基于服务规约匹配的业务构件获取

孟凡超[1]，初佃辉[1]，战德臣[1,2]，徐晓飞[1,2]

（1. 哈尔滨工业大学（威海）企业与服务智能计算研究中心，山东 威海 264209，mengfanchao74@163.com；
2. 哈尔滨工业大学 企业与服务智能计算研究中心，哈尔滨 150008）

摘 要：针对目前构件模型缺乏行为语义和脆弱接口的问题，提出了一个面向服务的业务构件模型，该模型采用有限状态机来描述业务构件的行为语义，采用基于XML的标记数据类型来表示业务数据类型，基于此模型，提出了动作、动作序列和业务构件3个层次的语义匹配关系，借鉴数据库查询语言，提出业务构件检索语言以支持业务构件的自动获取. 结果表明：所提出的方法为大规模的业务构件复用提供了有利的技术支持. 开发了业务构件库管理工具验证了该法方法的可行性和有效性.

关键词：业务构件；标记数据类型；业务操作服务；业务活动服务；服务匹配
中图分类号：TP311 **文献标志码**：A **文章编号**：0367－6234(2010)07－1112－05

Retrieval of business component based on service specification matching

MENG Fan-chao[1], CHU Dian-hui[1], ZHAN De-chen[1,2], XU Xiao-fei[1,2]

(1. Research Center of Intelligent Computing for Enterprises and Services, Harbin Institute of Technology at Weihai, Weihai 264209, China, mengfanchao74@163.com; 2. Research Center of Intelligent Computing for Enterprises and Services, Harbin Institute of Technology, Harbin 150001, China)

Abstract: To dynamically retrieve reusable business components that can satisfy the service request of user tier from component library, a service-oriented business component model based on service specification matching is proposed to solve the problems of semantic deficiency and weak interface. In this model, the behavior semantic of business component interfaces is described as finite state machine, and business data type is represented as sign data type based on XML. Based on this model, three semantic matching relationships are proposed. Refering to database query language, business component query language is proposed to provide the automatic retrieval of business components. The proposed method can provide effective technology support for large scale business component development and reuse. A business component library management tool is developed to verify the feasibility and effectiveness of this method.

Key words: business component; sign data type; business operation service; business activity service; service matching

目前，大型复杂企业应用软件的开发一般采用分层次的体系结构. 业务构件作为一种大粒度的软件构件是对企业业务逻辑的封装[1]. 业务构件的开发一般要与用户层的用户界面的开发相分离，这样可以提高业务构件的复用度，降低软件系统的开发成本，然而这同时也增加了业务构件的理解与组装的复杂度. 为了提高业务构件的复用效率，本文采用形式化规约来描述业务构件接口的行为语义，并提出了一种基于服务规约匹配的业务构件获取方法，基于该方法能够有效地指导业务构件的复用与组装，从而为基于构件的软件系统的大规模生产提供了有利的技术支持.

收稿日期：2009－11－16.
基金项目：国家自然科学基金资助项目(60773064)；国家科技重大专项资助项目(2009ZX01045-001-002-4)；国家高技术研究发展计划资助项目(2007AA01Z128，2008AA04Z101)；山东省自然科学基金资助项目(2007ZRA10003)；山东省科技发展计划资助项目(2008GG10004010)；江苏省科技发展计划资助项目(BE2009065)；哈尔滨工业大学(威海)校科学研究基金资助项目(HIT(WH)XB200901).
作者简介：孟凡超(1974—)，男，博士，讲师；
战德臣(1965—)，男，教授，博士生导师；
徐晓飞(1962—)，男，教授，博士生导师.

1　相关工作研究

针对构件获取问题，国内外研究人员已经提出了多种获取方法，其中有代表性的方法有：基于关键字的方法、基于刻面的方法和基于规约匹配的方法.

