软件系统与体系结构作业 2

161250049　软件工程 金鑫

作业具体要求:
分析常见体系结构模式（即分层，代理，MVC，管道过滤器，客户端 - 服务器，P2P，面向服务，发布 - 订阅，共享数据，多层和映射 - 减少）和质量属性之间的关系（ 通过识别它们对不同质量属性的可能影响，在讲座中引入了可用性，互操作性，性能，安全性，可测试性和可用性。 使用具有每个质量属性的体系结构模式的表（矩阵）来显示模式中应用的策略（下面的可修改性示例），然后分析对质量属性的影响可能带来的好处和惩罚。 每个模式可能会出现。 讨论可能与大多数模式无关的策略。

一、分析矩阵

模式	安全性																
	检测攻击				抵制攻击								对攻击做出反应			从攻击中恢复	
	检测焚烧	检测拒绝服务	验证邮件完整性	检测消息延迟	识别角色	验证角色	授权角色	限制访问	限制曝光	加密数据	单独的实体	更改默认设置	撤消访问权限	锁定电脑	通知角色	维护审计跟踪	恢复 见到可用性
分层模式				×	×	×	×			×	×	×			×		×
管道和过滤器模式				×						×							
黑板模式			×	×								×					
代理模式		×	×	×	×	×	×										×
模型-视图-控制器模式		×	×	×	×	×	×	×	×	×	×				×		×

点对点模式				×				×			×	×				×	
事件总线模式	×			×			×				×						×
客户端-服务器				×			×						×				×
主从设备模式			×	×					×		×		×				×
解释器模式				×						×							×

模式	性能											
	控制资源需求						管理资源					
	管理采样率	限制事件响应	确定事件的优先顺序	减少开销	约束执行时间	提高资源效率	增加资源	介绍并发	维护多个计算副本	维护多个数据副本	绑定队列大小	安排资源
分层模式		×	×	×	×							×
管道和过滤器模式						×						
黑板模式						×						
代理模式						×						
模型-视图-控制器模式						×	×			×		×
点对点模式		×			×							×
事件总线		×										×

模式												
客户端 - 服务器		×				×						×
主从设备模式		×						×				
解释器模式							×					

模式	互通性		
	定位	管理界面	
	发现服务	编排	裁缝界面
分层模式	×	×	×
管道和过滤器模式	×		
黑板模式	×		
代理模式	×		
模型-视图-控制器模式	×	×	×
点对点模式	×		
事件总线模式	×		
客户端 - 服务器	×	×	
主从设备模式	×		
解释器模式	×		

模式	可靠性																											
	检测故障									从故障中恢复														防止错误				
										准备和修复									再引入									
	Ping / Echo	监控	心跳	时间戳	理智检查	状态监测	表决	异常检测	自我测试	主动冗余	被动冗余	备用	异常处理	回滚	软件升级	重试	忽略错误行为	降解	重构	阴影	状态重新同步	不断升级重启	不间断转发	从服务中删除	交易	预测模型	例外预防	增加能力集
分层模式	×					×		×					×		×													
管道和过滤器模式								×																				

黑板模式							×							×												
代理模式							×																			
模型-视图-控制器模式							×	×	×	×				×											×	
点对点模式							×			×					×						×			×		
事件总线模式							×																			
客户端-服务器							×			×							×				×					
主从设备模式							×							×												
解释器模式						×									×											

模式	可测试性							
	控制和观察系统状态						限制复杂性	
	专业接口	录制/播放	本地化状态存储	抽象数据源	沙盒	可执行断言	限制结构复杂性	限制非确定性
分层模式	×		×			×	×	×
管道和过滤器模式	×							
黑板模式	×						×	
代理模式	×					×		

模型-视图-控制器模式	×		×			×	×	×
点对点模式	×					×		
事件总线模式	×							
客户端-服务器	×		×			×		
主从设备模式	×		×					
解释器模式	×							

模式	可用性						
	支持用户计划				支持系统倡议		
	取消	撤销	暂停/恢复	并集	维护任务模型	维护用户模型	维护系统模型
分层模式	×	×		×	×	×	×
管道和过滤器模式	×	×		×	×	×	
黑板模式			×				
代理模式		×	×				

模型-视图-控制器模式	×	×		×	×	×	×
点对点模式	×	×				×	
事件总线模式	×	×				×	
客户端-服务器	×	×				×	
主从设备模式	×	×				×	
解释器模式	×	×	×		×		×

二、优缺点

名称	优点	缺点
分层模式	一个较低的层可以被不同的层所使用。层使标准化更容易，因为我们可以清楚地定义级别。可以在层内进行更改，而不会影响其他层。	不是普遍适用的。在某些情况下，某些层可能会被跳过。
客户端-服务器模式	很好地建立一组服务，用户可以请求他们的服务。	请求通常在服务器上的单独线程中处理。由于不同的客户端具有不同的表示，进程间通信会导致额外开销。
主从设备模式	准确性——将服务的执行委托给不同的从设备，具有不同的实现。	从设备是孤立的：没有共享的状态。主-从通信中的延迟可能是一个问题，例如在实时系统中。这种模式只能应用于可以分解的问题。

管道-过滤器模式	展示并发处理。当输入和输出由流组成时，过滤器在接收数据时开始计算。轻松添加过滤器，系统可以轻松扩展。过滤器可重复使用。 可以通过重新组合一组给定的过滤器来构建不同的管道。	效率受到最慢的过滤过程的限制。从一个过滤器移动到另一个过滤器时的数据转换开销。
代理模式	允许动态更改、添加、删除和重新定位对象，这使开发人员的发布变得透明。	要求对服务描述进行标准化。
点对点模式	支持分散式计算。对任何给定节点的故障处理具有强大的健壮性。在资源和计算能力方面具有很高的可扩展性。	服务质量没有保证，因为节点是自愿合作的。安全是很难得到保证的。性能取决于节点的数量。
事件总线模式	新的发布者、订阅者和连接可以很容易地添加。对高度分布式的应用程序有效。	可伸缩性可能是一个问题，因为所有消息都是通过同一事件总线进行的。
模型-视图-控制器模式	可以轻松地拥有同一个模型的多个视图，这些视图可以在运行时连接和断开。	增加复杂性。可能导致许多不必要的用户操作更新。
黑板模式	很容易添加新的应用程序。扩展数据空间的结构很简单。	修改数据空间的结构非常困难，因为所有应用程序都受到了影响。可能需要同步和访问控制。
解释器模式	高度动态的行为是可行的。对终端用户编程性提供好处。提高灵活性，因为替换一个解释程序很容易。	由于解释语言通常比编译后的语言慢，因此性能可能是一个问题。