Clipboard:

- Systematic Editing with Speculative Analysis and Naming Synthesis

Lisa Hua
Supervisor: Miryung Kim
April 30, 2014

Motivating Example

org.eclipse.jdt.core.dom.DefaultCommentMapper.v9800 and v9801

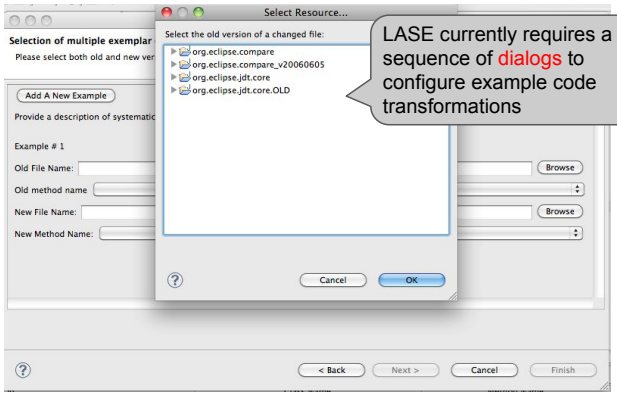
```
1. class DefaultCommentMapper {
2. - HashMap<ASTNode, int[]> leadingComments;
3. - HashMap<ASTNode, int[]> trailingComments;
4. + ASTNode[] leadingNodes;
5. + int leadingPtr;
6. + int trailingPtr;
7. + ASTNode[] trailingNodes;
8.
9. Comment[] getLeadingComments(ASTNode node) {
10. - if (this.leadingComments != null) {
11. -  int[] range = (int[])this.leadingComments.get(node);
12. +     if (this.leadingPtr >= 0) {
13. +   int[] range = null;
14. +   for (int i=0; range==null && i<=this.leadingPtr; i++) {
15. +   if (this.leadingNodes[i] == node)
16. +       range=this.leadingIndexes[i];
17. +   }
18.   if (range != null) {
19.  int length = range[1]-range[0]+1;
20.  Comment[] leadComments = new Comment[length];
21.  return  leadComments;
22.  }
23. }
24. return null;
25.}
26. Comment[] getTrailingComments(ASTNode node) {
27. - if (this.trailingComments != null) {
28. -  int[] range = (int[])this.trailingComments.get(node);
29. +     if (this.trailingPtr >= 0) {
30. +   int[] range = null;
31. +   for (int i=0; range==null && i<=this.trailingPtr; i++) {
32. +    if (this.trailingNodes[i] == node)
33. +        range=this.trailingIndexes[i];
34. +   }
35.   if (range != null) {
36.    int length = range[1]-range[0]+1;
37.    Comment[] trailComments = new Comment[length];
38.    return  trailComments;
39.  }
40. }
41. return null;
42.}
```

Use dialog to specify systematic edit

Naming Synthesis: Generate identifier names that do not exist before editing

Speculative Analysis: Proactively apply the transformation and check the compilation errors

Limitation 1
- Difficulty of specifying input



LASE currently requires a sequence of dialogs to configure example code transformations

Limitation 2 - Difficulty in producing readable variable names

A_{old} to A_{new}

```
1. public IActionBars getActionBars() {
2. - if (fContainer == null) {
3. +  IActionBars actionBars = fContainer.getActionBars();
4. +  if (actionBars == null && !fContainerProvided) {
5.     return Utilities.findActionBars(fComposite);
6.   }
7. - return fContainer.getActionBars();
8. + return actionBars;
9. }
```

B_{old} to $B_{suggested}$ by LASE

```
1. public IServiceLocator getServiceLocator() {
2. - if (fContainer == null) {
3. + IServiceLocator v_1 = fContainer.getServiceLocator();
4. + if (v_1  == null && !fContainerProvided) {
5.     return Utilities.findSite(fComposite);
6.   }
7. - return fContainer.getServiceLocator();
8. + return v_1;
9.}
```

