Managing Software Evolution Through Semantic History Slicing

Yi Li
University of Toronto

PhD Advisor: Marsha Chechik

ASE’17 Doctoral Symposium
Oct 30, 2017
Acknowledgements

Marsha Chechik  
U Toronto

Julia Rubin  
UBC

Chenguang Zhu  
U Texas, Austin
0.4.3

bazel-io released this on Dec 22, 2016 • 4966 commits to master since this release

Release 0.4.3 (2016-12-22)

Baseline: c645a45

Cherry picks:

- af878d0: Add coverage support for java test. (series 4/4 of open-sourcing coverage command for java test)
- 09b92a8: Rollback of commit 67b4d52.
- b11dd48: Fix bad bug with the parallel implementation of BinaryOperatorExpression. Turns out that ForkJoinTask#adapt(Callable) returns a ForkJoinTask whose Future#get on error throws a ExecutionException wrapping a RuntimeException wrapping the thrown checked exception from the callable. This is documented behavior [1] that I incorrectly didn't know about.
- 9012bf1: Fix scripts/packages/convert_changelog to read the changelog correctly
- 55c97bc: Release script: if master branch does not exist, fall back on origin/master

...
Release 0.4.3 (2016-12-22)

Baseline: c645a45

Cherry picks:

- **af878d0**: Add coverage support for java test. (series 4/4 of open-sourcing coverage command for java test)
- **b09b92a8**: Rollback of commit 67b4d52.
- **b11dd48**: Fix bad bug with the parallel implementation of BinaryOperatorExpression. Turns out that ForkJoinTask#adapt(Callable) returns a ForkJoinTask whose Future#get on error throws a RuntimeException wrapping a RuntimeException wrapping the thrown checked exception from the callable. This is documented behavior [1] that I incorrectly didn't know about.
- **9012bf1**: Fix scripts/packages/convert_changelog to read the changelog correctly
- **55c97bc**: Release script: if master branch does not exist, fall back on origin/master

...
Release 0.4.3 (2016-12-22)

Baseline: c645a45

Cherry picks:

- af878d0: Add coverage support for java test. (series 4/4 of open-sourcing coverage command for java test)
- 09b92a8: Rollback of commit 67b4d52.
- b11d48: Fix bad bug with the parallel implementation of BinaryOperatorExpression. Turns out that ForkJoinTask#adapt(Callable) returns a ForkJoinTask whose Future#get on error throws a RuntimeException wrapping a RuntimeException wrapping the thrown checked exception from the callable. This is documented behavior [1] that I incorrectly didn't know about.
- 9012bf1: Fix scripts/packages/convert_changelog to read the changelog correctly
- 55c97bc: Release script: if master branch does not exist, fall back on origin/master

...
Release 0.4.3 (2016-12-22)

Baseline: c645a45

Cherry picks:

- af878d0: Add coverage support for java test. (series 4/4 of open-sourcing coverage command for java test)
- 09b92a8: Rollback of commit 67b4d52.
- b11d48: Fix bad bug with the parallel implementation of BinaryOperatorExpression. Turns out that ForkJoinTask#adapt(Callable) returns a ForkJoinTask whose Future#get on error throws a ExecutionException wrapping a RuntimeException wrapping the thrown checked exception from the callable. This is documented behavior [1] that I incorrectly didn't know about.
- 9012bf1: Fix scripts/packages/convert_changelog to read the changelog correctly
- 55c97bc: Release script: if master branch does not exist, fall back on origin/master
Pitfalls of Cherry-Picking

Dec 2, 2016

v0.4.2 release [0.4.2]

Add coverage support for java test. (series 4/4 of open-sourcing coverage command for java test)

Rollback of commit 67b4d52

Fix scripts/packages/convert_changelog to read the changelog correctly

Dec 22, 2016

master release [0.4.3]

Release script: if master branch does not exist, fall back on origin/master

Debian repository: override section and priority fields

58 authors
279 commits
479 files changed

https://github.com/bazelbuild/bazel/issues/2246
Pitfalls of Cherry-Picking

Dec 2, 2016

58 authors
279 commits
479 files changed

Dec 22, 2016

v0.4.2
release [0.4.2]

Add coverage support for java test. (series 4/4 of open-sourcing coverage command for java test)

Rollback of commit 67b4d52

Fix scripts/packages/convert_changelog to read the changelog correctly

... 

Release script: if master branch does not exist, fall back on origin/master

Debian repository: override section and priority fields

release [0.4.3]

https://github.com/bazelbuild/bazel/issues/2246
Pitfalls of Cherry-Picking

Creating release candidate by cherry-picking

Expert’s instructions...

Ok so far the remaining bug seems like just tests setup fixes. I think we can go forward with another RC with the following fixes: 4a75349 4a75349 55c97bc 39e5a46 4fb37bc
Pitfalls of Cherry-Picking

Failed due to merge conflicts
Pitfalls of Cherry-Picking

Failed due to missing commit(s)

Creating new candidate with additional cherry-pick \texttt{acbcbc2}

```
$ scripts/release/release.sh create 0.4.3 c645a45204b5ee467b8387b0487b8a5136ba6d06f a878d84990d9eb12
$ vim src/test/shell/bazel/bazel_coverage_test.sh
$ git add src/test/shell/bazel/bazel_coverage_test.sh
$ git cherry-pick --continue; exit
```

Created 0.4.3rc4 on branch release-0.4.3.

Pushed:

```
$ scripts/release/release.sh push
```
Pitfalls of Cherry-Picking

iirina commented 22 days ago

Should we also cherry-pick 4975760? It fixes #2247 [someone accidentally added a usage of Java8-only code and some users can't built bazel on a Linux system with JDK7]. The culprit is ca99bb7 which is included in the candidate, but the fix isn't. However we should actually release today since I'll be OOO starting tomorrow.

