# 基于UML状态图的测试技术研究*)

章 涛 顾 庆 陈道蓄
(南京大学计算机系 南京 210093)

**摘 要** UML统一建模语言已经广泛应用于软件开发中,基于UML图的测试技术最近成为了一个研究的热点,其中对UML状态图的测试有着广泛应用前景。UML状态图是传统状态图的变体,增加了层次、并发、广播。所以UML状态图的测试方法建立在对传统状态机的测试方法的基础上,对层次、并发、广播的处理或转换。本文介绍了基于UML状态图的测试方法和测试工具的研究进展,最后讨论了未来可以研究的方向。

**关键词** UML状态图,软件测试

## A Survey about UML Statecharts Testing

ZHANG Tao GU Qing CHEN Dao-Xu
(Department of Computer Science, Nanjing University, Nanjing 210096)

**Abstract** UML (unified modeling language) has been widely used in software development. And how to testing UML statecharts is becoming a hotspot of research. UML statecharts is an improvement of classic Finite State Machine, plus hierarchy, currency, broadcast. So the test techniques of UML statecharts are based on classic test techniques of Finite State Machine and handling hierarchy, currency, broadcast. The article discusses the method based on UML statecharts testing and the current development of testing tools. At last, talk about the future work.

**Keywords** UML statecharts, Software testing

## 1 引言

随着当前的社会信息化过程高速发展,计算机和软件在国民经济和社会生活各方面的应用越来越深入。如航空、航天、交通、银行等系统中,软件的作用举足轻重,对它的质量提出了很高的要求。因为即使很小的软件错误,也可能造成整个系统的崩溃,造成极大的经济损失。而软件开发的各个阶段都要有人来参与,但人的工作和通信不可能完美无缺,出现错误也是难免的。同时,软件的功能和控制对象的复杂程度不断提高,软件的规模也在不断地扩大,对软件开发方法和生产的能力都有了更高的要求[1]。

人们提出了各种方法和技术来提高软件的质量。其中,以软件测试为中心的质量保障技术在软件开发的过程中得到了发展和应用,软件测试已经成为保证软件质量的关键技术之一。据统计,软件测试的工作量往往占整个软件开发周期的40%以上。随着软件规模的不断扩大,软件测试在整个软件开发周期所占的比重日益增大。

目前,面向对象的软件开发方法被人广泛接受。与传统的软件开发方法相比,面向对象引入了类、对象、继承等新的特性。由于面向对象中的继承、数据封装、动态绑定、多态等机制,使得面向传统软件开发方法为背景发展起来的测试技术,并不能完全适用于面向对象软件的测试。为此,人们依据传统的测试技术,研究出适合面向对象软件的测试技术,与传统的测试技术相比,如何生成和选择代表性的测试数据是面向对象软件测试的关键。

目前,UML(统一建模语言)已经成为事实上的面向对象标准建模语言,它可以定义系统的组织结构,包括系统分解的组成部分、它们的关联性、交互、机制和指导原则,包括了软件开发生命周期的整个过程。UML图可以全面地描述一个系统,从需求到设计,从一个类到整个系统。它定义了多种图元,从不同视角和层次描述系统的静态结构和动态特性。软件的UML模型可以作为测试该软件的依据,直接生成测试数据。怎样把UML图更加有效地运用到测试中,发挥它的最大的潜力,成为研究所要考虑的主要问题。

近些年对基于UML的软件测试技术的研究也很多[3~10],主要有基于状态图、协作图、顺序图三种图的测试方法。本文主要讨论基于UML状态图的测试技术。第2部分主要介绍基于传统的状态机的测试,第3部分介绍基于UML状态图的测试,第4部分介绍现有的自动测试工具,最后做一个总结。

## 2 基于状态图的测试

UML状态图是经典状态图面向对象的变体。人们常用它来描述对象、子系统、系统的动态行为。一般来说,它描述了面向对象类的一个对象在其生存期间的行为,具体描述了这个对象所有可能的状态以及由于各种事件的发生而引起状态之间的迁移。它在传统的有限状态机上增加了层次、并发和广播通讯机制,是一种有力的、灵活的状态迁移图。而基于UML状态图的测试大都以基于状态机的测试理论为基础。有限状态自动机是一种简单直观、理论成熟的规范模型。我们先讨论对有限状态机的测试。在讨论之前,先介绍一些相关的基本概念。

