Software Refactoring Using Multi-Objective Optimization

Ali Ouni

GEODES Software Engineering Research Group, University of Montreal
McDonnell Douglas Software Engineering Laboratory, Missourey University S&T

March 22, 2013
Outline

- Context
- Problem statement
- Proposal
  - Multi-objective search-based refactoring
- Evaluation
- Conclusion
Context

• Software is complex
• It changes frequently
  • Add new functionalities
  • Correcting bugs
  • Adaptation to environment changes
• Easiness to accommodate changes depends on software quality
  • Refactoring
Refactoring

- Refactoring
  - The process of improving a code after it has been written by changing its internal structure without changing the external behavior (Fowler et al., ‘99)
  - Examples: Move method, extract class, move attribute, ...

- Two refactoring steps
  1. detection of code fragments to improve (e.g., design defects)
  2. identification of refactoring solutions
Step 1: Design defects detection

- Design defect introduced during the initial design or during evolution
  - Anomalies, anti-patterns, bad smells…
  - Design situations that adversely affect the development of a software
  - Examples: Blob, spaghetti code, functional decomposition, ...

The Blob example

• Definition
  – Procedural-style design leads to one object with numerous responsibilities and most other objects only holding data or executing simple processes.

• Symptoms
  – A Blob is a controller class, abnormally large, with almost no parents and no children. It mainly uses data classes, i.e. very small classes with almost no parents and no children (Brown et al. ’98).
Step 2: Refactoring

Context:
- Problem statement
- Proposal
- Evaluation
- Conclusion

Librairy_Main_Control
- borrow_date_Book
- current_Book
- current_Catalog
- fine_Amount
- listOfCatalogs
- library_opened
- reserved_Book
- return_date_Book
- +add_Book()
- +archive_Catalog()
- +borrow_Book()
- +check_Availab_Book()
- +check_Delay_Book()
- +check_FineAmount()
- +check_ValueNetwork()
- +close_Library()
- +create_cheap_Book()
- +display_Book()
- +display_Catalog()
- +do_Inventory()
- +issue_LibraryCard()
- +list_Catalogs()
- +open_Library()
- +print_Catalog()
- +reserve_Book()
- +return_Book()
- +search_Book()
- +search_Catalog()
- +sort_Catalog()

Catalog
- ...+
- ...()

Data Class

Book
- borrow_date_Book
- current_Book
- current_Catalog
- fine_Amount
- listC_Catalogs
- library_opened
- reserved_Book
- return_date_Book
- +add_Book()
- +archive_Catalog()
- +borrow_Book()
- +check_Availab_Book()
- +check_Delay_Book()
- +check_ValidityCard()
- +create_cheap_Book()
- +display_Book()
- +display_Catalog()
- +do_Inventory()
- +issue_LibraryCard()
- +list_Catalogs()
- +open_Library()
- +print_Catalog()
- +reserve_Book()
- +return_Book()
- +search_Book()
- +search_Catalog()
- +sort_Catalog()

NewClass_1
- fine_Amount
- +check_FineAmount()
- +check_ValueNetwork()
- +issue_LibraryCard()

NewClass_2
- library_opened
- +close_Library()
- +open_Library()

Catalog
- current_Catalog
- listOfCatalogs
- +sort_Catalog()
- +add_Book()
- +list_Catalogs()
- +remove_Book()
- +search_Book()
- +search_Catalog()

Data Class

Blob

Refactoring
- Move method
- Extract class
- Move field
- Add association

...
Existing work

• Metric-based approaches
  – Search-based techniques
    • Find the best sequence of refactorings (Harman et al. ’07, O’Keeffe et al. ’08)
  – Analytic approaches
    • Study of relations between some quality metrics and refactoring changes (Sahraoui et al. ’00, Du Bois et al. ’04, Moha et al. ’08)

• Graph-based approaches
  – Graph transformation
    • Software is represented as a graph
    • Refactorings activities as graph production rules (Kataoka et al. ’01, Heckel et al. ’95)
Existing work

• Limitations
  – Difficult to define "standard" refactoring
  – Difficulty to propose refactoring solutions for each defect type
  – Correct defects separately
  – Do not consider the impact of refactoring
    • Correcting a defect may produce other defects
  – Do not consider the number of code changes
  – The semantic coherence is not concerned
  – Code change history is not considered
Refactoring challenges...

