硬件系统自动化测试的多视角建模及案例研究

孟翰[1],吴际[1],胡京徽[1]，刘超[1]，杨海燕[1]，孙新颖[1]

[1](北京航空航天大学 计算机学院,北京 100191)

摘 要：针对硬件设备的自动测试设备（ATE）开发通常是一个冗长、耗时的任务，开发者需要向来自不同方面的专业人员了解被测设备的外部端口、信号、测试流程和信号检查等多种类型的信息，逐步确定开发需求。在这个过程中，ATE 的开发人员遇到的最大困难是缺乏一种规范的模型,以描述由不同合作方提供的测试信息，导致产出的测试文档篇幅长、不易理解，容易出现错误等问题。本文提出面向 ATE 领域的多视角建模方法，来规范化描述 ATE 中的测试信息、检查信息的一致性,并以真实工业案例说明其有效性。

关键词：硬件设备自动测试系统；测试文档；多视角建模方法；一致性；

Modeling hardware system automatic test in multiple views: industrial case study

MENG Han[1], WU Ji[1], HU Jing Hui[1],LIU Chao[1],YANG Hai Yan[1]

[1](School of Computer Science and Engineering, Beihang University, Beijing 100084, China)

Abstract: Focusing on the development of ATE(Automatic Test Equipment) is a tedious task. It requires developers to understand various information about external ports, signals, test procedures, and signal inspections of measured equipment from specialists, which would confirm the demand of development. In this process, the most complicated issue for developers is lacking a normative model, which can describe the test information from various collaborators. This issue results in several problems of verboseness of test documents, difficulty of understanding and excess of errors. The objective of this paper is to propose a multi-angle views modeling method that oriented to ATE domain. This model can normalize the test information of ATE and support the consistency checking. In the meantime, the paper would give industrial case to demonstrate the effectiveness of the model.

Key words:ATE; test documents; multi-angle views modeling method;consistency checking

1 前言

随着计算机软硬件技术、网络技术，以及自动化和智能化技术的快速发展和广泛应用，在飞机、舰船、高铁、电力、医疗等众多领域已广泛采用各种复杂的自动化乃至智能化电子电器设备，用于各种检测和控制目的，因此，确保其符合规范要求和安全运行至关重要。为此，在这些设备和系统的生产制造及运行维护期间，需要配备相应的自动测试设备(ATE)，为设备自动检测提供必需的支持。

ATE 主要用于硬件设备的信号测试（如电压、电流、电阻等）。这些被测设备种类繁多，遵循的标准规范各异。即便是同类设备，也存在不同厂家和不同型号之间的差异。因此，对于这些设备，虽然存在许多相同或相似的测试需求，但各自也有很多不同的特殊要求，致使大多数的 ATE 都是针对特定领域中某一型号或某几种型号被测设备的专用设备。

使用 ATE 测试被测设备的基本方式是将其测试端口与被测设备的被测端口进行物理连接，通过指定的引脚向被测设备发送测试信号，同时接收其反馈信号，进而分析、显示和判断被测设备的状态是

到稿日期：2017-08-05 本文受科工局技术基础项目（编号：JSZL2014601B008）资助

孟翰(1993－)，男，硕士，主要研究领域为软件需求建模，Email:mengh_buaa@163.com；刘超(1958－)，男，博士，教授，主要研究领域为软件测试等；吴际(1974－)，男，博士，副教授，主要研究领域为软件安全性与可靠性等；杨海燕(1974-)，女，硕士，讲师，主要研究领域为软件测试等；胡京徽（1992-），男，硕士，主要研究领域为软件需求等；孙新颖（1982-），男，硕士，目前在北京天创凯睿科技有限公司任职，是本文研究的工业合作伙伴。

否符合其规范要求。此外，ATE 测试需要按照一定的流程步骤来组织测试动作。在测试中，完整地表达测试流程对于判断测试结果的正确性和进行必要的调试是至关重要。由此可见，开发 ATE 测试设备需要包含四类基本信息，即测试信号、测试设备和被测设备的引脚，以及测试流程。ATE 领域的测试文档是用于描述开发特定 ATE 测试设备所需要的诸如信号、引脚、测试流程等全部测试信息的文档。

