GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN

TU Clausthal

Simulating Software Refactorings based on Graph Transformations

Daniel Honsel[1], Niklas Fiekas[2], Verena Herbold[1], Marlon Welter[1], Tobias Ahlbrecht[2], Stephan Waack[1], Jürgen Dix[2], Jens Grabowski[1]

[1] Georg-August-Universität Göttingen
[2] Technische Universität Clausthal

Göttingen, April 28th, 2017

Outline

Motivation

Refactoring

Mining

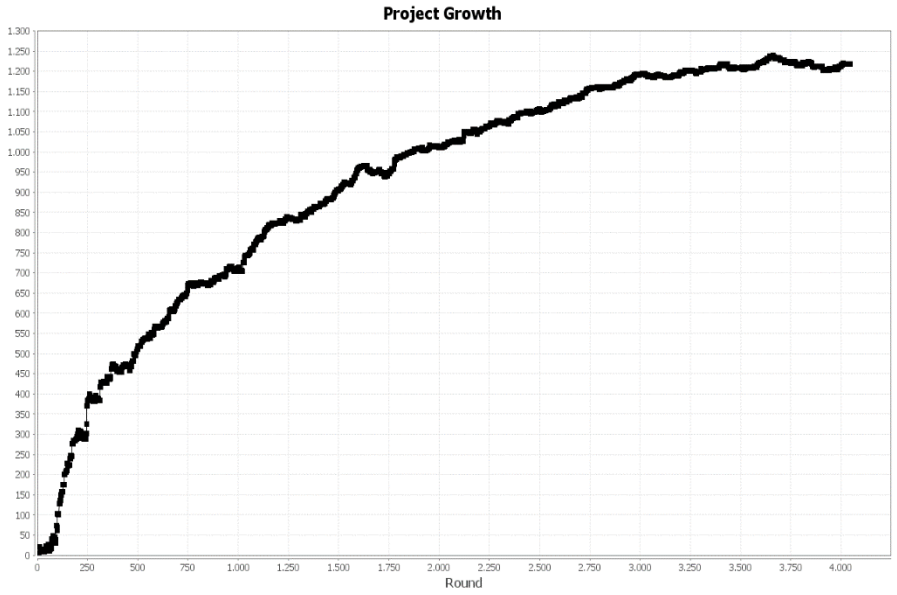
Results

Future Work

Simulating Software Processes

- Agent-based simulation model
  - Main entities: Developer, Software Entity
  - Required parameters come from mining software repositories

- Example: Project growth
  - Parameter:
    - Average size of a commit
    - Average number of commits of a developer per month



Relationships between Entities

- **Developer – Entity**: A weighted graph representing the work of developers on entities.
- **ChangeCoupling**: Dependencies between entities that are changed together several times. Input for the automated assessment.
  - Depends on developers behavior

References:
- Hattori, Lanza; *On the Nature of Commits*; 2008

Developers' Commit Behavior

- Entity **selection** for updates
  - Select **first entity randomly**
  - **Number** of entities to select depends on developer's intention: normal commit or bugfix
  - **Further entities** will be selected **based on** information about the **first one** and dependency graphs

- Apply the **commit**
  - Change involved entities and networks

→ Reduce randomness!

Outline

Motivation

Refactoring

Mining

Results

Future Work

Refactoring - Definition

Definition taken from:
Fowler, http://www.refactoring.com/

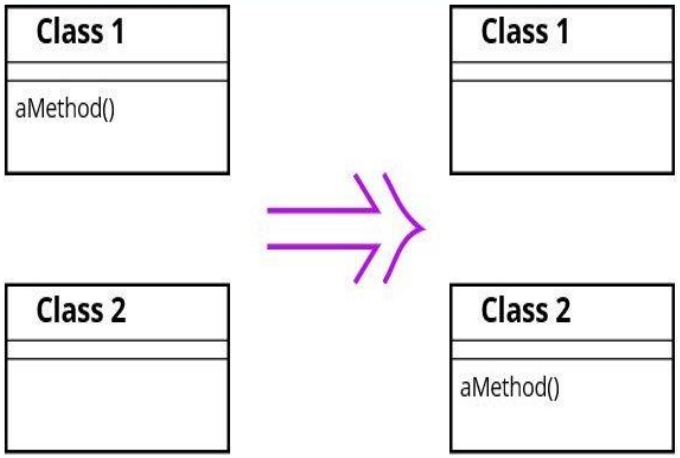
Definition: a change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior.

