ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.1, January 2006, pp.39−47
DOI: 10.1360/jos170039

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于特征的构件模型及其规范化设计过程*

王忠杰+, 徐晓飞, 战德臣

(哈尔滨工业大学 计算机科学与技术学院,黑龙江 哈尔滨 150001)

Feature-Based Component Model and Normalized Design Process

WANG Zhong-Jie+, XU Xiao-Fei, ZHAN De-Chen

(School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China)

+ Corresponding author: Phn: +86-451-86414906, Fax: +86-451-86412664, E-mail: rainy@hit.edu.cn, http://www.cs.hit.edu.cn

Wang ZJ, Xu XF, Zhan DC. Feature-Based component model and normalized design process. *Journal of Software*, 2006,17(1):39−47. http://www.jos.org.cn/1000-9825/17/39.htm

Abstract: Component-Based development method is thought to be an effective technique to tackle software crisis, but in practice it didn't reach the expectation, and currently there lack of normal forms and normalized methods to support identification and design of components with high reusability. This paper tries to solve this problem with feature space as a tool. Theory of feature and feature space is firstly introduced, and by analysis of dependencies between features' variability, the concept of feature dependency (FD) and four types of FDs are elaborated. Then a component specification model based on feature space is presented, in which component reusability is expressed by feature's 'type-value' variation mechanism and feature dependencies. After that, goals of component design and several reusability metrics are briefly discussed, and four component normal forms and the corresponding normalization algorithms based on feature space decomposition are presented in detail. A practical case is finally shown to validate the methods. The normal forms and normalized design methods realize the multi-grained and multi-form components' co-existence and the loosely composition-based coupling between components, which result in higher reusability, higher reuse efficiency, and lower reuse cost. The methods have been widely applied in the design and implementation of component-based Enterprise Resource Planning (ERP) systems, and have shown great theoretical and practical significance to component design.

Key words: component; feature space; feature dependency; reusability; normal form; component design

摘 要: 基于构件的软件复用是解决软件危机的重要手段,但目前还缺乏规范化的模式和方法以支持具有高复用性能的构件的识别与设计,借助特征空间作为工具以解决上述问题.首先介绍特征与特征空间的概念,从特征变化的相互依存关系入手,提出特征依赖的概念和 4 种具体的特征依赖.在此基础上,给出了基于特征空间的构件模型,使用特征的"型-值"机制与特征依赖表达构件的复用性.然后讨论了构件复用度的度量手段和规范化设计的目标,提出 4 种构件规范化模式(原子模式、基本模式、框架模式和内聚模式),研究了以特征空间分解为

* Supported by the National Natural Science Foundation of China under Grant No.60573086 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2003AA4Z3210 (国家高技术研究发展计划(863)); the National Research Foundation for the Doctoral Program of Higher Education of China under Grant No.20030213027 (国家教育部博士点基金)

Received 2004-12-11; Accepted 2005-06-02

基础的规范化方法,并通过实例加以验证.该方法实现了多粒度、多模式构件的共存和构件间基于组合的松散耦合,从而提高了构件的复用效率并降低复用成本.结果在企业资源计划(enterprice resource planning,简称 ERP)系统的构件化设计与开发中得到广泛应用,对指导构件设计具有较高的理论与实践价值.

关键词: 构件;特征空间;特征依赖;复用度;规范化模式;构件设计
中图法分类号: TP311 文献标识码: A

基于构件的软件复用[1]被认为是解决软件危机的重要手段.构件能够进行复用,必须具备两方面的因素:一个恰当的可以描述复用机制的构件模型,一种能够指导设计人员进行构件识别与设计的方法.构件模型以形式化方式描述了构件的结构、语义和非功能特性,是分析和评价构件行为和性能的依据[2],目前文献中能够进行精确语义定义的构件模型包括 3C 模型[3]、CBD/EC 模型[2]等.但它们注重从构件存储、检索、适配和组装等操作进行语义描述,而对构件的复用机制——如何表达可变语义以使构件具备较高的适应性——则缺乏深入刻画.如果构件只能表达确定的语义,那么它就无法通过语义的可变性来调整自身行为以适应不同的业务需求,其存在的意义也大为降低.

构件设计方法目前的相关研究成果可以归为两类:领域工程方法和基于聚合度-耦合度的方法.领域工程从特定领域一组相似的需求出发进行领域分析,确定通用性与可变性,构造特定领域体系结构,寻求可复用的业务语义并构造构件,追求构件复用度的提高,但忽视了复用成本和效率等因素.代表性的方法有面向特征的领域分析(FODA)[4]等.基于聚合度-耦合度的方法从考虑复用成本出发计算业务元素间全局相关性,按高聚合-低耦合原则对业务模型进行划分,并将子模型封装为构件,代表性的方法有 CRWD 矩阵法[5]和聚类分析法[6,7].这类方法得到的构件松散耦合,保证了较低的复用成本,但由于只是针对特定的业务模型进行划分,基本上没有考虑构件的复用性.

综合而言,构件设计尚缺乏严格的理论指导原则.目前,构件设计原则大都来源于面向对象方法中类的设计原则[8],如一般复用原则(CRP)、无圈依赖原则(ADP)等.但构件并不是类的简单聚集,这些原则并不完全适合于构件的设计.构件设计在很大程度上仍需借助设计人员的经验.

