25卷 第10期
2008年10月

微电子学与计算机
MICROELECTRONICS & COMPUTER

Vol.25 No.10
October 2008

upDSL:一种描述动态更新策略的领域特定语言

张迎春,黄林鹏
(上海交通大学 计算机科学与工程系,上海 200240)

摘 要:定义了一种刻画软件动态更新策略的领域特定语言upDSL.该领域特定语言可以对OSGi平台中模块动态更新策略进行结构化描述,并可提供动态更新关键步骤的必要信息,从而实现动态更新的可控性和安全性,并为不同的动态更新机制实现提供良好的描述和推理基础.

关键词:领域特定语言;动态更新策略;更新策略管理模块;OSGi
中图分类号:TP31 **文献标识码**:A **文章编号**:1000-7180(2008)10-0034-03

A upDSL for Describing Dynamic-update Policy

ZHANG Ying-chun, HUANG Lin-peng
(Department of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai 200240, China)

Abstract: This paper defines a domain specific language upDSL to describe dynamic update policies. This domain specific language describes modules' dynamic update in OSGi platform in a structural way, and provides key steps of dynamic update with necessary information, thereby promotes controllability and safety of dynamic update, and provides a policy basis for different dynamic updating mechanism.

Key words: domain specific language; dynamic update policies; update policies management module; OSGi

1 引言

动态更新是指,对运行中的程序(代码和数据),在不中断其运行的前提下进行更新.虽然目前对动态更新的研究已经很深入,但仍缺乏一种对更新策略的描述和刻画方式.动态更新策略可以为动态更新过程中的关键步骤(更新时机选择,更新顺序确定,状态转化等)提供必要的信息,实现动态更新的策略和机制分离,并且提供从系统层面上整体把握动态更新正确性的依据.文中定义了一种描述OSGi[1]平台中模块动态更新策略的领域特定语言(Domain Specific Language)——upDSL,并简要介绍了基于upDSL开发的动态更新策略管理模块.

2 动态更新策略定义

动态更新的目标是在不中断程序运行的前提下,完成对程序代码和数据的更新.动态更新策略是指可以实现这一目标的方案集合,其内容应包含:动态更新的对象;对象各个组成部分的更新策略;当动态更新遇到问题时,可以采取的措施;多个对象间可能存在的更新顺序约束;其他约束.

根据模块化和信息隐藏原则[2],更新对象的策略应包括其逻辑子层次的更新策略的描述,并且只需描述每个层次中的接口部分.例如:动态更新的对象如果是Java中的一个包(package),那么对于此包中包含的各个公共类,及其包含的各个公有成员,都应该指明它们的更新策略.不用对非公共类和私有成员的动态更新策略进行描述.

3 upDSL介绍

upDSL是用于描述OSGi平台下软件动态更新策略的领域特定语言.图1展示了该语言及其解释器与OSGi中其它模块的关系.

OSGi平台是动态的面向服务、基于构件的Java

收稿日期:2008-07-18
基金项目:国家自然科学基金项目(60673116)

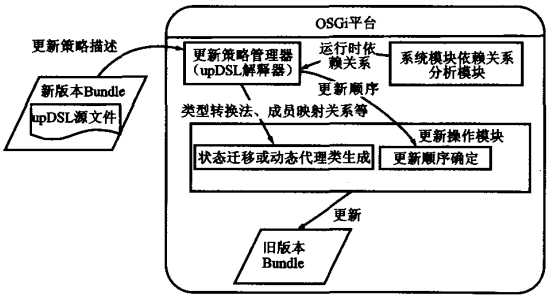

图1　upDSL及其解释器

框架. Bundle是该平台中唯一的部署单位[1]. 忽略不相关的细节，Bundle实际上是一系列Java包的集合，其中导出包(Export-Package)构成了Bundle的对外接口. Bundle间通过导入和导出包、注册和使用服务来进行交互，并由此产生依赖关系.

upDSL对OSGi平台中模块动态更新策略的描述分为如下几个层次：(1) 更新对象是Bundle；(2) 该Bundle导出包的更新策略；(3) 对于每个导出包，其公共类的更新策略；(4) 对于每个公共类，其公共成员的更新策略.

upDSL的源文件由新版本Bundle的开发者编写，并附加在新版本的Bundle中. 一个upDSL源文件就是对一个Bundle的动态更新策略描述的集合. 使用upDSL对动态更新策略进行描述的优点在于：(1) 可以用简洁的方式结构化的描述动态更新策略；(2) 动态更新所涉及的对象和操作易于描述(相比其它通用语言，领域特定语言的描述更贴近问题域)；(3) 使用upDSL刻画的信息可以方便的在动态更新过程中使用(与软件运行平台结合紧密).

