

构件可信性验证研究

胡燕京

(武警工程学院 电子技术系, 陕西 西安 710086)

摘 要: 基于构件“开发与应用”两者相分离的特点, 提出了一种面向用户的构件服务承诺机制. 为创建该机制, 从构件提供者角度, 定义了面向方法序列的构件服务, 给出了构件服务遵循的完备性质, 并提出一种符合完备性质的构件服务产生方法. 这种产生方法给出一种清晰的构件服务形式化描述即提供构件服务的“承诺”机制, 并且这种描述是第三方可验证的. 相关性质的证明以及示例分析验证了该机制的可行性.

关键词: 可信构件; 构件服务; 可信性; 服务约束

中图分类号: TP311

文献标识码: A

文章编号: 1000-7180(2011)05-0087-06

Research on Verification Trustworthiness of Software Component

HU Yan-jing

(Department of Electronics Engineering, College of the APF, Xi'an 710086, China)

Abstract: Based on the characteristics of “Development and Application’s separation”, a user-oriented component services commitment mechanism is proposed. To create the mechanism, this paper defines the approach sequence oriented component services from the component provider’s point of view, gives the comprehensive nature of component services, proposed a method for generating the component services which meet the comprehensive nature. The generating method gives a clear formal description of component services, means provide the component services commitment mechanism, and this description is a third-party verifiable. Proof of the related natures and cases study verifies the feasibility of the mechanism.

Key words: trusted software component; component service; trustworthiness; service constraint

1 引言

随着构件化软件工程的飞速发展, 人们在软件开发实践中越来越重视软构件的质量. 软构件是面向已有问题或对未来问题的预期, 能够作为求解广泛一类问题的软件系统的基本组成部分^[1], 旨在通过软构件的复用组装提高软件开发效率. 因此, 可复用构件技术使得人们对克服“软件危机”并促进软件工业的发展充满期望. 但是, 几十年的实践证明: 工业领域应用成熟的标准件组装式的产品开发模式不能直接套用或移植到软件开发领域^[2]. 主要原因是: 相对于工业领域已经成熟的一系列度量质量指标(如可靠性、性能等), 在软件领域关于软件构件的度

量尚没有形成较成熟的一致标准, 而且缺乏一种质量评估与验证的方法. 并且, 由于构件设计与软件开发是两个相对独立的过程, 这种“分离开发模式”使得开发可信的构件化软件面临困难.

构件本身的可信性程度将直接影响到构件化软件的可信性程度. 因此, 有关构件可信性的研究近十年来广受关注. 1998 年在澳大利亚 Monash 大学召开的“TOOLS PACIFIC 1998”会议上成立了第一个可信构件研究小组, 正式将可信构件研究引入到软件工程领域. Bertrand Meyer 等^[3]总结了构件可信性研究所采用的一些具体方法: 从技术角度包括契约设计、形式化验证(如正确性的数学证明)、程序语言(如面向对象技术)、可复用构件库设计、充分测试

收稿日期: 2010-07-28; 修回日期: 2010-09-30

基金项目: 国家自然科学基金(60842006); 陕西省自然科学基金项目课题; 武警工程学院基础基金研究项目

等;从管理角度有 CMM 等系统过程模型;从社会角度主要是用户对于构件开发者的信任评估等. 在深入分析可信构件实现方式及面临的挑战的基础上, Bertrand Meyer^[4]给出了可信构件的定义, 并提出了保证构件可信性的“ABCDE”构件质量模型. 该模型几乎涵盖了保证可信性的构件 5 个方面的质量属性, 所以其完全实现过于复杂. 而且实际应用中往往仅需要关注其中的某一个方面, 所以更适合作为一种参考模型.

在国内, 有关软件构件可信性的研究处于还起步阶段. 2003 年 10 月科技部批准成立软件构件标准工作组, 并于 2004 年在参考 ISO/IEC 9126[26-29]关于产品质量模型的基础上, 给出构件的两部分模型^[5]: 构件内部/外部质量模型和构件使用质量模型. 2005 年后, 国家自然科学基金委、“863”等国家科技计划中出现了可信软件理论、平台建设等重大科研立项.

