Patl: Safe Program Transformation between APIs with Many-to-Many Mappings

Yingfei Xiong
Peking University
API change is common!

- New APIs are released to replace older ones.
- Migrate a program to another platform.
- Discontinued API support.

Switch the use of old APIs to new ones!
Compatible API change

Client Program

API 1

compatible

API 2

Client Program

API 2
Incompatible API change

Client Program

API 1

incompatible

API 2

Client Program

API 2
Solution: Adapting Client Program

Adaptation

Client Program

API 1

incompatible

API 2

Client Program’

API 2

Client Program’
Adapting Client Program: Not easy

Client Program

Client Program'

Adaptation

No bug

Still readable
Adapting Client Program

- API users view:
  - A tool to automatically adapt the client program.
  - Adapted programs should be correct and readable.

- API developers view:
  - How to develop such transformation tools easily?

Transformation languages support:
- Specify transformation with rules.
Existing language support

• General purpose transformation languages:
  • Stratego [1], TXL [2]
  • Pro: expressive
  • Con: not specialized for API adaptation task, low-level

• API adaptation domain specific languages:
  • SWIN [3], Twinning [4]
  • Pro: specialized, ease-to-use
  • Con: captures only one-to-many mappings, less expressive

How can we help support many-to-many transformations?
Many-to-Many (M-to-M) transformation

- Definition: Match a sequence of statements in the source program, substitute them with another sequence of statements.
- E.g. (Swing to SWT)

```java
//rule rButton
(jb: JButton->Button,
  parent: JPanel->Composite) {
  - jb = new JButton();
  - parent.add(jb);
  + jb = new Button(parent, SWT.PUSH);
}
```

- Basic transformation: match and substitute.
M-to-M transformation

//rule rButton
(jb: JButton->Button, 
parent: JPanel->Composite) {
- jb = new JButton();
- parent.add(jb);
+ jb = new Button(parent, SWT.PUSH);
}

//Case 1:
jb = new JButton();
if(parent != null) {
    parent.add(jb);
}

//Case 2:
jb = new JButton();
s = jb.getUIClassID();
parent = new JPanel();
parent.add(jb);

//Case 3:
jb = new JButton();
defaultButton = jb;
parent.set(defaultButton);

Challenge:
The source sequence can appear in many different forms in the client program.

• Match them with the rules. 😊 (SmPL [5])
• Transform them safely. ☹️ → 😊 (Our approach)
Insight: guided-normalization

Transformation rule writer only need to consider basic transformation!!
Guided-normalization

- Normalize the source program
  - Semantics-preserving.
  - Touch less unrelated statements.
  - Matched statements appear consecutively after normalization.

- Preliminary: Program analysis
  - Analyze dependency and alias relations in the program to ensure normalization will not go wrong.
Our transformation pipeline: Patl

\[ \Pi, p \xrightarrow{\text{Match}} M \xrightarrow{\text{Guided normalization}} p' \xrightarrow{M} p'' \]

\(\Pi\): transformation rules.

\(p\): client program to be transformed.

\(M\): match instances

\(p'\): normalized program.

\(p''\): transformed program with new API use.

1 //Case 1:
2    jb = new JButton();
3 if(parent != null) {
4        parent.add(jb);
5    }

1 //Case 1':
2 if(parent != null) {
3     jb = new JButton();
4     parent.add(jb);
5 }

1 //Case 1':
2 if(parent != null) {
3     jb = new Button(parent, SWT.PUSH);
4 }

1 //Case 1:
2    jb = new JButton();
3 if(parent != null) {
4        parent.add(jb);
5    }

1 //Case 1':
2 if(parent != null) {
3     jb = new JButton();
4     parent.add(jb);
5 }

1 //Case 1':
2 if(parent != null) {
3     jb = new JButton();
4     parent.add(jb);
5 }

1 //Case 1':
2 if(parent != null) {
3     jb = new Button(parent, SWT.PUSH);
4 }

1 //Case 1':
2 if(parent != null) {
3     jb = new JButton();
4     parent.add(jb);
5 }

1 //Case 1':
2 if(parent != null) {
3     jb = new JButton();
4     parent.add(jb);
5 }

1 //Case 1':
2 if(parent != null) {
3     jb = new Button(parent, SWT.PUSH);
4 }
Guided-Normalization

\[ \Pi, p \longrightarrow M \]

- **Guided-shift**
  - Make statements matched by a rule appear in the same block.
- **Guided-rename**
  - Make aliases in these statements have the same name.
- **Guided-reorder**
  - Make matched statements appear consecutively.
Guided-normalization: example

```java
//rule rButton
(jb: JButton->Button,
 parent: JPanel->Composite) {
- jb = new JButton();
- parent.add(jb);
+ jb = new Button(parent, SWT.PUSH);
}
```

```java
btn = new JButton();
btn.setAlignmentX(alX);
System.out.print(alX);
b = panel != null;
if (b) {
    panel.add(btn);
} else {
defaultBtn = btn;
defaultPnl.add(defaultBtn);
}

normalize

1 System.out.print(alX);
2 b = panel != null;
3 if (b) {
4    btn = new JButton();
5    panel.add(btn);
6    btn.setAlignmentX(alX);
7 } else {
8    x = new JButton();
9    defaultPnl.add(x);
10    btn = x;
11    defaultBtn = btn;
12    btn.setAlignmentX(alX);
13 }
```
Guided-shift

