基于接口矩阵分析的构件变化检测方法的研究 *

李心科, 严鲁明
(合肥工业大学 计算机与信息学院, 合肥 230009)

摘 要: 针对构件的变化性问题一直都是基于构件的软件工程(CBSE)中的一个关键问题,首先对构件模型以及构件匹配原则进行形式化描述,构造了构件的接口关系矩阵;然后根据构件匹配原则,对构件接口关系矩阵进行分析,动态地检测构件的变化,判断构件变化可能会对其他相关构件以及整个系统产生影响;最后实现了构件变化分析的辅助工具 CIDT(component interface detecting tool),并在软件开发以及维护过程中使用 CIDT 对系统进行动态的检测和分析。

关键词: 基于构件的软件工程; 构件; 适应性; 变化检测; 接口矩阵
中图分类号: TP311.56 **文献标志码**: A **文章编号**: 1001-3695(2009)04-1360-03

Interface matrix-based detecting method for change of component

LI Xin-ke, YAN Lu-ming
(*Institute of Computer & Information, Hefei University of Technology, Hefei* 230009, *China*)

Abstract: This paper analyzed the change of component in the system, and estimated the impact on both the correlative component and the whole system. Firstly described the component model and adaptation principle in formal specification, and then constructed the connection matrix of component interface. According to the above information, developed a tool CIDT, which was used in CBSE to analyze the incidence of the changed component.

Key words: CBSE; component; adaptation; detecting of change; interface matrix

基于构件的软件工程(CBSE)已成为一种被广泛应用的软件工程模式。CBSE 的根本目的是为了解决软件危机问题,并促进软件的自动化开发进程。从目前的应用现状来看,基于构件的软件工程的优点在于:a)大大提高了软件的开发效率;b)提高了软件(构件)的可复用性;c)降低了软件开发成本;d)提高了软件的质量。

在基于构件的软件开发过程中,构件是开发的基本单元,新的软件系统是通过对已有构件的组装和集成实现的,因此构件的变化会对其他构件以及整个系统产生影响。构件变化的适应性的检测和相容性的判断是解决构件变化后系统正确性和稳定性问题的有效办法。

本文在对构件模型以及构件匹配原则进行描述的同时,构造了构件的接口关系矩阵,并通过接口关系矩阵来追踪和检测构件动态变化的影响范围和构件的相容情况。

1 相关研究情况

关于构件的变化检测,研究者提出了多种方法和机制。文献[1,2]中提出了一种符号级的构件检测机制,该机制通过定义参数流的语法,判断每个参数在代码中的执行方式与顺序来检测构件的适应性。符号级的检测方法可以准确的判断构件接口参数对于构件匹配的影响,但是该方法要求详细的了解构件内部结构和代码,不适合黑盒以及灰盒构件的检测。文献[3]通过构件交互接口的静态列表和链路来检测软件体系结构中构件动态变化的适应性,追踪构件变化的影响范围。该方法为检测构件的动态变化性提供了方便,但是由于一个构件有多个接口,且在软件中可能被不同的其他接口调用,因此,在该方法中需要对构件接口使用的每种情况都建立一条链路或者列表,增加了检测之前的工作量,不适应大型系统的检测。文献[4]提出了软件体系结构的语义模型,并利用三种构件适应结构来判断构件的适应性。语义模型可以有效的检测构件之间的匹配关系,但是由于缺乏整体性,该模型不利于软件体系结构的整体分析。

2 构件模型与构件匹配原则

为了便于描述构件以及构件接口之间的关系,在本章中将对构件、构件接口以及构件匹配原则进行形式化的描述。

定义 1 构件。在 CBSE 中,构件 $C=\langle I,F,S\rangle$。其中:$I=\{I_1,I_2,\cdots,I_n\}(n\in N)$ 为构件的接口集合,$I_i(1\leqslant i\leqslant n)$ 表示构件 C 的第 i 个接口;F 表示构件 C 的功能函数,$F(i)$ 与 $I_i(1\leqslant i\leqslant n)$ 相对应,表示接口 I_i 的功能函数;$S=\{S_1,S_2,\cdots,S_n\}(n\in N)$,$S$ 为构件 C 的状态集合,每个方法 $F(i)$ 均使构件达到一个相应的构件状态,记为 $F(i)\rightarrow S_i$。

定义 2 接口。在构件中,构件接口 $I=\langle I^{In},I^{out},D^{In},D^{out}\rangle$。其中:

$I^{In}=\{P_1,P_2,\cdots,P_k\}(k\in N)$ 表示接口 I 的输入接口,$\{P_i\}$ 表示 I^{In} 的参数集合;

$I^{out}=\{P_1,P_2,\cdots,P_s\}(s\in N)$ 表示接口 I 的输出接口,$\{P_i\}$ 表示 I^{out} 的参数集合;

D^{In} 表示 I 的输入接口 I^{In} 的接口约束集合;

D^{out} 表示 I 的输入接口 I^{out} 的接口约束集合。

收稿日期: 2008-07-22; **修回日期**: 2008-10-22 **基金项目**: 安徽省科技厅科技攻关重点基金资助项目(06012021A)

作者简介:李心科(1963-),男,江苏徐州人,副教授,博士,主要研究方向为软件工程、神经网络 (thinker_lee9268@163.com);严鲁明(1984-),男,安徽滁州人,硕士研究生,主要研究方向为软件工程.

为了方便表示,本文用 param(I^{In})表示接口 I 相应的输入接口参数集合,param(I^{out})表示接口 I 的输出接口参数集合;把一对相应的输入输出接口记为$\langle I^{In}, I^{out}\rangle$;构件 C 的第 i 个接口记为 C_I_i。

定义 3　接口相同。对于任意的接口 $I_1=\langle I_1^{In}, I_1^{out}, D_1^{In}, D_1^{out}\rangle$,$I_2=\langle I_2^{In}, I_2^{out}, D_2^{In}, D_2^{out}\rangle$,若$(\text{param}(I_1^{In})=\text{param}(I_2^{In}))\wedge(\text{param}(I_1^{out})=\text{param}(I_2^{out}))\wedge(D_1^{In}=D_2^{In})\wedge(D_1^{out}=D_2^{out})$,则称接口 I_1 与 I_2 相同,记为 $I_1\equiv I_2$。

在构件组装的过程中,相同的接口之间是可以相互替换的,具有相同使用接口的构件也是可以替换的。

定义 4　输入匹配。对于任意的输入接口 I^{In} 和输出接口 I^{out},若$(\text{param}(I^{out})=\text{param}(I^{In}))\wedge(D^{out}\subseteq D^{In})$,则称从接口输出 I^{out} 到输入接口 I^{In} 是输入匹配的,记为 $I^{out}\Delta I^{In}$。

定义 5　接口匹配。对于任意的接口 $I_1=\langle I_1^{In}, I_1^{out}, D_1^{In}, D_1^{out}\rangle$,$I_2=\langle I_2^{In}, I_2^{out}, D_2^{In}, D_2^{out}\rangle$,若存在 $I_1^{out}\Delta I_2^{In}$ 或 $I_2^{out}\Delta I_1^{In}$,则称 I_1 与 I_2 或 I_2 与 I_1 是接口匹配的,记为 $I_1\rightarrow I_2$ 或 $I_2\rightarrow I_1$。

满足接口匹配原则的接口之间是可以进行组装和连接的,接口匹配是构件匹配的基础。

定义 6　构件可连接性。对于构件 $C_1=\langle I_{C1}, F_{C1}, S_{C1}\rangle$,$C_2=\langle I_{C2}, F_{C2}, S_{C2}\rangle$,若存在一组接口 $I_i\in I_{C1}$,$I_j\in I_{C2}$,满足 $I_i\rightarrow I_j$,则称构件 C_1 与 C_2 可连接,C_1 与 C_2 之间具有可连接性,记为 $C_1\rightarrow C_2$。

在基于构件的软件开发过程中,可以根据以上的构件模型描述和构件连接原则来判断构件之间的可组装性,形成基本的软件体系结构。

3　构件接口矩阵与分析

3.1　构件关系分析

通过上文中提出的构件模型描述以及相关连接原则,可以初步得到软件系统的连接模型,利用有向图的相关理论,将软件模型中的每个构件看做有向图的一个节点,构件的连接看做有向边,得到相应的有向图,即构件关系图。构件关系图表示了软件体系结构中各个构件之间的连接关系。一个构件关系图可以表示为 $G=(C,E)$。其中:C 表示关系图中各个构件节点,简称为节点;E 为图中有向弧的集合。

定义 7　构件邻接性。对于构件 $C_i\in C$、$C_j\in C$,若 $\exists e\in E$ 且 $e=(C_i,C_j)$,则表示构件 C_i 与 C_j 相连。其中 C_i 的输出接口与 C_j 的输入接口相连,C_i 为输出构件,C_j 为输入构件。同时称构件 C_i 与 C_j 邻接,表示为 $C_i\Rightarrow C_j$。为了方便区别邻接构件中的输入/输出关系,将从 C_i 到 C_j 称为正向邻接,从 C_j 到 C_i 称为逆向邻接。

