Web软件的一种有效测试方法

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

摘 要 测试Web软件面临极大的挑战。从构造Web软件的页面流图出发,提出了一种测试路径生成的方法,以一个简单的Web登录系统SWLS(Simple Web Login System)为例对该方法进行了阐述,并给出了Web软件测试的一种有效模型。该方法给页面流测试技术提供了一个有意义的基础。

关键词 Web软件,页面流图,页面测试树,测试路径,Web测试模型
中图法分类号 TP311 **文献标识码** A

Efficient Web Software Testing Method

QIAN Zhong-sheng[1] MIAO Huai-kou[2]
(School of Information Technology, Jiangxi University of Finance & Economics, Nanchang 330013, China)[1]
(School of Computer Engineering & Science, Shanghai University, Shanghai 200072, China)[2]

Abstract Testing Web software faces great challenges. Starting from constructing the PFD(Page Flow Diagram) for Web software, this work proposed a test path generation approach, which was illustrated by the SWLS(Simple Web Login System) as an example and presented an effective Web testing model for Web software testing. This method provides a meaningful basis for page flow testing technology.

Keywords Web software, Page flow diagram, Page test tree, Test path, Web test model

1 引言

Web测试是保证Web软件质量的一种有效技术。然而,由于Web软件的特殊性和复杂性,很难直接使用传统的测试理论和方法。

典型地,Web软件比其它软件系统维护更频繁,这种维护通常由许多小的增量更新构成[1]。为了适应这种变更,Web测试方法必须自动化,且测试集也是自适应的。然而,Web软件带来重要而具有挑战性的测试问题,这些问题又不能直接用已有的对传统程序测试的方法来解决[2,3]。我们测试Web页面之间的导航,至于Web的表示细节,如页面元素的大小、位置以及颜色等不是本文的讨论范围。在测试时,Web软件内部的页面是受关注的,首先要加以考虑。这样,Web软件的测试范围就限制在内部页面以及任何直接由内部页面链接的页面中。任何链接到内部页面的页面不被考虑,因为要识别哪些页面是链接到内部页面是不可能的。不失一般性,这里把链接(link)和按钮(例如submit和reset)统称为链接(link)。我们的测试过程就是沿着页面测试树(边表示链接,结点表示页面)的一条路径往下走,直到叶子为止。该方法和已有方法的一个关键不同点是,这里提出了树的构造,而不是图的表示,有效地避免了可扩展性问题。

2 相关工作

传统的针对Web软件的测试[4,5]一般分为功能测试(如链接测试)、性能测试(如负载测试)、可用性测试(如导航测试)、客户端兼容测试以及安全测试等方面,这些测试主要还是处于手工阶段,依赖于测试者的经验。

Subraya等[6]提出了对象驱动的性能测试。他们将Web软件的行为分解为可测试的组件。与其不同,我们的方法是根据Web软件的功能需求而非行为来划分。

Andrews等[7]提出了从FSM导出测试用例的方法。该方法试图利用输入约束限制状态空间爆炸。理论上,Web软件可以完全用FSM建模,然而,即使是简单的Web软件也会遇到状态空间爆炸的难题。我们的方法以一种"分而治之"的方式处理Web软件,将被测Web软件(WAUT)分解成许多子Web软件。若分解合理,则每个子Web软件不会引起状态空间爆炸问题。

我们提出了使用用户会话日志的数据产生Web软件测试用例的方法[8],并给出了一种测试优化技术。用户会话日志的数据是通过HTML表单捕获的,记录在Web服务器上。本文的方法是灵活的,用户输入数据可以通过任何已有的方法获得。

Benedikt等[9]开发了一个动态导航工具VeriWeb,该工

到稿日期:2010-03-15 返修日期:2010-06-21 本文受国家863项目(2007AA01Z144),江西省自然科学基金资助项目(2010GQS0048),江西省教育厅科技计划项目(GJJ10120,GJJ10117),上海市重点学科建设项目(J50103)以及江西财经大学校级课题(04722015)资助。
钱忠胜(1977-),男,博士,讲师,主要研究方向为Web软件的建模与测试、软件形式化方法,E-mail:changesme@163.com;缪淮扣(1953-),男,教授,博士生导师,主要研究方向为软件自动化、软件形式化方法。

