# 软件模型代数性质的程序化验证

赵会群 黄榆涵
（北方工业大学计算机学院 北京 100144）

**摘 要** 软件模型代数的思想是通过引入进程代数来对软件体系结构进行建模。它将构件解释为变量，将连接子抽象为代数运算，并针对软件的特性建立了软件体系结构代数模型。在代数模型的基础上，讨论分析获得一系列能指导软件演化的代数性质。但是，上述研究都只对模型的代数性质进行了理论证明，实际上并无程序能够证明这些代数性质的正确性，同时也未给出这些性质的应用方法，使其缺乏可操作性。采用程序化验证的方法对代数性质进行了验证，并对这些性质的应用算法进行了研究，进一步丰富了软件的建模理论，也使得软件演化从理论研究转化为实际应用成为可能。

**关键词** 代数模型，代数性质，程序化验证，软件演化
**中图法分类号** TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.11.036

## Program Verification of Software Model's Algebraic Properties

ZHAO Hui-qun HUANG Yu-han
(Department of Computer Science, North China University of Technology, Beijing 100144, China)

**Abstract** Software model algebra is introduced to model software architecture by process algebra. It explains components as variables, abstracts connector as algebra operation, and builds the algebraic model of software architecture according to software features. Based on the discussion about algebraic model, we got a series of algebraic properties to guide software evolution. However, these studies have only theoretically proved the algebraic properties of the mode, no program can actually prove the correctness of these algebraic properties, and they did not give the property applications, making them lack maneuverability. This paper researched the algebraic properties of algorithms, did program verification, further enriched the software model, and also made it possible to change software evolution from theory into practical applications.

**Keywords** Algebraic model, Algebraic property, Program verification, Software evolution

## 1 引言

近年来，随着软件规模的不断扩大及软件复杂度的逐渐增加，人们逐步意识到软件体系结构的重要性。软件体系结构是指系统的各种结构，包含软件构件、这些构件的外部可见特性以及构件之间的相互关系[1]。而研究软件体系结构的一个首要内容是如何描述软件系统的体系结构。工程中常用直观、半形式化的图形建模方法来对软件体系结构进行描述[2]，这些方法较直观，便于开发人员之间的理解和交流[3]。但这些描述语言不能精确地描述系统，难以对其所表示的模型进行分析[1]，因此基于进程代数进行研究为软件体系结构的描述提供了一种新的形式化描述语言，使其能够精确、无二义性地描述系统。

基于进程代数的研究都运用代数的方法对软件体系结构进行建模，提出了代数模型，归纳了代数性质。然而这些研究都只对模型的代数性质进行了理论证明，实际上并无程序来证明这些代数性质的正确性，使其缺乏可操作性。本文将通过程序化验证的方法来证明代数性质在低层结构同样成立，即代码经过结构变化之后，系统功能与变化前一致。本文按照代数性质对C语言代码、BPEL代码、伪代码进行转化，以得到演化后的代码；并对演化前后的代码进行验证及分析，以进一步证明性质的正确性。本文提出了性质的应用算法，增加了代数性质的实用性，丰富了软件的建模理论。

袁博等[4]采用代数学方法对可重构构件、构件组合、可重构系统的属性和行为特征进行抽象，将构件组合定义成构件的“运算”实现，结合进程代数中算子的概念，定义了多种构件组合运算，建立了可重构系统的代数模型。张友生等[5]设计了一种基于代数理论的软件体系结构的描述方法，使用该方法描述了软件体系结构及其元素和构件的运算表达式，研究了构件运算表达式转化为文档的算法，实现了构件运算表达式转化为文档的系统原型，从而使软件演化由理论研究转化为实际应用成为可能。代飞等[6]针对软件演化过程元模型，引入进程代数对其进行扩展，提出软件演化过程元模型代数；使用进程项指定软件演化过程模型的代数语义，在进程代数

到稿日期：2016-10-09 返修日期：2016-12-23 本文受国家自然科学基金(61370051)资助。
**赵会群**(1960－)，男，博士，教授，主要研究方向为软件工程和可信软件；**黄榆涵**(1993－)，女，硕士，主要研究方向为软件演化，E-mail：316057215@qq.com(通信作者)。

