Evolution-aware API analysis of developer skills

Hakan Aksu

hakanaksu@uni-koblenz.de
Content

• Motivation
• Research Question
• Metrics
• Methodology
• Results
• Threats to Validity / Future Work
Motivation
Motivation

http://de.dreamstime.com/
Motivation
Motivation

Software-Repository
Java
SVN

API-Analysis

http://de.dreamstime.com/
Research Question

1. How can we semi-automatically obtain API-related developer expertise from the analysis of existing software projects with version history using MSR methods?
   1. How can we measure API-related developer expertise?
   2. What is the important information from the version history to get the API-related developer expertise?
   3. How can we create a system to obtain semi-automatically API-related developer expertise?

2. How can we create a connection between changed lines of code and the API without building the underlying project?
Metrics

- CT
  Commit Time
- #VER, #CF and #CL
  Number of versions (=commits), changed files, changed lines
- #NJCF
  Number of non-java changed files
- #QVER*, #QCF* und #QCL*
  Number of qualified versions, qualified changed files, qualified changed lines
- #UAE
  Usage of API elements
- #AQCL
  Number of API usage in qualified changed lines
- #DQCL
  Number of programming domain usage in qualified changed lines

  - Version are “qualified” if...
    - developer is not null
    - changed files are existing
    - It is not a reorganization
Methodology

Criteria for a suitable software project

• Some Developer
• Some changes on source code
• Some API usage

<table>
<thead>
<tr>
<th>JHotDraw</th>
<th>2000-today</th>
</tr>
</thead>
<tbody>
<tr>
<td># versions</td>
<td>800</td>
</tr>
<tr>
<td># changed java-files</td>
<td>~17 000</td>
</tr>
<tr>
<td># changed lines</td>
<td>~650 000</td>
</tr>
<tr>
<td># developer</td>
<td>11</td>
</tr>
</tbody>
</table>
Methodology

• Combine information of a **software repository** and an **API** in one Model
  → In which changed line is used an API element?
  → Which API is used in changed lines?
  → Data Model
Methodology

Data Model
Methodology
Methodology
Extractor Process

• Step1: Extract the information of the software repository using SVNRepositoryExtractor (and PackageNameExtractor)

• Step2: Proof and Reorganize the package names

• Step3: Search for APIs

• Step4: Extract the information of the APIs using ApiExtractor
Methodology
Extractor Process

• Step 5: Assign packages to domains
• Step 6: Extract the relation-data between \textit{ChangedLine} and API elements using \texttt{ApiUsageExtractor}
• Step 7: Extract the values for the metrics using \texttt{MetricExtractor}
Methodology
Extractor Process – Step 1

• Extract the information of the software repository

→Demo SVNRepositoryExtractor
(and PackageNameExtractor)
Methodology
Extractor Process – Step 2

• Proof and Reorganize the package names
  1. Remove the keyword „Static“
  2. Eliminate internal packages
  3. Eliminate subfolders
Methodology
Extractor Process – Step 2

- Remove the keyword „static“
Methodology
Extractor Process – Step 2

• All imports with prefix
  – org.jhotdraw...
  – CH.ifa...

are internal packages and can be eliminated
Methodology
Extractor Process – Step 2

• Eliminate subfolders

```java
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
```

Java.util.logging
Java.util.regex
Methodology
Extractor Process – Step 3

Search for APIs
Jarfinder.com
Maven.org
Methodology
Extractor Process – Step 4
• Extract the information of the APIs

