

文章编号:1001-9081(2009)03-0695-04

一种基于事件的 Web 程序测试模型

傅鹤岗, 陆艳军, 曾 刚

(重庆大学 计算机学院, 重庆 400030)

(luyj84@163.com)

摘 要:基于状态转换的测试方法是探测 Web 程序动态行为的有效途径。针对 Web 应用中复杂的动态页面, 通过改进现有 Web 测试方法, 提出了一种新的基于事件的测试模型, 用 Web 关系图(WARD)描述 Web 系统的结构, 用基于事件的状态迁移图(SMD)描述复杂动态页面的内部结构, 两者结合对 Web 应用系统全面建模。在实际 Web 系统中的实验证明该方法的建模是全面准确的。

关键词:Web 测试; 模型; Web 关系图; 状态迁移图; 事件

中图分类号: TP311.5 **文献标志码:** A

Web application test model based on action

FU He-gang, LU Yan-jun, ZENG Gang

(College of Computer Science, Chongqing University, Chongqing 400030, China)

Abstract: The testing method based on state transition is an effective way to test the dynamic behavior of Web application. By improving the existing Web testing method, this paper proposed a new testing model based on action to test complex dynamic Web pages in Web application system. The methods consisted of two steps: 1) describing the system structure of Web application with WARD; 2) describing the inner page structure of complex dynamic Web page with action-based SMD. At last, the results above were combined to model the whole Web application. Experiment on practical Web applications approves that the method is comprehensive and accurate.

Key words: Web testing; model; Web application relation diagram; state migration diagram; action

0 引言

随着网络的飞速发展和广泛应用, 网页数量正以指数级飞速增长, Web 程序的可用性、可靠性、互操作性和安全性等问题越来越受人们关注, 并提出了日益严格的要求^[1]。传统软件测试和技术很难实现对具有独特性质的 Web 程序的测试^[2], 探索适合于 Web 程序的测试方法和体系是当今软件界极具挑战性的课题。目前, 国内外已经对 Web 程序测试进行了大量的研究, 取得了一些初步的研究成果, 并形成了一些方法和工具。

对象状态测试是软件测试的一种重要途径^[3-4], 在将状态转换测试方法应用于 Web 程序过程中, Turine 等人提出了基于状态图的超文本模型^[5]。随后, Leung 等人^[6]对该方法进行了改进, Li 和 Conallen 等人则利用 UML 给 Web 程序建立导航模型^[7]。这些方法只能进行简单的动态可达性测试, 并未全面对 Web 程序的动态行为进行检查。

Kung 等人提出的 Web 测试模型 (Web Test Model, WTM), 从页面的状态依赖和导航行为两个方面对 Web 程序的行为特征进行了描述^[8], 但未给出状态和事件的确定方法, 缺乏实用性。

卢虹等人则对代表 Web 数据交互的会话进行分析, 用由状态变量组成的状态向量表示 Web 程序的状态从而进行状态转换测试^[9]。毛澄映等人综合运用导航行为和数据交互来描述程序的状态, 定义状态间的转换^[10]。这两种方法都强调整个 Web 程序的状态, 忽略了动态页面内部的状态切换。

针对现有的测试模型只考虑 Web 页面之间的数据交互

而忽略了 Web 页面事件引起内部状态变化, 本文提出一种新的 Web 测试模型, 重点考虑单页面内在事件驱动下的状态变迁, 应用于当前技术条件下 Web 应用系统, 特别是 Web 管理信息系统的测试建模需要。本方法在重庆电力公司电力调度系统中提出, 并得到了应用和完善。

1 基于事件的 Web 测试模型

在当前 Web 技术支持下的 Web 程序不仅能静态或动态显示数据, 还能通过复杂的动态 Web 页面完成许多应用程序的工作, 其中比较典型的 Web 系统有 Web 管理信息系统。Web 管理信息系统由数量有限的复杂动态页面组成, 且 Web 页面间的公有变量较少。针对上述特点, 本文提出了重点分析复杂动态页面的 Web 测试模型。为了全面准确地建立 Web 测试模型, 采用“分而治之”的思想, 以 Web 页面为中心将 Web 应用系统分割成若干个子集 (称为 Web 子集), 子集中包含一个 Web 页面, 以及这个 Web 页面事件用到的方法、服务器端对象, 其中一些公用的方法和服务器对象可以同时划分给多个的 Web 子集。该模型由两部分组成: 一个 Web 应用关系图 (Web Application Relation Diagram, WARD) 以及多个基于事件的 Web 状态迁移图 (State Migration Diagram, SMD)。前者是由 Web 导航图改进而来, 它用于描述 Web 子集之间链接关系及数据交互情况; 后者描述单个 Web 子集内部的事件以及状态变迁情况。

