AOP & Fractal

L. Seinturier, N. Pessemier, L. Duchien, O. Barais
INRIA Futurs (Jacquard) & LIFL (GOAL)

OW architecture meeting - Paris
25-26 march 2004

Plan

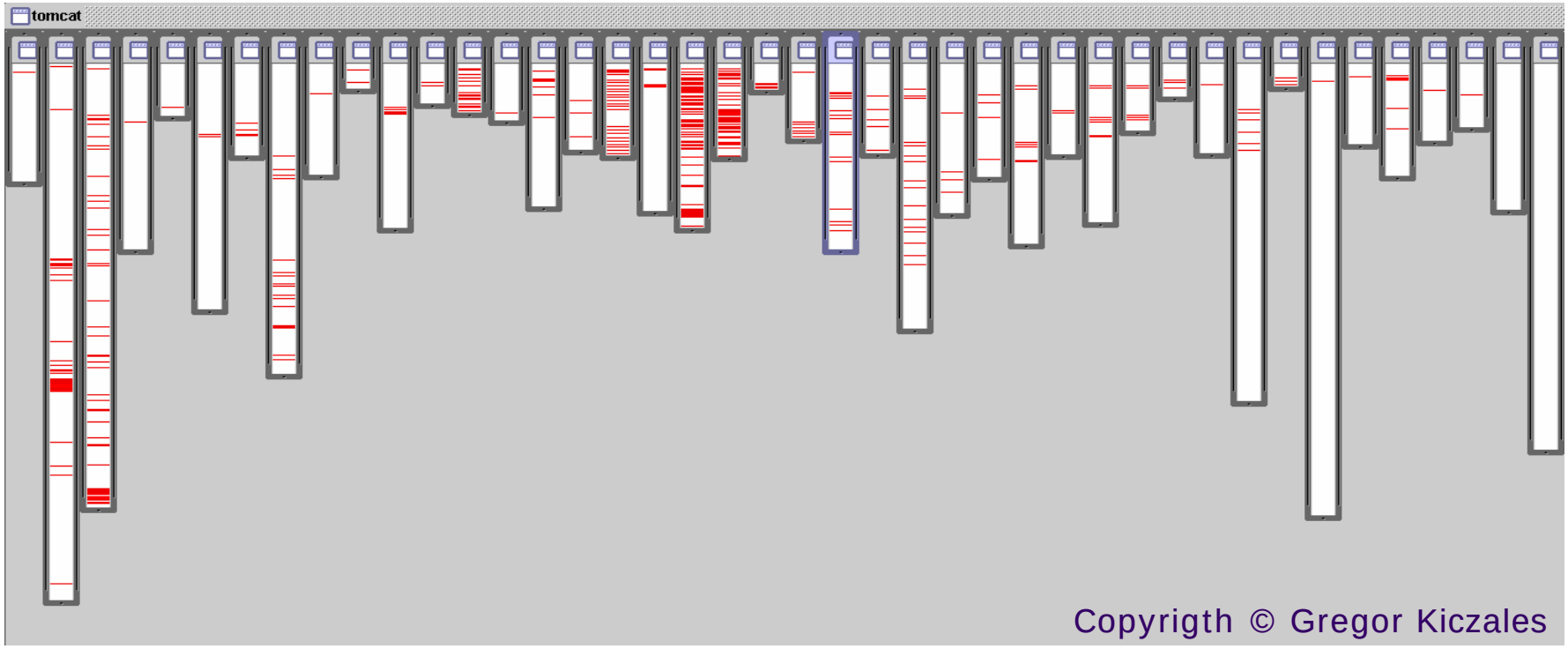
1. Why AOP?
2. AOP & components & ADLs
3. Our project for AOP & Fractal
4. Directions

Why AOP?
Founding case

Aspect Oriented Programming [Kiczales 97]

