形式概念分析与软件过程改进*)

沈夏炯[1,2] 白景华[3] 刘宗田[1] 向国全[2]

(上海大学计算机工程与科学学院 上海200072)[1] (河南大学计算机与信息工程学院 河南开封475001)[2]

(开封大学数理教研室 河南开封475001)[3]

Formal Concept Analysis and Software Process Improvement

SHEN Xia-Jiong[1,2] BAI Jing-Hua[3] LIU Zong-Tian[1] XIANG GUO-Quan[2]

(Department of Computer Engineering and Science, Shanghai University, Shanghai 200072, China)[1]

(Department of Computer and Information Engineering, Henan University, Kaifeng 475001, China)[2]

(Staff Room of Mathematics and Physics, Kaifeng University, Kaifeng 475001, China)[3]

Abstract The paper focuses on the method of formal concept analysis in data analyzing for software process improvement. It sets up that not only the Software Quality Guarantee Platform being developed should have the common functions including enterprise process defining, process cutting out, executing and monitoring, and the normal data analysis tools such as OLAP, but also one should develop the novel data analysis tools based on formal methodology by starting with maturity definition of process data, so as to discover valuable new knowledge from large database and support continuous software process improvement. Further more, the paper demonstrates the software process data analysis using concept lattice in an example.

Keywords Formal concept, Concept lattice, Process improvement, Association rule, CMM

1 引言

软件过程的改进是一个复杂的过程。影响软件过程改进的因素有很多，既有软件企业管理层对其意义的理解和倡导，也有软件过程实施当中数据的收集和分析。因为从理论、方法和技术上讲，后者涉及更多的富有挑战性的领域，所以数据的收集和分析(尤其是数据分析)更加直接地决定了过程改进的成功与否。比如，基于 CMM 的软件过程改进主要解决以下问题[6]：

·用常识表示成熟度级别和关键过程域的似是而非的表述——缺乏度量；

·从 TQM(全面质量管理)工作中获得的一般数据——数据定义不完备；

·对已公布的数量有限的案例的研究——简单的数据分析。

我们从中可以看出，对于数据分析的要求是上述关键因素的主要内容。但是到目前为止，还没有任何关于软件企业成熟度级别提高所产生影响的正式有效的研究报告。这正说明软件企业在努力提高管理水平和改进软件过程的同时，并没有比较完备地定义和收集过程数据，更没有真正地充分利用这些宝贵的数据可能带来的利益。尽管有不少与数据库管理系统捆绑的 OLAP 产品，然而它们只能用于数据超方体的切片和统计以及多粒度的下钻和上卷，不能发掘数据中蕴藏的大量有价值的未知知识。

结合目前正在进行的软件质量保障平台开发工作，我们认为：一个好的软件质量保障平台，除了能支持软件企业形成自己定义的软件过程，有效地跟踪，监控软件过程的实施以外，还应该具有强大的、实用的数据分析能力，以有力地支持软件企业的过程改进。欲达到这个目的，必须对过程数据进行完备的定义和存储。因此，研究基于形式方法的数据分析技术并和数据挖掘技术相结合是确保软件质量保障平台实用性的关键所在。

2 形式概念分析的基本理论

形式概念分析是一种数据分析方法。它以序论(Order theory)为基础，尤其是以完全格理论(The theory of complete lattices)为基础[1]，具有严密的数学基础。形式概念分析根据哲学中的“概念”论述，通过对概念的外延和内涵进行抽象，形式化地描述概念，对概念及其相互关联进行严格分类和界定，进而表达“知识”，支持相关领域的研究和应用。

形式概念分析方法首先考察信息表(information table)，根据信息表构造概念格(concept lattice，由 Wille R. 于1982年提出)[3]，然后结合数据挖掘技术从概念格中提取规则(rules)。

在形式概念分析中，信息表被定义为一个三元组 K=(U, D, R)，其中 U 和 D 是集合，而 R 是 U 和 D 间的二元关系，即 R⊆U×D。U 和 D 的元素分别被称为形式对象(简称对象)和形式特征(简称特征)，而 oRd(即(o, d)∈R)被读作“对象 o 具有特征 d”，如表1所示。

表1 对象和特征

	d
o	×

*)本文受国家自然科学基金“分布式概念格数学模型及算法研究”资助，项目编号60275022.沈夏炯 博士研究生，副教授，主要研究领域为软件工程、分布式/并行处理及分布式存储。白景华 讲师，主要研究领域为 Rough set 和数学建模方法。刘宗田 博士生导师，教授，主要研究领域为人工智能和软件工程。向国全 教授，主要研究领域为计算机网络和软件工程。