1.1 Web 应用关系图

Web 子集是以 Web 页面为核心, 其他资源都是为页面服务, 用页面名称标记子集。Web 子集通过 Web 页面的超链接

收稿日期: 2008-09-23; 修回日期: 2008-11-12。

作者简介: 傅鹤岗 (1950-), 男, 重庆人, 副教授, 主要研究方向: 软件工程、电子商务; 陆艳军 (1984-), 男, 湖北咸宁人, 硕士研究生, 主要研究方向: Web 技术、软件测试; 曾刚 (1981-), 男, 四川南充人, 硕士, 主要研究方向: 软件测试。

关系联系起来,构成 Web 系统。为了描述 Web 子集之间的关系,对导航图 PND^[8]进行改进,在导航关系中加入参数集表示子集之间的数据交互,形成 Web 应用关系图(WARD)。

定义 1 $WARD = (V, v_0, L, E)$ 是一个有向图。 V 表示 Web 子集的节点的有限集; $v_0 (v_0 \in V)$ 是 WARD 的起点; L 表示导航关系类型的有限集; $E \subset V \times L \times V$ 表示 Web 子集之间关系的边的有限集, $e \in E, e = (v_i, l_j, v_k)$,表示从 v_i 到 v_k 存在一条 l_j 类型的有向边, $v_i \in V, v_k \in V, l_j \in L$ 。

根据页面之间的参数情况,对导航关系进行分类。

定义 2 Web 页面 A 可以用超链接链到 Web 页面 B ,则 A 与 B 存在如下关系:

- 1) 若 A 与 B 之间无参数传递,则称 A 与 B 之间存在跳转关系或 A 可跳转到 B ,记为 $Lt = (t)$;
- 2) 若 A 向 B 传递了参数集 $T_1 (T_1$ 不为空),而 B 执行后未向 A 传递参数,则称 A 与 B 之间存在重定向关系或 A 可重定向到 B ,记为 $Lc = (c, T_1)$;
- 3) 若 A 向 B 传递了参数集 $T_1 (T_1$ 可为空),而 B 执行后向 A 传回了参数集 $T_2 (T_2$ 不为空),则称 A 与 B 之间存在调用关系或 A 可调用 B ,记为 $Ld = (d, T_1, T_2)$ 。

算法 1 为构建 WARD 的算法主要步骤。构建 WARD 的过程中,可以找出系统中无法到达的页面,经过动态分析,删除和添加所需要的边。利用 WARD,可以生成系统测试用例,完成系统测试。

算法 1 WARD 构建算法

输入:Web 子集集合 V ,其中起始页面所在子集为 v_0 ;可达子集集合 S , S 的初始值为 $\{v_0\}$;Web 子集超链接集 H , H 的初始值为空;边集 E ,初始值为空。

输出:边集 E 。

算法如下:

- 1) 从 S 中选择一个可达的 v_i ,遍历 v_i 找出 v_i 中存在的所有超链接,加入 H ;
- 2) 若 H 为空,转 6);
- 3) 在 H 中,选一个超链接 h , h 可达的页面 v_j 在 V 中,分析 h 的关系类型 l ,新建边 (v_i, l, v_j) ,并加入 E 中;
- 4) 若 v_j 在 S 中未曾出现过,则将 v_j 加入 S 中;
- 5) 从 H 中去掉超链接 h ,转 2);
- 6) 从 S 中移除 v_i ,若 S 不为空,则转 1);若 S 为空,则返回 E 。

1.2 Web 状态迁移图

Web 子集的核心是 Web 页面,Web 页面中存在大量的动态交互,伴随着 Web 事件执行,事件的执行调用 Web 子集中其他的资源。在对复杂的动态 Web 页面进行建模的过程中需要解决两个难题:一是过程复杂且数量巨大的动态交互;二是 Web 页面中事件执行过程中出现的函数嵌套调用。

