

引用格式:王影.可移植的软件集成测试平台设计与实现[J].微电子学与计算机,2019,36(11):37-42. [WANG Y. Design and implementation of a transplantable software integration test platform [J]. Microelectronics & Computer,2019,36(11):37-42]

可移植的软件集成测试平台设计与实现

王 影

(中国航发商用航空发动机有限责任公司 设计研发中心控制系统部,上海 200241)

摘 要: 为了构建通用的软件集成测试平台框架,设计了三类用于关联测试平台与被测软件的容器:1)“混合类型数据容器”用作访问软件接口的基础数据类型,对被测软件不同类型的输入、输出数据进行统一兼容性处理;2)“部件接入容器”用于封装软件部件,利用容器中的函数指针分量,间接调用软件部件入口模块;3)“部件接口访问容器”作为对部件输入、输出接口变量间接访问的媒介,实现测试平台与被测对象的解耦合.在此基础上,将测试平台框架划分为可变的初始化模块与固定的周期调度控制模块,实现测试平台与被测软件之间的逻辑隔离.案例分析表明,通过正确配置各类容器分量,方便实现软件集成测试平台的跨软件移植和复用.

关键词: 软件集成测试;测试平台;设计方案;移植;复用;通用

中图分类号: TP31

文献标识码: A

文章编号: 1000-7180(2019)11-0037-06

Design and implementation of a transplantable software integration test platform

WANG Ying

(R&D Center Dept. of Control Systems, AECC Commercial Aircraft
Engine CO.,LTD., Shanghai 200241, China)

Abstract: In order to construct general purpose used software integration testing platform architecture, three kinds of vessels are designed. Firstly, through using mixed type data vessels as basic data type to access software interfaces, various types of input and output data are treated uniformly for compatibility. Secondly, with the adoption of part connection vessels to encapsulate software components, the function pointer portions of the vessels are utilized to call the entry module of the software component indirectly. Finally, under the support of component interfaces accessing vessels, the input and output interface variables of components being tested are accessed indirectly, which decouples the test platform from the test objects. On this basis, the testing platform architecture is divided into variable initialization module and constant periodic scheduling control module, which realizes logical isolation between the platform and the software under test. Practical application analysis demonstrated that the software integration test platform is portable and reusable across software via proper configuration of various portions of the vessels.

Key words: software integration test; test platform; design proposal; transplant; reuse; generic

1 引言

目前软件测试平台主要有为单个软件定制的

专用测试平台^[1],为相同系列^[2-3]或接口兼容^[4-5]的软件构建的共享测试平台,为周期性运行的嵌入式软件构建的部件仿真测试平台通用方案^[6],支

持部件动态集成的软件集成测试平台通用方案^[7-8]等. 以上测试平台能够满足大多数软件测试要求,但是均未考虑可移植性,因此,测试平台的扩展能力有限,跨软件重构测试平台的成本较高. 为了增强软件测试平台跨软件复用的能力,郭素敏等人^[9]采用“软件插槽+构件”的开放通用的软件体系结构,测试开发工作简化为选用或二次开发满足特定需求的构件模块,显著减少不必要的重复劳动. 为了进一步消除测试平台的重复建设活动,解除测试平台与被测软件的耦合关系,本文在文献[6-7]的工作基础上,提出了一种可移植的全数字仿真软件集成测试平台方案,通过在测试平台与被测软件之间插入特殊“容器”,阻断二者直接交互的控制流和数据流路径,有效实现了逻辑隔离和跨软件复用.

2 容器设计说明

为了实现测试平台与被测软件之间的逻辑隔离,本文设计了三种类型的“容器”.

2.1 部件接入容器

将被调用的软件部件使用“部件接入容器”进行封装,容器以“部件调度结构体”表示,在 C 语言中定义如下:

```
typedef struct{
int partNo;//部件编号,从 0 开始
void (* FuncPtr)(void);//函数指针
int callPrd; //调度周期
int loopCnt; //调度计数
int toBeTested; //是否被测对象
}PartVesselType;
```

测试平台通过部件接入容器中的函数指针分量对被测试的软件部件入口模块进行间接调用,实现了测试平台与被测对象之间的解耦合. 所有软件部件分别占用部件接入容器的不同单元,容器的有效单元个数作为测试平台框架接口的一部分可配置. 部件接入容器示意图如图 1 所示.

