Summer Semester 2015

Software Engineering Design & Construction

Dr. Michael Eichberg
Fachgebiet Softwaretechnik
Technische Universität Darmstadt

Interface Segregation Principle

***I**nterface **S**egregation **P**rinciple*

Clients should not be forced to depend on methods that they do not use.

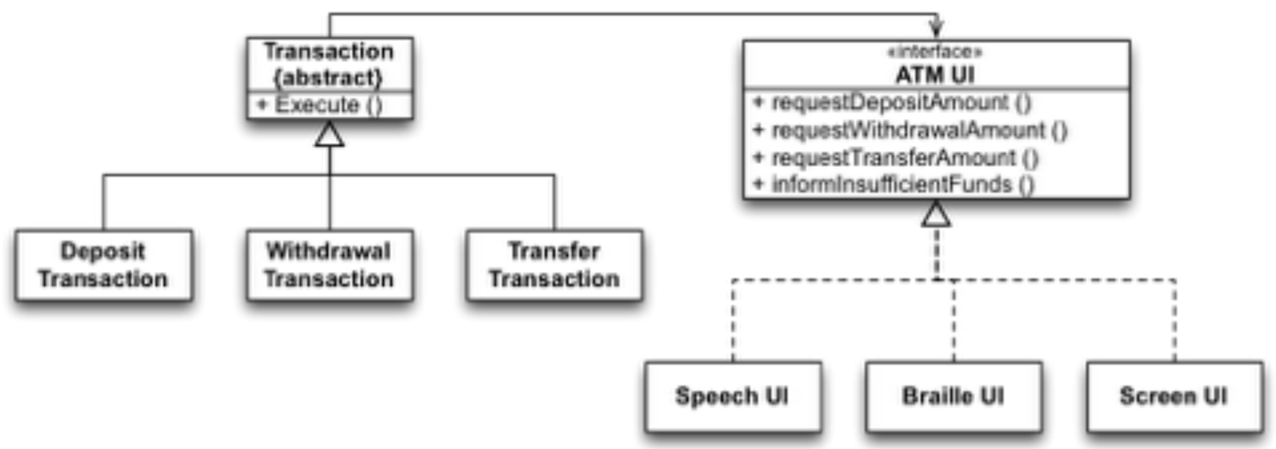
–Agile Software Development; Robert C. Martin; Prentice Hall, 2003

Introduction by Example

- Consider the development of software for an automatic teller machine (ATM):
  - Support for the following types of transactions is required: **withdraw**, **deposit**, and **transfer**.
  - Support for different **languages** and support for different **kinds of UIs** is also required
  - Each transaction class needs to call methods on the GUI
    E.g., to ask for the amount to deposit, withdraw, transfer.

Introduction by Example

- Initial design of a software for an automatic teller machine (ATM):



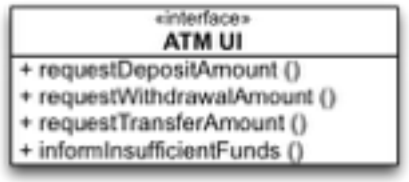
What do you think?

ISP tells us to avoid this. Each transaction class uses a part of the interface, but depends on all others. Any change affects all transactions.

A Polluted Interface

ATM UI is a polluted interface!

- It declares methods that do not belong together.
- It forces classes to depend on unused methods and therefore depend on changes that should not affect them.
- ISP states that such interfaces should be split.