从社会角度, 一种产品的可信性与生产者(或其品牌)的信誉度紧密相关. 而这种信誉的累积和提高, 在于生产者对于产品质量的承诺及用户对于这种承诺的认可^[6]. 从生产者角度, 产品质量的承诺建立在对于产品功能的充分测试基础上. 从用户角度, 通过实际使用完成对于产品质量的验证和生产者承诺的评估, 从而决定对于产品或生产者的可信性程度. 上述过程可以总结为一种面向生产者和用户双方的“承诺—评估”机制.

文中贡献是: 尝试从构件提供者角度, 创建一种构件服务的承诺机制并且这种机制是用户可验证的. 与目前已有研究方法不同的是, 文中从服务的角度来定义和度量构件的可信性.

要创建这样一种机制, 关键在于如何给出一种构件所能提供的服务的描述方法, 而且这种方法应该易于用户或第三方的验证实现. 具体研究思路是: 从构件的方法序列特征分析着手, 提出一种构件服务系统的形式化描述方法. 该方法使得构件能够提供何种服务成为构件提供者和构件用户双方的共享资源, 进而在两方之间达成一种约定即“用户只能在构件能够提供的服务中选择满足自己所需要的服务”. 因此从构件提供者角度, 依据构件的功能设计规范, 分析方法之间的关联关系, 确定构件能够提供的合理的方法有限序列即构件服务, 进而给出有效的构件服务系统描述, 并通过实施构件服务测试完善构件设计. 从用户角度, 一种有效构件服务的描述可以帮助用户理解构件的功能, 并基于自己的需求验证确认构件能够提供的有效服务.

文中主要工作: (1) 基于构件本身的功能逻辑分析, 提出一种描述构件服务的形式化方法——构件服务系统, 并定义了具体的系统组成元素及其推理规则; (2) 基于逻辑系统定义, 给出易于实现的构件服务有效性验证算法, 并给出具体的应用示例.

2 基本概念

文中采用如下结构描述一个构件、构件接口及接口中方法:

```
Component_name {
  Interface1: /可视接口 1
  Function1: /可视方法 1
  Function1_name(in_vars, out_vars); /方法型构造
  Preconditions: /调用方法 1 满足的前置条件
  Postconditions: /调用方法 1 满足的后置条件
  Function2: /可视方法 2
  Function2_name(in_vars, out_vars); /方法型构造
  Preconditions:
  Postconditions:
  .....
  Interface2_name:
  .....
}
```

文中从接口方法的粒度讨论构件服务构造及其有效性验证, 因此采用“ $\langle f_1, f_2, \dots, f_u \rangle$ ”的方法序列简化形式. 而一个具体的构件服务, 其本质是由构件接口中若干可视方法构成的有限方法序列, 记作 $\mathscr{s} = \langle f_1, f_2, \dots, f_u \rangle$.

定义 1 前缀(prefix, Pre) 对有序序列 $\mathscr{s} = \langle f_1, f_2, \dots, f_u \rangle$ (u 为整数), $\langle f_1, f_2, \dots, f_i \rangle$ ($i \leq u$) 称作 $\mathscr{s}$ 前缀, 记作 $\text{Pre}(\mathscr{s})$.

其中: $\langle f_1, f_2, \dots, f_{u-1} \rangle$ 称作 $\mathscr{s}$ 的直接前缀, 记作 $\text{Dpre}(\mathscr{s})$. 当 $\text{Dpre}(\mathscr{s})$ 存在时, 把 f_u 称作 $\mathscr{s}$ 的关键字. 这种情况下, 也可以把 $\text{Dpre}(\mathscr{s})$ 称作是 f_u 的直接前缀. $\langle f_1, f_2, \dots, f_u \rangle$ 称作其最大前缀.

定义 2 公共前缀(common prefix, CPre) 对任意有序序列 $\mathscr{s}_1 = \langle f_1, f_2, \dots, f_u \rangle$ 和 $\mathscr{s}_2 = \langle f_1, f_2, \dots, f_v \rangle$, 都包含前缀子列 $\langle f_1, f_2, \dots, f_i \rangle$ ($i < u$ 并且 $i < v$), 该前缀子列称作 $\mathscr{s}_1$ 和 $\mathscr{s}_2$ 的公共前缀, 记作 $\text{Cpre}(\mathscr{s}_1, \mathscr{s}_2)$.