由于 ATE 设备的多样性、专用性、多专业综合性等特点，目前尚缺少一种统一和规范的 ATE 测试信息描述方法。然而，使用自然语言编制描述测试文档，显然存在以下三个方面的问题：

1）描述的规范性：针对各项测试信息的描述往往被混杂在一起，难以准确识别和理解；随意的描述方式也使得针对各类测试信息所提供的具体信息之间存在显著差异。

2）描述的完整性：实际的测试文档中往往缺少部分必要的相关信息，如引脚测量信号类型缺失、信号量纲缺失等。

3） 描述的一致性：

3.1）采用（无约束的）自然语言描述方式，容易出现专业术语、缩略语等使用上的不一致，甚至矛盾（冲突）等问题。

3.2）由于涉及跨领域和跨专业的不同知识体系，不同人员提供的测试信息所使用的标准、概念、方法、策略、规则可能存在差异，从而导致测试文档中的信息经常产生歧义，进而造成开发效率低下的问题。

综上，如何规范化的描述 ATE 测试文档中的四类信息，如何检查各种描述中出现的不一致、内容缺失等现象，是提高 ATE 测试文档质量和开发效率的关键。基于此，本文提出一种面向 ATE 领域的多视角建模方法以解决上述问题。

本文的组织结构如下：第 2 章，介绍基于视图的模型构建；第 3 章介绍基于测试信息元模型的一致性检查规则。第 4 章介绍该模型在实际工业领域的应用。第 5 章介绍相关研究。

2 基于视图的模型构建

如第一章所述，目前的 ATE 测试文档中必须包含四类信息：信号（Signal）、被测设备引脚（Pinport）、测试设备引脚和测试流程（Test Procedure）。其中，每个被测设备引脚和测试设备引脚都必须和一个信号严格对应，即每个引脚都只能输入/输出一种类型的信号，测试流程需要引用引脚、信号这些测试信息来组成一组连续的测试步骤，因此本文将四类测试信息归纳为四类视图，为了符合 ATE 领域的名称习惯，各类视图的名称、内容定义如表 1 所示。

表 1 四类视图

视图名称	内容定义
信号定义视图	被测设备中包含的全部测试信号
IO 定义视图	测试信号与 ATE 被测设备的引脚之间的映射关系
测试能力视图	测试信号与 ATE 测试设备的引脚之间的映射关系
测试流程视图	描述 ATE 测试用例的执行流程或典型场景。

其中前三类视图描述了 ATE 测试设备和被测设备的物理特性，后一类视图描述了在实际测试时测试设备对被测设备执行的测试逻辑。

元模型是关于模型的模型，定义概念并提供用于创建该领域中的模型的构建元素。本章将从不同的视角来构建 ATE 领域的测试信息元模型。

2.1 基于硬件视图的模型构建

2.1.1 信号定义视图

为了完整捕获 ATE 领域中的信号类型信息，信号定义视图（SignalTypeView）通常定义了被测设备需要被测试的全部信号类型以及信号的属性、约束，每类信号需要覆盖到以下信息：

信号名称：该信号的唯一标识，相同的名称代表相同的信号类型。

量纲：该信号类型的单位名称。

取值：该信号类型的实际取值数值类型（整形、浮点型、0/1 型等）。

测量（添加）方式：在读取或添加该信号类型时需要的引脚数。

2.1.2 I/O 定义视图

ATE 领域的 I/O 定义视图（IODefinitionView）是为了捕获被测产品的所有引脚，以及所有引脚对应的信号类型，ATE 的被测产品中引脚对应的信号类型有严格的对应关系，此外对于信号的取值有严格的约束，具体如下：

引脚号：被测产品的引脚号的唯一标识。

信号方向：表示对应于该引脚的信号是自设备

输出还是输入到设备。

信号类型：对应于该引脚的信号类型。

标准值：可在此引脚添加的信号类型取值的标准值。

上限：在此引脚添加的信号取值的最大值。

下限：在此引脚添加的信号取值的最小值。

2.1.3 测试能力视图

测试能力视图（TestCapbilityView）定义了测试设备能够测量的全部信号类型以及信号类型与引脚的对应关系。测试工程师需要这些信息来决定测试设备和被测设备的具体接线方式（引脚和引脚如何相连），测试能力视图定义的每一个测试设备的引脚的测试能力需要具备以下信息：

