面向特征的反射式实时应用系统领域模型研究*)

黄 靖 卢炎生 徐丽萍

(华中科技大学计算机科学与技术学院 武汉 430074)

摘 要 本文主要研究实时应用系统领域工程中领域模型规约描述的问题,即开展实时应用软件设计与开发的垂直复用研究。通过扩展 FODA(Feature-Oriented Domain Analysis)方法,引入实时特征作为该领域的需求规约单元,并从语义的角度来认识实时特征及其领域需求组织方式,同时应用反射技术来增强领域需求规约的动态性,最终归结为将实时特征作为实时应用需求空间的一阶实体来认识和组织实时应用系统领域模型,以寻求今后对实时应用系统领域语义一致性和可演化性的逻辑表达。

关键词 特征,领域模型,实时应用,反射

The Research of Feature-Oriented Reflective Real-time Application Domain Model

HUANG Jing LU Yan-Sheng XU Li-Ping

(School of Computer Science & Technology, Huazhong University of Science & Technology, Wuhan 430074)

Abstract The research of feature-oriented real-time domain model focuses on the relative problems on domain engineering in the project of real-time application system, that is, the research of the vertical reuse of real-time application software. By extending FODA (Feature-Oriented Domain Analysis) method and introducing real-time feature as the software requirement specification unit, the research come to see real-time feature and it's domain requirement architecture from an semantic perspective. Reflection technology is also adopted to enhance the dynamics of domain requirement specification. Finally, real-time feature is seen as entity of requirement space in organizing real-time application system domain model.

Keywords Feature, Domain model, Real-time application, Reflection

1 问题的提出

软件复用是一种解决"软件危机",提高软件开发效率和质量,实现软件产业工业化生产方式的重要手段与途径,其相关技术层出不穷。其中,领域工程研究了具有相似和相近软件需求的应用系统领域所具有的共同特征和变化特征,抽象得到领域分析模型作为同一领域新的应用系统开发的基础,为软件开发提供了垂直面的复用技术支持,是软件复用思想与理论的一个实践应用扩展。领域工程分为领域的分析、设计和实现 3 个阶段,领域模型则是其分析阶段的产物,它服务于面向复用的软件开发活动(development for reuse),是对整个领域知识整体性的概要认识,同时也是软件垂直复用(vertical reuse)的可复用资产。在软件复用活动中,它具有两个方面的重要作用:一是为领域内新系统的开发提供可复用的软件需求规约;二是能指导领域设计阶段和实现阶段可复用软件资产的生产。

当前,领域工程研究活动众多,但均以领域建模为研究基础,代表性的工作有 FODA[1]、FAST[2]、DSSA[3~5] 和青鸟工程[6]等。而单独就实时应用系统领域模型的研究尚不多见。领域建模的实质是获取领域内的需求规约,并作为软件垂直复用的可复用资源,其中 FODA(Feature-Oriented Domain Analysis)方法是目前领域模型研究中最为成熟的一种方法。作者对实时应用系统领域模型的研究,就是以它为主要研究蓝本。

FODA 方法作为一种领域分析建模方法,其核心思想是识别并规划某个领域内的需求特征(Feature)。然而面对实时应用系统领域,这种方法同大多其它领域分析方法一样,都存在着以下一些问题:

· 表达不完善。

以往领域模型的构建,着重对同一领域中功能需求刻面的表达,缺乏对领域中各个系统非功能性特征的整体性单独描述。例如,在对各实时应用系统的需求分析中,缺乏一个明确的对该领域所应具有的实时性能及相关性能需求的统一规约描述。

· 组织结构不明确。

现有的领域工程方法研究成果中,领域模型的组织框架和建模形式众多,各有相应的用途,且不够统一,缺乏一个专门针对复用目的、适用于领域模型库统一管理、检索和按需变更的组织形式。

· 缺乏对需求变化性的支持。

已有的领域需求分析方法,大致都属于静态建模方法,即领域模型一旦确定,不易更改,不宜领域需求可能的变化以及模型相应的演化,缺乏对领域模型统一修改操作的动态性支持。尽管领域具有一定的内聚性和稳定性,但领域需求的变动性还是可能和存在的,此时需要对领域分析结果——领域模型,进行相应的按需更改。

