形式化的软件测试模型研究*)

赖祥伟　张为群　邱玉辉　周彦晖
(西南师范大学计算机与信息科学学院　重庆400715)

摘　要　传统方法中基于软件测试工程师经验的测试用例构造技术使得软件测试的质量很难得到保证。本文提出一种基于形式化方法的软件测试模型。该模型使用形式化描述语言RSL对软件设计进行描述，并在此基础上提出了自动生成软件测试各个阶段所需测试用例的方法框架以及相关实验结论。

关键词　软件测试模型，形式化方法，断言测试，UML，RSL

Software Testing Model Based on Formal Method

LAI Xiang-Wei　ZHANG Wei-Qun　QIU Yu-Hui　ZHOU Yan-Hui
(Faculty of Computer and Information Science, South West Normal University, Chongqing 400715)

Abstract　Software testing is an important method to keep the quality of software. The traditional method designs testing case based on developer's experience. It makes us difficult to keep the dependability and integrality of software. In the paper, we design a new software testing model based on formal method. In the method we get formal description of the software design using RSL, and give an automatic method to get testing case used to every phase of software develop by these formal description. And there is a demo case in the end.

Keywords　Software testing model, Formal method, Assert testing, UML, RSL

1　引言

随着软件技术的不断发展，开发者和用户对软件质量提出了更高的要求。为此，软件开发者试图从技术、管理等各层面控制软件开发过程，提高软件产品的针对性和可靠性，保证软件对于用户的使用价值。

在现有的技术中，开发者用于保证软件开发质量的主要手段可以归纳为以下五种：1)采用更加科学实用的软件开发过程模型；2)使用高效的软件设计和开发方法；3)使用基于严格数理逻辑的形式化方法指导软件开发过程；4)使用面向不同软件抽象层次的软件测试策略；5)在软件开发过程中应用适当的项目管理技术。在实践中，通过这些方法的综合应用，可有效地提高软件产品开发的效率和成功率，保证软件质量。在众多软件质量保证技术中，软件测试作为一种传统的、直接的、行之有效的方法在软件质量保证中起到了决定性的作用，其具体技术已被融入各种软件开发过程和方法中。

2　传统软件测试方法及技术缺陷

如图1所示，在传统的开发过程中，设计者、实现者和测试者一般通过几种手段对软件的设计和实现进行测试，以保证最终软件的质量。其中常用的方法有以下三种：

1)从设计者的角度，需要对软件的设计进行技术复审，以保证软件的设计符合用户的需求，同时也能够为开发人员所实现。技术复审需要审查软件设计各个阶段所进行的设计以及设计所产生的文档。

2)从实现者的角度，需要在编程过程中对代码进行检测，除了使用相应的调试技术和专用的代码检查工具之外，测试断言也是一种很好的方法。

3)从测试者的角度，需要对软件进行包括单元测试、集成测试、确认测试等一系列全面的测试，以保证最终软件符合用户需要，同时尽可能发现和修改程序中可能存在的错误和缺陷。

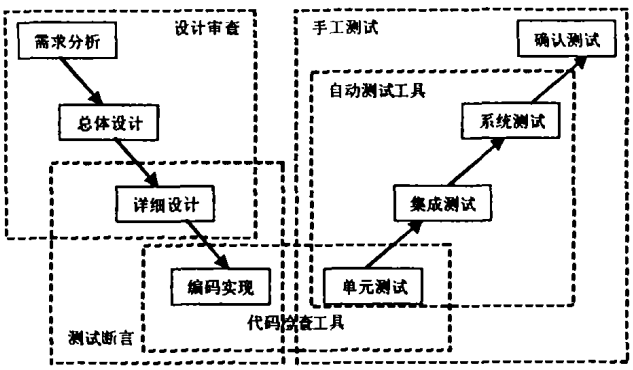

图1　传统的测试流程图

传统的测试技术提供了一系列具体的测试技术帮助软件设计开发人员在软件开发的各个阶段对软件的正确性和可靠性进行检查和测试，但是其中也存在这一些不可避免的问题和瑕疵。这些问题主要表现在：