具从一个给定的 URL 出发，通过非确定的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 PTT，每个测试用例都是测试步的树型结构。

Reza 等[10]分析了模型驱动的测试方法，讨论了基于状态图(statechart)的 Web 软件测试，但测试执行的自动化层次并不清晰。邓小鹏等[11]从体系结构、实现技术、组成成分、运行机制、运行环境、开发设计等方面分析了影响 Web 软件测试的因素，从模型驱动的测试、基于 Agent 的测试、WS 及 SOA 测试、性能测试等方面探讨了 Web 软件测试未来的研究方向。

3 测试路径的生成

Web 软件为用户提供了多种不同的使用方式，每个用户在浏览时都有自己独特的访问顺序。另外，用户不仅仅是浏览，还会触发操作和处理事务。通过超链接，用户可以在不同的页面间跳转。然而，一个悬挂(dangling)链接或一个不可达的页面会使用户感到迷惑。并且，Web 软件的结构通常是复杂的，也难以维护。为了确定页面流是否是合适的，是否满足需求，我们使用了页面流图(PFD, Page Flow Diagram)。PFD 直观地反映了 Web 软件中页面间的关系。

PFD 是一个三元组 (P,L,F)，其中 P 是页面的集合，包括 Web 软件中的页面和直接由它们链接的页面；$L(\subseteq P\times P)$ 是链接的集合；$F(\subseteq P\times L\times P)$ 是页面流的集合。图 1 是一个 Web 软件的 PFD 的例子(进入默认页的第一个链接用虚箭头表示)。

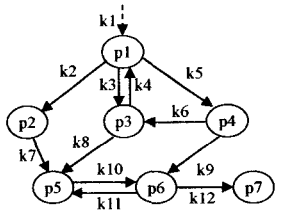

图 1 一个 Web 软件的 PFD 示例

PFD 是一个图形结构，这带来一些问题，例如，很难判断一条路径是否终止，且该路径可能存在循环，这就使得测试过程变得复杂。可以看出，基于 PFD 的测试过程是复杂和难以处理的。因此，我们使用基于 PFD 的页面测试树(PTT, Page Test Tree)进行测试。根据 PTT，可以产生比 PFD 更短的路径，但不丢失页面和链接的覆盖。PTT 是从 PFD 构造的一个生成树，构造方法见算法 1。

算法 1 PFD2PTT

输入：Web 软件的一个 PFD

输出：从 PFD 得到的一个 PTT

begin

(1) 把 PFD 的初始页标识符放入 FIRST 表；

(2) 若 FIRST 表为空，则转(6)；

(3) 从 FIRST 表选择第一个由 pid 标记的初始页标识符。若 pid 在 SECOND 表中，则转(5)；否则，把它加入 SECOND 表的末端；

(4) 若 pid 链接着其它页面，则

- 若其它的某些页面标识符出现在 FIRST 表或 SECOND 表中，则生成它们的复本；
- 保持 pid 和其它页面(或它们的复本)在 PFD 中的链接关系，且
- 把其它页面(或它们的复本)的页面标识符加入到 FIRST 表的末端；

(5) 从 FIRST 表中删除 pid，然后转(2)；

(6) 输出 PTT，该 PTT 仅保留了(4)中被保持的链接关系。

end

在算法 PFD2PTT 中，我们使用了 FIRST 和 SECOND 这两张表。FIRST 表中存放页面(结点)标识符，它们没有做标记(unmarked)；SECOND 表也是存放页面(结点)标识符，但它们是做了标记的(marked)。FIRST 表和 SECOND 表初始化为空。注意，页面标识符和它们的拷贝允许同时存在于 FIRST 表中。