→Demo ApiExtractor
Methodology
Extractor Process – Step 5

- Assign packages to programming domains

<table>
<thead>
<tr>
<th>ID</th>
<th>PACKAGENAME</th>
<th>DOMAIN_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>156</td>
<td>java.util.regex</td>
<td>14</td>
</tr>
<tr>
<td>158</td>
<td>javax.jnlp</td>
<td>17</td>
</tr>
<tr>
<td>167</td>
<td>com.sun.corba.se.spi.oa</td>
<td>6</td>
</tr>
<tr>
<td>173</td>
<td>sun.awt.geom</td>
<td>8</td>
</tr>
<tr>
<td>175</td>
<td>sun.swing</td>
<td>8</td>
</tr>
<tr>
<td>199</td>
<td>java.math</td>
<td>11</td>
</tr>
<tr>
<td>206</td>
<td>edu.umd.cs.findbugs.annotations</td>
<td>12</td>
</tr>
<tr>
<td>207</td>
<td>javax.annotation</td>
<td>12</td>
</tr>
<tr>
<td>208</td>
<td>sun.reflect.generics.reflectiveObjects</td>
<td>12</td>
</tr>
<tr>
<td>210</td>
<td>javafx</td>
<td>8</td>
</tr>
</tbody>
</table>

Total $\rightarrow$ 47 APIs
Methodology
Extractor Process – Step 6

• Extract the relation-data between ChangedLine and API elements

Iterate over changed lines
  – Take API elements, which are imported in the changed file
  – Tokenize the changed line
  – Iterate over tokens and search for API elements