- Number of code changes?

**Solution 1**
1. Move method
2. Extract class
3. Move field
4. Move method
5. Move method

**Solution 2**
1. Move method
2. Move method
3. Inline class
4. Move field
5. Extract class

**Solution 3**
1. Move method
2. Extract class
3. Move field
4. Move method
5. Move method

**Solution 4**
1. Move method
2. Move method
3. Inline class
4. Move field
5. Extract class
6. Move field
7. Extract class
Refactoring challenges...

- Semantic preservation?
Refactoring challenges...

• Reuse of good refactorings applied in the past?
Approach overview

Multi-objective search-based refactoring

A: List of possible refactorings

B: Design defects detection rules

C: Effort approximation

Input: Refactoring efforts

Output: refactoring efforts

D: Semantic similarity measures

Input: Source code + call graph

Output: semantic measures

E: Similarity with good recorded refactorings

Input: List of previous program versions

Output: recorded refactorings
Multi-Objective Refactoring

• See the refactoring task as a multi-objective optimization problem
  – Improve software quality
  – Minimize the number of code changes
  – Preserve the semantics
  – Maximize similarity with good recorded refactorings applied to similar contexts
NSGA-II overview

NSGA-II: Non-dominated Sorting Genetic Algorithm (K. Deb et al., ’02)

Parent Population

Offspring Population

Non-dominated sorting

Crowding distance sorting

Population in next generation
NSGA-II adaptation

- Representation of the individuals
- Creation of a population of individuals
- Creation of new individuals using genetic operators (crossover and mutation)
- Definition of fitness functions
Representation of individuals

- Individual = Refactoring solution
- Sequence of refactoring operations

1. moveMethod
2. pullUpAttribute
3. extractClass
4. inlineClass
5. extractSuperClass
6. inlineMethod
7. extractClass
8. moveMethod
Representation of individuals

- Specify the controlling parameters
  - Random selection for the initial population

<table>
<thead>
<tr>
<th>Refactorings</th>
<th>Controlling parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>move method</td>
<td>(sourceClass, targetClass, method)</td>
</tr>
<tr>
<td>move field</td>
<td>(sourceClass, targetClass, field)</td>
</tr>
<tr>
<td>pull up field</td>
<td>(sourceClass, targetClass, field)</td>
</tr>
<tr>
<td>pull up method</td>
<td>(sourceClass, targetClass, method)</td>
</tr>
<tr>
<td>push down field</td>
<td>(sourceClass, targetClass, field)</td>
</tr>
<tr>
<td>push down method</td>
<td>(sourceClass, targetClass, method)</td>
</tr>
<tr>
<td>inline class</td>
<td>(sourceClass, targetClass)</td>
</tr>
<tr>
<td>extract class</td>
<td>(sourceClass, newClass)</td>
</tr>
</tbody>
</table>
Population creation

- Population: set of refactoring solutions
Genetic Operators

1. Crossover

Solution A

| moveMethod |
| pullUpAttribute |
| extractClass |
| moveMethod |
| moveMethod |
| PushDownMethod |
| inlineClass |

Solution B

| extractClass |
| moveAttribute |
| pullUpAttribute |
| moveMethod |
| moveMethod |
| pullUpAttribute |
| moveAttribute |

Crossover (k=3)

Child A

| moveMethod |
| pullUpAttribute |
| extractClass |
| moveMethod |
| moveMethod |
| pullUpAttribute |
| moveAttribute |

Child B

| moveMethod |
| moveMethod |
| PushDownMethod |
| inlineClass |
| extractClass |
| moveAttribute |
| pullUpAttribute |

