软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2014,25(2):373−385 [doi: 10.13328/j.cnki.jos.004541]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

嵌入式 API 测试套生成方法和技术*

赵会群, 孙 晶, 张 爆, 王同林

(北方工业大学 信息工程学院,北京 100144)

通讯作者: 赵会群, E-mail: zhaohq6625@sina.com

摘 要: 随着嵌入式计算机系统应用的不断扩展,嵌入式系统的可靠性引起了学术界和工业界的广泛关注,也提出了很多增进可靠性的方法和技术.然而,现有的方法和技术在测试套生成方面论述不多,所以在处理大批量嵌入式系统测试工作中遇到了挑战.讨论抽象测试套生成方法和适配技术,提出了 LTS(labeled transition system)到 BT(behavior tree)的转换算法,从而使 TTCN(test and testing control notation)测试套可以通过转换嵌入式软件的 LTS 描述产生.还介绍了基于上述转换算法的嵌入式软件测试工具包,以及一个嵌入式物联网识读器测试案例研究.

关键词: 嵌入式软件;软件测试;测试与测试控制语言;标签转换系统

中图法分类号: TP311 **文献标识码**: A

中文引用格式: 赵会群,孙晶,张爆,王同林.嵌入式 API 测试套生成方法和技术.软件学报,2014,25(2):373−385. http://www.jos.org.cn/1000-9825/4541.htm

英文引用格式: Zhao HQ, Sun J, Zhang B, Wang TL. Method and technique of test suite generation for embedded API. Ruan Jian Xue Bao/Journal of Software, 2014,25(2):373−385 (in Chinese). http://www.jos.org.cn/1000-9825/4541.htm

Method and Technique of Test Suite Generation for Embedded API

ZHAO Hui-Qun, SUN Jing, ZHANG Bao, WANG Tong-Lin

(School of Information Engineering, North China University of Technology, Beijing 100144, China)

Corresponding author: ZHAO Hui-Qun, E-mail: zhaohq6625@sina.com

Abstract: With the rapid increase of embedded computer system applications, the reliability of embedded software has drawn particular attention from researchers and industries. Many methods for testing and verifying reliability of embedded software have been discussed. However, the existing methods are weak in test suite automatic generation and therefore difficult in tackling large numbers of embedded computer applications. In this paper, the method and the technique of generating abstract test suite and their adaptation to a computer platform are presented. An algorithm for translating a LTS (labeled transition system) into BT (behavior tree) is proposed. Consequently, the TTCN (test and testing control notation) abstract test suite that employs BT as logical structure can automatically be generated with respect to the LTS description of embedded software. A TTCN tool set based on the translation algorithm for testing embedded software is introduced, and case study of testing embedded system of Internet of things device is presented.

Key words: embedded software; software testing; TTCN (test and testing control notation); LTS (labeled transition system)

随着计算机系统应用的不断扩展,计算机嵌入式系统在一些关键领域中已经发挥了作用,如物联网、移动计算、信息物理融合等嵌入式系统应用等等.由于嵌入式系统对资源的限制、对实时性的高要求以及嵌入式软件与硬件的相关性,使得嵌入式软件的设计、开发和维护遇到了很多新的挑战,其中之一是嵌入式软件分析与测试方法和技术方面的挑战.

与传统的软件测试方法和技术不同,由于嵌入式软件的运行环境相对封闭,软件发布后维护的代价会很高,所以在软件发布之前,需要模拟嵌入式运行环境仿真测试软件的可靠性和一致性[1,2].由于运行环境与仿真环境

* 基金项目: 国家自然科学基金(61070030, 61370051); 北京市教委人才创新团队计划(4062012)
收稿时间: 2013-04-28; 修改时间: 2013-09-29, 2013-12-05; 定稿时间: 2013-12-17

存在差异,所以仿真测试的效果往往受到质疑.为了在软件运行过程中实时监控和分析嵌入式系统的运行状态,通常采用插桩式的监控手段,即,在嵌入式软件中插入侦听代码,以便及时报告系统运行状态[3,4].由于嵌入式系统对资源和实时性要求高,所以插桩测试与分析方法在一定程度上受到限制.

随着软件规模的增大,软件测试的代价也在急剧增加.通常,软件测试费用要占软件开发周期总投入的 50%以上[5].为了降低测试环节的成本,测试用例(套)的自动生成、测试用例的简化成为技术关键,也引起了学术界的关注,成为热点研究问题.目前,对测试用例生成的研究比较多,也取得了显著的研究成果,如,王林章等人[6]基于UML 协作图设计了一种测试用例生成方法,不仅根据协作图中的消息,而且还对消息后继中对象可能的实现和调用场景产生测试用例;张智秩等人[7]针对测试用例由于软件演化而失效的问题,研究如何从测试用例选择、修复和扩增这 3 个方面解决测试用例演化的问题;聂长海等人[8]针对被测计算机系统测试各种参数组合测试中测试表生成的效率问题,提出最佳配置点决策的贪心算法,解决了因影响因素多而不容易配置决策点的问题;Fraser 等人[9]提出了一种基于模型检测反例自动生成测试套的方法,给出模型检测和软件测试相结合的软件可靠性保障的方法.嵌入式软件的测试用例生成研究也是如此[10,11].

测试套不仅是测试用例的集合,同时还包括测试用例执行环境的配置参数、测试用例执行的步骤和指令[12].在基于模型的测试(model based test,简称 MBT)[13]方法中,为了优化测试用例,往往只关注测试用例的效能,而把测试环境的配置和执行步骤等细节留在执行测试过程中进行,这样可以提高测试用例的效能,也给测试套增加了可伸缩性,这种测试套称为抽象测试套.TTCN 是 ISO/ETSI 颁布的测试与测试控制语言,用于描写抽象测试套[14].

测试套生成是一项具有挑战性的技术,一方面需要根据软件规范说明生成某种抽象测试套描述语言程序,另一方面需要将抽象测试套配置到具体的执行环境中.鉴于测试用例自动生成已经有很好的研究成果,本文将结合物联网软件测试需求,集中讨论嵌入式软件测试套生成的方法和技术.采用 LTS 图作为嵌入式系统软件规格说明描述工具,采用 TTCN-3 作为抽象测试套描述语言,研究 LTS 到 TTCN-3 自动转换的算法;采用嵌入式系统 API 函数反向插入技术,解决 TTCN 抽象测试套的适配问题.

