# Logical Structure Extraction from Software Requirements Documents

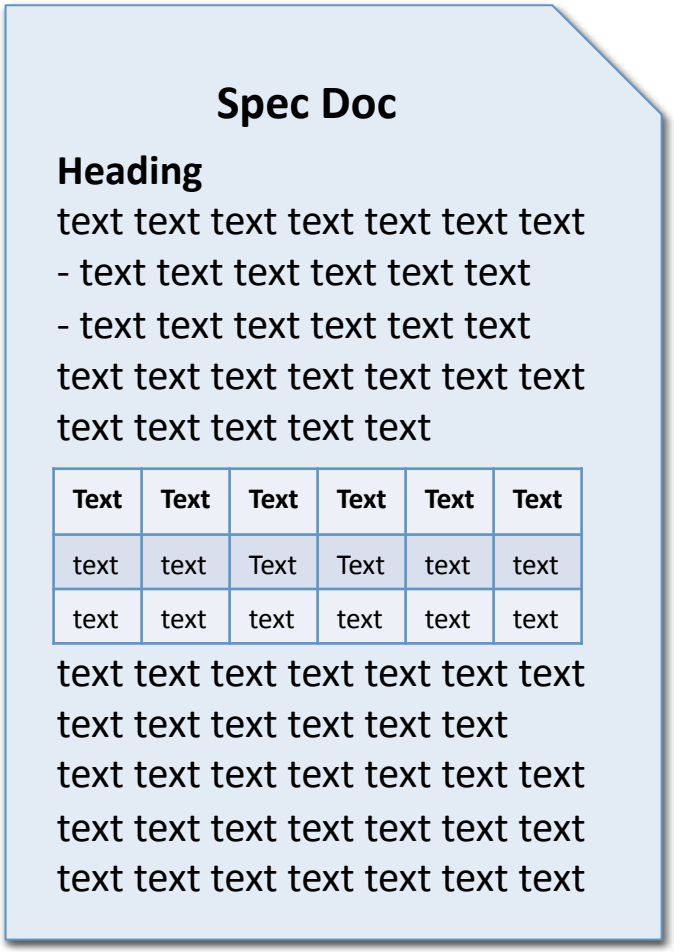
**Rehan Rauf, Michal Antkiewicz, Krzysztof Czarnecki**

**Generative Software Development Lab**

**University of Waterloo, Canada**

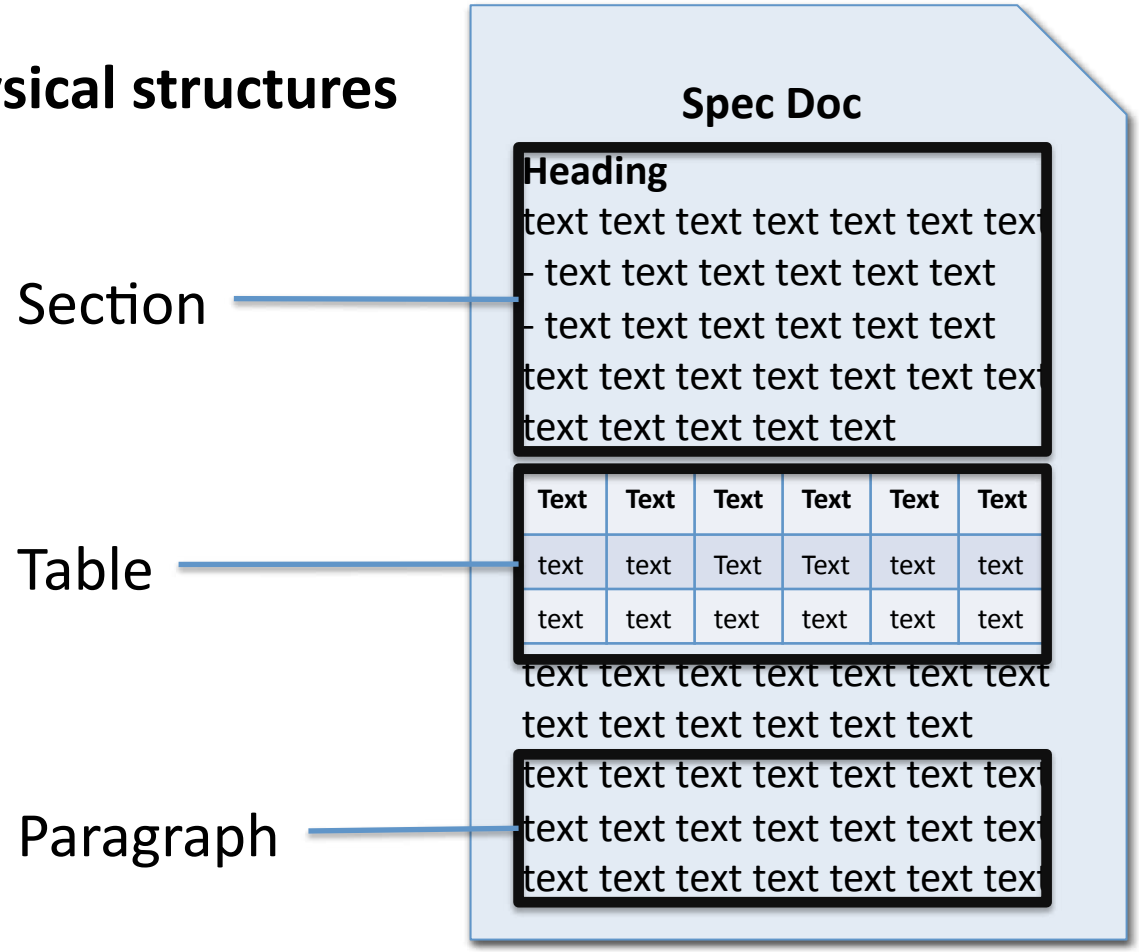
# The Idea

# Specification Documents

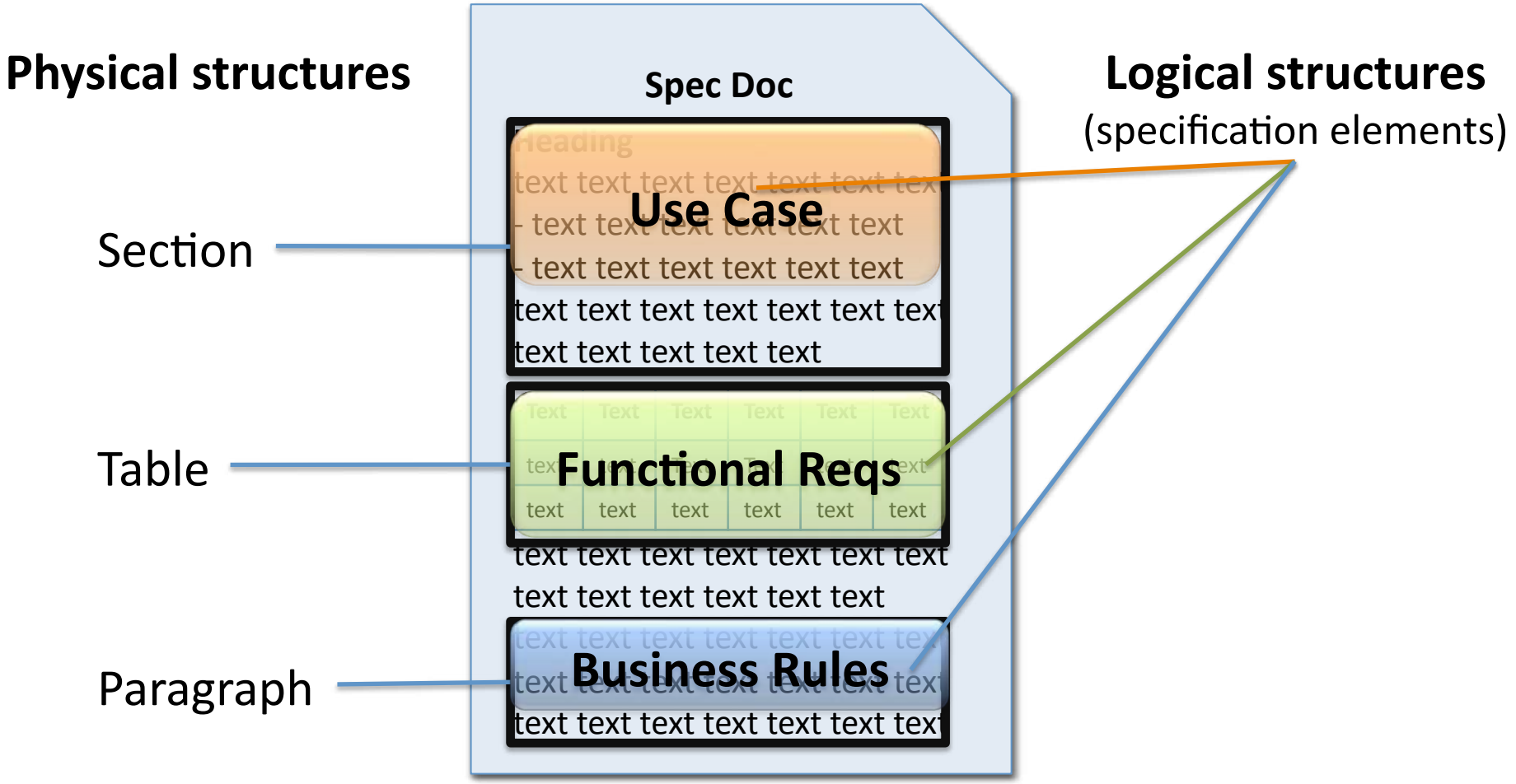


# Specification Documents

**Physical structures**



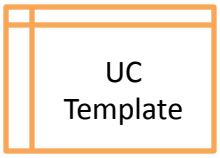
# Specification Documents

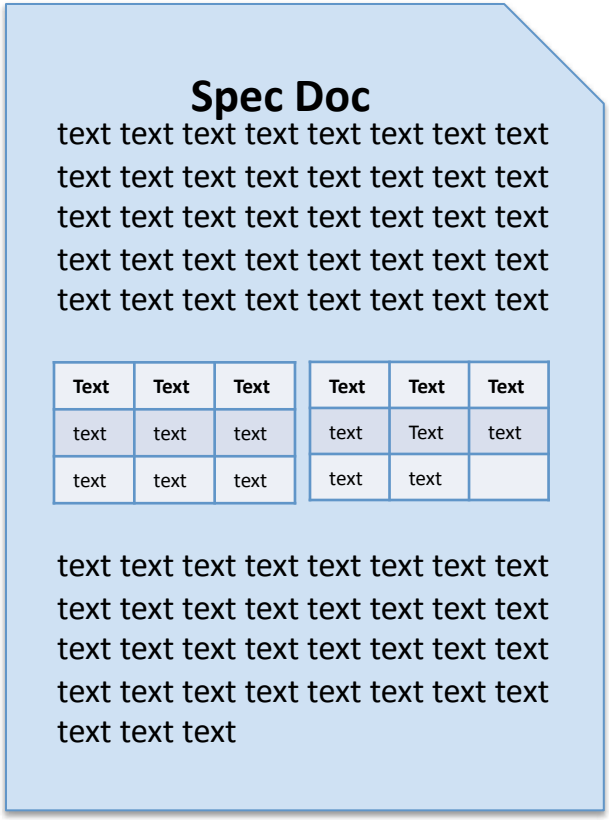


Recognize and extract **specification elements** based on **physical document structure**