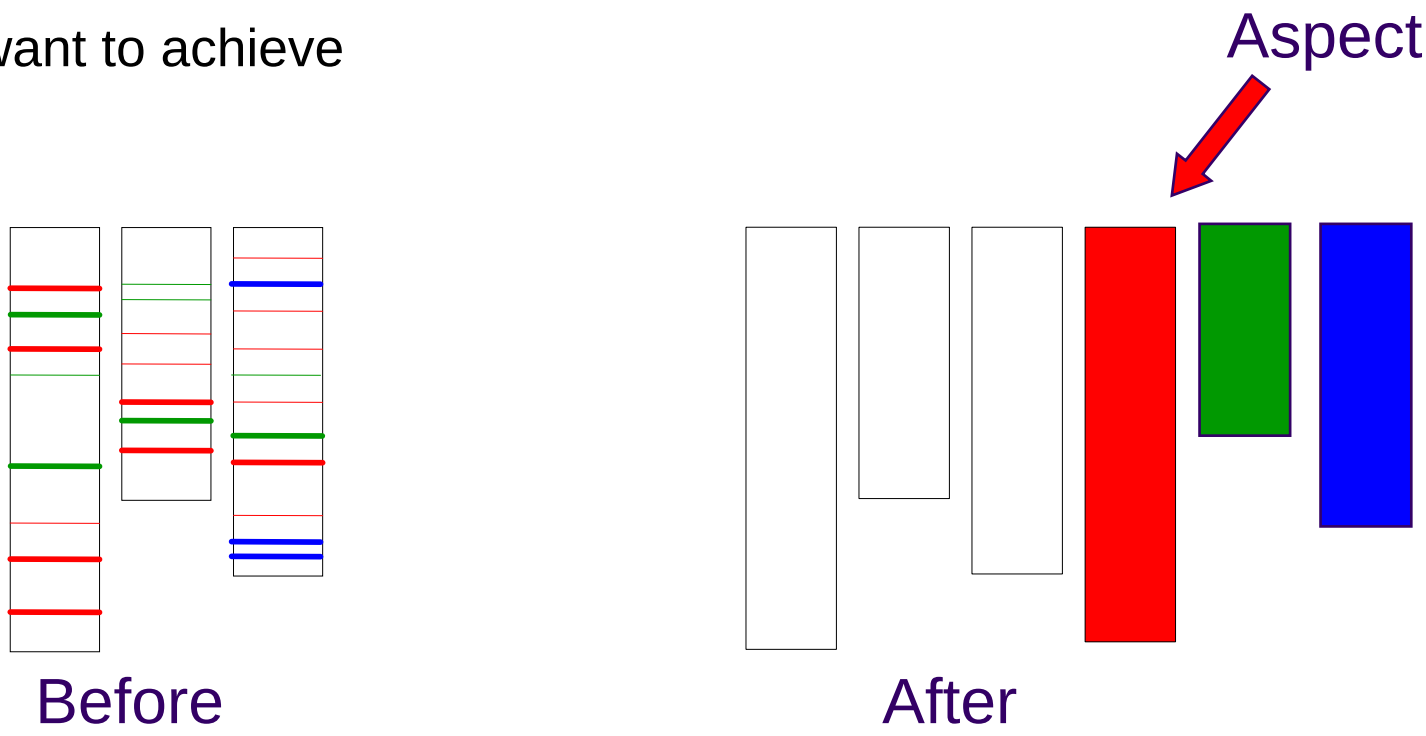
Where is logging in org.apache.tomcat

- not in just one place
- not even in a small number of places



Why AOP?

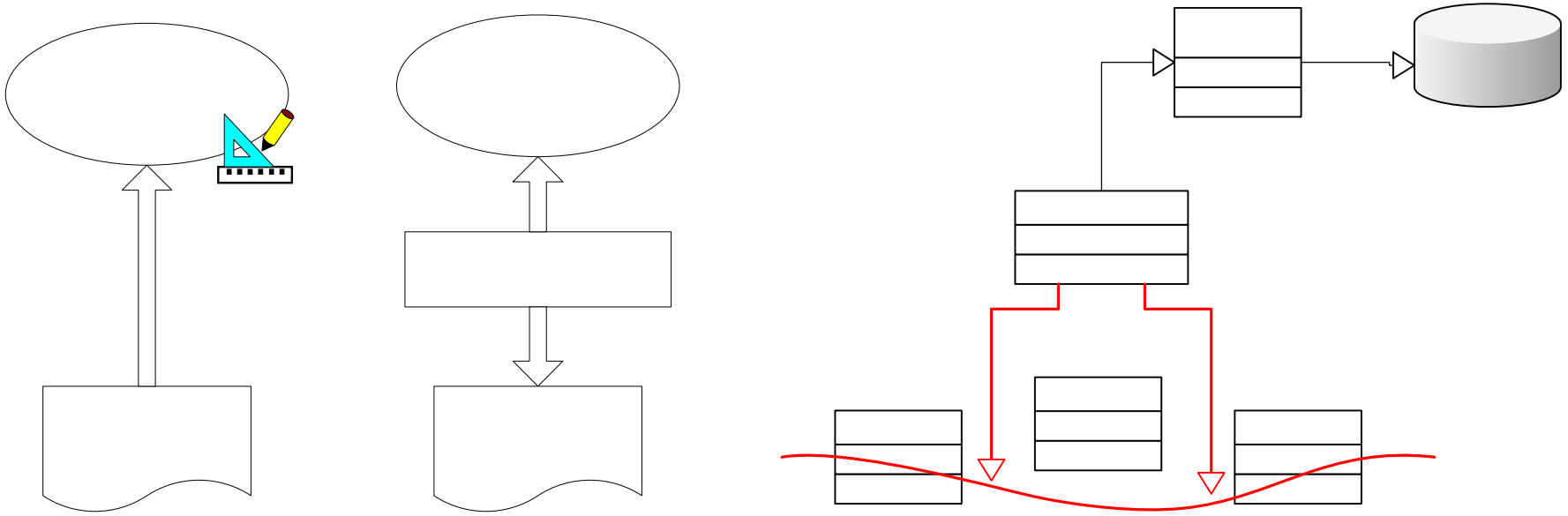
What we want to achieve



- Better modularity

Why AOP?