1)针对软件需求分析和设计过程中的技术复审以及后期的集成测试和确认测试，没有一种可靠的方法进行验证。

2)软件测试过于依赖开发测试人员的个人素质，测试用例多数由软件测试人员手工构造，测试效率低下，在大规模的

*)本文为教育部和重庆市"软件工程可视形式化"项目部分工作。赖祥伟　博士，主要研究领域为软件形式化方法、软件测试及软件Agent。张为群　教授，主要研究领域为软件工程及人工智能技术。邱玉辉　教授，博士生导师，主要研究领域为人工智能技术。周彦晖　讲师，主要研究领域为软件工程及形式化方法。

软件系统开发过程中测试的完整性只有依赖于测试配置管理工作来完成。

3)测试工作量巨大,开发者为了保证软件的质量不得不投入与设计成本相差无几的测试人员以保证软件测试的尽量全面。

4)缺乏对测试效果的有效评估机制。

3 形式化的软件测试模型

本文所提出的形式化测试模型是在现有软件测试模型基础上利用形式化方法对当前业界广泛使用的软件设计工具UML进行形式化描述和规约,并使用相应的方法提取测试用例辅助传统软件测试过程的具体实施模型。

3.1 UML图的形式化描述与规约

UML是一种图形化的设计语言,它为使用者提供了丰富的图形表示方式,使得使用者可以从不同的抽象角度对软件系统的特征进行描述。在使用UML进行软件设计建模时,设计者必须根据所需要描述的对象和系统动作选择适当的UML图,以达到设计效果[7]。同样在软件测试过程中,在不同的测试阶段,根据不同的测试目的,必须选择不同的UML图。

在UML强大的描述能力背后也不可避免地存在着诸如设计随意性强、缺乏严格设计语义等缺点。如果测试者使用设计过程中得到的UML图形直接设计和组织测试用例,这同传统的方法并没有本质的区别。在研究过程中,我们提出了使用形式化描述语言RSL对基本的UML图形进行形式化描述的具体方法[4,6,11]。使用对应方法能够将图形化的UML设计转化为RSL形式化描述。可以证明转化得到的RSL描述对于UML图形具有完整的描述能力[6],由此这些形式化的描述不仅是需求验证的重要工具,同时也是测试用例的重要来源。

3.2 形式化的测试用例生产技术

形式化的软件测试技术以传统的软件测试为基础,在原有测试流程的基础上,利用形式化方法精确的描述能力和严格的推理过程,使计算机能够自动完成传统测试流程中需要测试者手工完成的工作。该模型的工作原理及具体流程如图2所示。

