**摘要:** 本人于2016年1月参与浙江省某市公交集团“公交车联网一体化“项目，该系统为新能源营运车辆补贴监管、安全监控等方面提供全方位的软件支撑，在该项目组中我担任系统架构师岗位，主要负责整体架构设计与中间件选型。本文以该车联网项目为例，主要讨论了软件架构风格在该项目中的具体应用。底层架构风格我们采用了虚拟机风格中的解释器，因该公交共有几十种不同的数据协议，使用解释器风格可以满足整车数据协议兼容性需求；中间层关于应用层的数据流转我们采用了独立构件风格中的隐式调用，这种风格主要用于减低系统间耦合度、简化软件架构，提高可修改性方面的架构属性；应用系统层我们采用了B/S的架构风格，统一解决公交行业性难题“实施推广难、维护难”问题。最终项目成功上线，获得用户一致好评。

随着国家十三五计划中-能源战略的深入和推广，该市公交集团自2016年1月起全面停止采购燃油机公交车，规划到2020年纯电公交车采购占比必须在70%以上，同时配套将车联网方面的系统建设被列为工作重点。不管从新能源营运车辆补贴监管、安全监控或者公交公司自身的营运和机护需求，都要求有新的车联网系统对他们进行全方位的支持，而我司是该公交的主要仪表与can模块产品的主要供应商，全市4000多台车中有3000多辆是我司的产品，我司不仅掌握熟悉该公交整车数据而且在车联网底层can数据有非常明显的领域知识优势，因此2016年1月我司被该市公交集团委托建设公交集团车联网一体化项目。本项目组全体成员共有27人（不含业主方），我在项目中为担任系统架构师职务，架构小组共4人，我主要职责负责整体架构设计与中间件选型，4月份完成架构工作，整个项目共耗时了7个月，2016年8月顺利通过验收。  
在架构工作开始阶段，我们便意识到，架构风格是一组设计原则，是能够提供抽象框架模式，可以为我们的项目提供通用解决方案的，这种能够极大提高软件设计的重用的方法加快我们的建设进程，因此在我司总工程师的建议下，我们使用了虚拟机风格、独立构件风格以及B/S架构风格这三种较常用风格。虚拟机风格中的解释器架构风格能够提供灵活的解析引擎，这类风格非常适用于复杂流程的处理。独立构件风格包括进程通讯风格与隐式调用风格，我们为了简化架构复杂度采用了隐式调用风格，通过消息订阅和发布控制系统间信息交互，不仅能减低系统耦合度，而且还提高架构的可修改性。B/S架构风格是基于浏览器和服务器的软件架构，它主要使用http协议进行通信和交互，简化客户端的工作，最终减低了系统推广和维护的难度，以下正文将重点描述架构风格的实施过程和效果。  
底层架构我们使用解释器风格来满足整车数据协议兼容性需求。解释器风格是虚拟机风格中的一种，具备良好的灵活性，在本项目中我们的架构设计需要兼容好86种不同can数据协议，一般来说这种软件编写难度非常高，代码维护难度压力也很大，因此这个解释器的设计任务便很明确了，软件设计需要高度抽象、协议的适配由配置文件来承担。具体的做法如下，我们对各个车厂的can数据结构进行了高度抽象，由于can数据由很多数据帧组成，每个数据帧容量固定并且标识和数据有明确规定，因此我们将can协议中的ID和数据进行关系建模，将整体协议标识做为一个根节点，以canid作为根节点下的叶子节点，使用XML的数据结构映射成了有整车协议链-数据帧-数据字节-数据位这4层的数据结构，核心的代码采用jdom.jar与java的反射机制动态生成java对象，搭建一套可以基于可变模板的解释器，协议模板的产生可以由公交公司提供的excel协议文档进行转换得到，解释器支持协议模板热部署，这种可以将透传二进制数据直接映射成java的可序列化对象，将数据协议的复杂度简化，后期数据协议更改不会对软件产生影响，仅仅更改协议模板文件即可，最终我们使用了86个协议描述文件便兼容了这些复杂的can数据协议，规避了can数据巨大差异带来的技术风险。  
中间层我们使用独立构件风格中的隐式调用来简化构件间的交互复杂度，降低系统耦合度。主要的实现手段是，我们采用了一个开源的消息中间件作为连接构件，这个构件是apache基金会下的核心开源项目activemq，它是一款消息服务器，其性能和稳定性久经考验。由上文提到的解释器解析出对象化数据经过activemq分发到各个订阅此消息的应用系统，这些应用系统包括运营指挥调度、自动化机护、新能源电池安全监控等，这种多web应用的情况非常适合采用消息发布与消息订阅的机制，能够有效解决耦合问题，我们在编码的过程中发现只要采用这种风格的web应用，整个迭代过程效率极高，错误率降低，而且我们使用的spring框架，消息队列的管理完全基于配置，清晰简单，维护性良好，例如整车安全主题、运营调度主题、机护维修主题等消息队列分类清晰，可以随时修改其结构也能够随时增其他主题的消息队列，不同的web系统监听的队列也可以随时变换组合，基于消息中间件的架构设计能够让系统的构件化思路得到良好实施，总体来说这种架构风格带来了非常清晰的数据流转架构，简化了编码难度，减低本项目的二次开发的难度。  
应用系统层我们主要采用B/S的架构风格，主要用于解决公交推广难、维护难得问题。公交行业有一个明显的特点，公交子公司分布在全市各个地区，路途很远，且都是内网通讯，车联网络也是走的APN专网，一般是无法远程支持的，这给我们的系统推广以及后期维护带来了很大难题，我们可以想象如果使用C/S架构，更新客户端一旦遇到问题很可能需要全市各个站点跑一遍。这让我们在系统推广和维护方面面临较大压力。我们采用的B/S架构风格能够解决这个难题，并充分考量可现在的相关技术成熟度，例如现在的html5完全能够实现以前客户端的功能，项目中我们使用了大量的前端缓存技术与websocket技术，能够满足公交用户实时性交互等需求。这种风格中页面和逻辑处理存储在web服务器上，维护和软件升级只要更新服务器端即可，及时生效，用户体验较好，例如界面上需要优化，改一下Javascript脚本或者CSS文件就可以马上看到效果了。  
项目于2016年8月完成验收，这 1年内共经历了2次大批量新购公交车辆接入，这几次接入过程平稳顺利，其中协议解释器软件性能没有出现过问题，消息中间件的性能经过多次调优吞吐量也接近了硬盘IO极限，满足当前的消息交互总量，另外由于我们的项目多次紧急状态下能够快速适应can协议变动，得到过业主的邮件表扬。除了业主机房几次突发性的网络故障外，项目至今还未有重大的生产事故，项目组现在留1个开发人员和1个售后在维护，系统的维护量是可控的，系统运行也比较稳定。  
不足之处有两个方面，第一在架构设计的过程中我们忽略PC配置，个别PC因为需要兼容老的应用软件不允许系统升级，这些电脑系统老旧，其浏览器不支持html5，导致了系统推广障碍。第二在系统容灾方面还有待改善。针对第一种问题，我们通过技术研讨会说服可业主新购PC，采用两台机器同时使用方式解决。针对第二种问题我方采用了服务器冗余和心跳监测等策略，在一台服务暂停的情况下，另外一台服务接管，以增加可用性。