基于范畴论的用户界面模式语言

贾 伟[1,2], 华庆一[1], 张敏军[1], 陈 锐[1], 姬 翔[1]

[1] (西北大学信息科学与技术学院 西安 710127)
[2] (宁夏大学新华学院 银川 750021)
(huaqy@nwu.edu.cn)

摘 要: 针对现有用户界面模式语言无法在高级抽象层次上准确描述用户界面模式之间关系, 导致不能为基于模式的用户界面开发提供有效的结构化支持的问题, 根据用户界面模式集合所具有的层次结构特点, 提出一种基于范畴论的用户界面模式语言. 首先利用对象和态射定义用户界面模式以及用户界面模式之间的关系, 得到用户界面模式范畴; 然后使用子范畴和函子定义用户界面模式集合的层次以及层次之间的关系; 最后使用自然变换定义用户界面模式集合层次关系之间的关系, 对用户界面模式集合中的关系进行更高抽象层次的描述. 实例结果表明, 文中提出的用户界面模式语言能够与开发过程紧密结合, 为开发人员理解、查找和重用用户界面模式提供了有效的支持.

关键词: 用户界面模式; 界面开发; 范畴论; 模式语言; 态射
中图法分类号: TP391.41

User Interface Pattern Language Based on Category Theory

Jia Wei[1,2], Hua Qingyi[1], Zhang Minjun[1], Chen Rui[1], and Ji Xiang[1]

[1] (*School of Information Science and Technology*, *Northwest University*, *Xi'an* 710127)
[2] (*Xinhua College*, *Ningxia University*, *Yinchuan* 750021)

Abstract: In order to address the problem that existing user interface (UI) pattern languages are unable to provide effective structuralized support for the pattern-based UI development caused by the lack of accurate descriptions of relationships between UI patterns on a high level of abstraction, a UI pattern language is presented based on category theory in line with the hierarchical structural features of UI pattern collections. Firstly, UI patterns and their relationships were defined by objects and morphisms to obtain a UI pattern category; then hierarchies of UI pattern collections and their relationships were defined by subcategories and functors; and finally, relationships between the hierarchical relationships of UI pattern collections were defined by natural transformations to describe the relationships of UI pattern collections on a higher level of abstraction. Experimental results show that the proposed UI pattern language can be closely integrated into the development process to effectively support the developers for their understanding, querying and reusing of the UI patterns.

Key words: user interface pattern; interface development; category theory; pattern language; morphism

---

收稿日期: 2015-10-06; 修回日期: 2016-09-18. 基金项目: 国家自然科学基金(61272286); 高等学校博士学科点专项科研基金(20126101110006). 贾 伟(1980—), 男, 博士研究生, 讲师, CCF 会员, 主要研究方向为人机交互、用户界面工程; 华庆一(1956—), 男, 博士, 教授, 博士生导师, CCF 会员, 主要研究方向为人机交互、用户界面工程; 张敏军(1979—), 男, 博士研究生, 主要研究方向为人机交互、用户界面工程; 陈 锐(1979—), 男, 博士研究生, CCF 会员, 主要研究方向为人机交互、用户界面工程; 姬 翔(1979—), 女, 博士研究生, 讲师, CCF 会员, 主要研究方向为人机交互、用户界面工程.

用户界面模式又称为人机交互模式，它捕捉了界面设计的成功因素，在满足功能需求的同时，包含了可用性，在人机交互领域有着广泛应用[1-5]. 随着用户界面模式数量的不断增大，用户界面模式之间的关系和层次也日趋复杂，如何使界面开发人员有效地使用用户界面模式，已成为急需解决的问题. 近年来，一些学者将模式语言思想[6]应用到了用户界面模式的研究中[7-8]，提出了一些用户界面模式语言[9-13]，并且将其应用于界面开发中[14-16]，但是这些用户界面模式语言都存在不足. 例如，模式语言标记语言(pattern language markup language, PLML)和扩展模式语言标记语言(extended pattern language markup language, XPLML)对用户界面模式之间的关系描述不足，导致用户界面开发人员局限于对单个用户界面模式的使用，无法有效地找到关联的用户界面模式. 由于本体在知识的表示、重用和查找中发挥了重要的作用[17]，使用本体思想的用户界面模式语言(ontology-based pattern language, OBPL)能够描述单个用户界面模式及其之间关系；但是它是以集合论为基础的，而集合论只能对具体的用户界面模式进行描述，无法对用户界面模式及其之间的关系进行高层次的抽象描述.

范畴论[18-19]是在传统集合论基础上发展起来的，与集合论相比，范畴论关注于对象之间联系的研究，能够描述更抽象的层次概念和关系，已经在一些领域的应用中取得了良好的效果[20-25]. 因此，本文提出一种基于范畴论的用户界面模式语言，以实现对用户界面模式的有效组织，并采用图的形式为界面开发人员提供更加直观的表示方式，有利于在界面的整个开发过程中辅助界面开发人员实现对整体界面的设计和开发.

1 相关知识

如图 1 所示，在界面开发过程中，每个开发阶段都会用到相应抽象层次上的用户界面模式，用户界面模式中信息与开发过程的使用到的目标、场景和层次等信息形成了映射关系.

用户界面模式语言的目的是将用户界面模式与界面开发过程紧密结合在一起，以下是用户界面模式语言的定义:

定义 1. 用户界面模式语言是能够对用户界面模式集合及其内部界面模式关系进行有效的组织，并使其具备完整层次结构的方法.

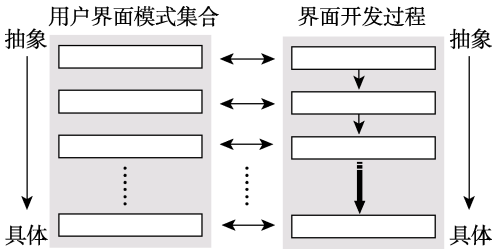


图 1 集合与界面开发的映射

针对现有用户界面模式语言的不足，本文将范畴论的结构化思想引入到用户界面模式语言中，以提高用户界面模式在界面开发中的有效性. 通常意义下，图是由一个非空的顶点集合和一个描述顶点之间多对多关系的边集合组成的一种数据结构，其表示为二元组 $G=(V,E)$；而在范畴论中，研究的重点是源节点集合与目标节点集合及其之间的关系，所以对图给出定义 2.

定义 2. 一个图是一个四元组 $G=(G_0,G_1,s,t)$，其中，G_0 是一个节点(对象)集合，G_1 是一个箭头(态射)集合，s 将箭头映射到节点的源对象，t 将箭头映射到节点的目标对象.