*)国防预研基金(10104010201)资助项目。**黄 靖** 博士研究生,主要研究方向为软件工程、分布式计算技术;**卢炎生** 教授,博士生导师,主要研究方向为软件工程、分布式计算技术、特种数据库等;**徐丽萍** 副教授,主要研究方向为软件工程、分布式计算技术等。

·规约中语义表达性不强。

以往领域模型中，对需求规约的语义表达支持不够强，其中语义表达单位不够明确，缺乏一定的形式逻辑推导能力。

针对这些问题，作者在实时应用系统领域模型的研究中，通过扩展 FODA 方法，引入实时特征作为该领域的需求规约单元，并从语义的角度来认识实时特征及其领域需求组织方式，同时应用反射技术来增强领域需求规约的动态性，最终归结为将实时特征作为实时应用需求空间的一阶实体来认识和组织实时应用系统领域模型，以寻求对实时应用系统领域语义一致性和可演化性的逻辑表达。

2 特征方法的引入与形式化规约

2.1 面向特征的由来

特征(Feature)概念起源于上世纪 90 年代初通信行业的“特征交互(Feature Interaction，简称 FI)”问题。随后逐渐融入到软件工程学科的发展中来，于 1992 年在 CMU，以 Kyo C. Kang 为首的研究小组提出了一份技术报告“Feature-Oriented Domain Analysis(FODA)-Feasible Study”，即面向特征的领域分析可行性报告中，首次将面向特征的方法引入到了领域工程中。它将特征作为一种“用户可见的领域侧面和特性”的需求规约单元，用于发现和描述相关软件系统的共性和变异，由具有共性的软件联合构成领域，从而在一个特定的领域里实现软件复用。至此，FODA 及面向特征的方法被公认为最实用的领域建模方法。在此基础上，目前又延伸出面向特征的软件复用方法(Feature-Oriented Reuse Method，FORM)[7]和特征工程(Feature Engineering)[8]等。

2.2 实时应用系统领域模型中特征的引入

面向特征的方法，帮助用户在软件需求规约层上封装了主题相关的内容，实现了“关注分离”(Separation of Concerns)[9]。这种关注分离的方法，是组织和分解软件系统的主要手段，使得软件成为较小规模的、更易管理或理解的片断，表达软件生命周期中不同开发者的角色区别和目标差异，其思想与软件设计阶段构件化的概念、目标及意图一致，因而有助于特征向构件的转换，以及软件需求分析与设计阶段的衔接。正是如此，在实时应用领域建模中引入了面向特征的方法。用特征建模的方法来形式化描述领域系统的需求规约语义，通过定义实时应用领域中的特征及特征间关系，反映领域相关软件所提供服务的存在本质，获取一种面向特征的领域体系结构，来作为领域内相关软件开发的垂直复用资源和参照标准。

我们针对实时应用领域自身的特点，对特征方法进行相应的扩充，将特征分为实时特征和非实时特征两类，分别表示实时应用领域模型中有时间约束特征的软件需求规约单元和无时间约束特征的软件需求规约单元。每个特征单元包含如下一些信息：特征标识符、特征类型、特征语义描述函数和特征父子关系函数。这样，在面向特征的实时应用系统领域模型中，不仅可以表达领域中系统的功能需求特征，还能有效地规约领域中系统的性能需求特征，从而能较全面地描述领域信息。

2.3 特征的形式化规约

特征的形式化规约，是一种实时应用系统领域的特征形式规格说明，用于描述领域模型的需求信息单元。我们利用特征形式规格说明的方法，把实时应用系统领域中共同的需求信息内容抽取出来，将不同的、有差异的细节信息形式化区分开来，作为领域分析、领域建模、领域设计的可复用单元，便于日后软件工程师们在领域工程中进行检索、修改等相关工作，支持软件的垂直复用。

这里，主要是用 Z 语言[10~13]作为书写规格说明的主体语言，来形式化规约特征单元。下面先给出预备定义 1、2、3，再根据 Z 语言是一种类型化语言的特点，给出实时应用系统领域中特征的形式化规约描述。

