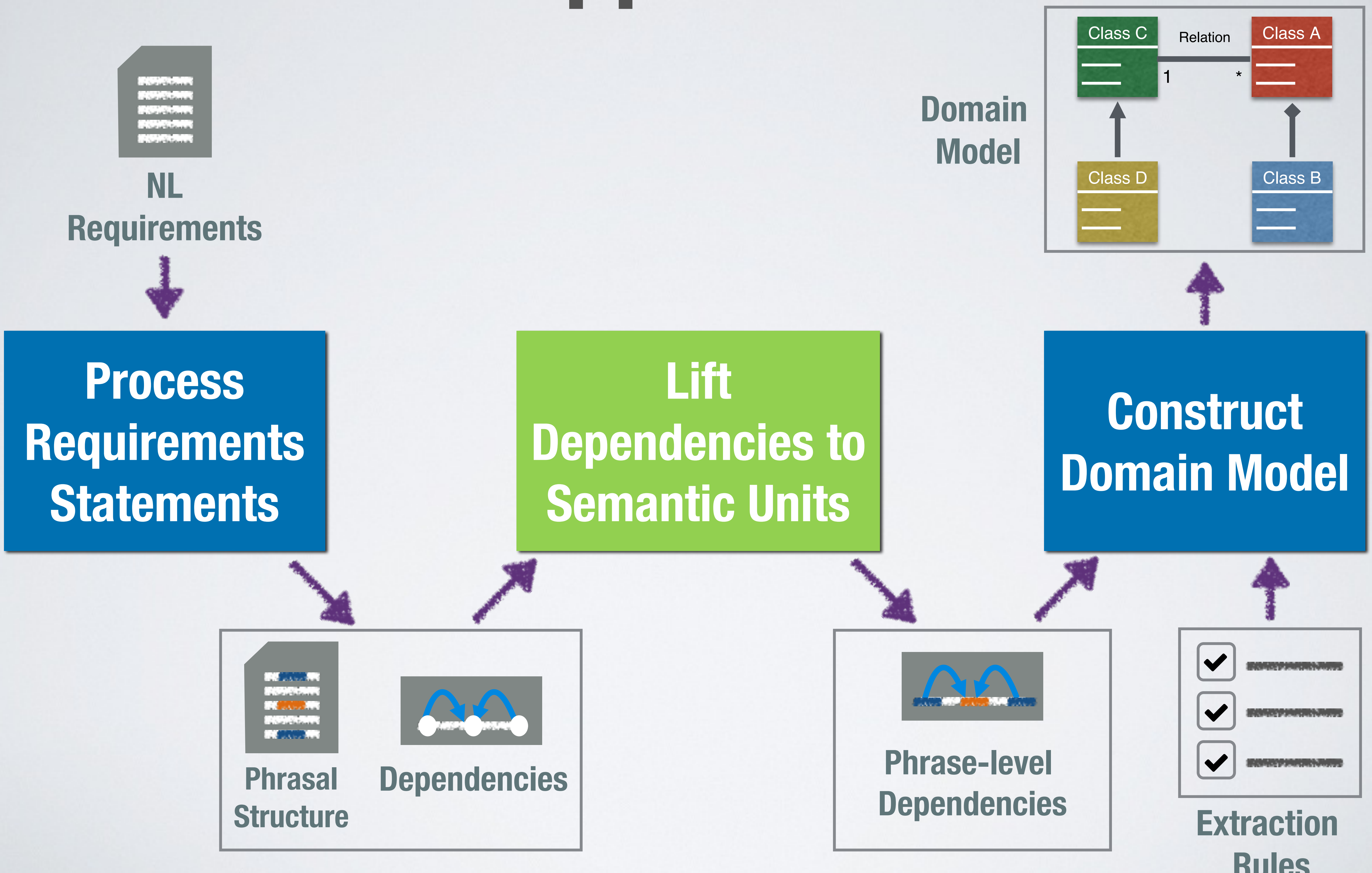
- **Multiple approaches exist** for extracting domain models or similar variants from requirements using **extraction rules**
  - Majority assume **specific structure**, e.g., restricted NL
  - Extraction of **direct relations only** but not indirect ones
  - **Limited empirical results** on industrial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

Approach

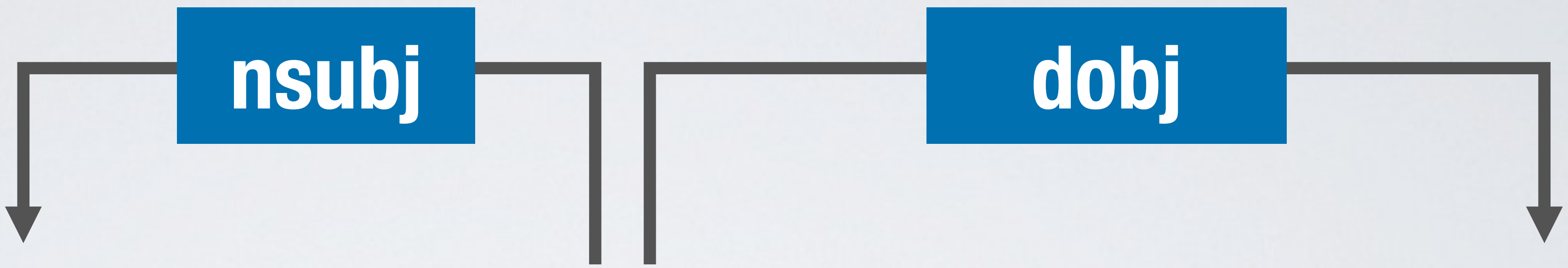


NL Requirements
Process Requirements Statements
Phrasal Structure
Dependencies
Lift Dependencies to Semantic Units
Phrase-level Dependencies
Construct Domain Model
Extraction Rules
Domain Model
Class C
Relation
Class A
1
*
Class D
Class B

