维普资讯 http://www.cqvip.com

基于结构模式的构件系统测试方法*)

许 静 庞 博 王庆人
(南开大学信息学院机器智能所 天津 300071)

摘 要 基于构件的开发模式使大型软件系统的开发提高了效率,增强了可靠性,但同时也给测试带来了诸多困难。本文从构件使用者的角度出发,提出了一种基于结构模式的测试方法,在借鉴 ADL 的基础上详细描述了这种方法的形式化表示,并给出了工具的设计方案。

关键词 构件系统,测试方法,软件体系结构

An Approach to Testing Component System Based on Structure Pattern

XU Jing PANG Bo WANG Qing-Ren
(Institude of Machine Intelligence, Nankai University, Tianjin 300071)

Abstract The increase of size and complexity of software system leads to develop large system with components. More and more developers focus on component-based system developing. There are many problems for testing these systems. This paper presents an approach, which is called structure pattern based testing method for users to test component systems. And a formal notation for describing this method is proposed.

Keywords Component-based system, Test methodology, Software architecture

1 引言

随着软件系统规模和复杂性的不断提高,构造大型系统时,系统内部的数据结构和算法细节已不是关心的主要问题,这一趋势的代表就是采用基于构件的系统开发方法,目前,基于构件的系统开发方法已成为主流的开发模式[1~3]。一次编写、重复使用的商业构件 COTS(Component off-the-shelf)将成为软件工程的"即插即用"的软件开发模式[2,5]。目前,有多个组织和公司制定了构件开发的标准或开发了相关产品,为构件、构件框架和接口建立了模型和技术规范,其中 OMG 的 CORBA、Microsoft COM/DCOM(或 VS. NET)以及 Sun JavaBean/EJB 占主导地位。

基于构件的开发模式在为建造大型软件系统提供了诸多方便的同时,也给测试这些系统带来了困难。由于构件系统本身的特殊性,传统的测试方法已不再适用于对该类系统的测试,需要新的理论方法和技术的支持。

构件(Component)可以定义为封装了数据、方法和实现的模块,具体特性在动态实现中设定。构件可大可小,小到一个算法模块,大到一个完整的应用系统[5]。构件系统(Component-Based system)是由构件组成的系统,有两方面的含义,一方面是指由构件生成的系统,另一方面是指构件库本身。

虽然对构件与构件系统还没有规范的、一致的定义,但是作为开发高效的、高可靠性的软件系统的重要手段,基于构件的开发模型已经成为了广泛使用的开发模式。本文的研究针对构件库的测试方法,从构件使用者的角度出发,在分析构件使用中的结构的基础上,提出基于结构模式的测试方法,同时给出形式化的描述方法。

2 构件系统测试的主要问题

对于构件库的测试,传统的测试用例(Test Case)的概念有两方面的含义,一方面为了测试构件本身,需要生成一些应用程序(Programs from components),另一方面为了测试这些应用程序,又要设计一些输入数据(Input Data for programs),如图 1 所示。由构件库可以生成的样本程序是多种多样的。本文定义应用程序一级的测试数据为样本程序集合,而对二级的输入数据定义为输入数据集合。

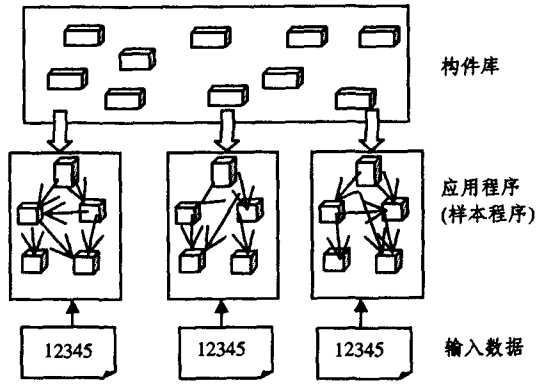

图 1 构件系统的测试数据定义

软件测试的过程可以划分为三个不同的阶段,即单元测试、集成测试、系统测试。但是,对构件的开发者(Component provider,构件的生产者)和构件的使用者(Component User,利用构件在应用领域进行二次开发者)[5~7],这三个阶段的定义和任务是不同的。

