ISSN1000-9825 Journal of Software 软件学报 2000,11(6):841~847

软件集成测试的群体协同工作模式及其特性*

刘超 张茂林 晏海华 杨峰 何智涛

(北京航空航天大学软件工程研究所 北京 100083)

E-mail: liuchao@buaa.edu.cn

摘要 该文论述了在大型软件的集成测试中开发组和测试组之间的协同工作方式及其对软件产品质量和开发效率的影响.文章着重讨论了集成测试的基本测试过程、必需的测试任务、测试人员的职责、其他各类相关人员的角色及其相互之间的各种关系,并通过采用标准建模语言UML(unified modeling language)给出了这种协同工作方式的可视化模型.文章还重点研究了软件问题报告的生命周期概念及其在软件质量保证过程中的作用.最后,通过实际案例的分析,说明了这种协同工作模式的一些特点.

关键词 软件工程,软件测试,软件集成测试,软件质量保证,群体协同工作.

中图法分类号 TP311

软件集成测试通常被定义为软件开发过程中的一个必需的测试阶段.此时,程序的整体结构框架和主要构件已经开发完成,并且正在或者已经组装成一个较为完整的可运行程序.所以,与单元测试相比,它无需为每个被测模块编制复杂的支撑程序,便可直接运行被测程序来测试各项功能,特别是测试这些功能的组合应用等复杂情况.但是,软件集成测试也面临着一些复杂的问题,这不仅涉及到具体的测试技术和策略,同时还带有复杂的过程性和群体协同性特点.

(1) 过程性.由于各个模块开发周期和质量的显著差异,致使软件集成实际上是一个很长的渐进过程.加之被测软件丰富的功能、多重的测试目标和多样的运行环境所造成的测试组合爆炸,集成测试必然是一个与开发中的某些工作并行的复杂过程.所谓软件集成测试必须在所有模块编制完成并且测试无错之后才能开始的简单做法常常是低效和不切实际的.

(2) 群体协同性.由于集成测试过程的并行化,开发群体内部和相关部门之间的协同工作方式便直接影响到集成测试的质量和效率.

针对软件集成测试的这些特点,本文第1节分析了软件集成测试的一种典型的测试过程模型,着重介绍基本的测试活动和工作流程以及这些测试活动与其他部门开发活动之间的协同关系.第2节介绍了软件测试文档的基本特性,重点介绍了软件问题报告的生命周期概念、反映其动态特性的状态转移模型及其与软件产品质量演进过程的关系.本文采用了标准建模语言UML(unified modeling language)[1],以可视化图形方式直观地描述这些概念及其相互之间的各种关系.

本文的研究工作是基于对最近4年里我们在多项软件产品测试项目中实际采用的测试方法和技术的分析总结.这些项目主要源于北京航空航天大学软件工程研究所与美国Lotus公司在软件质量工程和产品测试方面的合作以及我们在软件测试技术与工具领域的长期研究与实践.第3节通过对一个典型测试过程的分析,说明这种协同工作模式的一些基本特性.

---

* 本文研究得到国家"九五"科技攻关项目基金(No. 96-780-01-01)资助.作者刘超,1958年生,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.张茂林,1959年生,高级工程师,主要研究领域为软件测试.晏海华,1964年生,副教授,主要研究领域为软件测试.杨峰,1967年生,工程师,主要研究领域为软件测试.何智涛,1972年生,助理工程师,主要研究领域为软件测试.

