南京大學

NANJING UNIVERSITY

研 究 生 毕 业 论 文

（申请硕士专业学位）

论 文 题 目 基于程序切片的测试代码抄袭检测系统的设计与实现

作 者 姓 名 段定

专 业 名 称 工程硕士（软件工程领域）

研 究 方 向 软件工程

指 导 教 师 刘嘉 副教授 何铁科 助理研究员

2020 年 5 月 9 日

学　　号：MF1832034

论文答辩日期：2020 年 5 月 21 日

指 导 教 师：　　（签字）

The Design and Implementation of Test Code Plagiarism Detection System Based on Program Slicing

By

Ding Duan

Supervised by

Associate Professor **Liu Jia**

A Thesis
Submitted to the Software Institute
and the Graduate School
of Nanjing University
in Partial Fulfillment of the Requirements
for the Degree of
Master of Engineering

Software Institute

May 2020

南京大学研究生毕业论文中文摘要首页用纸

毕业论文题目：基于程序切片的测试代码抄袭检测系统的设计与实现

工程硕士（软件工程领域） 专业 2018 级硕士生姓名： 段定

指导教师（姓名、职称）： 刘嘉 副教授 何铁科 助理研究员

摘 要

软件测试是保障软件产品质量的重要手段之一，也是软件教育课程的重要组成部分。为提高学习效率，降低学习成本，出现了大批专注于软件测试技术训练的在线实践平台。尽管当前众多在线测试实践平台已能够实现大规模测试开发训练，但学生们并不是在封闭现场进行训练，他们是在不同地点进行在线训练，此时存在互相抄袭的可能性很大，如何保证训练质量成为难题。因此测试代码抄袭检测成为所有在线测试实践平台切实管控训练质量的关键举措之一。然而，通过人工审核的方式对海量测试代码实行检测并不可取，需要耗费大量的人力资源，由此如何实现自动化代码抄袭检测以降低审核成本，是目前面临的挑战之一。

为解决上述问题，本文对单元测试框架下设计的测试代码和生产代码深入分析，探究两者间存在的潜在差异，设计实现了一种基于程序切片的测试代码抄袭检测系统。为实现高质量的测试代码抄袭检测，本系统创新性地提出了静态双向程序切片技术，基于待测方法，从非标准测试代码中提取有效的测试片段，并进一步计算测试片段之间的相似度，基于相似度进行抄袭分析。根据此思路，本系统可划分为待测方法提取、测试片段提取、相似度计算、抄袭分析与测试报告生成共五个功能模块。

本系统前端采用 Angular2 框架技术，利用 Spring Boot 技术搭建后端框架，通过 MySQL 数据库存储数据信息。为保障系统高可扩展性与可用性，系统选取 Nginx 实现负载均衡，选用 Redis 作为缓存。

经过一系列严格的系统测试，本系统已实现功能性需求与非功能性需求，符合预期设想。通过在大量真实数据集上进行分析验证，实验结果证明本系统可以有效地检测出测试代码的抄袭行为，并且通过性能分析证实了本系统具有较好的健壮性。因此，本系统可以提升测试代码抄袭检测准确性，极大地节省了所需投入的人力成本，本系统对测试抄袭检测生态的建设起到了积极促进作用。

关键词：测试代码，测试片段，抄袭检测

南京大学研究生毕业论文英文摘要首页用纸

THESIS: The Design and Implementation of Test Code Plagiarism Detection System Based on Program Slicing

SPECIALIZATION: Software Engineering

POSTGRADUATE: Ding Duan

MENTOR: Associate Professor Liu Jia

Abstract

Software testing is one of the important means to ensure the quality of software products, and it is also an important part of the software education curriculum. In order to improve learning efficiency and reduce learning costs, a large number of online practice platforms focusing on software testing technology training have emerged. Although many online test practice platforms have been able to achieve large-scale test development training, students do not train in a closed field, they train online in different places, there is a great possibility of copying each other at this time, how to ensure the quality of training becomes a problem. Therefore, the test code plagiarism detection becomes one of the key measures for all online test practice platforms to effectively control the training quality. However, it is not advisable to detect massive test code by means of manual audits, which requires a lot of human resources. Therefore, how to realize automatic code plagiarism detection to reduce audit cost is one of the current challenges.

In order to solve the above problems, this thesis deeply analyzes the test code designed under the unit test framework and production code, explores the potential differences between them, designs and implements a test code plagiarism detection system based on program slicing. In order to achieve high-quality test code plagiarism detection, this system innovatively proposes a static two-way program slicing technology, extracting effective test fragments from non-standard test code based on the method under test, further calculates the similarity between test fragments, and carries out plagiarism analysis based on similarity. According to this idea, the system can be divided into five functional modules: method under test extraction, test fragment extraction, similarity calculation, plagiarism analysis and test report generation.

The front-end of the system adopts the Angular2 framework technology, uses the Spring Boot technology to build the back-end framework, and stores information data through the MySQL database. In order to ensure the high scalability and availability of the system, Nginx is selected to achieve load balancing and Redis is selected as the cache.

After a series of strict system tests, the system has achieved functional and non-functional requirements, which measures up to the anticipative assumptions. Through the analysis and verification on a large number of real datasets, the experiment results show that the system can effectively detect the plagiarism of test code, and the system has good robustness through performance analysis. Therefore, the system can improve the accuracy of test code plagiarism detection, greatly saving labor costs, and the system plays a positive role in promoting the construction of test plagiarism detection ecology.

Keywords: Test Code, Test Fragment, Plagiarism Detection

目录

表 目 录

图 目 录

第一章　引言

1.1　研究背景与意义

随着信息技术的进步，软件测试教育已成为当今的热点方向之一 [1]。为了提高学习效率，降低学习成本，出现了大量软件工程相关的实践平台，如 Leetcode，Pex4Fun [2]，MoocTest [1] 等。MoocTest 是最受欢迎的测试实践平台之一，其已承接了四届全国大学生软件测试大赛、ISTC 2017-2018、ICST 2019 和 ISSTA 2019 等测试竞赛以及全球数百项测试活动。

在线测试编程考试和测试竞赛拥有大量学生的参与，如何保证测试质量成为主要难题之一。例如，中国有 7000 多名学生参加了 CST 2017，并且非常值得注意的一点是，这些学生并不是在封闭的考场现场进行考试，他们是身在不同的地理位置进行线上考试。此时抄袭检测对于考试和竞赛中的在线编程至关重要。

然而，通过人工审核对大量测试代码进行检测并不可取，需要耗费大量人力资源，如何实现自动化抄袭检测以降低检测成本，是目前面临的挑战之一。近年来有部分研究者提出利用相似性度量技术来检测程序代码的抄袭行为。这些技术通过分析程序的语法、语义或结构来衡量程序代码的相似性。这些方法中包括 Plaggie [3]，FuzzyWuzzy [4]，Difflib [5] 等，它们主要是进行程序生产代码的抄袭检测，他们并没有专门针对测试代码的抄袭检测。目前在 Java 单元测试领域，还缺乏非常有效的抄袭检测技术，而将现有生产代码抄袭检测工具运用在测试代码抄袭检测场景下，检测准确率较低。学生为了躲避检测，常常在代码中修改变量或增加大量无用代码来混淆检测，现有检测工具常常会被这些手段影响，相似度降低而检测不出抄袭。基于以上痛点问题，本文提出了一个基于程序切片的测试代码抄袭检测系统，可以为在线测试实践平台提供自动化测试代码抄袭检测服务。

本文阐述的测试抄袭检测系统是唯一一个针对测试代码的抄袭检测系统，其能够提取有效测试片段，每一个片段都是一个最小粒度单位的测试，从而提高测试代码相似性检测的效果。运用“静态双向程序切片“技术先提取有效测试片段是本系统的一大创新点，其他方法一般都是把整段代码作为检测对象，而本系统采用的提前提取有效测试片段的技术，可以提取有效测试片段，摒弃冗

[1] http://www.mooctest.org/

余信息。因此，本文方法是在现有代码抄袭检测方法基础上，通过“静态双向程序切片”技术，将检测对象从整个文件细化到具体测试片段，对抄袭检测方法进行补充与完善。本系统需要通过人为设定的阈值对抄袭行为进行界定。当然，已有的这些方法无一例外都需要通过人为设定一个阈值，当一对学生的代码相似度大于阈值时，则可认为两名学生之间存在抄袭。

实验结果表明，本系统方法可以有效地提高测试代码相似性度量精确度，提高测试代码抄袭检测准确率，从而提升软件测试训练质量，提升软件测试教育质量。

1.2　国内外研究现状

代码抄袭检测对于软件教育是必不可少的。近年来众多研究者提出了不同的相似性度量方法，为软件抄袭检测提供了一些基础可用的工具。根据不同的相似度检测角度如基于 Token、基于结构、基于语法和基于语义等，可以将目前已有的许多抄袭检测工具分为不同的种类。

Plague [6] 由 G.Whale 于 1988 年提出，它是一种基于结构的抄袭检测器，可用于检测用 C 语言编写的 PLA-Giarism 代码。Yap 使用运行 karp-rabin 贪婪字符串平铺（RKS-GST）作为迈克尔 · 怀斯提出的比较算法。RKS-GST 适用于抄袭检测，因为它优先处理较长的子串，并且不受子串顺序的影响。Moss [7] 和 JPLAG [8] 提供了一种用于检测抄袭行为的 Web 服务。但是，如果源代码是机密信息，则不适合将代码发送到 Moss 或 JPLAG。Plague 在功能上与 JPLAG 相似，其源代码是开放的，允许其他人员在此基础上进行二次开发。但是，上面列出的相似性度量和工具主要用于检测业务生产代码抄袭，没有特定的测试代码抄袭检测工具。

测试的相似性度量是迄今为止与测试代码抄袭检测最为相关的度量，主要集中在测试报告或测试用例的相似性度量上。通常，测试报告由自然语言文本和一些屏幕截图组成。现有的测试报告相似性度量研究主要用于解决众包测试中的大量冗余问题。如冯洋等人 [9] 将自然语言处理技术与图像分析技术相结合，测量测试报告的相似性。TERFUR [10] 是一个模糊聚类框架，用于对众包测试报告进行聚类，以降低手动检查的成本。这些措施也可能适用于测试代码抄袭检测，但到目前为止，我们还没有看到任何实践。此外，它们都没有考虑测试行为（例如，测试用例用于测试对应待测方法）。对于测试用例相似性度量，大量的研究集中在测试用例优先级划分上 [11]。房春荣等人 [12] 使用程序实体的有序序列来度量测试用例的相似性。Noor 等人 [13] 使用测试用例的方法调用序列来度量相似度。另外，还有一部分人引入最近火热的机器学习。其基本类型

分为三种：监督学习、非监督学习和强化学习。由于代码的开源性、检测的复杂变化多样性，机器学习其技术本身的安全性问题受到诸多限制，因此对传统机器学习算法的抄袭检测进行改进显得非常迫切，许多研究者在抄袭检测领域里做出了许多工作，提出了许多有效检测算法、模型和系统。如使用 DSVM 网络进行检测，以及提出了一种混合算法，将二进制分类器和 k-NN [14] 算法结合起来以进行检测。吴斐 [15] 采用利用 N-gram [16] 表示待检测程序代码文本, 将程序代码文本转化成 N-gram 集合, 统计 N-gram 的出现频度, 将 N-gram 集合和频度放入向量空间中进行相似度计算，Xiong [17] 等人提出了将代码在编译和反汇编、编译线性化标识符、代码风格等方面的相似度作为神经网络的输入向量，尽可能识别语义方面的抄袭。他们主要关注的测试用例及代码是标准的和最小粒度的。然而，当用于非标准测试代码时，它们的性能并不令人满意。通过将切片技术引入被测方法的锚定和从非标准测试代码中提取片段，本系统的技术可以有效地提高测试代码抄袭检测中相似性度量的性能。

1.3　本文的主要工作

结合单元测试的知识，本文总结了以下两个测试代码的特征，这两个特征一般生产代码并不具备，有助于促进测试代码抄袭检测。第一，在 JUnit [18] 测试框架下设计的测试代码包括四个基本部分：初始化语句，用于设置初始状态；执行语句，调用正在测试的方法；断言语句，检查测试结果以及清理语句，执行必要的清理。测试代码非常结构化。第二，生产代码中的大多数方法都不是独立的，即方法存在调用其他方法，而测试代码中的测试用例相对独立，易于切片。

基于以上对测试代码特征的观察，本文创新地提出了一个专门用于检测测试代码抄袭的检测系统。本测试抄袭检测系统是第一个针对测试代码的抄袭检测系统，通过考虑测试代码和生产代码之间的差异，有效地结合现有相似性度量方法来深入检测测试代码的抄袭行为。本系统提前从非标准测试代码中提取有效的测试片段，通过自动消除测试代码中的混淆代码以还原测试用例的粒度。测试片段是测试粒度为“1”的测试用例的表示，换句话说，也就是测试抄袭检测系统度量测试片段的相似性而不是整段测试代码，从而提高了抄袭检测的准确性。最后，测试抄袭检测系统可视化每个测试代码的关键信息，包括测试目标，测试片段，测试片段的相似性，并报告检测结果。

本测试代码抄袭检测系统具有以下特点：基于有效的测试片段，测试抄袭检测系统为检测者提供了一个高效的自动检测测试代码抄袭的方法；测试抄袭检测系统是一个基于 Web 的工具，提供友好的界面，便于老师或检测人员使用。除了报告检测结果的描述外，测试抄袭系统还将锚定方法的所有抄袭测试片段

可视化，以便人工进行进一步检查；测试抄袭检测系统具有高度的可扩展性。测试抄袭检测系统中的相似度检测器等模块可较便捷地使用更高级的相似性度量算法等来代替。

本文所设计和实现的测试抄袭检测系统是慕测平台考试系统的一个子系统，其主要目标是针对在线考试人员分散，不在现场无法防止抄袭，人工检测抄袭工作量太大的问题，避免老师或检测人员耗时耗力地人工检测，提高检测准确性。

为了解决上述问题，本系统是从以下两个角度去进行设计的：

第一个角度是通过采用静态双向程序切片技术，进行待测方法的精确提取和对应的测试片段切片提取。老师或检测人员在检测学生或学生提交的测试代码时常常受到代码行数过多以及学生或学生为躲避检测，故意修改变量等的困扰。结合查阅的相关资料和业务现状，本系统没有采用传统的直接将两个学生的测试代码文件进行比对的方法，而是引入基于静态的双向切片技术来处理待测方法和测试代码片段，该技术主要适用于对待测方法的精确提取和对测试代码片段的精确提取。提取待测方法时，首先要对待测程序进行定位，分割和提取，关键点在于各个项目的待测代码各自具体方法定位及分割，代码和参数信息精确提取。测试代码片段提取时，关键难点在于如何对测试同一个待测方法的测试代码信息进行精确提取，这一阶段的关键在于测试代码与待测方法对应关系的确定，前向和后向相关代码语句的获取。该技术是通过分析测试代码的特点，通过提取待测方法与测试代码的对应关系，使得测同一个待测方法的测试代码片段得到精确提取，同时去除无关代码，解决直接对测试代码文件进行对比信息冗余、精确度较差的问题。

另一个角度是通过打通从检测任务配置到系统执行检测任务再到最后检测结果展示的全流程，实现全方位自动化检测，大大节省人力成本，方便检测人员在自动化给出的检测结果及检测报告基础上进行抄袭分析。本系统提供给老师或检测人员待测方法代码详情信息、测试片段详情信息、测试代码抄袭检测概要信息、抄袭关系关联视图、测试代码抄袭检测详情报告等。测试代码抄袭检测配置页面方便用户快速、便捷地对本次检测任务进行配置，而检测结果展示页面则通过既有测试代码抄袭检测概要信息视图及抄袭关系关联视图，又可以生成及下载抄袭检测详细报告，即可满足用户快速查看概要信息的需求，又可以满足用户查看详细检测报告的需求。

对此，本文阐述了一个测试抄袭检测系统的设计与实现。系统主要工作流程如下：首先，从待测项目中提取待测方法，并保存到数据库中，进行持久化存储后，作为后续测试片段提取的辅助输入；接着，对学生提交的测试代码进行分

析，结合静态双向程序切片技术，基于之前提取出的待测方法，按待测方法分类提取出测试代码片段；接着，针对测试代码片段计算其相似度；接着，对测试片段相似度基于阈值进行抄袭分析；最后，生成抄袭检测报告。根据以上工作流程，本系统可分为五个主要模块：待测方法提取模块，测试片段提取模块，相似性计算模块，抄袭分析模块和测试报告生成模块。本文使用在软件测试考试/竞赛中产生的真实项目测试代码数据集来评价测试抄袭检测系统。

综合以上思路及设计，本系统整体建立在 Spring Boot 框架之上，单独部署为外部服务，对其他服务耦合性很低。系统与其他服务之间数据交换主要基于 HTTP 协议，对于数据格式的变化可以非常迅速地反应，便捷易用。系统使用成熟数据库 MySQL，运用主从同步，对数据库中数据实时备份，从而提高数据可靠性；使用 Ngnix 和 Redis 中间件将服务端采取无状态化形式，从而提高水平拓展性；系统在事件处理等中，设计为异步形式，进一步降低耦合程度。

1.4　本文的组织结构

本文的组织结构如下：

第一章，引言部分。对测试抄袭检测的背景以及测试抄袭检测系统在各个领域的设计实现及意义做了一个简单的介绍。

第二章，相关技术综述。对于文中所需要使用到的高可用、前后端技术框架以及系统所调用的工具算法等进行了简单介绍。

第三章，系统的需求分析与概要设计。对系统的需求进行了分析，根据系统需求对系统的整体结构、各个模块、数据库以及算法进行了设计。

第四章，系统的详细设计与实现。对各个模块的具体设计和实现进行了详细描述。

第五章，系统的测试和实验评估。对系统进行功能和性能测试，将系统与业内常用抄袭检测工具进行对比实验，分析实验结果以验证系统可靠性。

第六章，总结和展望。对论文所完成的工作进行总结，并进一步阐述了系统未来工作重点。

第二章　相关技术综述

2.1　代码相似性度量技术

自 20 世纪 70 年代以来，人们引入了大量的工具进行完成数据源相似性代码测量工作，常用于解决代码克隆检测、违反软件许可和软件剽窃等问题。这些工具使用不同的计算方法来计算两个程序的相似性，大致可被分为基于度量、基于文本，基于令牌、基于树和基于图的方法。检测软件的早期方法相似性基于度量或软件度量。早期的代码相似性检测工具由 Ottenstein [19] 创建，基于 Halstead 复杂性测量，在编程课上注册学生程序中 47 个样本中发现了一对抄袭的样本。由经验分析，与其他较新的方法相比，这种方法的效率较低。

而一些基于文本代码相似性度量技术，主要是在比较两个字符串的基础上对源代码序列执行相似性检查，能够找到源代码的精确副本，并通过语法和语义修改找到相似的代码。一些支持技术被合并来处理语法变化，如变量重命名。有几个代码相似性分析器可以计算文本相似性。广泛使用的字符串相似性方法之一是找到最长的 NiCad [20] 克隆检测器采用的子序列（LCS），Plaggie [3]，YAP 的第一个版本，和 CoP。其他除了 LCS 之外，具有字符串匹配算法的基于文本的工具包括但不限于到，Duploc、Simian 和 PMD 的复制/粘贴检测器（CPD）。

为了从代码文本中抽象出一部分逻辑，很多代码相似性度量技术对源代码进行转化。标记流可以用于程序代码相似性度量。抽象级别可以通过定义令牌的类型来调整。依靠在如何定义标记的问题上，标记流可以将文本差异转化为只捕获程序的抽象序列。例如，如果程序中的每个字替换为 W 标记，语句 “int x=0;” 将类似于 “String s= “hi” ; “因为他们都共享一个形如” W W=W; “的标记流。不同的相似性度量例如后缀树、字符串对齐、Jaccard 相似性等，都可以应用标记序列或标记集。依赖工具包括 Shalock [21]，BOSS，Sim，YAP3，JPlag [8]，CCFinder，CP Miner，Moss [7] 和源代码相似性检测器系统（SCSDS）。基于令牌的技术被广泛使用在源代码相似性度量和非常有效的规模上百万 SLOC，例如，基于令牌的大规模克隆检测工具 SourcererCC。

这些代码相似性度量技术中，FuzzyWuzzy [4] 是其中效果非常好并且非常成熟的一个工具。使用格式塔模式匹配 PythonNGram 比较文本序列使用 n-grammes，FuzzyWuzzy 通过模糊搜索比较文本序列，使用模糊字符串标记匹配，做近似字符串的部分匹配，以及 scikit learn 的余弦相似性，提供数据挖掘和数据分析的机器学习库。并采用了 diff（经典的文件比较工具）和 bsdiff（二进制文件比较工

具）。通过使用 diff 或 bsdiff，我们发现使用提高模糊重复数据消除的敏感度阈值是非常重要的，因此本系统选用 FuzzyWuzzy 作为代码相似度量辅助工具。

2.2　程序切片技术

从切片方式的角度来看，切片的定义是基于切片准则的概念。根据 Weiser [22]，切片标准是一对 <p，V>，其中 p 是程序点，V 是程序变量的子集。一个切片准则上的程序切片 <p，V> 是保留程序点 p 处的原始程序 V 中的程序变量，即程序点 p 处的 V 与原始点相同程序和切片。作为原始的行为程序必须保存在任何输入上，韦瑟的切片被称为静态切片，用来区分从其他需要行为的切片形式保留在程序输入的子集上。Weiser 已经证明了满足此要求的语句子集是不可判定的。但是，近似值可以是通过计算一组数据流的最小解找到控制流图（CFG）节点与与该节点相关的变量符合切片标准。Weiser 提出的算法有另一种常用的定义：切片由一组程序语句和谓词组成直接或间接影响。在执行 p 之前，变量在 V 定义中，提出了另一种算法，将切片计算为程序的向后遍历依赖图（PDG），程序表示其中节点表示语句和谓词，而边包含有关控件和数据依赖项的信息。基于 PDG 的分层算法 <p，V> 类型的标准，其中 p 是程序点，V 是在 p.A 切片中引用的变量集对于这样的切片标准，包含一组节点直接或间接影响节点 p1 处 V 中的变量。这种切片形式定义后向切片，与前向切片相反，定义为一组程序语句和谓词受变量值计算的影响程序点 p。霍维茨等人。将基于 PDG 的算法扩展到基于系统依赖性计算程序间切片图表（SDG）。作者证明算法比原算法更精确 Weiser 的过程间切片算法，因为它解释了过程调用上下文。最近计算切片的算法改进图可达性在中给出。

