一种基于软件体系结构配置的测试用例生成算法研究*)

叶俊民[1] 赵 恒[2] 黄 萍[2] 王振宇[2]
(华中师范大学计算机科学系 武汉 430079)[1]
(中船重工集团第七研究院第709研究所 武汉 430070)[2]

摘 要 从软件体系结构配置生成测试用例是软件测试领域中的一个重要分支。本文首先提出了一组基于软件体系结构配置的测试标准及其计算规则。基于此,实现了基于软件体系结构配置的测试用例生成算法并分析了其时间复杂性。对人机结口(Man-Machine Interface,MMI)的实验表明,所提出的算法能够根据测试标准生成测试用例。

关键词 软件体系结构,测试用例生成算法,软件测试

A Software Architecture Configuration-Based Test Cases Generation Algorithm

YE Jun-Min[1] ZHAO Heng[2] HUANG Ping[2] WANG Zhen-Yu[2]
(Department of Computer Science, Huazhong Normal University, Wuhan 430079)
(709 Research Institute, China Shipbuilding Industry Corp, Wuhan 430070)

Abstract An important branch in the field of software testing is to generate test cases from software architecture configuration. In this paper, we firstly propose a set of configuration-based testing criteria and their computational rules. On this basis, we propose a software architecture configuration-based test cases generation algorithm. The algorithm is implemented and its time complexity is analyzed in this paper. Experiments with MMI(Man-Machine Interface) indicate that the proposed algorithm can generate test cases according to testing criteria.

Keywords Software architecture, Test cases generation algorithm, Software testing

1 引言

软件体系结构设计质量的好坏对未来软件的构造影响极大,因此有必要对软件体系结构进行测试。基于软件体系结构配置的测试是软件体系结构测试的重要组成部分。软件体系结构测试是人们对软件测试认识加深的一种反映,这种认识对软件测试具有深刻影响。软件体系结构配置说明了体系结构元素之间的拓扑结构,是构件与连接件的连接图示,描述了体系结构的结构信息。通过配置信息可以检查构件与连接件的连接是否适当,连接件能否正常通信,它们经过组合后所产生的语义与其所希望的行为是否一致。基于软件体系结构配置信息测试的核心,就是利用这些配置信息描述、产生测试用例。

软件体系结构测试技术的主要工作进展表现在:N. S. Eickelmann 和 D. J. Richardson 等人[1]从可测试角度出发,思考了将软件体系结构规格说明作为软件测试的基础这一问题,认为在软件体系结构层次考虑软件测试是可行的。P. Inverardi 和 A. L. Wolf[2]对化学抽象机(Chemical Abstract Machine,CHAM)的研究工作,以及 A. Bertolino[3,4]提出从软件体系结构的描述中导出测试用例的思想,为软件体系结构的分析和测试研究开创了先河。Jin Zhenyi 和 J. Offutt 等人[5]针对体系结构描述语言 Wright,从配置角度研究了测试用例生成技术。国内的东南大学也对软件体系结构配置测试进行了研究[6]。这些研究主要是基于特定软件体系结构描述语言,如 CHAM 和 Wright 等。本文的研究针对一般性的软件体系结构描述语言展开。

为了解决基于体系结构配置信息的测试用例生成问题,核心问题是:1)选择一种数据结构,以描述软件体系结构配置信息;2)定义软件体系结构层的测试覆盖准则,这是因为测试覆盖准则反映了测试目标和测试者的兴趣,是产生有效测试用例的保障;3)在此基础上,设计出一种测试用例生成算法。

2 描述软件体系结构配置信息的数据结构

定义 1(配置行为图,CBG)

CBG=(V,E)是一个由 V 和 E 两个集合构成的有向无环图,$V=V1\cup V2\cup V3$ 是非空有限的顶点集合。其中 $V1$ 是构件顶点的有限集合,$V2$ 是连接件顶点的有限集合,$V3$ 是端口顶点的有限集合;E 是 V 中的顶点对的有限集合,$P(V)$是各类顶点之间卫式条件关系描述的集合。

3 软件体系结构测试覆盖准则及其计算规则

为了有选择性地产生测试用例,需要一组体系结构层的测试覆盖准则。为此,我们选择了 J. Offutt 等人提出的测试覆盖准则[5],并对这些准则给出我们的定义。J. Offutt 等人提出的测试覆盖准则包括:构件内部依赖关系覆盖准则、连接件内部迁移覆盖准则、N2C 覆盖准则、C2N 覆盖准则、直接 N2N 覆盖准则、间接 N2N 覆盖准则和全连接构件覆盖准则。