- Each transformation to the code is a small change, but the concatenation of transformations can produce significant changes

Refactoring - Examples

Examples taken from:
Fowler, http://www.refactoring.com/

Move Method



Extract Method

```
void printOwing() {
  printBanner();

  //print details
  System.out.println ("name:  " + _name);
  System.out.println ("amount " + getOutstanding());
}
```

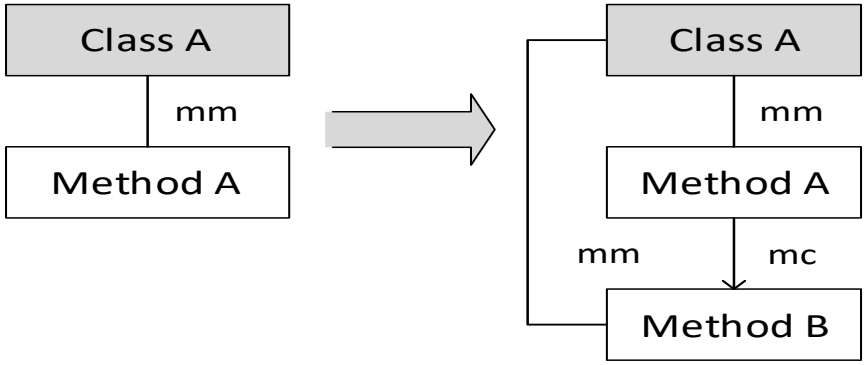
```
void printOwing() {
  printBanner();
  printDetails(getOutstanding());
}

void printDetails (double outstanding) {
  System.out.println ("name:  " + _name);
  System.out.println ("amount " + outstanding);
}
```

References:
- (1) Kreowski, Hinrichs, Kuske; *Some Essentials of Graph Transformation*; 2006

Rule based Graph Transformation (Basic Idea)

- Describe rules that express which part of a graph should be replaced by another graph
- Steps to transform a graph from a pre-state G to a post-state H:
  1. Find occurrence of L in G
  2. Delete from G all vertices and edges matched by L\R
  3. Paste to the result a copy of R\L => derived graph H

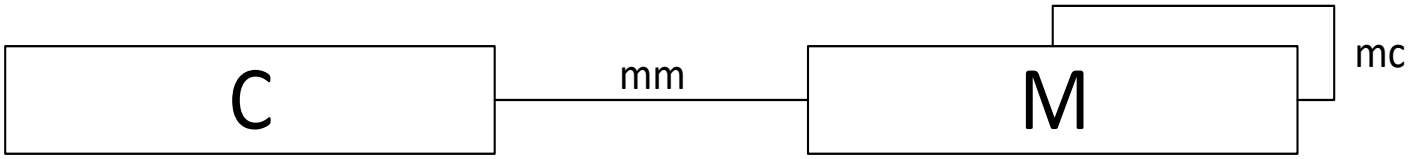


References:
Mens, Van Eetvelde, Demeyer; *Formalizing Refactorings with Graph Transformations*; 2005

Model Refactorings

- Additional graph
  - Nodes: Classes (C), Methods (M)
  - Edges: Method Membership (mm), Method Call (mc)

