计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2017,26(9):98−102 [doi: 10.15888/j.cnki.csa.005934]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 OSGI 分层动态的软件设计与开发①

魏东平, 李奉娟, 苑志朋

(中国石油大学(华东) 计算机与通信工程学院, 青岛 266580)

摘　要: 在传统的 java 框架下开发的应用软件系统缺乏模块化、动态化的管理能力. 在深入研究 OSGI 框架的基础上, 本文在井下作业管理系统的开发过程中, 采用 OSGI 规范与分层解耦思想结合的方法, 设计了系统的总体架构, 并解决了系统动态管理与服务层的问题. 本文在井下作业管理系统的开发中, 详细阐述了 OSGI 在系统中的应用, 包括 OSGI 扩展点机制以及 AOP 思想等, 提高了系统扩展能力与解决了数据同步更新的问题. 软件测试结果表明, 该模型提高系统的性能, 降低了系统各模块之间的耦合性, 实现了模块的动态管理, 增加了各组件的重用性与可扩展性, 也提高了系统的稳定性.

关键词: OSGI; 模块化; 动态管理; 井下作业管理系统

引用格式: 魏东平,李奉娟,苑志朋.基于 OSGI 分层动态的软件设计与开发.计算机系统应用,2017,26(9):98–102. http://www.c-s-a.org.cn/1003-3254/5934.html

Hierarchical Dynamic Software Model Based on OSGI

WEI Dong-Ping, LI Feng-Juan, YUAN Zhi-Peng

(College of Computer and Communication Engineering, China University of Petroleum, Qingdao 266580, China)

Abstract: The application software under the traditional java framework lacks the ability of modularization and dynamic management. Based on the in-depth study of OSGI framework, this paper proposes a method of combining the OSGI and the hierarchical decoupling. It uses the method to design the overall architecture of downhole operation management system, and also solves the problem of dynamic management and service layer. It describes the application of OSGI extension point mechanism and AOP in detail, then improves the system expansion ability and solves the problem of data synchronization update. The test results show that the model can improve the performance of the system, reduces the coupling between the modules, increases the reusability of components and scalability, and improves the stability of the system.

Key words: OSGI; modularity; dynamic management; downhole operation management system

1　概述

随着互联网技术的发展, 软件开发技术实现了由面向过程向面向对象的转变, 形成了企业级应用的开发模式. 特别是在 Java EE 开发领域, MVC(Model View Controller)开发模式、SSH 框架以及 EJB(Enterprise JavaBean)等都已经比较成熟. 但是, 基于这些技术开发的应用程序, 各个模块之间是紧耦合的, 没有考虑支持粗粒度的模块化编程[1,2].

OSGI(Open Service Gateway Initiative)是专门针对 Java 模块化制定的一个标准. OSGI 带来了规范化的模块划分、低耦合的模块间关系、统一的模块开发方式、可动态插拔的模块管理环境等. 本文研究了 OSGI 框架的特点, 结合分层解耦的思想, 以井下作业管理系统的开发为例, 详细阐述了 OSGI 在软件开发中的应用,

① 收稿时间: 2016-12-24; 采用时间: 2017-01-12

实现了井下作业管理系统的模块化动态管理.

2 相关技术

2.1 OSGI

OSGI 框架是一种分层架构, 如图 1 所示, 主要有模块层、生命周期层、服务层等. 模块层定义了 OSGI 模块的概念, 称为 Bundle. Bundle 既实现了逻辑模块化也实现了物理模块化. 生命周期层定义了 OSGI 是如何实现动态安装和管理 Bundle. 服务层使各个 bundle 之间以松耦合的方式进行交互.

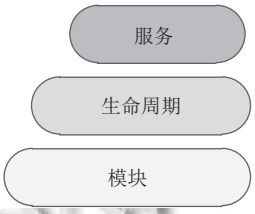

图 1 OSGI 结构图

2.2 AOP

AOP(Aspect Oriented Programming, 面向切面编程)的实现建立在 OOP 基础上, AOP 是通过关注点分离的原则将独立的逻辑片段模块化, 通过切面织入的方法将关注点应用于程序的多个地方. AOP 主要应用在权限、缓存、内容传递、错误处理、延时加载、性能优化、持久化、同步、事务等问题的解决中.

