Weekly Meeting 10/20/2015

Structural results format

1. PROBLEM STATEMENT

Problem: The code examples returned from code search engine or other code search tools are not structured.

Input: a set of code examples returned from code search engine.

Output: a list of code example clusters organized based on their structural similarity.

2. USAGE SCENARIO

To further illustrate that reuse task involves in multiple methods and classes, and the reuse task is hard without tool support, I imagine a scenario that user wants to add undo and redo actions for her Java Swing text editor application with undo/redo buttons, inspired by the evaluation task used in [4].

Query: undo redo TextEditor

Without knowing any APIs, developer first queries code search engine (CSE) with a free-form query `'undo redo TextEditor'`. Table 1 shows top 10 results returned from SearchCode CSE.

Before using our tool, she has to manually investigate all code search examples to identify which API calls are frequently used. She selects the first result from CSE and uses keyword search to locate undo and redo feature in 6 methods and regards them as search seed. She started to explore the relationships between these methods. Shown in Figure 1 (A), she removes the other 11 methods (865 LOC) that she regards as irrelevant to her reuse task. She recognizes that she needs to implement an `UndoAction` and `RedoAction` which are the subclass of `AbstractAction`, and overrides their actionPerform() method. In the actionPerformed() methods, she should invokes the `CompoundUndoManager.undo()` and `CompoundUndoManager.redo()` correspondingly. Without knowing anything about `CompoundUndoManager`, she has to query CSE again for 'CompoundUndoManager textmash'. This class consists of 17 methods (221 LOC) and she has to repeat to keyword search again to locate undo and redo feature in this class. With manual inspection, she notices that this `CompoundUndoManager` is a subclass of `javax.swing.UndoManager` and it overrides four methods {`canUndo, canRedo, undo, redo`} that are related to the feature. She notices that she actually does not need this `CompoundUndoManager` and decides to invoke its parent class `UndoManager` instead. She also notices that she does not need the `CannotRedoException` by using `UndoManager`.

After she integrates `UndoManager, UndoAction, and RedoAction` to her context, she tests it and it fails to perform the feature. She has to look at other examples, and she finds the fifth example from CSE is similar to the first example. Shown in Figure 1 (B), This example seems similar because it uses an `TextUndoManager`, and implements `UndoAction` and `RedoAction`. She notice that the API `doc.addUndoable EditListener(undoManager)` seems the one that she misses now, but again, she has to investigate the class `TextUndoManager` again.

With this example, we illustrate that salient API calls that implement a desired feature are always interleaving with other related elements. To finish a reuse task, developer has to remove irrelevant parts. However, it requires significant effort to tease out these auxiliary features without tool support. Since the examples always contain both main feature and auxiliary features, there seldom exists a perfect example for reuse and developers has to iteratively search for new examples and integrate it to their context until the integration results are examined.

Our tool is able to represent code examples returned from CSE in a structured manner. We cluster the examples that share similar API calls and extract the queried features. Shown in Figure 1 (C), our tool extracts a skeleton of reuse code example for the queried feature. At the same time, we group auxiliary features based on program slicing and lexical similarity so that the user can select the features that she wants, as shown in Figure 1 (D).

3. APPROACH

Our approach helps identify and integrate reusable examples in two phases:

3.1 Example Collection Phase.

We identify the main features from multiple partial programs derived from informal resources. Without confined to an established code corpus, we leverage web search engine to obtain a list of code examples in the form of partial programs and use partial program analysis [2] to infer partial type and resolve syntactic ambiguity.

Table 1: Search Result from Code Search Engine

No.	Name	Project	LOC	#M	#C	Involved objects and API calls
1	TextEditor	textmash	1271	12	1	UndoManager, UndoAction, RedoAction
4	ConsoleTextEditor	groovy-core	322	7	1	UndoManager, UndoAction, RedoAction
2	AndroidTextEditor	android-sdk	587	9	2	IActionBars.setGlobalActionHandler(), ActionFactory.UNDO, ActionFactory.REDO
8	LayoutCanvas	sdk	1721	3	2	IActionBars.setGlobalActionHandler(), ActionFactory.UNDO, ActionFactory.REDO
9	IUEditorContributor	buckminster	193	2	2	IActionBars.setGlobalActionHandler(), ActionFactory.UNDO, ActionFactory.REDO
5	PapyrusCDTEditor	papyrus	393	7	4	IActionBars.setGlobalActionHandler(), ITextEditorActionConstants.UNDO
6	AspectEditorContributor	eclipse-plugin	89	2	4	IActionBars.setGlobalActionHandler(), ITextEditorActionConstants.UNDO
10	WSDLContributor	eclipse	254	2	4	IActionBars.setGlobalActionHandler(), ITextEditorActionConstants.UNDO
3	DocumentUndoImpl	ide	1228	17	3	FileDocumentUndoManager
7	ATETextPane	antlrworks	650	7	5	AbstractUndoableEdit.undo(), AbstractUndoableEdit.redo(),

No. represents the rank from CSE. **#M** represents the number of methods that contain the query terms. **#C** represents the number of clusters the example belongs to.