4　upDSL语法规则定义

图2中是upDSL语法规则定义. 其中，成员声明(Member Declaration)、成员函数声明(Function declaration)、类型转换函数定义(Function Definition)以及包引用(Import Declarations)的语法与Java语法[3]相同.

upDSL的程序头部分的upgrade语句包含了：待更新的Bundle名、老版本号、新版本号以及更新顺序约束.

包的更新操作描述分为update和unexport两类. update操作是指定包的公有类更新操作的集合；unexport操作指明该包在升级后不再被导出.

类的更新操作描述包括：替换(replace)，增加(add)和删除(delete). 替换操作是指定类公共成员更新操作的集合，with语句用于声明该类升级后的新名字；增加操作指明了在升级后新增的公共成员；删除操作指明了升级后被删除的公有成员.

成员的更新操作描述包括：修改(alter)，增加(add)和删除(delete). 修改操作描述了新旧版本成员间的映射关系，对于成员类型兼容或者函数的升级，仅需指定前后成员的映射(用"->"操作符表示)，对于新旧版本成员类型不兼容(新版本成员类型不是旧版本类型的子类)的情况，需要定义这两个类型间的转换函数，以此保证动态更新的可行性[4]，upDSL的using…where语句用于定义这些类型转换函数；增加操作指明了升级后新增的公共成员；删除操作指明了升级后被删除的公有成员.

Natural Number	n	
Null Expression	ε	
Bundle Name	BN	
Package Name	PN	
Class Name	CN	
Function Definition	F	
Function Name	FN	
Function Declaration	FD	
Import Declaration	ID	
MemberDeclaration(Primitive type)	MD_PT	
MemberDeclaration(Complex type)	MD_CT	
Function Definitions	Fs	::= F | Fs Fs
Bundle Version Number	VN	::= n | VN.VN
Programs	P	::= upgrade BN_1 from VN_1 to VN_2 after BN_2 ID P-Actions | upgrade BN_1 from VN_1 to VN_2: ID P-Actions
Package Update actions	P_Actions	::= update PN_1 {C-Actions} | unexport PN_1 | P-Actions P-Actions
Class Update Actions	C_Actions	::= C-Replace | C-Add | C-Delete | C-Actions C-Actions
Class Replace Action	C_Replace	::= replace CN_1 with CN_2 {M-Actions}
Class Add Action	C_Add	::= add CN
Class Delete Action	C_Delete	::= delete CN
Member Declarations	MD	::= MD-PT | MD-CT
Member Update Actions	M_Actions	::= M-Alter | M-Add | M-Delete | M-Actions M-Actions
Member Alter Action	M_Alter	::= alter M-Mappings
Member Add Action	M_Add	::= add MD | add FD
Member Delete Action	M_Delete	::= delete MD | delete FD
Member Mappings	M_Mappings	::= $MD\text{-}PT_1 \rightarrow MD\text{-}PT_2$ | $FD_1 \rightarrow FD_2$; | MD-PT→MD-CT using FN FuncBindings | MD-CT→MD-PT using FN FuncBindings | $MD\text{-}CT_1 \rightarrow MD\text{-}CT_2$ using FN FuncBindings
Function Bindings	FuncBindings	::= where {Fs} | ε

图2　upDSL语法规则定义

4.1　upDSL示例程序

```
upgrade testBundle_A from 1.0 to 1.1 after testBundle_B:
update testPackage_A {
  replace testClass_A with testClass_A1 {
    alter Student s -> Teacher t using studentToTeacher
where {
      public Teacher studentToTeacher (Student s) {
        Teacher t = new Teacher();
        t.name = s.name; t.CV = sCVTotCV(s.CV);
        return t; }
      private tCV sCVTotCV(sCV) { //…}
  } }
```

```
replace testClass _ B with testClass _ B1 {
add double j; delete int k; }}
```

上面的示例中，testBundle _ A 的开发人员定义了此 Bundle 从 1.0 版本升级到 1.1 版本的更新策略.并且规定此更新必须在 testBundle _ B 更新完成之后进行.值得注意的是下面一些对 testPackage _ A 的升级操作：

(1) testClass _ A. Student s 对应升级到 testClass _ A1. Teacher t，这种升级是复杂类型(用户自定义类)间的不兼容转换，因此需要由程序员在其后的 where 块中定义转换函数(studentToTeacher).如果需要转换的类型中还包含其它复杂类型的成员，那么在 where 块中必须提供相应的转换函数.此示例中，sCVTotCV 函数用于将复杂类型 sCV 转换到 tCV 类型.

