硕士学位论文

克隆代码检索与克隆进化分析可视化系统的设计与实现

DESIGN AND IMPLEMENTATION OF CODE RETRIEVAL AND VISUALIZATION OF CLONE EVOLUTION ANALYSIS SYSTEM

赵雯

哈尔滨工业大学

2014 年 6 月

国内图书分类号：TP311.5　　学校代码：10213
国际图书分类号：681　　密级：公开

工程硕士学位论文

克隆代码检索与克隆进化分析可视化系统的设计与实现

硕士研究生：赵雯
导　　师：苏小红教授
申请学位：工程硕士
学　　科：计算机技术
所在单位：计算机科学与技术学院
答辩日期：2014 年 6 月
授予学位单位：哈尔滨工业大学

Classified Index: TP311.5
U.D.C.: 681

Dissertation for the Master Degree in Engineering

DESIGN AND IMPLEMENTATION OF CODE RETRIEVAL AND VISUALIZATION OF CLONE EVOLUTION ANALYSIS SYSTEM

Candidate: Zhao Wen
Supervisor: Prof. Su Xiaohong
Academic Degree Applied for: Master of Engineering
Speciality: Computer Technology
Affiliation: School of Computer Science and Technology
Date of Defence: June, 2014
Degree-Conferring-Institution: Harbin Institute of Technology

摘　要

在如今信息化快速发展的时代，软件系统的安全性与可靠性越来越被人们所重视。克隆代码是影响软件系统质量的一个重要因素，有效的加强克隆代码的分析与管理工作可以帮助提高软件质量。近年来克隆代码的相关研究工作从面向单一版本的软件系统逐渐扩展至多版本的软件系统。这种克隆进化信息则反映了克隆代码在不同软件版本中的变化情况，它对于软件缺陷的分析与改善有着不可忽视的实际意义。

目前克隆进化方面的研究多数集中在克隆家系和进化模式的提取上，对于这些克隆检测工具和克隆家系提取器得到的大量文本形式的信息，并不容易理解和管理。本文针对上述问题，基于克隆代码进化信息实现了克隆代码的检索及克隆家系的可视化工作。可视化工作中包含了克隆家系整体可视化视图和单个克隆家系含克隆群映射关系的详细视图；检索工作中包含了对指定克隆代码片段的检索和指定度量值的检索。本文系统适用于不同语言编写的开源软件项目，并支持友好的交互功能。

另外，在克隆进化分析方面，本文提出并实现了一种基于无监督学习模型的分析方法，结合克隆代码片段的静态度量与进化度量，从进化、容量、相似度三个方面，对克隆代码在发展过程中所表现出来的特征进行总结分析，为进一步的克隆代码评价和管理工作提供有效指导。

关键字：克隆进化分析；克隆代码检索；克隆家系可视化；无监督学习

Abstract

With the rapid development of information technology, people pay more and more attention to the security and reliability of software systems. Clone code is an important factor of affecting the quality of software systems. Effective analysis and management of the clone code can help improve software quality. In recent years, research of clone code focus on versions of the software systems instead of a single version. The evolution information reflects the clone code changes in different versions of software systems, and it has practical significance for the analysis and improvement of software defects.

Most research of the clone evolution focus on the extraction of the clone genealogy and evolution patterns. The large amounts of text information achieved from clone detection tools and clone genealogy extractors is not easy to understand and manage. Aiming at these problems, we implement clone code retrieval work and clone genealogy visualization work based on the evolution information of clone code. Visualization work includes the overall visual view and a single clone genealogy detail view containing clone groups mapping relations. And retrieval work includes the retrieval of the specified clone code fragment and the retrieval of clone code fragment with the specified metrics. The system applies to open source software projects of different languages, and it supports friendly interactive features.

In addition, we propose and implement an analytical method based on unsupervised learning models in the filed of clone evolution analysis. We use static metrics and evolution metrics to describe clone code, then summarize and analyze the clone code characteristics shown in the evolution to provide effective guidance for further evaluation and management of code clone.

Keywords:clone evolution analysis, clone code retrieval, clone genealogy visualization, unsupervised learning

目　录

第1章 绪　论

1.1 课题研究的目的与意义

1.1.1 课题来源

本论文的课题来源是国家自然科学基金项目“无定型克隆代码的检测与重构方法”（批准号：61173021）。

1.1.2 研究的目的和意义

随着社会体制的不断增强与计算机科学相关技术的迅速发展，软件的安全性与可靠性问题越来越受到社会各界的关注。很多的软件设计问题都来源于那些重复、不清晰、结构复杂的代码。这些代码中存在的潜在缺陷不仅降低了代码的可读性，而且对其准确性和可靠性都造成了一定的影响。克隆代码就是影响软件质量的一个重要因素，尤其是针对大规模的软件应用系统，研究表明克隆代码所占比例约为 7-23%[1,2]，甚至有时高达 50%[3]。克隆代码产生的主要原因之一是程序员在开发过程中对代码的拷贝粘贴，另外还包括设计模式、相似的框架和 API 的使用等其他因素[4]。

克隆代码对于软件系统是否有害是长久以来研究学者们关注的问题。较早期的观点[5]认为克隆代码对软件系统是有害的，导致了程序的规模较大，复杂性较高，难以理解，容易引入软件缺陷等问题，给代码维护增加了难度。随着人们对于克隆代码研究的深入，越来越多的研究表明克隆代码并不一定都是有害的，在有些情况下，克隆代码是非常合理和有效的设计方式[6]，对于不存在缺陷的代码片段进行克隆还可以节省项目开发时间和成本，还可以避免在编写新代码时引入缺陷的可能性[7]。为了探究对系统有害的克隆代码的特性和进化模式，进而对不同类型的克隆代码有针对性的进行维护，人们更加关注于克隆代码管理的工作。

传统的克隆检测结果一般针对单一源代码版本，而克隆进化信息则从发展的角度对克隆代码进行了描述，将离散的信息整合起来，这对于多版本软件系统的管理来说十分重要，开发人员可以分析克隆进化信息从而更加全面的了解克隆代码，更有助于软件缺陷的分析与改善。Laguë等人最早提出克隆进化的概念[8]并进行了相关的研究，随后 Kim 等人基于克隆进化的角度提出了克隆家系及进化模式的概念[9]，并构建了克隆家系提取器，推动了克隆进化方面研究的发展。克隆家系是指一个有向无环图，它描述了一个克隆群的进化情况。进化模式则是指相邻两个版本中相互映射的两个克隆群之间存在的克隆关系，Kim 定义的进化模式包含

了相同模式、增加模式、减少模式、一致变化模式和不一致变化模式。结合克隆进化信息，研究学者对克隆代码进行了进一步的分析与研究，发现积极的重构有时候并不是最好的维护克隆代码方式[10]，对于不稳定的克隆以及长时间存活和一致变化的克隆并不需要重构。因此，不同种类的克隆需要不同的维护支持方式，不能单纯的依靠传统的重构技术，而利用克隆进化的信息可以对克隆管理新方法的研究有所帮助。

基于以上分析，在克隆管理中合理的应用克隆进化信息进行分析是十分的必要。然而在实践过程中发现，大量文本形式的进化数据其实也并不利于研究与观察，所以本文的研究是基于克隆代码进化信息实现克隆家系的可视化及克隆代码的检索工作，将克隆家系的整体结构以及克隆群的演变过程以图形化的形式直观的呈现给用户，支持友好的交互功能，使得用户的检索与查看更加直观与便捷。并在此基础上，应用无监督学习方法对克隆代码进行分析与评价，得到与克隆代码特征相关的结论，为进一步克隆代码的理解与管理工作提供参考。

1.2 国内外研究现状及分析

1.2.1 国外研究现状

目前研究克隆代码的学者和团体较多，已有了一定的研究成果，相关的克隆检测技术、克隆重构技术和克隆代码检测工具都比较成熟，但是克隆进化及克隆代码分析与评价方面的研究相对较少，并且主要以国外的研究为主，研究的主要内容包括以下几个方面：

（1）克隆进化信息的提取技术

包括克隆群的映射、克隆家系的构建、克隆群进化模式的识别等方面内容。华盛顿大学的 Kim 等人在该方面做出了较突出的贡献，在提出克隆家系概念和克隆群进化模式的基础上针对于 Type-1 型克隆与 Type-2 型克隆开发了克隆家系提取器（Clone Ggenealogy Extractor，简称 CEG），他们在所有感兴趣的版本中检测克隆代码，然后在连续的版本间使用启发式规则来进行映射，并利用该工具在 2 个 Java 开源项目中进行了测试[9]。他们的工作成为了后来许多研究的理论基础，许多基于克隆家系模型的研究随后产生。Aversano[11]等人扩展了 Kim 的工作，研究了在进化活动发生时克隆代码如何维护，他们采用的方法是根据源代码库（如 SVN）提供的修改日志对连续的版本进行映射。Thummalapenta[12]采用了与其类似的研究来理解克隆代码的维护方法。Bakota 等人[13]提出了基于 AST 的机器学习方法来映射连续版本间的克隆代码，使用大量的度量值来确定两组克隆群之间的对应关系，时间复杂度较高。萨省大学的 Saha 等人就在此基础上采用更简单的相似度量值，

开发了适应于更多类型软件系统的克隆家系提取器，并应用于 17 个不同编程语言的开源软件系统进行验证[14]。随后又提出了近似克隆家系的框架 gCad[15]，将克隆进化的研究对象扩展到 Type-3 型克隆，他们在检测克隆代码过程中采用的方法是根据各修订版本的源代码变化进行映射[16,17,18]。但该框架仍存在一些局限，例如容易受函数结构变化的影响等。

（2）克隆代码的表示与可视化

传统的方法是用文件名与行号的方式来表示克隆代码，但是实践中发现这一方法对于物理位置过于敏感，在版本间代码位置变化较大时，不利于理解和分析克隆代码。因此出现了克隆区域描述符（Clone Region Descripotor，简称 CRD）[19,20]等其他表示方法，以提高克隆代码的跟踪效率。在可视化方面，基于源代码的可视化方法的相关研究较多，然而在针对克隆代码进化信息的可视化方面的却很少。Adar 和 Kim 开发了 SoftGuess 工具[21]来支持单一和多个版本的克隆代码的可视化，它包含了三个浏览器，分别对克隆家系，克隆片段的继承关系以及家系的依赖关系进行了可视化。Saha 等人针对该工具提出了改进意见[22]，用散布图的方式使得其能适应大型的软件系统，实现了从高层视图到低层视图的过滤效果。

（3）克隆进化分析与有害性评价

Kim 等人 2004 年研究了开发人员为何和如何在软件系统中引入克隆代码[23]。在 2005 年提出利用 CEG 理解克隆代码的方法[10]，并对软件系统中一致变化的克隆、不稳定的克隆、局部不可重构的克隆以及长期存活的克隆是否适合重构进行了讨论。并在 2011 年和香港科技大学的 Cai D 对长期存活的克隆做了更进一步的实证研究[24]，从 7 个大型开源软件系统的实验中总结长期存活克隆的特点，以及其与维护人员相关工作及克隆代码长度等的关系。

J.Krinke 在 2007 年对一致变化和不一致变化的克隆发展情况在 5 个开源软件系统中做了统计与分析[25]。在 2008 年对克隆代码与非克隆代码在软件进化过程中的稳定性进行了分析[26]，得出克隆代码要比非克隆代码更加稳定的结论。在 2011 年对克隆代码与克隆代码存活时间的长短进行了比较，对三个大型开源系统进行研究[27]，结果更加支持了先前得到的结论。

Saha 等人在 2010 年的研究[14]中表示克隆并不都是有害的，应尽可能的跟踪克隆在软件系统中进化情况来管理克隆。随后在 2011 年提出了处理近似克隆的家系框架 gCad 并检测分析了近似克隆家系[28]，并和 Zibran 对近似克隆在进化过程中的变化情况进行了研究[29]，探讨了软件规模与克隆的关系。在 2012 年和 Mondal 等人提出了一种统一的框架，适用于四种不同的克隆检测工具与不同语言的软件系统，来研究克隆代码在软件维护中的影响[30]，并对克隆代码与非克隆代码的稳定

性做出了对比[31]。在 2013 年提出对 Type-3 型克隆应该给予更多的维护[32]，因为其更容易导致不一致变化，而造成软件缺陷。并利用自己开发的克隆家系提取器支持进一步的克隆进化分析[33]。又和 Zibran 基于克隆家系的模式在克隆粒度、进化模式、变化次数等方面对克隆进化特征进行了研究[34]。

除以上在克隆代码进化分析方面较为突出的研究学者和团体以外，还有很多学者也涉及了相关的研究内容，并得出了相应的研究结论。E Dualali 在 2008 年开发工具 Clonetracker[35]来跟踪克隆代码的进化，以支持克隆代码的管理。在 2012 年，Bazrafshan 针对近似克隆的进化情况在 7 个开源软件系统中进行了研究[36]，发现近似克隆在进化过程中变化更多而且多数变化为不一致变化。Bettenburg 等人在发布版本上对不一致变化的克隆代码进行了实证研究[37]，发现只有 1.02%–4.00%的克隆家系会引发软件缺陷，不会对软件质量有明显的影响。Barbour[38]在 3 个跨越多个版本的的软件系统中进行实验，并定义了 8 种后期传播的进化模式，实验结果验证了这种后期传播对于软件系统的危害的严重性。2013 年，Pate 等人[39]对克隆进化的研究进行了回顾与总结，对现有研究中关注的问题进行了提炼。

1.2.2 国内研究现状

国内对于克隆代码的检测与重构方面的研究已有了较多的成果，可是关于克隆进化与克隆分析方面的研究则比较少。复旦大学的王海等人[40]提出了基于分组的增量式克隆检测方法，用来处理多版本的大规模软件系统，使得对于演化过程中的克隆代码检测的代价降低。北京大学的 Wang 等人[41]采用贝叶斯网络的方法对克隆代码的有害性进行了分析和预测。哈尔滨工业大学的慈萌[42]基于克隆区域描述符对克隆家系的提取方法进行了研究，提出了新的克隆群映射算法。而李智超[43]在此基础上采用支持向量机的方法在克隆代码有害性评价方面进行了研究。

1.2.3 国内外研究现状的分析

目前，国内外在克隆代码进化方面的研究思路较为统一，大多数的研究都是在 Kim 提出的克隆家系模型与克隆进化模式的基础上，以改进进化模式的定义，改进克隆群映射算法及如何利用克隆家系提取结果对克隆代码进行分析与维护而展开的。但是目前的研究中还存在着一些局限与不足。

首先是针对于克隆代码进化的可视化研究较少且并不完善。Adar 和 Kim 开发的 SoftGuess 工具虽然实现可视化的基本功能，但是并不适用于大型的软件系统，并且对于克隆家系的可视化形式过于简单，没有充分利用到克隆群映射的信息。而 Saha 等人只是对可视化提出了改进的意见，没有真正实现可视化的工作。多数其他学者的研究中并没有涉及到克隆家系及克隆进化模式的可视化问题。然而在

实际分析代码的过程中，可视化的操作是十分必要的。通过对克隆家系整体的可视化可以对克隆家系的进化情况有个直观全面的了解。另外，如果人工的从克隆家系挑选出感兴趣的版本或者克隆群会十分困难，而有效的克隆代码可视化与检索就可以降低这一工作的难度，使得分析工作更加便捷，这是现有的克隆代码进化可视化工具中所不能完成的。

其次在进行克隆分析时，并没有充分利用到克隆进化方面的信息。现有的克隆分析所依赖的一般都是特定的两组度量值，通过对比他们之间的变化情况而得出结论，并没有考虑到其他克隆度量对于该分析的影响。并且目前克隆分析的方法多数以统计进化数据的方式为主，这种人工分析查找的方式并不利于研究。还有一部分的研究是采用机器学习的方法进行克隆代码的评价，这类的研究一般都基于一个前提：如果克隆群在进化过程中发生了不一致变化，那么该克隆群就是有害的克隆。这主要是根据前期研究中得到的“不一致变化容易导致软件缺陷”的一些相关结论，然而也有研究显示只有少数的不一致变化会引发软件缺陷，显然这样的前提假设并不足以信服。所以，如何充分利用克隆代码的静态信息和进化信息对演化中的克隆代码进行分析还有待进一步研究。

基于以上分析，现有的工具与方法上还存在一定的不足，需要研究更加有效的克隆可视化方法以及克隆分析方法，为进一步的克隆维护和管理工作提供帮助。

1.3 课题研究的主要内容及章节安排

本文的主要研究内容是基于克隆进化信息实现克隆家系的可视化及克隆代码的检索工作，并进一步实现克隆代码的评价。在前期对多版本软件采用 NICAD 克隆检测工具[44]检测出精确克隆代码及近似克隆代码，并应用改进的克隆区域描述符对克隆代码进行建模并构建克隆家系的基础上，实现克隆家系模型的友好的可视化。从构建的克隆家系中提取克隆代码的静态特征及进化特征，对克隆代码的检索，并采用无监督学习方法对克隆代码进行分析与评价，获取对克隆维护有帮助的结论，最终构成一个完整的克隆代码检索与克隆进化分析可视化系统。

本文的各个章节内容安排如下：

第 1 章是绪论，介绍课题的来源，研究的目的与意义，并从克隆进化信息的提取技术，克隆代码的表示与可视化以及克隆进化分析与代码评价三个方面对国内外研究现状进行了介绍与分析，提出现状中存在的问题，并指出本文的主要研究内容。

第 2 章介绍克隆代码检索与克隆进化分析可视化系统的需求分析与系统的总体设计。

第 3 章介绍克隆代码检索与克隆进化分析可视化系统的详细设计，包括模块构成以及各个模块的类图和时序图的设计。

第 4 章介绍克隆代码检索与克隆进化分析可视化系统各个模块的具体实现过程，给出可视化、检索以及聚类算法。

第 4 章是系统的测试与结果的分析，采用不同规模不同语言的开源软件系统进行实验，获取与克隆代码相关的结论。

最后对整体工作进行总结，说明本论文存在的不足和有待改进的问题，并对进一步研究提出建议。

第2章 系统需求分析与总体设计

克隆代码检索与克隆进化分析可视化系统主要是在提取克隆家系的基础上，利用克隆进化信实现克隆进化代码的检索与可视化。在进行检索与可视化之前，需要利用克隆代码检测工具检测出各个版本测试系统中存在的克隆代码，根据克隆群映射算法，将相邻版本之间的克隆群依次进行映射，从而构建完整的克隆家系。本文的工作是以多版本的软件系统源代码文件作为输入，利用克隆家系提取器得到的克隆代码文件、含克隆区域描述符的克隆代码文件、克隆群映射文件以及克隆家系文件实现克隆进化代码的检索与可视化功能。并进一步的利用提取到的态度量和进化度量对克隆代码进行克隆进化分析，采用无监督学习算法得到聚类结果并归纳分析出对代码评价有意义的结论，从而为克隆维护与克隆管理提供帮助。

2.1 系统需求分析

2.1.1 功能性需求

本文的主要工作是基于克隆进化信息设计并实现一个克隆代码检索与克隆进化分析可视化系统。总体来说包含以下功能：

（1）结合克隆家系提取器得到的信息，对克隆家系与克隆群映射关系进行可视化，展示克隆代码的进化情况，提供友好的交互功能。

（2）提取克隆代码的特征度量，其中包括静态度量与进化度量。

（3）实现克隆代码检索功能，这其中包括了检索特定度量范围内的克隆代码片段和检索指定的感兴趣的克隆代码两方面的功能。

（4）对含有多维度量值的克隆代码样本集合进行无监督学习，分析学习结果，归纳总结对代码分析与评价有意义的相关结论。

2.1.2 非功能性需求

系统应满足的以下非功能性需求：

（1）性能需求：系统稳定，故障率较低，且运行效率较快，对于所有的请求能够在允许的时间范围内得到响应。

（2）易用性：在用户无任何操作经验的前提下，能够在短时间了解系统的全部功能，工具栏及界面的设计简单易懂，并提供友好的交互界面。

（3）适用性：适用于不同操作系统的运行环境，所占存储空间适当，可处理大型软件系统多个版本的源代码文件。

（4）集成性：集成克隆家系提取器、克隆代码有害性评价与本文各项功能。

（5）可靠性：由于用户操作不当而导致系统崩溃的概率较低，即时系统发生故障重新启动后也不会造成重大影响。

（6）安全性：系统安全，不会对运行环境造成危害。

（7）可扩展性：可以利用本文系统扩展其他功能的补充与完善。

2.2 系统功能设计

系统的用例图如图 2-1 所示。系统主要包括克隆代码可视化、度量值提取、克隆代码检索和克隆进化分析四个模块。其中，克隆代码可视化提供的功能有克隆家系整体的可视化视图和单个克隆家系的详细可视化视图以及相应的用户交互功能；克隆代码检索提供的功能有对于指定度量范围内的克隆代码的检索和对于指定代码片段的检索以及相应的用户交互功能；克隆进化分析的功能包括无监督学习和样本分析归纳结论。

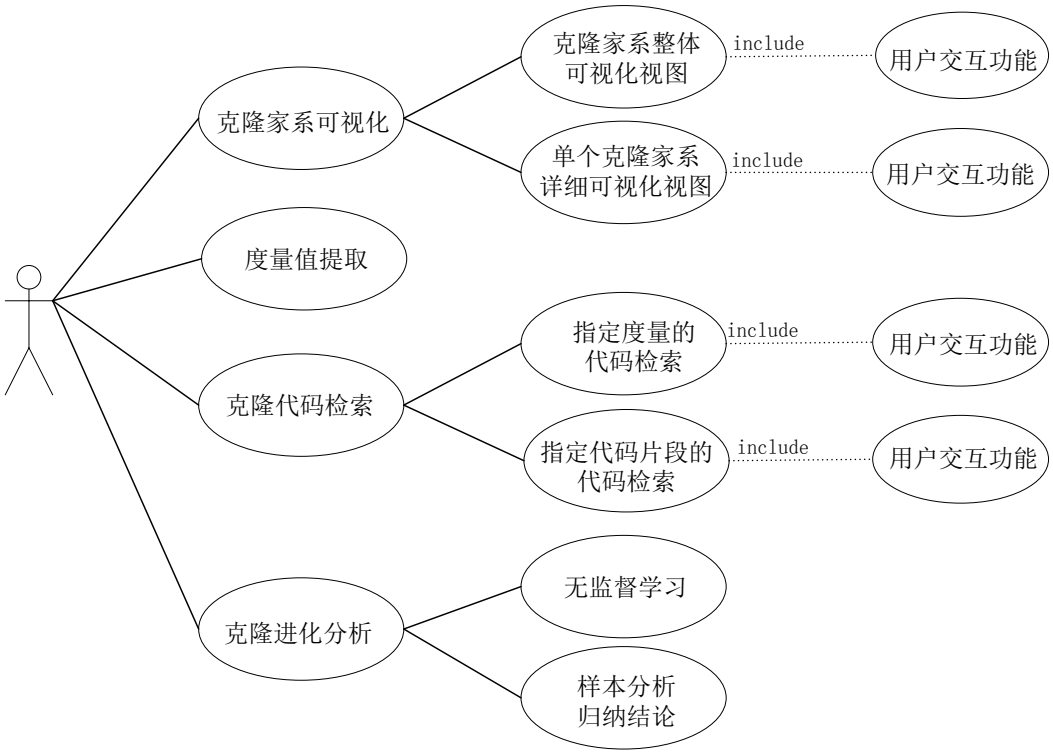


图 2-1 系统用例图

（1）克隆家系可视化：采用 Windows 图形设备接口 GDI+来实现克隆家系的可视化效果。GDI+主要是用来负责 Windows 系统与相关绘图程序间的信息交换。它提供的接口比较灵活，使用核心类 Graphics 可以完成基本的画图需求。为将克

隆家系的整体结构以及克隆进化模式的演变过程直观的呈现给用户，应向用户提供友好的交互式界面，并支持在可视化界面中有效的获取克隆进化信息。本文的可视化工作主要包括克隆家系整体视图、单个克隆家系详细视图及相关的交互功能等。

（2）度量值的提取：为了进一步实现对克隆代码的检索及对克隆代码进行分析评价工作，本文在对克隆代码提取必要的静态度量值的基础上，还利用构建的克隆家系提取了克隆代码的进化信息。克隆代码的静态度量值及进化度量构成克隆代码的特征集合，主要包括：四种常用的 Halstead 度量（操作符种类、操作数种类、操作符总量、操作数总量）、克隆粒度、克隆相似度、参数个数、克隆寿命、变化复次数及进化模式等。

（3）克隆代码检索：对于克隆检测工具检测出的大量结果，程序员并不容易理解和操作，克隆代码检索可以使得程序员有效的提取所需的代码片段，从而对这些指定的克隆代码做更深入的研究。本文支持两方面的克隆检索工作：一方面，对于给定的克隆代码片段，在检测结果中根据文本相似度查找与其互为克隆的代码片段；另一方面，允许用户自己设置所需代码片段的特征值，系统根据特征值信息检索符合条件的克隆代码片段。

（4）克隆进化分析：由于代码有害性并没有明确的定义，本文尝试采取无监督学习的方式，对克隆代码片段进行聚类。利用多版本源代码片段及提取的度量信息集合，进行样本聚类（采取模糊 C 均值聚类方法），试图从聚类结果中分析得到对代码评价有意义的信息，为进一步消除软件潜在缺陷提供有效指导。

（5）克隆代码检索与克隆进化分析可视化系统：结合克隆家系提取器和以上功能，实现一个以软件版本库为输入，提供克隆家系可视化、克隆代码检索及克隆代码评价的完整的代码检索与克隆进化分析可视化系统。以多个开源软件系统的不同版本文件作为实验对象，验证系统的有效性。所选取的测试系统为不同的规模，不同语言编写的开源软件。

2.3 系统体系结构

系统的整体体系结构层次图如图 2-2 所示， 从上到下依次分为表示层、控制层、功能层、传输层、数据层。

（1）表示层：表示层是面向用户的体系结构层，用户可以通过载入数据，在操作界面中控制不同请求按钮来引发相对应的功能，不同功能会将最终处理数据的结果反馈给表示层，使得用户得到输出数据。

