旋极科技
WATERTEK

旋极视界
WATERTEK VISION

北京旋极信息技术股份有限公司主办

软件工程专刊

软件安全　标准为先

P21　我的探索式软件测试实践

P40　FPGA验证中静态时序分析的运用

P63　SPARC V8目标码测试的实现方法

P77　适应ISO 26262安全标准的软件验证

2013年

该画是新墨西哥州一位知名画家画的当地的山
是合作伙伴Great River Technology公司赠送给旋极的新年礼物

软件安全 标准为先

北京旋极信息技术有限公司军工中心副总经理 岳庆敏

在当今的社会生活中，软件无处不在，在很多系统或者设备中，软件是否正常工作会影响到人类的生命安全，例如，航空电子设备、铁路信号控制系统、汽车电子、医疗设备等，这类软件的可靠性和质量至关重要，这类软件称之为安全关键软件（Safety Critical Software）。

“安全”是安全关键软件生命周期的核心焦点，这里的安全是指Safety而不是Security。虽然软件工程指导人们以工程化方法构建和维护有效的、实用的和高质量的软件，但是相对于人类掌握的其他技术，软件工程还很年轻，还处在初级阶段。而安全关键软件由于其对安全的高度关注，需要采用更加严格的软件工程方法才能满足要求。

各个安全关键行业针对自身行业的软件特点，都制定了安全相关的标准，例如，航空的机载软件安全标准DO-178B、铁路的EN 50128、汽车ISO 26262和MISRA、医疗设备ISO 62304。作为软件安全标准，具备强制性和约束力，为了达到安全的目标，标准要求必须要建立适当的流程和规范，根据软件失效的后果对软件的安全性进行定级，例如DO-178B把机载软件分为A-E五级，每一级的软件分别需要完成不同的活动以满足对应的目标。安全关键软件还需经过认证才可以运行在系统中，例如机载的适航认证，就是要有充分的证据表明机载软件的开发和验证完全遵循了其流程和计划，通过多次审查，确保最终的软件是安全的。

追求安全，意味着需要付出更大的代价，包括工作量、费用和周期，以DO-178B所要求的航空机载软件为例，软件的适航认证对软件研制单位而言是一个巨大的挑战，与传统的开发过程相比，要达到适航要求的软件开发所投入的费用和工作量是原来的2-5倍。

在安全关键软件的规范体系和研发流程方面，我国的相关行业起步较晚，但是当我们面临全球化的市场时，需要遵循国际上统一的软件安全标准，因为这些安全标准的认证已经形成了一种准入制度，只有通过了认证，软件才可以作为设备的组成部分正式上市并进行商业运行。

作为专注在嵌入式软件行业的专业公司，旋极信息一直致力于引进先进的软件工程化思想和技术，为国内的安全关键软件行业提供先进的平台与技术服务，重点关注航空航天、铁路、汽车、医疗等安全关键软件行业。我们通过自身的技术积累，开展多方合作，学习先进经验，为客户提供软件工程咨询、平台打造和技术服务。目前已经为多家国产商用飞机设备提供商提供DO-178B机载软件适航咨询和适航工具平台的构建服务。随着我国高速铁路和汽车电子等行业的飞速发展，我们也将会提供EN 50128和ISO 26262等认证咨询服务。

专注安全关键软件领域，在软件开发过程管理、软件验证与认证方面提供服务是我们现在和未来重要的发展方向，旋极信息期待与广大客户、合作伙伴和行业同仁们，一起共铸中国软件工程的美好明天。

《旋极视界》软件工程专刊

合作伙伴

软件开发管理类

IBM Rational专门提供面向软件开发与交付的产品、服务和最佳实践；并且覆盖了从设计到交付的整个项目生命周期，包括需求定义、设计与开发、变更与发布、质量管理、安全与循规管理等；旨在帮助企业提升软件交付的创新能力和投资收益。

Geensoft公司开发并销售适用于嵌入式或关键系统的工程工具。明星产品为全生命周期需求追踪工具Reqtify和多处理器/多总线系统实时体系的建模和验证工具RT-Builder。

Ellidiss软件公司是欧洲TNI集团子公司，总部在英国，法国设有研发中心，Ellidiss是实时嵌入式系统架构分析与设计解决方案的领先供应商，在国际航空，航 天，交通，国防和学术界有着广泛的用户。Ellidiss参与了AADL的标准起草，方法的推广。 并开发了世界上第一个支持AADL的商业工具，并同时支持UML 2.0，HRT-HOOD，需求分析和软件原型方法方面的技术。该公司与国际标准委员会保持紧密的关系，积极参与国际知名组织的研发项目，例如欧洲航天局，Ada语言组织 – SIGADA和Ada欧洲保持合作，并且与全球主要航空航天研发机构合作。

总线协议测试类

GRT公司提供在ARINC 818（FC-AV）和HOTLink Video领域领先的ARINC 818全套开发测试产品，同时为客户提供最全面和高品质的技术支持和销售服务。

TTTech公司是基于时间触发（Time-Triggered）技术和模块化安全平台的可靠性网络顶级供应商，能够提供符合Do254、Do178B、IEC61508、ISO26262要求的产品，能够为客户提供最全面的和高品质TTP（Time-Triggered Protocol）和TTE（Time-Triggered Ethernet）全套开发测试产品，提供最优质的技术支持和销售服务。

ICS公司为客户提供最全面的和高品质ARINC 825开发测试产品。ARINC 825是一个将CAN协议应用到机载环境下的通用标准。

瑞典Kvaser公司专著制造生产CAN总线测试产品的专业产家，是汽车/坦克领域、工业自动控制领域和医疗设备领域等工程师首先的测试工具和产品。系列接口产品有USB、PCMICA、PCI和P104+。

软件测试类

Verocel公司专门为安全关键软件领域中软件验证提供专业技术和服务。服务包括开发、评审软件计划和标准、软件需求和测试开发、软件结构覆盖率分析、生命周期数据可追踪性，以及外包支持。

Programming Research公司是专业从事软件设计方法学和软件编程规范研究的公司，是MISRA的主要起草者。

QA Systems公司致力于为科技开发中各个方面提供广泛的商业和技术服务。明星产品为C/C++单元集成测试工具和Ada语言单元测试工具。其在航空领域已取得广泛应用。

McCabe软件公司是由McCabe圈复杂度提出者Thomas McCabe先生创立发展的。McCabe IQ软件质量管理解决方案，为用户提供软件质量度量、软件结构分析、动态结构化测试的全面支持。世界上很多重要组织和公司采用McCabe IQ的质量管理套件，对其开发的关键软件进行质量分析和覆盖率测试。

Critical公司努力满足对在网络社会中不断增值的复杂任务和商业关键系统的验证和确认需求。采用Xception™故障注入技术，无论基础设计或COTS部分都可以在接近真实的条件下，通过最小侵入方式和可靠的容错机制确认，最终得到完美的验证。

Bender-RBT公司在基于需求测试领域是世界领先的专家。RBT过程首先需要保证规格说明是正确的、完整的、清晰的和逻辑一致的。从黑盒测试角度，一个完整的测试用例集要确保所有需求都被设计和代码所覆盖。RBT过程可适用于任何应用，不限语言种类，可在嵌入式系统、PC、客户机/服务器、基于网络、主机和基于应用的超级计算机上运行。

系统测试类

TechSAT GmbH公司经过二十五年发展，已成为航空业可靠的合作伙伴和集成与认证专家。其明星产品ADS-2系统测试产品是开放的工业标准商业现成（COTS）方案，用于对分布的实时系统（DRTS）进行开发、集成、测试等工作。

卷首语

01 软件安全 标准为先

资讯

06 旋极新闻

07 市场活动

GJB5000A认证与过程改进

10 实现GJB 5000A要求的需求管理和追踪平台研究

14 GJB 5000A三级中的软件测试过程改进

16 CMM在软件测试中的应用

软件测试新技术

21 我的探索式软件测试实践

23 嵌入式软件测试浅谈

25 线性独立路径覆盖测试用例自动生成方法研究

30 Linux 安全性测试要点浅谈

FPGA测试技术

33 武器装备可编程逻辑器件应用现状分析

37 型号FPGA应用现状、发展趋势及建议

40 FPGA验证中静态时序分析的运用

43 VMM搭建FPGA自动验证环境

46 动态仿真中代码覆盖率和功能覆盖率

49 提高FPGA复位设计可靠性的建议

适航验证技术

54 DO-178C的新变化和对机载软件认证的影响

57 通过基于目标（Objective-based）的标准获得航电软件的安全性认证

63 SPARC V8目标码测试的实现方法

67 DO-254规定的硬件生命周期数据要求探讨

69 基于ADS2的数控系统控制软件系统测试环境搭建与应用

行业软件安全标准

74 解析铁路软件安全标准EN50128与软件工具的支持
77 适应ISO 26262安全标准的软件验证

软件系统体系架构

82 基于AADL模型驱动架构设计法
——优化飞机综合实时仿真系统
88 AADL和Simulink的关系

产品应用

92 RBT在软件黑盒测试用例设计中的应用
95 需求追踪工具化对机载软件研制的影响
99 基于CANTATA++的软件单元测试
103 McCabe IQ在白盒测试中的应用体会
105 AdaTest95在型号控制软件中的应用

旋极人

107 软件工程技术专家
110 2012骄傲的事TOP10

2013年软件工程专刊 总第12期
主办：北京旋极信息技术股份有限公司
股票简称：旋极信息 代码：300324

投稿电话：010-82883939转637
投稿信箱：news@watertek.com
订阅电话：010-82883933转637
订阅邮箱：news@watertek.com
公司网址：www.watertek.com
地　　址：北京市北四环中路229号海泰大厦1006室
邮　　编：100083

（本杂志免费邮寄，邮件中请注明：姓名，单位，职务，地址，邮编，电子邮箱，电话）

旋极新闻

旋极信息当选北京软件行业协会第七届理事会副会长单位

2012年10月31日下午，北京软件行业协会第六届第二次会员代表大会、第七届理事会第一次会议在北京丽亭华苑酒店隆重召开，市经信委姜贵平副主任、姜广智处长出席会议并作出了重要讲话，会议由协会第六届理事会会长王文京主持召开。

北京软件行业协会由北京市经济和信息化委员会主管，成立于1986年10月21日，是在北京有着26年悠久历史影响和广泛会员基础的软件产业社团组织，目前已为近2000家软件信息服务企业提供价值服务，在北京乃至全国的软件行业有着重要影响力。

这次大会到场的企业200多家，用友、首信、中科软、太极等知名软件企业的领导参加会议并进行了新一届理事会选举。北京华胜天成科技股份有限公司总裁王维航当选为北京软协第七届理事会会长，北京中科希望软件股份有限公司董事长周明陶当选为监事长，龙飞再次被聘为协会秘书长。旋极信息董事长陈江涛先生当选为第七届理事会的副会长，旋极信息荣升为新一届的软协副会长单位。

旋极信息参与“学子阳光”慈善活动

2012年11月2日，由北京旋极信息技术股份有限公司赞助，共青团北京市委、北京青少年发展基金会、北京市学生联合会开展的“100365首善行动——‘学子阳光’首都大学生励志奖学行动仪式”在奥运大厦举行。北京青少年发展基金会秘书长陈淑惠、北京青少年发展基金会副秘书长张建华、团市委大学部副部长佟立成、旋极信息战略规划部总监阮亚占、公关企划部总监彭季出席了本次颁奖仪式。

此次“100365首善行动——‘学子阳光’首都大学生励志奖学行动”旨在奖励首都高校在社会公益领域有突出贡献的大学生，其目的在于培养首都青年学生的社会责任感，积极引导首都青年追求理想，努力进步，为营造和谐校园氛围和建设节约型社会贡献力量。

“我们把公益事业当作自己的信仰”，一位受奖励的大学生表示，“这次获奖对我来说是莫大的鼓舞和支持，我们立志在未来为中国公益事业的发展贡献自己更大的力量”。

北京青少年发展基金会秘书长陈淑惠向旋极信息颁发了荣誉证书，并感谢了对旋极信息为我国公益事业作出的贡献，最终实现公益理念和行为的真正传播与爱的传递。

和“阳光学子”们的合影

扶贫助学 重教情深

2012年11月29日中午，在河北省怀安县怀安城镇第一小学的操场上，举办了一个简单却温馨的捐助仪式，此次捐助是旋极信息向河北省怀安县的三所贫困小学和县委党校捐赠了千余套教学课桌椅。

仪式由怀安县县委副书记屈淳主持，怀安县政府和县教育局数位领导参加，旋极信息总经理助理赵廷荣、公关企划部总监彭季代表旋极信息参加仪式，怀安城镇第一小学的三百多名师生见证了这次捐赠仪式。

捐赠仪式中，总经理助理赵廷荣代表旋极信息发表讲话。他深切地表达了旋极信息作为一家上市公司，有很强烈的社会责任感；回馈社会，关注教育，尤其是帮助贫困地区的孩子们，是旋极信息和每个旋极人义不容辞的责任。同时，赵总深情地告诉孩子们：帮助别人的人也会获得很大的快乐，希望孩子们把这份爱心传递下

去。

最后，怀安县的诸位领导向旋极信息颁发了锦旗和铸有“扶贫助学 重教情深”的铭牌，以感谢旋极信息对贫困地区教育做出的贡献和善举。

县教育局长授予锦旗

焕然一新的课桌椅

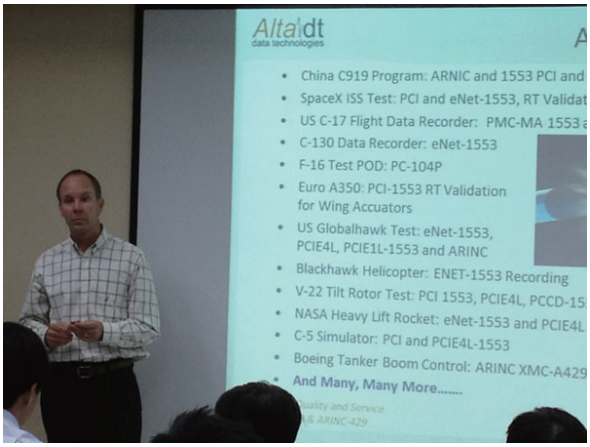
市场活动

Alta Data Technologies公司总裁Rick Schuh在沪进行总线产品技术交流

美国Alta Data Technologies公司总裁Rick Schuh先生受旋极信息邀请，于2012年10月30日在上海旋极与国内航空航天技术精英就“MIL-STD-1553B&ARINC 429总线产品”进行了面对面的技术交流。

Rick Schuh先生拥有30年航空电子系统测试经验，在本次技术交流中，他详细介绍和展示了其公司最新的MIL-STD-1553B&ARINC 429总线产品，并与到会的客户分享其在航空总线系统测试中的经验，帮助用户深入了解航空1553B、429总线的开发与测试技术，同时介绍了最前沿的测试和仿真技术。

会上，关注航空1553B、429总线开发与测试的航空航天技术精英与Rick Schuh先生进行了广泛的交流与探讨，取得了良好的反响。



Alta Data Technologies总裁做主题演讲

Verocel与旋极信息共同亮相风河2012中国开发者大会

全球领先的嵌入式和移动软件提供商风河公司于11月5至9日在北京、上海、深圳等国内重要城市巡回举行“Wind River 2012中国开发者大会”。

本届会议紧扣“全面智能化”这个全球性主题，为国内嵌入式系统开发者带来全球最新的趋势和技术，包括智能系统和物联网、智能网络、安卓系统、产业发展动态以及风河的最新技术。

本次大会设置了四个主题的现场分会场，主题分别涵盖：智能系统和物联网、智能网络、航空航天和国防以及安卓系统。

安全关键软件验证专业公司Verocel是风河公司的重要合作伙伴，长期为VxWorks操作系统开展DO-178B和IEC 61508等安全关键认证，作为本次大会的金牌赞助商，Verocel总裁George Romanski专程来华参加此次大会，并作为演讲嘉宾就“航空航天和工业领域

安全关键软件”为主题发表演讲，同时Verocel公司本次也在大会展示了全新升级的全生命周期追踪管理工具Verotrace以及其它软件验证工具，旋极信息作为Verocel公司在中国的合作伙伴共同参加了此次大会，与来自各种嵌入式行业的风河用户进行了充分的交流和互动。

此外，Intel、Freescale、Cavium Networks、Broadcom、Presagis、TTTech、Adlink等全球知名合作伙伴也做了精彩演示，让与会者大开眼界。

George Romanski为与会者讲解全新的软件验证工具

George Romanski发布主题演讲

航天科工集团第六届软件评测机构工作总结暨学术交流会

为全面总结中国航天科工集团公司软件评测机构2011-2012年度的各项工作，进一步提高软件评测机构管理和技术水平，由中国航天科工二院706所主办的“航天科工集团第六届软件评测机构工作总结暨学术交流会”于2012年11月1日在北京怀柔成功举办。作为航天科工软件评测单位的长期合作伙伴，旋极信息参加了此次大会。

2012年11月1日上午9点30分会议正式开始，由集团领导致开幕词。会议总结了2011年度航天科工集团各软件评测机构的各项工作，并由各软件评测机构的优秀论文作者代表发言。

2012年11月2日进行分组技术学术交流，旋极信息的软件测试专家出席学术交流会，听取各个测试同行的主题报告并积极参与讨论。旋极信息资深软件测试专家王鹏在下午的主题报告环节，以软件测试新技术为主题，为航天软件评测工作者分别介绍了系统仿真与故障注入测试、Baseline在单元测试中的应用与嵌入式软件目标码测试技术，并结合航天型号特点，分析了以上三种新技术在软件评测中的具体应用方式以及带来的效果。

GJB 5000A认证与过程改进

P10 实现GJB 5000A要求的需求管理和追踪平台研究

P14 GJB 5000A三级中的软件测试过程改进

P16 CMM在软件测试中的应用

实现GJB 5000A要求的需求管理和追踪平台研究

■ 文 | 上海旋极技术部软件工程化产品经理 闵蓓尔

摘要：针对GJB5000A军用软件研制能力成熟度模型中需求管理的过程域的要求，结合型号软件研制现状，对软件需求管理的相关技术和手段进行了分析，提供一个软件需求管理和追踪平台方案。供参考。

关键词：需求管理、需求追踪

1 引言

GJB 5000A在需求管理ReqM过程域中明确了需求管理的目标，即管理项目的产品和产品部件的需求，标识它们与项目的计划和工作产品之间的不一致性，并针对目标给出建议性的专用实践。

在型号软件研制过程中规定，与需求相关的活动有系统需求分析和软件需求分析，也规定了在系统需求分析和软件需求分析中需要进行的工作，包括需求设计方法和需求评审。目前在型号软件的需求开发过程中严格按照规定方法实施编制和设计，但是由于需求经常变更，怎样实现GJB5000A中关于需求管理的目标，保持需求和其他工作产品的一致性，是一个急需解决的问题。

在型号软件研制过程中，按照GJB5000A的要求，需求追踪是软件项目中一项十分重要的工作。据国外调查显示在众多失败的软件项目中，由于需求原因导致的故障约占到45%，因此，需求追踪将对软件项目能否最终实现产生至关重要的影响。

2 需求管理

在GJB5000A中将组织的软件研制能力成熟度分为5个等级，1级初始级，2级已管理级，3级已定义级，4级已定量管理级，5级优化级。军用软件研制能力成熟度模型见图1。

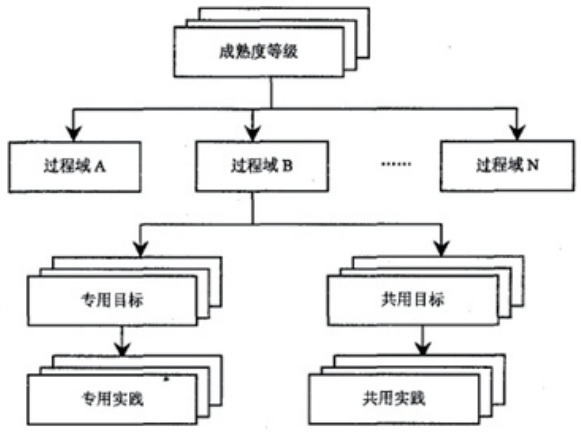


图1 军用软件研制能力成熟度模型

在每一个成熟度等级中包含多个过程域，每个过程域中包含专用目标和专用实践，共用目标和共用实践。GJB5000A规定在2级已管理级中具有工程类型的过程域需求管理（ReqM），与需求管理相关的专用目标和专用实践见表1。

成熟度等级	过程域（PA）	专用目标（SG）	专用实践（SP）	典型工作产品
等级2已管理级	需求管理 ReqM	专用目标1 管理需求	专用实践1.1 获得对需求的理解	a) 辨别合适的需求提供者的准则 b) 需求的评价和验收准则 c) 对照准则分析需求结果 d) 达成一致的需求集
			专用实践1.2 获得对需求的承诺	a) 需求影响评估 b) 对需求和需求更改的承诺的记录
			专用实践1.3 管理需求更改	a) 需求状态 b) 需求数据库 c) 需求决策数据库
			专用实践1.4 维护需求的双向可追踪性	a) 需求可追溯性矩阵 b) 需求跟踪系统
			专用实践1.5 标识项目工作与需求之间的不一致性	a) 记录不一致的文档，包括不一致的来源、条件和理由 b) 纠正措施

表1 与需求管理相关的专用目标和专用实践

需求管理过程域针对目标给出建议性的专用实践总结如下：

1） 要与需求提供者一起理解需求的含义；

2） 获得项目参与者对需求的承诺；

3） 当需求在项目期间演化时，管理需求的更改；

4） 维护需求和工作产品之间的双向可追溯性；

5） 标识项目计划和工作产品与需求之间的不一致性；

6） 要提供管理需求、需求开发的环境；

7） 将需求管理过程置于合适等级的控制之下；

8） 能与更高层管理者一起评审需求管理过程的活动。

3 需求管理和追踪平台实施

基于GJB5000A的要求和型号管理体系，我们确立了需求管理系统的整体结构和流程，如图2。

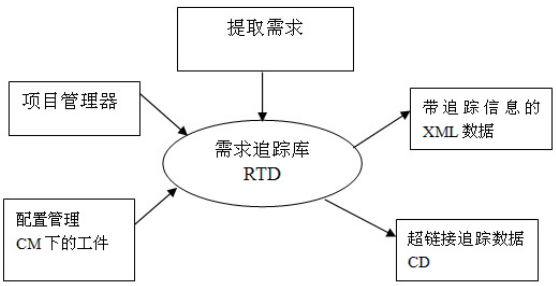


图2 需求管理系统的整体结构和流程

直接从Microsoft Word、Microsoft Excel、XML文件中提取需求导入到需求追踪数据库（RTD）。

项目管理器提供全程的项目设置控制，包括需求编号标识、配置管理系统、开发模式和评审状态转换方式等，并且只能被具有管理权限的用户更改。

需求可追踪性数据库（RTD）来存储和维护需求和支持需求工作产品的名称以及链接关系。这些工作产品包括源代码、设计组件、测试程序、功能测试的结果、覆盖率结果分析、评审状态、评审检查表。工作产品存储在用户的配置管理（CM）系统或操作系统的文件系统中，工作产品的名称及链接关系存储在RTD中。RTD与CM系统或文件系统进行交互，跟踪工作产品的可追溯性。

RTD首先生成XML数据形式的追踪信息，然后使用XSL STYLESHEET将所有的XML数据转换成HTML文件，这样通过任何浏览器都能够查看追踪过程。追踪数据也可以保存到CD光盘上，以便于验证。

3 需求管理相关技术

需求管理系统建立统一的需求追踪模型，记录系统开发需求、软件实现需求、设计、测试用例测试脚本、测试结果、评审检查单等资产，并进行追踪分析；配置管理系统收集开发文档和代码，进行系统基线、评审状态的管理。

3.1 建立需求标识和组织结构

建立需求开发层次，需求由开发需求、实现需求、源代码文件、功能函数组成。开发需求和实现需求可以建立高级需求和低级需求的层次。开发需求和实现需求的关系如图3。

开发需求是需求集合。包含硬件开发需求和软件\系统开发需求。 在集合之内的结构可以是有等级的或线性的，开发需求在集合之内可以被追踪到开发需求的其他集合。开发需求可以追踪到实现需求。

实现需求等级结构直接地对应于软件实施结构。包括目录级需求，源代码级需求，函数级需求。实现需求可以被链接到开发需求，可以进行源代码和测试覆盖的追踪。

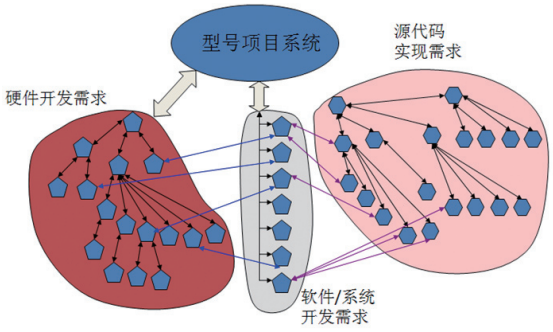


图3 开发需求和实现需求的关系

对每一个需求建立标识，标识与项目和需求层次相关。开发和实现需求标识数字格式的规格，指定字符的数字，指定需求层次数的数字，指定是否有次级需求数字或字符。例如需求标识可设立为aaaa[.bbbb]-nnn，aaaa表示高级需求，.bbbb 表示次级需求，-nnn表示最末级详细需求。

3.2 建立需求和开发工作产品的状态

在开发生命期中对需求和相应工件建立5个状态，他们分别是：

初始状态：还在开发中，可以修改和更新，不予追踪。

准备评审状态：不能修改，开发和更新已经完成，可以在配置管理库中的工作产品上打上基线标记，进行追踪。

失败状态：可以修改，不满足项目需求评审标准，在需求追踪数据库RTD中存储评审时间戳和评审记录，不产生评审检查表，可以在配置管理库中的工作产品上打上基线标记。

通过状态：不能被修改，满足项目需求评审标准，在需求追踪数据库RTD中存储评审时间戳和评审记录，产生评审检查表，检查表可以被签署，检查表可以放入配置管理中。可以在配置管理库中的工作产品上打上基线标记。

已签署状态：不能再修改，项目的生命周期已经完成，如果在配置管理库中存在相应的检查表，在该检查表上打上基线标记。

状态转移见图4。

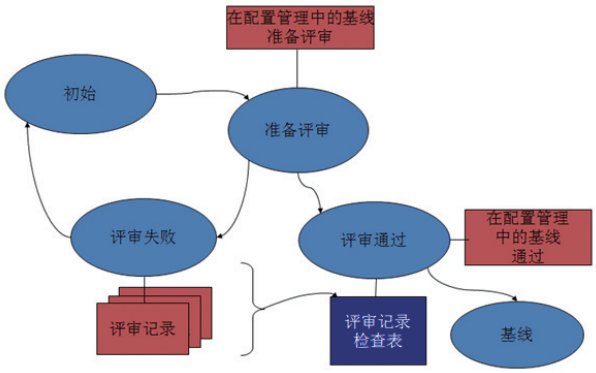


图4 状态转移

基线是建立需求和工作产品的集合，表明这个集合已经完成了开发，并且已经准备好去验证。当它在准备好评审状态时，需求和工作产品进入一个具体版本的基线。

对生命期中需求和相应的工作产品的审计建立一个瀑布模型，规定在需求评审完成后才可以进行设计和测试评审，在设计评审完成之后才可以进行代码评审，在代码和测试评审完成之后才可以进行测试结果评审。

3.3维护需求的双向可追踪性

在需求追踪数据库中建立需求和工作产品之间的链接关系，见图5。

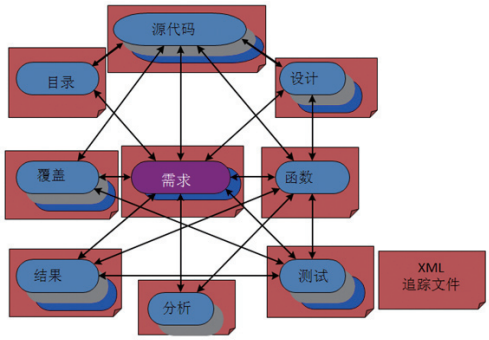


图5 需求追踪数据库中需求和工作产品之间的链接关系

图5中有深蓝色标记的工作产品表示是经过评审，是具有评审检查表的。

执行改变影响分析，当生命期中的工作产品状态变为初始或准备评审时，与之相关的工作产品将变为无效，需要重新评审。将跟踪生命周期的功能结果，包括参与测试、覆盖和结果影响分析。

4 需求管理工具

根据GJB5000A共有目标2和共有实践2.3的提示，可以引入需求管理工具来对型号软件的需求进行更好且更方便的管理，制度化已管理过程。

VeroTrace是一个需求和生命周期可追踪管理的工具，允许产生、管理和发布用于支持DO-178B软件认证/审核的所有数据。它可以追踪系统需求、软件需求，追踪与需求相关的工件，例如：源代码，设计组件，功能测试结果，覆盖率测试结果。

VeroTrace提供一个需求跟踪数据库，在数据库中可以添加、修改、删除、提问，追踪验证状态，执行在线验证，可以从配置管理中提取工件数据，创建验证检查列表，提供在工件和验证证据之间基本的跟踪XML文件。VeroTrace通过自动生成超链接的可追踪性信息，可以组织需求、与需求相关的工件及所有评审文件，到一个链接所有生命周期中可追踪性数据的CD-ROM，以便于验证和审核。

5 结束语

需求管理和追踪平台维护可追溯性需求与所有的软件工作产品。可以在配置管理系统中对软件工作产品设定基线，在对需求和软件构件评审后，可以自动生成审查核对清单。审查清单可以检入到配置管理系统，可以自动生成需求规范文档和可追踪性矩阵，可以产生一个完整的验证所有需求、软件构件、审查文档超链接的可追溯性的包来用于审计。

需求管理系统的目的是帮助型号软件在短期内建立一个需求管理流程，可以提高对项目目标的可视性，从而提高协作能力；帮助履行合同义务并满足规章制度的需求；更有效地响应瞬息万变的客户需求并提高控制能力；推动提供能够满足客户需求的高质量的系统和软件；文档导向的直观界面，支持轻松部署；全面的跟踪能力分析功能，可帮助确保不会漏掉任何需求；主动变更通知功能，可帮助确保不会遗漏变更并且全面分析变更影响，满足了GJB 5000A 需求关键过程域的要求。

参考资料

[1] GJB 5000A-2008军用软件研制能力成熟度模型

[2] RTCA/DO-178B 在空运系统和设备验证中软件需要考虑的问题

[3] GJB 438B-2009 武器系统软件开发文档

GJB 5000A三级中的软件测试过程改进

■ 文 | 旋极信息军工中心测试与控制部软件测试技术经理 任建国

旋极信息推荐的GJB5000A三级软件测试过程，很好的实现了软件测试过程管理和控制，此方案针对军工软件开发单位开展企业级实施，采用分布式方式部署，可以兼顾管理层、质量管理部门、软件开发部门、测试部门、保密办等各级单位的需求，可以结合本单位具体的管理措施和流程进行定制，以充分实现测试过程管理的电子化和自动化。

背景

目前国防系统嵌入式应用日趋复杂，而由于竞争要求产品快速上市，开发技术日新月异，而软件故障却日益突出。同时，由于来自上层的需求经常变化，导致应用系统的版本和变更管理的瓶颈日渐成为开发过程的瓶颈。

为解决军用软件研制过程中出现的问题，促使军用软件承制单位尽快提高其软件过程能力，降低军用软件的风险，中国人民解放军总装备部于2008年3月30日发布了GJB 5000A标准，从2008年6月1日起开始实施。GJB 5000A标准的制定旨在引进国外先进的管理经验，提高我国军用软件的质量，其目前已成为我国武器装备软件建设和发展事实上的标准。总装规定国防单位要承接软件开发项目，必须通过GJB 5000A的质量体系认证，引入软件开发的过程管理。

GJB 5000A标准的目标就是不断改进软件生产工艺，提高产品质量。GJB 5000A标准是在参考了 CMU-SEI组织的SW/CMM V1.2的基础上制定的，不仅给出相关过程成熟度的描述内容，还给出改进模式的指导和评估/评价模式的指导，可以说是一个针对武器装备软件开发团队的过程改进体系。

GJB 5000A为国防单位提供了一个系统的软件开发过程管理框架，在实践过程中还需要具体管理和技术平台的支持，才能更有效的结合软件开发的生命周期和项目管理过程。

GJB 5000A成熟度等级三的11个关键过程域

GJB 5000A成熟度等级三已定义级的重点集中在整个组织软件项目的过程管理上。因为根据多年的经验和教训，人们认识到软件开发面临的首要问题不是技术问题而是管理问题。在成熟度等级三，整个组织的项目已确保其过程按照方针进行策划并得到执行。

在成熟度等级三，由以下11个关键过程域组成：

1）决策分析和决定　　2）集成项目管理

3）组织过程定义　　4）组织过程焦点

5）组织培训　　6）产品集成

7）需求开发　　8）风险管理

9）技术解决方案　　10）确认　　11）验证

与软件测试关系最紧密的两个过程域：确认和验证，实际上是将测试作为关键过程域来管理。其专用目标和专用实践如下：

i. 确认

专用目标1准备确认

专用实践1.1 选择要确认的产品

专用实践1.2 建立确认环境

专用实践1.3 建立确认的规程和准则

专用目标2确认产品或产品部件

专用实践2.1 确认产品或产品部件

专用实践2.2 分析确认结果

ii. 验证

专用目标1准备验证

专用实践1.1 选择要验证的工作产品

专用实践1.2 建立验证环境

专用实践1.3 建立验证的规程和准则

专用目标2实施同行评审

专用实践2.1 准备同行评审

专用实践2.2 实施同行评审

专用实践2.3 分析同行评审数据

专用目标3验证所选的工作产品

专用实践3.1 实施验证

专用实践3.2 分析验证结果

其中确认关注于“构造了正确的产品”，验证关心“正确地构造了产品”，两者都是对软件测试方法、范围的扩展，始终贯穿于软件开发过程中。基于GJB5000A三级的软件测试流程主要有测试计划、测试设计、测试开发、测试执行、测试分析、缺陷管理，由于软件测试时存在于整个项目生命周期里的，所以测试流程也是置于需求管理、配置管理和风险管理之下，并可同时参与测量与分析。

确认策略

Validation，确认系统实现了正确的功能，用在最终的产品交付确认阶段，是一种融合归纳的综合手段。确认所开发的软件是否满足用户真正需求的活动。相当于，保持对软件需求定义、设计的怀疑，一切从客户出发，理解客户的需求，发现需求定义和产品设计中的问题。主要通过各种软件评审活动来实现。

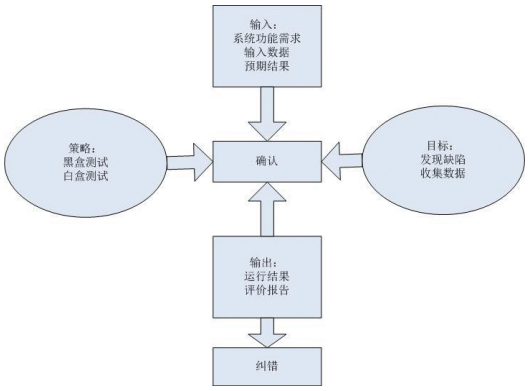


图1 确认

验证策略

Verification，验证系统没有做出错误的事情，用在测试的“集成”阶段，是一种抽丝剥茧的分析手段。“验证”是检验软件是否已正确地实现了产品规格书所定义的系统功能和特性。验证过程提供证据表明软件相关产品与所有生命周期活动的要求（如正确性、完整性、一致性、准确性等）相一致。相当于，以Spec为标准进行软件测试活动，验证软件产品和Spec的一致性。

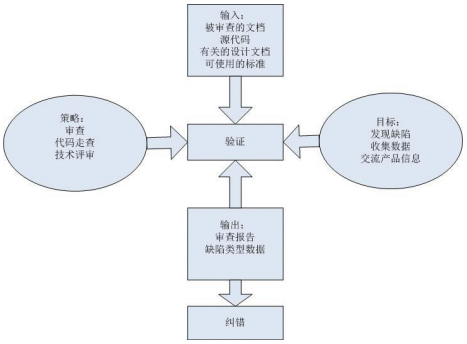
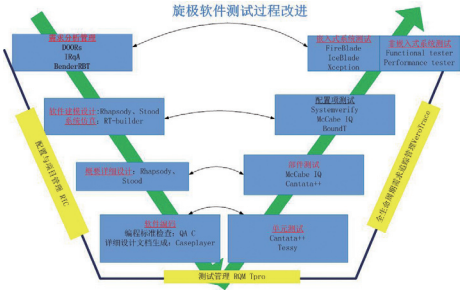


图2 验证

在测试计划过程中，针对软件任务书和软件需求准备测试，指定软件级别，定义单元测试、配置项测试、代码走查、同行评审行为，利用测试管理工具RQM或Tpro进行测试过程管理，同时借助黑盒测试用例生成工具如Bender RBT因果图建模分析此需求，生成测试用例；在测试设计过程中，评估测试用例，编写软件测试说明，并可借助需求管理工具Doors实现测试与软件需求的追踪；在测试开发中，准备测试运行环境，如特定硬件、虚拟平台或网络环境等，同时根据测试说明开发测试执行脚本；测试执行过程就是根据测试说明和测试脚本执行测试的过程，必要时需借助工具记录测试结果、分析覆盖率；在整个测试执行过程中，测试策略采取先静态，后动态，先黑盒，后白盒的方法，首先使用QAC对被系统代码进行标准符合性检查，然后搭建动态测试运行环境，使用Cantata++编写测试脚本，控制程序运行，得到功能测试报告与详细的覆盖率报告，最后使用McCabe IQ进行软件整体的质量分析；测试结果的分析就是对测试通过与否的分析，分析测试覆盖率，补充测试用例，记录软件缺陷，提交软件问题报告单，辅助软件改正bug；通过缺陷管理，追踪缺陷所关联的需求、设计和测试，对修改后的代码进行回归测试。



CMM在软件测试中的应用

■ 文 | 航天软件评测中心 苑淑云 冯志华

摘要：本文简单介绍了CMM内容及软件测试知识，根据CMM在软件开发过程中的应用，结合本软件测评单位对CMM的具体应用，重点讨论了如何将CMM应用在软件测试中，以提高软件测试质量和效率。

关键词：CMM模型、软件测试、软件测试过程、缺陷管理

1 引言

软件系统的规模和复杂性与日俱增，软件市场的风险越来越高，要求在更短时间内开发出更大规模软件的压力日益增大。这种形势不仅要求软件开发者更快地开发软件，同时也要求所开发出的软件具有能让客户满意的质量。当前，软件的质量愈来愈受到广泛的重视，而测试又是保证软件质量的重要手段。随着软件规模和复杂性的增大，软件开发交付周期缩短，软件测试的工作强度也随之增大。

软件测试是一项非常繁琐并且复杂的工作。如何才能够提高测试的效率和质量已经成为软件工作人员最为关注的问题。由美国卡内基•梅隆大学软件工程研究所提出的软件能力成熟度模型，简称CMM，提供了一个软件工程成果和管理方法的框架，自20世纪90年代初正式提出以来，CMM为生产流程的控制提供了可靠的保障。将CMM标准应用于软件测试行业，也能够极大地提高软件的成熟度和可靠性。下面就重点讨论如何将CMM应用于软件测试的一些具体方法。

2 软件测试

软件测试是为了发现错误而执行程序的过程。或者说，软件测试是根据软件开发各阶段的规格说明和程序的内部结构而精心设计一批测试用例(即输入数据及其预测的输出结果)，并利用这些测试用例去运行程序，以发现程序错误的过程。

2.1 软件测试过程

软件测试过程按照以下步骤进行，即单元测试、集成测试、系统测试和验收测试，如图2—1所示。

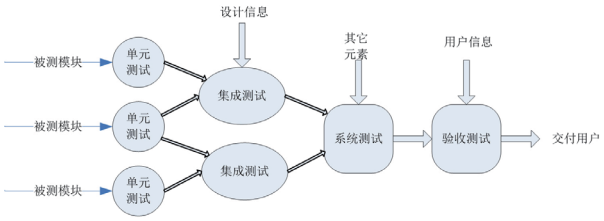


图2-1：软件测试过程图

单元测试集中对源代码实现的每一个程序单元进行测试，检查各个程序模块是否正确地实现了规定的功能。

集成测试就是在单元测试的基础上，根据设计规定的软件体系结构，把已测试过的模块组装起来，按照设计要求组装成为系统，在组装的过程中，检查程序结构组装的正确性。

系统测试是将软件放在整个计算机环境下，包括软硬件平台、某些支持软件、数据和人员等，在实际运行环境下进行一系列的测试。系统测试的目的是保证软件确实做了客户需要它做的事情。这种测试是通过执行用户场景模拟真实用户使用系统，以证明系统具有被期望的功能。严格地说，系统测试已经超出了软件工程的范畴。

验收测试即是要检查已实现的软件是否满足了需求规格说明中确定了的各种需求，以及软件配置是否安全、正确。

2.2软件测试过程

软件测试过程包括测试分析、测试设计、测试执行、缺陷分析、测试确认、测试总结。通常采用的测试方法是黑盒测试和白盒测试。

3 CMM简述

CMM的全称是软件能力成熟度模型，它是一个为软件组织在其开发和维护过程中获得控制、并向软件工程和优秀管理的文化进化提供指南的模型。CMM的体系结构由成熟度等级、关键过程域、过程能力和关键实践等内容构成。图3-1描述了构成CMM的5个成熟度等级。

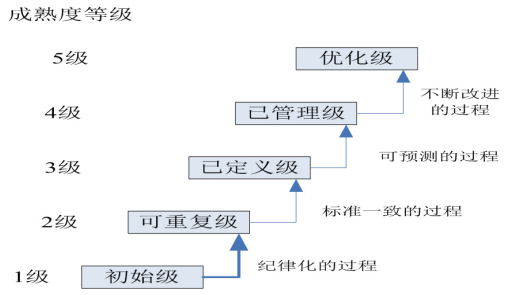


图3-1 CMM的5个成熟度等级

CMM为企业软件能力提供了一个阶段式的五级进程。任何开始采纳CMM体系的机构都一并归于第一级的起点，即初始级。除第一级外，每一级都设定了各自的目标组。如果达到了这一目标，则可向下一级推进，由于每个级别都必须建立在实现了完成它的全部级别的基础上，CMM等级的提高只能是一个渐进有序的过程。

处于一级的企业工作无序，项目过程中常放弃原来的规划，并且进度、预算、功能、质量不可预测；处于二级的企业过程制度化，初步实现标准化，可重复以前成功项目的环境与条件；处于三级的企业其开发过程均实现标准化、文档化；处于四级的企业其过程已定量化，可预测产品质量趋势并实现项目产品和过程的控制；处于五级的企业其过程可自发不断改进，可取得过程统计数据，并进行分析，从而得出最佳方法。

除第一级外，CMM的每一级都是按完全相同的内部结构构成的。每个成熟度等级包含了实现这一级目标的若干关键过程域(KPA)。每一级的每个KPA进一步包含若干关键实践(KP)。通过执行各个关键实践以完成关键过程域确定的一组目标。

4 CMM在软件测试中的应用

当前，随着CMM模型的逐步成熟，国内相当一部分软件企业开始有意识地接受并采用CMM作为其软件生产过程的制度体系和标准。同时，越来越多的软件企业已经逐步认识到软件质量的重要性以及软件测试在整个软件生命周期中的重要地位。但是，如何将CMM理论与软件测试技术相结合，并在CMM环境下应用软件测试技术和方法仍然是个难题。本文仅对CMM在软件测试过程管理的应用进行分析。

4.1需求管理

现有资料表明，在程序的故障中有56%的缺陷来自软件产品说明书，可见大部分缺陷是需求分析阶段导致的，所以测试需求管理在测试过程中占有极其重要的地位。

在开展一项测试项目时，首先要审核软件需求，确定软件特性和测试点，将测试点细化。然后，审核软件设计文档，确定该项目中应使用的测试方法和测试工具，着手准备建立测试环境。最后，根据上述测试分析，确定测试组成员，形成测试需求分析报告。

4.2项目策划管理

软件测试计划管理就是安排好测试流程。它主要包括软件测试策划、测试技术剪裁、测试进度管理等。在完成了测试分析和调研之后，就应该确定测试的大方向和长、中、短期目标。首先，制定项目大方向的长期目标，并开会讨论制定的目标、时间节点和项目组人员构成是否具有可行性，进度安排是否合理等。对计划制定的不当之处进行调整后，形成项目测试计划。然后，对项目测试计划进行细化，制定项目中、短期目标，对每个阶段时间和人员安排做出具体说明。

4.3项目跟踪和监督

这个关键过程域也可称为是对测试进度和质量的管理，是整个项目按进度节点如期完成的先决条件。通常，软件测试管理者监测软件活动主要通过在所选定的时间点或里程碑处，将实际的软件规模、工作量、时间表与计划相比较来确定进展情况，当确定未实现拟定目标时，采取纠正措施。这些措施可以包括修改剩余工作量计划或其他改进项目性能措施。通常采用周报、日报、项目例会、里程碑评审等方式了解项目进展情况，建立并使用软件度量过程来收集和分析项目实际状态数据，对项目的工作量、成本、规模、需求、质量、关键路径和关键计算机资源等方面进行跟踪监控，达到项目管理的目的。软件测试负责人可以通过测试计划管理工具，及时跟踪和修订测试计划，并汇报项目经理，从而使项目经理能够有效的进行项

