ECCO: Extraction and Composition for Clone-and-Own

Lukas Linsbauer
Stefan Fischer
Roberto Erick Lopez-Herrejon
Alexander Egyed
Motivation

- Companies often build custom-tailored software
- If successful, they build products for other customers
  - Clone-and-Own
- End up with similar software products
  - Also e.g. Testing, System Engineering, ...
Clone-and-Own

- a.k.a. Copy and Paste
- Steps:
  - Extracting reusable Code (copy)
  - Composing reusable Code (paste)
  - Completing Products
    - Adding missing Code
    - Modify or Remove Code
Drawing Application

- Line
- Rect
- Color
# Drawing Application

<table>
<thead>
<tr>
<th>Product 1 (P1)</th>
<th>Product 2 (P2)</th>
<th>Product 3 (P3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>Base</td>
<td>Base</td>
</tr>
<tr>
<td>Line</td>
<td>Line</td>
<td>Line</td>
</tr>
<tr>
<td>Wipe</td>
<td>Color</td>
<td>Rect</td>
</tr>
</tbody>
</table>

```java
class Canvas {
    List<Line> lines;
    void wipe()
    {
        this.lines.clear();
    } ...
}
class Line {
    Line(Point start){
        ...
    }
    ...
}
class Main extends JFrame{
    initContentPane{
        toolPanel.add(lineButton);
        toolPanel.add(wipeButton);
    }
    ...
}
```

```java
class Canvas {
    List<Line> lines;
    void setColor(String c){
        ...
    }
    ...
}
class Line {
    Line(Color c, Point start){
        ...
    }
    ...
}
class Main extends JFrame{
    initContentPane{
        toolPanel.add(lineButton);
        toolPanel.add(lineColorButton);
    }
    ...
}
```

```java
class Canvas {
    List<Rect> rects;
    void setColor(String c){
        ...
    }
    ...
}
class Rect {
    Rect(Color c, int x, int y){
        ...
    }
    ...
}
class Main extends JFrame{
    initContentPane{
        toolPanel.add(rectButton);
        toolPanel.add(colorsPanel);
    }
    ...
}
```
Clone-and-Own Example

- P4 (Base, Line, Rect, Wipe)

```java
class Canvas {
    List<Line> lines;

    void wipe() {
        this.lines.clear();
        this rects . clear();
    }
}

class Line {
    Line (Point start) {
        ...
    }
    ...
}
```

- Add Code
- Remove Code
- Order Code

```java
class Main extends JFrame {
    initContentPane() {
        toolPanel.add(lineButton);
        toolPanel.add(wipeButton);
    }
}
```

Complete Code: Missing Feature Interaction

Complete Code: Order Statements
Clone-and-Own

- **Benefits**
  - Pay as you go: no upfront investment, like for developing a Software Product Line (SPL)
  - Intuitive: easier to build individual products without having to consider variability
  - Unforeseeable future: Company might not know which features will be required in the future

- **Problems**
  - Lack systematic methodology and tool support
  - Extraction / Composition get significantly more complex with each new product
Goals

- Automate the Clone-and-Own approach
  - Extraction & Composition
- Support for different kinds of artifacts
  - Source code
  - Models
  - Tests
  - ...
ECCO (Extraction & Composition for Clone-and-Own)
Extraction

- Traces: associations between features or feature interactions and artifacts
- Order: Sequence of artifacts
- Dependencies: relationships between traces
Extraction – Big Picture

<table>
<thead>
<tr>
<th>Product 1 (P1)</th>
<th>Product 2 (P2)</th>
<th>Product 3 (P3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>Base</td>
<td>Base</td>
</tr>
<tr>
<td>Line</td>
<td>Line</td>
<td>Line</td>
</tr>
<tr>
<td>Wipe</td>
<td>Color</td>
<td>Rect</td>
</tr>
</tbody>
</table>

```java
class Canvas {
    List<Line> lines;
    void wipe()
    {
        this.lines.clear();
    }
}
class Line {
    Line(Point start){
        ...
    }
    ...
}
class Main extends JFrame{
    initContentPane()
    {
        toolPanel.add(lineButton);
        toolPanel.add(wipeButton);
    }
    ...
}
```

```java
class Canvas {
    List<Line> lines;
    void setColor(String c){
        ...
    }
    ...
}
class Line {
    Line(Color c, Point start){
        ...
    }
    ...
}
class Main extends JFrame{
    initContentPane()
    {
        toolPanel.add(lineButton);
        toolPanel.add(colorsPanel);
    }
    ...
}
```

```java
class Canvas {
    List<Rect> rects;
    void setColor(String c){
        ...
    }
    ...
}
class Line {
    Line(Color c, Point start){
        ...
    }
    ...
}
class Rect {
    Rect(Color c, int x, int y){
        ...
    }
    ...
}
class Main extends JFrame{
    initContentPane()
    {
        toolPanel.add(lineButton);
        toolPanel.add(rectButton);
        toolPanel.add(colorsPanel);
    }
    ...
}
```
Extraction – Big Picture