### 2.1 基本概念

*)国家863项目,2006AA012177,基于估算和可靠性技术的软件质量保障模型与工具。江苏省自然科学基金基础研究项目,BK2006115,基于复杂网络的测试覆盖策略研究。章 涛 硕士研究生,主要研究方向为软件测试。

**定义 1** 令 $M=(S,\Sigma,\Gamma,\delta,\lambda,s_0)$ 表示带输出的有限状态机，其中 $S$ 为非空有限状态集，$\Sigma$ 和 $\Gamma$ 分别为有限输入集和输出集，$\delta: S\times\Sigma\to S$ 确定下一状态，$\lambda: S\times\Sigma\to\Gamma$ 为输出函数，$s_0$ 为初始状态。

给定 $s,s'\in S, a\in\Sigma, b\in\Gamma$，若 $\delta(s,a)=s'$ 且 $\lambda(s,a)=b$，$a/b$ 表示从 $s$ 到 $s'$ 的弧，简记为 $s\xrightarrow{a/b}s'$；$L\in\Sigma\times\Gamma$ 代表输入/输出对，$L^*$ 是 $L$ 上的序列集。如果 $\exists s_1,\cdots,s_k,s_{k+1}\in S, k\geqslant 0, x=u_1,\cdots,u_k\in L^*$，则 $s_1\xrightarrow{u_1}s_2, s_2\xrightarrow{u_2}s_3,\cdots,s_k\xrightarrow{u_k}s_{k+1}$ 简写为 $s_1\xrightarrow{x}s_{k+1}$，称 $x$ 为从 $s_1$ 到 $s_{k+1}$ 的路径。

**定义 2** 设 $\Sigma$ 是有限输入集合，$V_1,V_2\subseteq\Sigma^*$，令 $V_1\cdot V_2=\{v_1v_2\mid v_1\in V_1\wedge v_2\in V_2\}$，$V_1-V_2=\{v\mid v\in V_1\wedge v\notin V_2\}$，则称 $V_1\cdot V_2$ 和 $V_1-V_2$ 是 $V_1$ 和 $V_2$ 的串联和差。$V^n=V\cdot V^{n-1}$。$V[K]$ 表示集合 $\{\varepsilon\}\cup V\cup V^2\cup\cdots\cup V^k$。

**定义 3** 如果 $\exists s,s'\in S, V\in\Sigma^*, \forall x\in V: \lambda(s,x)=\lambda(s',x)$ 称 $s$ 和 $s'$ 是 $V$ 等价，记为：$s\sim_V s'$，否则称 $V$ 区分 $s$ 和 $s'$；如果 $\forall V\subseteq\Sigma^*(s\sim_V s')$，称 $s$ 和 $s'$ 是等价的，记为 $s\sim s'$。

**定义 4** 给定一个有限状态机 $M$，令 $W$ 是一个输入序列的集合，若 $W$ 可以区分任意的状态对，$\forall s_i,s_j\in S: (\exists w\in W: \lambda(s_i,w)\neq\lambda(s_j,w))$ 则称 $W$ 是 $M$ 的特征集(characterization set)。

**定义 5** 输入序列集 $W_i$ 是状态 $s_i$ 的识别集当且仅当对 $S$ 内任意状态 $s_j$，若 $i\neq j$，则存在 $p\in W_i p$ 使得 $s_i$ 和 $s_j$ 的输出不同。

**定义 6** 给定一个有限状态机 $M$，令 $Q$ 是一个输入序列的集合，若 $\varepsilon\in Q$ 并且 $\forall s_i\in S: (\exists p\in Q: (s_0\xrightarrow{p}s_i))$ 则称 $Q$ 是 $M$ 的状态覆盖集。

**定义 7** 给定一个有限状态机 $M$，令 $Q$ 是一个输入序列的集合，若 $\varepsilon\in Q$ 并且对于任意状态迁移 $s_i\xrightarrow{x/y}s_j, s_i,s_j\in S$，存在输入序列 $p\in Q\wedge p\cdot x\in Q$，使得 $\delta(s_0,p)=s_i\wedge\delta(s_0,p\cdot x)=s_j$ 则称 $Q$ 为 $M$ 的迁移覆盖集。

### 2.2 基于状态机的测试方法

基于状态机的测试，基本原理是：给定一个 FSM A 和一个 $A$ 的被测实现 $M$，需要检测 $M$ 是否是 $A$ 的一个正确实现。要根据一定的输入序列来判断 $A$ 和 $M$ 的行为是否相同。$A$ 和 $M$ 之间的不同点表现在：状态数目、迁移数目、输出和迁移函数的实现。对于这个问题包括三个基本的问题。

- 存在性：是否存在这样一个序列？
- 序列长度：如果存在，这个序列需要有多长？
- 算法复杂度：怎样判断存在性和产生这样一个序列？

文[13]是对基于状态机的测试的一篇很好的综述。

为了实现充分测试，定义下面的测试准则：

- 状态覆盖：状态图的每一个状态至少被访问一次；
- 迁移覆盖：状态图的每一个迁移至少被激活一次。

基于有限状态机测试方法主要有：

U 方法。U 方法首先为状态机的每一个状态得到一个识别序列，该识别序列叫做单一输入输出序列（unique input/output sequence，UIO）。对某个状态的识别序列可以把这个状态从其它状态中区分出来，然后根据该识别序列构造测试输入序列。但并不是所有的有限状态机都存在 UIO。如果状态机不存在 UIO，则无法构造测试输入序列。

D 方法。该方法首先对有限状态机构造区分序列（distinguishing sequence），区分序列是指对于状态机的每一个可能的初始状态，区分序列产生的输出都不相同。方法的关键就在于构造区分序列，然后根据该区分序列构造测试输入序列。该方法产生的测试输入序列数目较少，但并不是每一个状态机都存在区分序列，因此限制了该方法的使用。

W 方法。该方法基于状态机的状态识别集来构造测试输入序列。每个状态机只要是精简的，都存在状态识别集，因此该方法的适用性较为普遍。生成测试用例的步骤是：

(1)估计被测实现中的最大状态数 $m$。对某个有限状态机的正确实现应该与规约说明中的状态机具有相同的状态数目，这里的估计值 $m$ 就是为了能够检测出某个有限状态机的实现中存在的额外状态，因此估计值 $m$ 要大于或等于规约说明中的有限状态机中的状态数目 $n$。估计值的选取是根据设计进行猜测得到的。

(2)根据 $A$ 产生迁移覆盖集 $P$、特征集 $W$，构造测试序列 $TS=P\cdot Z$，其中 $Z=\Sigma[m-n]\cdot W$ 是被测实现的特征集。其中 $\Sigma$ 为输入集。

这种方法视被测实现为黑盒，判定两者的初态是否相等。如果有 $k$ 个符号，构造 $P$ 和 $W$ 需要 $n^2k$，所有测试序列的总长度不超过 $n^2\cdot m\cdot k^{m-n+1}$。虽然 $W$ 方法保证了全故障覆盖，但输入序列数目太多，在实际应用中使测试效率低。

WP 方法。为了减少测试用例的数目，Fujiwara 提出了用于确定性有限状态机的 WP 方法。上述方法的 $P$ 代表“部分”，即在识别状态时，WP 方法仅用状态识别集 $W_i$ 代替 $W$ 方法中的整个 $W$，其故障检测能力与 W 方法相等。WP 生成测试用例的步骤是：

(1)估计被测实现中的最大状态数 $m$，同时构造迁移覆盖集 $X$、状态覆盖集 $C$、特征集 $W$ 和状态识别集 $W_i$，形成测试基 $(C, X, W, W_i)$。

(2)检验在规约说明中的状态在实现中可被识别并验证其实现与规约说明中一致，同时从初始状态到这些状态所经历的状态迁移也被验证，并且检验实现中是否存在额外状态。该部分的测试输入序可如下构造：$T_1=C\cdot X[m-n]\cdot W$。

(3)验证在上面没有被验证的状态迁移，测试输入序列可构造如下：

$$T_2=(P-C)\cdot X[m-n]\otimes W=\bigcup_{p_1\in(P-C)}\{p_1\}\cdot\Big(\bigcup_{p_2\in X[m-n]}\{p_2\cdot W_j\}\Big)$$

整个测试用例集 $T=T_1\cup T_2$。其实，$T_1$ 内的测试用例基本是 $T_2$ 某些元素的前缀，所以整个测试用例集可以用 $T_2$。

WP 方法的改进方法。用 WP 方法产生的测试序列不仅仅覆盖了正确的状态迁移，并且对于被测实现中的多余的状态迁移同样覆盖了。多余状态上的状态迁移对于测试来说必定都是错误的，因此不必对其覆盖就可确认。根据这点文[12]提出 R-WP 方法，在 WP 方法的第三步，用 $T_2=(P-C)\otimes W$ 来构造测试序列。这样在 $m$ 值较大时就可以减少较多的测试用例。