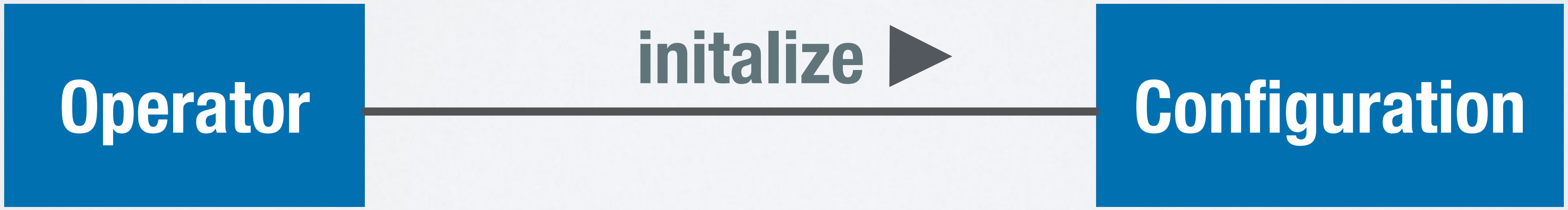
Approach


NL Requirements
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

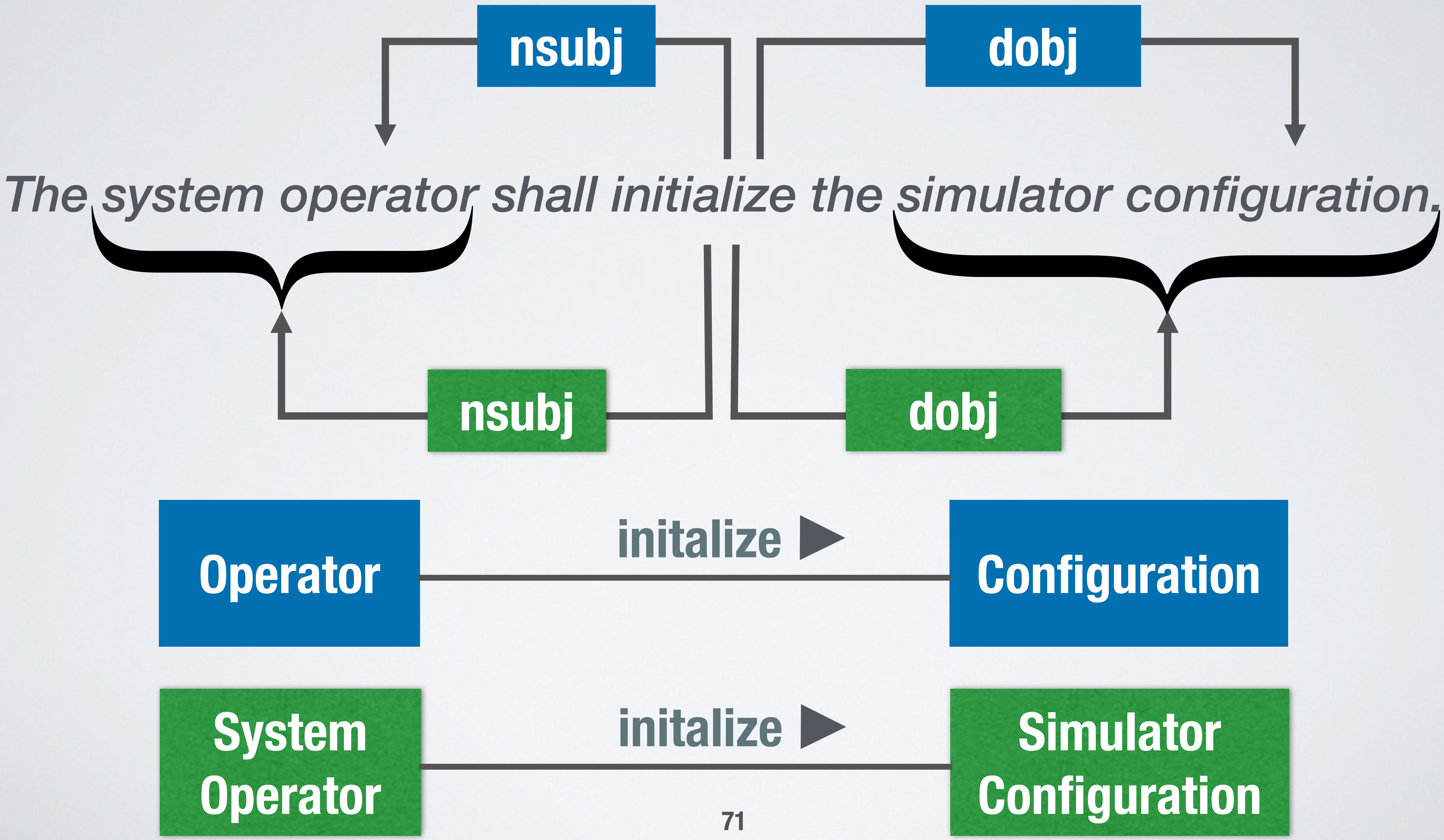
Grammatical Dependencies



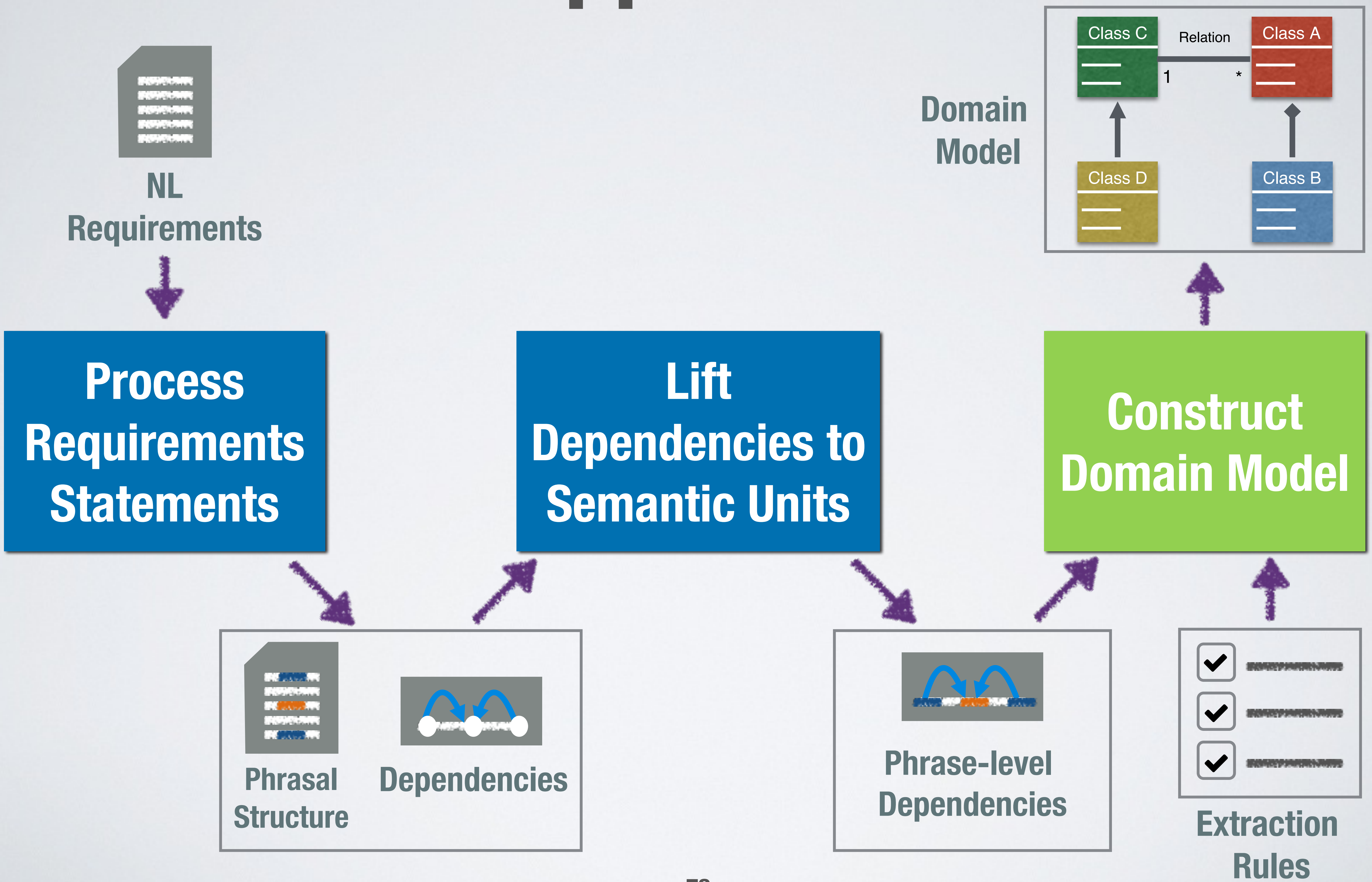
The system operator shall initialize the simulator configuration.