(2) 将 testClass _ B 中的 int k 删除，并向其中添加一个成员 double j.这两个操作和 int k -> double j 不等价，虽然后者也可以看作是删除一个 int 类型，并相应的增加一个 double 类型替代.但是后者的语义中包含了 k 和 j 之间存在映射关系，而在前一种方式中，k 和 j 之间并不存在任何关系.

5 基于 upDSL 的 OSGi 动态更新策略管理模块

基于 upDSL 语法规则定义，在 OSGi 中开发了动态更新策略管理模块(参见图 1).该模块解析 Bundle 中的 upDSL 源文件，生成相应的更新策略内部表示，并集中管理这些更新策略.

(1) 更新操作到来时如果特定模块上存在被“挂起”的更新，动态更新策略管理模块就需要根据 upDSL 对更新操作进行汇总.例如，在对 Bundle _ A 进行从 1.1 到 1.2 的更新时，如果 Bundle _ A 上有一个从 1.0 版本到 1.1 版本的更新操作被挂起.并且从 1.0 版本升级到 1.1 版本时，删除了某个公共元素；但在 1.1 版本到 1.2 版本的升级时，该元素又重新恢复.此时可以根据两次更新的 upDSL 文件进行分析，在最终进行 1.0 版本到 1.2 版本的升级中就不需要对该元素进行任何操作.

(2) 更新顺序的确定需要考虑 Bundle 间存在的依赖关系.一部分依赖关系可以通过分析系统运行时的情况获得，但这些信息并不足以确定更新顺序.为此 upDSL 中 after 语句描述了由开发者指定的模块更新顺序约束.结合这些约束以及系统运行时的依赖关系，动态更新策略管理模块即可确定动态更新的顺序.

(3) 为了保证动态更新的类型安全性，传统的动态更新对于更新操作有一些限制[5-6]，如新旧版本成员间的类型必须兼容等.但是在提供了新旧版本成员的类型转换函数的前提下，可以放宽对成员类型的约束，而不会破坏动态更新的类型安全性[4].upDSL 中 using…where 语句提供了相应的类型转换函数的定义.当动态更新遇到新旧版本成员类型不兼容的情况，如果 upDSL 提供了类型转换函数，那么此更新可以进行，否则动态更新管理模块将拒绝该更新操作，以保证动态更新的类型安全.

(4) 为了实现新旧版本成员间的状态迁移，动态更新管理模块通过 upDSL 中描述的新旧版本成员的映射关系，为状态迁移过程提供了必要的信息和依据[7].如果系统采用代理类的方式来解决动态更新问题[8].那么新旧版本成员间的映射关系可以辅助代理类的自动生成.

6 结束语

综上所述，对动态更新策略进行结构化的描述，可以为动态更新过程中的关键步骤提供必要的依据和信息，这为不同的更新机制的实现提供了良好的基础，也使得对动态更新的管理和验证工作可以方便的进行，提高了动态更新的可控性和安全性.

然而，目前 upDSL 是由模块的新版本开发者编写，其编写过程显得机械和繁杂，容易出错.因此，今后 upDSL 的源文件要能够通过对比新旧版本模块的功能描述自动生成.

另外，upDSL 对于分布式环境下的动态更新支持很有限.R-OSGi[9]可以将 OSGi 应用于分布式环境下，今后可以将 upDSL 扩充，使其能够与 R-OSGi 协作，从而实现对分布式环境下动态更新策略描述的支持.

参考文献：

[1] Alliance O S. OSGI service platform core specification Release 4[M]. French:[S.L], 2005.

[2] Ghezzi C, Jazayeri M, Mandrioli D. Fundamentals of software engineering[M]. Boston: Prentice Hall, 1991.

[3] Gosling J. The Java language specification[M]. Boston: Addison-Wesley Professional, 2000.

[4] Duggan D. Type-based hot swapping of running modules [J]. Acta Informatica, 2005, 41(4): 181-220.

(下转第 39 页)

$w(s,t)=1$)

(3) $M_{ot}(s)=\begin{cases}1 & \forall s\in S,\ {}^{\cdot}s=\varnothing\\ 0 & \forall s\in S,\ {}^{\cdot}s\neq\varnothing\end{cases}$

其中,$s=(T_1,T_2,(a_1,a_2))$ 的库所 s 为同步器,σ_p 为计划 p 中的变迁序列.

定义6　给定一个计划 p,$\Sigma_{ot}=(S_{ot},T_{ot};F_{ot},K_{ot},W_{ot},M_{ot})$ 为计划 p 对应的扩展的同步网系统.若 p 是正确的当且仅当 $\sum_{ot}$ 运行终止时到达的状态 M_{end} 满足:

$$M_{end}(s)=\begin{cases}1 & \forall s\in S_{ot},\ s^{\cdot}=\varnothing\\ 0 & \text{else}\end{cases}$$

定理1　定义6产生的计划库(PB)中的每个计划(plan)都是正确的.

