文章编号:1671-4598(2021)12-0022-05 DOI:10.16526/j.cnki.11—4762/tp.2021.12.005 中图分类号:TP319 文献标识码:A

# 基于模型的测试用例生成方法

蒲卿路,王月波,刘 涛,孙 云,李继秀
(西南电子技术研究所,成都 610036)

**摘要**:在大量软件出现的今天,除开软件的功能是否完善外,对软件本身提出了更高的安全性和稳定性要求;一款软件在上线前需要进行大量的测试,以便提升软件的质量;由于开发人员参与了软件的研发及上线流程,导致了看待软件问题的局限性,而测试人员在编写测试用例时,往往由于依据文档的不一致性,使得测试用例的价值大打折扣;并且在实际软件的开发流程中,测试环节与开发严重脱节;往往只是为了出相应的测试报告而去测试,偏离了测试的初衷;针对以上问题,提出基于模型的用例生成方法,能够基于工作流程图、判定表、状态转换等多种测试方法,并在该方法中应用边界值与等价类的思想,够贯穿整个软件研发的生命周期,在软件研发初期就能够参与测试,提出设计方案的不足;并且能够自动生成测试用例,提高测试人员的效率。

**关键词**:模型测试;边界值与等价类;工作流;判定表;状态转换图

# Use Case Generation Method Based on Model Testing

PU Qinglu, WANG Yuebo, LIU Tao, SUN Yun, LI Jixiu
(Southwest China Institute of Electronic Technology, Chengdu 610036, China)

**Abstract**: With the emergence of a large number of software, in addition to whether the functions of the software are perfect, higher security and stability requirements are put forward to the software itself. A piece of software needs to be tested a lot before it goes live in order to improve the quality of the software. As developers participate in the software development and online process, it leads to limitations in viewing software issues. When testers write test cases, the value of test cases is often compromised due to the inconsistency of the basis documents. And in the actual software development process, the test link is seriously out of touch with the development. It is often just to test for the corresponding test report, which deviates from the original intention of the test. In response to the above problems, the model-based use case generation method proposed in this article can be based on a variety of test methods such as work flow charts, decision tables, and state transitions, and the idea of boundary values and equivalence classes is applied in this method, which can run through the entire software R&D life cycle, you can participate in testing at the early stage of software R&D, and propose deficiencies in the design plan. And can automatically generate test cases to improve the efficiency of testers.

**Keywords**: model testing; boundary value and equivalence class; work flow; decision table; state transition diagram

## 0 引言

目前,随着科技的进步,软件的发展。各式各样的软件融入人们的日常生活中。购物,游戏,银行转账等活动都成为日常生活的一部分[1]。优秀的软件能够吸引大量的客户,为公司取得收益。同时在大量软件出现的今天,也对软件本身提出更高的安全性和稳定性。一款软件在上线前夕,需要进行大量的测试,以便提升软件的质量[2-3]。因此,对软件测试的方法,测试工具的研究为软件测试的质量带来了保证[4]。如何快速、高效地进行测试也是每一个测试人员及开发人员都必须重视的问题[5-7]。

目前大部分的测试用例都是由测试人员或者开发人员手工录入。针对同一类型的测试用例,其区别仅在于数据取值的不同。大量并且重复的测试用例设计,导致测试用例质量不高,不能够发现软件深层次的异常。同时,测试对软件开发的介入,往往需要有一定的代码基础才能开展。在软件需求验证阶段,传统的测试则不具有可行性。因此,有研究人员提出了基于模型的测试、优化和指导测试工作以及加速软件开发过程。

基于模型的测试,就是使用工作流程图、判定表、状态转换图描述出被测对象的实现逻辑。通过对上述模型进行路径的遍历,即可得出测试点。如果测试人员按照依据文档,建立模型时,发现模型不完整,逻辑混乱。也就代表软件的设计逻辑出现错误。使得软件开发过程与测试活动相结合[8]。在软件开发的后期,模型中各个路径的条件逐渐完整,使用参数定义,以及取值明确时,就可以在路径遍历的基础上,加入边界值与等价类的思想[9],将参数实例化,从而得出一个完整的测试用例。

