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GitHub开源社区的Bug定位系统的设计与实现
作 者 姓 名	方文强
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	陈振宇 教授

2019 年 5 月 30 日

学　　号：**MF1732024**

论文答辩日期：**2019 年5 月14 日**

指 导 教 师：　　（签字）

The Design and Implementation of Bug Localization System for GitHub Open Source Community

By

Wenqiang Fang

Supervised by

Professor **Zhenyu Chen**

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GitHub开源社区的Bug定位系统的设计与实现

工程硕士(软件工程领域) 专业 2017 级硕士生姓名：方文强

指导教师（姓名、职称）：陈振宇 教授

摘 要

GitHub作为现今最大的开源社区，其应用市场提供了代码质量检查、项目持续集成、进度管理等方面的工具供开发者使用，但依然缺乏有效的Bug定位工具。对于软件开发者，Bug修复是一项耗时耗力的任务。Bug定位是Bug修复过程中的重要环节。通过工具预测导致Bug产生的源代码文件，辅助Bug定位，能够提高Bug修复效率。

本文设计与实现了一个面向GitHub开源社区的Bug定位系统。本系统采用B/S架构，前端采用Vue.js 框架搭建，使用iView 组件库组织页面，通过axios包与服务端进行通信。服务端采用Spring Boot搭建并实现业务逻辑，基于RESTful原则定义通讯接口，同时集成MyBatis、log4j等中间件用于ORM映射和记录系统操作日志。数据持久层采用Mysql数据库存储信息。本系统集成实现了三种常用的基于信息检索的Bug定位技术：BugLocator、BLUiR+和AmaLgam+。本系统使用AST抽象语法树提取源代码文件信息，再将其与Bug报告一起输入给定位技术。系统根据Bug定位技术自身定义的指标进行计算，对每个源代码文件评分得出预测结果。开发者依靠Bug定位系统预测结果作为参考，辅助Bug修复。

本文选取了开源项目Zookeeper截止2018年10 月16 日产生的1243份Bug 报告，对本系统的Bug定位准确度进行了初步验证。系统预测结果Top1的命中率为52.53%，Top5的命中率为78.92%，Top10的命中率为87.45%。本系统性能测试模拟了500用户并发访问7个常用接口，平均响应时间为247ms，最高447ms，均未超过500ms，且响应错误率为0，满足小规模开发者群体使用的性能要求。

关键词： 开源社区，Bug定位，Vue.js，Spring Boot，信息检索

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 Bug Localization System for GitHub Open Source Community

SPECIALIZATION: Software Engineering

POSTGRADUATE: Wenqiang Fang

MENTOR: Professor Zhenyu Chen

Abstract

As the largest open source community in the world today, GitHub provides tools about code quality checking, project continuous integration, and schedule management for developers in its MarketPlace. But there is still no effective tools to help developers in fixing bug. For software developers, fixing bug is a time-consuming and laborious task. And bug localization is an important part of this process. Using tools to predicting buggy source code files can help developers locate bug and improve efficiency in fixing bug.

This paper designs and implements a bug localization system for the GitHub open source community. The system adopts the B/S architecture. The web page is built with Vue.js framework and iView component library. The browser side communicates with the server side through axios library. We defined communication interface based on RESTful API.To implement business logic, the server side is built with Spring Boot framework. At the same time, it integrated with components such as MyBatis for ORM mapping , log4j for recording operation logs and etc. For data persistence layer, we use Mysql to store the information. The system implements and integrates three IR-based bug localization technologies named BugLocator, BLUiR+, and AmaLgam+. It extracts information from source code files using the Abstract Syntax Tree. And then put this information together with bug reports. Bug localization technologies use them to calculate scores for each source code file. Finally, developers could check buggy source code files according to the results.

This paper selects a total 1243 bug reports created before October 16,2018 about the open source project Zookeeper to verify the bug location accuracy of this system

initially. The hit rate of Top1 in the bug localization result is 52.55%, the hit rate of Top 5 is 78.94%, and the hit rate of Top 10 is 87.45%. In addition, we simulated 500 users concurrently accessing 7 common interfaces in the system performance test. The results show that the average response time of the seven interfaces is 247ms, the highest is 447ms. All responses take less than 500ms and the error rate is 0. It meets the performance requirements used by small-scale developer groups.

Keywords: Open Source Community, Bug Localization, Vue.js, Spring Boot, Information Retrieval

目　录

表　目　录

图　目　录

第一章　引言

1.1　项目背景

开源社区包含海量的代码仓库，拥有众多的个人开发者和开发团队 [1]。这其中，个人开发者占据很大一部分比例，对于绝大多数个人开发者而言，通常是利用业余时间编写开源项目 [2]，此外还需要定位和修复他人在使用自己开源项目时所遇到的Bug。当用户在使用软件的过程中遇到Bug——软件的实际行为与预期效果不一致（即失效） [3]，可以通过提交Bug报告的方式向开发者反馈。开发者在收到Bug报告后，通过复现等方式以确认Bug报告的有效性，一旦Bug报告被确认为有效，则开发者就需要根据提交者的描述开展软件调试，定位导致Bug产生的源代码文件并对其修改，以修复该Bug [4]。而对于开源社区中已经具有一定规模的仓库而言，其所有者就需要花费大量时间和精力来确认和修复他人提交的Bug报告，代价十分高昂，不利于开源社区更好地发展。

基于以上所述情况，为开源社区提供一个Bug定位系统十分必要。在翻阅相关文献了解国内外现有Bug 定位技术后，系统选择实现和集成三种基于信息检索的Bug定位方法，提供给开源社区中的开发者们使用，省去他们了解该领域具体知识和自己动手实现工具的麻烦，只需要导入Bug报告集即可使用本系统提供的Bug定位算法，找出导致Bug产生的源代码文件。能够有效减少开源社区开发者在定位Bug上所需要花费的时间和精力，使其可以更好地专注于对自己开源项目进度的推进，为开源社区的发展做出更大的贡献。同时，为了收集这些Bug定位方法所需要的Bug报告信息，本系统还附带缺陷追踪 [5]功能供个人开发者使用，避免了以往使用issue 管理Bug报告而造成的混乱（因为issue 除了可以提交Bug之外，还会被用来记录未来需要添加的特性和关联pull request、里程碑等信息 [6]），使得个人开发者可以更加清楚地了解到自己开源项目当前的健康状况。

1.2　国内外研究现状

Bug定位技术被认为是从源代码中进行特征提取技术的一部分，即从众多源代码文件中识别出一个已经实现的特定功能模块的过程 [7]。目前国内外现有的Bug定位技术主要分为以下两种类型，一种叫做Spectrum-Based Bug Localization（SBBL，基于频谱的Bug定位技术） [8]，另一种名为Information

Retrieval-Based Bug Localization（IR-based，基于信息检索的Bug定位技术）。

Spectrum-Based Bug Localization也被称作为Spectrum-Based Fault Localization (SBFL) [9]。该方法是一种动态Bug定位技术，需要动态执行系统，获取程序的执行轨迹。通常需要先搭建好已经发现Bug程序的测试环境，再将该程序的所有源代码文件和两组测试用例的入参作为输入部分，其中一组测试用例的入参可以使Bug复现，而另一组测试用例的入参则能够使得测试用例正确执行通过。接着，运行这两组测试用例，SBBL技术会对这一过程进行监控，收集并记录该程序在此过程中产生的程序频谱，程序频谱主要指程序执行过程中产生的关于程序语句的覆盖信息（被覆盖为1，未覆盖为0），以及是否通过信息。可以用来收集程序频谱的工具有很多种，其中Cobertura使用最为普遍 [10]。基于收集到的程序谱，以及对每组测试用例入参的程序元素所得统计数据，SBBL方法就可以计算出每个程序元素的可疑性分数，目前用于计算程序元素可疑性分数的主流方式是使用Tarantula质量检测工具 [11]。程序元素的可疑性分数越高，则其越有可能是导致出现Bug 的原因。SBBL在计算出所有程序元素的可疑性分数之后，将程序元素按照可疑性分数进行降序排序，再将其输出并发送给开发人员作进一步人工检查。

Information Retrieval(IR)-Based Bug Localization技术是一种基于信息检索的静态Bug定位技术。首先，它将输入的Bug报告（即Bug报告中的摘要和描述等文本信息）视作为一种查询条件，并将源代码库中的程序元素（即源代码文件）视为一个文档，通过信息检索技术，根据程序元素与所查询的Bug报告的相关性对所有程序元素进行排序。这个技术背后的想法依据是，程序元素中包含的词汇与输入的Bug报告中词汇的重合度很高时，那么便可以认为这个程序元素极有可能是和发现的Bug 强相关。根据这个想法，通过使用文本检索模型（Text Retrieval Models）、IR-based等技术 [12]，计算各种程序元素和输入Bug报告之间的文本相似性，然后对程序元素按照其与Bug报告的文本相似性分数进行降序排序，最后再将排序结果输出给开发人员进行手动检查。

所有基于信息检索的Bug定位技术都需要从源代码文件和Bug报告文件中提取文本内容，再对提取出来的文本进行预处理。预处理步骤分为以下两步：第一步，从源代码文件中提取注释和标识符名称，对于Java文件，可以使用JDT工具轻松地将其从源代码文件中提取出来；第二步，在提取源代码注释、标识符和Bug报告的文本内容之后，对它们进行文本预处理 [13]。文本预处理的目的是标准化源代码和Bug报告中的词汇，其步骤通常按如下顺序进行：文本规范化，删除停用词和词干还原。传统基于信息检索的Bug定位技术主要使用隐性语义索引（LSI）、狄利克雷划分（LDA）和向量空间模型（VSM） [14]等方

法，利用Bug报告和源代码文件中出现的词汇进行Bug定位。除此之外通常还会通过对其他相关信息进行挖掘，以缩小定位范围、提高定位精度，主要思路有：（1）提供更多的Bug报告，利用机器学习相关技术，通过提高数据集量级以提高Bug定位精度；（2）对Bug 报告进行筛选和加工处理，提高Bug 报告质量或挖掘Bug 报告之间的关系来提高定位精度或缩小定位范围，如相似Bug报告的关联、提炼Bug报告以剔除噪声信息（称之为查询重排）等，这些措施都可以在一定程度上提高Bug定位结果的准确度。

通过以上对Spectrum-Based Bug Localization动态Bug定位技术类型和Information Retrieval-Based Bug Localization静态Bug定位技术类型的介绍和使用过程的阐述，能够清楚地了解两种技术类型的差异。IR-based 此类静态Bug定位技术相对于SBBL而言更轻量级，如果要使用SBBL 技术，那么就必须要为每一个项目都分别提供一个用于执行测试用例的测试环境，这就是SBBL 的局限性所在，因为对于不同项目，甚至于不同测试用例而言，所需要的测试环境都可能存在差别，无法一劳永逸地解决测试环境搭建的问题，做到单次搭建即可满足所有项目的使用需求。与动态Bug定位方法不同，静态Bug定位技术则只需要对项目包含的所有源代码文件和Bug报告做相应文本处理操作即可，而源代码文件和Bug报告在一定程度上可以视之为结构化文档，所以即便对不同项目而言，文本信息提取与处理步骤都大致相同，所以提供静态Bug定位技术给开源社区中诸多开发者们使用更贴合实际使用场景。

除了在使用便捷性上的差异，我们更需要关注两种类型Bug定位技术在定位准确度上的表现。对于信息检索技术，最常用的两项评估指标是MAP（Mean Average Precision）和MRR（Mean Reciprocal Rank）。MAP是反应系统在检索全部相关文档性能上的单值指标，系统检索出的相关文档越靠前（Rank 排名越高），MAP 值也就越高，若系统没有返回相关文档，则MAP值为0。其中单个主题平均准确率是每篇相关文档被检索出后准确率的平均值，主集合平均准确率是每个主题平均准确率的平均值 [15]。MRR则相对简单，它将正确结果出现在系统预测结果排名的倒数作为准确度 [16]。现有研究已经表明，基于信息检索的静态Bug定位技术在准确度上完全不逊于SBBL此类动态Bug定位技术。所以，面向开源社区的Bug定位系统拟采用静态Bug定位技术进行实现和集成，为开源社区中广大开发者定位Bug提供便利、减轻负担。

1.3　本文主要研究的工作

开源社区GitHub的应用市场（Marketplace）中，现有工具大多都是与代码质量检查、持续集成、项目管理方面相关，为了减少修复Bug给开发者们带来

的烦恼，从为开源社区发展做出一份贡献的角度出发，本文将设计并实现一个集三种基于信息检索的Bug定位技术于一体的系统，供开源社区中开发者们使用，以期减少他们在修复Bug上所需要耗费的时间和精力。从软件工程的角度和对开源社区GitHub 仓库缺陷管理现状的调研结果出发，以下将对目前搭建系统所需要解决的问题进行阐述：

首先，简化用户操作，使系统能够与开源社区GitHub信息完美对接十分必要 [17]。这样能够减少用户在源代码文件同步、历史Bug报告转移上耗费的时间，否则即使用户减少了定位Bug所要消耗的时间，但却需要花费更多时间用于向本系统迁移和同步源代码、Bug报告数据，这会大大降低用户使用体验，与我们开发本系统的初衷相违背。所以，能够接入GitHub、获取授权信息等十分必要，这可以使用户不需要手动向本系统上传源代码文件等信息。此外，为了方便用户迁移以往使用其它缺陷追踪系统所产生的历史Bug报告，还需要支持用户批量导入这些历史Bug报告，而不必逐个手工输入，基于该点，还需要定义一个能够被系统解析的历史Bug报告文件格式，用户将所有的历史Bug报告信息整理成该格式即可一键导入，减少不必要消耗的时间。

其次，对要实现和集成Bug定位算法的挑选也非常重要。开源社区中仓库数量众多，选择哪几种具有代表类型的定位算法才能满足绝大多数开源开发者的需求。基于这个问题我们对GitHub的开源仓库进行调研，最终将其大致分为如下三类：历史久远且信息量较少的开源项目、新成立的开源项目以及信息丰富的大型开源项目。对于较为久远的开源项目而言，在软件工程方面的意识不及如今这么严谨，可能只保留了历史Bug记录以及对应修复的源代码文件记录，对于这种类型的仓库，我们选取了BugLocator [18]这种方法进行了实现与集成；开源社区的日渐繁荣，新的开源项目数量愈加庞大，这种新的开源项目几乎不存在历史Bug报告信息，无法使用上一种BugLocator方法进行Bug定位，据此我们添加了BLUiR+ [19]这种Bug 定位方法；很多大型的开源项目，从创建伊始便使用了十分规范的Bug报告管理方式，每份Bug 报告都记录了丰富的Bug信息，如Bug报告的创建时间和最新变更时间、每个Bug 报告提交者的详细信息、更为准确的Bug报告描述信息等，为了充分利用这些信息，从而提升Bug定位的准确度，选取AmaLgam+ [20]作为第三种Bug定位技术进行实现和集成。

面向开源社区的Bug定位系统，以向开源社区的开发者提供便捷使用体验为基础，根据对开源社区现有仓库类型的划分结果，选择、实现并集成与之相对应的三种高准确度IR-based Bug定位算法。结合GitHub接入，缺陷追踪等功能模块帮助开源社区的开发者，降低其在修复Bug上所需要耗费的时间与精力，减轻为了修复Bug所造成的麻烦和困扰。

1.4　本文的组织结构

本文研究的面向开源社区的Bug定位系统，将依据软件的生命周期，从软件需求、设计、实现和测试四个环节，按照如下组织结构对其进行论述：

第一章为引言部分。详细地介绍了设计与实现面向开源社区的Bug定位系统的项目背景，给出国内外对于Bug定位技术的研究现状，阐述了基于这些Bug定位技术搭建面向开源社区的Bug定位系统所需要解决的问题和进行的研究工作，最后简要介绍了本文的组织结构。

第二章为相关技术综述部分。阐述了在设计和实现面向开源社区的Bug定位系统过程中，所涉及的相关技术和概念。涉及前端框架Vue.js和iView、通讯方式axios与RESTful API、Bug报告、文本预处理以及信息检索技术。

第三章为需求分析与概要设计部分。根据前期调研，提出开发者对该系统的基本需求，并对系统的整体设计思路进行了概述，将整个系统的功能模块拆分为用户信息功能模块、缺陷追踪功能模块和Bug定位功能模块，并以系统用例图和用例描述等方式加以详细描述。同时，对系统的数据库表单进行设计。

第四章为详细设计与实现部分。在第三章需求分析与概要设计的基础上，重点阐述了系统中用户信息模块、缺陷追踪模块和Bug定位模块核心部分的详细设计和具体细节，给出部分核心功能的代码实现和最终效果图，对Bug定位效果进行评估，并在最后一节中对系统进行系统测试和功能验证，确保系统正确实现。

第五章为总结与展望部分。对搭建面向开源社区的Bug定位系统和撰写论文期间所做的工作进行总结，并且就该Bug 定位系统的未来改进和发展方向作进一步规划和展望。

第二章　相关技术综述

本章主要内容为介绍在开发面向开源社区Bug定位系统过程中所使用到的部分关键概念和技术。主要包含前后端框架、通信方式、Bug报告、文本预处理以及信息检索。

2.1　前端技术

本系统可视化方案基于两种技术：Vue.js和iView。Vue.js是一套用于构建Web用户界面的渐进式框架；iView则是一套基于Vue.js的高质量UI组件库。将这两种技术框架结合在一起使用，可以满足使用本系统开源开发者的审美需求和提供良好人机交互体验。

2.1.1　Vue.js

Vue.js作为如今GitHub上star数最多的前端开发框架，吸引了大批开发人员的注意力，所以虽然其只是作者尤雨溪的个人秀，但却拥有着强有力的社区支持。它是一套用来构建Web用户交互界面的渐进式前端框架 [21]。与其它大型前端框架不同，Vue.js 从最初就被设计为是一种可以自底向上进行逐层应用的框架。Vue.js核心库将关注点主要聚焦于视图层，这使得其对使用者而言，不仅通俗简单、易于上手，还便于将其与其他第三方库或者已有项目进行整合统一。另一方面，当它与现代化的工具链以及各种支持类库结合在一起使用时，Vue.js也完全能够胜任为复杂的单页应用提供驱动。

HTML作为一种声明式静态语言，是编写静态Web页面的基础。但是一旦Web 页面涉及到动态修改需求时，就会显得十分笨重，以往的方法是通过jQuery操纵修改已经渲染完的HTML页面中包含的DOM元素来达到动态显示的效果 [22]，这一方式较为繁琐。而Vue.js则是按照MVVM模式（Model－View－ViewModel，模型－视图－视图模型）进行实现 [23]，可以达到模型与视图的双向绑定效果 [24]。

MVVM的实现如图 2.1所示。当Vue完成初始化视图后，通过观察者模式，Observer 会劫持监听所有data中定义的属性，当其中属性的值发生改变时，告知Dep数据已经发生变更。再由Dep通知Watcher，触发Watcher中Update 方法，以重新渲染视图，从而实现视图与模型之间双向绑定。

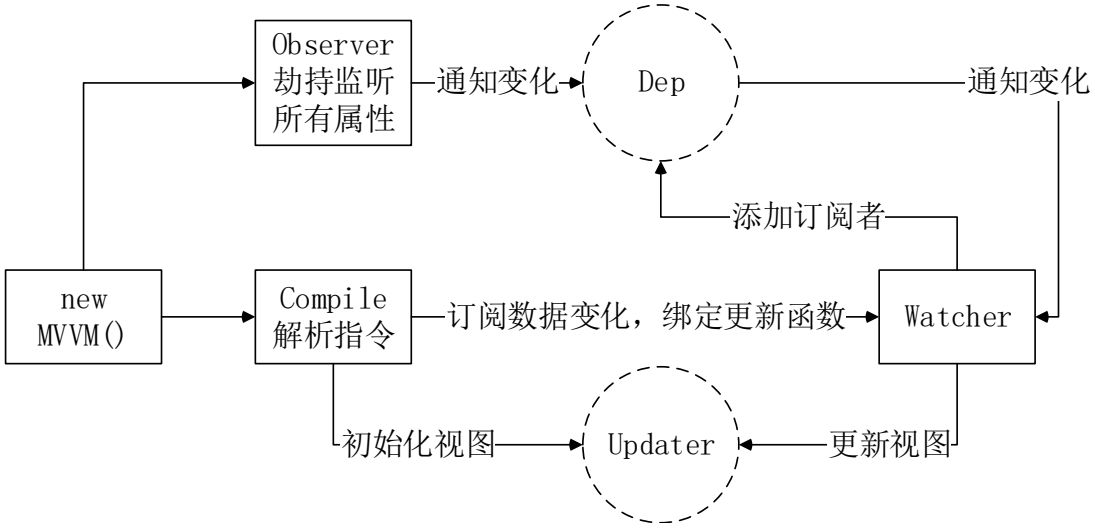

图 2.1: Vue.js双向绑定原理图

本系统需要视图能够根据数据进行动态渲染。鉴于Vue.js能够通过MVVM模式满足本系统在该方面的需求，同时兼具通俗简单、易于上手的优点，所以将使用其作为本系统的前端开发框架。

2.1.2　iView

iView是一套基于Vue.js前端开发框架的开源UI组件库，其主要服务对象是基于Vue.js搭建的Web应用 [25]。与传统的BootStrap 等UI框架不同，iView除了提供众多UI组件外，还为每种组件封装了常用的方法和事件，减轻了开发者不断造轮子的负担 [26]。只需要使用npm在前端工程中引入该UI组件库，即可对页面进行布局，使用各UI组件以及封装的监听事件。

iView作为Vue.js目前主流UI开发组件库之一，具有多种优点。譬如，它拥有众多漂亮、细致的UI组件，使用这些组件能够满足用户审美需求。此外它还提供功能丰富、高质量的事件响应，适用于绝大多数应用场景，开发者只需要根据实际需求编写接收响应后的业务处理逻辑即可。同时，iView支持自定义主题，能够满足用户个性化定制需求。其使用文档也足够丰富详细，方便本系统在开发时根据具体需求进行查阅和使用。最后，使用空间自由灵活，API 对开发人员十分友好，能极大提高本系统开发效率。

基于如上对iView简要介绍和对其优势之处的阐述。本系统采用iView作为UI组件库编写前端页面，使用其提供的UI组件和封装的事件响应，以满足在提升开发效率的同时，依然能够保证界面简介美观。

2.2　通信方式

2.2.1　Axios

Axios是基于promise开发出来的一种HTTP客户端，在浏览器或者Node.js环境下都可以流畅运行。它为处理其他节点的HTTP和XMLHttpRequests类型的请求提供了一个接口，除此之外，它还使用了polyfill 为ES6新的promise语法封装了请求 [27]。作为被Vue.js官方推荐用来替代vue-resource，和服务端进行通信的插件。

本系统选择Axios作为通讯方式主要基于以下七点要素：第一，能取消请求，满足特定场景下需求。第二，Axios支持promise，使得开发者在开发时可以根据服务端不同响应结果作出不同的操作。第三，能拦截请求和响应，捕获服务端返回的异常信息。第四，可以自动转换JSON 数据，省去对数据封装与拆封操作。第五，支持浏览器和node.js，兼容当前所有现代浏览器。第六，能够自动转换请求和响应数据，无需使用JS 实现HTTP请求解析。第七，能够在浏览器端实现跨域，降低开发过程中的调试代价。

2.2.2　RESTful API

REST（表现层状态转换）全称为REpresentational State Transfer，由Roy Thomas Fielding于2000年提出 [28]。REST是一种万维网软件架构风格，目的是便于让不同的软件或程序在互联网中可以相互传递信息。REST是基于HTTP而提出的一组约束和属性，它是一种设计风格而并不是标准 [29]。

RESTful API具有如下6点约束：一、客户-服务端（Client-Server）：通信只可以从客户端向服务端单方面发起，具体表现为请求-响应这种形式。二、无状态（Stateless）：通信的会话状态（Session State）应该全部由客户端负责维护，服务端不参与对会话状态的管理。三、缓存（Cache）：服务端响应客户端的内容可以被缓存在通信链的某一处，以达到改善网络效率的目的。四、统一接口（Uniform Interface）：作为通信链的组件，它们之间的相互通信必须按照约定的统一接口进行交互，以提高交互的可见性。五、分层系统（Layered System）：为了将架构分解成若干个层级，可以使用约束组件的行为的方式（即单个组件只能感知到与其交互的紧邻层，除此之外的层级对其而言都是透明的）。六、按需代码（Code-On-Demand）：支持通过下载并执行一些代码（例如Java Applet、Flash或JavaScript），对客户端的功能进行横向扩展。

RESTful API相较于其他两种主流Web服务实现方案（SOAP和XML-RPC）更加简洁 [30]，所以本系统通讯接口设计选择采用REST风格设计和实现。

2.3 Bug报告

当用户在使用某一个软件或系统的过程中发生了未知的错误，他们便可以通过向开发者提交一份关于该错误的文档以告知开发者，这样的文档就可以称之为一份Bug报告 [31]。每份Bug报告含有多条字段，在这诸多字段中，我们比较关注的有：Bug报告的区分标识符（id）、总结标题（summary）、Bug具体描述（description）、创建Bug报告的时间（open date）以及提交Bug报告的人员信息（reporter）。