在定义了图的概念的基础上,下面给出范畴论的一些基本概念的定义.

定义 3. 一个范畴是一个图，带有 2 个函数 $c:G_2\to G_1$ 和 $u:G_0\to G_1$；G_2 是路径长度为 2 的集合，G_0 中的元素称为源对象，G_1 中的元素称为目标对象. 函数 c 称为合成，如果 (g,f) 是可合成的箭头对，$c(g,f)$ 写成 $g\circ f$，称为 g 和 f 的合成. 如果 a 是一个对象，$u(a)$ 表示 i_a，i_a 是对象 a 的恒等，那么 2 个函数满足以下 4 个性质:

1) $g\circ f$ 的源是 f 的源，$g\circ f$ 的目标是 g 的目标;

2) $(h\circ g)\circ f=h\circ(g\circ f)$，只要当其中的一边有定义;

3) i_a 的源和目标都是 a;

4) 如果 $f:a\to a$，那么 $f\circ i_a=i_a\circ f=f$.

由定义 3 可知，一个范畴包括了以下 3 个组成部分:

1) 由多个对象组成的集合;

2) 由多个态射组成的集合;

3) 由复合律、结合律和恒等律组成的态射合成法则.

定义 4. 令 I 与 G 是图，G 中形式为 I 的一个图表是一个图同态 $D:I\to G$，I 称为图表 D 的形状图.

定义 **5.** 如果图表 $D:G\to I$，I 的一个对象 i 以及一族由 G 的对象加标的射 $u_a:D_a\to i$ 组成的射族 $\{u_a:D_a\to i|a\in G\}$，那么对 G 的每个射 $s:a\to b$ 有 $u_b\circ D_s=u_a$，则图表 D 称为余极限.

定义 **6.** 范畴 C_1 是范畴 C_2 的子范畴, 其中:

1) C_1 中的所有对象都是 C_2 中的对象, C_1 中的所有态射都是 C_2 中的态射;

2) C_1 中的态射的源和目标与 C_2 中态射的源和目标一样;

3) 如果 $f:a\to b$ 和 $g:b\to e$ 在 C_1 中, 则 C_2 中的合成 $f\circ g$ 在 C_1 中, 它也是 C_2 中的合成;

4) 如果 a 是 C_1 的一个对象, 则 C_2 的恒等态射 i_a 在 C_1 中.

定义 **7.** 函子 $F:C\to B$ 是一对满足下列条件的函数 $F_0:C_0\to B_0$ 和 $F_1:C_1\to B_1$:

1) 如果 $f:a\to b$ 在 C 中, 那么 $F_1(f):F_0(a)\to F_0(b)$ 在 B 中;

2) 对 C 中的任何对象 a，$F_1(i_a)=i_{F_0(a)}$;

3) 如果 $f\circ g$ 定义在 C 中, 那么 $F_1(g)\circ F_1(f)$ 定义在 B 中, 且 $F_1(g\circ f)=F_1(g)\circ F_1(f)$.

从上述范畴论基本概念的定义可以看出, 范畴论主要研究的是对象和对象之间的态射. 表 1 所示为本文引入范畴论的结构化思想后所涉及的一些范畴论概念及其相对应的用户界面模式含义. 通过范畴论的概念可以对用户界面模式集合进行更高层次的抽象描述, 实现与界面开发的结合, 有助于开发人员快速、有效地理解用户界面模式在整个开发过程中的整体结构性质.

表 **1**　范畴论与用户界面模式含义

范畴论概念	用户界面模式含义
对象	用户界面模式
态射	用户界面模式间的关系
范畴	用户界面模式语言
子范畴	用户界面模式的层次
函子	用户界面模式层次间的关系
自然变换	用户界面模式层次关系间的关系
推出	用户界面模式的组合过程

2　本文用户界面模式语言

用户界面模式之间存在组合、相似和等价等关系[26], 形成了具有复杂关系和多层次的用户界面模式网络, 本文使用范畴论理论对用户界面模式之间的关系和层次的定义. 如图 2 所示, P 表示任意一个界面模式, g 表示同一层次中的界面模式之间的关系, f 表示不同层次的界面模式之间的关系, n 表示不同层次的界面模式关系之间的关系, J 表示不同界面模式层次关系之间的关系. 将用户界面模式集合组织成一个由抽象到具体的多层次结构, 用户界面模式语言称为范畴; 每个界面模式层次称为子范畴, 每层包含相同抽象层次的用户界面模式集合, P 称为对象, g 称为态射, 不同界面模式层次之间的关系称为函子, 函子包括 f 和 n, J 称为自然变换.

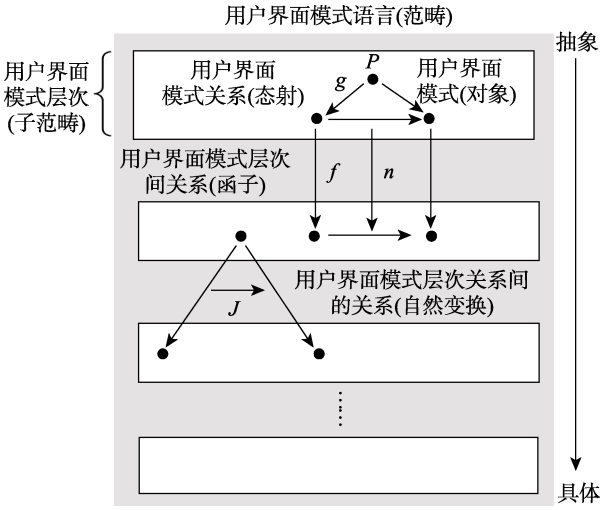


图 2　用户界面模式语言结构

基于范畴论的用户界面模式语言的相关概念定义如下:

定义 **8.** 设对象 $P=(\theta,T,B,H)$ 是一个四元组. 其中, θ 是用户界面模式标示符; T 是用户界面模式实现的目标集合, 包括任务目标和可用性目标; B 是属性集合, 包括用户界面模式的问题属性、上下文属性和解决方案属性; H 是用户界面模式在整个用户界面模式集合中所在的层次; 称 P 是用户界面模式.

定义 **9.** 设 2 个用户界面模式 $P_1=(\theta_1,T_1,B_1,H_1)$ 和 $P_2=(\theta_2,T_2,B_2,H_2)$，$P_1$ 和 P_2 之间的态射 $M:P_1\to P_2$ 包括标示符态射 $k:\theta_1\to\theta_2$，目标态射 $l:T_1\to T_2$，属性态射 $w:B_1\to B_2$，层次态射 $r:H_1\to H_2$，称 M 是 2 个用户界面模式间的关系.

