程序交互执行流程图及其测试覆盖准则*

刘 超

(北京航空航天大学软件工程研究所 北京 100083)

摘要 文章提出一种程序交互执行流程图 PIEF(program interactive execution flow chart),用于描述一个交互式软件的各种可能的交互执行过程.基于被测程序的交互执行流程图,进一步提出了功能测试的若干基本的测试覆盖准则、测试用例生成策略及其自动化方法.

关键词 软件工程,软件测试,交互式系统,软件测试准则,测试用例.

中图法分类号 TP311

有关软件测试方法和测试用例生成策略的研究已有许多重要的成果.[1~6]这些研究工作明确了软件测试的本质,归纳总结了软件测试的基本方法、准则、策略和技术,包括所谓的黑盒子方法(功能测试)和白盒子方法(结构测试).

结构测试是一种十分有效的测试手段.典型的白盒子测试,如语句覆盖测试、分支覆盖测试和路径测试等,针对程序内部的控制流结构制定测试的标准和测试策略.基于结构测试的软件测试工具可以直接捕获程序的执行情况,并自动记录和报告测试的覆盖率.比如北京航空航天大学软件工程研究所的软件分析与测试系列工具 SafePro[7,8],可以支持 C,C++,FORTRAN,Ada 和汇编等软件程序的分析与测试工作.

结构测试易于制定标准的原因在于它可以明确定义被测试对象的集合,比如模块中所有语句的集合、所有分支的集合、所有路径的集合等.但是,结构测试的主要问题是:

(1) 由于它仅关注程序的控制流结构,因而难于保证测试的充分性;

(2) 由于实际执行路径的数目通常是天文数字,使得结构测试一般仅限于单元测试阶段,而无法对整个系统运行过程中可能执行的"真实路径"的集合进行考查;

(3) 在面向对象软件中,常含有大量可再用的对象构件,而这些构件中的一些操作并非一定会在某个特定的应用中用到.也就是说,从系统的集成测试的角度讲,并非系统的每个构件中的所有操作或其路径都必须测试.因此,语句或分支覆盖测试准则不适合于系统的集成测试.

黑盒子测试方法则从系统或模块的外部要求和特性出发,从各种不同的角度对其进行"全方位"的测试,包括对基本功能、非法输入、边界和极端情况、时间和空间性能、兼容性、用户界面友好性的测试等.因而,功能测试的显著特点是:

(1) 与结构测试相比,考虑的因素更多、更全面,因而是保证软件测试质量必不可少的技术手段.

(2) 直接针对系统的各功能项,避免了单纯追求程序结构上的覆盖率.特别是对面向对象软件,加强测试的针对性可以减少大量与应用本身无关的冗余测试.

(3) 适用于从单一模块直至完整系统的任意级别的测试.

但是,功能测试一个最主要的问题是形式化程度低、规范性差,难以制定明确的测试覆盖标准和统一的测试策略.

一种典型的、比较规范化的功能测试方法是:首先根据功能需求制定一个测试大纲,详细明确地逐项列出所有需要测试的功能项和测试要点以及具体的测试步骤.然后依据测试大纲设计一组测试用例.当完成测试大纲规定的所有测试后,便完成一轮测试.采用这种测试方法,测试的充分性首先取决于测试大纲的完整性,即它对整个测试空间的代表性和覆盖程度.但是,由于系统需求规格说明书一般是用自然语言编写的,形式化程度低、规范性差、内容涉及面广,因而难以具体指导测试大纲的编制,更难以直接评判其完整性.

本文提出一种程序交互执行流程图 PIEF(program interactive execution flow chart),用以描述交互式软件的各种

---

* 本文研究得到国家航空科学基金资助.作者刘超,1958 年生,副教授,主要研究领域为软件测试,软件质量与可靠性,面向对象方法,软件工程环境.

本文通讯联系人:刘超,北京 100083,北京航空航天大学软件工程研究所

本文 1997-04-07 收到原稿,1997-06-12 收到修改稿

© 中国科学院软件研究所 http://www.jos.org.cn

可能的交互执行过程.基于被测软件的交互执行流程图,功能测试的一个基本准则是,要求通过设计各种适当的测试用例,对其进行测试,并达到对其交互执行流程图的一定的覆盖率.

1　测试用例空间及其交互执行流结构

所谓软件测试,是指通过以某种形式运行程序来检查其执行结果是否符合需求的一种技术手段.测试并不能证明一个程序的正确性,而是通过对选定的测试用例进行测试来试图发现程序中的错误,并通过纠正错误最终使程序的质量达到让人满意的程度.因此,测试的充分性是相对的.测试的关键问题之一就是要合理地设计一个有限的测试用例集合,最大概率地代表整个测试用例空间.

