文章编号:1671-4598(2019)02-0129-06 DOI:10.16526/j.cnki.11—4762/tp.2019.02.029 中图分类号:TP311 文献标识码:A

分布式复杂系统软件测试建模方法与应用研究

陈 强[1],陈 双[1],吴立金[2],韩新宇[2]
(1.海军研究院,北京 100161;2.中国船舶工业综合技术经济研究院,北京 100081)

摘要:针对当前分布式复杂系统软件测试周期短、测试质量要求高、传统的软件测试方法效率低下问题,结合基于模型的软件开发技术广泛应用的现状,研究了软件测试建模技术,定义了分布式复杂系统软件测试模型,提出了分布式复杂系统软件测试建模方法,并分析应用前景,能够支撑软件测试建模及用例生成工具的研制,为分布式复杂系统软件全过程自动化测试及测试复用提供技术方法。

关键词:分布式复杂系统;软件测试;测试建模;模型转换

Research and Application of Distributed Complex System Software Test Modeling Method

Chen Qiang[1], Chen Shuang[1], Wu Lijin[2], Han Xinyu[2]
(1. Naval Research Academy, Beijing 100161, China;
2. Institute of Marine Technology & Economy, Beijing 100081, China)

Abstract: At present, distributed software systems have short test time periods, high test quality requirements, and model-based software development technology is widely used, but traditional software test methods have a low efficiency, so full-process automated test technology is urgently needed. A software test modeling method is researched, by establishing a distributed complex system software test models, proposing the test modeling process, analyzing application prospects of the technology. This method provides technical for full-process automated test and test case reuse of complex software.

Keywords: distributed complex system; software test; test modeling; model conversion

0 引言

分布式复杂系统软件一般由多个单位分别研制各子系统,再由系统设计及集成单位进行集成组合为上一层系统。不同软件研制单位在研发软件时会使用不同的平台、不同的实现技术,当这些软件在组合为上级系统时,就会遇到集成与信息交互的问题。如果每一个软件在与其他软件集成与信息交互时,都单独定义一套业务规则,那么会使得整个系统开发变得难以维护且没有可持续性。另一方面,很多软件需要使用相同的信息,有些软件具有相似功能模块,如果只是因为使用不同平台开发的因素,而对这些信息或者功能模块进行全新的设计开发,既浪费精力,也无法满足后续类似型号的研制周期缩短的要求。为了实现跨平台集成、互操作和复用,以及更有效的对整个复杂系统进行分析和设计,缩短开发周期,分布式复杂系统软件模型辅助开发新模式逐步广泛应用。

分布式复杂系统软件测试任务十分艰巨,测试时间成为了影响研制进度的重要因素,其完成质量是整个系统质量的重要保证。在系统研制过程中,软件配置项及系统测试的介入时间晚,测试时间不足,在最后的系统联调试验阶段,其中绝大部分是软件问题。随着新研装备的交付使用,后续型号装备会陆续进入研制状态,其包括的组成部分更多,功能也相应更多,然而研制周期却进行了压缩,软件配置项及系统测试工作需更早、更高质量、自动化地开展。

然而,针对这两方面需求和现状,传统的测试方法显得效率不足:

1)传统的软件测试方法自动化程度低,主要由测试人员根据软件开发文档,人工分析、手动设计编写测试用例,搭建测试环境。效率低下,测试人员没有足够的时间关注测试设计。产生大量的测试需求、测试用例等,维护起来也相当不便。

2)传统的软件测试方法在配置项测试及系统测试阶段主要是基于软件规约测试方法,其思想是根据软件需求规约、软件设计规约等生成测试用例,构建外部数据模拟器等测试环境,测试软件功能、性能和外部接口,判断输出是否符合要求。在这个过程中,软件规约是测试生成和评估检验的依据。在模型辅助开发中,软件规约表现为各种分析、设计模型,如 UML 用例图、UML 交互图、UML 类图、UML 状态图、UML 活动图等。在这种情况下,使用规约测试无法直接使用这些软件分析及设计模型,而需要将这些分析及设计模型转换为具体的要求,再生成测试用例,构建测试环境等,这样势必造成效率低下,影响研制周期。