目的全局管理。

4.4软件质量保证

软件测试的质量保证通过缺陷管理、培训管理和建立质量管理小组3个方面实现。通过缺陷管理规定一个缺陷从发现提交到被确认又到被修复的整个过程。如果要提高组织的过程能力，就应该建立良好的培训制度。质量管理小组成员对软件测试的全过程进行监督和管理。

4.4.1缺陷管理

缺陷管理是利用缺陷管理工具来进行缺陷的跟踪与管理以及项目信息记录，包括缺陷管理、项目信息设置、用户信息设置、知识共享等功能。具体而言，缺陷管理模块，包括了缺陷新增、确认、修改、删除、查询、共享、导入、导出、实时发送电子邮件、生成编译申请表等操作，并跟踪每个缺陷状态。

4.4.2培训管理

在考虑项目现状和发展计划的基础上，分析测试小组人员的发展要求，提出测试人员对测试理论、测试技术等方面的培训要求。包括受训人员提交的培训申请。将申请表单提交到培训协调员处，由协调员统一整理报送相关负责人进行审批，并在一定时间内给出审批结果。

根据培训需求分析制定培训计划，确定培训教师、培训内容、培训地点。按事先制定的培训计划执行。

由培训协调员维护培训数据库，及时做到新数据库的录入、修改、历史数据的备份、删除等，并在需要的时候导出数据内容以供使用。同时，收集受训人员对培训的反馈意见，整理后输入培训数据库中，为后续工作提供有意义的参考。

4.4.3建立质量管理小组

质量管理小组成员由部门领导、项目经理和项目测试负责人共同组成，对测试项目质量进行监督和定期考察，确保测试过程规范，测试执行按标准执行，测试内容符合测试需求分析报告要求。按时期对测试产品进行内部评审。

4.5软件配置管理

软件配置管理活动是一项支持性、保障性的工作。配置管理计划应该在测试项目的开始之初制定，如果不在项目开始之初制定配置管理计划，它的直接后果就是项目管理的混乱并使配置管理活动成为一种“救火”的活动。因此及时制定配置管理计划在一定程度上是项目成功的重要保证。

在配置管理工作中，角色定义及权限分配是两个重要过程。角色是指配置管理流程的执行者和参与者。定义明确的角色有利于实现明确的授权和明晰的流程。

（1）配置管理员

整个配置管理库由配置管理员进行管理。配置管理员负责分配和修改其它成员的权限，要维护所有目录和配置项。

（2）测试负责人

测试负责人负责组织测试，给出测试计划和测试方案，并核定测试报告。测试负责人对所有模块都有读取权限，对测试用例和测试报告有读写权限。

（3）测试工程师

测试工程师负责完成测试工作，包括测试用例设计和测试执行，测试报告撰写。测试工程师对自己负责的模块有读取权限，对自己负责的测试用例和测试报告有读写权限。

（4）SQA工程师

SQA工程师拥有对所有项目的读取权限，及对SQA类文档目录的读写权限。

另外，除配置管理员外，其他人员都没有删除目录文件的权限，这是为了防止误删除操作带来不可挽回的损失。如果需要对目录进行删除操作，必须由配置管理员进行。

5 经验与体会

通过将CMM的成功经验应用到测试过程的管理和改进当中，取得了一定的成效，以下是在具体测试项目中实施CMM的几点体会。

5.1积累项目数据成为宝贵财富

在项目策划活动中关键的是做出各种估计，包括工作量、工期、资源数、所需关键计算机资源和风险等。根据项目的规模，依据组织的生产率因子，估计出完成项目所需的工作量，按照任务书的要求排出每个活动的工期，需投入的工作量，根据工期和工作量可得出每个活动所需的资源数，最后即可得到整个项目的工作量、工期、资源和成本估计。在项目开始之前，列出所需关键计算机资源（包括运行环境、测试环境等），并估计出何时要用，需要持续占用多长时间。对于可能出现的风险，综合考虑其

发生概率和损失后果列出优先级，估计风险的可能发生时间，并给出防范措施。

在项目进行过程中，要对这样估计数据进行追踪，将实际数据与估计数据进行比较，当偏差较大时，就需要采用措施纠正偏差。大量项目积累的数据将成为组织的一笔宝贵财富，可使得组织的项目计划和跟踪达到稳定，并能重复以前的成功，这点对于新成立的项目组和新的项目负责人尤其重要，可有效缩短项目负责人的培养周期。

5.2对目标实现过程进行记录和追溯

项目组要建立单独的需求追溯文件，并在项目工作过程中不断进行需求追溯，以保证项目的管理和工程工作均满足需求。对于测试项目，如果分配需求包括代码审查，则在进行测试计划和设计过程中，要利用代码审查追踪矩阵表建立被测软件的代码清单，在代码审查的过程中逐步填写代码特征、审查工作分配和审查完成情况，通过代码审查追踪矩阵表，项目负责人可动态掌握项目组代码审查工作的进度、责任人，以检查代码审查工作是否按测试计划要求完成。如果项目要求进行功能测试，则在进行测试计划和设计过程中，利用软件需求追溯矩阵表建立被测软件的需求规格说明清单，在测试用例设计及实施过程中逐步填写每个测试用例对需求的追溯关系以及测试用例的执行情况，以检查测试用例设计和测试实施对被测软件需求规格说明的覆盖及完成情况。

代码审查追溯矩阵和软件需求追溯矩阵除了可供项目负责人了解项目进展及对目标的完成情况外，它们还是项目工作产品的组成部分，分别包含在软件代码审查文档、软件测试说明文档和软件测试报告文档中。文档的阅读者及评审人可通过这两个追溯矩阵清晰地看到项目对代码审查和功能测试任务的完成情况，从测试用例的执行结果（通过、未通过、未执行）向前追溯，可清楚地知道软件的功能需求满足情况，发现的问题将影响哪些功能实现等方面。

5.3跟踪监督使管理细化有效

CMM的软件项目策划过程规范要求项目在启动时必须制定完整的“项目计划”，选定生命周期模型，确定阶段和里程碑计划，确定详细的工作分解结构（WBS），工作分解采用层层细化的方式，将所有的活动、任务分配给项目组的具体人员，确定负责人，明确职责。由于工作分解的力度适度，便于管理，在对月度计划进行跟踪时，当实际进度、工期和工作量与计划的相比偏差大于15%时，就能及时采取措施，进行纠偏，以保证项目的工作按计划完成。

合理有效的监督机制是提高和保证过程质量的关键。项目组成员每天在个人工作周报上填写工作日志，在周例会之前项目负责人审阅项目组成员的周报。项目组长将工作月报和阶段总结报告呈交项目经理进行审查，综合考虑进度、规模和工作量的偏差确定是否需要调整项目计划。软件质量保证人员按照项目的质量保证计划，根据项目的工作产品和工作记录，检查项目的开发及管理工作与项目计划、需求和规范是否符合，并填写项目过程SQA报告，说明和记录所开展的工作及发现的偏差和问题，对发现的问题及时报告责任人，并跟踪到问题的解决归零。这些跟踪监督活动，使项目组长对项目组成员的工作，项目经理对项目的工作都能做到心中有数，对保证项目按计划完成以及完成的质量都起着十分重要的作用。

6 总结

本文简单介绍了CMM模型和软件测试的基础知识，并结合航天软件测评中心对CMM的具体应用，讨论了如何将CMM模型应用在软件测试过程中，从而对提高软件测试质量和效率。实践证明，在项目策划、需求管理、项目跟踪与监督、质量保证和配置管理等关键区域上形成了一套既符合CMM过程改进要求，又紧密结合软件评测工作需要的规范、标准和程序，其效果已经在实际工作中得到体现。应用CMM取得的成果会进一步在软件评测工作起到积极和重要的作用。

参考文献

[1] 郭永梅 《软件能力成熟度模型在软件测试中的应用》全国高校软件工程专业教育年会(2007)论文集

[2] 董威 《基于CMM的软件测试技术及其应用》华东师范大学2005届工程硕士申请硕士学位论文

[3]张可东 庄燕滨 《软件工程与软件测试自动化教程》电子工业出版社 2002年7月

[4]周之英 《现代软件工程》科学出版社

[5]郁莉莉 《基于TMM的软件测试管理的研究与应用》复旦大学硕士学位论文 2006年

[6]朱艳 《基于CMM的软件测试管理支持系统研究》大连海事大学硕士学位论文 2006年

软件测试新技术

P21 我的探索式软件测试实践

P23 嵌入式软件测试浅谈

P25 线性独立路径覆盖测试用例自动生成方法研究

P30 Linux 安全性测试要点浅谈

我的探索式软件测试实践

文 | 成都旋极软件工程部经理 周珊

随着敏捷测试的推广，探索式测试逐渐受到大家的关注和重视。自动化测试作为敏捷测试的基石，手工测试作为自动化测试的重要补充，它所关注的是“意料之外”的软件缺陷，而探索式测试（Exploratory Testing）应该是我们在手工测试领域使用的最好的技术了。探索式测试作为一个研究性、启发性和严肃性并存的测试方法，是一般性测试的重要补充。

探索式软件测试是一种可以应用于广泛测试技术的方式或态度，《测试计算机软件》作者CemKaner明确地将探索式测试定义为“一种强调每位测试人员的自由和责任的测试，每个测试人员通过将学习、测试设计、测试执行和测试结果解释作为贯穿项目的平行活动，从而持续地优化他们的工作价值”。

我们可以通过探索式软件测试来制定我们的测试计划，推荐一款非常实用的商业思维导图软件X-MIND，我们在做探索式软件测试的时候要注意以下几个点：

1. 测试就是学习
2. 了解你的任务
3. 变化很重要
4. 分析很重要
5. 学会聚焦和散焦

以一个骰子游戏为例子。任选五个骰子，任意投一组数后作为输入，系统会告诉我们输出是什么。给定十分钟时间，通过几次投骰后判断出系统的规律，然后来进行测试，那么面对这个系统的测试我们的难点是什么呢？

第一，我们对这个系统并不了解，我们只知道这个系统得到输入后会有输出；

第二，我们没有完善的文档来分辨软件的功能和缺陷；

第三，我们没有可用的测试脚本。

假设我们把这个作为一个探索式测试的内容，我们用X-MIND来记录我们的思维过程，如下：

可以从图上清晰的看到我的思维过程，我总共做了四次尝试，第一次因为题目告诉我用5个骰子投的数作为输入，因此我随机投了三组数，得到了三个输出，本来想根据这三组数能推测出系统的逻辑，但发现是徒劳的；第二次，我想能不能尝试减少输入，因此我用了四个骰子投了两组数后，发现第二组数得到一个很奇怪的输出0，第三次我开始尝试用单个的骰子开始摆数，得到了结论1=0，2=0，3=4，4=0，5=6，6=0；最后一次我摆了两组数验证了我第三次尝试得出的结论，发现得出推论完全正确。

在这个简单的探索性测试中我们把上面总结的探索性测试的几个点都涵盖进来了：

第一，我们的测试就是我们学习了解这个系统的过程；

第二，了解我们的任务，弄清楚我们到底要测什么。因为探索式测试的缺点在于测试人员有可能在测试中没有重点，从而漫无目地的尝试各种情况来试图发现软件的缺陷，没有条理，缺乏方针的盲目测试，因此指导方法（注1）很重要。就好像游客到新城市，然后盲目彷徨地想找到景点一样，有个导游就完全不一样了，他可以帮我们弄清楚自己的目的地在哪里（对我们而言，就是软件本身要测什么），让探险计划有条不紊；

第三，变化是很重要的，如果我在第一次尝试的时候就不停的扔骰子确定输出来判断逻辑，那么我想我会无休止的做下去也很难找到系统的规律；

第四，我们在做尝试之前需要先分析，然后根据每次得到的结论分析下一次可能的情况，题目并没有明确的说明输入只能是五个数，那么我可以尝试用四个或者更少的一个我想要的输入来判断结果；

第五，聚焦和离焦。我们的测试输入大多数时候可以用“变化无穷”这个词来概括，当我们测试某个点的时候，什么时候该继续什么时候该退出这都需要测试人员做一个判断。根据任务的时间和我们的经验，适当的时候聚焦和离焦，以保证我们的任务按时按量完成。

通过上面的探索式测试我对系统的功能逻辑得到了基本的了解，那么我们就可以开始定制我们的测试计划了，同样我们也可以用X-MIND这个工具来罗列我们的测试大纲。虽然这个例子对于现实中千变万化的情况来说太过于简单，而且探索式测试也同样有很多的技术和方法在本文也没有具体的说明，但是也能概括我们在探索式测试中面临的很多问题和解决问题的方法。

软件工程领域没有银弹，在软件开发过程中所采用的方法或技术只能解决一部分问题，而不能彻底解决所有问题。在测试领域，这个特点更加明显，例如航空航天方面的软件，规模都是比较大的，测试更不可能穷尽，测试更不能做到百分之百，总是有不能测到的地方，这就给我们留下了足够的探索空间。探索式测试的出现正是因为在软件系统中存在许多未知的东西难以得到快速、简单的验证，需要我们转变思路，不要以固定的模式来完成测试，而是要换一种新的模式来进行测试，以提高测试效率。更重要的是让测试过程成为科学探索的一部分，将无味的测试工作变成有趣的探索，在享受工作的同时完成测试。

本文以我做的例子来阐述了探索式测试的几个注意点，也介绍了些我自己的经验，当然经验不等于事实，仅供参考，希望大家在测试这门学科上共同进步吧。

注1：探索式测试有两种指导方法，一种称为局部探索式测试法（exploratory testing in the samll），另一种称为全局探索式测试法（exploratory testing in the large），它们都可以帮助测试人员做具体的决策。

嵌入式软件测试浅谈

■ 文 | 旋极信息军工中心副总工程师 皮永辉

所谓嵌入式软件是指嵌入式系统或产品中的软件，与之相对应的就是所谓的通用计算机软件。嵌入式软件和通用计算机软件相比既有共性，也有不同。嵌入式软件的特别之处首先是软件与硬件密切相关，需要与硬件交互工作；其次是嵌入式软件的实时性要求较高，以承担实时约束、控制的重任；还有就是嵌入式系统的多样性，没有统一的架构和平台；当然，其交叉式开发的特点，也区别于通用计算机软件。正是因为这些特点，嵌入式软件的测试就需要我们另眼相待。

软件测试发展至今已形成了较为完整的理论、技术和策略，虽然它们大多针对通用计算机软件，但幸运的是其基本原理和典型方法同样适用于嵌入式软件。这就为我们测试嵌入式软件打开了思路：基于两种软件的共性，我们首先利用通用计算机软件的测试技术，即“拿来主义”，然后从嵌入式软件的特点出发，采取一些有针对性的方法，比如“交叉测试”。

交叉测试

交叉测试又称Host/Target测试，与交叉开发是相对应的。我们知道，嵌入式软件是运行在其目标硬件（Target）上的，而软件的开发，包括编程、编译、调试等却是在主机（Host）上完成的。这是两个互不相干的工作平台，通过所谓的“开发环境”实现二者的“交叉”。这就给我们提出了一个问题：嵌入式软件测试应该在主机上进行，还是在目标上进行？在主机上做，我们熟悉，方便，但无硬件实时交互，不能完全反映目标实际情况。在目标上做，环境真实，但不便实施，需要开发工具支持，往往受限于目标资源。二者各有利弊。

交叉测试实际上发挥了上述两方面的优势。先将大部分与硬件基本无关的代码在主机上测试，充分利用高级语言的可移植性和后面提到的通用计算机软件的测试技术。再将与硬件密切相关的代码在目标上测试和验证，此时需要开发工具的支持。当然，测试工具也必须适用于这一开发环境和目标硬件，或者说你所选的测试工具应该支持交叉测试。

嵌入式交叉开发有模拟调试（Simulator）、在线调试（Debugger）和仿真调试（Emulator）等多种形式，都可用于交叉测试。其中模拟调试是在主机上完全模拟目标的行为，无需硬件参与，非常方便，是交叉测试中经常用到的形式。

交叉测试主要适用于单元测试（功能测试）。除了交叉测试，针对嵌入式软件我们还有“全数字模拟测试”、“实时在线测试”、“故障注入测试”等方法，多用于系统级，以及性能、可靠性等方面的测试。

拿来主义

通用计算机软件测试技术包括白盒测试/黑盒测试、静态分析/动态测试、单元测试/集成测试等。这些技术都是针对高级语言的，都可以用于嵌入式软件。

静态分析很重要 统计表明，有经验的软件工程师平均每写1000行代码会出现100个错误。80%的错误可归咎于对编程语言的错误使用，这些错误往往要通过静态分析（源代码分析）而不是动态测试（功能测试）才能解决。所以软件工程专家Humphrey说，欲使动态测试取得好的效果，必须要有一个“良好的”产品投入测试，这就凸现了静态分析的重要性。或者说，在动态测试之前开展静态分析可以达到事半功倍的效果。静态分析实施简单、快捷，无需设计测试用例，不仅能够尽早地发现问题，提高测试效率，而且可以明显地降低开发成本。

随着技术的不断发展，静态分析早已不是当初的“自动词法/语法分析”了，发展成了真正的“源代码分析”技术。 源代码分析技术主要有代码规范性检查、代码质量分析/复杂性度量、代码缺陷及安全性检查等，如同我们要确保我们的“身体质量”，需要在“预防”、“强身”和“治病”三个方面下功夫一样，

这些技术正是提高“软件质量”的重要手段。PRQA、Goanna就是非常优秀的源代码分析工具，可用于“预防” 和“治病”，而McCabe则在软件质量分析方面独树一帜。

动态测试不可少 软件的功能最终要通过软件的运行来实现，所以动态测试必不可少，是验证软件功能最直接、最有效的手段。通常，对嵌入式软件会有较高的代码覆盖率以及性能和可靠性方面的要求，这也需要通过动态测试才能实现。前面提到的交叉测试，就是嵌入式软件动态测试的常用方法。动态测试需要从软件的需求（规格说明）入手，设计好测试用例。白盒测试、黑盒测试都可用于动态测试。

一般来说，动态测试是一项工作量较大而又比较繁琐的工作。对需求的正确理解，设计好测试用例是我们做好动态测试的前提。比如，需求定义是否完整？是否有二义性或相互矛盾的地方？测试用例什么时候开始准备呢？我们要达成怎样的代码覆盖目标？测试路径如何选择？需要多少个测试才能覆盖到所有功能？怎样才能使我们的测试既充分又不冗余？这些都是我们开展动态测试必须考虑的问题。

幸好也有自动测试工具帮助我们。Cantata和Tessy就是两个著名的支持嵌入式的动态测试工具，前者用于C/C++语言的单元/集成测试，后者更是专为嵌入式软件的单元测试而设计。另外，测试用例设计的好坏直接关系到动态测试的成效，如果想使测试的功能覆盖最大而测试的数量又最小，使用基于需求的测试用例设计工具——Bender-RBT就是明智的选择。

白盒黑盒相辅成 白盒测试是一种注重过程的测试方法，要求“内部的齿轮都吻合”。从被测代码的逻辑结构入手，测试其路径、控制、数据流等是否按预设的路线行进。如果过程的每个环节都正确，我们就有理由相信最后的结果也正确。

与白盒测试相反，黑盒测试是一种注重结果的测试方法，正所谓“不管白猫黑猫，抓到老鼠就是好猫”。黑盒测试多用于功能测试，在被测程序的接口上选择合适的输入参数，执行程序后再比较实际输出与期望的结果。

黑盒测试和白盒测试都是典型的软件测试方法，在实践中往往要综合使用，效果才好。

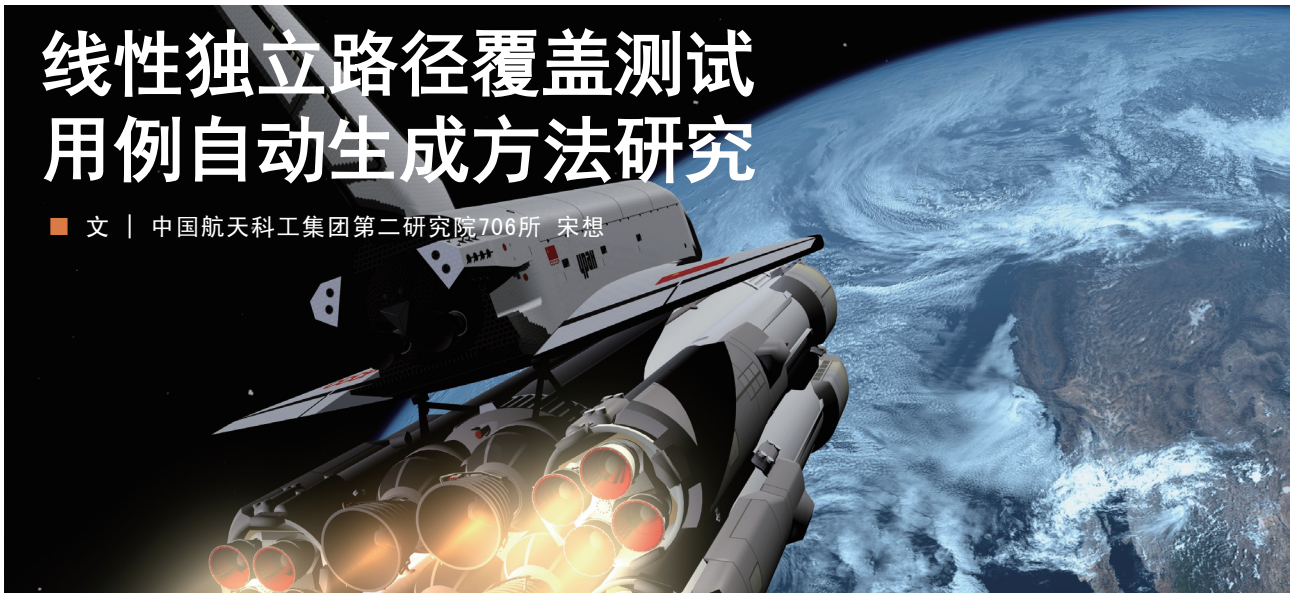
单元集成两步走 单元测试和集成测试是软件测试的两个阶段，反映的是一种测试策略。被测软件可以看成由若干个小的单元（比如函数）构成，按计划对这些单元逐个进行测试就是单元测试。单元测试通常使用“隔离法”，要用到“桩”和“驱动”。将单元逐步组合起来测试就是集成测试。显然，只有每个单元都测试通过了，集成测试才有意义。

结束语

以上简单地介绍了嵌入式软件的测试技术和方法，有些方面如系统测试、性能分析、回归测试等尚未涉及。总而言之，从技术上讲，嵌入式软件测试的目的是要试图回答以下两个问题：一是“软件功能对不对？”，二是“代码质量高不高？”。其实，一般的软件测试也是面对这两个问题，但由于嵌入式软件的专用性及特殊性，往往在这两方面有着更高的要求。对于第一个问题，还隐含着“性能好不好”的问题，需要通过动态测试去回答，首先分析需求定义，然后设计测试用例…，这就是“测试从需求开始”。对于第二个问题，还涉及到代码的可靠性、可移植和可维护等方面，使用源代码分析技术非常有效，这就是“质量从代码抓起”。需要指出的是，光有技术是不够的，我们还应该知道如何应用这些技术，以及懂得如何计划、组织和控制测试过程，这就是测试的管理问题。因此，技术和管理就成为软件测试稳步向前的“两条腿”，缺一不可。

线性独立路径覆盖测试用例自动生成方法研究

■ 文 | 中国航天科工集团第二研究院706所 宋想



摘要：在面向路径的测试用例生成问题中，路径覆盖是一种非常苛刻的覆盖准则，测试人员往往只能根据一定的准则对全路径集合的某个有限子集进行测试。线性独立路径覆盖测试就是一种对全路径的一个有限子集进行测试的方法。通过程序图求出程序的一组线性独立路径，然后分别针对线性独立路径组中的每条路径求出相应的测试用例，不仅繁琐，而且存在路径不可达问题。本文提出了一种直接生成线性独立路径测试用例的方法，不存在路径不可达问题。

关键词：面向路径测试；路径覆盖；线性独立路径覆盖；圈复杂度；测试用例自动生成。

一、绪言

1.1 背景及意义

软件自动测试的过程可分为测试用例的自动生成、被测程序驱动程序的生成以及执行结果的自动比较等几个部分。其中驱动程序模块用来模拟被测模块的运行环境;测试用例自动生成提供明确的测试数据和测试结果;执行结果的比较用来比较对应测试数据的被测程序的输出与预期结果是否一致。驱动程序模块以及执行结果的比较自动化较易实现，而测试用例自动生成部分由于往往要对程序的源代码和规则进行分析，因此实现起来相对较难。因此如何自动生成测试用例就成为解决软件自动测试问题的关键。目前，软件测试的自动化工具主要集中在测试用例的执行和维护以及测试覆盖情况的度量方面，测试用例的自动生成还远不尽如人意。而在软件测试过程中，动态测试占的比例很大，是软件测试的重要环节。动态测试的关键也是测试用例的生成问题。

测试用例自动生成技术可以为所测程序自动生成测试用例。开发这类程序的目的在于减少测试人员所必须付出的大量劳动，降低手工测试的高额成本，同时提高测试过程的可信赖程度。好的测试用例可以在测试过程中反复利用，同时，在测试过程中可以通过对测试用例的组织和跟踪来完成对测试工作的量化和管理。测试用例自动生成这一技术的实现，将大大改变以往靠直觉、经验产生测试用例的传统做法，无疑将使软件测试的效率获得显著提高，同时减少测试人员在编写大量测试用例过程中所付出的劳动。因此开展软件测试用例生成技术的研究并开发出适用化的测试用例生成系统，对实现软件测试过程自动化有着十分重要的现实意义。

1.2 论文研究内容

本文的主要研究内容如下：

1.线性独立路径覆盖测试用例的生成：目前，线性独立路径覆盖测试用例的生成都是先根据流程图求出程序的圈复杂度，然后根据流程图找出一组线性独立路径，再分别求出每条线性独立路径对应的测试用例。但由于机械地将源代码表达为有向图和程序路径，掩盖了代码中的逻辑信息，通过程序图得到的基路径在程序实际运行中由于存在相互矛盾的语义依赖关系有些是不可达的，也就无法得到相应的测试用例。为此，本文通过执行一个用例得到一

个路径向量，执行若干用例得到一个路径向量矩阵，然后比较矩阵的秩与圈复杂度的大小来判断这些用例的线性独立路径覆盖是否为百分之百。

2.将遗传算法应用于线性独立路径覆盖测试用例的生成：以往，遗传算法都是应用于生成满足覆盖某一条指定路径的测试用例，本文尝试用遗传算法生成满足线性独立路径覆盖的一组测试用例。

二、测试数据自动生成相关概念、技术及研究现状

2.1 测试数据自动生成相关概念

测试用例是一种为了实现软件测试有效性的常用工具，它是为了特定目的而设计的测试数据与相关的测试规程的一个特定的集合，或称为有效地发现软件缺陷的最小测试执行单元。

目前在许多软件项目开发过程中，测试人员在进行软件测试过程中，特别是在单元测试中采用手工方法设计和生成所需测试用例。当开发程序规模大及程序执行路径复杂时，手工编写测试用例的工作量很大，构造路径全覆盖测试用例非常困难且容易出错，造成测试效率不高，极大的影响软件的开发进程。自动化测试技术能有效的解决手工测试带来的问题，如果能让测试中的测试用例自动生成，将有效地减轻测试人员的劳动强度和提高测试准确性和执行效率。测试用例的自动生成技术成为了现在被广泛研究的测试理论与应用领域。

测试用例的自动生成是指通过特定的算法依据软件规约或程序结构自动构造测试用例的技术[6]。依据软件规约的测试用例自动生成又称为面向功能的测试用例自动生成，依据程序结构的测试用例自动生成又称为面向结构的测试用例自动生成。本课题主要研究的是面向结构的测试用例自动生成技术。

面向结构的测试用例自动生成技术是为了结构测试而产生的，它针对程序内部逻辑进行测试，并要求对程序的结构特性做到一定的覆盖，是“基于覆盖”的一种测试。结构测试是针对程序内部逻辑的测试，它要求对程序的结构特性做到一定的覆盖，是“基于覆盖”的一种测试。面向结构的测试用例生成方法分为三种：随机测试用例生成方法、面向目标的测试用例生成方法和面向路径的测试用例生成方法。通常来讲，结构测试用例生成一般采用最常见的基于“路径覆盖”的方式来产生，即在对程序单元进行结构测试时，根据指定的程序逻辑路径，在被测程序单元的参数空间域搜寻能执行该路径的实际数据[11]。

2.2测试用例自动生成技术国内外发展情况

目前，在测试用例生成研究领域中，国内外的学者们对于测试用例自动生成技术进行多方面的研究，提出了很多方法，并且对它们的方法与技术的研究还在不断地探索中。

D .Bird等人通过使用随机法自动生成快速产生大量的数据，对于输入空间简单的时候测试简单有效，但是当输入空间无穷大时，随机法则不适合生成测试数据进行测试。E.Weyuker等人提出通过从布尔规格说明自动生成测试数据的方法来进行测试。W .T .Tsai提出通过关系代数查询表示的规格说明来自动生成测试数据的方法。N .Gupta等人提出的迭代松弛法，这种方法采用程序谓词切片的思想，在给定域中任意选择一组输入数据，对路径上的每个谓词分支，确定其谓词切片和输入依赖集，推导出各谓词函数关于输入变量的线性约束，建立输入变量增量的线性方程系统，用高斯消元法求解后得到各输入变量的增量，从而获得一组新的输入。若路径上各谓词分支都是输入变量的线性函数，该方法或者通过一次迭代找到解，或者保证路径不可行。当谓词函数中含有输入变量的非线性函数时，因为该方法是用线性函数作为非线性函数的近似，所以需要迭代多次才有可能找到解，因此该方法对于非线性路径约束是不完备的。GuPta等人将该方法应用于分支覆盖的测试数据自动生成。单锦辉博士则将迭代松弛法改进之后再用于测试用例的自动生成。改进了GuPta方法，省略了该方法中构造谓词切片和输入依赖集的过程，直接计算各谓词函数的线性算术表示，建立输入变量的线性方程系统，求解后获得一组新的输入。

三、基于路径的测试用例自动生成

在软件测试中，面向路径的测试用例生成问题描述为：给定一个程序P和P中一条路径W，设P的输入空间为D，求x∈D，使得P以x为输入运行，所经过的路径为W。目前，基于路径测试用例自动生成的方法主要分为两种：一种是基于符号执行，另一种是基于程序直接执行[5]。

3.1 基于符号执行

符号执行是指静态地分析给定路径的有关变量和谓词，对路径上赋值语句的左部变量及分支谓词的变量依次用输入的符号表达式进行替换，使该路径的分支谓词为有关输入的符号值的等式或不等式方程组[5]。符号执行允

许程序输入常量、符号值、符号表达式等，以符号计算代替实际执行的数值计算，产生一个符号输入值的代数表达式－路径约束，它是选定路径的谓词系统。路径约束实际上是对输入数据的限制要求，由多个不等式（等式）组成。通过求解不等式，求取满足路径上各限制谓词的测试数据。若不等式无解，则相应的路径为不可行路径。符号执行能够判定路径的可行性，一次符号测试的结果代表了一类普通测试的运行结果。但在遇到循环、过程调用、动态数据结构、数组和指针处理时，符号执行实现困难[4]。

3.2 基于程序直接执行

程序直接执行方式是指在对被测程序的有关信息了解不充分的条件下，通过

实际执行被测程序，经过多次循环探测，并对测试结果进行评价后，最终确定测

试用例。一般来讲，这种方法在实现上需要某种搜索寻优算法的支持，它在随机

产生的输入变量的初始值的基础上，在参数变量的空间域进行某种有导向的搜索，

直至最终找到测试用例。由于这种生成测试用例的方法避免了符号执行带来的问

题，而且易于实现、方式灵活，因而受到越来越多的人的重视[5]。

四、线性独立路径覆盖测试用例的自动生成

软件的单元测试中控制流测试中诸如语句覆盖、分支覆盖、条件覆盖、判定-条件覆盖、路径覆盖等问题和数据流测试中的全定值覆盖、全引用覆盖等问题都可以归结为面向路径的测试用例生成问题。本文主要研究的是基于程序直接执行的路径覆盖问题。

路径覆盖是一种非常苛刻的覆盖准则，即使一个小程序包含的路径数也可能是非常庞大的。

如下面一段代码：

```
①  for(int i = 0;i<20;i++)
    {
②    if(A>0)
③        { …… }
     else
④        { …… }
    }
```

其程序结构图如图4-1所示：

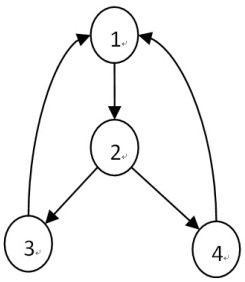


图4-1 程序结构图

这段程序可能的路径数为220。要做到百分之百路径覆盖，测试人员的工作量是非常巨大的。

测试人员往往只能根据一定的准则对全路径集合的某个有限子集进行测试。

4.1 线性独立路径的概念

Thomas McCabe于上世纪80年代提出了线性独立路径的概念,他认为可以将程序中所有可能的路径集合看成一个向量空间,那么在该向量空间中存在着一组基,通过对基的线性组合可以覆盖整个向量空间,因此McCabe认为找出向量空间中的基进行测试,如果基没有问题,则可以期待能够用基表示的一切路径组合都没有问题[7]，这组基也称为线性独立路径。

4.2 圈复杂度的概念

Mccabe 在1976年提出了圈复杂度的概念，他认为在一个强连通图G中，圈数等于最大的线性无关的环数。在所有的路径集合中，存在若干条线性独立路径，其他的所有路径都可以由这几条路径来线性表示，线性独立路径的条数就是圈复杂度[8]。

圈复杂度可以通过如下方法求得：

(1) 控制流图：V(G)=E-N+2p,p =1;

(2) 强流通图：V(G)=E-N+p，p为强流通区域的个数

(3) V(G)=R，R表示控制流图将将平面划分的区域的数量

(4) V(G)=p+1，其中p表示判定节点的数量，若判定为复合条件，则应将复合条件分解。每个case语句以及每个if，if else，语句都应算一个判定[9]。

4.3 传统线性独立路径覆盖测试用例生成方法及存在的问题

传统线性独立路径测试用例的的求法一般有以下三个步骤：

（1）根据程序的结构图求出程序的圈复杂度；

（2）根据流程图求出一组线性独立路径；

（3）针对线性独立路径组中的每条路径求出相应的测试用例；

传统线性独立路径覆盖测试用例的生成方法机械地将程序转换为结构图，完全忽视语义之间的相互依赖关系，导致其存在路径不可达问题，也就无法生成对应的测试用例。

以常用的三角形程序为例说明传统线性独立路径覆盖测试用例生成的过程以及存在的问题：

```
void  Triangle(int a, int b, int c)
{
BOOL  bIsATriangle;
1  if((a<b+c)&&(b<a+c)&&(c<a+b))
2  bIsATriangle=TRUE;
   else
3   bIsATriangle=FALSE;
4  if(! bIsATriangle)
5  printf(“Not a Triangle/n”);
   else
6  if(a==b && b==c)
7  printf(“Equilateral \ n”);
   else
8  if(a! =b && b! =c && a! =c)
9  printf(“Scalene \ n”);
   else
10 printf(“Isosceles\ n”);
   }
```

以上代码的结构图4-2如所示：

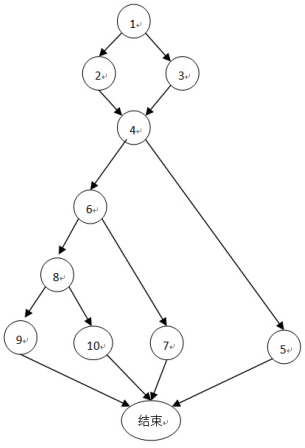


图4-2 三角形程序结构图

根据V(G)=E-N+2p,p =1 求其圈复杂度，将E = 14,N = 11代人得V(G) = 5.

下面根据McCabe的基线方法来确定基路径集合：首先挑出一个包含尽可能多判断节点的路径，不妨选择1-2-4-6-8-9-结束。接下来回溯基线路径，依次翻转每个判断节点，翻转判断节点8得到路径1-2-4-6-8-10-结束，翻转判断节点6，得到路径1-2-4-6-7-结束，翻转判断节点4得到路径集合1-2-4-5-结束，翻转判断节点1得到路径1-3-4-6-8-9-结束。由此得到下列基路径集合：

1-2-4-6-8-9-结束；

1-2-4-6-8-10-结束；

1-2-4-6-7-结束；

1-2-4-5-结束；

1-3-4-6-8-9-结束

传统线性独立路径测试用例生成的下一步就是针对基路径组中的每条路径分别生成对应的测试用例。

由于存在相互矛盾的语义依赖关系，3-4-6 在程序实际运行过程中是不可达的路径，所以上述基路径集合中1-3-4-6-8-9-结束是不可达的，也就无法生成对应的测试用例。

4.4 本文提出的线性独立路径覆盖测试用例生成算法

算法基于中国航天二院七0六所易加伟、宋晓秋提出的以下线性独立路径覆盖的充分性判别方法：

设待测程序有n行，根据其程序结构图得出其圈复杂度为V，并假设程序实际运行时可达的基路径数等于V。

（1）执行一个测试用例时，若经过某一行，记为1，未经过某行记为0，执行完测试用例后便得到一个n位的0、1向量。

（2）执行完m个测试用例后便得到一个m行n列的向量矩阵。求出矩阵的秩为R；

（3）若R<V,说明测试不够充分，没有测试到一组完整的基路径；若R=V，说明基路径覆盖率为百分之百；

（4）当用例不断增加时，得到的秩随用例数变化的曲线应该渐渐趋向平滑。

若直接应用上述充分性判别方法生成测试用例，存在两个问题：

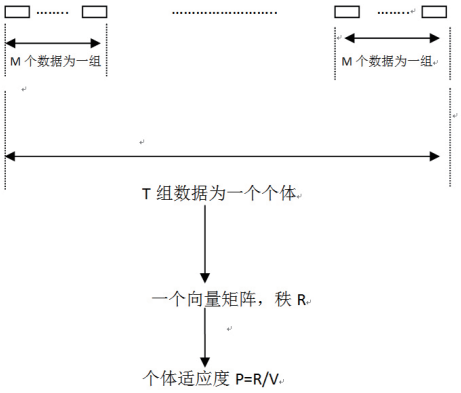
（1）当用例数逐渐增加时，得到的向量矩阵也越来越大，求矩阵的秩计算代价也增加了；

（2）用例的增加带有盲目性，不利于生成测试用例集的效率，而且用例数越多越背离基路径测试基的思想。

为解决以上问题本文采用一种进化的方法，固定向量

矩阵的维数，以矩阵的秩与圈复杂度的比值作为目标函数，不断进化找到一个向量矩阵使得目标函数具有最优值。

以遗传算法为寻优搜索算法来说明：设被测程序的输入为M个，M个数据称为一组，被测程序的圈复杂度为V。设T=>V，以T组数据的组合作为种群的一个个体，以该T组数据执行被测程序得到的向量矩阵的秩为R，以R和V 的比值P=R/V作为个体的适应度，以此适应度来进行进化。进化完后适应度最好的个体对应的T组数据作为测试用例。



本文提出的线性独立路径覆盖测试用例自动生成方法过程如下：

（1）根据程序的结构图求出程序的圈复杂度V；

（2）对源程序进行如下预处理：通过插装的方法来判断是否经过了某一行。考虑到每行都插装工作量过大，在每个分支的第一个语句后进行插装。通过判定节点统计程序中的分支数M,向量以数组C[M]来表示，元素初始值都为零。将C[m]=1插装在第m+1个分支中第一个语句后。执行完一个用例后就可以得到该用例对应的路径向量。

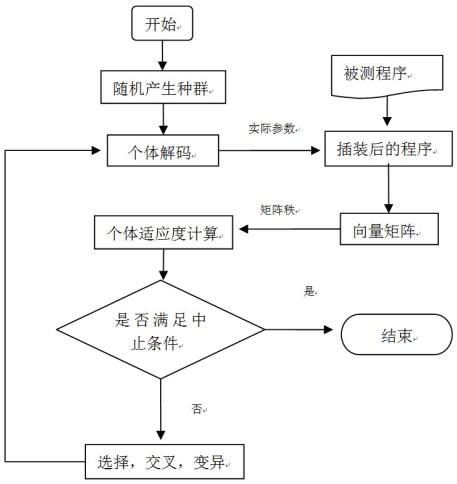


图4-3 算法模型

（3）进化求出适应度最好的矩阵向量，以其对应的T组数据作为线性独立路径覆盖的测试用例。

以遗传算法作为寻优搜索算法，则以上过程模型如图4-3所示。

参考文献：

[1] 贾松涛，张红卫. 面向路径的测试数据生成框架及应用.微计算机信息（管控一体化），2010，26卷2-3期

[2] Yan J, Zhang, J. An Efficient Method to Generate Feasible Paths for Basis Path Testing[J]. Information Processing Letters,2008,107(1): 87-92.

[3] 肖利琼. 软侧之魂[M]. 北京：电子工业出版社，2011

[4] 李军义. 软件测试用例自动生成技术研究[D]. 湖南大学，2007

[5] 曹烽燕. 基于基本路径测试的测试用例自动生成应用研究[D]. 大连海事大学，2007

[6] Chen H Y,Tse T H,Chen T Y. TACCLE:A Methodology for Object-Oriented Software Testing at the Class and Cluster Levels. ACM Transaction Software Engineering and Methodolog,

2001,10(4): 56-1

[7]易加伟，宋晓秋.线性独立路径覆盖率的软件测试充分性判别方法[J].计算机工程与设计,2005,26(12),3338-3340.

[8]傅博.基于蚁群算法的软件测试数据自动生成[J].计算机工程与应用,2007,43(12),97-99.

[9]单锦辉,王戟,马晓冬等.面向路径的测试数据自动生成工具及其图形界面TcK设计[J].计算机工程与应用,2002,1:7477

[10]荚伟,高仲仪.基于遗传算法的软件结构测试数据生成技术研究[J].北京航空航天大学学报,1997,23(1):36-40

[11]陈文萍.基于遗传算法的软件测试用例优化技术研究[D].北京:北京化工大学,2009.

[12]李义军.软件测试用例自动生成技术[D].长沙:湖南大学,2007.

[13]乔素琴.面向结构的测试数据技术[D].太原:太原理工大学,2009.

[14]D.Brown, R.Roggio, J.C.11, and C. Creary.An automated oracle for software testing [J].IEEE Transactions on Reliability, Vol.41, June1992, pp.272-280.

[15] N.Gupta, A.P.Mathur, andM.L.Soffa. Automated Test Data Generation using An Iterative Relaxation Method[C]// ACM SIGSOFT Sixth Internationa1SymPosium on Foundations of Software Engineering (FSE-6), pages: 231-244, Orlando, Florida, November1998.

[16] Grzegorz Kondrak, Peter van Beek. A Theoretical Evaluation of Selected Backtracking Algorithms [J].Artificial Intelligence, Volume89, Issue 1-2, anuary1997, PP.365-387.

[17]Kostas Stergiou. Representation and Reasoning with Non-binary Constraints [D]. A thesis submitted to The Univ. of STRATHCLYDE for the degree of PhD.

