Categorizing Software Applications for Maintenance

Collin McMillan\textsuperscript{1}
Mario Linares-Vásquez\textsuperscript{2}
Denys Poshyvanyk\textsuperscript{1}
Mark Grechanik\textsuperscript{3}

\textsuperscript{1}College of William & Mary
\textsuperscript{2}Universidad Nacional de Colombia
\textsuperscript{3}Accenture Technology Labs
Oceans of Code

• Programmers have created huge amounts of code

• How much code?
  – U.S. Bureau of Labor: 1.3m programmers in USA
  – Linux Journal Magazine poll: ~150 KLOC per programmer
  – ~195 billion LOC written in USA alone
    (comparison: ~650 billion sentences ever published)

What happens to all that code?
Oceans of Code

• Software Repositories are growing
  – SourceForge, 300k applications
  – FreeBSD Ports, 22k applications, **270 Million LOC**

• Corporate software development is also growing
  – Accenture, founded 1989, 250k employees
  – IBM, founded 1911, 425k employees
Oceans of (BUGGY) Code
Categorization is Useful
Categorization for Maintenance

• Software is more than Source Code
  – Binaries, Features, Bug Reports, etc.

• Domain analysis and Decision-Making
  – Are we maintaining unpopular features?
  – What differentiates our product from others?
  – Does similar software experience similar bugs?
How to Categorize?

• Manual Solutions
  – Self-reporting
  – Sorting / Cataloging

• Some problems
  – Legacy code
  – New categories
  – Number of applications labeled “other”

• An automated solution is desirable
The Categorization Game

• I will show you a fragment of code
• You have 15 seconds to categorize it

Text Editor  Web Browser  Music Player
import java.awt.event.*;
import javax.swing.*;
import javax.sound.midi.*;

/**
 * Illustrates general MIDI melody instruments and MIDI controllers.
 *
 * @version @(#)MidiSynth.java 1.15 99/12/03
 * @author Brian Lichtenwalter
 */

public class MidiSynth extends JPanel implements ControlContext {
    public void open() {
        try {
            if (synthesizer == null) {
                if ((synthesizer = MidiSystem.getSynthesizer()) == null) {
                    System.out.println("getSynthesizer() failed!");
                    return;
                }
            }
            synthesizer.open();
            sequencer = MidiSystem.getSequencer();
            sequence = new Sequence(Sequence.PPQ, 10);
        } catch (Exception ex) { ex.printStackTrace(); return; }

        Soundbank sb = synthesizer.getDefaultSoundbank();
        if (sb != null) {
            instruments = synthesizer.getDefaultSoundbank().getInstruments();
            synthesizer.loadInstrument(instruments[0]);
        }
        MidiChannel midiChannels[] = synthesizer.getChannels();
Done!

• Who thinks the code was from a text editor?
  MIDI music player

• We did not read the code

• We guessed based on the keyword clues
import java.awt.event.*;
import javax.swing.*;
import javax.sound.midi.*;

/**
 * Illustrates general MIDI melody instruments and MIDI controllers.
 * @version @(#)MidiSynth.java 1.15 99/12/03
 * @author Brian Lichtenwalter
 */

public class MidiSynth extends JPanel implements ControlContext {
    public void open() {
        try {
            if (synthesizer == null) {
                if ((synthesizer = MidiSystem.getSynthesizer()) == null) {
                    System.out.println("getSynthesizer() failed!");
                    return;
                }
            }
            synthesizer.open();
            sequencer = MidiSystem.getSequencer();
            sequence = new Sequence(Sequence.PPQ, 10);
        } catch (Exception ex) { ex.printStackTrace(); return; }

        Soundbank sb = synthesizer.getDefaultSoundbank();
        if (sb != null) {
            instruments = synthesizer.getDefaultSoundbank().getInstruments();
            synthesizer.loadInstrument(instruments[0]);
        }
        MidiChannel midiChannels[] = synthesizer.getChannels();
State-of-the-Art

• Categorize based purely on the keywords from source code

• Keywords as attributes for machine learning and classification

Relies on Source Code as Text
Machine Learning Approaches

Binary

Multiclass

“Winter is here.”

