高技术通讯 2013 年 第23 卷 第2 期:146 ~ 152

doi:10.3772/j.issn.1002-0470.2013.02.006

考虑构件有效可靠性的构件式系统可靠性评测①

郭 勇② 马培军 苏小红

(哈尔滨工业大学计算机科学与技术学院 哈尔滨 150001)

摘 要 针对目前基于构件的系统的可靠性评测方法没有充分考虑构件有效可靠性导致评测结果不够准确的问题,进行了构件有效行为、构件有效可靠性确定方法及系统可靠性评测方法的研究,提出了一种考虑了构件有效可靠性的评测方法。该方法首先通过系统UML模型抽取出场景规约中所描述的构件的有效行为,根据有效行为确定构件在系统中的有效可靠性,然后采用Markov理论对系统进行可靠性建模,进而对整个系统进行可靠性评测。通过对一个具体系统的可靠性的计算给出了评测结果,结果表明,考虑构件有效行为后,对基于构件系统的可靠性的评测更加合理。

关键词 有效可靠性,可靠性评测,基于构件的系统,构件有效行为,细粒度评测

0 引 言

近年来基于构件的软件开发方法日益受到人们的重视,人们在进行系统开发时可以选择通用构件。但通用构件一般都包含多种功能,而在实际使用这些构件时,常常仅用到构件的部分功能,恰好满足要求没有冗余功能的构件很少[1],这会给基于构件的系统的可靠性评测带来困难。基于构件的系统的可靠性评测模型主要分为基于路径的模型[2-5]、加模型[6-8]和基于状态的模型[9-12]。基于路径的模型主要考虑系统所有可能的执行路径,根据路径的可靠性估算出系统的可靠性;加模型并未明确考虑系统结构,用非齐次泊松过程(non-homogeneous Poisson process, NHPP)建模构件可靠性,通过构件的失效数据算出系统的可靠性;基于状态的模型假设系统的状态变化过程符合Markov理论,主要将应用程序建模成时间离散的马尔科夫链(discrete time Markov chain, DTMC)[10, 11]、时间连续马尔科夫链(continuous time Markov chain, CTMC)[9]及半马尔可夫过程(semi-Markov process, SMP)[12]。基于路径的模型和基于状态的模型通常都假设构件的可靠性为已知并且为常量[13-15],这种假设不尽合理[16],加模型虽然考虑了构件的评测,但没有具体考虑构件中实际被使用的行为。许多构件在系统中被使用的只是部分行为,而用构件全部行为对应的可靠性对系统进行可靠性评测显然是不准确的。相对未被使用或较少使用的行为而言,被频繁使用的构件行为对构件可靠性的影响更大。针对这种情况,本文提出了一种考虑构件有效行为的更细粒度的可靠性评测方法,该方法根据构件在系统中实际被使用的行为确定构件的有效可靠性,然后使用得到的结果对系统进行可靠性评测。该方法需要解决三个关键问题,即构件行为的抽取、构件在系统中使用概率的计算以及考虑构件行为后的构件可靠性的计算。

1 依据需求规约进行构件行为抽取

基于构件的系统的可靠性分析通常需要以下信息:用户使用系统的情况、系统各个构件的可靠性、构件之间交互的情况及软件运行环境的可靠性。本文将具体考虑构件的行为,通过构件的行为确定构件的有效可靠性。我们可以通过系统的统一建模语言(unified modeling language,UML)模型进行构件行为的抽取,来确定实际参与系统的构件行为。对这一过程的描述要用到场景(scenario)和协作图(collaboration diagram)的概念。场景是指某一功能的具体执行过程,即构件之间的交互。协作图能充分体

① 国家自然科学基金(61173021, 61073052)资助项目。

② 男,1967 年生,博士生;研究方向:软件可靠性评测,计算机通讯;联系人,E-mail: guoy@hit.edu.cn

(收稿日期:2012-04-10)

