Similarity management of 'cloned and owned' variants

Thomas Schmorleiz and Ralf Lämmel
University of Koblenz-Landau
What is similarity management and why do we need it anyway?
Consider similarity across variants of a system

We see a force-driven visualization of the similarity for some variants. These happen to be the Haskell-based variants for the HRMS in our case study.
How to manage many variants?

- Clone and own and sync non-systematically
- Clone and own but **migrate to software product line** eventually
- Clone and own but **leverage ‘virtual platform’**
  - **Locate features** and maintain as metadata
  - **Detect clones** and maintain as metadata
  - **Propagate changes** automatically and manually
  - Possibly even **clone explicitly** with appropriate operators
The underlying notion (ICSE 2014)

Flexible Product Line Engineering with a Virtual Platform

Michał Antkiewicz, Wenbin Ji, Thorsten Berger, Krzysztof Czarnecki
University of Waterloo, Canada

Ștefan Stănciulescu, Andrzej Wąsowski
IT University of Copenhagen, Denmark

Thomas Schmorleiz, Ralf Lämmel
Universität Koblenz-Landau, Germany

Ina Schaefer
Technische Universität Braunschweig, Germany

Key related work

This work: An annotation-centric approach

- Annotations for **maintenance invariants and tasks**
- No feature location, just clone detection
- **Tools:**
  - Similarity analysis and tracking over timeline
  - UI for annotation and TODO
  - Change propagation
  - Version control integration
Central research question

How to measure similarity of clowned- and-owned variants and effects of similarity improvement in a useful manner?

... so that the developer understands
- what changes should be looked at;
- whether similarity is improving overall;
- ...
Process of similarity management

1. **Analysis** of similarity revealing clone genealogy
2. **Review** of detected similarities leading to *maintenance tasks*
3. **Improvement** of similarity — automatically or manually
4. **Reestablishment** of diverged *maintenance invariants*
5. **Commitment** of changes
Similarity analysis
Similarity analysis — key concepts

- **Variants** (such as folders or branches)
- **Fragment snapshots** (such as methods in a version)
- Tracked **fragments** (across versions)
- **Similarity snapshots** (between fragment snapshots)
- **Similarity evolutions**
  - Always Equal
  - Diverge from Equal
  - Always Non-equal
  - Converge to Equal
Similarity analysis — schema
Similarity annotations
Similarity annotations

- **Maintenance invariants**
  - Maintain Equality
  - Maintain Similarity
  - Ignore Similarity

- **Maintenance tasks**
  - Restore Equality
  - Establish Equality
  - Remove Equality
  - Restore Similarity
  - Increase Similarity

The evolution of variants must preserve invariants. This may necessitate **automatic** change propagation or **manual** intervention by the developer.

The developer may register these tasks or they may originate from incremental similarity analysis after pulling in changes.
Tool support
Overall commit history
Variants on the timeline

Variants throughout the history of the haskell repository.

The first view shows variants over time. On the horizontal axis we show all commits at which at least one variant was created, renamed, or deleted. On the vertical axis we show the variants, sorted by creation time. If a variant was touched at a given commit point we create a color-coded circle where green, gray and red indicate variation creation, renaming and respectively deletion. As discussed in the section about extraction variants, we can also detect “splits” of variants. The view indicates such splits by lines connecting different variants. Hovering over a circle additionally reveals details about the underlying commit including its identifier (sha) and message.
Commit-centric similarity view

Similarities in 69 commits.

- Commit 0cc8e6d: 2
- Commit 553750c: 14
- Commit 65819a9: 2
- Commit 4906844: 2
- Commit 2093464: 4
- Commit c84b74a: 4
- Commit a2afdad: 20
- Commit de18ac9: 10
- Commit 8fab4ea: 6
- Commit 7afb5ff: 8
- Commit a4f73c0: 178
Commit-centric similarity view

<table>
<thead>
<tr>
<th>Similarities in 69 commits.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commit 4906844</td>
</tr>
<tr>
<td>Commit 2093464</td>
</tr>
<tr>
<td>Commit c84b74a</td>
</tr>
<tr>
<td>Commit a2afdad</td>
</tr>
<tr>
<td>Commit de18ac9</td>
</tr>
<tr>
<td>Variant haskellSyb</td>
</tr>
<tr>
<td>Variant monoidal</td>
</tr>
<tr>
<td>Variant haskellComposition</td>
</tr>
<tr>
<td>Commit 8fab4ea</td>
</tr>
<tr>
<td>Commit 7afb5ff</td>
</tr>
<tr>
<td>Commit a4f73c0</td>
</tr>
</tbody>
</table>

Variants modified by a commit
Commit-centric similarity view

Similarities in 69 commits.

