ISSN 1673-9418 CODEN JKYTA8
Journal of Frontiers of Computer Science and Technology
1673-9418/2008/02(06)-0627-14
DOI:10.3778/j.issn.1673-9418.2008.06.006

E-mail:fcst@public2.bta.net.cn
http://www.ceaj.org
Tel:+86-10-51616056

面向用户会话的 Web 应用测试用例生成及其优化 *

钱忠胜[1,2+],缪淮扣[1]

QIAN Zhongsheng[1,2+], MIAO Huaikou[1]

1. 上海大学 计算机工程与科学学院,上海 200072

2. 江西财经大学 信息管理学院,南昌 330013

1. School of Computer Engineering & Science, Shanghai University, Shanghai 200072, China

2. School of Information Technology, Jiangxi University of Finance & Economics, Nanchang 330013, China

+ Corresponding author: E-mail: changesme@163.com

QIAN Zhongsheng, MIAO Huaikou. Test case generation and optimization for Web application test based on user sessions. Journal of Frontiers of Computer Science and Technology, 2008,2(6):627-640.

Abstract: The test is an effective way to ensure the high quality and high reliability 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 test case generation and optimization is one of the key problems.An approach to generating and optimizing test cases is proposed for Web application test based on user sessions using genetic algorithm. A large volume of meaningful user sessions are obtained after purging their irrelevant information by analyzing user logs on the Web server. Most of the redundant user sessions are also removed by reducing them. For test reuse and test concurrency, it divides the user sessions obtained into different groups, each of which is called a test suite, and then prioritizes the test suites and the test cases of each test suite. So, the initial test suites and test cases, and their initial executing sequences are achieved. However, the test scheme generated by the elementary prioritization is not much approximate to the best one. Therefore, genetic algorithm is employed to optimize the grouping and prioritization. Meanwhile, an approach to generating new test cases is presented using crossover. The new test cases can detect errors

* the National Natural Science Foundation of China under Grant No.60673115 (国家自然科学基金); the National Grand Fundamental Research 973 Program of China under Grant No.2007CB310800 (国家重点基础研究发展规划(973)); the National High-Tech Research and Development Plan of China under Grant No.2007AA01Z144 (国家高技术研究发展计划(863)); the Research Program of Shanghai Education Committee under Grant No.07ZZ06 (上海市教委科研项目); Shanghai Leading Academic Discipline Project, No.J50103 (上海市重点学科建设项目).

Received 2008-01, Accepted 2008-05.

caused by the use of possible conflicting data shared by different users.

Key words: Web applications; genetic algorithm; test case; test suite; reduction; prioritization; common prefix

摘 要:测试是保证Web应用的高质量、高可靠性的一种有效手段,然而,由于其特殊性和复杂性,使得传统的测试理论与方法很难直接运用到Web应用的测试中,一个关键的问题就是测试用例的生成及其优化。提出了一种将遗传算法用于基于用户会话的Web应用测试用例生成及其优化的方法。通过分析服务器的用户日志,清除无关的数据,得到大量有意义的用户会话,利用约简技术进一步剔除其中的冗余。为便于测试的重用和并发执行,将用户会话进行合理的分组,每一组称为一个测试套件,并在测试套件之间以及测试套件内部(测试用例之间)进行初步的优先排序。这样就得到了初始的测试套件和测试用例,以及它们的初始执行顺序。这种初始的测试方案离最优解的近似程度还不是很高,需进一步利用遗传算法对它们进行分组优化并优先排序。同时提出了一种利用交叉算子产生新的测试用例的方法,新的测试用例可以检测不同用户共享数据时可能带来的冲突而产生的错误。

关键词: Web应用;遗传算法;测试用例;测试套件;约简;优先排序;公共前缀

文献标识码:A **中图分类号:**TP311

1 引言

随着Web应用在全球范围内快速地发展,越来越多的基于Web的应用相继出现,许多组织将Web应用集成到关键型业务系统中,如电子市场、电子银行、电子贸易以及公共管理服务等。Web应用广泛而迅猛地发展,使得对其严格的质量需求变得非常紧迫,因此Web应用测试也就显得越来越重要。然而,由于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,其异构性、动态性、连接的多样性、控制流程的可变性以及需要快速开发与发布等特性给Web应用的测试带来了新的挑战。

根据Web应用的特性,分析发现它的测试与传统应用的测试存在以下区别:

(1)Web应用是部署在Web服务器上的,可以被多个用户同时访问,但却只运行一个实例。每个用户通过客户端的Web浏览器与Web服务器进行交互。客户端的配置千差万别,通常会带来各种各样的错误。而传统的软件一般安装在用户的本地机器上,每个用户运行软件的一个实例。

(2)Web应用由许多不同的软件组件(如,Web服务器、数据库、中间件)和集成系统(ERP系统、内容管理系统)构成,这些都来自不同的供应商,并用不同的技术实现,它们构成了Web应用的技术结构基础。

(3)Web浏览器的函数提供的导航,如回退(back)、前进(forward)和刷新(refresh)等按钮,也是错误发生的根源。

(4)Web服务器通过cookies、URL重写和隐藏域(hidden field)来识别用户的标识,而传统的应用一般是通过全局变量保存用户的标识。Web应用的客户端可以影响Web应用的行为,例如,是否禁用cookies,是否允许下载ActiveX控件等。

(5)由于Web应用在全球范围内广泛使用,不同的语言以及不同的使用性等问题也会给Web应用的测试带来了极大的挑战。比如,不同语言中的不同长度的文本消息可能会导致页面布局困难。

(6)实现Web应用的不同种类技术的出现,也使得以适当的粒度定义和验证Web测试模型的各个组件变得非常困难,充分建立组件间的关系也是一大难题。

目前还没有比较系统的方法和工具对Web应用进行有效的测试,因而Web应用的测试重复、低效,

导致 Web 应用软件的开发可靠性不高、风险大。本文分析了 Web 应用测试中的关键问题：测试用例的生成及其优化。一个好的抽象测试用例应该围绕实际的用户输入产生其数据部分。Web 服务器的日志记录了日常的用户输入数据,以这些数据为基础产生测试用例并对其进行优化具有实际意义。文中研究了将遗传算法用于基于用户会话的 Web 应用测试用例生成及其优化的方法。在将一个用户会话转换成一个测试用例时,保存了用户输入的数据。

2 基于用户会话的测试

用户会话可以通过描述运行 Web 浏览器的用户的典型行为来获得，即根据用户浏览 Web 时的活动期和静默期的交替变化来产生。活动期被称为会话,可能包含多个 TCP 连接,通常由相同的主机打开,连接到不同的服务器。静默期是会话之间的等待期间,此时主机没有新的 TCP 连接，但可以保持已有的 TCP 连接。然而,活动期和静默期的识别是非平凡的。传统的方法[1]采用一个阈值,即,如果两个 TCP 连接间的时间小于阈值，则它们聚集在同一个会话中,否则一个新的会话开始。这种方法只有当指定的阈值能跟连接与会话间的到达时间的平均值相匹配时才适用。可是,想要事先知道这些值实际上是不现实的。如果阈值不能很好地匹配用户会话的统计行为,则基于阈值的方法在会话识别中是很容易出错的。

