Automatic Generation of Natural Language Summaries for Java Classes

Laura Moreno*  Andrian Marcus  Lori Pollock  K. Vijay-Shanker
Giriprasad Sridhara  Jairo Aponte
Understanding Code
Happy Developers

Yay, good comments!!

```java
public class Kernel {

    /** Unique instance of Kernel. To access Kernel, Kernel.getInstance() */
    private static Kernel instance;

    /** Defines if aTunes is running in debug mode */
    public static boolean DEBUG;

    /** Logger */
    private Logger logger = Logger.getLogger(Kernel.class);

    /** Application State of aTunes */
    public ApplicationState state;

    /** Constructor of Kernel */
    private Kernel() {
        // Static method to create the Kernel instance.
        public static void startKernel() {

            /** Creates all objects of aTunes: visual objects, controllers, an */
            private void startCreation() {

                /** Starts visual objects */
                private void startVisualization() {
```
Not So Happy Developers
Yay, comments again!!

- Automatically generate natural-language summaries of source code
A Solution

Yay, comments again!!

- Existing approaches on code summarization
  - Methods
  - Code fragments
  - Code changes
    - ...

A (tag File) extension for audio files. This entity class consists mostly of getters to the audio file's state.

- It provides access to:
  - <ul>
    - <li>tag</li>
    - <li>track number</li>
    - <li>bitrate</li>
    - <li>album</li>
    - <li>title</li>
    - <li>title or file name</li>
    - <li>frequency</li>
    - <li>genre</li>
    - <li>external pictures country</li>
    - <li>artist</li>
    - <li>duration and</li>
    - <li>external pictures array list</li>
  - </ul>

- It allows checking whether:
  - <ul>
    - <li>audio file has internal pictures</li>
  - </ul>

```java
public class AudioFile extends File {
  private Tag tag;
  private ArrayList externalPictures;
  private long duration;
  private int bitrate;
  ...
```
Our Approach

- Automatically generate natural-language summaries of Java classes

Yay, comments again!!
Challenges when Summarizing Classes

• Classes bundle together more than just methods

• Adding together all method descriptions would result in very large summaries

• Not all methods are equally relevant for describing the behavior
Our Approach

• Automatically generated class summaries
  – Support quick understanding of a class
  – General purpose
  – Independent of external knowledge

<table>
<thead>
<tr>
<th>Indicative</th>
<th>Brief description of the class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstractive</td>
<td>Info not explicit in the class</td>
</tr>
<tr>
<td>Generic</td>
<td>Important info of the class</td>
</tr>
</tbody>
</table>
Summarizing Classes

What information to include in the summaries?

How much information to include in the summaries?

How to generate and present the summaries?
What information to include in the summaries?

- **Generic responsibilities – domain independent**
  - E.g., a data class encapsulates only get and set methods

- **Specific responsibilities – domain dependent**
  - E.g., a class that represents a Playlist
What information to include in the summaries?

• Generic responsibilities – domain independent
  – Class stereotypes [Dragan et al., ICSM’10]
    • E.g., data class, entity, controller, boundary, etc.
Class stereotypes

[Dragan et al., ICSM’10]
[Moreno et al., ASE’12]
Class stereotypes

Class stereotypes taxonomy

Entity

Keeper of the data model and/or business logic

public class CdRipper {
    private static final String ARTIST_PATTERN = "%A";
    private static final String ALBUM_PATTERN = "%L";
    private static final String TRACK_TITLE_AND_NUMBER = "%TN";
    private CdRipper()
    public CDInfo getCDInfo()
    public boolean ripTracks(ArrayList<Integer> tracks,
    private String getFileName(ArrayList<String> titles,

[Dragan et al., ICSM’10]
[Moreno et al., ASE’12]
Class stereotypes

Entity
Minimal entity
Data provider
Commander
Boundary
Factory
Controller
Pure controller
Large class
Lazy class
Degenerate
Data class
Pool

[Dragan et al., ICSM’10]
[Moreno et al., ASE’12]
What information to include in the summaries?

• Specific responsibilities – domain dependent
  – Identifiers of methods and data members

```java
private Cdda2wav cddawav;
private Encoder encoder;
private ProgressListener listener;

public CDInfo getCDInfo() {
}

public boolean ripTracks(ArrayList<Integer> tracks,)

private String getFileNames(ArrayList<String> titles, 

private static String getStringFormatted(String s) {
```
How much information to include in the summaries?

• Methods and attributes relevant to the class
  – Class stereotypes [Dragan et al., ICSM’10]
  – Method stereotypes [Dragan et al., ICSM’06]

• Access-level heuristics
  – Private, protected, package-protected, public
How to present and generate the summaries?