基于关键字的方法是利用一个或多个关键字来表示构件，然后采用基于关键字匹配的检索方法来获取构件[2]. 该方法的优点是表示方便、易于维护和检索，但是缺乏描述复杂构件的能力. 基于刻面描述的方法是采用刻面来表示构件，其中每个刻面描述了构件的一个方面[3-5]. 基于刻面方法具有柔性高、扩展性好以及适应能力强等优点，但是该方法存在刻面使用效率不高问题. 基于规约匹配的方法是采用形式化规格说明来描述构件的接口和行为语义. 构件包含很多语义信息，语义抽象既能精确地描述构件的行为，又能说明构件的功能. 语义抽象的两个典型例子是语法描述和语义描述[6-8]. 语法描述是描述构件方法的参数类型信息，语义描述是描述构件的动态行为特征. 因此，基于规约匹配的方法主要包括语法匹配和语义匹配两方面内容. 目前，国内外学者已经提出了多种基于规约匹配的方法. 文献[9]提出了一个基于 Larch/ML 语言的代码级构件的表示与检索方法. 文献[10]针对大规模数学模型库，提出了基于功能规约的不精确匹配的模型检索方法. 文献[11-12]在其工具 ARIFS 中，使用 SCTL 描述构件，并给出了相应的分类、检索、度量与适配方法.

2　面向服务的业务构件模型

2.1　业务构件规约

业务构件作为一种粒度较大的构件，表达了一个自治的业务概念和处理的软件实现.

定义 1　业务构件可以定义为四元组 $BC = (n, PS, RS, SS)$. 其中，n 为业务构件的名称，PS 为业务构件提供服务集，RS 为业务构件请求服务集，SS 为业务构件的服务规约. SS 描述了业务构件的行为语义，可以抽象为一个有限状态机 $SS = (S, \Sigma, T, s_0, S_F)$. 其中，$S$ 为有限状态的集合，Σ 为动作的集合，$T \subseteq S \times \Sigma \times S$ 为状态迁移的集合，$s_0 \in S$ 为初始状态，$S_F \subseteq S$ 为终止状态集.

定义 2　动作可以定义为四元组 $a = (n, t, \mathrm{In}, \mathrm{Out})$. 其中，$n$ 为动作所对应的业务操作的名称；t 为业务操作的类型；In 为输入业务数据类型；Out 为输出业务数据类型.

定义 3　业务构件的一个执行片段可以定义状态与动作的交替序列，表示为 $\eta = s_0, a_0, s_1, a_1, s_2, \cdots, s_{n-1}, a_{n-1}, s_n$. 其中，$(s_i, a_i, s_{i+1}) \in T$, $i = 0, 1, \cdots, n-1$. 如果 $s_n \in S_F$，则称 η 为业务构件的一个运行.

定义 4　设 $\eta = s_0, a_0, s_1, a_1, s_2, \cdots, s_{n-1}, a_{n-1}, s_n$ 为业务构件的一个运行，由 η 上所有动作组成的序列称为 η 的一个合法动作序列，记为 $l(\eta) = a_0, a_1, \cdots, a_{n-1}$. 可以用 $L(BC)$ 表示业务构件 BC 所接受的合法动作序列集.

2.2　标记数据类型

目前大多数构件模型对于业务数据的描述仍采用传统的数据结构形式，然而这种描述方法受到脆弱接口问题的严重影响. 出现这种问题的主要原因是因为目前大部分接口技术都假设把操作名称编译到目标代码中的特定入口点，并且操作标记中的每个参数的位置和类型与任何调用方都必须匹配. 对于规模较小的软件系统来说，这还是可以管理的，但是对于快速多变的大型复杂企业应用软件来说，这种方法显然缺乏灵活性和可扩展性. 解决脆弱接口问题的一个主要方法是采用“标记数据”来表示业务数据[11]. 目前，描述标记数据的标准是“扩展标记语言 XML”. XML 已经被日益普遍承认和使用，许多标准化组织都开始使用 XML 描述自己的标准. 向许多其他 XML 的使用一样，标记数据也使用 XML 能力的一部分，使业务数据能够作为一种非常简单的文档形式在用户层和业务构件之间进行传输. 在处理标记数据时，业务构件必须对标记数据进行语法检查，并且按照名称抽取业务数据项. 因此，对于标记数据，可以不考虑数据项之间的顺序，只要它能够提供业务构件所需数据项的名称即可. 针对所提出的业务构件模型，本文采用 XML 文档的一个子集来描述每个业务构件所能够识别的标记数据. 为了对标记数据的语法进行检查，采用 DTD 来表示标记数据的元数据，即标记数据类型. 描述标记数据的 XML 文档只包含元素（元素之间可以嵌套），不包含属性（属性可以被看作是元素的一个特例）. 为了表达具有复杂结构的标记数据，允许描述标记数据类型的 DTD 中包含“?”，“+”，“*”和“|”操作符. 由于这些操作符可以进行多重复合，例如((a*)+). 为了规范 DTD 的表示，规定每个元素只能够具有“?”，“+”和“*”中的一种操作，或者“?”，“+”，“*”与“?”的二元复合，如果不满足上述规范，则可以进行相应的转化，例如，((a*)+)? =a*.

