基于模型的 Web 应用测试

缪淮扣 陈圣波 曾红卫

(上海大学计算机工程与科学学院 上海 200072)

(上海市计算机软件评测重点实验室 上海 201112)

摘 要 提出了基于模型的 Web 应用测试方法,包括建模、测试用例生成、测试用例的执行、模型以及测试用例的可视化等关键技术.设计并实现一个基于模型的 Web 应用测试系统.以 FSM 作为被测 Web 应用的形式测试模型,集成了模型转换器、测试目标分析器、测试序列生成器、FSM 和测试序列可视化以及 Web 应用测试执行引擎等工具.除支持状态覆盖、迁移覆盖、迁移对覆盖等传统的覆盖准则外,还改进/提出了优化状态迁移覆盖、完整消息传递覆盖、完整功能交互覆盖和功能循环交互覆盖等覆盖准则.该文以兴宁水库移民信息管理系统为例演示了该系统.

关键词 Web 应用;模型转换;测试用例生成;测试执行

中图法分类号 TP311 **DOI 号**: 10.3724/SP.J.1016.2011.01012

Model-Based Testing for Web Applications

MIAO Huai-Kou CHEN Sheng-Bo ZENG Hong-Wei

(*School of Computer Engineering and Science, Shanghai University, Shanghai* 200072)

(*Shanghai Key Laboratory of Computer Software Evaluating and Testing, Shanghai* 201112)

Abstract In this paper, a testing approach to model-based testing for Web applications is proposed which involves in Web application modeling, test generation, test execution and visualization for Web models and test sequences. The authors design and implement a model-based testing system for Web applications while the FSM is regarded as a formal testing model of Web applications under test. And this system integrates Model Transformer, Test Purposes Analyzer, Test Sequences Generator, Visualization tools for FSM and test sequences, Test Execution Engine, etc. Furthermore, the system not only supports the traditional test coverage criteria such as State Coverage, Transition Coverage, Transition Pair Coverage, but also the criteria proposed and improved including Optimized State and Transition Coverage, Complete Message Pass Coverage, Complete Function Interaction Coverage and Function Loop Interaction Coverage. Finally, the authors demonstrate the system taking the Xingning Reservoir Resettlement MIS as our Web application under test.

Keywords Web applications; model transformation; test cases generation; test execution

---

收稿日期:2010-10-30;最终修改稿收到日期:2011-05-21.本课题得到国家自然科学基金(60970007,61073050)、国家"八六三"高技术研究发展计划项目基金(2007AA01Z144)、国家"九七三"重点基础研究发展规划项目基金(2007CB310800)、上海市自然科学基金(09ZR1412100)、上海市科学技术委员会基金(10510704900)及上海市重点学科建设项目基金(J50103)资助.**缪淮扣**,男,1953 年生,教授,博士生导师,中国计算机学会高级会员,主要研究领域为软件工程、软件形式化方法.E-mail: hkmiao@shu.edu.cn. **陈圣波**,男,1975 年生,博士,主要研究方向为软件工程、Web 应用的建模、测试和验证. **曾红卫**,男,1966 年生,博士,研究员,主要研究领域为软件形式化方法、模型验证及测试.

1 引　言

Web 应用已经渗透到国计民生的各个领域，一个非常关键的问题是如何保证 Web 应用的可靠性和质量. 测试是提高软件可靠性和保证软件质量的一种最基本的手段. 目前，Web 应用的测试主要依赖测试工程师的直觉和经验，没有比较系统的方法和工具，测试是一个耗时的、代价昂贵的过程. 许多 Web 应用在没有进行充分测试的情况下投入运营，质量难以保证，导致 Web 应用软件的开发可靠性不高、风险大. 因此，迫切需要实现 Web 应用测试的自动化.

基于模型的语言（如 UML）、模型驱动技术（MDA）和以测试为中心的软件开发技术与方法的兴起和应用以及形式化验证技术的逐步成熟，使基于模型的软件测试方法与技术[1-5]在近几年得到了较广泛的关注，一些理论的和商业的工具相继出现. 基于模型的测试以明确描述系统预期行为的抽象模型为依据，根据模型覆盖准则自动生成抽象的测试用例：输入和预期的输出. 输入部分在被实例化后输入被测系统（the System Under Test，SUT）进行测试. 基于模型的测试实现测试用例生成、测试执行和测试结果判别的自动化.

Web 应用的测试可以分为非功能测试和功能测试[6]. 非功能测试包括性能测试、负载测试、可用性测试、兼容性测试和安全性测试等. 本文考虑的是功能测试.

目前，Web 应用的测试技术大多着重于客户端和服务端的静态测试，包括 link 检查、HTML 验证器、捕获/回放、安全性测试以及负载和压力测试等 Web 应用测试支持工具，大都是基于程序或源代码的静态验证和计量工具，不支持或很少支持 Web 应用的功能测试[7].

Web 应用由复杂的用户界面和不计其数的后端构件以各种方式集成在一起. 基于模型的测试方法为 Web 应用功能测试的自动化提供了一种非常有效的解决方案. 在基于模型的测试中，测试模型和基于测试模型生成的测试用例都是抽象的、独立于平台的，从而是可重用的. 测试执行时通过对测试执行环境的动态配置自动产生实例化的可执行的测试包. 这一特性将大大降低由于 Web 应用的异构性和动态性所带来的测试复杂度. 基于模型的 Web 应用测试方法改变了软件工程中“现在编程，以后测试”的工作方式，使得在整个软件生命期中都可以并行地进行测试工作，从而实现编码前的错误检测，大大降低测试的代价.

基于模型的 Web 应用的建模和测试，国内外有一些相关的研究正在开展[8-13]，Web 应用的测试研究只是分别考虑 Web 应用测试的一个或几个方面，大都没有考虑到 Web 应用中的系统级的完整功能测试问题.

本文主要研究基于模型的 Web 应用软件的测试技术和工具实现. 最终构建一套从模型出发产生测试用例并自动执行测试用例的系统工具，实现 Web 应用测试过程的自动化. 本文主要关注于 Web 应用的导航行为方面. 主要内容和贡献如下：第 2 节给出 Web 应用的建模方法、模型转换方法、模型归约方法，最终得到系统的有限状态机（Finite State Machine，FSM）模型；第 3 节根据已有的或改进的以及本课题组提出的测试准则进行测试生成，得到抽象测试用例；第 4 节给出整个基于模型的 Web 应用测试系统的设计与实现，包括系统框架、测试用例的可视化以及测试执行；第 5 节通过实例演示本工具的主要功能. 最后，对全文进行总结.

2 Web 应用建模和模型转换

基于模型的 Web 应用测试的首要任务是建立 Web 应用的抽象测试模型. 目前，已有不少 Web 应用建模方法，每一种方法都有不同的目的，所关注的 Web 特性也不一样. Isakowitz 等人[14]将关系管理方法论用于描述 Web 应用的设计. Coda 等人[15]给出了面向对象模型 WOOM，用高层抽象的原始实体来描述 Web 应用的开发. Gellersen 等人[16]提出了一种 Web 复合方式来结构化 Web 的开发过程. Conallen 等人[17]扩展 UML 对 Web 应用的体系结构进行建模. 文献[18-20]采用 statecharts 对 Web 导航、Web 元素以及这些元素之间的交互进行建模. Kung[21-22]等人提出了一种基于包括对象关系图、对象状态图、脚本簇图和网页导航图等多模型的测试生成方法. Andrews 等人[23]分析了构成 Web 应用的网页和软件构件之间的 8 种连接关系，提出了一种基于有限状态机（FSM）的 Web 应用建模和测试用例生成方法. 该方法通过对 Web 应用进行功能簇和逻辑网页的划分并用带约束的分层 FSM 表示逻辑网页及逻辑网页间的导航关系. 这种方法没有进一步考虑 Web 应用中软件构件的交互和合成

的测试问题.另外,Ricca 等人[24]用决策表来对 Web 应用中每个页面的行为进行建模,利用决策表产生测试用例.这一方案仅仅是在单元测试层面上对较为简单的页面有效,无法解决整个软件的功能测试问题.然而,这些方法很少有从 Web 应用的行为和功能上进行建模并提出功能测试方案的.

采用 FSM 模型描述软件的行为在实践中得到广泛的认同[25].FSM 提供了一个基本机制来对软件的复杂行为进行建模,不需要考虑该软件的底层实现[26],可以有效避免和具体实现相关的问题.FSM 有成熟的理论基础,并且可以利用形式语言和自动机理论来设计、操纵和分析,特别适合描述反应式软件系统.已有许多研究提出了一些直接从 FSM 模型派生测试用例的方法[27-29],并且这些方法已经趋于成熟.尽管 FSM 提供了一个对复杂 Web 应用行为进行建模的理论基础[30],但如何有效地用 FSM 来对 Web 应用建模以及如何有效来避免状态空间爆炸问题等给研究者提出了挑战.

我们的建模方法是用 UML 的不同图形,从不同的角度,针对 Web 应用的特点建立模型.本文采用 UML 状态图对 Web 应用的行为进行建模,然后将 UML 状态图转换成 FSM.为了克服或缓解 FSM 状态空间爆炸问题,采用测试目标归约的方法,用 UML 顺序图来描述测试目标,并从该 UML 顺序图产生测试用例规约,然后与由 UML 状态图转换成的 FSM 模型进行模型归约组合产生基于测试目标的 FSM,也就是约简的 FSM.然后将它们统一到一个形式化抽象模型 FSM 上.本文给出了模型投影、同步积以及去层次化等方法和技术.测试用例是根据抽象模型 FSM 生成的.

2.1 UML 状态图模型的形式化

为了便于理解以及进行后续的模型转换,先给出如下概念.

定义 1. 一个 FSM 模型是一个五元组 $A=(Q,L,\delta,q_0,q)$,其中 Q 是有限个状态的集合;L 是 A 上有限个迁移标签的集合;$\delta:Q\times L\to Q$ 是有限个迁移的集合,表示前状态通过标签中给出的事件触发使系统进入后状态;$q_0\in Q$ 是 A 的开始状态,$q\in Q$是 A 的接受状态,即终态,系统进入该状态时将停止接受任意事件触发.

假设迁移 $t(s,l,s')\in\delta$,记 $source(t)=s$,$target(t)=s'$,$label(t)=l$.