假定一个 PFD 中有 n 个链接，我们可以使用 PFD2PTT 算法在 $n+1$ 步内得到对应的 PTT。图 2 是从图的 PFD 产生的 PTT，其中，结点和边分别表示 PFD 中的页面和链接(加了下划线的结点是相应结点的拷贝)。根据 PFD2PTT 算法，在 FIRST 表和 SECOND 表中的页面(见表 1)是随着产生过程不断变化的。在表 1 中，每个加了下划线的页面都是前面步骤中导出的相应页面的拷贝。Ø 表示 FIRST 表或 SECOND 表是空的。

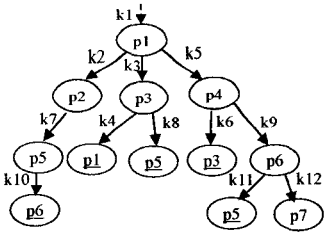

图 2 由图 1 的 PFD 构造的 PTT

表 1 使用 PFD2PTT 算法后 FIRST 表和 SECOND 表中每步的页面变化

step	FIRST	SECOND
1	p1	Ø
2	p2,p3,p4	p1
3	p3,p4,p5	p1,p2
4	p4,p5,p1,p5	p1,p2,p3
5	p5,p1,p5,p3,p6	p1,p2,p3,p4
6	p1,p5,p3,p6,p6	p1,p2,p3,p4,p5
7	p5,p3,p6,p6	p1,p2,p3,p4,p5
8	p3,p6,p6	p1,p2,p3,p4,p5
9	p6,p6	p1,p2,p3,p4,p5
10	p6,p5,p7	p1,p2,p3,p4,p5,p6
11	p5,p7	p1,p2,p3,p4,p5,p6
12	p7	p1,p2,p3,p4,p5,p6
13	Ø	p1,p2,p3,p4,p5,p6,p7

实际上，PTT 是一棵表达了页面顺序关系的树。

定义 1 后继函数 $И: P\to 2^P$ 定义了直接被某个页面链接的页面集。若页面 p 不链接任何其它的页面，则 $И(p)=\varnothing$。若 $p1$ 链接 $p2$，则 $p2\in И(p1)$，称 $p1$ 是 $p2$ 的前页(previous page)，$p2$ 是 $p1$ 的后页(next page)。自反传递闭包 $И^*$ 表达了页面间的顺序关系：$И^*(p)=\bigcup_{i\geqslant 0}И^i(p)$ 使得 $И^0(p)=\{p\}$，$И^1(p)=И(p)$。对 $\forall i\geqslant 1$，$И^{i+1}(p)=\bigcup_{q\in И(p)}И^i(q)$。若 $p2\in И^*(p1)$，则 $p1$ 是 $p2$ 的祖先(precedent page)，$p2$ 是 $p1$ 的后裔(subsequent page)。

定义 2 在 PTT 中，初始的页面(结点)称为根页(结

点)，进入根页的链接称为根链接，叶子页面(结点)称为尾页，进入尾页的链接称为尾链接，所有其它的页面(链接)称为分支页(链接)。一个尾页也可能是一个分支页或根页。

定义 3 一条路径是导航时从根链接开始的、结束于任何链接的可能的链接序列。路径可表示为 $k1\rightarrow k2\rightarrow\cdots\rightarrow kn$，其中 ki 是一个链接($1\leqslant i\leqslant n, n>0$)。

定义 4 一个测试路径是从根链接开始到尾链接结束的路径。

PTT 被用来建立路径表达式[12]，接着用测试翻译器(test translator)来转换成测试规格说明(见第 5 节)。路径表达式是一棵树中路径的代数表示，其中的变量是链接，它们可以通过操作算子"+"和"*"结合，分别表示选择和循环，括号可以组合子表达式。在图 2 中，一个路径表达式的例子是 $k1\rightarrow k3\rightarrow(k4+k8)$。对于此路径表达式，在链接 $k3$ 处遇到了选择，分别是 $k4=(p3,p1)$ 和 $k8=(p3,p5)$。一个 Web 软件的路径表达式的计算可以通过 Beizer 设计的 Node-Reduction 算法得到[12]。由于路径表达式直接反映了树中所有的路径，它们可被用来生成满足任何覆盖准则的链接序列。为了满足给定的准则，从路径表达式中选择最小数量的测试路径通常是一个繁琐的工作。然而，可以利用启发式算法计算最小值的近似解。