Linux 安全性测试要点浅谈

■ 文 | 西南电子设备研究所 叶嵩

随着计算机和网络应用的不断发展，操作系统的使用领域也随之拓展。Linux操作系统有着诸多优点，如开放源码，高度可定制，配套应用丰富和具有庞大的支持社区等等。相对于其他操作系统，Linux操作系统被更广泛地应用于大型服务器、桌面系统、移动计算设备和嵌入式控制设备中。特别是Linux在机顶盒和Android手机及平板设备的广泛使用，使其已经成为移动互联时代的主流计算平台。因此，Linux本身及其搭建的应用平台的安全性问题变得越来越值得研究。

在过去的十多年间，Windows的安全性问题更为人所关注，Linux通常被人盲目地认为是更加安全的操作平台。但是实际上没有绝对安全的操作平台，Linux也有自身的特点和伴随的脆弱性。分析和测试linux的脆弱点，并建立完整的解决方案是一项富有挑战性的工作。

Linux支撑的应用很广，不可能面面俱到。本文就Linux安全性测试中的一些要点进行分析，以抛砖引玉。

1. Linux访问控制

Linux的访问控制传统上不依赖于访问控制列表（ACL），虽然2.6.x内核加入了对ACL的支持，但ACL却不是内核的标准组成。这意味着Linux的绝大多数版本仍然依赖用户/用户组模式和文件的访问权限位。如果当某个可执行文件访问权限位设有setuid位（s）时，该文件启动的进程可以执行setuid操作，从而将进程的euid提升为root。通常恶意程序可以通过对LD_LIBRARY_PATH和LD_PRELOAD环境变量进行预加载设置，指向工作路径为拥有完全访问权的路径上的可执行模块。预先加载的恶意模块可以执行setuid。比较新的Linux内核在启动拥有setuid位的进程时禁用LD_LIBRARY_PATH和LD_PRELOAD环境变量加载设定的模块，但是在一些嵌入式Linux版本中仍然存在问题。如果用户在可执行文件主程序中使用了绝对路径的共享库，则以上安全问题在最新内核中依然暴露出来。稳妥的办法是应用程序自身代码中提供对模块加载路径的检查。禁止环境变量的使用并采用dlopen方式加载用户共享库。在设计安全性测试方案中，应包含对用户程序共享库加载方式的检查。

Linux的Capability功能提升了可执行文件安全，安全权限设定可针对可执行模块进行，并对父进程和子进程进行完全独立的设置。这样较传统方式有效地保证了执行程序的访问控制是安全合理的。但是Capability也有自身问题。例如当一个父进程启动一个设置Capability的子进程（权限高于父进程）后退出执行。此时子进程对系统资源的访问可能不被限制。因此对于使用了Capability的程序，应对程序的逻辑流程进行检查，确认对Capability的处理是否得当。

2. Linux程序缓冲区溢出

缓冲区溢出是软件系统遭受的最主要的安全威胁之一。缓冲区溢出的典型实例就程序缓冲区不做边界检查就接受任意长度的数据输入。如果数据放在栈里，内嵌在数据中的攻击代码可以改写返回的调用地址。如果数

据放在堆里，内嵌在数据中的攻击代码也可以更改诸如C++对象的虚函数表等变量的值。然后执行攻击者自己的代码，取得对系统的控制权。

考虑到部分Daemon以root权限运行，攻击者可能通过缓冲区溢出攻击获得root权限。这不符合系统安全性要求。Linux的Daemon应该调整为较低权限运行。在对基于Linux应用平台的安全性测试中，对root权限运行的Daemon应该充分进行扫描，确保为独自的安全账户运行。对普通用户程序也可这样。

避免缓冲区溢出的主要办法是在数据区（如堆栈）设立警示区。如果警示区的数据被改写，程序就会转入异常处理。对于开源的Linux来说，效果较之Windows和Macx系统差。因为Linux内存分配和Gcc编译器都是公开的，攻击者很容易构造欺骗警示区的代码。K9和Coverity等静态检查工具可以对程序缓冲区处理进行检测，发现访问越界缺陷。但是这些工具的算法暂时无法检验数据结构（如数组和网络接受端的buffer）内部的数据是否遭到非法覆盖。

一个可行的解决办法是对程序进行二进制代码级别的静态测试。采用VerOCode等工具或人工分析，及时发现代码缺陷点并修复缺陷。根据测试理论，发现代码缺陷的区域往往存在更大的新发现缺陷的可能，因此进一步的工作还可以采用代码反跟踪技术，对存在缺陷的代码区进行加密，以避免被反汇编和调试。

针对Linux开源平台的特点，可以充分运用代码静态检查工具进行源代码级和目标级代码测试。这些工具可以直接分析源代码，辅助人工以减少代码检查的时间。主要的工具有：

Frama-C：开源测试工具，采用形式语义技术，主要针对嵌入式Linux代码进行测试。

QAC：采用SMT建模技术，针对所有平台的源代码测试。类似的有Coverity和Caveat。

VerOCode:针对源代码和二进制代码均可以进行测试，主要发现代码安全缺陷。

3. Linux下WEB程序

Linux很早就是作为Web服务器承载平台使用。来自互联网上的大多数威胁针对Web服务器。同其他服务相比，Web服务不能替代而且主流的防火墙也不能对Web请求进行拦截。因此Web服务的安全更多依靠软件本身来保障。Web2.0是以创作分发为核心的Web服务，较之单向数据获取方式的传统Web，更复杂也更脆弱一些。

Http请求依赖于正则语法。由于正则语法和引擎的复杂性，应用程序和Web服务器在url请求分析上一直存在潜在缺陷，容易受到黑客攻击。Web2.0下可采用更隐蔽的url攻击。正则表达式DoS就是DoS攻击发展演变的重要方向。除了对正则引擎做前面提到的静态测试外，对应用程序中正则表达式本身的使用也要进行复杂的模拟测试。模拟可使用工具进行，如RegexBuddy。

Flash for Linux也是一个巨大的安全隐患。同Windows相比，所有Linux平台的Flash代码都非常差，拥有极高的代码隐患。如果用Adobe Actionscript开发的可移植应用，不经过测试在Linux上直接使用，通常都不能安全稳定的使用。但是由于flash前景不佳，所以缺乏测试工具对flash的网络程序进行测评。

SeLinux在服务器上可以保障Linux服务的安全，但对嵌入式系统来说只能依靠替代解决方案，如Android自己的沙箱模型。

对于应用程序本身，下载文件的处理也是关键。没有沙箱模型的情况下，默认文件存放在低权限用户区，并设定为不可执行。对没有本地验证签名的组件应该限制其运行权限。这些都是进行安全性测试应该重点考虑的方面。

4. 结束

本文简明地探讨了Linux安全性测试中值得关注的一些要点。限于篇幅，不能全面阐述，比较精深的内容只有另文描述或参考相关著作。

FPGA测试技术

P33 武器装备可编程逻辑器件应用现状分析
P37 型号FPGA应用现状、发展趋势及建议
P40 FPGA验证中静态时序分析的运用
P43 VMM搭建FPGA自动验证环境
P46 动态仿真中代码覆盖率和功能覆盖率
P49 提高FPGA复位设计可靠性的建议

武器装备可编程逻辑器件应用现状分析

文 | 北京京航计算通讯研究所 张国宇、于林宇、刘伟、陈国敏

摘要： 以各型军用飞机、导弹为代表的武器装备作为高技术集成系统，正朝着高集成度、高可靠性、高精度等方向发展。可编程逻辑器件由于具有集成度高、体积小、功耗低、速度快等诸多优点，在武器装备中获得了广泛应用。本文调研了国内可编程逻辑器件的应用现状，分析了当前可编程逻辑器件在我国武器装备应用过程中存在的问题，并总结了国内外可编程逻辑器件应用的成功经验，供各相关单位参考。

关键词： 可编程逻辑器件；应用现状；存在问题。

1、引言

可编程逻辑器件作为一种可编程使用的半定制芯片，采用硬件软件化的设计方法，通过硬件描述语言的方式配置器件内部的逻辑功能和输入/输出端口，可以将原来电路板级的设计放在芯片中进行，与传统的电路板级设计相比具有以下优势：

1. 降低产品的综合成本，用可编程逻辑器件设计和改进电子产品可大幅度减少印刷电路板的面积和接插件，降低装配和调试费用；

2. 提高了系统可靠性，大量中、小规模集成器件在向印刷电路板上装配时，往往会由于虚焊或接触不良好而造成故障，并且这种故障常常难以发现，给调试和维修带来了极大的困难。可编程逻辑器件具有很高的集成度，很多功能可以在一片芯片上实现，有效的避免了上述缺点，大大提高系统的可靠性；

3. 降低了系统功耗，可编程逻辑器件内部电路尺寸很小，互连线短， 分布电容小，驱动电路所需的功耗就大大降低；

4. 提高了电子产品的工作速度，芯片内部很短的连线能大大缩短延迟时间，并且内部电路不易受外界干扰，非常有利提高速度；

5. 可编程逻辑器件由于具有较高的集成度，大大减小了电子产品的体积和重量；

6.通过在系统编程、远程在线重构等技术降低维护升级成本，使得可编程特点突出，更改设计更加灵活。

与ASIC(Application Specific Integrated Circuit)相比，可编程逻辑器件具有显著的优势：开发周期短、前期投资风险小、产品上市速度快、市场适应能力强和硬件升级空间大；当产品定型和扩大产量后，在可编程逻辑器件中实现的设计也可迅速定制为专用芯片进行投产；在新工艺节点上用可编程逻辑器件转换和重新实现已有ASIC 产品，将使产品的升级更容易。

与通用DSP(Digital Signal Processing)器件相比，可编程逻辑器件利用并行架构实现DSP 功能，在不少应用场合性能可超过通用DSP处理器的串行执行架构。同时由于具有现场可编程和在线可编程特性，十分有利于产品的更改、升级。

同时由于可编程逻辑器件上集成了DSP微处理器等专用计算单元，其在实现硬件电路功能的同时，可以部分实现软件功能，在信号处理、图像压缩处理等领域已经成为首选方案。由于其运行方式为并行处理，可编程逻辑器件实现上述功能时，相对于软件的顺序处理方式具有更快的处理速度。

由于具有上述优势，可编程逻辑器件非常适用于小批量、多品种的军品设备的研制，在航空、航天等武器装备领域中得到了广泛的应用。

2、国外应用现状举例

美国、欧洲及日本等现已广泛使用可编程逻辑器件开发航天航空型号产品。在航天领域，可编程逻辑器件已经广泛应用在美国的空间探测卫星、机遇号火星车、标准3导弹、X-37B空天飞机、欧洲阿丽亚娜火箭及日本的H2A火箭等型号中；在航空领域，可编程逻辑器件更是成为绝大多数服役和在研型号的重要组成部分，已广泛应在法国达索公司的阵风战机、瑞典萨博公司的鹰狮战斗机、美国诺斯洛普•格鲁曼公司的B2隐身战略轰炸机、洛克希德•马丁公司的F22A、F35隐身战斗机等型号中。

3、国内应用现状举例

国内方面，可编程逻辑器件由于设计开发灵活、可靠性高等特点，可以方便的实现各种数字电路功能（如FFT算法、通信等）及部分软件功能（如控制、校验和等），已广泛应用于民用领域及军用领域，并发挥着重要作用。在民用领域方向，探月工程中的嫦娥三号卫星的大量关键功能（如有效载荷系统）是由可编程逻辑器件实现的，其上的可编程逻辑器件的数量已经大大超过了软件的数量，载人航天工程中的神舟八号飞船的控制单元、数据处理等系统同样是由可编程逻辑器件设计实现的。在军用领域方向，可编程逻辑器件应用在空军多个在研的武器装备中，实现制导控制系统、导引系统、动力系统、电气系统等各系统的综控机数模转换控制、开关控制、雷达脉冲压缩控制、目标压缩算法、发动机数字综合控制器控制、底层通讯等大量关键功能。部分型号可编程逻辑器件的数量已经超过了软件的数量，例如某型导弹在研制过程中的软件数量与可编程逻辑器件数量的比例在0.45：0.55左右。

4、当前存在的主要问题

虽然可编程逻辑器件已广泛应用在各型武器装备中，完成了大量关键设计，但通过对已开展测试情况进行分析，可编程逻辑器件的质量形势不容乐观，以某单位2011年5月至2012年2月测试情况为例，共有7型号武器装备可编程逻辑器件参加测试，总设计规模为39601行，共发现829个问题，其千行代码错误率为20.9‰，该错误率超过了软件的平均千行错误率。

当前国内武器装备可编程逻辑器件研制过程多数还是处于失控无序的状态，未进行有效的工程化管理及测试，存在较多的问题，具体如下：

4.1缺少相关研制标准和规范用于指导开发过程

由于各行业还没有颁布权威的可编程逻辑器件开发与管理规定，型号可编程逻辑器件开发处于粗放状态，各家单位、设计师完全按自己的想法进行设计和编程调试，对设计和测试的技术要点理解不够深入，重编码和调试，轻验证，同时存在部分参照软件要求进行开发的

情况。

4.2 工程化管理有待加强

目前，各行业可编程逻辑器件未进行有效管理或参考普通软件的管理，在可编程逻辑器件工程化管理方面存在一定的差距，主要表现以下几方面：

1.产品管理方面

未将可编程逻辑器件作为产品单独进行管理。可编程逻辑器件绑定于型号硬件设备中管理，多数单位未按照型号系统、分系统、设备等不同层次给出可编程逻辑器件配置项名称及代号，未形成“可编程逻辑器件配套表”。

2.文档管理方面

大部分单位主要参照软件文档编制规范，缺少体现型号可编程逻辑器件设计要求的文档编制规范，也未针对可编程逻辑器件出具独立的文档；个别单位在可编程逻辑器件文档完整性方面比较重视，出具了独立的任务书、需求及设计文档，但文档内容简单，难以作为设计或测试依据，且文档签署不规范。

3.评审管理方面

随硬件产品进行计划和相关评审，没有专门针对可编程逻辑器件开展评审，缺少评审技术要求。由于评审的不完善，可编程逻辑器件开发过程缺乏有效控制。

4.计划管理方面

可编程逻辑器件绑定于设备和硬件产品中进行计划，未进行单独的计划管理。

5.配置管理方面

目前多数单位将软件配置管理（“三库”）应用于可编程逻辑器件，未针对可编程逻辑器件特点建立配置管理规范，导致可编程逻辑器件配置管理不完善，如配置项中缺少综合约束文件、综合及布局布线后的网表和标准延时格式文件等，技术状态控制不完备。

6.质量管理方面

未将可编程逻辑器件质量控制工作纳入型号质量保证大纲；未明确测试要求；未建立故障报告、分析和纠正措施制度；未使用正版开发工具；未建立产品证明书制度。

7.人员管理方面

由于缺少人员培训、考试和资质认可的要求，一些单位从事可编程逻辑器件的开发人员为硬件工程师或软件开发人员，缺少可编程逻辑器件设计的专业知识，对硬件描述语言和电路实现原理的理解还不够深入。

4.3缺少完善的可编程逻辑器件测试与验证手段

可编程逻辑器件未开展测试或只进行了内部调试、验证工作，无法有效保证产品质量。当前部分研制方开展的内部测试验证一般是基于功能仿真、板级或系统级验证，大部分研制单位均不开展覆盖率仿真分析，基本不开展静态时序分析和时序仿真，无法保证测试的质量和充分性。

5、国内外值得借鉴的经验

正是意识到缺少可编程逻辑器件工程化管理规定和未开展有效的可编程逻辑器件测试已经严重影响了科技工程或型号研制工作的质量和进度，目前国内外在上述两方面已开展了大量工作。

5.1美国、欧洲建立了比较规范的管理体系

国外十分重视可编程逻辑器件工程化管理和测试验证工作，在机构设置、标准规范制定方面做了大量工作。

1. 成立了专门的保障机构

机构设置方面，美国空军早在2003年，就在新墨西哥的柯特兰（Kirtland）空军基地成立了可编程逻辑器件任务保证中心，用以开展可编程逻辑器件方面相关工作。

2. 制定了相关标准规范

在航空标准规范方面，与软件项目的RTCA/DO-178B《航空机载系统和设备软件保证》相对应，针对可编程逻辑器件项目，美国借鉴上个世纪90年代早期波音（Boeing）和美国航空联邦管理局（FAA）在777飞机项目上的合作的经验，于2000年制定了航空标准RTCA/DO-254《美国机载电子器件设计指南》，该标准描述了机载电子器件全生命周期的目标要求，规定了全生命周期定义和阶段的划分，针对器件全生命周期提出机构职

责、计划、每个阶段的目标及技术要求，尤其注重对于器件的设计验证的要求。研制过程中，依次进行计划建立、需求分析、概要设计、详细设计、确认与验证等阶段，各阶段严格开展评审。

从2005年起，DO-254在美国航空领域的主要军工企业得到强制采纳，例如雷神公司（宙斯盾反导系统标准3导弹制造商）、诺斯洛普•格鲁门公司（B2隐身战略轰炸机制造商）及洛克希德•马丁公司（F22A、F35隐身战斗机制造商）等，作为业界开展可编程逻辑器件研制开发工程化管理的指导性文件。

目前，该标准在欧洲已经被欧空局、瑞典萨博公司（鹰狮战斗机制造商）等多家机构和企业采纳。

在航天标准规范领域，欧洲航天局于2008年6月17日发布了标准号为ECSS-Q-ST-60-02a的《空间产品FPGA和ASIC研制保证》，对可编程逻辑器件的全生命周期进行管理。

5.2国内重大科技工程获得成功应用

航天系统的两大集团公司已经形成了比较完备的可编程逻辑器件工程化管理体系和测试管理体系。航天科技集团公司要求火箭、卫星、飞船等系统可编程逻辑器件研制过程严格执行工程化管理规定，所有关键、重要的可编程逻辑器件必须经过第三方测试验证。航天科工集团制定了Q/QJB 179-2010《可编程逻辑器件软件工程管理要求》，并贯彻到各型号研制过程中，成立了从事可编程逻辑器件工程化管理工作和测试工作的专门机构——航天三院可编程逻辑器件评测中心，有效的保证了型号研制的质量。航天系统加强可编程逻辑器件工程化管理的措施，在重大科技工程中得到了成功实践。

在探月工程一期“嫦娥一号”卫星研制期间，没有将可编程逻辑器件纳入技术状态管理，随设备一起交付使用，在地面分系统试验、系统联试及运行期间，由于可编程逻辑器件受外界环境因素影响较大，多次发生通讯故障、信号翻转、异常复位等可靠性问题。针对此类问题，工程中心以及工程两总在“嫦娥二号”的研制工作中，委托航天三院可编程逻辑器件评测中心编制并发布实施了《可编程逻辑器件项目开发实施细则》，并据此加强过程监督检查、阶段评审和验收，特别是明确要求对可编程逻辑器件独立于软件进行技术状态控制，必须经过有资质的第三方评测机构测试后，方可交付使用。由于严格执行上述措施，“嫦娥二号”卫星在运行期间，所有可编程逻辑器件运行正常，未发生故障。

6、总结

本文概述了可编程逻辑器件在我国武器装备广泛应用的原因，介绍了可编程逻辑器件在武器装备中的应用现状，分析了当前存在的问题，并从工程化管理及测试的角度介绍了国内外可编程逻辑器件应用的成功经验，对相关单位开展武器装备可编程逻辑器件研制应用具有较大借鉴意义。

参考文献

[1] 朱为. FPGA 器件及其应用[J]. 电子器件,1996,19卷

作者简介：

张国宇，男，北京京航计算通讯研究所，硕士研究生、控制理论与控制工程专业、技术员、工程师、邮编100074，E-mail: xueshen0175@163.com

型号FPGA应用现状、发展趋势及建议

■ 文 | 航天软件评测中心　冯志华　陈丽容

摘要：针对FPGA的特点，分析了现阶段型号FPGA软件的管理现状、研制现状和测试现状和发展趋势，并从型号FPGA软件文档要求、测试方法、设计评审、配置管理等方面，结合工程实践给出了型号FPGA软件研制、测试和管理方面的建议。

1. 概述

可编程逻辑器件（下文简称FPGA）已成为解决型号低功耗、高性能、高可靠、超小型化难题的关键技术。据有关资料显示，空客A380飞机共使用716片FPGA芯片，2006年美国空军F-22A猛禽战斗机综合核心处理机等设备中均大量使用FPGA芯片。我国自从2003年后，卫星型号大量使用FPGA器件，其中27颗卫星使用了约902片FPGA，在轨的27颗卫星中有5颗卫星共发生24次FPGA软件设计等问题。某武器型号弹上和地面设备使用SRAM、反熔丝、FLASH型3种类型合计100余片FPGA芯片，其中约20%故障与FPGA软件设计相关。随着FPGA芯片的广泛应用，其暴露出的质量问题正引起各方的高度关注。2000年4月航空无线电委员会（RTCA）发布《机载电子设备设计保障指南》，即DO-254标准，目前已成为美国联邦航空局（FAA）、欧洲航空安全局（EASA）、中国民用航空总局（CAAC）与世界其它航空认证机构在航空行业内强制执行的电子设备设计标准。如何针对FPGA软件特点，开展质量控制，已成为保证型号质量与可靠性所面临的紧迫问题。

1. FPGA的特点

FPGA芯片的种类繁多，常用于型号电子设备的FPGA芯片主要有Xilinx公司、Altera公司和Actel公司的FPGA芯片。Actel公司的基于反熔丝技术的FPGA芯片具有一次烧写编程（One Time Program）、无需配置过程和抗辐照等优势，但逻辑资源相对较小，适用于星载、弹载等高可靠环境下的电子设备中。Xilinx公司基于查找表技术，SRAM工艺的FPGA芯片，可在线重复编程，但需要一定的上电配置加载时间，提供宇航级芯片，支持片上嵌入式处理器和三模冗余（TMR）开发工具，由于其灵活性在型号电子设备中也使用较多。Altera公司基于SRAM工艺的FPGA芯片，其开发工具简单易用，可在线重复编程，目前只有扩展工业级芯片，在型号电子设备中也占有一定的份额。其主要优点包括：

（1）大规模的FPGA芯片等效于上千万门的逻辑单元，可以实现电子设备的小型化和集成化；

（2）FPGA与专用集成电路（ASIC）相比，开发周期短，开发软件投入少，开发成本低；

（3）FPGA的时钟延迟可达纳秒级，可以满足电子设备中高速处理和实时测控等方面应用；

（4）FPGA芯片可选择的种类很多，可根据不同的应用场合选择不同类型的芯片，以用于不同的电子设备；

（5）FPGA软件设计可采用硬件描述语言编程，其软件代码具有可移植性和可维护性；

（6）FPGA具有动态重配置的功能，可以动态刷新和改变所需实现的逻辑功能。

2. FPGA应用现状分析

2.1. 管理现状

2006年5月，航天二院型号颁布《可编程器件软件文档编制及管理要求》，针对软件文档等方面进行了相关要求。2008年起，国内针对921工程、探月工程等重大工程开始制定FPGA软件研制与管理要求，但各院对FPGA开发过程认识不同，主要表现在：1）IP（知识产

权核）问题，有的单位对IP 进行了一定的验证，而有的单位未进行相应的验证。2）验证过程不统一，有的单位第三方验证过程中包括板级验证，而有的单位没有包括板级验证。2008年12月，五院发布《航天器用FPGA产品研制管理要求》和《航天器用FPGA产品研制技术要求》。2011年1月，科工集团发布《可编程逻辑器件软件工程管理要求》。2012年2月，二院发布《二院型号FPGA软件设计实施规范》。

目前，国内还没有发布FPGA软件研制和管理方面的国军标和行业标准，各方对FPGA软件的研制管理尚处于起步阶段，现行的标准和规范加强了FPGA软件的研制管理，但是仍存在以下问题：1）缺少详细的技术指导规范；2)尚未形成FPGA软件测试具体要求;3)缺少有经验的FPGA软件开发、测试和管理人员；4）系统中FPGA设计人员和其它设计人员分工不明确；5）尚未建立FPGA软件培训体系。

2.2.FPGA软件研制现状

（1）开发过程

FPGA软件开发过程一般包括：需求分析，详细设计，设计输入（编码），功能仿真，逻辑综合，布局布线，时序仿真和时序分析，系统测试等过程。目前大部分单位对FPGA软件均开展功能仿真，但是很少开展时序分析、代码覆盖率分析和约束设置，设计师不太关注设计输入代码和最终硬件电路实现的一致性，尚沿用传统软件编程的思维观念。

（2）IP核复用

由于FPGA软件系统功能需求多，接口通用性强，因此研制单位在FPGA软件需求分析和开发中，一般会选择较为成熟的IP核或商用IP核进行集成，从而缩短开发周期，但会造成三方面问题：一是对IP核复用调研和论证不充分，集成后发现此类问题但又难于更改；二是复用或购买的IP核有缺陷无人维护；三是IP核的问题在某个分系统中暴露和更改后，没有举一反三，同步更改其它FPGA软件。

（3）内部测试

各单位设计水平参差不齐，研制阶段过程中对FPGA软件的内部测试不充分，很少开展编码规则检查、模块级功能仿真、跨时钟域检查和风险分析。

（4）配置管理

目前FPGA软件配置管理基本沿用传统软件配置管理办法，一般都建立了“三库”，但是由于FPGA软件开发工具种类多，版本多，FPGA软件版本与开发工具关联性强，加上现阶段各单位配置管理实施办法没有明确入库要求，在型号FPGA软件归零和FPGA软件版本升级时，缺少相关的工具和文件。

（5）评审过程

目前FPGA软件评审主要参照传统软件的评审过程，研制方一般进行需求分析、详细设计报告评审和交付验收评审，但是缺乏对设计输入（编码），功能仿真，逻辑综合，布局布线，时序仿真和时序分析等过程的技术评审。

（6）文档问题

有些单位将FPGA软件开发工作纳入到型号硬件设备中统一管理，FPGA软件没有独立任务书，FPGA软件需求规则说明书较空泛，设计与需求难于对应，FPGA软件详细设计报告出现文实不一致的情况较为普遍。

2.3.FPGA软件测试现状

（1）质量与进度

一般型号FPGA软件测试周期通常在2个月左右，但是由于以往FPGA软件没有纳入型号FPGA软件测试计划，各型号对FPGA软件测试的要求也各不相同，各家评测机构FPGA软件测试人员相对缺乏，随着型号FPGA软件提交评测和交付的时间日益缩短，导致型号FPGA软件测试的质量和进度不协调。

（2）测试方法

由于FPGA软件测试处于起步阶段，目前没有硬性规定FPGA软件测试类型和测试方法，有的单位开展编码规则检查和功能仿真，有的不做时序仿真，而采用静态时序分析和逻辑等价性验证来替代时序仿真工作，有的单位还对FPGA软件进行板级验证。各家评测机构采用的测试方法不尽相同，对测试质量难于评估。

（3）测试环境

FPGA软件设计与测试工具种类多，价格昂贵，导致现阶段各单位FPGA测试条件差异较大，其次，由于FPGA对外接口复杂，对涉及DSP和FPGA等紧耦合的场合或扩频信号处理算法，难于在基于PC机仿真环境下开展FPGA软件的仿真测试。

（4）被测件问题

主要问题有：a）研制单位出于对单位知识产权的保护，不愿意将FPGA软件技术文档提交给测评单位；b）研制方为了防止核心代码外泄，将其编译转化为IP核网表后提交给评测机构，这给评测方开展人工走查，代码覆盖率等测试带来很大不便。

3. 发展趋势及面临的挑战

随着微电子技术的不断发展，未来FPGA发展趋势如下：

（1） 高密度、高速度、宽频带；

（2） 低电压、低功耗；

（3） 抗辐照、容错

（4） 结构化ASIC；

（5） 动态重构；

（6） 基于高层次综合的高级语言编程；

（7） 片上SOC/SOPC；

（8） 数模混合FPGA；

（9） FPGA专用的设计、验证EDA工具。

但是，随着半导体工艺的发展，FPGA颗粒越来细，这给基于高层次综合的、数模混合的、高密度、低功耗、高性能的FPGA软件的测试带来严峻挑战。此外，由于进口FPGA保密性和可控性相对较差，而目前信息化装备中90%以上依赖进口FPGA芯片，高等级FPGA芯片的禁运已经制约了我国信息化装备的发展，为了满足装备的国产化发展的需要，信息化装备开始逐步采用国产FPGA芯片和全定制ASIC芯片，但是由于现阶段国产FPGA芯片开发和验证工具不完善，国产FPGA的稳定性和可靠性还未得到充分考核和验证。

4. 对策与措施

通过对型号FPGA应用现状的分析，对FPGA软件质量控制建议如下：

（1）进一步重视FPGA软件文档的规范性

在文档的完整性方面比较重视，但在内容的把关上还有待加强，主要是内容过于简单，或是文档不能及时更改，造成文文不符、文实不符的现象。

进一步落实FPGA软件管理要求，加强FPGA软件设计评审，规范文档编制要求，提高文档质量。

（2）加强FPGA软件配置管理和评审

各单位FPGA版本控制要求不统一，缺乏明确的FPGA软件配置管理办法，没有对入库FPGA软件的文件和文档进行有效标识， FPGA软件开发过程缺乏有效控制。建议型号后续研制加强FPGA软件版本控制，明确FPGA软件版本入库管理办法，针对FPGA软件研制过程开展各关键阶段的技术评审。

（3）运用先进的EDA技术和手段

应采用先进的测试工具实现故障Debug的自动化；应用先进设计和验证理念，如采用基于Team-based设计方法，基于UVM/OVM/VMM验证方法学、基于ESL的高层综合和验证方法，基于TLM的设计方法，基于Assertion等形式验证方法。此外，国产FPGA器件软件的测试验证工具和手段尚不健全，国产FPGA芯片供应商、FPGA用户和第三方EDA工具厂商需加强沟通与合作。

（4） 建立型号FPGA软件质量信息数据库

为了提高型号产品质量水平，建议FPGA软件质量信息数据库，让型号FPGA软件人员能信息共享，从而提高型号FPGA软件的整体水平。

FPGA验证中静态时序分析的运用

文 | 上海旋极技术部嵌入式产品经理 马翔

当FPGA设计布局布线后，如何确保设计中的每条路径都不违反寄存器的时序规格？一般需要分析的时序指标是建立时间和保持时间，这时Synopsys的PrimeTime是最合适的。PrimeTime是一个全芯片静态分析工具，能在短时间内完全分析几百万门的设计。不需要测试向量来激励关键路径，对设计使用激励总是不能遍历所有可能的路径，STA方法能覆盖所有有关键路径。

在ASIC设计流程中，为了减少流片次数，PrimeTime被广泛使用。PrimeTime与DC的许多命令是一样的，所有PrimeTime在ASIC设计流程中使用方便，设计师不需要花费太多时间学习。

PrimeTime支持多种文件格式，这也使PrimeTime能够灵活地适应到复杂的设计流程。PrimeTime能够读Verilog、VHDL、EDIF网表（2006年以后的PT不再有读edif的能力），以及多个延迟文件格式，如SDF、SPEF。对于标准单元和宏库，PrimeTime使用数据库db文件确定通过单元和传输到输出引脚的延迟。PrimeTime还需要SDC文件来定义设计约束。

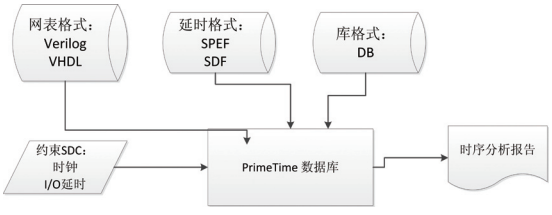


图1 PrimeTime输入输出图

一、STA方法检查时序冲突

PrimeTime在分析设计时，它会检查设计中的每个路径的setup和hold是否满足设计库的规格，它会对每个路径计算两次时间，确保路径在上升沿和下降沿都不会出现setup和hold冲突。

建立时间（setup time）是指在触发器的时钟信号上升沿到来以前，数据稳定不变的时间，如果建立时间不够，数据将不能在这个时钟上升沿被打入触发器；保持时间（hold time）是指在触发器的时钟信号上升沿到来以后，数据稳定不变的时间。

设计中每个路径是从一个寄存器到另一个寄存器，包括输入路径和输出路径。通过组合逻辑的时间必须小于时钟周期减去时钟不确定性、FF建立时间，以及上升沿时数据到Q引脚到时间。时钟不确定性是发起FF到捕获FF的时钟偏移。

$$T_{clogic} < (T_c - FF_{lauch(clk->Q)} - T_u - t_s)$$

（式1）

其中组合逻辑延时是通过单元的延时加上通过网线的延时。

当数据路径相对于时钟速度很慢时，会引起建立冲突。有多种方式解决建立冲突：

- 减少路径中缓冲器的数量
- 用两个更远的反相器替换缓冲器
- 降低时钟速度

二、为STA分析准备：时序约束

时序约束能够告诉PrimeTime设计的时序行为。有三个主要约束是必须定义的：时钟、输入延时和输出延时。时钟周期要求寄存器到寄存器的延时在该周期内。

输入延时定义从前向芯片的输出通过端口输入到一个寄存器的时间。输出延时是从一个寄存器的输出通过端口输出到后向芯片的输入。这三个约束将设计行为定义给PrimeTime。

1. 关于时钟

在PrimeTime内定义时钟需要提供输入端口、时钟频率以及时钟周期。设计经过布局布线后，时钟端口不再直接连接到FF，PrimeTime必须知道衍生时钟。如下定义一个2ns周期的时钟：

```
pt_shell> create_clock -period 2 [get_port clk]
```

如果内部时钟是通过计数器分频得到，必须将时钟定义在实例的引脚上：

```
pt_shell> create_generated_clock -name clock1 -source clk -divide_by 2 [get_pins FF1/Q ]
```

增加额外的裕度是一个很好的习惯，能够提高设计稳定性，命令set_clock_uncertainty将在计算每条路径的setup和hold时增加延时。对于一些FPGA芯片质量不能保证的货源，可考虑增加它们的裕度：

```
pt_shell> set_clock_uncertainty .1 [all_clocks]
```

在PrimeTime内，所有的时钟总是理想的，但实际情况下布局布线后时钟线上会有延时。set_propagated_clock命令要求PrimeTime实现各个时钟路径的延时。

```
pt_shell> set_propagated_clock [all_clocks]
```

2. 关于输入输出时序

PrimeTime假定数据信号在时间0到达端口输入，也不约束所有的数据路径。命令set_input_delay和set_output_delay是分别用来约束输入端口和输出端口。

set_input_delay定义信号从外部逻辑到端口输入的时间延时。在使用这个命令时必须指定一个时钟信号与输入信号相关，PrimeTime将自动计算信号通过内部逻辑花费的时间。

```
pt_shell> set_input_delay 2 -clock clk [get_port A]
```

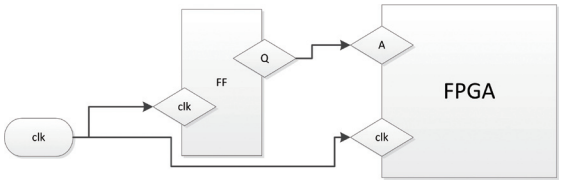


图2 输入延时

输出路径通过set_output_delay来约束，表示外部逻辑需要的总延时。PrimeTime将自动计算信号在内部逻辑中传播的时间延时。输出信号也必须参考一个时钟信号才能计算建立时间和保持时间。

```
pt_shell> set_out_delay 2 -clock clk [get_port A]
```

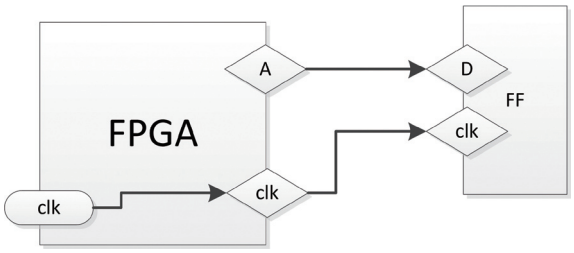


图3：输出延时

3. 关于false path

设计中总有一些时序路径不希望计算，这些路径与电路的功能无关，或者永远不会执行，又或者是异步时钟域的。

以下图为例，两个多路器被分时选通，此时多路器1的输入1不能通过多路器2的输入1到达多路器2的输出D，同理多路器1的输入2不能通过多路器2的输入2到达多路器2的输出D。此时这两条路径可以被移除。

```
pt_shell> set_false_paths -through mux1/a1 -through mux2/s1
pt_shell> set_false_paths -through mux1/a2 -through mux2/s2
```

当数据是从一个时钟域被发起，在另一个不同的异步时钟域被捕获时会引起亚稳态。这种潜在故障将被送往下一级寄存器，并引起错误往深处传播，解决办法是插入同步单元。使用以下命令让PrimeTime不检查下图这种跨时钟域的路径的setup和hold。

```
pt_shell> set_false_path -from clk1 -to clk2
pt_shell> set_false_path -from clk2 -to
```

clk1

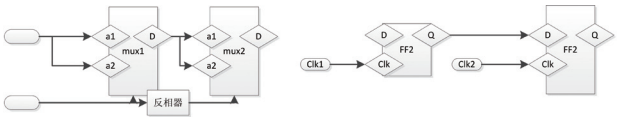


图4 设置了false path的电路对比

4. 关于多周期路径

多周期路径是指一个信号传播到终点的时间超过一个时钟周期。这时必须指定多少个周期之后信号被捕获。PriemTime必须被配置为在捕获沿之前检查保持冲突N个周期。

```
pt_shell> set_multicycle_path -setup 5 -to [get_pins FF/D]
```

三、产生TimeTime报告

报告是从PrimeTime数据库中产生的。PrimeTime有很多命令报告建立、保持以及设计规则冲突。为了减少PrimeTime报告的规模和复杂度。建议将设计分为组，一般可分为4个组：

- 输入到寄存器
- 寄存器到寄存器
- 寄存器到输出
- 输入到输出

每个路径从起点到终点，合法的起点是输入端口和寄存器的时钟引脚，合法的终点是输出端口和所有寄存器的输入口(包括使能等引脚)，除了时钟引脚。由于all_register包含所有的寄存器，因此报告输入到寄存器、寄存器到寄存器、寄存器到输出可以使用以下命令：

```
pt_shell> report_timing -from [all_registers -clock_pins] -to [all_registers -data_pins]
```

report_timing是非常有用的命令，它总是报告约束条件下设计的关键路径。具有大量参数：

- -from -to 报告将被约束在指定起点和终点的路径
- -path full_path 不止报告数据路径，还报告激发和捕获时钟路径。此时必须有set_propagated_clocks设置过
- -delay max -delay min 根据这个参数决定计算setup时间还是hold时间。max计算所有setup时间的路径。min是hold的。
- -max_paths -nworst 这两个选项控制报告中打印的路径数量。由于设计中可能有成千上万条路径，实际只要看引起失败的关键路径。-max_paths表示每个组报告的路径总数，默认为1，-nworst报告终点的数量。

四、TCL脚本：让分析更简单

PrimeTime通过脚本语言Tcl（Tool Command Language）进行控制，在Tcl下PrimeTime提供了一些非常有用的变量。运行PrimeTime的推荐方法是创建一个脚本，而不是将每条命令输入到提示符下。

在Tcl下创建变量如下（操作环境RHEL5 BASH）：

```
pt_shell> set home_dir "/home/user"
```

PrimeTime的一个强大特点是Linux环境中的变量能够被传给PriemTime，PrimeTime变量可以通过Linux环境变量初始化。

```
pt_shell>set home_dir $env(HOME)
```

在设计被读入PrimeTime时，相关的变量都会被创建，这些变量将用于设计约束、编写更快和更简单的脚本。在以下例子中：

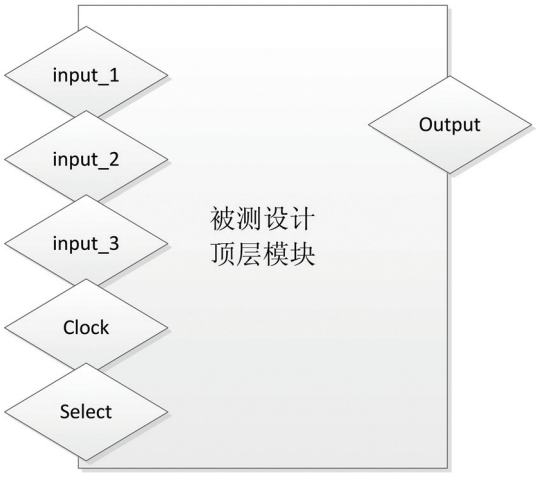


图5 STA流程

设计有5个主输入和1个主输出，以及几个寄存器到寄存器的路径。创建了两个新变量包含设计中所有的主输入和主输出。

（下转48页）

VMM搭建FPGA自动验证环境

■ 文 | 上海旋极技术部工程师 李茜

使用SystemVerilog语言与VMM验证方法学结合的方法搭建验证平台，已经成为FPGA测试的业界标准。利用SystemVerilog面向对象的封装、继承、多态的特性，在加上VMM提供的强大类库，使得FPGA验证具有很好的结构性和重用性，缩短了验证平台搭建的时间，提高了验证效率。并且利用SystemVerilog带约束的随机激励、断言和功能覆盖率等重要的验证特征能够大大提高验证的质量。

1. 引言

FPGA芯片设计的复杂度持续增长，如果继续采用传统的方法，不仅验证工作量大，效率低下，且常常面临测试不全面的风险。因此传统的验证方法已经不再让人觉得安全，为了尽快的弥补验证与现实产品设计之间的巨大差距，验证工程师们一直在寻找一种高效的、可重用的标准验证平台。

Verification Methodology Manual（VMM）是一种高级验证方法学，它以SystemVerilog语言为基础，适用百万门以上的大型设计。这个方法学是由一个标准库支持，包含了一系列的基类和功能类，具有多个通信协议，复杂的故障处理和验证IP库，用来搭建VMM的验证环境和验证组件。SystemVerilog语言中有许多重要的验证特征，适合应用在VMM中，对FPGA验证、SOC验证，甚至是NOC验证都是十分先进的验证解决方法。不仅可以实现一定程度的测试平台自动化，提升了测试平台的可重用性，还能够快速自动地查找许多不易发现的漏洞和错误。

2. SystemVerilog在FPGA验证中的应用

SystemVerilog语言在传统Verilog语言的基础上扩展了多种验证特征，其中功能覆盖率、带约束的随机激励、断言、接口模块等都是具有提高验证效率的语言特征方法。使用SystemVerilog语言对复杂的FPGA设计进行验证，可以在一定程度上提高验证效率。

2.1 接口模块

用Verilog写测试用例时，一般设计的多个模块间的信号传递都是在输入输出中通过名称或参数形式实现，需要在多个模块中对通信口重复声明。测试中，常常出现随时改动的情况，或添加新的信号的情况，这样需要在多个模块改动通信信号。如果设计非常复杂，这种改动会带来很大的工作量，并且非常容易出错。

SystemVerilog语言中具有接口定义的特征，将所有的信号定义在一个模块中，其他的测试模块实例接口模块即可，如图1所示。

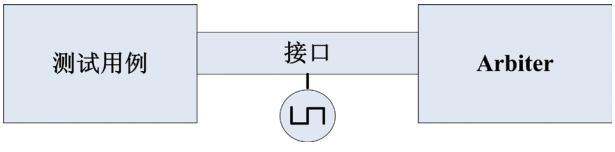


图1：接口定义

如果要对设计进行改动，只需要在接口模块中进行

添加或修改，而不会影响其他模块。并且接口中可以用modport将信号按所在的模块进行分组定义。

2.2 SystemVerilog的验证特征

用SystemVerilog来写测试用例，是将激励、任务、function、覆盖率和断言等内容封装在一个program中。用带约束的随机激励来替代手动输入激励，可以定义成一个单独的class。对数据进行相应约束，可以按验证工程师的意愿产生随机激励。以验证FIFO为例，可以随机产生输入数据和读写操作频率。

```
class random_write_data;
   rand logic [`WIDTH-1:0] data [`ADEPTH];
   rand logic [15:0]  data_rate;
 constraint reasonable {…}
endclass
```

定义覆盖组，也可以定义一个单独的覆盖点，同时采样所有的覆盖点，对产生的数据击中测试点的情况进行统计分析。SystemVerilog的验证方法都是面向对象的编程风格，将需要进行验证的任务或功能写成单独的类、task或function。覆盖组的使用与类相似，经过定义后也可以多次实例化，也可以将覆盖组定义在一个类中。

```
covergroup apb_trans_cov;
    addr: coverpoint tr.addr {
      bins zero = {0};
      bins onek = {1024};
      bins others = default;
    }
endgroup
```

断言是SystemVerilog的重要特征，vcs在库文件中有丰富的checker集，用户可以在测试顶层中直接实例化，实现设计的断言检查，找到bug。

```
assert_fifo #(.depth(128), .elem_sz(16),
.coverage_level_1(31),
.coverage_level_2(0), .coverage_
level_3(31) )
SVA_FIFO_inst (clk, rst_n, !push_req_n,
data_in, !pop_req_n, data_out );
```

SystemVerilog语言与HDL语言比较而言，在FPGA验证方面的优势更为突出。可以实现激励的自动化产生，参数化的接口定义等都极大的提高了验证效率。将高级验证方法学与SystemVerilog语言结合，充分利用了语言的验证功能，也是FPGA验证实现真正统一化、标准化和智能化的重要途径。

3. VMM验证环境搭建

VMM验证方法学提供了基于SystemVerilog的验证方法，具有层次化的验证结构，对所有测试通用的验证平台以及功能覆盖率为驱动的验证流程。充分利用SystemVerilog的面向对象的封装、继承、多态的特点，具有丰富的验证类库，用户能够充分利用类库，迅速建立易于控制的、高度自动化的、可重用的验证平台。

3.1 层次分明的测试

这里介绍一个APB总线系统的验证环境，如图2所示。包括了约束性的随机传输数据生成，自动比对机制，覆盖率搜集，断言检查等组件。

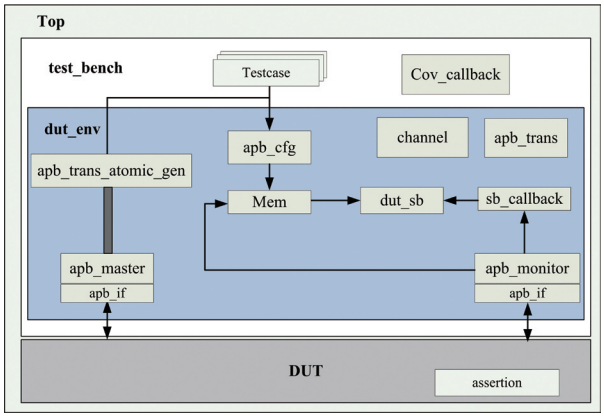


图2：分层的测试平台结构

testcase：包括对环境的配置和DUT的配置；apb_trans_atomic_gen用来产生验证事务；Mem：是DUT的参考模型，作为计分板的输入与DUT进行比对；apb_master：产生事务的时序；apb_trans：产生随机数据；cov_callback：是功能覆盖率模块，用来搜集覆盖点的随机产生频率。

3.2 组件功能

（1）随机事务

apb_trans_atomic_gen发生器在环境中产生了单

独的随机事务。apb_channel用来连接发生器和其他的事务组件。这里用的是vmm_data提供的宏的方法来定义的这两个类。用这种方式可以自动展开事务的方法，节省代码量，使得代码的重用性更高，更加简洁。主要包含enum {READ, WRITE}，addr，data，dir数据成员