从 FSM 的定义可以看出,状态均为基本状态或简单状态,即无嵌套状态,不存在子状态与子 FSM.由于 UML 状态图中会出现复合状态,去层次化是必需的.为了不丢失源模型的语义,必须先获取状态图中各状态之间的层次关系,再利用适当的模型转换规则生成无层次的 FSM 模型.为此,引入中间模型:层次有限状态机 HFSM(Hierarchical FSM),以记录目标模型中无法保留的层次信息.HFSM 可以有内部结构,可以将它看作许多个 FSM 模型以串行和(或)并行方式组合而成的复合模型,非最底层的状态可由其直接下层的子状态机表示,其定义需要在函数 ϕ 的基础上给出.

定义 2. 给定一个有限状态机的集合 $F=\{A_1,A_2,\cdots,A_n\}$,$Q(A_i)$ 表示集合中任意有限状态机 A_i 的状态集合,则

$\phi:\bigcup_{A\in F}Q(A)\to\mathbb{P}(F)$ 是集合 F 上的一个组合函数当且仅当

(1) $\exists_1 A\in F\wedge A\notin\bigcup\mathrm{ran}(\phi)$.ran 是值域函数,$A$ 表示最高层次的有限状态机.

(2) $\forall A\in\bigcup\mathrm{ran}(\phi)\cdot\exists_1 s\in\bigcup_{A'\in F\setminus\{A\}}Q(A')\cdot A\in\phi(s)$.

(3) $\forall S\subseteq\bigcup_{A\in F}Q(A)\cdot\exists s\in S\cdot S\cap\bigcup_{A\in\phi(s)}\cdot Q(A)=\varnothing$.

定义 3. HFSM 是一个二元组 (F,ϕ),其中 F 是一个有限的有限状态机集合,对于每两个顺序有限状态机 A_i,A_j $(0<i<n,0<j<n,i\neq j)$ 都有 $Q(A_i)\cap Q(A_j)=\varnothing$,$\phi$ 是集合 F 上的一个组合函数.设 s 为该 HFSM 中的任意状态,若 $\phi(s)\neq\varnothing$,则 s 为复合状态.

在这些模型定义的基础上,便可开始模型形式化的描述.

根据 HFSM 的定义,由 UML 状态图的拓扑结构信息得到一个有限状态机集合上的组合函数后,便可以利用 HFSM 数学形式表示该拓扑结构.而组合函数的建立过程自顶向下,将每层的复合状态映射到其对应的子 FSM,然后将其作为一个元素添加到组合函数中.若复合状态 s 是一个“或”状态,它对应的子 FSM 为 A_i,那么就有 $\phi(s)=A_i$ 且 $F=F\cup\{A_i\}$;若复合状态 s 是一个“与”状态,它的各个域对应的子 FSM 分别是 $A_1,A_2,\cdots,A_n$,那么就有 $\phi(s)=\{A_1,A_2,\cdots,A_n\}$ 且 $F=F\cup\{A_1\}\cup\{A_2\}\cup\cdots\cup\{A_n\}$.状态图中每一个被初始状态指向的状态是其所在层次的对应 FSM 的开始状态,而指向最终状态的状态则转换成为其所在层次的对应 FSM 的接受状态.

这样,原 UML 状态图的拓扑结构便完全由其

对应的 HFSM 模型存储起来，在这个 HFSM 的基础上，便可获取原 UML 状态图中任意两个状态间的层次关系，这个层次关系可由函数χ表示.

定义 4. 给定一个层次有限状态机 $HFSM(F,\phi)$，函数χ：

$$\bigcup_{A\in F}Q(A)\to\mathbb{P}(\bigcup_{A\in F}Q(A)),$$

$$\chi(s)=\{s'\mid\exists A\in F\cdot A\in\phi(s)\wedge s'\in Q(A)\}.$$

于是，UML 状态图的图形信息不仅形式化为数学表达形式，而且可以利用自定义函数判断状态间的关系并得到任意状态的所有子状态及其父状态.

定义 5. 给定一个层次有限状态机 $HFSM(F,\phi)$，满足条件 $C\subseteq\bigcup_{A\in F}Q(A)$的状态集合 C 是它的一个格局(Configuration)当且仅当：

(1) $\exists_1 s\in Q(\phi_{\text{root}})\cdot s\in C$；

(2) $s\in C\wedge A\in\phi(s)\Rightarrow\exists_1 s'\in Q(A)\cdot s'\in C$；

(3) $s\in C\wedge\exists s'\cdot s\in\chi(s')\Rightarrow s'\in C$.

定义 6. 给定一个层次有限状态机 $HFSM(F,\phi)$和它所有格局的集合 C，对于其任意一个状态 s，函数 $config$：

$$\bigcup_{A\in F}Q(A)\to\mathbb{P}(\bigcup_{A\in F}Q(A)),$$

$$config(s)=\{c_i\mid c_i\subseteq C\wedge s\in c_i\}.$$

定义 7. 给定一个层次有限状态机 $HFSM(F,\phi)$，其任意一个状态 sd 的缺省格局(Default Configuration)可表示为函数 $deconfig$：

$$\bigcup_{A\in F}Q(A)\to\mathbb{P}(\bigcup_{A\in F}Q(A)),$$

$$deconfig(sd)=X\Leftrightarrow\exists_1 X:config(sd)\cdot$$

$$\forall s\cdot(s\in X\wedge sd\notin\chi^*(s)\Rightarrow\bigcap q_0(\phi_i(s))\subseteq X).$$

定义 8. 给定一个 UML 状态图及其任意一条迁移 t，$Uexit$ 是指状态集合 $exit=\{exit_i\mid\forall j:N\cdot src_j(t)\in\chi^*(exit_i)\wedge dst_j(t)\notin\chi^*(exit_i)\}$中层次最高的状态，$Uenter$ 是指状态集合 $enter=\{enter_i\mid\forall j:N\cdot src_j(t)\notin\chi^*(enter_i)\wedge target_j(t)\in\chi^*(enter_i)\}$中层次最高的状态.

由以上定义可知，一个格局实际上是由层次有限状态机中的 $N(N\geq1)$个状态组成，它代表着某一时刻系统中所有被激活的状态. 所有格局都含有层次有限状态机顶层 FSM 的一个状态，并且，若某复合状态存在于某格局中，则其对应的各 FSM 中均有一个状态存在于该格局中. 函数 $config$ 是从任意状态 s 到所有包含 s 的格局集合的映射，函数 $deconfig$ 是从任意状态 sd 到包含 sd 的缺省格局的映射. $Uexit$ 表示任意迁移 t 离开的所有状态中层次最高的状态，则 $Uenter$ 表示任意迁移 t 进入的所有状态中层次最高的状态.

目标 FSM 模型的每个状态均是原 UML 状态图对应 HFSM 模型的一个格局，那么迁移便是触发一个格局到另一个格局的变化，由于格局是 HFSM 模型中数个状态的集合，所以原 UML 状态图中的每个迁移可被映射为目标 FSM 模型的数个迁移，而此数量由原 UML 状态图中的该迁移的源状态所属的格局数量决定. 设 $confTranSet$ 为目标 FSM 模型的迁移集合，可在前期工作得到的原 UML 状态图拓扑结构数学模型的基础上计算出集合 $confTranSet$.

在构建出目标 FSM 模型的迁移集合 $confTranSet$ 后，便可将所有与 $confTranSet$ 中迁移相关的状态置于一个集合内，这便是目标 FSM 模型的状态集合. 而其初始状态 $InitState$ 和接受状态 $AccState$ 则可由以下表达式确定：

$$InitState=deconfig(q_0(\phi_{\text{root}})),$$

$$AccState=config(q(\phi_{\text{root}})).$$

上述方法基本上将原 UML 状态图的迁移与状态映射为目标 FSM 模型的基本元素，但还有一种状态没有考虑到，这就是历史状态. 由于历史状态与普通状态的语义差异很大，故不在此处提及，而是为其专门设计一种转换方法，并于模型形式化的最后阶段执行.

至此，与原 UML 状态图模型语义等价的 FSM 模型就创建完成.

2.2　UML 顺序图模型的形式化

顺序图(sequence diagram)是强调消息的时间次序的交互图. 顺序图的建模对象是系统中指定对象的交互行为所构成的场景，从而反映与此场景相对应的系统功能，非常适合于描述测试目标(test purposes). 本文采用 UML 顺序图来描述测试目标，并将该 UML 顺序图转换成描述该测试目标的 FSM. 然后与 Web 应用的原系统 FSM 模型进行归约组合，即把 UML 顺序图转换得到的描述测试目标的 FSM 投影到 Web 应用的系统 FSM 模型，产生基于测试目标的 FSM. 在此，引入用于存储 UML 顺序图的描述结构.

定义 9.　一个顺序图是一个三元组 (Obj,Msg,Act)，其中 Obj 是在该顺序图描述场景中出现的对象集合；Msg 是在该顺序图描述场景中出现的消息集合；Act 是该顺序图描述场景中出现的发送并接收消息的动作集合，其中每一个元素可表示成

为一个四元组($obj_{origin}:Obj$, $obj_{dest}:Obj$, $msg:Msg$, $order$),其中 obj_{origin} 是发送消息的对象,obj_{desk} 是接收消息的对象,msg 是发送的消息内容,$order$ 是一个数字,它用来表示这个动作在顺序图中出现的所有动作中的次序编号.

对于任意一个 $act_i \in Act$,假设它可写作四元组(obj_1, obj_2, msg_i, num),那么 $orgin(act_i)=obj_1$, $dest(act_i)=obj_2$, $message(act_i)=msg_i$, $order(act_i)=num$,其中 num 可由该动作所在位置确定:$\forall act_i, act_j \in Act \cdot (act_j$ 的位置在 act_i 的直接下方$) \Rightarrow order(act_j)=order(act_i)+1$.

虽然所有的 UML 顺序图均用以描述对象间的交互活动,但展现的场景不同,情况也不尽相同.一种情况是顺序图表示待检测系统内部对象之间的交互,那么在形式化过程中,它的所有元素都需考虑在内;另一种情况是它表示待检测系统与系统以外的对象间的交互,这时只需提取那些与代表系统生命线的直线相交的动作,并以发出或接收将这些动作分类(任意 $msg_i \in Msg$ 前的"?"符号表示该消息对应的接收动作,任意 $msg_i \in Msg$ 前的"!"符号表示该消息对应的发出动作).