2. Mutation

Mutation (i=3, j=5)

| moveMethod |
| pullUpAttribute |
| extractClass |
| moveMethod |
| moveMethod |
| moveMethod |
| pushDownMethod |
| inlineClass |

| moveMethod |
| pullUpAttribute |
| moveMethod |
| extractSubClass |
| pushDownMethod |
| inlineClass |
Fitness functions

• Four fitness functions
  1. Quality
  2. Number of code changes
  3. Semantic coherence
  4. Similarity with recorded refactorings
Fitness functions

1. Quality
   – Minimize the number of detected defects

\[
\text{Quality} = \frac{\#\text{detected_defects}}{\#\text{detected_defects\_before\_refactoring}}
\]
Fitness functions

2. Code changes score

- Minimize code changes
- Given effort values for low-level refactoring
- Aggregated values for high-level refactoring

\[
\text{Number of changes} = \sum_{i=1}^{p} (n_i \times \text{Effort}_{LLR_i})
\]
## 2. Code changes score

<table>
<thead>
<tr>
<th>Low Level Refactorings</th>
<th>Create class</th>
<th>Delete Class</th>
<th>Add method</th>
<th>Delete method</th>
<th>Add Field</th>
<th>Delete Field</th>
<th>Create Relationship</th>
<th>Delete Relationship</th>
<th>Hide Method</th>
<th>Add Parameter</th>
<th>Remove Parameter</th>
<th>Modify Cardinality</th>
<th>Rename method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create class</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Delete Class</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add Field</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete Field</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create Relationship</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete Relationship</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hide Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add Parameter</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remove Parameter</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Modify Cardinality</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rename method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Fitness functions

2. Code changes score

<table>
<thead>
<tr>
<th>Effort Value</th>
<th>2</th>
<th>3</th>
<th>1</th>
<th>3</th>
<th>1</th>
<th>2</th>
<th>1</th>
<th>2</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extract Class</td>
<td>1</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>Extract interface</td>
<td>1</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>Extract method</td>
<td>1</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>Extract subclass</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>Extract superclass</td>
<td>1</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>Inline class</td>
<td>1</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>l</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>Inline method</td>
<td>1</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>Move class</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>Move Method</td>
<td>1</td>
<td>l</td>
<td>l</td>
<td>l</td>
<td>l</td>
<td>l</td>
<td>l</td>
<td>l</td>
<td>l</td>
<td>l</td>
<td>l</td>
<td>l</td>
<td>l</td>
</tr>
<tr>
<td>Move Field</td>
<td>1</td>
<td>l</td>
<td>l</td>
<td>l</td>
<td>l</td>
<td>l</td>
<td>l</td>
<td>l</td>
<td>l</td>
<td>l</td>
<td>l</td>
<td>l</td>
<td>l</td>
</tr>
<tr>
<td>Parametrize_Method</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>Pull up Field</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>Pull up Method</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>Push down Field</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>Push down Method</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>Replace subclass with field</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
<td>n</td>
</tr>
</tbody>
</table>
Extract class example

Person

Name
officeAreaCode
officeNumber
getPhoneNumber()

Extract class

Person

Name
getPhoneNumber()

PhoneNumber

officeAreaCode
officeNumber
getPhoneNumber()

\[
E_{\text{Extract\_Class}} = 1 \cdot E_{\text{Create\_Class}} + 2 \cdot E_{\text{Move\_Field}} + 1 \cdot E_{\text{Create\_Relationship}} + 1 \cdot E_{\text{add\_Method}}
\]

\[
= 1 \cdot 2 + 2 \cdot (E_{\text{add\_Field}} + E_{\text{delete\_Field}}) + 1 \cdot 1 + 1 \cdot 1
\]

\[
= 2 + 2 \cdot (1 + 2) + 1 + 1 = 10
\]
Fitness Functions

3. Semantics

- Minimize semantic errors
  - Vocabulary-based similarity (cosine similarity)
  - Dependency-based similarity (shared method call, shared fields)

• Intuition:
  – The correctness of proposed refactorings increase when applied to semantically connected elements.
Fitness Functions

