基于 Agent 的 Web 应用测试研究

付鹤岗,曾 刚
(重庆大学 计算机学院,重庆 400044)

摘 要:随着 Web 应用技术的发展,产生了一些新的技术特点,如动态行为、多种描述语言、新的数据处理机制、执行的不确定性等,这给软件测试者提出了严峻的挑战。为适应这些新特点,提出了一种基于 agent 的 Web 应用测试方法,借助 Beliefs－Desires－Intentions(BDI)模型生成 agent,结合控制流图和状态树对程序结构分析,获得测试序列集。实验结果显示测试序列能够在 agent 中实现,并体现出方法的灵活性和可扩展性。

关键词:Web 应用测试;agent;BDI 模型;测试序列
中图分类号:TP311.56 文献标识码:A 文章编号:1673－629X(2007)02－0138－05

Research for Agent－Based Web Application Test

FU He-gang,ZENG Gang
(Department of Computer Science, Chongqing University,Chongqing 400044,China)

Abstract:With development of Web application, some new characters are brought out, such as dynamic behaviors, heterogeneous representations, novel control flow and data flow mechanism, uncertain execution, etc. All these bring new challenges to the researchers. To meet these challenges, an agent－based approach for Web application testing is proposed in this paper. In experiment,agent was created based on Beliefs－Desires－Intentions model. Before this,control flow graph and object state tree are introduced to analyze the structure of program to get a test sequence. The experiment shows the result that test sequence can be implemented in the agent, which has the advantage of flexibility and extensibility.

Key words:Web application test;agent;BDI model;test sequence

0 引 言

近年来,Web 应用迅速发展,并广泛被大家接受,成功地应用于商业、政府的工作处理。Web 应用建立在 Internet 上,访问十分方便,并有开放的技术标准,一方面推动了 Web 应用的发展;另一方面也带来了新的特点,如动态行为、多种描述语言、新的数据处理机制、执行的不确定性等[1],这给软件测试提出了严峻的挑战。

动态行为:在 Web 应用中,服务器端根据用户请求信息,动态地产生 Web 页面,并将产生的信息响应给用户。这些发送到客户端的信息可能包括 JavaScript,VBScript 等,它们也有自己的行为,在客户端也会动态运行。这种行为是动态产生的,也是传统软件执行过程中不具有的,所以用传统的测试方法是很难测试动态行为的。

多种描述语言:因为 Web 应用是一种开放的技术,可以使用多种描述语言,例如:HTML,Java,XML,JavaScript,JSP 和厂家发行的应用组件等。各种语言和组件运行的平台可能是不同的,但在 Web 应用中可以轻松地组合起来,实现用户要求的功能,这也是传统软件没有的特点,使用传统的测试方法很难实现整个系统的测试。

新的数据处理机制:Web 应用技术让数据轻松地在 Web 应用的各个部分传递。例如 HTML 和 XML 文件可以携带数据在客户端和服务器之间自由传输。XML 文件内部的变量仍可为程序变量使用。这比传统单一的数据传递机制更有吸引力,同时也增加了测试的难度。

执行的不确定性:服务器可以根据用户当前请求信息独立执行,而不需要历史信息,所以系统的状态是根据用户不同的请求信息进入不同的执行状态,使得用户可以自己选择执行序列,例如点击‘返回’按钮改变执行序列。而传统的执行顺序不易改变,用传统的测试方法测试不确定的执行状态是非常困难的。

Web 应用的这些新特点比传统应用系统更复杂,用传统的测试方法来测试复杂的 Web 应用是不充分

收稿日期:2006－05－08
作者简介:付鹤岗(1950－),男,重庆人,副教授,硕士生导师,研究方向为软件工程、电子商务等。

的。现在很多的 Web 应用测试依赖开发者的经验,缺少系统的、灵活的、可扩展的测试方法。

目前,已经研究出了很多 Web 应用测试方法。基于 agent 的 Web 应用测试方法就是一种灵活的、可扩展的 Web 应用测试技术。

1　相关技术

Web 应用技术的新特点促进了 Web 应用的发展。同时,用户对软件质量的要求,也促使研究者在 Web 应用测试方面做了大量研究。

Tonella 和 Ricca 在传统测试方法的基础上,提出了二层 Web 应用测试模型[2]。结合高层的导航模型和低层的控制流模型,用白盒测试的方法完成整个 Web 应用的测试。这种测试方法十分复杂,并且没有可扩展性,因此,跟传统的单元测试非常相似。