定义 **10.** 设 $L=(P,M)$ 是一个二元组. 其中, P 是用户界面模式的对象集合; M 是用户界面模式中的态射集合, 并且如果存在 $P_1=(\theta_1,T_1,B_1,H_1)$, $P_2=(\theta_2,T_2,B_2,H_2)$, $P_3=(\theta_3,T_3,B_3,H_3)$ 和 $P_4=(\theta_4,$

$T_4, B_4, H_4)$; 若满足以下 3 个条件:

1) 如果 $f: P_1 \to P_2$, $g: P_2 \to P_3$, 则存在复合态射 $v = f \circ g$, 即 $v: P_1 \to P_3$;

2) 如果 $f: P_1 \to P_2$, $g: P_2 \to P_3, h: P_3 \to P_4$,则 $(f \circ g) \circ h = f \circ (g \circ h)$;

3) 如果对每一个对象 P, 存在一个恒等态射 $i_P: P \to P$, 使得对任意的 $f: P_1 \to P_2$ 都有 $i_P \circ f = f \circ i_P = f$;

则称 L 是一个用户界面模式范畴.

命题 **1.** 用户界面模式语言是由用户界面模式描述、用户界面模式之间的态射和态射合成法则组成的用户界面模式范畴.

根据定义 1 和定义 3 可知, 命题 1 成立.

定义 **11.** 设 $L_1 = (P_1, M_1)$ 和 $L_2 = (P_2, M_2)$ 是 2 个用户界面模式范畴, 若满足以下 4 个条件:

1) $P_2 \subseteq P_1$;

2) 对 $\forall a, b \in P_2$, $\forall f \in M_2$, 如果 $f: a \to b$, 则必有 $f \in M_1$;

3) 如果 $f: a \to b$ 和 $g: b \to e$ 在 L_2 中, 那么 L_1 中的复合态射 $f \circ g$ 也是在 L_2 中的复合态射;

4) 如果 P_1 是 L_2 的一个对象, 那么 L_1 中的恒等态射 $i_{L_1(P_1)}$ 也是在 L_2 中的恒等态射;

则称 L_2 是 L_1 子范畴.

命题 **2.** 用户界面模式语言中的层次由用户界面模式子范畴组成.

用户界面模式语言是由抽象到具体的多个层次组成的, 每一层中的用户界面模式都是用户界面模式语言的组成部分, 再根据命题 1, 显然命题 2 成立.

定义 **12.** 设 $L_1 = (P_1, M_1)$ 和 $L_2 = (P_2, M_2)$ 是 2 个用户界面模式范畴, $E: L_1 \to L_2$ 是一个映射. 若满足以下 4 个条件:

1) 对所有 $a \in P_1$, 都有 $E(a) \in P_2$;

2) E 是从 M_1 到 M_2 的同态映射, 对所有的 $m \in M_1$, 都有 $E(m) \in M_2$;

3) 对 L_1 中的任何对象 a, 有 $E(i_a) = i_{E(a)}$;

4) 对于 $f, g \in M_1$, $E_1(g \circ f) = E(g) \circ E(f)$;

则称 E 是用户界面模式范畴之间的函子.

命题 **3.** 用户界面模式语言中层次之间的关系是函子.

根据命题 1 和命题 2, 以及不同层次的用户界面模式之间存在组合、相似和等价等关系, 显然命题 3 成立.

定义 **13.** 设 $L_1 = (P_1, M_1)$ 和 $L_2 = (P_2, M_2)$ 是 2 个用户界面模式范畴, $E: L_1 \to L_2$ 和 $N: L_1 \to L_2$ 是 2 个函子, $J: E \to N$ 是一个映射, 该映射由 L_2 中一族 L_1 的节点加标的射 J_a 确定. 若满足以下 2 个条件:

1) 对 L_1 中的每个节点 a, 都有 $J_a: E(a) \to N(a)$;

2) 如图 3 所示的图表交换, 对 L_1 中的任何态射 $f: a \to b$, 都有 $N(f) \circ J_a = J_b \circ E(f)$;

则称 J 是一个自然变换.

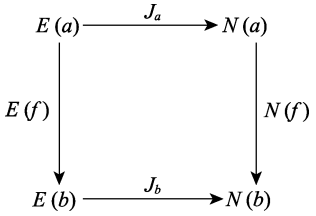


图 3 图表交换

命题 **4.** 用户界面模式语言中的层次关系之间存在的关系是自然变换.

根据命题 3 可知, 不同层次的用户界面模式之间存在的关系称为函子, 当存在多个函子时, 如果函子来自于同一个用户界面模式, 那么函子之间存在也会存在关系, 因此, 命题 4 成立.

在定义 9、定义 12 和定义 13 中, 利用态射、函子和自然变换分别定义了用户界面模式语言中存在的用户界面模式之间的关系, 用户界面模式层次之间的关系和用户界面模式层次关系的关系, 通过这 3 种关系可以将用户界面模式集合有效地组织在一起, 界面开发人员可以利用这 3 种关系查找用户界面模式的使用情况.

下面给出用户界面模式语言中推出的定义.

定义 **14.** 如图 4 所示, 设 P_1, P_2 和 P_3 是用户界面模式范畴的对象; $M_1: P_1 \to P_2$ 和 $M_2: P_1 \to P_3$ 是用户界面模式范畴态射. 若满足以下 2 个条件:

1) $M_1 \circ M_3 = M_2 \circ M_4$;

2) 对任意其他对象 P_5 和态射 $M_5: P_2 \to P_5$ 和 $M_6: P_3 \to P_5$, 存在唯一的态射 $M_7: P_4 \to P_5$;

则称 P_4 及 2 个态射 $M_3: P_2 \to P_4$ 和 $M_4: P_3 \to P_4$ 是一个推出.

在范畴论中, 推出的作用是实现对象之间的融合, 即形成融合和. 在一个范畴中, 2 个对象通过中间对象和态射联系在一起, 利用推出可以得到一个包含原来 2 个对象属性的新的对象. 为了证明用户界面模式的组合过程是推出, 给出范畴论

中的相关概念定义和命题.

