第6卷第6期
1995年6月

软 件 学 报
JOURNAL OF SOFTWARE

Vol. 6, No. 6
Jun. 1995

基于结构化功能规格说明的测试方法和工具*

林 振 吴定一

(华东理工大学计算机科学系,上海 200237)

摘要 由于程序的形式化验证技术还局限于比较小的程序,软件测试仍然是目前和今后相当长一段时间内保证大型软件质量和可靠性的主要手段.测试大型软件是一项既繁重又复杂的工作,计算机辅助软件测试将会大大降低测试工作量,提高测试效率.本文首先提出一种新的、简单有效的基于结构化功能规格说明的测试方法,然后阐述如何基于该方法设计并实现一个测试工具环境,以提高测试者的工作效率,减轻测试者的负担.

关键词 软件测试,软件规格说明,计算机辅助软件测试,测试工具环境.

在软件的整个生命周期中,软件测试是相当重要的一部分,随着软件规模和复杂性的增加,软件测试的工作量越来越多、难度越来越大、效率越来越低.为了减少软件测试的工作量,提高软件测试的效率,寻找适当的测试方法并开发相应的测试工具显然是十分必要的.

到目前为止,绝大多数的软件测试工具所采用的测试方法是基于程序的,其优点是能针对程序结构进行调试查错,易于控制被测程序的测试覆盖度,其缺点是不便于从用户角度检查软件系统功能实现的完整性和正确性,但这正是软件测试的根本要求和终极目标.因而,基于程序的测试方法不能完全满足软件测试的要求.为此,人们提出了一些基于系统功能规格说明的软件测试方法,其中值得注意的测试方法有因果图法[1]和类属分割法[2].因果图法能够根据软件系统的功能规格说明进行逻辑上完整而无冗余的测试,但其前提条件是要把软件系统的功能规格说明用因果图表示出来,对于一个复杂软件系统,用因果图表示其功能规格说明是非常困难的.类属分割法把软件系统的功能规格说明用参数和环境条件类属表的形式表示,较为简单易行,但需要反复修改参数和环境条件类属表,测试自动化程度和测试效率低.

本文阐述了一种新的、更简单、有效的基于结构化功能规格说明的测试方法,并且进一步阐述了基于该测试方法的测试工具的设计与实现.

1 基于结构化功能规格说明的测试方法

1.1 基于结构化功能规格说明进行测试的可行性

* 本文1993-08-02收到,1993-11-18定稿

作者**林振**,1968年生,助教,主要研究领域为软件测试,程序自动生成.**吴定一**,1937年生,教授,主要研究领域为软件工程.

本文通讯联系人:林振,上海 200237,华东理工大学计算机科学系

在软件工程学发展的早期,各种软件开发方法尚未得到充分发展,软件系统的功能规格说明往往是用自然语言写成的,是非结构化的.因而,尽管功能规格说明在软件测试中起着重要作用,但难以应用功能规格说明作为测试工作的标准.

随着软件工程学的发展,软件开发方法得到充分发展,结构化分析(SA)和结构化设计(SD)方法被广泛采用.基于结构化分析和设计方法的软件功能规格说明具有很好的结构,可以作为测试工作的标准.如果系统开发时未采用结构化分析和设计方法,功能规格说明是非结构化的,则可用结构化分析和设计方法抽取信息作成结构化的功能规格说明.

软件工程学的发展要求人们直接把解反映到人对问题的认识上,面向对象的开发方法和面向功能的结构化分析和设计方法都有可能反映人们对问题解的认识.当然,对于某些应用项目的一个小部分,用面向对象的方法开发可能更好,但对整个应用项目来说,结构化分析和设计方法更成熟,更为大家所熟知,基于结构化功能规格说明进行测试在目前是比较实际的.

1.2　两种结构化功能规格说明的结合使用

结构化功能规格说明有两种:一种是需求规格说明,另一种是设计规格说明.使用结构化分析方法将得到前者,使用结构化设计方法将得到后者.仅仅基于需求规格说明的测试方法无法进行针对性的敏感测试数据的选择,往往需要大量的测试数据才能发现软件系统隐含的错误,测试效率低.仅仅基于设计规格说明的测试方法虽然能克服基于需求规格说明的测试方法的不足,但设计规格说明是设计者对软件需求的理解,若设计规格说明不符合软件的功能要求,则基于设计规格说明的测试活动即使成功,也不能说软件系统的功能实现符合用户的要求.

因此,结合使用基于需求规格说明和设计规格说明的测试方法能取长补短,提高测试效率.

1.3　基于结构化功能规格说明的测试的策略

基于结构化功能规格说明的测试方法必须具有以下先决条件:

(1) 已经开发出层次结构的数据流图;

(2) 数据流图的最低层次是定义精确的数据处理基本活动;

(3) 采用数据字典定义数据流图中的数据项.

对于结构化分析和设计方法来说,满足上述 3 个条件显然是不成问题的.

基于上述 3 个条件,采用如下策略.

