基于扩展的粒度计算的软件体系结构模型:EGSA

李长云[1,2],李赣生[1],李 莹[1]
(1. 浙江大学计算机学院,浙江杭州 310027;2. 株洲工学院计算机系,湖南株洲 412008)

摘 要: 针对目前软件体系结构理论基础及 ADL 的不足,尤其是不支持体系结构风格自动发现的缺陷,通过对传统的粒度计算进行扩展,使其能处理粒度间结构关系,建立了软件体系结构模型 EGSA(Extended Granular Computing-Based Software Architecture Model). EGSA 可表达结构多维语义,支持体系结构分层构造、属性分析,并考虑了结构信息对复合构件属性的影响;通过解码和模糊集运算,EGSA 具备自动发现新的体系结构风格的能力. EGSA 是较完整的理论体系,为进一步的研究提供了理论工具.

关键词: 软件体系结构; 粒度计算; 信息粒度; 模糊集; 二部图
中图分类号: TP311 **文献标识码**: A **文章编号**: 0372-2112 (2005) 02-0271-05

EGSA: Extended Granular Computing-Based Software Architecture Model

LI Chang-yun[1,2], LI Gan-sheng[1], LI Ying[1]
(1. *Department of Computer Science & Technology, Zhejiang University, Hangzhou, Zhejiang* 310027, *China*;
2. *Department of Computer, Zhuzhou Institute of Technology, Zhuzhou, Hunan* 412008, *China*)

Abstract: Aiming at the deficiencies in fundamental theory of software architecture, especially the deficiency of discovering software architecture style automatically, the EGSA (Extended Granular Computing - Based Software Architecture Model) is built by extending traditional granular computing. The information of component and connector is described as fuzzy set, and multiple fuzzy matrix is used to express structural multidimensional semantic. Furthermore, the mapping between architectures of different level and the synthetic restraint on the property of software architecture are discussed. By the decoding and fuzzy set operation, the EGSA is capable of discovering new architecture style automatically. All above work have set up basic theory of software architecture and provided theoretic goundation for further study.

Key words: software architecture; granular computing; information granular; fuzzy set; bipartite graph

1 引言

软件体系结构是构件、连接件、约束的集合,是软件在设计构成上的基本的、可供设计选择的形态和总体结构. 软件体系结构的理论模型是软件体系结构的基础和核心. 一般认为,良好的软件体系结构模型应能够充分表达系统的主体结构、宏观特性和基本功能,支持体系结构分层构造和体系结构的分析、设计与验证,便于体系结构风格发现与复用.

目前,在软件体系结构的概念、体系结构描述语言(ADL)、体系结构风格的研究上都取得了一系列成果[1~10]. 体系结构的理论基础及其形式化 ADL 主要有:基于 CSP 的 Wright 和基于π语言的 Darwin,适于描述并发、分布的软件体系结构;基于有穷状态机的 ArTek 适于描述系统的拓扑结构关系;采用属性文法来描述分布式软件体系结构的 DSADL. 但这几种 ADL 都不支持系统的非功能属性分析. 基于时序逻辑的软件体系结构描述语言 XYZ/ADL,可以在统一的时序逻辑框架下描述系统静态语义到实现之间不同抽象层次的规范,便于体系结构的逐步求精描述以及相关性质分析[8]. 但所有上述 ADL 都不具有集成体系结构多视图的能力. Medvidovic 提出了一种 ADL 的分类和比较框架[5],详细分析了多种典型的 ADL 的优点与不足. 目前还没有完全具备良特性的软件体系结构理论模型及其 ADL,尤其是没有任何模型具有从相关领域体系结构中自动或半自动抽取体系结构风格的能力.

粒度计算是近年来提出并发展迅速的学科,广泛应用于人工智能、信号处理和数据挖掘领域,在数据聚类分析、知识发现和表示等方面使用较为成功. 粒度计算研究以某种形式聚集的信息的表达及其处理过程,不同尺寸的信息粒度形成一个金字塔结构[11]. 通过在设计过程中操纵信息粒度的大小来隐藏和揭示信息,粒度成为一种信息的抽象和具体化机制. 软件体系结构设计具有层次性,相邻的两个层次之间存在着一定的转换关系,不同层次的体系结构形成金字塔状. 如果把构件、连接件看作是其上多维属性信息的综合体-信息粒度,则软件体系结构的问题空间和粒度计算的问题空间本质上是相似的. 粒度计算主要的方法工具是模糊数学、粗糙集理论,用以处理模糊和不确定知识. 软件体系结构设计是软件设计的早期决策,也存在许多的模糊信息和不完全知识,正可借助粒度计算的工具处理.