→ Demo ApiUsageExtractor
<table>
<thead>
<tr>
<th>Lexical</th>
<th>Syntactical</th>
<th>Buildable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Split source code to lexical units and make the analysis on it.</td>
<td>e.g. Parser uses a syntactical analysis</td>
<td>The hole project is needed → The name resolution happens in the building process</td>
</tr>
<tr>
<td>System.out.println(“HelloWorld!”)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[System, out, println, Hello, World]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
sub x(a,b,c) {
  i = a
  while (i < 20) {
    if (odd(i) {
      i = i + b
    } else {
      i = 1 + c
      print("in else")
    }
    print("in loop")
  } // end-while
  print(i)
}
```


| ✔ | ✗ | ✗ |
Validation numbers on the lexical approach

<table>
<thead>
<tr>
<th></th>
<th>classifier</th>
<th>method names</th>
<th>enum constants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total association to tokens</td>
<td>52982</td>
<td>61309</td>
<td>1033</td>
</tr>
<tr>
<td>Clearly association to tokens</td>
<td>52521</td>
<td>23988</td>
<td>1018</td>
</tr>
<tr>
<td>percentage</td>
<td>99,1%</td>
<td>39,1%</td>
<td>98,5%</td>
</tr>
</tbody>
</table>
Methodology
Extractor Process – Step 7

• Extract the values for the metrics

→ Demo MetricExtractor
Results - CT

Commit Time
Results - CT

• How long has the developer worked on the software project? (time difference)
• How often has the developer committed? (#Commit Time = #Version)
• How is the commit frequency relative to the other developers?
## Results

<table>
<thead>
<tr>
<th>DeveloperID</th>
<th>#VER</th>
<th>#CF</th>
<th>#CL</th>
<th>#NJCF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>36</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>88</td>
<td>9495</td>
<td>52</td>
</tr>
<tr>
<td>3</td>
<td>90</td>
<td>916</td>
<td>20942</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>6</td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>14</td>
<td>308</td>
<td>19856</td>
<td>239</td>
</tr>
<tr>
<td>6</td>
<td>83</td>
<td>4507</td>
<td>279520</td>
<td>1624</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>452</td>
<td>2719</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>19</td>
<td>280</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>13</td>
<td>26</td>
<td>638</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>517</td>
<td>10070</td>
<td>298191</td>
<td>2939</td>
</tr>
<tr>
<td>11</td>
<td>35</td>
<td>194</td>
<td>2482</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>27</td>
<td>392</td>
<td>46066</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DeveloperID</th>
<th>#QVER</th>
<th>#QCF</th>
<th>#QCL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>36</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>88</td>
<td>9495</td>
</tr>
<tr>
<td>3</td>
<td>90</td>
<td>916</td>
<td>20942</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>6</td>
<td>19</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>24</td>
<td>1782</td>
</tr>
<tr>
<td>6</td>
<td>41</td>
<td>336</td>
<td>5907</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>12</td>
<td>397</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>19</td>
<td>280</td>
</tr>
<tr>
<td>9</td>
<td>11</td>
<td>26</td>
<td>638</td>
</tr>
<tr>
<td>10</td>
<td>434</td>
<td>5992</td>
<td>198976</td>
</tr>
<tr>
<td>11</td>
<td>35</td>
<td>194</td>
<td>2482</td>
</tr>
</tbody>
</table>
## Results - #DQCL

<table>
<thead>
<tr>
<th>Developer</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>birdscurrybeer</td>
<td>1</td>
</tr>
<tr>
<td>cfml</td>
<td>2</td>
</tr>
<tr>
<td>dnoyeb</td>
<td>3</td>
</tr>
<tr>
<td>gesworthy</td>
<td>4</td>
</tr>
<tr>
<td>jeckel</td>
<td>5</td>
</tr>
<tr>
<td>mrfloppy</td>
<td>6</td>
</tr>
<tr>
<td>mtnygard</td>
<td>7</td>
</tr>
<tr>
<td>pleumann</td>
<td>8</td>
</tr>
<tr>
<td>pmorch</td>
<td>9</td>
</tr>
<tr>
<td>rawcoder</td>
<td>10</td>
</tr>
<tr>
<td>ricardo_padilha</td>
<td>11</td>
</tr>
<tr>
<td>null</td>
<td>12</td>
</tr>
</tbody>
</table>

### APIs Results

<table>
<thead>
<tr>
<th>APIs</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Achieving</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>79</td>
<td>0</td>
</tr>
<tr>
<td>Basics</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3704</td>
<td>0</td>
</tr>
<tr>
<td>Component</td>
<td>8</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8440</td>
<td>6</td>
</tr>
<tr>
<td>Concurrency</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>317</td>
<td>0</td>
</tr>
<tr>
<td>Configuration</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1686</td>
<td>0</td>
</tr>
<tr>
<td>Distribution</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>22</td>
<td>0</td>
<td>0</td>
<td>40</td>
<td>3304</td>
<td>0</td>
</tr>
<tr>
<td>Format</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2106</td>