定义 3 序列环(sequence ring, SR) 对任意有序序列 $\mathscr{s} = \langle f_1, f_2, \dots, f_u \rangle$ (u 为整数), 如果 $\exists f \in \{f_i \mid 1 \leq i \leq u\}$ 并且 $\exists i, j$ ($j \geq i \geq 1$), 使得 $\langle f, f_i, \dots, f_j, f \rangle$ 是 $\mathscr{s}$ 的一个子列, 则子列 $\langle f, f_i, \dots, f_j, f \rangle$ 称作 $\mathscr{s}$ 的序列环, 记作 $\text{SR}(\mathscr{s}) = \langle f, <$

$f_i, \dots, f_j > \rangle$.

$(f, \langle f_i, \dots, f_j \rangle)$ 也可称作基于 f 的环.

定义 4 服务 (Service, S) 一个构件中^①的若干方法构成的有限序列 $\langle f_1, f_2, \dots, f_M \rangle$ (M 为一个常数) 称作一个服务 (S), 则 S 必须同时满足: (1) $\langle f_1, f_2, \dots, f_M \rangle$ 中所有 $f_i (i = 1 \dots M)$ 属于同一个构件; 并且 (2) 符合构件的功能设计规范, 即构件对外承诺 (提供) 的一项功能.

条件 (2) 实际上表明构件的功能逻辑, 这一点在领域构件中更显重要. 该条件揭示了构件服务的完备性. 从构件剪裁的角度给出该完备性质. 在给出性质前, 先引入方法依赖的概念.

定义 5 方法依赖 (Function Dependence) 在一个服务 S 中, 任意方法 f_i 依赖于 f_j 当且仅当序列中出现 f_i 时必有 f_j 是其直接或间接的前驱, 记为 $f_j \rightarrow f_i$. (方法依赖的另一层含义是: 当一个方法被调用时, 与之相依赖的其他方法同时被调用)

性质 1 构件剪裁保持完备性.

对于任意构件 C , 其方法集为 F_C , 如果: (1) 存在子构件 C_1 , 其对应方法集 F_{C_1} 和其相对于 F_C 的补集 F_S , 即 $F_C = F_{C_1} \cup F_S$, 并且 (2) 对任两个方法 $f_i \in F_{C_1}, f_j \in F_S$, 不存在 $f_j \rightarrow f_i$ 或者 $f_i \rightarrow f_j$ 的方法依赖, 则称构件 C 是可剪裁的.

(注: “-” (减号) 表示构件的方法集屏蔽; “ $\cup$ ” 表示方法集并关系; “ $\in$ ” 表示方法与方法集间的属于关系; “ $\rightarrow$ ” 称作方法依赖.)

构件服务特点:

(1) 序列有限性. 即对一个具体的构件服务 $S = \langle f_1, f_2, \dots, f_M \rangle$, (M 为一个常数);

(2) 允许存在序列环. 序列环表明, 在一个具体的构件服务中, 一个方法可以被重复调用.

3 构件原子服务

3.1 原子服务概念

定义 6 原子服务 (Atomic Service, AS) 构件服务 $S = \langle f_1, f_2, \dots, f_u \rangle (u \geq 1)$ 称作一个原子服务 (AS), 则 AS 必须满足: 对于 $\forall f_i (i = 1 \dots u - 1)$, $AS - \{f_i\}$ 不构成一个服务.

原子服务是构件能够对外提供的极小方法序列构成的服务, 该定义的等价表述是: $S = \langle f_1, f_2, \dots, f_u \rangle (u \geq 1)$ 是一个原子服务, 如果 $Dpre(S) = \langle$

$f_1, f_2, \dots, f_{u-1} \rangle$ 是其直接前缀. 需要说明的是, 在服务定义中, $\langle f_1, f_2, \dots, f_{u-1} \rangle$ 是 $\langle f_1, f_2, \dots, f_u \rangle$ 的直接前缀表示: f_u 不能单独构成一个服务.

一个构件接口的所有原子服务构成该接口的原子服务集, 记作 $AS_set = \{AS_1, AS_2, \dots\}$.

定理 1 原子服务是构件服务的前缀.