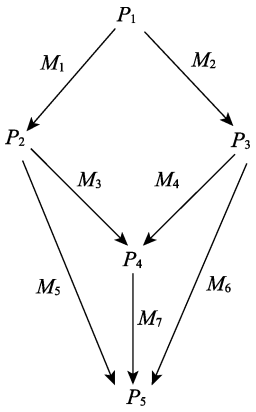


图 4　用户界面模式范畴的推出

定义 15. 设 $f,g:P_1\to P_2$ 是一对平行态射，若态射 $n:P_2\to P_3$ 满足以下 2 个条件:

1)　$f\circ n=g\circ n$;

2) 对任意的态射 $n':P_2\to P_3'$ 满足 $f\circ n'=g\circ n'$ 成立;

则称 $n:P_2\to P_3$ 是 $f,g:P_1\to P_2$ 的余等值子.

定义 16. 设 D 是范畴 C 的一个离散图表，若该图表的余极限 $\{u_a:D_a\to i\,|\,a\in G\}$ 存在，则称 $\{u_a:D_a\to i\,|\,a\in G\}$ 是对象族 $\{D_a\}_{a\in G}$ 的余积.

命题 5. 设 $\{u_a:D_a\to i\,|\,a\in G\}$ 是图表 $D:G\to I$ 的余极限，则 u_a 是一族集体满态射，即对任意一对态射 $f,g:i\to j$. 若 $f\circ u_a=g\circ u_a$ 对任意的 $a\in G$ 成立，则 $f=g$.

下面证明用户界面模式组合过程是推出.

证明. 设 P_1 是需求用户界面模式，P_2 和 P_3 是组成需求用户界面模式的 2 个子用户界面模式，P_4 是 P_2 和 P_3 是组合后的用户界面模式. 由 P_1，P_2，P_3 和 P_4 之间的关系可知，存在如图 5 所示的态射 $f:P_1\to P_2$，$g:P_1\to P_3$，$f':P_2\to P_4$ 和 $g':P_3\to P_4$. 由余极限的定义可知，存在 P_2 和 P_3 的余积 $P_2\xleftarrow{f'}U\xrightarrow{g'}P_3$，$n:U\to P_4$ 是平行态射 $f\circ f',g\circ g':P_1\to U$ 的余等值子.

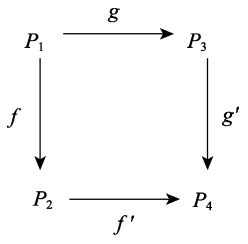


图 5　用户界面模式组合

1) 由如图 5 和上述说明显然可以看出 $f\circ f'=g\circ g'$，即满足交换性.

2) 设图 6 的界面模式组合是一个任意交换方形，P_4' 是由 P_2 和 P_3 组合后的界面模式，根据余积定义可知，存在一个态射 $<v,t>:U\to P_4'$，使得 $v=f'\circ<v,t>$，$t=g'\circ<v,t>$，得到 $f\circ f'\circ<v,t>=g\circ g'\circ<v,t>$. 由于 $n:U\to P_4$ 是平行态射 $f\circ f'$，$g\circ g':P_1\to U$ 的余等值子，则存在态射 $z:P_4\to P_4'$ 使得 $<v,t>=n\circ z$，得到 $v=f'\circ<v,t>=f'\circ n\circ z$，$t=g'\circ<v,t>=g'\circ n\circ z$. 由命题 5 可知，态射 z 是唯一的.

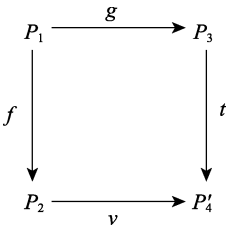


图 6　任意用户界面模式组合

通过 1), 2)和推出的定义可知，一个用户界面模式组合过程是一个推出，因此，利用推出实现用户界面模式组合是可行的.　　证毕.

本文定义的用户界面模式语言使用户界面模式具备了可理解性和追踪性. 从横向来看，同一层次内的态射、余极限和推出有助于界面开发人员理解和查找用户界面模式；从纵向来看，不同层次之间的映射函子能够帮助界面开发人员理解和查找各个界面开发阶段的用户界面模式，通过函子和自然变换能够分析不同抽象层次间的用户界面模式.

3　实例研究

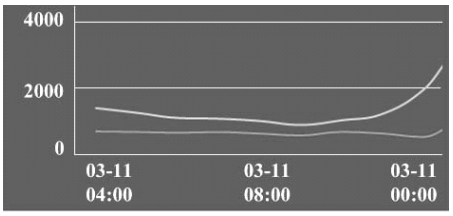
本文通过实例来说明基于范畴论的用户界面模式语言的特点和在用户界面开发中的应用.

3.1　用户界面模式语言的特点分析

3.1.1　用户界面模式语言的结构重用特点

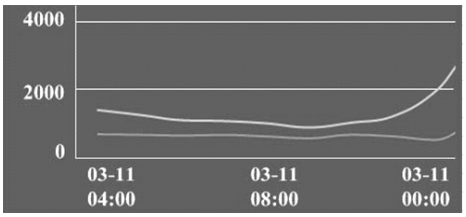
由于采用了范畴论的结构化思想，使用户界面模式语言具有了重用用户界面模式及其关系的特点. 下面通过实例进行说明，图 7 所示的 3 个用户界面模式，数据信息展示界面模式的目标是为用户提供易于理解的数据信息展现界面，该用户界面模式包含了数据总览图界面模式和表格界面模式；数据总览图界面模式以图形的形式为用户

提供展现数据的总体情况；表格界面模式以表格的形式为用户提供数据的详细内容.



事件	发生次数
页面信息	64802
频道输出	17108
点击左侧菜单	11892

a. 数据信息展示界面模式



b. 数据总览图界面模式

事件	发生次数
页面信息	64802
频道输出	17108
点击左侧菜单	11892

c. 表格界面模式

图 7 数据展示界面模式

设 P_1 是需求数据信息展示界面模式, P_1 属于 H_1 层，包含了实现的全局目标；在 H_2 层, P_2 是数据总览图界面模式； P_3 是表格界面模式, P_1' 是具体的数据信息展示界面模式. 如 8 图所示, P_1 属于 H_1 层, P_2, P_3 和 P_1' 属于 H_2 层； P_1, P_2, P_3 和 P_1' 之间通过态射 f, g, f' 和 g' 形成了层次结构关系.

从图 8 可以看出，在界面开发中，如果 H_1 层的设计目标 A 与 H_1 层的 P_1 中的全局目标相同，可以将由 P_1, P_2, P_3 和 P_1' 及其关系组成的层次结构作为一个整体重用到界面开发中，比选择单个用户界面模式更加有效. 现有的用户界面模式语言不具有重用用户界面模式及其关系的特点.