4. Similarity with good recorded code changes applied in the past

\[ Sim_{\text{refactoring \_ history}}(RO_i) = \sum_{j=0}^{n-1} w \times m \]

where \( n \) is the number of possible refactorings to use
\( m \) is the number of times that refactoring has been applied in the past
\[ w = \begin{cases} 
2 & \text{if the same refactoring has been applied in the past} \\
1 & \text{if a compatible refactoring has been applied in the past} \\
0 & \text{otherwise} 
\end{cases} \]

- **Intuition:**
  - Recorded code changes can be used to propose new refactoring solutions in similar contexts.
Evaluation

• Research questions

– RQ1: **Usefulness**: To what extent can the proposed approach correct design defects, preserves semantics, minimizes code changes, and reuse recorded changes to improve the automaton of refactoring?

– RQ2: **Comparison to State of the Art Search**: How does the proposed approach perform compared to existing approaches?

– RQ3: **SBSE Validation**: How does the use of NSGA-II perform compared to other mono and multi-objective algorithms: MOGA, Random Search and GA?
Evaluation

• Answer to research questions
  – Comparison of different objectives combinations
    • Q, S, CC, RC
  – Comparison to state of the art search
    • Harman et al 2007, Kessentini et al 2011
    • Empirical study with potential users
    • Comparison with expected refactorings applied to next release
  – Comparison to other mono and multi-objective algorithms
    • Random Search, MOGA, GA
Evaluation

• Evaluation Methods
  – Manual validation
    • Empirical study with potential users
  – Automatic validation
    • Comparison with expected refactorings applied to next release
## Evaluation

- Data: Six medium/large open source Java projects

<table>
<thead>
<tr>
<th>Systems</th>
<th>Release</th>
<th># classes</th>
<th># defects</th>
<th>KLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xerces-J</td>
<td>v2.7.0</td>
<td>991</td>
<td>66</td>
<td>240</td>
</tr>
<tr>
<td>JFreeChart</td>
<td>v1.0.9</td>
<td>521</td>
<td>57</td>
<td>170</td>
</tr>
<tr>
<td>GanttProject</td>
<td>v1.10.2</td>
<td>245</td>
<td>41</td>
<td>41</td>
</tr>
<tr>
<td>AntApache</td>
<td>v1.8.2</td>
<td>1191</td>
<td>82</td>
<td>255</td>
</tr>
<tr>
<td>JHotDraw</td>
<td>v6.1</td>
<td>585</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>Rhino</td>
<td>v1.7R1</td>
<td>305</td>
<td>61</td>
<td>42</td>
</tr>
</tbody>
</table>
## Evaluation

- **Data:** Six medium/large open source Java projects
  - Ref-Finder

<table>
<thead>
<tr>
<th>Systems</th>
<th>Expected refactorings</th>
<th>Collected refactorings</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Next release</td>
<td># Refactorings</td>
</tr>
<tr>
<td>Xerces-J</td>
<td>v2.8.1</td>
<td>39</td>
</tr>
<tr>
<td>JFreeChart</td>
<td>v1.0.11</td>
<td>31</td>
</tr>
<tr>
<td>GanttProject</td>
<td>v1.11.2</td>
<td>46</td>
</tr>
<tr>
<td>AntApache</td>
<td>v1.8.4</td>
<td>78</td>
</tr>
<tr>
<td>JHotDraw</td>
<td>v6.2</td>
<td>27</td>
</tr>
<tr>
<td>Rhino</td>
<td>1.7R4</td>
<td>46</td>
</tr>
</tbody>
</table>
Evaluation

• Method: Four metrics

  – defect correction ratio (DCR)

\[
DCR = \frac{\# \text{detected defects}}{\# \text{defects before applying refactorings}}
\]

  – refactoring precision (RP)

\[
RP = \frac{\# \text{meaningful refactorings}}{\# \text{proposed refactorings}}
\]
– reused refactoring (RR)

\[ RR = \frac{\# \text{used refactorings in the base of code changes}}{\# \text{refactorings in the base of code changes}} \]

