Recommending Source Code for Rapid Software Prototypes

Collin McMillan¹
Nikki Hariri²
Denys Poshvyvanyk¹
Jane Cleland-Huang²
Bamshad Mobasher²

¹College of William & Mary
²DePaul University
You own a Start-Up

In two weeks, you are meeting with this venture capitalist.

He might invest >$1 million in you.
Your Product

A new web-enabled device is on the market.

You can build the first and only web browser for it.

What do we do?
Study Existing Solutions

Feature 1
“browser tabs”

Feature 2
“search bar”

Feature 3
“image downloading”
How can we do that, too?

Feature 1
“browser tabs”

• APIs?
• Existing classes?
• Suitable algorithms?
Step One: Build a Prototype

• Initial version may be buggy, slow, etc.

• Prototypes often **thrown away**

• Known as *opportunistic programming*

Types of Prototypes

**Horizontal:** Locate all *features* to implement.

**Vertical:** *Implement* and *demonstrate* select features.
Which Features to Implement?

Recommend the Common Features
Extracting Features from a Software Repository

- Tabbed Browsing
- Print to PDF
- Search Bar
- Mip Map Dithering

- Email using Outlook
- Browse using Tabs
- Windows Integration
- Pop-up Blocker

Next: Finding Reusable Source Code

Features

• Tabbed Browsing
• Print to PDF
• Search Bar
• Mip Map Dithering
• Email using Outlook
• Windows Integration
• Pop-up Blocker

Source Code

Module A
Module B
...
Module C
Building Prototypes

Feature 1
“download manager”

Feature 2
“search bar”

Feature 3
“tabbed browsing”

Feature 1
“download manager for web browser”

OnDefaultWeb  springlo
~LobbyOptionsTab  springlo
OnBrowseEditor  springlo
OnBrowseWeb  springlo
OnApply  springlo
~MainOptionsTab  springlo
OnOpenGroupsTab  springlo
GetOptionsTab  springlo
prefs_helpers  xdvik
browser open data file  ethereal
selected ptree info answered cb  ethereal
midori web settings class init  midori
initialize items  xdvik
<table>
<thead>
<tr>
<th>Name</th>
<th>numcopies</th>
<th>MB downloaded</th>
<th>MB uploaded</th>
<th>status</th>
<th>% complete</th>
<th>Kb/s up</th>
<th>Kb/s down</th>
<th>ETA (s)</th>
<th>Fsize (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoastToCoastRemakeV2.s7d</td>
<td>not available</td>
<td>0</td>
<td>0</td>
<td>seeding</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>inf.s</td>
<td>4.350</td>
</tr>
<tr>
<td>DeltaGigueRevX.s7d</td>
<td>5.96690</td>
<td>48.8555</td>
<td>0</td>
<td>leeching</td>
<td>97.2726</td>
<td>341.722</td>
<td>12220</td>
<td>32.133</td>
<td></td>
</tr>
<tr>
<td>GreenCometBasic.s7d</td>
<td>7.98214</td>
<td>7.25</td>
<td>0</td>
<td>leeching</td>
<td>98.1913</td>
<td>25.5763</td>
<td>263</td>
<td>13.922</td>
<td></td>
</tr>
<tr>
<td>Islands_in_Van_10.s7z</td>
<td>not available</td>
<td>0</td>
<td>0</td>
<td>seeding</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>inf.s</td>
<td>9.912</td>
</tr>
<tr>
<td>TabulaV3(i).s7z</td>
<td>3.98913</td>
<td>6.25</td>
<td>0</td>
<td>leeching</td>
<td>99.9994</td>
<td>25.5505</td>
<td>664</td>
<td>22.840</td>
<td></td>
</tr>
</tbody>
</table>

Total Outgoing: 0.18 KB/s   Total Incoming: 393.45 KB/s   Status: connected
Feature 2
“web browser search bar”

web_search
start_search
open_in_browser
remote_command
WebBrowserV1_Create
WebBrowserV2_Create
midori_browser_set_property
openWebBrowser
misc_find_webbrowser
createWelcomePage
createMenu
browser_open_url
browser_open_url
entry_activate_event
ebview
start_search
remote_command
WebBrowserV2_Create
WebBrowserV2_Create
WebBrowser_V2_Create
remote_command
remote_command
remote_command
remote_command
remote_command
remote_command
ebview
ebview
pureadmin
ebview
wine-1.0
wine-1.0
midori
kadu
pureadmin
kadu
kadu
gimp-2.4
gimpshop
ebview
How do we create a prototype out of all this code?
Source Code of Mozilla Firefox 8.0:

~/firefox/src/browser/components/

- ..
- aboutfiles
- buildfiles
- certerrorfiles
- dirproviderfiles
- downloadsfiles
- feedsfiles
- migrationfiles
- placesfiles
- preferencesfiles
- privatebrowsingfiles
- safebrowsingfiles
- searchfiles
- sessionstorefiles
- shellfiles
- sidebarfiles
- tabviewfiles
- testfiles
- thumbnailsfiles

Feature 1
“download manager”

Feature 2
“search bar”

Feature 3
“tabbed browsing”
Highly coupled to other code.
Two Competing Goals

Domain Relevance vs. Low Coupling
MIDI music player for a small Java game. The music must play in the background and not interfere with the game play.