# ET – Extraction Tool

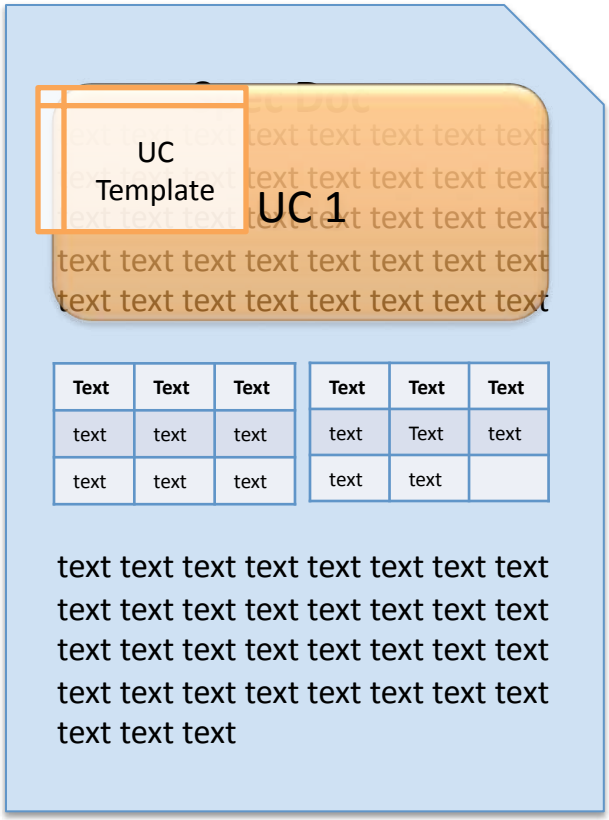
## searches for template instances



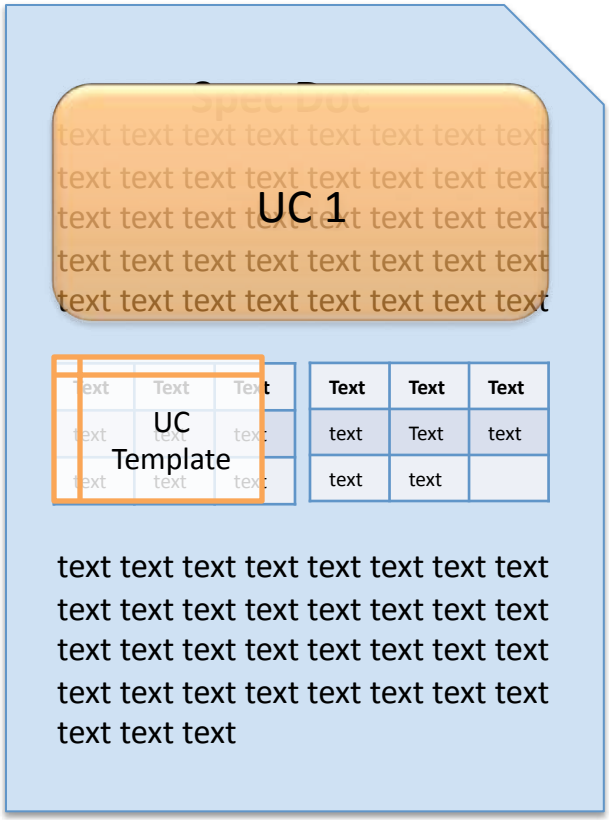
# ET – Extraction Tool

## searches for template instances



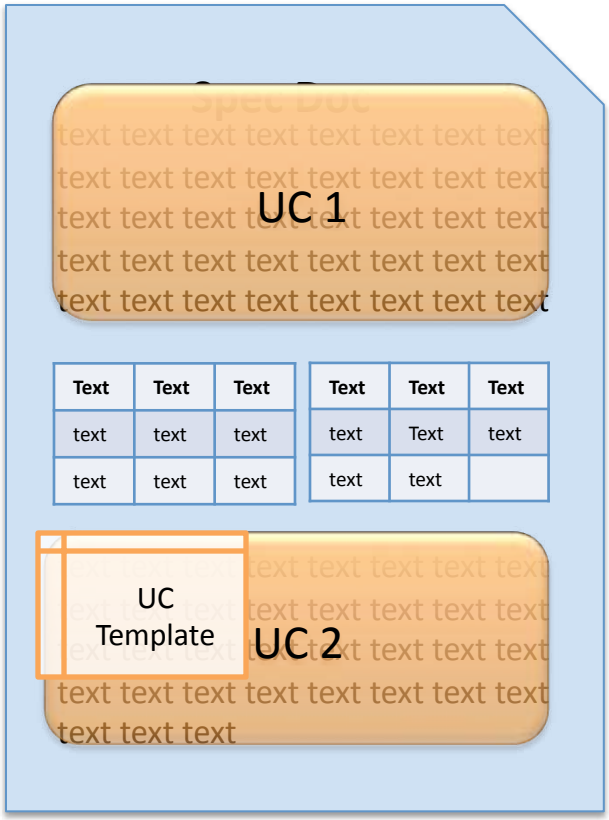
# ET – Extraction Tool

## searches for template instances



# ET – Extraction Tool

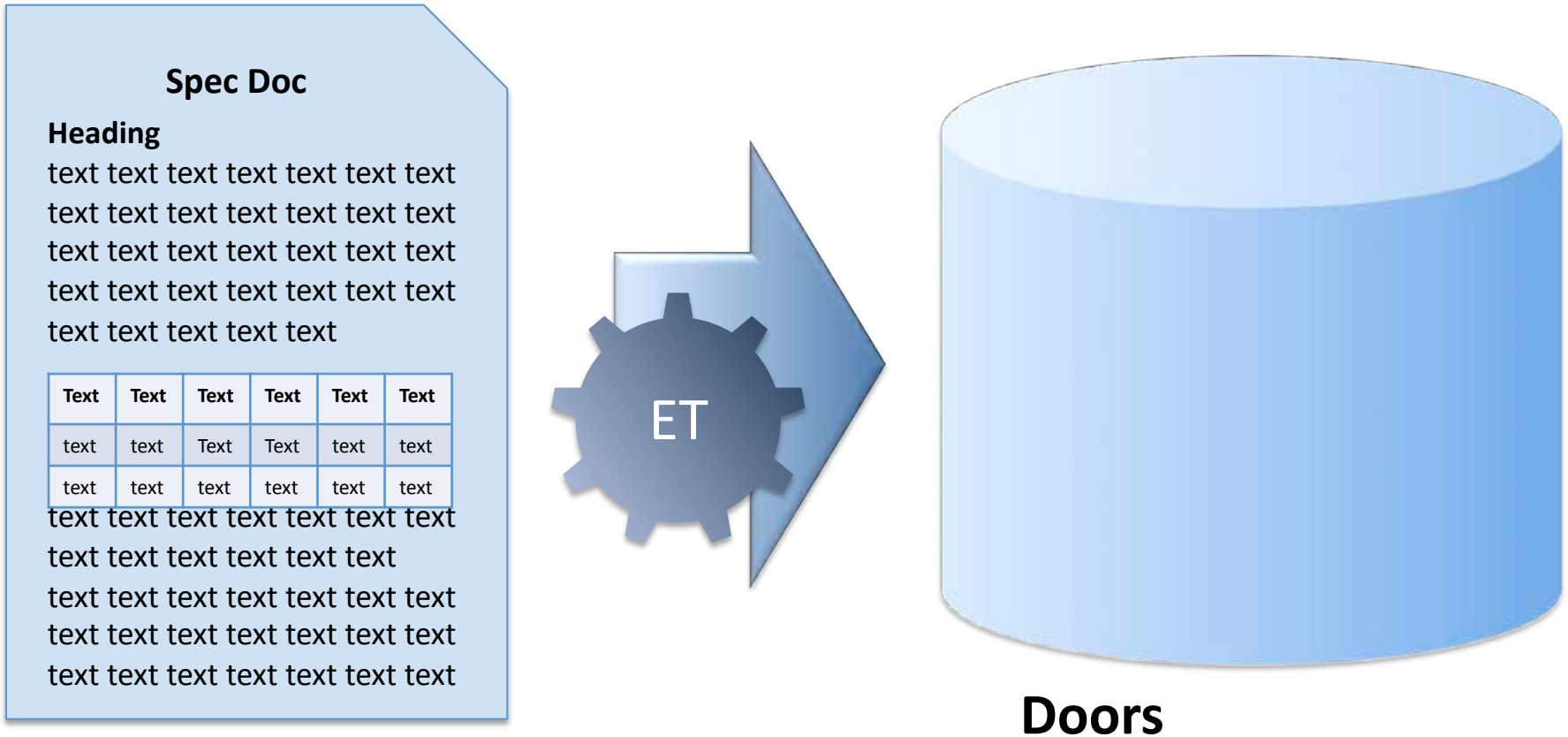
## searches for template instances



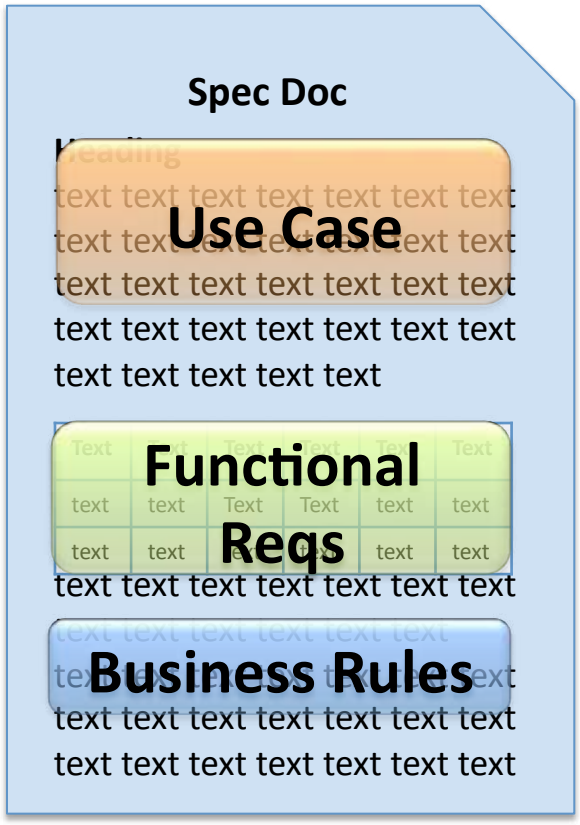
# Assumption: Documents have been authored with some template in mind