由于顺序图中对于后续模型归约组合的有用信息只有消息,因此目标 FSM 模型中的状态无需具有任何含义.本文用从序号 0 开始的数字来表示,以简化算法.下面是第 1 种情况下 UML 顺序图的形式化机制:创建一个新的 FSM 模型,写作(Q, L, δ, q_0, q),并进行如下初始化:$Q=\{0\}$, $L=\varnothing$, $\delta=\varnothing$, $q_0=q=0$.遍历 UML 顺序图的动作集合 Act 并依据其中每个元素 $act(act \in Act)$ 对新建的 FSM 模型进行如下操作:$Q=Q \cup \{order(act)\}$, $L=L \cup \{?message(act)\}$, $\delta=\delta \cup \{(order(act)-1, ?message(act)) \rightarrow order(act)\}$, $q=order(act)$.这样得到的 FSM 模型便是转换后的形式化的测试用例规格说明.

在第 2 种情况下,有价值的信息仅限于待检测系统发出和收到的消息,这些消息实际上代表着系统的行为.正因为我们需要将原 UML 顺序图中的每个动作拆解成为两个有向动作,即一个发出动作与一个接收动作,消息的顺序已不再能单从其对应动作的位置来判断了.

虽然以上规则可以判定大部分动作的先后顺序,但刚才提到的情景只有相关消息的传递时间已知的情况下才能决定它们相应发出和接收动作的排序.因此,除非获得时间信息,否则需要将所有可能的动作序列均写出,再分别建立对应的 FSM 模型.对于根据动作顺序判定规则得出的任意动作序列 s_i,建立一个新的 FSM 模型并初始化为:$Q=\{0\}$, $L=\varnothing$, $\delta=\varnothing$, $q_0=0$, $q=0$,采用上述方法可遍历 s_i 中的所有元素.这样,便把第 2 种情况的 UML 顺序图(测试目标)形式化为 FSM 模型.

2.3 模型归约组合

由于顺序图表示的动作序列的不唯一性,可能对于一个测试可能目标会产生多个 FSM 模型,这些模型都将分别被当作一个形式化后的测试目标对系统 FSM 模型进行归约组合操作.因此,下面给出的归约组合机制是单个测试目标的 FSM 模型对原系统 FSM 模型的归约组合操作,若需多次归约组合,叠加使用即可.

得到了形式化的测试目标后,便可以开始对原系统 FSM 模型进行归约组合.首先引入一些后面要用到的基本概念.

定义 10. 给定一个有限状态机 $A=(Q, L, \delta, q_0, q)$,一个执行片段 $frag=q_1 l_1 q_2 l_2 \cdots q_n$ 是状态与迁移交替有限序列 $q_i l_i q_{i+1} l_{i+1} \cdots q_n (0 \le i \le n)$ 满足条件 $((q_i, l_i, q_{i+1}) \in \delta (0 \le i \le n)$,并且 $q_1=q_0$.如果执行片段满足 $q_n=q_0$ 或 $q_n=q$,那么称 $frag$ 是 A 的一个运行.

定义 11. 给定两个序列 S_1 与 S_2,定义关系 $\_\propto\_$: $sequence \leftrightarrow sequence$

$S_1 \propto S_2 \Leftrightarrow \forall item_i, item_j \in ran(S_1) \wedge S_1^{-1}(item_i) < S_1^{-1}(item_j) \cdot item_i, item_j \in ran(S_2) \wedge S_2^{-1}(item_i) < S_2^{-1}(item_j)$.

定义 12. 给定一个有限状态机 $A=(Q, L, \delta, q_0, q)$ 及其所有执行片段组成的集合 $Frag_A$,函数 $reach: Q \rightarrow Q$ 定义为

$reach(q_i)=\{q_j \in Q \mid \exists frag \in Frag \cdot q_i, q_j \in frag \wedge i<j\}$.

定义 13. 给定两个有限状态机 $A=(Q, L, \delta, q_0, q)$, $A'=(Q', L', \delta', q_0', q')$, A 与 A' 的积 $A \times A'$ 也是一个有限状态机模型:$(Q_{A\times A'}, L_{A\times A'}, \delta_{A\times A'}, q_{0(A\times A')}, q_{A\times A'})$,其中

$Q_{A\times A'}=Q \times Q'$; $L_{A\times A'}=L \cup L'$;

$\delta_{A\times A'}=\{((q_i, q_i'), \Delta l) \rightarrow (q_j, q_j') \mid q_i, q_j \in Q \wedge q_i', q_j' \in Q' \wedge ((((q_i, ("?" \cup \Delta l)) \rightarrow q_j) \in \delta) \wedge ((q_i', ("!" \cup \Delta l)) \rightarrow q_j') \in \delta') \vee (((q_i, ("!" \cup \Delta l)) \rightarrow q_j) \in \delta \wedge ((q_i', ("?" \cup \Delta l)) \rightarrow q_j') \in \delta')\} \cup \{((q_i, q_i'), \Delta l) \rightarrow (q_j, q_i') \mid q_i, q_j \in Q \wedge q_i' \in Q' \wedge ((q_i, \Delta l) \rightarrow q_j) \in \delta\} \cup \{((q_i, q_i'), \Delta l) \rightarrow (q_i, q_j') \mid q_i \in Q \wedge q_i', q_j' \in Q' \wedge ((q_i',$

$\Delta l)\rightarrow q_j')\in\delta\}$；

$q_{0(A\times A')}=(q_0,q_0')$；

$q_{A\times A'}=(q,q')$.

定义 14.　给定一个有限状态机 $A=(Q,L,\delta,q_0,q)$及其所有执行片段组成的集合 $Frag_A$，对于每一个 $frag\in Frag_A$，可将其写作 $q_i l_i q_{i+1} l_{i+1}\cdots q_n$ $(0\leq i\leq n)$，它的所有标签组成的序列称为这个执行片段 $frag$ 的轨迹，记为 $trace(frag)=l_i l_{i+1}\cdots l_{n-1}$.

定义 15.　给定两个有限状态机 $A=(Q,L,\delta,q_0,q)$，$A^*=(Q^*,L^*,\delta^*,q_0^*,q')$，设 $Frag_A$ 与 $Frag_{A\times A^*}$ 分别是 A 与 $A\times A^*$ 的所有执行片段组成的集合，Cir 是 $A\times A^*$ 的所有运行组成的集合. 对于每一个执行片段 $frag\in Frag_{A\times A^*}$，它在 A 上的投影是其出现于 A 上的那部分标签序列，数学表达式可写为. $\pi_A(trace(frag))=(frag)\backslash\{l\mid l\in A\wedge l\notin A^*\}$.

函数 $cover:(Cir,Frag_A)\rightarrow 0/1$

$cover(cir,frag)=\{1\mid cir\in Cir\wedge frag\in Frag_A\wedge\exists frag'\in Frag_{A\times A^*}\cdot frag'\propto cir\wedge\pi_A(trace(frag'))=trace(frag)\}$

FSM 模型的执行片段可以代表其描述的对应系统的行为，因此，归约组合原 FSM 模型意味着，经归约组合后的模型应该包括描述测试目标中要求测试的功能的所有元素，并且将其余无关元素全部清除.

给定一个执行片段 $frag$ 和一个运行 cir，表达式 $frag\propto cir$ 与 $cover(frag,cir)$的唯一不同点在于，前者同时考虑 $frag$ 与 cir 的状态与迁移，以及它们之间的关系，而后者仅关注执行片段与运行中迁移的标签序列间的关系，并不包含相关的状态信息. 然而，我们在确定归约组合后模型中需要保留的标签序列后，将会在创建新模型时加入与这些标签相关的状态信息，这便意味着，要得到用户要求的简单模型，可以首先计算出一个模型 A^* 使得操作 $A\times A^*$ 满足如下条件（这里用 $Cir_{A\times A^*}$ 表示模型 $A\times A^*$ 的所有运行组成的集合）：

$\forall frag\in Frag_{RD}\cdot(\exists cir_A\in Cir_A\wedge frag\propto cir_A)\cdot(\exists cir\in Cir_{A\times A'}\cdot cover(cir,frag))$ 以及 $\rightarrow\exists A^{*'}\cdot(\forall frag\in Frag_{RD}\cdot(\exists cir_A\in Cir_A\wedge frag\propto cir_A)\cdot(\exists cir_{A'}\in Cir_{A\times A^{*'}}\cdot cover(Cir_{A\times A^{*'}},frag))\cdot(\exists frag'\in Frag_{RD}\cdot(\exists cir_{A'}\in Cir_{A'}\cdot frag'\propto cir_{A'})\cdot(\rightarrow\exists cir\in Cir_{A\times A^{*'}}\cdot cover(cir,frag'))))$.

当测试目标指定的执行片段确定后，便可构造出模型 A^*，它是由所有包含这些片段的运行轨迹所组成的，其所需满足的性质可表示为如下形式化的数学表达式：$(\forall l\in L_A\cdot l\in cir\in\{cir\in Cir_A\mid\exists frag\in Frag_{RD}\cdot trace(frag)\propto trace(cir)\}\cdot l\in L_{A^*})\wedge(\forall l\in L_A\cdot l\in cir\in\{cir\in Cir_A\mid\rightarrow\exists frag\in Frag_{RD}\cdot trace(frag)\propto trace(cir)\}\cdot l\notin L_{A^*})$. 由此可以得出，整个归约组合工作应该从以下两个集合的构建开始：集合 Cir^*，它是一组满足特定条件的运行所组成的集合$\{cir\in Cir_A\mid\exists frag\in Frag_{RD}\cdot trace(frag)\propto trace(cir)\}$；集合 $\overline{Cir^*}$，它是集合 Cir^* 的补集，可表示为$\{cir\in Cir_A\mid\rightarrow\exists frag\in Frag_{RD}\cdot trace(frag)\propto trace(cir)\}$.

但是，在求解上述两个集合之前，我们必须注意到由模型形式化后得到的原系统 FSM 模型中，各迁移的标签是不包含其类属信息的，即无法判断该迁移对应的触发事件是输入、输出或是内部事件. 另一方面，计算 $A\times A^*$ 时是将两者迁移中的输入输出事件进行比较与匹配，并合并相同的迁移项. 另外，对测试目标进行形式化时，由于顺序图的固有属性就是描述了消息发出与接收，其转化而来的 FSM 模型也是包含了输入输出信息的，我们需要将它与原 FSM 模型进行比较与匹配方能获取上述两个集合. 因此，在原 FSM 模型中添加迁移上标签的输入输出类别十分必要. 根据原 UML 状态图中迁移标签各部分的含义，事件代表对系统的外部刺激，即输入；动作则代表系统对该外部刺激的反应，即输出. 在模型形式化的过程中，原 UML 模型迁移与转换后 FSM 模型中对应迁移的标签是一致的，故原系统 FSM 模型的迁移标签也可按 UML 状态图语义加以分解. 我们在此把事件视为输入消息，而把动作视为输出消息，并标记于原系统 FSM 模型中.