1.3.1　产生数据项的测试数据集

用数据规范描述数据字典中定义的数据项的取值范围和类型,然后由数据项数据生成器自动生成每个数据项的测试数据集.

数据项的测试数据集是不同的测试用例(test case) 中所需的该数据项测试数据的总集,产生数据项测试数据集有两个目的:

(1) 不同测试用例所需的某数据项的输入测试数据可以从该数据项的测试数据集中取得,简化了输入测试数据的选择和产生过程.

(2) 不同测试用例所需的某数据项的测试预测结果也可以从该数据项的测试数据集中取得,这样既方便了测试预测结果数据的产生,也提高了测试结果数据的比较分析的效率.

1.3.2 确定软件系统3个层次的测试对象

第1层次的测试对象是数据流图中的基本活动,第2层次的测试对象是独立组合活动,第3层次的测试对象是系统各种应用情况.独立组合活动是指输入和输出数据间有清晰、易于掌握的独立的语义关系的多个相关基本活动的组合.系统应用情况是指系统在某种应用时涉及到的全部活动的组合.总之,所有测试对象都必须是可以独立测试的.

1.3.3 制订每个测试对象的测试用例

对于第1层次的测试对象至少需制订3种测试用例:正常用例、异常用例和混合用例.正常用例选用合法数据进行测试;异常用例选用非法数据进行测试;混合用例选用合法、非法数据相结合的方法进行测试.第2和第3层次的测试对象具有各种复杂的功能类,每个功能类又有许多基本功能.因而,制订第2和第3层次测试对象的测试用例时,需要对其功能进行细致的分析,划分出功能类和每个功能类的基本功能,对每个基本功能用典型数据和边值数据进行测试.

1.3.4 设计每个测试用例的测试数据规范

测试数据规范包括输入测试数据规范和输出测试数据规范.设计测试数据规范前,应该先分析软件系统功能规格说明,然后描述该测试用例的前置断言和后置断言,最后再设计测试数据规范.

1.3.5 根据测试数据规范自动生成测试数据

根据每个测试用例设计的输入测试数据规范自动生成输入测试数据,根据每个测试用例设计的预测结果数据规范自动生成测试的预测结果数据.

1.3.6 自动分析测试结果数据和自动生成测试报告

把执行动态测试的结果和预测结果进行比较分析,比较分析的结果以测试报告形式提供给测试者.

1.3.7 自动报告测试覆盖率(test coverage metric)

测试覆盖率是衡量测试完成程度的重要标志,是测试工具中十分必要的一部分.现有测试覆盖率计算标准有很多[3],我们通过统计测试用例总数和已经测试成功的测试用例个数来确定测试复盖率.测试覆盖率的报告以测试对象为单位,每个测试对象都有一张测试覆盖率报表,当一个测试对象的某一测试用例测试完毕,就自动更新该测试对象的测试覆盖率.

1.3.8 自动产生测试文档并提供维护文档的手段

整个测试过程自动产生测试文档,测试文档包括:数据项的数据规范,测试对象及用例,测试覆盖率,测试数据规范,测试报告.

测试文档的维护包括:文档的修改,文档的删除.

2 基于结构化功能规格说明的测试工具环境的设计与实现

2.1 测试工具环境的组成和系统结构

根据上述基于结构化功能规格说明的测试策略,我们开发了以下测试工具:

(1) 数据项数据生成器:生成数据项的测试数据集.

(2) 对象及用例编制工具:选择测试对象及制定每个测试对象的测试用例.

(3) 数据规范定义工具:定义每个测试用例的数据规范.

(4) 测试数据生成器:根据数据规范生成输入测试数据和预测结果数据.

(5) 分析和报告工具:分析测试结果并生成测试报告.

(6) 覆盖率报告工具:计算测试覆盖率并报告给测试者.

(7) 文档管理工具:产生和维护测试文档.

为了协调各个工具的运行,提高各个工具的使用效率,我们还开发出测试工具协调器,它负责管理各个工具的使用,使上述工具形成一个功能完备、使用方便的测试工具环境.测试工具环境的系统结构如图 1 表示.

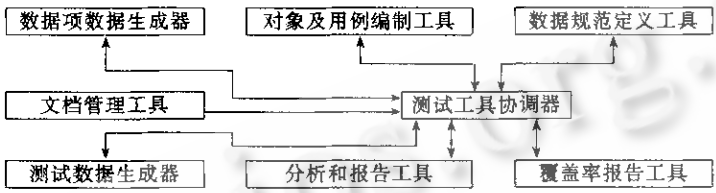

图1

2.2　测试工具协调器的设计和实现

2.2.1　设计策略

(1) 测试工具协调器根据测试活动执行过程的要求管理测试工具的使用.软件测试的执行过程是有顺序的,从选择测试对象、制订测试用例开始,然后设计测试数据规范、产生输入测试数据和预测结果数据,最后执行程序、得到测试报告[4].一个测试工具只是某个测试活动执行步骤中的辅助工具,所以测试工具协调器需按照测试过程的顺序要求来管理测试工具,以便测试工具能在相应的测试活动执行步骤中得到使用.

