



南京大學
NANJING UNIVERSITY

研 究 生 毕 业 论 文

(申 请 硕 士 专 业 学 位)

论 文 题 目 Web 应用自动化测试系统分析服务的设计与实现

作 者 姓 名 周赛

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

2020 年 5 月 20 日

学 号 : MF1832270

论文答辩日期 : 2020 年 5 月 23 日

指 导 教 师 :  (签字)



The Design and Implementation of the Analysis Service in Web Application Automation Testing System

By

Sai Zhou

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 2020

南京大学研究生毕业论文中文摘要首页用纸

毕业论文题目：Web 应用自动化测试系统分析服务的设计与实现

工程硕士（软件工程领域） 专业 2018 级硕士生姓名：周赛

指导教师（姓名、职称）：陈振宇 教授

摘 要

随着互联网化程度逐步加深，Web 应用以其易用性和便利性，广泛使用于各行各业和日常生活中，成为当前信息时代不可或缺的一部分。而面临日益增长变化的需求和开发迭代敏捷化的现状同时，Web 应用质量也成为开发测试人员和用户极为关心的核心领域。然而，现有 Web 应用质量检测手段，主要还是依靠手工测试或者脚本测试，不仅人力和时间成本消耗较大，而且测试效率也较低，且可重复性差。另一方面，作为回归测试，这些测试手段虽然也能发现一些具有明显表征的系统缺陷，但是仍然需要用户参与缺陷定位和测试报告填写，同时一些无表征但存在一定风险的缺陷也可能被漏报。因此，研究 Web 应用的自动化测试和测试分析就显得十分必要。

本文设计并实现了 Web 应用自动化测试系统中的分析服务部分，主要针对 Web 应用测试的缺陷扫描和缺陷分析，提供智能解决方案。应用依赖于负责自动化遍历测试的执行服务和调度组件记录的待测系统相关日志、截图等多源“黑匣子”数据，利用聚类算法和特征检测等方法，构建了 Web 应用相关缺陷体系，并对应实现和集成相关数据的缺陷检测器，最后通过检测发掘应用缺陷，构建待测应用的测试报告，同时支持对其他分析数据和可能缺陷类别的拓展和兼容需要。另一方面，系统还建立了初步的 Web 应用评价体系，从而实现对 Web 应用质量的多维度评估。整个系统基于 SpringBoot 框架暴露服务，供下游使用，使用 MongoDB、Redis 和 OSS 作为数据中心，利用 RabbitMQ 作为消息队列实现异步调用，并通过 Celery 实现子任务的调度和分发。

目前该系统已经部署，并面向企业和高校提供差异化支持。基于 50 个线上真实网站的实验，也表明系统能够有效发掘断链、资源加载失败、服务器端错误以及 JS 脚本错误等多种 Web 应用缺陷，同时人工抽样审查的系统缺陷分类正确率和导出测试脚本执行正确率均超过 95%，且不限于多样化的 Web 应用前端开发框架选型，能够为大部分业界网站提供自动化测试评估服务，并和人工测试、脚本测试以及众包测试等多种测试手段一道，为 Web 应用质量保驾护航。

关键词：Web 应用测试，自动化测试，测试分析，缺陷评估

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 the Analysis Service in Web
Application Automation Testing System

SPECIALIZATION: Software Engineering

POSTGRADUATE: Sai Zhou

MENTOR: Professor Zhenyu Chen

Abstract

With the gradual deepening of the Internet degree, web applications are widely used in all walks of life and daily life due to their simplicity and convenience, and have become an integral part of the current information age. Faced with the current situation of the ever-changing needs and agile development iterations, the web applications quality has also become a core area that developers, testers and users are extremely concerned about. However, the existing web application quality detection methods mainly rely on manual testing or script testing, which not only consumes a lot of manpower and time, but also has low testing efficiency and poor repeatability. On the other hand, as a regression test, although these testing methods can also find some system defects with obvious characterization, they still require users to participate in defect positioning and test report filling. At the same time, some non-characteristic defects with certain risks may not be find. Therefore, it is very necessary to study the automated testing and testing analysis of web applications.

This thesis designs and implements the analysis service part of the Web application automation testing system, mainly for the defect scanning and defect analysis of web application testing, and provides intelligent solutions. The system relies on multi-source "black box" data of the system under testing, such as related logs, screenshots and so on, which are recorded by the execution service and scheduling component responsible for automated traversal testing. Using methods such as clustering algorithms and feature detection, a web application related defect system is constructed, realizing and integrating some system defect detectors corresponding to relevant data. Finally, the system discovers defects through detection, and constructs test report of the application to be tested. At the same time, it supports the expansion and compatibility need

of other analysis data and possible defect categories. On the other hand, the system has also established a preliminary web application evaluation system to achieve multi-dimensional evaluation of web application quality. The entire system exposes services for downstream use based on the SpringBoot framework, uses MongoDB, Redis, and OSS as data centers, uses RabbitMQ as a message queue to implement asynchronous calls, and uses Celery to schedule and distribute subtasks.