1Guillaume Obozinski, “Multi-Class and Structured Classification”
Multiclass composed of binary classifiers

Application

Music Player

NOT Music Player

Web Browser

Text Editor
Problem:
Source Code is not always available

• Question of Ownership
Problem: Source Code is not always available

- **Client** owns the Source Code
Our Solution

• Use only **API calls** from binaries as attributes

• API calls can be extracted from binaries as dependencies

• API calls define critical functionality
APIs Appear Everywhere

Example API package:

`com.sun.java_cup.internal`

Used over **3000** times in **600 of 8000** different applications from Sourceforge.
import java.awt.event.*;
import javax.swing.*;
import javax.sound.midi.*;

/**
* Illustrates general MIDI melody instruments and MIDI controllers.
* @version @(#)MidiSynth.java 1.15 99/12/03
* @author Brian Lichtenwalter
*/
public class MidiSynth extends JPanel implements ControlContext {
    public void open() {
        try {
            if (synthesizer == null) {
                if ((synthesizer = MidiSystem.getSynthesizer()) == null) {
                    System.out.println("getSynthesizer() failed!");
                    return;
                }
            }
            synthesizer.open();
            sequencer = MidiSystem.getSequencer();
            sequence = new Sequence(Sequence.PPQ, 10);
        } catch (Exception ex) { ex.printStackTrace(); return; }

        Soundbank sb = synthesizer.getDefaultSoundbank();
        if (sb != null) {
            instruments = synthesizer.getDefaultSoundbank().getInstruments();
            synthesizer.loadInstrument(instruments[0]);
        }
        MidiChannel midiChannels[] = synthesizer.getChannels();
Two API-based Attributes

`javax.sound.midi.MidiSystem.getMidiDevice()`

- **Classes**
  - `javax.sound.midi.MidiSystem`

- **Packages**
  - `javax.sound.midi`
Cross Validation Experiment

Software Repository

Training Set

Application Extractor

Category Extractor

Attribute Extractor

Machine Learning Algorithm

Testing Set

Prediction Model

Category Predictions
Key Design Questions

• Which **Machine Learning Algorithm** to use?
  – Support Vector Machines (SVM)
  – Decision Trees
  – Naïve Bayesian

• Which **Attributes** to select?
  – Terms
  – API calls
# Different Configurations

<table>
<thead>
<tr>
<th></th>
<th>State-of-the-Art</th>
<th>Our Work</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Attributes</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Terms</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>API Classes</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>API Packages</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td><strong>ML Algorithms</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Decision Trees</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Naïve Bayes</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td><strong>Number of Apps</strong></td>
<td>1683</td>
<td>4031</td>
</tr>
</tbody>
</table>
# Software Repositories

**SourceForge** *(3,286 apps)*

<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
<th>Category</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bio-Informatics</td>
<td>323</td>
<td>Indexing</td>
<td>329</td>
</tr>
<tr>
<td>Chat</td>
<td>504</td>
<td>Internet</td>
<td>1061</td>
</tr>
<tr>
<td>Communication</td>
<td>699</td>
<td>Interpreters</td>
<td>303</td>
</tr>
<tr>
<td>Compilers</td>
<td>309</td>
<td>Mathematics</td>
<td>373</td>
</tr>
<tr>
<td>Database</td>
<td>988</td>
<td>Networking</td>
<td>360</td>
</tr>
<tr>
<td>Education</td>
<td>775</td>
<td>Office</td>
<td>522</td>
</tr>
<tr>
<td>Email</td>
<td>366</td>
<td>Scientific</td>
<td>326</td>
</tr>
<tr>
<td>Frameworks</td>
<td>1115</td>
<td>Security</td>
<td>349</td>
</tr>
<tr>
<td>Front-Ends</td>
<td>584</td>
<td>Testing</td>
<td>907</td>
</tr>
<tr>
<td>Games</td>
<td>607</td>
<td>Visualization</td>
<td>456</td>
</tr>
<tr>
<td>Graphics</td>
<td>313</td>
<td>Web</td>
<td>534</td>
</tr>
</tbody>
</table>

