第 29 卷 第 4 期　　燕山大学学报　　Vol. 29 No. 4
2005 年 7 月　　Journal of Yanshan University　　July 2005

文章编号：1007-791X (2005) 04-0291-03

依赖注入技术及其执行过程的形式化描述

胡启敏[1,2]，薛锦云[1,2]，钟林辉[1]

(1. 江西师范大学 计算机信息工程学院，江西 南昌 330027；2. 中国科学院软件研究所 计算机科学重点实验室，北京 100080)

摘　要：依赖注入技术指由构件运行平台在运行期根据系统配置文件中定义的构件间的依赖关系，将被调用构件实例化，并注入到调用构件之中。本文在分析研究依赖注入技术的基础上，用完全格工具形式化地描述依赖注入的执行过程。

关键词：依赖注入；基于构件的软件开发；完全格；形式化方法

中图分类号：TP301　　**文献标识码：**A

0　引言

依赖注入技术指由构件运行平台在运行期根据系统配置文件中定义的构件间的依赖关系，将被调用构件实例化，并注入到调用构件之中[1]。如果软件需求发生变更，调用构件依赖的被调用构件需要被其他构件代替时，就可以运用依赖注入技术，在不修改调用构件的情况下，通过修改软件系统的配置文件，即可为调用构件重新指定被调用构件。

深入研究依赖注入技术对合理设计软件体系结构，提高软件系统的可复用性有着重要意义。

形式化方法是基于数学工具，描述、分析系统性质的技术，可以用来开发、验证软硬件系统。形式化方法可以用来消除系统中存在的二义性、不完整性和不一致性[2]。

当前，扩充 CSP 而形成的 Wright 语言，软件模型检测（Model Checking）等被广泛应用于描述、验证软件体系结构。然而，上述工具主要用于描述构件之间的具体功能关系，并不适合于描述依赖注入技术的构件注入过程。

为了精确地描述依赖注入的构件注入过程，文中借助完全格这一数学工具，对该过程进行了形式化地描述。

1　依赖注入技术介绍

1.1　构件之间产生依赖关系的描述

面向对象程序设计是设计、实现构件的主要方法之一，本文将基于面向对象程序解释构件间的依赖关系及相关解决方案。假设有 B、C、D（可以是 EJB 或 JavaBean）构件，B 为调用者，C 或 D 为被调用者，则 B 和 C 或 D 之间产生了依赖关系。

1.2　依赖注入技术的引入

依赖注入技术起源于轻量级构件运行平台提供的控制反转机制（Inversion of Control）[2]，即由平台定位插件的具体实现。依赖注入技术用部署描述文件描述构件之间的依赖关系，在运行时由平台按部署描述文件，动态地将被调用构件注入到调用构件之中。如下所示：

```
Public class B {
    private Ainterface Ai;
    public void setAi（Ainterface Ai）{
//由构件运行平台调用该方法，并根据配置文件注入具体的被调用构件
     this.  Ai = Ai;
    }
}
```

假设 B 构件需调用 C 构件，则用 XML 定义的配置文件可以如下所示：

收稿日期：2005-05-18　基金项目：国家自然科学基金（No. 60273092），国家 973 重大基础研究前期研究专项（No. 2003CCA02800）

作者简介：胡启敏（1979- ），男，博士生。主要研究方向为软件形式化与自动化、软件体系结构。

```
<beans>
  <bean id="B" class="B">
  //调用构件
  <property name="Ai">
    //用于指向被调用构件的接口
    <ref local ="C" />
    //指定的被调用构件名称
   </property>
  </bean>
  <bean id="C"  class="C"> </bean>
  <bean id="D"  class="D"> </bean>
  //可以被调用的构件
</beans>
```

系统运行时，将由构件运行平台实例化 C 构件，然后自动为 B 构件注入。如果软件需求变更，B 构件需调用 D 构件，则只需将配置文件中的<ref local ="C" />**项目，改为**<ref local ="D" />**。**

依赖注入技术有效提高了构件可复用程度。由于耦合度低，依赖注入也为构件的单独测试和组装测试提供了良好的支持，为强调测试先行的极限编程[1]提供了借鉴。

2 依赖注入执行过程的形式化描述模型

2.1 构件模型的建立

构件模型是面向构件的软件开发方法的核心，是构件的本质特征及构件间关系的抽象描述。定义构件模型如图 1 所示[3]。

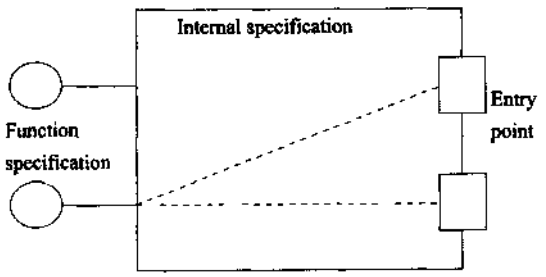

图 1 构件模型

Fig. 1 Component model

该构件模型主要分为外部接口和内部规约。外部接口包括：1）功能规约（function specification），构件供外部使用的接口；2）接入点（entry point），构件使用的外部接口。内部规约包括构件的语法约束、语义模型及其他[2]。

构件模型可以用三元组$Comp_A=(F_A, EP_A, IS_A)$表示，F_A对应构件的功能规约集合、EP_A对应接入点集合、IS_A对应内部规约。

若构件 A 依赖于构件 B，则有$EP_A \subseteq F_B$（即：构件 B 供外部使用的接口可以匹配构件 A 的接入点）若构件 A 依赖于多个构件：B…C，则有$EP_A \subseteq F_B \cup \cdots \cup F_C$。如果软件需求发生变化，构件 A 从依赖于构件 B 变为需依赖于构件 D，那么只要构件 D 也满足$EP_A \subseteq F_D$，则构件运行平台即可将 D 构件注入到 A 中。