3 系统的分析与设计

3.1 系统功能分析

井下作业管理是油田企业生产管理的重要组成部分, 是油田原油生产平稳运行和产量任务完成的重要保障. 随着信息技术的不断发展, 油田企业为优化井下作业管理流程, 需要对原来的井下作业管理系统进行改版升级, 我们对各个单位进行了深入的调研, 将他们的需求进行了整合, 确定了井下作业管理系统的主要功能模块. 各主要模块及功能可概括为:

(1) 计划管理: 实现生产井年计划的录入、修改、删除以及计划的实施情况进行查询, 例如设计进度与施工进度放入查询.

(2) 设计管理: 主要是根据计划编写施工设计, 设计模块又分为地质设计、工程设计、工艺设计. 该模块实现设计 word 文档的生成、在线编辑、在线审核、查看.

(3) 施工管理: 实现施工动态、作业监督的实时动态管理, 记录每天的施工情况并编写施工总结.

(4) 用户信息管理: 主要实现用户注册、信息修改、上传签名图片, 对于系统管理员可以对用户进行业务权限和数据权限管理.

3.2 系统总体设计

本文在系统总体架构设计过程中, 遵循 OSGI 规范, 并结合 J2EE 系统开发的分层解耦设计思想. 系统总体分层结构如图 2 所示. 数据库与持久层是软件体系结构的基础. 为降低各个功能插件之间的耦合度, 将各个功能插件通用的组件封装在通用组件层, 应用 OSGI 的面向服务组件模型[3]的设计思想将业务组件发布为 OSGI 服务, 在业务逻辑层调用这些服务. 表现层与业务逻辑层实现具体的业务流程. 根据关注点分离的思想将表现层与业务逻辑层划分为多个功能插件(Bundle)实现了模块划分, 每个功能插件之间的交互尽量使用服务的形式, 降低了各个功能插件之间的耦合度. 生命周期层是对体系结构中的功能插件的生命周期进行管理. 可以安全地在体系结构中安装和卸载 Bundle, 而不需要重启应用程序.

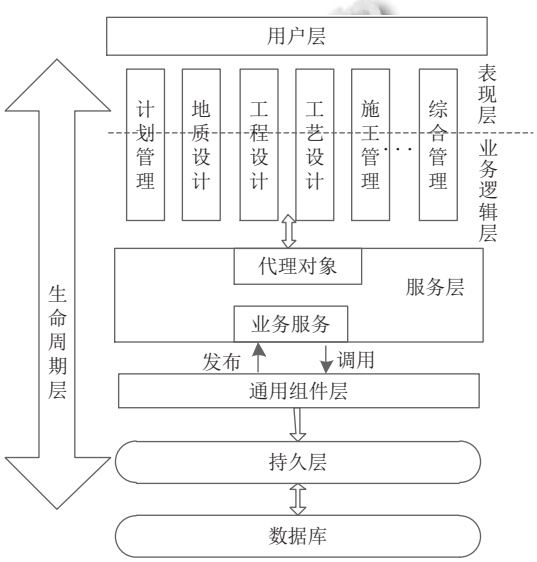

图 2 系统架构图

井下作业管理系统总体架构的设计充分体现了 OSGI 框架的三个分层, 而不是单纯的运用模块层, 但也存在一些需要解决的问题.

3.3 生命周期层

生命周期层实现了应用程序的动态管理, 相比传统的 J2EE 框架, 是很大的突破. 但是, 在软件总体架构设计中我们发现, 当一个 Bundle 被其他 Bundle 依赖时, 在更新这个 Bundle 后, 依赖这个 Bundle 的 Bundle 被加载时可能被更新为旧版本的 Bundle, 由此会引发被更新 Bundle 与其他 Bundle 的不一致, 不能真正做到动态更新. 为解决这一问题, 在动态更新 Bundle 时, 我们采取了更新-刷新控制机制[4], 具体步骤如下:

① 更新 Bundle. 此时新版本被放到了正确位置, 但旧版本的 Bundle 依然存在, 依赖它的 Bundle 依然从旧版本加载类, 维持有依赖关系的 Bundle 的稳定性.

② 从更新的 Bundle 开始, 重新计算受影响并存在依赖关系的 Bundle 的结构图, 任何引用了更新 Bundle 的导出包的 Bundle 都会被放到结构图中. 当结构图外部不存在引用了结构图内 Bundle 的 Bundle 时, 该结构图才被认定为完全绘制完成.

