FHistorian: Locating Features in Version Histories

**Yi Li** / UToronto
Chenguang Zhu / UToronto
Julia Rubin / UBC
Marsha Chechik / UToronto

Sep 27, 2017
Feature Location

“Feature location is the activity of identifying an initial location in the source code that implements functionality in a software system.”

Feature Location for SPLE

The “top-down” approach

core assets (features)

configurations + feature model

product outputs
Feature Location for SPLE

The “top-down” approach

core assets (features)

configurations + feature model

product variants

The “bottom-up” approach

product outputs
Feature Location for SPLE

1. Feature implementations (assets)
   - f1: □ ●
   - f2: ▲ ■
   - f3: ★
   - f4: ● ● ◆ △

2. Feature relationships (feature models)
   - f4
   - f3
   - f1
   - f2

The “top-down” approach

- Core assets (features)
- Configurations + feature model
- Product outputs

The “bottom-up” approach

From “ad-hoc” to “systematic”
Feature Location from Product Variants

Variant 1: f1, f2, f3
Variant 2: f1, f3
Variant n-1: f1, f2, f4
Variant n: f1, f3, f5
Feature Location from Product Variants

Variant 1: \( f_1, f_2, f_3 \)

Variant 2: \( f_1, f_3 \)

Variant n-1: \( f_1, f_2, f_4 \)

Variant n: \( f_1, f_3, f_5 \)

code elements
Feature Location from Product Variants

Variant 1

Variant 2

Variant n-1

Variant n

f1, f2, f3

f1, f3

f1, f2, f4

f1, f3, f5

Intersection-based feature location

code elements
Feature Location from Product Variants

Variant 1

Variant 2

Variant n-1

Variant n

f1, f2, f3

f1, f3

f1, f2, f4

f1, f3, f5

Intersection-based feature location

code elements
Feature Location from Product Variants

Variant 1: \( f_1, f_2, f_3 \)

Variant 2: \( f_1, f_3 \)

Variant n-1: \( f_1, f_2, f_4 \)

Variant n: \( f_1, f_3, f_5 \)

Intersection-based feature location

code elements
Feature Location from Product Variants

Variants:
- Variant 1: $f_1, f_2, f_3$
- Variant 2: $f_1, f_3$
- Variant n-1: $f_1, f_2, f_4$
- Variant n: $f_1, f_3, f_5$

Intersection-based feature location:

- $f_1$: □ ○
- $f_2$: △ ■
- $f_3$: ★
- $f_4$: ◆ ◇ ▲ ▼
- $f_5$: ● ◆ ▲ ▼

Code elements:
Feature Location from Product Variants

What if: Variant 1 also has $f_6$ and $f_7$?
Intersection-based FL:

- Only works well with a large number of variants
- Operates in *static* manner
- Feature labeling has to be exhaustive
Pitfalls of Intersection-Based Approaches

Intersection-based FL:

• Only works well with a large number of variants

• Operates in static manner

• Feature labeling has to be exhaustive

Reality:

• 3~10 products, ~50 features

• Maintained in version control systems (e.g., Git)
Feature Location in Version Histories

master

feature 1

feature 4

feature 2

feature 3

test 1
test 4
test 2
test 3

v0.1

v1.0
Feature Location in Version Histories

New features: \{f1, f2, f3, f4\}, tests: \{t1, t2, t3, t4\}
Feature Location in Version Histories

New features: \{f1, f2, f3, f4\}, tests: \{t1, t2, t3, t4\}

master

feature 1

feature 2

feature 3

feature 4

test 1

test 2

test 3

test 4

New features: \{f1, f2, f3, f4\}, tests: \{t1, t2, t3, t4\}

commits
Feature Location in Version Histories

New features: \{f1, f2, f3, f4\}, tests: \{t1, t2, t3, t4\}
History-Based vs. Intersection-Based

