数据挖掘在软件维护中的应用

苏绍勇　潘金贵

（南京大学计算机软件新技术国家重点实验室，南京大学计算机科学与技术系　南京 210093）

摘　要　软件维护是软件过程中十分重要但又难度较大的一个阶段，数据挖掘技术在软件维护中的应用大大改善了这一现状。本文从技术的适用性、一般应用步骤和主要应用领域等方面介绍了数据挖掘在软件维护中应用的发展历程，并在此基础上研究了所用到的关键技术及存在的问题。最后展望了未来的发展趋势。

关键词　数据挖掘，软件维护，程序理解，程序修改，数据预处理

Data Mining in Software Maintenance

SU Shao-Yong　PAN Jin-Gui

(State Key Lab. for Novel Software Technology, Department of Computer Science and Technology, Nanjing University, Nanjing 210093)

Abstract　Software maintenance is one of the most important and difficulty stages in software process. Fortunately, the data mining technology has greatly improved the situation of software maintenance. This paper introduces the history of the application of data mining in software maintenance with three aspects: theapplicability of the technology, the typical steps and the major application fields; and based on it we study the key issues. This paper concludes with the directions for future work.

Keywords　Data mining, Software maintenance, Program comprehension, Program changes, Data preprocessing

软件维护是软件生命周期中的最后一个阶段，也是最重要的一个阶段。要使软件在实际的、不断变化的环境中正常运行，就必须要不断修正错误、完善系统，这样才能发挥出软件的最大功效，软件维护已成为增强软件生命力的最重要途径。但是随着计算机程序愈来愈庞大、环境变化越来越频繁，加之各种文档的缺乏，软件维护变得越来越困难。平均说来，大型软件的维护成本高达开发成本的4倍左右[1]。目前国外许多软件开发机构把60%以上的精力用于维护现有软件。随着产品数量的增加和寿命的延长，版本的不断增长和更新，软件维护的工作量及成本还将不断攀升。

数据挖掘是20世纪90年代兴起的数据智能分析技术，它可以从巨量数据中获取有效的、新颖的、潜在有用的、最终可理解的模式。数据挖掘的这些特性决定了它将在软件维护领域起到积极而又有意义的作用：它可以在没有任何前提知识的情况下在大量的数据（如软件系统的源程序、各种文档等）中发现有用的信息，这些信息对软件的维护具有重要意义。数据挖掘技术还会大幅度减轻软件维护人员的工作量，把维护人员从读文档、看代码、理解程序等繁重的工作中解脱出来。

1　数据挖掘技术在软件维护中的适用性

数据挖掘在1989年8月美国底特律市召开的第十一届国际联合人工智能学术会议上正式形成。1998年De Oca在文[2]中正式提出了将数据挖掘应用于软件维护领域中，并分析了数据挖掘技术在软件维护领域的适用性以及其巨大的潜力，还给出了数据挖掘技术用于软件维护的一般步骤。此后，数据挖掘技术在软件维护领域得到了巨大发展。

将数据挖掘用于软件维护主要是因为[2]：第一，数据挖掘可以在大型数据库中发现数据元素间那些预先未知的、非平凡的关系；第二，数据挖掘技术可以在对挖掘对象没有任何预知知识的条件下获得有关信息；第三，数据挖掘适用于对大量的信息进行处理。数据挖掘的这些特点表明，在软件维护领域数据挖掘技术拥有巨大的潜力。由第一点可知，若软件系统可由数据库的形式来表示，那么就可以用数据挖掘技术来找出系统各部分之间的某些联系，也就是说数据挖掘技术可以在该系统中发现新知识。这些知识可以用来理解并维护系统。由第二点可知，即使预先不知道系统的功能及实现细节，数据挖掘技术照样可以发现系统各组件之间的联系。最后，由第三点可知，数据挖掘技术可以分析大型的软件系统而不影响其性能。事实上，软件系统的规模越大数据挖掘技术越有可能发现有用的信息。由以上分析可知，数据挖掘技术十分适合于软件维护领域，数据挖掘技术的出现将会有助于解决软件维护领域中的许多难题。