设符号 P 表示幂集运算，则在信息表 K 中，可以定义两个映射 f:P(U)→P(D)和 g:P(D)→P(U)，

$\forall O_1 \subseteq U: f(O_1)=\{d \mid \forall x \in O_1(xRd)\}$

$\forall D_1 \subseteq D: g(D_1)=\{x \mid \forall d \in D_1(xRd)\}$

它们被称为 P(U)和 P(D)之间的 Galois 联接。对于任意二元组$(O_1, D_1) \in P(U) \times P(D)$，如果同时满足$O_1=g(D_1)$和$D_1=f(O_1)$，则称该二元组是信息表 K 的一个形式概念(formal concept)。对于给定的概念(这里"概念"是形式概念的简称，下同)。

$C=(O_1, D_1)$

称O_1为形式概念 C 的外延，记为 Extent(C)；称D_1为形式概念 C 的内涵，记为 Intent(C)。K 的所有形式概念的集合被标记为 CS(K)。

CS(K)上存在一个重要的关系，即子概念-超概念关系(又称为例化-泛化关系，或前驱-后继关系)。其定义如下：给定形式概念(O_1, D_1)和(O_2, D_2)，如果$O_1 \subseteq O_2$(等价于$D_2 \subseteq D_1$)，则概念(O_1, D_1)是概念(O_2, D_2)的子概念(也称为后继)，概念(O_2, D_2)是概念(O_1, D_1)的超概念(也称为前驱)，记为

$(O_1, D_1) \leqslant (O_2, D_2)$

因为子概念-超概念关系满足自反性、反对称性和传递性，所以它是 CS(K)上的偏序关系。通过这个关系，我们得到一个偏序集$(CS(K), \leqslant)$，并且对于 CS(K)任意非空子集 S，S 中的任意两个形式概念都有最小上界和最大下界，所以偏序集$(CS(K), \leqslant)$是一个完全格，被称为信息表 K 的概念格，记为 L(K)。

概念格上的基本定理：设 K=(U, D, R)为一信息表，L(K)=(CS(K), ≤)是信息表 K 的概念格，那么 L(K)为一完全格。如果对于任意两个形式概念$C_1=(O_1, D_1)$，$C_2=(O_2, D_2) \in CS(K)$，其最小上界 sup 和最大下界 inf 分别为：

$\sup(C_1, C_2)=(g(f(O_1 \cup O_2)), D_1 \cap D_2)$

$\inf(C_1, C_2)=(O_1 \cap O_2, f(g(D_1 \cup D_2)))$

概念格可以图形化表示为有标号的线图(也称为 Hasse 图)，这使得给定数据背景的概念结构变得清晰和易于理解，从而实现了概念格的可视化。限于篇幅，关于概念格的其它内容，如内涵缩减[3]和生成算法[4]，请参阅有关文献，这里不再赘述。以下我们主要讨论如何运用形式概念方法从软件过程数据中提取关联规则(association rule)。

3 关联规则的基本概念

表2 一个事务数据库

Tid	项集
1	i_1, i_2, i_5
2	i_2, i_4
3	i_2, i_3
4	i_1, i_2, i_4
5	i_1, i_3
6	i_2, i_3
7	i_1, i_3
8	i_1, i_2, i_3, i_5
9	i_1, i_2, i_3

令$I=\{i_1, i_2, \cdots, i_m\}$为项(terms)的集合，m 是一个正整数，表示所考察的项的最大数目。项的集合被称为项集，含有 k 个项的项集称为 *k*-项集。这里项与问题相关，比如项是在一个测试步骤中发现有缺陷的模块，或者是一份账单上列出的商品类。事务(transaction)数据库 TD 中的每项事务 Tr 都有一个唯一的标识 Tid，且含有一个项集$T \subseteq I$。例如，表2是一个事务数据库，m=5。其中 Tid=3的事务所对应的项集是$T=\{i_2, i_3\}$。

我们主要考察关联规则。关联规则是形如$A \Rightarrow B$的蕴涵式，其中$A, B \subset I$且$A \cap B=\varnothing$。规则的一种客观度量是支持度(support)。规则的支持度表示满足规则的样本百分比。支持度是概率$P(A \cup B)$，其中$A \cup B$表示同时包含 A 和 B 的事务，即项集 A 和 B 的并。关联规则的另一种客观度量是置信度(confidence)。置信度是条件概率$P(B \mid A)$，既包含 A 的事务也包含 B 的概率。更形式地，支持度和置信度定义为：