- Dependency inversion



Why AOP?

AOP **primary** goal

- **modularize crosscutting concerns**

AOP secondary goals

- address dynamicity & reconfiguration
- (try) to address the issue of software composition

AOP & components & ADLs

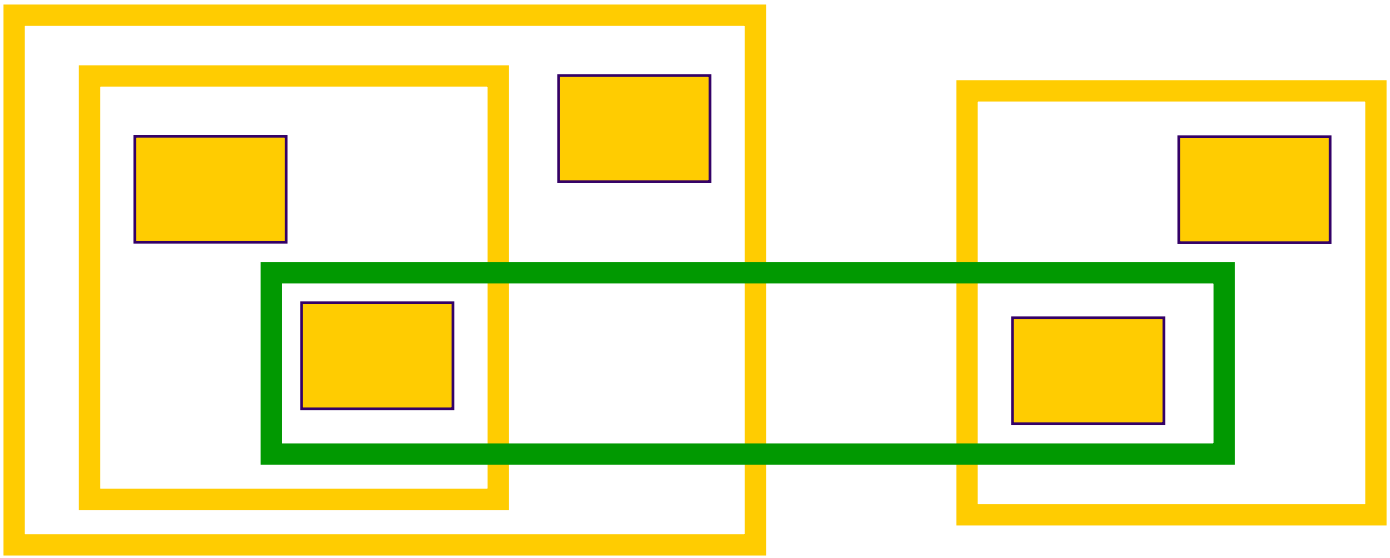
In terms of software engineering

- components: business logic
- aspects: ~~non functional~~ crosscutting logic
- ADLs: software maps
  - good support for business assemblies
  - poor support for crosscutting concerns
- Fractal controller
  - level of control on components
  - but crosscut policy ?
  - but aspect weaving (aka deployment) ?

AOP & Fractal

- **Our project** support for crosscutting concerns in Fractal

Abstract view



- Aspects cross the boundaries of membranes
- 1 domain of concern per aspect

AOP & Fractal

➢ Our project

Envisioned solutions

1. 1 controller per aspect
   - pros: efficiency (mix-in support by Julia)
   - cons: requires highly (over) skilled developpers
2. a generic aspect controller bound to aspect components
   - pros: closer to existing AOP framework (JBoss AOP, JAC, …)

➢ Short term goal: define & implement **2.**

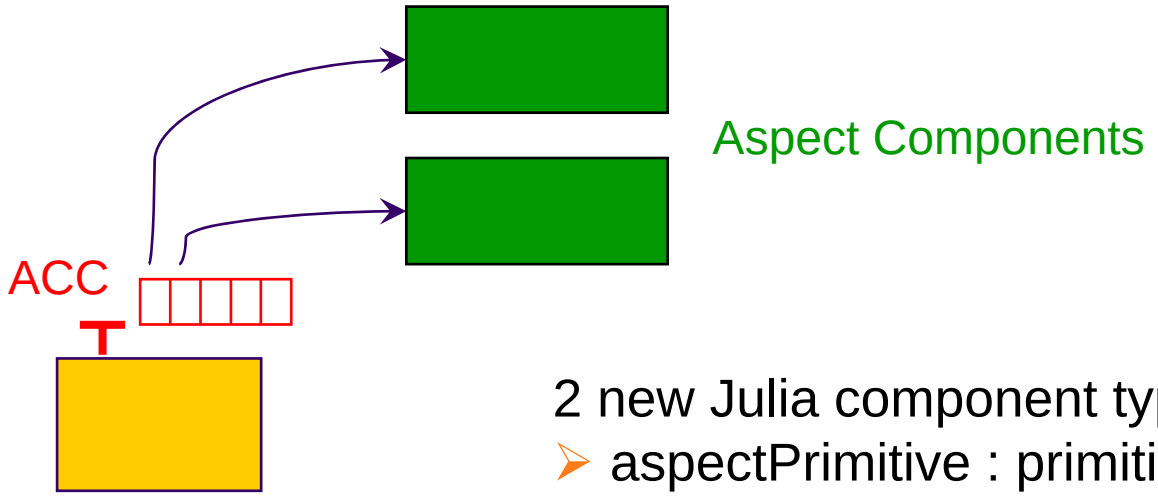
➢ Longer term goal: a common model for **1.** & **2.**