收稿日期:2018-08-15; **修回日期**:2018-09-06。

作者简介:陈 强(1981-),男,河北保定人,工程师,主要从事软件测评方向的研究。

本文针对分布式复杂系统软件的测试需求和现状，以及传统测试方法的不足，开展基于模型的全过程自动化测试技术研究，提出分布式复杂系统软件测试建模方法，能显著的提高配置项及系统测试的效率，提高测试质量。

1 测试建模方法研究总体框架

1.1 相关技术概念

1）软件设计模型，是指在软件研制过程中，在系统分析、需求分析、设计、编码过程中所产生的各类对软件功能、结构或行为的从开发角度的抽象描述。以 UML 模型举例，包括用例图、类图、序列图、活动图、交互图、状态图等。

2）模型辅助开发，是指整个软件开发过程中使用大量对软件系统的建模进行辅助，各个阶段的工作成果为各种设计模型，即对系统的各种描述。在模型辅助开发中，系统的可用信息使用设计模型进行描述，如以 UML 用例图、UML 交互图来表示软件需求，以 UML 类图、UML 状态图、UML 活动图来表示软件设计。

3）软件测试模型，是指在软件测试过程中，在测试分析、测试设计与实现过程中所产生的各类对软件功能、结构或行为的从测试角度的抽象描述。包括测试需求模型、测试用例模型、测试环境模型、测试数据模型等。

1.2 总体研究思路

开展基于模型的全过程自动化测试技术研究是当前提高分布式复杂系统软件测试效率、提升测试质量、进而保证软件质量的重要方向。该技术涉及到众多领域（包括软硬件仿真技术、软件测试框架技术、软件测试脚本技术等），其核心基础是基于软件开发模型构建全过程的软件测试模型，其研究方案如图 1 所示。

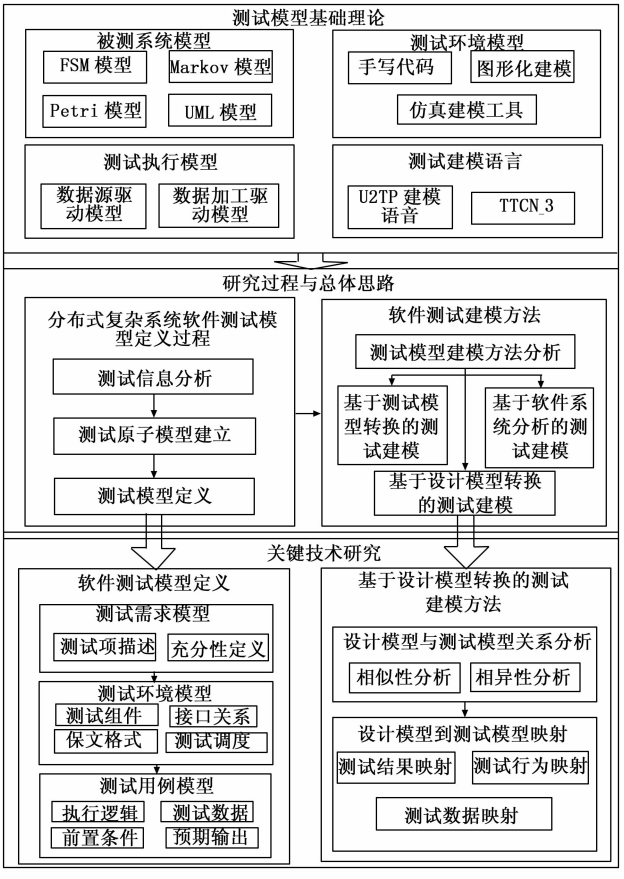

图 1 测试建模方法研究总体框架

1.2.1 测试模型调研与分析

调研现有的测试模型，包括被测系统模型、测试环境模型、测试执行模型以及测试建模语言，为分布式复杂系统软件测试模型定义奠定理论基础。现有的软件测试模型一般是局部的某一方面的对测试的描述，或单纯测试用例生成，或单纯测试环境，难以全过程使用；且由于这些模型需要再次对软件系统进行全新的建模，而没有充分考虑与开发过程模型的继承性与一致性，使得工作量增加，模型难以维护，也难以在项目之间复用。

