2013 年 3 月 第 3 期
Vol. 34 No. 3 2013

一种动态软件体系结构下的代码生成方法

范 玥[1,2],王淑玲[1]

[1](中国科学院 沈阳计算技术研究所 研究生部,沈阳 110168)
[2](中国科学院大学,北京 100049)
E-mail:fanyue@ sict. ac. cn

摘 要:代码生成技术是一种根据用户的输入自动产生代码的一种开发方式,具有规范代码,提高效率,降低代码错误率等优点,但目前针对动态系统的代码生成方法探索较少.本文通过分析动态扩展系统和基于模板的代码生成形式,提出了一种将其二者进行整合的动态自适应开发框架下的代码生成方法,并给出了在实现过程中即插即用、即删即无和热部署等关键问题的解决办法.最后通过实验验证了本方法的正确性和可行性,具有适用于复杂功能的动态系统开发过程的特点.

关 键 词:代码生成;动态系统;模板;开发框架
中图分类号:TP311 **文献标识码**:A **文 章 编 号**:1000-1220(2013)03-0515-05

A Code Generation Method Based on Dynamic Software Architecture

FAN Yue[1,2], WANG Shu-ling[1]

[1] (Shenyang Institute of Communicating Technology, Chinese Academy of Sciences, Shenyang 110168, China)
[2] (University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: Code generation is the technique of building and using programs to write other programs. The central idea is to create consistent and high quality code automatically according to what users input. The features of code generation include regulating code, productivity, reducing the rate of coding error, etc. However, there is insufficient research done on the technique of dynamic code generation. Therefore, the paper analyzes dynamic extendable system and the forms of code generation based on template, and then it puts forward a new idea that we can integrate the two techniques above. And the paper also presents the solutions to the key problems such as plug-and-play, hot deployment during implementation process. At last the correctness and feasibility of this method is proved by experiment, and the method applies to dynamic development process of complex function very well.

Key words: code generation; dynamic system; template; development framework

1 引 言

目前多数软件开发人员在进行系统代码编写时仍然采用手工编写代码的方式,这种方式在一定程度上虽然能够灵活控制开发流程,但是也存在其固有的弊端,比如在进行大型 ERP 软件开发时存在大量的重复劳动以及难于控制代码质量等问题.在此时需要一种方式来解决这些开发过程中所面临的问题,代码生成技术则是解决此类问题的一种思路.代码生成技术是一种自动生成程序源码及相关资源的自动化技术.相对于传统代码编写方式,自动代码生成技术具有更多的优势,能够让开发人员更专注于上层架构的研发.

软件系统的体系结构是指该系统由哪些构件组成,这些构件如何协作构成一个系统,以及装配的模式和约束等[1].在文献[2]中把软件体系结构在运行时的演化分为静态软件体系结构和动态软件体系结构.目前多数的研究都是基于静态软件体系结构,对于需求不断更新的系统来说,静态软件体系结构显然不能满足需求.本文从动态软件体系结构和代码生成技术两个研究点出发,介绍一种动态自适应框架下的代码生成技术.

2 开发模型分析

2.1 软件体系结构现状及分析

目前很多学者提出了软件体系结构相关的概念和定义,下面介绍几个具有代表性的定义[3].

(1) Garlan&Shaw 模型:

SA = {components, connectors, constrains}.

(2) Perry&Wolf 模型:

SA = {elements, form, rational}.

(3) CFRP 模型:

SA = {elements, interfaces, connections, connection semantics}.

(4) Vestal 模型:

SA = {component, idioms/styles, common patterns of interaction}.

(5) IEEE 610.12-1990 软件工程标准词汇中的定义:

Architecture = {component, connector, environment, principle}.

(6) Boehm 模型:

收稿日期:2012-11-09 收修改稿日期:2012-12-12 作者简介:范 玥,女,1981 年生,硕士研究生,工程师,研究方向为计算机应用;王淑玲,女,1965 年生,研究员,研究方向为软件工程.

SA = {components, connections, constraints, stakeholders' needs, rationale}.