不良的构件设计会抵消复用所带来的优点,为此需要能够严格表达可复用语义的构件模型和规范化模式.给出一种基于特征空间的构件模型,在借鉴 FODA[4]和 CBD/EC 模型[2]中使用特征表达业务语义的基础上,扩展了传统特征模型中的特征可变类型,提出特征依赖的概念,将业务元素之间单纯的语义依赖扩展为特征依赖,进而建立构件规范化模式,并借鉴基于聚合度-耦合度的设计方法,根据特征依赖对特征空间进行分解得到规范化构件,以改善构件设计质量,提高复用性能.

1 特征与特征依赖

1.1 特征与特征空间

定义**1**(特征). 特征是描述客观世界知识的本体,表现为描述特定应用领域所提供服务的术语或概念.

定义**2**(特征空间). 由一组特定领域相关的特征及其之间的特征依赖关系构成的语义空间,称为特征空间,表示为$\Omega=\langle F,D\rangle$,$F$ 为特征集合,D 为特征之间依赖关系的集合.

定义**3**(特征树). 特征空间Ω通常表示为特征树的形式,树中有且仅有一个根节点 f_{root} 表示Ω的根特征,树中直接相连的父子节点分别表示父特征和子特征.$child(f)$,$parent(f)$,$ancestor(f)$和 $descendant(f)$分别表示 f 的子特征、父特征、祖先特征和后代特征的集合.特征树中叶子节点称为原子特征,非叶子节点称为复合特征.父子特征是一种聚合关系,描述特征之间的“整体-部分”关系.

定义**4**(特征项). 特征项是特征的实例,表示特征在不同环境下可能的取值,反映了特征自身的变化性.$dom(f)$为 f 所有特征项的集合,称为 f 的值域.特征与特征项是一种例化关系,描述特征与特征项之间的“一般-特殊”关系.

f 的实例化是指从 $dom(f)$中选取满足特定需求语境的特征项的过程,记为 $\tau(f)$.特征向量 $Y=(f_1,f_2,\ldots,f_n)$的实例化可表示为 $\tau(Y)=(\tau(f_1),\tau(f_2),\ldots,\tau(f_n))$.对特征空间$\Omega$中包含的每一项特征 $f_1,f_2,\ldots,f_n$ 分别进行实例化,可以得到特征空间的实例集合,记为 $T(\Omega)$.$t\in T(\Omega)$称为Ω的一个实例,表示为 $t=(\tau_1,\tau_2,\ldots,\tau_n)$,$\tau_i\in dom(f_i)$称为 t 在特征 f_i 上的投影,可记为 $t[f_i]$.t 在特征集合 X 上的投影记为 $t[X]$.特征的特征项数目表征了特征本身的复用性,特征空间的实例数目则表征了特征空间可被复用的潜力.

图 1 给出了特征空间的一个实例,根特征提供“零件生产计划编制过程”服务.

