第35卷 第18期 计 算 机 工 程 2009年9月
Vol.35 No.18 Computer Engineering September 2009
·软件技术与数据库· 文章编号：1000—3428(2009)18—0072—03 文献标识码：A 中图分类号：N945

并行自动测试系统软件体系结构建模

卓家靖[1,2]，孟 晨[1]，方 丹[1]
(1. 军械工程学院导弹工程系，石家庄 050003；2. 武汉军械士官学校，武汉 430075)

摘 要：针对并行自动测试系统软件开发难度大的问题，对其软件体系结构进行研究，提出层次化风格的并行自动测试系统软件体系结构及五视图的软件体系结构描述模型，建立该软件体系结构的需求功能视图、框架视图、静态结构视图、动态过程视图和物理实现视图。该描述模型可以对并行自动测试系统的软件体系结构进行全面的分析，有利于指导系统的实际开发。

关键词：自动测试系统；并行测试；软件体系结构；软件体系结构模型

Software Architecture Modeling of Parallel Auto Test System

ZHUO Jia-jing[1,2], MENG Chen[1], FANG Dan[1]
(1. Department of Missile Engineering, Ordnance Engineering College, Shijiazhuang 050003;
2. Wuhan Ordnance Noncommissioned Officer School, Wuhan 430075)

【Abstract】 Aiming at the difficulty on software development of parallel auto test system, this paper proposes the layered software architecture of parallel auto test system and the description model for the software architecture. The description model includes five views: requirement function view, frame view, static structure view, dynamic process view and physical implementation view. The software architecture of parallel auto test system can analyze roundly through the description model. And it is benefit for guiding the development of parallel auto test system.

【Key words】 auto test system; parallel test; software architecture; software architecture model

1 概述

软件体系结构体现系统高层次的抽象，着重解决软件系统的结构和需求向实现平坦过渡的问题[1]。研究软件体系结构的目的是为软件系统提供合理的构架，重点解决应用系统开发中的总体结构问题。

在自动测试系统领域，尤其对于要求多个测试任务并行执行的并行自动测试系统来说，测试过程和测试仪器功能的复杂化和多样化使现代自动测试系统的组建，特别是测试软件的设计难度成倍增加。因此，需要构建良好的测试系统的软件体系结构来指导系统的开发。文献[2]研究了自动测试系统面向对象框架的开发方法，但该框架没有考虑并行测试系统的具体情况。文献[3]提出了基于CORBA并行测试软件的体系结构，其优势主要体现在分布式的系统应用中。文献[4]给出了基于统一建模语言(UML)的并行测试系统分析与设计，但没有对整个系统平台的体系结构进行完善的描述和分析。本文主要研究单处理器的并行自动测试系统软件体系结构，建立层次化风格的并行自动测试系统软件体系结构，并对该体系结构进行五视图的建模描述。

2 层次化并行自动测试系统软件体系结构

目前通用自动测试系统(Generic ATS, GATS)的软件体系结构主要依据IEEE发布的宽域测试环境标准(IEEE Standard for A Broad Base Environment for Test, ABBET)[5-6]。ABBET将测试系统划分成5个层次：产品描述层，测试需求/策略层，测试程序层，资源管理层和仪器控制层。并行自动测试系统具有串行自动测试系统的基本特性，其核心测试过程与串行自动测试系统的核心测试过程是一致的，因此，并行自动测试系统的软件体系结构也采用层次化的体系结构风格。

并行自动测试系统软件体系结构与传统串行自动测试系统的软件体系结构的不同在于其任务/资源管理层更加复杂。由于多任务的并行执行，在这一层中不仅资源管理的功能需要扩展，以满足测试资源安全共享的要求，还需要对测试任务进行管理，根据任务过程模型和任务调度模型合理调度任务的执行。

3 并行自动测试系统软件体系结构建模

对软件体系结构的建模需要从不同的视角进行描述。文献[7]提出了“4+1”视图模型。文献[8]从UML的角度将“4+1”视图模型映射到UML的各种视图。Rational公司在IEEE P1471推荐的体系结构描述的概念框架基础上起草了可重用的体系结构描述框架，建议从需求视点、设计视点、实现视点和测试视点4个视点出发描述体系结构[9]。

