doi:10.19306/j.cnki.2095-8110.2020.05.020

导航计算机测试软件的架构设计

秦振汉,郭双红

(航天科工惯性技术有限公司,北京 100074)

摘　要:为解决多种型号被测产品、在不同测试设备上的测试软件开发的问题,尽可能地缩短开发和调试时间,并确保软件的开发质量,提出了一种针对导航计算机测试这一特定领域,以提高扩展性和重用性为目标的软件架构。该架构提供了统一的测试软件设计和开发方法,规范了软件的实现过程,为测试软件的开发提供了一致性的解决方案。采用组件化的开发方法,将具有复用价值的内容封装为组件,为测试软件开发和运行提供了共享的基础平台;提供了开放式的体系结构,规定了内部的组织结构、组件类别与职责、接口规范、交互方式、组件开发方法,便于软件的扩展和修改。经过工程项目的验证,该架构满足在各种专用测试设备上、针对不同型号被测产品的测试软件快速、高质量开发的需要。

关键词:架构设计;组件;测试;导航计算机

中图分类号:V446.2　　**文献标志码**:A　　**开放科学(资源服务)标识码(OSID)**:

文章编号:2095-8110(2020)05-0159-06

Architecture Design for Navigation Computer Testing Software

QIN Zhen-han, GUO Shuang-hong

(Aerospace Science & Industry Inertial Technology Co., Ltd., Beijing 100074, China)

Abstract: In order to solve the problem of test software development of various types of products under test and on different test equipments, to shorten the development and debugging time as much as possible, and to ensure the quality of software development, a software architecture that can improve the scalability and reusability of navigation computer testing is proposed. The software architecture provides a consistent solution for the development of test software via the uniformization of design and implementation and the standard of implementation process. By using the component-based development method, the content with reuse value is packaged as components, which provides a shared basic platform for the development and operation of software. Meanwhile, the method provides an open architecture and specifies the internal organization structure, component category and responsibility, interface specification, interaction mode, and the component development method, which is convenient for the expansion and modification of software. The architecture meets the requirements of rapid and high-quality development of testing software for different products on various special test equipments via verification of projects.

Key words: Architecture design; Component; Test; Navigation computer

收稿日期:2019-12-10;**修订日期**:2020-01-13

作者简介:秦振汉(1981-),男,硕士,主要从事测试设备、测试软件的研发。E-mail:qzh13651056530@126.com

0 引言

导航计算机测试软件是一种部署在不同类型的测试设备上，实现对不同型号导航计算机产品的测试，以确定被测产品是否为满足功能和性能指标的应用软件。在导航计算机测试软件的开发中，面临着以下问题：

1)导航计算机作为被测产品，虽然种类有限，但型号繁多。每一类导航计算机产品，虽然功能上相近，但在个体上都存在差异。反映在测试软件上，在测试方法、测试流程、数据处理、测试激励与响应等方面有很多都是相同或相似的，但是由于受到软件开发方法、开发人员和开发运行环境的限制，这些软件的重用性很低。每当有新的软件需求时，都要对现有程序进行修改和完善，这不仅浪费了大量的时间和精力，同时测试软件的质量也很难得到保证。

2)一种测试设备，通常解决一种或几种产品的测试，无法兼容全部型号，由此逐渐衍生出针对不同类型产品的多型测试设备。这些测试设备受制于开发时间、成本和需求等因素，在硬件构成上采用了很多的专用测试仪器，有些仪器或板卡是为满足测试要求而定制开发的。当测试设备变更、升级或板卡替换时，测试程序都会受到影响，甚至重新开发。

出现这些问题的原因主要是，在测试软件开发初期缺乏统一的设计，每个软件只是针对单台测试设备，满足对某种型号产品的测试需要。这些软件本质上都是独立开发和维护的，即使存在复用，也主要体现在代码层面和模块层面。随着测试设备和被测产品的种类、型号的增加，测试软件的数量越来越多，给后期的修改、完善和功能扩展带来了巨大的困难。