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1999-02-07收到原稿,1999-06-09收到修改稿

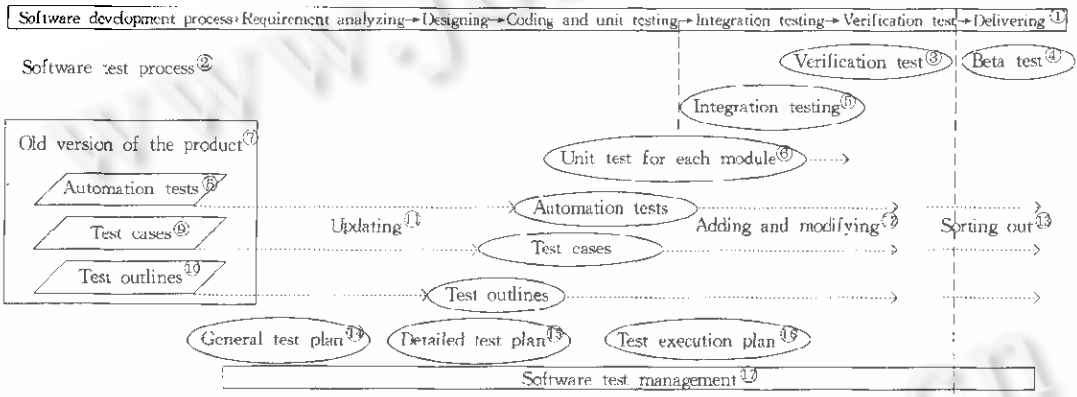
围绕着软件集成测试技术,我们近期的部分研究工作包括关于软件测试过程模型的研究[2]、交互式程序的执行流程的可视化描述及其测试覆盖准则[3]、面向对象程序逆向建模及其可视化类图的自动生成技术[4]以及面向对象程序的测试策略和工具[5~7]等.

Craig Kapland 等人[8]系统地研究并总结了 IBM 采用的软件质量保证方法和实践经验. William Perry[9]详细论述了"生命周期测试法"(life cycle testing approach).

1 软件集成测试过程

1.1 软件集成测试过程模型

图 1 描述了一个软件产品的测试过程. 它是文献[2]所提出的软件测试过程基本模型 POCERM 的一种细化形式,其中的测试实施阶段被进一步分解为单元测试、集成测试和确认测试这 3 个阶段. 集成测试过程横贯于整个开发过程,其各种测试活动之间存在并行性. 实际上,根据产品功能模块的划分,一个产品部门除了要设立若干个开发组之外,还应当设立相应的测试组,或称作质量保证组. 在开发的全过程中,测试组同时要对测试进行计划和设计,开发测试工具和建立测试环境,并在测试实施阶段进行产品测试.



①软件开发过程:需求分析→设计→编码与单元测试→集成测试→确认测试→提交,②软件测试过程,③确认测试,④Beta测试,⑤集成测试,⑥各模块的单元测试,⑦产品老版本,⑧自动测试,⑨测试用例,⑩测试大纲,⑪更新,⑫增补,⑬整理,⑭概要测试计划,⑮详细测试计划,⑯测试实施计划,⑰软件测试的管理.

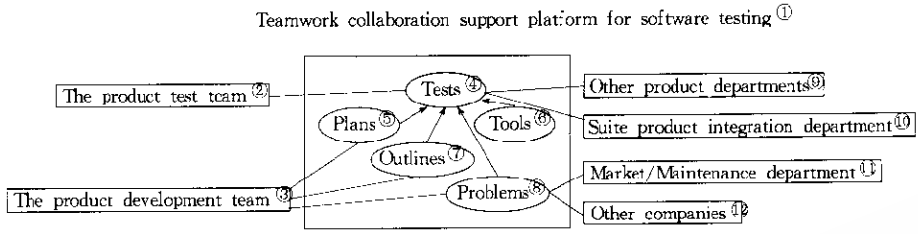
Fig. 1 Software test process
图1 软件测试过程模型

在图 1 中,虚线箭头表示这些测试大纲、测试用例和自动测试程序等的设计过程是一个逐步细化和完善的过程. 从产品老版本指向活动的虚线箭头表示这些工作是在原有成果的基础上进行的. 事实上,除非是开发一个全新的产品,否则,老版本的测试文档和工具等都是可以重用的. 一般而言,应尽早进行系统整体框架和核心功能模块的集成. 它既可以为单元测试提供更好的支撑程序,也有助于及早发现系统设计、复杂应用等多方面的问题.