*History*-based *dynamic* feature location
History-Based vs. Intersection-Based

**History**-based *dynamic* feature location

- More **flexible**:
  1. Implicit feature labeling: release notes
  2. Traceability of evolution information
  3. Effective even with limited numbers of variants
History-Based vs. Intersection-Based

**History-based** dynamic feature location

- More **flexible**:
  1. Implicit feature labeling: release notes
  2. Traceability of evolution information
  3. Effective even with limited numbers of variants

- More **accurate**:
  4. Captures runtime dependencies
  5. Focused search space: only considering changes within a history range
  6. Generates Light-weight feature models
Outline

1. Introduction

2. Background
   - Semantics-Preserving History Slice
   - Semantic History Slicing

3. FHistorian
   - FLocate: identifying feature implementations in histories
   - FHGraph: inferring feature relationships

4. Evaluation

5. Conclusion & Future Work
### Semantics-Preserving History Slice

<table>
<thead>
<tr>
<th>(H)istory</th>
<th>(T)ests</th>
<th>$H \models T$</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="https://via.placeholder.com/150" alt="Timeline Diagram" /></td>
<td>$T_1, T_2$</td>
<td>✔️</td>
</tr>
</tbody>
</table>
## Semantics-Preserving History Slice

<table>
<thead>
<tr>
<th>(H)istory</th>
<th>(T)ests</th>
<th>$H \models T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>![History Diagram]</td>
<td>(T_1, T_2)</td>
<td>✔</td>
</tr>
<tr>
<td>![History Diagram]</td>
<td>(T_1, T_2)</td>
<td>✗</td>
</tr>
</tbody>
</table>
Semantics-Preserving History Slice

<table>
<thead>
<tr>
<th>(H)istory</th>
<th>(T)ests</th>
<th>H ⊨ T</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="History 1" /></td>
<td>$T_1, T_2$</td>
<td>✓</td>
</tr>
<tr>
<td><img src="image2" alt="History 2" /></td>
<td>$T_1, T_2$</td>
<td>✗</td>
</tr>
<tr>
<td><img src="image3" alt="History 3" /></td>
<td>$T_1$</td>
<td>✓</td>
</tr>
</tbody>
</table>
# Semantics-Preserving History Slice

<table>
<thead>
<tr>
<th>(H)istory</th>
<th>(T)ests</th>
<th>H \models T</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="history1.png" alt="History 1" /></td>
<td>$T_1, T_2$</td>
<td>✓</td>
</tr>
<tr>
<td><img src="history2.png" alt="History 2" /></td>
<td>$T_1, T_2$</td>
<td>✗</td>
</tr>
<tr>
<td><img src="history3.png" alt="History 3" /></td>
<td>$T_1$</td>
<td>✓</td>
</tr>
<tr>
<td><img src="history4.png" alt="History 4" /></td>
<td>$T_2$</td>
<td>✓</td>
</tr>
</tbody>
</table>
Semantics-Preserving History Slice

<table>
<thead>
<tr>
<th>(H)istory</th>
<th>(T)ests</th>
<th>H ⊨ T</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="history1" alt="History" /></td>
<td>$T_1, T_2$</td>
<td>✓</td>
</tr>
<tr>
<td><img src="history2" alt="History" /></td>
<td>$T_1, T_2$</td>
<td>✗</td>
</tr>
<tr>
<td><img src="history3" alt="History" /></td>
<td>$T_1$</td>
<td>✓</td>
</tr>
<tr>
<td><img src="history4" alt="History" /></td>
<td>$T_2$</td>
<td>✓</td>
</tr>
</tbody>
</table>

Minimal semantics-preserving slice = feature implementing changes?
Semantic History Slicing

http://www.cs.toronto.edu/~liyi/cslicer [ASE’16]
Outline

1. Introduction

2. Background
   - Semantics-Preserving History Slice
   - Semantic History Slicing

3. FHistorian
   - FLocate: identifying feature implementations in histories
   - FHGraph: inferring feature relationships

