第25卷 第3期 计 算 机 工 程 1999年3月
Vol.25 № 3 Computer Engineering March 1999

·软件技术研究与实现·

中图分类号 TP311.5

构件规范匹配方法的探讨与设计

张升昕 任 杰 钱乐秋
(复旦大学计算机系 上海 200433)

摘要 介绍了对构件规范的研究，首先研究了逻辑子类的形式化模型，然后以此为基础，设计了构件规范匹配方法，开发了构件规范匹配的自动工具matchable，同时开发了构件生成器generator，它对匹配后的构件实现自动组装。

关键词 构架 构件 规范匹配 逻辑子类

The Research and Design of Component Specifications Matching

Zhang Shengxin Ren Jie Qian Leqiu
(Department of Computer Science, Fudan University Shanghai 200433)

【Abstract】This paper introduces our research in component specifications matching. We, at first, focuse on the formal model of logic subtype, and then, on the base of this model, design the method of component specifications matching. We also present two automated tools we developed: matchable for matching component specifications and generator for packaging components matched.

【Key words】Architecture; Component; Specifications matching; Logic subtype

软件重用是解决软件危机的十分重要的对策之一，然而在软件系统开发中却迟迟不能真正有效地实施它，其原因是软件开发人员往往熟悉传统的开发工具(如编程工具、文档管理工具、需求分析工具等)，而缺乏可以验证的构件重用方法，也缺乏有效的构件重用工具。构件规范匹配是构件重用过程中的步骤之一，我们尝试对它建立形式化的模型，并开发相应的工具。

1 方法特点

构件规范匹配方法具有以下特点：

(1) 形式化模型 由于构件规范匹配方法以形式化模型逻辑子类为基础，因此有助于保证方法的正确性和完备性；

(2) 自动规范分析 构件规范分析是十分烦琐的工作，我们开发的规范分析(Matchable)提供了规范自动匹配的手段，尽量减少了软件设计人员的参与，从而保证了规范匹配的可靠性；

(3)自动构件组装 开发的构件生成器(Generator)能自动地对匹配的构件进行组装，生成最终可重用的构件，基本上实现了构件匹配的全过程的自动化。

2 基本概念和定义

下面描述构件规范匹配方法的有关概念和定义。

虽然软件构架这个术语已经出现很久，对它的研究目前十分流行，但软件工程界对它并没有给出唯一的定义，不同的角度有不同的解释。一般而言，构架是指组成系统的构件的组合结构和构件之间的连接方式。在本文中，强调的是构件(客户和服务者)之间的连接，相关的概念有接口和绑定等。我们用CDL(构架描述语言)描述系统构架规范。下面介绍系统构架模型的主要组成部分：

(1)构件(Component) 组成系统的独立的功能单位，它可以是结构模式中的模块、面向对象模式中的类或者就是一组函数的集合等。构件分为原子构件(Atomic Component)和组合构件(Composite Component)两类，组合构件由原子构件组合而成。构件是重用的基本单位。

(2)客户(Client)和服务者(Server) 根据构件之间的实现关系可以将构件分为客户构件和服务者构件。客户构件通过调用服务者提供的服务实现自己的全部功能，而服务者要将自己实现的功能提供给客户。一个客户也可以是另一个客户的服务者，同样一个服务者可以是另一个服务者的客户。

(3)接口(Interface) 每个构件都有与其相连的构件接口，接口描述该构件的功能集合，接口分为'提供'(Provides)接口和'要求'(Requires)接口两种，一个构件可以有多个'提供'接口和'要求'接口。'提供'接口描述一个构件向其它构件提供的服务或功能实现，'要求'接口描述一个构件为了实现自己的功能必须从其它构件得到的服务。'要求'接口对于构件绑定十分重要，客户构件不必和某个特定的服务者固定地链接在二进制代码中，只要一个构件能够提供'要求'接口中指定的服务，它就有可能成为服务者。

(4)绑定(Binding) 如果一个构件能够满足另一个构件的要求，或者说两个构件能够构成客户／服务者的关系，就可以将它们连接起来以实现客户的功能，这种连接通过绑定来实现。绑定将客户的'要求'接口和服务者的'提供'接口清楚地连接在一起，并形成一个组合构件。

```
component Client
begin
  requires requiredFunction;
end;
component Server
begin
  provides providedFunction;
end;
```

*张升昕 男，26岁，研究生，主攻软件工程

收稿日期：1998—05—18

```
component Binding
begin
    Client clientInstance;
    Server serverInstance;
    bind clientInstance. requiredFunction
        to serverInstance.providedFunction;
end;
```

图 1 构件绑定

(5)配置(Configuration)特性 描述构件实现关于开发环境(编程语言、对象模型等)和运行环境(平台、中间件等)的依赖。在把客户和服务者封装为可执行的二进制代码时必须要用到这些配置特性。对于必需的配置特性，可以缺省。每个构件可以对应多种配置特性组合。在图2的例子中，构件Client的配置特性PASClient指出它是用Delphi语言编写的、运行在Windows 95上。

```
Configuration PASClient of Client
Begin
  Property OS Win95;
  Property language Delphi;
End;
```