引脚号：测试设备的引脚号的唯一标识。

端口类型：标识该组引脚的测量信号类型。

功能：用户备注的该组引脚的实际功能，可为空。

数据范围：设备测试能力允许测量的最大范围。

精度：设备测量的精度。

数量：该类测试能力需占用的引脚数。

总结三类视图，一共涉及了信号、引脚和视图三类元素，视图又可以被细分，这些类存在着不同类型的关联关系。此外，不同类的视图存在着引用相同类的对象的现象（IO 定义视图和信号定义视图都引用了信号类），这是造成 ATE 领域中信息不一致现象的主要原因。

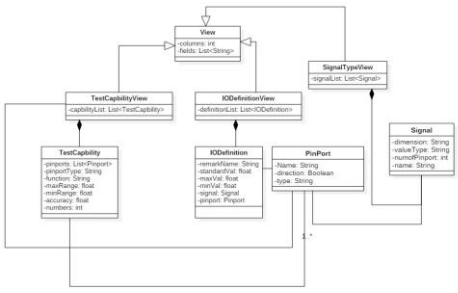

图 1 硬件视图信息构建的元模型

2.2 基于测试流程视图的模型构建

在有了上述三类基于硬件物理特性的视图后，测试工程师还需要一类视图来根据已有的信号、引脚、测试设备、被测设备信息撰写测试流程。

由于 ATE 的测试流程中经常会出现测试分支和异常情况，因此 ATE 描述的测试流程通常采用区分主测试流（MainTestFlow）和分支测试流（BranchValidationFlow）的方法，其中分支测试流根据应用场景不同又可细分为三类，全部四类测试流如下所示：

1）主测试流：描述理想情况下测试正确进行的测试流程。

2）特定分支测试流（SpecificValidationFlow）：描述主测试流某一步执行异常时的执行流程。

3）界限分支测试流（BoundedValidationFlow）：描述主测试流某几步执行异常时的执行流程。

4）全局分支测试流（GlobalValidationFlow）：描述主测试流全局执行异常时的执行流程。

区分主流程和分支流程的方法在使用者输入某个分支测试流程时屏蔽了其他流程的干扰，同时从视图上对各测试流进行了明确的分类。

有了测试流，测试流程的逻辑还不够完整，ATE 实际上富含丰富的测试动作。一般来说，ATE 的每一个测试步骤都严格对应一种测试动作，每种测试动作对应几个引脚、几个信号都有严格的约束。测试动作又可以被分为自动动作(AutomaticAction)（测试设备执行的动作）和人工动作(ManualAction)，由于篇幅限制，每种动作不再详细介绍，由测试流程视图中涉及的概念构建的元模型如图 2 所示。

