基于情景演算的用户界面模型验证方法

梁伟晟　李　磊
（中山大学软件研究所　广州 510275）

摘　要　用户界面设计是业务应用系统设计的重要组成部分，验证界面设计正确性与设计用户界面同样重要。界面关系的非直观性和动态性使得界面设计的验证并不容易。以初始界面开始得到最终界面的过程可以看作是规划问题。为此，我们引入情景演算来解决界面设计的验证问题。我们在经典情景演算中引入界面检验机制，用带检验机制的情景演算来表达界面的规划，在规划过程中实现界面设计的验证，最终通过规划结果判断界面设计的正确性。

关键词　用户界面，规划，情景演算，验证

A Situation Calculus-based Validation Approach to User Interface Model

LIANG Wei-Sheng　LI Lei
(Software Institute, Zhongshan University, Guangzhou 510275)

Abstract　User interface design is important part of business system design, so as the validation of user interface design. The non-visual and dynamic relation between interfaces makes validation difficulty. It is a planning problem to get goal interface from initial interface. Then we introduce situation calculus to solve the validation of user interface design. We introduce interface checking in classical situation calculus, apply situation calculus with interface checking to describe the user interface planning. The validation of user interface design is done during the planning and can be judged by the result of planning.

Keywords　User interface, Planning, Situation calculus, Validation

1　研究动机

用户界面设计是业务应用系统设计的重要组成部分。用户界面设计的方法和工具主要分为基于语言的设计工具、交互式图形描述工具和基于模型的生成工具。研究统计表明[1]，平均48%的系统设计开发工作量在于用户界面部分。验证界面设计正确性与设计用户界面同样重要。为了验证用户界面的正确性，国内外的研究提出了许多界面测试方法。L. White 提出了 Complete Interaction Sequences（CIS）方法[2]，使用 CIS 脚本转换器来生成测试用例。基于界面对象建模的测试[3]从用户/应用系统交互和应用系统/系统环境交互的视角，提出了一种界面类对象状态测试建模技术，支持面向对象应用软件的测试。基于界面构件关联图的测试技术[4]描述界面构件之间联结和制约关系，提出了软件测试覆盖准则和测试用例生成方法。基于扩展有限状态机测试的测试数据自动选取方法[5]方便了测试人员从中手工选择测试数据。由于界面输入和界面交互的复杂性，各种测试方法的测试用例覆盖程度往往不够理想，而且测试用例未必体现系统运行的实际业务流程。

A. Memon[6,7]研究了界面测试问题，以 Microsoft WordPad 为例子，引入自动规划方法产生界面的测试用例。该方法主要思想是将测试目标作为输入，通过规划技术产生达到测试目标的动作序列，即测试用例，从而一定程度上解决了覆盖性问题。然而该方法只是侧重于讨论界面测试用例的产生，没有考虑依据业务需求验证界面设计的问题。首先，界面测试用例只适用于已开发完成系统的测试，测试结果无法判定错误源自设计阶段还是开发阶段；其次，应该有相应机制在设计阶段检验界面是否正确，以免把错误带到开发阶段；此外，为体现实际的业务流程，界面设计的检验标准应该是用户需求。软件工程的实践表明，界面设计来源于业务需求，脱离需求讨论界面的设计是不合理的。即使界面设计已经完成，由于需求的变化引起界面设计的修改，同样需要验证修改后的界面设计的正确性。因此验证界面设计是否正确即是检验界面设计是否满足业务需求。

目前界面设计工具普遍采用“所见即所得”的设计方法，检验界面设计是否符合用户需求似乎很简单。然而一个令人忽略的事实是界面设计的内容不仅仅是直观的界面布局，还包含了非直观的界面关系。界面关系表现为业务操作引起的界面之间调用和转换关系。通过界面之间的转换，界面关系隐含了业务处理流程。这种非直观的界面关系是不能简单地被检验是否正确的。同时，界面转换的动态特性也使得“所见即所得”方法难以对大量的界面转换逐一检验。

