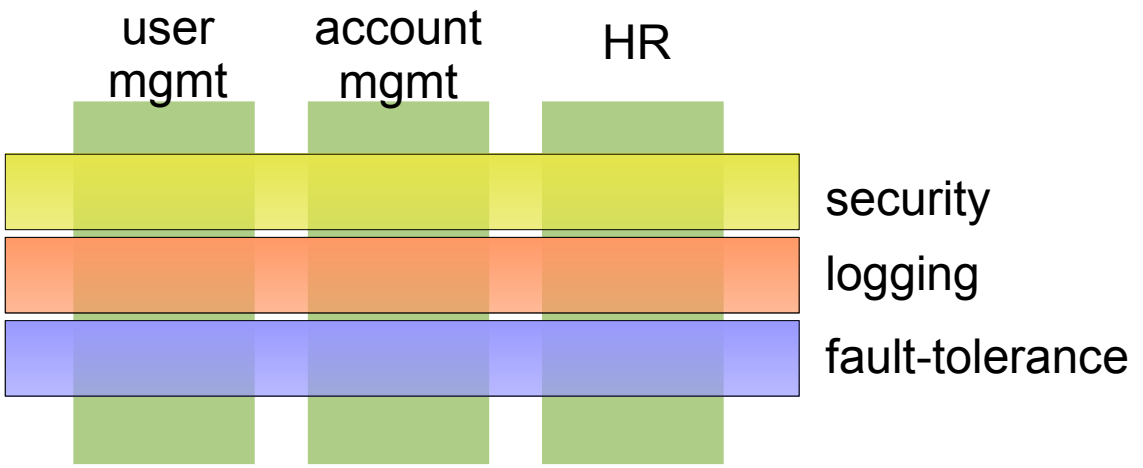
Introduction to
Aspect Oriented Programming (AOP)

AOP

Goal

- separation of cross-cutting concerns
  - organization leading to each component doing one thing only

Cross-cutting concern



Cross-cutting Concerns

Error detection	*Validation*
Security	*Business rules*
Caching	*i18N*
Logging	*Persistence*
Monitoring	*Transactions*

[1] http://en.wikipedia.org/wiki/Aspect-oriented_software_development#Examples_of_crosscutting_concerns

Motivation

Tangling

- occurs if the implementation (code) of multiple concerns is intermixed in a single module

Tangling Example

```
synchronized void put (SensorRecord rec ) {
// Check that there is space in the buffer; wait if not

if ( numberOfEntries == bufsize)
wait () ;

// Add record at end of buffer

// If at end of buffer, next entry is at the beginning

// indicate that buffer is available

notify () ;

}
```

