基于 ORD 和 FSM 的 Web 应用的建模与测试*)

钱忠胜[1,2] 缪淮扣[1] 陈圣波[1]

(上海大学计算机工程与科学学院 上海 200072)[1] (江西财经大学信息管理学院 南昌 330013)[2]

摘 要 Web 测试是保证高质量 Web 应用的一种有效技术。然而,由于其特殊性和复杂性,很难直接将传统的测试理论与方法学运用到 Web 应用的测试当中来。对 Web 应用进行了分析与建模,并对其进行测试,提出了一种可行的 Web 测试模型。首先得到页面流图(PFD, Page Flow Diagram),进而产生对象关系图(ORD, Object Relation Diagram),然后根据提出的算法将 ORD 转化为形式化的有限状态机(FSM, Finite State Machine)模型。基于 FSM 模型,提出了一种有效的测试路径自动生成方法,这些测试路径可以转化为 XML 语法的测试规格说明。测试引擎将测试规格说明作为输入最终产生测试报告。全文以所开发的一个小型的 Web 应用 SWLS(Simple Web Login System)为例进行阐述。

关键词 Web 应用,页面流图,对象关系图,有限状态机,形式化,测试路径,Web 测试模型

Modeling and Testing Web Applications Based on ORD and FSM

QIAN Zhong-sheng[1,2] MIAO Huai-kou[1] CHEN Sheng-bo[1]

(School of Computer Engineering & Science, Shanghai University, Shanghai 200072, China)[1]

(School of Information Technology, Jiangxi University of Finance & Economics, Nanchang 330013, China)[2]

Abstract Web testing is an effective technique to ensure the high quality of Web applications. Unfortunately, it is hard to directly employ the traditional test theories and methodologies because of the particularities and complexities of Web applications. The paper analyzes, models and tests Web applications, and then presents a practical Web test model. It begins with PFD (Page Flow Diagram) and then generates ORD (Object Relation Diagram), which is converted into FSM (Finite State Machine) model according to an algorithm proposed. An efficient approach to automatically generating test paths is presented based on the FSM model. The test paths can be translated into test specification in XML syntax. Finally, the test engine takes the test specification as its input and produces a test report. The SWLS (Simple Web Login System), a small Web application developed, is taken as an example throughout the paper.

Keywords Web applications, PFD, ORD, FSM, Formalization, Test path, Web test model

1 引言

随着网络互联技术、信息技术以及 Web 技术的发展,存在越来越多的基于 Web 的应用,在这些应用中,我们可以方便快捷地处理各种各样的信息。然而,Web 应用由不同种类的软件组件构成,它们之间以及和用户之间以全新的方式进行交互。软件组件必须满足高的可靠性、可用性和有效性需求。分析、建模和测试 Web 应用对软件开发者和设计者提出了新的挑战。

Web 应用有别于传统的软件系统,其测试带来了许多新的问题[1,2]。构建一个即使是简单的 Web 应用也是一项很耗时的任务,我们必须仔细分析和建模其结构和行为,Web 测试方法也需自动化,测试用例也要有自适应能力。然而,Web 应用提出了一些重要而具有挑战性的测试问题,这些问题不能直接用现有的对传统系统的测试技术来解决。本文从 Web 应用的需求出发,对其进行分析和建模,采用形式化的方法,即 FSM 自动产生测试路径,进而得到测试用例,转化为测试规格说明后作为测试引擎的输入,最终产生测试报告。整个过程很少人工干预,基本实现了自动化。

2 相关工作

已有一些研究探讨了 Web 应用测试领域的问题,也提出了一些方法和技术来有效地测试 Web 应用。每种方法和技术具有不同的起源并追求不同的测试目标以应对 Web 应用的特殊性。链接测试者主要发现网页间不正确的导航链接所产生的错误,比如到达一个不是所期望的页、悬挂链接(pending link)、断裂链接(broken link)以及链接到不可达的页。表单测试者创建脚本初始化一个表单,点击每个按钮,在文本域中输入预置的脚本,最后按提交按钮。兼容性测试者确认一个 Web 应用能够在不同的 Web 浏览器中执行时表现正确的行为。

