第 34 卷 第 6 期 计 算 机 工 程 2008 年 3 月
Vol.34 No.6 Computer Engineering March 2008
·软件技术与数据库· 文章编号：1000—3428(2008)06—0060—03 文献标识码：A 中图分类号：TP301

Web 应用程序建模和测试用例生成方法

刘焕洲，缪淮扣
(上海大学计算机工程与科学学院，上海 200072)

摘 要：根据基于组件的 Web 应用程序的特点，提出逻辑组件的概念。按功能将待测系统划分成逻辑组件，并在功能上将其对应到实际组件。利用有向图表示逻辑组件结构关系，自动机表示逻辑组件的行为关系，自动机的复合表示逻辑组件间的交互操作。根据复合的自动机，利用建立的操作映射表生成测试用例。分析了一个测试准则，并提出逻辑组件测试序列的覆盖度概念。
关键词：Web 应用；组件；测试模型；测试用例

Approach to Modeling and Generating Test Case for Web Application

LIU Huan-zhou, MIAO Huai-kou
(School of Computer Engineering and Sciences, Shanghai University, Shanghai 200072)

【Abstract】This paper proposes a conception of Logic Component (LC), which divides the system-under-test into LCs and ultimately mapping LCs to physical components. Digraphs are used to represent the structure relationship between LCs, and automata are used for action relationship. Interactions between LCs are described by the combination of automata, which use a created action mapping table to generate test cases. This article also proposes a test criteria and a LC sequence coverage according to this criteria.
【Key words】Web application; component; test model; test case

Web应用程序与传统软件有很大差别：服务器端会动态生成客户端执行的内容；两者对应用程序执行时的控制不同；Web应用程序的Session控制、Cookies控制和一些新的安全问题等，都是传统软件所没有的[1]。所以，Web应用程序的测试方法有别于传统软件的测试。本文将Web应用程序划分为逻辑组件，从结构和行为上进行建模，并测试、验证组件之间的交互是否符合需求要求，由单元测试保证组件的质量，由集成测试、系统测试保证“组装”的质量。

1 基本概念

目前有关组件还没有统一的定义，被广泛接受的定义为：一个软件组件是一个用来组装的单元，这个单元具有合约说明的特定接口和明晰的环境依赖性[2]。

在 Web 应用程序中，组件可以理解为是 Java Applet, ActiveX 控件, Java Bean, HTML 模板文件或用任意编程语言编写的程序或程序集合。为了在建模中区别于实际的组件，便于将 Web 应用程序作为黑盒看待，本文提出逻辑组件(Logic Component, LC)的概念。

定义 1 (逻辑组件) 从用户角度，将 Web 应用程序看作黑盒，按照功能将 Web 应用程序细化成的“组件”。逻辑组件由逻辑组件名称和两类接口(输入接口、输出接口)组成。逻辑组件通过不同的动作向输出接口发送数据或从输入接口接受数据。逻辑组件可以是若干个逻辑组件的集合。

按照功能将待测系统划分成 LC 得到的是一个层次化的结构。进行划分的最终目标是：在功能上将不再继续细化的 LC 对应到待测系统中实际的组件。这样就保证 LC 满足“已经充分测试”的假设，可以集中测试 LC 的交互性。

2 逻辑组件的建模

按照逻辑组件对 Web 应用程序的测试进行建模包括 3 方面：建立 LC，建立 LC 间结构关系，建立 LC 间行为关系。

2.1 逻辑组件的建立

例如一个基于 Web 的“测试注意事项专家系统(TAES)”，创建一个新的软件测试项目，回答系统提出的问题后会生成该项目的报告，告知测试该软件项目需要注意的事项。系统提供的功能有：创建新的项目(create project)，打开一个项目并回答问题(open project)，删除一个项目(delete project)，将一个项目改名(rename project)，生成项目报告(generate report)。在 Open project 中，用户可以打开一个项目并回答问题(answer)；回答完毕后可以将项目锁定，防止其他人修改(lock answer)。在 Report 页可以预览(preview)，在预览页面可以打印报告(print report)。

如图 1 所示，在该系统中，将功能划分为逻辑组件 Create, Open, Delete, Rename, Report。Open 又可以进一步细化为 Answer 和 Lock project。Report 可以细化为 Preview。Preview 细化为 Print report。而不再继续细化的 LC 已经过充分的测试，如 Answer, Lock answer 等。