AOP & Fractal

- **A new controller**

Aspect Component Controller (ACC)
based on the interception controller

- based on the interception controller
- manages a list of references towards aspect components (AC)



2 new Julia component types
- aspectPrimitive : primitive + ACC
- compositePrimitive : compositite + ACC

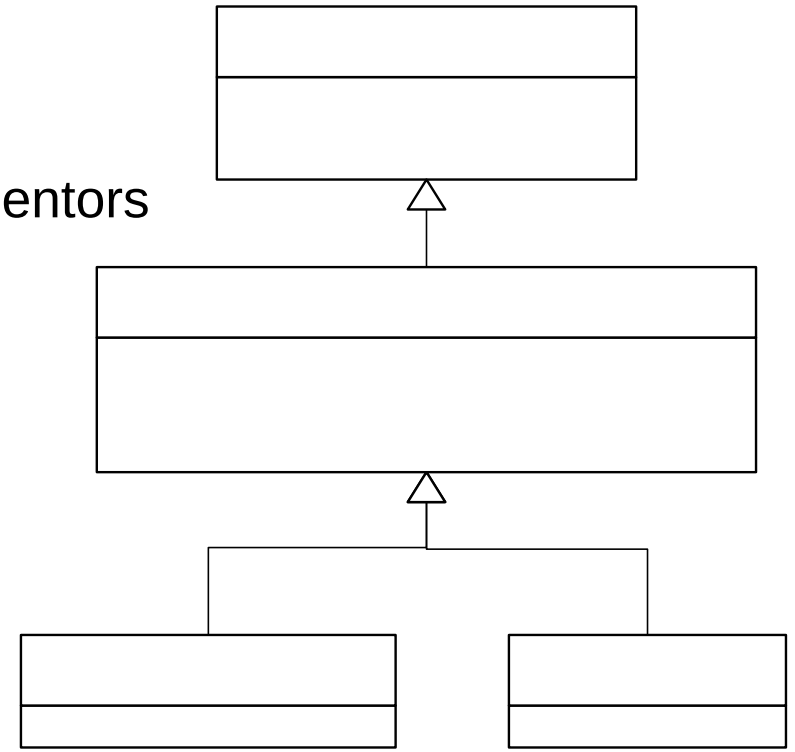
AOP & Fractal

➢ Aspect Component (AC) programming

AOP Alliance API

- open-source initiative
- emerge from aspect framework implementors (JAC, PROSE, Spring, Nanning, …)

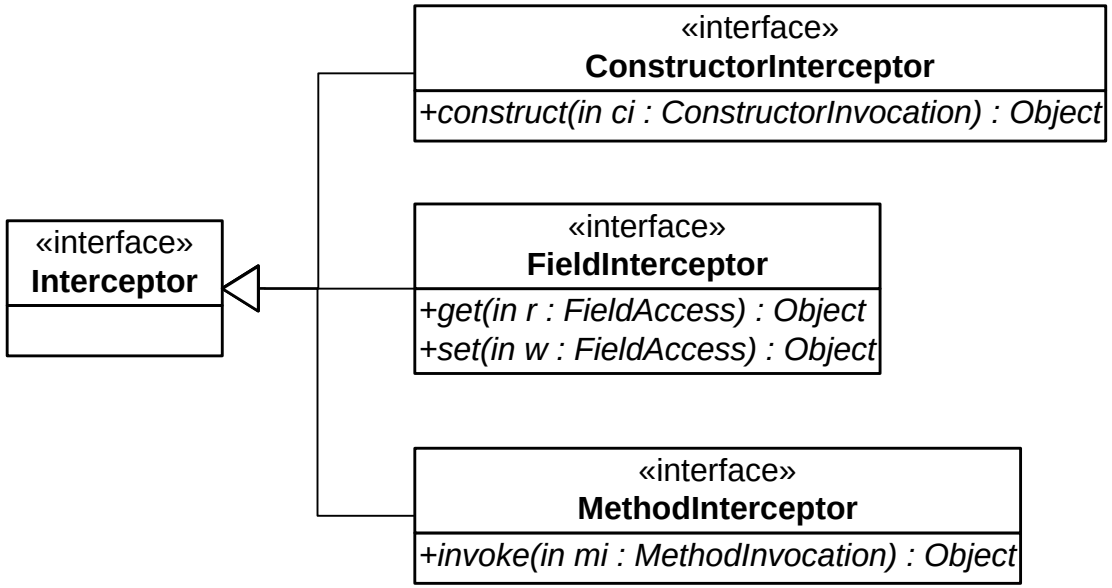
- standardize the API of aspect weavers for core concepts (join points, interceptors, advice, pointcut, aspect, introduction)