Failed due to missing commit(s)
Missing a fix this time.
Why is it so hard?
Why is it so hard?

Options?

1. Pick target commit(s)
2. Pick the entire history
3. Manually identify necessary commits
Why is it so hard?

Options?

1. Pick target commit(s)
2. Pick the entire history
3. Manually identify necessary commits
Why is it so hard?

Options?

1. Pick target commit(s)
2. Pick the entire history
3. Manually identify necessary commits

change
dependency
Why is it so hard?

Options?

1. Pick target commit(s)
2. Pick the entire history
3. Manually identify necessary commits

Existing version control tools:

• Code treated as plain texts
• Do not understand the semantics
• User provided semantic/logical grouping is inaccurate!
Challenges in Evolution Management

Gaps between individual changes and high-level system semantics:

• the history re-structuring challenge
• the change isolation and migration challenge
• the variability reverse engineering challenge
• and more …

This thesis aims to address some of them
What can we do?

Exploit existing artifacts:

• Strictly *structured data*

• Well-defined language *syntax* and *semantics*

• Carefully designed *test suites*
What can we do?

Exploit existing artifacts:

- Strictly *structured data*
- Well-defined language *syntax* and *semantics*
- Carefully designed *test suites*
Solution: Semantic History Slicing

Exploit existing artifacts:

- Strictly \textit{structured data}
- Well-defined language \textit{syntax} and \textit{semantics}
- Carefully designed \textit{test suites}

\begin{itemize}
  \item History: sequence of commits
  \item Criterion: set of tests
  \item Sub-history: well-formed: compiles & semantic preserving: passing tests
\end{itemize}
Solution: Semantic History Slicing

Exploit existing artifacts:

- Strictly *structured data*
- Well-defined language *syntax* and *semantics*
- Carefully designed *test suites*

**History:** sequence of commits +
**Criterion:** set of tests

**Sub-history:** well-formed: compiles &
semantic preserving: passing tests
Research Map

Problem Definition:

Semantic History Slicing

Techniques:

Static Slicing: CSLICER

Dynamic Slicing: DEFINER

Evolution Managing Framework: CSLICER-CLOUD

Validation:

Porting Functionalities
Creating Pull Requests
Feature Location

[ASE’15] [TSE’17] [SPLC’17]

Bug Localization

[ASE’15] [ASE’16]
CSLICER-CLOUD

<table>
<thead>
<tr>
<th>Test cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>LexerTest#testCR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definer</td>
</tr>
</tbody>
</table>

Submit

### History View

- **1 BRANCHES**

#### Master

- **[CSV-214] Adding a placeholder in the Lexer and CSV parser to store the...**
  - **Gary Gregory** Fr 2017-08-11
  - **aae6f90**

- **Javadoc.**
  - **Gary Gregory** Th 2017-08-10
  - **4d0f226**

- **Add default maven default goal (clean, test, clirr, rat and javadoc) and ru...**
  - **pascalschuma** Tu 2017-08-01
  - **bbf3ebe**

- **Add test data files "optd_por_public.csv" and "999751170.patch.csv" to ru...**
  - **pascalschuma** Tu 2017-08-01
  - **fb03b65**

- **JiraCsv203Test and JiraCsv213Test: add missing license header**
  - **pascalschuma** Tu 2017-08-01
  - **fe5cf5c**

Validation

A dataset for semantic changes in version histories [MSR’17]

• 98 items of semantic change data
• Collected from 10 open-source Java projects
• Ground-truth obtained from developer’s documentation and brute-force minimization
• Available at: https://github.com/Chenguang-Zhu/DoSC

DoSC
Feature Location for SPLE

The “top-down” approach

core assets (features)

configurations + feature model

product outputs
Feature Location for SPL
Feature Location for SPLA

The “top-down” approach

1. feature implementations (core assets)

The “bottom-up” approach

2. feature relationships (feature models)

From “ad-hoc” to “systematic”
Feature Location in Version Histories
Feature Location in Version Histories

New features: \{f_1, f_2, f_3, f_4\}, tests: \{t_1, t_2, t_3, t_4\}
Feature Location in Version Histories

New features: \{f_1, f_2, f_3, f_4\}, tests: \{t_1, t_2, t_3, t_4\}

- f_1:
- f_2:
- f_3:
- f_4:
Feature Location in Version Histories

New features: \{f_1, f_2, f_3, f_4\}, tests: \{t_1, t_2, t_3, t_4\}

 masters

 feature 1

 feature 4

 feature 2

 feature 3

New features: \{f_1, f_2, f_3, f_4\}, tests: \{t_1, t_2, t_3, t_4\}
Status and Next Steps

Current Status (a few months before graduation)

• Wrapping up thesis work
• Preparing a journal paper summarizing this line of work

Future Plans (comments and discussions please!)

• Deep integration with development tool chains
  - semantics-aware cherry-picking
• Leveraging the social aspect of software development
  - developer conversations
  - log messages
• User studies
Related Work

History Understanding and Manipulation

- History transformation [Muslu et al.][Servant & Jones]
- Change classifications [Falleri et al.]

Change Impact Analysis

- Chianti [Ren et al.], FaultTracer [Zhang et al.]
- Ekstazi [Gligoric et al.]

Bug Localization

- Delta debugging [Zeller et al.], Selective Bisection Debugging [Saha & Gligoric]
- Information retrieval-based approaches [Wang & Lo][Saha et al.]

Feature Location

- Dynamic FL, Software Reconnaissance [Wilde & Scully]
- Managing cloned variants [Rubin et al.]
Summary

Semantic History Slicing:

- Bridging gaps between text changes and program semantics
- Many applications in evolution management
- Validation on open source Git repositories