对于导航计算机测试软件的开发而言，一个清晰明确的软件架构是解决这些问题的关键，能够为测试软件的开发提供一个完整而统一的技术途径。软件架构最近才作为软件工程的一个独立研究领域出现，它对于软件开发的重要性和指导意义已经得到了广泛的共识，但目前对于软件架构这一概念还没有一个统一的定义。相关的研究人员、组织和机构分别从不同的角度，给出了不同的概念说明，在参考文献[1-3]中做了较为详细的阐述。抛开这些概念本身的差异，从测试软件开发的实用角度出发，软件架构是对整个软件系统的高层抽象，描述了该系统内部的组件以及组件间的相互作用，反映了系统设计决策的基本原理。

本文提出了一种基于组件的测试软件架构，其出发点不是某个单独的软件项目，而是针对导航计算机测试这一特定领域，以解决不同被测产品、在不同测试设备上的软件开发问题。通过软件复用方式，减少重复开发，提高了软件质量；同时，该架构提供了良好的可扩展性，能够适应后续不断增加和变化的软件需求，满足了研发和生产过程中测试软件快速开发的需要。

1 测试软件的基本设计思路

导航计算机测试软件采用了组件化开发方法。组件技术从根本上改变了软件的开发方式，可以通过对组件的组装，形成新的软件[4-5]，从而缩短了软件的开发周期，并提高了软件的开发质量。组件技术的核心是组件模型。目前比较常用的组件模型主要包括 CORBA、JavaBean 以及 COM/ActiveX、.Net 等。导航计算机测试软件一般运行于 Windows 操作系统中，综合考虑软硬件资源、开发习惯、后期维护等多方面因素，导航计算机测试软件采用了 .Net 组件模型。

大量经过验证的、功能多样的组件，是测试软件开发的基础。在导航计算机测试软件开发中，这些组件按性质可以大致划分为两类。

第一类为测试软件开发和运行所必须的基础组件，例如报表生成组件、接口定义组件、常用算法、界面控件、数据访问组件、领域模型组件等。该类组件反映的是软件开发中的共性内容，带有一般性，与具体的测试业务没有直接相关性，在测试软件中可以全局复用，也是软件开发和运行的必备要素。

第二类是软件开发中带有特定性质的个性组件。主要包括反映不同类型测试设备信息的系统交互组件；针对不同型号被测产品，实现不同功能需求的测试业务组件等。该类组件面向特定对象，具有特殊性。在软件开发和调试时，主要是针对这些元素进行修改、完善和补充。

这些组件主要来自对已有的知识和经验的归纳和总结。通过将具有复用价值的内容封装为独立的组件，建立了适用于导航计算机测试这一特定领域的组件库。当针对特定的软件需求进行应用软件开发时，可直接复用已有的基础，遵循公共的软件架构，将需要的组件装配在一起，以实现不同的软件功能。在这种开发方式下，不同项目的测试

软件开发问题，可以转换为依靠统一的软件架构、针对特定需求的组件开发与组件装配问题。

在导航计算机测试软件开发中，架构设计处于核心位置。它规定了软件的整体组织结构、内部组成元素及功能分工；提供了可纳入组件库的各种组件的接口规范；预先定义了测试软件的开发流程，指导软件开发活动。

2　测试软件的架构设计

软件架构是对系统的整体构想，详细描述了包含的架构图、组件、接口规范和交互机制等[6]。与独立的测试软件相比，导航计算机测试软件的架构设计不能只针对某个特定的软件项目，而应该覆盖导航计算机测试领域内的不同的软件需求；同时，能够提供灵活的扩展机制，适应多样化的需求变化。

