Precise and Complete Requirements? An Elusive Goal

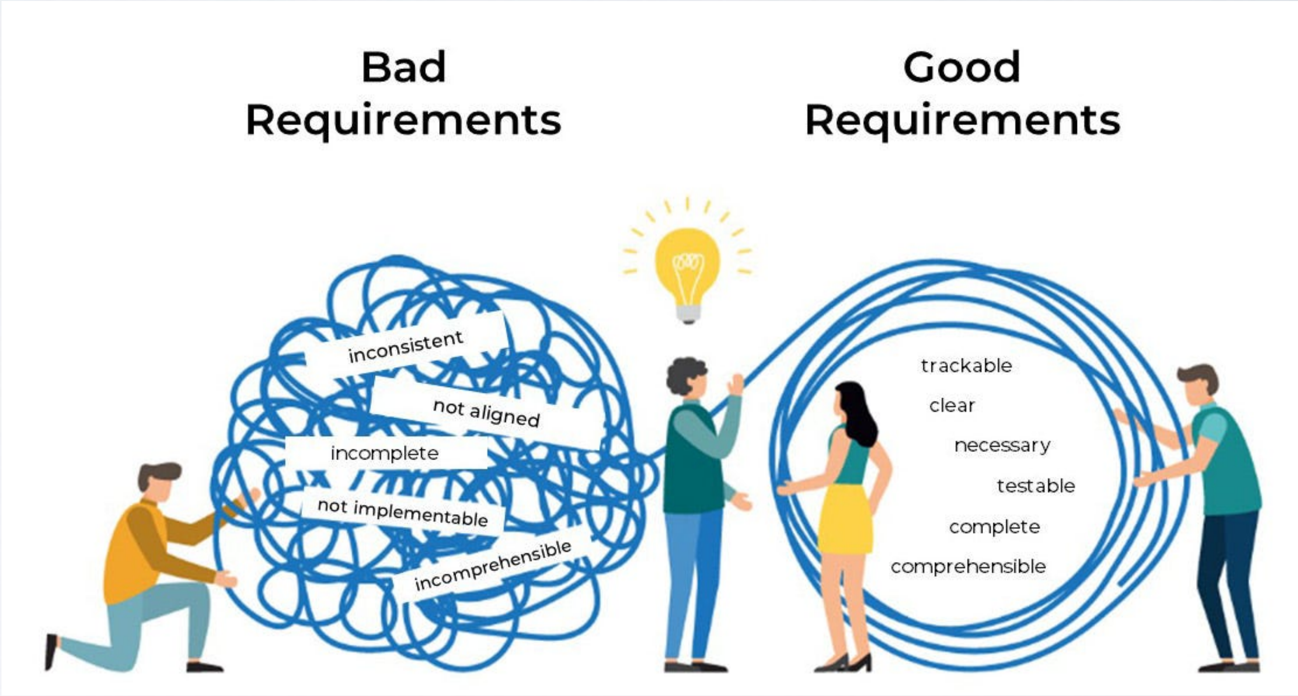
Lionel Briand

MO2RE 2024 Keynote

http://www.lbriand.info

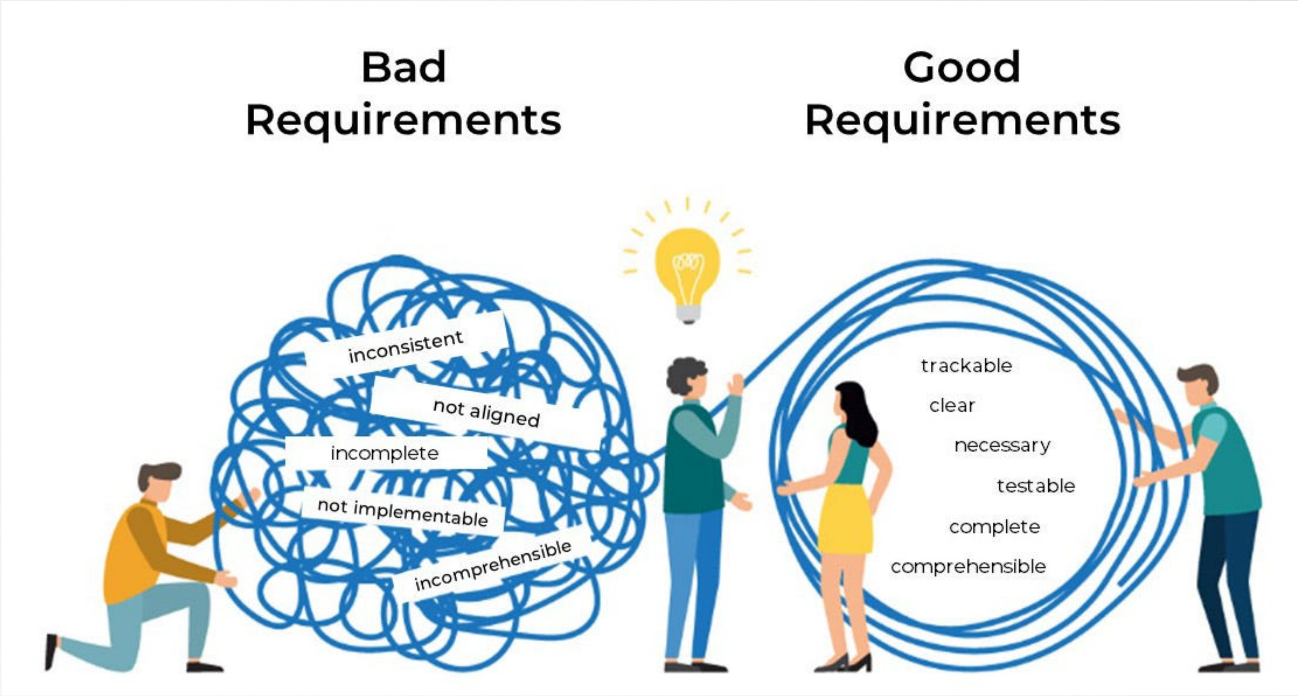
Canada Research Chairs
Chaires de recherche du Canada
Canada

Academic Assumptions



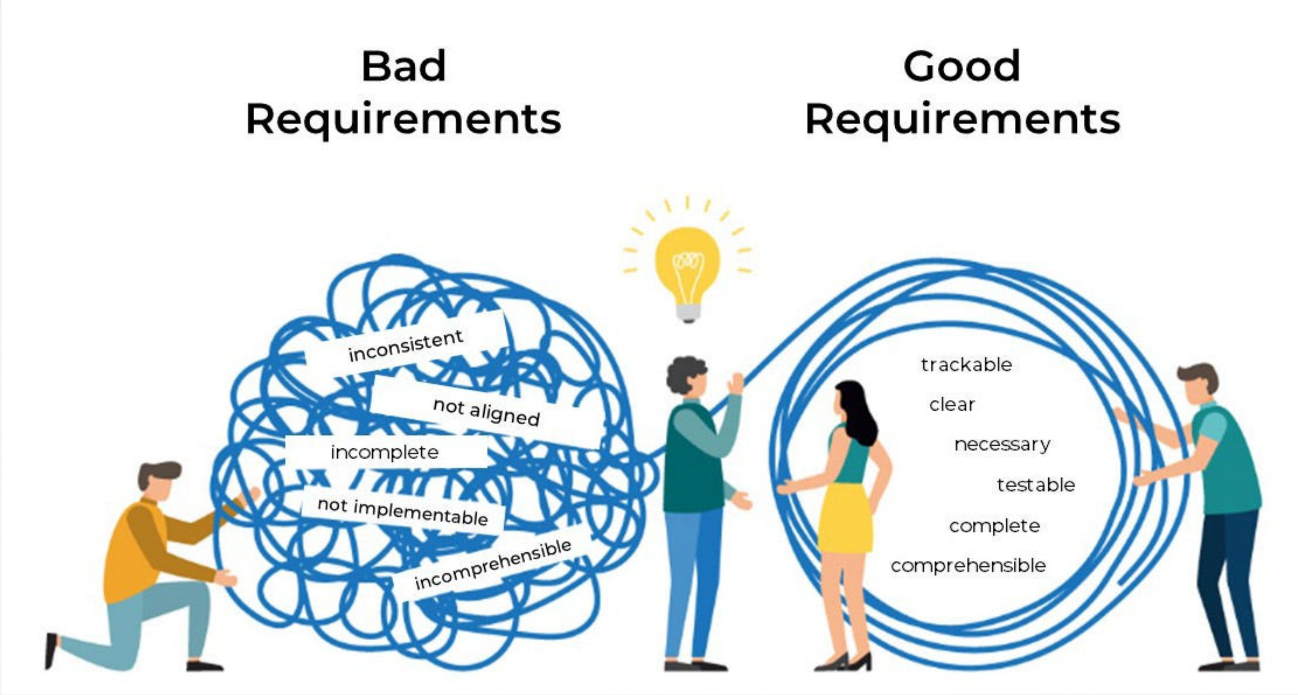
Academic Assumptions

- **Effective software development requires documented, precise, complete requirements up-front.**



Academic Assumptions

- Effective software development requires documented, precise, complete requirements up-front.
- Such requirements must remain up-to-date all along development, up to deployment, and after that during system evolution.

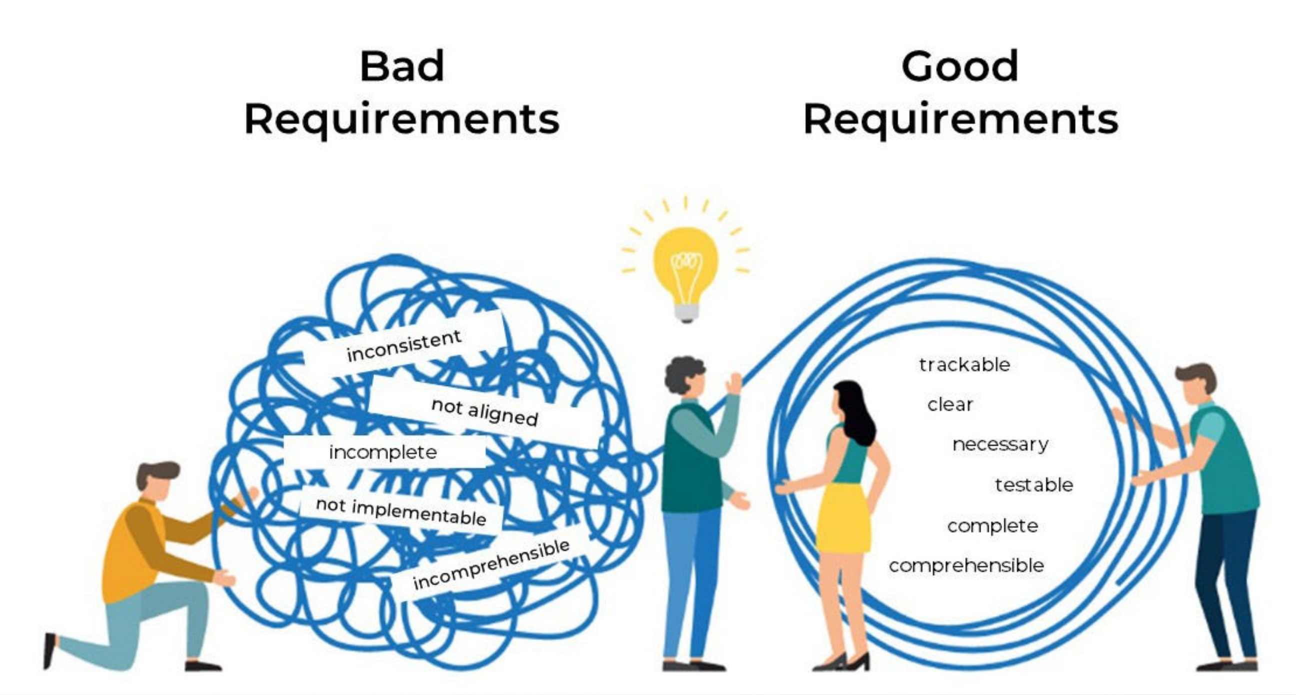


Academic Assumptions

- Effective software development requires documented, precise, complete requirements up-front.

- Such requirements must remain up-to-date all along development, up to deployment, and after that during system evolution.

- That sounds logical to many. (except agile folks), and drives (most of) academic research.



Quotes from Practice

Quotes from Practice

- “For many applications a highly detailed technical specification is just too much up front since a quick discussion could render their carefully typeset document useless. Instead, start with a vision. If everyone understands the overall picture then the requirements can get filled in along the way through discussions.”

Quotes from Practice

- "For many applications a highly detailed technical specification is just too much up front since a quick discussion could render their carefully typeset document useless. Instead, start with a vision. If everyone understands the overall picture then the requirements can get filled in along the way through discussions."
- "Steve Jobs believed that customers cannot describe exactly what they want the future products to look like, so it is your job to deliver them. So, unless you deliver custom software all the time, forget formal specs and start by creating prototypes…"

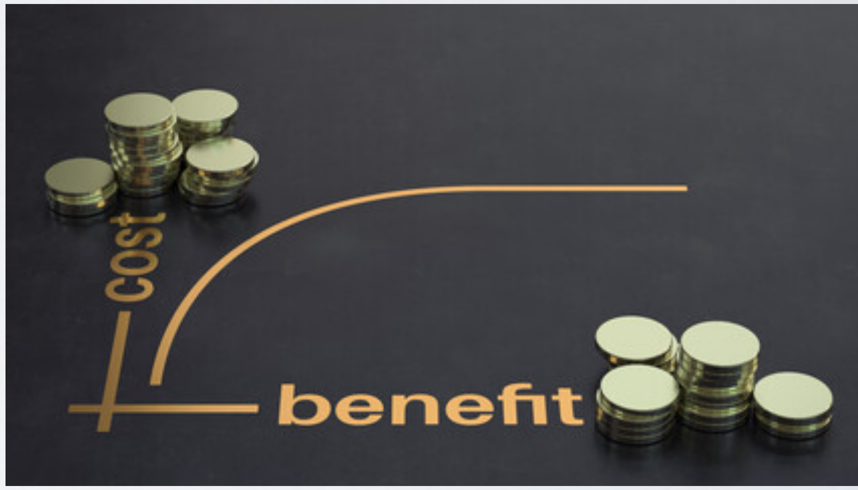
Quotes from Practice

- “For many applications a highly detailed technical specification is **just too much up front** since a quick discussion could render their carefully typeset document useless. Instead, **start with a vision**. If everyone understands the overall picture then the requirements can get filled in along the way **through discussions**.”
- “Steve Jobs believed that customers cannot describe exactly what they want the future products to look like, so it is your job to deliver them. So, unless you deliver custom software all the time, **forget formal specs and start by creating prototypes**…”
- “Unless you’re building something to a specification, you usually only have **a vague idea of what the final product should look like** at the time you start a project.”

Quotes from Practice

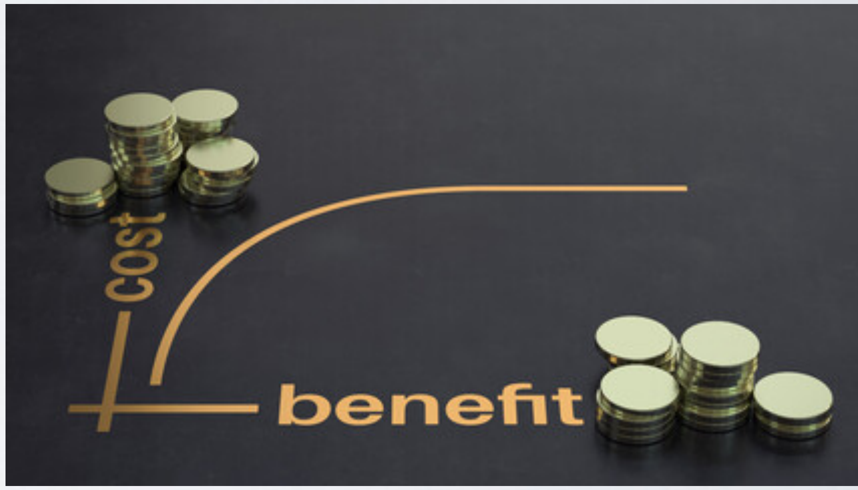
- "For many applications a highly detailed technical specification is just too much up front since a quick discussion could render their carefully typeset document useless. Instead, start with a vision. If everyone understands the overall picture then the requirements can get filled in along the way through discussions."
- "Steve Jobs believed that customers cannot describe exactly what they want the future products to look like, so it is your job to deliver them. So, unless you deliver custom software all the time, forget formal specs and start by creating prototypes…"
- "Unless you're building something to a specification, you usually only have a vague idea of what the final product should look like at the time you start a project."

State of Practice



State of Practice

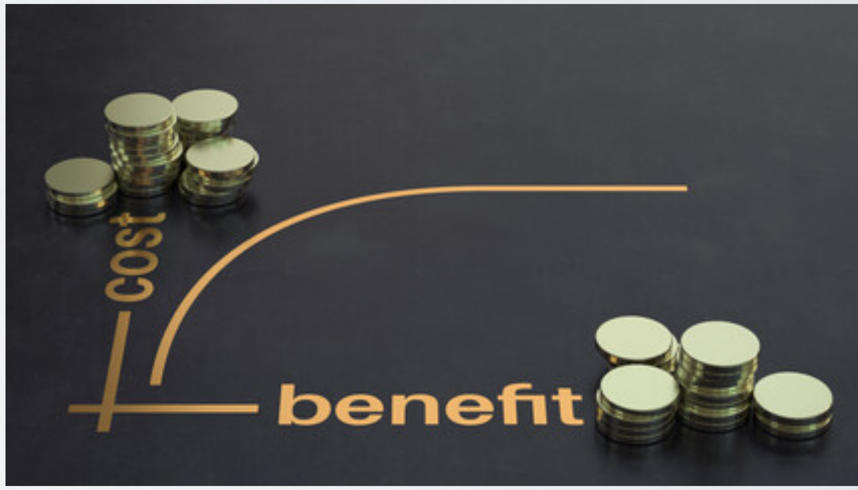
- In practice resources and time are limited, uncertainty about requirements is high



State of Practice

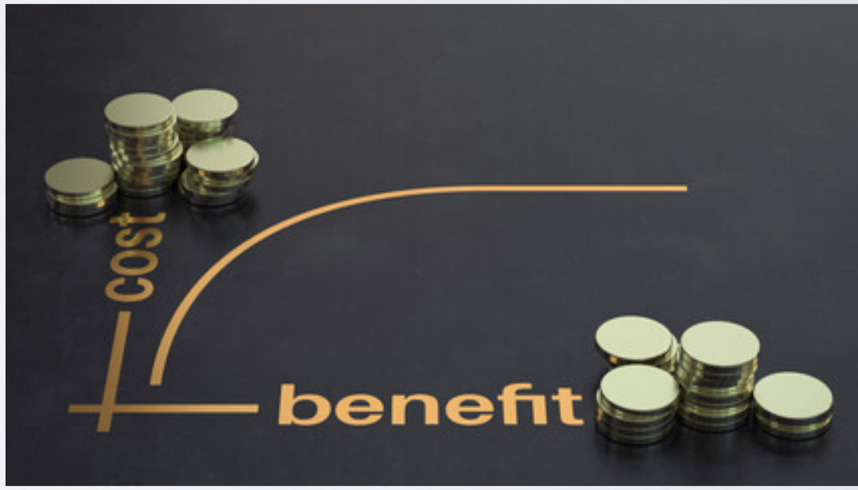
- In practice resources and time are limited, uncertainty about requirements is high