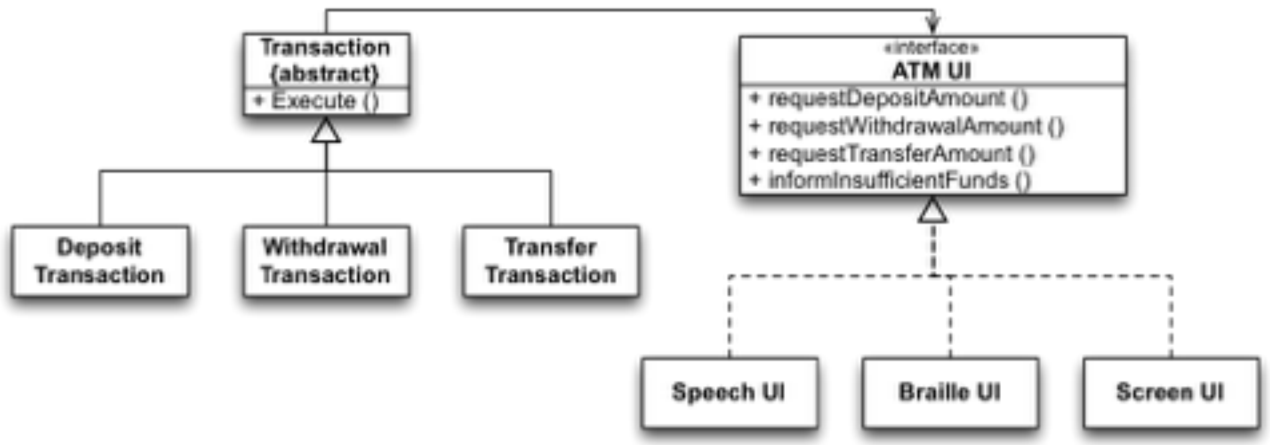

«interface»
ATM UI
+ requestDepositAmount ()
+ requestWithdrawalAmount ()
+ requestTransferAmount ()
+ informInsufficientFunds ()

The Rationale Behind ISP

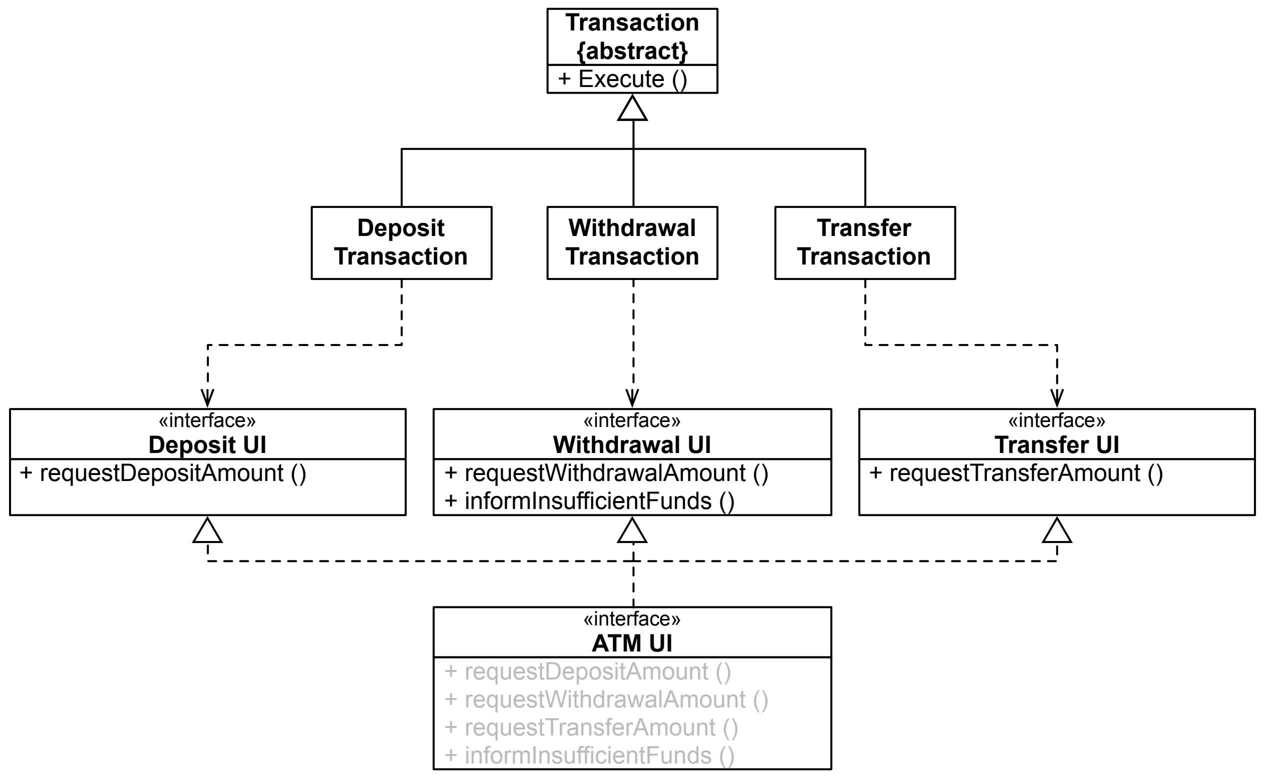
When clients depend on methods they do not use, they **become subject to changes forced upon these methods** by other clients.

This causes coupling between all clients!

How does an ISP compliant solution look like?