- If matched statements appear in different blocks, shift them into basic blocks.

```
1 btn = new JButton();
2 btn.setAlignmentX(alX);
3 System.out.println(alX);
4 b = panel != null;
5 if (b) {
  6   panel.add(btn);
7 } else {
8   defaultBtn = btn;
9   defaultPnl.add(defaultBtn);
10 }
```

```
1 System.out.println(alX);
2 b = panel != null;
3 if (b) {
  4   btn = new JButton();
  5   btn.setAlignmentX(alX);
  6   panel.add(btn);
7 } else {
  8   btn = new JButton();
  9   btn.setAlignmentX(alX);
10   defaultBtn = btn;
11   defaultPnl.add(defaultBtn);
12 }
```
Guided-Rename

- Aliases in matched statements are renamed to have the same names.

```
1 System.out.println(alX);
2 b = panel != null;
3 if (b) {
   4 btn = new JButton();
   5 btn.setAlignmentX(alX);
   6 panel.add(btn);
} else {
   8 btn = new JButton();
   9 btn.setAlignmentX(alX);
   10 defaultBtn = btn;
   11 defaultPnl.add(defaultBtn);
}
```

GuidedRename

```
1 System.out.println(alX);
2 b = panel != null;
3 if (b) {
   4 btn = new JButton();
   5 btn.setAlignmentX(alX);
   6 panel.add(btn);
} else {
   8 x = new JButton();
   9 btn = x;
   10 btn.setAlignmentX(alX);
   11 defaultBtn = btn;
   12 defaultPnl.add(x);
}
```
Guided-reorder

- Reorder matched statements so that they appear consecutively.

```
1 System.out.print(alX);
2 b = panel != null;
3 if (b) {
4    btn = new JButton();
5    btn.setAlignmentX(alX);
6    panel.add(btn);
7  } else {
8    x = new JButton();
9    btn = x;
10   btn.setAlignmentX(alX);
11   defaultBtn = btn;
12   defaultPnl.add(x);
13 }
```

Rename

```
1 System.out.print(alX);
2 b = panel != null;
3 if (b) {
4    btn = new JButton();
5    panel.add(btn);
6    btn.setAlignmentX(alX);
7  } else {
8    x = new JButton();
9    defaultPnl.add(x);
10   btn = x;
11   defaultBtn = btn;
12   btn.setAlignmentX(alX);
13 }
```
Guided-normalization: Safety

• How to ensure normalization is semantics-preserving?

• Semantics-preserving transformation primitives:
  • Primitive shift
  • Primitive swap
  • Primitive left-value renaming
  • Primitive right-value renaming
  • Fresh-variable introducing

• Safety: guided-normalization can be decomposed into transformation primitives. (Proof in the paper!)
Warnings in transformation

- Guided-normalization is not always applicable:
  - Dependency may be violated.

\[
\begin{align*}
x &= y.a(); \\
\text{if } (x) \{} \\
  &\quad y.b(x); \\
\{} \\

x &= y.a(); \\
\text{Send}(x, y); \\
  &\quad y.b(x); \\
\end{align*}
\]

- Our system will generate warnings in such cases rather than silently making mistakes.
Evaluation

• Q1: How important is guided-normalization in transforming programs between APIs?
• Q2: How many cases cannot be handled by our approach?
• Q3: How many warnings will be generated in real world cases?
Evaluation: set-up

• Three real-world cases:
  • Jdom $\rightarrow$ Dom4J
  • Google calendar v2 $\rightarrow$ v3
  • Swing $\rightarrow$ SWT

• Six open source projects using these APIs.

<table>
<thead>
<tr>
<th>Client</th>
<th>KLOC</th>
<th>Classes</th>
<th>Methods</th>
<th>Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>husacct</td>
<td>195.6</td>
<td>1187</td>
<td>5977</td>
<td>Jdom/Dom4j</td>
</tr>
<tr>
<td>serenoa</td>
<td>12.2</td>
<td>52</td>
<td>523</td>
<td>Jdom/Dom4j</td>
</tr>
<tr>
<td>openfuxml</td>
<td>112.5</td>
<td>727</td>
<td>4098</td>
<td>Jdom/Dom4j</td>
</tr>
<tr>
<td>clinicaweb</td>
<td>3.9</td>
<td>74</td>
<td>213</td>
<td>Calendar</td>
</tr>
<tr>
<td>blasd</td>
<td>9.7</td>
<td>199</td>
<td>729</td>
<td>Calendar</td>
</tr>
<tr>
<td>goofs</td>
<td>8.6</td>
<td>78</td>
<td>643</td>
<td>Calendar</td>
</tr>
<tr>
<td>evochamber</td>
<td>12.8</td>
<td>132</td>
<td>868</td>
<td>Swing/SWT</td>
</tr>
<tr>
<td>swingheat</td>
<td>2.3</td>
<td>30</td>
<td>186</td>
<td>Swing/SWT</td>
</tr>
<tr>
<td>marble</td>
<td>1.6</td>
<td>10</td>
<td>56</td>
<td>Swing/SWT</td>
</tr>
</tbody>
</table>