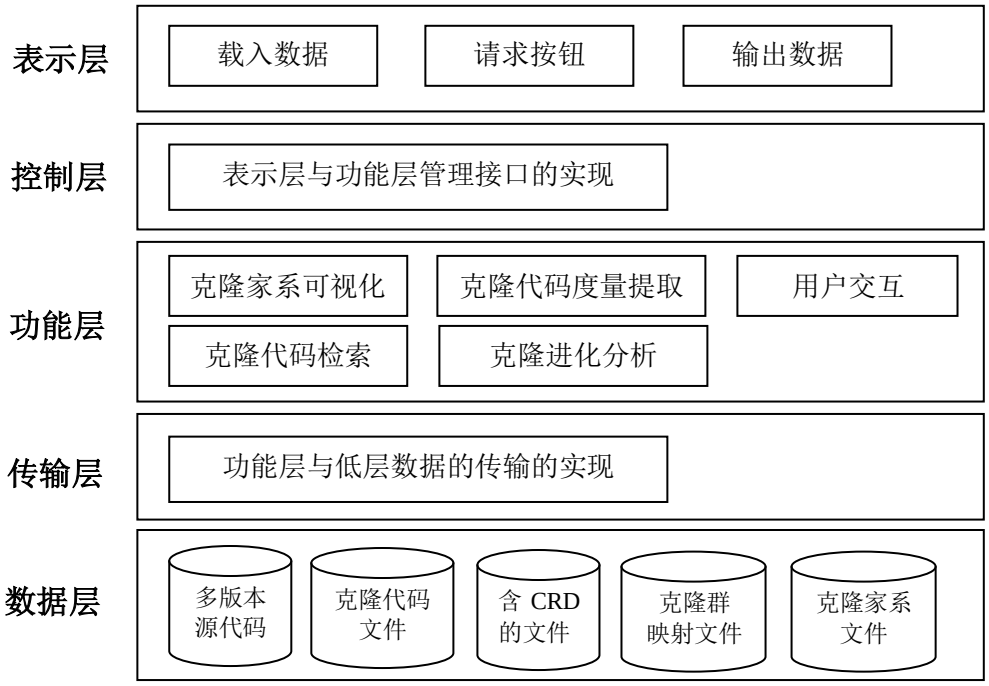


图 2-2 系统体系结构层次图

（2）控制层：控制层负责表示层与功能层之间的管理，对表示层的请求进行响应，并将功能层的结果反馈给表示层。

（3）功能层：功能层是系统的核心所在，对整个克隆代码检索与可视化系统起到了关键作用。主要实现了各个模块的具体功能，包括克隆家系的可视化、克隆代码度量值的提取、克隆代码的检索、克隆代码的进化分析以及相应的用户交互功能。功能层接收用户所选择的输入文件，通过传输层调用相关数据文件以响应用户请求，最终将输出数据发送给控制层，进一步反馈给用户。

（4）传输层：传输层负责数据层与功能层之间的数据交换与传输。主要是向功能层提供底层数据文件的读取，是功能层与数据层的纽带。

（5）数据层：数据层是整个系统的基础所在。数据层为功能层的不同模块提供相应的数据文件，其中主要包含了以下五种形式的数据文件。

a．多版本源代码文件：目标系统的不同版本的源代码文件，用户可以自行指定和选取，是克隆代码检测的最基础数据。

b．克隆代码 xml 文件：通过 NICAD 克隆代码检测工具对源代码文件进行检测的结果文件，每个版本的源代码文件对应一个克隆代码 xml 文件，文件中的信息按照克隆群序号依次显示，其中包括了克隆群中代码片段个数、克隆相似度、克隆代码片段所属文件、起止行号以及代码片段序号等。

c．含 CRD 的克隆代码 xml 文件：即添加了克隆区域描述符的克隆代码 xml 文件，以层次化的形式反映了克隆代码在文件、类、方法、块等不同层的特征。

d．克隆群映射 xml 文件：两个相邻的软件版本对应一个克隆群映射文件，表示了两个版本中克隆群相互映射的关系，其中包含了映射的源文件信息和目标文件信息、克隆群映射序号、映射的级别以及进化模式等内容。

e．克隆家系 xml 文件：基于相邻版本的克隆群映射关系构建的克隆家系表示文件，克隆家系文件中包含若干克隆直系，每个克隆直系通过进化序列号来表示映射关系，其中包含了源文件名和目标文件名的相关信息以及进化模式。

2.4 系统模块设计

本文的克隆代码检索与克隆进化分析可视化系统主要包括四大模块：可视化模块，度量值提取模块，代码检索模块和克隆进化分析模块。系统的模块图如图 2-3 所示。

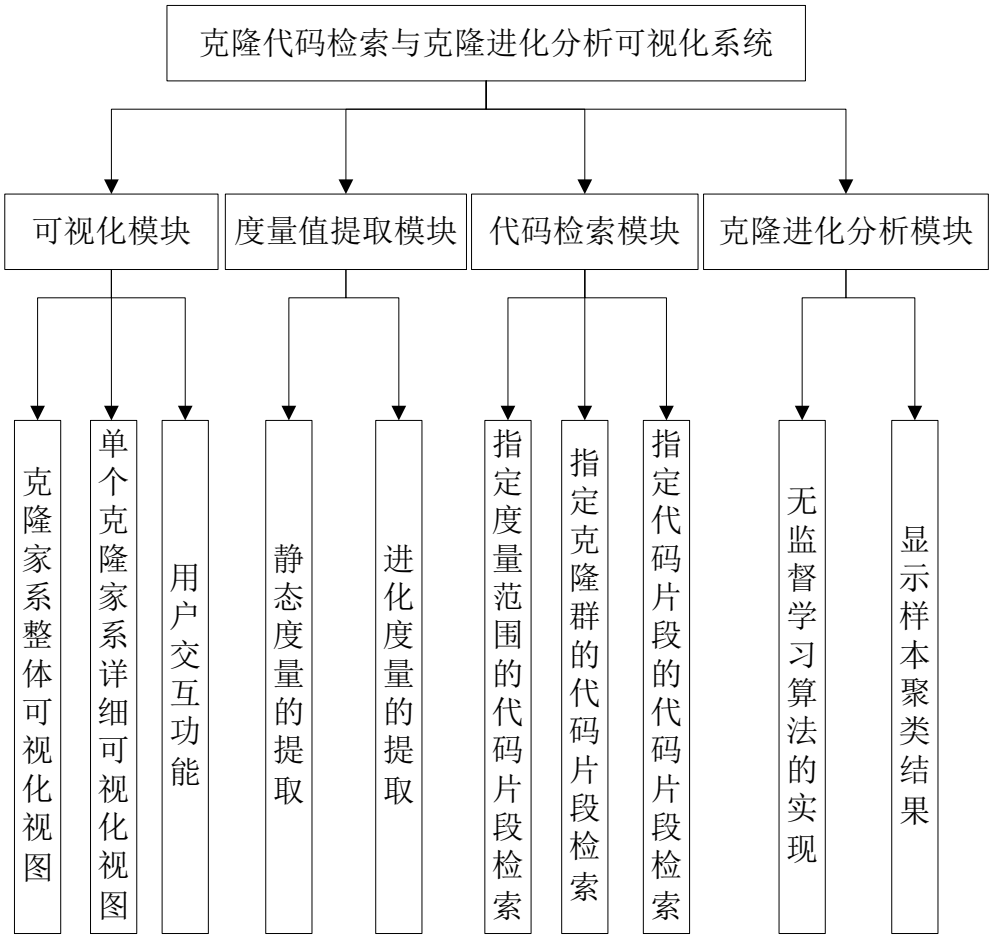


图 2-3 系统模块图

（1）可视化模块：将含 CRD 的克隆代码 xml 文件、克隆群映射 xml 文件以及克隆家系 xml 文件作为克隆家系可视化模块的输入文件，克隆家系可视化模块主要实现克隆家系整体可视化视图、含克隆片段映射关系的单个克隆家系详细视图以及添加相应的用户交互功能，展现克隆群的产生、发展直至消失的整个生命过程，帮助用户理解克隆代码在版本更新过程中的演化情况，并且多方面的交互功能使得用户对克隆家系信息的查看更加直观与便捷。

（2）度量值提取模块：克隆代码度量选择与提取是克隆代码检索与代码分析评价的基础，度量信息从克隆家系提取器生成的含 CRD 的克隆代码 xml 文件、克隆群映射 xml 文件以及克隆家系 xml 文件中获取，并以文本形式存储克隆代码度量，作为克隆代码检索的依据以及评价分析中的样本文件。本文所采用的克隆代码度量包括静态度量与进化度量两方面，分别从单一版本文件和多个版本文件的不同角度描述了克隆代码特征。

（3）代码检索模块：本文的克隆代码检索功能支持两种方式的检索。一方面，通过对用户感兴趣的指定克隆代码片段文件与以克隆群为单位的代码片段进行比较，根据比较其文本相似度的大小检索与指定代码片段互为克隆的代码片段。另一方面，基于已提取的克隆代码度量值文件，通过设置所需的代码片段的度量值进行检索，按度量而非代码文本对克隆代码进行检索和分析。

（4）克隆进化分析模块：利用多版本代码片段及已提取的度量值信息，以代码片段作为样本对其进行聚类，分析聚类结果，试图从聚类结果中分析得到对代码评价有指导意义的结论。

2.5 系统流程设计

2.5.1 系统主体流程

本文的克隆代码检索与克隆进化分析可视化系统整体框架如图 2-4 所示。采用 NICAD 克隆检测工具对多版本软件中精确克隆及近似克隆代码进行检测，用改进的克隆区域描述符表示克隆代码信息，并在完成相邻版本间克隆群的映射的基础上依照进化模式的定义识别进化模式，从而构建克隆家系并实现克隆家系模型的友好可视化。从构建的克隆家系提取克隆代码的静态特征及进化特征，结合这些特征实现对克隆代码的检索，并采用无监督学习方法对克隆代码进行进一步分析评价，最终构成一个完整的克隆代码检索与克隆进化分析可视化系统。

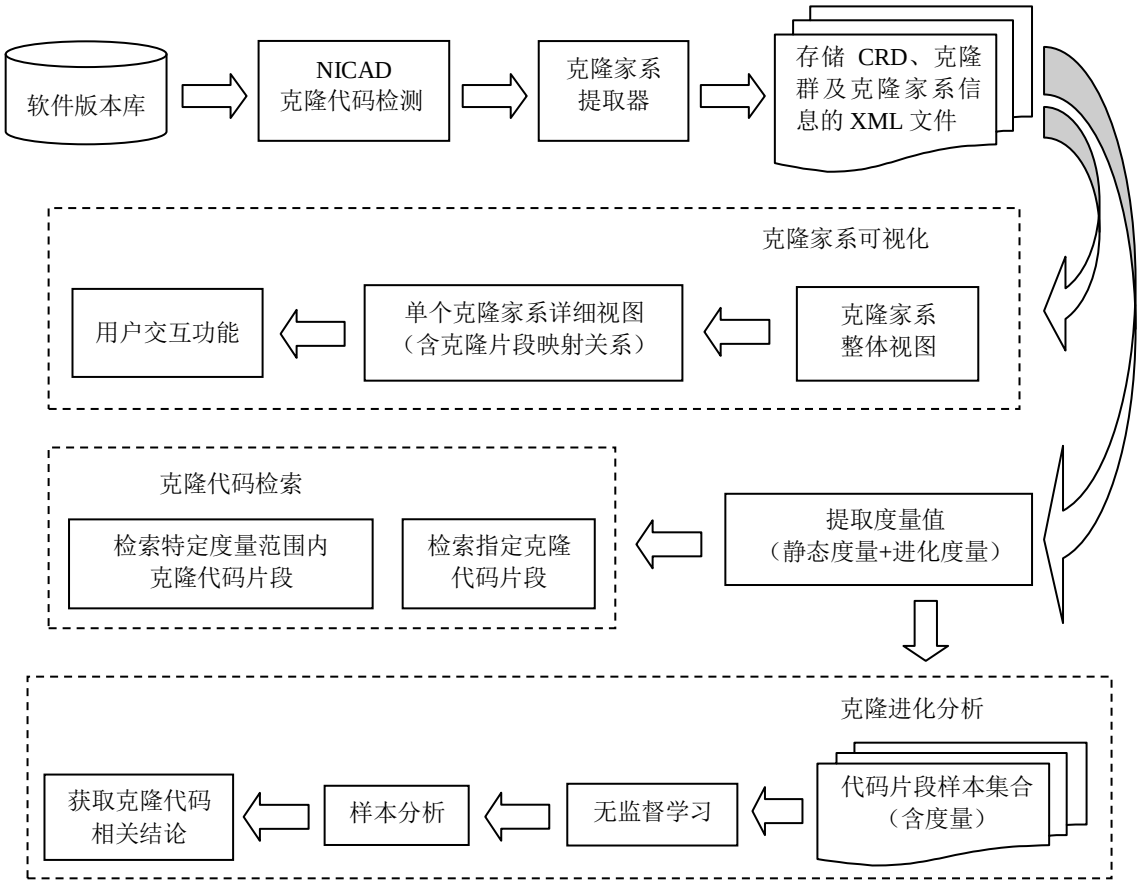


图 2-4 系统总体框架图

2.5.2 代码可视化流程设计

本文的代码可视化的工作包含两方面的内容，分别为克隆家系的整体可视化视图和单个克隆家系的详细可视化视图。克隆家系整体可视化展示了各个克隆家系从克隆群产生、发展变化直至消失的整个生命过程，形象直观地展示了克隆家系的整体演化情况。而单个克隆家系展示了克隆家系内部克隆代码片段的映射关系，为进一步对克隆代码片段的研究提供帮助。其中添加了相应的用户交互功能，包括查看 xml 文件、获取代码片段信息、查看源代码、查看源代码片段、对比代码片段、放大缩小视图等。

克隆家系的整体可视化视图流程如下所述：

（1）用户加载克隆家系 xml 文件；

（2）选择克隆家系整体可视化的按钮；

（3）执行生成可视化视图算法；

（4）用户界面显示克隆家系的整体视图。

单个克隆家系的详细可视化视图流程如下所述：

（1）用户加载含 CRD 的克隆代码 xml 文件和克隆群映射 xml 文件；

（2）选择某一感兴趣的克隆家系文件，或从整体可视化视图中选择某一感兴趣的克隆家系；

（3）选择单个克隆家系可视化的按钮，或双击整体可视化视图中的节点；

（4）执行生成可视化视图算法；

（5）用户界面显示单个克隆家系的详细视图。

2.5.3 度量值提取流程设计

度量值提取是实现代码检索功能和代码无监督学习功能的基础，用户必须获取克隆代码片段的度量值文件才可使用其后续功能。其中，度量值包含了静态度量和进化度量两部分，静态度量需要从源代码中获取，而进化度量是从克隆群映射文件中获取。度量值提取的流程如下所述：

（1）用户加载含 CRD 的克隆代码 xml 文件；

（2）选择预处理按钮，执行预处理操作，在原克隆代码文件中添加度量值信息，这里主要是从源代码中获取的静态度量；

（3）用户加载克隆群映射 xml 文件；

（4）选择提取度量按钮，利用改进的克隆代码文件和克隆群映射文件，执行度量值提取算法获取各个度量值。

2.5.4 代码检索流程设计

本文中系统的代码检索主要包含两种形式：第一种是对指定的当前多版本软件中的代码片段或从外界载入的代码片段利用文本相似度进行检索；第二种是利用上述提取到的度量值，对感兴趣的度量范围内的代码片段进行检索。代码检索可以帮助用户快速查找到感兴趣的代码片段进行分析，避免了人工查找速度慢、容易遗漏等缺陷。代码检索的流程如下所述：

方式一：

（1）用户加载多版本源代码文件和含 CRD 的克隆代码 xml 文件；

（2）选择指点代码片段检索按钮；

（3）从外界加载指定的克隆代码片段文件，或在当前多版本软件中设置指定克隆群中的代码片段；

（4）执行克隆代码检索算法；

（5）在用户界面显示克隆代码检索结果，并支持代码查看和对比功能。

方式二：

（1）用户加载克隆代码度量值文件；

（2）选择指定度量值检索按钮；

（3）设置感兴趣的检索克隆代码的各项度量值范围；

（4）执行克隆代码检索算法；

（5）在用户界面显示克隆代码检索结果，并支持代码查看和对比功能。

2.5.5 克隆进化分析流程设计

克隆进化分析工作分为两个部分，一部分是系统自动完成的，另一个部分需要人工进行分析处理。系统自动完成的是对克隆代码片段样本集合进行无监督学习的过程，而对于学习的结果则需要人工进行进一步的分析和总结。克隆进化分析的流程如下所述：

（1）加载克隆代码样本集合，即已提取的克隆代码度量值文件；

（2）选择无监督学习按钮；

（3）执行无监督学习算法；

（4）在用户界面显示学习结果，支持代码查看功能，同时输出结果文件；

（5）利用学习结果对克隆代码样本各个度量之间的关系进行人工分析，归纳总结出相应的有意义的结论。

2.6 本章小结

本章首先分析了本文克隆代码检索与克隆进化分析可视化系统的功能与非功能需求，然后根据系统需求，从系统功能、系统体系结构、系统模块、系统流程四个方面对系统进行了概要设计。系统的体系结构包括了表示层、控制层、功能层、传输层及数据层。整体架构中包含了四个模块，分别为可视化模块，度量值提取模块，代码检索模块和克隆进化分析模块，并给出了每个模块对应的工作流程。

第3章 克隆代码检索与克隆进化分析可视化系统的设计

3.1 可视化模块的设计

3.1.1 代码可视化模块的类图设计

代码可视化模块主要负责显示两种不同视图下的克隆家系情况，克隆家系整体可视化视图的类图设计如图 3-1 所示，单个克隆家系的详细可视化视图的类图设计如图 3-2 所示。

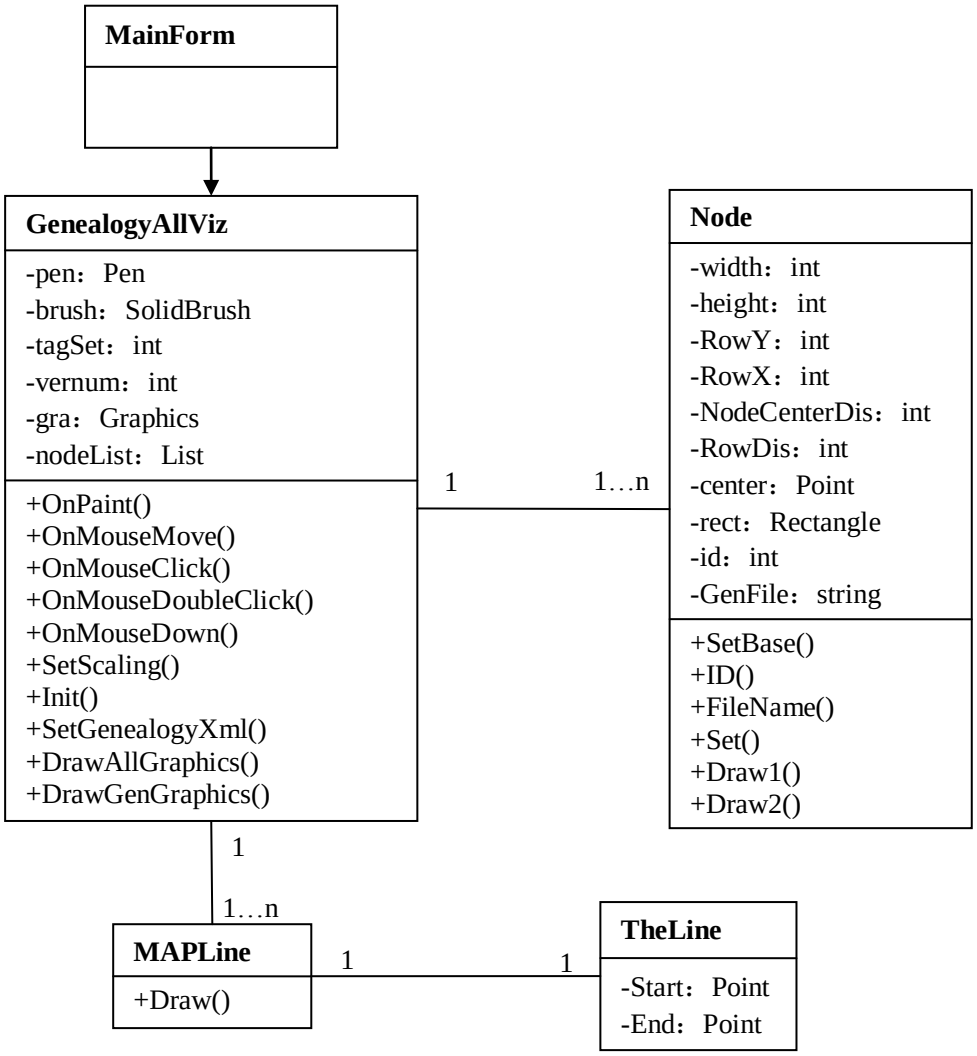


图 3-1 克隆家系整体可视化类图

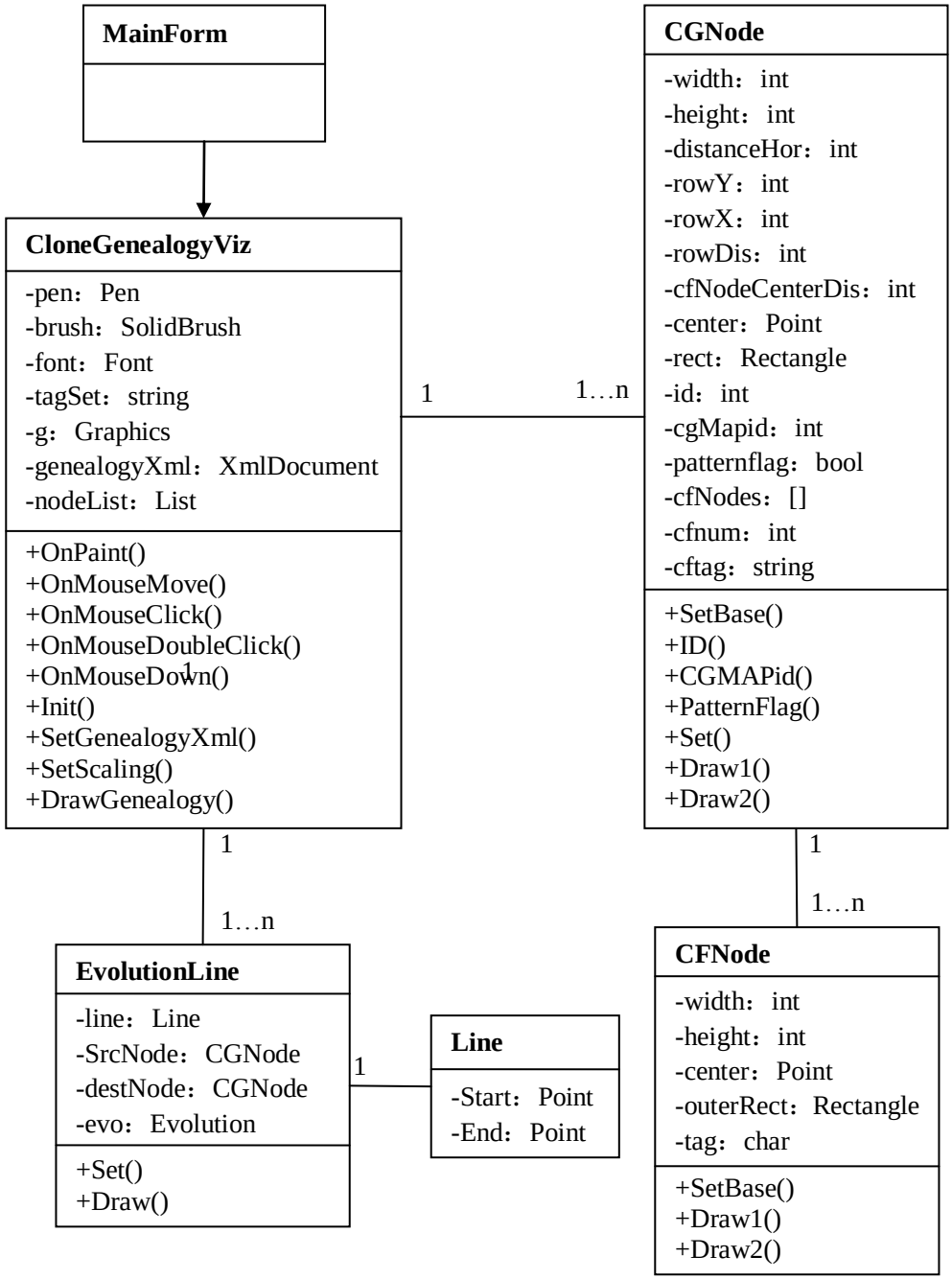


图 3-2 单个克隆家系可视化类图

在整体可视化视图的设计中，包含了五个主要类，类的详细功能设计与主要变量、函数的描述如下。

MainForm 类负责整个代码可视化与检索系统的界面显示控制，用户在界面中的请求直接传递给此类，再通过函数或参数传递来控制其他类执行相关操作。在

该模块中需要用到的主要是类中的 VisualizeAll_Click()函数，控制系统执行整个克隆家系可视化操作。

GenealogyAllViz 类负责执行克隆家系整体可视化的所有操作，主要变量与函数介绍如下：

（1）pen 为画笔，用来描绘克隆群节点的边框和克隆群之间的连接线；brush 为画刷，用来填充克隆群节点内部区域；tagSet 为标签集合，用来记录克隆群的个数；vernum 用来记录版本号；gra 为画图所需的对象；nodeList 用来存储所有克隆群节点信息。

（2）OnPaint()函数：执行绘图操作，同时控制滚动条的添加与缩放功能。

（3）OnMouseMove()函数：移动鼠标时的操作。

（4）OnMouseClick()函数：单击鼠标时的操作。当鼠标左键被按下且鼠标指向某一克隆群节点时，则在信息显示区域给出该克隆群的相关的克隆家系信息及进化信息，其中包括克隆家系序号、起止版本、克隆寿命、进化模式信息等。当鼠标右键按下或鼠标指向空白区域时，则清空信息显示区域。

（5）OnMouseDoubleClick()函数：双击鼠标时的操作。当鼠标左键被按下且鼠标指向某一克隆群节点时，则跳转至该克隆群节点所在克隆家系的详细视图。当鼠标右键按下或鼠标指向空白区域时，不执行任何操作。