Query Code Search Engine. We use code search engine SearchCode [5] to extract code examples based on free-form queries from users. We choose it because 1) it has good support for free-form queries, 2) it is an open source code search engine with over 7000 projects from Github, Bitbucket, Google Code, and Sourceforge, 3) it has complete API documentations and returns.

Locate Queried Features. Considering that source code examples always contain multiple features, we identify program elements that are related to the queried features. I use the mean of TF-IDF weight for each query term as a weighting factor and select all methods whose weighting score is bigger than a threshold. This approach is similar to prior works that use IR [3] and NL analysis [6] for feature location. I choose IR approach because other approaches require history or structural analysis that might not be feasible for partial program. TF-IDF $= avg(\log(1+f_{t,d}) \times \log \frac{N}{n_t})$, $f_{t,d}$ is the frequency of term t in method d, N is the total number of methods, n_t is the number of methods that have the term t.

Extract Structural Facts. Before we extract structural facts, we construct symbol table for partial program based on the partial program analysis [2]. We extract structural facts in an ontology instance. The ontology schema we use is the SEON Java ontology, which has been used in several prior works [4, 7]. Every extracted fact is represented in the ontology instance by one or more `(subject, predicate, object)` triples. For instance, the fact that class c_1 extends class c_2 is represented by triple (c_2, `subType`, c_1). To manipulate the relationship between different facts, we use Resource Description Framework (RDF) to represent each `(subject, predicate, object)` triple as an edge from *subject* to *object* labeled with *predicate*. We use `Apache Jena`, which is a Java framework for linked data applications, to create and process ontology instances. We define `subType` as transitive predicate as (c_1, subType, c_2) and (c_2, subType, c_3) $\rightarrow$ (c_1, subType, c_3).

3.2 Example Clustering Phase.

We take three steps in this phase. Based on the type facts extracted from partial program analysis, we first cluster similar code examples based on the number of shared class-level facts and API calls (i.e., `call` facts) to reduce the duplicated code snippets. Next, for each example in each cluster, we identify main facts that are shared among all examples, and perform slicing to identify related elements. Finally, we group auxiliary features based on program slicing and lexical similarity.

Table 2: Structural Fact Types

Level	Type	Description
class level	(c_1, subType, c_2)	class c_1 extends c_2
	(c_1, innerclass, c_2)	class c_1 has an inner class c_2
	(c, field, v)	class c has field v
	(c, method, m)	class c contains method m
method level	(m, parameter, v)	method m has parameter v
	(m, return, t)	method m returns an object of type t
	(e_1, accesses, e_1)	$e_1 = e_2$, e_1 and e_2 are expressions
	(m_1, call, m_2)	method m_1 calls method m_2
	(m, variable, v)	method m has variable m_2
	(o, name, s)	The name of object o is s

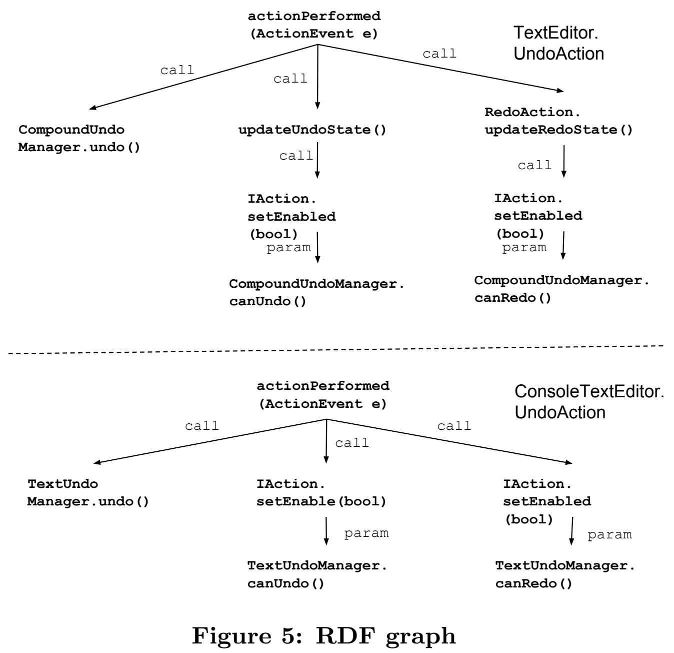

actionPerformed (ActionEvent e)
TextEditor. UndoAction
call
call
call
CompoundUndo Manager.undo()
updateUndoState()
RedoAction. updateRedoState()
call
call
IAction. setEnabled (bool)
IAction. setEnabled (bool)
param
param
CompoundUndoManager. canUndo()
CompoundUndoManager. canRedo()
actionPerformed (ActionEvent e)
ConsoleTextEditor. UndoAction
call
call
call
TextUndo Manager.undo()
IAction. setEnable(bool)
IAction. setEnabled (bool)
param
param
TextUndoManager. canUndo()
TextUndoManager. canRedo()

Figure 5: RDF graph

Cluster Examples. We greedily select the most suitable structural correspondence [1] and propagate the similarity of *object* nodes to the *subject* node.

