计算机系列教程

基于模型的软件验证与测试

郑 炜 编著

西北工业大学出版社
NORTHWESTERN POLYTECHNICAL UNIVERSITY PRESS

基于模型的软件验证与测试

郑 炜 编 著

西北工业大学出版社

图书在版编目（CIP）数据

基于模型的软件验证与测试/郑炜编著. —西安：西北工业大学出版社，2013.8
ISBN 978-7-5612-3767-0

Ⅰ.①基… Ⅱ.①郑… Ⅲ.①软件—测试 Ⅳ.①TP311.5

中国版本图书馆 CIP 数据核字（2013）第 202638 号

出版发行：西北工业大学出版社
通信地址：西安市友谊西路 127 号　　**邮编**：710072
电　　话：(029)88493844　88491757
网　　址：http://www.nwpup.com
印 刷 者：兴平市博闻印务有限公司
开　　本：787 mm×960 mm　1/16
印　　张：14.125
字　　数：280 千字
版　　次：2013 年 8 月第 1 版　　2013 年 8 月第 1 次印刷
定　　价：29.00 元

前　言

随着计算机应用的飞速发展，整个社会对计算机软件的需求在不断扩大；软件的规模不断地扩大，数量也飞速地增长。与此同时，软件质量却并没有得到显著提高，软件故障所带来的损失不容忽视，如何有效地提高和保证软件质量，成为计算机领域重要的研究课题。但是，当前的测试技术，存在很多缺点，直接影响着软件质量。

根据测试序列的来源，软件测试可以分为基于程序代码的测试和基于模型的测试两类。基于程序代码的测试通常采用白盒测试方法，主要用于程序的单元测试，可以提供较高代码覆盖度的测试序列。但基于程序代码的测试的一个最大缺点是无法确定“程序是否实现了软件需求”，并且跟具体的编程语言有关。基于模型的测试可以用在单元测试、集成测试和系统测试中，可以用来确认是否“程序正确地实现了其需求”。因为基于模型的测试可以发现程序没有实现的功能，并且跟具体的编程语言无关。同时基于模型的测试还为测试序列的重用提供了可能。

软件模型是对软件行为和软件结构的抽象描述。基于模型的软件测试的特点是在产生测试序列和进行测试结果评价时，都是根据被测应用的模型及其派生模型（一般称作测试模型）进行的。基于模型的测试最初应用于硬件测试，特别是被广泛应用于电信交换系统测试，目前在软件测试中得到了一定应用，并在学术界和业界得到了越来越多的重视。基于模型的软件测试要求充分理解被测软件。根据软件的需求构造可以用于测试的模型，其工作主要是根据测试目的确定测试对象和测试特征，针对被测软件的相关属性建立相应模型。

基于模型的软件测试在近年来得到了较为广泛的关注，它以明确描述系统预期行为的抽象模型为依据，根据模型覆盖测试准则自动生成抽象的测试序列，自动产生测试脚本，执行测试并自动评价测试结果，实现测试过程的自动化。总体上说，这方面的研究还处于方法探索阶段，比较实用的工具非常少。但这种方法可以使得软件测试和软件设计以及实现同时进行，改变了软件工程中“现在编程，以后测试”的工作方式，使得在整个软件生命期中都可以并行地进行测试工作，从而极大地提高了软件开发的效率。

本书试图在总结过去研究工作的基础上，提出了新的方法并进行了实验论证。全书分为六章。每章通过介绍具体测试的背景、研究现状，进一步提出了方法，并以实验进行了论证。

第 1 章介绍了基于模型测试的相关技术，阐述了基于有穷自动机的自动化验证平台的基本框架与各模块的功能设计，并给出复杂程度不同的模型进行模型转换，证明模型转换的正确性。

第 2 章介绍了 TTCN－3 相关技术，包括 TTCN－3 核心语言和 TTCN－3 测试系统的系

统架构，以及讲解 TTworkbench 的功能和模块划分。具体论述了将 TTCN－3 改造为可用于实时性测试的方法，通过一个抽象化具有代表性的例子讲解 TTCN－3 改造为实时性测试的一般过程。

第 3 章通过对 Web 服务进行基于树模型的形式化描述，将其转化为测试树，应用树模型相关理论，给出 Web 服务测试用例生成算法，并使用 SOAP UI 调用测试用例，实现 TestCase 自动执行及结果报告。最后将该方案在实际后台支付网关 Web 服务测试中进行应用，并给出测试结果及其分析报告。