以上各种模型从多角度来描述软件体系结构,对软件体系结构的构件及其之间的关系进行了定义,另外模型对限制因素以及连接件的拓扑结构等也进行了相应的描述. 文献[3]认为软件体系结构中的这些要素应满足相应的限制,遵循一定的设计规则,能够在相应的环境下进行演化. 在文献[4]、文献[5]和文献[6]中分别介绍了目前国内外对 SA 动态演化的执行工具. 这其中包括基于反射原理、基于构件操作、基于反演工作流、基于 Pi 演算、利用一个外部的体系结构演化管理器等具体执行方式.

目前动态软件体系结构的研究的逐步深入,基于动态软件体系结构的自适应软件体系结构,正交软件体系结构等正成为动态软件体系结构研究的一个方向[4]. 本文则是以 OSGI 的研究为基础进行的.

2.2 OSGI 框架结构

OSGI 规范由 OSGI 联盟制定,目的是建立一个基于 Java 的动态模型系统[7]. OSGI 的组成结构如图 1 所示.

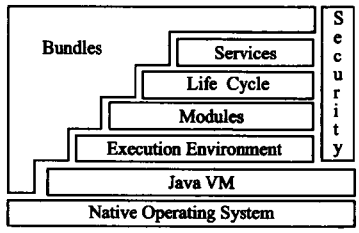

图 1 OGSI 结构图

Fig. 1 Structural diagram of OSGI

Bundles:提供一定功能或服务的代码和文件,这些功能可以通过代码共享或者对象共享的方式提供出来,这些组件通常由开发人员给出.

Services:由各个 Bundle 提供希望被别的 Bundle 所使用的功能,通过对象共享方式进行注册.

Life Cycle:对每一组件的安装、启动、运行、停止和卸载这些生命周期活动的管理.

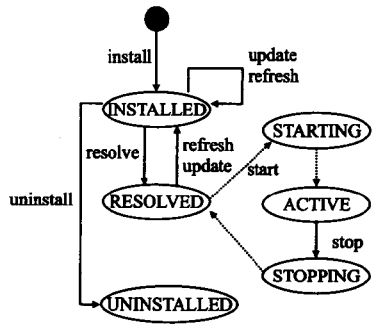

图 2 Bundle 的生命周期

Fig. 2 Bundle life cycle

Modules:定义了 Bundle 如何导入和导出.

Execution Environment:定义系统中可用的类和方法.

基于 OSGI 的应用,就是由一个个 Bundle 组成的. 每一个 Bundle 代表了一个独立的功能模块,通过 OSGI 系统把这些模块组织在一起,就形成了一个完整的系统. 同时通过不断开发新的 Bundle 来添加新功能,从而满足新的需求. 通过代码调用或者服务调用来重用已经开发 Bundle 的功能[8]. Bundle 的生命周期如图 2 所示.

从图 2 中可以看出,Bundle 可以被动态的安装、启动、停止和卸载.

2.3 XSLT 简介

目前采用 XML 文件来记录元数据被广泛采用,由于 XML 具有的跨平台等特性,它正逐渐成为各系统间进行数据交换的首选. 因此,本文采用 XML 文档来描述基本信息.

XSLT 主要是为 XML 语法之间的转换提供帮助,将模型转换成代码并使得规则与模型相分离. 由于 XML 的平台无关特性使得基于 XML 的代码生成技术可以不受应用平台的限制. 而 XSLT 语言只定义文档转换规则,与生成代码采用的语言无关,故可以生成任意编程语言的代码. 代码生成原理如图 3 所示.

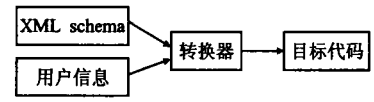

图 3 代码生成流程

Fig. 3 Code generation principle

通过整合文档建模工具,以及外部部署环境. 系统结构图如图 4 所示.

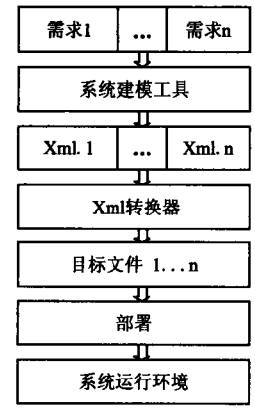

图 4 代码生成系统结构图

Fig. 4 Code generation system structure diagram

如图 4 所示,系统通过建模工具来完成对元数据信息的配置,进而把生成的 XML 文件传入到 XSLT 转换器,通过 XSLT 转换器完成对代码及其相关资源的生成,然后系统把代码传递给部署系统生成程序文件,通过配置工具的相关微调试,最后部署到运行环境中.