对于一个典型的批处理式程序,由于在其运行过程中不再有任何来自程序以外的操作和数据输入,所以其运行结果完全由程序的初始环境设置、参数和输入数据决定.假设一个批处理程序有 N 个输入参量,则其测试空间可以用一个 N 维的向量空间来表示.

但是,对于一个交互式程序,其执行状态和结果则由其初始状态(包括各种环境设置、原始数据和启动参数)、执行过程中输入的数据和操作及其发生的时间顺序来决定.如图1所示,其中 C_i 表示交互控制点,d_i 为导致其转向下一个控制点的输入数据或操作.我们称决定一个程序(或程序段)从其启动(甚至从其安装)开始,运行至一个可能的程序出错点为止的全部初始状态设置、输入数据和操作的序列为一个测试用例.例如,图1中的 $d_0d_1d_2d_3d_4d_3d_5d_6$ 是一个测试用例.对于一个程序的所有可能的(包括合法的和非法的)测试用例的集合,构成该程序的测试用例空间或测试空间.其中 d_0 为初始设置,$d_0,d_1,d_2,d_3,d_4,d_3,d_5,d_6$ 为输入数据及操作.

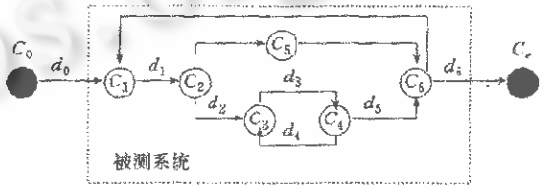

图1　交互式程序的交互执行过程

显然,这个测试空间是有结构的,并非一个简单的 N 维向量空间.其结构是由程序的各个控制点及其相互之间的转移关系(即时序关系)决定的.一个典型的交互式程序的测试空间可以表示为诸如 $D_0\cdot(*(D_1+D_2))\cdots\cdot D_{N-1}$ 的形式;其中"·"表示"串行","*"表示重复或循环,"+"表示"或",即 D_1 或者 D_2.

本文提出的程序交互执行流程图PIEF,用简明直观的方式刻画一个交互式程序的测试空间中各个控制点及其相互之间的控制转移关系.所谓程序的交互执行是指程序在执行过程中,其外部世界通过输入数据、实施某种操作或改变环境设置等方式来实时地控制和改变程序的执行流程和执行结果.

2　程序交互执行流程图 PIEF

定义 1. 程序交互执行流程图 PIEF 是一个有向图.其结点表示程序执行流程中一个可能的控制点,或称输入点,即允许程序以外的世界对其实施某种操作,包括输入数据、发出一条操作命令或改变外部环境设置等.图中的边表示从一个控制点向另一个控制点的转移,称作转移边.

控制点用"○"表示.控制点可带有属性信息,如控制点标识名、被控制的对象名、操作/输入模式、输入值域、时间参量(如延迟时间、允许等待时间)等.其中起始点和终止点在系统之外,用"●"表示.转移边用"→"表示.转移边也可带有若干属性信息,如转移标识名、转移条件、实现转移的输入数据或操作、与转移边所对应的程序段的功能说明等.图1即是一张简单的程序交互执行流程图.

定义 2. 层次化的程序交互执行流程图是程序交互执行流程图的扩充.其结点还可以表示一张子图,即另一张层次化的程序交互执行流程图.子图的结点用"⊙"表示,如图2所示.在层次化的 PIEF 图中,与子图相连的边除了一般转移边所带有的属性外,还需注明子图的内、外层边之间的对应或细化关系.

对于传统的过程型程序而言,其程序交互执行流程图 PIEF 与反映其程序内部控制流结构的程序控制流程图 CFC(execution flow chart)有着直接的对应关系.将传统的程序控制流程图做如下变换即可获得细化到源代码级的程序交互执行流程图:

(1) 在 CFC 图中标出所有控制点(即预先指明的各类"输入型"语句)的位置;

(2) 从图中内层子图开始,依次将不含控制点的子图用一条从其入口到出口的有向线段来取代;

实际上,程序交互执行流程图也可采用传统的程序控制流程图使用的图符.显然,对于一个结构化的程序,其程序的交互执行流程图也是结构化的,而将其转换成 PIEF 表示法也是直接的,只需将 CFC 表示法中的分支点和汇聚点分

© 中国科学院软件研究所 http://www.jos.org.cn

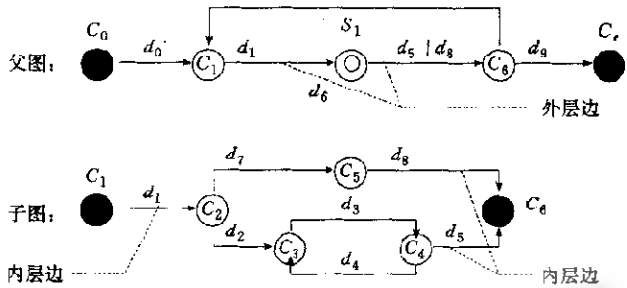

图2　交互式程序的执行控制流程图

别合并到其前、后端的控制点上.