– Automatic refactoring precision

\[ \text{RP}_{\text{recall}} = \frac{|\text{suggested refactorings} \cap |\text{expected refactorings}|}{|\text{expected refactorings}|} \in [0,1] \]

\[ \text{RP}_{\text{precision}} = \frac{|\text{suggested refactorings} \cap |\text{expected refactorings}|}{|\text{suggested refactorings}|} \in [0,1] \]
Empirical study

• Survey
  – Questionnaire

• Subjects
  – 18 subjects (graduate/undergraduate students, assistant professors, junior software developers)
  – Subjects are volunteers and familiar with Java programming
  – 2 to 13 years experience on Java programming
  – 6 groups
Empirical study

- Scenario

<table>
<thead>
<tr>
<th>Ref. Solution</th>
<th>Algorithm/Approach</th>
<th># Fitness Functions</th>
<th>Objectives considered</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solution 1</td>
<td>NSGA-II</td>
<td>2</td>
<td>Q, CC</td>
</tr>
<tr>
<td>Solution 2</td>
<td></td>
<td>2</td>
<td>Q, S</td>
</tr>
<tr>
<td>Solution 3</td>
<td></td>
<td>3</td>
<td>Q, S, CC</td>
</tr>
<tr>
<td>Solution 4</td>
<td></td>
<td>3</td>
<td>Q, S, RR</td>
</tr>
<tr>
<td>Solution 5</td>
<td></td>
<td>4</td>
<td>Q, S, CC, RR</td>
</tr>
<tr>
<td>Solution 6</td>
<td></td>
<td>2</td>
<td>Q, CC</td>
</tr>
<tr>
<td>Solution 7</td>
<td></td>
<td>2</td>
<td>Q, S</td>
</tr>
<tr>
<td>Solution 8</td>
<td>MOGA</td>
<td>3</td>
<td>Q, S, CC</td>
</tr>
<tr>
<td>Solution 9</td>
<td></td>
<td>3</td>
<td>Q, S, RR</td>
</tr>
<tr>
<td>Solution 10</td>
<td></td>
<td>4</td>
<td>Q, S, CC, RR</td>
</tr>
<tr>
<td>Solution 11</td>
<td>Random Search (RS)</td>
<td>2</td>
<td>Q, CC</td>
</tr>
<tr>
<td>Solution 12</td>
<td></td>
<td>2</td>
<td>Q, S</td>
</tr>
<tr>
<td>Solution 13</td>
<td></td>
<td>3</td>
<td>Q, S, CC</td>
</tr>
<tr>
<td>Solution 14</td>
<td></td>
<td>3</td>
<td>Q, S, RR</td>
</tr>
<tr>
<td>Solution 15</td>
<td></td>
<td>4</td>
<td>Q, S, CC, RR</td>
</tr>
<tr>
<td>Solution 16</td>
<td>Genetic Algorithm</td>
<td>1</td>
<td>Q + S + CC + RR</td>
</tr>
<tr>
<td>Solution 17</td>
<td>Kessentini et al.</td>
<td>1</td>
<td>Q</td>
</tr>
<tr>
<td>Solution 18</td>
<td>Harman et al.</td>
<td>2</td>
<td>CBO, SDMPC</td>
</tr>
</tbody>
</table>
## Empirical study

