# 论软件设计模式及其应用

【摘要】

2016年6月，我有幸参与了某地铁集团公司的基础信息平台建设，我在其中担任系统架构设计师一职。该平台以OA为中心，建设统一的流程引擎，表单系统，文件存储服务，Office文档在线处理服务，统一用户，单点登录，消息中心等基础信息服务模块，通过此平台可方便集成，扩展和二次开发。本文结合作者实践，就该平台的设计模式的应用主题讨论。该平台采用的多种多样的设计模式来保障系统质量，在流程引擎模块中采用解释器模式对审批节点进行灵活编排，达到系统的高度灵活和可扩展性；在消息中心模块中采用观察者模式，达到了消息的灵活分发。此次项目通过以上技术，系统的质量得到很好的保障，项目得以圆满成功，得到客户的一致好评。本文最后也总结了一些此次项目的不足与一些改进的思考。

【正文】

随着大城市的地铁的建设基础设施建设越来越快，地铁集团的队伍也越来越大，集团内部的信息化水平已经满足不了日常的运营办公，内部对信息系统的需求日渐突出，由此地铁集团公司高层决定开发一套信息基础平台来满足公司现阶段的业务处理和对日后的发展做好信息化支撑。我所在的某信息技术公司通过层层筛选有幸得到此项目，随后在2016年6月我们展开了项目团队的组建，其中由我担任系统架构设计师。经过和地铁公司信息部门领导的一系列沟通探讨，综合地铁集团的整体情况，我们进行了详细需求分析，得到了初步的需求规格说明书。由我领导的设计团队开始对整个平台进行模块划分、架构设计并对每一个子系统进行相应的系统建模，功能设计。整个平台以OA办公系统为中心展开建设，分别开发了流程引擎服务，动态表单服务，在线Office服务，统一用户服务，单点登录和消息中心等服务。

以下对平台部分功能模块进行简单描述。流程引擎服务采用BMPN2.0协议进行流程模板的定义，前端采用Flowchart.js进行可视化流程设计，后台采用Kafka消息队列技术进行任务的处理请求削峰，让每个请求得到妥善的处理 ，任务审批，节点流转。动态表单服务，功能包含业务表单构建服务，通过Web可视化拖拽方式完成表单结构定义用JSON表示；表单渲染服务通过表单的的定义，采用JS脚本语言构建Web的响应式页面，已完成表单渲染，采用响应式Web设计的表单能够根据客户端的尺寸来动态调整表单控件的布局和样式，达到很好的兼容性。

现在常见的设计模式分为创建型、结构型和行为型三种。创建型设计模式主要负责对象实例化的过程进行抽象封装起来，方便调用；创建模式包括单例模式、工厂方法模式、构建器模式、原型模式、抽象工厂模式。结构型模式主要负责将类或者对象实例组装起来，形成一个更大的结构，达到高类聚低耦合的效果；结构型模式包括适配器模式、桥接模式、组合模式、装饰器模式、享元模式、外观模式、代理模式。行为型模式主要负责划分类或者对象之间的责任和交互关系；行为型模式包括策略模式、观察者模式、责任链模式、命令模式、解释器模式、迭代器模式、状态模式、中介者模式、备忘录模式、模板方法模式、访问者模式。

从设计模式层次来看，我带领的设计师团队采用了多种设计模式，下面主要从以下几个模式进行详细描述。

一、解释器模式在流程引擎服务的应用。在流程引擎服务的建设中，由于流程引擎服务需要到达高度的可定制性和扩展性，我们的设计师团队一致决定采用解释器模式完成流程引擎的运行流转和解析工作。首先我们制作了一个流程设计工具，完成一个流程定义，形成一个具有语法规则的XML文档，其中包含了流程审批节点、条件节点、分支节点、跳转节点和动态扩展节点等等。其次在流程运行时服务中，将设计好的流程定义XML文档和当前流程状态表输入流程解释器，解释器利用既定的语法规则进行相应的业务逻辑处理，在动态审批节点部分还需要进行流程实例信息的变更，返回下一批流程节点，然后运行时服务进行相应任务条目的生成。当用户处理了相应的任务后，通过调用解释器进行流程实例XML定义的更新，以便跟踪流程图的变化状态。通过采用解释器模式，我们的流程引擎很好的支持了客户在业务流程方面的灵活性和可扩展性要求，保障了系统的可持续发展，为我们和客户都带来了巨大的价值。

二、观察者模式在消息中心服务的应用。在消息中心服务的建设中，我们采用了观察者模式进行应用消息分发的功能块的设计。由于平台的消息需要分发到不同的地方，比如门户系统、OA系统、知识中心、移动端等等，采用观察者模式后可以方便的添加或删除消息分发目的地，而不会影响到其他模块消息的分发，灵活的将代码进行了解耦，利于后期的维护。首先我们定义了一个接口IObserver表示观察者接口里面包含方法定义ReceivedMessage接受到消息；其次分别实现不同模块消息分发的类实现该接口，这里我们用Observer1、Observer2来临时表示下具体的观察者；然后我们构建了一个MessageClient类来统一发送消息，在这个类里面定义一个类型为IObserver数组的静态属性Observers，在MessageClient中定义一个方法AddObserver，用以将一个观察者加入到静态属性Oberservers数组中，例如这里将Observer1和Observer2的实例加入进来。最后在MessageClient中定义一个SendMessageToAll方法，功能是循环调用Observers中的对象的ReceivedMessage方法完成消息的推送。通过采用观察者模式，代码质量得到了很好的保障，方便了开发人员之间的交流，利于后期的维护了扩展，到达了我们预期的良好效果。

本平台于2017年9月份正式在地铁集团上线，通过我们对一系列的设计模式的利用，保障了系统的质量还同时提高了系统的可维护性，还有效的减少了系统的运维成本。通过3个月时间的无故障运行后，最终项目得到客户的高度认可，并对我们整个项目组做出了由衷的赞赏。

每一个软件的开发都并非完美，此次项目，我也发现了一些缺憾的地方，比如在设计模式的利用还有一些值得改进的地方，没有完全按照标准的方式来构建代码；在利用设计模式的过程中有些开发人员对设计模式的理解不到位，在前期培养的过程中花费了很大的力气。总体来说，这次让我能够更加全面的认识设计模式对于整个软件开发的重要性，并在今后的职业生涯中受益匪浅。