- Trade-offs are unavoidable



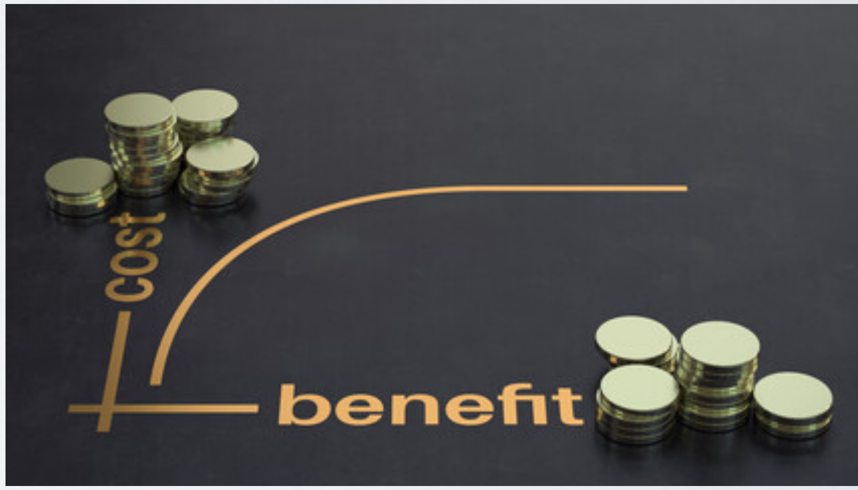
State of Practice

- In practice resources and time are limited, uncertainty about requirements is high
- Trade-offs are unavoidable
- Often, requirements are subject to such trade-offs, leading to imprecise, incomplete, or obsolete (documented) requirements



State of Practice

- In practice resources and time are limited, uncertainty about requirements is high
- Trade-offs are unavoidable
- Often, requirements are subject to such trade-offs, leading to imprecise, incomplete, or obsolete (documented) requirements
- Diversity: “RE practice differs according to the types of organization developing software, the types of products being developed, and the particular application domain of the product. “ (Lui et al. 2010)



Special Situations

Special Situations

- Requirements serve as a **contract**, to be validated during acceptance testing

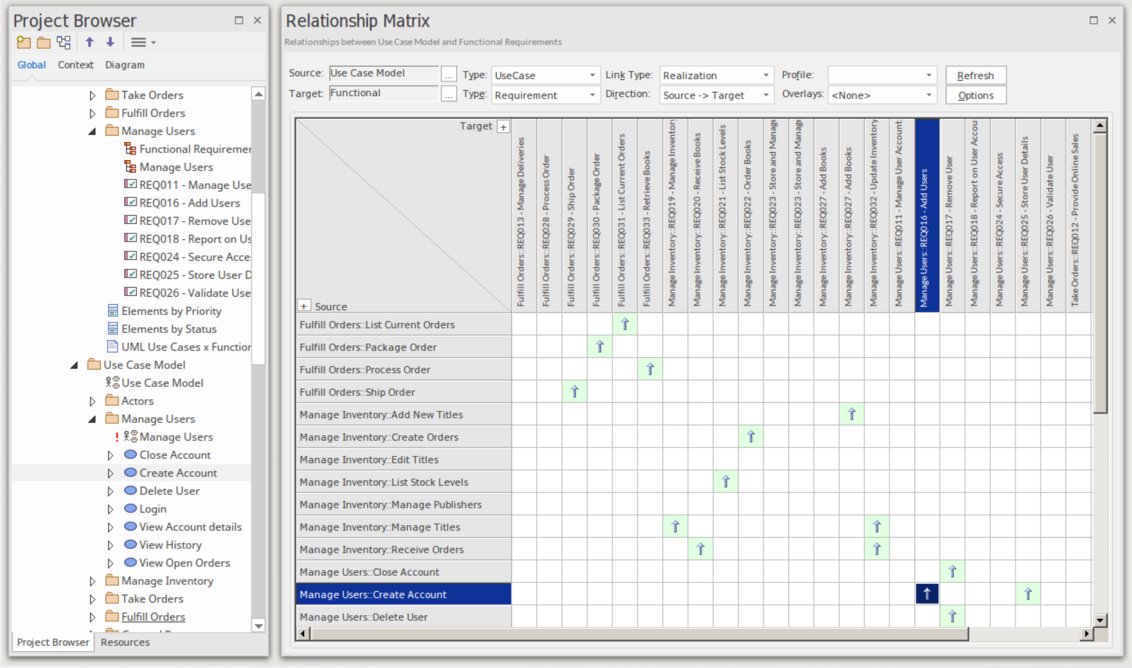
Special Situations

- Requirements serve as a contract, to be validated during acceptance testing
- Requirements traceability is required by standards or regulations, e.g., safety critical systems

Special Situations

- Requirements serve as a **contract**, to be validated during acceptance testing

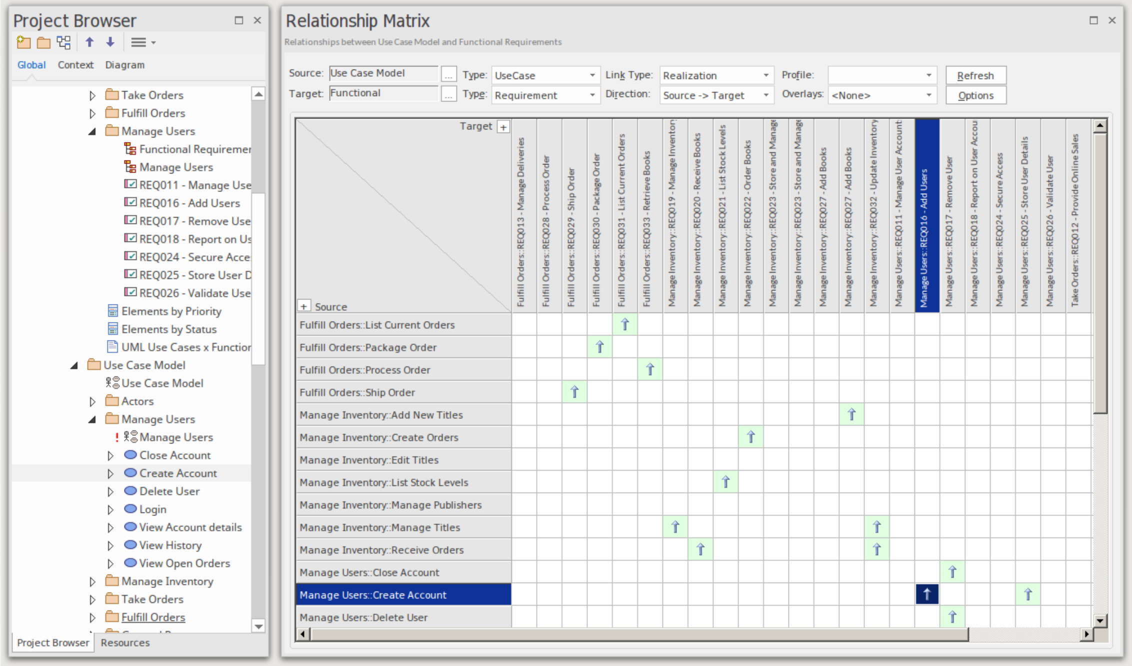
- Requirements **traceability** is required by standards or regulations, e.g., safety critical systems



Enterprise Architect
Sparx Systems

Special Situations

- Requirements serve as a **contract**, to be validated during acceptance testing
- Requirements **traceability** is required by standards or regulations, e.g., safety critical systems
- Even then, the precision and completeness of requirements are often **elusive goals**



Enterprise Architect
Sparx Systems

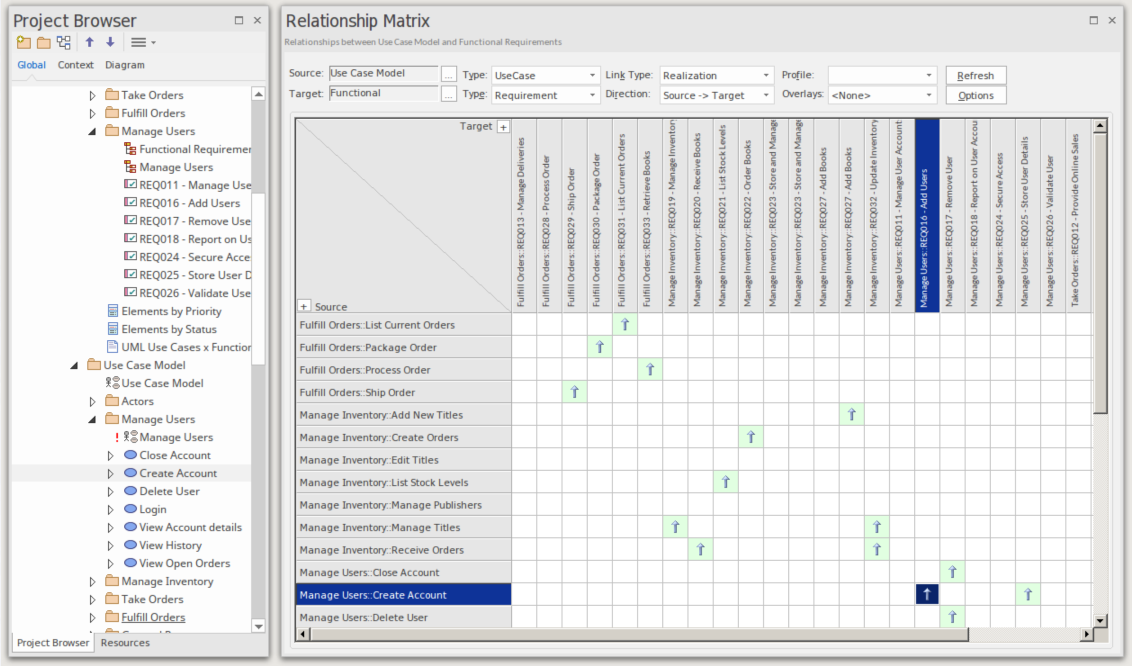
Special Situations

- Requirements serve as a **contract**, to be validated during acceptance testing

- Requirements **traceability** is required by standards or regulations, e.g., safety critical systems

- Even then, the precision and completeness of requirements are often **elusive goals**

- **The goal is only to achieve certification/regulatory compliance**



Enterprise Architect
Sparx Systems

Challenges: Liu et al. (2010)

Challenges: Liu et al. (2010)

- Customers do not have a clear understanding of system requirements themselves, including scope of the system, major functional features and nonfunctional attributes.

Challenges: Liu et al. (2010)

- Customers do not have a clear understanding of system requirements themselves, including scope of the system, major functional features and nonfunctional attributes.

- Users' needs and understanding constantly change.

Challenges: Liu et al. (2010)

- Customers do not have a clear understanding of system requirements themselves, including scope of the system, major functional features and nonfunctional attributes.

- Users' needs and understanding constantly change.

- Project schedule is too tight to allow sufficient interaction and learning period between customer and development team.

Challenges: Liu et al. (2010)

- Customers do not have a clear understanding of system requirements themselves, including scope of the system, major functional features and nonfunctional attributes.

- Users' needs and understanding constantly change.

- Project schedule is too tight to allow sufficient interaction and learning period between customer and development team.

- Broken communication links between customer, analyst and developer.

Challenges: Liu et al. (2010)

- Customers do not have a clear understanding of system requirements themselves, including scope of the system, major functional features and nonfunctional attributes.
- Users' needs and understanding constantly change.
- Project schedule is too tight to allow sufficient interaction and learning period between customer and development team.
- Broken communication links between customer, analyst and developer.

Liu et al., "Why Requirements Engineering Fails: A Survey Report from China", IEEE RE 2010

Recommendations: Liu et al.

Recommendations: Liu et al.

- **Link requirements with testing** and adopt a test-driven design process.

Recommendations: Liu et al.

- **Link requirements with testing** and adopt a test-driven design process.
- Develop RE tools that **better fit the real-world** needs of the customers and engineers.

Challenges: Martins and Gorschek (2020)

Challenges: Martins and Gorschek (2020)

- Focus on safety-critical systems, where precise and complete requirements are expected to be most common.

Challenges: Martins and Gorschek (2020)

- Focus on safety-critical systems, where precise and complete requirements are expected to be most common.
- All companies write their requirements documents in natural language. UML diagrams and other types of diagrams (as FTA and flowchart) are seldom used to describe the requirements.

Challenges: Martins and Gorschek (2020)

- Focus on safety-critical systems, where precise and complete requirements are expected to be most common.
- All companies write their requirements documents in natural language. UML diagrams and other types of diagrams (as FTA and flowchart) are seldom used to describe the requirements.
- Two companies used “formal methods”, but not from software engineering: Equations from control engineering

Challenges: Martins and Gorschek (2020)

- Focus on safety-critical systems, where precise and complete requirements are expected to be most common.
- All companies write their requirements documents in natural language. UML diagrams and other types of diagrams (as FTA and flowchart) are seldom used to describe the requirements.
- Two companies used “formal methods”, but not from software engineering: Equations from control engineering
- 61% use word processing and spreadsheets as RE tools

Challenges: Martins and Gorschek (2020)

- Focus on safety-critical systems, where precise and complete requirements are expected to be most common.
- All companies write their requirements documents in natural language. UML diagrams and other types of diagrams (as FTA and flowchart) are seldom used to describe the requirements.
- Two companies used "formal methods", but not from software engineering: Equations from control engineering
- 61% use word processing and spreadsheets as RE tools

Martins and Gorschek, "Requirements Engineering for Safety-Critical Systems: An Interview Study with Industry Practitioners", IEEE TSE 2020

Challenges: Martins and Gorschek (2020)

Challenges: Martins and Gorschek (2020)

- “When someone **does not understand a requirement** a person conversation is performed, usually people go to the team leader, systems engineers or safety engineer to get the understanding. This is done in an **informal way**.“

Challenges: Martins and Gorschek (2020)

- “When someone does not understand a requirement a person conversation is performed, usually people go to the team leader, systems engineers or safety engineer to get the understanding. This is done in an informal way.“

- “Practitioners from three companies mentioned that somehow it is necessary to produce more useful documents in order to meet the daily needs of the system developers. It seems that the requirements documents still are more to show compliance than to be really used by development teams.“

Challenges: Martins and Gorschek (2020)

- “When someone **does not understand a requirement** a person conversation is performed, usually people go to the team leader, systems engineers or safety engineer to get the understanding. This is done in an **informal way**.“

- “Practitioners from three companies mentioned that somehow it is necessary to produce more **useful documents in order to meet the daily needs of the system developers**. It seems that the requirements documents still are **more to show compliance** than to be really used by development teams.“

- “The **designers and programmers know what to do**, of course the requirements specification is there to drive it, but in details it is very common that the **requirements don’t really drive** the designers/programmers.”

Factors Affecting Practice

Factors Affecting Practice

- Lack of skills, training (often invoked by academics)

Factors Affecting Practice

- Lack of skills, training (often invoked by academics)
- Frequent requirements changes, uncertainty

Factors Affecting Practice

- Lack of skills, training (often invoked by academics)
- Frequent requirements changes, uncertainty
- Time pressure, lack of resources

Factors Affecting Practice

- Lack of skills, training (often invoked by academics)
- Frequent requirements changes, uncertainty
- Time pressure, lack of resources
- Lack of adequate tool support and automation

Factors Affecting Practice

- Lack of skills, training (often invoked by academics)
- Frequent requirements changes, uncertainty
- Time pressure, lack of resources
- Lack of adequate tool support and automation
- Result: Low return on investment (RoI) for precise, complete requirements (real or perceived?)

Widespread Situation

Widespread Situation

- **Informal, natural language requirements**

Widespread Situation

- Informal, natural language requirements
- Sometimes completed with informal diagrams (UML, SysML …)

Widespread Situation

- Informal, natural language requirements
- Sometimes completed with informal diagrams (UML, SysML …)
- Quality (precision, completeness, …) is relatively low

My Take

My Take

- Higher RoI for precise requirements would require effective support for:
  - Quality assurance
  - Change management
  - Traceability (e.g., to systems tests)

My Take

- Higher RoI for precise requirements would require effective support for:
  - Quality assurance
  - Change management
  - Traceability (e.g., to systems tests)
- Example projects next

Three Essential Technologies

- NL requirements QA
- NL requirements change impact (CI) analysis
- NL Requirements-driven acceptance testing

Requirements Quality Assurance

Requirements QA

Requirements QA

- **Manual QA is expensive and tedious**

Requirements QA

- Manual QA is expensive and tedious
- Especially in the context of frequent changes

Requirements QA

- Manual QA is expensive and tedious
- Especially in the context of frequent changes
- No systematic feedback

Requirements QA

- Manual QA is expensive and tedious
- Especially in the context of frequent changes
- No systematic feedback
- Leads to low quality requirements

Requirements QA

- Manual QA is expensive and tedious
- Especially in the context of frequent changes
- No systematic feedback
- Leads to low quality requirements
- Example project in the financial domain

Industrial Context and Artifacts

Industrial Context and Artifacts

- Global leading supplier of post-trading services

Industrial Context and Artifacts

- Global leading supplier of post-trading services
- 2500 customers in 110 countries

Industrial Context and Artifacts

- Global leading supplier of post-trading services
- 2500 customers in 110 countries
- Compliance with financial regulations

Industrial Context and Artifacts

- Global leading supplier of post-trading services
- 2500 customers in 110 countries
- Compliance with financial regulations
- Loosely follows the Rupp template

Industrial Context and Artifacts

- Global leading supplier of post-trading services
- 2500 customers in 110 countries
- Compliance with financial regulations
- Loosely follows the Rupp template
- 13 Software Requirements Specifications (SRS)

Industrial Context and Artifacts

- Global leading supplier of post-trading services
- 2500 customers in 110 countries
- Compliance with financial regulations
- Loosely follows the Rupp template
- 13 Software Requirements Specifications (SRS)
- 2725 requirements

Industrial Context and Artifacts

- Global leading supplier of post-trading services
- 2500 customers in 110 countries
- Compliance with financial regulations
- Loosely follows the Rupp template
- 13 Software Requirements Specifications (SRS)
- 2725 requirements
- Written by different business analysts

Industrial Context and Artifacts

- Global leading supplier of post-trading services
- 2500 customers in 110 countries
- Compliance with financial regulations
- Loosely follows the Rupp template
- 13 Software Requirements Specifications (SRS)
- 2725 requirements
- Written by different business analysts
- Relatively low quality, not used for validation

Industrial Context and Artifacts

- Global leading supplier of post-trading services
- 2500 customers in 110 countries
- Compliance with financial regulations
- Loosely follows the Rupp template
- 13 Software Requirements Specifications (SRS)
- 2725 requirements
- Written by different business analysts
- Relatively low quality, not used for validation

Controlled Natural Language: Rimay

Controlled Natural Language: Rimay

- Strike a balance between the usability of NL and the rigor of formal languages

Controlled Natural Language: Rimay

- Strike a balance between the usability of NL and the rigor of formal languages
- More specialized concepts and constructs than conventional templates

Controlled Natural Language: Rimay

- Strike a balance between the usability of NL and the rigor of formal languages
- More specialized concepts and constructs than conventional templates
- Qualitative analysis

Controlled Natural Language: Rimay

- Strike a balance between the usability of NL and the rigor of formal languages
- More specialized concepts and constructs than conventional templates
- Qualitative analysis
- Evaluation

Controlled Natural Language: Rimay

- **Strike a balance between the usability of NL and the rigor of formal languages**
- **More specialized concepts and constructs than conventional templates**
- **Qualitative analysis**
- **Evaluation**

Veziaga et al., “On systematically building a controlled natural language for functional requirements.”, Empir. Softw. Eng. 26(4): 79 (2021)

REQUIREMENT: SCOPE? CONDITION_STRUCTURE?
ARTICLE? ACTOR MODAL_VERB not? SYSTEM_RESPONSE.