Motivation

Scattering

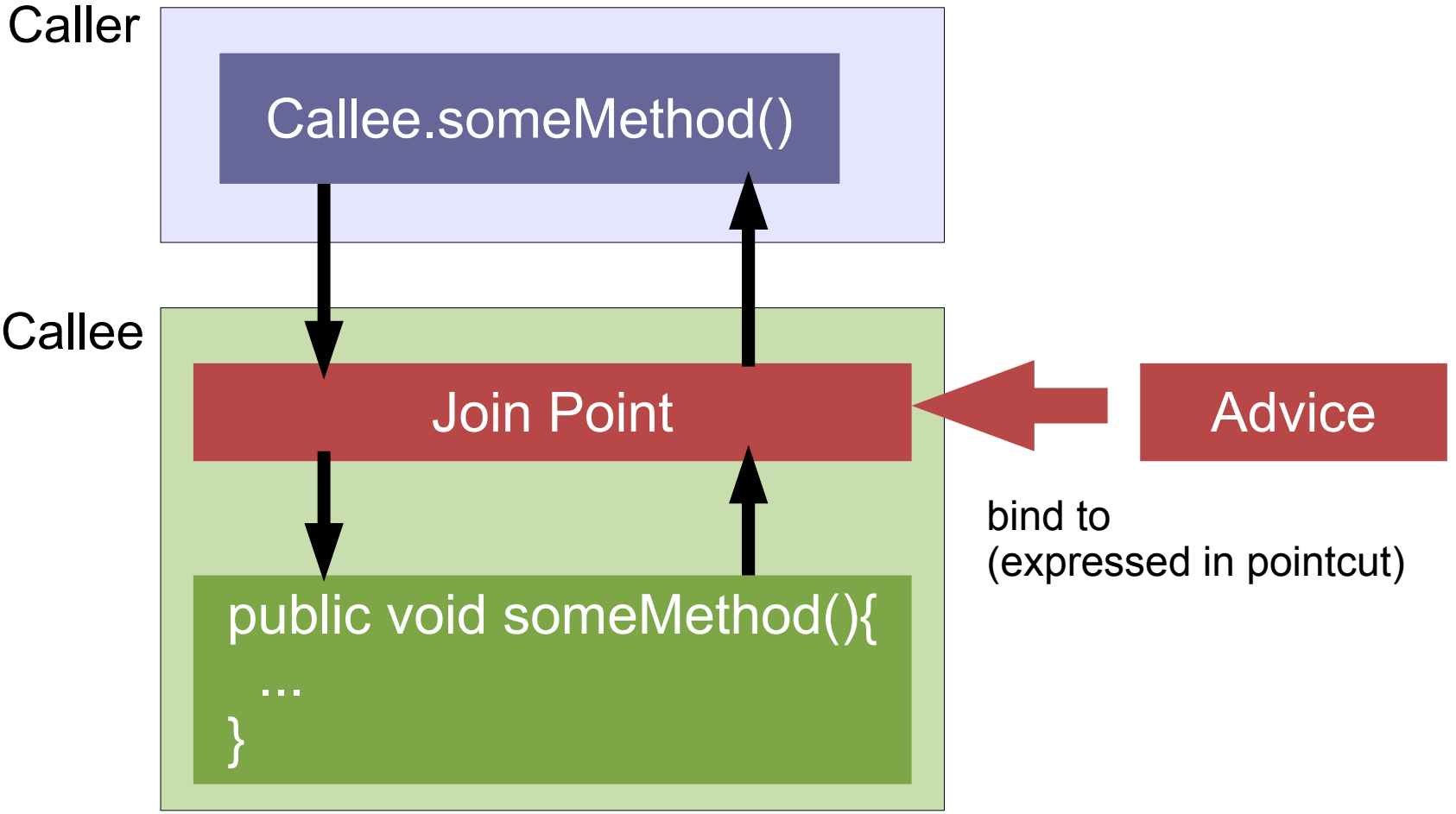
- occurs if the implementation of a single concern is spread over multiple modules

Scattering Example

```
public void service1 (  ) {

    try{

    // service1 code

    } catch (Exception e) {
        handleException(e);

    }

}
```

```
public void service2 (  ) {

    try{

    // service2 code

    } catch (Exception e) {
        handleException(e);

    }

}
```

Core Concepts



Core Concepts

advice

- the code implementing a concern
  - additional code that should be applied to existing core functionality
  - e.g.
    - logging
    - security

Core Concepts

join point

- an event in an executing program where the advice associated with the aspect may be executed

join point model

- many possible types of events
  - call events – calls to a method
  - execution events – the execution of a method
  - initialization events – class or object initialization
  - data events – accessing or updating fields
  - exception events

