基于软件演化历史识别并推荐重构克隆的方法

折蓉蓉 张丽萍
(内蒙古师范大学计算机与信息工程学院 呼和浩特 010022)

摘 要 现有克隆代码重构研究局限于单一版本的静态分析,忽略了克隆代码的演化过程,这导致在克隆代码重构决策方面缺乏有效的方法。因此文中首先从克隆检测、克隆映射、克隆家系以及软件维护日志管理系统中提取与克隆代码密切相关的演化历史信息;其次识别出需要重构的克隆代码,同时识别出跟踪的克隆代码,然后提取与重构相关的静态特征和演化特征,并构建特征样本数据库;最后对比多种机器学习的方法对,选出效果最佳的分类器推荐重构克隆。在7款软件近170个版本上进行的实验表明,推荐重构克隆代码的准确度达到90%以上,这为软件开发和维护人员提供了更加准确、合理的代码重构建议。

关键词 克隆代码,克隆重构,克隆跟踪,克隆家系,特征提取
中图法分类号 TP311.5 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.08.037

Method for Identifying and Recommending Reconstructed Clones Based on Software Evolution History

SHE Rong-rong ZHANG Li-ping
(College of Computer and Information Engineering, Inner Mongolia Normal University, Hohhot 010022, China)

Abstract The research on the existing clone code reconstruction is limited to a single version of static analysis while ignoring the evolution process of the cloned code, resulting in a lack of effective methods for reconstructing the cloned code. Therefore, this paper firstly extracted the evolution history information closely related to the clone code from clone detection, clone mapping, clone family and software maintenance log management system. Secondly, the clone code that needs to be reconstructed was identified, and the traced clone code was identified at the same time. Then, static features and evolution features were extracted and reconstructed and a feature sample database was built. Finally, a variety of machine learning methods were used to compare and select the best classifier recommended reconstruction of clones. In this paper, experiments were performed on nearly 170 versions of 7 software. The results show that the readiness for reconstructing cloned code is more than 90%. It provides more accurate and reasonable code reconstruction suggestions for software development and maintenance personnel.

Keywords Code clone, Clone refactoring, Clone tracking, Clone family, Feature extraction

1 引言

面对大量克隆代码在软件中存在的复杂性,仅检测出这些克隆代码并不能降低软件维护成本。因此,基于克隆检测的结果,研究者们又开展了克隆管理、克隆重构等方面的研究。克隆管理[1]就是采用多种技术,有效利用有益克隆而尽量避免有害克隆,或者充分利用克隆代码的积极作用来降低它的负面影响。为了有效管理克隆代码,重构逐渐成为人们关注的焦点。重构是在保持软件外部行为的前提下,通过调整软件的内部结构,来提高软件的可理解性、可维护性,并降低其维护成本[2]。研究表明,重构与软件的质量(如可维护性、稳定性、健壮性)有着密切的联系,经过重构的克隆代码往往比未经过重构的克隆代码具有更高的质量,因此重构对于保证软件质量有着重要的理论意义和应用价值。

现有研究表明[3],重构软件系统中所有的克隆代码是不切实际的,并且也不是所有的克隆代码都需要重构。因此,在软件进化的过程中,对克隆代码的重构要区别对待,不能一概而论。比如克隆代码经过若干次修改之后仍然出现同样的错误,这种情况表明这些共同变化的克隆片段可能质量较差,经过若干次修改后仍然没有得到改善,因此这类克隆代码需要重构。再比如,有些克隆代码在演化过程中存在无变化或独立演化的情况,这种情况对软件质量的影响也很小,这表明克隆代码并不适合重构。盲目地重构可能会影响软件中其他有益的代码,导致软件质量下降,因此,在对克隆代码进行有效

到稿日期:2018-06-26 返修日期:2018-10-19 本文受国家自然科学基金资助项目(61462071),内蒙古自然科学基金资助项目(2018MS06009),内蒙古教育厅资助项目(NJZY17049),内蒙古师范大学科研基金项目(2016ZRYB003)资助。
折蓉蓉(1991—),女,硕士生,主要研究方向为软件工程、软件分析;张丽萍(1974—),女,硕士,教授,CCF会员,主要研究方向为软件工程、软件分析,E-mail:cieczlp@imnu.edu.cn(通信作者)。

维护前识别出适合重构的克隆代码尤为关键。

综上所述,自动识别需要重构的克隆代码,将对软件中克隆代码的维护和管理有重要意义。本文使用机器学习的方法对需要重构的克隆代码进行推荐,从而能够将需要重构的克隆代码准确地呈现给用户。这将为软件开发和维护人员提供有价值的参考信息,从而能够降低维护成本,提高软件代码质量,获得更大的经济效益。

2　相关概念

2.1　克隆检测

克隆检测(Clone Detection)是指查找源代码中的克隆代码,并以克隆对(Clone Pair)或克隆群(Clone Group)的形式表现出来。克隆对是指相互之间存在克隆关系的代码段;克隆群是一些克隆代码段的集合,其中任意两个代码段都是克隆对。目前,克隆检测领域的研究已较为成熟,许多检测方法及技术被提出,主要包括:基于文本的克隆检测方法[4]、基于 Token 的克隆检测方法[5]、基于抽象语法树(Abstract Syntax Trees,AST)的检测方法[6]、基于程序依赖图(Program Dependence Graphs,PDG)的检测方法[7]、基于低级语言的检测方法[8]、基于 Metrics 的检测方法[9]等。

本实验小组在克隆检测方面也进行一些研究:张久杰等[10]实现了基于 Token 编辑距离的检测方法,该方法能有效检测 Type-3 克隆代码,解决了克隆检测中 Type-3 类型克隆检测的难题,也解决了以往对克隆代码分析的研究对象大多局限于 Type-1 和 Type-2 类型的问题,并为后续研究克隆代码的演化、克隆代码的性能分析提供了更加全面、准确的数据来源。

2.2　克隆映射

克隆映射(Clone Mapping)是对软件版本间的两个克隆建立映射关系,其映射条件是具有映射关系的两个克隆具有同源性,反映了克隆由前一版本到当前版本的进化过程。

目前,构建克隆映射的方法主要有 5 类:1)基于修改日志的方法[11],从版本控制系统(Subversion,SVN)中提取系统修改日志,计算出版本之间的变化情况,从而获得版本之间的映射关系;2)基于文本和位置的方法[12],依据文本相似度以及位置关系进行映射;3)基于 CRD(Clone Region Descriptors,CRD)的方法[13],根据克隆区域在代码库中的位置得到克隆代码的区域特征,然后根据区域特征和克隆区域在克隆代码库中的位置进行匹配并建立克隆映射;4)基于主题建模的方法[14];5)基于增量和提交事务的方法[15]。