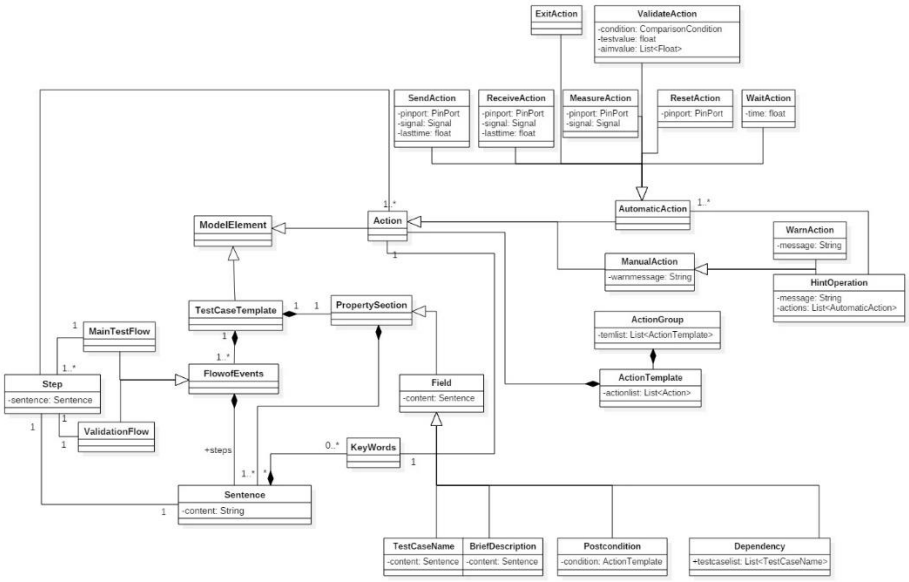

图 2 测试流程视图信息构建的元模型

2.3 整合 ATE 领域测试信息的模型构建

本文根据 ATE 中被测设备、测试设备的物理特性和测试流程分别构建了模型，将它们整合到一起可以得到涵盖 ATE 领域测试信息的元模型。有了元模型，视图中的所有元素都可以看作是元模型的实例化对象。

