**Chapter 7. Modifiability**

理解：可修改性概念。

Modifiability:可修改性是关于变化的，它的重点在于改变的成本和风险。涉及变更以及进行变更的时间或金钱成本，包括此变更影响其他功能或质量属性的程度。

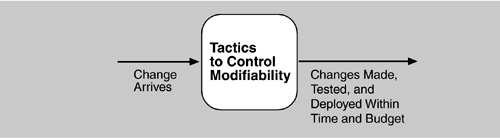
了解：可修改性一般场景。

|  |  |
| --- | --- |
| **Portion of  Scenario** | **Possible Values** |
| Source | 最终用户，开发人员，系统管理员 |
| Stimulus | 添加/删除/修改功能或更改质量属性，容量或技术的指令 |
| Artifacts | 代码，数据，接口，组件，资源，配置...... |
| Environment | 运行时，编译时，构建时，启动时，设计时 |
| Response | One or more of the following:   * 进行修改 * 测试修改 * 部署修改 |
| Response  Measure | 成本方面：   * 影响Artifacts的数量，大小，复杂性 * Effort * calendar time * 资金（直接支出或机会成本） * 此修改对其他功能或质量属性的影响程度 * 引入了新的缺陷 |

Example: 开发人员希望通过在设计时修改代码来更改用户界面。 这些修改在三小时内没有副作用。

掌握：可修改性战术。可修改性设计清单。

Tactics goal: 控制变更的复杂性，以及进行变更的时间和成本。



降低改变成本的策略包括使模块更小，增加内聚力和减少耦合。 延迟绑定还可以降低进行更改的成本。

Tactics:



1. **Reduce size of a module:**
2. **Split module(拆分模块)：**如果要修改的模块包含大量功能，则修改成本可能会很高。 将模块精简为几个较小的模块应该可以降低未来变更的平均成本。
3. **Increase cohesion(凝聚力) (SRP):**
4. **Increase Semantic Cohesion(语义连贯性):** 如果模块中的职责A和B不能用于相同的目的，则应将它们放在不同的模块中。 这可能涉及创建新模块，或者可能涉及将责任移交给现有模块。
5. **Reduce coupling(耦合)(OCP):**
6. **Encapsulate(封装):** 封装引入了模块的显式接口。
7. **Use an Intermediary(中介)：**考虑到责任A和责任B之间的依赖关系，可以通过使用中介来打破依赖关系。
8. **Restrict Dependencies(限制依赖性):** 限制给定模块与之交互或依赖的模块。
9. **Refactor(重构)：**当两个模块受同一变化影响时所采取的策略，因为它们（至少部分地）相互重复。
10. **Abstract Common Services(摘要共同服务)。**
11. **Defer building:** 一般来说，生命周期的后期我们绑定的值越多越好。如果我们设计具有内置灵活性的artifacts，那么实现这种灵活性通常比手动编码特定变化更便宜。然而，为了促进后期绑定更加昂贵，将机制（mechanisms）放在适当的位置。

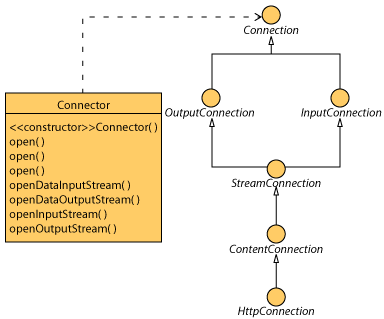
Design modifiability:

1. 确定可能发生的更改或更改类别。 对于每个潜在的变化或变化类别,确定需要添加，修改或删除以进行更改的职责。确定更改会影响其他职责。分配将在同一模块中一起更改的职责，以及将在不同时间在不同模块中更改的职责。
2. 确定在运行时可以更改哪些功能或质量属性以及这会如何影响协调。请确保此类更改会影响少量模块集。
3. 确定用于协调的设备，协议和通信路径可能会发生变化。 确保更改的影响仅限于一小组模块。
4. 对于那些需要考虑可修改性的元素，请使用协调模型来减少耦合（如发布/订阅），推迟绑定（如企业服务总线）或限制依赖性（如广播）。

Study case(看看就好系列):

