COMP3027J 课程
软件架构

可修改性及其策略

邓永健

北京工业大学计算机学院

数据挖掘与安全实验室（DMS 实验室）

大纲

1. 可修改性

2. 提升可修改性的策略

大纲

1. 可修改性的含义

2. 提升可修改性的策略

可修改性的含义

担忧

- 修改费用

- 系统的哪些部分被修改了

- 当修改发生时

谁进行的修改?

可修改性的含义

测量指标

完成修改所需时间

- 修改的人力资源成本
- 改造的经济成本
- ……

可修改性的意义——场景示例

刺激源

- 谁执行修改（开发人员/管理员/用户）

刺激

- 需要做出的具体修改

可修改性的意义——场景示例

文物

- 要修改系统的功能、用户界面或其他交互系统吗?

环境

这种修改在什么时候进行? 是在设计阶段、开发阶段还是运行阶段?

修改得越晚，就越不利。

可修改性的意义——场景示例

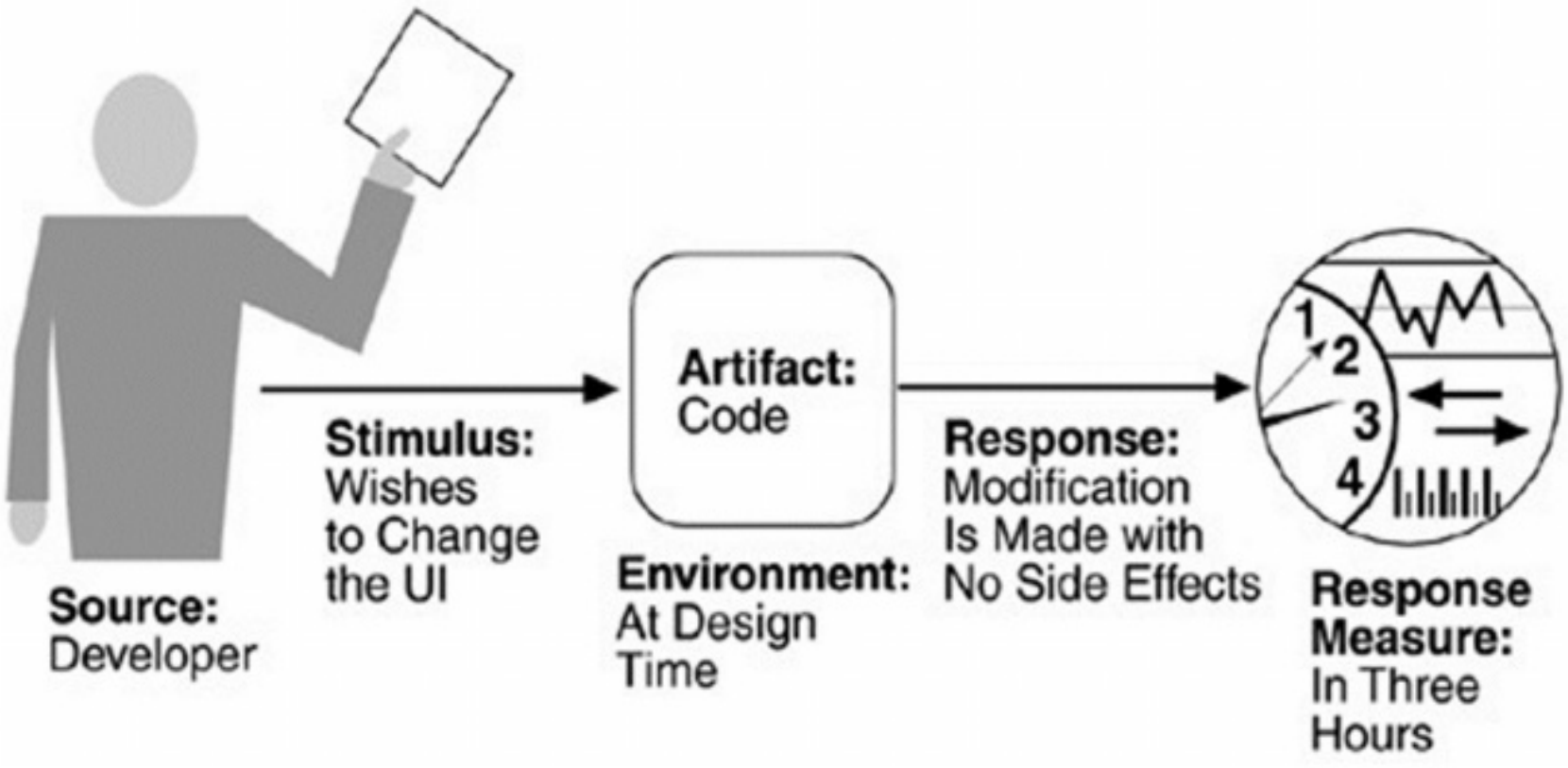
反应

操作人员必须了解如何修改、执行修改操作、测试以及部署。

应对措施

- 时间与成本

可修改性 场景 示例


Source:
Developer
Stimulus:
Wishes
to Change
the UI
Artifact:
Code
Environment:
At Design
Time
Response:
Modification
Is Made with
No Side Effects
1
2
3
4
Response
Measure:
In Three
Hours