Core Concepts

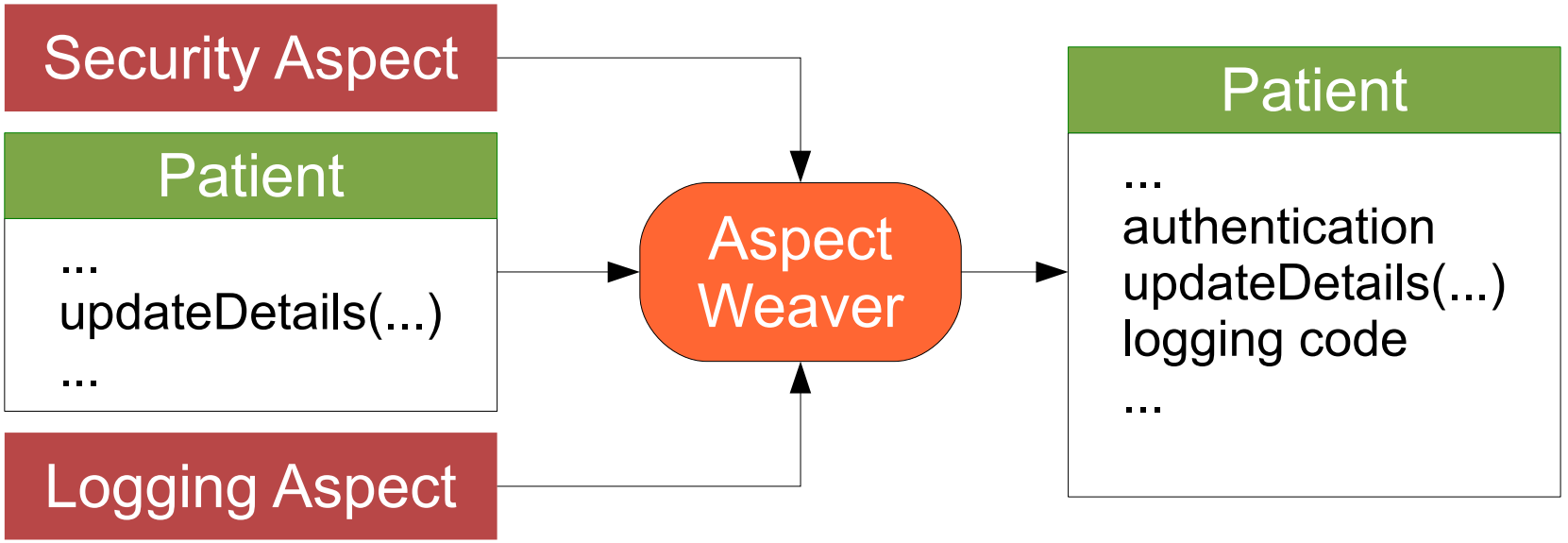
pointcut

- defines the join point where the associated advice should be executed
  - events are associated with particular items
- examples
  - before the execution of all *update methods*
  - after a normal or exceptional return from a method
  - when the *name field* is changed

Core Concepts

aspect

- A program abstraction that defines a cross-cutting concern. it includes
  - the definition of a pointcut
  - the advice associated with that concern



