Problem Statement: How can we inspect manual refactoring 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 test coverage or do not detect semantics-modifying refactoring requires developers to coordinate updates to caller required updates to type, method, field, and variable bindings.

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

**Missing Edits**

<table>
<thead>
<tr>
<th>Original Version</th>
<th>Manual Refactoring Version</th>
</tr>
</thead>
</table>
| 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 } ... 18 } 19 20 class EBBookManager extends BookManager { 21 22 double getPrice(Supply obj, int days) { 23 if(obj.isRecent()) 24 return days * 4; 25 else 26 return days * 2; 27 }

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

```java
return super.getPrice(book, days)
```

**Extra Edits**

<table>
<thead>
<tr>
<th>Pure Refactoring Version</th>
<th>Manual Refactoring Version</th>
</tr>
</thead>
</table>
| 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 } 13 void rent(Book book, int days){ 14 Client client = getClient(); 15 double price = getPrice(book, days); 16 registerRental(client, book, price); 17 } ... 18 } 19 20 class EBBookManager extends BookManager { 21 22 boolean checkTitle(Book book, String title) { 23 return book.getTitle().equals(title); 24 }

Detected semantics-modifying edits calling "book.clone()"

```java
...should return book.clone();
```