本文第 1 节讨论基于 LTS 的 TTCN-3 测试套生成算法以及测试用例的生成算法.第 2 节结合物联网识读器测试需求,讨论嵌入式软件抽象测试套的适配方法和技术.第 3 节应用上述研究结论、案例研究物联网识读器测试套生成实现.第 4 节介绍相关研究工作并给出本文研究结论.

1 基础知识

简单介绍测试套生成算法的基础知识,给出标签转换系统、μ-演算和行为树的概念.

1.1 标签转换系统(LTS)

标签转换系统 LTS[15,16]由状态集合、状态之间转换集合和转换使能活动集合构成,定义如下:

定义 1. 设 A 和 P 分别为活动和谓词集合,一个基于 A 和 P 的标签转换系统是一个四元组$\langle S,I,\rightarrow,\models\rangle$.其中,

1) S 是状态集合.
2) $I\in S$ 是状态集合中一个可区分的状态,称为初态.
3) $\rightarrow$是一个二元关系结合 $\xrightarrow{a}\subseteq S\times S$,称为转换,记为 $s\xrightarrow{a}t$.其中,$s,t\in S;a\in A$ 是一个转换使能活动.
4) $\models$是一个二元关系集合$\models\subseteq S\times P$,记为:$s\models p$,称谓词 $p\in P$ 在状态 $s\in S$ 被满足.

LTSs 也称为 Kripke 结构,是经典的模态逻辑模型,常用于并行理论中表达系统状态在活动的作用下实现系统进化的过程.

定义 2. 设 $LTS=(S,I,\rightarrow,\models)$是一个标签转化系统,称集合 $L(LTS)\subseteq A\times S\times P=\{\langle a,s,p\rangle\mid t\xrightarrow{a}s\ ,t,s\in S,a\in A,p\in P\}$是一个标签类,记为 L.特别地,当 L 用μ-calculus 状态公式编码时,称 L 为 LTS 的一个实例,记为 LI.

例如,$S=\{0,1,2\}$,$I=\{0\}$,$\rightarrow=\{\langle 0,1\rangle,\langle 1,1\rangle^*,\langle 1,2\rangle\}$,$P=\{\text{True},\text{False}\}$,$A=\{a_0,a_1,a_2,\ldots,a_9\}$是用数字表示的活动编号.$L(LTS)=\{\langle a_0,0,\text{True}\rangle,\langle a_1,1,\text{True}\rangle^*,\langle a_3,2,\text{False}\rangle\}$表达了一个活动 a_0,a_1 执行的实例.

1.2 μ-calculus

μ-calculus[17]是一种命题模态逻辑,通常用于描述一个 LTS 应该满足的性质,并用于模型检验.不仅如此,许多时态逻辑都可以用μ-calculus 进行编码,比如计算树逻辑 CTL(computation tree logic,简称 CTL)[18].

μ-calculus 中使用状态公式(STATE FORMULAS)描述应该满足逻辑条件.它的产生式定义如下:

F::="true"|"false"|"not" $F|F_1$ "or" $F_2|F_1$ "and" $F_2|F_1$ "implies" $F_2|F_1$ "equ" $F_2|$"<" R ">" $F|$
"[" R "]" $F|$"@" "(" R ")"$|X|$"mu" X "." $F|$"nu" X "." F.

其中,F 是逻辑公式;R 是正则表达式,产生式定义如下:

R::=$A|$"nil"$|R_1$ "." $R_2|R_1$ "|" $R_2|R$ "*"$|R$ "+".

其中,A 是活动公式,定义如下:

A::=*string*|*regexp*|"true"|"false"|"not" $A|A_1$ "or" $A_2|A_1$ "and" $A_2|A_1$ "implies" $A_2|A_1$ "equ" A_2.

例如,一个安全性质"没有故障发生"可以表示为[true*."OPEN!1".(not "CLOSE!1")*."OPEN !2"]false.

1.3 TTCN行为树

行为树(behavior tree,简称 BT)是一种用图形表示的形式化建模语言,通常用于软件工程的系统建模[19].作为一个主流软件测试技术,TTCN 采用行为树描述被测系统的行为.如图 1 所示,左侧是行为树结构,表示通过控制观察点(point of control and observation,简称 PCO)观测到的 TTCN 测试套与被测系统之间的交互过程.

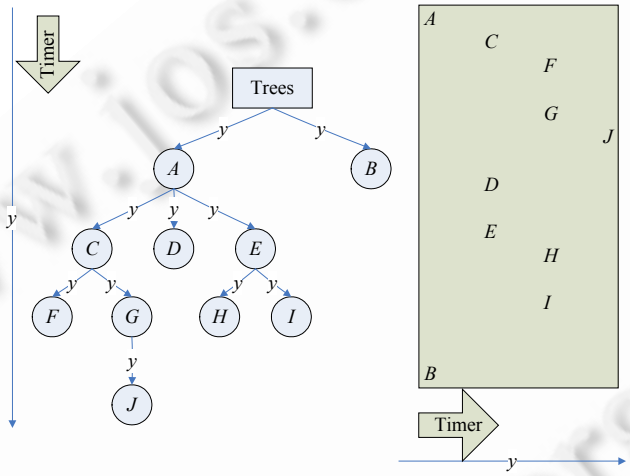

Fig.1 TTCN behavior trees

图 1 TTCN 行为树

所有同一棵子树的姊妹节点表示一个可替换的行为,如图 1 所示,各个替换集为{*A*,*B*},{*C*,*D*,*E*},{*F*,*G*},{*H*,*I*},{*J*},但不在同一棵子树的节点不属于同一个替换集合.下面给出行为树和实例的形式化定义.

定义 3. 一个 TTCN 行为树是一个三元组 $T=(N,\Phi,r)$,其中,

1) N 是有限行为节点 $n_1,n_2,\ldots,n_m$ 的集合,节点不一定唯一,因为同一个行为可以发生在不同的情景下.
2) r 是一个可区分的节点,称为根.
3) $\Phi:N\to N$ 是一个函数,对任意的一个节点 $n_0\in N$,一定有一个节点序列 $n_1,n_2,\ldots,n_k(0\leqslant k\leqslant m)$,称其为 n_0 的孩子节点,并称 $n_1,n_2,\ldots,n_k$ 是兄弟节点,或者为同一个替换集合的节点.
4) 每一个节点 n_i 是一个三元组(*ID*,*Beha*,*Qual*),其中,*ID* 是节点标示,*Beha* 表示节点上的操作,*Qual* 表示操作的约束.