由于 DTD 在对 XML 文档进行语法格式进行检

查时,需要区分元素间的位置,而业务构件对于标记数据的语法检查可以忽略同一层元素之间的位置,因此,为了区分 XML 规范中的 DTD 中,用 DTD$^+$ 来表示描述标记数据类型的 DTD. 为了描述的方便,用 DTD$^+$ 表示标记数据类型或者称为业务数据类型,XML$^+$ 表示标记数据或者称为业务数据.

```
<! DOCTYPE Checkbill [
<! ELEMENT Checkbill (Number, Standard, CheckObject,
Applicationbill?, Checker+, CheckItem*)>
<! ELEMENT Number (#PCDATA)>
<! ELEMENT Standard (#PCDATA)>
<! ELEMENT CheckObject (#PCDATA)>
<! ELEMENT Applicationbill (#PCDATA)>
<! ELEMENT Checker (#PCDATA)>
<! ELEMENT CheckItem (Item, Unit, Value>
<! ELEMENT Item (#PCDATA)>
<! ELEMENT Unit (#PCDATA)>
<! ELEMENT Value (#PCDATA)>]>
```

(a) DTD$^+$

```
<Checkbill>
 <Number>ZJ001</Number>
 <Standard>GB-2000</Standard>
 <CheckObject>原煤</CheckObject>
 <Applicationbill>05001</Applicationbill>
 <Checker>张三</Checker>
 <CheckItem>
  <Item>水分</Item>
  <Unit>%</Unit>
  <Value>10</Value>
 </CheckItem>
</Checkbill>
```

(b) XML$^+$

图 1 标记数据类型与标记数据

定义 5 设 D 为一个 DTD$^+$, X 为一个 XML$^+$,如果 X 满足 D 的格式要求,则称 X 为 D 的一个实例,可以用 Instance(X, D) 表示 X 是 D 的一个实例.

定义 6 设 D_1 和 D_2 为两个 DTD$^+$,如果对于 $\forall X \in \text{Instance}(D_1)$,都有 $X \in \text{Instance}(D_2)$,则称 D_1 为 D_2 的子标记数据类型,记为 $D_1 \subseteq D_2$. 如果 $(D_1 \subseteq D_2) \wedge (D_2 \subseteq D_1)$,则称 D_1 等价于 D_2,记为 $D_1 \equiv D_2$.

如果 $D_1 \subseteq D_2$,则说明 D_2 的实例集覆盖 D_1 的实例集,即 D_2 的表达能力要比 D_1 强. 例如,图 1(a) 为一个 DTD$^+$,图 1(b) 中的 XML$^+$ 是该 DTD$^+$ 的一个实例. 图 2 为另一个 DTD$^+$,并且图 1(b) 中的 XML$^+$ 也是该 DTD$^+$ 的一个实例. 从这两个 DTD$^+$ 的格式可以看出,图 1 中的 DTD$^+$ 是图 2 中的 DTD$^+$ 的一个子类型.

