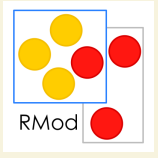
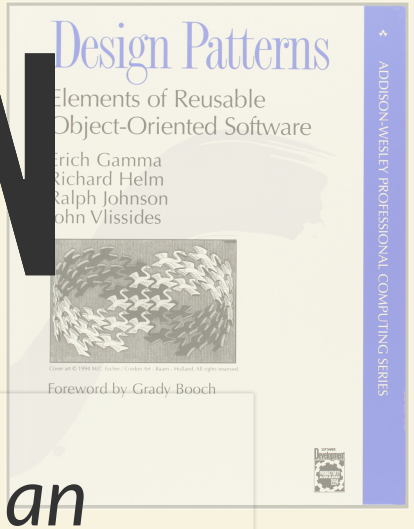
EMPIRICAL STUDY OF PROGRAMMING TO AN INTERFACE

B. Verhaeghe[1], **Christopher Fuhrman**[2],
L. Guerrouj[2], N. Anquetil[3] and S. Ducasse[4]

[1]Berger-Levrault, France
[2]École de technologie supérieure, Montreal, Canada
[3]University of Lille, CRIStAL
[4]RMoD - Inria Nord Europe



MOTIVATION



"Program to an interface, not an implementation."

Heuristic from *Design Patterns* book (GoF 1994).

Clients are less fragile if implementations change.

We have been teaching this to students for decades, but is there any empirical evidence to support it?

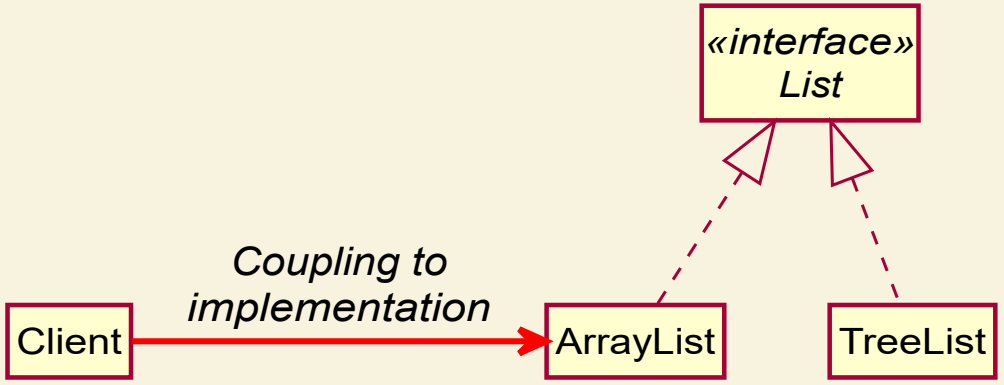
RELATED PRINCIPLES

- *Protected Variation* (C. Larman)
- *Open-Closed Principle* (B. Meyer)
- *Information Hiding* (D. Parnas)

WHAT IT LOOKS LIKE

Beginner design in Java

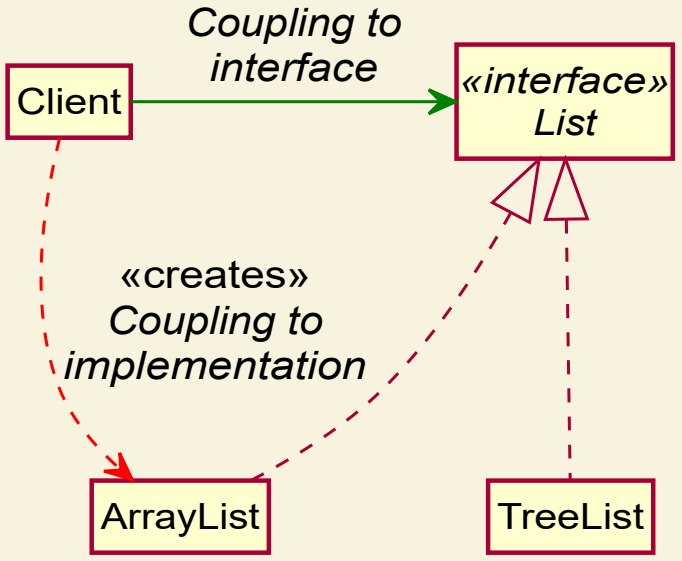
```
ArrayList myList = new ArrayList();
```