基于用户会话的测试是一个近似自动化的测试方法,它围绕真实的用户数据模拟以增强初始测试套件的可信度,当用户的操作配置(profile)不断变化时,可动态地增加测试用例和测试数据[2]。但当新的测试用例追加到测试套件中来测试新的或变化了的需求时,或维护测试套件的充分性时,测试套件的规模会随着上升，在修改了的 Web 应用上运行该测试套件的费用也会有所增加。这里先给出测试用例和测试套件的形式化定义。

定义 1（测试用例）一个测试用例是一个三元组 $t=(Pre,In,Out)$,它是能导致程序一次执行的输入集,其中,*Pre* 表示输入的前置条件,*In* 表示输入值,*Out* 表示期望输出。

对于一个测试用例,若其输入的前置条件 *Pre* 永真,则它的输入值 *In* 总会被“执行”。

定义 2（测试套件）一个测试套件是一个四元组 $S=<q_0,<t_1,\cdots,t_n>,<q_1,\cdots,q_n>,\Delta>$,其中 q_0 为初始测试状态,$<t_1,\cdots,t_n>$ 为对应于 q_0 的测试用例序列,$<q_1,\cdots,q_n>$ 为使用了测试用例后的测试状态序列,Δ 为测试用例的执行,使得 $q_i=\Delta(t_i,q_{i-1}),1\leqslant i\leqslant n$。

为了下面讨论方便,把一个测试套件看成是相关测试用例的集合[1],而忽略它的测试状态。在基于用户会话的测试中,一个测试用例是多个 HTTP 请求的序列,这些请求都和一个特殊的用户会话关联。一个会话开始于一个用户发出的一个请求,结束于此用户离开 Web 应用或会话超时。也就是说,一个用户会话被认为是一系列的 URL 和名字-值对的序列,是指某一用户在一个预先规定的时间段内发出的一系列请求。尽管如此，多数测试方法定义的用户会话不包括名字-值对,也不区分用户对 URL 的请求顺序。然而,两个用户会话既使请求相同的 URL，它们也会因为具有不同的值而可能覆盖程序代码的不同区域(发现的错误出现在不同的地方)。从同一个 IP 发出的请求若间隔一定的时间被认为是不同的会话。为了把一个用户会话转换成一个测试用例[2]，每一个记录下来的请求被转换成 HTTP 请求发送到 Web 服务器。已有不同的为用户会话构造测试用例的策略[3-4]。

运行测试套件中所有的测试用例要费很大的精力。研究人员已经建立了两种和原先的测试套件维持

[1] 当根据某些策略对测试用例的执行进行过排序时,一个测试套件就成为测试用例组成的序列了,而不是集合,因为集合的成员之间是没有顺序的。但若不产生混淆,本文沿用集合的概念。

[2] 除非特别说明,一般不加区分地使用“用户会话”和“测试用例”这两个概念,因为经过处理后的一个用户会话被转换成一个测试用例,它们之间是一对一的关系。

相同覆盖率的方法,用以探讨测试套件的效率和性能问题,即测试套件约简(reduction)技术[5-9]和测试套件优先排序(prioritization)技术[10-12]。

2.1 测试套件约简技术

测试套件约简技术对基于用户会话的测试是非常重要的,因为收集、分析和重放大量的测试用例要花费很大的代价。一般地,测试套件约简技术依次地考察其中的每一个测试用例,移除不会改变或影响测试需求的那些测试用例,并且保证得到的结果测试套件和原先的测试套件具有同样的性质,比如它们的测试覆盖率或错误检测能力一样。在生成测试套件后进一步进行约简很重要,因为后面加进来的测试用例可能包含前面的。许多约简技术不能保证约简后的测试套件是测试用例的最小集合(这些测试用例能产生原先的操作抽象),此外,也不保证被移除的测试用例不能检测到错误。

对于一个测试套件 S 来说,一个测试用例 t 是"冗余的",仅当 t 覆盖的测试需求是 $S-\{t\}$覆盖的测试需求的子集。为了从一个测试套件 S 中删除某个测试用例 t 并保持 S 的覆盖率,t 必须是冗余的。而一个测试用例 t 是"必要的",仅当 t 覆盖的测试需求不是 $S-\{t\}$覆盖的测试需求的子集。如果一个测试用例能唯一覆盖其中任何一个测试需求,则其必须加入到约简后的测试套件中,以保持需求的覆盖率。在测试套件 S 中的每一个测试用例 t 都可以被标识为冗余的或必要的。首先识别必要的测试用例,然后迭代地识别未被标记为必要的且最弱的测试用例(即,对测试需求的覆盖率作用最小的测试用例),删除该测试用例,接着识别新的、必要的测试用例。当所有的被原先的测试套件覆盖的测试需求同样被必要的测试用例集覆盖,就得到约简的测试套件。测试套件约简技术会丢弃某些测试用例,这可能会带来一定的缺陷。例如,随着测试的运行,被丢弃的测试用例覆盖的需求永远不能被其他的测试用例覆盖。

文献[13]从测试需求约简的角度考虑测试套件的优化,先给出可以精确描述测试需求间相互关系的测试需求约简模型,基于此模型提出了一种测试需求约简方法,可以获得精简测试需求集,作为测试套件生成和约简的基础,从而实现测试套件的优化。文献[2]提出了另一个基于需求的约简技术,它是一种试探法,假设 WA 是被测 Web 应用,R 是一个测试需求,S 是 WA 的一个测试套件,S'是通过分析 WA 和 S 而得到的约简的测试套件,步骤如下:

(1)初始化 S'为空;

(2)从 S 中选择一个候选的测试用例 t 放入 S';

(3)重复步骤(2)直到 S'满足需求 R。

2.2 测试套件优先排序技术

测试套件优先排序技术根据某个准则识别测试套件的顺序,和测试套件约简技术相比,它向新的测试套件中添加最强的测试用例(即,对测试需求的覆盖率作用最大的测试用例)。测试用例优先排序技术使得高优先级的测试用例比低优先级的要执行的早,以尽快满足某些需求。比如,测试人员可能希望以一个更快的速度提高被测 Web 应用的可覆盖代码的覆盖率,允许代码覆盖标准在测试阶段的早些时候得到满足。他们也可能希望在 Web 应用的可靠性方面较快地增强自信心。可见,测试套件优先排序技术本身不会丢弃测试用例。

