南京大学
NANJING UNIVERSITY

研 究 生 毕 业 论 文

（申请硕士专业学位）

论 文 题 目	软 件 测 试 多 维 评 估 系 统 的 设 计 与 实 现
作 者 姓 名	王 逸 璠
专 业 名 称	软 件 工 程
研 究 方 向	软 件 工 程
指 导 教 师	陈 振 宇　教 授 房 春 荣　助 理 研 究 员

2020 年 5 月 22 日

学　　号：**MF1832168**

论文答辩日期：**2020 年 5 月 23 日**

指 导 教 师：   （签字）

The Design and Implementation of Multidimensional Evaluation System for Software Testing

By

Yifan Wang

Supervised by

Professor **Zhenyu Chen**

Research Assistant **Chunrong Fang**

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：软件测试多维评估系统的设计与实现

工程硕士（软件工程领域） 专业 2018 级硕士生姓名：王逸璠

指导教师（姓名、职称）：陈振宇 教授 房春荣 助理研究员

摘 要

软件测试作为软件开发过程中必不可少的部分，尤其在开发和交付高质量的软件的情况下，其重要性不言而喻。随着市场对软件质量的要求不断提高，企业对高质量测试工程师的需求量越来越大。面对工业界软件测试人才需求日益增长的现状，各大高校和在线教育平台纷纷开设软件测试课程。在课程期末考核时，通过在软件测试教学平台上组织考试练习的方式鼓励学生提交测试代码。然而针对学生实际编写的测试代码，如何有效地考察测试代码，评估其测试效果，以便给出指导建议从而实现进一步的提高，成为了一个值得关注的问题。

本文基于公司软件测试教学平台实现了软件测试多维评估系统，该系统旨在解决如何系统地评估学生测试效果的问题。依据高校的人才培养目标和企业要求，本系统针对开发者单元测试、Web 应用自动化测试和移动应用自动化测试三种测试类型，结合测试代码和测试行为两方面，从 7 个维度对学生的测试效果进行评估。首先，系统获取学生的测试代码，对应不同的测试类型，选取并调用相应的测试代码分析工具对测试代码的特性进行提取。为保障代码分析过程的高可用性，采用 Celery 分布式系统实现分析任务的异步调度执行。其次，以分析提取到的代码特性以及学生考试提交记录数据为输入，根据不同的测试类型，应用提出的软件测试多维评估指标公式，计算得到测试评估的原始得分。由于不同指标分数在初步计算后值域不同，为了使各指标处于同一数量级，方便综合对比评价学生测试评估的结果，对分数应用标准化方法得到标准分。最后，将评估得到的指标分数整合，利用 Echarts 可视化库将其渲染展现供老师和学生直观全面地查看测试评估结果。同时系统支持展示各个维度的详情信息、历史评估记录以及对应的维度得分变化趋势图，通过对各维度评估详情等多方面展示，有助于树立测试能力提升的方向。

本文提出的软件测试多维评估系统旨在更全面地评估学生的测试效果，其评估指标在教学平台主办的软件测试大赛中得到广泛的应用，评估结果得到大赛专家组的一致认可。该系统目前已在软件测试教学平台上使用，通过多方面评估学生的测试效果，以便帮助学生发现不足，从而推动其进步。

关键词：软件测试，测试评估，多维评估，数据可视化

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 Multidimensional Evaluation System for Software Testing

SPECIALIZATION: Software Engineering

POSTGRADUATE: Yifan Wang

MENTOR: Professor Zhenyu Chen, Research Assistant Chunrong Fang

Abstract

Software testing is an essential part of the software development process, especially for the purpose of developing and delivering high-quality software. With the increasing requirements of the market for software quality, the demand for high quality test engineers is increasing. Faced with the increasing demand for software testing talents in industry, major universities and online education platforms are offering software testing courses. At the end of the course assessment, students are encouraged to submit test code in test exercises organized on an online software testing teaching platform. However, for the test code actually written by students, how to effectively investigate the test code and evaluate its test effect in order to give guidance and suggestions to achieve further improvement has become a problem worthy of attention.

In this thesis, we implement a multi-dimensional evaluation system of software testing based on the company's software test teaching platform. This system aims to solve the problem of how to systematically evaluate the effectiveness of students' test. According to the talent training objectives of colleges and universities and the requirements of enterprises, this thesis constructs a multi-dimensional evaluation system to evaluate students' testing effect from various aspects on the basis of 7 evaluation indicators in combination with test code and test behaviors, aiming at three test types of developer unit testing, Web application automation testing and mobile application automation testing. Firstly, the system obtains the students' test code, corresponding to different test types, selects and invokes the corresponding test code analysis tool to extract the characteristics of the test code. In order to guarantee the high availability of the code analysis process, Celery, a distributed asynchronous scheduler system, is adopted to realize the analysis task. Secondly, the system takes the extracted test code analysis

characteristics and the record data of students' exam submission as the input. And then the proposed multi-dimensional evaluation indicator formula of software testing is applied to calculate the original score of test evaluation. Since the value range of different index scores is different after the preliminary calculation, in order to make each indicator at the same order of magnitude, and to facilitate the comprehensive comparison and evaluation of the results of the student test evaluation, the standardized method is applied to obtain the standard score. Finally, the evaluation scores are integrated and rendered by Echarts visualization library for teachers and students to visually and comprehensively view the test evaluation results. At the same time, the system supports the display of the detailed information of each dimension, the historical evaluation records and the corresponding variation trend chart. By showing the details of each dimension evaluation and other aspects, it is helpful to establish the direction for the improvement of testing ability.

The multi-dimensional evaluation system for software testing proposed in this thesis aims to more comprehensively evaluate the test effects of students. Its evaluation indicators has been widely used in the software testing contest hosted by the teaching platform, and the evaluation result has been unanimously recognized by the expert group of the contest. The system has been used in the software test teaching platform, through a variety of evaluation of students' test effect, in order to help students find shortcomings, so as to promote their progress.

Keywords: Software Testing, Testing Evaluation, Multidimensional Evaluation, Data Visualization

目录

表 目 录

图 目 录

第一章　引言

1.1　项目背景与意义

在新形势的软件行业发展下，新技术的发展促进了互联网生态的迭代和软件媒介的多样化，用户群体的需求也随之增长。随着软件渗入到学习、办公、社交、购物、娱乐等场景的程度逐渐加深，用户越来越重视对软件质量的要求。软件的高质量往往通过测试来保证，测试贯穿于整个软件生命周期，其开销占比在软件开发过程中可高达 50% [1]，重要性不言而喻。

随着市场对软件质量要求的不断提高，企业对高质量的测试工程师需求量越来越大，同时也对开发岗位工程师的测试能力提出了要求。在很多软件公司中新员工入职首先需要测试其他开发人员的代码才能贡献自己的代码，因此对于计算机科学和软件工程专业的学生来说，掌握充分的测试知识、具备高水平测试能力是非常必要的。针对目前软件测试行业人才缺乏的问题，各大高校和在线学习平台针对此开设了软件测试课程，以加强对测试人才的培养。由于软件测试是以理论为主导，注重实际操作致力于应用在实际场景的学科，因此在课程考核时除了理论考试以外，通过在线测试教学平台组织线上考试练习，鼓励学生提交测试代码，以实战的方式来考察学生的软件测试掌握的程度。

根据建设性调整 (Constructive Alignment) [2] 理论的要求，要重视教学评估标准的设计。评估结果会给学生带来反馈，对学习过程产生一定的影响，因此科学地设立软件测试的评估标准可以推动学生学习测试知识的积极性，从而进一步提高其测试能力。然而，目前较少有关于软件测试多维评估相关的理论提出。由于测试种类繁多，这里主要针对开发者单元测试、移动应用自动化测试以及 Web 应用自动化测试提出评估标准，原因如下：

一个复杂庞大的系统都是由许多个单独的功能模块堆叠集成起来的，单元测试作为开发者测试最重要的组成部分，用于验证系统最小可测试单元（通常指函数或类）在与程序其他部分隔离的情况下的正确性。单元测试通常由开发人员编写，一般会伴随开发代码提交至代码库，是最接近代码底层实现的验证手段，通过结合单元测试框架简化测试工作，提高效率。

移动应用测试面临着多种挑战，比如多端发布——Android 和 iOS，多版本发布，多机型发布，这导致手工测试很难完全胜任更快更好的质量保证诉求，而伴随着持续交付、DevOps 等理念的落地，业界对移动应用自动化测试的需求也越来越多。

Web 应用由于与用户直接相关，又通常需要承受长时间的大量操作，因此 Web 项目的功能和性能都必须经过可靠的验证。为提高 Web 应用系统的服务质量，如何更好、更有效地测试 Web 应用的技术也是层出不穷。凭借可复用、易维护、定时处理、持续集成、可调试、测试结果自动通知等特点，Web 自动化测试成为目前 Web 应用测试主流的测试方式。Web 自动化测试按照测试人员的测试计划编写测试脚本，通过自动化测试工具进行自动测试，大大减轻了手工测试的工作量，降低了软件测试的时间和成本。

鉴于此，本文基于公司软件测试教学平台，针对学生参与线上考试提交测试代码的场景，结合其测试代码和测试行为提出多个评估指标，并建立评估体系，对其测试效果进行多维度评估。以平台支持的开发者单元测试、移动应用自动化测试和 Web 应用自动化测试为具体的测试实例类型，选取并调用对应的测试代码分析工具对代码进行分析，提取其代码特性并结合测试行为数据作为测试效果评估的数据支撑。应用测试效果评估指标在不同测试类型上具体体现的评估计算方法，初步得到测试效果在覆盖召回率等七个维度上的原始分。其后将分数标准化并整合，以可视化的方式渲染评估结果供老师和学生直观地查阅，结合各个维度对应的代码分析详情信息以及在历次考试中的评估表现，明确改进加强的方向。

1.2　国内外研究现状

软件测试是为了发现错误而执行程序的过程 [3]，其目的是尽可能地发现和改正被测软件的错误，对保证软件的质量和可靠性具有重要意义。软件可靠性作为软件质量的关键因素，是客户满意度的最基本要素 [4]。Cai 等人 [5] 基于故障检测与软件测试时长及代码被测程度的相关性，提出综合软件测试时间和覆盖率来预测软件可靠性的方法。随着市场和用户对于快速变更软件的需求逐渐上升，软件交付周期变得越来越短。为了在缩短开发时间的同时保证产品的质量，软件测试自身的质量和有效性需要得到一定的保障。

安金霞等人 [6] 提出一种基于多维度测试覆盖率的软件测试动态评价方法来定量地评价软件测试，结合基于代码的覆盖率（如分支覆盖、语句覆盖等）、基于需求的覆盖率（如功能覆盖等）和面向对象的覆盖率（如继承上下文覆盖等）10 多种测试覆盖度量指标，收集测试的监视信息，从软件测试能力、测试效率等多方面对测试动态分析和评价，来保证测试的充分性和有效性。

李军锋等人 [7] 指出软件测试质量评价本身需要快速准确地得出结果，若采用庞大的模型计算，其可行性和资源成本会受到限制。因此他们结合实际测试工作经验，提出一套简易的测试质量评价方法，主要针对于测试文档、抽测结

果、测试充分性和测试效率四个方面的评价，并且每个评价标准都赋予一定的权重，测试质量评分采用单项评分乘以权重的和。但是由于评价内容取值的权重设置方面还不够科学，结合实际项目评价时，评价结果不能准确反映其质量。

软件测试分为手工测试和自动化测试两大类，自动化测试和测试脚本开发现在已经成为软件行业的主流，软件工程师可以根据手工编写或者利用自动化工具生成测试代码。软件测试自动化可以大大降低测试开销，提高测试效率 [8]。Garousi 等人 [9] 提出和生产代码一样，测试代码也需要重视其代码质量，主要体现在两方面，功能方面考察其测试生产代码的正确性，非功能方面考察其是否易于验证和可维护性。

测试用例的设计和编写是软件测试活动中最重要的，是软件测试质量稳定的根本保障，可以帮助开发人员避免引入软件错误。G.Gousios [10] 和 M.Beller [11] 等人指出开发团队需要依靠测试用例的结果和代码审核来决定开发过程中代码是否可合并以及系统是否可以部署。此外，Zhang 等人 [12] 直接提出开发团队的生产效率取决于测试的质量。对此，Dimitrios [13] 等人建立一种评估测试代码质量的方法，受软件质量权威组织 SIG(Software Improvement Group) 质量模型 [14] 的启发，提出三个主要方面：完整性、有效性和可维护性。完整性针对于测试对生产代码的覆盖率，有效性表示测试代码检测缺陷和定位缺陷原因的能力，可维护性反映了测试代码根据生产代码变化进行调整的能力以及测试代码可以作为文档使用的程度。测试代码完整性通过代码覆盖率和断言-McCabe 率来评估，有效性通过断言密度和直接调用率评估，可维护性通过调整 SIG 质量模型评估。其中断言-McCabe 率是计算测试代码的断言个数与生产代码的 McCabe 圈复杂度 [15] 的比值，直接调用率是指覆盖到的生产代码方法被测试代码直接调用的比重，直接与间接调用的区分体现于该方法是否被其他生产代码方法调用。

关于考察测试充分性的指标——代码覆盖率，R. Just [16] 和 Andrews [17] 等人提出相对于其他代码覆盖率标准，变异测试为开发人员提供了更好且可信赖的测试用例有效性的度量，可以体现测试用例的缺陷检测能力。但是变异测试成本开销大，需要生成变异体、编译以及针对每个变异体执行测试套件。随着系统规模的扩大，变异分析的执行也面临着巨大的挑战。对此 Giovanni [18] 等人设计并评估了一个仅基于源代码（包括生产代码和测试代码）的质量指标来评估测试用例有效性的评估模型，通过比对生产和测试代码相关的 67 个因素和测试用例有效性的关系，包括测试代码的代码覆盖率、测试气味、源代码的代码复杂度、可读性等方面的因素，其中代码覆盖率采用 PITest 变异分析工具计算语句覆盖，测试气味通过 Bavota 等人 [19] 提出的分析工具检测，代码复杂度通过 Chidamber 和 Kemerer [20] 提到的面向对象指标套件计算，可读性通过

Scalabrino 等人 [21] 提出的模型分析得到介于 0-1 之间的可读性等级，最终提出较高的语句覆盖率和高质量的生产代码可以区分测试用例有效性的结论。

1.3　本文主要工作

本文主要对接公司软件测试教学平台，设计并实现了针对开发者单元测试、移动应用自动化测试和 Web 应用自动化测试的软件测试多维评估系统，作为对接软件测试教学平台提供评估测试效果服务的子系统，工作主要体现以下几个方面：

（1）设计并提出了软件测试多维评估体系，基于测试代码和测试行为对学生的测试效果进行评估，主要分为 7 个评估指标：覆盖召回率、覆盖准确率、缺陷检测率、测试兼容性、测试运行效率、测试编码效率、测试可维护性。

（2）设计并实现了针对测试代码的分析模块，包括代码分析工具的调研选择以及调用运行机制。由于一次考试涉及的待分析的测试代码的数量大、分析种类多，考虑系统的稳定性以及高可用性，通过 Celery 分布式异步消息队列，实现对分析任务的调度，将对应测试类型的代码分析工具封装为 Docker 镜像，由 Celery Worker 执行单元调用，以 RabbitMQ 作为消息中间件监听消息，利用 Redis 存储任务执行结果。

（3）设计并实现了对测试效果的评估模块，根据不同的测试类型来调用对应的评估脚本，以测试代码分析结果以及提交记录相关数据为输入，根据评估指标计算公式得到多个维度的原始分，由于原始分相对难以看出学生之间评估结果的对比以及个人进步的程度，因此应用数据标准化方法将原始分转换为标准分，使其具有一定的科学性和客观性。

（4）设计并实现了评估结果可视化模块，将上述工作得到的评估结果进行可视化渲染并区分粒度展现。粗粒度对应学生历次考试的评估记录总结性地展现测试效果，主要以测试评估多维雷达总图、多测试类型分布图方式展现。中等粒度对应学生在一场考试中的测试评估结果展现，包括考试得分排名等基础信息、每个试题对应的测试效果多维雷达图以及各维度分数在总体学生中的定位情况。细粒度对应学生在一场考试中的一个试题的某一维度的详情信息展现，如缺陷检测率中变异体杀死和存活的情况以及相应的变异体种类的分布情况等。另外考虑到移动设备的便捷性，利用自适应技术使页面同时支持 PC 端以及移动端设备的查看。总之，系统支持用户直观清晰地查询到测试评估的结果，以便教师提出下一步教学提高计划以及学生针对性地自发学习。

1.4　本文组织结构

本文的组织结构如下：

第一章引言，主要介绍了软件测试多维评估系统的项目背景以及相关研究内容，阐述了软件测试多维评估系统的研发目的和意义。

第二章相关技术综述，主要介绍系统开发过程中使用的相关技术，包括前后端的框架、可视化库及基于不同测试类型的代码分析工具、Docker、Celery 等。

第三章需求分析与设计，前后涉及到对系统需求、系统总体架构设计进行分析以及对功能模块的拆分及初步设计，其中系统需求包括功能性需求、非功能性需求以及对应用户场景的系统用例，总体架构包括系统架构图以及 4+1 视图，功能模块的初步设计包括模块架构图、流程图等。

第四章系统详细设计与实现，在系统需求分析和设计的基础上，详细地介绍测试代码分析模块、评估任务控制模块、测试效果评估模块以及评估结果可视化模块的详细设计与实现，并对设计与实现的合理性进行阐述。

第五章系统测试与案例分析，将系统部署至测试环境，对系统进行功能测试，并基于公司软件测试教学平台的考试实例对学生的测试效果进行评估分析，验证系统的可用性和可靠性。

第六章总结及展望，主要对项目进行总结，并提出研究以及开发过程中存在的问题和改进方向。

第二章　相关技术综述

2.1　前端相关技术

2.1.1　Vue 生态系统

Vue 生态系统是由 Vue.js 框架、Vue-CLI 脚手架等工具、Vuex 状态管理、VueRouter 路由管理等核心插件以及活跃的社区等构成的，最初由尤雨溪开发，后来通过繁荣活跃的社区一起开发构建，不断迭代版本，丰富形成了目前的 Vue 生态系统。

Vue 是一套用于构建用户界面的渐进式 JavaScript 框架。与其它大型框架不同的是，Vue 被设计为可以自底向上逐层应用 [1]，其渲染性能和更新性能相比较同类型的框架而言都是更优的。Vue 的响应式原理主要是对 Vue 实例的 data 选项通过 Object.defineProperty 方法遍历其所有属性，通过设定 getter 和 setter 方法对属性进行数据劫持，在属性被访问和修改时通知监听者 Watcher，监听者来触发组件的重新渲染。Vue 实现了基于组件的体系结构，每个组件作为可复用的 Vue 实例，包含 data、computed、watch、methods 以及生命周期钩子等选项，一个应用以一棵嵌套的组件树的形式组织起来。Vue 框架凭借轻量易上手、灵活开放等特点在前端框架中处于佼佼者的地位，在 Github 社区中相比较 React 和 Angular 等其他前端框架获得更多的 Star。

Vue-CLI 脚手架为 Vue.js 开发的工具，具有功能丰富、易于拓展、即刻创建原型等特点，提供了对 Babel、TypeScript、ESLint、PostCSS、PWA、单元测试和端对端测试开箱即用的支持 [2]，无论是从初始系统的原型设计，还是后来的开发过程，此工具都极大地提高了生产力。

Vuex 为 Vue.js 应用程序特有的状态管理模式，集中式地存储管理组件的状态，主要针对大型单页应用多个组件共享状态的情况，比如用户的登陆状态等。主要涉及到 State、Mutation、Action 等概念，State 即定义的组件间共享的状态，修改定义好的状态的唯一方式是提交 mutation，mutation 必须为同步函数，当涉及到异步操作时，通过分发 action 的方式来触发 mutation 以更改状态。

VueRouter[3] 是 Vue.js 官方的路由管理器，通过 Vue.js 结合 VueRouter 来构建单页面应用，利用定义路由来设定访问路径，不同于后端路由，前端路由主要是

[1] https://cn.vuejs.org/v2/guide/

[2] https://cli.vuejs.org/zh/

[3] https://router.vuejs.org/zh/

描述 URL 和组件之间的关系，通过监听 URL 的变化来渲染对应的组件，以达到切换页面的效果。此外，VueRouter 还提供导航守卫的功能，在涉及到路由变更时，可以通过设置导航守卫来增加限制逻辑处理路由的跳转，主要分为全局前置守卫、全局解析守卫、全局后置钩子、路由独享守卫、组件内守卫等，比如部分页面需在登录的情况下才能跳转，可以添加前置守卫判断跳转的路由是否需要登录，检查用户登录状态，若已登录则顺利进入路由页面，否则限制路由跳转将跳转路由指定为登录页面。系统中在查看评估结果可视化页面时，就配置了导航守卫。以上便是系统前端项目主要使用到的关于 Vue 生态的技术。

2.1.2　Vuetify UI 框架

Vuetify 作为 Vue.js 的头号组件库，该项目包含了由工具和插件、每周发布、大型社区、专业服务和长期支持组成的丰富的生态系统，支持 Chromium、Firefox、Edge、Safari 10+ 等多种浏览器，在配置 polyfill 的情况下还可以支持 IE11 和 Safari 9，为用户提供符合 Material 设计规范、样式美观大方的高质量组件库，Material 设计是一种设计语言，通过技术和科学的创新综合了良好设计的基本原理 [4]，组件库支持开箱即用，提供常用的基本组件，支持经过配置实现一定意义上的自定义，通过查看 API 列表和丰富的使用示例可以轻松上手。此外 UI 框架还支持响应式断点、主体、RTL 等定制化，组件库的样式与项目的设计原型样式高度贴合。

2.1.3　Echarts 库

Echarts 是一款基于 ZRender 的由纯 Javascript 编写的图表和可视化库，包含丰富的图表，支持 Canvas 和 SVG 两种方式渲染，并且提供了丰富的自定义配置选项，能够从全局、系列、数据三个层级去设置数据图形的样式。ZRender 是一个 2D 的矢量库，用于图形元素管理、渲染管理以及事件系统，可以支持多个渲染后端，包括 Canvas、SVG 和 VML。与 DOM 树结构类似的是，在 ZRender 中图形元素如矩形、圆形、文本等也是存储在树形结构中，叶子结点是可显示的，中间节点称为组，负责存储叶子结点的大小、旋转角度和位置等信息，对元素操作就会触发到 ZRender 的渲染更新，每次渲染都会对树结构进行遍历，更新每个节点的变换信息，将可显示对象发送到渲染队列中排序进行渲染。和 DOM 元素不同的是，可显示对象是在 Canvas 画布中画出来的，不会响应到鼠标点击事件，主要通过实现事件系统，来支持检测鼠标事件和事件冒泡，当检测到鼠标事件时，ZRender 遍历渲染器队列中所有可显示的内容，来确定鼠标事件的位置

[4] https://material.io/guidelines/

命中的可显示对象并响应事件。

与其他现有图表库相比，包括 HighCharts、Chart.js 和 C3，四个库都支持常见的图表，比如折线图、条形图和饼状图，HighCharts 通过声明 options 选项和指定数据来初始化图表，相比较 Echarts 可以支持组装需要的组件自由制定数据属性，HighCharts 输入数据的选项和结构受到了严格限制，此外 Charts.js 和 C3 不提供可视化设计的自定义功能，无法生成复杂的可视化效果。Echarts 的可扩展体系结构，包括组件和事件监听的进一步规范，提供了一种灵活的方式来支持图表设计和用户交互的自定义。HighCharts、Chart.js 和 C3 在这点上都不能支持。此外，Echarts 在 ZRender 引擎的支持下，可以使用 HTML5 Canvas 画布来呈现图表，Canvas 画布不涉及到复杂的 DOM 操作，而 SVG 绘图时每个图形都是以 DOM 节点的形式插入到页面中，若可视化比较复杂，会影响到渲染速度，只有 Chart.js 支持画布渲染，其他两个只能操作 SVG。在性能比较中，都分别渲染折线图、条形图和饼状图通过比较图表初始化时间和动画渲染的帧速率，得到 Echarts 和 Chart.js 初始化时间更短的结论，在帧速率测试中也表现良好 [22]。

2.2　Spring Boot

Spring Boot 是由 Pivotal 团队开发的基于 Spring4.0 的开源轻量级框架，相比较 Spring 框架，Spring Boot 极大地简化了应用的初步搭建以及开发过程，提供自动配置的“starter”项目对象模型（POM）来简化 Maven 配置，做到开箱即用；内置 Tomcat、Jetty 容器，支持项目独立运行，无需外部 Servlet 容器；无代码生成，也无需繁琐的 XML 配置；如另外，Spring Boot 还提供了各种非功能性的特性，比如 Actuator 监控应用，可以利用其监视和收集应用的运行情况。总之 Spring Boot 将业内的最佳实践都内置于框架内，提供很好的开发体验，可以让开发者放心地将精力集中于业务逻辑上。

本系统采用 Spring Boot 作为后端项目的框架，为前端以及在线测试教学平台提供可调用的 RestFul 接口，结合 Spring Data 简化对数据库访问的操作，通过添加注解对组件进行标注，严格地对代码逻辑进行分层设计，此外，通过配置日志能够帮助追踪程序问题的位置以及采集程序运行信息等。

2.3　代码分析工具相关

本系统重要功能模块之一就是代码分析，该模块主要是将学生提交的测试代码按照对应的维度调用相应的代码分析工具进行分析，接下来就针对其中几个分析工具进行详细介绍。

2.3.1　PITest

PITest[5] 是一款面向 Java 语言的变异测试工具，由 Henry Coles 等人负责开发和维护。变异测试，也称变异分析，通过在源代码引入变异算子更改源代码模拟缺陷，比如将表示式 $a+b$ 转换成 $a-b$，运行测试用例时由于代码修改应会导致异常行为，根据比对运行结果中变异体杀死的比例来评价测试用例的缺陷检测能力，反映测试用例的质量。PITest 作为一种针对 Java 语言的变异测试工具 [23]，相比较同类型的其他变异工具如 MuJava 和 Major 有以下优点：

（1）工具集成优势，支持与构建工具（如 Maven、Ant）、集成开发环境（如 Eclipse、IntelliJ 等）以及静态代码分析工具（SonarQube）良好集成。

（2）快速高效，将变异操作直接应用于字节码而非源代码，字节码操作在计算上开销不大，变异体生成分为两个阶段，在主控制进程中执行初始扫描，检查所有类并记录可能的变异标识符，标识符由变异体的准确位置和变异运算符的名称构成，只将变异标识符存储至内存，变异体的字节码在扫描过后立即丢弃，因此主进程可以保存数百万个变异标识符，针对每个变异体的测试通过创建子进程完成，并且通过分析不同测试用例在同一变异体上的运行状态差异，最大程度地减少变异运算的执行次数。