的统一框架下，基于等式推理验证软件演化过程模型的行为，使行为验证方式从模型推导变为代数推导，可以有效地支持软件演化过程的形式验证。赵会群等[7-8]基于进程代数的软件体系结构建模方法，建立了软件的代数模型，使其能够在更抽象的层面上讨论体系结构模型及其性质，使得模型的概括能力得到增强，从而可以指导一类软件设计与分析中的问题。

本文第 1 节概述了代数模型和代数性质；第 2 节结合案例验证了代数性质的正确性；第 3 节给出了代数性质的应用方法，提出了符合性质条件的转化算法；最后总结全文并展望未来。

## 2　软件模型代数性质简介

### 2.1　代数模型

本文的代数模型建立在文献[7]提出的网构软件体系结构代数模型的相关概念及定义的基础之上。通过将构件解释为变量，将连接子抽象为代数运算，并结合网构软件的特性，建立网构软件体系结构代数模型。构件与构件间的连接关系如下。

1)设 A，B 是论域 U 中的两个构件，若 A 通过发送一个消息“激发”B 来实现功能需求，则称构件 A，B 进行了一次“激发”连接；

2)设 A，B 是论域 U 中的两个构件，若 A 通过“调用”B 来实现功能需求，则称构件 A，B 进行了一次“调用”连接；

3)设 A，B 是论域 U 中的两个构件，若 A 和 B 通过相互之间的协作与同步来实现功能需求，则称 B 与 A 协同连接；

4)设 A，B 是论域 U 中的两个构件，若 A 和 B 并发运行，则称 A 与 B 并行连接；

5)设 A，B 是论域 U 中的两个构件，若 A 与 B 重复执行，则称 A 与 B 重复连接；

6)设 A，B 是论域 U 中的两个构件，若 A 与 B 通过进行选择来实现功能需求，则称 A 与 B 选择连接。

### 2.2　代数性质

代数性质是在代数模型的基础上采用数学方法来进行演绎的，以归纳软件体系结构的代数性质。

文献[7]将连接子抽象为代数运算，将 A 与 B 的 6 种连接分别称为激发运算、调用运算、协同运算、并行运算、重复运算、选择运算，记为$\oplus$，$\otimes$，$\Theta$，$\|$，$\cdot$ 和$+$，提出了一些代数性质，如“激发”等运算满足结合率，“协同”运算满足交换率，“重复”运算满足交换率，“选择”运算满足交换率，“重复”、“激发”与“使用”、“协同”与“并行”运算都对“选择”运算满足分配率等，并对这些性质进行了验证，使其具有理论依据。本文将从上述性质中挑选部分来进行程序化的验证。

(1)“调用”运算对“选择”运算满足分配律

该性质的代数表达式为$(A+B)\otimes C=A\otimes C+B\otimes C$，说明等式左边和右边是等价的。其中，等式左边表示先对 A 和 B 进行选择，再调用 C 的功能；等式右边表示在 A 调用 C 且 B 调用 C 以后，再对结果进行选择。

(2)“并发”运算对“选择” 运算满足分配律

该性质的代数表达式为$(A+B)\|C=A\|C+B\|C$，说明等式左边和右边是等价的。其中，等式左边表示先对 A 和 B 进行选择，再与 C 并行；等式右边表示对 AC 并行和 BC 并行后，再对结果进行选择。

(3)任意一个网构软件体系结构〈C，O，Ω〉的代数表达式 ISAA 都可以写成如下标准型：$ISAA=a_1\oplus C_1+a_2\oplus C_2+\cdots+a_k\oplus C_k$，其中称 $a_i\in C_j$ 为常量；$C_j\in C$，即 $i,j=1,\cdots,k$，$i$ 和 $j$ 可能不等。

该性质表明，一个软件体系结构的代数表达式可表示成一个范式，该范式由多个基本块选择连接组成。它提供了一种求解基本路径集的全新思路，使程序流的表达更加高效，体现出了模型代数的实际应用价值。