表 2.1显示了一份Bug报告的详细信息，该报告是从Eclipse的Bugzilla 上摘录下来的。它的ID为76138，描述的是Eclipse中的一个与Ant插件相关的Bug，该插件没有按照设置好的效果显示文本信息。Bug报告的ID为我们在版本控制系统中查找修复该Bug所修改的问题提供了依据，同时，它的创建时间也可以帮助我们识别出在此之前提交的那些Bug报告，而标题和描述字段被用来帮助我们理解用户遭遇到了一个什么样的Bug，Bug提交者的信息可以帮我们找出之前这位提交者所提交过的历史Bug 报告。此外，该表还列出为了修复Bug ID为76138的Bug报告所变更的两份源代码文件。

表 2.1: Bug报告信息及其修复的源代码文件

Bug报告ID	76138
创建时间	2004-10-12 21:53:20
总结标题	Ant editor not following tab/space setting on shift right
具体描述	This is from 3.1 M2. I have Ant-Editor-Display tab width set to 2, ”Insert spaces for tab when typing” checked. I also have Ant-Editor-Formatter-Tab size set to 2, and ”Use tab character instead of spaces unchecked. Now when I open a build.xml and try to do some indentation, everything works fine according to the above settings, except when I highlight a block and press tab to indent it. It’s the tab character instead of 2 spaces that’s inserted in this case.
提交者	Jing Xue
修复的源代码文件	org.eclipse.ant.internal.ui.editor.AntEditor.java org.eclipse.ant.internal.ui.editor.AntEditorSourceViewerConfiguration.java

面向开源社区的Bug定位系统所选取的三种IR-based定位技术，均是将Bug报告文本和源代码文件作为输入源信息，据此计算该项目中每个源代码文件导致该Bug产生的可疑性分值，最后再将分值排名靠前的十个源代码文件作为Bug定位结果，供开发人员手工确认时使用。

2.4　文本预处理

基于信息检索（IR-based）的Bug定位技术中，对文本预处理的技术通常分为以下三步：文本标准化（text normalization）、去停用词（stopword removal）以及词干提取（stemming） [32]。对文本进行预处理的目的是为了将源代码文件和Bug报告分割成一串词，这样就可以使用信息检索的技术对Bug 报告和源代码文件进行分析。在文本预处理这一步中，像类名、变量名这样的复合词将会被切割开来，同一词根的词将会被还原成词根的形式。

现在具体介绍文本预处理三个步骤。首先，对文本进行标准化，我们需要剔除其中的标点符号，对文本进行分词（将词从段落中提取出来，或是将标识符从源代码文件中提取出来），以及标识符分割 [33]。这一步中，对处理源代码文件处理是先将其转换成抽象语法树（Abstract Syntax Tree, AST），通过抽象语法树便可以很方便的提取出我们所需要的标识符，因本系统目前只支持对Java 项目进行缺陷定位，所以根据Java 标识符的驼峰式命名规范，对各标识符进行切分得到一系列的新词。举个例子，对于从抽象语法树中提取的方法名“getUserByCode”而言，经过驼峰切词后，便会得到四个新词：“get”、“User”、“By”和“Code”。同时，我们还需要保留原来的方法名“getUserByCode”。同理，对于类名“BinaryDocValuesRangeQuery”而言，经过文本标准化后，会得到“Binary”、“Doc”、“Values”、“Range”、“Query”和“BinaryDocValuesRangeQuery”共六个词。接着，我们需要剔除掉那些对文本语义影响很小的停用词，如“a”、“about”、“very”等，这里我们选用了NLTK库中包含的所有停用词。最后，我们需要对每个词提取其词干，如“plays”、“played”、“playing”，这三个词从词义上来说相同的，所以我们利用提取词干的方式将它们都还原成“play”的形式，提高该词的词频，在这个步骤中，我们选择使用现有的Porter Stemming算法 [34]来解决这一问题。

2.5　信息检索

信息检索（Information Retrieval，IR）的定义是从一个大型信息集合（通常存储在计算机上）里的众多非结构化的文档之中，提炼出我们所需要的文本

信息。从二十世纪末开始，随着计算机的小型化和互联网普及程度的增加，我们生活的世界开始涌现大量的信息 [35]，对于这种信息量快速增长的现象，我们称之为信息爆炸。而信息检索技术则可以帮助人们从繁杂的找寻到自己想要的信息，网络搜索引擎（谷歌、百度等）正是最典型的信息检索应用 [36]。2004年，Pew Internet Survey通过调查发现，92%的互联网用户都认为是一个获取日常信息的好地方 [37]。

当用户向信息检索系统中输入查询信息时，信息检索这一过程便开始了。查询是用户对于所需求信息的一种声明，如输入到网络搜索引擎中字符串文本。对于信息检索而言，查询并不能将集合中的对象唯一的标识出来，恰恰相反，集合中的多个对象都可能是与查询信息相匹配的，只是它们相互之间的相关度不一样。

对象通常是存储在数据库或其他信息集合中的实体，但是信息检索并不是简单的将用户的查询与数据库的信息进行匹配就行了。事实上，与现有的数据库SQL语句查询相反，信息检索返回的结果可能与查询相匹配，但也可能不匹配，所以需要对查询得到的结果进行排序。这也是信息检索与数据库查询之间最显著的差异 [38]。

大多数信息检索方法都会去计算信息集合中每个实体与查询条件之间的匹配分数，并根据这个分数进行排序，再将排序后的结果展现给用户。同时，用户对该过程进行的迭代，也会影响每个结果的分数从而使得查询效果更符合用户的期望 [39]。面向开源社区的Bug定位系统选取的三种基于IR-based的Bug定位算法也遵循这种方式，根据新提交的Bug报告结合源代码文件、历史Bug报告集合计算得出每个源代码文件与该Bug报告的匹配分数，再将排序后的定位结果返回给用户。本文将在第四章中对三种算法的计算框架与具体实现进行详细介绍。

2.6　本章小结

本章节是对面向开源社区的Bug定位系统开发过程中所选用的相关技术做介绍，具体涵盖了如下内容：根据前端开发实际需求选取了Vue.js框架和iView UI组件、前端通信客户端Axios和服务端接口设计原则、缺陷跟踪系统的主要实体Bug 报告以及文本预处理的方式与步骤，最后还对静态Bug定位方法的核心概念信息检索进行了相关介绍。

第三章　需求分析与概要设计

3.1　项目总体规划

本系统由用户信息功能模块、缺陷追踪功能模块、Bug定位功能模块三大部分组成。其中，用户信息功能模块负责维护从开源社区GitHub授权获取的用户信息，同步用户的开源仓库，关联用户的邮箱和为其提供新消息提醒，为之后的Bug 定位算法输入源代码文件和甄别Bug 报告提交者提供支撑；缺陷追踪部分可以理解为一个作用于Bug 报告整个生命周期的功能模块，从Bug 报告的提交到审核再到修复或关闭阶段，将Bug 提交者和开源社区的开发者联系在一起，为开发者提供接收Bug报告、缺陷追踪分支选择功能，支持导入历史Bug 报告，同时还为提交者提供一个反馈开源项目Bug的系统，保障开发者能够获取更加准确的Bug定位结果；Bug 定位功能模块属于该系统的核心组成部分，由它接收用户发起的Bug 定位请求、整理产生的历史Bug 报告数据、提取源代码文件和git log 中的存储的相关信息，将这些信息整合在一起执行相应的Bug 定位算法，最后再将所有运行的定位算法的结果反馈给开发者们参考。

除了上述功能以外，为了保证系统健康有序的发展，还引入信誉积分机制、定位结果满意度反馈机制，以达到减少恶意提交Bug报告的现象和为未来改进定位算法提供数据集支撑。在这一章节中，我们将会具体论述每个功能模块所对应的具体需求和应用场景、系统使用流程以及其他需要进行说明的设计。

3.2　系统需求分析

3.2.1　用户群体

本系统未来将会在开源社区GitHub的应用市场Marketplace上线，所以目标用户群体是活跃在开源社区中的开发者们，每一名开发者都可以通过GitHub 账号登录使用本系统，利用系统为自己代码仓库附加缺陷追踪功能，使开发者拥有一个可以集中处理自己Bug 报告的地方，而不需要再在众多的issue 中甄别出哪些属于Bug，哪些又属于提交者的疑问或帮助需求。再根据Bug报告创建并执行Bug 定位任务。同时，当其在使用其他开源项目过程中遇到Bug，可以通过本平台向该仓库提交Bug 报告。简言之，对于每个使用我们系统的用户，他即可以是Bug定位任务的创建者，又可以是Bug报告的提交者。关于这两种用户角色，以及他们使用各功能模块的流程，将在本章节后续小节中作详细介绍。

3.2.2　功能性需求

用户信息维护功能模块

该功能模块主要负责维护用户信息，共包含五项子功能点。首先，为了能够无缝对接开源社区，使开源社区开发者们都可以便捷地使用本系统提供的Bug定位服务，需要通过接口获取用户信息，初步计划使用OAuth2.0 方式对接GitHub，优先满足此最大开源社区中开发者的需求。其次，开源开发者在成功登录本系统后，需要浏览属于自己的开源仓库，如此才能做到为特定开源仓库添加缺陷追踪功能和进行Bug定位，所以系统还需要在已获得授权的前提下读取特定用户开源仓库信息。接着，某些用户可能会恶意向他人开源仓库提交Bug 报告，扰乱系统运营秩序，为了尽可能避免此类情况发生，系统需要为每个账户添加信誉积分属性，用以记录用户操作记录；而当信誉积分低于一定阈值时，则无法提交新的Bug报告。有些开源仓库所有者在使用本系统之前，通过GitHub 的issue 或其他缺陷追踪系统（如Bugzilla，Jira等）跟踪Bug 报告进展，为了保证Bug定位服务的准确性，将新Bug 报告的提交者与历史Bug报告提交者联系起来，我们需要邮箱关联的功能，将使用同一个邮箱的Bug 提交者视为同一事实人。最后，为用户提供新消息提醒功能，如果用户提交的Bug 报告有新回复，或者其所属的开源仓库发生变更，都需要以徽标的形式展现在页面中，同时系统还会根据用户个人设置，判断是否还需发送邮件通知。

缺陷追踪功能模块

该功能模块作为系统附带的缺陷追踪系统，同样也包含五个子功能点。第一，当开源开发者在系统上查看自己开源仓库时，可以根据自身需求选择为某个特定仓库添加缺陷追踪功能，以管理该仓库的Bug 报告和和使用定位服务。第二，一旦某开源仓库被其所有者纳入缺陷追踪，则其他开发者就可通过本系统提交关于该仓库的Bug 报告。第三，为了满足大版本发布后切换分支的需求，对于已经纳入缺陷管理的开源仓库，其拥有者可以设置对该仓库具体哪一条分支进行缺陷跟踪。第四，仓库所有者在使用该系统之前可能会使用其他缺陷追踪系统，对于存在于其他系统中的历史Bug 报告，本系统需要提供接口，支持用户使用某种格式的文件将这些历史Bug报告一次性批量导入，从而保证定位算法在使用历史Bug报告数据集计算时的准确性。第五，Bug报告提交者有查看自己历史提交Bug 报告的需求，开源仓库所有者也有查看自己仓库所有Bug报告的需求，所以需要为用户提供Bug 报告列表和详细信息展示功能。Bug 报告包含的内容除了在2.3 节中介绍的字段外，本系统还添加了状态（status）和评论（comments）信息，状态信息用以表示Bug 报告当前所位于的处理阶段，评论则为开发者们提供交流方式。

Bug定位功能模块

该功能模块是本系统核心所在，在实现和集成了三种基于信息检索Bug定位技术的基础上，大致划分为如下四个子功能点。首先，对于每份Bug报告，用户既可以根据自身需求勾选相应Bug定位方法并创建定位任务，也可以简单地一次性批量创建三个定位任务而不需要了解具体技术细节。其次，对每个Bug定位任务的步骤进行划分，实时显示进度可以减轻用户等待任务完成过程产生的焦虑情绪。再次，对于每个Bug 定位任务的结果，除了用列表将最终计算结果展现给用户，还需要使多种定位算法结果的交集部分高亮、提供每个结果文件的commit记录和关联Bug报告，使得定位结果信息更加立体和丰富。最后，为了收集用户对每个定位算法的满意情况，当其将Bug 报告的状态从已确认置为关闭时，如果基于该Bug 报告执行了Bug 定位任务，那么就需要其为这些定位任务结果的准确性进行打分，并提供修复该Bug所提交的commit信息。

3.2.3　非功能性需求

除了功能性需求，非功能性需求对于一个软件系统而言也非常重要，这关系到系统的健壮性和鲁棒性。以下将从可用性、安全性、性能、可靠性、可扩展性和其他非功能需求六个方面对其进行叙述。

可用性：高可用性的产品并不代表它们的界面十分简单，可用性并不是要让一切都变简单，而是可以提升用户在使用软件过程中的效率、有效性和满意度 [40]。效率代表执行预期任务的速度，有效性代表执行任务的结果正确与否，最终满意度则是终端用户在使用软件时的舒适程度。

安全性：用户只能进行自己有操作权限的操作，没有操作权限的操作无法执行。本系统的安全性要求有：用户只能查看自己的消息通知；只有仓库所有者才能更改所属Bug 报告的状态；只有仓库所有者才能为Bug报告创建Bug定位任务等。

性能：系统响应用户操作的时延必须要限定在一个可接受的范围之内，如各种信息查询操作响应时间要小于3秒，Bug定位任务执行时长要小于10 分钟。对于特殊情况，如调用GitHub OAuth响应速度取决于GitHub 本身，需要使用Loading等图标降低用户的焦虑。

可靠性：可靠性是指在软件系统的运行过程中，其不会发生故障的概率。即使是在用户做出了不合法的操作，也应当为用户返回一个友好性的提示，而不会直接导致整个软件系统崩溃、宕机而无法继续正常运行 [41]。如用户使用本系统批量导入历史Bug报告数据的json 文件不符合要求时，应提示用户“json文件格式无效”，而不是报HTTP 500 错误。

可扩展性：为了方便在系统开发搭建完成之后，对系统功能进行后续扩展，和为其他系统提供服务，整个系统的设计要做到高内聚低耦合，每个模块之间尽可能的相互独立。

其他非功能性要求：为了保证系统在遇到不可预期的错误时，能够尽快发现问题、找出原因，所有的用户操作和系统运行都应以日志的形式记录下来。

3.2.4　系统用例

系统用例图

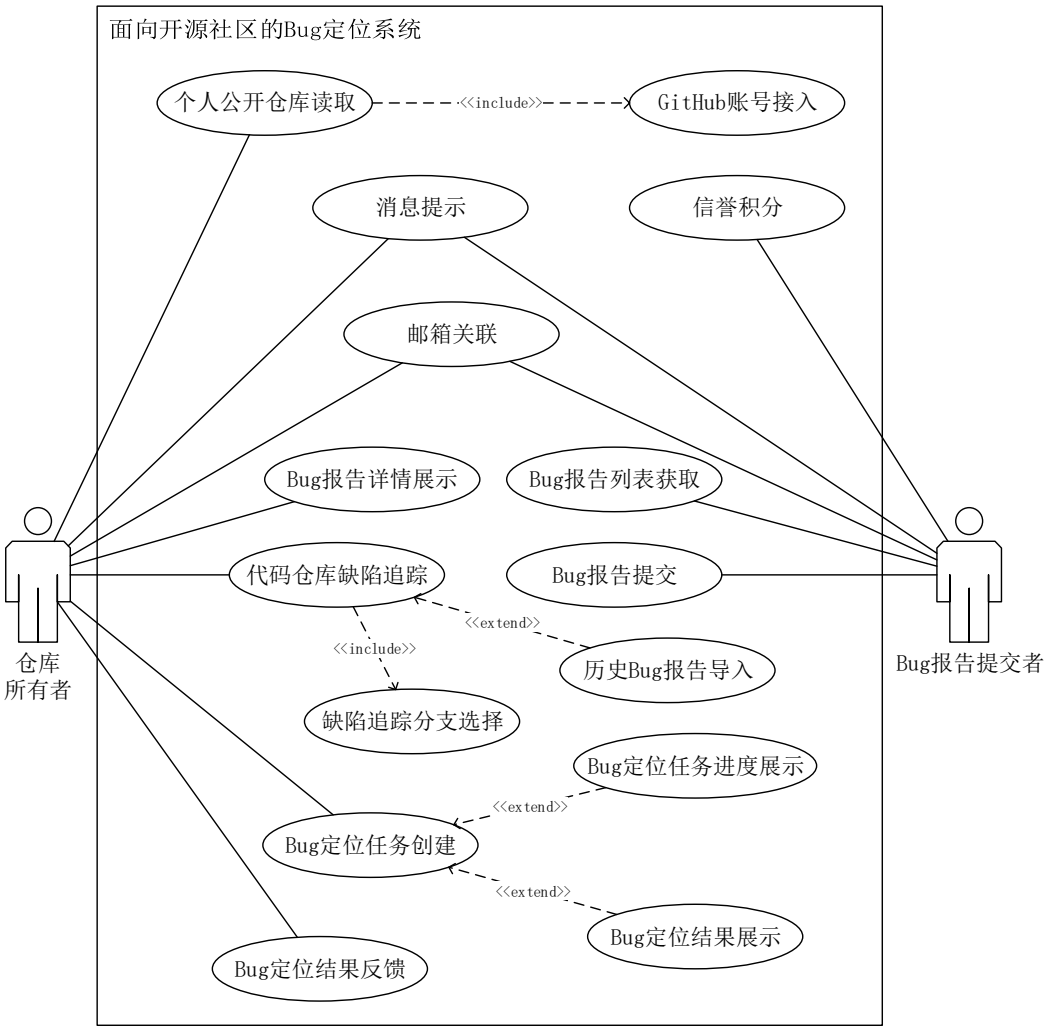

图 3.1: 系统用例图

统一建模语言（Unified Modeling Language，UML）是一种用于说明、可视化、构建和编写一个正在开发、面向对象、软件密集系统制品的开放方法。使

用UML 对复杂系统进行分析与设计，从一定程度上降低开发的复杂性，优化软件体系结构，增加软件系统可扩展性和可维护性。根据第3.1.2 节功能性需求，以及两种系统用户角色，我们给出如图 3.1 所示的系统用例图。对三个功能模块进行划分，共得到15个用例。本节后续将对各用例进行具体描述。

子用例描述

表 3.1详细描述了GitHub账号接入用例。为了方便验证开源仓库用户身份，后续能够合法获取用户开源仓库等信息，本系统设计该用例。用户点击主页右上方“使用GitHub账号登录”，在GitHub站点页面输入用户名密码，授权本系统同步其账号与开源仓库信息。

表 3.1: GitHub账号接入用例描述表

ID	UC_01
名称	GitHub账号接入
参与者	主要参与者：仓库所有者、Bug报告提交者 目的：同步用户的GitHub账号
描述	仓库所有者或Bug报告提交者在在打开系统主页后，点击页面右上方“使用GitHub账号登录”按钮，进行第三方登录，同步GitHub账号信息。
触发条件	仓库所有者或Bug报告提交者点击系统主页右上方“使用GitHub账号登录”按钮。
前置条件	仓库所有者或Bug报告提交者进入系统主页。
后置条件	仓库所有者或Bug报告提交者可以使用其他权限内的功能。
正常流程	1.仓库所有者或Bug报告提交者进入系统首页； 2.点击系统首页右上方的“使用GitHub账号登录”按钮； 3.页面跳转到GitHub站点的登入页面，用户输入GitHub 账户名与密码，点击确认授权按钮； 4.系统获取GitHub授权后，调用相关API接口接收用户信息，再将用户信息存储或更新到系统的数据库中。与此同时，页面提示用户正在获取授权信息中； 5.授权登录成功，页面跳转到系统首页。
异常流程	5a.若GitHub响应时间过长，无法获取用户GitHub账号信息，则提示用户“GitHub暂无响应，请稍后再试”，并将路由跳转回系统首页。

表 3.2 具体描述了本系统同步个人开源仓库用例。为了使用户可以便捷地使用本系统提供的缺陷追踪和Bug定位服务，将会为仓库所有者提供GitHub 仓库同步功能，用户在登录系统后点击查看自己的开源仓库后，即可在我的仓库页面浏览自己所属的所有开源仓库，方便后续对有缺陷追踪和Bug 定位需求的仓库附加相关功能，同时又可以减少无关开源仓库对本平台的影响。这样既方便了仓库所有者，也减轻了服务器的存储压力。

表 3.2: 个人开源仓库同步用例描述表

ID	UC_02
名称	个人开源仓库同步
参与者	主要参与者：仓库所有者 目的：为使用缺陷追踪和Bug定位功能提供便利
描述	仓库所有者只需要对同步后的有缺陷追踪或Bug定位需求的仓库进行选择。
触发条件	仓库所有者点击用户名下拉菜单中的“我的仓库”菜单按钮。
前置条件	仓库所有者已经授权GitHub账号登入本系统。
后置条件	用户可以对获取仓库的选项进行设置。
正常流程	1.仓库所有者点击用户名下拉菜单中“我的仓库”按钮； 2.系统根据仓库所有者ID通过GitHub提供的REST API获取该仓库所有者的所有仓库信息； 3.根据从GitHub获取的仓库信息，查询存储在系统数据库中与这些仓库相关的数据，再将这些信息和数据整合在一起返回给页面； 4.页面路由跳转，并将接收到的仓库信息展示给仓库所有者进行操作。
异常流程	3a.若GitHub响应时间过长，无法获取用户的仓库信息，则提示用户“GitHub暂无响应，请稍后再试”。

表 3.3 具体描述了用户邮箱关联用例。考虑到存在某些用户在使用本系统之前，会使用其他缺陷追踪系统（如Jira、Bugzilla）对Bug 报告进行管理的情形，为了在进行Bug 定位时能够确保正确区分出这些历史Bug报告提交者与使用我们系统提交Bug 报告的用户是否为同一事实人，我们提供了邮箱关联功能。即在系统集成的Bug 定位方法中，我们将使用同一个邮箱地址的用户视为同一事实人。

表 3.3: 邮箱关联用例描述表

ID	UC_03
名称	邮箱关联
参与者	主要参与者：Bug报告提交者 目的：找出在本系统提交Bug报告的人和历史Bug报告提交人之间的关系
描述	为Bug报告提交者提供邮箱关联功能，识别出其在其他系统提交的历史Bug报告。
触发条件	Bug报告提交者点击“绑定邮箱”按钮。
前置条件	Bug报告提交者已经使用GitHub账号登录，且尚未绑定邮箱。
后置条件	用户可以正常使用包括提交Bug报告在内的系统相关功能。
正常流程	1.Bug报告提交者点击个人信息页面中的“绑定邮箱”按钮，在弹出的对话框中输入自己的邮箱并点击“发送验证码”按钮，同时将该按钮置灰60秒钟； 2.Bug报告提交者在自己的邮箱中获取系统发送的邮件中包含的验证码，将该验证码输入流程1里对话框的验证码文本框中，再点击“确认”按钮进行验证； 3.系统校验验证码，通过后将该邮箱存入系统的数据库中。
异常流程	2a. Bug报告提交者的邮箱未接收到系统发送的验证码。可在“发送验证码”按钮置灰60s结束后，再次点击该按钮以重新触发步骤1； 3a. 若验证码校验失败，提示Bug 报告提交者“验证码错误”。

表 3.4详细描述了信誉积分用例流程。为防止某些用户恶意向他人开源仓库提交无效的Bug 报告，系统设定并建立了信誉积分机制。当新提交的Bug 报告状态被仓库所有者从待确认（UNCONFIRMED）状态修改为关闭（CLOSED）状态时，为仓库所有者提供了举报功能选项。若仓库所有者认为该Bug报告是由他人恶意提交，则可以选择举报该Bug报告提交者。系统将在仓库所有者举报后，按照信誉积分制度扣除此Bug报高提交者相应数值的信誉积分。当Bug报告提交者的信誉积分低于系统预设阈值时，则无法继续向他人开源仓库提交新的Bug 报告。若希望可以继续正常提交新Bug 报告，Bug 报告提交者可以通过使用系统的Bug 定位等其他功能，增加活跃度、提升自身信誉积分。直到信誉积分高于系统预设阈值时，用户才可以继续提交Bug 报告。

表 3.4: 信誉积分用例描述表

ID	UC_04
名称	信誉积分
参与者	主要参与者：Bug报告提交者，仓库所有者 目的：防止恶意提交Bug报告的情况，保证系统的健康发展
描述	当Bug报告提交者提交的Bug报告被仓库所有者举报后，信誉积分低于阈值，则无法继续提交新的Bug 报告。
触发条件	Bug报告提交者提交了Bug报告。
前置条件	Bug报告提交者被仓库所有者举报恶意提交Bug，从而使得该Bug报告提交者的积分低于系统预设阈值。
后置条件	Bug报告提交者无法继续提交新的Bug报告。
正常流程	1. Bug报告提交者提交Bug 报告； 2. 仓库所有者确认Bug报告后，认定其为恶意提交，举报Bug报告提交这并将Bug 报告状态由待确认转为关闭； 3. 系统根据仓库所有者的举报扣除Bug报告提交者信誉分值； 4. 若扣除后的信誉分值低于阈值，则Bug报告提交者无法继续提交新的Bug报告。
异常流程	若扣除后的信誉分值高于阈值，则Bug报告提交者依然还可以继续提交新的Bug报告。