本实验小组在克隆映射方面开展了一些研究。葛广帅等[16]提出了一种基于 LDA 和 DBSCAN 的软件多版本克隆群映射方法,通过计算克隆群之间的 JS 距离,利用 DBSCAN 建立了多版本克隆群映射。此映射工作将为后期软件多版本间的演化模式识别提供准确信息。

2.3　克隆演化

克隆演化(Clone Evolution)表示克隆在软件系统演化过程中表现出来的模式和特征。克隆代码会依据软件的演化而增加、修改和删除,因此克隆代码并不是静态的。在分析克隆及克隆带来的影响时,不仅要研究软件的单一版本中的克隆,还要研究克隆代码以时间为序的演变关系[17]。

Kim 等[18]提出的克隆家系(Clone Genealogy)框架最具代表性,通过克隆家系可以描述克隆群在软件的多个演化版本中的变化情况,他们解释了直系克隆(Lineage Clone)和克隆家系的概念,同时将克隆群的演化模式分为了 6 种。Saha 等[19]对 Kim 的研究工作做了进一步的完善和扩展,开发了一款克隆家系提取器 gCad,提高了克隆代码的映射速度和准确度。此外,CiMeng 等[20]提出了基于 CRD 克隆群映射的克隆家系构建方法,此方法结合文本相似度和位置重叠率来获取克隆映射关系,同时改进了演化模式,并最终提取出克隆家系。

本实验小组在克隆演化方面开展了一些研究。葛广帅等[21]提出了一种改进的克隆代码演化模式识别方法,该方法通过使用主题概率模型来计算代码位置重叠率及文本相似度等,根据代码位置重叠率及文本相似度建立映射并构建图模型,根据生产形式为克隆群标注短期演化模式,从而使用广度优先搜索算法提取克隆家系。这些工作为后续克隆代码的质量评估、有害性分析以及重构推荐都奠定了基础。

2.4　克隆重构

重构(Refactor)的概念由 Opdyke 等[22]提出,其指通过改变程序代码的内部结构而不改变程序代码的外部行为来提高代码的质量。在软件开发过程中,为提高开发效率,软件开发人员经常会复用已有的代码,这类代码通常都需要修改,而修改可能涉及大量的编辑工作,并且任何遗漏或错误的编辑都有可能引起 Bugs。因此亟需一种有效的方法或工具来帮助软件开发或维护人员修改此类代码,故在克隆代码的检测与映射的基础之上,克隆代码的重构与跟踪逐渐成为一个热点。

边奕心等[23]用一种计算简便的方法提取了适用于重构的克隆代码。Bakota 等[24]基于进化分析的方法提取了适用于重构的克隆代码,此方法适用于 Type-1,Type-2 克隆代码。Mondal 等[25]根据 SPCP 克隆代码中克隆片段的位置变化将其分为重构数据集和跟踪数据集。Higo 等[26]提出了一种基于克隆度量来提取克隆代码进行重构的方法。

本实验小组在克隆代码重构方面也进行了一些研究。刘冬瑞等[27]提出一了种克隆代码可重构性评估方法,该方法系统地评价了当前版本中克隆代码的可重构性,并按照克隆代码的重构等级将其从低到高进行排序。折蓉蓉等[28]提出了一种基于决策树推荐克隆重构的方法,该方法能够为克隆代码的重构识别提供技术支持和数据参考。这些工作为识别和推荐重构克隆代码奠定了基础。

国内外关于克隆代码的重构研究已经取得了一定的成果,但是目前研究大多局限于单一版本的克隆代码,很少有研究从软件演化历史方面对克隆代码加以分析,然而要更好地识别并推荐重构克隆,必须对克隆代码的演化历史展开更深

入的讨论。结合克隆检测、克隆映射、克隆家系和维护提交日志等理论和技术，同时根据克隆代码演化历史识别出需要重构的克隆代码，本文使用多种机器学习的预测模型来推荐重构克隆代码，同时选出效果最佳的机器学习方法进行预测。该方法能够提高克隆代码重构的效率，并确保重构的质量和安全，增强软件的可维护性和可理解性。这一研究的展开将为相关的程序开发和维护人员提供有力支持。

3 基于软件历史识别与推荐克隆重构的方法

本文提出的基于软件历史识别与推荐克隆重构的方法主要分为 4 个步骤：1)获取克隆演化历史信息；2)识别重构候选集与跟踪候选集；3)构建特征数据集；4)推荐用于重构的克隆。整个方法的流程如图 1 所示。

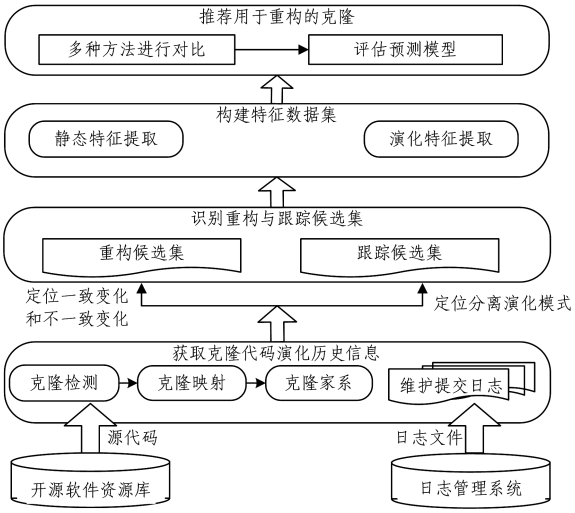

图 1 整体工作的流程图

Fig. 1 Fowchart of overall work

3.1 克隆代码演化历史信息的获取

3.1.1 检测克隆代码

克隆代码检测是研究克隆重构的基础，克隆检测可检测克隆代码以及克隆代码所在的位置，为识别重构克隆代码提供数据来源。因此本文使用目前比较成熟的克隆检测工具 NiCad 进行克隆检测，NiCad 是由加拿大皇后大学的 Roy 等[4]开发的克隆检测工具，其当前最新版本为 NiCad 4.0，适用于 C，Java，Python 等主流开发语言，能够检测 Type-1，Type-2，Type-3 的克隆代码，具有较高的查全率和查准率，而且时空复杂度很低。