1.2.2 分布式复杂系统软件测试模型定义

1）测试信息分析：分析分布式复杂系统软件配置项/系统测试涉及到的测试需求、测试用例、测试环境、测试数据等信息。对这些信息进行分解，得到底层要素，对底层要素进行归纳分类。

2）测试原子模型建立：针对各类底层要素，提取结构、功能和行为等特征，对各类底层要素建立测试原子模型，规定符号表示，给出文本存储方法。

3）测试模型定义：针对配置项/系统测试，对测试过程各阶段涉到的测试信息定义相应类别的测试模型，包括测试需求模型、测试用例模型、测试环境模型、测试数据模型等。给出由测试原子模型组合为各类测试模型的方式，规定各类测试模型的符号表示，给出文本存储方法。

1.2.3 分布式复杂系统软件测试建模方法

1）测试模型建模方法分析：分析分布式复杂系统软件开发中所使用到的各类设计模型，针对研究内容（1）给出的各测试模型（包括测试需求模型、测试用例模型、测试环境模型、测试数据模型等），分析各设计模型与之的关系，以及各类测试模型之间的关系，从而进一步得出各模型的建模方法，包括：通过设计模型转换得出、通过软件系统分析人工构建、通过测试模型之间相互转换得出。

2）基于设计模型转换的测试建模：此类模型至少包括测试需求模型。分析分布式复杂系统软件设计模型和测试模型的联系和差异，在此基础上建立设计模型到测试模型转换的映射规则，给出由映射规则和设计模型转换为测试模型的方法。建立映射时主要考虑结构、行为及约束三个方面的因素。

3）基于软件系统分析的测试建模：此类模型至少包括测试需求模型、测试环境模型、测试数据模型。根据分布式复杂系统软件测试目标，依据常规测试分析过程，对软件的功能、结构等逐层分解，形成测试原子模型。根据软件测试模型定义，将这些测试原子组合为各类测试模型。

4）基于测试模型转换的测试建模：此类模型至少包括测试用例模型。针对配置项/系统测试的测试用例设计要

求，研究测试用例生成策略。根据测试用例生成策略、测试需求模型，给出测试用例模型的建模方法，包括测试执行逻辑、测试数据、前置条件、预期输出的生成。给出生成测试用例模型的文本存储方法。

1.2.4　分布式复杂系统软件测试建模辅助工具研制与应用

对分布式复杂系统软件测试建模方法进行前景分析，开发分布式复杂系统软件测试建模辅助工具，包括测试需求建模工具、测试数据建模工具、测试环境建模工具、测试用例生成辅助工具。

2　测试建模技术基础研究

2.1　被测系统模型

被测系统模型是根据系统的需求、功能规格说明对系统进行建模，根据模型自动生成测试用例。其一般过程包括：建立模型、生成输入、生成期望输出。目前被测系统模型主要包括：

1）有限状态机模型。有限状态机（FSM）利用图的遍历算法，基于状态覆盖、转换覆盖、分支覆盖等覆盖准则，自动产生输入序列。扩展有限状态机（EFSM）增加了操作以及状态迁移的前置条件[1]，降低由纯 FSM 模型带来的状态爆炸问题，但增加了构造测试输入的复杂性。目前 FSM 缺乏自动构造的支持手段。

2）UML 模型。基于 UML 模型的测试主要集中于状态图和顺序图。状态图强调对复杂实时系统建模，提供了层次状态机框架，并提供并发机制描述。基于顺序图生成测试用例，通常是将其转化为流程图，采用基于路径的方法导出基本路径集，每一条路径即测试用例。UML 是一种形式化和半形式化相结合的建模语言，利用可视化建模元素可方便建立系统的图示模型，不同层次的模型可以重用。基于 UML 的软件测试成为研究热点[2]。

3）马尔可夫链模型。马尔可夫链是一种以统计理论为基础的统计模型。基于马尔可夫链的测试中比较有代表性的是基于使用剖面的可靠性测试用例自动生成方法。马尔可夫链实际上是一种有限状态机，只是在对应的迁移上标有概率特征。可以根据状态间的迁移概率自动产生测试用例，分析测试结果对可靠性指标进行度量[3]。