The comparison is done by investigating the lexical similarity and similar paths in RDF graphs. For example, when comparing UndoAction.actionPerformed() in Figure 1 (A), we first try to match the fact `(actionPerformed, calls, CompoundUndoManager. undo())`. We only look at the facts with the same type. We find the fact `(actionPerformed, calls, TextUndoManager.undo())` which is lexically similar though not identical to the target fact. We calculate the similarity score as follows: 1) We split the name for each element in the tuple by camel-case splitter and separators such as '.' and '('. Thus `TextUndoManager.undo()` is split as `[Text, Undo, Manager, undo]`. 2) We calculate the distance for each element as the $editDistance(s_1, s_2)/Min\ (|s_1|, |s_2|)$, thus the similarity score between `TextUndoManager.undo()` and `CompoundUndoManager.undo()` is 0.25. 3) The distance between two tuples are the average distance for three elements. Thus the distance of two facts are 0.08. We select the most similar fact by minimizing the similarity distance and we identify `(actionPerformed, calls, CompoundUndoManager.undo())` as the structural correspondence of the fact `(actionPerformed, calls, TextUndoManager.undo())`.

Next, we try to match the fact `(actionPerformed, calls, updateUndoState())`. Knowing that there is an edge from `updateUndoState()` to `setEnabled(undo.canUndo()` in RDF graph, we recursively match the fact `(updateUndoState, calls, setEnabled(undo.canUndo()))` with `(actionPerformed, calls, setEnabled(undo.canUndo()))`. The similarity score of *object* node `(updateUndoState, calls, setEnabled(undo.can Undo()))` is propagated to *subject* node `(actionPerformed, calls, updateUndoState())` as 0.4 compared to fact `(actionPerformed, calls, setEnabled(undo. canUndo()))` in class `UndoAction`. Our tool keeps on performing this matching until all facts in the target (Figure 1 (A)) is assigned a similarity score as 0 (i.e., fail to find correspondence) or a similarity score with other facts in the compared node.

Finally, the examples are clustered based on the number of *object* nodes (without outer edge). Our tool keeps on refining the large cluster to present a structured code examples to developers.

Extract Example Skeleton. We extract all matched *object* nodes (without outer edge) and use them as match seeds. We then perform dependency analysis to extract example skeleton. An example skeleton should start from an element without any inner edge and end at an element without any outer edge which covers all match seeds.

When selecting features, we use three rules to rank features in different examples: 1) the cost of reuse; 2) the number of resolved relationship; 3) the frequency of the feature in multiple examples; 4) the simplicity of code implementation.

The cost of reuse is measured as the number of outer edges in RDF graph, i.e., the more outer edges in RDF graph, the more expensive to reuse because it has more dependencies. We have rule 2) because some of the relationships can not be resolved in partial program. When user tries to reuse the example, they have to manually resolve these external relationships. For instance, in Figure 3 (D), the feature group B.1 (`mUndoAction = new LintEditAction(undoAction, getEditorDelegate().getEditor())`) is not selected because the class `LintEditAction` cannot be resolved based on this partial program. The first two rules ease the reusability of our suggested code example. The third rule makes sure that the feature is representative and the last rule improves the understandability of our results. The simplicity is measured as the length of the code. For instance, we prefer `actionBars.setGlobalActionHandler (ITextEditorActionConstants.UNDO, undo)` in Figure 4 (D) than `actionBars.setGlobalActionHandler (ITextEditorActionConstants.UNDO, getAction((ITextEditor) part, ITextEditorActionConstants.UNDO))` because the former one is easier to interpret.

Group Auxiliary Features. The auxiliary features are different example skeletons. (TBD)

Enable User Interaction. The user could select the features and our tool is able to extract related elements based on their selection. (TBD)

4. REFERENCES

[1] R. Cottrell, R. J. Walker, and J. Denzinger. Semi-automating small-scale source code reuse via structural correspondence. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2008, Atlanta, Georgia, USA, November 9-14, 2008*, pages 214–225, 2008.

[2] B. Dagenais and L. J. Hendren. Enabling static analysis for partial java programs. In *Proceedings of the 23rd Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2008, October 19-23, 2008, Nashville, TN, USA*, pages 313–328, 2008.

[3] D. Poshyvanyk, M. Gethers, and A. Marcus. Concept location using formal concept analysis and information retrieval. *ACM Trans. Softw. Eng. Methodol.*, 21(4):23, 2012.

[4] S. Rastkar, G. C. Murphy, and A. W. J. Bradley. Generating natural language summaries for crosscutting source code concerns. In *IEEE 27th International Conference on Software Maintenance, ICSM 2011, Williamsburg, VA, USA, September 25-30, 2011*, pages 103–112, 2011.

[5] SearchCode. https://searchcode.com.

[6] D. C. Shepherd, Z. P. Fry, E. Hill, L. L. Pollock, and K. Vijay-Shanker. Using natural language program analysis to locate and understand action-oriented concerns. In *Proceedings of the 6th International Conference on Aspect-Oriented Software Development, AOSD 2007, Vancouver, British Columbia, Canada, March 12-16, 2007*, pages 212–224, 2007.