本文使用 subversion 从版本管理器中获取某次提交日志结果作为一个版本的源码，然后使用 NiCad 进行克隆检测。检测结果以克隆对、克隆群和克隆函数的形式反馈出来，本研究只用克隆函数、克隆群形式的检测结果，并将其存储到 XML 文件中。下面以 ffmpeg 软件为例，介绍克隆检测结果。图 2 呈现了以克隆函数的形式反馈的部分内容检测结果，即一个代码片段的文件，其提供了每一个代码片段的文件路径、起始行号、结束行号、片段源码等信息。图 3 呈现了以克隆群形式反馈的部分检测结果，即克隆群文件，其中包含克隆片段和检测出来的克隆群信息。

```
〈sources〉
…
〈source file="/home/rongrong/…/KeyBindingsDialog.C" startline="73"
endline="78"〉
{
  Box topBox=new Box(BoxLayout.X_AXIX);
  topBox.add(new JLable("Binding:",JLabel.RIGHT));
  topBox.add(keyTF);
  getContentPane().add(topBox,BorderLayout.NORTH)
}
〈/source〉
…
〈/sources〉
```

图 2 以克隆函数形式反馈的部分检测结果

Fig. 2 Partial test results fed back in the form of a clonal function

```
〈/clones〉
…
〈class classid="13" nclones="4" nlines="20" similarity="80"〉
〈source file="/…/SimpleSearchRule.C" startline="95" endline="114"
pcid="983"〉〈/source〉
〈source file="/…/SimpleSearchRule.C" startline="140" endline="159"
pcid="993"〉〈/source〉
〈source file="/…/SimpleSearchRule.C" startline="161" endline="180"
pcid="988"〉〈/source〉
〈source file="/…/SimpleSearchRule.C" startline="118" endline="138"
pcid="988"〉〈/source〉
〈/class〉
…
〈/clones〉
```

图 3 以克隆群形式反馈的部分检测结果

Fig. 3 Partial test results fed back in the form of clonal groups

3.1.2 克隆代码映射

对软件版本间的克隆代码建立映射是追踪克隆的核心技术。如何计算这些相似关系，找出相似度最大的克隆实例并对它们建立映射关系，是克隆映射过程需要解决的问题。

本文采用了一种结合词频向量计算、克隆位置距离关系和克隆特征的分层映射方法[17]。该方法首先根据词频统计的向量空间计算出克隆群之间的相似度，将满足阈值的克隆群确定为具有映射关系的候选克隆群，再匹配克隆群特征。这些特征有开始行、结束行、代码行数以及克隆群所处的文件名。最后通过匹配克隆群特征确定克隆群的映射关系。具体如算法 1 所示。

算法 1 克隆映射算法

Input：Detect Result

Output：Mapping XML file

For $CC_{n+1,i}$ in V_{n+1} do

Statistic word frequency in $CC_{n+1,i}$ and store to $D_{n+1,i}$

Extract Feature item in $CC_{n+1,i}$ and store to $T_{n+1,i}$

  Standardize $D_{n+1,i}$

For $CC_{n,i}$ in V_n do

Statistic word frequency in $CC_{n,i}$ and store to $D_{n,i}$

Extract Feature item in $CC_{n,i}$ and store to $T_{n,i}$

Standardize $T_{n,i}$

For similarity between $D_{n+1,i}$ and $D_{n,j}$ do

Store to array cosin[]

For match between $T_{n+1,i}$ and $T_{n,j}$ do

Store to array T[]

If cosin[k]>=t && T[k]==True Then

$CC_{n+1,i} \rightarrow CC_{n,j}$

Else

$CC_{n+1,i} \rightarrow$ NULL

Return mapping results

3.1.3　克隆家系

克隆家系反映了克隆代码进行创建、繁殖和进化的过程，包括克隆片段的映射、克隆群的映射、克隆的演化模式等方面的信息，为推荐重构克隆代码提供所需的历史信息。构建克隆家系是在克隆代码检测和映射的基础上进行的。添加克隆演化模式是构建克隆家系的一个重要过程。其中演化模式主要包括静态、增加、减少、分离、合并、一致和不一致 7 种短期演化模式，具体的变化方式如图 4 所示。

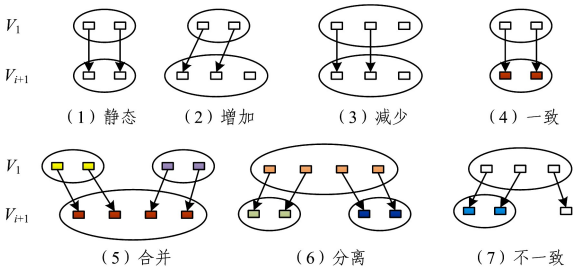

图 4　克隆群演化模式

Fig. 4　Evolution model of clonal groups

识别克隆演化模式的过程如下：1）对单个克隆片段进行分类；2）划分且识别克隆群演化模式，此过程是在克隆片段的基础上进行的。因为克隆片段是包含在克隆群中的，所以分阶段识别克隆演化模式能够较大程度地减少识别次数，提高识别效率。

为克隆群添加演化模式主要是根据克隆片段的数量以及内容的变化情况来决定的。假设版本 V_i 中的克隆群 CG_i 和版本 V_{i+1} 中的克隆群 CG_{i+1} 存在克隆映射关系，CF 为克隆群中的克隆片段，具体的识别步骤如下。

步骤 1　若 CG_i 和 CG_{i+1} 中片段数量相同且内容未发生任何变化，则添加“Static”模式；若 CF 的内容发生了相同的变化，则添加“Consistent”模式。

步骤 2　若 CG_{i+1} 中的 CF 为新增片段，则添加“Add”模式；若克隆片段的内容也发生了相同的变化，则添加“Consistent”模式。

步骤 3　若 CF 在下一版本 V_{i+1} 中消失，则添加“Subtract”模式；若 CF 的内容也发生了变化，则添加“Inconsistent”模式。

步骤 4　若 V_{i+1} 中两个克隆群都起源于 V_i 中的克隆群 CG_i，则添加“Split”模式；若 V_i 中两个克隆群都对应 V_{i+1} 中的克隆群 CG_{i+1}，则添加“Merge”模式。

3.1.4　克隆代码维护日志

在提取克隆代码演化历史信息的过程中，克隆检测、克隆映射以及克隆家系仅仅为克隆代码提供了“候选特征”，如何提取出与重构相关的特征，除了依据经验，还应该参考维护日志提供的修改或重构数据。本文选取具有维护提交日志的多版本软件作为分析对象，提取软件系统中每一个发布版本的维护日志，并对每一种克隆维护条目进行类型划分，并结合克隆演化过程的克隆变化属性特征进行分析，以确定可重构的克隆代码具备的特征。