文献[11]对优先排序问题进行了形式化的定义。给定三元组(S,Π,fit),其中 S 是一个测试套件,Π 是 S 的所有置换的集合,fit 是一个从 Π 映射到实数的适应度函数,用于计算所有置换的优先顺序。适应度函数表征了测试套件测试效率的高低。每一个置换 $\pi_i\in\Pi$ 包含 n 个测试用例,即 $\pi_i=(\pi_{i1},\pi_{i2},\cdots,\pi_{in})$。现在要找到 $\pi\in\Pi$,使得对 $\forall\pi'\in\Pi,\pi'\neq\pi$,有 $fit(\pi)\geqslant fit(\pi')$。适应度函数根据每个置换在 Web 应用中的代码覆盖率,给该置换赋予一个适应度值。优先排序技术可用来重新调整测试套件的顺序,以便在测试执行的早期提高错误检测率。

如果测试必须早些终止,排过序的测试套件比未排过序的测试套件在发现错误方面效率更高。直观地,一个测试套件若发现错误的可能性更大,并能在用户指定的时间内执行,则它具有更好的适应度。

另外,与测试套件效率和性能问题有关的还有测

试选择技术[14-15],它选择测试套件的一个子集来执行所有的变更,但这些测试套件一般不能提供和原先测试套件相同的测试覆盖率,可以认为是一种增量测试方法。

2.3 混合技术

测试套件约简技术与其优先排序技术一样,对提高测试效率是很有价值的。比如,运行一个大的测试套件可能要几个星期,但测试人员通常只运行测试套件的一个有代表性的子集。测试人员必须从测试套件中移除冗余的测试用例,但又不牺牲原先测试套件的大多数错误检测能力以及覆盖率特性。

当用来执行测试套件的时间较短时,测试用例优先排序技术也许是不划算的,这可能会简单地以任意顺序安排测试用例。当执行时间充足时,该技术是有益的,因为在这种情况下,较早地满足测试目标是有意义的。

由于测试套件优先排序技术本身不丢弃测试用例,这可避免测试选择技术和测试套件约简技术丢弃测试用例带来的缺陷。即使是在丢弃测试用例可以被接受的情况下,将测试套件优先排序技术同测试选择技术或测试套件约简技术相结合也是可取的。例如,在使用了测试选择算法或测试套件约简算法之后可以再对得到的测试套件进行优先排序。另外,若测试活动非预期地终止,则在时间花费所带来的更多的效益上,测试套件优先排序技术要比非优先排序技术的可能性大些。

3 测试用例生成及其优化

Web 应用给测试人员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,并且必须开发一个新的特定于 Web 应用的测试方法。

基于用户会话的测试反映了真实的使用情况,这是测试人员在开发阶段的早期所不能预测到的。从真实用户那里获取的会话数据对测试人员在内部产生的测试套件是一个有益的补充[3]。将遗传算法用于基于用户会话的 Web 应用的测试中,称其为基于遗传算法的 Web 应用测试。

3.1 用户会话的收集和约简

在每个站点服务器的日志文件中,每一条访问记录对应用户的每一次信息请求,记录的内容一般包括:发出请求的源(用户的 IP 地址)、发出请求的时间、请求的方式("GET"、"POST"等)、被请求信息资源的 URL、数据传输协议(如 HTTP)、状态代码、传输的字节数、客户端的类型。从 Web 应用当前的用户访问日志中分解出每个会话的原始活动历史记录只需对日志进行一次扫描即可。但这些原始的记录是很难直接进行组织的,必须经过预处理过程。首先清除无关数据(包括状态代码为错误的日志记录,代码 200 为成功,400 为错误;请求资源为嵌入资源,如脚本文件和多媒体文件.gif、.jpeg、.css 的记录),得到作为分析源的会话记录集合;其次扫描服务器中的日志文件来创建用户会话。每当遇到一个新的 IP 地址,便创建一个会话,然后把从该 IP 地址中发出的后续请求都加到这个会话中,条件是连续的两个请求的时间间隔不能超过事先设定的时间参数 max-session-idle-time,否则作为另一个会话的开始;最后得到所有用户会话的集合。

收集到的用户会话往往很多,而一个用户会话转换成一个测试用例发送到 Web 服务器中,因此可以利用用户会话约简技术去除"冗余的"用户会话,保留"必要的"用户会话。为便于讨论,下面给出几个重要的概念。

定义 3(URL 迹)一个用户会话请求的 URL 序列称为 URL 迹。

令 α 为一条 URL 迹,它的长度为迹中请求的 URL 的个数,记为$|\alpha|$。

定义 4(前缀)一条迹 α 是另一条迹 β 的前缀,当且仅当 α 是 β 的子序列且它们的起始符号相同。

定义 5(公共前缀)如果一条迹同时是多条迹的前缀,那么这条迹就成为多条迹的公共前缀。

定义 6(最大公共前缀)在两条迹的所有公共前

缀中最长的那个前缀是它们的最大公共前缀。

两条迹的最大公共前缀越长，则它们越相似,即它们的相似度越高。假设有 4 条迹 $\gamma_1=abcdefg$、$\gamma_2=abcdeh$、$\gamma_3=abcd$ 和 $\gamma_4=cde$,则迹 γ_3 是 γ_1、γ_2 的公共前缀,但不是最大的,γ_1 和 γ_2 的最大公共前缀是 $abcde$。而 γ_4 虽然是 γ_1、γ_2 的子序列,但不是 γ_1、γ_2 的前缀,因为 γ_4 和它们的起始符号不同。定义函数 isPrefix(α,β)判断迹 α 是否为 β 的前缀，即判断一个会话请求的 URL 迹相对于另一个会话请求的 URL 迹是否冗余,若是其前缀则返回 TRUE,否则返回 FALSE。基于 URL 迹的用户会话约简算法 ReduceUSession 如下所示(注意,这里把一个 HTTP 请求看成是迹的一个符号)。

基于 URL 迹的用户会话约简算法 ReduceUSession:

```
Algorithm:ReduceUSession
input:
  用户会话集 Λ={s1,…,sk},k 为用户会话总个数;
  用户会话分别请求的 URL 迹 U1,…,Uk
    其中 si 请求的 URL 迹为 Ui,1≤i≤k
output:
  约简的用户会话集 Γ
begin
  Γ=Φ;
  while(Λ 中还有未标记的用户会话)
    tag1=FALSE;
    tag2=FALSE;
    从 Λ 中选取一个未标记的用户会话 si 并标记其"已使用";
    for(Γ 中每一个用户会话 sj 请求的 Uj)
      if isPrefix(Uj,Ui)//si 请求的 URL 更多,sj 是冗余的
        Γ=Γ-{sj};
        tag1=TRUE;
      endif;
      if isPrefix(Ui,Uj)//此时 Γ 不变,si 是冗余的
        tag2=TRUE;
        break;//退出 for 循环
      endif;
    endfor;
    if tag1 ‖(! tag1 && ! tag2) //si 是必要的
      Γ=Γ∪{si};
    endif;
  endwhile;
  输出约简的用户会话集 Γ;
end.
```

给出的约简算法ReduceUSession 判定一个用户会话请求的 URL 迹 α 是否为别的用户会话请求的 URL 迹 β 的前缀,若是,则 URL 迹为 α 的用户会话被删除。利用该算法,得到的用户会话数量大大减少。