现构件之间的交互,它有两种形式:实例格式和一般格式。实例格式的协作图仅显示单一场景的交互,没有任何分支和循环;一般格式协作图包含顺序、分支和循环等,本文使用实例格式的协作图。下面给出协作图中构件的定义,其中协作图中的消息对应构件的外部行为。

定义 1　构件为一个七元组:$< ID, B, P, B', P', A, D >$,其中 ID 为构件的标识,B 为构件提供给外界的行为的集合,P 为构件对外行为协议的描述,B' 为构件内部行为的集合,P' 为构件内部行为协议的描述,A 为构件的应用域集(可具有不同的可靠性),D 为构件在不同应用域中可靠性相关数据。P、B 与协作图中消息的关系为 $P(B) \rightarrow M$,其中 M 是协作图中消息的集合。

定义 2　构件的行为 Bc 是指构件中含有的所有方法的集合,仅在构件内部使用的方法也称内部行为 B',外部其它构件可用的方法称外部行为 B,且有 $Bc = B \cup B'$。

下面的定义是针对系统 S 给出的。

定义 3　构件的直接有效行为 Bd:在 UML 协作图中至少出现一次的构件的外部行为称为构件的直接有效行为。即:$\forall f \in B$, $\exists s \in Sq$ 使得 $p(f) \rightarrow m$,其中 $m \in M, p \in P$,设 Sq 是系统 S 的协作图的集合,M 是协作图中消息的集合。

对于给定场景的协作图,我们主要的工作就是从协作图中抽取出构件的直接有效行为 Bd。要准确抽取构件的有效行为,首先要保证协作图的正确性和充分性。本文所提方法的前提是 UML 模型中的协作图充分体现了系统实际运行时各构件的行为。对于协作图的验证问题,可参考文献[17]。图 1是构件行为的抽取过程。

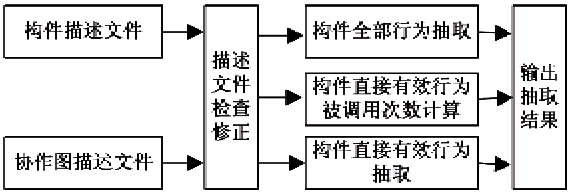

图 1　构件行为抽取过程

2　构件有效可靠性确定

抽取了系统中实际使用的构件行为后,可以根据构件的具体行为进一步确定构件在系统中的可靠性,为此引入有效可靠性的概念。

定义 4　构件有效可靠性的定义 Ra:假设 C 是系统 S 中的一个构件,C 有 n 个直接有效行为,Bd 为构件 C 的直接有效行为的集合,$\forall b_i \in Bd$,r_i 是 b_i 对应的可靠性,Bd 对应的可靠性即为构件 C 在系统 S 中的有效可靠性,$Ra = f(r_1, r_2, r_3, \cdots, r_n,)$。

当同一个构件应用于不同的系统时,该构件在具体系统中的有效可靠性不一定相同,因为该构件的有效可靠性取决于构件在该系统中的实际被使用的行为及使用频率。对于构件有效可靠性的确定可根据具体情况采用相应的方法。

第一种方法,如果提供了构件具体行为的可靠性,可根据每个构件实际被使用的行为进行可靠性计算。在进行构件行为抽取时我们同时得到了构件的直接有效行为及其在系统中的使用情况,依据这些信息可计算出各个构件在具体系统中的有效可靠性。不同的系统使用构件的方式不尽相同,这种情况下有效可靠性需根据具体系统来确定。

第二种方法:如果无法直接得到构件行为的可靠性,可考虑采用测试的方法。由于已经得到了构件实际的运行剖面,因此可以通过仿真的方式对构件进行测试,以确定构件的可靠性。有些构件开发商会提供构件的某些可靠性相关数据或给出一个测试规范和相应的测试集,这种情况下可使用这些规范和相关数据通过对构件有效行为的测试确定构件的有效可靠性。如果构件是按内建测试(Built-in-Test, BIT)[18]规范开发的,可使用内建测试函数对构件的有效行为进行测试,得到构件的有效可靠性。还可采用面向方面的编程(aspect oriented programming, AOP)技术或构件包装技术,加入相应的探测代码得到每个构件行为的调用信息,通过这些信息计算出每个行为的可靠性,然后再计算构件的有效可靠性。