2.1　分层体系结构

导航计算机测试软件架构采用了常用的分层式架构风格[7-9]，将体系结构划分为显示层、系统交互层、数据访问层和业务逻辑层等 4 层，如图 1 所示。

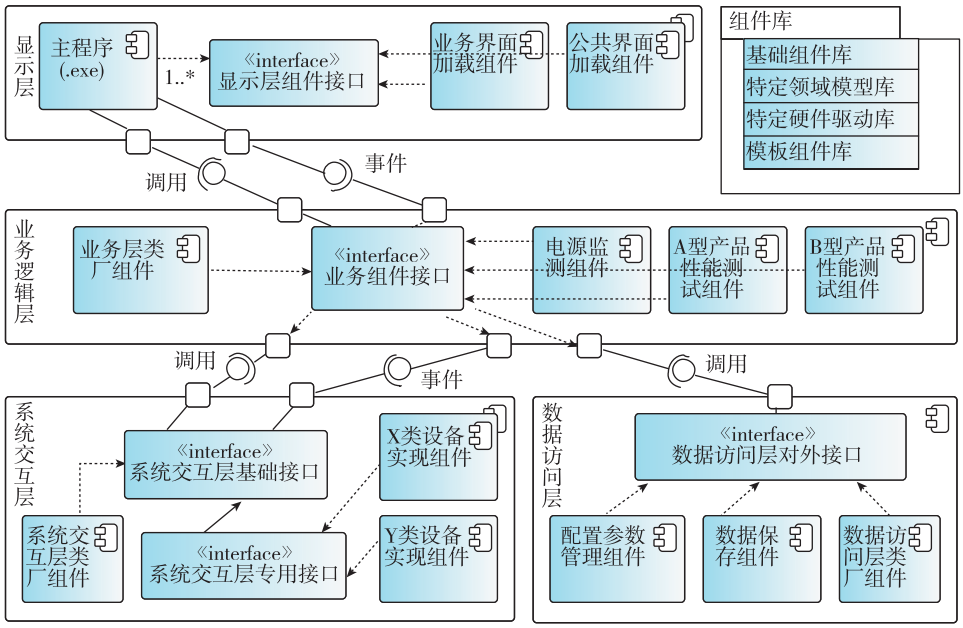

图 1　导航计算机测试软件的逻辑架构框图

Fig. 1　The logic architecture of navigation computer test software

1)显示层是应用程序和用户的交互界面。该层内包含了与具体业务无关的界面组件，并通过组合模式加载在主程序中。主程序以可执行程序形式存在，负责完成各组件的动态加载和组件间通信等，是所有组件赖以运行的基础。

2)数据访问层负责配置参数的访问、修改，测试数据、结果的保存，测试履历书的生成等。

3)系统交互层负责完成测试设备硬件(包括测试仪器、电气连接关系等)的封装，其主要功能是将软件的测试业务与具体的硬件设计分离，提高仪器互换性。导航计算机测试设备使用了很多非标仪器和板卡，无法使用诸如可编程仪器的标准命令(Standard Commands for Programmable Instruments，SCPI)、VXI 总线即插即用(VXI Plug&Play，VPP)规范、可互换虚拟仪器(Interchangeable Virtual Instruments，IVI)标准等仪器互换性技术和方法。仅以 IVI 驱动而言，目前只提供了 13 种仪器类驱动，且只适用于通用仪器[10-11]。但在导航计算机测试设备中，大多数仪器和板卡的驱动程序都不是按照这些标准开发的。为此，采用了基于功能分解的设计方法，忽略具体仪器或板卡的实现细节[12-14]，通过对测试设备所需实现功能的抽象，形成针对该设备的接口模型。

在导航计算机测试中，所需的功能项可细分为基本的仪器操作、产品切换、产品供电等共用类；以及各种通信数据收发、各种脉冲信号的输出控制、外部的恒流源控制、选通信号控制、振梁信号输出、DI/DO 操作、AD/DA 操作等专用类。每种测试设备需要实现的功能都是若干个功能项的组合，区别集中在功能项的种类和数量。为此，将带有共用性

质的功能项合并为基础接口，在定义后基本不会修改；将带有专用性质的功能项合并为专用接口，在软件设计时需要根据测试需求的不同进行修改和完善。例如，在图 1 中，X 类设备实现组件和 Y 类设备实现组件就是根据不同需求，单独开发的个性化组件。

4）业务逻辑层负责完成各种导航计算机的特定测试功能，包括常用的性能测试功能、在线升级功能、VF 标定功能和电源监测功能等。业务逻辑层内的组件只关注各自的业务流程与软件需求，例如，在图 1 中，业务逻辑层内部包含了电源监测组件，以及针对 A、B 两种型号被测产品的性能测试组件。这些业务组件直接反映了某种特定的测试需求，往往需要根据不同的业务需求进行定制开发。在业务层定义的组件接口中，包含了测试设备硬件控制接口、数据访问层控制接口、事件发送与接收接口等，开发人员只针对接口编程，不依赖具体实现，有效降低了组件间的耦合性。