另一种程序静态切片的并行算法由 Danicic [23] 等人提出的控制流图，程序被转换成并发网络并行执行产生切片的进程。也有人提出了计算任意控制下的向后静态切片流程和指针，提出了静态切片的不同应用。加拉赫和莱尔介绍了分解切片并讨论了它在软件维护中的应用。一个分解切片是相对于变量 v 定义的，独立于任何程序点。它是由计算的静态切片的并集变量 v 和所有可能的程序点 p。程序切片的其他应用包括软件测试，程序调试，测量，验证，程序并行化，程序集成，反向工程和程序理解，程序重组，并确定可重用函数。尤其是内部可重用函数的标识需要不同的切片标准定义允许，每当已识别搜索的函数，威瑟的切片标准只提供终点和输出集要提取的函数的变量。Visaggio 添加了根据切片标准搜索函数。他们介绍转换切片的概念，作为计算给定程序点的输出变量值从输入变量的值。计算转换切片类似于静态向后切片的计算，但只要定义输入变量的值包含在切片。另一方面，Cimitile 等人定义了不同的切片标准包括开

始和要提取的函数的结束语句。切片是在这其中形成一个 CFG 的一个子图。另外定义了一种确定切片标准的方法：用术语表示的搜索函数的规范先决条件和后决条件。

因为程序切片技术可以有效地提取出针对性待检测片段，防止出现对代码整体文本检测效果较差的问题，因此本系统结合前向程序切片技术和后向程序切片技术，在系统中运用静态双向切片技术，提取代码片段，以提高检测针对性。

2.3　基于 Angular2 的前端框架

AngularJS，即 Angular1，首先是由 Google 发布了于 2010 年发布。AngularJS 第一个推出“双向数据绑定”，引开发者们踊跃讨论，大量界面代码开发者们开始尝试使用。而 Angular2 [24] 是在 2016 年出现，在 ES6 基础上研发，明显提升性能水平，各种 Web 组件均可支持。此处简要介绍 Angular2 的诸多优点：优秀的跨平台水平是最显著的优点。当今 Web 平台诸多优势均被 Angular2 合理运用，免安装、便于跨平台移植，给用户一种非常便捷的体验。技术想法来自 Native React 等多个成熟框架，应用在此基础上采用独创方式构建。高速性能也是较为显著的优点。Angular2 代码依据虚拟机模板转变，优化方式也是针对 JS 虚拟机而采取，很有针对性。将 HTML 和 CSS 的秒级快速显示页面技术运用在服务器端渲染上，可以支撑 Node,PHP,.NET 等多种服务器。快速加载也是本技术通过新组件路由器可以实现的，通过让用户只需加载渲染请求需要的页面代码。高生产率同样也是 Angular2 的一大优势。易用而强健的模版语法，帮助 Angular2 中 UI 视图可以快速搭建 [25]，既保证了框架给予的高生产率，又将手写代码的优点保留。快速搭建环节、组件增加、部署检测等均可通过 Angular CLI 命令行工具进入。Angular2 还具备高可测试性。逻辑与对 DOM 操作被进行解耦，测试应用程序重点对待，可测试性代码显著增加。对于开发者而言，开发难度是决定是否选择该技术的一大考量，Angular2 不仅提高服务端和客户端代码可复用性，而且可以并行开发，大幅降低开发难度。Angular2 引入松耦合的思想，将前端开发模块化，将传统服务端的读物通过依赖注入给予 Web 客户端，比如视图控制的独立 [26]。

由于 Angular2 有这么多特点与优势，本系统使用基于 Angular2 的前端框架，客户端可以独立于服务端进行开发及交互。在系统开发过程中的效率大幅提升，开发完成后的测试也更加方便，渲染和加载页面更加便捷，将最好的交互体验给予用户。

2.4　基于 Spring Boot 的后端框架

Spring 框架是一种常见的应用型框架技术，目前大量主流的 Java 程序都会选择采用该框架，它的火热程度达到了已经常常被用来补充甚至替代 EJB 模型。大量优质的项目提升手段均存在于 Spring 框架，例如核心特性控制反转，对象容器化则是利用依赖注入实现控制反转，声明式事务管理是采用面向切面编程技术有助于相助提高开发效率，缩减开发周期。Spring 框架具有控制反转（IOC）特性，项目维护和测试因该特性而更加方便。

Spring Boot 使得使用者可以轻松地创建独立的、生产级的、基于 Spring 的应用程序，developers 可以“只是运行”即可使用该框架。Spring Boot 对 Spring 平台和第三方库有一个独到的见解，这样 developers 就可以从最少的麻烦开始了。大多数 Spring 引导应用程序需要最少的 Spring 配置。

Spring 具有以下特征：第一，创建独立的 Spring 应用程序。第二，直接嵌入 Tomcat、Jetty 或 Undertow（不需要部署 WAR 文件）。第三，提供自以为是的“starter”依赖项，以简化构建配置尽可能自动配置 Spring 和第三方库。第四，提供生产就绪的特性，如度量、健康检查和外部化配置。第五，完全没有代码生成，也不需要 XML 配置。它不仅支持所有流行数据访问框架，还可以对开发过程进行基于框架的事务管理。因此数据访问以及管理变得更加方便快捷。原本 Spring 框架研发者并不打算去迎合当时流行的 MVC 机制，但是最终大势所趋，还是引入了 MVC 机制，就有了 Spring MVC 框架。

Spring Boot 是一个新框架，它以特定方式配置的流程简化了 Spring 应用程序的初始构建和开发，以便对于 developers 来说，建模等操作非常方便。Spring Boot 可以帮助 developers 快速构建 Spring [27] 框架。为了在使用中更加便于进行配置、调试、测试等，Spring Boot 框架中包括 Web 容器，并且可以非常方便地移植使用，并带来脚本语言编写效率 [28]，具有许多特点和优势。

Spring Boot 有很多非常优秀的性质，比如说，第一点，Spring Boot 为方便用户使用，提供了非常易用的默认配置。第二点，该项目构建速度快，无需配置即可自动集成第三方框架。第三点，只有 Auto Config 和 Java Config，它可以完全替换 XML 配置文件并简化框架配置过程。第四点是嵌入式的 Servlet [29] 容器，降低了开发环境的要求。可以直接使用命令执行项目，也可以使用 jar 包执行。第五点，它提供了 starter POM [30]，使得包管理非常方便，大大减少了 Jar Shell 和依赖 Shell。第六点，该框架对运行时的各种情况进行实时监控，并且可以方便运维人员进行维护或测试。第七点，一般常见工具根本无需特别设置，即可直接集成，其他的几乎所有第三方插件也非常易于集成。基于 Spring Boot 的这些优点，本系统的服务器采用 Spring Boot 构建，可以提高本系统整个后端架

构的搭建、开发、测试和部署的全流程效率。

2.5　微服务

在最初的软件开发中，通常选择单一框架，即所有业务都写在同一个项目中，对于意外更改很难维护 [31]。微服务（也称为微服务体系结构）是一种体系结构样式，它将应用程序构造为高度可维护和可测试,松耦合的，独立部署的，围绕业务能力组织的，由一个小团队拥有的微服务架构。能够快速、频繁、可靠地交付大型、复杂的应用程序。它还使组织能够快速发展其技术堆栈。

微服务的优点如下：使得软件开发更加简单，对于复杂的大型系统，其中包括很多模块，如果采用单体架构模式，很难对系统中的各个模块有一个清晰的认识，或者很难独立使用。因此，此时若使用微服务架构，会能够使整个复杂系统的框架更容易理解。因此微服务架构会使得整个系统更加易于维护，质量更好或更容易更换。

微服务架构并不是一颗银弹。它有几个缺点。此外，在使用此架构时，必须解决许多问题。微服务架构模式语言是应用微服务架构的模式集合。它有两个目标：模式语言使用户能够决定微服务是否适合您的应用程序。模式语言使用户能够成功地使用微服务体系结构。单体架构模式是一个很好的起点，它是传统的架构风格，对于许多应用程序来说仍然是一个很好的选择。但是，它确实有许多限制和问题，因此对于大型/复杂的应用程序，更好的选择是微服务体系结构模式。

为了部署更多的应用程序，解决方案是部署自动化。微服务结合诸如 Jenkins、uDeploy、Capistrano、Chief、Puppet 或自定义脚本之类的工具，从而可以完全自动化微服务的部署。同样需要实现对部署中的一键式部署。

由于本系统中每个功能模块的服务可以划分为粒度较小的服务，可以在其他系统中使用，因此本系统使用微服务更容易、更灵活地拆分内部模块，减少了耦合性，提高了内聚性。

2.6　基于 Docker 的容器技术

容器技术的主要作用是实现逻辑的打包，是一种虚拟化技术。基于这种技术方法，可以将待打包的应用与真实操作步骤进行分离，有助于开发工作者能够在多种目标环境（如，公有云，私有云等）中高效地部署目标应用程序。具有强专业性的容器技术可以显著地突出开发团队与维护人员的关注内容：开发团队重点主要放在应用程序的逻辑思路上，相比之下，维护人员更加注重应用的

维护与部署，而着眼于细节（如软件版本等）的挖掘。与虚拟机类似，容器技术可以对具有依赖项的应用进行打包，对外提供一种包含独立运行环境的给服务，包含多种的轻量级操作单元。

更重要的是，该技术允许在操作系统内核中部署运行大量容器。具体来说，容器间相互共享操作系统内核，使得内存占用较少，启动加快。开发者可以快速创建独立的预测环境，而不局限于某一具体环境，这显著地降低了开发过程中工作压力。为了将系统中包括内存、硬盘等各种资源进行虚拟化管理，容器技术将整个部署环境需要的资源通过沙盒操作系统技术进行隔离处理。有许多可用的容器格式，Docker [32] 是最流行的容器格式，具有最佳的开发环境。因此，本系统也使用 Docker 作为容器。

2.7　本章小结

本章主要介绍了开发本系统所使用的相关理论与技术。首先介绍了代码相似性度量技术，包括本文使用工具 FuzzyWuzzy，以及代码片段提取技术程序切片技术，其次介绍了前端框架 Angular2 以及本系统使用的服务端框架 Spring Boot，接着简要介绍了微服务，将其用于实现服务粒度更小化以减少与其他服务之间的耦合度，最后阐述了容器技术 Docker，将其用于将服务沙盒化，便于服务部署等。

第三章　系统需求分析与概要设计

3.1　系统整体概述

本系统主要实现了测试代码抄袭检测，可提供高准确率抄袭检测、详细测试代码抄袭检测报告、代码之间的具体相似度和测试代码片段的详情信息。有测试代码抄袭检测需求的软件需求方在平台发布项目需求后，可以自行选择要检测的是哪些学生对，也可以自行选择要检测的是哪几道题目，并生成报告。另外，还可以返回检测出来的学生的测试代码之间的具体相似度矩阵和疑似抄袭的测试片段详情信息。

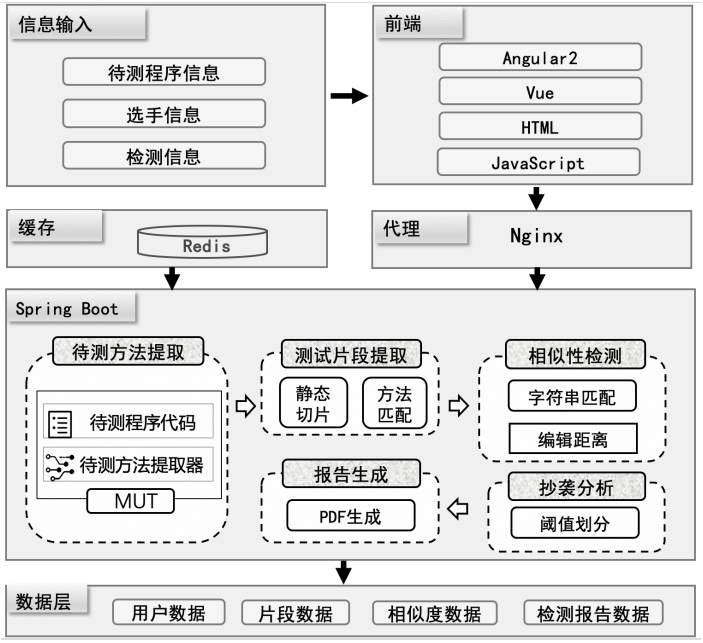


图 3.1: 整体框架图

本系统的整体框架图如图 3.1所示。前端服务采用了基于 Angular2 的前端框架，非常方便的前后端松耦合设计使得系统开发非常高效。前端界面在系统中向后端发送用户对其进行的各种 POST 请求操作，然后取得后端逻辑模块对其进行反馈的信息。前端界面获得待测程序信息、选手信息、检测信息输入等后，通过前端框架向后端发送。

系统后端其他模块都使用 Spring Boot 框架进行搭建及开发，主要包括待测方法提取模块、测试片段提取模块、相似度计算模块、抄袭分析模块以及报

告生成模块。其中待测方法提取模块内部包括待测程序代码和待测方法提取器两个部分，测试片段提取模块包括静态切片和方法匹配两部分，相似度计算模块则主要是基于编辑距离的字符串匹配。抄袭分析模块主要是基于阈值进行划分，而报告生成模块则是用于生成报告 PDF 文件。本系统使用 Redis [33] 缓存和 MySQL [34] 数据层来存储抄袭检测结果等数据。

表 3.1: 功能性需求列表

需求 ID	**需求名称**	**需求描述**
R1	查看考试信息	用户可以根据本人需要查看考试信息，包括考试名称、考试 ID、考试题目等。
R2	查看学生信息	用户可以根据本人需要查看参加考试的学生信息，包括学生编号、学生成绩等。
R3	查看待测方法详情	用户可以根据本人需要查看考试题目经待测方法提取后的待测方法详情代码。
R4	查看测试代码详情	用户可以根据本人需要查看学生提交的详细测试代码。
R5	查看抄袭检测概述	用户可以在检测结束后查看本次检测的结果概要信息。
R6	查看抄袭检测报告	用户可以在检测结束后查看本次检测生成的报告 PDF。

本系统核心逻辑部分由四个部分组成：待测方法提取器，测试片段提取器、相似性检测仪和抄袭分析器。测试抄袭检测系统旨在自动完成三个主要功能：提取有效的测试片段，测量测试片段的相似性，检测抄袭。首先，待测方法提取器静态地解析生产代码并提取所有待测方法；测试片段提取器负责碎片化任务，即提取从测试代码中锚定方法的有效测试片段。接着，相似性检测仪测量测试片段的相似性而不是原来的非标准测试代码本身。此外，抄袭分析器根据提取的测试片段的相似性，判断两个测试代码是否抄袭。如果测试代码相似度大于给定的相似性阈值，则将其视为抄袭。最后，可视化工具显示所有检测结果，包括抄袭的测试片段以及抄袭关系图，以便用户可以跟踪抄袭检测结果。

3.2　系统需求分析

3.2.1　功能性需求分析

根据本系统涉及到的用户和需求进行分析，对本系统需要实现的主要功能进行归纳。本系统功能性需求如表 3.1所示。

在用户方的业务流程中，首先用户方可以在系统配置页面中对抄袭检测进行配置，在考试/比赛项目列表中选取要检测的项目，查看相应考试信息，查看本次考试/比赛中参加学生信息，包括学生本身的信息以及提交测试代码的信息。当用户选取完要检测的项目后，用户可以选择对本次考试/比赛中所有参加学生进行检测，也可以选择通过配置界面选取其中部分参加学生进行检测，检测配置信息在确认后不可再改变。在抄袭检测过程中，选择生成自动化检测的报告由系统后台自动生成，另外检测会生成概要信息列表，具体相似度信息以及片段详情信息，检测成功后系统进入结果展示阶段。检测结果展示阶段用户可通过查看检测结果概要信息确认检测总体结果并验证与实际情况是否一致。

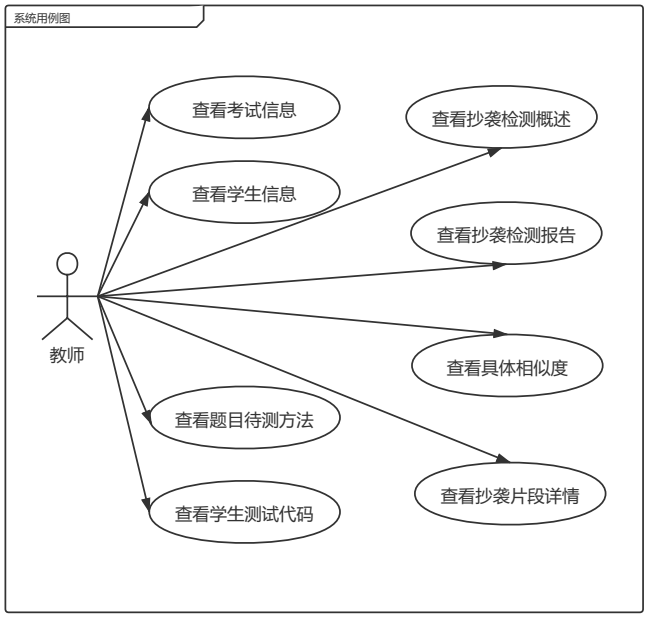


图 3.2: 系统用例图

用户如果想查看更详细的结果，可以选择查看具体检测报告。报告内包括检测时间、检测项目、检测范围、学生对相似度信息以及测试片段信息等。另外，用户可以选择查看学生代码之间具体相似度信息，包括代码中各个测试片段之间相似度。最后，用户如果想详细查看比较两个测试代码片段之间的直观相似情况，可以通过查看测试代码片段详情信息，具体查看某两个具体测试片

段的详细代码直观展示。

依据系统功能性需求分析，画出系统用例图，如图 3.2所示。本系统用例分别是查看考试信息、查看学生信息、查看考试/比赛的题目中的待测方法、查看学生的测试代码、查看检测结果的概要信息、查看检测报告、查看学生代码之间具体相似度信息、查看测试代码片段详情。

表 3.2: 查看考试信息用例描述表

ID	*UC*_1
用例名称	查看考试信息
用例参与者	教师 目的是教师查看本次考试/比赛的各种信息
前置条件	1. 教师成功登入系统 2. 系统已定义此用户角色
后置条件	无
基本操作流	1. 教师成功登入系统 2. 管理员点击选择某次比赛/考试 3. 系统显示该次考试/比赛对应的考试 ID、题目、时间等具体信息 4. 管理员点击查看某次考试/比赛的综合信息 5. 系统显示该次考试/比赛的最终综合信息

查看考试信息用例描述如表 3.2所示。进入本系统，在配置页面可以查看本系统中存在的所有考试列表，用户可以从中选择某次考试信息进行查看。考试信息包括考试 ID、考试名称、考试包含的题目和考试时间等具体信息。用户方可以通过系统配置对选择的考试或考试题目进行修改，可以自行添加或者移除要查看或检测的对象。在系统进入检测状态后，配置信息写入抄袭检测配置中，考试信息等配置无法再修改。

考试信息包括考试题目（待测程序）代码详情，用户可查看具体的详细代码文件内容，便于对后面提取出的待测方法以及检测结果中展示的测试目标进行对应，便于理解，其中待测方法即从考试题目程序中提取出来，而测试目标即与后面提取出的测试片段对应的待测方法。

表 3.3: 查看学生信息用例描述表

ID	*UC*_2
用例名称	查看学生信息
用例参与者	教师 目的是教师查看参加学生的具体信息
前置条件	1. 教师成功登入系统 2. 系统已定义此用户角色
后置条件	无
基本操作流	1. 教师成功登入系统 2. 教师点击查看某次考试/比赛中学生的信息 3. 系统显示该学生的详细个人信息，包括对应 ID、姓名、最终得分等 4. 管理员点击查看某位学生的提交的测试代码 5. 系统显示该学生提交的具体测试代码

查看学生信息用例描述如表 3.3所示。查看学生信息是本系统所有用户在检测配置阶段都会使用的功能，用户通过该功能了解某次考试具体有哪些学生，以及这些学生的具体信息，用户方还可以通过查看学生信息作为自己确定本次检测要选取的学生列表的依据，做到信息的作用最大化。

用户不仅可查看学生的基本个人信息，还可查看其参与某次考试/比赛的结果信息，包括对待测程序的分支覆盖情况、变异体杀死情况、考试得分等。

查看待测方法详情是本系统用户在点进配置详情界面后更细致查看信息的功能，考虑到本项目中每个考试的每个项目中具体待测方法都不同，故在界面中相应待测方法概述后添加可查看待测方法具体代码详情的按钮，即用户可选取关心的待测方法去查看具体代码。在选择某一待测方法代码详情进行查看后，用户可以更了解待测方法具体情况。用例描述如表 3.4所示。

待测方法是为后面进行测试片段提取提供基础保障的功能，当提取出待测方法后，才可以根据待测方法的锚定，去提取对待测方法进行测试的测试片段。本用例正是基于对待测方法的提取，持久化存储后，将待测方法详情信息返回给前端，交由前端进行展示。

表 3.4: 查看待测方法详情用例描述表

ID	*UC*_3
用例名称	查看待测方法详情
用例参与者	教师 目的是教师查看提取出来的待测方法的详情信息
前置条件	1. 教师成功登入系统
后置条件	无
基本操作流	1. 教师点击查看提取出来的待测方法 3. 系统显示提取出来的待测方法 4. 管理员点击查看待测方法的具体信息 5. 系统显示待测方法的具体信息

查看学生的测试代码功能主要适用于检测配置后期，用户可以查看被选中的学生其提交的测试代码详情信息，用户可以具体查看其提交测试代码中每一个方法以及每一个方法内容，处于检测阶段的学生测试代码无法被再次查看。查看学生的测试代码用例描述如表 3.5所示。

表 3.5: 查看学生测试代码用例描述表

ID	*UC*_4
用例名称	查看学生测试代码
用例参与者	教师 目的是教师查看学生提交的测试代码
前置条件	1. 教师成功登入系统 2. 已经定义这种用户角色
后置条件	无
基本操作流	1. 教师点击查看某位学生的测试代码 3. 系统显示该学生的测试代码 4. 管理员点击查看某位学生的测试代码的详细信息 5. 系统显示该学生的测试代码的详细信息

查看检测结果概要信息功能主要面向只需要对测试代码抄袭检测结果有一个宏观的、大体的认知的用户。在完成抄袭检测后，用户方可以第一时间在展示界面看到对于检测结果概要信息的展示，主要包括本次检测的项目名称、所有学生对数量以及其中抄袭学生对数量等。另外，在展示界面，本系统通过多样化的形式对概要信息进行展示，包括抄袭检测结果表和抄袭关系图等。用户方可以在本系统中上传、更新和查看概要信息。用户方也可以去界面抄袭检测结果生成处下载概要信息对应的检测文件。查看检测结果的概要信息用例描述如表 3.6所示。