(4)“调用”运算都能转化为“激发”运算

该性质表明，任何基于调用的程序都可以用基于消息激发的形式重构。基于调用的程序中各模块间的依赖度大、耦合度高；而基于消息的程序则相反，其传值不传址，依赖程度小、耦合度低，且与基于调用的程序相比，基于消息的代码运行时间较少，代码效率更高效。因此可将程序中基于调用的模块变为基于消息的模块，从而提高代码效率。

## 3　代数性质的程序化验证

代数性质的程序化验证即运行分别满足性质等式两边的不同结构的程序代码，并判断两者的运行结果是否相等，若相等，则性质正确，反之，则不正确。本节分别采用 BPEL 流程、伪代码、C 语言程序对 4 个代数性质进行验证，结果表明性质成立。

### 3.1　“调用”运算对“选择” 运算满足分配律的验证

BPEL(Business Process Execution Language)[9-10]是一种使用 Web 服务定义和执行业务流程的语言，其定义了 BPEL 主流程与外部服务之间的交互，外部服务由 WSDL 描述，WSDL 描述提供接口用于 BPEL 的调用[11]。因为 BPEL 中的服务是独立的，符合代数模型的构件性质，所以该性质欲通过 BPEL 流程来进行验证。本节对一个加减法组合服务的 BPEL 流程做相应的演化，其中 A 为 addService 加法服务，B 为 subService 减法服务，C 为 caculatorService 计算服务。图 1 给出了满足$(A+B)\otimes C$ 结构的 BPEL 流程图，选择运算符(加或减)之后再进行下一步的计算。图 2 给出了 $A\otimes C+B\otimes C$结构的 BPEL 流程图，当加法和减法并发计算后，再对运算符进行选择，最后根据选择直接输出结果。两种结构的运行结果一致，如图 3 所示，因此可以证明“调用”对“选择”运算满足分配率。