SCOPE: For MODIFIER? TEXT (and MODIFIER? TEXT),

SYSTEM RESPONSE: VALIDATE | ... 47 action phrases

Scope | Actor | Modal Verb

```
For each "line of the File", System  must
check that Share_Class_Identifier.Value contains "line.ISIN”.
```

System Response

Controlled Natural Language: Rimay

- Strike a balance between the usability of NL and the rigor of formal languages
- More specialized concepts and constructs than conventional templates
- Qualitative analysis
- Evaluation

Veziaga et al., “On systematically building a controlled natural language for functional requirements.”, Empir. Softw. Eng. 26(4): 79 (2021)

REQUIREMENT: SCOPE? CONDITION_STRUCTURE?
ARTICLE? ACTOR MODAL_VERB not? SYSTEM_RESPONSE.

SCOPE: For MODIFIER? TEXT (and MODIFIER? TEXT),

SYSTEM RESPONSE: VALIDATE | ... 47 action phrases

Scope | Actor | Modal Verb

```
For each "line of the File", System  must
check that Share_Class_Identifier.Value contains "line.ISIN”.
```

System Response

Quality Attributes

- Quality attributes enforced by Rimay
  - **Completeness:** Presence of all the information required for the requirement to be complete
  - **Correctness:** Presence of correct information content in the correct order of appearance
  - **Clarity:** Usage of structures, phrases, and words that are free of ambiguity
  - **Atomic requirements:** A requirement with a single system function

Smells

- **10 smells:** "1. Non atomic", "2. Incomplete requirement", "3. Incorrect order requirement", "4. Coordination ambiguity", "5. Not requirement", **"6. Incomplete condition", "7. Incomplete system response"**, "8. Incomplete scope", "9. Passive voice", and "10. Not precise verb"

Condition (Missing Actor)

```
When creating a new participant,
System-A must open in edit mode the detail participant screen.
```

System Response

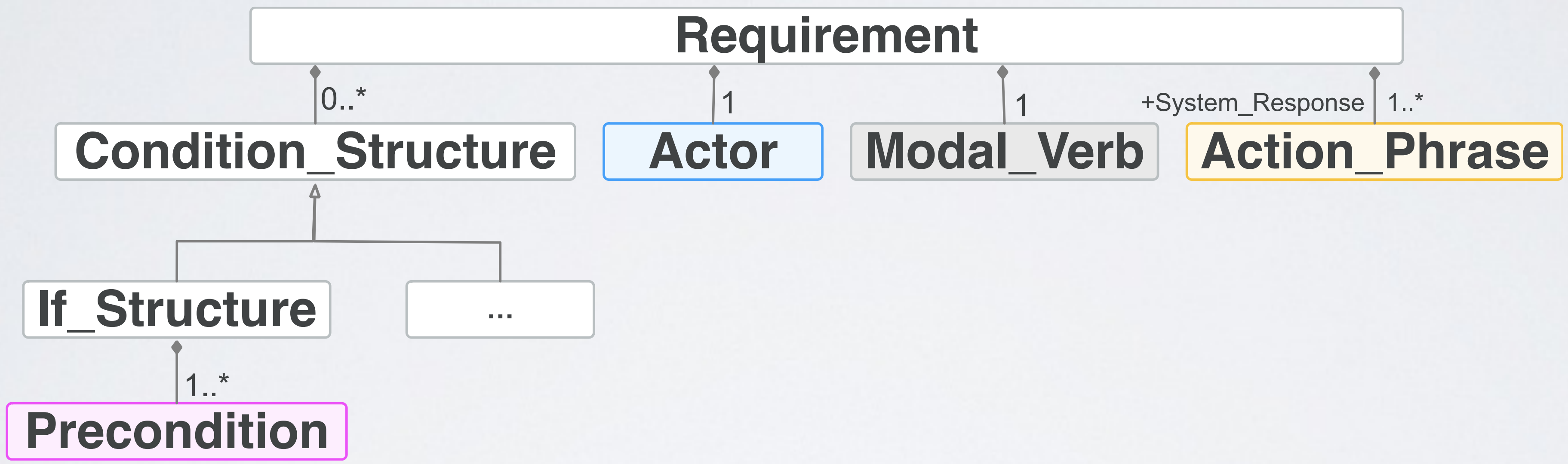
Condition

```
When a settlement request has reached the status "Settled"
then System-A must sed the settlement request to System-B.
```

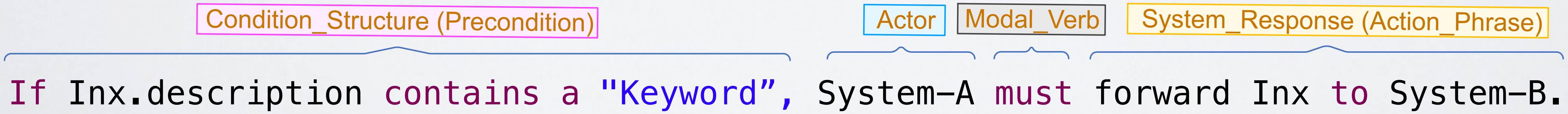
System Response (No verb)

Rimay Patterns

- Excerpt of the Rimay conceptual model:



- Example:



Rimay Patterns

Rimay Patterns

- **10 Rimay patterns:** "1. Scope and system response", "2. Scope, condition (precondition), and system response", "3. Scope, condition(Trigger), and system response", "4. Scope, condition (Time) and system response", "5. System response", "6. Condition(Precondition), and system response", "7. Condition(Trigger), and system response", "8. Condition (Time) and system response", "9. Scope, multiple conditions, and system response", "10. Multiple conditions, and system response"

Rimay Patterns

- **10 Rimay patterns:** "1. Scope and system response", "2. Scope, condition (precondition), and system response", "3. Scope, condition(Trigger), and system response", "4. Scope, condition (Time) and system response", "5. System response", "6. Condition(Precondition), and system response", "7. Condition(Trigger), and system response", "8. Condition (Time) and system response", "9. Scope, multiple conditions, and system response", "10. Multiple conditions, and system response"

- Pattern Name: System response

Rimay Patterns

- **10 Rimay patterns:** “1. Scope and system response”, “2. Scope, condition (precondition), and system response”, “3. Scope, condition(Trigger), and system response”, “4. Scope, condition (Time) and system response”, **“5. System response**”, “6. Condition(Precondition), and system response”, “7. Condition(Trigger), and system response”, “8. Condition (Time) and system response”, “9. Scope, multiple conditions, and system response”, “10. Multiple conditions, and system response”

- Pattern Name: System response

- Rimay Pattern: `The? Actor must <Action> (every "Text")?`

Rimay Patterns

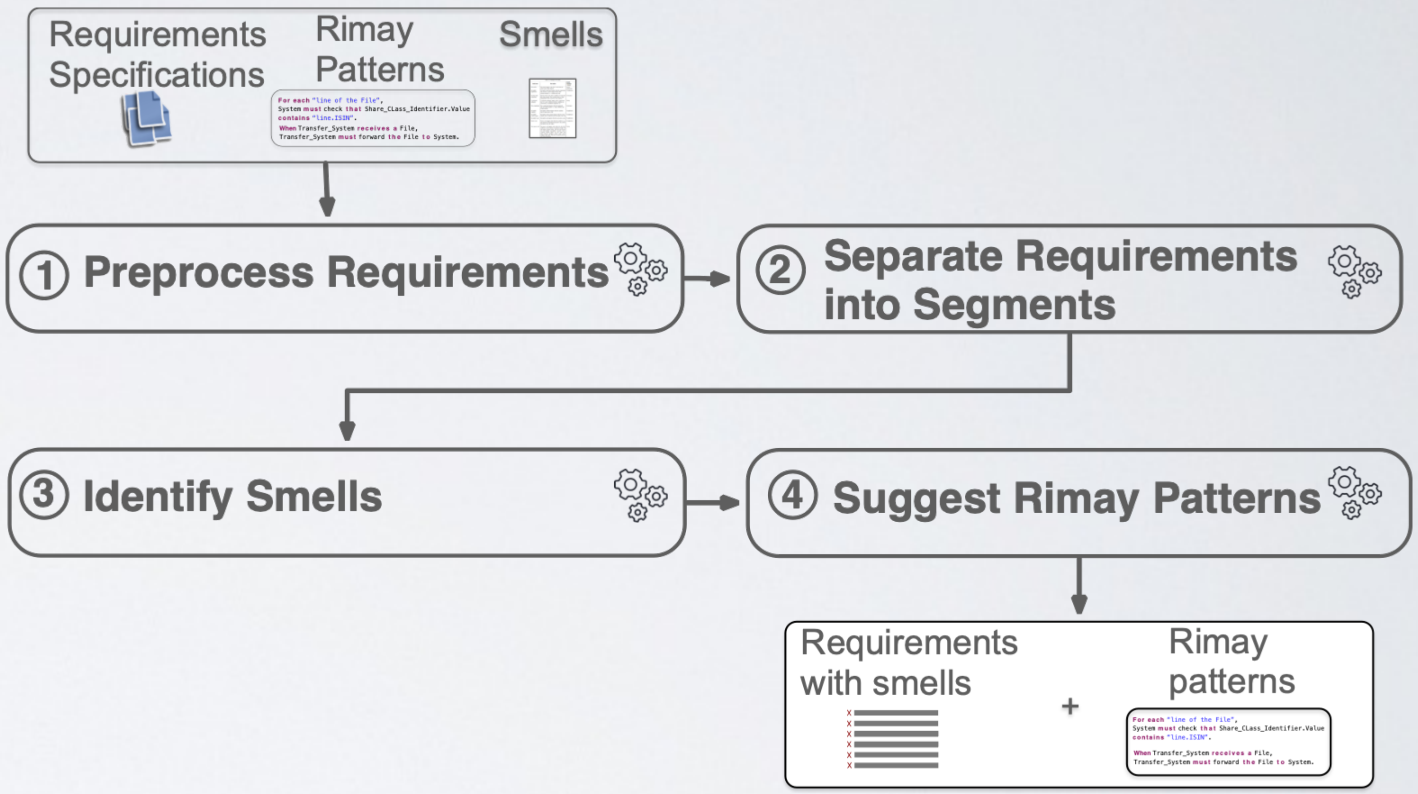
- **10 Rimay patterns:** “1. Scope and system response”, “2. Scope, condition (precondition), and system response”, “3. Scope, condition(Trigger), and system response”, “4. Scope, condition (Time) and system response”, **“5. System response”**, “6. Condition(Precondition), and system response”, “7. Condition(Trigger), and system response”, “8. Condition (Time) and system response”, “9. Scope, multiple conditions, and system response”, “10. Multiple conditions, and system response”

- Pattern Name: System response

- Rimay Pattern: `The? Actor must <Action> (every "Text")?`

- Example: `The System-A must link "allegement message-A" to the "outgoing message-123"`

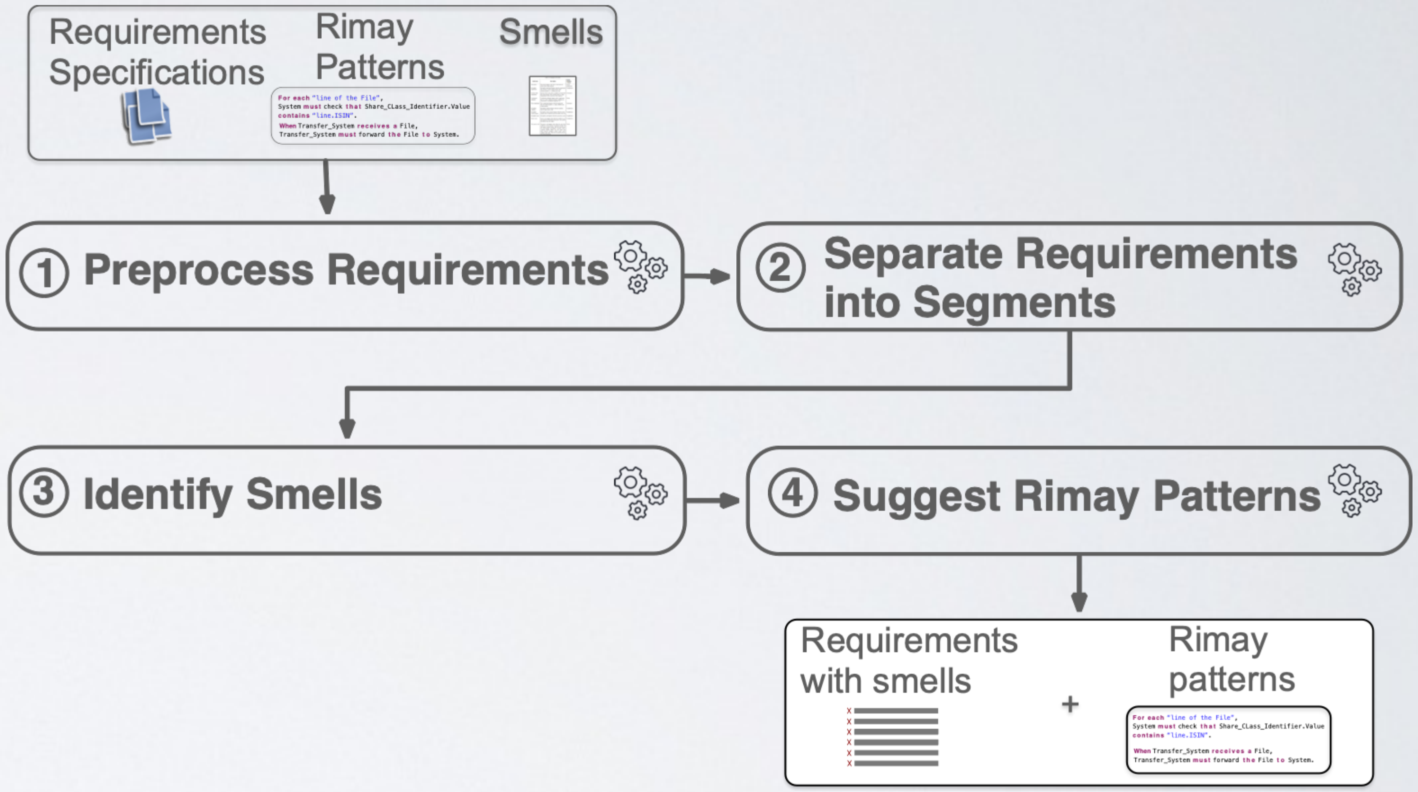
QA Approach: Paska



Veziaga et al., “Automated Smell Detection and Recommendation in Natural Language Requirements.”, IEEE TSE (2024)

QA Approach: Paska

1. **Apply preprocessing steps to requirements**

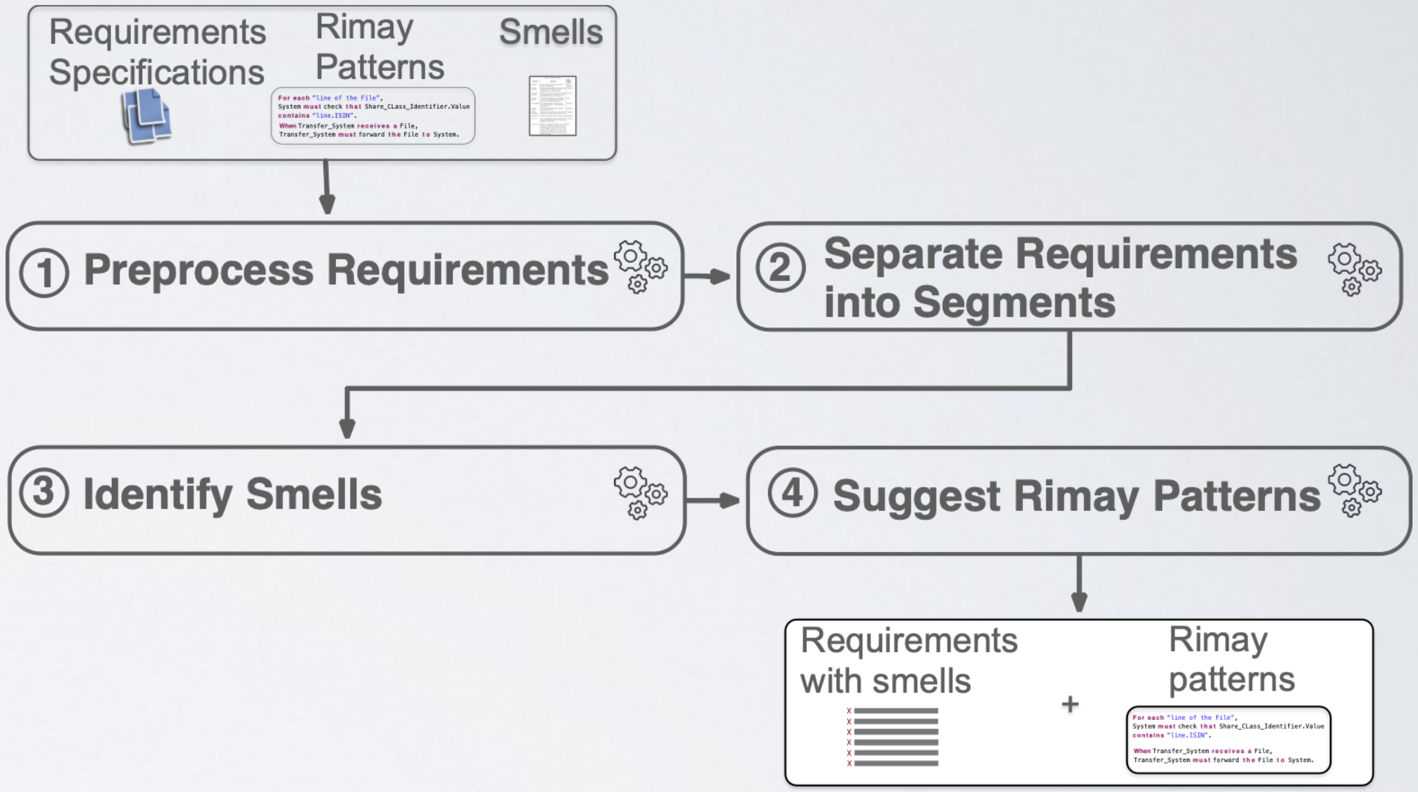


Veziaga et al., "Automated Smell Detection and Recommendation in Natural Language Requirements.", IEEE TSE (2024)