从文[5]上看,J. Offutt 等人并未对上述测试覆盖准则进行明确定义。下面我们将给出这些测试覆盖准则的定义。

1)构件内部依赖关系覆盖准则。要求:软件体系结构中所有构件端口之间的内部数据流或控制流依赖关系至少分析一次;或者,软件体系结构中的所有构件端口之间的行为应遵

*)本课题得到国防科技预研基金(413150601)资助。叶俊民 副教授、博士,主要研究方向为软件体系结构及其测试;赵 恒 博士生;黄 萍 高工;王振宇 研究员、博士生导师,主要研究方向为新型软件测试技术。

守某些执行规则的情况至少分析一次。

2)连接件内部迁移覆盖准则。要求:软件体系结构中所有连接件端口之间的内部数据流或控制流依赖关系至少分析一次;或者,软件体系结构中所有连接件端口之间的行为应遵守某些执行规则的情况至少分析一次。

3)N2C 移覆盖准则。要求:软件体系结构中的所有构件端口与对应的连接件端口之间存在的关联关系至少分析一次。

4)C2N 覆盖准则。要求:软件体系结构中的所有连接件端口与对应的构件端口之间存在的关联关系至少分析一次。

5)直接 N2N 覆盖准则。要求:软件体系结构中的 N2C、C2N 和直接构件的内部关系至少分析一次。

6)间接 N2N 覆盖准则。要求:如果构件 Ni 到构件 $Nj(i\neq j)$ 之间存在非直接关联路径,则该路径至少分析一次。

7)全连接构件覆盖准则。要求:一组构件 $N1,N2,\cdots,Ns$ 和一组连接件 $C1,C2,\cdots,Ct$,如果存在一个关系,将这些构件与连接件组合在一起,则结果路径 $N1C1\cdots CtNs$ 至少分析一次。

为了应用上述测试覆盖准则,我们以测试路径方式定义计算软件体系结构测试覆盖准则的计算规则。设 InfoTranInComponentPort(…)表示求通过构件端口的信息流迁移功能;InfoTranInConnectorPort(…)表示求通过连接件端口的信息流迁移功能;N2Cpath(…)表示求构件到连接件(N2C)路径功能;C2Npath(…)表示求连接件到构件(C2N)路径功能;DN2Npath(…)表示求直接构件到构件路径(DN2N)功能;IDN2Npath(…)表示求间接构件到构件(IDN2N)路径功能;AllPath(…)等函数分别表示求全路径(AllPath)功能,则上述 R1-R7 对应的计算规则可形式化描述如下:

R1(构件内部依赖路径计算规则):∃result{InfoTranInComponentPort(N. Ii,N. Ij)∧ ∨ result="DATA TRANSFER" ∨ result="CONTROL_TRANSFER")→return(result)};

R2(连接件内部迁移路径计算规则):∃result{InfoTranInConnectorPort(C. Ii,C. Ij)∧(result="DATA_TRANSFER" ∨ result="CONTROL_TRANSFER")→return(result)};

R3(N2C 路径计算规则):∃result{N2CPath(N. I:Interface_Typei,C. I:Interface_Typej)∧(result="Match")→return(result)};

R4(C2N 路径计算规则):∃result{C2NPath (C. I:Interface_Typei,N. I:Interface_Typej)∧ (result="Match")→return(result)};

R5(直接 DN2N 路径计算规则):∃result{DN2Npath(Ni. I:Interface_Typei,Nj. I:Interface_Typej)∧ (result="Match")→ return(result)};

R6(直接 DN2N 路径计算规则):∃result{IDN2Npath(Ni. I:Interface Typei,Nj. I:Interface_Typej,Nk. I:Interface_Typek)∧ (result="Match")→ return(result)};

R7(全连接构件覆盖计算准则):∃result{AllPath(Vi,Vj)∧Vi∈V∧Vj∈V∧i≠j∧(result="Match")→return(result)}。