定义 1 特征标识符集合 IDENTIFIER，是一个由所有用于描述每个特征单元的标识性概念或术语的个体组成的集合。集合中的概念或术语个体相互区别，作为区分特征的实体。它是一个给定集合，在面向特征的实时应用系统领域模型 Z 语言表示中，表示为[IDENTIFIER]。

定义 2 特征类型集合 FEATURETYPE，是一个用于区分特征单元是否具有时间约束性的标识符号集，集合体为{rtfeature，none_rtfeature}。它是一个给定集合，在面向特征的实时应用系统领域模型 Z 语言表示中，表示为[FEATURETYPE]。

定义 3 特征语义描述集合 þCSP，是一个客观世界信息表述的知识集合，是由语义形式化语言 CSP[14]及其扩展语言 þCSP[15~18]所组成的幂集。它是一个给定集合，在面向特征的实时应用系统领域模型 Z 语言表示中，表示为[þCSP]。

那么，由以上 3 个给定集合作为输入，定义特征单元的 Z 语言形式化规约。

定义 4 特征的 Z 语言形式化规约描述为：

[IDENTIFIER, FEATURETYPE, þCSP]

Feature

feature_id : IDENTIFIER
feature_type : FEATURETYPE
requirement : IDENTIFIER↔þCSP
offather : IDENTIFIER→IDENTIFIER

∀ feature_id$_1$、feature_id$_2$: IDENTIFIER • requirement(feature_id$_1$)= requirement(feature_id$_2$)⇒feature_id$_1$=feature_id$_2$;
∀ feature_id$_1$、feature_id$_2$: **dom**(offather) • offather(feature_id$_1$) = offather(feature_id$_2$) ⇒ requirement(offather (feature_id$_1$))= requirement(offather (feature_id$_2$))

或 Feature≙[feature_id，feature_type：IDENTIFIER；requirement：IDENTIFIER↔þCSP；offather：IDENTIFIER→IDENTIFIER | (∀feature_id$_1$、feature_id$_2$：IDENTIFIER • requirement(feature_id$_1$) = requirement(feature_id$_2$) ⇒ feature_id$_1$ = feature_id$_2$) ∧ (∀feature_id$_1$、feature_id$_2$：dom(offather) • offather(feature_id$_1$) = offather(feature_id$_2$)⇒requirement(offather (feature_id$_1$)) = requirement(offather (feature_id$_2$)))]。

定义 4 中：

①requirement 函数描述的是特征标识符集合 IDENTIFIER 到特征语义描述集合 þCSP 的映射关系。当需求规约的语义描述一致时，特征标识符相同，即为同一个特征单元，这样能利于在实时领域模型库中检索相同或类似的需求规约，实现垂直复用。

②offather 函数给出的是子特征标识符集合 IDENTIFIER 到父特征标识符集合 IDENTIFIER 的映射关系。它是一个单射关系，每个子特征只有一个父特征，每个父特征可以对应多个子特征。由此，实时领域模型中所有特征(父子特征)组成了一个多叉描述树，树中各个节点反映了该领域模型中特征间的继承关系。子特征继承父特征所有需求信息，并具

体细化部分需求规约，这样能很好地从整体和细节上描述领域信息，使得子特征关注需求规约的模块化划分，父特征侧重领域整体的表述。

即，设 f、g：Feature，如果 f∈g，那么 ran(f. requirement)∈ran(g. requirement)，说明 f 是 g 的细节规约描述，g 涵盖了 f 的需求信息；如果 f=g，那么 ran(f. requirement)=ran(g. requirement)，说明 f、g 描述了相同的需求规约信息。

③该定义给出的是一个特征的形式化描述范型，既能规约实时特征，又能表述非实时特征，分为特征型和特征项。其中，特征项是特征型的实例，是具体实时应用系统领域模型中的组成实体。

基于前面的这些描述，可给出领域模型中特征描述树的形式化规约。

定义 5 特征描述树的 Z 语言形式化规约描述为：

[Feature]