根据上述方法，可计算集合 Cir^* 与 $\overline{Cir^*}$ 此时便可根据这两个集合创建模型 A^*，使操作 $A\times A^*$ 执行后的结果模型只描述测试目标中要求的功能. 为此，我们设计了一个计算模型 A^* 中应存在的迁移序列的算法，大致原理是：每一个在集合 Cir^* 的任意运行中出现的标签都在模型 A^* 中，而那些只在集合 $\overline{Cir^*}$ 的运行中出现却不在集合 Cir^* 的任意运行中出现的标签则被排除在模型 A^* 之外.

根据上述方法对集合 $\overline{Cir^*}$ 中每一个元素，可计算出所有原 FSM 模型中应保留在归约组合后模型中的迁移的集合. 然后，识别出该集合中所有附带输入输出标记的迁移，并将每个标记取反，即输入符号改为输出符号，输出符号则改为输入符号，经此取反

处理后的迁移集合就构成了模型 A^*. 最后,对模型 A 与模型 A^* 执行操作 $A\times A^*$,计算结果即为按测试目标归约组合后的 FSM 模型 A'. 但此时的 A' 也将输入输出标记保留,标签处于分离状态,而且,由于它是两个模型执行 $A\times A^*$ 而来,故状态是复合的,其名字为两个原状态名的叠加. 这样,就完成了模型的归约组合.

3 测试生成

我们针对的是在某一抽象层次上的 FSM 的测试用例生成. 把测试用例的生成看成是从 FSM 中选择满足给定覆盖测试准则的有限执行序列的过程.

软件的形式模型或规格说明是自动生成测试用例的基础,这种技术被称为基于模型的测试. 有限状态机 FSM、标签迁移系统 LTS 和面向模型的规格说明语言如 VDM-SL、Z 或 Alloy 都可作为测试自动生成的形式模型[31-37]. FSM 模型具有成熟的理论基础,可以利用形式语言和自动机理论来设计、操纵和分析,便于用模型检验技术进行验证,又有可视化的效果,非常适合用来描述软件和构建测试模型.

在基于 FSM 的 Web 应用测试用例生成中,我们的方法是解析 FSM 的 SCXML 规格说明,生成满足测试准则的测试用例. 本节生成的测试用例都是抽象的,需要具体化才能执行. 通常,若不引起误解,就用测试用例来表示抽象测试用例的概念.

定义 16(迁移序列). 在 FSM 中,由若干个从一个状态到另一个状态的迁移组成的序列称为迁移序列.

定义 17(迁移路径). 在 FSM 中,从初始状态出发的一个迁移序列称为迁移路径.

定义 18(完全路径). 在 FSM 中,最后一个迁移指向终止状态的一个迁移路径称为完全路径.

本节只涉及抽象测试用例的生成. 得到的测试用例经实例化后可成为具体测试用例. 一个抽象测试用例可以实例化多个具体测试用例.

由迁移组成的序列隐含地包含了状态信息,因为一个迁移必定存在它的激发源(即源状态),也必定存在它的迁移目标(即目标状态). 另一方面,状态到状态的转换,必定要经过一个特定的迁移. 因此,基于以上考虑,本文在测试用例生成中,给出如下约定.

约定. 在根据测试准则生成测试用例时,最终测试用例约定统一写成:state〈transition〉state〈transition〉…state.

为了便于理解后续测试用例的生成方法,下面以"兴宁市水库移民信息管理系统"的登录模块为例来进行说明(如图 1 所示),图 1 中的 n 表示 username,p 表示 password. 用户在 Web 浏览器的地址栏输入相应的 URL,回车后,系统进入主页 MainPage,用户点击 login 链接后,登录页面 Login 将显示在 Web 浏览器中,在 Login 页面上有一个 text field 和一个 password field,分别用于输入用户名和密码. 当用户输入完用户名,光标移到 password field 时,此时,系统自动将对用户名进行语法检查(Checking),检查完毕后,系统将重新返回(return)到 Login 页面,并在 Login 页面上显示检查的结果. 同理,用户输入完密码后,系统也自动对密码进行语法检查. 这样,首先保证用户名和密码的语法正确,然后点击 Login 页面上的提交按钮 submit,系统将把用户名和密码一起发送到服务器进行验证,如果用户名和密码都正确,用户将进入管理中心 ManageCenter 页面. 否则,系统将给出一个出错页面 Error,当点击 Error 页面上的 OK 按钮后,系统又重新回到 Login 页面.

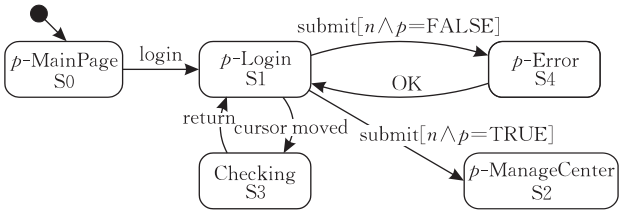

图 1 系统登录

状态覆盖测试准则. 要求测试用例集 TS 使 FSM 中的每一个状态至少被访问一次. FSM 中的每一个状态是可达的,因而每一个状态被访问一次是很容易做到的. 状态覆盖测试准则是这些测试准则中最简单、最容易被满足的测试准则,需要的测试用例也往往是最少的.

根据状态覆盖测试准则可以得到以下测试用例:

TC1: S0→S1→S3→S1→S2;

TC2: S0→S1→S4;

TC3: S0→S1→S3→S1→S3→S1→S3→S1→S2;

…

根据约定,上述测试用例应写成

TC1: S0〈login〉S1〈cursor moved〉S3〈return〉S1〈submit[$n\wedge p$=TRUE]〉S2;

TC2：S0〈login〉S1〈submit[$n \wedge p$=FALSE]〉S4；

TC3：S0〈login〉S1〈cursor moved〉S3〈return〉S1〈cursor moved〉S3〈return〉S1〈cursor moved〉S3〈return〉S1〈submit[$n \wedge p$=TRUE]〉S2；

…

迁移覆盖测试准则. 要求测试用例集 TS 使 FSM 中的每一个迁移至少被激活一次. 先使系统到达某一个状态(当前状态),如果一个迁移的触发事件被接受,并且这个迁移的卫式条件的值为真,则这个迁移被激发,将执行迁移的动作,转换到目标状态. 迁移覆盖测试准则也比较简单、比较易被满足,需要的测试用例也往往是比较少的.

根据迁移覆盖测试准则可以得到以下测试用例：

TC1：login→cursor moved→return→submit[$n \wedge p$=TRUE]；

TC2：login→submit[$n \wedge p$=FALSE]→OK；

…

根据约定,上述测试用例应写成：

TC1：S0〈login〉S1〈cursor moved〉S3〈return〉S1〈submit[$n \wedge p$=TRUE]〉S2；

TC2：S0〈login〉S1〈submit[$n \wedge p$=FALSE]〉S4〈OK〉S1；

…

虽然状态覆盖和迁移覆盖这两个测试准则都比较简单且容易被满足,但只从状态覆盖或迁移覆盖测试准则产生测试用例的方法还是有它的适用范围,因为不同应用领域对软件质量的要求不同以及对软件测试费用的开销不等.

迁移对覆盖测试准则. 要求测试用例集 TS 使 FSM 中的每一对相邻的迁移(〈射入迁移,射出迁移〉)至少测试一次,也就是说,迁移之间的交互至少要测试一次. 迁移对覆盖检查的是状态之间的接口.

如图 1 所示,存在以下迁移对：

login→cursor moved；

login→submit[$n \wedge p$=TRUE]；

login→submit[$n \wedge p$=FALSE；

cursor moved→return；

return→cursor moved；

return→submit[$n \wedge p$=TRUE]；

return→submit[$n \wedge p$=FALSE；

submit[$n \wedge p$=FALSE]→OK；

OK→cursor moved；

OK→submit[$n \wedge p$=TRUE]；

OK→submit[$n \wedge p$=FALSE]；

根据迁移对覆盖测试准则可以得到以下测试用例：

TC1：S0〈login〉S1〈cursor moved〉S3〈return〉S1〈submit[$n \wedge p$=TRUE]〉S2；

TC2：S0〈login〉S1〈cursor moved〉S3〈return〉S1〈submit[$n \wedge p$=FALSE]〉S4〈OK〉S1〈cursor moved〉S3〈return〉S1；

TC3：S0〈login〉S1〈cursor moved〉S3〈return〉S1〈submit[$n \wedge p$=FALSE]〉S4；

TC4：S0〈login〉S1〈submit[$n \wedge p$=FALSE]〉S4；

TC5：S0〈login〉S1〈submit[$n \wedge p$=TRUE]〉S2；

…

在软件中的许多错误可能是因为规格说明中状态之间复杂的交互关系(接口)引起的,为了检查这些交互类型的错误,可以使用迁移对覆盖测试准则生成的测试用例集,该准则要求所有的迁移都是使能的. 显然,满足迁移对覆盖测试准则的测试用例集也一定满足状态覆盖和迁移覆盖这两个测试准则,但它们测试软件的方式有很大的不同. 从上面得到的测试用例可以看出,TC3 是 TC2 的一部分,显然,应该对测试用例集进行优化,基于此,系统设计给出了优化的状态迁移覆盖准则.

优化的状态迁移覆盖准则. 要求对于测试用例集 TS 使 FSM 中的对于任何两条测试序列 $tc1$ 和 $tc2$,都会满足关系：$tc1$ 与 $tc2$ 是两个不相同的测试序列,其中 $tc1$ 序列不是 $tc2$ 的一部分,$tc2$ 也不是 $tc1$ 的一部分.

基于此,上述的测试用例集可写成：

TC1：S0〈login〉S1〈cursor moved〉S3〈return〉S1〈submit[$n \wedge p$=TRUE]〉S2；

TC2：S0〈login〉S1〈cursor moved〉S3〈return〉S1〈submit[$n \wedge p$=FALSE]〉S4〈OK〉S1〈cursor moved〉S3〈return〉S1；

TC3：S0〈login〉S1〈submit[$n \wedge p$=FALSE]〉S4；

TC4：S0〈login〉S1〈submit[$n \wedge p$=TRUE]〉S2；

定义 19(状态迁移序列). 在 FSM 中,迁移序列是从一个状态到另一个状态由状态和迁移组成的序列称为状态迁移序列.

