



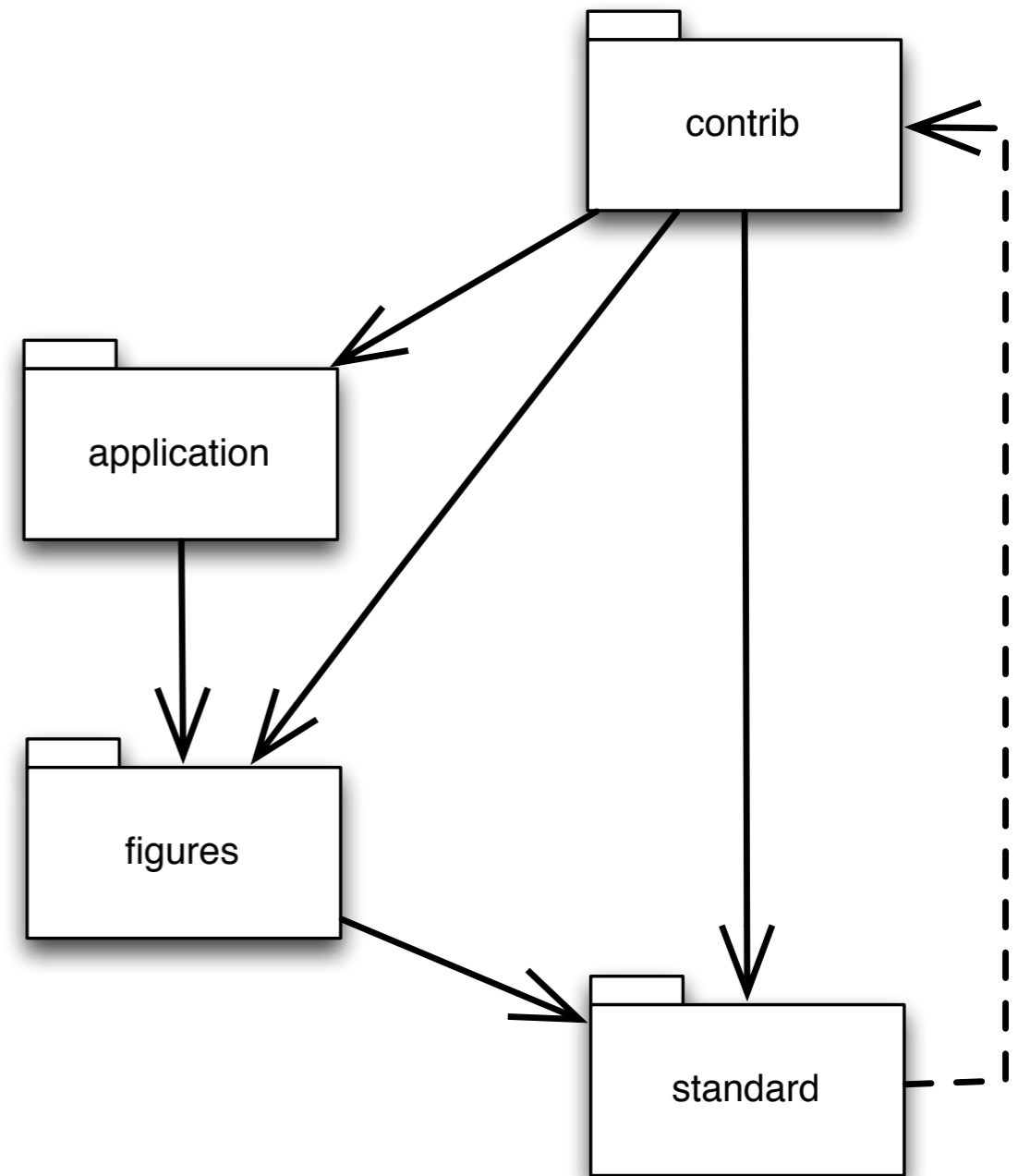
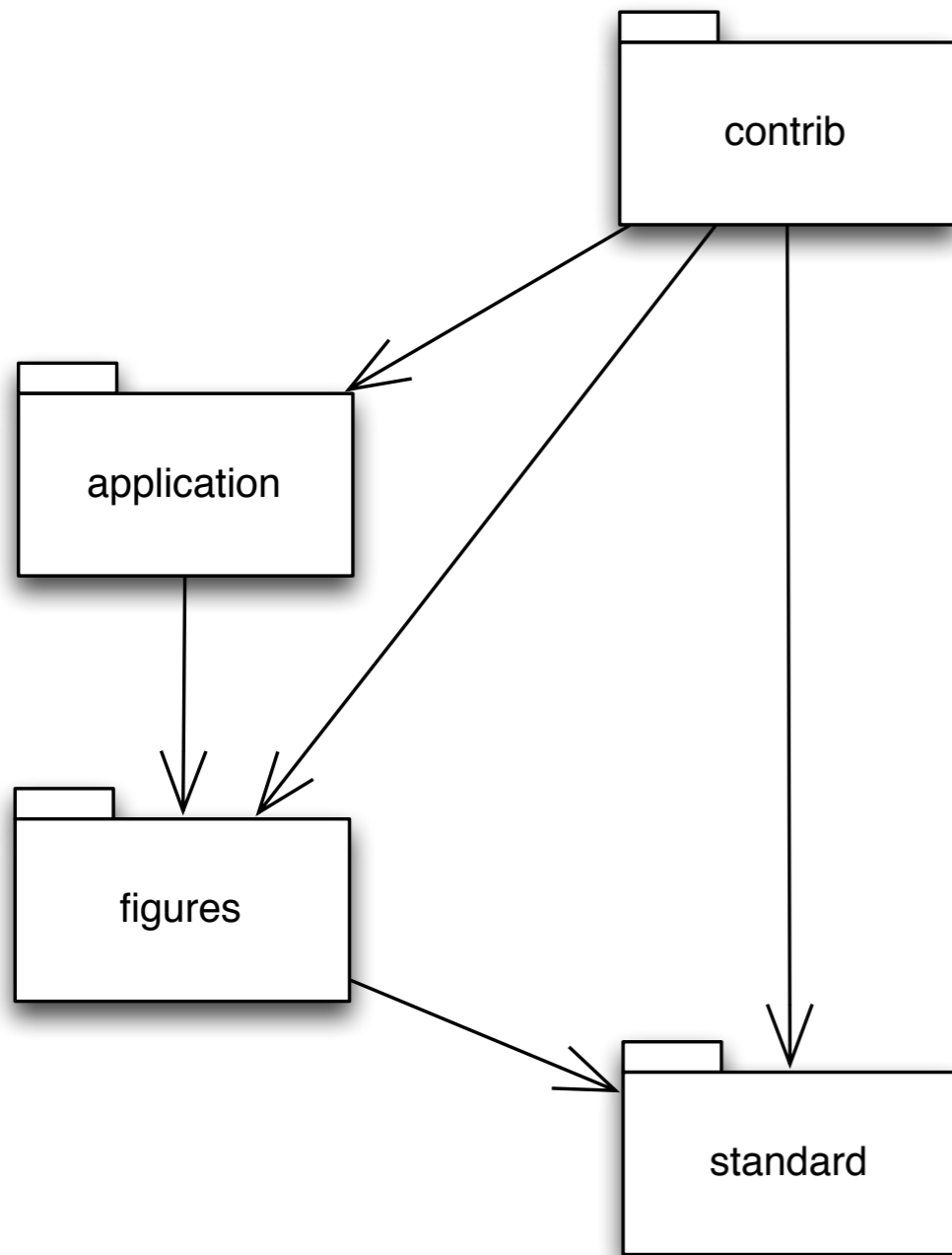
# Marea