WHAT IT LOOKS LIKE

Improved design…

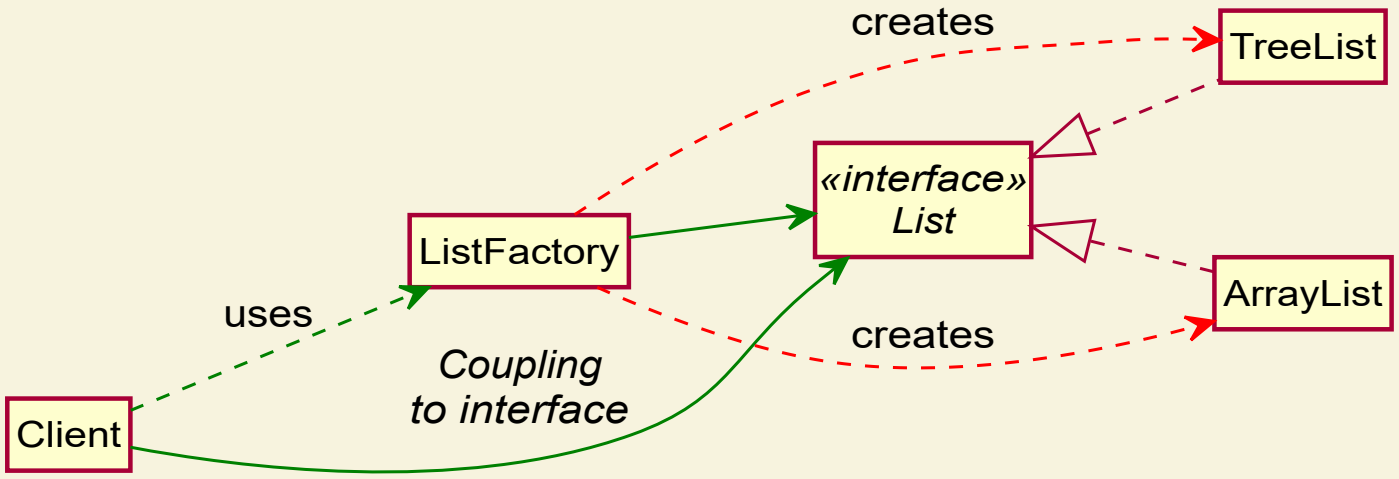
```
List myList = new ArrayList();
```



WHAT IT LOOKS LIKE

Factory design decouples client from implementation

```
List myList = ListFactory.create("...");
```



CONSEQUENCES OF PTI

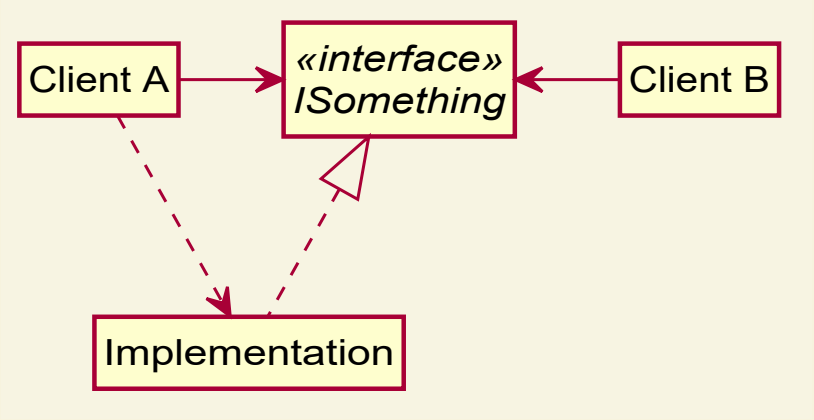
- Protect clients from changing implementation(s)
- Favor extensions to implementations
- Simplify the API
- Overcome single-inheritance limitation
- *Speculative Generality* (code smell)
- Additional level of hierarchy
- Factories decouple clients from implementations
- Changed interface breaks clients *and* implementations

EMPIRICAL STUDY OF THE CONSEQUENCES

- Human studies: qualities such as understandability, ease of extensibility, etc.
- Observational studies: Code repositories to see if client code is protected (change propagation), fault location, etc.

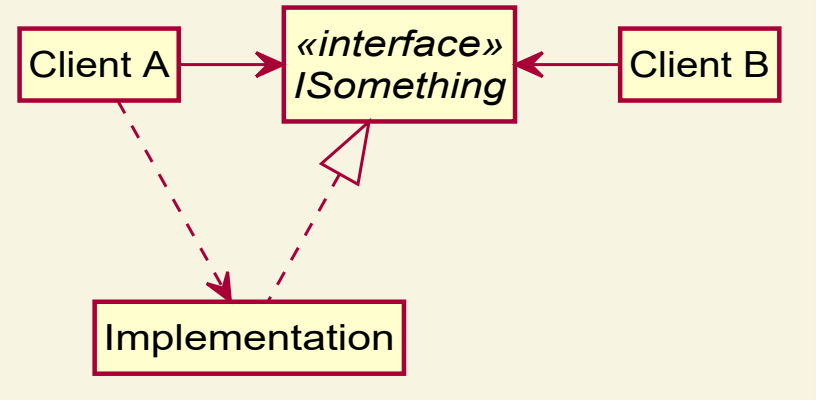
OBSERVATIONAL STUDY: PROTECTED VARIATION

- Cochange in Java projects (GitHub)
- Static analysis identifies cochange pairs (client, implementation)
  - Unprotected pairs (direct coupling)
  - Protected pairs (no coupling)



RESEARCH QUESTION

Do unprotected pairs cochange more often than protected pairs?