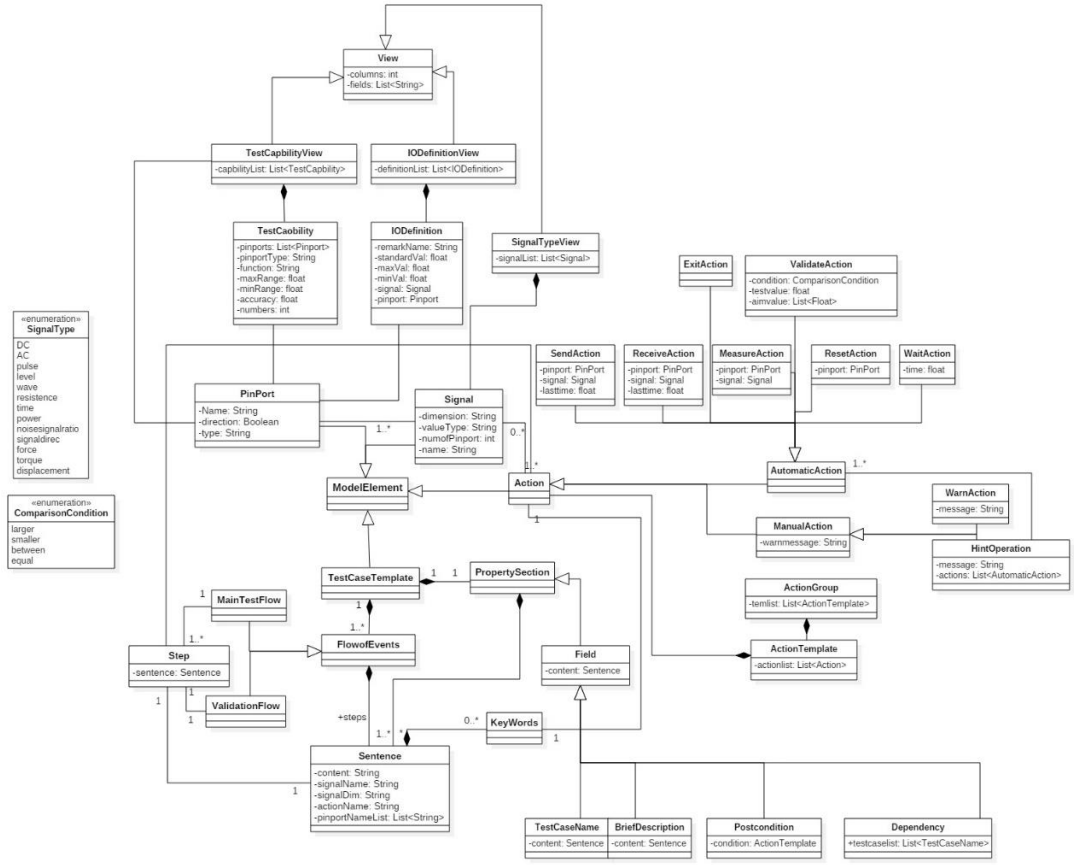

图 3 ATE 领域测试信息构建的元模型

3 基于测试信息元模型的一致性检查规则

如前文所述，造成ATE领域测试信息不一致的原因有二：

1)四类测试信息由不同参与者提供。

2)四类测试信息中存在着信息交叉，即不同视图对模型中的同一个类的对象共同引用、却有可能有不同的定义。

为了检查上述原因导致的不一致问题，本文提出了六类一致性检查规则用对象约束语言（Object Constraint Language）进行描述。

3.1 引脚类规则

针对一般情形，用户要求在测试流程中输入的引脚都已在 IO 定义视图中被定义。如果出现未被定义的引脚则被视为不合法输入，规则可被定义如下：

规则 1:测试流程视图中的引脚名称一定是引脚对象集合中的一个引脚对象的名称。

Context:Sentence

inv:self.pinportNameList->forAll(pin1|Pinport.allInstances()->select(pin2|pin2.name=pin1.name)->size()=1)

检查的错误如IO定义视图中只定义了引脚A1-A17，而实际测试流程中出现了A18引脚。

3.2 信号—引脚类规则

规则 2:测试流程视图中的所有信号名称与其所在步骤的引脚名称对应的引脚对象对应的信号名称相同。

Context:Sentence

inv:self.pinportNameList->forAll(pin1|Pinport.allInstances()->select(pin2|pin2.name=pin1.name)->forAll(pin3|pin3.type=self.signalName)

检查的错误如在测试步骤中引用的引脚只能用来添加或读取电压信号，但用户却用来对电流信号进行测试。

3.3 动作—引脚类规则

规则 3:测试流程视图中的“添加”动作对象对应步骤的引脚名称对应的方向必须为输入，所有“测量”动作对象对应步骤的引脚名称的方向必须为输出。

Context:Sentence

inv:self.allInstances->select(s1|s1.actionName=”send”)->forAll(s2|s2.pinportNameList->forAll(pin1|Pinport.allInstances()->select(pin2|pin2.name=pin1.name and pin2.direction=0)->size()=1)

inv:self.allInstances->select(s1|s1.actionName=”measure”)->forAll(s2|s2.pinportNameList->forAll(pin1|Pinport.allInstances()->select(pin2|pin2.name=pin1.name and pin2.direction=1)->size()=1)

被规则3检查出的问题是引脚规定的信号方向和实际测试动作方向的对应问题，由于篇幅所限，后续的规则不再用ocl语言表述。

规则4: 测试流程视图中添加（测量）动作对应的测试步骤中信号名称引用的引脚数与其实际信号对象在信号定义视图中的测量（添加）的值相等。

对于添加/测量动作，操作的信号需要的引脚数是被严格定义的，比如添加电压信号必须是双端即双引脚添加。

3.4 信号类规则

规则 5:测试流程视图中信号量纲名称与其信号名称在信号定义视图中的量纲属性相同。

规则5检查的是一类浅显易见的错误，即测试流程中的信号量纲与实际信号量纲不符。

规则 6:测试能力视图中定义的信号对象集合必须包含信号定义视图中定义的信号对象集合。

如果实际测试文档中出现了规则6检查出的问题，那么意味着目前的测试设备不具备测试被测设备的能力，因为测试设备上不支持某些信号的测试。

3.5 动作—信号—引脚类规则

IO 定义视图中信号值的标准值与上下界也是测试工程师在测试过程中非常关注的问题，对于输入（添加）信号，添加信号的值不得超过信号定义的取值范围，否则将可能导致灾难性后果（烧毁设备），而输出（测量）信号得到的值如果超过了信号定义的取值范围，则意味着测试异常的出现。

对于输入信号的取值问题，电压信号是相对特殊的，在两个引脚之间添加电压，实际上是取两个引脚电压之间的差值，但由于领域用户的普遍习惯，都采用两端引脚之间添加 xV 电压的描述方法，对于电压信号，需要对其差值进行简单的计算再进行

检查，如规则 7:

规则 7:测试流程视图中测试动作为“添加”且信号名称为“电压”的步骤，其添加的信号值大于等于前引脚对应电压值下限与后引脚对应电压值上限的差，小于等于前引脚对应电压值上限与后引脚对应电压值下限的差。