（3）测试框架兼容性好，支持测试框架 JUnit 和 TestNG，Delahaye 在对比变异工具兼容性时 PITest 获得兼容性最佳 [24]。

（4）测试报告高可读，清晰地程序语句覆盖和变异测试结果，且突出显示存活的变异体。

本文使用 PITest 工具来对学生提交的测试代码进行变异分析，通过解析测试报告获取变异体杀死和存活的比例，由此作为学生开发者单元测试中缺陷检测率的评估项。

2.3.2　Checkstyle

Checkstyle [25] 初步于 2001 年发布，是一个可帮助程序员编写遵循编码标准的 Java 代码的开发工具，通过其可以自动化检查代码是否符合编码标准，旨在对代码静态分析检查编码风格，目前支持 14 大类、162 个检查项，默认提供 Google/Sun 代码风格规范对应的配置文件，支持高度自定义配置。

Checkstyle 支持多种集成方式，官方发行版提供命令行和 Apache Ant 任务两种方式，通过第三方插件可以与项目管理工具软件 (Maven 等)、开发工具 (IntelliJ IDEA、Eclipse 等)、持续集成工具 (Jenkins 等) 集成，针对团队开发保持编码风格

[5] https://pitest.org/

和代码规范的一致性是非常重要的，这也是代码评审的目的之一，将 Checkstyle 配置于开发过程中使用便可有效地保持团队编码风格的统一。

本文使用 Checkstyle 工具对学生提交的测试代码进行编码风格分析，通过解析结果获取代码规范的违反情况，对测试的可维护性进行评估。

2.3.3　Selenium Grid

Selenium 是 thoughtworks 公司开发的用于 Web 应用程序测试的工具集，包括 WebDriver、IDE 和 Grid。Selenium Grid 作为 Selenium 的三大组件之一，支持分布式的测试执行，即在各个操作系统和浏览器配置上并行多个测试，使用 RemoteWebDriver 与浏览器通信，通过配置 DesiredCapabilities 类指定所需浏览器类型、版本和操作系统类型，达到减少测试脚本执行的总体时间，提高测试效率的效果。Selenium Grid 主要包括两个元素——一个 Hub 中心和多个 Node 节点，Hub 是 Grid 的主节点，作为“中介人”的角色，负责接收所有的测试请求并将其分配到相应的 Node 节点上，确保测试以正确的方式结束，Node 节点负责实际执行测试，可以配置在不同操作系统的多个机器上，Hub 一旦选定节点，测试脚本就会初始化 Selenium 命令，节点就会启动浏览器执行测试脚本。

Selenium Grid 主要应用于两个方面：（1）要针对多个浏览器类型、多个浏览器版本以及在不同操作系统上运行的浏览器运行测试；（2）为了减少测试套件所花费的时间 [26]。前者可以通过配置不同浏览器类型、不同浏览器版本以及不同操作系统的 Node 节点，在测试用例中分别指定待测试的浏览器环境配置，统一通过 RemoteWebDriver 将测试请求发送给 Hub，由其分发给 Node 实例执行测试过程，获取测试结果，是 Web 应用兼容性测试的理想解决方案，对于后者，可以配置相同浏览器类型、版本和操作系统的 Node 节点，同时并行执行测试脚本，从而减少运行时间。

相比较同类型的自动测试工具 Quick Test Professional（简称 QTP），选择 Selenium 的原因有：（1）项目开源；（2）支持丰富的浏览器种类，QTP 只支持 Firefox，Internet Explorer 和 Chrome；（3）支持多操作系统，QTP 只支持 Windows；（4）支持并行执行测试用例，QTP 在不使用 Quality Center 的情况下只能按顺序执行测试。

系统中通过 Selenium Grid 工具联合 Java 反射原理，将学生编写的测试脚本在不同浏览器环境的运行检验其通过情况，考察 Web 应用自动化测试脚本的兼容性、运行效率等特性。

2.4 Celery

Celery 是一个基于 Python 语言开发的简单灵活可靠的分布式系统，根据 BSD 许可发布，支持实时处理和任务调度。其架构主要由三部分组成，消息中间件 (Message Broker)、任务执行单元 (Celery Worker) 和任务执行结果存储 (Task Result Store)。Celery 本身不包含消息服务，需要利用第三方提供的消息中间件来完成任务消息的传输，支持集成 RabbitMQ [6]、Redis [7]和 Amazon SQS [8]等，详情如表 2.1所示。Celery Worker 作为任务执行单元，支持运行在不同的机器上，只需要配置同一个消息中间件即可，Worker 还可以监控一或多个任务队列。任务执行结果存储用来存储执行单元执行的任务结果，支持以不同方式存储任务的结果，包括 redis、MongoDB、SQLALchemy 等。完整处理流程为异步任务由请求触发后，Celery 获取到任务将其发送给消息中间件，执行单元对消息中间件进行监听，一旦监听到消息就执行任务，任务结束完成后将执行结果进行存储。

表 2.1: Celery 支持的消息中间件详情

名称	适配状态	支持监控	支持远程控制
RabbitMQ	稳定	是	是
Redis	稳定	是	是
Amazon SQS	稳定	否	否
Zookeeper	试验阶段	否	否

Celery 的主要应用场景在于：

（1）异步任务，针对用户触发操作响应请求需要长时间操作而用户不需要及时获取处理结果的场景，为了提高用户体验，可以采用异步任务的方式通过 Celery 异步执行，先给出请求反馈，待任务处理完成后再将结果返回给用户，提高应用的整体吞吐量和响应时间。

（2）定时任务，定时执行统计日志、数据备份等任务，Celery 任务队列支持定时触发任务，可以按照时间间隔或者 crontab 表达触发任务。

本文使用 Celery 实现对测试代码分析任务的调度，以 RabbitMQ 为消息中间件，每一种类型的分析工具对应一个消息队列，Celery Worker 负责监听队列，一旦监听到消息就会调用代码分析工具执行分析过程，分析任务的结果存储于 Redis 中。

[6] http://rabbitmq.com/

[7] https://redis.io

[8] https://aws.amazon.com/sqs/

2.5 Docker

Docker 是一个由 Docker Inc 公司团队基于 Go 语言开发的的开源引擎，根据 Apache 2.0 许可发布，提供了一种系统化的方法将 Linux 应用程序置于可移植的容器中实现自动化地部署。Docker 的底层基于 Linux Container（LXC），在其基础上扩展了内核级和应用程序级 API，构建成功能强大的工具集 [27]。LXC 是 Linux 内核的一个特性，将 Linux 进程沙盒化，实现进程之间相互隔离，控制各进程的资源分配。Docker 涉及到几个核心概念——镜像 (Images)、容器 (Container) 和镜像注册中心 (Registry)，内容如下：

Docker 镜像是 Docker 的基础，由松耦合的只读层构成，创建 Docker 镜像时每执行一个操作都对应创建一个新层，每个层为只读状态，除了最底层，其余每一层都有一个指针指向下一层，通过统一文件系统技术将不同的层整合成一个文件系统，具有高度的可移植性，支持共享、存储和更新。可以通过从镜像仓库中拉取镜像或者编写 Dockerfile 的方式构建镜像，Dockerfile 是镜像的描述文件，包含了构建镜像所需要的指令和参数，将应用所有的依赖项、构建工具和包都以脚本代码的形式配置。

Docker 容器使用 Docker 镜像来创建，是镜像的运行实例，可以包含一个或多个正在进行的进程，镜像对应了构建时的结构，而容器可以理解为一种运行时的结构。Docker 容器支持一系列操作命令，这也构成了其生命周期，包括创建、运行、休眠和销毁。Docker 的核心思想就是将应用整合到容器中，在容器实际运行，容器是为应用而生的，可以简化应用的构建、部署和运行过程，这个过程为“应用容器化”，包括编写应用代码、创建 Dockerfile、构建镜像三个步骤，容器化完成，以镜像的方式交付以容器的方式运行。

Docker 镜像注册中心包括公共和私有两种类型，Docker Hub 是官方提供维护的公共镜像注册中心，用于管理镜像仓库，支持用户推送拉取镜像 [28]，当涉及到要存储私有镜像时，可以通过 Docker Registry 搭建私有仓库来存储和管理内部的镜像，仅与团队或组织的其他成员共享镜像。在企业内部通常使用 Harbor 来对内部的镜像进行管理，Harbor 是一个开源的容器镜像注册中心，可以通过角色的访问控制来设置镜像的访问权限，扫描镜像中的漏洞，支持对授信镜像进行标注，旨在更安全地管理 Docker 镜像。

系统中通过容器化将封装的测试代码分析工具构建成 Docker 镜像，支持 start、stop 和 status 方法，方便分析任务执行过程中调用，镜像通过 Harbor 镜像注册中心统一管理。此外，系统前后端项目也利用 Dockerfile 进行镜像打包，支持快速部署便于后续的持续集成。

2.6　本章小结

本章主要介绍系统开发实现过程中使用到的关键技术，采用前后端分离的架构实现来降低页面和功能之间的耦合性，前端页面通过 Axios 调用后端的 RESTful 接口并以结构化的 JSON 数据进行交互。页面显示主要通过 Vue 前端生态，测试评估结果数据通过 Echarts 库进行可视化，服务器端采用 Spring Boot 框架，测试代码特性通过相应的代码分析工具提取，执行测试代码分析任务过程中运用了 Celery 和 Docker 等技术，系统基于这些技术和工具建立上层功能。

第三章　软件测试多维评估系统的需求分析与设计

3.1　系统概述

本文阐述的软件测试多维评估系统主要针对学生于在线测试教学平台上提交测试代码的场景，面向开发者单元测试、移动应用自动化测试以及 Web 应用自动化测试三种主流测试类型，对学生提交的测试代码进行多方面地分析，再根据提出的软件测试评估指标体系结合学生的测试行为和代码分析结果数据两方面综合考察，来对学生编写的测试代码产生的效果进行多维度评估，充分直观展现评估结果，起到教学反馈的作用。

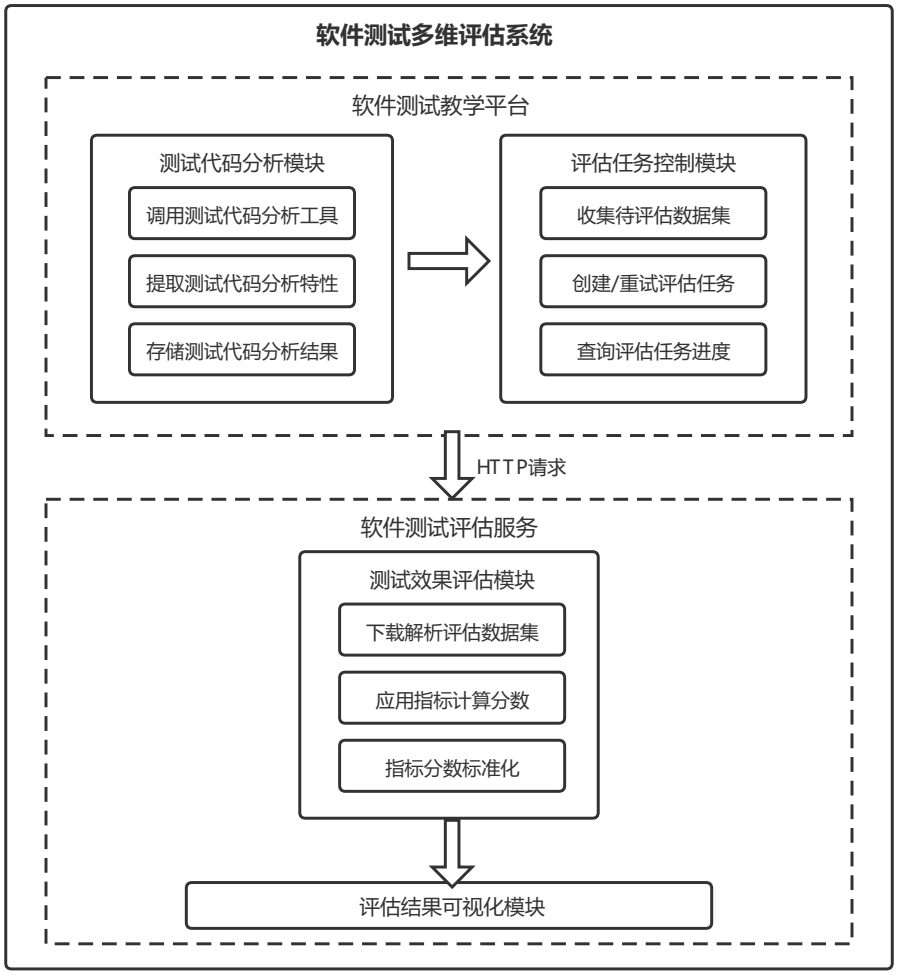

图 3.1: 软件测试多维评估系统模块图

本系统的模块图如图 3.1所示，根据需求描述和目标，从功能上看，软件测试多维评估系统可以分为测试代码分析模块、评估任务控制模块、测试效果评估模块以及评估结果可视化模块，这四个功能模块分别负责提取测试代码的维度特性、对接公司软件测试教学平台触发测试评估、制定评估指标体系并根据标准评估测试效果以及将评估结果数据可视化渲染展现等功能。考虑到教学平台本身在算分机制也需要涉及到测试代码分析的逻辑，本着代码复用的原则，将测试代码分析功能模块于教学平台服务端实现。

从具体业务交互流程上看，教师在测试教学平台上组织考察学生测试效果的考试，在学生提交测试代码后，管理员可以对当前考试的测试代码发起分析任务，通过查询当前考试的类型调用相应的代码分析工具，将分析结果保存在数据库中。教师点击测试评估按钮发起评估请求，教学平台响应请求收集待评估的数据集，即考试、试题、参加考试的学生的基本信息、学生提交记录以及对应的覆盖率得分等数据，并调用本系统的创建评估任务接口，创建评估任务成功后执行评估流程，在评估任务执行的过程中，教学平台通过轮询评估任务进度的查询接口实时反馈给教师评估进度。同样地，对于已经评估过的考试，教师可以点击重新评估按钮发起重估请求，等到评估任务结束，教师和学生可以分别在考试结果页面中点击评估结果入口的图标，以多维图的形式直观地查看到测试效果以及各维度指标得分的分布情况，点击各个维度查看对应的分析详情信息。此外，教师和学生可以点击导航栏的历史记录按钮，分别查看历史评估记录及对应记录的测试效果变化趋势图，点击首页查看评估效果评估的整体数据等，通过提供全面系统的评估结果达到教学反馈的作用，以便后续教师教学质量的提高以及学生测试能力的提升。

3.2　系统需求分析

系统需求分析是软件设计开发中重要的一环，本节分别从功能性需求分析、非功能性需求分析以及系统用例图详细介绍。

3.2.1　功能性需求分析

针对功能模块的划分，本系统的功能性需求粗粒度划分可以包括测试代码分析、评估任务控制、测试效果评估以及评估结果可视化查看，但是考虑到评估过程的复杂性以及可控性，还需进一步拆分。

测试代码分析模块的功能性需求列表如表 3.1所示，由教学平台的前端和服务端负责实现创建、执行、停止、重试测试代码分析任务，执行分析任务的实质是调用相应的分析工具，多维度地考察测试代码特性，分析任务完成后，将分析

得到的测试代码特性保存至数据库中，为后续测试效果评估提供数据支持。

表 3.1: 测试代码分析模块功能性需求列表

需求编号	需求名称	需求内容
R1	创建测试代码分析任务	管理员可以在学生提交测试代码后，点击信息管理中的计算分析按钮，查询到当前考试并选择代码分析的类型，触发测试代码分析过程
R2	查看测试代码分析进度	管理员可以在点击分析按钮后，进入测试代码分析进度页面，实时查看到代码分析进度
R3	停止测试代码分析任务	管理员可以在测试代码分析进度页面，点击停止按钮，停止测试代码分析任务
R4	重试测试代码分析任务	管理员可以在测试代码分析进度页面，点击重试按钮，重新执行测试代码分析任务

评估任务控制模块的功能性需求列表如表 3.2所示，页面交互方面在教学平台前端提供测试评估与重新评估的按钮，以及查询评估任务进度的进度条，具体实现逻辑涉及到教学平台服务端与软件测试评估服务的服务端的对接，软件测试评估服务的服务端分别提供创建、重试评估任务以及查询评估进度的 RestFul 接口支持调用，教学平台在调用创建与重试接口时负责查询收集待评估的数据集，即考试、试题、参加考试的学生的基本信息、学生提交记录以及对应的覆盖率得分等数据，将数据集上传至 OSS 服务器得到的 OSS URL 地址作为请求的参数。

表 3.2: 评估任务控制模块功能性需求列表

需求编号	需求名称	需求内容
R5	创建测试评估任务	教师可以在学生提交测试代码后，点击测试评估按钮，触发测试评估过程
R6	重试测试评估任务	教师可以对已经进行过测试评估的考试，通过点击重新评估按钮，重新评估测试效果
R7	查看评估任务进度	教师可以通过进度条的形式实时查看到评估任务的进度
R8	收集待评估的数据集	教学平台服务端对被发起评估任务的考试的相关数据进行查询收集传递至测试评估服务端

测试效果评估模块的功能性需求列表如表 3.3所示，主要包括下载并解析待评估的数据集，以及根据当前的测试类型调用相应的评估脚本，对学生的测试代码分析数据以及测试行为数据进行整合、预处理，根据指标公式计算得到原

始指标分数，再应用数学归一化方法将分数标准化，完成后将评估结果数据保存至数据库中。

表 3.3: 测试效果评估模块功能性需求列表

需求编号	需求名称	需求内容
R9	下载并解析待评估的数据集	下载并解析待评估的数据集，结合测试代码分析结果数据作为评估脚本的输入
R10	评估测试效果	系统根据考试的测试类型不同，调用相应的评估脚本，数据预处理后应用软件测试评估指标公式得到指标原始得分，应用数学归一化方法将分数标准化
R11	保存评估结果	系统将评估得到的测试评估结果保存至数据库中

评估结果可视化模块的功能性需求列表如表 3.4所示，主要包括对前序评估得到的结果数据进行可视化渲染，可以支持教师和学生查看其评估的结果，包括个人信息、考试排名相关数据以及测试效果多维图和对应的维度分析详情。

表 3.4: 评估结果可视化模块功能性需求列表

需求编号	需求名称	需求内容
R12	查看评估可视化结果	教师和学生可以在考试结果页面中通过点击评估结果入口的图标，查看评估结果的可视化页面，包括个人信息及考试排名、成绩分布、测试效果多维图以及各个维度分析的详情信息
R13	查看历史评估记录	教师和学生可以点击导航栏的历史记录按钮，查看测试效果评估的历史记录及与之对应的变化趋势图
R14	查看测试评估数据总览	教师和学生可以点击导航栏的首页按钮，查看不同测试类型的测试效果评估总体数据的可视化图表

3.2.2　非功能性需求分析

软件测试多维评估系统的主要目标是帮助教师以及学生对其测试效果进行多维度的评估，通过提供全面系统的评估结果达到教学反馈的作用，以便后续教师教学质量的提高以及学生测试能力的提升。系统在满足基本功能的前提下，还需要对系统的易用性、容错性、扩展性提出非功能性需求。

易用性：系统需要对用户体验方面提高重视，提供良好的人机交互，比如用户操作之后及时给出提示，提高响应速度减少用户等待的时间，尽量简化用户评估的操作。

容错性：系统需要提供较完善的错误处理机制，保证评估过程的完整性，应对评估数据量庞大的情况，系统也要相对稳定地运行。

扩展性：系统需要设计良好的接口模块，方便后续可以通过配置在不涉及重大改动的情况下扩展评估类型的情况。

3.2.3　系统用例设计

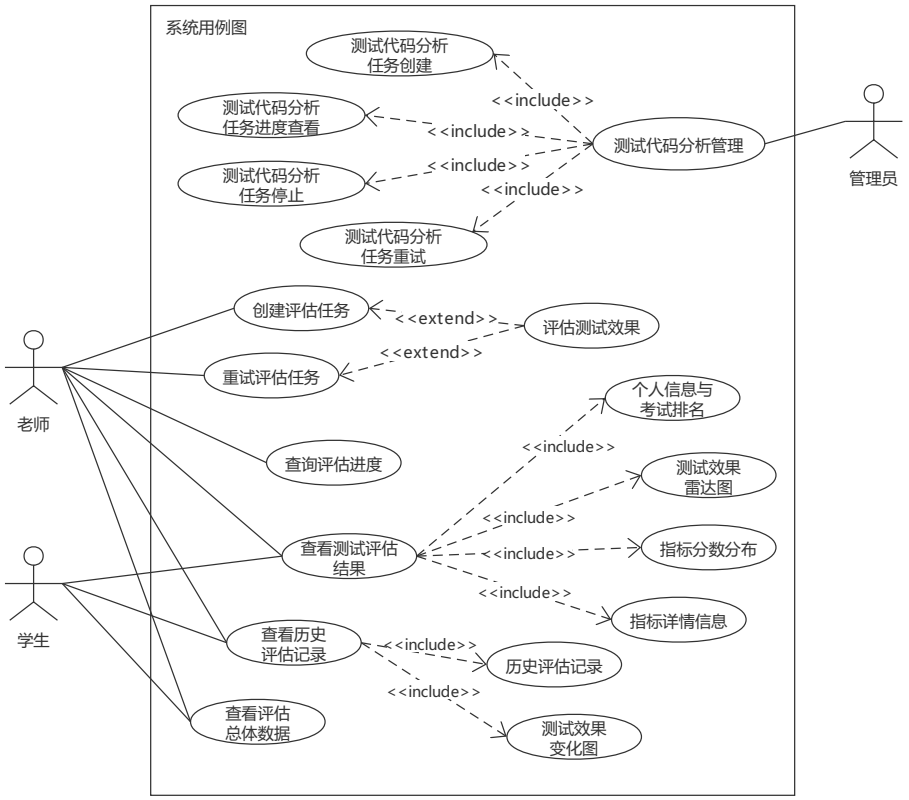

图 3.2: 系统用例图

软件测试多维评估系统用例图如图 3.2所示，描述了管理员、教师和学生在本系统中的所有用例，其中评估测试效果用例是创建评估任务和重试评估任务用例的扩展。管理员拥有测试代码分析管理的功能，包括测试代码分析任务的创建、进度查看、停止以及重试等功能；教师用户可以对评估任务进行创建、评估以及进度查询；此外教师用户和学生用户都可以对评估结果、历史评估记录以及评估总体数据进行查看。

下面以各功能模块为主体进行分析，分别详细描述每个用例，包括用例名称、参与者、用例描述、前置条件、后置条件、正常流程以及扩展流程。

测试代码分析模块对应着管理员的测试代码分析管理功能，主要负责根据测试类型的不同，通过执行相应的代码分析任务，即调用不同的代码分析工具

对学生提交的测试代码进行分析，提取能够评估其测试效果的维度特征数据，并将分析结果保存至数据库中。其用例描述如表 3.5所示。

表 3.5: 测试代码分析管理用例描述

ID	**UC1**
名称	测试代码分析管理
参与者	管理员
用例描述	管理员用户对应当前考试的测试类型创建并执行测试代码分析任务
前置条件	用户登录并进入系统，学生提交测试代码
后置条件	无
正常流程	1. 用户点击侧边栏中信息管理下的计算分析按钮，进入计算分析页面 2. 用户输入待分析的考试 ID 并查询，页面显示对应考试 ID 的考试试题列表 3. 用户在考试试题列表中点击分析按钮，进入测试代码分析页面，系统执行分析操作 4. 用户在测试代码分析页面可以查看任务总进度以及每个学生测试代码分析任务的进度 5. 用户点击停止按钮 6. 系统提示“正在停止”，停止分析任务，进度条保持不动 7. 用户点击重试按钮 8. 系统提示“正在重试”，重新执行分析任务，进度条从头开始
异常流程	2a. 用户输入不存在的考试 ID，系统提示“请重新输入正确的考试 ID” 6a. 系统请求停止分析任务失败，页面给出“停止失败”的提示 8a. 系统请求重试分析任务失败，页面给出“重试失败”的提示

评估任务控制模块是整个系统中软件测试评估服务与教学平台对接的中间模块，创建评估任务、重试评估任务以及查询评估进度分别对应平台的测试评估、重新评估以及查看评估进度的用户交互操作，可以细分为以下几个功能点。

（1）创建评估任务用例描述如表 3.6所示。

（2）重试评估任务用例描述如表 3.7所示。

（3）查询评估进度用例描述如表 3.8所示。

测试效果评估模块作为系统的核心模块，负责制定评估标准，下载并解析待评估的数据集，针对不同的测试类型，调用不同的评估脚本，对数据进行预处

表 3.6: 创建评估任务用例描述

ID	**UC2**
名称	创建评估任务
参与者	教师
用例描述	用户对参与当前考试学生的测试效果创建评估任务
前置条件	用户登录进入系统，学生提交测试代码
后置条件	用户可以查询评估进度
正常流程	1. 用户在考试结果页面点击测试评估按钮 2. 系统根据当前考试 ID 查询是否有对应的评估任务，若无，则创建评估任务 3. 系统在创建评估任务后开始异步执行评估过程，评估任务的状态更新为 “开始执行”，页面提示评估任务创建成功
异常流程	1a. 当前考试不符合系统支持评估的测试类型，页面不会出现测试评估按钮

理、评估并根据评估标准生成表示学生测试效果的数据。其用例描述如表 3.9所示，模块包含了以下功能点：

表 3.8: 查询评估进度用例描述

ID	**UC4**
名称	查询评估进度
参与者	教师
用例描述	用户查询评估测试效果的进度
前置条件	用户登录进入系统，点击测试评估或重新评估按钮
后置条件	用户可以查看评估结果
正常流程	1. 页面实时展示评估任务的进度条 2. 评估任务成功或者失败时弹出评估任务完成的提示框，失败时会罗列失败原因，点击确认，提示框关闭 3. 评估任务成功后，对于教师，可以看到学生列表中支持跳转到评估结果可视化页面的图标呈高亮状态，对于学生，可以看到考试结果页面增加查看评估结果的按钮
异常流程	无

表 3.7: 重试评估任务用例描述

ID	**UC3**
名称	重试评估任务
参与者	教师
用例描述	用户对参与当前考试的学生的测试效果重新评估
前置条件	用户登录进入系统，考试已经被评估过
后置条件	用户可以查询评估进度
正常流程	1. 用户在考试结果页面点击重新评估按钮 2. 系统根据当前考试 ID 查询对应的评估任务，重新异步执行评估过程，评估任务的状态更新为“开始执行”，页面提示重新评估成功
异常流程	无

