

文章编号:1008-1542(2012)01-0069-05

基于 WEKA 的软件维护性评价模型的实验研究

刘 丽^{1,2}, 朱小冬^{1,3}, 叶 飞¹

(1. 军械工程学院装备指挥与管理系, 河北石家庄 050051; 2. 石家庄机械化步兵学院教研部, 河北石家庄 050003; 3. 武汉大学计算机学院, 湖北武汉 430072)

摘 要:为了能够对软件维护性进行评价,以类级软件维护性数据集为例进行了软件维护性实验,应用 Logiscope 对类的维护性做了定性评价;运用 Krakatau Professional 收集类各个度量的度量值;通过 WEKA 得到了类级度量关键属性;最后,选取决策树分类方法训练分类模型,并利用训练好的分类模型对测试集进行测试,得到基本满足要求的类级软件维护性评价模型。

关键词:软件维护性评价;Logiscope;Krakatau Professional;WEKA;决策树分类器

中图分类号:TP301 文献标志码:A

Experimental research in evaluation model of software maintainability based on WEKA

LIU Li^{1,2}, ZHU Xiao-dong^{1,3}, YE Fei¹

(1. Department of Equipment Command and Management, Ordnance Engineering College, Shijiazhuang Hebei 050051, China; 2. Department of Teaching and Research, Shijiazhuang Mechanized Infantry Academy, Shijiazhuang Hebei 050003, China; 3. School of Computer, Wuhan University, Wuhan Hubei 430072, China)

Abstract: In order to evaluate software maintainability, an experiment was made about class dates of software maintainability. Class maintainability was qualitatively evaluated by applying Logiscope. Some class metrics were collected by using Krakatau Professional, metric attributes were analysed through WEKA and the key attributes were obtained. At last, professional class model was trained through decision tree classifier, and the model was tested.

Key words: evaluation of software maintainability; Logiscope; Krakatau Professional; WEKA; decision tree classifier

作为软件产品的重要质量特性,可维护性是软件开发阶段各个时期的关键目标,决定了软件可以被更改以满足用户要求或能检测到缺陷并予以改正的难易程度,其评估目的是确认软件系统的维护性是否达到设计或标准的要求。对软件可维护性进行度量和评价不仅有利于了解软件是否满足规定的可维护性要求,而且有助于及时发现设计缺陷,指导软件可维护性的分析与设计。

软件的最终产品是由源代码编译成的可执行文件,对软件的维护最终要落到对源代码的更改上。因此,对软件实现及实现后的基于源代码的软件维护性评价也是软件维护性评价的一个重要研究内容。通过基于源代码的软件维护性评价可以了解软件整体的维护性水平,并发现维护性薄弱环节,进而有针对性地改进源代码质量和维护性,最终保证软件具有良好的维护性。基于此,笔者试图对软件源代码的维护性进行评价。

笔者以面向对象软件的类维护性评价为对象,对收集的 300 个类级源代码(C++)进行了维护性评价实验,首先应用 Logiscope 依据 ISO 9126 软件质量模型对类的维护性进行了定性评价,对各类的维护性给

收稿日期:2011-10-13;修回日期:2011-11-11;责任编辑:王海云

基金项目:装备预先研究项目

作者简介:刘 丽(1980-),女,河北鹿泉人,讲师,博士研究生,主要从事装备软件保障与应用方面的研究。

出一个综合的优、良、可、差的结果。其次由于 ISO 9126 软件质量模型不易指导软件可维护性分析与设计,因此从规模、耦合度、内聚度、继承、复杂度 5 方面共选择了 21 个源代码度量集,运用 Krakatau Professional 对类的各个度量值进行度量,然后,考虑到度量数据过多,如果度量集都用来构造软件维护性评价模型,会导致构造模型的效率和模型的准确度严重下降,因此,提出了进行关键属性选择。最后,选取决策树分类方法训练分类模型,并利用训练好的分类模型对测试集进行测试,建立了基本满足要求的类级软件维护性评价模型,运用该评价模型可对系统中的各个类进行评价,发现系统维护性的薄弱环节——维护性差的类。