FeatureTree

feature : Feature
treenode_rel : ℙFeature→Feature

∃f : Feature, ∃rel : treenode_rel • f∈ran(rel)⇒∃g, h : Feature • (g, h∈dom(rel))∧(g∪h=f)∧(g∪h∈ran(rel))

或 FeatureTree ≙ [feature : Feature; treenode_rel : ℙFeature→Feature | ∃f:Feature, ∃rel : treenode_rel • f∈ran(rel)⇒∃g,h :Feature • (g,h∈dom(rel))∧(g∪h=f)∧(g∪h∈ran(rel))]。

3 实时应用领域的特征项空间

在实时应用领域的特征模型中，特征项通常不会孤立存在，而是由多个领域中相关的特征项构建成一个多维的空间体系结构——特征项空间(Feature Item Space)，简称特征空间，它是实时应用领域建模的基础。从反射(Reflection)模型的角度来说，它又是实时应用领域模型的元层(Meta Level)。

3.1 实时特征模型基层中的相关概念

实时应用领域中，通过特征项空间构造特征模型的基层，涉及一些相关概念。为论述方便，下面先给出这些概念的定义。

定义 6 实时应用领域特征项空间 $\sum^{RT}$。实时应用领域特征项空间是由一组实时应用领域相关的特征项、特征项间的关系以及它们的配置约束所构成的特征项体系结构，表示为 $\sum^{RT}::=\langle$Feature, Relationship, Configuration$\rangle$。

定义 7 特征项子空间 ε^n。实时应用领域中的特征项子空间，是指在实时应用领域特征项空间 $\sum^{RT}$ 中存在某特征项，它能作为一种宿主空间结构，由一组领域相关的子特征项、子特征项间关系及配置约束所构成，表示为 $\varepsilon^n::=\langle$feature, relationship, configuration$\rangle$。

定义 8 特征项划分 F。实时应用领域特征项空间的特征项划分即为领域需求的一种描述划分，每个特征项对应一个描述区间。实时应用领域的特征项空间 $\sum^{RT}$ 和特征项子空间 ε^n，都存在着特征项划分，分别表示为 $F_\Sigma=\{f_1, f_2, \cdots, f_n\}$、$F_\varepsilon=\{f_1', f_2', \cdots, f_m'\}$($i$, n, m 为自然数，$i\leqslant n$)，其中 f_n 为特征项空间 $\sum^{RT}$ 中一特征项，f_m' 是 f_i 的特征项子空间 ε^n 中的子特征项。

定义 9 特征项划分的正交性。对于任意一特征项划分 $F=\{f_1, f_2, \cdots, f_n\}$，若存在 $f_i\cap f_j=\phi(i,j\leqslant n,\ i\neq j)$，则表示该划分满足特征项划分正交性。

值得注意的是，定义中 $f_i\cap f_j=\phi(i,j\leqslant n,\ i\neq j)$ 仅表示领域需求描述划分的无重叠性，并不说明特征项间无关联。

定义 10 特征项划分的完备性。设 Ω 表示某领域需求概念的整体，对于该领域一特征项划分 $F=\{f_1, f_2, \cdots, f_n\}$，若存在 $\Omega=f_1\cup f_2\cdots\cup f_n$，则表示该划分具备特征项划分完备性。

3.2 实时特征项空间的特性

由以上定义的描述，不难看出实时应用领域特征项空间具有如下一些特性：

(1)描述性。

实时应用领域的特征项空间，提供了一种有效表达实时应用领域中需求共性和差异的结构及描述方法，是实时应用领域建模的首要概念。这种描述体系具有一定数学基础，形式化地表达了领域需求，并能在软件工程过程的后阶段方便设计人员进行有效的逻辑推理与验证，为软件的自动生成提供了理论基础。如领域分析中获取的特征项，可对应于设计与实现阶段的构件；特征项间的依赖关系及配置约束，映射为软件体系结构。