下面给出行为树实例的概念.

定义 4. 设 $T=(N,\Phi,r)$是行为树,称 $B\subseteq Beha\times Qual$ 是一个行为类,其中,$Beha=\{Beha_1,Beha_2,\ldots,Beha_{m-1},Beha_m\}$, $Qual=\{Qual_1,Qual_2,\ldots,Qual_{m-1},Qual_m\}$.特别地,如果 $Beha\times Qual$ 用 TTCN 编码,则称 $B(BT)$是行为树的一个实例,记为 $BI(BT)$.

例如,假设 $Beha=\{L!N\text{-}DATArequest,\{L?N\text{-}DATAindication,L?OTHERwise\}\}$,$Qual=\{pass,fail,none\}$.一个行为树实例为 $\{\langle L!N\text{-}DATArequest,pass\rangle,\langle L?N\text{-}DATAindication,pass\rangle\}$,或者 $\{\langle L!N\text{-}DATArequest,pass\rangle;\langle L?OTHERwise,none\rangle\}$.

2 基于 LTS 的 TTCN-3 测试套生成算法

本节讨论 TTCN 测试套生成算法.首先给出标签转换系统 LTS 到行为树 BT 的等价性证明.

2.1 LTS到BT的等价变换

首先论证 LTS 与 BT 的等价性,然后给出 LTS 到 BT 的转化算法.

定理 1. 设 $LTS=(S,I,\rightarrow,\vDash)$是一个标签转换系统,二元关系$\vDash$用$\mu$-calculus 编码,则一定存在一个从 LTS 实例到 BT 实例的一一映射.

证明:建立一一映射如下:

建立一个 BT 的根节点"r",让其与 LI 中的初始节点"i"对应.对$\forall l\in LI(LTS)$的所有标签做如下映射和演绎:

- 如果 $l=a(a\in A)$,则映射 $L(LTS)\subseteq A\times S\times P=\{\langle a,s,p\rangle|\ t\xrightarrow{a}s\}$到$\{\langle a,p\rangle s\}$,其中,让逻辑值{True,False}分别映射成行为裁决$\{pass,fail\}$;
- 如果 $l=R_1$ "." R_2,则映射 $L(LTS)\subseteq A\times S\times P=\{\langle l,s,p\rangle|\ t\xrightarrow{R_1\cdot R_2}s\}$到$\{\langle R_1,p_1\rangle s_1,\{\langle R_2,p_2\rangle s_2\}\}$,其中,$s_2$ 是 s_1 的子树,$s_1=s,p=p_1\wedge p_2$;
- 如果 $l=R_1$ "|" R_2,则映射 $L(LTS)\subseteq A\times S\times P=\{\langle l,s,p\rangle|\ t\xrightarrow{R_1|R_2}s\}$到$\{\{\langle R_1,p_1\rangle s_1\},\{\langle R_2,p_2\rangle s_2\}\}$,其中,$s_1$ 和 s_2 属于同一个替换集,s_1 和 s_2 是 S 的子树,$p=p_1\vee p_2$;
- 如果 $l=R_1$ "*" R_2,则映射 $L(LTS)\subseteq A\times S\times P=\{\langle t,s,p\rangle|\ t\xrightarrow{R_1*R_2}s\}$到$\{\{\langle R_1,p_1\rangle s_1\}^*,\{\langle R_2,p_2\rangle s_2\}\}$,其中,$\{\langle R_1,p_1\rangle s_1\}^*$表示至少重复 1 次或者无重复,$s_2$ 是 s_1 的子树,$s_1=s,p=p_1\wedge p_2$;
- 如果 $l=R_1$ "+" R_2,则映射 $L(LTS)\subseteq A\times S\times P=\{\langle l,s,p\rangle|\ t\xrightarrow{R_1+R_2}s\}$到$\{\{\langle R_1,p_1\rangle s_1\}^+,\{\langle R_2,p_2\rangle s_2\}\}$,其中,+表示$\{\langle R_1,p_1\rangle s_1\}^+$至少重复 1 次,$s_2$ 是 s_1 的子树,$s_1=s,p=p_1\wedge p_2$.

因此,定理成立.

定理 2. 令 $BT=(N,\Phi,r)$是一个行为树,BI 是它的行为实例,则一定存在一个 BI 到 LI 的一一映射,使得 LI 中的$\vDash$可以用μ-calculus 编码.

证明:根据定理 1 证明中建立的一一映射可知,可以找到一个逆映射,使得 BI 映射到 LI 也是一一映射,并且 LI 可以用μ-calculus 编码.证毕. □

根据定理 1 和定理 2 可知,LTS 与 BT 等价.

2.2 TTCN抽象测试套生成算法

本节给出 TTCN 抽象测试套生成算法.算法分 3 个部分:

(1) 编辑 LI,建立 LTS 描述;

(2) 根据 $LI(LTS)$与 $BI(BT)$的等价关系,生成 TTCN 抽象测试套框架;

(3) 根据 LI 二元关系$\vDash$的μ-calculus 规格说明,提取被测系统测试用例.

算法 1. TTCN 抽象测试套生成算法.

(1) 图形输入 LTS,编辑 LTS 约束

(2) 生成 TTCN 抽象测试套框架

Input: $LI(LTS)$

Output: $BI(BT)$

Begin

Generate a root node, let $r \in BI$

Read a element of LI to record $R=\langle l,s,p\rangle$

For each R Do

If $l=a$ then rewrite $\langle a,s,p\rangle$ to $\langle a,p\rangle_s$ replace True with pass and False with fail.