（6）OnMouseDown()函数：鼠标按下时的操作。

（7）SetScaling()函数：设置视图缩放功能。

（8）Init()函数：初始化画笔画刷画布等。

（9）SetGenealogyXml()函数：加载所有克隆家系文件。

（10）DrawAllGraphics()函数：绘制视图中所有克隆家系。

（11）DrawGenGraphics()函数：绘制视图中的单个克隆家系。

Node 类负责设置视图中克隆群节点的相关信息。主要变量与函数介绍如下：

（1）width 为节点宽，height 为节点高，RowX 为当前绘图中最后一列节点的横坐标，RowY 为当前绘图中最后一行节点的纵坐标，NodeCenterDis 为节点中心的垂直距离，RowDis 为每行节点的距离，center 为节点中心点，rect 为节点所在区域，id 为节点所表示的克隆群序号，GenFile 为节点所在的克隆家系文件名。

（2）SetBase()函数：设置节点的基本信息，包括宽、高、相关距离值等。

（3）ID()函数，FileName()函数：获取克隆群序号和克隆家系文件名。

（4）Set()函数：设置绘制克隆群节点的位置信息。

（5）Draw1()函数：绘制发生一致变化的克隆群节点。

（6）Draw2()函数：绘制发生不一致变化的克隆群节点。

MAPLine 类负责设置视图中克隆群节点之间的连接线信息。主要的函数为 Draw()，根据克隆群前后是否发生更改来绘制连接线。

TheLine 类负责标记连接线的起始点和终止点。

在单个克隆家系详细可视化视图的设计中，包含了六个主要类，类的详细功能设计与主要变量、函数的描述如下。

MainForm 类在该模块中主要发挥作用的是 VisualizeGenealogy_Click()函数，响应显示单个克隆家系详细视图的请求，控制系统执行相关操作，请求可以通过双击整体可视化视图中的克隆群节点或单击按钮完成。

CloneGenealogyViz 类负责执行单个克隆家系可视化的所有操作，主要变量与函数介绍如下：

（1）pen 为画笔，用来描绘克隆群节点的边框和克隆群之间的连接线；brush 为画刷，用来填充克隆群节点内部区域；font 为标签字体，用来标示克隆代码片段；tagSet 为标签集合，用来记录克隆群的个数；g 为画图所需的对象；genealogyXml 存储克隆家系文件；nodeList 用来存储所有克隆群节点信息。

（2）OnPaint()函数：执行绘图操作，同时控制滚动条的添加与缩放功能。

（3）OnMouseMove()函数：移动鼠标时的操作。

（4）OnMouseClick()函数：单击鼠标时的操作。当鼠标指向某一代码片段节点时，则高亮显示该节点。如果鼠标左键被按下，则在信息显示区域给出该代码片段所在克隆家系信息、进化信息及节点相关信息，其中包括克隆家系序号、开始版本、结束版本、克隆寿命、进化模式信息、节点所在版本号、克隆群序号、克隆群所含代码片段个数、克隆群映射序号、克隆片段序号等。如果鼠标右键按下，则显示扩展交互功能，包括代码片段的显示、完整代码片段的显示及当前版本代码片段与前一个版本的代码片段的差异对比结果的显示。当鼠标指向空白区域时，则清空信息显示区域。

（5）OnMouseDoubleClick()函数：双击鼠标时的操作。当鼠标左键被按下且鼠标指向某一克隆群节点时，则跳转至相应版本的含 CRD 的 XML 文件树形结构显示图，其中所选代码片段高亮显示，点击“+”展开即为代码片段以 CRD 表示的文件-类-方法-块信息，右键即可显示扩展交互功能，包括代码片段的显示、完整代码片段的显示。

（6）OnMouseDown()函数：鼠标按下时的操作。

（7）SetGenealogyXml()函数：加载所有克隆家系文件。

（8）SetScaling()函数：设置视图缩放功能。

（9）Init()函数：初始化画笔画刷画布等。

（10）DrawGenealogy()函数：绘制视图中所有克隆家系。

CGNode 类负责设置视图中克隆群节点相关信息。主要变量与函数介绍如下：

（1）width 为节点宽，height 为节点高，RowX 为当前绘图中最后一列节点的横坐标，RowY 为当前绘图中最后一行节点的纵坐标，distanceHor 为克隆群节点中心的垂直距离，rowDis 为每行节点的距离，cfNodeCenterDis 为代码片段节点的中心的垂直距离，center 为节点中心点，rect 为节点所在区域，id 为节点所表示的克隆群序号，cgMapid 为克隆群映射序号，patternflag 为进化模式标志，cfNodes 为克隆群内代码片段集合，cfnums 为克隆群内代码片段个数，cftag 为克隆群内代码片段标号。

（2）SetBase()函数：设置节点的基本信息，包括宽、高、相关距离值等。

（3）ID()函数，CGMapid()函数，PatternFlag()函数：获取克隆群序号、克隆群映射序号和进化模式标志。

（4）Set()函数：设置绘制克隆群节点的位置信息。

（5）Draw1()函数：绘制发生一致变化的克隆群节点。

（6）Draw2()函数：绘制发生不一致变化的克隆群节点。

CFNode 类负责设置视图中克隆群内代码片段节点相关信息。主要变量与函数介绍如下：

（1）width 为节点宽，height 为节点高，center 为节点中心点，outerRect 为节点所在区域，tag 为克隆群内代码片段标号。

（2）SetBase()函数：设置节点的基本信息，包括宽、高、相关距离值等。

（3）Draw1()函数：绘制发生一致变化的代码片段节点。

（4）Draw2()函数：绘制发生不一致变化的代码片段节点。

EvolutionLine 类负责设置视图中克隆群节点之间的连接线信息。主要变量与函数介绍如下：

（1）line 为相邻节点之间的连接线，srcNode 和 destNode 为相互映射的源代码片段和目标代码片段，evo 为克隆群之间的映射关系。

（2）Set()函数：设置相邻版本内相互映射的代码片段。

（3）Draw()函数：根据克隆代码片段前后是否发生更改来绘制连接线。

Line 类负责标记连接线的起始点和终止点。

3.1.2 代码可视化模块的时序图设计

代码可视化模块主要分为两个部分，克隆家系整体可视化的时序图如图 3-3 所示，单个克隆家系的详细可视化的时序图如图 3-4 所示。

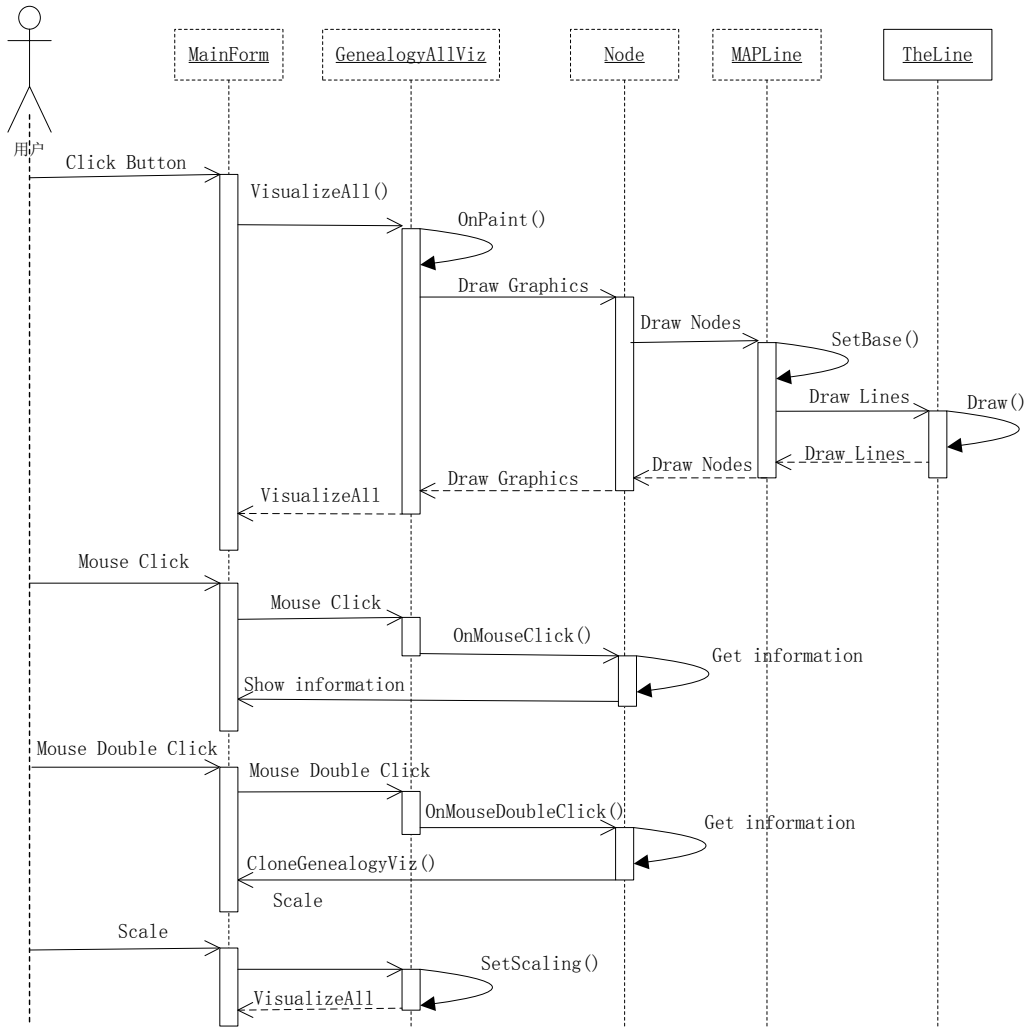


图 3-3　克隆家系整体可视化时序图

克隆家系整体可视化的工作过程的详细描述如下：

（1）用户按动界面中的克隆家系整体可视化请求的按钮，MainForm 类接收请求调用 VisualizeAll()函数，执行 GenealogyAllViz 类中相关的算法。算法根据当前目录中的克隆家系文件信息构建克隆家系整体可视化视图，描绘视图的操作由 OnPaint()函数触发，画图的过程中根据克隆家系文件信息先调用 Node 类描绘节点，再根据克隆群映射关系调用 MAPLine 类将该节点与已描绘的节点连线，所调用的函数中还包括了节点与连线位置信息的设置。将所绘制的图形信息一步一步返回至用户界面，最终显示出克隆家系整体可视化的结果。

（2）在用户界面中单击鼠标节点，如果鼠标指向的为某一克隆群节点，则执行 GenealogyAllViz 类中的 OnMouseClick()函数，调用 Node 类获取克隆群节点相关的进化信息，并返回至用户界面。

（3）在用户界面中双击鼠标节点，如果鼠标指向的为某一克隆群节点，则执行 GenealogyAllViz 类中的 OnMouseDoubleClick()函数，调用 Node 类获取克隆群节点相关的进化信息，将克隆群所在的克隆代码 xml 文件以树形结构的形式返回至用户界面。

（4）用户使用快捷键进行可视化视图缩放功能时，调用 GenealogyAllViz 类中的 SetScaling()函数进行缩放，并将缩放后的视图返回至用户界面。

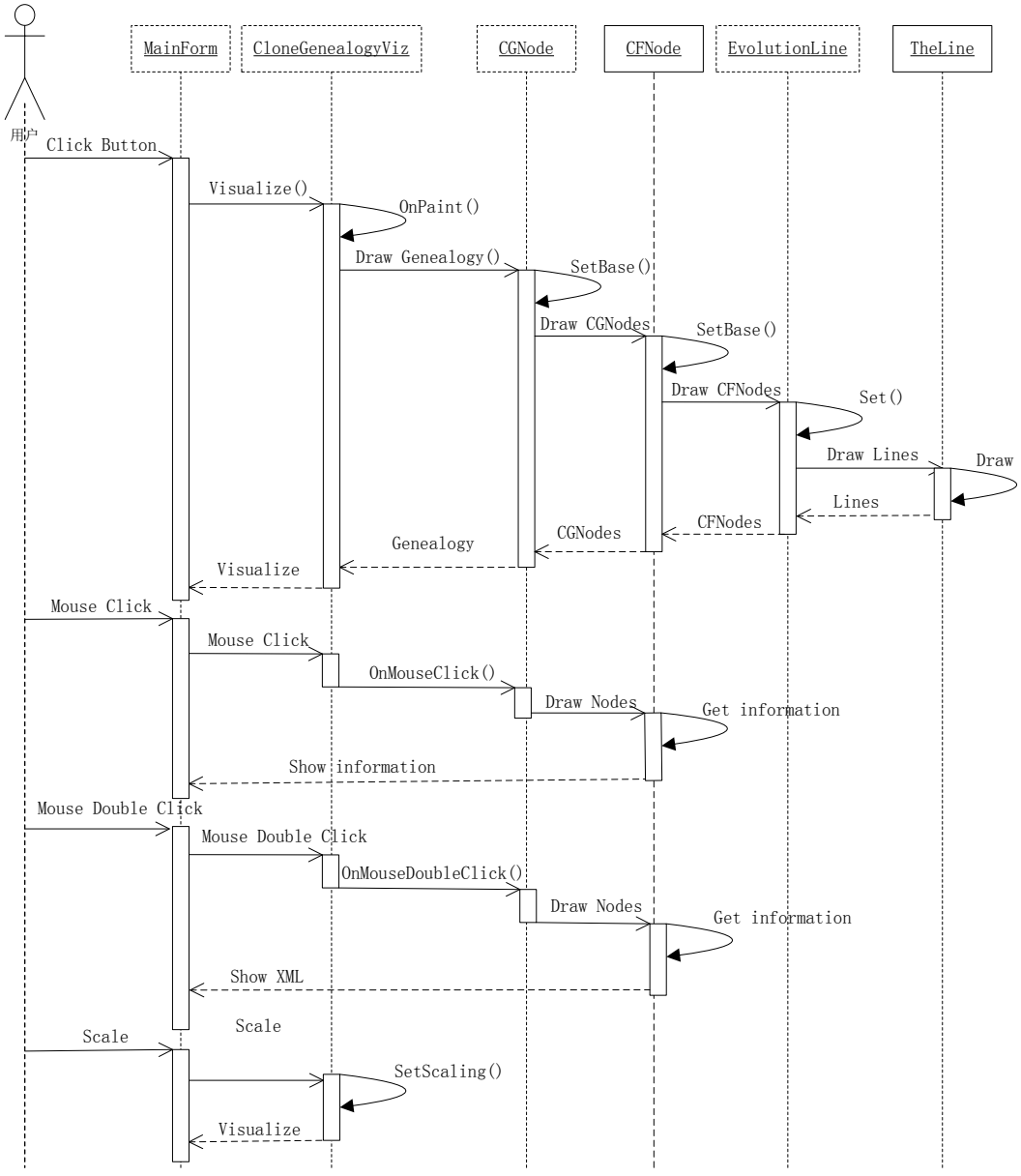


图 3-4　单个克隆家系可视化时序图

单个克隆家系可视化的工作过程的详细描述如下：

（1）用户按动界面中的单个克隆家系可视化请求的按钮，MainForm 类接收请求调用 Visualize ()函数，执行 CloneGenealogyViz 类中相关的算法。算法根据当前选中的单个克隆家系文件信息构建克隆家系可视化视图，描绘视图的操作由 OnPaint()函数触发，画图的过程中根据克隆家系文件信息先调用 CGNode 类描绘克隆群节点，再根据克隆代码文件信息调用 CFNode 类描绘克隆群中的代码片段节点，然后根据克隆群映射关系调用 EvolutionLine 类将当前代码片段节点与已描绘的代码片段节点连线，所调用的函数中还包括了节点与连线位置信息的设置。将所绘制的图形信息一步一步返回至用户界面，最终显示出单个克隆家系的可视化结果。

（2）在用户界面中单击鼠标节点，如果鼠标指向的为某一克隆群节点，则执行 CloneGenealogyViz 类中的 OnMouseClick()函数，判断是鼠标的左键按下还是右键按下，调用 CFNode 类获取克隆代码片段节点相关的进化信息，如果是鼠标左键被按下，则将代码片段信息返回至用户界面；如果是鼠标右键被按下，则显示选择菜单，使得用户进一步执行其他扩展的交互功能。

（3）在用户界面中双击鼠标节点，如果鼠标指向的为某一克隆群节点，则执行 CloneGenealogyViz 类中的 OnMouseDoubleClick()函数，调用 CFNode 类获取克隆群节点相关的进化信息，将代码片段所在的克隆代码 xml 文件以树形结构的形式返回至用户界面。

（4）用户使用快捷键进行可视化视图缩放功能时，调用 CloneGenealogyViz 类中的 SetScaling()函数进行缩放，并将缩放后的视图返回至用户界面。

3.2 度量值提取模块的设计

3.2.1 度量值提取模块的类图设计

度量值提取模块主要负责提取所有克隆代码片段的各个度量值并存储。该模块的类图设计如图 3-5 所示，包含了三个主要类，类的详细功能设计与主要变量、函数的描述如下。

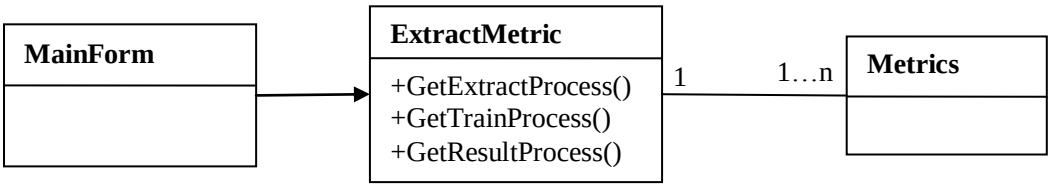


图 3-5 度量值提取模块类图

MainForm 类在该模块中主要发挥作用的是 ExtractCodeMetrics_Click()函数，响应用户在界面中单击提取度量值按钮的请求，控制系统执行相关操作。

ExtractMetric 类负责执行提取度量值的所有操作，主要函数介绍如下：

（1）GetExtractProcess()函数：执行度量值提取的操作。

（2）GetTrainProcess()函数：执行数据集训练的操作。

（3）GetResultProcess()函数：显示训练的结果，计算各项相关数值。

Metrics 类中包含了所有度量值的定义和提取算法。

3.2.2 度量值提取模块的时序图设计

度量值提取模块的时序图如图 3-6 所示。首先用户按动界面中的度量值提取请求的按钮，MainForm 类接收请求调用 ExtractCodeMetrics()函数，执行 ExtractMetric 类中相关的算法。算法根据当前目录中的克隆代码文件及克隆群映射文件，调用 GetExtractProcess() 函数提取度量值并存储至 Metrics 类中。最后调用 GetResultProcess()函数将度量值提取的结果返回至用户界面。

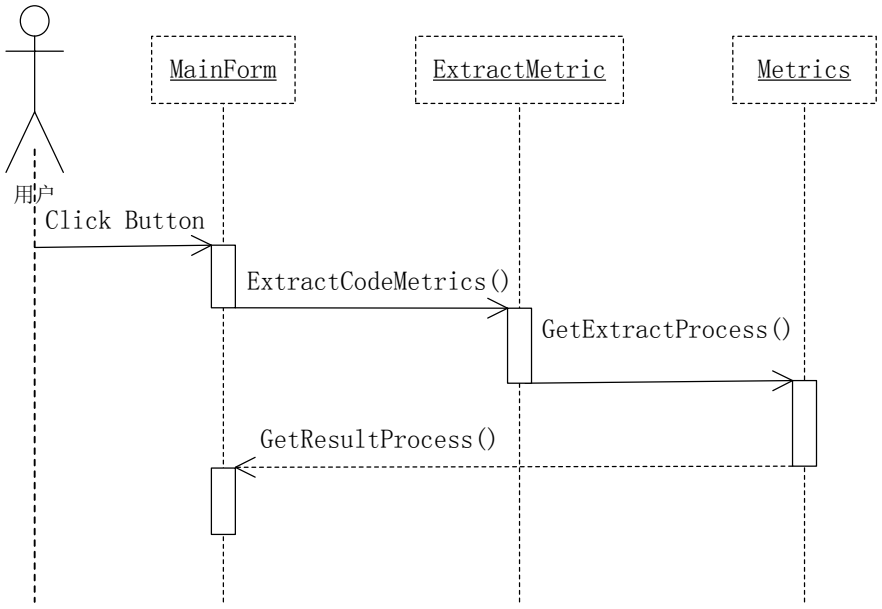


图 3-6　度量值提取模块时序图

3.3 代码检索模块的设计

3.3.1 代码检索模块的类图设计

代码检索模块主要负责。该模块的类图设计如图 3-7 所示，包含了四个主要类，类的详细功能设计与主要变量、函数的描述如下。

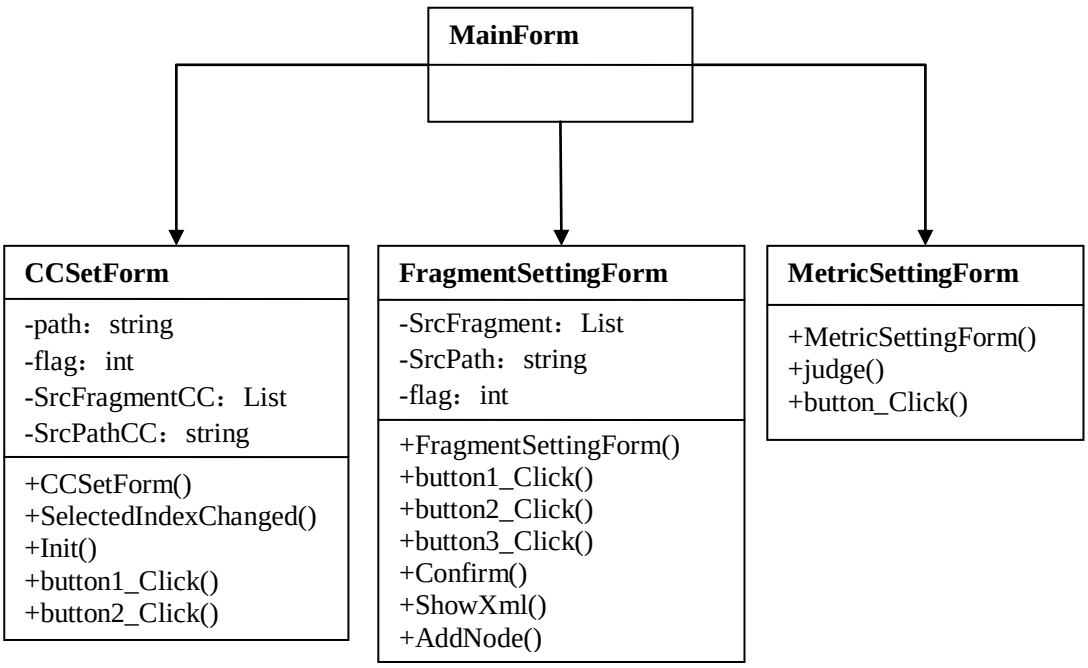


图 3-7 代码检索模块类图

MainForm 类负责响应用户在界面中的请求，控制系统执行相关操作，在该模块中发挥作用的主要函数如下：

（1）setCloneClass_Click()函数：响应用户在界面中单击克隆检索按钮中设置感兴趣的克隆群的请求。

（2）loadAFile_Click()函数：响应用户在界面中单击克隆检索按钮中设置感兴趣的克隆代码的请求。

（3）MetricSetting_ Click()函数：响应用户在界面中单击克隆检索按钮中设置感兴趣代码片段的度量值的请求。

CCSetFrom 类负责设置感兴趣的克隆群，提取对应的代码片段，并利用 FragmentSettingFrom 类完成克隆检索的所有操作，主要变量和函数介绍如下：

（1）path 为指定克隆群所在文件路径，flag 为执行哪种方式克隆检索的标识，SrcFramentCC 为指定克隆代码片段源文件，SrcPathCC 为指定克隆代码片段源文件路径。

（2）CCSetFrom()函数：初始化各个组件。

（3）SelectedIndexChanged()函数：在下拉菜单中选择感兴趣的克隆群。

（4）Init()函数：初始化选择菜单。

（5）button1_Click()函数：调用 FragmentSettingFrom 类完成克隆检索的操作。

（6）button2_Click()函数：取消相关设置的操作。

FragmentSettingFrom 类负责载入外界的克隆代码片段，并完成克隆检索的所有操作，主要变量和函数介绍如下：

（1）SrcFrament 为载入的克隆代码片段源文件，SrcPath 为载入的克隆代码片段路径，flag 为执行哪种方式克隆检索的标识。

（2）FragmentSettingFrom ()函数：初始化各个组件。

（3）button1_Click()函数：从外界载入指定的克隆代码片段的操作。

（4）button2_Click()函数：执行克隆检索的操作。

（5）button3_Click()函数：取消相关设置的操作。

（6）Confirm()函数：克隆检索的相关算法的执行与结果的显示。

（7）ShowXml()函数：显示检索结果树形结构图。

（8）AddNode()函数：检测结果中树形图节点的添加函数。

MetricSettingFrom 类负责，主要变量和函数介绍如下：

（1）MetricSettingFrom()函数：初始化各个组件。

（2）judge()函数：判断代码片段度量是否在设置值的范围之内。

（3）button_Click()函数：根据所设定的度量值范围，执行克隆检索的操作。

3.3.2 代码检索模块的时序图设计

代码检索模块的时序图如图 3-8 所示。代码检索主要支持两种方式，用户可以通过指定特定的代码片段或指定特定的度量值范围来进行检索工作。其中代码片段的指定还包括两种途径，一种是指定当前软件系统中的某一克隆代码片段，一种是指定外界载入的某一代码片段。代码检索模块的工作过程的详细描述如下：

（1）用户按动界面中代码检索选项中“FragmentSettings”菜单下的“Set Clone Class”的按钮，来指定感兴趣的克隆群进行检索，MainForm 类接收请求调用 setCloneClass()函数，弹出设置窗口，用户选择感兴趣的代码片段所在的软件版本号以及克隆群序号并确认，系统执行 CCSetForm 类中的 button1_click()相关的算法进行检索。检索的方式是先从源代码中提取出所选择的代码片段，然后调用 FragmentSettingForm 类中的 button2_click()函数进行代码片段检索。最后调用 ShowXml()函数将检索结果以树形结构的形式返回至用户界面。

（2）用户按动界面中代码检索选项中“FragmentSettings”菜单下的“Load A File”按钮，来从外界指定某一代码片段进行检索，MainForm 类接收请求调用 LoadAFile()函数，弹出设置窗口，用户从外界选择感兴趣的代码片段，设置其路径信息后确认，系统执行 FragmentSettingForm 类中的 button2_click()相关的算法进行检索。最后调用 ShowXml()函数将检索结果以树形结构的形式返回至用户界面。