B_{old} to $B_{suggested}$ by Clipboard

```
1. public IServiceLocator getServiceLocator() {
2. - if (fContainer == null) {
3. + IServiceLocator serviceLocator = fContainer.getServiceLocator();
4. + if (serviceLocator == null && !fContainerProvided) {
5.     return Utilities.findSite(fComposite);
6.   }
7. - return fContainer.getServiceLocator();
8. + return serviceLocator;
9.}
```

Awkward identifier names make it difficult for developers to comprehend the code inserted by automated program transformation

Limitation 3 - Lacking of speculating impacts of program transformation

org.eclipse.jdt.core.dom.DefaultCommentMapper.v9800 and v9801

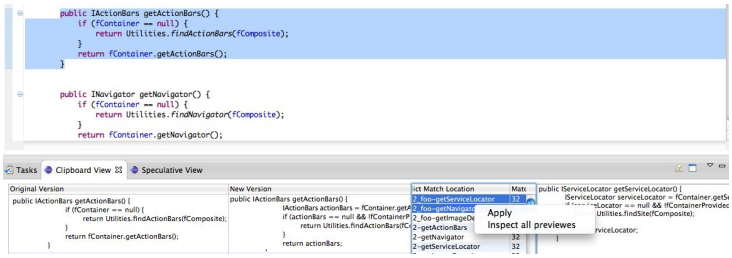
```
1. class DefaultCommentMapper {
2. - HashMap<ASTNode, int[]> leadingComments;
3. - HashMap<ASTNode, int[]> trailingComments;
4. + ASTNode[] leadingNodes;
5. + int leadingPtr;
6.
7. Comment[] getLeadingComments(ASTNode node) {
8. - if (this.leadingComments != null) {
9. -  int[] range = (int[])this.leadingComments.get(node);
10. +     if (this.leadingPtr >= 0) {
11. +   int[] range = null;
12. +   for (int i=0; range==null && i<=this.leadingPtr; i++) {
13. +   if (this.leadingNodes[i] == node)
14. +       range=this.leadingIndexes[i];
15. +   }
16.   if (range != null) {
17.  int length = range[1]-range[0]+1;
18.  Comment[] leadComments = new Comment[length];
19.  return  leadComments;
20.  }
21. }
22. return null;
23.}
24. Comment[] getTrailingComments(ASTNode node) {
25. - if (this.trailingComments != null) {
26. -  int[] range = (int[])this.trailingComments.get(node);
27. +     if (this.trailingPtr >= 0) {
28. +   int[] range = null;
29. +   for (int i=0; range==null && i<=this.trailingPtr; i++) {
30. +    if (this.trailingNodes[i] == node)
31. +        range=this.trailingIndexes[i];
32. +   }
33.   if (range != null) {
34.    int length = range[1]-range[0]+1;
35.    Comment[] trailComments = new Comment[length];
36.    return  trailComments;
37.  }
38. }
39. return null;
40.}
```

Missing relevant changes could introduce compilation errors and developers may want to see these errors in advance

Contribution

1. Demonstrate systematic edit via drag-and-drop
2. Synthesize variable name based on the naming pattern
3. Perform speculative analysis by informing them of the consequence of the edit

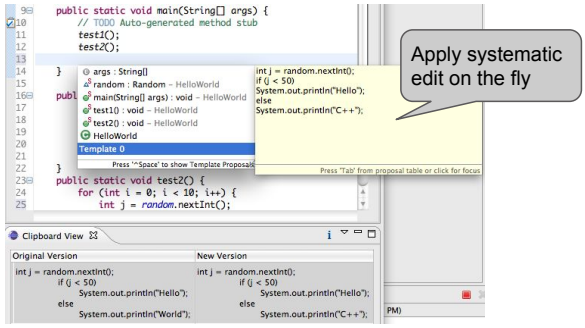
Solution 1- Multi-Clipboard



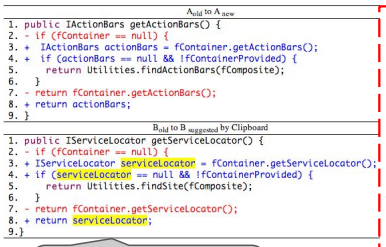
Demonstrate systematic edits via Drag-and-Drop