本文基于事件对 Web 页面构建状态,将动态交互过程融合在 Web 状态中,并提出了函数调用表的概念,形成基于事件的状态迁移图(SMD)。从状态图中可以得到 Web 子集的测试事件序列,在事件中加入动态交互过程可得到 Web 子集的测试用例。基于事件的 Web 状态迁移图(SMD)的具体定义如下。

定义 3 $SMD = (S, s_0, s, A, T, R)$,其中: S 表示 Web 页面状态的有限集; $s_0 (s_0 \in S)$ 是状态图中的起点; $s (s \in S)$ 是状态图的终点; A 表示 Web 子集中事件的有限集,Web 页面有大量的事件, A 只包含 Web 页面中有响应函数的事件(文后提到的事件如未加说明则都指有响应函数的事件); T

为状态转换集, T 中元素为状态图的有向边; R 表示依赖事件对(见定义 6)的集合,记为依赖集。

基于事件的状态迁移图可以用正则表达式表示,并生成测试事件序列,添加具体参数就可以得到测试用例。

1.2.1 Web 子集状态集 S

Web 页面中显示信息部分由大量 Web 标签组成,每个标签又有多个属性,信息量非常大,这给 Web 页面状态的建模带来了很大的难度。Web 标签都有着“所见即所得”的特性,标签及其属性在 Web 页面加载到浏览器时,就可以完全显示,因此在对 Web 页面建模时只需关心部分标签的关键属性。另外,如果关注过多的标签及其属性,则 Web 状态会随着标签的个数呈几何增长,但在实际状态图中,状态的数量是有限的,因此必须控制抽取标签的数目。

本文描述状态时,只关注触发事件的标签。同时,标签能否触发事件,则由标签的一个或多个属性控制,但在 Web 程序中只会选择其中的一个属性控制条件。将控制事件触发的标签属性抽取出来形成状态变量,所有的状态变量共同决定 Web 页面的状态。

记状态集 $S = \{s_0, s_1, \dots, s_n, s\}$, n 为状态变量决定的状态的个数, s_0 中状态变量的取值均为默认值,为状态图的起始节点, s 为 S 中特殊状态,为状态图的终止状态。

1.2.2 事件集 A

Web 页面可以触发很多事件,但只有部分事件有响应函数,事件集 A 中保存事件及响应函数信息,由 Web 标签名、函数调用表和控制条件集共同组成。记 $A = \{a_1, a_2, \dots, a_n\}$,其中 $a_i (0 \leq i \leq n)$ 表示一个事件及其响应函数的信息,可定义如下。

定义 4 事件 a_i 是一个四元组 $a_i = (N, B, F, C, I)$,其中: N 表示事件名; B 表示触发事件的 Web 标签名; F 表示响应事件的函数调用表; C 表示事件的控制条件集(控制条件集中元素是由多个条件组成的向量,决定了事件响应函数执行结果); I 表示事件影响到的标签集。

一个事件可能有一个或多个响应函数,另外,函数中又会调用一些其他的函数,因此一个事件的响应中有一些函数嵌套执行。为了描述这些函数之间的关系,提出了函数调用表的概念,采用类似广义表的结构表示,定义如下。

定义 5 函数调用表 $F = (f_1, f_2, \dots, f_n)$, F 为函数调用表的名称, n 是它的长度, $f_i (1 \leq i \leq n)$ 为 F 直接调用到的函数或 F 中调用了函数的逻辑判断, f_i 有三种情况:

- 1) 若 f_i 表示函数且 f_i 中未调用其他的函数, f_i 值为函数名。
- 2) 若 f_i 表示函数且 f_i 中调用其他的函数,将调用的函数用函数调用表表示,记函数名为 f ,函数调用表为 B ,则 $f_i = fB$;
- 3) 若 f_i 表示逻辑判断,则用一个特殊的标记 $[\]$ 来表示,将逻辑判断所覆盖程序段的函数调用情况记为 $T = [t_1, t_2, t_3, \dots, t_n]$,其中 n 为逻辑判断分支数, $t_i (1 \leq i \leq n)$ 存在如下几种情况:①若 t_i 中只调用一个函数,记函数名为 f ,则 $t_i = f$;②若 t_i 中调用了多个函数,将这些函数用函数调用表表示,则 t_i 的值为函数调用表名;③若 t_i 中未调用函数,则记 $t_i = 1$,此时 $f_i = T$ 。