(2)多维性。

实时应用领域的特征项空间是一个立体的多维领域需求表述空间，空间中每个特征项是一个维或基，特征项数即为空间的维数。对于一任意特征项空间 $\sum\in\sum^{RT}$，存在一种特征项划分 F，使得 $F=\{f_1, f_2, \cdots, f_n\}$，其中 $f_i(i\in 1,\cdots,n)$，n 分别是该划分下特征项空间 $\sum$ 的基和维数，且有 $f_i\cap f_j=\phi(i,j\in 1,\cdots,n)$ 和 $\Omega=f_1\cup f_2\cdots\cup f_n$。

(3)内含性。

特征项是实时应用领域特征项空间的一个基，也是特征项空间的一个领域需求表述单元，它同样可以细分为由一组相关子特征项、子特征项间依赖关系以及相应配置约束所构成的领域特征项子空间。设一特征项空间 $\sum\in\sum^{RT}$ 中某特征项 $f_i(i\in 1,\cdots,n)$，有 $f_i=\varepsilon(\varepsilon\in\varepsilon^n)$，其中 ε 满足特征项空间特征项划分的正交性、完备性，另外特征项自身是其特征项子空间的最大包容空间，即 ε 的一个特征项划分 $F_\varepsilon=\{f_i\}$。

(4)动态性。

实时应用领域的特征项空间具有一定的不确定性，其表现为特征项空间在横向、纵向两方面的动态性。

• 横向动态性

特征项空间中实时应用领域的特征项划分不唯一，存在着不确定性，在一定范围内，不同的领域分析人员可能会得到不同的正确的特征项划分。如 $F=\{f_1, f_2, \cdots, f_n\}$ 和 $F'=\{f_1'\ f_2', \cdots, f_m'\}$ 都是某一特征项空间 $\sum\in\sum^{RT}$ 的正确特征项划分，且满足特征项划分的正交性、完备性，但具有不同的基和维数。这通常是由于领域分析人员自身不同的领域经验所造成的，但若分析结果(特征项空间)一旦确定，是可以作为领域知识被后来的设计人员所复用的。

• 纵向动态性

在实时应用领域分析过程中，领域需求可能的变动会促使我们对前阶段领域分析的结果进行相应的调整和规划。这种调整与规划是在已有的特征项空间的基础上进行某些特征项或特征项间依赖关系或其配置约束的添加或删除，展现特征项空间的可扩展性，维系特征项空间纵向动态变化过程中的逻辑平衡性。

(5)反射性。

从实时应用领域模型计算反射的角度来看，实时应用领域特征项空间对应于该领域模型的基层。如前面所述，该领域特征项空间存在着动态性。考虑到这一点，我们在特征项空间之上构建了领域模型元层，能有效地根据需要对特征项空间进行相应的反射计算，保证了特征项空间中各组成因子在领域环境变化中的适应性，也为设计与实现阶段验证实时应用系统中相应的构件与体系结构提供了理论基础。

这些特性的体现，尤其是动态性、反射性，将在后面论述实时应用领域模型时给出。

3.3 实时特征空间的形式化规约

实时特征空间由特征空间元素组成，每个特征空间元素包含了3个方面的内容：特征单元的信息、特征间相互关联的信息、和特征间关系上附有的约束信息。其中，特征间相互关联的信息只是简单地表示两两特征之间关系的存在性；特征间关系上附有的约束信息则描述存在关系上的具体细节需求信息，比如特征单元间相互通信的接口信息、性能约束信息等，因而其语义信息较为丰富，在这里用CSP来进行规约描述。

特征空间元素有两个输入类型：Feature、℘CSP，分别用于定义空间元素里的特征单元和形式化描述特征间关系上附有的约束信息。

定义 11 特征空间元素 FeatureSpaceElement 的 Z 语言形式化规约描述为：

[Feature, ℘CSP]

FeatureSpaceElement

feature : Feature
feature_rel : Feature→Feature
feature_constraint : (Feature×Feature)→℘CSP

∀f, g : Feature • f≠g((feature_rel(f)≠feature_rel(g))∧
(feature_constraint(f)≠feature_constraint(g))
∃rel:feature_rel • f=**dom**(rel)∧**ran**(rel)⇒(f, g)=**dom**(feature_constraint)