定义 20(消息传递序列). 在 FSM 中,消息传递序列是状态根据消息传递的顺序组成的状态迁移序列.

定义 21(完整消息传递序列). 在 FSM 中,完整消息传递序列是由某个状态出发最终回到自身状态的消息传递迁移序列(这里并不考虑自迁移的情况).

由上述定义,我们可以给出完整消息传递序列测试准则.

定义 22(完整消息传递覆盖准则). 给定一个测试集 TS 和系统的一个 FSM 模型 M,当且仅当对于 M 的任意一条完整消息传递序列 t_c,$\exists t \in TS$,使得 t_c 是 t 的子序列,则测试集 TS 满足完整消息传递序列覆盖准则.

完整消息传递覆盖准则主要是针对 Web 应用中功能模块间的交互行为进行测试.所得到的测试用例集通过追踪功能间的消息传递,动态反映了其各个功能模块间的调用及其完整的交互流程.完整消息传递序列是由某个状态出发最终回到自身状态(自迁移除外)的消息传递迁移序列.完整消息传递覆盖要求测试序列集覆盖每一条完整消息传递序列至少 1 次.

根据完整消息传递覆盖准则的定义可知,测试用例要满足以下序列要求:

Seq1: S0→S1→…→S1;

Seq2: S0→S1→S3…→S3;

Seq3: S0→…→S4→…→S4;

所以,据此,可以得到以下测试用例:

TC1-1: S0〈login〉S1〈submit[$n \wedge p$=FALSE]〉S4〈OK〉S1;

TC1-2: S0〈login〉S1〈cursor moved〉S3〈return〉S1;

TC1-3: S0〈login〉S1〈cursor moved〉S3〈return〉S1〈cursor moved〉S3〈return〉S1;

…

TC2-1: S0〈login〉S1〈cursor moved〉S3〈return〉S1〈cursor moved〉S3;

TC2-2: S0〈login〉S1〈cursor moved〉S3〈return〉S1〈cursor moved〉S3〈return〉S1〈cursor moved〉S3;

TC2-3: S0〈login〉S1〈cursor moved〉S3〈return〉S1〈cursor moved〉S3〈return〉S1〈submit[$n \wedge p$=FALSE]〉S4〈OK〉S1〈cursor moved〉S3;

…

TC3-1: S0〈login〉S1〈submit[$n \wedge p$=FALSE]〉S4〈OK〉S1〈submit[$n \wedge p$=FALSE]〉S4;

TC3-2: S0〈login〉S1〈cursor moved〉S3〈return〉S1〈submit[$n \wedge p$=FALSE]〉S4〈OK〉S1〈submit[$n \wedge p$=FALSE]〉S4;

TC3-3: S0〈login〉S1〈cursor moved〉S3〈return〉S1〈cursor moved〉S3〈return〉S1〈submit[$n \wedge p$=FALSE]〉S4〈OK〉S1〈cursor moved〉S3〈return〉S1〈cursor moved〉S3〈return〉S1〈submit[$n \wedge p$=FALSE]〉S4;

…

通过上述的满足完整消息传递覆盖准则的测试用例,在宏观上对整个 Web 应用交互进行了功能性的测试,可以说是一种整体功能的测试.但是,在 Web 应用中的许多错误很可能是因为页面与页面、接口与接口、前台与后台、状态与状态之间复杂的交互关系引起的,以往的状态覆盖或迁移覆盖等测试准则几乎都是孤立地测试迁移或是状态,没有充分地测试迁移及状态之间的交互关系.为了检查这些类型的错误,更好地测试各个功能模块之间的交互行为,本节提出了完整功能交互覆盖准则.

定义 23(迁移对序列). 在 FSM 中,迁移对序列指从一个状态到另一个状态由两两相邻的迁移对组成的序列.

迁移对覆盖测试准则要求测试用例集 TS 使 FSM 中的每一组迁移对的交互至少测试一次.这样可以有效地测试各个状态之间的交互行为.但是,可以发现,若将迁移对覆盖简单地应用于 Web 应用上时,往往只是一种纯粹的迁移对覆盖.而其中的某些测试序列却是无用的,或者说会产生一些无用的测试片段.也就是说,不能保证该 Web 应用系统的每个功能都得到测试.

虽然通过上述的测试用例集也可以达到迁移对全覆盖的目的,但是会产生一些没有实际意义的测试用例,这些测试用例无法检测出实际的交互错误,从而影响了实际的功能交互测试的效果和整个测试效率.因此,我们提出了完整功能交互序列

覆盖准则.

定义 24(完整功能交互覆盖准则). 给定一个测试集 TS 和系统的一个 FSM 模型 M,TS 必须保证 M 中的每个相邻的交互行为都得到测试,并且保证 M 中的每个功能都得到测试.

根据完整功能交互覆盖准则的定义可知,测试用例要满足以下序列要求:

登录成功序列:S0→S1→S3→S1→S3→S1→S2;

登录失败序列:S0→S1→S3→S1→S3→S1→S4→S1;

所以,据此,可以得到以下测试用例:

TC1: S0〈login〉S1〈cursor moved〉S3〈return〉S1〈cursor moved〉S3〈return〉S1〈submit[$n \wedge p$=TRUE]〉S2;

TC2: S0〈login〉S1〈cursor moved〉S3〈return〉S1〈cursor moved〉S3〈return〉S1〈submit[$n \wedge p$=FALSE]〉S4〈OK〉S1;

通过上述两个准则生成的测试用例,我们从宏观和微观两方面对 Web 应用的功能交互进行了测试,可以说是较为全面的完成了对 Web 应用的功能交互测试.但是,需要指出的是,在上述的测试准则中,基本没有考虑循环的测试,比如:通过 Checking,发现用户名或密码不符合相应的语法规则,用户修改后再进行 Checking,这样的一个往复过程.另一个就是用户登录失败后,重新输入用户名和密码,再重新进行登录等.

在测试实践中,状态迁移过程中的循环是很复杂的,执行循环体 N 次和简单地重复此一回路 N 次不是等价的.在将 Web 应用按功能划分后得到的 FSM 图中,一个循环体可以看作是一个子功能模块.在一个 Web 应用实例中,各个子功能模块往往是运行最频繁的,例如在一次用户购物过程中系统存在导购子功能模块和付款子模块,用户往往会因选购多个商品多次使用导购模块,而最终只使用一次付款模块.Web 应用中的很多错误往往会在多次循环交互中产生,而随着循环次数的不同,产生错误的情况也各不相同.因此,有必要对 Web 应用的循环交互进行测试.

在 Web 应用中,尤其在 Web 导航的过程中,功能构件有着各种的依赖关系或约束关系,并且是按顺序执行的.因此在考虑有序性的前提下,很难将循环体单独抽出进行测试.单独对循环体进行测试是没有意义的,会导致一些前续的错误无法被发现.以上述的用户购物过程为例,如果不先登录系统,是无法进行后续的修改个人信息、购物、付款等操作的,或者说用户不登录系统而直接进行购物付款等操作本身就是非法的.单独对循环体进行测试就会导致这种错误无法被发现,也就失去了功能交互测试的意义.Web 导航是一个有序的过程,应该将其视作是一个整体.由此,给出以下定义.

定义 25(功能循环序列). 在 FSM 中,功能循环序列是由初始状态出发并出现循环的序列.

定义 26(功能循环交互覆盖准则). 给定一个测试集 TS 和系统的一个 FSM 模型 M,当且仅当对于 M 的任意一条功能循环序列 tc,$\exists t \in TS$,使得 tc 是 t 的子序列,则测试集 TS 满足循环功能交互覆盖准则.这里规定,每个循环至少被测试 2 次.

根据功能循环交互覆盖准则的定义,存在 3 种情形:

(1)用户名或密码通过 Checking 后发现不满足相应的语法要求,反复修改,反复 Checking,最终成功登录.

(2)用户名或密码满足相应的语法规则,但经过服务器验证发现有误,用户重新输入用户名和密码反复登录系统.

(3)用户名或密码不满足相应的语法规则,并且修改后用户名或密码经验证有错,用户反复登录.

因此,根据上述情形,得到以下测试用例:

TC1: S0〈login〉S1〈cursor moved〉S3〈return〉S1〈cursor moved〉S3〈return〉S1〈cursor moved〉S3〈return〉S1〈cursor moved〉S3〈return〉S1〈submit[$n \wedge p$=TRUE]〉S2;

TC2: S0〈login〉S1〈cursor moved〉S3〈return〉S1〈cursor moved〉S3〈return〉S1〈submit[$n \wedge p$ = FALSE]〉〈OK〉S1〈cursor moved〉S3〈return〉S1〈cursor moved〉S3〈return〉S1〈submit[$n \wedge p$ = FALSE]〉〈OK〉S1;

TC3: S0〈login〉S1〈cursor moved〉S3〈return〉S1〈cursor moved〉S3〈return〉S1〈cursor moved〉S3〈return〉S1〈cursor moved〉S3〈return〉S1〈submit[$n \wedge p$=FALSE]〉S4〈OK〉S1.

…

功能循环交互测试既针对循环体进行了测试,又保证了测试功能的完整性.从本节得到的测试用例中,根据实例的要求,有些并无实际意义或者说测试用例覆盖的该条测试路径实际上是不存在的.例

如根据状态覆盖测试准则得到的测试用例 TC2：S0〈login〉S1〈submit[$n \wedge p = \text{FALSE}$]〉S4；在现实中，用户输入完用户名和密码后要先经过 Checking 后，才会到 submit. 所以由 S0 迁移到 S1 后不经过 Checking 直接迁移到 S4 是不正确的，也是无实际意义的. 该问题本课题组已另作研究，不属于本文的研究范围.

上述测试用例生成方法已集成在基于模型的 Web 应用测试系统中.

4 基于模型的 Web 应用测试系统实现

4.1 系统框架

本文研究基于模型的 Web 应用软件测试技术和工具实现. 最终构建一套从模型出发产生测试用例并自动执行测试用例的系统工具，实现 Web 应用测试过程的自动化. 如图 2 所示，Web 应用测试系统以 FSM 作为被测 Web 应用的形式测试模型，集成了模型转换器、测试目标分析器、测试序列生成器、FSM 和测试序列可视化和 Web 应用测试执行引擎等工具.