[7] M. Würsch, G. Ghezzi, G. Reif, and H. C. Gall. Supporting developers with natural language queries. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1, ICSE 2010, Cape Town, South Africa, 1-8 May 2010*, pages 165–174, 2010.

(A) Cluster 1: textmash:TextEditor

```
public class TextEditor extends JTextPane {
 public UndoAction undoAction = new UndoAction();
 public RedoAction redoAction = new RedoAction();
 public CompoundUndoManager undo;
 HashMap<Object, Action> actions = new HashMap<Object, Action>();
 public TextEditor(Workspace workspace) {
   this.workspace = workspace;
   undo = new CompoundUndoManager(workspace);
   actions.put("undo", undoAction);
   actions.put("redo", redoAction);
   }
 public void discardUndoRedo() {
    undo.discardAllEdits();
    undoAction.updateUndoState();
    redoAction.updateRedoState();
   }
 public class UndoAction extends AbstractAction {
   public UndoAction()
{     super("undo"); }
  public void actionPerformed(ActionEvent e) {
   try {
     undo.undo();
     updateUndoState();
     redoAction.updateRedoState();
    } catch (CannotUndoException ex) {
    }}
 public void updateUndoState() {
     setEnabled(undo.canUndo());
   }}
 public class RedoAction extends AbstractAction {
   public RedoAction()
{     super("redo"); }
   public void actionPerformed(ActionEvent e) {
    try {
      undo.redo();
      updateRedoState();
      undoAction.updateUndoState();
     } catch (CannotRedoException ex) {
     }}
  public void updateRedoState() {
      setEnabled(undo.canRedo());
     }}
```

(B) Cluster 1: groovy: ConsoleTextEditor

```
public class ConsoleTextEditor extends JScrollPane {
    private UndoAction undoAction = new UndoAction();
    private RedoAction redoAction = new RedoAction();
    private TextUndoManager undoManager;
   private TextEditor textEditor = new TextEditor();
   private Document doc = textEditor.getDocument();
   public ConsoleTextEditor () {
     this.undoManager = new TextUndoManager();
     doc.addUndoableEditListener(undoManager);
     undoManager.addPropertyChangeListener(undoAction);
     undoManager.addPropertyChangeListener(redoAction);
     doc.addDocumentListener(undoAction);
     doc.addDocumentListener(redoAction);
 }
private class RedoAction extends UpdateCaretListener
 implements PropertyChangeListener {
   public RedoAction()
{     setEnable(false); }
  public void actionPerformed(ActionEvent ae) {
   undoManager.redo();
setEnabled(undoManager.canRedo());
undoAction.setEnabled(undoManager.canUndo());
 }
   public void propertyChange(PropertyChangeEvent pce) {
    setEnabled(undoManager.canRedo());
  } }
private class UndoAction extends UpdateCaretListener
implements PropertyChangeListener {
  public RedoAction()
{     setEnable(false); }
  public void actionPerformed(ActionEvent ae) {
   undoManager.undo();
   setEnabled(undoManager.canUndo());
   redoAction.setEnabled(undoManager.canRedo());
}
 public void propertyChange(PropertyChangeEvent pce) {
   setEnabled(undoManager.canUndo());
 }
   private abstract class UpdateCaretListener extends AbstractAction
    implements DocumentListener {
      public void actionPerformed(ActionEvent ae) {}
    }}
```

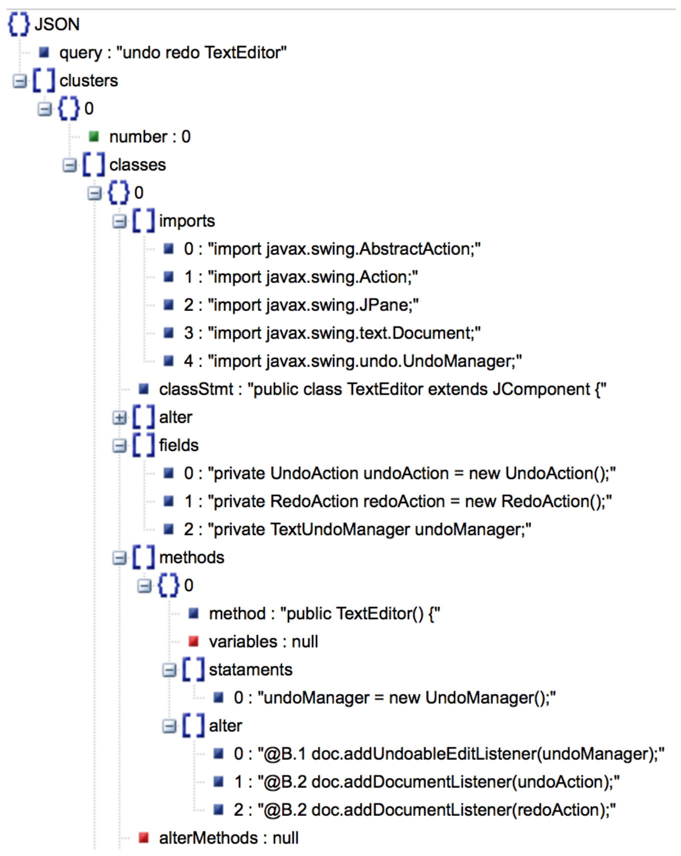
(C) Cluster 1: suggested skeleton

```
public class TextEditor extends JComponent {
    private UndoAction undoAction = new UndoAction();
    private RedoAction redoAction = new RedoAction();
    private TextUndoManager undoManager;

   public TextEditor () {
     undoManager = new TextUndoManager();





  }
 private class UndoAction extends AbstractAction {


  public void actionPerformed(ActionEvent e) {
    undoManager.undo();
    setEnabled(undoManager.canUndo());
   redoAction.setEnabled(undoManager.canRedo());
 }


  }
  private class RedoAction extends AbstractAction {


  public void actionPerformed(ActionEvent e) {
   undoManager.redo();
   setEnabled(undoManager.canRedo());
   undoAction.setEnabled(undoManager.canUndo());
 }


   }  }
class TextUndoManager extends UndoManager() {}
```