2　数据挖掘用于软件维护的过程

软件维护一般分为三步：程序理解、程序修改和重新验证程序。数据挖掘技术主要应用在程序理解和程序修改过程中。重新验证程序属于软件测试内容，较少涉及数据挖掘技术。

数据挖掘用于软件维护的过程一般也分为三个步骤[2]：

第一，数据预处理。即给软件系统定义一个数据库角度的视点。系统的数据库视点是指用数据库形式表示的系统。数据库中存储的数据来源于系统的所有信息（程序的各种文档、源代码等），数据库视点的选择也决定了数据挖掘技术所

苏绍勇　硕士研究生，主要研究领域为分布虚拟环境、软件工程；潘金贵　教授，博士生导师，主要研究领域为多媒体信息处理、多媒体远程教育系统。

能挖掘出的信息类型。数据库视点的选择是数据挖掘在软件维护领域十分关键的一步，它往往和数据挖掘算法的选择同时考虑。当然，在这一步中，并不一定是给系统定义数据库视点，根据所选用数据挖掘算法的不同，还可以有其他预处理方法，如为训练决策树预处理数据等。不过在软件维护领域中常用的数据挖掘算法主要是关联规则挖掘算法。

第二，执行数据挖掘，此过程包括选择并应用数据挖掘算法对系统的数据库视点进行挖掘，算法的选择取决于所挖掘信息的需要。

第三，评估、巩固并解释挖掘结果。对挖掘出的结果进行评估，以检验其有效性，然后再结合有关知识对软件维护工作进行指导。

2.1 数据预处理

数据挖掘工作的第一步是数据预处理，在软件维护领域对应的就是把软件系统以数据库形式表示出来。这个工作在上个世纪 90 年代初就有人开始做了。1990 年 Yih-Farn Chen 在文[3]中实现了把 C 语言源代码用关系形式表示，以支持子系统提取、绑定分析、死代码摒除、程序分层等软件活动。1992 年 Grass 在文[4]中用 CIA^{++} (C^{++} Information Abstractor)从 C^{++} 源程序中提取软件设计信息。CIA^{++} 从 C^{++} 源程序中获得信息，然后构建了一个关系数据库。这个数据库被用来恢复程序对象。1993 年，Narat 在文[5]中通过交叉引用文档来利用数据库支持对源代码的维护活动。所有上面这些工作虽然都是用数据库来表示目标系统，也都有信息的提取过程，但都没有用到数据挖掘技术，不过这些却给数据挖掘将来应用在软件维护中打下了很好的基础。

后来，随着数据挖掘技术在软件维护领域应用的深入，又出现了一些新的专门针对某些特定应用的数据预处理技术。例如：源代码属性的关系化形式表示、版本历史信息的关系化形式表示、修改记录的关系化形式表示等。

2.2 程序理解

程序理解是软件维护的第一步，也是最重要的一步，没有对目标系统的正确理解，后面的维护修改工作则无从谈起。程序理解的工作量很大，约占软件维护总工作量的一半[6]。如果出现程序员维护的是别人编写的程序、系统文档不准确、过时甚至不存在等情况，这个比例还会增长。数据挖掘技术特别适合于资料比较少甚至只有源代码的情况下进行程序理解。文档缺乏或不准确这些情况在软件维护过程中十分常见。鉴于这种情况，很多研究人员在这个问题上做了大量工作[7,8]，所以相应的数据挖掘在这方面的技术研究也比较多。

对源代码可以从几个层次上进行数据挖掘，程序理解相应也就有基于子系统层次的程序理解、基于函数层次的程序理解等多种层次上的理解。这些层次实际上是随着研究的深入，人们对程序的理解也由浅入深，从对程序整体笼统的把握到对程序更多细节的了解。下面我们主要对基于子系统层次和基于函数层次的程序理解做一个简要的介绍。