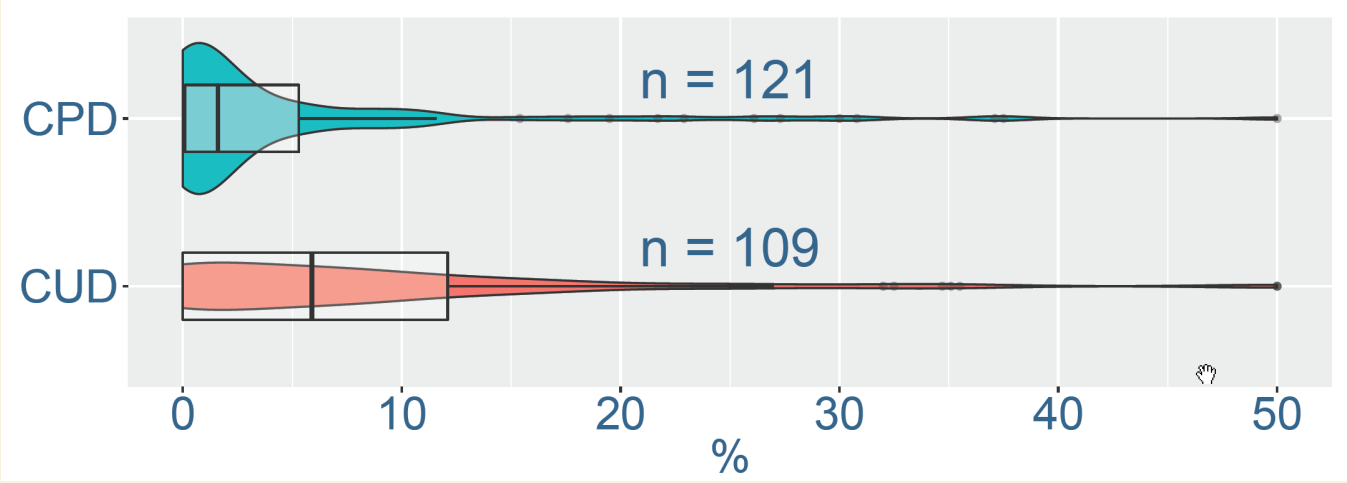
COCHANGE HISTORY

H is the history of commits.

- Cochanged Protected Dependencies ratio
  $CPD(\%) = \frac{|H \cap P|}{|P|}$
- Cochanged Unprotected Dependencies ratio
  $CUD(\%) = \frac{|H \cap U|}{|U|}.$

RESULTS

126 (popular) Java projects on GitHub



Ratios of CPD and CUD

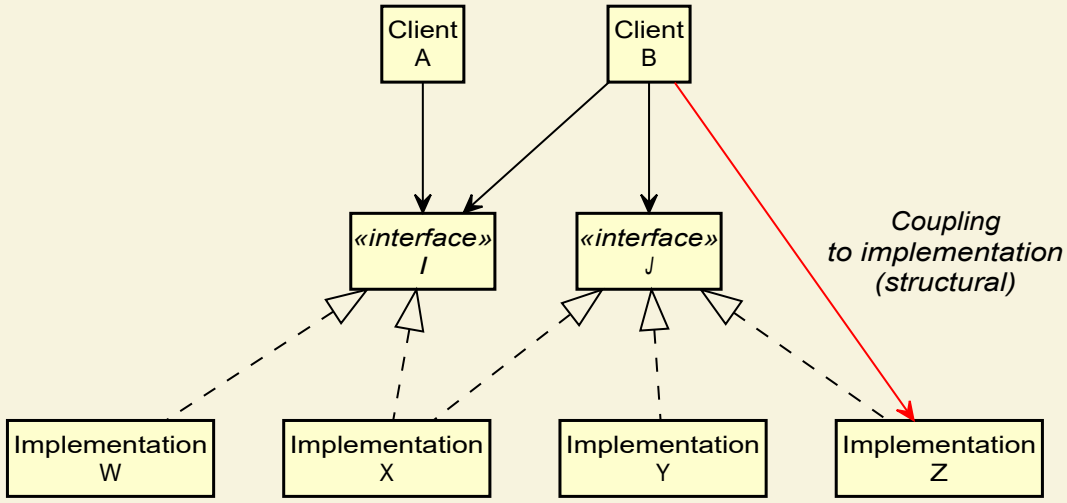
$p < 0.001$ and effect size (Cliff's) $d = 0.32$ (small)

FUTURE WORK

- Observational studies with more context
  - Classify projects (abstractness)
  - Microarchitectures (e.g., factories)
- Natural experiments for quality aspects

BACKUP SLIDES

CLIENT-IMPLEMENTATION PAIRS



Pair-oriented cochange analysis [Geipel12, Ajienka17]

Protected vs. *Unprotected* client-implementation pairs

$$P = (A \rightarrow W)(A \rightarrow X)(B \rightarrow W)(B \rightarrow X)(B \rightarrow$$
$$U = (B \rightarrow Z)$$

OTHER FORMS

- Facade pattern
- Dependency injection frameworks
- etc.