1 软件工具简介

1.1 Logiscope

Logiscope 是法国 Telelogic 公司推出的专用于软件质量保证和软件测试的产品。其主要功能是对软件做质量分析和测试以保证软件的质量,并可做认证、反向工程和维护,特别是针对要求高可靠性和高安全性的软件项目和工程。本文用的是 Logiscope 的 Audit 工具,它可以定位错误模块,可评估软件质量及复杂程度,对系统各类的维护性进行定性评价。

1.2 Krakatau Professional

Krakatau Professional 是一款软件度量工具,是为源代码质量而设计,是度量软件工具中的专家。该工具可对面向对象软件系统的各种特性进行度量。通过分析软件结构,在方法、类、模块和系统级上对软件维护性度量进行选择,并提供可视化的报告和图形化的输出。

1.3 WEKA

WEKA 是基于 JAVA 环境下开源的机器学习(machine learning)以及数据挖掘(data mining)软件。本文选用的是 WEKA 的 Explorer 数据处理模块,在此模块环境中,WEKA 提供了数据的预处理、数据格式的转化(从 CSV 格式到 ARFF 格式的转化)、各种数据挖掘算法(包括分类与回归算法、聚类算法、关联规则等),并提供了结果的可视化工具。

2 实验方法与流程

2.1 度量集

评估软件可维护性,指标的合理选取至关重要。本文结合 ISO 9126 的软件维护性定义,对软件可维护性进行分类(如图 1 所示)。从规模、耦合度、内聚度、继承、复杂度 5 方面共选择了 21 个源代码度量集^[1-4](如表 1 所示)。

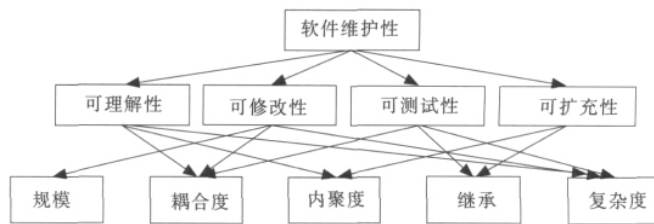


图 1 软件可维护性分类图

Fig. 1 Classification of software maintainability

2.2 软件维护性评价实验流程

本文对软件进行维护性评价,遵循图 2 所示的流程图。

2.2.1 数据预处理

数据预处理包括数据准备以及关键属性选择。

1) 数据准备

要构建维护性评价模型,需要获得 2 组数据,一组是关于软件本身设计特性的数据,它评价模型中的自变量,另一组数据是软件维护性数据,它是模型中的变量。本文是通过表 2 所描述的维护性实验来获取软件维护性数据集的。

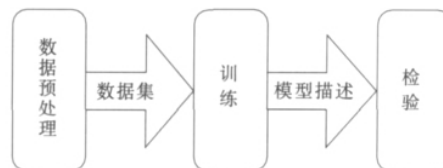


图 2 软件可维护性评价流程图

Fig. 2 Flow of software maintainability evaluation model

表 1 类级面向对象软件系统源代码度量集

Tab. 1 Class level OO software source code metrics set

序号	度量名称	度量定义及说明
1	NPavgC	平均方法参数数量(average number of method parameters)
2	OSavg	平均操作规模(average operation size)
3	CSAO	类的规模大小(包括属性和操作)class size(attributes&-operations)
4	CSO	类的规模大小(操作)class size(operations)
5	CSI	类的特殊索引(class specialisation index)
6	CBO	对象与类之间的耦合(coupling between object classes)
7	DIT	继承树的深度(depth of inheritance tree)
8	LCOM	缺少内聚方法(lack of cohesion methods)
9	LOC	代码行数(lines of code)
10	NAAC	增加的属性数量(number of attributes added)
11	NAIC	继承的属性数(number of attributes inherited)
12	NOAC	增加操作数(number of operations added)
13	NOIC	继承操作数(number of operations inherited)
14	NOOC	重复命令数(number of operations overridden)
15	NOCC	子类数(number of child classes)
16	PPPC	公有类/保护类的方法占的比例(percentage public/protected methods)
17	PA	私有属性的用法(private attribute usage)
18	RFC	类的响应(response for class)
19	SLOC	源代码行数(lines of source code)
20	TLOC	总代码行数(total lines of code)
21	WMC	类中方法占的比例(weighted methods in class)