证明: 用反证法. 原子服务集合 AS_set . 假定存在在一个构件服务 $S = \langle f_1, f_2, \dots, f_u \rangle$, 使得 $\forall AS_i \in AS_set$ 都不是 S 的前缀即不存在 $\langle f_1, f_2, \dots, f_i \rangle (1 \leq i < u) \in AS_set$. 根据定义 4 和定义 5, S 本身是一个原子服务. 根据定义 1, 原子服务是其本身的前缀, 这与假设矛盾. 证毕.

一个构件接口方法集记作 $F_set = \{f_1, f_2, \dots\}$.

基于序列特征分析, 定义两种类型原子服务:

定义 7 原子服务类型

类型 1: $\langle f_1, f_2, \dots, f_m \rangle (m \geq 1)$ 其中 $\langle f_1, f_2, \dots, f_{m-1} \rangle$ 是 f_m 的直接前缀.

示例 1: 网上银行业务构件 Auto_bank

Auto_bank {

Interface:

Function1: account_valid / 帐号验证方法

Function2: withdraw / 支取

Preconditions: 余额大于 0 & 支取数不大于余额

Function3: deposit / 储存

Function4: query / 账面信息查询

}

则构件 Auto_bank 的原子服务集合 $AS_set =$

$\{\langle \text{account_valid, withdraw} \rangle, \langle \text{account_valid, deposit} \rangle, \langle \text{account_valid, query} \rangle\}$. 其中 3 个原子服务具有公共前缀 $\langle \text{account_valid} \rangle$.

类型 1 的特例: 单一方法 $\langle f \rangle$ 构成的原子服务.

示例 2: 计算器构件 Calculator

Calculator {

Interface:

Function1: sum / 加法

Function2: subtract / 减法

Function3: multiply / 乘法

Function4: division / 除法

Preconditions: 被除数不为 0

}

^① 说明: 方法序列可以由同一接口的方法组成, 也可以由同一构件的若干接口的方法组成. 更准确地标示应该是“接口_方法”. 讨论方便, 文中省略接口表示.

则构件 Calculator 的原子服务集合 $AS_set =$

$\{ \langle \text{sum} \rangle, \langle \text{subtract} \rangle, \langle \text{multiply} \rangle, \langle \text{division} \rangle \}$

该构件的 4 个原子服务均分别由独立的单个方法构成。

类型 2: $\langle f_1, f_2, \dots, f_m \rangle (f, \langle f_i, \dots, f_j \rangle)$, 即存在基于方法 f 的环。

示例 3: 播放器构件 Video

Video {

Interface:

Function1: play / 播放

Function2: pause / 暂停播放

Function3: stop / 停止播放

Function4: vol_up / 调高音量

Function5: vol_down / 调低音量

}

则构件 Video 的原子服务集合 $AS_set =$

$\{ \langle \text{play}, \text{stop} \rangle (\text{play}, \langle \text{vol_up} | \text{vol_down} \rangle | \langle \text{pause}, \text{vol_up} | \text{vol_down} \rangle) \}$

注: “|” 是并列符号. 其具体解释举例: 根据定义 3, $(\text{play}, \langle \text{vol_up} | \text{vol_down} \rangle) \Leftrightarrow \{ \langle \text{play}, \text{vol_up}, \text{play} \rangle, \langle \text{play}, \text{vol_down}, \text{play} \rangle \}$, 即两者等价。

原子服务特征: (1) 原子服务也是一种构件服务; (2) 对于一个构件接口的任意两个原子服务 AS_1 和 AS_2 , 至少存在一个方法 f 不属于 AS_1 和 AS_2 方法集的交集。

特征 (2) 和定义 5 原子服务的概念相一致, 即可以通过特定的方法标识每一个原子服务。

3.2 服务扩展

定义 8 服务扩展 (service extension) 对于任一构件服务 $S = \langle f_1, f_2, \dots, f_u \rangle$ 和有限方法序列子集设 $f_{seq} = (f_i, \dots, f_j) (|f_{seq}| > 0, \text{即至少包含一个方法})$, 如果 $S' = \langle f_1, f_2, \dots, f_u, f_i, \dots, f_j \rangle$ 或 $\langle f_1, \dots, f_i, \dots, f_j, \dots, f_u \rangle$ 构成一个服务, 则 S' 称作服务 S 的扩展, 也称作服务的有效扩展, 简记作 $\langle S, f_{seq} \rangle$ 。