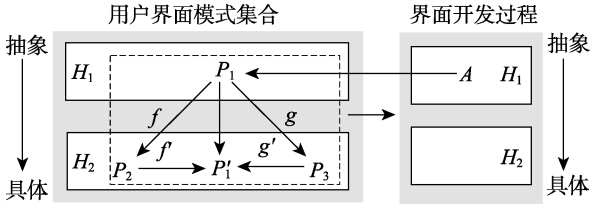
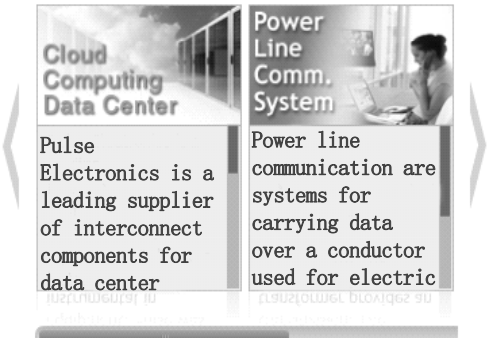


图 8 结构重用

3.1.2 用户界面模式语言的保结构映射特点

函子是 2 个用户界面模式层次间的保结构映射，目的是将上一层中的对象属性和态射属性映射到下一层中，实现不同层次间的同态映射. 下面通过实例说明函子的作用，导航界面模式通常将主导航界面模式和次导航界面模式配合在一起使用，如图 9 所示，列表菜单式导航界面模式属于主导航界面模式，以列表的形式为用户展现导航信息；页面轮盘式导航界面模式属于次导航界面模式，在页面中以滚动条快速浏览离散图片的形式为用户展现导航信息.

a. 列表菜单式导航界面模式



b. 页面轮盘式导航界面模式

图 9 导航界面模式

设 P_1 是导航界面模式, P_2 是主导航界面模式, P_3 是次导航界面模式, P_4 是列表菜单式导航界面模式, P_5 是页面轮盘式界面模式. 如图 10 所示, P_1 属于 H_1 层, P_1 由 P_2 和 P_3 组成, P_2 和 P_3 属于 H_2 层, P_2 具体界面模式为 P_4， P_3 具体界面模式为 P_5, 态射 t 表示 P_2 和 P_3 间的组合关系, P_4 和 P_5 属于 H_3 层, 态射 v 表示 P_4 和 P_5 间的组合关系.

从图 10 可以看出, P_2 和 P_3 通过函子 f, g 和 n, 将 P_2, P_3 和 t 映射到 H_3 层的 P_4, P_5 和 v 中, 保证了在 H_3 层中依然保留 H_2 层中 P_2 和 P_3 的各属性信息及其组合关系.

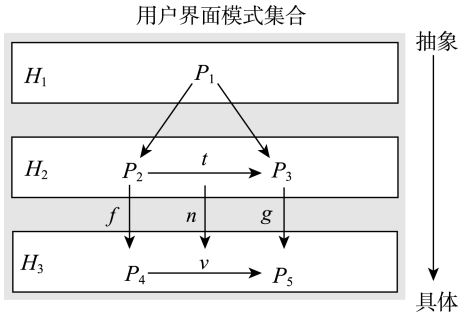
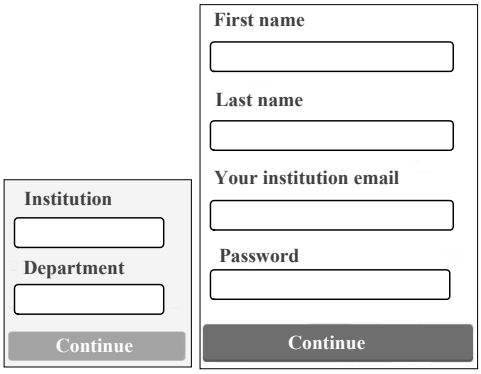


图 10　函子的作用

自然变换是 2 个函子间的保结构映射，目的是保证 2 个函子间的同态映射. 下面通过实例说明自然变换作用，多步骤表单界面模式和长表单界面模式如图 11 所示，其中多步骤表单界面模式是在多个页面中实现账号注册，长表单界面模式是在一个页面中实现账号注册.



a. 多步骤表单界面模式

b. 长表单导航界面模式

图 11　注册表单界面模式

设 P_1 是注册账号界面模式，P_2 是多步骤表单界面模式，P_3 是长表单界面模式. 如图 12 所示，P_1 属于 H_1 层，P_1 包括 P_2 和 P_3 两种用户界面模式，P_2 和 P_3 属于 H_2 层.

由自然变换的定义可知，P_1 与 P_2，P_3 都存在同态映射，表明 P_2 和 P_3 包含 P_1 的对象属性和态射

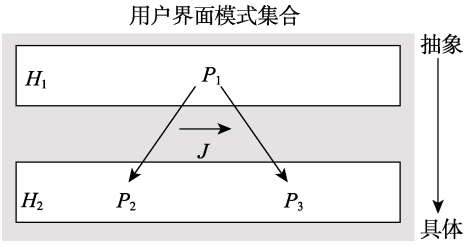


图 12　自然变换的作用

属性；H_1 层对用户界面模式使用场景的描述较为抽象，H_2 层对用户界面模式使用场景的描述更加具体；P_2 和 P_3 的场景不同，自然变换 J 表示如果设备屏幕较小或填写的内容较多，需要在多个页面实现账号注册，则使用 P_2；如果设备屏幕较大或在填写的内容较少，需要在一个页面中显示所有需填写的注册信息，则使用 P_3.

通过上述 2 个实例说明了函子和自然映射都属于同态映射，这样就使基于范畴论的用户界面模式语言具有层次间用户界面模式保结构映射和层次内用户界面模式保结构映射的特点，从而在界面开发过程中，开发人员能够根据设计目标或场景灵活选用用户界面模式，并且始终保证用户界面模式集合中的对象属性和关系保持一致. 现有的用户界面模式语言不具有保结构映射的特点.

3.2　用户界面模式语言的应用实例

实例中涉及的用户界面模式来自文献[12]，下面通过实例说明利用推出实现用户界面模式的组合的具体步骤.

设需要设计一个在网站购买商品的用户界面，界面开发中需要一个实现对商品信息浏览和记录候选商品信息的用户界面模式，该用户界面模式由商品浏览界面模式和购物车界面模式组合实现.

