



# Source code templates in Cha-Qeko/X

The screenshot displays the Cha-Qeko/X IDE interface. At the top, a source code editor shows a method signature: `[public void acceptVisitor(ComponentVisitor v) ...]@[invoked-by ?invocation]`. Below the editor is a table with the following data:

| Element                     | Textual Representation       | Element Kind      | Description |
|-----------------------------|------------------------------|-------------------|-------------|
| ▶ public void acceptVisitor | [public void acceptVisitor(C | MethodDeclaration |             |
| ▶ comp.acceptVisitor(v)     | [...acceptVisitor(...)]@[equ | MethodInvocation  |             |

To the right of the editor is a refinement panel with a green checkmark and the following options:

- Generalize method invocations.
- ▼ Refinement
  - Add directive invoked-by.
  - Add directive overrides.
  - Add directive refers-to.

Below the refinement panel, it states: "Requires matches to be invoked by the binding variable." A table below that shows:

| Operand                 | Value             |
|-------------------------|-------------------|
| Subject                 | [public void acce |
| Meta-variable (e.g. ?v) |                   |

At the bottom, the "Ekeko Query Results" panel is visible, showing "Query Variables" and "Query Stats". The "Query Stats" section includes a table with columns: "?MethodInvocator", "?invocation", and "?MethodDeclaratic".

| ?MethodInvocator    | ?invocation         | ?MethodDeclaratic |
|---------------------|---------------------|-------------------|
| comp.acceptVis...   | comp.acceptVis...   | ▲ Composite.a...  |
| cs.acceptVisitor... | cs.acceptVisitor... | ▲ Composite.a...  |
| comp.acceptVis...   | comp.acceptVis...   | ▲ Component....   |
| comp.acceptVis...   | comp.acceptVis...   | ▲ MethodDecl...   |

# Source code templates in Cha-Qeko/X

The screenshot displays the Cha-Qeko/X IDE interface. At the top, a window titled 'invokedby.ekt' shows a source code template: `[public void acceptVisitor(ComponentVisitor v) ...]@[invoked-by ?invocation]`. This code is highlighted with a red box and labeled 'Template'. Below the code editor is a table with the following structure:

| Element                     | Textual Representation       | Element Kind      | Description |
|-----------------------------|------------------------------|-------------------|-------------|
| ▶ public void acceptVisitor | [public void acceptVisitor(C | MethodDeclaration |             |
| ▶ comp.acceptVisitor(v)     | [...acceptVisitor(...)]@[equ | MethodInvocation  |             |

On the right side, a panel shows a green checkmark and a list of refinement options: 'Generalize method invocations.', 'Add directive invoked-by.', 'Add directive overrides.', and 'Add directive refers-to.'. Below this, it states 'Requires matches to be invoked by the binding variable.' and provides a table of operands and values:

| Operand                 | Value             |
|-------------------------|-------------------|
| Subject                 | [public void acce |
| Meta-variable (e.g. ?v) |                   |

At the bottom, the 'Ekeko Query Results' panel is visible, showing 'Query Variables' and 'Query Stats'. The 'Query Stats' section includes a table with columns for '?MethodInvocator', '?invocation', and '?MethodDeclaratic'. The table contains several rows of query results, including entries for 'comp.acceptVis...', 'cs.acceptVisitor...', and 'comp.acceptVis...'.

# Source code templates in Cha-Qeko/X

The screenshot displays the Cha-Qeko/X IDE interface. At the top, a window titled 'invokedby.ekt' shows a source code template: `[public void acceptVisitor(ComponentVisitor v) ...]@[invoked-by ?invocation]`. This template is highlighted with a red box and labeled 'Template'. Below the code is a table with the following structure:

| Element                     | Textual Representation       | Element Kind      | Description |
|-----------------------------|------------------------------|-------------------|-------------|
| ▶ public void acceptVisitor | [public void acceptVisitor(C | MethodDeclaration |             |
| ▶ comp.acceptVisitor(v)     | [...acceptVisitor(...)]@[equ | MethodInvocation  |             |

To the right of the code editor, a 'Refinement' panel is visible, containing a green checkmark and the text: 'Generalize method invocations. Refinement: Add directive invoked-by, Add directive overrides, Add directive refers-to. Requires matches to be invoked by the binding variable.' Below this, a table shows 'Operand' and 'Value' for 'Subject' and 'Meta-variable (e.g. ?v)'. The 'Subject' value is '[public void acce' and the 'Meta-variable' value is '?v'.