2.2.1 *基于子系统层次* 基于子系统层次的程序理解主要思想是将整个软件系统划分为几个部分，每个部分都是相对独立并且能完成一定功能的子系统，通过理解各个子系统进而了解整个系统。具体地将程序划分为几个部分的方法主要有以下两种。第一，基于关联规则的子系统划分。通过程序间共享文件的关系分割系统。ISA(Identification of Subsystems based on Association)就是这种方法的一个代表[3]。第二，基于聚类方法的子系统划分。这种方法的一个代表是基于模块依赖图(Module Dependency Graph，简称 MDG)的方法[9]。首先，它通过分析源代码构建系统的 MDG，然后根据 MDG 中模块及模块间的关系产生系统的理解视图。MDG 方法与一般方法不之处在于，它的挖掘对象不是通常的软件系统的关系数据库形式，而是关系图的形式。

这两种方法都是程序基础上的系统划分方法，它们划分所得的子系统一般包含几个程序或文件。它们都只提供一个系统的粗糙理解模型，展现给维护人员的是系统的整体轮廓，而不是系统的细节模型。不过，它们的理解模型构建的自动化过程大大节约了维护人员的人工看代码时间，进而缩短了软件维护的周期。更重要的是，这些方法的出现给研究人员指明了一个方向，即把数据挖掘技术应用到更底层的源代码上去，从而构造更为细致的理解模型，以便更大程度上帮助软件维护人员展开工作。

2.2.2 *基于函数层次* 由于基于子系统的程序理解模型不够精细，研究人员把数据挖掘的对象细化到函数层次上，提出了基于函数层次程序理解模型的数据挖掘方法。

Tjortjis 在文[10]中提出了数据挖掘代码聚集 (Data Mining Code Clustering，简称 DMCC)方法，它可以快速地从一个软件系统中提取出其基于函数的理解模型，并使该过程自动完成。DMCC 应用聚类算法，根据程序实体(如函数等)间的相似性，把程序划分为几个实体组。通过聚类算法获得的实体组可以替代系统的高层抽象(如子系统形式)对系统进行整体上的理解与把握。这种聚类获得的组可以降低程序理解的复杂度，进而有助于系统的维护。DMCC 的主要目的是提供一个系统的整体理解轮廓，而不是详细的细节模型。

Tjortjis 在文[11]中又提出了另一种方法。他把程序模块组织成〈code entity，attributes of this entity〉形式，通过关联规则挖掘算法模块间的关系，再根据这些关系将模块分组。由这种方法获得的模块组内部的模块间关系紧密，通常代表一定的程序功能，所以通过这些组获得的对程序的理解也是比较细致的，因为这些组是通过对程序模块(函数)的分析直接得来的。这种方法对单人或遵循相同规范的开发团队编写的源程序十分适用。不过这种方法也有其局限性，它不适用于多种语言编写的源代码，也不适用于多人编写的多种风格的源代码。

随着技术的不断成熟，数据挖掘在程序理解方面也出现了一些比较完善的框架模型。文[12]中提出了一种专门针对 C^{++} 程序源代码进行处理的软件框架结构。该结构首先给出了输入数据模型和数据预处理的方法，然后在此基础上构建了数据提取模块、数据转换模块、信息挖掘模块和结果评估模块。在整个系统框架中，数据的提取和转换是比较重要的，因为这两个模块负责从 C^{++} 的源代码中提取出有用的信息，并且该信息还得以数据挖掘算法能识别的形式表示出来。其余部分都具有通用性，提取出来的信息被存放在数据库中，然后用一定的数据挖掘算法和数据挖掘工具挖掘出知识，最后再对所得知识进行评估，识别出有用的知识用来帮助维护人员对程序进行理解。

综上所述，我们可以发现数据挖掘在程序理解中的应用发展存在两个趋势：第一，程序理解模型精细化；从基于子系统的理解到基于函数的理解，软件维护人员从源代码中能获得越来越多的对系统的细致理解。现在有些研究人员正在把理解模型细化到源代码语句(如循环结构、分支语句等)的层次上。第二，理解模型的系统化、规范化和自动化；数据挖掘

在软件维护中的应用从一开始的提出一些想法、算法到现在的整个程序理解过程的系统规范化，它的效率越来越高，作用也越来越大。像文[12]中提出的框架模型，其中数据挖掘部分用的是 Intelligent MinerTM，而不是自己做的专门的挖掘工具，这样研究人员可以把注意力集中在原始数据的提取和转换上，而不必关心后面的挖掘算法是否正确可靠。这样更符合软件工程的要求，既提高了效率，又增强了通用性。