Lift Dependencies to Semantic Units



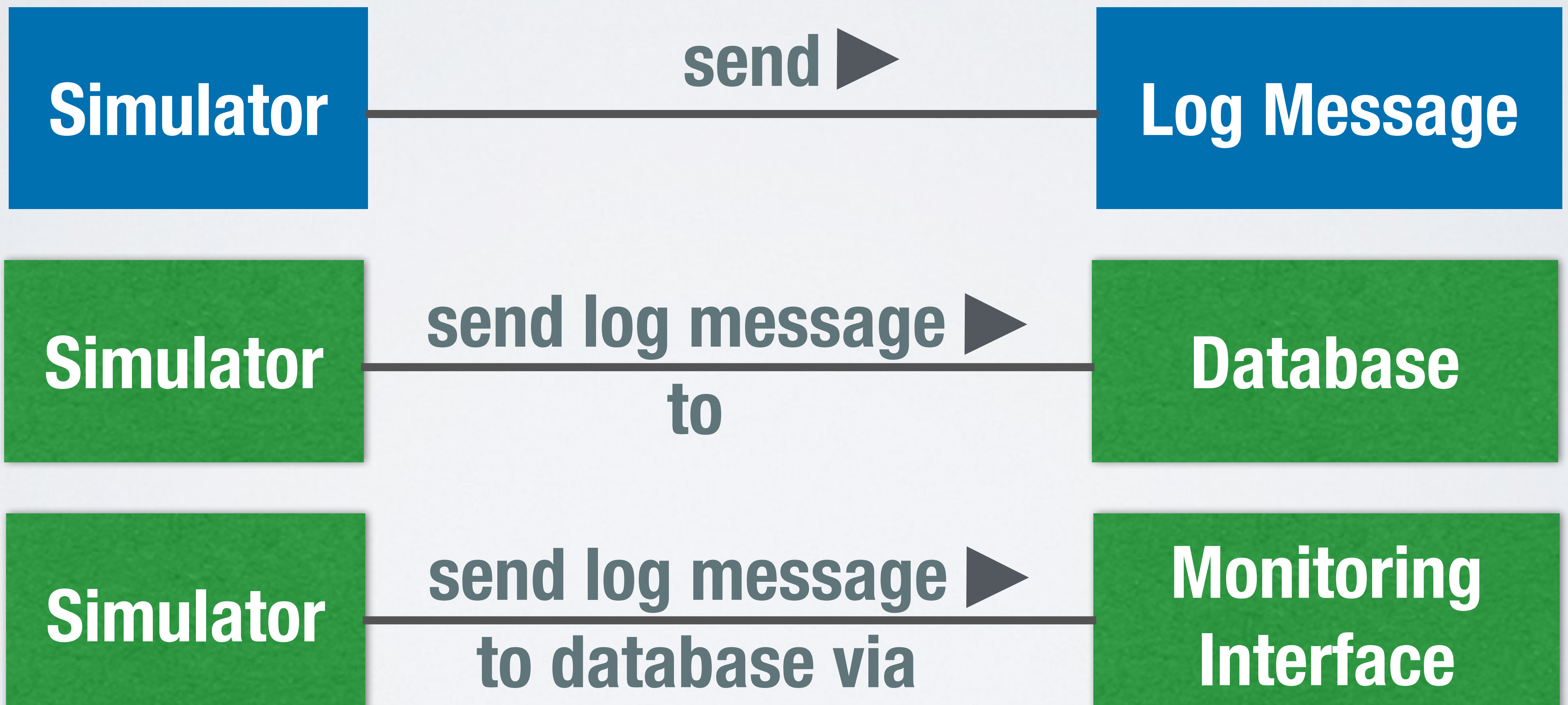
Approach



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

SES
your satellite company

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

Traceability

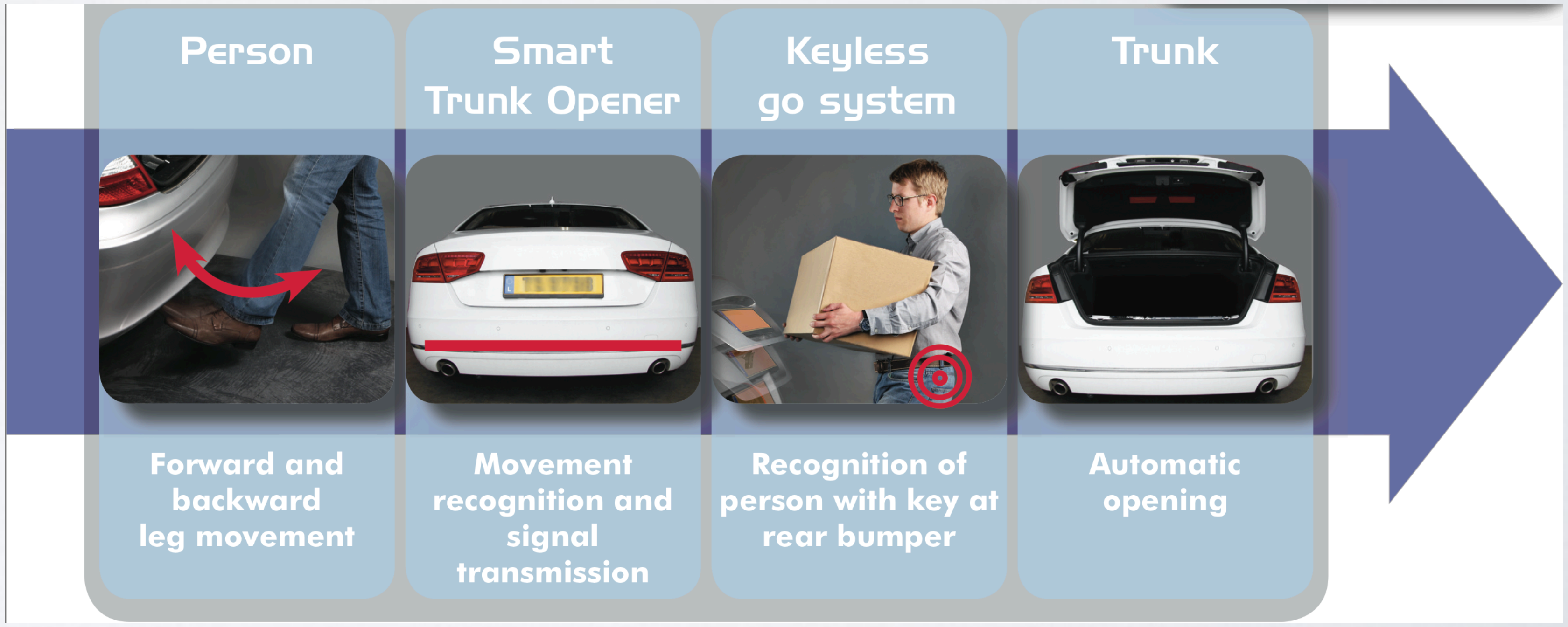
- In many domains, various types of traceability are required
- For example, in automotive (ISO 26262), traceability between requirements and system tests: requirements-driven testing
- Many requirements, many tests, therefore many traces …
- Automation is required

Context

IEE develops real-time embedded systems:

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

International Electronics & Engineering (IEE)



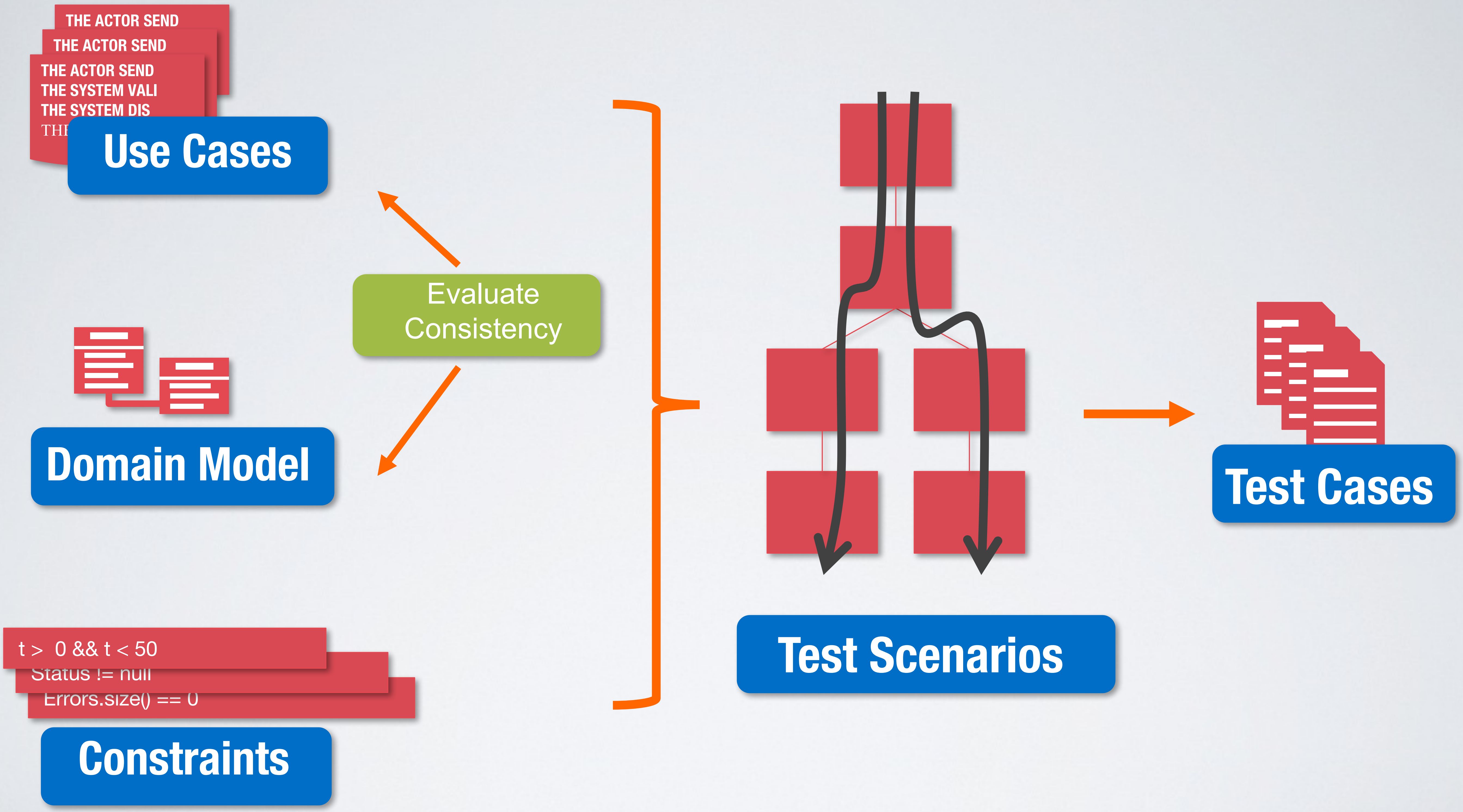
Objectives

- Support generation 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

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**
- **Conformance** is supported by a tool based on NLP

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.