实际上，这些模型描述方法都可归纳为从需求分析、静态结构、动态过程和物理实现4个方面对软件体系结构进行建模描述。这4个方面基本反映了系统构造的几个主要方面，但并不能完全反映软件体系结构的特点，尤其是软件体系结构风格不能从这些模型中得到体现。同时，这些模型之间相互的映射关系缺少描述，软件系统的开发过程不能得到清晰的表现。因此，本文提出一个五视图的软件体系结构模型，在需求功能视图、静态结构视图、动态过程视图和物理实现视图的基础上增加了一个描述系统解决方案的框架视图。系统框架视图可体现软件体系结构风格，通过它把系统的需求功能视图过渡到静态结构视图、动态过程视图和物理实现视图，可以实现各模型的相互映射，完整地描述软件体系结构。具体的视图构建可利用UML的各种框图实现。

作者简介：卓家靖(1979－)，男，博士研究生，主研方向：自动测试系统；孟 晨，教授、博士生导师；方 丹，博士研究生

收稿日期：2009-01-19 **E-mail：**daxiazjj@126.com

3.1 需求功能视图

需求功能视图主要从测试相关人员的角度来描述系统必须提供的功能，是系统开发目标的直接体现，有助于系统的开发。可利用用例图描述系统需求功能视图。并行自动测试系统的角色主要是用户和被测对象，用户可泛化为测试执行人员和测试开发人员。用例体现系统的功能，主要包括任务信息管理、系统测试、系统维护和系统服务等。具体的用例图如图 1 所示，其中，任务信息管理用例主要完成测试任务信息以及任务间过程模型信息的管理；系统维护功能主要包括系统的自检、自校及配置；系统测试功能主要完成系统对被测对象的测试；系统服务功能主要包括测试结果数据管理、故障诊断和系统帮助等。

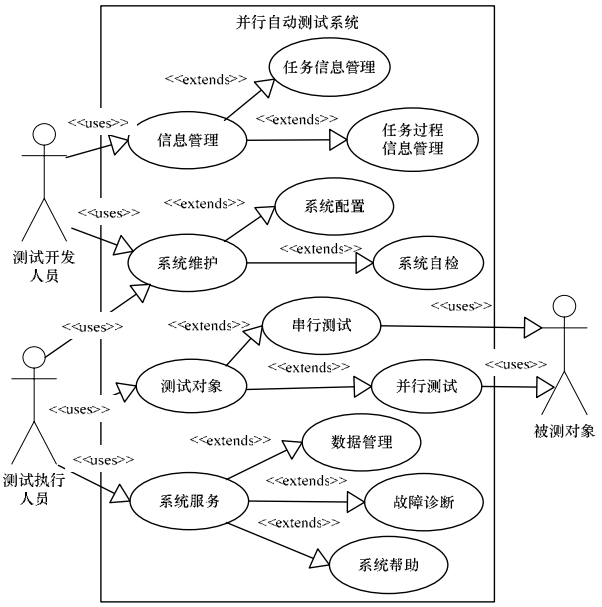

图 1　并行自动测试系统软件体系结构需求功能视图

3.2 框架视图

系统框架视图是对系统的构成进行大粒度的抽象，分析系统的子系统组成以及子系统之间的关系，体现软件体系结构风格，指导并制约结构视图、过程视图和物理视图的构造。框架视图可以利用包图表示。对于并行自动测试系统，测试执行人员或开发人员都是通过用户接口层的用户界面实现与系统的交互的，同时在测试程序层，系统不同功能需要相应的子程序来实现。系统测试对象功能的实现在仪器任务/资源管理层和仪器控制层都需要相应的程序来完成。因此，可建立如图 2 所示的系统软件体系结构框架视图。

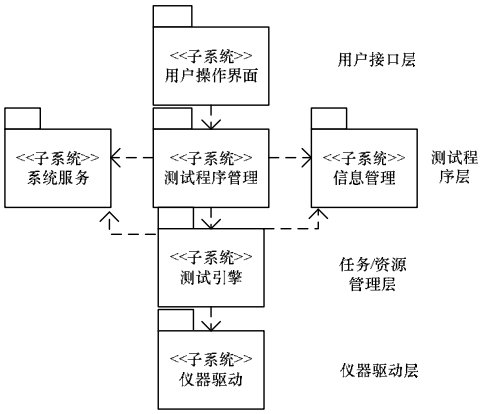

图 2　并行自动测试系统软件体系结构框架视图

3.3 静态结构视图

静态结构视图是对框架视图的细化，描述了系统具体的软件功能模块，体现了系统的静态结构。结构视图可用类图描述。框架视图已经对系统的结构做了初步的总结，因此，类图可以通过分析用例的实现，并在框架视图的基础上对系统进行细化来构建。体现系统软件体系结构静态模型的类图如图 3 所示。