目前的粒度计算主要研究粒度的表示、构造和不同尺寸粒度之间的转化,不处理信息粒度之间的结构关系,这和软件体系结构是不一致的. 为适应软件体系结构建模,我们拓展传

收稿日期:2004-03-15;修回日期:2004-07-18
基金项目:国家自然科学基金(No. 60373062);湖南省自然科学基金(No. 04JJ3052)

统的粒度计算理论,引进信息粒度间结构关系并用受限的二部图表示,以模糊集合表达粒度信息,建立基于扩展的粒度计算的软件体系结构模型:EGSA(Extended Granular Computing-Based Software Architecture Model).

2 基本概念和软件体系结构描述

信息粒度是粒度计算的基本处理对象,文[11]认为是由相似的、功能接近或空间相邻的元素集合.我们把它看成是多个属性信息的集合体,具体定义如下:

定义 1 信息粒度是包含了多个属性信息的信息实体,用三元组(P,V,F)表示,其中 P 表示属性集合,$V=\cup V_x(x\in P)$,V_x 是 x 属性的值域,$F:P\to V$ 是信息函数,反映了信息粒度的全部信息.

根据 IEEE 610.12-1990 软件工程标准词汇中的定义[10],构件是软件系统的结构块单元,是软件功能设计和实现的承载体.在此定义中,构件可以是对象、进程、库、软件产品、数据库或其它计算单元.从计算观点看,构件是具有计算属性的逻辑对象或实现单元,同时构件还包括其它的一些约束属性信息:可靠性、效率、接口特性、可维护性等.

定义 2 软件构件是这样的信息粒度(P,V,F),存在属性 $p_c(x)\in P$,p_c 为对输入变量 x(x 可以是任意类型的数)进行计算的函数.

构件和构件之间通过连接件建立和维持行为关联和信息传递,连接件定义了构件之间交互的规则并且给出了一些实现的机制.连接件并不一定对应具体的软件实现体,但一定规定了消息交换协议,如参数传递方式等.除此之外,连接件还可能具有方向性、响应度、角色等其它属性.我们定义:

定义 3 连接件是规定了构件之间消息交换协议的信息粒度,是应用于构件间进行连接通信的设计单元.

在目前工程实践中,自然语言是最方便使用的描述语言,但自然语言具有歧义性、不一致性、不准确性等缺陷.为克服这些缺陷又不失自然语言的易理解、方便性,使用模糊理论来规范、模型化自然语言.模糊集合论的目的之一就是给自然语言的语义分析建立一个数学模型,概念用某论域上的模糊集表示,同时定义语气算子和模糊算子、模糊集上的各种运算.在此基础上还提出了"用词计算"的概念[12].由于软件体系结构设计是软件设计的早期决策,存在许多的模糊信息和不完全知识,我们规定,信息粒度中属性值以模糊集合表示.设信息粒度具有某属性 p,建立关于 p 的论域 U_p,p 的属性值就可用 U_p 上的模糊集表达,U_p 的建立上下文相关.这种建立在模糊理论基础上的信息描述,近似于自然语言,又具有形式语言的严密性、可推理性.构件和连接件的设计就是尽量实现其所有属性满足要求值的过程.

Bass 等人[7]认为,软件体系结构包括构成系统计算或数据存储单元的构件、规范构件间交互行为和约束关系的连接件、构件和连接件构成的连通图结构即体系结构配置,反映了系统的总体结构.据此,我们定义:

定义 4 软件体系结构是一个由构件、连接件及其形成的拓扑结构构成的粒度空间.设构件集合为 C,连接件集合为 A,拓扑结构关系为 S,软件体系结构 $SA=(C,A,S)$.

传统的粒度空间定义为信息粒度的集合,信息粒度间不包括结构关系,我们的粒度空间处理信息粒度之间的结构信息,是传统粒度空间的拓展.

我们以图表示 SA,构件和连接件为节点,构件之间仅通过连接件建立关联,则 S 是构件和连接件之间的边集.每个连接件至少邻接两个构件,连接件相邻没有意义.规定一个软件体系结构中不允许存在孤立的构件或构件子集,不允许存在并行边.这是符合现实的,构件总是互相交互的,体系结构的一个部分总是和其它部分有着某种形式的联系.几个构件之间如果彼此有多种形式的交互,在语义上可以合并成更高抽象的一种交互形式.因此有:

定义 5 在构件和连接件是否关联的语义下,软件体系结构 $SA=(C,A,S)$ 是一个连通的简单二部图,其中 C,A 为节点集,S 是构件和连接件之间的边集.且具有以下约束:

(1) 对于 $\forall x\in A$,x 的度数至少为 2.