QA Approach: Paska

1. Apply preprocessing steps to requirements

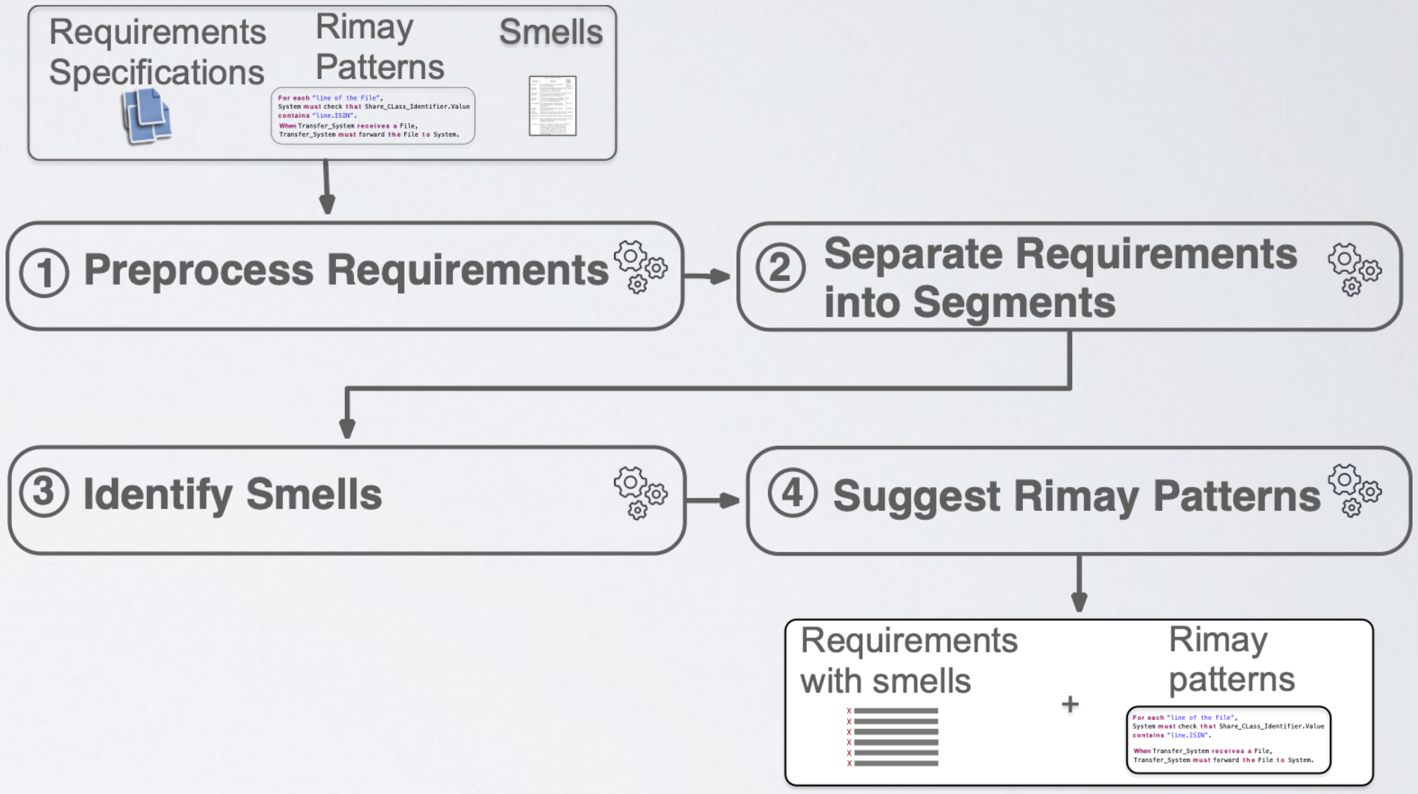
2. Divide requirements into scope, condition and system response



Veziaga et al., "Automated Smell Detection and Recommendation in Natural Language Requirements.", IEEE TSE (2024)

QA Approach: Paska

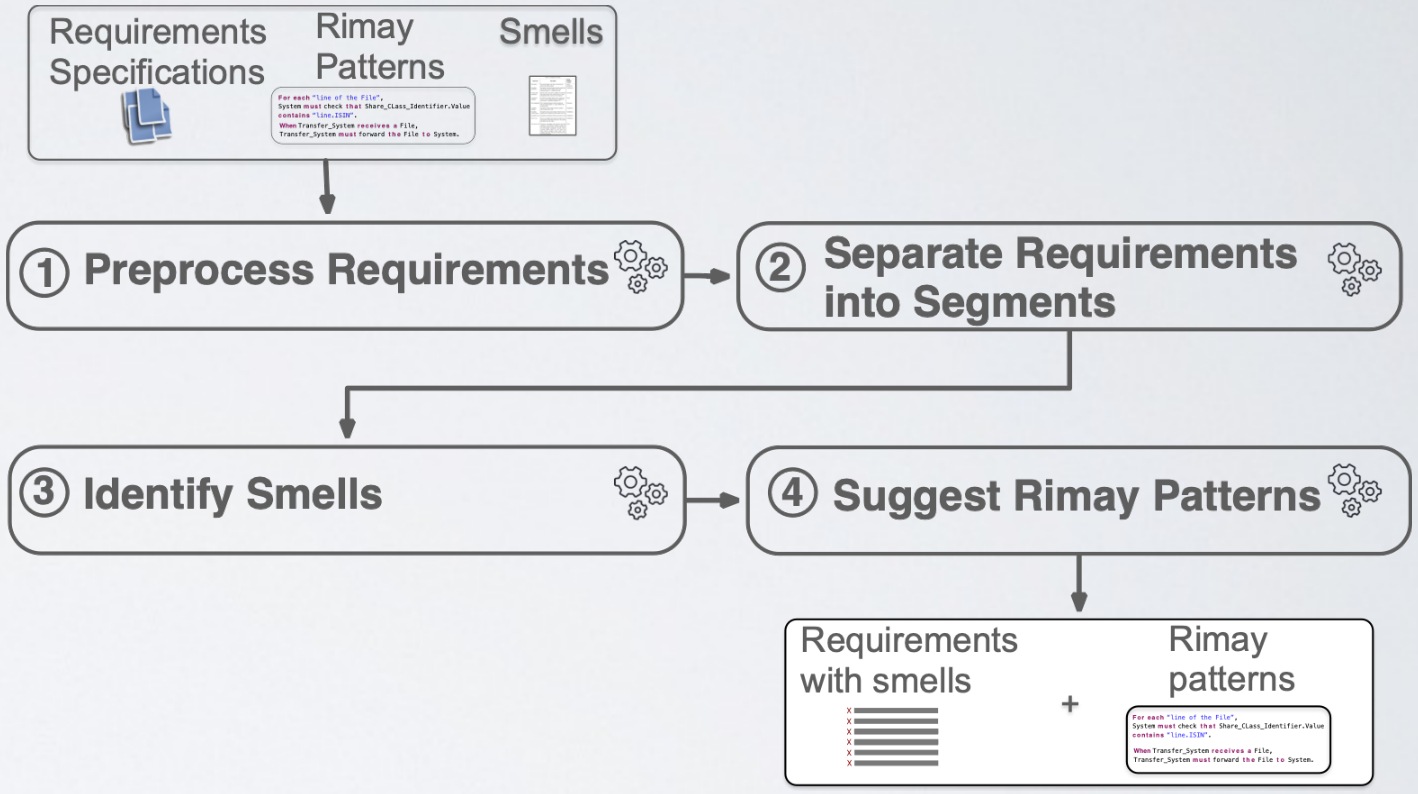
1. Apply preprocessing steps to requirements
2. Divide requirements into scope, condition and system response
3. Identify smells present in requirements



Veziaga et al., “Automated Smell Detection and Recommendation in Natural Language Requirements.”, IEEE TSE (2024)

QA Approach: Paska

1. Apply preprocessing steps to requirements
2. Divide requirements into scope, condition and system response
3. Identify smells present in requirements
4. Suggest a Rimay pattern to fix detected smells



Veziaga et al., “Automated Smell Detection and Recommendation in Natural Language Requirements.”, IEEE TSE (2024)

Paska: Evaluation

Paska: Evaluation

- Evaluation on unseen SRSs

Paska: Evaluation

- Evaluation on unseen SRSs
- Many instances of smells

Paska: Evaluation

- Evaluation on unseen SRSs
- Many instances of smells
- Overall precision and recall of 89% in detecting smells

Paska: Evaluation

- Evaluation on unseen SRSs
- Many instances of smells
- Overall precision and recall of 89% in detecting smells
- Overall precision of 96% and recall of 94% in suggesting Rimay patterns

Paska: Evaluation

- Evaluation on unseen SRSs
- Many instances of smells
- Overall precision and recall of 89% in detecting smells
- Overall precision of 96% and recall of 94% in suggesting Rimay patterns
- Based on the pattern it is straightforward to fix a requirement

Paska: Evaluation

- Evaluation on unseen SRSs
- Many instances of smells
- Overall precision and recall of 89% in detecting smells
- Overall precision of 96% and recall of 94% in suggesting Rimay patterns
- Based on the pattern it is straightforward to fix a requirement
- Promising tool for automated QA of natural language requirements for information systems

Paska: Evaluation

- Evaluation on unseen SRSs
- Many instances of smells
- Overall precision and recall of 89% in detecting smells
- Overall precision of 96% and recall of 94% in suggesting Rimay patterns
- Based on the pattern it is straightforward to fix a requirement
- Promising tool for automated QA of natural language requirements for information systems

Acceptance Testing Driven by Use Case Specifications

Acceptance Testing

Acceptance Testing

- Acceptance testing from requirements

Acceptance Testing

- Acceptance testing from requirements
- Systematically Identifying and designing test scenarios

Acceptance Testing

- Acceptance testing from requirements
- Systematically Identifying and designing test scenarios
- Overcome biases and blind spots

Acceptance Testing

- Acceptance testing from requirements
- Systematically Identifying and designing test scenarios
- Overcome biases and blind spots
- Traceability requirements – system tests often required

Test Cases from Use Cases

Test Cases from Use Cases

- Test case descriptions from use case descriptions

Test Cases from Use Cases

- Test case descriptions from use case descriptions
- Use case specifications are common and operational representations of requirements

Test Cases from Use Cases

- Test case descriptions from use case descriptions
- Use case specifications are common and operational representations of requirements
- Manual and laborious task

Test Cases from Use Cases

- Test case descriptions from use case descriptions
- Use case specifications are common and operational representations of requirements
- Manual and laborious task
- Context: Automotive software.

Test Cases from Use Cases

- Test case descriptions from use case descriptions
- Use case specifications are common and operational representations of requirements
- Manual and laborious task
- Context: Automotive software.

Test Cases from Use Cases

- Test case descriptions from use case descriptions
- Use case specifications are common and operational representations of requirements
- Manual and laborious task
- Context: Automotive software.
- Use cases were a basis for contractual agreements

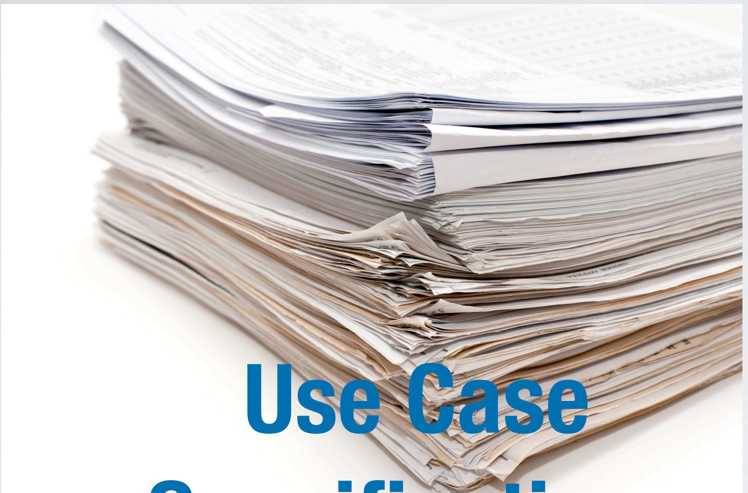
Test Cases from Use Cases

- Test case descriptions from use case descriptions
- Use case specifications are common and operational representations of requirements
- Manual and laborious task
- Context: Automotive software.
- Use cases were a basis for contractual agreements
- Automation based on NLP: UMTG

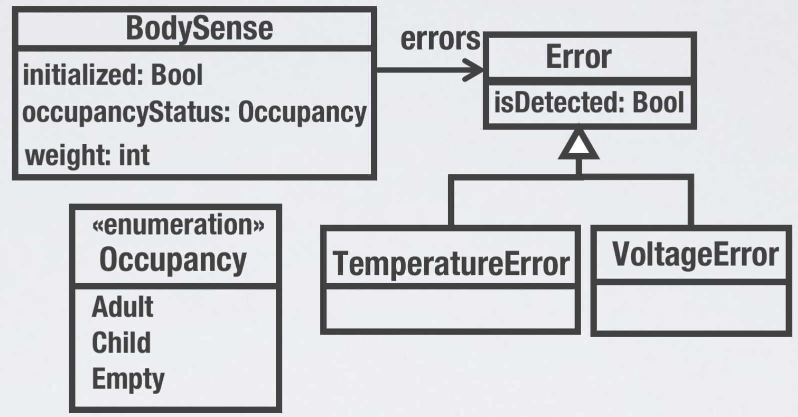
Test Cases from Use Cases

- Test case descriptions from use case descriptions
- Use case specifications are common and operational representations of requirements
- Manual and laborious task
- Context: Automotive software.
- Use cases were a basis for contractual agreements
- Automation based on NLP: UMTG

UMTG



Use Case Specifications (RUCM)



Domain Model

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

Constraints capturing the meaning of conditions

UMTG

https://sntsvv.github.io/UMTG/

Wang et al., "Automatic Generation of Acceptance Test Cases From Use Case Specifications: An NLP-Based Approach", IEEE TSE, 2020

Executable Test Cases

Restricted Use Case Specifications

- Restricted Use Case Modeling (RUCM)
- Experiments shows it yields **better use cases**
- Compliance is **tool-supported** (NLP)
- More **analyzable** with NLP

Yue et al., ACM TOSEM 2013

RUCM Specifications Example

Precondition: The system has been initialized

Basic Flow

1. The SeatSensor **SENDS** the weight **TO** the system.

INPUT STEP

2. **INCLUDE USE CASE** Self Diagnosis.

INCLUDE STEP

3. The system **VALIDATES THAT** no error has been detected.

CONDITIONAL STEP

4. The system **VALIDATES THAT** the weight is above 20 Kg.
5. The system sets the occupancy status to adult.

INTERNAL STEP

6. The system **SENDS** the occupancy status **TO** AirbagControlUnit.

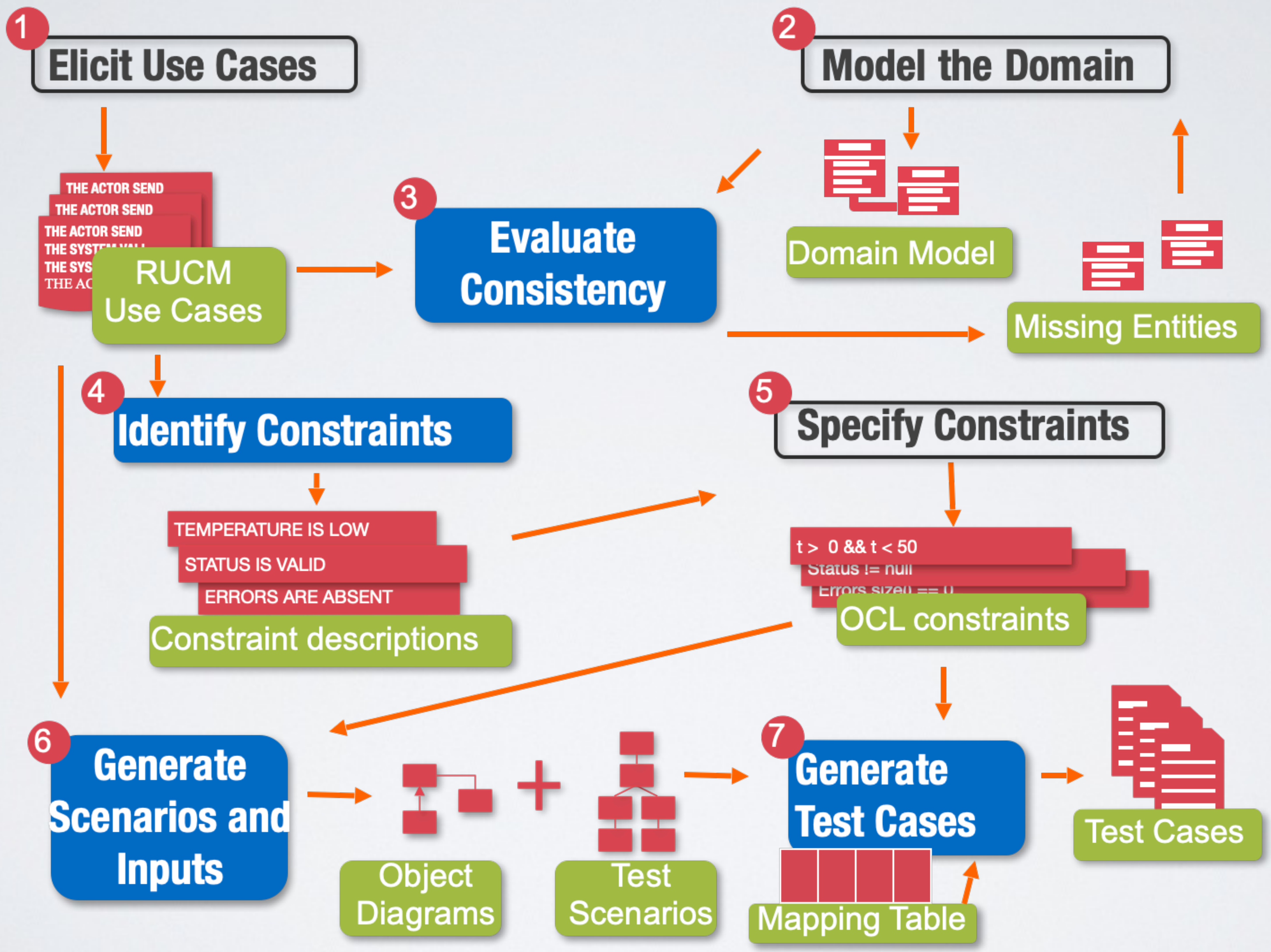
OUTPUT STEP

Alternative Flow

RFS 4.

1. **IF** the weight is above 1 Kg **THEN**
2. The system sets the occupancy status to child.
3. …
4. **RESUME STEP 6.**

UMTG Steps



1
Elicit Use Cases
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

UMTG: Empirical Evaluation

UMTG: Empirical Evaluation

- It is hard for engineers to capture all the possible scenarios involving error conditions. UMTG covers them.

UMTG: Empirical Evaluation

- It is hard for engineers to capture all the possible scenarios involving error conditions. UMTG covers them.

- 5 to 10 minutes to write each constraints (but we also developed a tool to generate them)

UMTG: Empirical Evaluation

- It is hard for engineers to capture all the possible scenarios involving error conditions. UMTG covers them.
- 5 to 10 minutes to write each constraints (but we also developed a tool to generate them)
- 10 secs/test case using a constraint solver based on Alloy

UMTG: Empirical Evaluation

- It is hard for engineers to capture all the possible scenarios involving error conditions. UMTG covers them.
- 5 to 10 minutes to write each constraints (but we also developed a tool to generate them)
- 10 secs/test case using a constraint solver based on Alloy
- Less than 10 minutes in total to generate all system test cases (54)

UMTG: Empirical Evaluation

- It is hard for engineers to capture all the possible scenarios involving error conditions. UMTG covers them.
- 5 to 10 minutes to write each constraints (but we also developed a tool to generate them)
- 10 secs/test case using a constraint solver based on Alloy
- Less than 10 minutes in total to generate all system test cases (54)

Inter-Requirements Change Impact Analysis

Traceability

Traceability

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

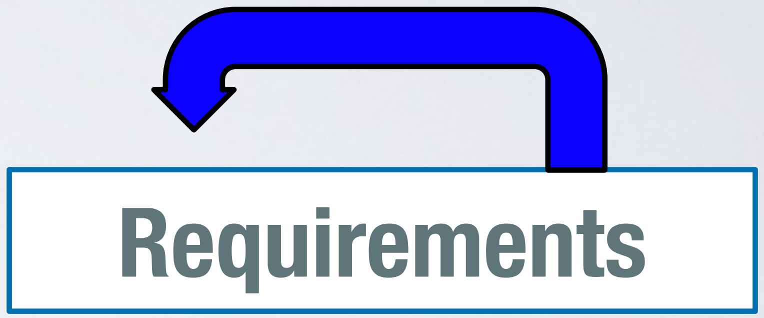
Traceability

- The ability to follow the life of software artifacts, in both a backward and forward direction, e.g., requirements, design decisions, test cases.
- Requirements traceability: Trace a requirement from its emergence to its fulfillment, e.g., acceptance test cases.

