第16卷 第1期
2006年1月

计算机技术与发展
COMPUTER TECHNOLOGY AND DEVELOPMENT

Vol.16 No.1
Jan. 2006

软构件的可测试性研究

白 雪,宋 雨,韩秀娟,剧树春
(华北电力大学 计算机科学与技术学院,河北 保定 071003)

摘 要:构件的可测试性是决定构件质量的关键因素,若能在构件设计阶段就考虑构件的可测试性问题,改善和提高构件的可测试性,那么构件质量就能得到很好的保障,进而减少系统开发时的测试成本。文中针对这个问题,讨论了影响构件可测试性的几个因素,分析了构件测试中存在的问题和构件测试要达到的目标,提出一种构造可测试性构件的通用体系结构,即在原有构件的基础上增加测试工具,把可测试性构件当作对包含嵌入式测试和跟踪工具的扩展单元。

关键词:构件测试;可测试性;基于构件的软件开发;构件质量
中图分类号:TP311.56 **文献标识码**:A **文章编号**:1005-3751(2006)01-0106-02

Research of Software Component Testability

BAI Xue, SONG Yu, HAN Xiu-juan, JU Shu-chun
(School of Computer Science & Technology, North China Electric Power University, Baoding 071003, China)

Abstract: The testability of software components is one of the important factors determining the quality of components. If people can consider the testability of the component while design it, the quality of component could be ensured and the cost in software test could be reduced at the same time. Survey some factors determining the testability of component, propose a model to build highly testable component by embedding testing and monitoring mechanisms.

Key words: component testing; testability; component-based software developing; component quality

0 引 言

基于构件的软件开发逐渐成为软件开发的主流范型。设计一个基于构件的系统,必须了解构件的功能,因此在整个开发过程中,构件会被测试很多次(构件单元测试、综合测试和系统测试)[1]以保证它满足开发要求。而测试构件本身又存在着许多的困难,因此若能在构件设计阶段就考虑构件的可测试性问题,改善和提高构件的可测试性,那么构件质量就能得到很好的保障,进而减少系统开发时的测试成本。针对这个问题,讨论了影响构件测试性的几个因素,构件测试中存在的问题和测试的目标,提出了一种改进构件可测试性的策略,并对其进行了验证。

1 构件测试

1.1 影响构件可测试性的因素

影响构件可测试性的因素包括:可观察性、可追踪性、可控制性和易理解性[2]。

(1)可观察性(observability)。

根据 Roy S. Freedman 所描述的,软件可观察性是指观测软件操作、输入变量和输出结果的难易程度。

(2)可追踪性(traceability)。

它是指跟踪构件内部的属性状态和构件行为。包含了两层意思:行为可追踪性;追踪的可控制。

(3)可控制性(controllability)。

可控制性是构件控制输入/出操作和行为的能力。

(4)易理解性(understandability)。

构件的易理解性取决于构件提供信息的多少和这些信息组织的好坏。首先是介绍构件功能的文档;其次是构件源代码的介绍,包括构件源代码和一些支持性代码(如安装代码、测试驱动等);最后是构件质量的介绍。

1.2 构件测试的目标

测试构件主要是为了解决以下两个问题:

①测试构件是否达到设计说明书的设计要求,是否实现了它的功能需求。

②在设计构件前检查结构和交互需求是否正确和完整,这与具体的软件应用环境有关。

1.3 构件测试中问题存在的原因

在基于构件的软件测试中存在着一些技术上的问题有待解决。出现这些问题的主要原因一是由于存在异质,二是信息缺乏[3,4]。

导致异质的原因有很多:对于同样的构件,即使规格相同也会有不同的执行方法;不同构件由不同的语言编写运行在不同的平台。

信息缺乏是由于构件提供者和构件使用者之间的交流代沟而产生的。构件提供者事先不知道构件将会用于

收稿日期:2005-04-05
作者简介:白 雪(1981—),女,贵州都匀人,硕士研究生,研究方向为软件工程;宋 雨,教授,研究方向为软件工程。

什么环境;另一方面,构件用户得不到足够的构件测试信息。

2　提高构件可测试性的策略

在基于构件的开发中,提高构件的可测试性的方法有很多,它们主要采用以下两种策略[3,5]:

第一种策略是建立一套新的测试技术并将其植入到构件内部,用户可以用其对任何构件进行测试,这种技术也被称为"plug－in－and－test"。

第二种策略是对构件进行扩展,在原有构件的基础上增加测试工具,如运行计时器、监控器等。这些方法被称为"built－in－tests"(BIT),用户可以通过一个标准的接口获得这些额外的信息。

3　可测试性构件的体系结构

文中重点讨论第二种策略[5],加入测试接口和跟踪接口,把可测试性构件当作对包含嵌入式测试和跟踪工具的扩展单元。对构件内部的测试和跟踪操作通过上面两个接口进行。图 1 给出了可测试构件的体系结构。

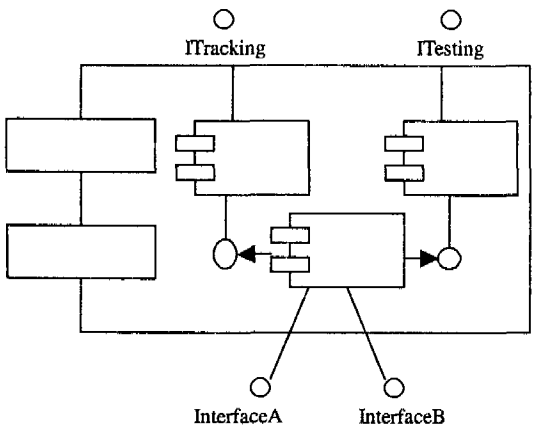

图 1　可测试性构件体系结构图

可测试性构件由 3 个子构件组成:待测构件(CUT)、跟踪构件(Itracking,用于提供跟踪接口)、测试构件(Itesting,提供测试接口)。

3.1　跟踪构件

跟踪构件提供了对原构件进行跟踪的机制。跟踪包括以下几种:操作跟踪、错误跟踪、状态跟踪,GUI 事件追踪和交互追踪在文中暂不涉及。跟踪构件提供了两个接口:ITracking 为可测试性构件的客户接口;ILogTrace 为待测构件接口。ITracking 接口使用到以下几个服务:

(1)operationalTrace(method: String, type: char)。将跟踪操作的代码嵌入到原构件中,其中 type 变量用于登记此接口调用其他构件和被其他构件调用的信息。

(2)StateTrace(attribute:String[ ])。将跟踪变量的状态的代码嵌入到原构件中。

(3)ErrorTrace(exception:String[ ])。将错误跟踪代码嵌入到原构件中。

(4)SetLog(file:File)。设置日志文件。原构件通过 ILogTrace 接口调用 SetLog 服务将跟踪信息写到日志文件中,外界用户无法对此接口进行操作。

3.2　测试构件

测试构件提供了 ITesting 可测试性构件的客户接口和 ILogAssert 原构件接口。通过 ITesting 接口,可将测试工具嵌入到原构件中,它使用到以下几个服务:

insertPrecondition(method: String, continue: boolean),
insertPostcondition(method: String, continue: boolean),
insertInvariant(method: String, continue: boolean)

当声明不符合规范时,会有以下两个响应:警告和继续执行,调用 ITesting.setLog(file: File)服务将警告写入一个日志文件,而原构件部分是否继续执行由 continue 变量控制。调用 getModel()和 getAssertions()方法返回构件行为模型和构件规格文件,这些文件可以用来自动生成测试用例和嵌入式代码。目前这样的工具还在开发中。