1.2 软件集成测试中的群体协同工作静态模型

在大型软件开发中必然涉及到许多人员和部门. 他们在各种开发活动中各自扮演着一个或多个角色,每个角色承担着一定的职责. 此外,各个角色的工作内容之间存在着一定的关联性,因此,各种角色之间还必须建立及时而有效的协同工作模式. 不言而喻,任务不明、责任不清,重开发、轻测试,各干各的、缺乏沟通和协同的开发方式是导致质量低劣、开发周期拖延的重要原因.

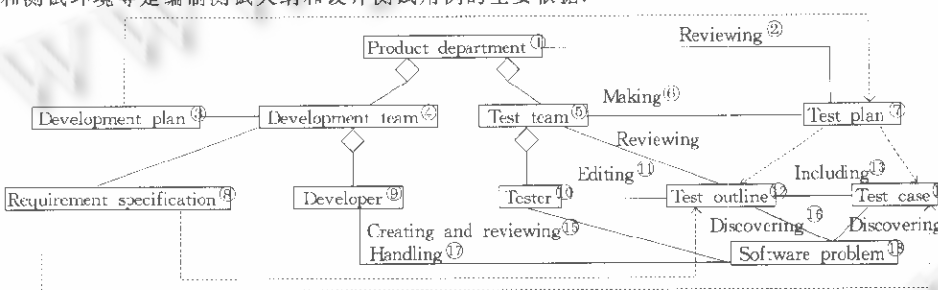
图 2 使用 UML 用例图描述了与测试相关的各项工作(用例)和与之相关联的部门(角色). 产品测试组负责软件的测试,包括一系列相关工作,如制定测试计划、编制测试大纲、开发测试工具、填写和追踪软件问题报告等. 此外,由于被测程序中可能使用了其他构件或产品,比如图形类库、打印机驱动程序等,或者被其他部门和公司使用,所以测试组与其他部门或公司之间需要建立一定的联系. 软件问题报告可能来自于不同部门或公司,并被转发给相应的部门或公司.



①软件测试协同工作支持平台,②本产品测试组,③本产品开发组,④测试,⑤计划,⑥工具,⑦大纲,⑧问题,⑨其他产品部,⑩套装产品集成部,⑪市场/维护部,⑫其他公司.

Fig. 2 Use case diagram of the teamwork collaboration support platform for software testing
图2 软件测试协同工作支持平台的用例图

图3使用UML类图进一步描述了与软件集成测试有关的各类对象及其相互关系.一个产品部可以包含若干个开发组和测试组,它们分别由若干名开发人员或测试人员组成.测试人员负责编制测试大纲、设计测试用例、填写和验证软件问题报告;测试组负责制定测试计划、评审测试大纲;测试计划则需通过产品部的评审.每个软件问题报告应当与发现该问题的测试大纲和测试用例之间建立对应关系.图中的虚线表示某个类对另一个类有某种约束,比如测试计划在测试进度安排方面应与开发计划协调.需求规格说明书以及测试计划中规定的测试准则和测试环境等是编制测试大纲和设计测试用例的主要依据.



①产品部,②评审,③开发计划,④开发组,⑤测试组,⑥制定,⑦测试计划,⑧需求说明,⑨开发人员,⑩测试人员,⑪编制,⑫测试大纲,⑬包含,⑭测试用例,⑮创建及检查,⑯发现,⑰处理,⑱软件问题.