第三种方法:前提是构件的整体可靠性已给出。设 Ri 为构件 C_i 的可靠性,该构件含有 n_i 个行为,其中通过构件行为抽取得知该构件有 k_i 个有效行为,构件 C_i 的有效可靠性 Ra 可按下式计算:

$$R_a = R_i^{k_i/n_i} \tag{1}$$

上面的三种方法中,第一种方法能较好地确认构件的有效可靠性,同时还不需要做太多的工作;第二种方法较为复杂,其结果的准确性依赖于仿真方式;第三种方法仅是一种近似的方法,只有当前两种方法都无法采用时,才会选用第三种方法。

3 UML 模型到 Markov 模型的转换

构件有效可靠性确定后,就可以使用已创建的 UML 模型进行系统可靠性评测,但直接使用 UML 模型不便于进行可靠性的评测,还需要进行转换。本文将根据系统的协作图构造基于状态的可靠性分析 Markov 模型。

3.1 转换过程

定义 5 用于可靠性分析的 Markov 模型为一个五元组 $< Q, s_1, s_k, M, R >$,其中 Q 为 k 个状态的集合 $\{s_1, s_2, \cdots, s_k\}$;一个状态定义为 $s_i = \{C_i\}$,其中 C_i 是一个 m 个构件的集合 $\{c_{i1}, c_{i2}, \cdots, c_{im}\}$;$s_1$ 为初始状态;s_k 为最终状态;M 为 $k \times k$ 状态转移概率矩阵;R 为 k 个状态对应的可靠性的集合,$R = \{r_1, r_2, \cdots, r_k\}$。

从 Markov 模型的定义可以看出,模型中的主要元素有三类:状态、状态间的转移概率和各个状态对应的可靠性。Markov 模型构造过程如图 2 所示。

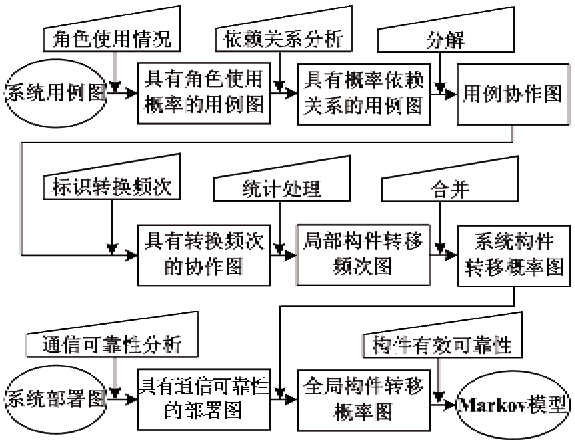

图 2 Markov 模型构造过程

3.2 有概率依赖关系的用例图的构建

定义 6 具有概率依赖关系的用例图是一个十元组 $< A, U, L, R, N, S, NL, P(a), F(a,u), V(i,j) >$,其中 A 为角色的有穷集合,用于表达事件的参与者;U 为用例的有穷集合,用于表示系统的功能;L 为角色与用例之间的连接;R 为用例之间使用关系,$R = \{\text{include, extend, generalization}\}$;$N$ 为注释,对用例图的相应说明;S 表示系统;NL 表示注释连接;$P(a)$ 表示角色 a 使用系统 s 的概率;$F(a,u)$ 表示角色 a 使用用例 u 的概率;$V(i,j)$ 表示用例 i 使用用例 j 的概率。