4　一种实现方法

可测试性构件的执行包括嵌入代码的生成和测试追踪构件执行两个部分。在嵌入代码的生成阶段,构件提供者需要建立测试库和跟踪库,测试库应包括构件自测代码、原构件规范的执行代码;而跟踪库应包括跟踪代码和监测工具的执行代码,为了保证原代码的独立性,嵌入代码将会放到由测试构件和跟踪构件生成的原构件的字节码中,代码插入的方式将会影响测试和跟踪构件的执行。

文中以堆栈构件 Stack 为例对可测试性构件的体系结构进行说明。堆栈构件包括两个操作:入栈、出栈。堆栈构件仅包含一个类,把合成后的可测试性构件称为 TStack 以区别于原构件 Stack。用 Java 语言作为工具,原构件的使用将通过字节码操作库 BCEL (Byte Code Engineering Library)实现。BCEL 的 API 函数库可完成静态分析和动态生成或者 Java 类文件的转换。API 函数提供修复指令的操作和服务。

跟踪构件的结构与测试构件的结构类似,以测试构件为例,图 2 给出了加入 ITesting 和 ILogAssert 接口后的测试构件体系结构图。嵌入代码可以通过 ITesting 接口调用 Testing 类执行,原构件接口中每一个公用的服务都对应一个 A<method name>类,而且每个服务都提供了两个接口 IA<method name>_insert 和 IA<method name>_check,它们仅在测试构件内部可见。

在预处理中调用 IAStack_pop_check.checkPrecondition()定义一个变量存储栈的长度,可以用它与运行后栈的长度进行比较。

5　结　论

文中对软构件的可测试性进行了讨论,分析了影响构件可测试性的因素,提出了一种可测试构件的体系结构。它主要的优点在于帮助构件使用者更好更有效地对构件

(下转第 110 页)

是 AutoCAD 的数据库类,为访问 AutoCAD 数据库提供直接接口;AcEd 类库是编辑器类,它为 AutoCAD 图形编辑器提供核心函数;AcRx 类库主要用于初始化和链接动态链接库,同时用于实时类注册和识别;AcG 类库提供了用于绘制 AutoCAD 实体的图形接口[5];在 ObjectARX 中,图形的绘制主要与图形数据库的操作有关,网络绘图主要实现异地机器的图形数据库中的图形实体绘制。在取得本地图形数据库中图形实体的数据并发送到异地后,再把接收到的图形数据转换为本地图形数据库的实体数据。这需先取得图形数据库的指针,再对收到的图形数据进行操作。AutoCAD 利用以上的常驻数据库对象(AcDbObject)和表示 ActiveX 控件模型的 COM 对象之间的链接关系,来实现其 AcitiveX 自动控件对象模型。这种链接包括两个单项的指针,其中一个是 COM 对象的 Iunknown 数据成员,它是用一个在 AcDbObject 对象中的临时反应器保存的;另一个是常驻数据库对象的 AcDbObjectId,它在 COM 对象中是作为一个成员变量被保存的[4],如图 3 所示。

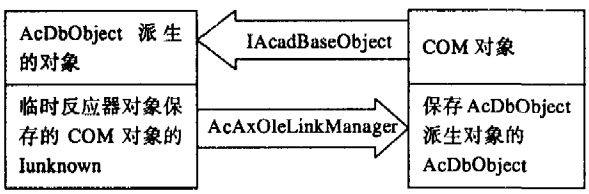

图 3 AutoCAD 和 COM 组件的通讯

DCOM 为 COM 在分布式计算环境下的拓展,DCOM 位于网络协议的最高层,相当于 OSI/ISO 协议的应用层,具有协议无关性,可以使用 TCP/IP,UDP,IPX/SPX,以及 NetBIOS 等底层面向连接或无连接协议[6]。利用 DCOM 实现跨机器通信,屏蔽了底层数据传输的细节,具有位置透明性。DCOM 对象在通过常驻数据库之间的链接关系获取 Iunknown 指针后,通过查询 QueryInterface 方法得到数据库对象 IacadBaseObject 的指针,从而对图形数据进行操作。

4 结 论

随着网络技术的发展,CSCW 已经成为一种先进工作模式。文中在基于传统 AutoCAD 软件的基础上,结合 CSCW 的思想,利用通用的协作软件 NetMeeting 及 ObjectARX,DCOM 技术,提出了一个新的协作平台来支持协同工作的要求。

参考文献:

[1] 丁建军,杨 岳,周咏翔.基于 COM/DCOM 技术的分布式协同设计系统的研究[J].机械设计,2003,20(5):9-11.