表 3.6: 查看检测结果概要信息用例描述表

ID	*UC*_5
用例名称	查看检测结果的概要信息
用例参与者	教师 目的是教师查看测试抄袭检测概要信息结果的
前置条件	1. 教师成功登入系统 2. 已经定义这种用户角色
后置条件	无
基本操作流	1. 教师成功登入系统 2. 教师点击进行某次检测 3. 系统根据教师的配置进行检测 4. 教师点击查看本次检测结果 5. 系统显示本次检测结果的概要信息

查看检测报告用例主要面向的是需要查看详细检测信息的用户方。用户方在等待系统完成对测试代码抄袭检测的一系列操作，包括前面的待测方法提取、测试片段提取、相似度计算及抄袭分析后，可在系统界面检测结果信息生成处，通过按钮选择查看或下载检测 PDF 文件，并设定下载路径，系统会将持久化存储的检测报告下载到指定路径。用户能够在下载检测报告后，阅读检测报告。检测报告中包含更详细的抄袭检测信息，包括每一个检测对详细信息、详细相似度矩阵及测试片段详情代码。信息之间可以相互跳转，点击详情可以查看每个测试代码片段间的具体相似度值，点击具体相似度值可以查看片段详情。查看检测报告用例描述如表 3.7所示。

表 3.7: 查看检测报告用例描述表

ID	*UC*_6
用例名称	查看检测报告
用例参与者	教师 目的是教师查看测试抄袭检测结果的报告 PDF
前置条件	1. 教师成功登入系统 2. 已经定义这种用户角色
后置条件	无
基本操作流	1. 教师成功登入系统 2. 教师点击进行某次测试抄袭检测 3. 系统按照教师的配置进行抄袭检测 4. 教师点击查看本次检测报告的 PDF 5. 系统显示本次检测报告的 PDF

3.2.2　非功能性需求分析

由于在现实在线软件测试考试中，常常会有大量学生同时在线提交大量测试代码，对大量代码的检测需要系统有较好的性能，因此对系统的性能方面的各项非功能性需求需要严谨分析。经过细致分析，本系统需要满足的非功能性需求如表 3.8所示。

衡量系统是否有应用价值，首先第一点就是它的可用性，系统在长期上线使用过程中不能出现任何单点故障；当在线考试同时在线学生越来越多时，对系统相应时间性能及能承受的同时并发量有非常高的要求，处于正常连接网路，接口响应耗时等应该在一定范围之内；系统必须要对登录的用户身份进行筛选认证，保证安全性；本系统所有接口设计都必须满足普遍的 API 风格，用户界面必须有人性化提示语；同时，由于系统检测一次需要遍历大量学生代码，耗时很长，不能因为其中一个问题而使整个流程中断，因此对系统的健壮性有较高要求；为提升数据可靠性和数据存储效率，本系统需要考虑多处备份设定；本系统必须在稳定服务器上部署，系统给予的 API 和 SDK 必须尽可能常见；本系统之内以及对外的关键业务抽象接口，如配置系统、存储系统，都必须便于进行换代更新，达到可拓展性要求。

表 3.8: 非功能性需求列表

需求名称	需求描述
可用性	系统任何单点不出现缺陷; 系统可用性以年为单位，具有 99% 保证，如果因为特殊情况导致系统服务出错或当机，必须要在 20 分钟内启动或使用备用集群保证服务不中断。
性能	处于正常连接网路，后端接口响应 data 耗时应该小于 250ms，较为复杂的读写和查找操作时间消耗应少于 550ms。
安全性	本系统对于用户身份进行筛选认证，识别恶意用户，对于全部起源请求进行分辨。
易用性	本系统所有 API 接口都必须便于理解，有详细注释，取名方式选用比较常见认可的方式; 用户使用界面必须有易理解提示，符合常见人机交互常识。
健壮性	本系统必须能够在绝大多数的项目上跑成功；对于部分恶意脚本，能够保证系统正常运行。
伸缩性	当使用量、并发数溢出系统负荷时，必须可较快部署多节点、多集群，并且对于上游用户可见。
可拓展性	本系统关键对外接口可便于修改，如配置系统、存储系统，便于达到换代、提升等需求。

3.3　系统概要设计

结合前文功能性和非功能性需求，本小节详细介绍本系统概要设计。本部分主要从实现层面来进行系统服务划分和架构设计，首先介绍系统各个子模块怎么划分及如何交互，接着介绍整体实现物理架构。本小节还会借助“4+1”视图的形式来对系统整体架构视图进行多维度解读。

3.3.1　系统服务划分与架构设计

本小节从实现的角度介绍本系统的服务划分。本系统使用微服务架构，遵循高内聚低耦合原则，对外提供统一 RESTFUL 风格的接口，对内则按照微服务原则进行服务划分。

前端服务采用了基于 Angular2 的前端框架，非常方便的前后端松耦合设计使得系统开发非常高效。前端界面在系统中向后端发送用户对其进行的各种 POST 请求操作，然后取得后端逻辑模块的反馈信息。前端界面获得待测程序信息、选手信息、检测信息等输入后，通过前端框架向后端发送。该部分会以独立前端服务的形式独立进行打包部署，与后端低耦合。

后端的服务则是在 Spring Boot 框架内部进行划分，利用 Spring Boot 与微服务之间非常良好的结合能力，依据微服务划分准则，并结合整个测试代码抄袭检测流程的特点，进行划分。要检测测试代码，必须首先对待测方法进行提取，再提取完成后才可基于待测方法对对应测试片段进行提取，接着可运用工具对相似度进行计算，再基于阈值进行抄袭分析。最后，为方便用户查看，可再生成检测报告。由此，将本系统划分为待测方法提取服务、测试片段提取服务、相似度计算服务、抄袭分析服务、检测报告生成服务。结合 K8s 技术，对每个服务进行独立部署，单独使用容器资源，方便移植，每个服务模块都在前一个出发点到来之时启动，并在完成自身逻辑和向后一个服务模块发送信息。每个服务模块内部则是依据业务逻辑进行设计，并且预设拓展愿景，具有良好的可拓展性。在用户服务的设计过程中，出于减少重复检测以及提高检测可靠性的目的，本系统选择将检测结果信息数据存入数据层。

对于在高并发下的负载均衡问题，本系统选取了业界非常流行的 Nginx 组件配合 K8s 进行负载均衡。同时，在有大量外部请求时，在外部如何高效快速访问内部 K8s 集群，本系统利用了 Nginx 组件中的解决方案，提升了本系统对外高并发提供服务的能力。

对于数据层服务的设计，本系统主要采取了缓存与持久化存储相结合的方法。一方面，为了提升数据存储读取效率，将部分高频率读取及存储的数据放入缓存进行存储，这样利用缓存读取存储速率快的特性，提升系统运行效率。另一方面，有一些重要数据，或者需要在数据层持久化存储，以便以后使用或查证的数据，本系统选择将其存入 MySQL 数据库，可通过数据库访问接口进行增删改查。

3.3.2　架构视图

“4+1”视图法，是当前软件架构工程领域中最出名且被最多人使用的一种方法 [35]。首先，其中“1”代表场景视图，一般从用户角度出发，描述使用场景，本文的图 3.2用例图就是该视图的具体表现。另外的“4”表示逻辑视图、进程视图、开发视图以及物理视图，它们分别从用户服务视角，集成人员视角，系统运维人员视角以及程序开发人员视角对系统架构进行描述。本小节将分别描

述本系统架构对应的逻辑视图、进程视图、开发视图以及物理视图，对系统的逻辑、进程状态、以及开发部署等进行描述。

逻辑视图主要是分析系统功能性需求，即用户需要的功能系统如何提供。在逻辑视图表现形式中，最主流的常常为 UML 类图，通常通过将功能抽象成一个个对象类，以 UML 类图形式展现，如图 3.3所示，其中 ExtractService 表示提取功能服务，DataService 表示数据处理服务，FileSerive 表示文件处理服务，ReportService 表示报告处理服务，UserService 表示用户服务，TestCodeService 表示测试代码处理服务，系统中的基础功能对象则如下：提取范围（ExtractCriteria）、文件服务（FileService）、提取服务（ExtractService）、软件服务（SoftwareService）、检测报告服务（ReportService）、数据服务（DataService）等。UserService 提供用户登录与注销等功能。

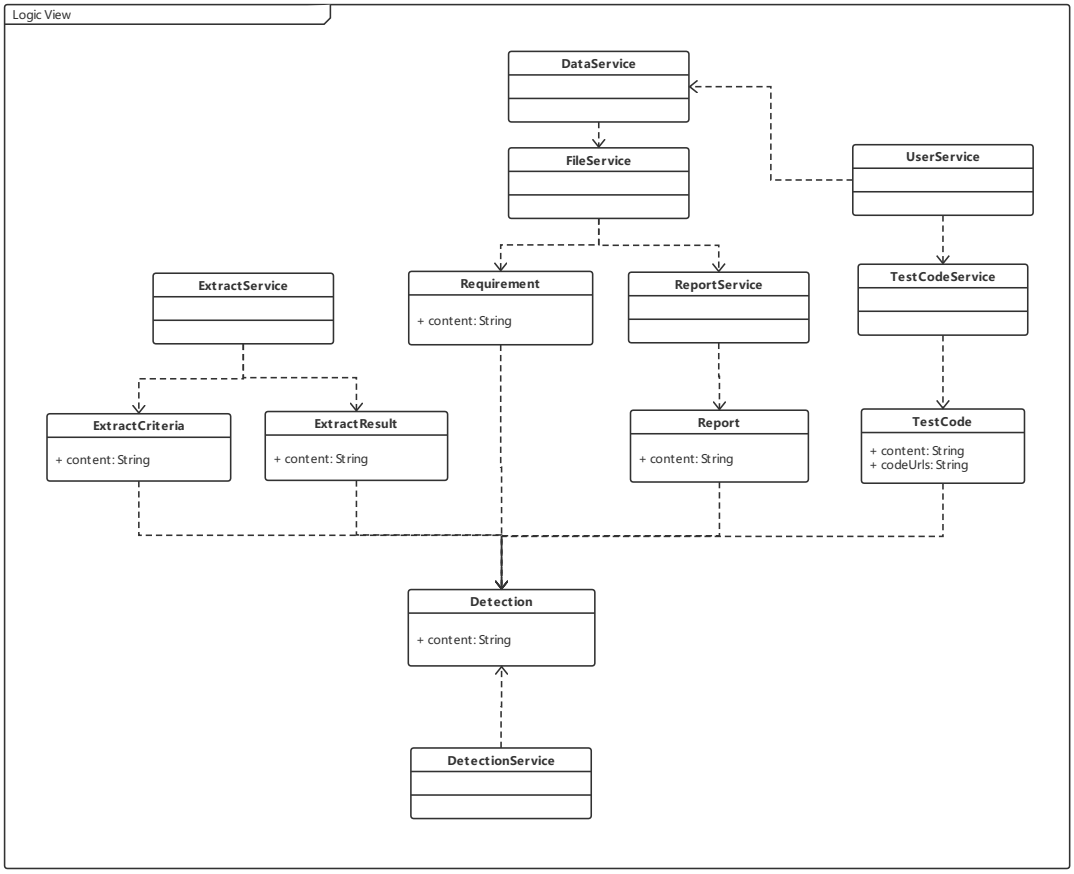


图 3.3: 逻辑视图

进程视图用于描述系统在并发性、分布性、系统完整性上的问题，进程视图中的主要抽象是进程对应关系，即关注对象为进程控制与执行以及进程间的互

相通信。如图 3.4所示，本系统中，当某服务意图访问其他服务时，会先向系统服务端查询其通信地址，再根据通信地址与 API 进行服务调用。

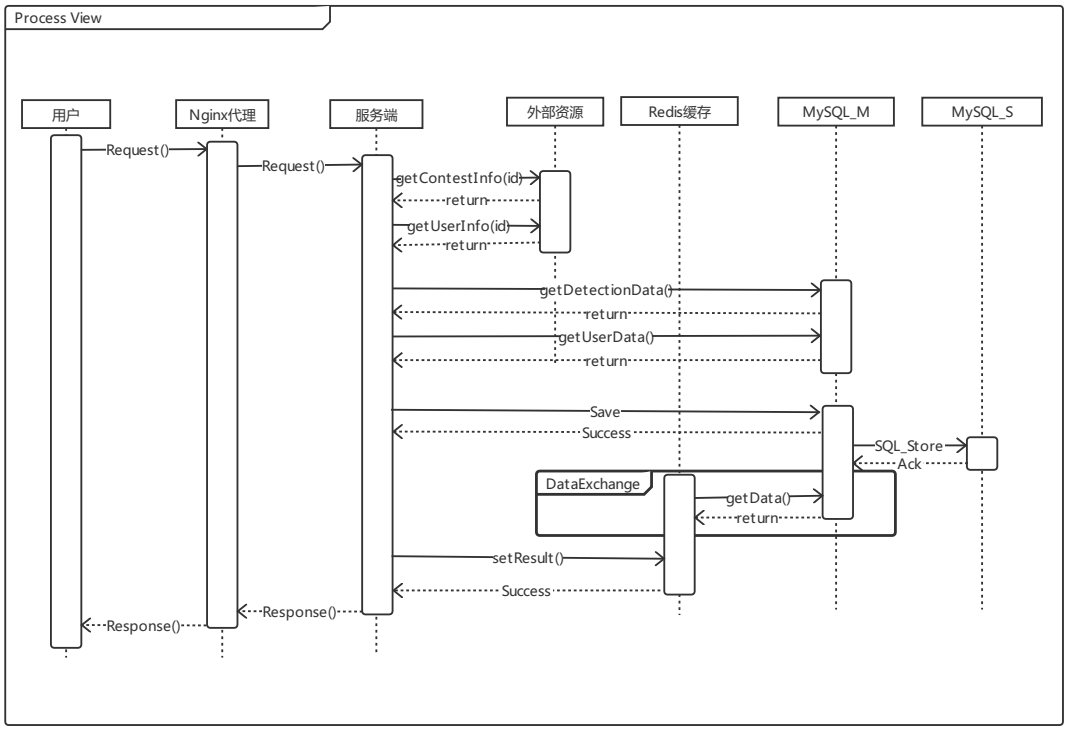


图 3.4: 进程视图

当用户在前端发起请求后，整个系统的进程启动，首先系统会将这些请求交给 Nginx 代理去处理，它通过代理以后将 Request() 发送给系统服务端。系统服务端通过 getContestInfo() 方法去向外部资源请求考试信息及题目代码等，获得返回后进行持久化存储。另一方面，服务端同理向外部资源请求获得学生相关信息及提交的测试代码。接着，在进行果抄袭检测的工作后，与 Redis 缓存以及 MySQL 数据层对检测结果等进行存储。Redis 缓存与 MySQL 之间通过 DataExchange 进行数据交换。

开发视图主要描述软件在开发环境下各项组织。本系统开发视图如图 3.5所示，对于前端 UI 部分，主要包含内容为数据结构 Model ，静态资源 JavaScript 、CSS 和 HTML ；而对于服务端部分，根据业务逻辑可划分成不同层次与功能，依次为 Controller 、Service 、Logic 等不同包。其中，Controller 包的职责在于对系统前端请求的接收与响应，和对控制器的调节；而与 Controller 包对接的 VO 包主要用于完成相关业务逻辑的交付与实现工作。Logic 包负责测试代码抄袭检测的业务逻辑处理。Service 包主要实现测试代码抄袭检测中各个微服务的实现

及处理。JDBC 包与 Redis-Connector 包分别提供缓存处理请求和 JDBC 接口，协助 Dao 包实现简单数据业务封装并与数据库进行交互。

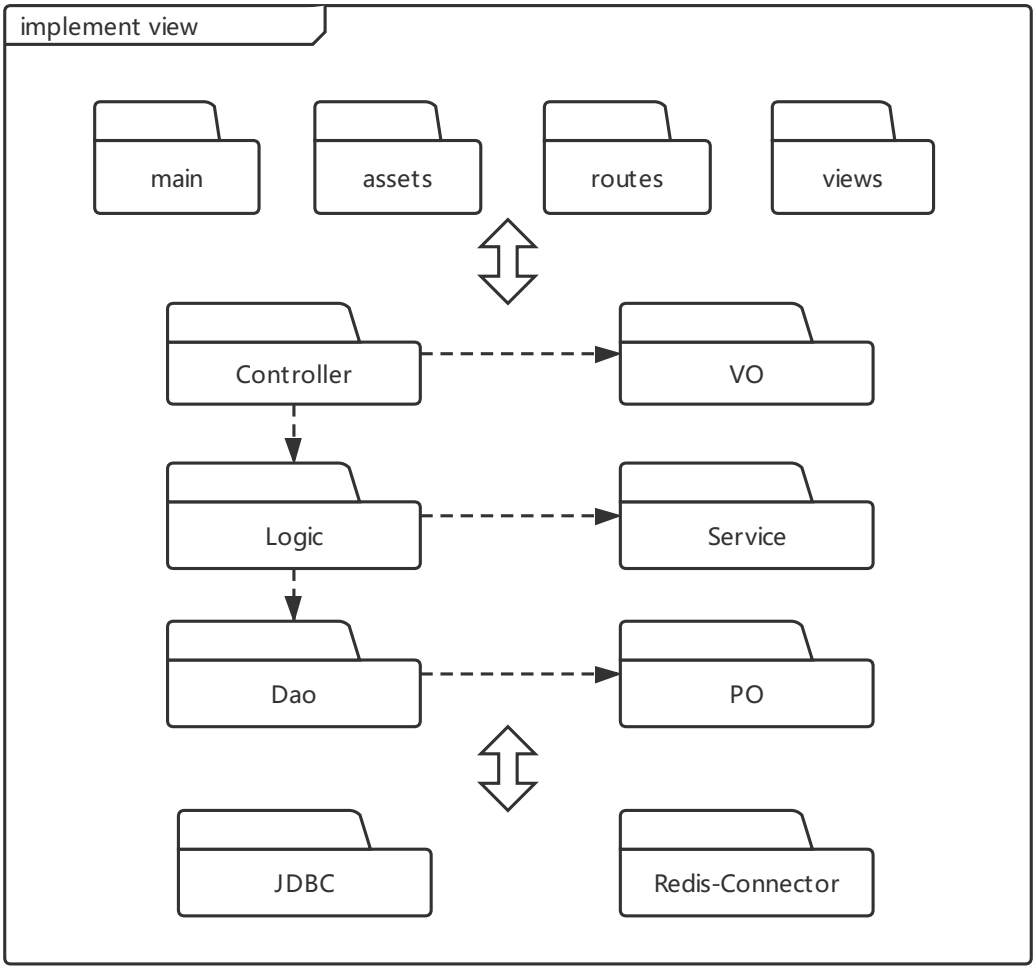


图 3.5: 开发视图

物理视图，是从部署工程师角度解读系统，关注软件的物流拓扑结，以及如何部署机器和网络来配合软件系统的可靠性、可伸缩性等要求。本系统全局部署视图可见图 3.6所示，用户通过 Chrome 等浏览器向 Web 服务器发送 HTTP 请求进行交互，前端服务器中的 Angular 2 应用程序成功接收请求后进而发送 HTTP 请求申请访问系统后端。在经过 FireWall 对恶意用户的筛选过了后，访问请求最先抵达 Nginx 服务器实现负载均衡，进而被分发至若干个以 Docker 为服务容器的应用服务器中。与此同时，应用服务内部的用户 Session 允许由 Redis 进行托管暂存，后期请求则可被随机分配给某一应用服务器以确保应用服务的水平可扩展化。此外，应用服务器可利用 Redis Protocol 获取需要的高频数据。

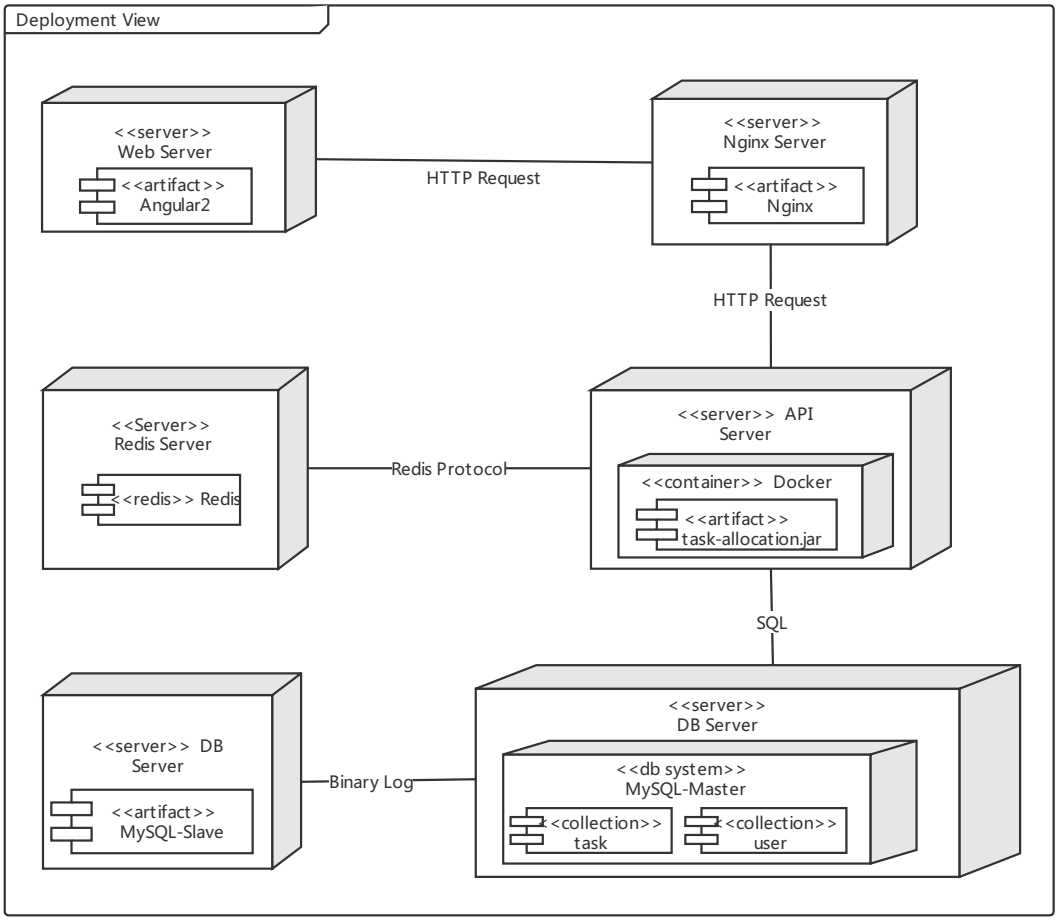


图 3.6: 物理视图

3.4　数据模型设计

本系统选用关系型数据库 MySQL 来对抄袭检测中的考试/比赛题目信息、学生代码信息、检测结果信息等进行存储，抄袭检测中操作到的业务数据都被存在 MySQL 中。因为这些数据都是结构化的数据，选用关系型数据库比较适合，而 MySQL 是关系型数据库 [36] 中主流的数据库，因此本系统最终选用来 MySQL 作为数据库进行存储。对存储在数据库中的业务对象进行枚举，主要是考试/比赛题目数据、学生代码信息数据、检测结果信息数据等。