- Commit 4906844
- Commit 2093464
- Commit c84b74a
- Commit a2afdad
- Commit de18ac9

  - Variant haskellSyb
  - Variant monoidal

  - File src/Main.hs
  - File src/Company/Ranking.hs

  - Variant haskellComposition

  - Commit 8fab4ea

Files modified by a commit
Commit-centric similarity view

Similarities in 69 commits.

- Commit 4906844 2
- Commit 2093464 4
- Commit c84b74a 4
- Commit a2afdad 20
- Commit de18ac9 10
- Variant haskellSyb 4
- Variant monoidal 4
- File src/Main.hs 2
  - Fragment pattern/rankingFailTest 1
  - Fragment pattern/rankingOkTest 1
- File src/Company/Ranking.hs 2

Fragments modified by a commit
Variant-centric similarity view
Variant-centric similarity view
Variant-centric similarity view

Variant exploration

Level: file

- **Variant haskellLogging** 82/126 annotated
- **Folder src** 82/126 annotated
  - **File Main.hs** 7/29 annotated
    - Fragment pattern/sampleCompanyLog: 1/1 annotated
    - Fragment pattern/cutLogTest: 1/1 annotated
    - Fragment pattern/deltasTest: 1/1 annotated
    - Fragment pattern/meanTest: 1/3 annotated
    - Fragment pattern/medianTest: 1/3 annotated
    - Fragment pattern/tests: 1/1 annotated
    - Fragment pattern/main: 1/19 annotated
  - **Folder Company** 75/97 annotated

- File src/Main.hs (in haskellWriter) 30.16% similar
- File src/Main.hs (in haskellLogging) 6.51% similar
- File src/Main.hs (in haskellScott) 5.40% similar
- File src/Main.hs (in haskellLambda) 4.05% similar
- File src/Main.hs (in haskellList) 4.05% similar
- File src/Main.hs (in haskellEngineer) 4.05% similar
- File src/Main.hs (in haskellLens) 3.86% similar
- File src/Main.hs (in haskellData) 3.86% similar
- File src/Main.hs (in haskellRecord) 3.86% similar
- File src/Main.hs (in haskellComposition) 3.68% similar
- File src/Main.hs (in haskellVariation) 3.52% similar
- File src/Main.hs (in haskellTermRep) 3.38% similar
- File src/Main.hs (in haskellBarchart) 3.24% similar
Variant-centric similarity view
View on specific similarity

Annotations

Similar fragments
Similarity: 0.90 --> 0.91

pattern/medianTest in
Variant: nonmonadic
Path: src/Main.hs

pattern/meanTest in
Variant: nonmonadic
Path: src/Main.hs
Annotation of a similarity
Automated metadata update =
incremental similarity analysis +
revision of annotations

Recommend invariant for unannotated similarities:
▷ ‘Maintain Equality’ — if diff ratio = 1 at HEAD.
▷ ‘Maintain Similarity’ — otherwise.
Subject to interactive tool support for a ‘TODO list’, these recommendations can be reviewed, confirmed, and altered by the developer.

The developer may accept or reject the invariants.
A threshold is assumed for similarities.
Automated metadata update = incremental similarity analysis + revision of annotations

Possibly turn invariant into task:
▷ 'Maintain Equality' $\mapsto$ ‘Restore Equality’
  if diff ratio $< 1$ at HEAD.
▷ ‘Maintain Similarity’ $\mapsto$ ‘Restore Similarity’
  if diff ratio has changed at HEAD.

The developer will need to act on the tasks eventually.
Automated metadata update = incremental similarity analysis + revision of annotations

_Possibly turn task into invariant:_

- ‘Restore Equality’ $\mapsto$ ‘Maintain Equality’
- ‘Establish Equality’ $\mapsto$ ‘Maintain Equality’
- ‘Increase Similarity’ $\mapsto$ ‘Maintain Equality’
  
  if diff ratio = 1 at HEAD.
- ‘Increase Similarity’ $\mapsto$ ‘Maintain Similarity’
  
  if the diff ratio has increased at HEAD.
- ‘Restore Similarity’ $\mapsto$ ‘Maintain Similarity’
  
  if the previous diff ratio is back at HEAD.

These tasks were thus resolved.
Automated metadata update =
incremental similarity analysis +
revision of annotations

Possibly turn task into invariant recommendation:
▷ ‘Restore Similarity’ ⷼ ‘Maintain Similarity’
  if the diff ratio has increased at HEAD.
▷ ‘Remove Equality’ ⷼ ‘Maintain Similarity’
  if diff ratio < 1 at HEAD.