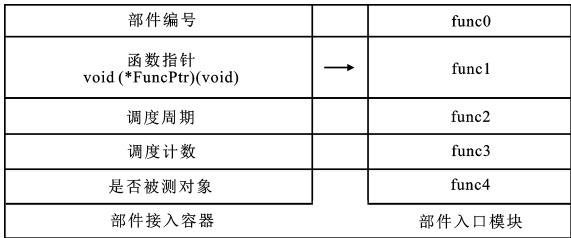


图 1 部件接入容器示意图

2.2 混合类型数据容器

为了兼容不同软件输入、输出接口数据类型的多样性,解决软件测试用例输入、测试结果输出中多种数据类型随机出现的问题,本文设计了一种特殊的“混合类型数据容器”,作为缓存用例文件中的输入数据和测试结果输出数据的基础数据类型. 混合类型数据容器中包含多种标准类型的分量,各分量表示方式和有效长度均可能不同,但是所有分量共享存储空间;每次访问该容器时,只有一个分量有效,如图 2 所示.

字符型c	%c
整型i	%d
短整型h	%hd
长整型l	%ld
浮点型f	%f
双精度类型d	%lf

图 2 混合类型数据容器示意图(以 C 语言为例)

混合类型数据容器主要包括两部分:

(1)数据类型代号

为了按照正确的数据类型来解析测试输入和输出,实时记录当前数据所匹配的数据类型,为不同数据类型自动设置适用的格式化输入、输出字符串(如 C 语言中字符型的“%c”、整型的“%d”、长整型的“%ld”、短整型的“%hd”、浮点型的“%f”、双精度型的“%lf”),数据类型代号定义为如下枚举类型:

```
typedefenum{
LONGTYPE=0,           // %ld
INTTYPE=1,            // %d
SHORTTYPE=2,          // %hd
CHARTYPE=3,           // %c
FLOATTTYPE=4,         // %f
DOUBLETTYPE=5         // %lf
}EnumTypeNum; //识别不同数据类型的代号
```

(2)实际数据类型

混合类型数据容器的实际数据类型定义为如下共用体类型:

```
typedef union{
long          longIO;
in            tintIO;
short         shortIO;
char          charIO;
float         floatIO;
double        doubleIO;
}TypeUnion; //混合数据类型
```

与之相应的混合指针类型定义为如下共用体类型:

```
typedef union{
long      * longIO;
int       * intIO;
short     * shortIO;
char      * charIO;
float     * floatIO;
double    * doubleIO;
}TypeUnionPtr;//混合指针类型
```

根据数据类型代号和实际数据类型指针的定义,混合类型数据容器定义为如下结构体:

```
typedef struct {
EnumTypeNumt   ypeId;//数据类型代号
TypeUnionPtr   realType; //数据类型指针
}MixType;
```

2.3 部件接口访问容器

为了访问各软件部件输入、输出外部接口,为每个被测试部件分配“部件接口访问容器”.该容器包括部件编号、部件外部输入接口总数、混合类型数据容器部件输入接口索引集(数据指针数组)、部件外部输出数据总数、混合类型数据容器部件输出接口索引集(数据指针数组)等分量,设每个部件输入、输出接口均不多于 MAXIO 个,以结构体形式定义如下:

```
typedef struct{
int partNo; //部件编号
int NoIns; //部件外部输入接口总数
//混合类型数据容器部件输入接口索引集
MixType ins[MAXIO];
int NoOuts; //部件外部输出接口总数
//混合类型数据容器部件输出接口索引集
MixType outs[MAXIO];
}PartInterfaceType;
```

通过混合类型数据容器式部件输入、输出接口索引集,静态关联软件各部件的输入、输出接口(以混合类型数据容器表示),以便间接访问部件输入、输出接口,建立输入与输出数据之间的时序同步关系.部件接口访问容器格式如图 3 所示.

为了实时记录测试结果,与部件接口访问容器相对应,为每个被测试软件部件构建“部件测试记录结构体”,以便实时缓存测试结果,建立输入与输出数据之间的时序同步关系.缓冲区长度定义为常量 TSTBUFLLEN,结构体定义如下:

```
typedef struct{
int partNo; //部件编号
```

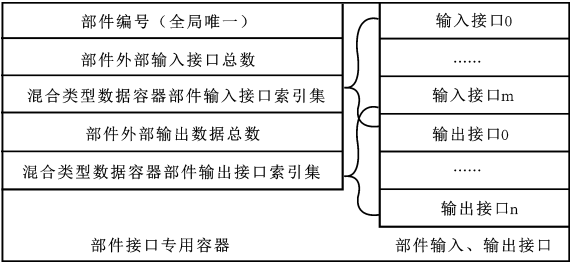


图 3 部件接口访问容器示意图

```
int recNo; //测试记录总数
//为每个部件分配测试输出缓冲区
struct {
//混合类型数据容器式部件输入接口
TypeUnioninputs[MAXIO];
//混合类型数据容器式部件输出接口
TypeUnionoutputs[MAXIO];
}testBuf[TSTBUFLLEN];
}PartRecordType;
```