（3）用户按动界面中代码检索选项中的“MetricSettings”按钮，来指定感兴趣的克隆代码的度量值范围，MainForm 类接收请求调用 MetricSetting()函数，弹出设置窗口，用户设置度量值范围数值后确认，系统执行 MetricSettingForm 类中的 judge()函数，对设定的度量值进行判定，按照度量要求进行检索。最后调用 ShowResult()函数将检索结果以树形结构的形式返回至用户界面。

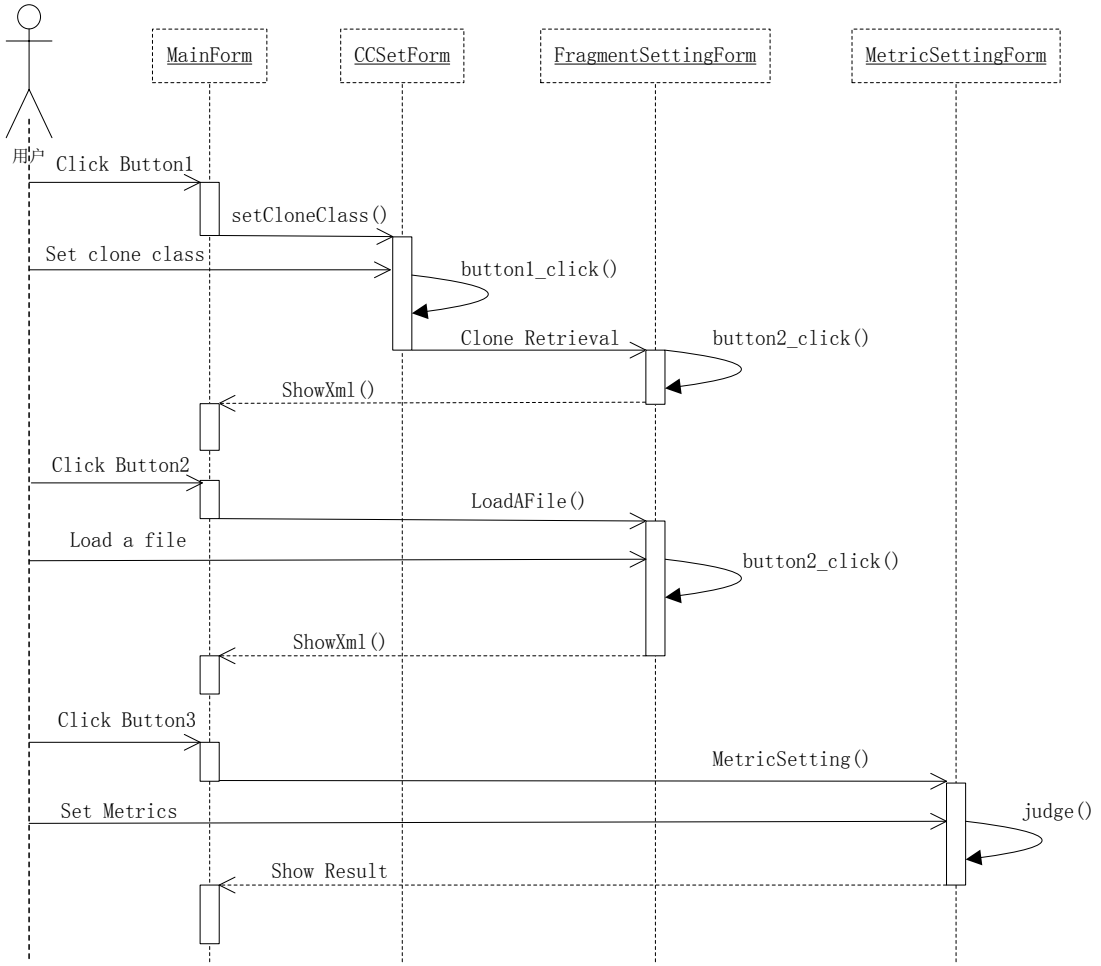


图 3-8　代码检索模块时序图

3.4 克隆进化分析模块的设计

3.4.1 克隆进化分析模块的类图设计

无监督学习模块主要负责。该模块的类图设计如图 3-9 所示，包含了三个主要类，类的详细功能设计与主要变量、函数的描述如下。

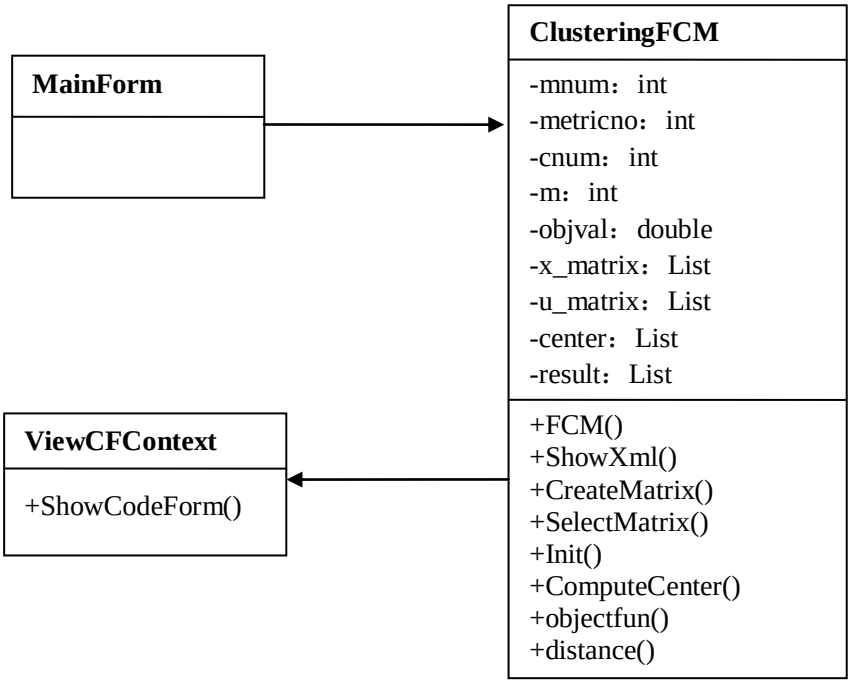


图 3-9 无监督学习模块类图

MainForm 类在该模块中主要发挥作用的是 Cluster_Click()函数，响应用户在界面中单击无监督学习按钮的请求，控制系统执行相关操作。

ClusteringFCM 类负责执行无监督学习的所有操作，主要变量和函数介绍如下：

（1）mnun 为所有提取度量的维数，metricno 为实际选取进行无监督学习的度量维数，cnum 为聚类的类别数，m 为模糊 C 均值聚类的参数，objval 为价值函数的值，x_matrix 存储所有样本度量值的矩阵，u_matrix 为隶属度矩阵，center 为聚类中心矩阵，result 为聚类结果矩阵。

（2）FCM()函数：执行模糊 C 均值聚类算法并显示聚类结果。

（3）ShowXml()函数：：显示聚类结果树形结构图。

（4）CreateMatrix()函数：根据已提取的代码片段度量构建样本矩阵。

（5）SelectMatrix()函数：筛选进行无监督学习所需的度量值。

（6）Init()函数：随机初始化隶属度矩阵。

（7）ComputeCenter()函数：计算聚类中心。

（8）objectfun()函数：计算价值函数。

（9）distance()函数：计算样本间的欧式距离。

ViewCFContext 类中主要用的函数为 ShowCFContext()函数，使得在聚类结果中可以查看相应的源代码内容。

3.4.2 克隆进化分析模块的时序图设计

无监督学习模块的时序图如图 3-10 所示。首先用户按动界面中的无监督学习请求的按钮，MainForm 类接收请求调用 Cluster_Click()函数，执行 ClusteringFCM 类中 FCM()函数进行模糊 C 均值聚类，再调用 ShowXml()函数将无监督学习结果以树形结构形式返回至用户界面。当用户在界面右键鼠标时可以请求查看源代码片段，调用 ViewCFContext 类执行查看源代码相关算法，调用 ShowCodeForm()函数显示查看结果窗口。

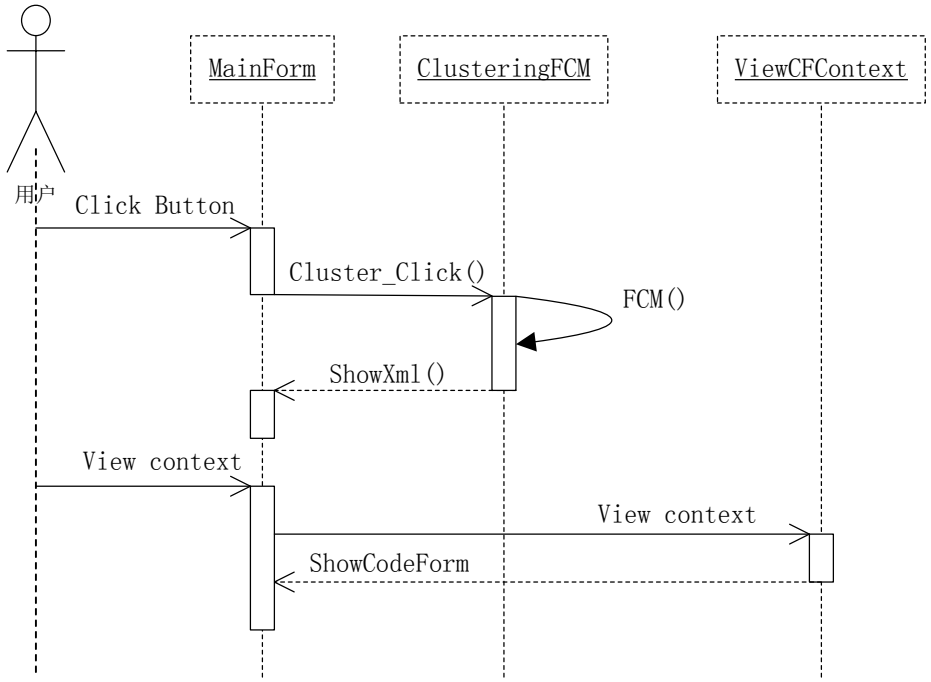


图 3-10　克隆进化分析模块时序图

3.5 本章小结

本章在系统总体设计的基础上，又进一步对各个模块进行了详细设计，其中包括了四个模块中的主要类设计和时序图的设计，详细描述了各个类的作用，类中变量及函数的作用以及类之间的调用关系。

第4章 克隆代码检索与克隆进化分析可视化系统的实现

4.1 可视化模块的实现

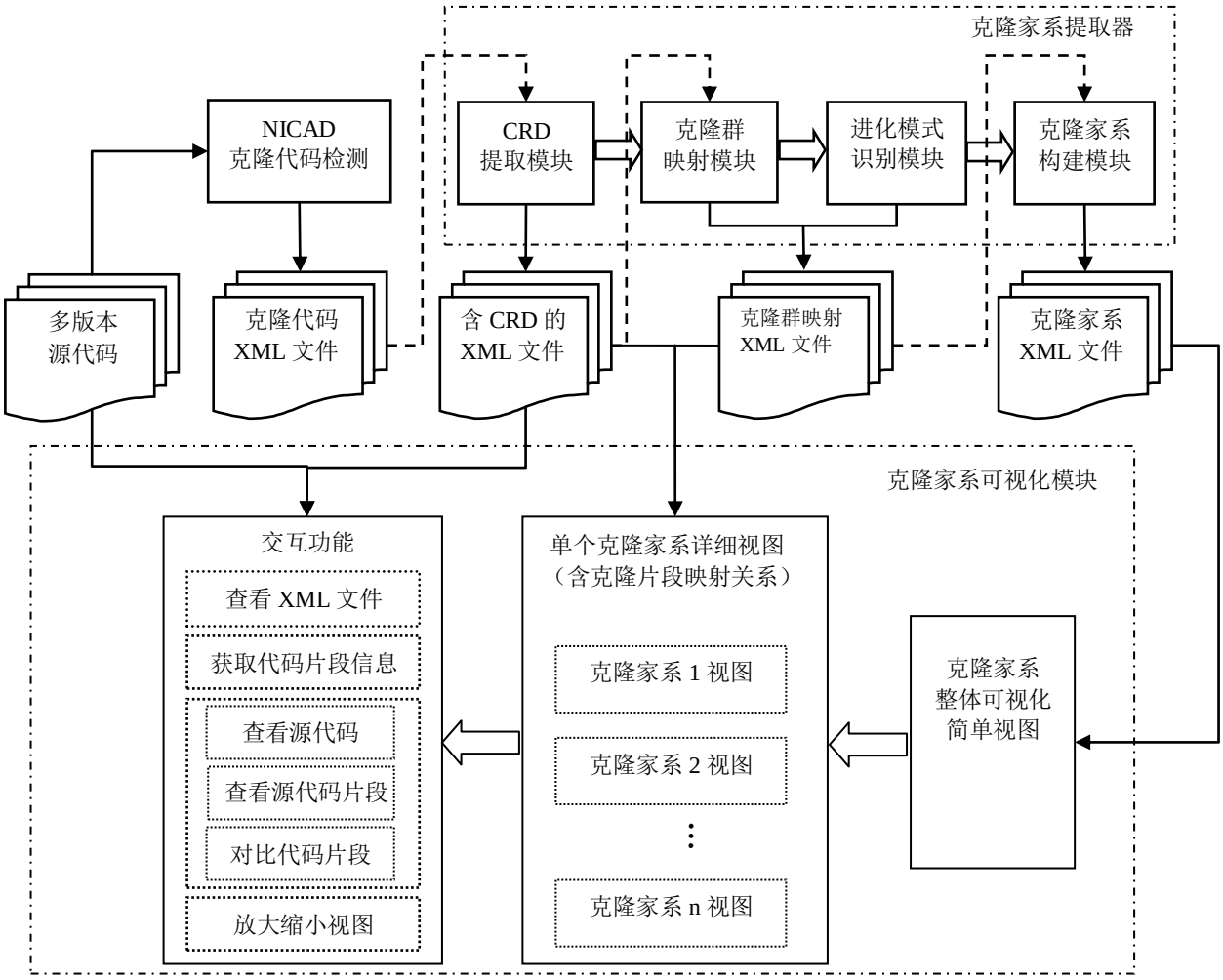


图 4-1 克隆家系可视化原理框图

克隆家系可视化的原理框架如图 4-1 所示。

在 Linux 系统下运行 NICAD 克隆检测工具检测多版本软件中的克隆代码（包括精确克隆及近似克隆），将包含克隆检测结果的克隆代码 XML 文件移至 Windows 系统下，作为克隆家系提取器的输入文件。克隆家系提取器主要包括 CRD 提取模块、克隆群映射模块、克隆群进化模式识别模块以及克隆家系构建模块，提取过程中保存相应的数据信息至相应的 XML 文件中。将克隆家系提取器得到的 CRD 文件、克隆群映射文件、克隆家系文件作为克隆家系可视化模块的输入文件，克

隆家系可视化模块主要实现克隆家系整体可视化视图、含克隆片段映射关系的单个克隆家系详细视图以及添加相应的用户交互功能，展现克隆群的产生、发展直至消失的整个生命过程，帮助用户理解克隆代码在版本更新过程中的演化情况，并且多方面的交互功能使得用户对克隆家系信息的查看更加直观与便捷。

4.1.1 克隆家系整体可视化视图的实现

克隆家系整体可视化展示了各个克隆家系从克隆群产生、发展变化直至消失的整个生命过程，形象直观地展示了克隆家系的整体演化情况。克隆家系整体可视化效果图如图 4-2 所示。

视图中表达的具体信息如下：

（1）行序号：表示克隆家系序号；

（2）列序号：表示克隆群所在的按时间顺序排序的软件版本序号；

（3）方块节点：每个方块节点表示一个克隆群，其中高亮标注的方块表示发生不一致变化的克隆群，未标注的方块则表示发生一致变化的克隆群；

（4）连接线：节点间的连接线表示克隆群的进化关系，其中高亮标注的连接线表示克隆群在进化过程中发生改变，未标注的连接线则表示克隆群未发生任何改变。

图 4-2 克隆家系整体可视化效果图

实现的交互功能主要包括以下 3 点：

（1）克隆群信息的提取：单击方块节点（即表示克隆群节点），在信息显示区域中给出该克隆群相关的克隆家系信息及进化信息，其中包括克隆家系序号、开始版本、结束版本、克隆寿命、进化模式信息等。

（2）双击克隆群节点，跳转至该克隆家系的详细视图。

（3）支持使用鼠标滚轮进行放大、缩小显示的功能。

克隆家系整体可视化的具体算法描述如下：

```
算法：DrawGenGraphics
输入：克隆家系 xml 文件 GenFiles
输出：克隆家系整体可视化视图
Begin
    round = 0；
    将当前描绘的克隆家系文件中的 Evolution 元素添加至 leftEvoList 列表中；
    while (leftEvoList.Count != 0)
        round++；
        指定 leftEvoList[0]为当前节点；
        if (round == 1)    描绘基节点；
        else
            if(当前节点为克隆直线的首节点)
                描绘首节点并与基节点连线；
                leftEvoList.Remove(当前节点)；
                当前节点入栈 s；
                //深度优先遍历绘图
                while (s.count != 0)
                    do{
                        flag = 0；
                        if (栈顶元素的孩子节点不为空)
                            取栈顶元素；
                            提取其孩子节点；
                            if(存在没访问过的孩子节点)        flag = 1；
                            else       flag = 0；
                            if (flag == 1)      // 描绘未被访问过的节点
                                if(没有发生分裂)     在同一层描绘节点；
                                else(发生分裂)  自动下移一层再描绘节点；
                                处理不一致变化的情况；
                                leftEvoList.Remove(当前节点)；
                                当前节点入栈 s；
                            else 更新绘图深度，即一层节点描绘完毕；
                        if (flag == 0)       s.Pop()；
                    } while (flag == 1)
```

```
                    end_while
                end_if
            end_if
        end_while
    End
```

4.1.2 单个克隆家系可视化视图的实现

单个克隆家系展示了克隆家系内部克隆代码片段的映射关系，为进一步对克隆代码片段的研究提供帮助。单个克隆家系中克隆群演化的可视化视图效果如图4-3所示。

视图中表达的具体信息如下：

（1）方块节点：表示代码片段，不同系统版本中相同字母标注的方块节点即表示其存在映射关系。

（2）矩形框：表示克隆群，大小由其所含克隆代码个数决定。其中高亮标注的矩形框表示发生不一致变化的克隆群，未标注的矩形框则表示发生一致变化的克隆群；

（3）连接线：节点间的连接线表示相邻版本间相映射的代码片段变化关系，其中高亮标注的连接线表示代码片段在进化过程中发生改变，未标注的连接线则表示代码片段未发生任何改变。

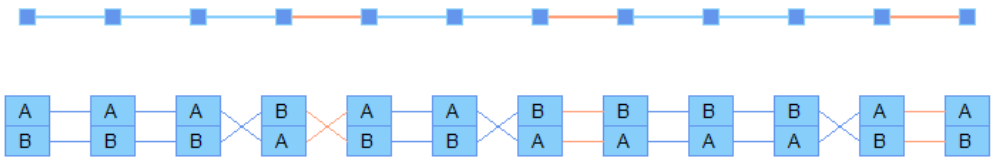


图 4-3 单个克隆家系可视化效果图

交互功能主要包括以下几个方面：

（1）克隆代码片段信息的提取：单击代码片段节点，高亮显示该节点并在信息显示区域中给出该代码片段所在克隆家系信息、进化信息及节点相关信息，其中包括克隆家系序号、开始版本、结束版本、克隆寿命、进化模式信息、节点所在版本号、克隆群序号、克隆群所含代码片段个数、克隆群映射序号、克隆片段序号等。

（2）双击代码片段节点，跳转至相应版本的含 CRD 的 XML 文件树形结构显示图，其中所选代码片段高亮显示，点击“+”展开即为代码片段以 CRD 表示的文件-类-方法-块信息，右键即可显示扩展交互功能，包括代码片段的显示、完整代码片段的显示。

（3）鼠标右键显示扩展交互功能，包括代码片段的显示、完整代码片段的显示及当前版本代码片段与前一个版本的代码片段的差异对比结果的显示。

（4）支持使用鼠标滚轮进行放大、缩小显示的功能。

单个克隆家系可视化的具体算法描述如下：

算法：DrawGenealogy

输入：用户选定的克隆家系 xml 文件，含 CRD 的克隆代码 xml 文件，克隆群映射 xml 文件

输出：单个克隆家系可视化视图

```
Begin
    round = 0；
    将当前描绘的克隆家系文件中的 Evolution 元素添加至 leftEvoList 列表中；
    while (leftEvoList.Count != 0)
        round++；
        指定 leftEvoList[0]为当前节点；
        if (round == 1)   描绘基节点；
        else
            if(当前节点为克隆直线的首节点)
                根据克隆群映射关系确定代码片段标签；
                描绘克隆群及代码片段，并描绘进化连接线；
                leftEvoList.Remove(当前节点)；
                当前节点入栈 s；
                //深度优先遍历绘图
                while (s.count != 0)
                    do{
                        flag = 0；
                        if (栈顶元素的孩子节点不为空)
                            取栈顶元素；
                            提取其孩子节点；
                            if(存在没访问过的孩子节点)       flag = 1；
                            else       flag = 0；
                            if (flag == 1)      // 描绘未被访问过的节点
                                根据克隆群映射关系确定代码片段标签；
                                处理不一致变化的情况；
                                描绘克隆群及代码片段，并描绘进化连接线；
```

```
                                leftEvoList.Remove(当前节点);
                                当前节点入栈 s;
                            else 更新绘图深度，即一层节点描绘完毕;
                        if (flag == 0)      s.Pop();
                    } while (flag == 1)
                end_while
            end_if
        end_if
    end_while
End
```

4.1.3 交互功能的实现

结合不同视图的需求实现交互功能，主要包括以下几个方面：

（1）查看 XML 文件；

（2）获取克隆群及代码片段信息；

（3）查看源代码：含 CRD 的 XML 文件树形结构显示图及克隆家系详细视图中，对相应代码片段右键均可查看源代码。

（4）查看源代码片段：含 CRD 的 XML 文件树形结构显示图及克隆家系详细视图中，对相应代码片段右键均可查看源代码片段。

（5）代码片段的对比：在克隆家系详细视图中右键相应代码片段可对比代码片段。这里采用基于最长公共子序列（LCS）的 diff 算法，通过 LCS 算法获取两段代码的公共行后，存储各自的唯一行（即不在公共行中的行）并显示。Diff 算法的伪代码如下：

```
算法：diff 算法
输入：代码片段 A，代码片段 B
输出：公共行序列 LcsItem，冲突行序列 ConfictItem
Begin
    以代码片段的每一行文本信息为单位计算 A 和 B 的最长公共子序列;
    if（行完全相同）
        将该公共行加入 LcsItem 中;
    else（行不完全相同）
        if（行相似度>阈值）
            将其作为公共行加入 LcsItem 中;
        else
```

将其作为冲突行加入*ConfictItem*中；
end_if
end_if
End

（6）可视化视图的放大缩小功能。

4.2 度量值提取模块的实现

克隆代码特征选择与提取是克隆代码检索与评价的基础，该模块的主要目的是利用已有克隆代码相关文件将所需度量值提取出来，并以文本形式存储这些度量信息，方便后续操作过程中随时使用度量值。本文所采用的度量值既包括了传统的静态度量，又增添了基于克隆家系提取器结果文件获取到的进化度量，对克隆进化的分析有更强的实际意义。

4.2.1 静态度量值提取的实现

静态度量是指仅从单一软件系统版本文件分析即可提取到的度量，这里包括了以下几方面度量：

（1）四种常用的 Halstead 度量：操作符种类、操作数种类、操作符总量、操作数总量。由于 Halstead 其他扩展度量都是基于这四种度量计算得到的，为了防止克隆进化分析结果过分依赖于此类度量，本文只选取四种基础度量从词法角度描述克隆代码的复杂情况。它的提取方法是根据源程序生成 token，依次查找符号表、操作符表和操作数表，统计相应 Halstead 度量的计数。

（2）克隆粒度：指克隆代码片段的代码行数，从含 CRD 的克隆代码 xml 文件的标签<endline>与<startline>中即可直接获得。

（3）克隆相似度：指克隆群内部克隆代码之间的相似程度，从含 CRD 的克隆代码 xml 文件的标签<similarity>中即可直接获得。

（4）参数个数：指克隆代码片段中函数所包含的参数个数，反映了函数间耦合度、函数体复杂性等信息。从含 CRD 的克隆代码 xml 文件的标签<nParam>中即可直接获得。

4.2.2 进化度量值提取的实现

进化度量是结合多个版本文件分析提取的度量，这里包括了以下几方面度量：

（1）克隆寿命：即一个克隆家系从出现至消失跨越的版本数，描述了克隆代码的稳定性，寿命很长的克隆一般为一致变化的克隆或局部不可重构的克隆，重构技术不能对其有所改善。而寿命很短的克隆并不稳定，对其重构也是不必要的。代码片段的克隆寿命的提取方法是从当前所在版本文件开始，根据克隆群映射文

件所提供的克隆群映射关系，向之前版本依次寻找相对应的代码片段，如果找到则将克隆寿命计数加 1，否则返回当前克隆寿命统计数值。

（2）变化次数：即为克隆代码片段在软件系统历史版本中经历改变的次数，它可以有效的描述克隆代码片段进化方面的信息。代码片段的变化次数的提取方法是从当前所在版本文件开始，根据克隆群映射文件所提供的克隆群映射关系，向之前版本依次寻找相对应的代码片段，查看其进化模式，如果不为静态模式则将变化次数计数加 1，直至找不到映射代码片段为止，最终返回变化次数统计数值。

（3）进化模式：即为相邻版本中克隆群的进化关系，包括静态、相同、增加、减少、一致变化、不一致变化和分裂模式。根据克隆家系提取器得到的克隆群映射文件中<EvolutionPattern>标签可直接获取进化模式，这里每种进化模式对应两个取值，用数值“1”表示发生该进化模式，用数值“0”表示未发生该进化模式。

4.3 代码检索模块的实现

克隆代码检测工具检测出来的克隆代码结果，一般是按照克隆群的方式进行罗列。大量的克隆代码结果，程序员并不十分容易理解和操作，克隆代码检索的实现使得程序员可以有效的提取所需的代码片段，从而对这些指定的克隆代码做更深一步的研究。