对构件系统的开发者,测试是基于代码的,所以测试的过

*)本文得到天津市科技攻关项目(项目编号:04310781R)资助。许 静 博士,副教授,主要研究方向为软件工程、软件测试、分布式系统;庞 博 硕士生,主要研究方向为软件工程、分布式计算;王庆人 教授、博导,主要研究方向为模式识别、软件工程。

维普资讯 http://www.cqvip.com

程和内容都是以设计的结构为基础展开的。而对于构件的使用者，得不到构件的源代码，在进行领域应用程序开发的过程中，也需要进行这三个阶段的测试。

(1)单元测试：由于构件本身的功能和特性已经被构件的开发者测试过了，因此我们定义构件使用者角度的单元是将两个或几个构件连接的实体。

(2)集成测试：集成的过程是根据本领域的特殊需要和构件提供的功能进行集成，形成本领域的样本程序集合，而不考虑其他领域的应用。

(3)系统测试：对样本程序集合的测试，测试内容包括在本领域的特殊功能和性能。

本文后续的讨论中，单元测试、集成测试和系统测试，都是从构件使用者角度定义。

在软件测试方法研究中要解决的问题有两个[8,9]：(1)如何选择系统的测试数据，即选择性问题；(2)选择多少数据即可，即数据充分性问题。

由于构件系统本身的特殊性，在构件系统的测试中，存在的两个主要问题是：

① 构件库提供了众多的构件，构件的使用者得不到构件的源代码，构件本身又隐藏了方法和实现，只通过接口与外部环境交互，对于构件使用者，如何由众多的构件集成而形成所需的样本程序集合，这是选择和使用构件的困难。

② 由构件库能够生成的样本程序可以有很多，结构也是多种多样，如果全部拿来进行测试，无疑造成时间和资源的浪费，而且有时也是难以做到的。所以如何选择有限的而且充分的样本程序集合，是对构件进行测试的另一个困难。

本文的研究旨在解决第一个问题。

3 基于结构模式的测试方法

3.1 软件体系结构

对构件库进行测试，需要开发样本程序作为测试数据，所以也需要多层次的设计，需要了解基于构件的开发方法和步骤。

然而，对构件系统的测试与基于构件的开发有所不同，基于构件的开发目的是生成可运行的应用程序，应用程序是单一的或特定的，所以系统有相对确定的结构和功能；而对构件系统的测试却要生成各种各样的应用程序，即样本程序集合，而且结构也要多种多样，以测试构件系统的可靠性和稳定性。所以，生成过程中要考虑构件之间的相依性。

所以需要一种方法能从系统级的角度给出构件之间的相依性关系，即系统的结构描述。为此我们引入软件体系结构的研究方法，讨论样本程序集合的生成方法。

另一方面，构件的存在某种程度上极大地依赖了构架技术，或环境、基础设施、计算平台等，只有在适当的构架中，软件才有可能被抽象和隔离，最终成为构件。因此，单独讨论构件是抽象而空洞的。构架不是操作系统、数据库或网络协议，也不完全是应用，而是在某种特定意义上的构件运行容器，层次上界于应用和基础设施之间。

3.2 基于结构模式的测试方法描述

Shaw 和 Garlan[11] 给出了设计系统体系结构时的必要元素，包括构件(Component)，连接件(Connector)，系统(System)，属性(Properties)，约束条件(Constraints)，系统模式(Styles)。

本文在借鉴体系结构描述方法的基础上，提出一种基于结构模式的集成测试方法，即构件系统的样本程序生成方法。在此方法中，除了描述构件系统的基本元素(构件、连接件)之外，重点考虑构件之间的相依性关系，包括结构相依性和语义相依性。为此，我们提出两个数据结构，用于定义这两种关系。

(1)结构相依性：在整个构件库集合中，根据构件所实现的功能的不同，构件可以分成不同的子集合，相同或相似功能的构件成为一个子集合；不同子集合中的某两个或多个构件在使用中会有一定互相依赖的关系，我们定义为结构相依性。

