基于模型的方法在软件测试领域的应用与发展①

马云云, 王金波, 张 弢, 王晓华

(中国科学院 空间应用工程与技术中心, 北京 100094)

摘 要: 伴随着软件在当今社会中扮演着日益重要的角色, 在各类软件或系统的开发过程中, 测试变得越来越重要. 随着面向对象技术的广泛应用和软件测试自动化的要求, 基于模型的测试(MBT)技术逐渐得到了软件开发和测试人员的认可和接受. 尽管有许多国内出版物讨论了基于模型的测试技术, 但还没有相关技术文章对 MBT 技术及工具的当前发展进行回顾与总结. 文章对当前 MBT 技术的通用模型、测试用例生成方法、测试目标选择等 MBT 方法进行回顾, 并对基于模型的自动化测试工具、应用实践进行了概述. 最后,探讨了 MBT 技术的未来发展与研究方向.

关键词: 基于模型软件测试; 测试模型; 测试用例; 测试目标; 测试工具

Survey on Model-Based Testing in Software Testing

MA Yun-Yun, WANG Jin-Bo, ZHANG Tao, WANG Xiao-Hua

(Technology and Engineering Center for Space Utilization, Chinese Academy of Science, Beijing 100094, China)

Abstract: Along with the increasingly important role of software in today's society, during the development process of software or system, testing becomes more and more important. With the wide application of object-oriented technology and the demand of automation testing, model-based testing (MBT) approach have been approved by software engineering. Although there are a number of publications discussing model-based software testing, we are lack of technical papers and publications presenting a review of the current advances in model-based software testing and automation tools. The goal of this paper is to review the model-based testing approach, it first discusses the popular models used in model-based testing, the test case generation methods, the testing target. Then, it summarizes MBT practices and discusses and compares the major model-based test tools. Finally, the paper close ends with a discussion of the challenges of current MBT approach, and where model-based testing fits in the future.

Key words: MBT; test model; test case; test target; test tools

软件质量对软件产品极其重要, 而软件测试是保证软件质量最后也是最重要的阶段. 随着软件系统复杂度的增加和软件产品商业竞争的加剧, 软件工程中对有效的软件测试方法和自动化测试工具需求越来越迫切.

在过去几十年中, 许多研究学者、软件开发者和工具商都致力于开发创新性的软件测试方法和工具来支持软件测试活动, 借助于这些方法和工具的应用, 来提高软件产品的质量; 然而直到现在, 每年由于软件错误带来的损失仍可达数十亿美元[13,14], 这是由三个方面的原因造成的: 第一, 大部分的测试工具关注于测试的自动执行, 缺乏内建的有效的测试方法和技术, 无法满足测试过程不同阶段的需要, 包括测试需求分析和策划, 测试用例设计和生成, 测试执行和测试结果确认以及测试覆盖率分析. 第二, 大部分自动化测试工具没有提供有效的自动化测试方案来支持回归测试, 比如变更影响域分析以及测试集的更新和重用. 第三, 缺乏面向领域应用特征的有效测试方法和可行的自动化测试手段及技术改进, 例如对特定领域复杂算法、多媒体通信、富媒体交互和基于移动系统

① 收稿时间:2015-03-25;收到修改稿时间:2015-05-07 [doi: 10.15888/j.cnki.csa.004880]

© 中国科学院软件研究所 http://www.c-s-a.org.cn

的服务访问等领域的测试.

基于以上原因, 越来越多的人投入到对MBT技术的研究中. MBT 技术是软件测试领域一种测试方法, 依据形式化规约或模型规定的预期行为来检查被测实现运行结果的正确性[8]. MBT 提供的系统研究方法可用来开发下一代自动化测试方法, 以满足软件测试过程中的不同需要. 很多的研究结果认为[9,10], MBT 是一种能够有效解决上述三个问题的有效手段.