AOP & Fractal

➢ Aspect Component (AC) programming

AOP Alliance API

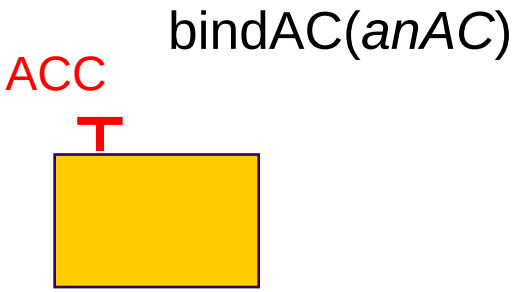


AC are MethodInterceptors for Fractal interfaces

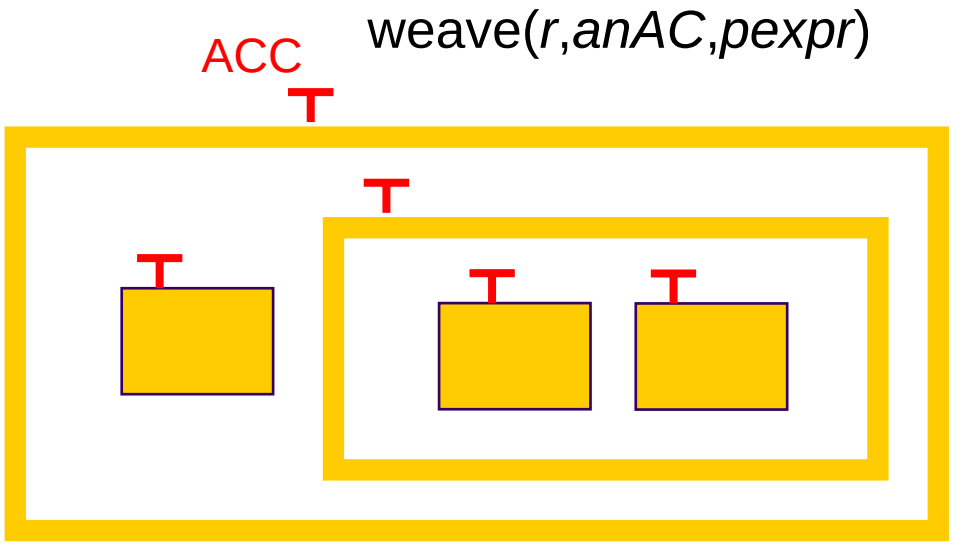
AOP & Fractal

➢ 2 ways for binding AC & components

➢ direct



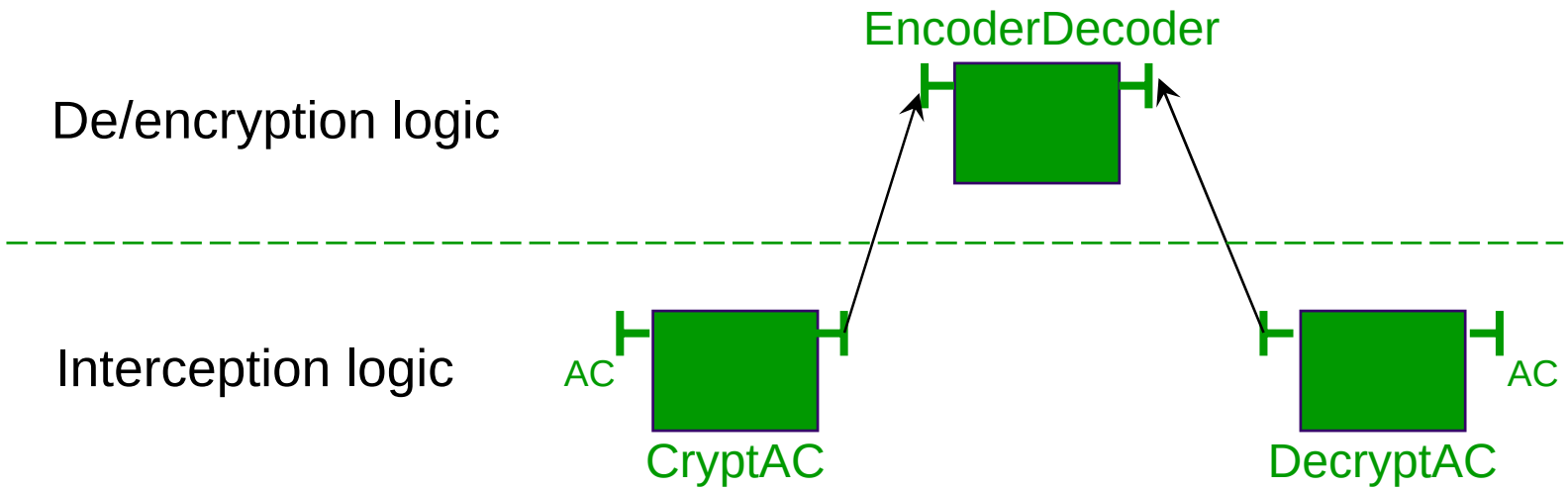
➢ weaving
- recursive traversal
- binding with components that match a pointcut expression