4）Petri 网模型。Petri 网易于表示系统变化发生的条件和变化发生后的系统状态，但不易表示系统中数据域或属性的具体变化或运算。Petri 网支持描述离散事件系统的动态行为，支持并发和异步性，既有严格的数学表述，又有直观的图形表达，在建模、性能分析以及测试中逐步得到应用。也有研究者提出一种 UML 状态图和 Petri 网结合的测试用例生成方法[4]，但是总体来说，该方法理论性较强，欠缺支持工具。

2.2　测试环境建模仿真技术

测试环境建模仿真指的是对被测软件的外围测试环境进行建模并仿真实现。测试环境建模方法有两种：一是直接程序设计即手写代码，二是可视化图形建模。采用直接程序设计方法要求建模者既要有较多的专业知识，又要有丰富的计算机编程经验。而采用可视化图形建模方法，则无需建模者具有较高编程水平，因其直观、快速、高效而受到日益普遍的重视。

测试环境通常与仿真工具相集成，如 Matlab/Simulink 仿真工具、Tech S. A. T 推出的第二代航电开发系统 ADS－2、北京航空航天大学开发的嵌入式通用仿真测试环境 GESTE、中船工业软件质量与可靠性测评中心研制的复杂系统软件自动化测试平台 SATP＋＋等。

测试环境建模仿真工具提供了丰富的模型库和可视化的建模环境，通常无需编程即可自动生成模型代码，具有通用化程度高、建模简单方便等优点。但缺点是通用仿真建模工具普遍价格昂贵，针对性较差，难以适用不同领域的专业知识，使其推广利用受到一定的限制。如果能充分利用各领域内的专业知识，针对特定的领域建立组件级的仿真测试环境模型，则可以进一步提高仿真模型的构建效率，并相应地提高仿真模型的再利用率。

2.3　测试执行驱动建模技术

测试执行驱动建模指的是按照测试用例步骤推进测试过程自动执行，加载测试输入数据，并收集实际输出数据。测试执行驱动模型一般分为两类：数据源驱动模型和数据加工驱动模型。

1）数据源驱动模型，在测试进行中，由测试环境加载已生成好的测试数据，按照测试数据的时间标签，调用 I/O 服务模块将其发送至被测系统。数据源驱动模型适用于规律非常复杂且不便于数据建模，以及计算非常消耗 CPU 资源的测试数据。通过在时间上将测试数据的产生与测试相互分离，降低了测试环境的计算强度，有效地节省了 CPU 资源。这类测试驱动方式的数据来源可以是长期积累下来的经验数据，也可以是由专门的工具生成的数据。但是它也有一定局限性，会浪费仿真测试环境的存储资源；不能动态的响应测试进行中的情况等。

2）数据加工驱动模型，在测试过程中由该模型进行运算并实时产生数据，然后调用 I/O 服务模块，将生成的测试数据发送到被测系统。由于该类模型需要根据外界提供的输入采取相应的计算来产生测试数据，就像一个数据加工厂，输入的是“数据原料”，产生的是“数据成品”。数据加工驱动有效的节省了测试环境的存储资源。在软件测试中，软件正确性无法根据一次单独的输出来判断，而是需要一组输出和几个时间段的输出的时间特性关系来判断，在该类系统中，在一个单独输出中的小“错误”或是偏离会在一段时间内被后面的输出补偿，无法确定一次独立的输出到底是否正确。因此就要求在能评估输出结果之前，测试运行必须持续相当长的一段的时间。数据加工驱动模型可以在线的生成大量的测试输入数据，而不需要在测试运行之前就生成大量的输入数据。

2.4　测试建模语言研究

2.4.1　U2TP 建模语言

自 2001 年起，OMG 扩展 UML 描述测试模型的领域无

关的标准建模语言，提出 UTP（UML Testing Profile）规范，规定了 UTP 的结构、内容、基于 MOF（Meta Object Facility）的元模型定义、用例及使用方法。但新版的 U2TP（UML 2.0 Testing Profile，2003 年发布）仍然没有完备的语义能够描述各种测试相关的概念。