If $l=R_1$ "." R_2, then rewrite $\langle R_1.R_2,s,p\rangle$ to $\{\langle R_1,p_1\rangle_{s_1},\{\langle R_2,p_2\rangle_{s_2\subseteq s_1}\}\}$

If $l=R_1$ "|" R_2, then rewrite $\langle R_1|R_2,s,p\rangle$ to $\{\{\langle R_1,p_1\rangle_{s_1\subseteq s}\},\{\langle R_2,p_2\rangle_{s_2\subseteq s}\}\}$

If $l=R_1$ "*" R_2, then rewrite $\langle R_1*R_2,s,p\rangle$ to $\{\{\langle R_1,p_1\rangle_{s_1}\}^*,\{\langle R_2,p_2\rangle_{s_2\subseteq s_1}\}\}$

If $l=R_1$ "+" R_2, then rewrite $\langle R_1+R_2,s,p\rangle$ to $\{\{\langle R_1,p_1\rangle_{s_1}\}^+,\{\langle R_2,p_2\rangle_{s_2\subseteq s_1}\}\}$

end

end

(3) 测试用例生成

Input: $BI(BT)$

Output: Set of Test Case

Begin

Read a element of BI to record $N_i=\{\langle R_i,p_i\rangle s_i\}$

For each N_i Do

If $R_i=a$ then $\langle a,p_i\rangle \in TC$

If $R_i=R_1$ "." R_2, then $\langle R_1,p_1\rangle \in TC$ and $\langle R_2,p_2\rangle \in TC$

If $R_i=R_1$ "|" R_2, then $\langle R_1,p_1\rangle \in TC$ and $\langle R_2,p_2\rangle \in TC$

If $R_i=R_1$ "*" R_2, then $\langle R_1,p_1\rangle\} \in TC$ and $\langle R_2,p_2\rangle \in TC$ //去除重复的测试用例

If $R_i=R_1$ "+" R_2, then $\langle R_1,p_1\rangle \in TC$ and $\langle R_2,p_2\rangle \in TC$ //去除重复的测试用例

end

end

算法的第 1 部分可以通过图形编辑方式输入软件系统运行状态描述的 LTS 图.第 2 部分生成 TTCN 抽象测试套,用替换集表示 BT.为了明确表示替换子集合,采用集合包含符号加以区分.第 3 部分用$\langle R_i,p_i\rangle s_i$ 表示测试用例.其中,R_i 是测试输入;p_i 是设计输出,用μ-calculus 的状态公式来表示.按照算法的设计,测试用例可以实现基本路径覆盖.算法的第 2 部分、第 3 部分可以同时执行,因此,该算法的时间复杂性应该为 $O(\log n)$,n 为 $LI(LTS)$ 节点个数.

例:一个通信系统的 LTS 如图 2 所示.依照算法 2,其 BT 和测试用例求解如下:

建立的一一映射,二者之间的对应关系为

$LI(LTS)$={⟨*Start*,*Initial*,True⟩,⟨*L*?*N-DATAindication*,2,True⟩,

⟨*L*?*OTHERwise*,3,False⟩,⟨*L*!*DATA*,4,True⟩*,⟨*L*!*Disconnect*,5,False⟩}.

用μ-calculus 编码 $LI(LTS)$得到:

{⟨*L*!*DATArequest**.*L*?*N-DATAindication*.*L*!*DATA*⟩True,⟨*L*!*DATArequest**.*L*?*OTHERwise*⟩False}.

编码后的 LI 经过一一映射得到 BI 如下:

$Map(LI)=BI=\{r,\{L!DATArequest,psaa\}_1\},\{\{L?N\text{-}DATAindication,pass\}_2,\{L?OTHERwise,Fail\}_3\},$

$\{\{L!DATA,pass\}_4\}$.

上述算法为实现 TTCN 测试套自动生成提供了支撑.然而,TTCN 测试套仅仅给出了测试执行的逻辑步骤,在测试执行前,还需要针对被测实现进行参数配置,通常把这些工作称为 TTCN 适配器实现.一般有两类适配器:平台适配器和被测系统适配器,前者实现对 TTCN 运行平台的支持,后者为被测系统配置具体通信方式.下面给出嵌入式系统的适配策略和物联网识读器 API 函数测试适配器实现的方法和技术.

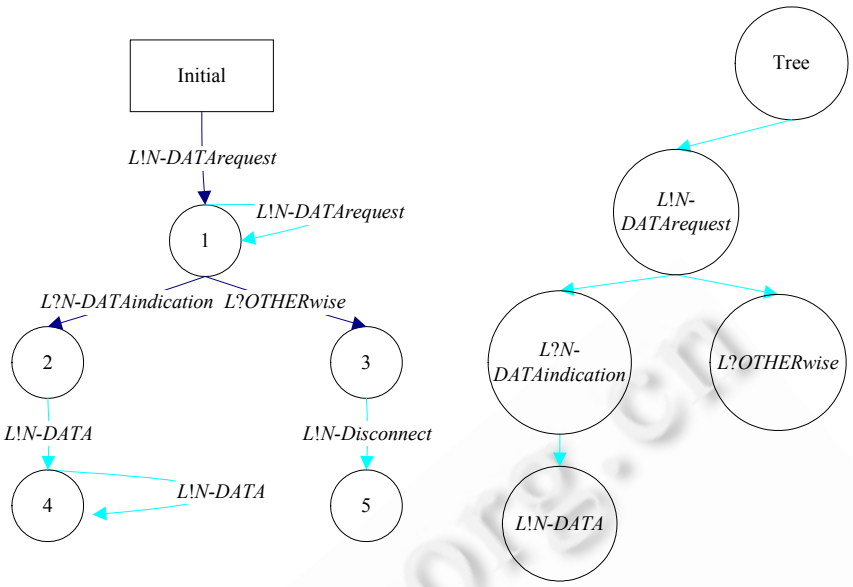

Fig.2 LTS and BT for communication dialog

图 2 一个通信系统的 LTS 和 BT

3 可伸缩的嵌入式系统适配器实现技术

作为一个国际标准,ISO/ETSI 并没有给出 TTCN 适配器的具体实现环节,也没有具体限制平台的结构.本节讨论嵌入式系统软件测试适配器的实现技术.

3.1 TTCN适配器的结构

按照 ISO/ETSI 定义的 TTCN 规范,TTCN 测试套需要通过被测系统(system under test,简称 SUT)适配器和平台适配器(platform adapter,简称 PA)来实现测试套与被测系统的通信,如图 3 中实线部分所示.