3 设计与实现

3.1 动态实现过程

动态软件体系结构由于处在动态的环境之中,必须动态调整自身的行为以处理新型的信息对象,必须伸缩自身的处

理能力以适应不同的用户群体. 而要满足这样的要求就必须有灵活的体系结构,以动态地增减其组成构件[9].

由于动态体系结构在使用中不可避免地要对组件进行改变升级,这种升级通常表现在两个方面:其一是对组件内部进行更新,这主要是因为组件自身可能出现算法效率过低等情况. 其二是组件增加了新的功能,需要对外提供新增的服务,在文献[5]中把此类更新称为扩展性更新. 这两类更新在系统运行时会导致系统体系结构发生变化,从而影响系统的一致性,为了保证系统各个功能组件能协同工作而不会彼此冲突或导致功能不可用,动态软件体系结构必须给出相应的解决方案. 在文献[5]中,对功能组件的新建、更新和删除这几种情况分别予以了分析.

本文的动态化研究基于 OSGI 框架提供的动态化服务,但是 OSGI 框架在实现过程中也存在着某些问题. 下面通过对 OSGI 框架实现的剖析,提出解决这些问题的实现方法.

3.1.1　模块动态添加

对于动态添加的理解是希望系统在生成新的 Bundle 后,此 Bundle 能够即刻正常使用. 下面从 package 依赖和服务依赖两个角度来进行阐述. 目前我们定义了四个 Bundle,分别是 XDA、XDB、XDC 及 XDD. 其中 XDA 对外提供了 DemoBusinessService 服务实现,此实现依赖于 XDB 对外 export 的 test.xosgi.domain 的 package 以及 XDD 对外 export 的 test.xosgi.storeService 的 package,实现同时还需要调用 StoreService,此实现由 XDC Bundle 对外提供,因此 XDC Bundle 也依赖于 XDD Bundle 对外 export 的 StoreService 接口所在的 package,同时增加了一个对当前运行时间的输出. 首先安装 XDA,此时由于 XDA 缺少所依赖的 package 以及 OSGI 服务,它对外提供的 OSGI 服务自然也无法激活,因此控制台输出未找到要调用的服务. 接着分别安装并启动 XDB 以及 XDC 和 XDD,然后调用 XDA 则相关服务能够正常启动,系统运行如图 5 所示.

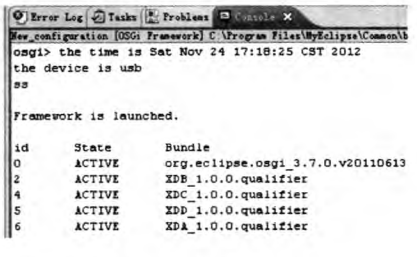

图 5　动态添加

Fig. 5　Dynamically add

由此可见,使用 OSGI 框架实现即插即用是可行的. 但是从实验结果的分析中,发现了 Bundle 装载并不是动态关联的. 因此,在通过 XSLT 生成 Bundle 文件时,必须在模版信息中添加对 Bundle 依赖关系的描述.

3.1.2　模块动态删除

对于系统中 Bundle 动态删除的期待是在停止 Bundle 后,其所对外 export 的 package 以及 OSGI Service 都不再可用. 依然采用前文的例子进行测试. 在停止 XDB Bundle 后,根据观察系统的输出并不会影响 XDA.

通过 uninstall XDB 然后进行 refresh 调用,系统提示未找到调用的服务. 由此可以看出,OSGI 框架对动态删除的支持还是能够满足系统的动态化要求的.

3.1.3　模块热部署

系统对热部署的期待是直接覆盖更新的 Bundle,所有新增的调用请求自动执行新的功能模块,并且保留更新的对象的状态. 对此,我们依然使用本文之前的 Bundle. 将 XDB 中的实现做相应的更改,更改之后直接进行文件覆盖,此时系统提示失败,原因是有进程在使用当前文件,通过 uninstall 然后执行 refresh 动作,继续覆盖 XDB 文件,然后重新安装 Bundle,启动和更新之后,此时系统提示成功. 由此可见,目前对于 OSGI 框架的实现以 Equinox 为例,要达到热部署的期待效果还是有一定的距离. 为了解决此类问题,系统在需要热部署某个 Bundle 时候,需要重新进行生成和安装.