设 $P_1=(\theta_1,T_1,B_1,H_1)$ 是界面开发需要的用户界面模式，$P_2=(\theta_2,T_2,B_2,H_2)$ 是商品浏览界面模式，$P_3=(\theta_3,T_3,B_3,H_3)$ 是购物车界面模式，$P_4=(\theta_4,T_4,B_4,H_4)$ 是 P_1 和 P_2 组合后的新用户界面模式. P_1 表示了 P_2 和 P_3 的用户界面模式需求的交集，包含了 P_2 和 P_3 要实现的全局目标；P_1 与 P_2，P_3 存在态射 $f:P_1\to P_2$ 和 $g:P_1\to P_3$，P_1 与 P_2 的态射 f 包括 $k_1:\theta_1\to\theta_2$，$l_1:T_1\to T_2$，$w_1:B_1\to B_2$ 和 $r_1:H_1\to H_2$，P_1 与 P_3 的态射 g 包括 $k_2:\theta_1\to\theta_3$，$l_2:T_1\to T_3$，$w_2:B_1\to B_3$ 和 $r_2:H_1\to H_3$，P_2，P_3 与 P_4 的态射为 $f':P_2\to P_4$ 和 $g':P_3\to P_4$.

对 P_1，P_2 和 P_3 的描述如下:

P_1 {identifier: shopping_browse

task_goal: find_information, browse_information, record_information

usability_goal: find_effciency, layout_reasonable, shopping_convenient

problem: where_find, what_look_for, how_record_information

solution: main_navigation, shopping_cart

context: web, *E*_commerce, PC

level: experience_level

}

P_2 {identifier: main_navigation

task_goal: find_information, browse_information

usability_goal: find_effciency, layout_reasonable

problem: where_find, what_look_for

solution: fixed_position, menu_navigation, top_page

context: web, *E*_commerce, PC

level: task_level

}

P_3 {identifier: shopping_cart

task_goal: record_information

usability_goal: shopping_convenient

problem: how_record_information

solution: shopping_cart(add_information, del_information, browse_information, modify_information)

context: web, *E*_commerce, PC

level: task_level

}

P_2 和 P_3 的组合算法步骤如下:

输入. P_1, P_2, P_3, f, g.

输出. P_4, f', g'.

Step1. 初始化设置 $P_4=\varnothing$, $f'=\varnothing$, $g'=\varnothing$, $\theta_4=\theta_1$, $\theta_4=\theta_4\cup\theta_2\cup\theta_3$, $k_1'=k_1'\cup k_1$, $k_2'=k_2'\cup k_2$, $H_4=H_1$, $H_4=H_4\cup H_2\cup H_3$, $r_1'=r_1'\cup r_1$, $r_2'=r_2'\cup r_2$.

Step2. 对于 T_1 中的所有 t_1, 如果 $l_1(t_1)$ 与 $l_2(t_1)$ 等价; 则将 t_2, t_3 与 T_4 合并, 将 l_1 与 l_1' 合并, 将 l_2 与 l_2' 合并.

Step3. 对于 B_1 中的所有 b_1, 如果 $w_1(b_1)$ 与 $w_2(b_1)$ 等价; 则将 b_2, b_3 与 B_4 合并, 将 w_1 与 w_1' 合并, 将 w_2 与 w_2' 合并.

Step4. 对于 T_1 中的所有 t_1, 如果 $l_1(t_1)$ 不属于 T_2 ,则将 t_2 与 T_4 合并, 将 $t_2\to t_4$ 与 l_1' 合并.

Step5. 对于 T_1 中的所有 t_1, 如果 $l_2(t_1)$ 不属于 T_3 ,则将 t_3 与 T_4 合并, 将 $t_3\to t_4$ 与 l_2' 合并.

Step6. 对于 B_1 中的所有 b_1, 如果 $w_1(b_1)$ 不属于 B_2, 则将 b_2 与 B_4 合并, 将 $b_2\to b_4$ 与 w_1' 合并.

Step7. 对于 B_1 中的所有 b_1, 如果 $w_2(b_1)$ 不属于 B_3, 则将 b_3 与 B_4 合并, 将 $b_3\to b_4$ 与 w_2' 合并.

Step8. 输出 P_4, f', g', 算法结束.

利用组合算法得到组合后的 P_4 描述如下:

P_4 {identifier: shopping_browse, main_navigation, shopping_cart

task_goal: find_information, browse_information, record_information

usability_goal: find_effciency, layout_reasonable, shopping_convenient

problem: where_find, what_look_for, how_record_information

solution: fixed_position, menu_navigation, top_page, shopping_cart(add_information, del_information, browse_information, modify_information)

context: web, *E*_commerce, PC

level: experience_level, task_level

}

用户界面模式组合过程和结果如图 13 所示, 利用推出将 2 个具有内在联系的用户界面模式 P_2 和 P_3 构建成一个更大的用户界面模式 P_4. 本文提出的基于范畴论的用户界面模式语言对用户界面模式 P_1, P_2, P_3 和 P_4 进行了准确的结构化描述, 在描述中包含了用户界面模式标识符 identifier、任务目标 task_goal、可用性目标 usability_goal、设计问题 problem、解决方案 solution、场景 context 和层次 level, 这些信息都是开发过程中开发人员必须了解的信息.

如图 14 所示, 从开发过程上看, P_1, P_2, P_3 和 P_4 中包含的信息与界面设计过程中的目标、场景和层次信息形成了映射关系. 通过 P_4 的描述可以看出, 组合后的用户界面模式 P_4 包含了 P_1 的需求, 满足了 P_1 的全局目标, 并保存了子用户界面模式的相关信息, 使用户界面模式与具体的开发过程的紧密结合, 这为开发人员在实际应用中理解和重用用户界面模式提供重要的参考依据. 从结构上看, 通过态射 f 和 g 清晰地表示了开发中需要的用户界面模式 P_1 需要 P_2 和 P_3 来实现; 态射 f' 和 g' 表示 P_2 和 P_3 组成了 P_4, 而且 P_2 和 P_3 中包含的