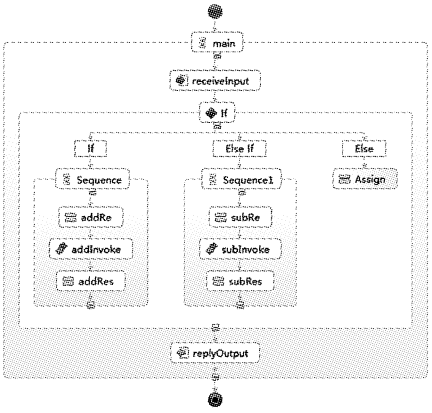


图 1　(A+B)⊗C 结构

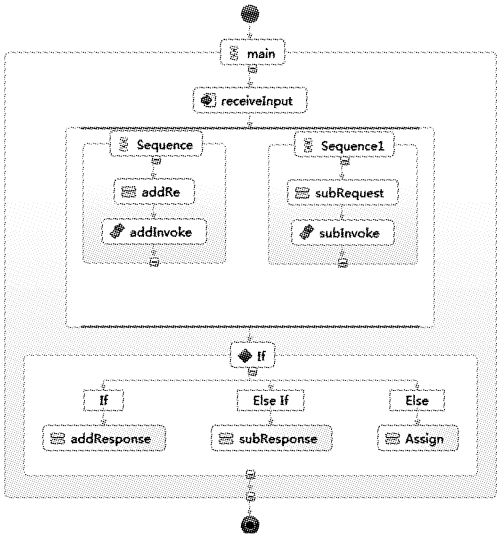


图 2 A⊗C+B⊗C 结构

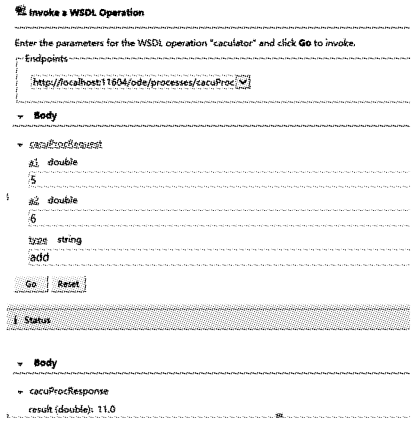


图 3 运行结果

### 3.2 “并发”运算对“选择”运算满足分配律的验证

同理,该性质也通过 BPEL 流程来验证,本节对一个旅行购票组合服务的 BPEL 流程做相应的演化,其中 A 为航空公司 1 的 Web 服务,B 为航空公司 2 的 Web 服务,C 为购买汽车票的 Web 服务。图 4 给出了满足(A+B)‖C 结构的 BPEL 流程图,选择航空公司与购买当地汽车票并发运行。而图 5 给出了 A‖C+B‖C 结构的 BPEL 流程图,航空公司 1、航空公司 2 分别与购买汽车票并发运行后,再对航空公司进行选择,从而得出结果。两种结构的运行结果一致,因此并行对“选择”运算满足分配率。

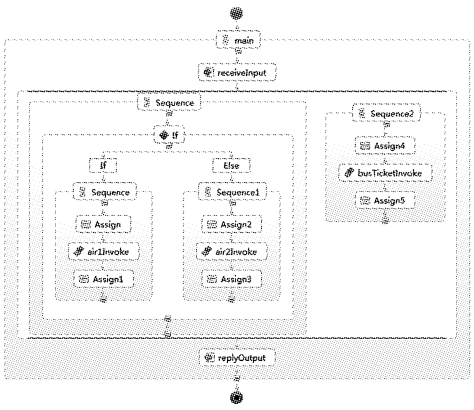


图 4 (A+B)‖C 结构

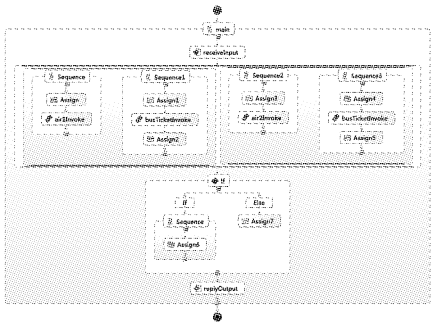

图 5 A‖C+B‖C 结构

### 3.3 代数模型范式的验证

由于该性质是对软件体系结构的代数表达式进行验证,因此选择描述程序流程的伪代码进行性质验证。本节对以下案例的伪代码进行处理,输出相对应的代数表达式。

案例:接受 3 个数作为输入,用作三角形的边,判断是否能组成三角形,若不能,则输出 no;若能,则输出三角形的类型。其伪代码如下:

```
1. Dim a,b,c As Integer;Dim IsATriangle As Boolean
2. Output("Enter 3 integers which are sides of a triangle")Input(a,b,c)
3. If (a<b+c) AND(b<a+c)AND(c<a+b) goto5
4. Then IsATriangle=True EndIf goto7
5. If !((a<b+c) AND(b<a+c)AND(c<a+b)) goto7
6. IsATriangle=False EndIf goto14
7. If IsATrangle AND [(a=b)AND (b=c)] goto9
8. Then Output("Equilateral") EndIf goto15
9. If IsATrangle AND [!((a=b)AND (b=c))] AND [(a<>b)
   AND(a<>c)AND(b<>c)] goto11
10. Then Output("Scalence") EndIf goto15
11. If IsATrangle AND [!(a=b)AND (b=c))] AND [!((a<>b)
    AND(a<>c)AND(b<>c))] goto13
12. Output("Isosecles") EndIf goto15
13. If !(IsATrangle)
14. Output("NOT a Triangle") EndIf
15. END Triangle
```

该伪代码可以转换为以下代数表达式:ISSA=n1 ⊕ n2 ⊕ n3 ⊕ n5⊕ n7 ⊕ n9 ⊕ n11 ⊕ n13 ⊕ n14 ⊕ n15+n1 ⊕ n2 ⊕ n3 ⊕ n5 ⊕ n7 ⊕ n9 ⊕ n11 ⊕ n12 ⊕ n15+n1 ⊕ n2 ⊕ n3 ⊕ n5 ⊕ n7 ⊕ n9 ⊕ n10 ⊕ n15+n1 ⊕ n2 ⊕ n3 ⊕ n5 ⊕ n7 ⊕ n8 ⊕ n15+n1 ⊕ n2 ⊕ n3 ⊕ n5 ⊕ n6 ⊕ n14 ⊕ n15+n1 ⊕ n2 ⊕ n3 ⊕ n4 ⊕ n7 ⊕ n9 ⊕ n11 ⊕ n13 ⊕ n14 ⊕ n15+n1 ⊕ n2 ⊕ n3 ⊕ n4 ⊕ n7 ⊕ n9 ⊕ n11 ⊕ n12 ⊕ n15+n1 ⊕ n2 ⊕ n3 ⊕ n4 ⊕ n7 ⊕ n9 ⊕ n10 ⊕ n15+n1 ⊕ n2 ⊕ n3 ⊕ n4 ⊕ n7 ⊕ n8 ⊕ n15。而案例的控制流图如图 6 所示,与代数表达式相符,因此代数性质成立。

