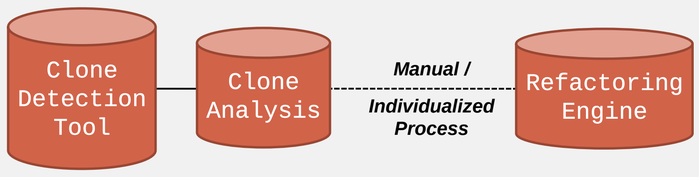
Maintaining Clones through Eclipse Refactoring Extensions

Robert Tairas (tairasr@cis.uab.edu)
Jeff Gray (gray@cis.uab.edu)

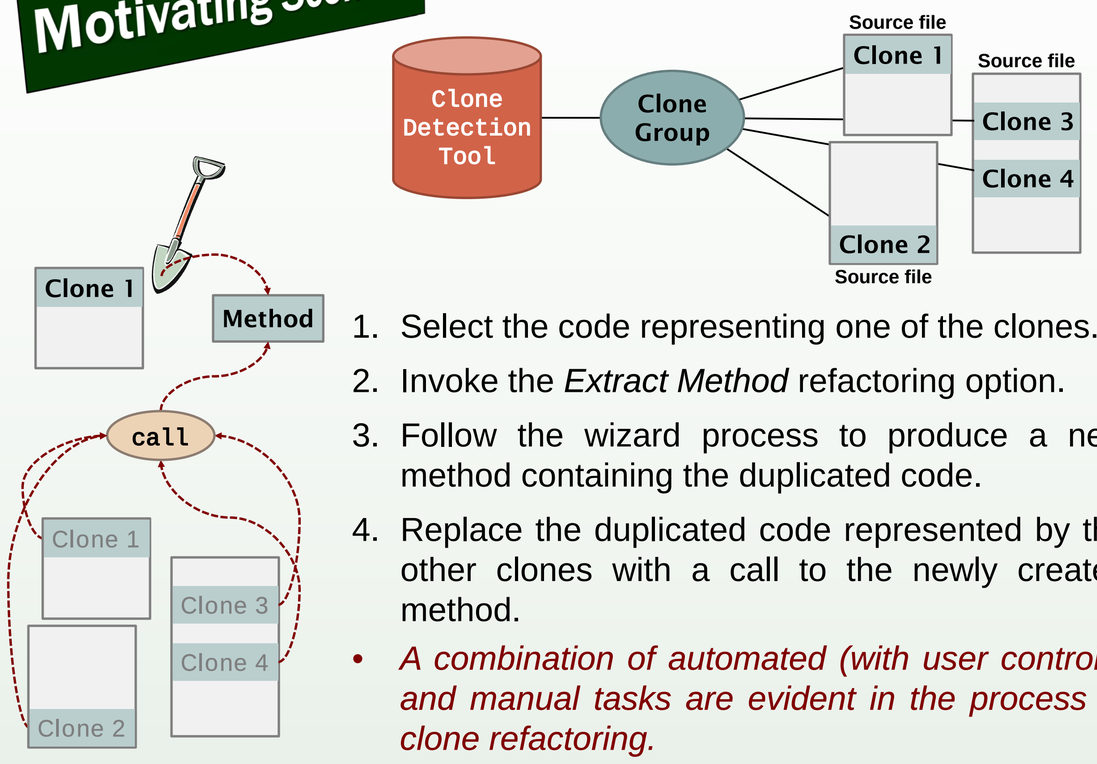
Software Composition and Modeling Laboratory
Department of Computer and Information Sciences
University of Alabama at Birmingham

Introduction



- Clone detection tools provide an automated way to discover sections of duplicated code.
- Clone analysis helps determine a group of clones to remove and replace with a single instance through refactoring tasks.
- IDEs such as Eclipse provide a framework to perform refactoring based on the section of code that is selected.
  - *Code ranges associated with the clones are manually communicated to the refactoring engine.*
  - *In some situations, the refactoring tasks are done separately for each clone, which suggests the need for more coordination for refactoring multiple clone instances.*

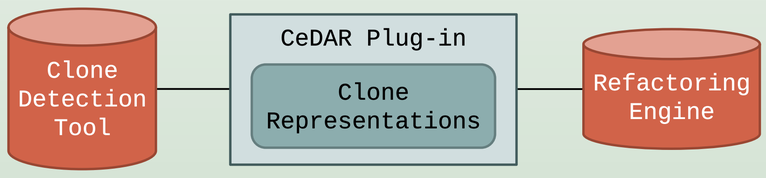
Motivating Scenario



1. Select the code representing one of the clones.
2. Invoke the *Extract Method* refactoring option.
3. Follow the wizard process to produce a new method containing the duplicated code.
4. Replace the duplicated code represented by the other clones with a call to the newly created method.