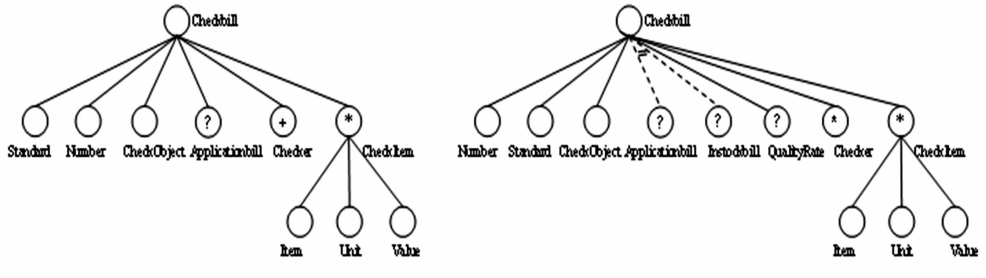
为了判断标记数据类型之间的父子关系,将每个 DTD$^+$ 映射为一棵无序标签树,然后根据标签树之间的匹配关系来判断标记数据类型之间的父子关系.

```
<! DOCTYPE Checkbill [
<! ELEMENT Checkbill (Number, Standard, CheckObject,
(Applicationbill?|Instockbill?), QualityRate?, Checker*, CheckItem*)>
<! ELEMENT Number (#PCDATA)>
<! ELEMENT Standard (#PCDATA)>
<! ELEMENT CheckObject (#PCDATA)>
<! ELEMENT Applicationbill (#PCDATA)>
<! ELEMENT Instockbill (#PCDATA)>
<! ELEMENT QualityRate (#PCDATA)>
<! ELEMENT Checker (#PCDATA)>
<! ELEMENT CheckItem (Item, Unit, Value>
<! ELEMENT Item (#PCDATA)>
<! ELEMENT Unit (#PCDATA)>
<! ELEMENT Value (#PCDATA)>]>
```

图 2 标记数据类型

定义 7 一个 DTD$^+$ 可以表示为一棵无序标签树 $T = (V, E, \text{root}(T))$. 其中, V 为树的节点集,每个节点表示一个元素,root(T) 为树的根节点,E 为边的集合,它是 V 上的二元关系,满足反自反、反对称和传递性. 如果 $(u, v) \in E$,则称 u 是 v 的父节点,记为 $u = \text{parent}(v)$. 使用 $u = \text{parent}\%(v)$ 表示 v 是 u 的可选元素, $u = \text{parent}\%(v_1, v_2, \cdots, v_n)$ 表示 $v_1, v_2, \cdots, v_n$ 是 u 的多选一元素. $u = \text{parent1}(v)$ 表示 v 不能重复,$u = \text{parent}^+(v)$ 表示 v 可以重复 1 到多次, $u = \text{parent}^*(v)$ 表示 v 可以重复 0 到多次.

例如,图 3(a) 为图 1(a) 中 DTD$^+$ 的标签树 T_1, 图 3(b) 为图 2 中 DTD$^+$ 的标签树 T_2. 在描述标记数据类型的无序标签树中,可选和重复关系蕴含父子关系.



(a) 标签树 T_1 (b) 标签树 T_2

图 3 标记数据类型的标签树

首先讨论深度为 1 的两个标签树之间父子类型的判定方法. 设 $T_1 = (V_1, E_1, \text{root}(T_1))$ 和 $T_2 = (V_2, E_2, \text{root}(T_2))$ 为两个深度为 1 的标签树,如果 T_1 和 T_2 之间满足以下条件,则 T_1 为 T_2 的一个子类型.

条件 1 $\text{label}(\text{root}(T_1)) \sim \text{label}(\text{root}(T_1))$. 其中, ~ 为两个标签的名称具有同义词关系.

条件 2 存在从 $V_1 - \{\text{root}(T_1)\}$ 到 $V_2 - \{\text{root}(T_2)\}$ 的子集 V_2 的一个单射函数 f,满足以下条件:

1) $u = v$, iff $f(u) = f(v)$, 其中,u 和 v 是 $V_1 - \{\text{root}(T_1)\}$ 中的节点;

2) $\text{label}(v) \sim \text{label}(f(v))$;

3) $u = \text{parent1}(v)$, iff $f(u) = \text{parent1}(f(v)) \vee f(u) = \text{parent?}(f(v)) \vee f(u) = \text{parent}^+(f(v)) \vee f(u) = \text{parent}^*(f(v))$;

4) u = parent?(v), iff $f(u)$ = parent?($f(v)$) $\vee$ $f(u)$ = parent*($f(v)$);

5) u = parent$^+$(v), iff $f(u)$ = parent$^+$($f(v)$) $\vee$ $f(u)$ = parent*($f(v)$);

6) u = parent*(v), iff $f(u)$ = parent*($f(v)$).

根据条件2可以将 $V_2 - \{\text{root}(T_2)\}$ 划分为两个子集 V_2 和 $V_2 = V_2 - \{\text{root}(T_2)\} - V_2$.