事件响应函数中存在逻辑判断,逻辑判断引起不同的程序分支执行,因此逻辑判断中的条件决定了事件响应函数的执行结果。对 Web 子集中事件进行分析时,必须对事件响应函数中的条件进行分析。

本文将事件响应函数中逻辑判断条件抽取出来,按照划

分等价类的方法,一个条件生成一个控制变量。例如:条件 $a \leq 0$ 对应的控制变量为 $\{a \leq 0, a > 0\}$, 条件 $a > b$ 对应的控制变量为 $\{a > b, a \leq b\}$ 。条件的个数决定了控制变量的个数,但是对于逻辑变量相同的条件,则可以按照等价类划分规则生成一个控制变量。例如:条件 $a > 0$ 和条件 $a < 6$ 对应的控制变量可以合并为一个 $\{a \leq 0, 0 < a < 6, a \geq 6\}$ 。所有控制变量的笛卡尔乘积得到的集合为控制条件集 C , 对于 C 执行路径相同的多个控制条件可以合并为一个, C 中的一个控制条件与事件响应函数的一条执行路径相对应。

事件执行过程,对 Web 页面的部分或者全部标签产生影响,包括获取标签数据或属性、向标签填充数据和修改标签属性等。标签集 I 的元素为这些受影响的标签。标签集 I 存在是为了便于检验事件的执行是否有效。

1.2.3 Web 子集转换集 T

记转换集 $T = \{t_1, t_2, \dots, t_n\}$, $t_i = (s_j, a, c, s_m)$ ($1 \leq i \leq n$, $s_j \in S$, $s_m \in S$, $a \in A$, $c \in C$, C_j 为起始状态, s_m 为结束状态, s_j 与 s_m 可以相同, a 为事件集 A 中的一个事件, c 为事件 a 一个控制条件), t_i 为状态图中的边或环, 环或边上标记事件名以及控制条件。若 T 中多个元素的起始状态、结束状态和事件均相同,则在 SMD 中,只用一条边表示,边上的多个控制条件由“/”分隔。

1.2.4 依赖集 R

Web 页面中事件与其响应函数存在映射关系。Web 标签属性可以静态指定映射关系,同时,程序也可通过操作 DOM 对象指定或移除映射关系。而 Web 程序都是以事件响应函数的方式存在,因此,一个事件可以添加或删除另一个事件与其响应函数的映射关系。为了描述事件之间的关系,提出了依赖事件对的概念,定义如下。

定义 6 依赖事件对 $r = (a, c, b, t)$, $a, b \in A$, c 为 a 的一个控制条件, t 为依赖关系的类型,有:

- 1) 若 a 中条件 c 对应的路径为事件 b 与其响应函数添加了映射关系,记为添加依赖 $t = 1$, 事件 b 的触发必须在事件 a 中条件 c 对应的路径执行之后;
- 2) 若 a 中条件 c 对应的路径将事件 b 与其响应函数的映射关系移除,记为移除依赖 $t = 0$, 事件 a 中条件 c 对应的路径执行之后,事件 b 不能被触发。

事件关系集 R 为依赖事件对的集合,依赖关系决定了测试事件序列中事件的先后顺序。事件关系集是生成测试事件序列的约束规则。

2 实验

重庆电力公司电力调度系统项目实现了 9 套主要功能相同但有细节差异的 Web 管理信息系统。该项目开发过程的特点是:根据国家电网公司电力调度操作规范开发模板系统,然后分地区实施,在实施中细化需求、完善差异性功能。这一特点使得在实施中有大量的回归测试,大大增加了软件的测试工作量,测试建模成为开发的重要环节,同时也为维护提供了依据。

2.1 WARD 实例

通过分析建模,得到了电力调度操作命令票模块的 WARD 图,如图 1 所示。该模块主要功能部分包括九个子集 $\{CZMLP. ascx, MlpNp. aspx, MlpSh. aspx, MlpZx. aspx, MlpAll. aspx,$

