Development History Granularity Transformations

Kıvanç Muşlu  Luke Swart  Yuriy Brun  Michael D. Ernst

- Microsoft, Tools for Software Engineers
- University of Washington, Computer Science & Engineering
- HaxGeo, Civic Software Development
- University of Massachusetts Amherst, Information and Computer Science
Development histories simplify tasks

Development histories are used to:

• localize bugs
• rollback mistakes
• understanding software evolution
• predicting failures
• ...
Different tasks require different granularities

- Why does my test fail?
  - binary search history

- How did a feature evolve?
  - (thin) slice history

- How can I retrieve discarded code?
  - custom
  - raw history
  - keystroke
Problem: development histories are inflexible

- automatically-managed histories
  [YoonM11, Mahoney12, NegaraCDJ14]
  - Fine-grained: extracting relevant information requires post processing

- manually-managed histories
  - Incomplete: might miss information
  - Course-grained: information might be intermingled with irrelevant one
Solution: multi-grained development histories

Our contribution:
make recording granularity transparent

• record a complete & fine-grained history
• automatically transform this history into more optimal granularities for the task at hand
Solution: multi-grained development histories
Outline

Transformations

Design
Transformations

**granularity transformations**
(group changes that satisfy ... and reorder history such that ...)

**transformation operations**
(intermediate operations)

**transformation primitives**
expand, collapse, group
Primitives: expand, collapse, and move

collapse(2, 4)
Primitives: expand, collapse, and move

collapse \((2, 4)\)
Primitives: expand, collapse, and move
Primitives: expand, collapse, and move
Operation: group (move + collapse)

- move (3, 2) & move (4, 3)
- move (7, 6)
- collapse (1, 3)
- collapse (2, 4)
Transformation: GroupCompilable (group)
All transformations

- **GroupCompilable**: `group(collapse)`

- **GroupFiles**: `group(collapse + move)`
  - for each modified file, creates a group containing all edits on this file
  - useful for manual inspection (e.g., VCS diff)

- **GroupCollocated**: `expand + group(collapse + move)`
  - creates a group for each contiguous edit
  - useful for separating tangled changes
Codebase Manipulation:
a design for multi-grained histories
Codebase Manipulation: a design for multi-grained histories
Codebase Manipulation: a design for multi-grained histories
Codebase Manipulation: a design for multi-grained histories
Contributions

• identify inflexibility problem of the current development histories

• propose multi-grained histories
  • Builds on three primitives: collapse, expand, move
  • History is automatically recorded
  • Developer uses the most optimal granularity for the current task

• Codebase Manipulation: one design for multi-grained histories