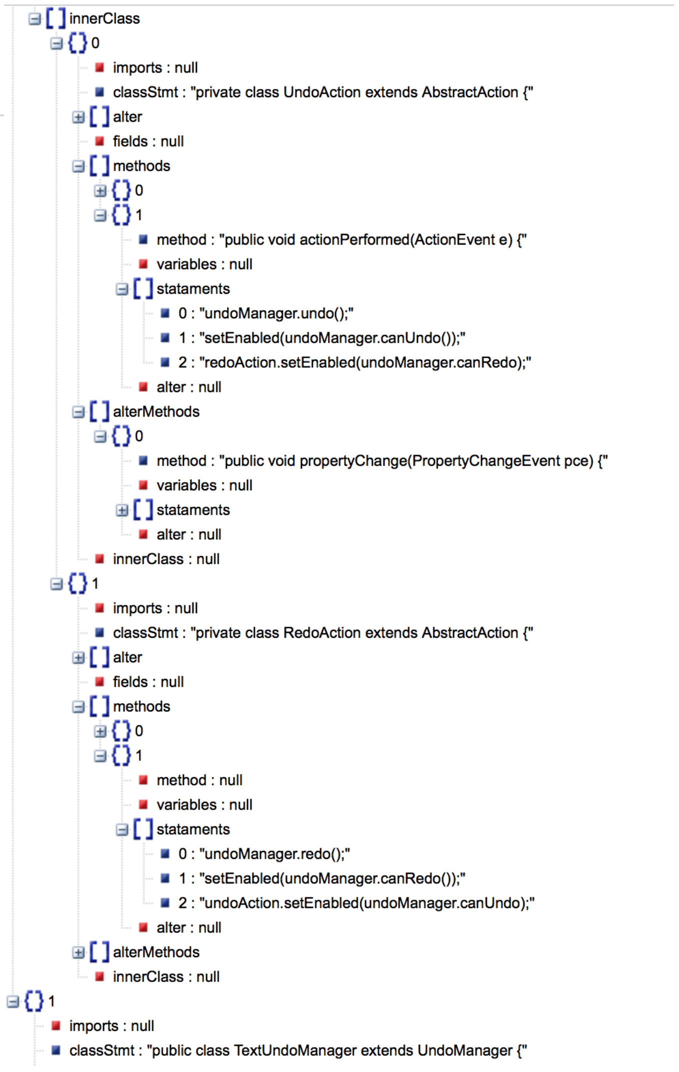
(D) Suggested skeleton with alternative features

```
public class TextEditor extends JComponent {
    private UndoAction undoAction = new UndoAction();
    private RedoAction redoAction = new RedoAction();
    private TextUndoManager undoManager;
//@B.1,B2  private Document doc = textEditor.getDocument();
   public TextEditor () {
     undoManager = new TextUndoManager();
     //@B.1 doc.addUndoableEditListener(undoManager);
     //@B.2 doc.addDocumentListener(undoAction);
     //@B.2 doc.addDocumentListener(redoAction);
     //@B.4 doc.addPropertyChangeListener(undoAction);
     //@B.4 doc.addPropertyChangeListener(redoAction);
  }
 private class UndoAction extends AbstractAction {
 //@B.2 implements DocumentListener
 //@B.4 implements PropertyChangeListener
  public void actionPerformed(ActionEvent ae) {
    undoManager.undo();
    setEnabled(undoManager.canUndo());
    redoAction.setEnabled(undoManager.canRedo());
 }
//B.4  public void propertyChange(PropertyChangeEvent pce) {
//B.4    setEnabled(undoManager.canUndo());  }
}
  private class RedoAction extends AbstractAction {
 //@B.2 implements DocumentListener
 //@B.4 implements PropertyChangeListener
  public void actionPerformed(ActionEvent e) {
   undoManager.redo();
   setEnabled(undoManager.canRedo());
   undoAction.setEnabled(undoManager.canUndo());
  }
  //@B.4  public void propertyChange(PropertyChangeEvent pce) {
  //@B.4    setEnabled(undoManager.canUndo());  }
  }  }
  class TextUndoManager extends UndoManager() {}
```