$MlpSample. aspx, MlpQuery. aspx, JXD. aspx, CZMLPPrint. aspx\}$ 。从系统主页 Home. aspx 通过跳转关系到达 CZMLP. ascx; CZMLP. ascx 页面是该模块的主控页面,可重定向到 MlpNp. aspx, MlpSh. aspx, MlpZx. aspx, MlpAll. aspx, MlpSample. aspx 中,并传入参数 objId,同时可跳转到 MlpQuery. aspx 中; MlpNp. aspx, MlpSh. aspx, MlpSample. aspx 均以调用 JXD. aspx, JXD. aspx 页面执行后传回参数 JXPBH; MlpZx. aspx, MlpAll. aspx, MlpSample. aspx 可重定向到页面 CZMLPPrint. aspx, 传入参数 objId, 动态生成页面 CZMLPPrint. aspx; MlpQuery. aspx 可重定向到 MlpNp. aspx, MlpSh. aspx, MlpZx. aspx, MlpAll. aspx, MlpSample. aspx, 并传入参数 objId。

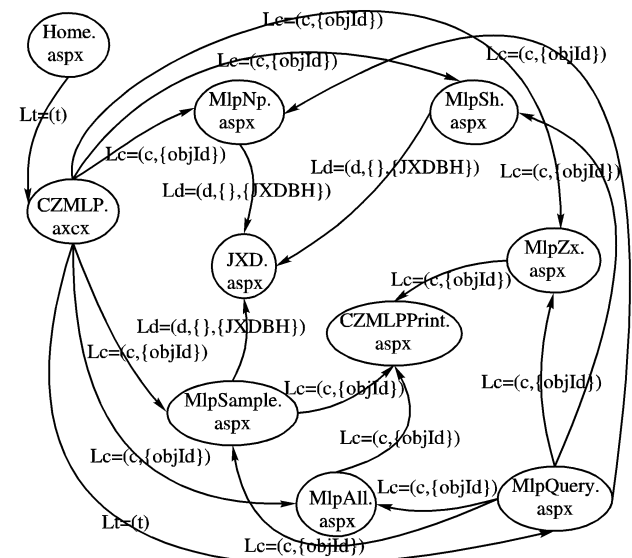


图1 电力调度操作命令票模块 WARD

2.2 SMD 实例

以操作命令票模块中典型操作命令票页面 MlpSample. aspx 所在的 Web 子集为实例,精简部分功能后进行分析处理,得到典型操作命令票页面的 SMD,如图 2 所示。

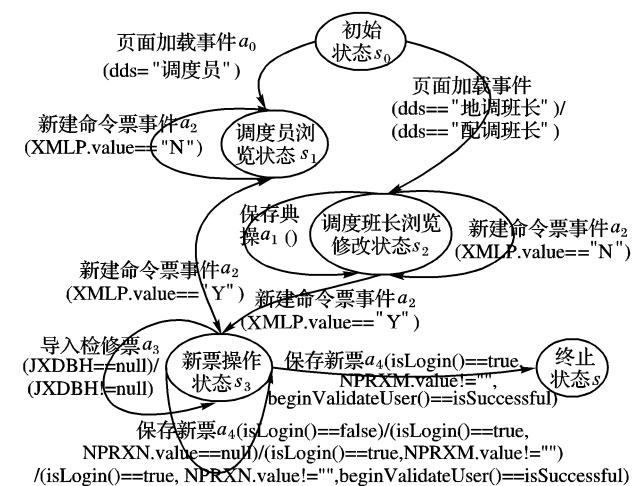


图2 电力调度操作命令票模块典型操作命令票页面子集 SMD

图2 生成过程如下:

1) 抽取状态集 S , 状态 s_0 (F 代表 false, T 代表 true, D 代表 disabled, V 代表 visible)。

起始状态 $s_0 = \{SJXD. D = F, XMLP. D = F, saveDC. V = F, saveMLP. D = F\}$; Web 页面加载 onload 函数之后状态 $s_1 = \{SJXD. D = D, XMLP. D = F, saveDC. V = T, saveMLP. D = D\}$; Web 页面加载 onload 函数之后状态 $s_2 = \{SJXD. D = D,$

XMLP. D = F, saveDC. V = F, saveMLP. D = D}; 新建典操事件之后的状态: $s_3 = \{ \text{SJXD. D} = \text{F}, \text{XMLP. D} = \text{D}, \text{saveDC. V} = \text{F}, \text{saveMLP. D} = \text{F} \}$ 。