不同的系统软件的维护日志不尽相同，本文选取多个系统软件进行分析，例如图 5 展示了 Junit 系统的部分维护提交日志。

图 5　维护提交日志(电子版为彩色)

Fig. 5　Maintenance submission log

图 5(a)展示了小型开源软件 Junit 的维护日志，包含了多条维护条目(图中显示了 3 条)，每个维护条目包括维护编号(commit)、维护人员(Author)、维护日期(Date)以及维护操作。图 5(b)展示了图 5(a)中 1 条维护条目的详细内容，此维护条目的操作为修复 JavaDoc 系统中的错字和格式。该维护条目修改了两处代码，红色部分为修改前的代码，绿色部分为修改后的代码。通过维护日志发现，软件代码会经常发生变化。通过检查多款软件的维护日志可以发现，经常修改的克隆代码多数是有问题的，而克隆代码的改变频率以及演化模式体现了克隆的修改特征，通常需要更高的维护代价。本文通过对其他系统维护日志的分析，将这些属性与克隆代码进行关联，同时将软件维护提交日志中的修改特征和已提取到的克隆演化历史信息相结合，以更加准确地确定克隆代码的重构集和跟踪集。

3.2 识别重构候选集和识别跟踪候选集

3.2.1 识别重构候选集

本文将经历过一致变化的克隆代码归入到重构候选集中。关注在演化历史过程中经过维护的克隆群内的克隆片段发生的一致性变化的情况，如图 6 所示。

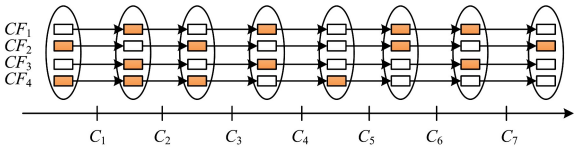

图 6 发生一致性变化的克隆代码片段

Fig. 6 Clone code snippet with consistent changes

图 6 中有阴影的片段表示发生变化的克隆片段，白色的片段表示未发生变化的克隆片段，$C_i(i=1,2,\cdots,n)$表示软件维护提交操作。可以看出，每个版本克隆群中的 4 条克隆片段经历了 7 次提交操作，克隆片段 CF_1 和 CF_3 绝大多数情况下都是同时发生变化(即在大多数情况下，两者的维护日志是相同的)，表明这两个克隆片段的质量较差，需要被维护，而经过两次及两次以上维护之后问题仍然存在，这种多次维护操作后质量仍然没有得到改善的克隆片段需要进行重构。因此本研究将经过多次维护操作后质量仍然没有得到改善的克隆片段添加到重构候选集。而 CF_4 和其他克隆片段没有发生共同变化的情况，这可能是因为克隆片段 CF_4 经历了独立演化，因此不划入代码重构候选集中。

3.2.2 识别跟踪候选集

本文将分离、合并的演化模式归入到跟踪候选集中。关注在克隆代码演化历史过程中克隆群经过维护操作后，其克隆片段发生的分离现象，并在之后的维护操作中重新合并一个克隆群。具体情况如图 7 所示。

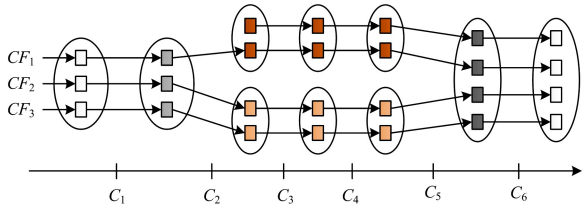

图 7 需要跟踪的克隆代码的变化

Fig. 7 Changes of clone code to be tracked

在提取软件系统的克隆家系时发现，有一部分克隆群中的克隆片段发生了分离和合并。克隆发生分离的原因可能是其中的一个或多个克隆片段经过维护之后，与克隆群内的其他片段不再具有映射关系，而与其他克隆群内的片段形成了映射关系。随着版本的推移，这些分离的片段经过维护之后可能会重新组成一个克隆群，进而出现一致或不一致的演化。因此须进一步跟踪这些克隆片段，即加入克隆代码跟踪候选集，持续跟踪并关注此类情况的后续变化。

构建克隆候选集的算法如算法 2 所示。首先根据每个修复日志中的提交 ID 和维护详细描述信息进行关联，提取出维护操作的代码文件；然后通过提交操作中的文件路径与检测出的克隆代码所在的路径关联，获取到与克隆代码相关的维护操作；最后通过代码的维护操作以及克隆代码演化模式匹配确定重构与跟踪候选集。

算法 2 克隆候选集的构建算法

```
Input:List<ChangeLog>,List<Detect Result>,lsChangeCloneLogs
Output:Set<Refactoring Clone>,Set<Tracking Clone>
For changeinfo in List<ChangeLogInfo> do
multiset ← Changeid at corresponding fixid from lsChangeLogs;
contain ChangeFiles.add(multiset);
End for
listClonePaths ←CloneDetectResults(xmlFiles)
For each ClonePathinlistClonePaths do
    If ClonePath exist in containChangeFiles then
lscontainChangeClone.add(containChangeClone)
End for
For each contain ChangeClone in lscontainChangeClone do
    If multiset in ConsistentPattern,InconsistentPattern
        Set<Refactoring Clone>.add(multiset)
    If multiset in SplitPattern
        Set<Tracking Clone>.add(multiset)
End for
return Set<Refactoring Clone> Set<Tracking Clone>
```

上述构建克隆重构和跟踪候选集的整体算法的具体步骤如下。

步骤 1 从日志管理系统中提取维护操作提交的日志文件。

步骤 2 获取含维护操作的文件后，将日志文件与克隆代码进行关联，本文使用 Nicad 检测克隆代码，并将克隆检测结果存储到 XML 文件中。

步骤 3 检测结果显示了克隆代码信息，具体包括克隆的起始行、结束行、代码行数以及克隆代码所在路径。提取上述信息，并将已提取的克隆代码文件路径与含有维护提交操作的文件路径进行匹配，以获得含有维护操作的克隆代码文件。

步骤 4 将获取到的关联文件与克隆家系中的演化模式进行匹配，如果维护操作导致克隆群发生了一致或不一致变化，则分析其内部的克隆片段的变化情况。若内部的两个及多个克隆片段共同发生变化，则将其添加到克隆重构候选集；如果维护操作引起克隆群发生了分离变化，则将此类情况添