表 3.5 描述的是消息提示用例。在Bug报告提交者已经提交Bug报告的前提下，当该Bug 报告的状态被仓库所有者修改、有开发者对该Bugb报告发表了新的评论信息或回复、亦或该Bug 报告所属的Bug 定位任务有了新的进度，本系统都会根据具体变更事项，首先向该Bug 报告的提交者和仓库所有者发送站内信，并修改数据库中的徽标信息，若是评论的回复信息，则是向评论者发送站内信；此后再检查这几个涉众的个人通知设置，若其设置了邮箱新消息提示功能，则将该事项的具体内容以系统官方邮件的形式发送给用户，达到能够及时进行消息提示的效果。

Bug报告信息的变更贯穿整个缺陷追踪流程，对于仓库所有者而言，能够及时了解自己所属开源仓库的健康状况以及信息变更情况十分重要。为了满足仓库所有者能够及时获悉自己所属开源仓库下Bug 报告变更情况的需求，面向开源社区的Bug 定位系统据此提供消息提示功能，包括发送邮件、站内信和徽标三种形式。

表 3.5: 消息提示用例描述表

ID	UC_05
名称	消息提示
参与者	主要参与者：Bug报告提交者，仓库所有者 目的：及时将变更通知利益相关方
描述	当Bug报告的状态、评论或者相关定位任务有了新的进展，将以新消息的形式通知有关用户。
触发条件	Bug报告或定位任务发生了变动。
前置条件	Bug提交者已经提交了Bug报告。
后置条件	利益相关方接收到了变更消息提示。
正常流程	1. Bug报告提交者/仓库所有者在Bug报告发布了新的评论，或仓库所有者修改了Bug报告的状态，亦或Bug报告所属的Bug定位任务有了新的进展； 2. 系统根据变更的实际情况，向相关涉众的消息盒子中新增消息，并根据用户设置选择是否发送邮件通知； 3. 相关涉众在变更发生后收到新的消息通知，在设置邮件通知新消息的情况下，也将会收到新的邮件。
异常流程	无

表 3.6详细描述了开发者为自己开源仓库附加缺陷追踪功能的流程。缺陷追踪功能实际上是允许开源仓库使用者在本系统提交有关该开源项目的Bug报告，以便及时将新发现的Bug 反映给仓库所有者。同时为仓库所有者开放历史Bug报告导入功能，从而使在新提交Bug 报告中执行的Bug 定位结果更加准确。即一切都是为了仓库所有者更方便地管理Bug 报告和使用Bug 定位技术这一主题服务。若仓库所有者不希望接收他人提交的Bug 报告，在仓库设置抽屉里勾选拒绝接收他人Bug 报告选项即可，这样仓库所有者可以免受他人提交Bug 报告打扰，做到只专注于自己导入的Bug 报告。

代码仓库缺陷追踪功能是面向开源社区的Bug定位系统的核心模块之一，同时它也是为Bug定位这一系统主题服务的。仓库所有者在个人开源仓库信息总览页面，可以查看到自己所有的开源仓库，打开Hide Fork开关屏蔽掉fork其他人的仓库后，仓库所有者可以根据自身的实际需求，选择某一属于自己的开源仓库，点击该仓库名下方的“为该仓库添加缺陷功能”按钮，从而为该开源仓库添加缺陷追踪功能。

表 3.6: 代码仓库缺陷追踪用例描述表

ID	UC_06
名称	代码仓库缺陷追踪
参与者	主要参与者：仓库所有者 目的：使仓库所有者更方便地管理Bug 报告
描述	方便仓库所有者管理Bug报告，不需要每次运行Bug定位任务都提供所有Bug报告和源代码，系统提供该缺陷追踪功能。
触发条件	仓库所有者点击某一仓库的“添加缺陷追踪功能”按钮。
前置条件	仓库所有者在GitHub中有公开的代码仓库。
后置条件	1. 仓库所有者可以设置代码仓库进行缺陷追踪的具体分支； 2. 仓库所有者可以批量导入该代码仓库的历史Bug 报告； 3. Bug报告提交者可以向该仓库提交Bug报告； 4. 仓库所有者可以根据该仓库的Bug报告新建Bug 定位任务。
正常流程	1. 仓库所有者点击仓库概览中某一个仓库的“为该仓库添加缺陷追踪功能”按钮； 2. 系统服务器通过访问GitHub提供的REST API获取该仓库的开发语言，以校验本系统是否支持为其提供Bug定位服务； 3. 校验通过后，将该仓库的信息添加到系统的数据库中，并返回“添加成功”消息提示。页面根据返回的成功响应，将“添加缺陷追踪”按钮更改为“缺陷跟踪设置”按钮。
异常流程	3a. 若开发语言校验没有通过，则返回语言校验通过的响应； 3b. 根据返回的响应，提示仓库所有者该仓库中不包含系统集成的Bug 定位方法所支持的语言。

表 3.7是对Bug报告提交用例的具体描述。该用例是缺陷追踪功能模块的基础子功能，目的是为了让使用开源项目的用户在遇到Bug 时有地方可以提交Bug 报告，这也便于开发者及时获悉自己开源项目的健康状况。当开源项目的使用者发现新Bug 时，在本系统Bug 报告列表点击“新建Bug报告”按钮，填入需要提交Bug 报告的开源仓库URL，系统将会实时判断该URL 所对应的仓库是否已经被纳入缺陷追踪功能之中，若未纳入缺陷功能或其开源仓库所有者拒收他人提交的Bug 报告，则清除该URL并告知Bug报告提交者详细信息。如若该URL 所对应的开源仓库允许提交新Bug报告，则正确填写Bug报告的其他字段即可顺利提交新Bug 报告。

表 3.7: Bug报告提交用例描述表

ID	UC_07
名称	Bug报告提交
参与者	主要参与者：Bug报告提交者 目的：提交新Bug报告，反馈使用开源项目时遇到的Bug
描述	Bug报告提交者通过系统向仓库所有者提交Bug报告。
触发条件	Bug报告提交者点击Bug报告列表右上“新建Bug报告”按钮。
前置条件	Bug报告提交者已使用GitHub 账号登录本系统。
后置条件	仓库所有者可以查看Bug报告提交者提交的Bug报告。
正常流程	1. 打开Bug报告列表页面，点击右上方“新建Bug报告”按钮； 2. 检查提交者邮箱绑定和信誉积分，打开新建Bug 报告抽屉； 3. Bug报告提交者将Bug所属开源仓库的URL、Bug 报告的标题、详细描述等信息填入相应信息栏，点击“提交”按钮； 4. 系统前端页面在校验新提交Bug报告的每个字段合法性后，将该Bug报告传输给服务器端进而存储到数据库中。
异常流程	2a. 若提交者尚未绑定邮箱，则提示“请绑定邮箱”信息，并跳转到个人信息页面； 2b. 若Bug报告提交者的信誉分低于阈值，则提示“您的信誉分未达标，请通过系统其他功能提升自己的信誉分”； 4a. 若Bug报告中存在非法的字段，则不提交该Bug 报告，同时向提交者提示该字段的填写规则； 4b. 若该URL指向的仓库，被所有者设定为“不接收新的Bug报告”，则提示用户“该仓库所有者暂不接收Bug报告”。

表 3.8 对仓库所有者为自己开源仓库选择缺陷追踪分支的用例作了具体描述。对于开源社区GitHub 上的很多项目而言，每当该项目一个大版本发布过后，都会在保留当前版本分支的基础上，再新建并切换到另一条分支，供下一版本开发继续使用，而不只是简单地一直使用master 这条分支进行版本开发。所以需要仓库所有者根据实际情况选择具体对哪一条分支使用缺陷跟踪功能和Bug 定位服务。当仓库所有者点击已经纳入缺陷管理的开源仓库下方“仓库设定”按钮，页面右侧会弹出该仓库的设定抽屉，服务端通过访问GitHub 提供的REST API接口获取该仓库分支信息，再将分支信息以下拉菜单的形式显示在抽屉中，供仓库所有者按照版本迭代情况自行切换。

表 3.8: 缺陷追踪分支选择用例描述表

ID	UC_08
名称	缺陷追踪分支选择
参与者	主要参与者：仓库所有者 目的：使仓库所有者可以根据版本迭代的实际情况自行切换具体某一条分支进行缺陷追踪
描述	为仓库使用者提供具体到分支粒度的缺陷追踪功能。
触发条件	仓库所有者在仓库总览页面中，点击某一已经纳入缺陷管理功能开源仓库的“仓库设定”按钮。
前置条件	仓库所有者已经为自己的该开源仓库添加缺陷跟踪功能。
后置条件	纳入缺陷追踪功能的开源仓库，已经将分支正确切换到用户所选择的分支上。
正常流程	1. 仓库所有者在仓库总览页面中，点击已经附加缺陷跟踪功能仓库下的“仓库设定”按钮； 2. 页面右侧弹出仓库设定抽屉，在分支选择下拉框中中，仓库所有者根据版本迭代实际情况，选择某一具体分支，点击“保存设置”按钮； 3. 系统提示仓库所有者“更新仓库设置成功”，并将相关设置保存到数据库中。
异常流程	2a. 若未接收到GitHub提供的REST API响应，则提示仓库所有者“获取分支信息失败，请稍后尝试”。

表 3.9 详细地描述了仓库所有者向本系统导入历史Bug 报告用例。设计该用例的目的是为了向Bug定位功能模块中定位算法提供更丰富的历史Bug 报告数据集。仓库所有者在使用该系统之前可能会使用其他缺陷追踪系统，对于存在于这些系统中的历史Bug 报告，系统需要提供接口，支持用户使用某种格式的文件将这些历史Bug报告一次性批量导入，从而保证定位算法在使用历史Bug报告数据集计算时的准确性。仓库所有者在具体开源仓库的“仓库设置”抽屉中，点击“历史Bug 报告模板下载”按钮，导出的JSON 文件格式的历史Bug报告模板，再将存储在其他缺陷追踪系统中的历史Bug 报告数据集整理成与该格式一致JSON 文件。再通过“仓库设置”抽屉，提交并上传该包含历史Bug报告数据集的文件。系统在浏览器端对该文件的格式进行校验，若符合规范则将其提交给服务端，再由服务端解析并保存到系统数据库中，完成批

量导入历史Bug 报告，随即在Bug 报告列表页面即可查看到刚刚批量导入的历史Bug报告。

表 3.9: 历史Bug报告导入用例描述表

ID	UC_09
名称	历史Bug报告导入
参与者	主要参与者：仓库所有者 目的：为仓库所有者将历史Bug 报告导入本系统提供途径
描述	将历史Bug报告集整理成特定格式的json 文件上传至本系统。
触发条件	仓库所有者点击仓库设定抽屉中“导入历史Bug报告”按钮。
前置条件	仓库所有者已经将需要导入历史Bug 报告数据集的开源仓库纳入缺陷追踪功能中。
后置条件	在Bug报告列表页面，可以查看到刚刚批量导入的所有历史Bug报告条目信息。
正常流程	1. 仓库所有者打开已经添加缺陷追踪功能仓库的“仓库设定”抽屉，点击抽屉中“下载历史Bug 报告模板”按钮，下载历史Bug报告json文件模板； 2. 根据模板，将历史Bug报告数据整理成与模板一致的文件； 3. 点击仓库设定抽屉中“上传历史Bug报告”按钮，添加整理后的历史Bug报告文件并上传； 4. 系统校验json文件格式，校验通过后将包含在其中的历史Bug报告数据存储到数据库中； 5. 页面提示“导入历史Bug 报告成功”，并可以在Bug报告列表页面查看到刚刚导入的历史Bug报告条目。
异常流程	4a. 若json文件的格式不符合要求，在页面上提示“历史Bug报告文件不合法”信息，并中止导入流程。

表 3.10为用户获取Bug 报告列表的用例描述表。系统新用户在尚不熟悉本系统的情况下，需要能够查看他人提交的Bug报告，再将其作为自己提交Bug报告的范本，通过这种方式达到尽可能减少低质量Bug报告的数量。此外，仓库所有者也需要对自己所属开源仓库拥有的Bug 报告有一个概览，方便其及时掌握自己所属开源仓库的健康状况。同时，对于Bug报告提交者而言，也需要查看自己以往提交的所有Bug 报告。根据如上需求，本系统设计了该获取Bug 报告列表用例。

表 3.10: Bug报告列表获取用例描述表

ID	UC_10
名称	Bug报告列表获取
参与者	主要参与者：系统用户 目的：用户可以查看自己提交的所有Bug报告，也可以查看自己仓库下的所有Bug报告
描述	展示用户提交的Bug报告或者自己仓库所属Bug报告的列表。
触发条件	用户点击“查看该仓库Bug报告”菜单按钮。
前置条件	用户已经使用GitHub账号登录本系统。
后置条件	用户可以查看到相应的Bug报告列表。
正常流程	1. 在系统主页、纳入缺陷管理的开源仓库等页面，用户点击“查看该仓库Bug 报告”按钮，页面跳转到Bug 报告列表页面，并根据约定接口向服务器发送请求； 2. 服务端根据接收到请求参数，查询隶属于该开源仓库的所有Bug 报告数据，按时间降序排列后发送响应到前端页面； 3. 页面根据服务器返回的Bug报告数据集，将其以表格的形式显示在页面中。
异常流程	3a. 若服务端响应的数据为空数组，则表明该开源仓库下暂无Bug 报告。页面提示“无相关Bug报告”对话框。

表 3.11具体描述了Bug报告详情展示用例的详细流程。每份Bug报告的详细信息包括：该Bug 报告在本系统的唯一标识符ID、标题总结Summary、该Bug报告的创建时间CreatedTime、最近更新时间LastChangeTime、Bug 报告提交者Reporter、当前状态信息Status、Bug 具体描述信息Description以及该Bug报告附属创建的所有Bug定位任务信息。此外Bug 报告还为提交者和仓库所有者提供了一个评论交流的方式，他们可以根据描述字段和评论评论信息对对方进行回复。Bug 报告详情展示是整个缺陷追踪模块中最重要的功能，通过它才能使Bug报告的生命周期变得完整，才能使用本系统的Bug定位服务。

设计Bug报告详情展示用例的初衷是为了帮助开发者提高对新提交Bug的理解。开源仓库所有者可以通过阅读Bug 报告中详细描述字段的方式，进一步了解该Bug表现形式和复现方法。同时，Bug报告提交者又可以根据Bug 报告状态字段知晓当前修复该Bug的进展信息。对于Bug详细描述表达模糊的Bug 报告，两者还可通过在当前Bug报告页面下，通过留言评论进行沟通。

表 3.11: Bug报告详情展示用例描述表

ID	UC_11
名称	Bug报告详情展示
参与者	主要参与者：Bug报告提交者，仓库所有者 目的：查看单个Bug报告的详细信息
描述	为用户提供查看单个Bug报告和评论信息功能。
触发条件	用户点击Bug报告信息列表中某一行最右侧操作栏的“查看Bug报告详细信息”按钮。
前置条件	用户获取的Bug报告列表信息不为空。
后置条件	页面显示该Bug报告的所有信息。
正常流程	1. 用户在Bug报告列表页面选中某一条Bug报告，点击其右侧的“查看详细信息”按钮； 2. 页面跳转到Bug报告详细页面，根据接口向服务器发起Bug报告详细信息请求，服务器从数据库中查询对应的Bug报告和评论信息，汇总后将其返回到页面； 3. 页面根据服务端返回的Bug报告详细数据响应，将该Bug 报告的各条目信息渲染在页面上。 4. 用户根据实际情况，在Bug报告页面下方进行评论留言。
异常流程	4a. 若该用户既不是Bug报告提交者，也不是该开源仓库所有者，则不能发表评论留言。

表 3.12 是对仓库所有者进行Bug 定位任务创建用例的详细描述。Bug定位任务功能模块是本系统所有功能模块的核心，其中创建Bug 定位任务这一子功能点又是该功能模块的基础。该用例为仓库所有者使用Bug 定位方法提供入口，用户可以在状态为已确认的Bug报告详细页面右上方，找到“创建Bug 定位任务”按钮。点击该按钮，页面右侧会弹出创建Bug 定位任务抽屉。在该抽屉中，勾选需要使用的Bug 定位算法（默认全选），允许其指定具体使用哪一种Bug 定位方法。此外，每种Bug 定位算法都会有简要的适用场景说明，仓库所有者可以通过了解阅读使用场景选择合适的定位算法，也可以在不去了解具体细节的情况下使用默认全选的方式批量使用和创建三种Bug 定位任务。勾选完需要使用的定位算法后，点击“创建并运行”按钮，即可批量创建隶属于该Bug 报告的Bug 定位任务。页面将创建定位任务请求发送到服务端，接收到创建成功响应后，即可实时查看各定位的进展情况。

表 3.12: Bug定位任务创建用例描述表

ID	UC_12
名称	Bug定位任务创建
参与者	主要参与者：仓库所有者 目的：为仓库所有者使用本系统集成的Bug定位方法提供入口
描述	仓库所有者根据自身需求，自由勾选或全选本系统提供的Bug定位算法并创建Bug定位任务。
触发条件	仓库所有者在单个Bug报告详细页面，点击页面右上方“创建Bug定位任务”按钮。
前置条件	该Bug报告属于仓库所有者已经纳入缺陷管理的仓库。
后置条件	创建Bug定位任务成功，可以启动该任务进行Bug定位。
正常流程	1. 仓库所有者在单个Bug报告详细页面，点击右上方“创建Bug定位任务”按钮； 2. 在右侧弹出的新建Bug 定位任务抽屉中，选择需要使用的Bug定位算法，点击“创建定位任务并开始运行”按钮； 3. 页面将Bug定位任务提交给服务端保存，并开始执行此Bug定位任务，同时Bug报告详情页面显示此定位任务，并可实时获取该定位任务的进度。
异常流程	2a. 若选择的Bug定位方法已经在该Bug报告中使用过了，提示仓库所有者“该Bug定位任务已经存在”。

表 3.13 是Bug 定位任务进度展示用例的详细描述。Bug定位任务的速度受克隆GitHub仓库当时的网速、历史Bug报告数据集的数量级、开源仓库源代码文件规模的大小以及所选取的Bug定位算法效率影响，这就造成特定情况下某个Bug定位任务会耗时较长的问题，为了减轻这个问题所带来的影响，有必要为仓库所有者提供Bug 定位任务进度展示功能，减少用户等待定位任务完成产生的焦急情绪。

面向开源社区的Bug定位系统将Bug报告的状态作为单独字段存储在系统表单中，其状态包括：定位队列中、正在克隆源代码仓库、正在对数据集进行预处理、正在定位Bug文件和定位已完成五种状态。前端页面每间隔一定时间对服务端发送请求轮询，以获取该Bug 报告所属的Bug 定位任务进度，直至所有Bug 定位任务的状态都变成定位已完成为止。每当定位任务状态与页面当前显示的状态不同，则将更新后的进度重新渲染到页面上。

表 3.13: Bug定位任务进度展示用例描述表

ID	UC_13
名称	Bug定位任务进度展示
参与者	主要参与者：仓库所有者 目的：实时显示定位任务进度，减轻仓库所有者在等待Bug定位任务完成的过程中产生的焦躁情绪
描述	为Bug定位任务提示实时的进度状态（克隆中、Bug报告整理中、文本预处理中、Bug定位中、已完成）信息。
触发条件	用户在创建Bug报告抽屉中，勾选需要的Bug定位算法后点击下方“创建定位任务并开始运行”按钮。
前置条件	该定位任务所属的Bug报告，隶属于仓库所有者已经纳入缺陷管理的开源仓库。
后置条件	该Bug定位任务实时更新展示此时的进度状态。
正常流程	1. 用户点击创建Bug报告抽屉的“创建定位任务并开始运行”按钮，页面显示该任务； 2. 页面定时向服务端发送请求，询问该Bug报告下定位任务的执行进度，服务端将查询到的进度返回给页面，页面根据返回消息对定位任务的状态进行更新； 3. 直到定位任务的状态更新为已完成，定时询问结束，定位任务状态不再更新。
异常流程	2a. 若Bug定位任务的状态被置为“发生错误”，则不再继续更新Bug定位任务状态，并显示“重新执行”按钮。

表 3.14 具体描述了本系统展示Bug 定位结果用例。基于信息检索的Bug 定位方法，最终的结果都是根据一系列输入信息为开源仓库中的每个源代码文件打分，分值越高的源代码文件存在Bug的可疑性就越大，面向开源社区的Bug定位实现和集成的三种基于信息检索的Bug 定位算法也是如此。对于每一种定位算法，最终的定位结果将依据此分数对源代码文件进行降序排列，选取排名前十的源代码文件作为定位的结果，再展现给仓库所有者进行辅助验证。

此外，除了展示每种定位算法计算得出的可疑分外，还要对多种定位算法计算结果的交集作高亮处理，提高仓库所有者检查该源代码文件的优先级。提供每个结果源代码文件commit记录和关联Bug 报告，使定位结果信息更立体丰富，方便开发人员修复Bug。

表 3.14: Bug定位结果展示用例描述表

ID	UC_14
名称	Bug定位结果展示
参与者	主要参与者：仓库所有者 目的：为仓库所有者提供Bug定位任务结果
描述	将Bug定位计算结果和相关关系展示给仓库所有者。
触发条件	用户点击Bug定位任务折叠面板。
前置条件	该Bug定位任务的计算已经完成。
后置条件	仓库所有者可以查看该Bug定位任务的结果。
正常流程	1. 点击已完成的Bug定位任务折叠面板，展开折叠面板； 2. 页面展示该定位方法的计算结果和相关信息；
异常流程	无

如表 3.15 所示，为开源仓库所有者对Bug定位结果进行反馈的用例描述表。为了收集每个Bug 定位方法的定位效果，在仓库所有者将每个含有Bug 定位任务的Bug报告状态转为fixed 时，需要其对所使用的定位方法的准确性打分。为系统在未来改进Bug定位算法提供数据集支撑。当仓库所有者点击“状态修改”按钮后，在弹出的对话框中选择将该Bug报告的状态变更为“fixed”时，同时填写修复该Bug提交的commit信息，以及对隶属于该Bug 报告的所有定位任务逐个打分。

表 3.15: Bug定位结果反馈用例描述表

ID	UC_15
名称	Bug定位结果反馈
参与者	主要参与者：仓库所有者 目的：收集仓库所有者在使用系统进行Bug定位的满意度
描述	变更Bug报告状态为fixed时，为该Bug 报告的定位任务打分。
触发条件	仓库所有者将Bug报告的状态改为fixed。
前置条件	属于该Bug报告的所有Bug定位任务都已经执行完毕。
后置条件	该Bug报告每个定位任务后都会显示仓库所有者的评分结果。
正常流程	1. 点击Bug报告“状态修改”按钮，将其变更为“fixed”； 2. 仓库所有者对各Bug定位方法的准确度打分并提交；
异常流程	2a. 若该Bug报告有未执行完成的定位任务则无法修改状态。

3.3　系统总体设计与模块设计

3.3.1　总体结构

从上一节描述的各个需求和编写的数个用例出发，整个系统总共可以分为三个功能模块。如图 3.2所示，这三个功能模块分别为用户信息模块、缺陷追踪模块以及Bug定位模块。在这三个模块下，每个模块还有一些细分的功能点，分别对应上一节编写的各个用例。

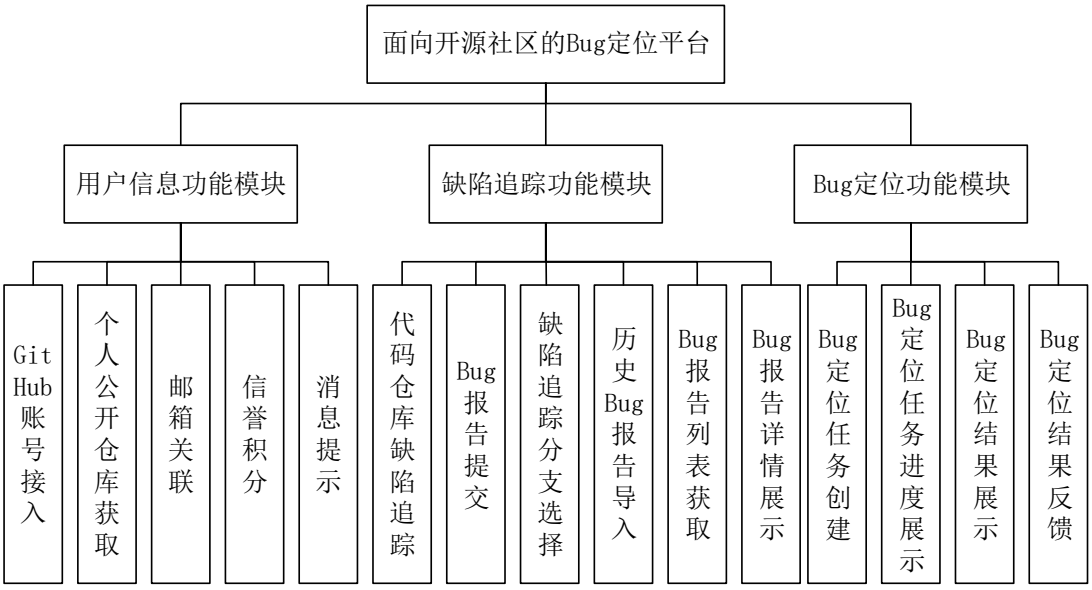

图 3.2: 系统总体结构

用户信息功能模块包含GitHub账号接入、个人公开仓库获取、邮箱关联、信誉积分和消息提示五个功能点；缺陷追踪功能模块包含代码仓库缺陷追踪、Bug报告提交、缺陷追踪分支选择、历史Bug报告导入、Bug报告列表获取和Bug报告详情展示六个功能点；Bug定位功能模块包括Bug定位任务创建、Bug 定位任务进度展示、Bug定位结果展示和Bug定位结果反馈四个功能点。三个模块总计十五个功能点。