U2TP 在 UML 元模型基础上添加了与测试相关的一系列概念定义，丰富了 UML 在测试领域的建模与应用。U2TP 主要提出以下 4 个部分的测试概念来描述测试的共同特征：测试架构（Test Architecture）、测试行为（Test Behavior）、测试数据（Test Data）和时间概念（Time Concepts）。

测试架构是对系统测试的整体环境搭建以及测试平台所包含的基本模块的静态描述[5]。包括：测试环境（Test context）、测试配置（Test configuration）、测试组件（Test Component）、被测系统（SUT）、裁决器（Arbiter）、调度器（Scheduler）、公用部分（Utility Part）。

测试行为定义的是测试平台执行测试过程的动态描述。包括：测试控制（Test control）、测试用例（Test Case）、测试调用、测试目标（Test objective）、激励（Stimulus）、观察（observation）、协作（Coordination）、缺省动作（Default）、判定（Verdict）、验证行为（Validation Action）、记录行为（Log Action）、测试记录（Test Log）。

测试数据定义了测试过程中用于测试组件与被测系统交互的数据相关的概念。在实际测试中测试数据往往根据被测系统的需求而确定，因此无法进行统一的定义。U2TP 只对测试过程中与测试数据相关的一些公有概念进行定义。包括：通配符（Wildcard）、数据池（DataPool）、数据分区（Data Partitions）、数据选择（Data selector）、编码规则（Coding Rule）。

为了能够对测试步骤执行进行时间管理与同步，需要对测试的操作与执行定义时间上的约束关系。测试环境通过量化时间来管理单个测试步骤的执行等待时间与判定。测试组件之间通过共同的时钟保证测试步骤执行的同步。时间概念包括定时器（Timer）、时区（Time Zone）。

2.4.2 TTCN－3

TTCN－3 是 ETSI（欧洲电信标准化组织）发布的针对测试的专用语言[6]。ETSI 发布的有关 TTCN－3 的标准包含一系列文档，此后 TTCN－3 发展迅速，得到了广泛应用，TTCN－3 标准也频繁的改进迭代，目前的最新版是 2009 年的 TTCN－3 标准 4.1。TTCN－3 指的是测试及测试控制表示法，灵活且功能强大，具备丰富类型系统等，但最大的不足之处是对于测试概念是基于文本的描述形式，难以理解，使用不便。

3 分布式复杂系统软件测试模型

3.1 分布式复杂系统软件测试模型要素分析

分布式复杂系统软件配置项/系统测试各种信息涉及的种类多，包括测试需求、测试环境、测试用例、测试数据等。各种测试信息逐层分解，又可包含更多的下层要素，这些要素之间存在一定的关联关系，且各要素功能、结构、行为上的特性存在多种表现方式。

测试环境包括测试组件（与被测系统交互的外围节点）、测试部署（测试组件之间以及测试组件和被测系统之间连接关系的集合）、测试调度器、测试裁决器等；测试部署包含各种交互接口及其连接关系，交互接口又包括以太网口、串口、CAN 口等。

测试用例包括测试执行逻辑（测试步骤及其关系）、测试数据、前置条件、预期输出；测试逻辑中又包括顺序、并发、反馈、周期执行方式等；测试数据包括单变量、单一类型数组、复杂类型数组等；单变量又包括多种类型。

可以说，测试信息是非常复杂、繁多的，现有的各类基于模型的相关测试技术一般是局部的某一方面的对测试的描述，或单纯测试用例，或单纯测试环境，另外也没有充分的考虑与开发过程模型的一致性。而测试模型是基于模型测试的核心，需要定义与开发过程模型相一致的、规范、完备的测试模型。

3.2 分布式复杂系统软件测试模型定义

测试模型需完整、规范描述测试信息；另一方面测试模型不能脱离开发过程，应与设计模型相一致，并且不能太复杂，可以工程应用，也有利于测试用例生成，本文提出的定义策略如下：

1）模型完整性依赖于软件测试信息分析是否完备。在分析软件测试信息时，以历史软件测试项目的数据作为基础，并参考软件测试项目数据，对测试信息进行自顶向下、层层细化的详细分析，并通过测试人员及开发人员不同的视角进行，从而得出各种分散的测试信息知识。对分析得出的分散知识点进行归纳、提炼，得到完备详细的测试信息分解体系，为抽象模型定义提供基础。分解分析主要包括以下方面：