表 3.9是考试/比赛待测方法在数据库中的字段描述。主要记录某个项目的待测方法数据的唯一标识，该方法的访问权限，该方法的参数等数据。

表 3.10是从学生提交的测试代码中提取的测试片段在数据库中的字段描述。主要记录该测试片段所属学生的 id，提取出来的测试代码片段内容，该代码片段的字符长度等数据。

表 3.9: 待测方法属性设计表

字段	类型	含义
id	int	一条数据的唯一标识
access	varchar	该方法的访问权限
arguments	varchar	该方法的参数
class_name	varchar	该方法所属的类名
is_constructor	varchar	该方法是否是构造方法
method_id	int	该方法的 hash 唯一标识
method_name	varchar	该方法的方法名
subject	varchar	该方法所属的项目名

表 3.10: 测试片段属性设计表

字段	类型	含义
id	int	一条数据的唯一标识
cid	int	该测试片段所属学生的 id
fragment	varchar	提取出来的测试代码片段内容
length	int	该代码片段的字符长度
mid	int	该测试片段测试到的待测方法的 id
state_num	int	该测试片段中的语句数量
subject	varchar	该测试片段所属的项目名

表 3.11是对学生两两进行相似度检测之后得到的相似度信息在数据库中的字段描述。主要记录两个学生该测试片段的相似度值，相似度值的类型：0 表示为百分数，1 表示为小数，进行相似度检测的学生的 id 等数据。

表 3.11: 相似度属性设计表

字段	类型	含义
id	int	一条数据的唯一标识
category	int	相似度值的类型：0 表示为百分数，1 表示为小数
cid1	int	进行相似度检测的学生一的 id
cid2	int	进行相似度检测的学生二的 id
mid	int	两位学生相似度检测的测试片段测到的待测方法的 id
sim_value	double	两个学生该测试片段的相似度值
subject	varchar	该测试片段所属的项目名

表 3.12: 抄袭分析属性设计表

字段	类型	含义
id	int	一条数据的唯一标识
cid1	int	进行抄袭分析的学生一的 id
cid2	int	进行抄袭分析的学生二的 id
label	int	表示该学生对是否抄袭的标签，0: 不抄袭，1: 抄袭

表 3.12是对学生对之间是否存在抄袭在数据库中的字段描述。主要记录两个学生该测试片段的相似度值，相似度值的类型：0 表示为百分数，1 表示为小数，进行相似度检测的学生的 id 等数据。

3.5　算法设计

算法是指解决具体问题的准确而完整的描述。高效的计算方法确实会在软件产品开发的流程中起到非常重要的作用。为了以非常少的成本、非常高的速度和非常好的质量完成本系统的开发，本文设计了以下算法。

3.5.1　测试抄袭检测系统整体算法设计

本抄袭检测系统整体算法设计概述如下。其中算法伪代码如算法 1所示。它将被测软件（$Software\ Under\ Test, SUT$）、每个学生的测试代码（$Test\ Code, TC$）、

相似度阈值 t 和相似度函数 $funSim(TF_{i,k}, TF_{j,k})$ 作为输入，最后输出候选抄袭对 $(PlagiarismPair, PP)$。

Algorithm 1 本抄袭检测系统整体算法

```
1:  // 第一阶段: 测试片段提取
2:  for all TC_i in TC do
3:      initialize an empty set TF_i, which is utilized to record the Test Fragments of ith
        student;
4:      for all M_j in MUT do
5:          // 根据待测方法 M_j，提取对应的测试片段 TF_{i,j}
6:          analyze TC_i to extract the test fragment TF_{i,j} that corresponds to M_j, and add
            TF_{i,j} into the ith student's test fragment list TF_i;
7:      end for
8:  end for
9:  // 第二阶段: 相似度计算
10: initialize Three-Dimensional Similarity Array TDSA
11: for i = 1 to num_student-1 do
12:     for j = i + 1 to num_student do
13:         for all M_k in MUT do
14:             // 获取进行相似度计算的测试片段对
15:             get the test fragments, TF_{i,k} and TF_{j,k}, that correspond to M_k from TF_i and
                TF_j respectively;
16:             if both TF_{i,k} and TF_{j,k} are not NULL then
17:                 TDSA[i][j][k] = funSim(TF_{i,k}, TF_{j,k});
18:             end if
19:         end for
20:     end for
21: end for
22: // 第三阶段: 抄袭分析
23: for i = 1 to num_student-1 do
24:     for j = i + 1 to num_student do
25:         // 若最大相似度值大于阈值 t，加入抄袭对
26:         if funMax(TDSA[i][j][]) ≥ t then
27:             add the pair ⟨i, j⟩ into PP;
28:         end if
29:     end for
30: end for
```

第一阶段。本抄袭检测系统从 SUT 识别测试中的方法M_1，M_2，…，M_n，并将它们存储在方法列表 MUT 中。对于每个测试代码 TC_i，本抄袭检测系统将其折射为一组测试片段 TF_i。在每个迭代（第 4-9 行）中，我们希望找到 TC_i 中每 M_j 的所有相关测试语句。这些与 M_j 相关的提取测试语句构成所谓的测试片段 $TF_{i,j}$。

第二阶段。本抄袭检测系统使用三维相似性数组 $TDSA$ 来记录每个 M_k 上成对学生的测试片段相似性值。由于时间和测试技能有限，学生只能测试 MUT 的某些部分。也就是说，一些测试片段可能为空值。给定两个非空测试代码片段 TF_{ik} 和 TF_{jk}，本抄袭检测系统基于 $funSim(TF_{i,k}, TF_{j,k})$ 计算相似度，并将该值放入 $TDSA[i][j]$。否则，本抄袭检测系统将默认相似性值 0 分配给 TF_{ik} 和 TF_{jk}。

第三阶段。本抄袭检测系统使用阈值分析来检测抄袭对。从直觉上看，相似度越高，就越有可能被认为是抄袭对。抄袭行为的认定与抄袭内容的大小以及抄袭位置的多少一般没有因果关系。因此，相似度最大的一对可以用于抄袭判断。$TDSA[i][j][]$ 记录 i 和 j 之间的相似性值。如果最大相似性，即 i 和 j 之间 $funMaxTDSA[i][j][]$ 的返回值大于 t，则 $\langle ij \rangle$ 对应为抄袭。否则，视为原创。在分析了所有测试码对后，完成抄袭检测过程。

本系统算法创新点主要在于设计了双向静态程序切片算法 [37]。假设我们要从 TC_1 提取片段，静态切片对程序依赖图 ($PDG = \langle N, E \rangle$) 有效。通常，节点 N 对应于可执行语句，边 E 对应于节点之间的依赖（即数据依赖和控制依赖）。边 $s_i \rightarrow s_j$ 意味着 s_j 依赖于 s_i。然后，其使用的变量 $USE = r1$ 和定义的变量 $DEF = d3$ 分别提供给随后的向后静态切片（BSS）和向前静态切片（FSS）。一旦 BSS 或 FSS 完成，双向静态切片停止并输出 BSS 结果和 FSS 结果。

3.5.2　待测方法提取算法设计

在单元测试中，测试用例设计适用于测试代码的所有单元。众所周知，每个方法/函数都被视为 Java 代码。因此，分析待测代码的前提是先分析测试代码。对于待测程序中的每个方法，待测方法提取器提取一些关键信息（例如 CN), 此提取的信息作为方法签名来表示，如以下公式 3.1所示。此外我们还进一步通过 Hash 函数构建方法签名，获取每个方法的唯一标识（MID)，这有助于后续流程。

$$\begin{gathered} \underbrace{CN_k(MN_k(PT_{(k,1)}, \cdots, PT_{(k,n)}) : RT_k;)}_{\text{signature of method } m_k} \\ CN : \text{class name};\quad MN : \text{method name}; \\ PT : \text{parameter types};\quad RT : \text{return type}; \end{gathered} \tag{3.1}$$

3.5.3　测试片段提取算法设计

直接计算两段未处理测试代码 TC_1 和 TC_2 的相似度值，并进行判断是否抄袭，这种做法是不合适的，因为它们的测试区分目标（M_1 和 M_2，分别在 TC_1

和 TC_2 中）；而测试代码 TC_1 和 TC_2 是非标准的（M_1 和 M_2 属于同一个测试用例）。此外，抄袭者可以添加一系列非相关语句进行混淆检测器。测试片段提取器通过引入双向静态切片技术，删除一组不相关的语句并提取有效的测试片段。测试片段提取器必须解决的另一个问题是标识实际测试目标。测试片段提取器去解析关键信息就像待测方法提取器，然后计算 MID。测试片段提取器通过判断是否存在的测试片段锚定标识测试目标 MID 是否存在，如使用从 TC_1 中提取的测试 M_1 的 TF_1。

对于 JUnit[18]framework，一个设计良好的测试应该满足但不限于两个规则：（1）命名约定：对于类 “C”，其对应的测试类的名称应该是大写的 “CTest” 或 “TestC”, 对于方法 “m”, 其对应的测试方法的名称应该是 lowerCamelCase[38] 中的 “testM” 或 “mTest”。（2） 单元测试粒度：每个测试用例应该只测试一个正在测试的方法，不应该将多个不相关的测试组合到一个测试用例中。此外，单元测试通常由四个基本部分组成，即 TC^i, TC^e, TC^a 和 TC^C。在设计良好的测试代码中，不难提取测试语句锚定测试方法。然而，初级测试人员（如学生）编写的测试代码总是远远不够好，特别是在高压力的考试或竞赛中。此外，许多测试代码可能不完整。MUT 中的某些方法不是有意或无意测试的；有些测试会丢失断言，等等。对于失败或崩溃的测试，提取测试片段仍然是一些挑战。静态切片首先由 Weiser [39] 提出，用来选择直接或间接影响语句中变量值的所有语句，即所谓的向后静态切片。随后，Horwitz [40] 等人提出了一种前向静态切片的方法来识别直接或间接受语句中变量值影响的语句。BSS 和 FSS 都依托于程序的依赖性（控制依赖性和数据依赖性）分析来从原始程序中提取一些代码语句。

受切片成功案例的启发，我们将双向（向后和向前）静态切片技术引入到测试片段提取中，简称 BSS-TFE。这是本系统的一大创新点。以下概述了 BSS-TFE 的详细信息。它将测试代码 TC_i 和测试下的方法 M_j 作为输入，并最终输出 TC_i 中的测试片段 TF_{ij}，该测试片段是为测试 M_j 而编码的。在 BSS-TFE 中，每个调用 MUT_j 的执行语句将被选为切片的关键点。在测试中，在调用 M_j 之前，它需要设置初始状态（例如，对象实例化）并准备基本参数。因此，TF_{ij} 应该包含用于初始状态设置和参数准备的语句, 条件向后静态切片可以满足需求。此外，在调用 M_j 之后，测试需要检查测试结果并执行任何必要的清理。因此，TF_{ij} 应该包含用于结果检查和资源清理的语句。使用条件 $\langle es_k, DEF\rangle$ 的前向静态切片可以满足需求。因此，es_k 上的双向静态切片的结果（即 $BSSR$ 和 $FSSR$）包含用于测试 M_j 的语句。分析完所有执行语句后，BSS-TFE 输出 TF_{ij}，算法完成。

下面以一个具体的例子来介绍本系统的核心技术，即本系统一大创新点：运用静态双向程序切片技术去进行测试代码片段提取。其能够提取有效测试片段，

每一个片段都是一个最小粒度单位的测试，摒弃了大量冗余信息，从而提高测试代码相似性检测的效果。本文用图 3.7来说明静态双向切片的过程，其中假设已有待测方法为 M_1，静态切片在程序依赖图（$PDG =< N, E >$）上进行，图 3.7中每个节点 N 对应于可执行语句，边 E 对应于依赖项（即数据依赖以及控制依赖）。一条边 $s_i \rightarrow s_j$ 意味着 s_j 依赖于 s_i。图 3.7(a) 显示了测试代码的 PDG，它包含 16 个节点和 22 条边。假设，只有节点 s_{13} 调用待测方法 M_1。因此，如图 3.7(b) 所示，选择 s_{13}（灰色节点）作为关键点。然后，同步进行反向静态切片（BSS）和正向静态切片（FSS）。BSS 提取 s_{13} 直接或间接依赖的语句，FSS 提取直接或间接依赖 s_{13} 的语句。一旦 BSS 和 FSS 完成，双向静态切片停止并输出 BSS 结果 $BSSR = s_2 \rightarrow s_5, s_8 \rightarrow s_{12}$（即图 3.7(c) 中的红色节点）和 FSS 结果 $FSSR = s_{14}$(即图 3.7(d) 中的蓝色节点)。最后提取出来的测试片段即图 3.7(d) 中的所有着色节点。

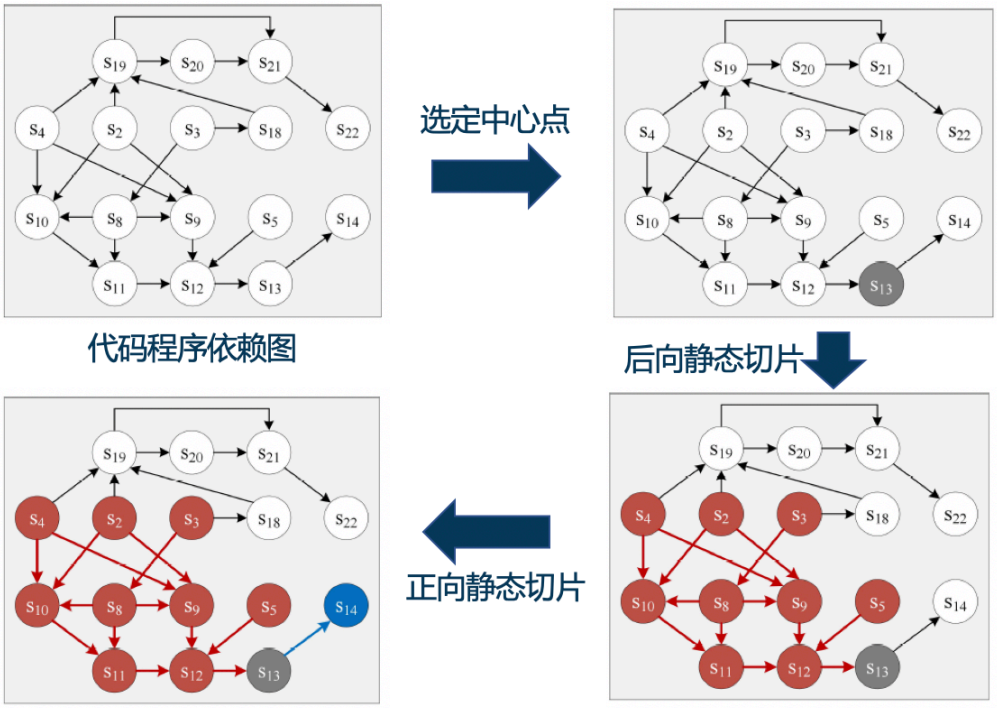


图 3.7: 静态双向程序切片示意图

3.5.4　相似度计算算法设计

相似度计算算法是通过集成一种已被证明是很有前途的抄袭检测高级工具 FuzzyWuzzy，来实现的。现有研究中的字符串相似性计算方法，在 FuzzyWuzzy 中，字符串 str1 和字符串 str2 之间的相似性通过公式 3.2计算，其中 |str1| 和 |str2|

分别对应于 str1 和 str2 中的字符数量，且 $Match^{char}$ 对应所有字符匹配的大小。一套更精细的相似性粒度，通过测量成对测试碎片，成为一个很好的指标，用于随后的抄袭检测。

$$sim(file_1, file_2) = 1 - \frac{2.0 * Match^{char}}{|file_1|^{char} + |file_2|^{char}} \tag{3.2}$$

3.5.5　抄袭分析算法设计

根据相似度计算仪获得的相似性，抄袭分析器到通过使用阈值分析方法进行抄袭对分析。直觉上相似度越高，该对被认为是抄袭对的可能性越大。抄袭的鉴定有与抄袭内容的大小以及抄袭位置没有因果关系。因此，可以利用具有最大相似度值的对进行抄袭判断。最大相似度为大于阈值将被视为抄袭，如，TF_1 和 TF_2 完全相同（相似度是 100%）。因此，我们可以得出结论 TF_1 和 TF_2 最有可能是抄袭。

3.6　本章小结

本章节主要对系统从涉众、功能性需求和非功能性需求进行分析并形成图表。在系统用例图和系统整体架构图的基础上，对整个系统架构从逻辑视图、进程视图、开发视图和物理视图进行多角度拆分与描述；接下来介绍了本系统中使用到的数据模型设计，重点描述抄袭检测数据模型的设计思路与字段含义。最后介绍了本系统涉及到的关键算法的设计。

第四章　系统详细设计与实现

4.1　待测方法提取模块的详细设计与实现

待测方法提取模块负责考试/比赛中待测程序的方法提取。待测方法提取模块在服务启动时向平台端发送请求，请求获得考试/比赛的项目题目代码，处理由平台端发出的考试/比赛的项目题目代码的 url 地址，并将 url 信息存储在 Redis 缓存中。然后该模块根据获取的 url 批量对项目题目代码进行下载及解压，接着对代码进行待测方法的提取。为了将模块获取的数据进行存储，本模块接入了 MySQL 作为待测方法提取底层实现。为了提高系统数据安全性，用户对集群进行数据读写前均由待测方法提取进行权限验证，并将用户凭证缓存至 Redis 中。

4.1.1　待测方法提取模块详细设计

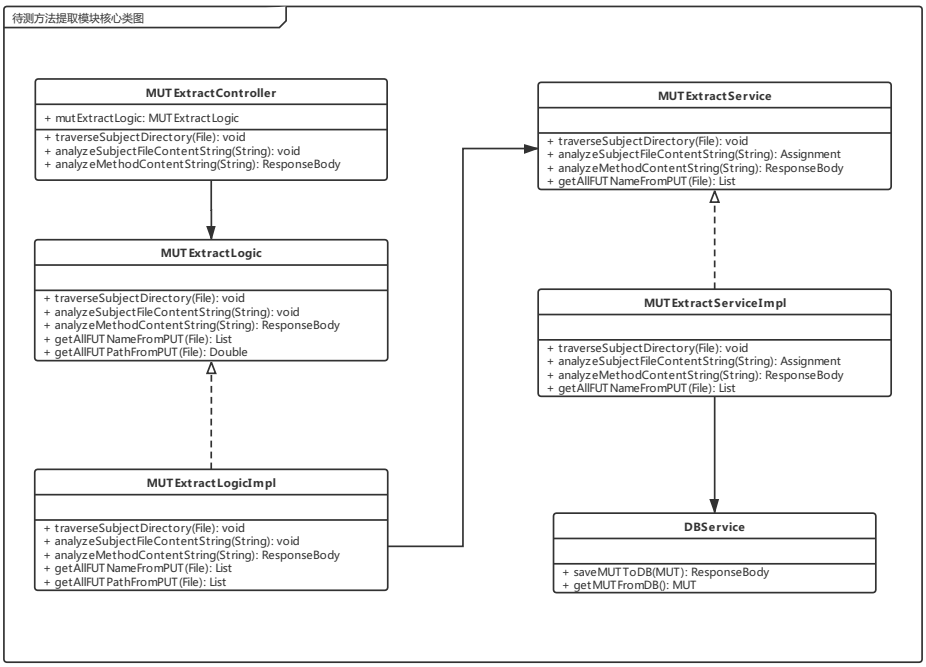


图 4.1: 待测方法提取模块核心类图

待测方法提取模块的核心类设计如图 4.1所示。其中，MUTExtractController 是本模块对外提供服务的控制器，持有对 MUTExtractLogic 的引用。analyzeSubjectContentString(String) 是分析提取出来的项目代码字符串关键逻辑方法，参数为 String 数据类型的变量，且内部存储了提取出来的项目代码字符串，最终得到分析结果。traverseSubjectDirectory(File) 为遍历项目路径方法，参数为 File 对象

数据类型变量，对象内部存储了项目代码所在文件路径，最终定位到本次提取的待测方法所在文件位置；analyzeMethodContentString(String) 中，参数为 String 类型，即提取出来的待测方法字符串，在本方法中再进行具体分析。

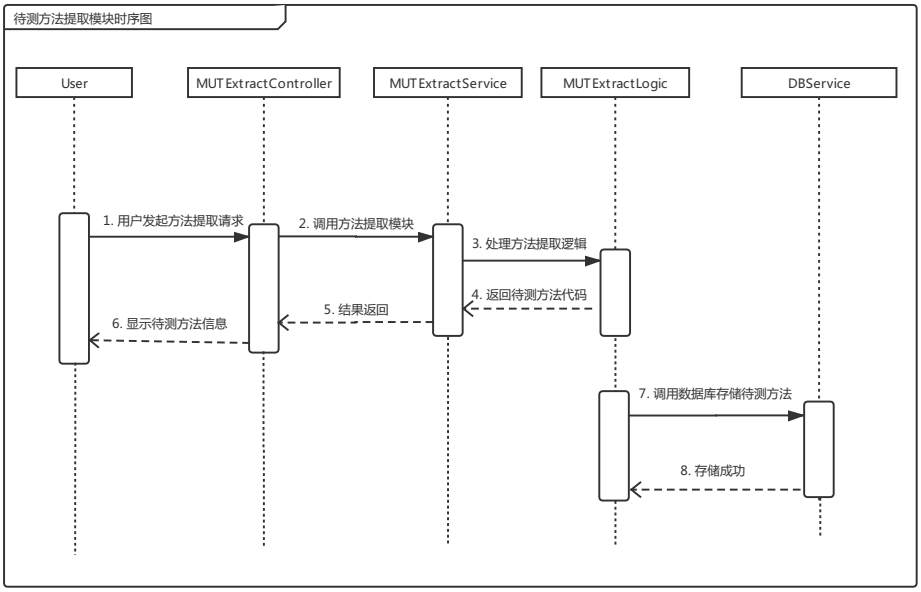


图 4.2: 待测方法提取模块时序图

MUTExtractLogic 为业务逻辑接口。前三个重复方法为 MUTExtractController 通过对 MUTExtractLogic 持有的引用进行的调用；getAllFUTNameFromPUT(File) 为具体获取待测程序文件名称方法，从待测程序中获取其各个代码文件名称，参数为 File 类型变量，返回值为 List 类型变量；getAllFUTPathFromPUT(File) 为具体获取待测程序文件路径方法，从待测程序中获取其各个代码文件路径，参数为 File 类型变量，返回值为 Double 类型变量。