A Tool for Breaking Dependency Cycles Between Packages




Master Project  
Bledar Aga

Software Composition Group  
University of Bern  
Fall 2014

# Dependency Cycles








# Problems

-  **Acyclic Design Principle**
-  **Maintenance and Testing costs**
-  **Modularity and Reuse**

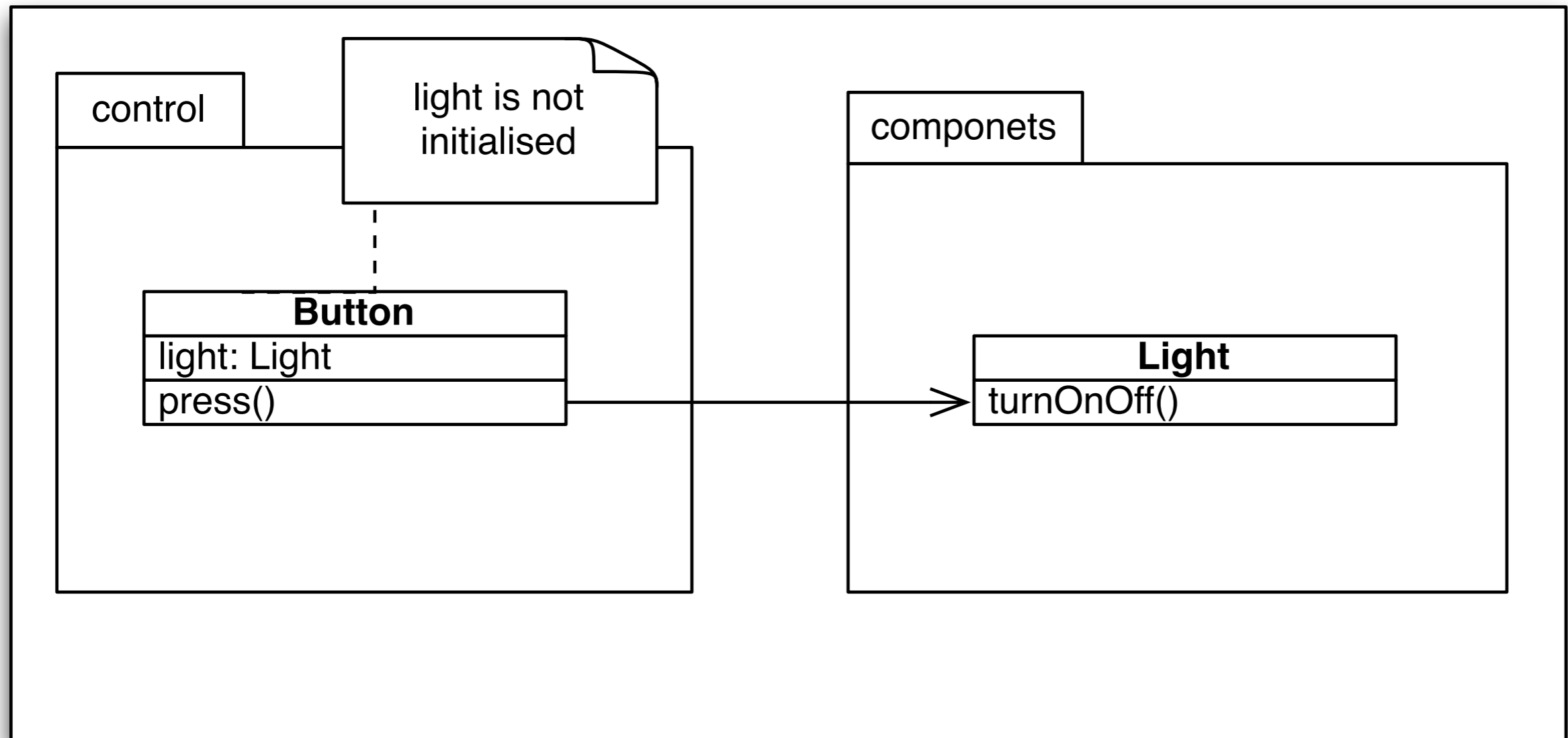
# Dependency Types

-  Reference
-  Inheritance
-  Invocation

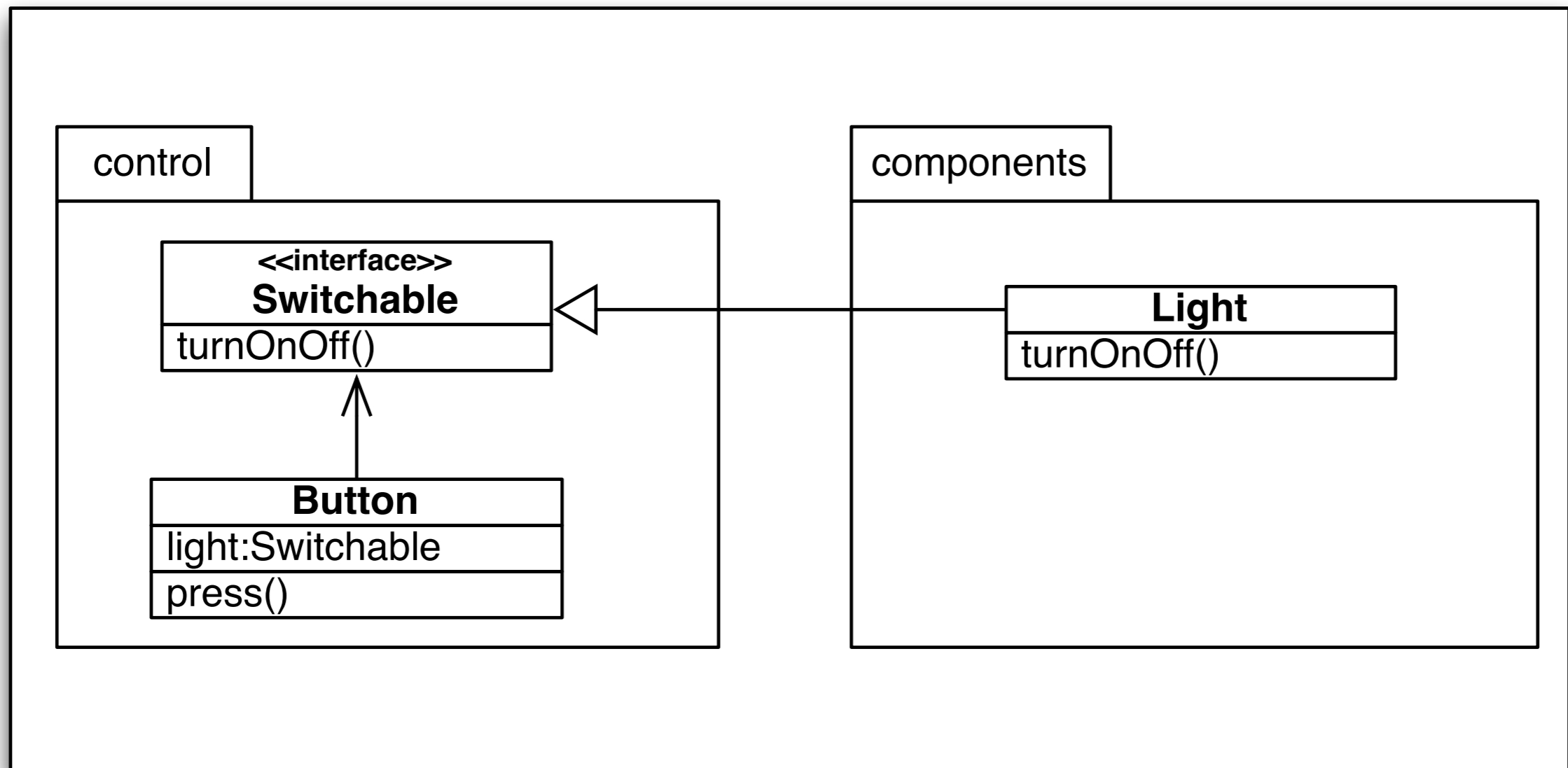
# Refactoring Methods

-  Move Class (*MC*)
-  Move Method (*MM*)
-  Using design patterns:
  -  Abstract Server Pattern (*ASP*)
  -  Dependency Injection (*DI*)