Fig. 3 Object classes and their relationships for software test (class diagram)
图3 测试对象类及其相互关系(类图)

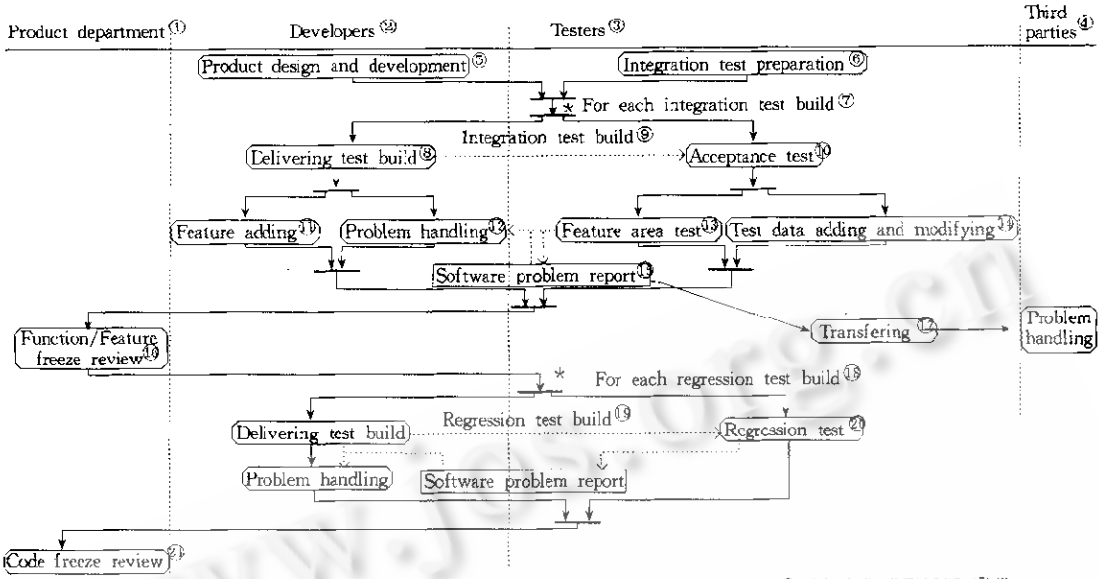
1.3 软件集成测试中的协同工作流程动态模型

图4使用活动图描述了测试人员的基本工作内容和工作流程,以及这些工作与产品开发人员的相关工作之间的并行性和约束关系.图中的活动用带圆角的矩形框表示,每个活动表示与测试相关的一项工作.泳道线区分了不同角色的责任."*"表示集成测试的周期性,即可能需要进行多次测试.在集成测试阶段,开发组依据开发计划,周期性地发送新的测试版.每个新提交的测试版应首先通过接受测试,然后进行全面的功能域测试,在功能冻结后再进行回归测试.前者通过一组基本的测试检验新提交的测试版是否可以接受,即是否存在严重阻碍测试正常进行的问题.功能域测试则依据测试计划和测试大纲,最大限度地测试软件的所有功能特性.回归测试是在所有功能不再作任何增删或修改,而且已发现的比较严重的错误,特别是"全局性"错误均已改正之后,对产品质量进行的"复查".回归测试也可能要进行若干次,因为任何小修改都可能引入新的错误.待所有问题均得到妥善处理后,冻结代码并生成最终产品.

由测试组填写的软件问题报告是一类对象,因此用矩形框表示.这些问题报告将由开发人员进行处理,或经确认后被转发给其他部门或公司.测试组的测试和开发组对问题的处理之间是并行地进行的,无需相互等待.

2 软件测试文档

软件测试文档与其他开发文档一样,有内容、格式、版本、权限、编制、评审工作流程以及与其他文档的关联性等.此外,软件测试文档还有以下一些特性:



①产品部,②开发人员,③测试人员,④第三方,⑤产品设计开发,⑥集成测试准备,⑦对每个集成测试版,⑧发送测试版,⑨集成测试版,⑩接受测试,⑪功能添加,⑫问题处理,⑬功能域测试,⑭测试资料增补,⑮软件问题报告,⑯功能冻结评审,⑰转发,⑱对每个回归测试版,⑲回归测试版,⑳回归测试,㉑代码冻结评审.