3.3.2　系统架构设计

面向开源社区的Bug定位系统使用Vue.js + iView编写Web页面，通过npm中的Axios库和Spring Boot 提供的RESTful API的方式进行HTTP 通信。如图 3.3 所示，为系统架构设计图。

系统所使用到的中间件中，Shiro作为安全框架，用于为系统提供身份验证、授权、密码与会话管理等功能；同时借助于Nginx的反向代理服务特性，解

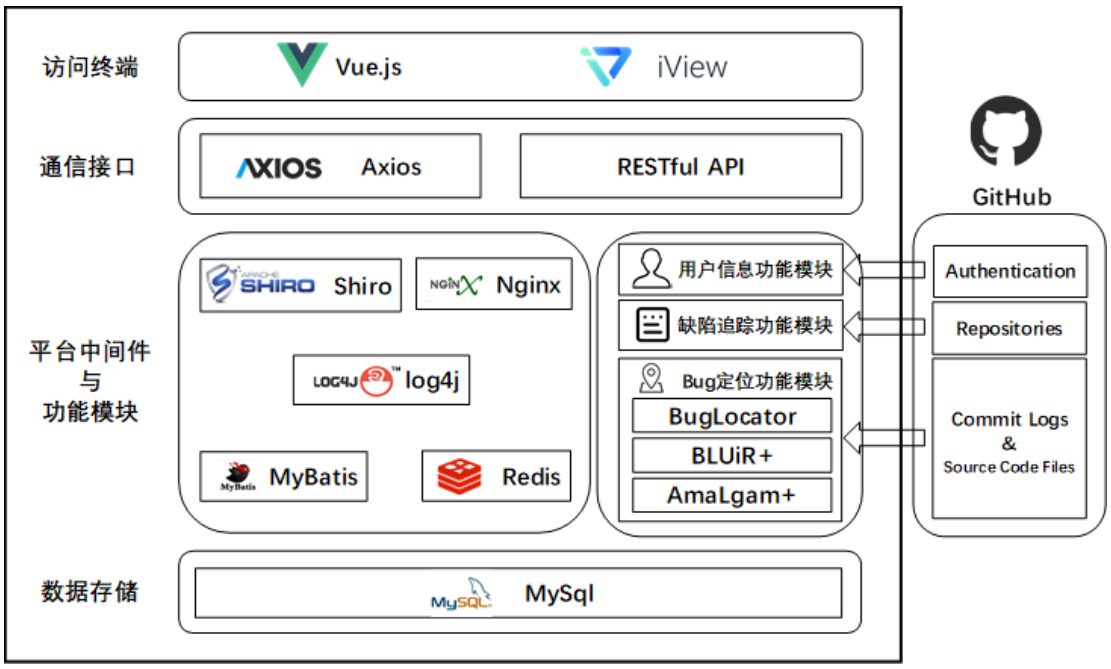

图 3.3: 系统架构图

决前后端分离架构中出现的浏览器不允许跨域的问题；为了保证整个系统的正常运行，能够发现运行时的错误、异常并及时复现解决，系统选用log4j为整个系统提供日志服务，记录各个服务的运行状态、用户的相关操作和系统运行期间的异常信息；为了使代码中的业务逻辑与数据访问逻辑分离，选择将Mybatis框架集成到系统之中，解除了sql语句与业务代码的耦合，提高系统的可维护性；考虑到实际场景中的读操作远大于写操作的情况，引入了Redis 作为缓存，减少与数据库通信的次数，减轻数据库的压力，提高系统响应速度。对于数据持久层，用户信息与Bug定位信息采用关系型数据库Mysql进行存储，为用户提供良好的系统使用体验。

系统主要包含三个功能模块在之前的章节已作具体描述。用户信息功能模块支持用户通过OAuth2.0 的方式关联GitHub第三方账号并登录到本系统，同时还对用户的权限进行限定，为用户提供信誉机制；缺陷追踪功能收集用户在项目开发时所遇到的Bug信息，提升系统核心功能——Bug定位功能的准确度，同时减轻了个人开发者或小型团队在Bug管理上的负担；Bug定位功能模块，作为整个系统的核心服务，基于不同业务场景实现并集成了三种基于信息检索的Bug定位方法供系统用户使用。

3.3.3　系统类结构设计

如图 3.4所示，为面向开源社区的Bug定位系统的系统类图。由于选用了Spirng Boot作为系统后端开发框架，因此每个控制器类都需要添加@ RestController 注释，包括用来处理用户信息的UserController 类，管理Bug 报告的BugReportController 类以及Bug 定位任务相关的BugLocalizationTaskController 类等，由Spring Boot自动解析和返回json、xml等格式的数据。

相关业务处理逻辑类需要添加@Service注释，如UserService、BugReportService、BugLocalizationTaskService 和RepositoryService 等，这样Spring Boot就可以自动扫描并将其注入到Controller控制类中。

3.4　系统数据库设计

通过对数据库表单的设计，可以清晰的看出系统所涉及的实体与要实现的功能之间的关联关系。面向开源社区的Bug定位系统为了解除Sql 语句与业务代码之间的耦合关系，提升整个系统的可维护性，将MyBatis这一ORM框架集成进来，开发时就不需要关心数据之间的具体关系与细节。本文将在接下来的几个段落中对本系统数据库表单的设计方案进行详细描述。

表 3.16 是用户信息表，表名为user。其主要作用是用来存放从GitHub 中获取的用户账号信息。其中id 为该表的主键，用于标识唯一一个用户，同时存储用户账户名、个人GitHub 信息页面的url、获取该用户仓库信息的API、Email地址、信誉积分数值以及用户是否接收消息通知设置。

表 3.16: 用户信息表

字段名	字段解释	字段类型	是否为空
id	用户的唯一标识	int	N
name	用户账号名称	varchar	N
html_url	用户GitHub个人信息页面的url地址	varchar	N
repos_url	获取该用户仓库信息的API	varchar	N
email	用户的email地址	varchar	Y
score	用户的信誉积分数值	int	N
notice_flag	当用户有消息通知时是否发送邮件	int	N

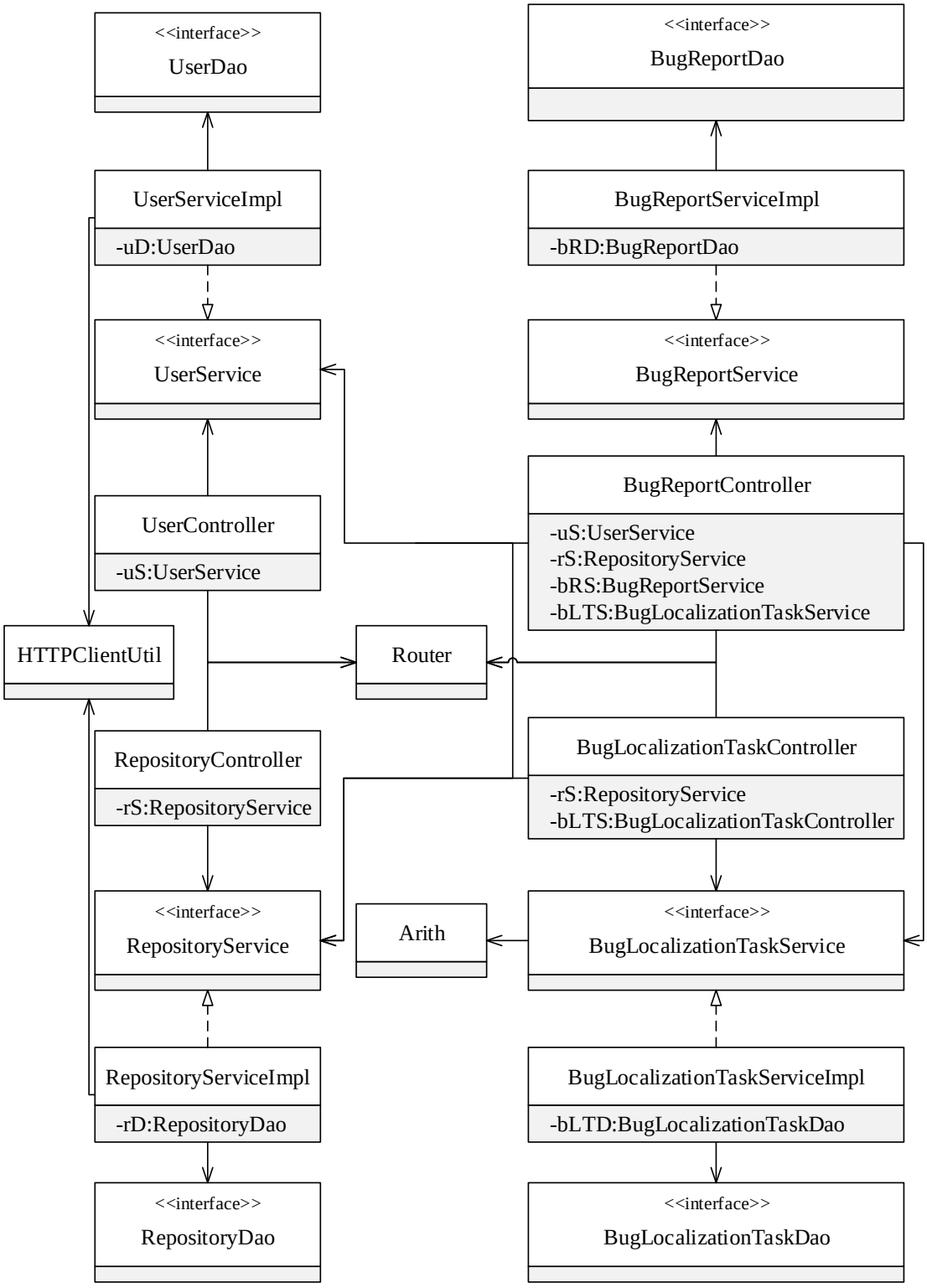
<<interface>>
UserDao
<<interface>>
BugReportDao
UserServiceImpl
-uD:UserDao
BugReportServiceImpl
-bRD:BugReportDao
<<interface>>
UserService
<<interface>>
BugReportService
BugReportController
-uS:UserService
-rS:RepositoryService
-bRS:BugReportService
-bLTS:BugLocalizationTaskService
UserController
-uS:UserService
HTTPClientUtil
Router
BugLocalizationTaskController
-rS:RepositoryService
-bLTS:BugLocalizationTaskController
RepositoryController
-rS:RepositoryService
<<interface>>
RepositoryService
Arith
<<interface>>
BugLocalizationTaskService
RepositoryServiceImpl
-rD:RepositoryDao
BugLocalizationTaskServiceImpl
-bLTD:BugLocalizationTaskDao
<<interface>>
RepositoryDao
<<interface>>
BugLocalizationTaskDao

图 3.4: 系统系统类图

表 3.17 为本系统的仓库信息表，表名为repository。其主要作用是用来存放仓库所有者的仓库信息。其中id为表的主键，用于唯一标识仓库，user_id 为外键，用来关联该仓库的所有者。同时该表还保存了仓库名、全名、克隆该仓库的url、GitHub页面url以及对该仓库进行缺陷追踪管理的分支名。

表 3.17: 仓库信息表

字段名	字段解释	字段类型	是否为空
id	仓库的唯一标识	int	N
user_id	仓库所属用户的id	int	N
name	仓库的名称	varchar	N
full_name	仓库的全名	varchar	N
clone_url	克隆仓库的url	varchar	N
html_url	仓库页面的url	varchar	N
branch	纳入缺陷追踪的分支名	varchar	N

表 3.18为Bug报告信息表，表名为bug_report。其主要作用是存储Bug 报告提交者提交的Bug 报告，以及仓库所有者导入的历史Bug报告。其中，id 为主键，唯一标识该Bug报告；repository_id 是外键，用于和该Bug 报告对应的仓库相关联。同时，还存储了该Bug报告的状态、标题、具体描述、创建的时间戳、提交者的email地址以及修改该Bug 报告所提交的commit。

表 3.18: Bug报告信息表

字段名	字段解释	字段类型	是否为空
id	Bug报告的唯一标识	int	N
repository_id	Bug报告所属的仓库的id	int	N
status	Bug报告的状态	int	N
summary	Bug报告的标题	varchar	N
description	Bug报告的具体描述	varchar	N
createTimestamp	Bug报告创建的时间戳	bigint	N
reporterEmail	Bug报告提交者的email	varchar	N
commit_sha	修复该Bug报告所实际修复代码的commmitSHA值	varchar	Y

表 3.19是本系统的定位任务信息表，表名为bug_localization_task。该表主要作用是存储开源仓库所有者在每个Bug报告中创建的定位任务信息。其中id

为定位任务的主键，是定位任务的唯一标识；bug_report_id 和repository_id 为外键，分别与该定位任务所属的Bug 报告和仓库关联。同时还存储采用的Bug 定位算法、运行状态、仓库所有者对该Bug定位任务的满意度评分。

表 3.19: 定位任务信息表

字段名	字段解释	字段类型	是否为空
id	定位任务的唯一标识	int	N
bug_report_id	定位任务所属的Bug报告id主键值	int	N
repository_id	定位任务所属的仓库的id	int	N
localization_algorithm	该定位任务使用的具体定位算法编号	int	N
status	定位任务的进行状态	int	N
score	仓库所有者对该定位任务的满意度	double	Y

表 3.20 是定位结果信息表，表名为bug_localization_task_result。他的主要作用是存储Bug定位任务的定位结果信息。其中id为主键，唯一标识定位结果的一条内容；bug_localizaiton_task_id为外键，与定位结果所属的定位任务相关联，同时，还存储了该条定位结果对应的源代码文件在GitHub 中的url以及它的可疑分数。

表 3.20: 定位结果信息表

字段名	字段解释	字段类型	是否为空
id	定位结果的唯一标识	int	N
bug_localization_task_id	定位结果对应的定位任务id主键值	int	N
source_code_url	可疑源代码文件的url	varchar	N
score	定位算法计算得出的该源代码文件的分数	double	N

3.5　本章小结

本章是对面向开源社区的Bug定位系统的需求分析与概要设计，包括系统的总体规划、面向的使用人群，提出开源社区开发者对该系统的基本需求，并

对系统的整体设计思路进行了概述，将整个系统的功能模块拆分为用户信息功能模块、缺陷追踪功能模块和Bug定位功能模块，并以系统用例图、用例描述表方式加以详细描述。同时，对系统数据库进行设计，给出每张表单所包含的所有属性，对每条属性的字段解释、类型和是否为空进行说明。

第四章　详细设计与实现

4.1　模块综述

在功能上，面向开源社区的Bug定位系统可以分为用户信息功能模块、缺陷追踪功能模块和Bug定位功能模块三部分。用户信息功能模块包括GitHub账号接入、用户开源仓库同步、邮箱关联、信誉积分以及新消息提示五个子功能；缺陷追踪功能模块可以细分为开源仓库缺陷追踪和Bug报告提交省察两部分；Bug定位功能模块可以划分为定位任务创建、实时任务进度获取、定位结果查看以及定位结果反馈。

4.2　用户信息功能模块

用户信息功能模块负责将开发者们在开源社区GitHub中的个人资料、开源仓库等信息同步到本系统中，便于用户为自己的开源仓库添加缺陷追踪功能，同时也为后续Bug定位服务提供源代码文件输入源信息。

4.2.1　GitHub账号接入

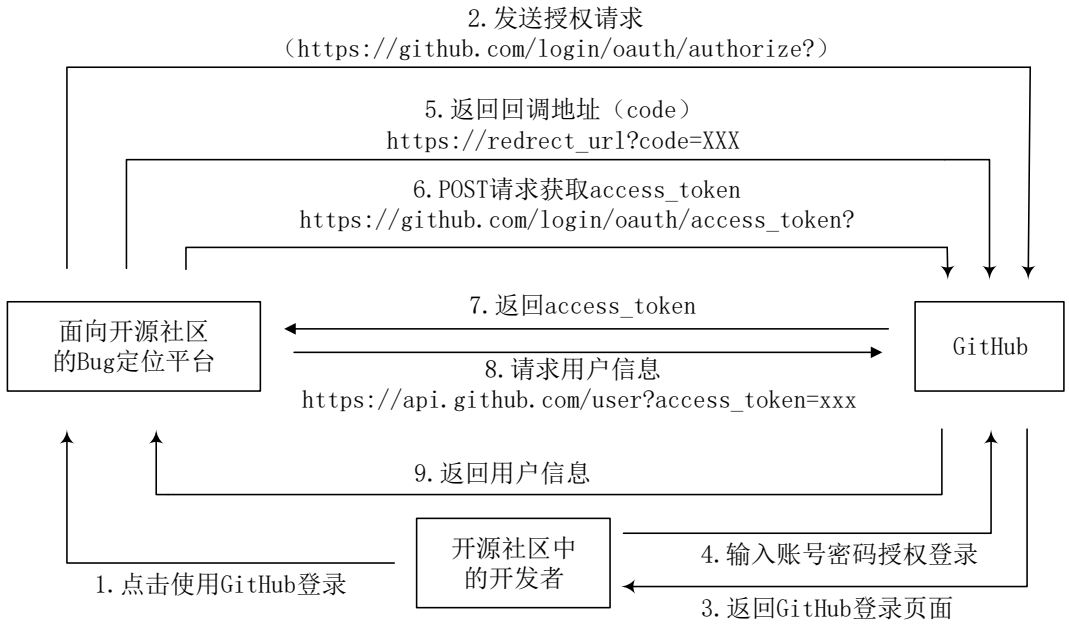

图 4.1: GitHub OAuth2.0授权流程

GitHub使用OAuth2.0方式开放用户授权。OAuth2.0是OAuth协议的延续版本，OAuth2.0关注客户端开发者的简易性。一种方式是通过组织在资源拥有者和HTTP服务商之间被批准的交互动作代表用户，另一种方式是允许第三方应用代表用户获取访问权限。如图 4.1所示，为GitHub 账号接入流程顺序图。

GitHub的OAuth首先通过页面重定位，将能够获取access_token的code 包含在回调地址中，系统解析回调页面的url，再根据code使用POST请求得到access_token，最后使用GitHub REST API返回的用户信息，进行后续相关操作。如图 4.2所示，描述了用户在本系统使用GitHub账号接入的顺序图。

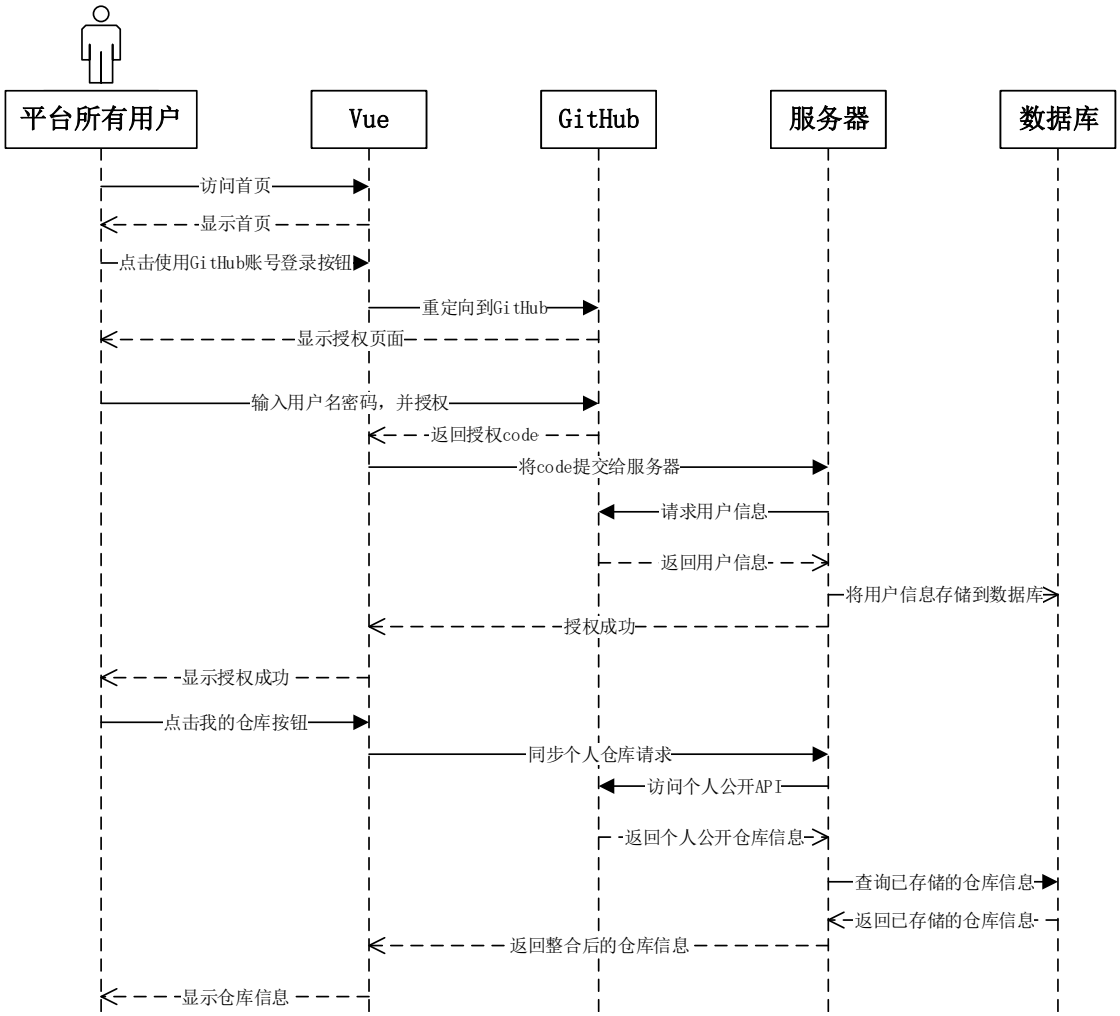

图 4.2: GitHub账号接入顺序图

此外，由于GitHub第三方登录和用户开源仓库同步等功能请求GitHub API都需要在服务端使用HTTP 协议，所以本系统基于Apache的httpcomponents工具包，封装了HTTP中的get和post请求，提供给服务器端与GitHub通信时使用。

其实现代码如图 4.3所示。上层模块调用该HTTP 客户端的httpGet或httpPost方法传入API接口，即可实现与GitHub的通信，代码12至13行为解析响应结果，最后将响应内容返回给上层模块。

Algorithm 1: 与 GitHub 通讯实现

Input: 通信连接的 URL
Output: GitHub 返回的响应内容

```
1  public class HttpClientUtil {
2      private static int successStatusCode = 200;
3      public static String httpGet(String url) {
4          String result = null;
5          CloseableHttpClient httpClient = HttpClients.createDefault();
6          HttpGet get = new HttpGet(url);
7          CloseableHttpResponse response = null;
8          try {
9              response = httpClient.execute(get);
10             if (response != null &&
11                 response.getStatusLine().getStatusCode() == successStatusCode) {
12                 HttpEntity entity = response.getEntity();
13                 result = entityToString(entity);
14             }
15             return result;
16         } catch (IOException e) {
17             //异常处理
18         }
19         return null;
20     }
21     public static String httpPost(String url, Map<String, String> map) {
22         //POST 具体实现
23     }
24 }
```

图 4.3: 与GitHub通讯的具体实现

4.2.2　用户开源仓库同步

用户开源仓库同步同样是通过GitHub的API得到。服务端将返回的json数据反序列化成Repository实体类，再将Repository实体类信息与已经存储在本系统数据库中的仓库信息对比关联。判断是否已经纳入系统的缺陷管理功能中，已经添加缺陷管理功能的仓库是否有新提交的Bug报告、新完成的定位任务、历史定位任务的准确率等指标，最后再将整合的结果统一返回给用户。个人公开仓库的RepositoryController如图 4.4所示，该控制类中的方法是整个仓库获取的业务流程的具体实现。

用户开源仓库同步后，服务端比对已经存储在系统数据库中的开源仓库信息，将这两种信息合并到RepositoryListVO中，返回给前端页面展示。如图4.5所示，为用户开源仓库同步页面实现效果图，其中1/1 代表用户认为Bug定位