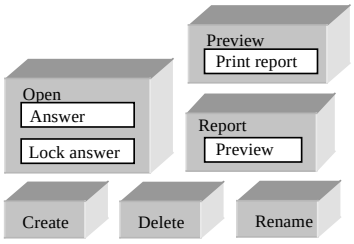

图 1 逻辑组件结构

基金项目：国家自然科学基金资助项目(60373072, 60673115)；上海市教委科技发展基金资助项目
作者简介：刘焕洲(1978－)，男，硕士，主研方向：软件测试，软件形式化方法；缪淮扣，教授、博士生导师
收稿日期：2007-03-25 **E-mail：**LiuHZ@126.com

2.2 逻辑组件间结构关系

逻辑组件的结构关系用一个有向图 LCD 表示：$L=(V, L, E)$，其中，V 是节点的集合，表示划分过程中的所有逻辑组件。$L=\{T1, T2\}$ 是表示任意两个逻辑组件 A 和 B 的关系种类，其中，$T1$ 表示 A 属于 B；$T2$ 表示 A 和 B 同属于一个 LC。即 $T1, T2: V \leftrightarrow V$，定义如下：

$T1 = =\{(x, y) | x \in y,$ 其中，$x \in V, y \in V\}$

$T2 = =\{(x,y) | \exists z \in V,$使得 $x \in z, y \in z$ 且 $x \neq y,$ 其中，$x \in V, y \in V\}$

$E \subseteq V \times V \times \{\text{Yes, No}\}$ 用来表示 LC 间是否要进行交互测试，仅在结构关系为 $T2$ 的 LC 之间进行交互测试，明确 LC 之间不需要进行交互测试时设置为 No。

LC 的交互性是测试的重点，表 1 显示了根据系统功能列出的 LC 结构关系。

表 1　逻辑组件间结构关系

逻辑组件	逻辑组件	类型	是否交互
Open	Create	$T2$	No
Open	Delete	$T2$	Yes
Open	Report	$T2$	Yes
Open	Rename	$T2$	Yes
Open	Answer	$T1$	N/A
Open	Lock answer	$T1$	N/A
Answer	Lock answer	$T2$	Yes
Lock answer	Answer	$T2$	No
Report	Preview	$T1$	N/A
Preview	Print report	$T1$	N/A

2.3 逻辑组件间行为关系

LC 之间的交互行为会由于浏览器的前进、后退、刷新、重新书写地址等功能而变得很复杂。本文为了降低模型的复杂程度、减少生成测试用例的数量，指定 LC 之间需要测试的交互行为，TAES 系统 LC 行为关系如图 2 所示。

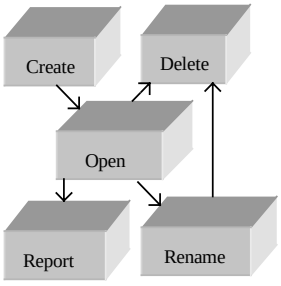

图 2　逻辑组件行为关系

LC 的行为和 LC 间的交互行为可以用有限自动机 M 来描述，定义五元组：$M=(Q, Act, d, I, S)$，其中，Q 是有限状态集；Act 是有限动作集；$\Sigma = ((X \cup \{-\}) \times Act \times (X \cup \{-\})) \setminus (\{-\} \times Act \times \{-\})$ 其中，$X = \{c | c$ 是逻辑组件名称，c 出现在 S 中$\}$；符号“－”表示空白，即没有指定逻辑组件；$d \subseteq Q \times \Sigma \times Q$ 是状态转换集；$I \subseteq Q$ 是非空初始状态集；S 是由逻辑组件名称构成的多元组。

对于逻辑组件 A 和 B，用 $(A, a, -)$ 表示 A 用动作 a 输出；$(-, a, B)$ 表示 B 用动作 a 输入；(A, a, B) 为交互，即 A 用动作 a 输出，同时 B 用动作 a 输入。

根据图 2 所示的逻辑组件行为，建立 LC 的动作，再根据有限自动机定义，逻辑组件的行为自动机如表 2 所示。

表 2　逻辑组件行为自动机

逻辑组件	逻辑组件自动机
Create	({$q0,q1$} , {$a01,a02$} , {($q0,(-,a01,Create),q0$),($q0,(Create,a02,-),q1$)} , {$q0$} ,($Create$))
Open	({$q0,q1,q2,q4$} , {$a02,a03,a04,a06$} , {($q0,(-,a02,Open),q0$),(q 0, (Open, $a03,-),q1$),($q0,(Open,a04,-),q2$),(q 0,($Open,a06,-),q$ 4)} , {q 0} , ($Open$))
Report	({q 0} ,{$a06$} , {(q 0,($-,a06,Report),q$ 0)} , {q 0} ,($Report$))

