文章编号:1001 - 9081(2006)03 - 0682 - 03

基于设计模型的信息系统自动测试方法

马 昕,顾 明
(清华大学 软件学院,北京 100084)
(h_m_friend@126.com)

摘 要:针对当前自动测试领域存在的问题,提出了一种基于设计模型的自动测试方法(Model Based Automated TestIng System,MATIS)。该方法利用用户界面自动生成方法,把设计模型中的类属性定义和实现中的控件属性组织在一起,构建描述界面的逻辑对照表,辅助测试脚本引擎执行自动测试脚本。借助设计模型中扩展的类定义,MATIS 方法可以自动生成测试用例和测试数据。MATIS 方法是一个较轻量级的方法,更贴近于实际的软件开发过程,可以有效地降低自动测试成本。

关键词:自动测试;信息系统;设计模型
中图分类号:TP311.52 **文献标识码**:A

Automatic testing method of information systems based on design model

MA Xin, GU Ming
(*School of Software, Tsinghua University, Beijing* 100084, *China*)

Abstract: MATIS, a model based automated testing system was put forward, reducing requirements during information systems test automation. MATIS changed the way how the test scripts scripted by using a logical property table constructed by user interface automatic generation method. An extended design specification produced in software design process was used to automate test generation and execution. This method is suitable for the development of information systems.

Key words: automated test; information systems; design model

0 引言

基于 GUI 构建的信息系统在社会日常生活中发挥着重要作用,它能处理并储存多种类型的数据,协助使用者分析信息并做出决策。信息系统一旦发生故障就会造成严重的后果。如何更好地保障信息系统的稳定性是软件测试工作面对的一个重要问题。

信息系统功能多,结构复杂,采用手工测试费时费力也难以保证质量。使用自动测试方法可以部分解决手工测试的问题,但自动测试目前还存在一些问题[1]:1)前期投入较大,需要建立专门的测试小组,花费较长的时间准备测试用例和脚本;2)难以有效地收集 GUI 用户界面信息并自动生成测试脚本;3)在完成系统实现(或部分实现)后才能开展测试工作。

针对自动测试目前存在的问题,研究人员提出了基于设计模型开展自动测试的方法。在系统设计阶段,开发人员按规范在设计模型中添加支持自动测试的信息,测试时使用分析工具提取这些测试支持信息,自动生成测试方案和测试用例。这种方法能有效地提高测试自动化的程度,缩短测试准备周期。

本文介绍一种基于设计模型的信息系统自动测试方法(Model Based Automated TestIng System,MATIS),利用描述用户界面的信息隔离系统设计与实现细节,自动构造界面导航脚本并生成测试数据。在保持系统设计独立性的前提下,MATIS 方法利用系统设计中包含的测试支持信息,有效地提高了自动测试的效率和灵活性。MATIS 方法不依赖于特定的设计语言和设计平台,为方便交流,本文使用 UML 来描述相关的系统设计。MATIS 方法有一个已实现的同名的原型系统,本文后面描述中单独使用的 MATIS 单词指该原型系统。

1 MATIS 方法概述

每种基于设计模型的自动测试方法都有自己的侧重点,使用设计模型中的不同部分,支持不同类型的自动测试。AGEDIS[2]项目给出了一种基于 UML 状态图的自动测试方法,主要支持功能测试。MODEST[3] 为基于 UP(Unified Process)开发的信息系统提供了支持自动测试的设计标准,涉及到用例图、类图、序列图和状态图等多种 UML 图。该方法使用多种特定版型(Stereotype)定义测试支持信息,可以支持功能测试、性能测试等多种自动测试类型。这些方法都有一系列的支持工具,用来自动生成和执行测试用例。

但目前的基于设计模型的自动测试方法也有一些不足:

1)过于依赖设计模型;

2)设计模型增加了大量严格的设计规范,给系统设计和变更管理带来了额外的负担;

3)定义测试支持信息时通常要用到系统实现细节,如定义边界类成员变量时需要说明其具体实现类型(TextField 等),这破坏了设计模型的完整性和抽象性;

4)没有为信息系统提供优化的支持。

与其他方法相比,MATIS 方法是一个较轻量级的方法,更贴近实际的软件开发过程,更容易管理系统变更。借助在类图中添加的少量测试支持信息,MATIS 方法可以完成自动的功能性单元测试和集成测试。针对现存问题,MATIS 方法有如下改进:

1)用到的设计模型信息较少,对设计模型的依赖性低;

2)使用用户界面自动生成方法构造 GUI 界面,生成界面

收稿日期:2005 - 09 - 26 **基金项目**:国家 863 计划项目(2003AA414030)
作者简介:马昕(1977 -),男,河南郑州人,硕士研究生,主要研究方向:产生式编程、自动化测试; 顾明(1962 -),女,辽宁沈阳人,副教授,主要研究方向:操作系统、分布式应用系统支撑平台、电子商务.