扩展操作符定义为 “ $\Rightarrow$ ”, 扩展操作或过程记作 $\{S, f_{seq}\} \Rightarrow \langle f_1, f_2, \dots, f_v, f_{seq} \rangle$ 或 $\langle f_1, \dots, f_{seq}, \dots, f_v \rangle$ 。

定义 9 扩展规则 (Extension Rule, ER) 把服务扩展过程中的操作方法或运算称作扩展规则. 所有扩展规则构成扩展规则集记作 ER_set 。

文中定义两种类型的扩展规则如下:

ER1: $\{S, \langle f_i, \dots, f_j \rangle\} \Rightarrow \langle S, f_i, \dots, f_j \rangle$

其中, S 为任一服务; $\langle f_i, \dots, f_j \rangle$ 是任一接

口方法序列。

ER2:

$\{S, (f, \langle f_i, \dots, f_j \rangle)\} \Rightarrow \langle f_1, \dots, f, f_i, \dots, f_j, f, \dots, f_u \rangle$

其中, 服务 S , 存在方法 f 使得 $SR(S) = (f, \langle f_i, \dots, f_j \rangle) (j \geq i \geq 1)$, 即 S 存在基于 f 的环。

实际上, 用户需求和构件服务在两者匹配情况下是等价的. 因此, 文中把能够满足用户需求的构件服务称作目标服务 (Goal Service, GS). 把扩展得到一个 GS 的过程中扩展产生的序列称作桥服务 (Bridge Service, BS)。

因此, 上述规则 ER1 和 ER2 中 S 的形式可能是原子服务 (AS) 或者桥服务 (BS). 其中, 原子服务调用记作 $\langle AS, \omega \rangle$ (ω 表示空方法)。

3.3 服务约束

一定的应用场景中, 一个构件服务通常满足特定方法依赖约束. 因此, 在实施服务扩展过程中, 必须要考虑满足这些约束的有效扩展. 为此, 给出下面的扩展约束定义。

定义 10 扩展约束 (Extension Constraint, EC) 对于任一构件服务 $S = \langle f_1, \dots, f_v \rangle$ (v 是一常数) 和接口方法 f , 如果 $S' = \langle f_1, \dots, f_v, f \rangle$ 不构成一个服务, 则该一次扩展过程是无效扩展, 简记为 $[S, f]$ 。

其中, $[S, f]$ 为无效扩展当且仅当存在 $f_s \in S$, 并且 $f_s \rightarrow f$ 依赖关系不成立。

Tai^[7] 和 Daniels^[8] 从方法的前置条件与后置条件是否匹配 (保持一致性) 角度, 给出了构件接口中两两方法间的约束关系分析. 即在定义 10 中, 对于任一构件服务 $S = \langle f_1, \dots, f_v \rangle$ (v 是一常数) 和接口方法 f , 如果序列 $\langle f_1, \dots, f_v, f \rangle$ 不构成服务当且仅当 $f_v \rightarrow f$ 不成立 (f_v 的后置条件与 f 的前置条件不匹配)。

借鉴文献^[7] 中约束定义方法, 文中从服务扩展得角度给出了 3 种服务扩展约束关系:

EC1: $\text{true}[S; \rightarrow f]$ 表示服务 S 和方法 f 扩展后总是得到一个有效服务。

EC2: $\text{pTrue}[S; \rightarrow f][K]$ 表示: 在某一具体应用场景 K (特定的输入数据) 中, 服务 S 和方法 f 扩展后得到一个有效服务。

EC3: $\text{pFalse}[S; \rightarrow f]$ 表示: 在某一具体应用场景 K (特定的输入数据) 中, 服务 S 和方法 f 扩展后序列不构成一个服务。

EC4: $\text{false}[S; \rightarrow f]$ 表示服务 S 和方法 f 扩展

后得到的序列总是不构成服务。

示例4 以计算器构件 Calculator 为例,其扩展约束为:(假定 S 为该构件的任一服务)

$EC1 = \{ \langle S, \text{sum} \rangle, \langle S, \text{subtract} \rangle, \langle S, \text{multiply} \rangle \};$