（2）驱动器apb_master

APB系统中需要将总线的读写操作展开成具体的指令，通过定义apb_master来实现。apb_master从channel中提取一个apb_trans对象，并在APB总线上执行读写周期。这个类是从vmm_xactor扩展而来，vmm_xactor中包括了总线功能模块，监视模块和generator。apb_master是继承了其中的do_read(), do_write() and do_idle()操作，用来控制总线事务的操作流程。

这里在执行一个事务前后都插入了回调，扩展了一个apb_master_callbacks类，用来明确的划分事务的独立进行，使得事务间的进行不会那么混乱。

```
virtual class apb_master_callbacks extends vmm_xactor_callbacks;
    virtual task master_pre_tx(apb_master xactor,…);   endtask
    virtual task master_post_tx(apb_master xactor,…);    endtask
    endclass: apb_master_callbacks
```

（3）参考模型和接口

参考模型是一个行为级的功能模型，主要完成的是信号读写的控制功能。将系统信号定义成一个接口模块，并用modport进行分类，在mem参考模型中调用接口中的信号即可。

（4）监视器和计分板

监视器和计分板从 vmm_xactor扩展，在监视器中用apb_sample()方法对总线信号进行采样，用out_chan.sneak()方法将执行的事务输出到channel，通过channel在输入到计分板中与参考模型进行比较。

计分板中的主函数定义了一个状态机对读写事务操作进行检查。

```
case(mas_tr.dir)
    apb_trans::WRITE: check_write(mas_tr, mon_tr, exp_data);
    apb_trans::READ:  check_read (mas_tr, mon_tr, exp_data);
    default: `vmm_fatal(log, "Fatal error: Scoreboard received illegal master transaction");
    endcase
```

（5）环境和结果

由于APB系统的事务简单，直接用事务发生器取代场景。验证环境的搭建可以灵活设计，没有特别的规定必须要定义每个层次的组件。功能覆盖率搜集，这里主要是对地址总线和数据总线数据进行采样统计。

定义dut_env类，将所有组件封装其中，并使用扩展 vmm_env。testbench要经过很多执行的阶段，从实例化、仿真到最后清理。 vmm_env能够帮助管理这些步骤，保证所有的命令都会执行。

```
Contains Synopsys proprietary information.
Compiler version F-2011.12; Runtime version F-2011.12;  Oct 31 14:43 2012
VCD+ Writer F-2011.12 Copyright (c) 1991-2011 by Synopsys Inc.
Normal[NOTE] on dut(env) at                    0:
    cfg.trans_cnt =          6
Normal[NOTE] on Scoreboard(Scoreboard) at                  550:
    @550: Starting scoreboard for 6 transaction
Normal[NOTE] on Scoreboard(Scoreboard) at                  750:
    CHECK OK ==>#1.0.0 : Write Addr=0x00 Data=0x64
Normal[NOTE] on Scoreboard(Scoreboard) at                  950:
    CHECK OK ==>#1.0.1 : Write Addr=0x64 Data=0x64
Normal[NOTE] on Scoreboard(Scoreboard) at                 1150:
    CHECK OK ==>#1.0.2 : Write Addr=0x64 Data=0x64
Normal[NOTE] on Scoreboard(Scoreboard) at                 1350:
    CHECK OK ==>#1.0.3 : Write Addr=0x00 Data=0x00
Normal[NOTE] on Scoreboard(Scoreboard) at                 1550:
    CHECK OK ==>#1.0.4 : Write Addr=0x400 Data=0x64
Simulation PASSED on /./ (/./) at                 1550 (0 warnings, 0 demoted errors & 0 demoted warnings)
$finish at simulation time                 1550
           V C S   S i m u l a t i o n   R e p o r t
Time: 1550
CPU Time:      0.100 seconds;       Data structure size:   0.5Mb
Wed Oct 31 14:43:29 2012
```

图3：验证结果报告

4. 总结

本文介绍了SystemVerilog语言用于FPGA验证的功能特征，并在这些特征的基础上，以APB系统为例，使用VMM验证方法学搭建了可重用的仿真验证环境。在这个环境中，可以支持随机输入的产生，通过参考模型的自动比对，利用功能覆盖率做驱动，对验证流程进行控制，使得验证环境的控制方便灵活，重用性高。

动态仿真中代码覆盖率和功能覆盖率

文 | 上海旋极技术部工程师 马翔

以FPGA为代表的可编程门阵列凭借着可配置重构的特性，在各领域得到灵活和广泛应用。随着FPGA的结构日益复杂、容量不断提高、IPCORE的使用，在单片FPGA上已经能够实现片上系统等复杂和高速的逻辑，推动简化板上片外器件。随之而来的是对复杂FPGA设计和验证的挑战，覆盖率作为功能验证是否完整的重要指标也更加重要起来。

1. FPGA验证中覆盖率

FPGA设计的本质是数字电路设计，因此与软件不同，必须保证设计的功能与时序同时满足设计要求。动态仿真是最常用和最直观的测试手段，也是目前FPGA开发测试过程中使用最为广泛的测试手段。使用“前仿加后仿”的方法，动态仿真能同时考虑到功能和时序，但限制于覆盖不完全。时序部分可以通过静态时序分析弥补，功能部分的验证完整指标则是覆盖率。

因为目前的FPGA动态仿真器的计算能力远远不能达到枚举FPGA设计中所有的可能状态。动态仿真一般根据所实现的功能产生测试向量，通过测试平台依次注入到被测目标模块中，最后查看测试平台输出来验证功能正确性。这种直接测试向量生成的方法遵守WYTWYVO原则，即What-You-Thought-of-is-What-You-Verify-only。为了达到满意的覆盖率，需要产生大量的测试向量尽可能多地组成各种传输组合。这一工作需要花费大量时间和精力于反复编写、仿真、验证的过程。对测试用例准确预测覆盖率也是非常困难的，这时，需要测试人员通过研究覆盖率分布追加测试用例。

目前的动态仿真器都能够监视和计算Verilog、VHDL和混合HDL语言设计的覆盖度量，通过仿真来了解设计中的哪些部分没有被测试到。仿真器的覆盖率功能简化了收集覆盖信息、统计信息、指出未覆盖部重复琐碎的工作。分析结果能够总体地或详细地给出覆盖情况，帮助增加测试来满足覆盖。

在动态仿真器角度看，覆盖分为代码覆盖和功能覆盖。代码覆盖又分为控制流覆盖和值覆盖。控制流覆盖监视的是HDL代码的行和分支是否被执行，包括行覆盖和分支覆盖；值覆盖监视信号和表达式的值变化，包括翻转覆盖、FSM（有限状态机）覆盖。功能覆盖用来确定有多少设计功能被仿真执行到，一般使用手工列表并逐一确认，也可使用SystemVerilog支持的覆盖组结构体，该结构体使系统监视变量和信号的值与变化。

除此之外，使用断言时的断言覆盖也是常用的一种覆盖度量。

2. 覆盖率数据库

覆盖率的产生过程基本一致，以下图的VCS覆盖流程图为例，要获得覆盖率数据需三步操作。在编译阶段设定允许收集的覆盖率度量，在仿真阶段可选择部分或全部这些度量进行收集，数据将存放在vdb的覆盖率数据库。覆盖报告从覆盖数据库产生，此时可对数据库的覆盖做处理。报告的产生手段是多样的。所有度量的事后处理方式是基本一样的，URG以批处理方式产生文本和网页格式的报告，DVE是在图形化界面下做覆盖率交

互分析，UCAPI是用于自定义开发应用的，使得自定义格式的报告成为可能。

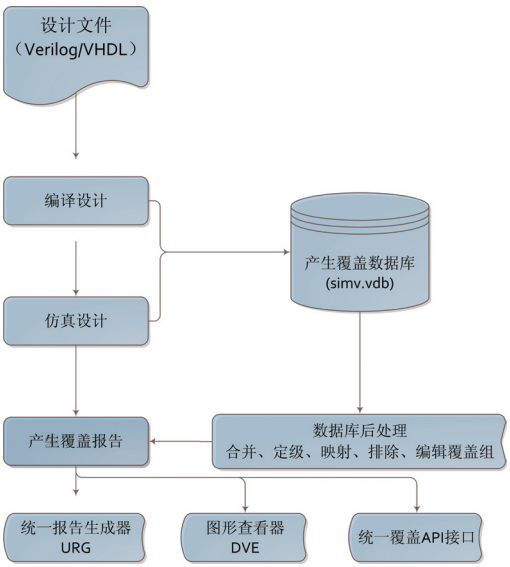


图1 VCS覆盖流程图

为了观察直观，以典型组合逻辑4位加法器addr4.v为例，该加法器的主体为以下always代码块。

```
always @(in1 or in2) begin
    sum = in1 + in2;
    if (sum == 0)
        zero = 1;
    else
        zero = 0;
end
```

针对该设计的测试平台文件为tb1.v，带覆盖率的VCS命令在Makefile内容如下：

```
allc:   cleanc compc runc

cleanc:
        rm -rf simv* csrc* *.log AN.DB DVEfiles ucli.key
compc:
        vlogan tb1.v addr4.v
        vcs tb1 -l comp.log -cm line+tgl+branch
runc:
        ./simv -l run.log -cm line+tgl+branch
```

在编译和仿真阶段，分别增加了一个-cm参数要求收集行、翻转、分支覆盖数据。由于addr4是组合逻辑，不包含时钟，因此这里不收集fsm覆盖。如需收集，只需用加号连在后面即可。

编译和仿真分别加参数是从仿真性能考虑的。仿真性能考虑的是占用主机内存、消耗的仿真时间。在编译阶段将收集覆盖的指令加入时，要求的覆盖度量被加入将略微降低性能，这时仿真执行文件simv具有这些度量的覆盖收集能力。但在仿真阶段如只收集编译时指定的部分覆盖度量，由于收集覆盖将增加计算量，写两次覆盖率参数是灵活性和性能间的权衡。

仿真结束后，功能目录下的simv.vdb是保存覆盖率数据库的目录。使用图形化界面查看较为方便，例如下图显示的翻转覆盖率，通过红-黄-绿的颜色判定显示各变量在仿真中翻转的覆盖情况，并在源码中用背景色表示。

```
$ dve  -  covdir simv.vdb
```

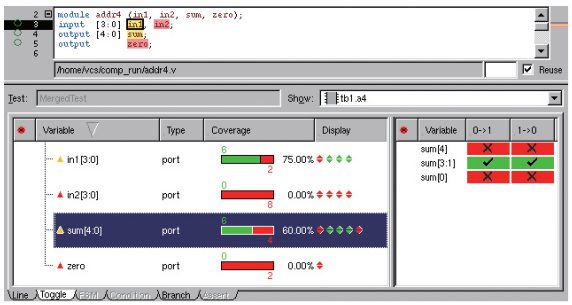


图2 覆盖率查看

DVE方式的缺点在于不方便将覆盖率报告固化保存到文件，需要使用URG产生的两种格式：txt和html，方便测试归档、报告转发和回归追溯。

```
%urg -dir simv.vdb -format both
```

3. 覆盖率合并

一些用户会为一个FPGA设计做多个动态测试工程。将多个测试工程的覆盖率合并得到的总覆盖率也是能够作为测试完整度的度量的。为了达到合并的目的，以下几点必须得到满足：

- 使用的VCS版本一致
- 运行平台一致，包括64位平台上的-full64参数
- 被测模块来一同一个拷贝
- 被测模块在测试平台下的路径是一致的，包括测试平台的模块名

以上几点都可以在测试启动时约束好。例如addr4的两个测试项目，使用DVE合并查看：

```
$ dve -covdir simv.vdb/ -covdir ../comp_run2/simv.vdb
```

此时tb1由于内容不同，其合并结果并没有参考价值，但被测模块的数据被合并了。如下图所示，第二测试项目对in2端口提供了激励，使in2的翻转覆盖率提高了：

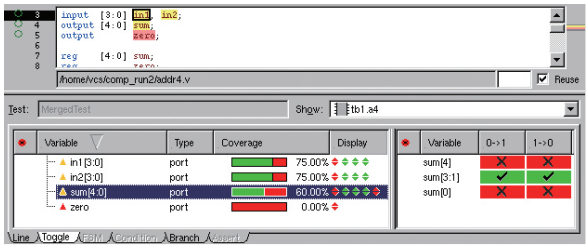


图3 覆盖率合并

在参数化的测试平台下，在一次编译后通过参数的设置在仿真阶段进行不同的仿真。为了让前一次的覆盖数据不被写掉，可将覆盖率数据写到数据库中的不同测试下。

```
% ./simv -l run0.log -cm line_tgl+branch +number=0 -cm_name test_0
% ./simv -l run1.log -cm line_tgl+branch +number=1 -cm_name test_1
% ./simv -l run2.log -cm line_tgl+branch +number=2 -cm_name test_2
```

查看覆盖率时与普通方法一样，看到的是整个数据库中各测试的合并值，但在test项目下能够查看到不同测试的各自覆盖率。

4. 关于功能覆盖

功能覆盖一般手工完成，最多使用Excel做编辑器。功能覆盖的先决条件是功能细分，一般在需求分析阶段已经完成。在执行仿真前需做以下几个工作：

1）对按层展开的功能细分做加权设置，即重要的功能占比较大，次要的功能占比较小。

2）对底层的最小功能描述实现确认方式，这一方式必须是能在仿真中被捕捉或能够调试出的。包括在什么状态下给出怎样的激励能够得到怎么样的结果。

3）对于一些不能枚举的条件，例如32位地址线，为了尽量提高覆盖率。对激励使用随机数激励，并对随机的产生加以约束。

4）对多次实例化的模块不必重复给激励，但需重复列功能细分表。在仿真结束后将高覆盖率的模块映射到低覆盖率模块，提高整体覆盖率。

现在功能覆盖在测试平台编写时一般使用covergroup定义覆盖组，以便在使用随机数进行自动激励产生时收集功能相关的覆盖信息。

（上接42页）

```
pt_shell> all_input
    {Input_1 Input_2 Input_3 Clock Select}
pt_shell> all_output
        {Output}
```

当在主输入设置约束时，用户可以用get_port指定所有的端口或独立的端口，

```
pt_shell> set_false_path -from [all_input] -to [all_output]
pt_shell> set_false_path -from [get_port Input_1] -to [all_output]
```

数据库的变量还包含每个网和引脚，需要注意的是，当设置一个约束，如false path或generated clock，总是在实例上设置约束而不是网上。一般在布局布线过程中网可以被重命名或移除，因此约束在网上会引起脚本错误。

PrimeTime设置约束时还可以接受正则表达式，这给约束到多个实例或引脚上提供方便。最后，不知道命令的用法可以使用帮助命令man，如果不想看命令描述而只是想看命令输入，可用-help参数。

```
pt_shell> man set_false_path
pt_shell> set_false_path -help
```

五、结束语

在通过大量测试平台进行动态仿真之前，静态时序分析是对设计进行时序分析最精确的方法。Synopsys 的PrimeTime软件是用户交互的，允许用户应用时序约束到设计上。大部分设计的时序特性由数据库提供。但是，用户需要结合时序分析经验和工具能力抽取不同的时序信息，包括路径延时、时钟性能、建立和保持时间。Tcl语言环境提供用户快速容易地访问需要的信息。

提高FPGA复位设计可靠性的建议

文 | 李丽华 王 栋 郑金艳 赵 静（北京市7200信箱9分箱 100074）

摘要：本文对FPGA设计中常用的复位设计方法进行了分类、分析和比较。针对各种复位方式的特点，提出提高复位设计可靠性的一些建议，对工程实践应用具有较好的指导意义。

关键词：FPGA复位；可靠性；异步复位；同步复位；内部复位

1. 引言

对FPGA芯片而言，在给芯片施加工作电压前，芯片内部的各个节点电位的变化情况都是不确定、不可控的，而这种不确定不可控的情况会使芯片在上电以后的工作状态出现错误。因此，在FPGA的设计中，为保证系统能可靠的进入工作状态，以及避免对FPGA输出关联的系统产生不良影响，FPGA上电后应进行复位，并保证复位的正确、稳定、可靠。

在FPGA的设计中，在大多数情况下复位电路的功能虽然能够正常完成，但是电路并未得到精确合理的设计，存在可靠性设计缺陷。为保证系统复位的可靠性，有必要对FPGA复位的可靠性设计方法开展研究。

2. 复位设计方法分类

根据与系统时钟域的关系，复位电路可以分为同步复位和异步复位。同步复位是指复位信号只有在时钟沿到来时，才能有效。否则，无法完成对系统的复位工作。异步复位是指无论时钟沿是否到来，只要复位信号有效，就对系统进行复位。

根据是否存在外部复位端口，复位电路又可分为外部复位和内部复位。外部复位，复位信号主要来自外部引脚的输入，如复位按钮、电源模块输出等。内部复位，复位信号主要由FPGA内部电路产生。

3. 复位设计方法的比较

3.1 同步复位与异步复位

3.1.1 同步复位

指定同步复位时，always的敏感表中仅有时钟沿信号，仅仅当时钟沿采到同步复位的有效电平时，才会在时钟沿到达时刻进行复位操作。用Verilog描述如下：

```
module Rst_Circuit(
                    Rst_n,
                    Clk,
                    D,
                    Q
                  );
input Rst_n;
input Clk;
input D;
output Q;
reg Q;
   always @(posedge Clk) //同步复位
       begin
           if (~Rst_n)
               begin
                Q <= 1'd0;
               end
           else
               begin
               Q <= D;
               end
       end
endmodule
```

如果目标器件或可用库中的触发器本身包含同步复位端口，则在实现同步复位电路时可以直接调用同步复位端。然而很多目标器件的触发器本身并不包含同步复位端口，这样复位信号与输入信号组成某种组合逻辑

（比如复位低电平有效，只需复位与输入信号相与即可），然后将其输入到寄存器的输入端。为了提高复位电路的优先级，一般在电路描述时使用带有优先级的if...else结构，复位电路在第一个if下描述，其它电路在else或else...if分支中描述。 同步复位电路综合后的RTL图如图1。

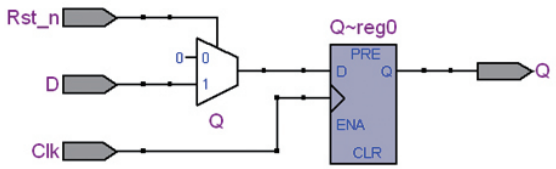


图 1 同步复位电路图

根据同步电路的特点，同步复位的优点如下：

（1）同步复位利于基于周期机制的仿真器进行仿真；

（2）使用同步复位可以设计100%的同步时序电路，有利于时序分析，其综合结果的频率往往较高；

（3）同步复位仅在时钟的有效沿生效，可以有效地避免因复位电路毛刺造成的亚稳态和错误。同步复位在进行复位和释放复位信号时，都是仅当时钟沿采到复位信号电平变化时才进行相关操作，如果复位信号树的组合逻辑出现了某种毛刺，此时时钟沿采样到毛刺的概率非常低，这样通过时钟沿采样，可以十分有效地过滤复位电路组合逻辑产生的毛刺，增强了电路稳定性。

同步复位的缺点如下：

（1）很多目标器件库的触发器本身并不包含同步复位端口，使用同步复位会增加更多逻辑资源；

（2）同步复位的最大问题在于必须保证复位信号的有效时间足够长，这样才能保证所有触发器都能有效地复位。由于同步复位仅当时钟沿采样到复位信号时才会进行复位操作，所以其信号的持续时间起码要大于设计的最长时钟周期，以保证所有时钟的有效沿都能采样到同步复位信号。事实上，仅仅保证同步复位信号的持续时间大于最慢的时钟周期还是不够的，设计中还要考虑到同步复位信号树通过所有相关组合逻辑路径时的延时，以及由于时钟布线产生的偏斜（skew）。这样，只有同步复位大于时钟最大周期，加上同步信号穿过的组合逻辑路径延时，再加上时钟偏斜延时，才能保证同步复位可靠、彻底。

3.1.2 异步复位

指定异步复位时，只需always的敏感表中加入复位信号的有效沿即可，当复位信号有效沿到达时，无论时钟沿是否有效，复位都会立即发挥其功能。用Verilog描述如下：

```
module Rst_Circuit(
                    Rst_n,
                    Clk,
                    D,
                    Q
                );
input Rst_n;
input Clk;
input D;
output Q;
reg Q;
    always @(posedge Clk or negedge Rst_n)
        begin
            if (~Rst_n)
                begin
                Q <= 1'd0;
                end
            else
                begin
                Q <= D;
                end
        end
endmodule
```

大多数目标器件（如FPGA和CPLD）和ASIC库的触发器都包含异步复位端口，异步复位会被直接接到触发器的异步复位端口，综合后的RTL图如图2。

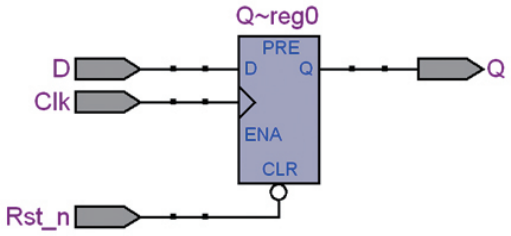


图 2 异步复位电路图

根据异步电路的特点，异步复位的优点如下：

（1）由于多数目标器件库的触发器都包含异步复位端口，异步复位会节约逻辑资源；

（2）异步复位设计简单；

（3）对于大多数FPGA，都有专用的全局异步复位/置位资源（GSR，Global Set Reset），使用GSR资源，异步复位到达所有寄存器的偏斜（skew）最小。

异步复位的缺点如下：

（1）异步复位的作用和释放与时钟沿没有直接关系，异步复位生效时问题并不明显；但是当释放异步复位时，如果异步复位信号释放时间和时钟的有效沿到达时间几乎一致，则容易造成触发器输出为亚稳态，形成逻辑错误；

（2）如果异步复位逻辑树的组合逻辑产生了毛刺，则毛刺的有效沿会使触发器误复位，造成逻辑错误。

3.2 外部复位与内部复位

外部复位，复位信号主要来自外部引脚的输入。复位信号在PCB板上可能会有来自其它线路的串扰，因此可能产生毛刺，在不需要复位系统时，毛刺信号导致系统误复位。

内部复位，FPGA上电配置完成后，由FPGA内部电路产生复位信号。复位信号与时钟同步。通常内部复位的设计方法是：设计一个初始值为0X0000的SRL16，将其输入接高电平，输出作为复位信号。

4. 复位可靠性设计方法

4.1 消除复位信号上的毛刺

在系统设计中，若采用低有效复位信号，可以按照图3中所示方法对复位信号中的毛刺进行消除。延时器件对数据进行延时的长度决定复位毛刺消除电路所能够避免的毛刺长度。而延时器件的延时长度也决定需要提供有效复位信号的最短时间。

如果复位信号高有效，则将图3中的或门改为与门使用。为了更好的消除毛刺，可以在复位毛刺消除电路后再加上寄存器对复位信号进行时钟同步，如图4。在通常的复位电路设计中，毛刺的长度很多情况下大于1个时钟周期，小于16个时钟周期。为节省资源，延时器件通常选用SRL16。SRL16是可以设置初始值，但不带复位功能16bit移位寄存器，可以通过A3~A0四根地址线选择从第几个寄存器输出。通常将其作为一个普通的16位移位寄存器来使用。图5显示了复位信号上3个周期的毛刺被电路有效消除。

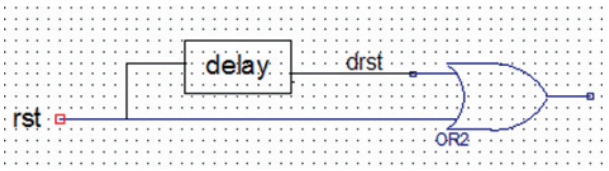


图3 复位信号毛刺消除电路

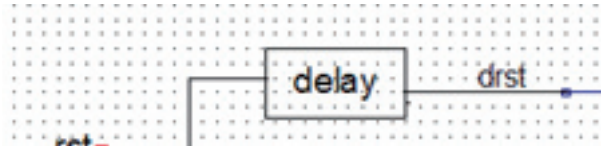


图4 复位信号毛刺消除改进电路

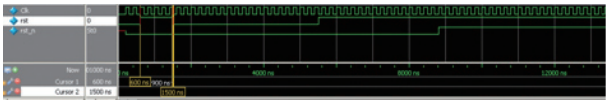


图5 毛刺被电路消除后结果示意图

4.2 异步复位同步释放

在有些应用中，复位信号需要在时钟尚未给出或不稳定的情况下传到后级，在时钟稳定之后，再撤去复位信号。此时需要使用异步复位来实现。由于异步复位时，时钟和复位信号关系的不确定性，易造成触发器输出亚稳态，引起逻辑错误。为保证异步复位的可靠性，通常采用异步复位，同步释放的方式。

所谓异步复位，同步释放就是在复位信号到来的时候不受时钟信号的同步，而是在复位信号释放的时候受到时钟信号的同步。通过一个复位信号综合器就可以实现异步复位，同步释放。用Verilog描述如下，综合后的RTL图如图6，其仿真结果图7表明该电路能有效的实现复位及脱离复位。

```
module ast(    Rst_n,
               Clk,
               D,
                Q );
input Rst_n;
input Clk;
input D;
output Q;
reg Q,Rst_Reg1_n,Rst_Reg2_n;
 always @(posedge Clk or negedge Rst_n)
  begin

 if (~Rst_n) begin
       Rst_Reg1_n <= 1'b0;
```

```
Rst_Reg2_n <= 1'b0;
      end
    else  begin
         Rst_Reg1_n <= 1'b1;
         Rst_Reg2_n <= Rst_Reg1_n;
       end
     end
always @(posedge Clk or negedge Rst_Reg2_n)
  begin
    if (~Rst_Reg2_n)
    begin
     Q <= 1'd0;
     end
    else begin
     Q <= D;
    end
  end
endmodule
```

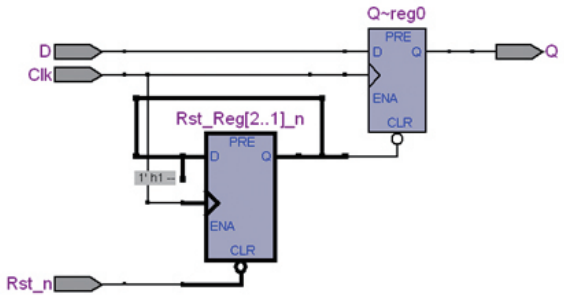


图6 异步复位、同步释放电路

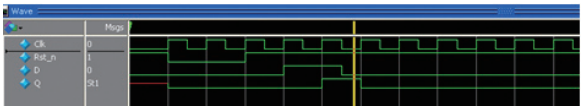


图 7 异步复位同步释放仿真结果

4.3采用专用全局异步复位/置位资源

全局异步复位/置位资源GSR（Global Set Reset）的主要作用是对系统中存在的所有触发器、锁存器、查找表单元的输出寄存器进行复位，不会占有额外的布线资源。使用GSR资源，异步复位到达所有寄存器的偏斜（skew）最小。

4.4 采用内部复位的设计方法

在不需要复位信号先于时钟信号产生的应用中，为了避免外部复位毛刺的影响、异步复位电路可能引起的亚稳态、以及减少资源的使用率，可以通过FPGA产生内部复位，然后采用异步的方式对FPGA内寄存器进行复位。由于该复位信号由FPGA内部产生，不会因外部干扰产生毛刺，同时与时钟同步，不存在因异步复位导致的亚稳态现象，可以保证系统可靠的复位。

5. 结论

本文对FPGA设计中常用的复位设计方法进行了分类、分析和比较。针对各种复位方式的特点，提出提高复位设计可靠性的一些建议，对工程实践应用具有较好的指导意义。

参考文献

[1]李忠琦，胡剑浩，FPGA中复位电路的设计研究，“2008中国通信集成电路技术与应用研讨会论文集”，2008

[2]Clifford E. Cummings, Don Mills, Steve Golson,Asynchronous &Synochronous Reset Design Techniques,2003

[3]李向涛，仵国峰，FPGA同步设计技术，无线通信技术，2003.3

[4]中电网元器件技术部，Xilinx公司FPGA设计技术问答精选，世界电子元器件，2003.2

适航验证技术

P54 DO-178C的新变化和对机载软件认证的影响

P57 通过基于目标（Objective-based）的标准获得航电软件的安全性认证

P63 SPARC V8目标码测试的实现方法

P67 DO-254规定的硬件生命周期数据要求探讨

P69 基于ADS2的数控系统控制软件系统测试环境搭建与应用

DO-178C的新变化和对机载软件认证的影响

■ 文 | 北京旋极军工中心副总经理 岳庆敏

背景

RTCA DO-178B《机载系统和设备合格审定中的软件考虑》自1992年发布以来，一直作为机载系统软件认证的主要参照标准，在安全关键软件的认证、流程改进、验证与测试等方面具有重要的地位。在过去的20年中，人们发现DO-178B中也存在着一些问题，此外这个标准也亟需解决新技术应用的问题。在SC-205工作小组7年时间努力工作的基础上，2011年12月，新版本DO-178C标准及其一系列增补标准正式由RTCA（航空无线电委员会）发布，共包括如下7部标准：

1、DO-178C，机载系统和设备认证中的软件考虑

2、DO-278A，通信、导航、监测和空中交通管理系统软件完整性保证的指导

2、DO-248C，DO-178C和DO-278A的支持信息

4、DO-330，软件工具审定的考虑

5、DO-331，基于模型开发和验证对DO-178C和DO-278A的增补

6、DO-332，面向对象技术和相关技术对DO-178C和DO-278A的增补

7、DO-333，形式化方法对DO-178C和DO-278A的增补

那么新的标准到底有哪些新的变化？将对机载软件认证带来怎样的影响？本文针对这两个问题做出一些分析。

标准更新的基本原则

SC-205工作组自2005年成立以来，主要的工作职责即更新DO-178B标准，工作组的职责说明中提到：

“DO-178C与DO-178B在保持向后兼容，对于现有文档的修改需要充分的关注软件开发支持系统安全性方面的当前技术和实践的状态，关注新出现的趋势，允许随着技术而修改”。

另外DO-178C标准特别注意保持技术的中立性，并不要求或者强调采用任何的技术手段来提高软件的安全性，同时并不要求或者建议用任何特定的流程，但是需要注意的是，流程需要在计划文档中体现，同时要有证据表明这些流程被很好的遵守了。

工作组在对DO-178B的修改问题上，持特别保守的态度，做到尽可能保持相同的章节号，维持现有的意图，解决存在的歧义，继续坚持基于目标的观点。工作组对每一次修改都要投票，曾经由于过分严格的投票规则（97%以上成员赞成才通过）导致工作进展缓慢而不得不修改投票规则（90%以上成员赞成票即通过）。

目标和活动

DO-178B是强调面向目标的，但是在标准中存在目标和活动并不是很清晰的内容，有些目标是隐含的，例如：“目标码到源代码的追踪分析”是隐含的一个目标，但是被描述为一个活动，在DO-178C中，“目标码到源代码的追踪分析”被明确描述为目标，在新的标准中，目标和活动描述得更加清晰准确。

同时DO-178C并不鼓励“检查单”的心态，如果认证申请单位不关注目标，而过多的关注活动，可能会导致其采用检查单的方式来确认是否完成了所有的活动，从而忽略了其目的，所有活动的目标是确保软件的安全性，而不是简单地确认每一项活动。

关于需求

DO-178C中允许需求中存在单层需求，也就是说对于简单的软件应用，可以不必强制要求创建两层需求（高级需求和低级需求），申请者需要对照自己的软件和流程来判断是否需要把高级需求和低级需求合并为一层需求。

DO-178C中允许在高级需求和低级需求中存在较高和较低需求层，即：系统需求→分配到软件的系统需求→高级需求→较高级/较低级需求→低级

需求→源代码。低级需求需要足够的详细清晰，足以支持源代码直接编写出来。

基于需求测试

在DO-178B中，始终强调基于需求测试，在DO-178C的§6.4.2中，基于需求测试被重点强调，因为这种方法被证明是最有效的发现错误的手段。

特别提出鲁棒性测试也必须是基于需求的，也就是说，不能基于代码进行鲁棒性测试，必须先具备鲁棒性需求，再进行鲁棒性测试。

对于采用逻辑方程式表示的需求，明确要求MC/DC测试，可以这样理解，即使对于C级软件，如果需求采用逻辑方程式表示，也要做MC/DC测试。

新标准同时要求，所有用来获取结构覆盖率的测试用例，必须加以分析，建立测试用例与需求之间的追踪关系，也就是说，“根据高级需求设计测试用例，看一下哪些代码没有覆盖再补充测试用例，直到满足覆盖率要求。”这种做法将会被认证当局否决。

以上要求总结为一句话：基于需求测试用来进行鲁棒性测试和覆盖率分析。

追踪性

在DO-178B中，要求生命周期数据要具备追踪性，但是并没有明确要求追踪关系是单向还是双向，在DO-178C中，明确提出，所有的追踪关系必须是双向的。即：

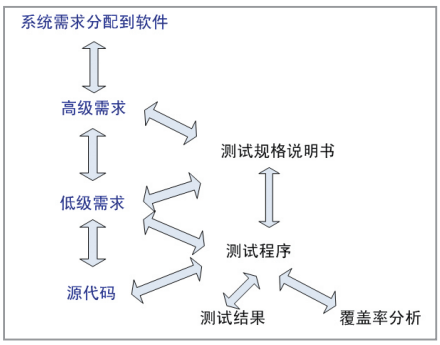


图1 生命周期数据追踪关系

在认证材料中，表现追踪性的方式可以是显式的，例如，追踪性矩阵、超级链接，也可以采用隐式的方法，例如，命名规范。追踪关系对于软件认证而言是非常重要的，全面而直观表示生命周期数据的追踪关系，有利于审查官快速对样本数据进行全面审查。

数据耦合和控制耦合

数据耦合和控制耦合是一种结构化覆盖率分析，用以确认基于需求的测试遍历过软件模块之间的数据和控制耦合。以前有些认证软件采用单元测试打桩的技术来进行数据耦合和控制耦合的验证，在DO-178C标准中，这种做法将不再会被接受。标准要求这个目标只能通过对集成后的映像文件进行测试来满足，而且，不允许对代码进行插装，如果插装代码，测试的对象就不是集成后的原始映像文件了。

下面给出一种建议的测试方式，测试程序和待测试的应用分别下载到目标板的内存中，待测试的应用是一个完整的二进制映像文件，并且是未插装的，测试程序逐个模块进行测试，而此时待测试应用仍然保持完整映像，这种情况下，测试可以验证数据耦合和控制耦合，当所有的测试通过后，再把覆盖率分析加上，重新运行相同的测试用例，获取覆盖率的结果，这种做法完全符合DO-178B和DO-178C的要求。

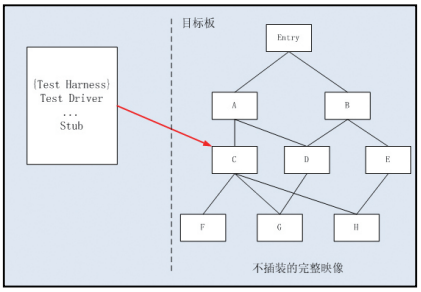


图2 符合DO-178C要求的一种测试方法

此外，还需注意，需要用审查和分析确保软件架构的完整性、源代码和软件架构的符合性，这些是单独的目标。

覆盖率分析

在DO-178B的§ 6.4.4.2 (b)中，关于覆盖率分析的内容如下：

“结构化覆盖分析可以在源代码级进行。但是如果是DO-178B A级软件，并且有些编译器产生的目标代码不能直接追踪到源代码中的语句，那么验证工作就需要采取额外的分析方法，即在目标代码的级别上验证编译器产生的代码序列的正确性。”

在DO-178C的§ 6.4.4.2 (b)中，关于覆盖率分析的内容修改为如下：

“结构化覆盖率分析可以在源代码、目标码、或者可执行的目标码上执行。并且与执行结构化覆盖率的代码形式是独立的。如果软件为A级，而编译器、连接器或者其他手段产生了一些多余的代码，并且不能直接追踪到源代码语句上，那么需要进行额外的验证以确认这些产生代码序列的正确性。”

注意：“多余的代码不能直接追踪到源代码语句上”是指那些由编译器或者链接器引入的分支或者副作用未明显的表现在源代码上。

这里明确提出结构化覆盖率分析可以在源代码、目标码或者可执行目标码上进行，也就是说三种覆盖率分析方法都是可以接受的。但是如果仅仅在源代码级开展覆盖率分析，还需要对目标码到源代码的追踪性进行额外的分析，而目标码覆盖率分析可以实现覆盖率的目标，同时也提供了目标码到源代码的追踪分析，被认为是一种更好的覆盖率分析方式。在很多机载软件认证中，这种方式被证明是很受局方认可的。

在讨论这个条款的过程中，编译器开发者提出，编译器在目标码中增加额外代码是有意的也是必要的，例如构造函数和析构函数。而大多数人认为开发人员看不到这些多余的目标码，所以目标码覆盖率分析很必要的。

参数数据

在机载软件中，存在一类数据，我们称之为参数数据（parameter data），这类数据具备如下特性：

1）不与应用软件链接在一起，分开下载；

2）运行时提供运行参数。

例如，IMA系统中的配置数据就是参数数据。

高级需求描述软件如何使用参数数据，低级需求定义参数数据的结构、属性和允许的值，对于参数数据，需要进行专门的验证，由于参数数据的每一个数据都是独特的，不能采用等价类法减少测试用例的数量，因此参数数据的验证比代码测试更加困难，可以采用对配置数据开发工具进行合格性审定的方法来替代对参数数据进行验证，例如，在VxWorks653的IMA系统中，XML格式的配置表由VerIMAx编译为二进制文件，下载到目标系统中，VerIMAx的编译模块通过了开发工具的合格审定，VerIMAx的配置表生成器和检查器作为验证工具进行了合格性审定，因此其输出结果二进制的配置数据表是无需进行验证的。

增补标准

与DO-178C一起发布的，还有4部增补标准（DO-330、DO-331、DO-332）分别对软件工具、基于模型开发、面向对象技术和形式化方法方面进行了阐述，这四部增补标准与DO-178C共同作为机载软件认证的标准族，可以单独购买。

这四本增补标准主要是保持对新技术的跟踪，明确了新技术如何应用于机载软件开发，也阐述了这些新技术的验证问题。

新标准带来的影响

由于DO-178C是完全兼容DO-178B的，如果充分理解了DO-178B，并且流程完全遵循了DO-178B，那么DO-178C对现在的机载软件开发组织而言，其影响是很小的。反之，如果对DO-178B的理解并不很深刻，可能在本文之前提到的一些目标实现上做法是不同的，那么当面临DO-178C的认证时，就需要做出很大的修改，需要重新制定计划，对每一项活动重新进行修正。

DO-178C本身并没有法律效力，直到相应的航空法规颁布之前，软件认证还是参考DO-178B。FAA的法规预计到2012年底会颁布，届时DO-178C就会成为机载软件认证审核的主要法定依据，当然我国的航空法规必将根据DO-178C进行更新。可以预见的是，由于国产大飞机C919按照计划2014年首飞，2016年交付，在这之前，其适航认证的工作必将面临DO-178B到DO-178C的转换的问题。因此，尽早开展DO-178C的研究，根据新标准对流程和活动进行调整，将会在未来软件取证中取得主动，这是需要引起我国民用航空领域足够重视的问题。

通过基于目标（Objective-based）的标准获得航电软件的安全性认证

文 | B.Scott Andersen and George Romanski, Verocel, Inc.
翻译 | 旋极信息军工中心测试与控制部高级工程师 王鹏

航电软件已成为如今飞行器设计基石。借助航电系统之优越性，飞行器得以减轻重量而降低燃油消耗、得以精确导航、得以提升发动机性能，以及获得其他好处。这些优越性使得现代飞行器成为飞行数据中心，由计算机控制或监控机载关键系统。于是，飞行器系统软件必须不遗余力确保安全。

FAA（Federal Aviation Administration）及其欧洲分部，联同机身、引擎和航电设备的主流制造商，制作了航电软件的开发指南，形成《机载系统和设备的合格审定中的软件考虑》[6]。在美国，由非盈利组织RTCA发布为DO-178B标准，而在欧洲则由EUROCAE发布为ED-12B标准。DO-178B中的指南表现为一系列目标和活动，软件产品的认证必须为之而满足或执行。

软件的安全评估及风险分析需要描述当其失效时对机身、机组人员和乘客产生的后果，以此确定软件之DAL（Design Assurance Level）水平。计有5级DAL水平如下（直接转引自DO-178B和FAA Advisory Circular AC 25.1309-1A）[1]：

灾难性：阻止继续安全飞行和着陆之失效状态。

危险/严重：降低飞行器性能、或降低机组人员克服不利操纵状态之能力的失效状态。所谓不利操纵状态表现为：（1）严重降低安全性余量或功能能力；（2）身体疲劳或高负荷使机组成员不能精确或完整完成其任务；或（3）对乘客之不利影响，包括对少数乘客严重或潜在的致命伤害。

较重：可能降低飞行器性能、或降低机组人员克服不利操纵状态之能力的失效状态。这些不利操纵状态表现为：显著降低安全性余量或功能能力、显著增加机组人员工作量或削弱其工作效率、或致乘客不适甚至伤害。

较轻：不会显著降低飞行器安全之失效状态，机组成员的活动可在其能力范围内很好完成。较轻的失效状态可以是：稍微降低安全性余量或功能能力、稍微增加机组工作负荷，如调整航线或引起乘客某些不便。

无影响：不影响飞行器工作性能或不增加机组工作量的失效状态。

DO-178B制定了从A到E的软件级别定义（Software Levels Definitions），大致对应上述DAL，A级软件

最为关键，E级则关键级别最低。目标之满足依赖于指定给项目的软件级别。本文讨论A级软件的活动和目标。

1 软件工程101

Nancy Leveson在其著作《Software: System Safety and Computers》4的前言中指出："一个明显的教训是，多数事故并非肇始于未知的科学原理，而毋宁是因为没有实施清晰标准的工程实践。第二个教训是，防止事故不是单独的技术维修所能胜任，更需要对系统开发与操作之全方位控制。"所以，软件开发过程的定义和控制是制造安全关键系统的首要因素。

读过DO-178B制定的目标和活动，在大学里学过软件工程课程的人应该感到熟悉。此处之关键在于其活动及工作产品是明确的和可重复的，所以必须制定软件生命周期模型，说明过程之间的迁移准则。要之，所有事情都需写下来。软件开发和验证团队应该使用一些计划和标准，其活动需经审核以保证遵循这些计划和标准。计划包括：

PSAC（Plan for Software Aspects of Certification），与认证机构交流整个项目计划的基本手段。

SDP（Software Development Plan），定义软件生命周期和开发环境。

SVP（Software Verification Plan），描述如何满足软件验证过程之目标。

SCMP（Software Configuration Management Plan），描述如何管理工作产品。

SQAP（Software Quality Assurance Plan），描述SQA（Software Quality Assurance）如何保证计划被遵循以及标准被满足。

DO-178B要求呈具3个标准：需求、设计和源代码。

要求必须提供与SCMP一致的SCM系统。

另外，必须提供问题追踪或缺陷追踪系统，记录软件本身问题或在遵循标准和过程时所产生的问题。所有这些活动、计划和标准（除PSAC）都是SEI[3] 2级或3级组织所应具备的。

2 没有意外（NO SURPRISES）

"安全"何所指？若谓十足安全乃绝无风险存在，则真实世界可用之物亦将绝无安全可言。是故，特定系统之安全，仅为相对命题，由系统之潜在风险、风险发生之概率、以及减轻风险之效果而定。

欲清点与了解系统风险，当首先知道系统之预期行为，否则难以知晓其错误。所以强大丰富之需求为构造安全关键系统所必需。需求应呈现为多个抽象层次，高者如机身之需求，沿之而下为多个系统、LRU、特定子系统之CSCI、子系统之高层需求和低层需求。抽象层次固可视飞行器和项目而定，但多层次实为需求之共性。

今再论何为安全，部分解答可总结为"不产生意外"（no suprises）。软件当满足所有需求，而且，亦不应发生非预期之功能。也即，软件做且只做需求之规定，不增不减。试想遇险于30,000英尺高空，你必无闲情了解软件还有哪些隐藏之额外功能。

3 可追踪性

前言"系统不应发生非预期之功能"，自然引发疑问：非预期之来源。需求既然开发于愈来愈低之层次，其语句必然含有更多细节与规范。低层需求必须为实现者提供附加信息，或者只是重复其父需求之描述。然则，低层需求或者有所增加，或者仅仅分解或细化其上层需求，其间如何区别？

需求层次之间（以及与项目其他工件之间）之关系由所谓"可追踪性"来管理。可追踪性描绘两个或多个项目之间的导航关系，例如，某上层软件需求可由此追踪至一个或多个下层需求。若此映射显示为上层需求之分解和细化（未增加新的行为），则无需额外的安全性分析。若低层需求不能直接追溯至上层需求，则此低层需求可能引进了非预期的功能，此时，低层需求（未经映射）应予以安全评估。

藉助可追踪性，最高层需求向下到每个抽象层次直到最低层需求之映射可次第而获。从最低层需求开始，

可追踪性亦持续至设计工件、源代码、验证方法和数据、以及其他相关工件。于是，从系统需求可一直追踪到相关代码和验证工具。

4 评审与独立性

每个需求和其他工件必须经过评审，评审活动则以项目计划文档中制定之准则为基础。另外，根据软件安全级别，特定工件之评审或需独立进行。所谓独立性，定义于DO-178B中：“隔离责任人，以完成目标评估。对软件验证过程，当验证活动由被验证产品之开发者以外之人执行，同时使用工具以替代人工验证活动，可保证独立性。对软件质量保证过程，独立性亦需权威机构介入，以保证行为正确。”