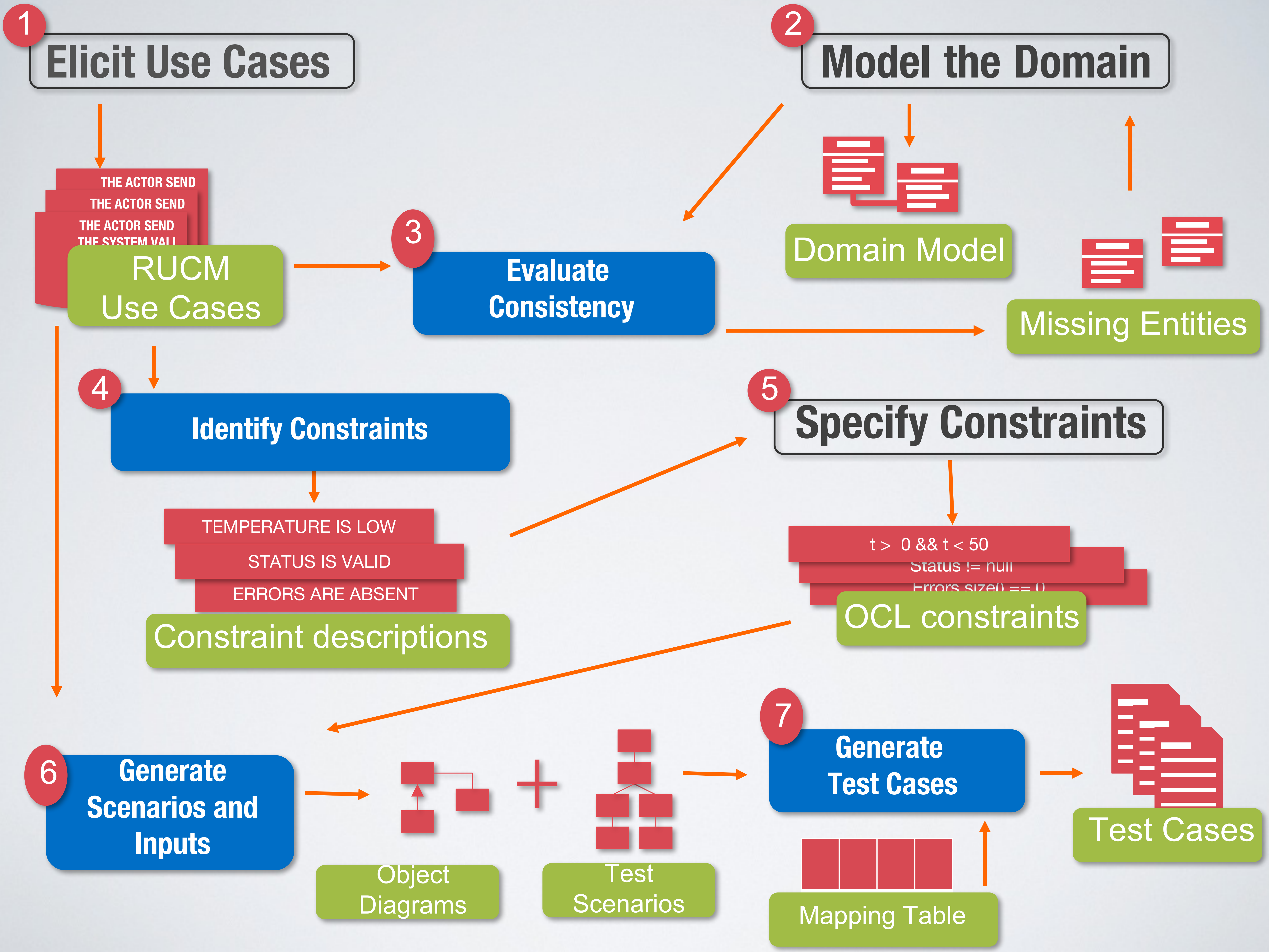
Specific Alternative Flow

RFS 3

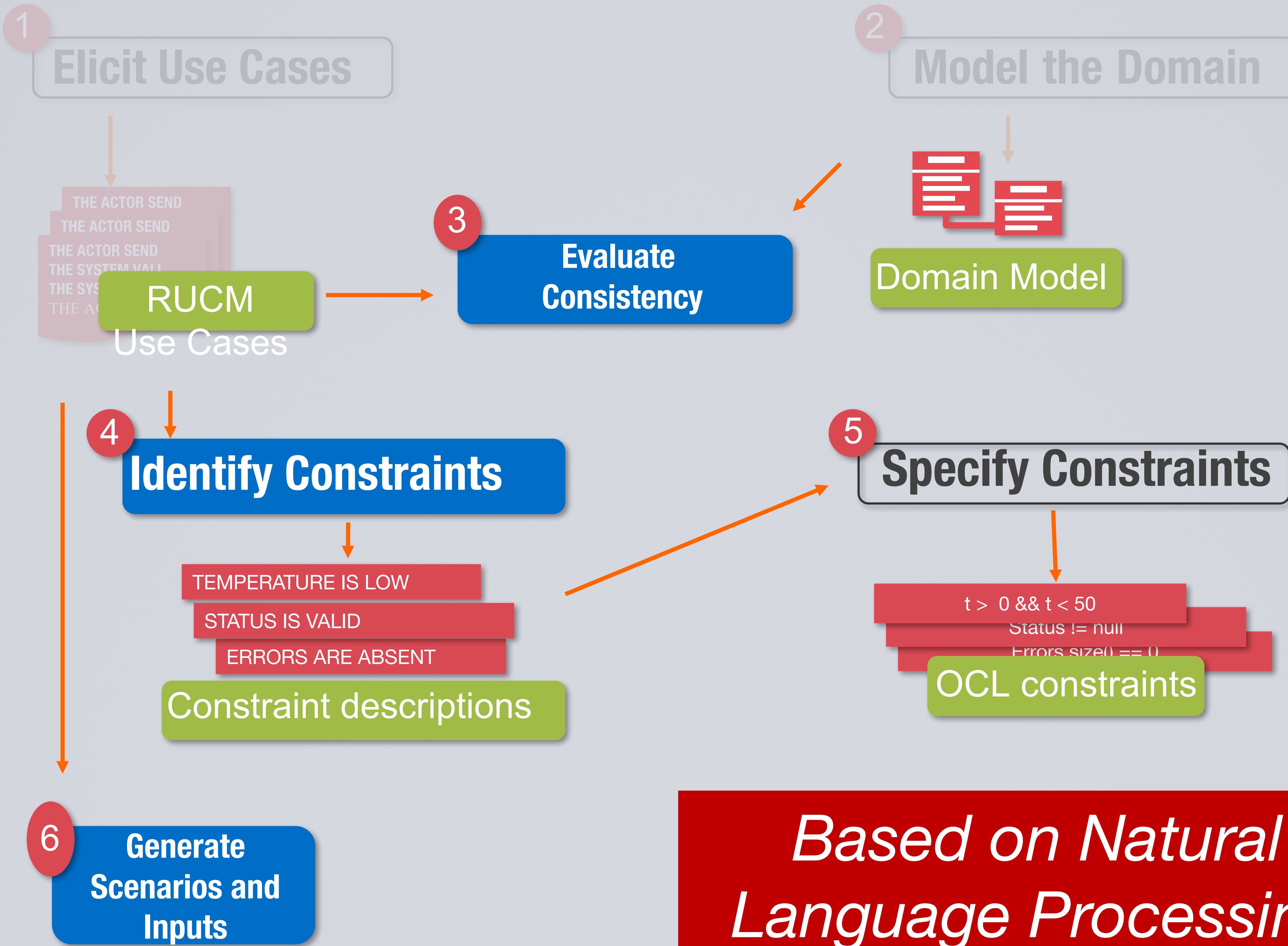
Postcondition: The previous occupant class for airbag control has been sent.

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
3
Evaluate
Consistency
Missing Entities
4
Identify Constraints
TEMPERATURE IS LOW
STATUS IS VALID
ERRORS ARE ABSENT
Constraint descriptions
5
Specify Constraints
t > 0 && t < 50
Status != null
OCL constraints
6
Generate
Scenarios and
Inputs
Object
Diagrams
Test
Scenarios
7
Generate
Test Cases
Mapping Table
Test Cases



Based on Natural Language Processing

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

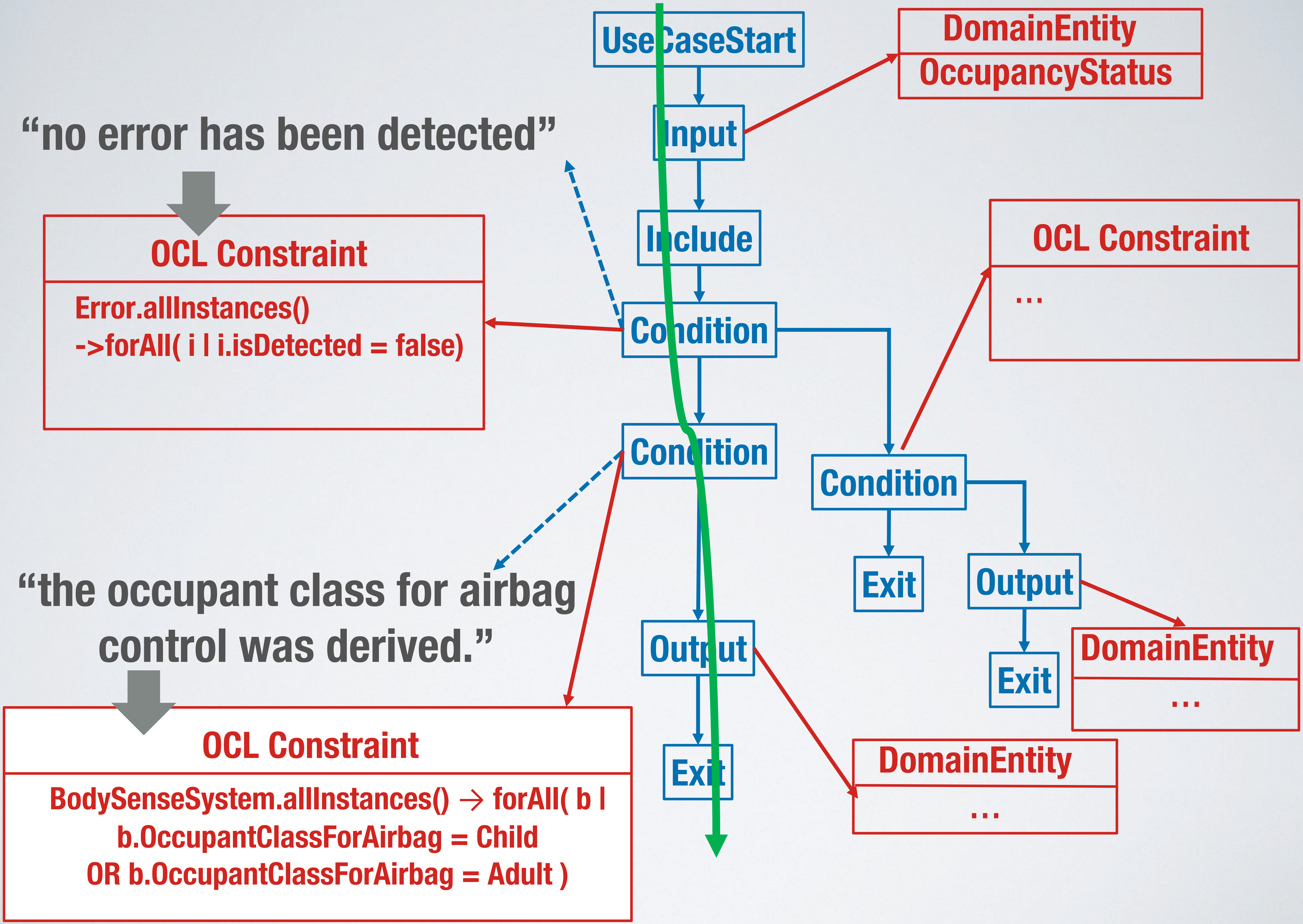
   DOMAIN ENTITY

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

   DOMAIN ENTITY

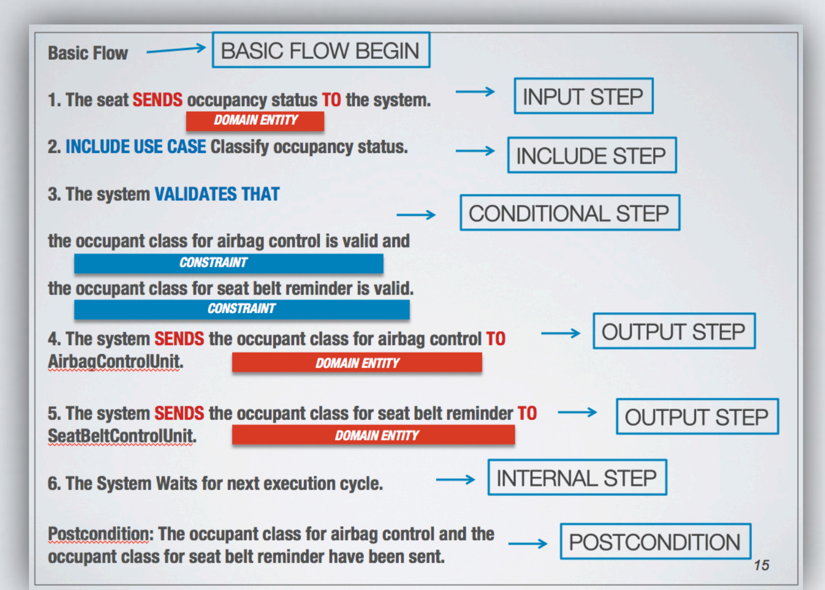
6. The System Waits for next execution cycle. → INTERNAL STEP

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



UseCaseStart
DomainEntity
OccupancyStatus
Input
“no error has been detected”
Include
OCL Constraint
OCL Constraint
Error.allInstances()
->forAll(i | i.isDetected = false)
Condition
...
Condition
Condition
Exit
Output
“the occupant class for airbag
control was derived.”
Output
DomainEntity
Exit
...
OCL Constraint
Exit
DomainEntity
BodySenseSystem.allInstances() → forAll(b |
b.OccupantClassForAirbag = Child
OR b.OccupantClassForAirbag = Adult)
...

