南京大学

研 究 生 毕 业 论 文

（申请工程硕士学位）

论 文 题 目 面向协作式众包测试的质量控制系统的设计与实现

作 者 姓 名 宋少行

学科、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

2019 年 5 月 27 日

学　　号：**MF1732114**

论文答辩日期：**2019 年 5 月 27 日**

指 导 教 师：　　（签字）

The Design and Implementation of Quality Control System for Collaborative Crowdsourcing Testing

By

Shaohang Song

Supervised by

Associate Professor **Jia Liu**

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 2019

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：面向协作式众包测试的质量控制系统的设计与实现

工程硕士（软件工程领域） 专业 2017 级硕士生姓名： 宋少行

指导教师（姓名、职称）： 刘嘉 副教授

摘 要

众包测试是招募线上众包工人完成测试任务的一种测试模式，它通常有竞争式、评选式和协作式三种实现方式。协作式众包测试能招募更多众包工人，收集更多Bug报告，因此得到广泛应用。协作式众包测试允许众包工人互相协作，而且Bug报告完全透明，这可能导致其中出现恶意众包工人。恶意众包工人为了个人利益，可能提交无效Bug报告、抄袭他人报告或随意审核等，这严重影响Bug报告质量，急需质量控制策略保证报告质量。已有的众包质量控制研究无法评估报告质量，不能识别恶意众包工人，所以不能直接运用到协作式众包测试中。为了解决这些问题，本文设计并实现了面向协作式众包测试的质量控制系统。

质量控制系统分为Bug报告有效性检测、Bug报告自动评估、反馈和监控、Bug 报告审核四个模块，从四个方面保证Bug报告质量。第一，系统分析Bug报告特征并提取质量指标，依据质量指标检测报告有效性，为评估报告质量和实时反馈奠定基础。第二，系统自动评估Bug报告的信息增益，并使用灰色关联分析法和熵权法对Bug报告自动评分，实现报告质量的等级评估，为管理者识别报告质量提供参考。第三，系统实时评估众包工人的行为，并记录失信行为，通过失信行为反馈提高众包工人的质量意识；系统提供众包工人失信行为与协作关系的实时监控，帮助管理者提早识别并剔除恶意众包工人。第四，系统提供交互友好的人工审核方式，通过展示报告的基础属性、信息增益和自动评估结果，使管理者可快速公正地审核报告。

在技术实现上，本文使用主流框架保证开发效率和稳定性，采用Angular2作为前端框架，Spring Boot作为服务端框架，MongoDB作为数据库。为了保证系统性能和可扩展性，采用Nginx实现负载均衡，Redis实现系统缓存。

质量控制系统通过了测试与实验评估，已经上线且运行情况良好。系统提高了Bug报告质量和众包工人的质量意识，使管理者能迅速识别Bug报告质量和恶意众包工人，同时使其能方便地审核海量Bug报告，促进了协作式众包模式下的质量控制研究。

关键词： 协作式众包测试，恶意众包工人，Bug报告质量，质量控制

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 Quality Control System for Collaborative Crowdsourcing Testing

SPECIALIZATION: Software Engineering

POSTGRADUATE: Shaohang Song

MENTOR: Associate Professor Jia Liu

Abstract

Crowdsourcing testing is a test model by recruiting online crowd workers to complete test tasks. It is usually implemented in three types: competitive, electoral, and collaborative. Collaborative crowdsourcing testing is widely used，because it recruits more crowd workers and collects more bug reports. Collaborative crowdsourcing testing allows crowd workers to collaborate，and bug reports are completely transparent, which can lead to malicious crowd workers. Malicious crowd workers may submit invalid bug reports, copy others' reports or review reports at will for personal benefit. These seriously affect the quality of bug reports and urgently need quality control strategies to ensure the quality of reports. Existing crowdsourcing quality control studies are unable to evaluate the quality of reports and cannot identify malicious crowd workers, so they cannot be applied to collaborative crowdsourcing testing directly. In order to solve these problems, the thesis designs and implements a quality control system for collaborative crowdsourcing testing.

The system is divided into four modules: Bug Report Validity Detection, Bug Report Automatic Evaluation, Feedback and Monitoring, and Bug Report Review, which guarantee the quality of bug reports from four aspects. First, the system analyzes the characteristics of bug reports, extracts the quality indicators, and detects the validity of reports according to the quality indicators, laying the foundation for the report quality evaluation and real-time feedback. Second, the system evaluates information gains of bug reports automatically, and scores the bug reports automatically with Grey Relational Analysis and Entropy Weight, which implements the quality level evaluation of reports and provides a reference for managers to identify the quality of reports. Third, the system evaluates the behavior of crowd workers in real time, records the malicious

behavior, and improves the quality awareness of crowd workers through malicious behavior feedback. It also provides real-time monitoring of crowd workers' malicious behavior and collaborative relationships, helping managers identify and eliminate malicious crowd workers early. Fourth, the system provides an interactive and friendly manual review method. By displaying the basic attributes, information gains and automatic evaluation results of the reports, the managers can review the reports quickly and fairly.

In terms of technology implementation, the thesis uses the mainstream frameworks to ensure development efficiency and stability, with Angular2 as the front-end framework, Spring Boot as the server framework, and MongoDB as the database. In order to ensure system performance and scalability, Nginx is used for load balancing, and Redis is used for system cache.

The quality control system passed the test and experimental evaluation. And it is running online in good condition. The system improves the quality of bug reports and the quality awareness of crowd workers, enables managers to identify the quality of bug reports and malicious crowd workers quickly, and makes it easy to review massive bug reports for managers. Also, it promotes quality control study in collaborative crowdsourcing mode.

Keywords: Collaborative Crowdsourcing Testing, Malicious Crowd Workers, Quality of Bug Reports, Quality Control

目　录

表　目　录

图　目　录

第一章　引言

1.1　项目背景与意义

1.1.1　众包测试

众包(Crowdsourcing)是Howe Jeff于2006年在美国《连线》杂志上首次提出的一种商业模式 [1]。它是互联网带来的一种分布式问题解决和生产组织模式，指一个公司或者机构把过去由员工执行的工作任务，以自由自愿的形式承包给非特定大众网络的做法 [2]。众包和以往的外包不同，外包是将工作任务承包给高度专业化的人员和资源，而众包强调数量庞大的非专业人员和资源。如果众包模式设计合理，在一定程度上能降低人力、时间和财务成本，为企业的发展提供更多的备选方案，提升资源配置效率 [3]。

众包模式广泛应用于各个领域，特别是软件工程领域。著名的Linux系统即是全球各地的众包编程者合作完成的杰作；人工智能所需的海量数据采集和标注也多采用众包模式完成。在软件测试方面，众包模式能在线招募众包工人，无需准备测试环境，而且测试速度快，其便捷性和低成本受到软件公司和个人的青睐。众包测试主要应用在以下场景：功能测试、兼容性测试、UI测试和测试用例生成等 [4]。

一般来说，众包测试主要参与者包括任务发布者、任务完成者和众包测试平台（简称众测平台），其中任务完成者即为众包工人 [5]。众测平台作为中介为任务发布者和众包工人提供在线平台，任务发布者将测试任务和需求发布到众测平台，众测平台选择众包工人集合并推送任务，众包工人选择感兴趣的任务执行，工人完成测试任务之后提交包含测试结果的Bug报告到众测平台。众测平台对大量质量参差不齐的Bug报告进行审核整理，最终形成可交付的软件缺陷反馈给任务发布者。众包测试能在短时间内召集大量众包工人参与测试任务，快速得到大量Bug报告，这是它显著的优点。但是，由于软件测试任务有专业知识的要求，大量非专业的众包工人可能很难完成任务，或者提交质量较差的报告，这对众测平台的设计和管理带来了很大挑战。

1.1.2　协作式众包测试

众包因工作方式的不同被分为竞争式、评选式和协作式 [6]。竞争式众包模式下参与贡献的众包工人彼此独立，且具有竞争关系，众包平台希望从海量众

包方案中选出最优方案；评选式众包模式以较大的样本量为基础，任务偏简单和同质化，由于单一样本不能解决目标问题，因而需要对整体样本进行统计分析以期获得最终解决方案；协作式众包模式任务异质化较强，众包平台希望通过样本聚合产生质的飞跃，而不仅仅是数量上的汇聚统计。

目前国内外的商业众包测试平台大都采用竞争式的众包模式，众包工人独立完成测试任务并提交Bug报告，众测平台审核海量报告并选择高质量的报告作为最优方案。由于众测平台的众包工人来自世界各地，测试能力水平不一致，因此其提交的报告质量参差不齐，存在测试任务完成度低，测试结果重复率高的问题。异地参与、水平不一致和非企业约束等不利因素给众包测试带来很大挑战 [7] [8] [9]。只有针对测试场景对众包工人提供信息引导，分解测试任务降低能力要求门槛，允许工人在测试时互相协作，同时对工人的操作进行实时反馈，才能使工人更好地完成测试任务，提供更高质量的测试结果，这即为协作式众包测试。

在协作式众包测试中，所有Bug报告对众包工人都是可见的，众包工人以Wiki（多人协作的写作系统）协作编辑的方式完成报告；众包工人不仅可以提交报告，也能对他人报告进行审核，帮助众测平台识别高质量报告。协作式众包测试通过提供特定协作方式，减轻对众包工人专业能力的要求，给众包工人提供多样的任务类型，使众包工人有更强的参与感和成就感；同时通过众包工人协作，使Bug报告聚合产生高质量报告，能以更快的速度交付给任务发布者高可用的软件缺陷列表。

1.1.3　众包的质量问题

众包模式能帮助任务发布者招募大量众包工人，利用众包工人的能力和智慧解决实际问题。伴随着众包的迅猛发展，各个领域有越来越多的众包工人参与到众包工作中来。由于众包模式采取面向不确定大众群体的工作方式，而且承接任务的众包工人具有独立匿名的特点，导致众包模式下工作结果的质量难以控制 [2]。

最初，很多众包工人参与众包工作，主要是出于兴趣爱好和自我认同，经济奖励和其他奖励对这样的工人是次要的吸引力。如今，越来越多的大众用户参与到众包工作中来，其中受经济等奖励所吸引的用户占据很大的比重。这些众包工人为了使自己的利益最大化，并不会认真完成众包任务，而是为了骗取奖励，提交质量低下的工作结果，这些人被称为恶意众包工人。恶意众包工人的行为违背了任务发布者的初衷，同时给众包平台带来管理难题和信誉损失。虽然众包模式使收集大规模的数据更容易，但是如果关心数据的质量，则必须

考虑质量控制问题 [10]。

协作式众包测试是众包模式的特殊应用，众包模式的质量问题在协作式众包测试中仍存在。受经济等奖励的吸引，恶意众包工人可能提交无效Bug 报告；由于协作式众包测试中报告等信息的全透明性，恶意众包工人可能抄袭他人报告；基于协作式众包测试提供的审核功能，恶意众包工人可能恶意审核报告，干扰众测平台对高质量报告的选择。因此，协作式众包测试必须采取质量控制措施识别无效Bug报告，识别并剔除恶意众包工人，采取公平的评估方法选择高质量报告，合理评估众包工人的贡献并按劳奖励。

本文针对协作式众包测试面临的质量问题，设计并实现质量控制系统，采取质量控制措施保证Bug报告的质量。系统既能警示众包工人以认真的态度完成测试任务，又能快速有效识别报告质量，减轻众测平台的审核负担，最终帮助管理者给任务发布者反馈高质量的软件缺陷，促进众包测试的可持续发展。

1.2　国内外研究现状

1.2.1　众包测试研究现状

在学术研究中，Mao等人全面概括了众包技术在软件工程中的应用，涵盖大量众包软件工程的应用模式和相关平台，并给出众包软件工程的定义：“众包软件工程是指由大量潜在的、未定义的在线工人以公开召集的形式，承担外部软件工程任务的行为” [11]。基于众包软件工程的定义，章晓芳等人定义众包软件测试为：“支持软件测试的所有众包活动，包括所有众包方法、技术、工具和平台” [12]。

在众包测试技术研究方面，章晓芳等人全面总结了国内外众包测试技术的研究重点，包括面向众包的测试优化和众包测试的自身优化 [12]。面向众包的测试优化主要针对特定类型的测试任务进行优化，如QoE（Quality of Experience，体验质量）测试、可用性测试和GUI 测试等。QoE测试为用户体验质量测试，传统的QoE测试以人工测试为主，成本较高且耗时较长，而众包QoE测试可快速收集不同地域和使用环境下的测试数据，降低成本并提高效率 [13]。众包可用性测试广泛应用于移动应用测试领域，早期研究主要集中在传统测试和众包测试的效果对比上 [14] [15]。Liu的研究表明，众包测试比传统方式更容易获取来自全球不同背景的测试数据，而且众包测试可以平行进行，能够提高任务的完成速度，但是众包测试的反馈数量和质量略低于传统测试方式 [14]。GUI 测试难点在于自动化的生成GUI测试用例，人工GUI测试费时费力，众包GUI测试被认为是持续开展GUI测试的有效手段 [16]。

除了将众包技术初步应用于特定类型的测试任务之外，研究者还深入研究众包测试的流程控制和运行机制问题，例如如何召集和管理众包工人、如何分解和设计测试任务、如何整合处理测试报告等。召集大量优质众包工人并进行有效管理是开展测试任务的前提，同时众包工人规模和参与时间对测试效果也有显著影响，文献 [17] 深入研究了具体影响方式和结果。面对复杂的众包测试任务时，任务分解和设计是必要的前期工作，已有研究人员将协作式众包测试问题看作是对众包工人任务分配的NP完全问题，并将其转化为整数线性规划问题来解决 [18]。测试报告整理与处理是众包测试自身优化的重要方面，而众包测试报告排序是有效提高报告处理效率的方法之一，一方面可以运用风险策略和多样性策略动态选择测试报告进行检查 [19]，另一方面可以综合文本和图片信息对测试报告进行优先级排序 [20]。

在工业应用上，随着众包模式的飞速发展，国内外众包测试平台层出不穷。国内的热门商业众包测试平台有：百度众测[1]、阿里云测众测[2] 和Testin云测[3]等；国外主流的云测试平台有TestDroid[4]和Sauce Labs[5]等。虽然上述众包测试平台前景优良，但在平台设计上多属于竞争型众包模式，靠网络社会大众，使海量众包工人互相竞争完成测试任务，众包工人之间仍然保持相对独立状态，并未实现人员协作，限制了群体智能的发挥。Tapscott对协作式众包的典型案例“维基百科”进行了详细分析，得出维基百科的迅速壮大得益于协作式众包的显著优势 [21]。Wu等人提出协作式众包受到越来越多的关注，但想实现从合作到协作，实现1 + 1 > 2的目标，任重而道远 [22]。

虽然众包测试已有广泛研究和相关平台实现，但是关于协作式众包测试的研究和平台较少。基于协作式众包测试的巨大优势和发展潜力，将来可能会出现越来越多的协作式众包测试平台，它们也将不可避免的受到质量控制问题的影响。因此，面向协作式众包测试的质量控制问题亟待解决。

1.2.2　众包质量控制研究现状

目前众包质量控制的研究工作主要集中在三个方面。第一是结果质量评估方法研究，这类工作通过各种方法评估众包结果的质量，进而识别恶意众包工人，剔除质量较差的结果和恶意众包工人。第二是众包工人的组织模型，这类工作通过建立合适的工人组织管理方式来控制众包结果的质量。第三是众包任

[1]http://test.baidu.com/
[2]https://mqc.aliyun.com/crowdtest/
[3]https://www.ztestin.com/
[4]https://cloud.testdroid.com/
[5]https://saucelabs.com/

务的设计，这类工作通过设计模式良好的众包任务，以期持续获得高质量的众包结果。

在结果质量评估方面，最常用的方法是使用黄金标准数据评估众包结果的质量 [23]。黄金标准数据是指拥有标准答案的一类数据，使用标准答案识别众包结果的质量，根据结果质量进而识别恶意众包工人。在相关性判断类型的众包工作中，还可以使用Kappa一致性检测的方法判断众包工人两次众包结果的一致性，进而评估结果质量与众包工人能力水平 [24]。协作式众包测试任务复杂，不同于相关性判断等客观简单的任务，同时也没有标准答案，传统上只能人工评估报告质量，效率低下且成本高。如何快速准确识别Bug报告质量是面向协作式众包测试的质量控制面临的一个难题。

在众包工人组织模型方面，Kochhar等人使用传统类似于公司人员管理方式的工人组织模型，签约工人可根据完成任务的表现被提升为管理者，进而承担更加困难收益也更高的任务 [25]。评估众包工人表现是该组织模型的核心难题，是其组织模型运转和保证结果质量的前提。协作式众包测试允许众包工人互相协作，是一种不同于传统竞争式众包的工人组织模型，评估众包工人表现也是质量控制的核心内容。因此如何评估众包工人的表现，进而根据评估识别恶意或能力较强的众包工人，是质量控制策略要重点解决的问题。

在众包任务设计方面，Sorokin等人发起了一项图片标注的众包应用实验，他发现以极低薪酬发布的任务会导致任务的接受速度很慢，很少有众包工人会对此任务感兴趣；以较高奖励额度发布的任务会更容易吸引效率低下的恶意众包工人 [26]。协作式众包测试为了使任务快速完成，一般都会有较高的任务奖励，因此很容易吸引恶意众包工人。研究人员还发现：复杂度较高的、需要创造力的任务，对那些只想简单快速完成任务的工人缺乏吸引力 [27]。为了降低众包测试的能力门槛，协作式众包测试提供协作编辑和审核等复杂度较低的任务，但是却容易吸引只想快速完成任务获取奖励的恶意众包工人。

综上所述，面向协作式众包测试的质量控制问题较为严峻，但是目前关于协作式众包测试的质量控制策略研究较少，传统的结果评估方法不能使用到Bug报告评估中，评估众包工人表现并识别恶意众包工人也较为困难，因此本文需要设计新的质量控制策略运用到协作式众包测试中。

1.3　本文的主要工作

针对以上的研究现状和众包质量控制存在的问题，本文从保证Bug报告质量的角度出发，开发了一个面向协作式众包测试的质量控制系统。质量控制系统是协作式众包测试系统（简称众测系统）的子系统，其主要功能为采取适当

质量控制措施，提高众包工人的质量意识，评估并保证Bug报告的质量，提早识别并剔除恶意众包工人。系统主要由四个模块实现质量控制策略。

众包工人在众测系统上以Bug报告的方式提交测试结果，报告可能有效也可能无效，为了帮助管理者识别报告有效性，质量控制系统需要分析报告特征，提取影响报告质量的指标，并根据指标评估报告有效性。此为质量控制系统的Bug报告有效性检测模块。

众包工人在众测系统上可以单独提交Bug报告，也可以对他人的报告进行复制修改（众测系统中称为Fork操作），提交基于他人报告的子报告，子报告都存在父报告。对于两种类型的报告，系统需要自动评估每个报告的质量等级，即对报告进行评分。单独提交的报告根据质量指标的好坏客观评分；Fork 产生的子报告由于借鉴了他人的劳动成果，系统需要评估子报告相对于父报告的信息增益，综合信息增益和父报告分数来评分。此为质量控制系统的Bug报告自动评估模块。

众包工人在众测系统上可以提交Bug报告，也可以对他人的报告进行审核。为了识别恶意众包工人，系统需要评估众包工人的每个行为，识别其做出的失信行为，并实时反馈给众包工人，提高工人的质量意识。同时系统需要提供监控模块，展示众包工人的失信行为和协作关系，帮助管理者提早识别恶意工人，并终止其参与测试任务的资格。此为质量控制系统的反馈和监控模块。

每个测试任务都将收集海量Bug报告，自动评估并不能确定报告描述缺陷的真实性，所以需要管理者人工审核。系统提供展示某个测试任务所有报告的功能，供管理者查看和审核。管理者参考自动评估结果审核报告，确定每个报告的最终质量等级。此为质量控制系统的Bug报告审核模块。

为了保证系统可用性，本文对系统进行了功能和性能测试。同时，为了保证系统所提评估方法的可靠性，本文对方法进行了实验评估。

1.4　本文的组织结构

本文的主要工作为设计并实现面向协作式众包测试的质量控制系统，因此本文的组织结构如下。

第一章为引言。介绍了本文背景众包测试、协作式众包测试和众包的质量问题，简述了国内外在众包测试和众包质量控制方面的研究现状，介绍了本文的主要工作。

第二章为相关理论与技术。介绍了本文使用的Angular2、Spring Boot、Word2Vec 、灰色关联分析法和熵权法等理论与技术。

第三章为质量控制系统的需求分析与设计。首先对系统做了整体概述，然后以用例图的方式介绍各个模块的具体需求，最后介绍了Bug 报告有效性检测、Bug报告自动评估、反馈和监控、Bug报告审核等四个模块的详细设计。

第四章为质量控制系统的实现。在第三章的基础上，以顺序图、代码和系统界面等方式详细介绍各模块的实现细节。

第五章为质量控制系统的测试与实验评估。首先介绍了系统的测试环境，然后对系统各模块的重要接口进行功能和性能测试，最后对系统所提方法进行实验评估。

第六章为总结与展望。对论文所做工作进行了总结，分析了工作中存在的问题，展望了质量控制系统未来的发展方向。

第二章　相关理论与技术

2.1　Angular2

2010年10月，Google首次发布了Web开源框架AngularJS，即为Angular1。Angular1首次提出的“双向数据绑定”概念引起开发者强烈关注，并受到众多前端开发者的喜爱。Angular2 于2016 年9月发布，是Angular1的升级版本。它基于ES6开发，性能上得到显著提高，能很好地支持Web开发组件。

Angular2有很多的特性与优点，本文对此简单介绍。第一为出色的跨平台能力。Angular2充分利用现代Web平台的各种能力，提供高性能、离线使用和免安装的App式体验，借助来自Ionic、NativeScript和React Native中的技术和思想，构建原生应用。第二是高速度和性能。Angular2将模板转换成代码，针对现代JavaScript虚拟机进行高度优化，获取框架提供的高生产率，同时保留所有手写代码的优点。在服务端渲染应用上，保持像只有html和css页面那样的瞬间展示表现，支持Node.js、.NET、PHP和其他服务器。同时借助新的组件路由器，Angular2可以实现快速加载，自动代码拆分机制可让用户仅仅加载用于渲染所请求页面的代码。第三是高生产率。Angular2通过简单而强大的模板语法，可快速创建UI视图，使用Angular CLI命令行工具，可快速构建、添加组件、测试和部署。第四是高可测试性。Angular2通过将应用逻辑与操作DOM解耦，重视应用程序的测试，显著提高代码的可测试性。第五是降低开发难度。Angular2将应用程序的客户端和服务端解耦，允许客户端和服务端并行开发，并且提高了双方的复用性。第六是模块化前端开发。Angular2遵循软件工程的MVC 模式，鼓励视图、数据和逻辑组件之间的松耦合，通过依赖注入为客户端Web应用带来了传统服务端的服务，例如独立于视图的控制。

基于Angular2的诸多特性与优点，质量控制系统前端使用Angular2构建，降低客户端与服务端之间的耦合，提高系统的开发效率和可测试性，使前端页面的加载和渲染速度更快，给用户提供更好的交互体验。

2.2　Spring Boot

Spring Boot是由Pivotal团队提供的全新框架，其简化了Spring应用的初始搭建以及开发过程，使用特定的方式进行配置，从而使开发人员不需要样板化的配置 [28]。字面上看，Boot是引导的意思，可理解为Spring Boot能帮助开发

者快速搭建Spring框架。Spring Boot自带Web容器，继承原有Spring框架的优点，简化Spring的使用过程，使编码、配置、部署和监控变得更简单，带来了脚本语言开发的效率，有诸多特性和优点。

Spring Boot的主要特性有以下几点。第一为遵循“习惯优于配置”的原则，使用Spring Boot只需要很少的配置，大多数情况下直接使用默认配置即可。第二为可无配置集成主流开发框架，拥有强大的社区支持和第三方插件。第三为简化框架配置过程，只需自动配置和Java Config，就可以完全替代XML配置文件。第四为内嵌Servlet容器，降低了对开发环境的要求，可直接使用命令执行项目，也可使用jar包执行。第五为优化包管理，通过提供starter POM使包管理变得非常方便，很大程度上减少jar hell和dependency hell。第六为实时监控功能，框架可实时监控运行中应用程序的状态，运维人员可实时查看应用指标并进行风险预警。

基于Spring Boot的诸多特性和好处，质量控制系统服务端采用它搭建，可获取更快的开发、测试和集成效率，保证服务端的高性能，同时非常方便的集成使用第三方插件。

2.3 Word2Vec

随着计算机应用领域的不断扩展，自然语言处理受到了人们的高度重视。为了使计算机能够处理自然语言，首先需要对自然语言进行建模，Word2Vec即为其中一种模型。Word2Vec 模型在2013年被提出，谷歌最早实现了该模型的开源工具 [29] [30]。Word2Vec可以根据给定的语料库，通过优化后的训练模型快速有效地将一个词语表达成向量形式，为自然语言处理领域的应用研究提供了新的方向。

Word2Vec本质上是一种单词聚类的方法，是实现单词语义推测和句子情感分析等功能的一种手段。Word2Vec主要包括两个神经网络语言模型。首先是Skip-gram模型，指使用一个词语作为输入，预测它周围的上下文 [31]。其次是CBOW模型，指使用一个词语的上下文作为输入，预测这个词语本身 [32]。通过使用这两个语言模型，Word2Vec可将One-hot Representation表示的词语降维为使用Distributed Representation表示，使用上下文信息确定词语语义 [33]。Distributed Representation 在形式上将词语表示为固定长度的稠密词向量，在功能上使相关或者相似的词语距离更近。Word2Vec在训练词向量的过程中使用两种训练技巧，第一是Hierarchical Softmax，它将N分类问题变成$log(N)$分类问题；第二是Negative Sampling，它将预测总体类别变为预测总体类别的一个子集。通过这两个训练技巧，极大降低了神经网络学习过程的复杂度。

Word2Vec可以将词语快速表达成向量形式，由于词语是NLP（Nature Language Processing，自然语言处理）里最细粒度的表达，所以Word2Vec的应用很广泛，既可执行词语层面的任务，也可作为很多模型的输入，执行较高层句子和文档层面的任务，例如计算相似度、句子情感分析和文档主题判别等领域。Word2Vec有很多开源实现，例如著名的gensim[1]，开源的java实现HanLP[2]等。

基于Word2Vec优秀的词嵌入与完成各种层面中文NLP任务的能力，本系统使用它实现句子和文档相似度的计算，使用HanLP训练并加载Word2Vec模型。

2.4　灰色关联分析法

灰色关联分析法是对一个系统发展变化态势定量描述和比较的方法，其基本思想是通过确定参考数据列和若干个比较数据列的几何形状相似程度来判断其联系是否紧密，它反映了曲线间的关联程度 [34]。在综合评价方面，灰色关联分析法通过计算关联度表示被评价对象与标准对象的接近程度。本文使用它计算Bug报告与标准报告之间的接近程度，从而对报告自动评分。以下详细介绍使用灰色关联分析法综合评价的过程。

(1) 收集评价数据

现有被评价对象集合$M = \left(m_1 \ ,\cdots \ ,m_m \right)$，对评价对象提取的评价指标集合$D = \left(d_1 \ ,\cdots \ ,d_n \right)$，被评价对象$m_i$指标$d_j$的值为$x_{ij}$，则形成的原始数据矩阵为$X$，如下所示：