(2) 不存在这样的集合 $A_1\subset A$,A_1 的元素个数大于 1,对于 $\forall x\in A_1$,存在某一非空子集 $C_1\subset C$,$\forall y\in C_1$,$(x,y)\in S$ 且 $\forall z\in(C-C_1)$,$(x,z)\notin S$.

图 1 显示了三个结构图,a 图是合法的软件体系结构图,b 图不符合定义 5,c 图 $a4$ 是图 $a1$ 和 $a2$ 合并而成,在语义上等价 b 图,为表示、处理的归一化、简单化,按定义 5 规定 b 图是不合法的软件体系结构图,c 图是合法的软件体系结构图.

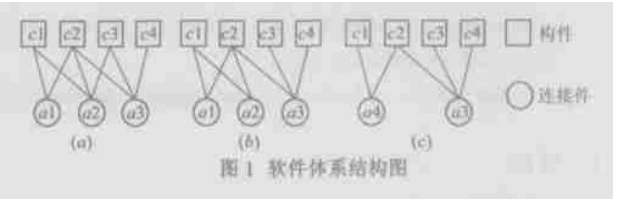

图 1 软件体系结构图

用邻接矩阵 X 表示 SA,设 C 的基数为 m,A 的基数为 n,$X=[x_{ij}]_{(m+n)\times(m+n)}$,由二部图性质,当 $(i\le m$ 且 $j\le m)$ 或 $(i>m$ 且 $j>m)$ 时,$x_{ij}=0$,同时邻接矩阵是对称矩阵,单纯从连接语义看,X 可以压缩成 $m\times n$ 矩阵 X',称为简化矩阵.图 1 a 的

$$X=\begin{bmatrix}0&0&0&0&1&1&0\\0&0&0&0&1&1&1\\0&0&0&0&0&1&1\\0&0&0&0&0&0&1\\1&1&0&0&0&0&0\\1&1&1&0&0&0&0\\0&1&1&1&0&0&0\end{bmatrix},\quad X'=\begin{bmatrix}1&1&0\\1&1&1\\0&1&1\\0&0&1\end{bmatrix}.$$

S 的边如果赋以权值,表示交互强度、频度等语义,S 图成为模糊图,X 和 X' 可解释为 C 和 A 上的模糊关系.

如果不同的属性,赋以不同的权值,表示在此属性维度上 C 和 A 中元素之间的模糊关系,得到此属性维度上矩阵 X_p 和 X'_p,$p\in P$.结构图右上部分 $x_{ij}(i\le m$ 且 $j>m)$ 还可表达连接件角色语义,左下部分 $x_{ij}(i>m$ 且 $j\le m)$ 表达构件端口语义,得到矩阵 X_r,元素取值可以不只是单个数,可以是一维向量数,以充分表达角色端口语义.由此,在每个属性维上形成一

个加权邻接矩阵,所有属性维上的加权邻接矩阵构成一个多重模糊矩阵,形如 $\begin{bmatrix} (x_{11}^1, x_{11}^2, \cdots, x_{11}^k) & \cdots & (x_{1n}^1, x_{1n}^2, \cdots, x_{1n}^k) \\ \cdots & \cdots & \cdots \\ (x_{m1}^1, x_{m1}^2, \cdots, x_{m1}^k) & \cdots & (x_{mn}^1, x_{mn}^2, \cdots, x_{mn}^k) \end{bmatrix}$. 因此,这种加权的多重模糊邻接矩阵表达了软件体系结构的多维属性及结构语义.

3　软件体系结构分层构造

在粒度计算处理信息粒度的过程中,处于不同层次的粒度空间形成一个金字塔结构. 信息粒度越粗的粒度空间,信息就越抽象、细节就越少,信息粒度越细的粒度空间,信息就越具体、细节就越丰富. 粒度计算模型化不同层次的粒度空间交流、转化、处理. 从粗粒度空间构造细粒度空间的过程叫编码,记为 Enc,反之叫解码,记为 Dec. 软件工程的设计原则之一是"从上到下,逐步求精",这对应了粒度计算的编码过程,还有一条"抽象化"原则,反映的是粒度计算的解码过程.

3.1　软件体系结构的结构层次关系

定义 6　在软件体系结构的分层构造过程中,如果有软件体系结构 SA_1、SA_2, SA_1 可经由有限个编码过程转化成 SA_2,称 SA_2 比 SA_1 细,记为 $SA_1 > SA_2$. SA_1 可经由 1 个编码过程转化成 SA_2,称 SA_2 是 SA_1 的直接细化,记为 $SA_1 \geq SA_2$.