4 基于 CBG 图描述的软件体系结构配置信息的测试用例生成算法

4.1 软件体系结构配置信息的测试用例生成过程

基于体系结构配置信息的测试用例生成过程如图 1 所示,即将软件体系结构描述语言换成 CBG 图形式,依据测试覆盖准则,可产生测试用例。

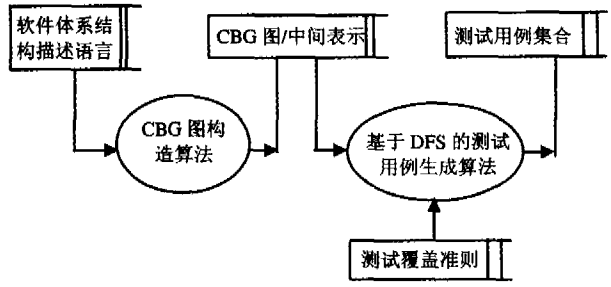

图 1 基于体系结构配置信息的测试用例生成过程

4.2 基于配置信息的测试用例生成

基于配置信息的测试用例生成,主要包括如下步骤:(1)通过软件体系结构描述语言构造 CBG 图和配置中间信息,并进行连接元素之间的类型匹配分析;(2)依据 CBG 图及其配置中间信息,分析软件体系结构描述语言中元素的依赖关系;(3)在(1)和(2)的基础上,利用测试覆盖准则,产生测试用例。

4.2.1 构造 CBG 图

在按照定义构造 CBG 图的同时,需要构造基于配置描述的各种中间信息,这些中间信息提供了用于分析的各类信息:①所有的构件名;②所有的连接件名;③所有的接口名;④所有的接口类型;⑤所有入口及其信息流类型;⑥所有出口及其信息流类型;⑦所有组合构件;⑧一个组合构件中的任意两个接口之间路径;⑨体系结构连接关系表。我们注意到,构件到连接件之间的连接关系、构件到构件之间的连接关系,以及构件到组合构件之间的连接关系集合,都可以通过体系结构连接关系表推导出来。

算法 1 CBG 及其中间表示的构造

输入:软件体系结构描述语言源文件。

输出:CBG 图及其配置中间文件。

step1. 初始化;//初始化各类数据结构

step2. 构造 CBG 图:

step2.1 扫描软件体系结构描述语言源文件,登记配置信息中的①~⑦项;

step2.2 构造约束表:如果软件体系结构描述语言中的单一构件、组合构件、单一连接件、组合连接件和系统描述中存在约束信息,则将其登记到对应的约束信息表中;

step2.3 构造连接关系并进行接口类型匹配分析,即记录配置中间信息中的⑧~⑨:

While 对每一条软件体系结构描述语言中的配置描述信息 do

{ 分析构件与连接件之间的连接关系;

分析元素之间的类型匹配关系;

if ∃连接关系∧∃类型匹配关系 then

{ 按如下规则,构造体系结构连接关系表:如果一个构件 $N1$ 的一个端口与一个连接件 C 的一个端口存在连接关系,且如果该连接件的另一个端口连接一个不同的构件 $N2(N1\neq N2)$,则可构造出构件 $N1$ 与构件 $N2$ 之间的连接关系;}}

4.2.2 体系结构元素之间的依赖分析

算法 2 体系结构的依赖分析

输入:CBG 图及其配置中间文件。

输出:元素之间依赖关系分析的结论。

step1. 初始化;//读入各种元素信息

step2. while∃简单元素 do

// 简单元素的结构依赖分析,简单元素包括单一构件或单一连接件

step2.1 如果该简单元素只有移出接口,则规定该简单元素中的内部依赖关系满足;

step2.2 if 该简单元素只有移入接口 then 产生出错信息;

step2.3 if ∃该简单元素的移入接口∧ ∃移出接口

then { 检查该简单元素是否存在约束条件;

if →∃约束条件 then 产生提示信息;

else 根据元素的移入端口行为、移出端口行为和约束条件,分析该元素的内部依赖关系并返回分析结果;}

endwhile;

step3. while ∃组合元素 do

//组合元素的结构依赖分析,组合元素包括组合构件和组合连接件

step3.1 //结构依赖分析

分析组合元素内部可达关系,即分析组合元素的移入端口、移出端口和组合元素的内部元素所组成的连接关系是否可达。规则如下:

step3.1.1 if 组合元素只有出口的情况

then ① 考察该组合元素的配置关系,即考察该组合元素的每个内部元素是否存在着逆向可达到该组合元素的内部首元素的路径;