# Application scenarios

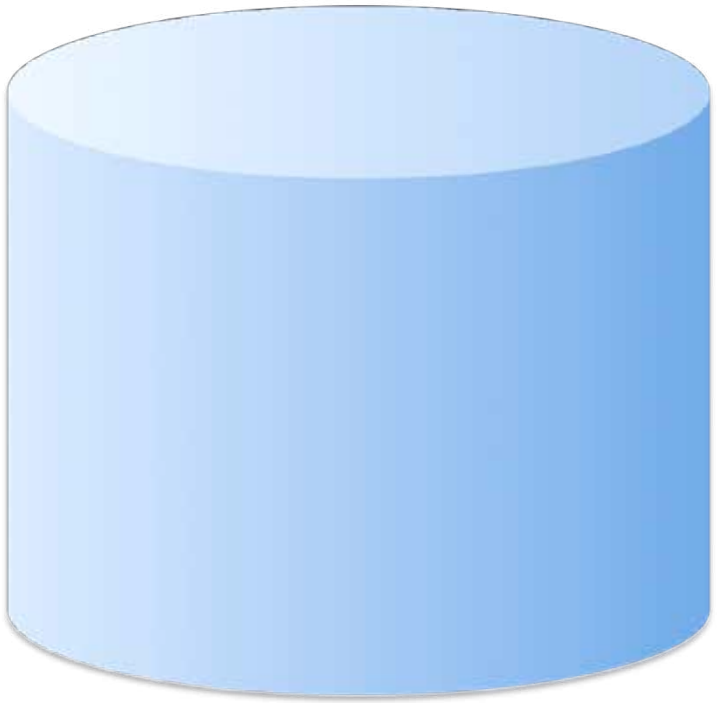
# Import to Requirements Mgmt Tools



# Import to Requirements Mgmt Tools

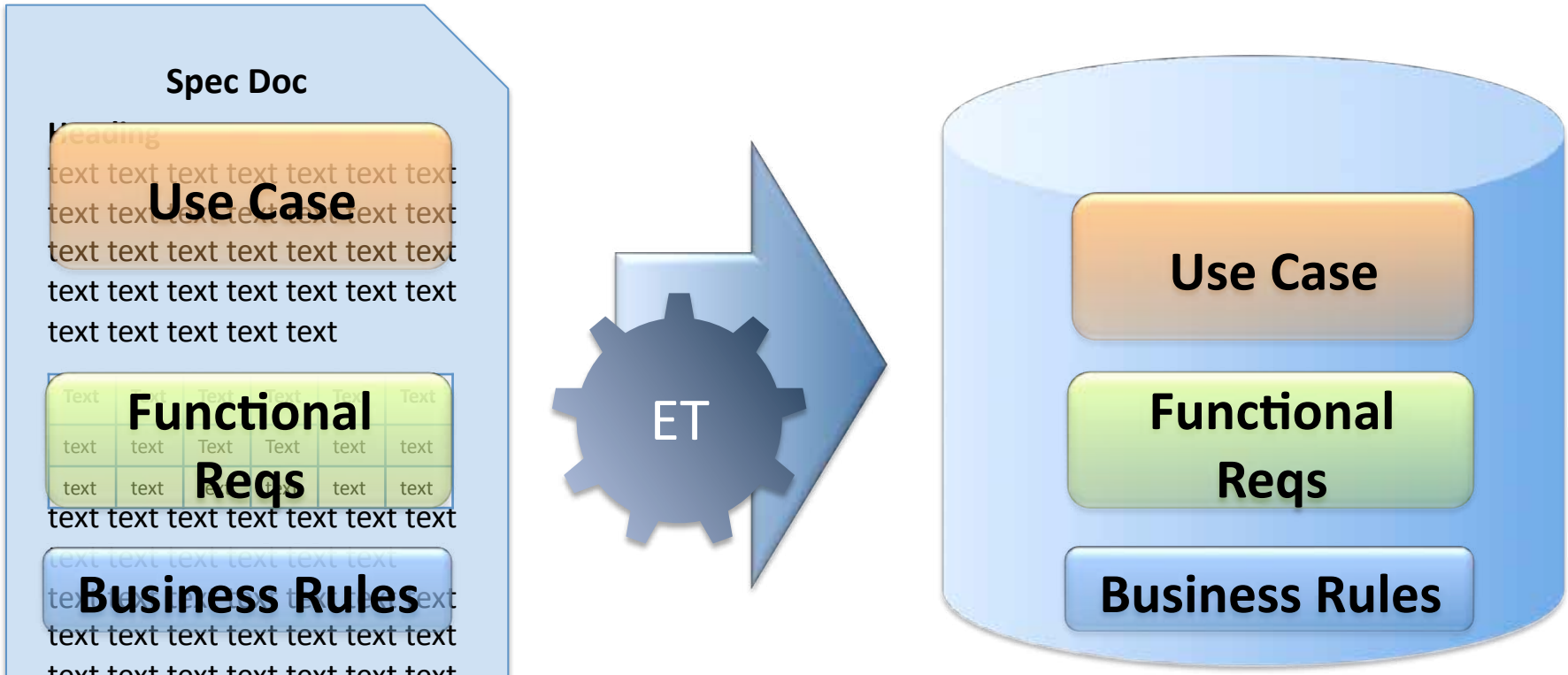






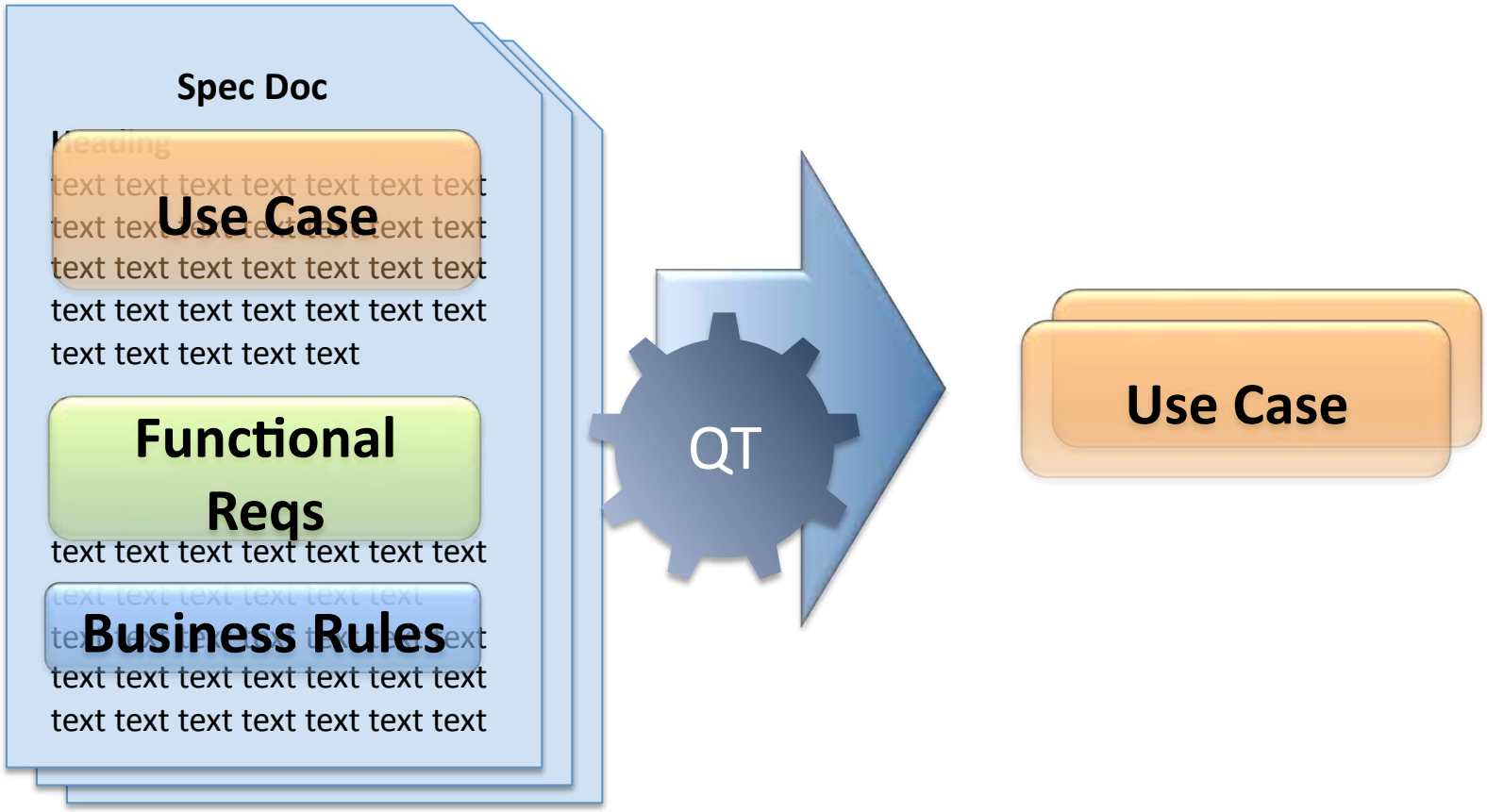
**Doors**
**HP Quality Center**
**Requisite Pro**
**…**

# Import to Requirements Mgmt Tools



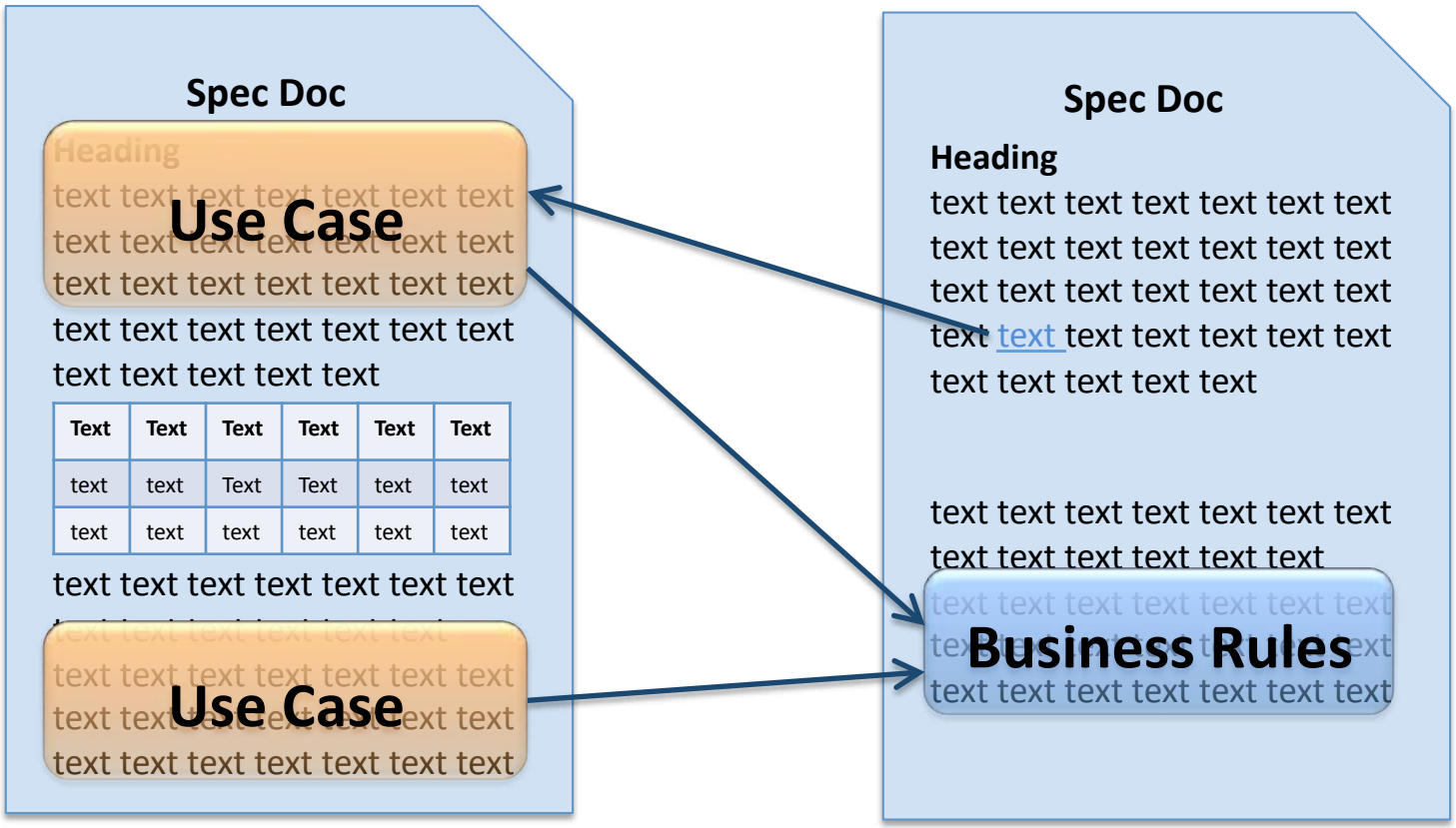

**Doors**
**HP Quality Center**
**Requisite Pro**
**…**

# Structured Query

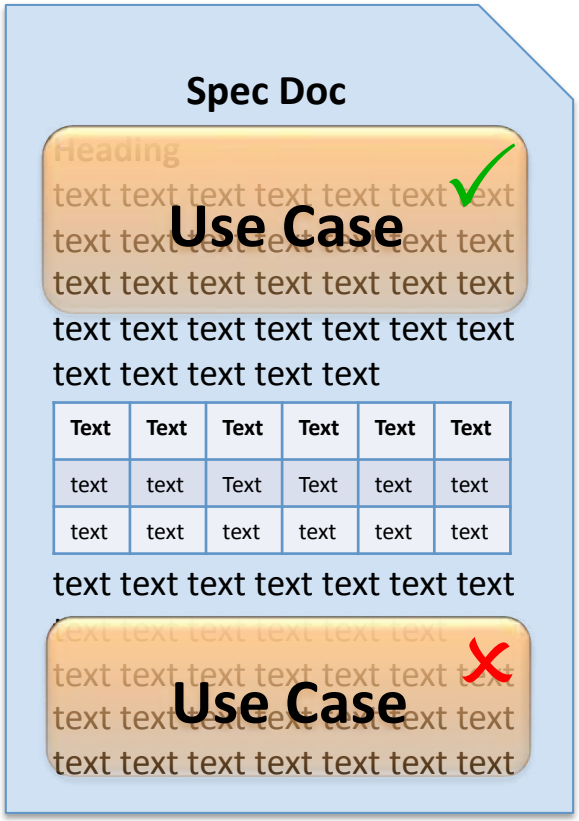



All use cases with actor = ‘customer’

# Tracing


Spec Doc
Heading
text text text text text text text
text text text text text text text
text text text text text text text
Use Case
text text text text text text text
text text text text text
Text Text Text Text Text Text
text text Text Text text text
text text text text text text
text text text text text text text
text text text text text text text
Use Case
Spec Doc
Heading
text text text text text text text
text text text text text text text
text text text text text text text
text text text text text text text
text text text text text
text text text text text text text
text text text text text text
Business Rules

# Template Conformance Checking



# Main Challenge: Logical and Physical Variation

# Challenge – Variation

**UC 1. Select Product Category**

1. User selects the option to display all product categories
2. System displays the list of product categories
3. User selects one product category
4. System shows the list of products in the selected product category