大纲

1. 可修改性

2. 提升可修改性的策略

提高可修改性的策略

-概述

目标

减少修改的时间和成本

方向 1：限制修改范围

尽量将修改所影响的软件范围控制得尽可能小。

方向 2：延迟绑定时间

允许软件在运行过程中灵活修改

提高可修改性的策略

-限制范围

模块内高内聚，模块间低耦合

尽量将程序的修改控制在单个模块内。

- 能够利用框架、中间件

考虑潜在的修改

有助于评估模块之间的职责划分

确保一处的修改仅影响一个模块

- 避免对同一模块进行不相关的多次修改

提高可修改性的策略

-限制范围

使模块通用化

“口译员”式方法

提高可修改性的策略

-限制范围

隐藏信息

面向对象机制中的可访问性（公有/私有）

保持一致的界面

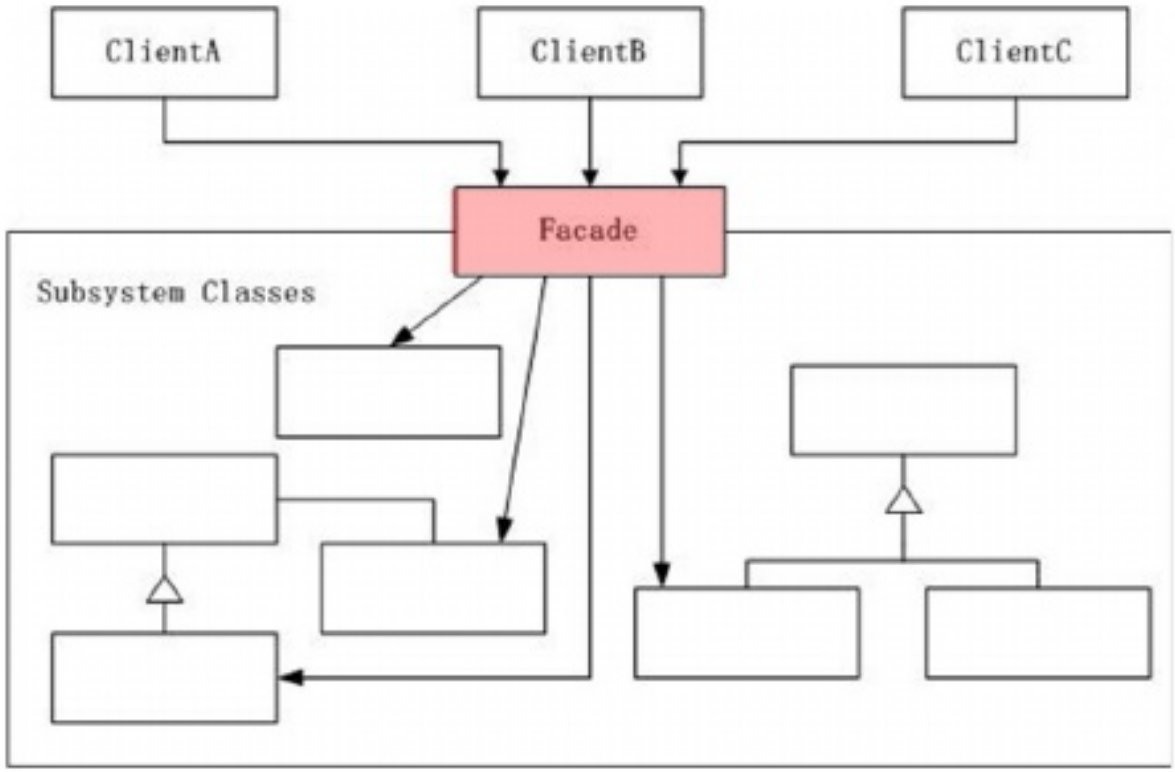
允许在接口的两侧进行独立更改，而不更改接口本身。

提高可修改性的策略

-限制范围

限制通信路径

设计模式中的外观模式



提高可修改性的策略

-限制范围

利用中间人

- 数据中介机构：数据共享的模式
- 服务中介：桥接、工厂方法等设计模式

提高可修改性的策略

-限制范围

名称服务器

查询所需资源/对象的位置，解决位置依赖关系

按需创建实例

在设计模式中利用创建型模式

提高可修改性的策略

-延迟绑定时间

配置文件

无需修改代码即可修改配置文件

```
<?xml version="1.0" encoding="utf-8"?>
<!-- The configuration file for SMSvcHost.exe -->
<configuration>
  <runtime>
    <gcConcurrent enabled="false" />
  </runtime>
  <system.serviceModel>
    <!-- SMSvcHost ETW traces are redirected by
default to an etwProviderId different from WCF's default.
-->
    <diagnostics performanceCounters="Off"