加到克隆跟踪候选集。

3.3　特征提取

特征提取是机器学习和模式识别领域常用的数据预处理方式，是训练机器学习模型的必要步骤。Steidl 等[29]认为在克隆代码质量检测中需要修改的克隆代码更容易发生在代码行数较长、语句数目较多、类个数以及函数个数较多的克隆代码中。本实验小组王欢等[30]提出在克隆代码特征选择的问题中，克隆代码中圈复杂度、分支语句比例过高的代码发生错误的可能性更大。基于以上研究，本文提取与此相关的一些特征来预测重构克隆代码，具体包括代码行数、语句数、类个数、平均每个类的方法数、函数个数、平均每个函数包含的语句数目、圈复杂度、分支语句比例以及注释比例等特征。可维护性指数反映的是代码维护的难易程度，该值越大表示可维护性越好。可维护性对开发人员来说特别重要，因此本文提取此特征来预测重构克隆代码。本文提取 13 类静态特征，详细信息如表 1 所列。

表 1　特征的详细描述

Table 1　Detailed description of features

特征类型	特征	描述
静态特征	代码行数	代码行数表示空行在内的代码行数
	语句数	语句是以分号结尾的
	分支语句比例	分支语句比例表示分支语句占语句数目的比例
	注释比例	注释比例表示注释行占总行数的比例
	类个数	类个数表示包括 class，struct 和 template 在内的类的个数
	平均每个类的方法数	平均每个类的方法数，即类方法数除以类个数
	函数个数	所有函数的个数
	平均每个函数包含的语句数目	总的函数语句数目除以函数数目得到该值
	函数圈复杂度	函数圈复杂度表示一个函数可执行路径的数目
	函数深度	函数深度表示函数中分支嵌套的层数
	程序容量	程序在词汇上的复杂性
	参数个数	参数个数表示函数体的复杂性程度
	可维护性指数	软件的可维护性是指理解、改正、改动、改进软件的难易程度
演化特征	克隆寿命	某个克隆存在于软件系统中所经历的版本数
	克隆演化频率	克隆代码在历史版本中经历的总改变次数与克隆寿命的比值
	克隆演化模式	克隆演化反映克隆代码存在、发展、变化的规律

克隆代码在演化中是不断发展变化的，仅根据单一版本的的特征反映重构克隆代码是不够的。本文从演化角度即使用演化特征来反映克隆代码发生重构的可能性，演化特征具体包括：克隆寿命、克隆代码演化频率、克隆代码演化模式。这 3 类特征反映了克隆代码在演化过程中发生的改变对软件质量的影响，将有助于分析克隆代码需要重构的可能性。

我们使用代码工具 SourceMonitor 对以上静态特征进行提取，SourceMonitor 是一款可对多种语言（C＋＋，C，C＃，VB. net，Java，Visual，Basic 和 HTML）编写的代码进行度量的工具，并且针对不同的语言，输出不同的代码度量值。输出结果如图 8 所示。

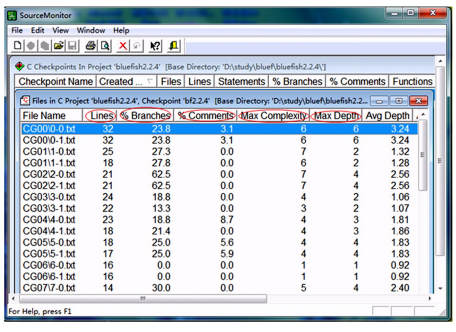

图 8　特征提取结果

Fig. 8　Feature extraction results

代码行数从 Lines 标签中获取，语句数从“Statements”标签中获取，分支语句比例从％Percent Branch statements 标签中获取，注释比例从％ comments 标签中获取，类个数从 classes 标签中获取，平均每个类方法数从“Methods per class”标签中获取，函数个数从“Functions”标签中获取，平均每个函数包含的语句数目从“Arerage statements per Method”标签中获取，函数圈复杂度从 Function complexity 中获取，函数深度从 Avg Depth 中获取。

静态特征中的可维护性指数计算公式为：$M=\mathrm{MAX}(0,(171-5.2*\ln(A)-0.23*(B)-16.2*\ln(C))*100/171)$，其中 A 代表程序容量，B 代表圈复杂度，C 代表代码行数。

本文提取的演化特征有克隆寿命、改变频率、演化模式 3 类，提取方法主要用本团队自主开发的克隆函数提取器 FCGE 提取，提取结果如图 9 所示。

```
1  postgresql-9.3.2.CG[1]<---(无变化)---postgresql-9.3.1.CG[1]<---(无变化)---postgresql-9.3.0.CG[1]
2  <---(无变化)---postgresql-9.2.6.CG[1]<---(无变化)---postgresql-9.2.5.CG[1]<---(无变化)---
3  postgresql-9.2.4.CG[1]<---(无变化)---postgresql-9.2.3.CG[1]<---(不一致变化)---postgresql-9.2.2.CG[1]
4  <---(增加)---postgresql-9.2.1.CG[1]<---(无变化)---postgresql-9.2.0.CG[1]<---(无变化)---
5  postgresql-9.1.11.CG[1]<---(无变化)---postgresql-9.1.9.CG[1]<---(无变化)---postgresql-9.1.8.CG[1]
6  <---(去除)---postgresql-9.1.7.CG[1]<---(无变化)---postgresql-9.1.6.CG[1]<---(无变化)---
7  postgresql-9.1.5.CG[1]<---(不一致变化)---postgresql-9.1.4.CG[1]<---(无变化)---postgresql-9.1.3.CG[1]
8  <---(不一致变化)---postgresql-9.1.2.CG[1]
9
0  postgresql-9.3.2.CG[2]<---(无变化)---postgresql-9.3.1.CG[2]<---(无变化)---postgresql-9.3.0.CG[2]
1  <---(无变化)---postgresql-9.2.6.CG[2]<---(无变化)---postgresql-9.2.5.CG[2]<---(无变化)---
2  postgresql-9.2.4.CG[2]<---(增加)---postgresql-9.2.3.CG[2]<---(不一致变化)---postgresql-9.2.2.CG[2]
3  <---(去除)---postgresql-9.2.1.CG[2]<---(无变化)---postgresql-9.2.0.CG[2]<---(无变化)---
4  postgresql-9.1.11.CG[2]<---(无变化)---postgresql-9.1.9.CG[2]<---(无变化)---postgresql-9.1.8.CG[2]
5  <---(一致变化)---postgresql-9.1.7.CG[2]<---(无变化)---postgresql-9.1.6.CG[2]<---(无变化)---
6  postgresql-9.1.5.CG[2]<---(无变化)---postgresql-9.1.4.CG[2]<---(不一致变化)---postgresql-9.1.3.CG[2]
7  <---(无变化)---postgresql-9.1.2.CG[2]
```