Core Concepts

weaver

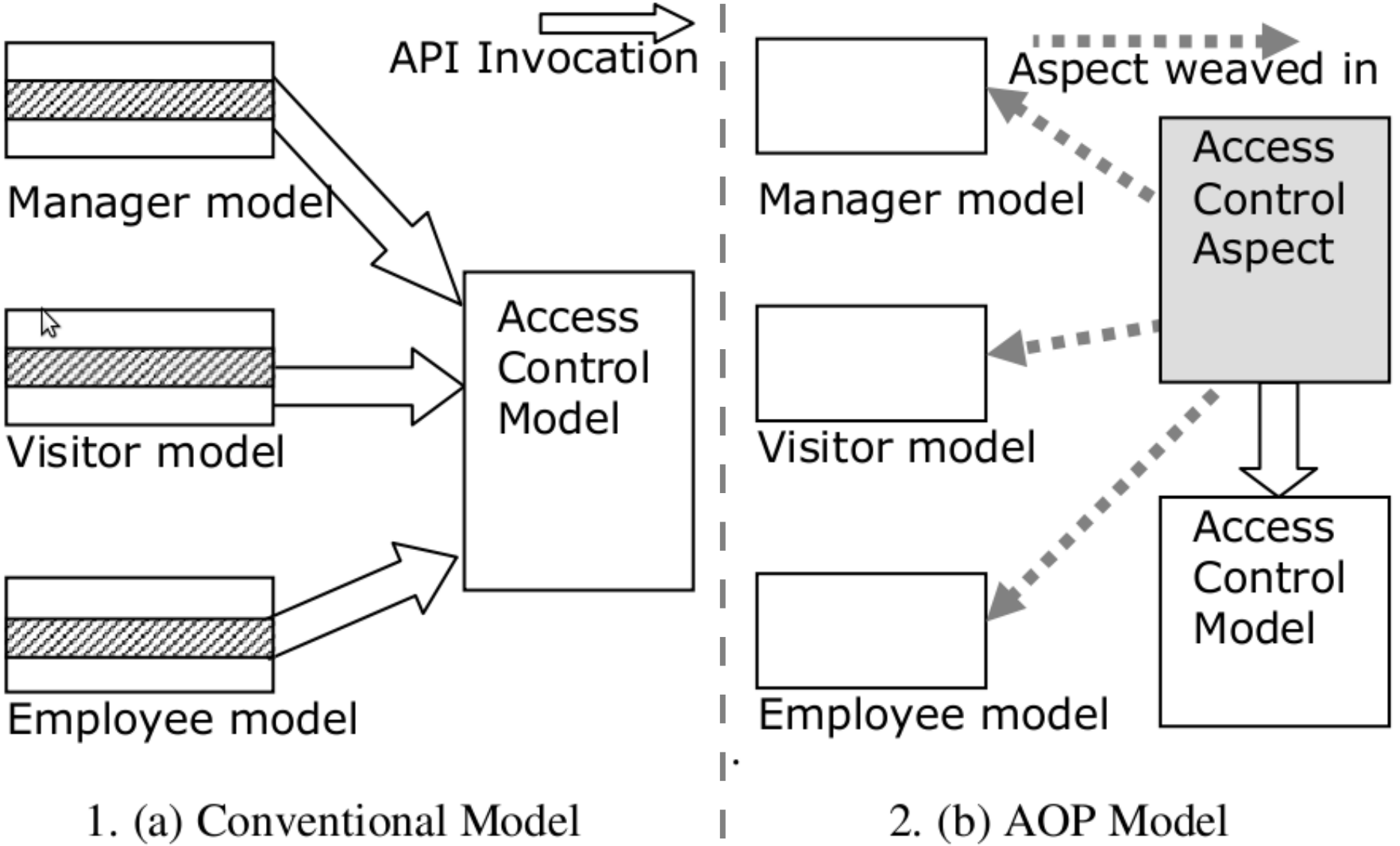
- responsible for inclusion of advice at the join points specified in the pointcuts
- three types of weaving
  - source code preprocessing
    - code to code translation first, then compilation
  - link time
    - most common
  - dynamic
    - join points are monitored and corresponding advices are integrated
    - performance penalties

Interfaces??

Conventional design

- a cross cutting concern can be modularized using interfaces
  - decouples the implementation
  - example: Log4J
- however, the client code still needs to embed interface code