• Summary structure having four parts

  /**
   * General description based on the interfaces, superclass
   * and/or the stereotypes of the class
   * Description of the structure based on the stereotype
   * Description of the behavior based on the selected methods
   * List of inner classes
   */

• Natural language phrases for variables and program statements [Sridhara et al., ASE’10]
Class Summary

org.argouml.ui.explorer.ExplorerTreeModel

A TreeModelUMLEventListener, ItemListener implementation, and DefaultTreeModel extension for explorer tree models. This class controls external objects, i.e., the majority of its methods are controllers or object creators.

It allows managing:
- children; and
- structure changed.

It also allows:
- handling model element changed;
- removing node from parent;
- handling model element added;
- handling model element removed; and
- inserting node into new child mutable tree node.

This class declares the helper class ExplorerUpdater.
Class Summary

```java
public class ExplorerTreeModel extends DefaultTreeModel
    implements TreeModelUIModelListener, ItemListener
{
    private Object[] rules;
    private Map modelElementMap;
    private Comparator order;
    private Vector updatingChildren = new Vector();
    private ExplorerUpdater nodeUpdater = new ExplorerUpdater();

    private ExplorerTree tree;

    class ExplorerUpdater implements Runnable {
        public ExplorerTreeModel(Object root, ExplorerTree model) {
            ...}

        public void modelElementChanged(Object node) { ...}

        public void modelElementAdded(Object node) { ...}

        private void traverseModified(TreeNode start, Object node) {
            ...}

        public void modelElementRemoved(Object node) { ...}

        public void structureChanged() { ...}

        public void updateChildren(TreePath path) { ...}

        private Vector reorderChildren(ExplorerTreeNode node) { ...}

        private void collectChildren(Object modelElement, List newChildren) { ...}
    }
}
```
JSummarizer
An Automatic Generator of NL Summaries for Java Classes

Java class

Stereotype identification
- Structural attributes extraction
- Method stereotype identification
- Class stereotype identification

Content Selection
- Stereotype-based filter
- Access-level filter

Text Generation
- NL phrase generation
- Summary construction

Class summary
Evaluating the Summaries

Adequate
• All the relevant info

Concise
• Only necessary info of the class

Expressive
• Readable and understandable
Potential users evaluating automatically generated summaries

- 22 programmers

Summaries of 40 Java classes – 2 systems

- aTunes: audio player and manager
- argoUML: UML modeling tool

Each programmer evaluated the summary of 6 classes

- 3 classes of each system
- Different stereotypes
- 132 evaluations
Evaluated Properties

Content adequacy
- Considering only the content of the description, do you think that it
  - Not missing any information
  - Missing some information *
  - Missing some very important information *

Conciseness
- Considering only the content of the description, do you think that it
  - Has no unnecessary information
  - Has some unnecessary information *
  - Has a lot of unnecessary information *

Expressiveness
- Considering only the way the description is presented, do you think that it
  - Is easy to read and understand
  - Is somewhat readable and understandable
  - Is hard to read and understand
Results – Conciseness

- Has no unnecessary information: 61%
- Has some unnecessary information: 36%
- Has a lot of unnecessary information: 4%
Results - Expressiveness

- 67% Is easy to read and understand
- 30% Is somewhat readable and understandable
- 4% Is hard to read and understand
Results - Content adequacy

- Not missing any information: 43%
- Missing some information: 26%
- Missing some very important information: 31%
Lessons Learned

- The summaries are good 😊 but not perfect 😞

- Content adequacy
  - In few cases, the methods considered relevant by the developer are not necessarily relevant to the stereotype of the class
  - NL phrase generation misses some information from the method signatures

- Conciseness
  - Summaries with extraneous info – classes with no stereotype
  - Interaction between classes is useful, except when those classes belong to Java libraries
Long Story Short

• Novel approach to generate summaries for Java classes
  – General-purpose summaries
  – Completely automated technique
  – Does not make use of existing documentation or external domain knowledge

• Study with potential users evaluating 40 summaries of classes
  – 69% of the summaries miss little or no important info about the classes
  – 96% of the summaries are concise
  – 96% of the summaries are readable and understandable
On-Going Work

• Improvement of the summaries based on the results of the study
  – Augment the stereotype taxonomy
  – Enhance filtering process
  – Enrich the natural language generation techniques

• Generation of task-specific summaries
  – Extrinsic evaluation – summaries supporting specific maintenance tasks
Come to the tool demo!

http://www.cs.wayne.edu/~severe/jsummarizer/