描述信息,这种方法更贴近软件的实际开发过程,也能有效地从设计模型中分离实现细节;

3)基于 GUI 构建,包含大量的增删改读等数据操作,MATIS 方法根据信息系统这两个主要特点设计了测试过程和测试支持信息,为信息系统测试提供了优化的支持;

4)同时支持自动构造测试脚本和手工编写测试脚本,具有较高的灵活性。

图 1 使用活动图的方式展示了 MATIS 方法的几个主要环节。

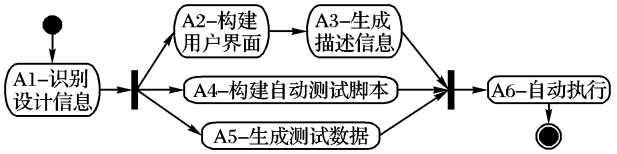

图 1　MATIS 方法各环节

A1 是 MATIS 方法的基础活动。依靠 A1 抽取的信息,A2 和 A3 这两个活动把系统设计中的实体定义和界面定义同系统实现细节联系在一起,生成用户界面描述信息。A4 和 A5 活动可以自动执行,生成测试脚本和测试数据。A2 - A3,A4 和 A5 等活动基于设计模型中不同的部分,这三个部分可以独立执行。A1 ~ A5 等活动共同构成了 MATIS 方法的准备阶段,在脚本和数据准备就绪后,A6 完成测试用例的自动执行。

2　MATIS 方法实现过程

MATIS 方法使用用户界面自动生成方法,把设计中的类属性定义和实现中的控件属性组织在一起,构建描述用户界面的逻辑对照表,辅助脚本引擎执行自动测试脚本。根据类图中的测试支持信息,MATIS 方法还可以自动构造测试脚本和测试数据。

2.1　识别设计信息

识别设计信息是 MATIS 方法的第一步,在执行该活动之前,要求系统设计模型按要求定义格式化的测试支持信息。

在使用 UML 描述的系统设计中,实体类通常用来描述持久层信息,对信息系统而言,实体类定义了数据库表结构。为了给测试提供支持,实体类成员变量中除了字段名、数据类型、字段长度等定义外,还需要定义别名、最大值、最小值、初始值、可选项列表等测试支持信息。别名作为类成员变量的助记符,主要目的是提高测试脚本的可读性。对没有定义别名的成员变量,MATIS 将直接使用变量名构造测试脚本。根据最大值、最小值、可选项列表等属性,MATIS 可以自动生成测试数据[4],支持边缘测试、单元测试等多种功能测试。

边界类描述显示层信息,通常情况下一个边界类对应用户界面的一个窗体,边界类的成员变量对应窗体控件。为了给测试提供支持,按钮等用来触发功能的控件需要添加别名、功能类型等附加属性。除了最基本的四种数据操作——插入(C)、读取(R)、修改(U)、删除(D)等,功能类型还包括新建记录(N)、刷新(F)、数据记录导航(G)、退出(X)等信息系统常用功能。图 1 中 A4 活动为增删改读等不同类型的系统功能自动生成测试脚本时会用到功能类型定义。

UML 描述的设计模型使用统一的 XMI 格式保存,使用其他语言和工具描述的系统模型用 MATIS 规定的 XML 格式保存,不论哪种方式,通过扫描有格式的设计文件,MAITS 都可以很方便地抽取类属性定义和测试支持信息。

2.2　构建用户界面并生成描述信息

信息系统用户界面中的大部分可视化控件都被用来接受用户输入的信息,如输入框、选择框等,其他控件供用户触发功能,如按钮、菜单项等。这些控件与实体类和界面类的成员变量有着直接或间接的逻辑关联,如实体类中定义的数据字段通常是接受输入的控件的数据来源和数据去向。按一定格式把设计模型和用户界面的相关信息记录下来,可以形成一个从逻辑上描述用户界面的对照表,即逻辑对照表。借助逻辑对照表,测试脚本中的界面导航指令只用说明对逻辑对象的操作即可,在执行测试脚本时,由脚本引擎等执行单元负责把逻辑名称转换成实际控件的名称。逻辑对照表有效地把本来需要定义到系统设计模型里的技术细节分离了出来,保持了系统设计的抽象性。