<td>14</td>
</tr>
<tr>
<td>GUI</td>
<td>4</td>
<td>64</td>
<td>4580</td>
<td>3</td>
<td>568</td>
<td>828</td>
<td>82</td>
<td>26</td>
<td>169</td>
<td>65796</td>
<td>650</td>
</tr>
<tr>
<td>IO</td>
<td>0</td>
<td>20</td>
<td>752</td>
<td>0</td>
<td>14</td>
<td>189</td>
<td>61</td>
<td>2</td>
<td>10</td>
<td>10523</td>
<td>90</td>
</tr>
<tr>
<td>Logging</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>348</td>
<td>0</td>
</tr>
<tr>
<td>Math</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>Meta</td>
<td>0</td>
<td>50</td>
<td>1754</td>
<td>2</td>
<td>246</td>
<td>520</td>
<td>2</td>
<td>2</td>
<td>53</td>
<td>19450</td>
<td>204</td>
</tr>
<tr>
<td>Output</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>46</td>
<td>0</td>
</tr>
<tr>
<td>Parsing</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>88</td>
<td>0</td>
</tr>
<tr>
<td>Persistence</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>37</td>
<td>0</td>
</tr>
<tr>
<td>Testing</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>92</td>
<td>0</td>
</tr>
<tr>
<td>Web</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>217</td>
<td>0</td>
</tr>
<tr>
<td>XML</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1415</td>
<td>0</td>
</tr>
</tbody>
</table>
### Results - #UAE

<table>
<thead>
<tr>
<th>APIs</th>
<th>Classifier</th>
<th>Methods</th>
<th>Enum</th>
<th>Constants</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ch.randelshofer.quaqua</code></td>
<td>50 (597)</td>
<td>159 (1757)</td>
<td>20 (151)</td>
<td></td>
</tr>
<tr>
<td><code>com.sun.javafx</code></td>
<td>11 (1227)</td>
<td>75 (3143)</td>
<td>6 (309)</td>
<td></td>
</tr>
<tr>
<td><code>java.awt</code></td>
<td>175 (501)</td>
<td>589 (1723)</td>
<td>11 (69)</td>
<td></td>
</tr>
<tr>
<td><code>javafx</code></td>
<td>93 (2391)</td>
<td>161 (5526)</td>
<td>18 (760)</td>
<td></td>
</tr>
<tr>
<td><code>javax.swing</code></td>
<td>335 (1697)</td>
<td>906 (3372)</td>
<td>12 (48)</td>
<td></td>
</tr>
<tr>
<td><code>org.apache.batik.ext.awt</code></td>
<td>14 (237)</td>
<td>79 (740)</td>
<td>0 (0)</td>
<td></td>
</tr>
<tr>
<td><code>org.apache.batik.svggen</code></td>
<td>0 (92)</td>
<td>2 (268)</td>
<td>0 (0)</td>
<td></td>
</tr>
<tr>
<td><code>org.jdesktop.application</code></td>
<td>22 (81)</td>
<td>71 (439)</td>
<td>0 (3)</td>
<td></td>
</tr>
<tr>
<td><code>org.jdesktop.swingworker</code></td>
<td>2 (5)</td>
<td>6 (18)</td>
<td>3 (3)</td>
<td></td>
</tr>
<tr>
<td><code>sun.awt.geom</code></td>
<td>1 (20)</td>
<td>5 (103)</td>
<td>0 (0)</td>
<td></td>
</tr>
<tr>
<td><code>sun.swing</code></td>
<td>1 (116)</td>
<td>52 (753)</td>
<td>1 (7)</td>
<td></td>
</tr>
</tbody>
</table>
## Results - #AQCL

<table>
<thead>
<tr>
<th>APIs</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>ch.randelshofer.quaqua</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2806</td>
<td>0</td>
</tr>
<tr>
<td>com.sun.javafx</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>199</td>
<td>0</td>
</tr>
<tr>
<td>java.awt</td>
<td>6</td>
<td>69</td>
<td>5204</td>
<td>1</td>
<td>568</td>
<td>867</td>
<td>24</td>
<td>96</td>
<td>211</td>
<td>58139</td>
<td>598</td>
</tr>
<tr>
<td>javafx</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1443</td>
<td>0</td>
</tr>
<tr>
<td>javax.swing</td>
<td>10</td>
<td>5</td>
<td>1620</td>
<td>4</td>
<td>0</td>
<td>475</td>
<td>64</td>
<td>82</td>
<td>102</td>
<td>54687</td>
<td>328</td>
</tr>
<tr>
<td>org.apache.batik.ext.awt</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>706</td>
<td>0</td>
</tr>
<tr>
<td>org.apache.batik.svggen</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>org.jdesktop.application</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>388</td>
<td>0</td>
</tr>
<tr>
<td>org.jdesktop.swingworker</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>64</td>
<td>0</td>
</tr>
<tr>
<td>sun.awt.geom</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>21</td>
<td>0</td>
</tr>
<tr>
<td>sun.swing</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>80</td>
<td>0</td>
</tr>
</tbody>
</table>
Results

1.1. How can we measure API-related developer expertise?

→ Compare developer with each other using the metrics
1.2. What is the important information from the version history to get the API-related developer expertise?

→ Data Model
1.3. How can we create a system to obtain semi-automatically API-related developer expertise?
2. How can we create a connection between changed lines of code and the API without building the underlying project?

➔ Using an lexical approach!
   Tokenizing a line and searching for API elements
Threats to Validity / Future Work

• No measurement scale -> we compare the developer
• We always use the newest versions of APIs from Maven, findjar,...
• We need more software project for our analysis
• Different repositories (E.g. Git, Mercurial) and languages (E.G. C++, Python, Ruby)