② 如果不存在①的路径,则返回出错信息;

step3.1.2 if 组合元素只有入口的情况 then 产生出错信息;

step3.1.3 if ∃该组合元素的移入接口∧∃移出接口 then

{ 考察该组合元素的移入接口和移出接口之间是否存在可达路径;

if ∃可达路径 then 返回结构依赖关系满足的结论;else 返回结构依赖关系不满足的结论;}

step3.2 //行为依赖分析

/*验证组合构件中的接口行为或计算行为是否满足组合构件定义的约束条件,返回行为依赖关系是否满足的判定结论;*/

if ∃该组合元素的移入接口∧∃移出接口 then

{ 检查该组合元素是否存在约束条件;

if→∃约束条件 then 产生出错信息;

else 根据组合元素的移入端口行为、移出端口行为和约束条件,分析该组合元素的内部依赖关系;}

endwhile;

4.2.3 测试用例生成算法

算法 3 基于 CBG 图的配置信息测试用例生成

输入:软件体系结构描述语言源文件;测试覆盖准则编号。

输出:测试用例集合。

step1. 依据软件体系结构描述语言源文件,使用算法 1,生成 CBG 图和配置中间文件;

step2. 依据 CBG 图和配置中间文件,使用算法 2,对该软件体系结构描述语言描述进行依赖关系分析;如果依赖分析出错,则输出出错信息,并终止本算法,否则转 step3;

step3. 依据测试覆盖准则编号和体系结构连接关系表,做如下步骤:

switch 测试覆盖准则的计算规则编号{

case='R1':依据 R1 进行分析并给出结论;break;

case='R2':依据 R2 进行分析并给出结论;break;

case='R3':依据计算规则 R3,遍历 CBG 图并生成测试路径,记录在 N2C 中;break;

case='R4':依据计算规则 R4,遍历 CBG 图并生成测试路径,记录在 C2N 中;break;

case='R5':依据计算规则 R5,遍历 CBG 图并生成测试路径,记录在 DN2N 中;break;

case='R6':依据计算规则 R6,遍历 CBG 图并生成测试路径,记录在 IDN2N 中;break;

case='R7':依据计算规则 R7,遍历 CBG 图并生成测试路径,记录在 CCOM 中;break;

default:break;}

step4. 合并所有生成的测试路径并记录在文件中:Stru_Based_TestingCase:=N2C∪C2N∪DN2N∪IDN2N∪CCOM;

4.3 基于配置信息的测试用例生成算法复杂性分析

算法 1 的本质为构造 CBG 图,显然其计算时间复杂性为多项式时间。设 CBG 中有 e 条边,利用测试覆盖准则,则算法 3 的算法的计算时间复杂性为 $O(e)$。已知进行依赖分析的算法 2 的计算时间复杂性至多为多项式时间完备的[7]。因此,基于软件体系结构配置信息的测试用例生成算法的计算时间复杂性是多项式时间。

5 实验情况

我们在一台 P4 机(主频 1.8GHz,128M 内存)上,对某 C^3I 中的人机交互接口 MMI(Man-Machine Interface)系统实例进行了实验,实验结果如表 1 所示。

表 1 MMI 实例的测试结果

	C^3I 系统中的 MMI 的实验情况
单一构件数目	20
单一连接件数目	38
两个端口之间的路径数	155
构件内部依赖分析	OK
连接件内部依赖分析	OK
N2C 路径数	536
C2N 路径数	599
直接 N2N 路径数	36
间接 N2N 路径数	415
全连接路径数	2960

结束语 本文设计了一种基于体系结构的配置信息和测试覆盖准则的测试用例生成算法。实验初步表明,该算法能

(下转第封三)

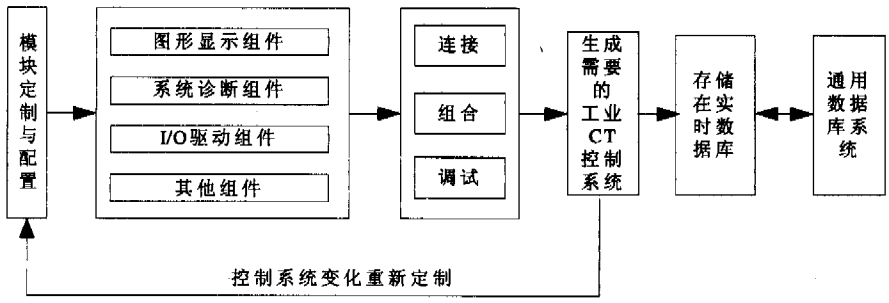

图4 工业CT控制系统组件组装简图

4 应用

本文采用COM(Component Object Model,组件对象模型)技术开发各个组件模块。下面以参数设置组件为例来说明其接口和调用,得到某工业CT控制系统的参数设置界面图,如图5所示。