OSGI 动态框架通过动态管理 Bundles 来提供对系统的动态变化的支持,虽然在诸多方面 OSGI 并不能完全称之为动态软件体系结构,但是作为一个新兴的动态管理框架,还是具有一定的研究价值. 由于 OSGI 在构建 Bundles 存在一定的弊端,为了让 OSGI 更加灵活且符合当前软件开发的要求,本文结合 OSGI 与 XSLT 来提出一种生成代码的解决方法,以此来提高开发过程的灵活性.

3.2　整合 OSGI 与 XSLT

目前,采用 XSLT 进行代码自动生成技术已较为成熟,但是生成的代码及其相关资源的部署和组件的动态支持则相对缺乏,因此使用 OSGI 的动态管理机制能够有效解决此类问题,使得自动生成的代码动态加入软件体系结构中,完成基于需求不断变化的支持. 由于在 OSGI 中,功能组件都是以 Bundle 的形式发布的,一个 Bundle 由类和其他资源组成,它可以为其他的 Bundle 提供服务,也可以导入其他 Bundle 中的资源包. 因此 XSLT 产出的目标代码即为 Bundle,经过整合 OSGI 编码规则后的代码生成流程如图 6 所示.

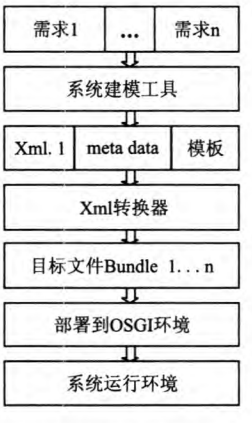

图 6　设计框架

Fig. 6　Design framework

为了提高系统扩展的能力,使本系统能够根据功能变化的需要进行相应的调整. 可以把图 6 中的功能组件封装成 OSGI 中的 Bundle,让其本身运行于 OSGI 环境中,则能够完成运行系统功能的动态扩展,这样代码生成模块自身也可以进行动态扩展,达到动态适应需求变化的能力要求. 整合之后的结构如下页图 7 所示.

如图7所示,基于XSLT的代码生成系统生成新的功能Bundle,新的功能Bundle部署于OSGI运行环境,此时系统增加了新的功能模块.在OSGI规范中,模块可以认为是由一个或多个Bundle组成,Bundle通过export-package的方式为其他Bundle提供了访问的途径,这样模块间即能够进行互访也能够进行有效的隔离.

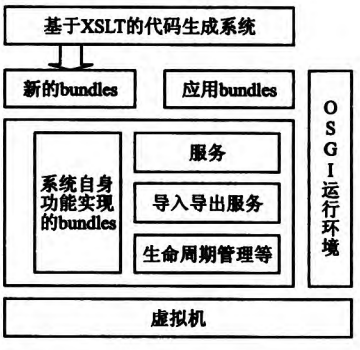

图7 系统组成

Fig. 7 System components

本系统的核心分为两个方面,首先功能组件能够根据需求动态生成,其次就是动态化方面的表现.目前在动态化方面通常体现在即插即用、热部署以及即删即无三个方面.这三点主要涉及到引用对象的迁移以及对象状态的保留,对于本系统,引用对象的迁移涉及为package依赖及OSGI服务依赖的迁移,而在对象状态的保留上,OSGI目前不提供支持.目前流行的OSGI框架实现系统对以上提到的三方面都提供了相应的实现,如Equinox实现.

4 实 验

实验硬件环境为Intel双核2.6GHz的CPU,2G内存.采用Windows XP操作系统,开发环境为Eclipse 3.5,数据库采用MySql 5.0版本,XSLT转换器使用Xalan,OSGI实现框架采用Equinox实现.