表 2 面向对象软件维护性实验描述

Tab. 2 Maintainability experiment(OO source code)

项 目	描 述
实验名称	面向对象软件维护性实验
实验目的	获取面向对象软件源代码度量及维护性数据
实验对象	对 300 个类级的源代码(C++)进行度量及维护性定性评价,这些系统大多数来自于网上的开源项目
实验工具	Logiscope, Krakatau Professional
实验步骤	1)用 Logiscope 对系统各类的维护性进行定性评价,给出一个综合的优、良、可、差的结果; 2)用 Krakatau Professional 对源代码进行分析,给出各个度量值(见表 1),得到类级源代码的设计特性
实验输出	类级维护性定性评价数据集及类级源代码度量数据集

通过上述实验,其部分输出结果见表 3。

表 3 部分结果

Tab. 3 A part of the output

类名称	类维护性	CSA	CSAO	CSO	DIT	LCOM	LOC	NAAC	RFC	SLOC	TLOC	WMC
CRobotViewApp	优	0	3	3	0		20	0	3	9	77	4
CBlockDlg	良	2	5	3	5	0	30	2	3	13	46	3
CBlockView	中	0	1	1	7		10	0	1	7	258	1
CAboutDlg	优	0	2	2	0		23	0	2	10	34	2
CDBParamater	优	4	6	2	9	0	32	4	2	14	52	2
CMainFrame	中	2	11	9	12	0	42	2	13	26	123	15
CDlgNewRecord	优	2	6	4	21		20	2	4	14	20	4

2) 关键属性选择

选择关键属性来建造软件维护性评价模型是本文建模中的一个重要问题,由于维数灾难,所列的度量集不可能都被用来进行软件维护性评价模型的训练。如果度量集都用来构造软件维护性评价模型,会导致构造模型的效率和模型的准确度严重下降。在收集的类的软件维护性度量之中,有太多的度量需要学习方案进行处理,其中部分属性是无关或者重复的。因此,需要对属性进行选择,选择出关键属性,消除高度相关的其他属性运用于学习中。

本文选用 WEKA 中的 CfsSubsetEval 评估器和多种搜索算法,属性选择的模式采用将数据全集作为训练和测试集, classperformance 属性作为分类属性,对类的度量数据集进行处理。CfsSubsetEval 评估器可逐一评估每个属性的预测能力和它们之间的重复程度,进而选择那些与类有高度关联但相互之间关联程度却较低的属性。根据数据特点及搜索算法与属性子集评估器的匹配原则,选取了 WEKA 工具中集成的属性空间搜索方法中的 5 种搜索算法进行了实验。它们分别是: BestFirst 算法、Greedy Stepwise 算法、ExhaustiveSearch 算法、GeneticSearch 算法和 RandomSearch 算法。由于篇幅有限,仅给出采用 BestFirst 算法对属性子集进行搜索的结果显示,如图 3 所示。