**Extensions:**

**3 .a** User chooses to sort the product categories list by name

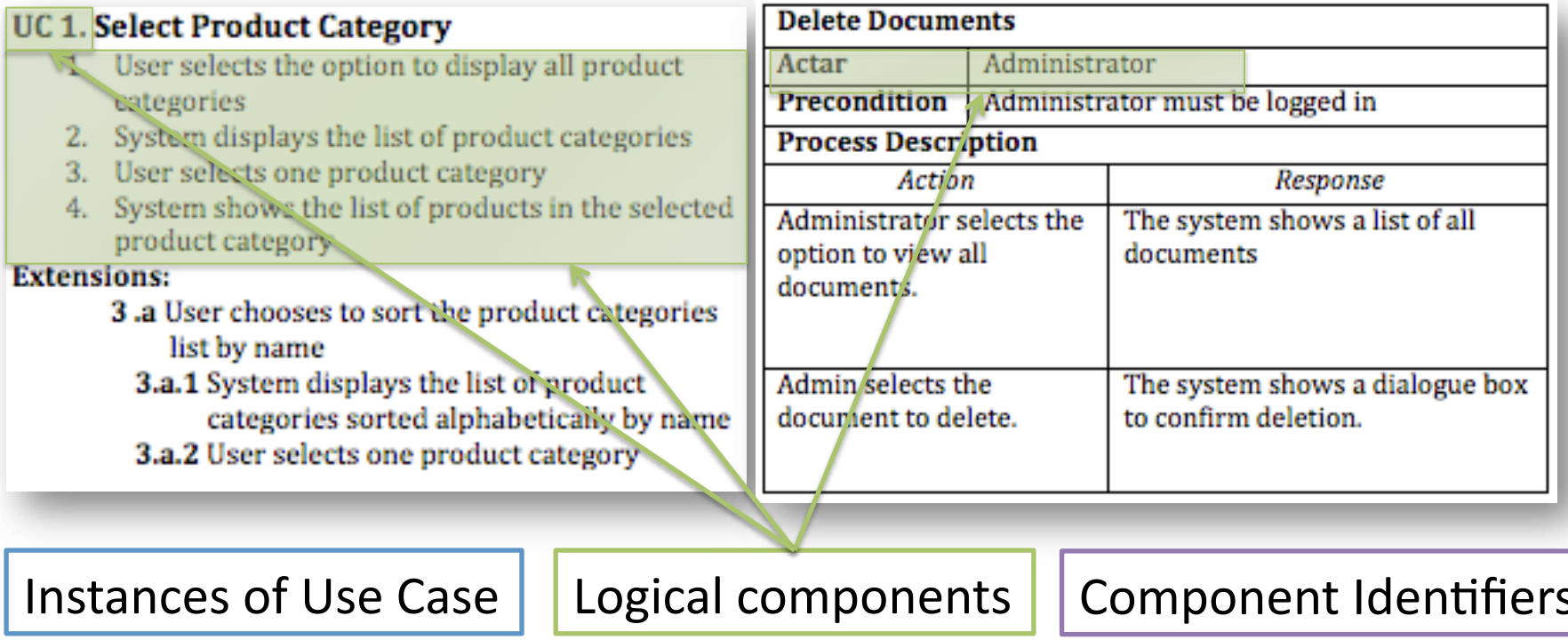
**3.a.1** System displays the list of product categories sorted alphabetically by name

**3.a.2** User selects one product category

| **Delete Documents** | |
|---|---|
| **Actar** | Administrator |
| **Precondition** | Administrator must be logged in |
| **Process Description** | |
| *Action* | *Response* |
| Administrator selects the option to view all documents. | The system shows a list of all documents |
| Admin selects the document to delete. | The system shows a dialogue box to confirm deletion. |

Instances of Use Case

# Challenge – Variation

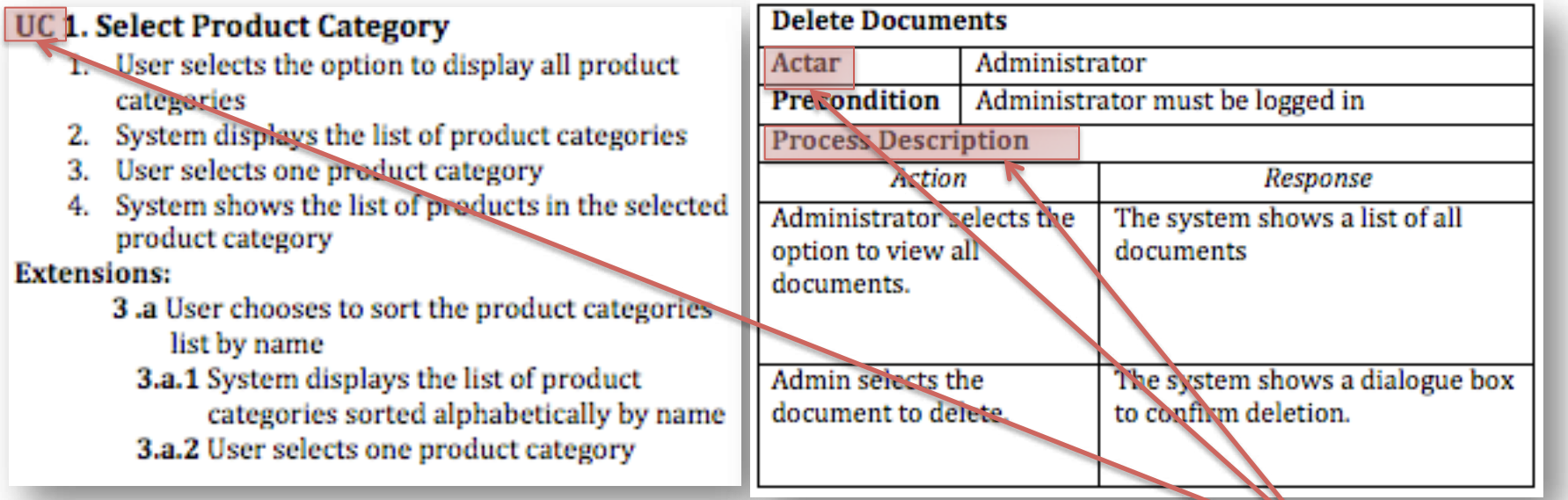



Instances of Use Case

Logical components

Component Identifiers

# Challenge – Variation



**UC 1. Select Product Category**

1. User selects the option to display all product categories
2. System displays the list of product categories
3. User selects one product category
4. System shows the list of products in the selected product category

**Extensions:**

**3 .a** User chooses to sort the product categories list by name

**3.a.1** System displays the list of product categories sorted alphabetically by name

**3.a.2** User selects one product category

| **Delete Documents** | |
|---|---|
| Actar | Administrator |
| **Precondition** | Administrator must be logged in |
| Process Description | |

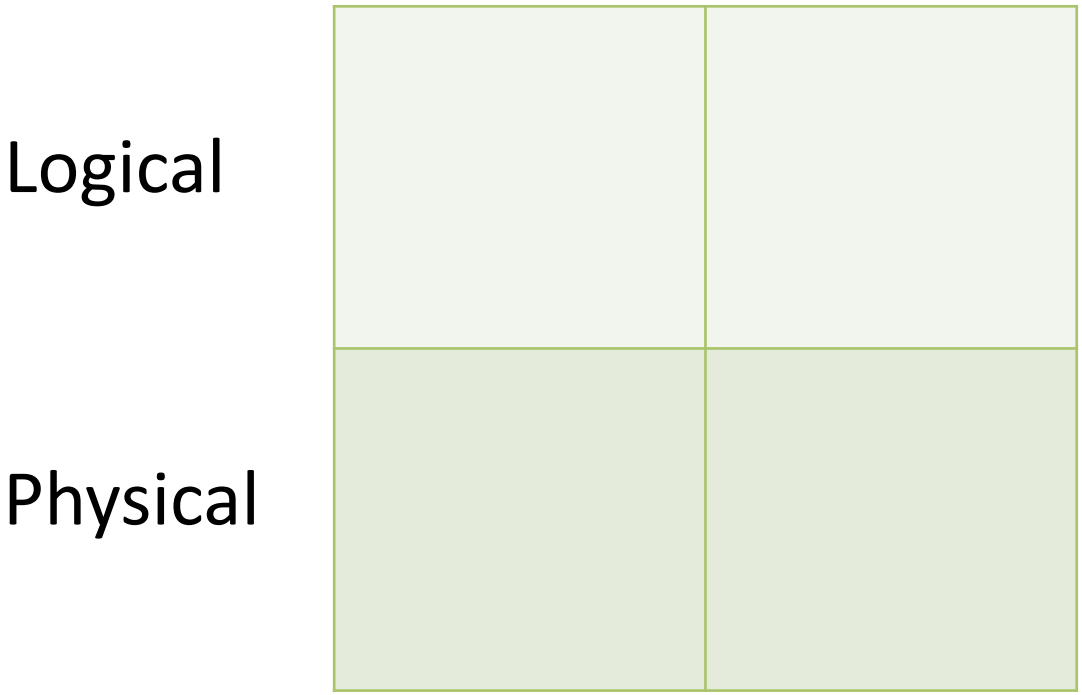
| *Action* | *Response* |
|---|---|
| Administrator selects the option to view all documents. | The system shows a list of all documents |
| Admin selects the document to delete | The system shows a dialogue box to confirm deletion. |

Instances of Use Case

Logical components

Component Identifiers

# Variation Types



# Designed Logical Variation

**UC 1. Select Product Category**

1. User selects the option to display all product categories
2. System displays the list of product categories
3. User selects one product category
4. System shows the list of products in the selected product category

Extensions:

3 .a User chooses to sort the product categories list by name

**3.a.1** System displays the list of product categories sorted alphabetically by name

**3.a.2** User selects one product category

***UC 10. Print Receipt***

1. User selects the option to print receipt
2. System prints a receipt for the purchased items
3. System prints customer name on the envelope

Optional component

# Designed Logical Alternatives

| **Display Public Documents** | |
|---|---|
| **Actor** | User |
| **Main Scenario** | |
| The user selects to the option to view public documents | |
| The system displays list of public documents | |

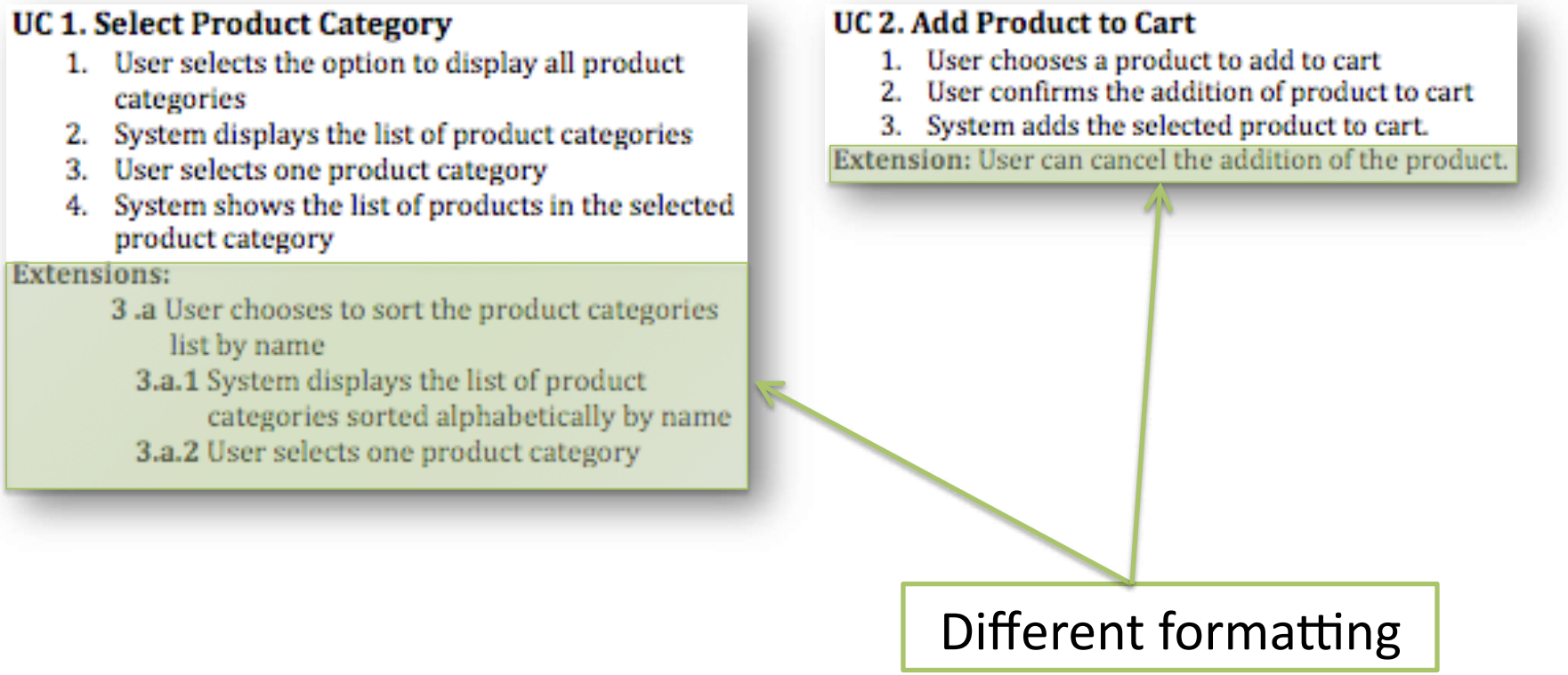
| **Delete Documents** | |
|---|---|
| **Actar** | Administrator |
| **Precondition** | Administrator must be logged in |
| **Process Description** | |

| *Action* | *Response* |
|---|---|
| Administrator selects the option to view all documents. | The system shows a list of all documents |
| Admin selects the document to delete. | The system shows a dialogue box to confirm deletion. |