定义 3. 并行程序交互执行流程图(Parallel-PIEF)是在上述串行程序交互执行流程图中增添了两种并行关系图元,以表示两个或两个以上的程序交互执行流程子图之间存在的并行关系及其相互之间的同步关系和约束关系.

(1) 并行关系

(2) 约束关系

图 3 是一个简单示例.

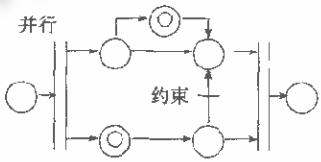

图3　程序交互执行并行关系流程图实例

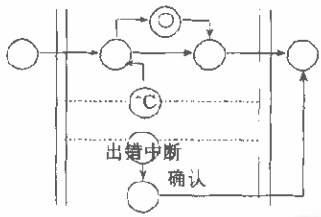

图4　程序交互执行流程图中的中断处理实例

中断处理也是一种并行关系,但是作为一类特出的并行关系,可用下列图元表示.

它表示中断可随时发生,以及当其发生时程序的执行流程被转移的情况.例如图 4 所示的实例.

在上述各种程序交互执行流程图中,忽略了所有程序的处理过程,如对数据的演算或加工过程、出错处理过程和中断处理过程等,也忽略了程序的处理结果,而唯一突出程序的"输入过程"或外界对程序执行的控制过程.这恰好突出了测试者在功能测试中,特别是在为其设计和选择测试用例时所要关注的重点.

有时,为了更明确地注明某个操作发生的位置,可以在图中的执行流程中加注一些辅助点,如信息的输出点、特定的加工或处理等.

3　功能测试准则

基于这样的测试空间,不难给出如下交互式程序功能测试的 4 个基本覆盖测试准则.

准则 1(控制点覆盖准则). 其 PIEF 图上的所有控制点至少要执行过 1 次.

准则 2(转移边覆盖准则). 其 PIEF 图上的所有转移边至少要执行过 1 次.

准则 3(逻辑路径覆盖准则). 其 PIEF 图上的所有路径至少要执行过 1 次.在运用此项测试准则时,必须对图中的循环作一定的限定,比如限定只考核两种情况,即不进入循环体和进入循环体 1 次或 1 次以上.经此简化后的路径称作逻辑路径或基本路径.

准则 4(代表值覆盖准则). 对于每一个控制点以及控制点上的每一个控制量,依据一定的测试策略选定一个有限的输入值集合,称作代表值集合,则每个控制点的代表值集合中的每一个值至少要被测试过 1 次.

© 中国科学院软件研究所 http://www.jos.org.cn

显然,在一般意义上,前3个准则的测试强度是依次递增的.准则1是最基本的.准则2是一个实用的测试覆盖准则.它完全包含准则1.准则2和准则4的结合是更强的实用准则.而准则3以及准则3与准则4的结合,则由于测试量上的组合爆炸问题限制了实用性.但是,对于高可靠性要求的系统,仍要对一些关键子系统和关键路径采用这类准则.

基于程序交互执行流程图PIEF的功能测试覆盖准则与基于传统的程序控制流程图CFC的结构测试覆盖准则之间有着某种对应关系,但并不能相互取代.

性质1. 对于一组测试,假如其语句覆盖率为100%,则其对PIEF图中的所有控制点的覆盖率亦可达到100%(准则1).但反之不然.

性质2. 对于一组测试,假如其CFC图中的分支覆盖率为100%,但它不能保证对PIEF图中的所有转移边的覆盖率也达到100%(准则2).反之也不成立.

这是因为,PIEF图中的一条转移边通常对应于CFC图中两个含有控制点的分支之间的一条路径,而分支覆盖不能保证一条特定的执行路径是否被执行过.同样,PIEF图中的所有转移边都被测试过,也不等于其CFC图中的所有分支都被执行过.