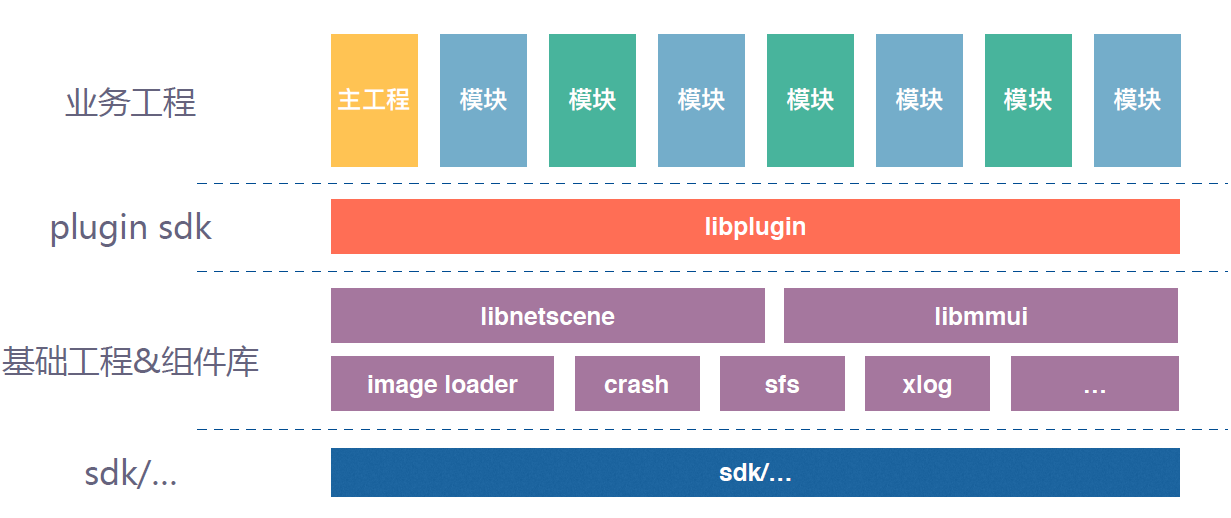
* Generic Connection Framework in MIDP 2.0

在 MIDP 1.0 中，该框架受到限制，只能处理 HTTP 连接，MIDP 2.0 使用工厂设计模式增加了 HTTPS、串口连接、套接字和更多连接。

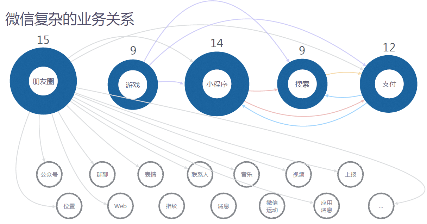
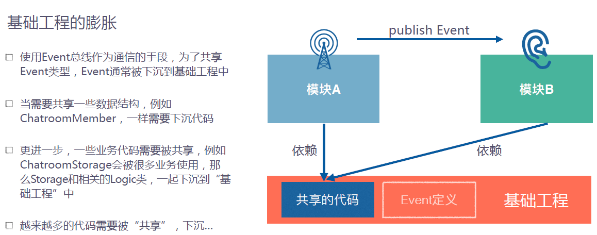


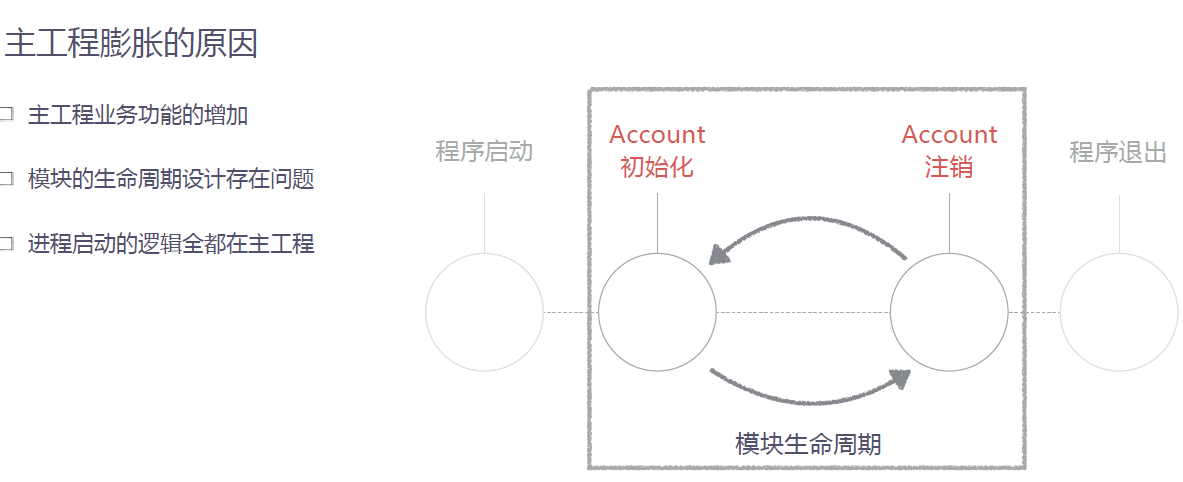
* 微信Android模块化架构重构实践

架构：



问题：



重构，正确生命周期：