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{pmatrix}$$

(2) 确定标准对象

标准对象是一个理想的比较标准，可以使用各指标的最优值（或最劣值）构成标准对象，也可根据评价目的选择其他参考值，记作m_0，如下所示：

$$m_0 = \left(x_{01} \ ,\cdots \ ,x_{0n} \right)$$

(3) 无量纲处理

对于越大越优型指标，无量纲处理过程如式2.1所示。对于越小越优型指标，无量纲处理过程如式2.2所示。

$$y_{ij} = \frac{x_{ij} - \min\left(d_j\right)}{\max(d_j) - \min(d_j)} \tag{2.1}$$

[1]https://radimrehurek.com/gensim/

[2]http://hanlp.com/

$$y_{ij} = \frac{\max(d_j) - x_{ij}}{\max(d_j) - \min(d_j)} \tag{2.2}$$

其中y_{ij}为x_{ij}无量纲处理之后的数据，$\min(d_j)$和$\max(d_j)$ 分别是d_j指标的最大值和最小值。矩阵X添加标准对象m_0，一并无量纲处理之后的矩阵Y 为:

$$Y = \begin{pmatrix} y_{01} & \cdots & y_{0n} \\ y_{11} & \cdots & y_{1n} \\ \vdots & \ddots & \vdots \\ y_{m1} & \cdots & y_{mn} \end{pmatrix}$$

(4) 计算关联系数

计算在指标d_j下待评价对象m_i与标准对象m_0的关联系数c_{ij}，如式2.3 所示。式中ρ为分辨系数，在0到1内取值。ρ越小，关联系数间差异越大，区分能力越强，通常ρ取0.5。

$$c_{ij} = \frac{\min\limits_i \min\limits_j |y_{0j} - y_{ij}| + \rho \times \max\limits_i \max\limits_j |y_{0j} - y_{ij}|}{|y_{0j} - y_{ij}| + \rho \times \max\limits_i \max\limits_j |y_{0j} - y_{ij}|} (1 \le i \le m, 1 \le j \le n) \tag{2.3}$$

当使用各指标的最优值（或最劣值）构成标准对象计算关联系数时，可使用式2.4所示方法计算关联系数，不仅使计算更简便，而且改进后的方法可以省略第三步无量纲化，避免无量纲化对指标的某些负面影响。使用式2.4计算时，如果标准对象m_0为最优值数据列，关联系数值越大越好；如果m_0为最劣值数据列，值越小越好。

$$c_{ij} = \frac{\min\limits_i |y_{0j} - y_{ij}| + \rho \times \max\limits_i |y_{0j} - y_{ij}|}{|y_{0j} - y_{ij}| + \rho \times \max\limits_i |y_{0j} - y_{ij}|} (1 \le i \le m, 1 \le j \le n) \tag{2.4}$$

(5) 计算总关联度

如果各指标在综合评价中所起的作用不同，权重为$W = (W_1, \cdots, W_n)$，可对关联系数求加权平均值作为待评价对象与标准对象的总关联度。待评价对象m_i的总关联度R_i由式2.5计算，表示m_i与标准对象的接近程度。

$$R_i = \sum_{j=1}^{n} (W_j \times c_{ij}) / n \tag{2.5}$$

2.5　熵权法

C.E.Shannon在1948年将信息熵的概念引入到信息论中，作为信息的一个度量。根据信息论的基本原理，信息是系统有序程度的一个度量，而熵是系统无序程度的一个度量，信息的增加意味着熵的减少，信息与熵成反比 [35]。

熵权法是利用信息熵原理的一种客观赋权方法，可根据数据分布计算评价指标的权重 [36]。本文使用熵权法计算Bug报告质量指标的权重，熵权法的具体应用步骤描述如下。

首先收集评价数据并进行无量纲处理，具体过程如灰色关联分析法所述，此处不再赘述。无量纲处理之后得到的矩阵为V，如下所示。

$$V = \begin{pmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & \ddots & \vdots \\ v_{m1} & \cdots & v_{mn} \end{pmatrix}$$

第二计算特征比重。设在指标d_j下，评价对象m_i的特征比重为p_{ij}，则p_{ij}计算方法如式2.6所示。

$$p_{ij} = v_{ij} / \sum_{i=1}^{m} v_{ij} \tag{2.6}$$

第三计算熵值。设指标的熵值为E_j，其计算方法如式2.7所示。

$$E_j = -1/\ln(m) \sum_{i=1}^{m} (p_{ij} \times \ln(p_{ij})) \tag{2.7}$$

第四计算差异系数。设指标d_j的差异系数为F_j，其计算方法如式2.8所示。对于各项指标，指标差异系数越大，表明评价对象中该指标差异越大，则该指标反映的信息量也越大，应该给予更大的权重。

$$F_j = 1 - E_j \tag{2.8}$$

最后确定指标熵权。经过以上过程，指标d_j的权重W_j 计算方法如式2.9 所示。对每个指标重复上述操作，即可得到所有指标的权重。

$$W_j = F_j / \sum_{j=1}^{m} F_j \tag{2.9}$$

2.6　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。首先介绍了本文使用的前端框架Angular2。其次介绍了本系统使用的服务端框架Spring Boot。然后介绍了提取文本特征使用的Word2Vec技术，用于报告文本分析和相似度计算。接着介绍了自动评估报告质量的灰色关联分析法，用于对报告自动评分。最后介绍了计算指标权重的熵权法，用于确定报告质量指标的权重。

第三章　质量控制系统的需求分析与设计

3.1　系统整体概述

质量控制系统依托的协作式众包测试系统简称众测系统，它为国内的M测试平台提供众包测试的服务。众测系统主要包括任务发布者、众包工人和管理者等用户。任务发布者可在众测系统上发布移动应用测试、Web测试和应用程序测试等测试任务。众测系统招募众包工人互相协作完成手工测试，并将测试结果以测试报告的形式提交到系统，每个测试报告包括多个Bug 报告，众包工人可以审核他人Bug报告。管理者审核整理Bug报告形成最终软件缺陷列表，交付给任务发布者。众测系统用户交互图如图 3.1 所示。

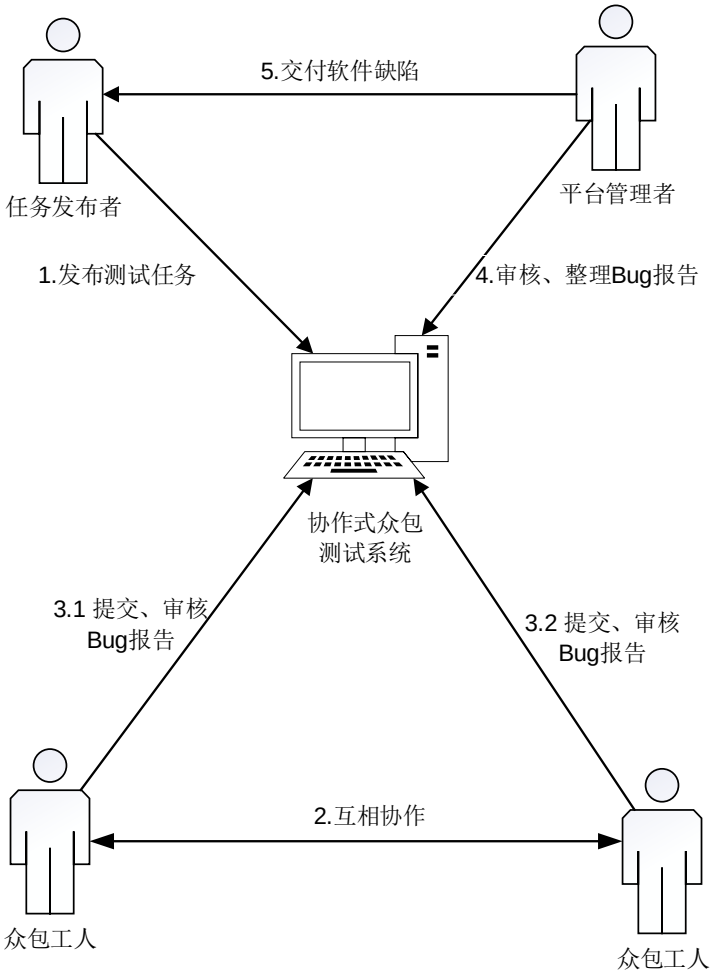

图 3.1: 众测系统用户交互图

协作式众包测试系统包括推荐系统和质量控制系统。推荐系统提供众包工人填写提交Bug报告、互相协作和审核报告等服务，质量控制系统在众包工人测试全过程采取质量控制措施，保证Bug报告的质量。每个众包工人在一次测试任务中提交一份测试报告，一份测试报告中包括多份Bug报告。众包工人与众测系统的交互图如图 3.2 所示。众包工人与系统有三种交互方式，第一，众包工人可不采取协作方式，独立创建新Bug报告并提交。第二，众包工人能查看所有已提交的Bug 报告集和推荐的相似Bug报告列表，并选择其认为不完善的报告进行复制修改（Fork）生成子报告，并提交子报告。第三，众包工人可以审核其他Bug报告，当认为报告有效时可点赞，否则点踩。

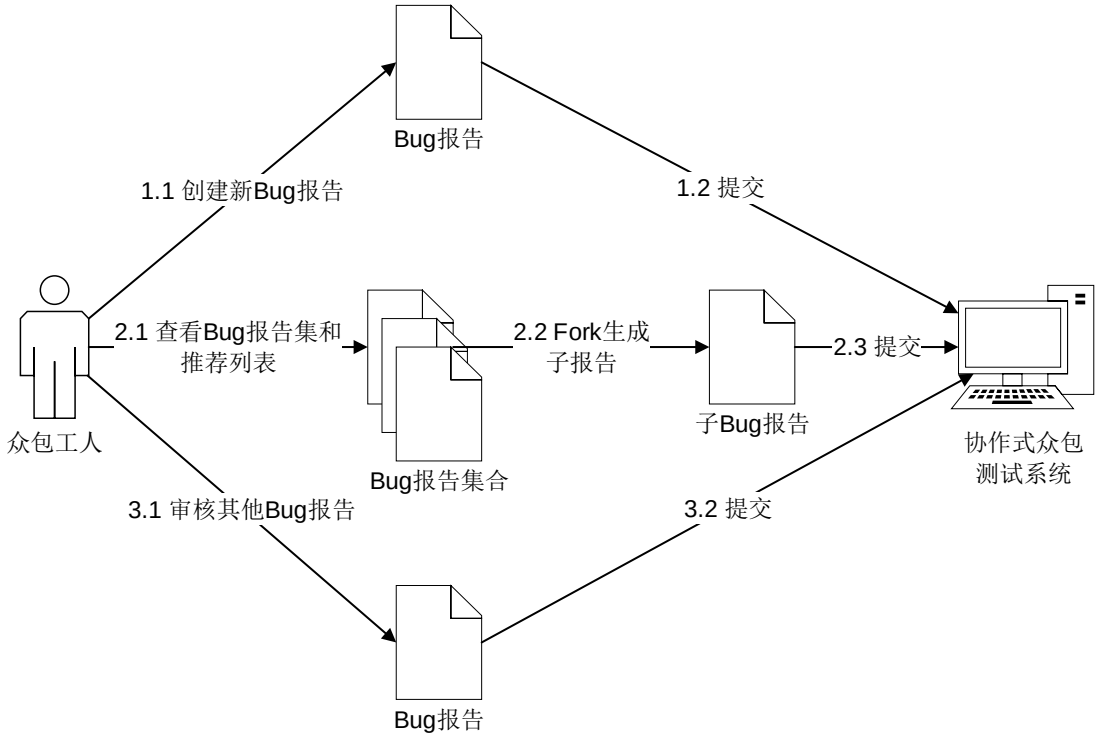

图 3.2: 众包工人与众测系统交互图

针对众包工人与众测系统的交互方式，质量控制系统从Bug报告有效性检测、Bug报告自动评估、反馈和监控、Bug报告审核四大模块控制Bug 报告的质量，各模块主要功能描述如下。

1. Bug报告有效性检测：提取影响Bug报告质量的指标，确定质量指标的正负向影响以及有效范围，检测报告有效性，为后续模块奠定基础。

2. Bug报告自动评估：自动评估报告信息增益，使管理者更易辨别对报告的

修改补充内容；自动评估所有Bug报告质量（本文量化为分数），为管理者审核报告提供参考，为评估众包工人贡献提供依据。

3. 反馈和监控：评估众包工人的每个操作，并给予实时反馈；展示众包工人的失信行为和协作行为，供平台管理者识别和剔除恶意众包工人。

4. Bug报告审核：提供Bug报告人工审核服务，通过展示报告详细信息供众测系统管理者审核报告。

3.2　系统需求分析

3.2.1　功能性需求

(1) Bug报告有效性检测模块

Bug报告有效性检测模块检测报告有效性，为评估并记录众包工人失信行为奠定基础，帮助系统通过实时反馈提高众包工人质量意识，帮助管理者识别恶意众包工人，从而提高报告数据集的质量。为了检测Bug报告有效性，系统首先需要分析Bug 报告特征，确定影响报告质量的指标。其次，系统需要确定质量指标的具体影响方式和指标合理范围。最终，系统需要根据质量指标及其影响方式检测Bug报告的有效性。在该模块中，管理者可查看Bug报告质量指标，管理质量指标合理范围，查看报告Bug报告有效性等。图 3.3为Bug报告有效性检测模块用例图。

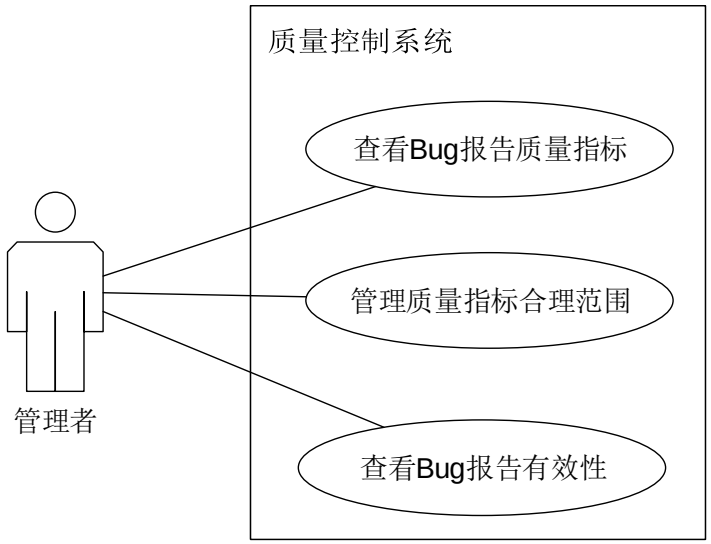

图 3.3: Bug报告有效性检测模块用例图

(2) Bug报告自动评估模块

Bug报告自动评估模块为管理者提供查看报告信息增益和查看报告自动评分的功能。该模块以公正的方式评估报告信息增益以及报告质量，帮助管理者识别众包工人的失信行为，同时为管理者审核报告提供参考。该模块需要提供自动计算两类Bug报告分数的功能，两类报告的描述如下。

1. 单一状报告：该类报告由众包工人独立创建，不是通过Fork创建，因而没有父报告，呈单一状，树状报告的根节点报告也属于单一状。

2. 树状报告：该类报告由众包工人通过Fork其他报告生成，有父报告，呈树状，树状报告的根节点报告由于没有父报告，因此属于单一状。

单一状报告由众包工人独立完成，属于个人贡献，因此其分数完全由报告质量指标确定；树状报告由于存在父报告，其贡献非单一众包工人独有，因此其分数由父报告分数和子报告信息增益共同决定。对于Bug报告集，系统首先提取单一状报告集，根据报告质量采用适当算法自动计算单一状报告分数。对于树状报告集，系统需要评估每个子报告相对于父报告的信息增益，然后采用适当算法依据树状报告根节点的分数和子节点的信息增益综合评分。图 3.4所示为Bug报告自动评估模块用例图。

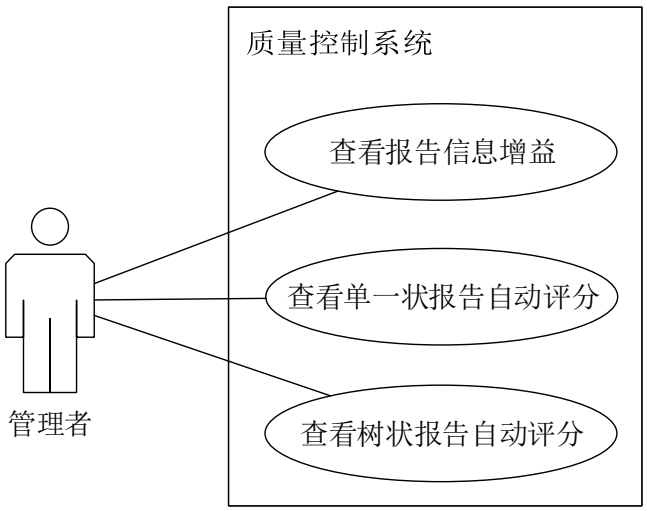

图 3.4: Bug报告自动评估模块用例图

(3) 反馈和监控模块

反馈和监控模块包括实时反馈功能和监控功能。众包工人在众测系统中主要有提交Bug 报告、Fork后提交Bug报告、点赞和点踩等操作，质量控制系统需

要对众包工人的每个操作给出实时反馈，对合理操作反馈鼓励，对不合理操作反馈警告。系统将不合理操作为失信行为，并将实时记录失信行为。表 3.1 所示为系统定义的众包工人操作合理性列表。

该模块为众包工人提供查看操作反馈的功能，包括报告有效性反馈、Fork 提交反馈、报告重复性反馈、点赞反馈和点踩反馈。当众包工人操作合理时，系统给出鼓励性反馈，促使其持续正向操作。当众包工人操作不合理时，系统给出警告性反馈，提高众包工人的质量意识，防止其持续负向操作。

表 3.1: 众包工人操作合理性列表

合理操作	不合理操作
提交有效Bug报告	提交无效Bug报告
提交的Bug报告与已有报告不重复	提交的Bug报告与已有报告重复
Fork父报告生成的子报告添加不同的信息	Fork父报告生成的子报告未添加不同的信息
点赞有效Bug报告	点赞无效Bug报告
点踩无效Bug报告	点踩有效Bug报告

该模块为管理者提供查看失信众包工人以及失信行为详情的功能，供管理者审核失信行为，提早识别并剔除恶意众包工人；同时为管理者提供查看众包工人协作关系的功能，包括点赞关系、点踩关系和Fork关系。协作关系从另一个角度展示众包工人的行为，能够帮助管理者识别众包工人的能力，识别恶意众包工人或众包团伙。图 3.5 所示为反馈和监控模块用例图。

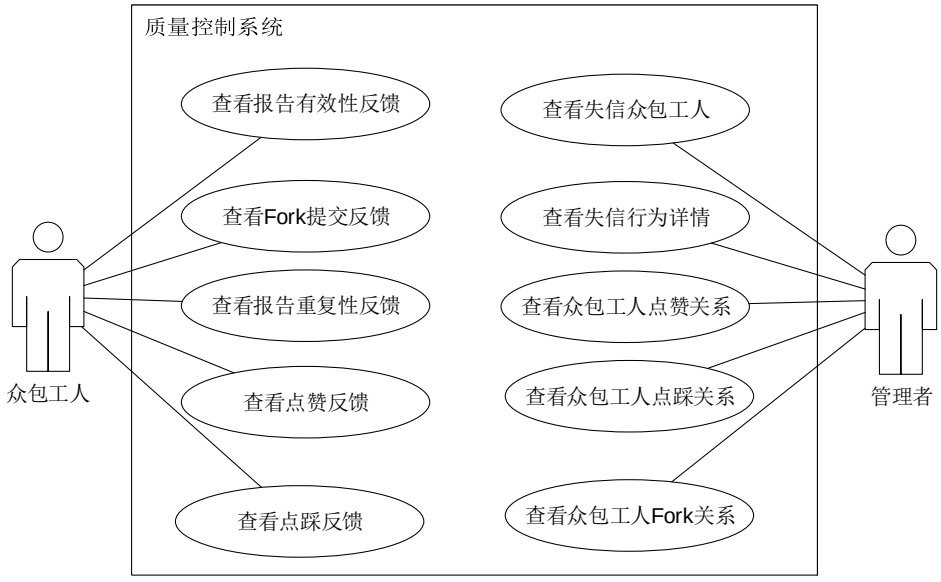

图 3.5: 反馈和监控模块用例图

(4) Bug报告审核模块

Bug报告审核模块主要供管理者使用，管理者可在该模块查看并审核所有Bug报告，该模块分为单一状报告审核和报告树审核两部分。众包工人Fork生成子报告的操作将使两个报告产生父子关系，一个根节点报告与其所有子报告组成一个树状集合，本文称之为报告树。图 3.6为Bug报告审核模块用例图。

在单一状报告审核中，管理者可查看按照创建时间排序的所有单一状报告列表，也可筛选特定页面的报告列表。管理者可查看列表中任意报告的详情，包括基本属性、有效性、自动得分和截图标注等信息，并可根据详细信息以报告评分的方式审核报告质量。

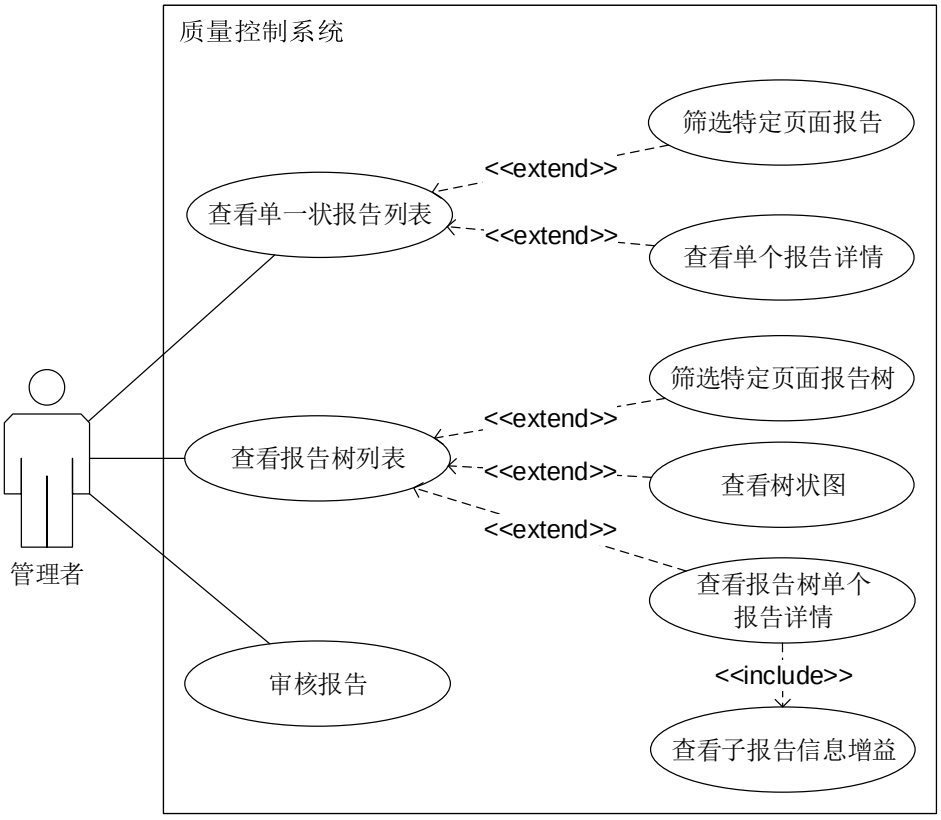

图 3.6: Bug报告审核模块用例图

在报告树审核中，管理者可查看按照创建时间或节点数目排序的所有报告树列表，可筛选特定页面的报告树列表。模块提供可视化树状图展示每棵报告树，管理者可查看树状图以及报告树上单个报告的详情。同时，管理者可查看报告的信息增益，快速识别子报告补充的信息。同样，管理者以报告评分的方式审核单个报告质量。

3.2.2　非功能性需求

(1) 性能

系统前端各个界面的响应时间99%不高于1秒，降低用户的等待时间，提高系统的交互体验。前端异步调用服务端的接口时，响应时间99%不高于1 秒。服务端接口可承受200个用户并发使用。

(2) 可扩展性

可扩展性是指系统本身适应业务环境或者运行环境变化的能力，系统应有较高可扩展性。服务端的有效性检测、自动评估和信息增益评估等算法做好接口抽象，方便以后对算法的增加、升级和替换。在添加新系统模块时，无需修改已有业务模块。在扩展已有业务模块功能时，无需重构已有模块，能在1周内衍生新功能。为了应对业务增长带来的性能压力，系统需要保持服务端水平扩展的能力，可通过增加服务器搭架集群的方式保证服务稳定。

(3) 可用性

可用性是指系统服务不中断运行时间占实际运行时间的比例，系统应有较高可用性。一周内崩溃不能超过1次，系统崩溃时可在10分钟内恢复正常。

(4) 可靠性

可靠性是指系统在一般或者特殊环境下完成规定功能的能力。系统的服务端和数据库都要有一定的容灾能力。单个Web服务器宕机时，系统能通过负载均衡等方法，保证其他Web服务器正常提供服务。对数据库进行备份，当主库出现故障时，系统可及时切换到备份库，保证数据库访问正常。

(5) 易用性

易用性是以用户为中心，尽量使软件系统符合用户的习惯和需求。为了使系统有较高易用性，系统前端界面简洁友好、自解释性强，人机交互体验应该满足大部分用户的要求。用户能够快速找到所需功能，且通过系统指引完成操作。前端提供及时、通俗的用户操作反馈，使用户在接触系统10 分钟之内就可熟练使用。

(6) 可维护性

可维护性衡量一个系统可修复性和可改进性的难易程度，可修复性指系统发生故障后能够排查故障并修复系统的可能性。可改进性指可对系统现有功能进行改进，并增加新功能的可能性。为了达到高可维护性，系统服务端需要有完备的日志记录，代码有详细注释，增加可维护性。服务端的业务类设计为具有单一职责的接口，使服务端的业务解耦，增加可改进性。

3.3　系统总体设计

3.3.1　架构设计

协作式众包测试系统是M平台的子系统，为M平台提供协作式众包测试的服务。M 平台为国内著名的综合测试平台，提供多种测试服务，拥有大量测试人员。众测系统又分为推荐系统和质量控制系统，推荐系统负责众包工人协作、引导和提交Bug报告，质量控制系统负责测试过程的质量控制。