Algorithm 2: 用户开源仓库同步具体实现

Input: 用户 ID，用户名
Output: 用户开源仓库信息

```
@RequestMapping(Router.Repository.GET_USER_RESPOSITORY)
public List<RepositoryVO> getUserRepository(int userId, String userName) {
    Gson gson = new Gson();
    String repositoriesInGitHubJson = repositoryService.getRepositoriesInGitHub(userName);
    List<RepositoryVO> repositoriesInGitHub = gson.fromJson(repositoriesInGitHubJson
        , new TypeToken<List<RepositoryVO>>() {}.getType());
    List<Repository> managedRepository = repositoryService
        .getManagedRepositoriesByUserId(userId);
    if (managedRepository.size() == 0) {
        return repositoriesInGitHub;
    } else {
        for (RepositoryVO repositoryVO : repositoriesInGitHub) {
            for (Repository repository : managedRepository) {
                if (repositoryVO.isSameRepository(repository)) {
                    repositoryVO.setManaged(true);
                    repositoryVO.setLocalizationCorrectlyCount(
                        bugLocalizationTaskService
                        .getLocalizationCorrectlyCountByRepositoryId(
                        repository.getId()));
                    repositoryVO.setBugLocalizationCount(
                        bugLocalizationTaskService
                        .getFinishedLocalizationTaskCountByRepsitoryId(
                        repository.getId()));
                    repositoryVO.setNewbugLocalizationCount(
                        bugLocalizationTaskService
                        .getNewFinishedLocalizationTaskCountByRepsitoryId(
                        repository.getId()));
                    repositoryVO.setBugReportCount(bugReportService
                        .getBugReportCountByRepositoryId(repository.getId()));
                    repositoryVO.setNewbugReportCount
                        (bugReportService
                        .getNewBugReportByRepository(repository.getId()));
                }
            }
        }
    }
    return repositoriesInGitHub;
}
```

图 4.4: 用户开源仓库同步具体实现

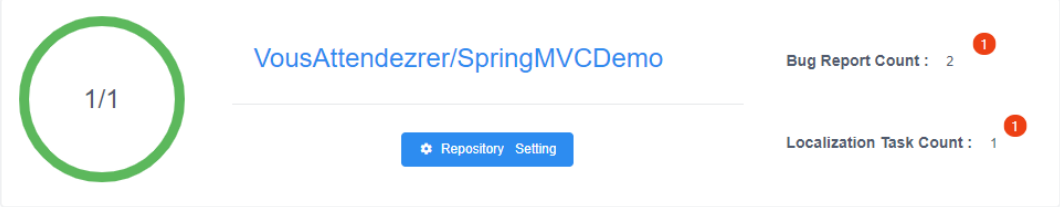

图 4.5: 用户开源仓库同步页面实现效果图

结果满意的任务数目/该开源仓库已经执行过的Bug 定位任务；VousAttendezrer/SpringMVCDemo为该开源仓库的全名；Bug Report Count: 2则是该仓库总共拥有的Bug 报告数，徽标1表示新收到的Bug报告数目；Localization Task Count: 1 代表该仓库已执行Bug 定位任务数目，徽标1 表示新完成的Bug定位任务数目。

4.2.3　邮箱关联

考虑到用户在使用我们的系统前，是使用其他缺陷追踪系统管理Bug报告的，为了能够区分出这些历史Bug报告的提交者与使用我们系统提交Bug报告的用户是否为同一个人，我们提供了邮箱关联功能。即在系统集成的Bug定位的方法中，我们将使用同一个email的人视为同一个事实人。如图 4.6所示，为用户邮箱关联的顺序图。

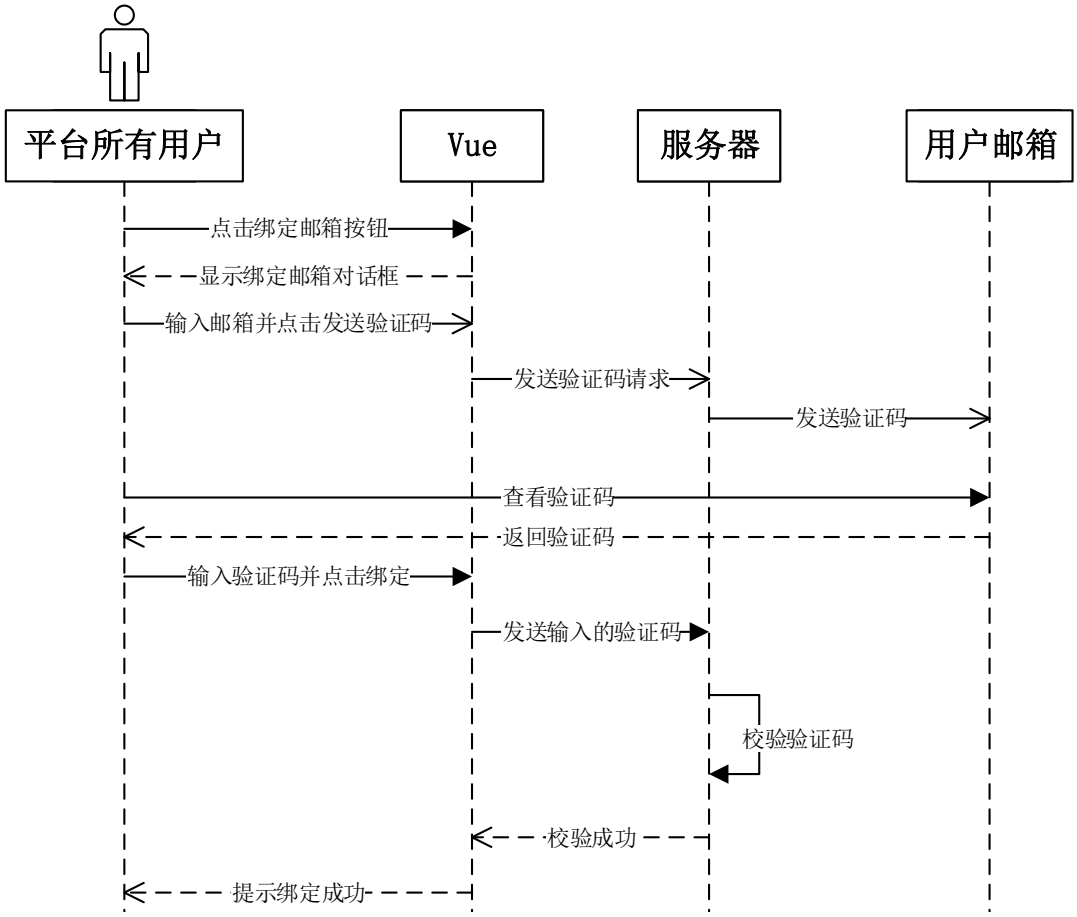

图 4.6: 用户邮箱关联的顺序图

邮箱关联功能的核心是邮件向用户的的邮箱发送邮件。本系统使用JavaMail实现该功能，MailUtil 实现Runnable类，保证了同时为多个用户提供邮箱关联

功能。主要过程分为三步：第一步，创建会话连接对象javax.mail.Session；第二部，创建邮件消息对象javax.mail.Message；最后一步，根据用户输入的邮箱地址发送邮件。使用JavaMail实现邮箱关联的具体代码如图 4.7 所示。

Algorithm 3: 用户邮箱关联功能

Input: 邮件服务器地址，验证码
Output: 否发送邮箱关联邮件成功

```
1  public class MailUtil implements Runnable {
2      private from = “njufangwenqiang@smail.nju.edu.cn”;
3      public boolean sendMail(String host, int code) {
4          Properties properties = System.getProperties();// 获取系统属性
5          properties.setProperty("mail.smtp.host", host);// 设置邮件服务器
6          properties.setProperty("mail.smtp.auth", "true");// 打开认证
7          try {
8              MailSSLSocketFactory sf = new MailSSLSocketFactory();
9              sf.setTrustAllHosts(true);
10             properties.put("mail.smtp.ssl.enable", "true");
11             properties.put("mail.smtp.ssl.socketFactory", sf);
12             // 1.获取默认 session 对象
13             Session session = Session.getDefaultInstance(properties, new Authenticator() {
14                 public PasswordAuthentication getPasswordAuthentication() {
15                     return new PasswordAuthentication(from, "xxx"); // 发件人邮箱账号、授权码
16                 }
17             });
18             // 2.创建邮件对象
19             Message message = new MimeMessage(session);
20             // 2.1 设置发件人
21             message.setFrom(new InternetAddress(from));
22             // 2.2 设置接收人
23             message.addRecipient(Message.RecipientType.TO, new InternetAddress(email));
24             // 2.3 设置邮件主题
25             message.setSubject("面向开源社区的 Bug 定位平台邮箱关联");
26             // 2.4 设置邮件内容
27             String content = "";
28             message.setContent(content, "text/html;charset=UTF-8");
29             // 3.发送邮件
30             Transport.send(message);
31              retrun true;
32         } catch (Exception e) {
33             //异常处理
34         }
35     }
36 }
```

图 4.7: 邮箱关联功能具体实现

4.2.4　信誉积分

为了防止有人恶意向他人的开源仓库提交Bug报告，影响系统的正常运行秩序，每个用户都会拥有一个信誉积分。一旦提交的Bug报告被该开源仓库的所有者举报为恶意Bug报告，则会扣除该提交者若干点信誉积分。当信誉积分

低于一定阈值时，则会被限制使用Bug报告提交功能。而使用系统的其他功能，如为自己的开源仓库添加缺陷追踪功能，新建Bug定位任务并运行即可增加信誉积分。如图 4.8所示，为信誉积分功能顺序图。

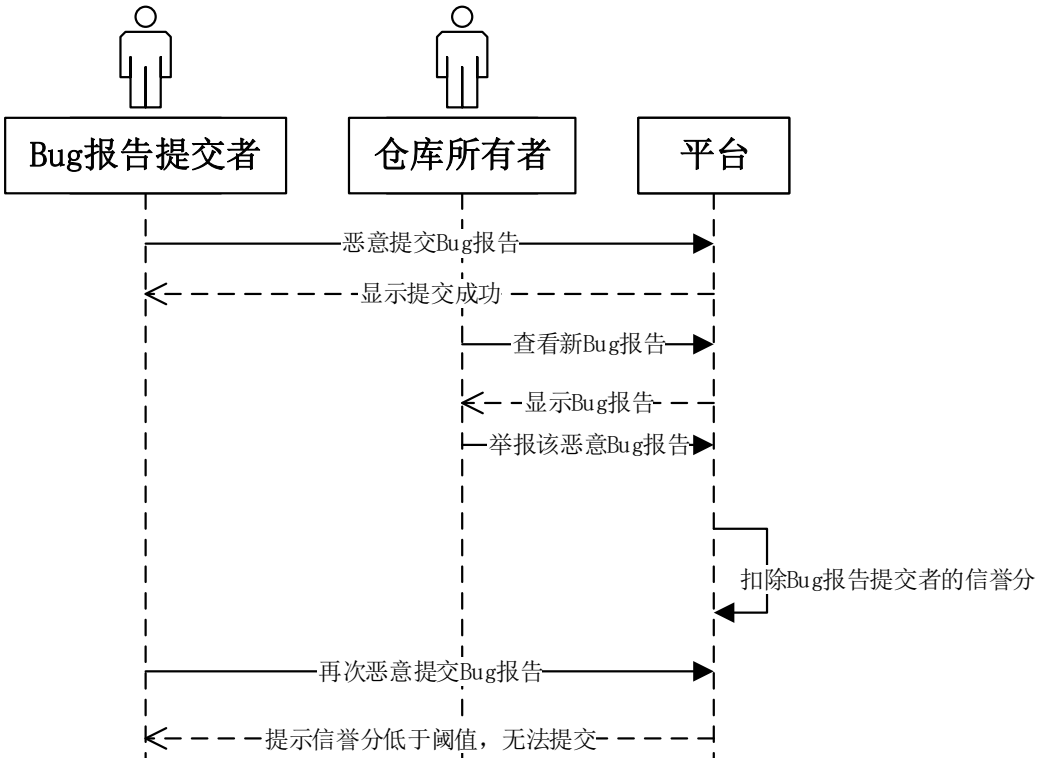

图 4.8: 信誉积分功能顺序图

在用户的个人设置页面，可以查看自己的信誉积分折线图。如图 4.9所示，该用户在2019年3月17日的积分为13分，点击该点可以继续查看3月17日当日改变积分的具体事项。

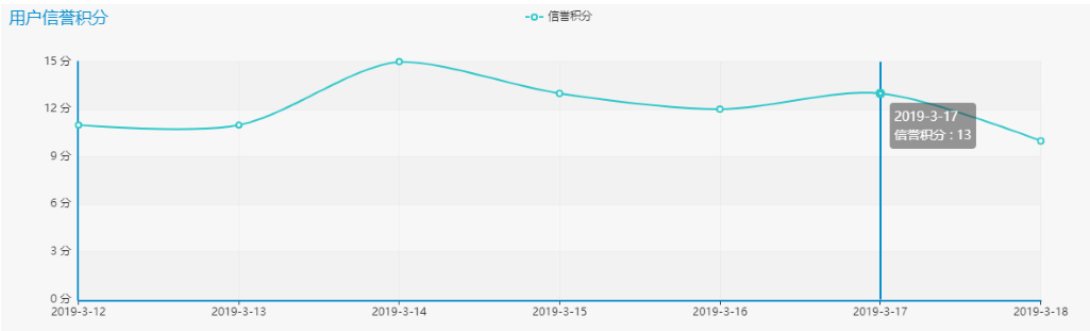

图 4.9: 信誉积分折线图

4.2.5　消息提示

Bug报告的更新涉及到整个缺陷跟踪流程的进展，为了使用户能够及时了解到每个Bug报告和定位任务的进展，系统提供了消息提示功能。若用户设置了接收消息提示，当Bug报告的状态、评论或者相关定位任务有了新的进展，将以邮件和站内信的形式告知仓库所有者，以便其能够及时知会。如图 4.10所示，为消息提示功能的顺序图。

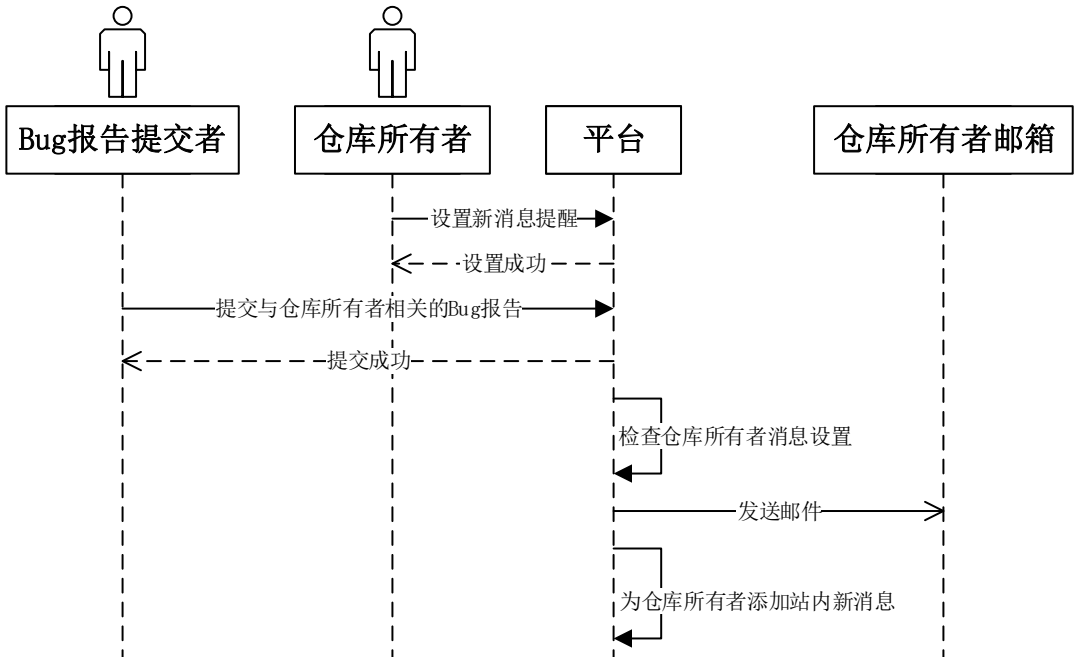

图 4.10: 消息提示顺序图

4.3　缺陷追踪功能模块

在该功能模块中，仓库所有者可以为自己的仓库添加缺陷追踪功能、选择缺陷追踪的分支、导入历史Bug 报告，Bug报告提交者可以提交Bug报告，两种角色都可以获取Bug报告列表、查看Bug报告的详细信息。

4.3.1　开源仓库缺陷追踪及历史Bug报告导入

开源仓库缺陷追踪是本系统的核心功能之一，同时也是为Bug定位这一核心主题服务的。仓库所有者可以根据自身实际需求，为自己的公共仓库添加缺陷追踪功能，更方便地管理Bug报告和使用Bug定位技术。对于很多GitHub上的项目而言，当一个大版本发布后，都会保留当前版本分支，并新建另一条

分支以供下个版本开发使用，而不是一直简单的使用master分支进行，所以需要仓库所有者选择具体使用哪一条分支进行缺陷跟踪和Bug定位。为方便仓库所有者将存在其他缺陷追踪系统中的Bug报告导入本系统，提升后续Bug 定位结果的准确度，还为仓库所有者提供一个可以导入历史Bug 报告的途径。如图4.11所示，为仓库所有者为自己的仓库添加缺陷追踪功能、选择缺陷追踪的分支、导入历史Bug报告的顺序图。

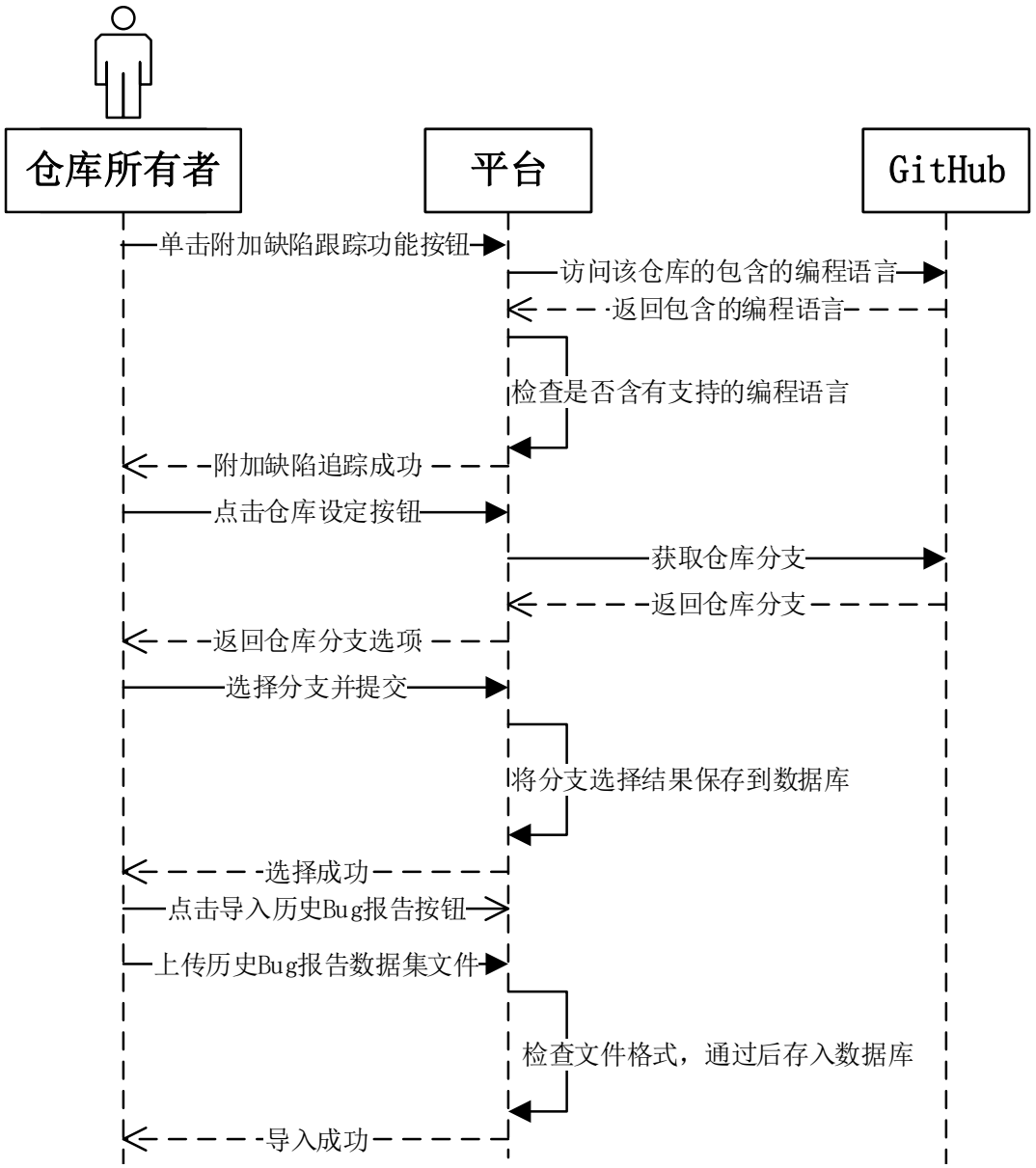

图 4.11: 开源仓库缺陷追踪及历史Bug报告导入顺序图

仓库所有者在将自己的开源仓库纳入缺陷追踪后，可以点击仓库的设置按钮，选择对哪条分支进行缺陷管理，设置是否接收他人提交的新的Bug报告。同时可以下载历史Bug报告模板，按照模板中的格式修改后上传即可导入历史Bug报告。历史Bug报告导入模板为json文件，包含一组历史Bug报告的标题summary、具体描述description、创建时间戳created_timestamp、提交者邮箱reporter、修复该Bug的时间戳以及commit信息。为了减轻服务器压力，本系统选择使用JS 在浏览器端对历史Bug报告文件进行校验，具体实现如图 4.12 所示。

Algorithm 4: 浏览器端对历史 Bug 报告格式校验实现

Input: 仓库所有者提交的历史 Bug 报告 Json 文件
Output: 校验成功则上传到服务端，否则提示格式失败

```
1  //读取历史 Bug 报告 JSON 文件
2  function readTextFile(file, callback) {
3      var rawFile = new XMLHttpRequest();
4      rawFile.overrideMimeType("application/json");
5      rawFile.open("GET", file, true);
6      rawFile.onreadystatechange = function() {
7          if (rawFile.readyState === 4 && rawFile.status == "200") {
8              callback(rawFile.responseText);
9          }
10     }
11     rawFile.send(null);
12 }
13
14 //校验 JSON 文件格式
15 readTextFile("/Users/Documents/workspace/test.json", function(text){
16 if (typeof text == 'string') {
17     try {
18         var obj=JSON.parse(text);
19         if(typeof obj == 'object' && obj ){
20             //格式校验通过
21         }else{
22             //格式校验不通过
23         }
24     } catch(e) {
25         //格式校验不通过的异常处理
26     }
27 }});
```

图 4.12: 历史Bug报告Json文件校验

4.3.2　Bug报告提交

当开发者将自己的仓库纳入到缺陷追踪功能中后，设置允许他人提交新的Bug 报告。那么使用其开源项目的用户在遇到Bug时，便可以通过本系统提交Bug 报告给开发者，开发者则能够及时获悉自己开源项目的健康状况，确定是否发现了新的Bug，这是缺陷追踪的基础。提交Bug报告前，系统会对提交者的邮箱关联和信誉积分状况进行判定，若未绑定邮箱或信誉积分不达标，则不允许提交Bug报告。如图 4.13所示，为Bug报告提交的顺序图。

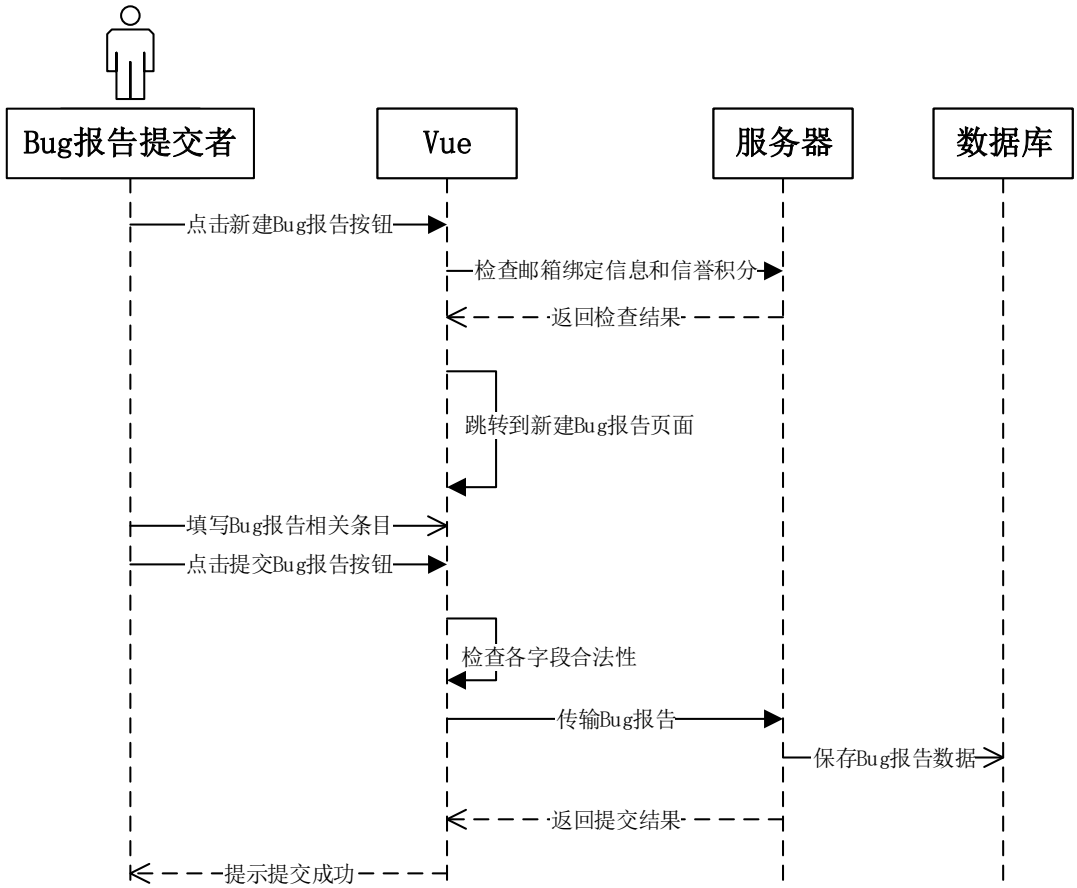

图 4.13: Bug报告提交顺序图

4.3.3　Bug报告列表与详细信息获取

用户需要对自己所属仓库的相关Bug报告有一个概览，同时也需要查看自己曾经提交过的所有Bug报告；仓库所有者可以根据Bug报告信息了解Bug的具体内容，反过来Bug提交者又可以根据Bug报告的状态知晓Bug解决的进展情况。Bug报告还为双方提供了一个交流的方式，是整个缺陷追踪模块中最重要的功