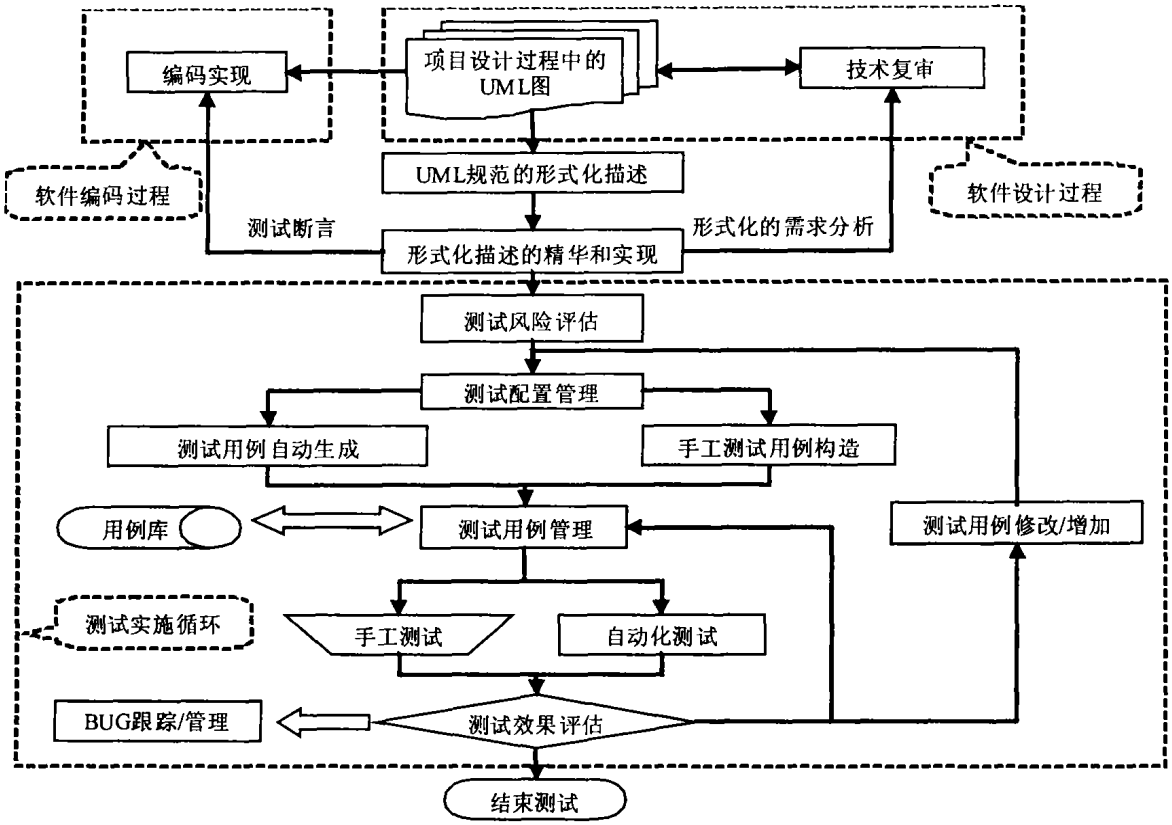

图2 形式化的软件测试模型

3.2.1 *形式化的技术复审* 技术复审过程中,使用形式化方法对用户需求进行描述,能够帮助设计者更加清晰地理解和把握客户的需求,从而保证设计与需求的一致性。这些形式化的描述也是设计者进行技术复审的重要依据之一,同时可以利用RSL自身的验证能力对产生的形式化需求进行验证,以保证设计过程的可靠性[5,11]。

3.2.2 *形式化的断言测试用例生产技术* 断言测试是当前软件测试过程中常用的测试方法之一,其主要优点在于将测试代码嵌入到程序实现代码中,在程序运行过程中对关键量的约束做实时检查,从而降低了测试工作量,并保证测试的正确运行。程序开发人员可以利用一定的方法从UML类图的RSL形式化描述中提取用于断言测试的测试用例,从而减低了手工设计测试用例的工作量,增加了测试的可靠性。由于形式化描述的良好数学特性,这些测试断言的产生是可以由软件系统自动完成的[1]。

3.2.3 *软件集成测试和确认测试过程中的测试用例生成* 在软件测试过程中,形式化方法的作用主要在于通过对系统设计的形式化描述以及形式化方法自身的精化、规约能力准确地描述系统设计,代替测试人员按照一定的规则生成软件测试各阶段所需的测试用例。虽然现有的方法还无法生成软件测试所需要的所有测试用例,但是这些方法确实可以大大减轻测试人员的工作量,提高测试的效率和覆盖率[2,10]。

3.2.4 *测试用例的管理与复用* 通过对于测试用例的形式化处理,能够更加方便地实现测试用例的管理,进而实现

测试用例的复用。目前具体的复用规则还在研究过程中。

4 形式化测试方法与传统测试方法的协调使用

传统的测试技术与形式化的软件测试技术并非彼此对立，相反它们之间有着相当紧密的关系，形式化的测试技术以传统测试技术为基础，同时也是传统测试技术的有益补充。两者在测试过程中的协调使用有利于提高测试的效率和覆盖率。

前面所提到的针对断言测试和集成、确认测试的软件测试技术所产生的测试用例并不是软件系统测试所产生的测试用例的总和。其作用主要在于通过一些行之有效的方法能够自动地从软件设计规范中提取与测试相关的元素，自动生成部分测试用例，从而大大降低原来需要由手工构造测试用例而导致的大量工作。采用上述方法后，可以把类中的边界值测试和系统测试中所有正常输入的黑盒测试等部分测试完全使用形式化方法代替，同时可以减少类中路径覆盖测试和系统测试中非正常输入的黑盒测试的用例数量，把主要的放在弥补形式化测试的遗漏上。同时形式化的软件需求检查也是设计评审中重要的工具之一。