AOP & Fractal Example

- **A gas station**

+ an encryption aspect



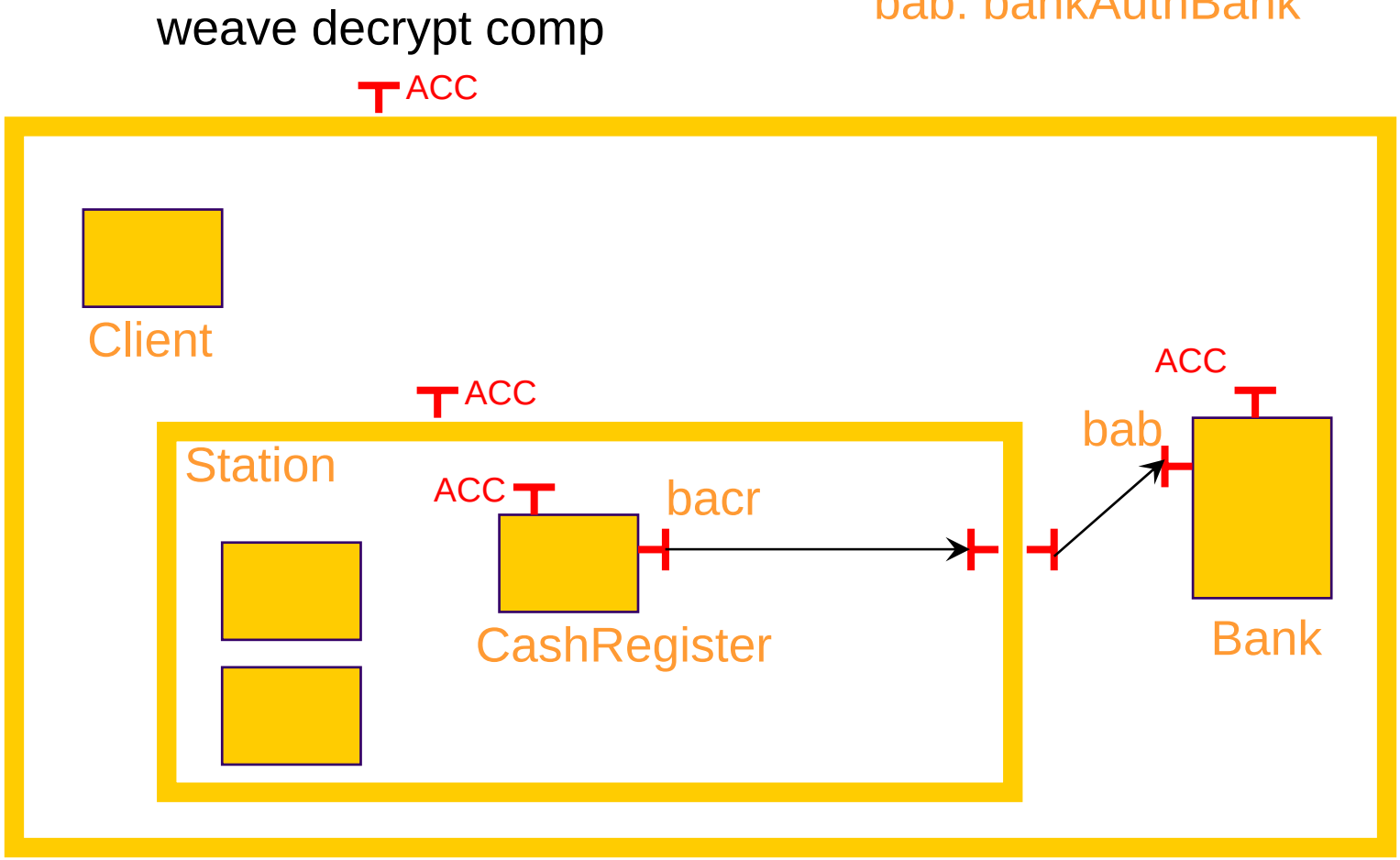
AC implements MethodInterceptor

AOP & Fractal Example

```
public class CryptAC implements AspectComponent, BindingController {

   private Encoder encoder;

   public Object invoke(MethodInvocation m) throws Throwable {
        String crypt = encoder.encrypt((String) m.getArgument(0));
        m.setArgument(0, crypt);
        System.out.println(">> CryptAC : " + crypt);
        return m.proceed();
   }

    // binding controller interface implementation
}
```