第 4 章介绍了几种谓词统计的故障定位技术以及人们在其基础上的各种改进，分析了 SOBER 算法的原理和特点并且对其进行了启发式算法优化，提高了其错误定位的覆盖率。并利用 Siemens Suite 提供的测试用例对优化后的算法进行了验证。

第 5 章介绍了回归测试的背景及研究现状，并通过不同版本的程序得出的数据，验证了 faulttracer 的通用性及对错误定位有着实际指导意义。

第 6 章对组合测试常用的参数配对组合覆盖方法进行了归纳与整理，并分析比较了各种方法的优缺点，同时介绍了 AETG 算法和 IPO 算法的改进部分以及用实验验证了基于组合测试的故障定位。

本书由郑炜编著。王婧、钱志刚、高晶、张靓、郭凯源、蔡璐、邹鹏里、刘志强、靳如一等在成书的过程中进行了大量辅助性工作，在此表示衷心的感谢！

由于水平有限，书中难免存在疏漏，敬请读者批评指正。

编　者

2013 年 7 月 5 日

目　录

第1章　基于FSM的软件测试

第1节　研究基础

1.1　背景

计算机软件系统的复杂性日益增加，不断增大的软件规模和对软件功能要求的提高也使得错误和缺陷发生的可能性大大增加。保证软件质量的手段有很多，软件测试就是其中最重要的一种。

据统计，在2000年，不充分的软件测试过程使得美国花费了590亿美元，测试工作已经占据了整个开发过程的40%～75%，测试过程中发现的与失效和需求有关的缺陷占到了40%～50%。传统的软件测试方法导致在软件开发过程的后期才发现大量缺点，带来大量返工，代价昂贵。因此，在开发的过程中越早地介入测试就能够越早地发现缺陷，从而减少测试代价和测试费用。

近些年，基于模型的软件理论发展迅速，基于模型的工程(Model Driven Engineering，MDE)被提出用于开发和维护复杂的计算机软件系统。基于模型的架构(Model Driven Architecture，MDA)就是MDE的一个例子。MDA在软件开发中的发展直接影响并促进了基于模型的测试(Model Driven Testing，MDT)的产生。MDT的提出将软件测试自动化理论推向了一个新的阶段，将自动化测试带入到软件开发流程更早期的环节中，便于更早地发现软件需求相关的缺陷，降低后期返工开销，为降低软件测试成本提出了新的思路。

为了适应软件系统的快速开发、部署和维护，保证软件系统质量，自动测试的实现势在必行。一般来说，企业对于自动化测试的要求往往集中在自动化软件测试管理流程，以达到始终一致的软件质量和可量化的、可衡量的测试过程管理；通过实现测试自动化，以提高测试案例的复用和实现内部标准化，从而提高测试效率。这就要求，自动化测试平台能够适应不同的工作环境、编程语言，拥有尽可能多的测试功能，以期对项目进行测试和验证。

目前，软件自动化测试主要从两个方面着手：基于模型的测试研究和基于代码的测试研究，因此，设计的测试方案也主要侧重这两个方面。

一方面，基于有限状态机的测试模型假设软件在某个时刻总处于某个状态，并且当前状态决定了软件可能的输入，同时，从该状态向其他状态的迁移决定于当前的输入，根据测试模型

对被测试程序中所有输入及其状态组合进行系统枚举，可以使测试实现系统化、集中化以及自动化。有限状态机模型特别适用于把测试数据表达为输入序列的测试方法，并可以利用图的遍历算法自动产生输入序列。

另一方面，与基于模型的测试不同，基于代码的测试是对已经编码完成的软件部件或系统进行测试，包括单元测试、集成测试、系统测试以及回归测试等。在这里，主要采用测试桩和测试驱动的策略。

测试驱动包括数据生成器和数据库操作模拟器，提供多样化的数据产生方式，如，从测试资料库选择，手动设定，根据给定的模式，正则表达式或者数据类型等。

当被测单元或者模块需要调用另一个过程时，需要打桩。这种可能性有两种，使用真实的过程或者用一个特定版本(桩)代替。如果该过程存在，使用真实的过程可最大限度地减少人的精力。然而，如果被调用的程序不存在或者不可用，使用桩是很有效的。由于被调用函数的故障或缺陷能够被排除，因此，桩能够更好地控制被测单元。这里要做的是，针对单个的函数，设计实验测试桩的自动生成。