基于 URL 迹的用户会话约简算法 ReduceUSession 与已有的约简方法不同。已有的约简方法大多数是依次地考察测试套件中的每一个测试用例,移除不会改变或影响测试需求的那些测试用例，即要区分"冗余的"和"必要的"测试用例,这在实际中是很难做到的,因为已有的方法很难在测试执行之前区分哪些测试用例(即冗余的)覆盖的测试需求被其他的测试用例覆盖了。而算法 ReduceUSession 分析一个会话请求的 URL 迹是否为另一个会话请求的 URL 迹的前缀,以此来识别冗余的测试用例,在实际中很容易做到,且覆盖了原先用户会话集请求的所有 URL 并保持 URL 请求的顺序,即保证了覆盖原先的测试需求。

3.2 用户会话的分组和优先排序

把利用基于 URL 迹的用户会话约简算法 ReduceUSession 得到的用户会话进行分组,每一组看成是一个测试套件。分组的目的是让测试用例得到重用,而且测试可以在不同的平台上并发(或并行)执行,减少测试的时间,提高测试的效率。另外还可以测试每组的用户会话之间的两两交互(见 3.3.4 节)。在同一组的用户会话之间"尽量"保持这种性质:访问的 URL 迹具有一定长度的最大公共前缀。这需要定义多个不连续的阈值 $\zeta_1,\zeta_2,\cdots,\zeta_k$ 且 $\zeta_i\geqslant 1,1\leqslant i\leqslant k$($k$ 一般选择的稍微大一些，因为一个 Web 应用的用户会话往往会很多,分组也多,同时也方便些),当用户会话访问的 URL 迹的最大公共前缀长度在一定的阈值之间时,就把它们分成一组。例如,假设给定了 3 个阈值 $\zeta_1=2,\zeta_2=4,\zeta_3=7$,则用户会话被分成 4 组 $S_1,S_2,$

S_3和 S_4,每组的最大公共前缀长度分别为$|\alpha|\leqslant 2$,$2<|\alpha|\leqslant 4$,$4<|\alpha|\leqslant 7$,$|\alpha|>7$(令 α 为最大公共前缀)。这 4 组可以分别在不同的平台上并发(或并行)测试,最大公共前缀在每组中进行测试时也可以得到重用。注意,这种分类方法并不唯一。例如,在前面假设的 4 条迹 $\gamma_1=abcdefg$,$\gamma_2=abcdeh$,$\gamma_3=abcd$ 和 $\gamma_4=cde$ 中,可以分为$\{\gamma_1,\gamma_2\}$和$\{\gamma_3,\gamma_4\}$这两组,每组的最大公共前缀长度分别满足 $4<|\alpha|\leqslant 7$ 和$|\alpha|\leqslant 2$;也可以分为$\{\gamma_1,\gamma_2,\gamma_3\}$和$\{\gamma_4\}$这两组,每组的最大公共前缀长度分别满足 $2<|\alpha|\leqslant 4$ 和$|\alpha|\leqslant 2$。当然,在每组中也不要求用户会话请求的 URL 迹两两之间的最大公共前缀长度满足规定的阈值,只需大部分之间满足即可(可以设置或随机产生一个百分比来度量)。这就需要找到一种折衷的方法对它们进行分类,即在分类和并发(或并行)测试以及测试重用的代价之间找到一种平衡。假设有 N 个约简后的用户会话被分成 K 组。一般来说,分出来的每组之间用户会话的个数基本相同,即近似于。此外,这种分组也是初步的,称之为“初步分组”。

公共前缀表达了用户会话共同的事件,或客户端用户相同或相似的操作。另一方面也说明了用户具有相同或相似的爱好,公共前缀长度越大就表现的越明显,这对应于大多数用户的情况。此外,请求的 URL 迹的最大公共前缀长度最小的那组用户会话比较特殊,该组中的用户会话展示彼此之间不同的 URL 请求,对 Web 应用具有不同的需求。这组用户会话通常发生“非常规的”事件,输入“不常用的数据”,属于“边界”情况,对 Web 应用来说,它们最容易出错。

鉴于此,将测试套件进行优先排序。公共前缀长度最小的那组测试套件排在第一位,接着把剩余的测试套件按公共前缀长度从大到小排列,直到公共前缀长度第二大的那组测试套件。在最大公共前缀长度$|\alpha|\leqslant 2$,$2<|\alpha|\leqslant 4$,$4<|\alpha|\leqslant 7$,$|\alpha|>7$ 分别对应的测试套件 S_1,S_2,S_3和 S_4中,优先排序后的测试套件顺序为:S_1,S_4,S_3,S_2,即,最先执行 S_1中的测试用例,接着执行 S_4和 S_3中的测试用例,最后执行 S_2中的测试用例。在每个测试套件中,则按对请求的 URL 的覆盖率大小的方式来优先排序测试用例,即,请求的 URL 迹的长度越大的测试用例越先得到执行;若同一个测试套件中有多个测试用例请求的 URL 迹长度相等,则它们之间的执行顺序是随机的。

这里的优先排序方法与已有的方法不同。已有的优先排序方法向新的测试套件中添加最强的测试用例,即对测试需求的覆盖率作用最大的测试用例,力求使得高优先级的测试用例比低优先级的要执行的早,以尽快满足某些测试需求。这些方法在测试执行之前,每次要找出最强的(或接近最强的)测试用例往往很难实现。在对测试用例利用优先排序方法之前,先对它们按公共前缀的特性进行分组。分组便于测试人员更有选择地执行某些组的测试用例,以查找某种类型的错误,因为同一组的测试用例往往能够发现相同或相似类型的错误。另外,分组可以使得测试用例得到重用,而且测试可以在不同的平台上并发(或并行)执行,减少测试时间并提高测试效率。我们还特别考虑了一类特殊的用户会话,即请求的 URL 迹的最大公共前缀长度最小的那组用户会话,这组用户会话通常包含“特别的”请求,是主要的错误来源。

3.3 基于遗传算法的 Web 应用测试

通过上面的分组和优先排序以后,就得到多个初始的测试套件和测试用例,以及它们的初始执行顺序。这种初始的测试方案发现错误的速率不是很快,不能较早地满足测试需求,即离最优解的近似程度还不是很高,下面利用遗传算法进一步对测试套件进行分组优化并优先排序。

1975 年美国的 Holland 教授首次系统提出了遗传算法的思想[16],从而吸引了大批的研究者,迅速推广到优化、搜索、机器学习等方面,并奠定了坚实的理论基础。遗传算法借用了生物遗传学的观点,通过自然选择、交叉和变异等作用机制,实现各个个体适应性的提高,是具有“生存+检测”迭代过程的寻优搜索算法。这一点体现了自然界中“物竞天择、适者生存”的进化过程。