D 方法、WP 方法测试过程分为二个阶段：第一阶段先检查每个状态都存在于被测实现中，第二阶段再检查被测实现中的每个迁移都被正确实现。U 方法和 W 方法是对 D 方法的扩展，D 方法是用一个输入序列区分所有状态，U 方法是用一个输入序列确定一个状态，W 方法是用多个序列区分所有

状态。WP方法是对上面几种方法的一般化，R-WP方法是对WP方法的一个简化。

## 3 基于UML状态图的测试

传统的有限状态机是平坦顺序的，无法描述复杂的系统。UML状态图引入了层次、并发、广播通讯，它主要刻画一些动态元素所有可能行为，如类实例在其生命周期内的行为动态变化过程或者实体之间的交互等。UML状态图的基本元素包括状态、迁移、动作和各种伪状态。

状态是对象在一定时期内的存在条件。UML的状态包括基本状态、复合状态和伪状态。基本状态不包括任何子状态。复合状态包括或状态和与状态，它们是一种状态的精华，允许包含多个子状态。当对象位于或状态时，实际位于其中的某一个子状态。当对象位于与状态时，它所有的子状态都是活跃的。伪状态包括初始伪状态(initial)、浅度历史(shallow history)、深度历史(deep history)、join、fork、条件分支(branch)和终止伪状态(final)。

迁移是对象对事件做出响应，从而改变状态的方式，它连接着两个状态：从一个源状态到一个目标状态。迁移结构是：

事件[条件]/动作[公式]

其中：

·“事件”是触发事件，“条件”是布尔条件，“动作”是执行的操作或者是引发的事件。引发的事件在状态机中广播。

·动作是对象状态机在某一点的原子行为。动作的执行不能被中断，动作可以出现在迁移中进入状态、离开状态时。

·活动是对象在某些状态执行的行为，可以被中断。在状态内，活动使用do关键字作为标记。一旦状态是活跃状态，将执行活动；若状态变为不活跃，在执行离开动作之前中止活动。

已有的测试方法很少考虑UML状态图的层次结构和并发结构，一般把它作为普通的有限状态机处理，以传统的基本状态机的测试方法(如W-方法)为基础来构造测试用例。Anyur[4]提出了基于UML状态图的测试用例生成方法，根据UML状态图所描述的对象行为规范定义一组覆盖准则，以生成测试用例。提出了传递覆盖准则、全谓词覆盖准则、全序列覆盖准则等四个测试准则，并且开发了测试用例的生成工具，跟Ration Rose结合可以直接从UML状态图生成测试用例。但它们所研究的UML状态图是不含层次和并发结构的。

当考虑层次和并发结构时，UML状态图模型 $M=(S, init, SH, \delta, L, EE, CE, AE)$[7]，其中

·$S$ 是非空有限状态集，$init \in S$ 是初始状态；

·$SH$ 是 $M$ 的状态层次结构；

·$\delta \subseteq 2^S \times L \times 2^S$ 是状态之间的迁移关系，$L$ 是迁移标记集；

·$EE$，$CE$，$AE$ 分别是事件集、条件集和动作集。

设被测实现的状态模型为 $M'$，基于UML状态机的测试原理是：通过一定的输入判断 $M$ 和 $M'$ 的行为是否相同。基于UML状态图的测试可以转化为判断 $M$ 与 $M'$ 是否同构的问题。$M$ 与 $M'$ 主要的不同之处包括：

·丢失状态(额外状态)。如果 $M$ 增加(删除)一定状态之后和 $M'$ 等价，称 $M$ 有丢失状态(额外状态)；

·丢失迁移(额外迁移)。如果 $M$ 增加(删除)一定迁移之后和 $M'$ 等价，称 $M$ 有额外迁移(丢失迁移)；这种类型的差异会导致整个对象行为发生变化；

·不正确动作。如果 $M$ 修改某个迁移的动作序列之后和 $M'$ 等价，称 $M$ 存在不正确动作；

·不正确激发事件。如果 $M$ 修改某个迁移的激发事件之后和 $M'$ 等价，称 $M$ 存在不正确激发事件；

·不正确布尔条件。如果 $M$ 修改某个迁移的布尔条件之后和 $M'$ 等价，称 $M$ 存在不正确布尔条件。

为了检测 $M$ 与 $M'$ 中的差异，要选择合适的测试方法，定义测试覆盖准则来选择测试用例，构造合适的输入序列。对复杂的UML状态图的测试用例的生成，要考虑到UML状态图的与状态、或状态，以及各种伪状态和层间的迁移。