在业务系统中，业务处理流程是从某个界面开始，经过一系列操作步骤以及相应的界面转换，最终在某个界面结束。界面描述了系统运行过程中的情景。从这个意义上讲，以初始界面开始得到最终界面的过程实际上是规划问题。因而，我们自然地引入情景演算理论来验证界面设计的正确性。情景演算[8]（Situation Calculus）是描述状态变化和动作推理的形式化方法。情形演算假定所有变化都是由动作产生的，在情景 s 下执行动作 a 将到达一个新的情景 s'。但是经典的情景演算理论并没有讨论规划是否正确以及如何使得规划正确的问题。规划的正确性与具体应用相关，实际的规划应用需

梁伟晟　博士研究生，主要从事数据库与知识库研究；李　磊　教授，博士生导师，主要从事数据库与知识库、逻辑程序设计研究。

要正确性检验机制来保证规划是满足需求的。对于本文讨论的界面设计验证，我们采用基于模型的用户界面设计方法对验证的对象(界面)建模，界面模型从业务需求获取，然后在经典情景演算中引入检验机制，用带检验机制的情景演算来表达界面的规划问题。在 s_0 到 s_n 的规划过程中，我们逐步对情景和动作进行检验，从规划的结果判断界面模型设计的正确性。在文章最后，我们证明了界面规划过程可以验证界面模型的正确性。

2 界面模型

采用基于模型的用户界面设计方法，文[9]介绍了从企业表单出发进行需求分析和获取，建立需求模型，从需求模型推导界面模型的方法。界面模型 I 包括界面表示模型 P、界面关系模型 R 和界面权限模型 A。界面表示模型描述界面的布局以及界面的构成元素，界面关系模型描述界面之间的关系，界面权限模型描述系统中各个角色的权限。

定义 2.1 界面表示模型 $P=\{id,LT,E\}$。id 是界面标识。将界面分割为不同的区域，LT 是表示界面布局的二叉树，树的每个节点 $Node=\{Areaid,lc,rc,VH,location\}$，$VH$ 表示节点孩子(区域)lc 和 rc 是垂直相邻 V 还是水平相邻 H，$location$ 表示节点孩子在何处相邻。E 是界面元素集合，界面元素包含属性$\{eid,type,location,size,area\}$，分别表示界面元素标识、类型、在所属区域内的位置、大小和所属区域。

定义 2.2 界面关系模型 $R=\{s_0,S,TS,F\}$，其中 s_0 是初始界面，F 是终止界面集，S 是界面集合，$TS:S\rightarrow S$ 是转换集合，表示业务操作引起界面之间调用和跳转。

定义 2.3 界面权限模型 $A=\{(r,ts)\mid r\in R,ts\in TS,R$ 是系统角色的集合，TS 是界面关系模型中转换的集合$\}$。权限模型是允许角色操作哪些界面转换的集合。

界面表示模型和权限模型体现了业务需求中对界面的非功能性需求，而界面关系模型体现了对界面的功能性需求。

3 带检验机制的情景演算

针对界面模型，我们定义几个相关的域：Situation 表示界面(情景)的域，Action 表示引起界面转换的操作(动作)的域，Layout 表示界面布局区域的域，Element 表示界面元素的域，Role 表示系统角色的域。用 s,a,l,e,r 分别表示各域中的变量。根据经典情景演算[10]，我们有：

- s_0 表示初始情景
- 函数 do：$Action\times Situation\rightarrow Situation$
- 谓词$\leqslant$：$Situation\times Situation$，$s\leqslant s'$表示 s'是 s 的后继情景
- 谓词 $Poss$：$Action\times Situation$，表示 s 情景下可以执行 a
- $(\forall P)P(s_0)\wedge(\forall a,s)[P(s)\supset P(\mathrm{do}(a,s))]\supset(\forall s)P(s)$，$P$ 是谓词 (公理 3.1)