表 3.9: 评估测试效果用例描述

ID	**UC5**
名称	评估测试效果
参与者	测试效果评估模块
用例描述	根据测试类型调用相应的评估脚本评估
前置条件	创建或者重试评估任务成功
后置条件	评估结果可视化模块渲染评估结果数据
正常流程	1. 下载待评估的数据集，更新评估任务的状态为“下载成功” 2. 解析数据集，更新评估任务的状态为“解析成功”，数据集作为评估脚本的输入 3. 调用对应当前测试类型的评估脚本，对解析的数据集结合查询的测试代码分析结果数据进行预处理，应用指标公式得到表示学生测试效果的原始分 4. 应用数学归一化方法将原始分数标准化，并将其保存至数据库中，并更新评估任务的状态为“评估测试效果成功”
异常流程	1a. 由于网络异常问题，下载待评估的数据集失败，评估任务的状态更改为“失败”，失败原因更新为“下载失败”，评估执行结束，提示异常信息 3a. 评估脚本执行失败，评估任务的状态更改为“失败”，失败原因更新为“评估测试效果失败”，评估执行结束，提示异常信息

（1）下载并解析待评估的数据集，作为评估脚本的输入。

（2）定义软件测试多维指标体系，根据测试类型的特点，分别提出软件测试评估指标，旨在多维度地对测试效果进行评估。

（3）评估测试效果生成结果数据，根据测试类型的不同，调用相应的评估脚本对解析的数据集进行预处理并评估得到可以表示学生测试效果的原始分。

（4）测试效果分数标准化，应用数学归一化方法将原始分数标准化。

表 3.10: 查看评估可视化结果用例描述

ID	**UC6**
名称	查看评估可视化结果
参与者	教师、学生
用例描述	对于教师可以通过点击学生列表中的评估结果入口图标查看评估结果可视化页面，对于学生，可以在考试结果页面点击查看评估结果按钮查看
前置条件	用户登录进入系统，评估执行模块执行成功
后置条件	无
正常流程	1. 教师点击学生列表中的评估结果入口图标，学生点击查看评估结果按钮 2. 页面跳转至评估结果可视化界面，可以查看到个人信息及考试排名、测试效果雷达图以及各个指标分数的分布图
异常流程	无

表 3.11: 查看历史评估记录用例描述

ID	**UC7**
名称	查看历史评估记录
参与者	教师、学生
用例描述	教师和学生可以点击评估结果页面导航栏的历史记录按钮，查看历史评估记录以及对应记录的测试效果变化趋势图
前置条件	用户登录进入系统
后置条件	无
正常流程	1. 教师和学生点击导航栏的历史记录按钮 2. 页面跳转至历史评估记录界面，查看历史测试考试的评估记录以及对应记录的测试效果变化趋势图
异常流程	无

评估结果可视化模块负责将评估结果相关数据可视化渲染展现，可以细分为以下几个功能点。

（1）查看评估可视化结果，其用例描述如表 3.10所示。

（2）查看历史评估记录，其用例描述如表 3.11所示。

（3）查看测试评估数据总览，其用例描述如表 3.12所示。

表 3.12: 查看测试评估数据总览用例描述

ID	UC8
名称	查看测试评估数据总览
参与者	教师、学生
用例描述	教师和学生可以点击评估结果页面导航栏的首页按钮，查看整合不同测试类型的测试效果评估总体数据的可视化图表
前置条件	用户登录进入系统
后置条件	无
正常流程	1. 教师和学生点击导航栏的首页按钮 2. 页面跳转至测试评估数据总览界面，查看整合不同测试类型的测试效果评估总体数据的可视化图表
异常流程	无

3.3　系统总体设计

3.3.1　系统架构设计

软件测试多维评估系统架构图如图3.3所示，分别为展示交互层、应用服务层、服务中间层、工具层以及数据存储层。整个系统采用前后端分离的方式开发，将前后端解耦，顺应互联网开发业界标准。下面就每一层依次具体介绍一下系统架构。

展示交互层，也就是项目前端，在权衡比较主流的前端框架后，凭借轻量易上手的特点选择并采用 Vue.js 生态，充分利用 Vue-cli 脚手架的功能进行项目的快速原型开发以及后期项目的部署配置，通过 Vue-router 来实现页面的路由。Vuetify UI 库作为 Vue.js 的头号组件库，且样式简洁大方，和原型设计图风格相符，因此选择其作为项目的 UI 框架。考虑到评估结果可视化的用户交互场景要求不高以及涉及页面渲染成本开销的问题，这里选择 Echarts 库来实现数据的可视化。系统前后端通过 Nginx 实现负载均衡，保证系统的性能以及稳定性。

应用服务层，采用 Spring Boot 框架，软件测试教学平台支持教师组织创建考试来考察学生对测试知识的掌握程度以及测试实践能力的高低，支持学生在线编写测试代码并提供阅卷打分的功能，本系统作为教学平台的子系统，主要负责针对学生在教学平台上的测试代码和测试行为数据对其测试效果进行评估的相关功能，主要涉及到三个功能模块：测试代码分析模块、评估任务控制模块以及测试效果评估模块。应用场景为学生们提交测试代码后，由管理员执行测试代码分析任务，分析结果为测试效果评估提供数据支持。由于一场考试涉及到大量学生的测试代码，考虑到系统的高效性，这里采用以 RabbitMQ 为消

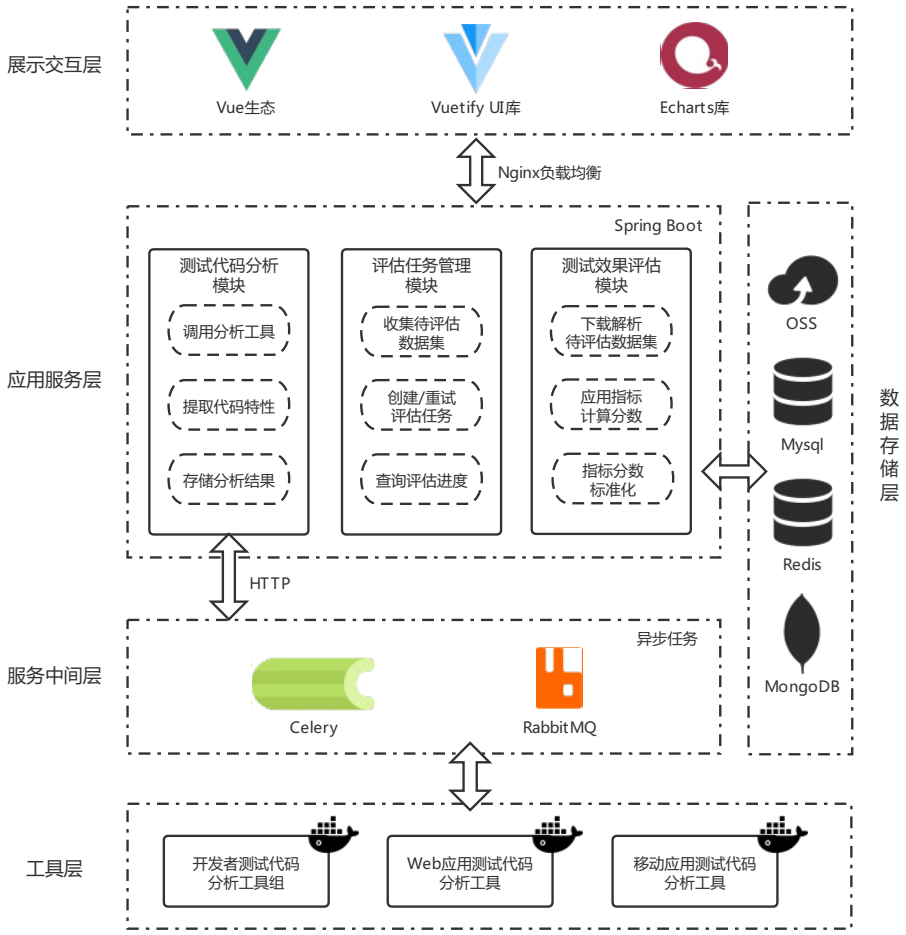

图 3.3: 系统架构图

息中间件的 Celery 分布式异步任务调用的方式，也就对应了系统架构图的服务中间层和工具层，查询 Mysql 数据库中的提交记录获取测试代码的存储地址作为异步任务的参数传给 Celery Worker，将工具打包成 Docker 镜像，通过 Celery Worker 拉取镜像以 Docker 容器的形式执行分析任务，代码分析完成后分析结果存储在 Redis 中，在 Celery 回调中通过 RabbitMQ 发送任务完成的消息，服务端监听 RabbitMQ 消息队列，一旦收到任务完成的消息查询分析结果进行解析并写入 MongoDB 中，作为后续以及教学平台其他服务的数据支持。由教师发起对学生的测试评估任务，评估任务控制模块包括评估任务的创建、评估进度的查询以及评估任务的重试，都以 RestFul 接口的形式支持教学平台调用；测试效果评估模块主要功能是根据不同的考试类型调用相应的 Python 评估脚本，结合考试原始数据以及测试代码特性数据进行数据预处理操作，依据相应的评估指标生成测试效果原始分数，获取到原始分后，应用数学归一化方法将原始分数标准

化并生成评估结果存储到 MongoDB 中，以便评估结果可视化页面的渲染生成。

数据存储层中，关于 Redis、Mysql 以及 MongoDB 数据库的应用在前文均有提及，OSS（对象存储）是阿里云对外提供的海量、安全和高可靠的云存储服务，这里主要用于存储学生考试或练习提交的代码，以及触发评估任务时教学平台和本系统对接收集的待评估的数据集文件。

3.3.2　系统 4+1 视图

本节将分别从逻辑视图、开发视图、进程视图、物理视图以及场景视图五个视角详细介绍本系统的总体架构，其中场景视图对应了前文 UML 描述的用例图，如图 3.2所示。

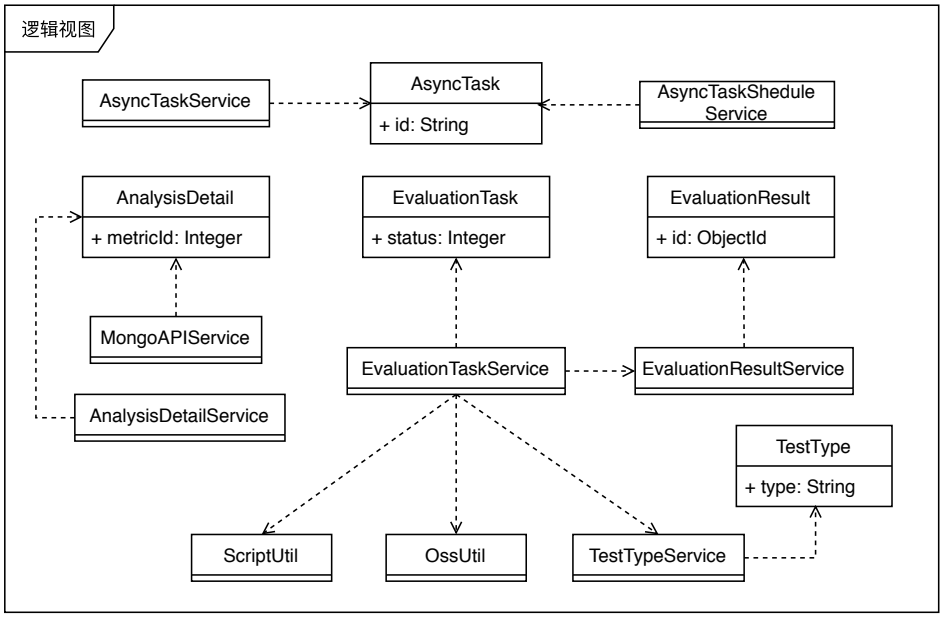

图 3.4: 逻辑视图

首先是逻辑视图，如图 3.4所示，逻辑视图是以最终用户为视角，描述系统提供给用户的功能和服务。其中 AsyncTaskService 负责提供异步分析任务的保存、更新、查询以及重试，AsyncScheduleService 负责管理异步任务具体执行过程，包括开始、暂停、重试以及获取结果等，通过调用 Celery 服务的接口将分析任务分配给 Celery Worker 实际执行分析测试代码的过程。EvaluationTaskService 负责提供评估任务的创建、重试以及查询进度，OssUtil 提供下载存储在 OSS 服务器上文件的功能，ScriptUtil 提供调用 Python 评估脚本的服务，EvaluationResultService 提供测试评估结果存储、查询以及历史评估记录信息查询等功能，MongoAPIService 和 AnalysisDetailService 分别负责对测试代码分析结果的保存更新和查询。

其次是开发视图，如图 3.5所示，开发视图是以开发人员的角度对系统的代码组织架构进行描述的产物。系统可以分为前端 UI、服务端业务逻辑以及技术服务，前端 UI 通过组件文件、Less、JavaScript 以及 Echarts 来构建用户交互界面，服务端业务逻辑根据功能和层级来划分，主要划分为 Controller 包、Logic 包、Service 包、Util 包，Controller 包为控制器管理，主要管理对接外部的 URL 请求，Logic 包对应业务逻辑功能，由 Controller 调用，Service 包负责为 Logic 包提供服务，主要调用实现数据增删改查的 Repository 包，其中 Service 包和 Logic 包中的代码通过接口和实现类的形式暴露功能，实现松耦合。技术服务主要是系统使用的基础服务，包括 Spring Data JPA、日志 Log4J2 等。

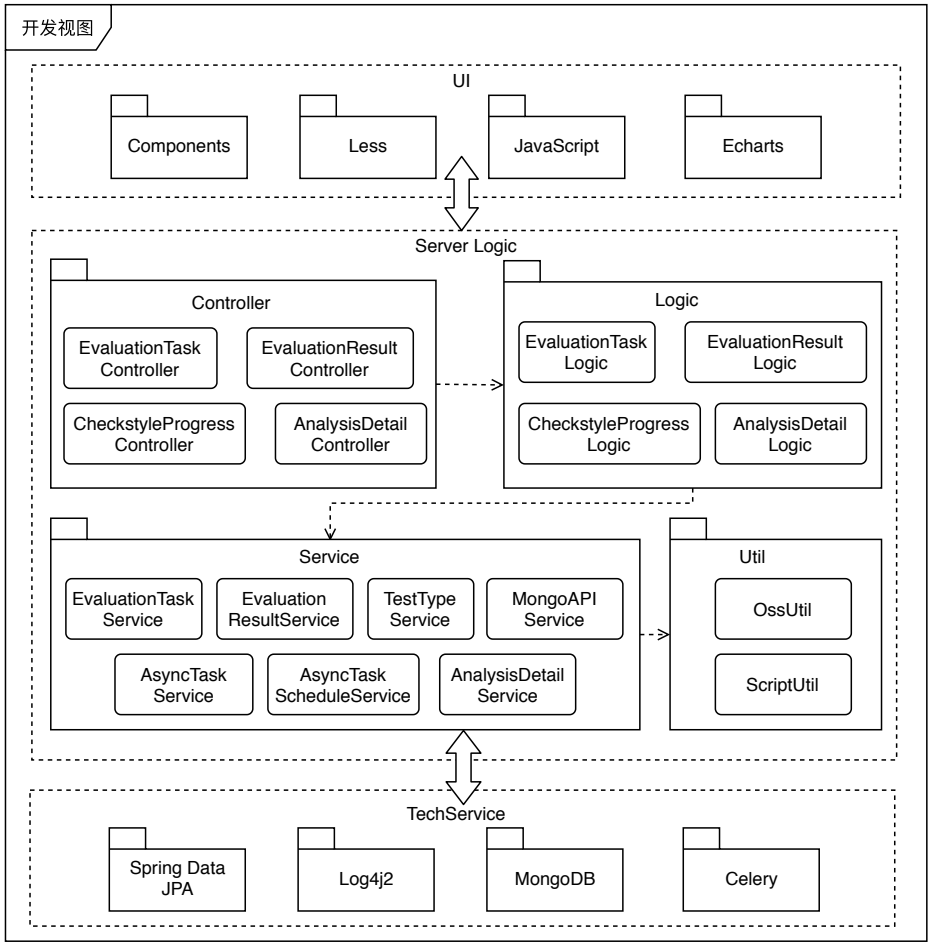

图 3.5: 开发视图

再次是进程视图，如图 3.6所示，进程视图是以系统集成者的视角描述系统的进程与通信，强调系统的并发行、分布性、容错性以及系统集成性，当系统开始执行测试代码分析任务时，主线程调用 Celery 的开始任务接口得到请求响

应信息，任务执行完成后分析结果存储于 Redis，并将任务完成的消息发送至 RabbitMQ 中的结果队列，RabbitListener 监听器线程监听到任务完成的消息，则向 Celery 查询分析结果将其存储至 MongoDB 中便于后续使用。触发测试评估任务时，主线程会启动新线程异步执行评估任务，评估完成后将评估结果存储至 MongoDB 并更新评估任务的状态。在这期间，系统支持查询评估任务进度请求的轮询，将进度实时反馈。

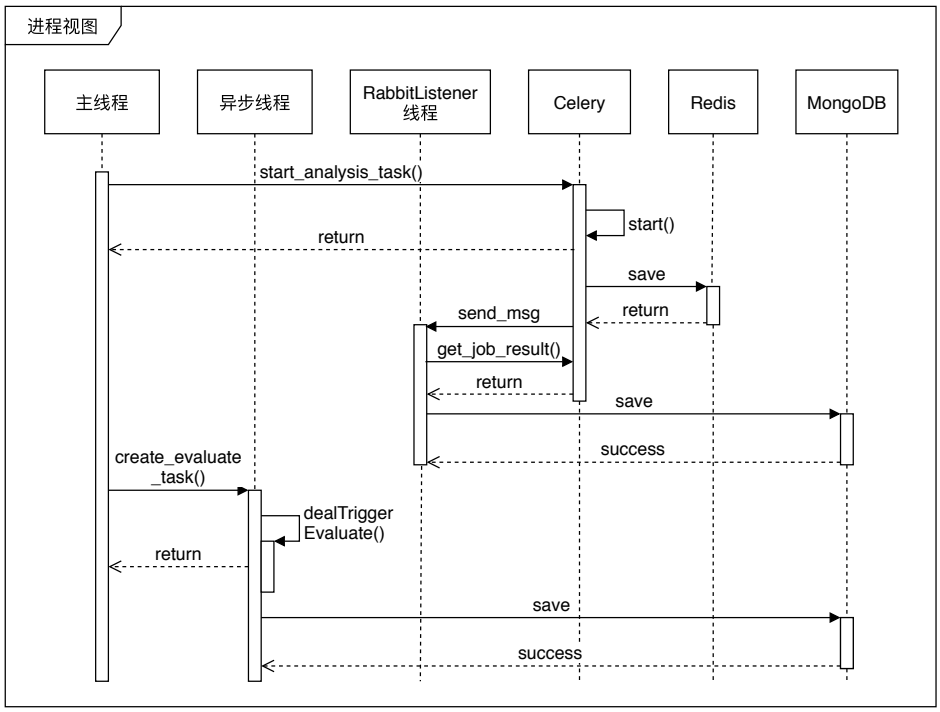

图 3.6: 进程视图

最后是物理视图，如图 3.7所示，物理视图是面向运维人员的视角对系统物理节点件的通信与部署情况的描述，主要考虑软件与硬件的映射问题以及系统的性能、规模以及可靠性等。前端 UI 项目利用 Webpack 构建打包为 dist 文件夹，将其部署于 Nginx 服务器，通过配置 Nginx 服务器作为反向代理，用于转发前端的请求，并解决系统前后端系统的跨域问题。系统后端项目打成 Jar 包单独部署在服务器上，系统涉及到对测试代码进行分析，测试代码文件存储在阿里云对象存储（OSS）服务器中，采用 Celery 分布式系统对分析任务进行异步调度，分配给 Celery Worker 任务执行单元来执行分析过程，Celery Worker 支持分散部署在不同服务器中，提高系统的稳定性。通过 RabbitMQ 作为消息中间件，分析结果存储在 Redis 服务器中，评估结果数据存储在 MongoDB 中。系统前后端均使用 Docker 进行容器化，通过配置 Jenkins 的 pipeline 实现自动化部署。

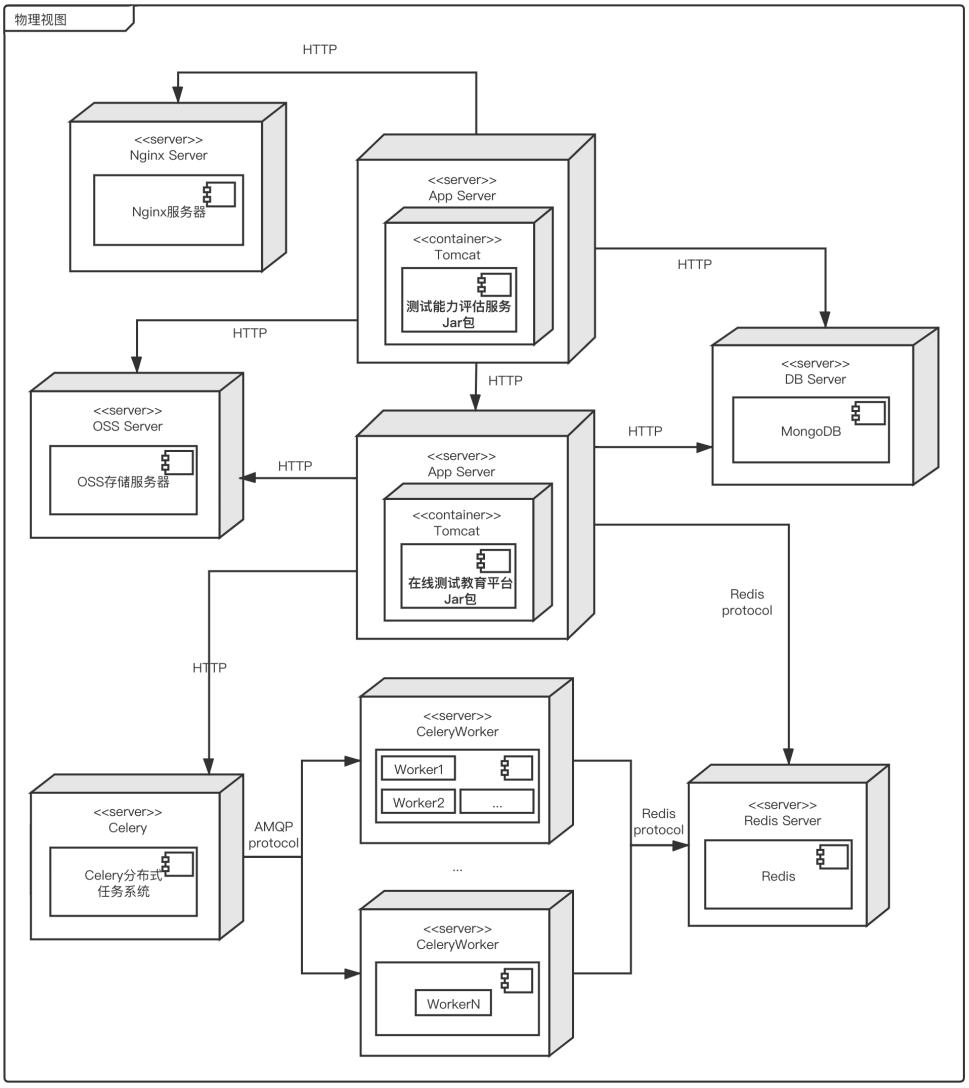

图 3.7: 物理视图

3.4　测试代码分析模块设计

3.4.1　架构设计

测试代码分析模块架构图如图 3.8所示，本模块主要负责执行测试代码分析任务如运行效率、编码风格分析等，提取可以表示代码特性的详情信息，作为测试效果评估模块的数据支撑，根据测试类型的不同，前期调研并选择了相应的代码分析工具对代码进行多方面地分析。考虑到测试代码分析结果可以提供给教学平台的其他服务使用，因此将这部分功能模块物理上放到教学平台的服务端实现。Carver 等人 [29] 通过实验证明，在借助代码覆盖工具的情况下编写

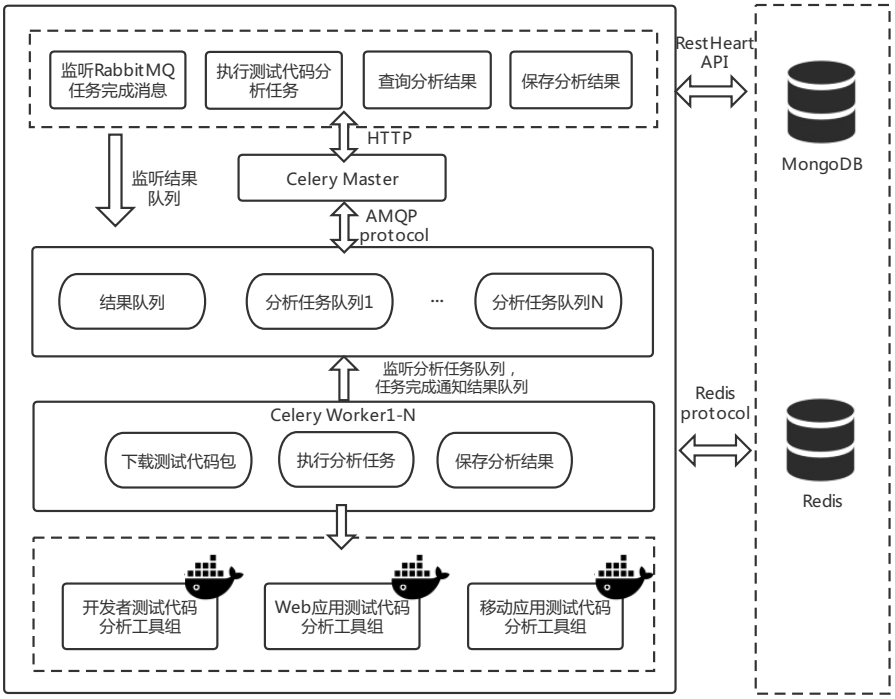

图 3.8: 测试代码分析模块架构图

测试用例，代码覆盖率会显著提高。为了充分发掘学生在代码覆盖方面的水平，在考试过程中，每提交一次测试代码，教学平台都会计算覆盖得分实时反馈。对此覆盖分析直接选取考试过程中的分析数据，其他维度的分析在考试结束后对学生最后一次提交的测试代码发起分析任务。

整个分析过程以异步任务的形式由 Celery 分布式任务队列分发给 Celery Worker 任务执行单元来执行，以 RabbitMQ 作为消息中间件，Redis 负责存储分析结果，Celery Worker 通过监听 RabbitMQ 中的分析任务队列，一旦监听到则执行分析任务，分析任务具体表现为下载学生的测试代码包，调用分析工具执行测试代码分析的过程，分析完成将结果存储至 Redis。在 Celery 回调逻辑中将分析任务完成的消息通过 RabbitMQ 发到结果队列，服务端负责监听 RabbitMQ 的结果队列，一旦收到任务完成的消息查询分析结果并解析，通过 RestHeart 的 RestFul 接口存储到 MongoDB 中，方便后续以及教学平台的其他服务使用。

这里提及的代码分析工具是对第二章提及的常用代码分析工具的封装，比如对代码编码风格分析的工具是对 Checkstyle 的封装，在原有工具的基础上增加了对扫描结果的解析与汇总，最后以 JSON 的格式给出测试代码违反代码风格的规则名、与之对应的个数以及代码行数等。分析工具统一打包成 Docker 镜