图 9　演化特征提取结果

Fig. 9　Extraction results of evolution features

3.4　推荐用于重构的克隆代码

在识别并构建出克隆代码重构与跟踪候选集之后，在克隆代码重构候选集中提取静态特征和演化特征，使用这些特征构建特征样本数据集并使用机器学习的方法推荐重构。

本文选取了 3 种使用比较广泛且预测性能较好的机器学习模型：朴素贝叶斯（Naive Bayes）、C4. 5（一种决策树算法）

以及贝叶斯网络(Bayesian Network)。朴素贝叶斯:此模型有稳定的分类效率,能处理多分类任务,尤其是数据量超出内存时可以一批批地增量训练,对缺失数据不太敏感且算法较简单。C4.5:此模型不仅易于理解和实现,而且数据准备较简单,可以同时处理数据型和常规型数据,可在较短时间内对大型数据源产生良好的效果。贝叶斯网络:对于不确定性问题,此模型具有强大的处理能力,能在有限、不完整的、不确定的信息条件下进行推理和学习。因此,本文使用上述 3 种机器学习模型进行预测,最终选择效果最好的模型来推荐重构克隆代码。

3.5 评估分类结果

在对未知数据样本进行预测时,有的样本被正确分类,有的样本被错误分类。因此,为了评价克隆重构预测模型的性能,本文使用分类器评价中常用的查全率、查准率、F 值等进行评价,其对应的混合矩阵如表 2 所列。其中,TP 表示正确的正例(True Positive,TP),FN 表示错误的反例(False Negative,FN),FP 表示错误的的正例(False Positive FP),TN 表示正确的反例(True Negative,TN)。

表 2 混合矩阵

Table 2 Mixed matrix

预测结果	正类	反类
正类	正确的正例(TP)	错误的反例(FN)
反类	错误的正例(FP)	正确的反例(TN)

查准率、查全率、F 值的定义如下:

$$Precision=\frac{TP}{TP+FP} \tag{1}$$

$$Recall=\frac{TP}{TP+FN} \tag{2}$$

$$F=\frac{2*Precision*Recall}{Precision+Recall} \tag{3}$$

查全率表示分类器正确预测克隆重构的比例,是分类问题中常用的指标,反映了分类器对数据集的整体分类性能。

4 实验与分析

4.1 实验数据选取

本实验选取的 7 款开源软件分别是:ffmpeg,lighttpd,smalltalk,fdisk,emacs,dovecot 和 clawsmail。这些软件的基本信息如表 3 所列。

表 3 实验软件的基本信息

Table 3 Basic information of experimental softwares

软件名称	开发语言	功能	代码行数
ffmpeg	C	多媒体工具	543560
dovecot	C	邮件服务器	214386
fdisk	C	磁盘管理	187958
smalltalk	C	程序集成开发境	3778858
lighttpd	C	Web 服务器	53378
emacs	C	文本编辑器	341986
clawsmail	C	邮件客户端	220235

4.2 实验结果

本文使用 weka 进行重构克隆代码的预测,在实验结果中选择“Correctly Classified Instances”的参数进行分析,“Correctly Classified Instances”表示当前参与分类的实例中被正确分类的实例数目,换句话说就是预测模型的准确度。本文选取朴素贝叶斯、决策树以及贝叶斯网络 3 种分类器对重构克隆代码进行预测,各个分类器的预测结果如表 4 所列。

表 4 预测模型的准确度

Table 4 Accuracy of predictive model

(单位:%)

软件系统	Naive Bayes	C4.5	Bayesian Network
clawsmail	89.8596	96.7293	97.1919
dovecot	54.0373	95.6522	96.2733
emacs	93.0591	97.6864	97.9493
fdisk	92.5926	88.8889	79.6269
ffmpeg	82.4297	97.1888	97.1888
lighttpd	87.6543	97.5309	97.5309
smalltalk	52.2613	97.9899	97.9899

根据预测模型的准确度可以得出如下结论:

(1)在 7 个软件系统中,使用朴素贝叶斯预测 emacs 系统的重构克隆代码的准确度为 93.0591%,但预测 smalltalk 系统的重构克隆代码的准确度为 52.2613%。

(2)在 7 个软件系统中,使用决策树预测 smalltalk 系统的重构克隆代码的准确度为 97.9899%,但预测 fdisk 系统的重构克隆代码的准确度为 88.8889%。

(3)在 7 个软件系统中,使用贝叶斯网络预测 smalltalk 系统的重构克隆代码的准确度为 97.9899%,但是预测 fdisk 系统的重构克隆代码的准确度为 79.6269%。

(4)使用朴素贝叶斯、决策树、贝叶斯网络预测重构克隆代码的准确度分别为 78%,96%,94%。

基于以上结论,得出决策树预测模型和贝叶斯网络预测模型的准确度最高,但是决策树预测模型更稳定。因此使用决策树分类器来推荐重构克隆代码的效果最佳。这就说明,决策树预测模型在推荐重构克隆方面要优于其他分类器。

本实验选取的评估验证方法为 K 折交叉验证方法,其基本思想是把输入数据集划分为训练集和测试集。因为测试集对于分类器是不可见的,所以通过训练集输出的结果进行交叉验证。

步骤 1 将数据集 D 划分为 k 个大小相似的互斥子集,即 $D=D_1\cap D_2\cap D_3\cap\cdots\cap D_k$,每个子集之间没有交集。

步骤 2 每次用 $k-1$ 个子集的并集作为训练集,余下的作为测试集,这样可以得到 k 组训练/测试集。

步骤 3 进行 k 次训练和测试,最终返回这 k 个结果的均值。

步骤 4 随机进行多次划分。

本实验选取 $k=10$,即 10 折交叉验证来评估分类器预测模型,之所以选择将数据集分为 10 份,是因为通过利用大量数据集,使用不同技术进行大量实验的结果表明 10 折交叉验证是获得误差最小的方法,而且也有理论依据可以证明。朴素贝叶斯、决策树、贝叶斯网络的验证结果如表 5-表 7 所列。3 个表分别显示了朴素贝叶斯、决策树、贝叶斯网络预测模型的精度、召回度、F 值。对于朴素贝叶斯预测模型,10 次验证的精度从 70.0%提高到 97.9%,召回率从 47.7%提高到 94.5%,F 值范围从 62.9%提高到 96.4%。对于决策树预测