显然, > 是可传递的.

定义 7　如果 $SA_1 \geq SA_2$, $SA_1 = (C_1, A_1, S_1)$, $SA_2 = (C_2, A_2, S_2)$,则 SA_1 可由 SA_2 经如下规定的解码过程形成:

(1) 对集合 C_2 进行某关系 R 划分得商集 C_2/R,对于 $\forall C_c \in C_2/R$,有集合 $A_c = \{a_1 \mid a_1 A_2, \exists x \in C_c, (x, a_1) \in S_2$ 且 $\forall y \in (C_2 - C_c), (y, a_1) \notin S_2\}$,设 $c_1 = (C_c, A_c, S_c)$, $S_c = \{(c, a) \mid c \in C_c, a \in A_c$ 有 $(c, a) \in S_2$ 且 $\forall c' \in (C_2 - C_c), (c', a) \notin S_2\}$, c_1 叫做复合构件, $C_1 = \{c_1\}$, $A_1' = A_2 - \cup A_c$.

(2) 设以 C_1 和 A_1' 的元素为点集形成结构 $SA_1' = (C_1, A_1', S_1')$, $S_1' = \{(c_b, y) \mid c_b \in C_1$,设 $c_b = (C_c, A_c, S_c)$, $y \in A_1'$, $\exists x \in C_c, (x, y) \in S_2\}$. SA_1' 不一定是合法的体系结构.

(3) 如果有元素个数大于 1 的集合 $A_{>1} \subseteq A_1'$, $\forall a_2 \in A_{>1}$,存在非空集合 $C_3 \subseteq C_1$, $\forall x \in C_3$,有 $(a_2, x) \in S_1'$,且 $\forall y \in (C_1 - C_3), (a_2, y) \notin S_1'$,则设 $A_1 = \{A_{>1}\} \cup (A_1' - A_{>1})$. $A_{>1}$ 叫做复合连接件.

(4) 以 C_1 和 A_1 的元素为点形成软件体系结构 SA_1, $S_1 = \{(x, y) \mid x \in C_1, y \in A_1, \exists a_3 \in y$ 有 $(x, a_3) \in S_1'\}$.

划分关系 R 可以依据构件功能相似性、耦合紧密度等. 图 2 展现了一个软件体系结构解码过程例子, SA_2 中构件划分为 $\{\{c1, c2\}, \{c3\}, \{c4, c5\}\}$,经解码 1,2 步骤得 SA_1', SA_1' 经 3,4 步骤得 SA_1.

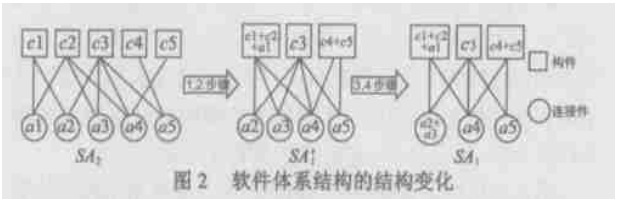

图 2　软件体系结构的结构变化

显然,定义 7 说明了软件体系结构分层构造中结构的分解和合成所应遵守的约束,可以清楚地跟踪构件和连接件的变化轨迹. 从形式上看,复合构件 (C_c, A_c, S_c) 与体系结构相似,但复合构件并不等同软件体系结构.

定理 1　设所有软件体系结构的集合为 $\{SA\}$,所有复合构件的集合为 $\{c_1\}$,则 $\{SA\} \subset \{c_1\}$,即所有的软件体系结构都是复合构件,但存在不是软件体系结构的复合构件.

证明　设有任意软件体系结构 (C, A, S),把 C 中所有元素作为一个划分解码,由定义 7 得整个软件体系结构为一个复合构件,因此 $\{SA\} \subseteq \{c_1\}$.

又图 2 中 SA_2 中 $c4, c5$ 是一划分,解码后得复合构件 $(c4 + c5)$, $(c4 + c5)$ 内部没有连接件,显然 $c4, c5$ 不相邻, $(c4 + c5)$ 构件的结构拓扑是零图,不连通,由定义 5 知体系结构的结构拓扑为连通图,所以 $(c4 + c5)$ 构件不是软件体系结构,由此 $\{SA\} \subset \{c_1\}$,得证.

