RefDistiller: A Refactoring Aware Code Review Tool for Inspecting Manual Refactoring Edits

Everton L. G. Alves,[1 2] Myoungkyu Song,[2] Miryung Kim[3]
[1] University of Texas at Austin, [2]Federal University of Campina Grande, [3]University of California, Los Angeles

Problem Statement: How can we inspect manual refatoring edits effectively?

Static Analysis for Inspecting & Detecting Potential Semantic Changes in Manual Refactoring Edits

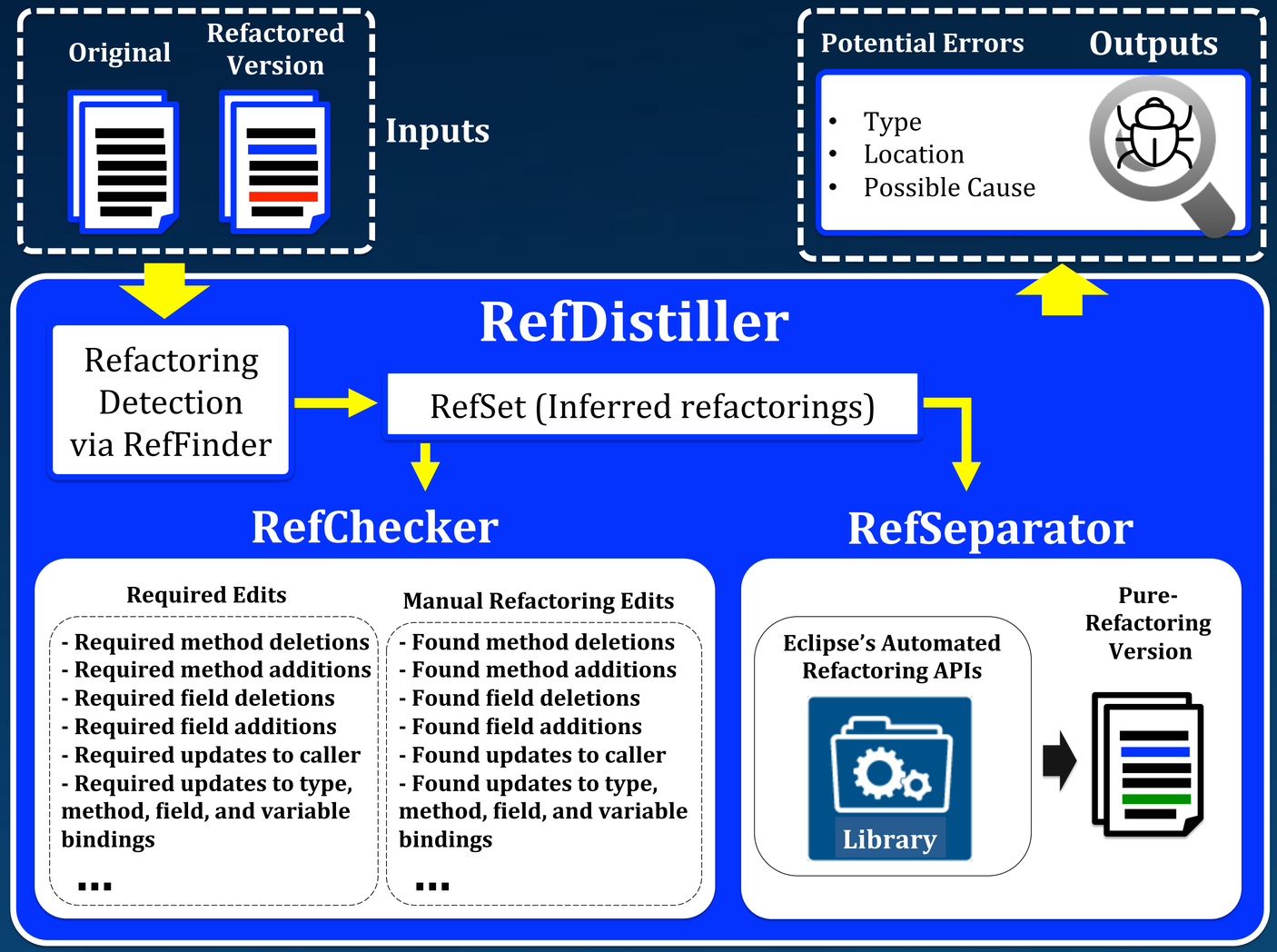
Problem

- Manual refactoring edits are error prone, as refactoring requires developers to coordinate related transformations and understand the complex inter-relationship between affected types, methods, and variables.
- Existing approaches either require having enough test coverage or do not detect semantics-modifying edits

Our Solution

To detect potential deviations from pure refactoring edits, **RefDistiller** incorporates two key techniques:

- **RefChecker** detects missing edits. Using refactoring templates, it checks for all required, constituent edits and required reference bindings.
- **RefSeparator** isolates extra semantics-modifying edits. It applies an equivalent pure refactoring to create a pure refactoring version and then compares the version against the manual refactoring version.



Practical Examples: Detecting Manual Refactoring Anomalies – Missing and Extra Edits

Missing Edits

Original Version

```
1 class BookManager extends SupplyManager{
2  ArrayList<Book> books;
3
4  Book findBook(String title, String  name){
5   for(int i=0; i < books.size(); i++){
6                Book book = books.get(i);
7                if (book.getTitle().equals(title)){
8                 return book;
9                }
10   }
11   return null;
12  }

13  void rent(Book book, int days){
14   Client client = getClient();
15   double price = getPrice(book, days);
16   registerRental (client, book, price);
17  }
     ...
}

1 class EBookManager extends BookManager {

2  double getPrice(Supply obj, int days) {
3   if (obj.isRecent())
4      return days * 4;
5   else