测试路径可以从路径表达式中自动地获取，然后得到测试用例(见第 5 节)。图 2 的 PTT 对应的完整路径表达式是 $k1\rightarrow(k2\rightarrow k7\rightarrow k10+k3\rightarrow(k4+k8)+k5\rightarrow(k6+k9\rightarrow(k11+k12)))$。在该路径表达式中，有 6 条测试路径，因为在 PTT 中存在 6 条尾链接。这些测试路径是 $k1\rightarrow k2\rightarrow k7\rightarrow k10$，$k1\rightarrow k3\rightarrow k4$，$k1\rightarrow k3\rightarrow k8$，$k1\rightarrow k5\rightarrow k6$，$k1\rightarrow k5\rightarrow k9\rightarrow k11$ 和 $k1\rightarrow k5\rightarrow k9\rightarrow k12$。在这些测试路径中，从根页到某个尾页产生不止一条路径是可能的。例如，路径 $k1\rightarrow k3\rightarrow k8$ 和 $k1\rightarrow k5\rightarrow k9\rightarrow k11$ 是正常导航时从 $p1$ 到 $p5$ 的两条测试路径。注意，路径 $k1\rightarrow k2\rightarrow k7$ 不是从 $p1$ 到 $p5$ 的测试路径，因为 $k7$ 不是该路径中的尾链接。

在 PTT 中，从根链接到每个尾链接的所有路径覆盖了所有的链接，称为全链接覆盖(full links coverage)。若从树中去除所有加下划线的结点和指向它们的相应边，则我们得到最小数量的测试路径，但覆盖了所有的页面，称为全页面覆盖(full pages coverage)。全页面覆盖测试是否所有的页面都可以被访问。显然，全链接覆盖的测试路径包含全页面覆盖的路径。

Web 软件通常包含大量的页面和链接，而页面又可以被各种链接以任意的方式连接，这使得 Web 软件极其复杂。而且 PFD 和 PTT 由于很大而不好处理。因此，对于一个非平凡的 Web 软件来说，穷尽的测试几乎不可能。

我们根据功能需求，将 Web 软件划分成许多模块的集合，每个模块都可以看成是一个子 Web 软件(subWA)，它们可以分别进行处理。假定一个 Web 软件划分成 m 个 sub-WA。对于子 Web 软件 WA_k，它的 PFD_k 和 PTT_k 可以分别构造。这样，我们就分别得到 $\{PFD_1, PFD_2, \cdots, PFD_m\}$ 和 $\{PTT_1, PTT_2, \cdots, PTT_m\}$。令 P_i 是 WA_i 中的页面集合，P_j 是 WA_j 中的页面集合，并有 $Q_i\subset P_i$ 和 $Q_j\subset P_j$。若 Q_i 中的每个页面都指向(或被指)Q_j 中的某个页面，则构造基于 $Q_i\cup Q_j$ 的 $PFD_{i,j}$($PTT_{i,j}$ 的构造就简单了)。因此，我们得到一个类似树的(tree-like)页面流图 TPFD，其中的叶子是子 Web 软件的 $PFD_1, PFD_2, \cdots, PFD_m$，分支结点由 $PFD_{i,j}$ 形成(若 Q_i 和 Q_j 存在联系，$1\leqslant i, j\leqslant m, i\neq j$，它们可以构成更高的层次)，树根是整个 Web 软件的 PFD。注意，根 PFD 是一个虚结点，它由所有子结点实现。类似树的页面测试树 TPTT 用同样的方法构造。这样，我们就以"分而治之"的方式解决了整个 Web 软件的测试。因此，整个 Web 软件的测试可以分解成多个子 Web 软件的测试，这简单得多，也更容易控制。子 Web 软件可以在不同的平台上由不同的测试人员并发测试，测试非常灵活。这也在一定程度上限制了空间爆炸问题的发生。此外，TPFD 和 TPTT 是 Web 软件的高层抽象。

