DOI:10.19651/j.cnki.emt.1802195

软件测试中的 Mock 辅助生成技术研究

冯俊池 连 尧 董 超 尹党辉
(军事科学院系统工程研究院 后勤科学与技术研究所 北京 100071)

摘 要: Mock 技术对提高软件单元测试的独立性和全面性起到了重要作用。现有的 Mock 生成框架需要人为编写 Mock 方法的逻辑,工作量较大且易引入人为错误。针对测试中 Mock 方法构建效率较低的问题,提出了一种基于数据驱动思想的 Mock 辅助生成方法,定义 Mock 模型描述要实现的内部逻辑,从而将输入输出数据独立于测试脚本。首先,通过对程序的分析,辅助确定待模拟方法及其输入输出参数;其次,针对输入输出关系进行建模;然后,根据输入输出关系为待模拟方法生成 Mock 方法;最后,将被测对象中的待模拟方法替换为 Mock 方法。通过实验,证明了所提出方法的有效性。

关键词: 软件测试;Mock 模型;自动化测试;数据驱动;Java 软件
中图分类号: TP311.5, TN06 **文献标识码**: A **国家标准学科分类代码**: 520.3050

Research on aided generation technology of Mock in software testing

Feng Junchi Lian Yao Dong Chao Yin Danghui
(Institute of Logistic Science and Technology, Academy of Systems Engineering, Academy of Military Sciences, Beijing 100071, China)

Abstract: Mock technology plays an important role in improving the independence and comprehensiveness of software unit testing. The existing Mock generation framework needs to write the logic of Mock method manually, which leads to heavy workload and possibly introduce human errors. In view of the low efficiency of Mock method construction in testing, an aided generation method of Mock based on data driven idea is proposed, which defines the Mock model to describe internal logic of the implementation, so that the input and output data are independent of the test script. Firstly, through the analysis of programs under test, the simulated method and its input and output parameters are found and analyzed. Secondly, the input and output relationship is modeled. Thirdly, the Mock method is generated for the simulated method based on the input and output relation. Finally, the simulated method in programs under test is replaced by the Mock method. The effectiveness of the proposed method is proved by experiments.

Keywords: software testing; Mock model; automated testing; data driven; Java software

0 引 言

在软件开发项目中,往往存在各种应用内外部的调用关系,以及与第三方系统的依赖,为测试带来了实施难度和进度上的不确定性。基于 Mock 的测试通过构造一系列符合预定义规则的模拟对象来与被测对象进行交互,从而判断被测对象在正常逻辑或异常逻辑下能否正常工作[1],在一定程度上避免了所依赖的真实对象未开发完成、不可确定、不可预测等情况导致的问题。这个模拟对象就是 Mock 对象,模拟的方法为 Mock 方法,用来代替真实对象和方法以便测试。

随着自动化测试[2-4]的发展,出现了一些 Mock 辅助生成工具,包括 JMock、EasyMock、JMockit 等[5-7]。这些工具能够针对被模拟方法搭建测试框架[8-9],但仍需要人工确定被模拟的方法并编写逻辑代码。数据驱动[10-11]是将测试数据独立于测试脚本的一种测试思想,从数据文件等其他资源中读取数据,通过变量参数化将读取的数据传到测试脚本中,减少了测试数据与测试代码的耦合度,提高了可维护性和可复用性。

本文提出了一种针对 Java 应用的 Mock 生成方法,将数据驱动思想应用于 Mock 方法生成过程中,使测试数据与代码相分离,基于程序分析与人工选择发现需要模拟的方法,通过描述输入输出关系生成 Mock 模型和构造 Mock 代码,模拟构造各种场景,增加测试的真实性和全面性,提

收稿日期:2018-09-29

高测试的自动化程度。

1 方法提取分析

在基于 Mock 的测试过程中,确定需要被 Mock 方法替代模拟的方法是第一步要做的工作,通过对程序的分析[12],可以提取其中的方法调用,结合自动和人工的方式来选择待模拟方法并进行分析。

Mock 方法通过构造和原方法签名以及返回值相同的方法,以替换原方法,读取输入数据,并返回指定的值。因此在构造 Mock 方法之前,需要获取待模拟方法的签名,即输入参数类型以及输出参数类型等必需信息。