At the bottom, the 'Ekeko Query Results' panel is shown. It includes a 'Mark Results' button and a 'Query Stats' section. Below these, a table displays query results, which are highlighted with a red box and labeled 'Matches':

| ?MethodInvocatio    | ?invocation         | ?MethodDeclaratic |
|---------------------|---------------------|-------------------|
| comp.acceptVis...   | comp.acceptVis...   | ▲ Composite.a...  |
| cs.acceptVisitor... | cs.acceptVisitor... | ▲ Composite.a...  |
| comp.acceptVis...   | comp.acceptVis...   | ▲ Component....   |
| comp.acceptVis...   | comp.acceptVis...   | ▲ MethodDecl...   |

# Source code templates in Cha-Qeko/X

The screenshot displays the Cha-Qeko/X IDE interface. At the top, a window titled 'invokedby.ekt' shows a source code template: `[public void acceptVisitor(ComponentVisitor v) ...]@[invoked-by ?invocation]`. Below the code is a table with columns: Element, Textual Representation, Element Kind, and Description. The table lists two elements: a MethodDeclaration and a MethodInvocation. To the right, an 'Operator list' panel shows a green checkmark and a list of operators: 'Generalize method invocations.', 'Add directive invoked-by.', 'Add directive overrides.', and 'Add directive refers-to.'. Below the operator list, a table shows 'Operand' and 'Value' for the selected operator. The bottom section of the IDE shows 'Query Variables' and 'Query Stats' panels. The 'Query Stats' panel displays a table of matches, with columns for '?MethodInvocation', '?invocation', and '?MethodDeclaration'. The matches include entries for 'comp.acceptVis...', 'cs.acceptVisitor...', and 'comp.acceptVis...'.

**Template**

```
[public void acceptVisitor(ComponentVisitor v) ...]@[invoked-by ?invocation]
```

| Element                   | Textual Representation       | Element Kind      | Description |
|---------------------------|------------------------------|-------------------|-------------|
| public void acceptVisitor | [public void acceptVisitor(C | MethodDeclaration |             |
| comp.acceptVisitor(v)     | [...acceptVisitor(...)]@[equ | MethodInvocation  |             |

**Operator list**

- Generalize method invocations.
- Refinement
  - Add directive invoked-by.
  - Add directive overrides.
  - Add directive refers-to.

Requires matches to be invoked by the binding variable.

| Operand                 | Value             |
|-------------------------|-------------------|
| Subject                 | [public void acce |
| Meta-variable (e.g. ?v) |                   |

**Matches**

| ?MethodInvocation   | ?invocation         | ?MethodDeclaration |
|---------------------|---------------------|--------------------|
| comp.acceptVis...   | comp.acceptVis...   | ▲ Composite.a...   |
| cs.acceptVisitor... | cs.acceptVisitor... | ▲ Composite.a...   |
| comp.acceptVis...   | comp.acceptVis...   | ▲ Component....    |
| comp.acceptVis...   | comp.acceptVis...   | ▲ MethodDecl...    |

# Motivation

- Code templates lower the learning curve of specifying program searches/modifications
- Hard to master: not always evident to produce only the desired matches

# Example

An acceptVisitor declaration-call pair:

```
public void acceptVisitor(ComponentVisitor v) {  
    Iterator<Component> i=elements.iterator();  
    while (i.hasNext()) {  
        Component comp=(Component)i.next();  
        comp.acceptVisitor(v);  
    }  
}
```

```
cs.acceptVisitor(vstor);
```

# Example

And another:

```
public void acceptVisitor(ComponentVisitor v) {
    Iterator<Component> i = elements.iterator();
    while (i.hasNext()) {
        Component comp = (Component) i.next();
        comp.acceptVisitor(v);
    }
}

comp.acceptvisitor(v);
```

# Example

And another:

```
public void acceptVisitor(ComponentVisitor v) {  
    System.out.println("Prototypical.");  
    v.visitPrototypicalLeaf(this);  
}
```

```
comp.acceptvisitor(v);
```

# Example

... and several others!

How to write a template that describes all of these?

# Example

Initial attempt:

```
public void acceptVisitor(...) ...
```

```
comp.acceptvisitor(v);
```

Too few matches!