6       return days * 2;
  }}
```

Manual Refactoring Version

```
1 class BookManager extends SupplyManager{
2  ArrayList<Book> books;
3
4  Book findBook(String title, String name){
5   for(int i=0; i < books.size(); i++){
```

Detected a problematic binding of the reference to "getPrice(book, days)", which has been affected by a manual Pull Up Method refactoring.
➔ It should instead call "super.getPrice(book, days)"

```
13  void rent(Book book, int days){
14   Client client = getClient();
15   double price = getPrice(book, days);
16   registerRental (client, book, price);
17  }

18  double getPrice(Supply obj, int days){

19   if (obj.isRecent())
20      return days * 4;
21   else
22      return days * 2;
}}

1 class EBookManager extends BookManager{
  }
```

Pull up Method

Original Position

Extra Edits

Pure Refactoring Version

```
1 class BookManager extends SupplyManager{
2  ArrayList<Book> books;
3
4  Book findBook(String title, String name){
5   for(int i=0; i < books.size(); i++){
6                Book book = books.get(i);
7                if (checkTitle(book, title)) {
8                 return book;
9                }
10   }
11   return null;
12
}
```

Extract Method

```
13 public boolean checkTitle(Book book, String title){
14  return book.getTitle().equals(title);
17 }
```

Manual Refactoring Version

```
   ...
4  Book findBook(String title, String name){
5   for(int i=0; i < books.size(); i++){
6                Book book = books.get(i);
7                if (checkTitle(book, name)){
8                   return book.clone();
9                }
10   }
11   return null;
}
```

Detected semantics-modifying edits calling "book.clone."
➔ It should remove a call to "clone()."

Extract Method

```
13 boolean checkTitle(Book book, String title) {
14  return book.getTitle().equals(title);
17 }
```

This work was supported by National Science Foundation under grants CCF-1149391, CCF-1117902, SHF-0910818, a Google Faculty Award, and by National Institute of Science and Technology for Software Engineering, funded by CNPq/Brasil, grant 573964/2008-4.