Traceability

- The ability to follow the life of software artifacts, in both a backward and forward direction, e.g., requirements, design decisions, test cases.
- Requirements traceability: Trace a requirement from its emergence to its fulfillment, e.g., acceptance test cases.
- Motivations:
  - Understand rationale
  - Certification, auditing, compliance with standards
  - Assess impact of change

Traceability between Requirements



- Natural language
- Sometimes structured (template)
- Hundreds of traces
- Domain terminology, concepts, and their relationships are key to discovering traces among requirements
- Rely on syntactic and semantic similarity measures

Requirements-Requirements

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

Requirements

SES
your satellite company

Case-A

160 Requirements
9 change scenarios

novay

Case-B

72 Requirements
5 change scenarios

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 Example

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

Challenge#1 - Capture Changes Precisely

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

Challenge#2 - Capture Change Rationale

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

Challenge#2 - Change Rationale

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

Rationales:

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

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

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

user help desk, Deleted
user document repository, Added

Solution Characteristics

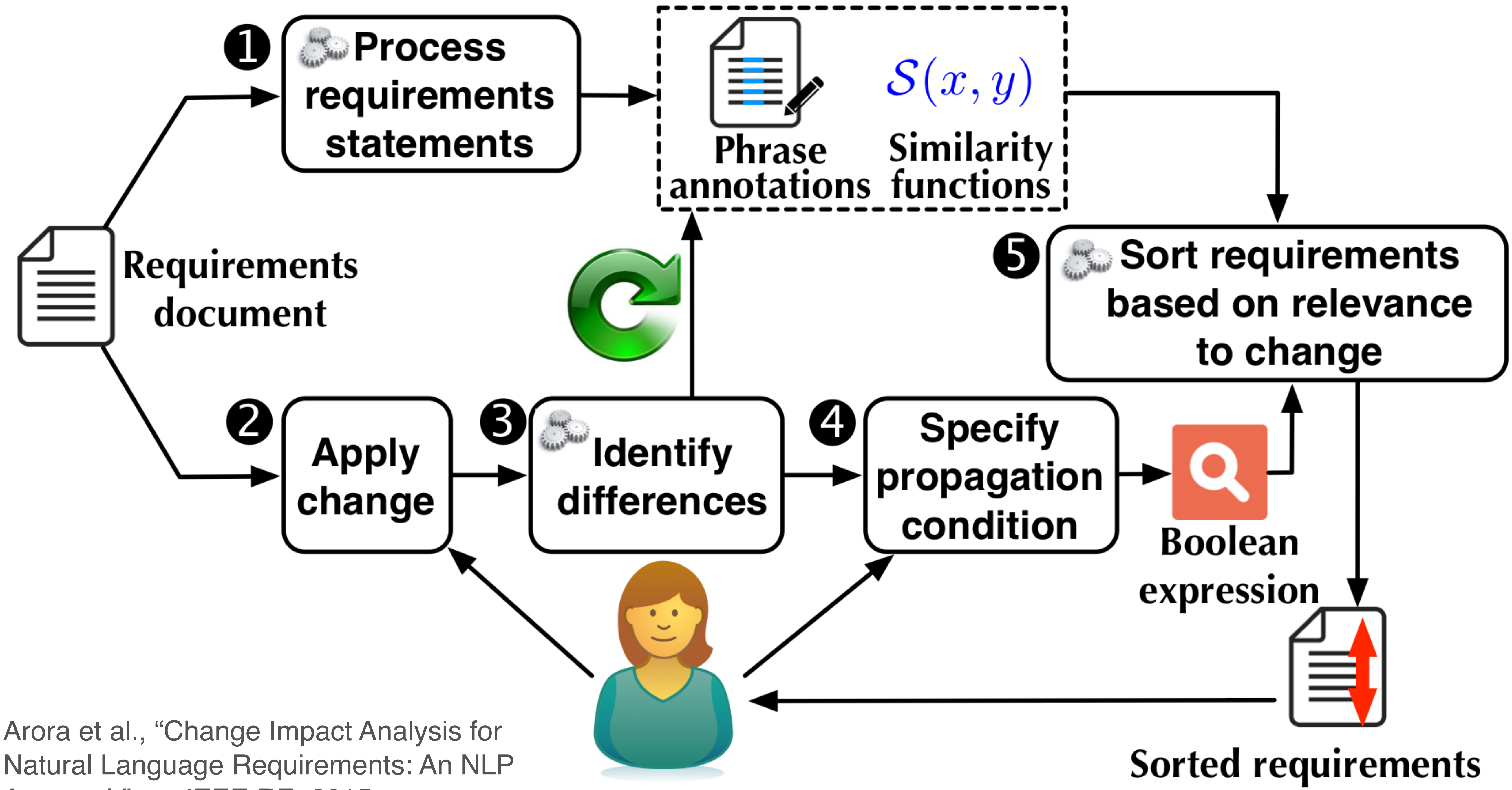
- **Accounts for the phrasal structure of requirements**

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

user help desk, Deleted
user document repository, Added

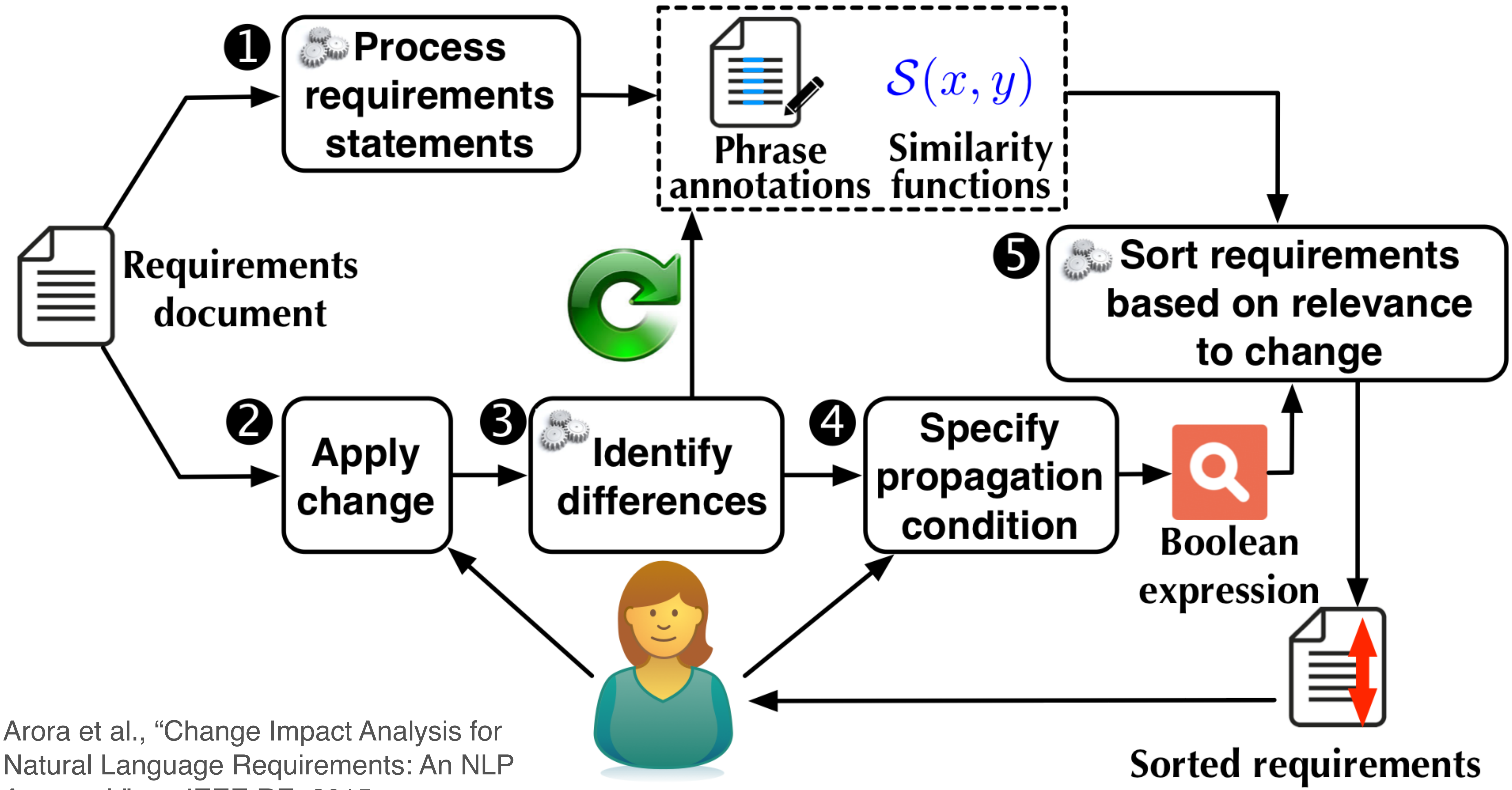
- **Account for semantically-related phrases that are not exact matches and close syntactic variations**

Approach



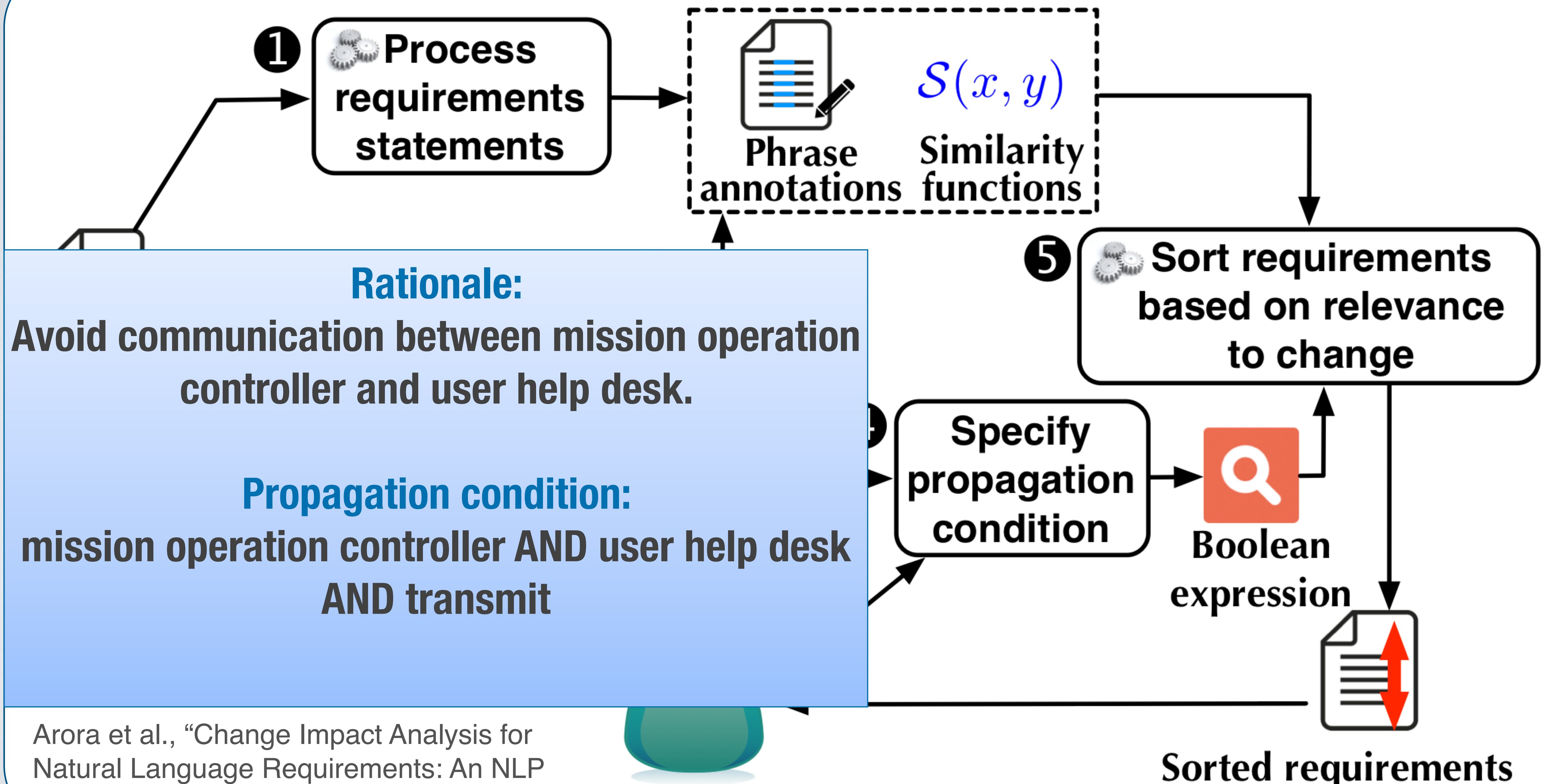
Arora et al., "Change Impact Analysis for Natural Language Requirements: An NLP Approach", IEEE RE, 2015

Approach



Arora et al., “Change Impact Analysis for Natural Language Requirements: An NLP Approach”, IEEE RE, 2015

Approach



Rationale:

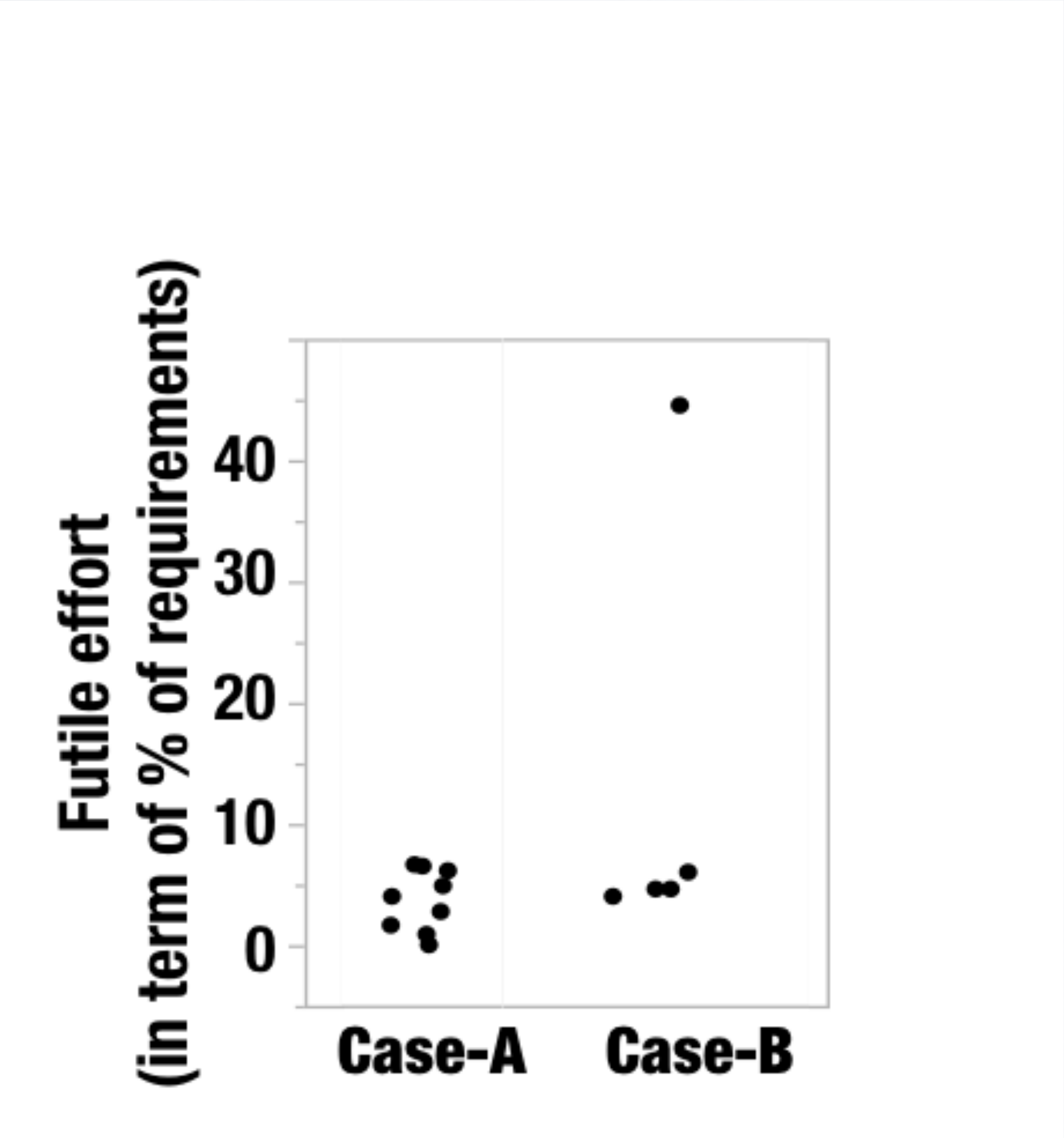
Avoid communication between mission operation controller and user help desk.

Propagation condition:

mission operation controller AND user help desk AND transmit

Arora et al., “Change Impact Analysis for Natural Language Requirements: An NLP Approach”, IEEE RE, 2015

How effective is our approach?