构件系统的结构相依性是从构件模型规约和领域模型中获得的。步骤如下：

1)建立领域模型，分解本领域的功能需求。

2)分析构件模型规范，划分构件为不同的子集合 A(为基础构件层)，子集合 B(上层构件)，子集合 C(上层构件)，等等；定义基础构件层为整个系统的语义构建层或语义解释层。

例如，VS. Net 提供的 Speech SDK 构件集合，可以形成基于 Web 的语音交互应用系统，根据构件集合中构件的不同功能，可以划分为三个子集合，语音交互构件子集合、页面显示构件子集合、命令响应及解析构件子集合。其中，对命令的响应及解析，是对系统的语义解释，此构件子集合定义为系统的基础构件层。

3)由基础构件层进行选择，形成结构的相依性关系；本文定义从系统的基础构件层到其他层的构件的结构依赖关系为结构相依性关系。

例如，对系统的不同语义命令解析的构件，需要语音的交互构件和识别构件，对解析的结果又需要页面的显示结构的支持，所以从语义解析构件层到其他层的构件都有一定的结构依赖关系，定义为系统的结构相依性关系。

4)给出从基础构件层到其他构件层的结构关系，用树形结构表示；在树形结构中，本文定义基础构件层的构件为根节点，在与其它层的结构关系中有两种情况：单向关系和双向关系；单向关系表示两个构件之间的结构支持和依存关系，双向关系除了表示结构的依存关系外，还表示结果数据的绑定关系。

例如：假设系统的构件集合根据功能的不同可以分成三个子集合，子集合 A 为基础构件层，构件有 $A1, A2, A3, \cdots\cdots$；子集合 B 有构件 $B1, B2, B3, \cdots\cdots$；子集合 C 有构件 $C1, C2, C3, \cdots\cdots$；在样本程序形成中的 $A1$ 在结构上依赖 $B1$ 和 $C2$，而 $C2$ 在完成功能的数据流动中，又需要动态绑定到 $A1$；等等。系统的结构相依性图可以表示为图 2，为了后续的处理方便，将图 2 转化为三棵结构相依性树，分别以子集合 A 的构件为根节点，如图 3。

(2)语义相依性：在同一构件子集合中，为完成应用系统的某种功能，必须按几个构件的某种顺序使用才能完成，本文定义这样的顺序为这几个构件之间的语义相依性。

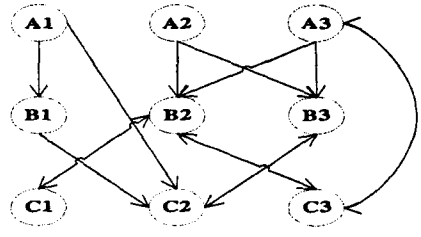

图 2 结构相依性图

维普资讯 http://www.cqvip.com

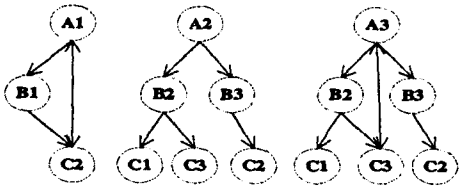

图 3 结构相依性树

例如，在 Speech SDK 构件集合形成语音应用程序中，语音交互功能的实现是按照一定的构件使用顺序的，先进行语音提示(Prompt)，再进行问答(Answer)，然后再进行对回答的验证操作(Validate)，所以这一集合的构件使用就有一定的顺序关系，称为构件的语义相依性。有些构件子集合中，由于构件功能的独立性，可以没有语义相依性关系。

构件子集合内的语义相依性是从构件模型规约和 Use Case 模型得到的。步骤如下：

5)建立 Use Case 模型，得到系统功能的准确语义表示。

6)解析 Use Case 模型，得到系统命令处理的解析树，再根据构件规范，建立构件子集合 A,B,C 内部的顺序关系；如果本构件子集合中的构件没有这样的顺序关系，则本构件层没有语义相依性关系。

7)给出每一构件层内部的语义相依性关系，由于构件之间有顺序关系，因此用矩阵来表示，如果本构件层有 N 个构件，则矩阵为 $N*N$。

$$\begin{bmatrix} a_{11} & a_{12} & \cdots\cdots & a_{1n} \\ a_{21} & a_{22} & \cdots\cdots & a_{2n} \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \\ a_{n1} & a_{n2} & \cdots\cdots & a_{nn} \end{bmatrix} \qquad a_{ij}=\begin{cases}0\\1\end{cases}$$