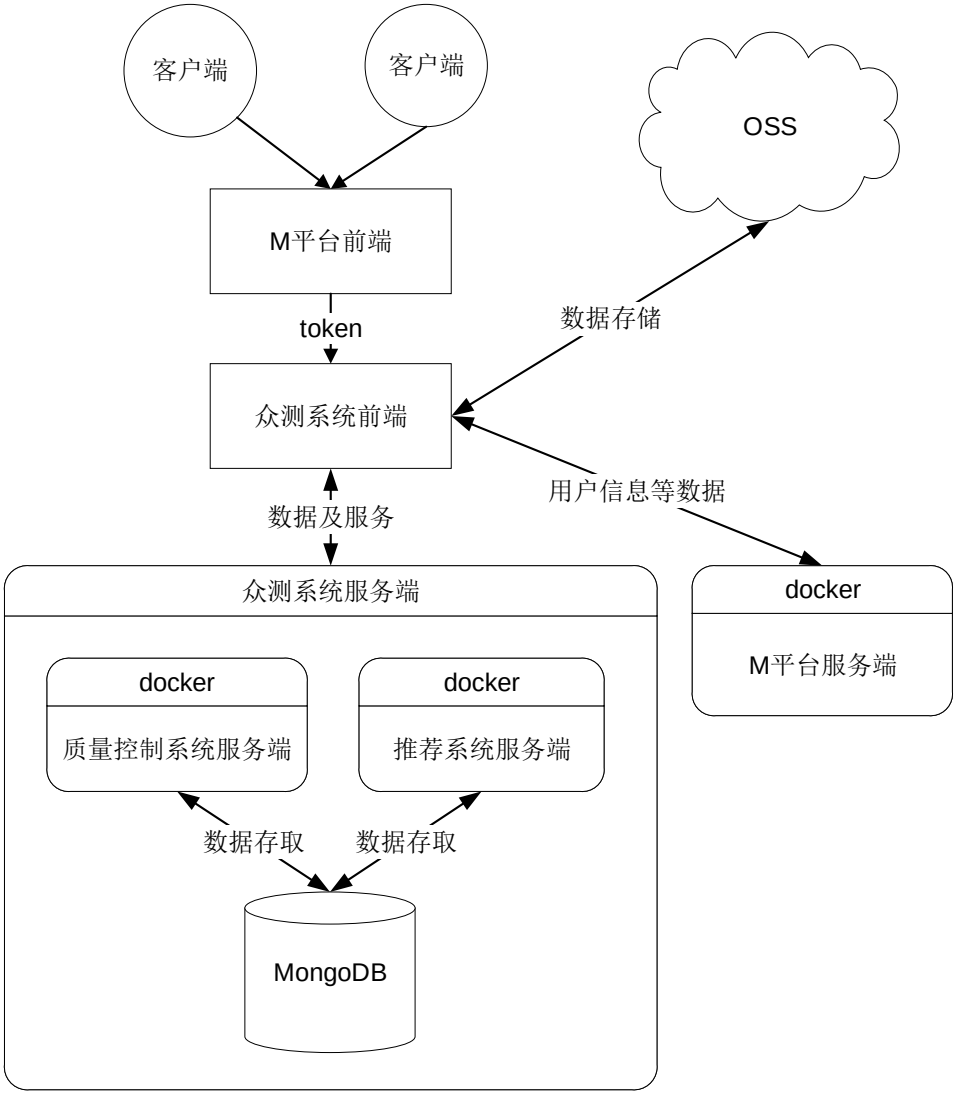

图 3.7: 众测系统架构图

图 3.7所示为众测系统的架构图。客户端首先访问M平台前端，在M平台前端可携带token信息跳转到众测系统前端，token 信息包括众包工人个人信息、测试任务信息等。众测系统前端与阿里云对象存储服务（Object Storage Service，

简称OSS）、众测系统服务端和M 平台服务端交互。OSS提供图片等数据的存储服务；M平台服务端提供用户服务和测试任务服务等，可供前端存取测试报告和测试任务；众测系统服务端包括质量控制系统服务端和推荐系统服务端，分别部署在服务器的docker中。推荐系统服务端提供众包工人协作、引导和提交Bug 报告等服务，质量控制系统在协作式众包测试的全过程采取各种质量控制策略，保证测试结果的高质量，两个系统都与同一个MongoDB数据库交互。

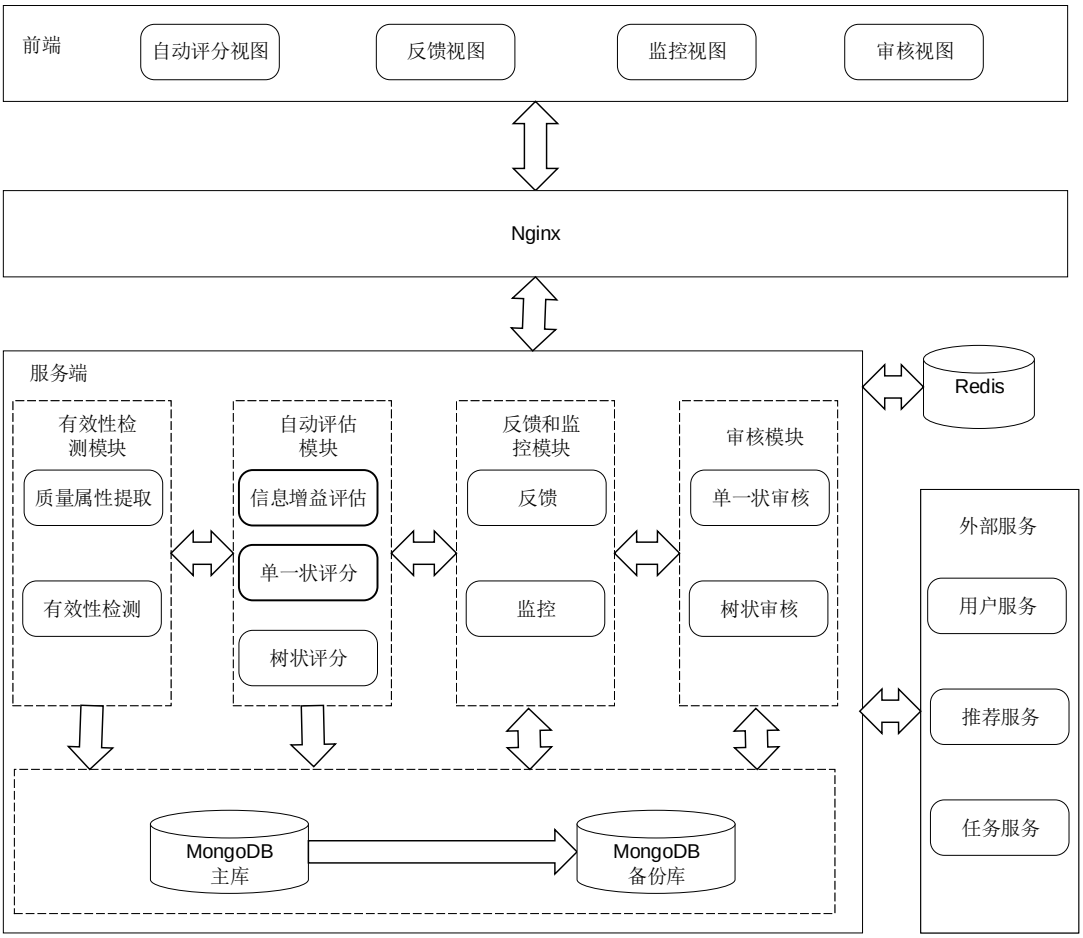

图 3.8: 质量控制系统架构图

质量控制系统的前端集成在M平台前端中，服务端以微服务的形式单独部署。集成前端使得与用户交互密切的前端与M平台保持一致，单独部署服务端使得与功能变更密切的服务端保持灵活性，方便服务的扩展和移植。

质量控制系统的架构图如图 3.8所示。前端主要包括自动评分视图、反馈视图、监控视图和审核视图。自动评分视图对应自动评估模块；反馈视图和监控视图对应反馈和监控模块；审核视图对应审核模块。前端采用Angular2框架，Angular2的组件化开发更易完成复杂业务逻辑，同时保持高效的渲染能力。可

视化库选择Bootstrap，它拥有大量UI组件，通过简单引入和调用即可实现预定好的各种样式。图形可视化库选择Echarts，Echarts基于canvas，拥有丰富的图表类型，且简单易用。前端逻辑处理兼用jQuery库，它语法简洁且兼容性高，极大地简化操作DOM、处理事件等操作，通过引入jQuery就能方便使用。前端消息通知使用PNotify插件，它提供无阻塞的通知，支持Bootstrap界面，可灵活定制且使用简便。

系统服务端采用Spring Boot框架，该框架提供丰富的业务功能，同时基于强大的社区支持可快速集成其他服务组件。服务端需要使用M平台和推荐系统提供的外部服务，对用户进行认证授权，获取推荐服务，获取待测应用软件和测试报告信息等。服务端四个模块之间互相关联，单个模块的业务服务可供其他模块调用，各个模板都可与数据库交互。

前端和服务端之间引入Nginx服务，Nginx通过反向代理实现服务端负载均衡，保证应用服务可迅速水平扩展，提高应用服务器的资源利用率。服务端使用Redis作为缓存数据库，存储用户session等需要在服务器集群共享的临时数据。与进程内缓存相比，共享缓存可使缓存数据不固定于一台服务器，使服务器无状态。数据库使用MongoDB，提高数据库查询速度和可扩展性；同时使用数据库主从备份，提高数据库的容灾能力。

3.3.2　4+1视图

软件架构用来处理软件高层次结构的设计和实施，但是通过单张视图难以捕捉所有的系统要点，为了处理大型复杂的架构，通常使用多个视图或视角组成的模型描述，4+1视图即为一种优秀的模型 [37]。4+1视图是指系统的场景视图、逻辑视图、进程视图、开发视图和物理视图，其关联关系如图 3.9所示。

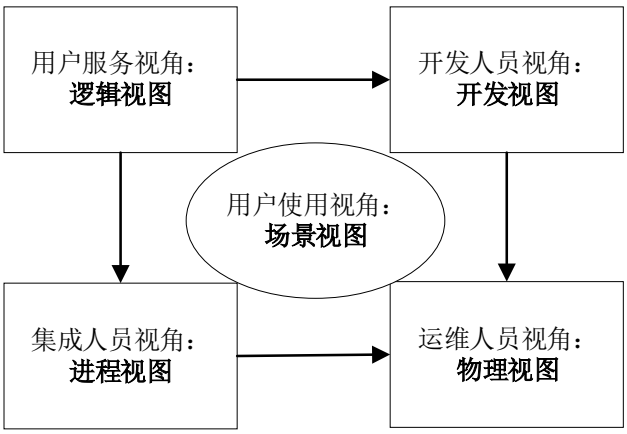

图 3.9: 4+1视图关联关系

本小节使用4+1视图详细介绍系统总体设计。场景视图关注用户使用场景，从需求角度描述系统设计，可使用UML用例图来描述，本系统的场景视图即为3.2.1 节的用例图。以下重点介绍本系统的逻辑视图、开发视图、进程视图和物理视图。

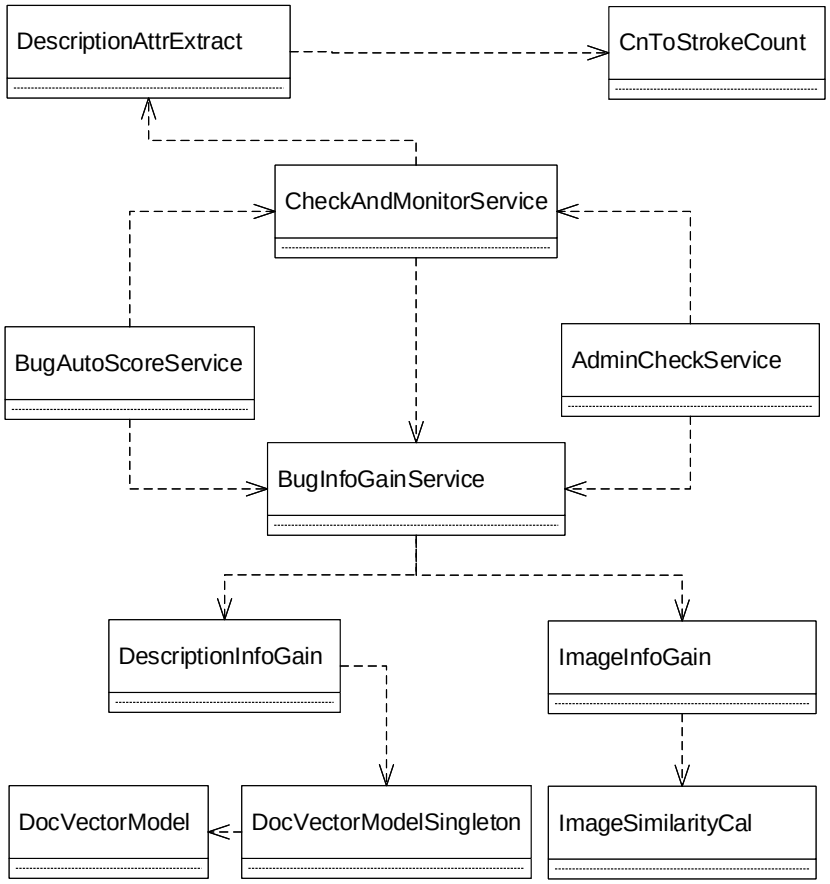

图 3.10: 逻辑视图

逻辑视图从用户服务角度描述系统，重点在于系统提供的用户服务，可使用UML类图表述。如图 3.10所示，将系统组件抽象为类构建逻辑视图。CheckAndMonitorService可提取Bug报告质量指标，依赖DescriptionAttrExtract来具体实现，DescriptionAttrExtract依赖CnToStrokeCount计算文本可读性。同时CheckAndMonitorService提供反馈和监控服务，BugAutoScoreService提供报告质量自动评估服务，AdminCheckService提供报告质量审核服务，三个服务类都依赖BugInfoGainService，后两个service依赖CheckAndMonitorService提取报告质量指标。BugInfoGainService提供报告信息增益评估服务，依赖DescriptionInfoGain获取报告文本信息增益，依赖ImageInfoGain获取报告图片信息增益。DescriptionInfoGain通过DocVectorModelSingleton获取单例类DocVectorModel，从而获

取Word2Vec模型用来计算文本之间的相似度。ImageInfoGain依赖ImageSimilarityCal获取图片特征，并计算图片相似度。

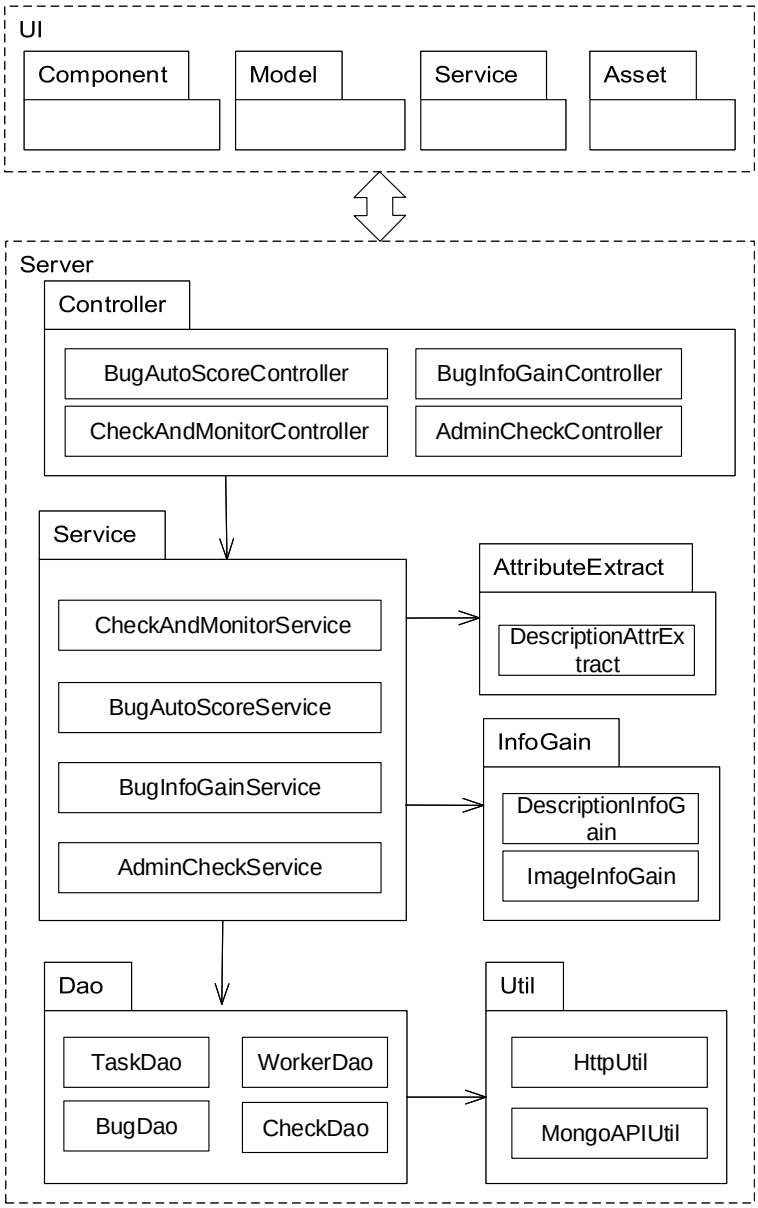

图 3.11: 开发视图

开发视图从开发人员角度设计系统，描述开发环境下的系统模块组织和管理，如图 3.11所示为系统的开发视图。前端UI包括组件Component层、数据结构Model 层、前后端交互Service层和静态文件Asset层。服务端模块按照功能和层级划分。Controller包负责控制器管理，接收并响应前端请求。Service包实现系统相关的业务逻辑，供Controller调用。AttributeExtract包负责报告属性分析以

及质量指标提取。InfoGain包负责报告信息增益评估的实现。Dao包通过Util包提供的HTTP服务，封装Mongodb REST API请求与数据库进行数据交互。

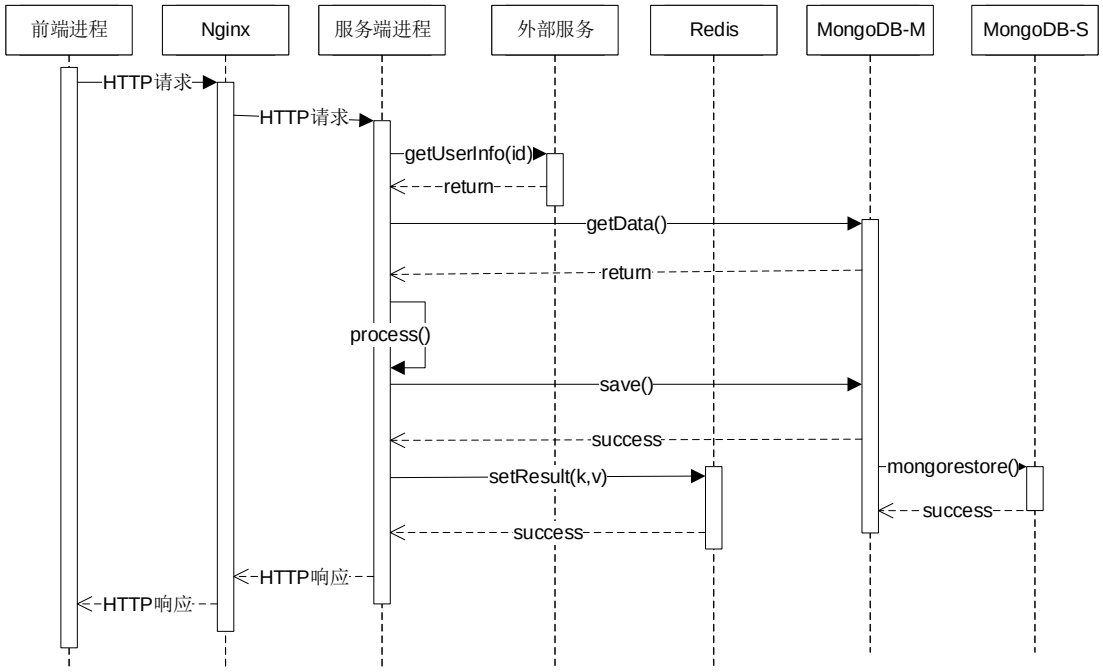

图 3.12: 进程视图

进程视图从系统集成者角度描述系统，重点关注系统的进程任务和进程通信，保证系统的整体性和性能。如图 3.12所示为系统的进程视图。系统前端进程发送HTTP请求到Nginx服务器，Nginx转发HTTP请求到服务端进程。若服务端需要外部数据，例如用户数据和任务数据等，则发送RESTful请求调用外部服务获取。若服务端需要数据库中存储的Bug报告和其他数据等，则服务端进程与MongoDB主库交互获取。获取数据后服务端进程进行相关业务处理。若业务处理过程需要持久化数据，服务端进程存储数据到MongoDB主库中。MongoDB主库每隔一段时间使用MongoDB API自动备份数据到备份库中。若业务处理过程产生热点数据，如众包工人点赞关系和点踩关系等，服务端进程存储热点数据到Redis中，保证该热点数据可被迅速获取，提高系统性能。业务处理结束之后，服务端进程返回HTTP响应到Nginx，Nginx返回HTTP响应到前端进程，前端进程展示业务处理结果。

物理视图面向运维人员视角，展示各个物理节点间的通信和部署情况，图3.13为系统物理视图。用户通过浏览器与系统前端项目交互，在前端项目上发送HTTP请求访问系统服务端，但需要经过防火墙过滤，防止恶意用户攻击。

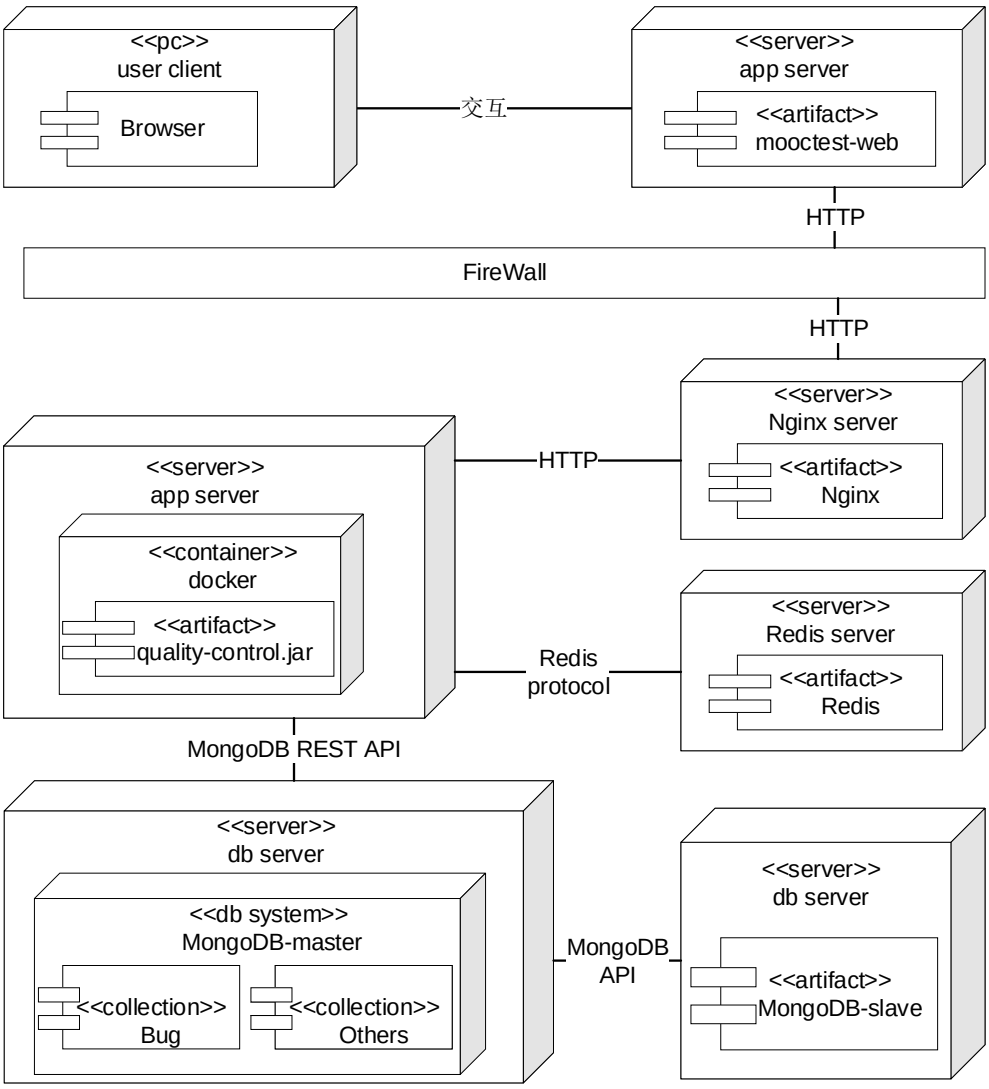

图 3.13: 物理视图

防火墙允许的访问首先访问Nginx服务器，Nginx服务器对请求进行负载均衡，分发请求到多个应用服务器。应用服务器使用docker作为应用服务的容器。应用服务可将请求的session托管到Redis保存，使得用户后续请求可随意被分发到任何一个应用服务器，保证服务的水平扩展性。同时应用服务可从Redis中获取热点数据，提高应用服务的性能。系统数据库单独部署，应用服务器通过MongoDB REST API与数据库进行交互，进行增删改查等操作。数据库分为主库和备份库，主库通过mongodump和mongorestore备份数据到备份库。一旦主库所在的服务器发生故障，系统可迅速切换到备份库，主从备份的设计提高数据库的容灾能力。

3.4　系统实体类与数据库设计

3.4.1　实体类设计

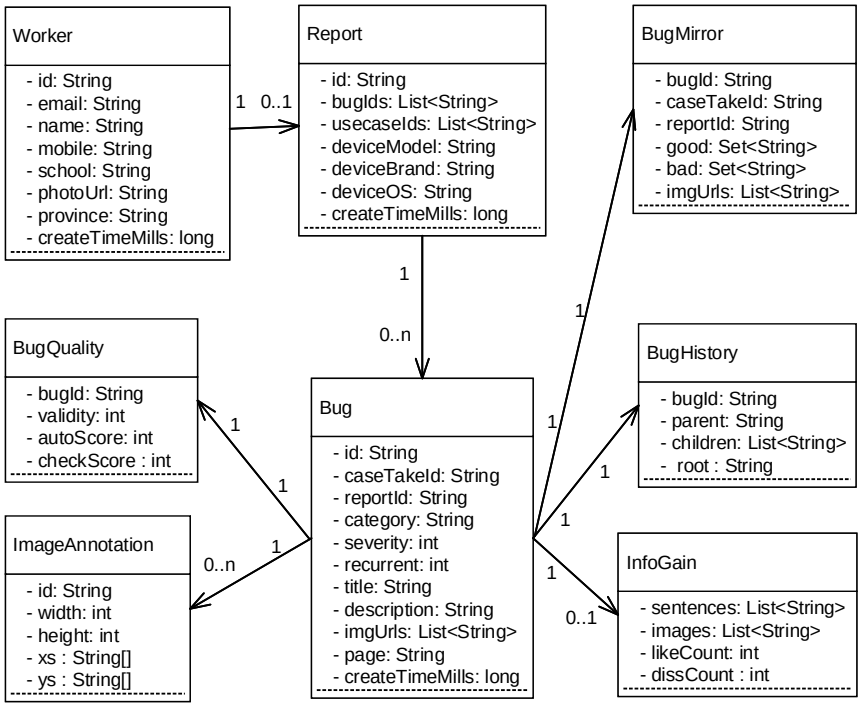

图 3.14: 系统实体类图

表 3.2: Worker类详述

属性	数据类型	含义
id	String	众包工人id
email	String	邮箱
name	String	姓名
mobile	String	手机
school	String	学校
photoUrl	String	个人头像图片在OSS上的地址
province	String	省份
createTimeMills	long	创建时间

质量控制系统的基本实体类设计如图 3.14 所示，以下详细介绍每个实体类以及实体类之间的关联关系。

Worker表示众包工人实体类，其属性具体含义如表 3.2 所示。每个Worker在众测系统中都将提交0个或者1个测试报告Report，Report类的具体含义如表 3.3

所示。每个测试报告Report类中有0或者多个Bug报告，即实体类Bug，其属性具体含义如表 3.4 所示。

表 3.3: Report类详述

属性	数据类型	含义
id	String	测试报告id
bugIds	List⟨String⟩	包含的Bug报告id 列表
usecaseIds	List⟨String⟩	包含的测试用例id列表
deviceModel	String	测试使用的设备型号
deviceBrand	String	测试使用的设备品牌
deviceOS	String	测试使用的设备操作系统
createTimeMills	long	创建时间