一般认为,复合构件是由多个构件组成,我们的模型反映复合构件是由多个构件及这多个构件内部交互的连接件构成,也包含形式为二部图的内部结构,揭示了构件是体系结构的组成成分而软件体系结构又是构件的递归语义关系. 复合构件内部结构信息表达也可依照软件体系结构的描述进行.

在定义 7 和结构邻接矩阵的多维语义表达的基础上可以建立起更复杂的结构语义模型. 对于从粗粒度向细粒度的粒度空间的编码过程,由于存在多种选择,如果考虑到结构邻接矩阵的多维语义,更是一个无限的设计空间,难以给出确切的规定. 事实上,软件体系结构的编码过程(细化设计),正是充分发挥设计人员的创造性的地方. 但无论如何,相邻的两层其结构拓扑受定义 7 规定的约束.

定理 2　在定义 7 的约束下,有 $SA_1 \geq SA_2 \geq SA3 \geq \cdots \geq SA_n$,对应的解码过程序列 $(Dec_1, Dec_2, \cdots, Dec_{n-1})$,则可设计一个等价的解码过程 Dec_x,经 Dec_x 过程 $SA_1 > SA_n$.

证明　要证明序列 $(Dec_1, Dec_2, \cdots, Dec_{n-1}) \Leftrightarrow Dec_x$,只要证明任意的 $SA_i \geq SA_{i+1} \geq SA_{i+2}$ 对应的解码过程序列 (Dec_i, Dec_{i+1}) 某一个 Dec_y 即可. 因为由 $(Dec_1, Dec_2, \cdots, Dec_{n-1}) \Leftrightarrow (Dec_{12}, Dec_3, \cdots, Dec_{n-1}) \Leftrightarrow (Dec_{123}, Dec_4, \cdots, Dec_{n-1}) \Leftrightarrow \cdots \Leftrightarrow Dec_x$. 下面证 $(Dec_i, Dec_{i+1}) \Leftrightarrow Dec_y$.

设 Dec_i, Dec_{i+1} 构件划分关系分别为 R_i、R_{i+1}, $SA_i = (C_i, A_i, S_i)$, $SA_{i+1} = (C_{i+1}, A_{i+1}, S_{i+1})$, $SA_{i+2} = (C_{i+2}, A_{i+2}, S_{i+2})$, $C_{i+1} = (C_c^i, A_c^i, S_c^i)$, $C_{i+1} = (C_c^{i+1}, A_c^{i+1}, S_c^{i+1})$, $C_{i+2} = (C_c^{i+2}, A_c^{i+2}, S_c^{i+2})$. 据定义 7,则 $C_c^{i+1} = C_c^i / R_i$, $C_c^{i+2} = C_c^{i+1} / R_{i+1}$,因此 C_c^{i+2} 是 C_c^i 的商集,设为 $C_c^{i+2} = C_c^i / R_y$. $\forall x, y \in C_c^i$ 且 $xR_i y \Rightarrow x\ R_{i+1}\ y \Rightarrow x\ R_y\ y$. 以 C_c^i / R_y 开始定义 7 的解码过程最后得 $SA_y = (C_y, A_y, S_y)$,解码过程即为所求 Dec_y. 只要证 $SA_{i+2} = SA_y$ 即得结论,也就是 $C_{i+2} = C_y$, $A_{i+2} = A_y$, $S_{i+2} = S_y$. 由于证明大长,但逻辑并不复杂,这里省略.

推论 1　在定义 7 的约束下,有 $SA_1 \geq SA_2 \geq SA_3 \geq \cdots \geq SA_n$,对应的编码过程序列 $(Enc_1, Enc_2, \cdots, Enc_{n-1})$,则可设计一个等价的解码过程 Enc_x,经 Enc_x 过程 $SA_1 \geq SA_n$.

证明　略.

定理 2 和推论 1 说明软件体系结构设计的层次可多可

少,层次多则设计步骤多,但每一步工作简单,较易设计,层次少则设计步骤少,但每一步工作复杂,难以设计,两种方式都可得到同样的最终设计结果.