文献[3]使用形式化的语言 Z 来分析,并提出了一个 Web 测试模型。Z 表示法是一个描述性的方法,能很好地对需求进行规格说明,但 Z 在描述行为方面不如 FSM 容易理解。我们采用 FSM 作为自动获取测试路径的形式化工具,提出的 Web 测试模型也对文献[3]的测试模型进行了扩展。A. Andrews 等[4]提出了从 FSM 推导测试的方法,测试序列的产生基于 FSM,并试图用输入的约束条件来减少状态空间。但他们没有很好地解决状态的存储问题,而我们采用 XML 文件来存

*)本文获国家自然科学基金(60373072 和 60673115)、国家 973 项目(2007CB310801)和上海市教委科学发展基金(05AZ70)资助。钱忠胜 讲师,博士生,主要从事 Web 应用的建模与测试、软件形式化研究;缪淮扣 教授,博导,主要从事软件自动化、软件形式化研究;陈圣波 博士生,主要从事基于 Agent 的 Web 应用的测试、软件形式化研究。

储测试路径,测试规格说明也是基于 XML 语法的。

D. C. Kung 等[5]描述了一个面向对象的 Web 测试模型 WTM 支持 Web 应用的测试。WTM 捕获测试制品(artifact)的结构和行为。Web 应用的每个实体被建模为一个对象,并利用对象关系图(ORD)描述对象以及它们之间的关系,利用页面导航图(PND,Page Navigation Diagram)描述导航行为,而交互对象之间的状态行为用对象状态图(OSD,Object State Diagram)来描述。此外,控制与数据流信息用块分枝图(BBD,Block Branch Diagram)和功能聚簇图(FCD,Function Cluster Diagram)描述。E. Hieatt 等[2]介绍了验收测试的方法,开发了一个测试工具表达系统的操作和 XML 形式的期望输出结果,用 XML 形式定义的输出结果类似于我们的测试规格说明。然而,D. C. Kung 和 E. Hieatt 等提出的方法都集中于单元测试,而我们的方法更关注于 Web 应用的整体测试。

Benedikt 等[6]开发了一个动态导航工具 VeriWeb,该工具从一个给定的 URL 出发,通过非确定地搜索动作序列得到链接序列。VeriWeb 的测试是基于图的,结点表示页,边表示 HTML 链接,图的大小采用剪枝策略来控制。而我们的方法基于 ORD 和 FSM,每个测试用例是测试步的树型结构。

文献[7]综述了 Web 应用的测试。文章指出了 Web 应用测试和传统软件测试之间的主要差别,这些差别是如何影响前者的,以及最近一些年来在 Web 应用测试领域的相关成果。但其主要关注 Web 应用的功能测试,即使讨论了部分非功能的需求方面。

3 Web 应用的建模

Web 应用利用许多新的语言、技术和编程模型,并用来实现高度交互的应用,这些应用具有高质量的需求。不同的交互风格、具有应用操作的超媒体特性以及复杂的信息结构的并存,对建模人员提出了许多新的问题,这使得他们需要寻求一种新的建模范型。