像供 Celery Worker 拉取调用，凭借 Docker 部署启动快、隔离性、可移植性以及支持版本控制的特性，保证了工具执行过程的高效、不受干扰，部署环境的可移植以及后期维护的便捷。

3.4.2　流程设计

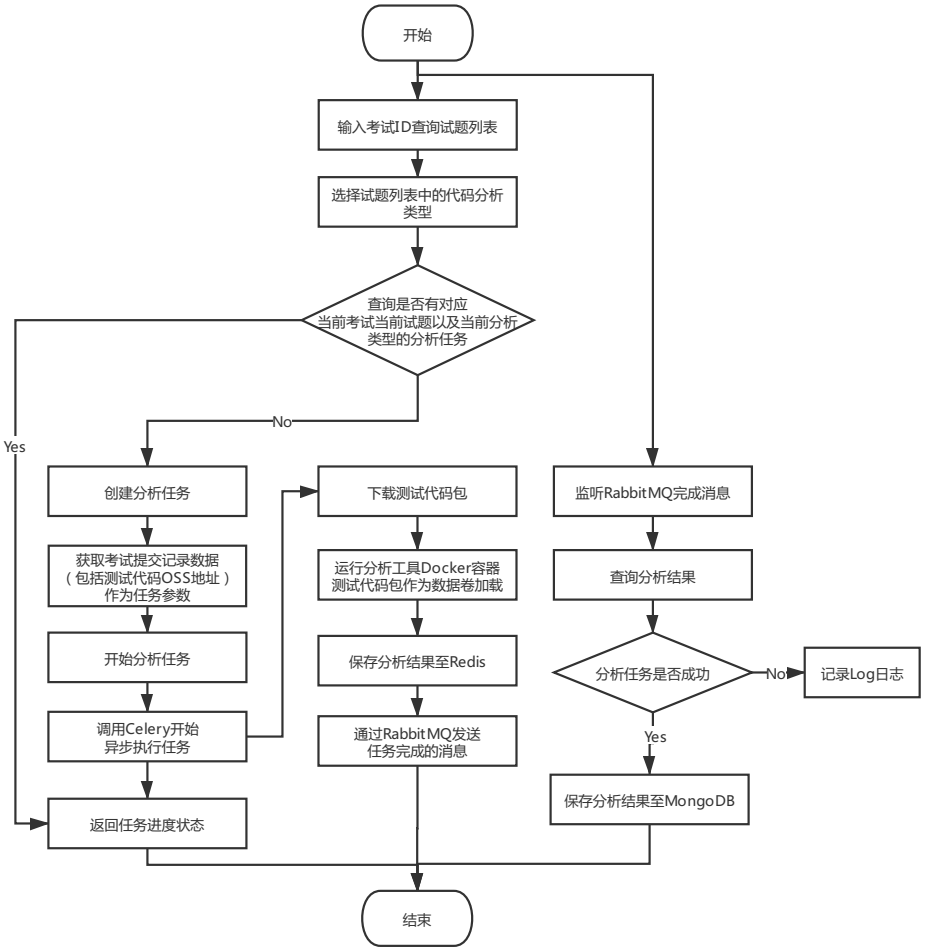

图 3.9: 测试代码分析模块流程图

测试代码分析模块流程图如图 3.9所示，在学生提交过测试代码后由管理员触发分析行为，输入待分析的考试 id 查询到对应考试的试题列表，选择试题列表中后面的分析类型，便创建了相应的分析任务并进入分析执行页面，可以查看到分析进度。首先查询是否有对应当前考试当前试题以及当前分析类型的分析任务，若有直接返回任务进度的状态，若没有则创建分析任务，其次收集获取学生在考试过程中提交测试代码记录数据，包括测试代码上传的 OSS 地址作为任务的参数，其后调用 Celery 开始异步调度分析任务，Celery Worker 负责下

载测试代码包，运行分析工具 Docker 容器，将下载的测试代码包作为数据卷加载至容器中，由 Docker 容器执行具体的分析工作，分析任务完成后将分析结果存储之 Redis 并通过 RabbitMQ 发送任务完成消息，系统服务端在此期间一直监听 RabbitMQ 的结果队列，一旦监听到任务完成的消息立即查询分析结果，判断任务是否成功，若成功则将分析结果解析并保存至 MongoDB，否则将其记录至 Log 日志。

3.5　评估任务控制模块设计

3.5.1　架构设计

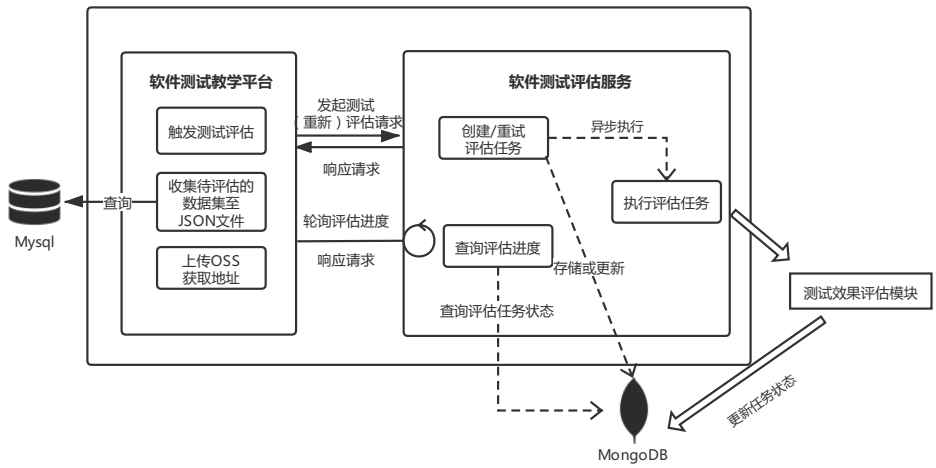

图 3.10: 评估任务控制模块架构图

评估任务控制模块架构图如图 3.10所示，本模块主要负责对评估任务的管理，分别为评估任务的创建、重试以及评估任务状态的查询，以 RestFul 接口的形式供教学平台调用。教师在教学平台的考试结果页面点击测试评估或者重新评估的按钮，就会发起创建或者重试评估任务的请求，同时查询收集待评估的数据集，包括测试分析结果以及其他考试原始相关数据等，虑到数据集的大小以及传输方式，将其以文件的方式写入 JSON 文件上传至 OSS 服务器，以上传的 url 地址作为调用创建或重试评估任务请求的参数，方便测试效果评估服务获取。创建或重试之后评估任务通过配置异步线程池异步执行，任务执行过程中涉及到任务状态的变更，包括开始执行、下载待评估的数据集、解析数据集、测试评估、保存数据库、评估成功、评估失败等状态，一旦变更就会更新数据库中异步任务的状态信息。在此期间，通过轮询查看评估任务进度的接口实时反馈

评估进度，任务到达最终状态也就是成功或失败时，结束轮询。

3.5.2　流程设计

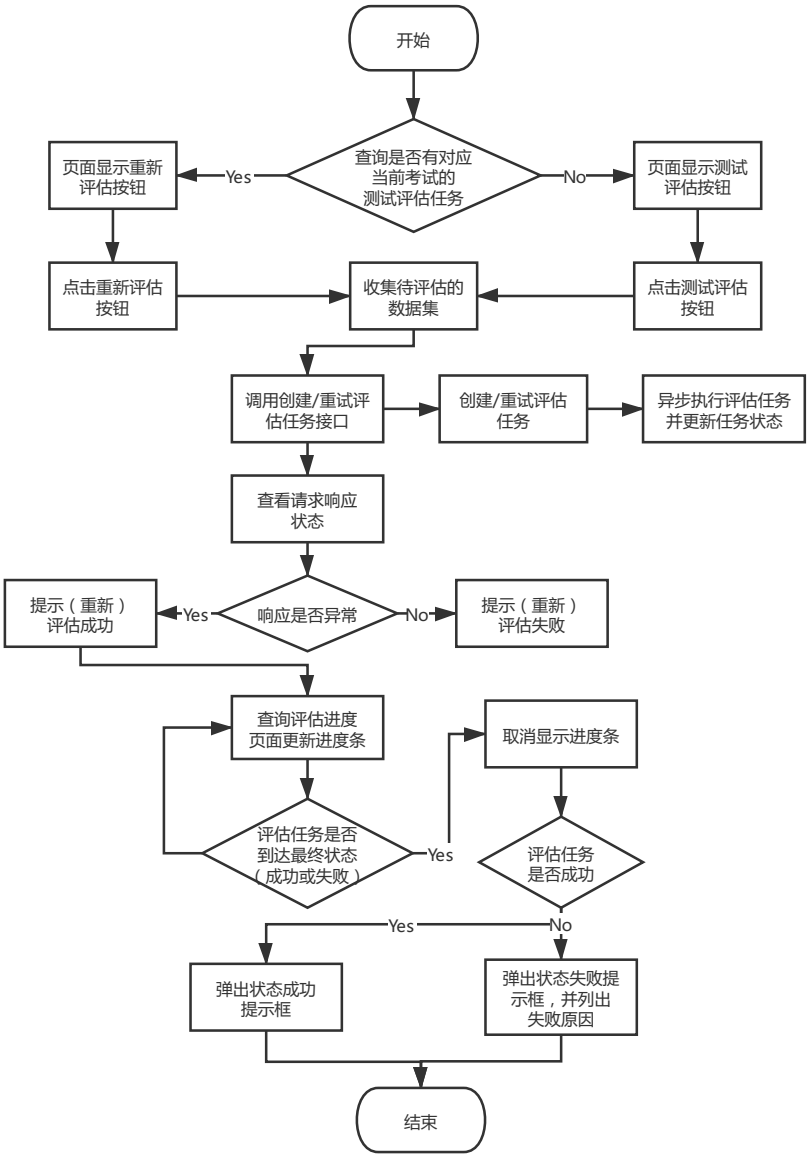

图 3.11: 评估任务控制模块流程图

评估任务控制模块流程图如图 3.11所示，主要涉及到组织考试的教师在结果页面进行测试评估或者重新评估的操作。首先查询是否有对应当前考试的测试评估任务，如有页面则显示重新估计按钮，否则显示测试评估按钮，在教师点击按钮后，对待评估的数据集进行收集，如前面架构设计所说将数据集以文件

的方式写入 JSON 文件上传至 OSS 服务器，以上传的 url 地址作为调用创建或重试评估任务请求的参数，评估任务被创建或重试后，主要执行逻辑采用异步的方式进行，直接返回请求响应。教学平台服务端查看请求响应的状态，若无异常则提示评估成功，否则提示评估失败，之后评估任务的进度通过轮询的方式查看并在页面上以进度条的方式展现，每次比较评估任务状态是否到达最终状态，即任务成功或失败，若到达最终状态则结束轮询，页面上取消显示进度条，弹出提示框显示任务状态，失败的话另外列出失败的原因。

3.6　测试效果评估模块设计

3.6.1　架构设计

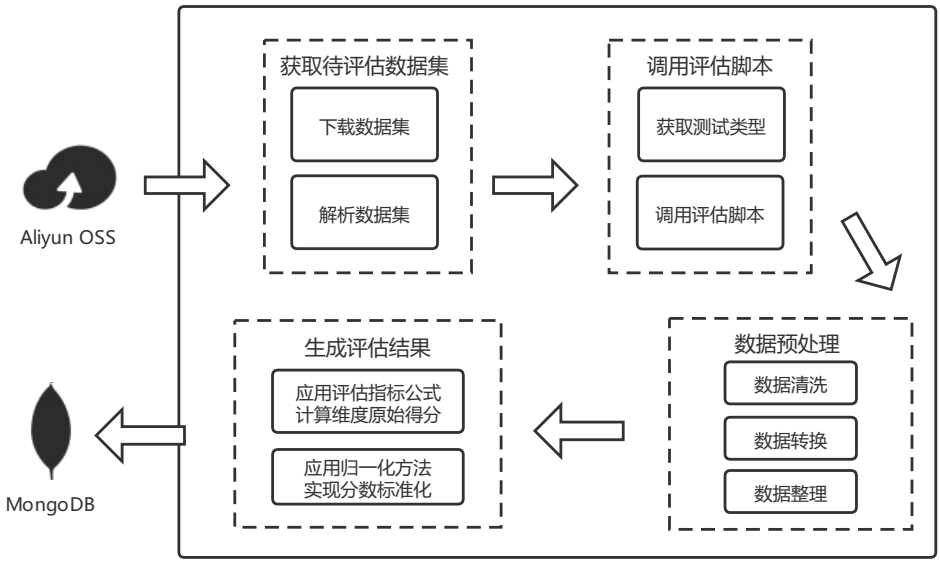

图 3.12: 测试效果评估模块架构图

测试效果评估模块架构图如图 3.12所示，本模块以待评估的数据集为数据输入，下载并解析数据集后，根据考试中每个试题的 targetType 和 caseType 字段获取其测试类型，再根据测试类型调用相应的评估脚本，首先进行数据的预处理，包括数据清洗、数据变换、数据整理，主要对空缺值的处理以及时间单位的统一操作等，将其整理为某个学生 Id 与对应的测试代码多个类型分析结果的形式，其后根据软件测试评估指标对整理好的数据项应用计算公式批量计算出指标数据，生成测试效果原始分，由于不同指标分数在初步计算后值域不同，且原始分相对难以看出个人定位以及进步程度，因此最后应用数学归一化方法将原

始分数标准化，为方便比较学生之间的测试效果以及在不同考试和不同试题中分数的分布变化，最后将标准分保存至数据库中供评估结果可视化模块渲染。

3.6.2　评估指标设计

计算指标分数基于软件测试多维评估体系之上，主要从覆盖召回率、覆盖准确率、缺陷检测率、测试兼容性、测试运行效率、测试可维护性、测试编码效率七个维度对测试效果进行评估。

覆盖召回率和覆盖准确率是将机器学习中的混淆矩阵概念与传统测试覆盖率相结合，主要是为了引入覆盖准确率来考察学生测试代码的冗余问题，参考了 Carver 等人 [29] 在评估学生测试能力时将测试套件的冗余程度作为评估项，假设一个学生获得 100% 的代码覆盖率，但实际 50% 的测试用例是多余的，这可以表明学生对测试有效性的掌握程度不高。混淆矩阵用于统计分类问题中对分类算法性能评估 [30]，通过混淆矩阵可以总结出召回率 (Recall)、准确率 (Precision)、特异性 (Specificity) 等分类指标，判断测试代码的覆盖情况可以归为分类模型中的二分类问题，对应的测试覆盖混淆矩阵如表3.13所示，覆盖召回率就等价于传统的测试覆盖率，其公式为 $Coverage_{recall} = TP/(TP + FN)$，其中 TP 为代码实际覆盖到的需求元素，$TP + FN$ 为需求覆盖元素，覆盖准确率主要考察测试代码实际覆盖元素的准确性，核实测试代码的冗余程度，其公式为 $Coverage_{precision} = TP/(TP + FP)$，其中 TP 为代码实际覆盖到的需求元素，$TP + FP$ 为测试代码覆盖元素。

表 3.13: 测试覆盖混淆矩阵

混淆矩阵	需求覆盖	非需求覆盖
实际覆盖	TP	FP
实际未覆盖	FN	TN

由于不同测试类型考察覆盖的对象不同，开发者单元测试、移动应用自动化测试和 Web 应用自动化测试对于覆盖召回率和覆盖准确率的计算方法各自有别，具体内容如下。

对于开发者单元测试而言，代码覆盖作为描述源代码被测试的程度与比例，是验证测试充分性的指标，根据代码覆盖分析工具的结果，即对应被测代码和学生测试代码的 Node 节点，覆盖召回率 ($Coverage_{recall\_develop}$) 考察学生测试代码覆盖的节点与被测代码节点重合的数量和源代码节点总数量的比值，如公式 3.1

所示。

$$Coverage_{recall\_develop} = \frac{Node_{covered} \cap Node_{meta}}{Node_{meta}} \times 100\% \tag{3.1}$$

其中 $Node_{covered}$ 为测试代码覆盖的节点个数，$Node_{meta}$ 为源代码节点个数。

覆盖准确率 ($Coverage_{precision\_develop}$) 考察学生测试代码覆盖的节点与源代码节点重合的数量和测试代码覆盖节点之间的比值，如公式 3.2 所示。

$$Coverage_{precision\_develop} = \frac{Node_{covered} \cap Node_{meta}}{Node_{covered}} \times 100\% \tag{3.2}$$

其中 $Node_{covered}$ 为测试代码覆盖的节点个数，$Node_{meta}$ 为源代码节点个数。

对于移动应用自动化测试而言，覆盖召回率 ($Coverage_{recall\_app}$) 是通过预设需求页面与控件，考察学生编写的测试脚本是否能够覆盖到需求页面以及控件，将脚本覆盖的页面及控件与需求页面及控件元素比对，计算重合数量与需求总数的比值，如公式 3.3 所示。

$$\begin{aligned} Coverage_{recall\_app} = & \frac{Widget_{covered} \cap Widget_{required}}{Widget_{required}} \times \alpha \\ & + \frac{Activity_{covered} \cap Activity_{required}}{Activity_{required}} \times (1-\alpha), \alpha \in (0,1) \end{aligned} \tag{3.3}$$

其中 $Widget_{covered}$ 为脚本覆盖的控件的数量，$Widget_{required}$ 为需求控件的数量，$Activity_{covered}$ 为脚本覆盖的页面的数量，$Activity_{required}$ 为需求页面的数量。

覆盖准确率 ($Coverage_{precision\_app}$) 是比对脚本覆盖的页面及控件与需求页面及控件元素，计算其重合的数量与脚本覆盖的页面及控件总数的比值，如公式 3.4 所示。

$$\begin{aligned} Coverage_{precision\_app} = & \frac{Widget_{covered} \cap Widget_{required}}{Widget_{covered}} \times \alpha \\ & + \frac{Activity_{covered} \cap Activity_{required}}{Activity_{covered}} \times (1-\alpha), \alpha \in (0,1) \end{aligned} \tag{3.4}$$

其中 $Widget_{covered}$ 为脚本覆盖的控件的数量，$Widget_{required}$ 为需求控件的数量，$Activity_{covered}$ 为脚本覆盖的页面的数量，$Activity_{required}$ 为需求页面的数量。

对于 Web 应用自动化测试而言，通过预设需求功能点，考察学生编写的测试脚本对于功能点的覆盖情况，覆盖召回率 ($Coverage_{recall\_web}$) 比对脚本覆盖的功能点数与需求功能点，计算其重合的数量与需求功能点总数的比值，如公式 3.5 所示。

$$Coverage_{recall\_web} = \frac{Function_{covered} \cap Function_{required}}{Function_{required}} \times 100\% \tag{3.5}$$

其中 $Function_{covered}$ 为脚本覆盖的功能点数，$Function_{required}$ 为需求功能点总数。

覆盖准确率 ($Coverage_{precision\_web}$) 通过运行学生的脚本比对脚本覆盖的功能点数与需求功能点，计算其重合的数量与脚本覆盖功能点总数的比值，如公式 3.6 所示。

$$Coverage_{precision\_web} = \frac{Function_{covered} \cap Function_{required}}{Function_{covered}} \times 100\% \tag{3.6}$$

其中 $Function_{covered}$ 为脚本覆盖的功能点数，$Function_{required}$ 为需求功能点总数。

缺陷检测率通过预设缺陷考察脚本检测缺陷的能力，对于开发者测试通过变异分析预设变异体达到效果，变异分析是一种对测试数据集的有效性、充分性进行评估的技术，通过在源代码中执行变异算子生成变异体，可以通过变异体杀死的情况评价每个测试用例的缺陷检测能力。缺陷检测率通过变异体杀死的百分比来衡量，如公式 3.7 所示。

$$BugDetectionRate = \frac{Mutant_{killed}}{Mutant_{total}} \times 100\% \tag{3.7}$$

其中 $Mutant_{killed}$ 为杀死的变异体个数，$Mutant_{total}$ 为被测代码中引入的变异体总数，分别对应测试代码分析结果中的 CaughtDTO 和 NodeDTO 的个数，根据此公式计算出的分值作为开发者单元测试的缺陷检测率指标得分。

测试兼容性考察测试代码在不同硬件、软件环境下运行成功的情况，通过计算在不同环境下运行成功次数与总次数的比值评估其兼容性，如公式 3.8 所示。

$$Compatibility = \frac{Run_{successed}}{Run_{total}} \times 100\% \tag{3.8}$$

其中 $Run_{successed}$ 和 Run_{total} 对于移动应用自动化测试是测试脚本在不同机型的设备中运行成功的次数以及总次数，对于 Web 应用自动化测试是不同浏览器环境下运行成功的次数以及总次数。开发者单元测试的兼容性问题具体体现在 JDK 版本不同易引起的差异，由于教学平台目前指定了 JDK 的版本，规避了兼容性问题，因此针对开发者单元测试此项不进行统计。

测试运行效率考察测试代码的运行时间，达到同样的测试效果即达到相同的覆盖召回率，测试代码运行的时间不同，因此评估运行效率时采用覆盖召回率的得分与运行时间的比值，如公式 3.9 所示。

$$Efficiency_{running} = \frac{Coverage_{recall}}{Time_{running}} \times 100\% \tag{3.9}$$

其中 $Coverage_{recall}$ 为覆盖召回率得分，$Time_{running}$ 为测试脚本运行时间，脚本运行时间对应着运行效率分析工具的结果。

测试编码效率考察学生编写测试代码的时间，达到同样的测试效果即达到相同的覆盖召回率，由于学生测试框架及语法掌握的熟练程度不同等原因，其编码时长不同，因此增加对编码效率的考察项。在保证覆盖召回率相同高的前提下，测试代码编写的时间越短，编码效率越高。通过计算覆盖召回率与编码时间的比值来充分地评估编码效率，如公式 3.10 所示。

$$Efficiency_{coding} = \frac{Coverage_{recall}}{Time_{coding}} \times 100\% \tag{3.10}$$

其中 $Coverage_{recall}$ 为代码覆盖率，$Time_{coding}$ 为编码时长。计算对应覆盖率最高得分的提交记录的提交时间与考试开始时间的差值即可得到测试编写时间，再将覆盖召回率得分对其求比值的分数作为编码效率得分。

测试可维护性主要对编码风格进行考察，培养良好的编码风格，可以提高代码的可读性、可理解性、可维护性。对于生产代码如此，测试代码亦是如此，甚至测试代码更需要注重良好的编程风格。如果测试不能保持整洁，那么脏测试就相当于没测试，也就失去了保证生产代码可扩展的一切要素 [31]。因此这里利用编码风格分析的结果作为可维护性维度评分项。以 Checkstyle 分析得到的代码违反规则的详情作为评分依据，根据 Google 的编码规范进行检测。Google 的编码规范主要分为 6 大类，分别为源代码文件基础、源代码文件结构、格式化、命名、编程实践、Javadoc 注解，每一类包括多个检查项。检测结果为违反的检查项名称及对应次数，因此考察可维护性评分时将检查项进行归类，对各类别进行计数，与所有学生违反该类次数的最大值进行比较，如公式 3.11 所示。

$$Maintainability = \frac{1}{n}\sum_{i=1}^{n}(1 - \frac{Violation_i}{\max(Violation_i)}) \times 100\% \tag{3.11}$$

其中 $Violation_i$ 表示违反第 i 类代码规范的次数，$\max(Violation_i)$ 表示考试同试题中所有学生测试代码违反第 i 类代码规范次数的最大值。

3.7　评估结果可视化模块设计

3.7.1　页面设计

评估结果可视化模块主要展示页面设计如图 3.13所示，主要分为四大部分，测试评估数据总览、历史评估记录、测试评估结果以及测试评估指标体系。测试评估数据总览主要对应某位学生结合各个测试类型的评估数据的总结展示，包括整合各个测试类型的测试评估总体图、测试评估类型分布图。测试效果历史评估记录包括各测试类型的历史评估记录以及对应的测试效果变化趋势图，以达到多方面地对评估总体结果进行展现，充分给予用户反馈的目的。测试评估

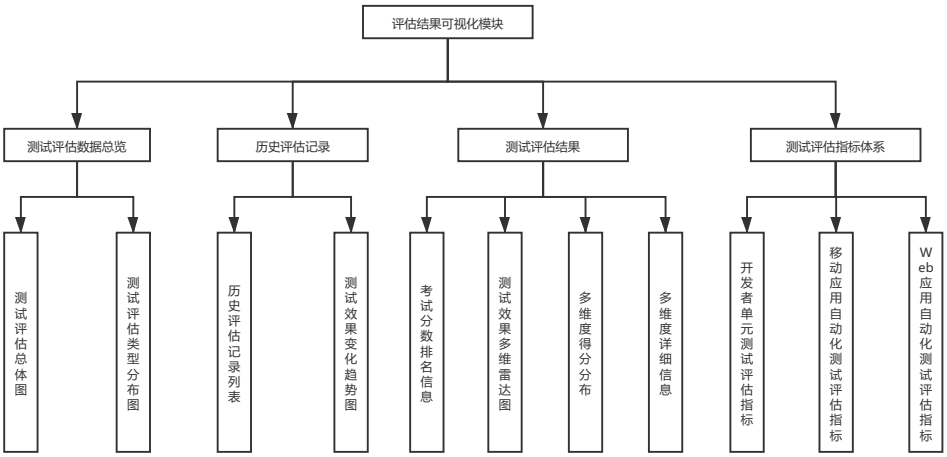

图 3.13: 评估结果可视化模块页面设计功能图

结果负责对应某场考试某位学生的测试评估结果显示，包括个人信息、考试分数与排名信息、测试效果多维雷达图、多维度分数分布以及各维度对应的详细信息等。测试评估指标体系负责展现各测试类型的测试评估指标介绍详情。

每一个页面对应一个页面组件，Vue 中页面组件对应一个.vue 文件，包括模块（Template）、JS 脚本 (Script) 和样式 (Style)，通过定义路由将组件映射到路由上，Vue Router 便可以利用动态路由匹配来匹配加载对应的组件，前后端交互采用基于 Promise 的 Axios 库来发送请求，数据以 JSON 的形式传输，前端接收到请求响应后将数据进一步处理赋值给页面中的属性完成数据的加载，框架捕获到数据的更改通过响应式原理渲染页面，涉及到多个组件中状态的管理采用 Vuex 统一实现管理。

3.7.2　实体设计

评估结果可视化模块实体关系图如图 3.14所示，前后端交互的时候定义页面显示的实体结构，包括学生基本信息、考试基本信息、试题基本信息、评估结果、评估维度等实体。这里定义的实体对应了前端页面显示的内容，页面主要涉及对评估结果的显示，评估结果与考试一一对应，一种考试包括多个试题，因此评估结果与试题之间是一对多的关系，针对不同类型的测试对应了多维度的评估，因此试题与评估维度是一对多的关系，能力维度表现的主体是学生，在一次评估结果中，一个学生包含了多个评估维度，其中每个维度包含 id、名称、数值以及超过数量，超过数量是指学生当前维度得分在本场考试中超过的同学数量。