At present, the system has been deployed and provides differentiated support for enterprises and universities. Experiments based on 50 real online websites also show that the system can effectively discover various web application defects such as broken links, resource loading failures, server-side errors, JavaScript errors and so on. At the same time, both the accuracy rate of system defect classification and the correct rate of export test script execution exceeds 95% based on manual sample and review, and system is not limited to the selection of diverse web application front-end development frameworks. It can provide automated testing and evaluation services for most industry websites. Together with a variety of testing methods such as manual testing, script testing, and crowdsourcing testing, it escorts web application quality.

Keywords: Web Application Testing, Automated Testing, Testing Analysis, Defect Evaluation

目录

| | |
|--------------------------------------|----------|
| 表 目 录 | ix |
| 图 目 录 | xii |
| 第一章 引言 | 1 |
| 1.1 项目背景及意义 | 1 |
| 1.2 国内外研究现状 | 2 |
| 1.2.1 Web 应用自动化测试研究现状 | 2 |
| 1.2.2 软件测试缺陷分析研究现状 | 4 |
| 1.3 本文的主要工作 | 5 |
| 1.4 本文的组织结构 | 5 |
| 第二章 相关技术 | 7 |
| 2.1 Web 应用测试相关技术 | 7 |
| 2.1.1 Selenium | 7 |
| 2.1.2 Crawljax | 8 |
| 2.1.3 Chrome Devtools Protocol | 8 |
| 2.2 服务端相关技术 | 9 |
| 2.2.1 Redis | 9 |
| 2.2.2 MongoDB | 10 |
| 2.2.3 RabbitMQ | 11 |
| 2.2.4 SpringBoot | 12 |
| 2.3 缺陷日志挖掘相关算法 | 12 |
| 2.3.1 缺陷日志聚类算法 | 12 |
| 2.3.2 Levenshtein 距离 | 13 |
| 2.3.3 Jaccard 系数 | 14 |
| 2.3.4 轮廓系数 | 14 |
| 2.4 本章小结 | 15 |

| | |
|---------------------------------|-----------|
| 第三章 系统需求分析与概要设计 | 17 |
| 3.1 Web 应用自动化测试系统整体概述 | 17 |
| 3.2 Web 应用自动化测试系统分析服务概述 | 18 |
| 3.3 Web 应用自动化测试系统需求分析 | 18 |
| 3.3.1 涉众分析 | 19 |
| 3.3.2 功能性需求 | 19 |
| 3.3.3 非功能性需求 | 21 |
| 3.3.4 系统用例图 | 21 |
| 3.3.5 系统用例描述 | 22 |
| 3.4 Web 应用自动化测试系统整体架构设计 | 27 |
| 3.5 Web 应用自动化测试系统分析服务概要设计 | 29 |
| 3.5.1 架构设计与模块划分 | 29 |
| 3.5.2 4+1 视图 | 31 |
| 3.5.3 持久化模型设计 | 34 |
| 3.6 本章小结 | 36 |
| 第四章 系统详细设计与具体实现 | 37 |
| 4.1 Web 应用缺陷分析框架 | 37 |
| 4.1.1 框架设计 | 37 |
| 4.1.2 流程设计 | 38 |
| 4.1.3 数据与类设计 | 39 |
| 4.1.4 关键代码 | 41 |
| 4.2 控制台日志缺陷聚类子模块 | 42 |
| 4.2.1 架构设计 | 42 |
| 4.2.2 流程设计 | 43 |
| 4.2.3 数据与类设计 | 44 |
| 4.2.4 关键代码 | 45 |
| 4.3 控制台日志缺陷判定子模块 | 47 |
| 4.3.1 架构设计 | 47 |
| 4.3.2 流程设计 | 48 |
| 4.3.3 数据与类设计 | 49 |

| | | |
|------------|------------------------|-----------|
| 4.3.4 | 关键代码 | 50 |
| 4.4 | 请求日志缺陷检测模块 | 51 |
| 4.4.1 | 架构设计 | 51 |
| 4.4.2 | 流程设计 | 52 |
| 4.4.3 | 数据与类设计 | 53 |
| 4.4.4 | 关键代码 | 55 |
| 4.5 | 软件质量多维评估模块 | 56 |
| 4.5.1 | 架构设计与维度分析 | 57 |
| 4.5.2 | 数据与类设计 | 58 |
| 4.5.3 | 关键代码与界面截图 | 59 |
| 4.6 | 本章小结 | 60 |
| 第五章 | 系统测试与实验分析 | 61 |
| 5.1 | 系统测试 | 61 |
| 5.1.1 | 测试目标与测试环境 | 61 |
| 5.1.2 | 功能测试 | 62 |
| 5.2 | 实验分析 | 65 |
| 5.2.1 | 实验对象 | 65 |
| 5.2.2 | 实验设计 | 66 |
| 5.2.3 | 实验结果 | 67 |
| 5.2.4 | 案例解读 | 68 |
| 5.3 | 本章小结 | 70 |
| 第六章 | 总结和展望 | 71 |
| 6.1 | 总结 | 71 |
| 6.2 | 下一步展望 | 71 |
| | 参考文献 | 73 |
| | 简历与科研成果 | 79 |
| | 致谢 | 81 |
| | 版权与原创性说明 | 83 |