3.2 属性合成和跟踪

软件体系结构是信息粒度空间,软件体系结构设计就是构造满足其信息要求的信息粒度及结构关系. 不同层次的粒度空间信息应该一致,高层次体系结构是低层次软件体系结构的抽象、约束、设计指导,低层次体系结构是高层次体系结构的具体化、设计实现. 不同层次的粒度空间信息抽象化程度不同,描述信息的属性集合也可能不同,粗粒度信息的某维属性可能由下一层次的几维属性合成,如高层次体系结构的正确性分解为低层次体系结构的可追踪性、完备性、一致性,可靠性包含容错性和健壮性等. 通过低层次的各分属性值求得高层次的实际合成属性值,再与高层次的合成属性设计要求值作比较,可评估低层次对高层次的实现程度. 这实际上是一种软件质量多级评价技术,对合成属性的求取可使用模糊数学中综合评判技术进行,文献[13]也提出了一种变权的评判技术,综合考虑了各因素(分属性)的优度和目标值(合成属性)关于因素的水平组态,这里不再详述. 另一方面,信息粒度的全部信息体现在其所有的属性值中,复合构件(复合连接件)的信息由其内部的构件、连接件和结构信息决定. 已知复合构件(复合连接件)组成成分的某属性值及结构信息在此维属性上的投影矩阵,可以推导出复合构件(复合连接件)在这维属性上的值. 下面讨论复合构件同维属性的合成.

定义 8 设复合构件(复合连接件) (P, V, F),内部结构表示为 (C_c, A_c, S_c), $C_c = \{c_1, c_2, \cdots, c_n\}$,各构件 $p(p \in P)$ 属性值 $V_p^{c_i}(i \in (1, \cdots, n))$, $A_c = \{a_1, a_2, \cdots, a_m\}$,各连接件 p 属性值 $V_p^{a_j}(j \in (1, \cdots, m))$, $V_p^{a_j} \in V)$,在 p 属性维上结构加权矩阵 X_p,则复合构件 p 属性值 $V_p(V_p \in V)$

$$V_p = F_p(V_p^{c_1}, V_p^{c_2}, \cdots, V_p^{c_n}, V_p^{a_1}, V_p^{a_2}, \cdots, V_p^{a_m}, X_p)$$

信息合成函数 F_p 满足以下几条性质.

(1) 属性值是模糊集, F_p 结果也应是模糊集; (2) $F_p(\Phi, \Phi, \cdots, \Phi, [0]) = \Phi$; (3) 如果 $x_i \subseteq x_i'$,则 $F_p(x_1, x_2, \cdots, x_k) \subseteq F_p(x_1', x_2', \cdots, x_k')$.

由于 F_p 把 $V_p^{a_j}$ 作为参数,考虑了构件之间交互连接件对复合构件属性的影响, X_p 作为参数,考虑了结构信息对复合构件属性的影响,这是所有其它模型没有做到的. 具体的 F_p 应依赖属性的具体特点而设计,如计算特性 V_c 可设计如下:

$$V_c = V_{c^1}^{c_1} \cup V_{c^2}^{c_2} \cup \cdots \cup V_{c^n}^{c_n} \cup V_{c^1}^{a_1} \cup V_{c^2}^{a_2} \cup \cdots \cup V_{c^m}^{a_m}$$

结构拓扑并不执行任何计算功能,所以 X_c 对结果值没有影响. 上式反映复合构件执行的功能是其各个构成元素执行的功能总和.

复合连接件同维属性的合成类似复合构件同维属性合成.

4 软件体系结构风格和软件体系结构风格发现

研究体系结构风格有助于更准确地把握具有特定风格的软件系统的各种特征,以便设计人员能在系统结构级上尽早达成共识,提高系统的可重用性[14]. 体系结构风格是对一类相似体系结构共性信息的抽象,体系结构风格既定义了构件及连接方式的各种属性,又规定了它们的组合规则和限制[6].

目前有许多的体系结构风格:管道和过滤器、层次、解释器、客户/服务器、消息总线等. 由于软件体系结构是系统的高层抽象,反映了软件的局部和总体计算构件的构成,以及这些构件之间的相互作用关系,因而较算法更稳定,更适合于复用,体系结构风格就是体系结构复用的主要方式. 体系结构风格的汇编总结和发现已成为体系结构研究的一个热点,尤其是特定领域的体系结构风格发现. 已知相关领域的多个软件体系结构描述,从中自动抽取领域体系结构风格,对发现新的软件体系结构风格,提高软件复用度和生产率具有重要意义.

定理 3 任意多个软件体系结构,通过解码过程,总可以使它们在拓扑结构上同构.