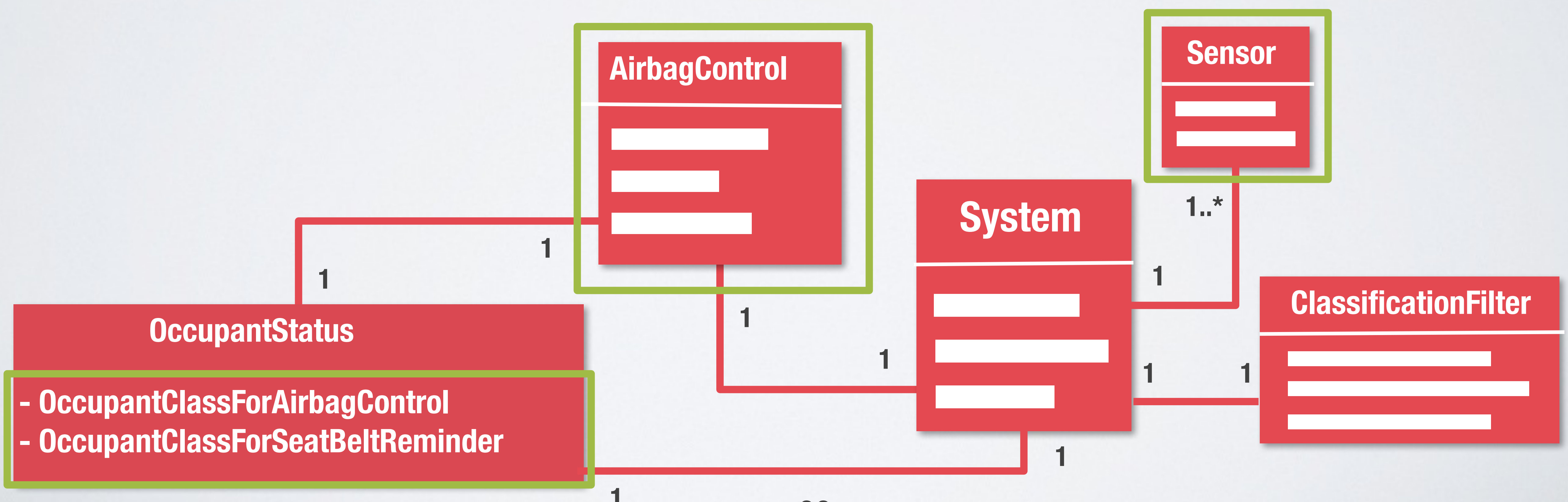
Evaluate Model Consistency


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

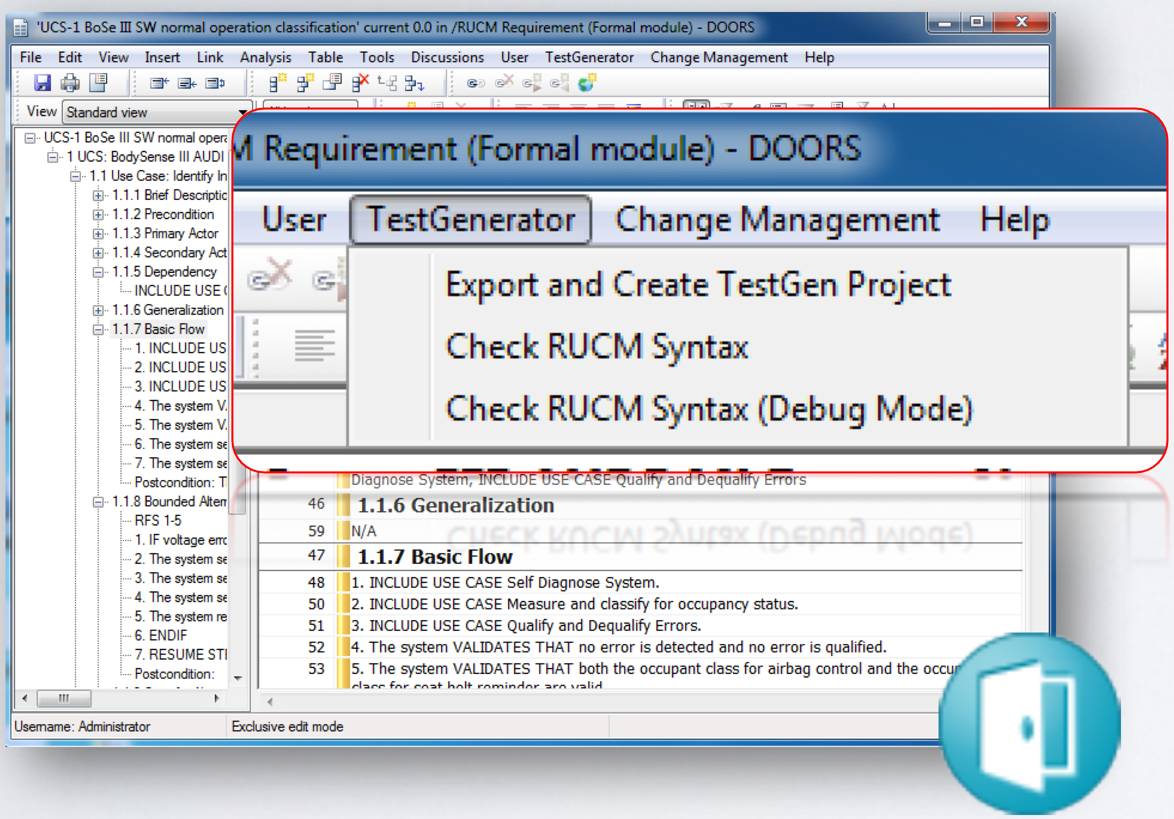
Tagged Use Case

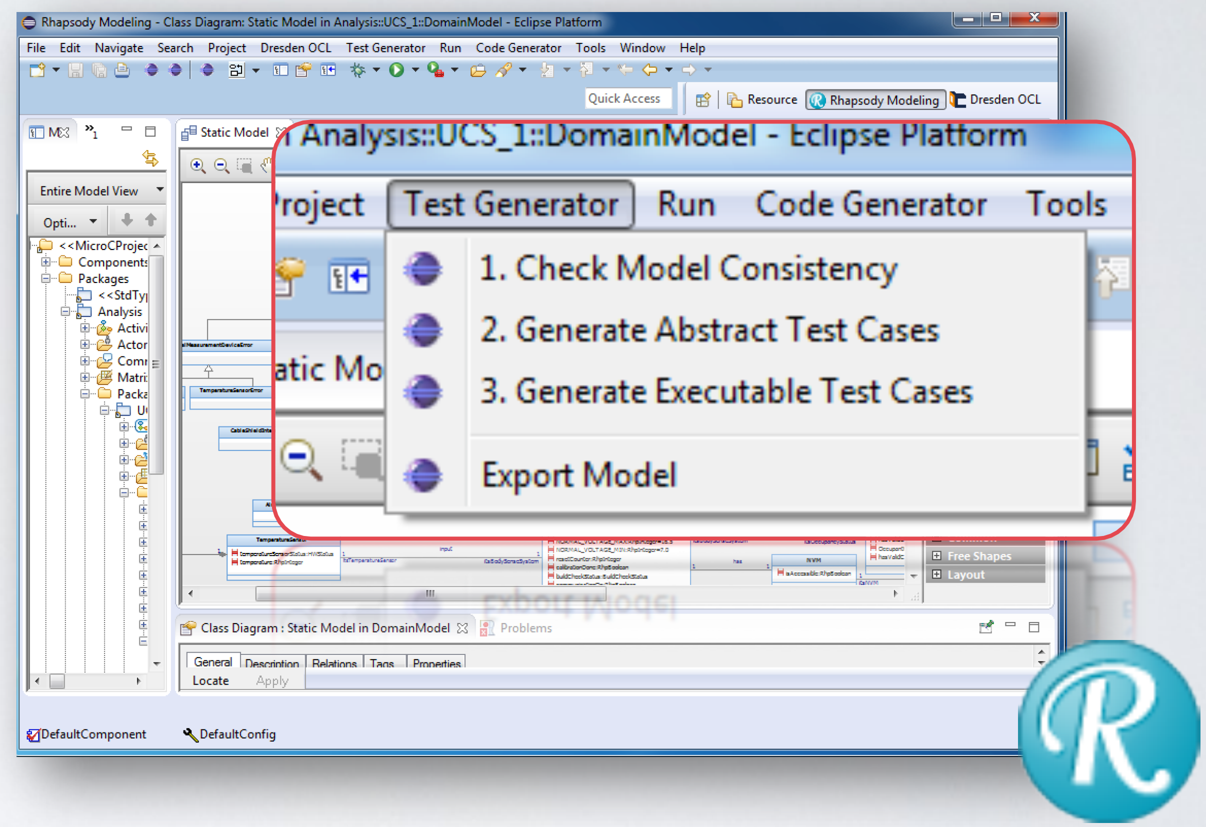
Airbag Control ✓
Classification Filter ?
Sensor ✓
Occupant Class for Airbag Control ✓
Occupant Class for Seat Belt Reminder ✓