以饮料自动销售系统为例具体说明其转换过程。图 3 所示为饮料自动销售系统的具有概率依赖关系的用例图。

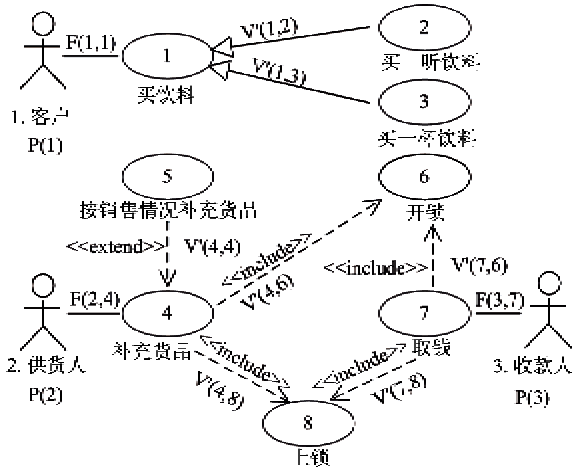

图 3 饮料自动销售系统用例图

系统包括三种角色,Customer、Collector、Supplier,这些角色使用系统的概率分别为 $P(1)$、$P(2)$、$P(3)$;系统被分解为八个用例 $U = \{$Buy Drink, Buy a can of drink, Buy a cup of drink, Restock, Restock according to sales, Unlock, Lock, Collect$\}$。根据 UML 知识可知"Include"关系中基用例使用与被包含用例的概率为 1,基用例使用扩展用例的概率 $\leqslant 1$,基础用例与与泛化用例之间的调用概率是 $\leqslant 1$。

为了区分用例和单一场景用例,在后面的叙述中将可分解为多个单一场景的用例称为普通用例,分解后的称为单一场景用例。

图 3 所示的具有概率依赖关系的用例图粒度较大,需要进一步分解为粒度较小的具有概率依赖关系的用例图,为此引入单一场景用例图的概念。

定义 7 图中的所有用例都分别对应一个单一场景,每个单一场景中不包含分支情况,这种用例图称为单一场景用例图。单一场景用例图通常也称实例格式的用例图。

用例事件流的描述是需求分析文档的一部分,有主事件流、候选事件流和其它事件流等,其中每一个事件流对应一个更细粒度的单一场景用例。"Buy a can of drink"用例包含单一场景用例如图 4 所示。可以根据经验得到顾客在使用机器时各个场景出现的概率,设"Success"的概率为 $w(1,1)$,"No drink"的概率为 $w(1,2)$,"Dibs wrong"和"No money return"的概率分别为 $w(1,3)$、$w(1,4)$。

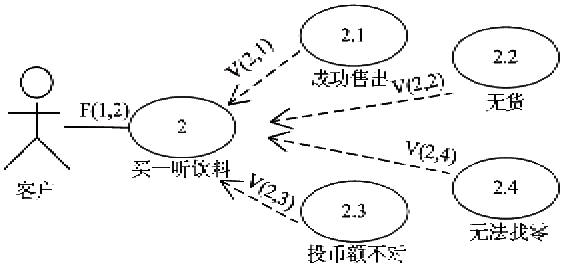

图 4　"Buy a can of drink"用例的单一场景用例图

根据图 4 可算出在所有单一场景用例图中角色 a 使用单一场景用例 z 的概率 $w(a,z)$，计算方法如下式所示：

$$w(a,z) = p_a \sum_{u=1}^{n_a} (F(a,u) \times v(u,z)) \qquad (2)$$

其中 P_a 为角色 a 使用系统的概率，$F(a,u)$ 为角色 a 在系统中使用普通用例 u 的概率，$V(u,z)$ 为普通用例 u 调用单一场景用例 z 的概率，n_a 为角色 a 所用到的普通用例数。

3.3　局部构件转移频次图的构建及转移频次的计算

系统包含四个构件。分别为 *Front*、*Register*、*Dispenser* 和 *SafeManager*。为进一步分析，下面给出单一场景用例的协作图。单一场景协作图中不存在分支，图中每个行为都必然要执行，只要知道每个场景的执行概率就可以算出构件总的执行概率，并且协作图与构件转移频次图相似，清楚的描述出了构件之间的关联及构件之间消息的传递，每个消息对应构件的一个行为。为了便于研究再加入一个 *Complete* 构件，表示该功能执行完毕。