3 方案通用性分析

3.1 核心部件的跨软件复用

将被调用的软件部件使用测试平台自定义的部件接入容器中的函数指针 * FuncPtr 进行封装,平台对软件部件的调用操作通过部件接入容器中的 * FuncPtr 间接进行,确保不对软件部件直接操作,实现了测试平台与被测试软件部件对象之间的解耦合.所有软件部件按照部件接入容器中的“调度周期”callPrd 分量来分别控制其运行频率.

3.2 测试用例文件中混合类型数据的解析和缓存

所有测试输入以用例文本文件形式注入,在用例文件中指定一行来定义每列数据的标准类型(如 long、int、short、char、float、double 等).用例文件解析时,设计了对混合数据类型输入的支持,为用例文件中不同类型的测试激励动态匹配格式化输入字符串(如 C 语言以格式化输入字符串“%c”匹配字符型等).根据用例文件为每列数据定义的标准类型,以等价的数据类型代号进而自动匹配其格式化输入字符串,同时将各列数据暂存到混合类型数据容器式输入缓冲区类型兼容的存储分量中,解决了混合类型测试数据的注入问题.

3.3 测试输入与输出数据的同步

通过为每个被测试软件部件分配“部件接口访问容器”,在测试平台初始化过程中利用数据指针,将各部件的输入、输出接口与部件接口访问容器中的部件输入、输出接口索引集(测试结果输出缓冲

区)一次性地建立静态关联。

当被测软件部件被周期性调度运行时,通过部件接口访问容器中的部件输入、输出接口索引集 * ins[]、* outs[],就可以间接操作各部件的真实输入、输出接口(来源于软件架构设计的结果,含开环系统和闭环反馈系统的输入、输出接口);将该部件所有输入接口当前值与计算产生的输出一起,动态匹配“部件测试记录结构体”中混合类型数据容器式部件输入、输出接口 inputs[]、outputs[]中相应数据类型分量,实现测试输入、输出的时间同步,解除测试平台与被测软件部件真实外部接口的耦合关系。

4 测试平台方案概述

软件全数字仿真测试平台由测试用例输入数据(A)、测试平台本体(B)和测试结果(C)三大部分组成。测试平台本体包括软件测试平台框架和被测试软件部件(测试对象);如为闭环仿真系统,被控对象模型视为一个软件部件进行调度,但不作为测试对象。被测试软件部件和被控对象模型通过“测试平台与软件部件的接口”与软件测试平台框架建立关联,是软件测试平台框架处理的对象。

软件全数字仿真测试平台框架与测试对象和被控对象模型紧密耦合在一个软件解决方案中(如. sln/. prj),通过编译、链接形成完整的软件全数字仿真测试平台,软件全数字仿真测试平台各功能模块如图 4 所示。

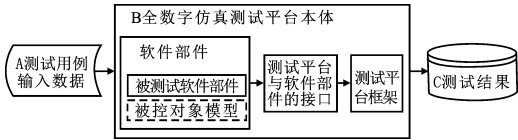


图 4 软件全数字仿真测试平台实现方案

为了方便软件实现,将软件测试平台框架分为初始化模块和调度控制模块两大部分。

(1)初始化模块是可变模块,与被测试软件之间紧密耦合,直接调用各软件部件初始化模块,确保执行部件测试前各部件处于确定的状态。为了与被测软件本体进行适应性配置,需要将用例文件(及闭环反馈)的输入数据与软件部件外部输入接口进行关联,将各部件的输入、输出接口与测试结果输出缓冲区建立链接,并将软件部件纳入“部件接入容器”中,使得周期性调度控制模块能够通过该容器实现对软件部件的间接调度。