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

Requirements Change Impact Analysis on Design

Requirements-Design Traceability

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

DELPHI
Innovation for the Real World

Requirements → Archi. & Design

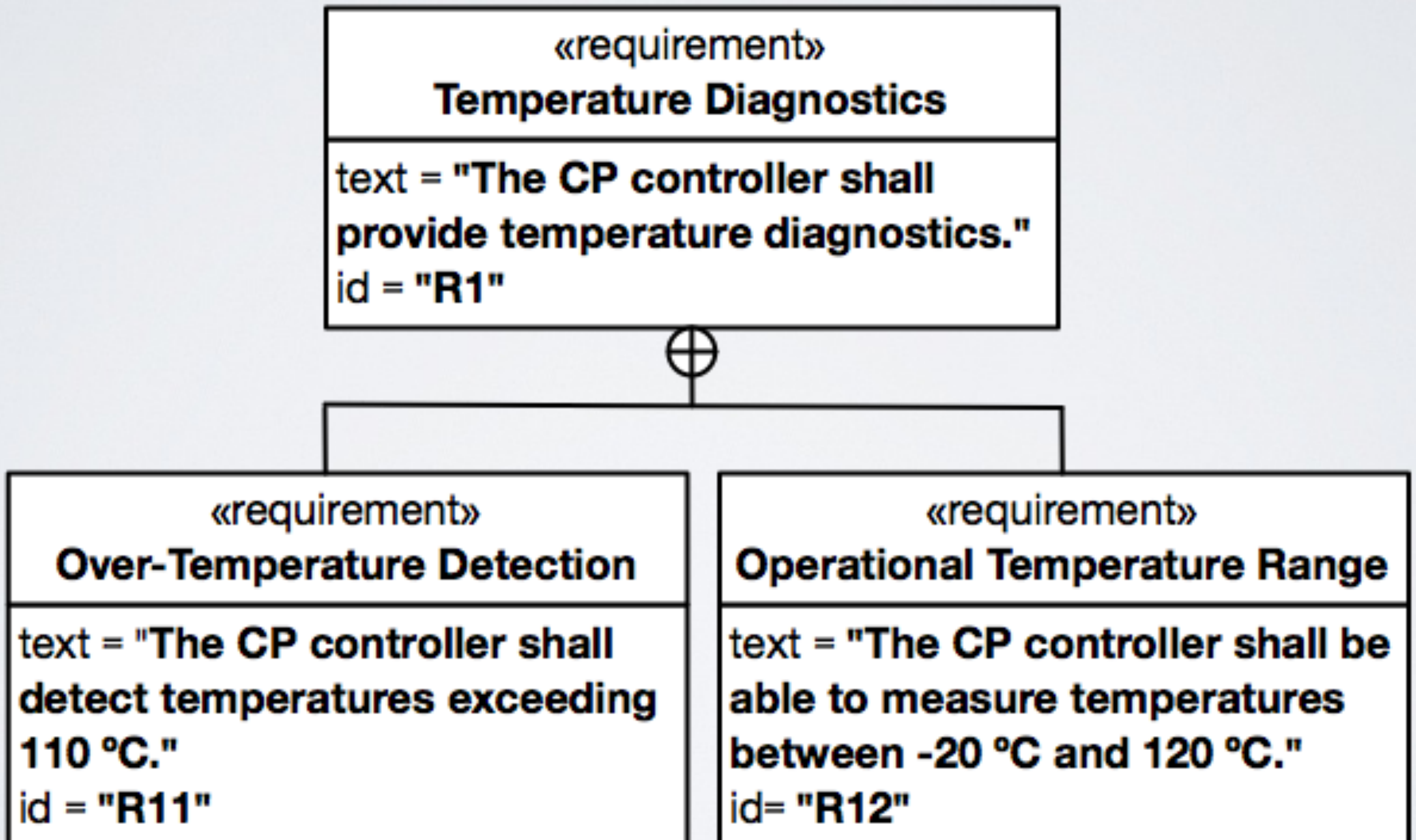
System Design Modeling

- **Systems Modeling Language (SysML)**
  - A subset of UML extended with systems engineering diagrams
  - A standard for systems engineering
  - Preliminary support for requirement analysis and built-in traceability mechanism

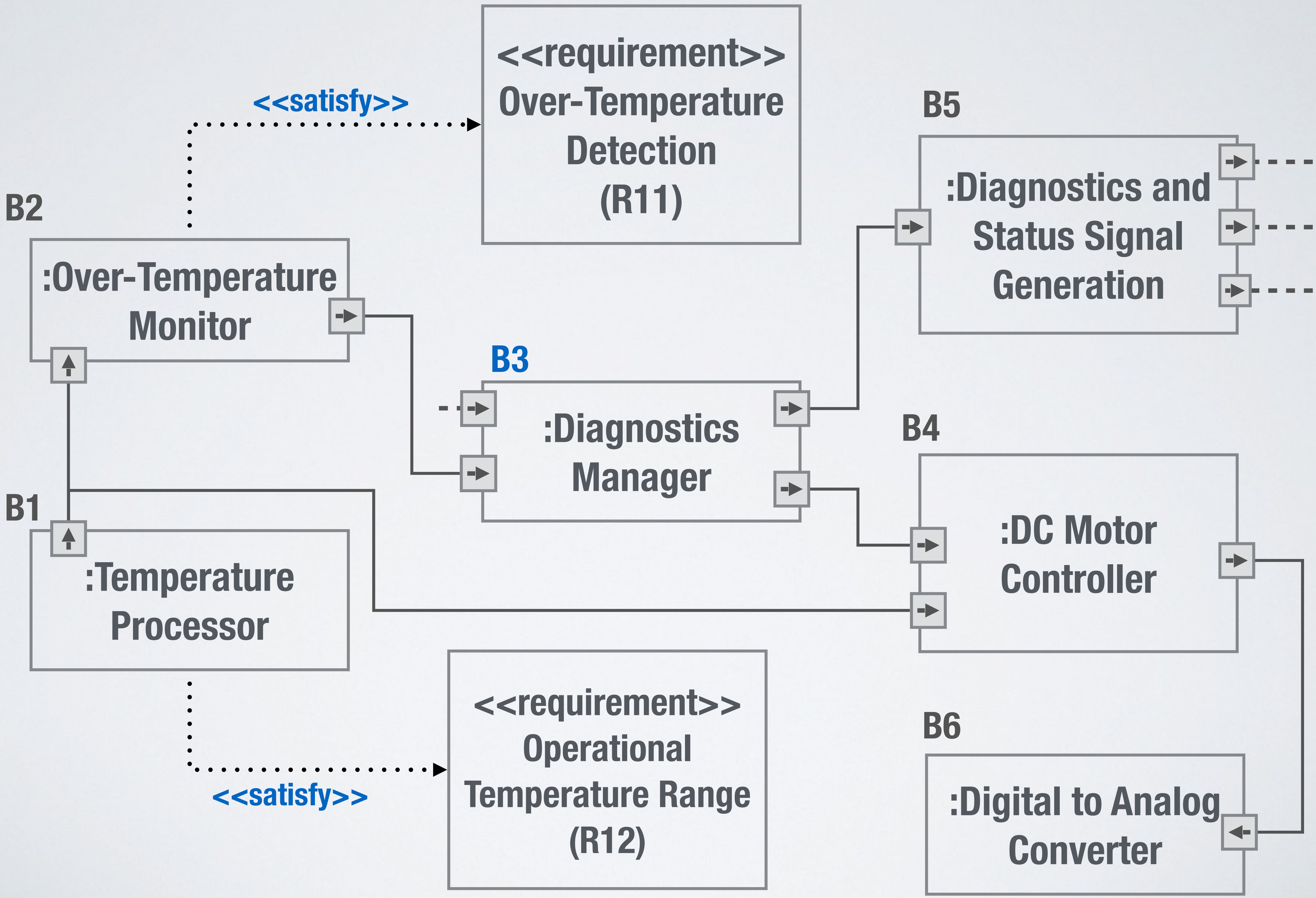
CIA Automation Goal

- Given a change in a requirement, our goal is to compute a set of (potentially) impacted design elements that includes
  - all the actually impacted elements (**high recall**)
  - very few non-impacted elements (**high precision**)

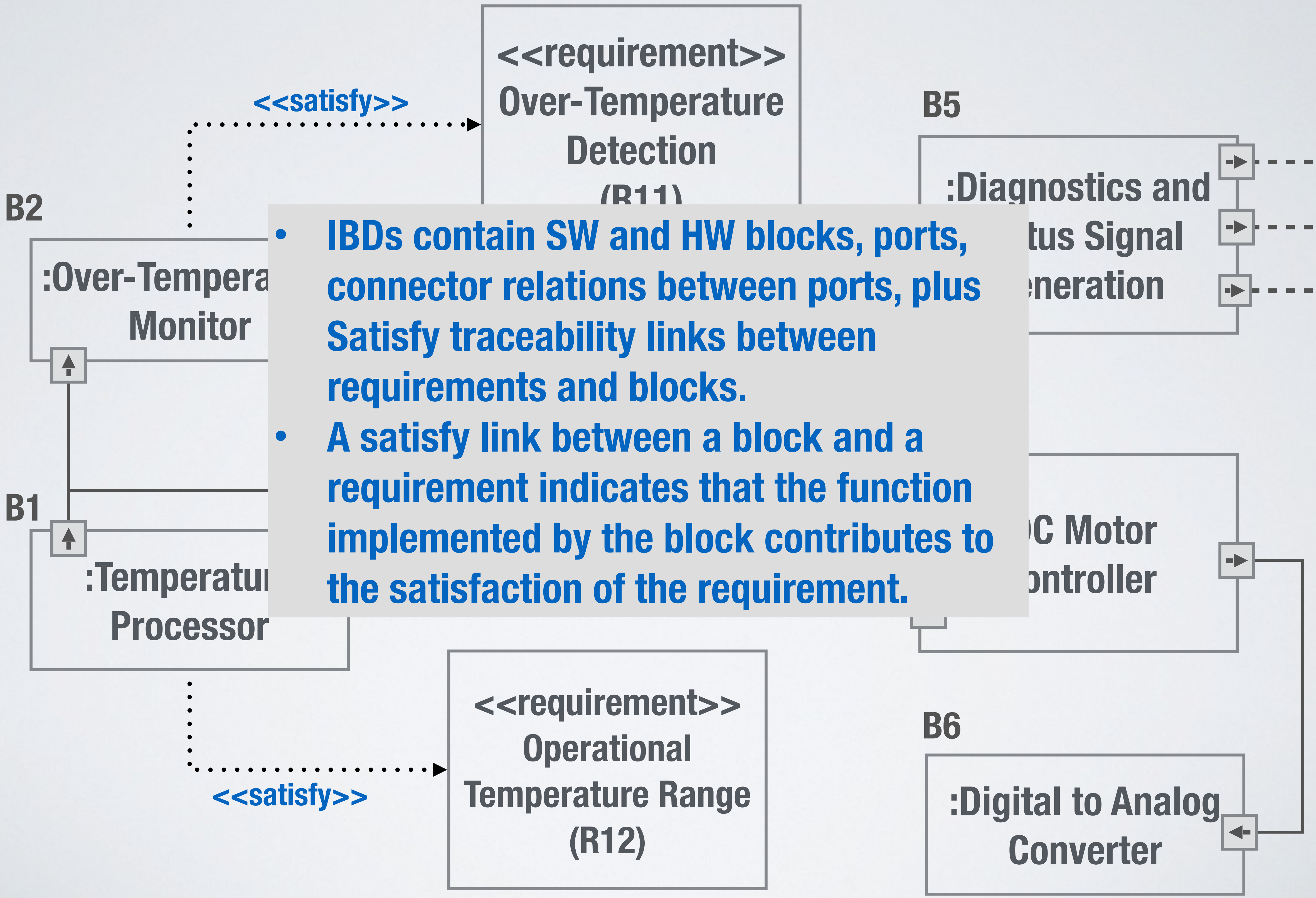
Requirements Diagram



Internal Block Diagrams (IBD)

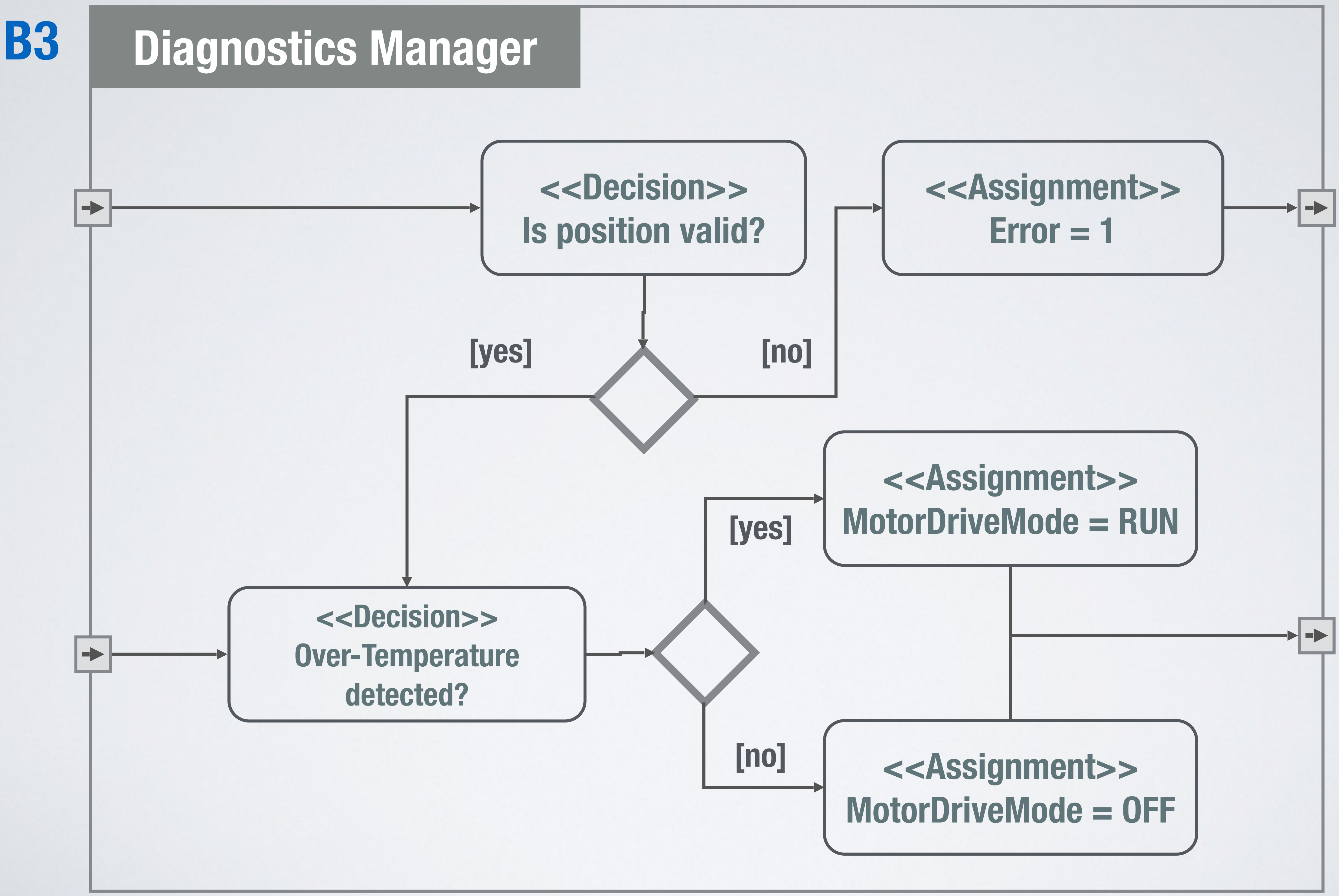


Internal Block Diagrams (IBD)

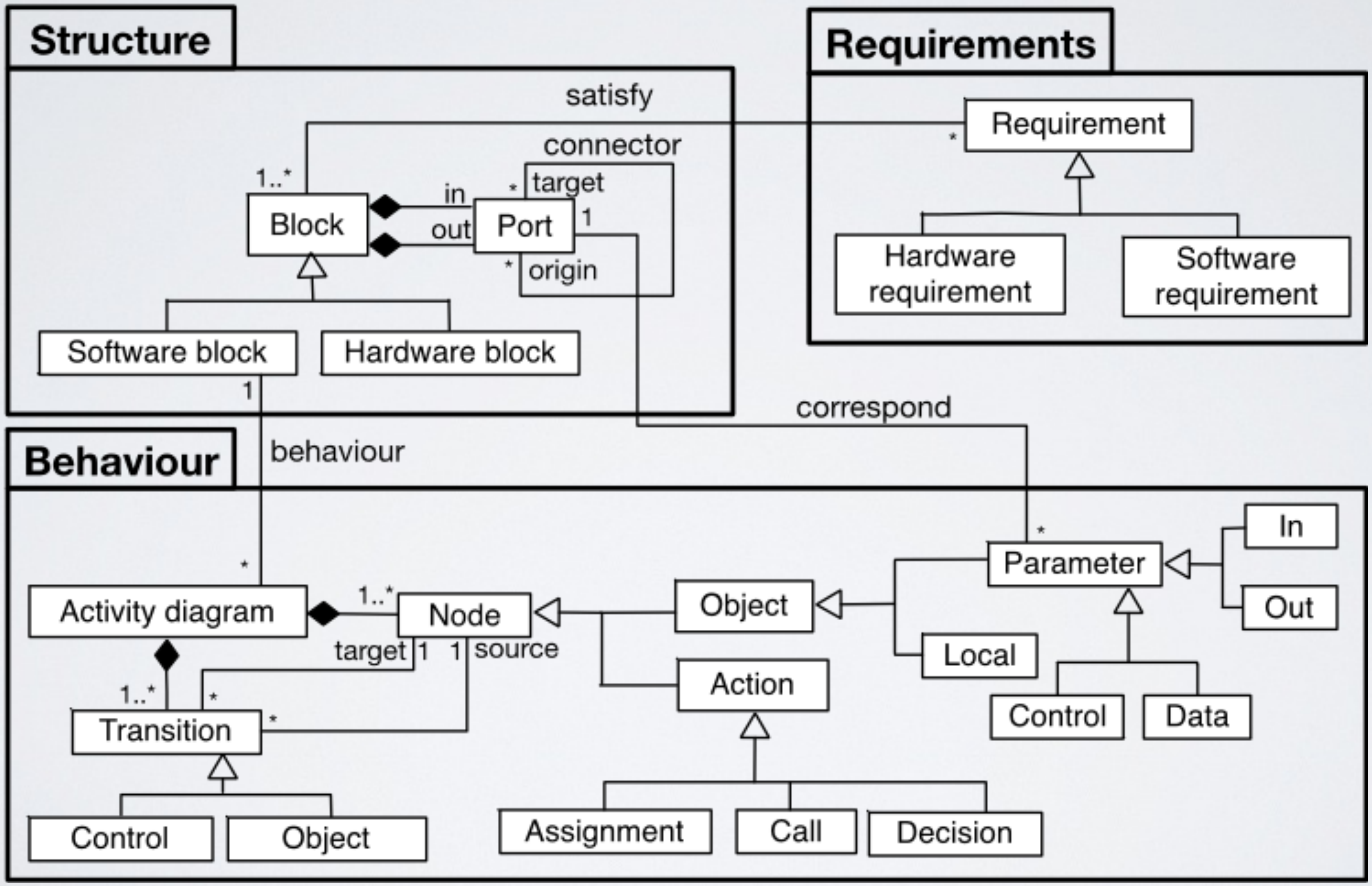


- **IBDs contain SW and HW blocks, ports, connector relations between ports, plus Satisfy traceability links between requirements and blocks.**
- **A satisfy link between a block and a requirement indicates that the function implemented by the block contributes to the satisfaction of the requirement.**

Activity Diagrams (AD)