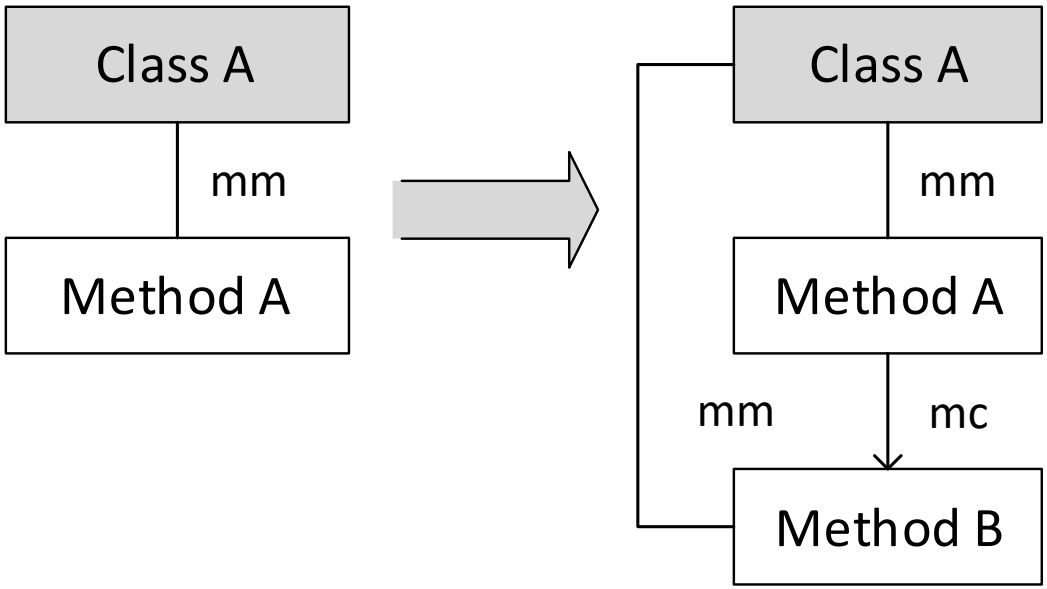
- Type Graph to restrict edge creation



Type Graph

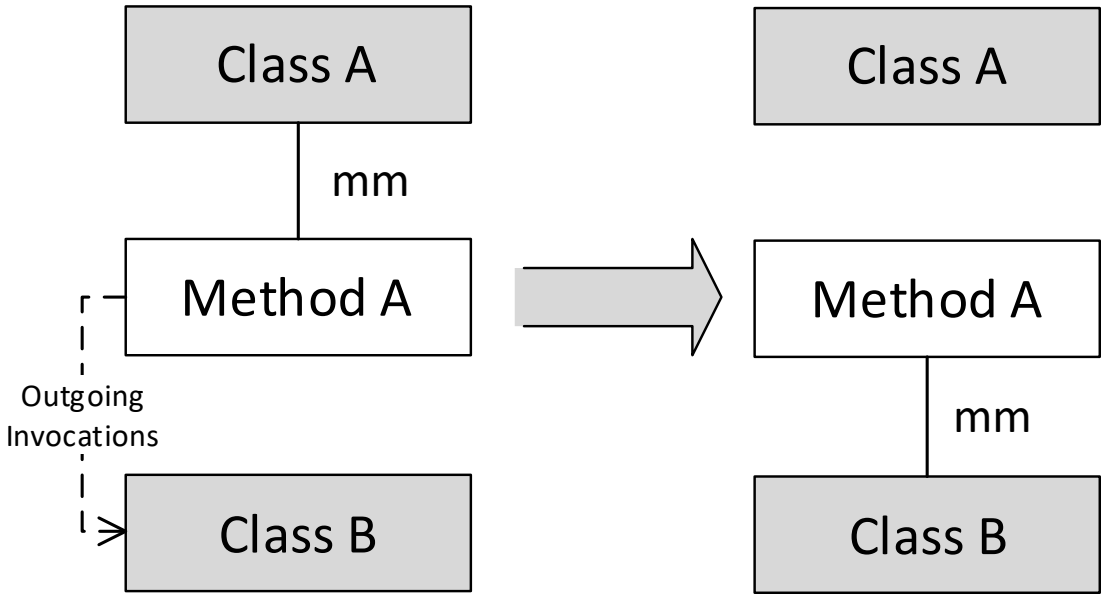
Rules for Refactorings (1)

Extract Method



Rules for Refactorings (2)