2.3 程序修改

程序理解的目的就是为了进行正确的程序修改，程序修改是软件维护中十分重要的一步。有关程序修改方面的研究不少，但数据挖掘技术在这个领域应用的研究却不多，这方面还有待于更进一步的研究。程序修改领域内的数据挖掘技术与程序理解领域内的略有不同，程序修改领域内的数据挖掘对象从单版本的程序源代码扩展到多版本程序源代码、程序的修改历史记录、软件的版本历史信息、CVS 等。

软件维护人员在修改一段代码前总想知道有哪些其他的代码或文件与这段代码相关，文[13]提出了解决这类问题的一种方法。这种方法是根据文件以前是否同时被修改或查看过来判断文件间的关系，利用相关文件间的信息（如共享文件名的前缀、共享程序的个数等）建立决策树，学习有关概念。为了更精确地指导代码的修改，Annie Ying 在文[14]又提出修改类型挖掘技术，修改类型是指过去经常一起被修改的文件。这种技术比文[13]中的方法更精确的原因是它对文件一起被修改的次数做了进一步的研究，根据次数进行关联规则的挖掘。这两种方法的共同点很明显，它们都是从文件被同时修改这一角度挖掘有用的信息。这一特点也决定了这类方法的是平台无关的、编程语言无关的，也就是说进行数据挖掘的程序可以有基于不同平台的模块、可以用多种不同的编程语言编写。

与在程序理解领域中一样，数据挖掘在程序修改领域中也向着精细化方向发展。上面提到的两种方法是基于文件层次的，它们寻找的是文件之间的关联性。当一个文件被修改时，这些方法可以预测出相应可能应该被修改的文件。文[15]提出了一种基于函数层次的数据挖掘方法，这种方法把程序修改信息细化到对函数的修改，然后组织成关系元组形式进行关联规则挖掘，提取出有用信息。文[15]还提出了相应的工具软件 ROSE(Reengineering Of Software Evolution)。

2.4 其他方面

数据挖掘技术在软件维护的其他方面也有着广泛的应用，例如对遗产系统的理解和维护[13,16]、可重用模式的识别[17]、设计恢复(design recovery)、体系结构恢复(architecture recovery)、对象识别(identity objects)、再造工程(reengineering)、逆向工程(reverse engineering)等等。限于篇幅在此就不一一做详细的介绍。

3 关键技术及存在的问题

数据挖掘技术能在软件维护中广泛应用得益于几个关键技术的研究与应用，主要包括数据预处理技术、数据挖掘算法、对挖掘结果的评估。近几年这些技术都有很大发展，但仍存在一些问题。下面我们就简要介绍一下这些技术及其存在的问题。

3.1 数据预处理技术

数据挖掘一开始应用于商业，想要将之应用于软件维护，必须做必要的数据预处理。数据预处理的好坏直接影响着后续工作。目前，在软件维护领域中用到的数据预处理方法主要都是针对特定问题的，没有一套统一的规范方法。主要是根据程序代码本身的特性（如上文所提的〈code entity, attributes of this entity〉[11]）或文件之间的关系（两文件经常被同时修改[13]）构造关系形式的数据，为后面的关联规则算法或聚类算法等提供输入数据。在早期还存在基于分析的方法的数据预处理技术。

数据预处理是数据挖掘应用在软件维护中的重要一环，而且必不可少。要使挖掘算法更有效地挖掘出有用信息，就必须为它提供准确、简洁、合适的数据。然而从需维护的软件系统中能实际得到的数据通常很杂乱，从程序源代码到各种文档，从单版本的信息到各种历史数据，各种数据都有其自身的格式。杂乱的格式导致数据集成困难，所以已有的数据预处理大多针对一种或有限的几种数据进行，缺乏统一的、应用于所有数据源的数据预处理技术。