构件的邻接性表示构件关系图中构件之间的直接关系,当某一构件发生变化时,最可能受影响的是与该构件之间具有邻接关系的构件。

通过构件模型的描述(定义 1)可以知道,一个构件包含多个构件接口,不同的接口对应了不同的功能,构件之间是通过接口进行连接的。构件关系图紧紧反映了构件之间的关系,因此要准确地对构件进行检测就必须确定构件接口的关系。

定义 8　接口邻接性。在构件关系图中,构件 $C_i\in C, C_j\in C, C_i=\langle I_{Ci}, F_{Ci}, S_{Ci}\rangle$,$C_j=\langle I_{Cj}, F_{Cj}, S_{Cj}\rangle$,若 $C_i\Rightarrow C_j$ 且接口 $I_m\in I_{Ci}$,$I_n\in I_{Cj}$ 是构件间的连接的接口,则称接口 I_m 与 I_n 是邻接接口,表示为 $I_m\Rightarrow I_n$,称 C_i 与 C_j 时关于接口(I_m,I_n)邻接的;将从 I_m 到 I_n 称为正向邻接,从 I_n 到 I_m 称为逆向邻接。

定义 9　接口连通性。在构件关系图中,构件 $C_i\in C, C_j\in C, C_i=\langle I_{Ci}, F_{Ci}, S_{Ci}\rangle$,$C_j=\langle I_{Cj}, F_{Cj}, S_{Cj}\rangle$,如果:

a)在构件关系图中至少存在一条路径使得从 C_i 到 C_j 是可达的,即 $C_i\Rightarrow\cdots\Rightarrow C_j$;

b)$I_m\in I_{Ci}$,$I_n\in I_{Cj}$ 分别为 C_i,C_j 的连接接口,即 $I_m\Rightarrow\cdots\Rightarrow I_n$;

则称 I_m 与 I_n 是接口连通的,记为 $I_m\sim I_n$,I_m 与 I_n 正向连通,I_n 与 I_m 逆向连通。

结论 1　在软件体系结构 S 中,当构件 $C_i=\langle I_{Ci}, F_{Ci}, S_{Ci}\rangle$ 的接口 I_m 发生变化时,在 S 中任意的构件 $C_j=\langle I_{Cj}, F_{Cj}, S_{Cj}\rangle$ 满足:$I_n\in I_{Cj}$,$I_m\Rightarrow I_n$ 或 $I_n\Rightarrow I_m$,$I_m\sim I_n$ 或 $I_n\sim I_m$ 时,构件 C_j 将受到构件 C_i 的影响。

3.2　构件接口矩阵

为了便于讨论,本文只考虑理想状态下的构件关系图,即简单有向无环图。

本文在 3.1 节中阐述了构件关系图以及关系图中构件之间的关系和接口之间的关系。根据有向图理论每个有向图都与一个邻接矩阵相对应。在构件关系图中,关系图的邻接矩阵反映的是构件之间的关系,为了准确地反映构件接口的关系,本文将根据接口关系和邻接矩阵得出构件的接口矩阵。

根据简单有向图理论可知构件关系图的邻接矩阵可表示为 $A=(a_{ij})$。其中:

$$a_{ij}=\begin{cases}1;\text{从 } C_i \text{ 到 } C_j \text{ 是正向邻接关系}\\0;\text{从 } C_i \text{ 到 } C_j \text{ 不是正向邻接关系}\end{cases}$$

图 1 表示了一个构件关系图,$G=(C,E)$。

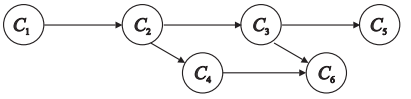

图1 构件关系图

图 1 的邻接矩阵为

$$A=\begin{pmatrix}0&1&0&0&0&0\\0&0&1&1&0&0\\0&0&0&0&1&1\\0&0&0&0&0&1\\0&0&0&0&0&0\\0&0&0&0&0&0\end{pmatrix}$$

本文的目的是要通过细化软件体系结构中构件之间的关系,通过接口来确定构件之间的相互影响。根据上文叙述的接口之间的关系,可以将构件关系图中的邻接矩阵转换为相应的接口矩阵。