另外，还要有结果分析模块。在软件被测单元按照测试用例给定的输入执行后，用输出结果和预期结果进行比较，完成结果分析，并且，根据不同规则的覆盖原则手动或自动生成结果报告，为软件的质量提供强有力的数据。期望能为软件自动化测试设计并实现一套完整的解决方案。

1.2 现状简介

1.2.1 基于有穷状态机的自动化测试以及相关技术的研究现状

(1)基于模型的软件测试现状。基于模型的测试最早开始用于硬件系统，广泛应用于对电信交换系统的测试，目前在软件测试中得到了一定的应用，并在学术界和工业界获得了越来越多的重视。

参考文献[6]中对近几年出现的 MDT 技术进行了总结和分析；Yuan 等在参考文献[7]中提出了一个对基于网络服务过程自动产生测试用例的方法，将 BPEL(Business Process Execution Language)和 UML 活动图用来定义测试过程(Process Under Test，PUT)；UML 测试标准(UML Testing Profile，UTP)与测试和测试控制标记法的概念被用于构建模型；UTP 标准为创建一个简洁的测试模型定义了一个框架，在这个框架之下可以为系统自动产生符合 MDA 的测试用例，并且可以完成一部分可能的转换，同时这样的测试模型还适用于单元测试、部件测试、集成测试和系统测试。

Baker 和 Jervis 提出了另一个相似的方法，除了为系统自动生成测试用例来保证编码的正确性以外，他们提出了一个在软件设计早期检验设计模型的机制。在他们的试验中，考量了时效和并发性。这个方法已经成功地用于很多项目。

Cavarra 等人和 Crichton 等人提出了基于扩展的 UML 自动产生测试用例的机制，而不再在 UTP 的基础上研究测试用例的生成。他们利用 UML 的特性，提出了两个基于模型的结构，一个是用扩展的类图、实例图和状态图来描述测试的特性，另一个是利用实例图和状态图的组合来捕捉测试指令(Test Directives，TD)。

(2)模型间转换研究现状。UML 模型的形式化在工业上已经引起广大的注意。The precise UML group 是其中一个最活跃的组织之一，这个组织是由那些对 UML 如何通过使用模型符号提供精确语义感兴趣的人构成的。Borges 等将 UML 类图和 OhCirus. Latella 等提出的描述 UML 元素的语言结合起来将 UML 状态图转换成 Promela 语言。Traore 等提出了 UML 状态图到可以自动化模型测试的 PVS 的转换理论。然而，却很少能找到研究 UML 模型到有穷自动机模型的研究者。Erich 等提出了表示状态图的层次化的有穷自动机模型，解决了层次问题，却没有提出 UML 模型到无层次有穷自动机模型的转换方法。Lai 等将时间扩展的 UML 状态图转换到时间自动机。

上海大学缪淮扣等成功地将 UML 模型转换到无层次的有穷自动机模型，其算法以层次化有穷自动机模型作为中间的处理模型，将 UML 模型先转换到层次化的有穷自动机模型，然后进而转换到扩展有穷自动机模型。上海大学陈圣波的转换方法虽然可以直接由 UML 模型转换到有穷自动机模型，但是对复合状态的支持有限，对并发性的处理有所欠缺。

(3)模型规则验证现状。确认是构建一个正确的系统，验证是保证构建系统的正确。需求工程在将有限状态机模型提交给设计阶段之前，必须进行模型验证，如果在验证过程中发现错误和缺陷，应及时进行修正和弥补。软件越早发现错误其维护成本越低。因此，对有限状态机模型在定制规则上的验证研究，可以为软件的后续开发节约成本和缩短时间。FSM 模型规则验证指的是采用某种方法对已存在的目标模型进行检查、校验，判断该目标模型是否符合某种规则。如果存在违反规则的地方，应该通过某种方式反映给用户。为了从根本上保证软件系统的可靠安全，包括图灵奖得主 A. Pnueli 在内的许多计算机科学家都认为，采用形式化方法对系统进行形式化验证和分析，是构造可靠、安全软件的一个重要途径。随着 UML 和有限状态机技术的不断发展，模型的思想深入到了软件开发和测试的各个阶段。对模型的验证的探讨和借鉴验证结果生成测试用例的研究成为了软件测试研究的一个重要方向。