规模化的软件测试不仅要求测试者注重测试实体的具体测试技术，更强调对于整个测试流程的管理和配置。软件测试是一项系统工程，没有宏观的控制就不可能取得满意的测试效果。形式化的软件测试技术弥补了传统测试技术中过分依赖测试者个体能力的弊病，使得测试用例的生成和使用有了系统的、准确的标准，相对于传统的测试技术而言有其不可替代的优越性。另一方面，传统的测试技术(基于测试者个人素质构建测试用例的技术)也有其优点，其中许多优点是自动化的测试技术所不可代替的。我们可以很容易地比较出一个经验丰富的测试者与一个成熟的自动化测试系统各自的优缺点。

表1 传统软件测试技术与形式化软件测试技术

	传统测试技术	形式化测试技术
测试用例构造	依赖测试者个人素质构造	依据系统规范自动构造
测试用例数量	少	多
测试实施	手工	自动
测试时间	短	相对较长
测试覆盖率	低	高
测试针对性	能够根据测试经验和测试实施情况主动调整测试过程，对于难点和有问题部件作重点测试	缺乏对特殊情况的适应性
测试成本	高	低
应用范围	所有测试过程均可使用	部分测试过程
测试对象	所有软件	部分软件

通过以上的比较可以发现，传统的软件测试技术与相似的软件测试技术有着各自不同的优缺点，重要的是这些优缺点很多都是互补的。两者的配合使用往往能够起到取长补短的效果。

5 形式化测试模型的应用实例

为了验证形式化软件测试模型的实用性，我们对一个典型的电梯控制系统软件开发过程进行了实际验证。

5.1 系统的技术复审

设计得到对应的 UML 图形后，使用 RSL 语言对 UML 图形进行形式化的描述。首先可以通过 RSL 自身的验证能力在设计阶段对设计结果进行正确性审查。在审查通过后，对 RSL 描述进一步进行精化，得到更加完整的系统形式化描述。

5.2 测试断言生成实例

系统的对象和它们之间的关系可以用类图3表示。

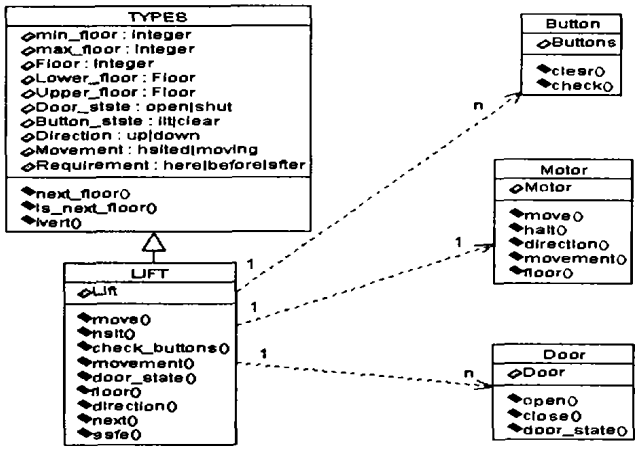

图3 Lift 类图

其相应对象的形式化描述为：

```
scheme DOOR =
  class
    type Doors
    value
      open:T. Floor×Doors→Doors,
      close:T. Floor×Doors→Doors,
      door-state:Door→T. Floor→T. Door-state
    axiom
      [door-state-open]
        ∀ f1,f2:T. Floor,s:Door·
          door-state(open(f1,s))(f2)≡
          if f1=f2 then T. open else door-state(s)(f2) end,
        [door-state-close]
          ∀ f1,f2:T. Floor,s:Door·
          door-state(close(f1,s))(f2)≡
          if f1=f2 then T. shut else door-state(s)(f2) end
end
```

在对楼门的描述中，只有两个动作：*open* 和 *close*，以及一个楼门状态的描述量：*door_state*。在对应的描述中给出了两个相关性条件：*door_state_open* 和 *door_state_close*。根据测试用例的生成规则可以对应地产生四个测试用例：