定义 9　接口矩阵。设构件关系图为 $G=(C,E)$,$C=\{C_1,C_2,\cdots,C_n\}(n\in N)$,$B=(b_{ij})$ 为 G 的接口矩阵。其中:

$$b_{ij}=\begin{cases}(I_s,I_t);I_s\in I_{cj},\text{从 } I_s \text{ 到 } I_t \text{ 是正向邻接关系}\\(0,0);\text{构件 } C_i \text{ 到 } C_j \text{ 不是邻接关系}\end{cases}$$

接口矩阵反映了构件中接口之间的邻接关系。由构件关系图的邻接矩阵和接口矩阵的定义可以看出,邻接矩阵与接口矩阵是可以相互转换的,邻接矩阵中 $a_{ij}=0$ 的位置对应了接口矩阵中 $b_{ij}=(0\ 0)$;$a_{ij}=1$ 反映了接口矩阵中相应的位置存在接口邻接关系。

以图 1 为例,假设图 1 中邻接构件之间的接关系如下:

$$C_1\_I_1\Rightarrow C_2\_I_1;$$
$$C_2\_I_1\Rightarrow C_3\_I_1;$$
$$C_2\_I_2\Rightarrow C_4\_I_1;$$

$$C_3\_I_1 \Rightarrow C_5\_I_1;$$
$$C_3\_I_2 \Rightarrow C_6\_I_1;$$
$$C_4\_I_1 \Rightarrow C_6\_I_2$$

有上述关系可以得到图 1 的接口矩阵为

$$B=\begin{pmatrix}(0\,0) & (I_1\,I_1) & (0\,0) & (0\,0) & (0\,0) & (0\,0)\\ (0\,0) & (0\,0) & (I_1I_1) & (I_2I_1) & (0\,0) & (0\,0)\\ (0\,0) & (0\,0) & (0\,0) & (0\,0) & (I_1I_1) & (I_2I_1)\\ (0\,0) & (0\,0) & (0\,0) & (0\,0) & (0\,0) & (I_1I_2)\\ (0\,0) & (0\,0) & (0\,0) & (0\,0) & (0\,0) & (0\,0)\\ (0\,0) & (0\,0) & (0\,0) & (0\,0) & (0\,0) & (0\,0)\end{pmatrix}$$

4 构件变化检测机制与工具实现

在 CBSE 中,构件的变化是构件内部功能方法的变化。对于构件 $C=\langle I,F,S\rangle$,$F(i)$对应了一个特定的 $I_i(I_i\in I)$,构件的变化是 $F(i)$的变化,因此,构件的变化可以映射到相应的接口上。本文根据上述关系,在构件发生变化时,通过构件关系图以及构件接口矩阵来检测构件变化的影响范围,判断构件变化的相容性。

4.1 构件变化检测机制

本文在 3.2 节中定义了构件关系图和构件接口矩阵。由构件接口矩阵的定义可以知道,在接口矩阵中 a_{ij}反映了从构件 i 到构件 j 的正向接口邻接的关系,同时也反映了从构件 j 到构件 i 的逆向接口邻接的关系。因此对于构件接口矩阵有以下性质:a)接口矩阵的第 i 行的所有向量表示了构件 C_i的各个接口与其他构件接口之间的正向邻接关系;b)接口矩阵的第 j 列的所有向量表示了构件 C_j的各个接口与其他构件接口之间的逆向邻接关系。

根据构件接口矩阵的性质以及构件内部特性,当软件体系结构中构件发生变化时,可以通过对构件接口矩阵的遍历来检测构件变化对整个体系结构产生的影响,具体检测方法将在下文中具体介绍。

设构件关系图 $G=(C,E)$,$C=\{C_1,C_2,\cdots,C_n\}(n\in N)$,构件 $C_i=\langle I_{Ci},F_{Ci},S_{Ci}\rangle\in C$,当构件 C_i发生变化且对应的变化接口为 $I_p\in I_{Ci}$时,则构件变化的检测方法如下:

a)由构件关系图的邻接矩阵和构件接口匹配关系得出构件接口矩阵,$B=(b_{ij})$;

b)在构件接口矩阵中,检索第 i 行的向量 $b_{ij}=(I_s,I_t)(j=1,2,\cdots,n)$,找出 $I_s=I_p$的向量,即查找接口 $C_k\_I_t$,从 $C_k_I_t$到 I_p呈逆向邻接关系,记接口集合为 $A-=\{C_k\_I_t\}$;