<table>
<thead>
<tr>
<th>ID</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>Music plays in the background</td>
</tr>
<tr>
<td>140</td>
<td>Sound supported.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>765</td>
<td>Support for MP3, WAV, MIDI</td>
</tr>
<tr>
<td>741</td>
<td>MIDI Learn for every parameter</td>
</tr>
<tr>
<td>744</td>
<td>Volume Normalization</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Project Name</th>
<th>Features</th>
<th>Package ID</th>
<th>Package Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>vyer</td>
<td>765, 28, 744</td>
<td>46153</td>
<td>works.lib.sound</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- MusicResourceLocator</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- SoundLibrary</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- SoundResourceLocator</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- SoundPlayer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- JavaMidiMusicPlayer</td>
</tr>
<tr>
<td>wisl</td>
<td>140</td>
<td>185732</td>
<td>info.wisl</td>
</tr>
<tr>
<td>orbas</td>
<td>741</td>
<td>92092</td>
<td>org.apache.orbas.stuli.threadpool</td>
</tr>
</tbody>
</table>
Our Solution: Set Coverage of Features

Set Coverage of Features

Repositories

“Tabbed Browsing”
“Search Bar”
“Download Manager”

Set Cover Algorithm

Modules

Recommendations

- Tabbed Browsing
- Search Bar
- Windows Integration

Module A

- Print to PDF
- Download Manager

Module B

Problem: Computing the Set Cover is NP-complete
Set Coverage: Our Approximation

- **Greedy** Approximation to Set Coverage
- Choose the module that is **least-expensive** to add
- Based on **Coupling Between Objects (CBO)**

Module A is less coupled than Module B. Module A or Module B?
Use PageRank to Estimate Cost of Adding the Modules.
Easiest to reuse: E

Hardest to reuse: D
Continue grouping nodes until all features are covered…
Putting it all together
Software Archive

Modules from Sourceforge / Merobase¹
• 241,655 Java packages
• 13,701 Java projects
• 440 Million Total Lines of Code

Features from Softpedia
• 1,135 Features
• 493,347 Feature Descriptions

Evaluation – User Study

To compare Prefab with Portfolio, a state-of-the-art source code search engine.

31 Java Programmers participated
- Students at William & Mary
- 14 had professional experience
- Average 4.8 years of programming knowledge
Remember our Goals

Domain Relevance

Low Coupling
State-of-the-art Search Engines

Domain Relevance

Low Coupling
State-of-the-art Search Engines

Feature 1
“download manager”

Feature 2
“search bar”

Feature 3
“tabbed browsing”

Type in your search terms to begin

What do we type here?
Enter a description of the product.

MIDI music player for a small Java game. The music must play in the background and not interfere with the game play.

Submit
Step 2

Select **features** relevant to the query.

<table>
<thead>
<tr>
<th>ID</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔️</td>
<td>28  Music plays in the background</td>
</tr>
<tr>
<td>✔️</td>
<td>140 Sound supported.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔️</td>
<td>765 Support for MP3, WAV, MIDI</td>
</tr>
<tr>
<td>✔️</td>
<td>741 MIDI Learn for every parameter</td>
</tr>
<tr>
<td>✔️</td>
<td>744 Volume Normalization</td>
</tr>
</tbody>
</table>
Step 3

Determine which features are implemented in each module.

<table>
<thead>
<tr>
<th>Project Name</th>
<th>Features</th>
<th>Package ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>vyger</td>
<td>765, 28, 744</td>
<td>4615</td>
</tr>
<tr>
<td></td>
<td>Graphic RPG MOD with AI; Maps with 2d Tiles; Editor to draw tile maps; Adding spells (such Time, Stop etc);</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Package Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>wotlas.libs.sound</td>
</tr>
<tr>
<td>MusicResourceLocator</td>
</tr>
<tr>
<td>SoundLibrary</td>
</tr>
<tr>
<td>SoundResourceLocator</td>
</tr>
<tr>
<td>SoundPlayer</td>
</tr>
<tr>
<td>JavaMidiMusicPlayer</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Project Name</th>
<th>Features</th>
<th>Package ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>wisl</td>
<td>140</td>
<td>185732</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Package Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>info.wisl</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Project Name</th>
<th>Features</th>
<th>Package ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>orbas</td>
<td>741</td>
<td>92092</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Package Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>org.huihoo.orbas.rtutil.threadpool</td>
</tr>
</tbody>
</table>

Participants evaluate which packages implement which features.
## Cross-Validation Design

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Group</th>
<th>Approach</th>
<th>Task Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>Portfolio</td>
<td>T1</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>Prefab</td>
<td>T2</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>Prefab</td>
<td>T3</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>Portfolio</td>
<td>T4</td>
</tr>
</tbody>
</table>
Features Covered Per Package

- **p**: $2 \cdot 10^{-4}$
- **F**: 13.4
- **$F_{crit}$**: 3.85

The diagram compares the features covered in the Portfolio and Prefab packages, showing quantiles and statistical measures.
Time Per Query (in Minutes)

<table>
<thead>
<tr>
<th></th>
<th>Portfolio</th>
<th>Prefab</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{F}_\text{crit}$</td>
<td>3.90</td>
<td></td>
</tr>
<tr>
<td>$p$</td>
<td>$2 \cdot 10^{-10}$</td>
<td></td>
</tr>
<tr>
<td>$\text{F}$</td>
<td>46.5</td>
<td></td>
</tr>
</tbody>
</table>
Conclusions

• Recommend source code **modules** that implement a **set of features**

• Combines two state-of-the-art approaches

• Evaluated in a **user study** with 31 programmers

• All data online:
