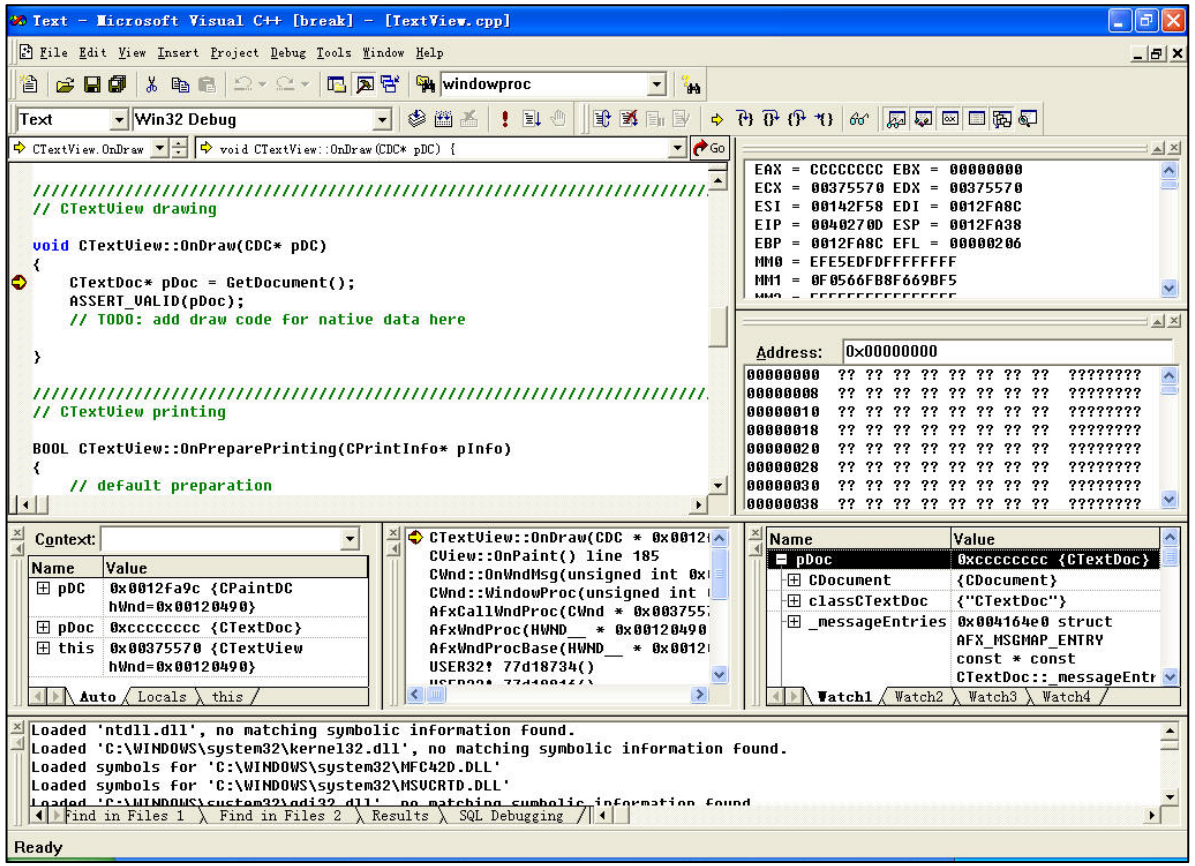
软件体系结构风格（三）

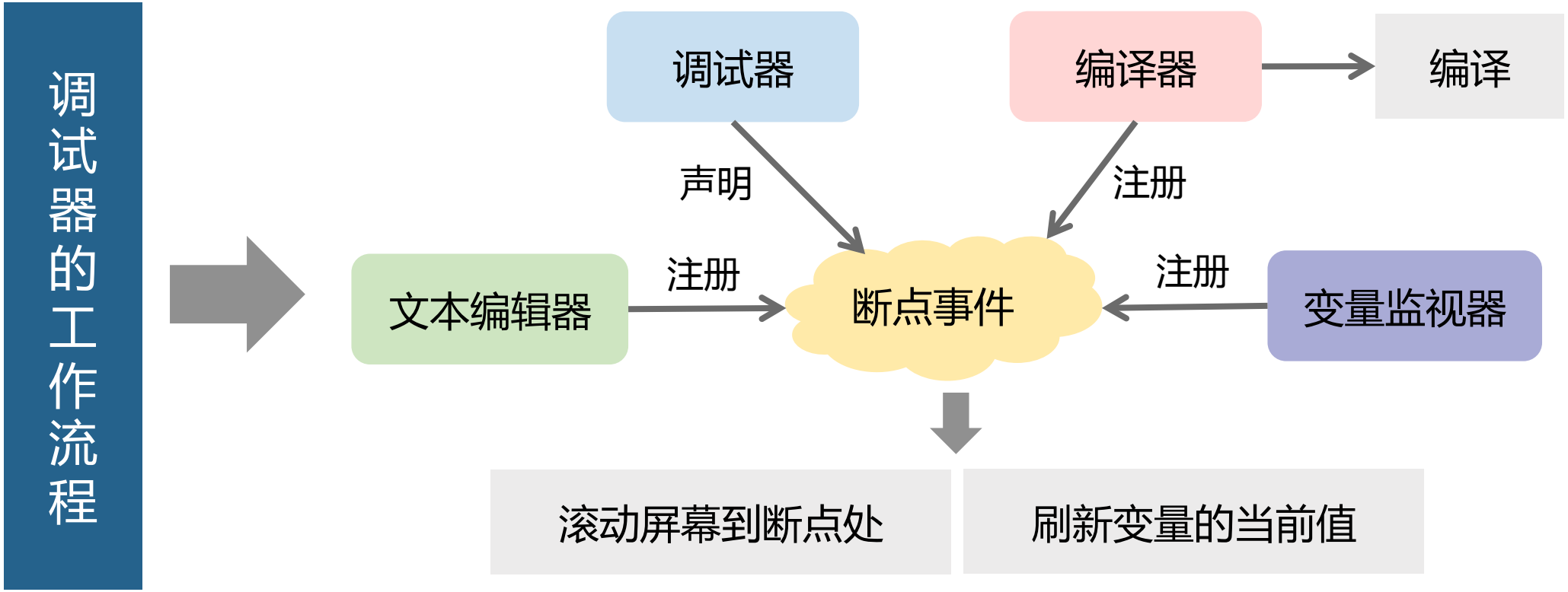
清华大学软件学院 刘强

事件风格

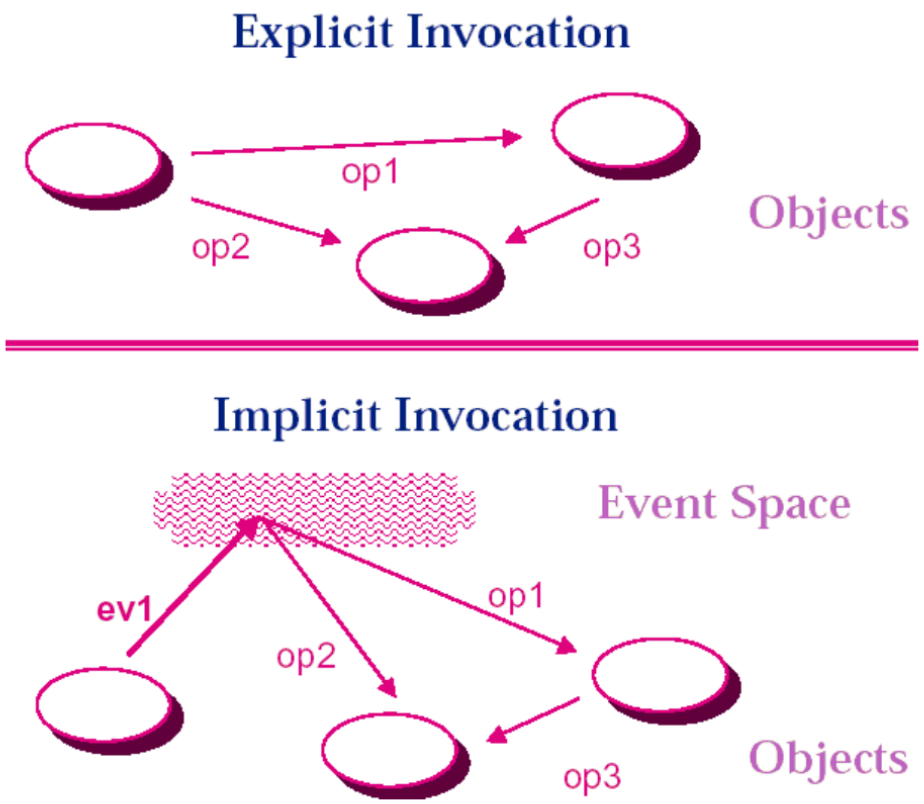


程序调试器的体系结构

事件风格


调试器的工作流程
调试器
编译器
编译
声明
注册
文本编辑器
注册
断点事件
注册
变量监视器
滚动屏幕到断点处
刷新变量的当前值

事件风格



显式调用：

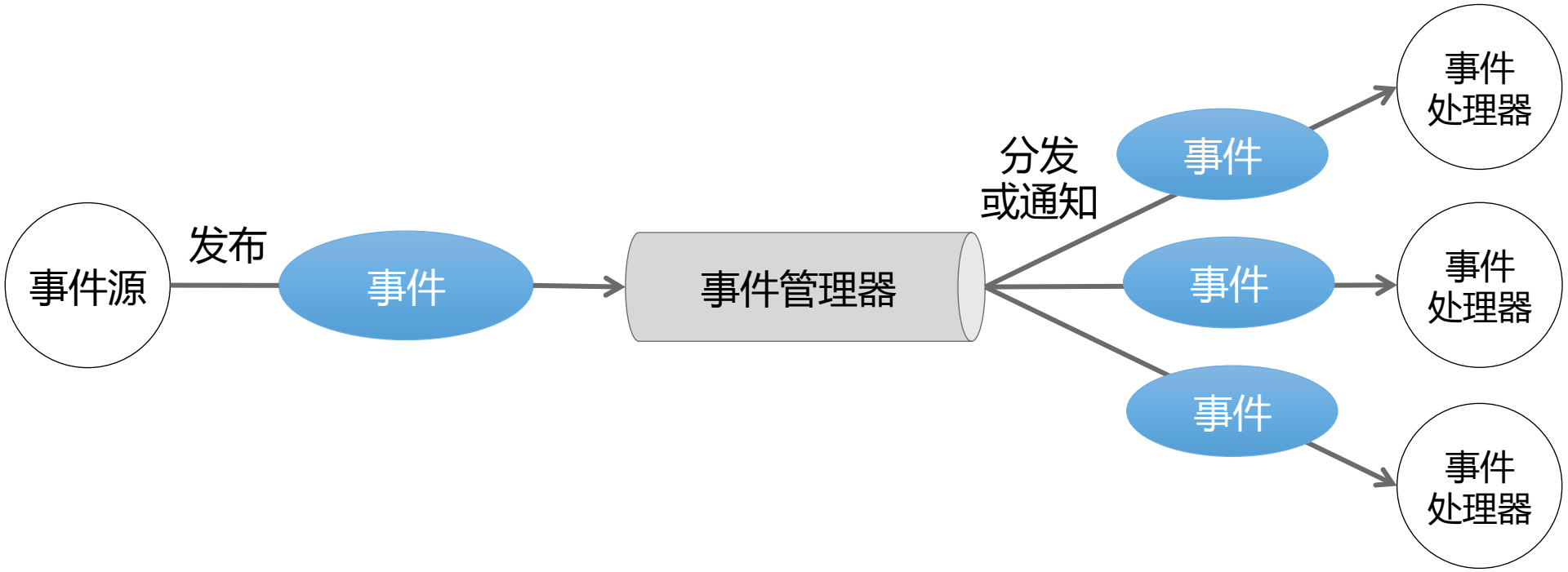
- 各个构件之间的互动是由显性调用函数或程序完成的
- 调用过程与次序是固定的、预先设定的

隐式调用：

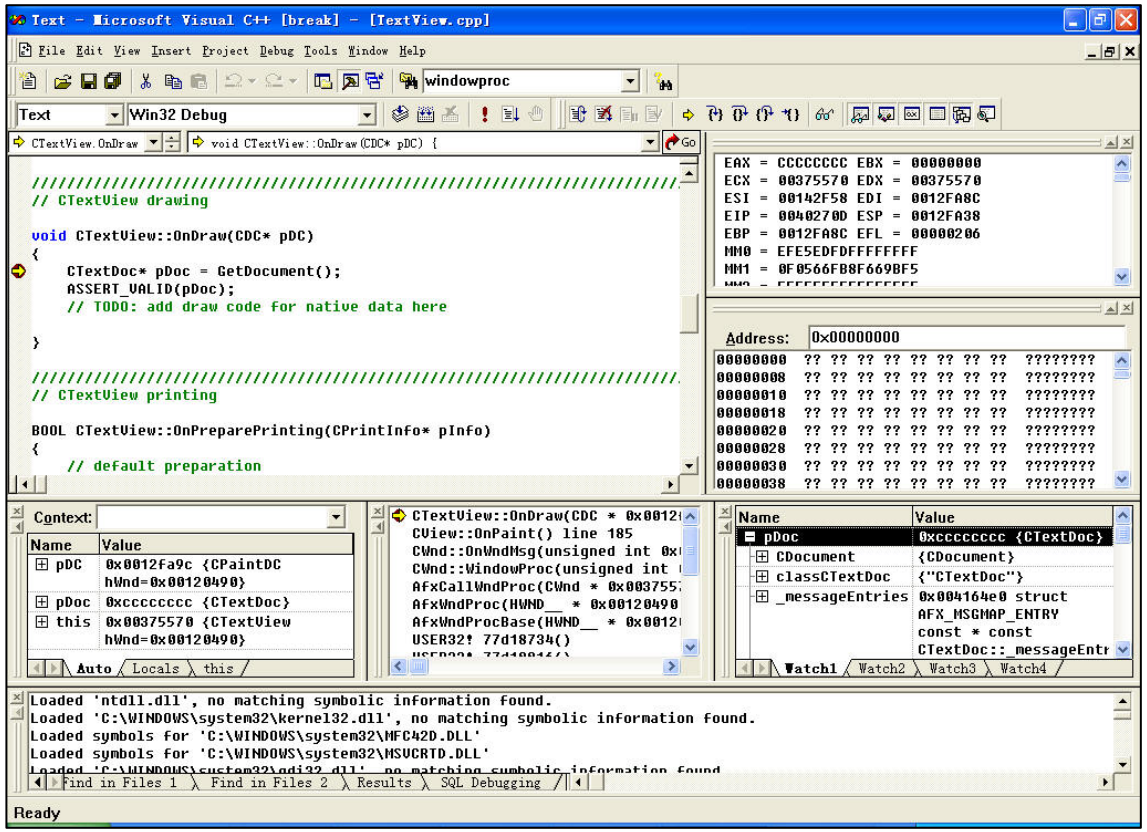
- 调用过程与次序不是固定的、预先未知
- 各构件之间通过事件的方式进行交互

事件风格

事件系统是将应用看成是一个构件集合，每个构件直至发生对它有影响的事件时才有所动作。



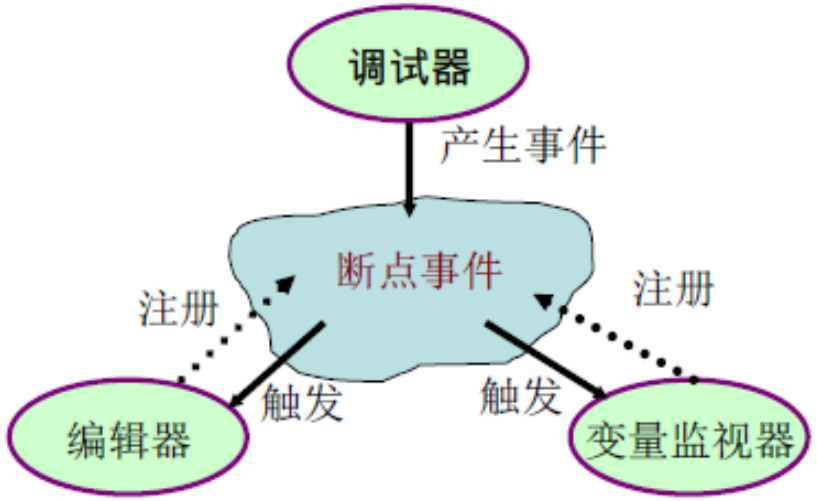
事件风格



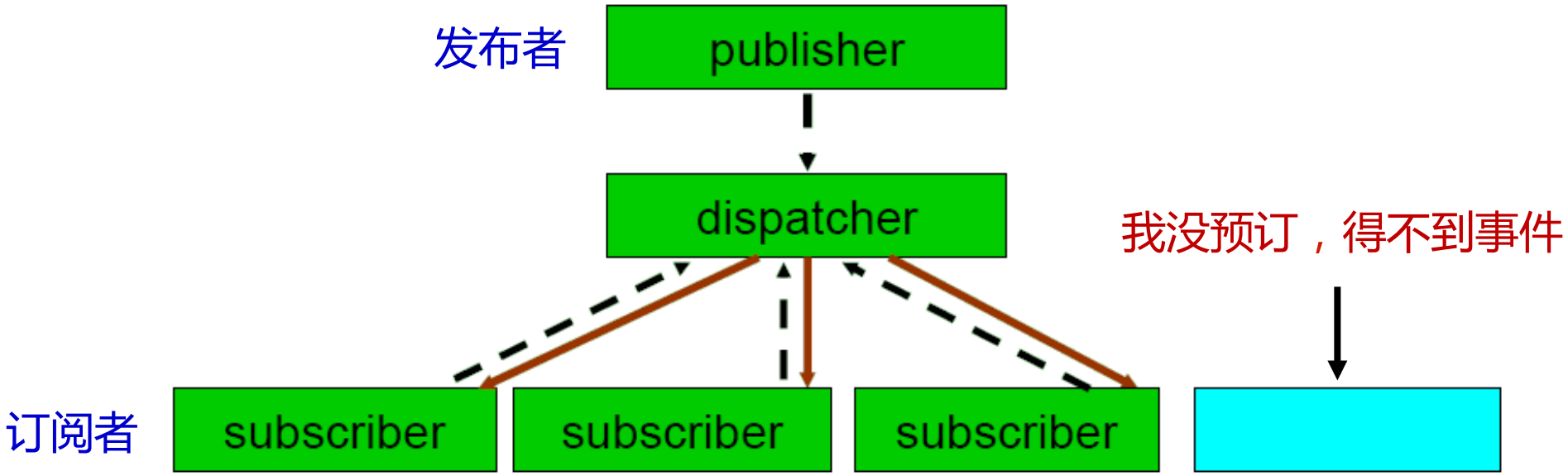
事件源：调试器

事件处理器：编辑器与变量监视器

事件管理器：IDE（集成开发环境）

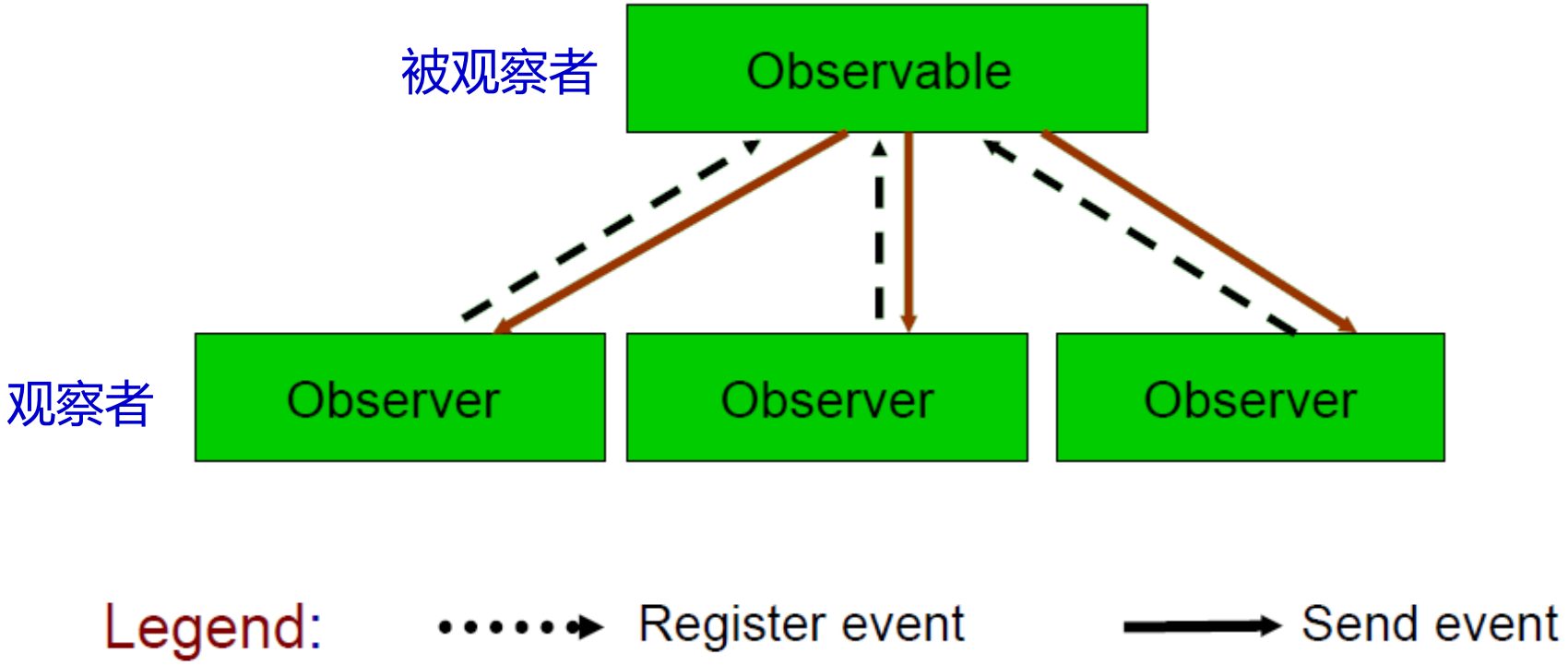


事件风格的实现策略之一：选择广播式



说明：这种方式是有目的广播，只发送给那些已经注册过的订阅者。

事件风格的实现策略之二：观察者模式



软件体系结构风格的选择