定义 3.1 V_s 是与情景相关的谓词：$Layout\times Element\times Situation\rightarrow Boolean$。

V_s 定义了界面表示模型中对象之间的关系。

定义 3.2 V_r 是与情景相关的谓词：$Role\times Action\times Situation\rightarrow Boolean$。

定义 3.2 表示：如果在情景 s 下，角色 r 可以执行动作 a，那么 V_r 为真。V_r 定义了界面权限模型中对象之间的关系。

定义 3.3 V_a 是与情景相关的谓词：$Action\times Situation\rightarrow Boolean$。

定义 3.3 表示：如果在情景 s 下可以执行动作 a，那么 V_a 为真。V_a 定义了界面关系模型中对象之间的关系。由定义 3.2 和 3.3，显然可以得到下面的推论 3.1。

推论 3.1 如果 V_r 为真，那么 V_a 为真，反之不成立。

定义 3.4 X 是用户界面需求相关的领域公理集合：$x\in X$，x 是谓词，x 的谓词符号是 V_s、V_r 和 V_a 之一。

情景 s 的正确性：X 是领域公理集合，用 $l(s)$，$e(s)$分别表示 s 的界面区域集和界面元素集。如果 $\forall l\in l(s)$，$\forall e\in e(s)$，$V_s(l,e,s)$与 X 是一致的，即 $V_s(l,e,s)\wedge X$ 是真的，那么情景 s 是正确的，否则 s 是无效的。情景 s 的正确性记为 $Valid\_S(s)\equiv\bigwedge_{\forall l\in l(s),\forall e\in e(s)}V_s(l,e,s)$。当 s 是正确时，$Valid_S(s)=$True，否则 $Valid_S(s)=$False。$Valid_S(s)$检验界面表示模型的正确性。

动作 a 的正确性：s 是当前情景，r 是角色，X 是领域公理集合。如果 $V_r(r,a,s)$和 $V_a(a,s)$与 X 都是一致的，即 $V_r(r,a,s)\wedge X$ 和 $V_a(a,s)\wedge X$ 都是真的，并且对于 $s'=do(a,s)$，有 $Valid_S(s')$是真的，那么动作 a 在情景 s 下对于角色 r 是正确的，否则是无效的。由于动作 a 的执行总是基于当前情景，我们讨论动作的正确性时不再特别说明是基于哪个情景，默认为当前情景。动作 a 对于 r 的正确性记为 $Valid\_A(a,r)\equiv V_r(r,a,s)\wedge V_a(a,s)\wedge Valid\_S(do(a,s))$。根据推论 3.1，$Valid\_A(a,r)=V_r(r,a,s)\wedge Valid\_S(do(a,s))$。

用 $r(a)$表示允许执行动作 a 的角色集合。如果 $\forall r\in r(a)$，$Valid_A(a,r)=$True，那么动作 a 是正确的，否则是无效的。忽略 $Valid\_A$ 的参数 r，动作 a 的正确性记为

$$Valid\_A(a)\equiv\bigwedge_{\forall r\in r(a)}V_r(r,a,s)\wedge Valid\_S(do(a,s))$$

$Valid_A(a)$检验界面关系模型和权限模型的正确性。

推论 3.2 对于定义 2.1、2.2 和 2.3 的界面模型，假设 $S=\{s_0,\cdots,s_n\}$是所有的情景，$A=\{a_0,\cdots,a_m\}$是所有的动作。如果 $\forall s\in S$，$Valid_S(s)=$True，并且 $\forall a\in A$，$Valid_A(a)=$True，那么界面模型是正确的。

定义 3.5 Φ 是零情景：Φ 是情景，$Valid_S(\Phi)\equiv$False。

零情景的例子是空白界面，空白界面在任何情况下都是无效的。另一方面，显然不存在某一界面，在任何情况下都是正确的，即$\neg\exists s$，$Valid_S(s)\equiv$True。