UML 是面向对象系统分析、设计的标准的建模语言，自从成为建模语言事实上的标准后，就得到学术界的推崇和工业界的支持. UML 也是 Web 应用开发的主要建模工具，开发过程中建立的各种 UML 模型提供了系统的使用、结构、行为和部署等视图，也是测试的主要信息来源. 然而，由于 UML 缺乏精确的语义，直接用 UML 模型推导测试用例时缺乏所需的语义信息，难以自动化. 系统采用 UML 状态图描述 Web 应用的行为，开发一个将 UML 状态图形式化为 FSM 的模型转换器. 同时，考虑到测试用例爆炸问题，采用 UML 顺序图描述特定的测试目标，测试目标分析器从 UML 顺序图产生针对特定测试目标的测试用例规约，然后与 Web 应用的整体 FSM 模型进行模型组合产生基于测试目标的 FSM.

本文选用了 SCXML 作为 FSM 的文本表示方式. 建立了 SCXML 元素与 FSM 模型的对应关系.

FSM 可视化工具主要实现将内部存储的 FSM 模型图形化，以便于用户直观观察和理解 FSM. 测试序列生成器按照设计好的测试覆盖准则自动地从 FSM 产生抽象的测试序列集. 除支持状态覆盖、迁移覆盖、迁移对覆盖等传统的覆盖准则外，我们还提出/改进了优化状态迁移覆盖、完整消息传递覆盖、完整功能交互覆盖和功能循环交互覆盖等覆盖准则. 测试序列可视化工具主要是实现在系统 FSM 图形上彩色显示单条测试序列路径以及彩色显示整个测试序列集中所有测试序列路径. 测试执行引擎根据被测 Web 应用测试执行配置，实例化抽象的测试序列，自动生成测试执行脚本并解释执行脚本，产生测试执行迹，从而可以动态、形象地展示出测试执行的全过程.

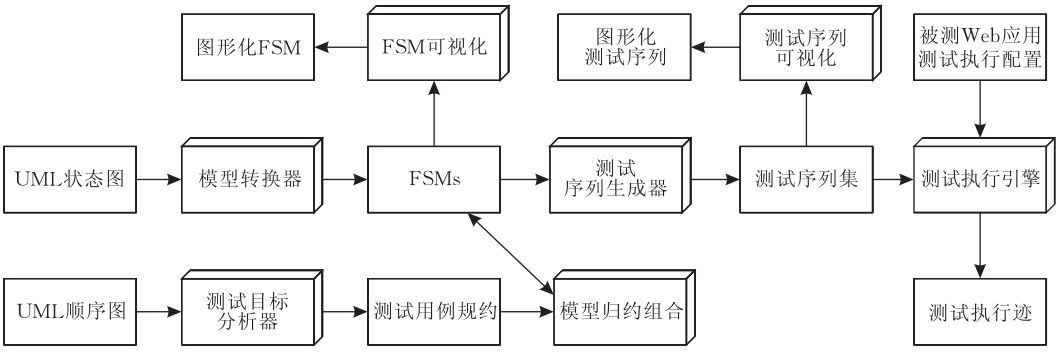

图 2 基于模型的 Web 应用测试系统实现框架

4.2 测试用例可视化

对于一个给定的 FSM 模型(SCXML 表示)，采用第 3 节给出的测试覆盖准则，系统就可以产生相应的测试用例. 本文使用开源的绘图工具 Graphviz 2.26[①] 实现了 FSM 的图形化显示和测试用例的可视化. 既可以在 FSM 展现每一条测试用例，也可以用不同颜色展示多条测试用例.

整个可视化过程可分解为若干子过程：首先读取 SCXML 格式的文本和以及根据不同测试准则得到的测试用例的文本，然后从中抽取用于图形生成的相关信息并存于自己定义好的数据结构中，最后，使用 Graphviz 来布局及映射得出图形. 图 3 给出了该系统 FSM 以及测试用例可视化实现的基本框

① Graphviz - Graph Visualization Software, http://www.graphviz.org/.

架，由各子过程划分出各功能模块.

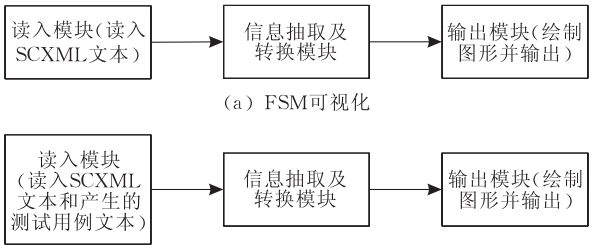

图 3　FSM 以及测试用例可视化工具的架构图

利用 Graphviz 提供的 API，将数据结构中的模型信息写入 Graphviz 能够识别的 dot 文件，再由程序代码自动开启 Graphviz 的执行库，以新生成的 dot 文件为输入，得到其相应的图形类数据流，从而获得一个与原文本模型一致的图片文件，并将其交给输入输出模块，最终呈现给用户. 测试用例的可视化与此类似，只是在显示的每一条测试用例序号上增加了一个单击事件，当触发该事件后，系统根据面板上显示的 FSM 的文本 SCXML 以及该测试用例的文本进行匹配，匹配后生成 dot 文件，并且在该 dot 文件中相应的位置处给出颜色标记，这样在调用 Graphviz 时，就显示了 FSM 图形，并且该图形中对应的测试用例部分采用不同颜色进行标识，如图 4 所示. 同理，测试用例集的显示与此类似，不再详述.

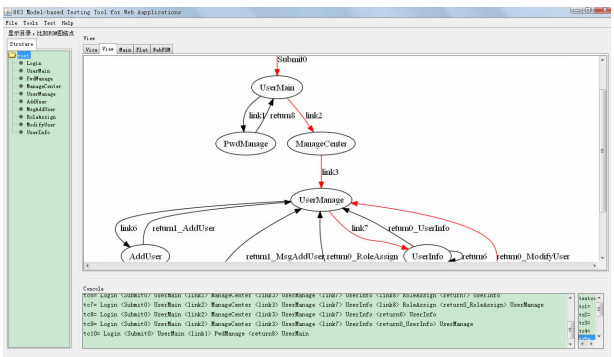

图 4　一个测试用例在测试模型上的显示

4.3　测试执行

4.3.1　测试执行引擎

测试执行引擎是整个自动化测试工具的重要部分，由测试脚本和输入数据所驱动. 执行引擎模拟多用户的操作行为对被测 Web 应用进行功能测试. 本节给出 Web 应用测试自动化工具中测试执行引擎的构建及实现.

测试引擎根据 Web 应用三层结构所设计，分别对应了客户端、服务器端及数据库端.

该引擎使用多线程技术，实例化一组客户端来模拟多用户同时对 Web 应用进行操作. 客户端向 Web 应用发出请求，对被测系统进行两种方式的检查：一是对 Web 服务器端的响应进行检查；二是对数据库端进行数据检查. 并且在任一个客户端中可定义多个检查项目. 测试执行引擎与测试工具中各组件按照一定顺序分工合作. 测试脚本经过验证通过后，测试执行引擎根据脚本每个用例中的变量定义从外部数据文件中获取测试数据，然后实例化为多个具体测试用例对 Web 应用进行并发的请求操作，通过检查响应中检查点中内容来判断结果的正确性，最后利用执行引擎中记录的测试日志来生成测试报告.

4.3.2　测试脚本

测试脚本一般指可被自动化测试工具执行的针对一个特定测试的一系列指令，由测试执行引擎解析和执行，因此是自动化测试实现中的关键部分.

测试脚本可使用测试自动化工具自动生成，或由手工创建(记录)，或用程序语言编程来完成，也可综合前三种方法来完成. 脚本语言的设计应达到下面几个要求：易读性、易维护性、复用性、高效率和健壮性. 良好的测试脚本将为测试者提供灵活的使用空间，增加测试工具的测试准确度，减少测试人员的工作，同时也可以降低对测试人员的综合素质要求.

到目前为止，Web 的功能测试在国内还处于探索阶段，还没有专门可用的工具，即使国外的主流测试工具，如 Winrunner，Rational Robot，Quick Test 等，其测试脚本语言都存在很大的局限性，脚本复杂，难于掌握，对测试人员水平要求很高.

本自动化测试工具采用基于 XML 的测试脚本，测试脚本由工具自动生成，结构清晰易读，测试人员可轻松编辑及维护. 该脚本拥有自己的语法规则和保留字，并采用数据驱动方式将测试输入存储在独立的数据文件中而不是绑定在脚本中，从而提高了脚本复用率，降低了测试开销. 脚本定义测试用例、测试步、测试输入和检查点即预期输出等信息，并通过检查点来判断结果是否正确. 为了使基于 XML 格式的文档拥有可视化效果，这里使用了可扩展样式表语言 XSL(Extensible Stylesheet Language). XSL 可以将 XML 转换成 HTML，可以过滤和分类 XML 数据，可以对一个 XML 文档的部分进行寻址，可以基于数据值格式化 XML 数据，可以向不同设备输出 XML 数据.

执行引擎通过向被测 Web 应用发送测试脚本中定义好的 HTTP 请求来模拟多用户访问 Web 应

用的场景,并通过分析 HTTP 响应数据或者探测数据库中数据来检测系统的功能.本测试工具通过实例化抽象测试序列来自动生成或人工手动建立方式来生成脚本.脚本基于 XML 进行开发,利用 XML 的嵌套结构可方便定义复杂的测试脚本.

```
〈testsuite〉
    〈environment/〉
    〈testcase〉
        〈definition/〉
        〈teststep〉
            〈request/〉
            〈response/〉
        〈/teststep〉
        …
    〈/testcase〉
    …
〈/testsuite〉
```

图 5 测试脚本的总体结构

测试脚本主要定义测试用例及其输入和预期输出,由测试包(TestSuite)、测试用例(TestCase)、测试步(TestStep)三层结构嵌套而成.如图 5 所示.

(1)测试包.每个测试脚本包含一个测试包,一个测试包包含一个或多个测试用例,测试包为用来确认被测系统逻辑错误的测试用例的集合.

(2)测试用例.一个测试用例包含一个或多个测试步,是完成一个明确的测试目标的测试,即具体条件下对 Web 应用某个场景的具体行为.

(3)测试步:与 Web 应用的一次交互行为,即一次 HTTP 请求定义和一次预期 HTTP 响应定义.

4.3.3 测试执行引擎的实现

在满足所有测试所需条件后,测试执行引擎协调测试脚本、测试数据及被测 Web 应用 3 个对象,动态地调用脚本解析器和测试数据来实例化每个测试场景,通过判断响应页面中检查点来对 Web 应用进行测试.脚本可根据测试人员对抽象序列的描述,通过脚本生成器自动获得;脚本解析器在执行过程中需多次被使用,因此以类的形式封装了所有所需方法作为一个辅助类;测试执行使用多线程技术并行地来对不同的测试场景进行测试,并记录测试日志.