(2)周期性调度控制模块是固定模块,与被测试软件之间无耦合,是本方案的基础和核心,可从一个软件测试平台直接移植到另一个软件部件测试平台,实现整体复用。

基于文献[6]中的软件部件测试平台主体框架流程图,本文软件测试平台框架可变模块和固定模块的关系、及各子模块的组成如图 5 所示。

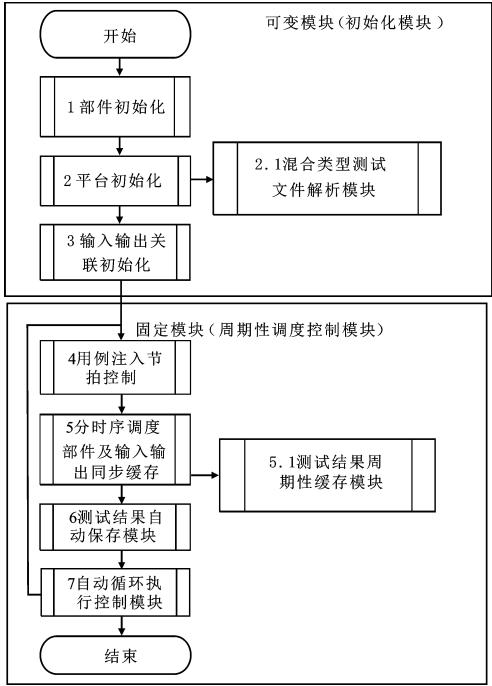


图 5 初始化模块和周期性调度控制模块示意图

5 实例分析

某型号航空发动机燃油控制系统软件主要包括 TBV 伺服回路位置和喘振告警处理两大功能,分为两大部件,各部件输入、输出接口全部通过全局变量进行传递。“部件接入容器”数组定义为 PartVessel-Type vessel[2],“部件接口访问结构体”(部件接口访问容器)数组定义为 PartInterfaceType partIOs [2],TBV 控制部件“部件测试记录结构体”数组(测试记录缓冲区)定义为 PartRecordType testData [2]。

TBV 控制部件(部件 0)入口模块为 void tbvCtrl(void),运行周期为 5 ms,根据发动机状态 EngineState,改变开度(),以提高发动机启动和加减速过程中高压压气机的喘振裕度。

该部件输入、输出接口及其关系如表 1 所示。

partIOs[0]元素各分量为 TBV 控制部件接口访问容器,内容如表 2 所示。

表 1 TBV 控制部件输入、输出接口表					
接口类型	接口名称	数据类型	输入输出关系		
输入	EngineState	int	1-起动	2-加速	3-减速
输出	Ltbv	double	0.5	1.0	0.1

表 2 TBV 控制部件接口访问容器内容		
分量名称	分量标识	值
部件编号	partNo	0
输入接口总数	NoIns	1
输入接口索引集	typeId	INTTYPE
	realType, intIO	&EngineState
输出接口总数	NoOuts	1
输出接口索引集	typeId	DOUBLETTYPE
	realType, doubleIO	&Ltbv

喘振告警部件(部件 2)入口模块为 void surgeAlarm(void),运行周期为 20 ms. 通过压力比信号 A 值设置喘振告警信号有效(接通)或无效(断开):当 A 值大于等于阈值 0.5 时,测试接口设备软件应置喘振告警信号有效(接通);当 A 值小于阈值 0.5 时,测试接口设备软件应置喘振告警信号无效(断开).

输入、输出接口如表 3 所示.

表 3 喘振告警控制部件输入、输出接口表				
接口类型	接口名称	数据类型	输入输出关系	
输入	A_value	float	≥0.5	<0.5
输出	surge	int	1—有效	0—无效

partIOs[1]元素各分量为喘振告警部件接口访问容器,内容如表 4 所示.

表 4 喘振告警部件接口访问容器内容		
分量名称	分量标识	值
部件编号	partNo	1
输入接口总数	NoIns	1
输入接口索引集	typeId	FLOATTYPE
	realType, floatIO	&A_val
输出接口总数	NoOuts	1
输出接口索引集	typeId	INTTYPE
	realType, intIO	&surge

如果针对 TBV 控制部件和喘振告警部件同时进行测试,vessel[2]数组元素各分量如表 5 所示.

表 5 vessel[2]数组元素各分量内容			
分量名称	分量标识	vessel[0]	vessel[1]
部件编号	partNo	0	1
函数指针	* FuncPtr	tbvCtrl()	surgeAlarm()
调度周期	callPrd	5 ms	20 ms
调度计数	loopCnt	1	4
是否被测对象	toBeTested	1	1

软件集成测试用例文件名为 testFile.csv,时序节拍控制数值的量纲为 5 ms,用例文件格式及测试数据定义如表 6 所示.