(1) $f1=f2 \rightarrow door\_state(open(f1,s))(f2)\equiv T.open$

(2) $f1\neq f2 \rightarrow door\_state(open(f1,s))(f2)\equiv door\_state(s)(f2)$

(3) $f1=f2 \rightarrow door\_state(close\ (f1,s))(f2)\equiv T.shut$

(4) $f1\neq f2 \rightarrow door\_state(close\ (f1,s))(f2)\equiv door\_state(s)(f2)$

这四个用例保证了在楼门的 *open* 和 *close* 操作必须遵守这样的规则：在同一时刻只有一个楼门可以打开，而且打开的楼门必须是电梯所在楼层的楼门。对于一个电梯系统而言，这个规则是必须遵守的，所以测试者可以在任何楼门的 *open* 和 *close* 操作执行时使用这两个测试用例对系统进行相关性条件检查，以保证系统的安全运行。同样，使用系统的前置条件和后置条件组合可以生成其它7个 DOOR 对象的测试用例。这些测试用例将被直接转化为测试断言写入系统代码中。使用同样的方法可以生成其它静态对象的测试断言。

5.3 集成、确认测试用例的生成

电梯是一个不断运行的动态系统，使用状态图能够很好地描述系统的动作变化和相应的控制条件。在电梯系统设计

中，设计者需要考虑许多动态关系。在这里以最为典型的箱体动作状态图为例说明利用消息序列方法对系统进行测试的方法。其状态图如图4所示。

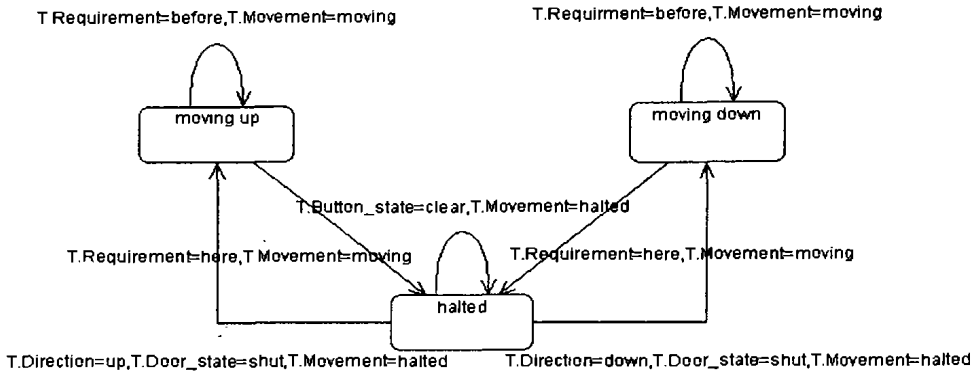

图4 Lift 运行状态图

在状态图中，电梯共有停止、向上运行和向下运行三种状态。各种状态之间通过一定的关系相互转换。严格地说，状态之间转换的条件在编码实现时并不是使用消息机制来完成的。但是在生成测试用例时，同样可以把状态间的转换条件作为一种特殊的消息来处理，这样有利于简化测试用例的生成方法。对于系统的状态图可以从中抽象出下列的消息描述：

```
scheme MESSAGE =
  class
    type
      object,oper,para,
      message(dispatcher: object,destination: object,operation: oper,parameters: {| ps: para-set • (∀ p: para⇒ p∈ para-set)})
value
      m-up-up(moving-up, moving-up, Null, T.Requirement = before∪T.Movement=moving),
      m-up-ha(moving-up,halted,NULL,T.Requirement=here ∪T.Movement=moving),
      m-ha-up (halted, moving-up, NULL, T.Direction = upT.Door-state=shut∪T.Movement=halted),
      m-ha-ha(halted, halted, NULL, T.Button-state = clear ∪ T.Movement=halted),
      m-down-down (moving-down, moving-down, NULL, T.Requirment=before,T.Movement=moving),
      m-down-ha(moving-down, halted, NULL, Requirement = here∪T.Movement=moving),
      m-ha-down(halted, moving-down, NULL, T.Direction = downT.Door-state=shut∪T.Movement=halted),
end
```