表 3.4: Bug类详述

属性	数据类型	含义
id	String	Bug报告id
caseTakeId	String	属于的测试任务id
reportId	String	属于的测试报告id
category	String	缺陷的类别，分为不正常退出、功能不完整、用户体验、页面布局缺陷、性能、安全和其他七个类别
serverity	int	缺陷的严重等级，分为待定、较轻、一般、严重和紧急五个等级，分别用1-5表示。
recurrent	int	缺陷可复现的程度，分为其他、无规律、小概率、大概率和必现五个等级，分别用1-5表示
title	String	Bug报告的题目
description	String	关于缺陷的描述
imgUrls	List⟨String⟩	缺陷截图在OSS上的地址集合
page	String	缺陷在软件上发生的页面
createTimeMills	long	创建时间

Bug类是系统最重要的实体类，每个Bug实例代表一个Bug报告。属性id 为报告的唯一标识符，caseTakeId为报告所属的测试任务id，reportId为报告所属的测试报告id。category、serverity 和recurrent分别为缺陷的类别、严重等级和可复现程度。title为报告题目，代表该报告的摘要。description为报告的描述，包括所发现缺陷的前提、步骤和缺陷表现形式等。imgUrls为缺陷截图在OSS上的地址集合，page为缺陷所在的软件页面，createTimeMills为报告的创建时间。

每个Bug报告都对应1个BugMirror类实例，BugMirror类封装Bug报告的部分属性和审核信息，用于Bug报告的列表展示，其属性具体含义如表 3.5 所示。

表 3.5: BugMirror类详述

属性	数据类型	含义
bugId	String	Bug报告id
caseTakeId	String	参与的测试任务id
reportId	String	属于的测试报告id
good	Set⟨String⟩	点赞该Bug报告的测试报告id集合
bad	Set⟨String⟩	点踩该Bug报告的测试报告id集合
imgUrls	List⟨String⟩	缺陷截图在OSS上的地址集合

表 3.6: BugHistory类详述

属性	数据类型	含义
bugId	String	Bug报告id
parent	String	父报告id，无父报告则为“null”
children	List⟨String⟩	子报告id列表，无子报告则为空
root	String	所在报告树根节点报告id

表 3.7: InfoGain类详述

属性	数据类型	含义
sentences	List⟨String⟩	子报告增益的句子描述集合
images	List⟨String⟩	子报告增益的截图OSS地址集合
likeCount	int	子报告增益的点赞数
dissCount	int	子报告增益的点踩数

每个Bug报告都存在1个与之对应的BugHistory类实例，BugHistory封装报告历史信息，包括父报告和子报告信息等，其属性具体含义如表 3.6 所示。

InfoGain类封装树状报告的增益信息，只有存在父报告的子报告才存在该类实例，其属性具体含义如表 3.7所示。

ImageAnnotation类封装Bug报告截图的标注信息，存储每个截图的标注，1个截图可对应多个ImageAnnotation类实例，表示截图上的多个标注。类中id为标注标识符，height 和width 属性分别为截图的高度和宽度。截图标注以点集的形式存储，xs表示点集的横坐标集合，ys 表示点集的纵坐标集合。

BugQuality类封装Bug报告的质量信息，每个报告都存在1个BugQuality实例。属性bugId表示其属于的Bug报告id，validity表示报告有效性，autoScore代表报告的自动评分，checkScore代表报告的人工审核得分。

3.4.2　数据库设计

基于MongoDB性能高、灵活性好和易于扩展的优势，系统使用MongoDB作为数据库。由于MongoDB是非关系型数据库，且没有数据库模型的约束，因而不能使用ER图（Entity Relationship Diagram，实体联系图）来表述设计。为了保证质量控制系统的高性能和易扩展性，本文为系统的每一个实体类设计对应的MongoDB Collection，Collection存储的文档即为实体类对应的实例，文档均为BSON格式，存储类实例的属性与值。表 3.8为数据库中Bug报告对应的Bug Collection 文档详述，其他实体类对应的Collection设计与实体类设计差异不大，Collection字段与实体类属性相同，仅字段类型与属性类型存在差异，此处不做赘述。

表 3.8: Bug Collection文档详述

字段	数据类型	含义
id	String	Bug报告id
caseTakeId	String	参与的测试任务id
reportId	String	属于的测试报告id
category	String	缺陷的类别，分为不正常退出、功能不完整、用户体验、页面布局缺陷、性能、安全和其他七个类别
serverity	int32	缺陷的严重等级，分为待定、较轻、一般、严重和紧急五个等级，分别用1-5表示。
recurrent	int32	缺陷可复现的程度，分为其他、无规律、小概率、大概率和必现五个等级，分别用1-5表示
title	String	Bug报告的题目
description	String	关于缺陷的描述
imgUrls	Array	缺陷截图在OSS上的地址集合
page	String	缺陷在软件上发生的页面
createTimeMills	String	创建时间

MongoDB默认开启一个HTTP协议的端口提供REST服务，使用REST API HTTP 请求可对MongoDB进行CURD（Create、Update、Retrieve、Delete 等操作简称）操作，表 3.9所示为使用REST API HTTP请求详述。质量控制系统通过Spring Boot的RestTemplate类封装数据库操作请求。RestTemplate是Spring Boot用于同步client端的核心类，简化程序与HTTP服务的通信，并满足RESTful原则。程序代码给RestTemplate提供HTTP URL，其可将响应转化为实体类或其他结构。使用RestTemplate能大幅简化表单提交的难度，提升系统开发效率。

表 3.9: REST API HTTP请求详述

请求类型	用途	响应
POST	创建新纪录	数据库中插入新纪录
GET	读取记录	从数据库返回指定记录
PUT	更新记录	更新数据库中指定记录
DELETE	删除记录	数据库中指定记录被删除

3.5　系统模块详细设计

3.5.1　Bug报告有效性检测模块

Bug报告有效性检测模块根据报告收集的基本信息，提取质量指标，并根据质量指标确定报告有效性，为反馈众包工人操作和自动评估奠定基础。

(1) Bug报告属性分析

众包工人在众测系统上测试时，在每个测试任务中都要提交一份测试报告，每个测试报告都包含多个Bug报告，根据工业界对测试报告和Bug报告的定义，众测系统使用实体类Report 和Bug分别封装测试报告和Bug报告，其具体属性已在实体类设计中详细描述。

本文重点介绍Bug报告的page属性。由于众包工人在众测系统上手工测试时，需要安装并运行软件来发现软件缺陷，通常发现的缺陷发生在特定软件页面内。为了方便定位并复现缺陷，众测系统抽象出“page”属性记录缺陷所在的页面，作为缺陷的位置标签。为了保持Bug报告中page属性的一致性，众测系统管理者对每个待测应用或者APP生成一个软件页面结构表格文件。表 3.10 为途牛APP页面结构示例，该APP下有两个一级页面“搜索”和“我的”，“搜索”页面下有两个二级页面“推荐目的地”和“热门搜索”，“推荐目的地”页面下有两个三级页面“地区选择”和“查看更多”。众测系统自动读取表格文件为测试任务生成一致的页面选择框，供众包工人选择报告所在页面。

表 3.10: 途牛APP页面结构示例

一级页面	二级页面	三级页面
搜索	推荐目的地	地区选择
		查看更多
	热门搜索	品类
		周边
		主题
我的	推荐有礼	推荐方式

(2) Bug报告质量指标

质量是一个模糊的概念，质量评价依赖于不同的评价标准。本文参考文献 [38]和 [39]对Bug报告质量影响因素的研究，从形态学、词法和语义方面提取Bug报告的多个质量指标。在众测系统上，众包工人能以点赞或点踩的方式审核他人Bug报告，点踩和点踩类似于公众投票，故点赞和点踩数目可作为报告的质量指标。结合众测系统要求众包工人填写的报告补充信息：测试环境、截图和截图标注，本文从形态学、词法、语义、审核和补充信息等五个方面切入，提取影响报告质量的多个指标。表 3.11 详述本文设计的Bug报告质量指标。

表 3.11: Bug报告质量指标

类别	指标	含义
形态学	文本长度	Bug描述的文本长度
	可读性	理解文本的难易程度
	句子长度	描述中每个句子的平均长度
词法	不精确词	不精确的词汇，常导致描述的不确定性。例如：也许，大概
	代词	指代性词汇，常导致文本描述不清晰。例如：这个，那个
	举例词	举例子的描述词，常导致文本描述更易理解。例如：比如，举例
	步骤词	描述中形容步骤的词汇。例如：第一，第二
语义	否定词	否定性词汇，否定词的出现常意味着缺陷的出现。例如：无法，不能
	行为词	形容软件行为的动词。例如：崩溃，退出
	动作词	形容人为动作的动词。例如：安装，点击
	界面组件	形容软件界面组件的名词。例如：按钮，菜单
审核	点赞数	众包工人对报告的点赞数目
	点踩数	众包工人对报告的点踩数目
补充	测试环境	测试时的设备环境
	截图	缺陷的截图
	截图标注	众包工人对截图的标注

对于表 3.11中所述的质量指标，本文从Bug 报告的基础属性中提取。由于形态学、词法和语义指标都与缺陷的语义有关，因此从报告的description属性中提取。description属性实际上是中文长文本，系统使用中文NLP工具HanLP来处理description，对中文文本进行断句、分词、词语匹配等。五种类型的指标提取方法如下。

1) 形态学指标

文本长度即为Bug报告descriptioon长度；可读性参考文献 [40]提供的可读性公式，如式3.1所示，其中X_1 表示较难词汇的比例，X_2和X_3 表示句子数和平均

笔画数；句子的平均长度指description中每句话的平均长度。

$$Y = 14.9596 + 39.07746 \times X_1 + 1.011506 \times X_2 - 2.48 \times X_3 \tag{3.1}$$

2) 词法指标

词法类型的所有指标都属于词汇性的指标，首先使用HanLP将description 分词，去除停止词后留下主干词汇。本文联系文献 [38]作者Chen等人，得到其实验中总结的不精确词、代词、举例词和步骤词的词典，通过对比description主干词和词法字典中词汇统计指标数值。表 3.12 所示为Chen等人提供的指标字典词汇示例。

表 3.12: 指标字典词汇示例

类别	指标字典	词汇示例
词法	不精确词	好的、坏的、中等的、适度的、温和的、充足的、高效的
	代词	它、它们、他、他们、她、她们、那、那个
	举例词	例如、例、如、即、注意、比如、举例
	步骤词	第一、一、(1)、(一)
语义	否定词	没、没有、决不、毫不、既不、也不、无处、都不
	行为词	崩溃、失败、退出、安全、布局、运行、特殊、识别
	动作词	查看、扫描、滑动、切换、登录、刷新、提交
	界面组件	工具栏、按钮、菜单、对话框、窗口、界面、滚动条、屏幕

3) 语义指标

语义指标与词法指标的提取过程类似，通过对比description主干词和Chen提供的语义字典中词汇统计各个指标值。

4) 审核指标

点赞和点踩Bug报告的信息存储在与Bug报告关联的BugMirror类，其good和bad属性分别记录点赞和点踩该报告的工人列表，取两个属性的长度即为点赞和点踩数目。

5) 补充指标

众包工人提交的测试报告Report类中有deviceModel、deviceBrand和device-Model 属性，代表测试的设备环境，若众包工人规范填写了该属性，则指标值为1，未规范填写则为0。截图指标是指Bug报告所包含的截图数量，即Bug类中imgUrls属性的长度。截图标注指标是指众包工人在所有截图上标注的总数，通过ImageAnnotation类获取。

提取质量指标之后，质量控制系统使用实体类BugAttribute封装质量指标，该类详述如表 3.13 所示。

表 3.13: BugAttribute类详述

属性	数据类型	含义	属性	数据类型	含义
id	String	Bug报告id	action	double	动作词
description	String	文本长度	interfaceElement	double	界面组件
readability	double	可读性	itemization	double	步骤词
sentenceLength	double	句子长度	imageCount	double	截图数量
imprecise	double	不精确词	likeCount	double	点赞数
anaphoric	double	代词	dislikeCount	double	点踩数
directive	double	举例词	imageAnnotation	double	截图标注
negative	double	否定词	environment	short	测试环境
behavior	double	行为词	score	double	自动评分

(3) Bug报告有效性检测

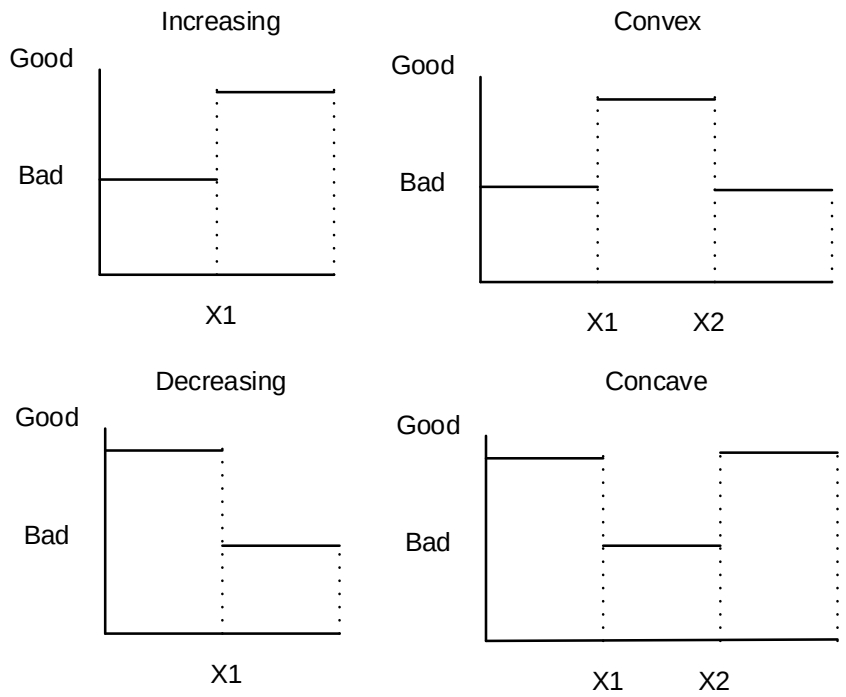

图 3.15: 四种阶跃变换函数

提取报告质量指标值之后，系统需要确定每个量化指标的质量等级。一般而言，质量可分为两个不同的等级：好与差、高与低等。为了将指标数值转化为质量等级，本文引入阶跃变换函数，

通常阶跃变换函数分为四类：Increasing、Decreasing、Convex 和Concave四种，图 3.15所示为四种阶跃变换函数。其中Increasing和Decreasing函数涉及一

表 3.14: 质量指标变换函数及有效区间

类别	指标	函数	有效区间(x 代表指标值)
形态学	文本长度	Convex	$15 \le x \le 30$
	可读性	Concave	$x \le -1, x \ge 3$
	句子长度	Convex	$8 \le x \le 15$
词法	不精确词	Decreasing	0
	代词	Decreasing	0
	举例词	Increasing	$x \ge 1$
	步骤词	Increasing	$x \ge 1$
语义	否定词	Increasing	$x \ge 1$
	行为词	Increasing	$x \ge 1$
	动作词	Increasing	$x \ge 1$
	界面组件	Increasing	$x \ge 1$
审核	点赞数	Increasing	$x \ge 1$
	点踩数	Decreasing	0
补充	测试环境	Increasing	1
	截图	Increasing	$x \ge 1$
	截图标注	Increasing	$x \ge 1$

个参数，而后两种函数涉及两个参数。通过设置合理的参数值，四种函数可将连续的指标值映射到相应的等级。对于Increasing函数，指标值大于参数$X1$时被认为质量好，小于$X1$被认为质量差；Decreasing与Increasing正好相反。对于Convex函数，指标值处于$X1$和$X2$之间时被认为质量好，否则是质量差的；Concave与Convex正好相反。例如对于指标文本长度，一份好的Bug报告应该保持文本长度既不太长也不太短，此处可使用Convex的阶跃变换函数。通过参考文献 [38]及询问测试领域专家，本文各质量指标采取的阶跃变换函数和有效区间如表 3.14所示。

通过表 3.14中的有效性范围，可将各质量指标转换为离散值0和1，0代表该指标无效，1代表该指标有效。例如对于形态学类型中的文本长度指标，当报告描述的文本长度不小于15且不大于30 时，则文本长度指标值被转换为1，否则被转换为0；对于词法类型中的不精确词指标，当报告描述中未出现任何不精确词汇时，不精确词指标被转换为1，否则被转换为0; 对于语义类型中的否定词指标，当报告描述中出现1次或多次否定词时，否定词指标被转换为1，否则被转换为0。获取所有指标离散值后，系统检测报告有效性。本文定义，若Bug 报告有效指标总数超过*sum_threshold*，则该报告有效，否则该报告无效。*sum_threshold* 的值由验证实验确定。

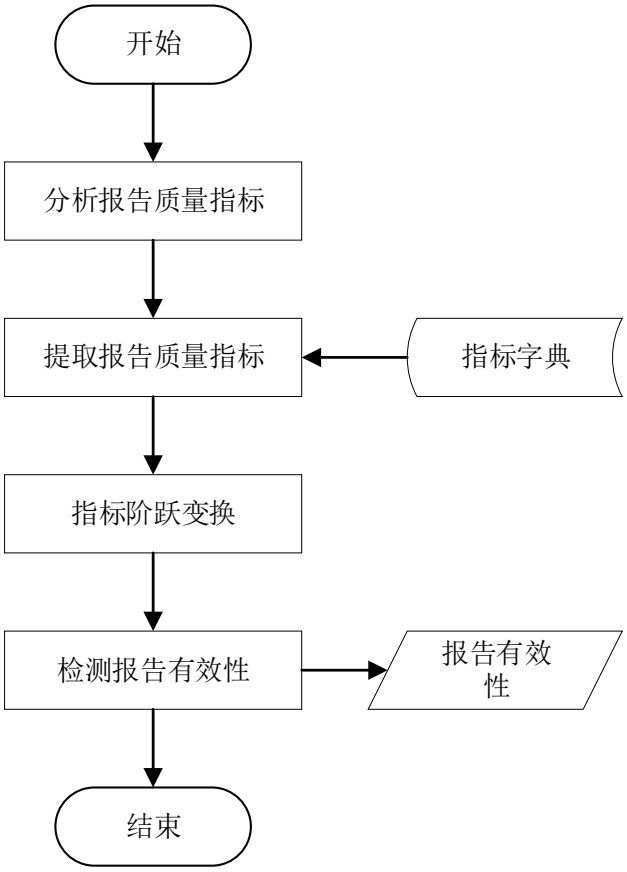

图 3.16: Bug报告有效性检测流程图

图 3.16所示为Bug报告有效性检测流程图。首先本文分析确定报告质量指标。其次根据指标字典和报告属性提取报告质量指标。然后根据指标的阶跃变换函数对各指标进行阶跃变换。最终，检测每个报告有效质量指标总数是否超过*sum_threshold*，超过则判定该报告有效，否则无效。

(4) 核心类设计

图 3.17所示为Bug报告有效性检测模块核心类图。CheckAndMonitorService类负责实现有效性检测功能，向外提供checkBug()方法检测报告有效性。Service类拥有BugDao接口的实现类实例，通过该实例完成与数据库的交互，获取Bug、BugHistory和BugMirror等实体类信息。同时Service类依赖MongoAPIUtil类生成数据库查询语句，依赖DescriptionAttrExtract类提取报告质量指标。DescriptionAttrExtract类依赖CnToStrokeCount计算报告文本可读性，依赖BasicTokenizer类对报告文本进行分词，依赖DataPathUtil类获取指标字典及Word2Vec语料库的存储地址。

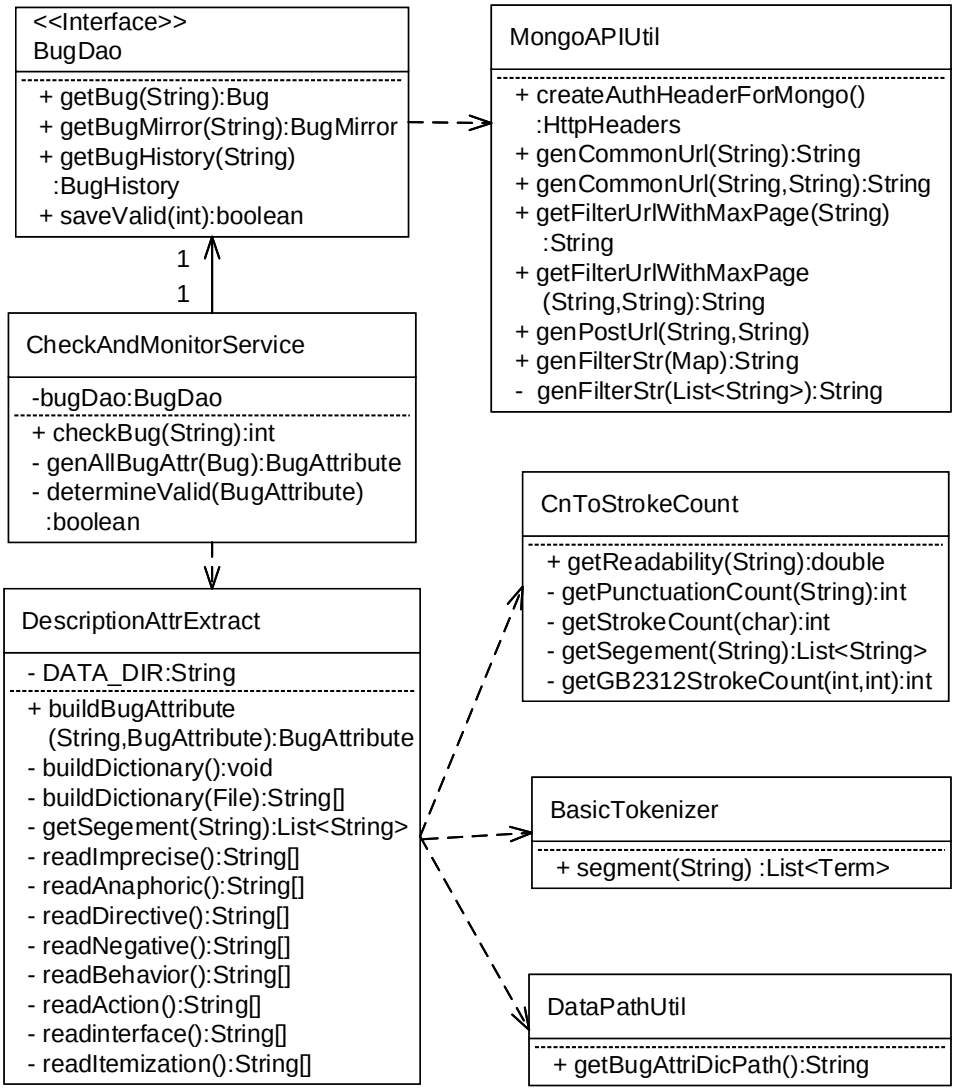

图 3.17: Bug报告有效性检测模块核心类图

3.5.2　Bug报告自动评估模块

(1) 报告信息增益评估

众测系统中所有的Bug报告都是信息透明的，众包工人在填写Bug报告过程中，系统根据用户输入推荐相似报告列表，也会根据当前工人完成任务的情况进行测试任务推荐和审核推荐。众测系统允许众包工人对他人Bug报告复制修改，增加信息以提高报告质量，同时减少报告的重复率，该操作称为Fork操作。经过Fork操作之后，原报告称为父报告，修改提交的新报告称为子报告，一个根节点报告和众多子报告将形成一棵Bug报告树，图 3.18所示为一棵Bug报告树示例。该报告树由5个Bug报告组成，其中序号为698的报告为该报告树的根

节点；根节点下有三个子报告，分别为712、717和917；917报告有一个子报告为194。

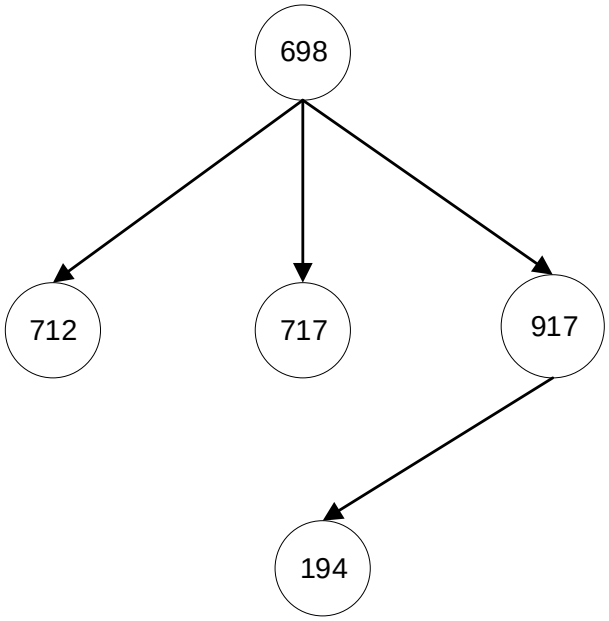

图 3.18: Bug报告树示例图

信息透明的Bug报告初衷为了减少报告重复率，使众包工人修改补充从而提高报告质量，同时自动聚类相似的Bug报告。但是，系统可能存在恶意众包工人，将信息透明作为抄袭作弊的途径，使得提交的子报告并无信息增益，甚至与父报告完全相同。为了检测并避免此类行为，质量控制系统必须评估子报告的信息增益。

在Bug报告属性中，描述和截图是重要的质量属性，众包工人审核的点赞数和点踩数也反映了报告的质量，因此系统从描述、截图、点赞数和点踩数四个方面评估报告信息增益。点赞点踩数的差异通过比较数目即可直接得到，句子和截图的差异对比较为复杂，以下分别描述。

句子为中文文本，对于文本相似度，本文使用中文维基百科语料库，经HanLP训练之后得到本文的Word2Vec模型，通过该模型提取两个句子的向量，计算余弦相似度作为文本相似度。同时，系统进行实验并人工确定相似度阈值*document_threshold*，相似度大于该阈值的两个句子被认为是相同句子，无实际语义的增益。通过对比父子报告的所有句子，根据相似度筛选出子报告不同于父报告中任一句子的句子，该句子即为子报告文本的增益信息。

截图为图片，对于图片相似度，本文首先使用Lire[1]框架提取图片的FCTH特征或者CEDD特征，再使用Lire算子提供的距离度量方法计算图片相似度。同时，系统进行实验并人工确定相似度阈值*image_threshold*，相似度大于该阈值的两张图片被认为是相同图片，无实际增益。对比父子报告所有截图，筛选出子报告不同于父报告中任一截图的截图，该截图即为子报告图片的增益信息。

图 3.19所示为Bug报告信息增益评估架构图。首先将父子报告的描述分割为句子，下载父子报告的截图到本地，同时获取父子报告点赞点踩数目。然后，比较父子报告句子的文本相似度和图片相似度，筛选出子报告补充的句子和截图，同时计算子报告增益的点赞数目或者点踩数目。经过该过程，系统可提取出子报告相对其父报告的增益信息，包括句子、截图和点赞点踩数等，并将其建模为InfoGain 类，InfoGain 类在表 3.7 中已经详细描述。