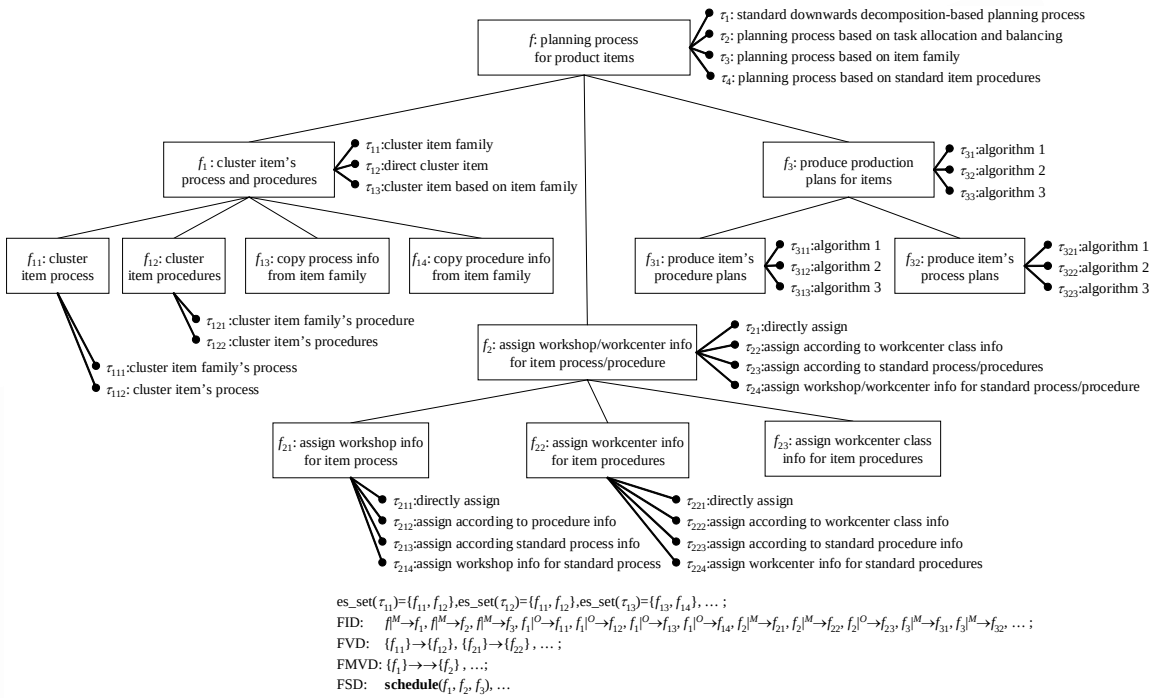

Fig.1 An example of feature space model

图1 特征空间示例

1.2 特征依赖

定义**5**(必要子特征集合). 特征 f 的实例可以通过其子特征集合的实例表示.设 f 存在子特征集合 $child(f)$,对$\forall\tau\in dom(f)$,必然$\exists Y\subseteq child(f)$,$Y=\{f_1,f_2,\ldots,f_n\}$,对$\forall f_i\in Y$,至少存在一个特征项 $\tau_i\in dom(f_i)$,使得 τ可由 $\tau_1,\tau_2,\ldots,\tau_n$ 唯一确定,则称 Y 是特征实例 τ的必要子特征集合,记为 es_set(τ).

定义**6**(**FID**). 特征 f 与特征集合 Y 之间存在特征完整性依赖,当且仅当 $Y\subseteq child(f)$,且对于 f 的每一个特征项 τ,唯一确定了 Y 的一个子集 Y',使得 Y'为 τ必要子特征集合 es_set(τ)的子集,即 $Y'\subseteq$es_set(τ).记为 $f|\rightarrow Y$.

FID 实际上定义了子特征相对于父特征的可选性约束.对 $f|\rightarrow Y$,可细分为必选特征依赖、可选特征依赖、单选特征依赖和复选特征依赖 4 种类型,分别表示为 $f|^M\rightarrow g$, $f|^O\rightarrow g$, $f|^S\rightarrow Y$, $f|^T\rightarrow Y$.传统特征建模中将这 4 类特征依赖称为特征的可选性[2],描述了子特征相对于父特征的可选性,但没有明确将父特征的特征项与子特征的可选性关联在一起.定义 6 对此进行了改进.

FID 实际上可看作父特征的“值”与子特征的“型”之间的依赖关系,称为“值-型”依赖,父特征的一个特征项决定了哪些子特征是构成该特征项的必要子特征.

定义**7**(**FVD**). 特征集合 X 与 Y 是特征空间Ω中特征集合 F 的子集.对Ω的实例化空间 $T(\Omega)$中的任意两个实例 t_1 和 t_2,如果 $t_1[X]=t_2[X]$,则 $t_1[Y]=t_2[Y]$,则称 Y 特征取值依赖于 X,记为 $X\rightarrow Y$. $X\rightarrow Y$ 表示 X 的一个实例唯一地确定了 Y 的一个实例.

定义**8**(**FMVD**). 特征集合 X,Y 与 Z 是特征空间 Ω 中特征集合 F 的子集,并且 $Z=F-X-Y$.特征多值依赖 $X\rightarrow\rightarrow Y$ 成立,当且仅当对 Ω 的实例化空间 $T(\Omega)$ 中的任意实例 t,t 在 (X,Z) 上的每一个实例值对应一组 Y 的值,这组值仅取决于 X 值而与 Z 值无关.

FVD/FMVD刻画了两组特征的特征项之间的依赖关系,称为“值-值”依赖,一组特征的实例取值唯一或多值决定了另一组特征的实例取值.

定义**9**(**FSD**). 特征语义依赖是指特征之间存在的语义关联关系.根据特征类型的不同,特征语义依赖可能有多种类型,如业务操作特征之间的时序约束,业务对象特征之间的关联、继承、组合等关系,业务活动特征之间的“事件-条件-行为(ECA)”约束等.一般地,使用谓词 $\boldsymbol{P}(X)$ 表示特征集合 X 中包含的特征之间要满足 $\boldsymbol{P}$ 的约束.$\boldsymbol{P}$ 的具体表达形式可视具体的约束类型而定.

FSD 与特征本身的取值无关,它刻画了特征本身“型”之间的约束,属于“型-型”依赖.

2 基于特征的构件模型

2.1 构件特征模型

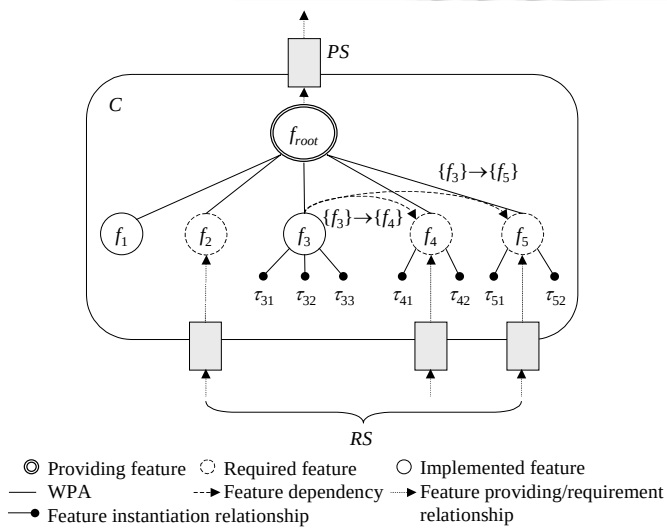

Fig.2 An example of feature-based component

图 2 基于特征的构件模型

定义**10**(构件特征模型). 一个构件定义了特定业务领域的一个特征子空间.构件特征模型可表示为 $C\langle cid,f_{root},F,D,PS,RS\rangle$,其中 cid 为 C 的唯一标识;f_{root} 为构件特征空间的根特征,是构件所能够提供的最大粒度的特征;F 为构件特征空间包含的特征集合;D 为构件特征之间依赖关系集合;PS 为构件所能提供的特征集合;RS 为构件所需求的特征集合.易知 $F-RS$ 为构件 C 自身实现的特征集合.