目前测试行业主要存在以下问题:

1)测试与开发脱节。测试的开展需要有一定的代码基础才能进行,因此测试被至于软件开发周期的后半段。

**收稿日期**:2021-06-26; **修回日期**:2021-08-03。
**作者简介**:蒲卿路(1996-),男,四川成都人,硕士,主要从事软件测试与测试开发方向的研究。
**引用格式**:蒲卿路,王月波,刘 涛,等.基于模型的测试用例生成方法[J].计算机测量与控制,2021,29(12):22-26,32.

同时，大部分软件开发更新过程较快，测试人员开展测试工作时，被测对象的功能与测试需求已经严重不符。测试工作难以开展。

2）测试用例需要手动录入，如果一个测试项包含多个参数，测试用例的设计则需要对每一个参数的取值情况进行全组合。在没有用例自动生成工具的帮助下，测试人员的工作效率极为低下。

3）开展测试任务时，通常由多个测试人员同时开展测试分析与设计，容易造成测试数据的混乱。测试数据的可追踪性和恢复性也没有保障。

针对以上问题，本文基于模型测试，实现了测试用例的自动生成。有效地解决了上述软件测试中的种种问题，提高了测试人员的工作效率与被测软件的质量。本文后续内容主要对各个典型模型的生成测试用例的算法与数据结构的设计进行讲述。

## 1　测试需求模型构建

测试需求分析是进行软件测试的前提和基础。测试需求是指依据软件需求，根据测试目标而确定的被测软件的测试属性。测试属性是指软件测试人员结合测试意图和被测软件本身的特点分析得到的测试对象、范围和内容等要素。测试意图是通过软件的测试类型加以实现的，从不同角度验证软件的质量属性[10]。系统测试需求建模是指依据软件需求规格说明，从测试的角度出发，确定被测软件的测试属性，明确被测系统的输入输出及其映射关系[11]，建立被测系统的工作流程，判定表，状态转换等模型[12-13]。

如图 1 所示，传统的系统开发由一系列的文档构成，因此需要分析被测软件的相关文档，明确软件的测试属性并划分测试项[14]。软件文档作为软件的主要产品之一，为软件测试提供了主要的信息。从需求文档中可以提炼出系统的组成、功能、性能，软件的运行状态、运行时序，接口信息以及显性和隐性的软件质量属性等。依据软件功能进行测试项的划分，确定测试类型，并创建相对应的模型。

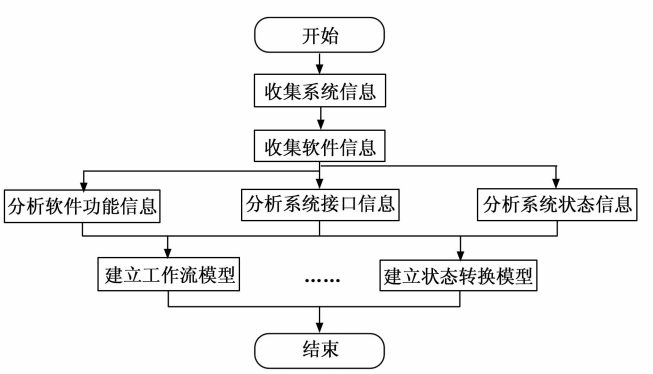


图 1　条件解析后生成的二叉树

## 2　基于工作流程图的测试用例生成方法

### 2.1　模型的创建

首先，将流程图中的不同节点类型统一划分为：开始节点、结束节点、动作节点、判定节点。所有流程图必须具有开始节点和结束节点，代表流程的开始与结束。动作节点，代表流程进行到此处时，需要完成指定的内容，例如设置参数 A 为 True，参数 B 自增 1 等。该内容的定义需要测试人员根据需求文档或者使用说明书，设计方案等进行创建。如果软件处于需求分析阶段，则该节点可不需要与具体参数挂钩，使用一段文字描述即可。判断节点，代表流程进行到此处时，需要根据某参数的具体取值，从而走向不同的下一个节点。