当构件层 A 中的 a_i 必须在 a_j 之前使用时，$a_{ij}=1$；否则，$a_{ij}=0$。

4 形式化描述方法

4.1 ADL 介绍

体系结构描述语言(Architecture Description Language，ADL)是对软件体系结构的一种形式化的描述方法。是软件体系结构组织的热门课题，在每年的软件体系结构会议中，都有大量的 ADL 方面的研究涌现[12,13]。

应用已有的 ADL 用于构件系统的测试有如下的障碍：

(1)已有的 ADL 主要解决系统的架构问题，所以其重点在于对不同系统的模式设计；而对于构件系统的测试，要解决的主要问题在于由已有的构件形成不同的应用程序，所以系统的重点在于构件之间的连接问题，不同的连接可以组成不同结构的应用程序。

(2)已有的 ADL 解决系统的设计问题时，只考虑构件或系统的复用，没有考虑应用系统的自动生成问题；而在测试过程中，要考虑测试的充分性和效率问题，所以模型的描述应能支持代码的自动转换。已有的 ADL 没有针对测试问题提出的，所以不能满足测试的要求。

为此本文定义一种专门用作测试的体系结构描述语言 TADL(Testing ADL)。TADL 是对本文提出的基于结构模式的测试方法的形式化描述，同时，TADL 也是一种 ADL，也应具备 ADL 的某些特征。

4.2 TADL 的模型描述

构件(Components)：构件的描述可以分成构件接口、语义信息和实现。一般的构件模型可以表述成：

构件::=〈构件接口，语义信息，构件实现〉

在构件系统的测试中，构件的来源主要是 COTS，它们本身都有各自的模型，所以在 TADL 中定义了与测试有关的特征。包括特征描述接口(Interface)和实现接口(Implementation(ports))。

(1)特征接口是指构件本身的形象描述特征，例如显示特征，标识特征等。

(2)实现接口包括了对外提供的功能接口和对外要求的功能接口，对外提供的功能描述了构件所实现方法的外部参数、触发条件、结束条件、时间、延时等。对外要求的功能接口描述了构件实现的前提条件，包括了引用类型、参数传递和数据绑定等，所以 TADL 中的构件模型可以表述成：

构件::=〈构件特征接口，构件实现接口〉

构件特征接口::=〈显示特征，标识特征〉

构件实现接口::=〈对外提供的功能接口，对外要求的功能接口〉

对外提供的功能接口::=〈过程参数，触发条件，结束条件〉

对外要求的功能接口::=〈引用类型，参数传递，数据绑定〉

另外，在一个构件集合中，每个构件所完成的功能都是不同的。按照构件完成功能的差异，我们将具有相似功能的构件集合在一起，形成构件的子集合 ComponentSet，不同的构件子集合在形成应用程序的过程中处于不同的层，最底层的构件子集合定义为基础构件层。TADL 中的一个构件子集合包括了定义的构件层名和子集合中的基本元素。

连接件(Connectors)：表示构件之间的连接方式。目前的 ADL 多数研究的是 Client/Server 体系结构，所以定义的连接件限定在 RPC，Pipe 等[11]，没有更细的更丰富的连接件模型，而在测试构件系统时，关心的是由构件可以组成的不同结构，所以连接件的类型和模型表示是 TADL 中的关键。

通过对 COM 对象模型的分析和 COM 在实际应用中的总结，概括出在生成服务器端的应用程序时，所用到的典型的构件连接件有：

(1)功能调用(Function Invocation)：通过对某一主类的实例化，来完成某种功能。

(2)消息传送(Message Passing)：通过消息的传送来实现某一功能。

(3)事件驱动(Event Driver)：由事件来驱动另一构件功能的实现。

(4)条件触发(Condition Trigger)：由单个条件或多个条件的组合以驱动另一功能的实现。

(5)回调(Call Back)：主要用于异步处理，发消息到系统消息队列，继续执行，毋须等待，系统完成后，由系统回调进行信息反馈。

(6)多线程服务(Multi-thread Service)：服务器端构件，同时处理多个线程的请求。

TADL 的连接件的模型，主要包括两部分：角色分配描述和特性描述。角色(Role)定义了相互连接的构件之间的期望行为，规范了参加连接的构件的职责。通过设定不同的角色，定义了构件之间的操作规则，增加了构件之间操作的可追踪性，实现构件的连接。