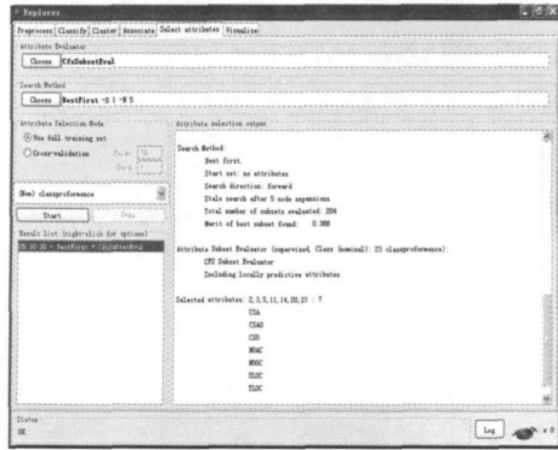


图 3 BestFirst 算法属性选择结果

Fig. 3 Outcome of attribute choice base on BestFirst

综合以上 5 种算法搜索属性子集的结果,可知第 2, 3, 4, 14, 16, 19(对应的是 OSavg, CSAO, CSO, NOOC, PPC, SLOC)项属性是关键属性,其在软件维护体系中,占更重要的作用。

2.2.2 软件维护性评价模型建立与检验

本文采用 C4.5 决策树算法^[5-8]来构建软件维护性评价模型。C4.5 算法是目前应用最为广泛的机器学习算法。该算法可以描述成从一个假设空间中搜索一个拟合训练样例的假设。被算法搜索的假设空间就是可能的决策树的集合。该算法以一种从简单到复杂的爬山算法遍历这个假设空间,从空的树开始,然后逐步考虑更加复杂的假设,目的是搜索到一个正确分类训练数据的决策树。它通过 2 个步骤来构造决策树:决策树的生成阶段和剪枝阶段。

1) 决策树生成阶段

决策树生成阶段,即建立一个模型。建立过程可简述如下:从根节点开始在每个节点上按照给定标准选择测试属性,然后按照相应属性的所有可能取值向下建立分枝、划分训练样本,直到一个节点上的所有样本都被划分到同一个类,或者某一节点中的样本数量低于给定值为止。此阶段需要有一个训练集作为输入,因此,将前 270 个实例作为一个训练集来训练分类器。首先将训练数据集载入 WEKA 下的 Explorer 模块,去掉 class-name 一列属性,然后选用 J48 算法(它是 C4.5 决策树分类算法在 WEKA 中的实现),模型测试模式选择 10 折交叉验证(将初始数据集划分为 10 个互不相交的折,每个折的大小大致相等,训练和测试进行 10 次迭代),分类属性为 classperformance,如图 4 所示。点击“Start”开始训练模型,最后得到模型并保存模型,其图形化形式如图 5 所示。该图是对软件维护性进行分类,其中每个内部节点(矩形框)代表对某个属性的一



图 4 模型训练算法选择及参数设置

Fig. 4 Arithmetic of choice and setup of parameters

270 个实例作为一个训练集来训练分类器。首先将训练数据集载入 WEKA 下的 Explorer 模块,去掉 class-name 一列属性,然后选用 J48 算法(它是 C4.5 决策树分类算法在 WEKA 中的实现),模型测试模式选择 10 折交叉验证(将初始数据集划分为 10 个互不相交的折,每个折的大小大致相等,训练和测试进行 10 次迭代),分类属性为 classperformance,如图 4 所示。点击“Start”开始训练模型,最后得到模型并保存模型,其图形化形式如图 5 所示。该图是对软件维护性进行分类,其中每个内部节点(矩形框)代表对某个属性的一

次检测,一条边代表测试的结果,每个叶子节点(椭圆框)代表一个类。最上面的节点是根节点。

决策树符合 IF-THEN 规则,如下所示。

Rule 1: IF CSAO ≤ 9 and NOOC > 0 THEN 良

Rule 2: IF CSAO > 9 and CSO > 15 THEN 中

Rule 3: IF CSAO > 9 and CSO ≤ 15 THEN 良

Rule 4: IF CSAO ≤ 9 and NOOC ≤ 0 and PPPC ≤ 41.22 and OSavg > 0 THEN 良

Rule 5: IF CSAO ≤ 9 and NOOC ≤ 0 and PPPC > 41.22 and CSAO ≤ 7 THEN 优

Rule 6: IF CSAO ≤ 9 and NOOC ≤ 0 and PPPC > 41.22 and CSAO > 7 THEN 良

Rule 7: IF CSAO ≤ 9 and NOOC ≤ 0 and PPPC ≤ 41.22 and OSavg ≤ 0 and SLOC ≤ 8 THEN 优