为了实现充分测试，定义下面的测试准则：

·状态覆盖。UML状态图的每一个状态至少被访问一次。

·迁移覆盖。UML状态图的每一个迁移至少被激活一次。

·布尔覆盖。UML状态图的布尔条件各取true或false一次。

·谓词覆盖。UML状态图的所有迁移的布尔条件的每个谓词分别取true或false一次。

### 3.1 基于有限自动机的测试方法

因为UML状态图是有限自动机的变体，很多的测试方法直接把UML图转化成为有限自动机。转化的方法是先把UML图用LTS做形式化的描述，然后转化为有限自动机，再做模型检验或者测试[10]。

文[10]先把UML状态图转化为扩展的FSM，这样便可以展平层次和并发结构，再按照传统的测试方法来生成测试用例。UML状态图在某一时刻通常不只一个状态是活跃的。当前活跃状态实际上是通过一个状态树来描述的，这个状态树的根是顶状态，树的叶是简单状态，树干是复合状态。一个状态配置就描述了一个状态树，它表示了当前的活跃状态，而状态之间的迁移就可以用状态配置之间迁移来表示。

一个状态配置且满足下面的规则：1)顶状态属于状态配置；2)如果状态配置包含一个或状态 $s$，那么它一定包含 $s$ 的一个直接子状态；3)如果状态配置包含一个与状态，那么它一定包含 $s$ 的所有直接子状态。

下面给出一个咖啡出售机的状态图来说明。

在图中，状态 *COM* 是组成状态也是顶状态，它的子状态有 *OFF*，*ON*，*MONEY* 和 *ZERO* 等。状态 *ON* 是与状态，它被直接分解成两个正交连接的子状态 *COFFEE* 和 *MONEY*。按照状态配置的定义，它包括了四个状态配置，它们各自所包含的状态为：{*COM*, *ON*, *COFFEE*, *Not Empty*, *MONEY*, *ZERO*}，{*COM*, *ON*, *COFFEE*, *Not Empty*, *MONEY*, *NOZERO*}，{*COM*, *ON*, *COFFEE*, *Empty*, *MONEY*, *ZERO*}，{*COM*, *ON*, *COFFEE*, *Empty*, *MONEY*, *NOZERO*}。同理，状态 *COM* 包含了五个状态配置，如图1右图所示。这样UML状态图就成为这五个状态配置以及它们之间的状态迁移，即一个扩展的FSM。状态图的层次和并发结构被展平。文中提出了得到这五个状态配置以及它们之间的迁移的算法。最后通过传统的基于状态机的测试方法来生成测试用例。

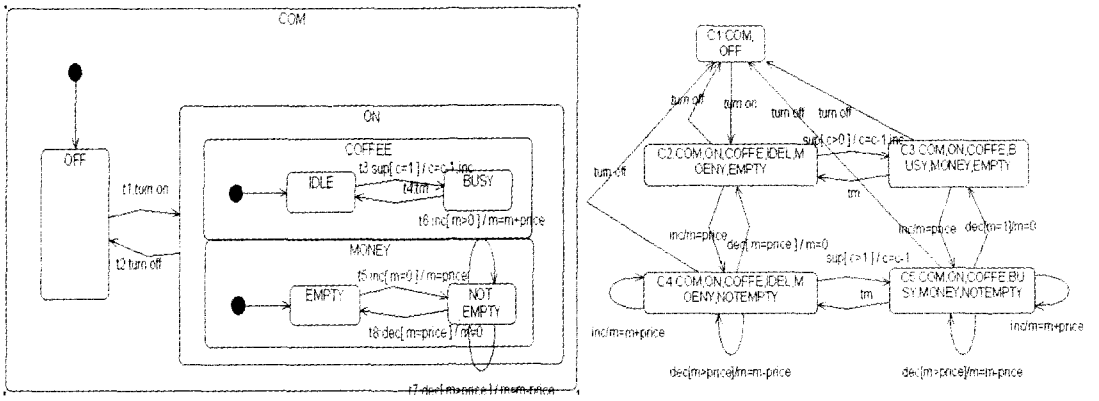


图 1

### 3.2 基于 WP 的测试方法

文[8]直接对层次和并发进行处理。视每个复合状态为抽象状态，分别构造主 UML 状态图和复合状态所对应的子 UML 状态图，再通过定义合适的合成规则构造整个 UML 状态图的测试用例集。