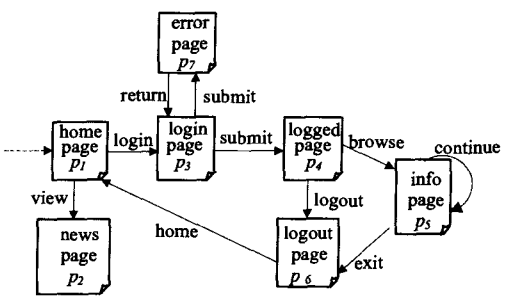

图 1 SWLS 的 PFD

为了描述所提出的方法,以我们开发的一个小型 Web 应用 SWLS(Simple Web Login System)为例,它的页面流图(PFD)如图 1 所示。主页 home page(p_1)由虚箭头指示,一般可以认为一个称为 blank 的页用来对一个 Web 应用的主页发出请求。p_1 由 news page(p_2)和 login page(p_3)这两个页组成,它们分别显示在 p_1 的两个框架(frame)中。用户在登录页 p_3 输入用户 id 和口令,然后点击 submit 按钮请求服务器的验证。如果用户 id 和口令都是正确的,就显示 logged page(p_4)页。相反,只要用户 id 和口令其中一个输入值有误,将会显示一个包含错误信息的 error page(p_7)页。在 p_4 页,用户可以通过 browse 按钮进入 info page(p_5)页浏览个人的私人信息。如果 p_5 页太长,用户可以使用网页内链接(intra-page link)continue 浏览同一页 p_5 的不同部分。当用户在 p_4 页按 logout 按钮或在 p_5 页按 exit 按钮时,将会显示 logout page(p_6)页,在此页,他还可以通过链接 home 回到主页 p_1。如果需要的话,此时他可以重新登录。必须注意的是,每次进入登录页 p_3 时,用户 id 和口令的值必须初始化为空。

3.1 ORD 模型

一个典型的 Web 应用包含许多页和相关的软件组件。一个 Web 页包含显示给用户的内容和指向其它页的导航链接,在本文中,一个页被定义为 Web 浏览器在单个窗口中所显示的信息,该信息可能是一个驻留在服务器上的静态的 HTML 文件,或者是软件组件执行后得到的动态输出结果。一个 Web 应用涉及的软件组件可以是 ASP(Active Server Pages)、JSP(Java Server Pages)、CGI(Common Gateway Interfaces)、ActiveX 控件、Java Beans 或服务器端包含的 HTML 文件,甚至是远程 Web 服务等。

Web 页和软件组件之间有多种类型的关系。一个服务器页 p 可以重定向(redirect)一个 HTTP 请求到服务器页 q。通过超链接属性 href,一个链接可以从一个页链接(link)到另一个页。框架和框架集元素使得一个页可以包含(consist of)一些其它的页,每一个对应一个框架。Web 页可以通过超链接属性或表单的动作属性发送请求来调用(call)软件组件。而软件组件可以动态产生(build)新的 Web 页以响应请求。此外,软件组件之间也可互相访问。

为了表示一个 Web 应用的实体以及它们之间的关系,我们采用对象关系图(ORD)[5]表示其对象模型。

定义 1 一个 ORD $R=(V,L,E)$ 是一个有向图,其中 V 是一个有限对象的集合;L 是一个有限非空的标签集,表示关系的类型;$E(\subseteq V\times L\times V)$ 是对象之间关系的集合。

此外,ORD 允许递归地定义,因此,一个复杂的对象可以表示成层次结构,使不同的层次对应不同的抽象和细节。

在设计阶段,一个 Web 应用可以表示为一个对象模型 ORD,使得

- $V=V_p\cup V_c$,其中 V_p 是 Web 页的集合,包括客户端页面和服务器端页面,V_c 是关联的软件组件的集合,软件组件接收 Web 页的请求或者动态产生 Web 页;
- $L=\{$redirect,link,consist of,call,build$\}$,包含的关系有五种类型;
- $E(\subseteq V\times L\times V)$ 表示 Web 页和软件组件之间的关系。

上面提到的 SWLS 的页面流图是该 Web 应用的一个用户视图,而 ORD 模型是从设计的角度来考虑的。SWLS 的 ORD 模型如图 2 所示,其中折角框表示 Web 页,圆角矩形表示软件组件。

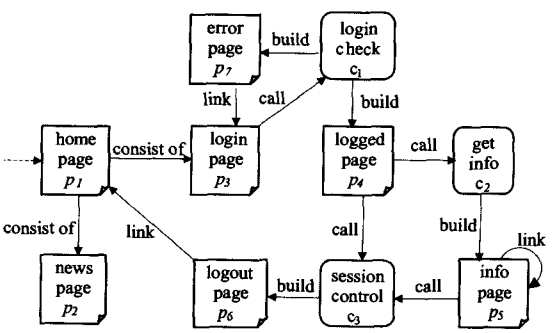

图 2 SWLS 的 ORD 模型

3.2 FSM模型

Web应用通常运行在一台或多台连接于Internet的服务器上，用户可以通过浏览器与其进行交互。为了自动产生测试用例，我们需要形式化地建模他们的行为，所使用的形式化行为建模工具是有限状态机(FSM)[8,9]。