<table>
<thead>
<tr>
<th>Subject groups</th>
<th>Systems</th>
<th>Algorithm / Approach</th>
<th>Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group A</td>
<td>GanttProject</td>
<td>NSGA-II Genetic Algorithm</td>
<td>Solution 1-5, 16</td>
</tr>
<tr>
<td></td>
<td>Xerces</td>
<td>MOGA, Harman et al.</td>
<td>Solution 6-10, 18</td>
</tr>
<tr>
<td></td>
<td>JFreeChart</td>
<td>RS, Kessentini et al.</td>
<td>Solution 11-15, 17</td>
</tr>
<tr>
<td>Group B</td>
<td>GanttProject</td>
<td>MOGA, Harman et al.</td>
<td>Solution 6-10, 18</td>
</tr>
<tr>
<td></td>
<td>Xerces</td>
<td>RS, Kessentini et al.</td>
<td>Solution 11-15, 17</td>
</tr>
<tr>
<td></td>
<td>JFreeChart</td>
<td>NSGA-II Genetic Algorithm</td>
<td>Solution 1-5, 16</td>
</tr>
<tr>
<td>Group C</td>
<td>GanttProject</td>
<td>RS, Kessentini et al.</td>
<td>Solution 11-15, 17</td>
</tr>
<tr>
<td></td>
<td>Xerces</td>
<td>NSGA-II Genetic Algorithm</td>
<td>Solution 1-5, 16</td>
</tr>
<tr>
<td></td>
<td>JFreeChart</td>
<td>MOGA, Harman et al.</td>
<td>Solution 6-10, 18</td>
</tr>
<tr>
<td>Group D</td>
<td>AntApache</td>
<td>NSGA-II Genetic Algorithm</td>
<td>Solution 1-5, 16</td>
</tr>
<tr>
<td></td>
<td>JHotDraw</td>
<td>MOGA, Harman et al.</td>
<td>Solution 6-10, 18</td>
</tr>
<tr>
<td></td>
<td>Rhino</td>
<td>RS, Kessentini et al.</td>
<td>Solution 11-15, 17</td>
</tr>
<tr>
<td>Group E</td>
<td>AntApache</td>
<td>MOGA, Harman et al.</td>
<td>Solution 6-10, 18</td>
</tr>
<tr>
<td></td>
<td>JHotDraw</td>
<td>RS, Kessentini et al.</td>
<td>Solution 11-15, 17</td>
</tr>
<tr>
<td></td>
<td>Rhino</td>
<td>NSGA-II Genetic Algorithm</td>
<td>Solution 1-5, 16</td>
</tr>
<tr>
<td>Group F</td>
<td>AntApache</td>
<td>RS, Kessentini et al.</td>
<td>Solution 11-15, 17</td>
</tr>
<tr>
<td></td>
<td>JHotDraw</td>
<td>NSGA-II Genetic Algorithm</td>
<td>Solution 1-5, 16</td>
</tr>
<tr>
<td></td>
<td>Rhino</td>
<td>MOGA, Harman et al.</td>
<td>Solution 6-10, 18</td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th>Systems</th>
<th>Approach</th>
<th>DCR</th>
<th>RP</th>
<th>Changes score</th>
<th>RP-automatic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xerces</td>
<td>NSGA-II</td>
<td>83% (55</td>
<td>66)</td>
<td>80%</td>
<td>3369</td>
</tr>
<tr>
<td></td>
<td>Harman et al. '07</td>
<td>N.A</td>
<td>40%</td>
<td>2669</td>
<td>8% (3</td>
</tr>
<tr>
<td></td>
<td>Kessentini et al. '11</td>
<td>89% (59</td>
<td>66)</td>
<td>37%</td>
<td>4873</td>
</tr>
<tr>
<td>JFreeChart</td>
<td>NSGA-II</td>
<td>86% (49</td>
<td>57)</td>
<td>83%</td>
<td>1979</td>
</tr>
<tr>
<td></td>
<td>Harman et al. '07</td>
<td>N.A</td>
<td>37%</td>
<td>3269</td>
<td>0% (0</td>
</tr>
<tr>
<td></td>
<td>Kessentini et al. '11</td>
<td>91% (52</td>
<td>57)</td>
<td>37%</td>
<td>3158</td>
</tr>
<tr>
<td>GanttProject</td>
<td>NSGA-II</td>
<td>85% (35</td>
<td>41)</td>
<td>80%</td>
<td>2640</td>
</tr>
<tr>
<td></td>
<td>Harman et al. '07</td>
<td>N.A</td>
<td>23%</td>
<td>4790</td>
<td>0% (0</td>
</tr>
<tr>
<td></td>
<td>Kessentini et al. '11</td>
<td>95% (39</td>
<td>41)</td>
<td>27%</td>
<td>4158</td>
</tr>
<tr>
<td>AntApache</td>
<td>NSGA-II</td>
<td>78% (64</td>
<td>82)</td>
<td>77%</td>
<td>4458</td>
</tr>
<tr>
<td></td>
<td>Harman et al. '07</td>
<td>N.A</td>
<td>40%</td>
<td>6987</td>
<td>04% (3</td>
</tr>
<tr>
<td></td>
<td>Kessentini et al. '11</td>
<td>80% (66</td>
<td>82)</td>
<td>30%</td>
<td>6587</td>
</tr>
<tr>
<td>JHotDraw</td>
<td>NSGA-II</td>
<td>86% (18</td>
<td>21)</td>
<td>80%</td>
<td>2158</td>
</tr>
<tr>
<td></td>
<td>Harman et al. '07</td>
<td>N.A</td>
<td>37%</td>
<td>3654</td>
<td>10% (4</td>
</tr>
<tr>
<td></td>
<td>Kessentini et al. '11</td>
<td>% (21)</td>
<td>43%</td>
<td>3798</td>
<td>7% (3</td>
</tr>
<tr>
<td>Rhino</td>
<td>NSGA-II</td>
<td>85% (52</td>
<td>61)</td>
<td>80%</td>
<td>1698</td>
</tr>
<tr>
<td></td>
<td>Harman et al. '07</td>
<td>N.A</td>
<td>37%</td>
<td>2698</td>
<td>0% (0</td>
</tr>
<tr>
<td></td>
<td>Kessentini et al. '11</td>
<td>87% (53</td>
<td>61)</td>
<td>37%</td>
<td>3156</td>
</tr>
<tr>
<td><strong>Average (all systems)</strong></td>
<td>NSGA-II</td>
<td>84%</td>
<td>80%</td>
<td>2717</td>
<td>35%</td>
</tr>
<tr>
<td></td>
<td>Harman et al. '07</td>
<td>N.A</td>
<td>36%</td>
<td>4011</td>
<td>4%</td>
</tr>
<tr>
<td></td>
<td>Kessentini et al. '11</td>
<td>89%</td>
<td>35%</td>
<td>4288</td>
<td>9%</td>
</tr>
</tbody>
</table>
Automatic Evaluation