提取分析待模拟方法的流程如图 1 所示。首先,对被测程序代码进行分析,构造语法分析树;其次,对程序的语法分析树进行遍历,找到方法调用;然后,针对找到的方法调用,判断被调用对象是否在程序中,即是否已存在相关实现代码,若不存在或其属于外部调用,则将其列入候选对象,也可加入人工干预,对待模拟方法进行筛选;确定待模拟方法后,提取相关必需信息并保存;最后,遍历完被测程序,即得到待模拟方法集合。

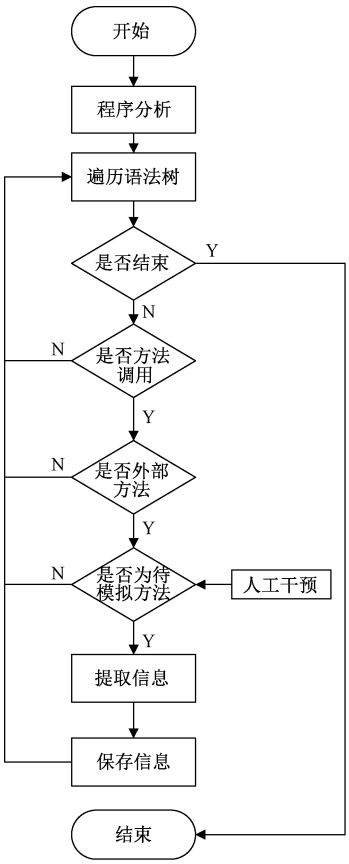

图 1 待模拟方法提取分析流程

以 ClassTest 作为被模拟类,objectTest 为其对象,staticMethod 为静态方法,nonStaticMethod 为非静态方法,方法调用类型形式如图 2 所示。

① ClassTest. staticMethod
② objectTest. staticMethod
③ objectTest. nonStaticMethod
④ (new ClassTest()).nonStaticMethod

图 2 方法调用类型

在对待模拟方法的分析中,需要提取被测方法的方法名、所属类的类名、输入参数及参数类型、输出参数类型等信息。但对于某些情况,不仅需要待模拟方法签名中的参数,影响其运行状态的参数也要考虑。如对于图 2 中第 4 种情况,当 ClassTest 创建对象实例时采用有参构造方法,则其构造方法中的参数也会影响对象的行为,因此也需要作为输入参数考虑。同理,第 2 和第 3 种情况中的对象创建时所使用构造方法中的参数也是需要采集的信息。

2 Mock 方法模型

Mock 方法的核心在于如何对指定的输入返回指定的输出,也就是输入数据与输出数据之间关系的确定。通过建立 Mock 方法模型,定义形式化的输入输出关系,能够有效实现 Mock 方法生成的自动化。

定义 1:每个 Mock 方法定义为由类名、方法名和一组输入输出关系组成的模型,其中类名 className 以“Mock_原类名”的形式定义,方法名 methodName 和原方法保持一致。

MockMethod:=(className, methodName, relationSet)

relationSet:=relation *

定义 2:每条输入输出关系 relation 包含输入关系 inputRule 和输出关系 outputRule,当方法的输入数据满足输入关系时,则方法的输出数据满足对应的输出关系。

relation:=(inputRule, outputRule)

定义 3:输入关系由零至多条参数规则组成,参数规则以参数三元组或谓词语句的形式表达。

inputRule:={parameterTriple | predicateExpression} *

parameterTriple:=(name,type, valueRule)

valueRule:=value | regrx | dataSource

参数三元组包含参数名 name、参数类型 type 和取值规则 valueRule,其中取值规则可以采用参数值、正则表达式或数据源的形式描述。

谓词语句以判断条件的形式描述输入数据之间所要满足的约束关系,将其加入到 Mock 方法中,可以根据逻辑判断条件来选择不同的输出。

当 Mock 方法获取输入数据时,根据指定的取值规则或判断条件,判断输入数据是否满足输入关系,针对每一条输入输出关系,当输入关系中的每一条规则均满足时,则 Mock 方法返回满足输出关系的值。