5）组件库是各种类型组件的集合，其中既包括公共性组件，例如，定义接口规范、常用工具、算法的基础组件；管理电子履历模板、配置文件模板等的模板管理库等；也包括反映特定测试设备的硬件驱动库和反映特定业务领域的领域模型库。

2.2 交互方式设计

在该架构设计中采用了两种交互方式：1）显式调用方式，即上层组件通过接口，调用临近下层提供的服务。例如，业务层组件可以通过系统交互层的对外接口，实现对硬件设备的直接控制；2）采用事件机制的发布-订阅方式[4,15]。例如，系统交互层组件将采集到的电源数据封装为事件消息，通过事件发送接口，发布给业务逻辑层的事件订阅组件；后者通过事件接收接口获得信息，完成监测数据的处理。针对处于同一层次的组件，采用基于事件的转发方式，即先将交互信息发送到主程序，再由后者将信息转发到目标组件中。

通过接口调用和事件机制，降低、简化了各组件的对外依赖性，使软件可以在不修改、少修改的前提下，快速应对业务需求的变化。

2.3 组装方法设计

根据组装模式的不同，可以分为静态组装和动态组装。对于测试软件开发和运行所必须的共性组件，直接通过引用方式，由软件集成开发工具完成组装。但对于各种后续开发的、以独立的插件形式发布的个性化组件，必须使用动态组装方式。动态组装的组件信息统一保存在一组配置文件中，该文件采用 XML 形式，内部包括了各种组件的存在目录、命名空间、类名和加载顺序等基本信息。

动态组装功能由主程序发起，但具体的执行交由各层的类厂组件。类厂组件根据配置文件的内容，使用反射机制动态创建类型的实例。通过这种动态组装方式，使测试软件具有良好的扩展性，当需要增加或修改软件功能时，只需要在配置文件中修改或增加该组件的属性信息。

3 测试软件的开发

组件库提供了测试软件开发的基础材料，软件架构规定了测试软件的整体结构，明确了组件的装配方式和交互方式，但两者本身并不实现具体的测试功能。在面向某型号导航计算机的测试需求时，需要从已有组件库选择必须的组件，并开发针对特定需求的个性化组件，然后按照预先定义的开发流程，将这些组件组装在一起，形成针对该型号产品的应用软件。

3.1 组件的开发

在组件库中，共性组件可以直接复用，不必重新开发，因而组件开发的重点是按照软件架构的规定，开发针对特定需求的个性化组件。在该软件架构中，待开发的组件集中在业务逻辑层和系统交互层内。业务逻辑层组件和系统交互层组件的组合方式如图 2 所示。

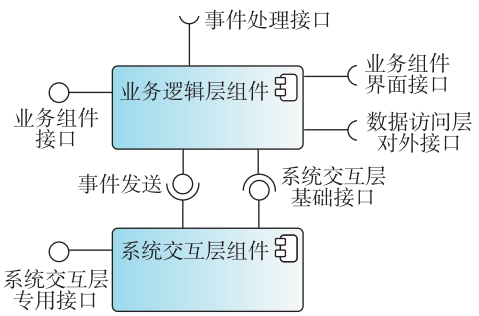

图 2 组件关系示意图

Fig. 2 The diagram of component relationship

在导航计算机测试中，核心的测试业务包括测试过程中的电源监测、功能和性能指标测试、老化测试、产品标定测试、监控上传等。这些业务组件遵循架构中定义的业务组件接口，通过事件接口实现组件间的隐式交互；通过数据访问层对外接口获

取和保存测试数据；通过系统交互层基础接口，实现与测试设备硬件的交互。在业务层组件开发时，这些核心组件一般以大颗粒度的、带有界面的控件形式存在，以确保每个组件能够完整地实现一项相对独立的业务需求。因为需求相对独立，所以修改和新增组件时，不会对其他组件产生太大影响。