MBT 的应用范围相当广泛, 在不同的软件开发模型, 不同的业务领域和不同的机构中均可应用. 例如 MBT 可以应用于嵌入式、事务处理、通信应用等领域; 同时从另方面来看, 对于采用瀑布、敏捷、增量和 V 模型等开发过程模型的系统, 也同样可以应用MBT技术. 根据 Robert V. Binder 于 2014 年进行的调查[12], 平均情况下, MBT 技术可以降低 29%的 bug 遗漏, 35%的测试成本和 30%的测试时间延期.

本文从模型驱动测试的角度, 综述近几年MBT的研究成果, 剖析一些重要概念, 总结研究热点并分析现有研究的不足. 第 1 节介绍常用测试模型、测试用例生成、测试目标选择等方面基本的 MBT 方法; 第 2 节介绍 MBT 现有的建模与测试工具及工具间的差异对比; 第 3 节对全文进行总结并讨论 MBT 技术未来可能的研究与发展方向.

1 基于模型的测试方法

关于基于模型的测试方法包含了多个方面, 本节重点关注以下几个方面:

① 测试模型: 讨论 MBT 技术中常用的测试模型.

② 测试用例生成: 讨论 MBT 方法中测试用例的生成机制, 以及实现测试用例自动生成所要满足的条件.

③ 测试目标: 测试目标在不同的方法中有着很大差异, 有些以设计模型为目标, 有些仅以实现为目标.

1.1 测试模型

基于模型的测试中用到许多软件模型, 本节对常用的模型进行综述, 包括: 有限状态机, 状态图, UML 状态机、语义模型和马尔可夫链模型.

(1) 有限状态机

有限状态机可以有效地表达软件系统和相关过程中基于状态的行为. 有限状态机由 5 个部分组成(I、S、T、F、L). I 代表系统的输入, S 代表系统的所有状态集合, T 代表在特定状态下, 当输入作用于系统时, 系统状态发生转换的功能, F 代表系统终止时的最终状态集, L 代表系统启动所进入的状态.

有限状态机的主要限制是其复杂性问题. 一般来说, 如果系统的复杂性在某个程度之下, 模型很容易设计和维护, 但随着系统复杂性的增加, 模型变得难以设计和维护. 除此之外, 有限状态机不支持并发, 因此这类模型不能用于并发模型.

(2) 状态图模型

状态图使用分层架构, 其中某个特定状态可以分解为一个或多个下一层级的有限状态机. 状态图模型支持并发、可以描述影响状态迁移的外部条件. 面向复杂、实时软件系统的测试, 需要考虑系统的并发和行为顺序, 对上述系统进行测试, 采用状态图模型要比有限状态机更有效. 但与有限状态机类似, 状态图对于复杂的系统也难于设计和维护.

(3) UML 状态机

UML 状态机与有限状态机类似, 但加入了更复杂的特征. UML 状态机可以用来对动态行为建模, 比如: 类、用例、子系统甚至是整个系统, 通常被用来对类的动态行为进行建模. 由于 UML 可以精确地描述软件功能间的连接形式, 且这些连接形式很容易通过自动化的方式进行操控, 因此选择 UML 模型用于测试具有很大的优势[15].

(4) 语义模型

通常情况, 语义模型用来与其他输入语言配合来描述程序的语法, 易于编写、核查和维护. 当系统拥有多种解析器或多种协议时该模型尤其有用[18].

(5) 马尔可夫链模型

马尔可夫链是一个具有马尔可夫性的离散随机过程. 具有马尔科夫性意味着当前状态、未来状态与过去状态独立, 即当前的状态描述完整地捕获了所有影响到过程未来演变的信息. 在每一时刻根据特定的概率分布, 系统可能从当前状态转变为另一个状态, 或维持同样的状态.

马尔可夫链通常用有向图描述, 每一条边都表示从一个状态到另一个状态变迁的几率. 马尔可夫链在结构上与有限状态机类似, 可以被认为是概率自动机. 它的主要价值不单体现在测试用例生成, 还体现在它可以搜集和分析失效数据, 作为评估可靠性和平均故

© 中国科学院软件研究所 http://www.c-s-a.org.cn

障间隔期(MTTF)的手段[16].