能。如图 4.14所示，为Bug报告列表与详细信息获取功能的顺序图。

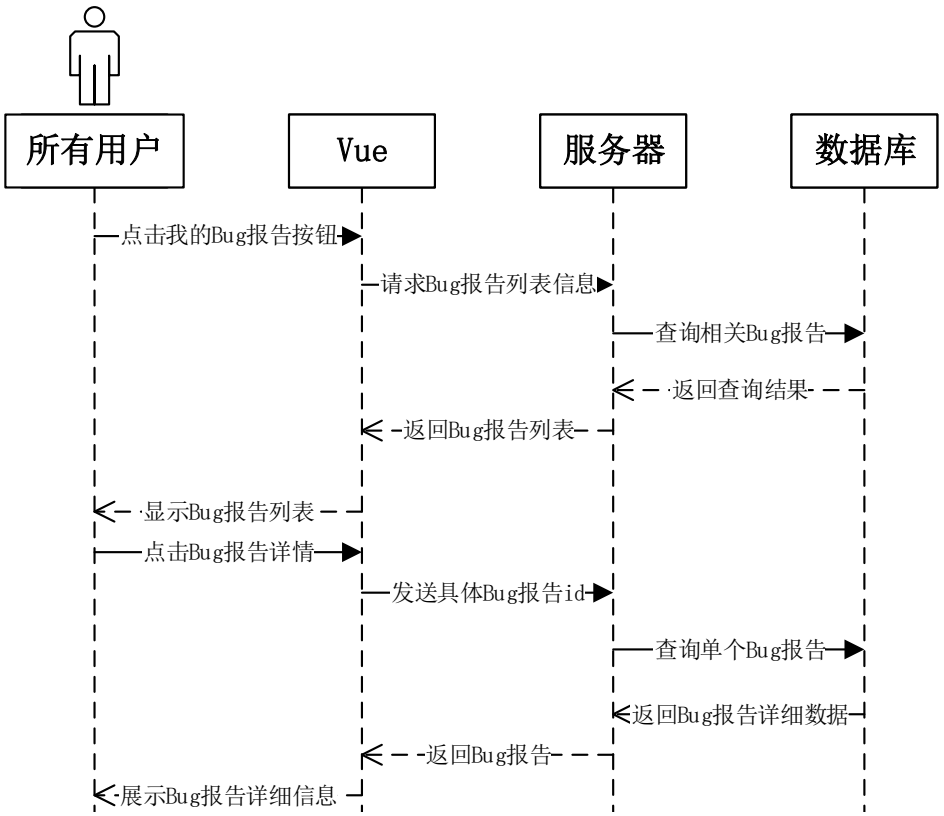

图 4.14: Bug报告列表与详细信息获取顺序图

Bug报告详细页面包含Bug报告的ID、标题信息Summary、所属于开源仓库全名Repository、该Bug报告提交的时间Reported Time、该Bug报告现今的状态Status、Bug报告提交者的信息（用户名及绑定邮箱）Reporter 以及对Bug 的详细描述内容Description。其中，ID全系统唯一，一个确定的ID值对应存储在本系统中唯一一个Bug报告，而不是只在同一个开源仓库下才保持唯一。点击所属开源仓库的全名链接，可以跳转到该开源仓库对应的GitHub页面。当Bug报告的状态为VERIFIED时，会显示“新建Bug定位任务按钮”。如图 4.15 所示，为Bug 报告详细信息页面的实现效果图。

4.4　Bug定位功能模块

4.4.1　Bug定位任务创建及进度实时获取

Bug定位任务功能模块为本系统的核心，其中创建Bug定位任务功能又是该功能模块的基础，它为仓库所有者使用Bug定位方法提供了入口，对于状

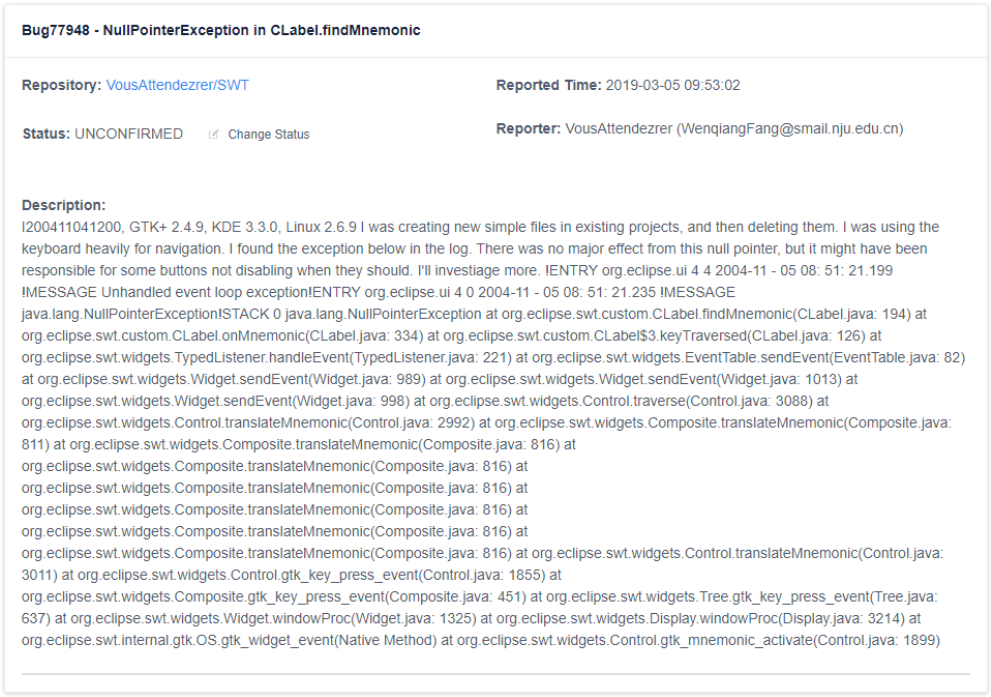

图 4.15: 详细Bug报告实现效果图

态Stauts为CONFIRMED的Bug报告，点击新建Bug定位任务按钮即可一次性运行系统集成的三种Bug定位方法。Bug定位任务的速度受克隆GitHub仓库的网速、历史Bug报告数据集的数量、仓库的规模以及不同的Bug定位算法影响，可能存在耗时较久的问题，所以为Bug定位任务提供进度展示功能，减少用户等待定位任务完成产生的焦虑。

如图 4.16所示，为Bug定位任务创建及进度实时获取流程图。仓库所有者点击“创建Bug定位任务”，页面展开创建定位任务抽屉，仓库所有者再根据自身需求勾选定位算法并提交创建。前端页面将Bug定位任务传递给服务端，服务端根据请求创建相应定位算法，并返回创建结果响应，页面提示仓库所有者创建结果。接着，Vue每隔一段时间向服务端发送请求，查询各Bug定位任务的进展，服务端将查询结果返回，页面根据响应更新Bug定位任务状态，直至所有Bug定位任务的状态都变为已完成。

4.4.2　Bug定位任务计算与结果查看

本系统实现并集成三种Bug定位方法。分别为BugLocator、BLUiR+和Ama-Lgam+。这三种定位算法都需要分别对Bug报告和源代码文件的信息进行文本预处理，本系统选取开源项目Lucene 的Analyzer 工具包实现所需文本预处理

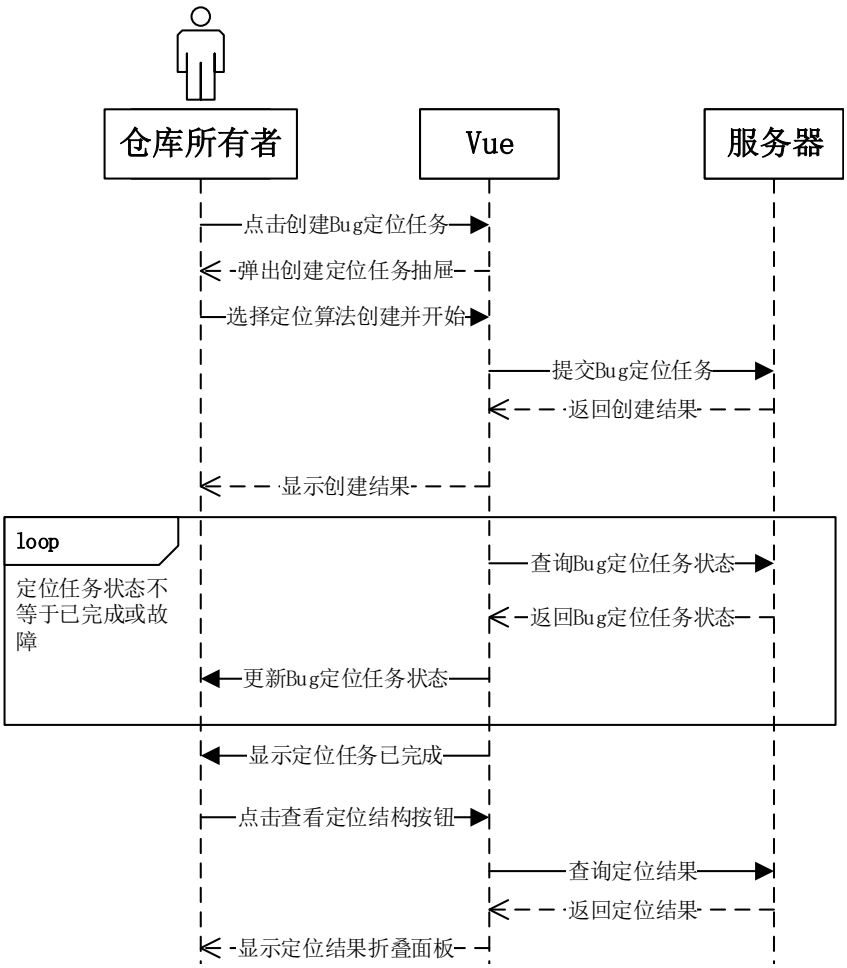

图 4.16: Bug定位任务创建及进度实时获取流程图

技术。如图 4.17 所示，Lucene 分析器包括一个分词器(Tokenizer) 和多个过滤器(TokenFilter)。使用分词器Tokenizer对Bug报告和源代码字符流进行切割，过滤器TokenFilter 根据字符流切割结果进行筛选。Analyzer 在tokenStream 方法中先使用Tokenizer，接着按照文本预处理顺序要求使用多个TokenFilter进行处理。

本系统基于Lucene分析器，重写TokenStream的createComponents方法，实现了标点符号去除、停用词去除、复合词切割和词干提取等文本预处理技术，满足各Bug定位方法需求。如图 4.18所示，为复合词切割具体代码实现。

BugLocator

对于较为久远的开源项目而言，存在一些仓库只保留了历史Bug记录以及对应的所修复的代码文件记录。对于这种类型的仓库，我们选取了BugLocator这种定位算法法进行实现与集成。BugLocator总共分为两个部分进行计算，第一个部分是将新提交并已被仓库所有者确认有效的Bug报告与原有已经修复

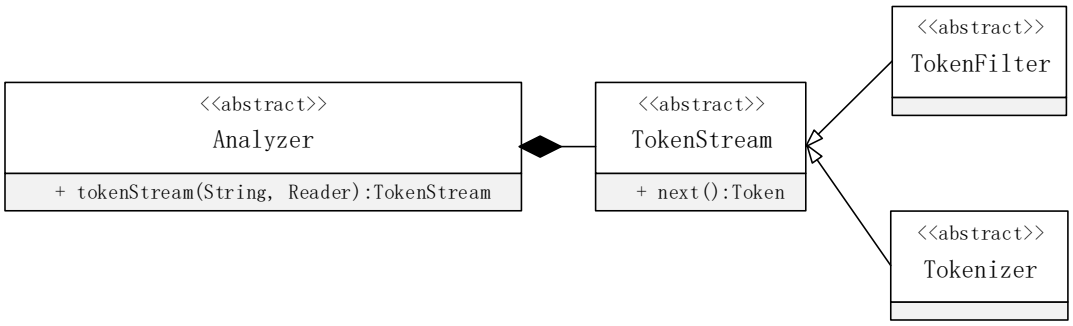

图 4.17: 文本预处理实现类图

Algorithm 5: 文本预处理重写 TokenStream 的 createComponents 方法

```
1  protected TokenStreamComponents createComponents(String fieldName) {
2      Tokenizer tokenizer = new CharTokenizer() {
3          @Override
4          protected boolean isTokenChar(int c) {
5              return Character.isLetter(c);
6          }
7      };
8      TokenFilter tokenFilter = new WordDelimiterGraphFilter(tokenizer,
9          (GENERATE_WORD_PARTS | SPLIT_ON_CASE_CHANGE),
10         CharArraySet.EMPTY_SET);
11     return new TokenStreamComponents(tokenizer, tokenFilter);
12 }
```

图 4.18: 文本预处理复合词切分代码实现图

的Bug 报告记录进行相似度匹配，计算新旧Bug 报告的文本余弦相似度，赋给以往修复的源代码文件一个相似度分值SimiScore；第二个部分是对现版本的源代码文件生成索引，将新提交的Bug 报告作为查询条件，辅以每个源代码文件的大小，逐一计算每个源代码文件的向量空间模型分值rVSMScore。最后将SimiScore 和rVSMScore 按一定权重求和，得出最终的定位结果，即每个源代码文件与该提交的Bug 报告的相关程度分数。图 4.19 为BugLocator 进行Bug定位的整体实现框架。

系统声明并初始化BugLocaotr 类，传入bugReportId，接着调用BugLocator 类的runBugLocator() 方法，启动该定位算法。BugLocator 类调用SimiScoreCalculator类的getSimiScore方法，该方法根据传入的BugId，通过BugReportDao

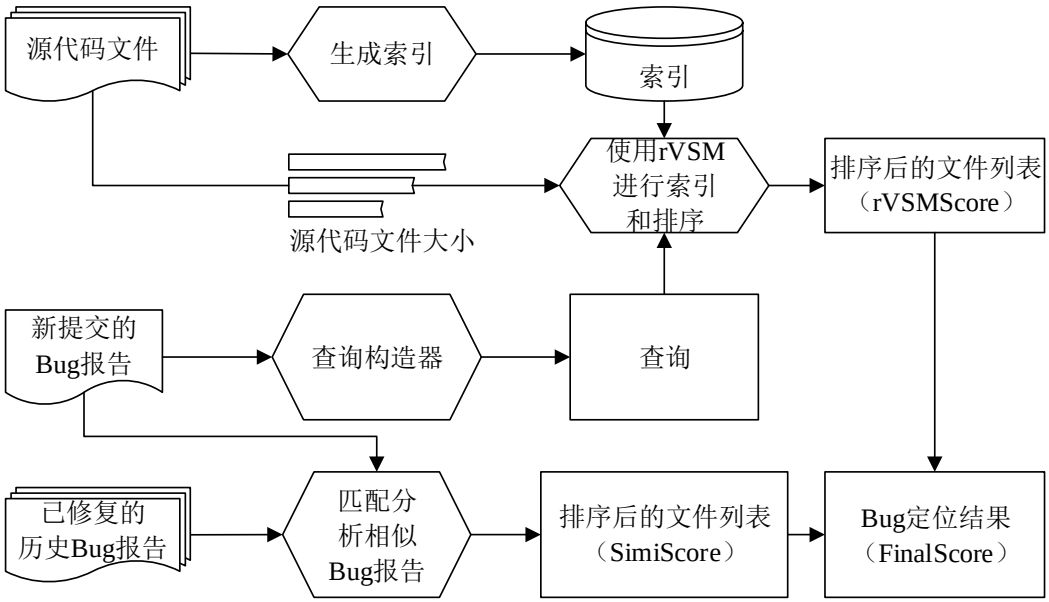

图 4.19: BugLocator定位方法整体框架

读取数据库中该Bug报告的所有信息。根据读取的Bug报告中repository_id、createdTimeStamp字段再通过BugReportDao从数据库中读取该开源仓库所有已被修复历史Bug报告的summary、description、fixedFiles信息。接着，getSimiScore 方法将获取的所有Bug 报告经文本预处理后转换为词向量。调用SimilarityCalculator 的calculateSimilary 方法，计算新提交的Bug 报告与每个历史Bug 报告的余弦相似度。SimiScoreCalculator 将SimilarityCalculator 返回的余弦相似度分数，作为各历史Bug报告所修复源代码文件的SimiScore。若一个源代码文件具有多个SimiScore，则取其平均值。计算完SimiScore后，BugLocator 类接着调用RVSMScoreCalculator的getrVSMScore方法，该方法通过SourceCodeFileReader将该开源仓库节点checkout到Bug报告提交时间点，读取所有源代码文件的内容。接着，RVSMScoreCalculator将读取得到的各源代码文件进行文本预处理并向量化，使用SimilarityCalculator的calculateSimilarity方法逐个计算代表每个源代码文件的向量与Bug报告向量的余弦相似度。再通过SourceCodeFileReader的getSourceCodeTermCount 方法获取每个源代码文件的长度，利用该长度作为权重对每个源代码文件与Bug报告的余弦相似度加权，得出各源代码文件的rVSMScore。最后，BugLocator的calculateFinScore 方法遍历返回的SimiScore和rVSMScroe 两组Map 数据，按照权重2：8对两组Map 数据加权求和，得出最终的localizationResult。再将该定位结果返回给BugLocalizationTaskService 加以保存。如图 4.20所示，为BugLocator定位算法实现类图。

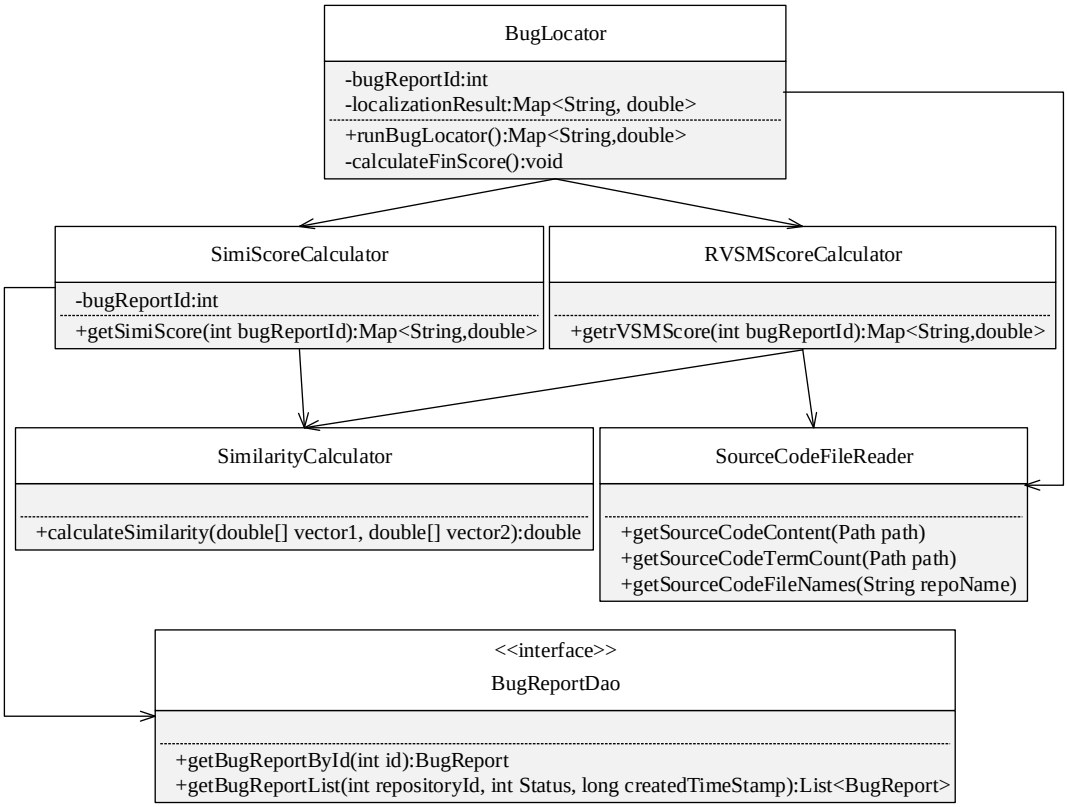

图 4.20: BugLocator定位方法实现类图

BLUiR+

随着开源社区的日渐繁荣，新的开源项目的数量愈加庞大。考虑到新的开源项目是几乎不存在历史Bug报告这一信息，无法使用上一种BugLocator方法来进行Bug定位，我们添加了BLUiR+这种Bug定位方法。

BLUiR+并不需要用到历史Bug报告信息。如图 4.21 所示，它将当前版本的源代码文件转换成AST抽象语法树，提取出每个源代码文件的类名、方法名、变量名以及注释，分别进行文本处理将它们转换成结构化文档并作为索引；同时再将新提交的Bug报告标题和Bug描述信息也分别进行文本处理，作为查询条件。对源代码文件中的四种信息和当前Bug报告的两种信息，两两计算它们之间的文本余弦相似度后求和，便可以得出每个源代码文件与该提交的Bug报告的相关程度分数。

系统声明并初始化BLUiR类，保存新提交的Bug 报告ID。通过调用runBLUiR 方法启动该定位算法。BLUiR 类通过BugReportDao 提供的查询接口，根据Bug 报告ID 从数据库中获取该新提交的Bug 报告内容，抽取其中的summary 和description 字段。接着，BLUiR 类读取该开源仓库下所有源代码文件，对每

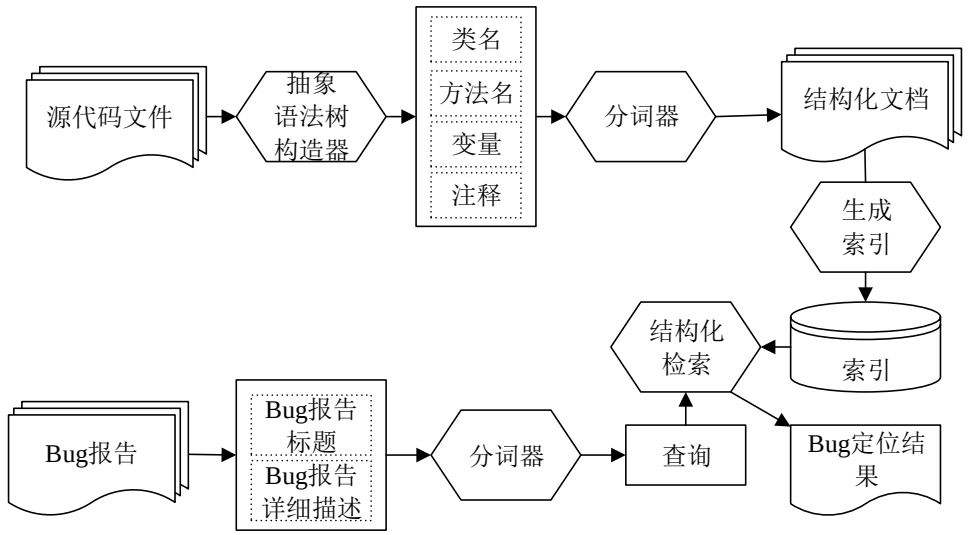

图 4.21: BLUiR+定位方法整体框架

个源代码文件逐个调用抽象语法树解析器ASTConverter，通过getClassNames 方法获取类名、getMethodsNames 获取方法名、通过getVariableNames 获取变量名、通过getComments 获取注释信息。使用TextPreprocess 类提供的process 方法，分别对Bug 报告和源代码文件的各个字段进行文本预处理。其中，StopWordAnalyzer 类用于去除停用词，CompoundWordsAnalyzer 用去切分驼峰词，PorterStemmberAnalyzer 用于词根还原。然后，BLUiR 类使用Vectorizer 类提供的generateVector 方法，将经过文本预处理后的4种源代码文件信息和2种Bug 报告信息转换为空间向量。最后，将该Bug 报告的两条空间向量，逐个与每份源代码文件的四条空间向量交叉运算，求两两之间余弦相似度，8 个余弦相似度求和结果即为每份源代码文件的Bug 定位分数。BLUiR 类计算完成后，将结果保存在localizationResultMap 中，供BugLocalizationTaskService 类调用和存储。图 4.22 为BLUiR+定位算法的实现类图。