定义2 一个FSM Q是一个五元组(S,Σ,δ,s_0,F)，其中S是一个状态集合；Σ是迁移(transition)的集合；δ是一个从$S\times\Sigma$到S的状态转换函数，$\delta(s_1,t)=s_2$表示在状态s_1，当迁移为t时，将转换到状态s_2；s_0是唯一的初始状态；F是终止状态的集合。

为满足本文的需要，这里扩展FSM Q为一个六元组$(S,\Sigma,\delta,G,s_0,F)$，其中$G$是卫(guard)的集合；状态转换函数$\delta$重定义为从$S\times\Sigma\times G$到$S$的转换，$\delta(s_1,t,g)=s_2$表示在状态$s_1$，当迁移为$t$，卫为$g$时，将转换到状态$s_2$。$g\in G$表达了用户潜在的输入信息，可以是某个特定的上下文(比如用户输入了正确的值)，也可以是常量True，False等。为表达方便，我们称扩展的FSM仍为FSM。

下面给出从ORD模型$R=(V,L,E)$到FSM模型$Q=(S,\Sigma,\delta,G,s_0,F)$的转换算法：

(1)对于任一个结点$v\in V$，建立一个状态$s\in S$，v和s之间是一一对应的关系；

(2)$\Sigma=V$；

(3)对于任一个关系$(v_1,k,v_2)\in E$，建立一个状态转换$(s_1,v_1,g,s_2)\in\delta$，使得s_1，s_2分别为v_1，v_2在(1)中建立时对应的状态，g为给出的一个初始的卫；

(4)把第一个进入的页面(一般为主页)结点对应的状态作为初始状态s_0；

(5)把初始状态同时作为一个终止状态，如果有某些状态到其它任何一个状态都不可达，则把这些状态也作为终止状态；

(6)引入一个新的终止状态f，使在(5)中得到的每个终止状态q都转换到f，即$(q,k,g,f)\in\delta$，其中迁移k为状态q对应的结点所表示的页，g为给出的一个初始的卫，q不再作为终止状态，即$F=\{f\}$；

(7)在(3)、(6)中引入的所有初始的卫构成卫的集合G。

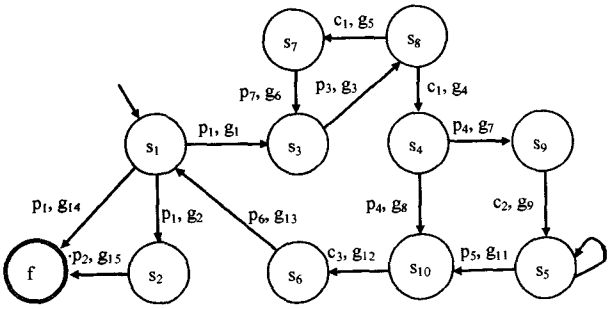

图3 SWLS的FSM状态转换图

根据给出的转换算法，可得出SWLS的FSM $Q=(S,\Sigma,\delta,G,s_0,F)$的状态转换图(STD，State Transition Diagram)如图3所示，其中$S=\{s_1,s_2,s_3,s_4,s_5,s_6,s_7,s_8,s_9,s_{10},f\}$，$\Sigma=\{p_1,p_2,p_3,p_4,p_5,p_6,p_7,c_1,c_2,c_3\}$，$\delta=\{(s_1,p_1,g_1,s_3),(s_1,p_1,g_2,s_2),(s_3,p_3,g_3,s_8),(s_8,c_1,g_4,s_4),(s_8,c_1,g_5,s_7),(s_7,p_7,g_6,s_3),(s_4,p_4,g_7,s_9),(s_4,p_4,g_8,s_{10}),(s_9,c_2,g_9,s_5),(s_5,p_5,g_{10},s_5),(s_5,p_5,g_{11},s_{10}),(s_{10},c_3,g_{12},s_6),(s_6,p_6,g_{13},s_1),(s_1,p_1,g_{14},f),(s_2,p_2,g_{15},f)\}$，$G=\{g_1,g_2,g_3,g_4,g_5,g_6,g_7,g_8,g_9,g_{10},g_{11},g_{12},g_{13},g_{14},g_{15}\}$，$s_0=s_1$为初始状态，$F=\{f\}$为终止状态的集合。特别注意，这里的终止状态不表示用户导航行为的终止，而是指一条可能的测试路径的结束。