节点与节点之间的连线称为边。当该边的起点为判断节点时，则需要为边添加条件，代表该条件满足时，才可经过该边。这里与传统的流程图中的真假分支有一定区别。因为在软件实现中，逻辑的分支不一定为’if’和’else’，还需要考虑包括使用’else if ’的多个可选边存在的情况。因此在判断节点本身不存储具体的参数判定逻辑，而是在边上进行声明。

通过使用上述 4 种类型的节点的组合与边，即可完成模型的创建。

### 2.2　模型的路径遍历

工作流模型测试的核心思想是遍历从开始节点至结束节点的每一条路径，一条路径也就对应着一个测试点。如果工作流程中出现环的情况下，生成的路径中还需要覆盖到该环或者多个环组合的情况[15-16]。

为解决上述问题，决定采用有向图的数据结构，对用户创建的工作流进行底层数据的存储。每一个节点代表图中的顶点，节点到节点的路径中为图中的边。这样，就把用户画出的流程图转换为数据结构中的有向图。通过对图的遍历，即可完成工作流的路径遍历。

遍历路径时，首先使用深度遍历的算法思想，使用一栈或队列存储路径中的节点。从起点开始，遍历其下一个可到达的节点，并将其入队，再遍历该节点的下一个可达节点。并判断该节点是否为结束节点，或者没有下一个节点时，则返回，并将栈顶的节点出栈。如果已经到达结束节点，则将栈中的节点存储在最后的结果中，代表一条完整的路径。如此循环。利用上述思想能够得出图中的所有简单路径[17-18]（没有重复节点出现的情况）。但此方法不能将有环的路径遍历得到。因此还需要找出图中的所有环，遍历所有简单路径与环，查看路径中是否包含环的起点。如果有则将环放入一个临时的数组中，这样，就会把该简单路径中涉及到的所有相关环进行记录。最后使用排列组合的思想，将多个环组合后，插入至简单路径中。那么一条包含环的复杂路径即可生成完毕。其流程如图 2 所示。

### 2.3　条件的解析与测试用例生成

通过上述的路径遍历方法即可得出测试点。而对于测试点，对测试点中不同参数的进行取值，则可得出完整的测试用例。因此用例的生成需要将路径中的参数实例化，而在实例化时则需要运用边界值与等价类的思想。