Figure 1: Cluster 1: source code and expected results



```
{} JSON
   query : "undo redo TextEditor"
   [] clusters
      {} 0
         number : 0
         [] classes
            {} 0
               [] imports
                  0 : "import javax.swing.AbstractAction;"
                  1 : "import javax.swing.Action;"
                  2 : "import javax.swing.JPane;"
                  3 : "import javax.swing.text.Document;"
                  4 : "import javax.swing.undo.UndoManager;"
               classStmt : "public class TextEditor extends JComponent {"
               [] alter
               [] fields
                  0 : "private UndoAction undoAction = new UndoAction();"
                  1 : "private RedoAction redoAction = new RedoAction();"
                  2 : "private TextUndoManager undoManager;"
               [] methods
                  {} 0
                     method : "public TextEditor() {"
                     variables : null
                     [] mstataments
                        0 : "undoManager = new UndoManager();"
                     [] alter
                        0 : "@B.1 doc.addUndoableEditListener(undoManager);"
                        1 : "@B.2 doc.addDocumentListener(undoAction);"
                        2 : "@B.2 doc.addDocumentListener(redoAction);"
               alterMethods : null
```



```
[] innerClass
   {} 0
      imports : null
      classStmt : "private class UndoAction extends AbstractAction {"
      [] alter
      fields : null
      [] methods
         {} 0
         {} 1
            method : "public void actionPerformed(ActionEvent e) {"
            variables : null
            [] stataments
               0 : "undoManager.undo();"
               1 : "setEnabled(undoManager.canUndo());"
               2 : "redoAction.setEnabled(undoManager.canRedo);"
            alter : null
      [] alterMethods
         {} 0
            method : "public void propertyChange(PropertyChangeEvent pce) {"
            variables : null
            [] stataments
            alter : null
      innerClass : null
   {} 1
      imports : null
      classStmt : "private class RedoAction extends AbstractAction {"
      [] alter
      fields : null
      [] methods
         {} 0
         {} 1
            method : null
            variables : null
            [] stataments
               0 : "undoManager.redo();"
               1 : "setEnabled(undoManager.canRedo());"
               2 : "undoAction.setEnabled(undoManager.canUndo);"
            alter : null
      [] alterMethods
      innerClass : null
{} 1
   imports : null
   classStmt : "public class TextUndoManager extends UndoManager {"
```

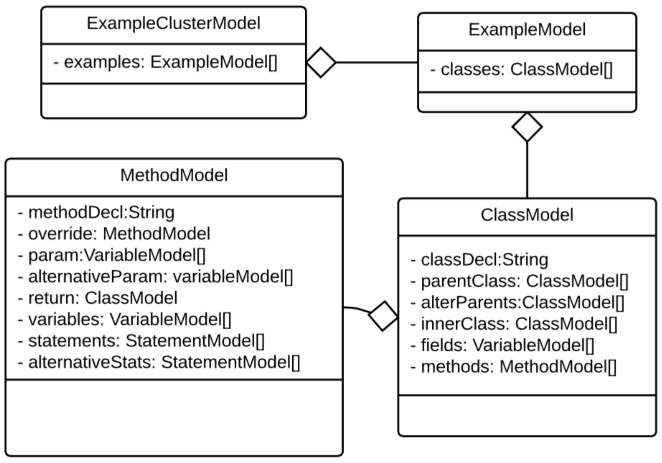

ExampleClusterModel
- examples: ExampleModel[]
ExampleModel
- classes: ClassModel[]
MethodModel
- methodDecl:String
- override: MethodModel
- param:VariableModel[]
- alternativeParam: variableModel[]
- return: ClassModel
- variables: VariableModel[]
- statements: StatementModel[]
- alternativeStats: StatementModel[]
ClassModel
- classDecl:String
- parentClass: ClassModel[]
- alterParents:ClassModel[]
- innerClass: ClassModel[]
- fields: VariableModel[]
- methods: MethodModel[]

Figure 2: Screenshot for cluster 1

(A) Cluster 2: android: AndroidTextEditor

```java
public abstract class AndroidTextEditor extends FormEditor
   implements IResourceChangeListener {
   private TextEditor mTextEditor;
 public void createTextEditor() {
   mTextEditor = new TextEditor();
   IDocumentProvider provider = mTextEditor.getDocumentProvider();
   mDocument = provider.getDocument(getEditorInput());
   mDocument.addDocumentListener(new IDocumentListener() {
     public void documentChanged(DocumentEvent event) {
        onDocumentChanged(event);
      }  });  }
private void createUndoRedoActions() {
IActionBars bars = getEditorSite().getActionBars();
 if (bars != null) {
 IAction action = mTextEditor.getAction(ActionFactory.UNDO.getId());
 bars.setGlobalActionHandler(ActionFactory.UNDO.getId(), action);
 action = mTextEditor.getAction(ActionFactory.REDO.getId());
 bars.setGlobalActionHandler(ActionFactory.REDO.getId(), action);
 bars.updateActionBars();
        }  }    }
```