③ 结构图中处于 Active 状态的 Bundle 被停止并切换至 Resolved 状态, 处于 Resolved 状态的 Bundle 都会被切换至 Installed 状态, 处于 Uninstalled 状态的 Bundle 会被从结构图移除.

④ 对于结构图中余下的 Bundle, 框架会重启之前处于 Active 状态的 Bundle, 并对这些 Bundle 以及其所依赖的 Bundle 进行解析.

⑤ 框架触发 Refreshed 类型的事件, 对整个框架进行更新操作.

3.4 服务层

服务是实现 Bundle 间交互的途径. 使用服务能够降低 Bundle 之间的耦合, 更加有利于软件的重用. 目前比较常用的发布与使用服务的方式有声明式服务和 Blueprint. 在软件总体架构设计过程中, 我们对这两种方式的内部原理进行了研究. 声明式服务采用的是级联的方式, 也就是激活或停用组件基于依赖是否能够满足, 不能很好地应对在 Bundle 更新期间服务取消和发布对框架的影响. 而在 Spring DM (Spring Dynamic Modules)基础上定义的 Blueprint 容器规范, 即 Blueprint 是一个 OSGI 的 IOC(Inversion of Control)[5] 规范, 采用了代理的方式, 将代理注入到组件中, 而不是注入到真正的服务对象中. 利用代理提供服务动态性阻塞的机制, 当服务从服务注册中心移除时, 阻塞机制会阻塞调用线程并阻塞等待一段时间, 若超时到期并没有找到新的服务替代, 代理就会抛出异常, 所以在 Bundle 更新期间, 这种方式可以减少服务取消发布对框架的影响. 基于以上研究, 在模型的服务层采用了 Blueprint 的方式发布引用服务.

4 系统实现

4.1 系统各功能模块的关系

井下作业管理系统主要由计划管理、设计管理、施工管理等模块组成, 从图 3 的抽象实体类之间的关系可以看出, 计划模块要反映设计状态以及施工进度的更新, 所以设计、施工的数据更新要与计划的数据更新同步. 传统的方法是在数据库建立视图, 但因关联的表及属性众多, 对视图的管理非常复杂. 我们在系统开发中借助 AOP 思想解决了这一难题, 下面以设计管理的实现为例做简单说明.

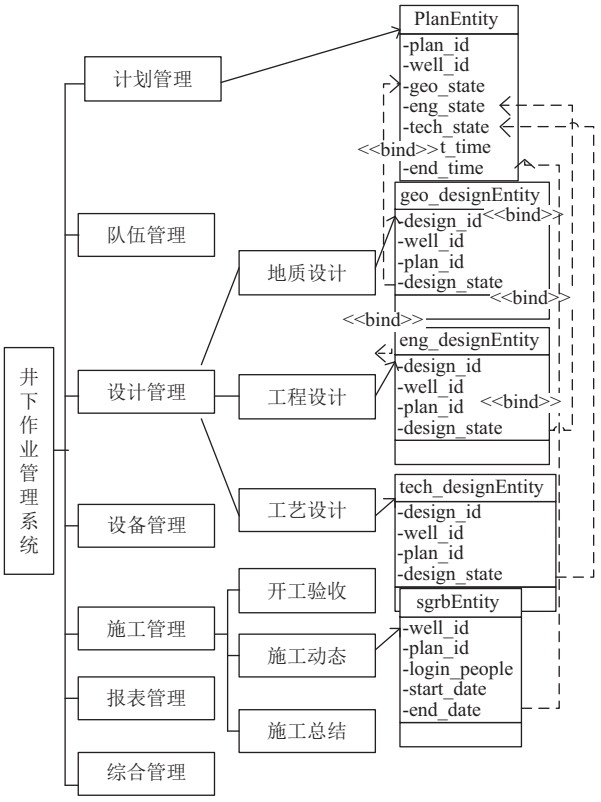

图 3 系统功能模块关系图

4.2 井下作业管理系统的设计管理模块

作业设计是井下作业过程中的重要部分. 在井下作业管理系统中, 设计管理模块主要包括设计数据的录入、设计的 word 文档的生成及审核过程三个流程.