4 实例分析

考察我们开发的一个典型的小型 Web 软件：简单 Web 登录系统 SWLS(Simple Web Login System)。其页面流图如图 3 所示。

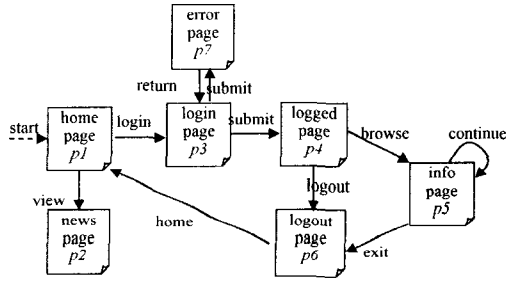

图 3 SWLS 的页面流图

从第一个页面(用虚箭头指示，通常，可以用一个 blank 页面对一个 Web 软件的第一个页面发出请求)开始，也就是 home page($p1$)，用户点击链接 view 进入 news page($p2$)查看新闻，或者点击按钮 login 进入 login page($p3$)。在页面 $p3$，用户输入 userid 和 password，接着点击 submit 按钮。一旦这样，userid 和 password 就发送到 Web 服务器进行验证。若 userid 和 password 都是正确的，将装载 logged page($p4$)，否则出现包含错误消息的 error page($p7$)。在 logged page，可以点击链接 browse 进入 info page($p5$)查看安全信息。若 $p5$ 太长，用户可以点击页内链接 continue 查看它的不同部分。当用户点击按钮 exit 或 logout 时，会显示 logout page($p6$)。然后，用户可以回到 home page 再次登录。注意，每次显示 login page 时，userid 和 password 域都必须清空。

根据 PFD2PTT 算法，可以得到图 4 表示的 PTT，它是由 Web 软件 SWLS 的 PFD 计算而来。

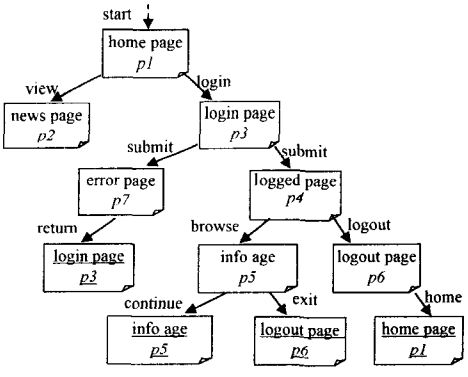

图 4 SWLS 的页面测试树

基于 SWLS 的 PTT 得到的路径表达式是 start→(view+login→(submit→return+submit→(browse→(continue+ex-

it)+logout→home)))。因此，存在 5 条测试路径，它们都起始于 home page。这些测试路径是：

- start→view
- start→login→submit→return
- start→login→submit→browse→continue
- start→login→submit→browse→exit
- start→login→submit→logout→home

5 测试模型

为了测试 Web 软件，必须生成测试用例。前面的测试路径可以很容易地构造测试用例。我们的测试模型（见图 5）定义的测试用例是带有输入值的测试路径。因此，若测试人员为同一个测试路径提供不同的用户输入值，则该测试路径可能构造多个测试用例。用户输入值可以通过多种现存的方法得到，例如 Offutt 等人提出的输入单元[13]。测试规格说明是由测试集、测试用例和测试步（test step）组成的一个层次结构[14]。在我们的测试规格说明中，测试用例被定义为测试步的树型结构。一条路径上的链接代表先后执行的测试步。换句话说，一个后链接（next）代表的测试步在它的前链接（previous）代表的测试步之后执行。兄弟链接（Sibling）表示测试路径中可选的测试步。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[14]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 Web 软件 SWLS 最后一个测试路径 start→login→submit→logout→home 的一个类似的测试规格说明：

```
〈testsuite name="the SWLS test suite"〉
  〈testcase name="the last test case"〉
    〈teststep name="start"〉>
      the request and response specification of test step start
      〈teststep name="login"〉
        the request and response specification of test step login
        〈teststep name="submit"〉
          the request and response specification of test step submit
          〈teststep name="logout"〉
            the request and response specification of test step logout
            〈teststep name="home"〉
              the request and response specification of test step
              home
            〈teststep〉
          〈teststep〉
        〈teststep〉
      〈teststep〉
    〈/teststep〉
  〈/testcase〉
.....
〈/testsuite〉
```

