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**

**Spec Doc**

**Heading**

Text text text text text text text
- text text text text text text
- text text text text text text text

<table>
<thead>
<tr>
<th>Text</th>
<th>Text</th>
<th>Text</th>
<th>Text</th>
<th>Text</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>text</td>
<td>Text</td>
<td>Text</td>
<td>text</td>
<td>text</td>
</tr>
<tr>
<td>text</td>
<td>text</td>
<td>text</td>
<td>text</td>
<td>text</td>
<td>text</td>
</tr>
</tbody>
</table>

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
Specification Documents

Physical structures

Section

Table

Paragraph
Specification Documents

Physical structures
- Section
- Table
- Paragraph

Logical structures
- Use Case
- Functional Reqs
- Business Rules

(Logical structures (specification elements))
Recognize and extract specification elements based on physical document structure
ET – Extraction Tool
searches for template instances

Spec Doc

<table>
<thead>
<tr>
<th>Text</th>
<th>Text</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>text</td>
<td>text</td>
</tr>
<tr>
<td>text</td>
<td>text</td>
<td>text</td>
</tr>
<tr>
<td>text</td>
<td>text</td>
<td>text</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Text</th>
<th>Text</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>Text</td>
<td>text</td>
</tr>
<tr>
<td>text</td>
<td>text</td>
<td>text</td>
</tr>
</tbody>
</table>
ET – Extraction Tool
searches for template instances

<table>
<thead>
<tr>
<th>Text</th>
<th>Text</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>text</td>
<td>text</td>
</tr>
<tr>
<td>text</td>
<td>text</td>
<td>text</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Text</th>
<th>Text</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>Text</td>
<td>text</td>
</tr>
<tr>
<td>text</td>
<td>text</td>
<td>text</td>
</tr>
</tbody>
</table>
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

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 text text

<table>
<thead>
<tr>
<th>Text</th>
<th>Text</th>
<th>Text</th>
<th>Text</th>
<th>Text</th>
<th>Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>text</td>
<td>Text</td>
<td>Text</td>
<td>text</td>
<td>text</td>
</tr>
<tr>
<td>text</td>
<td>text</td>
<td>text</td>
<td>text</td>
<td>text</td>
<td>text</td>
</tr>
</tbody>
</table>

text text text text text text text text
text text text text text text text text
text text text text text text text text

door
HP Quality Center
Requisite Pro
...

13
Import to Requirements Mgmt Tools

Spec Doc

Use Case

Functional Reqs

Business Rules

ET

Doors
HP Quality Center
Requisite Pro

...
Import to Requirements Mgmt Tools

Spec Doc  
Use Case  
Functional Reqs  
Business Rules

Use Case  
Functional Reqs  
Business Rules

Doors  
HP Quality Center  
Requisite Pro  
...

15
Structured Query

All use cases with actor = ‘customer’
Template Conformance Checking

Spec Doc

Use Case

Use Case
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

<table>
<thead>
<tr>
<th>Action</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrator selects the</td>
<td>The system shows a list of all documents.</td>
</tr>
<tr>
<td>option to view all documents.</td>
<td></td>
</tr>
<tr>
<td>Admin selects the document to</td>
<td>The system shows a dialogue box to confirm</td>
</tr>
<tr>
<td>delete.</td>
<td>deletion.</td>
</tr>
</tbody>
</table>

Instances of Use Case
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
<table>
<thead>
<tr>
<th>Action</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrator selects the option to view all documents.</td>
<td>The system shows a list of all documents</td>
</tr>
<tr>
<td>Admin selects the document to delete.</td>
<td>The system shows a dialogue box to confirm deletion</td>
</tr>
</tbody>
</table>

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
<table>
<thead>
<tr>
<th>Act</th>
<th>Administrator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-condition</td>
<td>Administrator must be logged in</td>
</tr>
<tr>
<td>Process Description</td>
<td></td>
</tr>
<tr>
<td>Action</td>
<td>Response</td>
</tr>
<tr>
<td>Administrator selects the option to view all documents.</td>
<td>The system shows a list of all documents</td>
</tr>
<tr>
<td>Admin selects the document to delete</td>
<td>The system shows a dialogue box to confirm deletion.</td>
</tr>
</tbody>
</table>