性质3. CFC图的分支覆盖准则加上PIEF图的转移边覆盖准则弱于CFC图的路径覆盖准则.这是显然的,因为PIEF图的转移边覆盖测试相当于对程序中的部分路径段的测试.但是,由于这些路径段常常是具有代表性的"关键"路径,代表了控制点之间的"直接的"转移,所以,PIEF图的转移边覆盖(准则2)可以看作是对CFC图上的分支覆盖测试的补充,同时也是对CFC图的路径覆盖测试的简化.单元测试中CFC图的分支覆盖准则加上系统测试中PIEF图的转移边覆盖准则以及代表值覆盖准则4,构成一组很强的实用的测试准则.

PIEF图实际上是一个用以刻画外界对系统的交互控制过程的系统静态结构模型.其控制点对应程序中的"输入语句",转移边对应"输入语句"之间可能的执行路径,而与程序动态运行状态无关.因而,它比用有穷状态机FSM(finite state machine)表示的程序的动态模型易于构造,并且可以通过分析源程序自动生成.

一般而言,CFC图为程序的单元测试提供了结构测试的覆盖准则.针对一个程序模块的测试,其控制量是程序内部的数据,因而需要设计专门的驱动模块和桩模块为其建立运行环境和提供相应的控制量.与之相比较,PIEF图更侧重于对系统交互过程的宏观描述.其控制点接受的是来自系统外部的控制数据,如键盘输入.在组合测试或系统测试中,由于PIEF图中的转移边对应程序中(跨模块)的典型路径段,因而它为系统的功能测试提供了实用的路径测试覆盖准则,同时又避免了针对整个程序内部的CFC图中的路径进行测试所引发的组合爆炸问题.

性质4. 基于PIEF和CFC图的各级路径覆盖测试及其并集与基于FSM的程序状态及其转移边的覆盖测试之间互不完全覆盖.这是显然的.对应PIEF图和CFC图的同一条路径,可以有不止一个值状态及其变换,比如,编辑器中的"Undo"选项可以处于"enable"或"disable"两种不同状态,其工具子选单中的选项内容会依据程序的状态而动态地改变等,尽管不同状态通常会引发不同的执行路径,但这不是必需的,它与程序的具体功能和实现有关.而另一方面,程序中可能含有死路径,因此即便所有的程序状态均被测试过,程序中仍可能有未被测试的路径.

4　测试用例及其设计

显然,对于交互式程序,其测试用例不再是一个N维数据,而是一个数据序列.根据程序的交互执行流程图,不难给出测试用例的定义.

4.1　测试用例

定义4. 一个交互式程序的测试用例由两部分组成:

(1) 该程序的交互执行流程图中,从起始点(通常对应程序的初始化)到图中某一个控制点的一条可能的路径,即沿此路径上的所有控制点的序列.此序列被称作一次测试的测试步骤;

(2) 在每个控制点上,对每一个输入量选定一个输入值.

假设在程序的交互执行流程图中,从起始点到图中某一个控制点的一条可能的路径上含有$N+1$个测试步骤(控制点),每个控制点的输入数据空间分别为$D_i, i=0,1,\ldots,N$,d_i是D_i中的一个值,则一个线性的数据序列$d_0d_1\ldots d_n$表示一个测试用例,其中d_0表示程序启动前初始状态的设置,$d_i(i=1,\ldots,n)$是对应程序的一次输入要求而给予的基本数据或操作.d_{i+1}可以与d_i同时发生或发生于其后.d_i中亦可含有有关其发生时间、运行环境状态变化等方面的信息.当被测试的应用中拥有多个对象时,d_i中还可含有被实施的对象的信息,比如对象的标识符等.

依据上述定义,一个测试用例$d_0\ldots d_n$同时又包含有若干个子测试用例$\{d_0\ldots d_i, i=0,\ldots,n\}$.因为,一般而言,

© 中国科学院软件研究所 http://www.jos.org.cn

每一次输入或每执行一步操作 d_i 都可能引发出程序的1个甚至多个错误.

4.2 测试用例的设计与生成

依据上述测试准则和测试用例的定义,测试用例设计与生成可分解为如下3个基本步骤:

(1) 生成程序交互执行流程图.

(2) 根据程序交互执行流程图设计测试步骤和编制测试大纲.测试大纲中详细列出了为达到对其流程图的预定的测试覆盖率而必须测试到的路径的集合.

(3) 设计并生成各个控制点上的代表值集合.对测试大纲中规定的每条测试路径中的每个测试步骤,对其每个输入量,设计选定一组有限的输入值集合,当测试沿着这条路径执行到该控制点时,从选定的代表值集合中以某种方式(如随机地或按其它策略)选定一个输入值,从而构成一个新的测试用例和一个新的测试.