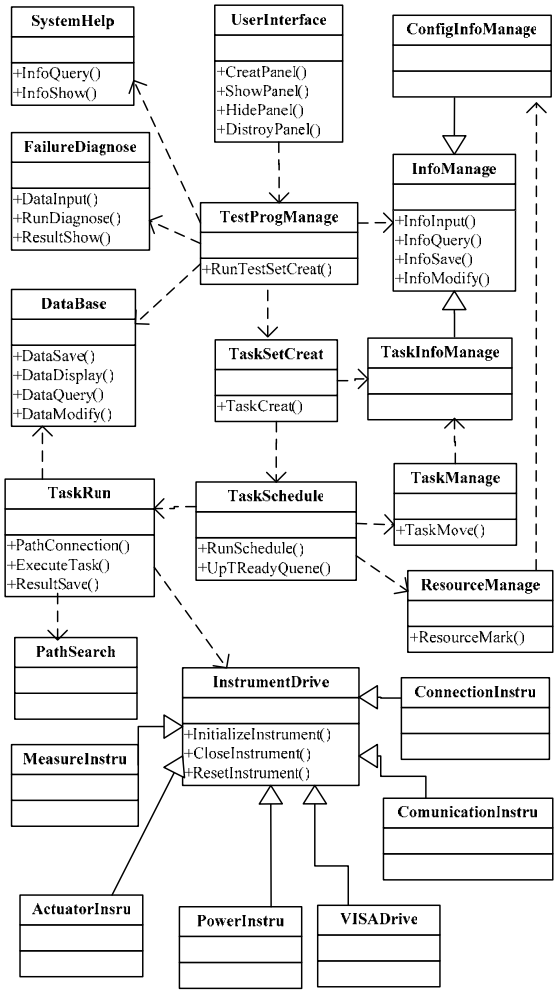

图 3　并行自动测试系统软件体系结构静态结构视图

用户操作界面子系统中包括用户操作界面类(UserInterface)。用户操作界面类在前台向用户提供各种操作面板和操作按钮，在后台通过调用测试程序管理类的功能完成用户指定的操作。系统服务子系统中主要包括测试数据库类(Database)、故障诊断类(FailureDiagnose)和系统帮助类(SystemHelp)。Database 类完成测试数据库的管理；FailureDiagnose 类根据故障诊断策略，调用测试数据库类读取测试数据进行故障诊断；SystemHelp 类主要提供帮助信息的查询。测试程序管理子系统主要包括测试程序管理类(TestprogManage)，主要根据用户不同的操作请求调用相应的应用子系统来实现具体的功能，在任务执行过程中响应用户对测试程序的控制。信息管理子系统主要包括信息管理类(InfoManage)，具体又泛化为任务信息类(TaskInfoManage)和系统配置信息类(ConfigInfoManage)，主要完成系统配置信息和任务信息的输入、查询、保存和修改等。测试程序执行管理子系统主要包括任务集创建类(TaskSetCreat)、任务管理类(TaskManage)、资源管理类(ResouceManage)、任务调度类

(TaskSchedule)、连接路径搜索类(PathSearch)和任务执行类(TaskRun)，主要实现系统核心测试过程的任务调度和运行功能。仪器驱动子系统主要包括仪器驱动类(InstrumentDrive)和 VISA I/O 接口驱动类(VISADrive)，主要提供各种硬件资源的驱动程序，驱动硬件操作，实现对被测对象的激励与测量。仪器驱动类具体又可泛化为各种类型的仪器类。

3.4 动态过程视图

所有系统均可表示为 2 个方面：静态结构和动态行为。本文用类图描述了并行自动测试系统的静态结构，但只描述了系统中各对象之间的静态关系，无法具体地表示每个用例的实现过程。要详细了解系统功能的实现，还需要对系统的动态行为过程进行分析和描述。由于并行自动测试系统中最重要也最复杂的就是并行测试对象用例的实现过程，因此本文仅对该用例建立活动图来描述其动态过程。

并行自动测试系统测试对象用例的核心测试过程是：测试程序管理类响应测试执行人员在用户操作界面的启动操作后，调用并行任务调度类，任务调度类根据任务管理类信息和资源管理类信息调度任务进入就绪队列，就绪队列中的任务通过调用仪器驱动程序并行地执行。该过程具体的活动图如图 4 所示。