1.2 测试用例的生成

自动化软件测试极大地降低了软件测试的成本. 自动化测试生成基于规范描述语言实现, 规范描述语言具有形式化的语义. 本节讨论基于模型的测试用例自动生成方法.

众多研究成果提出了不同的测试用例的生成方法, 例如, 基于场景、基于模型、面向路径、面向目标的方法[1]. 基于模型的技术从 UML 视图中识别测试用例, UML 视图包括了: 活动图、时序图、状态机图、类或对象视图等. 面向路径的测试基于软件的静态或动态控制流: 静态路径测试通过符号表达式来执行程序路径; 动态路径测试通过运行时测试来执行程序路径. 面向目标的方法识别测试用例以覆盖选择的目标, 例如状态或分支, 目标的选择与采用的路径无关.

Rayadurgam[2]提出了一种用于结构覆盖准则的测试用例自动生成方法, 该方法对于可用有限状态机来表示的软件产品均可生成满足预定覆盖率要求的完整的测试用例集. 他给出了一种适合于定义测试用例生成方法, 且易于提取软件产品有限状态的形式框架. 这里, 软件产品可以是程序代码、软件规范或需求模型.

Aynur[3,4]在 Offutt 提出的基于状态的技术基础上, 给出了一种测试数据生成方法: 采用 TSL 语言描述测试用例中所有的元素, 从 UML 状态机图中产生一系列测试数据. 在他的方法中, 他对测试需求和测试说明做出了如下定义: 测试需求是测试过程必须满足和覆盖的测试内容; 测试说明是对测试用例的明确描述, 包括与测试需求和准则有关的测试数据.

Samuel[5]提出了一种从 UML 时序图生成测试序列的方法. 他认为现有的测试序列生成技术不能够覆盖 UML 时序图中的很多关键特征, 通过考虑 UML 时序图的关键特征, 如循环、分支和终止特征等, 提出了一种基于遍历算法的自测试序列自动生成方法.

M.Prasanna 提出了一种针对于面向对象系统的测试用例生成方法, 他认为测试用例可以通过分析对象在内部或外部激励下产生的动态行为来生成, 并提出了一种基于遗传算法的测试用例生成方法[6].

Ranjit Swain 等人提出了一种功能最小化技术生成测试用例 [7]. 该技术用状态机图来逐步生成测试用例, 覆盖了状态、状态迁移和行为. 通过对简单谓词决定的边界进行测试来使得测试用例的数量最小化, 并实现对迁移路径的覆盖.

1.3 测试目标

测试目标描述了测试用例所作用的系统. 基于模型的测试方法可以将早期的抽象 UML 设计模型、功能 UML 模型、实现的单元或完整的系统作为目标. 具体的, 在功能系统设计层如果测试用例是由早期的 UML 设计模型生成的, 测试目标即为功能 UML 设计.

文献[17,18]以实现作为目标. Mingsong[18]提出一种从活动图来生成测试用例的方法, 并在 Java 实现环境中执行. Javed[17]从顺序图生成测试用例, 提供一种使用 xUnit 实现测试的方法.

其他研究人员以系统的 UML 模型作为测试目标来检测设计是否失效. 文献[19]使用自动生成的中间结果来模拟目标运行过程, 文献[20]使用 TTCN-3 标准在系统上执行测试用例, 文献[21]提出了一种对单元至整个系统实现各层级进行测试的方法.

2 基于模型的测试工具及对比

目前有许多开源的和商业的测试工具支持 MBT 技术. 最先支持 MBT 技术的商业工具是发布于 1996 年的 IBM Rational 和 Microsoft 的 Visual Studio 集成测试工具. 根据 Robert V. Binder 在 2014 年进行的调查[11], 在使用 MBT 测试工具的工程人员中, 53%使用商业工具, 30%使用自研工具, 17%使用开源测试工具. 在商业工具中 Spec Explorer、Matelo 和 Conformiq Tool Suite 的市场占有率分别为: 27%、20%和 13%.