These tasks are probably resolved.
The developer needs to confirm.
## Working with a TODO list

<table>
<thead>
<tr>
<th>Name</th>
<th>Auto?</th>
<th>Via rule?</th>
<th>Via eq-class?</th>
<th>Intent</th>
<th>Fragment 1</th>
<th>Fragment 2</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increase Similarity</td>
<td>Manual</td>
<td>No</td>
<td>No</td>
<td>data/Employee in</td>
<td>data/Employee in</td>
<td>data/Department in</td>
<td>1 → 0.98</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Variant: wxHaskell</td>
<td>Variant: happlack</td>
<td>Variant tmvar</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Path: src/Company/Company.hs</td>
<td></td>
<td>Path: Company.hs</td>
<td></td>
</tr>
<tr>
<td>Restore Equality</td>
<td>Failed</td>
<td>No</td>
<td>No</td>
<td>data/Department in</td>
<td>data/Department in</td>
<td>data/Department in</td>
<td>1 → 0.99</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Variant: happlack</td>
<td>Variant tmvar</td>
<td>Variant tmvar</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Path: Company.hs</td>
<td></td>
<td>Path: Company.hs</td>
<td></td>
</tr>
<tr>
<td>Establish Equality</td>
<td>Manual</td>
<td>No</td>
<td>No</td>
<td>pattern/company in</td>
<td>pattern/company in</td>
<td>pattern/company in</td>
<td>0.95 → 0.95</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Variant: happlack</td>
<td>Variant tmvar</td>
<td>Variant tmvar</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Path: SampleCompany.hs</td>
<td></td>
<td>Path: SampleCompany.hs</td>
<td></td>
</tr>
</tbody>
</table>
Integration into Git workflow: Basic workflow for reference
CHAPTER 6. INTEGRATING WITH GIT

- **propagated**: All changes have been propagated via automatically executing annotations.
- **unpropagated**: Due to file edits or new commits changes may have to be propagated.

Figure 6.2: Model of the Ann-specific Git workflow.

Figure 6.2 shows a model of the extended Git workflow. After cloning the repository it is not registered with Ann and is therefore in states unpropagated and unsynchronized. The user can then either start editing and will bring the workspace into a dirty state, or he or she can execute `git ann init` to extract all metadata and thereby make it synchronized with the history of the repository. Once the user plans to commit changes he or she first calls `git ann propagate` to trigger automatic annotation execution. This will bring the repository into states propagated and synchronized. The user can then commit the changes, resulting in propagated and unsynchronized states since the new commit was not inspected for new metadata yet. As in the standard Git workflow the user then first pulls before pushing commits. Because pulling may result in file changes the repository gets into an unpropagated state. Finally the user pushes the new commits.

We have not included the annotation process into this workflow since it is independent and can be performed at any time after the initial metadata extraction.

Integration into Git workflow: Enhanced workflow
Case study
Case study on 101haskell: 
Haskell-based variants of a HRMS

- **haskellStarter.** Contribution with small language footprint.
- **haskellComposition.** Use of recursive data types.
- **haskellVariation.** Use of multiple constructors per type.
- **haskellFlat.** A Haskell-based data model illustrative for data parallelism.
- **wxHaskell.** GUI programming in Haskell with wxHaskell.
- **hxtPickler.** XML data binding for Haskell with HXT’s XML pickler.

... 

36 variants in total
101haskell is the Haskell slice of the 101companies project

Company X: Swing + JDBC
Company Y: SWT + Hibernate
Company Z: GWT + MongoDB

... which is a community project aiming at a knowledge base about software technologies and languages based on implementations of a human-resources management system.

http://101companies.org/
101’s Human Resources Management System

- Total salaries
- Increase salaries
- Cut salaries
- Edit employee data
- Import / export company data

A hint at the feature model
Stepwise process in case study

- **Annotations** for some evolutions *(A1)*
  - Always equal
  - Converge to Equal
  - Diverge from Equal
- **Restore equalities automatically** *(ARE)*
- **Annotations** for another evolution *(A2)*
  - Always Non-equal
- **Establish equalities manually** *(MRE)*
- **Increase similarities manually** *(MIS)*
Metrics for similarity management

- Equality classes (non singletons)
  - ↓ total number / ↑ max size / ↑ median size / ↑ average size

- Fragments
  - total number / ↓ distinct / ↑ shared / ↓ unshared
  - variants sharing ↑ average / ↑ median

- Annotations
  - ↑ Maintain Equality / ↓ Establish Equality / ↓ Increase Similarity

