Recovering Traceability Links between a Simple Natural Language Sentence and Source Code Using Domain Ontologies

Takashi Yoshikawa  **Shinpei Hayashi**  Motoshi Saeki
Department of Computer Science
Tokyo Institute of Technology
Japan
Doc-to-code traceability is important
- For reducing maintenance costs
- For software reuse & extension

Focus: recovering sentence-to-code traceability
- In some software products, there's only documents of simple sentences without any detailed descriptions
  e.g. under an agile process
Aim

- To precisely detect **set of methods** related to the **input sentence**

**NL sentence (a set of words)**
Users can draw a plain oval.

**Source code (a set of methods)**
- drawOval()
- getColor()
- setPixel()
- getColorPallette()
- getCanvas()
- DrawPanel()
- OvalTool()
- writeLog()
Problem

- How to precisely get the set?
  - Word similarity: leads to false positives/negatives

Users can **draw** a plain **oval**.

Source code (a set of methods)

- drawOval()
- getCanvas()
- getColor()
- setPixel()
- getColorPallette()
- DrawPanel()
- OvalTool()
- writeLog()

False
positives

False
negatives
Problem

- How to precisely get the set?
  - Word similarity: leads to false positives/negatives
  - Method invocation: leads to false positives

Users can draw a plain oval.
Another criterion required

- To judge whether a method invocation is needed
- Considering the problem domain
Domain Ontology

- Formally representing the knowledge of the target problem domain
  - As relationships between concepts (words)

An ontology for painting tools (excerpt)

A concept “canvas” is a possible target to “draw”.

The “draw” function concerns a “color” concept.
Our Solution

- Choosing method invocations by using domain ontologies

Users can **draw** a plain **oval**.

**NL sentence (a set of words)**

**Source code (a set of methods and their invocations)**
System Overview

**Inputs**

- Sentence
- Source Code
- Domain Ontologies

**Extracting Words**

- Splitting, stemming, removing stop words...

**Call-graph**

- Method invocation analysis

**Prioritizing**

- Call-graph

**Outputs**

- Words in the Sentence
- Words in the Code
- Sentence-related Code fragments

**Ordered**

- Sentence-related Code fragments
Procedure

1. Root selection
2. Traversal
3. Results extraction

Source code
(a set of methods and their invocations)

NL sentence
(a set of words)

\{w_a, w_b\}
Procedure

1. **Root selection**
   - Choose the methods having the words of the input sentence
   - The words become the methods' roles

2. **Traversal**

3. **Results extraction**
Procedure

1. Root selection
2. Traversal
   - Traverse method invocations from the roots iff the invocation satisfies one of the traversal rules
3. Results extraction

NL sentence (a set of words) \{w_a, w_b\}  
Source code (a set of methods and their invocations)
Traversals

- **3 traversal rules**
  1. Sentence-based rule
  2. **Ontology-based rule**
  3. Inheritance rule

- **Role extraction**
  - Words used in the rules are extracted as the callee’s role

**Rule #2 (ontology-based)**

```
getCanvas()
role: {canvas}
```

```
drawOval()
role: {draw, oval}
```

```
draw
```

```
canvas
```

```
getCanvas()
role: {canvas}
```

```
```
Procedure

1. **Root selection**
2. **Traversal**
   - Traverse method invocations from the roots *iff* the invocation satisfies one of the *traversal rules*
   - *Method roles* are also extracted by the rules
3. **Results extraction**
Procedure

1. Root selection
2. Traversal
3. Results extraction
   - The traversed set of methods are the candidates of sentence-related code fragment

NL sentence (a set of words) \{w_a, w_b\}

Source code (a set of methods and their invocations)

\(m_1\) role: \{w_a\}
\(m_2\) role: \{\ldots\}
\(m_3\) role: \{\ldots\}
\(m_4\) role: \{w_b\}
\(m_5\) role: \{\ldots\}
\(m_6\) role: \{\ldots\}
\(m_7\)
\(m_8\)
Case Study