Deeper decomposition

**Different methodologies lead to logical variation**

# Designed Physical Variation



# Accidental Variation

**Logical**

Missing components, e.g., actor

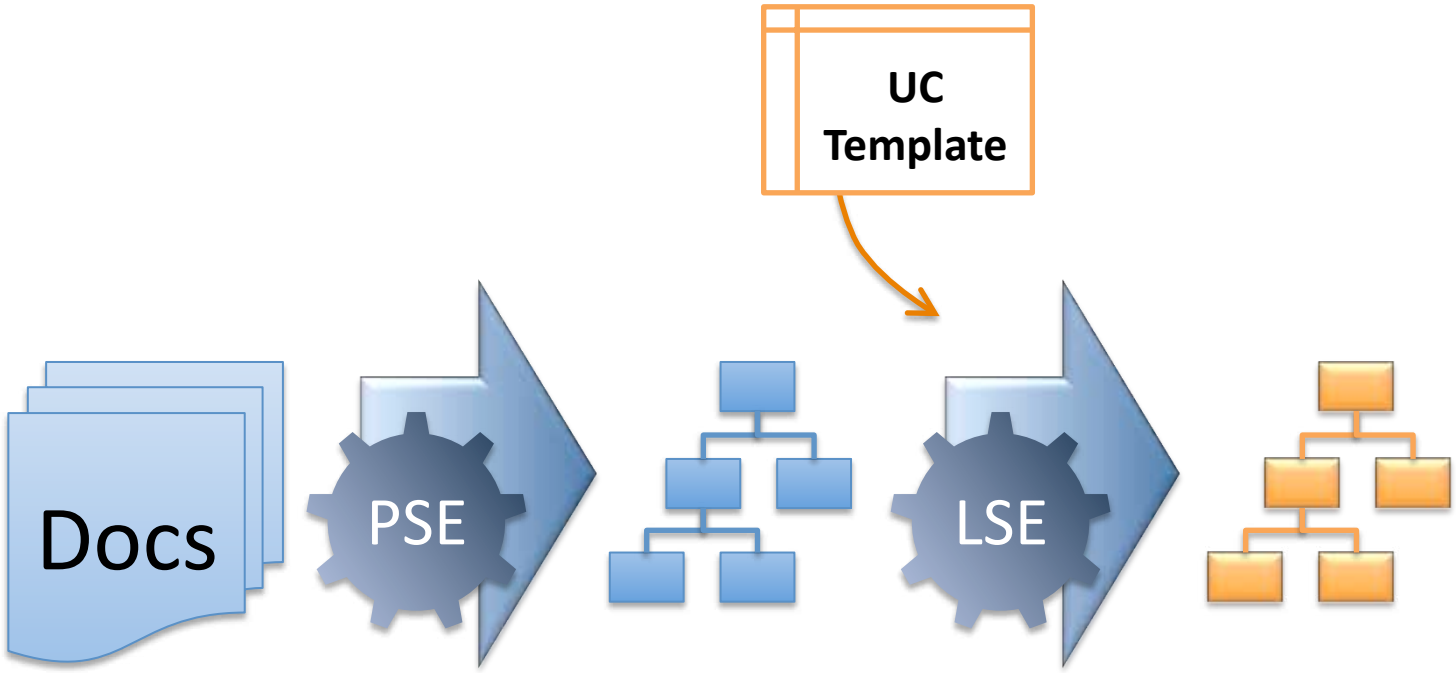
**Physical**

Spelling mistakes, e.g., “Actar”

Style inconsistency, e.g., *italics* instead of **bold**

# Solution

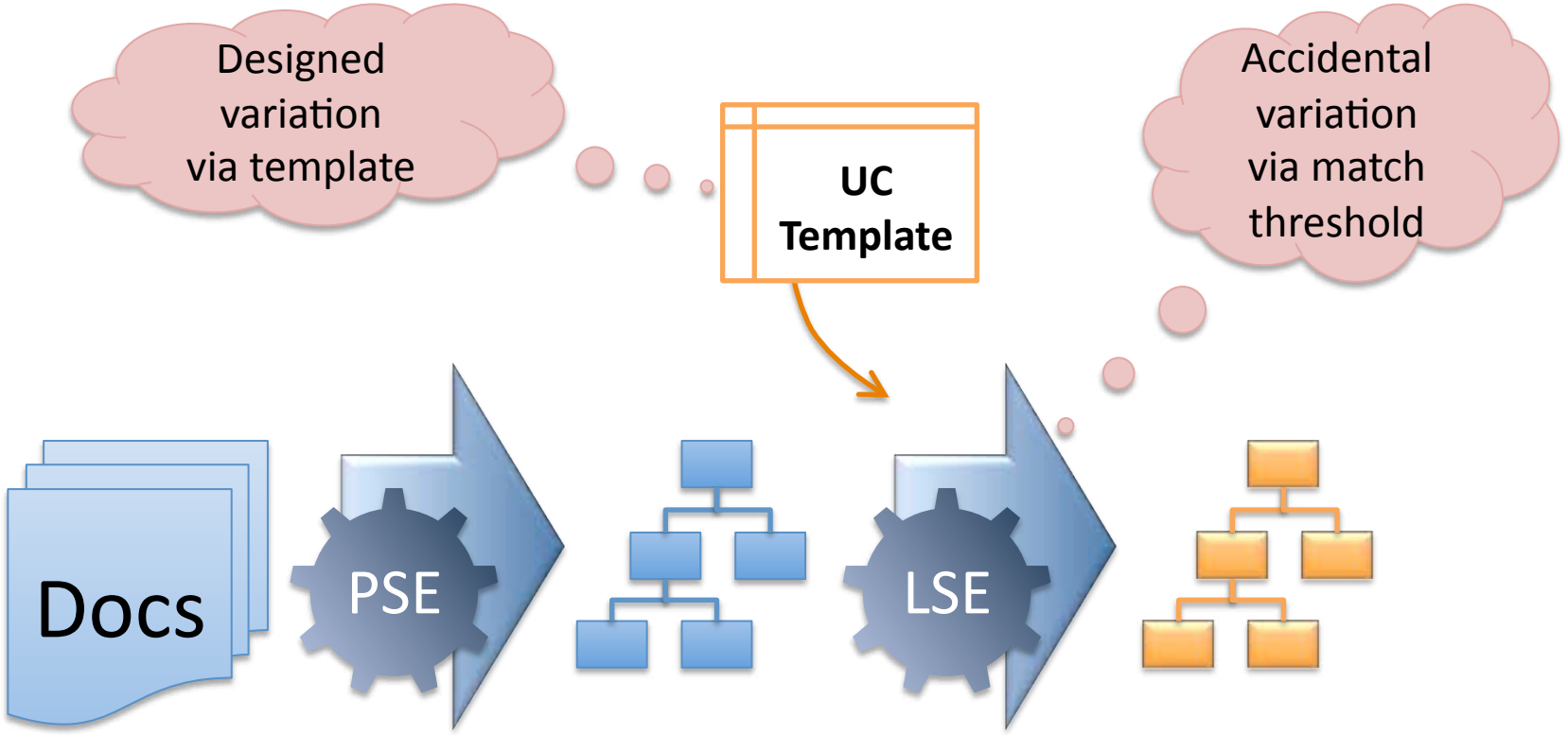
# ET – Extraction Tool




**Physical components**
Sections, lists, table cells

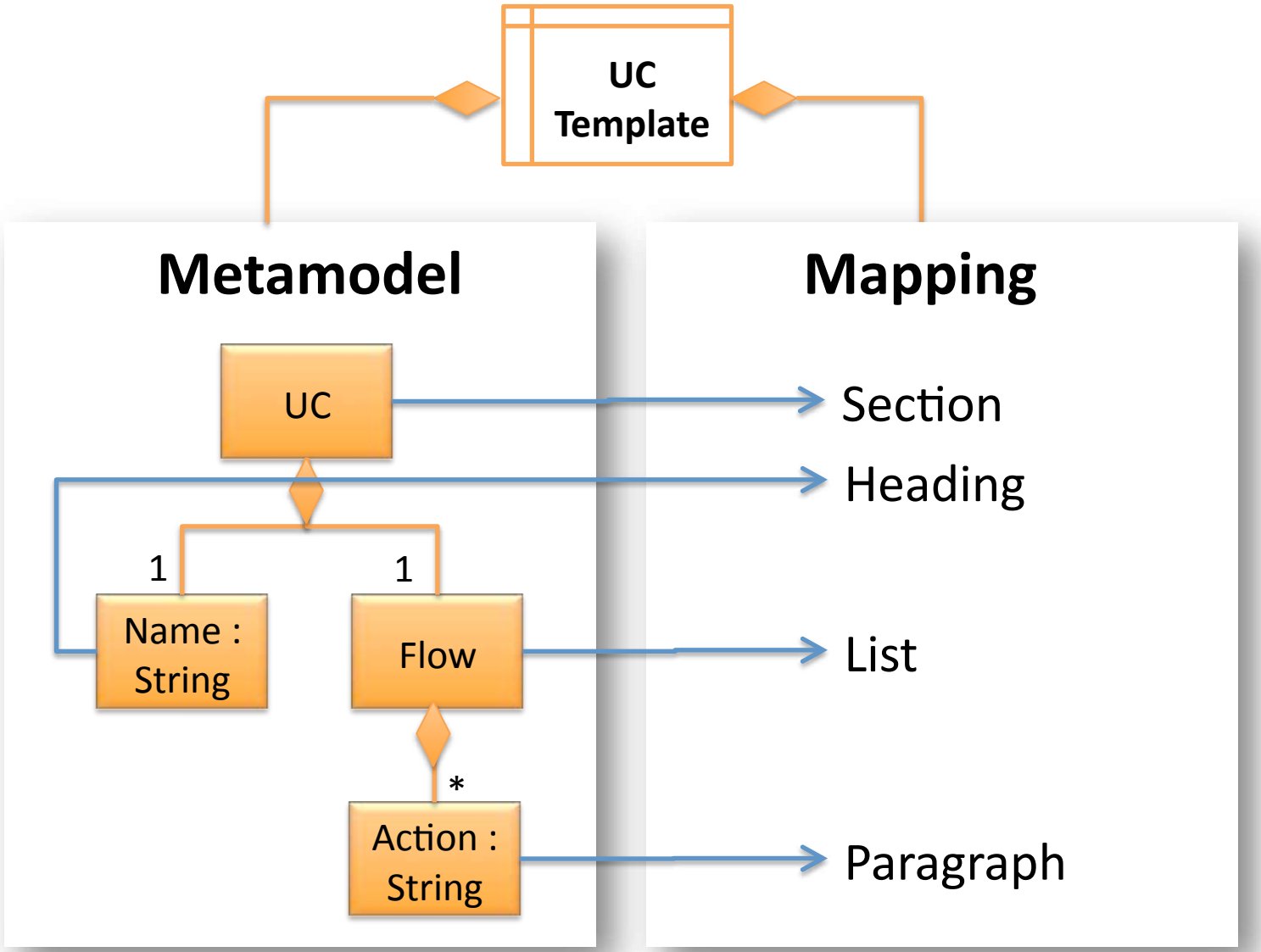
**Logical components**
Actor, flow, extensions

# ET – Extraction Tool




**Physical components**
Sections, lists, table cells

**Logical components**
Actor, flow, extensions




Templates expressed in the lightweight modeling language Clafer

# Example Template

**UC 1. Select Product Category**

1. User selects the option to display all product categories
2. System displays the list of product categories
3. User selects one product category
4. System shows the list of products in the selected product category

**Extensions:**

**3 .a** User chooses to sort the product categories list by name

**3.a.1** System displays the list of product categories sorted alphabetically by name

**3.a.2** User selects one product category

**UC 2. Add Product to Cart**

1. User chooses a product to add to cart
2. User confirms the addition of product to cart
3. System adds the selected product to cart.

**Extension:** User can cancel the addition of the product.

***UC 10. Print Receipt***

1. User selects the option to print receipt
2. System prints a receipt for the purchased items
3. System prints customer name on the envelope

```
1  abstract UseCase1 : LogicalStructure
2    'SectionMapping
3      [sectionTitleStyle = Bold12]

4    ID : LogicalComponent
5      'SectionTitleMapping
6        [sectionTitlePattern="{UC 'NUM'} * ]"

7    Name : LogicalComponent
8      'SectionTitleMapping
9        [sectionTitlePattern=" UC 'NUM' {*} ]"

10   Flow : LogicalComponent
11     'ListMapping
12     FlowItem : LogicalComponent 1..*
13       'ParagraphMapping

14   Extensions : LogicalComponent ?
15     xor Mapping
16       'SectionMapping
17         [sectionTitleText="Extensions:"]

18       'TextBlockMapping
19         [identText="Extension"]
20         [delimiter=":"]
```

# Logical Structure

**UC 1. Select Product Category**

1. User selects the option to display all product categories
2. System displays the list of product categories
3. User selects one product category
4. System shows the list of products in the selected product category

**Extensions:**

**3 .a** User chooses to sort the product categories list by name

**3.a.1** System displays the list of product categories sorted alphabetically by name

**3.a.2** User selects one product category

**UC 2. Add Product to Cart**

1. User chooses a product to add to cart
2. User confirms the addition of product to cart
3. System adds the selected product to cart.