证明 把软件系统视作一个整体复合构件,假设其无限可分,则拓扑结构在同构意义上的所有软件体系结构都可通过如图 3 a 所示的分层编码过程得到,其中第 k 层的每种拓扑结构含 k 个构件. 把每一个软件体系结构的拓扑结构作为节点,相邻层的软件体系结构间的解码或编码关系作为边,则形成 b 图所示图结构. 根据定义 7 的解码过程可知,第 k 层的一种拓扑结构能解码为第 $k-1$ 层中的至少一种拓扑结构,若规定只取解码结果中的任意一种拓扑结构作为解码结果,则 b 图将变换为 c 图所示的树结构. 显然,树上的任意多个节点总有共同的祖先节点,而由节点到其祖先节点的路径意味着解码过程,意即任意多个软件体系结构,通过解码过程,总可以达到在拓扑结构意义上共同的软件体系结构. 极端的情况,每一个软件体系结构都可解码成它们自己分层构造过程中软件体系簇的最粗粒度,即根节点,其拓扑结构都是一个孤点图,自然同构. 因此,定理成立.

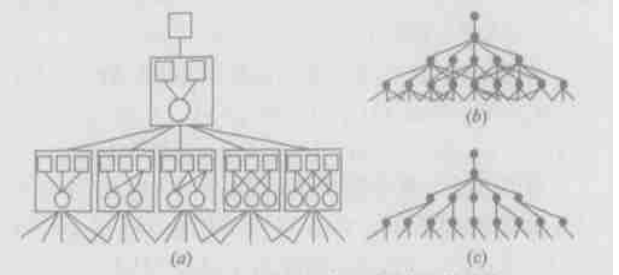

图 3 分层构造形成的软件体系结构拓扑结构

定义 9 任意多个拓扑结构同构的软件体系结构,设 $SA = (C, A, S)$, $SA_1 = (C_1, A_1, S_1)$, $SA_2 = (C_2, A_2, S_2)$, $\cdots$, $SA_n = (C_n, A_n, S_n)$,在同构定义中, $c_i^j(c_i^j \in C_i)$ 互为对应, $d_i^j(d_i^j \in A_i)$ 互为对应, $s_i^j(s_i^j$ 为 S_i 上的第 j 条边) 互为对应,属性集为 P,如果满足: $\forall x \in P$, $c^j(c^j \in C)$ 的 x 属性值 $V_x = V_x^1 \cap V_x^2 \cap \cdots \cap V_x^n$ (V_x^i 是 c_i^j 在 x 属性上的值), $d^j(d^j \in A)$ 的 x 属性值 $V_x' = V_x^1 \cap V_x^2 \cap \cdots \cap V_x^h$ (V_x^f 是 d_i^j 在 x 属性上的值), S 在 x 属性维上的邻接加权矩阵 $X_x = (X_x^1 \cap X_x^2 \cap \cdots \cap X_x^n)$, ($X_x^i$ 是 S_i 在 x 属性维上的邻接加权矩阵,矩阵 $\cap$ 运算定义为取同构对应边 s_i^j 的最小权值),则 SA 是从 SA_i 抽取的体系结构风格.

定义 9 反映了软件体系结构风格是一簇体系结构的共享的结构和语义. 这一簇体系结构共同的结构、属性信息、约束

越多,抽取的体系结构风格领域信息就越多、越具体,灵活性和适应性就越差,反之体系结构风格抽象度和复用性就越高,但可供复用的领域信息却越小.

图 4 显示了从两个软件体系结构发现一个简化的客户/服务器风格的过程.

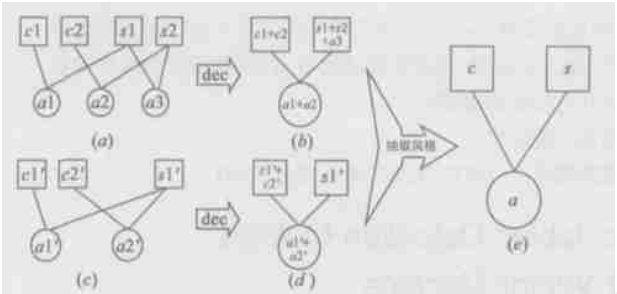

图 4　软件体系结构风格发现

非形式地说明,(1)图表示一工资人事系统体系结构,(1)图 $s1$ 提供工资事务服务,$s2$ 提供人事事务服务,$c1$ 向 $s1$ 请求工资查询、统计服务,$c2$ 向 $s2$ 请求人事信息查询、编辑服务,$a1$,$a2$,$a3$ 连接件协议为 RPC;(3)图表示一商场事务处理系统体系结构,$s1$ 提供联机销售服务,$c1'$,$c2$ 表示销售处理终端,分别通过 $a1'$,$a2$ 同 $s1$ 请求在线销售服务,$a1'$,$a2$ 连接件协议为 RMI.两个系统的体系结构拓扑并不同构.(1)经解码过程得(2)图,(3)经解码过程得(4)图,(2)(4)图同构.由 3.2 节知(2)图中语义为 $(c1 + c2)$ 构件通过 $(a1 + a2)$ 连接件的 RPC 协议向 $(s1 + s2 + a3)$ 请求工资、人事信息服务,(4)图中语义为 $(c1' + c2')$ 构件通过 $(a1' + a2')$ 连接件的 RMI 协议向 $s1$ 请求在线销售服务.(2)(4)体系结构经定义 11 中体系结构风格抽取运算得(5)体系结构,其语义为 c 构件(客户端)经连接件 a 的协议向 s 构件(服务端)请求服务,这正符合客户/服务器结构定义.严格地说,构件、连接件的属性值应用模糊集表达,扎德在文[12]中提出一种把自然语言构造成模糊集、模糊句型的方法,不再详述.因此,我们的方法是可行的.