Fig. 4　Collaboration workflow of software integration test (activity diagram)
图4　软件集成测试协同工作流程(活动图)

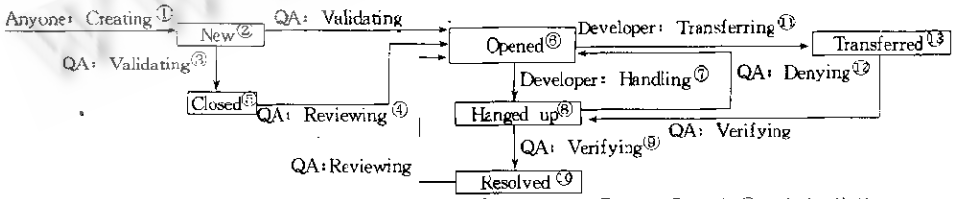
(1) 测试大纲和测试用例在测试实施的过程中,必然要根据实际测试情况进一步加以充实.

(2) 测试大纲和测试用例与软件问题报告之间的关联性是随着软件问题报告的增加而随时创建的.

(3) 软件问题报告不仅是对问题的发生条件、再现步骤和问题性质的说明,更重要的是它记载着问题的处理进程.如果说硬件产品的质量检测是在"筛选"合格产品的话,软件测试和改错工作则是改进提高其质量所必需的过程.因此,软件问题生存周期在客观上反映了测试的进程和被测软件的质量演进过程.

本节将着重描述软件问题报告的生命周期概念,以及在群体协同工作方式下各类人员对每个软件问题报告所负的职责和被授权的操作.有关面向软件集成测试的软件测试大纲和测试用例的设计方法以及自动测试技术等课题将分别在其他论文中作专题研讨.

任何被授权的人,不论是否测试人员,都可以创建新的软件问题报告,如图5所示.这个新的报告必须经过质量保证人员的验证.如果验证问题存在,则将其置为"打开"状态,并将其转给该功能的开发者等待处理;否则被关闭.开发者将检查报告中指出的问题,如果确认是其他功能模块或子系统的问题,则转发给相应的部门;否则,当问题处理完毕后,在报告中填写好处理意见,然后将其置为"挂起"状态,转回给质量保证人员进行确认.经过确认的报告被置为"已解决"状态.



①任何人:创建,②新建,③质保:验证,④质保:复查,⑤关闭,⑥打开,⑦开发者:处理,⑧挂起,⑨质保:确认,⑩解决,⑪开发者:转发,⑫质保:否定,⑬转发.

Fig. 5　Life cycle of the software problem report
图5　软件问题报告生命周期

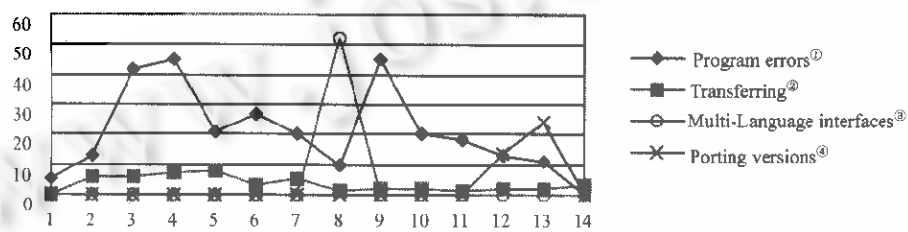
应当指出的是,一些已经解决的问题经常因为程序的某些改动而再次出现.如果我们有完备的问题记录,则可以确保在测试中对这些问题进行复查,从而避免它们被带入最终产品中.因此,运用这种软件问题报告的生命

周期技术可以有效地避免最终产品中仍然存在曾经发现过的错误.也就是说,它从方法上保证了只要开发者对程序的任何改动不再引入新的错误,则软件中残留的问题数将是递减的,从而保证了随着测试进程的推进,软件质量将呈递增趋势.

