摘要:

2017年7月，我参与某互联公司自主研发全国运营的网约车出行平台。该平台主要是为了解决客户在线预约网约车。包含用户叫车、自动派单、司机接单、行程结算、车辆管理等功能模块。我在该项目中担任系统架构师设计师一职，主要负责整个系统的架构设计。本文以该项目为例，主要论述软件系统机构风格在该项目的具体应用。通过规则系统来实现营销规则的自定义达到通过规则系统来自定义营销的目的；通过隐式调用风格实现系统配置的更新达到系统参数可以支持系统参数动态调整的目的；通过层次结构风格对系统进行层次划分实现架构复用可以达到子系统的拆解和系统组件的抽取提升代码复用目的。事实证明，使用这些技术手段使得项目整体能够克服项目中遇到的各种问题。最终项目得以顺利完成，取得预期目标，获得用户好评。

正文:

2017年7月，我参与某互联公司自主研发全国运营的网约车出行平台。我公司致力于网络出行市场服务是首批获得当地“网约车出行牌照”, 并且先后获得四川、河南、海南、山东等各地的合法运营牌照背景下。以构建一个合法、合规安全的网约车出行环境为使命。以解决用户打车难、提升用户的出行幸福感，拉动了产业链的发展，构建了一个完整的出行生态链的为目标，构建一个全国性的网约车出行平台。 网约车出行平台包括乘客端、司机端、后台管理系统三部分组成。乘客端供乘客查询车辆、发布订单、支付车费、评论司机；司机端供司机车辆信息认证、出车接单、乘客接送、车费提现等；管理系统主要是提供系统报表查询、规则配置、乘客管理、司机管理、分公司管理、账务管理等。本平台提供了专车、快车、出租车业务，乘客可以通过自身的需求来选择不同类型业务出行，司机需要上传自身拥有的运营车辆等证照信息到平台审核，只有当平台审核通过后才能正常的再平台上合法运营。我在该项目中担任系统架构师设计师一职，主要负责整个系统的架构设计。

出行平台定位是一个全国性的互联网出行平台，在系统的设计过程我们结合项目情况解决架构复用高效以及高效组织构建完整的系统。我们分析了常用的几种系统架构风格：1、虚拟机架构风格它包含解释器和规则系统风格，解释器风格包含解释引擎，代码存储区，解释引擎当前工作状态，以及代码被解释执行的进度。在解释器风格的系统中含有一个虚拟机，可以仿真硬件执行的过程和一些关键的应用，缺点是执行效率比较低。规则系统主要是包括规则包、规则解析器、规则/数据选择器和工作内存。一般用于风控决策、人工智能等2、独立构建风格它包含进程通讯和隐式调用。进程通讯它的构建是独立的过程连接件是消息传递，构件通常是命名过程，消息传递可以是点对点、异步或者同步的方式来进行远程调用。隐式调用构件不直接调用过程，而是触发一个广播或者多个事件，交互的连接件以过程之间的隐式调用来实现的。主要优点为软件复用提提供了强大的支持，为构件的维护和升级带来了便利。缺点就是放弃了对系统计算的控制。下面就具体论述系统中用到的几种架构风格的实施过程。

1、通过规则系统来实现营销规则定义。

在系统开发过程中，运营部门给我提到希望能够定制营销规则。支持在用户和系统交互的各个节点能够自定义营销内容。如：可以在用户注册、推荐新用户、分享朋友圈下载APP、下单立减、下单随机立减等。都需要通过规则的定义然后当客户到达一定触发节点后来执行这些规则最后给出营销结论。为了解决这个运营问题我们采用了虚拟机架构风格中的规则系统来实现，我们开发了一个基于规则的营销管理系统。首先我们需要定义规则集每个规则包含：营销场景、用户节点、触发规则、奖励规则等规则参数。然后将这些规则参数定义到规则库中存储。针对每一个规则变量我们需要在规则解析器中添加对应的规则解析器。如果客户来触发这些规则就通过数据/规则选择器来执行响应的规则包，执行完规则包后根据对应的营销规则结论。把最终的营销策略或者营销奖励发放到用户的个人中心。通过规则系统的使用能够很好的为系统提供规则的定制而且不需要再调整规则参数阀值得时候修改代码。增加了系统的健壮性。

2、通过隐式调用风格实现系统配置的更新

在系统中我们通过隐式调用风格实现系统配置基于广播通知的方式来实现更新，在系统开发的初期我们采用配置文件的方式将系统的配置写入到对应文件中，配置文件会随着项目发布文件一同打包。在项目运行一段时间过后发现有些系统模块或者系统功能参数阀值需要调整，来达到运营控制或者存在业务风险需要调整。此时我们只能通过修改项目配置文件中的参数对项目进行打包重新发布系统，这个过程会导致业务的中断降低系统业务运行的连续性。针对这个问题我们决定采用隐式调用的方式来刷新配置。我们将我们的业务子系统接入到配置中心上，就是在每个业务系统启动的过程中需要注册到配置中心，如果我们在配置中心上配置并且发布配置过后。配置中心会过广播的方式来广播对应的变更系统参数信息内容通知对应的接入业务系统，业务系统收到通知后就会调用对应的刷新配置过程来刷新配对象实现系统参数配置的变更，不需要重新启动服务。从而保证了业务的连续性，也提升了系统的灵活性。

3、通过层次结构风格对系统进行层次划分实现架构复用

在系统中我们通过层次结构风格对系统进行层次划分实现架构复用，由于在系统的开发过程中我们会按照业务的需要定义一些组件来满足基础的业务需求，在各个系统中会存在很多能够复用的模块或者程序代码，如：各种工具类、数据库连接组件、协议解析组件等。我们在项目的初期抽象了一些基础组件：通用工具组件、数据库组件、协议解析组件等。然后我们系统进行子系统拆分了用户系统、订单系统、账务系统、营销系统等。每个系统都需要做数据的持久化、报文协议解析、数据缓存等。然后我们在依次对此抽离成一个独立组件的组件。每当我们拆分出一个系统或者组件我们发现我们的架构复用度将更高。可以让一些复杂的问题能够按照步骤逐渐的被分解，简单化。但是我们系统依赖层次更加复杂。目前来说很难找到一个很正确的合适的层次抽象方法。不过对于目前系统中的层次拆分还是相对比较合理对于，能够实现很好的架构复用和复杂问题的拆解，而且便于我们对组件版本的规范化管理。

经过全体成员的不懈努力。在2018年2月，先后在四川、重庆、河南、贵州、海南等城市全国开展内测。2018年6月，全国正式发布运营。上线1年多程序一直稳定可靠运行。无较大线上生产事故、系统平稳运行。通过规则系统、隐式调用、层次结构风格规范了系统架构的定义，增加了系统架构的复用性。同时也满足了项目的业务需求。上线以来得到了多个地区交通部门的点名表扬和和上万用户的好评。也为我们后续的开发、迭代、运维奠定了一个良好的基础。

项目上线至今运行1年多进入产品优化迭代阶段一直运行稳定运行，无较大生产事故。能够通过架构风格运用能够很好的实现架构的复用，层次的划分以及提升架构的灵活性。实践证明，项目能够顺利上线，并且稳定运行，性能良好。与系统架构风格的运用是离不开的，他能够反映系统中的共性结构和语意特征，并且能够指导各个模块和子系统有效的组织完成一个完整的系统，而且实现了不同系统的代码共享，能够按照常用的规范化的方式来组织，便于不通的开发人员能够很容易的理解系统架构。