文章编号：1005-8451（2008）01-0030-03

插件技术在铁路站场组态软件的应用与研究

刘伟杰[1]，高 辉[2]，高利民[3]
（1.河南工业大学 信息科学与工程学院，郑州 450001；2.郑州铁路职业技术学院 信息工程系，郑州 450000；3.中国铁道科学研究院 基础设施检测中心，北京 100081）

摘 要：铁路站场组态软件是联锁系统中获取站场信息的重要组成部分，是生成站场静态数据的提供者。随着铁路的发展，铁路的每次大提速都需要对站场的描述进行较大的改动，为了解决这个问题，通过使用插件技术来设计站场组态软件，使每次对于站场的改动减少对软件的影响，只需要对改动的部分以插件的方式进行添加和修改即可。

关键词：组态软件；计算机联锁；设计模式；插件
中图分类号：TP391 文献标识码：A

Application and resarch of plug-in technology on railway in configuration software of stations and yards

LIU Wei-jie[1], GAO Hui[2], GAO Li-min[3]
(1. Institute of Information Science and Engineering, Henan University of Technology, Zhengzhou 450001, China;
2.Department of Information Engineering, Zhengzhou Railwy Polytechnic College, Zhengzhou 450001, China;
3.Infrastructure Inspection Research Institute, China Academy of Railways Sciences, Beijing 100081, China)

Abstract: The railway configuration software was important componet of access to information on station and yards, provider of generating static data. Along with the development of railway, when the railway speeded up, it was needed to change staion desription largely. To solve this porblem,it was designed stations configuration software by plug-in technology, influenced the software lessen, needed to added and revise for changing the part by plug-in

Key words: configuration software; computer interlock; design mode; plug-in

随着软件工程和基于面向对象设计思想的发展，解决软件开发设计中由于软件规模的不断加大以及软件开发成本所带来的问题，在不同的时期提出了不同的软件解决办法。插件系统是在面向对象思想发展的基础上，基于面向对象中的封装性、复用性、以及模块的独立性等特点，插件技术成为当前软件发展的一个主要方向。插件系统不同于组件系统，组件的出现简化了开发的工作量，插件系统和组件系统在封装一定的业务方面具有相同的功能，插件系统也具有封装性。但作为一个插件系统来说，实现一个通用的插件可以在更大粒度上进行复用，插件是比组件更加高层的一种模块封装方式。因此，本文研究了铁路站场的特点和联锁系统对站场图形的要求，对软件的分析从体系结构出发，采用了插件系统的体系结构，实现了联锁系统中对站场图形界面和数据自动生成的要求。使系统在编译发布之后，进行某个功能的扩充时，即使修改了代码，也不需要重新编译发布。适应了由于铁路提速带来的联锁系统中站场组态软件的改动的问题。

收稿日期：2007-08-13
作者简介：刘伟杰，助教；高 辉，助教。

1 站场组成结构

铁路车站的信号设备组成主要有道岔、信号机、区段、轨道电路、交叉渡线、复式交分、超限、变更按钮、半自动、机务段和场间联系等。联锁系统要根据站场信号设备的组成来生成对站场中设备的控制。根据联锁系统对站场图形生成的要求，组态软件的结构如图1。