我们的 Web 测试模型扩展了 X. Jia 等[14]提出的模型，如图 5 所示。最开始，使用 Web 软件常用的需求和分析方法得到 PFD，然后根据 PFD2PTT 算法得到 PTT。测试翻译器（test translator）从 PTT 中获取页面关系来进一步构造路径表达式[12]，并把它们翻译成测试规格说明的测试脚本框架。该框架定义了测试套件（标记〈testsuite〉）、测试用例（标记〈testcase〉）。每个测试用例都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〉）。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。

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://mytestwebsite/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〉
```

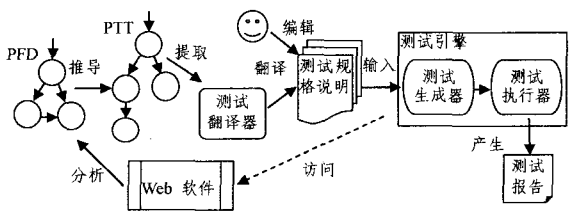

图 5 一个 Web 测试模型

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中产生的。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访问 Web 软件的客户端。

结束语 本文的 Web 测试方法的其中一个主要优点是，不需要访问后端的源代码。为了从 PFD 得到 PTT，提出了一个转换算法。通过构造路径表达式，从 PTT 可以得到测试路径。我们也给出了用 XML 描述测试路径的一个可能方式。测试引擎利用测试规格说明作为输入并输出测试报告。文中还提出了两个重要的概念：全链接覆盖和全页面覆盖。全页面覆盖表达了覆盖所有页面的最小数量的测试路径。以一个简单的 Web 登录系统 SWLS 阐述了所提出的测试路径生成方法，这给页面流测试方法提供了一个有意义的基础。

（下转第 159 页）

然后，软件精度和安全等问题经过修正(浮点型变为双精度；对 COM 接口设置缓冲区防御机制，如限制字符长、判断是否含有二进制代码和采用更安全的函数等) 重新度量，统计后的质量评价结果为 0.98X。

结束语 如何对软件的质量进行科学的评测是一个系统工程。应根据软件的特点、应用环境、适用人群、技术架构等提炼出软件隐含的质量需求。本文给出了一个针对不同行业软件质量评测模型，研发人员和用户在定义待测软件的质量评价标准时，可参考本模型确定其行业软件的度量项以及各组成部分的权重。例如，工程测量类软件由于更多地应用于地理、交通、能源等行业工程建设，对软件的数据精确度、计算准确度和数据的便于传输处理的通用性具有较高的需求，所以在制定工程测量类软件的质量评测模型时，应适当提升功能的权重，特别是精确性和准确性等方面的需求。此外，A，B，C，D 度量类之间的关联规则还需要深入研究，因地制宜，从使用人员和应用平台等多方面考虑，这样才能使软件得到更充分的改进和有效的应用。

参 考 文 献

[1] GB/T16260.1-ISO/IEC 9126-1. 第 1—4 部分软件工程-产品质量[S]. 2006

[2] Al-Kilidar H, Cox K, Kitchenham B. The use and usefulness of the ISO/IEC 9126 quality standard[C]// 2005 International Symposium on Empirical Software Engineering (ISESE 2005). Noosa Heads, Australia. IEEE, 2005: 126-132

[3] Garcia F, Bertoa M, Calero C, et al. Towards a consistent terminology for software measurement[J]. Information and Software Technology, 2006, 48(8): 631-644

[4] 孙梦璘，甘志强. 航天型号软件代码质量度量评估实现[J]. 系统工程与电子技术，2009，31(4)：956-133

[5] 柳永坡，张志良，吴际忠，等. 石油应用软件的测试方法研究与实践[J]. 石油地球物理勘探，2008，43(6)：731-735

[6] Alipour H, Isazadeh A. Software Reliability Assessment Based on a Formal Requirements Specification [C]// International Conference on Human System Interactions. Publisher: IEEE Computer Society, 2008: 311-316

[7] Alipour H, Isazadeh A. Software Reliability Prediction Based on a Formal Requirements Specification[C]// Advances in Computer Science and Engineering 13th International CSI Computer Conference. Kish Island, Iran, 2008: 816-820