$EC2 = \{ \langle S, \text{division} \rangle [\text{被除数不为 } 0] \};$

$EC2 = \{ \langle S, \text{division} \rangle [\text{被除数是 } 0] \};$

$EC3 = \phi(\text{空}).$

构件接口的所有无效扩展(即 $EC3$ 和 $EC4$)构成该接口的扩展约束集,记作 EC_set 。

定义10 体现构件的功能逻辑,因此构件提供者需要依据构件的功能设计规范,具体分析方法之间的相继关联关系,以确定构件能够提供的合理的方法有限序列即构件服务。在确认原子服务,服务的有效/无效扩展等工作中,构件的功能逻辑或设计规范尤其重要,因此为了便于工作的自动化实施,需要给出有效的构件功能逻辑描述,确定构件接口的服务扩展约束。

基于接口方法粒度的功能设计规范即 $\{EC1, EC2\}$,并记作 $DesignSpec$ 。

3.4 相关算法

算法1:判断一方法序列是否满足 $DesignSpec$

```
Function IsValid(SeqType seq)
BEGIN
If seq  $\in$  DesignSpec then return true;
return false;
END
```

算法2:确定扩展约束集

```
Function ExtCon(SetType AS_set, SetType F_set)
BEGIN
 $EC\_set = \phi$ ;
For each atomic service  $AS \in AS\_set$  do
For each function  $f \in F\_set$  do
If IsValid( $\langle AS, f \rangle$ ) = false then
 $EC\_set = EC\_set \cup \{ \langle AS, f \rangle \}$ 
END
```

4 构件服务的形式化描述

4.1 构件服务系统

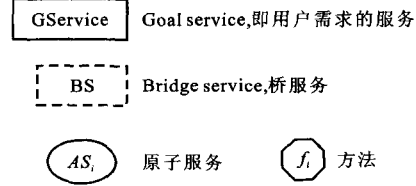
定义11 构件服务系统 一个构件接口的所有服务描述为一个四元组 $\langle AS_set, F_set, ER_set, EC_set \rangle$,其中: AS_set 是该接口所有原子服务的集合;接口方法集 F_set ; ER_set 是服务扩展规则集; EC_set 为扩展约束集。

定理2 一个构件服务能够由某一个原子服务

通过服务扩展构造得到。

该定理的证明步骤等同于以下构件服务构造过程。

先给出服务构造过程图,其中主要元素包括:



构造过程按照“自下往上”扩展方向进行,如图1所示。

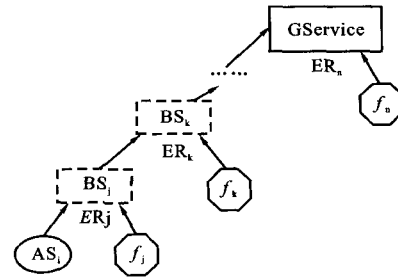


图1 构件服务的构造过程

其中:最左下方的 AS_i 称作该服务的原子服务。

示例5:以网上银行业务构件 Auto_bank 为例,其对应的构件服务系统是 $\langle AS_set, Rules \rangle$:

(1) $AS_set = \{ AS_1, AS_2, AS_3 \}$,其中:

$AS_1 = \langle \text{account_valid}, \text{withdraw} \rangle$;

$AS_2 = \langle \text{account_valid}, \text{deposit} \rangle$;

$AS_3 = \langle \text{account_valid}, \text{query} \rangle$

(2)扩展规则: $ER_set = \{ ER_1 \}$

(3)扩展约束: $EC_set = \{ EC_1, EC_2, EC_3 \}$,

其中:

$EC_1 = \{ \langle S, \text{deposit} \rangle, \langle S, \text{query} \rangle \}$

$EC_2 = \{ \langle AS_1, \omega \rangle, \langle S, \text{withdraw} \rangle \}$

$EC_3 = \{ \langle S, \text{account_valid} \rangle \}$

假定有一用户需求:从账户中取一定数目的钱,然后查询账户余额.该需求可以描述为 $GService = \langle \text{account_valid}, \text{withdraw}, \text{query} \rangle$. 则该服务构造验证过程如图2所示。

4.2 构件服务的自动构造过程

4.1 节中给出的构件服务构造过程本身就是一个针对用户需求有效性的验证过程,即判断构件是能够提供满足用户需求的服务。

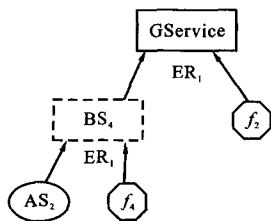


图2 $\langle \text{account_valid}, \text{withdraw}, \text{query} \rangle$ 构造过程

算法是否满足用户需求的构件服务自动构造和验证

输入: 用户需求 R_{seq} (方法序列及相关约束)

输出: True (有效需求) / False (无效需求)

主要数据: 原子服务集 AS_set , 扩展规则集 ER_set , 扩展约束 EC_set .