表 6 软件集成测试用例文件表头内容		
Time	EngineState	A_value
int	int	float
0	1	0.3f
2	2	0.6f
3	3	0.1f
5	3	0.2f

测试记录缓冲区 testData[0]元素为 TBV 控制部件测试记录结构体,缓冲区内内容组成及测试结果如表 7 所示(其中 inputs[0]. intIO= partIOs[0]. ins[0]. realType, intIO、outputs[0]. doubleIO= partIOs[0]. outs[0]. realType, doubleIO):

表 7 TBV 控制部件测试记录结构体内容			
分量名称	部件编号	测试输出缓冲区	
		输入接口	输出接口
分量标识	partNo	inputs[0]. intIO	outputs[0]. doubleIO
值	0	1	0.5
		1	0.5
		2	1
		3	1
		3	1
		3	0.1
		3	0.1
		3	0.1

testData[1]元素为喘振告警部件测试记录结构体,缓冲区内内容组成及测试结果如表 8 所示.(其中 inputs[0]. floatIO= partIOs[1]. ins[0]. realType, floatIO、outputs[0]. intIO= partIOs[1]. outs [0]. realType, intIO).

由于 TBV 控制部件每个调度周期运行一次,最终维持第 4 个测试数据 3 的数值;喘振告警部件每 4 个调度周期运行一次,因此第 2 个测试数据

0.6f在未处理前即被第3个测试数据0.1f覆盖,自第3个调度周期以后,始终维持第4个测试数据0.2f的数值.根据以上测试记录缓冲区的内容,软件集成测试结果正确.

表 8 喘振告警部件测试记录结构体内容

分量名称	部件编号	测试输出缓冲区	
		输入接口	输出接口
分量标识	partNo	inputs[0].floatIO	outputs[0].intIO
值	1	0.3f	0
		0.1f	0
		0.2f	0
		0.2f	0

6 结束语

本文提供了一种可移植的全数字仿真软件集成测试平台方案,在保持“周期性调度控制模块”完全不变的前提下,只要修改初始化模块,并与被测试部件一起创建项目工程(.sln/.prj),就可以方便地生成软件全数字仿真集成测试平台,实现了测试平台构建方案的复用.如果被测试软件升级,只要其接口不变,就可以覆盖被测试部件,重新编译生成新版本软件测试平台.即使被测对象接口改变或换成其他测试对象,也可以通过替换被测试软件、重新配置初始化模块,快速生成新的软件测试平台.

另外,本文支持以各种标准数据类型分别定义不同的测试输入,在向被测试软件注入激励时保持相同的数值精度进行数据类型匹配和解析,确保了测试结果的准确性和有效性.

最后,本文为每个被测试软件部件分配了专用的输入、输出接口“容器”,只需操作这些“容器”,就可以间接作用于软件部件的真实接口,实现了测试平台与被测软件部件外部接口的解耦合,同时保证

了测试输入和输出的同步,方便判断测试结果的正确性,同时能够保存、积累原始测试数据,以支持软件架构和算法的持续优化.后续研究中,将继续探索将周期性调度模块封装在静态、动态链接库的设计方案,以更加灵活的形式实现软件集成测试平台的跨软件复用.

参考文献:

[1] 徐遵磊,张永强,张哲,等.暗物质粒子探测卫星有效载荷综合测试系统的设计与实现[J].天文学报,2018,59(3):39-49.

[2] 丁芳颐,刘江.运载火箭通用发射软件测试平台设计[J].现代防御技术,2015,43(4):178-183.

[3] 魏许,张怡文.飞轮通用测试软件平台的设计与实现[J].计算机测量与控制,2017,25(12):129-132.

[4] 郭旺,丁晓明,唐海鹏,等.半实物环境下嵌入式软件通用测试平台研究[J].西南大学学报(自然科学版),2015,37(4):62-66.

[5] 李文波,周志飞.嵌入式软件通用仿真测试平台研究[J].机车电传动,2017,57(3):105-109.

[6] 王影,刘卉,赵娟.软件部件仿真测试平台的设计与实现[J].计算机工程与设计,2017,38(11):3061-3065.

[7] 中航商用航空发动机有限责任公司.软件集成测试方法及平台:中国,201710860810.8[P].2017-10-19.

[8] 王影.软件集成测试平台的研究与设计[J].计算机工程与设计,2018,39(8):2675-2681.

[9] 郭素敏,徐克宝,吴波,等.基于构件技术的通用自动测试系统软件设计[J].计算机测量与控制,2013,21(10):2859-2861.

作者简介:

王影女,(1974-),硕士,高级工程师.研究方向为软件测试、程序切片、机载软件适航标准 DO-178C.
E-mail:wyltm@126.com