本节对市场占有率处于前三的工具进行简要介绍并对其功能进行对比.

(1) Spec Explorer

Spec Explorer 是微软发布的一款与 Visual Studio 紧密整合的 MBT 工具. 用户可以通过 Spec Explorer 对一个软件系统的期望行为进行建模, 并自动生成能够在 Visual Studio 测试框架下运行的测试代码. 模型可以用当前主流的程序设计语言 C#开发, 然后通过 Cord 语言脚本对模型进行配置和裁剪.

(2) Matelo

Matelo 是一个基于马尔可夫链模型的测试工具, 来源于欧盟的一个项目. Matelo 基于马尔可夫链模型和静态测试方法来自动构建测试用例, 支持结构分析并生成质量报告, 可以在软件开发、测试过程中精确

© 中国科学院软件研究所 http://www.c-s-a.org.cn

评估软件的可靠性和性能. 当生成状态机图后, 可使用用户定义的方式自动计算状态之间的迁移, 进而对可靠性进行评估.

(3) Conformiq Qtronic

Conformiq Qtronic 是一个用于嵌入式软件开发、测试的 MBT 工具. Conformiq Qtronic 基于 UML 模型, 关注自动化测试、执行和分析, 使用 C/C++、C#和 Java 作为建模语言. 根据测试部署的 API 插件可以采用不同的方式执行测试. Conformiq Qtronic 测试生成器使用 UML 状态机作为测试模型, 自动执行测试、生成报告. 确定了 UML 状态图后, Conformiq 会进行模型分析, 生成测试用例, 对模型各个方面进行覆盖. 此外, Conformiq 还可以对非确定行为生成测试用例.

下面对三个主流的自动化测试工具及其特征进行比较(表 1).

表 1 自动化测试工具比较

	Spec Explorer	Matelo	Conformiq Qtronic
模型	FSM、ASM	UML、MSC 状态机	UML 状态机
特征	测试生成、测试执行、测试结果检查、覆盖分析	测试生成、测试执行、测试结果检查、覆盖分析	测试生成、测试执行、测试结果检查、覆盖分析
测试覆盖	基于 FSM 的测试覆盖率分析	基于模型的测试覆盖率分析	REQ、基于状态、分支和边界
测试层级	单元、系统测试	单元、系统测试	单元、系统测试
应用	.Net 组件、Web Service	各类应用	嵌入式软件、电子交易系统
建模和测试语言	C#、ASML	TTCN-3	C/C++、C#、Java

3 总结和未来发展方向

综上对 MBT 技术的阐述, 目前大部分 MBT 工具关注于测试用例的生成和执行以及测试覆盖率分析. 这些工具使用抽象模型来表示基于使用场景或基于状态的行为. 随着软件技术的发展, 以上的这些理论和工具已经不能够满足不同系统应用的自动化测试需要. 下面对基于模型测试技术几个未来研究和发展方向进行总结:

(1) 开发更有效的测试模型

研究更多的具有革命性的模型用来支持面向测试的建模和分析, 使用基于非状态的软件行为和特征进行建模, 例如各种系统配置, 复杂结构等, 同时能够考虑软件的功能、性能及其他特性.

(2) 回归测试自动化

回归测试在软件生命周期中的重要性已经被一致认同. 为了有效地支持这一测试过程, 在 MBT 过程中需要支持回归测试的工具. 目前大多数测试工具还没有提供有效的针对软件变更和影响性分析的回归测试方案. 回归测试必须能够追踪软件相关的模型和测试集的变更及其影响, 以便可以正确更新测试集.