3 软件集成测试的群体协同工作特性分析及应用实例

本节仅以一个软件升级版本的集成测试为例,通过对已发现的软件问题的性质、修改过程以及与其他相关产品开发过程关联性等方面的分析,说明软件集成测试的群体协同工作的一些基本特性.

该产品由多个独立子工具组成,基本规模为104M字节,共有上千个功能域.在升级版本中,新加功能和被修改的功能域约占总数的10~15%,测试组为此增补和修改的测试大纲约占总数的15%.在集成测试初期,主要增改部分均已完成集成,所以其多语种版本的移植工作实际上在集成测试期间便已经同步启动.这样做显然有利于及时发现和改正程序中影响移植的问题和保证同步发布产品的多语种版本.图6给出了在对应各测试版的测试周期中发现的产品错误数目的分布情况.



①程序错,②转移,③多语言接口,④移植版.

Fig. 6 Classification and time based distribution chart

图6 软件问题分类及时间分布图

特性1. 软件问题分类及时间分布曲线的波峰数与软件测试期间产品质量的稳定性成反比.波峰数主要与下列4个因素成正比:

(1) 新加功能和被修改功能开始集成的时间离散度.例如,本例中以测试周期9为代表的波峰是由于此测试版中集成进来第2批新功能和有较大幅度修改的功能.

(2) 被测软件的测试过程与其移植版本或应用产品的开发过程的并行度.例如,本例中在测试周期8、12和13上分别迭加了与此产品的移植版有关的软件问题数.

(3) 各功能域测试时间的离散度.这与测试策略和测试计划有关.通常,应当预先确定问题较多的功能域,并采取优先和强化测试这些功能域的测试策略.

(4) 程序修改的波及效应.由程序修改不当引入大量错误的测试版应当被测试组的接受测试发现并拒绝,从而防止由此引起的产品质量波动.

特性2. 软件问题改正的延迟时间与开发人员工作量成正比,工作量与软件问题数量、软件问题报告提交的密度、问题修改所需的时间(天数或小时数)以及开发人员在此期间承担的其他任务量成正比.因此,任何有效的测试策略和测试计划必须能够有效地防止在测试后期才发现会引起开发人员极大工作量的软件问题.

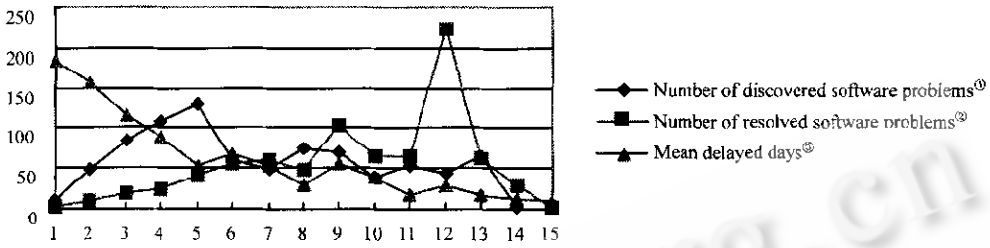
$$\text{平均延迟时间}=\frac{\sum^{\text{所有问题}}(\text{确认已解决日期}-\text{发现日期})}{\text{软件问题报告数}}.$$

例如,在本例中,解决软件问题的平均延迟时间约36天左右,如图7所示.

必须说明的是,解决软件问题的延迟时间不同于开发人员修改这个问题所需的时间,因为开发人员在此期间可能有其他工作.这个延迟时间反映了软件测试和程序改错工作之间的相对独立性和集成测试的实际进度,是计划回归测试开始时间的重要参数.

特性3. 回归测试的总时间长度与软件问题曲线上的波峰数和平均周期长度成正比.例如,本例中最后4个测试周期为回归测试周期,其中时间长度大体与软件问题报告时间分布曲线的平均波动周期长度相当.在回归

测试期间,每个周期发现的需要改正的"程序错"数目在10个左右,呈递减趋势,而且严重级别较低,改正的延迟时间很短.