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

Generating OCL Constraints

- May be a challenge in practice
- NLP: Semantic Role Labeling
- Determine the role of words in a sentence (e.g., affected actor)
- Match words with corresponding concepts in the domain model
- Generate an OCL formula

Semantic Role Labeling (SRL)

"no error has been detected"

A1 verb

Error.allInstances()->forAll(i | i.isDetected = false)

A1 verb

A0: actor that performs an activity

A1: actor that is affected by the activity described in a sentence

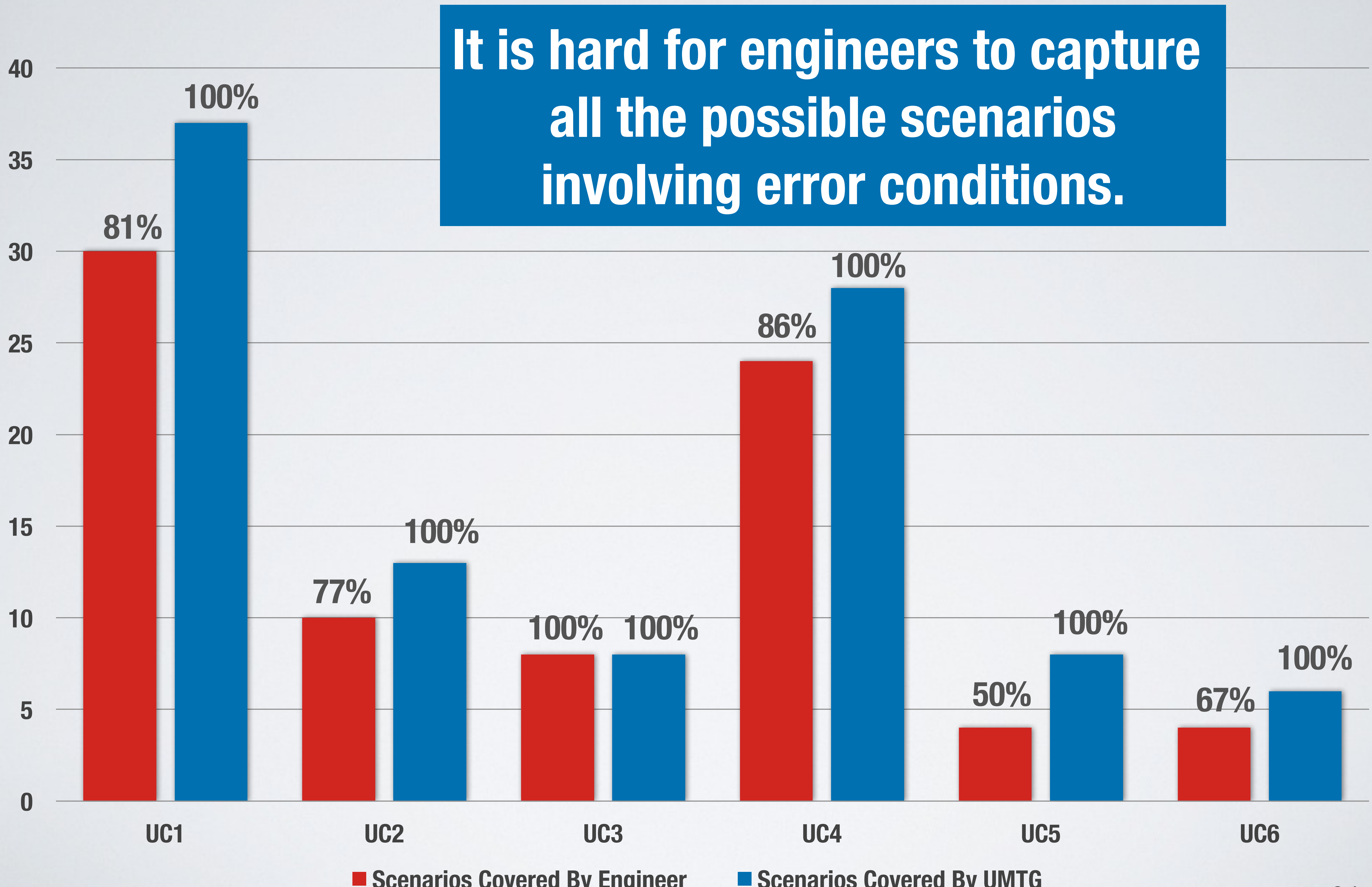
"The system detects temperature errors

A0 verb A1

TemperatureError.allInstances()->forAll(i | i.isDetected = true)

A1 verb

Effectiveness: scenarios covered



Supporting Product Lines and Requirements Configuration in Use-Case Driven Development

Configuring Requirements

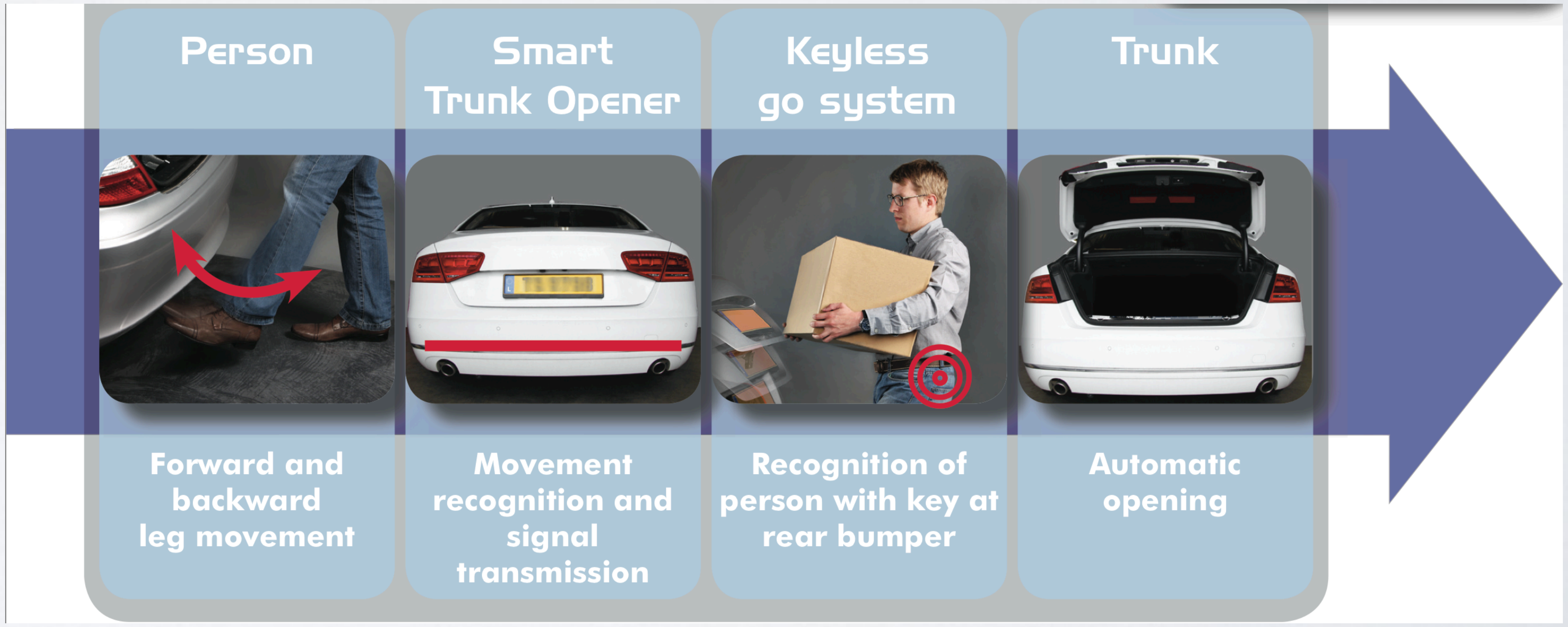
- Many software systems are part of product families targeting **varying needs among multiple customers**
- Requirements typically need to be tailored or configured for each customer
- Because of interdependencies among such decisions, this is often error-prone and complex
- **How do we support this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