3.2 挖掘算法

挖掘算法是数据挖掘技术中最核心的环节，它的好坏直接关系着挖掘的效率和质量。在软件维护中用到的挖掘算法主要有关联规则挖掘算法、聚类算法和决策树分类算法，其中以关联规则挖掘算法和聚类算法最为常见。关联规则挖掘算法主要用来发现程序或代码之间的相关性，以便理解程序功能或指导程序修改；聚类算法主要用来将大型复杂的软件系统划分为规模较小、功能单一的子系统，以便深入理解程序的功能或作用；决策树分类算法在文[13]中有所提及，它主要是把源程序文件归为 3 类，相关类、不相关类和潜在相关类，这样在进行程序修改时可以判定哪些文件需要同时修改，哪些文件可能需要同时修改，哪些文件无需同时修改。

在软件维护中用到的所有挖掘算法基本上都是来自数据挖掘领域的通用算法，几乎没有专门针对软件维护的高效算法。今后的一个研究方向就是开发专门针对软件维护领域的高效算法。

3.3 评估标准与方法

对挖掘结果的评估是数据挖掘的最后阶段，它可以判断挖掘算法的有效性。挖掘的结果好，那软件维护人员可以十分方便地理解和修改程序，节约时间，提高效率。现在研究人员已经提出了预测度(predictability)和兴趣度(interestingness)等标准，并通过实验对他们自己提出的挖掘算法或框架进行了评估。

现在，主要的评估手段是实验，并且都是对特定算法、特定框架的应用进行评估，缺乏通用的评估标准与方法。尤其在程序理解方面，更是没有什么好的方法。程序理解本来就没有硬性指标，对程序理解的好与差没有明显的区分标准。这些方面都是今后进行研究要解决的问题。

结论与展望 通过本文的介绍我们可以看到数据挖掘已经在软件维护领域有着广泛的应用，并且取得了显著的效果，数据挖掘的应用可以在很大程度上提高软件维护的效率。但是我们也应当看到，作为一门具有广泛应用的新兴学科，数据挖掘的一般原理与软件维护领域需要的有效的数据挖掘工具之间，还存在不小的差距。在数据挖掘应用于软件维护的初期，研究人员也大多是针对特定问题进行的特定研究，提出的方法或算法缺乏通用性。并且初期的研究多局限于理论研究，没有出现软件维护领域的数据挖掘软件或系统原型。数据挖掘技术在软件维护领域应用的进一步发展方向主要有以下几个方面：

第一，寻找适合于软件维护领域的灵活的数据预处理技术，从数据源中获得尽可能多的信息，并以标准的形式提供给数据挖掘算法。灵活的数据预处理技术可以根据不同的应用（如程序理解、程序修改等）处理不同形式（如程序文档、程序源代码等）、不同粒度（如程序源代码中的文件、函数、语句等）的原始数据。

第二，寻找适合于软件维护领域不同应用的高效挖掘算法，进一步加强数据挖掘在软件维护领域的应用。高效的挖掘算法是指可以根据不同的需要快速、准确地挖掘出有用的信息。

第三，制定统一有效的评估标准，进一步加强对挖掘结果的评估。

第四，构建适用于软件维护领域的、支持多种数据挖掘功能同时每一种功能又支持多种方法的自动化数据挖掘工具，加大数据挖掘技术在软件维护领域的实际应用。

参 考 文 献

1 陈世鸿，朱福喜，等．软件工程原理及应用．武汉：武汉大学出版社，2000. 243～252

2 Oca C M D，Carver D L. Identification of Data Cohesive Subsystems Using Data Ming Techniques. In：Proc. of the Intl. Conf. Software Maintenance，1998. 16～23

3 Chen Y-F，Nishimoto M Y，Ramamoorty C V. The C Information Abstraction System. IEEE Transaction on Software Engineering，1990，16(3)：325～334

4 Grass J E. Object-Oriented Design Archaeology with CIA＋＋. Computing Systems：The Journal of the USENIX Association，1992，5(1)：5～67

5 Narat V. Using a relational database for software maintenance：a case study. In：Proc. of the IEEE Conf. on Software Maintenance，1993. 244～251

6 Grubb P，Takang A A．Software Maintenance：Concepts and Practice. Second Edition，Hackensack：World Scientific Publishing，2000

