基于构件的软件测试技术研究

The research of component – based software testing technique

马良荔 (武汉 华中科技大学计算机科学与技术学院 430074)
(武汉 海军工程大学计算机工程系 430033)
卢炎生 (武汉 华中科技大学计算机科学与技术学院 430074)
刘孟仁 (武汉 海军工程大学计算机工程系 430033)

1 引言

过去的几十年,软件开发模式发生了很大的改变。随着软件开发机构对开发成本、周期要求的提高,产生了软件构件化。

软件构件技术的提出解决了面向对象技术无法使大量结构相似的应用程序结构得到重用的矛盾。因此构件技术对软件重用和集成具有非常重大的意义,已成为当前软件领域主流技术和研究热点。

继面向对象的设计方法之后,基于构件的软件设计方法正在逐渐成为新的趋势。由于构件的特点,构件的测试问题应该从构件的开发者与构件的使用者两个角度来看,其原因是:一是他们使用的环境(包括编程语言、操作系统、硬件平台)可能不同;二是拥有的资源不同,构件开发者拥有构件源代码,构件使用者可能只有构件的可执行代码;三是测试目的不同,构件的开发者需要测试构件的所有功能,构件的使用者只关心与他有关的构件功能。

2 概述

2.1 构件及测试技术概述

构件技术最大的问题是单个构件在整个系统中的故障隔离以及相应整个集成模块的测试策略。软件构件可使软件部分得到重用,软件的每一部分都经过严格定义,彼此独立。关键是如何使之组合在一起。

与软件测试相关的构件特性有:

(1) 构件的可观测性。其中定义了如何观测构件的行为、输入参数和输出参数。构件接口的定义在确定构件的可观测性上起决定作用;

(2) 构件可跟踪性。构件可以跟踪其属性和行为的状态;

(3) 构件可控制性。表明很容易对构件的输入/输出、操作和行为进行控制;

(4) 构件可理解性。表明提供了多少构件信息以及如何表示。

测试软件构件要解决如下问题:

(1) 检查构件与软件规范是否一致,并完成其功能需求;

(2) 检查实现的系统是否反映了规范中所描述的结构和交互需求(假定这些需求是正确而且完整的)。

2.2 构件测试存在的问题

构件测试中存在的问题主要集中在如何对构件进行集成。目前,软件开发小组在开发构件时方法混乱,这样,很难用统一、不变的测试技术处理不同的需求。

理想情况下构件不仅可执行,可相互调配,而且具有可测试性。由于需要进行规范、定义测试结构,并且要有内置测试接口支持其与构件测试工具和平台的交互设计,这样就使得设计理想情况下的构件非常困难。另外,就是开发能测试构件的测试工具和测试平台也非常困难,因为构件中要使用一种以上的实现语言和技术。

构件测试中系统测试和单元测试和以前传统的软件系统不同,在构件生成方通过单元测试的构件用在构件使用方就不一定可用。在系统级,必须要考虑构件之间的交互,所以,在构件使用方,需要开发系统集成测试计划。在系统集成时,仅仅考虑单个构件的可靠性是不够的。系统测试的另外一个问题就是冗余测试和容错测试,冗余测试中构件单元测试过程中的测试充分性准则可使用在同一个构件的系统级测

试中。由于是否能有效进行容错是保证系统顺利运行的重要因素，因此容错代码也要进行测试。

3 构件测试技术

3.1 构件验证

要保证构件的可靠性就必须要使用相应的证明方法，即证明构件能满足开发人员的需求，并且能对给定系统产生高质量的影响。构件证明方法要遵循以下3至4个质量评估技术来确定构件是否适于某个系统：

(1) 第1步：黑盒构件测试。这类测试与所选择的测试用例有关，不考虑软件语法。黑盒测试需要可执行的构件，一个输入和一个 oracle 以完成相应的测试过程。Oracle 用以确定失效是否是通过检查每个输入的输出产生的。黑盒测试将预期结果与实际结果进行比较，这就是功能测试或接口测试。该方法使用基于系统操作原型的黑盒测试，也就是通过测试用例将软件投入应用以确定构件的质量能否在系统中运行。

(2) 第2步：系统级故障注入技术。该方法通过在整个系统中引入 bug 或故障来实现测试，该方法并不是真的在系统中找 bug，而是表明当某个构件失效时，系统会变得有多糟。

该方法的使用称为接口传播分析(Interface Propagation Analysis，IPA)，IPA 必须明确要注入哪种构件失效模式以及期望何种系统失效模式。实际上，IPA 通过构件之间的接口破坏所要传播的状态，称为破坏函数(perturbation function)的软件程序在系统执行时以破坏状态替代初始状态，系统失效模式包括系统输出数据、故障系统数据和要破坏的构件之间的数据流。