模型，10 次验证的精度从 60%提高到 98%，召回率从 75%提高到 99.8%，F 值范围从 66.7%提高到 99%。对于贝叶斯网络预测模型，10 次验证的精度从 94.9%提高到 98%，召回范围从 80.4%提高到 99.8%，F 值范围从 87.1%提高到 99%。表 5－表 7 中的数据表明，使用 3 种预测模型，10 次验证的精度、召回率、F 值均达到 90%以上，同时也验证了决策树预测模型的召回率和 F 值更高。因此本文所构造的分类器的效果比随机选择执行的效果更好。

表 5　Naive Bayes 的测试结果

Table 5　Test results of Naive Bayes

软件系统	查准率	查全率	F 值
clawsmail	0.970	0.925	0.947
dovecot	0.976	0.535	0.629
emacs	0.984	0.945	0.964
fdisk	0.700	0.875	0.778
ffmpeg	0.982	0.835	0.902
lighttpd	0.973	0.899	0.934
smalltalk	0.979	0.477	0.641

表 6　Bayesian Network 的测试结果

Table 6　Test results of Bayesian Network

软件系统	查准率	查全率	F 值
clawsmail	0.972	0.998	0.986
dovecot	0.963	0.991	0.981
emacs	0.979	0.994	0.990
fdisk	0.949	0.804	0.871
ffmpeg	0.972	0.998	0.986
lighttpd	0.975	0.996	0.988
smalltalk	0.980	0.997	0.990

表 7　C4.5 的测试结果

Table 7　Test results of C4.5

软件系统	查准率	查全率	F 值
clawsmail	0.976	0.990	0.983
dovecot	0.963	0.994	0.978
emacs	0.979	0.997	0.988
fdisk	0.600	0.750	0.667
ffmpeg	0.972	0.998	0.986
lighttpd	0.975	0.994	0.988
smalltalk	0.980	0.997	0.990

4.3　同类实验对比分析

当前国内外使用机器学习的方法来预测需要重构的克隆代码的方法较少，而使用机器学习方法来预测需要重构的克隆代码的方法中具有代表性的是 Wang 等[31]提出的方法。在检测克隆代码时，Wang 等[31]使用了 Iclones 检测，而本文使用 NiCad 来检测克隆代码，同时本文与 Wang 等[31]提取的特征也不完全一致，其中本文研究的项目为 C 项目而 Wang[31]所使用的项目为 Java 项目，因此两种方法不具有对比性。本实验小组刘冬瑞等[27]曾使用贝叶斯网络预测需要重构的克隆代码，其使用了 Fclones 进行克隆检测，而本文使用 Nicad 进行克隆检测；刘冬瑞等[27]根据 ISO 软件质量标准来选取特征，而本文从静态和演化两大类中提取特征，因此本文仅对推荐克隆重构做对比实验。其中本文研究和刘冬瑞等[30]的研究所使用的项目均为 C 项目，且均使用贝叶斯网络模型进行预测。实验平台同为操作系统 Ubuntu14.04 64 位，8GB 内存，2 核 CPU。版本信息如表 8 所列。

表 8　推荐克隆重构对比实验软件信息

Table 8　Recommended clone reconstruction comparison experimental software information

软件	开始版本	结束版本	commits	重构实例
lighttpt	1.4.15	1.4.39	53378	308
smalltalk	2.0.11	3.2.5	3078858	381
clawsmail	1.9.100	3.9.3	220235	289

对比实验的结果如表 9 所列。从推荐克隆重构结果中分析软件 lighttpt 的查准率、查全率、F 值。与刘冬瑞等[27]的方法相比，本文研究的查准率提高了 9%，查全率提高了 11.4%，F 值提高了 8.4%。从推荐克隆重构结果中分析软件 smalltalk 的查准率、查全率和 F 值。与刘冬瑞等[27]的方法相比，本文研究的查准率提高了 9.1%，查全率提高了 7%，F 值提高了 5.2%。从推荐克隆重构结果中分析软件 clawsmail 的查准率、查全率和 F 值。与刘冬瑞等[27]的方法相比，本文研究的查准率提高了 11.7%，查全率提高了 7%，F 值提高了 5.9%。综上，本文推荐克隆代码重构方法的查全率、查准率、F 值均高于刘冬瑞等[27]提出的方法。

表 9　推荐克隆重构的同类实验结果对比

Table 9　Similar experiments comparison of recommendation reconstruction clone

软件	方法	查准率	查全率	F 值
lighttpt	刘冬瑞等[27]方法	0.926	0.882	0.904
	本文方法	0.975	0.996	0.988
smalltalk	刘冬瑞等[27]方法	0.889	0.990	0.934
	本文方法	0.980	0.997	0.986
clawsmail	刘冬瑞等[27]方法	0.865	0.991	0.927
	本文方法	0.972	0.998	0.986

结束语　本文从克隆检测、克隆映射、克隆家系以及软件维护日志管理系统中提取与克隆代码密切相关的演化历史信息，并识别出需要重构和跟踪的克隆代码候选集，进而使用不同的机器学习方法推荐需要重构的克隆代码。本文使用朴素贝叶斯、决策树、贝叶斯网络 3 种预测模型在 7 款软件中进行预测，得出使用决策树预测模型的效果最佳，准确度达到了 95%。本研究可以为克隆重构提供更好的资源分配，从而改进克隆管理。

本文研究仍然存在一些不足之处，例如目前本文方法只能推荐用 C 语言编写的开源软件，同时推荐方法单一。本文在后续研究中将继续改进，尝试使用深度学习、半监督学习来推荐克隆重构。

参 考 文 献

[1] BALAZINSKA M, MERLO E, DAGENAIS M, et al. Advanced Clone-analysis to Support Object-oriented System Refactoring [C]// Proceedings of the Seventh Working Conference on Reverse Engineering. IEEE press, 2000: 98-107.

[2] KIM M, SAZAWAL V, NOTKIN D, et al. An empirical study of code clone genealogies [J]. AcmSigsoft Software Engineering Notes, 2005, 30(5): 187-196.

[3] ROY C K, CORDY J R, KOSCHKE R. Comparison and evaluation of code clone detection techniques and tools: A qualitative approach[J]. Science of Computer Programming, 2009, 74(7): 470-495.

[4] ROY C K, CORDY J R. Near-miss function clones in open

source software:an empirical study[J]. Journal of Software Maintenance & Evolution Research & Practice,2010,22(3):165-189.

[5] BASIT H A,PUGLISII S J,SMYTH W F,et al. Efficient token based clone detection with flexible tokenization[C]// TheJoint Meeting on European Software Engineering Conference and the ACM Sigsoft Symposium on the Foundations of Software Engineering:Companion Papers. ACM,2007:513-516.