对于非电压信号的输入，只需保证其输入值在对应引脚规定的范围内即可，如规则 7:

规则 8:测试流程视图中测试动作为“添加”且信号名称不为“电压”的步骤，其添加的信号值大于等于其全部引脚对应信号值下限的最大值，小于等于其全部引脚对应信号值上限的最小值。

3.6 属性值检查规则

四类视图中的每种信息都有其实际意义，而在实际的测试需求文档中，经常出现信息漏填的丢失现象，针对这类问题可以定义规则如下：

规则 9:信号类、IO 定义类、测试能力类和测试流程类的所有对象的所有属性不能为空（Null）。

4 案例分析

如前文所述，目前 ATE 的测试文档有以下难题：

1） 测试信息描述的规范性

2） 测试信息描述的完整性

3） 测试信息描述的一致性

为了验证本文模型以及一致性检查规则是否能够有效的检测、解决以上问题，本文联系了 ATE 领域相关的工业部门，将此模型用于规范化描述测试文档中的测试信息，并提供了两个指标作为模型验证结果的参考

1）该模型对于不同测试文档中各类测试信息的覆盖情况。

2）该模型针对各测试文档检查出的不一致性错误数和实际错误数。

4.1 案例选取

本文选取了合作单位所提供的两个 ATE 测试设备开发工业案例，分别为某型直升机控制增稳系统测试设备及某型自动驾驶仪计算机测试设备。

控制增稳系统能够提高飞机稳定性并改善操纵系统的飞行控制系统。该设备的测试设备软件部分采集主增益系数、反馈参数、前馈放大系数，硬件部分通过采集俯仰角度信号进行测试[2]。

自动驾驶仪是现代飞机的重要组成部分，对保证自动驾驶仪的正常工作有重要意义。自动驾驶仪计算机测试设备就是为了验证自动驾驶仪的正确性而设计的。该测试设备主要由硬件部分、上位机主控软件、下位机驻留软件三大部分组成。测试设备可完成设备自检、自动测试、手动测试等功能[1]。

两个案例针对的被测设备结构较为复杂、开发文档篇幅冗长以及涉及的领域术语较多，对于验证本文提出模型的有效性有较强说服力。

4.2 案例结果及分析

为了开展案例研究，本文的工业合作伙伴邀请了实际的 ATE 开发工程师三名，对其进行了为期一周的培训，该三人使用本文产出模型来对上述案例的测试文档进行整理，根据反馈结果进行了三次模型的迭代，模型对于测试信息的描述情况如表 2 所示。

表 2 模型在案例一、二的应用结果

	信号类实例	被测设备引脚类实例	测试设备引脚类实例	测试流程	测试步骤	测试动作
控制增稳系统	36	148	71	48	497	8
V1.0	28	113	60	48	402	6
V1.1	34	138	65	48	482	6
V1.2	34	138	65	48	488	7
自动驾驶仪	48	188	90	42	513	10
V1.0	30	132	67	42	398	8
V1.1	45	184	84	42	465	8
V1.2	45	184	88	42	499	10

在模型 v1.0 中，许多信号类型没有被明确定义，造成了许多引脚、测试步骤都无法被描述，在模型 v1.1 中加入了更多信号，二次迭代后发现除了 ATE 测试设备进行的设备自动测试动作外，实际测试过程中经常会有人工测试动作参与，三次迭代更新了测试动作。最终的模型有个别信号、引脚和动作没有覆盖（渐变信号、循环动作等），这是因为这些信号和动作的还没有找到一种合适的描述方式。

根据第三章中定义的规则检查出的测试文档的不一致信息如表 3 所示（定义的规则都以模型 v1.2 最终版为准）。

表 3 案例一、案例二一致性检查结果

	案例 1		案例 2	
	模型查错数	有效错误数	模型查错数	有效错误数
规则 1	13	8	16	13
规则 2	8	8	3	3
规则 3	11	11	13	13
规则 4	2	2	0	0
规则 5	1	1	0	0
规则 6	9	9	5	5
规则 7	4	4	5	5
规则 8	6	5	9	8
规则 9	18	3	21	5
未检查错误	7		5	