由于实际软件项目大多有复杂的功能需求和大量的数据交互处理,涉及到众多需要处理的内容,但是实际上无非是前台用户调用和后台数据处理两个部分.为了验证本文提出的方法,在此搭建一个实验平台.实验平台完成两部分内容,一部分为数据存储,另一部分为系统交互界面.由于目前的软件体系结构中系统数据处理部分相对固定,并不需要频繁的大量更新和创建.而用户交互部分随着业务的不断变更,必然有大量的创建编码工作,当前的自动代码生成技术大多使用在此方面的创建上.因此本实验的界面交互部分采用XSLT技术自动生成,然后动态部署到OSGI环境中,完成验证过程.

实验首先定义实体部分,实体为一个POJO类:USER,为其封装到实体功能模块TBDomain中,此Bundle通过export对外提供服务.并在数据库中为此实体创建对应的同名表.接着定义数据访问部分模块TBStore,此Bundle模块完成数据的存取功能,此Bundle需要依赖TBDomain对外export的test.tbo.tbdomain,同时此Bundle需要对外export test.tbo.tbstore提供数据存取服务.至此,平台的数据存取服务搭建已经结束.在测试过程中,由于系统采用的是MySql数据,需要添加外部驱动,而OSGI的类加载机制要求所有的外部架包必须通过MANIFEST.MF文件进行声明,所以必须修改MANIFEST.MF文件导入MySql驱动.经过测试,系统能够正常运行,数据已经也能够正常添加到数据库中,系统运行结果如图8所示.

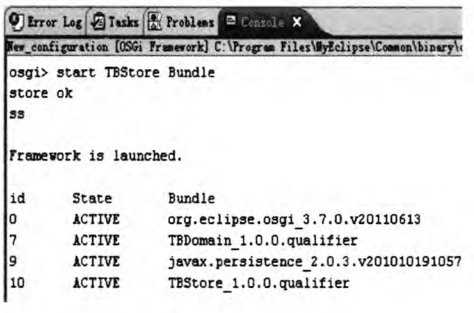

图8 基础模块

Fig. 8 The base module

接着定义系统的交互Bundle,此部分采用XSLT进行自动生成.因此首先定义模版部分以及元数据描述.为了能够清楚的描述生成过程,此部分依然采用USER的界面操作,并只提供有代表意义的创建页面.USER的添加界面只需要定义两个基本数据即name和age.首先设计模版部分的XML文件UI.xml,文件中定义界面的基本布局部分.通过Xalan对此文件的解析生成相应的源码文件,此过程可以规范代码的结构,避免一些常见的错误.此Bundle需要依赖之前定义的test.tbo.tbdomain和test.tbo.tbstore.系统运行结果如图9所示.

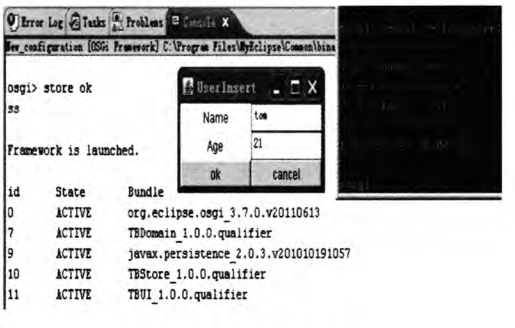

图9 系统运行结果

Fig. 9 Running results

通过实验表明XSLT与OSGI的整合是完全可行的,通过对实验结果的扩展能够大量动态生成需要频繁更新的组件部分,达到规范化,动态化的系统开发.

5 结 论

本文提出了一种新颖的代码构建方式.通过整合OSGI与XSLT使得代码的规范性得到了保证,同时也节省了大量的开发时间.同时通过使用OSGI与XSLT的整合,能够在一定程度上实现软件系统的动态扩展.通过实验分析,它适合需要大量具体功能的软件系统的开发过程,比如软件平台的交互部分等.

References:

[1] Shaw M, DeLine D, Klein D, et al. Abstractions for software architecture and tools to support them[J]. IEEE Transactions on Software Engineering, 1995, 21(4):314-335.

[2] Oquendo F. π-ADL: an architecture description language based on the higher-order typed π-calculus for specifying dynamic and mobile software architectures[J]. ACM Software Engineering Notes, 2004, 29(4): 1-13.

[3] Sun Chang-ai, Jin Mao-zhong, Liu Chao. Overviews on software architecture research[J]. Journal of Software, 2002, 13(7): 1230-1231.

[4] Li Qiong, Jiang Ying. Overview on research of dynamic software architecture[J]. Application Research of Computers, 2009, 26(6): 2354-2355.