构件特征模型还应满足以下约束条件:(1) $F\subseteq descendant(f_{root})$;(2) 对 $\forall f\in F-RS$,如果 $child(f)\neq\varnothing$,则对 $\forall g\in child(f)$,要么 $g\in F-RS$,要么 $g\in RS$;(3) 对 $\forall f\in RS$,必有 $parent(f)\notin RS$;(4) $PS\neq\varnothing$.

图 2 给出了构件特征模型的一个示例.

2.2 构件交互模式

不同构件通过交互实现完整的特征空间.构件之间存在的关系由各自包含的特征之间的特征依赖决定,可分为 3 类:组合、依赖与关联.组合关系由特征完整性依赖产生的“特征需求-特征提供”关系所导致,依赖关系由特征取值/多值依赖所导致,而关联关系则由特征语义依赖所导致.

设 $C_1\langle f_{root},F,R,PS,RS\rangle$ 与 $C_2\langle f_{root},F,R,PS,RS\rangle$ 属于同一个特征空间 Ω,

(1) 如果 $\exists f_1\in PS(C_1),\exists f_2\in RS(C_2)$,且 $f_1=f_2$,则 C_1 与 C_2 之间存在组合关系,表示为 $C_1\xrightarrow{f_1=f_2}C_2$;

(2) 如果 $\exists X\subseteq F(C_1),\exists Y\subseteq F(C_2)$,使得 $X\rightarrow Y$ 或 $X\rightarrow\rightarrow Y$ 在 Ω 中成立,则 C_1 与 C_2 之间存在依赖关系,表示为 $C_1\xrightarrow{X\rightarrow Y}C_2$ 或者 $C_1\xrightarrow{X\rightarrow\rightarrow Y}C_2$;

(3) 如果 $\exists X\subseteq F(C_1),\exists Y\subseteq F(C_2)$,使得 $\mathbf{P}(X\cup Y)$ 是 Ω 中的一个特征语义依赖,则表明 C_1 与 C_2 之间存在语义关联关系,表示为 $C_1\xleftarrow{\mathbf{P}(X\cup Y)}C_2$.

2.3 构件复用性能的度量

定义**11**(特征粒度). 特征 f 的粒度定义为特征所描述的服务的大小,分为相对粒度 $FG_R(f)$和绝对粒度 $FG_A(f)$.$FG_R(f)$用 f 的子特征的数目加以度量,$FG_R(f)=|child(f)|$;$FG_A(f)$则是用 f 的所有后代中原子特征的数目加以描述, $FG_A(f)=|\{g|g\in descendent(f)\wedge child(g)=\varnothing\}|$.

定义**12**(构件粒度). 构件粒度是衡量构件大小和复杂度的指标.与特征粒度相似,也分为相对粒度 $CG_R(C)$和绝对粒度 $CG_A(C)$.$CG_R(C)$定义为 C 中包含的特征的数目,$CG_R(C)=|F(C)|$;$CG_A(C)$则刻画了 C 的特征子空间在整个特征空间中所处的位置,可用根特征的绝对粒度来度量,$CG_A(C)=FG_A(f_{root})$.

定义**13**(特征复用频度). 特征 f 的复用频度 $RF(f,\Omega)$定义了 f 在Ω内可能被复用的机率,分为绝对复用频度 $RF_A(f,\Omega)$和相对复用频度 $RF_R(f,\Omega)$.$RF_A(f,\Omega)$定义为特征空间的所有实例 $T(\Omega)$中包含特征 f 的实例的数目 $|T(f,\Omega)|$.对$\forall t\in T(\Omega)$,如果 $t[f]\neq\varnothing$,则 $t\in T(f,\Omega)$.$RF_R(f,\Omega)$定义为特征空间的所有实例 $T(\Omega)$中包含特征 f 的实例的比率:$RF_R(f,\Omega)=\left|T(f,\Omega)\right| \Big/ \left|T(\Omega)\right|$.

定义**14**(构件复用度). 构件复用度 $CR(C)$定义为构件能够被复用的机会的大小,分为绝对复用度 $CR_A(C)$和相对复用度 $CR_R(C)$.$CR_A(C)$可以通过 C 的特征空间所能实现的实例的总数目 $T(C)$来度量,表示为 $CR_A(C)=|T(C)|$. $CR_A(C)$越大,表明 C 所能够适应的业务需求的变化就越灵活,就具有越大的复用度.

$CR_R(C)$描述了 C 在特征空间Ω中可被复用的机会的大小,可通过 C 中每一个提供特征 f 在Ω内的相对复用频度 $RF_R(f,\Omega)$的平均值来度量,即 $CR_R(C)=\sum_{f\in PS(C)} RF_R(f,\Omega) \Big/ \left|PS(C)\right|$.

定义**15**(构件复用成本). 构件复用成本 $CRC(C)$定义为复用一次 C 所耗费的成本,可分为对构件进行实例化的成本 $CI(C)$、与其他构件组合的成本 $CC(C)$两部分,即 $CRC(C)=CI(C)+CC(C)$.