2.2 依赖注入状态模型的建立

用三元组$S=(e, f, c)$表示依赖注入过程中的某个状态，其中e表示该状态下已经被实例化的构件集合，f表示处于激活状态的构件接口集合（当构件被实例化以后，该构件的接口处于激活状态），c表示未被实例化的构件集合。用 Typeof (k, K) 表示 k 为构件 K 实例化以后得到的一个实例。

用前后置断言表示状态转换函数 TF（Si）= Sj 如下：

Pre：$\exists K \in Si.\ c \wedge (EP_K=\varnothing \vee EP_K \subseteq Si.\ f)$

Post：$Sj.\ e=Si.\ e \cup \{k\} \wedge Sj.\ c=Si.\ c-\{K\} \wedge Sj.\ f=Si.\ f \cup F_K \wedge$ Typeof (k, K)

状态转换函数反映了依赖注入的执行机制：如果某构件不需要注入其他构件或已经注入所有依赖的构件，则该构件可以被实例化，同时该构件可以被注入到其他构件中。

2.3 基于完全格的依赖注入过程的形式化描述

以依赖注入过程中所有可能的状态为论域 A，在论域 A 上存在关系≤，对于 A 中任意两个状态 Si, Sj，若 $Si.\ e \subseteq Sj.\ e$，则有 Si≤Sj，由于≤满足自反、反对称、传递性，故序偶<A, ≤>为偏序集。特别地，如果状态转换函数将状态 Si 一步转换为状态 Sj，则没有其他状态 Z 可以满足 Si≤Z，Z≤Sj，所以有状态 Sj 覆盖状态 Si。

在论域 A 中存在两个特殊状态S_0与S_F，S_0为注入初始状态，S_F为注入结束状态。以图 2 所示依赖关系为例， A 依赖于 B、C，B 依赖于 D，易知 $S_0.\ e=\varnothing$，$S_0.\ f=\varnothing$，$S_0.\ c=\{A, B, C, D\}$；$S_F.\ e=$

{a, b, c, d} ∧Typeof (a, A) ∧Typeof (b, B) ∧ Typeof(c,C) ∧Typeof(d,D), S_F. f={F_A, F_B, F_C, F_D}, SF.c=∅。

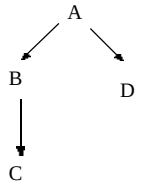

图 2　构件依赖关系

Fig. 2　Dependency of components

在偏序集<A, ≤>**中，任意两个元素** Si, Sj **都有最小上界** Sa, Sa.e=Si.e∪Sj.e**，和最大下界** Sb, Sb.e=Si.e∩Sj.e**，故偏序集**<A, ≤>**构成格。** A **中的初始状态**S_0**为格的全下界，结束状态**S_F**为格的全上界。同时易证：对于任意**L⊆A**有最大下界和最小上界，所以**<A, ≤>**可以构成完全格。**

以图 2 **所示依赖关系为例，它所对应的所有可能的依赖注入状态为**S_0.e= (∅、S_1.e= {d}、S_2.e= {c}、S_3.e= {b, d}、S_4.e= {c, d}、S_5.e= {b, c, d}、S_F.e= {a, b, c, d}**，这些状态可以构成完全格，该完全格对应的哈斯图如图** 3 **所示。**

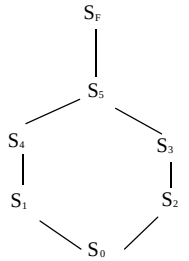

图 3　依赖注入状态的哈斯图

Fig. 3　Hasse diagram of states of injection

在完全格<A, ≤>**中，如果存在连接初始状态**S_0**与结束状态**S_F**的链**S_1, …, S_K**，则该链为依赖注入从起始至完成状态的一个可能的运行途径。由此易知：连接**S_0, S_F**的所有的链为依赖注入过程所有可能的执行途径。**

3　小结

通过依赖注入技术，构件运行平台只需读取简单的配置文件就可为调用构件注入被调用构件，完成构件组装过程。依赖注入技术大大提升了构件的可复用性，为实现动态、可灵活配置的软件系统提供了借鉴。

使用完全格工具，可以更加精确、完整地描述依赖注入的几种机制，包括：依赖注入如何应对软件需求的变化，依赖注入的起始、结束状态以及注入过程中可能经过的执行途径等。

在今后的研究中，将对构件运行平台建立形式化的描述模型，以便分析在不同的运行平台下，依赖注入表现出的新特性。

参考文献

[1] Ian Sommerville. Software Engineering (Seventh Edition) [M]. Pearson Education Limited, 2004.

[2] **梅宏，陈峰，冯耀东，等**. ABC**：基于体系结构、面向构件的软件开发方法** [J]. **软件学报**, 2003,14 (4): 721-732.

[3]Martin Fowler.Inversion of Control Containers and the Dependency Injection Pattern [EB/OL], http: //martinfowler.com/articles/injection.html.

Dependency injection and its execution's formal description

HU Qi-min[1, 2], XUE Jin-yun[1, 2], ZHONG Lin-hui[1]

(1. College of Computer Information Engineering, Jiangxi Normal University, Nanchang, Jiangxi 330027, China; 2. Laboratory of Computer Science, Institute of Software Chinese Academy of Sciences, Beijing 100080, China)

Abstract: Dependency injection means that component platforms create called components and inject them to their caller according to their dependency defined in file. Based on analyzing and studying dependency injection, a formal description for it's execution with a kind of mathematical tool named complete lattice is given in this paper.

Key words: dependency injection; component-based software development; complete lattice; formal methods