AOP & Fractal Example

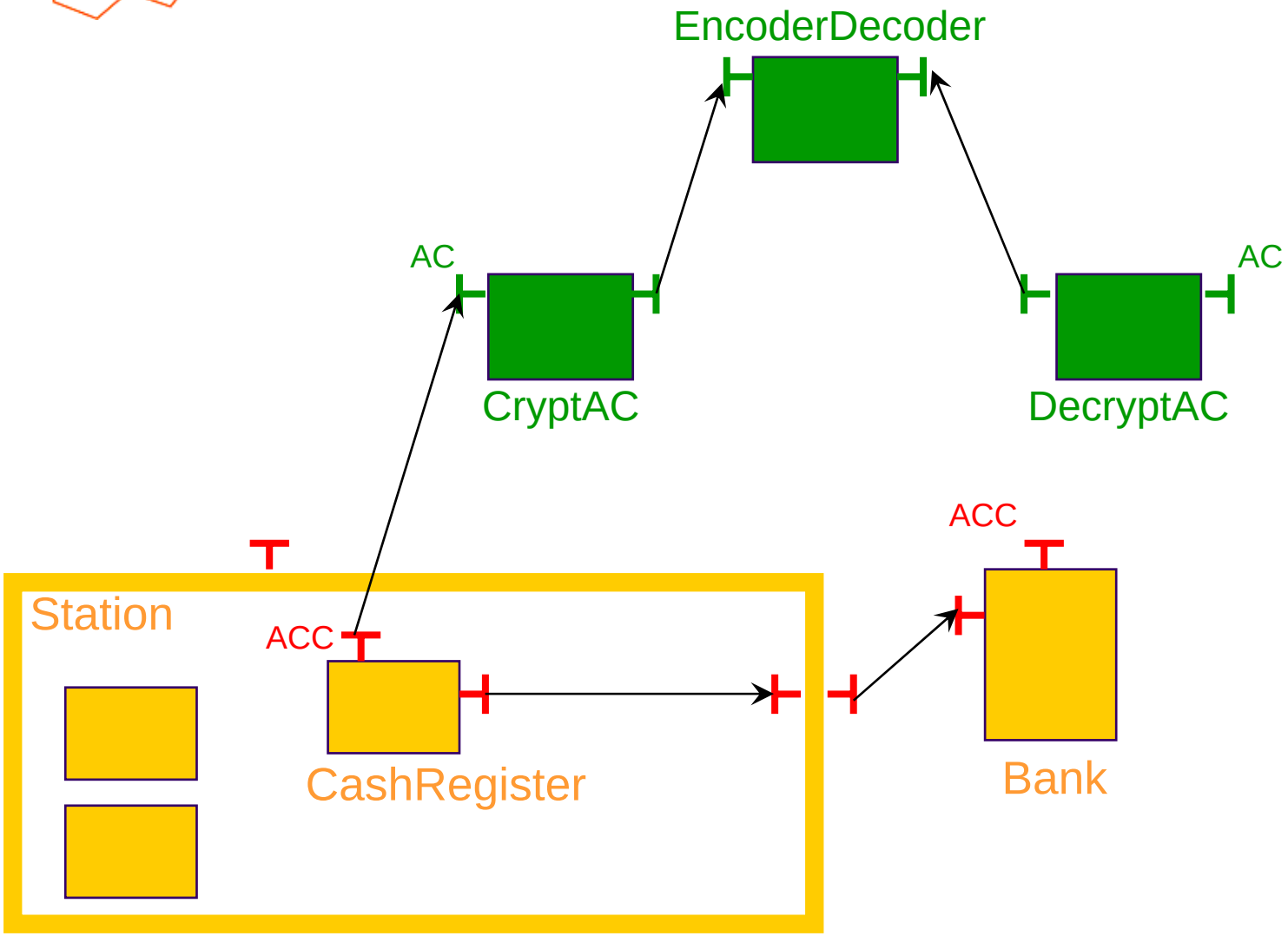
bacr: bankAuthCashRegister
bab: bankAuthBank



AOP & Fractal Example

```
AspectController acc;
AspectComponent ac;

acc = (AspectController)rootComp.getFcInterface("aspect-controller");
ac = (AspectComponent)cryptComp.getFcInterface("aspectComponent");
acc.weave(
  rootComp, ac,
  new AdvancedPointcutExp(    // Java regular expression
    ".*",                     // component name ( .* = ALL )
    "bankAuthCashRegister",   // interface name
    "askAuth.*")              // method name
);
```