根据基于消息序列的测试用例提取规则，可以产生电梯动作的测试用例，如表2。这些测试用例都会被用在系统的集成测试和确认测试中。其它测试用例的产生也按照类似方法进行。

表2 lift 状态图的测试用例输入和预期结果

测试输入	预期结果
T. Movement = moving, T. Requirement = before	moving-up
T. Movement = halted, T. Requirement = here,	halted
T. Movement = halted, T. Direction = up, T. Door-state=shut	moving-up
T. Movement=halted, T. Button-state=clear	halted
T. Requirment = before, T. Movement = moving	Moving-down
T. Movement=moving, Requirement=here	halted

5.4 测试结果分析

在使用形式化的测试模型对软件进行测试的同时，我们使用传统的方法对软件系统进行了各个阶段的测试。在使用同样人力、物力资源的情况下，两种测试在不同阶段消耗的时间比较如图3，而两种模型所产生的测试用例数量比较如图4所示。

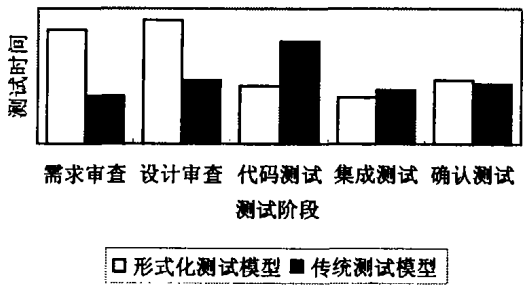

图3 测试时间比较图

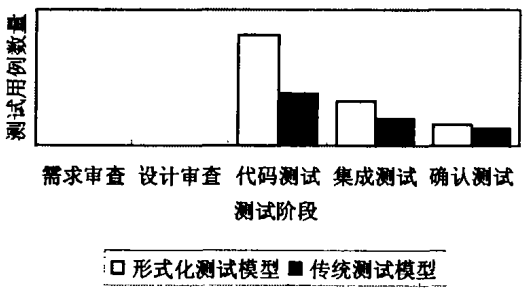

图4 测试用例数量比较图

测试时间分析：由图3、4可以发现形式化的软件测试技术相对与传统的软件测试技术而言，由于加入了对于设计需求的形式化描述工作，因此在需求审查和设计审查阶段需要投入更多的成本。在整个系统的测试过程中，形式化测试模型于传统测试模型所消耗的时间比例为1.32:1。应该看到，随着软件规模的增大，形式化模型时间上的缺陷相对于巨大的开发时间而言将是微不足道的。

测试完整性分析：由于测试用例的生成不再完全依赖测试者手工构造，从而大大提高了测试用例的生成数量(这些用例不是重复的)，使得测试更加全面可靠。

测试成本分析：形式化的测试方法能够有效地降低测试人员手工构造测试用例的数量和复杂程度，从而降低测试的平均成本。就具体测试对象而言，形式化模型与传统模型相比其单个用例的成本比例为1:1.43。

可见，相对于传统测试模型而言，形式化的软件测试模型有着严格、全面、自动化程度高等一系列优点。提供一体化的

创建油井采油过程计算机远程监控系统的研究*)

王建华[1] 张 军[1] 俞兰芳[2]

(哈尔滨师范大学计算机科学系 哈尔滨150025)[1] (哈尔滨市职工医学院 哈尔滨150080)[2]

摘 要 本文首先简要介绍了计算机远程监控系统及电力线数字通讯技术,然后阐明了创建油井采油过程计算机监控系统的主要功能和作用。

关键词 计算机远程监控系统,电力线数字通讯技术

The Study on Establishing Computer Access System of the Process of Excauating Oil from Oil Well

WANG Jian-Hua[1] ZHANG Jun[1] YU Lan-Fang[2]

(Department of Computer Science,Haerbin Normal Univesity,Haerbin150025)[1]

(Haerbin Staff and Workers Hospital,Haerbin150080)[2]

Abstract This ariticle concisely introduces remote computer access,systemand digital power line communications technologies. Then clarifies the main functions and uses of establishing remote computer access system of the process of the process of excavating oil from oil well.