（1）测试需求包括分解测试项、测试项输入、测试项输出、测试项充分性要求等；

（2）测试环境包括测试组件、测试部署、测试调度器、测试裁决器等；

（3）测试部署包含各种交互接口及其连接关系；

（4）交互接口，包括 CAN 接口、以太网接口、RS232 接口等；

（5）测试用例包括测试执行逻辑（测试步骤及其关系）、测试数据、前置条件、预期输出；

（6）测试执行逻辑中包括顺序、并发、反馈、周期执行方式等；

（7）测试数据包括数据类型、数据长度、数据范围等；

（8）数据类型包括单变量、单一类型数组、复杂类型数组等。

2）测试模型不能脱离开发过程，应与设计模型相一致。在获取完备详细的测试信息分解体系后，尽量参考设计模型建立测试原子模型、定义测试模型，使得测试模型语义规范、清晰、准确。在定义测试原子模型时，可以参考 UML 的模型元素、TTCN 测试和测试控制语言的类型元

素；在定义测试模型时，可参考各类 UML 设计模型（用例图、交互图、活动图等）、UML 2.0 Test Profile 测试剖面。

3）为了避免测试模型过于复杂，采取一定的简化措施。如对于测试数据模型，主要考虑数据字段的物理含义，而少考虑或不考虑字段长度、字段 MSB 等属性信息，减少了复杂性，也保证了测试用例生成时具有足够的信息可用。

4）可以考虑在测试模型中增加测试用例生成策略的相应的模型元素属性，以利于模型解析及测试用例生成[7]。如预置一定的设计策略，如等价类划分、边界值分析等，为以测试用例生成策略解析测试用例生成模型提供便利。

4　基于设计模型转换的测试建模方法

4.1　设计模型与测试模型的相似性分析

设计模型和测试模型都体现了系统需求，但是应用领域、应用目的不同，导致设计模型与测试模型之间存在很多联系和差异，分析其关联和差异是模型转换的必由之路。由于设计模型和测试模型的应用领域都具有庞大的体系，设计模型和测试模型本身也非常复杂，特别是针对分布式复杂系统软件这样庞大的系统，包含多个分系统，分析两种模型的关联和差异具有非常大的难度。采用的解决途径是分析设计模型与测试模型之间的区别和联系，并在此基础上实现从设计模型到测试模型的映射，从而构建软件测试模型。

4.1.1　设计模型与测试模型的相似点

设计模型和测试模型都体现了系统需求，建模都需考虑结构、行为及约束三个方面，因此设计模型和测试模型可以从这三个方面考虑其相似之处。

1）结构方面：设计模型中的结构图包括类图、对象图、构件图、包图等；测试模型中需求模型指对软件功能、性能等测试需求，可以体现在设计模型的类图上；测试模型中测试环境模型指与被测软件交互的类、对象、构件等组成的测试配置，这些测试配置可以体现到设计模型中类、对象、构件图上。

2）行为方面：设计模型使用活动图、交互图、状态图等描述系统的行为，这些行为可以体现在测试模型中测试用例模型上。

3）约束方面：测试模型的约束主要体现在测试数据模型上，测试数据模型可认为是设计模型中行为视图下的某个约束条件下的行为，测试数据是约束条件的等价划分。

4.1.2　设计模型与测试模型的异同点

设计模型与测试模型描述的是不同领域内的概念[8]。两者除了上述相似的概念外，测试模型还需对设计模型进行扩展来描述测试域，测试模型特有的概念有：

1）测试目标，测试目标是生产测试用例的依据，通常测试设计策略，如测试用例设计方法，以方法覆盖准则。

2）判定是对测试用例执行结果的判定，至少包括通过、测试结果与期望结果不一致两种测试结果。

4.2　设计模型到测试模型的映射

设计模型和测试模型之间存在关联，通过模型转换将设计模型转换为测试模型，可快速建立可复用、可扩展、可演化的测试模型。直接使用这些开发模型，转换为测试过程中所需要的测试需求、测试用例等，必定能大大提高测试的效率，实现全过程的自动化测试。