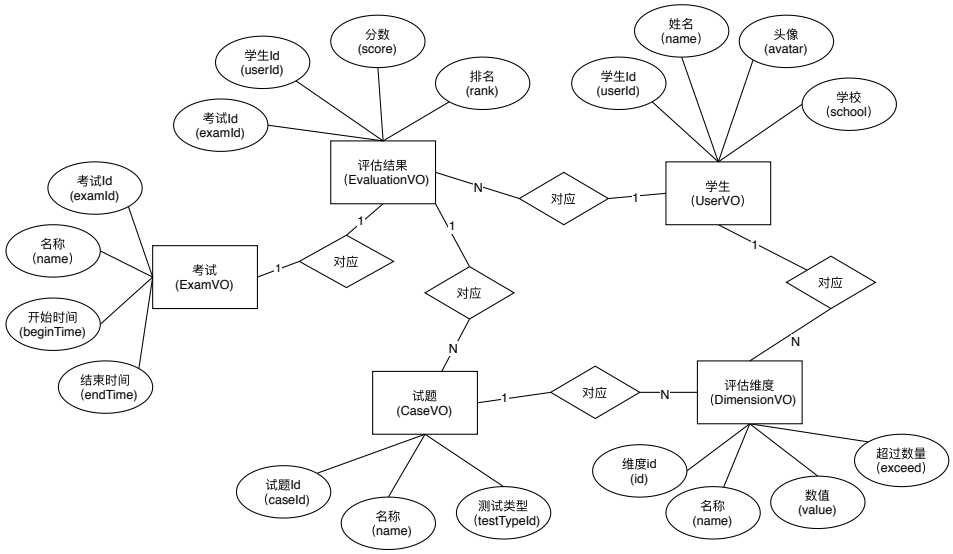

图 3.14: 评估结果可视化模块实体关系图

3.8　本章小结

针对学生在软件测试教学平台上提交测试代码的实际场景，本章首先对软件测试多维评估系统的需求进行分析，主要包括功能性需求、非功能性需求、系统用例设计，其次分析系统的总体设计，包括技术架构、4+1 视图以及功能模块的划分，主要分为测试代码分析模块、评估任务控制模块、测试效果评估模块以及评估结果可视化模块。最后分别详细阐述对各个模块的初步设计。

第四章　软件测试多维评估系统的详细设计与实现

本章基于前文系统的需求分析以及架构总体设计，对划分的功能模块包括测试代码分析模块、评估任务控制模块、测试效果评估模块以及评估结果可视化模块在初步设计的基础上进行详细设计与实现。

4.1　测试代码分析模块设计与实现

测试代码分析模块主要负责根据测试类型创建并执行对应的测试代码分析任务，分析任务执行的实质就是下载学生提交的测试代码，调用相应的分析工具提取可以描述测试代码特性的分析结果，将分析结果进行存储供后续测试代码评估模块使用，作为评估的数据支撑。此外，测试代码分析模块还提供用户交互的功能，支持管理员在学生提交测试代码后对当前考试触发测试代码分析的操作，涵盖对测试代码分析任务的创建执行、停止以及重试功能。下面分别从测试代码分析任务管理以及测试代码分析特性提取两大块进行详细描述。

4.1.1　测试代码分析任务管理

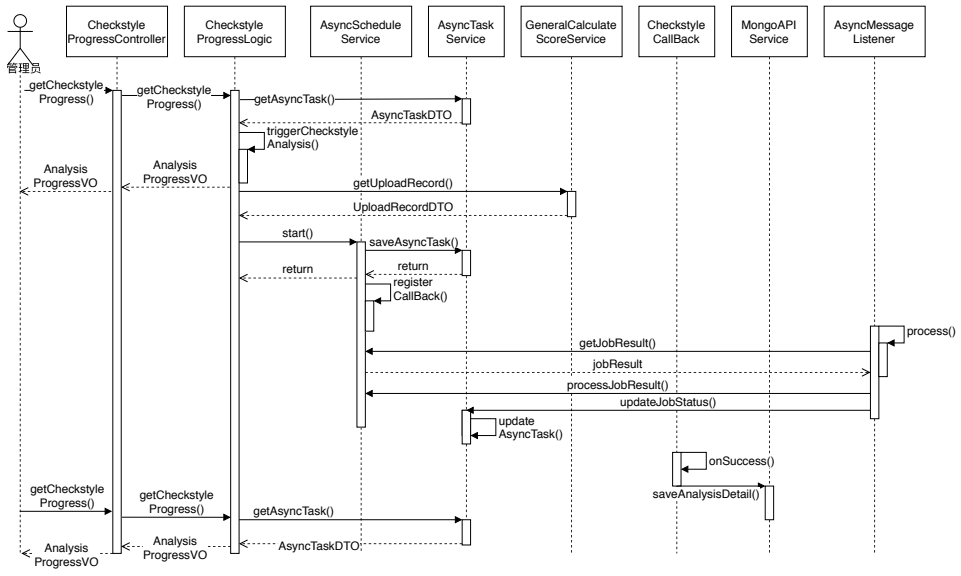

图 4.1: 测试代码分析任务管理时序图

如图 4.1所示是测试代码分析任务管理的创建分析任务部分。管理员通过前端页面执行测试代码分析任务，前端框架调用分析工具对应的 ProgressController

中的 getProgress 接口，不同的工具对应不同的 Controller，比如这里以编码风格分析工具为例是调用的 CheckstyleProgressController 中的 getCheckstyleProgress 接口，主要逻辑是一致的。getCheckstyleProgress 接口主要调用了 CheckstyleProgressLogic 中的 getCheckstyleProgress 方法，方法首先会通过 AsyncTaskService 的 getAsyncTask 方法根据考试 Id 和试题 Id 来数据库中是否有对应分析任务，若无则通过调用 CheckstyleProgressLogic 中的 triggerCheckstyleAnalysis 方法来新建并执行分析任务。

triggerCheckstyleAnalysis 方法逻辑主要包括调用 GeneralCalculateScoreService 类的 getLatestUploadRecord 方法来获取学生最新提交记录的测试代码以及相关信息，将其作为分析任务的参数，参数包括 params 和 context, 都是以 List<Map<String,String» 的方式封装。其后调用 AsyncScheduleService 中的 start 方法开始执行分析任务，将 params 和 context 参数进行整理，每对应一组 params 和 context，就在异步任务 AsyncTaskDTO 中新建一个 AsyncJobDTO 对象，每个 AsyncJobDTO 对象对应参与当前考试当前试题的每个学生的分析工作。

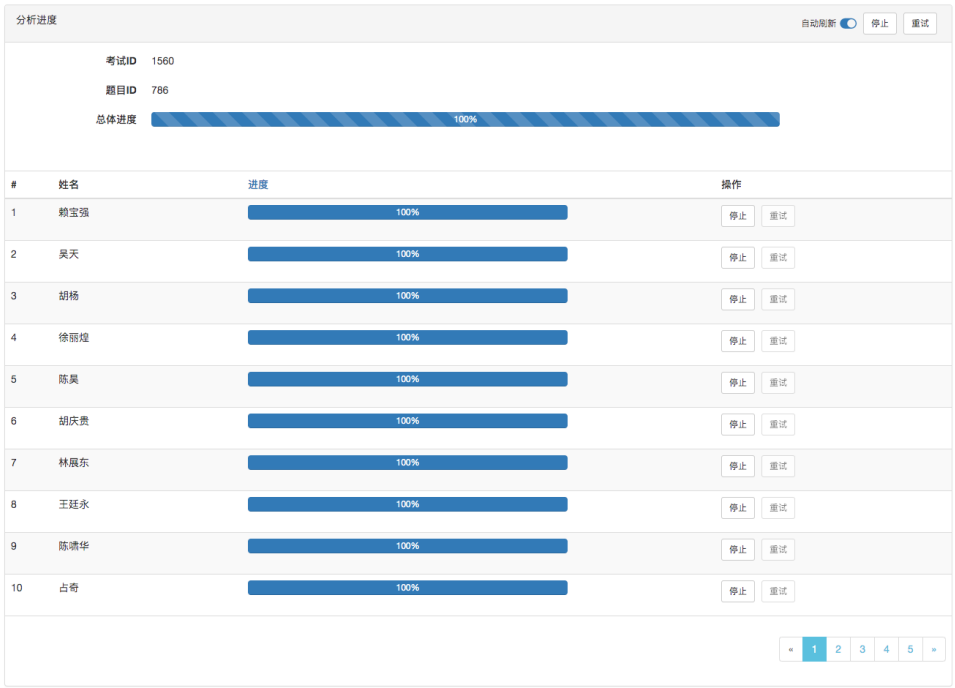

图 4.2: 测试代码分析任务管理界面示意图

创建完分析任务后调用 Celery 服务器的 RestFul 接口调用分析工具分析测试代码，并且注册异步任务回调方法，在 Celery 执行完成后会通过 RabbitMQ 发出任务完成的消息，AsyncMessageListener 监听器负责监听 RabbitMQ 的任务结

果队列，一旦收到任务完成的消息，就执行 process 方法，主要调用 AsyncScheduleService 类的 getJobResult 方法获取分析结果，再通过 processJobResult 方法触发回调方法，通过 MongoAPIService 的 saveAnalysisDetail 将结果存储到数据库，作为后续工作以及教学平台其他服务的数据支撑，在此期间支持轮询对应的 getProgress 方法对分析任务进度的实时查看，测试代码分析任务管理界面示意图如图 4.2所示。

表 4.1: 测试代码分析任务管理主要类

分类	主要包含的类
监听器类	AsyncMessageListener
控制器层	MutationProgressController、CheckstyleProgressController、RunTimeProgressController、AppAnalysisProgressController、WebAnalysisProgressController
逻辑层	MutationProgressLogic、CheckstyleProgressLogic、RunTimeProgressLogic、AppAnalysisProgressLogic、WebAnalysisProgressLogic
服务层	AsyncTaskService、AsyncScheduleService、GeneralCalculateScoreService、MongoAPIService
数据访问层	UploadRecordDao、GradeGeneralDao
数据类	AsyncTask、AsyncJob、AnalysisDetail、AnalysisProgressVO、AnalysisJobVO

如表 4.1所示，给出了测试代码分析任务管理中主要涉及到的类的信息，主要分为控制器层、逻辑层、服务层、数据访问层以及数据类，控制器层负责接收并处理请求，处理的过程通过调用逻辑层来实现，逻辑层又可以通过多个基础的服务层类完成对应的业务逻辑，数据访问层负责实体数据库的增删改查操作。主要包括 MutationProgressController 等以 ProgressController 为结尾的类负责对外提供接口，和对应实现业务功能的 Logic 类，Logic 类主要调用了负责保存、更新、查询以及重试分析任务的 AsyncTaskService 类，负责控制分析任务具体执行过程如开始、暂停、重试的 AsyncScheduleService 类，以及获取结果以及负责查询学生提交记录相关信息的 GeneralCalculateScoreService 类，MongoAPIService 类负责在回调逻辑中将分析结果存储至 MongoDB 中，此外还包括数据库访问层类 UploadRecordDao、GradeGeneralDao 以及描述异步任务的实体类。

AsyncTask 数据类的详细描述如表 4.2所示，表示对应一场考试中一个试题对应一种分析类型的测试代码分析任务，可以拆分为多个面向学生粒度的分析工作 AsyncJob，其详细描述如表 4.3所示，由于一场考试涉及到多个学生，分析任务工作量较大，将其拆分为一个个的分析工作，缩小任务调度的单位，也能更

表 4.2: AsyncTask 类详情列表

字段	含义	类型	备注
id	任务编号	ObjectId	主键，不为空
sessionId	调度编号	String	任务经 Celery 调度生成的任务 ID
queueName	队列名称	String	任务发送的消息队列名称
toolName	工具名称	String	任务对应的工具名称
type	工具类型	Short	任务对应的工具类型
jobs	分析工作	List<AsyncJob>	是分析任务的拆解，对应到每个学生测试代码的分析工作

表 4.3: AsyncJob 类详情列表

字段	含义	类型	备注
jobId	工作编号	String	主键，不为空
status	状态	int	工作执行的状态，包括进行中、成功和失败
params	执行参数	Map<String,String>	执行工作需要的参数，比如测试代码包 OSS 地址等
context	回调参数	Map<String,String>	执行工作完成后传给回调逻辑的参数，比如学生提交记录的 Id 号，在回调逻辑中需要根据提交记录 Id 来保存分析结果信息

加准确地对分析过程进行管控。AnalysisDetail 数据类的详细描述如表 4.4所示，表示学生在一场考试一个试题对应一种分析类型的分析结果详情，作为后续测试效果评估以及教学平台其他服务的数据支撑。

表 4.4: AnalysisDetail 类详情列表

字段	含义	类型	备注
id	详情编号	ObjectId	主键，不为空
examId	考试编号	long	分析结果对应的考试编号
caseId	试题编号	long	分析结果对应的试题编号
userId	用户编号	long	分析结果对应的用户编号
metricId	指标编号	int	分析结果对应的指标编号
content	分析结果详情	Object	分析结果的详情信息，如变异分析中的变异体的杀死、存活以及对应代码等详情信息

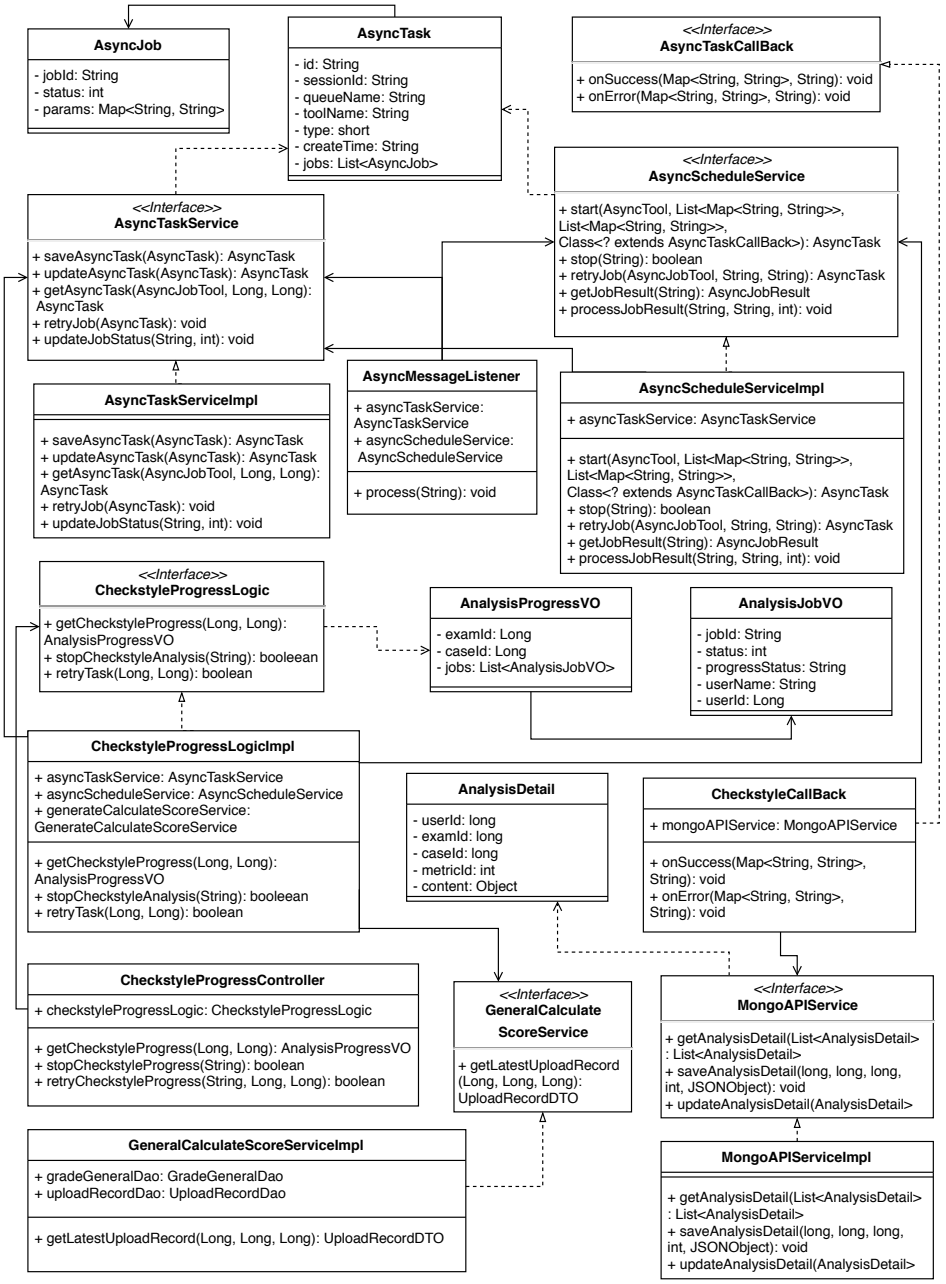

图 4.3: 测试代码分析模块核心类图

测试代码分析模块核心类图如图 4.3所示，AsyncTask 类作为异步任务的实体类，一个异步分析任务就是指对当前考试中一个试题执行代码分析的过程。由于一场考试涉及到大量学生，每个学生对应当前试题的测试代码分析作为异步分析任务中的一个 job，即 AsyncJob。AsyncTaskService 接口定义了异步任务的保存、更新、查询以及重试方法，由 AsyncTaskServiceImpl 类来实现对应的方法。AsyncScheduleService 接口对应的是异步任务具体执行过程的方法，包括开

始、暂停、重试以及获取结果等，同样地，通过 AsyncScheduleServiceImpl 类实现对应的方法。

```
@RabbitListener(queues = "#{rabbitMQConfiguration.JOB_RESULT_QUEUE}")
public void process(@Payload String result) {
    // ...处理接收的消息
    JSONObject jResult = JSONObject.fromObject(sResult.toString());
    String jobId = jResult.getString("jobId");
    int returnCode = jResult.getInt("returnCode");
    int jobStatus = returnCode==ReturnCodeConstants.SUCCESS?
                    AsyncJobStatus.SUCCESS:AsyncJobStatus.FAIL;
    try{
        // 调用 getJobResult 方法获取分析结果
        String resultDetail = asyncScheduleService.getJobResult(jobId).getResult();
        // 调用 processJobResult 方法将分析结果传入回调逻辑
        asyncScheduleService.processJobResult(jobId, resultDetail, returnCode);
        log.info("receiver:"+ sResult);
    } catch (Exception e) {
        // 异常处理，将错误收集至日志中
        jobStatus = AsyncJobStatus.FAIL;
        e.printStackTrace();
        log.error("receive job result error "+ e.getMessage());
    } finally {
        // 更新分析任务状态
        asyncTaskService.updateJobStatus(jobId, jobStatus);
    }
}
```

图 4.4: AsyncMessageListener 监听器关键代码

Celery 通过集成 Flask 作为客户端实例提供开始、暂停、重试的接口，AsyncScheduleServiceImpl 类主要是对 Celery 对应功能的接口发出请求，Celery 收到执行分析任务的请求后，将 AsyncTask 中的 AsyncJob 集合分发到 Celery Worker 执行。AsyncTaskService 和 AsyncScheduleService 类作为异步任务的通用逻辑，对应具体分析工具的逻辑在于配置 AsyncJobTool 类型的参数，如 AsyncScheduleService 中的 start 方法，需要提供 AsyncJobTool 参数，每个分析工具都会在 AsyncJobTool 类中注册。核心类图中以编码风格分析工具为例，CheckstyleProgressLogic 接口定义了具体编码分析任务的开始、停止与重试方法，CheckstyleProgressLogicLogicImpl 实现类中通过调用 AsyncTaskService 和 AsyncScheduleService 的方法实现具体的功能，并由 CheckstyleProgressController 控制器调用，对应页面交互中的测试代码分析管理功能，AnalysisProgressVO 和 AnalysisJobVO 为页面显示对象对应了测试代码分析进展页面的总进度和对应具体学生测试代码分析 Job 的进度。其他每对应一个分析工具就有对应的 ProgressLogic 以及

ProgressController 类。

```
private void saveAnalysisDetail(String result, long userId, long examId, long caseId, long
uploadId) {
    JSONObject jsonObject = JSONObject.parseObject(result);
    log.info("result jsonObject:" + jsonObject);
    for (String key : jsonObject.keySet()) {
        JSONObject detail = jsonObject.getJSONObject(key);
        switch (key) {
            case "recall":
                JSONObject reason = detail.getJSONObject("reason");
                // 将分析结果保存至 mongoDB
                processDetailDataToAnalysisDetail(reason, userId, examId, caseId,
EvaluationMetricConstant.COV_RECALL);
                break;
            case "precision":
                reason = detail.getJSONObject("reason");
                processDetailDataToAnalysisDetail(reason, userId, examId, caseId,
EvaluationMetricConstant.COV_PRECISION);
                break;
            // 分维度保存，其他维度的省略
            ......
            default:
                break;
        }
    }
}
```

图 4.5: 保存分析结果关键代码

AsyncMessageListener 监听器负责监听 RabbitMQ 任务结果队列，在接收到任务完成的消息后负责调用 AsyncScheduleService 对应的方法获取分析结果并触发回调将分析结果保存至数据库中，通过调用 AsyncTaskService 更新任务的状态，下面分别是异步任务完成结果监听和保存分析结果的部分关键代码，分别如图 4.4和图 4.5所示。

4.1.2　测试代码分析特性提取

测试代码分析特性提取对应的是封装的测试代码分析工具内部的逻辑，按照测试类型可以将工具分为开发者单元测试代码分析工具、移动应用自动化测试代码分析工具以及 Web 应用自动化测试代码分析工具。其中，编码风格分析是三者俱有的，主要区分点在于三种测试代码包的项目结构不同，在触发分析任务时根据测试类型判断对应的待测路径，将其以参数的形式传入配置编码风格工具检测的路径。

开发者单元测试代码分析工具包括覆盖分析、变异分析、运行效率分析、编码风格分析等，以变异分析的过程举例，如图 4.6所示，Entrance 作为工具的入口类，MutationWorker 类主要负责实现具体的开始、停止、返回结果的功能。测试代码包由 Celery Worker 下载并挂载到工具 Docker 容器的指定的路径下，由 extractZipFile 方法对测试代码包解压，通过 runPitest 方法来调用 PITest 的 jar 包对测试代码进行分析，分析结果通过 parseResult 方法解析为 MutationResult 类的实例。TransferUtil 类负责将分析结果转化为对应的 NodeDTO 和 CaughtDTO，分别对应了试题执行变异算子生成变异体的 node 节点与学生杀死的变异体的 node 节点，将结果以 JSON 的形式输出，更新分析状态为完成。

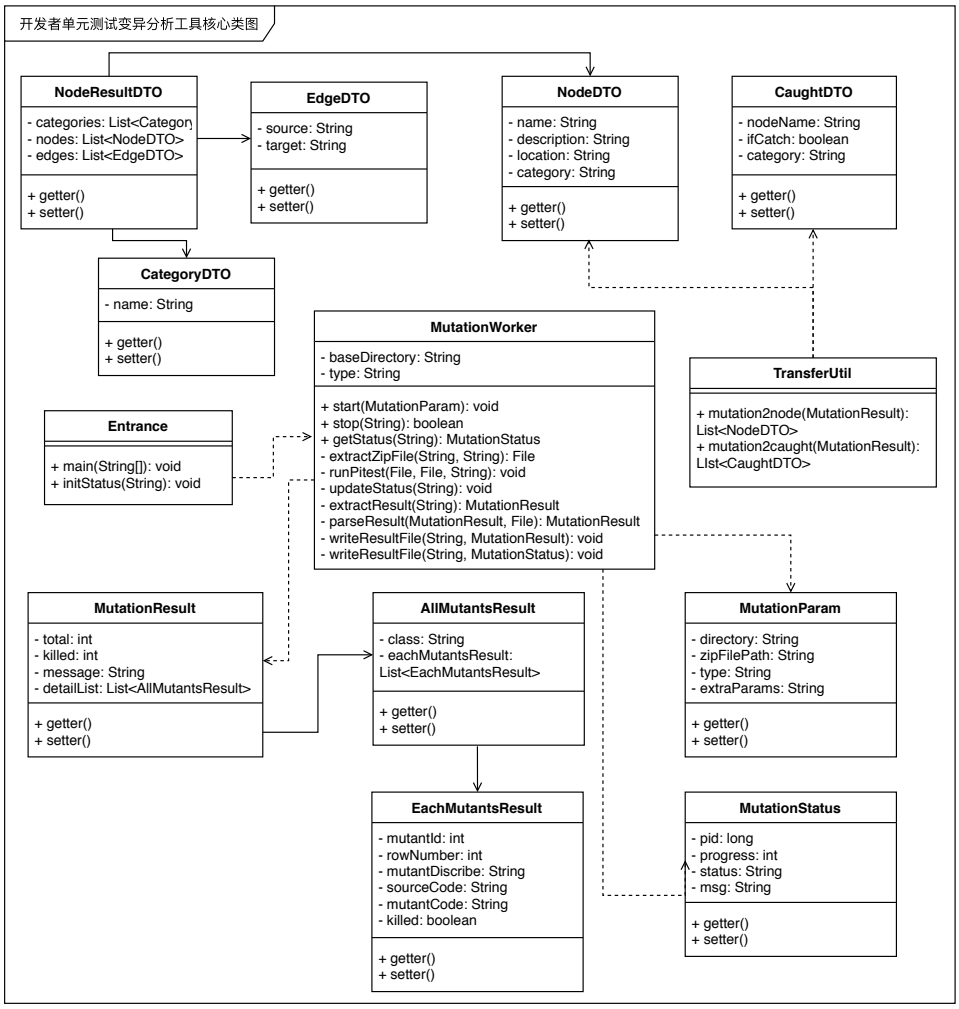

图 4.6: 开发者单元测试变异分析工具核心类图

移动应用自动化测试代码分析工具包括覆盖分析、兼容性分析、运行效率分析、编码风格分析等。学生提交的移动应用测试代码包中包含了考试过程中

执行脚本的过程数据，涉及到脚本覆盖页面和控件元素情况、脚本运行总次数、成功次数以及脚本运行时间等。在考试过程中，学生使用教学平台提供的包含插件的 Eclipse 来编写并执行自动化测试脚本。前序工作通过对 SeleniumJavaClient 和 AppiumJavaClient 中的 click 方法以及 getElementById、findElementByXPath 等一系列获取元素的方法进行改写并重新编译，在脚本执行相关方法时改写逻辑负责记录脚本覆盖到的页面以及控件元素，以写文件的方式输出。同时插件负责记录脚本运行总次数、成功次数以及运行时间等数据，也将其以写文件的方式输出。借由此，分析工具主要流程在于对测试代码包进行解压，读取过程数据文件，将数据以 JSON 的形式输出，更新分析状态为完成。

```
public class Example {
    public static void test(WebDriver driver) {
        try {
            // 学生在此编写脚本代码
        } catch (Exception e) {
            // 异常处理
        }
    }

    public static void main(String[] args) {
        // 主方法负责新建 ChromeDriver 调用执行 test 方法
        WebDriver driver = new ChromeDriver();
        try {
            test(driver);
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            driver.quit();
        }
    }
}
```