条件 3　对于 $V_2 - \{\text{root}(T_2)\}$ 中的节点，应满足以下条件：

1) 如果 $v \in V_2$，则 u = parent?(v) $\vee$ u = parent*(v) $\vee$ u = parent%($v_1, v_2, \cdots, v, \cdots, v_n$), $v_1, v_2, \cdots, v_n \in V_2 - \{\text{root}(T_2)\}$, u = root(T_2);

2) 如果存在关系 root(T_2) = parent%($v_1, v_2, \cdots, v_n$). 设 $v_1, v_2, \cdots, v_n$ 中的前 k 个元素 $v_1, v_2, \cdots, v_k$ 在 V_2 中，后 $n-k$ 个元素 $v_{k+1}, \cdots, v_n$ 在 V_2 中，如果 $k \geqslant 2$，则在 V_1 中也存在关系 root(T_1) = parent%($u_1, u_2, \cdots, u_k, \cdots, u_m$)，并且有 $u_1 = f(v_1) \wedge \cdots \wedge u_k = f(v_k)$.

根据上述方法可以判断两个是深度为 1 的标签树之间的父子关系. 对于深度 >1 的标签树，可以按照广度优先搜索，依次判断两个相对应单层标签子树之间的父子关系.

3　服务规约匹配

业务构件通过对外提供服务来满足用户层的服务请求. 业务构件服务规约中的每个动作可以看作是粒度最小的服务单元. 首先给出动作之间的语义匹配关系.

定义 8　设 $a_1 = (n_1, t_1, \text{In}_1, \text{Out}_1)$ 和 $a_2 = (n_2, t_2, \text{In}_2, \text{Out}_2)$ 为两个动作，如果 $(t_1 = t_2) \wedge (\text{In}_1 \subseteq \text{In}_2) \wedge (\text{Out}_1 \subseteq \text{Out}_2)$，则称 a_2 语义匹配于 a_1，记为 $a_1 \leqslant a_2$.

定理 1　动作之间的语义匹配关系满足自反和传递性，即：1) $a \leqslant a$；2) $(a_1 \leqslant a_2) \wedge (a_2 \leqslant a_3) \Rightarrow a_1 \leqslant a_3$.

证明　自反性是很显然，下面证明传递性. 设 $a_1 = (n_1, t_1, \text{In}_1, \text{Out}_1)$，$a_2 = (n_2, t_2, \text{In}_2, \text{Out}_2)$ 和 $a_3 = (n_3, t_3, \text{In}_3, \text{Out}_3)$ 为 3 个动作，且 $a_1 \leqslant a_2$ 和 $a_2 \leqslant a_3$. 根据已知条件：则有：1) $t_1 = t_2$，$t_2 = t_3$；2) $\text{In}_1 \subseteq \text{In}_2$，$\text{In}_2 \subseteq \text{In}_3$；3) $\text{Out}_1 \subseteq \text{Out}_2$，$\text{Out}_2 \subseteq \text{Out}_3$. 根据1) 可以推出 $t_1 = t_3$，根据业务数据类型的定义以及 2) 和 3)，可以推出 $\text{In}_1 \subseteq \text{In}_3$ 和 $\text{Out}_1 \subseteq \text{Out}_3$，根据动作语义匹配关系定义，则有 $a_1 \leqslant a_3$.

定义 9　设 $l = a_1, a_2, \cdots, a_n$ 和 $l' = a_1', a_2', \cdots, a_n'$ 为两个动作序列，如果 $a_i \leqslant a_i'$，则称 l' 语义匹配于 l，记为 $l \subseteq l'$.

定理 2　动作序列之间的匹配关系满足自反和传递性，即：1) $l \leqslant l'$；2) $(l_1 \leqslant l_2) \wedge (l_2 \leqslant l_3) \Rightarrow l_1 \leqslant l_3$.