c)检索第 i 列的向量 $b_{ji}=(I_s,I_t)(j=1,2,\cdots,n)$,找出 $I_t=I_p$的向量,即查找接口 $C_k\_I_s$,从 $C_k_I_s$到 I_p呈正向邻接关系,记为集合 $A-=\{C_k\_I_s\}$;

d)重复 b),找出与集合 $A-=\{C_k_I_t\}$中各个元素呈逆向邻接关系的构件接口,放入 $A-=\{C_k_I_t\}$中;

e)重复 c),找出与集合 $A+=\{C_k_I_s\}$中各个元素呈正向邻接关系的构件接口,放入 $A+=\{C_k_I_s\}$中。

根据构件接口矩阵的性质,在检索后的集合中,I_p与 $A-$中的元素 $C_k_I_t$为正向连通关系,I_p与集合 $A+$中元素 $C_k_I_s$呈逆向连通关系(图 2),集合 $C^*=A+\cup C_i\cup A-$是构件 C_i的接口 I_p变化后影响的范围。

在上述检测方法中,构件集合 C^*是原系统构件的子集,即 $C^*\subseteq C$。因此,可以得到 $G^*=(C^*,E^*)$是图 $G=(C,E)$的子图,由 G^*中构件构成的系统是原系统的一个子系统。所以当构件发生变化后,判断系统的正确性以及构件变化带来的影响等价于判断由 C^*构成的子系统的正确性,这就避免了以往构件变化后需要对整个系统或者系统中一个较大范围进行检测的情况。

4.2 构件变化检测工具 CIDT 的实现

在上述理论的基础上,本文开发了相应构件变化检测工具 CIDT,并将 CIDT 在安徽省科技计划网上申报评审系统的开发中进行了应用。CIDT 主要由三个功能部分组成,即构件匹配判断模块(CIM)、构件接口关系生成模块(CIR)和构件变化追踪模块(CID)。CIDT 结构如图 3 所示。

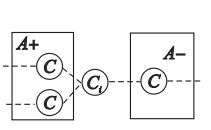

图2 构件变化检索关系图

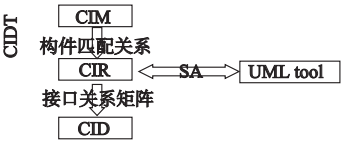

图3 CIDT结构关系

CIDT 主要有以下功能:a)根据输入的构件来判断构件间的匹配关系;b)根据已有的匹配关系产生软件体系结构中构件关系模型,并产生相应的构件接口矩阵;c)构件发生变化时,对变化进行追踪,检测变化的影响范围,产生受影响的构件列表。在 CIDT 中为了方便建模引入了 UML 模型,并利用 Rational ROSE 工具对软件体系结构进行 UML 建模。

5 结束语

本文通过对构件的形式描述,引入构件接口矩阵来表示软件体系结构中构件之间的接口关系。当构件发生变化时,本文利用接口矩阵提出了一种新的构件变化检测方法。该方法在已有方法的基础上更精确地确定了构件变化的影响范围,为解决 CBSE 中构件变化带来的软件一致性问题和软件测试提供了参考。

参考文献:

[1] 胡海洋,吕建,马晓星,等.面向对象范型体系结构中构件行为相容性研究[J].软件学报,2006,17(6):1276-1286.

[2] XIE Xiong, ZHANG Wei-shi. A checking mechanism of software component adaptation[C]//Proc of the 5th International Conference on Grid and Cooperative Computing(GCC 2006). Washington DC: IEEE Computer Society, 2006: 347-354.

[3] FENG Tie, MALETIC J I. Applying dynamic change impact analysis in component-based architecture design[C]//Proc of the 7th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing. Washington DC: IEEE Computer Society, 2006: 43-48.

[4] BRANDON M, PERRY A. Spartacas: automating component reuse and adaptation[J]. IEEE Trans on Software Engineering, 2004, 30(9):587-600.

[5] 楚旺,钱德沛.以体系结构为中心的构件模型的形式化语义[J].软件学报,2006,17(6):1287-1297.

[6] 孙莹,陈松乔.接口连接式构件组装的一种形式化方法[J].计算机科学,2007,33(7):253-256.

[7] 柯尧,赵保华,屈玉贵.基于组件系统的可靠性分析[J].北京邮电大学学报,2005,28(6):115-119.

[8] CRAIG D C, ZUBEREK W M. Verification of component behavioral compatibility[C]//Proc of the 2nd International Conference on Dependability of Computer Systems. Washington DC: IEEE Computer Society, 2007: 347-354.