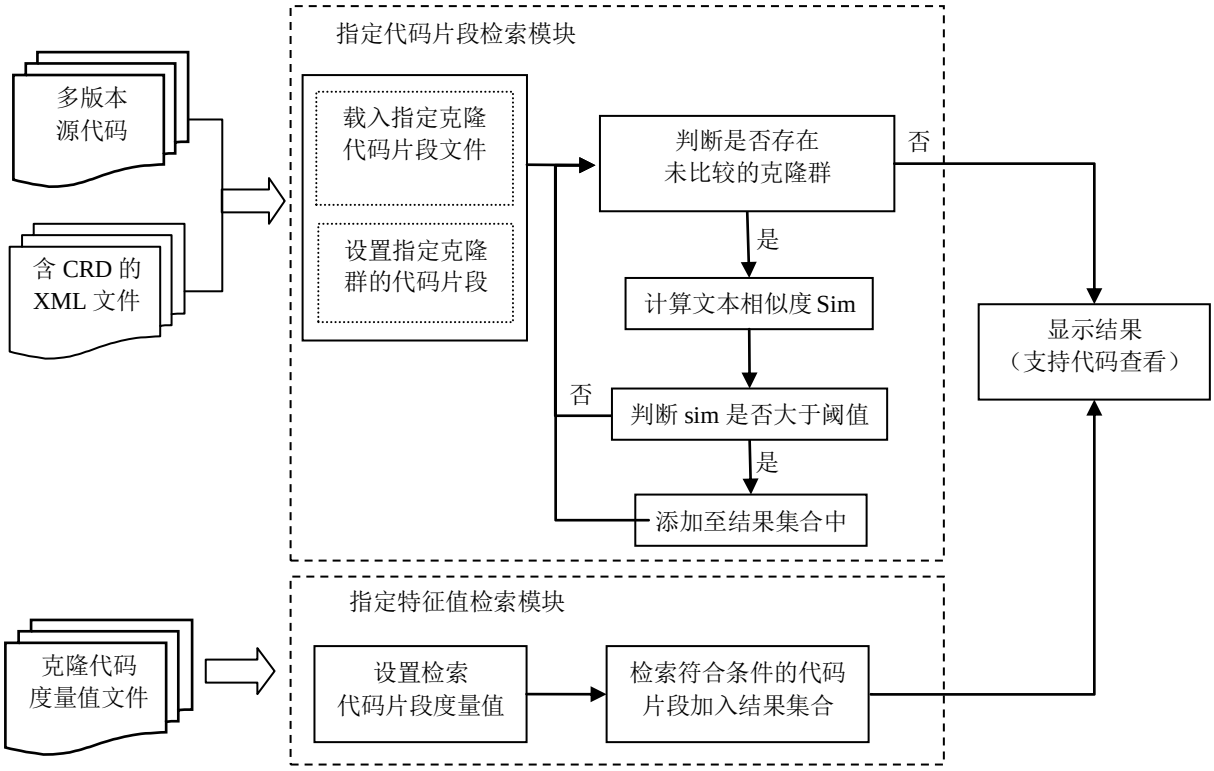


图 4-4 克隆代码检索原理框图

克隆代码检索的原理框图如图 4-4 所示。本文的克隆代码检索功能支持两种方式的检索。第一种方式中，系统通过对用户感兴趣的指定克隆代码片段文件与以克隆群为单位的代码片段进行比较，根据比较其文本相似度的大小检索与指定代码片段互为克隆的代码片段。第二种方式中，系统基于已提取的克隆代码度量值文件，通过设置所需的代码片段的度量值进行检索，按度量而非代码文本对克隆代码进行检索和分析。

4.3.1 指定代码片段检索的实现

指定代码片段检索的主要目的是在已有各个版本的克隆代码中检索与指定代码片段互为克隆的代码片段。首先要设置指定的克隆代码片段，即用户感兴趣的代码片段，本文提供以下两种指定方式：（1）用户可以从外界载入一个新的克隆代码片段的文件；（2）用户可以通过设置软件版本号与该版本中对应的克隆群编号来指定感兴趣的克隆群中的代码片段。然后进行检索工作，在检索过程中以克隆群为单位进行，这是因为克隆群内部的代码片段本身已经互为克隆代码。依次计算每个克隆群中的代表代码片段与指定代码片段的文本相似度值（diff 算法），大于预先设定的阈值则将该克隆群中的所有克隆代码都加入结果集合中。最后显示检索结果。其中显示的信息包括检索到符合条件的克隆群个数，按照版本号和克隆群号依次排列的克隆代码信息，包括代码片段所属文件、起止行号、代码片段序号等，同时支持鼠标右键对检索到的克隆代码片段进行查看功能。对于从外界载入的代码片段还支持其与检索到的代码进行对比查看的功能。

4.3.2 指定度量值检索的实现

指定度量值检索的主要目的是在已有各个版本的克隆代码中检索符合指定度量值的代码片段。首先在度量值设定模块中对所需克隆代码的各项属性进行设定，各项属性均以范围形式给出，对于未输入设定值的项则默认为对其无特定要求。在检索过程中将设定值与已提取的克隆代码特征值文件相匹配，将符合条件的代码片段加入结果集合中，最后显示检索结果。其中显示的信息包括检索到符合条件的克隆代码个数，按照版本号依次排列的克隆代码信息，同时支持鼠标右键对检索到的克隆代码片段进行查看功能。

4.4 克隆进化分析模块的实现

克隆进化分析是利用多版本代码片段及已提取的度量值信息，以代码片段作为样本对其进行无监督学习，分析聚类结果，试图从聚类结果中分析得到对代码评价有指导意义的结论。这里所采用的为模糊 C 均值算法。传统的聚类算法为硬划

分，每个样本对象严格的被划分至某一类，其界限十分明显。然而包含多维度量的代码片段样本不能明确划分为有害或无害，故采用模糊聚类方法。代码进化分析模块的框架如图 4-5 所示。

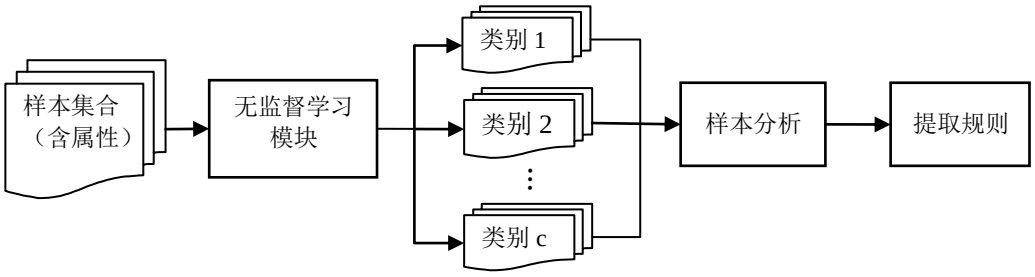


图 4-5 克隆进化分析模块原理框图

采用模糊 C 均值算法对克隆代码进行聚类的流程图如图 4-6 所示，算法中采用的距离均为欧几里得距离。具体实现步骤如下：

（1）首先构建实体-属性矩阵，其中，实体-属性矩阵中行表示样本（即代码片段），列表示所提取的多维度量值。

（2）随机初始化隶属矩阵 U，根据约束条件的要求进行归一化的处理，使得矩阵满足公式(4-1)。

$$\sum_{i=1}^{c} u_{ij} = 1, \forall j = 1,..., n \tag{4-1}$$

（3）应用公式(4-2)和公式(4-3)计算 c 个聚类中心 c_i（i=1，2，…，c）。

$$c_i = \frac{\sum_{j=1}^{n} u_{ij}^{m} x_j}{\sum_{j=1}^{n} u_{ij}^{m}} \tag{4-2}$$

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{d_{ij}}{d_{kj}} \right)^{2/(m-1)}} \tag{4-3}$$

（4）应用公式(4-4)计算目标函数的值。如果它的值小于预先设定的阈值，或者相对上一次迭代中价值函数值的改变量小于预先设定的阈值，则算法停止。

$$J(U, c_1, ..., c_c) = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} \sum_{j}^{n} u_{ij}^{m} d_{ij}^{2} \tag{4-4}$$

（5）计算新的 U 矩阵。返回步骤 3。

（6）由此，得到 c 个聚类中心点向量和 $c \times n$ 的隶属矩阵。根据模糊集合中的最大隶属原则就可以确定每个实体应该归至哪个类。从而得到分类结果。

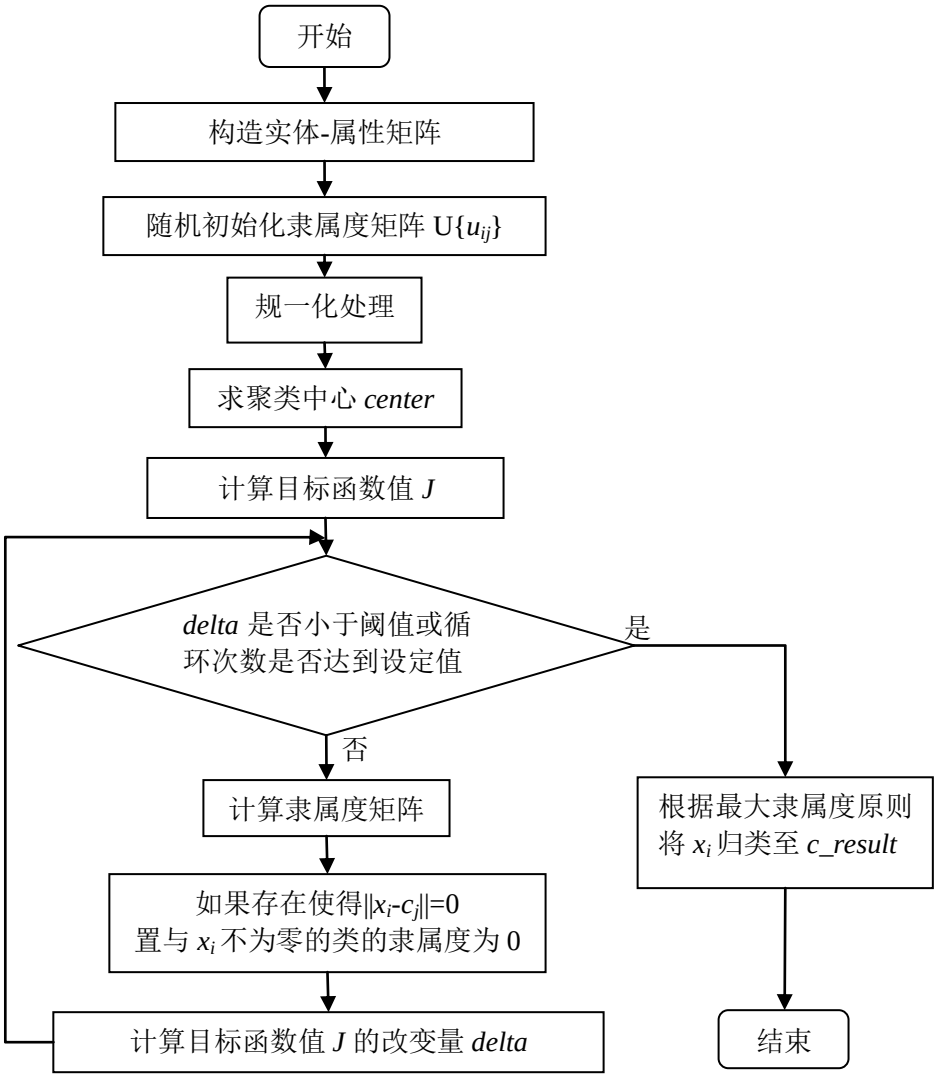


图 4-6 模糊 C 均值算法流程图

应用 UCI 机器学习资源库下载的数据集 iris 对模糊 C 均值算法的聚类效果进行了验证，其中，数据集 iris 中含 3 类数据，每类有 50 个样本，应用模糊 C 均值聚类的结果如表 4-1 所示。

表 4-1 模糊 C 均值算法聚类结果（iris 数据集）

聚类类别	1	2	3	分类错误个数	准确率	总准确率
1	50	0	0	0	100%	
2	0	47	12	12	94%	90%
3	0	3	38	3	76%	

模糊 C 均值算法聚类效果整体来说较好，准确率较高。进一步应用其对代码片段进行聚类，以树形结构按不同类别显示聚类结果，并支持右键对相应代码片段查看的功能。

4.5 本章小结

本章对克隆代码检索与克隆进化分析可视化系统各个模块的具体实现过程进行了详细描述。在可视化模块中介绍了描绘整体克隆家系可视化视图与单个克隆家系可视化视图的算法以及详细的交互功能。度量值提取模块中从静态度量和进化度量两个方面对其实现过程进行了描述。代码检索模块中分别描述了指定代码片段和指定度量值的检索方法的实现。克隆进化分析模块中介绍了利用无监督对代码进行分析的过程以及算法的具体步骤。

第5章 基于克隆进化分析的代码检索与可视化系统的测试

5.1 测试环境

系统测试的软件环境如下：

Window 7 操作系统；

Microsoft Visual Studio 2010 软件开发工具。

系统测试的硬件环境如下：

CPU 和主频：Inter(R) Pentium(R) Dua l-Core CPU 3.20GHz；

内存：2G。

5.2 测试方案

选取六组开源软件系统的源代码作为测试系统，六组系统分别由 Java 语言、C 语言、C#语言编写，代码规模、代码版本数及应用领域都各不相同。测试系统的具体信息如表 5-1 所示。

表 5-1 测试系统信息

系统名称	编程语言	版本个数	代码行数	克隆代码个数
DNSJava	Java	38	9905-23963	611
jEdit	Java	22	47474-110384	6636
wget	C	9	15328-75979	378
conky	C	22	10447-21430	928
ProcessHacker	C#	36	10308-85007	2244
iTextSharp	C#	18	169091-199002	15397

对于可视化模块、度量值提取模块与代码检索模块测试的目的在于是否达到其功能与非功能性需求，而对于克隆进化分析模块除了对其无监督学习的过程进行测试外，还需要进一步对学习结果进行分析，详细内容见 5.4。

5.3 系统模块的测试

如图 5-1 所示，系统主体界面中包括了菜单栏、数据文件信息栏、结果显示窗口以及克隆信息显示栏。菜单栏中包括了克隆家系提取器的相关按钮和本文各个功能按钮。在进行系统测试之前，用户需要在菜单栏的“File”选项中指定测试系统源文件，并加载其对应的克隆代码 xml 文件。含 CRD 的克隆代码 xml 文件、克隆

群映射文件及克隆家系文件可以由系统生成，也可以由用户从外界载入。设置成功后，这些文件都会直接显示在数据文件信息栏中。

5.3.1 可视化模块的测试

用户选择“VisualizeAll”按钮，则显示当前所设置的软件的克隆家系整体可视化视图，如图 5-1 所示为测试系统 wget 从版本 1.06.0 至版本 1.13.0 的克隆家系整体可视化视图。单击视图中的方块节点（即克隆群节点），则在下方克隆信息显示栏中给出该克隆群的克隆家系信息和进化信息。

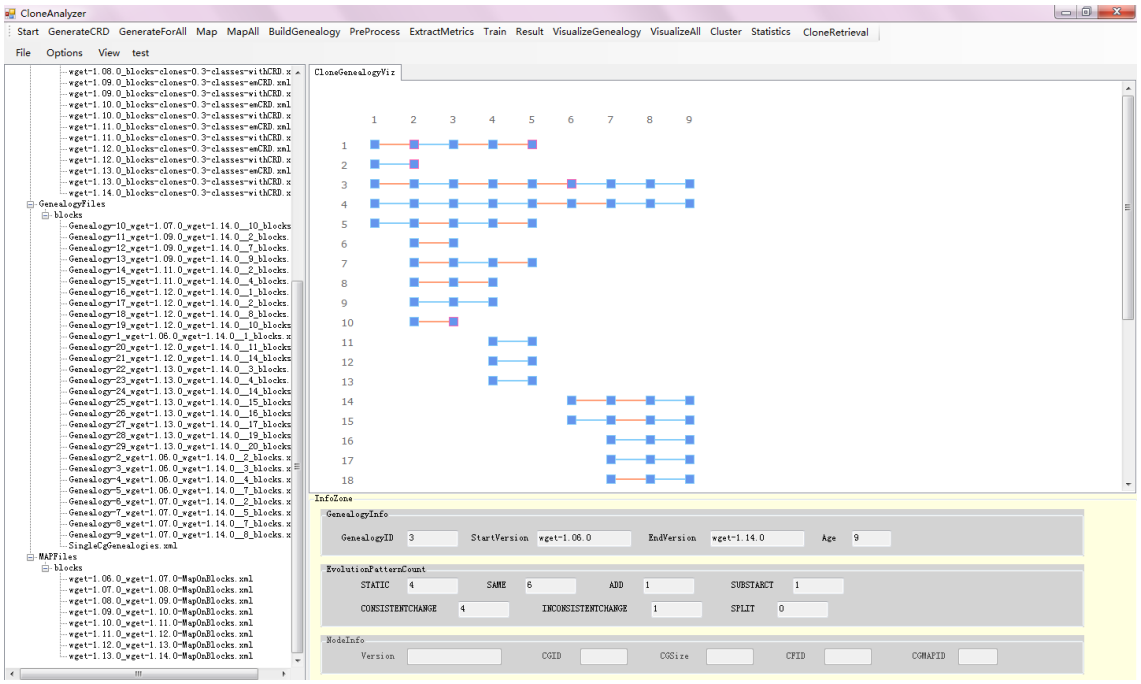


图 5-1 克隆家系整体可视化视图

双击克隆家系整体可视化视图中的克隆群节点或选中某一克隆家系文件单击“VisualizeGenealogy”按钮都可以查看单个克隆家系详细视图，如图 5-2 所示为图 5-1 中序号为 3 的克隆家系详细可视化视图。在该视图中单击代码片段节点，即可高亮显示选中节点并在下方克隆信息显示栏中给出该代码片段的相关信息。

双击克隆家系详细可视化视图中的代码片段节点，则显示相应的含 CRD 的克隆代码 xml 文件树形结构并高亮显示所选代码片段信息，如图 5-3 所示。在含 CRD 的克隆代码 xml 文件树形结构显示中，右键<source>节点支持用户查看源代码和源代码片段的功能。而在克隆家系详细可视化视图中，右键代码片段节点显示菜单栏（如图 5-4 所示），支持查看源代码、源代码片段（如图 5-5 所示）及当前版本代码片段与前一个版本的代码片段的差异对比功能（如图 5-6 所示）。