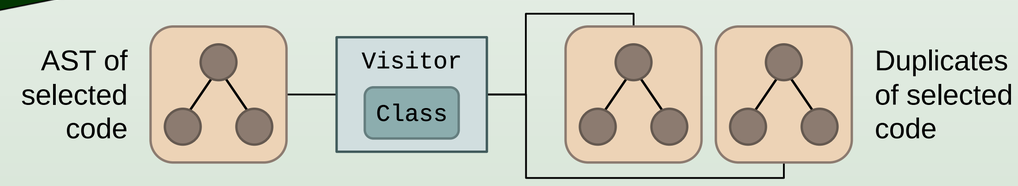
- *A combination of automated (with user controls) and manual tasks are evident in the process of clone refactoring.*

Our Approach

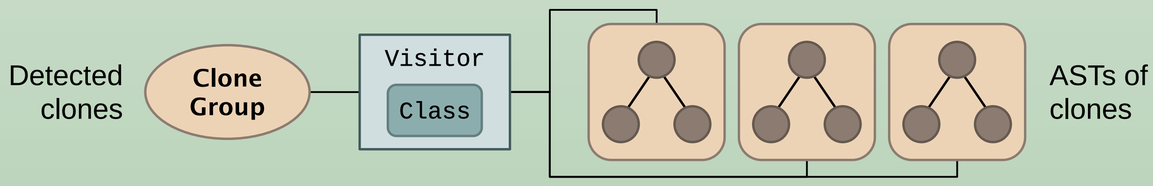


- The CeDAR (Clone Detection, Analysis, and Refactoring) plug-in parses a clone detection tool's report containing clone locations and groupings.
- The differences among a group of clones are identified by evaluating the corresponding AST nodes through a suffix tree.
- The nodes of clones selected for refactoring are passed to the refactoring engine.
- The refactoring engine's capabilities are extended to allow for more simultaneous refactorings of clones.

Passing Duplicates



- The Eclipse *Extract Method* option allows for the removal of similar code segments.
  - Detection is based on an internal sub-tree comparison on the AST of the class containing the selected section of code (above).
  - When duplicates of the selected code are found, the user is given the option to also refactor these instances.
- In our extension, the results of a clone detection tool are used to identify the location of the duplicated code (below).



Finding Differences

Clone 1

```
if (!delete(f)) {
  String message = "Unable to delete file "
    + f.getAbsolutePath();
  if (failonerror) {
    throw new BuildException(message);
  } else {
    log(message, quiet ? Project.MSG_VERBOSE
      : Project.MSG_WARN);
  }
}
```

Clone 2

```
if (!delete(dir)) {
  String message = "Unable to delete directory "
    + dir.getAbsolutePath();
  if (failonerror) {
    throw new BuildException(message);
  } else {
    log(message, quiet ? Project.MSG_VERBOSE
      : Project.MSG_WARN);
  }
}
```

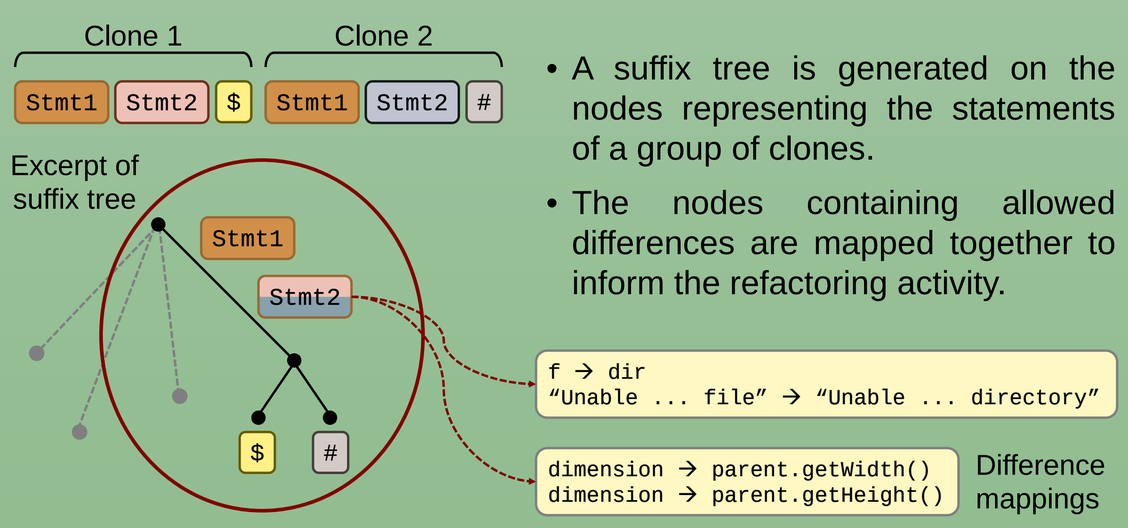
Snippet in Clone 1

```
int width = dimension – insets.right;
...
int height = dimension – insets.bottom;
```

Snippet in Clone 2

```
int width = parent.getWidth() – insets.right;
...
int height = parent.getHeight() – insets.bottom;
```

- Clones contain differences, such as variable names and string literals.
- Differences can also be between variables and method calls.



- A suffix tree is generated on the nodes representing the statements of a group of clones.
- The nodes containing allowed differences are mapped together to inform the refactoring activity.

Refactoring Clones

Extract Method

- We extended the refactoring to utilize clone information from detection tools (see "Passing Duplicates") and included more parameterized differences in the refactoring (utilizing information from "Finding Differences").
- Detected clones can be refactored at the same time.
- *Are refactorings containing more complicated differences evident and in demand?*

Extract Utility Method

- Eclipse can support Pull-up Method refactoring of identical method signatures in sub-classes.
- We extended the refactoring to perform the initial code extraction into a method and subsequent extraction into a new utility class (i.e., non-pull-up cases).
- *Clones within a method require composite refactorings (i.e., Extract Method/Pull-up Method or Extract Method/Extract Class). Should such refactorings be done separately or simultaneously?*

This material is based upon work supported by the National Science Foundation under Grant No. 0702764.