(4)状态图提取研究现状。在状态图提取方面，大部分的研究内容都是从其他的一些非 FSM 模型元素向状态图转换。如 I. Khriss 等在 1999 年发表的文献中，从多重的协作图中提取出状态图；还有 Johann Schumann 等在 2000 年发表的文献中，提供了从场景模型提取出协作图，再向状态图转换的思路。由于协作图的格式决定了它们更适合在分析活动中使用，因而协作图特别适合用来描述少量对象之间的简单交互。随着对象和消息数量的增多，理解协作图将越来越困难。

还有一些研究人员将研究内容定位于从 FSM 中提取分层的状态图模型，也就是说他们采用了 OR 类型的复合状态。如 Hua Chu 等在 2006 年发表的文献中，先从场景模型(scenarios models)转换成 FSM，然后再通过提取 OR 类的组合状态来增加层次性。

我们很难找到一些从 FSM 中提取并行和分层状态图的例子。在参考文献[24][25]中 Ashish Kumar 提出采用 AND 型复合状态的状态图的方式来获得分层和并发状态图的模型的方法。

(5)测试覆盖研究现状。测试覆盖是软件测试的一种度量方法，可以针对不同的测试需求，选择相应的测试覆盖标准。软件结构测试是软件测试中的一个重要部分，它需要生成数据来覆盖程序中被选定的特定元素，如语句覆盖、分支覆盖、路径覆盖等。在软件结构测试中，测试员往往会选择有一定效果且开销较小的覆盖准则，最常用的就是分支覆盖准则。与语句覆盖相比分支覆盖具有更高的覆盖强度，与路径覆盖相比，开销更小却能达到合适的覆盖强度。分支测试技术是一种性价比最高的控制流覆盖方法，被大多数软件开发商所采用。

目前，在分支测试上已经形成了诸如 0－1 规划、变异分析(mutation analysis)、表约简(table reduction approach)、测试块划分和基于遗传算法的路径生成等方法，尤以 Bertolino 等人提出的基于非约束边的覆盖方法最为突出，该方法简单、实用且灵活，他们提出的测试路径生成算法 FTPS，首先由程序 DD 图来生成支配树和蕴含树，以求出非约束边集 UE，然后按一定的策略从非约束边集中选取一条非约束边，从支配树和蕴含树中产生一条由该非约束边对应的叶节点到根节点的路径，即为 DD 图的子路径(可能是非连续的边序列)，最后在由子路径划分的 DD 子图中递归地将边补齐，使其连续化成一条覆盖该非约束边的测试路径用例，并将路径覆盖到的非约束边从 UE 中去除。循环后两步直到所有的非约束边都被覆盖，便得到测试路径用例集，但是随着结构的复杂度增加，时间和空间开销将会很大。

传统的分支覆盖方法会产生多余的测试数据，从而降低了测试数据的测试效率；同时，此方法用于质量评估的过程中由于刚开始进行测试时，测试数据达到的覆盖率会很高，这使得在整个质量评估过程中，质量评估都是过高的，对于质量评估，显然用较低的覆盖率便发现同样多的问题更好。基于非约束边的覆盖方法在避免和改善了传统分支覆盖方法带来的缺陷的同时，由参考文献[32]："覆盖所有非约束边的路径集必定是一个分支覆盖路径集且为最小集"，可知基于非约束边的覆盖方法在寻找分支覆盖路径集时具备灵活、简单、效率高的特点，且对大型程序的处理能力好。因此我们在非约束边的覆盖方法的已有研究的基础上进行深入的探讨。

1.2.2 基于模型测试的特点

基于模型的软件测试技术大大提高了测试的自动化水平，模型的可复用性为测试用例的设计大大节约了时间和开销，同时在一定程度上缓解了测试失效辨识问题。面向对象开发方

法通常使用模型对软件系统规范进行说明，而这些模型中又含有很多可以直接被软件测试使用的信息，所以基于模型的软件测试可以把软件测试工作提前到开发过程的早期进行。基于模型的测试具体优点如下：