然而在生成测试用例时，用户输入的条件可能不是简单的‘A＞0’。而是可能出现多个参数组合的情况，如

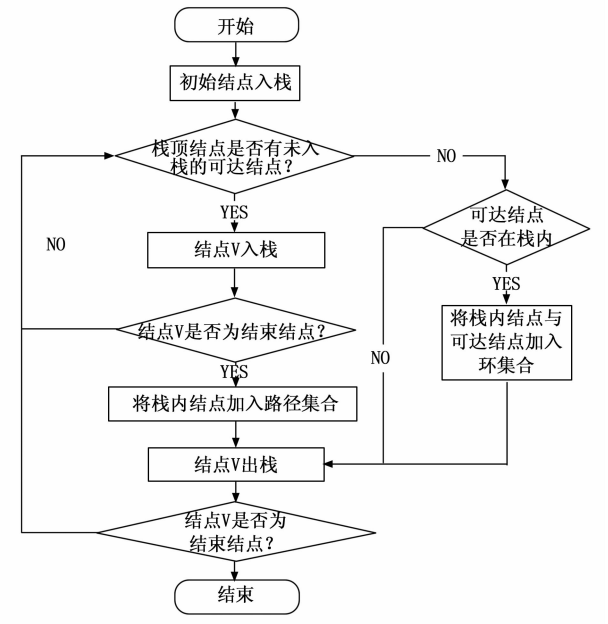


图 2 路径遍历算法

‘(A>0 && B>0) ‖C>10’。因此对条件的解析也是生成测试用例的难题。

为解决上述问题，决定采用二叉树与栈对条件进行解析。在解析具体的条件前，首先需要确定符号之间的优先级。本方法采用 C 语言的语法，用全局变量的形式存储两个符号间的优先级关系。在这里要注意的是，不同语言之间的优先级关系可能不一致，如‘a&1==1’，在 python 中会优先计算‘a&1’，但在 C 语言中会优先判断‘1==1’，因此为避免上述问题带来的干扰，建议测试人员在输入条件时加上括号，转变为‘(a&1) ==1’。这样，不管任何语言，都会优先计算‘a&1’。同时，同一个符号在不同语句中的意义不同，优先级的关系也就不同。例如‘a-b>0’，‘-a>0’。‘-’号在第一个表达式中代表减号的意义，第二个代表取反的意义，两个的符号的优先级在 C 语言中是不同的。又比如，‘*a’，‘a*b’，‘*’号，在第一个表达式中是指针的含义，第二个表达式则是乘号。

使用括号，对条件的语义进行限制后，即可开展条件的解析。本文采用两个栈完成条件的解析。

第一个栈用于存储操作符，第二个栈用于存储除开操作符的变量名或者数字。首先对条件表达式进行循环，依次判断该字符是不是操作符。如果是则将其入至操作符栈。并且在入栈之前判断是否为空，以及当前栈顶的操作符的优先级是否大于即将入栈的操作符。如果大于，则出栈当前栈顶的元素，创建一个新的二叉树节点，并将操作数栈中的数据连续出栈两次，将其作为该二叉树的左右节点。最后再将新节点入操作数栈。如果该字符不是操作符，则将其先加入一个临时变量中，并进入下一次循环。直到字符为操作符时，先把临时变量中的数据存入数据栈，并清空临时变量。再继续操作符的操作流程。因为用户定义变量名时可能是多个字符，例如 time，count 等，并不是简单的 a，b 的形式。所以需要一个临时变量，来获取完整的变量名，或者是变量的取值。

例如某条件表达式为‘A==0‖(A!=1 && A! =0)’。按照上述方法可以得出如图 3 所示。

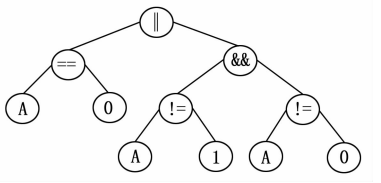


图 3 条件解析后生成的二叉树

由上述算法及图 1 可知，所有的参数只会出现在树的叶子节点上，参数节点的父节点即为判断的条件。参数节点的左右同级节点则为具体取值。得到具体取值后，则可通过边界值与等价类的思想，生成数据，也即为生成测试用例。

生成数据时，会根据参数的类型，步进，是否有固定取值（枚举）等情况综合进行考虑。在上述例子中，定义变量 A 为枚举型，他的取值只可能是-1，0，1。所以不用判断生成的数据是否合法。而是直接使用枚举值作为生成的测试数据。如是其他数据类型，如 INT（整形），则会将生成二叉树中的叶子节点中的数据提取出来。生成取值本身，取值加步进，取值减步进，3 个数据。如果用户没有定义参数的步进，那么默认是取 1。有时，当用户定义的参数类型为 UINT（无符号整形）时，并且取值为 0，那么自动生成的-1，将会引起错误。因此在生成数据时，还需要判定生成的数据是否超过了参数类型的取值范围，踢出无效的数据。提出无效数据的可以将无效数据代入条件中。因为在工作流模型中，如果要走某一条边，则必须使得该边的条件为真。为验证该条件在使用了具体取值是否为真。则需要计算出表达式最后的结果。解决上述问题其实也可以通过栈与树的形式去进行计算。但在此处，本方法使用了 python 的 eval 函数。该函数可将字符串转变为可执行的语句。因此原表达式‘A==0‖(A!=1&&A! =0)’，在替换后变成了“-1==0‖(-1!=1 && -1! =0)”（A 取-1 时）。通过 eval（‘-1==0‖(-1!=1 && -1! =0)’）可直接得出语句的真假。