图2 配置规范

以上描述了构件规范匹配方法所基于的系统构架模型，图3则描述了规范匹配在构件重用全过程中的地位：

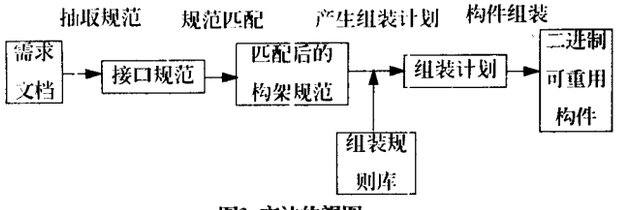

图3 方法的视图

3 构件规范匹配方法

构件规范匹配方法，就是对构件规范的分析过程，以判断客户的要求能否被服务者的服务去实现。规范分析是一个十分繁琐而又必需的工作，应尽量采取形式化的方法，减少设计人员的参与。我们专门开发了规范分析器(Matchable)来自动匹配构件规范。下面先介绍逻辑子类的概念系统，它是构件规范匹配方法的基础。

逻辑子类：描述两个接口规范之间的实现关系。如果一个构件能实现规范S_j就能实现规范S_k，则称S_j是S_k的逻辑子类。那么，实现S_j的构件就能透明地代替用来实现S_k的原构件。逻辑子类分为结构子类和行为子类两种，结构子类是行为子类的基础。首先给出结构子类(记为$\leqslant_s$)的递归定义：

·基本结构子类包括有关基本数据类型的子类关系：Short$\leqslant_s$ Long，Float$\leqslant_s$ Double；

·结构化数据类型(A、B)的子类关系定义为：

如果对于结构化数据类型B中的任意类型为T的字段a，A中存在一个字段名也为a，且类型为T的结构子类的字段，则A$\leqslant_s$ B；

·操作(f、g)的子类关系定义为：

如果f的所有输入参数都是g的'相应'输入参数的超类，并且g的所有输出参数和返回类型都是f的'相应'输出参数和返回类型的超类，则f$\leqslant_s$ g；在这里，'相应'是指参数在参数表中具有相同的参数名，或在相同的位置。

·接口规范(A、B)的子类关系定义为：

如果对于接口规范B中的任意类型为T的操作a，A中存在一个操作名也为a，且类型为T的子类的操作，则A$\leqslant_s$ B；

```
Interface InsertRecordA
begin
    States{UnAuthorized(init);Authorized};
    Void Authorized(in string user,in string password)
        Postcondition(Authorized);
    Boolean InsertRecord(in string record_name,in double
value,in long date)
        Precondition(Authorized);
end;
Interface InsertRecordB
begin
    Boolean InsertRecord(in string password,in string record_
name,in float value, in short date);
end;
```

图4 逻辑子类

在图4中，接口InsertRecordA是接口InsertRecordB的结构子类，InsertRecordB中的操作InsertRecord是接口InsertRecordA的操作InsertRecord的子类。

结构子类定义只根据结构来判断，特别是要求同名匹配会引起规范分析的困难和错误。行为子类定义从语义角度来弥补结构子类定义的不足。下面给出行为子类的定义：

·操作(f、g)的子类关系定义为：

如果操作g的前置条件(Precondition)蕴涵操作f的前置条件，并且操作f的后置状态(Postcondition)并上操作g的前置条件蕴涵操作g的后置状态，则操作f是操作g的子类；

·接口(A、B)的子类关系定义为：

如果接口A的不变部分等价于接口B的不变部分，并且接口B的协议是接口A的协议的子集，即，接口B所能接受的操作序列包括接口A所能接受的操作序列，则接口A是接口B的子类。

结构子类定义和行为子类定义的一个本质区别是行为子类关系是个不可判定的关系，因为证明操作条件之间的逻辑蕴涵关系并没有可行的方法，好在实际的规范分析方法中只需要大致的判断就可以了。根据结构子类定义，采用自动推理技术来实现Matchable。

Matchable对构件规范分析以后，就要对它们实施绑定。对于完全符合结构子类关系的理想情况，可完全自动化绑定；但在实际工作中，往往有例外的情形：尽管构成子类关系，但接口规范并非完全相同或者根本就不构成子类关系，对于前者只能进行手工绑定，对于后者Matchable可以提供帮助实施半自动绑定，无论手工绑定还是半自动绑定都需要用到一个转换器来协助。转换器提供客户和服务者之间接口的转换，它的接口和客户的'要求'接口相同，而用服务者的'提供'接口来实现。转换器分别处理以下两种情况：