Rule 8: IF CSAO ≤ 9 and NOOC ≤ 0 and PPPC ≤ 41.22 and OSavg ≤ 0 and SLOC > 8 THEN 中

2) 剪枝阶段

利用训练好的分类模型预测软件维护性,来检验预测的准确度。此阶段需用非训练集的实例检验,因此,将后 30 个实例用来测试预测结果的准确度。通过选择图 4 中的“Supplied test set”找到测试集的路径,将包含 30 个实例的测试集载入到模型中。得到预测输出与测试数据集中的分类的不同,对比可得其预测的准确率为 80%,基本达到了对软件维护性评价的目的。运用该评价模型可对系统中的各个类进行评价,发现系统维护性的薄弱环节——维护性差的类。

3 结 语

通过维护性实验获取了现阶段源代码度量的数据以及可维护性的关键属性,利用这些数据构建了基于类级源代码度量的软件可维护性评价模型,模型经过检验表明,达到了对软件维护性评价的目的。运用该评价模型可对系统中的各个类进行评价。

在下一步的研究中,将尝试对系统级和模块级的维护性进行评价;并运用多种方法对数据集进行分析和建模,以建立最优的软件维护性评价模型。

参考文献:

[1] THWIN M M T, QUAH T S. Application of neural networks for software quality prediction using object-oriented metrics[J]. Journal of Systems and Software, 2005, 76(2): 147-156.

[2] LI W, HENRY S. Object-oriented metrics that predict maintainability[J]. Journal of Systems and Software, 1993, 23(2): 111-122.

[3] BANDI R K, VAISHNAVI V K, TURK D E. Predicting maintenance performance using object-oriented design complexity metrics[J]. IEEE Transactions on Software Engineering, 2003, 29(1): 77-87.

[4] FENTON N E, NEIL M. Software measurement: Uncertainty and causal modeling[J]. IEEE Software, 2002, 19(4): 116-122.

[5] 张云涛, 龚 玲. 数据挖掘原理与技术[M]. 北京: 电子工业出版社, 2004.

[6] TODOROVSKI L, DZEROSKI S. Combining multiple models with meta decision trees[J]. Principles of Data Mining and Knowledge Discovery, 2000, 19(10): 69-84.

[7] YAO Y, FU Z L, ZHAO X H, et al. Combining classifier based on decision tree[A]. Information Engineering, 2009[C]. Taiyuan: ICIE'09 WASE International Conference, 2009. 37-40.

[8] 李 萍, 李法朝. 基于决策树的知识表示模型及其应用[J]. 河北科技大学学报(Journal of Hebei University of Science and Technology), 2009, 30(2): 87-91.

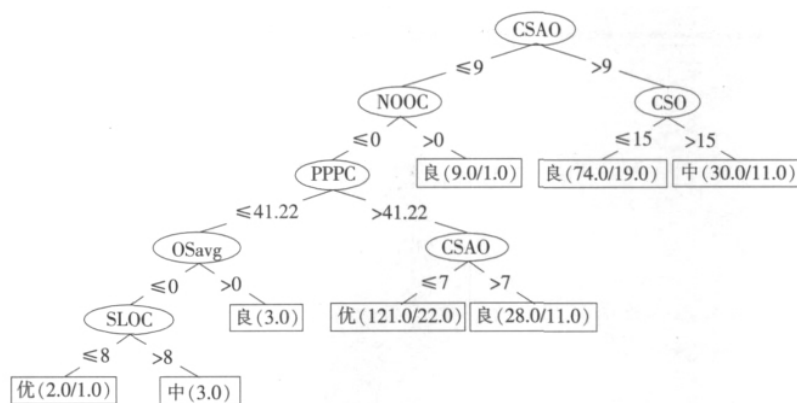


图 5 决策树分类模型的图形化表示

Fig. 5 Graphic show of decision tree classifier model