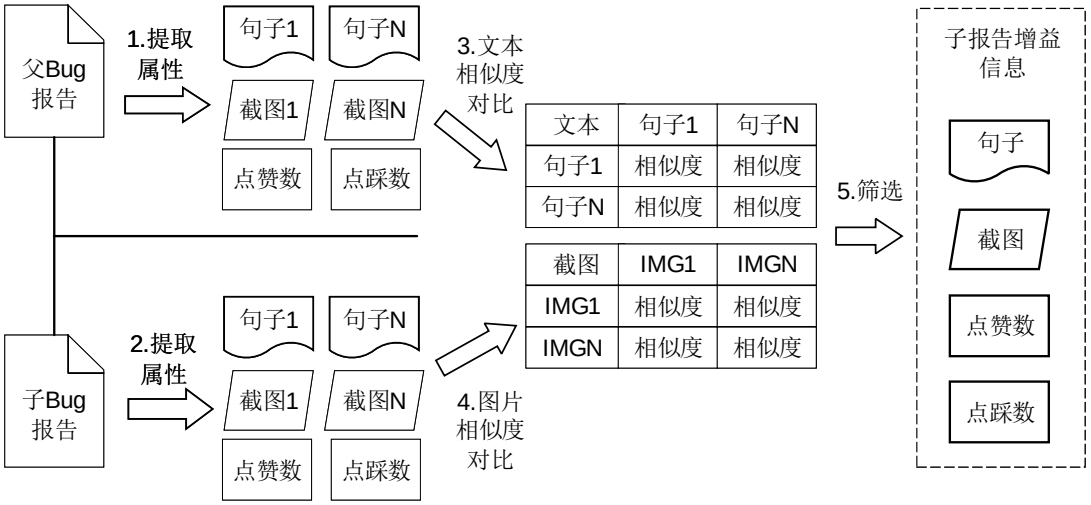

图 3.19: Bug报告信息增益评估架构图

(2) 报告自动评分

众测系统的一次测试任务结束时，管理者需要对Bug报告进行质量评估，并以报告质量评估众包工人的贡献。由于每个任务都产生数量庞大的Bug 报告，人为评估效率低下且公平性难以保证，因此需要质量控制系统对报告进行自动评估，本文量化为报告的得分。自动评分的结果可作为报告的最终评分，也可作为管理者审核的协助手段。

由于协作式众包测试中众包工人可单独或者Fork提交Bug报告，因此Bug报告包括单一状和树状报告两种。单一状报告没有父报告，而树状报告有父报告，质量控制系统对两种类型报告采取不同的自动评分策略。

[1]http://www.lire-project.net/

1) 单一状报告自动评分

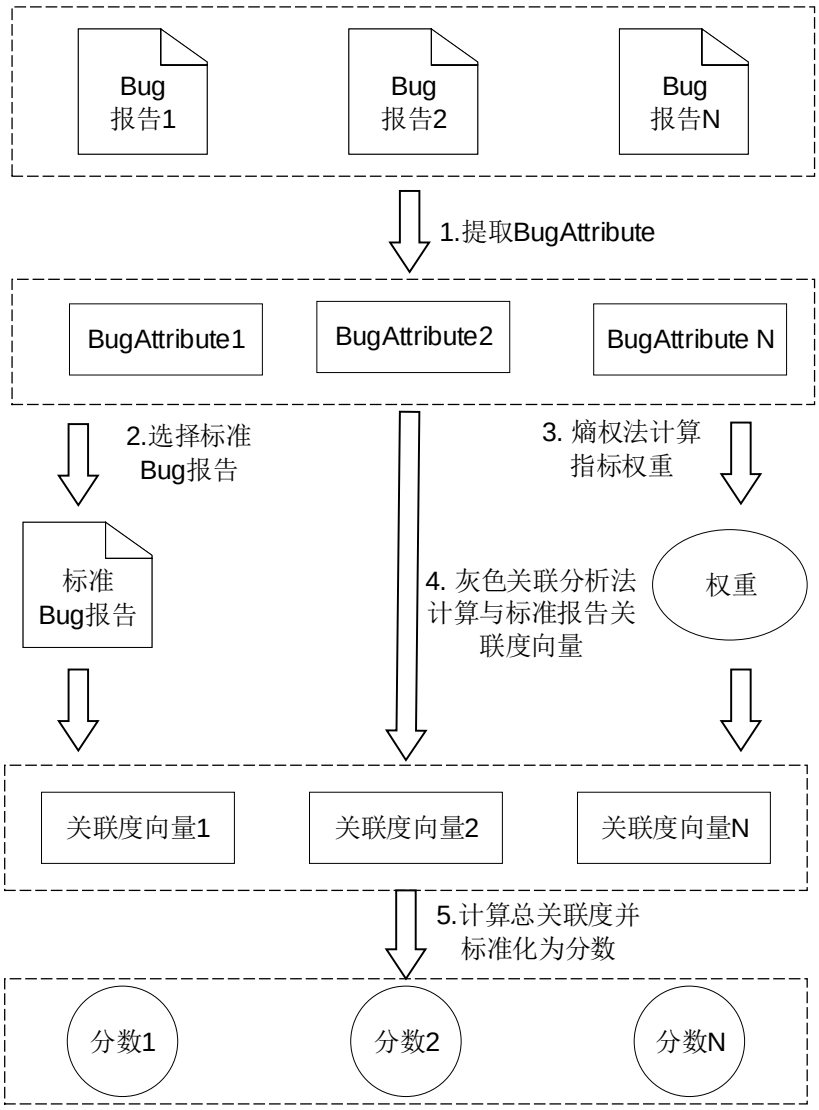

图 3.20: 单一状Bug报告自动评分架构图

众包工人独立完成并提交的报告为单一状Bug报告，单一状Bug报告与传统的竞争式众测系统提交的报告相同，都属于众包工人独立的劳动成果，因此本文依据报告的质量指标对报告评分。

图 3.20为单一状Bug报告自动评分架构图。首先，系统提取某次测试任务的所有单一状报告，对每个报告提取其质量属性BugAttribute类实例。其次，系统根据表 3.14确定的质量指标有效区间，选择每个指标集合中最优指标组成标准Bug报告。同时，系统根据BugAttribute集合利用熵权法确定各个质量指标在评估中所占的权重W。然后，利用灰色关联分析法计算每个报告质量指标与标

准报告质量指标的关联度向量C，指标向量C与指标权重W 的乘积m即为各个报告与标准报告的总关联度。最后，将m标准化到分数区间(本文报告评分的区间为0-10分)即可得到每个报告的最终评分s，即完成对单一状报告的自动评分。

2) 树状报告自动评分

众包工人经过Fork操作提交的报告称为树状Bug报告，该类报告不属于众包工人独立的劳动成果，而是对他人报告的修改补充，建立在他人的成果之上，因此需要采用不同于单一状报告的评分策略。

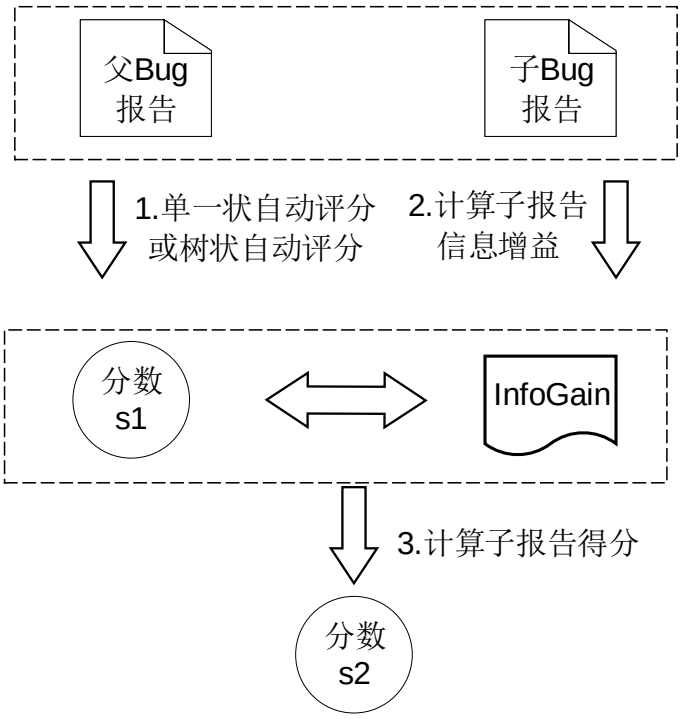

图 3.21: 树状Bug报告自动评分架构图

图 3.21为树状报告自动评分架构图。树状报告由于存在父报告，每个树状报告都存在于特定的Bug报告树T中。由于T 的根节点报告root没有父报告，属于单一状报告，则可通过单一状报告自动评分的方法计算root的分数。得到root分数之后，就可将其看做图3.19 所示的父报告，root分数即为父报告的分数$s1$，其子报告的基础分数继承父报告的分数，也为$s1$。

然后，使用前文所述的子报告信息增益评估方法得到子报告信息增益InfoGain。对于InfoGain中子报告的文本增益sentences，组合句子为文本并提取质量指标，包括directive、negative、behavior、action、interfaceElement 和Itemization 。若子报告的增益句子中含有上述质量指标，则子报告增加$score1$分。对于InfoGain中子报告的图片增益images，子报告每增加一张不同截图，则

增加*score*2分。对于InfoGain中的likeCount，子报告得到的点赞数目增加则增加*score*3 分，减少则减少*score*3分。对于InfoGain中的dissCount，子报告得到的点踩数目减少则增加*score*4分，增加则减少*score*4 分。

子报告基础分*s*1经过信息增益导致的分数变化之后，即可得到子报告的得分*s*2，由于本文所定的报告分数区间为0-10分，*s*2不得少于0分，也不得大于10分，否则将其调整为0或10分。

root报告的子报告只是报告树中深度为1的报告，对于深度为2的报告，只需将深度为1的报告作为父报告，深度为2的报告作为子报告即可。对其余深度的节点评分重复上述过程即可。

(3) 核心类设计

图 3.22为Bug报告自动评估模块核心类图。BugAutoScoreController为自动评分入口，其中的noParentAutoScore()方法处理前端的单一状报告评分请求，hasParentAutoScore()方法处理树状报告评分请求，并且持有BugAutoScoreService类实例完成具体业务逻辑。图中未展示BugInfoGainController类，该类处理子报告信息增益评估请求，调用BugInfoGainService类实现信息增益评估，可为所有模块提供服务。

BugAutoScoreService类的handleNoParentAutoScore()方法实现单一状报告自动评分的具体业务逻辑，handleHasParentAutoScore()方法实现树状报告自动评分的具体业务逻辑，其内部的其他方法为这两个方法提供特定服务。同时该Service持有CheckAndMonitorService类的实例，通过该实例获取Bug报告的质量属性。

BugAutoScoreService通过其持有的TaskDao和BugDao实例与数据库交互，获取需要评分的Bug报告列表以及其他实体类信息；通过其持有的BugAutoScoreDao实例获取或存储报告的自动评分信息。

BugInfoGainService提供获取报告信息增益的能力，其对外提供获取报告信息增益的getInfoGain()方法。该Service依赖ImageInfoGain类获取报告的图片增益，通过调用其getDiffImage()方法获取子报告中增益的图片；依赖DescriptionInfoGain类获取报告的文本增益，通过调用其getDescriptionInfoGain()方法获取子报告中增益的文本。

ImageInfoGain依赖ImageSimilarityCal类计算不同图片之间的相似度，抽象出ImageSimilarityCal类专一负责图片相似度计算，便于以后的算法替换和扩展。DescriptionInfoGain类依赖DocVectorModelSingleton类获取HanLP的Word2Vec单例类，实现文本相似度计算。

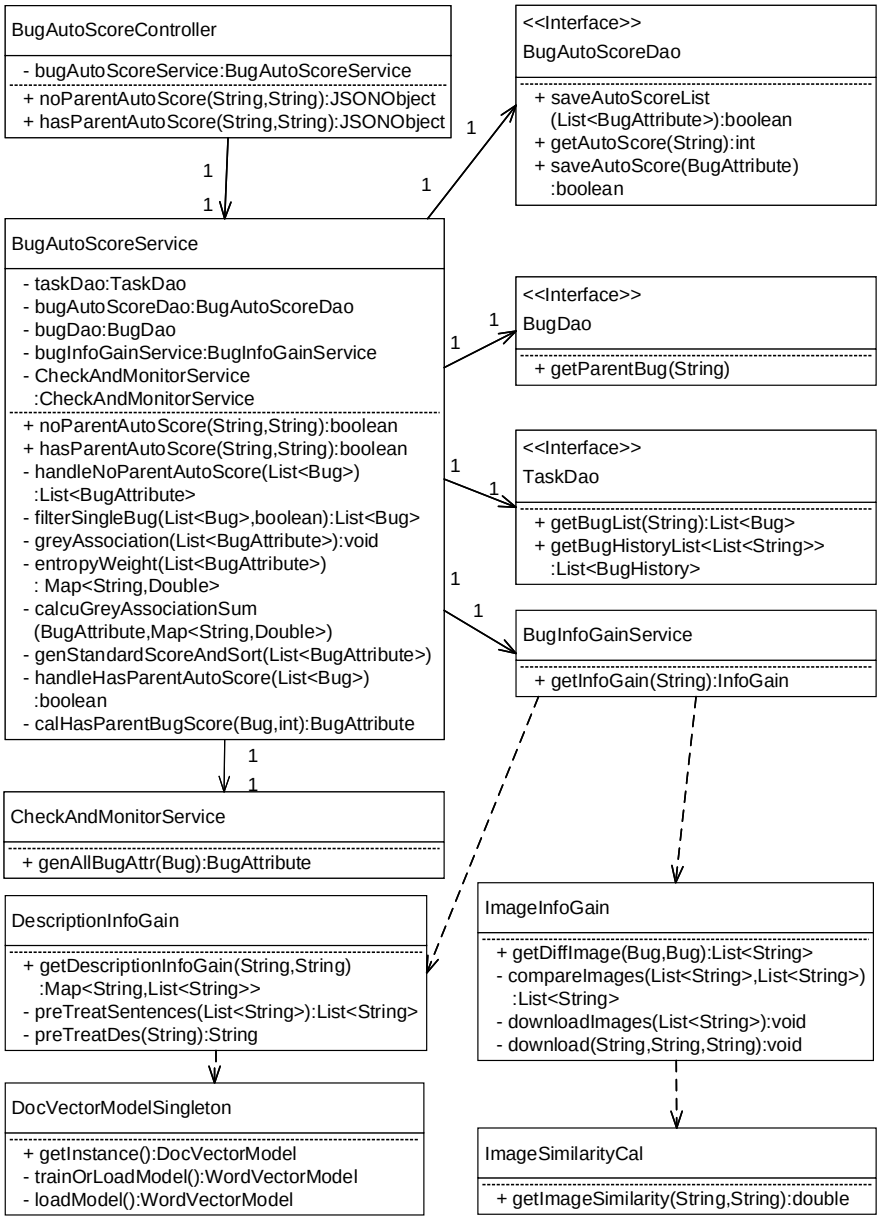

图 3.22: Bug报告自动评估模块核心类图

3.5.3　反馈和监控模块

众包工人为了得到经济或其他奖励，在众测系统上可能采取不合理操作，如表 3.1 所述，不合理操作称为失信行为。为了激励众包工人的合理行为，警告失信行为，质量控制系统对众包工人的每个操作都进行评估和实时反馈，并记录其失信行为。

同时，为了使管理者实时查看众包工人的失信行为及交互情况，系统设计

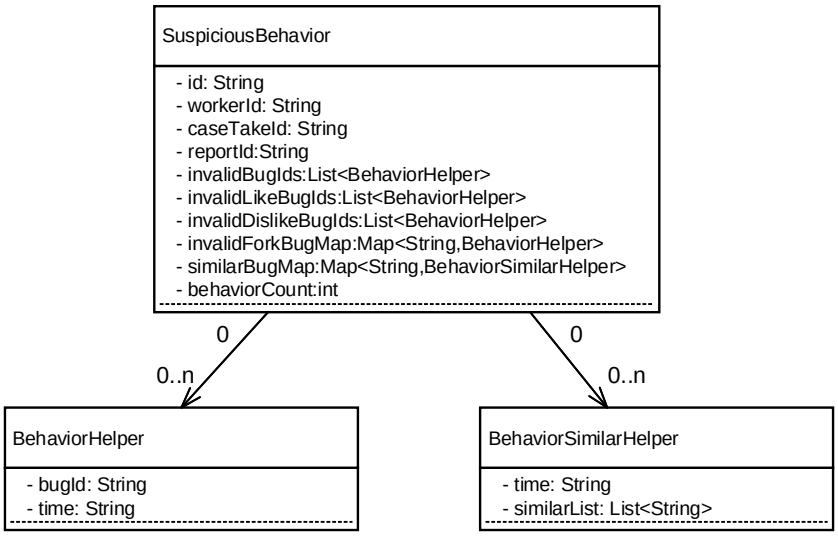

图 3.23: SuspiciousBehavior及其关联类图

了监控模块，实时展示众包工人的失信列表、众包工人Fork关系图、点赞关系图和点踩关系图等，帮助管理者提早发现恶意众包工人，并终止其参与测试任务的资格。

表 3.15: SuspiciousBehavior类详述

属性	数据类型	含义
id	String	唯一标识符
workerId	String	众包工人id
caseTakeId	String	测试任务id
reportId	String	测试报告id
invalidBugIds	List⟨BehaviorHelper⟩	提交的无效Bug 报告id 与时间集合
invalidLikeBugIds	List⟨BehaviorHelper⟩	恶意点赞的Bug 报告id 与时间集合
invalidDislikeBugIds	List⟨BehaviorHelper⟩	恶意点踩的Bug 报告id与时间集合
invalidForkBugMap	Map⟨String, BehaviorHelper⟩	key为提交的无信息增益的子报告id,value 为Fork 的父报告id与时间
similarBugMap	Map⟨String, BehaviorSimilarHelper⟩	key为提交Bug报告id，value为相似Bug报告id集合与提交时间
behaviorCount	int	失信行为数量

为了方便记录众包工人的失信行为，系统设计SuspiciousBehavior、BehaviorHelper和BehaviorSimilarHelper三个实体类，三个类之间的关系如图 3.23所示。其中BehaviorHelper类为帮助类，存在属性bugId 表示Bug报告id，属性time表示时间。BehaviorSimilarHelper类也为帮助类，存在属性time表示时间，属

性similarList 表示Bug报告id集合。其中SuspiciousBehavior类属性较多，本文以表 3.15介绍其详情。为了持久化存储众包工人的恶意行为，系统在MongoDB数据库设计SuspiciousBehavior集合，集合属性与实体类SuspiciousBehavior属性对应，本文不再做详细描述。

(1) 反馈模块

根据表 3.1所述的失信行为与合理行为，首先介绍Bug报告有效性反馈。Bug报告有效性检测在本文3.5.1节已经详细介绍，众包工人每次提交报告时，系统评估报告有效性，并将有效性保存到MongoDB的BugQuality集合。若自动检测报告有效，则在交互界面反馈报告有效；若自动检测报告无效，则将报告无效原因反馈，报告质量属性与反馈原因的映射关系如表 3.16所示。同时，系统将该众包工人提交的无效Bug报告id和时间记录到SuspiciousBehavior集合该用户的文档中。反馈报告有效可以鼓励众包工人，反馈无效及其原因，能警告众包工人，并提供建议修改方向，从而提高测试结果的质量。

表 3.16: 质量指标与反馈原因映射

质量指标	反馈原因
description长度小于15	描述过短
sentenceLength小于7或者大于15 imprecise大于1	表达语义不清
negative为0 behavior为0 action为0 interfaceElement为0 directive为0	缺陷描述不清
imageCount为0	无缺陷截图
imageAnnotation为0	无截图标注

测试任务不希望收集重复的Bug报告，众测系统在众包工人填写Bug报告时会推荐相似报告列表，工人可对其审核或者Fork修改。为防止恶意众包工人提交重复报告，质量控制系统将提交重复报告列为失信行为。工人提交的任何一个Bug 报告都会经过重复度计算。报告的重复度计算按照如下规则进行：首先，系统提取所有已提交的报告集合，筛选出与待检测报告page 属性一致的初步集合。然后，系统加载本文3.4.2节介绍的Word2Vec 模型，使用余弦相似度计算初步集合中报告description 与待测报告description的相似性，根据预先确定的报告相似度阈值*similar_threshold*，相似度大于该阈值的报告即为与待测报告相似的报告。若没有任何已提交的报告与待测报告相似，系统交互界面反馈鼓励众包

工人，否则提示存在Bug 报告与提交的报告相似，警告众包工人，同时记录提交报告id、检测时间和相似报告id 列表到数据库中。

众测系统提供Fork操作，以减少众包工人提交重复的Bug报告，同时使众包工人对Bug报告做出有益修改从而提高Bug报告质量。恶意众包工人可能随意Fork他人报告而不做任何有益修改，因此质量控制系统将Fork后提交无信息增益的Bug 报告列为失信行为。众包工人Fork提交的报告为子报告，系统首先按照3.4.2节所述的子报告信息增益评估方法计算子报告的信息增益InfoGain，并检查InfoGain的sentences和images属性是否存在值。若sentences存在值，即说明子报告添加了有益描述；若images存在值，即说明子报告添加了不同截图；若都不存在值，即说明子报告未做任何有益修改，众包工人行为失信。若检测出众包工人行为未失信，系统交互界面反馈鼓励工人，否则警告其失信行为，并将子报告id、父报告id 和检测时间记录到数据库中。

众测系统提供点赞点踩的审核功能，使众包工人在测试过程中可对他人Bug 报告做出审核，帮助众测系统管理者识别有效报告。恶意众包工人可能随意点赞或点踩，提交的无效审核数据将干扰管理者审核，因此众测系统将恶意点赞和恶意点踩列为失信行为。众包工人每次点赞和点踩时，系统从数据库查询被审核报告的有效性，并以此作为判定依据。若众包工人点赞有效报告或者点踩无效报告，系统反馈并鼓励众包工人的正向行为；若众包工人点赞无效报告或者点踩有效报告，系统反馈并警告众包工人的失信行为，并将审核的报告id、检测时间记录到数据库中。

(2) 监控模块

为了方便众测系统管理者实时查看测试任务中众包工人的概况，更早的发现恶意众包工人，质量控制系统设计了监控模块，监控模块主要有众包工人失信行为和协作关系监控。

失信行为监控展示众包工人的失信行为，由于反馈模块已经检测和记录了众包工人的失信行为到MongoDB的SuspiciousBehavior集合，因此失信行为监控只需获取特定任务的众包工人列表和失信行为列表即可。质量控制系统不提供用户服务，因此需要调用M平台的服务来获取用户的详细信息。首先，系统根据任务caseTakeId查询SuspiciousBehavior类的实例列表，并根据类中behaviorCount 属性排序列表。然后，系统通过SuspiciousBehavior类中的workerId调用M平台后端服务获取用户列表。最终，服务端将SuspiciousBehavior实例列表和Worker实例列表返回到系统前端，供前端展示。前端使用Angular2构建worker-monitor组件，该组件负责众包工人失信行为的展示。

管理者在前端查看失信行为列表时，可查看某个众包工人的失信行

为详情，在失信行为详情中能查看每条失信行为对应的Bug报告详情，以此判断失信行为是否真实有效。若众包工人的失信行为过多，超过了阈值*suspicious_threshold*，系统判定此众包工人为恶意众包工人，管理者可通过M平台终止该众包工人参与测试任务的资格，减少其对Bug报告数据集的负面影响。

众包工人协作监控提供众包工人点赞关系、点踩关系和Fork关系查看。管理者通过查看并参考协作关系和协作详情，判定众包工人的能力等级，识别恶意众包工人和恶意众包团伙。

众包工人之间的点赞和点踩关系通过BugMirror类记录，表3.5已经详细描述BugMirror 类。首先，系统根据测试任务caseTakeId获取该任务下的BugMirror实例集，然后，根据每个BugMirror中的reportId统计参与任务的Report 实例集nodes，根据BugMirror实例集中的reportId、good和bad字段统计Report实例之间的点赞和点踩关系links。得到nodes和links之后，根据数据库中的StuInfo集合找到reportId与workerId的对应关系，从而将nodes和links中的reportId替换为workerId 。最后，使用M平台提供的数据服务将Worker 实例信息补充完整，即可得到完整的nodes和links，并将数据返回前端可视化展示。前端使用Angular2构建worker-analysis 组件，并以Echarts 中的graph图可视化众包工人的点赞和点踩关系。点击关系图上的众包工人节点时，系统展示该工人被点赞和点踩的总数，以及该工人的所有报告列表，每个报告的有效性、点赞数、点踩数、Fork数和报告详情。点击关系图上的连接边时，系统展示A众包工人点赞或者点踩B众包工人的具体Bug 报告详情。

众包工人之间的Fork关系通过BugHistory类记录，表3.6已经详细描述BugHistory 类详情。首先，系统根据caseTakeId获取该测试任务下的BugMirror集合，并根据BugMirror的id 获取BugHistory列表，根据BugHistory 筛选出parent和children 不全为空的BugMirror列表，即筛选出存在Fork关系的BugMirror列表。然后，将Bug报告之间的Fork 关系转换为Report 之间的Fork 关系，统计存在Fork关系Report 实例集nodes，以及它们之间的Fork关系links。最后，根据数据库的Stu-Info集合与外部提供的用户服务接口，将nodes和links中的reportId转换为worker-Id，并补充Worker实例信息，即可生成最终的nodes和links。将数据返回给前端组件worker-analysis，同样使用Echarts的graph图构建众包工人之间的Fork关系。点击关系图上的众包工人节点时，系统展示该工人被Fork的总数，以及该工人的所有报告列表。点击关系图上的边时，系统展示A众包工人Fork B众包工人的具体信息，包括父报告和子报告详情等。

(3) 核心类设计

图 3.24所示为反馈和监控模块核心类图。反馈和监控模块的所有前端请求都由CheckAndMonitorController类接收并处理，CheckAndMonitorService为该模块的核心业务处理类，该类通过BugDao、CheckAndMonitorDao、TaskDao和WorkerDao等类与数据库交互，完成数据查询和存储工作。BugInfoGainService为该模块提供获取报告信息增益的能力。

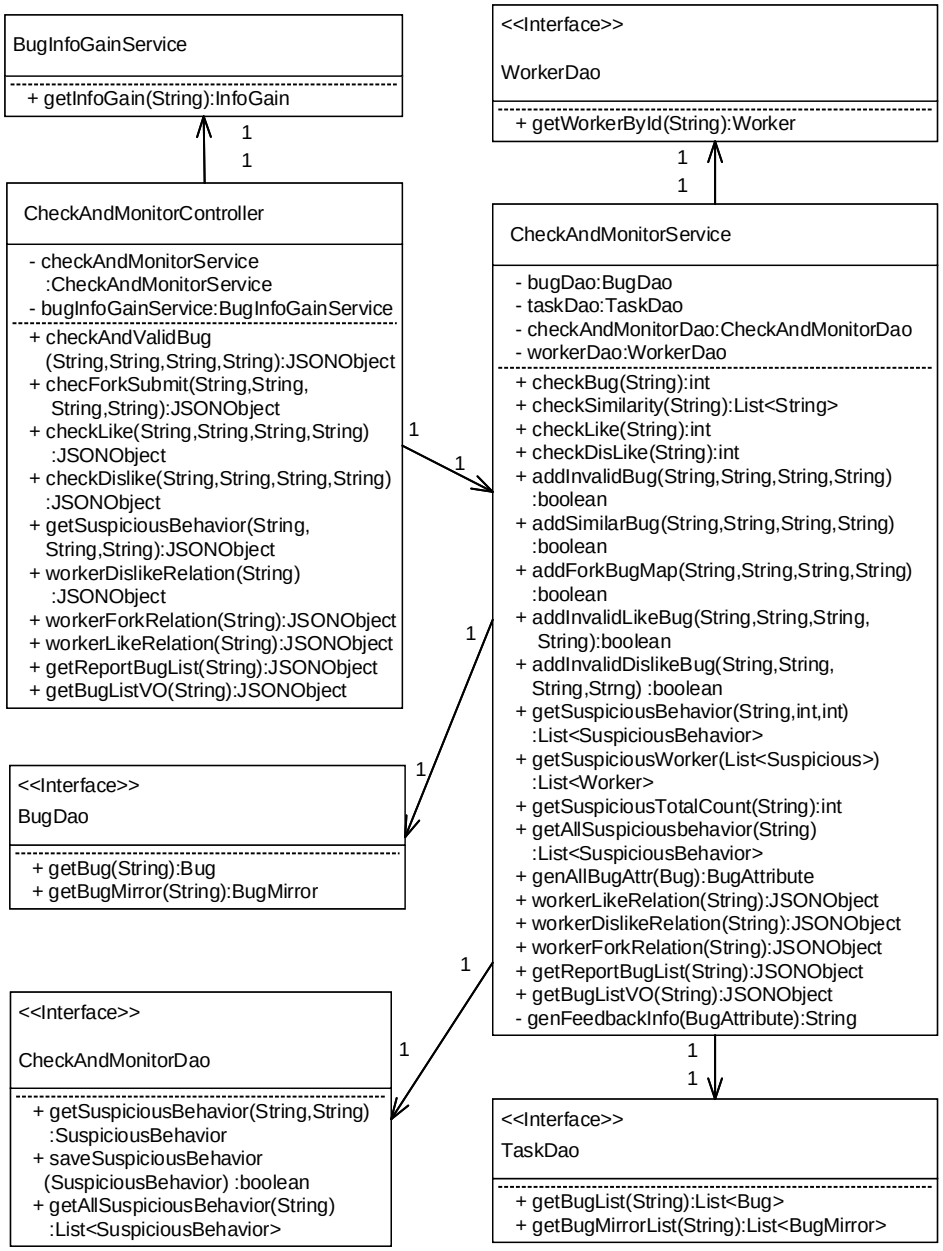

图 3.24: 反馈和监控模块核心类图

3.5.4　Bug报告审核模块

Bug报告审核模块供众测系统管理者使用，管理者在该模块可查看特定测试任务下的所有Bug报告，包括单一状报告和树状报告，查看报告的基础属性、有效性和自动评分等，并依据报告信息对报告进行审核。该模块主要包括单一状报告审核和报告树审核。