定义 10　设 BC_1 和 BC_2 为两个业务构件，$L(BC_1)$ 和 $L(BC_2)$ 分别为它们的有效动作序列集. 1) 如果对于任意 $l \in L(BC_1)$，存在且只存在唯一 $l' \in L(BC_2)$，满足条件 $l \leqslant l'$，则称 BC_2 等价匹配于 BC_1，记为 $BC_1 \rightarrow_{\text{Equi}} BC_2$；2) 如果对于任意 $l \in L(BC_1)$，存在 $l' \in L(BC_2)$，满足条件 $l \leqslant l'$，则称 BC_2 扩展匹配于 BC_1，记为 $BC_1 \rightarrow_{\text{Extend}} BC_2$；3) 如果对于任意 $l' \in L(BC_2)$，存在 $l \in L(BC_1)$，满足条件 $l \leqslant l'$，则称 BC_2 偏序匹配于 BC_1，记为 $BC_1 \rightarrow_{\text{Part}} BC_2$；4) 如果存在 $l \in L(BC_1)$，$l' \in L(BC_2)$，满足条件 $l \leqslant l'$，则称 BC_2 修改匹配于 BC_1，记为 $BC_1 \rightarrow_{\text{Modi}} BC_2$.

在上述匹配关系中，等价匹配的约束条件最强，修改匹配的约束条件最弱. 这 4 种匹配关系构成了一个柔性匹配模型.

4　业务构件的获取

业务构件的获取是指从业务构件库中检索到满足用户层服务请求所需的业务构件的过程. 业务构件的检索类似于数据库的查询，其查询结果是一组满足检索条件的业务构件集. 下面类比数据库查询语言 SQL，给出业务构件检索模型的表示方法. 业务构件的检索模型可以定义为 Retrieve：$(Q, CL) \rightarrow RC$. 其中，Q 为业务构件检索条件表达式，RC 为业务构件库 CL 中满足检索条件 Q 的业务构件集.

业务构件的检索条件可以表示为 $Q ::= Q \wedge Q \mid Q \vee Q \mid \neg Q \mid q\theta BC$ 的形式. 其中，$\theta \in \{=, >, \geqslant, <, \leqslant, \odot, \neq\}$ 为匹配关系符. 下面则给出基本匹配关系表达式的含义：

1) $q = BC$ 表示 BC 等价匹配于 q. 如果 $Q = (q = BC)$，则有 $RC = \{q \mid (q \in CL) \wedge (q \rightarrow_{\text{Equi}} BC)\}$.

2) $q > BC$ 表示 BC 扩展但不等价匹配于 q. 如果 $Q = (q > BC)$，则有 $RC = \{q \mid (q \in CL) \wedge (q \rightarrow_{\text{Extend}} BC) \wedge \neg (q \rightarrow_{\text{Equi}} BC)\}$.

3) $q \geqslant BC$ 表示 BC 扩展匹配于 q. 如果 $Q = (q \geqslant BC)$，则有 $RC = \{q \mid (q \in CL) \wedge (q \rightarrow_{\text{Extend}} BC)\}$.

4) $q < BC$ 表示 BC 偏序不等价匹配于 q. 如果 $Q = (q \leqslant BC)$，则有 $RC = \{q \mid (q \in CL) \wedge (q \rightarrow_{\text{Part}} BC)\}$.

5) $q \leqslant BC$ 表示 BC 偏序匹配于 q. 如果 $Q = (q \leqslant BC)$，则有 $RC = \{q \mid (q \in CL) \wedge (q \rightarrow_{\text{Part}} BC)\}$.

6) $q \odot BC$ 表示 BC 修改匹配于 q. 如果 $Q = (q \odot BC)$，则有 $RC = \{q \mid (q \in CL) \wedge (q \rightarrow_{\text{Modi}} BC)\}$.

7) $q \neq BC$ 表示 BC 不匹配于 q. 如果 $Q = (q \neq BC)$, 则有 $RC = \{q \mid (q \in CL) \wedge (q \rightarrow_{\text{Non}} BC)\}$.

根据逻辑操作符以及基本匹配关系表达式可以构造复杂的查询条件. 例如, 如果 $Q = (q \geqslant BC_1 \wedge q \leqslant BC_2)$, 则有 $RC = \{q \mid (q \in CL) \wedge (q \rightarrow_{\text{Extend}} BC_1) \wedge (q \rightarrow_{\text{Part}} BC_2\}$.