2 站场组态软件设计

2.1 主体框架设计

主框架的设计的重点在于如何能找到新的插件

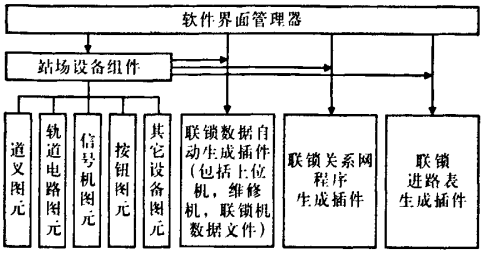

图1　站场组态软件功能结构图

以及如何调用它们。所以主程序框架和插件之间的通信关系是基于插件体系开发的重点内容之一，这种通信方式的建立条件是：(1) 必须有一个契约，应用程序要声明有哪些功能是可以被插件使用的，并且具备什么条件才能成为应用程序的插件；(2) 应用程序不依赖于插件，也就是说，没有其中的任何一个插件，应用程序也可以很好地运行；(3) 应用程序必须有一种策略来获取插件存在的位置，只有获得插件存在的位置，才有可能调用插件实现的功能；(4) 应用程序可以通过某种方式动态的加载插件。因此，主程序规定了一些协议，而这个协议就是应用程序和插件之间进行交互的依据和凭证。应用程序必须声明有什么样的功能可被插件使用，并且插件必须符合什么条件才能被使用。反之，插件必须要知道应用程序提供什么样的功能，才能将自己的功能融入到应用程序的体系中。主体框架结构如图2。

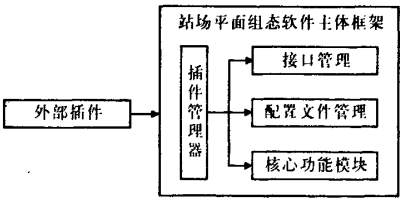

图2　主体框架结构图

2.2　插件的建立

插件是具有能够插入到系统中并实现具体功能的模块部分并通过接口和主体程序进行通信，因此，插件包括了功能实现代码和能够被系统识别插件的配置文件。由于在设计系统框架的过程中并不知道具体的插件的功能和结构，所以，插件的设计通过使用软件设计模式中的Builder模式和抽象工厂模式来建立。

Builder设计模式是将一个复杂对象的构建与它的表示分离，使得同样的构建过程可以创建不同的表示。Builder模式是一步一步创建一个复杂的对象，它允许用户可以只通过指定复杂对象的类型和内容就可以构建它们。Builder模式的结构如图3。

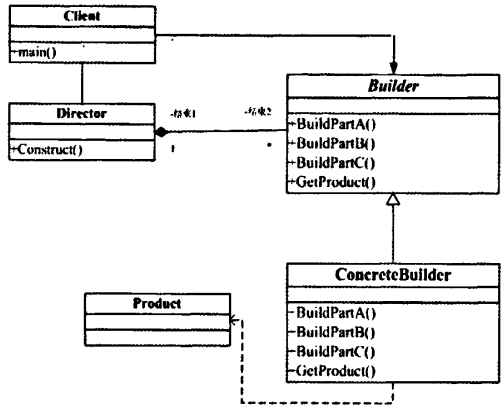

图3　Builder模式结构图

通过使用Builder设计模式，可以使一个插件具有几个不同的功能。

抽象工厂模式提供一个创建一系列相关和相互依赖对象的接口，而无需制定他们具体的类。创建一个工厂的接口，而不关心具体的工厂，这样一种设计思路就是抽象工厂模式。抽象工厂模式可以让我们在程序中通过使用不同的具体工厂来配置抽象工厂接口而替换一个新的产品系列。也就是说，把工厂也抽象，不再是具体的类，而是一个工厂的接口，就变成了抽象工厂。在站场组态软件设计的过程中，通过使用抽象工厂接口可以得到具体插件的一个实例化对象来调用插件的具体功能。站场组态软件中抽象工厂的结构如图4。

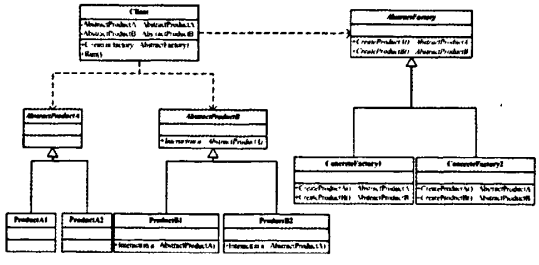

图4　抽象工厂模式结构图

一个插件包括了它的功能实现部分和配置文件的信息，配置文件是一个XML的文档，它描述了插件的具体信息，包括了插件的名字、提供者、版权、路径、运行插件所需要的依赖类库以及扩展点等一