图 5 为图 4 中单一场景用例"Success"的协作图。通过协作图可以分析出构件在实现"Buy a can of drink"这个功能时的构件间的交互次数，通过对构件交互次数的统计就可以确定构件的转移频次。

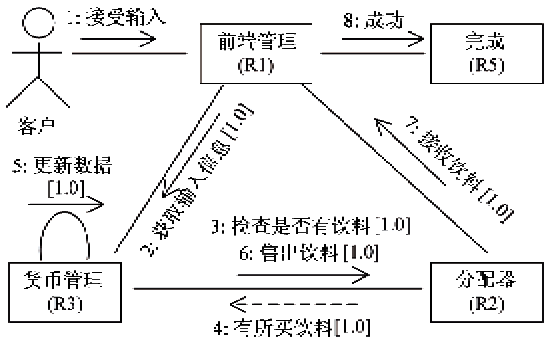

图 5　"Success"用例的协作图

根据局部构件转移频次图可计算出普通用例中构件之间的交互频次。设 m_{ijk} 为构件 i 与构件 j 在普通用例 k 的单一场景 z 中的交互次数，该值可通过分析系统 UML 模型文件计算得到。普通用例 k 共有 n_k 个单一场景，系统共有 N 个角色，则构件 i 与构件 j 在普通用例 k 中的总的交互频次 λ_{ijk} 计算方法如下式所示：

$$\lambda_{ijk} = \sum_{a=1}^{N} \left(\sum_{z=1}^{n_k} m_{ijz} \times w(a,z) \right) \qquad (3)$$

按式(3)计算得到"Buy drink"、"Collect"、"Restock"三个功能用例中构件的相互调用频率值，计算结果是归一化前的值，可能会大于 1，最后再进行归一化处理。

3.4　Markov 模型生成

上面计算出了每个普通用例中构件之间的交互情况，系统中包含多个普通用例，因此需将每个普通用例中的同一构件的交互情况统计出来，得到整个系统中构件总的交互情况，下面讨论具体计算方法。

设 f_{ij} 为构件 i 在整个系统中调用构件 j 的频次，系统有 K 个普通用例，则 f_{ij} 可用下式计算得出：

$$f_{ij} = \sum_{k=1}^{K} \lambda_{ijk} \qquad (4)$$

为了准确评测系统的可靠性，还需要统计出每个构件被调用的频次，设 u_i 为构件 i 在整个系统中被调用的频次值，系统共有 n 个构件，则 u_i 可用下式计算得到：

$$u_i = \sum_{x=1}^{n} f_{ix} \qquad (5)$$

可靠性 Markov 模型中需要构件间的转移概率，这个值可以通过 f_{ji} 和 u_i 得到。设 τ_{ij} 为构件 i 到构件 j 的转移概率，τ_{ij} 可通过下式计算得到：

$$\tau_{ij} = f_{ij}/u_i \qquad (6)$$

系统构件转移图描述了系统中构件间的转移情况，在可靠性分析时还需要加入每个构件的可靠性信息。为便于分析所构建的 Markov 模型中还要包括功能完成后的正常终止状态 C 和失效状态 F，系统构件转移图中的每一个构件都对应 Markov 链中的一个非吸收状态，起点 Start 和终点 Complete 转换为 Markov 链中的起始状态和终止状态，最后生成的 Markov 可靠性模型如图 6 所示。

图 6 中 R_1、R_2、R_3、R_4、R_5 为构件 *Front*、*Dispenser*、*Register*、*SafeManager*、*Complete* 最终被确认的有效可靠性，整个系统的可靠性为 Rs。