对于相似点可以直接从设计模型演化到测试模型；对于不同点则建立测试策略完善测试模型。从设计模型到测试模型的具体映射过程如下：

1）测试结构映射：设计模型中用例图、类图表示软件需求，将用例图、类图直接映射为测试模型中需求模型；设计模型与测试模型中被测件交互的包、构件、类映射到测试模型中的测试数据模型、测试环境模型。

2）测试行为映射：设计模型中状态图中的每个状态、每个迁移，活动图中的每个活动，交互图中的执行顺序等映射为测试目标，以及映射为测试用例模型中被测件与测试构件间的交互序列；设计模型中状态、活动、交互的前置条件映射测试用例中的前置条件，后置条件映射测试期望结果；根据测试执行中可能出现的结果与期望结果的比较，设置相应的判定值。

3）测试数据映射：设计模型中行为图的约束条件分解为数据划分，与测试行为关联，并根据不同测试策略映射为测试模型中测试数据模型。其中的测试策略如等价类划分、随机测试、边界值等。

5　应用前景分析

基于模型的全过程自动化测试技术可以实现较高的经济与社会效益，一方面针对单个项目可以实现全过程的自动化测试，大大提高测试效率；另一方面针对具有延续性的项目，可以实现测试资源复用，也间接提高了测试效率。

1）分布式复杂系统软件研制过程中模型辅助开发逐步广泛应用，基于模型的全过程自动化测试可以直接复用设计模型，转换为测试模型，进而利用自动化手段生成测试用例、仿真测试环境、自动驱动测试执行。提高了测试自动化程度，也使配置项及系统测试可以尽早介入，帮助设计人员较早的发现设计中的缺陷以及模型本身的问题，提高系统开发与测试的效率。

2）单纯的文档形式的测试需求、测试用例集与测试数据可读性不强，无法帮助测试人员很好的理解被测系统设计，而测试模型可以提供清晰准确的系统设计，帮助测试人员了解被测系统，将时间花在重要的测试用例设计环节，而非湮没在手工生成大量的测试用例工作中，提高测试的质量与效率[9]。

3）测试模型源自于设计模型，当软件功能发生变化需要调整设计模型时（这种情况在分布式复杂系统软件开发中经常出现），测试模型也能随之进行相应的修改来适应新的测试需求。因此在测试过程中，只需维护测试模型，而无需直接维护测试需求及大量的测试用例，减少了维护工作量。

4）采用基于模型的全过程自动化测试方法形成可复用、组件化的测试模型（尤其是面向特定应用领域的专有

测试模型），以及相应的测试工具集，这也为在后续型号软件测试中复用前期测试成果提供了可能性和手段。

另外，对于软件测试来说，长远目标则是实现测试组件标准化、通用化、型谱化。针对分布式复杂系统软件，由于其应用特点，很多采用了面向对象程序语言进行设计开发，呈现出构件化、分布式、跨平台等特点，为解决跨平台的通信、计算等问题，中间件技术逐步被广泛应用。为了实现不同中间件、平台之间的集成和互操作，模型辅助开发模式被逐步广泛采用。针对这种模式开发的软件，也十分适合采用基于模型的全过程自动化测试技术。因此，分布式复杂系统软件测建模方法在软件开发机构及测试机构中都具广阔的应用前景。

6 小结

分布式复杂系统软件测试周期短、测试质量要求高，自动化测试及测试资源复用需求迫切。本文研究分布式复杂系统软件测试模型，给出适合于分布式复杂系统软件测试描述的各类软件测试模型及建模方法，包括对测试需求、测试用例、测试环境、测试数据等的描述，为自动化测试、提高测试效率提供方法和手段。针对分布式复杂系统软件，相对于传统的软件测试方法，本文提出基于模型的全过程自动化测试方法，能显著的提高配置项及系统测试的效率，提高测试质量，进而保证型号软件质量，具有重大的经济与社会效益。

参考文献：

[1] 董焕珍，等. 扩展有限状态机到场景的转化 [A]. 全国计算机新科技与计算机教育学术大会 [C]. 2008.

[2] 刘青香. 基于 UML 交互概览图的测试方法研究 [D]. 重庆：重庆大学，2013.

[3] 陈丽敏. 基于马尔可夫链模型的软件可靠性测试方法研究 [D]. 电子科技大学，2010.