遗传算法以一种群体中的所有个体为对象,并利用随机化技术指导对一个被编码的参数空间进行高效搜索。其中,选择(selection)、交叉(crossover)和变异(mutation)构成了遗传算法的基本算子;参数编码、初始群体的设定、适应度函数的设计、遗传操作设计、控制参数设定这5个要素组成了遗传算法的核心内容。作为一种高效的全局优化搜索算法,在解决大空间、多峰值、非线性、并行处理等高复杂度问题时显示了独特的优势和高效性。

这里将遗传算法运用到Web应用的测试中,进一步优化得到的初始测试套件和测试用例,以及它们的初始执行顺序,以得到较优的满足测试需求的测试套件和测试用例。把运用基本算子选择、交叉和变异产生一个下一代群体的过程称为一次迭代。要获得满意的解,需要进行多次迭代。下面介绍一次迭代的选择、交叉和变异的过程。

3.3.1 选择

选择算子从群体中按某一概率 p_s 成对选择个体,某个体的被选择概率与其适应度值成正比,最通常的实现机制是轮盘赌(roulette wheel)策略[16]。在选择过程中,适应度高的个体被直接复制到下一代群体中。适应度越高,产生后代的概率就越高,说明与预期效果越接近。假设在前面得到 K 个测试套件(初始群体),根据给出的优先排序方法得到的测试套件的优先级,按从高到低排列测试套件为:$S_1,S_2,\cdots,S_K$。在实际应用中,综合测试套件的错误覆盖率和测试运行代价设计适应度函数,并按从大到小的顺序排列,假设为:$f_1,f_2,\cdots,f_K$,其中 f_i 是 S_i 的适应度,$1\leqslant i\leqslant K$。某个个体的被选择概率用其适应度和所有个体适应度之和的比值来计算,即,测试套件 S_i 的被选择概率 $p_s^{S_i}$ 为:$f_i/\sum_{j=1}^{K}f_j$。显然,K 个测试套件的被选择概率之和等于1。

根据前面的讨论,适应度 f_1 和 f_2 分别对应着两个特殊的测试套件 S_1 和 S_2,它们的最大公共前缀长度分别是最小的和最大的。将 S_1 和 S_2 直接作为下一代的两个个体(也可根据实际情况,多选择几个适应度高的个体直接进入下一代),以避免万一它们没有被选中的情况发生。现在随机产生两个数 $g_1,g_2\in[0,1]$,并分别随机选择两个被选择概率大于或等于 g_1、g_2 的测试套件 S_i 和 S_j。将 S_i 和 S_j 作为双亲利用交叉和变异算子(见3.3.2节和3.3.3节)产生两个下一代个体 S_i' 和 S_j'。重复这种选择过程,直到产生了足够多的下一代测试套件。有一点要注意的是,某些具有更高的被选择概率的测试套件没有被选中作为父代,而一些被选择概率低的测试套件反而被选中了,这种情况是合理的,也符合自然界的生物进化论的观点,因为事物存在必然性的同时,也存在偶然性。

3.3.2 交叉

交叉算子将被选中的两个父代个体的基因链按概率 p_c 进行交叉,生成两个新的下一代个体。交叉位置是随机的,其中 p_c 是一个系统控制参数(一个简单的方法是随机产生 p_c)。将交叉得到的两个新的下一代个体替换它们的两个父代个体。测试套件的交叉算法TSCrossover如下所示。

测试套件的交叉算法 TSCrossover:

```
Algorithm:TSCrossover
input:
  父代测试套件 Si 和 Sj;
  交叉概率 pc
output:
  下一代测试套件 Si'和 Sj'
begin
  Si',Sj'←Si,Sj;
  随机产生一个数 g∈[0,1];
  if pc≥g
    随机产生一个整数 g'∈(0,min(|Si'|,|Sj'|)];
    //|Si'|,|Sj'|分别为 Si',Sj'中测试用例的个数
    一一交换 Si'和 Sj'中从位置 g'开始的测试用例,直到
    任何一个测试套件中不再有测试用例交换为止;
  endif;
```

输出测试套件 S_i'和 S_j';

end.

假设测试套件 S_i=<c_1,c_2,c_3,c_4,c_5,c_6,c_7,c_8,c_9>有 9 个测试用例,S_j=<c_{10},c_{11},c_{12},c_{13},c_{14},c_{15},c_{16}>有 7 个测试用例,开始交叉位置为 3,则 S_i和 S_j经过交叉以后得到两个下一代:S_i'=<c_1,c_2,$\boldsymbol{c_{12}}$,$\boldsymbol{c_{13}}$,$\boldsymbol{c_{14}}$,$\boldsymbol{c_{15}}$,c_{16},c_8,c_9>,S_j'=<c_{10},c_{11},$\boldsymbol{c_3}$,$\boldsymbol{c_4}$,$\boldsymbol{c_5}$,$\boldsymbol{c_6}$,$\boldsymbol{c_7}$>,其中黑体表示的测试用例是从父代测试套件中相应位置对应的测试用例一一交换得来的。

3.3.3 变异

变异算子将新个体的基因链的各位按概率 p_m进行变异,其中 p_m是一个系统控制参数(一个简单的方法是随机产生 p_m)。变异概率不能太大,否则会导致不稳定,它实际上是子代基因按小概率扰动产生的变化。变异可防止群体进化停滞不前或冻结,若无变异,则新群体中的测试数据值会局限于初始化的数值。对利用交叉算法产生的测试套件 S_i'和 S_j',分别对它们按一定的概率进行变异。同样,将变异得到的新的个体替换变异前的个体。变异算法 TSMutation 比较简单,如下所示。

测试套件的变异算法 TSMutation:

```
Algorithm:TSMutation
input:
  交叉算法得到的测试套件 S_i'和 S_j';
  变异概率 p_m
output;
  变异后的测试套件 S_i'和 S_j'
begin
  for each S in {S_i',S_j'}
    for each c in S
    随机产生一个数 g∈[0,1];
    if p_m≥g
      将 c 和排在它前面的测试用例交换位置;
    endif;
  endfor
  输出测试套件 S;
endfor
end.
```

假设变异概率 p_m=0.015,在用交叉算法 TSCrossover 得到的其中一个测试套件 S_i'=<c_1,c_2,c_{12},c_{13},c_{14},c_{15},c_{16},c_8,c_9>中,随机产生的数分别为:$g(c_1)$=0.246,$g(c_2)$=0.580,$g(c_{12})$=0.025,$g(c_{13})$=0.012,$g(c_{14})$=0.632,$g(c_{15})$=0.073,$g(c_{16})$=0.431,$g(c_8)$=0.193,$g(c_9)$=0.059,由于 $p_m \geq g(c_{13})$,则 c_{13}产生变异,需要和前面的测试用例 c_{12}交换位置。变异后的测试套 S_i'为:<c_1,c_2,$\boldsymbol{c_{13}}$,$\boldsymbol{c_{12}}$,c_{14},c_{15},c_{16},c_8,c_9>。