Example



1. (a) Conventional Model

2. (b) AOP Model

Aspect Oriented Design

Intent

- design process that uses aspects

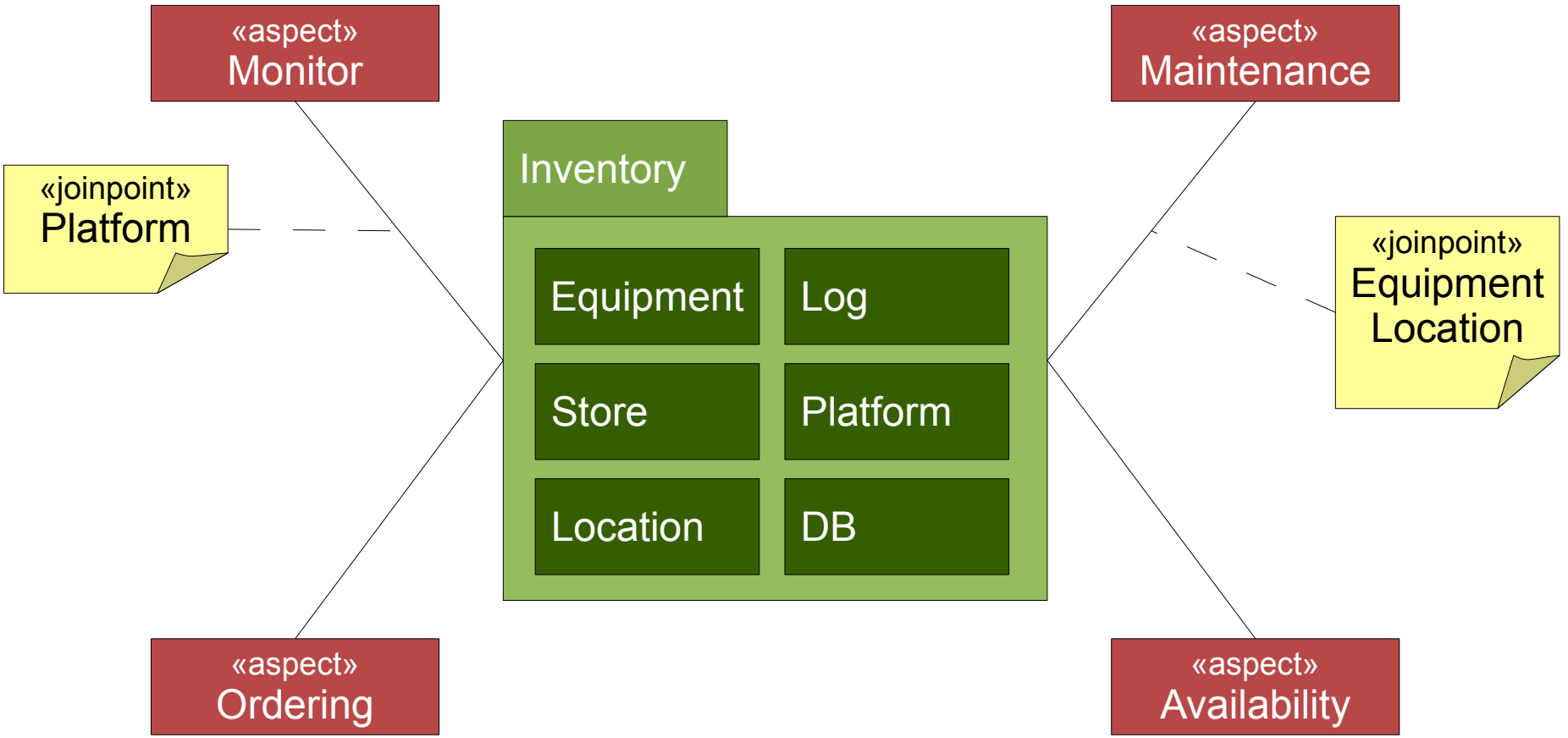
Identification

- start with use case diagrams
- look for common features

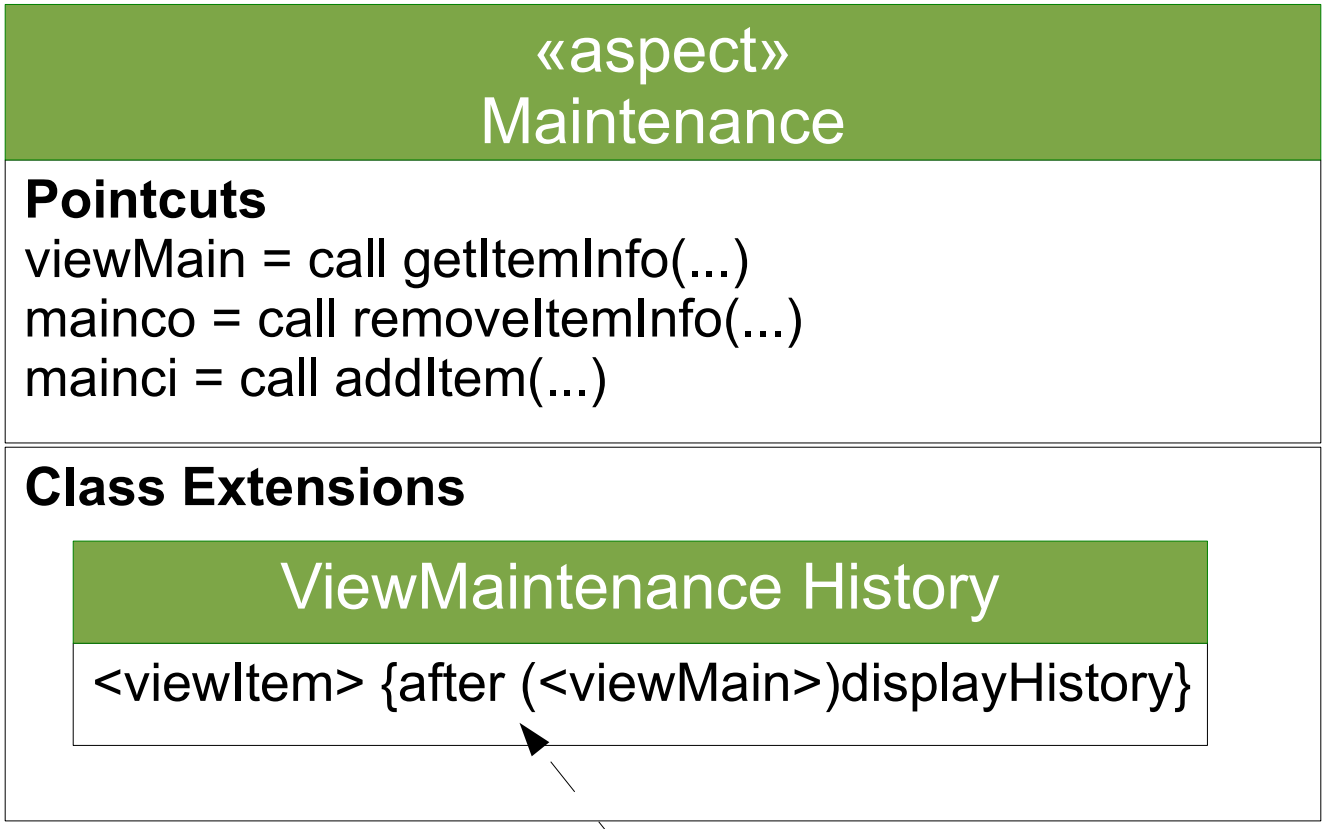
Design

- the outcome of AOD process is an AOD model

Example of AOD model



Example of AOD model



In the method viewItem, after the call to the method getItemInfo, a call to the method displayHistory should be included to display the maintenance record.