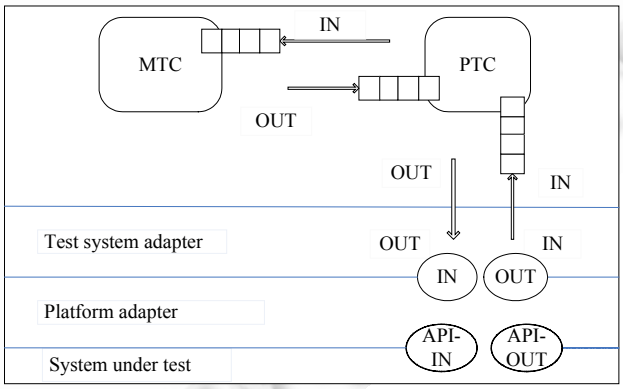

Fig.3 Architecture of TTCN system

图 3 TTCN-3 系统体系结构

被测系统适配器的主要任务是完成测试组件端口与被测试系统端口的连接,通过测试组件端口向被测试系统端口发送测试用例,并接收被测系统的反馈.平台适配器是将测试系统适配到网络平台上,同时实现测试系统对被测系统外部函数的同步调用.PA 中的一个重要的部件就是时钟,由它来协调和控制测试套与被测系统的

通信过程.图 3 中的主测试组件 MTC(master testing component,简称 MTC)可以同时协调和控制多个并行测试组件 PTC(parallel testing component,简称 PTC).适配器结构如图 3 所示.

3.2 嵌入式系统软件测试适配器的设计

为了支持二次开发,嵌入式系统软件一般暴露可供调用的 API 函数,物联物识读器也为开发商提供了嵌入式 API 函数.按照 EPC Global 技术标准[20],EPC 识读器 API 函数应该满足如下基本要求:

- 函数名与参数应该与相关标准一致;
- 函数完成的功能应该与标准一致;
- 函数执行效能虽然与平台有关,但应符合标准的基本要求;
- API 函数可以包含非标准规定函数集合,但与标准规定的函数集的交不能为空.

上述基本要求也构成了嵌入式系统软件的测试目的.为了检验嵌入式系统的可靠性和依照标准的一致性,需要逐一对 API 进行验证.一个具有挑战性的工作是:如何设计嵌入式系统适配器,使其对一类基于标准的 API 函数进行自动适配.为此,设计了如图 3 所示(虚线部分)的嵌入式系统测试适配器.一个关键技术是:在原来的测试系统结构中增加了两个反嵌入式接口,把嵌入式系统软件的 API 插入平台适配器中,当测试系统通过调用 API 函数进行测试进程时,可以直接驱动嵌入式系统软件,并反馈测试结果给测试系统.下节将通过一个案例解释该项技术的具体应用和实现.

4 工具开发与实验分析

本节结合美国 Impinj 公司研发的第二代超高频 RFID 读写器 UQC-R420(A),介绍嵌入式软件测试套测试工具开发与实验测试的分析过程,进一步支持提出方法和技术.

4.1 物联网识读器嵌入式软件抽象测试套的生成

为了方便 TTCN 抽象测试套设计,ISO/ETSI 在发布 TTCN 核心语言的同时,也给出了与核心语言等价的其他表现形式,其中,GFT(graphical presentation format)图就是 TTCN 的一种图形化表示,与核心语言的对应关系可以在文献[14]中查到,这里不再赘述.比较 LTS 图和 GFT 图,二者之间的等价映射十分简单,而且其逻辑等价性可以根据第 1 节的论证以及 ISO/ETSI 标准间接得出,所以,实际开发 TTCN 抽象测试套的过程中可以采取下面的步骤:

1) 建立 LTS 到 GFT 的映射;
2) 建立 GFT 图到 TTCN 抽象测试套的映射;
3) 提取 LTS 状态公式描述的规格说明,并用 GFT 图的消息序列表述;
4) 提取消息序列,产生测试用例.

经过多年的努力,北方工业大学软件工程研究室开发了一款基于 ISO/ETSI 的 TTCN-3 测试工具 TTRun1.1.该工具包括:

- 编辑环境,可实现 GFT 图和 TTCN 核心语言文本的编辑;
- TTCN 核心语言编译器,可以实现 GFT 到 TTCN 生成,并把 TTCN 核心语言程序编译成 Java 语言执行代码(编译器根据开源代码改写);
- TTCN 核心语言适配器库,用于管理各种测试专用适配器.

按照 EPC Global 标准[20],UQC-R420(A)嵌入了所有 LLRP(low level reader protocol)协议规定的识读器管理软件模块,并通过 API 函数提供二次开发调用接口.图 4 是物联网识读器的 LTS,表现了识读器在接到外部函数调用后,其工作状态的迁移.它共有 4 个状态,分别是 CONNECT,DISABLE,INACTIVE,ACTIVE.图 5 是与之对应的 GFT 图编辑器截图,其中,LTS 图中的标签对应于 GFT 图的消息和消息箭头;LTS 图中的状态节点对应 GFT 图中的控制观察点的消息队列,并按照时间轴依次排列.