图 4.7: Web 应用自动化测试代码结构

Web 应用自动化测试代码分析工具包括覆盖分析、兼容性分析、运行效率分析、编码风格分析等，以兼容性分析为例进行具体介绍，常见的 Web 兼容性测试主要包括操作系统、浏览器、分辨率和网速等方面的兼容性测试，这里主要对浏览器兼容性进行测试，浏览器的核心在于其内核，用来解释网页语法并将其渲染至网页显示，不同的浏览器内核对网页的语法解释不同，主要内核包括 Trident、Gecko、Webkit 以及 Blink 等。针对浏览器兼容性分析，利用第二章介绍的 Selenium Grid 配置多个多种浏览器类型的负责执行脚本的 Node 实

例，将脚本分发到不同浏览器执行查看执行通过结果。Web 应用脚本测试出题预设的自动化测试代码结构如图 4.7所示，代码主要包括两个方法，主方法负责创建 ChromeDriver 调用 test 方法，test 方法的内容为学生编写具体的测试脚本代码，参数为主方法中的 chromeDriver 实例。基于此，利用 Java 反射机制从学生代码包中的 class 文件入手，通过 UrlClassLoader 的 loadClass 方法将 class 文件动态加载后调用 newInstance 方法创建当前 class 的实例，创建完成后调用 getDeclaredMethod 方法获取 test 函数，新建对应 Selenium Grid 中 Hub 测试中心点的 RemoteWebDriver 对象，配置对应浏览器类型的 DesiredCapabilities 实例，最后调用 invoke 方法将 remoteWebDriver 作为参数传入即可达到将测试脚本在指定的浏览器环境下执行的效果，具体代码如图 4.8所示。

```
private static JsonObject execute(String remoteUrl, DesiredCapabilities cap, String scriptPath){
    // 将执行 Web 测试脚本兼容性的以 JsonObject 类型输出
    JsonObject result = new JsonObject();
    try {
        // 新建 RemoteWebDriver 实例
        RemoteWebDriver driver = new RemoteWebDriver(new URL(remoteUrl), cap);
        // 通过 URLClassLoader 加载脚本代码 Example 类
        URLClassLoader loader = new URLClassLoader(
                new URL[]{new URL("file:" + scriptPath)});
        Class<?> myClass = loader.loadClass("Example");
        // 新建实例
        Object o = myClass.newInstance();
        // 获取脚本代码中 test 静态方法
        Method staticMethod = myClass.getDeclaredMethod("test", WebDriver.class);
        staticMethod.setAccessible(true);
        // 将新建的 RemoteWebDriver 实例作为参数传入，执行 test 方法
        staticMethod.invoke(o, driver);
        result.addProperty("browser", cap.getBrowserName());
        result.addProperty("flag", true);
        driver.quit();
    } ...// 异常处理，并对结果 flag 赋值为 false
    return result;
}
```

图 4.8: 利用反射及 RemoteWebDriver 执行 Web 脚本关键代码

考虑到代码可复用性，将以上逻辑构建为 Jar 包供 Web 兼容性分析工具调用，将 DesiredCapabilities 的浏览器类型及版本号、操作系统类型、Selenium Grid 中 RemoteDriverUrl 地址以及学生测试代码包路径地址以参数的形式传入，在一次分析过程中，将学生代码分别在 Selenium Grid 中配置的不同浏览器版本的 Node 节点上运行并查看脚本执行通过情况。分析工具负责在执行完成后对每一

个类型的浏览器的通过情况进行汇总整理，以 JSON 的形式输出结果。

4.2　评估任务控制模块

4.2.1　评估任务控制模块概述

评估任务控制模块主要涉及到对评估任务的创建、重试以及进度查询，由在线测试教学平台页面上进行触发，通过 RestFul 接口调用测试评估服务的方式实现具体的功能，评估任务执行过程通过配置线程池异步执行。

4.2.2　评估任务控制模块设计与实现

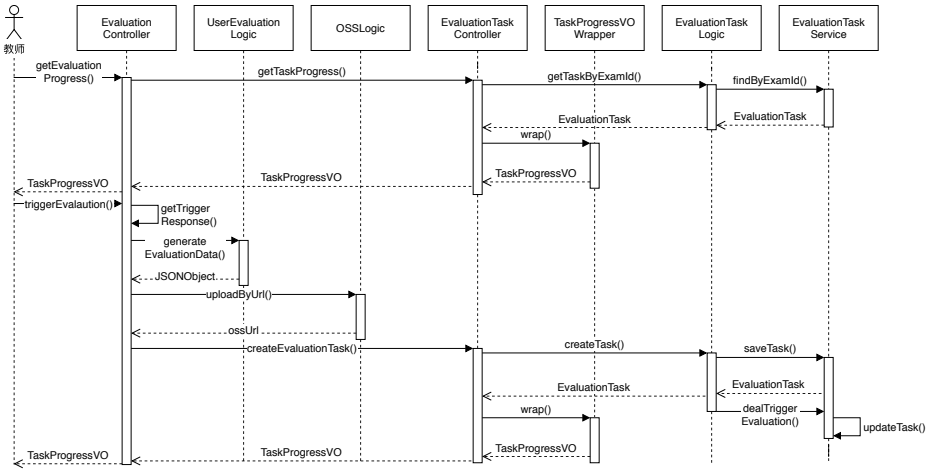

图 4.9: 评估任务控制模块时序图

如图 4.9所示是评估任务控制模块的创建评估任务及进度查询部分。在考试结果页面加载时前端调用服务端的 getEvaluationProgress 接口获取当前考试对应的评估任务的进度状态，服务端调用软件测试评估服务中 EvaluationTaskController 控制器类中的 getTaskProgress 接口。该接口以考试 Id 作为请求参数，软件测试评估服务接收到请求后，调用 EvaluationTaskLogic 中的 getTaskByExamId 方法，方法内部通过 EvaluationTaskService 的 findByExamId 方法来具体查找对应考试 Id 的评估任务。查找到任务后，将其通过 TaskProgressVOWrapper 包装类的 wrap 方法进行包装成 TaskProgressVO 的形式返回。TaskProgressVO 保留评估任务实体类的状态以外还添加了将评估任务的状态转化为百分比的字段，方便后续前端显示任务进度。前端逻辑对应 TaskProgressVO 定义状态，通过对比 status 值显示测试评估按钮还是重新评估按钮。

```
private ResponseEntity<JSONObject> getTriggerResponse(String requestUrl, Long examId) {
    String fileName = "exam_result.json";
    // 调用 generateEvaluationData 方法获取待评估的数据集
    InputStream inputStream = userEvaluationLogic.generateEvaluationData(fileName,
examId);
    String uploadUrl = "evaluation/exam_" + examId + ".json";
    // 将数据集上传至 OSS 服务器获取其 OSS url 地址
    String ossUrl = ossLogic.uploadByUrl(uploadUrl, inputStream);
    log.info("ossUrl: " + ossUrl);
    // 删除本地 json 文件
    File tmpFile = new File(fileName);
    if (tmpFile.exists()) {
        tmpFile.delete();
    }
    // 调用测试能力评估服务提供的接口，将 examId 和 ossUrl 作为请求参数传入
    RestTemplate rt = new RestTemplate();
    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);
    JSONObject json = new JSONObject();
    json.put("examId", examId);
    json.put("ossUrl", ossUrl);
    HttpEntity<JSONObject> httpEntity = new HttpEntity<>(json, headers);
    return rt.exchange(requestUrl, HttpMethod.POST, httpEntity, JSONObject.class);
}
```

图 4.10: getTriggerResponse 方法关键代码

教师点击测试评估按钮后，前端框架请求服务端的 triggerEvaluation 接口，服务端接收到请求后调用 getTriggerResponse 方法，方法代码如图 4.10所示，此方法主要调用 UserEvaluationLogic 类中的 generateEvaluationData 方法来收集待评估的数据集，将数据集以文件流的方式输入至 JSON 文件中，并通过 OSSLogic 中的 uploadByUrl 方法上传至 OSS 服务器，将上传的 OSS url 地址以及考试 Id 作为调用软件测试评估服务中 EvaluationController 类的 createEvaluationTask 接口的请求参数，接收到请求后，调用 EvaluationTaskLogic 的 createTask 方法，方法内部调用 EvaluationTaskService 的 saveTask 方法创建并保存评估任务，保存的评估任务返回后再调用 dealTriggerEvaluation 方法执行评估过程，此方法通过配置线程池异步执行，任务执行过程中涉及到任务状态的变更，包括开始执行、下载待评估的数据集、解析数据集、测试评估、保存数据库、评估成功、评估失败等状态，一旦变更就会更新数据库中异步任务的状态信息，在任务执行的过程中，前端轮询进度状态接口，将评估进度实时反馈给用户，触发测试评估界面示意图如图 4.11所示。

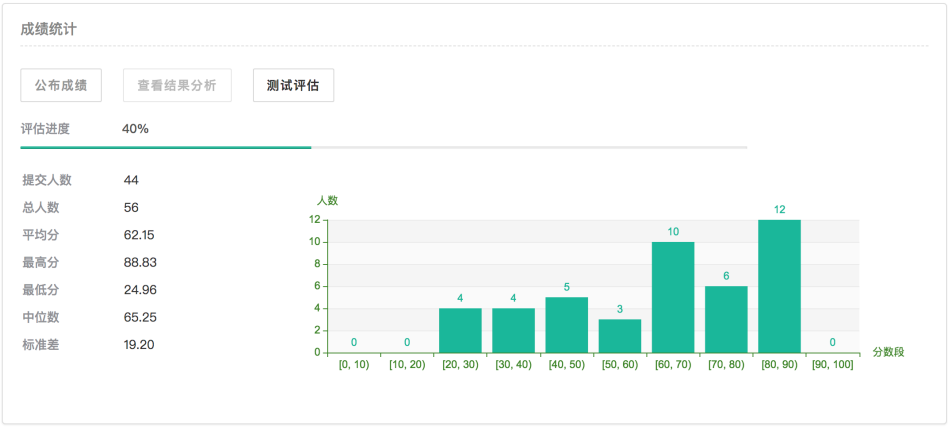

图 4.11: 触发测试评估界面示意图

评估任务控制模块中主要涉及到的类如表 4.5所示，控制器层包括 EvaluationController、EvaluationTaskController 类负责对外提供接口，其中 EvaluationController 主要支持教学平台前端页面的调用，而 EvaluationTaskController 负责支持教学平台服务端对接调用。逻辑层包括负责收集待评估的数据集的 UserEvaluationLogic、负责上传文件至 OSS 服务器的 OSSLogic 以及负责创建、重试评估任务以及查询进度的 EvaluationTaskLogic，EvaluationTaskLogic 主要调用 EvaluationService 的方法实现业务逻辑，TaskProgressVOWrapper 是对返回评估任务进度状态的包装类，此外还包括数据库访问层类 EvaluationTaskRepository 以及描述评估任务以及任务状态的实体类。

表 4.5: 评估任务控制模块主要类

分类	主要包含的类
控制器层	EvaluationController、EvaluationTaskController
逻辑层	UserEvaluationLogic、OSSLogic、EvaluationTaskLogic
服务层	EvaluationTaskService、TaskProgressVOWrapper
数据访问层	EvaluationTaskRepository
数据类	EvaluationTask、TaskProgressVO

表 4.6是对 EvaluationTask 类详细设计的描述，表示对应一场考试的测试评估任务对象，作为实体类存储于 MongoDB 中，属性包括评估任务的编号、对应的考试编号、创建时间、状态、状态描述以及对应的待评估数据的 OSS url 地址等基本信息。

如图 4.12所示为评估任务控制模块的核心类图，由于篇幅限制这里主要对软件测试评估服务涉及的核心类图进行描述，EvaluationTaskController 接收并处

表 4.6: EvaluationTask 类详情列表

字段	含义	类型	备注
id	任务编号	ObjectId	主键，不为空
examId	考试编号	Long	评估任务对应的考试编号
createTime	创建时间	Long	评估任务创建的时间戳
status	任务状态	Integer	任务的状态
reason	状态描述	String	任务状态描述，如下载完成、解析完成以及任务失败的原因等
ossUrl	评估数据集地址	String	对应评估任务的带评估数据集的 OSS 存储地址

理评估任务的创建、重试以及查询进度相关请求，EvaluationTaskLogic 定义了对应的业务逻辑功能接口，由 EvaluationTaskLogicImpl 实现类通过调用 EvaluationTaskService 类中的方法具体实现，创建评估任务的逻辑主要为先查询数据库中是否有对应请求中 examId 的评估任务，如果有直接将其查询的状态通过 TaskProgressVOWrapper 包装类包装为对应页面进度的 TaskProgressVO 的页面显示对象返回，若无则通过 createTask 方法创建评估任务并通过包装类包装返回，执行评估任务的逻辑通过 EvaluationTaskService 中定义的 dealTriggerEvaluation 异步方法实现，执行过程中支持轮询查看任务进度的接口。

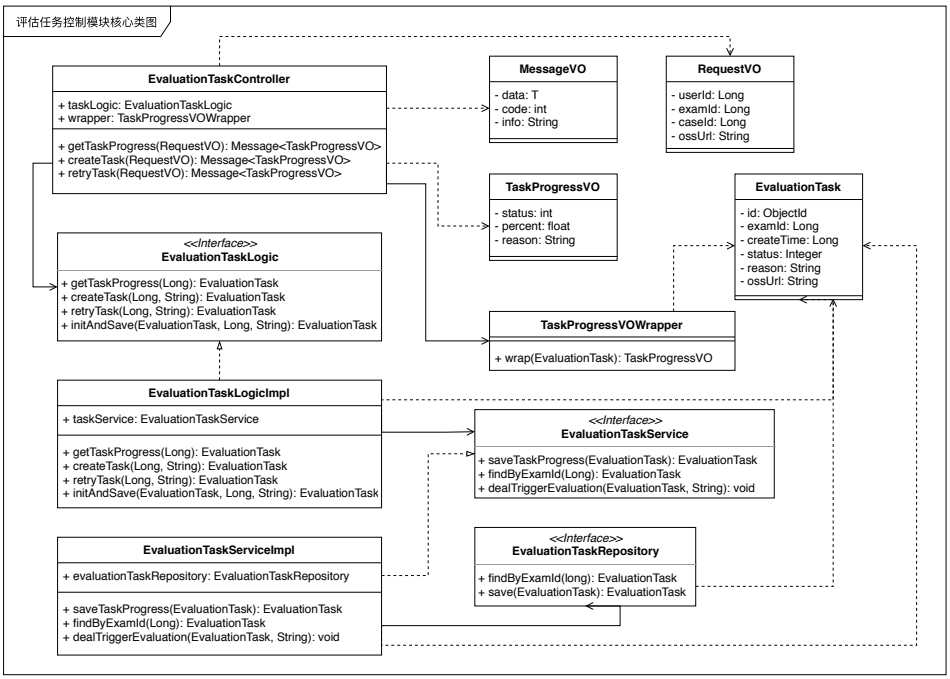

图 4.12: 评估任务控制核心类图

dealTriggerEvaluation 方法使用 Spring Boot 中的 @Async 注解来实现异步方法，添加 @EnableAsync 启用异步注解，在调用此方法时会另起一个线程来执行，通过自定义线程池设置核心线程数、最大线程数以及缓冲队列，在异步调用的 dealTriggerEvaluation 方法添加 @Async 注解时指定线程池名称。

4.3　测试效果评估模块

4.3.1　测试效果评估模块概述

测试效果评估模块作为本系统的核心模块，主要负责建立软件测试评估指标体系，获取待评估的数据集，通过评估脚本进行数据预处理、提取测试代码以及行为特性，并根据指标来对测试效果的多个维度计算给出各项分值，最后应用标准化方法统一指标分数的值域区间并保存至数据库中，整个模块从流程上来说作为评估任务执行的过程。

4.3.2　测试效果评估模块设计与实现

如图 4.13所示是测试效果评估模块的时序图，评估任务执行时调用的 dealTriggerEvaluation 方法便是测试效果评估模块的入口，首先调用 OSSUtil 工具类的 downloadByUrl 方法来下载待评估的数据集，下载成功后通过调用 EvaluationTaskRepository 操作数据库更新任务状态，将数据集解析后再次更新任务状态，获取数据集中的 caseDataList 字段循环对考试中的试题进行评估，根据试题的 targetType 和 caseType 字段通过 TestTypeService 的方法查询对应的测试类型，通过调用 AnalysisDetailService 类中的 getDetailByExamId 方法获取当前考试多类型分析结果的详细信息，在定义的 ScriptPathConstant 常量类中找到对应的评估脚本文件地址，调用 ScriptUtil 类中的 executeScript 方法执行对应的评估脚本，脚本执行结果以获取脚本执行的输出为准，若执行过程发生异常则更新任务状态为失败，否则解析输出为 EvaluationResult 集合的形式通过 EvaluationResultService 的 saveOrUpdateEvaluationResultList 进行保存，并更新评估任务状态，整个 dealTriggerEvaluation 方法逻辑若执行完成且过程没有发生异常，则任务到达最终状态评估成功，否则前面任意步骤出错，任务状态都会更新为评估失败，记录失败原因并且结束方法执行。

测试效果评估模块中主要涉及到的类如表 4.7所示，服务类包括实现评估任务控制的 EvaluationTaskService 类，这里主要是利用其 dealTriggerEvaluation 方法，负责根据试题以及测试目标的相关信息判断其测试类型的 TestTypeService，

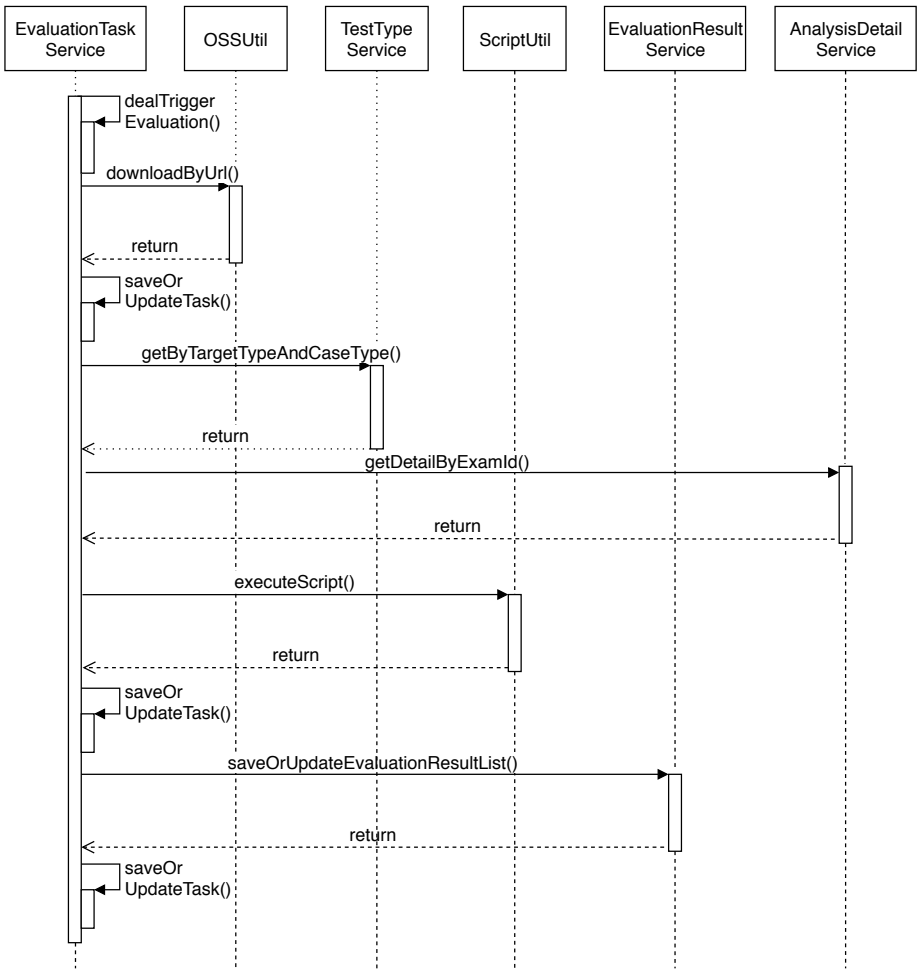

图 4.13: 测试效果评估模块时序图

负责对评估结果增删改查的 EvaluationResultService 以及负责查询分析结果详细信息的 AnalysisDetailService。工具类包括负责 OSS 上传下载操作的 OSSUtil 和调用执行评估脚本的 ScriptUtil。数据库访问类 EvaluationTaskRepository、TestTypeeRepository、EvaluationResultRepository 以及 AnalysisDetailRepository 分别负责数据类 EvaluationTask、TestType、EvaluationResult 以及 AnalysisDetail 的数据库操作。

EvaluationResult 类的详细设计描述如表 4.8所示，表示一个学生在一场考试中的一个试题的测试评估结果，作为实体类存储于 MongoDB 中，属性包括评估结果的编号、对应的考试编号、试题编号、用户编号、测试类型编号以及评估指标得分数据集合等，这里主要列出这几个主要属性，为了在评估结果可视化页面内容显示得更加丰富，还包括考试基本信息、学生基本信息、考试分数及排

表 4.7: 测试效果评估模块主要类

分类	主要包含的类
服务类	EvaluationTaskService、TestTypeService、EvaluationResult-Service、AnalysisDetailService
工具类	OSSUtil、ScriptUtil
数据访问类	EvaluationTaskRepository、TestTypeRepository、Evaluation-ResultRepository、AnalysisDetailRepository
数据类	EvaluationTask、EvaluationResult、TestType、AnalysisDetail

名、试题基本信息等相关属性。其中评估指标得分类的描述如表 4.9所示，包括评估指标编号、名称和对应的分数。

表 4.8: EvaluationResult 类详情列表

字段	含义	类型	备注
id	结果编号	ObjectId	主键，不为空
examId	考试编号	Long	评估结果对应的考试编号
caseId	试题编号	Long	评估结果对应的试题编号
userId	用户编号	Long	评估结果对应的用户编号
testTypeId	测试类型编号	Integer	评估结果对应的测试类型编号
graphData	指标得分数据	List<GraphData>	评估结果中多个指标得分的集合

表 4.9: GraphData 类详情列表

字段	含义	类型	备注
id	编号	int	评估指标的编号
name	名称	String	评估指标的名称，比如覆盖召回率、运行效率等
value	分数	Double	评估指标的分数

如图 4.14所示是测试效果评估模块中的核心类图，EvaluationTaskService 中的 dealTriggerEvaluation 方法作为此模块的触发点，是评估任务开始执行的逻辑，任务执行过程中涉及到任务状态的更新使用 EvaluationTaskService 的 saveOrUpdateTask 来实现，具体通过 EvaluationTaskRepository 完成评估任务的数据库更新。通过调用 OSSUtil 工具类的 downloadByUrl 方法下载待评估的数据集，下载成功后对数据集解析，利用 TestTypeService 类中的 getByTargetTypeAndCaseType 方法获取考试中试题的测试类型，调用 ScriptUtil 的 executeScript 方法执行评估脚本并收集脚本执行的输出，其代码如图 4.15所示，若脚本执行无异常，则

将输出解析成 EvaluationResult 集合，利用 EvaluationResultService 的 saveOrUpdateEvaluationResultList 方法对评估生成的数据保存，主要通过 EvaluationResultRepository 完成对评估结果的数据库更新。

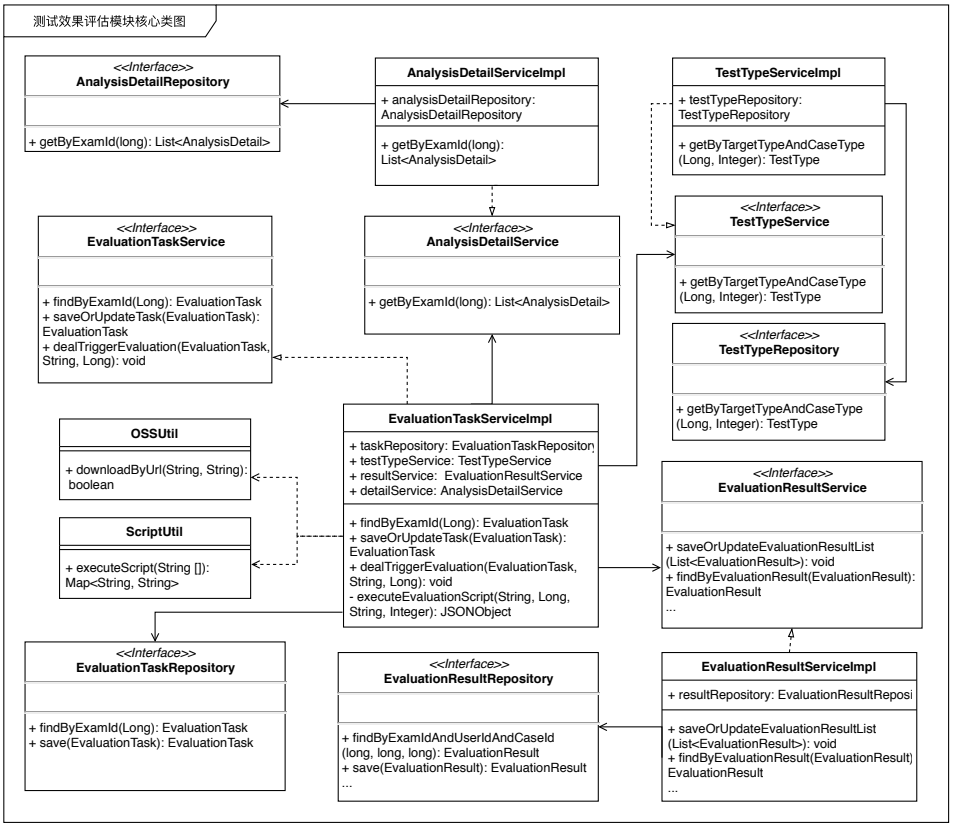

图 4.14: 测试效果评估模块核心类图

4.3.3　测试效果评估脚本的设计与实现

以上为测试效果评估模块的输入获取以及输出存储的相关内容，下面就评估脚本的主要逻辑进行详细阐述。针对不同测试类型提供相应的评估脚本，评估脚本的主要逻辑都是相同的，区分点在于根据评估指标计算指标分数的逻辑不同，主要包括预处理、指标分数计算、分数标准化及结果输出四个部分。