Challenges

Testing

- unit testing – easy
- woven code – not so easy
  - depends on the combination of aspects
    - e.g. weave order

Tight coupling

- between main code (concern) and aspect code
- correctness
  - correctly specifying pointcuts is important
- matching
  - like regex

Challenges

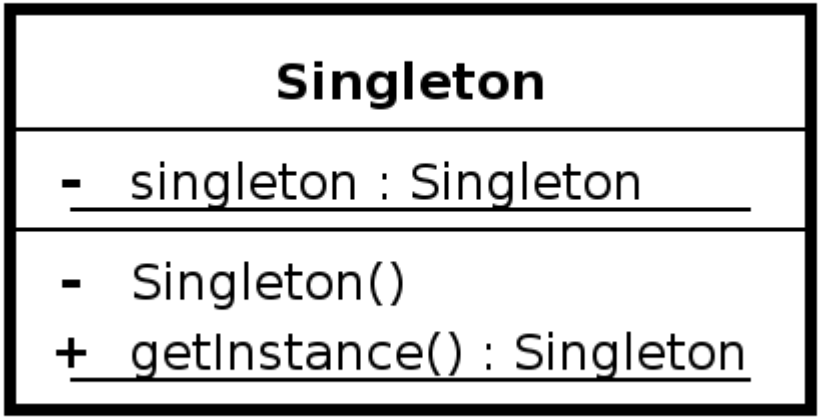
Evolution

- FACT: code evolves/changes with time
- aspects must also change **correspondingly**