[6] DUALA-EKOKO E,ROBILLARD M P. Clonetracker:tool support for code clone management[C]// Proceedings of the 2008 International Conference on Software Engineering. New York:ACM,2008:843-846.

[7] APDAN M,AKTAS M,YIGITI M. On the Structural Code Clone Detection Problem:A Survey and Software Metric Based Approach[C]// Computational Science and Its Applications (ICCSA 2014). Springer International Publishing,2014:492-507.

[8] CUOMO A,SANTONE A,VILLANO U. A novel approach based on formal methods for clone detection[C]// Proceedings of the 2012 International Workshop on Software Clones. Piscataway,NJ:IEEE,2012:8-14.

[9] CALEFATO F,LANUBILE F,MALLARDO T. Function clone detection in web applications:a semiautomatedapproach[J]. Journal of Web Engineering,2004,3(1):3-21.

[10] ZHANG J J,WANG C H,ZHANG L P,et al. Clone code detection based on Token edit distance [J]. Journal of Computer Applications,2015(12):3536-3543. (in Chinese)
张久杰,王春晖,张丽萍,等. 基于 Token 编辑距离检测克隆代码[J]. 计算机应用,2015(12):3536-3543.

[11] BARBOUR L,KHOMH F,ZOU Y. Late propagation in software clones[C]// Proceedings of the 27th IEEE International Conference on Software Maintenance. Washington DC:IEEE Computer Society,2011:273-282.

[12] SAHA R K,ROY C K,SCHNEIDER K A. An automatic framework for extracting and classifying near-miss clone genealogies [C]// IEEE International Conference on Software Maintenance. IEEE,2011:293-302.

[13] HOTTA K,HIGO Y,KUSUMOTO S. Clone Tracking based on Similarity of CRD[J]. Technical Report of IeiceSs,2013:113-117.

[14] ZHANG R X,ZHANG L P,WANG C H,et al. Clonal group mapping method based on topic modeling technology[J]. Computer Engineering and Design,2015(6):1524-1529. (in Chinese)
张瑞霞,张丽萍,王春晖,等. 基于主题建模技术的克隆群映射方法[J]. 计算机工程与设计,2015(6):1524-1529.

[15] GÖDE N,KOSCHKE R. Incremental Clone Detection[C]// European Conference on Software Maintenance & Reengineering. 2009:219-228.

[16] GE G S,LIU D S,HOU M. Software multi-version clonal group mapping method based on LDA and DBSCAN[J]. Journal of Computer Applications,2017,34(2):481-486. (in Chinese)
葛广帅,刘东升,侯敏. 基于 LDA 和 DBSCAN 的软件多版本克隆群映射方法[J]. 计算机应用研究,2017,34(2):481-486.

[17] BARBOUR L,KHOMH F,ZOU Y. An empirical study of faults in late propagation clone genealogies[J]. Journal of Software:Evolution and Process,2013,25(11):1139-1165.

[18] KIM M,SAZAWAL V,NOTKKIN D,et al. An Empirical Study of Code Clone Genealogies[C]// Proceedings of the 2005 10th European Software Engineering Conference Held Jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering. New York:ACM,2005:187-196.

[19] SAHA R K,ROY C K,SCHNEIDER K A. An automatic framework for extracting and classifying near-miss clone genealogies [C]// Proceedings of the 2011 IEEE International Conference on Software Maintenance. Piscataway,NJ:IEEE,2011:293-302.

[20] MENG C,SU X H,WANG T T,et al. A New Clone Group Mapping Algorithm for Extracting Clone Genealogy on Multiversion Software[C]// International Conference on Instrumentation. 2013:848-853.

[21] GE G S,LIU D S,ZHANG L P,et al. Evolutionary Trace Construction and Pattern Recognition of Clone Code Based on Graph Model[J]. Computer Engineering,2017,43(5):47-54. (in Chinese)
葛广帅,刘东升,张丽萍,等. 基于图模型的克隆代码演化痕迹构建及模式识别[J]. 计算机工程,2017,43(5):47-54.

[22] OPDYKE W F. Refactoring Object Frame Works [M]. Illinois:University of Illinois at Urban-Champaign,1992:18-35.

[23] BIAN Y X. Research on Process Extraction Method of Reconfigurable Clone Code [D]. Harbin:Harbin Institute of Technology,2014. (in Chinese)
边奕心. 可重构克隆代码的过程提取方法研究[D]. 哈尔滨:哈尔滨工业大学,2014.

[24] BAKOTA T. Tracking the Evolution of Code Clones[C]// The 37th International Conference on Current Trends in Theory and Practice of Computer Science. Novy′ Smokovec,Slovakia:Springer,2011:86-98.

[25] MONDAL M,ROY C K,SCHNEIDER K A. SPCP-Miner:A tool for mining code clones that are important for refactoring or tracking[C]// IEEE,International Conference on Software Analysis,Evolution and Reengineering. IEEE,2015:484-488.

[26] HIGO Y,KUSUMOTO S,INOUE K. A metric-based approach to identifying refactoring opportunities for merging code clones in a Java software system[J]. Journal of Software Maintenance and Evolution:Research and Practice,2008,20(6):435-461.

[27] LIU D R,LIU D S,ZHANG L P,et al. Prediction of cloned code quality based on Bayesian network[J]. Computer Science,2017,44(4):165-168. (in Chinese)
刘冬瑞,刘东升,张丽萍,等. 基于贝叶斯网络预测克隆代码质量[J]. 计算机科学,2017,44(4):165-168.

[28] SHE R R,ZHANG L P,HOU M,et al. Method for recommending clone reconstruction based on decision tree[J]. Journal of Computer Applications,2018,38(7):213-219,245. (in Chinese)
折蓉蓉,张丽萍,侯敏,等. 基于决策树推荐克隆重构的方法[J]. 计算机应用,2018,38(7):213-219,245.

[29] STEIDL D. Feature-based detection of bugs in clones[C]// International Workshop on Software Clones. IEEE,2013:76-82.

[30] WANG H,ZHANG L P,YAN S,et al. Feature selection model in cloned code harmful prediction[J]. Journal of Computer Applications,2017,37(4):1135-1142. (in Chinese)
王欢,张丽萍,闫盛,等. 克隆代码有害性预测中的特征选择模型[J]. 计算机应用,2017,37(4):1135-1142.

[31] WANG W,GODFREY M W. Recommending Clones for Refactoring Using Design,Context,and History[C]// IEEE International Conference on Software Maintenance and Evolution. IEEE Computer Society,2014:331-340.