有时在条件中有多个参数时，例如‘A>0‖B>0’，A 生成的数据有-1，0，1B 生成的数据也为-1，0，1 但是要是表达式成立，则是需要 A，B 两者进行组合。因此需要把参数进行组合后再次代入表达式中，才能得出合适的取值。而该组合的方法是程序自动进行全组合，不再需要测试人员手动进行组合。测试人员只需要对自动生成的测试用例进行适当的删减即可。

## 3 基于判定表的测试用例生成方法

### 3.1 模型的创建

基于工作流的测试模型分析完毕后，判定表的分析则

显得稍微简单，但也有不同之处。判定表由参数，规则，动作组成。使用表格的方式完成对参数，规则，与动作的定义。表格中的列可分为参数与动作。每一列中的不同单元格（cell）存储着参数的条件与动作的结果。

表格中的行，即为规则（rule）。可以理解为多个参数判定条件的组合。同工作流程图，在使用该模型时，也需要对参数的类型，步进等信息进行定义。测试人员在创建模型时，和工作流构建时相同，需提取文档中的相关信息后，构建判定表。

### 3.2　判定表测试用例生成

在进行自动生成测试用例之前，需要对 rule 中的条件进行组合。因为不同单元格之间是且的关系，需要同时使得单元格中的条件成立。所以在生成 rule 的条件时使用‘&&’操作符。如某规则生成的条件为‘Age＜＝25 && Gender ＝＝ Male && （Type ＝＝ Family ‖Type＝＝ mid）’。

最后将上述表达式使用同工作流条件解析的思想，将其解析为二叉树后，再提取参数及其取值，并利用边界值与等价类的思想，生成测试数据后，再使用 eval 函数去除不合适的数据。那么一条 rule 的测试数据即可自动生成完毕。

## 4　基于状态转换图的测试用例生成方法

### 4.1　模型的创建

状态转换图与工作流类似，可将状态视为工作流中的节点，状态至状态之间的转换，可以理解为节点至节点的边。但是状态转换图中并没有起始节点和结束节点的概念，只存在一个初始状态的节点[19]。并且状态转换图中测试点的概念也与工作流的的概念不同[20]。在工作流程中，从起始节点至结束节点的一条路径即为一个测试点。而在状态转化图中，测试的目的是测试不同状态之间是否按照规定的条件进行转换，并且可以存在状态之间的多次循环转换，循环转换次数需要由测试人员凭经验决定。往往在实际测试项目中，简单的从状态一转换至状态二并不会出现问题。但在循环转换时，状态一至状态二，状态二再转换至状态一，状态一再转换至转二时，会出现错误。其他情况同理。

### 4.2　模型的路径遍历

通过上述分析，直接使用状态转换图模型，采用深度遍历的算法，不能得出的正确的测试路径。本方法提出将状态转换图转换至状态转换树，得出状态转换树后再使用深度遍历的思想。因为状态之间可以循环转换，所以该树可以无限衍生下去。因此需要定义一个深度的概念，该概念与二叉树的深度不同，当所有状态都遍历到，所有状态至状态的边都遍历到时，此时生成的树，称为深度为 1 的树。如果测试人员想对循环状态转换进行测试，则可以继续对该树进行拓展，如进行带有一次循环状态测试，则该树需要向下再拓展一层。因此最后的测试点需要由用户决定深度后再去生成。一个典型的状态转换图如图 4 所示。