证明:由于在任务有效分解的 Petri 网中,对于任意一个变迁 t,只存在两种情况:

(1) $|{}^{\cdot}t|=1\wedge|t^{\cdot}|=1$,这是对应"或"分解;

(2) $|{}^{\cdot}t|=1\wedge|t^{\cdot}|>1$,这是对应"与"分解.

因此在计划 p 对应的扩展的同步网系统中 $\forall t:|t^{\cdot}|=1$,每个变迁的发生都不会产生冗余的标识信息,因此当网系统运行终止时,只有后集为空的库所具有1个标志,其他库所均无标志.因此可以得出该命题成立.

3　结束语

文中从多主体系统的主体计划入手,分析了主体计划的生成方法,将 Petri 网分析技术和逻辑分析结合在一起,通过对网系统的可达性进行分析,并结合相关的同步网的理论,找出所有的可执行计划集合组成计划库,并对计划库中的每个计划的正确性进行了形式化的验证,从而保证在多主体系统的任务分解的可靠性,为以后研究多主体系统的任务动态分配机制打下基础.

参考文献:

[1] Fikes R, Nilsson N. STRIPS: a new approach to the application of theorem proving to problem solving[J]. Artificial Intelligence, 1971(2):189-208.

[2] Bui D, Jamroga W. Multi-agent planning with planning graph[C]// Proc. Of the 3rd International Workshop on Hybrid Methods for Adaptive Systems. Finland, Oulu, 2003:558-565.

[3] 方欢, 吴哲辉. 基于 Horn 子句集的 Pr/T 网可达树的方案求解[J]. 系统仿真学报,2005,10(s1):163-165.

[4] Castilho M. A petri net based representation for planning problems[C]//Proceedings of International Conference on Knowledge Based Computer System. India, Hyderabad, 2004.

[5] 方欢, 崔焕庆,王丽丽.任务分解的 Petri 网方法及有效性研究[J].安徽理工大学学报:自然科学版, 2008,28(1):85-89.

[6] 袁崇义. Petri 网原理与应用[M]. 北京:电子工业出版社, 2005:225-258.

[7] 吴哲辉. Petri 网导论[M].北京:机械工业出版社华章分社,2005.

作者简介:

方　欢　女,(1982-),硕士,讲师.研究方向为 Petri 网理论及应用、多主体系统、人工智能.

(上接第36页)

[5] Zhang S, Huang L P. Formalizing class dynamic software updating[C]// Proceedings of the Sixth International Conference on Quality Software. Shanghai, 2006: 403-409.

[6] Zhang S, Huang L P. Type-safe dynamic update transaction[C]// Computer Software and Applications Conference. Annual International, 2007.

[7] Hicks M. Dynamic software updating[D]. USA:University of Pennsylvania, 2001.

[8] Orso A, Rao A, Harrold M J. A technique for dynamic updating of Java software[J]. Software Maintenance, 2002: 649-658.

[9] Rellermeyer J S, Alonso G, Roscoe T. Building, deploying, and monitoring distributed applications with Eclipse and R-OSGI[J]. Proceedings of the 2007 OOPSLA workshop on eclipse technology eXchange, 2007: 50-54.

作者简介:

张迎春　男,(1984-),硕士.研究方向为分布式计算和程序设计语言.

upDSL:一种描述动态更新策略的领域特定语言

作者： 张迎春， 黄林鹏， ZHANG Ying-chun， HUANG Lin-peng
作者单位： 上海交通大学计算机科学与工程系,上海,200240
刊名： 微电子学与计算机 ISTIC PKU
英文刊名： MICROELECTRONICS & COMPUTER
年，卷(期)： 2008,25(10)

参考文献(9条)

1. Alliance O S OSGI service platform core specification Release 4 2005

2. Ghezzi C;Jazayeri M;Mandrioli D Fundamentals of software engineering 1991

3. Gosling J The Java language specification 2000

4. Dlaggan D Type-based hot swapping of running modules[外文期刊] 2005(04)

5. zhang S;Huang L P Formalizing class dynamic software updating 2006

6. Zhang S;Huang L P Type-safe dynamic update tratlsaction 2007

7. Hicks M Dynanmic software updating 2001

8. Orso A;Rao A;Harrold M J A technique for dynamic updating of Java software 2002

9. Rellermeyer J S;Alonso G;Roscoe T Building,deploying,and monitoring distributed applications with Eclipse and R-OSGI 2007