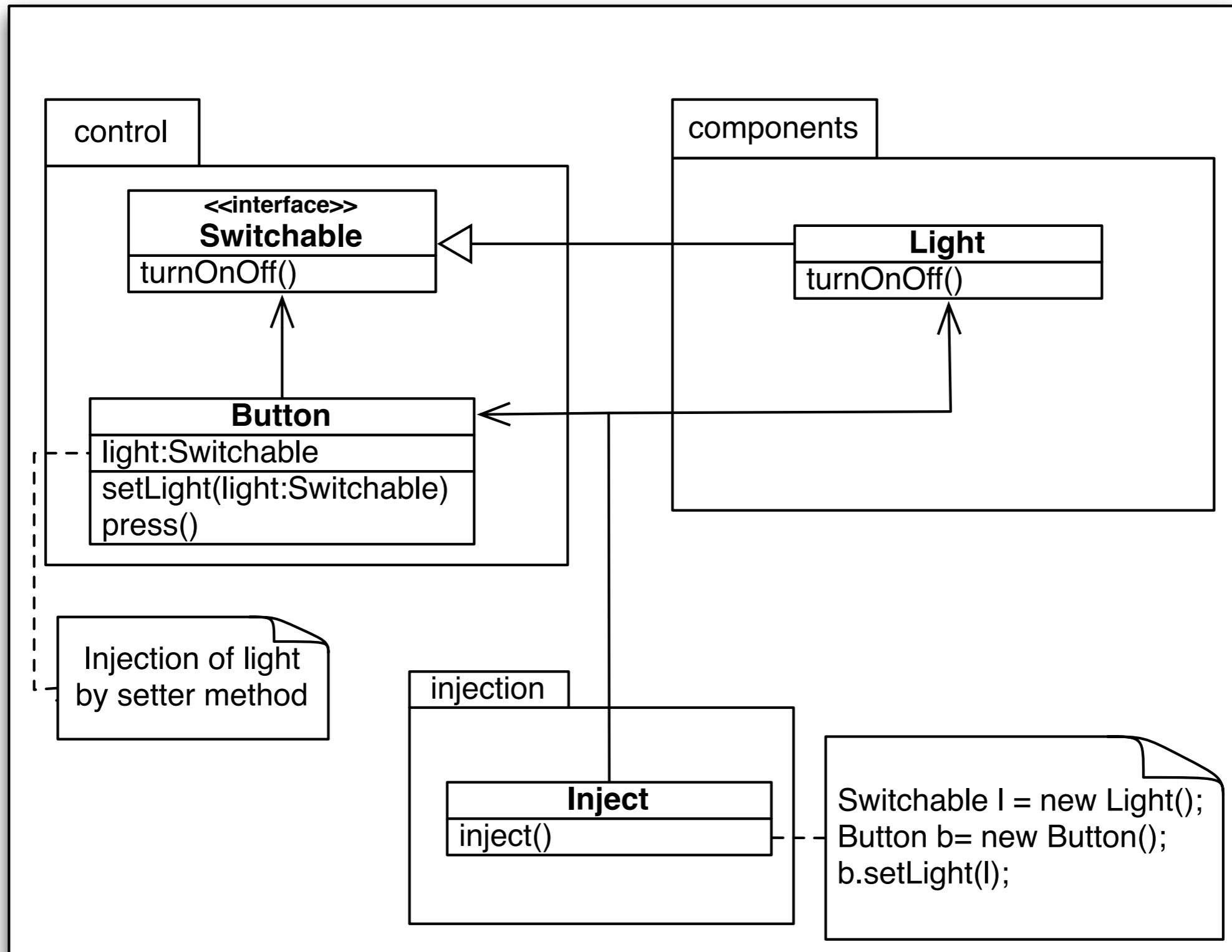
# ASP Example



# ASP Example



# ASP + DI Example



# Applicability of Refactoring

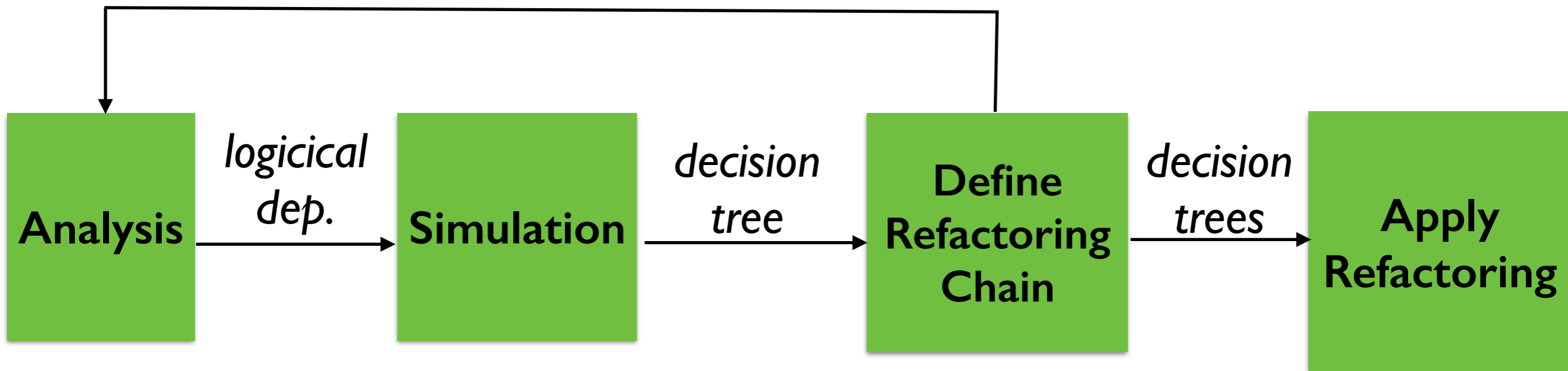
———	Move Class	Move Method	ASP	ASP+DI
Inheritance	Yes	No	No	No
Class Variable	Yes	No	Yes	No
Initialised Class Variable	Yes	No	No	Yes
Local Variable	Yes	Yes (Limited)	Yes	No
Initialised Local Variable	Yes	Yes (Limited)	No	Yes
Parameter	Yes	Yes (Limited)	Yes	No
Return Type	Yes	Yes (Limited)	Yes	No
Invocation	Yes	Yes (Limited)	Yes (Limited)	No