DO-178B并未制定如何进行评审。某些组织为此召集评审员，召开会议、收集时间、以评审组之名义签署评审。任何见证或参与评审组之人，都将发现此评审高度依赖于评审员之智慧以及召集评审会议之组织文化。这种方式之风险，概言之，正因每人都是责任人，故无人为此负责。

Verocel，身为提供软件验证服务的公司，采用不同凡俗的评审方式。它不指派一组工程师执行评审，而是指派单一工程师，单独负责工件及其评审的完备性和正确性。虽如此，被指派工程师并非独立奋战，他仍会征求其他小组成员或工件提供者之帮助、回答问题、提供进一步分析或要求澄清。然而在最后阶段，评审检查表上只会出现一人之签署，工件及其评审之质量由其一人负责。

5 工件构成与可追踪性之结构方法

论及工程工件及其追踪关系之组织，应从实践角度出发。工程通常含有成百上千之需求，如何管理？追踪关系如何维护？兼之，DO-178B提出一系列目标却未说明这些目标如何满足。这些都取决于开发团队，他们在项目计划文档中提供每个目标之满足细节，包括工件组织和追踪性。开发团队与认证机构（或其代表）必须对此计划达成共识。

需求管理有多种商业手段。一些组织使用文字处理工具和电子表格（协同一些特殊过程）以达此目标。而Verocel则发现这些手段均嫌不足，于是开发了自己的需求和工件管理系统，以关系数据库为其后台存储。此系统名为VeroTrace，直接管理需求文本，保持需求与其他工件如设计部件、源文件、测试过程和测试结果之间的参考关系，维护于CM（配置管理）系统中。因此VeroTrace满足可导航的追踪性目标。既然DO-178B未曾规定管理模式，我们就能够创建令人信服的组织和追踪系统，而非仅仅是一摞纸质检索卡。从实践角度出发，大型软件开发项目确实需要自动化地管理项目工件、评审状态以及追踪性。

6 好的需求

何为“好的需求”？DO-178B的目标为：(a) 符合系统需求，(b) 准确性和一致性，(c) 与目标计算机的兼容性，(d) 可验证性，(e) 符合标准，(f) 可追踪性，以及 (g) 算法问题。对这些题目，DO-178B均有说明。比如，目标“符合标准”就建议应有某些标准得以遵守。实际中，项目计划文档要包含需求之标准和开发指南、设计部件、编程标准、测试开发标准和其他软件开发指南。这些标准与指南之文档化有助于SQA评估开发过程是否遵守标准，并当其没有遵守时予以正确动作之要求。

需求开发标准尚需额外准则。例如，需求之标识应唯一，以便在评审和追踪关系中不致混淆；需求应具备版本标识，以便识别其变更并评估变更之影响；需求之作者应予以标记，否则无法保证评审之独立性；同样的原因也适用于需求之评审者。DO-178B的一个目标通常产生另外一组必须满足的隐含活动与目标，藉此满足初始目标。

这听来似乎需要大量工作。诚然，许多工作需要正确执行。该体系之意图在于提供一套结实可靠之需求，具有与同级抽象层次（若有需要）和其他抽象层次的追踪性，确保没有“非预期之功能”。唯有如此方能开始评估软件是否安全。

7 瀑布模型

通常所谓“瀑布模型”之软件开发过程遵循下面步骤：指明所有需求、完成架构和设计文档、然后是编码。实际却并非如此，几乎没有大型项目可循此而成。正因认识到这一点，DO-178B指南在第3章中声明：“本文件指南并未描述一个首选的软件生命周期，但描述了组成大多数生命周期的独立过程，以及这些过程之相互联系。过程之分离也不暗示执行该过程之组织结构，至于每个软件产品，其软件生命周期终是由这些过程构造。”进而，DO-178B文中12.1.4d一节声明：“逆向工程可用于再生成软件生命周期数据，以弥补其在满足本文件所含目标时的不足和缺陷。”概括之，重要的是满足DO-178B提出的所有目标，而满足之次序不做规定。

FAA对逆向工程之实践持有相当的关注，并委任George Romanski（Verocel Inc.）和Mike DeWalt（Certification Services Inc.）研究此问题。（DeWalt其时已回到FAA做为飞行器计算机软件的首席科学和技术顾问）。二人之研究成果，Reverse Engineering Software and Digital Systems[2]，证实在被调查的项目中有68%使用了某种形式的逆向工程。该报告还揭示了其他可资注意的事实，其结论表明软件开发过程无需“瀑布模型”也可成功。

8 确认与验证（Validation and Verification）

我们常说Validation和Verification，究竟何意？Validation，即“是否构建了正确的产品”；Verification，即“是否正确地构建了产品”[5]。FAA之所以组织逆向工程研究，也是因为一个明显的观察点是，逆向工程可以重申代码做了什么而不是确定代码应该做什么。Validation——确定是否构建了正确的产品——也是安全过程之重要组成部分，没有捷径达此目标。

逆向工程研究使用了Verocel从13个项目中提取的数据，含250,000个e-LOC（有效代码行）（对C程序，有效代码行不含空行、注释行、只有单一括号或else或其他关键字的行）。这些项目所报之问题可分为如下门类：设计错误、注释错误、文档错误、错误处理问题、测试错误、结构覆盖问题、已修改的功能、需求错误和编码错误。发现这些问题之途径也包含：评审、分析（用于工件逆向工程之人工检查）、观察（与特定工件无关之人工检查）、beta测试/功能测试、结构覆盖分析、或系统测试。

于所发现问题中，77%由工程师使用工程手段发现，这其中54%来自分析、8%来自观察、15%来自评审。也就是说，工程师在努力确定软件是否实现了它们应该实现的功能时，发现了3/4的软件问题。唯成功执行Validation工作之项目工件，方可成功执行Verification工作。

9 软件验证（Software Verification）

可通过任意一种手段或结合多种手段完成软件验证，最常见手段自然是“测试”。基于需求之测试把需求作为测试准则，产生之测试报告最终要说明被测软件是否满足这些需求。

次常见手段为“分析”。测试非万能，有相当需求难以或无法通过测试验证，例如。若有需求要求锁中断，在特定操作期间形成一关键段，那么中断之锁定未必能从外部检测。这种情况下，适当做法是形成一个分析文档，提供该需求被满足之证据。

测试与分析决非仅有之验证方法。其他如“形式化方法”可证明算法之精确性或对软件需求之匹配性。又或者如压力传感器之类小型系统，“穷举的输入测试”（exhaustive input testing）可完全覆盖软件的输入范围。甚至产品的运行历史也可在特定环境下用做验证方法。

10 结构化覆盖分析

关于SCA（结构化覆盖分析），DO-178B如是说：“此种分析之目的在于确定在基于需求之测试过程

中，哪些代码结构未被执行。……结构化覆盖分析在源代码上执行，除非软件级别为A且编译器产生之目标代码无法直接追踪至源代码语句。果如此，则需采用额外的验证手段，即执行在目标代码上，验证所产生代码序列的正确性。目标代码不能直接追踪至源代码的显见例子为，编译器生成的目标代码中的数组边界检查。”

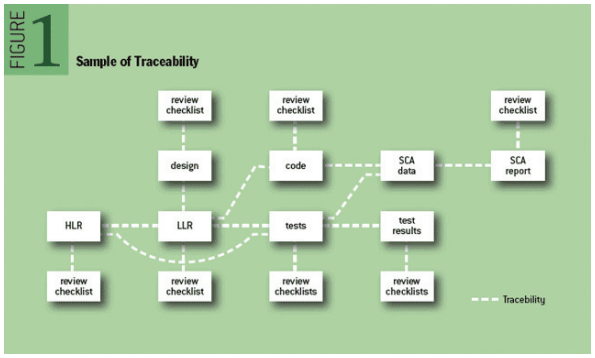
SCA背后之理念相当简单：若测试完备且软件也完全满足需求，那么理应得到100%的代码覆盖。然而有多种理由可说明这不是真命题。例如，对软件做健壮性检查可发现有些执行路径是不可达的，因为在常规测试条件下不能创造它们所对应的异常状态。

SCA的缺口也可能表明其他问题：基于需求之测试用例或过程的缺点、软件需求不充分、或者“死代码”。所谓“死代码”是指不能追踪到任何需求之代码，因此或者应该把死代码剔除出系统，或者需要更新需求（如果该代码功能是必要的）。

飞行期间不希望执行不活跃代码（Deactivated Code），对应之需求应出现在软件中（这些代码也许只用于地面维护活动）。重要之处在于证明在不希望运行这些代码时，它们没有被意外执行。

在显示被测软件之代码覆盖率或目标码覆盖率之外，还需进一步分析代码之间的数据耦合与控制耦合。编译器之输出既不可信，就需测试和结构化覆盖分析方法，证明源代码和目标代码已被完全覆盖，且任何不一致都得以补救和记录。与编译器类似，链接器同样不可信。SCA数据和控制耦合数据也是可评审之工件。

Figure 1显示从高层需求到SCA报告及其评审检查表之间的追踪网络，虚线表示追踪关系。该图仅是示例（并不完整），通过分析进行验证的文档并未描绘出来。即便如此，它也呈现了典型的追踪网络。



11 开发和验证工具

如前所述，编译器和链接器都不可完全信任，它们只是生产飞行软件之开发工具（Development Tools）。我们能制造合格的开发工具（Qualified Development），其输出无需检测。虽说制造这样工具（及其验证）的指南，等同于制造和验证飞行软件的指南，然而有时制造这样全套验证工件的工作是物有所值的。

合格的开发工具之输出无需进一步验证工作。

相比之下，制造合格的验证工具（Qualified Verification）的工作不那么繁重。这样的工具可以替代评审员。它只要求有需求、基于需求之测试、以及适当的由DO-178B概括的文档。Verocel即有这样一个合格的验证工具，VerOLink，帮助执行控制耦合分析。

12 DER的角色

在美国，FAA虽是认证机构，但不为每个项目提供开发飞行器软件之工程资源。而是开发了一个DERs（Designated Engineering Representatives）系统，FAA藉此把项目监察职责委派给经过特殊培训和适当认证的工程师，这个被委派的工程师隶属于独立的公司或者制造商本身。FAA始终执行项目之最终签署，而DERs则执行大部分工作，跟踪项目进展，评估项目产生的材料。

指派给项目的DERs在项目之不同阶段召集四个SOI（Stages of Involvement）会议。第一个会议，SOI-1，覆盖项目认证计划，评审那些保证项目得以顺利进行之关键材料。包括，验证获得成功认证所必需之进程和过程、确保有一个妥善坚实的计划以满足每个DO-178B目标、保证DER和软件开发者就项目的所有重要部分都已达成共识。因此，第一个会议之要点在于，保证认证计划适当，令每个人都认为如果遵循了该计划且所需工件已齐备，那么就将顺利获得认证。

第二个会议，SOI-2，评审第一个会议后产生的问题，评估计划之任何改变所造成的影响，研究项目目前产生的材料之质量。成功经历SOI-2阶段的项目，其最

小状态当为，50%的需求已经开发和评审、50%的设计工件已完成和评审、50%的源代码已经完成和评审，以及所有这些工件具备追踪关系。此会议之目标不是评估或评审所有需求或其他工件，而是尽早识别问题，以便当问题的成本不致特别昂贵、破坏力不致特别巨大时采取正确的行为。这个会议所需材料之规模或多或少为此处说明之50%，因其完全介于DER和开发团队设置的目标和里程碑之间。

第三个会议，SOI-3，发生于开发全部完成，且验证工作完成50%之际。验证工作可采用多种方式：测试、分析、形式化方法、或项目计划中描述并经过同意的任何手段。

最后一个会议，SOI-4，发生于已完成项目的所有认证证据都已生成时，主要是评审最终认证包的可读性。

DER处于对立的角色。他/她也可作为资产而存在，帮助项目团队认识其计划漏洞和过程缺陷；甚至也可作为促进者，提出建议，帮助项目摆脱束缚。DER不能参与开发过程计划；DER的工作是评估材料，而非开发他们以后将要评估的材料。好的DER应坚韧不屈，确保项目产生高集成度的软件。

13 总结

DO-178B虽非规范性的，但其准则和目标却很广泛。一篇短文中难以尽述DO-178B之要求，这里我们突出强调了DO-178B能指导生产安全软件之要点。必须开发良好的需求，进行审查，保证它们准确描述了软件所应实现之功能。这些需求和所有软件开发工件（包括设计、代码、测试、测试结果、结构化覆盖数据）通过一个可导航之追踪网络建立连接，确保每个需求都能满足，确保没有非预期之功能。而另一端，结构化覆盖分析发现未被测试覆盖的代码，死代码也因此得以标记和移除。

所有这些都赖以坚实的可记录的工程实践和成熟的软件开发环境，借助适当的过程和控制，就可以制造令人信任的安全软件。

参考

1. Federal Aviation Administration. 1998. System Design and Analysis, Advisory Circular AC 25.1309-1A (June 21).

2. Federal Aviation Administration. 2011. Reverse Engineering Software and Digital Systems, April 2011 [Draft Report]. To be published by the FAA William J. Hughes Technical Center.

3. Humphrey, W.S. 1990. Managing the Software Process. SEI Series in Software Engineering. Reading, MA: Addison-Wesley Publishing Company.

4. Leveson, N. 1995. Safeware: System Safety and Computers. Addison-Wesley Publishing Company.

5. Rakitin, S.R. 2001. Software Verification and Validation for Practitioners and Managers, second edition. Norwood, MA: Artech House.

6. RTCA. 1992. Software Considerations in Airborne Systems and Equipment Certification (DO-178B).

SPARC V8目标码测试的实现方法

■ 文 | 上海旋极技术部软件工程产品经理 闵蓓尔

1. 背景

星载计算机是卫星最核心的部分，目前32位RISC处理器为主流。美国SUN公司公布了SPARC的VHDL语言模型，使越来越多人关注SPARC结构。欧空局和ATMEL公司合作，生产出SPARC结构的宇航级抗辐射芯片，成功应用在其卫星上。目前我国的相关研究单位也展开对SPARC结构的研究。欧洲Gasier Research根据SPARC V8结构设计了Leon2核并公布。国内的研究多以大容量FPGA为载体，实现Leon2核，并依据需求来加入自己的模块。国内星载计算机将越来越多的采用SPARC V8。

星载软件大部分使用高级语言，如C和ADA来进行编制，编译器不同、编译选项的不同都会产生的不同目标代码。在以往的项目中开始遇到对目标代码分析不彻底而出现的质量问题，因此目标码的覆盖率测试是星载软件的一个必须进行的测试项目。下面针对SPARC V8提出一个目标码测试的解决方法。

2. 测试方法

VerOCode是美国Verocel公司推出的目标代码级测试工具，VerOCode是一个不需要特殊硬件的执行跟踪分析工具。被测试的代码不需要插装（不添加记录执行状态的功能调用）。应用代码和VerOCode监控在目标计算机SPARC V8上执行，执行的数据图表搜集到一个宿主机PC上。根据收集的执行数据图表，链接器符号信息和编译器产生的清单，可以显示出哪些指令执行了，哪些指令没有执行，以及条件指令执行过程中的条件代码状态。测试方法见图1。

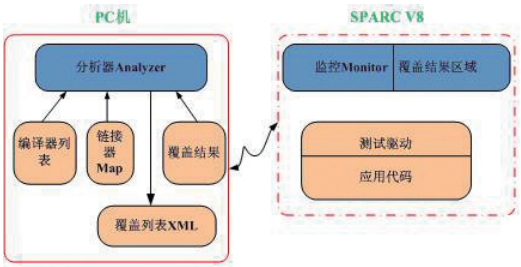


图1 VerOCode测试方法

- 驻留在V8目标系统中的监控程序负责搜集覆盖率数据，并把覆盖结果存放在专门的覆盖结果区域；
- 基于PC机的分析工具Analyzer，将SPARC V8上传的覆盖结果进行分析，标记编译器列表文件，链接器产生的Map文件，说明源代码文件中通过功能的测试达到的覆盖率，保存这些信息到XML文件用于分析；
- 基于PC机的Coverage Editor，在不修改原始覆盖率清单的情况下，能够通过注解显示没有执行（没有覆盖）的指令。

3. 监视器

Monitor是目标机内存驻留程序模块，监视被测程

序的运行，创建表格，反映出被执行的指令及相关的条件代码。

测试装置负责Monitor的初始化，启动和停止，以及从目标机到主机上传输覆盖率结果。监控原理见图2。

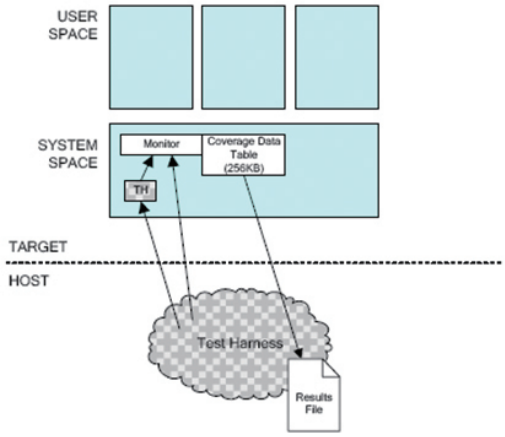


图2　监控原理

Monitor提供一组接口（Monitor API），测试装置使用这些接口可控制覆盖率处理过程。对载入的被测程序没有限制。能够针对不同类型的程序获取覆盖率数据，包括设备驱动和板级支持包（BSPs）。

首次启动Monitor之前，覆盖率范围、数据表格和覆盖率模块需要通过vcmonInitializeCoverage 或vcmonInitializeCoverageEx例程进行动态配置。当Monitor停止运行时，可能会多次调用两个例程，仅最后一次调用有效。

```
void vcmonInitializeCoverage( void* startAddress, /* coverage start address */ int size /* size of coverage data table */ );
```

参数：

startAddress：覆盖范围的起始地址。覆盖范围是程序内存中处理器指令的一段连续区域，用于存放收集的覆盖数据。

Size：覆盖范围大小表示成许多代码块（每块64KB程序内存大小）。由于监测器的静态覆盖数据表格的大小当前是256KB，表格的每个字节覆盖4个指令（16字节），覆盖范围最大为4MB，也就是64块。

如果size小于或等于0，或大于64，假设缺省值是64块，即代码的4MB区域的最大覆盖范围被覆盖了。

覆盖率范围和模式配置之后，经用户判别，可以启动（vcmonStartCoverage）和停止（vcmonStopCoverage）覆盖率集合。注意，如果有必要的话，覆盖集合可能被分开初始化，启动和停止。

当覆盖率测试完成时，覆盖率结果通过一个转化例程vcmonTransferResults或vcmonTransferResultsToBuffer从Monitor读取。用户可以使用任何可通信的信道，从Monitor到主机PC，上传接收到的覆盖率结果（格式化的表格）。主机上的测试装置会将结果保存成一个文件。文件内容可由分析器直接使用。

4. 测试实例

实例是在北京时代民芯科技有限公司的MTX0106板，XGC LEON Ada编译器环境下的一段ADA源代码。V8 RAM的地址从0X40000000开始。

- 在被测代码中加入监控API

```
procedure Hello2 is
   use MTX0106IO;
   use Interfaces;
xxx: Unsigned_32;
type v1 is array (1..4000) of character;
buff:v1;
r:integer;

begin
xxx :=Disable_Irq(16#ffff#);
for I in  1..4000 loop
   buff(I):=' ';
end loop;
VeroCodeAPI.Initialize_Coverage(System'To_Address (16#40004670#), 2);
VeroCodeAPI.Start_Coverage;
   Serial4_Init(115200,0,0);
   Serial4_Putc('H');
   Serial4_Putc('e');
   Serial4_Putc('l');
   Serial4_Putc('l');
   Serial4_Putc('o');
   Serial4_Putc(' ');
   Serial4_Putc('W');
   Serial4_Putc('o');
   Serial4_Putc('r');
```

```
        Serial4_Putc('l');
        Serial4_Putc('d');
    Delay1ms (20);
    Serial4_Puts("你好");
    VeroCodeAPI.Stop_Coverage;
    r:=VeroCodeAPI.Transfer_Results_To_
buffer(True,buff'Address,4000);
    for I in  1..4000 loop
        Serial4_Putc(buff(I));
    end loop;
    end Hello2;
```

监控Monitor中的vcmonInitializeCoverage函数负责初始化monitor，VeroCodeAPI.Initialize_Coverage(System'To_Address (16#40004670#), 2)表示覆盖统计起始地址是0X40004670，覆盖区域的块数是2，大小是128KB。

vcmonStartCoverage表示开始覆盖率监控，vcmonStopCoverage表示停止覆盖监控。在停止覆盖监控后，通过

r:=VeroCodeAPI.Transfer_Results_To_buffer(True,buff'Address,4000) 向指定的Buff输出数据，在程序开始的地方，定义了4000大小的字符类型的数组Buff。在最后将Buff数组中的覆盖数据通过串口输出。Verocode也支持直接向串口输出覆盖监控数据。

在整个监控的设置中，覆盖统计起始地址，覆盖数据区的大小以及缓冲区数组的大小可以根据被测代码的实际情况进行设定。

- 运行被测代码

使用XGC leon进行Build,产生执行文件hello2：

```
leon-elf-gnatmake  -I./hardware -I/opt/leon-
ada-1.8.3/lib/gcc-lib/leon-elf/2.8.1/adainclude
-f -g -O0 hello2.adb -largs hardware.o -largs  ./
hardware/VeroMon.o -T CPCI_BM3803_xgc_sram.x.mon
-Wl,-Map=hello2.map -I.
```

启动Debug下载运行hello2：

```
grmon -leon2 -uart /dev/ttyS1
debug>lo hello2
debug>run
```

通过串口收集到了Verocode监控的覆盖信息：

```
H 40004670 4002466C 00004000 00000107
D 00000000 0000001E 00000000.00003FFF.BFFF000F.FFFFFFDC.003FFFFF.FFFFFFFF.FEFFFF00.00000000
  0FFFC000.00000000.00000000.00000000.0003FFFF.FFFFFFFF.FFFFFFFF.FFFC0000
  00000000.00000000.003C0000.DC0EFDC0.00000000.00000000.03FEFFF7.00000000
  003FEFF0.3BFC0DC0.00000000.0EF703F0.0003F7E3.C0000000
Z 00000780 000007E2
T 00000002
```

将监控覆盖信息保存到Result.res文件中。

- 分析监控覆盖信息

Analyzer校对来自目标机上的覆盖信息，此信息具有汇编级别的源码列表。分析器用符号和信息来注释列表，使覆盖信息的分析容易理解。列表中的单个指令地址用特定字符标记，表明该指令是否被执行过，条件分支和跳转指令的情况下，分支是否执行过。

首先建立hello2.adb的lst文件，

```
Objdump -S hello2.o >hello2.lst
```

调用Verocode的Analyzer命令：

```
Analyzer.exe -ADDR 0x4000490c -LST hello2.
lst -res result.res -cov FunctionUnderTest-
hello.xml
```

-A[DDR] address 参数，地址为需要进行目标码覆盖分析的函数起始地址,通过Map文件，在这里设置目标覆盖分析的起始地址是0x4000490c。

```
.text._ada_hello2
                0x4000490c      0x1b8
 .text._ada_hello2
                0x4000490c      0x1b8 ./hello2.o
                0x4000490c            _ada_hello2
```

经过分析器分析后，产生FunctionUnderTest-hello.xml文件。

- 启动VerOCode Coverage Editor显示覆盖率列表

运行命令CovEditor.exe FunctionUnderTest-hello.xml，显示界面见图3：

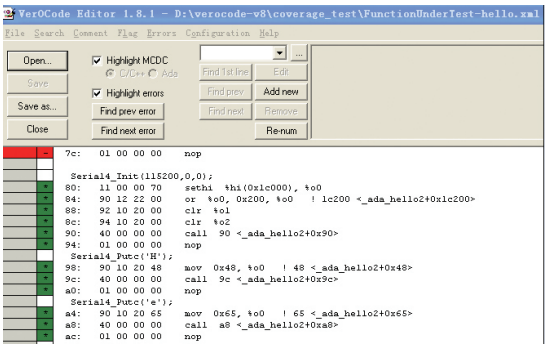


图3 CovEditor界面

VerOCode Coverage Editor（VCE）显示覆盖率列表的一个只读拷贝，高亮显示覆盖的和没有覆盖的代码行。这里是VerOCode标记过的示例覆盖率列表：

```
*  0034 stw r10, 36(r1)
>  0038 bne cr1, 0060
-  003c stfd f1, 40(r1)
```

其中：

- 星号（*）表示执行过的指令；
- 大于号（>）表示条件仅为真；
- 小于号（<）表示条件仅为假。一个条件如果真和假都满足过，就会被标记为执行的指令，即星号；
- 减号（-）表示指令没有执行过。

File菜单中的快捷按钮。点击其中的“Generate HTML report…”，生成HTLM报告，报告提供了COVERAGE SUMMARY和COVERAGE LISTING。

COVERAGE SUMMARY总结了指令总行数、已执行指令行数、未执行指令行数和执行不完整条件行。

COVERAGE SUMMARY:

Number of instructions in the listing:	110	
Instructions executed:	48	43.64%
Instructions not executed:	62	56.36%
Incomplete conditions:	0	0.00%
Status:	*** COVERAGE INCOMPLETE ***	

COVERAGE LISTING则显示了源代码、覆盖信息和注释的综合显示。

COVERAGE LISTING:

```
1
2     hello2.o:     file format elf32-sparc
3
4
5     Disassembly of section .text._ada_hello2:
6
7     00000000 <_ada_hello2>:
8     with MTX0106IO;
9     with Interfaces;
10    with VeroCodeAPI;
11    --with Interfaces.C.Strings;
12
13    procedure Hello2 is
14 -    0:   9d e3 b0 00    save  %sp, -4096, %sp
15 -    4:   9c 03 bf e8    add  %sp, -24, %sp
16    buff:v1;
17    r:integer;
18
19    begin
20
21    xxx :=Disable_Irq(16#ffff#);
22 -    8:   11 00 00 3f    sethi  %hi(0xfc00), %o0
23 -    c:   90 12 23 ff    or  %o0, 0x3ff, %o0   ! ffff <_ada_hello2+0xffff>
24 -   10:   40 00 00 00    call  10 <_ada_hello2+0x10>
25 -   14:   01 00 00 00    nop
26 -   18:   d0 27 bf f4    st  %o0, [ %fp + -12 ]
27
```

5. 总结

目前行业内常见的目标码覆盖率分析的原理，都是采用软硬件相结合的方式。这种方法存在着一定的缺陷：在目标码中插装会改变目标码形态；采用硬件探针必须关闭CPU模块的Cache功能；采用模拟器技术会使整个运行环境失真过大并且运行效率过低。

VerOCode在整个执行跟踪过程中都无需特殊的硬件支持，有利于对代码进行监视。VerOCode不需要对被测代码进行插装，这对于内存具有严格要求的嵌入式系统来说，是非常有益的。

通过以上的实践证明，VerOCode可以很好的完成SPARC V8系列目标码的覆盖分析。

DO-254规定的硬件生命周期数据要求探讨

■ 文 | 上海旋极技术部工程师 李茜

通过DO-254标准认证是一个复杂精细的过程，在设计、验证、测试、管理和核查等各阶段都需要提供充分的数据作为符合证据。通过该认证的数据要具有可靠性、可追溯性和可验证性，其中包含了多种计划书和报告。本文对申请DO-254认证所需要的数据和过程做了详细的介绍，为通过机载电子硬件设计标准提供参考。

1. 引言

目前，复杂电子硬件在飞行器上的大量使用为航空航天安全和认证带来了新的挑战。这些挑战是因为硬件越来越复杂，导致设计错误越来越难管理约束，使飞行器功能造成了严重失效。为了有效遏制这种风险的加大，在设计和验证过程中采用更连续和可验证的方法准确定位硬件设计错误是十分必要的。图1反映了DO-254应用于复杂电子硬件设计的硬件生命周期。作为一个过程的规定，DO-254不是对执行的细节作出规定，也不是对单个开发工具进行评定，这个过程的认证强调的是在最高的置信水平上规定的过程能够实现可验证性和可重复性。

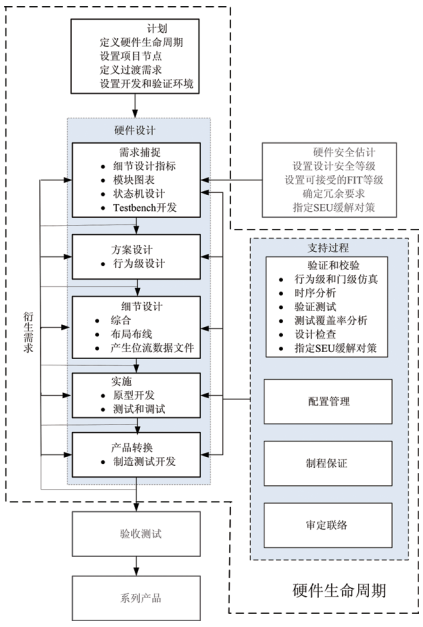


图1：复杂电子硬件的DO-254硬件生命周期

在上述DO-254流程中，决定于设计是否符合该标准的关键是验证的所有结果（包括仿真波形、回归状态、覆盖率图表等）必须是可追溯的，能够连接到形式需求的。这种可追溯性可以是自动的也可以是手动的、并且工具的输出能力决定了自动化的程度。

根据不同的安全等级标准，不同的机载系统规模，不同的硬件复杂性，在通过DO-254认证时认证机构要求提供的数据和材料也不尽相同，一般表现在数据的范围、数量和细节上。这些数据和材料都来自硬件设计生命周期中，作为设计保障和认证的要求提供参考证据。

2. 硬件计划

要通过DO-254认证需要指定多种硬件计划，形成统一的文档，这些硬件计划一般在申请认证的初期就需要提交认证机构，并与认证机构达成一致。一般的硬件计划分为：硬件方面的认证计划（PHAC）、硬件设计计划、硬件验证计划、硬件核实计划、硬件配置管理计划和硬件过程保障计划六种。最重要的是PHAC，其他的硬件计划可以包含其中进行提交。

PHAC定义了硬件认证的过程、步骤、方法和标准，形成统一的文档，用来取得认证机构的权威认证。PHAC一旦通过，表示认证机构和认证申请人在认证的过程和提供的数据等方面达成一致，这些都是用来满足硬件认证的参考资料。PHAC应该包括系统概述、硬件概述、认证注意事项、硬件设计生命周期、硬件设计生命周期数据、附加事项、迭代方法和认证计划安排。

3. 硬件设计数据

硬件设计数据是开发硬件过程中产生的表征硬件结

构、特征和功能的各种规格、文档和图表。主要包括需求数据、设计表征数据、验证数据、硬件验收测试标准和各种报告。需求数据是指要开发的硬件需要满足的功能、特性、安全性、质量、可靠性等各项指标。硬件设计表征数据和验证数据是通过DO-254认证最重要的数据。

3.1 硬件设计表征数据

3.1.1 概念设计数据

概念设计数据是描述硬件元件的结构和功能设计的重要数据，包括：一些设计的高级描述，如结构框图或HDL定义，这些数据强调的是硬件的主要功能和这些功能间的信息流传递。表现硬件元件排列的机械结构，如表现外包装、电路板排线、连接器位置和主要内部连接线的示意图等也是重要的数据，在开发过程中应注意有效保存。除了基本的结构特征，机载硬件系统可能还具有专门针对适航条件的一些特殊结构数据。例如硬件中具有电磁干扰、强光保护和防震等特殊组成，如使用了人体工学设计、光学特征和显示方案等资料，均是满足DO-254标准的重要数据。顶层硬件元件的功能描述和初步硬件安全评估数据也属于概念设计数据的范畴。

3.1.2细节设计数据

根据设计的不同层次划分，细节设计数据可能包括顶层模块设计图、功能集合块设计数据、内部连接数据、单模块数据、HDL硬件描述、可靠性数据、测试方法数据和一些不使用的功能列表、安装控制数据、软硬件接口数据等，这些数据都是以需求为驱动的，用来表明设计满足指定的硬件设计安全等级。例如设计中指定的软硬件接口数据，如内存地址、分配给数据下载的内存地址域、时序和序列信息及调用软硬件接口需要的其他信息等都是细节设计的重要数据。

3.2 验证核实数据

验证核实数据为硬件满足开发需求和设计目标提供保证，包括硬件复查和分析测试的结果。

3.2.1 可追溯性数据

DO-254标准中要求产生的所有验证数据均是可追溯性的，就是在需求、细节设计、功能实现和验证数据间建立一个相关连接，有助于配置控制、修改和硬件核查。

硬件追溯性数据是指系统需求分配的各个硬件需求间的相关数据、硬件需求和细节设计数据间的相关数据、细节设计数据和完工的硬件元件或硬件集合间的相关数据、各个需求（包括衍生需求）、细节设计数据和验证过程及结果间的相关数据和追溯性分析的结果。

3.2.2 复查和分析过程及其结果

验证设计后，要进行复查和分析。这里需要对复查和分析的方法结构、细节指导、工具使用等进行说明。最后要给出问题报告和正确功能的报告，以及复查分析结论，需要对设计进行复查定性评估和定量分析。进行复查的结果就要求数据具有可追溯性，方便定位故障问题原因，和评估安全等级的功能满足情况。

3.2.3 测试步骤和测试结果

测试步骤要对硬件元件验证核实的方法、环境、执行的指令和环境定性测试等进行记录说明。这里要记录的还包括测试的目的、测试环境的搭建、测试输入数据、预期的测试结果和要求的覆盖率指标。测试结果包含了每个测试阶段的数据、硬件适航性测试的数据，结果的解释，包括分析复查和真实测试达到的覆盖率等数据都要求。

3.3 报告

经过设计和验证后，验收中提供的硬件验收测试标准报告，主要是对测试的关键属性和测试约束等进行说明，保证测试的硬件元件可以被正确地重复验证。

DO-254还要求提供问题报告、硬件配置管理报告、硬件过程断言报告和最后的硬件完成总结。具体是对验证的问题、配置操作、修改历史、复查审计等过程提供详细的说明报告。硬件完成总结是对所有过程是否满足PHAC计划中提出的需求，并强调能够实现硬件适航目标。

4. 总结

在申请DO-254过程中，需要提供详尽的PHAC计划，将设计的功能需求和适航需求转化成硬件生命周期、安全评估和配置管理等过程的具体实现方法，提供每个过程实现的全面报告和数据，在需求、细节设计、功能实现和验证数据间建立可追溯性数据连接，实现数据的可重现验证。

基于ADS2的数控系统控制软件系统测试环境搭建与应用

■ 文 | 中航工业航空动力控制系统研究所软件部测试主管　王栋

1 概述

发动机数控系统控制软件属于军用A类软件，根据GJB/Z 141-2004要求，A类软件必须进行系统测试，GJB/Z 141-2004中还提到“系统测试的目的是在真实系统工作环境下检验完成的软件配置项能够和系统正确连接，并满足系统/子系统设计文档和软件开发任务书规定的要求”，DO-178B中也提到“一个优秀的测试环境应能把软件加载到目标机中，并在目标机环境的高保真仿真环境中对其进行测试。”“因为某些错误只能在这种环境中被发现。”因此，一个好的系统测试环境对于软件系统测试来说是十分必要的。

我所有自研的测试设备，能够基本支持所内项目软件系统测试工作，但是存在一定的问题：a）信号稳定性不足，温飘现象明显；b）模型运行实时性不好，这将导致测试结果会与真实环境运行结果存在一定差异，影响测试结果可信度，且自研设备不直接支持故障注入与自动化测试功能。为了解决这些问题，在改进所内自研设备的设计的同时，另一方面在市场上寻求技术较为成熟、运行稳定可靠的系统测试环境，通过一段时间的调研，我们选择了德国TechSAT生产的ADS2。ADS2是Avionics Development System 2nd Generation的缩写，即第2代航空电子设备开发系统。

通过一段时间的摸索与实际使用，ADS2能够解决现有所内自研测试设备所面临的问题：

a）信号稳定性：通过12小时连续使用，设备未发现有温飘现象，信号精度在信号量程的0.1%量级；

b）实时性：采用VxWorks嵌入式实时操作系统保证模型运行的实时性；

c）故障注入：提供硬故障与软故障两种故障注入模式；

d）自动化测试：提供脚本语

言机制，能够实现自动化测试。

本文主要介绍基于ADS2搭建数控系统控制软件系统测试环境的方法以及在某项目上的实际应用情况。ADS2基本使用方法请参考相应使用手册，本文将不做具体描述。

2 环境组成

整个系统测试环境主要由以下几个部分组成：

a）数字电子控制器，含数控系统控制软件

b）能够匹配电子控制器输入输出信号的ADS2设备

c）电子控制器与ADS2的连接电缆

d）连接ADS2的办公计算机（安装ADS2软件）

3 环境原理

ADS2模拟了整个数控系统被控对象的状态，发动机模型在ADS2的VxWorks操作系统中运行，并根据模型计算结果，通过ADS2相应的硬件板卡输出电信号，模拟传感器采集信号输出给电子控制器，电子控制器（含数控软件）采集信号并根据控制需求进行处理后输出控制信号，ADS2设备采集控制信号，通过执行机构模型模拟相应执行机构动作，并计算得到发动机模型所需物理量，发动机模型运算后，得到新状态的参数再次通过ADS2设备硬件板卡输出，不断循环。如图1所示：

图1 系统测试环境基本原理框图

4 环境搭建方法

ADS2设备主要由ADS2机柜与ADS2软件两部分组成，在完成硬件配置与必要的电缆连接后，剩下的工作主要是使用ADS2软件完成。如图2所示：

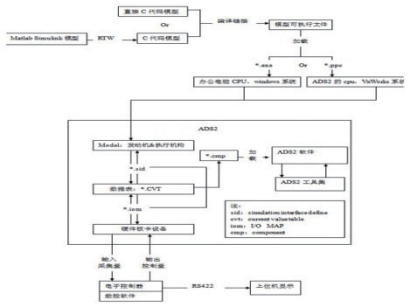


图2 基于ADS2的数控系统控制软件系统测试环境详细框图

4.1 模型编译

需要对被控对象模型进行编译，生成可执行文件，模型可以是源代码，也可以是动态链接库，唯一区别在于动态链接库只能编译成exe，无法编译成ppc加载至ADS2的嵌入式实时操作系统中运行，如图3所示。

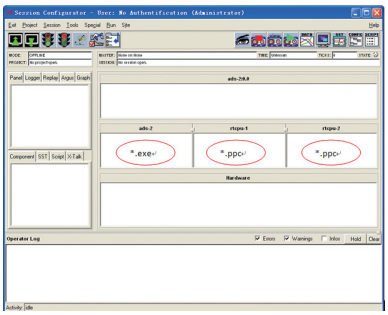


图3 模型加载区域示意

4.2 定义CVT

CVT：Current Value Table，是ADS2软件运行的核心部分，所有需要用到的变量均定义在CVT中，模型计算会刷新CVT，ADS2信号输出会读取CVT，ADS2信号输入会刷新CVT。

CVT与硬件板卡之间靠IOM文件关联。

CVT与模型之间靠SID文件关联；

4.3 定义IOM

IOM：I/O Map，定义CVT变量与硬件板卡之间映射关系，例如：通过IOM文件定义CVT中变量A关联到板卡AA中第1路，如果A是输出信号，则AA板卡第1路将会按照设定时序不断读取A值并输出，如果A是输入信号，则通过AA板卡第1路按设定时序不断采集并将采集值刷新至CVT中的变量A。

4.4 定义SID

SID：Simulation Interface Definition，定义模型与CVT之间交互接口的变量，通过SID的定义，会自动生成*.h头文件，使得SID定义的CVT变量对于模型来说是可见的，模型能够按需对定义的CVT变量读取或写入。

4.5 定义CMP

CMP：component，将所需的CVT、IOM、SID以及所需要用到的模型可以执行文件封装成一个组件，可以将多个不同时序运行的模型放在一个组件里面，如图4所示，model每5帧执行一次，model_inner每1帧执行一次：

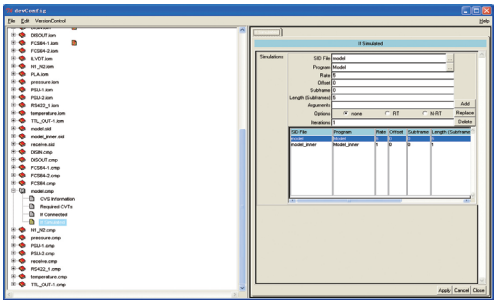


图4　CMP配置界面

4.6 定制面板

根据自己的需要，使用ADS2软件自带的面板设计功能进行控制面板设计，并设置控件关联相应的CVT变量，即可实现对所需监视变量的监视效果，如图5所示。

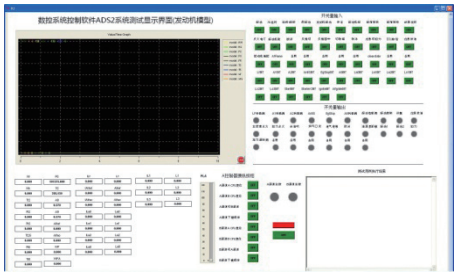


图5 界面示意图

4.7 整体调试

通过整体系统调试后，能够初步实现软件系统测试的基本功能。能够进行数控软件正常功能的验证，但是这是不够的，因为数控软件的故障对策功能还无法验证，环境还未实现故障注入功能。

5 故障注入功能

ADS2提供了硬件与软件两种故障注入方式：

a）硬故障注入：FIBO

b）软故障注入：Error Injection

5.1 FIBO

FIBO是Fault Insertion & Breakout System的缩写，能够支持同时最多4路信号的断线、对地短路、信号间短路等多种硬件故障模式，如图6所示：

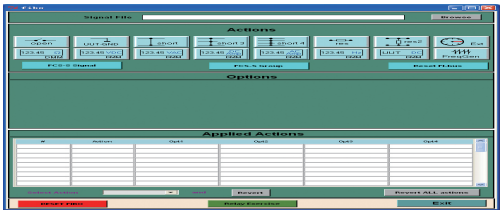


图6 FIBO控制界面

出于对被测对象安全考虑，目前只使用到了信号断线故障模式，其他硬件故障模式暂时未做尝试。

5.2 Error Injection

此软故障注入模式运用较多，主要用于模拟传感器信号异常，如信号采集超范围或者突变等情况，也用于对信号拉偏检查FADEC的控制是否符合要求。

5.2.1 基本原理

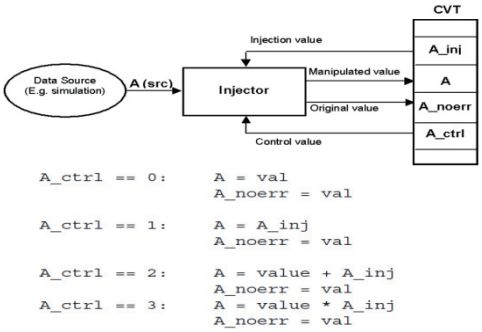


图7　Error Injection原理

如图7所示，对于CVT变量A来说，如果进入故障注入模式，则会自动生成额外的3个变量：

A_inj：故障注入值

A_noerr：无故障值

A_ctrl：故障注入模式，0表示无故障注入，1表示直接设置故障注入值，2表示在当前信号基础上叠加故障注入值，3表示在当前信号基础上乘以故障注入值。

5.2.2 故障注入模式设置方法

设置变量进入故障注入模式，只需要在定义CVT变量的时候，在面板上将Error Injection选项勾选即可。如图8所示：

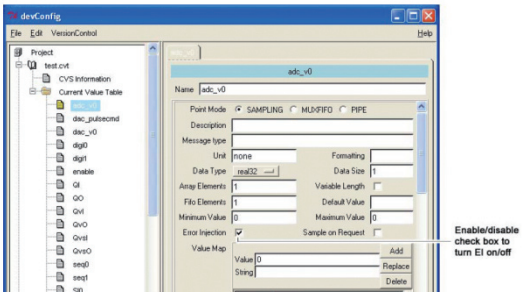


图8 Error Injection选择模式

5.2.3 简单案例

如果我们要模拟T2传感器故障：

首先：在定义T2变量的时候勾选Error Injection选

项；

其次：通过控件对T2_inj变量赋值来设定故障注入值，一般为信号突变值；

最后：设置T2_ctrl=1来选用直接设置故障注入值的模式使能输出，即可实现T2传感器信号突变模拟，检测数控软件对应的传感器诊断功能是否正常。

6 自动化测试

ADS2软件自带的脚本语言功能能够让我们实现自动化测试。

下面以一个简单的脚本例子来举例说明：

```
//ChannelAtoB :通道A切B
procChannelAtoB
{
accesses
{
out
   {
        uint8 DISOUT::DISOUT2_10;//A切B
    }
}
body
{
set DISOUT::DISOUT2_10 = 1;
delay 2 seconds;
set DISOUT::DISOUT2_10 = 0;
}
}
```

以上脚本实现了通道A切B功能的自动执行，如果手动执行的话，需要去点击关联了DISOUT::DISOUT2_10 CVT变量的按钮控件。从中可以看出CVT的核心地位，所有的脚本操作实际是直接操作了CVT变量，而CVT变量又直接反应到了与其关联的对象中，如控件、硬件板卡输入输出通道、模型计算等。