| Total    | 359.2 | 2489    | 13293   | –           |
### Evaluation: result

<table>
<thead>
<tr>
<th>Client</th>
<th>CF</th>
<th>CL</th>
<th>W</th>
<th>U</th>
<th>I</th>
<th>MM</th>
<th>GN</th>
</tr>
</thead>
<tbody>
<tr>
<td>husacct</td>
<td>42</td>
<td>852(100%)</td>
<td>0(0%)</td>
<td>0(0%)</td>
<td>0</td>
<td>0(0%)</td>
<td>0(0%)</td>
</tr>
<tr>
<td>serenoa</td>
<td>8</td>
<td>273(98.9%)</td>
<td>0(0%)</td>
<td>3(1.1%)</td>
<td>0</td>
<td>9(3.3%)</td>
<td>0(0%)</td>
</tr>
<tr>
<td>openfuxml</td>
<td>72</td>
<td>983(94.8%)</td>
<td>0(0%)</td>
<td>54(5.2%)</td>
<td>15</td>
<td>2(0.2%)</td>
<td>0(0%)</td>
</tr>
<tr>
<td>clinicaweb</td>
<td>5</td>
<td>81(100%)</td>
<td>0(0%)</td>
<td>0(0%)</td>
<td>8</td>
<td>34(42%)</td>
<td>0(0%)</td>
</tr>
<tr>
<td>blasd</td>
<td>5</td>
<td>26(63.4%)</td>
<td>8(19.5%)</td>
<td>7(17.1%)</td>
<td>0</td>
<td>13(50%)</td>
<td>2(15.4%)</td>
</tr>
<tr>
<td>goofs</td>
<td>13</td>
<td>100(80.0%)</td>
<td>12(9.6%)</td>
<td>13(10.4%)</td>
<td>27</td>
<td>27(27%)</td>
<td>0(0%)</td>
</tr>
<tr>
<td>evochamber</td>
<td>9</td>
<td>587(98.3%)</td>
<td>10(1.7%)</td>
<td>0(0%)</td>
<td>0</td>
<td>330(56.2%)</td>
<td>109(33.0%)</td>
</tr>
<tr>
<td>swingheat</td>
<td>21</td>
<td>653(100%)</td>
<td>0(0%)</td>
<td>0(0%)</td>
<td>0</td>
<td>461(70.6%)</td>
<td>394(85.5%)</td>
</tr>
<tr>
<td>marble</td>
<td>6</td>
<td>488(98.6%)</td>
<td>0(0%)</td>
<td>7(1.4%)</td>
<td>0</td>
<td>240(49.2%)</td>
<td>220(91.7%)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>181</td>
<td>4043(97.3%)</td>
<td>30(0.7%)</td>
<td>84(2.0%)</td>
<td>50</td>
<td>1116(27.6%)</td>
<td>725(65.0%)</td>
</tr>
</tbody>
</table>

CF = number of changed files, CL = number of changed lines, percentages in CL = CL / (CL+W+U), W = the number of lines of code that have warnings, percentages in W = W / (CL+W+U), U = number of lines that PATL cannot transform, percentages in U = U / (CL+W+U), I = number of lines impossible to transform, MM = number of lines that are involved in many-to-many mappings, percentages in MM = MM / CL, GN = number of lines that require guided normalization, percentages in GN = GN / MM.
Evaluation: result

Transformation Rules

<table>
<thead>
<tr>
<th>Transformation</th>
<th>Rules</th>
<th>Classes</th>
<th>Methods</th>
<th>M-to-m</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jdom/Dom4j</td>
<td>84</td>
<td>12</td>
<td>77</td>
<td>12(14.3%)</td>
</tr>
<tr>
<td>Calendar</td>
<td>42</td>
<td>14</td>
<td>45</td>
<td>21(50.0%)</td>
</tr>
<tr>
<td>Swing/SWT</td>
<td>110</td>
<td>40</td>
<td>82</td>
<td>54(49.1%)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>236</td>
<td>66</td>
<td>204</td>
<td>87(36.9%)</td>
</tr>
</tbody>
</table>

Empirically, Patl is ease-to-use! 😊
An example not handled by Patl

```
XMLOutputter out = new XMLOutputter();
for (Element e : rulesElements)
    rulesToRegister += out.outputString(e);
```

```
StringWriter sw = new StringWriter();
XMLWriter out = new XMLWriter(sw);
for (Element e : rulesElements)
    out.write(e);
    rulesToRegister += sw.toString();
```
Limitations

- Solves only statement level transformation, not class level transformation.
  - E.g. Inheritance from an old API class.
- Does not model synchronization in transformation.
  - E.g. A method may change from synchronized to unsynchronized.
Conclusion

• Guided-normalization helps enhance transformation language support to solve M-to-M transformation programs in API adaptation.

• Guided-normalization:
  • Safe: semantics-preserving.
  • Help ease transformation tool developing.
References