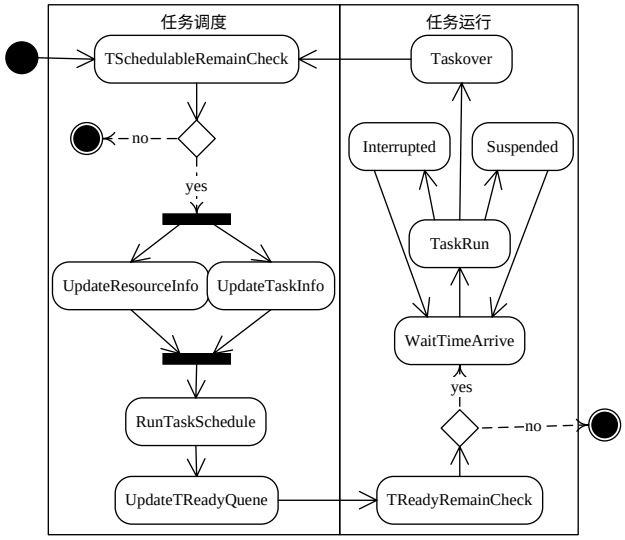

图 4　并行自动测试系统软件体系结构动态过程视图

测试任务调度的运行是由测试管理程序或任务运行结束信号启动的，之后先判断是否有剩余的可调度任务，若有，则更新任务和资源信息，调度可调度任务进入就绪任务队列，输出新的就绪任务队列；若没有剩余的可调度任务，则任务调度活动结束。任务运行活动首先检查任务就绪队列是否为空，若不为空，则按时间片轮转并发地运行任务就绪队列中的各个就绪任务，每个就绪任务在运行过程中可能被中断或阻塞，返回或唤醒后都需要重新回到就绪任务队列，等待时间片到达再继续执行，每次任务运行结束时，都向任务调度发送任务结束信号，启动新一次的任务调度活动；若就绪任务队列为空，则任务都运行完毕，活动结束。通常，任务调度与任务运行是并行执行的，任务调度活动结束并不表示整个活动的结束，此时任务运行活动还在进行，并且任务运行过程中存在多个实例并行运行，每结束一个实例就向任务调度发送消息，任务调度每接收一个任务结束消息就运行一次。只有所有任务都运行结束，整个任务调度与运行活动才结束。

3.5 物理实现视图

软件系统开发的结果是得到具体的可执行和应用的软件物理制品，这些软件制品最终要在硬件系统上部署，在物理节点上运行，因此，完整地描述软件体系结构还需要建立描述软件部署情况的物理实现视图。具体的描述方法可以利用部署图表示。对于本文研究的并行自动测试系统，主要考虑单处理器的情况，所有的软件都运行在测试控制器上。系统硬件设备还包括各种硬件资源(激励设备、测量设备、开关设备、电源设备和总线接口设备等)和被测对象。系统部署图如图 5 所示。

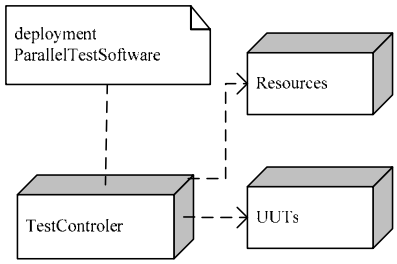

图 5　并行自动测试系统软件体系结构物理实现视图

4　结束语

并行测试是一门新兴的测试技术，并行自动测试系统运行过程复杂，软件开发难度大，因此，深入研究其软件体系结构对推动并行测试技术的发展和应用是十分必要和重要的。本文建立了层次化的并行自动测试系统软件体系结构，并构建了功能视图、框架视图、结构视图、过程视图和物理视图对该软件体系结构的需求分析、设计方案、静态模型、动态模型和系统部署等 5 个方面进行详细的描述，可指导并行自动测试系统的应用开发。

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.

[2] 徐小良. 自动测试系统的面向对象框架开发方法研究[D]. 杭州: 浙江大学, 2003.

[3] 郑　鑫, 陈希林, 周越文. 基于 CORBA 的导弹通用测试平台并行测试实现[J]. 计算机工程, 2007, 33(5): 249-251.

[4] 夏　锐, 肖明清. 基于 UML 的并行自动测试系统的设计与实现[J]. 计算机工程, 2007, 33(9): 62-63, 108.

[5] IEEE Standard 1226-1998 IEEE Trial-use Standard for A Broad Based Environment for Test[S]. 1998.

[6] IEEE Standard 1226.3-1998 IEEE Trial-use Standard for Software Interface for Resource Management for A Broad Based Environment for Test[S]. 1998.

[7] Kruchten P. The 4+1 View Model of Architecture[J]. IEEE Software, 1995, 12(6): 42-50.

[8] Booch G. Software Architecture and the UML[Z]. (2006-08-13). http://portal.acm.org/citation.cfm.

[9] 张友生. 软件体系结构[M]. 北京: 清华大学出版社, 2004.

编辑　张　帆