<table>
<thead>
<tr>
<th>Product 1 (P1)</th>
<th>Product 2 (P2)</th>
<th>Product 3 (P3)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Extraction – Big Picture

<table>
<thead>
<tr>
<th>Product 1 (P1)</th>
<th>Product 2 (P2)</th>
<th>Product 3 (P3)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Diagram showing the process of extraction with labels and arrows indicating the flow of information.](image-url)
Extraction – Feature Overlaps

- Base
- Line
- Rect
- Color

- Base
- Line
- Wipe

- Base
- Line
- Color

- P1
- P2
- P3
Extraction – Artifact Overlaps

Introduction

Example

ECCO

Evaluation
Extraction

Feature Overlaps:

Artifact Overlaps:
Extraction

Feature Overlaps:

- Rect
- Base/Rect
- Color/Rect
- ...
- Color/Rect/¬Wipe
- ...
- Wipe/ Rect/¬Color
- ...
- Wipe/ Rect/
- Base/Rect
- Color/Rect
- ...
- Base/Wipe
- Line/Wipe
- Base/Line/Wipe
- Wipe/
- ~Color
- ...

Artifact Overlaps:

- Canvas
- rects
- lines
- wiCol()
Composition

- **Input**: extracted information, set of selected features
- **Output**: composed product from artifacts in selected traces, hints for completion
Composition – P4

- Selected Features:
  - Base
  - Line
  - Rect
  - Wipe

- ECCO provides the following Hints:
  - Missing Features / Feature Interactions
  - Surplus Features / Feature Interactions
  - Unresolved References
  - Alternative Orderings
Completion – P4(Base, Line, Rect, Wipe)

```java
class Canvas {
    List<Line> lines;
    List<Rect> rects;
    void wipe() {
        this.lines.clear();
    }
}
```

```java
class Line {
    Line(Point start) {
        ...
    }
    ...
}
```

```java
class Rect {
    Rect(Color c, int x, int y) {
        ...
    }
    ...
}
```

```java
class Main extends JFrame{
    initContentPane() {
        toolPanel.add(lineButton);
        toolPanel.add(rectButton);
        toolPanel.add(wipeButton);
    }
}
```

Only lines are removed

Still some color code

Alternative orders

Composed Incomplete Product

- Features, Artifacts, Hints, ...

Completion

Complete Product

- Features, Artifacts
Evaluation – Case Studies

<table>
<thead>
<tr>
<th>Case-Study</th>
<th>#F</th>
<th>#P</th>
<th>LoC</th>
<th>#Art</th>
</tr>
</thead>
<tbody>
<tr>
<td>Draw</td>
<td>5</td>
<td>12</td>
<td>287 - 473</td>
<td>491</td>
</tr>
<tr>
<td>ZipMe</td>
<td>7</td>
<td>32</td>
<td>5K - 6.2K</td>
<td>5.2K+</td>
</tr>
<tr>
<td>VOD</td>
<td>11</td>
<td>32</td>
<td>4.7K - 5.2K</td>
<td>5.5K+</td>
</tr>
<tr>
<td>GameOfLife</td>
<td>15</td>
<td>65</td>
<td>874 - 1.9K</td>
<td>1.3K+</td>
</tr>
<tr>
<td>ArgoUML</td>
<td>11</td>
<td>256</td>
<td>264K - 344K</td>
<td>192K+</td>
</tr>
<tr>
<td>ModelAnalyzer</td>
<td>13</td>
<td>5</td>
<td>35K - 59K</td>
<td>94K+</td>
</tr>
</tbody>
</table>

#F: Number of Features  
#P: Number of Products  
LoC: Range of Lines of Code  
#Art: Number of Distinct Artifacts
Evaluation – Scheme
Evaluation - Recall

\[
Recall[\%] = \frac{TP}{TP + FN} = \frac{|commonArtifacts|}{|originalArtifacts|}
\]

- Complete product
- Composed product

Draw, ZipMe, VOD, GameOfLife, ArgoUML, ModelAnalyzer
Evaluation - Precision

![Graph showing precision calculation formula and Venn diagram for complete and composed products.]

\[
\text{Precision} = \frac{TP}{TP + FP} = \frac{|\text{commonArtifacts}|}{|\text{composedArtifacts}|}
\]
Summary

- 10 products in the product portfolio already lead to an average precision and recall of higher than 95% for newly composed products (for our case studies).

- \textit{ECCO} guidance (Hints) can help narrowing down the locations that have to be considered during the completion
Thank you!

Thank you for your attention!

Contacts:

lukas.linsbauer@jku.at

stefan.fischer@jku.at
Want to learn more?

- Lukas Linsbauer, Roberto E. Lopez-Herrejon, Alexander Egyed: Recovering traceability between features and code in product variants. SPLC 2013
- Stefan Fischer, Lukas Linsbauer, Roberto Erick Lopez-Herrejon, Alexander Egyed: Enhancing Clone-and-Own with Systematic Reuse for Developing Software Variants. ICSME 2014

Demonstration Video:
- [https://www.youtube.com/watch?v=N6gPekuxU6o](https://www.youtube.com/watch?v=N6gPekuxU6o)