定义 4：输出关系由一条或多条参数规则组成。

outputRule：=(type，outValueRule)

outValueRule：= value | interval | dataSource | outputRule *

Mock 方法的返回值为基本类型或对象，基本类型为具体值，对象则需要满足某些属性约束。针对基本类型，数据描述类型可以采用参数值、参数区间或数据源的形式。当类型为参数区间时，从给定的区间内随机选取数据作为返回值；当类型为数据源时，则从指定位置（文件或者数据库）读取数据。

针对对象类型，type 为对象所对应的类，在创建类及对象时，需要指定其成员变量及取值，因此每个对象包含多条参数规则 outputRule。在返回的对象中，包含了所指定的成员变量，每个成员变量的取值同样按参数规则生成。

3　基于 Mock 辅助生成的测试过程

基于 Mock 辅助生成的测试过程如图 3 所示，主要分为 5 个阶段，分别是测试分析、Mock 建模、Mock 生成、Mock 替换以及执行测试阶段。

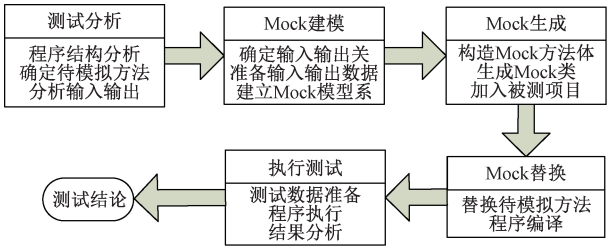

图 3　基于 Mock 辅助生成的测试过程

在测试分析阶段，主要针对被测方法进行结构分析，确定待模拟方法并分析输入、输出参数等内容，获取 Mock 所需的必要信息；在 Mock 建模阶段，主要是根据从待模拟方法获得的信息建立 Mock 模型，确定输入输出关系，并准备相关数据；在 Mock 生成阶段，构造 Mock 方法体，生成 Mock 方法所在的类并加入被测项目中；在 Mock 替换阶段，采用 Mock 方法替换被测程序中的待模拟方法，并对被测程序进行编译和集成构建[13]；在测试执行阶段，主要包括准备测试数据[14]、执行被测程序以及分析测试结果等内容。

4　实验与分析

以常用的三角形判别程序[15]为待模拟方法进行实验，其方法调用形式为：

String triType＝Tri.analysisTriType(int a，int b，int c)；

由以上调用语句可知 analysisTriType 为类 Tri 的静态方法，参数属性如表 1 所示。

通过自动分析提取方法 analysisTriType 的相关属性，

表 1　方法参数属性

参数	类型	说明
a	int	边长
b	int	边长
c	int	边长
triType	String	三角形类型

建立 Mock 模型如图 4 所示。模型中采用直接指定数值的方式来约束输入参数，也可从数据库或者文件中进行读取。

```
(Mock_Tri, analysisTriType, relationSet)
relationSet={relation1, relation2, relation3, relation4, …}
relation1={{a, int, 8) (b, int, 8) (c, int, 8)},(String, "Equilateral"))
relation2={{a, int, 8) (b, int, 8) (c, int, 6)},(String, "Isosceles"))
relation3={{a, int, 5) (b, int, 3) (c, int, 6)},(String, "Scalene"))
relation4={{a, int, 2) (b, int, 3) (c, int, 6)},(String, "Not_Triangle"))
```

图 4　Mock 模型（值）

针对 analysisTriType 方法，为输入参数指定数值的方式很难列举出所有情况，因此可以采用谓词语句描述参数的约束规则，如图 5 所示，分别针对 4 种不同的输出，给出了输入参数应该满足的约束关系。

```
(Mock_Tri, analysisTriType, relationSet)
relationSet={relation1, relation2, relation3, relation4 }
relation1=( (a==b&&a==c),(String, "Equilateral"))
relation2=( ((a==b)&&(b!=c)&&(c<a+b)||(a==c)&&(b!=c)&&(b<a+c)
||(b==c)&&(a!=c)&&(a<c+b)),(String, "Isosceles"))
relation3=(((a+b>c)&&(a+c>b)&&(b+c>a)&&a!=b&&b!=c&&a!=c),
(String, "Scalene"))
relation4=(((a+b<=c)||(a+c<=b)||(b+c<=a)),(String, "Not_Triangle"))
```