3　交互测试用例生成

在 Web 应用程序中，组件要支持分布式访问，要测试组件的单节点访问和多节点并行访问。本文提出的模型和测试用例生成只涉及单节点情况。

定义 2 (组件测试序列) 组件交互测试中，组件按先后顺序组成的序列。

定义 3 (待测组件集合) 由进行交互测试的组件组成的不重复集合。

定义序列的 In 操作，用来判断一个序列中的两个相邻的元素是否在另外一个序列中也是作为相邻的两个元素出现。

定义 4 (序列逻辑操作符 In) 设 β 为任意集合，$TA: \text{seq}\beta$ 为序列，$TA(i)$ 表示序列 TA 的第 i 个元素，$i \in 1 \cdots \#TA$。对于 1 个有两个元素的序列

$TB = =\langle a, b\rangle, TB \text{ In } TA \Leftrightarrow \exists i: 1 \cdots \#TA \cdot TA(i) = a \wedge TA(i+1) = b$

在 Web 应用程序中有些组件有依赖关系或约束关系，是按串行顺序操作的。本文提出一个单结点顺序测试逻辑组件的测试准则——单节点测试串行逻辑组件集合。

设 $TR = \{\langle X, Y\rangle : \text{seq}R | X \in R, Y \in R \setminus \{X\}\}$，其中，$R$ 为待测组件集合。

对于 TR 中的每一个元素$\langle X, Y\rangle$，由逻辑组件测试序列组成的集合 J 中都至少存在一个对应的逻辑组件测试序列 T，使得$\langle X, Y\rangle$ In T 成立。形式化表示为

$\forall TS: \text{seq}TR; i: 1..\#TS; \exists T \in J \cdot TS(i) \text{ In } T$

测试准则要求：待测组件集合中任意两个组件之间要进行交互测试，并且还要考虑执行顺序。

生成串行逻辑组件的测试序列也可对逻辑组件采用EC (Each Choice), t-wise等策略，或使用这些策略的组合[1,3]，以在测试效果与测试用例数量之间取得平衡。

3.1 测试用例生成

根据上面给出的准则，结合 LC 行为关系和结构关系，写出需要的组件测试序列：$\langle Create, Open, Report\rangle$, $\langle Create, Open, Rename, Delete\rangle$等。并检查：在 LC 结构关系中指出不进行交互测试的项，不应该出现在任何测试序列中，以减少测试用例。

对于得到的每个序列，由序列中 LC 的自动机复合生成一个新的自动机，用来生成测试用例。

序列$\langle Create, Open, Report\rangle$ 的自动机可以由序列中的 3 个逻辑组件的自动机复合生成，如下所示：

$M1$＝({$q0, q1, q2$}, {$a01, a02, a06$}, {($q0$, (-, $a01$, $Create$), $q0$), ($q0$, ($Open$, $a02$, -), $q1$), ($q1$, ($Report$, $a06$, -), $q2$)}, {$q0$}, ($Create$, $Open$, $Report$))。

将动作映射到具体的操作后，根据自动机设计需要的测试用例。建立 LC 动作映射表，如表 3 所示，根据自动机 $M1$ 可以得到测试用例。逻辑组件 Create 的自动机如图 3 所示。

表 3　逻辑组件动作映射

动作	操作		数据输入/输出
a01	打开 Create 页面		
	Path1	输入{TestData01}	输入：PrjA
		点击“提交”，创建成功	输出：创建成功
	Path2	输入{TestData02}	输入：InvalidName
		点击“提交”，创建失败	输出：创建失败
a02	打开 Open 页面		
	默认选中的项目名称是最近创建的项目		
	点击“打开”，正常打开项目		
	回答所有问题		任意回答(Yes 或 No)
a06	打开 Report 页面		
	默认选中的项目名称是最近打开的项目		
	点击“生成报告”，正常生成项目报告		