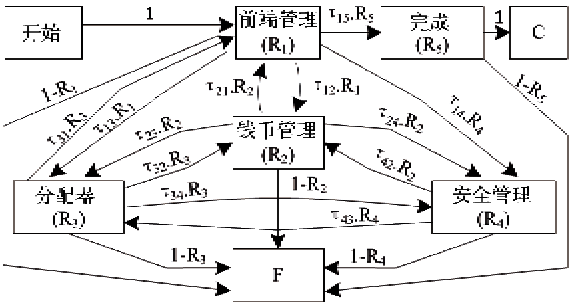

图 6 系统的 Markov 可靠性模型

4 系统可靠性计算

可根据得到的 Markov 模型,采用文献[11]中的方法进行可靠性计算,计算方法如式

$$R_s = S(1,n) \times R_n = (-1)^{n+1} \frac{|E|}{|I-Q|} \times R_n \tag{7}$$

所示。其中 R_s 为系统可靠性,$S(1,n)$、R_n、E、I、Q 等变量的具体含义见文献[11]。

5 实验及结果分析

以图 3 所示系统为例说明系统可靠性的计算方法。系统共有三种角色,使用系统的概率分别为 $P(1)=0.8$、$P(2)=0.1$、$P(3)=0.1$。构件 *Front*、*Register*、*Dispenser*、*SafeManger* 和 *Complete* 分别用 C_1、C_2、C_3、C_4、C_5 来表示。

下面以“Buy drink”为例具体描述上述过程。“Buy drink”用例共有两个普通用例:“Buy a cup of drink”和“Buy a can of drink”,根据经验这两个用例的使用概率分别为 0.5,即 $V(1,2)=V(1,3)=0.5$,而对于角色 1(*customer*)来说 $F(1,1)=1$。

表 1 为计算得到的“Buy a can of drink”的场景交互次数即 m_{ijk} 的值及各单一场景的使用概率。

表 1 构件在“Buy a can of drink”中各单一场景交互次数(m_{ijk})

场景 1 $v(2,1)=0.8$	场景 2 $v(2,2)=0.1$	场景 3 $v(2,3)=0.05$	场景 4 $v(2,4)=0.05$
$m_{121}=1$,	$m_{122}=1$,	$m_{123}=1$,	$m_{124}=1$,
$m_{151}=1$,	$m_{152}=1$,	$m_{153}=1$,	$m_{154}=1$,
$m_{231}=2$,	$m_{212}=2$,	$m_{213}=1$,	$m_{214}=2$
$m_{311}=1$,	$m_{232}=1$,	$m_{233}=2$,	
$m_{321}=1$	$m_{322}=1$	$m_{313}=1$,	
		$m_{323}=1$	

根据式(2)得到各角色在“Buy a can of drink”中使用单一场景用例的概率 $w(1,1)=0.64$, $w(1,2)=0.08$, $w(1,3)=0.04$, $w(1,4)=0.04$,其它均为 0。根据式(3)得到构件 i 与构件 j 在用例“Buy drink”中的总的交互频次 λ_{ijk} 如表 2 所示。按同样的方法可以算出构件在其他单一场景中总的交互次数。

表 2 构件在“Buy drink”中总的交互次数(λ_{ijk})

	C_1	C_2	C_3	C_4	C_5
C_1	0	0.8	0	0	0.8
C_2	0.28	0	1.44	0	0
C_3	0.68	0.76	0	0	0
C_4	0	0	0	0	0
C_5	0	0	0	0	0

通过式(4)可以计算出系统中构件之间总的交互情况。表 3 是按式(4)计算得到的系统中全部构件之间的交互频次,表 4 是按式(6)计算的构件之间的总的转移概率。

表 3 系统中构件间的交互频次(f_{ij})

	C_1	C_2	C_3	C_4	C_5
C_1	0	0.87	0.07	0.18	0.88
C_2	0.43	0	1.44	0.05	0
C_3	0.868	0.76	0	0.012	0
C_4	0	0.2	0.2	0	0
C_5	0	0	0	0	0

表 4 系统中构件间的转移概率(τ_{ij})