Recommend relevant edit locations and apply multiple edits simultaneously

Solution 1- Multi-Clipboard (contd)



Solution 2 - Naming Synthesis



Provide reasonable identifier names based on the naming pattern

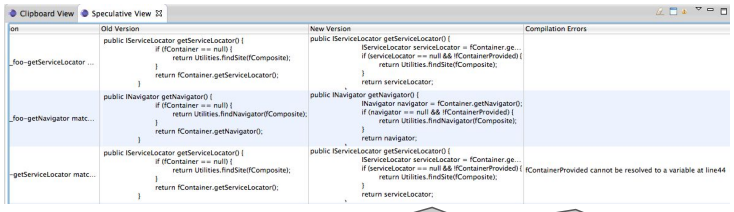
1. Extract edit operations:
 Insert: `IActionBars actionBars = fContainer.getActionBars()`

2. Compare matched nodes:
1) IActionBars → IServiceLocator
2) fContainer.getActionBars() → fContainer.getServiceLocator()

3. Extract different character sequence:
1) ActionBars → ServiceLocator
2) ActionBars → ServiceLocator

4. Consider case sensitive:
actionBars → serviceLocator

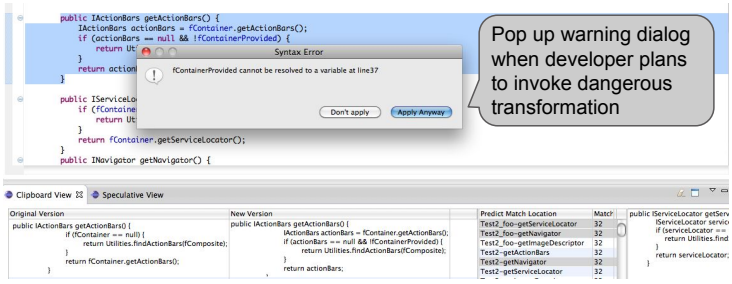
Solution 3 - Speculative analysis



Display the code snippet after the likely systematic edit

Shows compilation error introduced by the likely transformation

Solution 3 - Speculative analysis (contd)



Pop up warning dialog when developer plans to invoke dangerous transformation

1. Apply the systematic edit
2. Return Eclipse compilation errors
3. Undo the change if any errors occur.

Evaluation

- User study
  - blind experiments
    - identifier name
    - auto-systematic edits
  - User study
    - Compare with manual editing
  - Questionnaire

Related Work

- Systematic edits and code completion
  - Sydit, LASE, Cookbook
  - GraPacc
- Speculative Analysis
  - Quick Fix Scout
  - Crystal
- Naming Pattern
  - Class name pattern
  - Method name vs implementation

Future Work

- Opportunistic Refactoring
  - Extract method and recommend method name
- Naming Synthesis with constraint solver

Contribution

1. Demonstrate systematic edit via drag-and-drop
2. Synthesize variable name based on the naming pattern
3. Perform speculatively analysis by informing them of the consequence of the edit

Reference

1. J. Jacobellis, N. Meng, and M. Kim. Cookbook: In situ code completion using edit recipes learned from examples. In proceeding to ICSE'14.
2. N. Meng, M. Kim, and K. S.McKinley. Lase: Locating and applying systematic edits by learning from examples. ICSE '13, pages 502–511, 2013.
3. K. Muslu, Y. Brun, R. Holmes, M. D. Ernst, and D. Notkin. Speculative analysis of integrated development environment recommendations. OOPSLA '12, pages 669–682, 2012.