系统交互层组件必须实现该层定义的基础接口和专用接口。在软件架构中，每个型号的测试设备，都对应一个系统交互组件和若干个电气连接文件。系统交互组件通过调用对应硬件的驱动程序实现了对各种测试仪器和板卡的控制功能。测试设备的电气连接关系以表格或配置文件的形式单独封装。在配置文件内部描述了测试设备的对外连接端口，以及各连接端口与硬件仪器间的映射关系。当交互层组件建立后，测试软件根据这些配置文件，实现对各硬件仪器和板卡的控制。

以产品切换控制功能为例，在硬件层面，切换控制功能可以采用外设部件互连标准（Peripheral Component Interconnect，PCI）总线形式的专用板卡，也可以采用标准的继电器仪器。在交互层组件开发时，对于前者，使用硬件厂商提供的应用程序编程接口（Application Programming Interface，API）函数；对于后者可直接调用 IviSwtch 类驱动[16-17]。在交互层组件开发和调试完毕后，除非硬件发生变化，否则该组件和配置文件基本不会修改。

3.2　测试软件的快速开发流程

在架构设计时，预先定义了几种典型场景下的软件开发流程，为软件开发人员提供指导，便于后续的应用软件开发、修改和完善。

场景 1：在全新的测试设备上，进行首个测试软件的开发。开发人员的工作包括两点，首先针对软件架构中的系统交互层，抽象出与该设备功能对应的专用接口，开发系统交互层组件，该组件开发完毕后可直接复用；其次，根据测试产品的测试需求，分解具体的测试业务，开发测试业务组件，形成的组件成果可作为后续其他型号开发的模板。

场景 2：日常的完善性或更正性维护。从实际情况看，这些维护工作基本集中在业务逻辑层。在组件进行针对性修改后，可直接进行替换。

场景 3：增加对新增产品的测试支持。一种测试设备经过扩展后，可以实现对新增型号产品的测试。此时，采用增量化开发方法，针对新增的需求，开发特定的业务组件，并通过插件方式增加到已有的程序中。在开发过程中，可直接以现有的业务组件为模板，衍生出特定的业务组件。

场景 4：测试软件的移植。测试设备的硬件在升级和改造后，已有的测试软件可以快速移植，避免了重复开发。在架构设计时，与硬件相关的内容已经统一划分在系统交互层中，同时业务层组件只和系统交互层对外接口关联，所以在移植前，只需要重新开发和修改交互层组件，即可直接进行替换。

4　结论

本文以导航计算机测试这一特定领域为目标，设计了基于组件的分层式软件架构，为测试软件开发提供了统一的解决方案。该方案规范了软件开发的整体流程，避免了测试软件的分立开发问题；通过架构复用和组件复用等方式，使得测试软件可以在一套基础平台下开发，减少了重复劳动，提高了软件的开发质量和开发效率。

目前，该软件架构已经在多种导航计算机测试软件项目中得到应用。从使用情况看，在软件开发的前期，因为需要搭建软件的整体架构、开发共性组件、建立配置文件、进行软件的调试和验证等，工作量较大，占用时间较长。但是这些基础工作完毕后，其成果可以直接复用在后面的所有软件项目中，并且随着组件库的不断更新、扩展和完善，开发周期大幅缩减；同时，这些软件遵循相同的体系结构，软件间具有很强的延续性，有效降低了修改、维护和管理的难度，可以很方便地针对不同需求进行功能扩展，满足不同被测产品、在不同测试设备上的测试软件开发需要。

参考文献

[1] 孙昌爱，金茂忠，刘超．软件体系结构研究综述[J]．软件学报，2002，13(7)：1228-1237.
Sun Chang'ai，Jin Maozhong，Liu Chao. Overviews on software architecture research[J]. Journal of Software，2002，13(7)：1228-1237(in Chinese).

[2] 马苏拉．软件体系结构的广义视角研究[J]．计算机工程，2012，38(23)：42-46.
Ma Sula. A general view of software architecture research[J]. Computer Engineering，2012，38(23)：42-46(in Chinese).

[3] 朱雪阳．软件体系结构形式描述研究[D]．北京：中国科学院大学，2004.
Zhu Xueyang. Study on formal description of software architecture[D]. Beijing：University of Chinese

Academy of Sciences, 2004(in Chinese).