MUTExtractLogicImpl 是对 MUTExtractLogic 接口中方法的实现，MUTExtractService 为服务端接口，且 MUTExtractLogicImpl 依赖该接口实现功能；MUTExtractServiceImpl 是对 MUTExtractService 接口中方法的实现。

DBService 是实现本模块与数据库的交互操作类。saveMUTToDB(MUT) 将提取得到待测方法内容存储到数据库，参数为 MUT 类型，内部为提取得到的待测方法信息；getMUTFromDB() 为从数据库获取待测方法内容信息，返回结果为 MUT 类型，即获取的待测方法内容信息。

如图 4.2所示为待测方法提取模块的时序图，其展现了此模块各类间的逻辑交互，以及具体的适配处理过程。用户抵达系统后进行请求，通过触发 Angular2 HTML 模版绑定操作事件以发送请求，MUTExtractorController 调用 MUTExtractorService 完成请求获得待测程序的 url 地址后，根据 url 批量对待测程序进行下

```
private static void analyzeSrcFileContentString(String srcFileContentStr,String subject) {
    //省略变量初始化代码
    List<String> words = Arrays.asList(srcFileContentStr.split(" "));
    for (int i = 0; i < words.size(); i++) {
        String word = words.get(i);
        //没有内部类
        if (!hasInnerClass) {
            if (word.contains("{")) {
                for (int j = 0; j < word.length(); j++) {
                    if (word.charAt(j) == '{') {
                        leftBracketsNum++;}}}
            if (word.contains("}")) {
                // 省略分析后括号代码，与前括号同理
            }
            if (word.equals("class")) {
                // 类名与"{"未分开, 如: class Test{ ... }
                if (words.get(i + 1).contains("{")) {
                    newClassName = words.get(i + 1).split("\\{")[0];
                    words.set(i + 1, "{");
                //类名中包含"<"或">"
                } else if(words.get(i + 1).contains("<")||
                  words.get(i + 1).contains(">")){
                    newClassName = words.get(i + 1).split("<")[0];
                }else{
                    newClassName = words.get(++i);
                }
            }
            // 类结束
            if (leftBracketsNum == rightBracketsNum) {
                //省略类结束处理代码，同理
            } else {
            //省略有内部类处理逻辑，与无内部类同理
}}}
```

图 4.3: 待测方法提取模块关键代码示例图

载解压，接着分析程序，提取待测方法。与此同时，Controller 调用 DBService 进行存储待测方法，以便后续模块使用。

4.1.2　待测方法提取模块具体实现

如图 4.3所示为待测方法提取模块功能的主要逻辑代码，涉及待测方法定位、待测方法 String 字符串分析与各种不同类方法情况下的字符串提取。为不影响阅读，此处省略了持久化层调用与异常处理等非核心代码，仅展示正常路径下 analyzeSrcFileContentString 中主要核心代码。其中参数 srcFileContentStr 指代的是从比赛/考试题目项目代码文件中提取出来的代码 String 字符串，包括代码中的各种待测方法，其中包括了其私有方法等，并且在该 analyzeSrcFileContentString 方法中完成对是否有内部类 innerClass 的处理，以及具体大括号内部待测方法的精准定位。数据结构 words 处理通过空格进行分割获得的代码字符串 List，通过对 words 进行分析，处理其中具体有几层方法结构等，并对结果进行提取存储。其中 equals（class）为对待测方法特殊标识符的精准定位，可以通过这个定位来提取待测方法到指定位置。

4.2　测试片段提取模块的详细设计与实现

测试片段提取是本系统中的关键基础服务，为后面的测试片段相似度分析以及抄袭分析等相关的业务服务提供支持。测试片段提取模块详细架构设计如下所示。测试片段提取在服务启动时通过获取前端用户配置信息选取要提取的学生测试代码信息，先与上一个待测方法提取模块同理，由 url 地址获取学生提交的测试代码后，基于静态双向程序切片技术，对测试代码进行片段化，并持久化存储。

4.2.1　测试片段提取模块详细设计

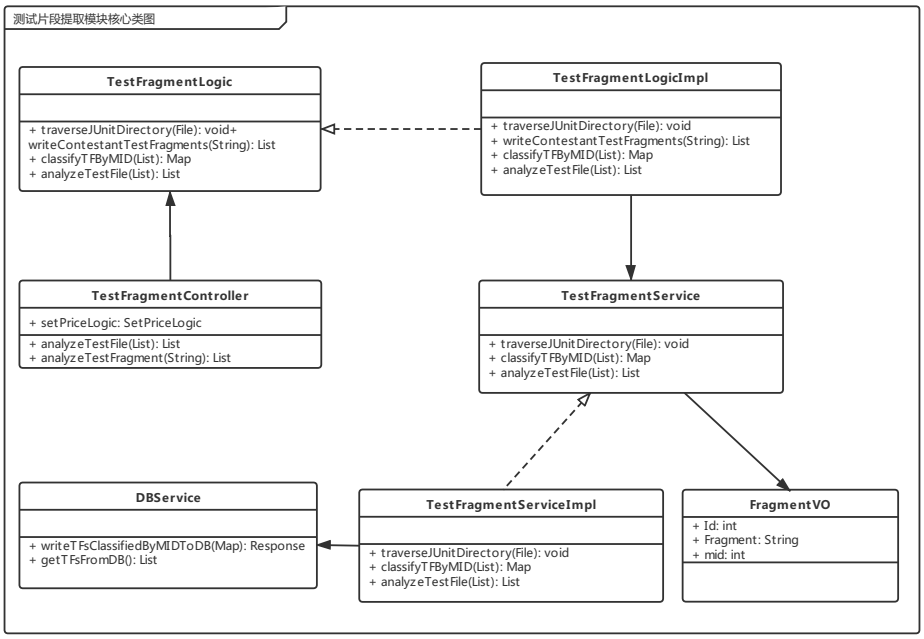


图 4.4: 测试片段提取模块核心类图

如图 4.4所示是测试片段提取核心类图设计，TestFragmentController 是本服务对外提供服务的控制器，主要提供测试片段提取服务。TestFragmentController 通过调用外部提供的接口实现基于程序切片的测试片段提取。TestFragmentController 还通过 traverseJUnitDirectory(File) 实现了遍历测试代码文件路径。

TestFragmentLogicImpl 是对 TestFragmentLogic 接口中方法的实现，TestFragmentService 为服务端接口，且 TestFragmentLogicImpl 依赖该接口实现测试片段提取功能；TestFragmentServiceImpl 是对 TestFragmentService 接口中方法的实现。FragmentVO 是存储测试片段信息的数据结构类，内部由 id，fragment，mid 等成员变量，来唯一标识一个测试片段。

DBService 是实现本模块与数据库的交互操作类。writeTFsClassifiedByMID-

ToDB(Map) 将按待测方法 id，即 mid，分类的测试片段内容存储到数据库，参数为 Map 类型，内部为按 mid 分类的测试片段信息；getTFsFromDB() 为从数据库获取测试片段内容信息，返回结果为 List 类型，即获取的测试片段内容信息。

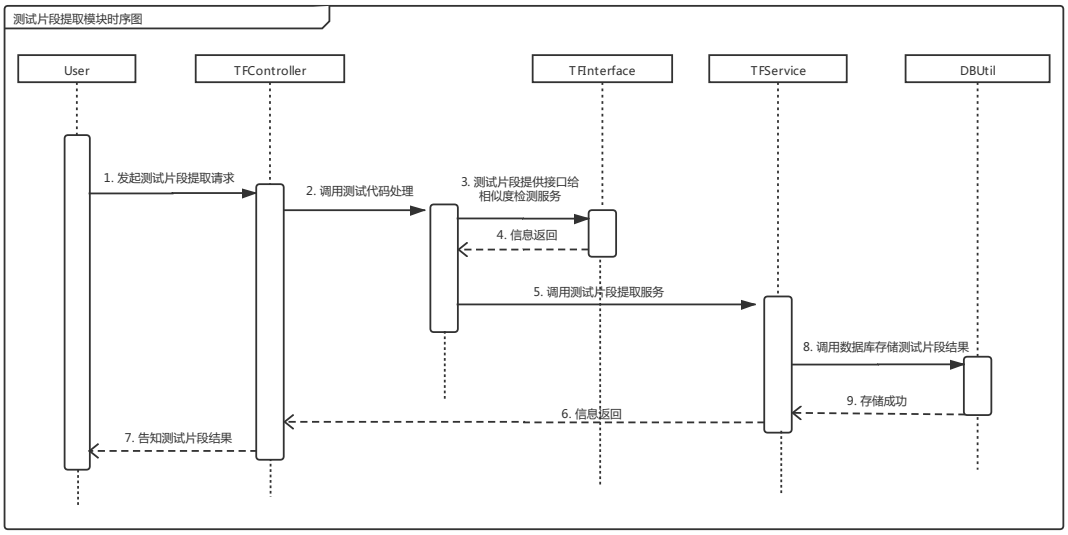


图 4.5: 测试片段提取模块时序图

如图 4.5所示是测试片段提取提供测试片段提取功能的服务调用顺序。在接受到测试片段提取请求时，前端 User Interface 向后端发送 HTTP GET 请求以获取测试代码，此时触发 InvokeService 服务完成用户与测试任务之间的匹配工作；在适配成功后，TestFragmentController 调用分配服务模块 TestFragmentService 中的实现方法，同时通过 assignDao 返回的结果以引入测试代码片段数据结构。在测试代码片段提取完成后，TestFragmentService 调用 DBUtil 进行分配数据存储，TestFragmentController 将提取结果返回至前端服务，用户界面中与属性对应的 HTML 模版将展示自动测试代码相关信息。

4.2.2　测试片段提取模块具体实现

如 4.6所示为测试片段提取中关键功能的代码片段，主要的提取逻辑在 analyzeTestFileContentString 方法中，为了展示重点，删除了部分非重要代码，图中主要展示了关键核心逻辑代码，包括对测试代码前后处理方法上注解“@Before”，“@After”等的识别提取，以及对测试方法代码上注解“@Test”等进行识别提取等。此处简化具体识别提取逻辑，仅展示 testCaseString.contains 与 extractTestCaseNameFromTestCaseString 方法中的部分核心代码来展示如何进行提取。每个提取出来的测试片段均可得到一个 String 值，也就是该测试片段存储于 MySQL

```
private static List<InvokeMethodModel> analyzeTestFileContentString(List<MUTModel> mutMode
lList,String testFileContentString) {
       List<String> testCaseStringList = extractMethodWithJUnitAnnotationFromString(testFi
leContentString);

      for (String testCaseString : testCaseStringList) {

         if (testCaseString.contains("@Before")|| testCaseString.contains("@org.junit.Be
         fore")|| testCaseString.contains("@After")|| testCaseString.contains("@org.juni
         t.After"))
         {
           continue;
         }
         if (!testCaseString.contains("@Test")&& !testCaseString.contains("org.junit.Tes
         t")){
               continue;
             }
         String testCaseName = extractTestCaseNameFromTestCaseString(testCaseString);

         //省略处理try-catch块
         List<String> sentenceAmongTestCaseStringList = getSentenceEndedWithSemicolonFrom
         String(testCaseString);
         //去除形如"new TernaryTree"这种创建对象等语句，只提取跳用被测方法的语句
         List<String> sentenceContainsInvokeMethodList = getSentenceContainsInvokeMethod
         (sentenceAmongTestCaseStringList);
         int sentenceContainsInvokeMethodListSize = sentenceContainsInvokeMethodList.size
         ();
         itInvokeMethod(sentenceContainsInvokeMethodList, testCaseName, invokeMethodMode
         lList);

         return incokeMethodModelList;

}
```

图 4.6: 测试片段提取模块关键代码示例图

中具体值。用户通过该具体值可从系统中查看或下载诸如测试具体片段信息等文件。analyzeTestFileContentString 方法中通过调用 getSentenceEndedWithSemicolonFromString(testCaseString) 获得具体的 Sentence 值并放到 List 中，将 Sentence 列表转化为输出流后写入 sentenceAmongTestCaseStringList 中，用户客户端可通过配置界面中的信息来判断学生提交测试代码文件中测试片段是否提取成功并保存。把 Java 测试片段对象存入 MySQL 先需要将 Java 测试片段对象转成 String 对象，通过哈希生成全局唯一 Id 作为临时测试代码片段编号，通过 StringMapper 将对象写入 String 对象，再调用传输至 MySQL 的接口去存储到数据库，最后将刚刚生成的临时对象删除。

4.3　相似度计算模块的详细设计与实现

4.3.1　相似度计算模块详细设计

相似度计算是本系统最核心的服务之一，为后面进行抄袭检测抄袭分析服务、检测报告生成、生成相似度矩阵、检测概要结果显示等重要功能提供基础。相似度计算在服务启动时从配置接口获取要分析的测试代码与端口信息。相似

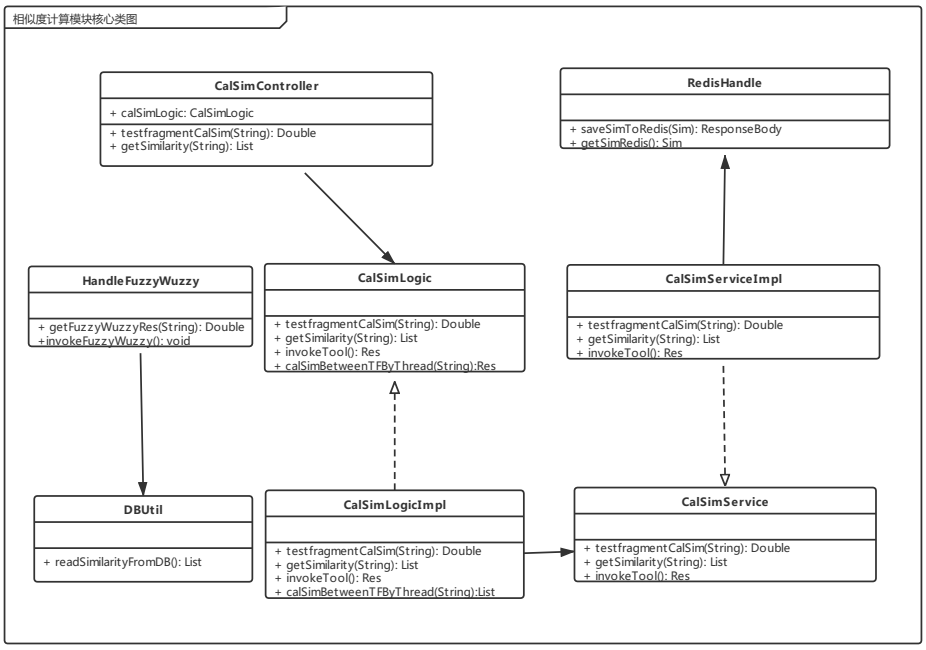


图 4.7: 相似度计算模块核心类图

度数据是本系统中最核心的数据，相似度数据会被存储至数据层中，数据层在一定程度上可看成是只有增操作和查操作的分布式数据库，相似度的每次变更都被记录在数据层中，根据学生编号即可获得此学生被检测代码相似度在任一时刻的状态。从前面模块获取片段，然后两两间进行基于编辑距离的字符串匹配，计算相似度。

如图 4.7所示是相似度计算核心类图设计。其中，CalSimController 是本模块对外提供服务的控制器，持有对 CalSimLogic 的引用。testFragmentCalSim(String) 为进行相似度计算的方法，测试任务参数为 String 类型的变量，相似度计算返回值为 Double 类型；getSimilarity(String) 为获取相似度计算结果的方法，最终结果信息返回值为 Double 对象类型。CalSimLogic 为业务逻辑接口，前两个重复方法为 CalSimController 通过对 CalSimLogic 持有的引用进行的调用。invoke-FuzzyWuzzy() 为获取调用 FuzzyWuzzy 进行字符串匹配的方法，通知信息返回值为 String 类型；getFuzzyWuzzyRes(String) 为获取 FuzzyWuzzy 返回值的方法，返回值为 String 类型。CalSimLogicImpl 为对 CalSimLogic 接口中方法的实现，CalSimService 为服务端接口，CalSimLogicImpl 依赖本接口实现功能。

如图 4.8所示为相似度计算中相似度计算功能的业务调用序图，查询相似度以及更新相似度等功能均与之类似。CalSimController 在接受到相似度计算请求后，调用 CalSimLogic 中创建相似度接口。CalSimLogic 对相似度参数进行必要检验后，调用 CalSimService 进行相似度计算，再由 DBService 将相似度计算结

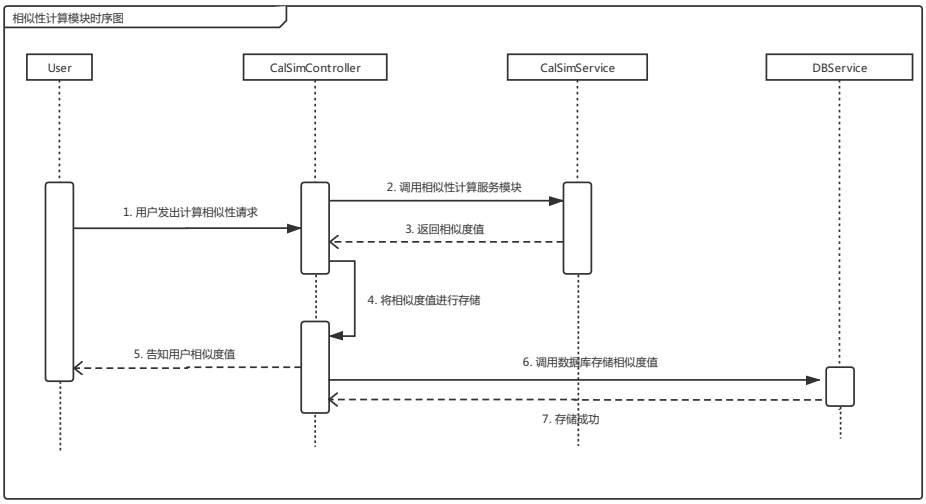


图 4.8: 相似度计算模块时序图

果存入 MySQL 中并在 Redis 作缓存。整个流程为，用户发出计算相似度请求，接着调用相似度计算服务模块，通过 CalSimService 处理后返回相似度值，再调用数据库处理部分进行存储，最终将相似度值返回。

4.3.2　相似度计算模块具体实现

如 4.9所示即为相似度计算功能模块核心代码，为不影响论文整体展示，此处省略参数检查与对象转换代码，仅展示核心代码。该 fuzzyParticalRatio 方法是本系统调用 FuzzyWuzzy 用于进行相似度匹配的核心方法，主要是对字符串进行了部分匹配，并且加上对 String 超长和执行超时的处理。初始化 simValue 值是整个相似度计算流程的前提，后面进行 FuzzyWuzzy 调用处理是系统中最为关键的功能之一。代码中 “s1.length()+s2.length()>=47000” 部分会先对参数中传来的 String 类型字符串 s1、s2 进 行字符串长度检验，有效性检验。通过 Callable 去生成运用线程处理技术将其 FuzzySearch.partialRatio 部分匹配比对字符串的操作作为耗时操作进行执行，这样再通过 Future 及 submit 去进行异步监听执行时间，任务处理超时时间设定为 60 秒，即一分钟。在生成具体相似度值 simValue 后，该 fuzzyParticalRatio 方法会将 simValue 相似度值返回出去并 shutdown 关闭线程，并且后面会调用数据层方法将 simValue 相似度值存储进 MySQL。这些 simValue 相似度值即为两个测试代码片段之间通过编辑距离字符串比对获得的相似度。通过一系列赋值操作，最终在存储时，会给 simValue 值配置上某次考试项目名称信息、两个学生对信息，最终转化为 SimValueVO，再调用 CalSim-Service 对 SimValueVO 对象进行存储。

```
public static int fuzzyPartialRatio(String s1, String s2) {
    int simValue = 50;
    final ExecutorService exec = Executors.newFixedThreadPool(1);
    //String 超长时返回默认值
    if(s1.length()+s2.length()>=47000){
        return simValue;
    }
    Callable<Integer> call = new Callable<Integer>() {
        public Integer call() throws Exception {
            //开始执行耗时操作
            return FuzzySearch.partialRatio(s1, s2);
        }
    };
    try {
        Future<Integer> future = exec.submit(call);
        //任务处理超时时间设为 60 秒
        Integer res = future.get(1000 * 60,
TimeUnit.MILLISECONDS);
        simValue = res;
    } catch (TimeoutException ex) {
        System.out.println("处理超时啦....");
        ex.printStackTrace();
    } catch (Exception e) {
        System.err.println("处理失败");
        e.printStackTrace();
    }
    exec.shutdown();
    return simValue;
}
```

图 4.9: 相似度计算模块关键代码示例图

4.4　抄袭分析模块的详细设计与实现

抄袭分析模块向教师等用户提供根据计算出的学生测试代码之间相似度等，基于阈值分析去进行抄袭分析的功能。抄袭分析模块服务在前期待测方法提取，测试片段提取，相似度计算等步骤都已完成后被启动，从相似度计算模块获取本次检测后获取各学生各测试代码片段之间相似度，处理由展示端发出的关于基于阈值去划分是否抄袭相关的功能，例如在前端展示抄袭检测结果概要信息等。

抄袭分析模块根据相似度值及设定阈值去分析被检测学生对之间是否存在抄袭，并将结果存至 MySQL，将抄袭分析结果进行持久化存储。同时本系统结合缓存技术，对数据进行分批存储。这样的设计，使得整个系统的数据更加便于追溯和管理。对于学生提交的测试代码，前面模块进行片段化后，会产生多个测试代码片段，两两之间比对会产生多个相似度值，选取其中相似度最大值进行抄袭分析。软件抄袭分析可结合本系统其他配套模块的结果去结合本模块逻辑生成最终分析结果，分析结果中包含被检测学生信息，被检测学生 ID 编号，代码 Hash 值等关键信息。本系统结合这些信息去确定被检测学生信息以及其中抄袭学生对信息，以便在用户对结果进行对照查看和验证。