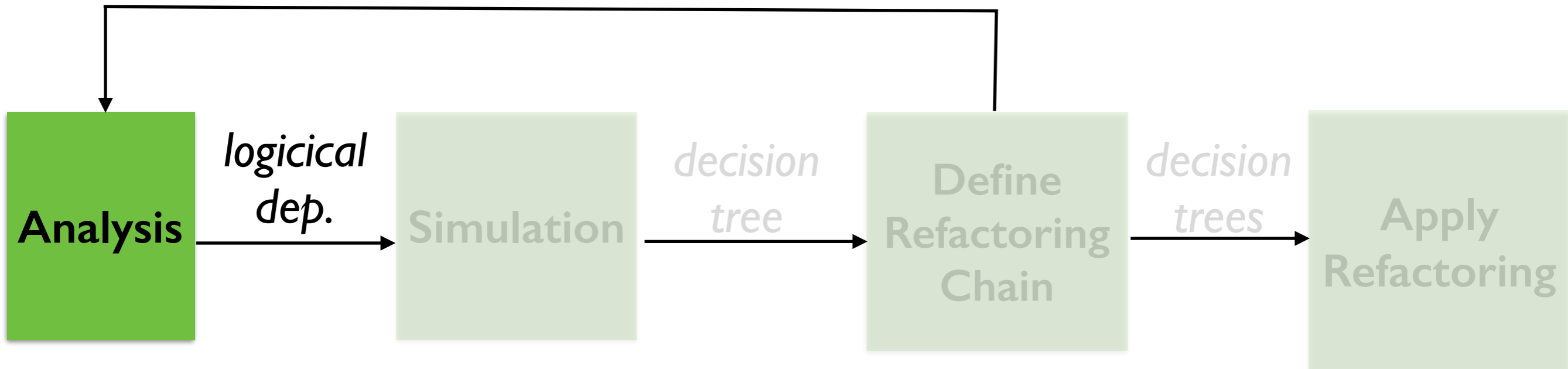
# Existing Tools

	Detection	Removal
JooJ	Yes	No
Jepends	Yes	No
Stan	Yes	No
Pasta	Yes	Yes
Lattix	Yes	Yes
Structure 101	Yes	Yes
ECOO	Yes	Yes