引入检验机制的情景演算的谓词 $Poss_V$ 和函数 do_V 定义如下：

$Poss_V\subseteq Action\times Situation$

$Poss_V(a,s)=Poss(a,s)\wedge Valid\_S(s)$

do_V：$Action\times Situation\rightarrow Situation$

$$do_v(a,s)=\begin{cases}do(a,s) & \text{如果 } Valid\_A(a)\text{为真}\\ \Phi & \text{如果 } Valid\_A(a)\text{为假}\end{cases}$$

前提条件公理：$Poss_V(a,s)$

结果公理：$Poss_V(a,s)\supset(Valid\_A(a)\supset Valid\_S(do_V(a,s)))$

后继状态公理：$Poss_V(a,s)\supset Valid\_S(do_V(a,s))\vee(\neg Valid\_A(a)\wedge do_V(a,s)=\Phi)$

4 验证过程和效果

X 是领域公理，s_0 是初始界面，s_n 是目标界面，前面讨论的基于情景演算的界面规划问题表示为 $P=(X,Situation,Action,Layout,Element,Role,s_0,s_n)$。

结论：I 是定义 2.1、2.2 和 2.3 的界面模型，$P=(X, Situation, Action, Layout, Element, Role, s_0, s_n)$的规划过程可以验证界面模型 I 的正确性。

证明：不失一般性，假设 s_0 是任意的某个初始界面，s_n 是目标界面，并且在界面关系模型 R 中存在 ts 转换序列，使得 $s_0 \xrightarrow{ts} s_n$。

对于规划问题 $P=(X, Situation, Action, Layout, Element, Role, s_0, s_n)$，

1）如果不存在 s_0 到 s_n 的规划，根据 $Poss_V$ 和 do_V 的定义，以及动作和界面转换之间的对应关系，有 $Valid\_S(s_0)=$ False 或者 s_0 到 s_n 的规划终止于零情景 Φ。于是可以验证界面模型 I 是无效的。

2）如果存在 $a=\langle a_0,\cdots,a_{n-1}\rangle$，使得 a 是 s_0 到达 s_n 的动作序列。即

$$s_0 \xrightarrow{a} s_n \equiv (\exists a_0,\cdots,a_{n-1})\ s_n = do_V(\langle a_0,\cdots,a_{n-1}\rangle, s_0) \wedge s_0 \leqslant s_n$$

根据 $Poss_V$ 和 do_V 的定义，$Valid_S(s_0)=$True，$Valid_S(s_n)=$True，并且 $\forall a \in \langle a_0,\cdots,a_{n-1}\rangle$，$Valid_A(a)=$True。根据公理 3.1，对于 s_0 到 s_n 规划过程中的每一个 s，也有 $Valid_S(s)=$True。

考虑 s_0 的任意性，可以验证界面模型 I 是正确的。

综合 1）和 2），得到 P 的规划可以验证界面模型 I 的正确性。

主要工作总结 我们通过引入问题相关的检验机制，基于情景演算的规划，为用户界面设计的验证提供了方法。我们的主要工作是：首先，对界面建模，建模的依据是业务需求，从而使得验证过程能够体现实际的业务流程；其次，提前在设计阶段检验界面是否正确，而不采用为应用系统生成测试用例的方式来验证界面，避免把错误带到开发阶段；第三，利用情景演算处理复杂状态变化的优势以及自动推理和计算能力，将验证转化为在规划过程中自动完成，解决了需要逐一对各种界面状态进行正确性判断的问题。但是，我们可以看到，前面讨论的检验机制在情景正确性和动作正确性判断过程中，对同一情景的正确性判断处理了两次，因此，更进一步的研究是简化验证过程，提高规划的效率，从而提高界面设计验证的效率。

参 考 文 献

1 Schlungbaum E. Model-based User Interface Software Tools: Current state of declarative models. GIT-GVU-96-30, Graphics, Visualization & Usability Center, Georgia Institute of Technology, Atlanta, November 1996