4.4.1　抄袭分析模块详细设计

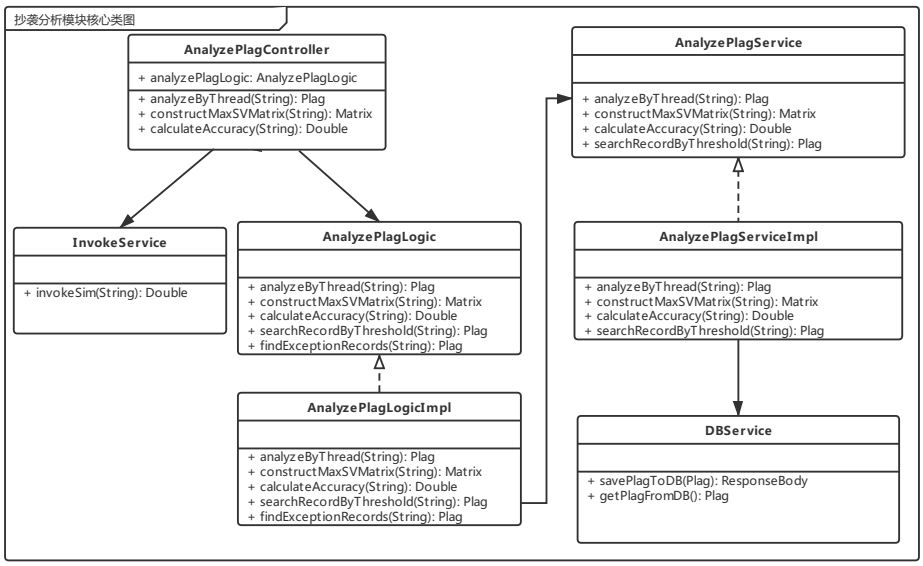


图 4.10: 抄袭分析模块核心类图

如图 4.10所示为抄袭分析模块核心类图。其中，AnalyzePlagController 是本模块对外提供服务的控制器，持有对 AnalyzePlagLogic 的引用。analyzeByThread (String) 为根据设定的阈值进行划分的方法，提交的测试参数变量为 String 类型，返回值为 Boolean 类型；caluculateAccuracy(String) 为计算抄袭分析结果的准确率的方法，测务参数变量为 String 对象数据类型，List 类型返回值为进行准确率列表。

如图 4.11所示是抄袭分析模块提供抄袭分析功能的服务调用时序图。AnalyzePlagController 在收到进行抄袭分析的请求后，调用 AnalyzePlagLogic 中进行抄袭分析的接口。系统上传的 VO 包含相似度值等信息，这代表着对学生对测试片段代码进行相似度计算的结果信息，其他模块先前已经传至数据层中。通过相似度值，AnalyzePlagLogic 调用 AnalyzePlagLogic 中的逻辑方法可以得到分析结果对象，再由 AnalyzePlagLogic 将分析结果对象与 VO 中的信息进行对比，若符合则通过 AnalyzePlagService 与 DBService 将抄袭分析解果返回前端并存入数据层，若两者不相符合则停止流程。

4.4.2　抄袭分析模块具体实现

如图 4.12所示为抄袭分析模块核心代码，本代码逻辑根据前面相似度度量模块得到最大相似度去进行抄袭分析，此段代码摘自 AnalyzePlagLogic 的实现类

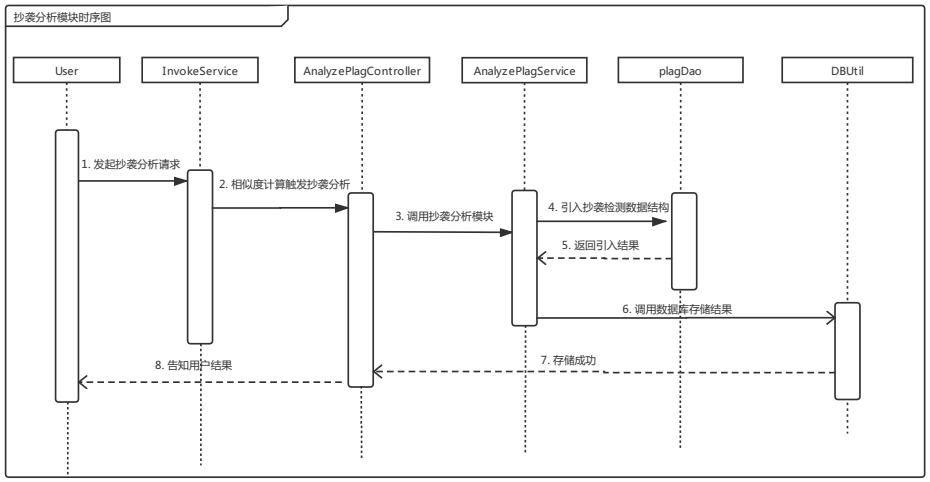


图 4.11: 抄袭分析模块时序图

AnalyzePlagLogicImpl。本 constructMaxSVMatrixWithLabel 方法，其中参数 simMatrixFilePath 和 maxSVMatrixFilePath 会先被校验其参数有效性，若缺少该路径有误或缺少关键组件则直接异常返回。通过 FileUtil 中的 writeMatrixToCSVFile 方法，可以将最大相似度等信息存储到 csv 文件中，其中包含检测学生对 ID 信息，最大相似度信息及是否抄袭标签信息。然后通过 reader.readRecord() 去将相似度信息进行获取以供分析，rowContents 用来存储获取的 Value 信息，之后可以根据设置的阈值 Threshold 去划分是否抄袭。接下来验证抄袭结果并将结果存储到 PlagAnalysisVO 中，若结果与预期相符合则通过 SoftwareService 调用 DataDao 将 PlaAnalysisVO 存入数据层中并反馈给前端，若不相符合那么就异常退出。

4.5　检测报告生成模块详细设计与实现

报告生成模块为用户方提供报告生成、报告下载等功能。报告生成模块在服务启动时向其他服务请求写入抄袭分析结果等信息，处理由展示端发出的检测报告相关请求，例如检测报告生成与下载检测报告等。报告生成模块将检测报告文件与检测报告路径存至文件系统，将检测报告路径信息存储到服务器端文件系统。本系统将具体报告 PDF 文件存储在云服务器上，而数据库中仅仅存储报告文件路径信息。报告 PDF 主要是由本模块逻辑代码自动生成，PDF 中包含检测信息，检测时间，报告 ID 值等关键信息。用户在查看报告时，可以根据这些信息对检测有一个大致的了解，并可根据信息去查找需要查看和验证的内容部分。

```
private static void constructMaxSVMatrixWithLabel(String simMatrixFilePath, String maxSVMatrixFile-
Path) {
    CsvReader reader = null;
    try {
        reader = new CsvReader(simMatrixFilePath, ',', Charset.forName("UTF-8"));
        File targetFile = new File(maxSVMatrixFilePath);
        if (targetFile.exists()) {
            targetFile.delete();
        }
        String[] headerArray =  {"<CID1, CID2>", "MAX_Sim_Value", "Class(0 or 1)"};
        FileUtil.writeMatrixToCSVFile(targetFile, headerArray, null, true);
        List<String[]> contentBeLabeledList = new ArrayList<>();
        while(reader.readRecord()) {
            count++;
            if (count == 1) continue;
            String[] rowContents = reader.getValues();
            String cidString = rowContents[0];
            int length = rowContents.length;
            String labelString = rowContents[length - 1];
            int maxSimValue = 0;
            for (int index = 1; index < length - 1; index++) {
                int simValue = Integer.parseInt(rowContents[index]);
                if (simValue > maxSimValue) {
                        maxSimValue = simValue;
                    }
                }
                contentBeLabeledList.add(new String[] {cidString, maxSimValue + "", labelString});
                if (contentBeLabeledList.size() % 100 == 0) {
                    FileUtil.writeMatrixToCSVFile(targetFile, null, contentBeLabeledList, true);
                    contentBeLabeledList.clear();
                }
            }
        //省略exception处理
}
```

图 4.12: 抄袭分析模块关键代码示例图

4.5.1　检测报告生成模块详细设计

如图 4.13所示为检测报告生成模块核心类图。PDFGenerateController 为提供服务的控制器，主要提供抄袭检测报告生成功能。PDFGenerateLogic 负责处理核心逻辑，提供创建 PDF 文件、生成报告主体部分等功能。PDFGenerateLogicImpl 继承 PDFGenerateLogic 中创建 PDF 文件、生成报告主体部分等接口，实现对创建报告文件的具体逻辑。与上文提到的 AnalyzePlagService 类似，PDFGenerateService 依赖 PDFGenerateLogicImpl 来实现服务层检测报告生成功能。PDFContent 与 PDFContentVO 对象为同一结构在不用软件层级中不同体现形式。PDFGenerateServiceImpl 实现 PDFGenerateService 的具体接口，实现具体逻辑。DBService 则是实现该模块与数据层各种增删改查哦交互，主要包括将 PDF 存放路径存储到数据库，以及从数据库读取 PDF 文件存放路径等操作。

如图 4.14所示是检测报告生成模块提供抄袭检测报告生成功能的服务调用时序图。PDFGenerateController 在收到前端生成抄袭检测报告请求后，调用 PDFGenerateService 中生成抄袭检测报告文件服务接口。用户上传的 PDFContentVO 包含检测报告存放路径值等，这代表着检测报告 PDF 存储在服务器上的地址，用户先前已经传至 MySQL 中。对检测报告路径存储以及读取等操作，均由 PDF-

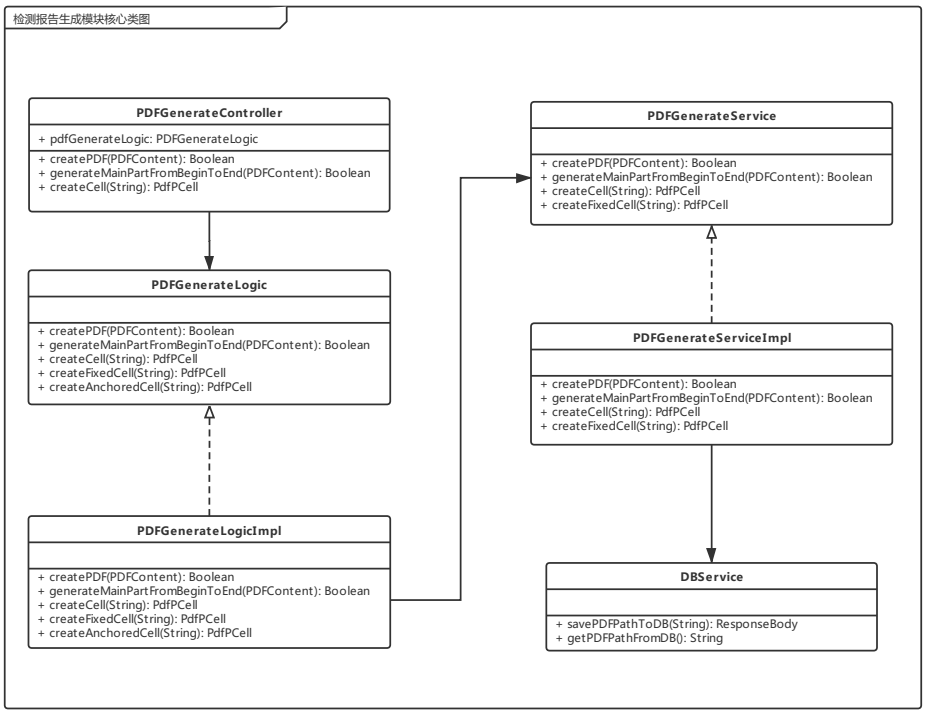


图 4.13: 检测报告生成模块核心类图

GenerateLogic 调用 DBService 中的存储和读取方法去得到 PDFContentPath 字符串，再由 PDFGenerateLogicc 将 PDFContentPathzi 字符串再向 PDFGenerateService 返回，PDFGenerateService 得到后再加结果向 PDFGenerateController 返回，最终由 PDFGenerateController 将待测报告生成信息返回给用户。

4.5.2　检测报告生成模块具体实现

如图 4.15所示为检测报告生成模块生成和存储检测报告 PDF 的核心代码，此段代码摘自 PDFGenerateLogic 的实现类 PDFGenerateLogicImpl。pdfContent 会先被校验其内容有效性，若缺少关键字段如检测结果值（即根据前面相似度计算得到的相似度值，通过抄袭分析模块得到的抄袭分析结果等）则直接异常返回。另一个参数 isAll 是 boolean 类型，通过此来分辨本次生成检测报告的是一次对全部学生检测还是对部分学生检测。

createPDF 方法的具体代码逻辑解释如下：首先通过 document.addTitle 方法，设置本检测报告的标题。接着通过 PdfPTable NoAndTime 去设置本次检测报告的编号和时间。接着通过 pdfContent.getSubject() 获得本次检测的待测项目，即比赛/考试题目。接着通过 getPlayers() 获得本次检测涉及的学生 Id 等信息。然后如果 isAll 为 true，则直接设置显示全部学生信息；否则设置显示通过 generatePayerStr 方法去生成的检测学生名单信息。接着通过 threStr 去设置显示本次

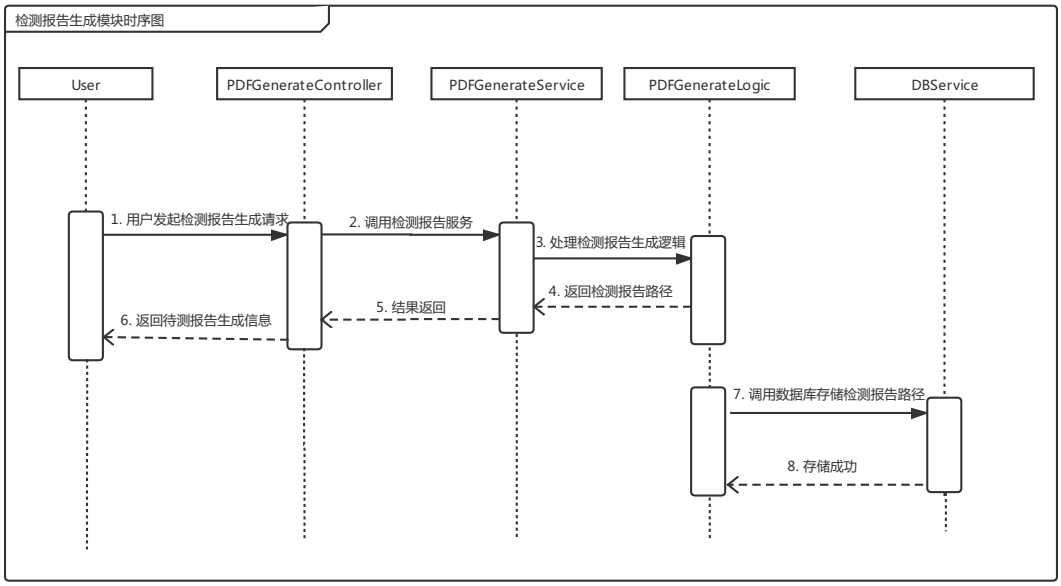


图 4.14: 检测报告生成模块时序图

检测设置的相似度阈值。接下来通过 result 去显示本次检测结果信息。最后通过 totalAndPlag 去显示共检测学生对信息以及其中抄袭学生对。

```
public boolean createPDF(PDFContent pdfContent,boolean isAll){
          //省略检测时间设置
          document.addTitle("测试脚本相似度检测报告");
          document.open();
          //省略字体设置
          Paragraph title = new Paragraph("测试脚本相似度检测报告",titleFont);
          title.setAlignment(1);
          document.add(title);
          document.add(new Paragraph("\n"));
          //省略格式设置
          PdfPTable NoAndTime = createTwoParallel("","No: "+dateID,Element.ALIGN_LEFT,"检测时
          间：",dateString,Element.ALIGN_RIGHT);
          document.add(NoAndTime);
          String subjectStr = pdfContent.getSubject();
          Paragraph subject = new Paragraph(titleChineseStrEnglish("题目：",subjectStr));
          List<Integer> playerIDs = pdfContent.getPlayers();
          String playerStr = "";
          Paragraph players;
          if(isAll){
              playerStr = playerIDs.size()+"";
              players = new Paragraph(titleChineseStrEnglish("检测选手范围：全部选手 共计：
          ",playerStr));
          }else {
              playerStr = generatePlayerStr(playerIDs);
              players = new Paragraph(titleChineseStrEnglish("检测选手范围：",playerStr));
          }
          String threStr = pdfContent.getThreshold()*100+"%";
          Paragraph threshold = new Paragraph(titleChineseStrEnglish("相似度阈值：",threStr));
          Paragraph result = new Paragraph("检测结果",FontChinese2);
          PdfPTable totalAndPlag = createTwoParallel("共检测选手对：
          ",""+totalPairs,Element.ALIGN_CENTER,"抄袭选手对",""+plagPairs,Element.ALIGN_CENTER);
          return true;
}
```

图 4.15: 检测报告生成模块关键代码示例图

4.6　系统实例展示

测试抄袭检测系统已集成入国内某知名在线软件测试训练平台 M 成为一个关键子系统，负责检测学生提交测试代码中的抄袭行为。每位在 M 平台上的注册者可以很容易地使用它。进入系统，首先可以查看考试信息以及对本次检测进行配置，图 4.16显示了检测配置界面。

“待测方法列表”视图中列出了所有关于“Tarjan”项目待测方法的信息（对应于图 4.16），首先可以看到方法 ID，即唯一标识待测方法的 ID。除此之外，用户还可以看到方法名，即待测方法名称；参数类型，即该待测方法中参数数据类型；类名，即该待测方法所属 Class 类名称；返回类型，即该待测方法返回值数据类型。当用户点击“详情”按钮后会显示该待测方法详细具体的生产代码。

“考试结果信息”视图则显示学生所取得考试成绩的所有信息以及提交的测试代码（对应于图 4.16）。教师用户可以根据这些信息（例如分数）选择她/他想要检测的学生。同样，在点击“详情”按钮后将显示详细的测试代码。

在检测配置界面，老师用户可以对要检测的学生对象进行配置，通过这些配置，可以个性化地选择部分想检测的学生进行检测，或者是全部学生进行检测。那么主要是通过点选前面的复选选框来实现功能。老师配置要检测的学生，配置完成后，点击“检测”按钮，即可开始检测。

图 4.16: 检测配置界面截图

一旦完成抄袭检测，测试抄袭检测系统将可视化层上展示所有检测结果。在图 4.17中，它显示了抄袭检测结果信息界面。在这个界面的顶部是检测结果的概述，下面则是检测列表所有学生对的统计表和抄袭关系图。“统计结果”视图中为本次检测所有学生对相似度及是否抄袭信息，点击其中某组学生对检测结果信息后的“详情按钮，可以进一步查看该组学生对具体详情信息。“抄袭关系图”视图中则以圆形来代表一个个选手，无抄袭关系选手用蓝色表示，之间没有

连线；有抄袭关系选手用橙色表示，之间有连线。通过图表的形式，更生动形象地向用户展示抄袭检测结果信息。这样，用户可以非常直观地根据颜色的区分以及连线关系，了解到学生之间的抄袭关系，甚至可以分析出连带关系。

图 4.17: 检测结果信息界面截图

图 4.18: 检测结果详情界面截图

当用户点击抄袭检测“统计结果”视图中每个学生对结果后的“详情”按钮，会跳转到详情界面，显示两两学生对之间相似度及测试片段情况详情。在详情界面中（对应于图 4.18），测试抄袭检测系统显示了每对学生的三个最有可能是抄袭的测试片段对的相似度，以及测试目标和测试片段对的详细代码。

No: 20200118104340605　　检测时间：2020-01-18 10:43:40

题目：Tarjan

检测选手范围：全部选手 共计：85

相似度阈值：80.0%

检测结果

共检测选手对：3570　　抄袭选手对：127

序号	选手ID对	最大相似度	是否抄袭	详情信息
1293	<52518,52819>	100%	是	详情
1297	<52518,56128>	100%	是	详情
1299	<52518,52917>	100%	是	详情
1315	<52518,53054>	100%	是	详情
1316	<52518,54414>	100%	是	详情
1318	<52518,56195>	100%	是	详情

图 4.19: 检测报告部分截图

最后，若检测人员或老师希望查看或下载详细检测报告，系统检测报告生成模块可以自动生成生成检测报告文件供检测人员或老师下载查看。以某次软件测试比赛总决赛赛题“Tarjan”为例，检测报告部分截图如图 4.19，检测报告第一部分为本次检测相关信息，包括本检测报告 ID，本次检测时间，比赛/考试题目，本次检测选手范围以及相似度阈值。下面第二部分则为检测结果，最上面首先可以看到本次检测共检测选手对以及其中抄袭选手对。下面则为检测结果表，包括序号、选手 ID 对、最大相似度、该选手对是否抄袭以及选手代码片段详情信息，其中选手对从上到下按照最大相似度由大到小进行排序，这样保证抄袭选手对在前面标红显示，老师可以先看到抄袭选手对。点击相似度值，可以跳转到相似度矩阵，显示该选手对各个测试片段具体相似度值。点击详情信息这一列下的“详情”，可以跳转到具体测试片段查看详细代码。

4.7　本章小结

本章主要对本系统中关键模块进行详细设计并介绍实现细节。本章介绍的关键服务有待测方法提取模块、测试片段提取模块、相似度计算模块、抄袭分析模块以及检测报告生成模块，通过类图、时序图等方式进一步介绍详细设计思路。最后，采用展示系统界面与描述关键代码相结合的方式详述系统各模块的实现细节，为后一章进一步介绍系统功能性测试与非功能性测试做好充分准备。

第五章　系统测试与实验评估

5.1　系统测试

要对系统进行测试，需要首先对系统进行部署。本系统通过利用 Jenkins、K8s、Docker 等工具去进行部署。本系统借助 Jenkins 负责代码持续集成部署，由 Docker 通过 K8s 和 Docker 部署系统各个子服务。

系统测试是在实际运行环境下对被测系统进行的一系列严格有效地测试，通过与系统需求定义作比较，以发现可能存在的潜在问题，保证系统的正常运行。依据前文中对系统需求的描述，测试代码抄袭检测系统需要保证良好的高可用与高性能。由此，本小节将从功能与性能方面对系统各模块进行严格测试。

5.1.1　测试目标与测试环境

第一，功能测试 (Functional Testing)，主要根据系统特性、业务描述对系统各功能进行验证，根据功能测试用例逐项测试，使得交付系统能够满足功能需求，实现软件测试代码抄袭的高效检测以提高抄袭检测效果。

第二，性能测试 (Performance Testing)，主要负责检测，在不同压力流量下获取系统各性能指标，度量系统相对于预定目标的差距以进一步调优性能，确保测试代码抄袭检测系统在多种情况下的稳定运行。