·构件规范构成子类关系，但客户和服务者的接口并非完全相同，它们仍然不能直接通信，可以用转换器连接；

·构件规范根本就不构成子类关系，产生一个转换器模板，它包含客户可以被服务者提供的服务，而将未被服务者提供的服务留给设计者去填写。

4 工具Matchable的实现

以上面讨论的结构子类模型为基础，使用自动推理技术实现接口规范匹配工具Matchable，它的算法如下：

```
Main:  输入：接口规范A、B
       输出：匹配成功与否；如果失败，列出匹配的操作
       Matchable=true;
       MatchSet=null;//MatchSet是匹配的操作集合;
       for all 操作Oper：T in B do
       begin
            if not 存在操作Oper：T' in A then Matchable=
                                        false
                 else if operation(T',t)=false
                           then Matchable=false
                           else 将(T',T)加入MatchSet;
       ebd;
       if Matchable then return 匹配成功
            else return MatchSet;
operation：输入：操作f、g的参数列表和返回类型
       输出：f是否g的子类
       for all 输入参数InArg：T in f do
       begin
            找到g中对应的输入参数InArg：T';
            if data(T',T)=false then return false;
       end;
       for all输出参数OutArg：T in g do
       begin
            找到f中对应的输出参数OutArg：T';
            if data(T',T)=false then return false;
       end;
       return data(f返回类型，g返回类型);
data：输入：数据类型A、B
       输出：A是否B的子类
       if A=B then return true;
       if A=short and B=long then return true;
       if A=float and B=double then return true;
       for all 字段field：T in B do
       begin
          if not存在字段field:T ' in A then return false;
          if data(T',T)=false then return false;
       end;
       return true;
```

5 构件生成

一旦分析了客户和服务者的接口规范，就要将匹配的构件绑定，再按统一的接口标准将绑定的构件组装为可执行的二进制代码，成为可重用构件，这个转化过程就是构件生成。

构件生成过程要依据构件的配置信息，首先产生一个组装计划，然后按组装计划实施组装。为此专门开发了构件生成器(Generator)来产生组装计划，它是一个基于知识的专家系统，该系统包含一个规则库，规则库中存储两类规则，第一类规则定义各种生成工具(规范描述语言处理器、编译器、链接器等)的功能和限制；第二类规则定义用户的偏好，当有多个组装计划可供选择时，专家系统凭借此偏好作出决定。规则库可以根据需要(比如新的生成工具的出现)扩充。Generator分析客户和服务者的配置信息，决定接口之间的互连应选择哪些配置选项，然后进行必要的调整，最后产生组装命令将客户和服务者相连。作为专家系统Generator的推理基础，目标运行环境的互连能力和各种编程语言同运行环境的匹配都必须被形式化地描述，然后作为规则加入规则库中。组装计划以命令文件的形式给出，执行命令文件就完成了组装过程。

6 结论

我们以逻辑子类模型为基础，研究了构件规范匹配方法，并开发了规范分析器Matchable以及构件生成器Generator。该方法具有如下优点：形式化模型、自动分析、自动组装。随着分布式系统的发展和Internet的普及，我们下一步的工作是让该方法在Internet上得到进一步的实现。

参考文献

1 Batory D、Geraci B J. Validating Component Composition in Software System Generators. In Proceedings of the Fourth International Conference on Software Reuse. IEEE Computer Society Press，1996-04

2 Garlan D,Perry D E. Introduction to the Special Issue on Software Architecture. IEEE Transactions on Software Engineering，1995，21(4)：269-274

3 Callahan J,Purtilo J. A Packaging System for Heterogeneous Execution Environments. IEEE Transactions on Software Engineering，1991，17(6)：626

Sun等公司推出Merced架构的Solaris软件

Sun公司与富士通、NCR、西门子和东芝等著名公司合作，日前推出了采用Merced架构的Intel IA-64 Pre-silicon软件开发环境下的Solaris操作系统。

Solaris向Merced的初步移植具有对现有的Solaris Intel IA-32应用软件的向后兼容性。Sun公司计划在1999年上半年，以Merced架构的Solaris早期版本的完全64位开发环境，支持一些独立软件开发商和Intel系统合作伙伴。Sun还准备对Solaris操作环境进行优化，以便全面具有Merced的特性和能力。

在IA-64 Pre-silicon软件环境下运作Solaris，对于Solaris IA-64操作系统的发展是一个重要的里程碑，这是Merced架构背后隐藏着的对整个业界都有着巨大发展潜力的标志，Intel公司表示愿意与Sun公司合作，将这一工作向前推进。

这个Solaris操作环境对目前在Merced架构下实用的3200多个Intel应用软件给予完全的兼容。这就意味着，如果独立软件开发商采用的是不需要64位地址的或不需要完全的Merced性能的应用软件，则他们就不必改变或重新编制其现有的32位Solaris环境下的Intel应用软件，也能享用Merced架构的优势。（新）