Kung 等提出了一种面向对象的 Web 测试模型[3~5]。从对象、行为和结构三方面分析 Web 应用,并定义了一系列的图,如导航状态图、交互过程控制流图、对象控制流图等。在此基础上提出了四层数据流测试方法。但有一个潜在的问题,当 Web 应用存在多种描述语言时,就需要扩展测试方法,在这种 Web 测试模型中是不能实现的。

文中提出了一种基于 agent 的 Web 应用测试方法。该方法将 Web 应用分为三层:方法层、对象层和 Web 应用层,按照至底向上的关系在各层建立测试 agent,负责各层的测试任务。这里引进了 Belief Desire Intention (BDI)模型用于生成 agent 对象,定义 agent 执行的活动。

2　基于 agent 的 Web 应用测试

面向 agent 的软件测试是一项新的研究方向,由于其自治的特点,在复杂的软件测试工作中逐渐受到重视。基于 agent 的 Web 应用测试方法在 Web 应用的各层上创建 agent,负责各自的测试任务。通过组合模式把这些测试 agent 组合在一起,协作完成整个系统的测试。各个 agent 是建立在 BDI 模型和 UML 的基础上的。每个 agent 是一个独立的对象,因此可针对具体的 Web 文档,定义合适的测试方法和测试模型。

2.1　Beliefs－Desires－Intentions 模型

BDI 模型是一种合理的推理行为,已经成功地应用于 Web 应用中。基于 agent 的 Web 应用测试方法建立的 agent 继承了 BDI 模型定义抽象类,如 Belief, Goal, Plan, Agent, Agent Communication Act 等,用于 agent 在执行过程中的测试活动;也定义了新的图,如 Agent Goal diagram, Use Case Goal Diagram, Agent Domain Model, Agent Sequence Diagram 等[6],在静态分析系统结构的基础上,获取 agent 执行目标、测试序列、测试结果,使得 BDI 模型也能适应 Web 应用测试。具体证明参考文献[6]。

BDI 模型描述了一个测试过程。其中 Beliefs 描述了被测试组件和测试结果。包括测试组件所需要的测试方法、测试的类别。被测试的目标对象可以是一个源文件,也可以是一个对象。Desires 记录测试要求达到的测试标准。目前有以下几种测试标准:语句覆盖率,链接覆盖率和路径覆盖率,选择不同的测试标准对测试结果都有很大的影响。具体请参考文献[7]。Intentions 描述了一个测试计划,执行测试的一系列活动,包括执行顺序、代价、前提条件、后续条件。

因此,在明确了测试内容和测试目标的情况下,接下来是生成测试 agent。基于 agent 的测试方法定义了 agent 的类型,根据数据流图确定了 agent 的执行序列。因此,创建一个 agent 是非常容易的,agent 内可定义多个测试方法,用来测试不同的内容。笔者根据 BDI 模型提出了如下测试步骤:

(1) 收集测试对象,建立测试环境。明确测试对象以及测试所需要的环境参数以及桩函数。选择合适的测试方法。

(2) 制定测试标准。定义测试要求达到的代码覆盖率或节点可达性。

(3) 借助 UML 和 BDI 模型新定义的图,分析测试用例,获取测试序列。

(4) 执行测试用例,记录测试结果。

(5) 评价测试结果是否达到测试标准。如果没有达到测试要求,跳转到第(3)步执行;否则,测试完成。

因此,借助 BDI 模型分析 Web 应用系统,能够轻松地获得 agent 的测试序列。并且根据 Web 文档的不同选择合适的测试方法,按照以上步骤,创建 agent 执行测试用例,完成测试任务,这体现了测试 agent 自治的特点。Web 应用中存在多种数据信息,如:文本、图像和声音等,和不同的开发语言,如 JAVA, JavaScript, HTML 等,agent 选择最适合的测试方法或测试工具执行测试,这体现了 agent 在执行过程中的灵活性。一旦按照 BDI 模型得到了 Agent Goal diagram, Use Case Goal Diagram, Agent Domain Model, Agent Sequence Diagram 等图[6],就可以生成 agent,自动完成测试,这也体现了 agent 的自动化特性。

