SIMILARITY ANALYSIS VIA HISTORY ANNOTATIONS*

*work in progress

Thomas Schmorleiz · University of Koblenz-Landau
MOTIVATION

- Software is developed as sets of variants due to conflicting requirements
- Cloning commonly used when to develop such variants:
  - Adopt new features, test cases:
MOTIVATION

• Cloning commonly used when to develop such variants:
  • Initialize new variants based on others

• **Advantages:** Easy to create variants & independent developers

• **Disadvantages:** Redundancy, out-of-sync artifacts, lack of control

• Established approach: Product line engineering (PLE)

• Comes with migration risks
MOTIVATION

• Vision paper published:

  Flexible Product Line Engineering with a Virtual Platform

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

  Stefan Stânciulescu, Andrzej Wąsowski
  IT University of Copenhagen, Denmark

  Thomas Schmorleiz, Ralf Lämmel
  Universität Koblenz-Landau, Germany

  Ina Schaefer
  Technische Universität Braunschweig, Germany

• Core idea:

  • Low-risk transition from cloning to a product line
  • Identify a set of clone operators
AGENDA

1. Virtual platform, operators and propagate
2. Overall process
3. Infrastructure
4. Metadata
5. UX
6. Change propagation
OPERATORS

- **Virtual platform**: Set of reusable assets distributed among variants

- **Cloning-related operators** are applied to the VP:
OPERATORS

- Locate feature: Find all asset fragment fragments for a given feature
- Clone assets: Copy & paste assets from a source variant to a target variant
- Propagate changes: Push changes from a original variant to a cloned variant (or vice versa)
IDEA

- For propagate we need a cloning graph

- How to extract it from an existing repository?
IDEA

• How to extract the cloning graph from an existing repository?

• **Idea**: Let user annotate similarities in the repository history as clones and generate cloning graph based on the metadata
PROCESS

• Guided by a web application.

• Initially, user selects Git repo from local file system:

```
Repositories
- repo3 (/Users/tschmorleiz/Temp2/repo3/)
- 101haskell (/Users/tschmorleiz/Projects/101/101haskell/)
```
PROCESS

• Next, various extractors are called:

1. **Script extraction**: Used operations throughout history

2. **Variation extraction** including renaming detection

3. **Fragment extraction**, where fragments are consecutive lines of code
The application then provides various views of the repo.
• The application then provides various views of the repo
PROCESS

• **Informed decision**: User select range of commits

4. **Similarity extraction**: For every new fragment we detect highly similar fragment at the commit point
   • Similarity based on diff ration
   • Threshold for “highly similar” set by user

5. **Divergence extraction**: When did once highly similar fragments diverge?
PROCESS

• **Finally**: Similarities are presented to the user
  • Grouped by variations, files, and commits
  • Additionally shown as edges in the variation graph
• User annotates those similarities edges which were indeed caused by cloning
• Based on those edges a cloning graph is created
METADATA

• **User-provided**: Annotations of similarity edges

• Annotated edges hold an **intent**: Reason for cloning:

  ```
  data Company = Company String [Dept]
  data Dept = Dept String Employee [SubUnit]
  data Subunit = DU Dept | EU Employee
  ```

  “Company” fragment cloned. Intent: Simplification

  ```
  data Company = Company String [Dept]
  data Dept = Dept String Employee [Employee] [Dept]
  ```
METADATA

• **User-provided**: Annotations of similarity edges
  
  • E.g. used to decide whether changes can be automatically propagated
• **Generated**: Cloning graph based on annotated similarity edges, connecting containing variants
UX

• High amount of data to be processed by user

• Good user experience (UX) design principles should be followed

• Different views should be provided

• Feedback regarding annotation process
UX

• Iterative annotation process:
CHANGE PROPAGATION

• **propagate** inputs:
  • Repository
  • Generated cloning graph
  • Pushes changes along cloning edges
  • Utilizes intents to decide whether to push automatically or first get confirmation by user
  • Conflicts may involve automated or manual merging
CONCLUSION

• We have developed a web application for history annotation to implement propagate

• Currently missing:
  • Divergence extractor
  • Additional views

• Possible future work: Enable implementation of additional operators
REFERENCES


FEEDBACK, QUESTIONS?