Context

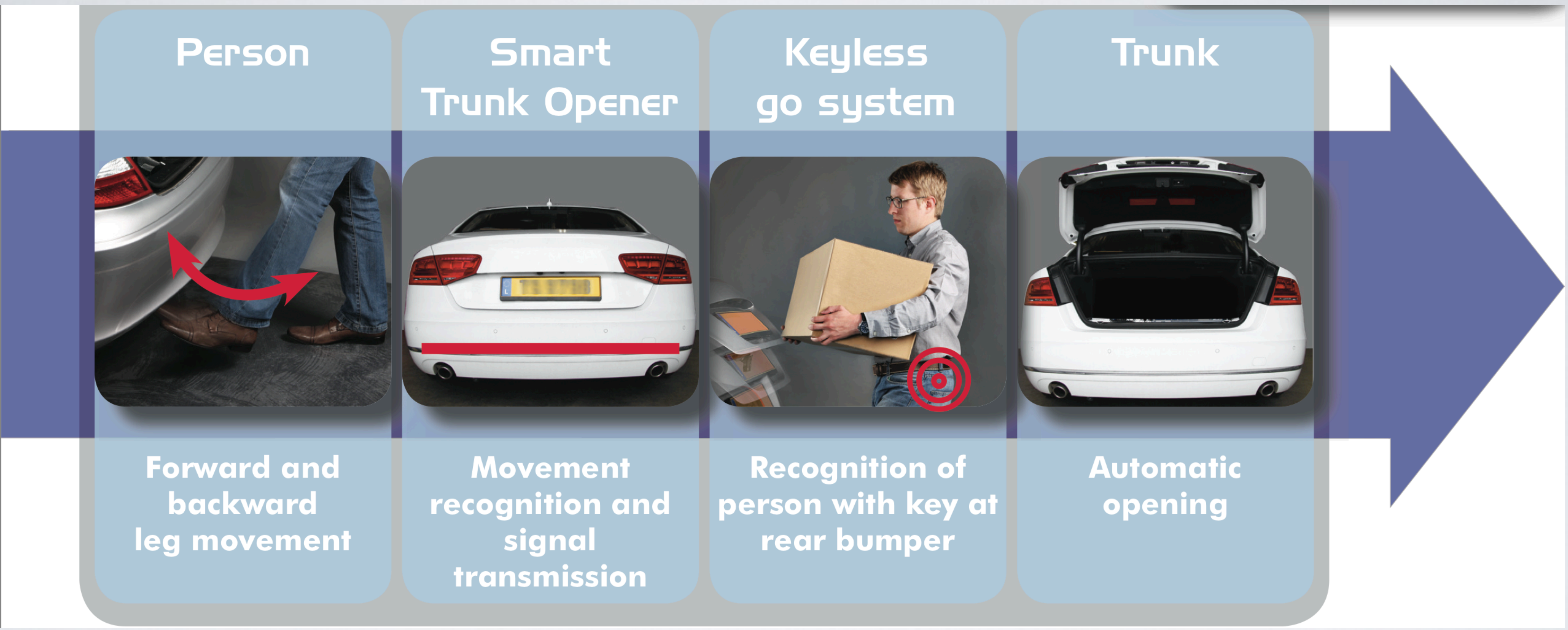
IEE develops real-time embedded systems:

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

International Electronics & Engineering (IEE)



Smart Trunk Opener (STO)



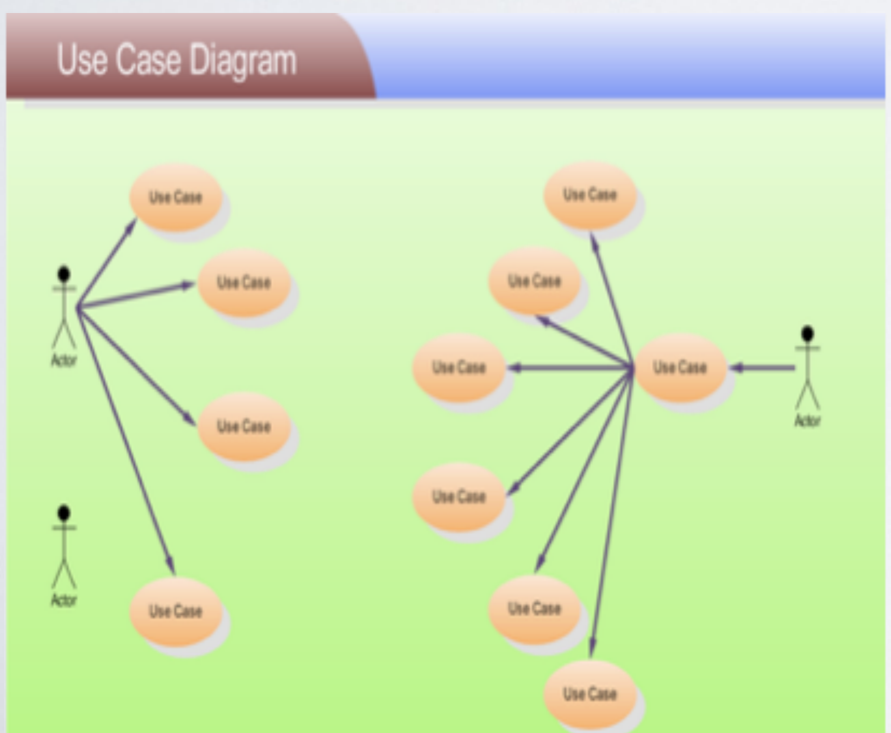
STO Provides automatic and hands-free access to a vehicle's trunk (based on a keyless entry system)