4. Evaluation

5. Conclusion & Future Work
FHistorian = FLocate + FHGraph
FHistorian = FLocate + FHGraph
FLocate: Locating Feature Implementations

Based on Definer [ASE’16]

- Foreach feature \( f \), find a minimal slice: \( H_f \) s.t. \( H_f \models T_f \)

- Factoring out other features: \( f = H_f \setminus H_{f'} \) for all other \( f' \)

- Hunk minimization (details in paper…)

\[
\begin{align*}
\delta_1 & : i: \text{int } f1() \{ \text{return 1;} \} \\
\delta_2 & : j: \text{int } f2() \{ \text{return } f1() + 1; \} \\
\delta_3 & : k: \text{int } f3() \{ \text{return } f1() - 1; \}
\end{align*}
\]

\[
\begin{align*}
T_f_1 : f1() &= 1, T_f_2 : f2() &= 2, T_f_3 : f3() &= 0
\end{align*}
\]
Based on Definer [ASE’16]

- Foreach feature \( f \), find a minimal slice: \( H_f \) s.t. \( H_f \models T_f \)
- Factoring out other features: \( f = H_f \setminus H_{f'} \) for all other \( f' \)
- Hunk minimization (details in paper…)

**FLocate: Locating Feature Implementations**
FLocate: Locating Feature Implementations

Based on Definer [ASE’16]

- Foreach feature \( f \), find a minimal slice: \( H_f \) s.t. \( H_f \models T_f \)
- Factoring out other features: \( f = H_f \setminus H_{f'} \) for all other \( f' \)
- Hunk minimization (details in paper…)

\[
\begin{align*}
\delta_1 & : i:\text{int} \quad \boxed{f_1()} \{ \text{return 1;} \} \\
\delta_2 & : j:\text{int} \quad \boxed{f_2()} \{ \text{return } f_1() + 1; \} \\
\delta_3 & : k:\text{int} \quad \boxed{f_3()} \{ \text{return } f_1() - 1; \}
\end{align*}
\]

\[
\begin{align*}
T_{f_1} : f_1() & = 1, \\
T_{f_2} : f_2() & = 2, \\
T_{f_3} : f_3() & = 0
\end{align*}
\]
**FLocate: Locating Feature Implementations**

Based on Definer [ASE’16]

- Foreach feature $f$, find a minimal slice: $H_f$ s.t. $H_f \vDash T_f$
- Factoring out other features: $f = H_f \setminus H_{f'}$ for all other $f'$
- Hunk minimization (details in paper…)

\[
\begin{align*}
\delta_1 & : \mathbb{H} \ni \text{int } f1() \{ \text{return } 1; \} \\
\delta_2 & : \mathbb{H} \ni \text{int } f2() \{ \text{return } f1() + 1; \} \\
\delta_3 & : \mathbb{H} \ni \text{int } f3() \{ \text{return } f1() - 1; \}
\end{align*}
\]

\[
T_{f_1} : f1() == 1, T_{f_2} : f2() == 2, T_{f_3} : f3() == 0
\]

\[
\begin{align*}
H_f & : \text{red} - \text{blue} - \text{purple} \\
H_{f1} & : \text{red} \quad f_1 : \text{red} \\
H_{f2} & : \text{red} - \text{blue} \quad f_2 : \text{blue} \\
H_{f3} & : \text{red} - \text{purple} \quad f_3 : \text{purple}
\end{align*}
\]
FHGraph: Inferring Feature Relationships

**Light-weight feature model:**

- **Depends-on**
  \[(f_2 \rightarrow f_1) \iff (H_{f_1} \subseteq H_{f_2})\]

  *Reflecting runtime dependencies*

- **Relates-to**
  \[(f_2 \leftrightarrow f_1) \iff (H_{f_1} \cap H_{f_2} \neq \emptyset)\]

  *Revealing underlying connections*
FHGraph: Inferring Feature Relationships

Light-weight feature model:

- Depends-on
  \((f_2 \rightarrow f_1) \iff (H_{f_1} \subseteq H_{f_2})\)
  - Reflecting runtime dependencies

- Relates-to
  \((f_2 \leftrightarrow f_1) \iff (H_{f_1} \cap H_{f_2} \neq \emptyset)\)
  - Revealing underlying connections

\[ H_{f_2} : \]
\[ \bullet \longrightarrow \bullet \]
\[ \text{depends-on} \]

\[ H_{f_3} : \]
\[ \bullet \longrightarrow \bullet \]
\[ \text{depends-on} \]

\[ H_{f_1} : \]
\[ \bullet \]
FHGraph: Inferring Feature Relationships

Light-weight feature model:

- **Depends-on**
  
  \[(f_2 \rightarrow f_1) \iff (H_{f_1} \subseteq H_{f_2})\]

  Reflecting runtime dependencies

- **Relates-to**
  
  \[(f_2 \leftrightarrow f_1) \iff (H_{f_1} \cap H_{f_2} \neq \emptyset)\]

  Revealing underlying connections

\[H_{f_2} : \]

\[H_{f_3} : \]

\[H_{f_1} : \]
Outline

1. Introduction

2. Background
   - Semantics-Preserving History Slice
   - Semantic History Slicing

3. FHistorian
   - FLocate: identifying feature implementations in histories
   - FHGraph: inferring feature relationships

4. Evaluation

5. Conclusion & Future Work
Evaluation

FHistorian:

• Implementation: bitbucket.org/liyistc/gitslice
• Data set [MSR’17]: github.com/Chenguang-Zhu/DoSC

Research questions:

• How accurate are the feature location results?
• Are the inferred feature relationships useful?
Preparing subjects:

• Take a release history (ideally with JIRA issue tracking)
• Go through each feature (64)
• Identify feature tests (36)

<table>
<thead>
<tr>
<th>Project &amp; Release</th>
<th>Features</th>
<th>#New</th>
<th>#Tested</th>
</tr>
</thead>
<tbody>
<tr>
<td>commons-csv v1.3</td>
<td></td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>commons-compress v1.13</td>
<td></td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>commons-io v1.4</td>
<td></td>
<td>18</td>
<td>9</td>
</tr>
<tr>
<td>commons-io v2.2</td>
<td></td>
<td>15</td>
<td>7</td>
</tr>
<tr>
<td>commons-lang v3.4</td>
<td></td>
<td>17</td>
<td>10</td>
</tr>
</tbody>
</table>

New Feature \( \{ f_1, f_2, \ldots, f_n \} \)

- [MATH-814] - Kendalls Tau Implementation
- [MATH-851] - Add convolution
- [MATH-958] - Pareto distribution is missing
- [MATH-977] - Add Halton sequence generator
- [MATH-978] - StorelessCovariance to be map/reducible
- [MATH-987] - SimpleRegression needs to be map/reducible
Evaluation Subjects

Preparing subjects:

- Take a release history (ideally with JIRA issue tracking)
- Go through each feature (64)
- Identify feature tests (36)

<table>
<thead>
<tr>
<th>Project &amp; Release</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#New</td>
</tr>
<tr>
<td>commons-csv v1.3</td>
<td>7</td>
</tr>
<tr>
<td>commons-compress v1.13</td>
<td>7</td>
</tr>
<tr>
<td>commons-io v1.4</td>
<td>18</td>
</tr>
<tr>
<td>commons-io v2.2</td>
<td>15</td>
</tr>
<tr>
<td>commons-lang v3.4</td>
<td>17</td>
</tr>
</tbody>
</table>

New Feature \( \{ f_1, f_2, \ldots, f_n \} \)

- [MATH-814] - Kendalls Tau Implementation
- [MATH-851] - Add convolution
- [MATH-958] - Pareto distribution is missing
- [MATH-977] - Add Halton sequence generator
- [MATH-978] - StoreLessCovariance to be map/reducible
- [MATH-987] - SimpleRegression needs to be map/reducible