靠手工生成逻辑对照表是不实际的,由此产生的大量维护工作会完全抵消自动测试的便利。只有依靠基于模型的用户界面自动生成方法来生成和维护逻辑对照表才能满足工程化的需要。用户界面设计和生成模型,如 Tadeus[5]、FMP[6]等,其原理是从用户需求出发,以任务为中心,辅以实体关系说明,形成界面表示单元,并最终自动产生用户界面。利用用户界面自动生成方法产生用户界面是系统实现过程的一部分,与实际的软件开发过程可以紧密结合在一起。

MATIS 方法使用从系统设计模型中抽取的实体类和边界类定义来构建实体关系说明,使用扩展的属性标记方法定义表示单元,主要定义如下:

对象定义:: = <普通属性 > <扩展属性 > <映射控件属性 >
普通属性:: = <名称 > <类型 > <长度 >
　　<最大值 > <最小值 > <缺省值 >
扩展属性:: = <别名 > <标题 > <显示类型 > <可选项列表 >
映射控件属性: = <控件名称 > <控件类型 >
　　<控件坐标 > <控件长度 > <控件宽度 >

表示单元是生成用户界面的基础定义,描述了逻辑对象和界面控件之间的关系。依据表示单元可以得到满足自动测试需要的逻辑对照表,包括对象名称与控件名称的对应关系,还有数据类型、可选项列表、控件类型等支持自动测试执行的信息。

由于逻辑对照表隔离了系统设计和系统实现,测试脚本与系统实现之间解除了紧耦合的关系。这使得自动测试的准备工作有可能提前到系统设计阶段,也就是说在构建系统设计模型的同时就可以创建测试脚本。当系统发生变更时,使用自动搜索比对程序去分析测试脚本,检查其中定义的逻辑对象名称等信息,可以发现失效测试脚本,这大大减轻了脚本维护工作的难度。利用用户界面自动生成方法维护逻辑对照表的过程也比较自然,开发人员根据系统变更修改用户界面,同时自动更新逻辑对照表,这有效地避免了由于设计与实现不一致造成的测试失败。

2.3　构建自动测试脚本

MATIS 方法采用了测试脚本这种方式完成界面导航并控制测试逻辑。如果去掉测试脚本这个中间环节,单纯依靠设计模型去完全自动地生成和执行测试用例,就需要在设计模型里定义系统逻辑的部分中添加大量的测试支持信息和约束条件,这给系统设计工作造成了比较重的负担。MATIS 依靠逻辑对照表和边界类中附加的功能类型定义,可以为功能测试自动生成测试用例,如以窗体为单位测试增删改功能,这满足了信息系统大部分的测试需求。对于复杂的测试用例,需要测试人员按照 MATIS 自定义的脚本规格手工编写测试脚本。采用测试脚本的方式使得测试工作的灵活性更高。

测试脚本包含了一系列的关键字,用来说明执行的功能。主要的关键字有:Import(导入指定的数据文件)、Set(为指定

的数据对象赋值)、Select(设置选择、列表、单选、网格等控件的当前选择项)、Trigger(触发功能控件)等。

2.4 生成测试数据

根据前面实体类成员变量的附加属性定义,MATIS 可以自动构造出测试初始化数据集、黑盒测试数据集、用于边缘测试的数据集、用于容错性测试的数据集和随机测试的数据集等几种测试数据。测试数据保存在外部数据文件中。

2.5 自动执行

有了自动测试脚本和测试数据,MATIS 可以自动执行测试工作,步骤如下:

1)创建一次测试交互(Test Transaction);

2)装载自动测试脚本和测试数据;

3)初始化数据库,完成必要的清理或插值工作;

4)启动待测试软件(SUT),初始化用户界面;

5)执行测试,检测测试结果,记录测试日志;

6)测试结束,清理数据库;

7)结束该次测试交互。

MATIS 自动执行测试的部分主要包括三个模块:测试脚本引擎、支持库和测试驱动,结构如图 2 所示。

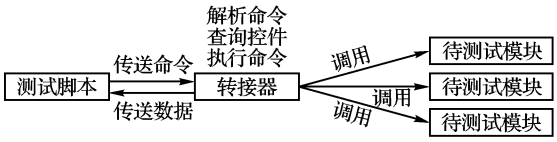

图 2 测试执行模块结构简图

测试脚本引擎解释执行测试脚本中定义的图形界面导航、执行业务功能、验证测试结果等命令,主要包括脚本解释执行器和测试脚本运行环境两部分,完成装载测试脚本,执行脚本,导入外部数据文件,检测测试逻辑,与测试驱动模块通信等功能。测试脚本引擎还负责在执行测试时动态地匹配数据对象和导入的测试数据,形成实际的赋值和操作语句。