简单地判断某一个具体的应用应该采取何种体系结构是非常困难的，需要借助于丰富的经验。

绝大多数实际运行的系统都是几种体系结构的复合：

- 在系统的某些部分采用一种体系结构而在其他部分采用另外的体系，故而需要将复合几种基本的体系结构组合起来形成复合体系结构。
- 在实际的系统分析和设计中，首先将整个系统作为一个功能体进行分析和权衡，得到适宜的和最上层的体系结构；如果该体系结构中的元素较为复杂，可以继续进行分解，得到某一部分的局部体系结构。

将焦点集中在系统总体结构的考虑上，避免较多地考虑使用的语言、具体的技术等实现细节上。

软件体系结构风格的选择

技术因素

- 使用何种构件、连接件
- 在运行时，构件之间的控制机制是如何被共享、分配和转移
- 数据如何通讯
- 数据与控制如何交互

质量因素

- 可修改性：算法的变化；数据表示方式的变化；系统功能的可扩展性
- 性能：时空复杂性
- 可复用性

基于经验的选择原则

- 层次化的思想在任何系统中都可能得到应用
- 如果问题可分解为连续的几个阶段，那么考虑使用顺序批处理风格或管道-过滤器风格
- 如果核心问题是应用程序中数据的理解、管理与表示，那么考虑使用仓库或者抽象数据类型（ADT）/OO风格
- 如果数据格式的表示可能发生变化，ADT/OO可限制这种变化所影响的范围
- 如果数据是持久存在的，则使用仓库结构

基于经验的选择原则

- 如果任务之间的控制流可预先设定、无须配置，那么考虑使用主程序-子过程风格、OO风格
- 如果任务需要高度的灵活性与可配置性、松散耦合性或者任务是被动性的，那么考虑使用事件系统或C/S风格
- 如果设计了某种计算，但没有机器可以支持它运行，那么考虑使用虚拟机/解释器体系结构
- 如果要实现一些经常发生变化的业务逻辑，考虑使用基于规则的系统

基于经验的选择原则

软件体系结构的选择和应用

需要经验，也需要创新

谢谢大家！

THANKS