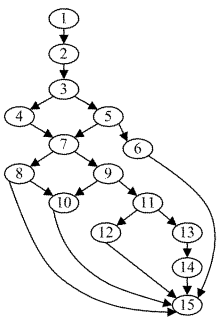


图 6 控制流图

### 3.4 “调用”运算都能转化为“激发”运算的验证

将含有“调用”的模块分为若干程序段，并对其进行重新构造，令每一个程序段中不再含有对其他模块的“调用”，使它们之间只有“消息”的传递。本节对求解 3 个数中的最大数的 C 程序进行转化，图 7 给出了基于“调用”的程序，图 8 给出了基于“消息”的程序，前者的运行结果如图 9 所示，后者的运行结果如图 10 所示，两者的运行结果一致，因此该性质成立。

```
#include<stdio.h>
#include<time.h>
int x1, x2, x3, max;
void input()
{
    scanf("%d%d%d",&x1,&x2,&x3);
}
int maxnum(int a,int b,int c)
{
    max=a;
    if(b>max)
      max = b;
    if(c > max)
      max = c;
    return max;
}
void output(int x)
{
      printf("max=%d\n\n",x);
}
int main()
{
      input();
      max=maxnum(x1,x2,x3);
      output(max);
      return 0;
}
```

图 7　基于“调用”的程序

```
#include<stdio.h>
#include<time.h>
int main()
{
    int maxnum(int x1,int x2,int x3);
    int x1, x2, x3, max;
    scanf("%d%d%d", &x1, &x2, &x3);
    max = maxnum (x1,x2,x3);
    printf("max = %d\n", max);
    return 0;
}
int maxnum(int x1,int x2,int x3)
{
    int max;
    max = x1;
    if(x2>max)
      max = x2;
    if(x3 > max)
      max = x3;
    return max;
}
```

图 8　基于“激发”的程序


图 9　基于“调用”的运行结果

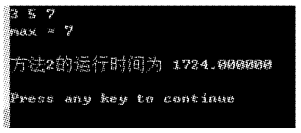


图 10　基于“激发”的运行结果

## 4　代数性质的应用

上节已对代数性质进行了验证，并得到了性质均成立的结果。本节将针对各代数性质的应用给出相应的算法，使其能够将符合一种结构的程序转化为符合另一种结构的程序，以令代数性质能够实际应用到软件演化中。

### 4.1 “调用”运算对“选择”运算满足分配律的应用

BPEL 中包含各类节点，例如 if，assign，sequence，invoke，flow 等[8]。找到各节点的内容，并分别存入相对应的数组中，最后按照转换后的顺序重新对节点进行排序，并将其对应的节点移动到指定的节点下。根据此过程得到 BPEL 流程的演化算法，如算法 1 所示。

**算法 1**　将(A＋B)⊗C 结构的 BPEL 转化为 A⊗C＋B⊗C 结构的 BPEL

begin
输入：一个满足(A＋B)⊗C 结构的 BPEL 文件
输出：一个满足 A⊗C＋B⊗C 结构的 BPEL 文件
if 语句包含 process 节点
　then 继续读入文件，直到语句中包含〈if〉节点，并将内容存入 head 中
else if 语句包含 condition 节点
　then 继续读入文件，直到语句中包含〈/condition〉节点，并将内容存入 condition[]中
else if 语句包含 sequence 节点
　then 把当行内容存入 sequence[]中
else if 语句包含 assign 节点
　then 继续读入文件，直到语句中包含〈/assign〉节点，将把内容存入 assign[]中
else if 语句包含 invoke 节点
　then 把当行内容存入 invoke[]中
else 语句中包含 reply 节点
　then 继续读入文件，直到语句结束，并将内容存入 tail 中
按照 head condition sequence assign invoke assign 的顺序写入文件
end