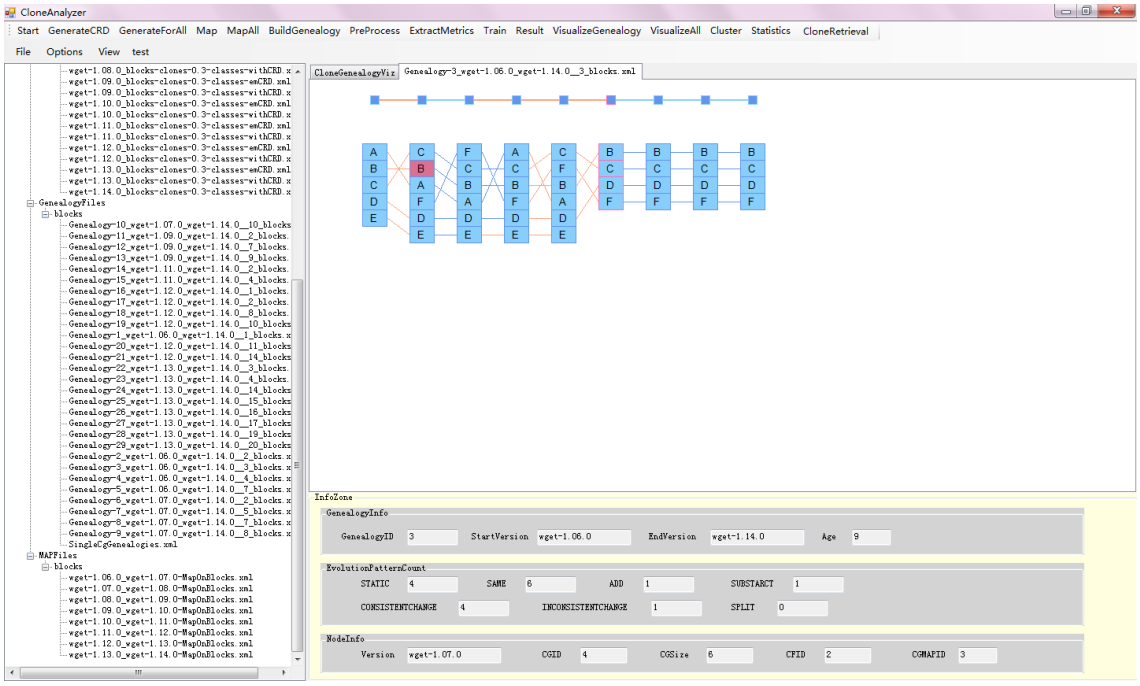


图 5-2 单个克隆家系可视化视图

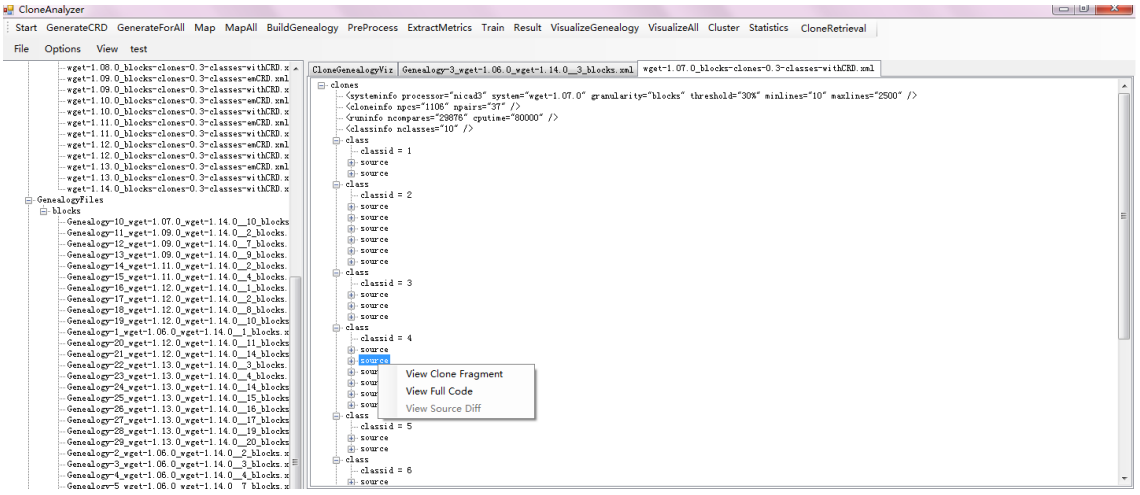


图 5-3 含 CRD 的 XML 文件树形结构显示图

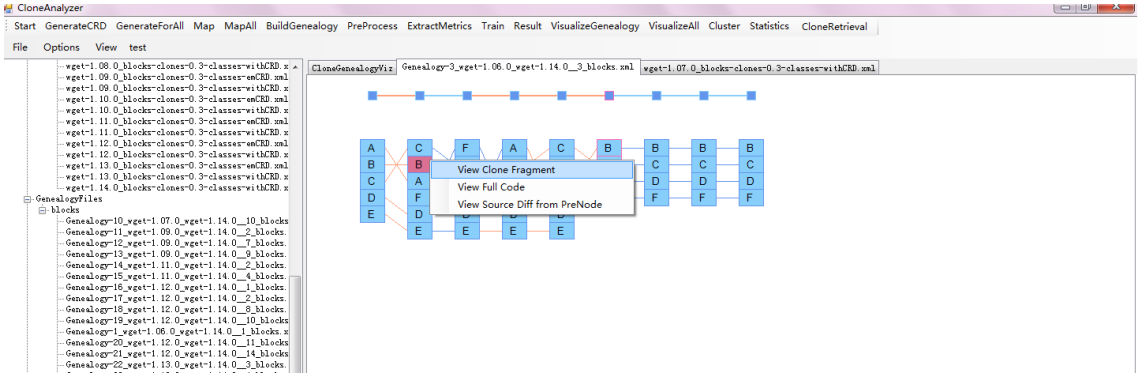


图 5-4 克隆家系详细视图右键支持代码查看功能示意图

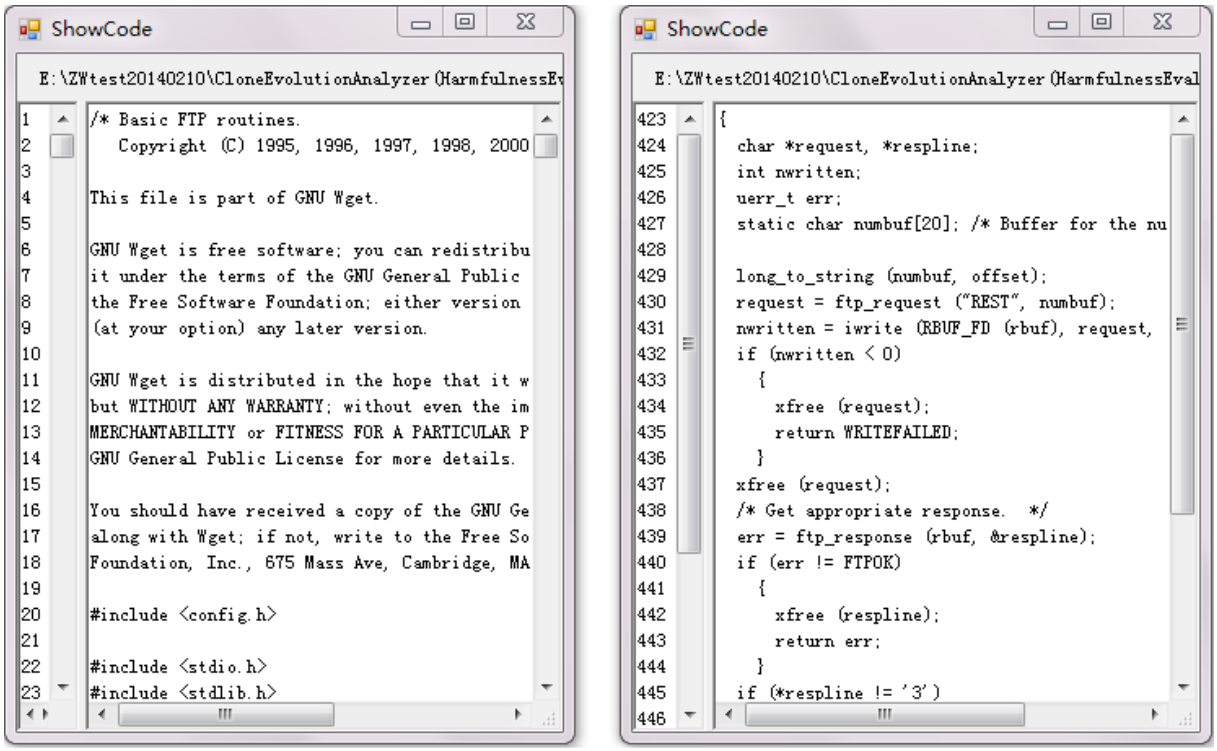


图 5-5 查看源代码及代码片段示意图

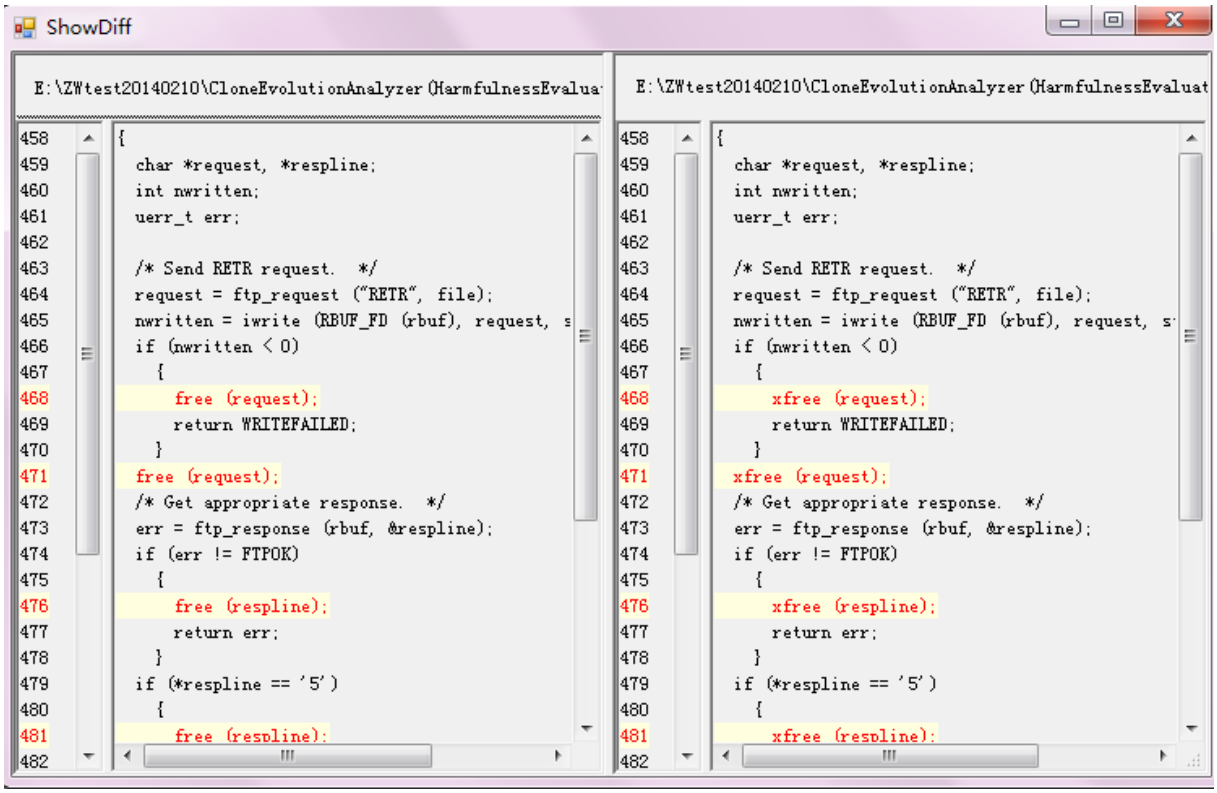


图 5-6 代码片段对比示意图

5.3.2 度量值提取模块的测试

用户选择“ExtractMetrics”按钮，则系统对当前选择目录下的克隆代码文件进行度量值的提取算法，所生成的度量值文件内容如图 5-7 所示，文件中依次存储了代码片段样本的路径、克隆群序号、代码片段序号及所有的度量值信息。

```
-1 E:\wget-results\emCRDFiles\blocks\wget-1.06.0_blocks-clones-0.3-
classes-withCRD.xml 1 1 1:49 2:1 3:0 4:1 5:0 6:0 7:226 8:75 9:13 10:36
11:1 12:0 13:0.71 14:4 15:301 16:49 17:1690.02766308 18:0.0738461538462
19:13.5416666667 20:124.802042812 21:22885.7912709 22:1271.43284838 23:0
24:0 25:1 26:0 27:1 28:0 29:0
-1 E:\wget-results\emCRDFiles\blocks\wget-1.06.0_blocks-clones-0.3-
classes-withCRD.xml 1 2 1:45 2:1 3:0 4:1 5:0 6:0 7:134 8:65 9:13 10:33
11:1 12:0 13:0.71 14:4 15:199 16:46 17:1099.18882926 18:0.0781065088757
19:12.803030303 20:85.8538020483 21:14072.9478897 22:781.830438317 23:0
24:0 25:1 26:0 27:1 28:0 29:0
-1 E:\wget-results\emCRDFiles\blocks\wget-1.06.0_blocks-clones-0.3-
classes-withCRD.xml 1 3 1:46 2:1 3:0 4:1 5:0 6:0 7:246 8:71 9:15 10:29
11:1 12:0 13:0.71 14:4 15:317 16:44 17:1730.63982311 18:0.0544600938967
19:18.3620689655 20:94.2508072679 21:31778.1277864 22:1765.45154369 23:0
24:0 25:1 26:0 27:1 28:0 29:0
-1 E:\wget-results\emCRDFiles\blocks\wget-1.06.0_blocks-clones-0.3-
classes-withCRD.xml 1 4 1:39 2:1 3:0 4:1 5:0 6:0 7:134 8:61 9:15 10:30
11:1 12:0 13:0.71 14:4 15:195 16:45 17:1070.91135378 18:0.0655737704918
19:15.25 20:70.2236953301 21:16331.3981452 22:907.299896956 23:0 24:0
25:1 26:0 27:1 28:0 29:0
```

图 5-7 度量值文件示意图

5.3.3 代码检索模块的测试

5.3.3.1 指定代码片段检索的测试

用户选择“CloneRetrieval”菜单下的二级菜单“FragmentSettings”中的“Load A File”按钮，系统则弹出加载外界文件的窗口，如图 5-8 所示，用户选择感兴趣的代码片段并按下“Confirm”按钮，系统执行检索算法并返回检索结果，如图 5-9 所示，并支持右键查看源代码、源代码片段及代码片段对比功能。

用户选择“CloneRetrieval”菜单下的二级菜单“FragmentSettings”中的“Set Clone Class”按钮，系统则弹出设置克隆群窗口，如图 5-10 所示，用户选择感兴趣的克隆群并按下“Confirm”按钮，系统执行检索算法并返回检索结果，如图 5-11 所示，并支持右键查看源代码、源代码片段及代码片段对比功能。

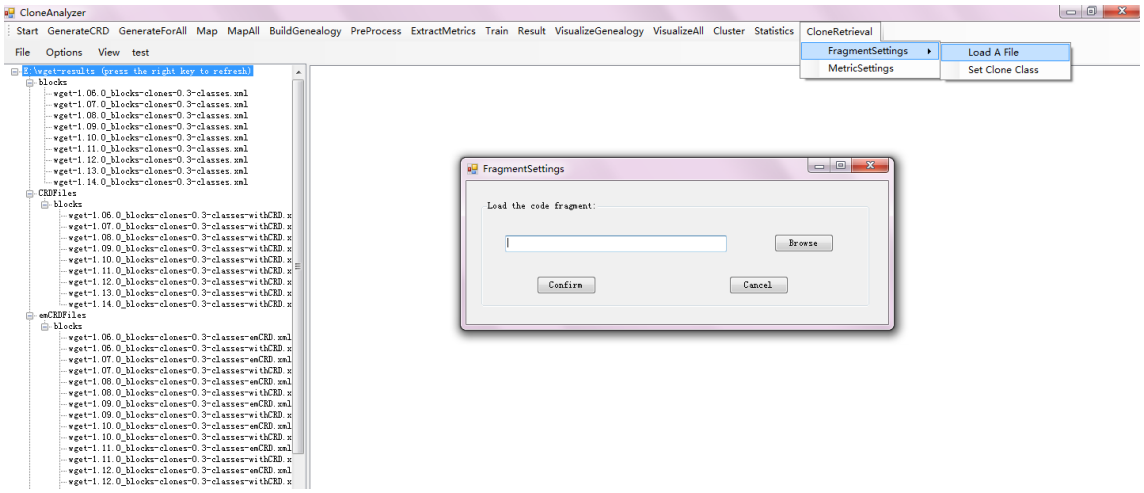


图 5-8 载入指定代码片段示意图

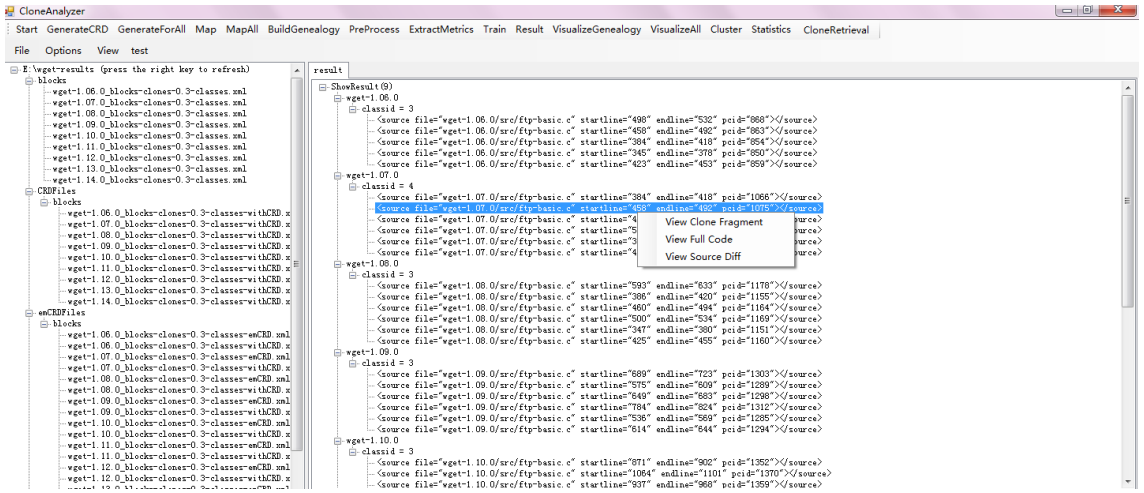


图 5-9 载入指定代码片段检索结果示意图

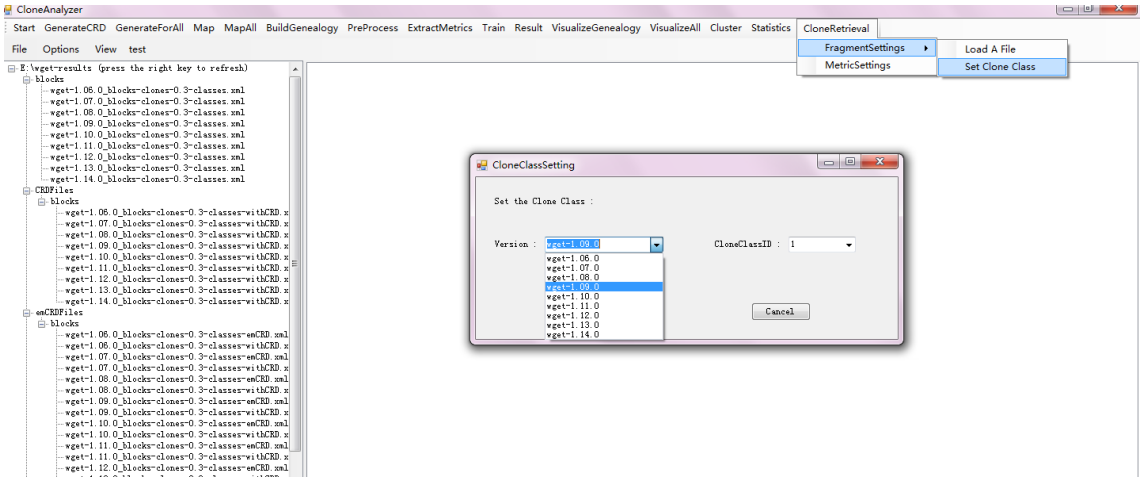


图 5-10 指定克隆群示意图

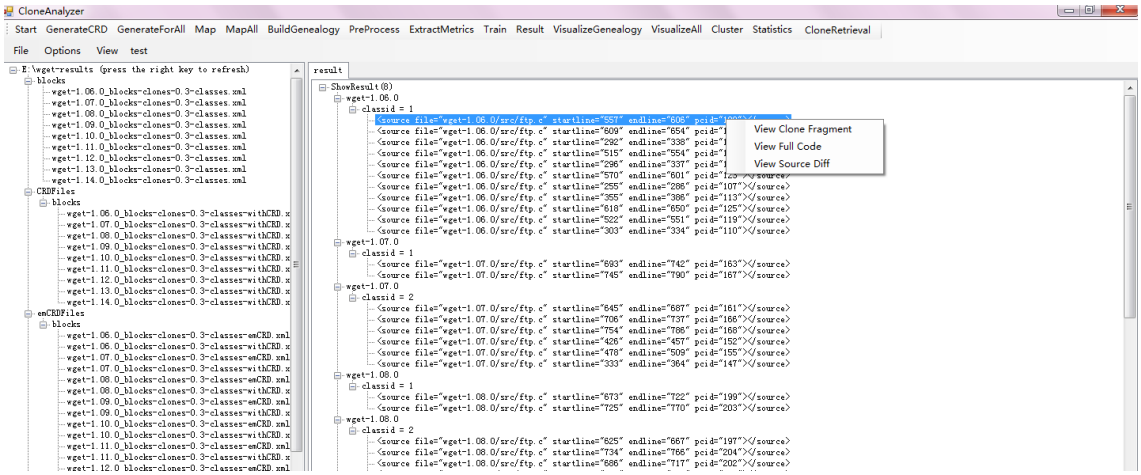


图 5-11 指定克隆群检索结果示意图

5.3.3.2 指定度量值检索的测试

用户选择“CloneRetrieval”菜单下的二级菜单“MetricSetting”按钮，系统则弹出设置度量值范围的窗口，如图 5-12 所示，用户设置感兴趣的代码片段度量值所在范围后按下“确定”按钮，系统执行检索算法并返回检索结果，如图 5-13 所示，并支持右键查看源代码、源代码片段功能。

图 5-12 指定度量值检索设置示意图

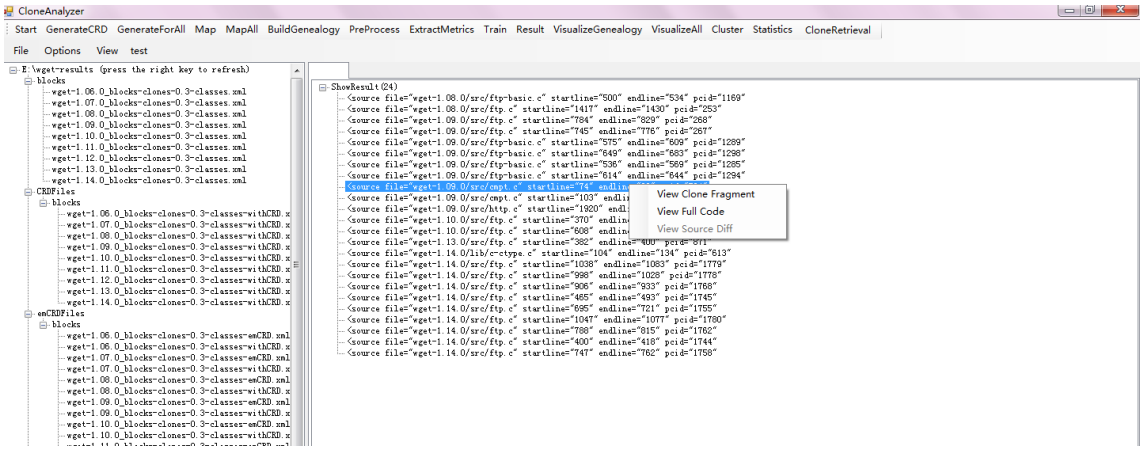


图 5-13 指定度量值检索结果示意图

5.3.4 克隆进化分析模块的测试

用户选择“Cluster”按钮，系统根据提取的度量值文件执行聚类算法，返回聚类结果如图 5-14 所示，并支持右键查看源代码、源代码片段功能。

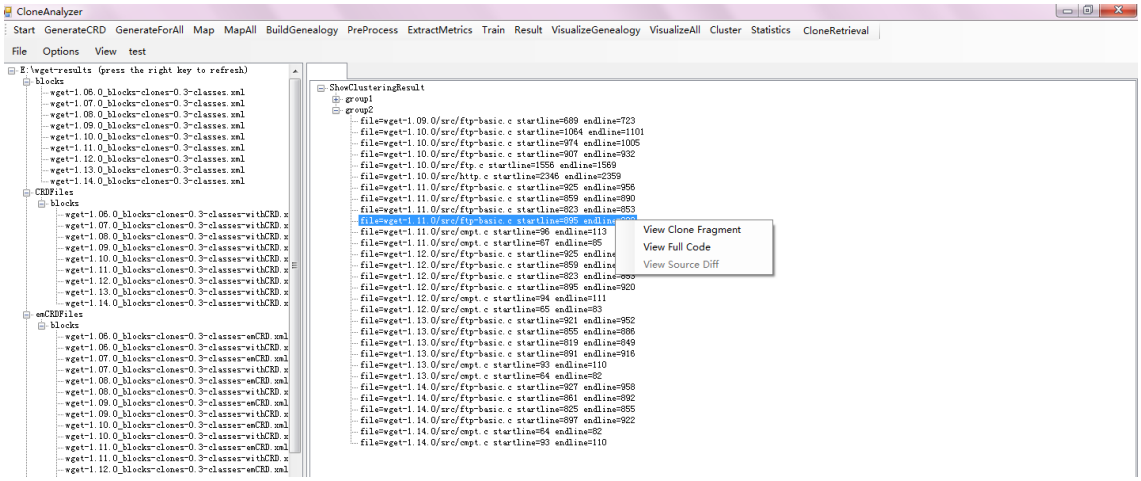


图 5-14 聚类结果示意图

5.4 克隆进化分析的实验

对于克隆进化分析模块无监督学习得到的聚类结果，进一步实验并分析，观察聚为同一类的代码片段样本所共有的特点，试图得出对克隆代码分析有意义的结论，为代码评价与管理提供指导。克隆进化分析的实验主要从三个方面展开，选取不同的度量值对代码片段样本进行聚类，分别探究了代码片段在进化、容量、相似度方面所表现出来的特征。

5.4.1 进化分析实验

5.4.1.1 进化分析实验一：探究克隆寿命与进化模式的关系

表 5-2 Java 语言测试系统进化分析实验一聚类结果

			DNSJava		jEdit	
个数统计		克隆寿命	age<=11	age>11	age<=9	age>9
			485	126	5812	824
		不一致变化	32	0	248	6
		增加模式	66	0	1067	19
		减少模式	9	0	97	6
		分裂模式	-	-	40	3
比率统计	P1	不一致变化	6.60%	0	4.27%	0.73%
		增加模式	13.61%	0	18.36%	2.31%
		减少模式	1.86%	0	1.67%	0.73%
		分裂模式	-	-	0.69%	0.36%
	P2	不一致变化	100%	0	97.64%	2.36%
		增加模式	100%	0	98.25%	1.75%
		减少模式	100%	0	94.17%	5.83%
		分裂模式	-	-	93.02%	6.98%

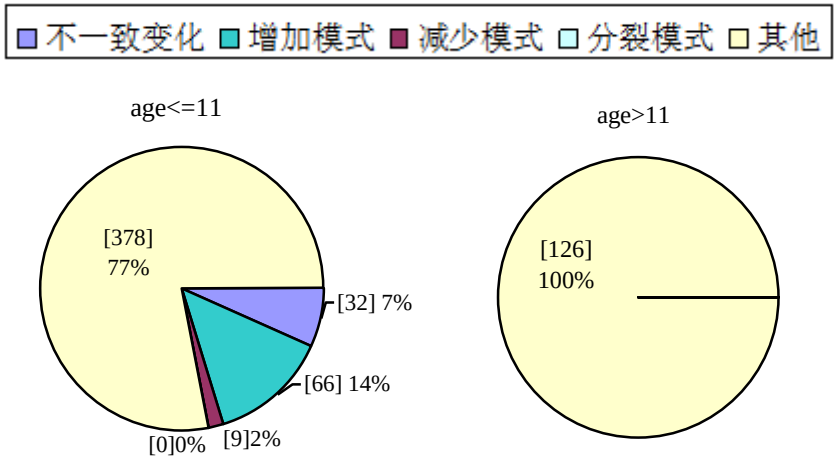
表 5-3 C 语言测试系统进化分析实验一聚类结果

			wget		conky	
个数统计		克隆寿命	age<5	age>=5	age<=9	age>9
			350	28	816	112
		不一致变化	46	4	109	3
		增加模式	222	0	331	1
		减少模式	16	4	73	0
		分裂模式	6	0	12	2
比率统计	P1	不一致变化	13.14%	14.29%	13.36%	2.68%
		增加模式	63.43%	0	40.56%	0.89%
		减少模式	4.57%	14.29%	8.95%	0
		分裂模式	1.71%	0	1.47%	1.79%
	P2	不一致变化	92%	8%	97.32%	2.68%
		增加模式	100%	0	99.70%	0.30%
		减少模式	80%	20%	100%	0
		分裂模式	100%	0	85.71%	14.29%

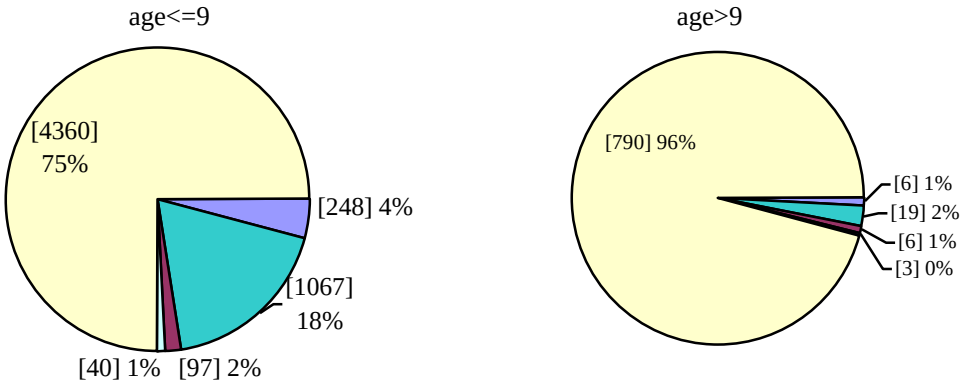
表 5-4 C#语言测试系统进化分析实验一聚类结果

			ProcessHacker		iTextSharp	
		克隆寿命	age<=10	age>10	age<=10	age>10
个数统计			1755	489	8699	6698
		不一致变化	169	1	900	11
		增加模式	447	2	1098	21
		减少模式	153	1	92	11
		分裂模式	49	0	26	3
比率统计	P1	不一致变化	9.63%	0.20%	10.35%	0.16%
		增加模式	25.47%	0.41%	12.62%	0.31%
		减少模式	8.72%	0.20%	1.06%	0.16%
		分裂模式	2.79%	0	0.30%	0.04%
	P2	不一致变化	99.41%	0.59%	98.80%	1.20%
		增加模式	99.55%	0.45%	98.12%	1.88%
		减少模式	99.35%	0.65%	89.32%	10.68%
		分裂模式	100%	0	89.66%	10.34%

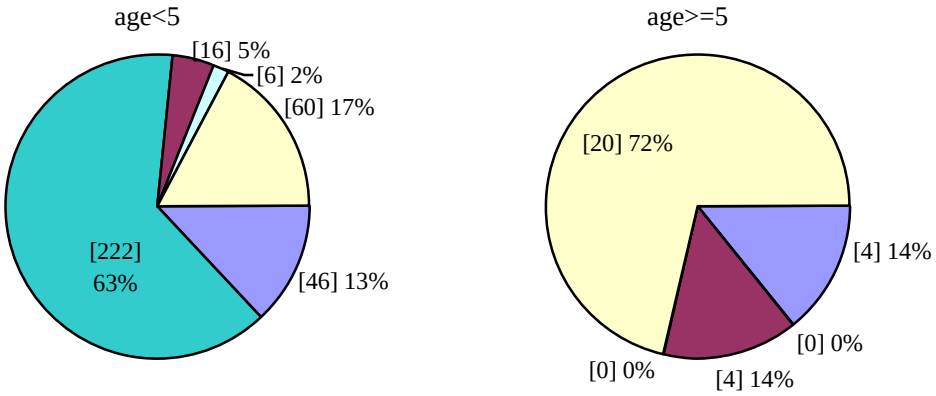
选取 9 维进化度量进行聚类，其中包括了克隆寿命、克隆变化次数及七种进化模式。聚类结果的统计表格如表 5-2、5-3、5-4 所示。以 DNSJava 测试系统为示例，在取聚类个数 c=2 时，分析聚类结果，发现克隆代码片段样本恰好按照克隆寿命 age 是否大于 11 完全分离开来。分别统计每类中发生不一致变化、增加、减少以及分裂模式的次数，并计算比率 P1、P2。在这里，比率 P1 表示的是同一类的代码片段中发生某一进化模式的代码片段个数占该类代码片段总数的百分比；比率 P2 表示的是某一类的代码片段中发生某进化模式的代码片段个数占所有代码片段中发生该进化模式的代码片段总数的百分比。每类中不同特征的代码片段个数与其对应的比例 P1 的饼状图如图 5-15 所示。