状态集 $S = \{s_0, s_1, s_2, s_3, s\}$ 。

2) 抽取事件集 A , 如下所示:

$a_0 = (\text{页面加载事件}, \text{pageload}, (\text{Page_Load}, \text{webOnload}(\text{loadData}, \text{dcState}(\text{getJs}))), \{(\text{userjs} == \text{"地调班长"}), (\text{userjs} == \text{"配调班长"}), (\text{userjs} == \text{"调度员"})\}), a_1 = (\text{保存典操事件}, \text{savaDc}, (\text{savaDc}), \{ \}), a_2 = (\text{新建命令票事件}, \text{XMLP}, (\text{newMLP}), \{(\text{XMLP. value} == \text{"Y"}), (\text{XMLP. value} == \text{"N"})\}), a_3 = (\text{导入检修票}, \text{SJXD}, (\text{selectJXD}), \{(\text{JXDBH} == \text{null}), (\text{JXDBH!} = \text{null})\}), a_4 = (\text{保存新票}, \text{saveMLP}, (\text{save}([\text{saveMLP}], [\text{saveMLP}])), \{(\text{isLogin}() == \text{false}), (\text{isLogin}() == \text{true}, \text{NPRXM. value} == \text{null}), (\text{isLogin}() == \text{true}, \text{NPRXM. value} == \text{""}), (\text{isLogin}() == \text{true}, \text{NPRXM. value!} = \text{""}, \text{beginValidateUser}() != \text{isSuccessful}), (\text{isLogin}() == \text{true}, \text{NPRXM. value!} = \text{""}, \text{beginValidateUser}() == \text{isSuccessful})\})。$

事件集 $A = \{a_0, a_1, a_2, a_3, a_4\}$ 。

3) 状态转换集 T , 如下所示:

以状态 s_0 为起点的迁移有: $t_1 = (s_0, a_0, (\text{userjs} == \text{"调度员"}), s_1)$, $t_2 = (s_0, a_0, (\text{userjs} == \text{"地调班长"}), s_2)$, $t_3 = (s_0, a_0, (\text{userjs} == \text{"配调班长"}), s_2)$; 以状态 s_1 为起点的迁移有: $t_4 = (s_1, a_2, (\text{XMLP. value} == \text{"N"}), s_1)$, $t_5 = (s_1, a_2, (\text{XMLP. value} == \text{"Y"}), s_3)$; 以状态 s_2 为起点的迁移有: $t_6 = (s_2, a_2, (\text{XMLP. value} == \text{"N"}), s_2)$, $t_7 = (s_2, a_1, (), s_2)$, $t_8 = (s_2, a_2, (\text{XMLP. value} == \text{"Y"}), s_3)$; 以状态 s_3 为起点的迁移有: $t_9 = (s_3, a_3, (\text{JXDBH!} = \text{null}), s_3)$, $t_{10} = (s_3, a_3, (\text{JXDBH} == \text{null}), s_3)$, $t_{11} = (s_3, a_4, (\text{isLogin}() == \text{false}), s_3)$, $t_{12} = (s_3, a_4, (\text{isLogin}() == \text{true}, \text{NPRXM. value} == \text{null}), s_3)$, $t_{13} = (s_3, a_4, (\text{isLogin}() == \text{true}, \text{NPRXM. value} == \text{""}), s_3)$, $t_{14} = (s_3, a_4, (\text{isLogin}() == \text{true}, \text{NPRXM. value!} = \text{""}, \text{beginValidateUser}() != \text{isSuccessful}), s_3)$, $t_{15} = (s_3, a_4, (\text{isLogin}() == \text{true}, \text{NPRXM. value!} = \text{""}, \text{beginValidateUser}() == \text{isSuccessful}), s)$ 。

状态转换集 $T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{13}, t_{14}, t_{15}\}$ 。

4) 依赖集 R 。

典操子集的事件之间无依赖关系, 则 R 为空集。

5) 状态图 SMD 以及生成的测试事件序列。

典型操作票页面状态迁移图 SMD 如图 2 所示, 该图可以用正则表达式 $s_0((t_1 s_1(t_4 s_1)^+ t_5 s_3)/((t_2 s_2/t_3 s_2)(t_6 s_2/t_7 s_2)^+ t_8 s_3))(t_9 s_3/t_{10} s_3/t_{11} s_3/t_{12} s_3/t_{13} s_3/t_{14} s_3)^+ t_{15} s$ 表示, 根据一定测试标准生成测试用例集。