以案例 1 为例，案例 1 中检查出了 66 处错误，而有效错误数为 46，无效错误集中出现在规则 9，这是因为在 ATE 中许多测试信息是存在默认值的，因此属性值缺失实际上代表的是取默认值，而并不是信息提供者的疏忽；规则 4、5 检查的错误较少，这是因为该规则定义的规范属于领域常识，实际很难出错；规则未检查出的错误可被分为两类，一是如上文所述，目前的模型还有个别信号、动作未被定义，二是模型检查的不一致现象是基于视图中交叉的信息，因此不交叉的信息或者交叉的信息都出现一致的错误都是无法被检查的。模型对于案例 1 和案例 2 的检错率分别为 86.8%和 89.6%。

4.3 总结

本文对 ATE 案例的分析结果以及与 ATE 领域从业人员的具体讨论，可得到以下结论：

1）通过模型化的方法能够整合并准确描述 ATE 测试需求文档中的核心概念（信号、引脚、测试流程等），因而该模型具备有效性，目前此模型已应用于实际的 ATE 测试设备开发过程中。

2）该模型能够检查出视图之间交叉信息的绝大多数不一致错误，不能检查出的错误是由于个别信号、动作还未在模型中被定义，因此是本文后续的研究内容；而对于非交叉的信息，该模型不具备检查能力。

3）该模型目前仅应用于案例一和案例二的测试文档中，其在 ATE 领域中的通用性有待进一步验证。

5 相关研究

目前 ATE 在国内外由于众多需求推动，都处于高速发展状态。ATE 测试设备正处于专用自动测试系统向通用自动测试系统的转变过程中。在通用 ATE 技术方面，按照模块化、系列化、校准化的要求，基于通用总线各类自动测试技术正陆续推出[3]。

目前自动测试系统软件的开发模式主要有四种，传统的测试软件开发模式主要以测试流程为依据，由上至下一步完成，其通用性、扩展性、维护性都很差；基于测试数据库的软件开发模式是通过数据库来实现测试项目的选择和测试流程的控制，由于其性能较差，因此不适用于需要实时采集和处理的系统；基于测试流程设计语言的开发模式目的在于构建通用测试平台，与前两种模式相比，这种模式灵活性和执行速度较高，但要求复杂的解释或编译软件；应用于国际标准 ATS 体系结构的开发模式主要是按照 ABBET 标准为基础实现测试诊断信息的共享与重用，即测试软件合理的进行分层配置，这种模式目前存在许多技术难点[4,5,6]。

通用自动化测试系统的软件按 IEEE126 ABBET[7]标准可分为 6 层；被测设备描述层、测试需求、策略层、测试程序层、测试资源管理层、仪器控制层和硬件层。其主要目的是实现面向信号的开发思想，开发测试/激励/控制信号功能借口函数，结合测试信号描述 XML 文档，完成测试程序中虚拟仪器到实际硬件仪器的映射[8]。

LabVIEW[12,13]是专为测试、测量和控制应用而设计的系统工程软件，可快速访问硬件和数据信息 LabVIEW 编程环境简化了工程应用的硬件集成，同时，LabVIEW 还降低了编程的复杂性。LabVIEW 可以与其他软件和开源语言进行互操作，并复用使用其他软件和语言编写的代码。目前 ATE 测试设备开发通常都以 LabVIEW 作为实现工具。

OCL（Object Constraint Language）[9]是针对某一特定模型的约束语言，通过在模型元素上附加 OCL 逻辑表达式来表达指定的规则，它是一种约束语言同时也是一种查询语言。是一种用于施加在指定的模型元素上约束的语言。

随着面向对象技术的广泛应用，从模型中获得测试所需要的信息的方法（基于模型的测试方法）成为自动化测试领域中的重要分支。典型模型主要包括：有限状态机、UML 模型和马尔科夫链模型三种。有限状态机的表达方式通常有状态迁移图和状态矩阵，通过分析状态覆盖或迁移覆盖方法得到测