AmaLgam+

很多大型的开源项目，从创建伊始便使用了十分规范的Bug报告管理方式，每个Bug报告都记录了较为丰富的信息，如Bug报告的创建时间和最新变更时间，每个Bug报告提交者的详细信息，更为准确的Bug报告描述信息。考虑到充分利用这个信息，以提升Bug定位的准确度，选取AmaLgam+作为第三个Bug 定位技术进行实现和集成。

AmaLgam+ 总共分为了五个组件。修复Bug可能会导致引入一些新的Bug，于是Version History组件使用正则表达式去匹配了过去一段时间内的git log 中的那些用于修复bug 而提交的commit 记录；Similar Report 组件则借鉴了BugLocator

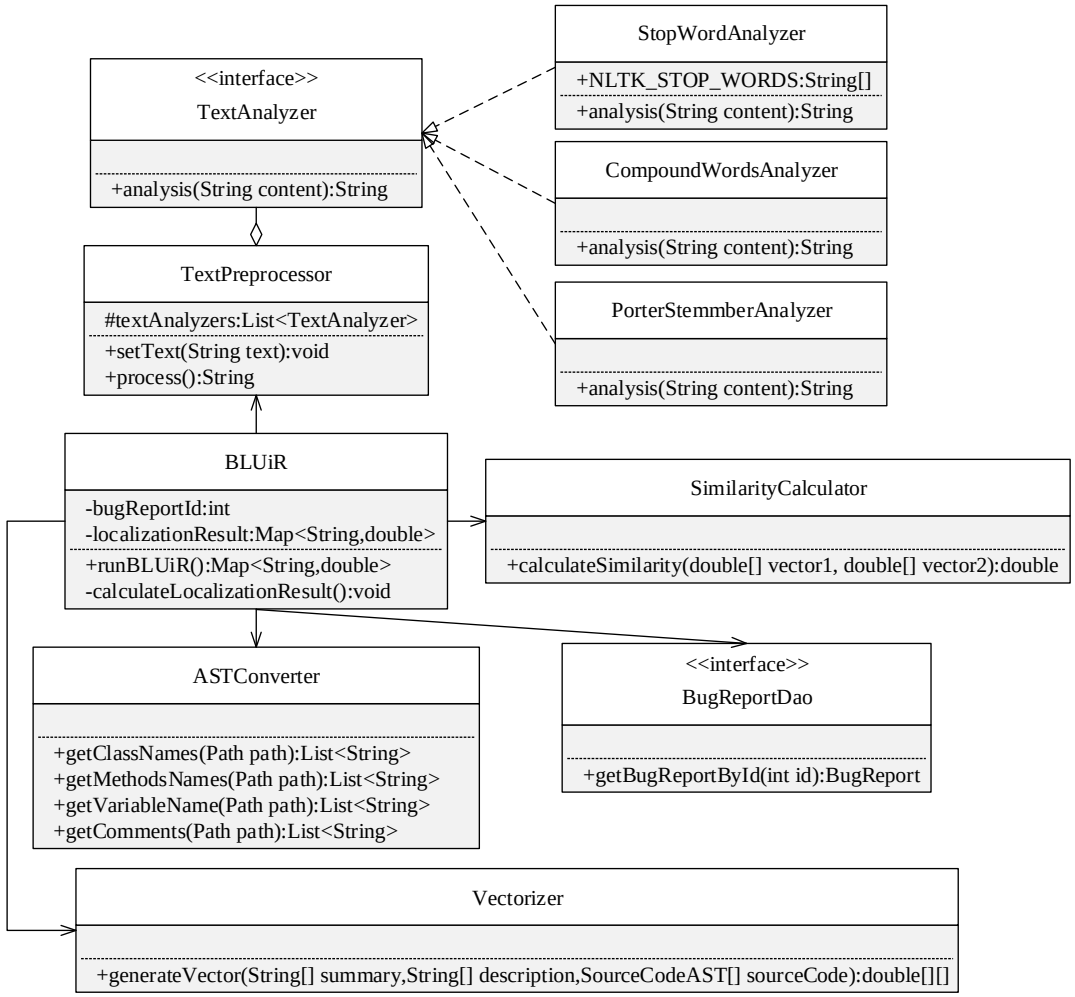

图 4.22: BLUiR+定位方法实现类图

中计算SimiScore的部分，用于衡量提交的Bug报告与以往历史Bug报告的相似程度；Structure组件则完全采用的BLUiR+的方法来计算当前Bug报告与每个源代码文件的关联程度；此类大型开源项目的Bug报告中，Bug 描述部分通常都会包含有程序出错时的堆栈信息，Stack Trace 组件则将这些堆栈信息提取出来，对出现在堆栈信息中的源代码文件根据它们的出栈顺序对其赋予权重；这种级别的开源项目往往不会只是单个开发者，都会有自己的测试人员，而测试人员发现Bug 后也是通过提交Bug报告的方式告知开发人员，所以对于同一个Bug报告提交者，可以认为提交的Bug 报告在功能模块上都是相关的，Reporter Information组件便是将这些同功能模块的源代码文件标识出来，增加它们的可疑性。最后，将五个组件所计算的指标加权求和，得出最终的Bug定位结果。如图 4.23所示，为使用AmaLgam+进行Bug定位的整体框架。

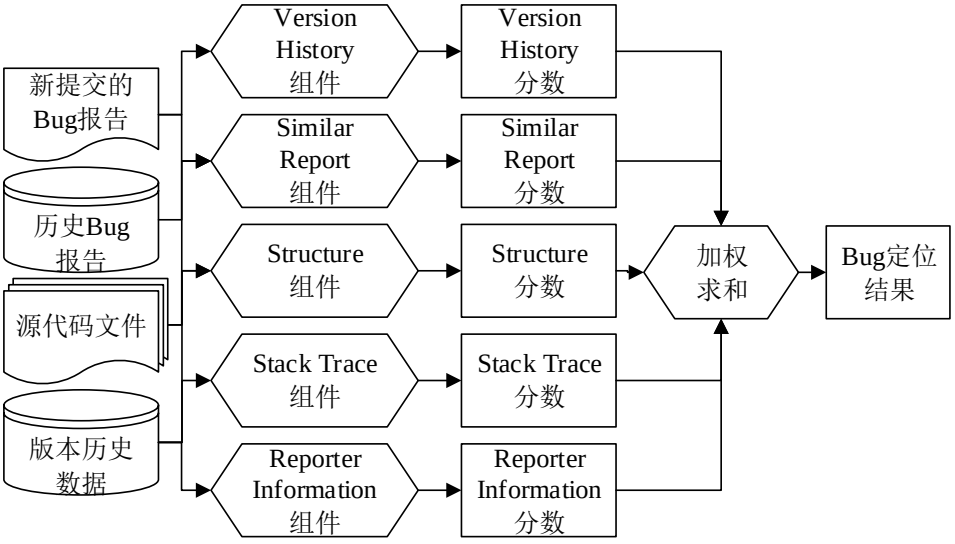

图 4.23: AmaLgam+定位方法整体框架

图4.24为AmaLgam+定位算法的实现类图。系统声明并初始化AmaLgam类，传入需要运行该定位算法的Bug报告ID，通过runAmaLgam 方法启动定位任务。第一部分，AmaLgam调用VersionHistoryCalculator类计算版本历史分数，VersionHistoryCalculator 类使用readGitLog方法，读取此Bug报告所属仓库的git log信息，将PAST_DAYS天内的所有commit记录解析出来，本系统将PAST_DAYS值设置为15天，接着通过findCommitForFixBug方法使用正则表达式“(.*fix.*)|(.*bug.*)”匹配近15天为修复Bug而修改的源代码文件，getVersionHistoryScore 赋予这些修改的源代码文件VersionHistoryScore，离Bug报告提交时间越近，则分数越高。第二部分，AmaLgam通过SimilarReportCalculator 调用BugLocator用于计算SimiScore的类SimiScoreCalculator，给出历史Bug报告所修复的源代码文件SimilarReportScore。第三部分，AmaLgam类调用StructureCalculator，使用BLUiR算法计算bug报告、源代码文件信息之间的余弦相似度，得出各源代码文件的StructureScore。第四，AmaLgam调用StackTraceCalculator获取StackTrace-Score，getStackTraceScore首先根据Bug报告ID，获取该Bug报告description信息，使用findStackTrace方法通过正则表达式“at [~()]+([~()]+.java:[0－9]*)”找出堆栈信息，将堆栈深度取倒数，赋给每个源代码文件StackTraceScore；第五，ReporterInformationCalculator 根据Bug报告ID查询提交者的email 字段，再使用getHistorySubmitBugReportsId得到该提交者曾经提交的所有Bug报告ID，调用getFixedSourceCodePackages 将这些历史Bug报告涉及的所有包名整理出来，ReporterInformationCalculator再将这些包下的源代码文件的ReporterInformation-Score赋值为1。由calculateLocalizaitonResult 汇总Map数据，得出定位结果。

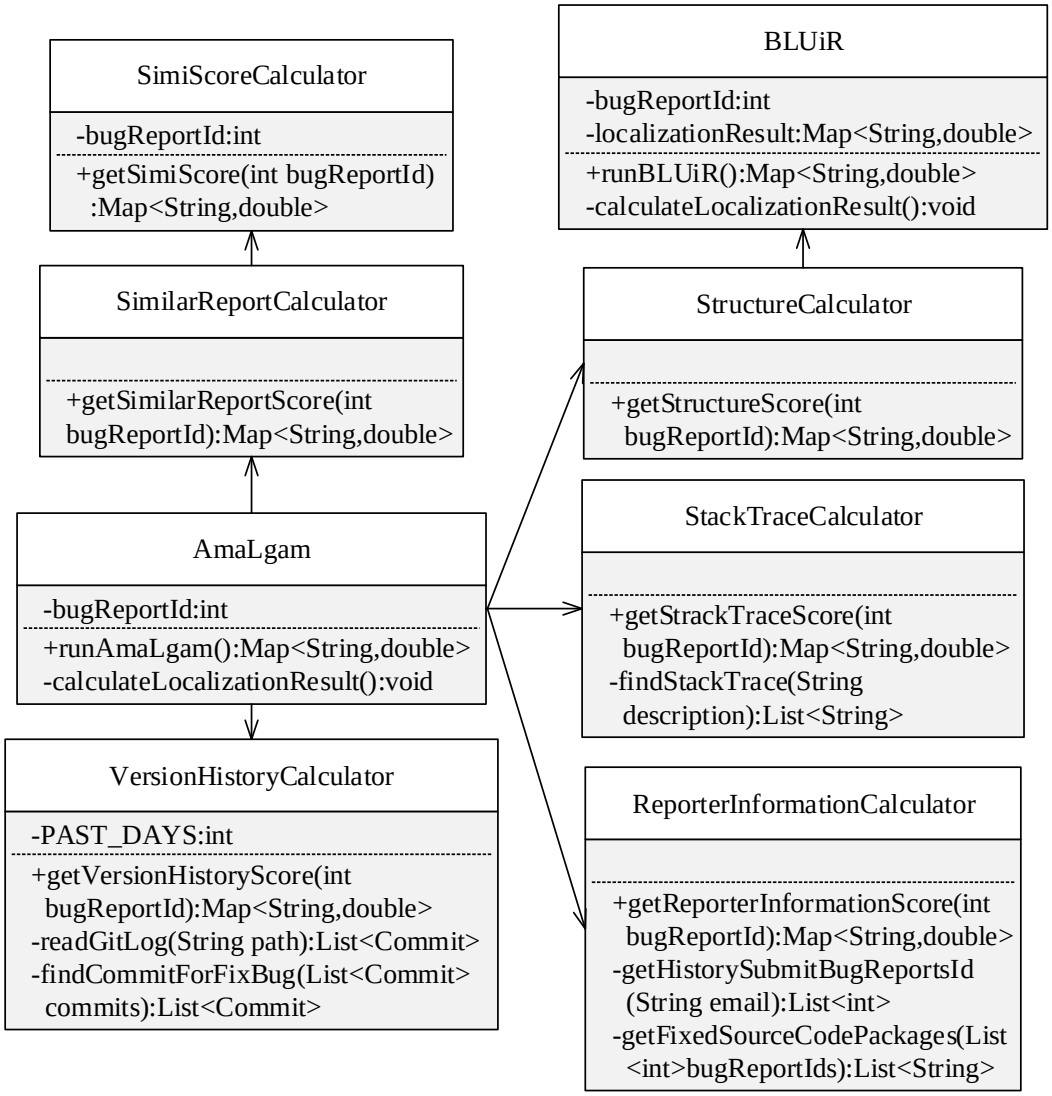

图 4.24: AmaLgam+定位方法实现类图

三种定位方法最终的定位结果将对源代码文件依据各计算得分进行降序排列，选取排名靠前的源代码文件作为定位的结果，根据各定位方法分别展现给仓库所有者再进行人工验证。

图 4.25为Bug定位结果实现效果图。每张定位结果表的第一列为源代码文件名，第二列对应每个定位算法根据Bug报告为每个源代码文件计算得出的可疑性分数（表格按照该分数降序排列），第三列为操作按钮。若某一源代码文件被三种定位方法都认为可疑性很大，则会将该源代码文件名高亮，提高开发者检查该源代码的优先级，如图中三种定位算法都预测了org.eclipse.swt.ole.win32.Variant.java，遂源代码文件名单元格背景色高亮。

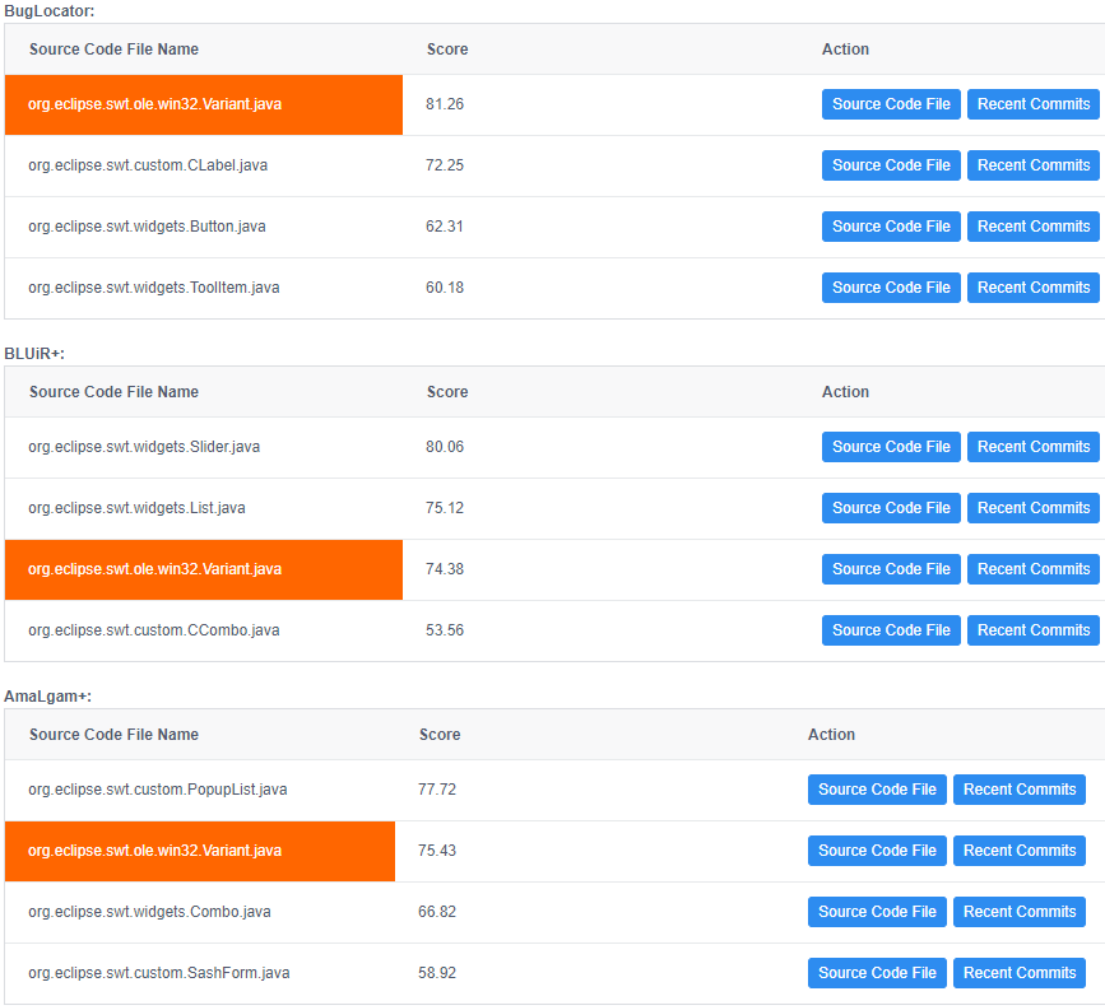

图 4.25: Bug定位结果实现效果图

点击Source Code Filea按钮可以跳转到该源代码文件在GitHub 中的页面，点击Recent Commits 按钮可以查看该源代码文件最近的commit热力图，如图4.26所示。若最近commit 次数为0，则是该源代码文件导致Bug 产生的可能性较低。点击热力图，可以查看该日commit信息的详细记录。

缺陷定位效果评估

为了验证实现的Bug定位准确程度，使用Python的Scrapy框架爬取了开源仓库ZooKeeper截止2018年10月16的所有状态为FIXED或RESOLVED的Bug报告，共计1243份，使用这些Bug报告对三种Bug定位算法进行准确度验证。

对于TOP K准确度的定义如下，若三种定位算法中有任意一种的前K项结果包含实际修复的源代码文件，则认为本系统在TOP K 内命中。如图 4.27 所示，为本系统对爬取的1243 份Bug报告进行定位的TOP K成功预测数目。其

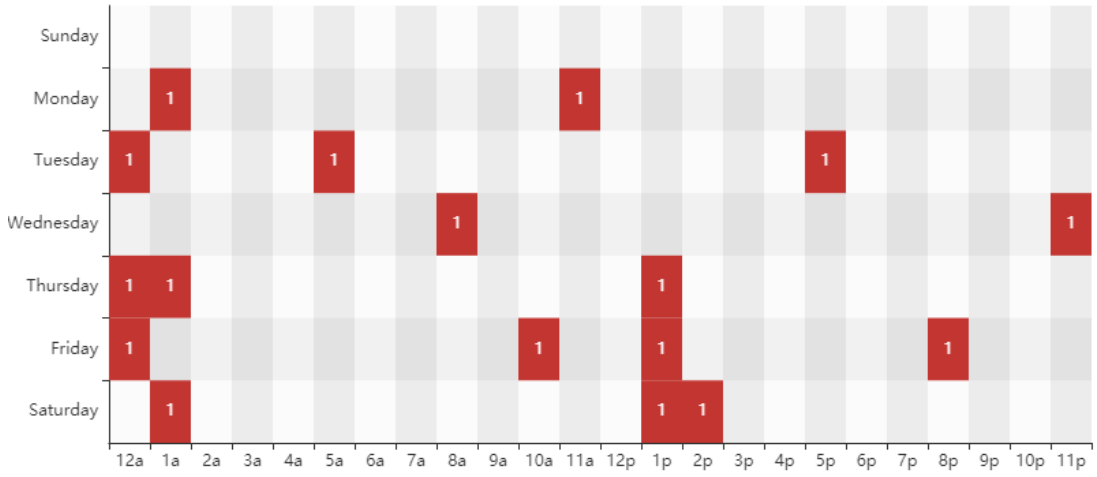

图 4.26: 源代码文件最近一周commit热力图

中TOP 1的命中数目为653份，TOP 5的命中数目为981 份，TOP 10的命中数目为1087份，预测成功率分别为52.53%、78.92%和87.45%。由此可以看出，本系统Bug定位准确率能够达到辅助开发者进行Bug定位的效果。

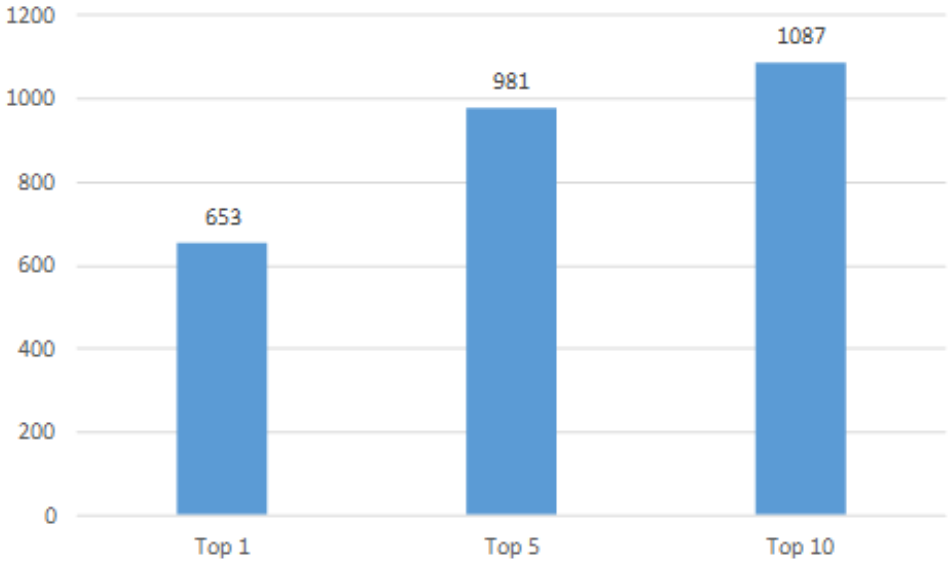

图 4.27: 本平台定位算法Zookeeper Bug报告TopK命中数目

4.4.3　Bug定位结果反馈

为了收集每个Bug定位方法的定位效果，在仓库所有者将每个含有Bug定位任务的Bug报告状态转为fixed时，需要其对所使用的定位方法的准确性打分，

为系统在未来改进Bug定位算法提供数据集支撑。如图 4.28所示，为Bug定位结果反馈顺序图。

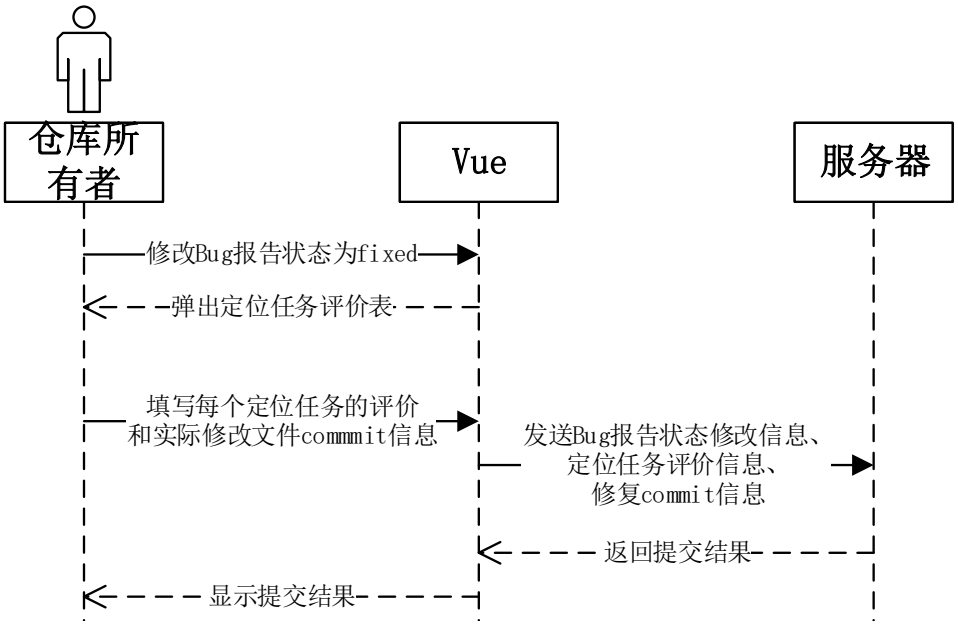

图 4.28: Bug定位结果反馈顺序图

4.5　系统测试

本章前几节给出了系统三个模块的详细设计与具体实现，目前本项目代码已经开源到GitHub中[1]。为了保证系统能够满足设计之初的功能性与非功能性需求，本节将依据上述内容对面向GitHub开源社区的Bug 定位系统展开测试，分别介绍测试环境、三个功能模块的测试内容以及性能测试。

4.5.1　测试环境

进行测试工作前需要搭建测试环境，因本系统是在Windows环境下进行和完成开发，而在实际的生产环境中服务端应当是部署到Linux服务器上，所以对系统所用到工具和中间件的版本进行严格的控制。系统测试环境以阿里云轻量级服务器为基础搭建并展开系统测试工作。

表 4.1是系统的测试环境，为了保证系统能够正常地对外网提供服务，部署服务器具体需要开放如下端口：一、Nginx服务器监听8080 端口，允许外网访问前端可视化页面；二、Java 虚拟机负责服务端的正常运行。因本系统服务端