5 业务构件组装的企业应用软件开发

目前已经开发了业务构件库管理系统来实现本文所提出的方法. 在基于构件组装的企业应用软件开发方法中, 可以将系统的开发分为: 用户界面开发者、业务构件开发者、界面与业务构件的装配者和构件库管理者四类角色, 其中, 用户界面生成者负责用户界面的开发, 为了提高用户界面的开发效率, 可以采用代码生成器辅助完成; 业务构件开发者负责业务逻辑的实现, 这也是企业应用软件的核心部分; 为了实现业务构件的自动组装, 可以将其存储在构件库中, 构件库管理者负责业务构件的描述、分类和维护工作; 界面与业务构件的装配者根据用户界面的服务需求从构件库中自动获取满足其需求的业务构件. 图 4 描述了基于业务构件组装的企业应用软件开发过程.

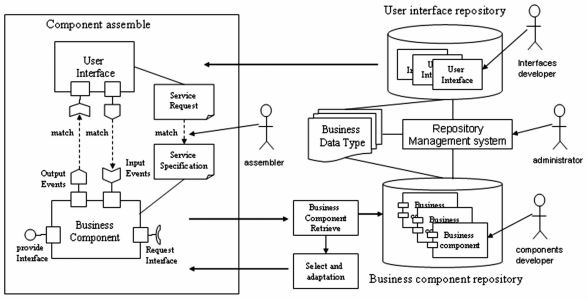


图 4 基于业务构件组装的企业应用软件开发过程

6 结 论

1) 采用标记数据类型描述业务数据类型, 解决了目前构件模型缺乏行为语义和存在脆弱接口的问题;

2) 针对标记数据类型的特点, 提出标记数据类型之间父子关系的概念, 并给出了判断标记数据类型之间父子关系的方法, 以此为基础, 提出了动作、动作序列和业务构件 3 个层次的语义匹配关系, 为业务构件的获取提供了理论基础;

3) 开发了业务构件库管理工具以支持本文所提出的方法的可行性和有效性.

参考文献:

[1] HERZUM P, SIM O. 基于组件的企业开发[M]. 韩柯, 译. 北京: 机械工业出版社, 2005.

[2] FRAKES W B. A case study of a reusable component collection in the information retrieval domain[J]. Journal of Systems and Software, 2004, 72 (2): 265 – 270.

[3] DAMIANI E, FUGINI M G, BELLETTINI C. A hierarchy-aware approach to faceted classification of object-oriented components[J]. ACM Transactions on Software Engineering and Methodology, 1998, 8(3): 215 – 262.

[4] 王渊峰, 薛云皎, 张涌, 等. 刻面分类构件的匹配模型[J]. 软件学报, 2003, 14(3): 401 – 408.

[5] 肖建, 肖岚, 黄毅, 等. 基于领域本体的业务构件描述与组合匹配[J]. 计算机集成制造系统, 2009, 15(9): 1836 – 1843.

[6] 马亮, 孙家肃. 基于规约匹配的构件检索[J]. 小型微型计算机系统, 2002, 23(10): 1153 – 1157.

[7] HEMER D. Specication-based retrieval strategies for component architectures[C]//Proceedings of the 2005 Australian Software Engineering Conference. Washington: IEEE Computer Society, 2005: 233 – 242.

[8] ZAREMSKI A M, WING J M. Specification matching of software components[C]//Proceedings of the 3rd ACM SIGSOFT Symposium on Foundations of Software Engineering. New York: ACM, 1995: 6 – 17.

[9] ZAREMSKI A M, WING J M. Specification matching of software components[J]. ACM Transactions on Software Engineering and Methodology, 1997, 6(4): 333 – 369.

[10] ZHUGE H. An inexact model matching approach and its applications[J]. Journal of Systems and Software, 2003, 67(3): 201 – 212.

[11] VILAS A F, MARTINEZ B B. Approximate retrieval of incomplete and formal specifications applied to vertical reuse[C]//Proceedings of the International Conference on Software Maintenance (ICSM'02). Washington: IEEE Computer Society, 2002: 618 – 627.

[12] REDONDO R P D, ARIAS J J P, VILAS A F, *et al*. Approximate retrieval of incomplete and formal specifications applied to horizontal reuse[C]//Proceedings of 28th Euromicro Conference. Washington: IEEE Computer Society, 2002: 90 – 97.

(编辑 张 红)