3.3.4 用户会话的交互测试

利用交叉算子还可以产生新的测试用例,这些测试用例包含不同用户的请求信息,目的是要检测不同用户共享数据时可能带来的冲突而产生的错误。交叉体现了用户会话之间交换信息的思想。在某个测试套件 S 中,利用交叉算子产生一个测试用例的步骤如下:

(1)对 S 中的一个测试用例 c,产生一个随机数 $g \in [0,1]$,如果给定的交叉概率不小于 g,那么:

①把 c 中从 r_1到 r_i的请求复制到待产生的测试用例中,其中 i 是一个随机数,$i \in [1,|c|]$,$|c|$为 c 中 URL 迹的长度;

②随机选择 S 中的一个测试用例 d,$d \neq c$,逆序查找第一个和 r_i有相同 URL 的 r_j,如果未找到,则选择另一个测试用例 d,直到找到为止或达到规定的次数;

③若找到 d,则把 d 中从 r_j以后的所有请求加入到待产生的测试用例的后面,该测试用例就是一个新产生的测试用例,否则转(②);

(2)重复(1),直到考察了测试套件中的每一个测试用例。

在测试用例 d 中使用了逆序查找请求的方法,是因为在同一个测试套件中的两个测试用例 c 和 d 的最大公共前缀长度一般大于 1,当从 r_1到 r_i对应的 URL 迹是这个最大公共前缀的前缀时,采用顺序查找的方法就达不到产生不同测试用例的目的。例如,

假设 $c=u_1u_2u_4u_3u_5u_6u_8u_7$,$d=u_1u_2u_4u_3u_7u_4u_5u_9$。先随机地从 c 中复制 $u_1u_2u_4$ 给待产生的测试用例 t(t 暂时等于 $u_1u_2u_4$),此时 i=3,即 $r_i=u_4$。现在从 d 中逆序查找第一个和 r_i 有相同的 URL 的 r_j,找到了 j=6 时 $r_j=u_4$,则将 d 中从 r_6 以后的所有请求,即 u_5u_9 加入到 t 的后面,得到新的测试用例 $t=u_1u_2u_4u_5u_9$。若从 d 中顺序查找的话,则找到了 j=3 时 $r_j=u_4$,此时将 d 中从 r_3 以后的所有请求,即 $u_3u_7u_4u_5u_9$ 加入到 t 的后面,得到的 $t=u_1u_2u_4u_3u_7u_4u_5u_9$,这实际上等于 d 本身,而并没有产生新的测试用例,这是因为 c 中从 r_1 到 r_i(这里 i=3)对应的 URL 迹 $u_1u_2u_4$ 是 c 和 d 的最大公共前缀 $u_1u_2u_4u_3$ 的前缀。

此外,一般不为一个测试用例进行变异操作,因为交换相邻的两个请求 r_i 和 r_{i+1} 后产生的新的测试用例可能没有任何实际意义,这是由于前面的请求 r_{i-1} 可能永远无法到达 r_{i+1}(注意,此时请求 r_i 已在 r_{i+1} 的后面了)。

3.4 测试方法讨论

在从服务器得到大量的用户会话以后,首先利用基于 URL 迹的用户会话约简算法 ReduceUSession 去除“冗余的”用户会话,保留“必要的”用户会话,以减少测试用例的数量。然后根据公共前缀的概念把相似的用户会话进行分组并优先排序,得到多个初始的测试套件和测试用例,以及它们的初始执行顺序。由于这种初始的测试方案离最优解的近似程度还不是很高,进一步利用遗传算法对测试套件进行分组优化并优先排序。

交叉算法 TSCrossover 将被选中的两个父代个体的基因链按概率 p_c 进行交叉,生成两个新的下一代个体。而变异算法 TSMutation 将新个体的基因链的各位按概率 p_m 进行变异,将交叉得到的两个新的下一代个体替换它们的两个父代个体。同样,将变异得到的新的个体替换变异前的个体。这样做的原因是:

(1)交叉和变异过程得到的新的个体(测试套件)被认为是更优的,它的执行能在更短的时间内找到错误;

(2)无论是测试套件的交叉还是变异过程都不会改变测试用例本身,只是它们的执行顺序改变了而已,而测试用例执行顺序的改变却会影响到能否较早地发现错误,以及较早地发现何种类型的错误。

因此,测试套件的交叉和变异都不会产生更多的测试套件和测试用例,但可以提高测试的执行效率。至于将交叉算子用于用户会话的交互测试,虽然测试用例的个数会增加,但这些新的测试用例可以检测不同用户共享数据时可能带来的冲突而产生的错误,即检测的错误类型不一样,错误覆盖率也有所增加,测试用例的质量提高了。

在测试执行过程中,可以并发(或并行)地执行每个测试套件,一旦执行到某个测试套件的某个测试用例而满足测试需求时,就不再执行剩余的测试用例和测试套件了,这就大大地减少了测试的执行时间。注意,在多个测试套件之间以及每个测试套件的测试用例之间都是有先后执行顺序的。由于将用户会话(测试用例)进行了分组,所以测试人员还可以有选择地先执行某些测试套件,以更大的概率和更快的速度发现某些类型的错误。

3.5 实验分析

为了阐述所提出方法的有效性,我们开发了一个小型的 Web 应用:简单 Web 登录系统(Simple Web Login System,SWLS)为例进行说明,如图 1 所示。

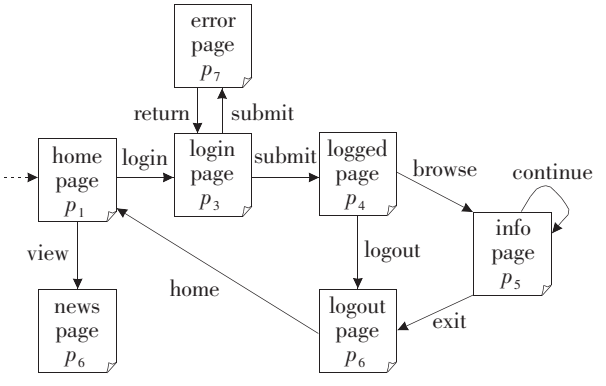

Fig.1 A Simple Web Login System

图 1 一个简单的 Web 登录系统

从主页 p_1 开始，用户可以通过链接 view 查看新闻页面 p_2，或者点击 login 按钮进入登录页面 p_3。在页面 p_3，用户输入用户名和密码并点击 submit 按钮，若用户名和密码都正确，则进入已登录页面 p_4，否则弹出错误页面 p_7。从页面 p_4，用户可以进入页面 p_5 浏览 Web 应用的安全信息（p_5 自身还存在页内导航），也可以点击 logout 按钮进入退出页面 p_6。在页面 p_6，可以重新回到首页 p_1，并可再次登录。注意，每次出现页面 p_3 时，用户名和密码输入框都必须初始化为空。