或 FeatureSpaceElement≙[feature：Feature；feature_rel：Feature→Feature；feature constraint：(Feature×Feature)→℘CSP | (∀f, g：Feature • f≠g⇒(feature_rel(f)≠feature_rel(g))∧(feature constraint(f)≠feature constraint(g)))∧(∃rel：feature_rel • f=dom(rel)∧ran(rel)⇒(f, g)=dom(feature constraint))]。

定义中，谓词部分表达了每个特征单元拥有属于自己的、与其它特征的关系信息和关系的约束信息。

由特征空间元素 FeatureSpaceElement 的定义与描述，可得实时特征空间的表述：FeatureSpace≙℘featureSpaceElement，即实时特征空间是实时空间元素的幂集。通过这种表示方法，不仅能表述实时应用领域特征项空间 $\sum^{RT}$，还涵盖了对所有特征项子空间 ε^{rt} 的描述。

值得区分的是，实时特征空间与前一节中给出的特征描述树具有不同的规约意义：特征描述树规约的是特征间的继承关系，反映了领域工程中需求信息的细化过程；而实时特征空间则描述的是特征间的组合关系，专注领域需求信息的模块划分。两者在实时应用系统领域建模中相互补充，相得益彰。

4 反射式特征与反射式特征空间

面向特征的建模方法经过扩展，引入到实时应用系统领域中后，能够很好地组织和描述实时应用领域内共有的需求信息。然而，领域知识并非是一成不变的，或者不同的领域却有着某些相似的领域知识，这时就要求特征及特征空间能够支持领域需求信息的变化性，更好服务于领域知识体的垂直复用。这里，作者在特征及特征空间中引用反射技术，实现特征及特征空间可能需要的动态调整，增强实时应用系统领域中面向特征建模方法的灵活性和高效性。

4.1 实时应用系统领域中的反射式特征

支持能动态调整的反射式特征，分为前状态和后状态，即特征调整前的模式和调整后的模式。

定义 12 特征前状态 Feature，即特征调整前的模式，用 Z 语言形式化规约描述为：

[IDENTIFIER, FEATURETYPE, ℘CSP]

Feature

feature_id : IDENTIFIER
feature_type : FEATURETYPE
requirement : IDENTIFIER↔℘CSP
offather : IDENTIFIER→IDENTIFIER

∀ feature_id$_1$、feature_id$_2$: IDENTIFIER • requirement(feature_id$_1$)=
requirement(feature_id2)⇒feature_id$_1$=feature_id$_2$;
∀ feature_id$_1$、feature_id$_2$: **dom**(offather) • offather(feature_id$_1$)=
offather(feature_id$_2$)⇒requirement(offather (feature_id$_1$))=
requirement(offather (feature_id$_2$))

或 Feature≙[feature id, feature_type：IDENTIFIER；requirement：IDENTIFIER↔℘CSP；offather：IDENTIFIER→IDENTIFIER | (∀feature_id$_1$、feature id$_2$：IDENTIFIER • requirement(feature id$_1$)=requirement(feature id$_2$)⇒feature id$_1$=feature id$_2$)∧(∀feature_id$_1$、feature_id$_2$：dom(offather) • offather(feature_id$_1$)=offather(feature_id$_2$)⇒requirement(offather (feature_id$_1$))=requirement(offather (feature_id$_2$)))]。

定义 13 特征后状态 Feature′，即特征调整后的模式，用 Z 语言形式化规约描述为：

[IDENTIFIER, FEATURETYPE, ℘CSP]

Feature′

feature_id′ : IDENTIFIER
feature_type′ : FEATURETYPE
requirement′ : IDENTIFIER↔℘CSP
offather′ : IDENTIFIER→IDENTIFIER

∀ feature_id$_1$′、feature_id$_2$′ : IDENTIFIER • requirement′ (feature_id$_1$′)=
requirement′ (feature_id$_2$′)⇒feature_id$_1$′ =feature_id$_2$′ ;
∀ feature_id$_1$′、feature_id$_2$′ : **dom**(offather′) • offather′ (feature_id$_1$′)=
offather′ (feature_id$_2$′)⇒requirement′ (offather′ (feature_id$_1$′))=
requirement′ (offather′ (feature_id$_2$′))

