摘要：

本人于2018年1月参与了中石化XX油田XX采油厂“用电管理系统”的项目建设，该系统建设目标是实现分单位、分线路、分系统评价、优化、考核，全面提升采油厂用电管理水平。在该项目组中我担任系统架构师一职，主要负责系统整体架构设计。本文以该项目为例，先介绍几种常用的软件架构风格及其特点，然后讨论软件架构风格在该项目中的具体应用。整个系统采用调用返回风格中的面向对象及层次结构风格设计，主要包括数据层、中间层、应用层。其中数据层采用虚拟机风格中的解释器来满足多种数据协议的兼容性需求；中间层采用了数据流、独立构件等多种架构风格以减低系统间耦合度、简化软件架构，提高系统灵活性、可修改性等方面的架构属性；应用层采用了MVC设计模式实现了数据、显示和处理分离。最终系统上线后，获得了用户的一致好评。

正文：

“用电管理系统”项目是采油厂能源管控中心系统的一个子系统。能源管控中心是中石化集团公司十三五规划中的“能效倍增”计划在胜利油田分公司的示范应用项目，该示范项目能够在实现企业节能目标管理、能源计量统计、节能潜力识别、能效分析优化的同时，有效支撑企业实施节能技术改造、促进企业用能水平不断提升。“用电管理系统”的建设目标是建立覆盖厂、区两级用电管理一体化体系，实现分单位、分线路、分系统评价、优化、考核，达到电网运行质量实时监控、异常情况精准管控、能耗总量全面受控，按照运行产量的方式运行电量，全面提升采油厂用电管理水平。该项目功能设计参考PDCA闭环管理的理念，共设计包括用电计划、用电分析、用电优化、用电考核、设备管理等五大功能模块。

我作为单位技术骨干之一，主持并参与了项目计划制定、需求分析、整体架构设计与技术选型、底层设计、部分编码等多项工作。下面，我将首先介绍软件系统开发中常用的软件架构风格及具体含义，然后详细介绍“用电管理系统”的分析和设计过程中所采用的架构风格及原因。

软件架构风格是描述某一特定应用领域中系统组织方式的惯用模式。常用的软件架构风格有数据流风格，调用/返回风格，独立构件风格，虚拟机风格，仓库风格。数据流风格包括批处理序列风格与管道-过滤器风格，其每一步处理都是独立，顺序执行的，适用于简单的线性流程。调用/返回风格包括主程序/子程序风格，数据抽象和面向对象风格，层次结构风格，其主要思想是将复杂的大系统分解为一些小系统，以便降低复杂度，增加可修改性。独立构件风格包括进程通信风格，事件驱动系统（隐式调用）风格，其特点是每个构件都是独立的个体，它们之间不直接通信，以便降低耦合度，提高灵活性。虚拟机风格包括解释器风格，基于规则的系统风格，此类架构风格具有良好灵活性。仓库风格包括数据库系统风格，超文本系统风格，黑板系统风格，其以数据为中心，善

于管理数据信息，适合大量数据的应用场合，适用于复杂的逻辑系统。除此之外，还有DSSA，REST、分布式等架构风格。

项目启动后，在架构设计工作的开始阶段，我们便意识到，软件架构风格可以为我们的项目提供架构级的通用解决方案。这种架构级的软件重用可以极大提高我们的系统建设进程。由于油田系统对安全性、可靠性、可用性和扩展性要求很高，我们决定在公司技术顾问的建议下，采用调用返回架构风格中的面向对象和层次架构风格作为该系统的软件体系架构。最终，我们将系统分为应用层，中间层，数据层。应用层负责具体业务和视图展示，如系统首页的电网拓扑展示、单位用电、线路用电等，其又分为视图层与业务逻辑层，采用 MVC 设计模式实现了数据、显示和处理分离，中间层除负责为应用层提供通用服务支持如系统管理服务，session 管理服务等之外，还提供预报警模型、能效优化模型、优化方案推送模型等中间件支持，其又细分为中间件层、逻辑处理层与数据接口层。而数据层负责提供数据存储访问服务，如数据库服务，缓存服务，文件服务，搜索服务等。接下来，我将分层次详细介绍三层层次体系结构的设计过程。