(1) 单一状报告审核

众包工人未经Fork操作直接提交的报告将呈单一状形式，管理者可对单一状报告进行审核。首先，管理者能够查看单一状报告的列表，列表中展示报告的题目以及审核完成情况。可按照时间顺序展示所有报告的列表，也可按照报告page属性筛选出特定页面的报告列表。列表须分页展示，管理者可查看任意页的列表。其次，管理者可查看列表中单个报告的详情，详情模块须展示报告的基础属性，以及报告的有效性和自动得分。对于报告的截图，系统须展示截图的原图以及众包工人所做的标注信息。最后，根据单个报告的详细信息，管理者可审核报告，审核以评分的方式进行，管理者可对报告人工评定分数，分数范围为0-10分，分数的高低代表管理者对报告质量等级的评定。

(2) 报告树审核

众包工人经Fork操作提交的报告将形成报告树，管理者可对报告树进行审核。首先，管理者可查看报告树列表，列表以报告树根节点报告的题目作为整棵树的摘要展示，同时显示该树的高度、宽度、结点个数和审核完成情况。可按照时间顺序展示所有的报告树列表，也可按照报告树所在页面筛选特定页面的报告列表。列表也须分页展示，管理者可查看任意页的列表。其次，管理者可查看列表中单个报告树的树结构。为了方便管理者查看，系统使用Echarts可视化树结构为树结构图，图中对已审核节点和未审核节点做出可视化区分，方便管理者完成审核任务。然后，管理者可查看树中任意节点的报告详情。对于根节点报告，展示模块展示的信息与单一状报告相同。对于非根节点报告，展示模块除了展示报告的基本属性、有效性、自动评分等信息之外，还要展示其相对于父报告的增益信息。增益信息包括：点赞点踩数的变化，描述中增添的句子增益，截图中增添的不同截图。增益信息的展示帮助管理者识别子报告信息增加的部分，方便管理者对子报告合理审核。最后，管理者可根据报告详情，对报告进行审核，审核的方式与单一状报告审核方式相同。

(3) 前端设计

图 3.25所示为Bug报告审核模块前端组件图。前端使用Angular2框架构建，分为Model、Component和Service层，Model层为实体类层，定义前端使用的实

体类结构；Component为组件层，实现前端页面的组件化和具体展示；Service层为前端服务层，封装前端的请求接口，与服务端进行交互。

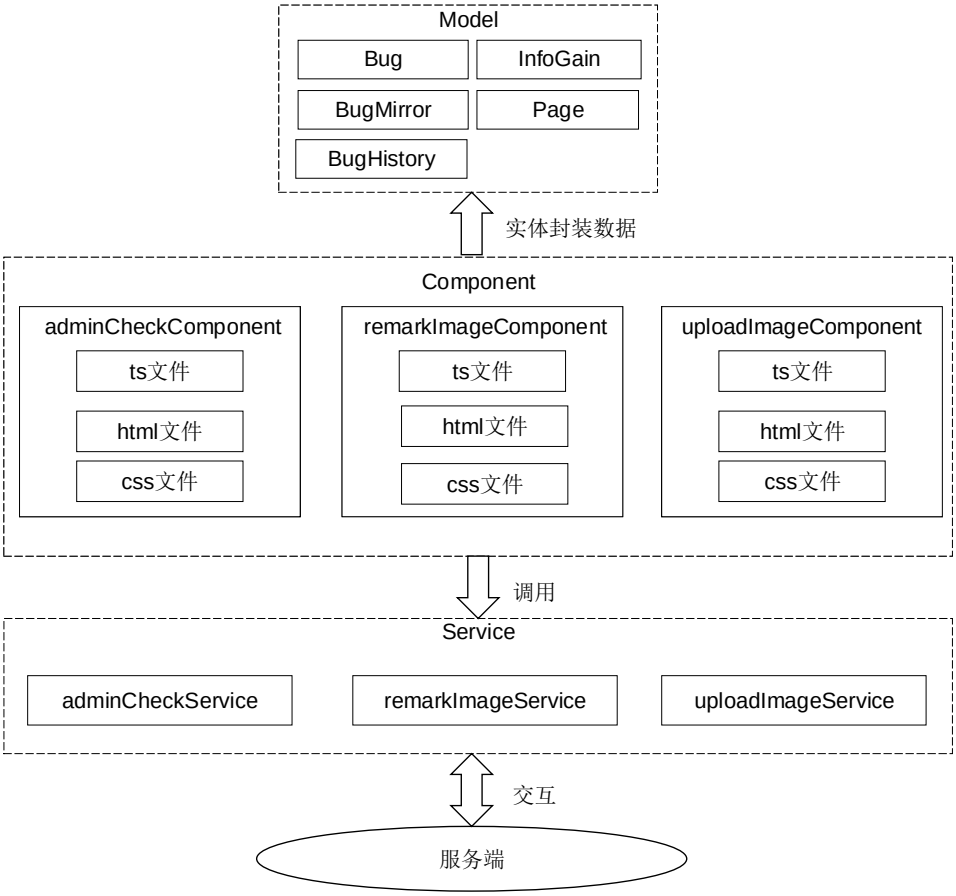

图 3.25: Bug报告审核模块前端组件图

Model层包括Bug、BugMirror、BugHistory、InfoGain和Page等实体类，前四个实体类与服务端实体类结构相似，Page类为前端分页使用的实体类。Page类主要包括以下属性：previous、next、currentPage、totalPage 和pageSize等，其中previous表示前一页码，next表示后一页码，currentPage表示当前页码，totalPage表示总页码，pageSize表示每页显示的列表数目。前端组件通过持有Page类实例完成分页展示及其相关功能。

Component层由adminCheckComponent、remarkImageComponent和uploadImageComponent组成，其中adminCheckComponent为审核模块前端主要组件，引用另外两个组件，是其父组件。uploadImageComponent组件提供展示报告的截图缩略图列表和滚动查看等功能，remarkImageComponent提供展示截图原图和标注信息的功能。每个Component都由三部分组成：ts文件、html文件和css文

件。ts 文件是组件的控制层，负责响应用户事件、前端请求的发出和接收；html文件为控件的展示页面；css文件为html文件的DOM属性设置文件。

Service层有adminCheckService、 remarkImageService 和uploadImageService，分别为对应的组件提供与服务端交互的接口，供组件中的ts文件调用。

(4) 核心类设计

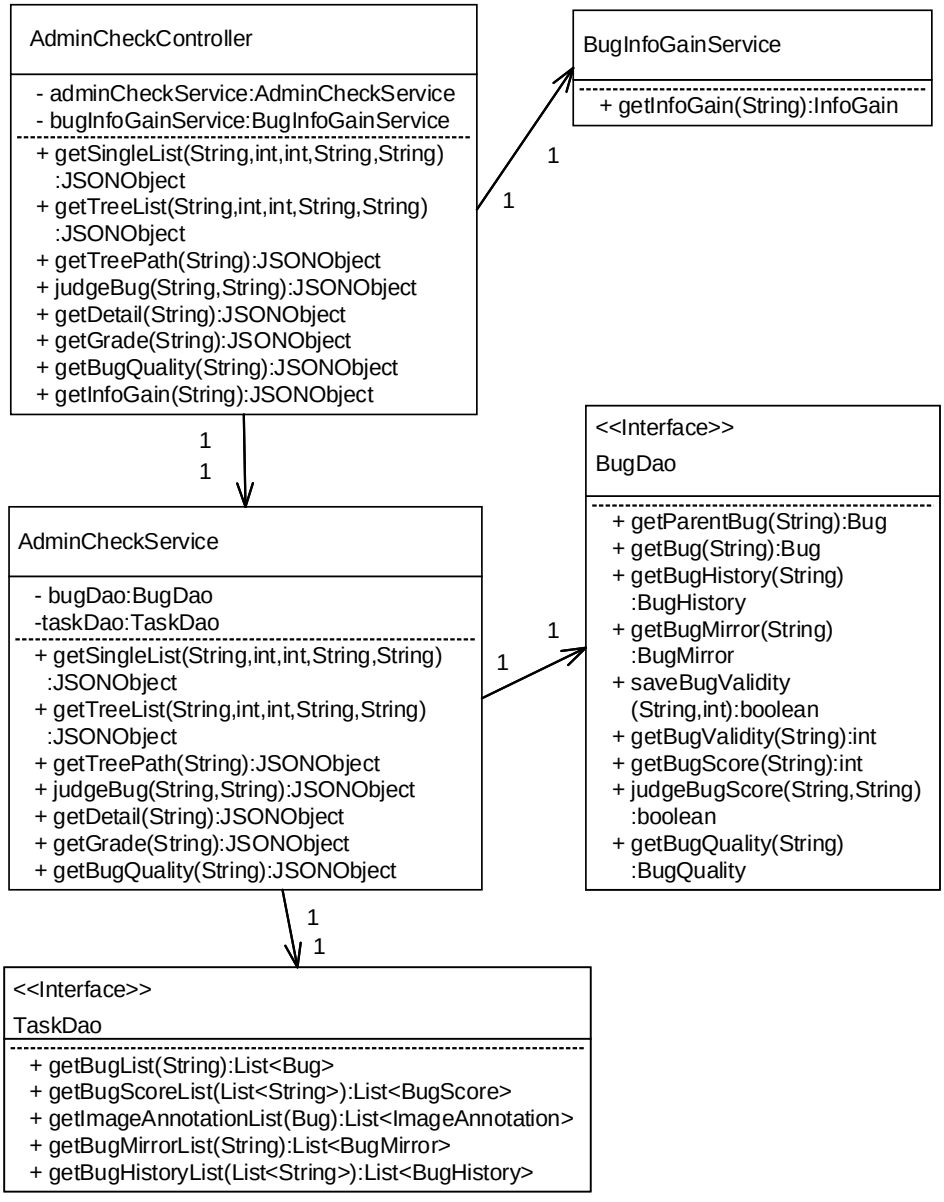

图 3.26: Bug报告审核模块核心类图

图3.26为Bug报告审核模块核心类图。审核模块的所有前端请求都由Admin-CheckController类接收并处理，AdminCheckService为该模块的主要业务处理类，

BugInfoGainService类提供获取报告信息增益的能力。BugDao和TaskDao提供与数据库交互的能力，帮助该模块完成数据查询和存储。

3.6　本章小结

本章首先使用交互图概述了质量控制系统的整体业务逻辑，并提出了系统的四个模块：Bug报告有效性检测、Bug报告自动评估、反馈和监控、Bug 报告审核。其次对系统进行需求分析，使用用例图对四个模块进行功能性需求分析，并对整个系统进行非功能性需求分析。然后以架构图的方式介绍了系统的总体设计，包括架构设计和4+1视图。接着使用类图和表格详述的方式介绍了系统实体类与数据库设计。最后以图表和文字说明的方式详细介绍了四个模块的具体设计，为系统实现做准备。

第四章　质量控制系统的实现

4.1　Bug报告有效性检测模块

4.1.1　报告有效性检测顺序图

图 4.1所示为报告有效性检测顺序图，详细展示了各个业务类之间的交互，以及检测报告有效性的具体处理过程。

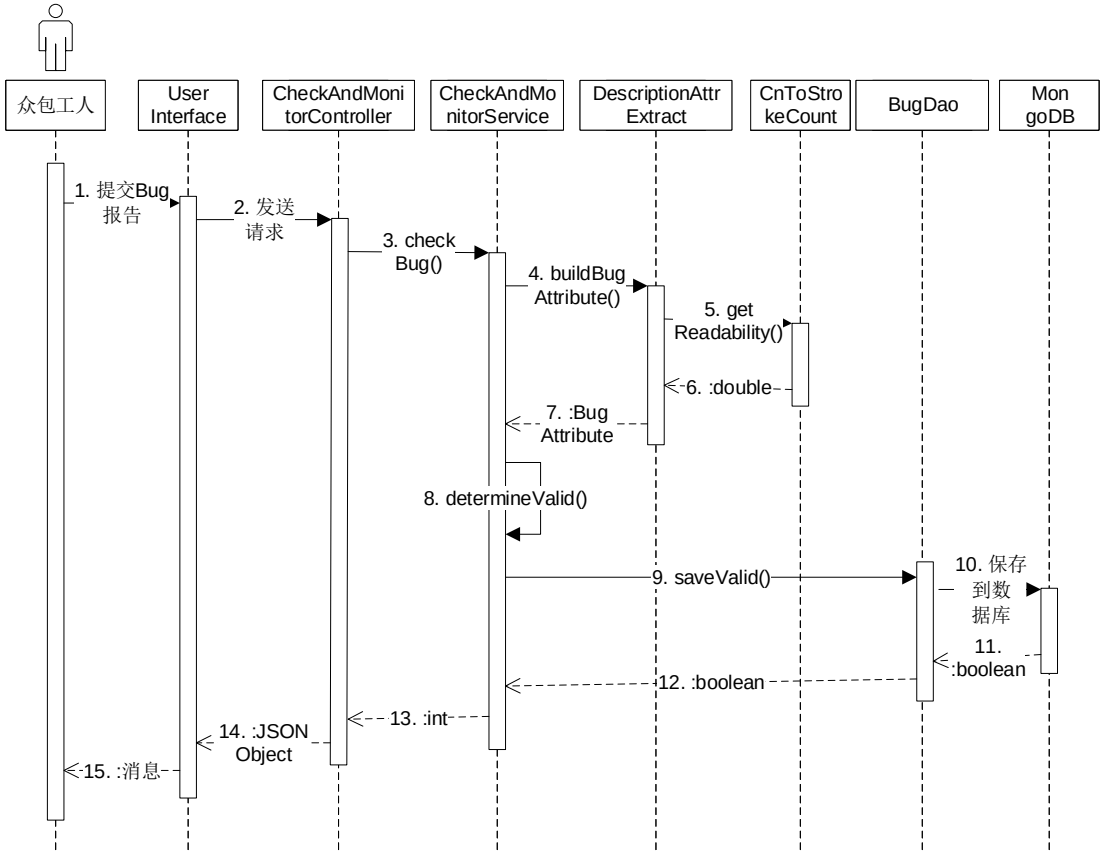

图 4.1: 报告有效性检测顺序图

众包工人提交报告之后，前端发送请求给CheckAndMonitorController处理，Controller调用CheckAndMonitorService的checkBug()方法完成报告有效性检测。Service首先调用DescriptionAttrExtract的buildBugAttribute()方法获取报告质量属性，DescriptionAttrExtract调用CnToStrokeCount类的getReadability()方法获取报告文本可读性。然后，Service调用自身的determineValid()方法检测报告有效性。

获取报告有效性之后，Service调用BugDao的saveValid()方法保存该报告有效性到MongoDB中。最后，Controller返回检测结果到前端界面，界面提示报告有效或者无效。

4.1.2　代码与结果

图 4.2所示为DescriptionAttrExtract类的buildBugAttribute()方法核心代码图，包括系统对文本长度、可读性、句子平均长度和不精确词等指标的统计过程。根据表 3.14确定的各指标合理范围，统计处于合理范围的指标数量。通过实验验证和询问专家，系统确定当*sum_threshold*超过8 时，即认为报告有效，否则认为报告无效。有效性检测方法在M平台2018年全国大学生软件测试大赛数据集上做了验证，在途牛APP和途牛网两个测试任务上，以人工有效性检测结果为标准，使用准确率和召回率对自动检测结果做了验证，结果如表 4.1所示。

```
public static BugAttribute buildBugAttribute(String desc, BugAttribute bugAttri){
    int sentences = 1;
    int imprecise = 0;

    List<String> desKeywordList = getSegment(desc);
    String[] desKeywords = (String[])desKeywordList.toArray(new
                           String[desKeywordList.size()]);
    for (int j = 0; j < desc.length(); j++) {
        for (int k = 0; k < TOKEN.length(); k++)
            if (desc.charAt(j) == TOKEN.charAt(k))
                sentences++;
    }
    for (int j = 0; j < IMPR.length; j++) {
        for (int k = 0; k < desKeywords.length; k++)
            if (desKeywords[k].equals(IMPR[j]))
                imprecise++;
    }

    bugAttri.setSentenceAvergeLenght((double) desc.length() / sentences);
    bugAttri.setDescription(desc.length());
    bugAttri.setReadability(CnToStrokeCount.getReadability(desc));
    bugAttri.setImprecise(imprecise);
    //省略部分代码
    return bugAttri;
}
```

图 4.2: buildBugAttribute()方法核心代码图

通过表 4.1可知，系统的有效性检测结果平均准确率为79.6%，平均召回率为92.4%，检测的准确率较高，召回率很高。准确率没有召回率高的原因在于，本系统的有效性检测是基于报告文本描述、截图和审核等质量指标，无法确定报告提交的缺陷真实性。质量指标很好的报告可能描述的是虚假缺陷，对于该类报告，人工审核必定判定为无效，但本检测系统依旧判定为有效，因此准确

率不会非常高。召回率很高的原因在于，人工判定为真实有效的报告，其质量指标必定不会很差，检测系统大概率会判定该类报告有效。

表 4.1: 报告有效性检测结果

测试任务	报告总数	人工检测有效数目	自动检测有效数目	同时检测有效数目	准确率	召回率
途牛APP	201	152	182	141	77.5%	92.7%
途牛网	174	125	141	115	81.6%	92.0%

4.2 Bug报告自动评估模块

4.2.1 报告信息增益评估实现

报告信息增益评估由BugInfoGainService类的getInfoGain()方法实现，其具体实现过程描述如下。

首先，Service通过调用DescriptionInfoGain类的getDescriptionInfoGain() 方法获取子报告描述上的增益。DocVectorModelSingleton类可训练和加载HanLP中的Word2Vec模型DocvectorModel，并提供DocvectorModel类的单例，使用单例模式可减少内存占用并提高响应速度。DescriptionInfoGain获取DocvectorModel单例，并使用其similarity()方法计算描述中句子的相似度，该方法原理为文本的余弦相似度，本系统直接调用该方法，不必重新实现余弦相似度计算。

其次，Service通过调用ImageInfoGain的getDiffImage()方法获取增益的图片。ImageInfoGain首先将报告的截图下载到本地，并调用ImageSimilarityCal 类中的getImageSimilarity() 方法计算图片相似度，根据相似度阈值判定子报告增加的截图，并将其返回给BugInfoGainService。ImageSimilarityCal类的相似度计算通过Lire实现，首先使用Lire获取图片的CEDD 或者FCTH特征，生成图片的GlobalFeature类，使用该类的getDistance()方法计算两张图片的相似度。

最后，Service获取协作增益。报告的协作增益为点赞数和点踩数的变化，Service通过调用BugDao获取父子报告的BugMirror实体，根据BugMirror中bad和good属性的确定子报告协作增益信息。

经过多次实验验证，本系统确定的文本相似度阈值*document_threshold*为0.77，确定的图片相似度阈值*image_threshold*为0.8，此时能较好评估子报告句子和截图的信息增益。表 4.2展示了子报告信息增益示例。

表 4.2: 子报告信息增益示例

类型	属性	详情
父报告	描述	1.推荐目的地查看更多时，可能造成APP 停止运行或崩溃2.推荐目的地查看更多时，可能造成APP需要加载时间较长
	截图	1542415944891_Screenshot_2018-11-17-08-33-58.png
	点赞数	6
	点踩数	0
子报告	描述	每次重新打开应用进入搜索页面，点击推荐目的地的查看更多，一定出现以下情况的一种：1.APP崩溃，停止运行2.页面卡顿，长时间空白后才进入到正确的查看更多页面
	截图	1542423921135_Screenshot_2018-11-17-10-58-11.png 1542423925971_Screenshot_2018-11-17-10-58-33.png
	点赞数	2
	点踩数	0
信息增益	描述	每次重新打开应用进入搜索页面，一定出现以下情况的一种
	截图	1542423925971_Screenshot_2018-11-17-10-58-33.png
	点赞数	减少4
	点踩数	无变化

4.2.2　单一状报告自动评分实现

图 4.3所示为单一状报告自动评分顺序图，展示了业务类之间的交互，以及自动评分的具体过程。

管理者点击单一状自动评分之后，前端发送请求给BugAutoScoreController处理，BugAutoScoreController调用BugAutoScoreService的noParentAutoScore()方法完成单一状报告自动评分。BugAutoScoreService先调用TaskDao的getBugList()获取该测试任务下的Bug列表，其次调用CheckAndMonitor的genAllBugAttr()获取每个报告的质量属性BugAttribute，然后调用自身的handleNoParentAutoScore()方法自动评分。BugAutoScoreService调用自身的greyAssociation()方法计算各指标与标准报告的关联度，调用自身entropyWeight()方法使用熵权法计算各指标权重，之后通过CalcuGreyAssociationSum()和genStandardScoreAndSort()方法计算每个报告总关联度并标准化为报告分数。生成分数之后，BugScoreService调用BugAutoScoreDao存储报告分数到MongoDB数据库中。最后，BugAutoScore-Controller返回业务处理结果JSONObject到前端界面，前端根据返回结果提示管理者自动评分成功或者失败。

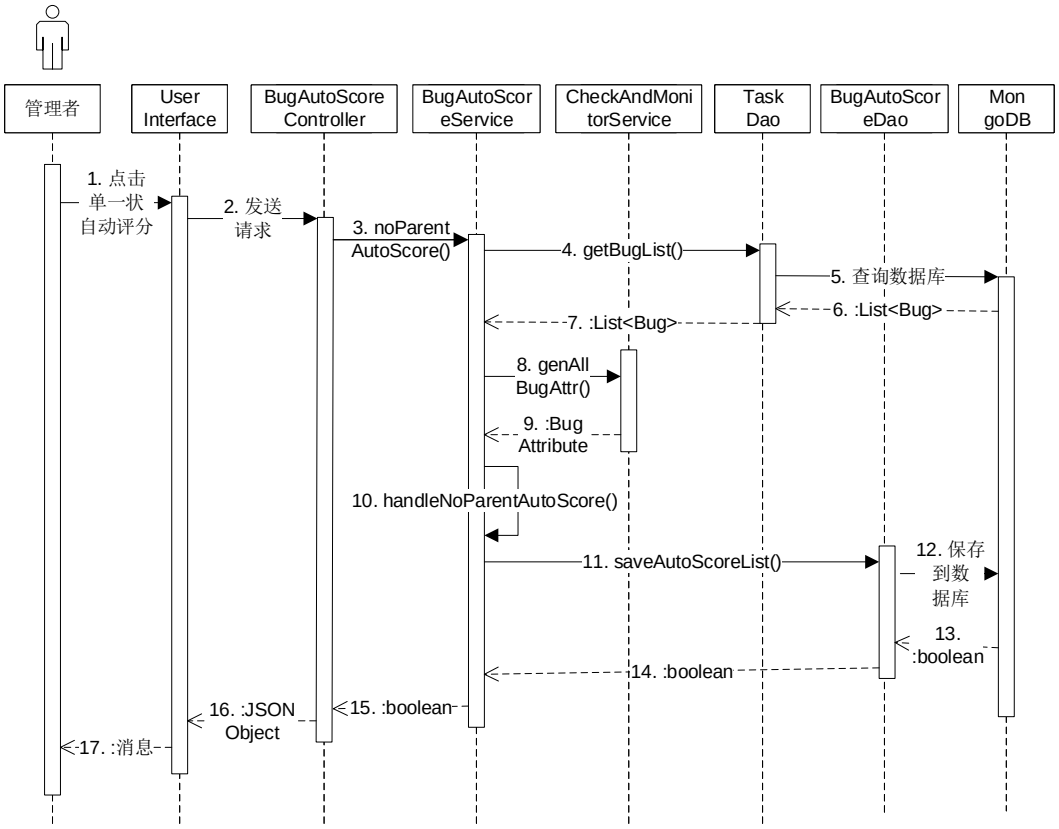

图 4.3: 单一状报告自动评分顺序图

4.2.3　树状报告自动评分实现

树状报告自动评分的顺序图与单一状报告自动评分类似，此处不再赘述，只详细描述其实现流程。BugAutoScoreService负责提供树状报告自动评分服务，调用自身handleHasParentAutoScore()方法实现。handleHasParentAutoScore()首先获取待评分子报告的父报告评分$s1$，然后调用calHasParentBugScore()方法计算子报告的评分。

计算子报告评分时，calHasParentBugScore()首先通过BugInfoGainService获取子报告的信息增益InfoGain。然后，从InfoGain的增益句子中提取形态学、词法和语义指标值。若存在任意一项指标，即在父报告的基础上增加0.5分。同时查看InfoGain中是否存在增益截图，若存在一张截图则增加1分。最后，查看协作指标点赞和点踩数的变化情况。若子报告点赞数增加，则增加1分，否则减少1分；若子报告点踩数减少，则增加1分，否则减少1分。经过该过程的分数变化，子报告评分可能小于0分或者大于10分，小于0的映射为0分，大于10的映射为10分，则可获得子报告的最终评分$s2$。

```
private BugAttribute calHasParentBugScore( Bug bug, int parentScore){
        double initScore = parentScore;
        double baseScore = 0.5;

        InfoGain infoGain = bugInfoGainService.getInfoGain(bug.getId());
        bug.setDescription(concatSentences(infoGain.getSentences()));
        BugAttribute bugAttr = new BugAttribute(bug.getId());
        DescriptionAttrExtract.buildBugAttribute(bug.getDescription(), bugAttr);

        if(infoGain.getImages().size()>0){
            initScore += infoGain.getImages().size();
        }
        if(infoGain.getDissCount()>0){
            initScore -= 2;
        }
        else if(infoGain.getDissCount()<0){
            initScore += 1;
        }
        if(infoGain.getLikeCount()>0){
            initScore += 1;
        }
        else if(infoGain.getLikeCount()<0){
            initScore -= 1;
        }
        if(bugAttr.getDirective()>0){
            initScore += baseScore;
        }
        //省略部分代码
        initScore = initScore>10?10:initScore;
        initScore = initScore<0?0:initScore;
        bugAttr.setScore(Math.ceil(initScore));
        return bugAttr;
}
```