- Variants
  - uniqueness ↓ average / ↓ median
<table>
<thead>
<tr>
<th>Equality classes</th>
<th>INIT</th>
<th>A1</th>
<th>ARE</th>
<th>2A</th>
<th>MEE</th>
<th>MIS</th>
<th>Δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>total (non-trivial)</td>
<td>95</td>
<td>93</td>
<td>85</td>
<td>85</td>
<td>−10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>max size (non-trivial)</td>
<td>13</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>+7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>median size (non-trivial)</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>+1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>average size (non-trivial)</td>
<td>4.08</td>
<td>5.02</td>
<td>4.89</td>
<td>4.89</td>
<td>+0.84</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fragments</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>total</td>
<td>925</td>
<td>925</td>
<td>925</td>
<td>925</td>
<td>±0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>distinct</td>
<td>632</td>
<td>617</td>
<td>595</td>
<td>595</td>
<td>−37</td>
<td></td>
<td></td>
</tr>
<tr>
<td>shared</td>
<td>388</td>
<td>401</td>
<td>415</td>
<td>415</td>
<td>+27</td>
<td></td>
<td></td>
</tr>
<tr>
<td>unshared</td>
<td>537</td>
<td>524</td>
<td>510</td>
<td>510</td>
<td>−27</td>
<td></td>
<td></td>
</tr>
<tr>
<td>variants sharing (median)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>±0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>variants sharing (average)</td>
<td>1.46</td>
<td>1.51</td>
<td>1.55</td>
<td>1.55</td>
<td>+0.09</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Similarities</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>median</td>
<td>0.9885</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>+1.163%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>average</td>
<td>0.94186</td>
<td>0.96495</td>
<td>0.96795</td>
<td>0.96802</td>
<td>+2.777%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Similarity evolutions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Always Equal</td>
<td>777</td>
<td>777</td>
<td>780</td>
<td>780</td>
<td>+3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Converge to Equal</td>
<td>176</td>
<td>582</td>
<td>712</td>
<td>712</td>
<td>+536</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diverge from Equal</td>
<td>371</td>
<td>47</td>
<td>47</td>
<td>47</td>
<td>−324</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Always Similar</td>
<td>714</td>
<td>714</td>
<td>650</td>
<td>650</td>
<td>−64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Annotations</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maintain Equality (auto)</td>
<td>0 (0)</td>
<td>953 (858)</td>
<td>1359 (1264)</td>
<td>1359 (1264)</td>
<td>1492 (1387)</td>
<td>1492 (1387)</td>
<td>+1492</td>
</tr>
<tr>
<td>Restore Equality (auto)</td>
<td>0 (0)</td>
<td>324 (212)</td>
<td>0 (0)</td>
<td>0 (0)</td>
<td>0 (0)</td>
<td>0 (0)</td>
<td>±0</td>
</tr>
<tr>
<td>Establish Equality (auto)</td>
<td>0 (0)</td>
<td>0 (0)</td>
<td>0 (0)</td>
<td>130 (67)</td>
<td>0 (0)</td>
<td>0 (0)</td>
<td>±0</td>
</tr>
<tr>
<td>Remove Equality</td>
<td>0 (0)</td>
<td>0 (0)</td>
<td>0 (0)</td>
<td>0 (0)</td>
<td>0 (0)</td>
<td>0 (0)</td>
<td>±0</td>
</tr>
<tr>
<td>Increase Similarity (auto)</td>
<td>0 (0)</td>
<td>3 (0)</td>
<td>3 (0)</td>
<td>4 (0)</td>
<td>4 (0)</td>
<td>0 (0)</td>
<td>±0</td>
</tr>
<tr>
<td>Maintain Similarity (auto)</td>
<td>0 (0)</td>
<td>44 (20)</td>
<td>44 (20)</td>
<td>206 (123)</td>
<td>272 (145)</td>
<td>276 (145)</td>
<td>+276</td>
</tr>
<tr>
<td>Ignore Similarity</td>
<td>0 (0)</td>
<td>0 (0)</td>
<td>0 (0)</td>
<td>421 (0)</td>
<td>421 (0)</td>
<td>421 (0)</td>
<td>+421</td>
</tr>
<tr>
<td>Variants</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>uniqueness (median)</td>
<td>52.47%</td>
<td>49.00%</td>
<td>47.34%</td>
<td>47.34%</td>
<td>−5.13%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>uniqueness (average)</td>
<td>53.34%</td>
<td>51.39%</td>
<td>50.00%</td>
<td>50.00%</td>
<td>−3.34%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Concluding remarks

• New tool-supported, systematic process for similarity management. Without tool support, without a systematic workflow, the status of similarity is simply not known or hard to monitor for improvement!

• Incidentally, the approach helps in improving code quality and in feature location, too.
Thanks.