或 Feature′≙[feature_id′, feature_type′：IDENTIFIER；requirement′：IDENTIFIER↔℘CSP；offather′：IDENTIFIER→IDENTIFIER | (∀feature_id$_1$′、feature_id$_2$′：IDENTIFIER • requirement′(feature_id$_1$′)=requirement′(feature id$_2$′)⇒feature id$_1$′=feature id$_2$′)∧(∀feature id$_1$′、feature_id$_2$′：dom(offather′) • offather′(feature_id$_1$′)=offather′(feature_id$_2$′)⇒requirement′(offather′ (feature_id$_1$′))=requirement′(offather′ (feature_id$_2$′)))]。

由两者的定义可以看出，它们的区别在于特征后状态在特征前状态的基础上加了一个模式修饰符′。而这两个状态，也正是反射式特征中最重要的两个转换模式。

在特征单元上，可以进行的反射操作主要有3种：查找所需特征、修改特征和删除特征。每种操作的功能用途不一，但都是为了支持特征的动态调整。

定义 14 反射式特征的查找操作 Lookup，其 Z 语言形

式化规约描述为：

[IDENTIFIER, FEATURETYPE, ÞCSP]

Lookup

Ξ Feature
feature_id?, feature_id! : IDENTIFIER
feature_type?, feature_type! : FEATURETYPE
requirement?, requirement! : IDENTIFIER↔ÞCSP
offather?, offather! : IDENTIFIER→IDENTIFIER

---

feature_id?=feature_id!
feature_type?=feature_type!
requirement?=requirement!
offather?=offather!

由定义可以看出，Lookup 操作后，特征的前后状态一致，没有变化，因而用 Z 语言规范中 Ξ 符号给出标识。

定义 15 反射式特征的修改操作 Update，其 Z 语言形式化规约描述为：

[IDENTIFIER, FEATURETYPE, ÞCSP]

Update

Δ Feature
feature_id?, feature_id! : IDENTIFIER
feature_type?, feature_type! : FEATURETYPE
requirement?, requirement! : IDENTIFIER↔ÞCSP
offather?, offather! : IDENTIFIER→IDENTIFIER

---

feature_id?≠feature_id!
feature_type?≠feature_type!
requirement?≠requirement!
offather?≠offather!

由定义可以看出，Update 操作后，特征的前后状态改变，因而用 Z 语言规范中符号 Δ 给出标识。

定义 16 反射式特征的删除操作 Delete，其 Z 语言形式化规约描述为：

[IDENTIFIER, FEATURETYPE, ÞCSP]

Delete

Δ Feature
feature_id? : IDENTIFIER
feature_type? : FEATURETYPE
requirement? : IDENTIFIER↔ÞCSP
offather? : IDENTIFIER→IDENTIFIER

---

feature_id′ =Feature\feature_id?
feature_type′ =Feature\feature_type?
requirement′ =Feature\requirement?
offather′ =Feature\offather?

由定义可以看出，Delete 操作后，特征的前后状态同样发生了改变，因而用 Z 语言规范中符号 Δ 给出标识。

由此，定义 12、13 描述了反射式特征的元数据，定义 14、15、16 则给出了反射式特征的元对象协议(MOP)。那么，反射式特征表示为：$Feature_{reflective}$ ::=〈Feature, Feature′, Lookup, Update, Delete〉。

4.2 实时应用系统领域中的反射式特征空间

由 3.3 节可知，特征空间元素 FeatureSpaceElement 是特征空间的基本组成体，因而在反射式特征空间的描述中，首先应当给出反射式特征空间元素的描述。

同反射式特征一样，为支持能动态调整，反射式特征空间元素也分为前状态 FeatureSpaceElement 和后状态 FeatureSpaceElement′，即特征空间元素调整前的模式和调整后的模式。组合之后的反射式特征空间也是如此，它们作为反射机理中的元数据，都支持查找、修改和删除 3 种反射操作。关于反射式特征空间元素及反射式特征空间一系列的定义和概念类似于反射式特征的 Z 语言形式化描述，这里不再重复。反射式特征空间表示为：$FeatureSpace_{reflective}$ ::=〈FeatureSpace, FeatureSpace′, Lookup, Update, Delete〉。