图 4.4: calHasParentBugScore()方法核心代码图

得到$s2$之后，handleHasParentAutoScore()调用BugAutoScoreDao的方法将评分存储到数据库。图 4.4所示为calHasParentBugScore()方法的核心代码图。

4.3　反馈和监控模块

4.3.1　反馈实现

(1) Bug报告有效性反馈

众包工人提交每一个Bug报告时都发送有效性检测请求到CheckAndMonitorController，它调用CheckAndMonitorService的checkBug()方法检测报告有效性，有效性检测在本文4.1节已经实现。若检测报告有效，即反馈报告有效，对众包工人做出鼓励。若报告无效，调用自身私有的genFeedbackInfo()方法获取无效原因，反馈众包工人报告无效和无效原因，同时Controller调用CheckAndMonitorService的addInvalidBug()方法记录此失信行为到数据库，addInvalidBug()方法调用CheckAndMonitorDao的saveSuspiciousBehavior()方法保存失信行为到数据库。图 4.5为Bug报告有效性反馈界面图，界面图展示当前提交的Bug报告很可能无效，警告工人慎重提交，同时反馈了可能的无效原因。

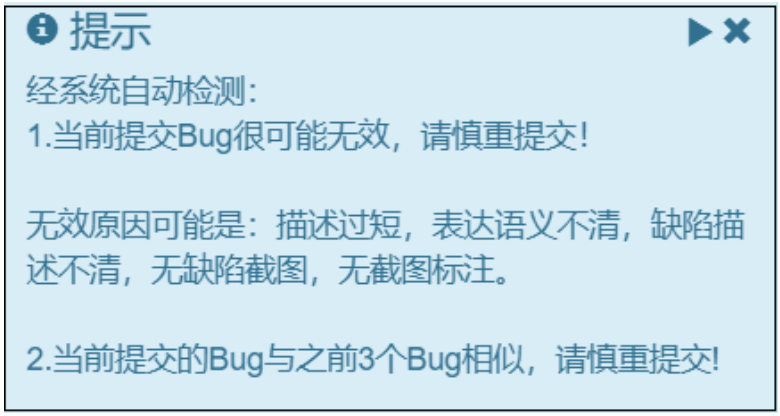

图 4.5: Bug报告有效性反馈界面图

(2) 重复Bug报告反馈

众包工人提交Bug报告时都发送请求到CheckAndMonitorController检查报告重复性，Controller调用CheckAndMonitorService的checkSimilarity()方法检测报告重复性。首先，系统获取已经提交的报告列表，并根据工人提交报告的page属性筛选出与提交报告在同一页面的报告。然后使用HanLP的DocVectorModel类的similarity()方法计算待检测报告描述与筛选报告描述的相似度，若相似度大于*similar_threshold*则认定两个报告相似。经过多次实验验证，本系统选取*similar_threshold*值为0.8，此时可较好的判定重复报告。若不存在与待检测报告重复的报告，系统反馈众包工人最先发现此缺陷；若存在与待测报告重复的报告，系统反馈众包工人存在重复报告，并警告此为失信行为。同样，Controller 调用CheckAndMonitorService的addSimilarBug()方法将失信行为保存到数据库。

(3) Fork提交反馈

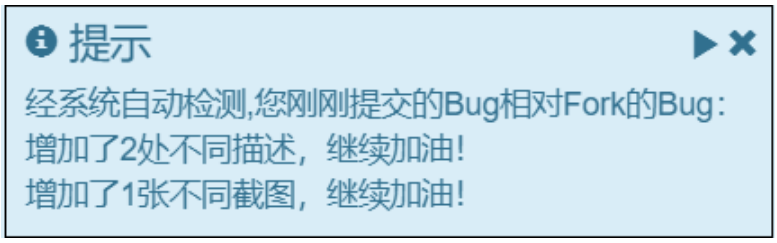

图 4.6: Fork提交反馈界面图

众包工人Fork他人报告提交子报告时都将提交请求到CheckAndMonitorController 检查信息增益情况，Controller中checkForkSubmit()方法处理该请求。它调用BugInfoGainService 的getInfoGain()方法获取子报告信息增益。若子报告添加

不同的描述或截图增益，系统反馈众包工人子报告信息增益情况；若无描述和截图增益，系统反馈众包工人子报告无增益，警告其此为失信行为。同样，Controller调用CheckAndMonitorService的addForkBugMap()方法将失信行为保存到数据库。图 4.6为Fork提交反馈界面图。

(4) 审核反馈

众包工人点赞和点踩审核时都将发送请求给CheckAndMonitorController检查审核有效性。Controller中的checkLike()方法检测点赞有效性，Controller调用CheckAndMonitorService的checkLike()方法实现业务逻辑，该方法首先查询点赞的报告有效性，若点赞有效报告则众包工人的点赞合理，系统反馈众包工人点赞合理；若点赞无效报告则众包工人点赞不合理，系统反馈众包工人存在恶意点赞行为，警告其此为失信行为。同样，调用CheckAndMonitorService的addInvalidBug()方法记录该失信行为到数据库。点踩行为的检测反馈与点赞类似，只是点踩无效报告为合理行为，点踩有效报告为失信行为，业务处理过程与点赞过程类似，此处不再赘述。

4.3.2　监控实现

(1) 失信行为监控

图 4.7: 失信行为监控界面图

失信行为监控模块展示所有众包工人的失信行为列表，供管理者查看并发现恶意众包工人。管理者点击失信行为查看时，前端发送请求被CheckAndMonitorController拦截，Controller中的getSuspiciousBehavior()方法处理该请求：首先，调用CheckAndMonitorService的getSuspiciousWorker()方法获取失信众包工人列表。其次，调用其getSuspiciousBehavior()方法获取众包工人的失信行为列表，同时将列表按照失信行为数目排序。最后，调用getSuspiciousTotalCount()方

法获取该测试任务下的失信众包工人个数，供前端分页展示列表。失信行为监控前端界面由WorkerMonitorComponent组件实现，该模块通过WorkerMonitorService发送请求与服务端交互。

图 4.8: 报告详情界面图

图 4.7所示为失信行为监控界面图。界面最左侧分页展示失信众包工人列表，简要介绍该工人姓名、单位和疑似失信行为个数。同时提供分页查看列表操作，可点击首页、上一页、下一页和末页等，也可输入合理页码直接跳转。点击列表中任意一个众包工人，界面右侧将展示该工人所有失信行为详情与时间。失信行为包括提交无效Bug、提交相似Bug、恶意Fork、恶意点赞和恶意点踩等。点击失信行为详情中的报告id，界面弹出Bootstrap模态框展示报告的点赞数、点踩数、基本属性和截图等，帮助管理者判断失信行为真实性，图 4.8为报告详情界面图。

(2) 协作关系监控

众包工人协作关系监控模块展示众包工人之间点赞、点踩和Fork关系。管理者查看众包工人点赞关系时，前端发送的请求被CheckAndMonitorController的workerLikeRelation()方法拦截，该方法调用CheckAndMoniorService 的workerLikeRelation()方法完成业务功能。

图 4.9所示为众包工人点赞关系界面图。界面左侧以图的形式展示众包工人

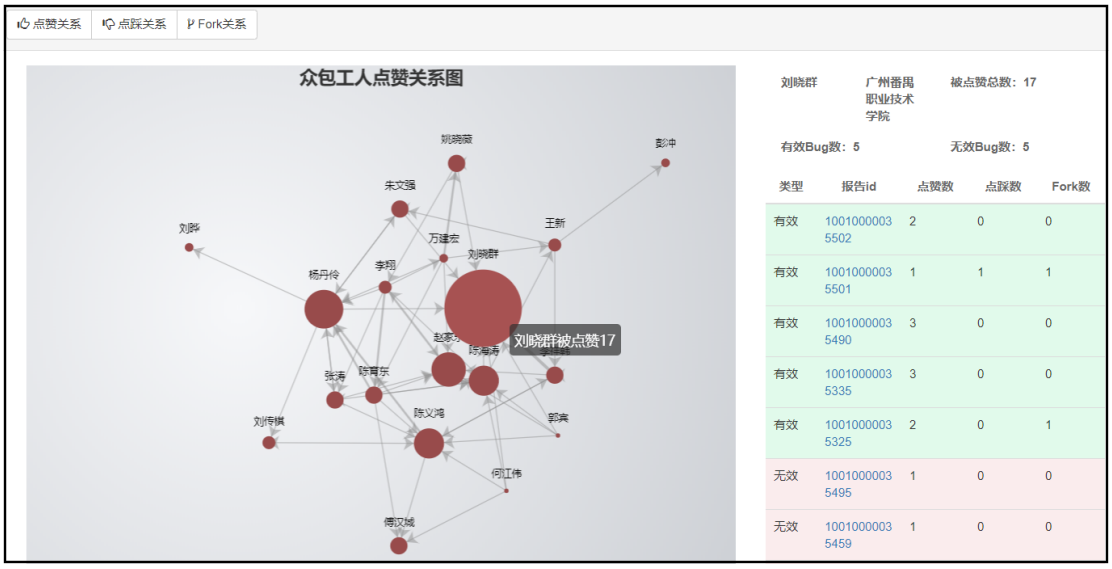

图 4.9: 众包工人点赞关系界面图

之间的点赞关系，图中点代表众包工人，点的大小代表该工人被点赞数目大小；图中边代表一个点赞关系，非箭头处工人点赞箭头指向的工人。点击图上任意点，界面右侧将展示该工人的测试详情，以列表的形式展示其提交的有效和无效报告，用绿色底色代表有效，红色底色代表无效。点击任意报告id，界面都将弹出模态框展示报告详情，帮助管理者判断报告质量，从而判断众包工人的能力。点击图上任意一条边，界面右侧都将展示众包工人A点赞众包工人B的报告列表，帮助管理者查看点赞关系详情。

管理者查看众包工人点踩关系时，前端发送的请求被CheckAndMonitorController的workerDislikeRelation()方法拦截，该方法调用CheckAndMonitorService的workerDislikeRelation()方法完成业务功能。图 4.10所示为众包工人点踩关系图，与点赞关系类似，右侧以图展示点踩关系，图中点的大小代表该工人被点踩数目大小。可点击图上点查看众包工人的报告列表，点击图上边查看众包工人点踩详情。查看点踩关系能帮助管理者发现能力较差或者恶意众包工人。

管理者查看众包工人Fork关系时，前端发送的请求被CheckAndMonitorController的getWorkerForkRelation()方法拦截，该方法调用CheckAndMonitorService的workerForkRelation()完成业务功能。图 4.11所示为众包工人Fork关系界面图。图上点代表众包工人，点的大小代表工人被Fork 的数目大小；图上边代表一个Fork 关系。点击图上点时界面右侧展示该众包工人的报告列表；点击图上边时，界面右侧展示子报告和父报告id，可点击报告id查看报告详情。该功能帮助管理者发现被Fork次数较多的众包工人，该工人很可能有较强的测试能力。

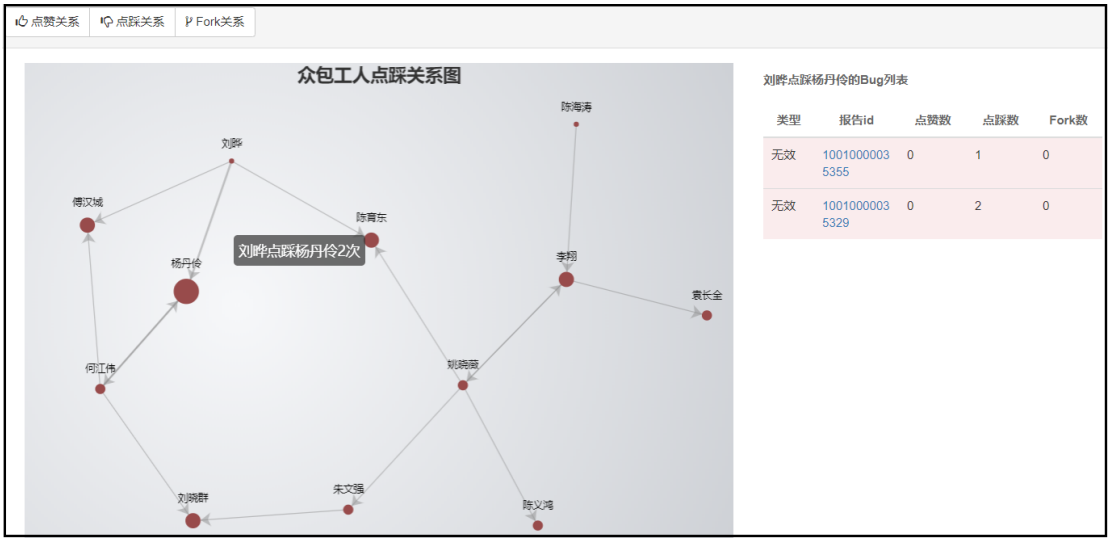

图 4.10: 众包工人点踩关系界面图

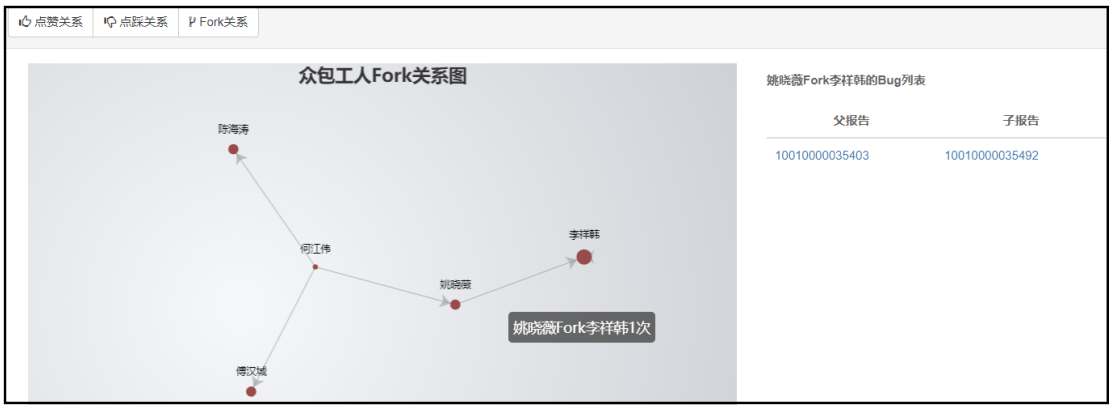

图 4.11: 众包工人Fork关系界面图

4.4　Bug报告审核模块

4.4.1　单一状报告审核实现

图 4.12为单一状报告审核界面图。点击界面左上方的单一状之后前端将发送请求到服务端，并被服务端的AdminCheckController中的getSingleList()方法拦截，该方法调用AdminCheckService中的getSingleList()获取当前页的单一状报告列表。如界面左侧所示，列表展示该报告的标题、和审核完成情况，审核完成以“√”显示，否则以“×”显示。点击列表中的任何一个报告，前端将发送请求被Controller拦截，AdminCheckService中的getDetail()、和getBugQuality()方法将被调用，并返回前端报告详情。界面最右侧展示报告详情，包括题目、页面、描述、点赞数、点踩数和自动评分等。点击界面右下侧的图片缩略图，将

图 4.12: 单一状报告审核界面图

打开报告截图和截图标注界面，如图 4.13所示，图中红色标出部分为众包工人做出的截图标注。

图 4.13: 截图及标注界面图

根据报告详细信息，管理者可审核报告，审核以评分的形式进行，可在审核界面右上侧的选择框选择0-10分作为审核结果。审核时，前端发送请求被Controller的judgeBug()方法拦截，该方法调用AdminCheckService类的judgeBug()方法完成业务逻辑。界面最上侧的选择框可供管理者筛选出特定页面的报告，“清空选择”按钮可供清除选择的页面。

4.4.2　报告树审核实现

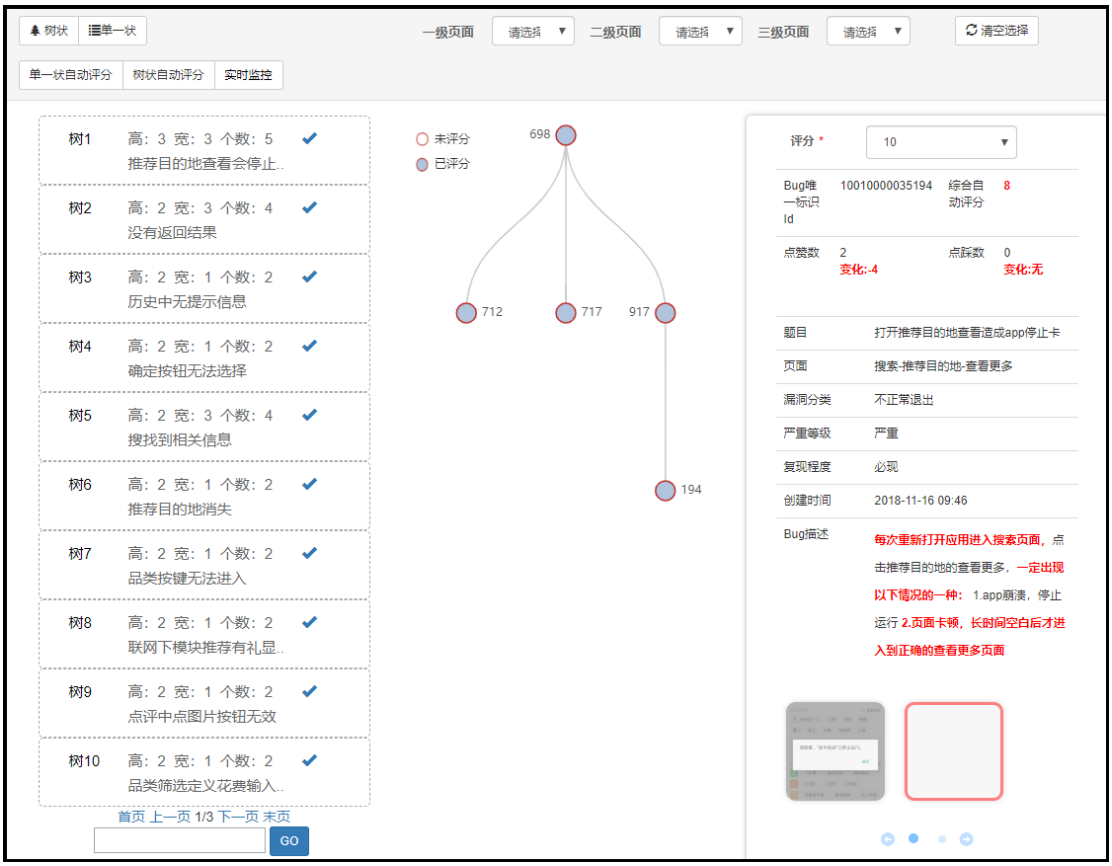

图 4.14: 报告树审核界面图

图 4.14所示为报告树审核界面图。点击界面左上方的树状之后前端将发送请求到服务端，并被服务端AdminCheckController的getTreeList()方法拦截，该方法调用AdminCheckService的getTreeList()获取当前页的报告树列表。如界面左侧所示，列表展示该报告树的高、宽、摘要和审核完成情况，审核完成以“√”显示，否则以“×”显示。点击列表中任何一个报告树，前端将发送请求被Controller拦截，AdminCheckService的getTreePath()方法将被调用，返回此报告树的结构详情。报告树的结构详情在界面中间展示，树结构使用Echarts的Tree图绘制，可清晰展示该报告树的所有报告。当点击树中某一节点时，前端发送请求获取该节点报告的详情。与单一状报告所不同的是，当点击根节点报告时，由于其没有父报告，右侧详情展示与单一状详情展示完全一致。当点击非根节点报告时，前端除了获取报告详情之外，还将发送请求调用BugInfoGainService的getInfoGain()方法，获取子报告的信息增益，并以红色字体展示信息增益，包括点赞数和点踩数的变化，文本增益和截图增益等。增

益信息的展示帮助管理者更容易辨别父子报告的不同之处，方便其审核报告。其余查看截图和标注、审核报告、筛选报告树等操作流程与单一状报告审核界面相同，此处不再赘述。

4.5　本章小结

本章主要介绍了质量控制系统的Bug报告有效性检测、Bug报告自动评估、反馈和监控、Bug报告审核四个模块的实现。根据第三章的具体设计，以顺序图、关键代码、系统界面和文字说明的方式，详述了各个模块的具体实现细节，为下一章系统测试与实验评估奠定基础。

第五章　质量控制系统的测试与实验评估

5.1　测试环境

质量控制系统设计及实现之后，本文依据系统需求进行功能和性能测试，表 5.1为测试环境详述。系统前端基于Angular2.4构建，部署在ECS（Elastic Compute Service，弹性计算服务）服务器上，前端项目依赖运行环境Node.js 8.9.0和Npm 6.4.1。系统服务端基于Spring Boot 2.0.2构建并部署在ECS服务器的Docker上，依赖环境为JDK1.8、Tomcat 8.0.23和Docker 17.09；MongoDB部署在操作系统为Debian 8的ECS服务器上，其版本为3.2.16。

本文在用户计算机上对系统进行测试，使用Chrome浏览器访问系统前端，根据设计的测试用例对各个模块进行功能测试，并使用JMeter 5.1.1对各模块重要接口进行性能测试。JMeter是Apache 开发的基于Java的压力测试工具，可用于对服务器模拟巨大的负载，测试服务器的整体性能。

表 5.1: 测试环境

组件	设备	运行程序	依赖环境
前端	ECS服务器(Ubuntu 18.04)	Angular 2.4	Node.js 8.9.0，Npm 6.4.1
服务端	ECS服务器(4G内存，50M带宽)	Spring Boot 2.0.2	JDK1.8，Tomcat 8.0.23，Docker 17.09
数据库	ECS服务器(Debian 8)	MongoDB 3.2.16	无
功能测试工具	用户计算机(Windows 10)	Chrome浏览器73.0(64 位)	无
性能测试工具	用户计算机(Windows 10)	JMeter 5.1.1	JDK1.8

5.2　功能与性能测试

5.2.1　Bug报告有效性检测模块测试

Bug报告有效性检测模块最重要的功能为报告有效性检测，因此本模块重点测试有效性检测接口。有效性检测接口首先提取Bug报告质量指标，并根据质量指标影响方式和合理范围确定每个指标的有效性，最后依据有效指标的数量判定Bug报告有效性。若Bug报告质量属性达到要求则判定为有效，未达到要求则判定无效。本文对有效性检测接口进行功能和性能测试。

(1) 功能测试

表 5.2所示为Bug报告有效性检测模块接口测试用例详述。本文设计两个测试用例分别测试该接口检测有效报告和无效报告的功能，结果表明，接口可正确检测报告有效性，系统正确反馈检测结果。

表 5.2: Bug报告有效性检测模块接口测试用例

用例编号	测试功能	测试步骤	预期结果	结果
QC-VALID-1	检测无效报告	1.参加特定测试任务，新建Bug报告 2.随意填写报告的缺陷属性和描述，不上传缺陷截图和截图标注 3.提交报告	1.检测报告无效 2.反馈报告无效原因	通过
QC-VALID-2	检测有效报告	1.参加特定测试任务 2.认真填写缺陷属性及描述 3.上传缺陷截图并标注截图 4.提交报告	检测报告有效并反馈	通过

(2) 性能测试

有效性检测接口在众包工人每次提交报告时都会被调用，需要承受高并发和大容量，因此接口必须有高性能。本文在JMeter上模拟200个众包用户在10秒之内各发送3次有效性检测请求，图 5.1所示为JMeter模拟并发的界面截图，图中Numbers of Threads属性为模拟的线程数目，Ramp-Up Period属性为模拟的时间间隔，Loop Count 属性为每个线程发送请求的次数。

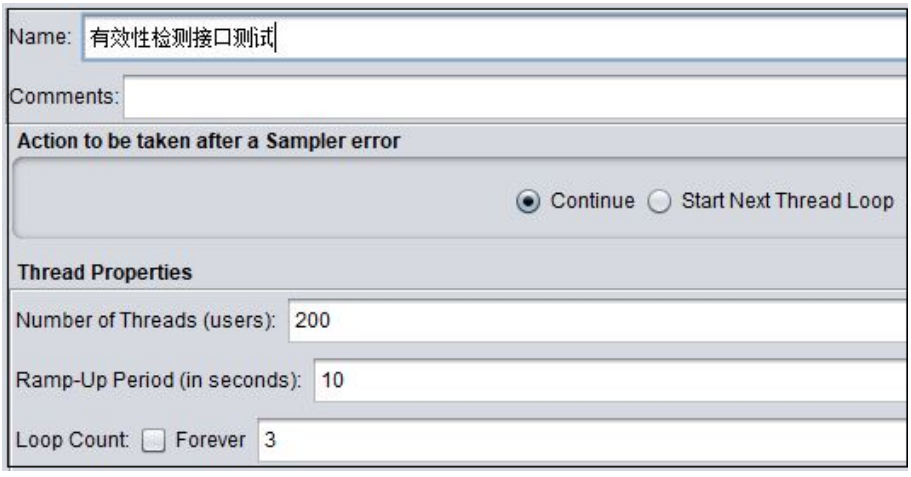

图 5.1: JMeter模拟并发界面图

接口的JMeter测试汇总图如图 5.2所示。由图可知，10秒之内的600次请求全部被正确处理，600次请求的平均处理时间为33毫秒，有99%的请求在41毫秒

内被响应，最长的响应时间为683毫秒，接口的吞吐量接近60/秒。图 5.3所示为接口的Jmeter响应时间图，可知所有请求的响应时间比较平稳，极个别请求响应时间较长。综上，有效性检测接口并发性能好，响应时间短，满足系统的性能要求。

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput
有效性检测接口	600	33	32	35	37	41	29	683	0.00%	59.8/sec
TOTAL	600	33	32	35	37	41	29	683	0.00%	59.8/sec

图 5.2: 有效性检测接口的JMeter汇总图

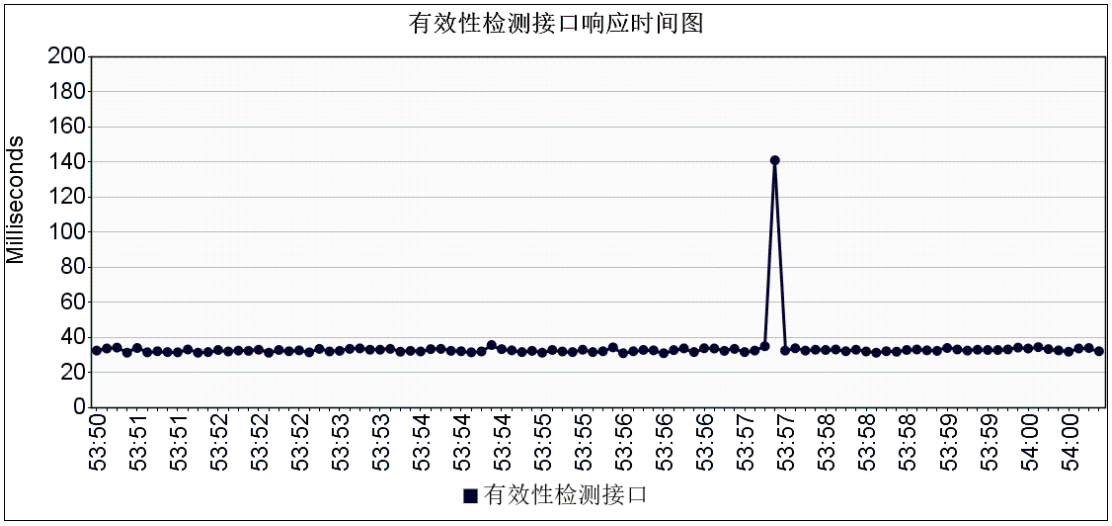

图 5.3: 有效性检测接口的JMeter响应时间图

5.2.2　Bug报告自动评估模块测试

Bug报告自动评估模块主要有计算单一状报告分数、评估子报告信息增益和计算树状报告分数三个重要功能，因此本模块重点测试单一状报告自动评分接口、信息增益自动评估接口和树状报告自动评分接口，对三个接口进行功能和性能测试。