Keywords Remote computer access system,Digital power liner communications technologies

1 引言

计算机远程监控系统,传统上也叫"三遥"或"四遥"系统,当前也称为分布式数据采集系统(SCADA 系统),叫法虽不同,但工作原理及组织结构大致相同,一般采用无线数据通讯方式或无线有线相结合的方式。它是仪器仪表、电子技术、现代通讯技术、计算机软件等的综合运用。计算机远程监控系统可以极大地提高人们的生产自动化水平和生产效率,已经被广泛应用于许多行业和领域。

2 电力线跨(变压器)台区工频传输技术简介

人类在发现、使用和推广电能的过程中,建立了庞大的、遍及全球的物理连接网络。人们期望能将现有的庞大的电力网作为信息通讯的物理网络,将电力输送网和通讯网合二为一,经济地实现"智能大厦"、"智能小区"。由于在电力系统中使用坚固可靠的电力线作为信号的传输媒介,可节省大量的通道建设投资,再加上信息传输稳定可靠,安全保密以及能够同时复用远动信号等特点,使得这种电力系统独有的通信方式在数字微波、一点多址、光纤、甚高频等通信方式相继出现的今天仍得到持续的发展。因此,电力网被国外传媒喻为"未被挖掘的金山"毫不为过。

该系统以电力配电网为介质,能够利用现有配电网实现无中继、无桥接设备的跨变压器台区在不同电压等级之间的自动抄表,是一种不同于传统电力载波抄表的新型系统。如图1所示,系统由位于二次变电所的主站与位于用户端采集模块组成,系统完全以10kV 配电线路及220V 低压入户线路为信息传输媒介,在用户变压器附近无需增加附属设备,可实现配用电信息的配电网传输。各个二次变电所的主站通过电力系统的网络与管理服务器相连,在管理中心的客户终端可以对用户端采集模块进行访问、控制、管理。

信号的发送采用电压过零调制的办法,在电压过零点附近可以用较小的调制功率实现信号的叠加,同时电压过零点自然提供了数据传输过程中信号检测的同步。信号的检测采用差分接收技术可以从电网大噪声背景中将微弱调制信号检测出来。

调制信号分为下行电压信号与上行电流信号。下行电压信号传输方向从主站到采集模块,代表的是命令信息,以电压过零点附近电压的微弱畸变来实现信息表示,下行电压信号

*)黑龙江省教育厅科研资助项目(10511020),黑龙江省科技攻关计划资助项目(GC02A133)。

测试工具,克服其技术难度较大、推广相对较难这一问题将是我们下一个阶段研究的主要方向。

参 考 文 献

1 赖祥伟.基于 UML 的形式化面向对象软件测试用例辅助生产技术:[硕士论文]. 2003
2 赖祥伟,张为群.基于消息序列的形式化面向对象软件测试技术.计算机科学,2002,29(10)
3 Brooks F. No Silver Bullet: Essence and Accidents of Software Engineering. IEEE Computer,20(4)
4 周彦晖,张为群.软件形式化与可视化模型的转换.计算机科学,2003,30(7)
5 周彦晖,张为群.形式化与可视化结合的 FDOOM 软件开发方法.计算机科学,2003,30(9)
6 The RAISE Method Group: George C,et al. The RAISE Development Method. TERMA Electronic AS,Denmark,1999
7 Rumbaugh B G,Jacobsen J. The Unified Modeling Language User Guide. Addison-Wesley,1999
8 Binder R V. Testing Object-Oriented System: Models Patterns and Tools. Addison-Wesley,2000
9 McGregor J D,Sykes D A. A Practical Guide to Testing Object-Oriented Software. Addison-Wesley,2001
10 李留英,王戟,齐治昌. UML statecharts 的测试用例生成方法.计算机研究与发展,2001,38(6)
11 罗密.面向对象系统需求说明的形式化规范:[硕士论文]. 2003
12 Jacobson J. Object-Oriented Software Engineering, A Use Case Driven Approach. Addison-Wesley,1992