Release notes

Features

Commons Math / MATH-814
Kendalls Tau Implementation \( f_1 \)
Preparation of subjects:

- Take a release history (ideally with JIRA issue tracking)
- Go through each feature (64)
- Identify feature tests (36)

<table>
<thead>
<tr>
<th>Project &amp; Release</th>
<th>#New</th>
<th>#Tested</th>
</tr>
</thead>
<tbody>
<tr>
<td>commons-csv v1.3</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>commons-compress v1.13</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>commons-io v1.4</td>
<td>18</td>
<td>9</td>
</tr>
<tr>
<td>commons-io v2.2</td>
<td>15</td>
<td>7</td>
</tr>
<tr>
<td>commons-lang v3.4</td>
<td>17</td>
<td>10</td>
</tr>
</tbody>
</table>
Preparing subjects:

- Take a release history (ideally with JIRA issue tracking)
- Go through each feature (64)
- Identify feature tests (36)

<table>
<thead>
<tr>
<th>Project &amp; Release</th>
<th>Features</th>
<th>#New</th>
<th>#Tested</th>
</tr>
</thead>
<tbody>
<tr>
<td>commons-csv v1.3</td>
<td></td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>commons-compress v1.13</td>
<td></td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>commons-io v1.4</td>
<td></td>
<td>18</td>
<td>9</td>
</tr>
<tr>
<td>commons-io v2.2</td>
<td></td>
<td>15</td>
<td>7</td>
</tr>
<tr>
<td>commons-lang v3.4</td>
<td></td>
<td>17</td>
<td>10</td>
</tr>
</tbody>
</table>

New Feature $\{f_1, f_2, \ldots, f_n\}$

- [MATH-814] - Kendalls Tau Implementation
- [MATH-851] - Add convolution
- [MATH-958] - Pareto distribution is missing
- [MATH-977] - Add Halton sequence generator
- [MATH-978] - Storeless Covariance to be map/reducible
- [MATH-987] - SimpleRegression needs to be map/reducible

Evaluation Subjects

$E = \{f_1, f_2, \ldots, f_n\}$

Features $f_1$

Feature tests $T_{f_1}$

release notes

Commons Math / MATH-814
Kendalls Tau Implementation

[MATH-814] Added Kendalls tau correlation, Thanks to Matt Adereth.
Results

Comparing with developer annotations:

- **Ground truth**: extracted from change logs and release notes (not always perfect)
- Perfect match on 15/36 features
- Finding more changes, occasionally missing changes
Results

Comparing with developer annotations:

- **Ground truth**: extracted from change logs and release notes (not always perfect)
- Perfect match on 15/36 features
- Finding more changes, occasionally missing changes

Reasons for the differences:

- **Conceptual** vs. **operational**
- Missing minor optimizations: not affecting tests
- Discovering **hidden dependencies**
Results: Feature Relationships

relates-to:

A  B

depends-on:

A  B

COMPRESS 374

COMPRESS 369

COMPRESS 368

COMPRESS 327

COMPRESS 373

COMPRESS 360

COMPRESS 327'

COMPRESS 373

COMPRESS 369

COMPRESS 374

COMPRESS 368

Seekable
Results: Feature Relationships

relates-to:
A → B

depends-on:
A → B

COMPRESS 374
COMPRESS 369
COMPRESS 373
COMPRESS 368
COMPRESS 327

Seekable

COMPRESS 374
COMPRESS 369

Hidden feature

Hidden feature
Conclusion & Future Work

FHistorian: History-based feature location

• More *flexible* and more *accurate*
• Exploiting version control data
• Identifying feature implementations dynamically
• Inferring light-weight feature models

What’s next?

• Extracting feature meta information automatically
• Generating richer feature models
Questions?

New features: \{f1, f2, f3, f4\}, tests: \{t1, t2, t3, t4\}

Yi Li
University of Toronto
liyi@cs.toronto.edu