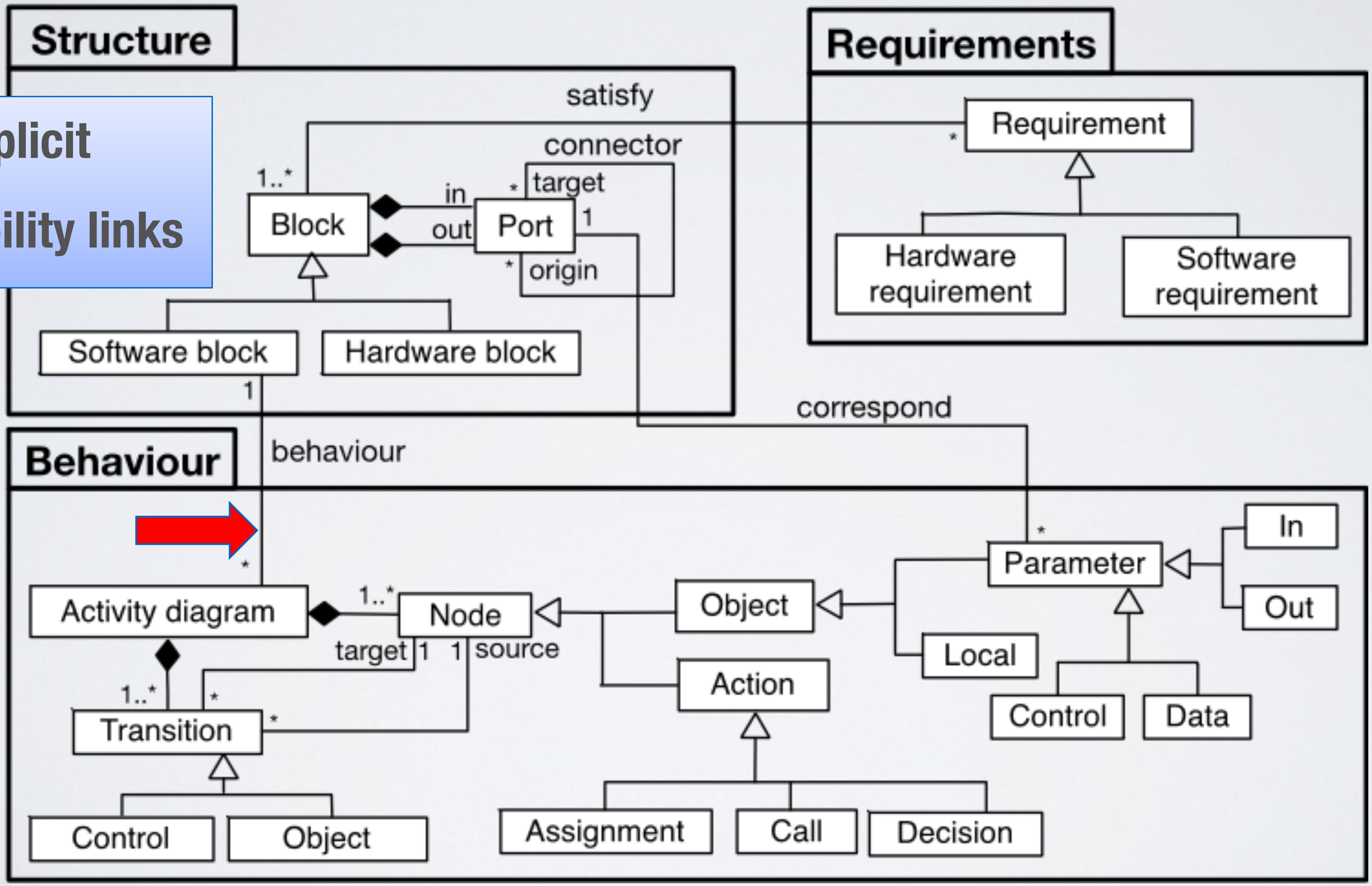
Traceability Information Model



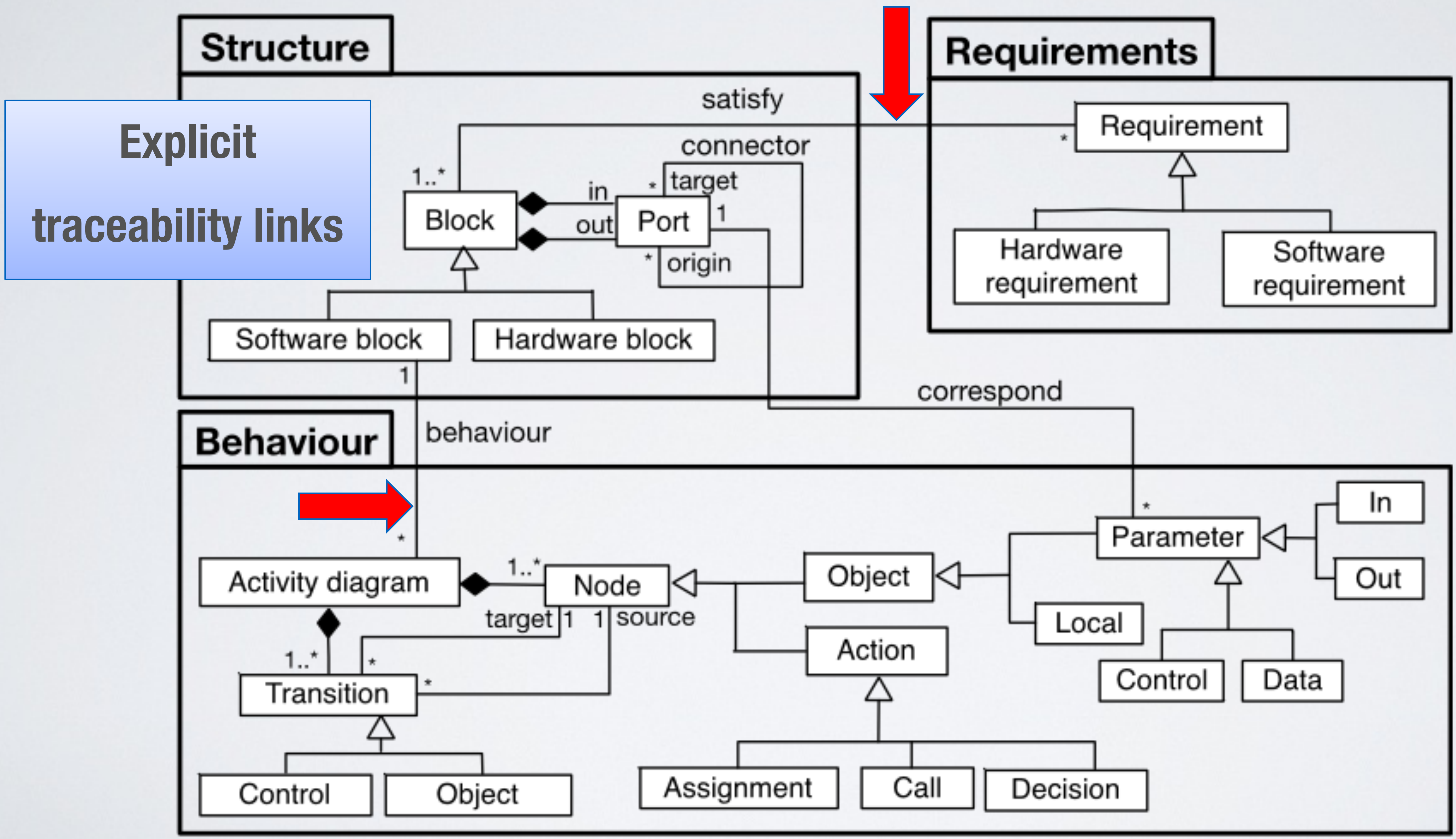
Traceability Information Model

Explicit

traceability links



Traceability Information Model



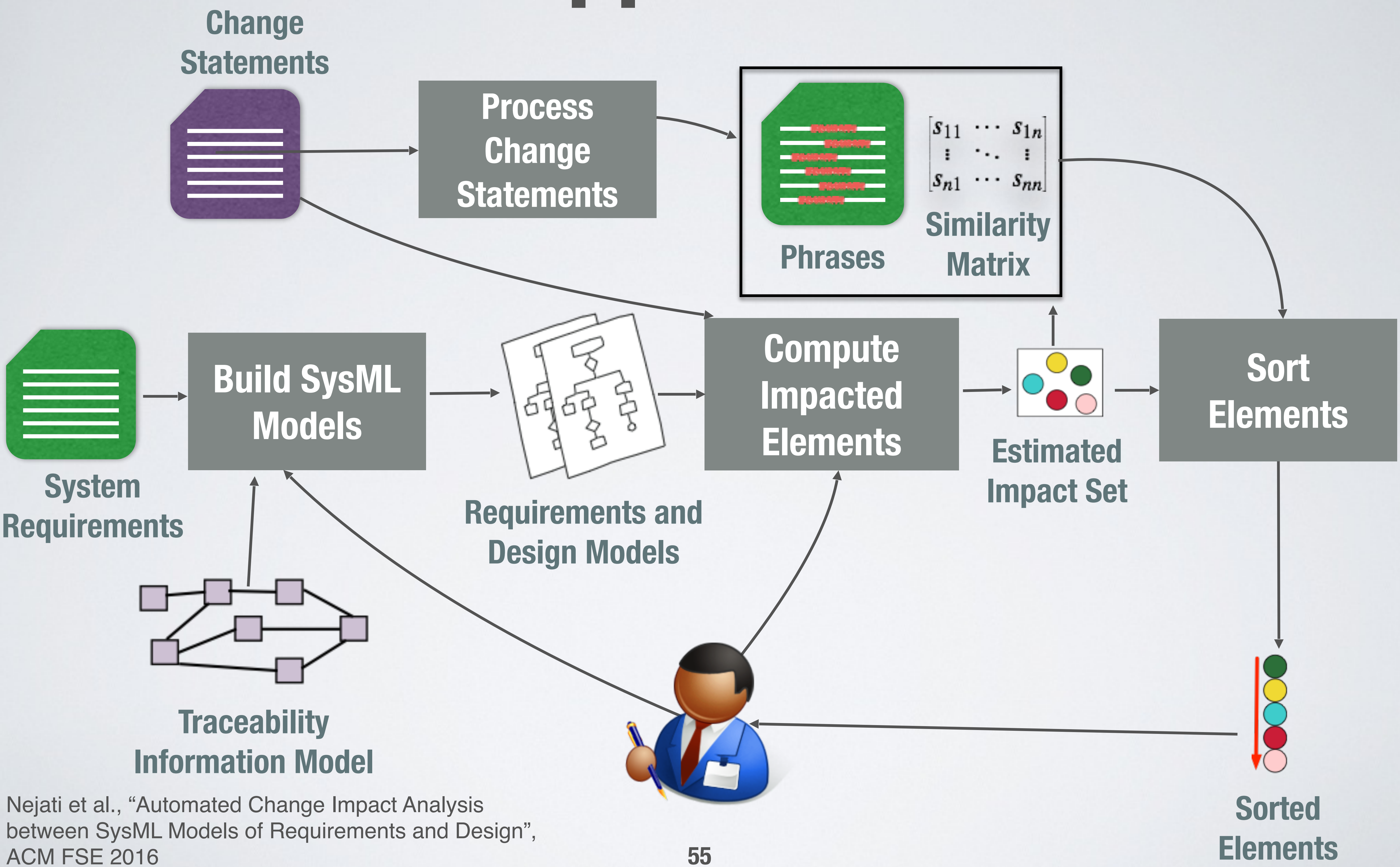
Our CIA Approach

Structural Analysis

Behavioral Analysis

Natural Language Processing Analysis

Approach



Nejati et al., “Automated Change Impact Analysis between SysML Models of Requirements and Design”, ACM FSE 2016

Case Study

DELPHI
Innovation for the Real World

Electronic Variable Cam Phaser (CP)

- Includes mechanical, electronic and software components
- Adjusts the timing of cam lobes with respect to that of the crank shaft in an engine, while the engine is running.
- CP is safety-critical and subject to the ISO 26262 standard.

Summary

Summary

- We provided an approach to automatically identify the impact of requirements changes on system design

Summary

- We provided an approach to automatically identify the impact of requirements changes on system design
- Our approach includes:
  - A SysML modeling methodology with acceptable traceability cost
  - An algorithm for impact computation that combines models' structure, behavior and textual information

Summary

- We provided an approach to automatically identify the impact of requirements changes on system design
- Our approach includes:
    - A SysML modeling methodology with acceptable traceability cost
    - An algorithm for impact computation that combines models’ structure, behavior and textual information
- Industrial case study: Our hybrid approach reduces the number of elements inspected from 370 to 18

Summary

- We provided an approach to automatically identify the impact of requirements changes on system design
- Our approach includes:
  - A SysML modeling methodology with acceptable traceability cost
  - An algorithm for impact computation that combines models' structure, behavior and textual information
- Industrial case study: Our hybrid approach reduces the number of elements inspected from 370 to 18
- Scalable approach: A few seconds to compute and rank estimated impacted elements

Conclusions

Main Takeaway Message

Main Takeaway Message

- In most cases, for many reasons, providing guidance to architects and developers does not seem to be a **sufficient motivation to document requirements** in a precise and complete form.

Main Takeaway Message

- In most cases, for many reasons, providing guidance to architects and developers does not seem to be a **sufficient motivation to document requirements** in a precise and complete form.

- We somehow need to **increase the RoI** of writing such requirements if we want practice to change.

Main Takeaway Message

- In most cases, for many reasons, providing guidance to architects and developers does not seem to be a **sufficient motivation to document requirements** in a precise and complete form.

- We somehow need to **increase the RoI** of writing such requirements if we want practice to change.

- **We need to develop practical technologies that increase RoI**

A Way Forward

A Way Forward

- **Many opportunities:** (1) Acceptance test automation, (2) Change impact analysis (e.g., for safety certification or regulatory compliance), (3) Automated QA, and more.

A Way Forward

- **Many opportunities:** (1) Acceptance test automation, (2) Change impact analysis (e.g., for safety certification or regulatory compliance), (3) Automated QA, and more.

- With a focus on **natural language** requirements.

A Way Forward

- **Many opportunities:** (1) Acceptance test automation, (2) Change impact analysis (e.g., for safety certification or regulatory compliance), (3) Automated QA, and more.
- With a focus on **natural language** requirements.
- With a **high degree of robustness** to unrestricted, flawed NL requirements

A Way Forward

- **Many opportunities:** (1) Acceptance test automation, (2) Change impact analysis (e.g., for safety certification or regulatory compliance), (3) Automated QA, and more.

- With a focus on **natural language** requirements.

- With a **high degree of robustness** to unrestricted, flawed NL requirements

- Keeping in mind **scalability**

Requirements for AI Systems

Requirements for AI Systems

- No source code from which to derive intent

Requirements for AI Systems

- No source code from which to derive intent
- Components for which precise functional requirements are difficult to express, e.g., pedestrian detection

Requirements for AI Systems

- No source code from which to derive intent
- Components for which precise functional requirements are difficult to express, e.g., pedestrian detection
- Safety, robustness, and security requirements are critical though --- their specification will increasingly be required by regulations

Requirements for AI Systems

- No source code from which to derive intent
- Components for which precise functional requirements are difficult to express, e.g., pedestrian detection
- Safety, robustness, and security requirements are critical though --- their specification will increasingly be required by regulations
- Operational Design domain must be specified (a form of requirement)

Requirements for AI Systems

- No source code from which to derive intent
- Components for which precise functional requirements are difficult to express, e.g., pedestrian detection
- Safety, robustness, and security requirements are critical though --- their specification will increasingly be required by regulations
- Operational Design domain must be specified (a form of requirement)
- There is an opportunity for impact here for the RE community!

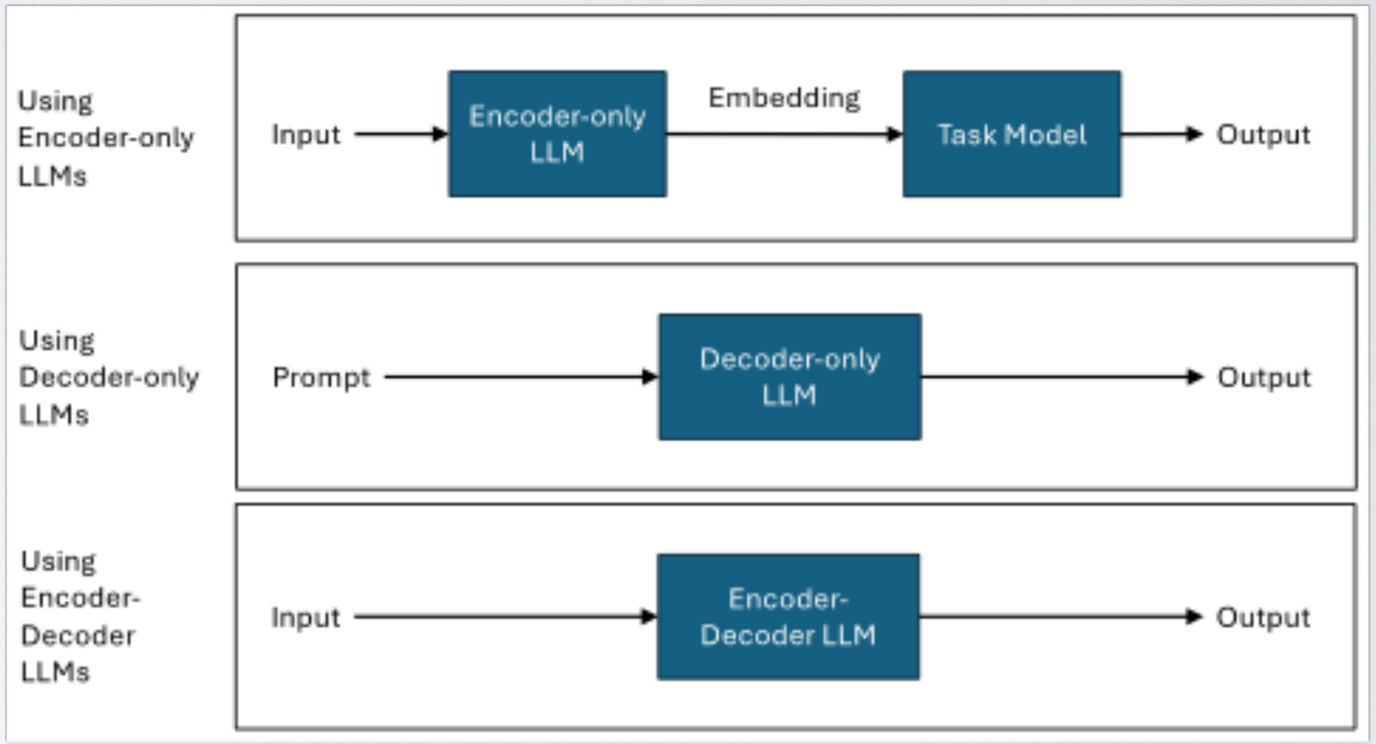
Operational Design Domain

- An operational Design Domain (ODD) refers to the specific conditions under which a system or technology, like an Autonomous Vehicle (AV), is designed to function safely and efficiently.

An ODD includes characteristics such as:

- **Geographic location:** roads, highways, or regions where the system is intended to operate.
- **Environmental conditions:** weather and light conditions such as daytime, nighttime, fog, rain, or snow.
- **Traffic conditions:** types of other road users (vehicles, pedestrians, cyclists), traffic density, and road infrastructure.
- **Operational constraints:** legal restrictions, speed limits, or other rules that the system must adhere to

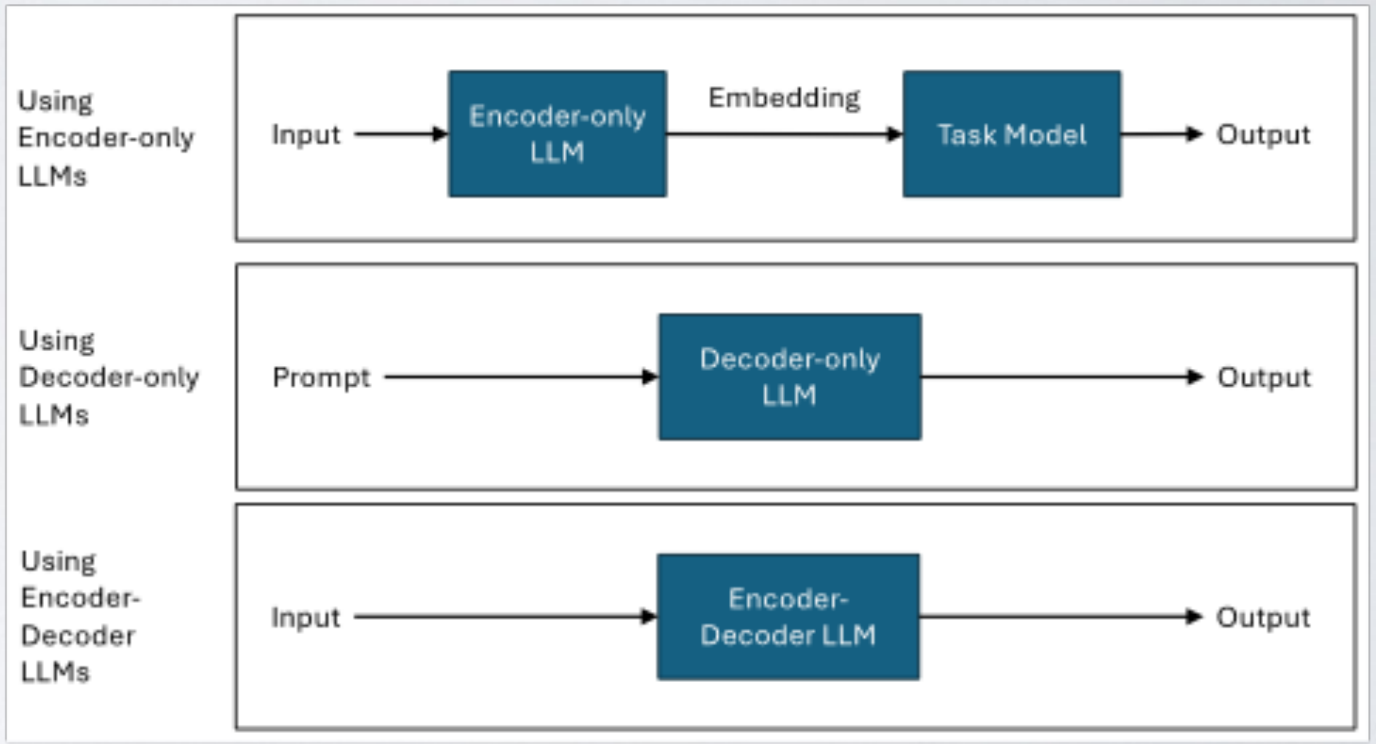
LLMs in Requirements Engineering



Vogelsang and Fishbach, "Using Large Language Models for Natural Language Processing Tasks in Requirements Engineering: A Systematic Guideline", ArXiv, 2024