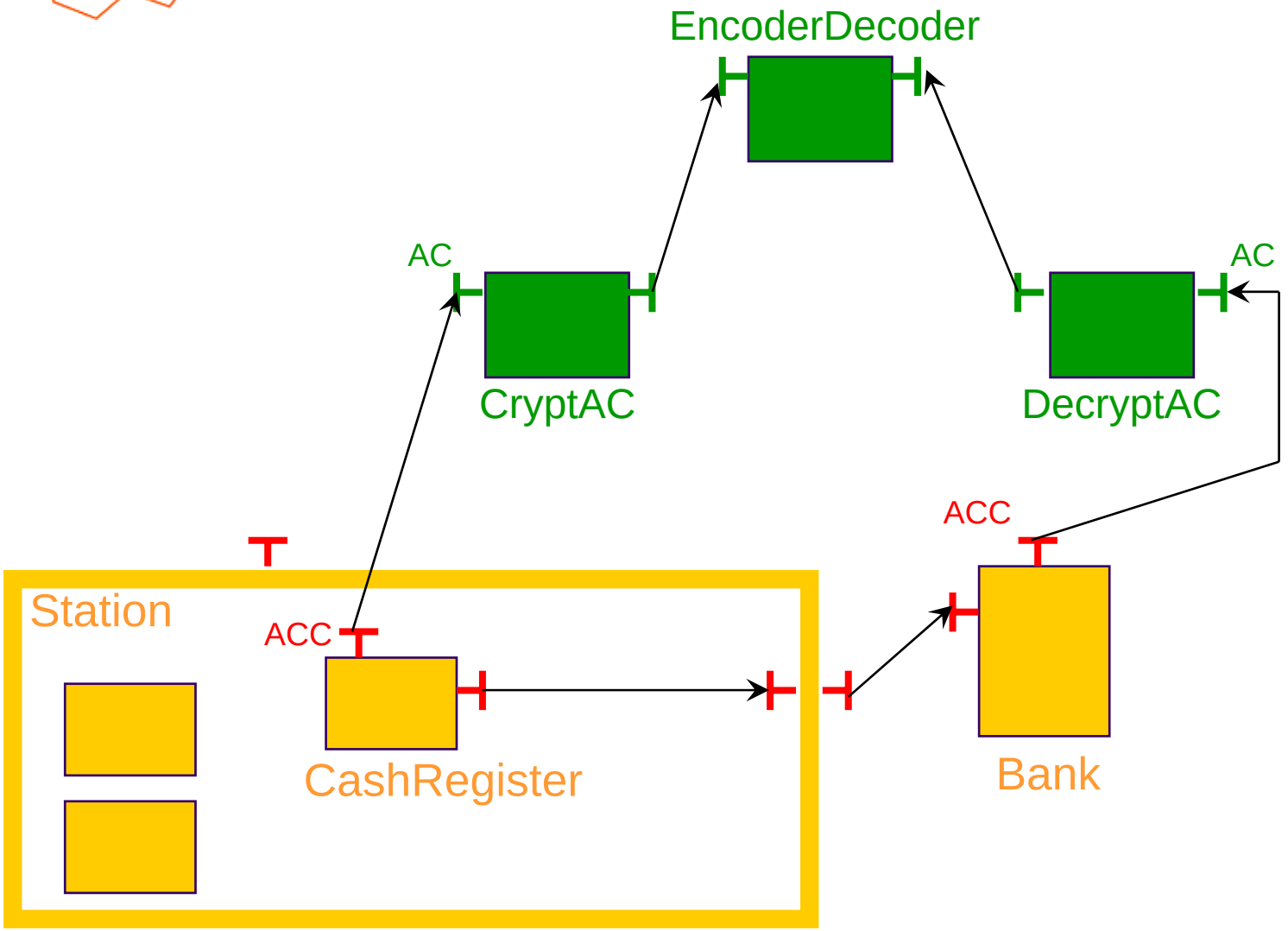
AOP & Fractal


EncoderDecoder
AC
AC
CryptAC
DecryptAC
ACC
Station
ACC
CashRegister
Bank

AOP & Fractal

```
AspectController acc;
AspectComponent ac;

acc = (AspectController)rootComp.getFcInterface("aspect-controller");
ac = (AspectComponent)decryptComp.getFcInterface("aspectComponent");
acc.weave(
  rootComp, ac,
  new AdvancedPointcutExp(    // Java regular expression
    ".*",                     // component name ( .* = ALL )
    "bankAuthBank",           // interface name
    "askAuth.*")              // method name
);
```

AOP & Fractal


EncoderDecoder
AC
AC
CryptAC
DecryptAC
ACC
Station
ACC
CashRegister
Bank

Conclusion

- Aspect Component controller (ACC)
  - intercept method calls on Fractal interfaces
  - manage a list of references towards aspect components
- Aspect Component (AC)
  - implement some method interception logic
- pointcut expressions
  - declaratively specify where AC must be bound
  - 3 regular expressions for
    - component names
    - interface names
    - method names
- weaving: traversal of the composite to find matching interfaces