2.2　基于 agent 的 Web 应用测试方法

在面向对象的 Web 应用设计中,Web 应用都可分为三层体系结构:方法层,对象层,Web 应用层,各层负责不同的功能实现,并有至底向上的关系(如图 1 所

示)。根据此特点,在设计 Web 应用系统测试中,分别在 Web 应用各层上设计测试模型,生成 agent,分别完成各层的测试任务。按照至底向上的关系把各个 agent 集成为一个测试系统。

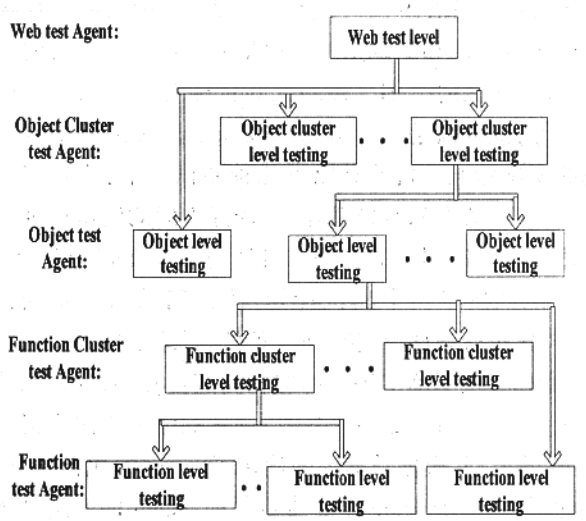

图 1 Web 应用测试各层 agent 对象图

Web 应用各层上建立的 agent 对象是相对独立的,因此根据测试对象的特点,选择合适的测试方法,如功能测试、结构测试和行为测试。也可以采用不同的测试用例分析方法,如数据流和控制流图、状态图、生成树等,提取测试用例。各层 agent 负责 Web 应用各层的测试任务,这是其它层 agent 不能完成的。

但是 agent 与 agent 之间是可以交互的。对象级测试 agent 需要方法级 agent 支持其完成对象级的测试任务;Web 应用级的测试 agent 运用组合模式将各层测试 agent 组合起来,按照某种测试序列实现整个 Web 系统的测试。从方法级到 Web 应用级,自底向上实现整个系统的测试。

2.3 Web 应用分层测试

以上分析了测试模型及 Web 应用测试方法,下面具体介绍 Web 应用测试在各层上的实现。

* 方法级测试(Function Level Testing)。

方法级测试主要是对系统内定义的方法的测试,通过分析方法定义变量的定义使用链,借助控制流图(Control Flow Graph)清楚地描述出来,分析选择合适的测试用例及测试序列[4,8]。CFG 是由节点和带注释的边构成的,其中节点表示一句执行语句,也就是变量的定义或者使用;而边定义了数据流的控制情况。每一个 CFG 都只有一个入口和一个出口,这符合程序中方法的定义规则。

在方法级测试中,从数据流图分析变量的定义 – 使用链,生成测试用例,容易计算出代码覆盖率,作为一项测试标准。

接下来是按照 BDI 模型生成测试 agent。定义测试方法用以测试程序中定义的方法。值得注意的是由 BDI 模型可以生成多个 agent,每个 agent 可以定义多个方法,这个特点增强了基于 agent 测试方法的灵活性和可扩展性,解决了 Web 应用技术存在的多种描述语言和多种数据处理机制的测试问题。

* 对象级测试(Object Level Testing)。

分析系统中对象在执行过程中的状态变迁,借助状态图获得对象状态序列,即得到了该对象的测试序列。对象执行过程中可能存在与其他对象交互的行为,借助一棵多对象的状态树[3]描述各个对象的状态及对象与对象间的交互行为。状态树是各个对象的状态集成在一起,由节点和边构成的,节点描述了对象关系图中各个对象的状态,边则是触发状态变化的某个操作。所以状态生成树也描述了对象集中每个对象的状态变化过程,它是状态图的扩展。从树的根节点到每个叶子节点描述了系统中对象的状态变化过程,即是对象和方法的调用序列。