(1) 功能测试

表 5.3所示为Bug报告自动评估模块接口测试用例详述。本文首先对存在单一状报告的任务进行单一状报告自动评分测试，系统可正常评分；又对不存在单一状报告的任务进行测试，系统提示不存在单一状报告无法评分，单一状报告自动评分接口各功能正常。同样，本文对树状报告进行了评分成功与失败的测试，树状报告自动评分接口功能正常。对于报告信息增益评估，本文分别测试存在和不存在信息增益的子报告，接口均可正确识别子报告的信息增益。

表 5.3: Bug报告自动评估模块接口测试用例

用例编号	测试功能	测试步骤	预期结果	结果
QC-EVAL-1	单一状报告自动评分成功	1.新建测试任务，在该任务中提交一些单一状Bug报告 2.点击单一状自动评分	所有已提交单一状报告均评分完成	通过
QC-EVAL-2	单一状报告自动评分失败	1.新建测试任务，在该任务中不提交任何单一状报告 2.点击单一状自动评分	系统提示不存在单一状报告，无法评分	通过
QC-EVAL-3	评估存在信息增益的子报告	1.在填写报告页面Fork一个Bug报告，并添加不同描述和截图 2.点击提交报告	系统反馈添加描述和截图增益	通过
QC-EVAL-4	评估不存在信息增益的子报告	1.在填写报告页面Fork一个Bug报告，不做任何修改提交 2.点击提交报告	系统反馈未添加任何增益	通过
QC-EVAL-5	树状报告自动评分成功	1.新建测试任务，在该任务中提交单一状报告和树状报告 2.点击单一状报告自动评分 3.点击树状报告自动评分	所有已提交树状报告均评分完成	通过
QC-EVAL-6	树状报告自动评分失败	1.新建测试任务，在该任务中提交单一状报告和树状报告 2.直接点击树状报告自动评分	系统提示需先完成单一状报告自动评分	通过
QC-EVAL-7	树状报告自动评分失败	1.新建测试任务，在该任务中提交单一状报告，但不提交任何树状报告 2.点击单一状报告自动评分 3.点击树状报告自动评分	系统提示不存在树状报告，无法评分	通过

(2) 性能测试

单一状和树状报告自动评分接口只有管理者可以调用，一般每次测试任务只需调用一次，因此对性能要求不高。信息增益评估接口在众包工人Fork提交子报告时都会被调用，因此需要较高性能，本文重点测试信息增益评估接口的性能。

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput
信息增益评估...	600	224	223	234	238	250	215	254	0.00%	56.5/sec
TOTAL	600	224	223	234	238	250	215	254	0.00%	56.5/sec

图 5.4: 信息增益评估接口的JMeter汇总图

本文在JMeter上模拟200个众包用户在10秒之内各发送3次信息增益接口请求，接口的JMeter测试汇总图如图 5.4所示。由图可知，10秒之内的600次请求全部被正确处理，600次请求的平均处理时间为224毫秒，有99%的请求在250毫

秒内被响应，最长的响应时间为254毫秒，接口的吞吐量接近57/秒。因此，信息增益评估接口并发性能好，响应时间短，满足系统的性能要求。

5.2.3　反馈和监控模块测试

反馈和监控模块包块反馈和监控两个部分，其中反馈部分的重要接口包括报告有效性检测和信息增益评估等，已在前文详细测试。监控部分包括失信行为监控和众包工人协作关系监控，本文重点对其进行测试。

(1) 功能测试

表 5.4所示为反馈和监控模块接口的功能测试用例详述。本文首先测试失信行为查看接口，设计QC-MONITOR-1测试用例通过测试。然后设计QC-MONITOR-2用例测试点赞关系查看接口，该接口正常显示点赞关系。同样，点踩和Fork关系查看接口也通过功能测试。

表 5.4: 反馈和监控模块接口测试用例

用例编号	测试功能	测试步骤	预期结果	结果
QC-MONITOR-1	失信行为查看	1.使用工人A的账号参加某测试任务 2.在该任务中恶意提交无效报告 3.在该任务中恶意Fork提交未增益报告 4.查看监控界面失信列表 5.查看该工人失信行为详情	失信工人列表中出现工人A，A的失信行为详情包括提交无效报告和恶意Fork	通过
QC-MONITOR-2	点赞关系查看	1.使用工人A的账号在某任务提交报告S 2.使用工人B的账号在相同任务中点赞报告S 3.查看点赞关系图	点赞关系图展示工人B点赞工人A，并且显示点赞的报告为S	通过
QC-MONITOR-3	点踩关系查看	1.使用工人A的账号在某任务提交报告S 2.使用工人B的账号在相同任务中点踩报告S 3.查看点踩关系图	点踩关系图展示工人B点踩工人A，并且显示点踩的报告为S	通过
QC-MONITOR-4	Fork关系查看	1.使用工人A的账号在某任务中提交报告S 2.使用工人B的账号在相同任务中Fork报告S生成报告H 3.查看Fork关系图	Fork关系图展示工人B Fork工人A，且父报告为S子报告为H	通过

(2) 性能测试

在系统未来的改进计划中，所有信息都将公开透明，众包工人将能够看到所有的失信行为以及协作关系，因此对监控模块接口的性能要求很高。本文对失信行为查看接口、三种协作关系查看接口分别进行了性能测试。

首先，本文在JMeter上模拟200个众包用户在10秒之内各发送3次失信行为查看接口请求，同样，以相同的并发条件发送协作关系查看接口请求。四个接口的JMeter测试汇总图如图 5.5所示。由图可知，四个接口10秒之内的600次请求全部被正确处理。失信行为查看接口平均响应时间为214毫秒，点赞关系查看接口平均响应时间为219毫秒，点踩关系查看接口的平均响应时间为237毫秒，Fork关系查看接口的平均响应时间为249毫秒。因此，监控模块四个接口都满足性能要求。

Label	# Sam...	Average	Median	90% Line	95% Line	99% Line	Min	Maxim...	Error %	Throughput
失信行为...	600	214	204	218	223	233	194	3655	0.00%	56.8/sec
点赞关系...	600	219	218	230	234	238	210	300	0.00%	56.6/sec
点踩关系...	600	237	235	247	249	266	228	274	0.00%	56.2/sec
Fork关系...	600	249	246	263	272	283	233	571	0.00%	56.3/sec

图 5.5: 监控模块接口的JMeter汇总图

5.2.4　Bug报告审核模块测试

审核模块主要有报告树查看接口、单一状报告查看接口和审核接口，本文对三个接口进行功能和性能测试。

(1) 功能测试

表 5.5所示为Bug报告审核模块接口测试用例详述。本文首先测试报告树查看接口，设计QC-CHECK-1测试用例通过测试。然后设计QC-CHECK-2用例测试单一状报告查看接口，该接口正常展示报告详情。最后，设计QC-CHECK-3测试审核接口，接口正常评分并记录分数到数据库。

(2) 性能测试

系统中的所有Bug报告公开透明，众包工人能够查看所有报告信息，但是不对其提供审核功能；同时系统上可以存在多个管理者并发审核报告，因此系统对该模块接口的性能要求很高。本文对报告树查看接口、单一状报告查看接口和审核接口分别进行了性能测试。

首先，本文在JMeter上模拟200个众包用户在10秒之内各发送3次报告树查看接口请求，同样，以相同的并发条件发送单一状报告查看接口和审核接口请求。三个接口的JMeter测试汇总图如图 5.6 所示。由图可知，三个接口10秒之内的600次请求全部被正确处理。报告树查看接口平均响应时间为217毫秒，单一状报告查看接口平均响应时间为195毫秒，审核接口的平均响应时间为32毫秒。因此，三个接口都满足性能要求。

表 5.5: Bug报告审核模块接口测试用例

用例编号	测试功能	测试步骤	预期结果	结果
QC-CHECK-1	查看报告树中的报告	1.新建测试任务，提交树状报告 2.在审核页面查看报告树列表 3.点击列表中特定项 4.点击树状图中特定节点	系统展示报告树列表，报告树图结构，以及特定节点报告详情	通过
QC-CHECK-2	查看单一状报告	1.新建测试任务，提交单一状报告 2.在审核页面查看单一状报告列表 3.点击列表中特定项	系统展示单一状报告列表以及报告详情	通过
QC-CHECK-3	管理者审核报告	1.查看任意一个报告详情 2.使用选择框为报告评分	系统反馈评分成功，界面上该节点类型变为已评分	通过

Label	# Sam...	Average	Median	90% Line	95% Line	99% Line	Min	Maximu...	Error %	Throughput
报告树查...	600	217	213	227	229	238	207	705	0.00%	56.7/sec
单一状报...	600	195	194	203	208	217	185	221	0.00%	56.9/sec
审核接口	600	32	32	35	37	42	29	208	0.00%	59.7/sec

图 5.6: 审核模块接口的JMeter汇总图

5.3　实验评估

为了验证质量控制系统是否提升数据质量，以及质量控制系统算法的合理性，本文分别对比质量控制系统保障下的测试任务T1和未保障下的测试任务T2，对两个测试任务的数据集进行分析。测试任务T1为M平台上的途牛APP任务，测试任务T2为M平台上的众悦学车APP任务。

表 5.6: 报告有效性对比

sum_threshold	测试任务	Bug报告总数	有效报告数目	有效率
7	众悦学车APP	531	415	78.2%
	途牛APP	201	182	90.5%
8	众悦学车APP	531	324	61.0%
	途牛APP	201	146	72.6%

首先，根据本文的Bug报告有效性检测方法，对两次测试任务的报告有效性进行检测，表 5.6为报告有效性对比。当*sum_threshold*为7时，质量控制系统可使途牛APP 任务报告有效率为90.5%，比未进行质量控制的众悦学车APP高出了12.3%。当*sum_threshold*为8时，质量控制系统使途牛APP任务有

效率为72.6%，比众悦学车APP 高出11.6%。综合来看，质量控制系统使报告有效率平均提高了12.0%，使Bug报告质量得到了有效改善。

其次，本文分析M平台上两个质量控制系统保障下的测试任务途牛APP和途牛网，对自动评分的正确率进行验证。以人工评分的结果s_p为标准，判定自动评分s_a是否与s_p 一致。本文定义，$|s_a - a_p| \leq 1$ 时人工评分和自动评分一致，否则评分不一致。表 5.7 为两个测试任务评分一致性结果，由表可知途牛APP自动评分和人工评分一致率为80.1%，途牛网一致率为82.2%，平均一致率为81.2%，自动评分算法效果较好。

表 5.7: 人工评分和自动评分一致性结果

测试任务	Bug报告总数	一致数	不一致数	一致率
途牛APP	201	161	40	80.1%
途牛网	174	143	31	82.2%

最后，本文分析信息增益评估效果，以途牛APP和途牛网两个测试任务的Bug报告为数据集，人工对比所有父子报告的文本增益，判定系统是否成功筛选出所有的增益句子，实验中使用的*document_threshold*为0.77。表 5.8所示为两个测试任务的文本增益验证结果，途牛APP共有32对Fork生成的父子关系，其中30对正确筛选出文本增益，正确率为93.8%；途牛网14对父子关系中，正确筛选出13 对父子关系，正确率为92.9%。文本增益评估的平均正确率为93.4%，效果较好。

本文经过验证实验确定的*document_threshold*为0.77。表 5.9展示了未筛选出所有文本增益的三对报告详情，三对报告中存在不相同的句子对，但是计算得到的相似度都超过了阈值，所以未能成功筛选。例如编号为1的父子报告中，父报告存在句子a“无法进行下一步邀请好友或者进行砍价的操作”，子报告存在句子b“同时也无法进行确认进行搜索”，因为两个句子的结构相似，计算得到的相似度为0.817，因此未能成功筛选句子b。不能因此调大阈值，阈值太大将筛选出大量相似无增益的句子，失去评估的意义，本文允许存在适量的误差。

表 5.8: 文本信息增益评估结果

测试任务	父子关系数目	正确筛选	错误筛选	正确率
途牛APP	32	30	2	93.8%
途牛网	14	13	1	92.9%

表 5.9: 错误筛选的报告详情

编号	类型	文本描述	子报告文本增益	错误筛选及原因
1	父报告	搜索30030554没有返回搜索结果，无法进行下一步邀请好友或者进行砍价的操作	没有返回下拉可选列表	a.无法进行下一步邀请好友或者进行砍价的操作 b.同时也无法进行确认进行搜索 未筛选出b，因为a与b相似度为0.817
	子报告	搜索栏输入30030554后，没有返回下拉可选列表，同时也无法进行确认进行搜索		
2	父报告	首次点开发现页面下的“旅行视频”点开后会黑屏	点击发现页面下的“关注”、“推荐”等各选项出现bug：无规律复现瞬间黑屏	a.首次点开发现页面下的“旅行视频”点开后会黑屏 b.黑屏短暂时间后又可以正常显示页面内容 未筛选出b，因为a与b相似度为0.791
	子报告	点击发现页面下的“关注”、“推荐”等各选项，出现bug：无规律复现瞬间黑屏，黑屏短暂时间后又可以正常显示页面内容		
3	父报告	进入牛人新品，点击左侧按钮无字符部分无反应	仅文字部分点击有反应，初步判断为超链接未附加至外层DIV导致，影响用户体验	a点击左侧按钮无字符部分无反应 b.点击立即抢购按钮无反应 未筛选出b，因为a与b相似度为0.786
	子报告	进入牛人新品，点击立即抢购按钮无反应，仅文字部分点击有反应，初步判断为超链接未附加至外层DIV导致，影响用户体验		

5.4　本章小结

本章主要对照第三章提出的系统需求，对Bug报告有效性检测、Bug报告自动评估、监控和反馈、Bug报告审核四个模块的重要接口，在高并发的情况下，进行了功能和性能测试。测试结果表明，需求分析阶段所提目标均已达到，系统具有较高的完整性和稳定性。同时，对有效性检测方法、自动评分方法和信息增益评估方法进行实验验证，结果表明，本文所提方法均取得了较好效果，可以满足用户的使用。

第六章　总结与展望

6.1　总结

为了满足随着互联网高速发展而越来越迫切的众包测试需求，充分发挥海量众包工人的群体智能优势，提高测试效率和质量，协作式众包测试受到广泛关注。但是协作式众包测试面临非专业众包工人带来的测试结果质量不可控的问题，已有研究尚未解决协作式众包测试的质量控制问题。为了解决该问题，本文设计并实现了面向协作式众包测试的质量控制系统，从以下四个方面保证了Bug报告质量。

第一，Bug报告有效性检测模块分析Bug报告特征并确定质量指标，通过阶跃变换函数确定指标的合理范围与正负向影响，设置合理阈值检测Bug报告有效性，为评估Bug报告质量等级与反馈众包工人操作奠定基础。

第二，Bug报告自动评估模块利用文本和图片分析技术提取子报告信息增益，定量评估协作过程中每个众包工人的贡献；同时利用灰色关联分析法和熵权法对报告自动评分，确定报告的质量等级，为管理者审核报告奠定基础。

第三，反馈和监控模块利用有效性检测模块和自动评估模块对报告质量的判断，实时评估众包工人的每个操作，记录其失信行为，并通过反馈提高众包工人的质量意识；同时提供众包工人失信行为和协作关系的实时监控，供平台管理者识别并剔除恶意众包工人。

第四，Bug报告审核模块全面展示报告基础信息和自动评估信息，为管理者审核报告提供有效参考；同时设计并实现了交互友好的报告审核方式，帮助管理者快速审核海量报告。

在技术实现上，系统使用Angular2和Spring Boot加快开发速度，使用Nginx实现请求负载均衡，使用Redis缓存加速数据访问，使用MongoDB数据库主从备份提高数据库容灾能力。在系统测试上，各个模块的接口都通过了功能和性能测试，完成了既定需求。在实验评估上，报告有效性检测和自动评估算法取得了较好结果。质量控制系统使Bug报告有效性提高了12.0%，整体上提高了Bug报告数据集的质量，促进了协作式众包测试下的质量控制研究。

6.2　展望

当前系统还存在很多不足和可改进的地方，未来可以从以下四个方面对质

量控制系统进行完善和改进，使系统取得更好的质量控制效果。

第一，目前对Bug报告的质量评估都基于报告基本属性，无法自动判断其描述缺陷的真实性。为了解决该问题，系统可对待测软件进行自动化分析，得到可疑缺陷列表，进而自动判断报告真实性。

第二，系统提取质量指标时使用了其他研究者提供的指标字典，该字典中的词汇具有普适性，不具有典型性，导致评估不同软件的Bug报告质量时，评估方法的准确性可能不同。为了解决该问题，系统可分析待测软件的说明书和界面，自动生成面向该软件的指标字典，增加评估的准确性。

第三，系统只能保证Bug报告质量，无法根据高质量报告生成可交付给任务发布者的缺陷列表。为了解决该问题，系统可进一步对高质量报告进行聚类和融合，以便更快速地得到可交付的缺陷列表。

第四，随着业务量的增长，系统数据库可能遇到延迟和堵塞等问题，导致业务服务的响应变慢。为了解决该问题，系统可进一步构建数据库集群，或者对数据库进行分库和分片等。

参考文献

[1] J. Howe, The rise of crowdsourcing, Wired magazine 14 (6) (2006) 1–4.

[2] 张志强, 逄居升, 谢晓芹, 周永, 众包质量控制策略及评估算法研究, 计算机学报36 (8) (2013) 1636–1649.

[3] 冯剑红, 李国良, 冯建华, 众包技术研究综述, 计算机学报38 (9) (2015) 1713–1726.

[4] N. Leicht, N. Knop, C. Müller-Bloch, J. M. Leimeister, When is crowdsourcing advantageous? the case of crowdsourced software testing, in: 24th European Conference on Information Systems, ECIS 2016, Istanbul, Turkey, June 12-15, 2016, 2016, p. Research Paper 60.

[5] S. Zogaj, U. Bretschneider, J. M. Leimeister, Managing crowdsourced software testing: a case study based insight on the challenges of a crowdsourcing intermediary, Journal of Business Economics 84 (3) (2014) 375–405.

[6] 陈英奇, 赵宇翔, 朱庆华, 科研众包视角下公众科学项目的任务匹配模型研究, 图书情报知识No.183 (3) (2018) 6–17.

[7] H. Shen, Z. Li, J. Liu, J. E. Grant, Knowledge sharing in the online social network of yahoo! answers and its implications, IEEE Trans. Computers 64 (6) (2015) 1715–1728.

[8] P. Cheng, X. Lian, L. Chen, J. Han, J. Zhao, Task assignment on multi-skill oriented spatial crowdsourcing, IEEE Trans. Knowl. Data Eng. 28 (8) (2016) 2201–2215.

[9] M. Bogers, A. Afuah, B. Bastian, Users as innovators: A review, critique, and future research directions, Journal of Management 36 (4) (2010) 857–875.

[10] M. Lease, On quality control and machine learning in crowdsourcing, in: Human Computation, Papers from the 2011 AAAI Workshop, San Francisco, California, USA, August 8, 2011, 2011.

[11] K. Mao, L. Capra, M. Harman, Y. Jia, A survey of the use of crowdsourcing in software engineering, Journal of Systems and Software 126 (2017) 57–84.

[12] 章晓芳, 冯洋, 刘峘, 陈振宇, 徐宝文, 众包软件测试技术研究进展, 软件学报29 (1) (2018) 69–88.

[13] B. Gardlo, S. Egger, M. Seufert, R. Schatz, Crowdsourcing 2.0: Enhancing execution speed and reliability of web-based qoe testing, in: IEEE International Conference on Communications, ICC 2014, Sydney, Australia, June 10-14, 2014, 2014, pp. 1070–1075.

[14] D. Liu, R. G. Bias, M. Lease, R. Kuipers, Crowdsourcing for usability testing, Proceedings of the American Society for Information Science and Technology 49 (1) (2012) 1–10.

[15] C. Schneider, T. Cheung, The power of the crowd: Performing usability testing using an on-demand workforce, in: Information Systems Development, Reflections, Challenges and New Directions [Proceedings of ISD 2011, Heriot-Watt University, Edinburgh, Scotland, UK, August 24 - 26, 2011], 2011, pp. 551–560.

[16] E. Dolstra, R. Vliegendhart, J. A. Pouwelse, Crowdsourcing GUI tests, in: Sixth IEEE International Conference on Software Testing, Verification and Validation, ICST 2013, Luxembourg, Luxembourg, March 18-22, 2013, 2013, pp. 332–341.

[17] G. Calikli, A. B. Bener, Empirical analyses of the factors affecting confirmation bias and the effects of confirmation bias on software developer/tester performance, in: Proceedings of the 6th International Conference on Predictive Models in Software Engineering, PROMISE 2010, Timisoara, Romania, September 12-13, 2010, 2010, p. 10.

[18] Y. Tung, S. Tseng, A novel approach to collaborative testing in a crowdsourcing environment, Journal of Systems and Software 86 (8) (2013) 2143–2153.

[19] Y. Feng, Z. Chen, J. A. Jones, C. Fang, B. Xu, Test report prioritization to assist crowdsourced testing, in: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015, Bergamo, Italy, August 30 - September 4, 2015, 2015, pp. 225–236.

[20] Y. Feng, J. A. Jones, Z. Chen, C. Fang, Multi-objective test report prioritization using image understanding, in: Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, ASE 2016, Singapore, September 3-7, 2016, 2016, pp. 202–213.

[21] D. Tapscott, A. D. Williams, Wikinomics: How mass collaboration changes everything, Penguin, 2008.

[22] W. Wu, W. T. Tsai, Z. Hu, Y. Wu, Towards a game theoretical model for software crowdsourcing processes, in: Crowdsourcing, Springer, 2015, pp. 143–161.

[23] H. Zhai, T. Lingren, L. Deleger, Q. Li, M. Kaiser, L. Stoutenborough, I. Solti, Web 2.0-based crowdsourcing for high-quality gold standard development in clinical natural language processing, Journal of medical Internet research 15 (4) (2013) e73.

[24] Y. Baba, Statistical quality control for human computation and crowdsourcing, in: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden., 2018, pp. 5667–5671.

[25] S. Kochhar, S. Mazzocchi, P. Paritosh, The anatomy of a large-scale human computation engine, in: Proceedings of the acm sigkdd workshop on human computation, ACM, 2010, pp. 10–17.

[26] A. Sorokin, D. A. Forsyth, Utility data annotation with amazon mechanical turk, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2008, Anchorage, AK, USA, 23-28 June, 2008, 2008, pp. 1–8.

[27] G. Génova, J. M. Fuentes, J. L. Morillo, O. Hurtado, V. Moreno, A framework to measure and improve the quality of textual requirements, Requirements Engineering 18 (1) (2013) 25–41.

[28] F. Gutierrez, Spring Boot Messaging, Messaging APIs for Enterprise and Integration Solutions, Springer, 2017.

[29] Q. V. Le, T. Mikolov, Distributed representations of sentences and documents, in: Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014, 2014, pp. 1188–1196.

[30] R. Xu, T. Chen, Y. Xia, Q. Lu, B. Liu, X. Wang, Word embedding composition for data imbalances in sentiment and emotion classification, Cognitive Computation 7 (2) (2015) 226–240.

[31] A. Lazaridou, N. T. Pham, M. Baroni, Combining language and vision with a multimodal skip-gram model, in: NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015, 2015, pp. 153–163.

[32] M. Nii, Y. Tuchida, T. Iwamoto, A. Uchinuno, R. Sakashita, Nursing-care text evaluation using word vector representations realized by word2vec, in: 2016 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2016, Vancouver, BC, Canada, July 24-29, 2016, 2016, pp. 2165–2169.

[33] Z. Cui, X. Shi, Y. Chen, Sentiment analysis via integrating distributed representations of variable-length word sequence, Neurocomputing 187 (2016) 126–132.

[34] 刘思峰, 蔡华, 杨英杰, 曹颖, 灰色关联分析模型研究进展, 系统工程理论与实践33 (8) (2013) 2041–2046.

[35] C. E. Shannon, Prediction and entropy of printed english, Bell System Technical Journal 30 (1) (1951) 50–64.

[36] 程启月, 评测指标权重确定的结构熵权法, 系统工程理论与实践30 (7) (2010) 1225–1228.

[37] P. Kruchten, The 4+1 view model of architecture, IEEE Software 12 (6) (1995) 42–50.

[38] X. Chen, H. Jiang, X. Li, T. He, Z. Chen, Automated quality assessment for crowdsourced test reports of mobile applications, in: 25th International Conference on Software Analysis, Evolution and Reengineering, SANER 2018, Campobasso, Italy, March 20-23, 2018, 2018, pp. 368–379.

[39] M. Allahbakhsh, B. Benatallah, A. Ignjatovic, H. R. M. Nezhad, E. Bertino, S. Dustdar, Quality control in crowdsourcing systems: Issues and directions, IEEE Internet Computing 17 (2) (2013) 76–81.

[40] S. J. Yang, A readability formula for Chinese language, University of Wisconsin–Madison, 1970.

简历与科研成果

基本情况 宋少行，男，汉族，1995 年 10 月出生，河南省洛阳市人。

教育背景

2017.9～2019.6 南京大学软件学院 硕士

2013.9～2017.6 东北大学软件学院 本科

这里是读研期间的成果（实例为受理的专利）

1. **宋少行**，韩鹏飞，“一种面向协作式众包测试的质量控制方法”，申请号：201910268371.0，已受理。
2. 韩鹏飞，**宋少行**，“一种面向众包测试平台的协作方法”，申请号：201910269760.5，已受理。

致　　谢

首先，我想感谢我的导师刘嘉老师，以及陈振宇老师、房春荣老师和何铁科老师，感谢各位老师在我硕士期间给予的指导和帮助。在毕业设计阶段，感谢四位老师从最初选题到项目开发，直至论文的最终完成，持续给予我建设性的指导意见，并耐心给我答疑解惑。本篇论文的完成离不开老师们的帮助和指导，在此我要向四位老师表达最崇高的感谢和敬意。

其次，我要感谢实验室的李玉莹学姐、杨乙霖学姐，感谢她们在我开发项目和撰写论文过程中提供的建议和帮助，感谢她们帮助我审阅论文并提出修改意见。同时，我还要感谢曹玉瑶、梅杰、韩鹏飞、孙伟松、汪小发、谭若骐、陶德彬等同学，感谢他们与我一同讨论和解决问题，帮助我完成项目和毕业论文，同时使我学习到很多知识和优秀品质。

然后，我要感谢硕士阶段南京大学软件学院对我的培养，让我从一个开发能力薄弱的人，成长为一个具有较完整软件领域知识体系和较强开发能力的硕士毕业生，感谢软件学院所有老师的辛勤付出和谆谆教诲。

我还要感谢我的父母和妹妹，感谢他们在我读书阶段对我的鼓励和支持，感谢他们在我遇到困难和挫折时的关心和理解。

最后，感谢为审阅本论文而辛勤劳动的各位专家学者和老师们，在以后的成长过程中，我一定会继续努力，不辜负大家对我的期望。