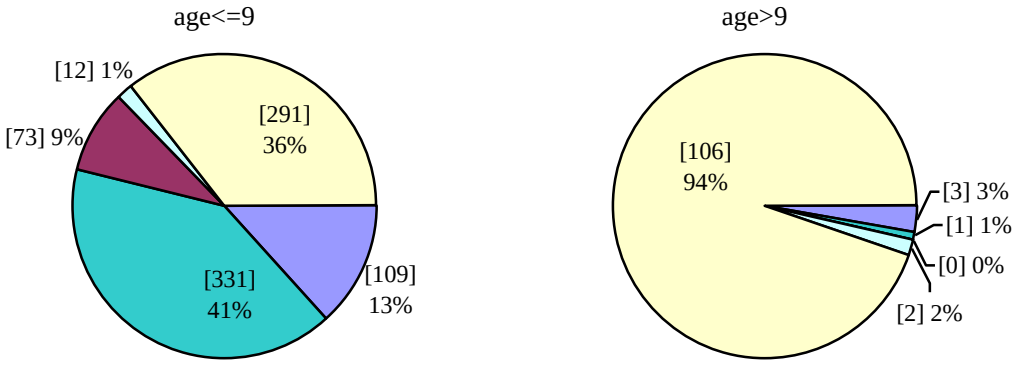
(a)DNSJava 系统进化分析实验一聚类结果饼状图



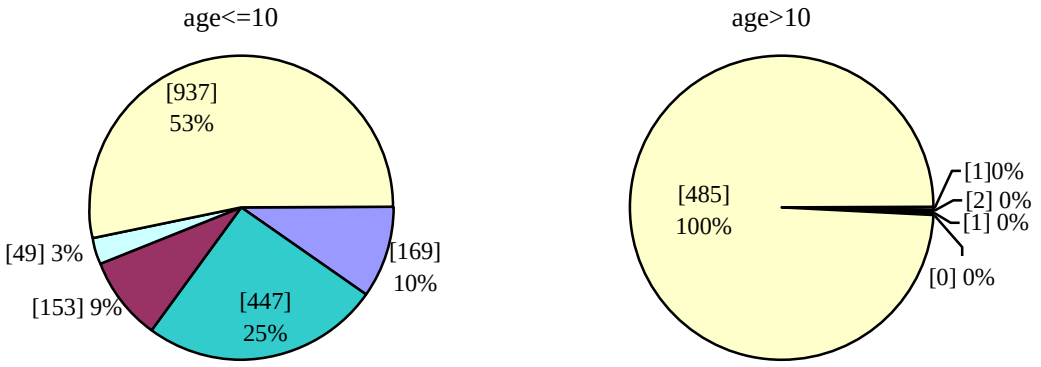
(b)jEdit 系统进化分析实验一聚类结果饼状图



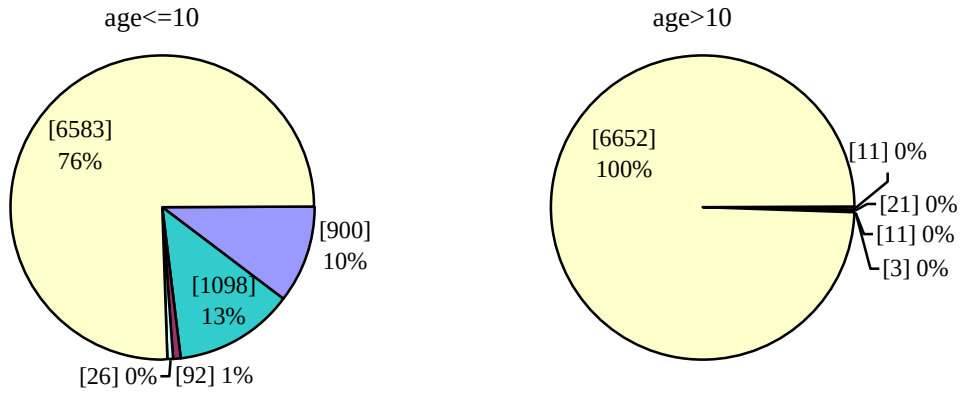
(c)wget 系统进化分析实验一聚类结果饼状图



(d)conky 系统进化分析实验一聚类结果饼状图



(e)ProcessHacker 系统进化分析实验一聚类结果饼状图



(f)iTextSharp 系统进化分析实验一聚类结果饼状图

图 5-15 测试系统进化分析实验一聚类结果饼状图

观察统计表格和饼状图，发现不一致变化、增加模式、减少模式和分裂模式都一般发生在克隆寿命较短的代码片段组中。从绝对数量上来说，以上数据均符合该规律；而从比例上来看，有个别数据的表现并不明显。比如 wget 系统中克隆寿命较短的代码片段中发生减少模式的比例要低于克隆寿命较长的代码片段中的比例，这主要是因为在测试系统的所有代码片段中发生减少模式的总个数比较少，故分布至不同克隆寿命时规律并不明显。

查看源代码分析以上结论的原因，发现不一致变化较常出现在软件系统版本的初期阶段和中期阶段（故克隆寿命较短），初期阶段的克隆代码不够成熟需要作较大的改动，而中期主要表现为克隆代码的调整与功能修改等。而从长期来看，克隆代码片段的进化趋于稳定，较少再发生修改。比如，jEdit 系统在软件系统版本中期阶段的 4.03 版本中删除了 ColorOptionPane.java 和 StyleOptionPane.java 文件，它们分别控制了系统中 Color 和 Style 的相关设置，而在相邻的 4.10 版本中添加 SyntaxHiliteOptionPane.java 文件，对其 Color 和 Style 进行控制，导致产生不一致变化，其克隆群的进化过程如图 5-16 所示。

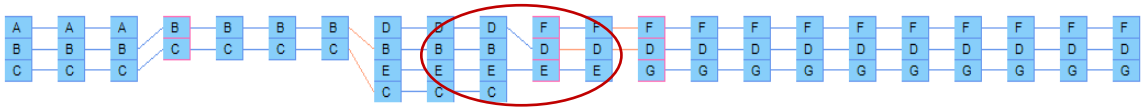


图 5-16 进化分析实验一示例图

实验过程中还发现，聚类个数 c 的选取对实验结果并无明显的影响。同样以 DNSJava 测试系统为示例，在取聚类个数 c=3 时，克隆代码片段样本恰好按照克隆寿命 age<=6，6<=age<=18，age>18 分离开来，在进行分析时可以将前两类看作一组，最后得到的结论也是类似的。

5.4.1.2 **进化分析实验二：**

探究克隆代码片段的变化次数与其所在克隆群不一致变化的关系

除去实验一种克隆寿命度量的影响，选取其余 8 维进化度量进行聚类，其中包括了变化次数及七种进化模式。在这里，变化次数是指代码片段在历史版本中发生改变的次数，而不一致变化是克隆群因不一致的修改而发生的变化（例如克隆群内增加了新的克隆代码片段，或删除/重构了原有的代码片段等），因此会出现变化次数为 0 时，即克隆代码片段本身在进化过程中并未发生改变，因代码片段所在克隆群发生变化而产生不一致的情况。

聚类结果的统计表格如表 5-5、5-6、5-7 所示。分析聚类结果，发现克隆代码片段样本恰好按照变化次数 n 的多少完全分离。分别统计每类中发生不一致变化的次数，并计算比率 P1、P2。每类中发生不一致变化的代码片段个数与其对应的比例 P1 的饼状图如图 5-17 所示。

表 5-5 Java 语言测试系统进化分析实验二聚类结果

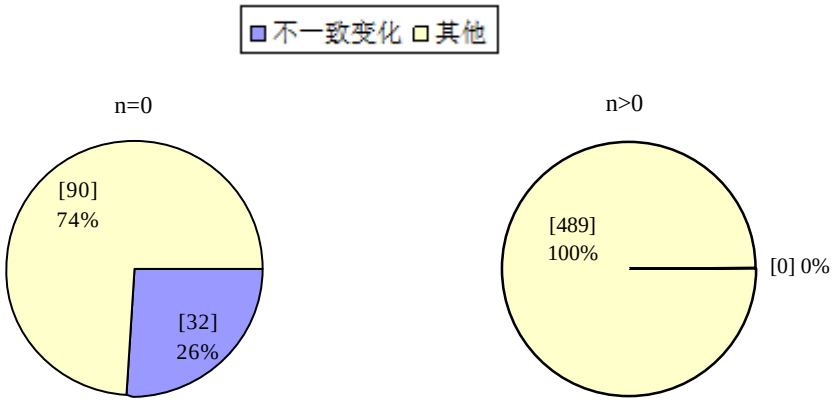
			DNSJava		jEdit	
个数统计		变化次数	n=0	n>0	n=0	n>0
			122	489	6264	372
		不一致变化	32	0	244	10
比率	P1	不一致变化	26.23%	0	3.90%	2.69%
统计	P2	不一致变化	100%	0	96.07%	3.93%

表 5-6 C 语言测试系统进化分析实验二聚类结果

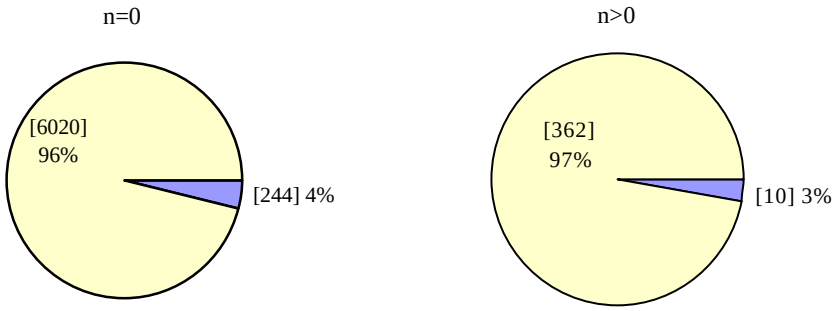
			wget		conky	
个数统计		变化次数	n=0	n>0	n<=1	n>1
			324	54	816	122
		不一致变化	46	4	107	5
比率	P1	不一致变化	14.20%	7.41%	13.11%	4.10%
统计	P2	不一致变化	92%	8%	99.11%	0.89%

表 5-7C#语言测试系统进化分析实验二聚类结果

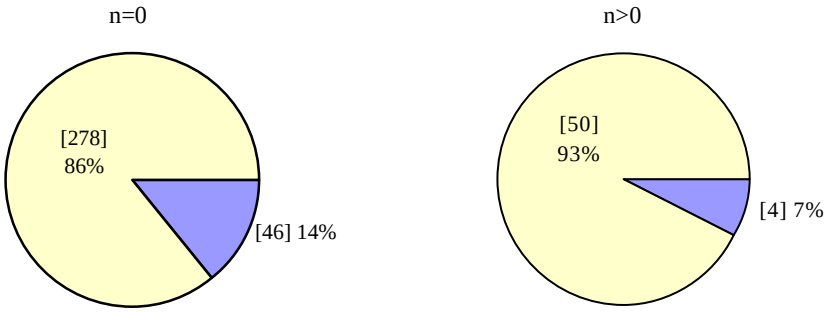
			ProcessHacker		iTextSharp	
个数统计		变化次数	n<=1	n>1	n=0	n>0
			2089	155	1131	14266
		不一致变化	166	4	911	0
比率统计	P1	不一致变化	7.95%	2.58%	80.55%	0
	P2	不一致变化	97.65%	2.35%	100%	0



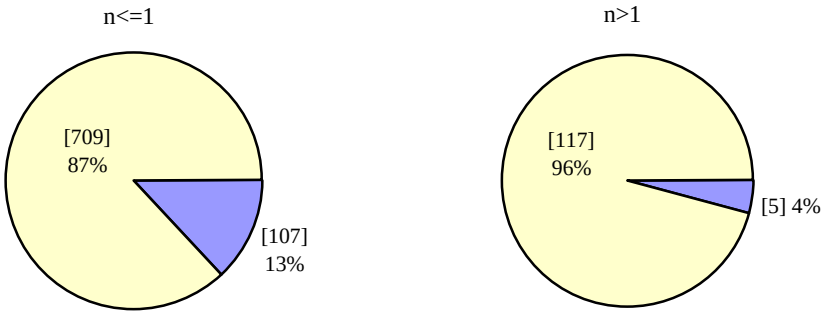
(a)DNSJava 系统进化分析实验二聚类结果饼状图



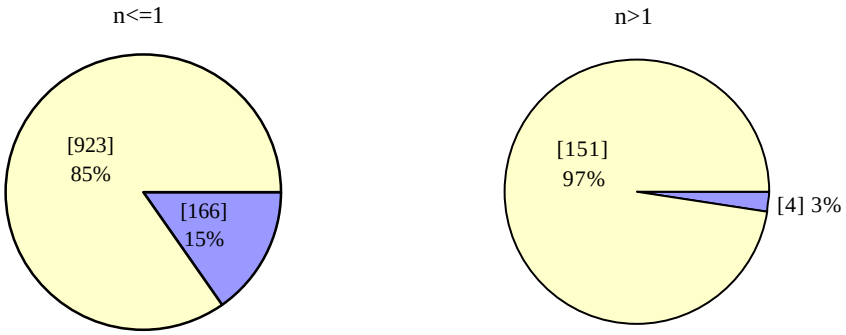
(b)jEdit 系统进化分析实验二聚类结果饼状图



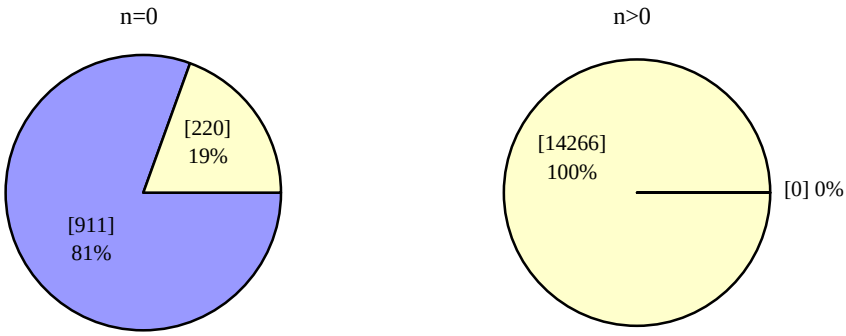
(c)wget 系统进化分析实验二聚类结果饼状图



(d)conky 系统进化分析实验二聚类结果饼状图



(e)ProcessHacker 系统进化分析实验二聚类结果饼状图



(f)iTextSharp 系统进化分析实验二聚类结果饼状图

图 5-17 测试系统进化分析实验二聚类结果饼状图

观察统计表格和饼状图，发现不一致变化一般发生于历史变化次数较少的代码片段组中。查看源代码分析其原因，由于不一致变化常出现在克隆寿命较短的代码片段中，寿命较短导致其本身可能发生的变化次数也较少，多数情况表现为克隆代码片段在进化过程中并未发生更改，但是其所在的克隆群因增加了新的克隆代码片段或删除/重构了原有的克隆代码片段而发生了不一致变化，上述图 5-16 的示例也能说明这类情况。

5.4.1.3 进化分析实验三：探究克隆寿命、克隆变化次数与不一致变化的关系

综合进化实验一和二的结果发现克隆寿命与变化次数都与是否发生不一致变化有一定的关联，为了进一步探究它们之间的关系，利用克隆寿命、变化次数、一致变化和不一致变化这 4 维度量进行聚类，聚类结果的统计表格如表 5-8、5-9、5-10 所示。分析聚类结果，发现克隆代码片段样本恰好按照克隆寿命 age 的长短完全分离开来。分别统计每类中发生不一致变化的次数，并计算比率 P1、P2。为了观察变化次数的规律，将不同类别中的克隆变化次数的统计数字作成折线图，如图 5-18 所示。

表 5-8 Java 语言测试系统进化分析实验三聚类结果

			DNSJava		jEdit	
个数统计		克隆寿命	age<5	age>=5	age<=5	age>5
			272	339	4025	2611
		不一致变化	28	4	229	25
比率统计	P1	不一致变化	10.29%	1.18%	5.69%	0.96%
	P2	不一致变化	87.5%	12.5%	90.16%	9.84%

表 5-9 C 语言测试系统进化分析实验三聚类结果

			wget		conky	
个数统计		克隆寿命	age<5	age>=5	age<=7	age>7
			350	28	688	240
		不一致变化	46	4	107	5
比率统计	P1	不一致变化	13.14%	14.29%	15.55%	2.08%
	P2	不一致变化	92%	8%	95.54%	4.46%

表 5-10 C#语言测试系统进化分析实验三聚类结果

			ProcessHacker		iTextSharp	
个数统计		克隆寿命	age<=4	age>4	age<5	age>=5
			1197	1047	4418	10979
		不一致变化	150	20	885	26
比率统计	P1	不一致变化	12.53%	1.91%	20.03%	0.24%
	P2	不一致变化	88.24%	11.76%	97.15%	2.85%

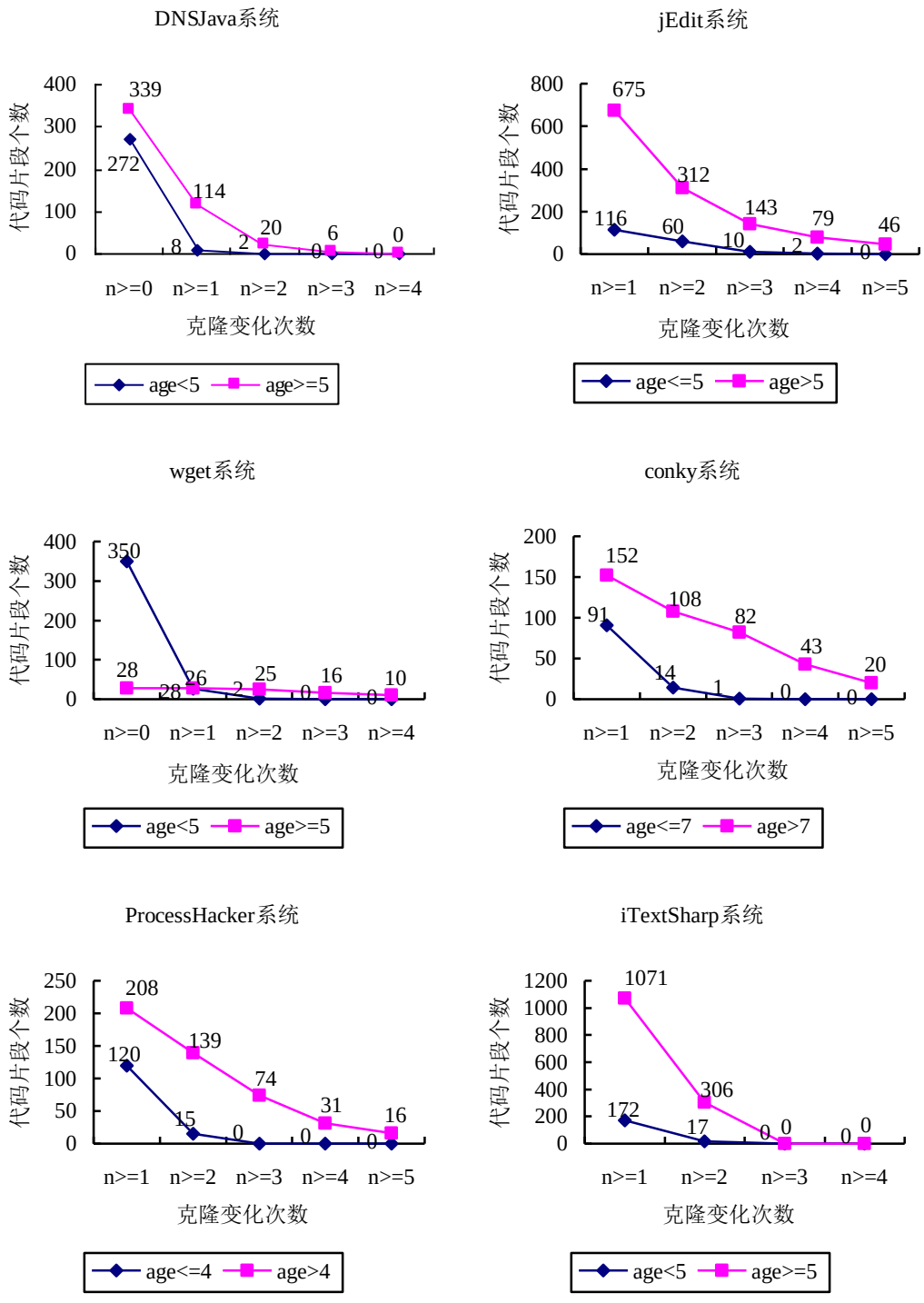


图 5-18 测试系统进化分析实验三折线图

观察统计表格，发现不一致变化一般出现在克隆寿命较短的代码片段中，这与实验一中得到的结论相同。观察折线图中每类在不同变化次数约束条件下的代码片段个数，可以发现变化次数较多的代码片段一般出现在克隆寿命较长的代码片段中，而克隆寿命较短的代码片段一般来说变化次数也都较少。

综合以上三组进化分析实验，得到以下结论：

（1）变化次数较多的代码片段一般出现在克隆寿命较长的代码片段中；

（2）克隆寿命较短的代码片段发生改变的次数较少(或不发生改变) ；

（3）不一致变化一般出现在克隆寿命较短/变化次数较少(或不发生改变)的代码片段中。

5.4.2 容量实验

选取克隆粒度和 9 维进化度量进行聚类，聚类结果的统计表格如表 5-11、5-12、5-13 所示。分析聚类结果，发现克隆代码片段样本恰好按照克隆粒度 loc 的长短完全分离开来。分别统计每类中发生不一致变化的代码片段个数，并计算比率 P1、P2。为了观察克隆寿命较长的代码片段的规律，将不同类别中的克隆寿命统计数字作成折线图，如图 5-19 所示。

表 5-11 Java 语言测试系统容量实验聚类结果

			DNSJava		jEdit	
个数统计		代码规模	9905-23963		47474-110384	
		克隆粒度	loc<16	loc>=16	loc<=30	loc>30
			547	64	5388	1248
		不一致变化	26	6	207	47
比率统计	P1	不一致变化	4.75%	9.38%	3.84%	3.77%
	P2	不一致变化	81.25%	18.75%	81.50%	18.50%

表 5-12 C 语言测试系统容量实验聚类结果

			wget		conky	
个数统计		代码规模	15328-75979		10447-21430	
		克隆粒度	loc<90	loc>=90	loc<=38	loc>38
			362	16	821	107
		不一致变化	50	0	101	14
比率统计	P1	不一致变化	13.81%	0	12.30%	13.08%
	P2	不一致变化	100%	0	87.83%	12.17%

表 5-13 C#语言测试系统容量实验聚类结果

			ProcessHacker		iTextSharp	
个数统计		代码规模	10308-85007		169091-199002	
		克隆粒度	loc<=38	loc>38	loc<=95	loc>95
			2000	244	2000	244
		不一致变化	160	10	160	10
比率统计	P1	不一致变化	8%	4.10%	5.95%	5.08%
	P2	不一致变化	94.12%	5.88%	96.49%	3.51%

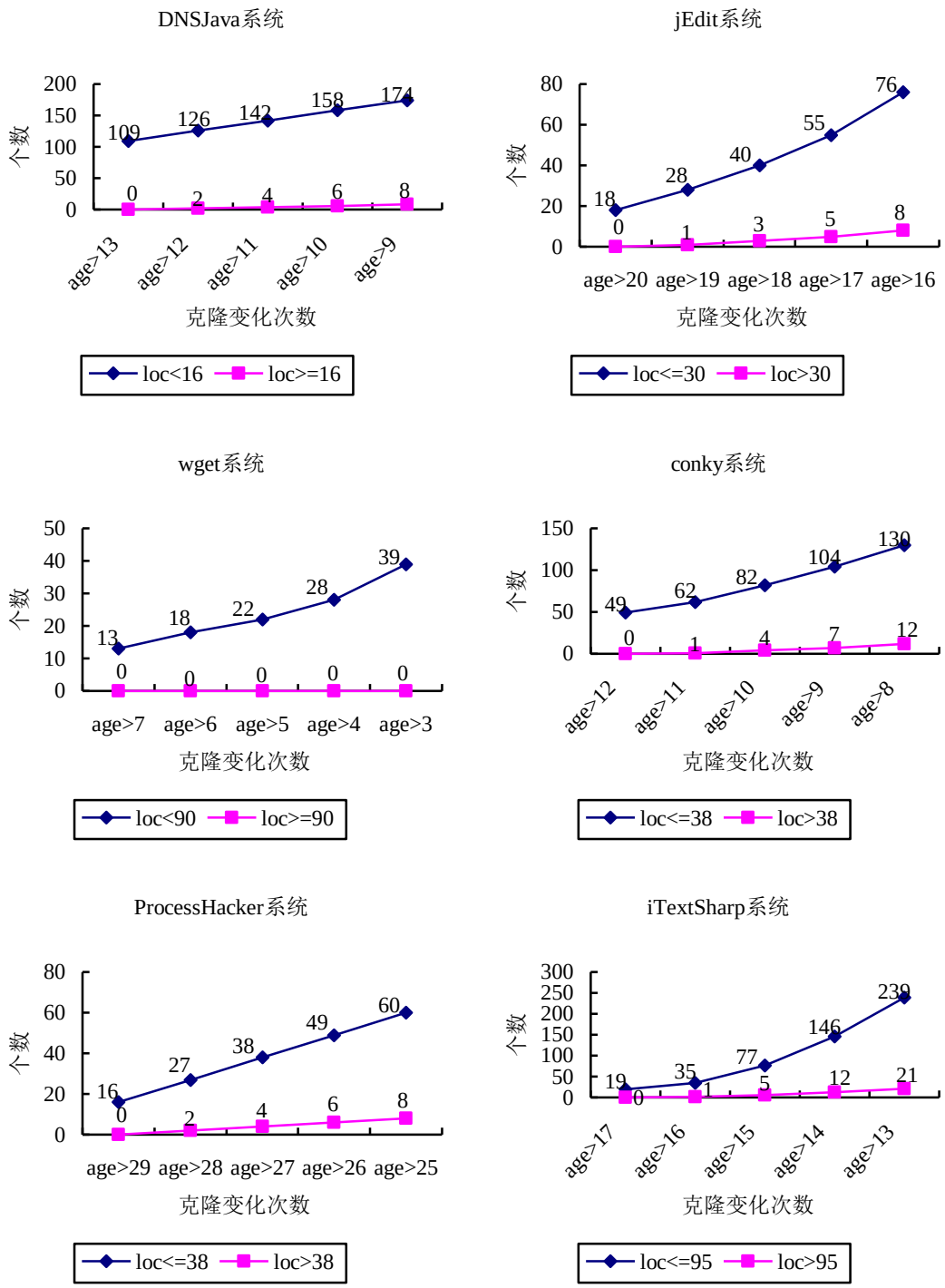


图 5-19 测试系统容量实验折线图

观察统计表格，发现克隆粒度较小的代码片段组中发生不一致变化的代码片段个数与比例一般要高于克隆粒度较大的代码片段。观察折线图中每类在不同克隆寿命约束条件下的代码片段个数，发现克隆寿命特别长的代码片段一般都出现在克隆粒度较小的一类中。总结得到以下两个结论：