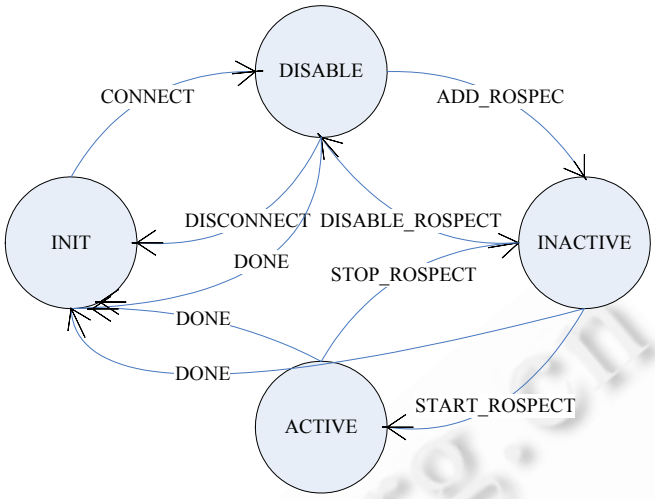

Fig.4 An LTS of IoT reader system

图 4 物联网识读器系统的 LTS

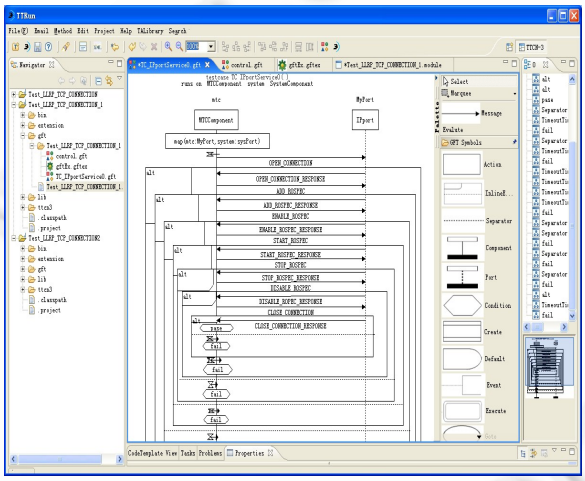

Fig.5 Interface copy of GFT model

图 5 GFT 模型界面

4.2 物联网识读器嵌入式软件测试适配器的实现

TTRun1.1 设计实现了一个适配器库,用于管理各类适配器的配置、维护以及测试用例的编码/解码.针对物联网识读器嵌入式软件 API 测试的特点,开发了一个专用的适配器,采用第 2 节中的适配器结构设计思想,把被测系统的 API 函数反嵌入适配器中,嵌入的 API 函数有:

demo.connect(“*speedwayr*-10-61-*c*0.*local*”);

demo.enableImpinjExtensions(·);

demo.factoryDefault(·);

demo.setReaderConfiguration(·);

demo.deleteAccessSpecs(·);

demo.getReaderConfiguration(·);

demo.addRoSpec(·);
demo.enableRoSpec(·);
demo.startRoSpec(·);
demo.stopRoSpec(·);
demo.disableRoSpec(·);
demo.deleteRoSpec(·);
demo.disconnect(·);

上述 API 函数通过 import 语句自动加载到适配器中,再经过环境参数配置,如 IP 地址等,即可实现识读器被测 API 函数的调用,从而完成测试任务.

4.3 测试实验结果分析

针对 UQC-R420(A)识读器设计了两个测试实验,分别测试识读器的识读率和状态切换的可靠性.

实验 1.

- 测试目的:测试识读器软件 ACTIVE 状态下对 EPC 标签的识度率.
- 测试方法:选择部分识读器,分别对多个 EPC 标签的识读率进行测试.为了得到大样本数据,每个实验重复 10 次(表 1 仅列出 5 次),根据每次实验所读取的标签个数统计每个识读器的识度率.
- 测试输入:30 个 EPC 标签,4 个识读器.
- 测试工具:TTRun1.1.
- 测试输出:表 1 和图 6.

Table 1 Statistic table of reading rate of UQC-R420

表 1 UQC-R420 识读器识度率统计表

Reader	T1	T2	T3	T4	T5
Reader1	26	25	25	25	25
Reader2	25	25	25	25	25
Reader3	25	26	26	25	26
Reader4	24	27	26	26	26

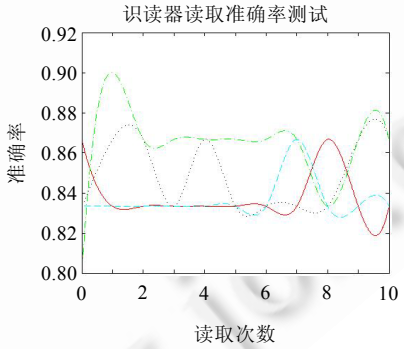

Fig.6 A statistic chart of reading rate

图 6 识读器识读率统计图

- 测试结论:4 个被测识读器系统均发生漏读故障,平均识读率约为 80%.
- 故障发现:对被测识读器进行冗余测试,即,一个识读器同时附加多个天线对 EPC 标签进行识读操作,并在 EPC 标签的上、左、右分别放置天线,这时,识读器的识读率接近 100%.因此,漏读故障归结为识读过程中多标签重叠、相互屏蔽干扰所致,并非软件故障.该故障可以通过适当增加天线数来解决.

实验 2.

- 测试目的:测试识读器工作状态 CONNECT,DISABLE,INACTIVE 和 ACTIVE 的可靠性.
- 测试方法:选择部分识读器,分别对其进行状态驱动测试,通过连续切换状态 20 次(表中仅列出 5 次),获得状态转换的响应时间,反馈状态的可靠性.
- 测试输入:CONNECT,DISCONNECT,ADD_ROSPECT,DISABLE_ROEPEC,START_ROSPECT,STOP_ROSPEC 等消息对象测试用例.
- 测试工具:TTRun1.1.
- 测试输出:表 2 和图 7.

Table 2 Feedback time of UQC-R420 reader

表 2 UQC-R420 识读器响应时间

Reader	T1	T2	T3	T4	T5
Reader1	687	297	281	282	297
Reader2	596	282	282	281	281
Reader3	565	297	547	281	297
Reader4	596	282	297	281	297

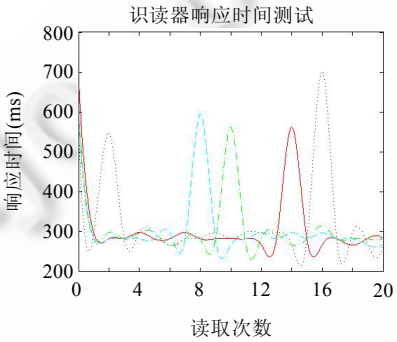

Fig.7 Feedback time of UQC-R420 reader

图 7 识读器 UQC-R420 响应时间

- 测试结论:4 个识读器表现出不一致的工作状态,启动的响应时间不一致.
- 故障发现:在测试过程中,采用网络流量监控机制观察网络状态变化,发现被测识读器的上述功能与网络状态变化有密切关系;当网络状态不稳定时,会造成输出响应时间不一致的现象.

5 相关研究比较与结论

随着计算机嵌入式系统应用领域的不断扩展,嵌入式系统软件质量保障方法和技术遇到了新的挑战,同时也得到了业界的关注,近期的相关研究也体现得比较明显,主要集中在嵌入软件测试用例生成、模拟测试方法和嵌入式软件代码分析等.