7 李莹，张琴燕．程序理解．计算机应用研究，2001(6)：40～43

8 李必信，郑国梁，李宣东，张勇翔，梁佳．软件理解研究与进展．计算机研究与发展，1999，36(8)：897～906

9 Mancoridis S，et al. Using Automatic Clustering to Produce High-Level System Organizations of Source Code. In：Proc. of the 6th Intl. Workshop Program Understanding，1998. 45～53

10 Tjortjis C，Layzell P J. Using Data Mining to Assess Software Reliability. In：Proc. of the 12th Intl. Symposium Software Reliability Engineering，2001. 221～223

11 Tjortjis C，Sinos L，Layzell P. Facilitating Program Comprehension by Mining Associated Rules from Source Code. In：Proc. of the 11th Intl. Workshop Program Comprehension，2003. 125～132

12 Kanellopoulos Y，Tjortjis C. Data Mining Source Code to Facilitate Program Comprehension：Experiments on Clustering Data Retrieved from C＋＋ Programs. In：Proc. of the 12th Intl. Workshop Program Comprehension，2004. 214～223

13 Shirabad J S，Lethbridge T C，Matwin S. Supporting Maintenance of Legacy Software with Data Mining Techniques. In：Proc. of the 17th IEEE Intl. Conf. on Software Maintenance，2001. 22～31

14 Ying A T T，Murphy G C，Ng R，Chu-Carroll M C. Predicting Source ode Changes by Mining Change History. IEEE Transaction on Software Engineering，2004，9：574～586

15 Zimmermann T，Weigerber P，Diehl S，Zeller A. Mining Version Histories to Guide Software Changes. In：Proc. of the 26th Intl. Conf. on Software Engineering，2004

16 Shirabad J S，Lethbridge T C，Matwin S. Mining the Maintenance History of a Legacy Software System. In：Proc. of the Intl. Conf. on Software Maintenance，2003

17 Michail A. Data Mining Library Reuse Patterns using Generalized Association Rules. In：Proc. of the 22nd Intl. Conf. on Software Engineering，2000

（上接第 218 页）

行环境都具有多样性，所以其开发平台和运行平台应该同时构件化并要有机联系起来，才能最适合嵌入式软件的特点。为此，本文提出了 CBMESP 模型，它同时关注了开发平台与运行平台的构件化，并将二者的构件模型统一起来，从而使其可以应用于各种嵌入式领域而不必更改该模型，只需调整构件库中的具体构件即可，具有普遍适用性。这样，CBMESP 不但加强同一领域内，也加强了领域之间的重用性（包括共享整个模型架构、基于该模型的开发过程与方法以及构件库中的具体构件）；CBMESP 强调并提供了开发平台与运行平台（应用软件）统一的基于构件的定制方式，更好满足了嵌入式软件开发的多样性要求；CBMESP 根据嵌入式软件特点提出构件模型由三个可以独立实现和运行的部分组成，并解决了各部分之间信息的传递问题，较好适应了嵌入式软件的交叉开发过程和嵌入式系统资源有限的特点。在文章最后，我们用实际例子说明了 CBMESP 模型的特点。

下一步的研究工作包括不断地充实我们的构件库以及开发基于其它平台的构件容器，使构件容器具有分布式处理能力，以便使本模型提出的规范能够应用到更为广泛的领域，以便提高相应领域的嵌入式软件开发效率。

参 考 文 献

1 WindRiver corporation. http://www.windriver.com/products/platforms/consumer_devices

2 OSEK Group：OSEK/VDX Network Management-Concept and Application Programming Interface. Version 2.51，2000

3 QUALCOMM Incorporated：http://brew.qualcomm.com/brew/en

4 杨芙清，王千祥，梅宏，陈兆良．基于复用的软件生产技术．中国科学(E辑)，2001，31(4)：363～371

5 Rogerson D. Inside COM. Microsoft Press，1997

6 Schmidt D C，Kuhns F. An Overview of Real Time CORBA Specification. IEEE Computer，2000，33(4)：56～63

7 Verral M S，Morgan L. Tool Integration in CASE Environments：the Software Bus. Computer-Aided Software Engineering，Fifth International Workshop，1992．46～49

8 Henning M，Vinoski S. C++ Based CORBA Advanced Programming. 北京：清华大学出版社，2000