**Extension:** User can cancel the addition of the product.

***UC 10. Print Receipt***

1. User selects the option to print receipt
2. System prints a receipt for the purchased items
3. System prints customer name on the envelope

```
abstract UseCase1 : LogicalStructure
  'SectionMapping
    [sectionTitleStyle = Bold12]

  ID : LogicalComponent
    'SectionTitleMapping
      [sectionTitlePattern="{UC 'NUM'} * ]"

  Name : LogicalComponent
    'SectionTitleMapping
      [sectionTitlePattern=" UC 'NUM' {*} ]"

  Flow : LogicalComponent
    'ListMapping
    FlowItem : LogicalComponent 1..*
      'ParagraphMapping

  Extensions : LogicalComponent ?
    xor Mapping
      'SectionMapping
        [sectionTitleText="Extensions:"]

      'TextBlockMapping
        [identText="Extension"]
        [delimiter=":"]
```

# Mapping

**UC 1. Select Product Category**

1. User selects the option to display all product categories
2. System displays the list of product categories
3. User selects one product category
4. System shows the list of products in the selected product category

**Extensions:**

**3 .a** User chooses to sort the product categories list by name

**3.a.1** System displays the list of product categories sorted alphabetically by name

**3.a.2** User selects one product category

**UC 2. Add Product to Cart**

1. User chooses a product to add to cart
2. User confirms the addition of product to cart
3. System adds the selected product to cart.

**Extension:** User can cancel the addition of the product.

***UC 10. Print Receipt***

1. User selects the option to print receipt
2. System prints a receipt for the purchased items
3. System prints customer name on the envelope

```
abstract UseCase1 : LogicalStructure
  'SectionMapping
    [sectionTitleStyle = Bold12]

  ID : LogicalComponent
    'SectionTitleMapping
      [sectionTitlePattern="{UC 'NUM'} * ]"

  Name : LogicalComponent
    'SectionTitleMapping
      [sectionTitlePattern=" UC 'NUM' {*} ]"

  Flow : LogicalComponent
    'ListMapping
    FlowItem : LogicalComponent 1..*
      'ParagraphMapping

  Extensions : LogicalComponent ?
    xor Mapping
      'SectionMapping
        [sectionTitleText="Extensions:"]

      'TextBlockMapping
        [identText="Extension"]
        [delimiter=":"]
```

# Regular Expressions

**UC 1. Select Product Category**

1. User selects the option to display all product categories
2. System displays the list of product categories
3. User selects one product category
4. System shows the list of products in the selected product category

**Extensions:**

**3 .a** User chooses to sort the product categories list by name

**3.a.1** System displays the list of product categories sorted alphabetically by name

**3.a.2** User selects one product category

**UC 2. Add Product to Cart**

1. User chooses a product to add to cart
2. User confirms the addition of product to cart
3. System adds the selected product to cart.

**Extension:** User can cancel the addition of the product.

***UC 10. Print Receipt***

1. User selects the option to print receipt
2. System prints a receipt for the purchased items
3. System prints customer name on the envelope

```
abstract UseCase1 : LogicalStructure
  'SectionMapping
    [sectionTitleStyle = Bold12]

  ID : LogicalComponent
    'SectionTitleMapping
      [sectionTitlePattern="{UC 'NUM'} * ]"

  Name : LogicalComponent
    'SectionTitleMapping
      [sectionTitlePattern=" UC 'NUM' {*} ]"

  Flow : LogicalComponent
    'ListMapping
    FlowItem : LogicalComponent 1..*
      'ParagraphMapping

  Extensions : LogicalComponent ?
    xor Mapping
      'SectionMapping
        [sectionTitleText="Extensions:"]

      'TextBlockMapping
        [identText="Extension"]
        [delimiter=":"]
```

# Lists

**UC 1. Select Product Category**

1. User selects the option to display all product categories
2. System displays the list of product categories
3. User selects one product category
4. System shows the list of products in the selected product category

**Extensions:**

**3 .a** User chooses to sort the product categories list by name

**3.a.1** System displays the list of product categories sorted alphabetically by name

**3.a.2** User selects one product category

**UC 2. Add Product to Cart**

1. User chooses a product to add to cart
2. User confirms the addition of product to cart
3. System adds the selected product to cart.

**Extension:** User can cancel the addition of the product.

***UC 10. Print Receipt***

1. User selects the option to print receipt
2. System prints a receipt for the purchased items
3. System prints customer name on the envelope

```
abstract UseCase1 : LogicalStructure
  'SectionMapping
    [sectionTitleStyle = Bold12]

  ID : LogicalComponent
    'SectionTitleMapping
      [sectionTitlePattern="{UC 'NUM'} * ]"

  Name : LogicalComponent
    'SectionTitleMapping
      [sectionTitlePattern=" UC 'NUM' {*} ]"

Flow : LogicalComponent
  'ListMapping
  FlowItem : LogicalComponent 1..*
    'ParagraphMapping

  Extensions : LogicalComponent ?
    xor Mapping
      'SectionMapping
        [sectionTitleText="Extensions:"]

      'TextBlockMapping
        [identText="Extension"]
        [delimiter=":"]
```

# Component Nesting

**UC 1. Select Product Category**

1. User selects the option to display all product categories
2. System displays the list of product categories
3. User selects one product category
4. System shows the list of products in the selected product category

**Extensions:**

**3 .a** User chooses to sort the product categories list by name

**3.a.1** System displays the list of product categories sorted alphabetically by name

**3.a.2** User selects one product category

**UC 2. Add Product to Cart**

1. User chooses a product to add to cart
2. User confirms the addition of product to cart
3. System adds the selected product to cart.

**Extension:** User can cancel the addition of the product.

***UC 10. Print Receipt***

1. User selects the option to print receipt
2. System prints a receipt for the purchased items
3. System prints customer name on the envelope

```
abstract UseCase1 : LogicalStructure
  'SectionMapping
    [sectionTitleStyle = Bold12]

  ID : LogicalComponent
    'SectionTitleMapping
      [sectionTitlePattern="{UC 'NUM'} * ]"

  Name : LogicalComponent
    'SectionTitleMapping
      [sectionTitlePattern=" UC 'NUM' {*} ]"

  Flow : LogicalComponent
    'ListMapping
    FlowItem : LogicalComponent 1..*
      'ParagraphMapping

  Extensions : LogicalComponent ?
    xor Mapping
      'SectionMapping
        [sectionTitleText="Extensions:"]

      'TextBlockMapping
        [identText="Extension"]
        [delimiter=":"]
```

# Optional Components

**UC 1. Select Product Category**

1. User selects the option to display all product categories
2. System displays the list of product categories
3. User selects one product category
4. System shows the list of products in the selected product category

**Extensions:**

**3 .a** User chooses to sort the product categories list by name

**3.a.1** System displays the list of product categories sorted alphabetically by name

**3.a.2** User selects one product category

**UC 2. Add Product to Cart**

1. User chooses a product to add to cart
2. User confirms the addition of product to cart
3. System adds the selected product to cart.

**Extension:** User can cancel the addition of the produc

***UC 10. Print Receipt***

1. User selects the option to print receipt
2. System prints a receipt for the purchased items
3. System prints customer name on the envelope

```
1  abstract UseCase1 : LogicalStructure
2    'SectionMapping
3      [sectionTitleStyle = Bold12]
4    ID : LogicalComponent
5      'SectionTitleMapping
6        [sectionTitlePattern="{UC 'NUM'} * ]"
7    Name : LogicalComponent
8      'SectionTitleMapping
9        [sectionTitlePattern=" UC 'NUM' {*} ]"
10   Flow : LogicalComponent
11     'ListMapping
12     FlowItem : LogicalComponent 1..*
13       'ParagraphMapping
14   Extensions : LogicalComponent ?
15     xor Mapping
16       'SectionMapping
17         [sectionTitleText="Extensions:"]
18       'TextBlockMapping
19         [identText="Extension"]
20         [delimiter=":"]
```

# Physical Alternatives

**UC 1. Select Product Category**

1. User selects the option to display all product categories
2. System displays the list of product categories
3. User selects one product category
4. System shows the list of products in the selected product category

**Extensions:**

**3 .a** User chooses to sort the product categories list by name

**3.a.1** System displays the list of product categories sorted alphabetically by name

**3.a.2** User selects one product category

**UC 2. Add Product to Cart**

1. User chooses a product to add to cart
2. User confirms the addition of product to cart
3. System adds the selected product to cart.

**Extension:** User can cancel the addition of the product.

***UC 10. Print Receipt***

1. User selects the option to print receipt
2. System prints a receipt for the purchased items
3. System prints customer name on the envelope

```
abstract UseCase1 : LogicalStructure
  'SectionMapping
    [sectionTitleStyle = Bold12]

  ID : LogicalComponent
    'SectionTitleMapping
      [sectionTitlePattern="{UC 'NUM'} * ]"

  Name : LogicalComponent
    'SectionTitleMapping
      [sectionTitlePattern=" UC 'NUM' {*} ]"

  Flow : LogicalComponent
    'ListMapping
    FlowItem : LogicalComponent 1..*
      'ParagraphMapping

  Extensions : LogicalComponent ?
    xor Mapping
      'SectionMapping
        [sectionTitleText="Extensions:"]

      'TextBlockMapping
        [identText="Extension"]
        [delimiter=":"]
```

# Templates with Tables

| **Delete Documents** | |
|---|---|
| **Actar** | Administrator |
| **Precondition** | Administrator must be logged in |
| **Process Description** | |
| *Action* | *Response* |
| Administrator selects the option to view all documents. | The system shows a list of all documents |
| Admin selects the document to delete. | The system shows a dialogue box to confirm deletion. |

| **Display Public Documents** | |
|---|---|
| **Actor** | User |
| **Main Scenario** | |
| The user selects to the option to view public documents | |
| The system displays list of public documents | |

```
abstract UseCase2 : LogicalStructure
  'TableMapping

  Name : LogicalComponent
    'CellMapping
      [colIndex=1
       rowIndex=1]

  Actor : LogicalComponent
    'HCellBlockMapping
      [identText="Actor"]

  Precondition : LogicalComponent ?
    'HCellBlockMapping
      [identText="Precondition"]

  xor Flow
    ProcDesc : LogicalComponent
      'ColumnMapping
        [colTitleText = "Process Description"]
      Action : LogicalComponent
        'ColumnMapping
          [colTitleText = "Action"]
      Response : LogicalComponent
        'ColumnMapping
          [colTitleText = "Response"]
    MainScenario : LogicalComponent
        'ColumnMapping
          [colTitleText = "Main Scenario"]
```

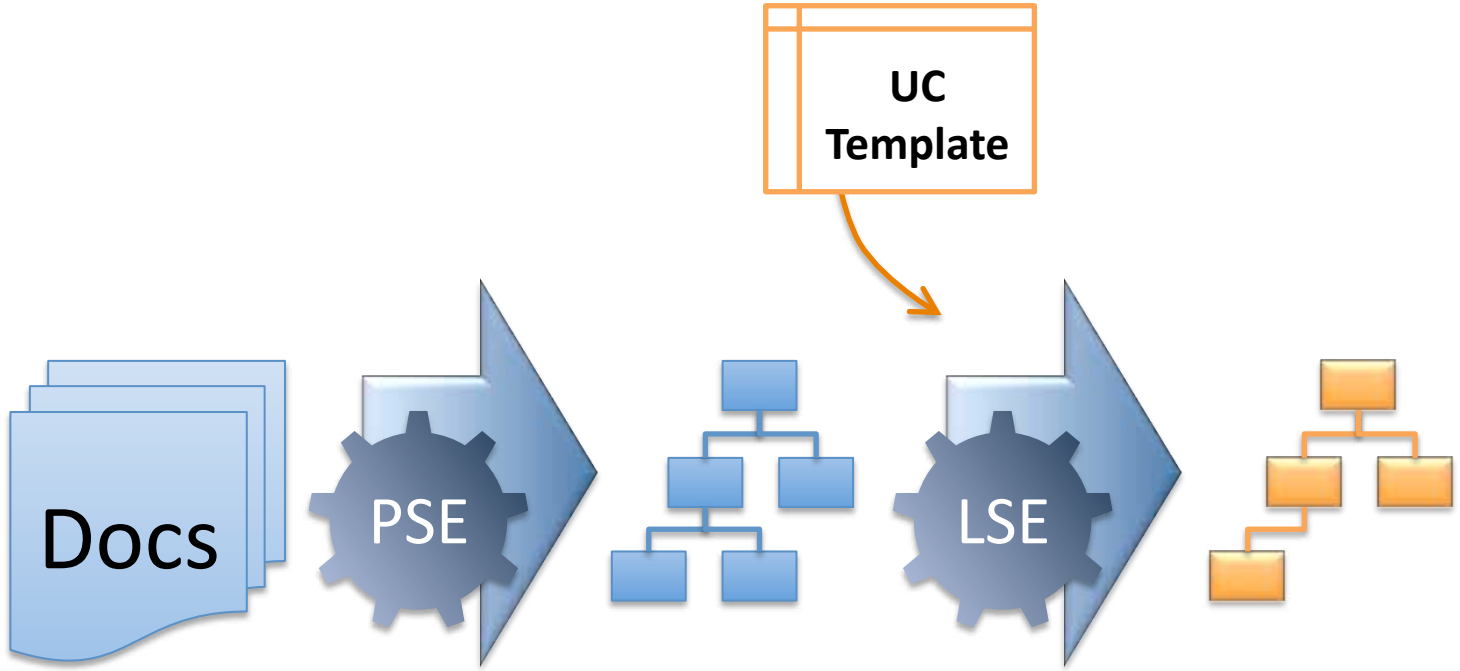
# Logical Alternatives

| **Delete Documents** | |
|---|---|
| **Actar** | Administrator |
| **Precondition** | Administrator must be logged in |
| **Process Description** | |
| *Action* | *Response* |
| Administrator selects the option to view all documents. | The system shows a list of all documents |
| Admin selects the document to delete. | The system shows a dialogue box to confirm deletion. |

| **Display Public Documents** | |
|---|---|
| **Actor** | User |
| **Main Scenario** | |
| The user selects to the option to view public documents | |
| The system displays list of public documents | |

```
abstract UseCase2 : LogicalStructure
  'TableMapping

  Name : LogicalComponent
    'CellMapping
      [colIndex=1
       rowIndex=1]

  Actor : LogicalComponent
    'HCellBlockMapping
      [identText="Actor"]

  Precondition : LogicalComponent ?
    'HCellBlockMapping
      [identText="Precondition"]

xor Flow
  ProcDesc : LogicalComponent
    'ColumnMapping
      [colTitleText = "Process Description"]
    Action : LogicalComponent
      'ColumnMapping
        [colTitleText = "Action"]
    Response : LogicalComponent
      'ColumnMapping
        [colTitleText = "Response"]
  MainScenario : LogicalComponent
      'ColumnMapping
        [colTitleText = "Main Scenario"]
```