系列的对插件的描述部分。在组态软件设计的过程中，为了实现对这些信息的提取，建造了一个类用来对这些信息的描述。

2.3 站场图形组态软件插件管理器的建立

系统框架和插件结构建立以后，如何将插件有效管理并加载到应用程序中是插件系统最终要完成的功能。插件管理器是通过插件的配置文件，读取有关插件的详细信息，通过将插件插入到插件树上来管理插件，因此，插件管理器设计的重点是如何从配置文件中解析出插件的信息，插件的依赖部分，扩展部分以及插件要插入的位置等，实现有效地管理。由于整个系统中插件管理器是唯一一个管理插件的部分，因此，对于插件管理器的设计采用单件模式进行设计。

这个插件树是在程序运行时动态创建的。对应于该插件树提供了一个XML结构的文件，树的所有路径（这些路径是逻辑路径）都定义在插件配置文件中。XML配置文件的结构说明了新增插件位置、类名、插件的属性等。

插件管理器主要提供了以下的方法：

(1) CreateAddInTree( )

这是一个静态方法，该方法首先判断插件树是否为空，即插件树是否已经创建，如果没有创建，则实例化一个默认的插件树对象。然后创建一个内部的文件服务实例，内部的文件服务的作用是管理加在插件树上的插件列表。循环判断插件列表，若在插件列表中还有未被初始化过的插件，则实例化一个新的插件；否则退出循环。将该新的插件针对一个插件文件进行初始化；调用插件树实例的InsertAddIn方法将刚初始化好的插件插入到树中。

(2) SetAddInDirectories( )和GetAddInDirectories( )

这两个方法分别为获得XML文件中插件的路径和在插件树的XML文件中要插入的路径。

(3) InsertAddIns( )

这个方法实现了插件插入到系统中。该方法首先对于插件列表文件进行访问，对于在插件列表中的每个插件生成一个实例对象，并把它插入到插件树中。

2.4 站场图形组态软件插件系统构造过程交互过程

站场图形组态软件系统构造过程交互如图5。

从上面的交互图可以看出，插件系统创建时系统做了以下事情：(1) 系统调用AddInSingleton这个类的一个静态方法CreateAddInTree()；(2) 在

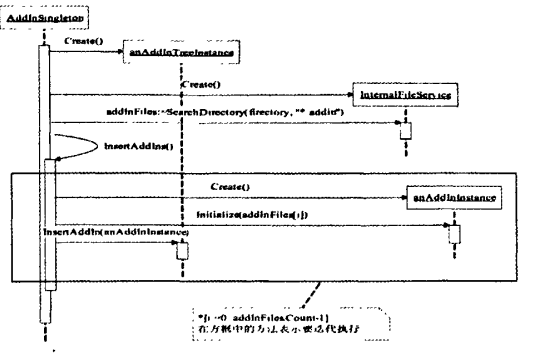

图5 插件系统构造过程交互图

AddInSingleton方法内部创建一个AddIn-Tree的实例；(3) 创建一个内部的文件服务实例InternalFileService；(4)调用InternalFileService的SearchDirectory方法，得到一个关于插件文件的列表；(5) 若在插件列表中还有未被初始化过的插件，则实例化一个新的插件；否则退出循环将该新的插件针对一个插件文件进行初始化；(6) 调用插件树实例的InsertAddIn方法将刚初始化好的插件插入到树中；(7) 循环5，6步骤。

3 结束语

基于插件结构的站场组态软件的设计解决了由于铁路提速造成的联锁系统站场描述改动的问题，在本论文课题中使用了插件技术的体系框架对整个软件部分进行了架构。在设计上使用了设计模式中的单件模式、Builder模式、工厂模式和command模式。在系统设计中，采用了松耦合的方法，使程序的扩展更加容易，用户不需要了解太多的内部知识，只要根据系统提供的接口就可以增加系统的功能。

参考文献：

[1] 刘 真. 软件体系结构[M]. 北京：中国电力出版社，2004，9.

[2] Erich Gamma. 设计模式：可复用面向对象软件基础[M]. 李英军，马晓星，蔡 敏，刘建中. 北京：机械工业出版社，2000，9.

[3] Alan Shalloway James R.Trott . 设计模式解析[M]. (第2版) 徐言声. 北京：人民邮电出版社，2006，7.

[4] 徐宏兴. 插件体系结构软件方法研究[D]. 四川大学硕士论文，2005，5.