4.3.4 测试执行

在上述测试脚本的解释类的辅助下,测试执行引擎按一定顺序解析测试脚本,按照脚本中测试场景(即脚本中 testcase 节点)的个数实例化一个或多个线程,即模拟多用户来对具体 Web 应用进行并行测试.由于该测试执行引擎基于数据驱动方式,可以减少需要执行所有测试数据所需要的测试脚本量,即可以使用很少的脚本来产生大量的测试用例.例如,在一个测试场景中有 5 个顺序执行的测试步,其中两个测试步用到了外部数据文件作为测试输入,假设测试步 2 用到了 3 组测试数据;测试步 4 用到了 4 组数据,则最终该测试执行引擎会生成 12 个具体测试用例来覆盖所有的可能的测试数据组合.执行引擎会组合所有的可能,把不同的测试数据引入同一个场景来产生多种测试用例,通过 HTTPUnit 测试工具来对指定的 Web 应用进行测试.以下为测试执行的核心伪代码:

```
//对于每个测试场景实例化以下线程进行执行
private void run(){
  根据具体测试数据对该测试场景生成所有可能的测试用例
  foreach 测试用例 {
    foreach 测试步〈teststep〉{
    根据辅助类 ParseXML 中定义的方法来提取测试所需数据;
    提取测试步〈teststep〉标记中各属性值 name、loop 等;
    提取HTTP 请求〈request〉标记中属性值 url、method 及
        所需提交参数 parameter;
    提取预期 HTTP 相应〈response〉标记中 statuscode 及所
        需检查的内容 check;
    使用HTTPUnit 测试工具对 Web 应用发出请求并获得
        响应;
    调用辅助类 WebAnalyse 中对 Web 应用或数据库检查的
        6 中方法来检查响应;
    输出 Web 测试日志;
    设置测试输出关键信息 reportDetails 用于测试报告;
    }
  }
}
```

5 工具演示

下面对基于模型的 Web 应用测试系统的运行进行简单描述.系统采用 ArgoUML 对 Web 应用建模,图 6 是兴宁水库移民信息管理系统应用实例的状态图.

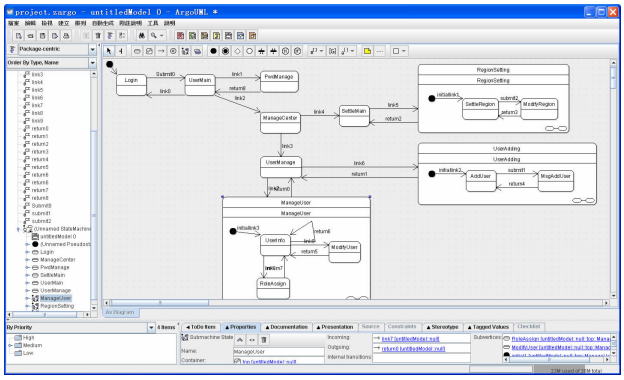

图 6 应用实例的状态图

通过 ArgoUML 将上述的描述 Web 应用导航的 UML 状态图保存为 XMI 格式文件,启动本文设计并开发的基于模型的 Web 应用测试系统工具

(Model-based Testing Tool for Web Applications)，加载通过 ArgoUML 得到的 XMI 格式文件，点击 tools 菜单下的“Transform XMI2FSM”，就完成了 UML 状态图到形式化的 FSM 模型的转换，点击 tools 菜单下的“Visualization”，就可以看到图形化的 FSM，如图 7 所示，其中最上层图中左边有 4 个节点，表示 4 个复合状态.

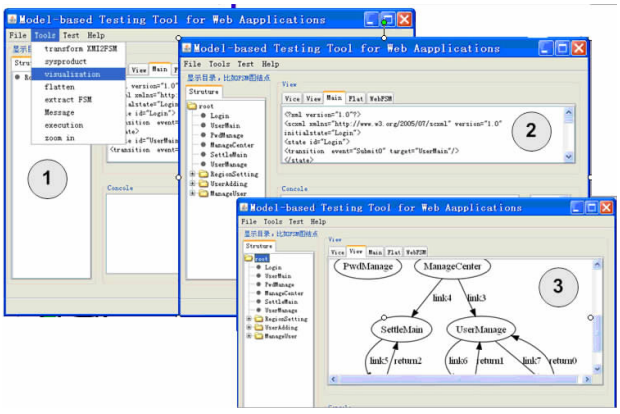

图 7　模型转换与 FSM 的可视化

根据 FSM 的定义(见定义 1)，其状态均为基本状态或简单状态，也就是说，无嵌套状态或复合状态. 所以，对上述得到的采用 SCXML 描述的带复合状态的 FSM 进行展平(去层次化)，点击 tools 菜单下的“flatten”，这样就得到了展平后的 SCXML 格式的 FSM 模型，点击 tools 菜单下的“Visualization”，就可以对展平后的 FSM 进行可视化，如图 8 所示. 另外，模型规约与此类似.

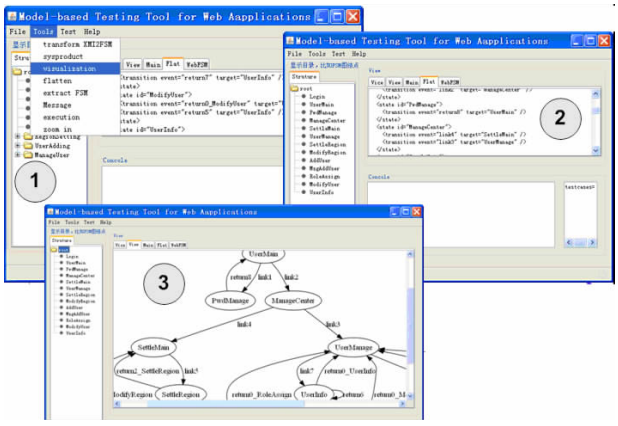

图 8　FSM 的去层次化

选择不同的测试准则可自动产生抽象的测试用例，如图 9 所示.

FSM 可视化和单条测试用例覆盖的测试路径可用不同的颜色自动显示，如图 10 所示.

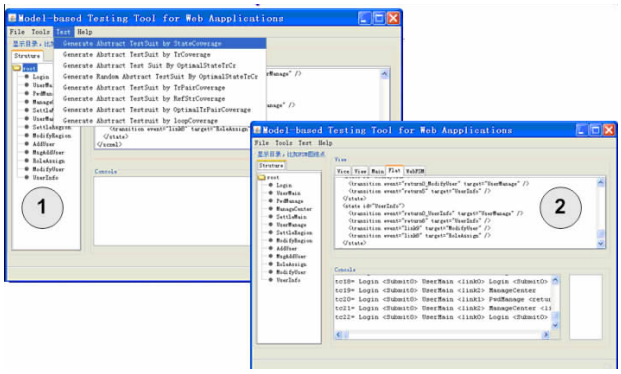

图 9　选择测试准则自动产生测试用例

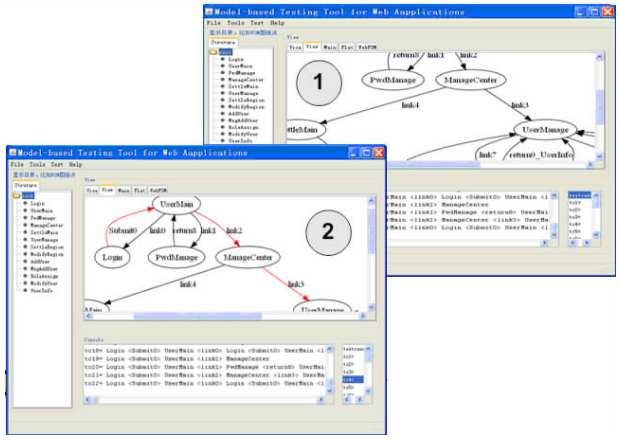

图 10　FSM 和测试路径的可视化

在产生抽象的测试用例后，实例化测试用例并产生测试脚本，测试引擎解释执行脚本，产生测试报告，如图 11 所示.

6　结束语

Web 应用是一种典型的应用程序. Web 应用本身越来越复杂，同时它所使用的开发语言和开发模型在不断发展. 所有这些因素都给分析、建模和测试带来了很大的难度. 目前的测试主要依赖测试工程师的直觉和经验. Web 应用测试被认为是一个耗时、代价昂贵的过程. 因此，迫切需要一套系统的 Web 应用测试方法并能实现测试过程的自动化. 本文正是基于以上目的，设计并实现了一个基于模型的 Web 应用测试系统，以 Web 应用的 UML 状态图作为系统测试模型，采用 UML 顺序图描述测试目标，通过转换和组合，构成 FSM 测试模型. 实现了测试生成的自动化、测试模型以及测试用例的可视化、测试执行的自动化等.

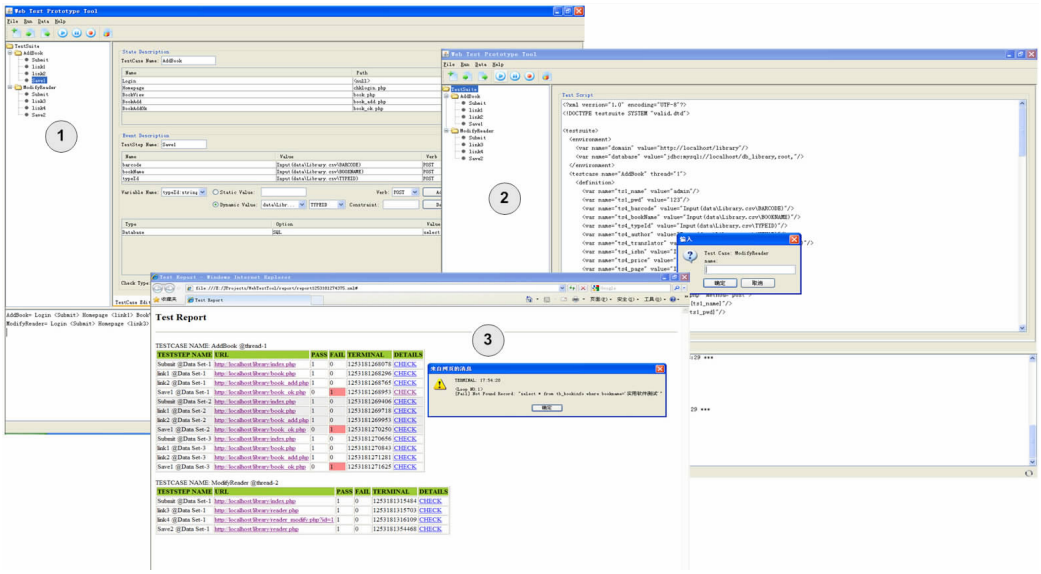

图 11 测试执行

参 考 文 献

[1] Pretschner Alexander, Lotzbeyer Heiko, Philipps Jan. Model based testing in evolutionary software development//Proceedings of the 12th IEEE International Workshop on Rapid System Prototyping. IEEE Computer Society, Washington, DC, USA, 2001: 155-161