	C_1	C_2	C_3	C_4	C_5
C_1	0	0.435	0.035	0.09	0.44
C_2	0.224	0	0.75	0.026	0
C_3	0.5293	0.4634	0	0.0073	0
C_4	0	0.5	0.5	0	0
C_5	0	0	0	0	0

假设系统中构件的可靠性分别为 $Rc_1=0.992$、$Rc_2=0.9866$、$Rc_3=0.993$、$Rc_4=0.986$、$Rc_5=1$,根据得到的系统中各构件的有效行为来确定各构件的有效可靠性,系统中的初始数据及计算结果如表 5所示。

表 5　系统初始数据及计算结果

	C_1	C_2	C_3	C_4	C_5
Rc	0.992	0.9866	0.993	0.986	1
Bc	12	4	8	9	1
Ba	7	4	6	5	1
Ra	0.9953	0.9866	0.9947	0.9922	1

Markov 模型中各构件的可靠性是根据 *Ra* 和构件调用频次计算的得到的,其计算式为

$$R(i) = (R_a(i))^{u_i} \tag{8}$$

其中 u_i 是根据式(5)计算得到的,$u_1 - u_5$ 的值分别为 1.298、1.83、1.71、0.242、0.88,$R(i)$ 为构件 i 在 Markov 状态模型中的可靠性,$Ra(i)$ 为构件 i 的有效可靠性。按式(8)计算得 Markov 模型中各构件的可靠性值为 $R(1) = 0.9939$、$R(2) = 0.9756$、$R(3) = 0.9910$、$R(4) = 0.9981$、$R(5) = 1$。根据式(7)计算得系统可靠性评测值为 $Rs = 0.9319$。

作为对比,再计算一下未具体考虑构件有效行为,而是将整个构件的可靠性作为 Markov 模型中各构件对应状态的可靠性时,计算得到的系统可靠性。将式(8)中的 $Ra(i)$ 用表 5 中的 $Rc(i)$ 来替换,计算得未考虑构件有效行为的各构件对应的可靠性为 $R_1 = 0.9896$、$R_2 = 0.9756$、$R_3 = 0.9881$、$R_4 = 0.9966$、$R_5 = 1$。计算得 $Rs = 0.9193$。

通过上面计算结果可以看出,本例中考虑构件有效行为后系统可靠性评测结果比不考虑构件有效行为时的值高出了 1.37%。在不考虑构件有效行为时,构件的可靠性是综合了构件全部行为的可靠性值,如果在实际系统中未使用构件的全部行为,则未被使用的行为对系统的可靠性没有影响,这部分行为带来的故障应被排除在系统可靠性评测之外。

6　结　论

针对目前基于构件系统可靠性评测中将构件可靠性作为常量来进行评测而带来的评测结果不够合理的问题,本文提出了考虑构件有效行为的基于构件的系统可靠性评测方法,该方法粒度更细,能使评测结果更接近于系统实际运行时的可靠性。本文采用了系统的 UML 模型来进行构件有效行为的抽取,给出了确定构件有效可靠性的方法,在确定了构件的有效可靠性后,本文给出了系统 Markov 模型的建立方法,并给出了实验结果。由计算结果看出,通过本文方法评测的系统的可靠性通常会高于未考虑构件行为时的可靠性。如果不考虑构件的有效行为,在构件选择时就有可能将符合可靠性要求的构件当成不符合要求的构件,从而缩小了构件的选择范围。考虑构件行为后可以在更大范围内进行构件选择,使构件的选择更具灵活性并且可以降低系统的开发成本。

未来的工作包括研究如何抽取构件内部的有效行为以及用于容错和例外处理的行为,以及这些行为对系统可靠性的影响,这样可以进一步提高构件有效可靠性评测的准确性。

参考文献

[1] 张岩, 胡军, 于笑丰等. 场景驱动的构件行为抽取. 软件学报, 2007, 18 (01): 50-61

[2] Shooman M L. Structural models for software reliability prediction. In: Proceedings of the 2nd International Conference on Software Engineering, San Francisco, USA, 1976. 13-15