状态转换图可以看作是FSM的图形化表示，状态转换函数也能很好地体现出来。例如，在SWLS的语义环境下，$\delta(s_1,p_1,g_1)=s_3$表达了用户在状态s_1进入了页面p_1，点击login按钮(该动作的结果由g_1表达)，转到状态s_3(对应页p_3)进行登录；而$\delta(s_1,p_1,g_2)=s_2$表达了用户在状态s_1进入了页面p_1，点击*view*按钮(该动作的结果由g_2表达)，转到状态s_2(对应页p_2)查看新闻。

4 Web应用的测试

Web应用给我们带来了具有挑战性的测试问题，这些问题不能直接用现成的针对传统系统的测试技术来解决。因此，测试技术需要有适应能力(adaptation)，并且必须开发一个新的特定于Web应用的测试方法。

4.1 测试路径的生成

下面给出本文对路径和测试路径的概念。

定义3 一条路径为FSM中从初始状态到终止状态的一个迁移序列。

定义4 一条测试路径为FSM中长度至少为2的最短路径。

"最短"是指该路径不包含其它的路径，即不将其它路径作为它的子路径(前缀或后缀)，也不包含多余的(即出现不必要的重复)迁移。例如，假设α=abcdaef，β=abcda和γ=abcbcda是三条路径，由于α包含β，将β作为它的前缀，所以α不是测试路径；γ中连续出现了两次"bc"，显然出现了不必要的重复，故γ也不是测试路径；而β是长度至少为2的最短的路径，它是一条测试路径。

定义5 一个路径表达式是由操作符*、+、·、|和括号(、)连接起来的FSM中迁移组成的顺序串。

测试路径由路径表达式给出，因此首先应该得到路径表达式。先给出两条转换规则和一条求解规则，如图4所示。为表达方便，这里省略了卫$g(\in G)$，但不影响路径表达式的生成。规则$S\to pR$可以理解为当用户从页面p导航到下一个页面时，其状态从S转换到了R。若$S\to pR_1$，$S\to pR_2$，…，$S\to pR_n$，则简记为$S\to pR_1|pR_2|\cdots|pR_n$。

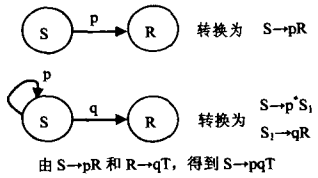

图4 两条转换规则和一条求解规则

根据给出的两条转换规则和一条求解规则，由SWLS的FSM模型可以求解出：

$$s_1\to(p_1(p_3c_1p_7)^*p_3c_1(p_4|p_4c_2p_5^+)c_3p_6)^*(p_1|p_1p_2)f$$

该转换表达了从初始状态s_1经过中间一系列状态，最终到达终止状态f。因此得到路径表达式$(p_1(p_3c_1p_7)^*p_3c_1(p_4|p_4c_2p_5^+)c_3p_6)^*(p_1|p_1p_2)$，其中$p^*$表示$p$在表达式中可以出现零次，出现一次，或出现多次，$p^+$表示$p$至少出现一次。符号·表示连接操作(concatenation)，通常被省略，即S

$\rightarrow p \cdot R$ 就是 $S \rightarrow pR$。符号|表示或操作(or),即 $S \rightarrow pR|qQ$ 就是 $S \rightarrow pR$ 或 $S \rightarrow qQ$。操作*和$^+$有相同的优先级,比连接操作·和或操作|要高,而连接操作·的优先级又高于或操作|。这些操作都是左结合的。由路径表达式,可以推导出以下10条可能的路径:

(1) p_1

(2) $p_1 p_2$

(3) $p_1 p_3 c_1 p_4 c_3 p_6 p_1$

(4) $p_1 p_3 c_1 p_4 c_2 p_5{}^+ c_3 p_6 p_1$

(5) $p_1 p_3 c_1 p_7 p_3 c_1 p_4 c_3 p_6 p_1$

(6) $p_1 p_3 c_1 p_7 p_3 c_1 p_4 c_2 p_5{}^+ c_3 p_6 p_1$

(7) $p_1 p_3 c_1 p_4 c_3 p_6 p_1 p_2$

(8) $p_1 p_3 c_1 p_4 c_2 p_5{}^+ c_3 p_6 p_1 p_2$

(9) $p_1 p_3 c_1 p_7 p_3 c_1 p_4 c_3 p_6 p_1 p_2$

(10) $p_1 p_3 c_1 p_7 p_3 c_1 p_4 c_2 p_5{}^+ c_3 p_6 p_1 p_2$

可以看出,路径(1)的长度小于2,不是测试路径。路径(7)、(8)、(9)和(10)分别包含路径(3)、(4)、(5)和(6),将它们作为了自己的子路径,此外,路径(2)也是它们的子路径,所以路径(7)、(8)、(9)和(10)不是测试路径。而路径(2)、(3)、(4)、(5)和(6)不包含其它的路径,不将其它路径作为它们的子路径,也不包含多余的迁移,因此它们是所求的5条测试路径。

注意,测试路径中可能出现 p^+(即至少出现一次 p),这好像和测试路径中"最短"的概念相违背,我们认为这是一种特殊情况并保留这种表示形式。如,在得到的(4)和(6)这两条测试路径中出现 $p_5{}^+$,表示用户在页面 p_5 可能先有多次的网页内导航(由于 p_5 太长)以浏览不同的部分,然后再转到下一个页面。此外,从路径表达式推导出的路径往往有无数条,我们选取那些可能产生测试路径的路径。对一个Web应用来说,这样的路径是有限的而且容易产生,只要避免出现多余的迁移,即,让路径表达式中的*操作所代表的迁移序列分别出现零次或一次即可,不考虑两次或两次以上的情况,因为这样的路径不可能是测试路径。假定存在路径表达式 $a(bcd)^* ea$,我们选取迁移序列"bcd"分别出现零次或一次,就得到两条路径 aea 和 $abcdea$,这两条路径才有可能是测试路径。若迁移序列"bcd"出现两次,得到的路径 $abcdbcdea$ 不可能是测试路径。

4.2 测试模型

要对Web应用进行测试,必须产生测试用例(test case)。一个测试用例是一条测试路径,并赋予测试路径中的每个对象以一定的输入值。前面得到的FSM模型中的卫 $g(\in G)$ 表达了用户在每个对象中潜在的输入信息。我们利用S. Elbaum等[10]提出的用户会话数据(user session data)得到输入信息,该信息是从HTML表单捕获的,是"名字-值"对的形式。同时结合J. Offutt等[11]提出的输入单元(input unit)作为测试数据。

在本文的Web测试模型中,测试用例进一步转换为测试规格说明(test specification)。一个测试规格说明是由测试套件(test suite),测试用例和测试步(test step)组成的一个层次结构。在测试规格说明中,一个测试套件是若干个完成某一特定测试功能的测试用例集,一个测试用例是测试步的树型结构,而一个测试步是可以直接执行测试的。我们的测试规格说明基于请求规格说明(request specification)、响应规格说明(response specification)和谓词规格说明(predicate definition),并采用XML的语法,是文献[3]的一个扩展。请求规格说明定义了HTTP请求的模式,响应规格说明指定了同一个测试步中,对应于HTTP请求的HTTP响应的断言,谓词规格说明定义了测试结果的断言。

下面是登录页p3关于请求和响应规格说明的一个可能的测试步:

```
〈request url = "http://simplewebloginsystem/login.asp"〉
    〈parameter name = "name" value = "computer"/〉
    〈parameter name = "password" value = "hello"/〉
〈/request〉
〈response〉
    〈match op = "contains" regexp = false select = "/html/body"
value = "Login Error!"/〉
〈/response〉
```