系统测试环境是用于执行测试用例的一系列软件和硬件的集合，如表 5.1所示。其中，硬件环境指测试必需的运行 *Angular* 2.4、*Spring Boot* 2.1.1 与 *MySQL* 5.7.16 程序的 *ECS* 服务器、客户端中的用户计算机等辅助硬件设备所构成的环境；软件环境指本系统运行时的 *Chrome* 浏览器等应用软件构成的环境。前端服务运行环境依赖于 *Npm* 6.4.1 与 *Node.js* 8.9.0，后端服务运行环境则依赖于 *JDK* 1.8、*Docker* 17.09 与 *Tomcat* 8.0.23，且上述服务均于阿里云服务器中单独部署。

在用户计算机上使用 *Chrome* 73.0 浏览器与系统前端交互，通过不同测试用例的设计来对系统中各模块进行功能测试；性能测试则是利用 *Postman* 7.20.1 对关键接口进行测试。除数据库连接外，应用间均采用 RESTful 接口进行通信。

5.1.2　功能测试

功能测试无需清楚了解模块内部具体数据模型，而是通过黑盒测试，即功能测试用例的设计与执行，来检验系统行为是否满足预期用户需求。

表 5.1: 系统测试环境表

服务	硬件环境	软件环境
前端服务	ECS 服务器 (CentOS 7)	Angular 2.4 Node.js 8.9.0 Npm 6.4.1
后端服务	ECS 服务器 (4G 内存，100M 带宽)	Spring Boot 2.1.1 JDK 1.8 Tomcat 8.0.23 Docker 17.09
数据库服务	ECS 服务器 (CentOS 7)	MySQL 5.7.16
系统测试服务	用户计算机 (Mac)	Chrome 73.0(64bit) JDK 1.8 Postman 7.20.1

测试用例设计

功能测试无需清楚了解模块内部具体数据结构的设计，而是通过黑盒测试，即功能测试用例的设计与执行，来检验系统行为是否满足预期用户需求。本系统功能测试主要从各层，并结合测试用例设计来展开。因各层负责业务特性不同，测试重点又各有不同。Dao 层主要与数据库等数据源进行交互，故测试重点主要为连通性与语法正确性。Service 层负责各细粒度微服务模块业务逻辑，故测试重点为各个服务的交互及正常运作。Logic 层负责参数校验与服务调用，故测试重点为异常参数校验。

如表5.2所示，在系统用户正式进行检测之前，需要通过选择考试/比赛的操作触发配置操作，配置中系统会自动对考试项目代码进行分析，提取待测方法。该测试用例模拟了检测用户从用户登入、考试配置、提取待测方法的全过程，在预期情况下，若使用系统的检测用户，该用户有资格执行测试检测任务，其选择测试考试项目后，配置页面会显示具体待测方法信息，系统提示提取待测方法成功。

如表5.3所示，在系统用户正式进行检测之前，需要通过选择考试/比赛的操作触发系统的测试片段的提取，通过分析学生提交测试代码，提取出合适的测试片段。该测试用例模拟了检测用户从用户登入、考试配置、测试片段提取的全过程，在预期情况下，若进行检测的检测用户，其配置操作符合规范，则该用户将有资格执行测试片段提取，提取成功后系统提示该用户测试片段提取成功。此测试片段提取功能测试用例将在测试用例 UC_TC_1 执行完毕后开展。该用例

表 5.2: 待测方法提取测试用例设计表

用例 ID	UC_TC_1
所属模块	待测方法提取模块
测试目标	系统完成对待测方法自动提取工作，并验证提取结果是否准确
操作步骤	1. 检测用户登入成功 2. 用户查看比赛/考试题目列表 3. 用户点击进行待测方法提取
预期结果	1. 系统显示用户首页页面 2. 系统展示考试列表信息 3. 系统将用户与本次测试方法提取任务进行综合验证，验证成功提示待测方法提取成功并展示提取结果，否则提示失败

模拟了检测用户查看学生测试代码片段提取结果的一系列操作步骤，包括查看学生测试代码具体信息、查看测试片段详情等。考虑到相关功能需求并不复杂，因此该测试用例设计也较为简单，其重点在于系统状态更新的准确性与及时性。

如表5.4所示，本测试用例主要检测对考试/比赛中各位学生的测试代码片段之间相似度进行计算的功能，通过验证相似度计算的准确性来验证计算效果。该测试用例模拟了检测用户从用户登入、检测配置到进行检测的全过程，在预期情况下，若系统中检测用户，其用户登入成功，并且按照规范成功配置了要检测的考试及学生列表，并且系统成功进行了测试片段之间相似度计算，那么本相似度计算测试用例预期结果为，首先系统显示用户首页页面，接着系统显示考试及学生相关信息列表，最后将用户本次相似度计算结果进行综合验证。

如表5.5所示，在进行相似度计算并获取相似度结果后，会自动触发抄袭分析模块，根据相似度结果及设定阈值进行抄袭分析。通过将相似度值与阈值进行比较筛选出抄袭学生对。该测试用例模拟了检测用户从用户登入、检测配置到进行检测的全过程，在预期情况下，若使用系统的检测用户，其检测配置符合系统设定规范，则该用户将有资格执行检测任务，并最终对抄袭分析结果进行综合验证。

如表5.6所示为检测报告生成测试用例设计表，为提高报告可读性，测试报告按照检测用户阅读习惯编排。在系统用户正式进行测试报告生成之前，需要通过选择考试/比赛并选择生成测试报告操作触发系统的自动生成检测报告，通过验证检测报告内容等信息，验证该功能有效性。该测试用例模拟了检测用户

表 5.3: 测试片段提取测试用例设计表

用例 ID	UC_TC_2
所属模块	测试片段提取模块
测试目标	系统完成对学生测试代码片段自动提取工作并验证片段提取结果是否准确
操作步骤	1. 检测用户登入成功 2. 用户查看学生测试代码列表 3. 用户点击进行测试片段提取
预期结果	1. 系统显示用户首页页面 2. 系统展示学生测试代码列表信息 3. 系统将用户与本次测试片段提取任务进行综合验证，验证成功后提示测试片段提取成功并展示提取结果，否则提示失败

从用户登入、检测查询到生成报告的全过程，在预期情况下，若本系统中的检测用户，其生成报告配置符合系统预先设定规范，则该用户将有资格执行生成检测报告操作，系统提示该用户检测报告生成成功。

如表 5.7为本系统功能测试结果表，如表中所示，所有测试用例都成功通过，可见本系统完成对之前设计的功能点的全部实现。实际测试结果符合预期，这表明被测对象-测试代码抄袭检测系统的功能已达到预设的需求，符合软件产品设计。

本系统严格按照测试用例描述步骤，依次执行所有并记录结果到表 5.7中，可以看出执行结果全部通过，表明系统完成了需求分析里提炼的系统功能需求，被测系统行为与系统功能计划吻合。

5.1.3　接口测试

接口测试是测试系统组件间接口的一种测试。接口测试主要是为了对系统所有对外接口测试其功能实现及稳定性。本系统采用前后端分离，前后端通过 RESTful API 进行数据传输，数据对象一致在系统对接中至关重要 [41]。接口测试在本系统中主要负责验证交互数据格式。

本系统主要使用 Postman 工具发送 HTTP 请求对后端服务接口的可访问性与返回对象的数据结构正确性进行验证。限于篇幅，本小节仅展示相似度计算功能接口测试用例图，以及实现数据验证的 Postman 测试代码。接口测试首先

表 5.4: 相似度计算测试用例设计表

用例 ID	UC_TC_3
所属模块	相似度计算模块
测试目标	系统完成对代码相似度的自动计算工作，并验计算证结果是否准确
操作步骤	1. 检测用户登入成功 2. 用户查看考试及学生列表并配置 3. 用户选择进行相似度计算
预期结果	1. 系统显示用户首页页面 2. 系统展示考试及学生列表信息 3. 系统将用户本次相似度计算结果进行综合验证，验证成功后提示用户相似度计算成功并展示计算结果，否则提示失败

需要前面的功能测试全部通过，并且各个逻辑层功能实现，在此基础上验证接口对外实现一致性及性能。

接口创建接口测试结果如下。本接口测试为了模拟真实场景下的用户请求，模拟了三次，每次有 10 个 POST 请求，从参数正常、参数异常和无权限三个角度对接口进行测试，并利用 Postman 工具中的脚本对返回结果进行格式校验。本系统对后端返回参数进行包装，通过 code 字段表示 HTTP 状态码。

5.1.4　性能分析

对于本系统而言，性能的关键点之一在于系统的健壮性。为了进行健壮性分析，本系统选取某著名在线软件测试实践平台 M 上的 40 个数据集进行了实验。选取的这 40 个数据集均为较有代表性的，并且有 50 个学生以上参加的项目。按照这样的标准去选取的主要原因在于，参加人数更多的项目，其中更容易存在抄袭，更有实验价值。

通过对系统的健壮性进行提升，并进行了一系列的健壮性实验，最终本系统本次性能分析实验中跑的 40 个数据集达到了 100% 的通过率，其中 93% 的数据集中所有测试脚本全部检测通过，而余下 7% 的数据集虽然其中少量脚本未能检测成功，但其他大部分没有问题的脚本均可成功检测，并最终可以输出检测结果，保证系统整体流程正常运行，健壮性得到了保障。

系统性能的另一个关键点在于检测耗时。本文通过在 40 个数据集上进行实

表 5.5: 抄袭分析测试用例设计表

用例 ID	UC_TC_4
所属模块	抄袭分析模块
测试目标	系统完成对学生测试代码的抄袭分析工作，并验证抄袭分析结果是否准确
操作步骤	1. 检测用户登入成功 2. 用户查看比赛/考试列表 3. 用户选择学生进行抄袭分析
预期结果	1. 系统显示用户首页页面 2. 系统展示比赛/考试列表信息 3. 系统对本次抄袭分析结果进行综合验证，验证成功后提示用户抄袭分析成功并展示分析结果，否则提示失败

验后并进行统计分析。本系统检测对象为学生的测试脚本，每两个学生之间组成检测对。如 n 个学生会产生 n*n/2 个检测对。通过实验得出结论：在 150 个测试脚本以下，即大约一万个检测对以下，检测需要 30 分种以内；当测试脚本数量达到 600 以上，检测耗时已经达到几小时。目前系统中小于 50 个学生是实时的；一般大于 50 个学生是做成离线的，点击检测后，不会让用户一直等待，在后台跑，等结果出来再提示用户。另外，将本系统耗时与基准工具 Difflib 和成熟工具 Plaggie 进行对比，发现本系统检测耗时性能较好。

本系统引入 MySQL 和 Redis 存储学生提交的测试代码、比赛/考试题目项目代码上传文件和检测结果 PDF 等文件。其中对与各种代码数据及检测结果数据的存储性能要求非常高。特别需要在多性能场景下验证系统数据反馈性能，本文在多线程情况下进行性能分析，如 5.8为性能测试分析表。

表 5.9为上述测试用例执行对比，可以显著看出经过 Redis 缓存后，系统 QPS 有很大比率提升。在只读的期望情况下，缓存给系统带来近一倍的性能提升。但此处仅针对读操作且不涉及缓存更新与脏读数据，所以在实际情况中这种提升会有一定程度下降。除此以外，还可以观察到在连接数上升后，系统 QPS 上升趋势减缓，经过分析 Wrk 执行结果，可发现该现象是由连接数过多从而导致超时查询变多，进而降低了 QPS 增速。

未来，本系统还会对性能提升进行进一步研究。除了利用缓存来提升性能，还会考虑通过对检测方式、对象进行研究，来提高时间性能。可以考虑利用学生抄袭之间的传递性，或者只检测高分或分数接近的学生，降低检测耗时。

表 5.6: 检测报告生成测试用例设计表

用例 ID	UC_TC_5
所属模块	检测报告生成模块
测试目标	系统完成对学生测试代码的检测报告生成工作，并验证报告生成是否准确
操作步骤	1. 检测用户登入成功 2. 用户查看考试列表 3. 用户配置信息进行检测报告生成
预期结果	1. 系统显示用户首页页面 2. 系统展示考试列表信息 3. 系统将用户与本次检测报告生成任务进行综合验证，验证成功后提示用户检测报告生成成功并展示报告结果，否则提示失败

表 5.7: 功能测试用例执行结果表

测试用例 ID	**功能测试执行结果**
UC_TC_1	执行通过
UC_TC_2	执行通过
UC_TC_3	执行通过
UC_TC_4	执行通过
UC_TC_5	执行通过

5.2　实验评估

本文实验评估部分首先介绍实验设计，对比工具，然后将就实验结果，从多个角度来进行对比分析，通过将使用程序切片技术与不使用程序切片技术进行对比，验证本系统创新点“静态双向程序切片”技术的作用。最后，通过将本系统与 Difflib 和 Plaggie 进行对比，评估本系统测试代码抄袭检测效果。

5.2.1　实验设计

本文实验在 MoocTest 中 40 个数据集上进行实验，这些数据集均为超过 50 名学生参与并提交测试代码的真实在线考试数据集。在这些数据集中，学生需要设计和编写测试用例来测试所有的 Java 项目的待测方法。全部 40 个数据集的信息如表 5.10所示，对所有数据集按参加人数进行排序，展示信息包括数据集名称和参加人数。

表 5.8: 性能测试分析表

用例编号	线程数（个）	连接数（个）	时间（秒）
pt-1	6	200	40
pt-2	12	400	40
pt-3	24	800	40

表 5.9: 性能对比结果表

用例编号	无缓存 QPS	有缓存 QPS
pt-1	376	864
pt-2	762	1399
pt-3	1291	2275

表 5.10: 实验数据集信息表

数据集	Datalog	ALU	RBT	Tarjan	TerT	AStar	TrieT	OST
参加人数	619	596	596	587	526	526	526	526
数据集	Hotel	Hash	KDT	SegT	RBBST	EQee	PHM	Square
参加人数	487	487	439	439	439	417	417	417
数据集	MoreTr	FBul	STM	BinH	Calcu	Chef	NBC	TAN
参加人数	396	396	321	321	271	271	271	271
数据集	ITClo	Anag	BFU	JCLO	Jipa	Sudo	BST	StF
参加人数	224	224	224	177	177	177	168	168
数据集	ACO	MSD	MaI	Quad	CMD	Luna	Sch	BPT
参加人数	108	108	87	87	56	56	56	56

对于每个数据集，本文利用研究生进行双保险检查标注。首先，所有测试代码平均分为两组 T_1 和 T_2。每套都由独立的两位研究生进行检查标注。在 T_1 和 T_2 检查和标记之后，将标注结果有不同意见的学生对提供给另外两名研究生作最后决定。

实验准备如下：对于在每个数据集上进行的实验，本文首先建立 TPDS（测试代码集），每个数据集中的测试代码数如上，如 Datalog 数据集有 619 个学生的测试代码，数据集信息如表 5.10所述。其次，对于 TPDS 中每个学生的测试代码，本系统提取锚定待测方法的测试片段。

本文设计了以下两个系列实验。首先，本文选取本系统与另外两个与之相似的代码/文本抄袭检测工具进行对比实验，为了验证本系统抄袭检测的效果。接

着，本文对基于测试文件（simfile）进行检测和基于测试片段（simfrag）进行检测（本文分别称为非程序切片和程序切片），进行对比实验，验证本系统核心技术“静态双向程序切片”的作用。在实验中，一个学生的代码可以包含多个文件。对于非程序切片，本文使用要检测的文件对来表示两个学生，一对检测文件对来自两个学生。对于程序切片，本文使用要检测的片段对来表示两个学生，一堆检测片段对来自两个学生。最后，本文基于阈值进行抄袭分析并生成测试抄袭检测结果。对于两个系列实验的结果，本文基于评估指标计算准确率并进行分析。

5.2.2　评估指标

本文采用三种评估指标来评估本系统的有效性，即：精度、召回率和 F_1 度量值。精度对应于抄袭对在阈值检测的学生对中所占比例。召回率对应于检测出来的学生抄袭对占所有被人工标记出来的抄袭对比例。对于测试代码抄袭检测，精度和召回率都很重要，所以本文采用 F_1 度量值来评价本系统。F_1 度量值是精度和召回率的调和平均。

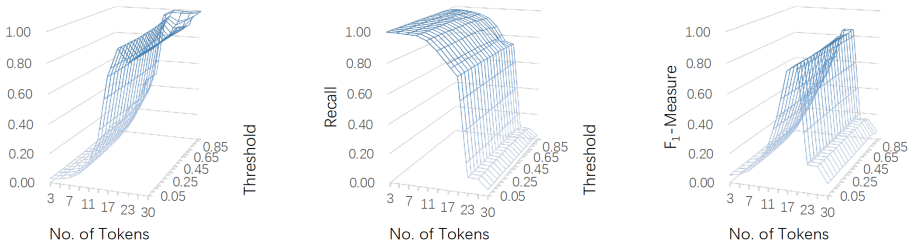
5.2.3　对比工具

为了验证本系统的性能，本文以本系统与两个经典的工具进行比较，Plaggie 和 Difflib。Plaggie 是一个开源的抄袭检测工具，是专门设计的用于检测用 Java 编写的生产代码的抄袭。Difflib，作为基线工具，可以用来比较文本并生成增量，基于这些文本的相似性可以计算测试代码的相似行数，以便进一步检测抄袭行为。

由于 Plaggie 在不同场景中的性能受参数 minMatchLength 的影响，本文对 Plaggie 做了一个补充实验，探索一个相对合适的 minMatchLength 值，使 Plaggie 能选取一个相对较好的性能状态参与对比实验场景。实验结果表明，当 minMatchLength=17 时，Plaggie 可以在场景中获得相对最优的性能，具体结果如图 5.1所示。

5.2.4　实验结果

本文把本系统与 Difflib 及 Plaggie 进行对比，验证本系统抄袭检测效果，其中 Plaggie 选取上述补充实验中最优性能来参与对比。对于每个工具检测得到的结果，本文计算了基于对测试代码数据集标注结果的分析。选取的是一组相似性度量工具中的佼佼者 Plaggie 和 Difflib 来与本系统进行对比实验。本实验是通过将本系统与相似的相似性度量工具在多个数据集上进行实验，并计算最终结果的精度、召回率和 F_1 度量值，并最终将结果进行统计分析。



(a) Precision of Plaggie　(b) Recall of Plaggie　(c) F_1-Measure of Plaggie

图 5.1: Plaggie 实验结果图

图 5.2主要显示所有实验中, 表 5.10中数据集的 F_1 度量值的平均值结果，以此为代表来展示及分析本系统与其他工具对比实验结果。直觉上，相似度越高，抄袭的可能性就越大。在实验分析中，本文发现三种工具的准确率（F_1）随着相似度阈值的增加，抄袭检测结果得到提升，这与直觉是一致的。多次实验证明，当阈值取大于 0.8 时，抄袭分析的结果才较为准确，这是因为测试代码由于结构性等特点，本身相似程度就较大，若相似度阈值过低，会误报很多非抄袭的学生对。因此图中展示阈值从 0.8 到 1 之间的实验结果，可以看到本系统抄袭检测准确率高于 Difflib 和 Plaggie，最高可达 90% 以上。因此，可以验证本系统的抄袭检测准确率优于基准工具 Difflib 以及检测代码抄袭的成熟工具 Plaggie。

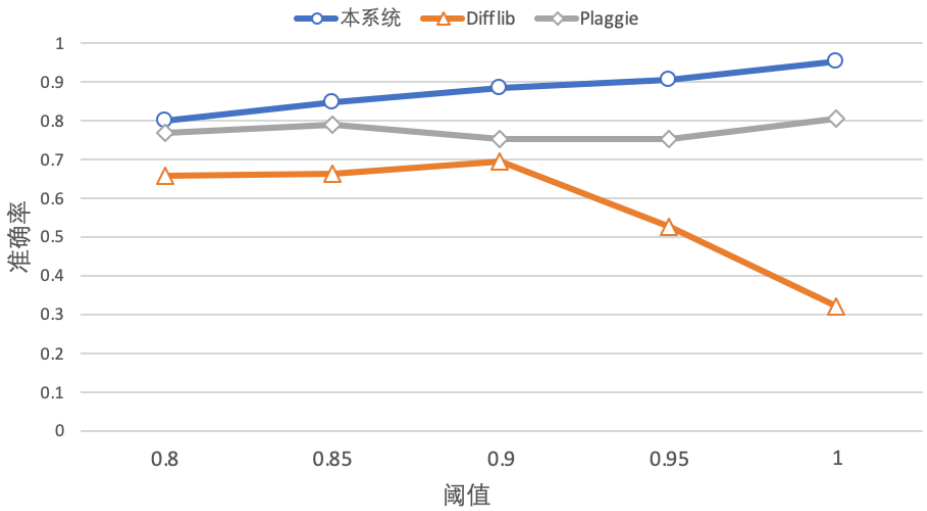


图 5.2: 本系统与其他工具对比实验结果图

基于上图结果，本文详细分析本系统针对测试代码的抄袭检测方法相比现有生产代码抄袭检测工具的优势。另外，引入了基准工具 Difflib 来作为 baseline。本文分析当使用 Difflib(TD),FuzzyWuzzy(TF) 和 Plaggie(TP) 时，召回率随

阈值从 0.00 到 1.00 显示精确的结果。因为 Plaggie 在不同场景中表现会受到影响，通过参数'minMatchLength'，本文已经完成对 Plaggie 探索一个相对合适的"minMatchLength"值，以便 Plaggie 可以在片段化中达到相对良好的性能状态。实验结果表明 minMatchLength=17，Plaggie 可以实现相对片段化场景中的最佳性能，以及结果如图 5.1所示。对于 Difflib，当阈值 =0.4 时，它可以获得相对最好的性能。当阈值 =1.0 时较好性能则为 FuzzyWuzzy 和 Plaggie。再根据精确性、召回率和 F_1，本文发现本系统比 Plaggie 和 Difflib 性能更好，换句话说，本系统比 Plaggie 和 Difflib 更适合测试代码抄袭检测的场景。分析缘由，一方面，这是由计算方法引起的。使用 Difflib 度量相似性，如果两个文件中的两行，即使差异很小，Difflib 仍然会将它们标记为不同的。例如，假设学生 A 写了一个名为 F_A 的测试文件，学生 B 编写了一个名为 F_B 的测试文件。文件 F_A 中的测试代码从文件 F_B 复制并对每个语句进行修改（例如修改标识符）。在在这种情况下，Difflib 生成的输出 (F_A,F_B) 将减小，这也导致了 simD(F_A,F_B) 相对而言要小。但是，通常的做法是抄袭后学生会进行修改以混淆检测。因此，由 Difflib 进行的相似度计算通常比本系统小，而漏报很多抄袭。另一方面，本文发现很多学生抄袭他人的测试代码，未做任何修改，因此 Difflib 的相似度为 1.0。所有这些都可以解释为什么 Difflib 的性能可以在很小的阈值情况下，反而获得较好的性能。此外，本文还发现，当 Plaggie 达到相对较好的效果时，存在很多假阳性对（误报学生对）。综合以上分析，本文可以得到结论：本系统在测试抄袭检测场景中，比面向生产代码的工具 Plaggie 和基准工具 Difflib 效果好。