(3) 建立标准

目前还缺乏针对软件测试和测试自动化中建模和分析的标准. 比如, 不同的测试工具使用不同的动态模型, 会生成不同测试工作产品. 这涉及很多问题, 例如测试资源的重用、工具集成、测试脚本共享等. 因此, 在基于模型和自动化的测试中必须建立起一套统一的模型和工具标准.

参考文献

1 Kaur P, Gupta G. Automated model-based test path generation from UML diagrams via graph coverage techniques. IJCSMC, 2013, 2, 7: 302−311.

2 Rayadurgam S, Heimdahl MPE. Test-sequence generation from formal requirement models. High Ssurance Systems Engineering (HASE'01). 2001, 10. 23−31.

3 Offutt J, Liu SY, Abdurazik A, Ammann P. Generating test data from state-based specifications. The Journal of Software Testing, Verification and Reliability, 2003, 3: 23−53.

4 Abdurazik A, Offutt J. Generating test cases from UML specifications. George: George Mason University, 1999, 5, 45−54.

5 Samuel P, Joseph AT. Test sequence generation from UML sequence diagrams. Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing. 2008. 879−887.

6 Prasanna M, Chandran KR. Automatic test case generation for UML object diagrams using genetic algorithm. International Journal of Advances in Soft Computing and its Applications. 2009,7, 1(1):19−32.

7 Swain R, Panthi V, Behera PK, Mahapatra DP. Automatic test case generation based on state machine diagram. International Journal of Computer Information Systems, 2012

© 中国科学院软件研究所 http://www.c-s-a.org.cn

4(2): 99−124.

8 Pretschner A, Prenninger W, Wagner S, Kuhnel C, Baumgartner M, Sostawa B, Zölch R, Stauner T. One evaluation of model based testing and its automation. ICSE ’05 Proc. of the 27th International Conference on Software Engineering. 2005. 392−401.

9 Pretschner A, Philipps J. Methodological issues in model-based testing. Model-Based Testing of Reactive Systems. 2005: 281−291.

10 Philipps J, Pretschner A, Slotosch O, Aiglstorfer E, Kriebel S, Scholl K. Model based test case generation for smart cards. 8th International Workshop on Formal Methods for Industrial Critical Systems. 2003. 168−192.

11 Binder RV. 2011 Model-based testing user survey-results and analysis. Robert V. Binder’s Technology and Businiss Blog. 2011

12 Binder RV, Kramer A, Legeard B. 2014 Model-based testing user survey: Results. Robert V. Binder’s Technology and Businiss Blog. 2014

13 Tassey G. The economic impacts of inadequate infrastructure for software testing. National Institute of Standards and Technology. 2002, 5: 2-3.

14 Leveson NG. Software: System Safety and Computers. Addison-Wesley Professional. 1995,5.

15 Abdurazik A, Offutt J. Using UML collaboration diagrams for static checking and test generation. Third Internationl Conference on the Unified Modeling Language. New York, UK. 2000, 10.

16 El-Far IK, Whittaker JA. Model-based software testing. In: Marciniak JJ, ed. Encyclopedia of Software Engineering. 2002, 1. 825−837.

17 Javed AZ, Strooper PA, Watson GN. Automated generation of test cases using modeldriven architecture. Proc. of the ICSE 2nd International Workshop on Automation of Software Test. 2007.

18 Chen M, Qiu X, Li X. Automatic test case generation for UML activity diagrams. International Workshop on Automation of Software Test. Ast 2006. Shanghai, China, May. 2006. 2−8.

19 Caverra A, Daves J, Thierry J, Mounier L, Hartman A, Olvovsky S. Using UML for automatic test generation. International Symposium on Software Testing and Analysis, 2002. 201−210.

20 Bouquet F, Grandpierre C, Legeard B, Peureux F. A test generation solution to automate software testing. The 3rd International Workshop on Automation of Software Test. 2008. 45−48.

21 Hartmann J, Vieira M, und Axel Ruder HF. UML-based test generation and execution. Siemens Corporate Research, 2004.

© 中国科学院软件研究所 http://www.c-s-a.org.cn