Move Method



Outline

Motivation

Refactoring

Mining

Results

Future Work

A Code Refactoring Dataset

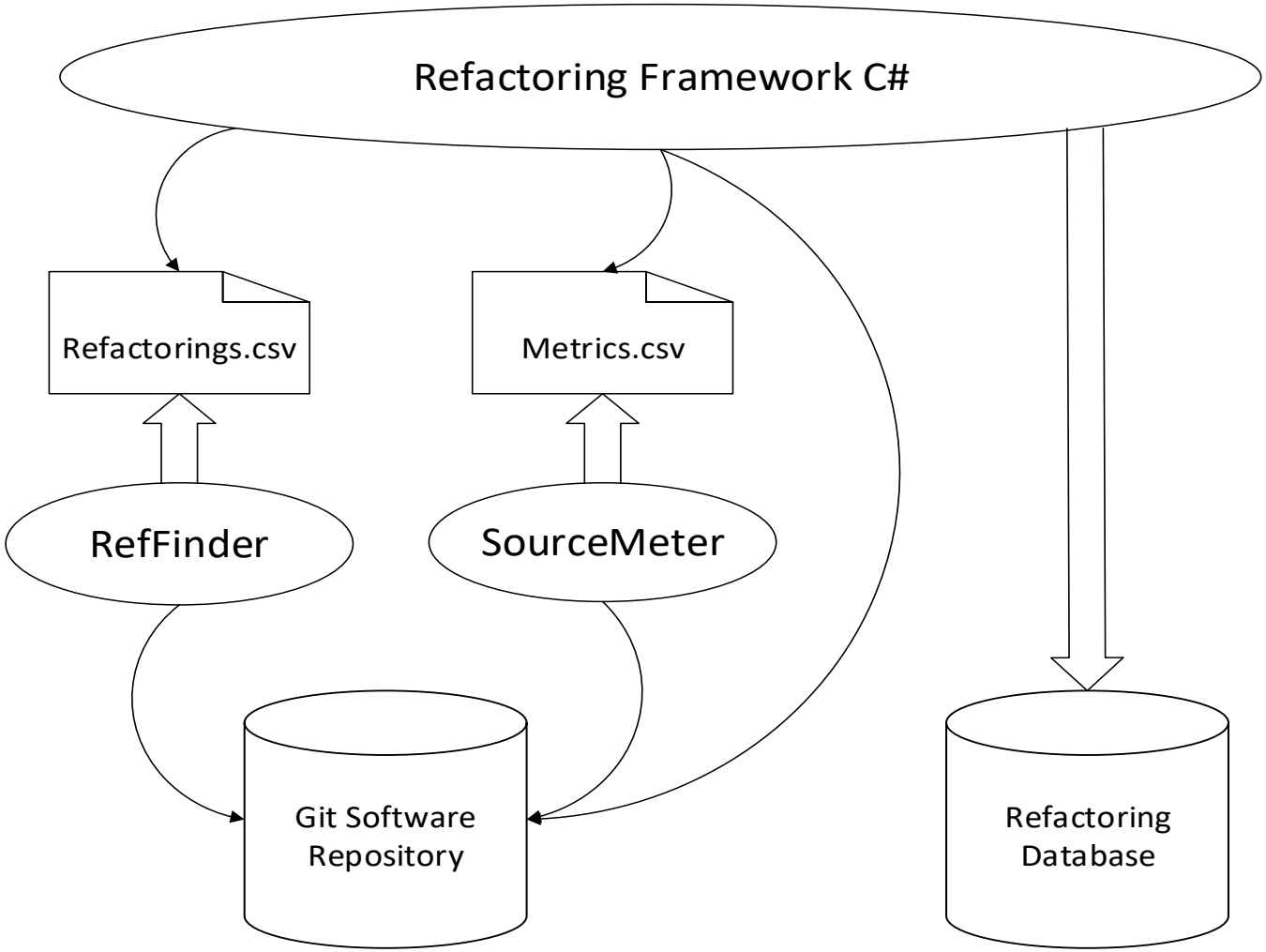
References:
Kádár, Hegedüs, Ferenc, Gyimothy; *A Code Refactoring Dataset and Its Assessment Regarding Software Maintainability*; 2016

- Published Refactoring Dataset includes refactorings of 7 open source projects between several releases and source code metrics
  - Most effected source code metrics by refactorings:
    - **SIZE** (LOC, NOS)
    - **COUPLING** (NOI)
    - **CLONE** (CC)