嵌入式软件测试用例生成的研究主要还是沿用传统软件测试方法:首先建立被测系统软件模型,再把模型状态的迁移条件作为测试用例.殷永峰等人[21]针对嵌入式系统软件在实时性和反应式等方面的特点,采用 UML 分别建立嵌入式软件的静态和动态测试模型,并基于 UML 图提出了嵌入式软件测试用例生成算法.杨广华等人[22]提出了一种将场景和模式方法用于嵌入式软件测试用例的设计与生成的方法,通过对被测软件系统需求进行分析建模,将建立的场景模型划分到不同的场景模式中,依据场景模式构建测试场景的状态图,遍历场景状态图以获取测试执行路径,确定相关的测试数据,设计并生成测试用例.Conrad 等人[23]结合基于模型的开发方法讨论如何产生测试用例,借助分类树的概念,在逻辑层面辅助测试脚本的生成;再把生成的测试脚本映射到实际

执行测试中,从而实现从逻辑设计到物理执行的测试过程,文中通过大量实际例子支持测试脚本生成方法.Malte Lochau,Ursula Goltz[24]针对嵌入式软件中人(环境)与设备交互时容易产生错误的现象,建立了交互特性模型——特性网络(feature network),在充分肯定基于模型测试方法在嵌入式软件测试中的作用基础上,提出了基于特性模型的嵌入式软件测试用例生产算法,有效地控制了测试用例的数量,为嵌入式系统测试降低了测试代价.

与传统的白箱测试方法相似,当已知嵌入式系统软件代码的逻辑结构,同时在嵌入式软件系统资源允许的情况下,可以采用插桩的方法,即,通过在源代码中嵌入一段侦听程序来获得嵌入式系统的运行状态信息,从而分析软件系统的可靠性、一致性和性能等特性.程晓菊等人[25]提出了一种基于函数切片的嵌入式软件回归测试用例简约方法,基于函数代码影响域的概念,建立影响域相关的函数依赖图(函数切片),并给出了全局函数依赖到函数切片的转换算法.Hsiung 等人[3]提出了一个嵌入式系统验证框架(verifiable embedded real-time application framework,简称 VERTAF),生成宿主的模型检验程序代码,实时完成系统的检测任务,框架中采用 UML 中的类图、时序图和扩展序列作为嵌入式系统的模型描述,VERTAF 根据上述描述产生模型检验器的代码.Ganesan 等人[4]提出了一个嵌入式系统插桩框架,其主要特点是可以在嵌入式软件开发过程中对代码进行实时的单元测试,从而及早发现代码的缺陷.把该嵌入式系统插桩框架应用于 NASA 飞机软件生产线,取得了良好的效果.Chae 等人[10]发现,随着嵌入式系统的广泛使用,开发实时编译器成为一个不断增长的需求,重定位技术成为嵌入式系统程序编译器的主流技术,然而在执行代码检测时,通常的做法是以源语言代码作为测试用例产生的依据,从而造成测试套冗长.因此,他们提出了一种基于中间代码的测试用例生成方法,可以有效地减少测试用例的数量.Seo 等人[11]针对嵌入式系统的限制,开发了一个轻型测试模型,并实现了相应的工具,利用该模型,可以有效地采集测试数据,为分析和评价系统性能提出支持.Suri 等人[26]针对嵌入式系统中资源限制、软件功能对资源需求过载等特点,提出了一种容错框架,通过区分功能关键等级,屏蔽次要功能的错误,保证关键功能的执行,从而保证系统的可靠性等级.

与上述研究工作不同,本文主要研究测试套自动生成方法以及利用该方法解决嵌入式软件 API 函数自动测试问题.虽然测试套包含测试用例选择内容,但本文的重点在测试逻辑执行部分,而在测试用例的选择方面,采用黑箱测试法中的边界值方法,对测试用例选择没有做过多的讨论.案例介绍了美国 Impinj 公司生产的 UQC-R420(A)超高频 RFID 读写器 API 函数的可靠性测试工作,虽然该 API 函数设计符合 EPC Global 规范,但实现代码并不开放,所以本文也没有涉及代码插桩方法方面的内容,与上述同类研究的内容也有很大的区别.

本文提出了 LTS 到行为树转换的理论方法,为 TTCN 抽象测试套生成算法的提出奠定了理论基础.本文还提出了嵌入式系统 TTCN 测试套自动生成的技术方案,并实际检验了该方案的可行性.该方案的具体步骤为:

1) 根据 LTS 自动生成嵌入式系统的 TTCN 抽象测试套,该测试套包含了测试步骤和测试指令.本文第 2.2 节中提出的算法阐述了该技术内容.
2) 根据 L(LTS)类实例 *LI* 中的μ-calculus 状态公式编码,自动提取测试例生成.本文第 2.2 节提出的算法的第 2 部分给出了技术细节.
3) 测试环境适配器的设计,实现抽象测试套与被测系统的通信和测试监控.本文第 3 节给出了完整的技术实现细节介绍.

由于存在实现技术细节的挑战,所以目前测试套自动生成的方法和技术研究不多见.本文尝试把测试套分成抽象测试套、测试用例和测试适配器这 3 个步骤来实现,解决了测试套自动生成中技术细节不易实现的难题,对业界有参考价值,对学术界有贡献.

本文仅对 LLRP 协议实现了物联网识读器测试适配器的通用设计,还不能实现其他嵌入式系统测试的适配工作,为此,我们将在抽象测试套适配器方法和技术方面进一步开展研究工作.

References:

[1] Yin YF, Wang YC, Liu B. Design and implementation of embedded software simulation test script language. Computer Engineering and Design, 2006,27(12):2130−2132 (in Chinese with English abstract).

[2] Jiang CW, Yang SK, Liu B. Simulation and modeling for embedded software test. Computer Engineering, 2008,34(4):87−89 (in Chinese with English abstract).

[3] Hsiung PA, Lin SW. Automatic synthesis and verification of real-time embedded software for mobile and ubiquitous systems. Computer Languages, Systems & Structures, 2008,34:153−169. [doi: 10.1016/j.cl.2007.06.002]

[4] Ganesana D, Lindvall M, McComas D, Bartholomew M, Slegel S, Medina B, Krikhaar R, Verhoef C, Montgomery LP. An analysis of unit tests of a flight software product line. Science of Computer Programming, 2012. http://www.eisevier.com/com/locate/scico [doi: 10.1016/j.scico.2012.02.006]

[5] Amman P, Offutt J. Introduction to Software Testing. Beijing: China Machine Press, 2010. 1−20.

[6] Wang LZ, Li XD, Zheng GL. An approach to generate integration test cases based on UML collaboration diagrams. Acta Electronica Sinica, 2004,32(8):1290−1296 (in Chinese with English abstract).