(1)基于模型的测试技术可以对不同类型的软件系统进行测试，例如并发系统、实时嵌入式系统等。

(2)由于模型是对系统的高度抽象，它突出了待测试系统的关键因素，忽略非关键因素，主次分明。

(3)模型可重用性高，共享性好。

(4)很多模型都具备较为完善和成熟的理论基础和基本技术，便于基于模型的测试顺利开展。

(5)根据模型还可能获得和输入相应的预期输出。虽然基于模型的测试具备以上一些显著的优点，但它的一些局限性也是不容忽视的。

(6)对测试人员的要求较高，需要测试人员具备一定的理论基础，他们必须熟悉所使用的模型及相关原理，同时，还要熟悉相关建模工具的使用、脚本语言，以及具备一定的编程基础。

(7)需要一定的前期投入，例如模型的选择、软件功能的划分、模型的构造等。

(8)模型的某些固定缺陷有时无法克服，比如状态爆炸问题。

(9)制定模型的测试覆盖标准是基于模型测试的一个重要问题。

(10)对于大型系统，创建、维护和管理模型也是需要考虑的问题。

基于模型的测试技术在软件测试领域得到了越来越广泛的应用，并在学术界和工业界都得到重视。针对模型在软件测试中的应用，基于模型的软件测试技术还要在模型本身、测试充分性、测试失效辨识以及测试自动化等方面加深研究力度。

1.2.3　基于模型测试的系统需求

(1)功能描述。根据 FSM 及其基于模型的测试特点，本书所涉及的测试平台具有以下的功能：

1)模型转换。

- 使用 ArgoUML 软件绘制 UML 状态图，并将其导出为 XMI 文档。
- 打开软件，在文件菜单栏下选择打开 UML 工程选项。
- 在弹出的主操作窗口中选择打开 XMI 文档，并点击“FSM 文档转化”按钮进行相应的转换。
- 点击 FSM 图形显示按钮，弹出主窗口并显示处理后的图形。

2)规则验证。

- 打开软件绘制 FSM 模型原型。
- 点击“状态机”菜单栏下的“模型规则验证”按钮，进行 FSM 模型规则验证。
- 阅读模型规则验证报告，根据规则验证报告，用户可以对 FSM 模型作适当的修改。