首先是应用层。应用层主要采用 SpringMVC 这一基于 J2EE 平台的 MVC 框架，主要通过 Vue+BootStrap 等技术实现。我们将系统根据业务功能进行水平划分，这有助于代码管理与维护。我们将系统分为用电分析，用电优化，用电考核等多个子系统，这里以用电优化子系统为例。用电优化子系统主要提供油井提液单耗指标的散点图、柱状图、数据表等不同形式的分类展示、标杆井信息展示、优化方案自动推送等多个功能模块。功能模块调用中间层的服务支撑，如标杆井展示需要调用服务层由标杆计算模型的计算结果，通过模块间的通信服务，实现标杆井信息与优化方案的自动推送。另一方面为了满足用户提出的全厂电力系统拓扑图要能够自由定制的需求，我们综合考虑多种方案后选择了使用 HTML5 Canvas 技术。HTML5 是互联网的下一代标准，是构建以及呈现互联网内容的一种语言方式. 被认为是互联网的核心技术之一。最终采用 HTML5 Canvas 技术实现的拓扑图的非常轻巧，性能出色，能够支持上万图元，操作流畅，并且支持矢量图形，无极缩放等。另外还有动静分离，动态资源静态化等，这里不再赘述。

其次是中间层。中间层采用了 Spring、Shiro 等服务框架实现。随着服务器规模的扩大，开发人员的增多，每个应用都变得复杂，臃肿，存在大量代码重复，另一方面数据库的连接数压力较大，所以我们采用了服务化方案，即应用层和数据层中增加一个服务层。从结构上来看，系统结构更为清晰明了，更为立体。稳定性上来看，许多散落的代码成为了通用服务，并交付专门的团队负责运维。出于对成本与技术成熟度的考虑，我们采用了开源的 Spring 框架。通过 Spring 框架的使用，缩短了开发周期，减少了开发费用和维护费用，提高了开发的成功率。另外基于安全性方面考虑，我们选择 Apache

Shiro，它是一个功能强大且易于使用的 Java 安全框架，为开发人员提供了一个直观而全面的解决方案,用于身份验证、授权、加密和会话管理。通过使用 Spring+Apache Shiro 的分层设计，实现了各层次低耦合，高内聚,并严格遵循了 web 安全的相关规范，通过前后台双重验证，参数编码传输，密码 SHA-256 加密存储，shiro 权限验证等手段，从根本上避免了 SQL 注入，XSS 攻击，CSRF 攻击等常见的 web 攻击手段。除此之外，还有服务线程池隔离，分布请求合并，服务调用端的流控处理，异步服务调用，通过 Future 方式对远程服务调用的优化等问题，限于篇幅，不再赘述。

最后是数据层。数据层涉及缓存，文件系统，数据库，数据通知服务等模块。我们选择了 MyBatis 作为持久层框架，实现了业务逻辑和数据访问逻辑分离，使系统的设计更清晰，更易维护，更易单元测试，极大提高了系统的可维护性。由于用户对数据的访问具有集中性，为防止频繁读取数据库导致系统性能下降问题，我们选择采用开源的 J2Cache 两级缓存框架，即一级采用 Ehcache，二级采用 Redis 缓存技术，通过该框架的使用，提高了系统性能和访问效率。

最终项目成功上线，正常运行了近一年，收到各方好评。尤其是标杆井推荐与耗电量推算功能，为采油厂的节能工作提供了良好的技术和数据支持。该项目也成功获得了当年度分公司科技进步创新一等奖。在系统的架构设计中，我们引入了层次架构的设计思想，有效地降低了维护成本，提高了系统的开放性，可扩展性，可重用性以及可移植性。当然还是存在一些问题的。如系统采用 http 协议，易被非法劫持，可以将协议修改为https来解决。还有就是旧系统在数据库层面存在很多无效的存储过程与定时任务，对数据库性能造成了影响。对这种高水平高价值的遗留系统问题，我们采用了重构整合的方式进行了改造，保证了系统的延续性。这些都是我在今后的系统架构设计工作中需要注意与改进的地方，也是日后我应该努力的方向。

楚楚的资料库，仅供个人学习