[1]https://github.com/VousAttendezrer/bulingbuling

表 4.1: 测试环境表

参数	参数详情
机型	阿里云轻量级应用服务器
CPU	Intel(R) Xeon(R) Platinum 8163 CPU @ 2.50GHz
内存	2GB
硬盘	40GB SSD云盘
操作系统	CentOS 7.5
JVM版本	1.8.0_202
Nginx版本	1.14.2 version
Mysql版本	MySQL Server 8.0
Git版本	git version 2.20.1

开发框架选取Spring Boot，该框架开发完成打包后自行携带Tomcat服务器，无需额外部署。由服务端监听8081端口以保证后端服务的正常访问；三、Mysql负责监听3306端口号，并在测试的过程中开发远程连接登录配置，保证服务端能够正常访问数据库信息。

4.5.2　用户信息功能模块测试

用户信息功能模块测试用例如表 4.2所示，根据该模块所属的GitHub账户授权、开源仓库获取、邮箱关联、信誉积分和消息通知子功能模块，总共设计并验证了5个功能测试。对该模块进行功能测试的目的是为了保证系统能够在已经获取用户授权的情况下，正确地从GitHub 中接收用户和开源仓库信息，并结合使用本台产生的活动，将其存储到系统数据库中。

4.5.3　缺陷追踪功能模块测试

缺陷追踪功能模块的测试用例如表 4.3所示，该功能模块主要包含为开源仓库添加缺陷追踪功能、提交新的Bug报告、纳入缺陷管理的开源仓库可以选择缺陷追踪分支、按照一定格式批量导入历史Bug 报告、查看Bug 报告列表和每份Bug报告的详细信息等子功能，共设计并验证了五个功能测试。对该模块进行功能测试目的，是为了保证开发者可以正确地使用本系统、提交Bug报告、切换开源仓库分支、导入历史Bug报告、查看Bug报告列表和详细信息这些基础功能，通过这些系统基础功能的正确运行保证Bug 定位任务的正常执行，从而为用户提供舒适、满意、准确的Bug定位服务。

表 4.2: 用户信息功能模块测试用例表

用例编号	用例描述	输入/步骤	预期输出	测试结果
TUS1-1	在用户授权情况下能正确获取GitHub账号信息	点击使用GitHub账号登录按钮后，在GitHub 授权页面输入用户名密码并勾选所需权限	页面跳转回系统主页，页面右上方使用GitHub 账号登录按钮变为GitHub 账户名	通过
TUS1-2	获取用户的开源仓库信息	点击右上方用户名下拉菜单中的我的开源仓库按钮	页面跳转到个人开源仓库，显示所属该账号的开源仓库	通过
TUS1-3	用户绑定自己日常使用的邮箱账户	个人信息页面点击邮箱绑定按钮，正确填写邮箱后点击发送验证码	用户查看系统发送的邮件，填写接收到的验证码即可绑定成功	通过
TUS1-4	信誉积分低于阈值限制提交报告	当账户信誉积分低于阈值时，在Bug报告列表页面新建Bug 报告	页面提示信誉积分不足，可通过使用系统其他功能提升信誉积分	通过
TUS1-5	Bug 报告的状态发生改变时通知变更消息	仓库所有者在Bug报告详细页面，将Bug报告的状态由未确认状态修改为已确认状态	对应Bug 报告提交者接收到站内信提示，同时邮箱收到Bug报告状态变更邮件	通过

4.5.4　Bug定位功能模块测试

Bug定位功能模块的测试用例如表 4.4所示，该功能模块包含Bug定位任务创建、Bug 定位任务进度实时跟踪、Bug定位任务结果展示及定位准确度反馈等子功能，据此设计并验证了四个测试用例。对该功能模块进行功能测试的目的，为了保证系统核心功能Bug定位能够按照预期正常运行，能够正确的创建Bug定位任务、获取定位任务的进度、查看Bug定位结果并对定位结果进行反馈，此外还保证用户界面的友好性，能够正确高亮和显示定位结果的交集、最近的commit信息和历史关联缺陷报告的信息。同时确保实现并集成的三种基于信息检索的Bug定位算法计算结果是正确的，保证定位结果的准确性。

表 4.3: 缺陷追踪功能模块测试用例表

用例编号	用例描述	输入/步骤	预期输出	测试结果
TUS2-1	用户可以为自己的开源仓库添加缺陷追踪功能	在仓库总览页面，选择任一仓库，点击该仓库名下方的添加缺陷追踪功能按钮	添加缺陷追踪按钮变为仓库设定按钮，可以点击仓库设定按钮更改缺陷追踪的分支	通过
TUS2-2	向其他仓库提交缺陷报告	点击提交报告按钮，正确填写各项缺陷报告信息	前端页面提示缺陷报告提交成功，可以查看详细信息	通过
TUS2-3	对开源仓库缺陷追踪选择具体分支	点击开源仓库名下的仓库设定按钮，在分支选择选项中切换分支并保存	该开源仓库新接收到的缺陷报告会被收纳到该分支下进行管理	通过
TUS2-4	将历史缺陷报告批量导入到系统中	将其他缺陷追踪的缺陷报告导出并整理成给定JSON格式，上传至系统	在缺陷报告列表页面，可以查看到刚刚批量导入的缺陷报告	通过
TUS2-5	查看缺陷报告列表和单份缺陷报告详细信息	点击缺陷报告列表按钮，选择任一缺陷报告条目，点击最右列操作栏的查看详情按钮	缺陷报告列表显示近期提交的Bug 报告，缺陷报告详细页面可以正确显示各条目信息	通过

4.5.5　性能测试

对于性能测试，我们使用Apache旗下的JMeter工具对本系统进行压力测试，以确保能够满足非功能性需求中在性能方面的要求。首先，使用JMeter创建进程组，考虑到系统上线初期的用户群体数量不会太大，我们将Number of Treads参数值设置为500，再将Ramp-Up Period参数值设置为1，这两个参数的含义是通过JMeter创建500个虚拟用户，模拟这500个虚拟用户在1秒内使用HTTP客户端访问本系统开放的部分RESTful API接口。接着，在工具中配置要测试API接口的方法名和参数等信息，运行测试。

表 4.4: Bug定位功能模块测试用例表

用例编号	用例描述	输入/步骤	预期输出	测试结果
TUS3-1	对状态为已确认的缺陷报告创建定位任务	在状态为已确认的缺陷报告页面，点击右上角创建定位任务按钮，开始执行定位任务	缺陷报告下方展示刚刚运行的定位任务，其中第一个任务状态为正在克隆，其它为队列中	通过
TUS3-2	实时获取定位任务进度	运行定位任务后，每隔一段时间观察定位任务状态变更	定位任务的状态从克隆变更为定位中再变更为已完成	通过
TUS3-3	获取缺陷定位结果的源文件和分数	在缺陷报告页面下方，展开定位任务信息，查看定位结果和分数	三种定位结果的交集会被高亮，所有源代码文件都包含可疑性分数	通过
TUS3-4	收集仓库所有者对定位结果的反馈	将已确认状态的缺陷报告转变为已修复，在弹出对话框中对定位任务打分	提交评分后，隶属于该缺陷报告的定位任务信息，都包含有评分结果	通过

如图 4.29所示，为使用JMeter对本系统部分API接口进行测试所得到的测试结果。该测试结果中，每一行的内容代表JMeter对本系统某个API进行性能测试的结果。userInfoDecode为解析用户信息接口，其平均响应时间为447ms，响应时间中位数为457ms，响应异常率为0；submitNewBugReport 为提交新Bug报告接口，其平均响应时间为247ms，响应时间中位数为211ms，响应异常率为0；uploadHistoryBugReports 为批量导入历史Bug 报告接口，其平均响应时间为282ms，响应时间中位数为286ms，响应异常率为0；getSingleRepositoryBug-Reports 为获取单个开源仓库所有Bug报告接口，其平均响应时间为242ms，响应时间中位数为208ms，响应异常率为0；checkUserEmailResult 为检查用户是否关联邮箱接口，其平均响应时间为208ms，响应时间中位数为174ms，响应异常率为0；signInWithGitHubCode 为使用GitHub账号登录本系统接口，其平均响应时间为168ms，响应时间中位数为139ms，响应异常率为0；changeRepository-Setting 为修改具体开源仓库设定接口，其平均响应时间为137ms，响应时间中位数为103ms，响应异常率为0。

Label	# Samples	Average	Median	Error %
userInfoDecode	500	447	457	0.00%
submitNewBugReport	500	247	211	0.00%
uploadHistoryBugReports	500	282	286	0.00%
getSingleRepositoryBugRep...	500	242	208	0.00%
checkUserEmailResult	500	208	174	0.00%
signInWithGitHubCode	500	168	139	0.00%
changeRepositorySetting	500	137	103	0.00%
TOTAL	3500	247	196	0.00%

图 4.29: 系统部分接口性能测试结果图

从以上部分测试结果可以看出，在500线程数的压力测试下，本系统服务端接口的响应时间能够保证平均响应时间和响应时间中位数都不超过500ms，且发生响应异常的机率为0。初步满足了需求设计之初，对本系统的性能要求。

4.6　本章小节

本章详细描述了面向开源社区的Bug报告定位系统中各模块关键部分的详细设计与具体实现，包括用户信息模块、缺陷追踪模块、Bug定位算法。通过顺序图、关键代码展现、流程图以及实现效果截图的方式，阐述了关键功能点的详细设计与具体实现，并在最后一节通过功能测试用例对系统进行系统测试以保证系统正确运行。

第五章　总结与展望

5.1　总结

本文通过对开源社区GitHub的详细调研和对基于信息检索的Bug定位技术的研究学习，设计和实现了面向开源社区的Bug定位系统。为开源社区的开发者们提供了简便、先进的Bug定位技术，节约他们在定位Bug过程中所需要耗费的时间和精力。

系统核心功能包括用户信息功能模块、缺陷追踪功能模块和Bug定位功能模块。首先，用户信息功能模块使用clientID和access_token等OAuth2.0方式，取得开源社区GitHub授权，获取开源社区开发者许可后同步其账号信息；取得开发者开源仓库clone权限，为Bug定位任务提供源代码文件输入；设立信誉积分机制，杜绝用户恶意提交Bug报告的现象；提供邮箱关联功能，以验证用户事实人身份和通知Bug报告、定位任务变更信息。然后，缺陷追踪功能模块通过检查开源仓库开发语言信息以允许用户为所属仓库提供缺陷追踪功能；允许其他用户在使用开源仓库遇到Bug时，将其以Bug报告的形式提交给开发者进行验证和修改；导入使用其他缺陷追踪系统产生的Bug报告，为Bug定位任务提供更多更有效的Bug报告数据集；开发者还可以修改Bug 报告的状态信息，做到对自己开源仓库健康状况的统筹管理；最后，核心Bug 定位功能模块允许用户为状态是已确认的Bug报告创建定位任务；根据自身需求选择基于信息检索的Bug定位算法，或者傻瓜式批量运行所有Bug定位算法；获取定位结果，从多维度对预测Bug的源代码文件进行展示；并在对结果准确度进行反馈。

系统具有良好的可拓展性，模块与模块之间都是松耦合的。上层模块不直接依赖下层模块的数据结构，而是通过元数据传递消息信息。元数据中只包含对内容所属事物的描述，并不包含对内容的定性，所有定性工作均由获得元数据的上层模块完成。下层可以随时添加新特性，或者增加新的模块，而新的模块只需要满足已经预先定义好的接口规范即可通过元数据向上层模块传递信息。同理，上层模块也只需要修改对下层模块的引用即可实现模块切换。

系统已经部署在阿里云服务器上，所有预先设计的功能模块都可正常运行，达成初步阶段目标。同时本系统的源代码已经开源在GitHub 中。未来将上线GitHub 的应用市场MarketPlace，供开源社区的开发者们辅助Bug 定位使用。

5.2 展望

本系统虽然已经搭建完成，完成了初步设计目标。但由于个人水平有限和时间限制因素，依然存在着诸多不足之处。

首先，对于缺陷追踪功能模块而言，本系统基于为Bug定位服务的理念对该模块做了简化设计。而事实上，现有主流的缺陷追踪系统（Bugzilla和Jira）本身都是十分庞大和复杂的，系统在这一块上可以将其做的更加深入，更加完备，如全文检索、直接从Bugzilla、Jira和GitHub issue 自动同步Bug 报告等。

其次，实现并集成的几种基于信息检索的Bug 定位算法都是面向Java编程语言的，并没有涉及其他编程语言，从这个方面来说，系统的广度太窄，未来可以引入针对其他语言的Bug定位算法。

最后，系统在设计之初就提供了定位结果反馈功能，同时要求用户在修复Bug 后提交修复的commit 信息，这也为后续改进定位算法提供了足够庞大的数据集。

参考文献

[1] F. Chatziasimidis, I. Stamelos, Data collection and analysis of github repositories and users, in: N. G. Bourbakis, G. A. Tsihrintzis, M. Virvou (Eds.), 6th International Conference on Information, Intelligence, Systems and Applications, IISA 2015, Corfu, Greece, July 6-8, 2015, IEEE, 2015, pp. 1–6.

[2] W. Ma, L. Chen, X. Zhang, Y. Zhou, B. Xu, How do developers fix cross-project correlated bugs?: a case study on the github scientific python ecosystem, in: S. Uchitel, A. Orso, M. P. Robillard (Eds.), Proceedings of the 39th International Conference on Software Engineering, ICSE 2017, Buenos Aires, Argentina, May 20-28, 2017, IEEE / ACM, 2017, pp. 381–392.

[3] R. Krishna, A. Agrawal, A. Rahman, A. Sobran, T. Menzies, What is the connection between issues, bugs, and enhancements? (lessons learned from 800+ software projects), CoRR abs/1710.08736.

[4] T. Zhang, J. Chen, G. Yang, B. Lee, X. Luo, Towards more accurate severity prediction and fixer recommendation of software bugs, Journal of Systems and Software 117 (2016) 166–184.

[5] A. S. S. Fiaz, N. Devi, S. Aarthi, Bug tracking and reporting system, CoRR abs/1309.1232.

[6] Y. Pérez-Riverol, L. Gatto, R. Wang, T. Sachsenberg, J. Uszkoreit, F. da Veiga Leprevost, C. Fufezan, T. Ternent, S. J. Eglen, D. S. Katz, T. J. Pollard, A. Konovalov, R. M. Flight, K. Blin, J. A. Vizcaíno, Ten simple rules for taking advantage of git and github, PLoS Computational Biology 12 (7).

[7] A. N. Lam, A. T. Nguyen, H. A. Nguyen, T. N. Nguyen, Combining deep learning with information retrieval to localize buggy files for bug reports (N), in: M. B. Cohen, L. Grunske, M. Whalen (Eds.), 30th IEEE/ACM International Conference on Automated Software Engineering, ASE 2015, Lincoln, NE, USA, November 9-13, 2015, IEEE Computer Society, 2015, pp. 476–481.

[8] R. Abreu, P. Zoeteweij, R. Golsteijn, A. J. C. van Gemund, A practical evaluation of spectrum-based fault localization, Journal of Systems and Software 82 (11) (2009) 1780–1792.

[9] T. Hoang, R. J. Oentaryo, T. B. Le, D. Lo, Network-clustered multi-modal bug localization, CoRR abs/1802.09729.

[10] T. B. Le, R. J. Oentaryo, D. Lo, Information retrieval and spectrum based bug localization: better together, in: E. D. Nitto, M. Harman, P. Heymans (Eds.), Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015, Bergamo, Italy, August 30 - September 4, 2015, ACM, 2015, pp. 579–590.

[11] J. A. Jones, M. J. Harrold, Empirical evaluation of the tarantula automatic fault-localization technique, in: D. F. Redmiles, T. Ellman, A. Zisman (Eds.), 20th IEEE/ACM International Conference on Automated Software Engineering (ASE 2005), November 7-11, 2005, Long Beach, CA, USA, ACM, 2005, pp. 273–282.

[12] G. Salton, C. Buckley, Term-weighting approaches in automatic text retrieval, Inf. Process. Manage. 24 (5) (1988) 513–523.

[13] S. Wu, Y. Li, Y. Xu, Deploying approaches for pattern refinement in text mining, in: Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006), 18-22 December 2006, Hong Kong, China, 2006, pp. 1157–1161.

[14] G. Gay, S. Haiduc, A. Marcus, T. Menzies, On the use of relevance feedback in ir-based concept location, in: 25th IEEE International Conference on Software Maintenance (ICSM 2009), September 20-26, 2009, Edmonton, Alberta, Canada, IEEE Computer Society, 2009, pp. 351–360.

[15] Y. Yue, T. Finley, F. Radlinski, T. Joachims, A support vector method for optimizing average precision, in: W. Kraaij, A. P. de Vries, C. L. A. Clarke, N. Fuhr, N. Kando (Eds.), SIGIR 2007: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, The Netherlands, July 23-27, 2007, ACM, 2007, pp. 271–278.

[16] O. Chapelle, D. Metlzer, Y. Zhang, P. Grinspan, Expected reciprocal rank for graded relevance, in: D. W. Cheung, I. Song, W. W. Chu, X. Hu, J. J. Lin (Eds.),

Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009, ACM, 2009, pp. 621–630.

[17] S. Rivard, S. L. Huff, Factors of success for end-user computing, Commun. ACM 31 (5) (1988) 552–561.

[18] J. Zhou, H. Zhang, D. Lo, Where should the bugs be fixed? more accurate information retrieval-based bug localization based on bug reports, in: 34th International Conference on Software Engineering, ICSE 2012, June 2-9, 2012, Zurich, Switzerland, 2012, pp. 14–24.

[19] R. K. Saha, M. Lease, S. Khurshid, D. E. Perry, Improving bug localization using structured information retrieval, in: E. Denney, T. Bultan, A. Zeller (Eds.), 2013 28th IEEE/ACM International Conference on Automated Software Engineering, ASE 2013, Silicon Valley, CA, USA, November 11-15, 2013, IEEE, 2013, pp. 345–355.

[20] S. Wang, D. Lo, Amalgam+: Composing rich information sources for accurate bug localization, Journal of Software: Evolution and Process 28 (10) (2016) 921–942.

[21] S. H. Hough, K. Kancleris, L. Brody, N. Humphryes-Kirilov, J. Wolanski, K. Dunaway, A. Ajetunmobi, V. Dillard, Guide picker is a comprehensive design tool for visualizing and selecting guides for CRISPR experiments, BMC Bioinformatics 18 (1) (2017) 167:1–167:10.

[22] F. Bellucci, G. Ghiani, F. Paternò, C. Porta, Automatic reverse engineering of interactive dynamic web applications to support adaptation across platforms, in: C. Duarte, L. Carriço, J. A. Jorge, S. L. Oviatt, D. Gonçalves (Eds.), 17th International Conference on Intelligent User Interfaces, IUI 2012, Lisbon, Portugal, February 14-17, 2012, ACM, 2012, pp. 217–226.

[23] E. Curry, P. Grace, Flexible self-management using the model-view-controller pattern, IEEE Software 25 (3) (2008) 84–90.

[24] N. Omerbegovic, N. Omerbegovic, A. Subasi, Effects of two-way data binding on better user experience and easier development of clinical information systems,

in: A. Paschke, A. Burger, P. Romano, M. S. Marshall, A. Splendiani (Eds.), Proceedings of the 7th International Workshop on Semantic Web Applications and Tools for Life Sciences, Berlin, Germany, December 9-11, 2014., Vol. 1320 of CEUR Workshop Proceedings, CEUR-WS.org, 2014.

[25] H. Li, K. Leung, T. Nakane, M. H. Wong, iview: an interactive webgl visualizer for protein-ligand complex, BMC Bioinformatics 15 (2014) 56.

[26] Z. Zheng, N. Ahmed, K. Mueller, iview: A feature clustering framework for suggesting informative views in volume visualization, IEEE Trans. Vis. Comput. Graph. 17 (12) (2011) 1959–1968.

[27] J. Aguinaga, How it feels to learn javascript in 2016 (2016).

[28] R. T. Fielding, R. N. Taylor, Principled design of the modern web architecture, in: C. Ghezzi, M. Jazayeri, A. L. Wolf (Eds.), Proceedings of the 22nd International Conference on on Software Engineering, ICSE 2000, Limerick Ireland, June 4-11, 2000., ACM, 2000, pp. 407–416.

[29] C. Pautasso, O. Zimmermann, F. Leymann, Restful web services vs. "big"' web services: making the right architectural decision, in: J. Huai, R. Chen, H. Hon, Y. Liu, W. Ma, A. Tomkins, X. Zhang (Eds.), Proceedings of the 17th International Conference on World Wide Web, WWW 2008, Beijing, China, April 21-25, 2008, ACM, 2008, pp. 805–814.

[30] F. Belqasmi, J. Singh, S. Y. B. Melhem, R. H. Glitho, Soap-based vs. restful web services: A case study for multimedia conferencing, IEEE Internet Computing 16 (4) (2012) 54–63.

[31] N. Bettenburg, S. Just, A. Schröter, C. Weiss, R. Premraj, T. Zimmermann, What makes a good bug report?, in: M. J. Harrold, G. C. Murphy (Eds.), Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2008, Atlanta, Georgia, USA, November 9-14, 2008, ACM, 2008, pp. 308–318.

[32] T. B. Le, F. Thung, D. Lo, Predicting effectiveness of ir-based bug localization techniques, in: 25th IEEE International Symposium on Software Reliability Engineering, ISSRE 2014, Naples, Italy, November 3-6, 2014, 2014, pp. 335–345.

[33] R. Auckenthaler, M. J. Carey, H. Lloyd-Thomas, Score normalization for text-independent speaker verification systems, Digital Signal Processing 10 (1-3) (2000) 42–54.

[34] P. Willett, The porter stemming algorithm: then and now, Program 40 (3) (2006) 219–223.

[35] M. Sanderson, Christopher d. manning, prabhakar raghavan, hinrich schütze, *Introduction to Information Retrieval*, cambridge university press 2008. ISBN-13 978-0-521-86571-5, xxi + 482 pages, Natural Language Engineering 16 (1) (2010) 100–103.

[36] A. López-Ortiz, Search engines and web information retrieval, in: A. López-Ortiz, A. M. Hamel (Eds.), Combinatorial and Algorithmic Aspects of Networking, First Workshop on Combinatorial and Algorithmic Aspects of Networking, CAAN 2004, Banff, Alberta, Canada, August 5-7, 2004, Revised Selected Papers, Vol. 3405 of Lecture Notes in Computer Science, Springer, 2004, pp. 183–191.

[37] D. H. R. Spennemann, The internet and daily life in australia: An exploration, Inf. Soc. 22 (2) (2006) 101–110.

[38] B. J. Jansen, S. Y. Rieh, The seventeen theoretical constructs of information searching and information retrieval, JASIST 61 (8) (2010) 1517–1534.

[39] F. Can, Information retrieval data structures & algorithms, by william b. frakes and ricardo baeza-yates (book review), SIGIR Forum 27 (3) (1993) 24–25.

[40] P. J. Thomas, R. D. Macredie, Introduction to the new usability, ACM Trans. Comput.-Hum. Interact. 9 (2) (2002) 69–73.

[41] J. Rosenberg, Tutorial: A quick introduction to software reliability modeling, in: B. W. Boehm, D. Garlan, J. Kramer (Eds.), Proceedings of the 1999 International Conference on Software Engineering, ICSE' 99, Los Angeles, CA, USA, May 16-22, 1999., ACM, 1999, p. 689.

简历与科研成果

基本情况 方文强，男，汉族，1994 年 10 月出生，安徽省铜陵市人。

教育背景

2017.9～2019.6 南京大学软件学院 硕士

2012.9～2016.6 厦门理工学院计算机与信息工程系 本科

专利与项目：

1. 刘嘉，刘锦涛，**方文强**，邹卫琴，李玉莹，“一种基于Star信息和README文档的Github相似仓库推荐”，申请号：201810092877.6，已受理。
2. 国家自然科学基金项目（面上项目）：基于开发者社交网络的软件维护技术（61472176）, 2015-2018
3. 国家自然科学基金项目（面上项目）：软件数值稳定性的检测与修复技术研究（61772260）, 2018-2021

致　　谢

时光荏苒，硕士生涯已临近尾声。虽然这两年时光非常短暂，但是在我的心中却弥足珍贵，因为它不仅充满了酸甜苦辣，更包含着收获与成长。

在本论文即将完成之际，谨此向本人的导师陈振宇教授致以衷心的感谢！本论文是在陈老师悉心的指导下完成的，从最初的论文的方向和选题，到论文各级目录的确定，再到论文摘要的撰写，最后到论文内容的编写直至完成，均得到了陈老师的指导和建议。陈老师以严谨的治学态度、精益求精的工作作风和对敏锐的洞察力，及时纠正了我在整个论文准备期间进行方向上的偏差、论文表述侧重点的不妥之处，在本人心中留下了深刻的印象，并将继续成为本人在今后学习和工作过程中学习的榜样和前进的动力。

同时感谢论文准备前期一起合作的邹卫琴学姐和刘锦涛同学，相互支持、共同推进进展、解决遇到的疑难问题，他们不仅是合作上的伙伴，更是生活中的益友。此外，还要感谢家人、室友及其他同学的一路同行，谢谢他们的支持、理解和帮助，没有他们就没有这篇论文的顺利完成。

最后，我要向在百忙之中抽空参与评审本论文的各位老师、参与本人论文答辩的老师和专家评委们，表示由衷的感谢！

版权及论文原创性说明

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：　　年　　月　　日