构件实例化的过程为:依据特征完整性依赖,选定各特征的必要子特征集合;依据特征取值依赖,为选定的每一个特征进行特征项选取.因此,实例化成本可定义为 $CI(C)=C_D\times|D(C)|+\sum_{f\in F(C)}(RF_R(f,C)\times C_F)$,$C_D$,$C_F$ 为处理一项特征依赖的成本、对一项特征进行实例化的成本.$RF_R(f,C)$表示了 f 被选中的概率.构件组合是指在构件需求特征与其他构件相应的提供特征之间建立接口连接的过程,可通过 $CC(C)=C_B\times|RS(C)|$计算,C_B 为接口连接的单位成本.

定义**16**(构件复用效率). 构件复用效率 $CRE(C)$定义为使用该构件构造特定业务的特征空间时所能直接提供的子特征空间的大小.子空间越大,该构件参与建模的效率就越高,因此可用构件本身的绝对粒度来度量复用效率,即 $CRE(C)=CG_A(C)$.

2.4 与其他构件模型的比较

与传统构件模型相比,基于特征的构件模型具有以下优点:

(1) 面向复用的设计

特征模型的优点就是“型-值”机制所导致的特征可复用性,再加上特征空间本身就是对同一领域相似业务模型的抽象,从特征空间映射得到的构件也具有较高的复用性.传统构件通常是针对特定的业务模型进行划分得到的[5−7],仅满足特定业务需求,无法适应变化,复用性能难以保证.

(2) 多粒度构件共存

由于特征空间能够表达不同粒度的特征并通过父子关系组织为特征树的形式,经过映射,不同粒度的构件可同时存在于构件空间.小粒度构件具有较高的复用度,大粒度构件则具有较低的复用成本和较高的复用效率.在构造应用系统时,可以灵活选择合适粒度的构件进行复用.传统构件模型则缺乏对多粒度的支持,粒度基本相似,相当于特征空间中处于同一层次的特征(如文献[6]的 business activity 层、文献[5,7]的 use case 层),构件复用性能难以进行灵活调整.

(3) 基于组合的松散耦合

在传统构件模型中,构件间的唯一关系为语义关联(如文献[5]中 use case 间的〈include〉,〈extend〉等),并将接

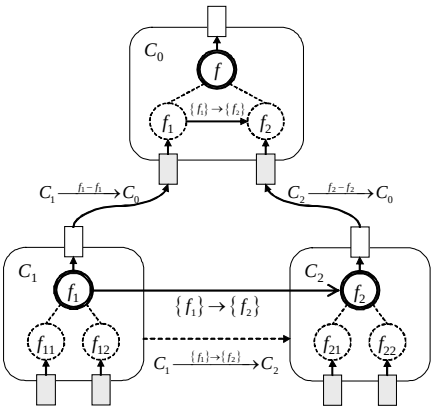

Fig.3 Composition-Based loosely coupling between components

图 3 基于组合的松散耦合

口调用作为构件交互的唯一手段.而接口调用是导致构件耦合的主要因素.虽然文献[5−7]也给出了方法以降低耦合度,但仍然难以适应业务的快速变化.基于特征的构件模型通过 4 种特征依赖描述特征间关系,并据此衍生出构件间 3 种交互关系,较完备地描述了构件间结构和语义约束.其中,组合关系属于显式结构关系,存在于大粒度-小粒度构件之间,通过接口调用完成;存在于同种粒度构件之间的依赖和关联关系属于隐式关系,不需要通过接口调用完成.本文采用的机制是将这些隐式关系通过显式的组合关系上升到更高一层的大粒度构件内部来完成,构件间依赖和关联可转化上层大粒度构件内部特征间的特征取值依赖或语义依赖,从而降低了小粒度构件间的耦合.

从图 3 给出的示例可以看出,C_1 的特征 f_1 与 C_2 的特征 f_2 之间的特征取值依赖 $\{f_1\}\to\{f_2\}$ 通过组合关系 $C_1 \xrightarrow{f_1-f_1} C_0$ 和 $C_2 \xrightarrow{f_2-f_2} C_0$ 转化为 C_0 内部的特征取值依赖,从而消除了 $\{f_1\}\to\{f_2\}$ 所导致的 C_1 与 C_2 之间的耦合.

3 构件规范化模式与设计方法

3.1 构件规范化模式

定义**17**(原子构件模式). 构件 $C\langle cid,f_{root},F,D,PS,RS\rangle$ 称为原子构件,当且仅当 $|F|=1,D=\varnothing,F=\{f_{root}\},PS=\{f_{root}\},RS=\varnothing$.

原子构件具有最小粒度,实现特征空间内的原子特征.由于不包含任何特征约束,可最大程度地复用特征的每一个特征项,从而使绝对复用度达到最大.但由于粒度最小,复用效率也最低.

定义**18**(基本构件模式). 构件 $C\langle cid,f_{root},F,D,PS,RS\rangle$ 称为基本构件,当且仅当 F 仅由 f_{root} 及其子特征集合 $child(f_{root})$ 构成,且 $RS=child(f_{root})$.