- RP-automatic
Refactoring results of different objectives combinations with NSGA-II

- Refactoring Precision
Refactoring results of different objectives combinations with NSGA-II

- Defects Correction Ratio
Refactoring results of different objectives combinations with NSGA-II

- Code Changes Score
Algorithms comparison

Defects Correction Ratio

- Xerces-J
- JFreeChart
- GanttProject
- AntApache
- JhotDraw
- Rhino

Systems

DCR

NSGA-II
GA
MOGA
Random search
Algorithms comparison

Refactoring Precision

Systems

- Xerces-J
- JFreeChart
- GanttProject
- AntApache
- JHotDraw
- Rhino

- NSGA-II
- GA
- MOGA
- Random search
Algorithms comparison

Code changes reduction

- Xerces-J
- JFreeChart
- GanttProject
- AntApache
- JHotDraw
- Rhino

- NSGA-II
- GA
- MOGA
- Random search

NSGA-II multiple execution

RP (%), DCR(%), RR (%), Code changes score (*100), Time (min)
Conclusion

• A Multi-objective approach for software refactoring (defects correction)
  – Defects correction, semantics preservation, code changes reduction, refactoring reuse

• Validation
  – Six medium/large java open-source java systems
  – Comparison with state-of-the-art approaches
  – Comparison with other mono and multi-objective algorithms
  – Promising Results

• Future work
  – Statistic test
  – Design defects in SOA?
  – Refactoring of SOA…
Future work

- SOA design defects detection: Multi-service, Tiny service, ...
- **Multiservice**: Implements a multitude of methods, is not easily reusable because of low cohesion of its methods, is often unavailable due to overload.
• Future work
  – Tiny Service: Small service with few methods which requires several coupled services to complete an abstraction.
  – Refactoring of SOA?
Thanks for your attention

Questions? Comments? Suggestions?