4.3 非确定性关系的确定化

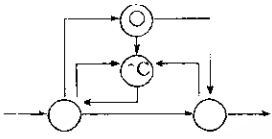

图5 关于^C的一张确定化变换子图

对于程序交互执行流程图中的非确定性关系,首先应将其确定化,即将一个随时可以发生的输入动作用1个或若干个在PIEF图中的确定位置上发生的动作取代,从而消除图中的非确定性关系.实际上,非确定性关系的确定化过程就是为其制定测试大纲的过程.例如,对于图4中的中断^C,假定我们希望对每两个控制点之间发生此中断的情况均进行测试,则我们可以将图中关于^C的并行关系转换成图5,称作关于^C的确定化变换子图.

5 结束语

传统的测试大纲是测试者用手工方式编制的内容冗长的文档.这种文档虽然可以详细规定针对系统的每一项功能的测试步骤,但是由于缺乏对测试空间整体结构的描述,无法客观地确定测试覆盖标准和评价测试大纲的充分性.事实上,正是由于缺乏功能测试的客观标准,在实践中,测试大纲内容的粗细程度随意性很大,很难有效地控制软件测试的质量.采用程序交互执行流程图方法,测试大纲的制定受到了程序交互执行流程图的规范,从而使得整个测试工作有法可依.PIEF方法已经在SafePro/C++'96版的系统测试中得到应用.我们对于该系统的交互式用户界面,依据其PIEF图制定了系统测试的测试大纲,从而保证了测试组在开发人员不直接参与的情况下,独立地完成对该系统的全面测试.

对于交互式软件,程序交互执行流程图为评判其功能测试的覆盖情况提供了一种客观的和规范化的度量准则.就功能测试而言,对程序交互执行流程图的覆盖测试是基本的和必需的.同时,绘制完整的程序交互执行流程图也为测试者编制测试大纲提供了客观依据和技术手段.对于一个复杂的交互式应用软件,通常可以根据其用户操作手册以及需求文档和设计文档方便地绘制其完整的程序交互执行流程图.基于现有的软件设计开发工具和源代码的逆向分析工具,亦可开发出相应的程序交互执行流程图的生成和分析工具.

参考文献

1 Myers G. The Art of Software Testing. New York: John Wiley & Sons, Inc., 1979

2 Beizer B. Software Testing Techniques. New York: Van Nostrand Reinhold Co., Inc., 1990

3 David C Kung, Jerry Gao, Pei Hsia. On regression testing of object-oriented programs. Journal of Systems Software, 1995,32(1):21~40

4 刘超,金茂忠.软件测试过程的基本模型POCERM.北京航空航天大学学报,1997,23(1),56~60
(Liu Chao, Jin Mao-zhong. Software test process model——POCERM. Journal of Beijing University of Aeronautics and Astronautics, 1997,23(1):56~60)

5 李健,金茂忠.对象状态测试.北京航空航天大学学报,1997,23(1):98~104
(Li Jian, Jin Mao-zhong. Object state testing. Journal of Beijing University of Aeronautics and Astronautics, 1997,23(1):98~104)

6 英伟,高仲仪.基于遗传算法的软件结构测试数据生成技术研究.北京航空航天大学学报,1997,23(1):36~40
(Jia Wei, Gao Zhong-yi. Research of software structural test data generation based on genetic algorithms. Journal of Beijing University of Aeronautics and Astronautics, 1997,23(1):36~40)

7 吴鹏程,金茂忠.基于对象模型的C++程序静态分析器.北京航空航天大学学报,1997,23(1):105~110
(Wu Peng-cheng, Jin Mao-zhong. C++ program static analyzer based on object relation diagram model. Journal of Beijing University of Aeronautics and Astronautics, 1997,23(1):105~110)

© 中国科学院软件研究所 http://www.jos.org.cn

8 徐红,钱红兵,陈曦. Ada 软件的动态测试技术研究. 北京航空航天大学学报,1997,23(1):18～24
(Xu Hong, Qian Hong-bing, Cher Xi. Study of dynamic testing techniques for Ada software. Journal of Beijing University of Aeronautics and Astronautics, 1997,23(1):18～24)

Program Interactive Execution Flow Chart and Its Coverage Test Criteria

LIU Chao

(*Software Engineering Institute Beijing University of Aeronautics and Astronautics Beijing* 100083)

Abstract A PIEF (program interactive execution flow chart) is proposed, which shows intuitively the interactive execution process of a program, that is, all the input points in the program with possible transitions among them. Furthermore, in terms of the PIEF, the author puts forward a set of coverage criteria for the functional tests, strategies for the test case generation and related automation methods.

Key words Software engineering, software test, interactive systems, software © 中国科学院软件研究所 http://www.jos.org.cn