基本构件实现特征空间内的非原子特征,而原子构件也可看作一类特殊的基本构件($RS=\varnothing$),二者本质上是一致的,都是将一个特征直接映射为一个独立构件.基本构件的粒度比原子构件的粒度要大,因此其复用效率有所提高.但由于包含了较多的需求特征,复用成本也随之提高.

定义**19**(框架构件模式). 基本构件 $C\langle cid,f_{root},F,D,PS,RS\rangle$ 称为框架构件,当且仅当对 $\forall f\in F-RS, RF_R(f,C)=1$,对 $\forall f\in RS, RF_R(f,C)<1$.

框架构件在基本构件的基础上,满足了“共同复用”原则[8],将具有相同祖先且必然被共同复用的特征聚合在一起,而与这些固定特征相关的可变特征则封装在其他构件中,这与传统的面向对象中“框架”的概念类似.

“共同复用”原则使得框架构件的相对复用度达到最大.另外,由于固定特征无须实例化,复用成本达到最小.

定义**20**(内聚构件模式). 基本构件 $C\langle cid,f_{root},F,D,PS,RS\rangle$ 称为内聚构件,当且仅当以 $F-\{f_{root}\}$ 中的特征为节点、以特征取值依赖和多值依赖为边形成的(超)图是连通的,即特征集合 $F-\{f_{root}\}$ 是关于 FVD/FMVD 的闭包.

内聚构件包含了通过特征取值依赖关系而关联在一起的特征,这些特征在复用时必然要同时进行特征项的选取,即特征值的变化,满足了“共同变化”原则[8].

3.2 构件规范化模式生成算法

构件设计过程可看作是遵循特定规范化设计原则,将领域特征空间 Ω 划分为一组子空间 $\{\Omega_1,\Omega_2,\ldots,\Omega_n\}$,并将每一个 Ω_i 映射为一个独立构件 C_i 的过程.针对 4 类规范化模式,给出相应的生成算法.

3.2.1 原子/基本构件生成算法

该算法得到的所有构件满足原子构件或基本构件模式,是最简单的一种划分算法,其基本思想是将特征空

间中的每一个特征直接映射为构件.

算法 **1**. 原子构件/基本构件生成算法.

输入:特征空间$\Omega\langle F,D\rangle$.

输出:原子构件/基本构件集合 *BCS*.

(1) 令 $BCS=\varnothing$;

(2) 对$\forall f\in F(\Omega)$且 $child(f)=\varnothing$,构造原子构件 C,并令 $F(C)=\{f\}$,$f_{root}(C)=\{f\}$,$D(C)=\varnothing$,$PS(C)=\{f\}$,$RS(C)=\varnothing$.将 C 加入 BCS,$BCS=BCS\cup C$;

(3) 对$\forall f\in F(\Omega)$且 $child(f)\neq\varnothing$,构造基本构件 C,并令 $F(C)=\{f\}\cup child(f)$,$f_{root}(C)=\{f\}$,$D(C)=\{d|d\in D(\Omega)$且 d 的左部与右部特征均包含在 $F(C)$中$\}$,$PS(C)=\{f\}$,$RS(C)=child(f)$.将 C 加入 BCS,$BCS=BCS\cup C$.

3.2.2 框架构件生成算法

该算法的基本思想是:对一个基本构件中包含的子特征按照相对复用度进行分组,将所有具有相同的相对复用度的特征合并在一起,形成框架构件.

算法 **2**. 框架构件生成算法.

输入:原子构件/基本构件集合 *BCS*.

输出:框架构件集合 *FCS*.

(1) 令 $FCS=\varnothing$;

(2) 对$\forall C\in BCS$ 且 $RS(C)\neq\varnothing$,C 为基本构件,置标记为 0;对$\forall C\in BCS$ 且 $RS(C)=\varnothing$,C 为原子构件,置标记为 1;

(3) 如果 *BCS* 中所有构件标记均为 1,算法结束;否则进入第(4)步;

(4) 任取 *BCS* 中标记为 0 的构件 C.对 C 的所有需求特征 $RS(C)$按照相对复用频度 $RF_R(f_i,C)$的大小进行分组,得到 $F_1,F_2,\ldots,F_n,F_1\cup F_2\cup\ldots\cup F_n=RS(C)$,且每一组中所有特征的 $RF_R(f_i,C)$值均相等.

如果 $n=1$,表明 $RS(C)$中所有特征的相对复用频度都相等,则 C 无须进行分解,将 C 加入 *FCS*,并为其置标志为 1,继续执行(4);否则,考察每一个 $F_i(1\le i\le n)$:

如果$|F_i|=1$,即 F_i 仅包含唯一的特征,则略过 F_i;

否则,对每一个 $F_i(1\le i\le n)$,构造构件 C_i 和一个虚拟特征 vf_i,令 $f_{root}(C_i)=vf_i$,$F(C_i)=F_i\cup\{vf_i\}$,$PS(C_i)=F_i$,$RS(C)=F_i$.同时,在 vf_i 和 F_i 中的每一个特征 f_{ij} 之间构造特征完整性依赖 $vf_i|^M\rightarrow f_{ij}$,并将其加入 $D(C_i)$中.另外,将 $D(C)$中所有左部与右部特征均包含在 $F(C_i)$中的特征依赖复制到 $D(C_i)$中.将 C_i 加入 *FCS*,$FCS=FCS\cup C_i$;

此时,基本构件 C 包含的子特征被划分为$\{C_1,C_2,\ldots,C_m\}$共 $m(1<m\le\lfloor n/2\rfloor)$个框架构件.