在每个页面都分别植入了 1 个错误，这样至少有 7 个错误（注意，除了手工植入的错误，Web 应用本身可能还会存在错误）。在对 Web 服务器日志进行扫描并清除无关的数据后，得到分析源的会话记录集，再次扫描后可以创建 89 个用户会话。利用基于 URL 迹的用户会话约简算法 ReduceUSession 对这些会话进行约简，最后只剩下 17 个用于生成测试用例的用户会话[3]，约简比率达到 80.9%。应该来说，对于同一个 Web 应用，用户会话越多，约简的效果会越明显，这是因为从统计的角度来看，会话越多意味着一个会话请求的 URL 迹成为另一个会话请求的 URL 迹前缀的可能性就越大。将约简了的 17 个用户会话（测试用例）组成的集合记为 Γ_1；现在将 Γ_1 分组并优先排序，得到测试套件的初始执行序列$<S_1, S_2, S_3>$，记为 Γ_2，其中 S_1, S_2, S_3 又分别包含 7 个、6 个、4 个测试用例的初始执行序列；利用遗传算法进一步处理，得到测试套件的执行序列$<S_1', S_2', S_3'>$，记为 Γ_3，即 S_1', S_2', S_3' 由 S_1, S_2, S_3 交叉和变异得到。

让 Γ_1、Γ_2 和 Γ_3 中的测试套件（和测试用例）分别对该 Web 应用进行测试，都能找到植入的 7 个错误，另外还发现了 1 个错误，即可以直接在浏览器的地址栏输入页面 p_2 的 URL 地址而查看该页。但 Γ_2 和 Γ_3 比 Γ_1 的执行时间要短的多，而 Γ_3 的时间又要比 Γ_2 的短一些。这说明在执行时，分组并优先排序得到的测试套件（和测试用例）比未进行分组和优先排序的要快，而经过遗传算法进一步处理的又更快一些。可以预测，对于大型的 Web 应用，本文的测试方法带来的正面效果会更明显。

4 相关研究

在 Web 应用的测试领域，已提出了一些测试方法和技术[17-20]。Kung 等[18]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）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）来描述。此外，控制与数据流信息用块分枝图（BBD）和功能聚簇图（FCD）描述。Hieatt 等[17]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。然而，Kung 和 Hieatt 等提出的方法都主要集中于单元测试，未关注 Web 应用的整体测试。文献[20]提出了基于数据流的 Web 应用的测试方法。Lucca 等[21]对 Web 应用的测试进行了综述，文章指出了 Web 应用测试和传统软件测试之间的主要差别，这些差别又是如何影响前者的，以及最近一些年来在 Web 应用测试领域的相关成果。但其主要关注的是 Web 应用的功能测试，即使也讨论了部分非功能的需求。这些测试方法大多数是在传统方法的基础上扩充而来的，然而，测试 Web 应用需要许多不同于传统软件测试的新方法，此外，这些测试方法[17-20]也都不是围绕实际的用户会话产生测试数据。

Elbaum 等[23]对基于用户会话测试的错误检测能力和成本效益进行了分析，发现该技术能揭示一定类型的缺陷，但不能揭示“很少输入的数据”产生的错误。他们表示，增加用户会话的数量可以提升效益，但同时也会增加测试的费用。增量概念分析技术[22]可以用来动态地分析用户会话，并不断地最小化维护的用户会话的数量。文献[24]将遗传算法和形式概念分析

[3] 为便于测试的执行，对用户会话进行了适当的加工处理。

相结合,以提供对测试数据与相应的测试执行间关系的追踪。

Sthamer[25]在其博士论文中详细地探讨了遗传算法的不同编码方式、不同适应度函数对不同结构软件的测试用例优化效率的影响,为其他学者的研究工作提供了不少宝贵的经验。部分学者也对遗传算法用于测试用例生成进行了研究[26-27],但他们的算法以单一优化为目标,侧重一次性优化计算。而测试是连续和反复的系统过程,动态连续优化计算更利于提高测试性能。文献[28]探讨了将遗传算法应用于生成覆盖指定路径测试数据时的关键问题,详细介绍了实验设计来分析影响遗传算法效率的因素。Berndt 和 Watkins[29]列举了近年来在基于遗传算法的测试数据生成方法中的进展,包括使用系统判据、模型、仿真等替代真实系统,以避免反复执行真实系统带来的开销,以及在各个遗传算子上所做的改进。这些研究[23-29]除文献[23]之外都利用了遗传算法分析测试问题,但考虑的都不是 Web 应用的测试。文献[23]讨论了基于用户会话的 Web 测试,但没有深入分析测试用例的优化问题。

与他们方法的不同是,本文将遗传算法用于基于用户会话的 Web 应用测试用例的生成以及优化,从分析 Web 服务器的用户日志,提取用户会话入手对 Web 应用的测试开展研究,具有一定的理论指导意义和实用价值。

5 结束语

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。它们利用许多新的语言、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,给测试人员带来了许多新的问题[17,30],这使得他们需要寻求一种或多种新的测试方法。

Web 应用测试的前提是生成高质量的测试用例。本文通过在服务器端捕获一系列用户事件产生的结果,即 URL 和名字-值对的序列,利用约简技术、分组和优先排序技术、遗传算法产生测试用例并对其进行优化,与在客户端捕获用户事件的方法相比,在大量用户存在时非常有效,是一种高效的 Web 应用测试方法。主要贡献如下:

(1)提出了一种将遗传算法用于基于用户会话的 Web 应用测试用例生成及其优化的方法;

(2)定义了 URL 迹、前缀、公共前缀、最大公共前缀等一系列重要概念,这为用户会话的约简和分组提供了方便;

(3)设计了一个基于 URL 迹的用户会话约简算法,利用该算法得到的用户会话数量大大减少,但覆盖了原先用户会话集请求的所有 URL 并能保持 URL 请求的顺序;

(4)提出了一种用户会话分组和优先排序的方法,此方法可以提高测试的执行效率;

(5)提出了一种利用遗传算法的交叉算子产生新的测试用例的方法,新的测试用例包含了不同用户的请求信息,可以检测不同用户共享数据时可能带来的冲突而产生的错误。即,交叉体现了用户会话之间交换信息的思想,有利于用户会话的交互测试;

(6)为测试的重用和并发(或并行)执行提供了一种策略,可以大大减少测试执行时间,降低测试费用。

利用基于用户会话的测试方法可以分析 Web 应用的一些特性。例如,如果在一个 Web 应用中,大多数识别出的会话所包含的连接数都非常少,这意味着用户通常可能利用较少的 TCP 连接就能获得所需要的数据,请求的外部对象也是有限的,或者用户在某些页上花费了足够长的时间以完成某个事务。

Web 应用的测试是一个复杂的系统工程,要获得一套有效而实用的测试方案并非易事。本文只针对测试用例的测试覆盖率指标分析一个测试用例的优劣,然而还有诸多因素需要考虑,例如各个测试用例本身的运行代价(比如 CPU 时间),包括测试用例的实际执行时间、测试用例的加载时间、保存测试用例的测试状态的时间,以及不同的测试准则对一个测试用例优劣的影响等,将是今后的研究考虑的问题。

References:

[1] Nuzman C, Saniee I, Sweldens, et al. A compound model for TCP connection arrivals with applications to LAN and WAN[J]. Computer Networks, Special Issue on Long-Range Dependent Traffic, 2002,40(3):319-337.