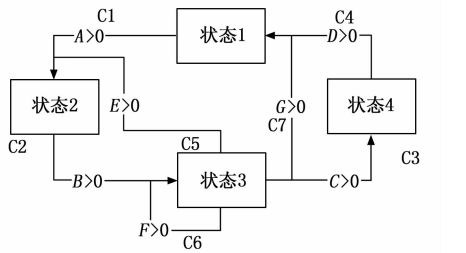


图 4　某典型状态转换图

定义状态 1 为初始状态。将状态图转换至状态树后，可得如图 5 所示。

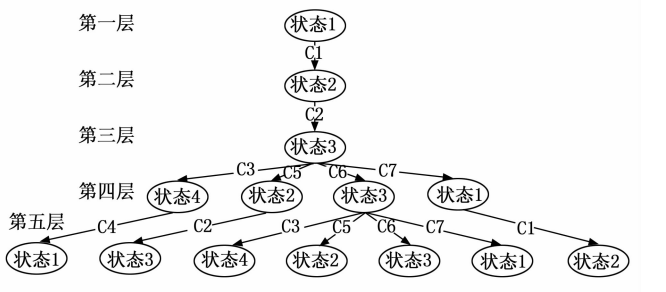


图 5　深度为 1 的状态转换树

将状态转换图转换为状态树后，则可开始测试用例的生成，生成方法与工作流类似。

在实际测试过程中，某复杂功能的实现，其够成的模型可能是状态转换图与工作流模型的组合。假如，上图中状态 1 转换至状态 2 的条件为某工作流。因此需要将工作流与状态转换图进行关联。最后生成的测试点是状态图本身测试点与工作流本身测试点的乘积。

## 5　实验结果与分析

为验证基于模型的测试用例生成方法的正确性与完整性，将该方法应用于实际项目中。某型号项目具有如下处理流程：

1）电台向显控盒发送工作参数。

2）显控盒是否收到 ACK/NAK。

3）如果收到 ACK，则判断是否收到工作参数。

4）如果收到 NAK 或者未收到 ACK，则电台重新发送。超过两次未收到 ACK，或者收到 NAK，则流程结束。

5）收到 ACK，并且收到工作参数，则进行合法性校验。校验成功，则保存至 FLASH，并修改参数，流程结束。校验不成功则回退参数，流程结束。

6）未收到参数，并且未超时，则继续等待接收参数。超时后，回退参数，流程结束。

### 5.1　实验步骤

1）测试人员根据上述流程绘制模型。并检查模型是否完整。

2）如果模型完整，则使用基于工作流的测试用例生成方法，生成测试用例。

3）测试人员根据自动生成的测试用例，执行测试。

4）测试人员采用手动设计测试用例的方法，进行用例的设计并执行。

5）将自动生成的用例与手动生成的用例进行对比，检查自动生成用例的完整性与正确性。

### 5.2 结果分析

上述流程经过建模后结果如图 6 所示。

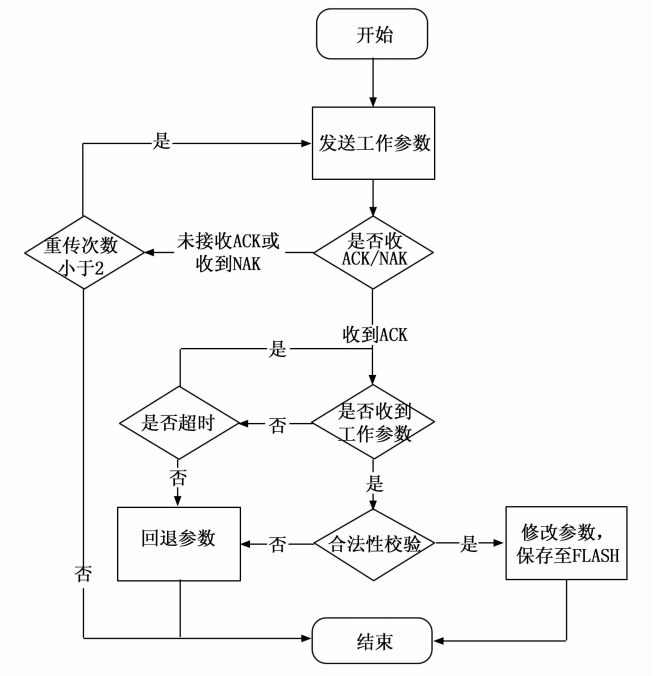


图 6 基于工作流建立的测试模型

模型建立完成，不缺少模型的必须组件，每个判定都具有相应的分支。则可得出设计流程正确。运用测试用例生成方法后，可得出简单路径 4 条，环 2 个。最终算法运用排列组合的思想，得出完整路径 14 条。也即为 14 个测试用例。其中一个测试用例为：开始→发送工作参数→是否接收 ACK/NAK（期望结果，收到 ACK）→是否收到工作参数（期望结果，收到参数）→合法性校验（期望结果，校验成功）→修改参数，保存至 FLASH→结束。其他用例不一一列举。测试人员根据上述用例，并按照流程执行，即可验证对应软件逻辑是否按照预期进行编写。