# Our Solution



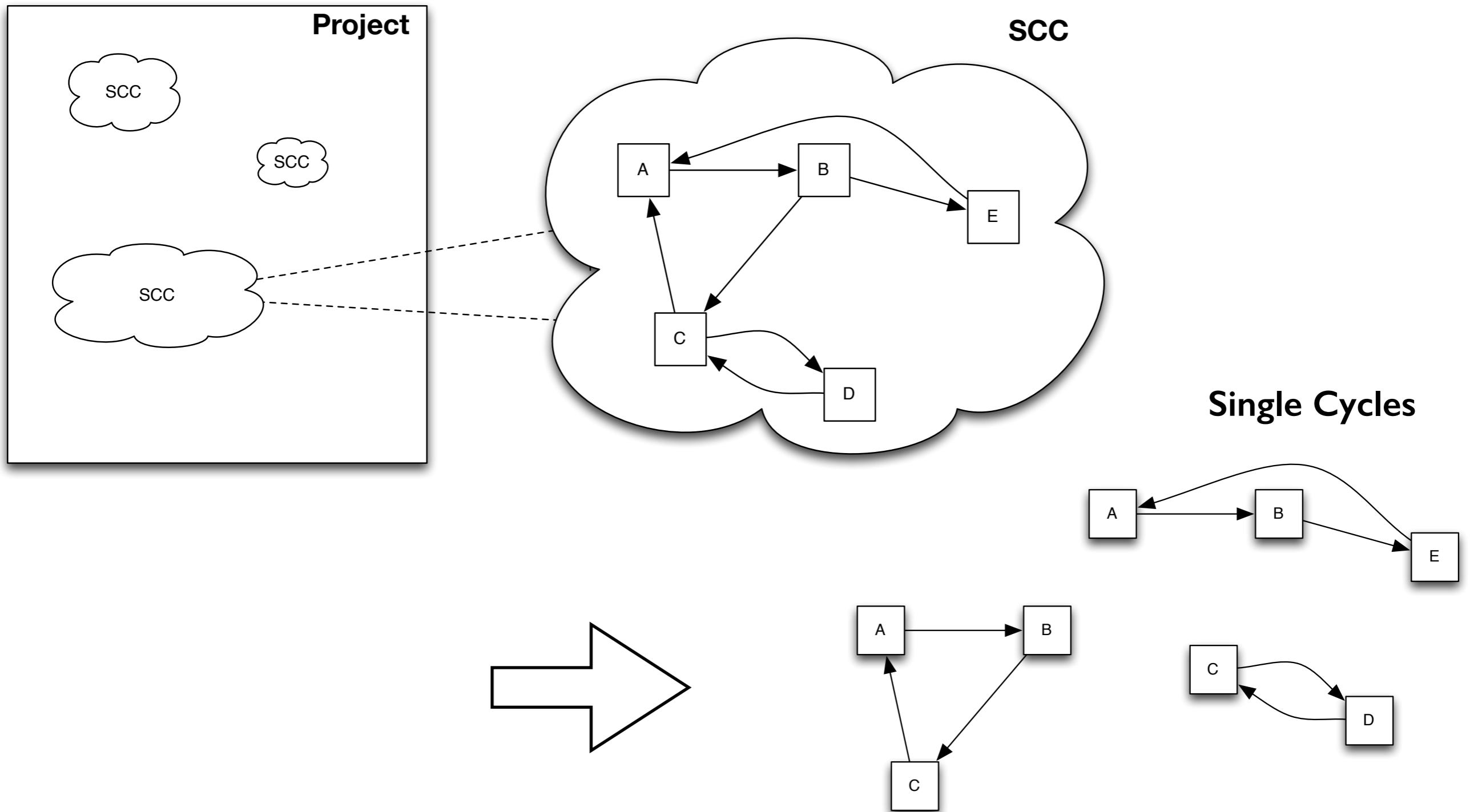
# Our Solution



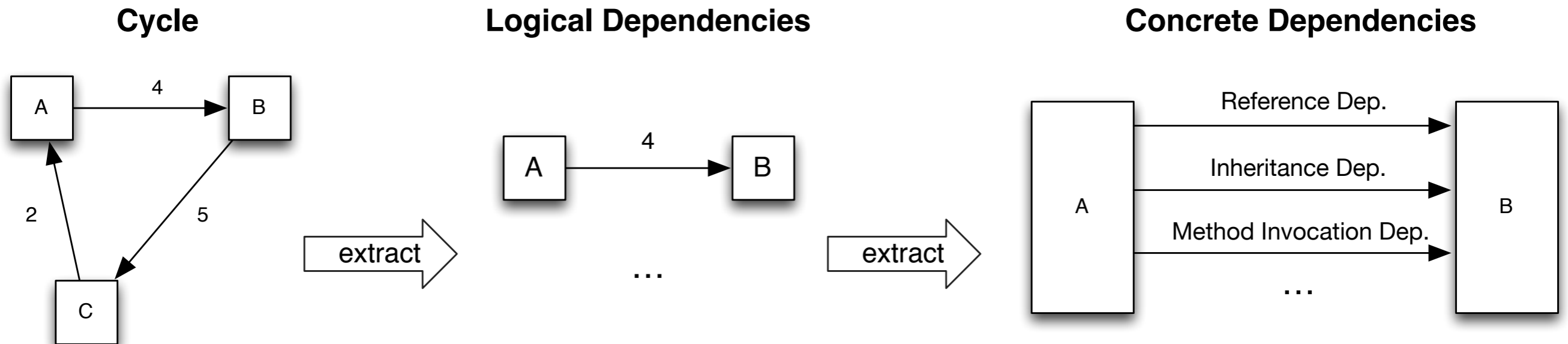
↻ Detect cycles

↻ Propose dependency to break

# Cycle Detection



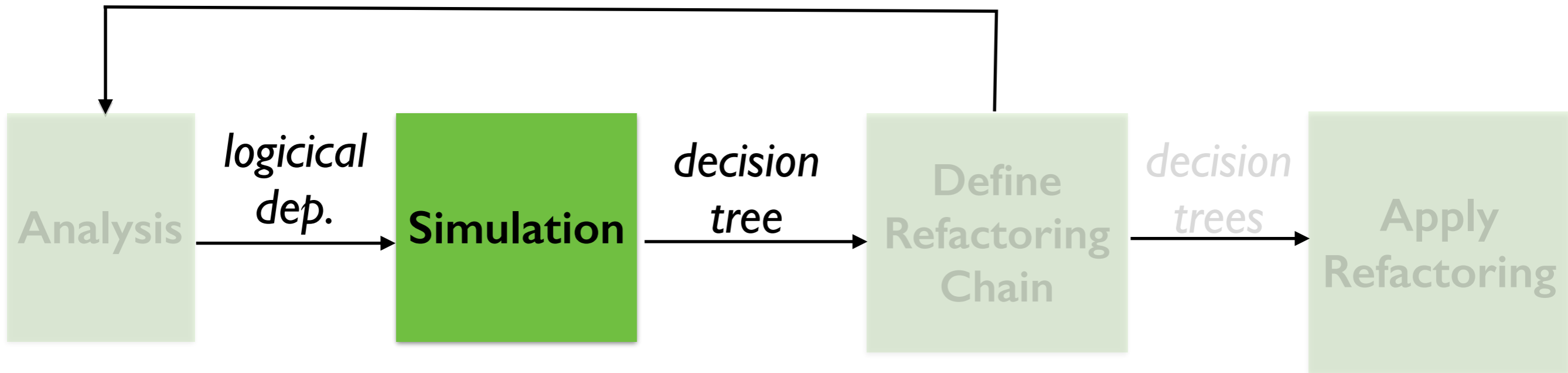
# Dependencies



Sort Logical Dependencies by:

% Shared Dependencies

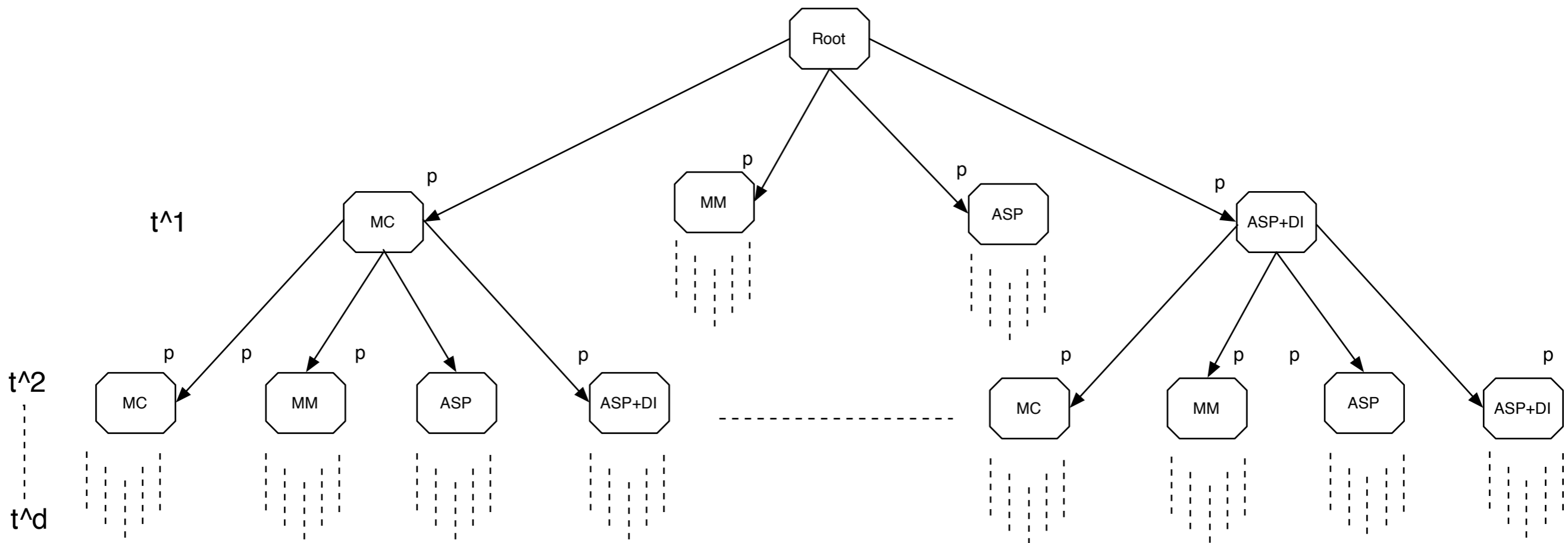
# Our Solution



↻ Simulate refactorings

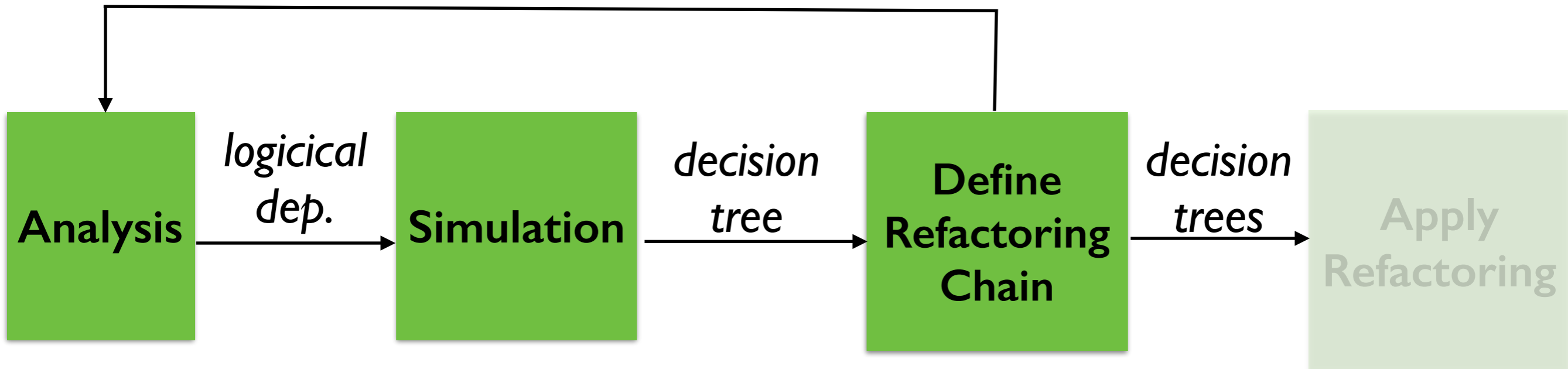
↻ Propose best refactoring sequence

# Decisional Tree



$$P = w_c \times \frac{1}{\# \text{ cycles} + 1} + w_d \times \frac{1}{\text{depth}} + w_i \times \frac{(1 - I_{from}) + (1 - I_{to})}{2} + w_a \times \frac{(A_{from}) + (A_{to})}{2}$$