另外，为了证实本系统采用的"静态双向程序切片"技术（简称程序切片）的有效性，本文对直接用测试文件代码进行检测和对提取的测试片段代码进行检测进行了实验。本文进行实验，并将结果显示在图 5.3中。在本实验中，本文使用 Difflib、FuzzyWuzzy（本系统）和 Plaggie 三种相似性度量工具验证了程序切片的有效性，并在精确性、召回率和 F_1 度量下进行了比较。如图 5.3所示，子图（a）-（c）分别说明了使用 Difflib 时进行非程序切片（文件）和程序切片（片段）的精度、召回率和 F_1 度量。与 Difflib 类似，后六个子图形（d）-（i）分别表示使用 FuzzyWuzzy 和 Plaggie 时执行非片段（File）和片段（Fragment）的精度、召回率和 F_1 度量。在每个子图中，水平轴表示本文配置的阈值（即 0.05、0.10、…、1.00）。沿垂直方向轴则是表示精度值（或召回率，F_1 度量）。本文使用三角形点的蓝色曲线来表示非程序切片的结果，使用圆形点的红色曲线来表示程序切片的结果。具体结果分析如下。

然后，基于实验结果，比较程序切片与非程序切片，对程序切片的有效性进行分析。为了便于理解，本文通过使用测试代码示例来解释实验结果。现在，假

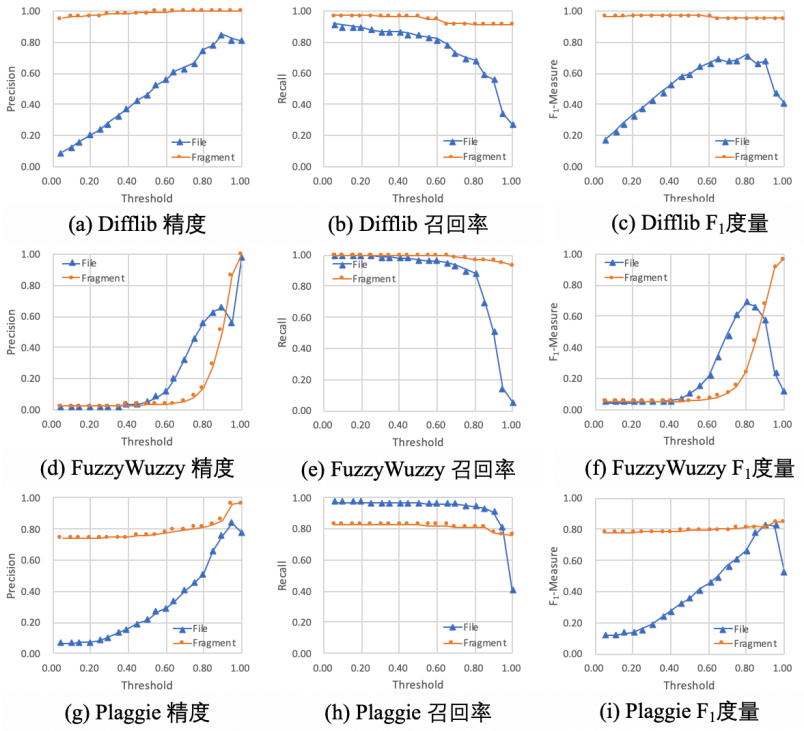


图 5.3: 基于文件与基于片段检测对比实验结果图

设文件 1、文件 2 和文件 3 分别是学生 A、B 和 C 编写的测试文件，其中 B 和 C 是抄袭对。在精度方面，如图 5.3-(a),(d) 和 (g) 所示，片段化场景中 Difflib 和 Plaggie 的精度总是高于文件场景中的精度。对于工具 FuzzyWuzzy，当相似度阈值阈值大于 0.4 小于 0.9 时，文件的精度高于片段化，但当阈值大于 0.95 时，则相反。原因是片段化计算的相似度值总体上高于文件。例如，本文使用 FuzzyWuzzy 计算 A 和 B 之间的相似性。在文件场景中，$sim(A,B) = sim(file_1, file_2) = 56$，而在片段化场景中，$sim(A,B) = sim(TF_1, TF_2) = 67$。同样，A 和 C，B 和 C 也是一样的。因此，由相同阈值选择的抄袭对的数目相应增大，特别是假阳性（误报）对。因此，综合来说，可以得出结论，程序切片可以提高抄袭检测工具的精度。对于召回率，如图 5.3-(b),(e) 和 (h) 所示，片段化场景中 Difflib 和 FuzzyWuzzy 的召回率总是高于文件场景中的召回率。对于 Plaggie，当阈值阈值大于 0 小于 0.95 时，文件的召回率高于片段化，但当阈值大于 0.95 时，Plaggie 片段化场景下的召回率较高，原因是文件场景下召回率迅速下降，而在片段化中召回率则平稳下降。阈值很小时（例如阈值 =0.05），片段化场景中 Plaggie 的召回率较低（在 FuzzyWuzzy 和 Difflib 中，召回率接近于 1.0）。主要原因是，对于 Plaggie，首先将测试片段 TF_1 和 TF_2 分别转换为两个 tokens:tokens1（三个 token）和 tokens2（四个 token）从而

计算 TF_1 和 TF_2 的相似性。在这种情况下，如果 minMatchLength=4,sim(B,C)=0，即当 minMatchLength>minTokens 时，本来是抄袭对，但因 token 数量不够，相似性仍为 0。由于 token 数量不固定，无论阈值是多少，总会有真正的抄袭对被省略，所以召回率不会达到特别高的水平。此外，如图 4-(e) 显示，当阈值从 0 增加到 1 时，FuzzyWuzzy 的召回率变化很小，在片段化场景中几乎接近 1.0。而对于 F_1 度量，可以看到三个工具的实验结果在阈值大于 0.85 以后都是片段化更好。综合以上分析，本文可以发现程序切片对于提高相似度检测工具在抄袭检测中的性能是有意义的。

5.3　本章小结

本章根据需求，对系统的基础功能进行了测试，同时，设计了测试用例对系统进行了测试，然后利用 Postman 对系统进行了接口测试，另外对本系统进行了性能分析。在真实的实验条件下，进行两个系列实验，一方面，将本系统与业内具有竞争性的抄袭检测工具进行了对比实验，验证了本系统进行测试代码抄袭检测的有效性；另一方面，将不使用程序切片技术与使用程序切片技术进行对比实验，进一步验证了本系统采用程序切片技术的有效性。

第六章　总结与展望

6.1　总结

为解决传统软件抄袭检测过程中存在的通过改变参数变量、粘贴进大量无用代码来躲避检测和检测软件生产代码抄袭工具与检测软件测试代码抄袭场景不适用等问题，本系统创新性地利用测试代码静态双向切片技术、片段化分析技术的优势，通过提取考试/比赛题目项目代码、学生提交测试代码片段，提高了测试抄袭检测精度。并且本系统利用提取出来的测试片段，针对两两测试片段进行相似度计算，而不是对整个代码文件进行相似度计算，进一步提升来测试抄袭检测精度。本系统对抄袭检测结果进行持久化存储，检测结果交付过程中涉及到的代码实体信息均在 MySQL 与 Redis 中可追踪，进一步优化了传统检测结果交付流程，增强了系统的健壮性与稳定性。

本文设计与实现了一种基于 Web 的可检测抄袭行为的测试代码分析工具，即测试代码抄袭检测系统，其通过考虑测试用例的特点可以从非标准测试代码中提取有效的测试片段。测试抄袭检测系统可以很便捷地集成到其他测试代码分析工具中。除了警告可能存在抄袭现象的测试代码外，测试抄袭检测系统还借助强大的 Web 用户界面，可以显示最有可能抄袭的测试片段详情，从而方便用户进一步检查。基于提取测试片段技术，本测试抄袭检测系统对测试代码的相似性度量更加合理，从而提高了测试代码抄袭检测的准确性，进而提升软件测试教育质量。

6.2　展望

基于程序切片的测试代码抄袭检测系统的职责在于对在线软件测试实践平台中的考试和比赛的测试代码进行自动化抄袭检测，主要是 Java 单元测试。但从长远发展的层面上看，随着开发语言的多样化、在线教育参与人数越来越多，本系统仍需要不断进行优化完善。后续工作可以从以下方面着手。

第一，本系统在功能上还可以进一步拓展，当前测试代码抄袭检测仅仅局限于对 Java 单元测试框架 Junit 下的代码进行检测，为了应对多种开发语言的测试代码，本系统还可以去拓展对于 C++、Python 等其他编程语言的测试代码的抄袭检测服务。另外，除了对开发者测试的测试脚本进行抄袭检测外，本系统还可在未来拓展对于 Web 测试，移动 APP 测试等其他领域测试的抄袭检测。

第二，本系统内接口设计时已基本考虑底层技术与底层服务的可替换性，例如文件系统与缓存系统，不同底层技术之间的组合是否会带来更好地性能或者更高的稳定性，还需要进一步的探索与测评。与此同时，底层技术的选用应可在配置文件中进行配置。

第三，虽然系统已尽可能注意性能优化，但随着检测脚本数量进一步上升，本系统在时间消耗上还有待改进，此时需要通过研究学生之间抄袭的传递性，或者通过只检测部分高分代码，或者检测部分分数相近的学生代码，来提升耗时方面的性能。

第四，本系统需要依赖人为设定的相似度阈值来进行检测，阈值的设定很大程度上影响了最终检测结果。为每个项目找到最佳阈值是非常困难的。未来可以考虑通过运用机器学习等技术，去除对阈值的依赖。

最后，本系统还有和其他编程语言代码相似度检测相结合的可能性。系统中程序切片技术同样可用于其他编程语言检测，相似度计算可以对其他语言选取合适的工具进行检测，丰富了软件测试训练中可进行抄袭检测的范围选择。

参考文献

[1] M. Staron, Action Research in Software Engineering - Theory and Applications, Springer, 2020.

[2] N. Tillmann, J. de Halleux, T. Xie, S. Gulwani, J. Bishop, Teaching and learning programming and software engineering via interactive gaming, in: 35th International Conference on Software Engineering, ICSE '13,San Francisco, CA, USA, May 18-26, 2013, pp. 1117–1126.

[3] A. Ahtiainen, S. Surakka, M. Rahikainen, Plaggie: Gnu-licensed source code plagiarism detection engine for java exercise, in: Proceedings of the 6th Koli Calling, 2006, pp. 141–142.

[4] Fuzzywuzzy: Fuzzy string matching in python, https://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/, accessed: 2019.

[5] Difflib-helpers for computing deltas, site: https://docs.python.org/2/library/difflib.html, accessed: 2018.

[6] G. Whale, Plague: plagiarism detection using program structure, School of Electrical Engineering and Computer Science, 1988.

[7] K. W. Bowyer, L. O. Hall, Experience using "MOSS" to detect cheating on programming assignments, in: Proceedings of the 29th FIE, 1999, pp. 13–18.

[8] L. Prechelt, G. Malpohl, M. Philippsen, Finding plagiarisms among a set of programs with JPlag, Journal of UCS 8 (11) (2002) 1016–1038.

[9] Y. Feng, J. A. Jones, Z. Chen, C. Fang, Multi-objective test report prioritization using image understanding, in: Proceedings of the 31th International Conference on Automated Software Engineering, 2016, pp. 202–213.

[10] H. Jiang, X. Chen, T. He, Z. Chen, X. Li, Fuzzy clustering of crowdsourced test reports for apps, ACM Transactions on Internet Technology 18 (2) (2018) 18:1–18:28.

[11] Q. Luo, K. Moran, D. Poshyvanyk, A large-scale empirical comparison of static and dynamic test case prioritization techniques, in: Proceedings of the 24th International Symposium on Foundations of Software Engineering, 2016, pp. 559–570.

[12] C. Fang, Z. Chen, K. Wu, Z. Zhao, Similarity-based test case prioritization using ordered sequences of program entities, Software Quality Journal 22 (2) (2014) 335–361.

[13] T. B. Noor, H. Hemmati, A similarity-based approach for test case prioritization using historical failure data, in: Proceedings of the 26th International Symposium on Software Reliability Engineering, 2015, pp. 58–68.

[14] S. Dong, M. Sarem, Ddos attack detection method based on improved KNN with the degree of ddos attack in software-defined networks, Vol. 8, 2020, pp. 5039–5048.

[15] 吴斐，唐雁, 基于 n-gram 的 vb 源代码抄袭检测方法, No. 2, pp. 86–91.

[16] J. Graovac, M. Mladenovic, I. Tanasijevic, Ngramspd: Exploring optimal n-gram model for sentiment polarity detection in different languages, Intell. Data Anal. 23 (2) (2019) 279–296.

[17] H. Chang, A. Mockus, Evaluation of source code copy detection methods on freebsd, in: Proceedings of the 2008 International Working Conference on Mining Software Repositories, MSR 2008 (Co-located with ICSE), Leipzig, Germany, May 10-11, 2008, pp. 61–66.

[18] R. Mukherjee, K. S. Patnaik, Prioritizing junit test cases without coverage information: An optimization heuristics based approach, IEEE Access 7 (2019) 78092–78107.

[19] K. J. Ottenstein, An algorithmic approach to the detection and prevention of plagiarism, ACM SIGCSE Bulletin 8 (4) (1976) 30–41.

[20] C. K. Roy, J. R. Cordy, Nicad: Accurate detection of near-miss intentional clones using flexible pretty-printing and code normalization, in: Program Comprehension, 2008. ICPC 2008. The 16th IEEE International Conference on, 2008.

[21] M. Joy, M. Luck, Plagiarism in programming assignments, Education IEEE Transactions on 42 (2) 129–133.

[22] M. Weiser, Program slicing, IEEE Transactions on software engineering (4) (1984) 352–357.

[23] S. Danicic, M. Harman, Y. Sivagurunathan, A parallel algorithm for static program slicing, Information Processing Letters 56 (6) (1995) 307–313.

[24] N. BniLam, E. Tanghe, J. Steckel, W. Joseph, M. Weyn, ANGLE: angular location estimation algorithms, IEEE Access 8 (2020) 14620–14629.

[25] J. Li, X. Wei, G. Wang, S. Zhou, Improved grid algorithm based on star pair pattern and two-dimensional angular distances for full-sky star identification, IEEE Access 8 (2020) 1010–1020.

[26] Y. Hong, J. J. F. Martinez, A. C. Fajardo, Day-ahead solar irradiation forecasting utilizing gramian angular field and convolutional long short-term memory, IEEE Access 8 (2020) 18741–18753.

[27] K. Guntupally, R. Devarakonda, K. Kehoe, Spring boot based REST API to improve data quality report generation for big scientific data: ARM data center example, in: IEEE International Conference on Big Data, Big Data 2018, Seattle, WA, USA, December 10-13, 2018, 2018, pp. 5328–5329.

[28] K. Pohl, Requirements Engineering - Fundamentals, Principles, and Techniques, Springer, 2010.

[29] M. Fujimoto, W. Matsuda, T. Mitsunaga, Protecting struts 2 from OGNL related attacks by using servlet filter, in: 13th Asia Joint Conference on Information Security, AsiaJCIS 2018, Guilin, China, August 8-9, 2018, pp. 8–14.

[30] D. Lee, D. Choi, Analysis of the reliability of a starter-generator using a dynamic bayesian network, Reliab. Eng. Syst. Saf. 195 (2020) 106628.

[31] P. Levis, N. Patel, D. Culler, S. Shenker, Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks, in: Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation - Volume 1, 2004.

[32] Anderson, Charles, Docker [software engineering], IEEE Software 32 (3) 102–c3.

[33] W. Saad, L. Abidi, H. Abbes, C. Cérin, M. Jemni, Wide area bonjourgrid as a data desktop grid: Modeling and implementation on top of redis, in: 26th IEEE International Symposium on Computer Architecture and High Performance Computing, SBAC-PAD 2014, Paris, France, October 22-24, 2014, pp. 286–293.

[34] N. F. Naim, A. I. M. Yassin, W. M. A. W. Zamri, S. S. Sarnin, Mysql database for storage of fingerprint data, in: Proceedings of the 13th UKSim-AMSS International Conference on Computer Modelling and Simulation, Cambridge University, Emmanuel College, Cambridge, UK, 30 March - 1 April, 2011, pp. 293–298.

[35] Kruchten, P.B., The 4+1 view model of architecture, IEEE Software 12 (6) 42–50.

[36] E. F. Codd, Relational database: A practical foundation for productivity, Commun. ACM 25 (2) (1982) 109–117.

[37] W. Sun, X. Wang, H. Wu, D. Duan, Z. Sun, Z. Chen, MAF: method-anchored test fragmentation for test code plagiarism detection, in: Proceedings of the 41th ICSE SEET, 2019, pp. 110–120.

[38] Google java style guide, site: https://google.github.io/styleguide/javaguide.html, accessed: 2018.

[39] M. Weiser, Programmers use slices when debugging, Communications of the ACM 25 (7) (1982) 446–452.

[40] S. Horwitz, T. W. Reps, D. W. Binkley, Interprocedural slicing using dependence graphs, in: Proceedings of the 6th ACM SIGPLAN Conference on Programming Language Design and Implementation, 1988, pp. 35–46.

[41] G. Zhang, D. Liang, J. M. Zhang, Image analysis measurement of soil pinproceedings movement during a soil-structure interface test, Computers Geotechnics 33 (4/5) 248–259.

简历与科研成果

基本情况 段定，男，汉族，1996 年 12 月出生，江苏省泰州市人。

教育背景

2018.9 ~ 2020.6 南京大学软件学院 硕士

2014.9 ~ 2018.7 南京大学软件学院 本科

读研期间的成果

专利

1. 陈振宇，**段定**，孙伟松，王兴亚，巫浩然，孙泽嵩，“一种基于机器学习的测试抄袭检测方法”，申请号：2019100556994，已受理。
2. 陈振宇、孙伟松、王兴亚、**段定**、巫浩然、赵源、孙泽嵩，”一种基于测试代码片段相似性的测试程序抄袭检测方法”，申请号：201810561223.3，已公示。
3. 刘嘉，孙伟松，徐光耀，**段定**，巫浩然，陈振宇，王兴亚，“一种基于行为相似性的智能合约安全风险评估方法”，申请号：2019104992386，已受理。

论文

1. Weisong Sun, Xingya Wang, Haoran Wu, **Ding Duan**, Zesong Sun, Zhenyu Chen. MAF: Method-anchored test fragmentation for test code plagiarism detection. ICSE 2019-SEET。

致　　谢

当写完毕业论文之后，我要谢谢各位在这一路上教导和帮助过我的人。首先感谢我的导师何铁科老师，何老师在学习和生活上都教导我挺多。在毕业设计阶段，感谢何老师一直负责任地指导我。多亏何老师的敦促与帮助，我因此按时完成了全部毕业设计。

其次，我还要谢谢 iSE 实验室的各位老师们。大四以及硕士阶段，我在 iSE 实验室学习了大量前沿技术，特别是从一开始只会写代码到学会注意代码可拓展性。特别要感谢陈振宇老师，陈老师博学多才、见多识广，在论文选题时期给予我许多宝贵的意见。在论文写作过程中更是悉心指导，多次对论文提出修改建议。再次向陈老师表示衷心的感谢！同时，也要感谢实验室其他老师的指导和帮助！另外，感谢国家自然科学基金：软件工程教育与培训会议（61210306018）支持！

在 iSE 实验室里，我要特别感谢跟我在同一个小组里的各位师兄师弟们。首先感谢王兴亚师兄一直以来的关心和照顾，特别是我刚进实验室那段时间，兴亚师兄耐心地手把手教我，引领我走进研究生的生活。另一个要特别感谢的是孙伟松师兄，他不仅一直以来在学术上指导我科研，还帮助我提升编码能力，他在软件工程、写论文、做科研实验等很多方面都教会了我很多。此外，还要感谢朱晨乾师弟，平时学习科研中，他一直对我鼎力相助。当然，我还要感谢陪伴我研究生生涯的同学们，在找工作和写毕设的过程中，我感受了他们具备的特性——有信心、有耐心、有责任心，与大家一起学习工作的两年非常愉快。

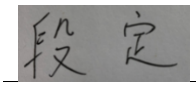
然后，我要感谢母校南京大学对我的栽培。算上本科生涯，我已经在南京大学软件学院学习生活了 6 年，今年我 24 岁，我人生的四分之一都在这里度过，培养了深厚的感情。从仙林机房到鼓楼费彝民楼，软件学院在我心中永远铭记；从懵懵懂懂到拿到名企 offer，母校也见证了我的成长。我会牢记诚朴雄伟的校训，体现在我的工作生活中，不负母校的栽培。

最后，感谢我的父母，他们教会我做人，给予我最好的条件，他们认真负责的态度一直我学习的榜样，他们给我带来的亲情是我一辈子要去守护的。在我找工作和写毕设遇到困难的时候，他们虽然不能给予直接的帮助，但一直给予我心灵上的安慰。感谢他们这么多年来的培养与付出，我一定会努力工作关心家人，成为像他们一样有责任心有担当的人。

附件二

《学位论文出版授权书》

本人完全同意《中国优秀博硕士学位论文全文数据库出版章程》（以下简称“章程”），愿意将本人的学位论文提交“中国学术期刊（光盘版）电子杂志社”在《中国博士学位论文全文数据库》、《中国优秀硕士学位论文全文数据库》中全文发表。《中国博士学位论文全文数据库》、《中国优秀硕士学位论文全文数据库》可以以电子、网络及其他数字媒体形式公开出版，并同意编入《中国知识资源总库》，在《中国博硕士学位论文评价数据库》中使用和在互联网上传播，同意按“章程”规定享受相关权益。

作者签名： 

2020 年 5 月 21 日

论文题名	基于程序切片的测试代码抄袭检测系统的设计与实现				
研究生学号	MF1832034	所在院系	软件学院	学位年度	2020
论文级别	☐硕士 ☑硕士专业学位 ☐博士 ☐博士专业学位 (请在方框内画钩)				
作者 Email	983117808@qq.com				
导师姓名	刘嘉副教授，何铁科助理研究员				

论文涉密情况：

☑ 不保密

☐ 保密，保密期（_____年_____月_____日 至 _____年_____月_____日）

注：请将该授权书填写后装订在学位论文最后一页（南大封面）。