5　结论和进一步的工作

本文扩展传统的粒度计算,引进信息粒度之间交互结构关系,在此基础上给出了构件、连接件、软件体系结构的概念和描述;用模糊集描述构件、连接件的粒度信息,受限的二部图描述结构拓扑,多重模糊矩阵表达结构多维语义,体系结构设计就是构造满足其信息要求的粒度空间;讨论了不同层次的体系结构的拓扑映射关系和属性合成约束,特别是考虑了结构信息对复合构件属性的影响;EGSA 从一个全新的角度为软件体系结构建模,运用粒度计算的概念、方法工具,使体系结构模型具备从相关领域体系结构中自动抽取新的体系结构风格的能力.EGSA 为进一步研究软件体系结构的相关问题奠定了基础.下一步的工作是继续完善 EGSA 理论,同时以 EGSA 为基础,研制体系结构 ADL 及其支持工具,建立体系结构质量评价模型,构造领域相关、属性类型相关的具体信息函数和属性合成函数.

参考文献:

[1] Huiqun Yu, Xudong He, Yi Deng, Lian Mo. A formal method for analyzing software architecture models in SAM[A]. 26th annual international computer software and applications conference[C]. Oxford, England, 2002. 26 - 29.

[2] Marsala, Peyman S. Issues in modeling and analyzing dynamic software architectures[A]. Proceedings of the international workshop on the role of software architecture in testing and analysis[C]. Marsala, Sicily, Italy, 1998. 368 - 372.

[3] Bernardo M, Ciancarini P, Lorenzo D. Architecting families of software systems with process algebras[J]. ACM Transactions on Software Engineering and Methodology, 2002, 11(4): 386 - 426.

[4] Aldrich J, Chambers C, Notkin D. ArchJava: connecting software architecture to implementation[A]. 2002 international conference on software engineering[C]. Orlando, Florida, USA, 2002. 187 - 197.

[5] Medvidovic N., Richard N. A classification and comparison framework for software architecture description languages[J]. IEEE transactions on software engineering, 2000, 26(1): 70 - 93.

[6] Garlan D, Shaw M. An introduction to software architecture[R]. Carnegie Mellon University, 1994, CMU/SEI-94-TR-21.

[7] Bass L, Clement P, Kazman R. Soft architecture in practice[M]. Boston: Addison-wesley, 1998, 221 - 228.

[8] 朱雪阳,唐稚松. 基于时序逻辑的软件体系结构描述语言 XYZ/ADL[J]. 软件学报, 2003, 14(4): 713 - 720.

[9] Hofmeister C, Nord T, Soni D. Applied software architecture[M]. Boston: Addison-wesley, 2000.

[10] IEEE. 610. 12 - 1990 - 1998, IEEE Glossary of Software Engineering Terminology[S].

[11] Bargiela A, Pedrycz W. Granular Computing: An Introduction[M]. Boston: Kluwer Academic Publishers, 2002.

[12] Zadeh L. Fuzzy logic = computing with words[J]. IEEE Transactions On Fuzzy Systems, 1996, 4(2): 103 - 111.

[13] 汪培庄,李洪兴. 模糊系统理论与模糊计算机[M]. 北京:科学出版社, 1996.

[14] 焦文品,史忠植. 用 XYZ/E 形式化体系结构风格[J]. 软件学报, 2000, 11(3): 410 - 415.

[15] 张铃,张钹. 模糊商空间理论(模糊粒度计算方法)[J]. 软件学报, 2003, 14(4): 770 - 776.

作者简介:

李长云　男,1972 年生,副教授,博士研究生,湖南耒阳人,研究方向:软件体系结构和软件自动化. E-mail: lichangy@sina.com.

李赣生　男,1940 年生,教授,博士生导师,浙江杭州人,研究方向:编译理论和软件自动化.