试路径[14,15]。UML 具有描述层次状态的能力，并提供并发机制的描述，因此可以针对单个类行为进行建模，符合测试的描述方式，M.E.Viera 等人利用 XMI 工具，读取 UML 状态图的模型信息，据此信息生成测试用例脚本和测试桩模块。马尔科夫链是一种描述软件的使用统计模型，经常被应用于软件统计测试中。基于马尔科夫链的测试充分性准则一般要求对马尔科夫链的覆盖与实际使用保持一致[17]。目前基于马尔科夫链模型的统计测试在工业界应用非常广泛。

6 结束语

本文提出一种 ATE 领域的多视角建模方法来对该领域的测试信息进行规范化的描述，同时提供了检查各类视图中信息不一致的能力。经过实际工业案例的应用表明，本文建模方法构建的模型具备有效性。目前此模型已经应用于实际的 ATE 测试设备开发过程中。目前的研究还有一定的局限性。第一，该模型目前应用的案例数较少，因此其通用性有待进一步验证；第二，该模型对于个别信息的描述能力不够完备，部分信息的描述方式还在商榷中；第三，根据模型提出的一致性检查规则没有完全考虑到领域从业者的常识性信息，因此检查出的错误有部分是冗余的。这些问题是本研究后续改进的重点内容。

参考文献

[1] Isard M. Autopilot:automatic data center management[J]. Acm Sigops Operating Systems Review, 2007, 41(2):60-67.

[2] Mj V D L, Dudoit S, Pollard K S. Augmentation procedures for control of the generalized family-wise error rate and tail probabilities for the proportion of false positives.[J]. Statistical Applications in Genetics & Molecular Biology, 2004, 3(1):Article15.

[3] Yang M. Application of Standardized Hypertext and Info Database to Auto Test Equipment Software[J]. Computer Measurement & Control, 2009, 25 suppl 1(25 Suppl 1):131-8.

[4] Drenkow G.Future Test System Architecture[A].IEEE Autotestcon[C].2004.

[5] Ross w A.The Impact of Nest Generation Test Technology on Aviation Maintenance[A].IEEE Autotestcon [C],2003.

[6] IEEE Std 1226-1998,IEEE trial-use standard for a broadbased environment for test(ABBET)[S].

[7] Wilaon Weng,Ofelia Antonia Jianu,A Smart Sensing Unit for Vibration Measurement and Monitoring[J].IEEE/ASME Transactiona on Mechatronica,2010,2(15);70-78.

[8] 李恩辉，梁旭.支持自动测试系统并行测试的软件设计[J].测控技术，2008,27（3）：7-9.

[9] OMG (Object Management Group. Object Constraint Language 2.3.1[J]. Cmis.brighton.ac.uk, 2012, 91(44Part2):137--145

[10] Richters M, Gogolla M. Validating UML Models and OCL Constraints[J]. Lecture Notes in Computer Science, 2000, 1939:265--277.

[11] Akehurst D H, Bordbar B. On Querying UML Data Models with OCL[C]// International Conference on the Unified Modeling Language, Modeling Languages, Concepts, and TOOLS. Springer-Verlag, 2001:91-103.

[12] Sumathi S, Surekha P. LabVIEW based Advanced Instrumentation Systems[J]. 2007.

[13] Klinger T. Image Processing with LabVIEW and IMAQ Vision[C]// Pearson Education, 2003.

[14] Beizer B. Black-box testing: techniques for functional testing of software and systems[M]. John Wiley & Sons, Inc., 1995.

[15] Jorgensen A, Whittaker J A. An API testing method[C]. Proceedings of the International Conference on Software Testing Analysis & Review (STAREAST 2000). 2000.

[16] Vieria M., Dias M., Richardson D.. Object-Oriented Specification-Based Testing Using UML Statechart Diagrams[J]. New York: ACM Press, 2000. 246-247.

[17] Poore J H. Introduction to the special issue on: model-based statistical testing of software intensive systems[J]. Information and Software Technology, 2000, 42(12): 797-799.

作者联系方式：

孟翰 Email: mengh_buaa@163.com

13501175561 010-82317640

北京航空航天大学 学院路 37 号 100191