(3) 第3步：操作系统测试(OST)。该方法就是系统级故障注入技术，因为 OST 检查引入商用构件时系统的容忍性以及系统功能的完备性，运行初始状态，对输出信息不进行任何修改。

(4) 第4步：系统级故障注入技术。要使用软件包装程序，包装程序的概念起源于需要使用构件，而不是其可靠性。通过限制构件的功能运行软件包装程序，包装程序不能修改构件的源代码，而是间接地用匿名调用方式限制其功能。有两种包装程序：一种是检查和限制构件的输入，另一种是在交付系统之前检查输出。

该方法的缺点是：

(1) 因为黑盒测试不在系统级实施，因此要生成针对系统级中构件行为的测试工具，用以验证可靠性；

(2) 黑盒测试有时不能执行大部分代码，这是因为系统测试过程中故障检测机制的问题；

(3) 存在一些严重的问题，比如特洛依木马，很难用黑盒测试检测出来；

(4) 开发精确的 oracle 程序、生成输入以及创建测试驱动程序的成本都非常高；

(5) 可将系统级测试故障注入技术列入最坏情况测试技术，因为该技术集中在通过注入故障考察系统的鲁棒性；

(6) 如果能检测到系统中的错误，操作系统测试就是有效的，因此，表明了系统的容错性；但是会有许多象木马程序这种无法检测。需要大量的系统级测试使得系统能够处理真正的构件失效。这就产生了最有利误差，不能保证故障检测技术；

(7) 包装程序并不是很简单的，非法输入有时会将包装程序弄乱(fool)；设计或实现包装程序中人的错误会降低其价值。

3.2 构造构件元数据

使用构件元数据的方法提供了汇总信息(Summary Information，称为构件元数据)来分析和测试构件的框架。元数据基于不同的信息，有特定的上下文和需要。对提供的每种元数据有唯一的格式和标签。构件生成方将汇总信息嵌入到软件构件中。

该方法的主要步骤有：

(1) 构件生成方为使用分析技术为构件使用方收集元数据，不需要源代码的构件使用方使用元数据；允许构件使用方查询构件中的信息；

(2) 每个操作的输入空间可分成各子域集合，汇总信息与每个子域相关；

(3) 使用开发工具给出操作输入，在查询后使用构件的汇总信息以测试系统中的构件。

元数据描述了构件的静态和动态特性，具有如下特点：

(1) 提高了程序分析的准确性；

(2) 元数据可依据构件使用方所需的特定功能进行改变，信息存储方法很灵活；

(3) 元数据充分考虑了构件定制的要求；

(4) 该方法为用户提供了合适的查询机制,可以系统、方便地实现测试;

(5) 该方法开发标准的元数据符号,要确定标准的元数据附属信息(后面跟着第三方构件生成方)的格式是相当困难的;

(6) 该方法目前仅仅在小程序上进行了测试,需要进一步深入。

3.3 用于构件集成测试的测试模型的 UML

测试模型使用 UML 次序和协作图抽取构件接口之间存在的故障,将基于开发过程的 UML 与测试过程连接在一起。UML 测试模型包括节点(即集成目标)和消息流(表明节点之间的交互)。

UML 开发阶段使用 Rational Objectory Process,分别为四个阶段:开始(inception)、精化(elaboration)、构造(construction)和转换(transition)以建造 UML 模型。构件集成测试由以下步骤完成,见图 1。

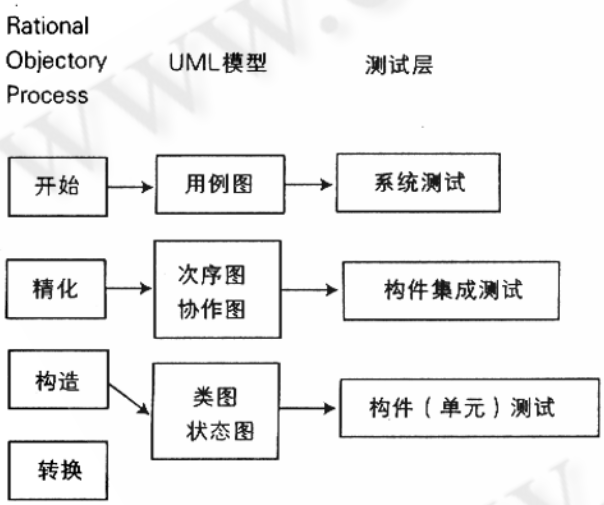

图 1

(1) 基于事件流构造 UML 测试模型。首先,提取正常和异常事件流的次序图,在出现并发事件的情况下产生每个事件流的协作图,然后基于构件中的消息转换模式将协作图和次序图划分成 ASF(Automatic System Function,自动系统功能)单元。