### 4.2 “并发”运算对“选择” 运算满足分配律

本节将对 BPEL 代码进行转化，如算法 2 所示。

**算法 2**　将(A＋B)‖C 结构的 BPEL 转化为 A‖C＋B‖C 结构的 BPEL

begin
输入：一个满足(A＋B)‖C 结构的 BPEL 文件
输出：一个满足 A‖C＋B‖C 结构的 BPEL 文件
if 语句包含 process 节点

then 继续读入文件，直到语句中包含〈flow〉节点，并将内容存入 head 中
else if 语句包含 condition 节点
then 继续读入文件，直到语句中包含〈/condition〉节点，并把内容存入 condition[]中
将 else if 语句包含 sequence 节点
then 把当行内容存入 sequence[]中
else if 语句包含 assign 节点
then 继续读入文件，直到语句中包含〈/assign〉节点，并将内容存入 assign[]中
else if 语句包含 invoke 节点
then 把当行内容存入 invoke[]中
else 语句中包含 reply 节点
then 继续读入文件，直到语句结束，并将内容存入 tail 中
将对应的节点数组移动到指定的节点数组下，重新排序并写入文件中
end

### 4.3 代数模型范式的应用

本节将对程序的伪代码进行处理，生成程序的模型代数表达式，而该模型表达式即为标准型，如算法 3 所示。

**算法 3** 求出程序的代数表达式

begin
输入：一个已划分好基本块的伪代码文件
输出：a1⊕C1+a2⊗C2+…+ak⊕Ck 型的代数表达式
读取该伪代码文件
找到结点的前驱后继，生成 SuccessorMap(关系表)
进行递归，生成代数表达式：
当该结点为最后一个结点时，退出递归，返回当前结点号；
当结点有一个后继结点时，加入结点的关系，拼接表达式，再次进行递归；
当结点有多个后继结点时，加入结点的关系，拼接表达式，再继续循环递归，直至最后一个结点；
对代数表达式进行去括号操作
end

### 4.4 “调用”运算都能转化为“激发”运算的应用

本节将对 C 语言代码进行处理，将基于调用的 C 语言程序转化为基于消息的 C 语言程序，如算法 4 所示。

**算法 4**

begin
输入：一个基于调用的 C 程序文件
输出：一个基于消息的 C 程序文件
if 语句包含 #
then 当行内容存入 block[i]中，i++
else if 语句包含 return
then 自 return 以后的所有语句都存入 tail 中
else
if 语句为变量声明
then 当行内容存入 statement[j]中，j++
else if 语句包含 scanf 节点
then 适当改写当行内容并存入 statement[j]中，j++
else if 语句包含 printf 节点
then 适当改写当行内容并存入 statement[j]中，j++
else
then 把当行内容存入 statement[j]中，j++
按照 block[]，statement[]，tail 的顺序写入文件
end

**结束语** 本文首先通过案例对 4 个代数性质进行程序化的验证，对分别运行两种结构的程序所得到的结果进行对比，得到了其运行结果一致的结论，自此，代数模型的代数性质不仅有理论的验证，还有了程序的支撑。最后又提出了代数性质的应用算法，使符合一种结构的程序能够自动转化为另一种结构，而不用手动修改程序，节省了大量时间，提高了效率，使代数模型的实用性大大增加，不仅丰富了建模理论，同时还为软件演化提供了一种新的思路，使得软件演化从理论研究转化为实际应用成为可能。当然，本文只对 4 个代数性质进行了验证，未来还需验证更多的性质，以进一步丰富代数模型。

## 参 考 文 献


[1] GAO H, ZHANG L, FAN Z Q. Software architecture description technique based on template[J]. Jounal of Beijing University of Aeronautics and Astronautics, 2008, 34(1): 122-126. (in Chinese)
高晖，张莉，樊志强. 基于模板的软件体系结构描述技术[J]. 北京航空航天大学学报，2008，34(1)：122-126.