(5) 集合 *FCS* 包含了所有对 *BCS* 中基本构件进行划分得到的框架构件.

在第(4)步中,如果严格按照 $RF_R(f_i,C)$的相等性进行分组,可能会造成分组过多,导致最终得到的构件绝对粒度过小.在实际应用中,也可将条件改为:如果$|RF_R(f_i,C)-RF_R(f_j,C)|\le\psi$($\psi$为一个阈值),则特征 f_i,f_j 就属于同一分组.可根据构件绝对粒度的满意程度来调整ψ值的大小.

3.2.3 内聚构件生成算法

该算法的基本思想是:对一个基本构件中包含的子特征按照特征取值依赖/多值依赖的闭包进行分组,将所有处于同一特征依赖闭包中的特征合并在一起,并为之构造虚拟父特征,形成内聚构件.

算法 **3**. 内聚构件生成算法.

输入:原子构件/基本构件集合 *BCS*.

输出:内聚构件集合 *CCS*.

(1) 令 $CCS=\varnothing$;

(2) 对$\forall C\in BCS$ 且 $RS(C)\neq\varnothing$,C 为基本构件,置标记为 0;对$\forall C\in BCS$ 且 $RS(C)=\varnothing$,C 为原子构件,置标记为 1;

(3) 如果 *BCS* 中所有构件标记均为 1,算法结束;否则进入第(4)步;

(4) 任取 *BCS* 中标记为 0 的构件 C.对 C 的所有特征取值依赖/多值依赖 $D_V(C)(\subset D(C))$进行分析:

对 $D_V(C)$中的每一个特征依赖 $X_i\rightarrow Y_i$(或 $X_i\rightarrow\rightarrow Y_i$),求 X_i 在 $D_V(C)$上的闭包 X_i^+.所有的 X_i^+形成集合 $CF(C)$;

对任意 $F_i,F_j\in CF(C)$,如果 $F_i\cap F_j\neq\varnothing$,则将 F_i 与 F_j 合并形成 $F_i\cup F_j$ 并加入 $CF(C)$,同时从 $CF(C)$中删除 F_i,F_j.重复该步骤,直到 $CF(C)$所有的 F_i,F_j 均不能再合并为止;

如果$|CF(C)|=1$,表明 $RS(C)$中所有特征处于同一个特征闭包,则 C 无须进行分解,将 C 从 BCS 移至 CCS,并为其置标志为 1,继续执行(4);否则,考察每一个 $F_i\in CF(C)$:

如果$|F_i|=1$,即 F_i 仅包含唯一的特征,则略过 F_i;

否则,对每一个 F_i,构造构件 C_i 和一个虚拟特征 vf_i,令 $f_{root}(C_i)=vf_i,F(C_i)=F_i\cup\{vf_i\},PS(C_i)=F_i,RS(C)=F_i$.同时,将 $D(C)$中所有左部与右部特征均包含在 $F(C_i)$中的特征依赖复制到 $D(C_i)$中.将 C_i 加入 CCS,$CCS=CCS\cup C_i$.

此时,基本构件 C 包含的子特征被划分为$\{C_1,C_2,\ldots,C_m\}$共 $m(1<m\leq\lfloor n/2\rfloor)$个内聚构件.

(5) 集合 FCS 包含了所有对 BCS 中基本构件进行划分得到的框架构件.

在该算法中,如果按照特征间语义依赖的数目和强度进行子特征划分,就转化为传统的基于聚合度-耦合度的构件划分算法[5−7].与这些算法相比,算法 3 的优势体现在:(1) 采用特征取值/多值依赖而不是语义依赖作为计算特征全局相关性的依据,不需要为各种类型的语义依赖设定权值;(2) 通过计算特征关于取值/多值依赖的闭包实现划分,不需要人为地设定划分的阈值.这两点避免了对设计者经验的过度依赖.

4 实例研究

针对图 1 给出的特征空间进行构件设计,在应用算法 1 之后,得到

$BCS=\{C\langle`ac_{11}',\{f_{11}\},\varnothing,\{f_{11}\},\varnothing\rangle,C\langle`ac_{12}',\{f_{12}\},\varnothing,\{f_{12}\},\varnothing\rangle,C\langle`ac_{13}',\{f_{13}\},\varnothing,\{f_{13}\},\varnothing\rangle,C\langle`ac_{14}',\{f_{14}\},\varnothing,\{f_{14}\},\varnothing\rangle,$
$C\langle`ac_{21}',\{f_{21}\},\varnothing,\{f_{21}\},\varnothing\rangle,C\langle`ac_{22}',\{f_{22}\},\varnothing,\{f_{22}\},\varnothing\rangle,C\langle`ac_{23}',\{f_{23}\},\varnothing,\{f_{23}\},\varnothing\rangle,C\langle`ac_{31}',\{f_{31}\},\varnothing,\{f_{31}\},\varnothing\rangle,$
$C\langle`ac_{32}',\{f_{32}\},\varnothing,\{f_{32}\},\varnothing\rangle,C\langle`bc',\{f\},D,\{f\},\{f_1,f_2,f_3\}\rangle,C\langle`bc_1',\{f_1\},D_1,\{f_1\},\{f_{11},f_{12},f_{13},f_{14}\}\rangle,$
$C\langle`bc_2',\{f_2\},D_2,\{f_2\},\{f_{21},f_{22},f_{23}\}\rangle,C\langle`bc_3',\{f_3\},D_3,\{f_3\},\{f_{31},f_{32}\}\rangle\}.$

共包含 9 个原子构件和 4 个基本构件.