a sense for innovation IEE

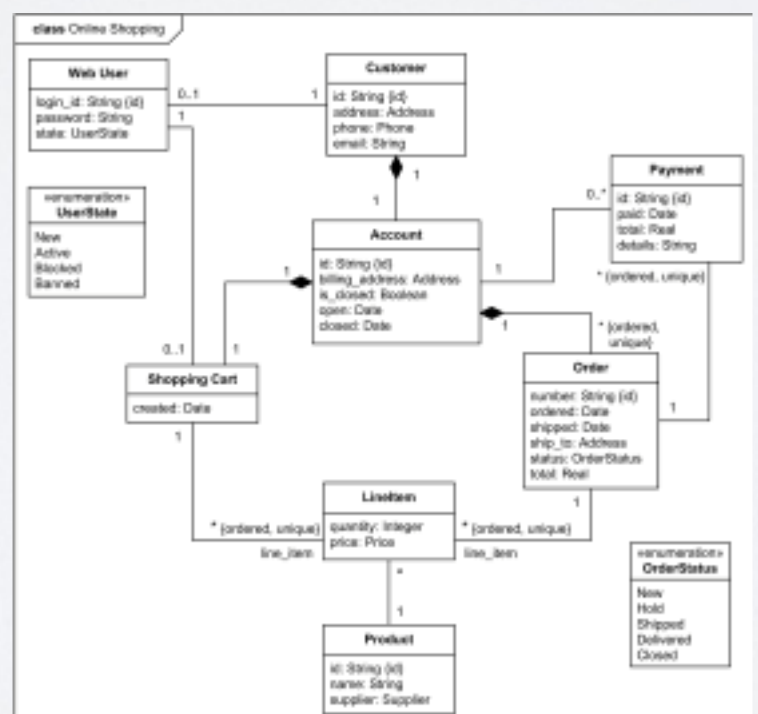
IEE Requirements Engineering

Use Case-Driven Development

Use Case Diagram



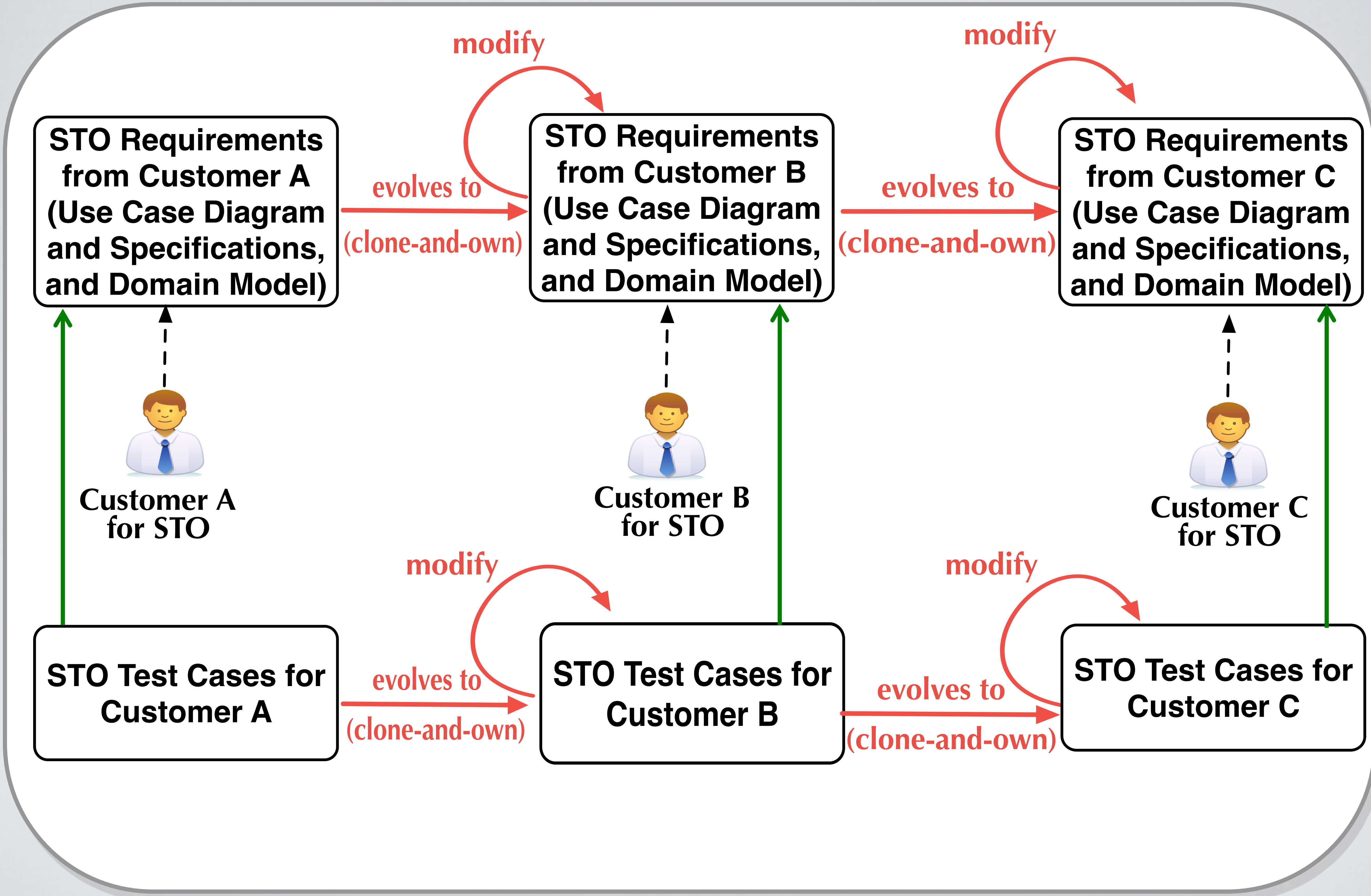
Domain Model



Use Case Specifications

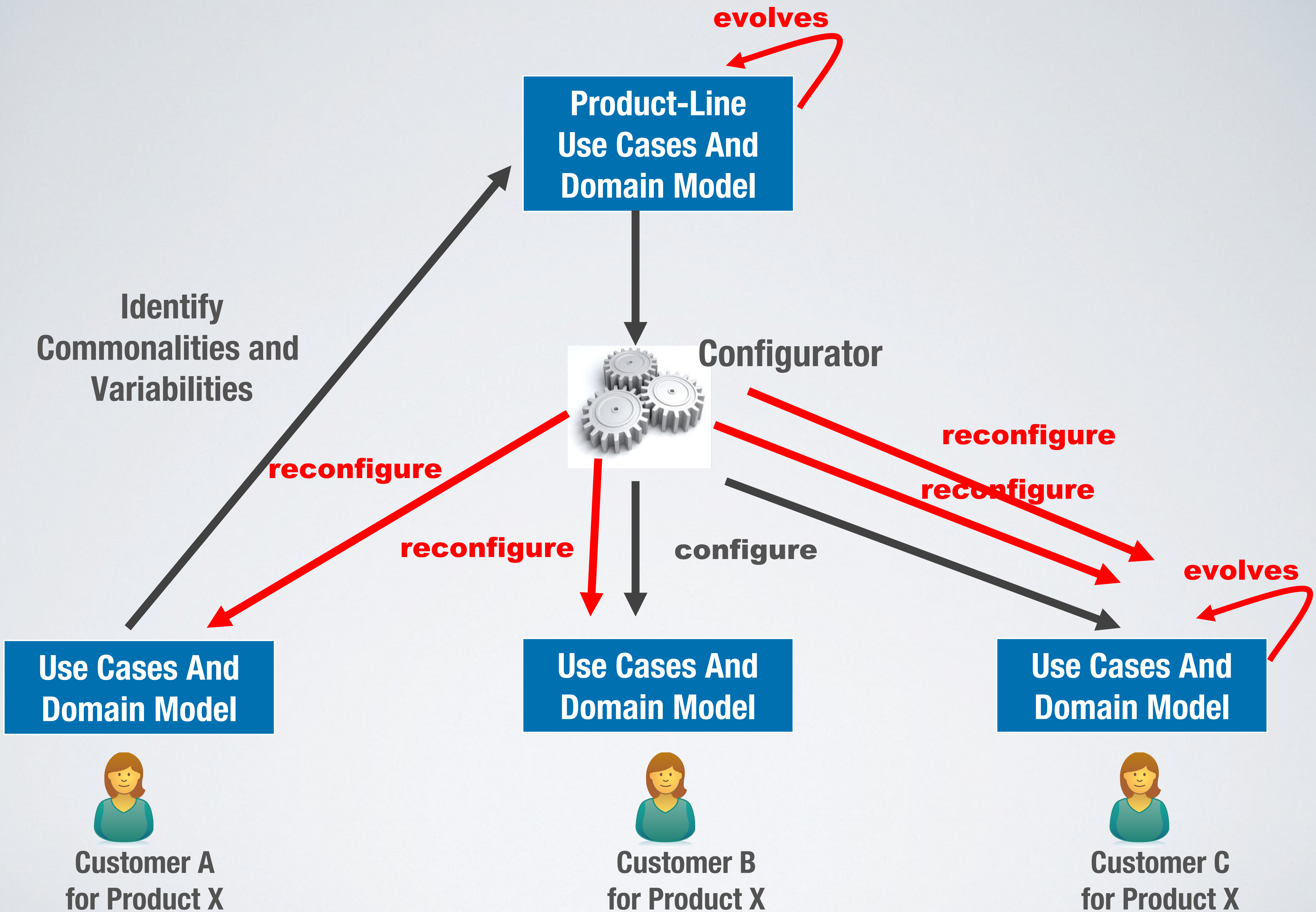
Use Case ID:	3
Use Case Name:	Deposit check
Actors:	Customer
Description:	Deposit cash without using ATM card by using E-Card system.
Preconditions:	1- The Customer has an activated E-Bank username and password. 2- The agreement should be signed by the customer. 3- The check must be valid.
Postconditions: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

Dealing with Multiple Customers

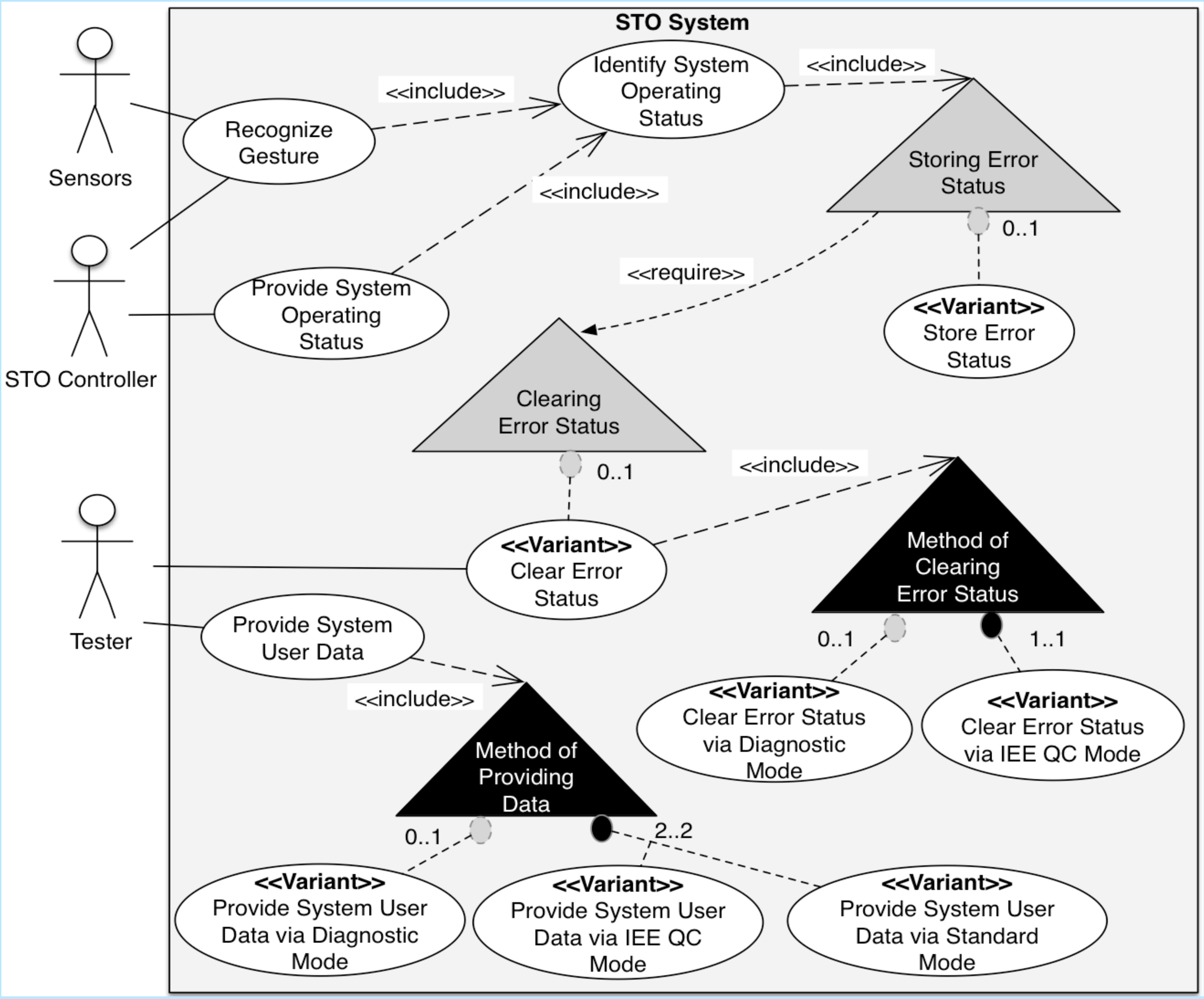


Product Line Approach

- A Product Line approach tailored to practice and with minimal overhead (adoption)
- Restricted and analyzable use case specifications (RUCM)
- No feature modeling!
- 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)



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/
- NLP is a key instrument
- 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

RE Applications

- Requirements to support a **shared understanding** among many stakeholders in large projects, e.g., 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., system testing
- Requirements to **support change control**
- Requirements to support **product-line configuration**

Forms of Requirements

- **Natural language** statements, complying or not with templates
- **Use case stories**, following various templates
- **Use case specifications**, possibly structured and restricted
- **Mixing models and NL**, 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

- No “right” way to express requirements

- Domain complexity and criticality
- Regulatory compliance, e.g., standards
- Project size, team distribution, and number of stakeholders
- Background of stakeholders and communication challenges
- Presence of product lines with multiple customers
- Importance of early contractual agreement
- Frequency and consequences of changes in requirements

Automation is required to justify the cost of rigorous requirements engineering

In most cases, we don’t have practical and scalable automation solutions

Conclusions

- NLP technology now provides many opportunities for automation
- But more attention to NL requirements analysis is needed in research
- Many applications, diversity of contexts and types of requirements
- Account for practicality and scalability
- More (reported) industrial experiences, as working assumptions play a key role

Acknowledgements

- Mehrdad Sabetzadeh
- Chetan Arora
- Fabrizio Pastore
- Chunhui Wang
- Arda Goknil
- Ines Hajri
- Shiva Nejati

Analyzing Natural-Language Requirements:

Industrial Needs and Scalable Solutions

UU/SIKS Symposium on Natural Language in Requirements Engineering, 30 November 2017

Lionel Briand
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
- [Submitted] C. Wang et al., Automated Generation of Constraints from Use Case Specifications to Support System Testing

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