特性描述定义了构件连接过程中的部分特性，包括角色调用顺序，可以同时使用的用户数量最大值，同步或异步，使用的协议等。所以，TADL 中的连接件可以表述成：

连接件::=〈角色定义，特性描述〉

角色定义::=〈构件行为 1，构件行为 2〉

特性描述::=〈行为顺序，用户数量，同步或异步，使用协

维普资讯 http://www.cqvip.com

议〉

系统结构(SystemStructure):包括系统整体结构和构件关系结构,系统整体结构定义构件与连接件的整体关系,构件关系结构定义不同构件子集合中的构件之间的关系。TADL中的系统结构可以表述成:

系统结构::=〈系统整体结构,构件关系结构〉

系统整体结构::=〈构件子集合,连接件子集合,基础构件子集合,组织规范〉

构件关系结构::=〈原子结构,指针〉

原子结构::=〈构件名,所处子集合名,关系定义〉

关系定义::=〈单向关系|双向关系〉

单向关系::=〈方向标志,原子结构,指针〉

双向关系::=〈方向标志,原子结构,指针〉

构件语义约束(Semantic Map):描述了构件之间的语义关系。系统结构描述的是构件之间的拓扑关系,而构件语义约束则从构件功能定义出发,在构件的不同子集合内部,给出构件使用时的语义连接顺序和规则。在TADL中,根据构件子集合本身的功能特点,为必要的构件子集合定义了构件语义关系。如果构件子集合有 n 个构件,则此构件子集合的构件语义约束为二维的 $n*n$ 的表结构,构件之间的语义关系是有方向的,此结构为自动生成领域应用程序提供了生成规则。

```
SemanticMap ={
        ComponentSetName;
        Table ={ a_ij }n*n // 当第 i 个构件到第 j 个构件由一
                个顺序规则时,a_ij =1
        }
```

系统模式(Styles):表示一类体系结构族,或称风格。目前的体系结构风格主要有:客户端/服务器式体系结构,管道式体系结构,对象连接式体系结构,接口连接式体系结构,插头插座式体系结构和分层递阶式体系结构[11]。在TADL中,目前只定义了一类结构,就是分层的结构模式。

系统模式::=〈构件子集合,连接件集合,系统结构,构件语义约束,系统规范〉

5 样本程序生成工具设计

本文定义的形式化描述语言TADL可用于描述系统的模型,同时提供了丰富的连接件类型来表达系统构件之间的通信关系,对每一个系统元素的定义都是基于实现的,所以为样本程序生成提供了方便的数据结构。

对于构件系统样本程序的生成,要解决的主要问题是:①如何从众多的构件集合中选择需要的构件;②按照怎样的结构来集成构件,以形成样本程序。本文提出的TADL通过定义构件的体系结构给出了集成构件,形成样本程序的方案;通过定义构件子集合间的结构关系和构件子集合内的语义关系,给出了选择构件的方案。具体过程包括:①建立应用领域的领域模型;②建立Use Case模型;③解析Use Case模型、划分构件集合、对应构件基础层;④定义构件子集合内的语义关系;⑤生成不同构件子集合间的结构关系树;⑥集成构件,形成样本程序。

结束语 本文在分析构件系统的特点的基础上,提出了一种基于结构模式的测试方法,增加对构件系统内部不同层次的信息描述,包括不同构件子集合间的结构关系描述,以及构件子集合内部的语义关系描述,为构件的使用者集成构件生成测试用样本程序提供必要的信息。同时给出了本方法的形式化描述,用于对待测构件系统的模型描述,便于转化为可执行代码,实现样本程序的生成工具。本方法同样适合构件的开发者进行集成测试和系统测试,为构件系统提供安全保证,使得这种开发模式发挥更大的作用。