对于或状态，首先将每个或状态视为子状态图，构造每个子状态的测试基，再将或状态视为基本状态，构造整个状态图的测试基。根据规则合并这些子状态和主状态的测试基，用 WP 方法对得到的测试基构造测试用例。具体方法如下：

(1)将或状态视为抽象的基本状态，构造或状态所在的主状态图的测试基($C_{main}$，$X_{main}$，$W_{main}$，$W_i$)，$X_{main}$，包括进入或状态的迁移和从或状态的子状态出发进入主状态图的迁移。

(2)构造或状态所对应的子状态图的测试基($C_{sub}$，$X_{sub}$，$W_{sub}$，$W'_i$)。

(3)构造整个对象的状态图的测试基($C$，$X$，$W$，$W_{mi}$)，其中 $C=C_{main}\cup$｛主状态到达或状态的所有路径｝$\cdot C_{sub}\setminus$｛主状态到达或状态的路径｝，$W=W_{main}\cup W_{sub}$，$W_{mi}=W_i\cup W'_i$，$X=$｛｛$X_{main}$ 内目标状态是或状态的迁移｝$\cdot$｛或状态内初始迁移｝｝$\cup X_{sub}$。

(4)用 WP 方法构造测试用例。

对于与状态，当它是活跃的时候，它所对应的所有子状态也是活跃的，所以对与状态的处理和或状态不同。假设状态图中包含一个与状态，与状态包含二个或状态，分别构造二个或状态的测试基，再把这二个或状态的测试基合并成一个新的测试基。

(1)构造与状态的两个或状态的测试基($C_1$，$X_1$，$W_1$，$W_{1i}$)和($C_2$，$X_2$，$W_2$，$W_{2i}$)。

(2)$\forall j,k\in\{1,2\}$，$j\neq k$，$C_a=C_j\cdot C_k$，$X_a=(\{\varepsilon\}\cup X_j)\cdot(\{\varepsilon\}\cup X_k)$，$W_a=W_1\cup W_2$，$W_{ai}=W_{1i}\cup W_{2i}$。

(3)视与状态为抽象状态，构造主状态图的测试用例基($C_{main}$，$X_{main}$，$W_{main}$，$W_i$)。

(4)根据对或状态的处理用相同的方法构造测试用例。

对包括 join 或 fork 的状态图，将由 join 或 fork 构成的复合迁移作为层间迁移处理，单独处理这两个伪状态。文[9]证明了上述规则保证了全故障覆盖。相对于 W 方法，这样的方法生成了较少的测试用例。

### 3.3 基于 UML 状态图的统计测试方法

上面介绍的测试方法都是基于确定性的过程生成测试用例。文[10]提出了对 UML 状态图统计测试的方法。

软件统计测试与传统的软件测试不同，它要求基于软件使用模型产生测试例对软件系统进行测试。软件统计测试也称为软件可靠性测试，因为根据统计测试结果可以估计被测试软件的可靠性。

文[11] 基于统计测试的方法，提出一个基于迁移覆盖的测试准则。设 $S$ 为迁移覆盖集，$P$ 为每次执行 $S$ 中每个迁移的覆盖概率中的最小值。对于一个测试用例集，设有 $N$ 个测试用例，$N$ 必须足够大，使得 $S$ 中的每个迁移都可以被充分地覆盖。软件的质量定义为 $q_N$，表示 $S$ 中最不可能被覆盖的迁移在测试用例集中覆盖一次的概率。可以得到 $(1-p)^N=1-q_N$，如果 $q_N$ 已知，可得 $N=\ln(1-q_N)/\ln(1-p)$。

为了满足这一测试准则，怎样构造输入集的分布和生成测试输入集成为关键，文[11]中提出了 Functional distribution algorithm，算法的关键步骤是在执行路径中每次选择最少被覆盖的迁移来生成测试序列。文[11]中最后叙述了开发相应自动测试工具的设计思路。