**ShareJar** *(745 apps)*

<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chat &amp; SMS</td>
<td>320</td>
</tr>
<tr>
<td>Dictionaries</td>
<td>30</td>
</tr>
<tr>
<td>Education</td>
<td>90</td>
</tr>
<tr>
<td>Free Time</td>
<td>120</td>
</tr>
<tr>
<td>Internet</td>
<td>180</td>
</tr>
<tr>
<td>Localization</td>
<td>20</td>
</tr>
<tr>
<td>Messengers</td>
<td>50</td>
</tr>
<tr>
<td>Music</td>
<td>50</td>
</tr>
<tr>
<td>Science</td>
<td>20</td>
</tr>
<tr>
<td>Utilities</td>
<td>190</td>
</tr>
<tr>
<td>Emulators</td>
<td>30</td>
</tr>
<tr>
<td>Programming</td>
<td>10</td>
</tr>
<tr>
<td>Sports</td>
<td>40</td>
</tr>
</tbody>
</table>
Research Questions

RQ₁ Which machine learning algorithm is most effective for software categorization?

RQ₂ Which level of API granularity, classes or packages, is more effective for categorization?

RQ₃ Are the API classes or API packages as effective as words from source code for categorization?
Evaluation Metrics

• True Positive Rate
  – Proportion of correct links that were found
  – Analogous to Recall

• False Positive Rate
  – Proportion of incorrect links that were found
  – Analogous to Fall-Out
RQ$_1$: Machine Learning Algorithms

SVM outperforms DT and NB.
RQ₂: API Classes vs. Packages

API packages outperforms API classes.
RQ$_3$: API Packages vs. All Terms

**True Positive Rate**

- API packages
- Terms

**False Positive Rate**

- API packages
- Terms

API packages performs nearly as well as Terms.
Statistical Tests

• Friedman Test with Nemenyi’s Post-Hoc Procedure
  
  $H_0$ There is no statistically-significant difference between the TPR of SVM and DT.
  $H_1$ There is no statistically-significant difference between the TPR of SVM and NB.
  $H_2$ There is no statistically-significant difference between the FPR of SVM and DT.
  $H_3$ There is no statistically-significant difference between the FPR of SVM and NB.

<table>
<thead>
<tr>
<th>$H$</th>
<th>$q_{critical}$</th>
<th>$q_{observed}$</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H_0$</td>
<td>26.59</td>
<td>140.5</td>
<td>Reject</td>
</tr>
<tr>
<td>$H_1$</td>
<td>26.59</td>
<td>132.5</td>
<td>Reject</td>
</tr>
<tr>
<td>$H_2$</td>
<td>26.59</td>
<td>141.5</td>
<td>Reject</td>
</tr>
<tr>
<td>$H_3$</td>
<td>26.59</td>
<td>118.0</td>
<td>Reject</td>
</tr>
</tbody>
</table>
Anecdotal Example

Top term, API class, and API package in Email category of Sourceforge.

<table>
<thead>
<tr>
<th>Type of Feature</th>
<th>Feature</th>
<th>Apps in Category with Feature</th>
<th>Total Apps with Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Term</td>
<td>replyto</td>
<td>8</td>
<td>33</td>
</tr>
<tr>
<td>Package</td>
<td>sun.net.www</td>
<td>8</td>
<td>300</td>
</tr>
<tr>
<td>Class</td>
<td>com.sun.jlex.internal.CEmit</td>
<td>8</td>
<td>300</td>
</tr>
</tbody>
</table>
Conclusions

• We present an approach for software categorization

• Our approach categorizes using API calls

• We replicated a state-of-the-art study and showed:
  – SVM is the best of three selected ML algorithms
  – API packages outperform API classes as attributes
  – API packages perform as well as terms for categorization

• Our approach does not rely on source code

http://www.cs.wm.edu/semeru/catml/