[4] 崔尚森，孙 琳. 基于 UML 状态图和基本 Petri 网生成测试用例 [J]. 交通信息与安全，2006，24 (4)：116－119.

[5] 邓璐娟，李金萌，董东晓. 自动化测试框架技术及应用 [J]. 计算机测量与控制，2016，24 (9)：86－88.

[6] 王文庆，张圣琨. 基于 TTCN－3 的 LISP 协议一致性测试 [J]. 西安邮电大学学报，2017，22 (4)：109－113.

[7] 谢 林，杨 扬. 基于模型的进路建立过程测试用例自动生成 [J]. 铁道标准设计，2017，61 (2)：109－116.

[8] 刘冬懿，金茂忠，刘 超，等. 从 UML 设计模型到测试模型的研究 [J]. 计算机应用研究，2007，24 (5)：56－59.

[9] 孙大成. 基于 UML 时序图的测试用例自动生成系统的设计与实现 [D]. 北京：北京工业大学，2017.

(上接第 117 页)

表 2 窗口长度为 N 的不同窗函数性能对比

故障现象	维修训练过程(步骤)	原因
开机后电源指示灯亮，显示器不亮，系统不启动	(1)加电； (2)检查 EMC 模块的输入、输出电压； (3)若 EMC 模块输入电压小于 20 V 则输入电瓶电压不够。否则， (4)若 EMC 模块输出电压小于 18 V，则 EMC 模块故障。否则， (5)检查与开关相连的电缆线端主板电源插座是否有 24 V 电压，没有电压则断线。否则， (6)检查电源适配器的输出电压是否 24 V，没有电压则电源适配器故障。否则， (7)检查与电源适配器相连的主板电源插座是否有 24 V 电压，没有电压则断线。否则， (8)检查电源转换模块的输入是否有 24 V，没有电压则断线。输出是否 12 V 左右，没有电压或低于 10 V，或者高于 15 V，则电源转换模块故障。否则， (9)如果主板电源插座电压正常，则主板上的其他电源模块故障。	(1)电瓶电压不够； (2)EMC 模块故障； (3)电缆断线； (4)电源适配器故障； (5)电源转换模块故障； (6)主板故障。

2）该仿真器具有故障设置和故障撤销功能，便于故障再现，维修人员可以在教员的指导下进行维修训练，也可进行自学，大大提高了维修人员的实际技能。

3）利用仿真器进行维修训练，无安全风险，价格低，节省了维修训练费用，为装备维修训练提供了很大的便利，便于推广应用。

参考文献：

[1] 于永利，陶凤和. 复杂装备多媒体维修训练系统的设计 [J]. 计算机工程，2000，26 (4)：83－84.

[2] 谭继帅，郝建平. 浅析当前装备维修训练的发展趋势 [J]. 设备管理与维修，2007，34 (11)：11－12.

[3] 陈建明，刘军辉，丑 力. 某型战车驾驶员任务终端检测仪的设计 [J]. 计算机测量与控制，2010，18 (12)：2792－2794

[4] 胡文华，赵 喜，段修生，等. 某型火控雷达维修训练模拟器设计与应用 [J]. 计算机测量与控制，2016，24 (9)：143－145.

[5] 范志良，刘光斌. 多通道 GPS 卫星信号仿真器设计与实现 [J]. 计算机工程与应用，2009，45 (18)：78－80.

[6] 张 磊，冀海燕，卢文忠. 模拟器设计与维修训练应用仿真研究 [J]. 现代防御技术，2011，39 (1)：153－156.

[7] 张 柳. 装备作战单元维修保障建模与仿真 [M]. 北京：国防工业出版社，2016.

[8] 矫永康，李小民，毛 琼. 改进模糊层次分析法在虚拟维修训练评估中的应用 [J]. 计算机工程，2014，40 (10)：314－317.

[9] 钟京立，张 辉，马 侦. 电子信息装备维修保障能力的 BP 神经网络评估 [J]. 现代电子技术，2015，38 (2)：11－14.

[10] 陈建明，王洪艳，金传洋. 基于实物仿真的装备维修训练系统设计 [J]. 设备管理与维修，2014，41 (2)：15－17.