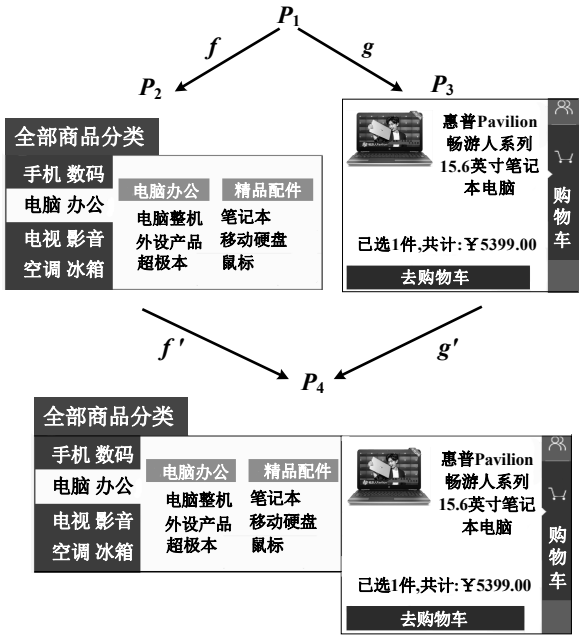


图 13 用户界面模式组合

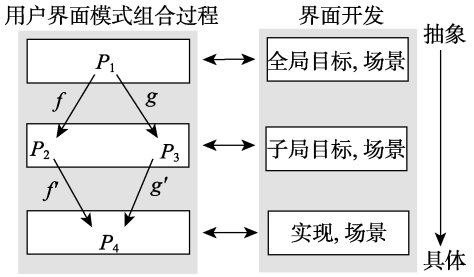


图 14　组合过程与界面开发的映射

各类信息也通过态射 f' 和 g' 传递给 P_4; 从设计需求的用户界面模式到最终用户界面模式组合的整个过程使用了范畴论的推出概念并以图的形式进行表示, 这为开发人员理解需求目标和用户界面模式关系提供了较为简洁和明确的表达方式, 满足了开发人员快速理解开发目标和用户界面模式结构的需求. 因此, 在辅助开发人员有效地使用用户界面模式方面, 本文提出的基于范畴论的用户界面模式语言是可行的.

为了便于量化评价各用户界面模式语言, 本文根据用户界面模式语言在开发过程中的使用情况, 给出一个评价语言术语集合和评分标度. 如表 2 所示, 评价等级分为好、较好、一般、较差、差 5 种评价等级, 对应的评分标度值分别是 5, 4, 3, 2, 1.

如表 3 所示, 将用户界面模式语言 PLML, XPLML, OBPL 与本文提出的基于范畴论的界面模式语言(category theory-based pattern language, CTBPL)进行评价比较. CTBPL 能够进行高层次的抽象描述, 将用户界面模式抽象为对象, 忽略开发人员不需要的细节问题, 用态射来表示用户界面模式之

表 **2**　评价术语及评分

评价等级	评分值
好	5
较好	4
一般	3
较差	2
差	1

表 **3**　评价结果的比较

用户界面模式语言	描述关系	理解组成和结构	查找的难易度	结合度	综合评价
CTBPL	5	5	4	5	19
PLML	1	1	1	1	4
XPLML	3	3	2	2	10
OBPL	4	4	3	4	15

间的多种关系, 将实例中的整个过程抽象为推出, 为开发人员提供便于理解的描述方式. 界面开发是一个反复迭代的过程, 开发人员要不断地回溯前面开发阶段中使用过的用户界面模式, CTBPL 保留了前面开发阶段中使用过的用户界面模式的相关信息, 能够与开发过程紧密结合, 所以 CTBPL 的综合评价最高.

PLML 不能描述 P_1, P_2, P_3 和 P_4 之间的语义关系, 没有提供用户界面模式的结构图, 导致开发人员在开发过程中难以理解用户界面模式的组成和结构, 查找用户界面模式较为困难, 与开发过程结合度最差.

XPLML 在 PLML 的基础上进行了扩展, 通过添加语义元数据来描述 P_1, P_2, P_3 和 P_4 之间的语义关系, 提高了描述用户界面模式之间关系的能力. 但 XPLML 缺乏简洁的表示方式, 在描述用户界面模式和辅助开发人员理解用户界面模式方面评价一般. 此外, XPLML 依然在扁平结构下采用文档类型的定义, 不支持结构化的查找, 不利于用户界面模式的回溯查找, 在查找用户界面模式方面以及与开发过程的结合程度方面评价较差.

OBPL 利用本体构建用户界面模式语言, 增强了对 P_1, P_2, P_3 和 P_4 之间关系的描述, 能够帮助开发人员理解查找用户界面模式, 但是OBPL缺乏简洁的方式表示用户界面模式及其之间的组成、结构和关系, 导致在描述用户界面模式、辅助开发人员理解用户界面模式和查找用户界面模式方面还存在不足. 此外, OBPL 能够辅助开发人员回溯使用过的用户界面模式, 在与开发过程的结合程度方面评价较好, 但是其不能描述需求分析中的全局目标和子目标关系, 只能描述具体用户界面模式的语义关系, 造成无法按照开发需求将用户界面模式的各类关系抽象成更高层次的概念, 与开发过程的结合还存在不足.

下面通过用户界面模式语言可以描述的用户界面模式层次结构占全部用户界面模式集合层次结构的比重、可以描述的用户界面模式关系占全部用户界面模式集合关系总数的比重, 以及在开发中由一个用户界面模式能查找到的其他相关的用户界面模式占全部相关用户界面模式的平均比重这3项指标, 衡量CTBPL, PLML, XPLML和OBPL在描述用户界面模式集合和查找相关用户界面模式方面的表现. 如图 15 所示, CTBPL 对应的 3 项指标明显高于 PLML, XPLML 和 OBPL 对应的 3 项指标,

说明CTBPL在描述用户界面模式集合和查找相关用户界面模式方面要优于PLML, XPLML和OBPL.

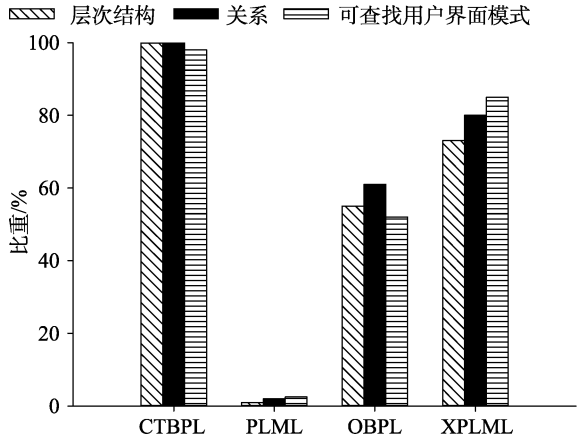


图 15 不同用户界面模式语言 3 项指标的比较

通过上述分析与比较可以看出, 现有的用户界面模式语言在实际应用中都存在不足之处, 本文提出的 CTBPL 在各方面的表现较好, 因此, 将本文提出的CTBPL应用在界面开发中是有效的.