对象级测试的目标则是按照状态树分析方法的调用序列和状态行为。测试其可达性,是否存在死锁,状态转换是否合理。

接下来就是继承 BDI 模型生成对象级 agent,定义对象级 agent 方法。对象级 agent 方法可以调用方法级 agent 执行方法级测试,使得对象级测试与方法级测试统一起来,集成一个统一的测试系统。因为对象级 agent 可以根据需要加入到测试系统中,这也体现了 agent 测试系统的可扩展性,解决了 Web 技术动态执行的测试问题。

* Web 应用级测试(Web Application Level Testing)。

Web 应用级测试是组合各个低层 agent,使它们协同工作,共同完成 Web 级测试,如图 2 所示。

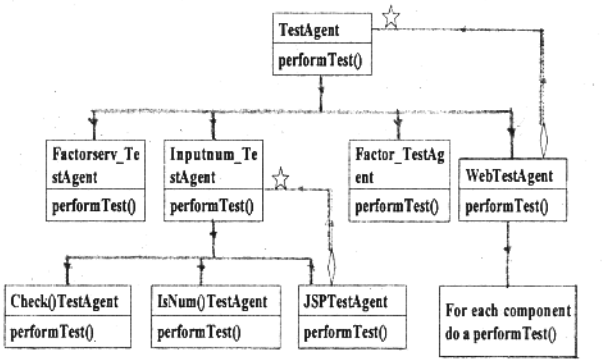

图 2 Web 级测试

Web 级测试的目标就是测试各个 agent 工作的一致性,执行序列的正确性。

从 Web 应用的分层测试方法来看,各层上的 a-

gent 在选择各自的测试方法执行测试过程中,可以根据测试内容的不同生成 agent,agent 也定义了不同的测试方法,执行该部分的测试活动。所以,基于 agent 的 Web 应用测试系统是一种灵活的、可扩展的测试方法,非常适合 Web 应用测试。

3　实例研究及结果分析

实验在 Win2000 平台下,用 SUN 公司提供的开发包 jdk5.0 和 APATCHE 公司提供的免费服务器 tomcat5.5.7 开发的一个 Factor 系统进行测试。Factor 系统由三个 JSP 文件(innum.jsp, success.jsp,error.jsp)、两个类文件(Factor.class, FactorServlet.class)和软件开发环境组成。实验按照 Web 分层测试原理在方法级和对象级测试,主要选择了两个类文件,测试它们方法中变量的定义使用链和对象的状态。

3.1　方法级测试及结果分析

在方法级的测试过程中,生成了三种测试 agent:方法级总 agent、交互方法测试 agent、方法测试 agent。在程序中对应为:MethodMainTestAgent,InnerMethodTestAgent,MethodTestAgent,都可以生成多个 agent,每个 agent 都可以实现多个测试方法,如图 3 所示。

```
源代码:
public class MethodTestAgent {
  public static boolean testCheck (String num, String expect) {
    Factor factor = new Factor();
    boolean f = factor.check(num);
    if (expect.equals(String.valueOf(f))) {
      return true ;
    } else {
      return false ;
    }
  }
  public boolean testMethodN() {
    return true ;
  }
}
用log4j记录的实验结果:
MethodMainTestAgent - MethodMainTestAgent start!
InnerMethodTestAgent - InnerMethodTestAgent start
InnerMethodTestAgent - test factorial()param str=3expect=6
MethodTestAgent - Agent start to test Check:
MethodTestAgent - test check()parameter num=3expect=true
Factor - define num=0123456789.
Factor - ch=3
Factor - is number
MethodTestAgent - 3 is a number:true
MethodTestAgent - the check() result is equal to our expect

InnerMethodTestAgent - test factorial()
****************** test check(3) ******************
Factor - define i=1
Factor - c-use i=3; c-use num=3
Factor - c-use i=6; c-use num=2
Factor - c-use i=6; c-use num=1
Factor - return the result i=6
InnerMethodTestAgent - the factorial() result is equal to
our expect
MethodMainTestAgent - MethodMainTestAgent end!
```