文[11]并没有展平 UML 状态图，测试准则也只是简单地考虑了迁移覆盖。基于 UML 的统计测试还有很多工作可以做。而统计测试必将成为软件测试研究的新的热点。

### 3.4 测试工具的开发

在本节中，对现有的自动测试工具进行介绍。现有的测试工具基本上都是一个原型，实用性的测试工具还不多，主要都是提出了测试工具的类图，还没有实际的用于业界的产品。测试工具开发上应建立通用的、可重用的测试平台。实现自动确定集成测试顺序的工具和可以自动化的测试用例生成工具。能和现有的 Rose 工具紧密结合。

已有的工具如下表所示：

| 工具名称 | 工具开发者 | 基于的算法 | 是否与 Rose 结合 |
|---|---|---|---|
| UMLTest | Jeff Offott | 文中提出基于迁移覆盖和全故障覆盖的测试方法 | 是 |
| N/A | Li Liu Ying | 基于 w-，wp-方法 | 否 |
| Agedis Project | Agedis Project[14] | 把类图，状态图转化成中间表示(IF)，再作为模型检验和测试用例的输入 | 否 |
| N/A | Philippe Chevalley | 统计测试 | 是 |

**总结** 目前对 UML 状态图的测试面临的主要困难来自于两个方面：一方面为了达到故障的覆盖率，需要生成较多的

(下转第 280 页)

系统测试，但比起手工测试，自动化系统测试更加依赖于软件系统。另一方面，自动化测试系统本质上只是一种工具，工具只能判断实际结果与期望结果之间的差别。自动化测试结束后，必须对自动化系统测试的输出结果进行分析，判断其正确性。因此，确保自动化测试系统自身的质量比自动化系统测试本身更加重要。

(2)测试过程中的安全性

自动化测试过程的安全性问题包括两个方面的内容。一是用来验证集成在系统内的保护机制是否能够在实际中保护系统不受非法的侵入。二是在测试过程中，由于错误的关联性，并不是所有的组件缺陷都能够得以修复。某些组件缺陷虽然能够修复但在修复的过程中可能会引入新的缺陷。很多组件缺陷之间是相互矛盾的，一个矛盾的消失可能会引发另外一个矛盾的产生，如我们在解决通用性的缺陷后往往会带来执行效率上的缺陷。更何况在缺陷的修复过程中，还要受到时间、成本等方面的限制，因此无法有效、完整地修复所有的软件缺陷。因此评估软件缺陷的重要度、影响范围，选择一个折中的方案或是从非软件的因素(比如提升硬件性能)考虑软件缺陷成为实施自动化测试过程中需要重点考虑的问题。

(3)测试用例的选择

测试用例是指为实施一次测试而向被测系统提供的输入数据、操作或各种环境设置。测试用例驱动着软件自动化测试的执行过程，它是对测试大纲中每个测试项目的进一步实例化。无论测试自动化做得怎样成功和出色，如果测试案例本身是失败的，那么测试结果也变得毫无意义。因此，慎重选择测试用例是直接关系到自动化测试能否取得预期效果的先决条件。选择测试用例时，应当首先关注其代表性，即能够代表各种合理和不合理的、合法的和非法的、边界和越界的，以及极限的输入数据、操作和环境设置等，一个好的自动化测试用例必须具有较强的针对性，能够体现自动化测试的效率。其次要使测试执行结果的正确性是可判定的或可评估的，以后对同样的测试用例，系统的执行结果应当是相同的。

**结束语** 软件测试是一项复杂的工作，是一门综合的学科，需要不断地进行探索。由于基于组件的软件开发本身还存在一些未能解决的问题，需要不断发展和完善，因此对基于组件的软件测试这项工作更需要进一步研究。本文提出的基于组件的软件系统自动化测试过程模型充分考虑了软件开发过程与测试过程应该并行进行的思想，将不同测试阶段可以使用的各种自动化测试框架和方法整合起来，融入软件开发的全过程，有效地提高了组件软件系统的自动化测试效果。当然，组件软件测试标准虽然已经有了初稿，但其中的不完善之处是显而易见的，通用的组件软件测试方法也需要明确和进一步开发。所有这些问题决定了组件软件技术在展示它巨大魅力的同时，也向人们提出了一个一个的挑战。相信随着自动化测试和自动测试框架的成熟，开发和维护测试将节省出大量的时间用于提高测试覆盖率和测试本身的质量，从而大幅度提高基于组件的软件系统的运行质量。