4 结 语

范畴论具备良好的分析和表达能力, 注重对象之间的内在联系, 为界面开发人员对用户界面模式的理解、查找和重用提供了理论工具. 本文利用范畴论的结构化思想和原理给出了用户界面模式语言, 通过分析范畴论在用户界面模式语言中的特点和在用户界面模式组合中的作用, 说明基于范畴论的用户界面模式语言的有效性和可行性.

参考文献(References):

[1] Engel J, Märtin C, Forbrig P. Tool-support for pattern-based generation of user interfaces[C] //Proceedings of the 1st International Workshop on Pattern-Driven Engineering of Interactive Computing Systems. New York: ACM Press, 2010: 24-27

[2] Vanderdonckt J, Simarro F M. Generative pattern-based design of user interfaces[C] //Proceedings of the 1st International Workshop on Pattern-Driven Engineering of Interactive Computing Systems. New York: ACM Press, 2010: 12-19

[3] Nguyen T D, Vanderdonckt J. User interface master detail pattern on Android[C] //Proceedings of the 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems. New York: ACM Press, 2012: 299-304

[4] Thongmool G, Phankokkruad M. Analysis of interaction user interface patterns and usability study in computer assisted instruction for Tablet PC[C] //Proceedings of the IEEE International Conference on Control System, Computing and Engineering. Los Alamitos: IEEE Computer Society Press, 2014: 472-477

[5] Wetchakorn T, Prompoon N. Method for mobile user interface design patterns creation for iOS platform[C] //Proceedings of the 12th International Joint Conference on Computer Science and Software Engineering. Los Alamitos: IEEE Computer Society Press, 2015: 150-155

[6] Alexander C W, Ishikawa S, Silverstein M, *et al*. A pattern language: towns, buildings, construction[M]. Oxford: Oxford University Press, 1977

[7] Todd E, Kemp E, Phillips C. What makes a good user interface pattern language?[C] //Proceedings of the 5th Conference on Australasian User Interface. Darlinghurst: Australian Computer Society Press, 2004, 28: 91-100

[8] Seffah A. The evolution of design patterns in HCI: from pattern languages to pattern-oriented design[C] //Proceedings of the 1st International Workshop on Pattern-Driven Engineering of Interactive Computing Systems. New York: ACM Press, 2010: 4-9

[9] Fogli D, Provenza L P, Bernareggi C. A design pattern language for accessible web sites[C] //Proceedings of the International Conference on Advanced Visual Interfaces. New York: ACM Press, 2010: 307-310

[10] Deng J, Kemp E, Todd E G. Focussing on a standard pattern form: the development and evaluation of MUIP[C] //Proceedings of the 7th ACM SIGCHI New Zealand Chapter's International Conference on Computer-Human Interaction. New York: ACM Press, 2006: 83-90

[11] Kruschitz C. XPLML: a HCI pattern formalizing and unifying approach[C] //Proceedings of the CHI'09 Extended Abstracts on Human Factors in Computing Systems. New York: ACM Press, 2009: 4117-4122

[12] Welie M V, van der Veer G C. Pattern languages in interaction design: structure and organization[C] //Proceedings of IFIP TC13 International Conference on Human-Computer Interaction. Amsterdam: IOS Press, 2003: 527-534

[13] Henninger S, Ashokkumar P. An ontology-based infrastructure for usability design patterns[OL]. [2015-10-06]. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.77.2053

[14] Li Xiang, Hua Yixin, Wang Shuang, *et al*. The research of pattern language for GIS user interface[J]. Journal of Image and Graphics, 2010, 15(7): 1014-1022(in Chinese)
(李 响, 华一新, 王 双, 等. 通过模式语言研究 GIS 用户界面[J]. 中国图象图形学报, 2010, 15(7): 1014-1022)

[15] Lin J, Landay J A. Employing patterns and layers for early-stage design and prototyping of cross-device user interfaces[C] //Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. New York: ACM Press, 2008: 1313-1322

[16] Radeke F, Forbrig P, Seffah A, *et al.* PIM tool: support for pattern-driven and model-based UI development[C] //Proceedings of the 5th International Conference on Task Models and Diagrams for Users Interface Design. Heidelberg: Springer, 2006: 82-96

[17] Hu Yujie, Li Shanping, Guo Ming. Ontology-based product knowledge representation[J]. Journal of Computer-Aided Design & Computer Graphics, 2003, 15(12): 1531-1537(in Chinese)

(胡玉杰, 李善平, 郭　鸣. 基于本体的产品知识表达[J]. 计算机辅助设计与图形学学报, 2003, 15(12): 1531-1537)

[18] Lawvere F W, Schanuel S H. Conceptual mathematics: a first introduction to categories[M]. Cambridge: Cambridge University Press, 1997

[19] Gillibert J, Retoré C. Category theory, logic and formal linguistics: some connections, old and new[J]. Journal of Applied Logic, 2014, 12(1): 1-13

[20] Ehresmanna A C, Gomez-Ramirez J. Conciliating neuroscience and phenomenology via category theory[J]. Progress in Biophysics and Molecular Biology, 2015, 119(3): 347-359

[21] Zhu M, Grogono P, Ormandjieva O, *et al*. Using category theory and data flow analysis for modeling and verifying properties of communications in the process-oriented language Erasmus[C] //Proceedings of the International C* Conference on Computer Science & Software Engineering. New York: ACM Press, 2014: Article No.24

[22] England D. A category theory approach to HCI[C] //Proceedings of the 27th International BCS Human Computer Interaction Conference. Swinton: British Computer Society Press, 2013: Article No.55

[23] Beheshti R, Sukthankar G. Analyzing agent-based models using category theory[C] //Proceedings of the IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologies. Los Alamitos: IEEE Computer Society Press, 2013, 2: 280-286

[24] Pauly A, de Brecht M. Descriptive set theory in the category of represented spaces[C] //Proceedings of the 30th Annual ACM/IEEE Symposium on Logic in Computer Science. Los Alamitos: IEEE Computer Society Press, 2015: 438-449

[25] Harris D R. Modeling faceted browsing with category theory to support interoperability and reuse[C] //Proceedings of the 15th ACM/IEEE-CS Joint Conference on Digital Libraries. New York: ACM Press, 2015: 275-276

[26] Janeiro J, Barbosa S D J, Springer T, *et al*. Improving the search for user interface design patterns through typed relationships[M] //IFIP Advances in Information and Communication Technology. Heidelberg: Springer, 2010, 332: 3-14