测试人员不借助算法，运用常用的用例设计方法手动设计用例 14 个。并将用例与自动生成的用例进行详细对比，每个手动设计的用例都与自动生成的用例内容相同。因此，该算法能够正确生成测试用例，并且在用例完整性，正确性上都得到了保证。

其他现有的测试用例自动生成方法，不能够用于判定表，状态转换图等模型的用例生成，并且，只对路径的遍历进行算法研究，没有对路径中判定分支的具体取值进行计算，不能给出经过边界值与等价类过滤后得出的可选取值。而在上述实验中，重传次数小于 2（count＜2），通过对表达式的解析，会获取出 2 的左右边界及自己（1，2，3），并将其作为分支的可选取值对自动生成的用例进行实例化，使得用例不再抽象。

当然，该流程在实际项目中属于简单的处理逻辑。当流程较为复杂，分支较多时，采用基于模型的用例生成方法，能够快速地生成完整的测试用例，不会因为人为的因素，或多或少地生成用例。并且整个测试的时间也会大幅度减少，并有助于理清测试人员的测试思路。提高测试效率。

## 6 结束语

随着面向对象的软件开发技术的不断发展和广泛应用，基于模型的软件测试，不仅可以把软件测试工作提前到开发过程，因此能够较早的参与测试，并发现系统设计的不足，减少了后期的维护成本与软件测试的成本。

本文分别针对工作流，判定表，状态转换图 3 种模型的测试用例生成进行了详细的说明。为解决测试行业当前主要问题提供一定的帮助与贡献。

后期将进一步对算法进行深入优化。如存在并发场景的测试模型，该如何进行测试用例的生成等。

**参考文献：**

[1] 冯婷婷. 基于 UML 活动图模型测试用例生成的研究 [D]. 南京：南京大学，2020.

[2] KAUR P，BANSAL P，SIBAL R. Prioritization of test scenarioderived from UML activity diagram using path complexity [C] //Proceedings of the CUBE International Information Technology Conference，2012 (1)：355－359.

[3] 章晓芳，冯 洋. 众包软件测试技术研究进展 [J]. 软件学报，2018，29 (1)：69－88.

[4] 杨 波. 软件测试自动化应用及研究 [J]. 工程技术，2016 (6)：244.

[5] 聂 鹏，耿 技，秦志光. 软件测试用例自动生成算法综述 [J]. 计算机应用研究，2012，29 (2)：401－405，413.

[6] 伦立军，赵辰光，丁雪梅. 软件测试充分性研究 [J]. 计算机工程与应用，2004，40 (3)：60－62.

[7] 卫延伟. 白盒测试系统的设计与研究 [D]. 武汉：华中师范大学，2013.

[8] 蒋夏军. 基于 UML 活动图及混合遗传算法的测试场景生成 [J]. 计算机工程与应用，2018 (7)：30－32.

[9] 兰 华. 基于 Z 规格说明的软件用例自动生成 [J]. 计算机学报，1999 (3)：963－969.

[10] 徐宏喆. UML 自动化测试技术 [J]. 航天控制，2010 (5)：64－69.

[11] 郝小蕾，徐神鹏，储 星. 基于需求模型的测试数据分析工具 [C] //中国航空学会、中国航空研究院. 第六届民用飞机航电国际论坛论文集，中国航空学会、中国航空研究院：中国航空学会，2017：5.

[12] 高 猛. 实时嵌入式软件系统测试需求建模研究 [J]. 航天控制，2010 (5)：64－69.

（下转第 32 页）