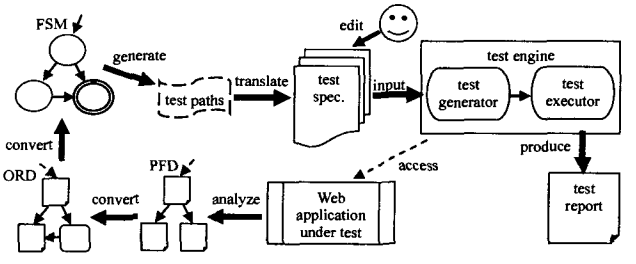

图5 Web测试模型

我们的Web测试模型扩展了X. Jia等[3]提出的模型,如图5所示。最开始,使用Web应用常用的需求和分析方法得到页面流图,分析页面流图得到ORD,进一步转化为FSM模型。通过FSM模型自动产生测试路径,进而转化为XML语法结构的测试脚本框架(即测试规格说明框架)。该框架定义了测试套件(标记〈testsuite〉),测试用例(标记〈testcase〉)。每个测试用例是多个有序的测试步(标记〈teststep〉),每个测试步测试一个页或一个软件组件。测试步定义了HTTP请求(标记〈request〉)、期望的响应(标记〈response〉)和带有条件定义的谓词(如〈not〉,〈and〉,〈or〉,〈match〉和标记〈forall〉)。其它可能的元素如参数(标记〈parameter〉)可以通过分析FSM模型来定义。测试脚本框架不包含输入变量的值,这些值会在以后追加。某些输入信息必须手动创建,如用户id和口令。其它的输入可以从HTML文件中自动获取或随机产生。此外,测试脚本框架不包含硬编码的期望输出,也就是说,Web测试模型定义了带有空XML标记〈response〉的框架,因为其响应信息将由测试人员手动添加。

在对测试脚本框架的必要信息进行了编辑之后,测试人员得到一个具有XML语法的形式化的测试规格说明,它作为测试引擎(test engine)的输入。测试规格说明包含测试用例模板。测试引擎中有一个测试生成器(test generator),它能够从测试规格说明中获取测试路径并产生测试用例(假如指定了测试准则)。测试引擎的测试执行器(test executor)接着执行测试用例并用测试喻(test oracle)验证其结果。也就是说,测试执行器可以把测试用例的测试序列提交给Web服务器,并为每个表单赋予合适的输入值。执行以后,测试引擎产生一个测试报告(test report)概括所有测试用例的测试结果。对于这种评估方法,测试人员只要在Web浏览器打开输出页,并检查每个给定输入的输出结果是否正确。在回归检查中,不再需要用户的干预,因为测试喻是在前面的测试迭代(iteration)中产生的(且经过了手工检查)。当然,万一有差别,仍然需要用户的干预。这样,测试引擎就表现为如同一个

(下转第291页)

```
if r1> =bestprice1 then
{transcationman1= transcationman[index];bestprice1=r1;}
if r2> =bestprice2 then
```

图5 加干扰码后买化妆品代码

比如我们想买一化妆品(cosmetic),想问其价格,部分程序如图4。

显而易见,这个代理易被恶意代理攻击得到它收集的最好价格150和提供商是transcationmanA。我们用意图扩大方法加入问效果(effect)和问见效时间(effecttime)等干扰码,防止恶意代理知道我们的真实意图,部分程序如图5。

采用意图分割方法同样能达到隐藏代理真实意图不被恶意代理攻击的目的,本文略。

结束语 以Agent为中介的电子商务是一种全新的在线商店模式,将Agent技术用于电子商务领域是当前的研究热点。恶意Agent带来的灾害比PC病毒更严重,因此能够检测并防御恶意Agent的电子商务体系是非常安全而吸引顾客的。

本文提出了一种适用于当前电子商务市场的SCAAM和实现策略,它能够解决以Agent为中介的电子商务中的恶意Agent危害问题,使电子商务交易公平和完整可靠。

参考文献

[1] 阿瑟.斯加利,威廉,等.B2B交易场:电子商务第三次浪潮.现代出版社,2001

[2] 梅绍祖,吕殿平.电子商务基础.清华大学出版社,2000

[3] Hindriks K V, de Boer FS, van der Hoek W, et al. Agent Programming with Declarative Goals. Intelligent Agents VII: Agent Theories Architectures and Languages, July 2000