# Example

Second attempt:

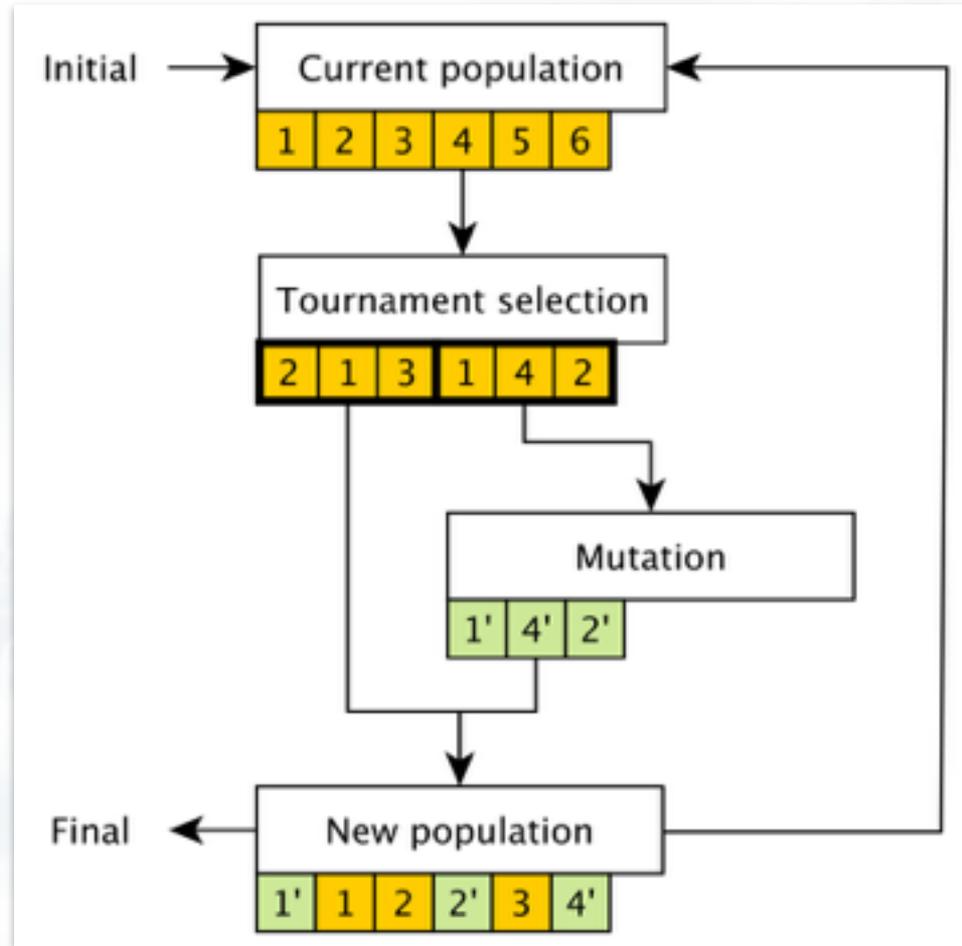
```
public void acceptVisitor(...) ...  
....acceptvisitor(...);
```

Too *many* matches!

# Recommending edits

- How to assist users if a template produces too few/many matches?
- Genetic search approach: Find a sequence of edits leading to more desired matches
- User marks what is desired/undesired

# Genetic algorithm



# Fitness function

- Determines “how good” a template is
- Produces a number between 0 and 1
- Consists of two components:
  - F-score: how many (un)desired matches?
  - Partial score: For each desired match, what % of the template corresponds to the desired match?

# Recommender system

invokedby-initi solution.ekt New Recommendation 1

Input template: [invokedby-initial.ekt](#)

| ?MethodDeclaration12211        | ?MethodInvocation12309  |
|--------------------------------|-------------------------|
| Composite.acceptVisitor        | cs.acceptVisitor(vstor) |
| SuperLogLeaf.acceptVisitor     | comp.acceptVisitor(v)   |
| MustAliasLeaf.acceptVisitor    | comp.acceptVisitor(v)   |
| Component.acceptVisitor        | comp.acceptVisitor(v)   |
| PrototypicalLeaf.acceptVisitor | comp.acceptVisitor(v)   |
| MayAliasLeaf.acceptVisitor     | comp.acceptVisitor(v)   |
| Composite.acceptVisitor        | comp.acceptVisitor(v)   |
| OnlyLoggingLeaf.acceptVisitor  | comp.acceptVisitor(v)   |

| Gen | Best Fitness    | Best F1         | Best Partial    |
|-----|-----------------|-----------------|-----------------|
| 0   | 0.2455607476... | 0.2222222222... | 0.4556074766... |
| 1   | 0.2461904761... | 0.2222222222... | 0.4619047619... |
| 2   | 0.2474489795... | 0.2222222222... | 0.4744897959... |
| 3   | 0.2605468749... | 0.2222222222... | 0.60546875      |
| 4   | 0.2611111111... | 0.2222222222... | 0.6111111111... |

