Generating rules for API Evolution

David Wettstein, 17.05.16
API Evolution

- APIs change over time
- Changes need to be updated in your code manually
Project goal

• API changes can cause problems in projects using an older version of the API
• Get automatically detected and packaged rules, such that you know what changed
APIEvolutionMiner (AEM)

- APIEvolutionMiner\(^1\)
  - extract rules by monitoring API changes
  - mined at revision level in code history

APIEvolutionMiner (AEM)

- Investigated
  - Jenkins CI jobs from Inria RMoD
  - AEM source code
APIEvolutionMiner (AEM)

- Result: We can run it for any project per script
Problems with AEM

• Custom rule format:

• Wanted:
Problems with AEM

- AEM does not create RBLintRule!
- Main task:
Our approach

• Focus on automatically discovered changes of type “one to one”

• Generate subclasses of RBTransformationRule within code
  • using subclass: and compile:
Difficulties

- Differences in change pattern:
  - RTLlabelled new => RTLabeled new
  - ? builder: => ? addedInBuilder:
  - ? translateBy: => ? translate to:
RBTransformationRule

• Uses RBParseTreeRewriter for doing a search & replace in the parse tree
• Works also with placeholders (e.g. `@object`)
Solution

- **RTLabeled new => RTLabeled new**

This one is easy 😊
Solution

- `? builder:  =>  ? addedInBuilder:`
What about this?

• $\text{translateBy}: \Rightarrow \text{translate:to}:$

• Different number of arguments! Which one to take?
Example: Roassal2

- 68 automatic rules by AEM
- 57 convertible into RBTransformationRule
Results in Critic Browser

- Load Roassal2 version 1.4
- 706 critics found by 37 of 57 rules
Results in Critic Browser

• ? builder: => ? addedInBuilder:
Results in Critic Browser

• Load Roassal2 version 1.28 (13.05.16)
• 346 critics found by 20 of 57 rules
Future work

• Integrate the code into a CI service

• AEM still runs on Pharo 2
Summary

- APIs change over time
- APIEvolutionMiner cannot update changes automatically
- Our extension can do this