[4] 祁明,卓光辉.多智能代理网络购物系统的设计与分析.计算机工程与设计,2001(6)

[5] Caglayan A, Harrison C. Agent Sourcebook. Wiley Computer Publishing, John Wiley & Sons, Inc., 1997

(上接第281页)

访问Web应用的客户端。

结束语 Web应用在全球范围内得到快速的发展,许多组织将Web应用集成到业务关键型的系统中,如电子市场、电子银行、电子贸易以及公共管理服务。由于Web应用的这种广泛而迅猛的发展,使得对其严格的质量需求变得非常紧迫,Web应用的测试为满足这种需求扮演着越来越重要的角色。然而,测试是一个非常耗时、耗财以及耗可计算资源的过程,其有用性可能受到用户交互的限制。显然,实用的测试是使其过程自动化。我们对Web应用建模并采用形式化的测试方法对其进行确认与验证,整个过程很少人工干预,基本实现了自动化。

Web应用测试的主要目的是发现需求中的错误,验证和已定义需求中规定的应用行为的一致性。本文所做的工作是独特的,其主要贡献如下:

(1)使用形式化的建模工具FSM精确地捕捉和定义Web应用的结构和行为;

(2)提出了将ORD转化为FSM模型的算法;

(3)根据给出的转换规则和求解规则,基于FSM模型,提出了一种有效的测试路径自动生成方法,这些测试路径可以转化为XML语法的测试规格说明;

(4)从Web应用的需求本身出发,对其进行分析和建模,提出了一种可行的Web测试模型,根据该模型,最终可以产生测试报告。

对于一个大的Web应用,可以把其分解(比如根据功能、结构或导航行为)成多个子Web应用,然后针对每个子Web应用利用本文提出的方法分别进行建模和测试。当然,各个子Web应用之间的关系也须加以考虑,对此我们将另文给出。

下一步的工作是对本文的建模与测试方法加以改进,进一步提升其自动化程度,并构建一个原型工具实现提出的Web测试模型。

参考文献

[1] Stout G A. Testing a Website: Best Practices. A Whitepaper. http://www.reveregroup.com.

[2] Hieatt E, Mee R. Going Faster: Testing the Web Application. IEEE Software, 2002, 19 (2): 60-65

[3] Jia X, Liu H, Qin L. Formal Structured Specification for Web Applications Testing. 2003 Midwest Software Engineering Conference, Chicago, USA, June 2003

[4] Andrews A, Offutt J, Alexander R. Testing Web Applications by Modeling with FSMs. Software and Systems Modeling, 2004

[5] Kung D C, Liu C H, Hsia P. An Object-Oriented Web Test Model for Testing Web Applications // Proceedings of the 1st Asia-Pacific Conference on Web Applications (APAQS 2000). IEEE Press, New York, USA, 2000: 111-120

[6] Benedikt M, Freire J, Godefroid P. VeriWeb: Automatically Testing Dynamic Web Sites // Proceedings of 11th International World Wide Web Conference. Honolulu, HI, May 2002

[7] Lucca G A D, Fasolino A R. Testing Web-based Applications: The State of the Art and Future Trends. Information and Software Technology, 2006 (48): 1172-1186

[8] Ackroyd M. Object-oriented Design of a Finite State Machine. Journal of Object-Oriented Programming, June 1995: 50-59

[9] van Gurp J, Bosch J. On the Implementation of Finite State Machines // Proceedings of the 3rd Annual IASTED International Conference on Software Engineering and Applications. Scottsdale, Arizona, USA, Oct. 1999: 172-178

[10] Elbaum S, Karre S, Rothermel G. Improving Web Application Testing with User Session Data // Proceedings of the 25th International Conference on Software Engineering. Portland, Oregon, May 2003: 49-59

[11] Offutt J, Wu Y, Du X, et al. Bypass Testing of Web Applications // Proceedings of the 15th IEEE International Symposium on Software Reliability Engineering. Saint-Malo, Bretagne, France, Nov. 2004