Instances of Use Case  Logical components  Component Identifiers
Variation Types

Logical

Physical

Designed

Accidental
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

<table>
<thead>
<tr>
<th>Actor</th>
<th>User</th>
</tr>
</thead>
</table>

**Main Scenario**
- The user selects to view public documents.
- The system displays the list of public documents.

### Delete Documents

<table>
<thead>
<tr>
<th>Actor</th>
<th>Administrator</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Precondition</strong></td>
<td>Administrator must be logged in</td>
</tr>
</tbody>
</table>

**Process Description**

<table>
<thead>
<tr>
<th>Action</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrator selects the option to view all documents.</td>
<td>The system shows a list of all documents.</td>
</tr>
<tr>
<td>Admin selects the document to delete.</td>
<td>The system shows a dialogue box to confirm deletion.</td>
</tr>
</tbody>
</table>

**Deeper decomposition**

Different methodologies lead to logical variation
Designed Physical 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 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.

Different formatting
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

UC Template

Docs → PSE → LSE

Physical components
Sections, lists, table cells

Logical components
Actor, flow, extensions
ET – Extraction Tool

- Designed variation via template
- Accidental variation via match threshold

UC Template

Docs → PSE → LSE

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

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="."]
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

```plaintext
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
      [indentText="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
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
   [indentText= "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
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 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

```java
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 = ":"]
```
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 the cart
2. User confirms the addition of product to cart
3. System adds the selected product to the 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
## Templates with Tables

### Delete Documents

<table>
<thead>
<tr>
<th>Action</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrator selects the option to view all documents.</td>
<td>The system shows a list of all documents.</td>
</tr>
<tr>
<td>Admin selects the document to delete.</td>
<td>The system shows a dialogue box to confirm deletion.</td>
</tr>
</tbody>
</table>

### Display Public Documents

<table>
<thead>
<tr>
<th>Actor</th>
<th>Main Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>The user selects the option to view public documents. The system displays list of public documents.</td>
</tr>
</tbody>
</table>
Logical Alternatives

**Delete Documents**

<table>
<thead>
<tr>
<th>Action</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrator selects the option to view all documents.</td>
<td>The system shows a list of all documents.</td>
</tr>
<tr>
<td>Admin selects the document to delete.</td>
<td>The system shows a dialogue box to confirm deletion.</td>
</tr>
</tbody>
</table>

**Display Public Documents**

<table>
<thead>
<tr>
<th>Actor</th>
<th>User</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main Scenario</td>
<td></td>
</tr>
<tr>
<td>The user selects the option to view public documents</td>
<td></td>
</tr>
<tr>
<td>The system displays list of public documents</td>
<td></td>
</tr>
</tbody>
</table>

** xor Flow **

```
xor Flow
```

```
abstract UseCase2 : LogicalStructure
`TableMapping`

Name : LogicalComponent
`CellMapping`
[ coliIndex=1
  rowlIndex=1 ]

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

Precondition : LogicalComponent
`HCellBlockMapping`
[ identText="Precondition"]

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

Docs → PSE → LSE → UC Template

**Physical components**
- **Basic**: Paragraph, cell, graphic
- **Composite**: Sections, lists, tables, ...
Physical Structure Extraction

Docs → PSE → LSE

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
  ...

Docs
Template Development

1) Write template manually

2) Verify extraction

3) Refine template
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

1) Mark up sample document

2) Extract template

UC Template

3) Verify extraction

3) Refine template
Summary
Evaluation results

43 real-world documents

Nearly all instances perfectly recognized
Questions?

Evaluation results
43 real-world documents
Nearly all instances perfectly recognized

Application scenarios
Conformance
Use Case
Functional Reqs
B. Rules

Tracing
Spec Doc
Use Case
Functional Reqs
B. Rules

Import
Use Case
Functional Reqs
B. Rules

Query
Use Case

Template development
1. abstract UseCase : LogicalStructure
2. 'SectionMapping
3. [sectionTitleStyle = Bold12]
4. ID : Attribute
5. 'SectionTitleMapping
6. [sectionTitlePattern="{UC \"NUM\"\} *"]
7. Name : Attribute
8. 'SectionTitleMapping
9. [sectionTitlePattern=" UC \"NUM\" \"]
10. Flow : Attribute
11. 'ListMapping
12. FlowItem : Attribute 1..*
13. 'ParagraphMapping
14. Extensions : Attribute ?
15. 'xor Mapping
16. 'SectionMapping
17. [sectionTitlePattern="Extensions:"
18. 'pBlockMapping
19. [identElem='extension']
20. [delimiter='?']