2 White L, Almezen H. Generating test cases for GUI responsibilities using complete interaction sequences. In: Proceedings of the 11th International Symposium on Software Reliability Engineering (ISSRE 2000), October 2000. 110～121

3 顾玉良，王立福. 界面类对象建模技术研究. 计算机工程，1999，25(7)：21～23

4 杜栓柱，谭建荣，陆国栋. 基于界面构件关联图的软件功能测试技术. 计算机研究与发展，2002，39(2)：148～152

5 张涌，钱乐秋，王渊峰. 基于扩展有限状态机测试中测试输入数据自动选取的研究. 计算机学报，2003，26(10)：1295～1303

6 Memon A M, Pollack M E, Soffa M L. Using a Goal-driven Approach to Generate Test Cases for GUIs. In: Proceedings of the 21st International Conference on Software Engineering, ACM Press, May 1999. 257～266

7 Memon A M, Pollack M E, Soffa M L. Plan Generation for GUI Testing. The Fifth International Conference on Artificial Intelligence Planning and Scheduling, AAAI press, April 2000. 226～235

8 McCarthy J. Situations, actions and causal laws: [Technical report]. Stanford University, 1963. Reprinted in Semantic Information Processing (M. Minsky ed.), MIT Press, Cambridge, Mass, 1968. 410～417

9 梁伟晟，李磊. 基于表单的业务系统界面逻辑模型获取的研究. 计算机工程，已录用

10 Levesque H, Pirri F, Reiter R. Foundations for the Situation Calculus. Linköping Electronic Articles in Computer and Information Science, Linköping, Sweden, 1998, 3(18)

11 李磊. 会议报告：机器学习在需求工程中的应用. 见：第八届中国机器学习学术会议，2002

12 Reiter R. Proving properties of states in the situation calculus. Artificial Intelligence, 1993, 64: 337～351

（上接第 254 页）

语义，并且 UML 状态图直接反映了系统组件（类和 Aspects）的动态行为，所以并发系统的行为语义在设计和实现阶段保持了相关的一致性。今后的工作应该是结合具体 AOP（Aspect-Oriented Programming）环境如 AspectJ，实现并发软件系统的面向侧面代码自动化生成。

参 考 文 献

1 Kiczales G, Lamping J, Mendhekar A, Maeda C, et al. Aspect oriented programming [A]. In: Proceedings of the 11th Europeen Conf. Object-Oriented Programming [C]. Berlin: Springer-Verlag, 1997. 220～242

2 Andrews J H. Process-algebraic foundations of aspect oriented programming [A]. In: Proceedings of the Third International Conference on Metalevel Architectures and Separation of Crosscutting Concerns [C]. Berlin, Heidelberg: Springer-Verlag, Sept. 2001. 187～209

3 Loughran N, Rashid A. Mining Aspects [A]. In: Proceedings of Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design, Workshop [C]. Enschede, Holland: IEEE Computer Society Press, 2002. 15～23

4 Moreira A, Ara′ujo J, Brito I. Crosscutting Quality Attributes for Requirements Engineering [A]. In: Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering [C]. New York: ACM Press, 2002. 167～174

5 Pnueli A. A temporal logic of concurrent programs [A]. 18th IEEE Symposium on Foundation of Computer Science [C]. Providence, Rhde Island: IEEE Computer Society Press, 1977. 67～77

6 Richner T, Ducasse S. Recovering high level views of object-oriented applications from static and dynamic information [A]. In: Proceedings of International Conference on Software Maintenance [C]. Oxford, UK: IEEE CS Press, 1999. 13～22

7 Booch, Rumbaugh, Jacobson. The Unified Modeling Language User Guide [M]. New York: Addison Wesley, 1999. 25～37

8 Manna Z, Pnueli A. The Temporal Logic for Reactive and Concurrent Systems: Specification [M]. New York: Spring-Verlag, 1991. 103～176