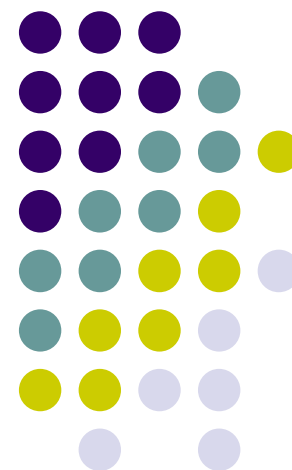
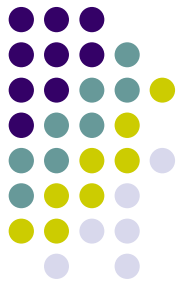


# AOP in the Academia

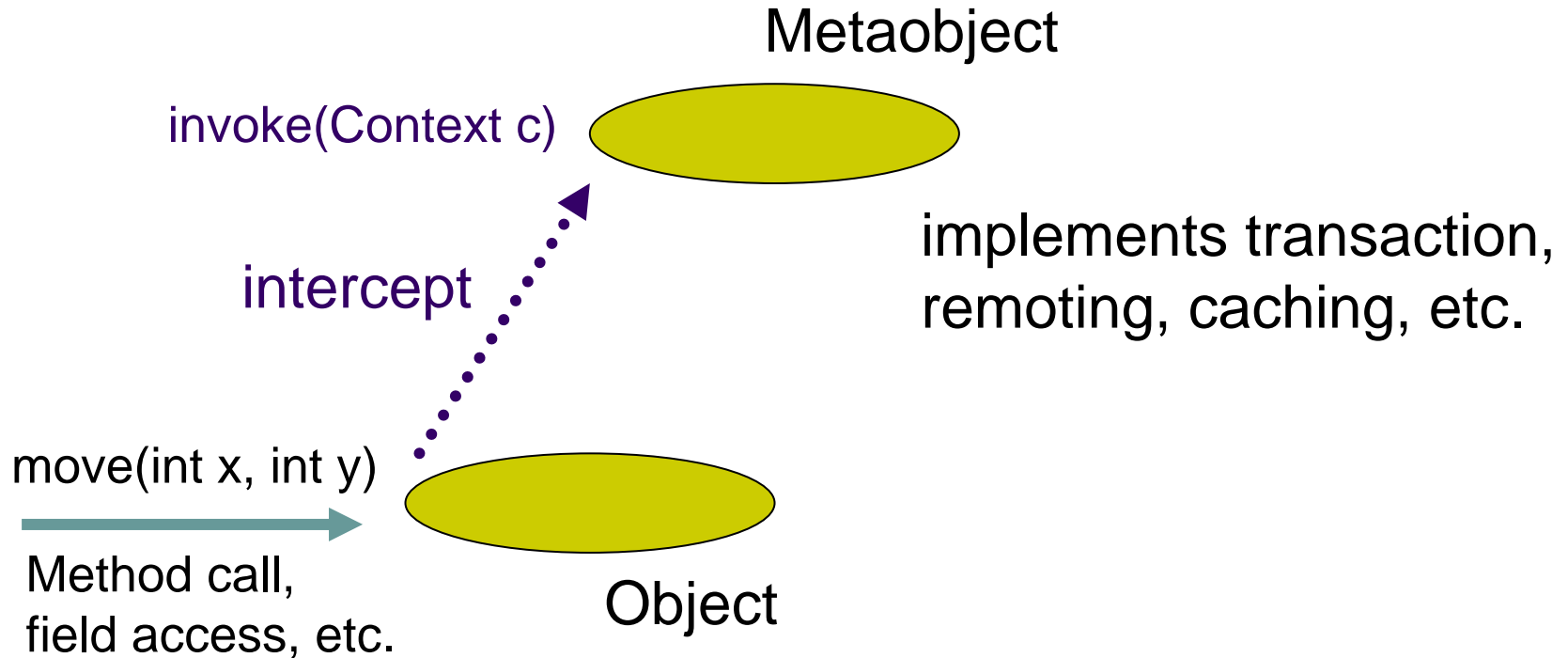
Shigeru Chiba  
Tokyo Institute of Technology  
Japan





# Metaobject (or Reflection)

- From late 80's to 90's



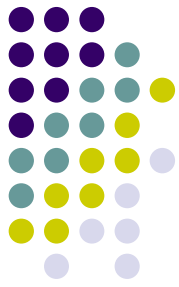
# Your Program

- Just one annotation

```
/**
 * @metaclass TransactionClass
 */
public class ShoppingCart {
    public void addItem(Item i) { ... }
    :
}
```

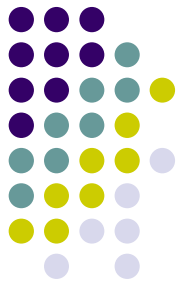


This is not a real example.