etwProviderId="{f18839f5-27ff-4e66-bd2d-
639b768cf18b}"/>
  </system.serviceModel>
</configuration>
```

```
[Global]
MessageTitle=智能自制内容工具
Button[0]=确定(&O)
Button[1]=取消(&C)
Button[2]=关闭(&C)
[Project]
Title=新建项目
Button[0]=名称:
Button[1]=位置:
Button[2]=浏览(&B)...
Text[0]=当前目录已经存在，是否覆盖此目录?
[PutCode]
Title=请输入相应数字
Button[0]=编号:
Button[1]=如果存在相同编号直接覆盖
```

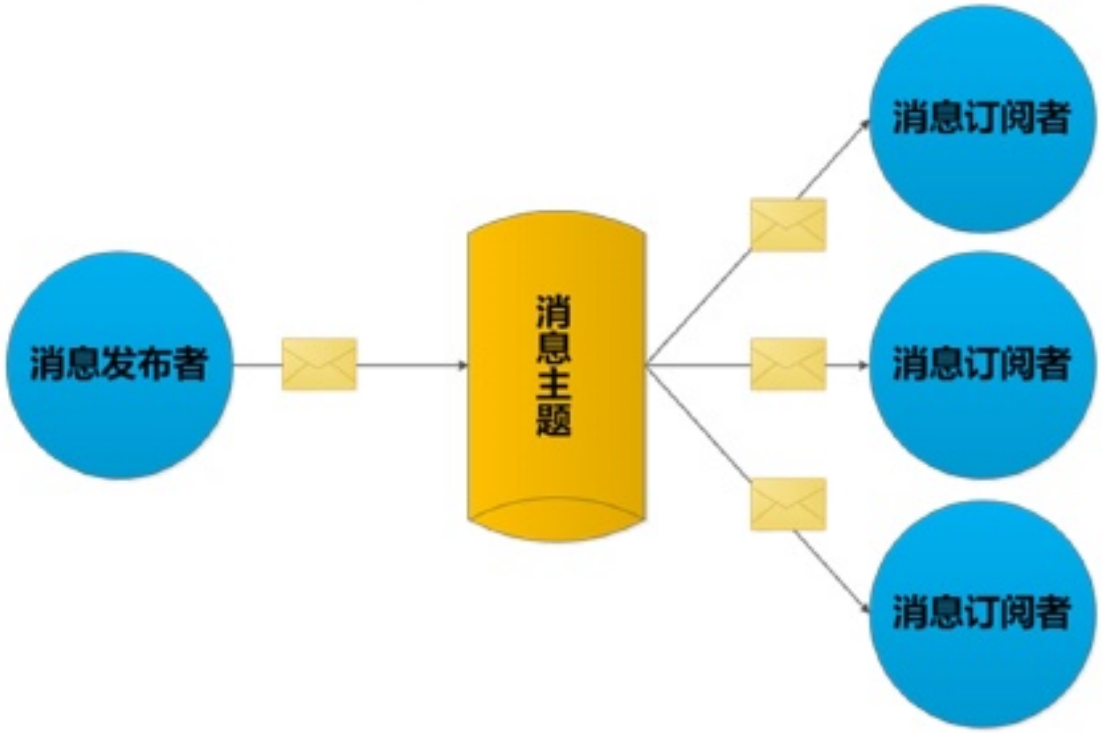
提高可修改性的策略

-延迟绑定时间

发布-订阅模式

- 在软件架构风格部分（基于事件的系统）中已作介绍

- 观察者模式



提高可修改性的策略

-延迟绑定时间

发布-订阅模式

- 微博示例

提高可修改性的策略

-延迟绑定时间

多态性

使用不同的子类来实现不同的功能

```
class Animal {
  public void eat() {System.out.println("进食");}
}
class Dog extends Animal {
  public void eat(){System.out.println("狗吃肉");}
}
class Cat extends Animal {
  public void eat(){System.out.println("猫吃鱼");}
}
```

```
public class Test {
  public static void main(String[] args){
    //定义为狗
    Animal a = new Dog();
    a.eat();
    //变成猫
    a = new Cat();
    a.eat();
```

可修改性 - 概要

可修改性方面的担忧

- 修改费用

提高可修改性的策略

- 限制修改范围

延迟绑定时间

谢谢你!