● Evaluation target: JDraw 1.1.5
  - Picked up 7 sentences from JDraw's manual on the Web
  - Prepared an ontology for painting tools
    • including 38 concepts, 45 relationships
  - Prepared control (answer) sets by a expert

● Evaluation Criteria
  - Calculating precision and recall values by comparing the extracted sets with the control sets
## Results

<table>
<thead>
<tr>
<th>Input sentences</th>
<th>Use of ontology</th>
<th>Yes Prec.</th>
<th>Yes Recall</th>
<th>No Prec.</th>
<th>No Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. &quot;plain, filled and gradient filled rectangles&quot;</td>
<td>Yes</td>
<td>0.83</td>
<td>0.94</td>
<td>1.00</td>
<td>0.19</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>0.82</td>
<td>0.98</td>
<td>1.00</td>
<td>0.21</td>
</tr>
<tr>
<td>2. &quot;plain, filled and gradient filled ovals&quot;</td>
<td></td>
<td>1.00</td>
<td>0.35</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>3. &quot;image rotation&quot;</td>
<td></td>
<td>0.22</td>
<td>0.68</td>
<td>1.00</td>
<td>0.58</td>
</tr>
<tr>
<td>4. &quot;image scaling&quot;</td>
<td></td>
<td>0.40</td>
<td>1.00</td>
<td>0.67</td>
<td>1.00</td>
</tr>
<tr>
<td>5. &quot;save JPEGs of configurable quality&quot;</td>
<td></td>
<td>0.74</td>
<td>0.95</td>
<td>0.74</td>
<td>0.95</td>
</tr>
<tr>
<td>6. &quot;colour reduction&quot;</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7. &quot;grayscaleing&quot;</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

- Accurate results by using the ontology
  - precision > 0.7 for 3 cases
  - recall > 0.9 for 4 cases
## Results

<table>
<thead>
<tr>
<th>Input sentences</th>
<th>Use of ontology</th>
<th>Yes Prec.</th>
<th>Yes Recall</th>
<th>No Prec.</th>
<th>No Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. &quot;plain, filled and gradient filled rectangles&quot;</td>
<td></td>
<td>0.83</td>
<td>0.94</td>
<td>1.00</td>
<td>0.19</td>
</tr>
<tr>
<td>2. &quot;plain, filled and gradient filled ovals&quot;</td>
<td></td>
<td>0.82</td>
<td>0.98</td>
<td>1.00</td>
<td>0.21</td>
</tr>
<tr>
<td>3. &quot;image rotation&quot;</td>
<td></td>
<td>1.00</td>
<td>0.35</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>4. &quot;image scaling&quot;</td>
<td></td>
<td>0.22</td>
<td>0.68</td>
<td>1.00</td>
<td>0.58</td>
</tr>
<tr>
<td>5. &quot;save JPEGs of configurable quality&quot;</td>
<td></td>
<td>0.40</td>
<td>1.00</td>
<td>0.67</td>
<td>1.00</td>
</tr>
<tr>
<td>6. &quot;colour reduction&quot;</td>
<td></td>
<td>0.74</td>
<td>0.95</td>
<td>0.74</td>
<td>0.95</td>
</tr>
<tr>
<td>7. &quot;grayscaling&quot;</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

- **Improvement by using the ontology**
  - Improved recall for 1\textsuperscript{st} and 2\textsuperscript{nd} cases
  - Detected traceability for 3\textsuperscript{rd} case
- **Bad results (degradation, no effect) also occurred**
Conclusion

Domain ontologies give us valuable guides for traceability-recovering and feature-location.

● **Summary:**
  - Proposed a technique to find a set of methods related to the given NL sentence by using domain ontologies
  - Showed the feasibility of our approach with a case study of JDraw

● **Future work**
  - Automated construction of domain ontologies
  - Case study++