由于井下作业管理系统涉及多个单位以及多种井型, 所以在设计模块会有多种设计类型, 不同设计类型在数据的录入、设计的 word 文档的生成及审核过程中会有不同, 需要设计模块具有很好的扩展性, 而且要与计划模块在数据更新上具有同步性.

在设计模块, 我们采用了 OSGI 扩展点机制[6,7]和 AOP 思想实现[8], 如图 4 所示.

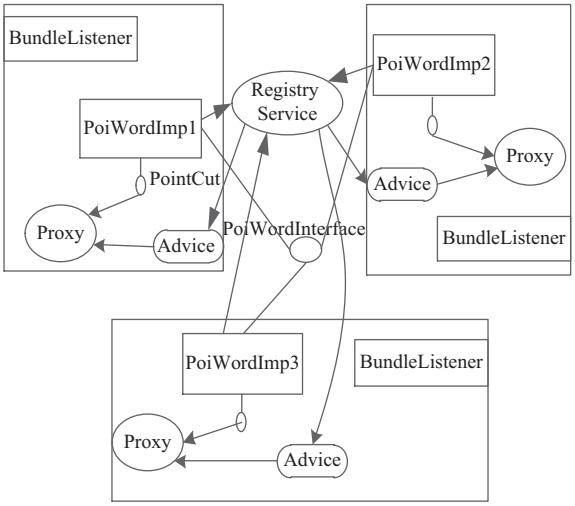

图 4 设计模块扩展与 AOP 同步更新示意图

以生成 word 组件为例. PoiWordInterface 作为扩展点的接口类, 并定义扩展契约, 其他的扩展插件按照扩展点约定的契约实现不同扩展, 同时扩展点注册表会对所有的扩展进行管理, 并且会负责监听扩展插件的生命周期, 加强扩展点扩展的稳定性, 在 PoiWordInterface 所在插件调用不同的扩展实现, 实现不同类型的设计. 每当生成了设计文档, 就说明设计状态变为设计中, 设计的状态要与计划显示的数据同步. AOP 是实现横切关注点的工具, 通过定义切入点和通知实现, 将 PoiWordImp 组件中的生成 word 文件方法定义为切入点, 将从服务层调用的修改计划实体类属性值的服务定义为后置通知, 在生成 word 文档同时修改计划模块数据, 实现了设计模块与计划模块数据更新的同步进行.

图 4 还描述了设计数据录入与审核流程的扩展实现及同步更新.

OSGI 扩展点机制和 AOP 思想的结合既实现了逻辑片段的模块化也降低了插件之间的耦合性, 某一扩展插件的启动与卸载不会影响到其他扩展插件的运行, 提高了系统稳定性.

5 系统使用及测试情况

我们在井下作业管理系统的开发中, 采用了 OSGI 框架与分层解耦思想结合的方法, 取代了基于 SSM 框架的架构. 目前, 该系统已在油田企业推广使用. 系统的开发采用 Java 语言, 开发工具是 Eclipse 3.7+ JDK1.6, 系统框架使用 OSGI Eclipse Equinox.

我们分别对采用 2 种方法开发的井下作业系统进行了测试, 对比了系统的启动时间、系统中生成 word 文档的响应时间, 如图 5、图 6 所示. 可以看出, 基于本文的方法开发的井下作业管理系统在启动时间及响应时间上都有很大提高. 这是因为, OSGI 采用的不是 Java 传统的树状类加载架构, 而是网状的类加载架构, 基于这种模型, 插件既可以加载自己的类也可以直接委派其他类加载器加载本插件中用到的其他插件中的类, 这是一个并行运行的过程.