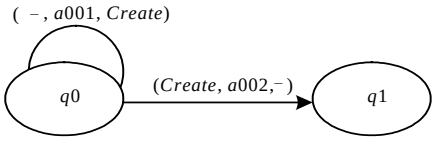

图 3　逻辑组件 Create 的自动机

测试用例的执行步骤为：打开 Create 页面，输入 PrjA，点击“提交”，创建成功；打开 Open 页面，选择栏中默认选中的项目名称是最近创建成功的项目 PrjA，点击“打开”，正常打开项目，回答所有问题；打开 Report 页面，选择栏中默认选中的项目是上一步打开的项目名称 PrjA，点击“生成报告”，正常生成项目报告。

测试用例的验证过程为：(1)项目名称能否接收用户输入的合法与不合法字符。(2)创建成功与否后，Open 页面的默认选中项是否满足设计要求，在这种情况下是否可以正常打开、回答问题并生成项目报告。动作映射表中 *a*01 的操作 Path1 验证基本路径(happy path)。Path2 是测试输入合法性。表 3 中输入数据可扩展为输入空间，以提供更多的数据，如各种控制字符等，达到更好的测试效果。

另外，逻辑组件 Open 的自动机是由逻辑组件 Answer 和 Lock answer 的自动机复合而成。因此，将这两个 LC 生成的测试用例操作步骤与测试用例结合，可得更详细的测试用例。

3.2　测试覆盖度

对LC进行交互测试的最理想情况是对所有LC使用AC(All Combination)策略[4]生成组件测试序列，但是这样代价太大。上文提出的准则减少了组件测试序列的数量，分析该准则的组件序列覆盖度：

假设LC的总数为n。若第i个LC中不再细化的组件用待测组件集合R_i来表示，U_i为R_i中组件的数量，V_i为R_i中不进行交互测试的组件数量，则$U_i - V_i$为需要进行交互测试的组件数量。按照前文给出的准则，R_i内测试序列最少有 $W_i=(U_i-V_i\times((U_i-V_i)-1)$ 个。所有测试序列最少有 $Z=\prod_{i=1}^{n}W_i$ 个。

若生成测试用例过程中使用到的测试序列数量为 G，则序列覆盖度=G/Z。

3.3　提高测试覆盖度的方法

对于生成测试用例过程中未使用到的测试序列，可以进一步肯定或否定这样的测试序列。

(1)肯定：加入到组件结构关系表和行为关系表中，即增大序列覆盖度中 G 的值，提高覆盖度。

(2)否定：加入到组件结构关系表，指出这些组件不需要交互测试，即减小覆盖度中 Z 的值，提高覆盖度。

4　结束语

本文提出的逻辑组件概念，便于用自动机来描述组件行为，用自动机复合来描述组件复合。文中测试用例由组件测试序列得到，提出了一个单结点测试串行逻辑组件集合的准则，并根据此准则得到测试序列覆盖度。

用本文提出的建模方法，可以与 Web 应用程序的功能相对应，便于建模、理解和维护模型。另外，在软件设计有所变动时，可根据结构关系和行为关系判断出受此逻辑组件影响的组件测试序列，并进一步得到受影响的测试用例。

参考文献

[1] WU Ye. Modeling and Testing Web-based Applications[R]. GMU, Tech Rep: ISE-TR-02-08, 2002.

[2] Szperski C. Component Software: Beyond Object-oriented Programming[M]. 2nd ed. New York: ACM Press, 2002.

[3] Grindal M, Lindström B, Offutt J. An Evaluation of Combination Strategies for Test Case Selection[J]. Empir. Software Eng., 2006, 11(4): 583-611.

[4] Rook P. Controlling Software Projects[J]. Software Engineering Journal, 1986, 1(1): 7-16.

(上接第 47 页)

参考文献

[1] Hinden R, Deering S. Internet Protocol Version 6 (IPv6) Addressing Architecture[S]. RFC 3513, 2003-04.

[2] 李领治, 郑洪源, 吴庆丰, 等. 一种基于扩张方法的选播路由算法[J]. 华南理工大学学报, 2006, 34(6): 79-83.

[3] Fang Hao, Zegura E W, Ammar M H. QoS Routing for Anycast Communications: Motivation and an Architecture for DiffServ Networks[J]. IEEE Communications Magazine, 2002, 40(6): 48-56.

[4] 李领治, 郑洪源, 吴笑凡, 等. 应用层 QoS 选播流路由优化系统的构架与实现[J]. 兰州大学学报, 2006, 42(2): 86-91.