应用算法 2 对 BCS 中基本构件进行规范化,得到

$FCS=\{C\langle`fc_1',\{vf_1\},D_1,\{f_{11},f_{12}\},\{f_{11},f_{12}\}\rangle,C\langle`fc_2',\{vf_2\},D_2,\{f_{13},f_{14}\},\{f_{13},f_{14}\}\rangle,C\langle`fc_3',\{vf_3\},D_3,\{f_{21},f_{22}\},\{f_{21},f_{22}\}\},$

共 4 个框架构件.

应用算法 3 对 BCS 中包含的基本构件进行规范化,得到的 CCS 与 FCS 等价,表明 FCS 中的 4 个框架构件同时也是内聚构件.最终得到的规范化构件形成如图 4 所示的交互模式.

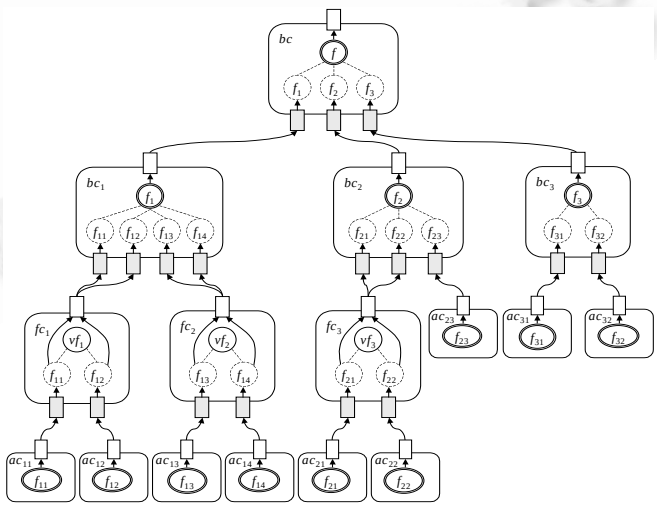

Fig.4 Normalized component sets and relationships between components

图 4 规范化的构件集合及其交互关系

5 结束语

特征模型在软件复用领域一直是非常有用的工具.采用特征空间建立构件语义模型,在传统特征依赖的基础上,扩展得到 4 类特征依赖,并提供了构件的规范化设计模式和设计算法.经过工程中的实践和应用,证明该方法设计得到的构件具有较高的复用性能,对指导构件的设计具有较高的应用价值.

References:

[1] Mili H, Mili A, Yacoub S, Addy E. Reuse-Based Software Engineering: Techniques, Organization, and Controls. New York: John Wiley and Sons, Inc., 2002.

[2] Jia Y. The evolutionary component-based software reuse approach [PhD Thesis]. Beijing: Graduation School, the Chinese Academy of Sciences, 2002 (in Chinese with English abstract).

[3] Edwards SH. A formal model of software subsystems [Ph.D. Thesis]. Columbus: The Ohio State University, 1995.

[4] Kang KC, Cohen SG, Hess JA, Novak WE, Peterson AS. Feature-Oriented domain analysis (FODA) feasibility study. Technical Report, CMU/SEI-90-TR-21, Pittsburgh: Software Engineering Institute, Carnegie Mellon University, 1990.

[5] Lee SD, Yang YJ, Cho ES, Kim SD, Rhew SY. COMO: A UML-based component development methodology. In: Proc. of the 6th Asia Pacific Software Engineering Conf. Takamatsu: IEEE Computer Society Press, 1998. 54–63. http://csdl.computer.org/comp/proceedings/apsec/1999/0509/00/05090054abs.htm

[6] Lee JK, Jung SJ, Kim SD, Jang WH, Ham DH. Component identification method with coupling and cohesion. In: Proc. of the 8th Asia-Pacific Software Engineering Conf. Macau: IEEE Computer Society Press, 2001. 79–88. http://csdl.computer.org/comp/proceedings/apsec/2001/1408/00/14080079abs.htm

[7] Xu W, Yin BL, Li ZY. Research on the business component design of enterprise information system. Journal of Software, 2003,14(7):1213−1220 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/1213.htm

[8] Martin RC. Agile Software Development: Principles, Patterns, and Practices. New York: Prentice Hall, 2002.

附中文参考文献:

[2] 贾育.基于演化计算构件的软件复用方法[博士学位论文].北京:中国科学院研究生院,2002.

[7] 徐玮,尹宝林,李昭原.企业信息系统业务构件设计研究.软件学报,2003,14(7):1213−1220. http://www.jos.org.cn/1000-9825/14/1213.htm

王忠杰(1978－),男,山东龙口人,博士生,主要研究领域为可重构信息系统,软件重构与重用,软件工程.

战德臣(1965－),男,博士,教授,博士生导师,主要研究领域为现代企业管理,数据与知识工程,软件重构与重用.

徐晓飞(1962－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为企业智能计算,管理与决策信息系统,数据库,企业资源计划与供应链管理技术,电子商务与商务智能,知识工程及应用.