# ET – Extraction Tool




**Physical components**

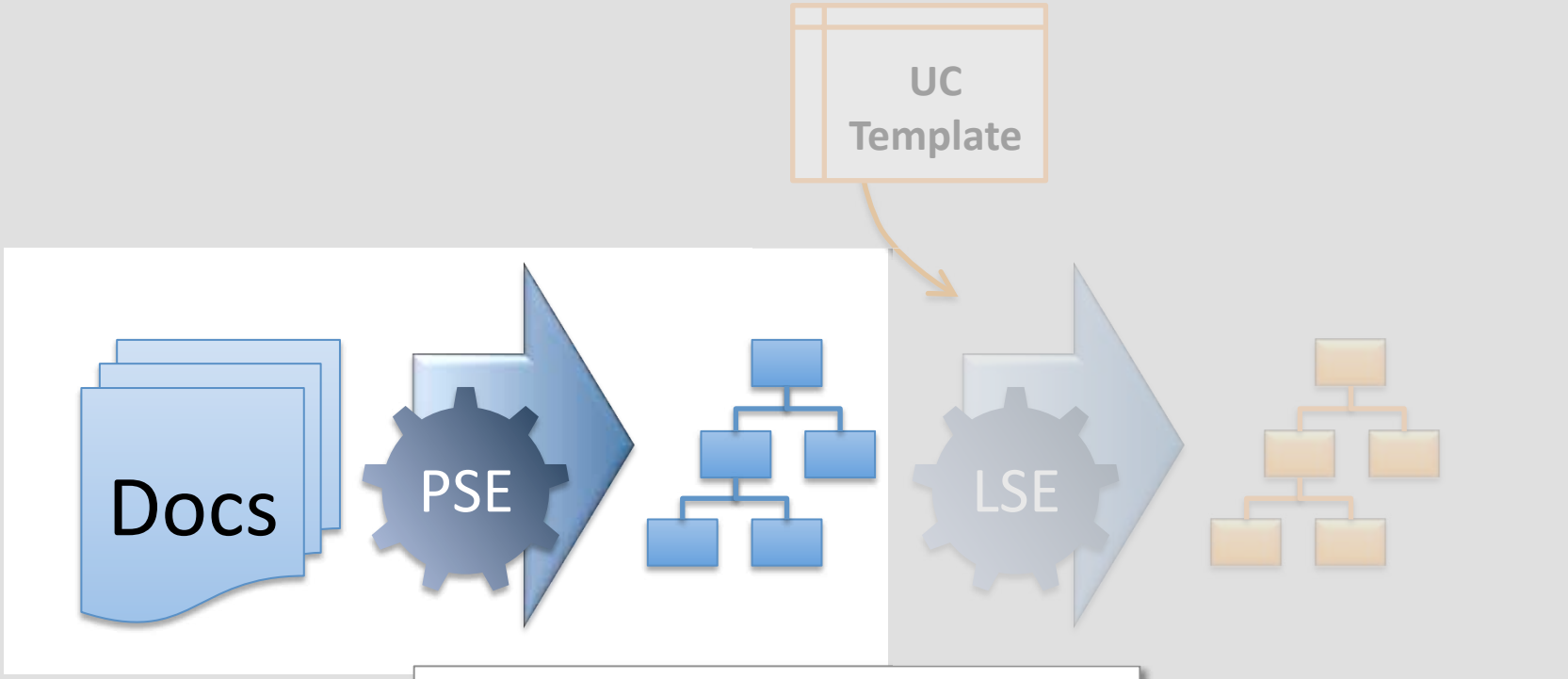
**Basic**: Paragraph, cell, graphic

**Composite**: Sections, lists, tables, …

**Logical components**

Actor, flow, extensions

# Physical Structure Extraction




**Physical components**

**Basic**: Paragraph, cell, graphic

**Composite**: Sections, lists, tables, …

**Logical components**

Actor, flow, extensions

**Only part dependent on document-format**

# Evaluation

# Can we extract logical structures from real-world documents?

# Document Set

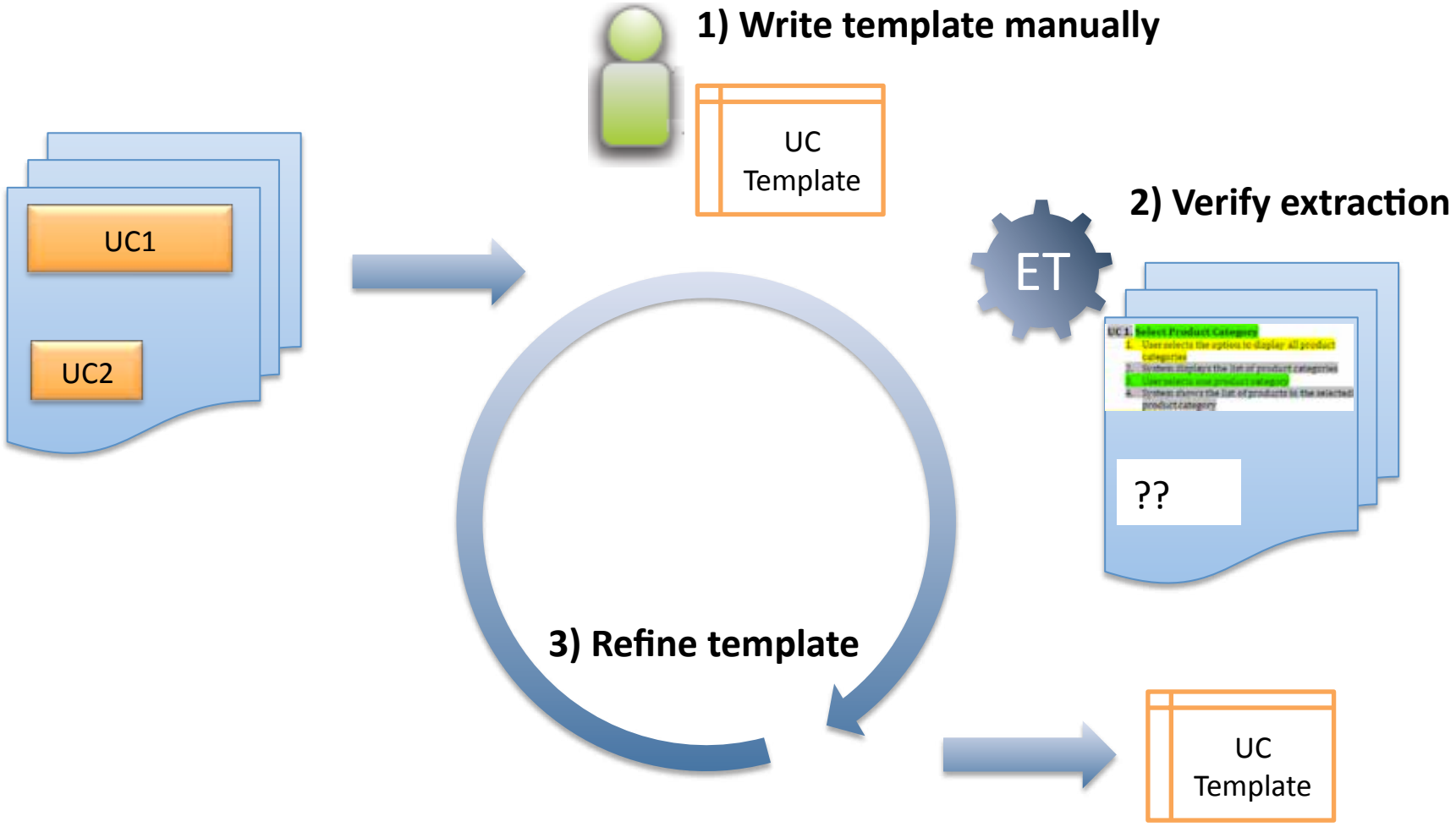
**43 documents**

- 24 from 3 companies
- 11 from public sources
- 6 student projects
- 2,000 to 23,000 words

**Content**

- Use Cases
- Data Objects
- Business Rules
- Functional Reqs
- Non-Functional Reqs
- …

# Template Development



# Results

**36 logical structures**

- Use cases, data objects, business rules, …
- Template sizes from 3 to 52 LOC
- Total 942 instances

**Nearly all instances perfectly recognized**

- 100% recall for 33 templates; over 80% for remaining 3
- 100% precision for 35 templates; 87% for remaining 1

**Error causes**

- Severe formatting problems, e.g., manual line breaks
- Forgotten ids

# Other Questions

**Amount & kind of template change in refinement**

1% – 25% LOC affected during refinement

81% changes concern optionality (add '?' or component)

**Amount of iterations**

1 instance (11 cases) to 50% of all instances (6 cases)

e.g., 10 out of 20 (2 cases); mostly simple edits, add `?'

**Implication**

Start with few examples, then edit the template based on expert knowledge (e.g., add `?')

# Related Work

## Import to Req Mgmt Tools

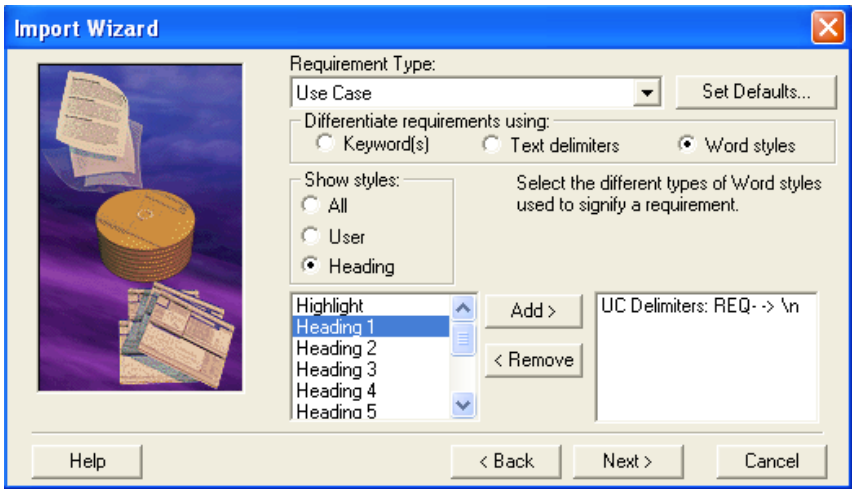
- Tools prescribe document structure
- Manual markup for fine-grained extraction