Step1: 检查是否存在原子服务 $AS_i \in AS_set$, 满足 $R_{\text{seq}} = AS_i$. 如果存在, 则输出 True, 并转向 Step4 结束执行; 否则 Step2;

Step2: 检查是否存在原子服务 $AS_j \in AS_set$, 使得 $\text{Pre}(R_{\text{seq}}) = AS_j$. 如果存在, 则 AS_j 为确定的首原子服务, 并 $BService = AS_j$, 转向 Step3; 否则, 输出 False, 并转向 Step4 结束执行.

Step3: 检查是否存在原子服务 $AS_k \in AS_set$ 并且扩展规则 $ER_k \in Rules$, 满足 $(BService, AS_k) = \rightarrow R_{\text{seq}}$. 如果同时存在相应的 AS_k 和 R_k , 则判断是否 $(BService, AS_k) = \rightarrow R_{\text{seq}}$ 成立. 如果成立则输出 True, 并转向 Step4 结束执行; 否则 $BService = (BService, AS_k)$, 转向 Step3 继续执行.

如果不存在相应的 AS_k 或 (和) R_k , 则输出 False, 并转向 Step4 结束执行.

Step4: 结束.

5 结束语

构件化软件开发已经成为软件技术总体发展趋势之一^[9], 可信构件的研究也自然成为具有重要意义的研究目标之一. 可信构件的研究无论是面向软件技术还是过程管理, 面向事先可信保证还是事后构件验证, 其目标都是提高构件的可信度^[10]. 可信构件的质量模型为构件可信验证提供标准, 构件元模型为研究构件可信度与构件属性之间的关系奠定基础. 构件可信度保证研究也必须建立在这两者基础上.

文中基于构件“开发与应用”两者相分离的特点提出一种构件服务承诺机制. 通过序列特征分析给

出了一种构件服务的形式化描述方法. 当然, 仅有这种承诺机制是不够的, 因为它只是从构件提供者角度给出一种有效构件服务的描述.

参考文献:

- [1] Lv Jian, MA Xiaoxing, Tao Xianping, et al. Research and development of internetwork [J]. Science in China Ser. E Information Sciences, 2006, 36(10): 1037–1080 (Ch).
- [2] Overhage S, Thomas P. CompoNex: a marketplace for trading software components in immature markets[C]// Proceedings of Net ObjectDays. Erfurt, Germany: CompoNex, 2003: 145–163.
- [3] Meyer B, Mingins C, Schmidt H. Providing trusted components to the industry[J]. Computer, 1998, 31 (5): 104–105.
- [4] Meyer B. The grand challenge of trusted components [C]// Proceedings of 25th International Conference on Software Engineering. Portland, Oregon: IEEE, 2003: 660–667.
- [5] 张晓梅, 张为群. 一种基于信任机制的网构软件的构件选择方法研究[J]. 计算机科学, 2010, 37(2): 161–164.
- [6] Luders F, Lau K K, Ho S M. Specification of software components, building reliable component-based software systems [M]. Boston, London: Artech House Computing Library, 2002.
- [7] Mastsumoto M. Highly reliable component-based software development by using algebraic behavioral specification[C]// Proceeding of 3rd IEEE International Conference on Formal Engineering Methods. Washington, D C, USA: IEEE Computer Society, 2000: 35–43.
- [8] Daniels F J, Tai K C. Measuring the effectiveness of method test sequences derived from sequencing constraints[C]// Proceedings of the Technology of Object-Oriented Languages and Systems table of contents. Santa Barbarn, CA, USA: IEEE, 1999: 74–83.
- [9] 杨美清. 软件工程发展思索[J]. 软件学报, 2005, (1): 1–5.
- [10] 王远, 吕建, 徐锋, 等. 一个适用于网构软件的信任度量及演化模型[J]. 软件学报, 2006, 17(4): 682–690.

作者简介:

胡燕京 男, (1980—), 博士研究生, 讲师. 研究方向为系统安全防护、软件安全.