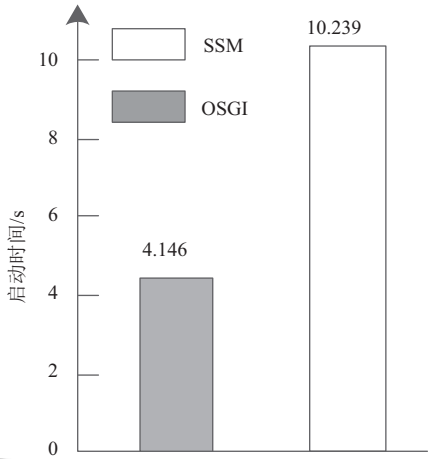

图 5 启动时间比较

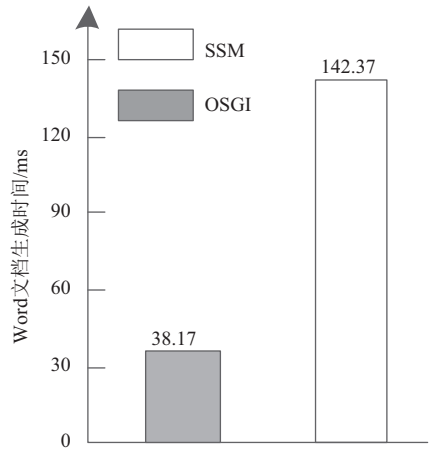

图 6 响应时间比较

OSGI 是一个模块化的框架, 模块化可以使系统的代码在一定范围内实现自治. 而且 OSGI 的面向服务组件模型要求程序员规范化代码的书写, 去掉程序中的一些冗余代码, 提高组件的复用能力, 使系统达到了瘦身的效果, 减少了代码量. 对两个系统工程设计模块的代码量对代码量进行比较, 结果如图 7 所示, 在功能基本相同的情况下, 在 OSGI 规范下可以有效减少代码量, 提高开发效率.

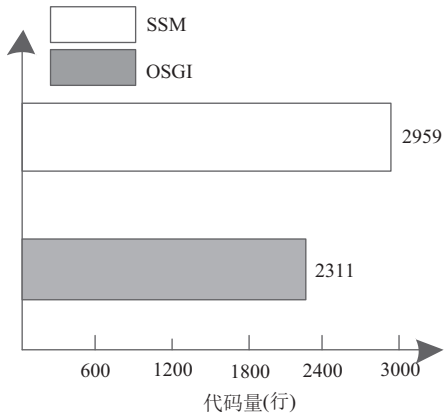

图 7 代码量比较

另外, 对于要求 7*24 小时不间断运行的井下作业管理系统来说, 提高系统稳定性十分重要. 在 OSGI 模型下, 系统的每个插件都具有生命周期, 各个模块的启动、停止、动态更新都可以实现管理, 因而可以做到动态地增加或禁止某项功能、更新某个模块, 可以在不中断运行的情况下自动更新升级, 这无疑提高了系统稳定性.

6 结语

OSGI 是一个松耦合、面向服务的应用程序开发框架. 我们在 OSGI 框架基础上, 结合分层解耦思想, 对井下作业管理系统进行总体设计与开发. 不仅降低了各个模块之间的耦合度, 还改善了系统的开发效率, 提升了系统性能. 由于不需要重新部署整个系统, 从而极大地改善了系统维护和部署的难度.

参考文献

1 Knoernschild K. Java 应用架构设计: 模块化模式与 OSGI. 张卫滨, 译. 北京: 机械工业出版社, 2013.
2 Mohammed M, Elish M, Qusef A. Empirical insight into the context of design patterns: Modularity analysis. Proc. of 7th International Conference on Computer Science and Information Technology (CSTT). Amman, Jordan. 2016.
3 Li WB, Zhang YB, Jin J. Research of the service design approach based on SCA_OSGi. Proc. of IITA International Conference on Services Science, Management and Engineering. Zhangjiajie, China. 2009. 392–395.
4 Hall RS, Pauls K, McCulloch S, 等. OSGI 实战. 郭庆, 李楠, 李守超, 等译. 北京: 人民邮电出版社, 2013.
5 杨扬, 侯红, 郝克刚. 基于容器的 IOC 控制反转模式的研究. 计算机应用与软件, 2009, 26(4): 17–19.
6 陈江浩, 余卫东, 雷晓阳, 等. 基于 OSGI 的跨平台测井软件设计及应用. 测井技术, 2014, 38(5): 587–591.
7 朱珠. OSGI 框架下 REST 架构风格的数据中心环境监测系统的分析与设计[硕士学位论文]. 北京: 北京交通大学, 2010.
8 Alexandersson R, Öhman P, Ivarsson M. Aspect oriented software implemented node level fault tolerance. Proc. of the 9th IASTED International Conference on Software Engineering and Applications. Phoenix AZ, USA. 2007. 57–74.