图 3　源码及测试结果

MethodTestAgent 可定义多个测试方法。在为每个方法生成测试 agent 或测试方法之后,最重要的是确定测试序列。本例中实验结果也描述了测试 agent 执行的序列。在 InnerMethodTestAgent 执行 factorial()测试过程中,先调用 MethodTestAgent 执行 check()测试。实验结果验证,它与数据流图定义的数据流顺序是一致的,如图 4 所示。根据数据流图描述得到的测试序列能够在基于 agent 的测试方法中实现。

实验结果还记录了执行过程中各个变量的定义使用链,它与数据流图描述的控制结构是一致的。从而轻松计算出代码覆盖率,评价测试结果是否达到测试标准。本次实验选择了 3 个具有代表性的测试用例,实验结果分析如表 1 所示,选择这 3 个测试用例覆盖了程序执行的所有路径,也覆盖了程序执行的所有代码。因此,基于 agent 的测试方法能够执行 Web 应用方法级的测试。

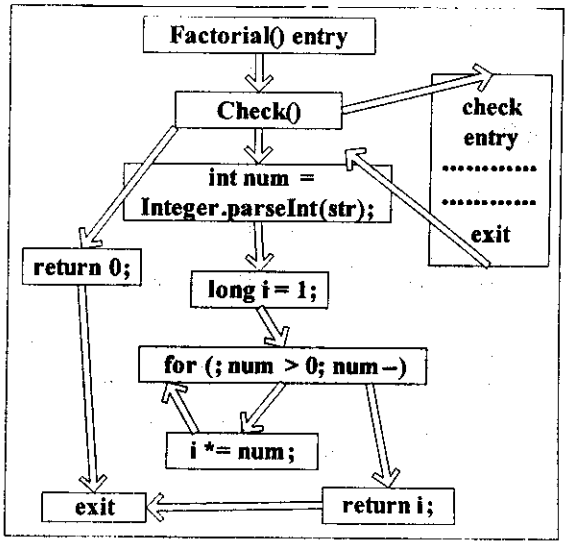

图 4　数据流图

表 1　factorial()方法测试结果分析

测试用例	预期结果	总代码	覆盖代码	代码覆盖率
letter	false	12	7	58.3%
3.f	false	12	7	58.3%
23	true	12	10	83.3%

3.2　对象级测试及结果分析

对象级测试分析各个对象的状态,构造一棵测试状态树(生成状态树参考文献[3]),从状态树的根节点到每个叶子节点都构成一条测试路径。这条测试路径表示了某个对象在系统执行过程中的各种状态(如图 5 所示),用对象级 agent 测试对象的各个状态。图 6 记录了 Obj1TestAgent 测试 Factor 对象 4 个状态(start,wait,test check(), test factorial(), end)的实验结果,实现对 Factor 对象的状态测试,这与状态图生成的状态树所描述的状态过程是一致的。实验结果也记录了每个状态测试过程中调用方法级测试 agent 对方