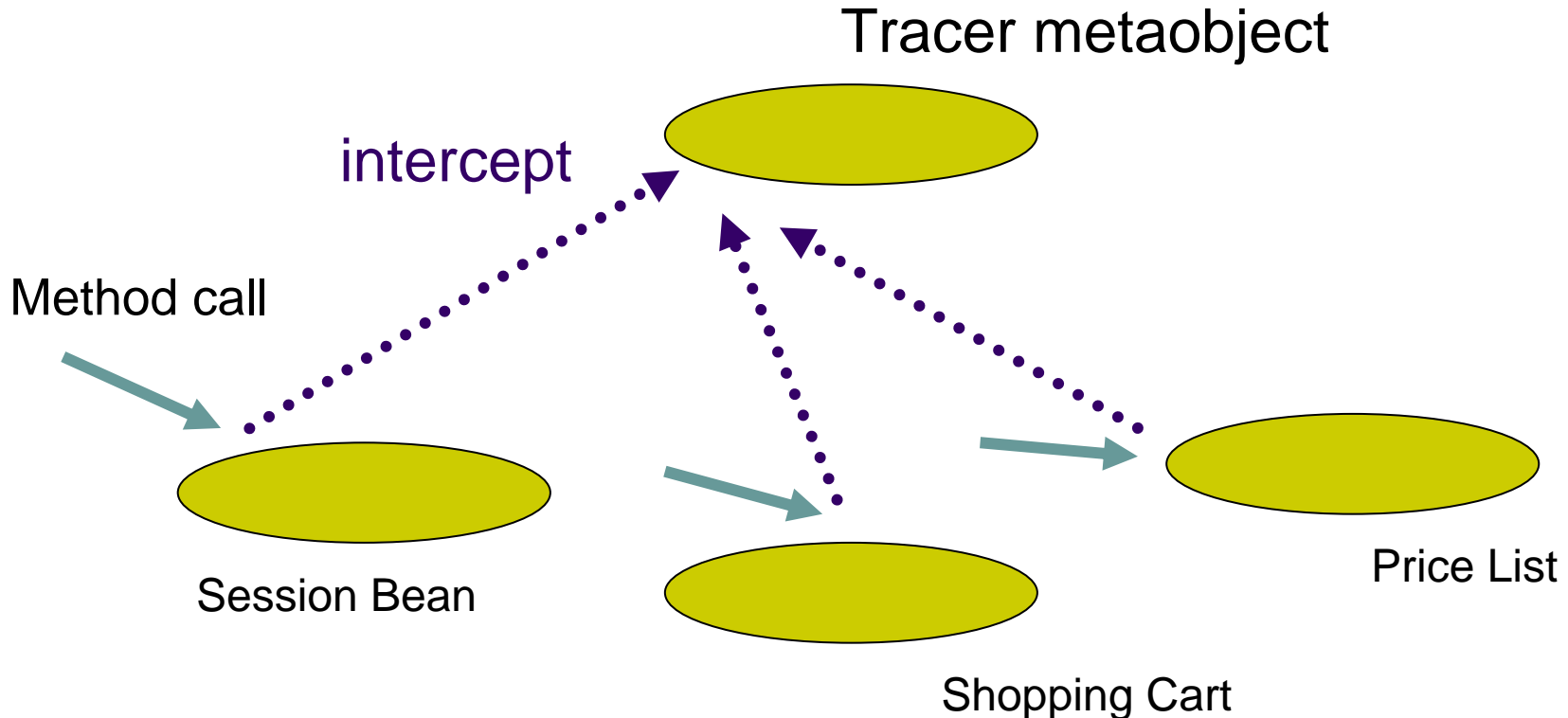
# Why do we need AOP?

- In 2003
  - Crosscutting Concerns
  - Non invasiveness  
(a.k.a. obliviousness)



# Crosscutting Concerns

- Trace a shopping session.





# Bad Modularity

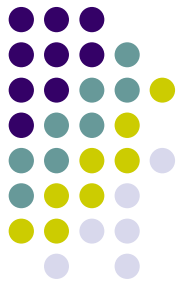
- Your program is ...

```
/**
 * @metaclass Tracer
 */
class MySession {
    :
}
```

```
/**
 * @metaclass Tracer
 * @on getPrice(int id)
 */
class PriceList {
    :
}
```

```
/**
 * @metaclass Tracer
 * @on addItem(Item i)
 * @on save$10()
 */
class ShoppingCart {
    :
```

```
}
```



# For better modularity by AOP

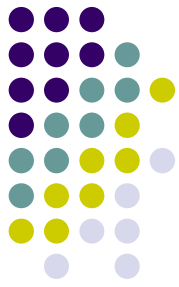
- Separate the concern and put it into a single module.

```
<pointcut advice="Tracer">  
  <class name="PriceList">  
    <on method="getPrice"/>  
  </class>  
  :  
</pointcut>
```

```
class MySession {  
  :  
}
```

```
class PriceList {  
  :  
}
```

```
class ShoppingCart {  
  :  
}
```



# Non-Invasiveness

- a.k.a. Obliviousness
  - Don't mix the crosscutting concern with a class.

This is part of another module.  
Editing it may cause an  
unexpected problem.

```
/**  
 * @metaclass Tracer  
 * @on getPrice(int id)  
 */  
class PriceList {  
    :  
}
```



# Tags are still useful

- Tags can represent attributes of the class.
  - Not attributes of the crosscutting concern.

```
<pointcut advice="Tracer">
  <tag name="Shopping"/>
</pointcut>
```

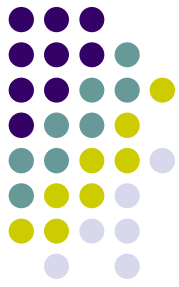
```
/**
 * @tag Shopping
 */
class PriceList {
    :
}
```

```
/**
 * @tag Shopping
 *     addItem(Item i)
 * @tag Shopping
 *     save$10()
 */
class ShoppingCart {
    :
}
```



# FAQ: debugging

- Does AOP make debugging difficult?
  - Or does non-invasiveness make ... ?
- Answer:
  - Tool support (e.g. Eclipse AspectJ plugin)
  - Does polymorphism in OOP make debugging difficult?



# Summary

- Metaobject (or Reflection)
  - Interception
  
- Aspect-Oriented Programming (AOP)
  - Crosscutting Concerns
  - Non-invasiveness