[7] Zhang ZY, Chen ZY, Xu BW, Yang R. Research progress on test case evolution. Ruan Jian Xue Bao/Journal of Software, 2013, 24(4):663−674 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4379.htm [doi: 10.3724/SP.J.1001.2013.04379]

[8] Nie CH, Jiang J. Optimization of configurable greedy algorithm for covering arrays generation. Ruan Jian Xue Bao/Journal of Software, 2013,24(7):1469−1483 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4326.htm [doi: 10.3724/SP.J.1001.2013.04326]

[9] Fraser G, Wotawa F, Ammann P. Issues in using model checkers for test case generation. Journal of Systems and Software, 2009, 82:1403−1418. [doi: 10.1016/j.jss.2009.05.016]

[10] Chae HS, Woo G, Kim TY, Bae JH, Kim WY. An automated approach to reducing test suites for testing retargeted C compilers for embedded systems. The Journal of Systems and Software, 2011,84:2053−2064. [doi: 10.1016/j.jss.2011.04.023]

[11] Jooyoung Seo, Byoungju Choi, Sueng-wan Yang. Lightweight embedded software performance analysis method by kernel hack and its industrial field study. The Journal of Systems and Software, 2012,85:28−42. [doi: 10.1016/j.jss.2011.03.049]

[12] Kahlouche H, Viho C, Zendri M. An industrial experiment in automatic generation of executable test suites for a cache coherency protocol. In: Proc. of the Int'l Workshop on Testing of Communicating Systems (IWTCS'98). Chapman and Hall, 1998. 1−8.

[13] Utting M, Legeard B. Practical Model-Based Testing: A Tools Approach. San Francisco: Morgan Kaufmann Publishers, 2010. 1−456.

[14] ETSI Standard. ETSI ES 201 873-1 V2.2.1-TTCN-3. 2003. http://www.ttcn-3.org/doc/es_20187301v020201p.doc

[15] Tretmans J. Model based testing with labeled transition system, formal methods and testing. Lecture Notes in Computer Science, 2008,4949:1−38. [doi: 10.1007/978-3-540-78917-8_1]

[16] van Glabbeek RJ. Bisimulation. 1993. http://www.cse.unsw.edu.au/~rvg/pub/Bisimulation.pdf

[17] Mateescu R. Local model-checking of modal mu-calculus on acyclic labeled transition systems. In: Proc. of the 8th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2002). Springer-Verlag, 2002. 1−36.

[18] Antonik A, Huth M. Efficient patterns for model checking partial state spaces in CTL∩LTL. Electronic Note in Theoretical Computer Science, 2006,58:41−57. [doi: 10.1016/j.entcs.2006.04.004]

[19] Colvin R, Grunske L, Winter K. Probabilistic timed behavior trees, in integrated formal methods. In: Proc. of the 6th Int'l Conf. (IFM 2007). LNCS 4591, Oxford): Springer-Verlag, 2007. 157−175. [doi: 10.1007/978-3-540-73210-5_9]

[20] EPCglobal. EPC tag data standards. Version 1.1, Rev.1.24, EPCglobal Standard Specification, 2004. http://www.epcglobalinc.org/standards_technology/EPCTagDataSpecification.124.pdf

[21] Yin YF, Liu B, Jiang TM. Test cases generation of embedded software testing based on UML technique. Application Research of Computers, 2008,25(10):3018−3021 (in Chinese with English abstract).

[22] Yang GH, Qi X, Shi YS. Design of embedded software test cases based on scenario pattern. Computer Engineering, 2010,36(15): 89−91 (in Chinese with English abstract).

[23] Conrad M, Fey I, Sadeghipour S. Systematic model-based testing of embedded automotive software. Electronic Notes in Theoretical Computer Science, 2005,111:13−26. [doi: 10.1016/j.entcs.2004.12.005]

[24] Lochau M, Goltz U. Feature interaction aware test case generation for embedded control systems. Electronic Notes in Theoretical Computer Science, 2010,264:37−52. [doi: 10.1016/j.entcs.2010.12.013]

[25] Cheng XJ, Li RF. Study of embedded software regression test based on function slice. Computer Engineering, 2012,38(2):54−56 (in Chinese with English abstract).

[26] Suri N, Jhumka A, Hiller M, Pataricza A, Islam S, Sârbu C. A software integration approach for designing and assessing dependable embedded systems. The Journal of Systems and Software, 2010,83:1780−1800. [doi: 10.1016/j.jss.2010.04.063]

附中文参考文献:

[1] 殷永峰,王轶辰,刘斌.嵌入式软件仿真测试脚本语言的设计与实现.计算机工程与设计,2006,27(12):2130−2132.

[2] 蒋崇武,杨顺昆,刘斌.面向嵌入式软件测试的仿真建模.计算机工程,2008,34(4):87−89.

[6] 王林章,李宣东,郑国梁.一个基于 UML 动作图的集成测试用例生成方法.电子学报,2004,32(8):1290−1296.

[7] 张智轶,陈振宇,徐宝文,杨瑞.测试用例演化研究进展.软件学报,2013,24(4):663−674. http://www.jos.org.cn/1000-9825/4379.htm [doi: 10.3724/SP.J.1001.2013.04379]

[8] 聂长海,蒋静.覆盖表生成的可配置贪心算法优化.软件学报,2013,24(7):1469−1483. http://www.jos.org.cn/1000-9825/4326.htm [doi: 10.3724/SP.J.1001.2013.04326]

[21] 殷永峰,刘斌,姜同敏.基于 UML 的嵌入式软件测试用例生成方法研究.计算机应用研究,2008,25(10):3018−3021.

[22] 杨广华,齐璇,施寅生.基于场景模式的嵌入式软件测试用例设计.计算机工程,2010,36(15):89−91.

[25] 程晓菊,李仁发.基于函数切片的嵌入式软件回归测试研究.计算机工程,2012,38(2):54−56.

赵会群(1960－),男,辽宁沈阳人,博士,教授,CCF 高级会员,主要研究领域为软件工程,物联网.
E-mail: zhaohq6625@sina.com

张爆(1990－),男,硕士,主要研究领域为软件测试.
E-mail: 373169359@qq.com

孙晶(1968－),女,副教授,主要研究领域为软件工程.
E-mail: sunjing8248@126.com

王同林(1990－),男,硕士,主要研究领域为物联网.
E-mail: wangtonglin54@126.com