- Not detailed enough for the estimation of simulation parameters
  - We require metrics for each transition of consecutive commits

Mining Framework - Overview

References:
- https://sites.google.com/site/reffindertool/
- https://www.sourcemeter.com/



Outline

Motivation

Refactoring

Mining

Results

Future Work

Results of mining junit (1)

- We analyzed 2069 commits (from 03/2002 to 11/2016)
- 18.2% of all commits contain at least one refactoring
- Number of found refactorings

Type	Per Commit Transition	Complete
Extract Method	164	222
Inline Method	136	235
Move Method	171	1766

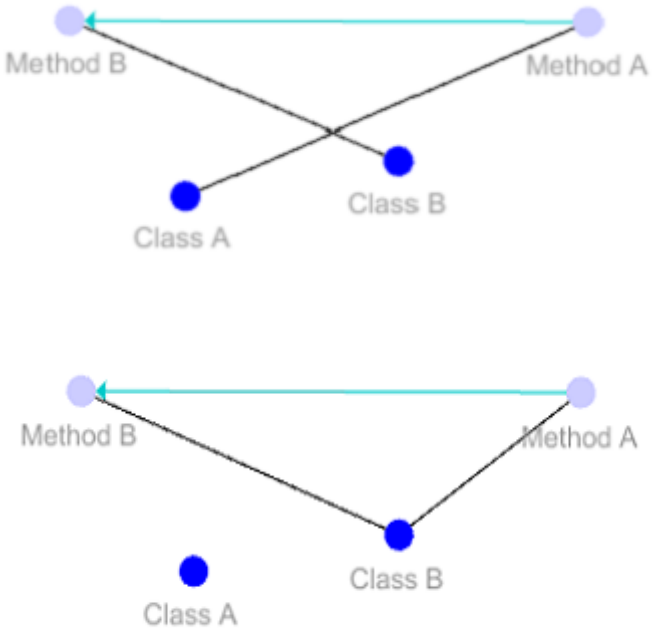
Results of mining junit (2)

- Analyzed metrics for Extract Method and Inline Method:
  - Size (LOC), Complexity (McCC)
  - Extract Method: Size (13) and Complexity (3) of the class increase
  - Inline Method: Size (18) and Complexity (3) of the class decrease

- Analyzed metrics for Move Method:
  - Size (LOC), Coupling (NOI)
  - Only 5.5% of the methods are moved to an existing class
    - NOI reduced in base class but increased in new class

Implementation of Transformation Rules

- Simulation model for each refactoring
- Test each rule separated on a suitable software graph
- Simulation results of Move Method



Outline

Motivation

Refactoring

Mining

Results

Future Work

Future Work

- Investigate further projects
- Consider further refactorings
- Identify patterns for “bug fixes” and “add features”
- Integrate this model into the simulation of software projects using the BDI approach

Thank you for your attention!

daniel.honsel@cs.uni-goettingen.de

Supported by

SWZ Clausthal – Göttingen
Simulationswissenschaftliches Zentrum

www.simzentrum.de

Results of mining junit (3)

Extract Method

Complexity (McCC)

- Base Method: 2.5
- Δ Base Method: -0.2
- Δ Class: 2.8

Size (LOC)

- Base Method: 9.8
- Δ Base Method: -1.4
- Δ New Method: 5.2
- Δ Class: 13.2

Inline Method

Complexity (McCC)

- Caller Method: 2.0
- Inlined Method: 1.4
- Δ Caller Method: -0.03
- Δ Inlined Method: -1.4
- Δ Class: -3.13

Size (LOC)

- Caller Method: 7.2
- Inlined Method: 5.3
- Δ Caller Method: 0.6
- Δ Inlined Method: -5.2
- Δ Class: -17.8

Results of mining junit (4)

Move Method

Only 5.5% of the methods are moved to an existing class

Size (LOC)

Base Class: 35.2
Target Class: 4.1
Method: 5.4
Δ Base Class: -28.3
Δ Target Class: 29.2

Coupling (NOI)

Base Class: 3.4
Target Class: 0.4
Δ Base Class: -2.6
Δ Target Class: 2.8

Simulation Model