[8] 聂南，谢晓东，甘勇，等. 基于 XML API 的组件扩展接口变异测试方法[J]. 计算机科学，2008，35(6)：283-287

[9] 陈锦富，卢炎生，谢晓东. 软件错误注入测试技术研究[J]. 软件学报，2009，20(6)：1425-1444

(上接第 155 页)

测试的关键是自动化。PTT 可以根据给定的 PFD2PTT 算法自动地从 PFD 得到。测试翻译器从 PTT 中获取路径表达式并把其翻译成测试规格说明的测试脚本框架。测试引擎读取并执行测试规格说明进而生成测试报告。总体上，除了手工构造 PFD 和对测试脚本框架的小小编辑修改之外，我们的 Web 测试方法是自动化的。

为了构造 PFD，我们结合了传统的需求分析方法和 Web 软件的特性。为了生成测试用例，一种方法是将随机输入应用到 PFD 中，直到每个链接至少经过一次。然而，路径可能包含循环，这使得它变得很长。此外，为了获得经过每个链接一次且仅一次的测试路径，PFD 必须是一个欧拉图。若 PFD 不是一个欧拉图，则构造最优长度的测试路径是一个 NP 完全问题。

在测试时，可能存在这样的情况，即只有提供某个特定的输入，特定的链接才会出现在某个页面中。下一步的工作是探讨这种新情况下 Web 测试的问题，以改进提出的测试路径生成方法，并开发一个原型验证本文提出的 Web 测试模型。

参 考 文 献

[1] Li Z, Alaeddine N, Tian J. Multi-Faceted Quality and Defect Measurement for Web Software and Source Contents[J]. Journal of Systems and Software, 2009, 83(1): 18-28

[2] Hower R. Web Site Test Tools and Site Management Tools. Software QA and Testing Resource Center[OL]. http://www.softwareqatest.com/qatweb1.html. Accessed on Feb. 2010

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

[4] Dobolyi K. An Exploration of User-Visible Errors to Improve Fault Detection in Web-based Applications[D]. School of Engineering and Applied Science, University of Virginia, 2009

[5] Halfond W G J, Anand S, Orso A. Precise Interface Identification to Improve Testing and Analysis of Web Applications[C]// The 18th International Symposium on Software Testing and Analysis. ACM, NY, USA, 2009

[6] Subraya B M, Subrahmanya S V. Object Driven Performance Testing of Web Applications[C]// The First Asia-Pacific Conference on Quality Software. HongKong, China, Oct. 2000

[7] Andrews A, Offutt J, Alexander R. Testing Web Applications by Modeling with FSMs [J]. Software and Systems Modeling, 2005, 4(3): 326-345

[8] 钱忠胜，缪淮扣. 面向用户会话的 Web 应用测试用例生成及其优化[J]. 计算机科学与探索，2008，2(6)：627-640

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

[10] Reza H, Ogaard K, Malge A. A Model based Testing Technique to Test Web Applications Using Statecharts[C]// ITNG' 08. Washington, DC, USA, 2008: 183-188

[11] 邓小鹏，邢春晓，蔡莲红. Web 应用测试技术进展[J]. 计算机研究与发展，2007，44(8)：1273-1283

[12] Beizer B. Software Testing Techniques (2nd edition)[M]. International Thomson Computer Press, 1990

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

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