（1）不一致变化一般发生在克隆粒度较小的代码片段中（或者说，一般不发生在克隆粒度较大的代码片段中）。

（2）克隆寿命特别长的代码片段一般为克隆粒度较小的代码片段（或者说，一般不为克隆粒度较大的代码片段）。

以上结论中，我们更加强调不一致变化和克隆寿命特别长的代码片段一般不为克隆粒度较大的代码片段，这里克隆寿命特别长是相对于软件总版本数来说的，指那些经历了大部分软件版本的代码片段，与进化分析实验中探究的较长克隆寿命的意义并不相同。另外，尽管不一致变化和克隆寿命特别长的代码片段一般都生于克隆粒度较小的代码片段组中，它们各自所指的并不是相同的代码片段，并不意味它们之间存在任何关联。

查看源代码分析不一致变化一般不发生在克隆粒度特别大的代码片段中的原因，主要是由于克隆粒度较大的代码片段一般数量相对较少，其中一部分在进化过程中不发生改变，一部分发生改变后粒度减少或克隆代码片段消失，所以发生不一致变化的可能性较少。而克隆寿命特别长的代码片段一般克隆粒度不会特别长，这是因为克隆粒度较大的代码片段不稳定，在进化过程中发生修改后，内容相对变化较大，无法被克隆检测工具识别，即被认定为非克隆代码片段，因此寿命一般不会特别长。以上分析的具体示例如图 5-20 所示，图中每个大写英文字母表示部分源代码，原克隆群中的某一克隆代码片段表示为{A，B，C，D}，在进化过程中，其结构和内容更改较大，变为右侧框图中的形式，导致只有{A}这一部分被识别为克隆代码，使得克隆代码片段的粒度减少，有时甚至会出现无法被识别为克隆代码的情况，即克隆代码片段消失。

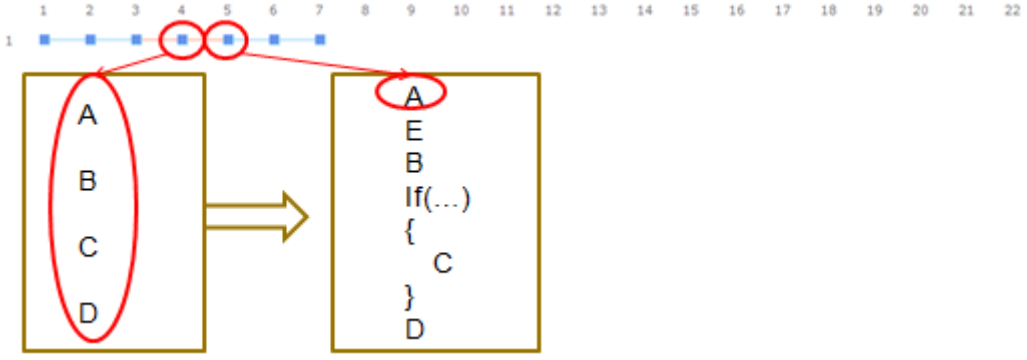


图 5-20 测试系统容量实验示例图

5.4.3 相似度实验

选取克隆相似度（即克隆群内代码片段的相似程度）和 9 维进化度量进行聚类，聚类结果的统计表格如表 5-14、5-15、5-16 所示，个数与比例 P1 的饼状图如图 5-21

所示。分析聚类结果，发现克隆代码片段样本恰好按照克隆寿命 age 的长短完全分离开来。分别统计每类中克隆相似度为 1.0 的代码片段个数(即完全相同的代码片段)，并计算比率 P1、P2。其中 iTextSharp 测试系统在本组实验中的效果不好，主要原因是其克隆群内代码片段的相似程度普遍都很高，甚至完全一样。

表 5-14 Java 语言测试系统相似度实验聚类结果

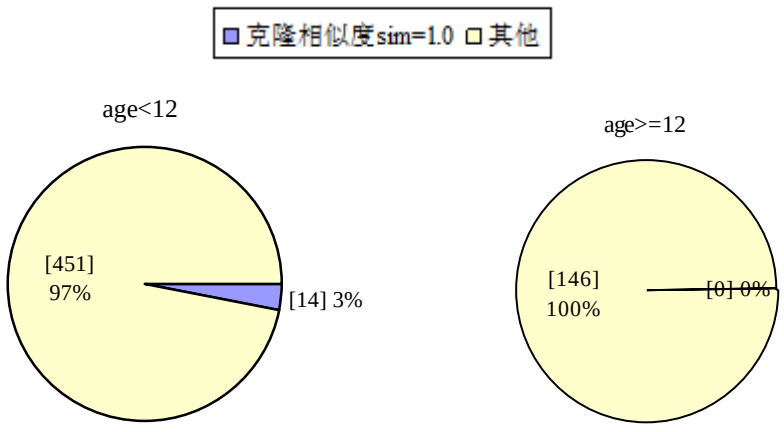
			DNSJava		jEdit	
个数统计		克隆寿命	age<12	age>=12	age<=13	age>13
			465	146	6451	185
		克隆相似度 sim=1.0	14	0	732	4
比率统计	P1	克隆相似度	3.01%	0	11.35%	2.16%
	P2	克隆相似度	100%	0	99.46%	0.54%

表 5-15 C 语言测试系统相似度实验聚类结果

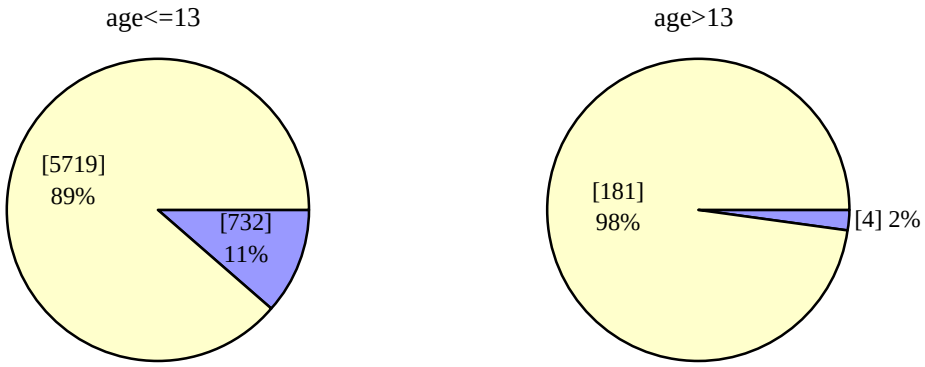
			wget		conky	
个数统计		克隆寿命	age<5	age>=5	age<=9	age>9
			350	28	817	111
		克隆相似度 sim=1.0	32	0	60	6
比率统计	P1	克隆相似度	9.14%	0	7.34%	5.41%
	P2	克隆相似度	100%	0	90.90%	9.10%

表 5-16 C#语言测试系统相似度实验聚类结果

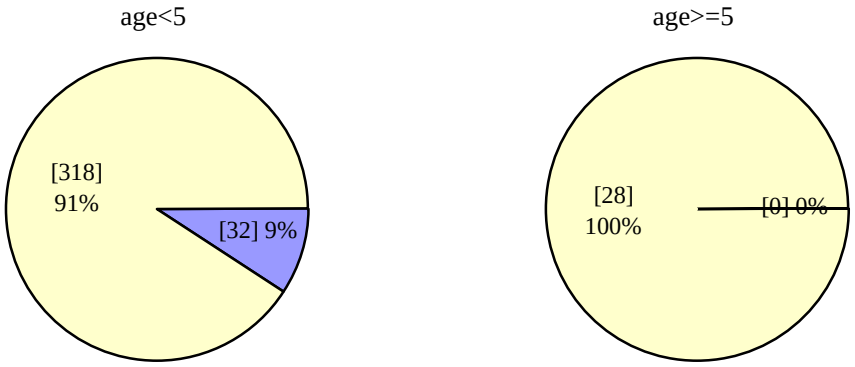
			ProcessHacker		iTextSharp	
个数统计		克隆寿命	age<=10	age>10	age<=10	age>10
			1755	489	8699	6698
		克隆相似度 sim=1.0	48	4	1765	1685
比率统计	P1	克隆相似度	2.74%	0.82%	20.29%	25.16%
	P2	克隆相似度	92.31%	7.69%	51.16%	48.84%



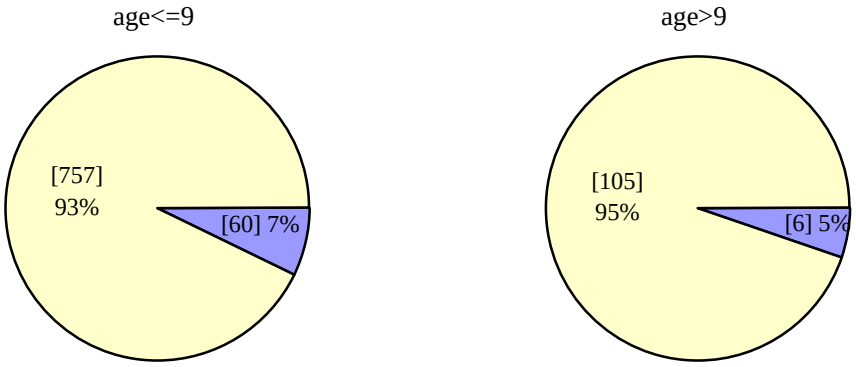
(a)DNSJava 系统相似度实验聚类结果饼状图



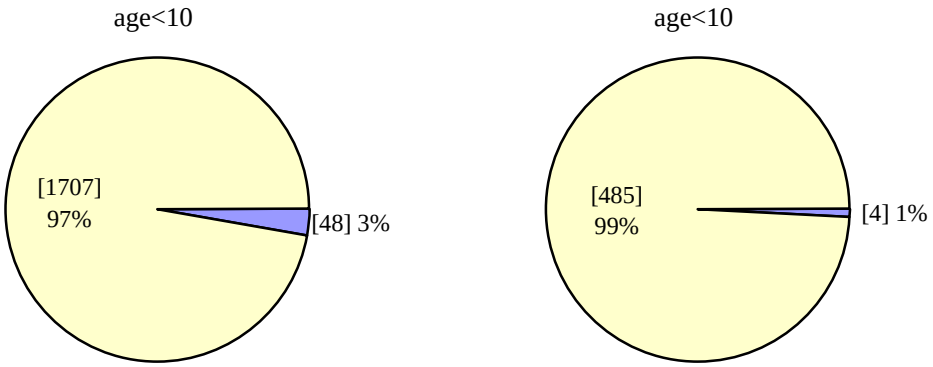
(b)jEdit 系统相似度实验聚类结果饼状图



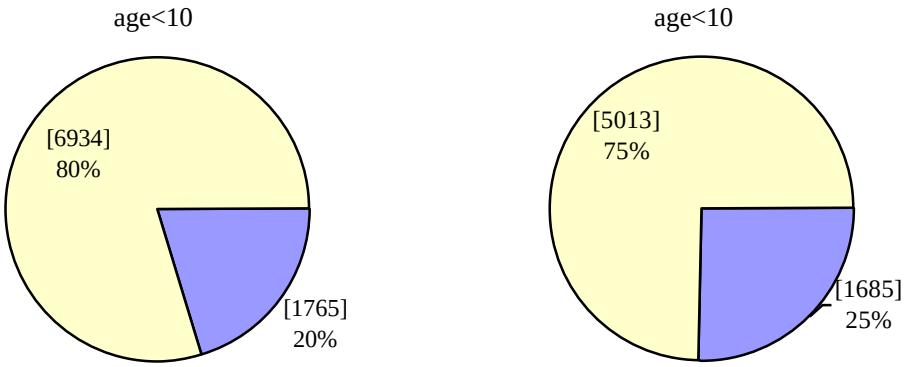
(c)wget 系统相似度实验聚类结果饼状图



(d)conky 系统相似度实验聚类结果饼状图



(e)ProcessHacker 系统相似度实验聚类结果饼状图



(e)iTextSharp 系统相似度实验聚类结果饼状图

图 5-21 测试系统相似度实验聚类结果饼状图

观察统计表格和饼状图，发现克隆寿命较短的代码片段组中克隆相似度为 1.0 的代码片段个数和比例一般都要高于克隆寿命较长的代码片段组。总结得到以下结论：克隆相似度特别高的代码片段一般为克隆寿命较短的克隆，（或者说一般不为寿命特别长的代码片段）。

查看源代码分析原因主要有以下两种情况，一是克隆相似度很高的代码片段在后续版本中被删去，二是克隆相似度很高的代码片段所在文件在后续版本中被删去。这导致找不到相映射的克隆群，使得克隆群消失，所以克隆寿命较短。

5.5 本章小结

本章对克隆代码检索与克隆进化分析可视化系统的各个模块进行了测试，通过示例介绍了各个模块的操作流程和界面构成，并针对克隆进化分析模块进行了三组实验，分别从进化、容量和相似度方面归纳出了与克隆代码特征相关的结论。

结 论

本文主要介绍了克隆代码检索与克隆进化分析可视化系统的设计与实现过程。给出了系统的需求分析、功能设计、体系结构设计、模块设计及流程设计，并针对于各个不同模块给出了相应的类图设计、时序图设计以及具体的实现算法。最后对各个模块的功能进行了测试，并通过几组实验得到了对代码评价与管理有意义的相关结论。本文完成的具体工作如下：

（1）针对于克隆代码相关的大量文本信息并不便于理解与分析的问题，本文提出并实现了克隆家系的两种可视化视图，所有克隆家系的整体可视化视图和单个克隆家系的详细可视化视图，并支持友好的用户交互功能。

（2）作为克隆代码检索与克隆进化分析的基础工作，本文实现了克隆代码片段静态度量与进化度量的提取功能。

（3）为了使得用户更快捷和方便的查找克隆代码进行深入研究工作，本文实现了两种方式的克隆检索工作：一种为指定代码片段的检索，用户可以从外界载入代码片段，也可以指定当前测试系统中感兴趣的克隆群中的代码片段；另一种为指定度量值的检索。

（4）本文提出了一种新的基于无监督学习的克隆进化分析方法，基于无监督学习模型分不同度量实验组对代码片段进行聚类，观察不同类别中代码片段的特性，归纳得出了一些有价值和有意义的结论，为进一步进行克隆代码管理与有害性评价提供指导。

（5）结合克隆家系提取器与以上功能，本文实现了一个以软件版本库为输入，提供克隆代码检索、克隆家系可视化以及克隆进化分析的原型系统，并以多个开源软件系统作为测试对象，验证了系统的有效性。

本文的系统功能较为完善，但是仍然存在着一些方面的不足。在克隆代码可视化方面，本文偏重与克隆进化信息的图形化，对于不同克隆群中的克隆代码片段内部关联没有深入探究。在克隆进化分析方面，可以考虑采用不同的聚类方法优化实验结果。另外，在聚类结果的分析中，如果能够结合测试软件系统开发人员修改不同版本代码时的日志信息，则能更好的理解克隆代码在进化过程中的变化规律，对克隆管理更加有帮助。

参考文献

[1] Roy C K, Cordy J R. A survey on software clone detection research[R]. Technical Report 541, Queen's University at Kingston, 2007.

[2] 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.

[3] Cordy J R. Comprehending reality-practical barriers to industrial adoption of software maintenance automation[C]//Program Comprehension, 2003. 11th IEEE International Workshop on. IEEE, 2003: 196-205.

[4] 史庆庆, 孟繁军, 张丽萍, 等. 克隆代码技术研究综述[J]. 计算机应用研究, 2013, 30(6): 1617-1623.

[5] Ducasse S, Rieger M, Demeyer S. A language independent approach for detecting duplicated code[C]//Software Maintenance, 1999.(ICSM'99) Proceedings. IEEE International Conference on. IEEE, 1999: 109-118.

[6] Kapser C, Godfrey M W. " Cloning considered harmful" considered harmful[C]// Reverse Engineering, 2006. WCRE'06. 13th Working Conference on. IEEE, 2006: 19-28.

[7] Rieger M, Ducasse S, Lanza M. Insights into system-wide code duplication[C]// Reverse Engineering, 2004. Proceedings. 11th Working Conference on. IEEE, 2004: 100-109.

[8] Lague B, Proulx D, Mayrand J, et al. Assessing the benefits of incorporating function clone detection in a development process[C]//Software Maintenance, 1997. Proceedings., International Conference on. IEEE, 1997: 314-321.

[9] Kim M, Sazawal V, Notkin D, et al. An empirical study of code clone genealogies[J]. ACM SIGSOFT Software Engineering Notes, 2005, 30(5): 187-196.

[10] Kim M, Notkin D. Using a clone genealogy extractor for understanding and supporting evolution of code clones[C]//ACM SIGSOFT Software Engineering Notes. ACM, 2005, 30(4): 1-5.

[11] Aversano L, Cerulo L, Di Penta M. How clones are maintained: An empirical study[C]//Software Maintenance and Reengineering, 2007. CSMR'07. 11th European Conference on. IEEE, 2007: 81-90.

[12] Thummalapenta S, Cerulo L, Aversano L, et al. An empirical study on the maintenance of source code clones[J]. Empirical Software Engineering, 2010, 15(1): 1-34.

[13] Bakota T, Ferenc R, Gyimothy T. Clone smells in software evolution[C]//Software Maintenance, 2007. ICSM 2007. IEEE International Conference on. IEEE, 2007: 24-33.

[14] Saha R K, Asaduzzaman M, Zibran M F, et al. Evaluating code clone genealogies at release level: An empirical study[C]//Source Code Analysis and Manipulation (SCAM), 2010 10th IEEE Working Conference on. IEEE, 2010: 87-96.

[15] Saha R K, Roy C K, Schneider K A. An automatic framework for extracting and classifying near-miss clone genealogies[C]//Software Maintenance (ICSM), 2011 27th IEEE International Conference on. IEEE, 2011: 293-302.

[16] Göde N, Koschke R. Studying clone evolution using incremental clone detection[J]. Journal of Software: Evolution and Process, 2013, 25(2): 165-192.

[17] Nguyen T T, Nguyen H A, Al-Kofahi J M, et al. Scalable and incremental clone detection for evolving software[C]//Software Maintenance, 2009. ICSM 2009. IEEE International Conference on. IEEE, 2009: 491-494.

[18] Lozano A, Wermelinger M. Tracking clones' imprint[C]//Proceedings of the 4th International Workshop on Software Clones. ACM, 2010: 65-72.

[19] Duala-Ekoko E, Robillard M P. Tracking code clones in evolving software[C]// Proceedings of the 29th international conference on Software Engineering. IEEE Computer Society, 2007: 158-167.

[20] Duala-Ekoko E, Robillard M P. Clone region descriptors: Representing and tracking duplication in source code[J]. ACM Transactions on Software Engineering and Methodology (TOSEM), 2010, 20(1): 3.

[21] Adar E, Kim M. SoftGUESS: Visualization and exploration of code clones in context[C]//ICSE. 2007, 7: 762-766.

[22] Saha R K, Roy C K, Schneider K A. Visualizing the evolution of code clones[C]// Proceedings of the 5th International Workshop on Software Clones. ACM, 2011: 71-72.

[23] Kim M, Bergman L, Lau T, et al. An ethnographic study of copy and paste programming practices in OOPL[C]//Empirical Software Engineering, 2004. ISESE'04. Proceedings. 2004 International Symposium on. IEEE, 2004: 83-92.

[24] Cai D, Kim M. An empirical study of long-lived code clones[M]. Fundamental approaches to software engineering. Springer Berlin Heidelberg, 2011: 432-446.

[25] Krinke J. A study of consistent and inconsistent changes to code clones[C]//Reverse Engineering, 2007. WCRE 2007. 14th Working Conference on. IEEE, 2007: 170-178.

[26] Krinke J. Is cloned code more stable than non-cloned code?[C]//Source Code Analysis and Manipulation, 2008 Eighth IEEE International Working Conference on. IEEE, 2008: 57-66.

[27] Krinke J. Is cloned code older than non-cloned code?[C]//Proceedings of the 5th International Workshop on Software Clones. ACM, 2011: 28-33.

[28] Saha R K. Detection and analysis of near-miss clone genealogies[D]. University of Saskatchewan, 2011.

[29] Zibran M F, Saha R K, Asaduzzaman M, et al. Analyzing and forecasting near-miss clones in evolving software: An empirical study[C]//Engineering of Complex Computer Systems (ICECCS), 2011 16th IEEE International Conference on. IEEE, 2011: 295-304.

[30] Mondal M, Rahman M S, Saha R K, et al. An empirical study of the impacts of clones in software maintenance[C]//Program Comprehension (ICPC), 2011 IEEE 19th International Conference on. IEEE, 2011: 242-245.

[31] Mondal M, Roy C K, Rahman M S, et al. Comparative stability of cloned and non-cloned code: An empirical study[C]//Proceedings of the 27th Annual ACM Symposium on Applied Computing. ACM, 2012: 1227-1234.

[32] Saha R K, Roy C K, Schneider K A, et al. Understanding the evolution of type-3 clones: an exploratory study[C]//Mining Software Repositories (MSR), 2013 10th IEEE Working Conference on. IEEE, 2013: 139-148.

[33] Saha R K, Roy C K, Schneider K A. gcad: A near-miss clone genealogy extractor to support clone evolution analysis[C]//Software Maintenance (ICSM), 2013 29th IEEE International Conference on. IEEE, 2013: 488-491.

[34] Zibran M F, Saha R K, Roy C K, et al. Evaluating the conventional wisdom in clone removal: a genealogy-based empirical study[C]//Proceedings of the 28th Annual ACM Symposium on Applied Computing. ACM, 2013: 1123-1130.

[35] Duala-Ekoko E, Robillard M P. Clonetracker: tool support for code clone management[C]//Proceedings of the 30th international conference on Software engineering. ACM, 2008: 843-846.

[36] Bazrafshan S. Evolution of near-miss clones[C]//Source Code Analysis and Manipulation (SCAM), 2012 IEEE 12th International Working Conference on. IEEE, 2012: 74-83.

[37] Bettenburg N, Shang W, Ibrahim W M, et al. An empirical study on inconsistent changes to code clones at the release level[J]. Science of Computer Programming, 2012, 77(6): 760-776.

[38] Barbour L J. Empirical studies of code clone genealogies[J]. 2012.

[39] Pate J R, Tairas R, Kraft N A. Clone evolution: a systematic review[J]. Journal of Software: Evolution and Process, 2013, 25(3): 261-283.

[40] 王海, 林云, 彭鑫, 等. 基于分组的代码克隆增量检测方法 2013 年 12 月 10 日[J]. 计算机科学与探索.

[41] Wang X, Dang Y, Zhang L, et al. Can I clone this piece of code here?[C]// Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering. ACM, 2012: 170-179.

[42] 慈萌. 基于 CRD 克隆群映射的克隆家系提取方法研究[D]. 哈尔滨工业大学大硕士论文. 2013

[43] 李智超. 基于支持向量机的克隆代码有害性评价方法研究[D]. 哈尔滨工业大学, 2013.

[44] C.K.Roy, J.R.Cordy, NICAD: Accurate Detection of Near-Miss Intentional Clones Using Flexible Pretty-Printing and Code Normalization[C]//The 16th IEEE International Conference, 2008：172-181

哈尔滨工业大学学位论文原创性声明和使用权限

学位论文原创性声明

本人郑重声明：此处所提交的学位论文《克隆代码检索与克隆进化分析可视化系统的设计与实现》，是本人在导师指导下，在哈尔滨工业大学攻读学位期间独立进行研究工作所取得的成果，且学位论文中除已标注引用文献的部分外不包含他人完成或已发表的研究成果。对本学位论文的研究工作做出重要贡献的个人和集体，均已在文中以明确方式注明。

作者签名：　　日期：　　年　　月　　日

学位论文使用权限

学位论文是研究生在哈尔滨工业大学攻读学位期间完成的成果，知识产权归属哈尔滨工业大学。学位论文的使用权限如下：

（1）学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文，并向国家图书馆报送学位论文；（2）学校可以将学位论文部分或全部内容编入有关数据库进行检索和提供相应阅览服务；（3）研究生毕业后发表与此学位论文研究成果相关的学术论文和其他成果时，应征得导师同意，且第一署名单位为哈尔滨工业大学。

保密论文在保密期内遵守有关保密规定，解密后适用于此使用权限规定。

本人知悉学位论文的使用权限，并将遵守有关规定。

作者签名：　　日期：　　年　　月　　日

导师签名：　　日期：　　年　　月　　日

致　谢

硕士研究生生活的两年时光就要过去了，在这段时间的课题研究过程中，我受到了很多人的帮助与爱护，他们不仅仅对我的研究起到了的推动的作用，更给予我人生道路上的指引。

首先要感谢我的导师苏小红教授。真的荣幸能够成为苏老师的学生，从本科毕业设计到硕士研究生三年来的学习生活中，苏老师一直细心的指导我们。依照我们的个人兴趣，帮助我们选择论文题目；并在每次会议的报告中帮助我们指出研究中的问题，提供给我们好的解决思路；对待我们的文献总结和论文都会认真的阅读和修改。苏老师还时常分享给我们一些对生活对个人发展有意义的启示，她对待工作和研究的态度给了我很深的影响。

感谢王甜甜老师。每次组会时给予我们课题上的指导，推荐给我们优秀的文献资料，给我们研究中存在的疑惑提供建议，推进我们的工作进度，让我们更清楚的看到研究重点，她温和的性格使得我们小组的成员都更加的亲近。

感谢张凡龙师兄。师兄总是积极的和我沟通论文的进展，指出我的研究中存在的缺陷与不足。在论文研究中遇到了困难，师兄会及时的帮助我寻求解决办法，给了我很多启发，使得我的论文顺利完成。

感谢一起走过两年时光的同学：郭琦、董冲、杨劭君。他们是我在学习生活中的伙伴，共同关注彼此的论文进展，互相的沟通中缓解了很多工作和生活上的压力，帮助我减轻了烦恼。

感谢实验室的其他同学和老师，给我的研究提供了无私的帮助。

最后还要感谢我的父母。他们是我最坚实的依靠，无论何时都给予我最大的支持与鼓励，让我时刻都充满勇气。

我会牢牢记住这些帮助过我的每一位老师与同学，踏踏实实的走好未来的每一步。