(2) 测试工具协调器控制测试活动的状态,保证测试活动的前后一致性.包括:

〈1〉 测试活动执行顺序的控制.比如,没有测试数据时就不能执行程序.

〈2〉 不同测试步骤的文档的一致性控制.比如,当一个测试对象的某一测试用例的测试数据规范改变了,该测试用例的测试报告就变成过时.

(3) 测试工具协调器管理状态文件

每个测试对象都建立一个状态文件,状态文件记录测试对象已经达到的测试步骤,测试工具协调器负责管理状态文件.

2.2.2　实现算法

测试工具协调器的算法如下:

(1) 接收用户输入的操作命令(即用户要求执行哪一个测试步骤).

(2) 判断是否存在测试对象,若不存在测试对象,提示用户选择测试对象,转(1).否则,转(3).

(3) 根据状态文件判断输入的操作命令是否合法,若输入的操作命令当前状态下不允许,报错并且转(1).否则,转(4).

(4) 根据操作命令调用相应的测试工具.

(5) 等到测试工具执行完毕,修改状态文件中的相应信息.

2.3　测试工具的设计与实现

在设计和实现各个测试工具时,采用了统一的信息描述形式和界面处理,不同测试工具

的功能通过各自的核心处理实现.

2.3.1 统一的信息描述形式

不同测试工具要输入和输出的测试信息基于一个描述形式,即结构化描述语言(Structured Description Language,简称SDL).SDL提供了4种基本结构类型,各种复杂的信息结构均可通过这4种基本结构构造出来.这4种基本结构是:

(1) 组合结构:一个结构可以由若干子结构组成.表示为S:{S1,S2,...,Sn}

(2) 选择结构:一个结构可以是列举的几个结构中间的一个.表示为S:#{S1,S2,...,Sn}或者表示为S:|S1|S2|...|Sn

(3) 重复结构:一个结构可以是基于一类结构的有限重复.表示为S:*{Si}

(4) 预定义结构:已经预先定义、不需作进一步阐明的结构.表示为S:Sa

预定义结构包括整数(INTEGER)、浮点数(FLOAT)、字符串(STRIING)和图式正文(PATTERN).

2.3.2 统一的界面处理

测试工具的界面用于输入和输出测试信息,通过使用通用结构编辑器USE[5]来实现不同测试工具的统一界面处理.USE采用单一的编辑程序,在不同的结构规则库下可支持不同对象的结构编辑,结构规则库内存放由SDL描述的抽象和具体结构规则,抽象结构规则指USE要编辑的信息的结构规则,具体结构规则指USE要编辑的信息的外观显示规则.

使用USE实现不同测试工具的统一界面处理的具体方法为:

(1) 用SDL描述各个测试工具要输入和输出的测试信息的结构,作为USE结构规则库中的抽象结构规则.

(2) 用SDL描述各个测试工具要输入和输出的测试信息的外观显示规则,作为USE结构规则库中的具体结构规则.

2.3.3 不同的核心处理

不同测试工具的功能主要通过不同的核心处理实现,各个测试工具的核心处理有公用部分,也有专用部分,在设计和实现不同测试工具核心处理时采用先建立公用模块库,在公用模块库基础上建立各个核心处理的专用模块,具体实现细节这里不细述.

3 结束语

上述测试工具环境的工作平台是SunSparc系列工作站及兼容机、SunOs 4.1.1及以上版本和OpenWindows 2.0.测试工具环境已经集成到上海亚士帝信息工程公司的CASE产品ISEE(Integrated Software Engineering Environment)中,用于测试用ISEE开发的应用系统是否符合应用开发的要求,测试工具的使用效果很好.

参考文献

1 Elmendorf W R. Functional testing of software using cause—effect graphs. ASSC'75 RECORD, IEEE Serv. Ctr., New York, 1975.

2 Ostrand T J, Balcer M J. The category—partition method for specifying and generating functional tests. CACM, 1988,31(6):676—686.

3 Sneed H M. Data coverage measurement in program testing. Software Engineering Service, 1986, **21(3)**: 34—40.
4 Myers G J. The art of software testing. New York: John Viely & Sons Inc., 1979.
5 Wang L P, Ju D H, Gu Y Q. USE——a universal structured editor. IBID, 1986, **10(8)**: 159—166.

TEST METHOD AND TOOLS BASED ON STRUCTURED FUNCTIONAL SPECIFICATION

Lin Zhen　Wu Dingyi

(*Department of Computer Science, East China University of Science and Technology, Shanghai* 200237)

Abstract　Software testing will continue to be an important method for ensuring correctness of large scale software systems. Computer—aided software testing can greatly enhance testing power and reduce testing cost. In this paper, first presents a new powerful software test method based on structured functional specification, then depicts how to design and implement a test tool environment based on the test method.

Key words　Software testing, software specification, computer—aided software testing, test tool environment.