支持库中封装了测试脚本常用的宏、基础函数和系统命令,如处理打开文件对话框的流程,取当前日期时间的函数等。测试脚本引擎在初始化运行环境,执行测试脚本时都会用到支持库里封装的功能。

在测试脚本和待测试模块之间,需要一个驱动模块起枢纽作用,负责完成解析测试命令,传送测试数据,执行赋值操作等具体工作。该模块被称为转接器。转接器的功能是独立于平台的,与具体的应用功能也没有关系,只由测试命令、测试通信协议等决定。转接器依靠动态反射机制[8]获得 SUT 包含的控件列表,通过查找逻辑对照表定位目标控件,根据控件类型设置控件属性,完成赋值、触发功能等实际操作。

3 实验结果

本文使用图书管理系统中的读者信息维护模块作为待测试软件,该模块实现了对图书信息增加、删除和修改的功能。读者类是该模块的实体类,包含读者编号、姓名、年龄、性别、联系电话、读者类型、发证日期等属性。读者信息维护类是界面类,包括增加、删除、修改、退出等按钮控件和多个输入信息的控件。

使用 MATIS 中的自动界面生成程序配置生成读者信息维护窗体,包括与实体类属性对应的 EditBox 控件,数据导航用的 Grid 控件和表示增加、删除等功能的 Button 控件,由此产生逻辑对照表的部分定义见表 1。分别使用 MI 公司的 WinRunner 和 MATIS 对该模块进行功能测试,结果如表 2 所示。

表 1 逻辑对照表部分内容

逻辑名称	控件名称	控件类型	可选项
读者编号	edtReaderID	EditBox	无
姓名	edtReaderName	EditBox	无
年龄	edtReaderAge	EditBox	无
性别	cboGender	ComboBox	0:男,1:女
增加信息	btnInsert	Button	无

表 2 测试结果

比较项目	WinRunner	MATIS
设计阶段预备工作	无	在设计模型中定义附加属性
测试脚本行数	40	5
脚本生成方式	手工	自动
编写脚本时间	10min	可忽略
测试数据生成方式	手工	自动
一组数据的准备时间(5 条)	5min	可忽略
脚本调试时间	5min	可忽略
执行方式	自动	自动
执行效果	成功执行功能测试	成功执行功能测试

由测试结果可以看出,在取得相同效果的前提下,MATIS 方法有效地缩短了准备测试脚本和测试数据的时间,提高了自动测试的效率。

4 结语

本文介绍了一个轻量级的基于设计模型的自动测试方法——MATIS。依靠在设计模型中增加的测试支持信息和用户界面自动生成方法,MATIS 方法把系统实现的细节同系统设计相隔离,既利用了系统设计的信息,又保持了设计的完整性和抽象性,同时有效地提高了自动测试的效率。

MATIS 方法可以很好地支持对功能模块实施功能测试,但在进行程序结构性测试和逻辑测试时还必须依靠手工编写测试脚本。

参考文献:

[1] DUSTIN E, RASHKA J, PAUL J. 自动化软件测试——入门、管理与实现[M]. 北京:清华大学出版社, 2003.

[2] HARTMANN A, NAGIN K. The AGEDIS Tools for Model Based Testing[A]. International Symposium on Software Testing and Analysis (ISSTA 2004)[C]. Boston, Massachusetts, USA, 2004.

[3] NETO PS, RESENDE R, PADUA C. A method for information systems testing automation[J]. Lecture Notes in Computer Science, 2005,3520. 504 - 518 .

[4] RUTHERFORD MJ, WOLF AL. A Case for Test-Code Generation in Model - Driven Systems[J]. PFENNING F, SMARAGDAKI Y, ed. GPCE 2003, LNCS 2830[C], 2003. 377 - 396.

[5] DA SILVA P, GRIFFITHS T, PATON N. Generating User Interface Code in a Model-Based User Interface Development Environment[A]. Proceedings of Advanced Visual Interface 2000[C], 2000. 155 - 160.

[6] 万建成, 孙彬. 支持用户界面自动生成的界面模型[J]. 计算机工程与应用, 2003,39(18):114 - 118.

[7] MOSLEY DJ, POSEY BA. 软件测试自动化[M]. 北京:机械工业出版社, 2003.

[8] MAES P. Concepts and Experiments in Computational Reflection [J]. Proceedings of the Conference on Object-Oriented Programming Systems, Languages and Applications(OOPSLA'87)[C]. Orlando, Florida, 1987. 147 - 155.