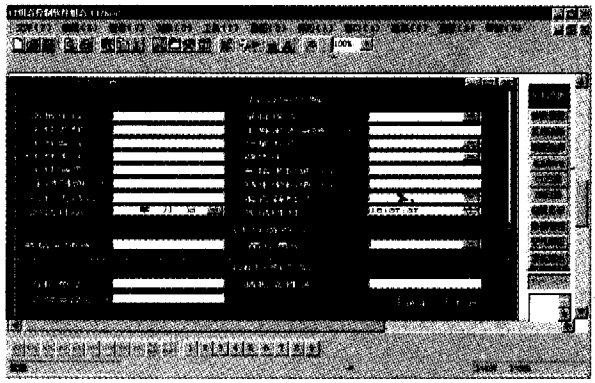

图5 工业CT控制系统参数设置界面图

参数设置组件(PropertySetUp)主要用来生成一设置系统扫描参数和设置视频窗体大小的对话框,并将设置好的参数以消息的形式传递给数据库管理组件。供其他组件读取获得其动作的输入参数。

在PropertySetUp组件中存在着以下几个接口:

• Register_Class:用于在搭建好的工业CT开发平台进行注册。

• Set_Property_P:用于为P赋一个新值。P可以是组件内部状态的一个真实的变量,也可以是一个虚属性。

• Get_Property_P:与Set_Property_P对应,用于读取属性值。

• Initialize_State:用一个包含初始化的参数进行调用,对组件内部的状态进行初始化。

• Save_State:保存数据。

• Create_Form:创建所需对话框。

• Send_Data:返回参数到数据库管理组件。

结束语 组件技术大大提高了软件代码的重用性,可以在"远远高于语句级"的层次上进行开发。在工业CT控制软件开发中,当用户需求发生变化的时候,可以通过组件的重新选择配置来迅速构成新的控制系统,有效地解决了工业CT控制软件定制开发问题,缩短了控制系统的开发周期,降低了开发成本。

参 考 文 献

1 潘爱民. COM原理与应用[M]. 北京:清华大学出版社,1999

2 邹益仁. 现场总线控制系统的设计和开发[M]. 北京:国防工业出版社,2003

3 马国华. 监控组态软件及其应用[M]. 北京:清华大学出版社,2002

4 HU Wen-Hu. Study by Application Framework Meta-Model Based Component Technology [J]. JOURNAL OF SOFTWARE, 2004,15(1):1~8

5 CHEN Li. Component-Based Design of Industry Control Software [J]. JOURNAL OF SYSTEM SIMULATION, 2001, 13(4):446~449

6 肖兵. 实时控制系统的面向对象软件结构与实现方法[J]. 计算机研究与发展,1995,32(8):49~53

7 LI Xia. Configuration Approach for Open Architecture Controller [J]. COMPUTER INTEGRATED MANUFACTURING SYSTEM,2004,10(6):672~676

(上接第268页)

够依据相关的计算策略产生测试用例。

参 考 文 献

1 Eickelmann N S, Richardson D J. What Makes One Software Architecture More Testable Than Another? In: Proceedings of the 21st International Software Architecture Symposium, 1996. 65~67

2 Inverardi P, Wolf A L. Formal Specifications and Analysis of Software Architectures Using the Chemical Abstract Machine Model. IEEE Transactions on Software Engineering, 1995, 21(4): 100~114

3 Bertolino A, Inverardi P, Muccini H. Deriving Test Plans from Architectural Descriptions. In: ACM Proc. Int Conf on Software Engineering(ICSE2001), 2000. 220~229

4 Bertolino A. An Approach to Integration Testing Based on Architectural Descriptions. In: Proceedings of 3th International Conference on Engineering of Complex Computer Systems. IEEE Computer Society Press, 1997. 77~84

5 Jin Zhenyi, Offutt J. Deriving Tests from Software Architctures. In: The Twelfth IEEE International Symposium on Software Reliability Engineering, Hong Kong, PRC, Nov. 2001. 308~313

6 聂长海,徐宝文. 软件配置测试的测试方案设计. 软件学报,2003,14(增刊):149~154

7 古天龙,蔡国永. 网络协议的形式化分析与设计. 北京:电子工业出版社,2003. 163~176