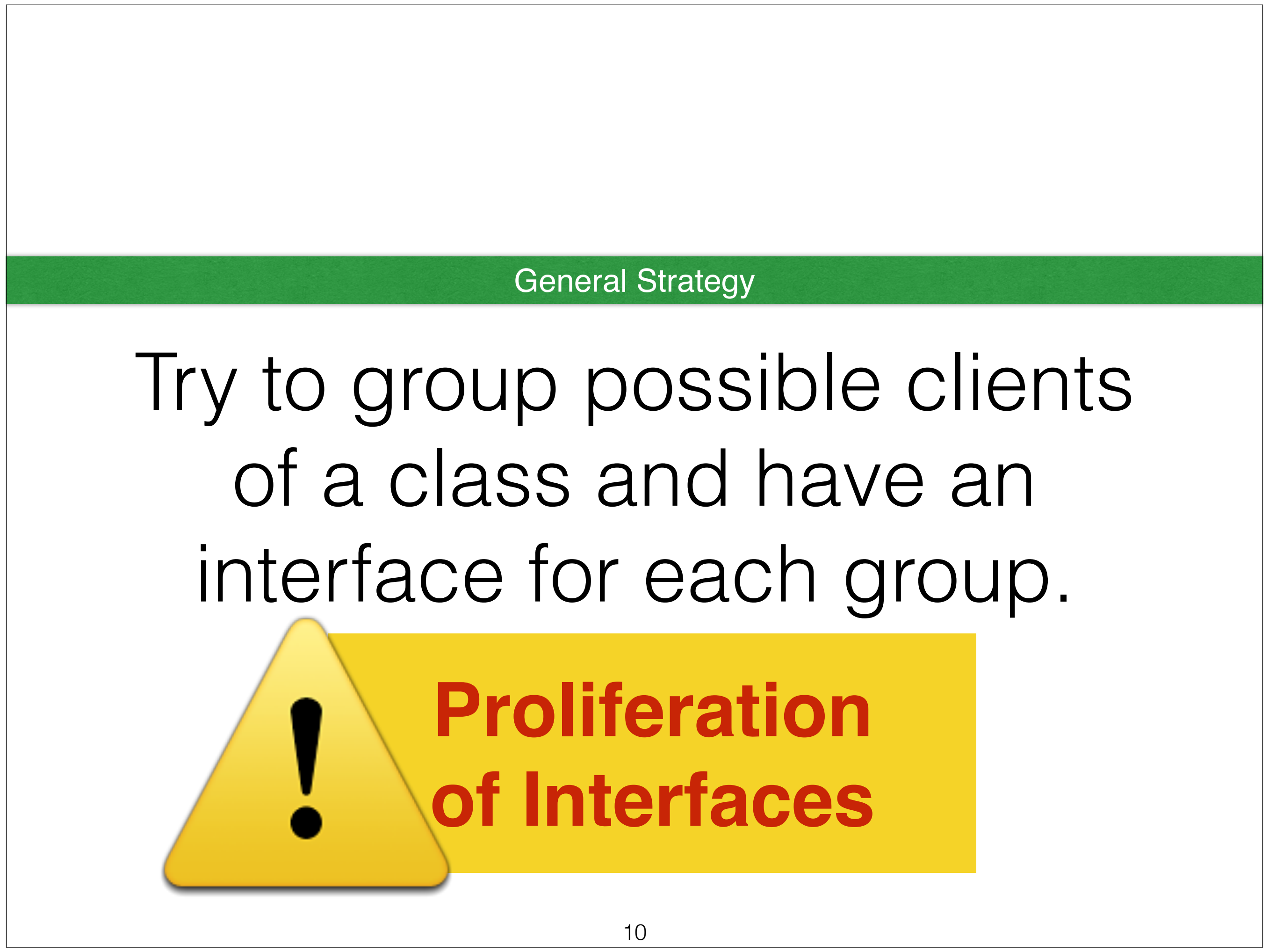
An ISP Compliant Solution



Transaction {abstract}
+ Execute ()
Deposit Transaction
Withdrawal Transaction
Transfer Transaction
«interface» Deposit UI
+ requestDepositAmount ()
«interface» Withdrawal UI
+ requestWithdrawalAmount ()
+ informInsufficientFunds ()
«interface» Transfer UI
+ requestTransferAmount ()
«interface» ATM UI
+ requestDepositAmount ()
+ requestWithdrawalAmount ()
+ requestTransferAmount ()
+ informInsufficientFunds ()

General Strategy

Try to group possible clients of a class and have an interface for each group.



General Strategy

Try to group possible clients of a class and have an interface for each group.

Proliferation of Interfaces

Segregating interfaces should not be overdone!

If you overdue the application of the interface segregation principle, you will end up with 2n-1 interfaces for a class with n methods.

Recall that, in general, a class implementing many interfaces may be a sign of a violation of the single-responsibility principle.

***I**nterface **S**egregation **P**rinciple*

Clients should not be forced to depend on methods that they do not use.

–Agile Software Development; Robert C. Martin; Prentice Hall, 2003