## 参考文献


1 Marilani L. A Fault Taxonomy for Component-Based software:[Technical report]. University of Study Milano-Bicocca Italy, 2002

2 郑人杰. 实用软件工程(第二版). 北京:清华大学出版社,2003

3 张克东,庄燕滨. 软件工程与软件测试自动化教程. 北京:电子工业出版社,2002

4 马瑞芳. 计算机软件测试方法的研究. 小型微型计算机系统,2001,24 (12):2211～2213

5 King S, Hammond J, Chapman R. Is Proof More Cost Effective Than Testing? IEEE Transactions on Software Engineering, 2000,26(8):675～686


(上接第 267 页)

测试用例，如何减少测试用例生成的数量是测试方法主要研究的方面。另一方面，UML 状态图也在不断发展，不断有新的元素加入进来，对这些新加入的特性也要进行测试，对测试技术提出了新的要求。

基于 UML 状态图的测试是测试技术的一个热点研究课题，在实际中有着广阔的应用，发展潜力巨大。对它的研究也在不断完善中。本文是对现有的研究成果和应用的一个综述，主要讨论了基于 UML 状态图测试的理论基础、现有的测试技术和测试工具的开发。

## 参考文献


1 单锦辉,姜瑛,孙萍. 软件测试研究进展. 北京大学学报,2005,41(1):134～145

2 Chow T S. Testing software design modeled by finite state machines. IEEE Trans on Software Engineering, 1978, 4 (3) :178～187

3 Bogdanov K, Holcombe M, Singh H. An automated Test Set Generation for Statecharts. In: Lecture Notes in Computer Science 1641. Berlin: Springer, 1999. 107～121

4 Offutt A J, Abdurazik A. Generating tests from UML specifications[C]. In: Second International Conference on the Unified Modeling Language, UML'99, 1999

5 Offutt A J, Xiong Y, Liu S. Criteria for generating specification based tests[C]. In: Proceedings of Fifth IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'99), Las Vegas, Nevada, USA, October 18221, 1999. 119～129

6 Abdurazik A, Offutt J. Using UML collaboration diagrams for static checking and test generation. In: Proceedings of the Third International Conference on the Unified Modeling Language (UML '00). 383～395

7 Li Liu-ying, Qi Zhi-chang. Test selection from UML statecharts technology of object oriented languages and systems[C]. In: 1999, TOOLS 31, Proceedings, Sept. 1999. 273～279

8 Li Liu-ying, Wang Ji, Qi Zhi-chang. A test cases generation method for UML statecharts[J]. Journal of Computer Research and Development, 2001, 38 (6) :691～697 (in Chinese)

9 Li Liu-ying. Research and implementation of testing techniques for UML:[Phd dissertation]. National University of Defense Technology, Changsha, 2000(in Chinese)

10 Kim Y G, Hong H S, Bae D H. Test cases generation from UML state diagrams. Software, IEE Proceedings-[see also Software Engineering, IEE Proceedings], 1999, 146(4): 187～192

11 Chevalley P, Thevenod-Fossc P. Automated Generation of Statistical Test Cases from UML state Diagrams[C]. In: Proceedings of the 25th Annual International Computer Software and Applications Conference, 2002

12 Gnesi S, Latella D, Massink M. Formal Test-case Generation for UML Statecharts. In: Proceedings of the Ninth IEEE International Conference on Engineering Complex Computer Systems Navigating Complexity, 2004

13 Lee D, Yannakakis M. Principles and Methods of Testing Finite State Machines - A survey. Proceedings of the IEEE, 1996, 84 (8):1099～1123

14 The Agedis Project. The Agedis Home Page, 2003. http://www.agedis.de/index.html

15 张涌,钱乐秋,王渊峰. 基于确定有限状态机的测试输入序列的选取. 计算机研究与发展,2002(9): 1144～1150

16 董威,王戟,齐治昌. UML Statecharts 的模型检验方法. 软件学报, 2003(14):750～756

17 颜炯, 王 戟, 陈火旺. 基于 UML 的软件 Markov 串使用模型构造研究. 软件学报, 2005(16):1386～1393

18 王林章,李宣东,郑国梁. 一个基于 UML 协作图的集成测试用例的生成方法. 电子学报,2004(8):1290～1296