[3] Krishnamurthy S, Mathur A P. On the estimation of reliability of a software system using reliabilities of its components. In: Proceedings of the 8th International Symposium On Software Reliability Engineering, Albuquerque, New Mexico, 1997. 146-155

[4] Fiondella L, Gokhale S S. Software reliability with architectural uncertainties. In: Proceedings of the 2008 IEEE International Symposium on Parallel and Distributed Processing, Miami, USA, 2008. 1-5

[5] Hsu C J, Huang C Y. An adaptive reliability analysis using path testing for complex component-based software systems. *IEEE Transactions on Reliability*, 2011, 60 (1): 158-170

[6] Goseva-Popstojanova K, Trivedi K S. Architecture-based approach to reliability assessment of software systems. *Performance Evaluation*, 2001, 45 (2): 179-204

[7] Xie M, Wohlin C. An additive reliability model for the analysis of modular software failure data. In: Proceedings of the 6th International Symposium on Software Reliability Engineering, Toulouse, France, 1995. 188-194

[8] 侯春燕, 崔刚, 刘宏伟. 实现构件软件可靠性分析的改进的可加模型. 高技术通讯, 2011, 21 (3): 267-272

[9] Gokhale S S, Michael Rung-Tsong L. A simulation approach to structure-based software reliability analysis. *IEEE Transactions on Software Engineering*, 2005, 31 (8): 643-656

[10] Wei Y, Shen X-H. Heterogeneous architecture-based software reliability estimation: Case study. In: Proceed-

ings of the 3rd International Conference on Convergence and Hybrid Information Technology, Busan, Korea, 2008. 286-290

[11] Cheung R C. A user-oriented software reliability model. *IEEE Transactions on Software Engineering*, 1980, SE-6 (2): 118-125

[12] Kubat P. Assessing reliability of modular software. *Operations Research Letters*, 1989, 8 (1): 35-41

[13] Gokhale S S, Trivedi K S. Reliability prediction and sensitivity analysis based on software architecture. In: Proceedings of the 13th International Symposium on Software Reliability Engineering, Annapolis, USA, 2002. 64-75

[14] Cortellessa V, Grassi V. A modeling approach to analyze the impact of error propagation on reliability of component-based systems. In: Proceedings of the 10th International Symposium on Component-Based Software Engineering, Heidelberg, Germany: Springer Berlin, 2007. 140-156

[15] Wang W L, Pan D, Chen M H. Architecture-based software reliability modeling. *Journal of Systems and Software*, 2006, 79 (1): 132-146

[16] Immonen A, Niemela E. Survey of reliability and availability prediction methods from the viewpoint of software architecture. *Software and Systems Modeling*, 2008, 7 (1): 49-65

[17] 胡军, 于笑丰, 张岩等. 基于场景规约的构件式系统设计分析与验证. 计算机学报, 2006, 29 (04): 4513-4525

[18] Yingxu W, King G, Wickburg H. A method for built-in tests in component-based software maintenance. In: Proceedings of the Third European Conference on Software Maintenance and Reengineering, Amsterdam, Holand, 1999. 186-189

Evaluating a component-based system's reliability by considering the effective component reliability

Guo Yong, Ma Peijun, Su Xiaohong

(School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001)

Abstract

In view of the problem that the current reliability evaluation methods for component-based software systems are inaccurate because they do not consider the component effective reliability, a new reliability evaluation method for component-based systems with the feature of considering the effective component reliability is presented based on the studies of determining components' effective behaviors and their effective reliability in a system, and evaluating the system's reliability. The proposed method extracts the effective behaviors of the components described in the scenario-based specifications through the description file of the system's unified modeling language (UML) models. And then, the reliabilities of the components are determined according to their effective behaviors; the evaluation model is made according to the Markov theory. The evaluation results of a specific case system are given, and they show that the consideration of the effective behaviors of component can make the evaluation of a software system more reasonable.

Key words: effective reliability, reliability evaluation, component-based software system, component effective behavior, fine-grained evaluation