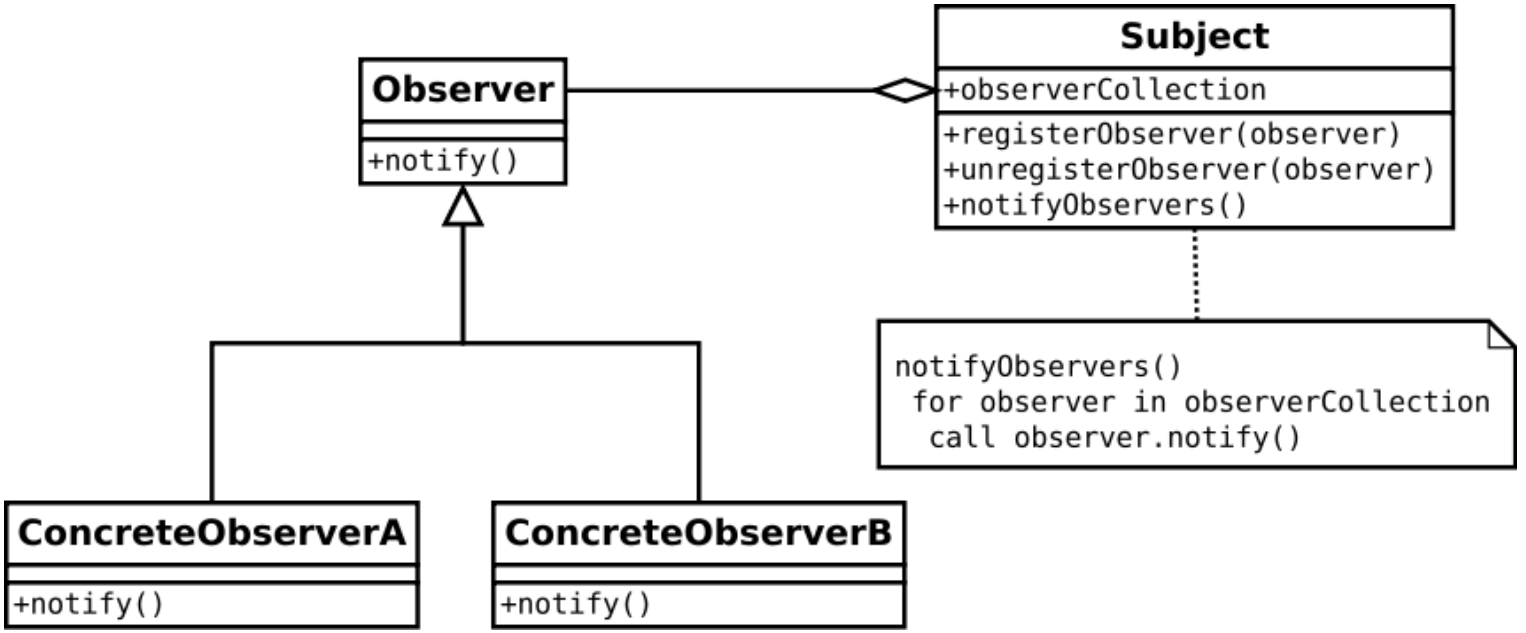
Design Patterns & AOP

Singleton Pattern

- crosscutting concerns:
  - object creation
  - count management



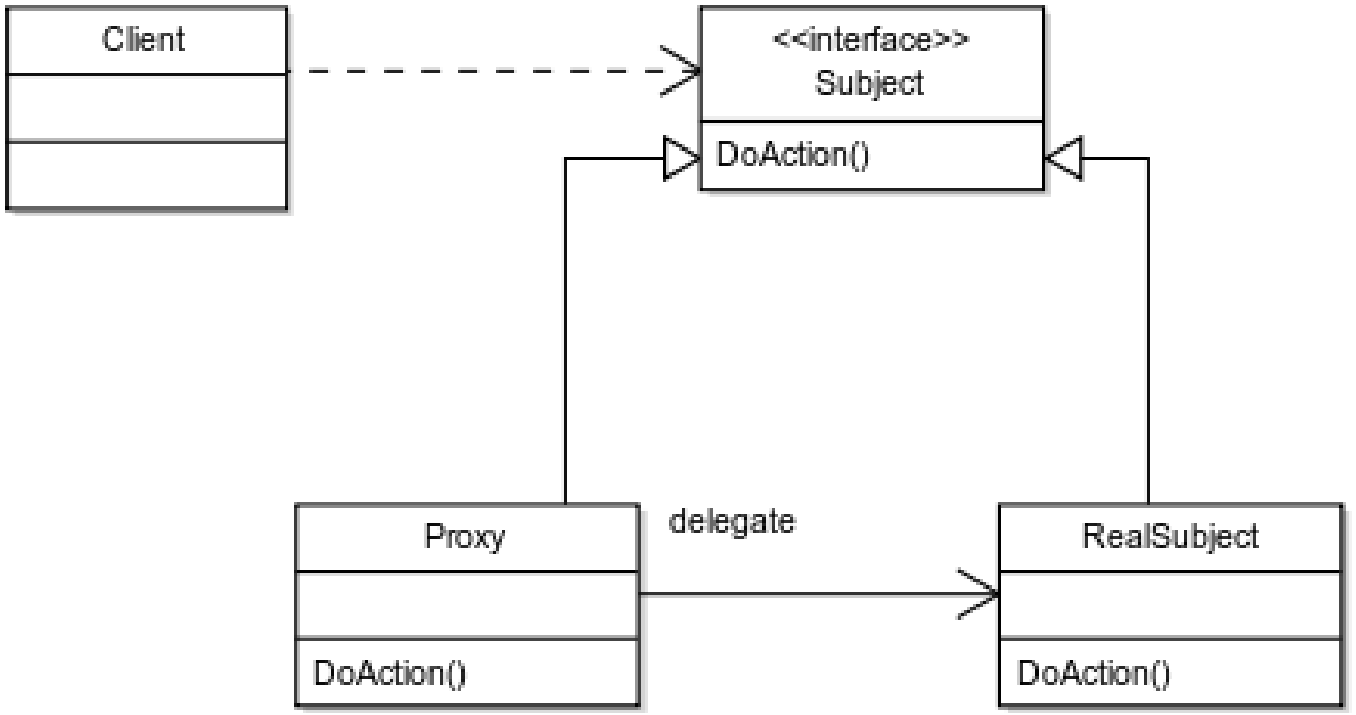
Design Patterns & AOP



Observer

- crosscutting concerns
  - observer management
  - notification

Design Patterns & AOP



Proxy

- Cross-cutting concern
  - DoAction()

Design Patterns & AOP

How is AOP different from

- proxy design pattern
- decorator design pattern