[2] Tonella Paolo, Ricca Filippo. Statistical testing of Web applications. Journal of Software Maintenance and Evolution: Research and Practice, 2004, 16(2): 103-127

[3] Utting Mark, Pretschner Alexander, Legeard Bruno. A taxonomy of model-based testing approaches. Software Testing, Verification and Reliability, Wiley Online Library, 2011: 1-16

[4] Neto Arilo C. Dias, Subramanyan Rajesh, et al. A survey on model-based testing approaches: A systematic review//Proceedings of the 1st ACM international Workshop on Empirical Assessment of Software Engineering Languages and Technologies: Held in Conjunction with the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE'07). ACM, New York, USA, 2007: 31-36

[5] Tang Yun-Ji, Miao Huai-Kou, Qian Zhong-Sheng. Approach to modeling and testing Web applications based on functional components. Computer Science, 2009, 36(7): 124-127, 169 (in Chinese)
(唐云吉, 缪淮扣, 钱忠胜. 一种基于功能构件的 Web 应用建模与测试方法. 计算机科学, 2009, 36(7): 124-127, 169)

[6] 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(12): 1172-1186

[7] Hieatt Edward, Mee Robert. Going faster: Testing the Web application. IEEE Software, 2002, 19(2): 60-65

[8] Xu Lei, Xu Bao-Wen. Research on testing framework for Web applications. Journal of Southeast University (Natural Science Edition), 2004, 34(6): 751-755(in Chinese)
(许蕾, 徐宝文. Web 应用测试框架研究. 东南大学学报(自然科学版), 2004, 34(6): 751-755)

[9] Zhou Xiao-Yu, Xu Lei, Xu Bao-Wen, Chen Huo-Wang. Automatically testing for the Web application. Computer Science, 2005, 32(1): 125-127(in Chinese)
(周晓宇, 许蕾, 徐宝文, 陈火旺. Web 应用的自动测试. 计算机科学, 2005, 32(1): 125-127)

[10] Chen Sheng-Bo. Modeling and verifying Web applications [Ph. D. dissertation]. Shanghai University, Shanghai, 2008 (in Chinese)
(陈圣波. Web 应用建模和验证方法研究[博士学位论文]. 上海大学, 上海, 2008)

[11] Liu Huan-Zhou, Miao Huai-Kou. Approach to modeling and generating test case for Web application. Computer Engineering, 2008, 34(6): 60-62(in Chinese)
(刘焕洲, 缪淮扣. Web 应用程序建模和测试用例生成方法. 计算机工程, 2008, 34(6): 60-62)

[12] Qian Zhong-Sheng, Miao Huai-Kou, Chen Sheng-Bo. Modeling and testing Web applications based on ORD and FSM. Computer Science, 2008, 35(9): 278-281, 291(in Chinese)
(钱忠胜, 缪淮扣, 陈圣波. 基于 ORD 和 FSM 的 Web 应用的建模与测试. 计算机科学, 2008, 35(9): 278-281, 291)

[13] Hu Rong, Miao Huai-Kou, Liu Huan-Zhou. An approach to modeling Web applications integration testing. Computer Science, 2007, 34(6): 253-257(in Chinese)
(胡蓉, 缪淮扣, 刘焕洲. 一种基于 Web 软件集成测试的建模方法. 计算机科学, 2007, 34(6): 253-257)

[14] Isakowitz Tomas, Stohr Edward A, Balasubramanian P. RMM: A methodology for structured hypermedia design. Communication of the ACM, 1995, 38(8): 34-44

[15] Coda Francesco, Ghezzi Carlo, Vigna Giovanni, Garzotto

Franca. Towards a software engineering approach to Web site development//Proceedings of the 9th International Workshop on Software Specification and Design(IWSSD'98). Ise-Shima, Japan, 1998: 8-17

[16] Gellersen Hans-W, Gaedke Martin. Object-oriented Web application development. IEEE Internet Computing, 1999, 3(1): 60-68

[17] Conallen Jim. Modeling Web application architectures with UML. Communications of the ACM, 1999, 42(10): 63-70

[18] Turine M A S, Oliveira M C F, Masiero P C. A navigation-oriented hypertext model based on state-charts//Proceedings of the 8th ACM Conference on Hypertext (Hypertext'97). Southamption, UK, 1997: 102-111

[19] Leung Karl R P H, Hui Lucas Chi Kwong, Yiu S M, Tang Ricky W M. Modeling Web navigation by statechart//Proceedings of the 24th International Computer Software and Applications Conference (COMPSAC'00). IEEE Computer Society, Washington, DC, USA, 2000: 41-47

[20] Oliveira M C F, Turine M A S, Masiero P C. A statechart-based model for hypermedia applications. ACM Transactions on Information Systems, 2001, 19(1): 28-52

[21] Kung David C, Liu Chien-Hung, Hsia Pei. An object-oriented Web test model for testing Web applications//Proceedings of the 24th Annual International Computer Software and Applications Conference (COMPSAC'00). Taipei, Taiwan, 2000: 537-542

[22] Liu Chien-Hung, Kung David C, Hsia Pei, Hsu Chih-Tung. Object-based data flow testing of Web applications//Proceedings of the 1st Asia-Pacific Conference on Quality Software (APAQS'00). Hong Kong, China, 2000: 7-14

[23] Andrews Anneliese, Offutt Jeff, Alexander Roger. Testing Web applications by modeling with FSMs. Software Systems and Modeling, 2005, 4(3): 326-345

[24] Ricca Filippo, Tonella Paolo. Analysis and testing of Web applications//Proceedings of the 23rd International Conference on Software Engineering (ICSE'01). Toronto, Ontario, Canada, 2001: 25-34

[25] Binder Robert V. Testing object-Oriented Systems: Models, Patterns, and Tools. Addison-Wesley, 1999

[26] Ipate Florentin. Bounded sequence testing from deterministic finite state machines. Theoretical Computer Science, 2010, 411(16-18): 1770-1784

[27] Chow T S. Testing software design modeled by finite-state machines. IEEE Transactions on Software Engineering, 1978, 4(3): 178-187

[28] Fujiwara Susumu, Bochmann Gregor von, Khendek Ferhat, Amalou Mokhtar, Ghedamsi Abderrazak. Test selection based on finite state models. IEEE Transactions on Software Engineering, 1991, 17(6): 591-603

[29] Offutt Jeff, Liu Shao-Ying, Abdurazik Aynur, Ammann Paul. Generating test data from state-based specifications. The Journal of Software Testing, Verification, and Reliability, 2003, 13(1): 25-53

[30] Wagner Ferdinand, Schmuki Ruedi, Wagner Thomas, Wolstenholme Peter. Modeling Software with Finite State Machines: A Practical Approach. Auerbach Publications, 2006

[31] Miao Huai-Kou, Liu Ling. A test class framework for generating test class from Z specifications//Proceedings of the 6th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'00). Tokyo, Japan, 2000: 164-171

[32] Offutt Jeff, Liu Shao-Ying. Generating test data from SOFL specifications. Journal of systems and software, 1999, 49(1): 49-62

[33] Offutt Jeff, Liu Shao-Ying, Abdurazik Aynur, Ammann Paul. Generating test data from state-based specifications. The Journal of Software Testing, Verification, and Reliability, 2003, 13(1): 25-53

[34] Amyot Daniel, Roy Jean-francois, Weiss Michael. UCM-driven testing of Web applications//Proceedings of the 12th International SDL Forum. Springer, Grimstad, Norway, 2005: 247-264

[35] Dick Jeremy, Faivre Alain. Automating the generation and sequencing of test cases from model-based specifications//Jim Woodcock, Peter Gorm Larsen eds. Proceedings of the 1st International Symposium of Formal Methods Europe on Industrial-Strength Formal Methods (FME'93). Springer-Verlag, London, UK, 1993: 268-284

[36] Stocks Phil, Carrington David. A framework for specification-based testing. IEEE Transactions on Software Engineering, 1996, 22(11): 777-793

[37] Hierons Robert. Testing from a Z specification. The Journal of Software Testing, Verification, and Reliability, 1997, 7(1): 19-33

MIAO Huai-Kou, born in 1953, professor, Ph. D. supervisor. His current research interests include formal methods and software engineering.

CHEN Sheng-Bo, born in 1975, Ph. D., lecturer. His current research interests include formal methods and software testing.

ZENG Hong-Wei, born in 1966, Ph. D., professor. His current research interests include formal methods and software testing.

Background

Testing is a most fundamental approach to improving the reliability and the quality-assurance of software. A key problem of software testing is how to select the effective test cases from infinite input state space of program to satisfy the requirements of testing. As a kind of typical Web software, Web applications possess some unique features such as multi-constituents, heterogeneity, easy to dynamic configuration and composition. Model-based testing provides an effective solution to implement the automation of functional testing for Web applications.

In this paper, a model-based testing approach for Web applications is proposed, which involves in Web modeling, test generation, test execution and visualization for Web models and test sequences.

Innovation:

1. presents an approach to transformation from UML model, describing user requirements, to FSM;

2. presents an approach to testing model reduction based on test purposes represented in UML sequence diagram;

3. proposes or improves some coverage criteria such as Optimized State and Transition Coverage, Complete Message Pass Coverage, Complete Function Interaction Coverage and Function Loop Interaction Coverage. These criteria are implemented in the proposed test tools.

4. provides a method for the visualization of FSM and test cases such that a FSM model is displayed in a graph, and test cases are highlighted in various colors.

5. presents a test execution method involved design and generation of test script, design and implementation of test execution engine.

This work was supported by the National Natural Science Foundation of China under grant No. 60673115, and the National High Technology Research and Development Program (863 Program) of China under grant No. 2007AA01Z144. The tools for model-based Web application testing has been confirmed by 863 expert group appointed by the Ministry of Science and Technology of China, and are used on trial in Shanghai Development Center of Computer Software.