3)状态图提取。

• 对于给定的 FSM，选取合适的状态组。

• 选择“状态机”菜单栏下的“状态图提取”，开始对选定的状态组进行提取。

• 提取完成后，会弹出界面显示等价的状态组和转换条件，并可以通过该界面对提取后的信息进行保存。

4)模拟执行。

• 对状态图进行模拟，按照输入数据通过模拟器将当前的状态节点转换到下一状态节点，至到达终态节点为止。

• 若状态图是程序的基于改进的 DD 图，则可以生成基于分支覆盖的测试路径集合。

(2)数据流程图(见图 1-1)。

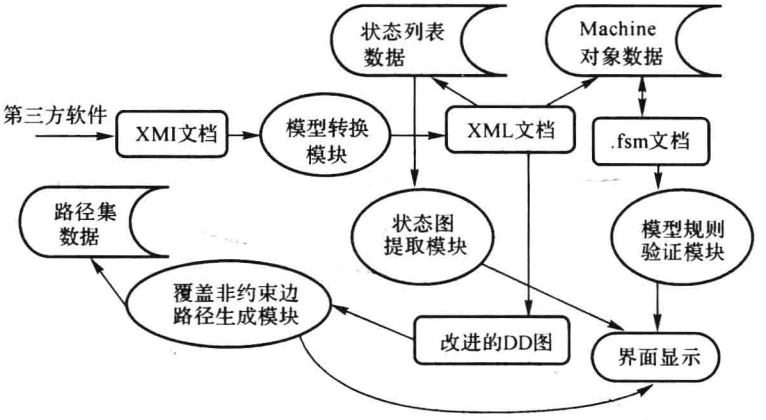

图 1-1　数据流程图

(3)用例图(见图 1-2)。

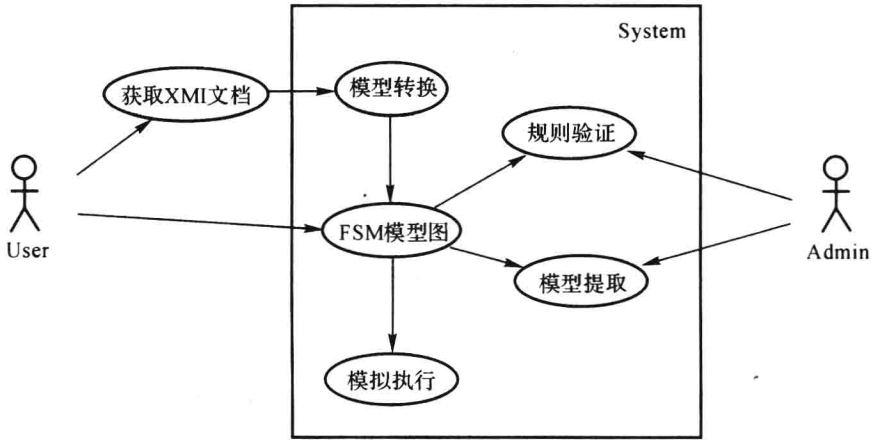

图 1-2　用例图

(4)关键用例(User case)(见表 1-1～表 1-4)。

表 1-1　模型转换测试用例

用例名称	模型转换
用例 ID	UC_001
描述	测试人员根据选择的 XMI 文档进行模块转换并显示结果
角色	Tester
前提条件	利用第三方软件将 UML 状态图导出为 XMI 文档
输入	XMI 文档
处理步骤	打开要进行转换的 XMI 文档 进行模型转换生成 XML 文档(.fsm 文件) 点击模型显示按键,将转换后的模型在主窗口上进行显示
输出	转换完成后的 FSM 模型图
异常	AutoTest. Exception
业务规则	
备注	

表 1-2　规则验证测试用例

用例名称	FSM 模型规则验证
用例 ID	UC_002
描述	测试人员根据需求对 FSM 模型进行规则验证,对规则功能进行测试,被测试模型应包含违反和满足规则的状态或转移
角色	Tester
前提条件	利用该软件绘制或转换的 FSM 模型
输入	被测试模型的 machine 对象
处理步骤	验证状态机模型是否存在二义性转移 验证状态机模型是否存在长度为 2 的重复路径 验证状态机模型是否有初态和终态 验证状态机模型是否存在异常状态 验证状态机模型转移是否完整 验证状态机模型所有状态、终态是否可达 验证计算模型路径长度的功能 验证所有的转移是否和状态连接

续 表

用例名称	FSM 模型规则验证
用户输出	以对话框的形式显示规则验证报告
异常	AutoTest. Exception
业务规则	测试人员根据规则绘制模型并进行规则验证功能测试
备注	

表 1-3 状态图提取测试用例

用例名称	状态图提取
用例 ID	UC_003
描述	根据测试人员选择的状态进行等价状态组的提取
角色	Tester
前提条件	被选取的状态应该有相应的并发性的说明
输入	选取的状态列表
处理步骤	选取状态组 选择状态图提取功能菜单
输出	弹出对话框显示等价的状态组和转换条件
异常	AutoTest. Exception
业务规则	
备注	

表 1-4 模拟执行测试用例

用例名称	模拟执行
用例 ID	UC_004
描述	测试人员根据选择的状态图的 xml 文档运行模拟器，检查运行结果；根据程序 DD 图的 xml 文档生成测试路径集合
角色	Tester
前提条件	基于改进的 DD 图的 xml 文档
输入	XMI 文档

续 表

用例名称	模拟执行
处理步骤	打开目标 XMI 文档 运行模拟器，输入转换数据，触发运行按钮运行模拟器。 点击路径集合生成按钮，生成分支覆盖测试路径集合
输出	分支覆盖测试路径集合
异常	AutoTest. Exception
业务规则	
备注	

第 2 节　设　计

2.1　测试平台的设计

软件模型是对软件行为和软件结构的抽象描述。软件行为可以用系统输入序列、活动、条件、输出逻辑或者数据流进行描述，软件结构则使用组件图、部署图等进行描述。针对测试任务，通过对软件功能和结构进行抽象并用易于理解的方式进行描述，获得的模型就是对被测试软件系统精确的描述，可以用于软件测试。一般对软件不同行为要用不同模型进行描述。

基于模型的软件测试技术可以根据测试模型对被测试程序中所有输入及其状态组合进行系统枚举，因此可以使测试实现系统化、集中化以及自动化。基于模型的软件测试技术属于基于规范的软件测试范畴，其特点：在产生测试用例和进行测试结果评价时，都是根据被测试应用程序的模型及其派生模型进行的，这些模型包括有限状态机模型、UML 模型、马尔可夫链模型等。