# Recommender system

invokedby-initi solution.ekt New Recommendation 1

Input template: [invokedby-initial.ekt](#) 1. Select input template

| Method                         | Method                  |
|--------------------------------|-------------------------|
| ?MethodDeclaration12211        | ?MethodInvocation12309  |
| Composite.acceptVisitor        | cs.acceptVisitor(vstor) |
| SuperLogLeaf.acceptVisitor     | comp.acceptVisitor(v)   |
| MustAliasLeaf.acceptVisitor    | comp.acceptVisitor(v)   |
| Component.acceptVisitor        | comp.acceptVisitor(v)   |
| PrototypicalLeaf.acceptVisitor | comp.acceptVisitor(v)   |
| MayAliasLeaf.acceptVisitor     | comp.acceptVisitor(v)   |
| Composite.acceptVisitor        | comp.acceptVisitor(v)   |
| OnlyLoggingLeaf.acceptVisitor  | comp.acceptVisitor(v)   |

| Gen | Best Fitness    | Best F1         | Best Partial    |
|-----|-----------------|-----------------|-----------------|
| 0   | 0.2455607476... | 0.2222222222... | 0.4556074766... |
| 1   | 0.2461904761... | 0.2222222222... | 0.4619047619... |
| 2   | 0.2474489795... | 0.2222222222... | 0.4744897959... |
| 3   | 0.2605468749... | 0.2222222222... | 0.60546875      |
| 4   | 0.2611111111... | 0.2222222222... | 0.6111111111... |

# Recommender system

invokedby-initi solution.ekt New Recommendation

Input template: [invokedby-initial.ekt](#) 1. Select input template

| ?MethodDeclaration12211        | ?MethodInvocation12309  |
|--------------------------------|-------------------------|
| Composite.acceptVisitor        | cs.acceptVisitor(vstor) |
| SuperLogLeaf.acceptVisitor     | comp.acceptVisitor(v)   |
| MustAliasLeaf.acceptVisitor    | comp.acceptVisitor(v)   |
| Component.acceptVisitor        | comp.acceptVisitor(v)   |
| PrototypicalLeaf.acceptVisitor | comp.acceptVisitor(v)   |
| MayAliasLeaf.acceptVisitor     | comp.acceptVisitor(v)   |
| Composite.acceptVisitor        | comp.acceptVisitor(v)   |
| OnlyLoggingLeaf.acceptVisitor  | comp.acceptVisitor(v)   |

2. Select desired matches

| Gen | Best Fitness    | Best F1         | Best Partial    |
|-----|-----------------|-----------------|-----------------|
| 0   | 0.2455607476... | 0.2222222222... | 0.4556074766... |
| 1   | 0.2461904761... | 0.2222222222... | 0.4619047619... |
| 2   | 0.2474489795... | 0.2222222222... | 0.4744897959... |
| 3   | 0.2605468749... | 0.2222222222... | 0.60546875      |
| 4   | 0.2611111111... | 0.2222222222... | 0.6111111111... |

# Recommender system

invokedby-initi solution.ekt New Recommendation 1

3. Start

Input template: [invokedby-initial.ekt](#) 1. Select input template

| ?MethodDeclaration12211        | ?MethodInvocation12309  |
|--------------------------------|-------------------------|
| Composite.acceptVisitor        | cs.acceptVisitor(vstor) |
| SuperLogLeaf.acceptVisitor     | comp.acceptVisitor(v)   |
| MustAliasLeaf.acceptVisitor    | comp.acceptVisitor(v)   |
| Component.acceptVisitor        | comp.acceptVisitor(v)   |
| PrototypicalLeaf.acceptVisitor | comp.acceptVisitor(v)   |
| MayAliasLeaf.acceptVisitor     | comp.acceptVisitor(v)   |
| Composite.acceptVisitor        | comp.acceptVisitor(v)   |
| OnlyLoggingLeaf.acceptVisitor  | comp.acceptVisitor(v)   |

2. Select desired matches

| Gen | Best Fitness    | Best F1         | Best Partial    |
|-----|-----------------|-----------------|-----------------|
| 0   | 0.2455607476... | 0.2222222222... | 0.4556074766... |
| 1   | 0.2461904761... | 0.2222222222... | 0.4619047619... |
| 2   | 0.2474489795... | 0.2222222222... | 0.4744897959... |
| 3   | 0.2605468749... | 0.2222222222... | 0.60546875      |
| 4   | 0.2611111111... | 0.2222222222... | 0.6111111111... |

*Demo*

# Summary

- Recommender system to assist Cha-Qeko/X users
- Future work:
  - Generalize instances of a systematic change
  - Migrate e.g. a bug fix between different versions of the same software system



Automated Generalization and Refinement of Source Code Templates Using Mutation Operators

*Tim Molderez and Coen De Roover*

*23rd IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER'16), Tool track*