(B) Cluster 2: android-sdk: LayoutCanvas

```java
public class LayoutCanvas extends Canvas {
   private IAction mUndoAction, mRedoAction;
   private LayoutEditorDelegate mEditorDelegate;;
  @Override
   public void updateGlobalActions(@NonNull IActionBars bars) {
   ITextEditor editor = mEditorDelegate.getEditor().getStructuredTextEditor();
     if (mUndoAction == null) {
         IAction undoAction = editor.getAction(ActionFactory.UNDO.getId());
       mUndoAction = new LintEditAction(undoAction,
         getEditorDelegate().getEditor());
     }
       bars.setGlobalActionHandler(ActionFactory.UNDO.getId(), mUndoAction);
     if (mRedoAction == null) {
         IAction redoAction = editor.getAction(ActionFactory.REDO.getId());
       mRedoAction = new LintEditAction(redoAction,
       getEditorDelegate().getEditor());
    }
      bars.setGlobalActionHandler(ActionFactory.REDO.getId(), mRedoAction);
      bars.updateActionBars();
    }  }
```

(C) Cluster 4: eclipse: PapyrusCDTEditor

```java
public class PapyrusCDTEditor extends CEditor {
    protected IAction textUndo, textRedo;

  @Override
  public void createPartControl(Composite parent) {
      IActionBars actionBars = getEditorSite().getActionBars();
     if (actionBars != null) {
         textUndo = actionBars.getGlobalActionHandler
           (ITextEditorActionConstants.UNDO);
        textRedo = actionBars.getGlobalActionHandler
     (ITextEditorActionConstants.REDO);
     actionBars.setGlobalActionHandler
       (ITextEditorActionConstants.UNDO, textUndo);
    actionBars.setGlobalActionHandler
      (ITextEditorActionConstants.REDO, textRedo);
    actionBars.updateActionBars();
     }
   }
```

(D) Cluster 4: eclipse: AspectEditorContributor

```java
public class AspectEditorContributor extends MultiPageEditorActionBarContributor
 private IEditorPart activeEditorPart;
 @Override
 public void setActivePage(IEditorPart part) {
  activeEditorPart = part;
  IActionBars actionBars = getActionBars();
  if (activeEditorPart != null && activeEditorPart instanceof ITextEditor) {
    IActionBars siteActionBars = ((IEditorSite)activeEditorPart.
        getEditorSite()).getActionBars();
  siteActionBars.setGlobalActionHandler(ITextEditorActionConstants.UNDO,
    getAction((ITextEditor)activeEditorPart, ITextEditorActionConstants.UNDO));
  siteActionBars.setGlobalActionHandler(ITextEditorActionConstants.REDO,
    getAction((ITextEditor)activeEditorPart, ITextEditorActionConstants.REDO));
  siteActionBars.updateActionBars();
  } }
 protected IAction getAction(ITextEditor editor, String actionID) {
   return (editor == null ? null : editor.getAction(actionID));
  }  }
```

(E) Suggested Skeleton for Cluster 2

```java
public class TextEditor extends JComponent {
 public void createUndoRedoAction() {
  ITextEditor editor = new TextEditor();
  IActionBars actionBars = getEditorSite().getActionBars();
     if (actionBars != null) {
       IAction undoAction = editor.getAction
         (ActionFactory.UNDO.getId());
       actionBars.setGlobalActionHandler
         (ActionFactory.UNDO.getId(), undoAction);
       IAction redoAction = editor.getAction
         (ActionFactory.REDO.getId());
       actionBars.setGlobalActionHandler
          (ActionFactory.REDO.getId(), redoAction);
       actionBars.updateActionBars();
} }
```

(F) Suggested Skeleton for Cluster 4

```java
public class TextEditor extends JComponent {
 public void createUndoRedoAction() {

  IActionBars actionBars = getEditorSite().getActionBars();
   IAction undo = actionBars.getGlobalActionHandler
     (ITextEditorActionConstants.UNDO);
   if (actionBars != null) {
     IAction redo = actionBars.getGlobalActionHandler
       (ITextEditorActionConstants.REDO);
    actionBars.setGlobalActionHandler
      (ITextEditorActionConstants.UNDO, undo)
    actionBars.setGlobalActionHandler
      (ITextEditorActionConstants.REDO, redo)
    actionBars.updateActionBars();
  } }
```

(G) Suggested Skeleton for Cluster 2, 4

```java
1. public class TextEditor extends JComponent {
2.  public void createUndoRedoAction() {
3.    IActionBars actionBars = getEditorSite().getActionBars();
4.
5.    if (actionBars != null) {
6.      IAction undo = ...

7.      IAction redo = ...

8.     actionBars.setGlobalActionHandler(..., undo)

9.     actionBars.setGlobalActionHandler(..., redo)
10.   actionBars.updateActionBars();
11.  } }
```

(H) Alternative features for Suggested Skeleton

```java
1.
2.
3.
4. //@B.1 ITextEditor editor = new TextEditor();
5.
6. //@A.1 IAction undo = actionBars.getGlobalActionHandler.(ITextEditorActionConsta
6. //@B.1 IAction undo = editor.getAction(ActionFactory.UNDO.getId());
7. //@A.1 IAction redo = actionBars.getGlobalActionHandler.(ITextEditorActionConsta
7. //@B.1 IAction redo = editor.getAction(ActionFactory.REDO.getId());
8. //@A.1 actionBars.setGlobalActionHandler(ITextEditorActionConstants.UNDO, undo)
8. //@A.1 actionBars.setGlobalActionHandler(ActionFactory.UNDO.getId(), undo);
9. //A.1   actionBars.setGlobalActionHandler(ITextEditorActionConstants.REDO, redo)
9. //A.1   actionBars.setGlobalActionHandler(ActionFactory.REDO.getId(), redo)
 } }
```