①软件问题发现数,②软件问题解决数,③平均延迟天数.

Fig. 7 Software problem discovered-resolved chart

图7 软件问题发现-解决曲线

特性4. 一个软件问题的检验、分析、修改和确认工作所涉及的人数和这些人之间进行意见交流的次数反映了群体协同工作的强度,同时也反映了软件问题的复杂度.

$$\text{软件问题报告平均介入人数}=\frac{\text{每个软件问题报告涉及的人数之和}}{\text{软件问题报告数}},$$

$$\text{软件问题报告平均意见交换数}=\frac{\text{每个软件问题报告中意见交换记录数之和}}{\text{软件问题报告数}}.$$

在本项目抽检的100个已改正的问题中,有81%经历过4~6人的检查处理,有80%的问题的意见交换次数在5~9次之间,最多达20次之多.这些人的意见均记录在软件问题报告库中.

特性5. 测试人员之间、测试人员与开发人员和其他相关部门之间的交互量是群体协同工作的协同性的基本度量.通常,可以用相互之间的电子邮件、通话和传真等的数量之和来计算,并可以由此推算协同工作的代价和工作量.例如,在本例中,人员之间主要通过电子邮件交互,人均每日收发电子邮件约5~10封.

参考文献

1 Flowler M. UML Distilled-Applying the Standard Object Modeling Language. Reading, MA: Addison-Wesley Publishing Company, 1997

2 Liu Chao, Jin Mao-zhong. Software test process model-POCERM. Journal of Beijing University of Aeronautics and Astronautics, 1997,23(1):56~60
(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)

3 Liu Chao. Program interactive execution flow chart and its coverage test criteria. Journal of Software, 1998,9(6):458~463
(刘超.程序交互流程图及其测试覆盖准则.软件学报,1998,9(6):458~463)

4 Liu Chao, Li Jian, Shen Hai-hua. Reversed automatic generation of visualized class diagram of object-oriented program. Journal of Beijing University of Aeronautics and Astronautics, 1998,24(4):411~414
(刘超,李建,沈海华.面向对象程序可视化类图的逆向自动生成,北京航空航天大学学报,1998,24(4):411~414)

5 Wu Peng-cheng, Jin Mao-zhong. Testing model and strategies of object-oriented software. In: Yang Fu-qing, He Xin-gui eds. Progress on Software Engineering——Technology, Methodology and Practice. Beijing: Tsinghua University Press, 1996. 342~344
(吴鹏程,金茂忠.面向对象软件测试模型与测试策略.见:杨芙清,何新贵编.软件工程进展——技术、方法和实践.北京:清华大学出版社,1996.342~344)

6 Li Jian, Jin Mao-zhong. Object state testing. Journal of Beijing University of Aeronautics and Astronautics, 1997,23(1):98~104
(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)

7 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
(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)
8 Kaplan C, Clark R, Tang V. Secrets of Software Quality-40 Innovations from IBM. New York, NY: McGraw-Hill, Inc., 1995
9 Perry W. Effective Methods for Software Testing. New York, NY: John Wiley & Sons, Inc., 1995

Teamwork Collaboration Model of Software Integration Testing and Its Characteristics

LIU Chao ZHANG Mao-lin YAN Hai-hua YANG Feng HE Zhi-tao

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

Abstract In this paper, the authors study the teamwork collaboration model between the development teams and testing teams for the software integration tests, which significantly affects the product quality and the development efficiency. The authors also discuss the basic process of the integration test, the required test tasks, responsibilities of testers, roles of any other relevant people, and the relationship among them. With the Unified Modeling Language (UML), the visualized models are given. Furthermore, the concept of life cycle of the software question report is introduced and its important role in the software quality assurance process is shown. Finally, the test results of software products are analyzed to show several features of the model.

Key words Software engineering, software testing, software integration testing, software quality assurance, teamwork collaboration.