$Support(A \Rightarrow B)=P(A \cup B)$

$Confidence(A \Rightarrow B)=P(B \mid A)$

一般来说，我们需要找出的是那些支持度和置信度分别大于或等于用户指定的最小支持度(min_sup)和最小置信度(min_conf)的关联规则。

那么，如何从大型数据库中找出关联规则?其过程可以分为以下两个步骤[2]：

·找出所有的最大频繁项集及其支持度(如果项集满足支持度阈值，称它为频繁项集)；

·根据找到的频繁项集导出所有的置信度大于或等于用户指定的置信度阈值的关联规则。

比如，如果指定支持度计数值(为直观，没有采用百分比)为2，上例中可找到两个频繁项集$\{i_1, i_2, i_3\}$和$\{i_1, i_2, i_5\}$。进一步计算可信度，只考察$\{i_1, i_2, i_5\}$，所有可能的关联规则形式的组合以及对应置信度如表3所示。如果设定置信度阈值为70%，则只有第2、第3和第6共3条关联规则被生成。

表3 频繁项集中候选规则和置信度

$i_1 \wedge i_2 \Rightarrow i_5$	confidence=2/4=50%
$i_1 \wedge i_5 \Rightarrow i_2$	confidence=2/2=100%
$i_2 \wedge i_5 \Rightarrow i_1$	confidence=2/2=100%
$i_1 \Rightarrow i_2 \wedge i_5$	confidence=2/6=33%
$i_2 \Rightarrow i_1 \wedge i_5$	confidence=2/7=29%
$i_5 \Rightarrow i_1 \wedge i_2$	confidence=2/2=100%

概念格除了能够从数据中分类和定义概念外，它还可用于发现对象之间、属性之间的依赖关系。这包括两层含义：

(1)扫描部分或全部格结构，生成将来可以使用的规则集；

(2)浏览格结构，以检验某条给定的规则是否成立。

第一种情况对于基于知识的环境系统很有用，它通过生成新的规则集来增强知识的基础。第二种情况的动机在于，常常会有这样一种情况，即某人想去检验某条假设是否成立，这时我们用不着去从格中生成所有的规则集，而只需要在格中找到包含假设前件的最小格节点，然后检验该格节点的内涵是否也包含假设的后件即可；如果包含，则假设成立，否则不成立。

我们可以观察到项目集之间的关系能用概念格很好地建模。给定一个概念格的格节点(X, Y)，如果把 X 看作事务集，Y 看作项集，一个概念格的格节点就表示具有项集 Y 的最大事务集。实际上，在规则挖掘中，X 的内容并不重要，重要的是 X 元素的个数，即|X|。根据这一点，如果|X|大于支持度阈值 t，则 Y 为一频繁项集。

4 实例分析

表4所示的信息表记录了某软件企业的测试小组在一次测试中发现有缺陷的模块和对应的缺陷类。

表4 有缺陷的模块

		a	b	c	d	e	f	g	h	i
1	模块1	×	×					×		
2	模块2	×	×					×	×	
3	模块3	×	×	×				×	×	
4	模块4	×		×				×	×	×
5	模块5	×	×		×		×			
6	模块6	×	×	×	×		×			
7	模块7	×		×	×	×				
8	模块8	×		×	×		×			

各属性含义是：a-代码错，b-参数类型错，c-参数个数错，d-输出错，e-循环错，f-分支错，g-类型错，h-变量错，i-指针错。

根据模块缺陷信息表，使用 Gordin 算法[4]我们得到对应的概念格，如图1所示。图中数字是信息表中模块的编号，字母是缺陷编号。并且为简洁起见，概念采用简写形式，如({5，6，7，8}，{a，d})简写为(5678，ad)。

如果给定支持度阈值为37%，则图中着色概念的内涵即为频繁项集，如(568，adf)，其支持度为3/8＝37.5＞37%，其频繁项集为{a，d，f}。根据上述寻找关联规则步骤的第二步，我们可求出与概念(568，adf)相应的关联规则。首先计算所有候选关联规则及其置信度，如表5所示。