[2] ZHU X Y. Study on Formal Deseription of Sotfware Architecture[D]. Beijing: Graduate University of Chinese Academy of Sciences, 2005. (in Chinese)
朱雪阳. 软件体系结构形式描述研究[D]. 北京：中国科学院研究生院(软件研究所)，2005.

[3] HUANG Z B, ZHANG G Q. A New Method about the Description of Software Architecture[J]. Microelectronics & Computer, 2006, 23(12): 82-84, 88. (in Chinese)
黄正宝，张广泉. 一种新型的软件体系结构描述方法研究[J]. 微电子学与计算机，2006，23(12)：82-84，88.

[4] YUAN B, WANG B Q. Algebraic Model of Reconfigurable System Based on Component Operation[J]. Journal of Software, 2012, 23(10): 2735-2745. (in Chinese)
袁博，汪斌强. 基于构件运算的可重构系统代数模型[J]. 软件学报，2012，23(10)：2735-2745.

[5] ZHANG Y S. Study on Methods for Description of Software Architecture and Software Evolution Based on Algebra Theory [D]. Changsha: Central South University, 2007. (in Chinese)
张友生. 基于代数理论的软件体系结构描述及软件演化方法研究[D]. 长沙：中南大学，2007.

[6] DAI F, LI T, XIE Z W, et al. Towards an Algebraic Semantics of Software Evolution Process Models[J]. Journal of Software, 2012, 23(4): 846-863. (in Chinese)
代飞，李彤，谢仲文，等. 一种软件演化过程模型的代数语义[J]. 软件学报，2012，23(4)：846-863.

[7] ZHAO H Q, SUN J. An algebraic model of Internetware software architecture[J]. Chinese Science: Information Science, 2013, 43(1): 161-177. (in Chinese)
赵会群，孙晶. 网构软件体系结构代数模型[J]. 中国科学：信息科学，2013，43(1)：161-177.

[8] ZHAO H Q, SUN J. An Algebraic Model of Service Oriented

Trust worthy Software Architecture[J]. Chinese Journal of Computers, 2010, 33(5): 890-899. (in Chinese)
赵会群，孙晶. 面向服务的可信软件体系结构代数模型[J]. 计算机学报，2010，33(5)：890-899.

[9] ALVES A, ARKIN A, ASKARY S, et al. Web services business process execution language version 2. 0[EB/OL]. (2007-04-11)[2012-08-19]. http://docs. oasis-open. org/wsbpel/2. 0/OS/wsbpel-v2. 0-OS. html.

[10] SUN J, LI D F. Study of algorithm on evolution for BPEL structure[J]. Application Research of Computer, 2016, 33(9): 1-6. (in Chinese)
孙晶，李东方. 一种 BPEL 结构演化算法研究[J]. 计算机应用研究，2016，33(9)：1-6.

[11] ZHAO H Q, YIN C R. Research of model checking method and technology guided by testing purpose[J]. Application Research of Computer, 2015, 32(8): 2387-2390, 2394. (in Chinese)
赵会群，殷朝冉. 测试目的引导的模型检测方法与技术研究[J]. 计算机应用研究，2015，32(8)：2387-2390，2394.

(上接第 239 页)

来越高。希望本文所提方案能够缓解目前以手工为主的繁琐的测试过程中的由 Android 设备、IOS 设备系统和硬件的多样性造成的麻烦。

此外，在未来的工作中还需进一步完善已有的工作，在原有的基础上增加测试的项目，如安全性测试、可靠性测试，完善动作-脚本 API 映射表，对路径算法不断进行完善等，从而使之真正摆脱人工干预，功能更加强大。在研究工作中，可以将其不断扩展到资源泄漏测试、能耗测试等多个方面。

## 参 考 文 献

[1] 中国互联网信息中心. 第 37 次《中国互联网络发展状况统计报告》[OL]. http://www. cnnic. net. cn/hlwfzyj/hlwxzbg.