5 面向特征的反射式实时应用系统领域模型

这样，可由前几节中的定义，建立面向特征的反射式实时应用系统领域模型。该模型表示为：$rtFODM_{reflective}$ ::=〈Feature, FeatureTree, FeatureSpace, $Feature_{reflective}$, $FeatureSpace_{reflective}$〉。模型中每个部分由前面相应的描述体具体表述。

结束语 领域模型研究是软件垂直复用研究的重要组成部分，当前专注于实时应用系统领域的这方面研究尚不多见。实时应用系统领域具有许多区别于其它应用领域的自身特性，因而其建模的方法也应有所不同。本文通过扩展 FODA 方法，引入实时特征作为该领域的需求规约单元，并形式化规约实时特征及其领域需求的各类组织方式，同时应用反射技术来增强领域需求规约的动态性，最后依据这些研究成果，给出面向特征的反射式实时应用系统领域模型，逐步描述了本文开头之处 4 个问题的解决方案。在今后的工作中，我们将以此研究为基础，进一步寻求、强化对实时应用系统领域语义一致性和可演化性的逻辑表达。

参 考 文 献

1 Kang K C, Cohen S G, Hess J A, et al. Feature-oriented domain analysis(FODA) feasibility study : [Technical Report]. Pittsburgh : Carnegie Mellon University, Software Engineering Institute, 1990
2 Weiss D M. Family-oriented Abstraction, Specification, and Translation, The FAST Process. Keynote talk at Computer Assurance Conference(COMPASS), June, 1996
3 Tracz W, Coglianese L. DSSA Engineering Process Guidelines : [Technical Report]. IBM Federal Systems Company, December, 1992
4 Tracz W. DSSA(Domain-Specific Software Architecture): pedagogical example. ACM SIGSOFT Software Engineering Notes, 1995, 20(3)
5 Tracz W. Domain-Specific Software Architecture (DSSA) frequently asked questions(FAQ). ACM SIGSOFT Software Engineering Notes, 1994, 19(2)
6 常继传，梅宏，杨芙清. 青鸟系统中可复用软件构件的表示与查询. 电子学报，2000，28(8)：20～23
7 Kang K C, Kim S, Lee J, et al. FORM: A feature-oriented reuse method with domain-specific reference architectures. Annals of Software Engineering, 1998, 5: 143～168
8 Keck D O, Kuehn P J. The feature and service interaction problem in telecommunication software systems: a survey. IEEE Transaction on Software Engineering, 1998, 24(10): 779～796
9 Tarr P, Ossher H, Harrison W, et al. N Degrees of Separation: Multi-Dimensional Separation of Concerns. Proceedings of the International Conference on Software Engineering(ICSE'99), May 1999
10 Diller A Z. An Introduction to Formal Method. London: John Wiley & Sons, 1990
11 缪淮扣. Z 规格说明中的前置条件的简化. 软件学报，1997，8(9)：709～715
12 Spivey J M. Understanding Z, A Specification Language and its Formal Semantics. Cambridge: Cambridge University Press, 1998
13 Potter B, Sinclair J, Till D. An Introduction to Formal Specification and Z. London: Prentice Hall, 1992
14 Hoare C A R. Communicating Sequential Processing. London : Prentice Hall, 1985
15 Schneider S. An Operational Semantics for Timed CSP. Oxford University Computing Laboratory : Information and Computation Programming Research Group
16 Lowe G. Prioritized and Probabilistic Models of Timed CSP. Oxford University Computing Laboratory : Information and Computation Programming Research Group, 1991
17 Lowe G. Specification and Proof of Prioritized, Timed CSP Processes. Oxford University Computing Laboratory : Information and Computation Programming Research Group, 1992
18 Davies J, Schneider S. A Brief History of Timed CSP. Oxford University Computing Laboratory : Information and Computation Programming Research Group, 1992