[5] Yu Zhen-hua, Cai Yuan-li, Xu Hai-ping. On modeling approach for dynamic software architecture[J]. Journal of Xi'an Jiaotong University, 2007, 41(2): 168-171.

[6] Oreizy P, Medvidovic N, Taylor R N. Architecture-based runtime software evolution[C]. Proc of ICSE'20, Washington DC: IEEE Press, 1998:177-186.

[7] The OSGI architecture[EB/OL]. http://www.osgi.org/Technology/WhatIsOSGI/, 2012.

[8] Lu Qin, Wu Yong-ming. An approach of self-adaptive software design based on supervisory control[J]. Computer Applications and Software, 2004, 21(1): 18-19.

[9] Ma Xiao-xing, Zhang Xiao-lei, Lv Jian. Description and implementation of dynamic software architectures: a reflective approach[J]. Journal of Nanjing University, 2004, 40(2): 148-149.

附中文参考文献:

[3] 孙昌爱,金茂忠,刘　超. 软件体系结构研究综述[J]. 软件学报, 2002, 13(7):1230-1231.

[4] 李　琼,姜　瑛. 动态软件体系结构研究综述[J]. 计算机应用研究, 2009, 26(6):2354-2355.

[5] 于振华,蔡远利,徐海平. 动态软件体系结构建模方法研究[J]. 西安交通大学学报, 2007, 41(2):168-171.

[8] 陆　勤,吴永明. 一种基于 SC 的自适应软件设计方法[J]. 计算机应用与软件, 2004, 21(1): 18-19.

[9] 马晓星,张小蕾,吕　建. 自省的动态软件体系结构描述与实现[J]. 南京大学学报, 2004, 40(2): 148-149.

一种动态软件体系结构下的代码生成方法

作者： 范玥， 王淑玲， FAN Yue， WANG Shu-ling
作者单位： 范玥,FAN Yue(中国科学院沈阳计算技术研究所研究生部,沈阳110168;中国科学院大学,北京100049)，
王淑玲,WANG Shu-ling(中国科学院沈阳计算技术研究所研究生部,沈阳,110168)
刊名： 小型微型计算机系统 ISTIC PKU
英文刊名： Journal of Chinese Computer Systems
年，卷(期)： 2013,34(3)

参考文献(14条)

1. Shaw M;DeLine D;Klein D Abstractions for software architecture and tools to support them[外文期刊] 1995(04)

2. Oquendo F π-ADL:an architecture description language based on the higher-order typed π-calculus for specifying dynamic and mobile software architectures 2004(04)

3. Sun Chang-ai;Jin Mao-zhong;Liu Chao Overviews on software architecture research[期刊论文]-Journal of Software 2002(07)

4. Li Qiong;Jiang Ying Overview on research of dynamic software architecture[期刊论文]-Application Research of Computers 2009(06)

5. Yu Zhen-hua;Cai Yuan-li;Xu Hai-ping On modeling approach for dynamic software architecture[期刊论文]-Journal of Xi'an Jiaotong University 2007(02)

6. Oreizy P;Medvidovic N;Taylor R N Architecture-based runtime software evolution 1998

7. The OSGI architecture 2012

8. Lu Qin;Wu Yong-ming An approach of self-adaptive software design based on supervisory control[期刊论文]-Computer Applications and Software 2004(01)

9. Ma Xiao-xing;Zhang Xiao-lei;Lv Jian Description and implementation of dynamic software architectures:a reflective approach[期刊论文]-Journal of Nanjing University 2004(02)

10. 孙昌爱;金茂忠;刘超 软件体系结构研究综述[期刊论文]-软件学报 2002(07)

11. 李琼;姜瑛 动态软件体系结构研究综述[期刊论文]-计算机应用研究 2009(06)

12. 于振华;蔡远利;徐海平 动态软件体系结构建模方法研究[期刊论文]-西安交通大学学报 2007(02)

13. 陆勤;吴永明 一种基于SC的自适应软件设计方法[期刊论文]-计算机应用与软件 2004(01)

14. 马晓星;张小蕾;吕建 自省的动态软件体系结构描述与实现[期刊论文]-南京大学学报 2004(02)