# Our Solution



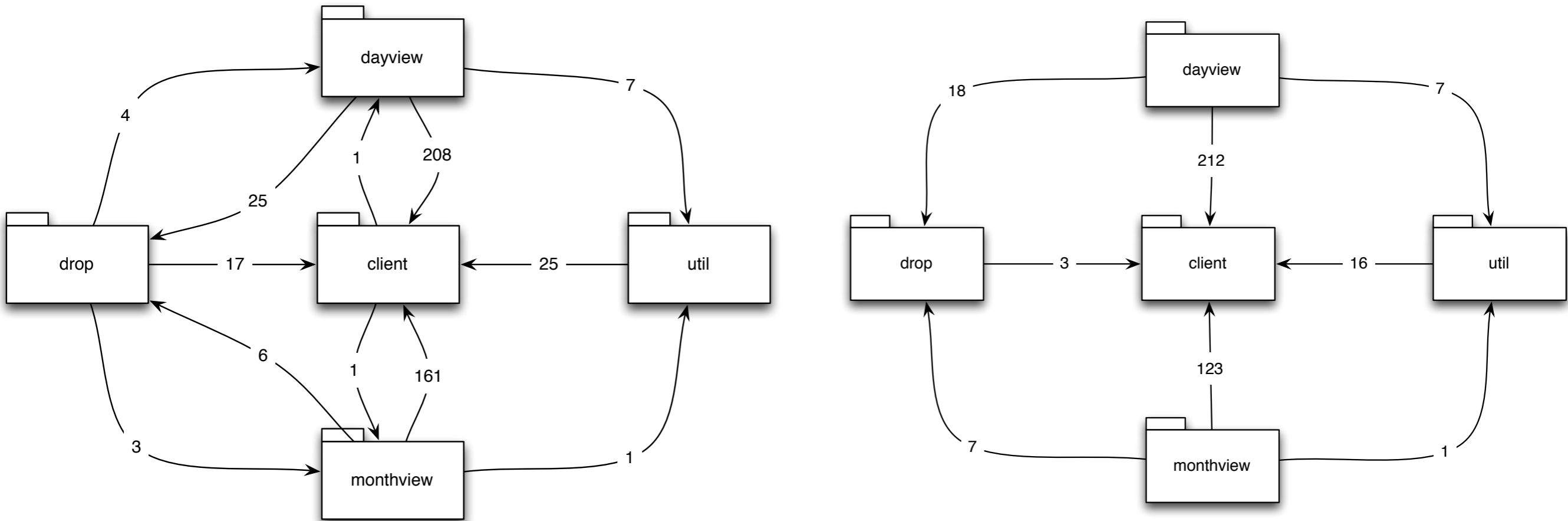
↻ Choice of the best path

↻ Repeat until no more cycles



**DEMO**

# Case study su GWT Calendar



7 cycles

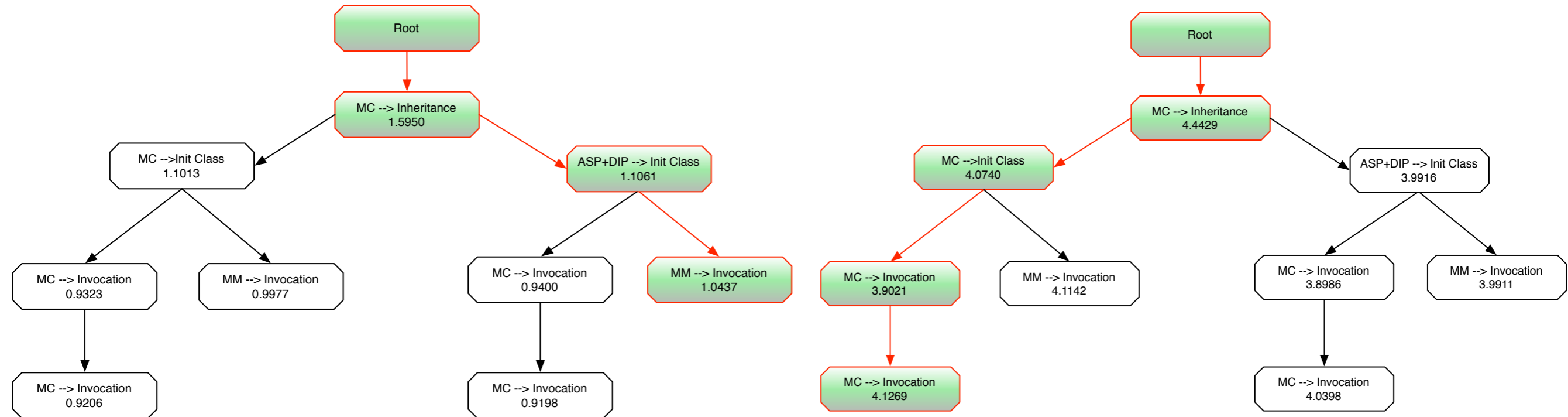
5 refactoring actions



4 iterations

0 cycles

# Case study su JHotDraw



stability + abstractness

stability

# Integration with Dicto

```
<?xml version="1.0"?>
<results id="1422311583">
  <rules>
    <rule id="1" failed="true" value="Test cannot contain cycles">
      <subrule id="1001" failed="true" value="Test cannot contain cycles " testedBy="DIMareaCycles">
        <error>
          <fix>Move Class [StandardDrawingView] from [org::jhotdraw::standard] to [org::jhotdraw::contrib]
            Move Class [DragNDropTool] from [org::jhotdraw::standard] to [org::jhotdraw::contrib::dnd]
            Move Class [javadraw] from [org::jhotdraw::contrib] to [org::jhotdraw::samples::javadraw]
            ...
          </fix>
          <cause>org.jhotdraw.test containCycles = True</cause>
          <details>org::jhotdraw::test - org::jhotdraw::test::samples::javadraw
            org::jhotdraw::test - org::jhotdraw::test::contrib
            org::jhotdraw::test - org::jhotdraw::test::samples::pert
            org::jhotdraw::test - org::jhotdraw::test::figures
            org::jhotdraw::test - org::jhotdraw::test::standard
            org::jhotdraw::test - org::jhotdraw::test::util
          </details>
        </error>
      </subrule>
    </rule>
  </rules>
</results>
$
```

# Future Work

- ↻ User interface interaction
- ↻ Export the decisional trees in a file
- ↻ Automate the refactoring decision integrating with Eclipse

# Summary

- ↻ Dependency cycles difficult topic
- ↻ Without a proper tool support engineers cannot solve the problem
- ↻ Marea assistes engineers in resolving dependency cycles



Smalltalkhub: **BledarAga/Marea**