当按照脚本语言规则，将测试用例操作步骤（包括故障注入）脚本化，我们就能得到测试用例的自动执行，在测试脚本中，增加期望结果与实际结果的对比并给出结论，我们就得到了测试结果的自动判定，从而实现了利用ADS2脚本语言机制进行自动化测试的功能。

7 实际应用情况

ADS2在现有实际应用中主要的功能定位如下：

a）支撑数控软件自动化回归测试工作；

b）支撑数控软件系统测试与升级验证工作；

c）支撑数控软件HIL开发调试；

实际应用数据如表1所示：

表1 ADS2实际应用数据

编号	ADS2事项	数据
1	搭建系统测试环境数量	4个项目
2	年平均使用时间	1600小时
3	搭建环境周期	2周*1人
4	自动化测试用例个数	1336个
备注：搭建环境周期不含插头采购周期与连接电缆加工周期		

8 小结

通过基于ADS2搭建数控系统控制软件系统测试环境，我们能够解决现有设备信号稳定性差、模型运行实时性差的问题，同时，ADS2提供了故障注入、自动化测试的完整解决方案，为所内测试工作提供了“一个好的系统测试环境”。

参考文献

TechSAT GmbH202009_USM-ADS2_1007.pdf，2012-03-12

作者简介：

王栋，籍贯江苏无锡，生于1983年5月，毕业于南京航空航天大学，2005年至今就职于中航工业航空动力控制系统研究所软件部测试管理室，曾担任多个项目测试主管。

行业软件安全标准

P74 解析铁路软件安全标准EN50128与软件工具的支持

P77 适应ISO 26262安全标准的软件验证

解析铁路软件安全标准 EN 50128与软件工具的支持

文 | 北京旋极军工中心测试与控制部软件测试技术经理 任建国

轨道交通因具有运量大、安全正点、快捷舒适及污染小等特点，使建立以轨道交通为主的交通系统成为改善城市交通的有效途径。轨道交通系统在设计建造时采用了许多安全措施来保障运营安全，但由于轨道交通系统的运营工作牵涉到千百万乘客的安全正点出行，对建设和谐社会的影响重大，如何实现列车安全、保证高效的运行是目前轨道交通领域亟待解决的问题。由于轨道交通信号系统在提高运输效率、保证行车安全等方面具有决定性作用，因此，在轨道交通领域里，人们对安全相关系统的研究主要集中于轨道交通信号系统。

以IEC 61508国际标准为基础，欧洲电气化标准委员会(CENELEC)下属SC9XA委员会，制定了以计算机控制的信号系统作为对象的铁道信号标准（图1），它包括以下4个部分：

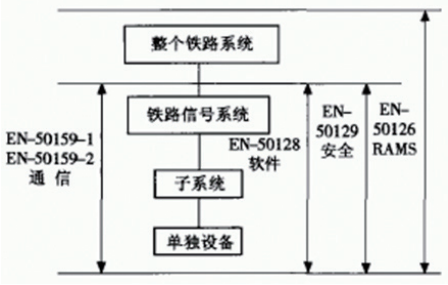


图1 CENELEC铁路标准关系图

EN 50126，铁路应用——可靠性，可用性，可维持性和安全性（RAMS）的规格和说明；

EN 50129，铁路应用——信号领域的安全相关电子系统；

EN 50159-1，铁路应用——通信，信号和处理系统；

EN 50128：铁路应用通信、信号和处理系统——铁路控制和防护系统软件；由于在信号系统中采用计算机越来越广泛，由软件来承担安全性需求的比重越来越大，因此软件安全性问题变得更加突出。为此，EN 50128标准针对软件的安全保证提出了相关的规范和设计标准，EN 50128标准的关键概念是其对软件的安全完整性水平的考虑。本文主要讨论EN 50128标准中的软件测试认证问题。

EN 50128标准详细规定了铁路控制和防护设备用的可编程电子系统开发所需的程序和技术要求。它适用于任何有隐含安全性的领域。这些应用系统的范围涵盖了安全苛求系统（如安全信号）、非安全苛求系统（如管理信息系统）。这些系统可能通过采用专用多处理器、可编程逻辑控制器、分布式多处理器系统、大规模集中处理器系统或者其它架构来实现。

EN 50128标准确定了从最低0级到最高4级的5个软件安全完整性等级SIL的技术和措施。其中1~4这四个级别涉及安全相关软件，0级涉及非安全相关软件。对0级进行标准化是为了让非安全相关系统软件向安全相关系统软件进行平滑转变。后文的附表给出了各个软件安全完整性等级和非安全相关等级要求的技术和措施。欧标没有给出某一风险应适用于哪个软件安全完整性等级的具体指导意见。SIL结论需要考虑许多因素包括应用的特性、其他系统承担的安全性功能范围和社会以及经济因素。

EN 50128标准是面向目标和过程的，它强调的是一种目标导向的做法：一方面，它要求给出明确的功能和安全性目标；另一方面它要求给出验证这些目标的方式；最后它要求给出达成目标的指标及证明。

EN 50128所规定的软件生命周期

根据EN 50128标准Figure 4 – Illustrative Development Lifecycle 2 中所规定的软件生命周期，有两类过程构成了软件生命周期的一条主线，它们就是开发过程和验证过程。

① 开发过程

进入软件生命周期后，EN 50128规定了如下开发过程：

●软件需求过程（Software requirements）。这一过程的工作就是进行软件安全性需求设计。它包含了软件的功能要求、安全性要求、执行时间和内存上的约束、软硬件之间的接口等内容。

●软件框架设计过程（Architecture and design）。它的工作就是把需求细化成为软件结构。它包含了软件的输入、输出、数据流、控制流、资源限制、以及调度和通信机制等。

●软件组件设计（Component Design）。把框架设计分解为多级组件，分别对各个组件进行功能和算法设计。

●实施过程和测试（Component implementation and testing）。这一过程把低层组件设计实现成为源代码，源代码集成、编译并链接成为在目标环境上的可执行代码。测试即为验证过程。

② 验证过程

验证过程的作用就是“找出并报告软件开发过程中可能引入的错误”。对于针对于代码的验证过程，常用的验证手段除了包含静态测试、单元测试、集成测试、系统测试，还包含故障注入测试、软件复杂度度量等等。作为对安全性要求极为苛刻的EN 50128标准，规定除了对开发过程要进行验证之外，还要有指标来证明这些验证工作已经做得足够充分，这就是覆盖率验证。例如测试是一项验证工作，为了评价测试工作做得够不够，可以用代码的覆盖率来判断，也就是说，测试工作中所用到的测试用例是否已经足够地覆盖了整个程序。

EN 50128标准提出了V&V，也即 Verification 和 Validation。业界默认解释为，Verification 乃“验证”之义，而Validation乃“确认”之义。Verification，验证系统没有做出错误的事情，用在测试的“集成”阶段，是一种抽丝剥茧的分析手段。Validation，确认系统实现了正确的功能，用在最终的产品交付确认阶段，是一种融合归纳的综合手段。假设“软件质量”或者“系统合格”为一断言，它需要充分必要条件。那么Verification就对应着必要条件，Validation对应充分条件。即是说Verification 是验证软件功能需求说明书，是面向研发的；Validation是确认用户需求，是面向用户的。

上面所述的过程构成了EN 50128所规定的软件生命周期主线。

符合EN 50128规范的软件测试平台由多款专业测试工具构成，其中包括英国PR公司的代码规则检查工具QAC/C++，澳大利亚Red Lizard公司的缺陷检测工具Goanna，英国QAsystem公司的单元集成测试与功能验证工具Cantata，美国McCabe软件公司软件质量及覆盖率验证工具McCabe IQ。以上工具完全符合EN 50128规范的要求，可以帮助客户实现EN 50128规范所要求的SIL验证工作，按照SIL4对应的标准要求进行编写。

1.1 符合EN 50128标准的静态分析工具

EN 50128标准对软件静态分析的要求，下表是从EN 50128的Table A.4-Software design and Implementation中抽取的EN 50128指南，以及工具的支持情况。

指南条款	工具	描述
Structured methodology/Programming 选择结构化方法编程	McCabe IQ	完全支持McCabe圈复杂度度量，反映程序结构化程度
Design and coding standards 设计和编程标准	QAC/QAC++ Goanna	支持全面的检测和多种编程标准MISRA C 2004、MISRA C++ 2008
Analysable Programs 可分析的程序	QAC/QAC++ Goanna McCabe IQ	支持代码静态分析，并生成流程图，辅助代码分析
Strongly Typed Programming Language 强类型编程语言	QAC/QAC++ Goanna McCabe IQ	支持C、C++语言。

下表是从EN 50128的Table A.5-Verification and Testing 中抽取的EN 50128指南，以及工具的支持情况。

指南条款	工具	描述
Static Analyisis 静态分析	QAC/QAC++ Goanna McCabe IQ	支持全面的静态分析、规则分析、质量度量等功能
Metrics 复杂度度量	QAC/QAC++ Goanna McCabe IQ	支持复杂度度量，包括代码行、圈复杂度、Halstead复杂度等

下表是从EN 50128的Table A.12-Coding standards中抽取的EN 50128指南，以及工具的支持情况。

指南条款	工具	描述
Coding Standard 编程标准	QAC/QAC++ Goanna	支持全面的检测和多种编程标准MISRA C 2004、MISRA C++ 2008
Coding Style Guide 编程风格指南	QAC/QAC++ Goanna	支持编程风格检查，定制命名规则
No Dynamic Objects 不使用动态对象	QAC/QAC++ Goanna	提供动态运行时错误检查，包括动态对象的检查
No Dynamic Variables 不使用动态变量	Goanna	提供动态运行时错误检查，包括动态变量的检查
Limited Use of Pointers 限制使用指针	QAC/QAC++ Goanna	定制检查器，标识并限制中断的使用
Limited Use of Recursion 限制使用递归	QAC/QAC++ Goanna	定制检查器，标识并限制中断的使用。
No Unconditional Jumps 不允许使用无条件跳转	QAC/QAC++ Goanna	通过MISRA检查器验证goto，longjmp的使用
Limited size and complexity of Functions, Subroutines and Methods 限制函数、子进程、方法的大小和复杂度	QAC/QAC++ Goanna McCabe IQ	提供各种复杂度的统计，包括代码行统计、圈复杂度统计
Entry/Exit Point strategy for Functions, Subroutines and Methods 限制函数、子进程、方法的入口、出口点策略	QAC/QAC++ Goanna	出口点可以用返回语句的度量来标识。
Limited number of subroutine parameters 限制子进程的参数个数	QAC/QAC++ Goanna	扇入扇出度量，可视化显示源代码，标识软件接口，从而加以限制。
Limited use of Global Variables 限制使用全局变量	QAC/QAC++ Goanna	全局变量可以通过架构的图形可视化来标识，确定和记录证明其使用的目的。

下表是EN 50128的Table A.19-Static analysis中抽取的EN 50128指南，以及工具的支持情况。

指南条款	工具	描述
Boundary value analysis 边界值分析	QAC/QAC++ Goanna	支持数组越界检查。
Checklists 检查单	QAC/QAC++ Goanna	规则列表和报告以检查单的形式给出。
Control flow analysis 控制流分析	QAC/QAC++ Goanna McCabe IQ	控制流发呢系可以通过架构的图形可视化来标识。
Data flow analysis 数据流分析	QAC/QAC++ Goanna McCabe Data	MISRA检查器可检测数据未初始化、解引用。 McCabe Data可以在流程图中标记出特定变量。
Walkthroughs / Design Reviews 软件走查/设计审查	Goanna	获得先进图灵奖的模型检测技术，即使在最复杂的函数中，也可完美覆盖运行路径

1.2 符合EN 50128标准的动态测试工具

EN 50128标准对动态测试的要求，下表是从EN 50128的Table A.5-Verification and Testing中抽取的EN 50128指南，以及工具的支持情况。

下表是从EN 50128的Table-A.13-Dynamic analysis and testing中抽取的EN 50128指南，以及工具的支持情况。

指南条款	工具	描述
Dynamic analysis and testing 动态分析和测试	Cantata Tessy	支持动态覆盖率分析和功能测试
Test Coverage for code 代码覆盖率分析	Cantata Tessy	支持代码语句覆盖、分支覆盖、MC/DC覆盖率等。
Functional and black box testing 功能和黑盒测试	Cantata Tessy	支持基于需求的测试用例，通过图形用户界面GUI的使用，创建基于需求的测试用例，可以指定参数的输入和期望值，返回值，全局变量，桩函数，支持黑盒功能测试。
Interface test 接口测试	Cantata Tessy	完全支持接口的自动分析，接口数据包括参数、全集变量、外部调用。

EN 50128标准对动态测试的要求，下表是从EN 50128的Table A.14-Functional/Black Box Test中抽取的EN 50128指南，以及工具的支持情况。

指南条款	工具	描述
Test Case Execution from Boundary Value Analysis 边界值分析	Cantata++	提供表驱动的测试，自动生成接口数据的边界值。
	Tessy	根据分类树方法，设定边界值作为测试用例
Test Case Execution from Error Guessing错误推断	Cantata Tessy	可根据凭“积累的经验”和专业的判断所收集到的数据来建立。
Test Case Execution from Error Seeding故障注入测试	Cantata	提供Wrap封装功能，可模拟软硬件故障的发生，进行故障注入测试
Equivalence Classes and Input Partition Testing 等价类划分	Cantata	提供测试用例管理器
	Tessy	提供分类树方法的图形编辑环境，可以把输入和输出按照取值来分类组合而成为测试用例。
Structure-Based Testing 结构化测试	McCabe Test	完全支持

指南条款	工具	描述
Boundary Value Analysis 边界值分析	Cantata++ Tessy	支持边界值分析法设计的测试用例
Equivalence Classes and Input Partition Testing 等价类划分	Cantata++ Tessy	支持等价值划分法设计的测试用例

EN 50128标准对动态测试的要求，下表是从EN 50128的Table A.21-Test Coverage for Code中抽取的EN 50128指南，以及工具的支持情况。

指南条款	工具	描述
Statement 语句覆盖	Cantata Tessy	完全支持
Branch 分支覆盖	Cantata Tessy	完全支持
Compound Condition 组合条件覆盖	Cantata Tessy	完全支持
Data Flow 数据覆盖率	McCabe Test	支持
Path 路径覆盖	McCabe Test	支持

适应ISO 26262安全标准的软件验证

■ 文 | 旋极信息军工中心测试与控制部高级工程师 王鹏

1 概述

随着国民经济的不断发展，我国已明显进入汽车时代。人们对汽车品质的要求，促使汽车复杂程度不断提高，越来越多的控制单元具备与安全相关的功能。汽车安全系统功能的不断升级和提高也伴随着更多的安全风险。由于电气器件、电子设备、可编程电子器件在汽车控制领域的大量使用，一些全新的安全问题不断的被媒体曝光。例如虽然脚踩刹车却发现车辆不断加速的踏板门事件，或者使用定速巡航系统时车辆突然加速或者突然减速，再或者使用自动泊车系统时，车辆突然失去控制造成事故等等。因此，安全是未来汽车发展的首要问题之一。

汽车的新功能不仅仅是辅助驾驶，还在于车辆的动态控制和主动被动的安全系统等日益触及的安全工程领域。这些功能在未来的发展和集成将更加看重安全系统研发过程，并且要求提供满足所有安全目标的证据。为了降低由于安全问题而导致的召回费用，在前期的产品设计阶段，如何采用技术手段来规避潜在风险，已成为了汽车企业迫切需要解决的问题。

IEC 61508和ISO 26262标准定义了对功能安全管理的要求、功能安全评估的要求以及管理和评估工作中所需文件的要求。功能安全管理系统的认证中，将单独的安全项目和整个公司以及其组织架构整合起来的方式，是合理有效的。

ISO 26262改编自IEC 61508，适合公路车辆的电气/电子系统的特殊应用。这些改编将应用于由电气、电子和软件构成的安全系统的安全功能的全生命周期的所有活动。随着系统日益复杂化的趋势，存在于软件和机械执行机构的系统故障和随机故障的风险也在日益增多。ISO 26262通过提供可行的要求和步骤避免这些风险的发生。

ISO 26262标准是面向目标和过程的，它强调的是一种目标导向的做法：一方面，它要求给出明确的功能和安全性目标；一方面它要求给出验证这些目标的方式；再一方面它要求给出达成目标的指标及证明。

符合ISO 26262规范的软件验证可以通过多种测试方法完成，例如代码规则检查、软件缺陷检查、单元与集成测试、软件质量及覆盖率验证等。实现这些方法的工具应该完全符合ISO 26262规范的要求，可以帮助客户实现ISO 26262规范所要求的ASIL验证工作。

下面按照ASIL4对应的标准要求逐一说明。

2 验证活动

2.1符合ISO 26262标准的静态分析

ISO 26262标准对软件静态分析的要求，可参考ISO 26262-6的Table-1、Table-4、Table-9/10/11。

下面是从Table-1 - Topics to be covered by modeling and coding guidelines中抽取的ISO 26262指南，以及工具的支持情况。

指南条款	工具	描述
Enforcement of Low Complexity 降低复杂度	QAC/QAC++ McCabe IQ	完全支持McCabe圈复杂度，具有在整个开发过程中强制执行复杂度门限的能力。
Use of Languages Subsets 使用语言子集	QAC/QAC++	广泛的相关代码规则支持审查，支持MISRAC/C++规则。
Enforcement of strong typing 强类型检查	QAC/QAC++	工具提供完整的强类型缺陷检测
Use of defensive implementation techniques 使用防御性技术	QAC/QAC++	全面的检测和多种编程标准（MISRA）的支持，可认为是防御性编程。
Use of established design principles 使用既定的设计原则	QAC/QAC++ McCabe IQ	可视化图形工具可以生成代码的结构化框架。
Use of style guides 使用风格指南	QAC/QAC++	支持定制检查器，扩展规则，提供命名规则检查、编程风格检查。
Using of naming conventions 使用命名规范	QAC/QAC++	支持定制检查器，扩展规则，提供命名规则检查、编程风格检查。

下表是从ISO 26262-6的Table-4 - Principles for software architectural design中抽取的ISO 26262指南，以及工具的支持情况。

指南条款	工具	描述
Hierarchical structure of software components 软件组件的层次结构	QAC/QAC++ McCabe IQ	可视化显示源代码，产生代码的层次结构与调用关系图
Restricted size of software components 软件组件的大小限制	QAC/QAC++ McCabe IQ	工具可度量软件组件大小，如代码行数统计
Restricted size of interfaces 限制接口大小	McCabe IQ	扇入扇出度量，可视化显示源代码，标识软件接口，从而加以限制。
High cohesion within each software component 每个软件组件的高凝聚性	McCabe IQ	提供相关的凝聚性度量
Restricted coupling between software components 限制软件组件之间的耦合性	McCabe IQ	提供相关的耦合性度量
Restricted use of interrupts 限制使用中断	QAC/QAC++ McCabe IQ	定制检查器，标识并限制中断的使用。

下表是ISO 26262-6的Table-9 - Design principles for software unit design and implementation 中抽取的ISO 26262指南，以及工具的支持情况。

指南条款	工具	描述
One entry and one exit point in subprograms and functions 子程序出入口单一	QAC/QAC++ McCabe IQ	出口点可以用返回语句的度量来标识。
Initialization of variables 变量的初始化	QAC/QAC++	全面检查未初始化变量。
Avoid global variables or else justify their usage 避免使用全局变量，或者给出使用的理由	QAC/QAC++ McCabe	全局变量可以通过架构的图形可视化来标识，确定和记录证明其使用的目的。
No implicit type conversions 没有隐式类型转换	QAC/QAC++	MISRA检查器中含有对类型转换的检查。
No hidden data flow or control flow 没有隐藏的数据流和控制流	QAC/QAC++	MISRA检查器可检测跳转
No unconditional jumps 没有无条件跳转	QAC/QAC++	MISRA检查器可检测
No recursions 没有递归调用	QAC/QAC++	MISRA检查器可检测

Table-10- Methods for the verification of software unit design and implementation 中提出的控制流分析、数据流分析、静态分析都可以通过软件测试工具自动完成，同时工具也提供Table-11-Methods for the informal verification of software unit design and implementation提出的代码Inspection审查和Walkthrough走查的辅助。

2.1.1 MISRA规则符合性检查

MISRA是汽车工业C/C++语言编程指导，是嵌入式C/C++语言的编程指导。2004年，MISRA出版了该规范的新版本——MISRA C:2004。规则总数达到了141条。共有强制规则121条，推荐规则20条。随着很多汽车厂商开始接受MISRA C编程规范，MISRA C标准已经成为业界最为著名的安全代码标准。2008年，MISRA出版了该规范的C++版本——MISRA C++:2008。

针对MISRA C检查我们推荐使用QAC工具。

2.1.2 软件质量度量

复杂度Metrics已经发展成为提供对软件复杂度和质量的客观度量方法，1976年，Thomas McCabe先生于1976年成立了McCabe Software公司，同时提出软件质量定量的度量的技术——McCabe Cyclomatic Complexity Metric（圈复杂度），能进行软件结构化测试和度量。

McCabe IQ 可以对给定的软件进行多达105种的复杂度度量，包括模块(函数/方法)级、程序级(一个或者多个源文件)、系统级(一个或者多个程序)，同时McCabe IQ给出的度量都有门限值，最大值和最小值构

成了一个”正常”或者”理想”的范围，McCabe IQ给出的门限是业界的标准，但是它们可以按照用户应用软件和环境的特殊要求调整。McCabe IQ提供软件质量的追踪，质量趋势分析。

2.2符合ISO 26262标准的单元测试

ISO 26262标准对单元测试的要求，可以参考ISO 26262-6的Table-12/13/14。

下面是从Table-12-Methods for software unit test中抽取的ISO 26262指南，以及工具的支持情况。

指南条款	工具	描述
Requirement-based test 基于需求的测试	Cantata Tessy	支持基于需求的测试用例，通过图形用户界面GUI的使用，创建基于需求的测试用例，可以指定参数的输入和期望值，返回值，全局变量，桩函数。
Interface test 接口测试	Cantata Tessy	完全支持接口的自动分析，接口数据包括参数、全集变量、外部调用。
Fault injection Test 故障注入测试	Cantata	提供Wrap封装功能，可模拟软硬件故障的发生，进行故障注入测试
Resource usage test 资源使用测试	Cantata Tessy	支持在真实目标板上运行单元测试，并通过仿真器完成测试数据下载和测试结果上传，完成资源使用的测试。
Back-to-back test between model and code, if applicable 模型和代码之间的背靠背测试（如适用）	Cantata Tessy	支持由Matlab或Rhapsody生成代码的测试

下表是从ISO 26262-6的Table-13 - Methods for deriving test cases for software unit test中抽取的ISO 26262指南，以及工具的支持情况。

指南条款	工具	描述
Analysis of requirement 需求分析	Cantata Tessy	完全支持基于需求的测试用例生成技术。
Generation and analysis of equivalence classes 等价类划分	Cantata	提供测试用例管理器
	Tessy	提供分类树方法的图形编辑环境，可以把输入和输出按照取值来分类组合而成为测试用例。
Analysis of boundary values 边界值分析	Cantata	提供表驱动的测试，自动生成接口数据的边界值。
	Tessy	根据分类树方法，设定边界值作为测试用例
Error guessing 错误推断	Cantata Tessy	可根据凭“积累的经验”和专业的判断所收集到的数据来建立。

下表是从ISO 26262-6的Table-14-Structural coverage metrics at the software unit level中抽取的ISO 26262指南，以及工具的支持情况。

指南条款	工具	描述
Statement coverage 语句覆盖	Cantata Tessy	完全支持
Branch coverage 分支覆盖	Cantata Tessy	完全支持
MC/DC(Modified Condition/ Decision Coverage) 修正条件/判定覆盖	Cantata Tessy	完全支持

2.2.1 单元测试

ISO 26262 第9节明确了单元测试证明该软件单元满足软件单元规格说明书且不包含不期望的功能。为了实现这一目标，测试工具应该支持ISO 26262 Table 12 - 软件单元测试方法，比如采用“分类树法”（戴姆勒克莱斯勒发明的）设计测试用例、自动生成基于需求的测试用例、接口测试、诸如Out-Of-Memory的故障模拟等。

2.2.2 覆盖率分析

ISO 26262 Table 14提出的语句覆盖、分支覆盖、MC/DC覆盖分析是验证测试完整性的一项指标，最基本的语句覆盖验证代码行是否执行过，分支覆盖验证所有分支判定True和false分支是否都至少执行过一次，MC/DC覆盖验证多条件逻辑判定的一部分的子条件都能独立影响条件语句本身的结果。覆盖率测试的难点在于白盒测试用例的设计，需要跟踪用例以验证用例所对应的执行路径。

测试工具一般都实现了覆盖率分析，但如果能在预先指定覆盖率目标的前提下，工具自动生成满足此目标的测试用例集合，自然是一种最佳方案。

2.2.3 支持的嵌入式平台

ISO 26262 Table 12提出的Resource usage test（资源使用测试）必须在真实目标上运行测试，而真实目标上运行测试的难点在于数据的获取。嵌入式软件测试获取测试数据有3种方式：实际的物理端口、开发工具的虚拟IO功能、读取内存数据Buffer。

实际的物理端口方式就是目标机和主机之间具备物理的通信方式，比如以太网、串口、并口、USB等，在测试的时候测试工具可以直接使用这种通信程序，和主机通信，实现测试数据获取。

开发工具虚拟IO方式，开发嵌入式软件一般需要支持交叉开发方式的开发工具，一般高级的开发工具IDE，具备虚拟IO功能，可直接进行文件操作，如：Tornado、Workbench、TI CCS等。

读取内存数据Buffer，若目标系统既没有物理通信端口，开发工具也没有虚拟IO功能，还可以采用读取内存数据的方式，修改测试工具的底层库，把输出的数据写入buffer中，在测试过程中或者测试执行后，使用开发工具读取内存的功能把缓存中的数据读取出来，上传到主机中做进一步报告分析。

2.3符合ISO 26262标准的集成测试

ISO 26262标准对集成测试的要求，参考ISO 26262-6的Table-15/16/17。

下表是从Table-15 - Methods for software integration test中抽取的ISO 26262指南，以及工具的支持情况。

指南条款	工具	描述
Requirement-based test 基于需求的测试	Cantata	支持基于需求的测试用例，通过图形用户界面GUI的使用，创建基于需求的测试用例，可以指定参数的输入和期望值，返回值，全局变量，桩函数。
Interface test 接口测试	Cantata	完全支持接口的自动分析，接口数据包括参数、全集变量、外部调用。
Fault injection Test 故障注入测试	Cantata	提供Wrap封装功能，可模拟软硬件故障的发生，进行故障注入测试
Resource usage test 资源使用测试	Cantata	支持在真实目标板上运行单元测试，并通过仿真器完成测试数据下载和测试结果上传，完成资源使用的测试。
Back-to-back test between model and code, if applicable 模型和代码之间的背靠背测试（如适用）	Cantata	支持由Matlab或Rhapsody生成代码的测试

下表是从ISO 26262-6的Table-16 - Methods for deriving test cases for software integration test中抽取的ISO 26262指南，以及工具的支持情况。

指南条款	工具	描述
Analysis of requirement 需求分析	Cantata	完全支持基于需求的测试用例生成技术。
Generation and analysis of equivalence classes 等价类划分	Cantata	提供测试用例管理器
Analysis of boundary values 边界值分析	Cantata	提供表驱动的测试，自动生成接口数据的边界值。
Error guessing 错误推断	Cantata	可根据凭“积累的经验”和专业的判断所收集到的数据来建立

下表是从ISO 26262-6的Table-17 - Structural coverage metrics at the software unit level中抽取的ISO 26262指南，以及工具的支持情况。

指南条款	工具	描述
Function coverage 函数覆盖	Cantata	完全支持
Call coverage 调用覆盖	Cantata	完全支持

软件系统体系架构

P82 基于AADL模型驱动架构设计法

P88 AADL和Simulink的关系

基于AADL模型驱动架构设计法——优化飞机综合实时仿真系统

文 | Jean Casteres, Tovo Ramaheriraryny，空中客车仿真事业部
翻译 | 西安旋极软件技术支持工程师 魏欣

摘要:

随着飞机机载设备系统日益复杂，对机载系统架构的验证成为了一个极其挑战的工作。“铁鸟”综合仿真平台正是在这种情况下开发的集成测试设备，用来尽可能模拟真实系统环境，以满足机载设备系统测试的需求。

“铁鸟”上使用的计算机主机平台和接口设备都在快速演化，要加强预测仿真应用并提升仿真器架构的性能，关键是选择合适的测试设备架构平台。

本文介绍了AADL的最新发展成果，通过AADL建模来模拟测试设备仿真器以确保目标实现。通过一个真实的工业应用演示建模仿真器实现架构器计算平台的过程。

首先，讲述基于AADL语言如何构建仿真应用模型。其次，通过生产者-消费者范式，显示如何将此概念应用在主机平台上进行仿真平台建模的过程。最后，举例证明了AADL仿真器能够替代当前使用的仿真器。

关键字：建模，实时，仿真，AADL

1. 序言

当今飞机机载系统的复杂性正在以一个极快的速度提升。航空公司的苛刻要求促使飞机制造商提出复杂的解决方案，从降低燃料消耗，增加产能，到降低噪音等。新增的复杂系统使得工程师在备选的飞机结构之间做出选择成为了特别具有挑战性的事情。

为了处理复杂性，现在普遍采用计算机系统来处理飞机制造的某些特定技术领域。而采用的仿真技术，降低了飞机复杂系统测试和平台链（从仿真器研制到集成测试平台）仿真的成本，是支持飞机项目的有效方式。

本文介绍了使用AADL进行集成测试工具的仿真结果。集成测试工具是一个模拟器，用在飞机项目的开发链上，即当飞机设备可用时，对实际飞机设备进行测试。项目的主计算机需要满足两点：满足涵盖集成测试工具发展的方向要求和基础功能。从硬件到其运行的软件，对仿真器进行建模和对仿真应用进行抽象，都是在其仿真的应用对象的基础上进行仿真器的架构定义[1]。本文介绍了该想法的实现过程以及开发环境的功能。文章结构如下：

第一部分，介绍采用AADL进行飞机仿真应用程序建模的方法。强调了AADL环境[2]以及使用的关键参数的选择。

第二部分，介绍生产者-消费者范式。描述了使用该范式的硬件建模方法。

第三部分，介绍了AADL环境中的时间处理模型。描述了仿真过程中对时间三个不同用法的根本差异，以及这些用法在测试设备仿真环境中的影响。

最后，说明确认比较的结果。把从实际测试设备上获得的度量值和使用AADL环境获得的预测值进行比较。

2. 仿真应用程序建模

A. 测试平台综合仿真平台演示

空气动力学，飞机发动机，飞行动力学，电气和液压系统涵盖了飞机装配成型的所有知识领域。在整个飞机项目周期中，不同的仿真平台参与在辅助设计和生产过程中。仿真平台覆盖仿真器研究(EPOPEE)，仿真器开

发(A/C-1和桌面仿真器)以及一体化仿真器A/C-0。它们分别支持飞机研制的不同阶段，即飞机首飞前的5年，2.5年和1年。

“铁鸟”集成概念可以追溯到协和飞机项目：各种飞机系统之间的兼容性检查可以在地面上以极低的成本进行[3]。而要实现真实的飞机系统之间的兼容性，需要在一体化模拟实验平台中进行验证和确认：因需要真人飞行员参与模拟实际的航空电子设备和界面，则必须通过实时仿真来确认。

与“铁鸟”或飞机零点（A/C-0）仿真器连接的综合仿真平台，用来辅助首飞和进行飞机认证，并作用于飞机的整个生命周期。综合仿真平台的开发过程提供以下建模能力：自然飞行返回，飞机发动机，与真实飞机设备关联的飞机环境。

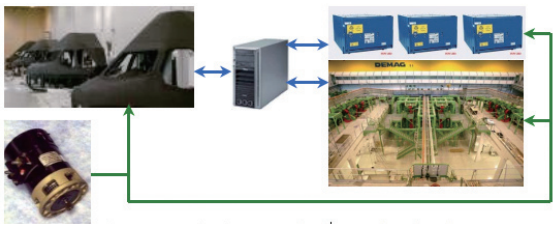


图1：综合仿真平台的基本元件

系统需要在快速改进和变化的过程中组成综合模拟器。这些变化通常来自飞机的需要。在飞机项目管理中，各个领域不同的专业知识会导致不同的仿真结果。各种测试平台可以进行合理的分离。

若想让项目达成目标，则需要引入新技术并在集成模拟器上先进行测试。主机的底层技术和集成接口需要适应升级和快速变化，这种以指令级并行和显式并行指令计算机为特色的趋势将愈演愈烈，而且这些都需要编译工具的支持[4]。仿真的目标是在开发过程中对综合仿真平台进行建模。软件应用程序和软件运行的硬件架构是在日常环境建模中的两个主要支柱，建模的目的是为了检验不同架构与技术解决方案的最优性能。

B. 仿真环境选择

文献[5]强调了常用的快速原型环境。由于综合仿真平台的实时显示功能，被选环境必须能够调度模型中的各种进程和线程。这就要求集成模拟器架构的仿真行为要在一个纯粹的模拟环境中进行。硬件和应用软件程序需要在相同的环境下快速建模以便能够形成一个快速原型环境。

UML语言、时序和时间剖面分析[6]、开源和商业解决方案如I-Logic Rhapsody、Rational Rose ，通过比较服务和性能，以上这些方法和工具都可用来搭建仿真环境。同时，集成环境如Ptolemy, Workbench, Csim, Matlab 和Scilab 都可以用于详细检查。

但是，以上方法和工具没有一个能够完全满足所有的需求和约束。最后，AADL被选中，因为它能够对复杂软件和硬件架构进行建模，具有独特的仿真能力[2]，并且具有容易将新属性添加到对象的能力。另外，在时序分析方面，AADL结合Cheddar[7]的能力，也是一个决定性优势。。

C. 应用软件模型的关键参数

为了能够模拟测试设备仿真器，环境需要满足某些约束：

- 对飞机模拟应用程序建模；
- 对主机和接口建模；
- 收集处理电源请求，IO活动，总线和网络流量的性能信息；
- 用一个可信赖的格式显示结果，以一个足够的精度水平促使架构优化；
- 为执行权衡提供一个快捷的接口去改变性能参数；
- 为了轻松地链接到其他仿真平台，需要支持两个主要的权衡点：计算能力平衡和交叉远程节点的分布仿真。

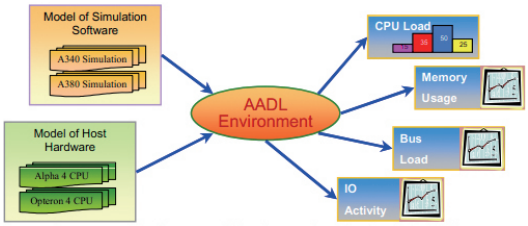


图2： “综合仿真平台”模拟器中的基本组件

用于评估各种体系结构的关键参数有：计算能力，IO活动，内存使用和总线传输负载。在综合仿真平台中承担各种任务的软件模型需要在硬件模型之上完成各种任务，如模拟处理器、阀门、接口卡和网卡总线。

整个系统是复杂的，需要一个通用概念来表示综合仿真平台架构中的所有元素。

3. 生产者-消费者 范例

以对模拟器执行流程进行建模为目标意味着在系统中执行的所有软件都有能力计算CPU使用、内存消耗、IO传输等情况。快速原型环境需要观察系统的性能和瓶颈。为了说明此目的，我们将引入一个通用范式。

A. 模拟器测试平台的仿真

使用生产者-消费范式，说明硬件设备和仿真应用程序将被组合进AADL环境中以便仿真测试平台。

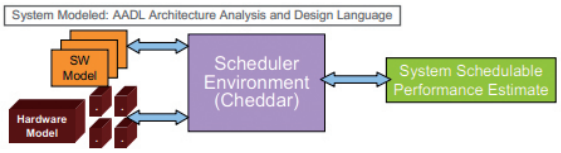


图3： AADL环境用于模拟综合仿真平台

在飞机模拟程序中，使用了建模语言AADL，对以下内容建模：

- 仿真应用软件结构
- 仿真应用硬件平台和设备

B. 生产者-消费者基本范式

进程是一个程序中的执行实例[8]（与“任务”同义）。每一个任务都有唯一的识别号及和该号码相关联的优先级。在文献[9]中解释了调度策略算法计算进程的优先级过程。

调度器可作为裁判在一组请求中根据调度策略选择一个请求进行激活。接下来将分配被激活的请求资源。

消费者向中心提出资源请求：调度者将被看作资源的生产者。

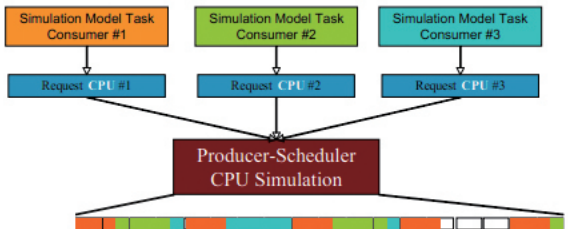


图4：生产者调度在CPU上的任务运行

在计算机世界中，任务是一个期望“运行”的进程，CPU是使期望可行的设备资源。请求CPU计算时间的任务可被看成一个能够授予CPU设备资源的调度。

集成试验台的仿真应用通过任务驱动建模。任务消耗CPU资源：一个任务代表一个使用CPU的请求，如图4所示。调度器调度任务并计算系统的负载。

建模软件的任务不仅请求时间的计算，也请求其它资源，如共享内存访问，输入/输出网络访问。在仿真环境中，软件仿真应用程序任务用一个实体表示其模型，它能够请求三种不同类型的服务：处理能力，内存和IO访问。

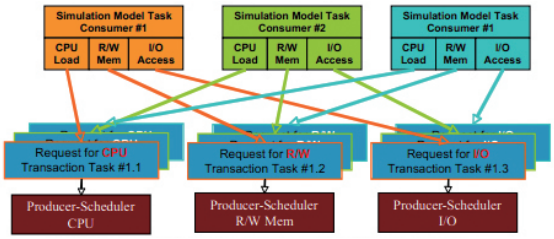


图5： 模型化软件任务的资源请求

资源的生产者不会瞬间产生所有请求资源的总量。资源的请求首先被建模，以事件和调度的方式表示请求的任务或者按序接受资源。

除处理CPU请求之外，对数据流的读/写请求（即：共享内存段）也可以转换成事务任务。同样，对数据流的I/O请求也可做此处理。

生产者-调度者对三种类型的资源以同样的方式调度事务任务。所以生产者-调度者也表示了总线的使用：如hyper-transport 或 PCI。

该方案具有实时性和独立性：首先其遵循仿真标记序列，并且在序列中反映了请求应答次序。时间通过可操作的变量保持，以便统计延迟所带来的不良影响。

同样的模式也适用于模拟通信网络的外围，如Internet。

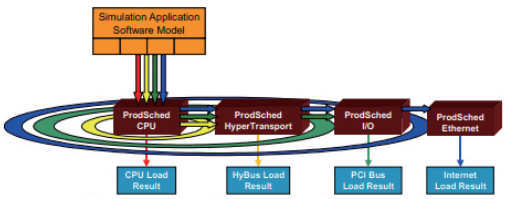


图6： 硬件设备建模范式

如上图，基础设施硬件可用三个同心环来表示其建模结构，可提供基于服务的事务给提出请求的仿真模型分配任务。不同的元素使得计算机基础设备被同化为资源的生产者和消费者，因为这些元素可以处理并发请求并仲裁它们。这是因为它们的行为类似于一个调度程序：整个环境可以使用范式建模。

C. 事务描述

仿真建模任务会消耗生产者提供的资源服务：每

个生产者是一个调度程序，该程序应答了事物请求后提供资源。事务是一个请求/反应序列，模拟了“现实”世界的资源的请求和授予，支付的性能损失被用于更新“携带”事务本身的变量。

例如，一个共享内存访问事务可以用图7表示。

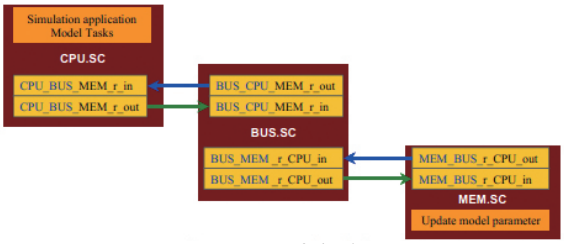


图7： 事务链

通过使用带有优先级链路的两个任务对事务建模。父任务被关联到子任务时变得活跃，子任务在任务容量耗竭时变得活跃。子任务可以在不同的调度者上，因此可以成为其他资源的生产者。优先级链路允许两个调度器代表一个事务流并被授予资源。事务从一个生产者流动到下一个生产者花费一个simulation tick。而simulation tick和仿真时间无关：事务链最后的模型只会更新变量的“时间”属性，并使用事务在现实世界耗费的时间作为模拟值。

用于事务的任务和涉及到的生产者/调度器总结如下表。

Scheduler Origine	Tâche origine	Tâche liée	Scheduler Destination
CPU	CPU_BUS_MEM_r_out	BUS_CPU_MEM_r_in	BUS
BUS	BUS_MEM_r_CPU_out	MEM_BUS_r_CPU_in	MEM
MEM	MEM_BUS_r_CPU_out	BUS_MEM_r_CPU_in	BUS
BUS	BUS1_P1_MEM_r_out	P1_BUS1_MEM_r_out	CPU

Table 1: Transaction Tasks modeling

表1： 事物任务建模

模拟环境对于AADL语言提供的任务有双重的使用：

- 软件应用程序任务建模
- 各种资源生产者之间的事务建模；任务的容量代表了事务的数量

使用抽象概念对提供的软件应用程序和硬件基础设备建模。这印证了为什么AADL语言结合AADL仿真环境会成为目标快速原型环境的最好解决方案。

4. 实时仿真的时间基准抽象

测试平台包含实际的飞行设备，该设备有自己的时间源。这就是实时约束应用于测试平台模拟器的原因。但是，快速原型环境通常是基于性能预测来做权衡的，所以它在此例中不适用。

时间是一个物理变量，在满足系统实时上限的前提下，为了评估测试平台基础设备的预见性能，需要对时间加以考虑并转换成图形。在第二级仿真中，时间是模型的结果，作为一个物理性质的考虑因素（如长度或者压力），与计算相关的参数同属一个基准。

为了把计算和模拟的环境同环境本身区分开来，环境本身的一个时间分类被提出。它并不是文献[10]一样正式的描述，而只是一个概述。

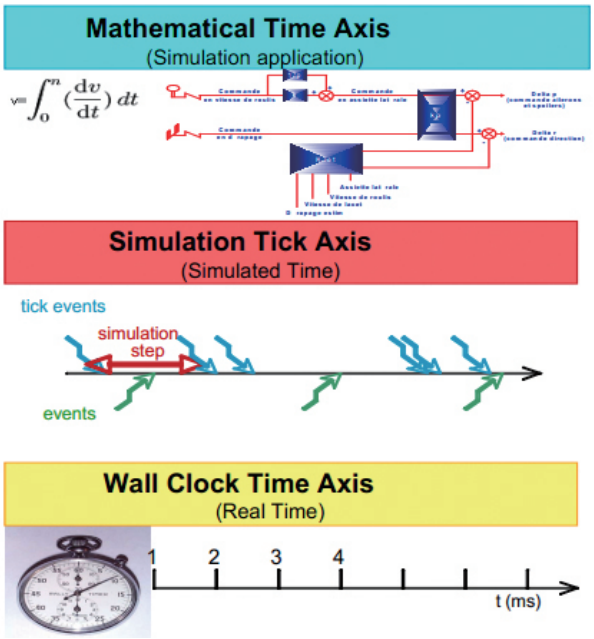


图8：时间显示范式

数学时间轴是进入模型内部计算的时间值。带有时间的变量被模型使用来计算以时间为参数的方程。

仿真时钟轴是tick与模型之间的步长同步的结果。仿真的内核处理模拟对象间的时长。仿真步长是仿真器的两个模拟时序间的同步。相对于模拟时序，仿真步长的长度没必要一致。

仿真对象可以是“ticked”或没有，在后一种情况下，对象是纯粹的基于事件的。

最后，时钟轴是实时的，即物理时间自然流逝。此轴核查数学和tick轴，以便在模拟器被请求时能满足上限并提供实时行为。

集合模拟器仿真的目的是计算性能，并提供预测运行时间。因此，时间在数学轴上，也是软件或硬件模型请求各种事务的不利因素。

生产者/消费者范式用来对系统和运载数学模拟时

间值的事务建模。对于性能计算，不同的资源生产者根据系统的处罚或延迟更新这些值。

上述的时间概念和生产者/消费者范式用于表示使用AADL语言的集成模拟器架构。

5. 确认和结果

为了使AADL环境作为未来架构选择的推断工具，必须开展验证和确认。

AADL的两个主要元素是集成模拟器软件模型和硬件基础设备模型。“真实的”模拟应用能够运行在集成模拟器上，并且性能数据可以被度量。验证方法是，比较在实际集成模拟器平台的度量和使用AADL环境获得的预测。目标是比较软件应用程序开发模型的精度，另一方面，比较硬件基础设备模型的精度。

本文中的精度是指，实时集成模拟平台度量的性能数据和使用AADL模拟环境计算的性能数据之间的差距。

A. 首先验证CPU负载活动

第一步，进行一组测试以验证相关的CPU负载性能评估；

CPU load %	Set1	Set2	Set3	Set4	Set5	Set6	Set7	Set8
Platform	18	20	26	15	23	32	50	52
AADL SimEnv	22	18	28	19	24	33	42	57
Delta	22	-10	8	27	4	3	-16	10

表2：CPU负载评估

一个“Set”代表了一系列运行在单个CPU上的模拟应用软件模型。在第一组中，由于在模型中低估了IO的传输能力，结果差别最大。然而，第一个验证活动结果鼓励了AADL环境的发展。

值得一提的是，有一组测试是在两个超级传输架构替代品进行的。一个架构关注处理器内部的通信：三分之二的超级传输链接都是关联到另外的八核架构上。另一个显示其中一个链接到IO桥。AADL环境强调不链接到接口的软件模拟将使第一个架构受益。

B. 应用程序验证

采取两个步骤来验证准确性：一为硬件模型，另一个为应用软件模型。

首先，用于验证新软件架构的环境是可选择的：确保在一个给定的基础设施上，环境能准确地模拟软件架构替代品是非常重要的。

然后，使用环境研究针对集成模拟器的不同的硬件基础设备的替代品。可以预见，AADL环境应该链接到能够模拟基础设备指定元素的其它额外模拟环境。

为了验证软件模型的准确性，硬件基础设备被固定，并选择了一组不同的应用软件。应用软件在AADL环境下被建模，并且在度量的时候运行在集成模拟平台上。

		Set1	Set2	Set3	Avg.Delta
Application #1	Platform	19	17	30	
	AADL SimEnv	16	15	33	
	Delta	3	2	3	2,67
Application #2	Platform	21	50	52	
	AADL SimEnv	15	44	59	
	Delta	6	6	7	6,33
Application #3	Platform	15	30	40	
	AADL SimEnv	11	34	39	
	Delta	4	4	1	3,00

表3：软件应用模型验证

这组集合代表了一组运行在单个CPU上的模拟软件的模型。因为负载是CPU利用百分比，Δ是平台的负载度量和AADL环境调度的预测负载的差别。

AADL环境预测的准确性介于平台度量值的2.7%和6.4%之间。该数字足够使得AADL环境有足够的信任作为快速原型环境以研究应用软件架构权衡。

为了验证硬件模型精度，软件模拟程序需运行在不同的硬件基础设备模型上。硬件基础设备在AADL环境被建模，并运行在集成平台软件模型上。当被度量时，集成平台运行“实际”应用程序。

		Set1	Set2	Set3	Avg.Delta
Infrastructure #1	Platform	19	17	30	
Conf1	AADL SimEnv	16	15	32	
	Delta	3	2	2	2,33
Infrastructure #1	Platform	24	15	22	
Conf2	AADL SimEnv	21	12	25	
	Delta	3	3	3	3,00
Infrastructure #1	Platform	22	70	30	
Conf3	AADL SimEnv	18	100	34	
	Delta	4	30	4	12,67
Infrastructure #1	Platform	13	30	40	
Conf4	AADL SimEnv	11	34	39	
	Delta	2	4	1	2,33
Infrastructure #2	Platform	20	20	45	
Conf1	AADL SimEnv	18	17	41	
	Delta	2	3	4	3,00
Infrastructure #2	Platform	30	80	40	
Conf2	AADL SimEnv	25	100	37	
	Delta	5	20	3	9,33
Infrastructure #2	Platform	20	37	40	
Conf3	AADL SimEnv	19	39	39	
	Delta	1	2	1	1,33

表4： 硬件基础设备模型验证

一系列的AADL仿真环境集模型被分配给一个单独的CPU。因为负载是CPU的利用率，Δ是平台的负载度量和AADL环境调度的预测负载的差别，如前面所述。

AADL环境预测的准确性介于平台度量值的2.4%和12.7%之间。该数字足够使得AADL环境有足够的信任作为快速原型环境以研究应用软件架构权衡。

上述表中显示的数字表明快速原型环境把开发的AADL环境置于一个非常高精度范围内 。一个有着两个

主要根源的小的变化结果可以观测到：性能度量小的变化被用于订阅AADL对象的参数。第二个，一些软件模型的IO传输是AADL对象的参数。这个参数可以度量，但对于某些模块必须评估。

另外一个收获是在初始化和最终的验证步骤之间。为了获得更为精确的数字，需要改变在实际平台上度量应用程序的工具。引入一个基于Linux主机的允许使用更为精确的性能度量新工具，这些工具应用在软件模型上，并从AADL环境中获得更为准确的预测。

6. 总结

集成模拟器使用的环境允许预测所需基础设施的误差，其范围在5%到15%之间。

对复杂系统进行仿真，正在并将越来越多的使用。使用的模型用更高程度的保真度反映现实，因此要求仿真基础设备有更高的性能。

仿真世界的融合已经发生：飞机仿真和其它学科的共性已经发现，这为对更加复杂的系统进行系统仿真提供了依据。

对新领域进行探索会对建模技术带来新的挑战：仿真架构需要提供更好的性能服务，硬件基础设备需要更新。这也同样适用于低成本的主流解决方案采用的输入/输出架构。

有三方面强调需要快速原型环境，其中一方面上述已经提及。

此外，新的虚拟技术最终将所有产业的各方面参与系统建设；把更多的关注用于仿真架构以容纳模拟的所有组件。

这些新的挑战使仿真学科成为一个充满激情的领域，也强调了在仿真领域需要标准化的形式体系。AADL环境是系统仿真架构预测一个很好的候选对象。本文的结果显示了对于今天项目的复杂目标进行仿真器仿真的精度和可信性，并有助于标准化倡议。

参考文献：

[1] JM Calluaud, J Casteres, S Gaudaire,”Technology evolution of aircraft simulator for real equipments validation”, Erts 2008, p5, 2008

[2] As-2 Embedded Computing Systems Committee, “SAE Architecture Analysis and Design Language (AADL)”, AS5506/1, June 2006 see also: AADL home and Telecom Paris website.