法的测试。因此,基于 agent 的测试方法能够实现对象的状态测试。

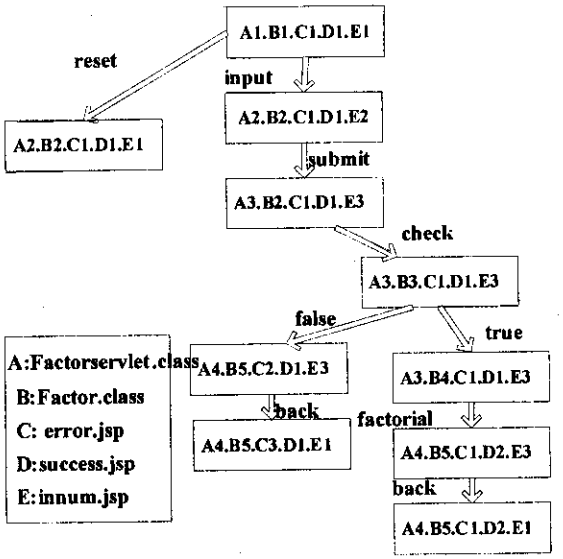

图 5 系统对象状态树

```
源代码:
public static void testObj1Agent() {
    Factor f = new Factor();
    String input1 = "8.f";
    MethodTestAgent.testCheck(input1,"false");
   String input2 = "8.f";
    InnerMethodTestAgent.testFactorial(input2,"false");
  }
log4j记录的实验结果:
Obj1TestAgent - Factor Obj start!
Obj1TestAgent - Factor Object is waiting for
input
MethodTestAgent - Agent start to test Check:
      ·····test method check()·····
Obj1TestAgent - call MethodTestAgent to test
method check()
      ·····test method factorial()·····
Obj1TestAgent - call InnerMethodTestAgent to
test method factorial()
Obj1TestAgent - Factor Obj end!
```

图 6 Factor 对象状态测试结果

4 总 结

提出了一种基于 agent 的 Web 应用测试方法,并用实验证明 agent 是可以根据需要灵活创建的,并适时加入到测试系统中,体现了基于 agent 测试方法的可扩展性,解决了 Web 应用带来的执行不确定、新的数据处理机制、多种描述语言、动态行为等新特点。实验结果显示,这种基于 agent 的测试方法能够实现 Web 应用的测试。但在 agent 协作工作时,怎样确定其执行顺序,还需要进一步完善。

参考文献:

[1] Qi Y, Kung D, Wong E. An agent - based testing approach for Web applications [C]// 29th Annual International Computer Software and Applications Conference, COMPSAC 2005. Edinburgh, UK: [s. n.], 2005: 45 - 50.

[2] Tonella P, Ricca P. A 2 - layer model for the White - box testing of Web applications [C]// Proceedings of International Workshop on Web Site Evolution, 2004. Chicago, Illinois: [s. n.], 2004: 11 - 19.

[3] Kung D C, Liu C H. Hsia P. An object - oriented Web test model for testing Web applications [C] // 24th Annual International Computer Software and Applications Conference, COMPSAC 2000. Taipei, Taiwan: [s. n.], 2000: 537 - 542.

[4] Liu C H, Kung D C, Hsia P, et al. Object based data flow testing of Web applications, Quality Software [C]// 2000. Proceedings. First Asia - Pacific Conference. Hong Kong, China: [s. n.], 2000: 7 - 16.

[5] Liu C H, Kung D C, Hsia P, et al. Structural testing of Web applications [C]// 11th International Symposium on Software Reliability Engineering (ISSRE '00). San Jose: [s. n.], 2000: 84 - 96.

[6] Kung D C. An agent - based framework for testing Web applications [C]// 28th Annual International Computer Software and Applications Conference, COMPSAC2004. Hong Kong, China: [s. n.], 2004: 174 - 177.

[7] Huo Qingning, Zhu Hong, Greenwood S. A Multi - Agent Software Environment for Testing Web - based Applications [C]// 27th Annual International Computer Software and Applications Conference, COMPSAC2003. Dallas, Texas, USA: [s. n.], 2003: 210 - 215.

[8] 刘 勇,曾 明,朱 利,等. 基于数据流的软件测试序列自动生成技术研究[J]. 微电子学与计算机,2005,22(5): 131 - 135.

(上接第 137 页)

参考文献:

[1] Lee M D. Fast Text Classification Using Sequential Sampling Processes [C] // Proc. of the 14th Australian Joint Conf. on Artificial Intelligence. Berlin: Springer - Verlag, 2001: 309 - 320.

[2] 司卫国,赵捧未. 移动 Agent 在信息检索中的应用研究[J]. 电子科技,2004(11): 57 - 60.

[3] 姚天顺. 基于特征相关性的汉语文本自动分类模型的研究[J]. 小型微型计算机系统,1998,19(8): 49 - 54.

[4] 《中图法》编委会. 中国分类主题词表[S]. 北京: 北京图书馆出版社,2005.

[5] GB12200. 1 - 90 汉语信息处理词汇 01 部分:基本术语[S]. 1990.