[2] LUO Z J, WU W J, YANG M. Mobile Internet: Terminal Device, Networks and Service[J]. Chinese Journal of Computer, 2011, 34(11): 2029-2051. (in Chinese)
罗军舟，吴文甲，杨明. 移动互联网：终端，网络与服务[J]. 计算机学报，2011，34(11)：2029-2051.

[3] ATKINSON C, KUHNE T. Model-driven development: a metamodeling foundation[J]. IEEE Software, 2003, 20(5): 36-41.

[4] MELLOR S J, CLARK T, FUTAGAMI T. Model-driven development: guest editors' introduction[J]. IEEE Software, 2003, 20(5): 14-18.

[5] BALASUBRAMANIAN K, GOKHALE A, KARSAI G, et al. Developing applications using model-driven design environments [J]. Computer, 2006, 39(2): 33-40.

[6] BUTLER M. Android: Changing the mobile landscape[J]. Pervasive Computing, IEEE, 2011, 10(1): 4-7.

[7] GOADRICH M H, ROGERS M P. Smart smartphone development: iOS versus Android[C]// Proceedings of the 42nd ACM Technical Symposium on Computer Science Education. ACM, 2011: 607-612.

[8] KOCHAN S G. Programming in Objective-C[M]. Addison-Wesley Professional, 2011.

[9] ABOGHARAF A, PALIT R, NAIK K, et al. A methodology for energy performance testing of smartphone applications[C]// 2012 7th International Workshop on Automation of Software Test (AST). IEEE, 2012: 110-116.

[10] ANAND S, NAIK M, HARROLD M J, et al. Automated concolic testing of smartphone apps[C]// Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering. ACM, 2012: 1-11.

[11] ZADGAONKAR H. Robotium Automated Testing for Android [M]. Packt Publishing Ltd, 2013.

[12] AMALFITANO D, FASOLINO A R, TRAMONTANA P, et al. Using GUI ripping for automated testing of Android applications[C]// Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering. ACM, 2012: 258-261.

[13] HU C, NEAMTIU I. Automating GUI testing for Android applications[C]// Proceedings of the 6th International Workshop on Automation of Software Test. ACM, 2011: 77-83.

[14] MORZAEO N, MALEK S, PĂSĂREANU C S, et al. Testing Android apps through symbolic execution[J]. ACM SIGSOFT Software Engineering Notes, 2012, 37(6): 1-5.

[15] BORMAN M. Developing, and testing, a theoretical framework for inter-organisational systems (IOS) as infrastructure to aid future IOS design[J]. Information Systems and e-Business Management, 2006, 4(4): 343-360.

[16] PENN J. Test iOS Apps with UI Automation: Bug Hunting Made Easy[M]. Pragmatic Bookshelf, 2013.

[17] MELLOR S J, BALCER M, FOREWORD B J I. Executable UML: A foundation for model-driven architectures[M]. Addison-Wesley Longman Publishing Co., Inc., 2002.

[18] LODDERSTEDT T, BASIN D, DOSER J. Secure UML: A UML-based modeling language for model-driven security[M]// 《UML》2002—The Unified Modeling Language. Springer Berlin Heidelberg, 2002: 426-441.

[19] JENSEN C S, PRASAD M R, MØLLER A. Automated testing with targeted event sequence generation[C]// Proceedings of the 2013 International Symposium on Software Testing and Analysis. ACM, 2013: 67-77.

[20] SKIENA S. Implementing discrete mathematics: combinatorics and graph theory with Mathematica[M]. Addison-Wesley Longman Publishing Co., Inc., 1991.

[21] TARJAN R E. Fast algorithms for solving path problems[J]. Journal of the ACM (JACM), 1981, 28(3): 594-614.

[22] TARJAN R E. A unified approach to path problems[J]. Journal of the ACM (JACM), 1981, 28(3): 577-593.

[23] LEI B, WANG L Z, BU L, et al. Robustness Testing for Components Based on State Machine Model[J]. Journal of Software, 2010, 21(5): 930-941. (in Chinese)
雷斌，王林章，卜磊，等. 基于状态机模型的构件健壮性测试[J]. Journal of Software, 2010, 21(5): 930-941.