参 考 文 献

1 Weyuker E J. Testing Component-Based Software: A cautionary tale. IEEE Software, Sept.-Oct. 1998
2 Kozaczynski W, Booch G. Component-based Software Engineering, IEEE Software, Sept.-Oct. 1998
3 Voas J. Maintaining Component-Based System. IEEE Software, July-Aug. 1998
4 Rosenblum D S. Challenges in Exploiting Architectural Models for Software Testing. In: Intl. Workshop on The Role of Software Architecture in Testing and Analysis, Marsala, Sicily Italy, 1998
5 Harrold M J. An Approach To Analyzing and Testing Component-Based system. In: ICSE'99 Workshop on Testing distributed Component-Based System, May 1999
6 Orso A, Harrold M J. Component Metadata For Software Engineering Tasks. Proc. of EDO 2000, LNCS Vol 1999
7 Harrold M J. Testing: A Roadmap. In: 22nd Intl. Con. on Software Engineering, June 2000
8 Rothermel G, Harrold M J. Analyzing Regression Test Selection Techniques. IEEE Transactions on Software Engineering, Aug. 1996
9 Rothermel G, Harrold M J. Empirical Studies of a Safe Regression Test Selection Technique. IEEE Transactions on Software Engineering, June 1998
10 Binder R V. 面向对象系统的测试. 人民邮电出版社, 2002
11 Shaw M, Garlan D. Software Architecture. Prentice Hall, April 1996
12 Allen R J, et al. Formal Modeling and Analysis of the HLA Component Integration Standard. In: Proc. of the Sixth Intl. Symposium on the Foundations of Software Engineering (FSE-6), Nov. 1998
13 Allen R, Garlan D. A Formal Basis for Architectural Connection. ACM Transactions on Software Engineering and Methodology, July 1997

(上接第160页)

在远程教育、企业客户咨询等方面。广阔的应用前景正推动着自动问答技术的快速发展,相信在不久的将来问答系统的研究将会取得重大的突破并得到广泛的应用。实现计算机理解自然语言的提问正是计算机理解人类语言的开始。

参 考 文 献

1 http://www.ai.mit.edu/projects/infolab/
2 http://misshoover.si.umich.edu/~zzheng/qa-new/
3 Chang Yi, Xu Hongbo, Bai Shuo. TREC 2003 Question Answering Track at CAS-ICT. In: The Twelfth Text Retrieval Conf. Gaithersburg, Maryland, 2003. 147~152
4 Wu Lide, Huang Xuanjing, Zhou Yaqian, et al. FDUQA on TREC2003 QA task. In: The Twelfth Text Retrieval Conf. Gaithersburg, Maryland, 2003. 246~254
5 郑实福,刘挺,秦兵,李生. 自动问答综述. 中文信息学报, 2002, (6): 46~53
6 Li X, Roth D. Learning Question Classifiers. In: Proc. of the 19th Intl. Conf. on Computational Linguistics (COLING). Taipei, Taiwan, 2002. 556~562
7 Zhang D, Lee W S. A Language Modeling Approach to Passage Question Answering. In: The Twelfth Text Retrieval Conf. Gaithersburg, Maryland, 2003. 489~496
8 Ravichandran D, Hovy E. Learning Surface Text Patterns for a Question Answering System. In: Proc. of the 40th Annual Meeting of the Association for Computational Linguistics (ACL). Philadelphia, July 2002. 41~47
9 Wang G, Chua T-S, Wang Y-C. Extracting Key Semantic Terms from Chinese Speech Query for Web Searches. In: The Eleventh Text Retrieval Conf. Gaithersburg, Maryland, 2002. 389~396
10 Wu Min, Zheng Xiaoyu, Duan Michelle, et al. Question Answering By Pattern Matching, Web-Proofing, Semantic Form Proofing. In: The Twelfth Text Retrieval Conf. Gaithersburg, Maryland, 2003. 578~586
11 Voorhees E M. Overview of the TREC 2003 Question Answering Track. In: The Twelfth Text Retrieval Conf. Gaithersburg, Maryland, 2003. 54~69