2.3 实验分析

实验表明, 与现有的建模方法相比较, 基于事件的建模方法针对 Web 系统构建一个 Web 应用关系图和多个 Web 状态迁移图, 有如下几个优点: 1) 系统级和页面级同时建模, 降低了建模的复杂度, 减小了建模难度, 提高了建模的可行性; 2) 同时考虑到系统级和页面级的问题, 构建的模型更加全面准确; 3) 页面级的建模适应了 Web 程序中动态页面越来越复杂的发展新趋势; 4) 从模型中可以得到系统级的测试用例,

也可以得到页面级的测试用例, 可用于系统测试、验收测试、回归测试、项目维护工作。

3 结语

基于事件的测试方法实现了对 Web 程序的全面建模, 适用于复杂的 Web 系统。该模型是在重庆电力公司电力调度系统项目中提出、应用和完善。基于事件的测试方法与其他测试方法相比较, 适应当前复杂的动态 Web 页面实现应用程序功能的新特点。本模型重在分析复杂的动态 Web 页面建模, 但对于简单的动态 Web 页面建模过于复杂, 对于简单动态 Web 页面占多数的 Web 系统不适用。下一步的工作, 修改方法中的状态迁移图, 使其适应更多 Web 系统的建模需要, 另外结合现有的测试工具, 进行 Web 自动化测试的研究。

参考文献:

- [1] LUCCA G A D, FASOLINIO A R, FARALLI F, *et al.* Test Web applications[C]// Proceedings of International Conference on Software Maintenance (ICSM'02). Montreal: IEEE Computer Society, 2002: 310-319.
- [2] 许蕾, 徐宝文, 陈振强. Web 测试综述[J]. 计算机科学, 2003, 30(3): 100-104.
- [3] CHOW T S. Testing software design modeled by finite-state machines[J]. IEEE Transaction on Software Engine, 1978, SE-4(3): 178-187.
- [4] KUNG D, SUCHAK N, GAO J, *et al.* On object state testing[C]// Proceedings of Computer Software and Application Conference (COMPSAC94). [S.l.]: IEEE Computer Society, 1994: 222-227.
- [5] TURIRE M A S, OLIZEIRA M C F, MASIERO P C. A navigation-oriented hypertext model based on statechart[C]// Proceedings of the 18th ACM Conference on Hypertext (Hypertext'97). New York: ACM, 1997: 102-111.
- [6] LEUNG K, HUI L, YIU S, *et al.* Modeling Web navigation by state-chart[C]// Proceedings of Computer Software and Application Conference (COMPSAC2000). Taipei: IEEE Computer Society, 2000: 41-47.
- [7] LI J F, CHEN P, CHEN J. Modeling Web application architecture with UML[C]// Proceedings of 36th International Conference on Technology of Object-Oriented languages and Systems. Washington, DC: IEEE Computer Society, 2000: 265-274.
- [8] KUNG D C, LIU C H, HSIA P. An object-oriented Web test model for testing Web applications[C]// Proceedings of Asia-Pacific Conference on Quality Software (APAQS2000). Washington, DC: IEEE Computer Society, 2000: 111-120.
- [9] 卢虹, 徐宝文. 一种 Web 应用的状态测试方法[J]. 计算机工程与应用, 2002, 38(2): 55-57.
- [10] 毛澄映, 卢炎生. 基于状态转换的 Web 程序测试方法研究[J]. 计算机科学, 2005, 32(5): 219-223.
- [11] YANG J T, HUANG J L, WANG F J, *et al.* An object-oriented architecture supporting Web application testing[C]// Proceedings of the Twenty-Third Annual International Computer Software and Applications Conference (COMPSAC'99). Washington, DC: IEEE Computer Society, 1999: 122-127.
- [12] MARCHETTO A, TONELLA P, RICCA F. State-based testing of Ajax Web applications[C]// Proceedings of IEEE International Conference on Software Testing, Verification, and Validation. Washington, DC: IEEE Computer Society, 2008: 121-130.