图 1－3 和图 1－4 分别是系统功能结构图和平台设计框架图。

本章选择有限状态机模型，原因如下：

(1)有限状态机技术非常成熟，便于理解和实际应用推广。

(2)图灵研究表明，世界上任何事物，都可以用有限状态机和无限状态机来表示。那么如果我们在一定条件下，将无限状态机转换为有限状态机，就可以通过有限状态机来解决所有问题。

(3)有限状态机生成测试用例比较方便可靠。

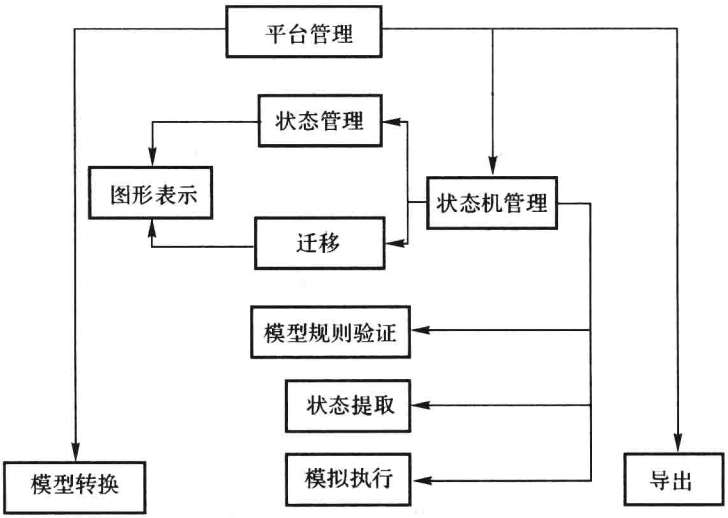

图 1-3　系统功能结构图

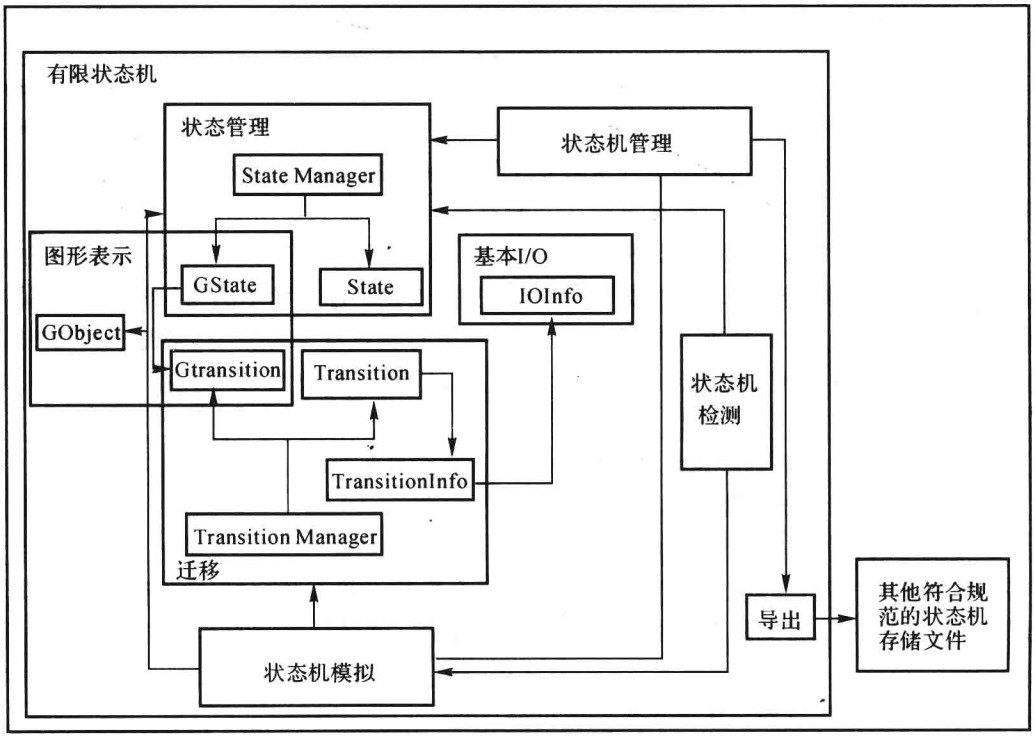

图 1-4　平台设计框架图