[3] C. Coureau, D. Liot, B. Mattos, « Airbus Engineering Simulation Methods -Past and

Future Trends », p5, § 5, 2004

[4]John L.Hennessy, David A. Patterson, “Computing Architecture: A quantitative Approach”, Third Edition, Morgan Kaufmann, p376, 2003

[5] G. Laurens, “Prototypage rapided”un Simulateur et étude des performances”, Internship Airbus, p18, 2006

[6] The OMG, “A UML Profile for MARTE”, Modeling and Analysis of Real-time and Embedded Systems (MARTE), OMG doc. #: ptc07-08-04

[7] Cheddar : a Flexible Real Time Scheduling Framework. F. Singhoff, J. Legrand, L. Nana, L. Marcé. ACM SIGAda Ada Letters, volume 24, ACM Press, New York, USA. December 2004, ISSN:1094-3641. F. Singhoff and A. Plantec. “What is Cheddar”, at AADL scheduling home site .

[8]W.RichardStevens,”Advanced Programming in The Unix Environment”, Addison-Wesley, 1993

[9] Giorgio, C. Buttazzo, “Hard Real-Time Computing Sysyems: Predictable Scheduling Algorithms and Applications”, Kluwer Academic Publisher, 1997

[10]Bernard P. Zeigler, Herbet Praehofer, Tag Gon Kim”Theory of Modeling and Simulation”, 2nd Edition, Academic Press Elsevier 2000

[11] Gerard Fleury, P. Lacomme, A. Tanguy “Simulation à Evènement discrets”, Eyrolles, 2007

[12] Jean Casteres, Tovo Ramaherirariny ,Aircraft integration real-time simulator Modeling with AADL for architecture tradeoffs

AADL和Simulink的关系

文 | Peter Feiler, Carnegie Mellon University
翻译 | 西安旋极软件技术支持主管 张琨

Simulink建模工具用于仿真控制系统和受控设备。其模型仅能开发一个控制器，如电子舵机控制器，或者具有相互作用的控制系统，如巡航控制和刹车系统。每个控制器和受控设备部件可采用同一级别的多个仿真模型表示。

当依据模型进行模拟运行时，仿真引擎确定所有控制器和受控设备模型块执行的顺序。换句话说，控制器模型块和受控设备模型块不是并发执行的。

可以通过实时处理工具将模型块转换为可执行的代码。经过实时工具处理后，模型块可被转换为单个可执行程序。既可以为控制器产生单个程序，也可以将控制器与设备的仿真模型合并转换为可执行程序。

AADL的主要作用是定义嵌入式应用系统的运行时架构，该架构由一组运行在单处理器或多处理器上的一组并发线程（任务）构成。这些线程可以是真实的手动设备或仿真受控器模型，也可以是面向真实物理设备或物理设备仿真的连接。

在下面的章节里，我们用多个应用例子中说明AADL与Simulink建模工具之间的相互作用。

接口定义文件

我们假设用户正在建立一个控制器的Simulink模型。首先，用户需要定义受控设备的接口，建立受控设备的模型。

AADL将控制器定义为AADL控制部件，将受控设备定义为AADL设备部件。参考Simulink模型文本文件定义维系每个部件。

对于控制部件和设备部件，用户定义端口并设定与端口相关的数据特征。Matlab/Simulink通常允许用户定义数据的基本类型，例如有符号/无符号的8/16/32位整型等，并将识别出与基本类型不匹配的数据类型。用状态流、控制流相关的数据字典定义应用数据类型。

在AADL规范中定义控制部件传递的数据特征，如基本类型（例如uint16）和应用数据类型（例如speed）。其中，用户能定义传输数据的单位（比如kph或mph）和值域的范围；定义数据流的特性，比如连续变量的最大变化率（即不连续的拐点）；或者数据流中可能出现的各种错误的可能性（例如传感器延迟，或处理数据超时。通过在部件和接口上的注释，可以注释一些关于部件的其它文件信息，如丢失数据等）。 [SVM 2004, Krogh 2003, Krogh 2004, Emmeskay 2006]。

AADL也能处理如“控制器对受控设备发出控制命令时的延迟时间是多少？”这样不明确的文件条款。伊利诺伊大学香槟分校的工作已经证实了AADL这种Assumption Management Framework (AMF)的能力。

模拟块之间的接口

我们充分利用AADL的优势，将会更加准确地定义部件（块）之间的接口。这些接口关系来源于Simulink。

从Simulink模型中提取架构信息并进行注释。问题是当Simulink模型改变时，如何保证AADL模型同步变化。这个问题有两种解决方式：

第一种方式，需要系统验证管理工具集来保证修改的同步[SVM 2004]。用户控制从Simulink模型中提取架构模型的详细程度，然后对架构模型进行注释。如果Simulink模型发生改变，提取架构的方法能够识别当前

架构模型和新提取的架构模型之间的区别，从而相应地更新当前模型，确保在新旧体系架构模型中不丢失任何部件的注释信息。

这种方式较接下来的第二种方式更为先进一些。第二种方式中，一旦建立起AADL体系结构模型，Simulink模型被分割成适当的分片，每个分片对应AADL体系结构模型中的一个节点。于是这些Simulink分片被记录为架构模型的组成部分。用户的操作方式为，在架构模型中操纵架构，而通过模型分片修改模型的算法部分。这样就从分片和架构中产生复合的Simulink模型。

总之，我们在一个部件内建立足够并发执行的信息。这虽不是AADL关注的主要焦点，但通过它也确实能支持部件的并发执行，并可用来补充模型的一致性检查。而一致性检查则正是部件建模语言所欠缺的功能。

不同精度的模型

通常情况下针对某一个物理部件存在不同精度的Simulink模型，控制算法也是多种多样。这些模型表示的是同一个系统，从概念上来讲应具有相同的系统架构。

在AADL中，部件类型描述部件的外部接口，部件实体描述部件的具体实现；同一部件只有一个部件模型，但可以有多个部件实现。对应不同的Simulink模型，或是由于部件的内部结构不同，这种情况下可由多个部件实现（即子部件及其连接关系，子部件本身对应于Simulink模型或Simulink模型的内部块）。文献[Feiler 2006]中更加详细地描述了如何建模系统多样性和系统配置。

系统确认管理工具集（SVM2004）已经解决了这个问题，并说明了模型多样性如何关联到相同的架构模型，以及怎样从单个的Simulink模型中导出同一架构的AADL模型。SVM的重点是通过执行自动化验证过程（仿真和模型检测），自动化地进行模型确认，从而保证需求得以满足。这项工作目前正在被Emmeskay商业化[Emmeskay2006]。

并发执行单元

在项目开发过程进行到一定程度时，根据Simulink模型建立的概念系统架构需要转换为嵌入式软件系统的具体实现。AADL主要关心的问题，就是接下来要描述的嵌入式软件系统的运行时架构。

用户使用AADL并发执行单元线程，定义实际运行的系统架构、周期、基线、最长执行时间（worsecase execution time）。类似的，用户也可以定义任务之间的相互通信及采样数据的时间特征（包括立即和延迟通信）。

线程代码可以通过Real-Time Workshop从Simulink模型产生。换句话说，AADL模型的注释信息可以产生构建脚本，自动生成代码并将代码段加载到运行映像中。

Honeywell公司已经使用MetaH工具实现了这一功能。MetaH是AADL标准的源头，它由Honeywell公司在DARPA基金资助下开发，用于多个飞机项目中。Honeywell与美国陆军AMRDEC合作，将MetaH工具集用于高度不稳定的强实时控制系统中[Feiler 2000, McDuffie1, McDuffie2]。其时，他们对相应的Beacon工具集进行了版本修定，可以自动地从控制系统模型产生MetaH模型，这些模型有控制系统、硬件架构（包括单、多处理器配置），从软件到硬件的绑定形成可实时执行的控制系统，软件运行效率高且占用内存资源少。后来，由DARPA MoBIES项目支持，Carnegie Mellon大学执行的TimeWeaver/SysWeaver项目也使用了这一工具，包括自动代码生成和Simulink模型的链接。

用户还可以在AADL进程中指定受保护的地址空间，这是容错系统划分的需要，在运行时限制错误和故障的传播。如果重视软件的可靠性，则AADL的附录“Error Model Annex”[AS5506/1 2006, Feiler/Rugina 2006]可用故障信息的方式注释在AADL架构中，包括可靠性、可用性、完整性和故障树分析。以随机过程模型和故障树为表现形式的可靠性模型支持风险分析，以AADL模型和错误模型自动生成的FMEA，利用AADL模型的连接和绑定信息，可以确定所有错误传播路径。这种能力已在MetaH工具集中得到支持，并且包含在开源AADL工具环境（OSATE）中。

集成系统的Simulink模型

人们已将Simulink作为一个体系结构建模工具。子

系统的块用于表现一般部件，部件的多样性则由可配置的子系统块表示。

类似的，在同一个项目中，需要将多个独立子系统（如油门控制，刹车系统，巡航控制）集成到一个系统模型中。在Simulink工具中可以做到这一点，其方法是将表示应用程序的若干子系统模块组合形成控制系统模型，同时采用同样的方法形成受控系统模型。至此，我们从Simulink中获得了并发执行任务的子模块（表示某一部件的模块）及执行的相关特征信息。

一种可能的方法是定义一组Simulink模块，和AADL实时运行架构模型的一一对应关系。使用Simulink作为图形输入前端，建立和维护实时运行系统架构，然后将这个模型通过文本或XML格式转换成AADL模型，这样充分利用了分析工具的优势，并能方便地构建嵌入式系统模型。

反之，也可以将AADL架构模型转换成Simulink模块库，用户就可以在Simulink环境中熟悉和了解所建的模型。

模拟设备

AADL允许用户指定其嵌入式系统软件运行的计算机平台，例如，处理器、内存、以及总线/网络。AADL也允许用户表示与嵌入式系统交接的物理设备。使用AADL设备组件，用户定义设备和计算机平台组件之间的连接，定义应用软件的逻辑接口，包括端口/连接、共享访问、以及子程序调用服务。

设备可以用来表示成独立的传感器和执行器，或者表示成带有传感器和执行器端口的完整设备。在相关的仿真运行环境中，设备实现可以定义成纯软件设备及由设备生成的不同数据流。硬件在回路（in the loop）的嵌入式系统中，不同的设备实现具有不同的硬件驱动软件和硬件初始化参数，能很好地实现软件硬件绑定。

参考文献

AADL and Simulink by Peter Feiler

2. [SVM 2004] System Verification Manager, A DARPA MoBIES funded project led by Prof. B. Krogh at CMU/ECE, http://www.ece.cmu.edu/~webk/svm/

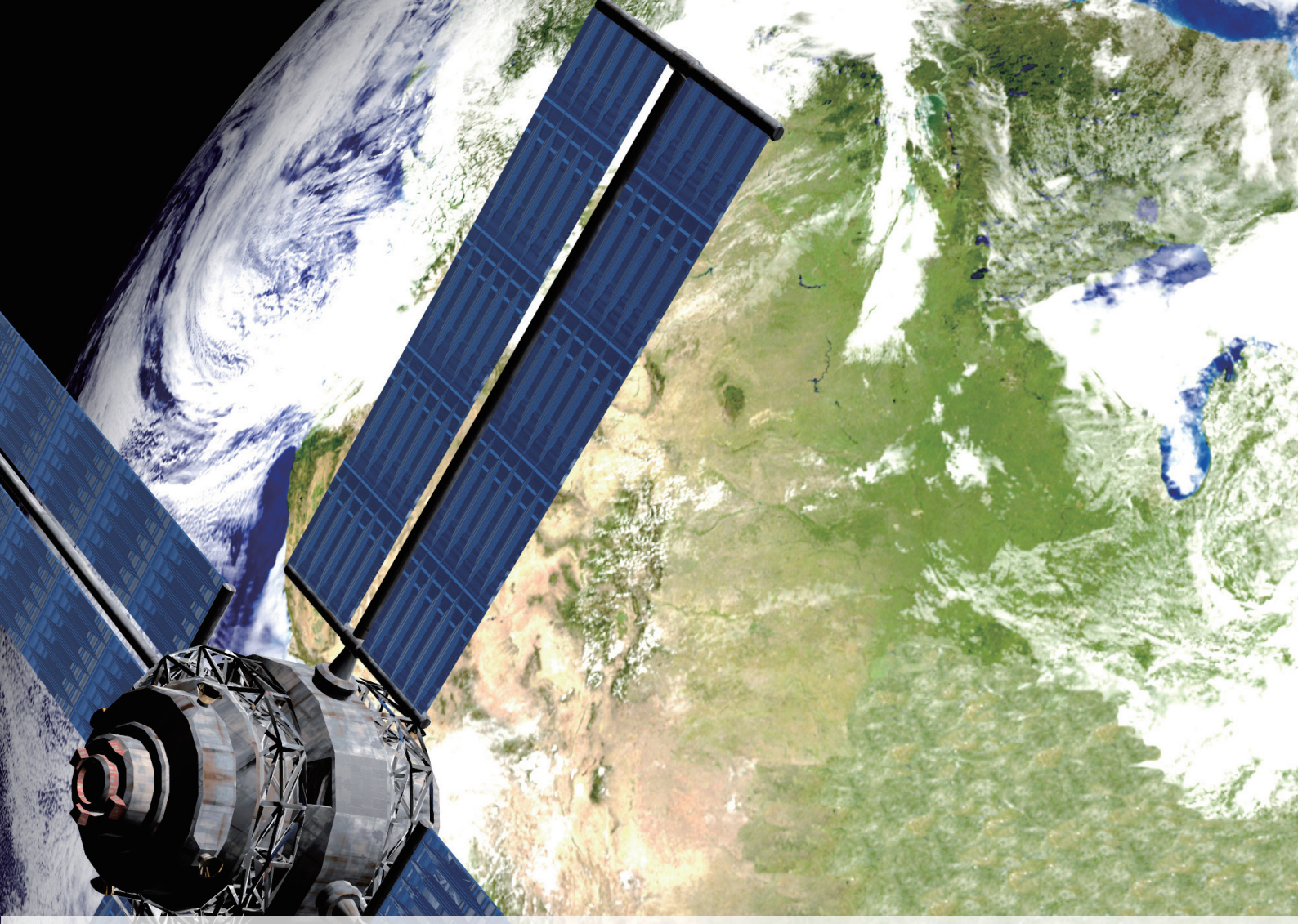
3. [Krogh 2003] Bruce H. Krogh, Peter H. Feiler, Shiva Sivashankar, and Bill Aldrich, "SVM: System Verification Manager for Model-Based Development of Embedded Control Systems", Proceedings of EMSoft, July 2003.

4. [Krogh 2004] Bill Aldrich, Ansgar Fehnker, Peter H. Feiler, Zhi Han, Bruce H. Krogh, Eric Lim, and Shiva Sivashankar, "Managing Verification Activities Using SVM", Proceedings of Sixth International Conference on Formal Engineering Methods (ICFEM), Nov 2004.

5. [Emmeskay 2006] John Absmeier (Delphi Corporation), Tuhin Das, Swaminathan Gopalswamy, Ravi S. Paike (Emmeskay, Inc), "Development of an Automated Control System Verification Platform for a Solid Oxied Fual Cell", Proceedings of the 4th International Conference on Fuel Cell Science, Engineering and Technology (Fuel Cells 2006), June 2006, Irvine, CA.

6. [Feiler 2000] Peter H. Feiler, Bruce Lewis, Steve Vestal, "Improving Predictability in Embedded Real-Time Systems", Software Engineering Institute, MU/SEI-2000-SR- 011, 2000, http://www.sei.cmu.edu/publications/documents/00.reports/00sr011.html.

7. [Feiler 2006] System Configuration With AADL, P. Feiler, Technical Report, draft,2006.



产品应用

P92 RBT在软件黑盒测试用例设计中的应用

P95 需求追踪工具化对机载软件研制的影响

P99 基于CANTATA++的软件单元测试

P103 McCabe IQ在白盒测试中的应用体会

P105 AdaTest95在型号控制软件中的应用

RBT在软件黑盒测试用例设计中的应用

文 | 上海机电工程研究所 殷文雄

一. 引言

软件测试可以分为黑盒测试和白盒测试两类。对于白盒测试，有诸如语句覆盖率，分支覆盖率，MC/DC覆盖率等度量对测试的完备性进行衡量。而对于黑盒测试，同样也需要对测试完备性的衡量，而仅仅依靠需求跟踪矩阵的手段进行衡量测试完备性过于粗粒度，因此寻找一种能保证测试完备性的黑盒测试用例设计方法显得非常有必要。

本文将通过案例来介绍如何利用RBT工具，在软件黑盒测试中采用因果图法设计用例，保障测试完备性，同时提高软件测试水平和错误发现率。

二. 因果图法

因果图法是从需求规格说明书的描述中找出因（输入条件）和果（输出或程序状态的改变），通过因果图转换为判定表，最后为判定表的每一列设计一个测试用例。这种方法的起源是硬件数字集成电路的测试方法（敏感路径法），能够设计出有效的测试用例，而舍弃对测试没有贡献的测试用例。并且覆盖全部功能。

因果图的主要步骤有：

1)分析软件规格说明描述中， 哪些是原因(即输入条件或输入条件的等价类)， 哪些是结果(即输出条件)，并给每个原因和结果赋予一个标识符。

2)分析软件规格说明描述中的语义，找出原因与结果之间， 原因与原因之间对应的关系，根据这些关系，画出因果图。

3)由于语法或环境限制， 有些原因与原因之间,原因与结果之间的组合情况不可能出现，为表明这些特殊情况， 在因果图上用一些记号表明约束或限制条件。

4)把因果图转换为判定表。

5)把判定表的每一列拿出来作为依据,设计测试用例。

三. RBT的介绍

Bender-RBT 是基于需求的测试用例设计工具，通过分析和鉴别系统的需求说明来设计最小数目的测试用例并满足最大的功能覆盖。通过评估应用系统的需求分析来鉴别错误或逻辑上的不一致。

Bender-RBT有两个主要功能，因果图和正交法。本文主要介绍RBT的因果图功能，RBT针对因果图提供了两点非常有效的支持：

- 针对上节因果图的步骤，RBT提供从第一步至第三步的可视化操作环境，用户可以在RBT提供的界面上绘制因果图的各节点和节点之间的关系。
- 针对上节因果图步骤中的第四步和第五步，RBT实现因果图至用例的自动转换，用户无需再绘制复杂的判定表。

四. RBT案例实现

应用RBT的因果法设计测试用例的具体步骤为：

1）分析软件规格说明书对应某条功能的描述中哪些是原因，哪些是结果。原因是输入或输入条件的等价类，结果是输出条件；

2）在RBT上绘制出对应的节点，并用记号表明约束条件或限制条件；

3）对需求加以分析并设置节点与节点的关系，表示为因果图之间的关系图；

4）利用RBT生成测试用例。

案例为某嵌入式软件的两段需求，具体内容如下：

该嵌入式软件对探测设备输出的“捕获信号”进行判断，若已经捕获信号，控制“绿灯亮控制指令”输出低电平绿灯亮，控制“音响控制指令”输出低电平音响响，提示已捕获；否则判断探测设备未捕获信号，控制“绿灯亮控制指令”输出高电平，控制“音响控制指令”输出高电平，无声光信号提示，继续捕获信号。

当捕获信号后且“第一档”被用户按下后，软件检测输入信号“第二档信号”，若在0.5秒内检测到第二档按下的高电平，则执行“自动操作”；若在0.5秒后检测到第二档按下的高电平，则执行“手动操作”；若未检测到第二档按下的高电平，则不执行操作。

首先从需求中分析出原因节点，中间结果节点以及结果节点。

● 原因节点：捕获信号；第一档信号按下；第二档级延迟0.5s按下；第二档级立刻按下；第二档弹开。

● 中间结果节点：第一档按下第二档弹开；第一档与第二档皆弹开。

● 结果节点：绿灯状况；音响状况；手动操作；不操作；自动操作。

分析完毕后，在RBT中设计因果图，打开RBT软件，选择因果图模板，建立名为“RBTExample”的因果图工程。依照以上分析创建12个节点。每个节点的具体内容见表3-1，图3-1显示节点CatchInfor对应的设置界面，图3-2为添加完毕后，“RBTExample”工程显示界面。表3-1

Node Name	True State Description	False State Description
CatchInfor	捕获信号	未捕获信号
TriggerFirstPush	第一档按下	第一档未按下
TriggerSecondPushDelay	第二档级延迟0.5s按下	/b
TriggerSecondPushImmediate	第二档立刻按下	/b
TriggerSecondRealse	第二档弹开	/b
GreenLightOn	绿灯亮	绿灯暗
VideoOn	音响响	音响不响
AutoOp	自动操作	/b
HandOp	手动操作	/b
FirstPushSencondRelease	第一档按下，第二档放开	/b
FirstAndSencondRelease	第一档与第二档放开	/b
UnOp	不操作	/b

（注，添加节点时，节点名称不能为中文，因为软件并不支持中文名称，而在True State Description，False State Description对应栏中可以填写中文，测试用例生成后，将以该部分作为用例描述显示出来。）

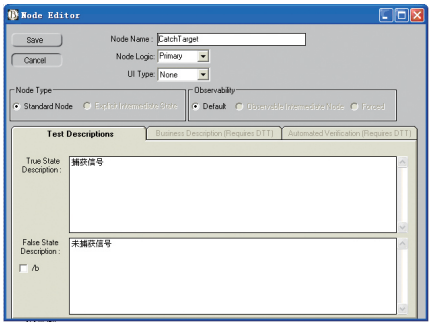


图3-1

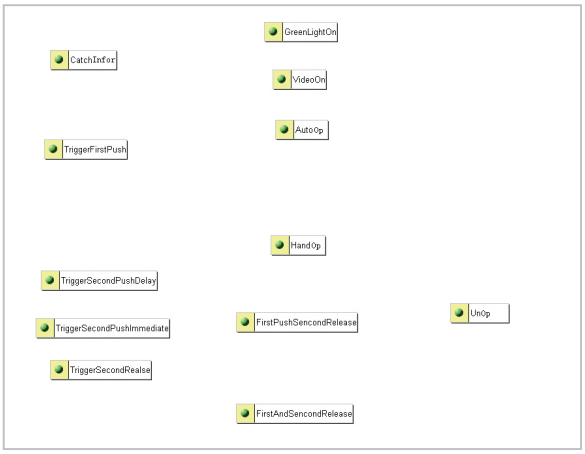


图3-2

对于原因节点，它们之间还存在约束关系，由于硬件约束，用户必须在按下第一档，接着才能按下第二档，故节点TriggerSecondPushDelay和节点TriggerSecondPushImmediate依赖于节点TriggerFirstPush，需在RBT中添加REQ约束。另外节点TriggerSecondPushDelay，TriggerSecondPushImmediate和FirstAndSencondRelease是互斥的，有且只有一个为真，故需要为这三个节点添加ONE约束。图3-3显示添加约束后的“RBTExample”工程显示界面。

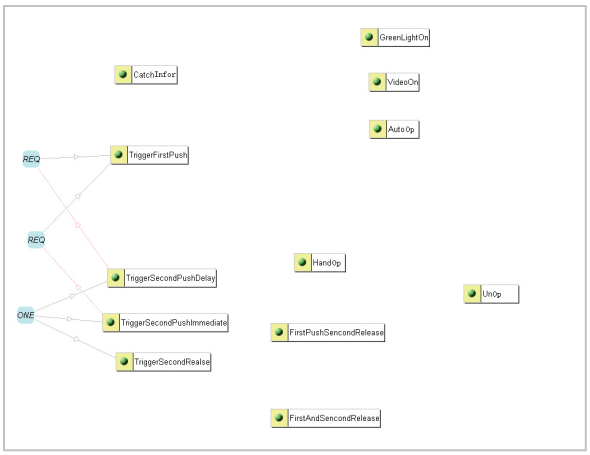


图3-3

然后需要设置因果图节点之间的关系，对于结果节点GreenLightOn和节点VideoOn而言，若要节点CatchTarget为真，则该两节点都为真，反之则都为假，故将节点GreenLightOn与节点CatchTarget连接，将节点VideoOn与节点CatchTarget连接，关系都选为simple。

对于结果节点HandOp，若要节点HandOp为真，则节点CatchOp，节点TriggerFirstPush和节点TriggerSecondPushDelay都得为真，故将节点CatchOp，节点TriggerFirstPush和节点TriggerSecondPushDelay与节点HandOp相连接，关系选为and。

对于结果节点AutoOp，若要节点AutoOp为真，则节点CatchOp，节点TriggerFirstPush和节点TriggerSecondPushImmediate都得为真，故将节点CatchOp，节点TriggerFirstPush和节点TriggerSecondPushImmediate与节点AutoOp相连接，关系选为and。

对于中间结果节点FirstPushSencondRelease，要求TriggerSecondRealse以及TriggerFirstPush皆为真，故将节点TriggerSecondRealse和节点TriggerFirstPush与节点FirstPushSencondRelease相连接，关系选为and。

对于中间结果节点FirstAndSencondRelease，要求TriggerSecondRealse为真且TriggerFirstPush为假，故将节点TriggerSecondRealse和节点TriggerFirstPush与节点FirstAndSencondRelease相连接(其中节点TriggerSecondRealse和节点TriggerFirstPush连线属性选择为FALSE)，关系选为and。

对于结果节点UnOp，若要节点UnOp为真，则节点CatchInfor为假或者节点FirstAndSencondRelease为真或节点FirstPushSencondRelease为真，故将节点CatchInfor，节点FirstAndSencondRelease和节点FirstPushSencondRelease与节点UnOp相连接(其中节点CatchOp和节点UnOp连线属性选择为FALSE)，关系选为or。

图3-4为完成建立节点之间连接后的“RBTExample”工程显示界面。

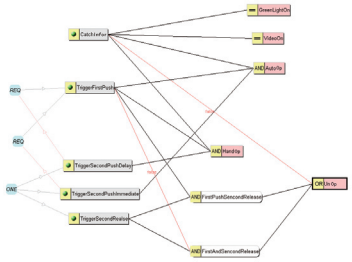


图 3-4

由于因果图存在中间结果节点，所以可能导致不可观测的情况发生。通过对实际情况进行分析，中间结果节点在实际测试过程中是可以观测到的，因此需要设置节点FirstAndSencondRelease和节点FirstPushSencondRelease为可观察节点。点击节点属性，在Observablity一栏中选中Observable Intermediate Node选项，如图3-5。

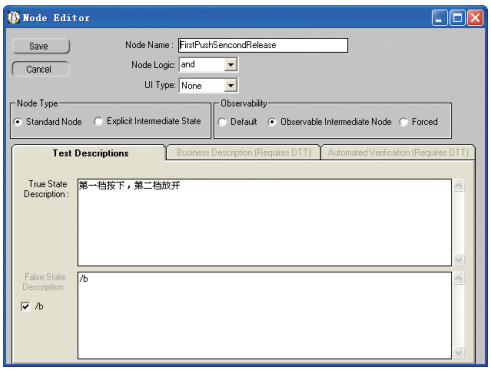


图3-5

因果图设计完毕后，点击Generate,在弹出对话框中点击Run New,运行分析因果图，运行完毕后选择其中的Batch Test Defination Report，即为测试用例，见图3-6。

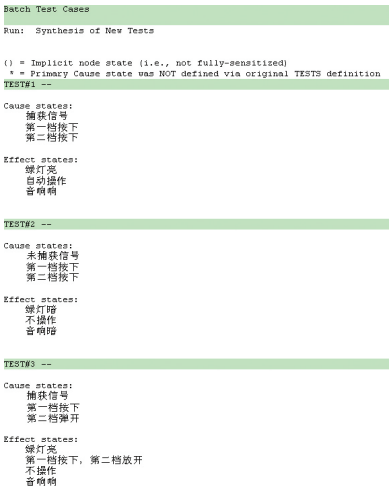


图3-6

五. 总结

RBT为测试人员利用因果图方法测试软件提供了极大的便利，但前提是需要测试者绘制出正确的且有效的功能因果图。因果图所需的信息一般是从需求规格说明，概要设计说明中获取，因此对文档描述的具体性和准确性也有一定的要求。

需求追踪工具化对机载软件研制的影响

文 | 上海飞机设计研究院 吴健

摘要：本文首先介绍了DO-178B对需求追踪的要求，然后对需求追踪活动的特点进行分析，得出需求追踪工具化的必要性，进而分别介绍了三款业界主流的需求追踪工具，最后结合机载软件实际研制过程，论述了需求追踪工具化对软件开发、验证、质量保证和软件认证的影响，旨在为机载软件项目经理在计划阶段选择合适工具实现DO-178B中“需求追踪”的目标提供参考。

关键词：需求追踪 工具 DO-178B

一、DO-178B对需求追踪的要求

DO-178B在第5.5节“可追踪性”和第6.4.4.1节“基于需求的测试覆盖分析”中提出了需求追踪的要求。机载软件的合格审定要求申请人：

1） 提供系统需求和软件需求之间的可追踪性，以验证系统需求的完全实现和显示派生需求。

2） 提供低级需求和高级需求之间的可追踪性，以显示派生需求和软件设计过程做出的结构设计决策，并验证高级需求的完全实现。

3） 提供源代码和低级需求之间的可追踪性，以验证没有未实现的源代码以及低级需求的完全实现。

4） 每一项需求都有测试用例。测试用例满足正常范围和鲁棒性测试的准则。

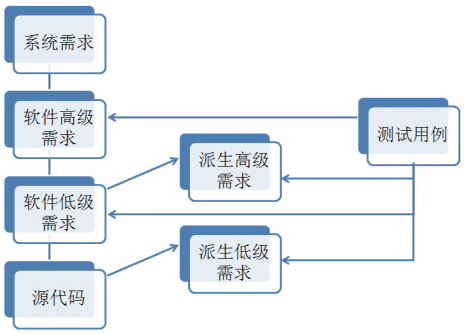


图1：DO-178B对需求追踪的要求

以上四项要求适用于软件等级为A、B和C的机载软件。A级软件更是对基于需求的测试覆盖分析提出了独立验证的要求。

二、需求追踪的工具化

DO-178B提出的需求追踪的概念简单明确。项目经理会根据机载软件项目的不同而选择不同的实现方法，并写入软件开发计划。人+过程+工具=产品。项目经理总是先会想到使用一个适合的工具来满足这个软件认证的目标。

一种简单的做法是在下一级生命周期资料中直接写入上一级被追踪资料的信息（比如需求标识或需求内容）来说明可追踪性。但是这种做法只建立了自下而上的单向追踪关系的，而且难于提取可追踪性信息。如果要说明上一级资料的变更会影响哪些下一级资料以及哪些资料是派生需求，就需要维护一张双向的追踪关系表。

最容易想到的一种做法就是采用Microsoft EXCEL样式表来维护这张追踪关系表。项目成员大都熟悉EXCEL的操作，而且通过VBA编程可以扩展EXCEL功能，建立这张追踪关系表应该不成问题。不过，这种做法的缺点在于维护困难。软件开发中唯一不变的就是变更。系统需求、软件需求、源代码、测试用例的数量庞大，级别不同，类型各异，在如此庞大的数据量面前，EXCEL的执行效率会明显降低。另外，由于以上生命周期资料都存放在配置管理库中，EXCEL维护的追踪关系表中数据的完整性和正确性还需要通过人工来检查。即使有人二次开发了EXCEL与配置管理库的接口来自动检查数据来源的完整性和正确性，由于EXCEL及其扩展的工具鉴定非常困难，人工检查仍然必不可少。

目前业界有三款工具很好地解决了上面出现的问题，在机载软件研制项目中得到了广泛应用。

1、DOORS

IBM公司的Rational DOORS是业界公认的全球使用范围最广的需求管理工具。在DOORS中可以输入需求，也可以从多种格式的文件(Microsoft Word, Excel, PowerPoint, Outlook，普通文本或RTF文件)导入需求。民用飞机主制造商（AIRBUS、BOEING、COMAC）和大型航电供应商（Honeywell、Rockwell Collins、GE Aviation）都使用DOORS管理客户需求和系统需求。基于系统工程师对DOORS更加熟悉和工具环境一致性的考虑，机载软件需求管理也会继承性地选择使用DOORS。另外，绝大部分机载软件研制使用的工具都有和DOORS的接口，支持需求的导入和导出。

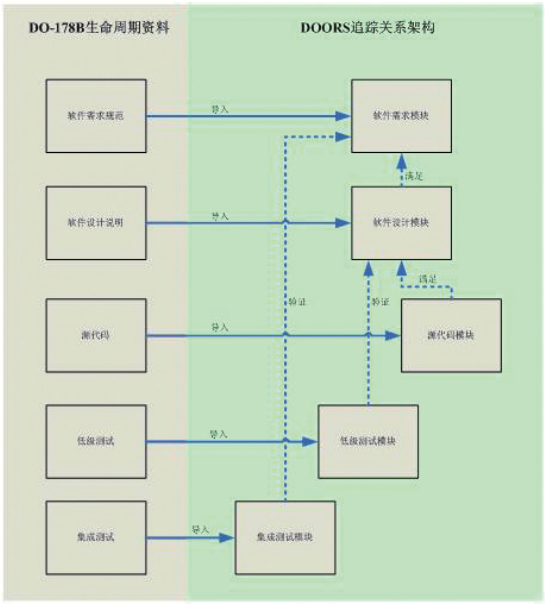


图2 DOORS中DO-178B生命周期资料的追踪关系架构

要在DOORS中进行需求追踪，需要为每个被捕获的需求指定入向链接和出向链接，然后就可以选择使用可追踪性分析、追踪列(Traceability Column)和追踪浏览器（Traceability Explorer）等多种DOORS的功能来跟踪与管理这些需求了。

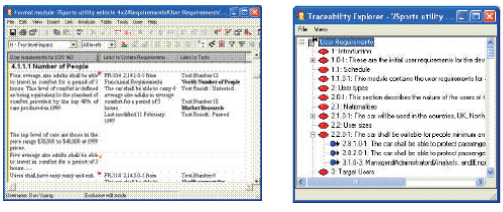


图3 DOORS的追踪列和追踪浏览器[3]

2、VeroTrace

美国的Verocel公司有很丰富的RTCA/DO-178B认证项目的经验，其客户包括WindRiver、AdaCore、Aonix、GE Aviation、Fenwal Safety Systems、BOEING、TTTech等。Verocel总结和提炼其适航认证的经验，以需求管理为核心，开发出VeroTrace这个功能强大的工具。

VeroTrace提供图形用户接口，用户可以定义和编辑：

1） 系统需求；

2） 软件高级需求；

3） 软件低级需求。

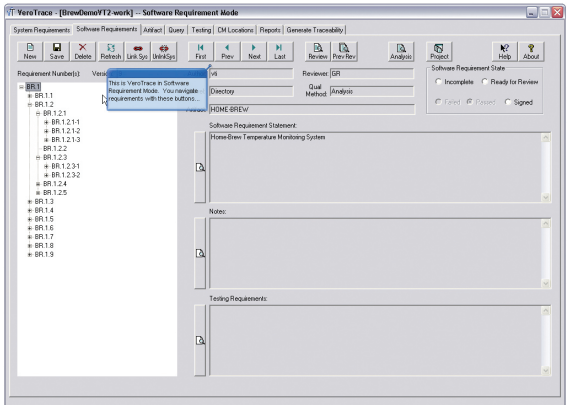


图4 VeroTrace的图形用户接口[4]

如果以上需求存在于Microsoft Word文档或DOORS需求管理工具中，VeroTrace还提供工具帮助直接导入这些需求。

VeroTrace支持从配置管理库中提取工件（Artifact）。和需求相关的工件，例如设计文件，源代码，测试用例，测试程序，测试结果，覆盖率分析结果，都能在VeroTrace中查询、添加、修改和删除。

VeroTrace支持建立和检查需求与需求以及需求和工件之间的双向追踪关系，实现需求和工件的完全追踪。需求之间更是支持建立横向和纵向两种追踪关系，既可以将系统需求和产品需求做横向的对应，还可以将产品需求进一步分解形成软件需求。另外，派生需求可以在VeroTrace中明显地标识出来，存入可追踪性数据库。不正确的和丢失的追踪关系可以通过VeroTrace的验证工具自动检查到，结果更加准确。

VeroTrace支持在线评审需求和工件。从配置管

理提取工件用于评审，需求和工件的评审意见被记录到可追踪性数据库，自动生成评审检查表，并检入检查表到配置管理库。通过工具设置和数字签名策略，VeroTrace可以强制要求需求和工件作者与评审人员不是同一个人，从而满足高软件级别软件的某些目标需要独立验证的要求。

VeroTrace可以方便地确定一项需求的整个状态。每项需求和工件都有五个评审状态（Not Ready，Ready for Review，Failed，Passed和Signed）。通过查询功能，可以快速确定：

1） 需求相关的所有工件是否已经导入？

2） 需求和工件的评审状态是什么？

3） 需求处于整个生命周期的什么阶段？

这些结果将有力地支持项目进度管理和变更的影响性分析。

值得一提的是，VeroTrace可以自动生成CD-ROM格式的可浏览的认证证据。超链接的生命周期可追踪性数据有极大的优势。它不仅消除了枯燥的人工生成和交叉索引数据的工作，还向DER（Designated Engineering Representative，指定工程代表）和客户提供了清楚简洁，易于审阅的工作界面，显著地提高软件认证的效率和软件产品的置信度。

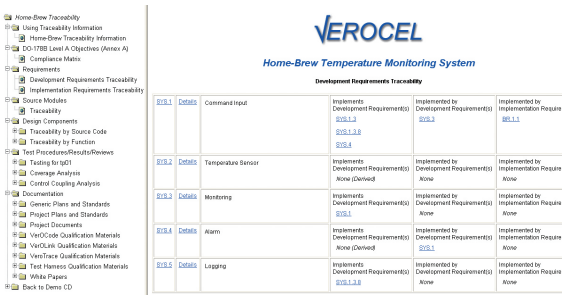


图5：VeroTrace CD-ROM格式可浏览的认证证据[4]

3、Reqtify

Reqtify由GEENSOFT公司开发，后被法国达索公司收购。Reqtify用于追踪和影响性分析，是专门针对基于文件的高度可定制化、易用和易集成的需求管理工具。Reqtify支持工具鉴定。Reqtify在航空领域非常知名，其用户包括NASA、AIRBUS、BOEING、GE、Rockwell Collins、THALES、SAFRAN、LIEBHERR等。

Reqtify并不存储需求，它可以用于统计需求管理信息、维护需求跟踪关系和定制报告。虽然需求和工件信息存在于其他工具和资料中，但是Reqtify强大的兼容性仍然可以帮助用户出色地完成工作。Reqtify几乎和主流的商用工具都有现成的接口：

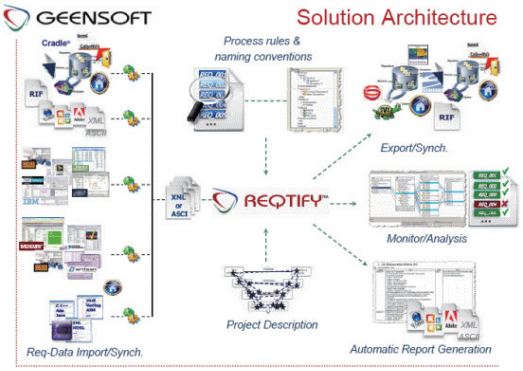


图6 Reqtify的解决方案架构[5]