Figure 3: Cluster 2, 4: source code and suggested skeleton

(C) Cluster 2: suggested skeleton

```
public class TextEditor extends JComponent {











   public void createUndoRedoActions() {

     ITextEditor editor = new TextEditor();


     IActionBars actionBars = getEditorSite().getActionBars();
     if (bars != null) {

       IAction undoAction = editor.getAction
         (ActionFactory.UNDO.getId());


       actionBars.setGlobalActionHandler
         (ActionFactory.UNDO.getId(), undoAction);



       IAction redoAction = editor.getAction
         (ActionFactory.REDO.getId());




       bars.setGlobalActionHandler
          (ActionFactory.REDO.getId(), redoAction);


       bars.updateActionBars();
    }  }
```

(D) Cluster 2: suggested skeleton with alternative features

```
public class TextEditor extends JComponent {
//@A.1 ITextEditor editor;
//@A.1 public void createTextEditor() {
//@A.1 editor = new TextEditor();
//@A.1 IDocumentProvider provider = mTextEditor.getDocumentProvider();
//@A.1 mDocument = provider.getDocument(getEditorInput());
//@A.1 mDocument.addDocumentListener(new IDocumentListener() {
//@A.1 public void documentChanged(DocumentEvent event) {onDocumentChanged(event);
//@B.1 IAction mUndoAction = new LintEditAction(redoAction,
//@B.1   getEditorDelegate().getEditor());
//@B.1 IAction mRedoAction = new LintEditAction(redoAction,
//@B.1   getEditorDelegate().getEditor());
   public void createUndoRedoActions() {
//@B.2 (@NonNull IActionBars actionBars)
     ITextEditor editor = new TextEditor();
//@A.1 editor = new TextEditor();
//@B.3 ITextEditor editor = getStructuredTextEditor()
     IActionBars actionBars = getEditorSite().getActionBars(); //@deleteif B.2
     if (bars != null) {
//@B.1 if (mUndoAction == null) {
       IAction undoAction = editor.getAction(ActionFactory.UNDO.getId());
//@B.1 mUndoAction = new LintEditAction(undoAction,
//@B.1 getEditorDelegate().getEditor());
//@B.1 }
       actionBars.setGlobalActionHandler
          (ActionFactory.UNDO.getId(), undoAction);
 //@B.1 actionBars.setGlobalActionHandler
//@B.1   (ActionFactory.UNDO.getId(), mUndoAction);
//@B.1 if (mRedoAction == null) {
       IAction redoAction = editor.getAction
          (ActionFactory.REDO.getId());
//@B.1 mUndoAction = new LintEditAction(undoAction,
//@B.1  getEditorDelegate().getEditor());
//@B.1 }
       bars.setGlobalActionHandler
          (ActionFactory.REDO.getId(), redoAction);
 //@B.1 actionBars.setGlobalActionHandler
//@B.1   (ActionFactory.REDO.getId(), mRedoAction);
       bars.updateActionBars();
    }  }
```

(C) Cluster 4: Suggested Skeleton

```
public class TextEditor extends JComponent {

 public void setActivePage(Composite parent) {

   IActionBars actionBars = getEditorSite().getActionBars();


   IAction undo = actionBars.getGlobalActionHandler
     (ITextEditorActionConstants.UNDO);


   IAction redo = actionBars.getGlobalActionHandler
     (ITextEditorActionConstants.REDO);


   actionBars.setGlobalActionHandler
      (ITextEditorActionConstants.UNDO, undo)
   actionBars.setGlobalActionHandler
      (ITextEditorActionConstants.REDO, redo)
} }
```

(D) Cluster 4: Suggested Skeleton with alternative features

```
public class TextEditor extends JComponent {

 public void setActivePage(Composite parent) {
//@B.1 (IEditorPart part)
   IActionBars actionBars = getEditorSite().getActionBars();
//@B.1 IActionBars actionBars = ((IEditorSite)activeEditorPart.
//@B.1  getEditorSite()).getActionBars();
   IAction undo = actionBars.getGlobalActionHandler
     (ITextEditorActionConstants.UNDO);
//@B.1 IAction undo = getAction((ITextEditor) part,
//@B.1   ITextEditorActionConstants.UNDO)
   IAction redo = actionBars.getGlobalActionHandler
     (ITextEditorActionConstants.REDO);
//@B.1 IAction undo = getAction((ITextEditor) part,
//@B.1    ITextEditorActionConstants.UNDO)
   actionBars.setGlobalActionHandler
     (ITextEditorActionConstants.UNDO, undo)
   actionBars.setGlobalActionHandler
     (ITextEditorActionConstants.REDO, redo)
} }
```

Figure 4: Cluster 2, 4: more alternative features