[4] 赵海源，王丽芳，蒋泽军．基于组件化思想的测控软件开发平台设计与实现[J]．电子设计工程，2013，21(8)：80-83.
Zhao Haiyuan, Wang Lifang, Jiang Zejun. Design and implementation of general development platform for measurement and control software based on componentization of ideas[J]. Electronic Design Engineering, 2013, 21(8): 80-83(in Chinese).

[5] 尹禄高，刘旺开，沈为群．基于组件与设计模式的测控系统[J]．计算机测量与控制，2010，18(2)：360-362.
Yin Lugao, Liu Wangkai, Shen Weiqun. Measurement and control system based on components and design patterns[J]. Computer Measurement & Control, 2010, 18(2): 360-362(in Chinese).

[6] 王志刚，胥茜．软件架构核心问题与关键决策探讨[J]．软件导刊，2017，16(12)：57-63.
Wang Zhigang, Xu Qian. Discussions of the core issues and key decisions of software architecture[J]. Software Guide, 2017, 16(12): 57-63(in Chinese).

[7] 刘锋，孙泳．设计模式及组件技术在业务逻辑层中的应用[J]．计算机系统应用，2011，20(10)：154-159.
Liu Feng, Sun Yong. Application of design patterns and component technology to business logic layer[J]. Computer Systems Applications, 2011, 20(10): 154-159(in Chinese).

[8] 王雷，王智广．改进的三层架构的研究与应用[J]．计算机工程与设计，2017，38(7)：1808-1812.
Wang Lei, Wang Zhiguang. Research and application of improved three-tier architecture[J]. Computer Engineering and Design, 2017, 38(7): 1808-1812(in Chinese).

[9] Mitra S. Using UML modeling to facilitate three-tier architecture projects in software engineering courses [J]. ACM Transactions on Computing Education, 2014, 14(3): 1-31.

[10] IVI Foundation. IVI-3.1: driver architecture specification[S]. Revision 3.8., 2019: 1-144.

[11] 陈宝华，席泽敏，王俊茂，等．自动测试系统互换性技术研究[J]．电子测量技术，2010，33(7)：1-29.
Chen Baohua, Xi Zemin, Wang Junmao, et al. Study for interchangeability technology of auto test system [J]. Electronic Measurement Technology, 2010, 33(7): 1-29(in Chinese).

[12] 杨锁昌，安幼林，冯振声．基于功能封装的仪器驱动程序可互换性实现技术研究[J]．计算机测量与控制，2009，17(6)：1225-1228.
Yang Suochang, An Youlin, Feng Zhensheng. Implementation technique of instrument driver interchangeability based on functional package[J]. Computer Measurement & Control, 2009, 17(6): 1225-1228(in Chinese).

[13] 贾海明，安幼林．功能封装的仪器驱动程序可互换性实现技术[J]．兵工自动化，2009，28(11)：79-92.
Jia Haiming, An Youlin. Function of instrument package driver technology interchangeability[J]. Ordnance Industry Automation, 2009, 28(11): 79-92(in Chinese).

[14] 叶小杰，龙兵，谢志富，等．自动测试系统中的仪器设备驱动通用封装研究[J]．计算机测量与控制，2011，19(4)：975-977.
Ye Xiaojie, Long Bing, Xie Zhifu, et al. Study of virtual instrument generic packaging in automated test system[J]. Computer Measurement & Control, 2011, 19(4): 975-977(in Chinese).

[15] Behnel S, Fiege L, Muhl G. On quality of service and publish-subscribe[C]// Proceedings of 26th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'06). Lisboa: IEEE, 2006: 20-25.

[16] IVI Foundation. IVI-4.6: IviSwtch class specification [S]. Revision 4.0., 2017: 1-132.

[17] 唐希浪，肖明清，薛辉辉，等．IVI.NET 仪器驱动及其应用研究[J]．测控技术，2015，34(10)：111-114.
Tang Xilang, Xiao Mingqing, Xue Huihui, et al. Research on IVI.NET instrument driver and its application[J]. Measurement & Control Technology, 2015, 34(10): 111-114(in Chinese).