## Wrapper induction

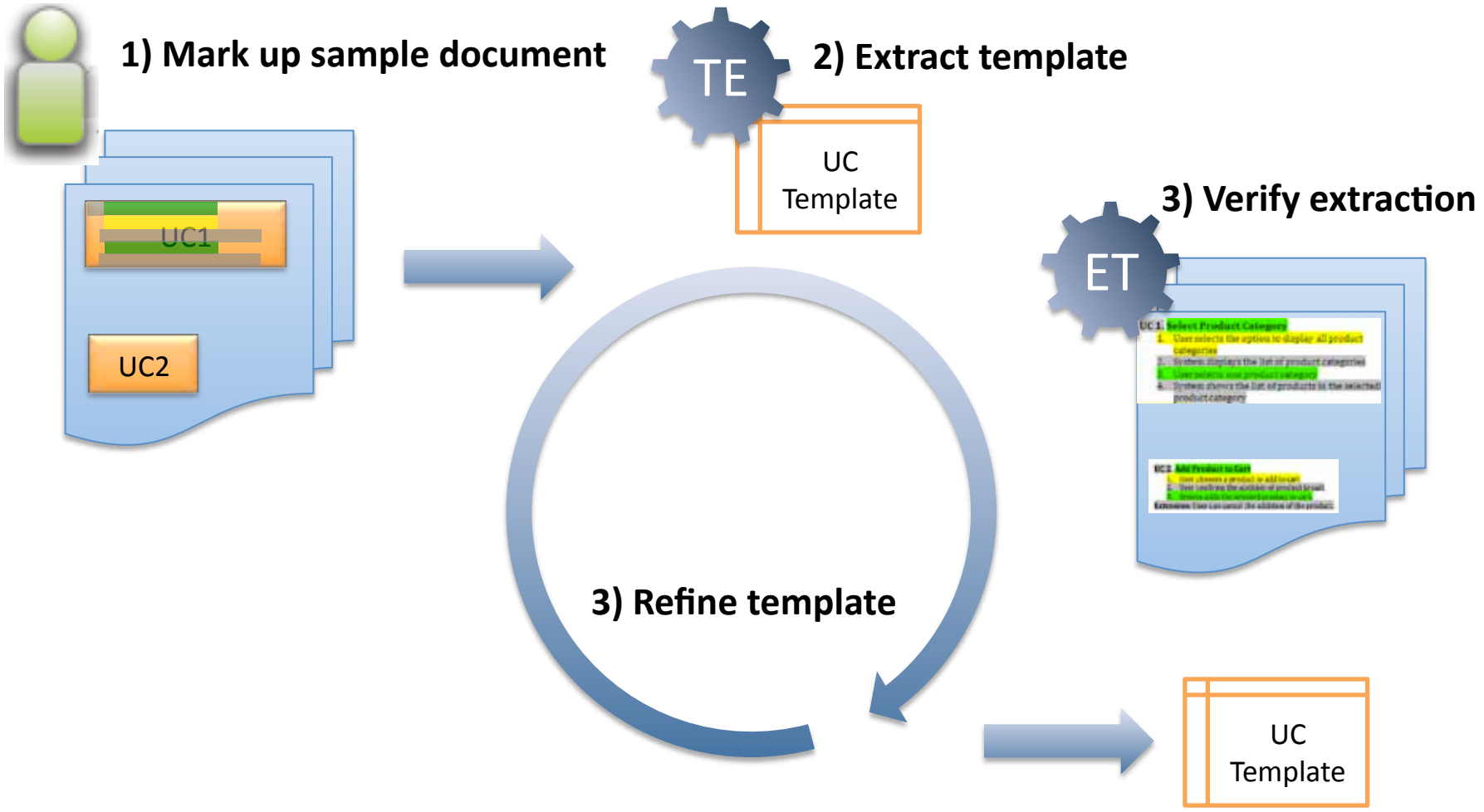
- Machine generated docs (web pages)
- Induced Regex not human readable (no modeling language)

## Natural language processing

- Can benefit from structure-induced semantic tags

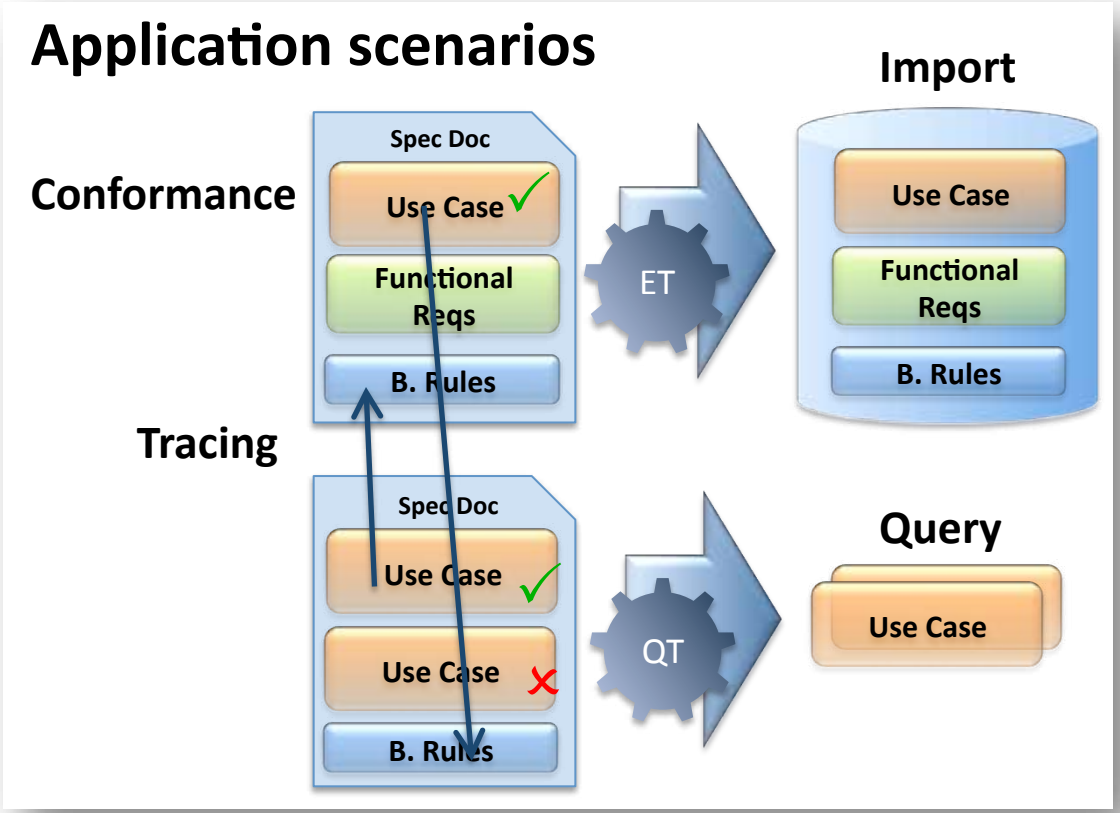


# Future: Template by Example



# Summary

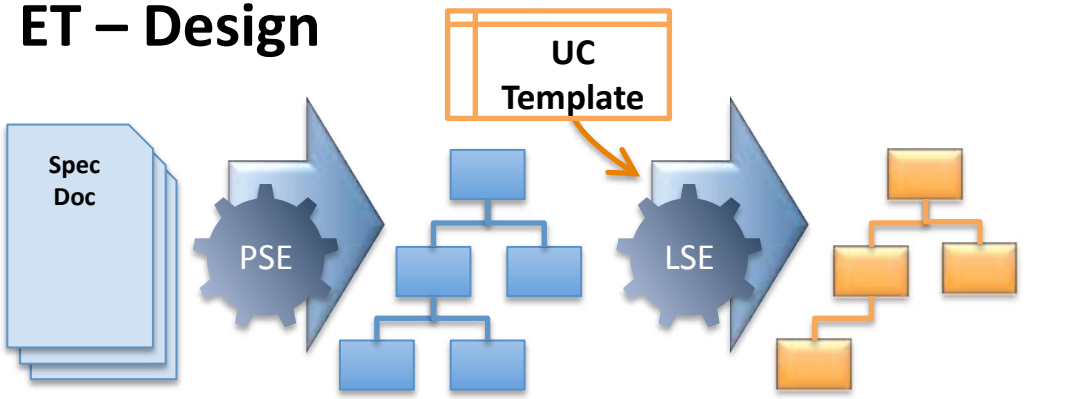
## Application scenarios




## Template development

```
abstract UseCase1 : LogicalStructure
  'SectionMapping
    [sectionTitleStyle = Bold12]

  ID : Attribute
    'SectionTitleMapping
      [sectionTitlePattern="{UC 'NUM'} * ]"

  Name : Attribute
    'SectionTitleMapping
      [sectionTitlePattern=" UC 'NUM' {*} ]"

  Flow : Attribute
    'ListMapping
    FlowItem : Attribute 1..*
      'ParagraphMapping

  Extensions : Attribute ?
    xor Mapping
      'SectionMapping
        [sectionTitleText="Extensions:"]

      'TextBlockMapping
        [identText="Extension"]
        [delimiter=":"]
```

## ET – Design

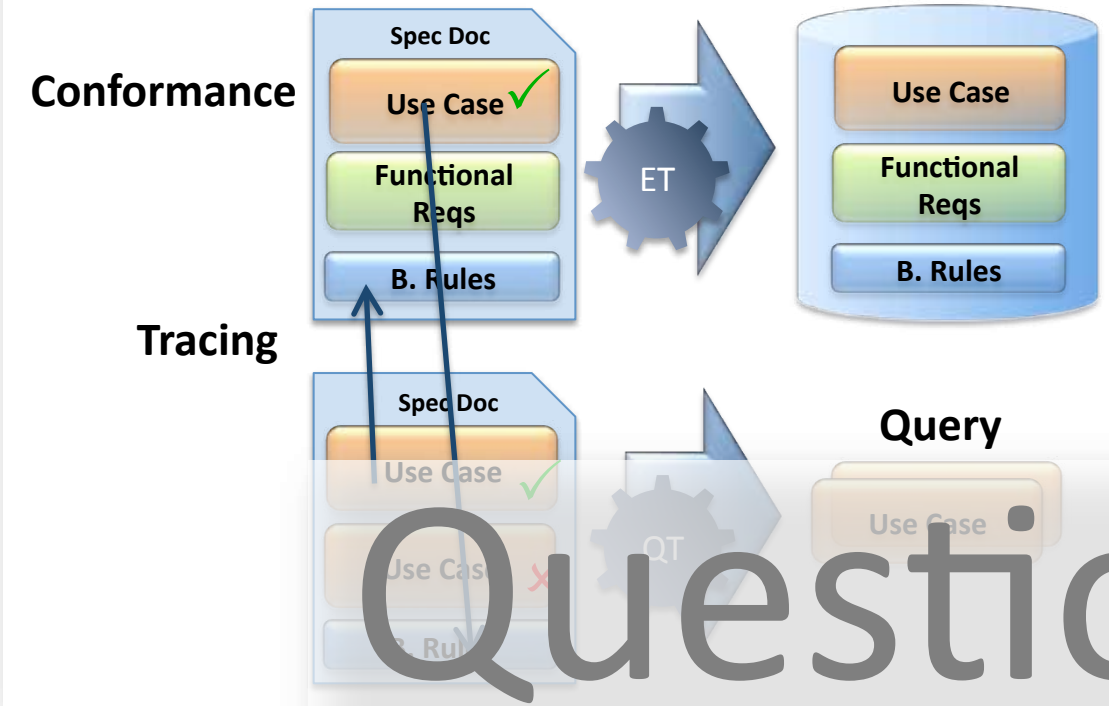



## Evaluation results

43 real-world documents

Nearly all instances perfectly recognized

## Application scenarios

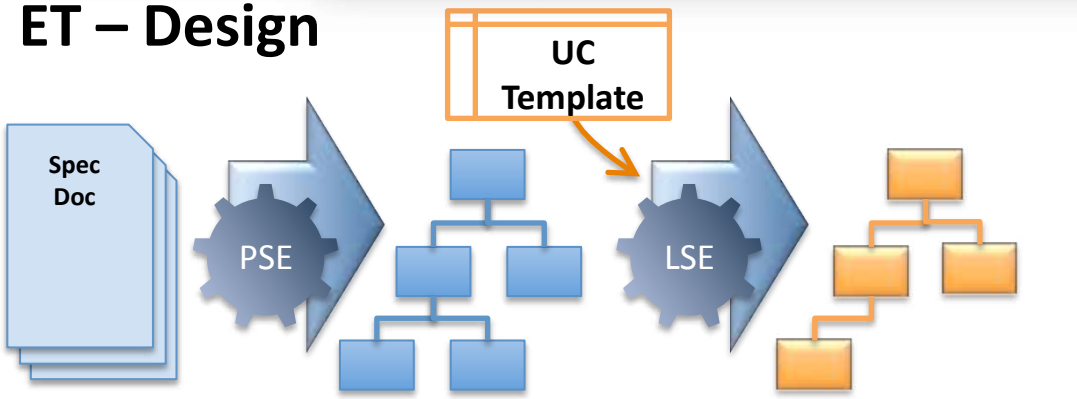



## Template development

```
abstract UseCase1 : LogicalStructure
  'SectionMapping
    [sectionTitleStyle = Bold12]

  ID : Attribute
    'SectionTitleMapping
      [sectionTitlePattern="{UC 'NUM'} * ]"

  Name : Attribute
    'SectionTitleMapping
      [sectionTitlePattern=" UC 'NUM' {*} ]"

  Flow : Attribute
    'ListMapping
    FlowItem : Attribute 1..*
      'ParagraphMapping

  Extensions : Attribute ?
    xor Mapping
      'SectionMapping
        [secti                    t="Extensions:"]
        TextBlockMapp
        [identText                xtension"]
        [delimite                 ":"]
```

Questions?

## ET – Design




## Evaluation results

43 real-world documents

Nearly all instances perfectly recognized