1） Office工具：Word、Excel、PDF、Text；

2） UML工具软件：Rose、Objecteering、Rhapsody；

3） 建模仿真工具：Simulink、Statemate、Scade；

4） 代码语言文件：C、C++、Ada、SDL、VHDL、Verilog、Matlab(.m)文件；

5） 配置管理软件：Clearcase,、CVS,、PVCS；

6） 硬件设计工具：VisualElite、VNCover；

7） 需求管理工具：DOOORS、 RequisitePro。

对于一些其他类型的文件，Reqtify都可以通过API来捕获需求、属性、覆盖信息以及链接。

在需求追踪方面，Reqtify和VeroTrace最大的不同在于Reqtify用可视化替代了超链接，通过需求和工件条目之间的连线，可以清楚地查找上下游的追踪关系。

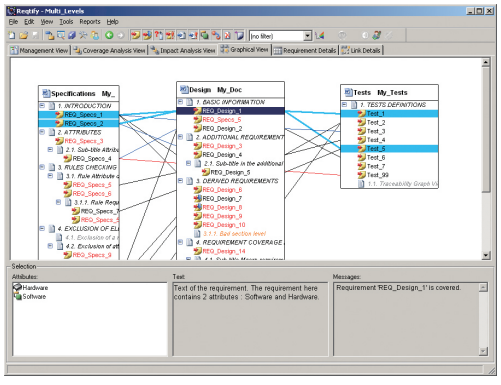


图7：Reqtify的追踪关系图形化视图[5]

三、对软件开发和验证的影响

需求追踪工具化对软件开发过程的影响从软件需求过程开始。高级需求直接通过对系统需求和系统结构的分析产生。需求通过整理就可以直接写入DOORS或VeroTrace的开发需求（Development Requirement），以需求条目的形式管理并建立追踪关系。对于不能追踪的派生需求要及时指明其类型。在需求开发的同时就可以开发测试用例。根据需求可以确定一系列的测试输入、执行状态以及期望结果，同时建立与需求的追踪关系。在需求评审中，评审人员会仔细分析需求的内容和追踪关系，同时检查测试用例，确定需求的可验证性。

需求通过评审后就可以进入软件设计过程了。在设计过程中，软件结构被定义。高级需求在软件设计过程中进一步分解，产生一级或多级有继承关系的较低级的需求，并补充派生需求。低级需求是不用更多信息，可以直接实现源代码的软件需求。低级需求同样写入DOORS或VeroTrace中的实现需求（Implementation Requirement）。值得关注的是，近年来软件设计越来越多地采用MBD（Model-Based Development，基于模型的开发）的方法来定义软件结构，设计逻辑算法以及自动生成代码，部分取代了文本化的低级需求。针对这种趋势，例如Esterel SCADE Suite提供了RM Gateway，可以在DOORS中生成与SCADE模型对应的需求模块。通过该模块就可以在DOORS中与高级需求模块建立追踪关系。由于SCADE模型文件是XML格式，VeroTrace提供的XML提取工具也可以捕获SCADE模型信息形成实现需求。

每一个软件开发过程都可能产生派生需求。派生需求是不能直接追溯到更高级需求的需求。一个派生需求的例子是需要为特定目标机开发的中断处理程序。高级需求可包括派生需求，低级需求也可包括派生需求。派生需求对安全性需求的影响由系统安全性评估过程确定，评估结果将写入DOORS或VeroTrace中派生需求的安全性考虑的属性中，便于后期的评审和验证。

进入软件编码过程。手工编写的代码和未经过工具鉴定的代码生成器生成的代码仍然需要追溯到低级需求，并在代码评审中进行人工检查。在DOORS中，将代码导入形成源代码模块，然后与低级需求模块建立追溯性。VeroTrace可以分析源代码中函数名称，形成函数列表。Reqtify则利用强大的正则表达式匹配功能，捕获并维护源代码和低级需求的追踪关系。对于经过工具鉴定的代码生成器，例如SCADE KCG，其自动生成的代码是不需要代码评审的，模型与代码的追踪关系由经过工具鉴定的KCG代码生成器保证。

软件验证过程的目的是检测和报告在软件开发过程中可能已形成的错误。软件验证方法主要包括测试、评审和分析。需求追踪工具化对软件验证的影响是不言而喻，在工具的帮助下建立起需求和工件的追踪关系。在阶段评审和变更影响分析之前，工具给出的一份追踪分析报告，可以有效的帮助评审人员确认是否实现了软件验证过程的总目标：

1） 分配给软件的系统需求已发展成满足这些系统需求的软件高级需求；

2） 高级需求已发展成满足这些高级需求的软件体系结构和低级需求；

3） 软件体系结构和低级需求已发展成满足低级需求和软件体系结构的源代码。

四、对质量保证和合格审定的影响

机载软件的质量由公司的SQA（Software Quality Assurance，软件质量保证人员）和适航审定中心授权的DER共同保证。他们要保证：

1） 软件计划和批准满足合格审定基础和DO-178B目标；

2） 软件开发过程和综合过程符合批准的软件计划和标准；

3） 软件开发过程和综合过程满足软件生命周期过程的转换规则；

4） 进行了软件产品的符合性评审。

SQA独立于软件开发和验证团队，参与和见证部分开发和验证活动，抽查软件产品和过程资料。对发现的不符合项，SQA会形成记录并提交软件开发和验证团队纠正。SQA通过需求追踪工具提供的追踪关系可以客观地了解已开发生命周期数据的状态和项目进展情况，并及时向项目经理及高层经理做出真实可靠的汇报。

（下转102页）

基于CANTATA++的软件单元测试

文 | 上海航天控制技术研究所 谭晓宇

背景

随着软件系统的日趋复杂，可靠性要求的不断提高，在软件开发各阶段进行相应的软件测试显得非常重要。其中，单元测试检查了每个软件单元是否正确的实现软件需求、功能、接口和其他设计约束。通过单元测试可以及早发现软件代码中绝大多数的编码错误和逻辑错误，可以为进一步的软件研制清扫“隐患”。实际工程中，仅仅依靠软件测试人员的人工统计分析已不能有效满足软件单元测试的有效性要求，因而软件单元测试的工具选择就显得尤为重要。Cantata同时具有静态测试和动态测试功能，同时操作简便，功能完善，界面友好，较好的满足了软件单元测试的各项要求。

1 软件单元测试

1.1 软件单元测试的定义

软件单元测试是对软件基本组成单元进行的测试。单元测试要求：（1）严格按照软件需求文档，对软件单元的功能、接口进行测试；（2）测试用例的输入应至少包含有效数值、无效数值和边界数值；（3）语句、分支覆盖率必须达到100%，对于无法达到100%的情况必须进行详细说明，并采取其他测试手段补充测试。软件单元测试包含静态测试、白盒测试、黑盒测试等。实际工程中一般以静态测试和白盒测试为主、黑盒测试作为补充。

1.2 软件单元测试过程

单元测试主要包括以下几项工作：制定单元测试计划、建立单元测试环境（被测代码、测试驱动模块、桩模块、测试报告，四者相关联）、进行单元测试用例设计、执行单元测试用例、单元测试结果判定、分析测试是否通过（对于不通过的情况，对源代码或测试用例进行必要的修改后回归测试）、完成单元测试报告。其中，Cantata可以为用户建立初步的单元测试环境包括自动生成测试脚本，自动“打桩”等功能，并自动完成单元测试结果的判定，最终生成测试报告。通过CaatppReporter软件可以自动生成.doc格式单元测试报告。用户可以集中精力完成单元测试用例设计和单元测试结果分析，从而提高了测试的全面性和有效性。

2 CANTATA++软件单元测试

2.1 测试系统架构

CANTATA++在授权的情况下，允许使用服务器-客户端形式架构测试平台：即在服务器上使用一个软件狗，客户端通过访问网络浮动license访问服务器通过许可认证，进行软件测试。这种方式使单元测试可在任意地点进行，不再局限于某一软件测试实验室。这种方式在实际工程运用过程中起到了较好的作用。

2.2 静态分析

Cantata静态分析功能提供了超过300个针对函数，类和文件范围的代码质量及复杂度度量。实际工程上主要关心的度量有：代码行总数、注释行总数、源代码行总数、参数数量、函数的圈复杂度、最大语句嵌套深度、函数中所有语句的语句嵌套层次总和等。Cantata除了可以提供以上分析数据外，还可以做一些标准代码质量度量和面向专家对象度量。

如某段被测程序单元：

```
//--取符号。
long Sign_long(long Val) //Integer
{
   long Result;
      if (Val==0)
         Result = 0 ;
      else
           {if (Val<0)
         Result = -1 ;
         else
         Result = 1 ;
        }
      return Result ;
}
```

Cantata的静态分析可得到168项分析结果，这里截取6项我们主要关心的分析项，结果如表1所示下：

表1：静态分析结果

分析项目	结果
"LINE_SOURCE"代码行总数	14
"LINE_COMMENT"注释行总数	2
"LINE_CODE"源代码行总数	13
"NESTING_MAX"最大语句嵌套深度	2
"NESTING_SUM"函数中所有语句的语句嵌套层次总和	6
"HALSTEAD_PARAMS"参数数量	1
"MCCABE"函数的圈复杂度	3

通过Cantata的静态分析，可以检测代码质量和复杂性度量，从而帮助用户确定哪些代码会更容易受到代码缺陷的影响，此外也可评估出测试所需时间。这些度量都被收集和储存到逗号分隔符（CSV）文件中，便于用户查看。

2.3 测试用例设计

单元测试中，最核心的就是测试用例的设计，设计测试用例主要有“逻辑覆盖法”和“路径覆盖法”。在实际工程中，结合这两种方法可以达到较好的测试效果。首次设计测试用例时，即可使用“路径覆盖法”，按照预先已有的程序流程图设计测试用例。执行完一次测试后，根据测试结构，判断逻辑是否全覆盖。若未全覆盖，再按照“逻辑覆盖法”补充设计测试用例。

在进行针对2.2节中的被测单元，我们设计以下测试用例，其中用例2故意设计出错：

用例1：输出值= 4；期望输出值=1；

用例2：输出值= -4；期望输出值= 0；

通过建立测试项目工程、导入测试程序后，我们通过Cantata“测试脚本管理器”选择检查语句覆盖率100%，分支覆盖率100%。Cantata自动建立测试环境，并可以自动生成一个初始的测试脚本如下：

```
/* run_tests() contains calls to the individual test cases,
you can turn test*/
/* cases off by adding comments*/
void run_tests()
{
    test_Sign_long(1);
    rule_set("*", "*");
EXPORT_COVERAGE("test_Sign_long.cov", cppca_export_
replace);
}
void test_Sign_long(int doIt){   //32bit integer
if (doIt) {
    /* Test case data declarations */
    long Val = NOT_SET;
    long expected_returnValue = NOT_SET;
    long returnValue;
    /* Set global data */
    initialise_global_data();
    /* Set expected values for global data checks */
    initialise_expected_global_data();
    START_TEST("test_Sign_long",
               "<Insert test case description here>");

        /* Expected Call Sequence  */
        EXPECTED_CALLS("");
            /* Call SUT */
            returnValue = Sign_long(Val);
            /* Test case checks */
            CHECK_S_INT(returnValue, expected_returnValue);
            /* Checks on global data */
            check_global_data();
        END_CALLS();
    END_TEST();
}}
```

此脚本中，void test_Sign_long(int doIt)为测试用例，在此测试用例中，我们重点关注的是以下部分：测试输入值Val、期望输出值expected_returnValue，实际输出值returnValue。CHECK_S_

INT(returnValue, expected_returnValue)为检测函数，用于检测实际输出值和期望输出值是否一致。

通过修改这个脚本中的输入值和期望输出值，我们可以完成对这个被测单元的测试用例设计。

我们修改后的单元测试用例关键变量如下：

用例1：long Val = 4; long expected_returnValue = 1;

用例2：long Val = -4; long expected_returnValue = 0;

2.4 测试结果分析

通过Cantata执行测试用例后，可得到图形化的测试结果。测试结果中，我们可以看到Cantata为我们自动检测期望输出与实际输出是否一致，打勾的部分为一致，红色打叉的部分为不一致。如图1所示

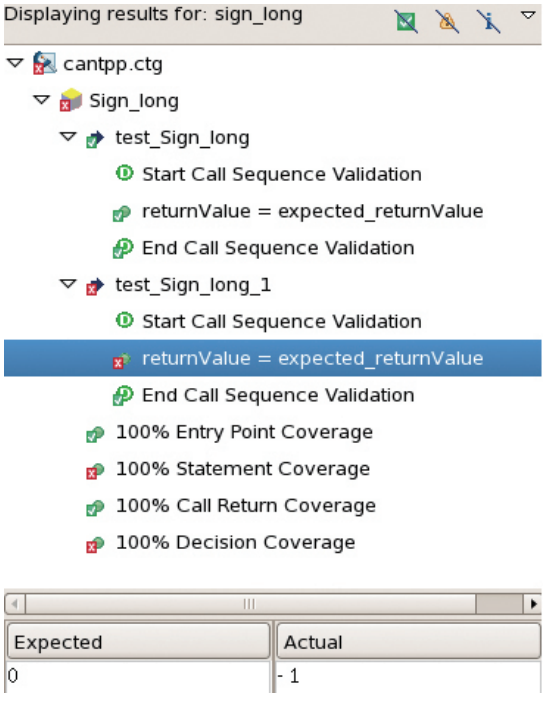


图1 测试结果图

从图中可以看出，测试用例2（test_Sign_long_1）的期望输出与实际输出不符（期望输出为0，实际输出为-1），同时语句覆盖率和分支覆盖率未达到100%。Cantata提供了覆盖率细节图像化显示，如图2所示：

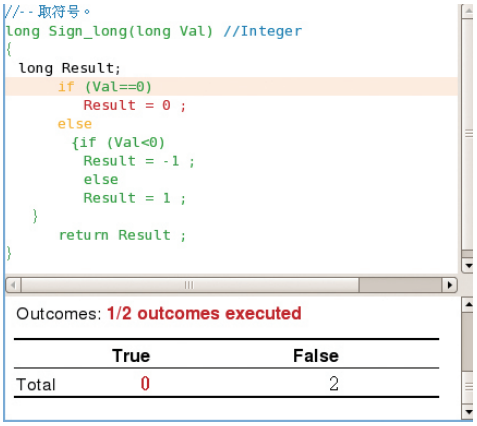


图2 覆盖率细节图

覆盖到的代码为绿色，未覆盖的代码为红色，未覆盖的逻辑为黄色。通过察看逻辑覆盖细节，我们可以看到语句未全覆盖的原因是语句“Result=0;”未执行；逻辑未全覆盖的原因是语句“if(Val==0)”执行“TURE”0次。

通过CANTATA++的图形化显示，我们可以很快的发现问题，并对测试用例作出修改。修改测试用例2，并新加测试用例3 ：

用例2：long Val = -4; long expected_returnValue = -1;

用例3：long Val = 0; long expected_returnValue = 0;

执行新的测试用例后得到最终测试结果图如图3所示：从图上可以看出，测试全部通过了。

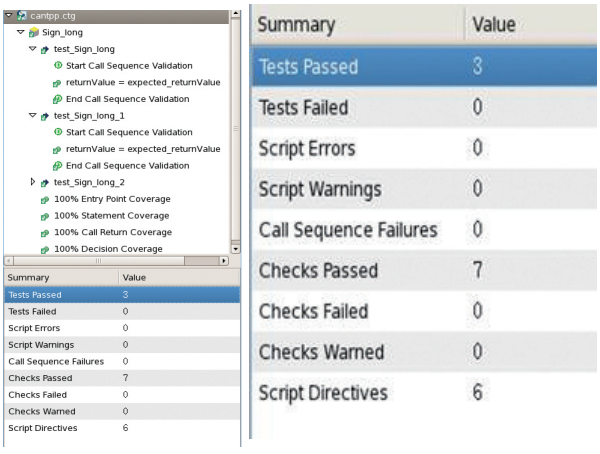


图3：最终测试结果图

2.5 测试报告生成

Cantata在测试完成后，自动生成一个.ctr格式的单元测试报告。对于实际工程中，我们更希望得到的是

一个.doc格式的测试报告。使用上海旋极公司开发的一个小软件CantppReporter，即可实现上述要求。该软件将Cantata自动生成的.ctr格式的测试报告和测试脚本向联结，自动生成了.doc格式的单元测试报告。2.2中的软件单元测试报告如所示。

表2：软件单元测试报告

测试模块名称	sign_long.c			

测试输入	long Val = 4;//FFFFH=4294967295
	long expected_returnValue = 1;
	long returnValue;			

桩模块	No stub			
期望结果	returnValue= expected_returnValue			
测试结果				
实测结果	returnValue=1			
通过情况	PASSED			

测试输入	long Val = -4;
	long expected_returnValue = -1;
	long returnValue;			

桩模块	No stub			
期望结果	returnValue= expected_returnValue			
测试结果				
实测结果	returnValue=-1			
通过情况	PASSED			
测试用例标识	3			

测试输入	long Val = 0;
	long expected_returnValue = 0;
	long returnValue;			

桩模块	No stub			
期望结果	returnValue= expected_returnValue			
测试结果				
实测结果	returnValue=0			
通过情况	PASSED			
总覆盖率	语句	1.0f+2%	分支	1.0f+2%
测试用例执行人			执行时间	

3 小结

单元测试是软件研制开发必不可少的一项工作，但因为单元测试的繁琐、耗时等特点，大多数开发人员在进行单元测试时往往面临着在提高产品质量与缩短测试时间之间进行选择的问题。IPL提供的产品Cantata正是应实际需要，在合理的时间耗费下提供给软件开发者的一种强有效的软件测试工具。Cantata可以支持对C/C++语言的测试，具有一整套包含功能测试、覆盖率分析和静态分析的功能。Cantata可以自动建立测试环境、自动生成测试脚本、自动生成测试报告、对测试结果进行图像化显示，从而能够满足开发者进行高效的单元的需求。该产品能帮助提高生产率，帮助软件开发人员更有效地完成软件的单元测试、集成测试和代码覆盖率分析。

（上接98页）

比起SQA的公司内审计，DER的审查会更加正式和严格。DER的审查过程依据FAA Order 8110.49，软件批准指南，主要以软件评审的方式分阶段介入机载软件的研制过程。软件评审分为四种类型：

1） 软件计划评审

2） 软件开发评审

3） 软件验证评审

4） 最终认证软件评审

DER最常用的审查手段就是抽取一次软件变更，从变更影响性分析，实施变更，重新验证到批准变更，从头到尾地彻底检查文档代码的修改以及中间活动产生的记录，确认变更的操作流程符合批准的软件计划和标准，变更后的产品符合需求并且不会给系统或飞机带来安全性的危害。需求追踪工具生成的追踪关系报表非常有助于DER这种“顺藤摸瓜”似的审查方式。尤其是VeroTrace，其生成的CD-ROM格式的可浏览的认证证据在最终认证软件评审中是非常受DER和客户欢迎。由于CD-ROM中包含完整的生命周期资料和清晰的追踪关系，DER可以快捷的定位数据和查看追踪关系，避免了不必要的查找和解释，节省DER软件评审的时间，同时也降低了软件认证的风险。友好完整快捷的展示界面为VeroTrace赢得了在航空业界的良好声誉。

参考文献

[1] RTCA DO-178B，机载系统和设备合格审定中的软件考虑

[2] FAA Order 8110.49，软件批准指南

[3] DOORS 8.1使用手册

[4] Verocel Life Cycle Traceability Demonstration — Beer Brewing System

[5] Reqtify Training Course

McCabe IQ在白盒测试中的应用体会

文 | 西南电子电信技术研究所 陈高树

曾几何时，数据处理类型的软件规模不过万行，测试人员尚能使用手工方式完成软件代码的测试和分析，尽管耗时耗力；然而，随着信息技术的高速发展，大规模系统平台软件的逐渐出现，测试人员要想使用纯手工方式深入了解软件代码的内部结构从而进行软件的充分测试已不可能：一方面，程序结构会随着代码规模的增加而成“指数”增加，；另一方面，项目开发人员的增加会加大软件代码的缺陷率和复杂度。

试想，当一名测试人员看着眼前数万行甚至十几万行的软件代码，而手中无任何白盒测试工具时，他会作何感想，是斗志昂扬还是望而却步呢？答案不言而喻。

所幸聪明的IT人员开发出了各式各样的白盒测试工具供测试人员使用，而旋极信息所推荐的McCabe IQ便是这其中的佼佼者。笔者曾使用McCabe IQ在多个项目中进行结构化测试，把其中的测试体会与读者共享。

1 作为测评项目负责人的使用

情景一：某个软件在现场运行过程中存在不规律的宕机情况，用户要求查明真相。此时，上级领导总会询问，测评中心，此事你们怎么看？

当测评中心的测评项目负责人面对如此庞大而又已经成型的软件代码，首要问题便是了解代码的组织结构，把握整个代码的全局。然而，常常由于开发项目组生产力的关系，要想从开发人员提供的设计文档中了解整个代码的组织关系是很有难度的，于是，熟练的测评项目负责人便会使用集成到VS IDE中的McCabe-Battlemap迅速地生成各模块之间的可视化关系图（如图1所示），从而了解整个软件的函数调用关系。利用可视化的结构图，测评项目负责人可根据其耦合关系将软件代码分割为多个部件，辅以工具提供的其他分析组件（如：通过菜单选择Metrics-Basic Reports-Module Metrics生成各模块的度量报告；通过菜单选择Metrics-Graphical-Scatterplot生成散点图；通过菜单选择 Metrics-Graphical-Kiviat生成Kiviat报告等），即可合理的分配相应的部件给测评人员，使测评项目组能够快速协调地开展软件的结构化测试，找出软件的出错原因。

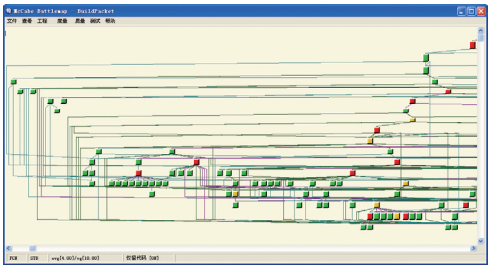


图1 可视化的代码组织结构图

除此之外，还可以从图中了解各个代码模块的复杂度，预估错误发生的可能性，从而在进行全面进攻的同时，组织相关测评人员重点进攻缺陷风险高发的模块。

情景二：在某个软件的测评结果评审过程中，评委总会询问测评项目负责人如何确定该测评项目已经充分测试，或者会询问测评项目组如何保证测试质量。

当多数测评项目负责人面对如此简单而又难以回答的问题时，如果没有准确的数据展示，很难让委托方心服口服。实际情况是，我们的测评项目负责人从容地打开McCabe IQ工具，从以下两方面的展示回答评委：

（1）可视化的覆盖率图形：通过选择菜单“Testing-Testing Data-Import”导入测试用例执行后的输出数据，在覆盖率模式下，向评委展示可视化的覆盖率图（如图2所示）。其中Battlemap颜色反映了在指定覆盖率模式下的执行测试的类型和测试程度，这要比空泛的结构化和复杂程度的报告好很多，直接给评委

以简单明了的覆盖说明。

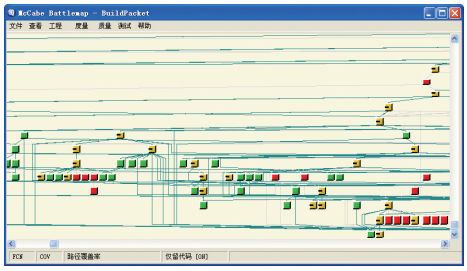
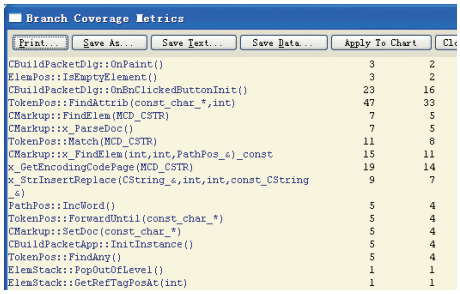


图2 可视化的覆盖率

（2）覆盖率报告：当评委看了可视化的覆盖率图形后，可继续展示各种覆盖率报告，通过真实有效的精准数据加以佐证，使回答有理有据、客观公正。比如通过选择菜单“测试-报告-分支覆盖..”（如图3所示）。当然McCabe IQ除了分支覆盖以外，还有诸如代码覆盖、路径覆盖等等。



Branch Coverage Metrics

Print... | Save As... | Save Text... | Save Data... | Apply To Chart

CBuildPacketDlg::OnPaint()	3	2
ElemPos::IsEmptyElement()	3	2
CBuildPacketDlg::OnBnClickedButtonInit()	23	16
TokenPos::FindAttrib(const_char_*,int)	47	33
CMarkup::FindElem(MCD_CSTR)	7	5
CMarkup::x_ParseDoc()	7	5
TokenPos::Match(MCD_CSTR)	11	8
CMarkup::x_FindElem(int,int,PathPos_&)_const	15	11
x_GetEncodingCodePage(MCD_CSTR)	19	14
x_StrInsertReplace(CString_&,int,int,const_CString_&)	9	7
PathPos::IncWord()	5	4
TokenPos::ForwardUntil(const_char_*)	5	4
CMarkup::SetDoc(const_char_*)	5	4
CBuildPacketApp::InitInstance()	5	4
TokenPos::FindAny()	5	4
ElemStack::PopOutOfLevel()	1	1
ElemStack::GetRefTagPosAt(int)	1	1

图3 分支覆盖率报告

2 作为测评人员的使用

前面讲了项目负责人的使用，我们再看看测试人员的使用：

测试准备：一般而言，我们已习惯了微软的VSIDE环境，不再想用命令行方式操作，还好McCabe提供了IDE插件，通过“idesetup MS.NET -ver x.0”将本工具集成到对应X版本的VS IDE环境中，帮助测评人员在项目中使用起来得心应手，在VS IDE环境中通过菜单“工具-McCabe-Target Configuration”和“工具-McCabe-Scope Configuration”依次完成目标配置（包括插桩配置）和分析范围的配置。

测试计划：通过测试准备，即可整装待发了。在Testing 菜单中选择Test Plan -Unit Level，打开各模块的测试路径图（如图），在右边McCabe IQ已经为我们测评人员做好了完美的测试计划。你只需设计出对应的测试用例完成对应的计划就可以完成此次测评任务，非常完美。

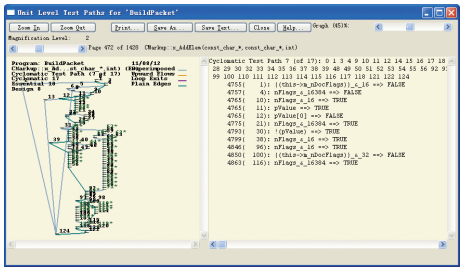


图4 测试路径图

测试实施：很好，计划已经完成，现在进入漫长的测试实施之旅了。面对伟大的测试计划，测评人员很容易设计出对应的测试用例，然后从工程目录\projectname_inst\插入插装后的代码，运行产生包含插装信息的可执行文件，执行所设计的测试用例，当执行完测试用例后通过菜单“Testing-Testing Data-Import”导入测试执行后的输出数据*.out，然后在覆盖率模式下的Battlemap上，选择被测模块，右键打开其弹出菜单选择“未测试边的流图/列表…”（如图5所示），定位测试过的代码，直观地检验测试覆盖情况，并由此帮助测评人员进行测试用例的再设计和重构，用最少的测试用例完成最大的测试覆盖，如此进行测试迭代，保证测试的充分性以及测试的效率性。

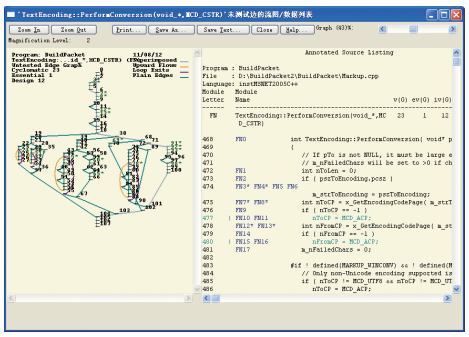


图5 测试路径覆盖图

3 总结

在多次使用McCabe IQ 进行软件代码的白盒测试后，我们的测评人员面对纷乱如麻的代码时也能从容应对，利用其各项功能将软件代码逐一解剖，各个击破，使白盒测试不再深不可测，也不再使测评人员望而生畏。面对如此便利的工具，你是否有跃跃欲试的冲动呢？

AdaTest95在型号控制软件中的应用

文 | 上海航天控制技术研究所 李芳华

在型号控制软件的单元与组装测试中使用AdaTest95测试工具对软件进行了单元和组装测试。为了方便使用该测试工具和提高测试效率，对AdaTest95使用过程中需要注意的一些问题和事项进行了总结。

1. 测试准备阶段

正常配置AdaTest95 License，并键入命令ipgada95 --projdir:/工作目录；注意此工作目录应该是之前已经建立好的目录；建议每个测试单元应建立独立的工作目录；

2. 测试脚本生成阶段

完成以上注意事项后根据测试脚本向导生成测试脚本文件，见图1：

图1

点击测试脚本向导进入测试用例定义，在设置测试用例时向导中必须使用的界面见图2。

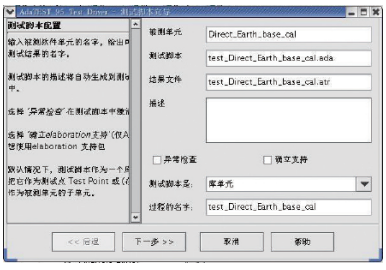


图2

只要在图2中键入被测单元；其他是自动生成的。完成操作后点击下一步，进入下一页面的测试。见图3。

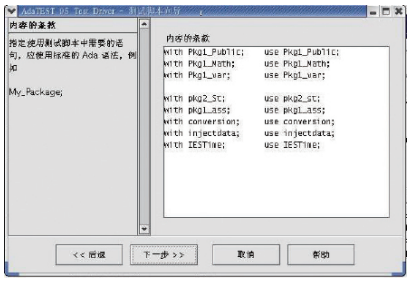


图3

图3中需要将被测包中使用到的所有变量和子程序所在包名输入，有遗漏可以在编译时修改*.ada文件补充。然后进入下一步。见图4。

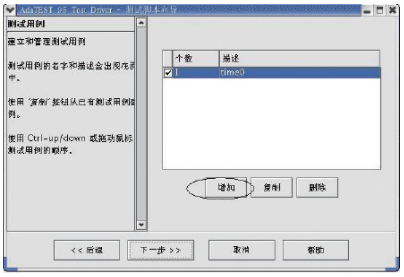


图4

点击增加，进入测试用例生成器，见图5：

图5

点击编辑测试数据进入下图，对测试脚本中需要关注的变量进行定义，如有遗漏则在测试脚本的编译中有错误提示，对测试脚本进行相应修改调试，见图6。

图6

选择确定退出，点击下一步，完成了测试脚本的生成。然后设置分析选项，点击分析向导定义过的需求，进入分析向导，此操作主要是设置测试需要分析统计的覆盖率类型。在姿控应用软件中需要关注的测试覆盖率为语句覆盖率和判定覆盖率，见图7：

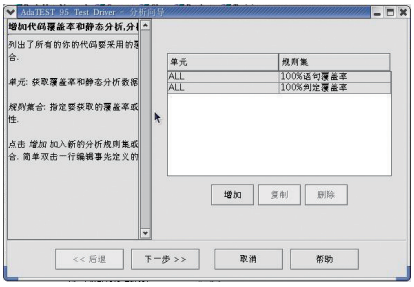


图7

点击下一步，设置要插装的文件，见图8：

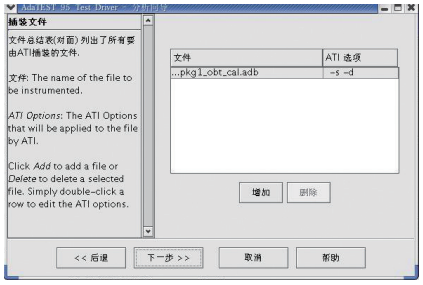


图8

之后在向导中点击保存并产生，产生测试脚本（含对分析文件的插装）。

3. 测试用例调试及分析阶段

在正常生成*.ada文件以后通过编译运行生成测试结果。该文件语法规则符合Ada语言编程规则，所以整个编译运行过程同所有Ada语言编写的程序处理一致。同时也可以在测试脚本生成过程中发现数据错误后直接修改*.ada文件进行更改后再编译，使用非常灵活，提高了测试效率。

有可能因为在使用测试脚本向导生成测试脚本过程中输入错误或不完整等原因需要对测试用例进行调试修改。具体修改内容如下：

1） 必须注释掉*.ada文件中测试工具自动插入的对被测单元的with;加入对被测单元所在程序包的with（在Ada程序中with关键字后只能跟程序包名而不能跟程序单元名）；

2） 需要注释掉*.ada文件中测试工具自动插入的declare语句；

3） AdaTest95预设的9种类型检查如下：Integer/Boolean/Character/Wide_Character/String/Wide_String/Float/Duration/Address

如果需要检查的变量类型不在AdaTest95预设的9类内，需要派生一个新的类型检查程序来检查该变量。例如要对unsigned_16类型的变量进行检查，需要派生出一个新的检查程序：

```
Procedure CHECK is new GENERIC_MODULAR_CHECK (Any_Modular=>unsigned_16);
```

4） 需要特别注意全局变量不需要在测试用例中定义；

5） 为了提高测试效率，每个被测单元只需要通过向导生成第一个测试用例，其它需要增加的测试用例可以通过直接修改代码来实现；

6） 如果被测单元有错，更改后回归测试只需要键入命令“ati -s -d 包名”；然后再根据需要修改*.ada文件来修改测试用例。

4. 总结

AdaTEST 95完全解决型号控制软件Ada代码单元测试中的所有问题，它包括动态测试、覆盖率分析和静态分析等功能，特别是ADA测试脚本，帮助我们灵活和顺利的解决了在测试中遇到的问题，完成了单元测试任务。

软件工程技术专家

“从事软件工程工作，需要无限学习的精神和持续开放的视角。”

任建国 旋极信息军工中心软件测试技术经理

2000年至今就职于北京旋极信息技术股份有限公司，现任职软件测试技术经理，负责软件测试的开发、维护及技术支持工作。在技术上较深地掌握了各家公司的各类软件测试工具，深入研究学习国外多家软件测试工具生产厂商的多种产品，通过了IBM全球Rational Quality Management专家技术认证考试，拥有12年软件工程化工作的丰富经验。

曾经参加过的测试与培训项目有：参与旋极信息和空间中心联合举办的“探月项目软件测试培训”；参与航天某研究所配置管理项目培训和系统搭建；参与中电某研究所GJB5000A管理平台搭建与培训；参与神州飞船某测试项目；参与兵总某机车测试项目；参与航天某系统测试项目；参与航空某适航测试项目；受公司委派参加了英国PRQA公司在伦敦举办的软件规则高级认证培训，并得到证书；通过了IBM全球Rational Quality Management专家技术认证考试。

“不谋全局者，不足以谋一域。在软件工程领域，无论从事多微小具体的工作，都需要有软件工程化的大思路、大想法。”

王鹏 旋极信息军工中心技术支持高级工程师

1993年毕业于哈尔滨船舶工程学院自动控制系，2003年加入北京旋极信息技术股份有限公司，现任军工中心测控部高级工程师，从事软件工程产品的技术支持工作，业务能力覆盖软件工程各个领域。曾经参与客户的多个软件测试项目；译有MISRA C/C++的编程标准；帮助客户为通过GJB 5000A二级而搭建配置管理平台；对国际流行的建模方法有所研究，经验丰富。

为人处事热情亲切，专业知识丰富，期望为我国军用软件的成长做出贡献！

闵蓓尔 上海旋极技术部软件工程化产品经理

2000年进入上海旋极，目前担任软件工程化产品经理，主要从事航空航天软件工程和软件测试产品的技术支持工作，曾经参与了华东地区多个研究所软件工程化和软件测试项目的咨询和技术支持工作，特别是对软件测试验证具有丰富的经验。对软件的DO-178B验证过程具有深刻的理解，并且已经在国内研究所成功开展了DO-178B软件验证的技术工作。

温文尔雅，耐心细致，象一缕春风一样为客户送去解决问题的办法。上海旋极，使命必达。

周恩军 上海旋极副总经理

山东大学计算机科学系本科毕业，华东理工大学研究生毕业，硕士学历，2004年进入上海旋极，目前担任上海旋极副总经理，主要负责销售团队和技术支持团队的建设和管理。曾参与多个航天研究所嵌入式开发和嵌入式测试项目的建设，在关键软件开发、软件工程和系统测试实施方面具有丰富的经验。

嵌入式开发出身，转入软件工程领域，专攻嵌入式软件测试验证，在技术追求的路上不曾停步。

皮永辉 旋极信息军工中心副总工程师

1989年毕业于哈尔滨工业大学，硕士学位。2000年加入深圳旋极，主要负责软件工程化方面的技术支持工作。

“我愿和团队一起，为西南地区军工软件工程建设、为成都旋极软件工程的发展，挥洒汗水、精心钻研，力争做出独特贡献！”

周珊 成都旋极软件工程部经理

2005年加入成都旋极，从软件测试技术支持入门，到软件工程部经理，在软件工程领域走过了7年的时间，参与了某所多个型号分系统的测试支持工作，参与并主导了某型号机型机载软件自动化辅助设计平台软件、飞参数据分析软件、新一代航空总线仿真设计平台软件等软件项目，得到了客户的高度认可与评价。

“我在多年的工作中，深刻体会到“软件工程化”难在“工程”两字，即如何采用工程的思想对软件进行构建和维护，如何在惯常的解决问题思维中提升对产品质量的态度，如何落实工程化的每一步。希望《旋极视界》软件工程专刊能为大家打开一些新的视角！”

张琨 西安旋极软件技术支持主管

2005年入职，2005年9月至2012年9月于北京旋极测控部软件组工作，目前任西安旋极软件技术支持主管。主要从事软件工程和软件测试产品的技术支持工作，曾经参与多个研究所的软件工程化项目和软件测试项目的咨询和技术支持工作，从2009年至今，受邀参与实施多个软件测试项目。

“软件工程化是把工程化的管理方法应用到软件的开发和维护中，即用系统化、规范化、数量化等工程原则和方法管理软件。”

魏欣 西安旋极软件技术支持工程师

2012年加入西安旋极，河南工业大学硕士学历，主攻方向为软件自动化测试。软件测试是软件工程的重要组成部分，软件测试作为验证软件需求，确认软件功能，确保软件质量的重要手段，其对于软件的开发活动有着不可替代的作用。由于软件规模的增大，功能更加复杂，开发周期更长，借助自动化测试工具对软件进行测试成为提高测试速度，确保测试质量的一个重要手段。

走过一年了，分享下我们在软件工程领域骄傲的事吧！

骄傲的事TOP10

旋极信息测试外包服务成果丰硕

2012年，旋极信息积极开展测试外包服务工作。先后承接烟台某单位软件配置项测试任务、空间中心某项目单元测试、航天三院某研究所Vdsp TS201系统单元测试。

筚路蓝缕，以启山林。旋极信息派员全程参与了多个项目的测试过程，既帮助客户完成了项目，也丰富了测试咨询服务的经验。定制测试策略、编写测试计划、设计测试用例、记录测试过程、总结测试报告，有力证明旋极信息具备完整的高水准的测试外包服务能力。

旋极信息出色完成VxWorks系统单元测试项目

2012年8月至9月，旋极信息参与某测评中心的VxWorks系统单元测试项目，此项目服务于航空某关键型号代码。其中复杂模块占总量的30%，涉及底层硬件操作的模块占总量的80%。旋极信息提供的单元测试工具Cantata++保证了项目的按时完成，通过Baseline测试用例生成过程，满足了双百覆盖率的要求。

Verocel CEO受上海旋极邀请赴沪培训

2012年，Verocel公司总裁George Romanski先生受上海旋极邀请，于7月在上海旋极与国内航空航天界相关人士就“机载软件适航认证过程与方法”主题进行了交流。

大型飞机重大专项是党中央、国务院建设创新型国家，提高我国自主创新能力和增强国家核心竞争力的重大战略决策，是《国家中长期科学与技术发展规划纲要(2006-2020)》确定的16个重大专项之一。

中国商飞公司是国家大型飞机重大科技专项的实施主体。中国商飞公司成立后，组建了一支来自全国47家单位，超过300人的大型客机联合工程队，举全国之力，聚全国之智，组织开展大型客机的技术经济可行性研究、总体技术方案论证和关键技术攻关，总体设计、系统规划、科学论证我国大型客机研制的总体蓝图。

上海旋极为C919大飞机提供适航认证服务

2012年上海旋极在软件产品的市场运用上取得显著成绩。最值得一提的是，携手美国Verocel公司为大型飞机重大专项C919项目提供符合DO-178B适航认证标准的机载软件开发支持工具，加快C919航电系统机载软件通过DO-178B认证，为适航认证提供帮助。适航认证是一架飞机生产、销售、运营的前提条件，是民航飞机进入国际市场的门槛。Verocel产品作为适航认证的辅助工具将作为中央计算资源平台应用于中国大飞机C919项目。

旋极信息第7次亮相中国国际国防电子展

2012年5月9日至11日，第八届中国国际国防电子展览会在北京展览馆盛大开幕！旋极信息作为国内领先的嵌入式系统整体解决方案服务商，携MARS、IceBlade、FireBlade、DT1553等多款拥有自主知识产权的产品亮相该展。

这是旋极第7次参加国防展。本届展会旋极公司共有航电测试、光纤网络、航空总线、适航认证、智能终端等数款丰富产品闪耀亮相。会上，旋极主推产品MARS远程光纤调试网络，并用MARS将IceBlade故障注入系统、FireBlade系统仿真测试平台、DT1553、AFDX测试仪及多媒体视频设备相互连通，展示了一个完整的航电设备调试测试网络，吸引了专业观众的热切关注。

成都旋极交付科研院所军用级应用软件

2012年，成都旋极完成并交付科研院所军用级应用软件——机载软件自动化集成平台。

机载软件自动化开发集成平台作为软件开发与测试环境的一部分，贯穿到整个软件开发测试全过程中，为软件开发测试提供必备的支持环境。该平台支持多用户开发，并自动产生相应的软件源代码，有效地减少开发人员的工作量，提高软件的开发效率和质量；另一方面为开发人员和维护保障人员提供新型的自动化测试组织与实现的工具。

机载软件自动化开发集成平台是成都软件工程部，整体研发力量的综合体现，该款软件也是软件工程部成立以来交付用户的第一款软件产品。整体的壮大离不开部分，成都旋极软件工程部这支生力军正在逐渐壮大，为旋极信息软件工程事业添砖加瓦。

成都旋极研发成功1394设计&仿真软件

1394B协议是在1394A协议的基础上，提升了速度和技术的新一代火线协议，其发展趋势迈向军事领域，将作为新一代军事领域实验室平台搭建的基础协议。

新一代1394B协议采用了高效的仲裁和先进的数据译码（8B/10B）机制，在减少信号失真同时带来更高的传输速率，较1394A速度至少提高1倍。

1394设计&仿真软件是2012年成都旋极为科研院所设计研发的基于1394B协议的网络仿真平台。此平台快速搭建基于1394B协议的实验室模型，利用XML文档存储，使用户方便快捷地读取多种实验室模型拓扑设计方案，轻松完成对网络环境和1394B数据包的设定。本软件的交付，标志着成都旋极已具备满足军用研发单位提出各种场景的应用级软件研发能力，是成都旋极综合实力提升的体现。

深圳旋极深度服务中国赛宝实验室

2012年8月，深圳旋极向中国赛宝实验室提供嵌入式系统测试产品及技术服务，用于该实验室的测试平台构建。这意味着深圳旋极由过去多年致力于嵌入式系统测试产品的推广，向更广阔的嵌入式测试领域方面发展。

GJB 5369审查遭难题 西安旋极QAC深入报告获好评

随着GJB 5369-2005标准在西安很多航空研究单位的推广和落实，发现原有的标准检查平台出现了漏报和误报的问题，于是各使用单位对静态分析工具提出了标准审查分析的要求。

用户按照GJB 5369各项规范编写了一百多个检测代码，然后采用不同工具进行审查。对比相同代码采样的分析结果，原有平台在一些强制类标准上有着严重漏报的情况，而旋极产品QAC不存在类似的漏报，且能准确深入的报告被测代码中的隐藏问题。经过对GJB 5369检测代码的逐条分析，可以认为QAC对GJB 5369标准完全支持，能够完全替代原有平台。

我们向用户提供了针对检测代码的QAC分析报告，有效弥补了原有平台漏报的不足，得到客户的一致认可和好评。（报告可向西安旋极索取）

T-Plan销售赢青睐 西安旋极售前售后两手抓

测试管理类工具的推广，难在对市场的预热、对客户理念和需求的把握。测试过程管理的自动化明确项目目标、积累项目经验以及提升测试效率具有重要的意义，对于测试执行单位是不可或缺的。

西安旋极进行了长期不懈的宣传和客户试用工作，终于在2012年开花结果，T-Plan取得了较好的销售成绩。客户认识到在目前的发展阶段，改进测试过程管理能力是很有必要的。而且在项目中实施测试分析、测试设计和测试管理过程，可以使软件测试过程遵循统一的标准，达到测试过程的顺序结构可视化和文档生成的自动化。

这个项目中，我们针对用户需求定制了个性化的售后方案，力求能帮助用户真正提升测试过程管理水平。


国内领先的嵌入式系统整体解决方案服务商

旋极视界

WATERTEK VISION

与您见证旋极成长

内部资料　免费赠阅
京内资准字2012—L0112号

地址：北京市海淀区北四环中路229号海泰大厦1006室
电话：010-82883933　传真：010-82883858
网址：www.watertek.com