[2] Sprenkle S, Sampath S, Gibson E, et al. An empirical comparison of test suite reduction techniques for user-session-based testing of Web applications, Technical Report 2005-09[R]. University of Delaware,2005.

[3] Elbaum S, Karre S, Rothermel G. Improving Web application testing with user session data[C]//Proceedings of the 25th International Conference on Software Engineering, Portland,Oregon,May 2003, 2003:49-59.

[4] Sampath S, Mihaylov V, Souter A, et al. A scalable approach to user-session based testing of Web applications through concept analysis[C]//Proceedings of the Automated Software Engineering Conference,Sept 2004.

[5] Chen T Y, Lau M F. Dividing strategies for the optimization of a test suite[J]. Information Processing Letters, 1996,60(3):135-141.

[6] Offutt J, Pan J, Voas J M. Procedures for reducing the size of coverage-based test sets[C]//Proceedings of the 12th International Conference on Testing Computer Software, 1995:111-123.

[7] Rothermel G, Harrold M J, Ostrin J, et al. An empirical study of the effects of minimization on the fault detection capabilities of test suites[C]//Proceedings of the International Conference on Software, Maintenance, 1998:34-43.

[8] Wong W E, Horgan J R, London S, et al. Effect of test set minimization on fault detection effectiveness[J]. Software-Practice and Experience, 1998,28(4):347-369.

[9] Nie Changhai, Xu Baowen. A minimal test suite generation me-thod[J]. Chinese Journal of Computers, 2003,26(12): 1690-1696.

[10] Elbaum S, Malishevshy A, Rothermel G. Incorporating varying test costs and fault severities into test case prioritization[C]//Proceedings of the International Conference on Software Engineering, 2001:329-338.

[11] Rothermel G, Untch R H, Chu C, et al. Prioritizing test cases for regression testing[J]. ACM Transactions on Software Engineering and Methodology, 2001,27(10):929-948.

[12] Srivastava A, Thiagarajan J. Effectively prioritizing tests in development environment[C]//Proceedings of ISSTA, 2002: 97-106.

[13] Zhang Xiaofang, Xu Baowen, Nie Changhai, et al. An approach for optimizing test suite based on testing requirement reduction[J]. Journal of Software, 2007,18(4): 821-831.

[14] Rothermel G, Harrold M J. A safe, efficient regression test selection technique[J]. ACM Transactions on Software Engineering and Methods, 1997,6(2):173-210.

[15] Chen Y, Rosenblum D, Vo K. Test tube: A system for selective regression testing[C]//Proceedings of 16th International Conference on Software Engineering, May 1994, 1944:211-222.

[16] Holland J H. Adaptation in natural and artificial system[M]. Michigan, USA: University of Michigan Press,1975.

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

[18] Kung D C, Liu C H, Hsia P. An object-oriented Web test model for testing Web applications[C]//Proceedings of the 1st Asia-Pacific Conference on Web Applications. New York, USA: IEEE CS Press, 2000:111-120.

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

[20] Liu C H, Kung D C, Hsia P. Object-based data flow testing of Web applications[C]//Proceedings of the 1st Asia-Pacific Conference on Quality Software. Hong Kong, China: IEEE CS Press, 2000:7-16.

[21] 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.

[22] Godin R, Missaoui R, Alaoui H. Incremental concept formation algorithms based on galois (Concept) lattices[J]. Computational Intelligence, 1995,11(2):246-267.

[23] Elbaum S, Rothermel G, Karre S, et al. Leveraging user session data to support Web application testing[J]. IEEE Transactions on Software Engineering, May 2005.

[24] Khor S, Grogono P. Using a genetic algorithm and formal concept analysis to generate branch coverage test data automatically[C]//Grünbacher P, Wiels V, Stirewalt K K. Proceedings of the International Conference on Automated Software Engineering. Linz, Austria: IEEE CS Press, 2004:346-349.

[25] Sthamer H H. The automatic generation of software test data using genetic algorithms[D]. Wales, UK: Department of Electronics and Information Technology, University of Glamorgan, 1996.

[26] Berndt D, Fisher J, Johnson L, et al. Breeding software test cases with genetic algorithms[C]//Proceedings of the 36th Hawaii International Conference on System Sciences, 2003:17-24.

[27] Pargas R P, Harrold M J. Test-data generation using genetic algorithms[J]. The Journal of Software Testing, Verification and Reliability, 1999,9(4):263-282.

[28] Jia Xiaoxia, Wu Ji, Jin Maozhong, et al. Some experiment analysis of using generic algorithm in automatic test data generation[J]. Journal of Chinese Computer Systems, 2007,28(3):520-525.

[29] Berndt D J, Watkins A. Investigating the performance of genetic algorithm-based software test case generation[C]// Ramamoorthy C V. Proceedings of the International Symposium on High Assurance Systems Engineering. Tampa, Florida, USA: IEEE CS Press, 2004:261-262.

[30] Stout G A. Testing a Website: Best practices[EB/OL]. A Whitepaper,http://www.reveregroup.com.

附中文参考文献:

[9] 聂长海,徐宝文.一种最小测试用例集生成方法[J].计算机学报,2003,26(12):1690-1695.

[13] 章晓芳,徐宝文,聂长海,等.一种基于测试需求约简的测试用例集优化方法[J].软件学报,2007,18(4):821-831.

[28] 贾晓霞,吴际,金茂忠,等.应用遗传算法自动生成测试数据的实验分析[J].小型微型计算机系统,2007,28(3):520-525.

QIAN Zhongsheng was born in 1977. He received his B.S. degree from Jiangxi Normal University in 1999, M.S. degree from Nanchang University in 2002, and now is a Ph.D. candidate in Computer Science at Shanghai University. He has also been with the School of Information Technology at Jiangxi University of Finance & Economics since 2002. His research interests include software engineering and Web application test, etc.

钱忠胜(1977-),男,江西鹰潭人,博士生,1999 年于江西师范大学计算机科教专业毕业,2002 年于南昌大学获计算机软件与理论硕士学位,目前是上海大学计算机应用专业博士生,同时也是江西财经大学信息学院教师,主要研究领域为软件工程,Web 应用的测试等。

MIAO Huaikou was born in 1953. He received M.S. degree in Computer Science from Shanghai University of Science & Technology in 1986. He is currently a full professor and Ph.D. supervisor in School of Computer Engineering & Science, Shanghai University. His research interests include software engineering and software test automation, etc.

缪淮扣(1953-),男,江苏淮阴人,1986 年毕业于上海科技大学计算机科学系,获计算机应用工学硕士学位。目前是上海大学计算机学院副院长,教授,博士生导师,主要研究领域为软件工程,软件测试自动化等。