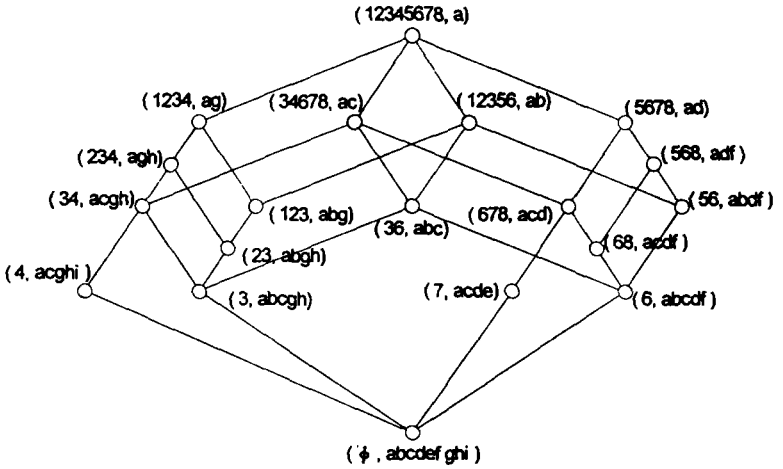

图1 模块缺陷对应的概念格

表5 概念格中所有候选关联规则及其置信度

候选关联规则	置信度	规则含义
a∧d⇒f	3/4 = 75%	如果模块有代码错和输出错，则有分支错
a∧f⇒d	3/3 = 100%	如果模块有代码错和分支错，则有输出错
d∧f⇒a	3/3 = 100%	如果模块有输出错和分支错，则有代码错
a⇒d∧f	3/8 = 37.5%	如果模块有代码错，则有输出错和分支错
d⇒a∧f	3/4 = 75%	如果模块有输出错，则有代码错和分支错
f⇒a∧d	3/3 = 100%	如果模块有分支错，则有代码错和输出错

假设置信度阈值为90%，则第2、3、6条规则被输出。如果置信度阈值降低为70%，则会输出更多的关联规则。需要强调的是，对关联规则的理解一定要结合其置信度。

结论 形式概念分析方法具有严谨的数学理论基础，因而在各个研究、应用领域具有强大活力。尤其在智能信息处理方面，因其对概念完整简洁的抽象能力、表达能力和处理能力，而受到越来越多学者的关注。K. Devlin 在其著作"Turning Information to Knowledge"中使用3个式子这样表述数据、信息和知识之间的关系[5]：

Data = Signs + Syntax

Information = Data + Meaning

Knowledge = Internalized information + Ability to utilize the information

通过前面的论述不难发现，概念格不仅仅能够从数据中分类和定义概念，发现对象之间、属性之间的依赖关系等信息，还很好地利用概念格中的信息形成知识。另外，概念格还是研究分布式处理和存储的有力工具。在软件过程改进的实施过程中，存在大量的分布数据需要存储，需要分析和利用，为形式概念分析方法的应用提供了广阔的前景。当然，概念格也存在自身的问题，比如因描述数据过于完备而导致存储量太大。对于海量数据分析和存储应用来说，这一问题愈发突出。目前人们正就此开展研究，并提出不少有益的方法[7]。在研究和应用中，我们提出信息表格同构的判定问题，通过解决此问题，也可以从一个侧面解决概念格的存储问题。当然，这需要大量的、系列的工作为基础，目前我们正在积极开展这方面的研究。

参考文献

1 Ganter B, Wille R. Formal Concept Analysis-Mathematical Foundations. Springer-Verlag, 1999

2 Han Jiawei, Kamber M. Data Mining: Concept and Techniques. Simon Fraser University, Canada, 2000

3 Wille R. Reconstructing Lattice Theory: an Approach Based on Hierarchies of concepts. In: Rival I, ed. Ordered Sets, Reidel, 1982. 445～470

4 Gordin R. Incremental concept formation algorithm based on Galois (concept) lattices. Computational Intelligence, 1995, 11(2): 246～267

5 Devlin K. Infosense - Turning Information to Knowledge, Freeman, New York, 1999

6 [美]卡耐基梅隆大学软件工程研究所，刘孟仁等译. 能力成熟度模型(CMM)：软件过程改进指南. 电子工业出版社，2001

7 刘宗田. 容差近似空间的广义概念格模型研究. 计算机学报，2000，23(1)

8 谢志鹏，刘宗田. 概念格与关联规则发现. 计算机研究与发展，2000，37(12)：1415～1421