(2) 完整的 UML 测试模型可从 ASF 中提取的节点和消息流;

(3) 通过将测试用例选择准则用于 UML 测试模式来选择测试用例。

该方法的主要特点:

(1) 该方法基于 UML 协作和次序图,用广泛应用的标准记号建立该过程,这就必须学习新的记号和语言以理解所提出的方法;

(2) 可使测试技术自动化,但是不真正实施;

(3) 该方法假设集成系统中的所有构件均通过了到单元测试,而且是用黑盒测试完成的;

(4) 选择测试准则并不基于测试充分性标准,这样会生成更好、更有效的测试用例;

(5) 该技术的准备时间很短。

3.4 使用断言定义语言规范和测试构件

该方法针对软件构件的单元测试。使用特别适用于测试的规范语言 ADL 来形式化地记录软件构件的行为。另一个相关语言 TDD 用来系统地描述测试软件构件的测试数据。

在该方法中,以系统方式用不同的测试输入来运行程序。通过检查程序的结果或是描述构件行为的规范所涉及的功能来确定行为的正确性,进行单元测试需要以下构件:

(1) 要测试的功能;

(2) 执行该功能的测试数据,通过测试数据描述(TDD)文件来规范测试数据;

(3) 确定功能能否正确执行的方法,从 ADL 规范产生的断言检查处理功能的任务;

图 2 说明了执行单元测试所需的步骤,单元测试所需的三个构件在左边的一栏,测试中的功能和 ADL 规范之间的双箭头说明这两个实体之间的联系,这两个实体构成断言检查。

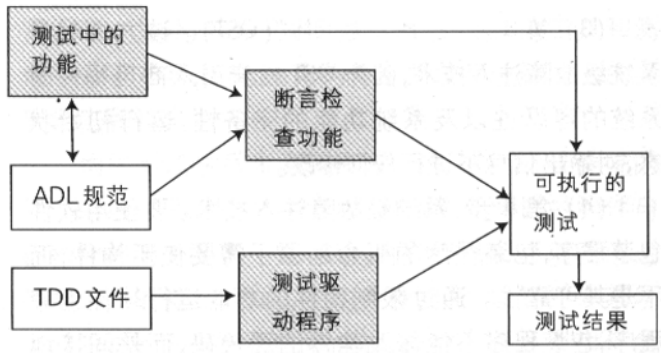

图 2

还有,第三个构件 TDD 文件由测试驱动程序组成,测试驱动程序可以调用带有不同测试数据集的测试功能。三个阴影方框连在一起执行测试,当执行测试文本时,产生描述功能如何在给定测试数据上实施的测试结果。该方法的特点是:

(1) 要以系统方法用许多不同的测试输入运行程序,通过检查程序的结果或描述构件功能行为的规范来确定行为的正确性;

(2) 测试数据选择减少了测试过程的冗余性,并使确认过程自动化。

4 结论

除上述描述的构件测试方法外,还有构件接口测试(CIT)、构件中的内置测试、用于构件测试的并发测试(PACT)、使用回顾机制的构件测试方法、用构件接口图(CIG)来测试软件构件等等方法。

构件测试方法已经从构件开发方的角度还有构件测试方的角度进行了相关研究,用于单个构件可靠性和整个系统可靠性的有效分析技术的应用已经展开。但是在该领域还有一些问题需要解决:首先,要制定用于测试特定领域的构件有效的测试策略,以便今后重用;其次,如果将元数据作为解决构件测试的方案,那么在不同的第三方构件生成方创建元数据标准将会需要大量的合作和协作;第三,可通过改进实现构件的语言来提高构件的可靠性;最后,除了测试用例的自动化以外,还需要对测试用例进行排序并给出优先级;测试场景的范围应更广泛,以便用于将构件适用于更宽范围内的使用模式。

参考文献

1 Clemens Szyperski, Component Software – Beyond Object Oriented Programming, Addison Wesley, 1997.

2 E. J. Weyuker, Testing Component – Based Software: A Cautionary Tale, IEEE Software, Vol. 15, No. 5, September/October 1998.

3 Jefferey M. Vaos, Certifying Off – the Shelf – Components, IEEE Computer, June 1998.

4 Jefferey M. Vaos, A Defensive Approach to Certifying COTS Software, Technical Report, Reliable Software Technologies Corporation, August 1997.

5 Allessandro Orso, Mary Jean Harrold, David Rosenblum, Component Metadata for Software Engineering Tasks, In Proc. 2nd International Workshop on Engineering Distributed Objects, Davis, CA, November 2000.

6 Gary A. Bundell, Gareth Lee, John Morris, Kris Parker, A Software Component Verification Tool, In the Proceedings of International Conference on Software Methods and Tools, 2000. SMT, 2000.