图 5　Mock 模型（谓词语句）

以 Mock 模型为输入生成的 Mock 代码如图 6 所示，在

```
public class Mock_Tri {
  …
  public static String analysisTriType(int a, int b, int c){
    if(a==b&&a==c){
      return "Equilateral";
    }
    …
  }
```

图 6　Mock 代码

模型中设定的输入输出关系在方法的内部逻辑中实现,从而避免了手工编写方法逻辑的繁琐。通过将 Mock 方法替换被测程序中的待模拟方法并进行编译,在执行时,Mock 能够按照预设的逻辑运行,根据输入的三角形边长输出其类型,从而达到了支持测试过程进行的目的。

5 结 论

针对现有 Mock 测试工具无法生成内部逻辑代码的问题,本文结合数据驱动测试思想,提出了一种 Mock 辅助生成方法,通过定义 Mock 模型,指定输入输出之间的对应关系,使 Mock 拥有与被模拟对象一致的接口,用来替代被测代码中所需的对象。该方法实现了 Mock 内部逻辑代码的自动生成,并将测试数据和代码分离,提高了测试的自动化程度和效率。实验结果表明,生成的 Mock 方法能够按预先设置的规则返回数据,检查被测程序是否按照所期望的逻辑进行工作,从而达到测试目的。

参考文献

[1] 尹春娇.自动化单元测试中 MOCK 技术的研究与应用[D].合肥:安徽大学,2011.

[2] VAHID G, MIKA V M. When and what to automate in software testing? A multi-vocal literature review[J]. Information and Software Technology, 2016, 76(8): 92-117.

[3] 韩东,郭士瑞,高剑,等.基于自动测试系统的测试数据格式标准化研究[J].电子测量技术,2017,40(6):47-52.

[4] 邓璐娟,李金萌,董东晓.自动化测试框架技术及应用[J].计算机测量与控制,2016,24(9):86-88.

[5] 陈丽萍,张勇,丁智敏.自动化单元测试框架 EasyMock 分析及其应用[J].巢湖学院学报,2014,16(6):34-38.

[6] ANDREA A, GORDON F, RENE J. Private API access and functional mocking in automated unit test generation[C]. 2017 IEEE International Conference on Software Testing, Verification and Validation, New York: IEEE Press, 2017: 126-137.

[7] JAFADEESH N, TAO Y L. Using mock object frameworks to teach object-oriented design principles[J]. Journal of Computing Sciences in Colleges, 2010, 26 (1): 40-48.

[8] 孟燕.自动化测试技术中 Mock 框架的建设与应用[J].武汉理工大学学报(信息与管理工程版),2017,39(3):364-367.

[9] SHAIKH M, WANG X Y. An empirical study on the usage of mocking frameworks in software testing[C]. The 14th International Conference on Quality Software, New York: IEEE Press, 2014: 127-132.

[10] NEIL B. A deep dive into a data-driven world of test[C]. 2017 IEEE AUTOTESTCON, New York: IEEE Press, 2017: 1-8.

[11] 柳晓华,李劲华.基于 BDD 的数据驱动自动化测试方法[J].青岛大学学报(自然科学版),2016,29(3):89-92,96.

[12] 王克朝,成坚,王甜甜,等.面向程序分析的插桩技术研究[J].计算机应用研究,2015,32(2):479-484.

[13] 姜文,刘立康.基于持续集成的 C/C++软件覆盖率测试[J].计算机技术与发展,2018,28(3):37-41,46.

[14] 高雪笛,周丽娟,张树东,等.基于改进遗传算法的测试数据自动生成的研究[J].计算机科学,2017,44(3):209-214.

[15] 喻琴仪.基于遗传算法的路径覆盖测试用例生成技术研究[D].衡阳:南华大学,2015.

作者简介

冯俊池,硕士、研究实习员,主要研究方向为自动化测试。E-mail:fengjct@foxmail.com

连尧,硕士、工程师,主要研究方向为后勤信息化。

董超,硕士、助理研究员,主要研究方向为软件测试。

尹党辉,硕士、工程师,主要研究方向为软件测试。