预处理涉及到数据清洗、数据转换和数据整理，针对部分数据缺失的情况，对数值型数据采用零值填充，对时间型数据采用相同列的最大值填充，对时间类型的数据的格式进行转换，统一为时间戳的方式，方便计算编码时长即提交时间与考试开始时间的差值，其后将数据整理为某个学生 Id 与对应的测试代码多个类型分析结果的形式，统一处理为 Pandas 的 DataFrame 格式，其中分析结果只选取方便后期应用公式的基础数据部分，比如变异分析结果选取变异体总

```
public static Map<String, String> executeScript(String cmds[]) {
    // 执行脚本结果通过 Map<String, String>形式返回
    Map<String, String> res = new HashMap<>();
    try {
        // 使用 Runtime 和 Process 类执行脚本
        Process process = Runtime.getRuntime().exec(cmds);
        log.info("pid is: ", process);
        // getInputStream()得到执行脚本控制台的输出信息
        BufferedReader in = new BufferedReader(new
InputStreamReader(process.getInputStream()));
        String stdOut;
        String info = "";
        while ((stdOut = in.readLine()) != null) {
            log.info(stdOut);
            info = stdOut;
        }
        in.close();
        res.put("info", info);
        // 同样的，通过 getErrorStream()得到执行脚本控制台的错误信息将其存入 Map
        ...
        res.put("errorInfo", errorInfo);
    } catch (Exception e) {
        e.printStackTrace();
        res.put("errorInfo", e.getMessage());
    }
    return res;
}
```

图 4.15: executeScript 方法关键代码

数和杀死的个数，通过二者便可计算缺陷检测率得分。

指标分数计算以分析结果数据为基础，应用第三章提出的测试评估指标中罗列的公式得到七个维度的得分，其中移动应用自动化测试的覆盖召回率和准确率计算公式中 α 取 0.4。以测试可维护性计算为例，如图 4.16所示为可维护性计算的关键代码。由于不同指标分数初步计算得到后处于不同的值域内，为了使各指标处于同一数量级，适合综合对比评价，因此对分数应用标准化方法。常用标准化方法包括最小值-最大值标准化 (Min-Max Normalization)、Z 分数标准化以及小数定标标准化方法，这里选取最小值-最大值标准化方法对各指标分数进行线性归一，同一场考试同一试题中学生在每个维度上的表现分数最高的为 100 分，最低的为 0 分。结果输出主要对数据进行遍历整理为对应 EvaluationResult 评估结果实体类集合的格式作为评估脚本的输出，外部对脚本输出调用 JSONObject 的 parseArray 方法将其转为实体集合将其保存至 MongoDB 中。

```
# 处理 checkstyle 数据
def deal_checkstyle(check):
    df = pd.DataFrame(check)
    if df.empty or len(list(df)) <= 1:
        return df
    ......
    # 显示所有列
    for label, value in df.items():
        // 将违反规则项分类
        res = is_check_and_get_class(label)
        if res['is_check']:
            df[res['class']] = df[res['class']] + value
    df = df.reindex(columns=final_index)
    df2 = df.copy()
    // 应用测试可维护性计算公式
    for label, value in df2.iloc[:, 2:].items():
        df2[label] = 1 - df2[label] / df2[label].max()
    for i in range(df2.shape[0]):
        df2.iloc[i, 1] = df2.iloc[i, 2:].sum() / len(checkstyle_index) * 100
    df.iloc[:, 1] = df2.iloc[:, 1]
    return df
```

图 4.16: 测试可维护性计算关键代码

4.4　评估结果可视化模块

4.4.1　评估结果可视化模块概述

评估结果可视化模块以展示测试评估结果及用户交互界面为主，前端基于 Vue 框架开发，结合 Echarts 可视化库将测试评估的结果数据渲染展现至页面提供给用户查看。前端通过 Vue Router 路由管理器对页面组件进行路由配置，涉及到多页面共享的属性状态利用 Vuex 实现状态管理。前后端通过 RestFul 接口进行数据的交互，前端发送请求后将处理过的响应数据赋值给页面变量，Vue 框架通过响应式原理追踪捕获到被修改的属性，通知属性对应的 watcher 实例将其重新渲染至页面。此外通过配置媒体查询实现页面的响应式，支持 PC 端和移动端查看。该模块主要分为测试评估数据总览、历史评估记录、测试评估结果以及测试评估指标体系四个部分，下面对其设计与实现进行详细说明。

4.4.2　评估结果可视化模块设计与实现

评估结果可视化模块采用前后端分离的架构来实现，该模块的交互时序图如图 4.17所示，主要涉及到评估结果、维度详情、历史记录以及测试评估数据总览页面。教师和学生在考试结果页面通过点击查看测试评估的按钮跳转至评估

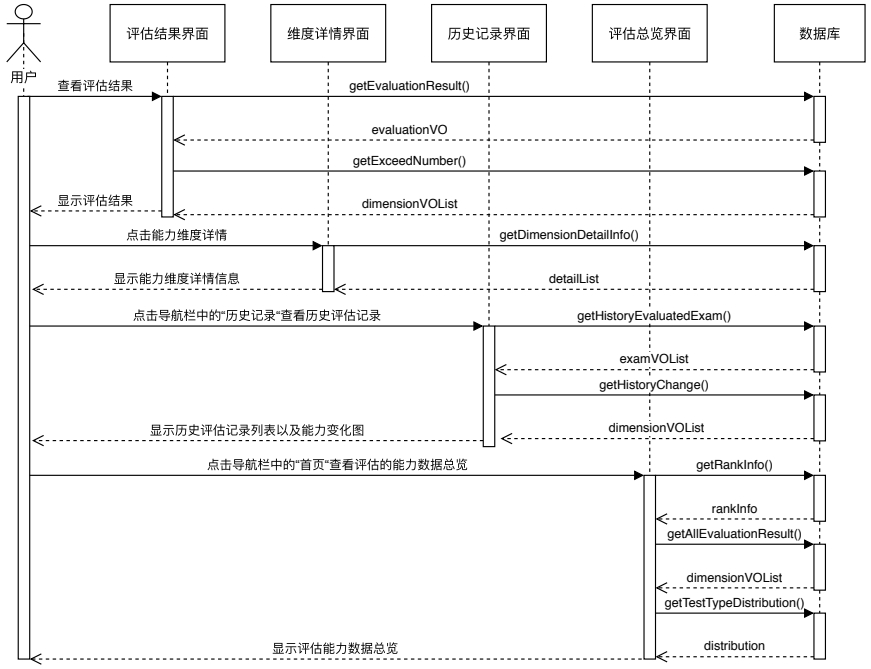

图 4.17: 评估结果可视化模块时序图

结果页面，主要接口为 getEvaluationResult 和 getExceedNumber，分别查询对应当前学生在当前考试的测试评估结果以及各个维度分数超过的同学数量，评估结果界面主要显示考试的基本信息、学生的考试得分排名、对应考试中每个试题的测试效果雷达图及维度分数超过学生的比例等，其界面如图 4.18所示。

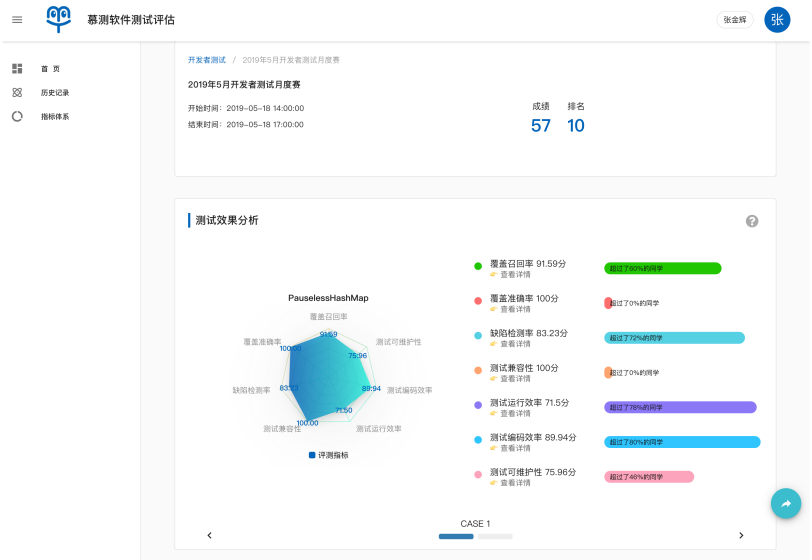

图 4.18: 评估结果界面示意图

其中测试效果雷达图采用 Echarts 可视化库实现，定义试题集合 caseList，每个试题都包括雷达图显示的配置信息 options 对象，请求 getEvaluationResult 接口得到相应数据后，通过 dealCaseGraph 方法对每个试题的 options 对象中的属性进行赋值，代码如图 4.19所示，由于 JavaScript 中对象赋值是将引用指向了同一个地址，若直接赋值会影响到其他试题中的 options 对象，因此这里采用深拷贝的方式对 options 对象赋值，针对 Vue 中数组和对象实例直接更改属性值不会触发更新的情况，采用 Vue 包装过的 splice() 方法使数组更新可以被检测到。

```
// 处理 getEvaluationResult 请求响应数据中的试题集合
dealCaseGraph(caseList) {
    // 针对 Vue 中已创建的数组和对象实例直接更改属性值不会触发更新的情况
    // 这里采用 Vue 包装过的 splice()方法使数组更新可以被检测到
    this.caseList.splice(0, this.caseList.length);
    for (let i = 0; i < caseList.length; i++) {
        let item = caseList[i];
        // dealGraphData 将能力数据处理成 Echarts 中 options 对应的数据格式
        let [indicator, data, dimensions] = this.dealGraphData(
                item.graphData
        );
        item.dimensions = dimensions;
        data = data.map(item => item.toFixed(2));
        options.radar.indicator = indicator;
        options.series[0].data[0].value = data;
        // 通过 deepCopy 方法来实现对象的深拷贝
        item["options"] = deepCopy(options);
        // 采用 Vue 包装过的 push()方法使数组更新可以被检测到
        this.caseList.push(item)
    }
}
```

图 4.19: dealCaseGraph 方法关键代码

用户点击查看详情按钮跳转至维度详情页面，查看到对应此维度的详细信息，主要涉及覆盖召回率、覆盖准确率、编码效率、缺陷检测率、兼容性和可维护性这几个维度。前端请求 getDimensionDetailInfo 接口获取详情信息数据进行可视化展示。以缺陷检测率为例，其界面如图 4.20所示，以饼状图的形式分别展现变异体杀死数量和存活数量的比例，同时还展示对应的变异算子种类的介绍、变异体分布以及对应的源代码和变异代码列表，具体为运算符型、数值型、方法返回值型等变异体的占比情况。

用户点击导航栏中的历史记录按钮跳转至历史评估记录页面，可以分测试类型查看历史记录以及最近 5 次的测试效果变化趋势图，界面如图 4.21所示。前

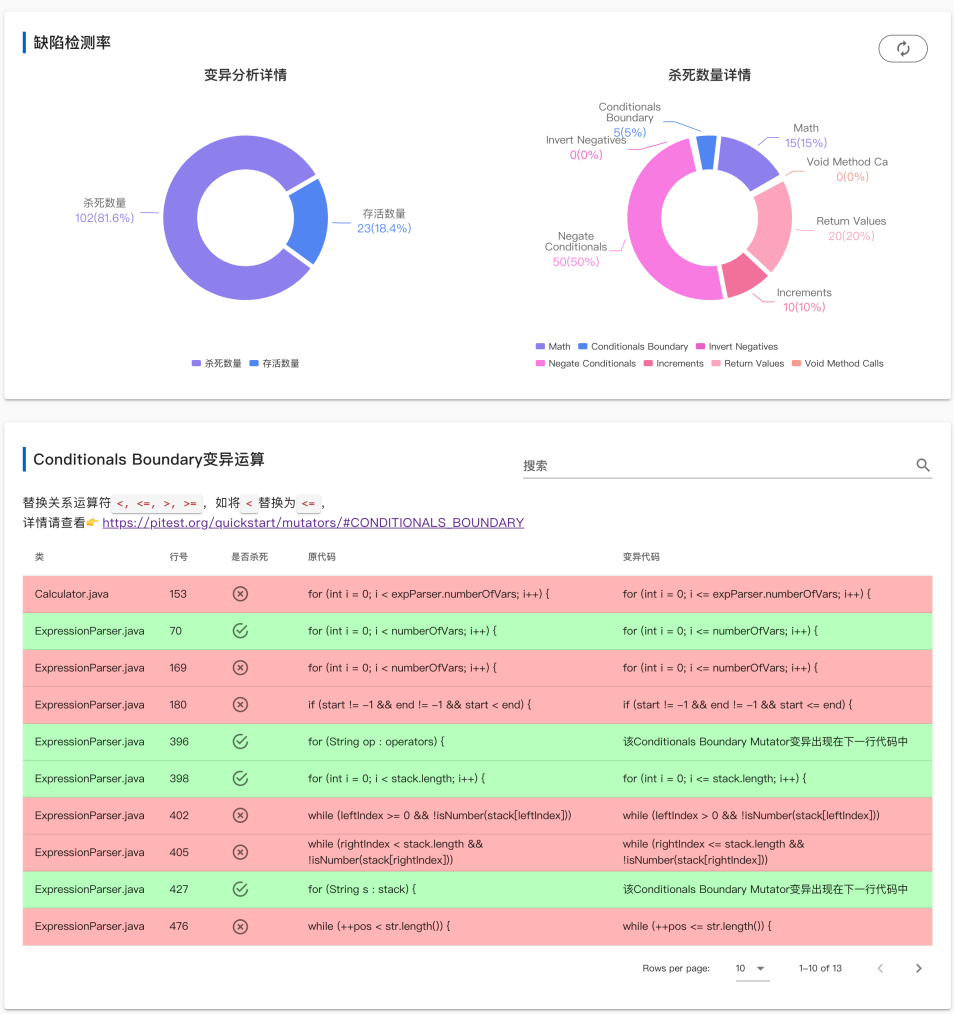

图 4.20: 缺陷检测率详情界面示意图

端请求 getHistoryEvaluatedExam 接口获取历史评估记录，将其以列表的形式展现。点击列表中的记录即可跳转至对应的评估结果页面，测试效果变化图展示当前测试类型的多个维度的变化曲线，通过 getHistoryChange 接口获取数据，以折线图的形式展现。页面包括多个测试类型的按钮组，点击便可切换查看不同测试类型的历史记录和测试效果变化图。

用户点击导航栏中的首页按钮跳转至测试评估数据总览页面，页面从上到下分别展示个人基本信息、测试评估总览图、测试评估类型占比图以及最近评估记录等模块，界面如图 4.22所示。个人基本信息除了姓名、学校、头像以外通过请求 getRankInfo 接口获取到学生的最近排名和历次最佳排名信息。测试评估总览图是将学生历次参加的考试评估结果以雷达图的形式展现，前端调用 getAllDimensionResult 接口获取数据，可以点击雷达图中的图例切换测试类型查

图 4.21: 历史评估记录界面示意图

看测试评估的总数据。测试评估类型占比图以四分之三圆环图展现历次评估测试类型的占比情况，调用 getTestTypeDistribution 接口获取数据。以上需要可视化的数据都是经过相应的处理再赋值给对应图表的 option 对象，Vue 框架的监听器捕获到数据的变化对页面进行渲染更新。

图 4.22: 测试评估数据总览界面示意图

此外，用户点击导航栏中的指标体系按钮便可查看软件测试指标体系界面，罗列每个测试类型的评估指标，以静态页面的形式展示，点击指标按钮便会显示相应的指标介绍。

4.5　本章小结

本章对系统的测试代码分析模块、评估任务控制模块、测试效果评估模块以及评估结果可视化模块的详细设计与具体实现进行了详细的阐述，通过时序图、主要类表和核心类图配合说明，给出了关键函数的代码以及系统页面截图。

第五章　软件测试多维评估系统测试与案例分析

基于前文第三章、第四章的需求分析、设计以及具体实现，本章主要介绍软件测试多维评估系统的系统测试与案例分析。

5.1　系统测试

5.1.1　测试目标

基于测试环境对系统有效性进行测试，设计测试用例并执行，检测系统的核心功能模块的实现是否可以满足系统需求，包括测试代码分析、评估任务控制、测试效果评估以及评估结果可视化页面的查看等功能，通过测试来保障系统的质量。

5.1.2　测试环境

如表 5.1所示为系统测试环境配置，软件测试评估服务和在线测试教学平台的前端和服务端项目均以 Docker 容器化的方式部署于阿里云服务器运行，利用 Jenkins 持续集成。项目前端与服务端之间采用 Nginx 提供的反向代理进行前后端交互。系统采用 Celery 分布式系统实现异步任务的调度，以 RabbitMQ 为消息中间件，保证系统的高可用性和稳定性。数据存储方面，系统采用 MySQL 数据库存储学生代码提交记录相关信息，MongoDB 来存储分析任务、代码分析结果、评估任务以及评估结果等数据，并使用 Redis 作为缓存服务存储分析任务及分析结果。

表 5.1: 系统测试环境

参数	参数详情
云服务器	阿里云服务器
操作系统	Ubuntu 16.04 64 位
CPU	4
内存	16G
数据库	MySQL 5.7.18、MongoDB 4.0.6、Redis 5.0.3
容器	Docker 19.03.8
其他软件	Nginx 1.15.7、Celery 4.4.0、RabbitMQ 3.7.4

5.1.3　功能测试

功能测试的测试用例主要针对不同模块的功能进行设计，通过给定输入和操作对比期望的输出和实际输出，来验证功能的正确性，对系统中可能出现的缺陷进行修复保证功能符合产品预期，提高系统的稳定性。主要从测试代码分析、评估任务控制、测试效果评估和评估结果可视化展开，其中评估任务控制和测试效果评估是一个连续的过程，评估任务控制包括任务的创建、重试以及进度查询，而测试效果评估是作为评估任务执行的过程，将二者放一起进行测试。下面就测试代码分析、测试效果评估和评估结果可视化三个部分相关测试用例进行描述。

（1）测试代码分析

表 5.2是测试代码分析功能的测试用例，主要验证测试代码分析任务的创建、停止、重试以及进度查询功能的正确性，包括创建测试代码分析任务、停止分析任务、重试分析任务等测试项，实际测试结果符合预期，测试用例全部通过。

表 5.2: 测试代码分析功能测试用例

测试项	操作/输入	预期结果	测试结果
创建测试代码分析任务	1. 管理员登录； 2. 点击侧边栏中信息管理下的计算分析按钮； 3. 输入待分析的考试 Id 查询； 4. 选择其中一个试题后面的分析操作按钮；	1. 登录成功； 2. 进入计算分析页面； 3. 显示试题分析列表； 4. 进入测试代码分析页面，可以查看到任务总进度以及每个学生测试代码分析任务的进度。	通过
停止测试代码分析任务	点击停止按钮	页面提示“正在停止”，停止分析任务，进度条保持不动	通过
重试测试代码分析任务	点击重试按钮	页面提示“正在重试”，重新执行分析任务，进度条从头开始	通过

(2) 测试效果评估

表 5.3是测试效果评估功能的测试用例，主要验证测试评估任务的创建、重试、进度查询以及评估结果可视化页面入口显示功能的正确性。测试项包括创建测试评估任务和重试评估任务，实际测试结果符合预期，测试用例全部通过。

(3) 评估结果可视化

表 5.4是查看评估结果页面的测试用例，主要验证评估结果页面展示功能的正确性。评估结果页面从在线测试教学平台的考试结果页面进行跳转，由于一

表 5.3: 测试效果评估功能测试用例

测试项	操作/输入	预期结果	测试结果
创建测试评估任务	1. 教师登录； 2. 点击侧边栏中“考试”下的“管理考试”按钮； 3. 在考试列表中选择一场考试点击； 4. 点击“测试评估”按钮； 5. 点击是否开始评估对话框的“确认”按钮； 6. 点击评估任务完成对话框的“确认”按钮；	1. 登录成功； 2. 进入考试列表页面； 3. 进入考试结果页面，若考试尚未评估，显示“测试评估”按钮； 4. 页面显示“当前考试尚未被评估，是否开始评估?”的提示对话框； 5. 对话框关闭，页面提示“正在创建评估任务”以及“评估任务创建成功”，同时展示评估进度，在评估任务完成后，页面弹出评估任务执行完毕，任务状态成功的提示框； 6. 学生列表中显示评估结果页面入口图标按钮。	通过
重试测试评估任务	1. 点击“重新评估”按钮； 2. 点击是否重新评估对话框的“确认”按钮； 3. 点击评估任务完成对话框的“确认”按钮；	1. 页面弹出“当前考试已评估，是否重新评估?”的提示对话框； 2. 对话框关闭，页面提示“正在重试”以及“重新评估成功”，同时展示评估进度，在评估任务完成后，页面弹出评估任务执行完毕，任务状态成功的提示框； 3. 学生列表中显示评估结果页面入口图标按钮。	通过

场考试涉及到多个试题，以滑动窗口的形式展示，每一个试题有对应的测试评估多维雷达图以及维度分数超过同学的比例图。点击维度名称可以跳转到查看维度详情信息页面，因此测试项包括跳转评估结果页面、切换试题查看评估结果以及查看维度详细信息，实际测试结果符合预期，测试用例全部通过。

表 5.5是查看历史评估记录的测试用例，验证历史评估页面功能的正确性。主要包括历史评估记录和测试效果变化趋势图的查看、切换测试类型查看对应的历史记录以及选择其中一个评估记录跳转至评估结果页面的测试项，实际测试结果符合预期，测试用例全部通过。

表 5.4: 查看评估结果测试用例

测试项	操作/输入	预期结果	测试结果
跳转评估结果页面	1. 教师登录； 2. 点击侧边栏中 “考试” 下的 “管理考试” 按钮； 3. 在考试列表中选择一场考试点击； 4. 点击学生列表中的任意学生对应的评估结果页面入口图标按钮；	1. 登录成功； 2. 进入考试列表页面； 3. 进入考试结果页面，已经评估过的考试的学生列表中会有评估结果页面入口图标按钮； 4. 页面跳转至评估结果页面，展示学生在当前考试的评估结果信息。	通过
切换试题查看评估结果	选择查看不同试题的评估分析结果	滑动展示相应试题的评估分析结果	通过
查看维度详细信息	点击维度指标名称	页面跳转至维度详细信息页面，以缺陷检测率为例，以饼图展示变异体杀死和存活的比例以及对应的变异体种类的占比，以列表展示变异体在试题的各个 Java 类中的原代码和变异代码详情。	通过

表 5.5: 查看历史评估记录测试用例

测试项	操作/输入	预期结果	测试结果
查看历史评估记录	点击导航栏中的 “历史记录”；	默认展示开发者单元测试的历史评估记录和对应的测试效果变化趋势图	通过
查看不同测试类型的评估记录	切换不同测试类型按钮	展示相应类型的评估记录	通过
查看评估结果详情	点击列表的任意一行记录	进入对应的评估结果页面	通过

表 5.6是查看测试评估数据总览的测试用例，对应评估结果可视化前端项目的首页。该测试用例主要验证测试评估数据总览页面功能的正确性，包括查看测试评估数据总图、评估测试类型分布图以及查看最近评估记录的测试项，实际测试结果符合预期，测试用例全部通过。

表 5.6: 查看测试评估数据总览测试用例

测试项	操作/输入	预期结果	测试结果
查看测试评估数据总览	点击导航栏中的“首页”	展示测试评估数据总览页面，包括学生最近排名和最佳排名、测试评估数据总图、评估测试类型分布图以及最近评估记录	通过
查看最近评估记录	点击任意测试类型的图标	若有相应类型的最近评估记录则进入最近一次的测试评估结果页面，否则提示暂无当前类型的最近评估记录	通过

表 5.7是查看软件测试评估指标体系的测试用例，主要验证软件测试评估指标体系页面功能的正确性。测试项为查看软件测试评估指标体系以及切换指标显示指标的介绍信息，实际测试结果符合预期，测试用例全部通过。

表 5.7: 查看软件测试评估指标体系测试用例

测试项	操作/输入	预期结果	测试结果
查看软件测试评估指标体系	点击导航栏中的“指标体系”	展示软件测试评估指标体系页面，包括三种测试类型的测试评估指标介绍	通过
切换指标查看指标介绍信息	点击任意指标项	显示相应的指标介绍信息	通过

5.2　案例分析

为了进一步验证本系统在真实场景的可用性，本节结合公司软件测试教学平台丰富的考试案例进行评估。首先从教学平台的历史考试中选取了 14 场考试，其中开发者单元测试 10 场，移动应用自动化测试 3 场，Web 应用自动化测试 1 场，涉及参与考试学生 718 人次，提交记录 26666 条。

下面对开发者单元测试类型的一次考试评估结果进行具体分析，考试名称为“2019 年 03 月开发者测试月度赛”，参加考试的学生在各指标维度得分如表 5.8所示，使用 Kolmogorov-Smirnov 检验方法 [32] 检验维度评分是否符合正态分布，当应用 Kolmogorov-Smirnov 方法得到的 pvalue 值大于 0.05 则认为评分满足正态分布。从表中可以看到覆盖召回率、缺陷检测率、测试编码效率和测试可维护性满足正态分布。由于开发者单元测试要求测试代码对生产代码全覆盖，根据第三章评估指标计算公式 3.2可以得到每个学生覆盖准确率都为 100 分。前

文提到关于开发者测试的兼容性主要体现在 JDK 版本的不同，由于教学平台已经指定 JDK 版本规避此问题，因此兼容性不纳入为评估项，为了雷达图的可观性，兼容性评分都取 100 分。

表 5.8: 2019 年 03 月开发者测试月度赛评估结果

	覆盖召回率	覆盖准确率	缺陷检测率	测试兼容性	测试运行效率	测试编码效率	测试可维护性
$student_{01}$	97.4359	100	100	100	87.8137	27.4972	0
$student_{02}$	87.1795	100	81.6327	100	90.5771	18.1328	55.9830
$student_{03}$	69.2308	100	73.4694	100	62.3004	52.3687	46.3209
$student_{04}$	82.0513	100	87.7551	100	81.0956	21.6308	73.6171
$student_{05}$	87.1795	100	89.7959	100	89.4745	24.0126	67.0953
$student_{06}$	94.8718	100	91.8367	100	95.6501	100	70.4578
$student_{07}$	94.8718	100	93.8776	100	100	34.5956	65.1984
$student_{08}$	69.2308	100	71.4286	100	74.4803	16.8468	34.3454
$student_{09}$	79.4872	100	69.3878	100	82.6099	14.7380	95.0427
$student_{10}$	100	100	95.9184	100	99.7572	19.9190	65.1210
$student_n$	...	...	...	...	...	...	...
mean	53.5960	100	59.1350	100	54.4358	19.0977	79.2349
std	37.9700	0	37.4481	0	38.3363	18.8917	21.6706
kstest	0.1023	NaN	0.0537	NaN	0.0445	0.2330	0.0949

基于测试评估得分是以计算标准分为结果，采取的是相对得分，因此在对各维度分析的过程中，采用两两比对的方式进行分析。这里我们选取学生 A 和学生 B 在试题“SplayTreeMap”中的测试效果分析，他们在试题中的测试效果多维雷达图如图 5.1所示。由于测试准确率和测试兼容性两个维度都为 100，这里主要对其他五个维度进行比较。