[2] 储 备,罗满良,蔡 青.基于 AutoCAD 的工程 CAD 实时协同研究[J].机械科学与技术,2001,20(1):158-160.

[3] 王 文,李治柱.CSCW 技术的研究与实现[J].微型电脑应用,2003,19(10):8-10.

[4] 王 安,蒋寿伟,蒋萍萍.基于 DXF 图形数据实现网络传输的研究[J].计算机辅助工程,2002(1):30-34.

[5] 张振宇,梁补女,刘彦国.用 ObjectARX 开发 AutoCAD 2000 功能[J].兰州工业高等专科学校学报,2003,10(1):20-24.

[6] 鱼 滨,吴丽贤,和 力.DCOM 的分布式体系结构及实现机制分析[J].计算机应用研究,2003,55(7):24-25.

(上接第 107 页)

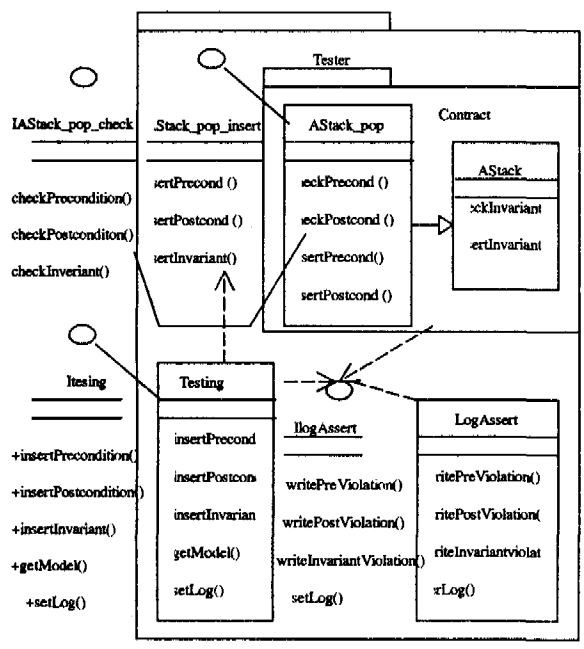

图 2 TStack 测试构件体系结构图

进行测试,而与此同时,构件提供者也无需提供构件原代码。缺点是增加了构件代码长度。目前国内对构件可测试性的研究还处于起步阶段,还有很多问题需要解决,例如缺少测试的统一标准;如何进一步提高测试性能等,这也是以后进一步研究的方向。

参考文献:

[1] Weyuker E. Testing Component - Based Software - a Cautionary Tale[J]. IEEE Software, 1998,15(5):54-59.

[2] Gao J. Component Testability and Component Testing Challenges. Proc CBSE'00, 2000. URL[EB/OL]. www.sei.cmu.edu/cbs/cbse2000/papers/18/18.pd,2000.

[3] Beydeda S,Gruhn V. State of the Art in Testing Components [A]. In: 3rd International Conference on Quality Software [C]. Dallas: IEEE Computer Society, 2003.146-153.

[4] 于 洁,杨海燕,高仲仪,等.软件的可测试性设计[J].计算机工程与应用,2003(3):124-126.

[5] Beydeda S,Gruhn V. Merging components and testing tools - The self - testing COTS components (STECC) strategy[A]. In: Proc Euromicro Conference Component - based Software Engineering Track [C]. Turkey: IEEE Computer Society, 2003.107-114.