LLMs in Requirements Engineering

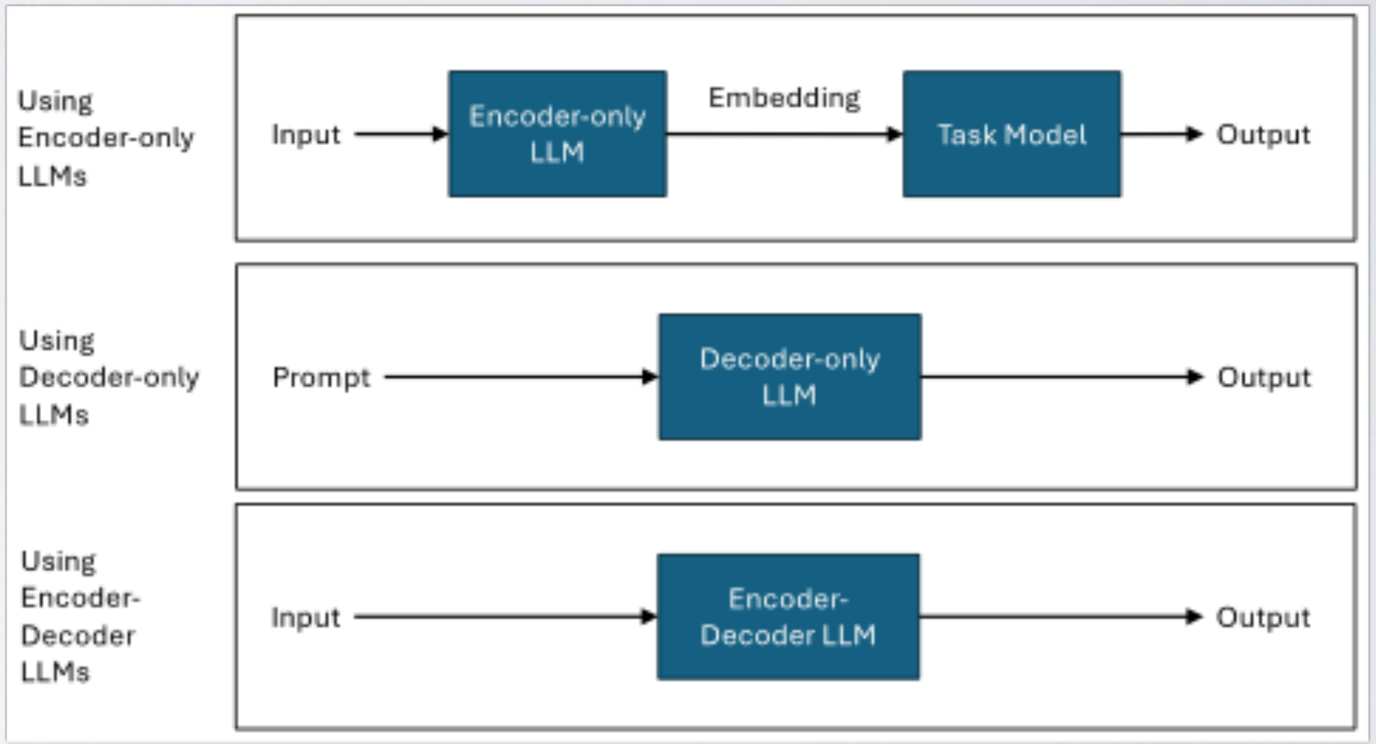
- **Requirements generation**



Vogelsang and Fishbach, "Using Large Language Models for Natural Language Processing Tasks in Requirements Engineering: A Systematic Guideline", ArXiv, 2024

LLMs in Requirements Engineering

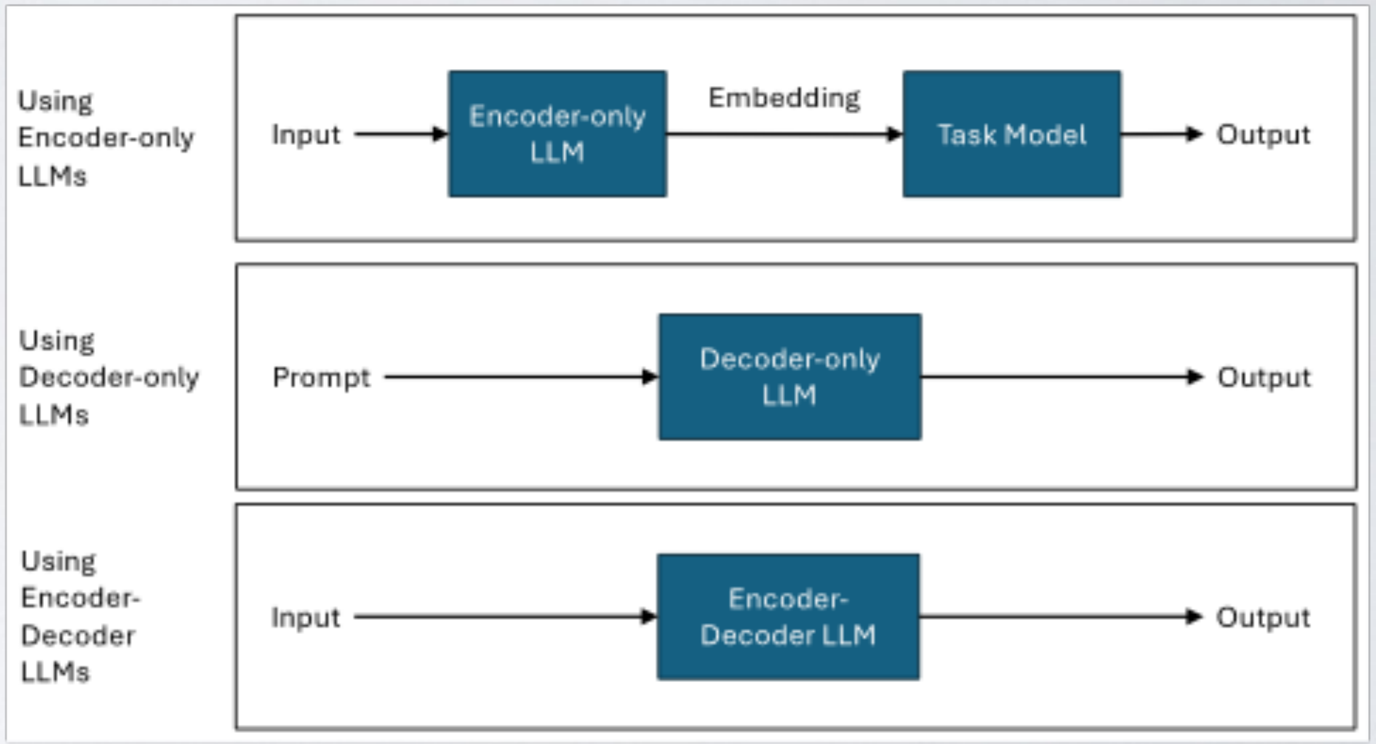
- **Requirements generation**
- **Requirements completion**



Vogelsang and Fishbach, “Using Large Language Models for Natural Language Processing Tasks in Requirements Engineering: A Systematic Guideline”, ArXiv, 2024

LLMs in Requirements Engineering

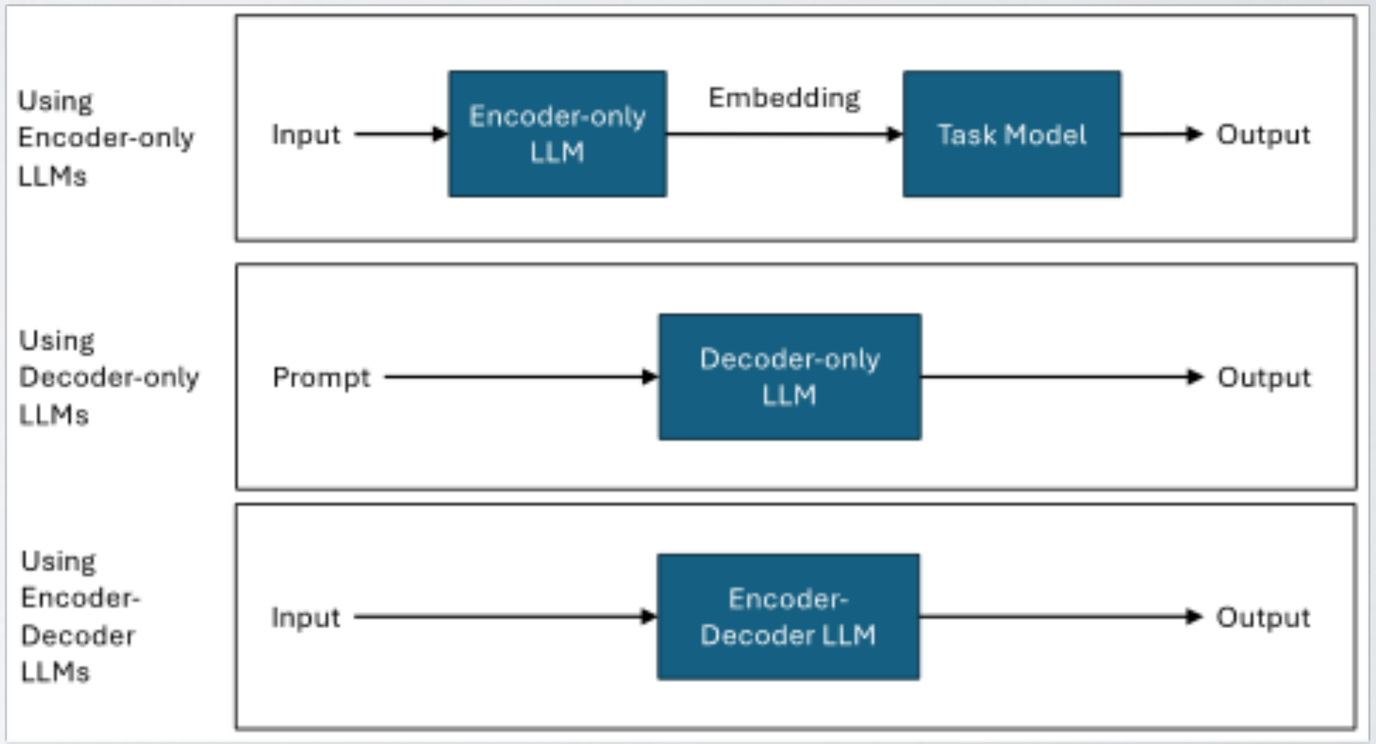
- Requirements generation
- Requirements completion
- Requirements to test cases



Vogelsang and Fishbach, "Using Large Language Models for Natural Language Processing Tasks in Requirements Engineering: A Systematic Guideline", ArXiv, 2024

LLMs in Requirements Engineering

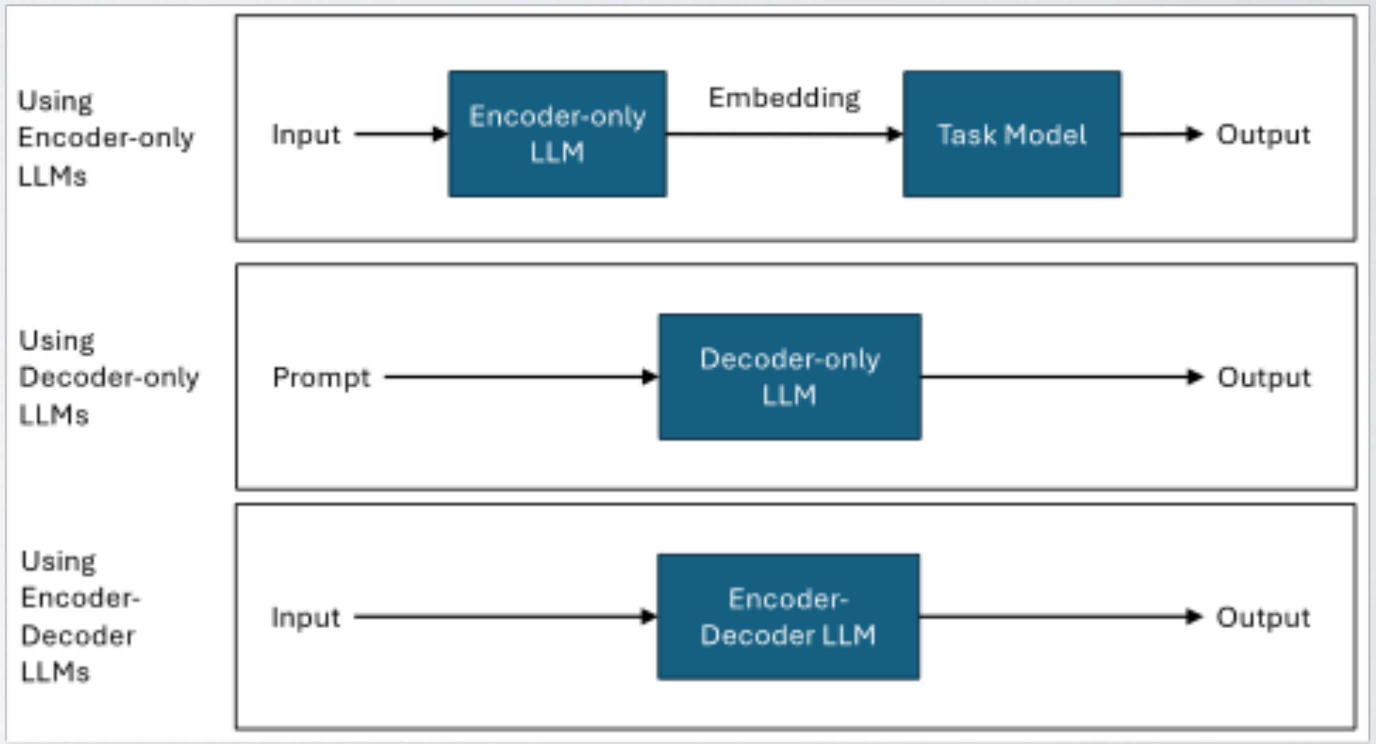
- Requirements generation
- Requirements completion
- Requirements to test cases
- Requirements classification



Vogelsang and Fishbach, "Using Large Language Models for Natural Language Processing Tasks in Requirements Engineering: A Systematic Guideline", ArXiv, 2024

LLMs in Requirements Engineering

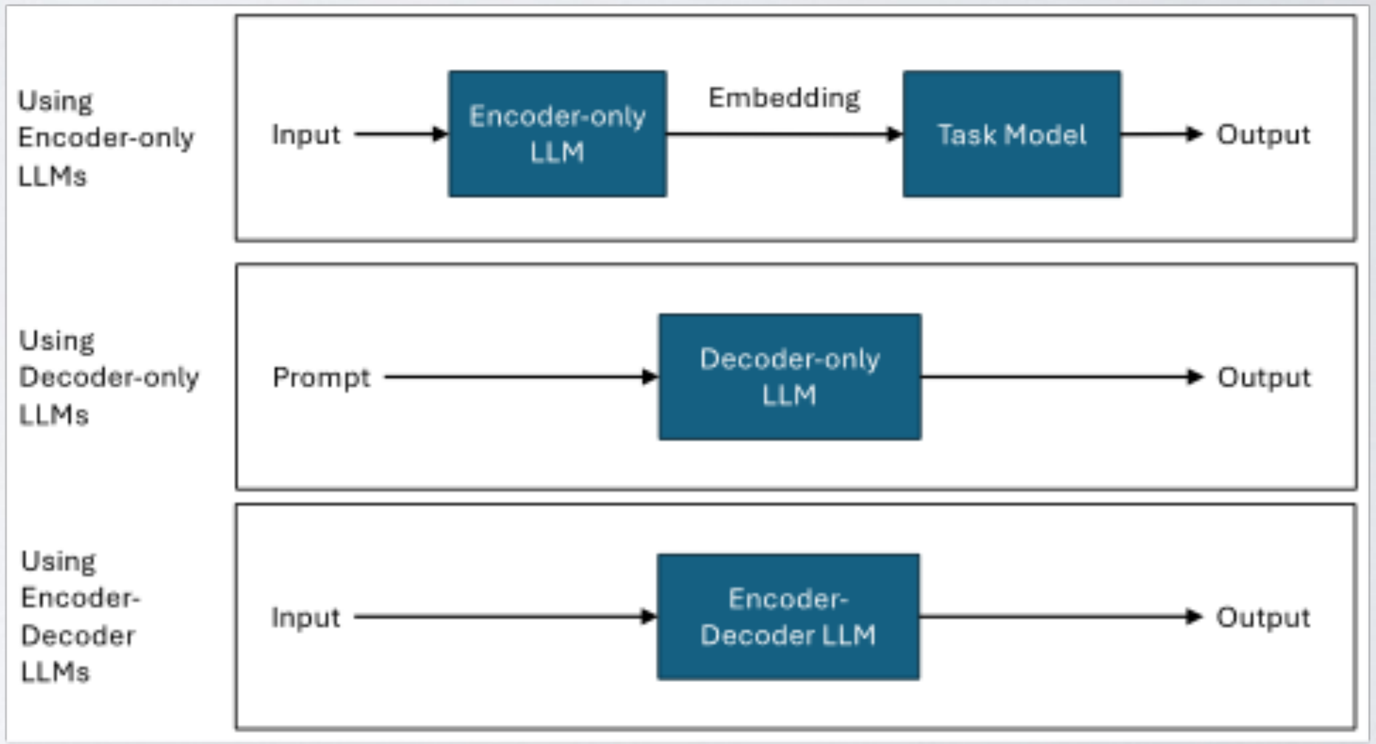
- Requirements generation
- Requirements completion
- Requirements to test cases
- Requirements classification
- …



Vogelsang and Fishbach, "Using Large Language Models for Natural Language Processing Tasks in Requirements Engineering: A Systematic Guideline", ArXiv, 2024

LLMs in Requirements Engineering

- Requirements generation
- Requirements completion
- Requirements to test cases
- Requirements classification
- …
- May render automation more affordable and practical



Vogelsang and Fishbach, "Using Large Language Models for Natural Language Processing Tasks in Requirements Engineering: A Systematic Guideline", ArXiv, 2024

Precise and Complete Requirements? An Elusive Goal

Lionel Briand

MO2RE 2024 Keynote

http://www.lbriand.info

SyMeCo Fellowship

- SyMeCo is a Marie Skłodowska-Curie postdoctoral fellowship programme coordinated by Lero
- Co-funded by Science Foundation Ireland and the EU
- 16 fellowships of 2-year duration based in Ireland across 8 Higher Education Institutions
- Open to researchers of any nationality

Prof Lionel Briand
lionel.briand@lero.ie

Prof Ita Richardson
ita.richardson@lero.ie

SyMeCo

2nd Call for Proposals

NOW OPEN

6 postdoc fellowships

Deadline: 19th of June, 2024

SCAN ME

https://symeco.lero.ie/

Science Foundation Ireland sfi For what's next

Co-funded by the European Union

LERO