传统教学中以代码覆盖率作为衡量测试效果优劣的评价依据，公司软件测试教学平台引入了变异分析作为其中一个分项，那么基于这个前提，从图中可以看到学生 A 在覆盖召回率和缺陷检测率得分都比学生 B 要高，所以按照传统的评分规则，学生 A 的总分则高于学生 B。但是这样的评价是不全面的，结合二者在测试兼容性、测试运行效率、测试可维护性三个维度上的表现来看，明显学生 B 在三个维度上的表现是更优的。下面结合每个维度的详情数据对比进行分析，并以学生 A 为例给出具体计算过程。

首先是覆盖召回率的详情数据，利用代码覆盖工具将源代码和测试代码转换成 Node 节点，测试代码覆盖的节点与被测代码节点重合即为覆盖节点，否则为未覆盖节点。学生 A 已覆盖 39 个节点，未覆盖 47 个节点，学生 B 已覆盖 38

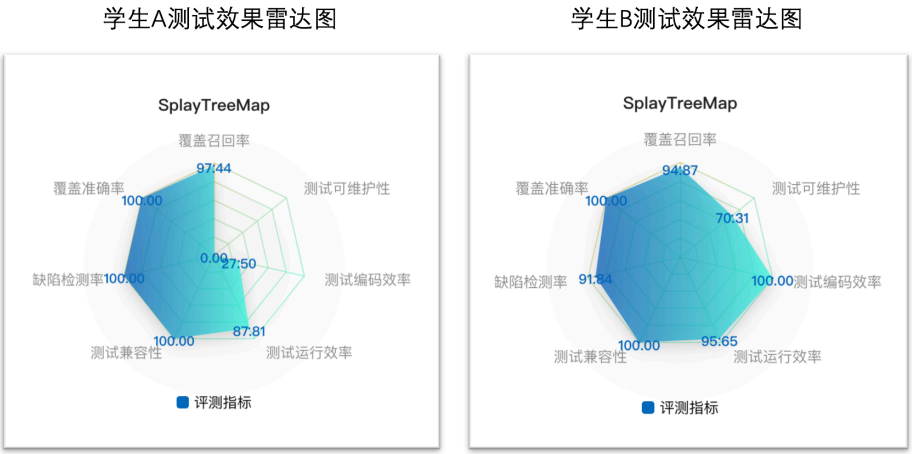

图 5.1: 学生 A、B 测试效果多维图对比

个节点，未覆盖 48 个节点。根据开发者单元测试的覆盖召回率计算公式 3.1，其中 $Node_{covered} \cap Node_{meta}$ 即已覆盖的节点数量 39，$Node_{meta}$ 为源代码节点个数 86，计算得到学生 A 的覆盖率召回率原始得分为 45.34，同理计算得到学生 B 的覆盖率召回率原始得分为 44.18，应用标准化方法后可以分别得到标准分 97.44 分和 94.87 分，可以看出学生 A 的覆盖率召回率高于学生 B。

其次是缺陷检测率的详情数据，利用变异分析工具运行测试代码，分析得到变异体杀死的比例，变异体杀死率越高，说明测试代码的检错性越强。除了变异体杀死存活的分布情况，系统还支持展示每个变异体的源代码和变异代码。变异体总数为 118，学生 A 变异体杀死数量为 50，学生 B 变异体杀死数量为 46，相差 4 个，约占总数的 3%。根据缺陷检测率计算公式 3.7，结合学生 A 分析结果数据，$Mutant_{killed}$ 对应变异体杀死数量 68，$Mutant_{total}$ 对应变异体总数 118，计算得到学生 A 的缺陷检测率原始得分为 42.37，同理学生 B 的缺陷检测率原始得分为 38.98，应用标准化方法后可以分别得到标准分 100 分和 91.84 分，可以看出学生 A 的缺陷检测率高于学生 B。

再次是运行效率的详情数据，通过计算执行代码覆盖的运行时间，考察达到同样的测试效果即达到相同的覆盖召回率，测试代码运行的高效性。从图 5.2中可以看出，学生 A 代码运行时长为 50.288357 秒，覆盖召回率原始得分约为 45.34，学生 B 代码运行时长为 45.092494 秒，覆盖召回率原始得分约为 44.18 分。根据运行效率计算公式 3.9，$Coverage_{recall}$ 为覆盖召回率得分 45.34，$Time_{running}$ 为运行时间 50.288357 秒，计算得到学生 A 的运行效率原始得分为 90.16，同理学生 B 的运行效率原始得分为 97.98，应用标准化方法后可以分别得到标准分 87.81

分和 95.65 分。在覆盖召回率相似的情况下，学生 B 的运行效率高于学生 A。

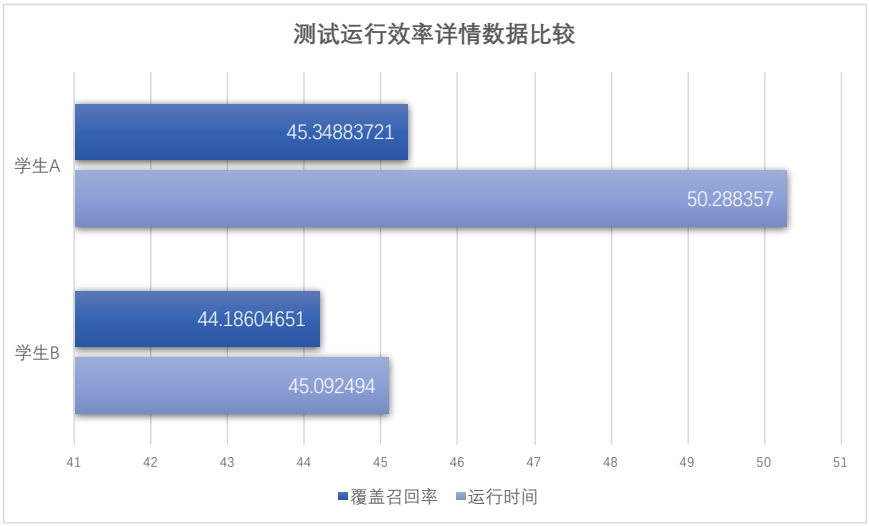

图 5.2: 学生 A、B 测试运行效率详情数据对比图

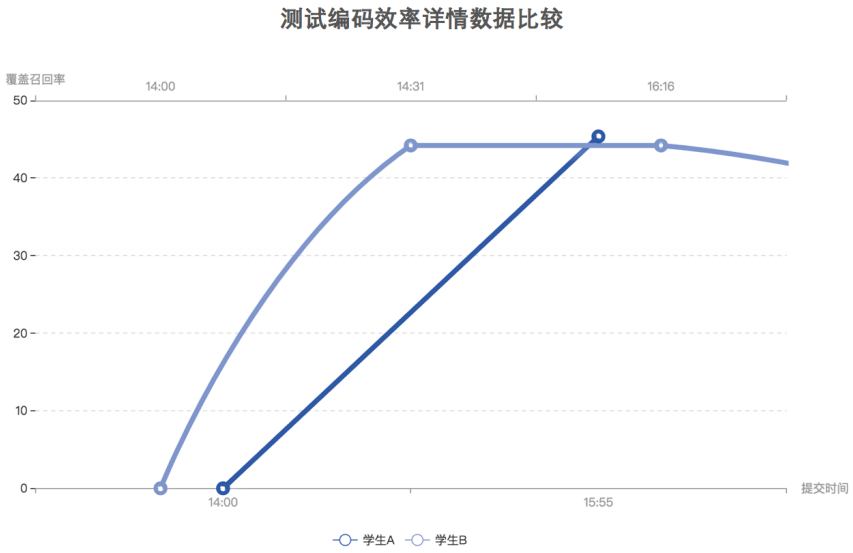

图 5.3: 学生 A、B 测试编码效率详情数据对比图

接着是编码效率的详情数据，由于学生测试框架及语法掌握的熟练程度不同等原因，其编码时长不同，通过计算覆盖召回率与编码时长的比值，获取单位覆盖得分的编码效率。考试开始时间为 14 点，从图 5.3中可以看出，学生 A 在考试开始 1 小时 55 分 13 秒时取得覆盖召回率得分 45.34 分。相比较学生 A，学生 B 在考试开始 31 分钟 15 秒覆盖召回率就达到 44.18 分，达到单位代码覆盖率

的编码时长更短，即编码效率更高。根据编码效率计算公式 3.10，$Coverage_{recall}$ 为覆盖召回率得分 45.34，$Time_{coding}$ 为编码时间，通过计算提交时间与考试开始时间的时间差得到，换算以秒为单位后为 6913.016 秒，应用公式计算得到学生 A 的编码效率原始得分为 0.655，同理学生 B 的编码效率原始得分为 2.35，应用标准化方法后可以分别得到标准分 27.50 分和 100 分。在覆盖召回率相似的情况下，学生 B 的编码效率高于学生 A。

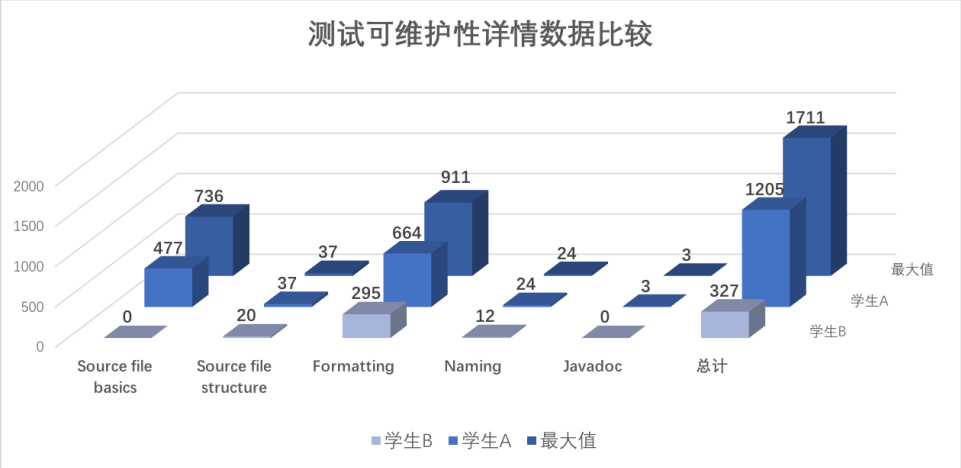

图 5.4: 学生 A、B 测试可维护性详情数据对比图

最后是测试可维护性的详情数据，良好的编码风格可以提高代码的可读性、可理解性、可维护性。利用编码风格分析工具对测试代码进行分析，分析得到违反的代码规则名称及对应的次数，主要对违反代码规则的次数进行比较。从图 5.4中可以看出，学生 A 的代码违反了 5 类代码规则，其中 664 个为格式化类问题，共违反了 1205 次。学生 B 的代码违反了 3 类代码规则，共违反了 327 次。根据可维护性计算公式 3.11，分别将每个违反代码规则类别对应的次数和最大值带入公式，计算得到学生 A 的可维护性原始得分为 12.46，同理学生 B 的可维护性原始得分为 2.35，应用标准化方法后可以分别得到标准分 0 分和 70.31 分。可维护性详情信息页面支持给出违反规则的分类以及具体代码行号信息，如图所示 5.5通过行号结合学生 A 的代码 5.6，可以看出其代码规则违反的情况。对比之后，可以得到学生 B 的测试可维护性更优的结论。

综上所述，在学生 A 和学生 B 在覆盖召回率及缺陷检测率上评分表现相近的前提下，通过对其他指标的比较可以进一步评估其测试效果，这也正是体现了多维评估的价值。代码覆盖作为检验测试充分性的标准之一，Andrew 提到高覆盖率并不能确保代码的质量，覆盖率很高的代码不一定不包含缺陷 [33]。而达到相同的代码覆盖率所耗费的时间也是不同的，实验经验证明测试耗费的时间

详细列表

搜索

类名	行号	检查项	建议
SplayTreeMap_ESTest	33	AnnotationLocationMostCases	Annotation 'EvoRunnerParameters' should be alone on line.
SplayTreeMap_ESTest	33	AnnotationLocationMostCases	Annotation 'RunWith' should be alone on line.
SplayTreeMap_ESTest	33	LineLength	Line is longer than 100 characters (found 106).
SplayTreeMap_ESTest	34	AbbreviationAsWordInName	Abbreviation in name 'SplayTreeMap_ESTest' must contain no more than '2' consecutive capital letters.
SplayTreeMap_ESTest	34	TypeName	Type name 'SplayTreeMap_ESTest' must match pattern '^[A-Z][a-zA-Z0-9]*$'.
SplayTreeMap_ESTest	39	LocalVariableName	Local variable name 'splayTreeMap_Entry0' must match pattern '^[a-z]([a-z0-9][a-zA-Z0-9]*)?$'.

图 5.5: 学生 A 的可维护性详情列表界面示意图

两个注解建议分别单独占一行，违反AnnotationLocationMostCases规则

```
32
33 @RunWith(EvoRunner.class) @EvoRunnerParameters(useVNET = true, separateClassLoader = true, useJEE =
   true)
34 public class SplayTreeMap_ESTest extends SplayTreeMap_ESTest_scaffolding {
35
36   @Test(timeout = 4000)
37   public void test00()  throws Throwable  {
38       Integer integer0 = new Integer(0);
39       SplayTreeMap.Entry splayTreeMap_Entry0 = new SplayTreeMap.Entry(integer0, "",
   (SplayTreeMap.Entry) null);
40       Object object0 = splayTreeMap_Entry0.setValue("");
41       assertNotNull(object0);
42       assertEquals("", object0);
43   }
44
```

违反TypeName命名规则，类名不能包含下划线

违反LineLength规则，该行超过100个字符

违反LocalVariableName命名规则

图 5.6: 学生 A 代码中违反代码规范实例

越长，检测出的缺陷也就越多，但是一味通过消耗大量时间才能发现缺陷，这并不是高质量的测试，因此在评估测试效果时将效率方面的考察纳入评估标准中，这里主要考察编码效率和代码运行效率。同时，无论是生产代码还是测试代码，要重视代码规范的遵守，培养良好的编码风格有利于提高代码的可读性、可维护性，尤其在团队开发中。

通过以上案例分析，进一步证明了系统在真实场景下良好的评估效果。系统基于学生的测试代码和测试行为，通过分析代码提取其特性，再应用提出的评估指标对测试效果计算评估，支持教师和学生查看测试评估的多维度及多粒度结果呈现。与单维度只考察代码覆盖率或者缺陷检测率相比，系统从多方面对学生的测试效果进行评估，提供直观清晰的评估结果，教师可以基于此更充分地给出教学反馈。同时，学生也可以明确地发现自己的不足，更有针对性地通过练习提升自己。

5.3　本章小结

本章首先对系统功能进行测试，根据功能模块设计对应的测试用例验证功能的有效性和完整性，其次基于在线测试教学平台丰富的考试数据进行案例分析，根据每个维度的详情数据信息比较学生的测试效果评估结果，验证多维度分析的全面性和有效性。

第六章　总结与展望

6.1　总结

软件测试作为保证和提高软件质量的重要手段 [7]，在整个软件开发过程中的重要性不言而喻。从教育层面来看，软件测试是一门以理论为主导，注重实际操作强调应用的学科。除了理论考试之外，高校以结合在线测试教学平台组织在线考试的方式鼓励学生提交测试代码，通过实战考察掌握程度。基于建设性调整理论，科学地设计评估标准可以带来反馈，带动学生的学习积极性，从而进一步提高其测试能力。出于以上考虑，本文针对学生在公司软件测试教学平台参加测试考试的场景，基于其测试代码和测试行为，设计并实现了针对开发者单元测试等三种主流测试类型的软件测试多维评估系统。

本文首先介绍了系统的项目背景和意义以及软件测试评估相关的研究现状，其后介绍了系统使用到的关键技术和工具，包括用于项目前端开发的 Vue 生态，支持评估结果可视化渲染的 Echarts 库，负责提取测试代码特性的代码分析工具 PITest 等，负责调度异步任务的 Celery 分布式任务管理系统以及支持应用容器化的 Docker。

本文在系统需求分析后对核心功能进行划分，包括测试代码分析部分、评估任务控制部分、测试效果评估部分和评估结果可视化部分。首先，测试代码分析部分选取了不同测试类型的测试代码分析工具，将其进一步封装成 Docker 镜像，以参加考试的学生对应一个试题提交的测试代码为主体创建异步分析任务，通过 Celery 进行异步任务的调度，调用分析工具获取分析的测试代码特性并存储。其次，评估任务控制部分负责对评估任务的创建、重试以及查看进度操作。测试效果评估模块作为评估任务执行的过程，从测试代码和测试行为两方面，提出面向不同测试类型的软件测试评估指标，以测试代码分析结果和学生提交测试代码的过程数据为输入，根据指标公式计算相应的维度得分，再通过应用标准化方法得到标准分。最后，由评估结果可视化部分来对生成的评估结果分粒度进行可视化，包括多维度指标得分、指标维度的详细信息、历次评估记录以及能力变化趋势图，结合图表的形式直观清晰地充分展示评估结果，提供教学反馈的作用。

系统目前已经与公司软件测试教学平台集成，提供软件测试评估的服务，可以有效地根据学生参与考试提交的测试代码以及提交记录数据多维度地评估其测试效果，提供反馈。

6.2 展望

软件测试多维评估系统已经作为公司软件测试教学平台的服务内测使用，本系统仍存在进一步改进和优化的地方，未来系统需要在以下方面作出优化和改进：

（1）本文针对三种主流类型的软件测试提出 7 个维度的评估指标，指标的提出参考了国内外关于软件测试评估方面的理论实证等。目前每一个指标通过调用代码分析工具提取代码特性，再应用指标公式得到分数，未来对应一个指标可以提取更多的代码特性结合起来进行分析，比如可维护性，当下通过考察违反代码规范的情况评估，可以结合测试代码坏味道、代码可读性来评估测试的可维护性。系统实现的异步分析任务调度机制可以支持扩展新的分析工具，实现更多类型的代码分析支持。根据大量评估结果数据，不断调整评估标准，提供评估的合理性。

（2）本文针对 Web 应用自动化测试兼容性评估，目前只提供了 Windows 操作系统下的 Chrome、Firefox 和 Internet Explorer 三种浏览器类型考察测试脚本运行的通过情况，对于其他操作系统、其他浏览器以及浏览器的版本方面尚未考察，鉴于系统在分析脚本兼容性是采用 Selenium Grid 工具，基于工具的灵活可扩展性，未来通过配置注册更多的测试实例 Node 节点，便可达到拓展丰富测试环境的效果，充分评估测试的兼容性。

（3）本文提出关于评估软件测试的指标，实现了结合测试代码和测试行为对学生参与测试类型考试中的测试效果进行多维度评估，因开发周期的限制，未对学生在查看测试评估结果后的考试成绩进一步分析及评估，只是将历史评估记录的数据进行简单地整合和展现，未考察学生在知悉多维度的评估结果后对其测试能力提升的反馈作用，不过本系统对评估多维度结果和学生考试成绩排名数据进行了收集，为未来分析设立评估指标对学生的测试能力提升的推动性提供了数据基础。未来将分析评估指标和测试能力的相关性，拓展指标评估的主体以及场景，不局限于教育场景内，向工业界迈进。

参考文献

[1] M. Ellims, J. Bridges, D. C. Ince, The economics of unit testing, Empirical Software Engineering 11 (1) (2006) 5–31.

[2] J. B. Biggs, Teaching for quality learning at university: What the student does, McGraw-hill education (UK), 2011.

[3] G. J. Myers, C. Sandler, T. Badgett, The art of software testing, John Wiley & Sons, 2011.

[4] M. R. Lyu, et al., Handbook of software reliability engineering, Vol. 222, IEEE computer society press CA, 1996.

[5] X. Cai, M. R. Lyu, Software reliability modeling with test coverage: Experimentation and measurement with a fault-tolerant software project, in: The 18th IEEE International Symposium on Software Reliability (ISSRE'07), IEEE, 2007, pp. 17–26.

[6] 安金霞, 王国庆, 李树芳, 朱纪洪, 基于多维度覆盖率的软件测试动态评价方法, 软件学报 21 (9) (2010) 2135–2147.

[7] 李军锋, 顾滨兵, 李海浩, 软件测试质量评价方法, 计算机与现代化 (09) (2018) 38.

[8] 姚实颖, 肖沙里, 谭霞, 唐跃林, 软件测试自动化中建立可维护脚本的技术, Ph.D. thesis (2003).

[9] V. Garousi, M. Felderer, Developing, verifying, and maintaining high-quality automated test scripts, IEEE Software 33 (3) (2016) 68–75.

[10] G. Gousios, A. Zaidman, M.-A. Storey, A. Van Deursen, Work practices and challenges in pull-based development: the integrator's perspective, in: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, Vol. 1, IEEE, 2015, pp. 358–368.

[11] M. Beller, G. Gousios, A. Zaidman, Oops, my tests broke the build: An explorative analysis of travis ci with github, in: 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR), IEEE, 2017, pp. 356–367.

[12] Y. Zhang, A. Mesbah, Assertions are strongly correlated with test suite effectiveness, in: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, 2015, pp. 214–224.

[13] D. Athanasiou, A. Nugroho, J. Visser, A. Zaidman, Test code quality and its relation to issue handling performance, IEEE Transactions on Software Engineering 40 (11) (2014) 1100–1125.

[14] I. Heitlager, T. Kuipers, J. Visser, A practical model for measuring maintainability, in: 6th international conference on the quality of information and communications technology (QUATIC 2007), IEEE, 2007, pp. 30–39.

[15] T. J. McCabe, A complexity measure, IEEE Transactions on software Engineering (4) (1976) 308–320.

[16] R. Just, D. Jalali, L. Inozemtseva, M. D. Ernst, R. Holmes, G. Fraser, Are mutants a valid substitute for real faults in software testing?, in: Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2014, pp. 654–665.

[17] J. H. Andrews, L. C. Briand, Y. Labiche, Is mutation an appropriate tool for testing experiments?, in: Proceedings of the 27th international conference on Software engineering, 2005, pp. 402–411.

[18] G. Grano, F. Palomba, H. C. Gall, Lightweight assessment of test-case effectiveness using source-code-quality indicators, IEEE Transactions on Software Engineering.

[19] G. Bavota, A. Qusef, R. Oliveto, A. De Lucia, D. Binkley, Are test smells really harmful? an empirical study, Empirical Software Engineering 20 (4) (2015) 1052–1094.

[20] S. R. Chidamber, C. F. Kemerer, A metrics suite for object oriented design, IEEE Transactions on software engineering 20 (6) (1994) 476–493.

[21] S. Scalabrino, M. Linares-Vasquez, D. Poshyvanyk, R. Oliveto, Improving code readability models with textual features, in: 2016 IEEE 24th International Conference on Program Comprehension (ICPC), IEEE, 2016, pp. 1–10.

[22] D. Li, H. Mei, Y. Shen, S. Su, W. Zhang, J. Wang, M. Zu, W. Chen, Echarts: A declarative framework for rapid construction of web-based visualization, Visual Informatics 2 (2) (2018) 136–146.

[23] H. Coles, T. Laurent, C. Henard, M. Papadakis, A. Ventresque, Pit: a practical mutation testing tool for java, in: Proceedings of the 25th International Symposium on Software Testing and Analysis, 2016, pp. 449–452.

[24] M. Delahaye, L. Du Bousquet, Selecting a software engineering tool: lessons learnt from mutation analysis, Software: Practice and Experience 45 (7) (2015) 875–891.

[25] O. Burn, Checkstyle, http://checkstyle. sourceforge. net/.

[26] I. Altaf, J. A. Dar, F. ul Rashid, M. Rafiq, Survey on selenium tool in software testing, in: 2015 International Conference on Green Computing and Internet of Things (ICGCIoT), IEEE, 2015, pp. 1378–1383.

[27] D. Bernstein, Containers and cloud: From lxc to docker to kubernetes, IEEE Cloud Computing 1 (3) (2014) 81–84.

[28] J. Turnbull, The Docker Book: Containerization is the new virtualization, James Turnbull, 2014.

[29] J. C. Carver, N. A. Kraft, Evaluating the testing ability of senior-level computer science students, in: 2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T), IEEE, 2011, pp. 169–178.

[30] D. M. Powers, Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation.

[31] R. C. Martin, Clean code: a handbook of agile software craftsmanship, Pearson Education, 2009.

[32] H. W. Lilliefors, On the kolmogorov-smirnov test for normality with mean and variance unknown, Journal of the American statistical Association 62 (318) (1967) 399–402.

[33] A. Glover, In pursuit of code quality: Don’ t be fooled by the coverage report, IBM Developer Works blog post (2006) 1–2.

简历与科研成果

基本情况 王逸璠，女，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

2014.9～2018.6 南京大学金陵学院 本科

读研期间成果

1. 房春荣，史洋洋，蒋燕，**王逸璠**，“一种功能粒度上基于语义信息的源代码相似度评估方法”，申请号：2019109519971，已受理。
2. 国家自然科学基金项目：软件开发中海量信息的融合反馈机制与支撑平台（61690201），2017-2021
3. 国家自然科学基金项目：基于可理解信息融合的人机协同移动应用测试研究（61802171）, 2019-2021

致　　谢

时光荏苒，两年的硕士研究生生活转眼即将落幕，在这两年的时光里，我学习专业知识，参与项目实践，收获了很多，这其中离不开导师、同学、亲友的支持和帮助，在此对他们致以最诚挚的感谢。

首先感谢研究生导师陈振宇教授和房春荣老师，项目方面从需求阶段到如今落地实现，他们给予了很多建议和帮助，论文方面也悉心指导，为我扩展思路，提了很多建设性的意见。感谢两位老师在我研究生期间对我的指导和帮助，并且给我这个机会可以加入到 iSE 实验室这个大家庭，实验室浓厚的学习氛围和丰富的项目实践促进我成长。

感谢实验室的黄勇老师，在他的指导下完成了项目的技术选型，在技术方面遇到难题的时候，可以耐心地一起讨论为我传授经验。同时也很感谢实验室的同学们在项目完成过程中的热心帮助和全力支持，从他们身上我也学会了很多项目开发技术以及认真坚持的态度。

感谢我的父母，感谢他们在身后的陪伴，每一次次鼓励和一声声问候都激励着我不断前进。

最后感谢论文评审和答辩委员会的各位老师们，感谢您们的辛苦付出。

版权与原创性说明

任何收存和保管本论文的单位和个人，未经作者本人授权，不得将本论文转借他人并复印、抄录、拍照或以任何方式传播，否则，引起有碍作者著作权益的问题，将可能承担法律责任。

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含其他个人或集体已经发表或撰写的作品成果。本文所引用的重要文献，均已在文中以明确方式标明。本声明的法律结果由本人承担。

作者签名： 

日期：2020 年 5 月 26 日

《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：王逸涛

2020 年 5 月 26 日

论文题名	软件测试多维评估系统的设计与实现				
研究生学号	MF1832168	所在院系	软件学院	学位年度	2020
论文级别	☐学术学位硕士 ☑专业学位硕士 ☐学术学位博士 ☐专业学位博士 （请在方框内画钩）				
作者 Email	wang_yifan11@163.com				
导师姓名	陈振宇，房春荣				

论文涉密情况：

☑ 不保密

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