表 目 录

| | | |
|------|---------------------------------|----|
| 3.1 | 系统涉众分析 | 19 |
| 3.2 | 系统功能需求列表 | 20 |
| 3.3 | 系统非功能需求列表 | 21 |
| 3.4 | 系统用例列表 | 23 |
| 3.5 | 创建自动化测试任务用例描述 | 23 |
| 3.6 | 查看测试报告用例描述 | 24 |
| 3.7 | 查看缺陷列表用例描述 | 25 |
| 3.8 | 查看缺陷详情用例描述 | 25 |
| 3.9 | 查看多维评估数据 | 26 |
| 3.10 | 审核自动化测试任务 | 27 |
| 3.11 | ConsoleLog 类详情列表 | 35 |
| 3.12 | ConsoleLogCluster 类详情列表 | 35 |
| 3.13 | BugType 类详情列表 | 35 |
| 3.14 | PageInfo 类详情列表 | 36 |
| 3.15 | RequestInfo 类详情列表 | 36 |
| 4.1 | StateVulnerability 类详情列表 | 40 |
| 4.2 | BugDetail 类详情列表 | 41 |
| 4.3 | ProcessContext 类详情列表 | 54 |
| 4.4 | RequestEventProcess 类详情列表 | 55 |
| 5.1 | 测试环境说明 | 61 |
| 5.2 | 创建自动化测试任务相关测试用例 | 62 |
| 5.3 | 审核自动化测试任务相关测试用例 | 63 |
| 5.4 | 查看自动化测试任务相关测试用例 | 63 |
| 5.5 | 启动自动化测试任务相关测试用例 | 64 |
| 5.6 | 测试报告相关测试用例 | 64 |
| 5.7 | 维度评估相关测试用例 | 65 |

图 目 录

| | | |
|------|-------------------------|----|
| 3.1 | 系统工作流程 | 17 |
| 3.2 | 系统用例图 | 22 |
| 3.3 | 系统整体架构设计图 | 28 |
| 3.4 | 分析服务架构设计图 | 30 |
| 3.5 | 逻辑视图 | 31 |
| 3.6 | 进程视图 | 32 |
| 3.7 | 开发视图 | 33 |
| 3.8 | 部署视图 | 34 |
| 4.1 | 缺陷分析框架设计图 | 38 |
| 4.2 | 缺陷分析框架系统顺序图 | 39 |
| 4.3 | 缺陷分析框架设计类图 | 40 |
| 4.4 | 场景分析模板类的部分代码 | 41 |
| 4.5 | 控制台日志缺陷聚类子模块架构设计图 | 43 |
| 4.6 | 控制台日志缺陷聚类子模块系统顺序图 | 44 |
| 4.7 | 控制台日志缺陷聚类子模块设计类图 | 45 |
| 4.8 | 控制台缺陷日志预处理部分代码 | 45 |
| 4.9 | DBSCAN 聚类算法实现部分代码 | 46 |
| 4.10 | 聚类结果的一个示例簇集 | 47 |
| 4.11 | 控制台日志缺陷判定子模块架构设计图 | 48 |
| 4.12 | 控制台日志缺陷判定子模块系统顺序图 | 48 |
| 4.13 | 控制台日志缺陷判定子模块设计类图 | 49 |
| 4.14 | 控制台日志缺陷判定相关部分代码 | 50 |
| 4.15 | 请求日志缺陷检测模块架构设计图 | 51 |
| 4.16 | 请求日志原始格式的一个示例 | 52 |
| 4.17 | 请求日志缺陷检测模块系统顺序图 | 53 |
| 4.18 | 请求日志缺陷检测模块设计类图 | 54 |

| | | |
|------|-----------------------|----|
| 4.19 | 请求日志过程构建部分代码····· | 55 |
| 4.20 | 单个请求缺陷检测部分代码····· | 56 |
| 4.21 | 软件质量多维评估模块架构设计图 ····· | 57 |
| 4.22 | 软件质量多维评估模块设计类图 ····· | 58 |
| 4.23 | 兼容性评估相关部分代码 ····· | 59 |
| 4.24 | 某应用兼容性维度评估页面····· | 60 |
| 5.1 | 待测应用集领域和技术类型分布图 ····· | 66 |
| 5.2 | 实验对象自动化测试结果统计 ····· | 67 |
| 5.3 | 测试报告人工审查结果统计····· | 68 |
| 5.4 | 某订餐网站测试报告概况 ····· | 68 |
| 5.5 | 页面操作断链缺陷详情页面····· | 69 |
| 5.6 | JS 引用错误缺陷详情页面····· | 69 |

第一章 引言

1.1 项目背景及意义

万物互联的时代背景下，Web 应用凭借其便利性、易用性和强大的扩展支撑能力，在世界范围内广泛普及和使用，并成为了软件行业的主流趋势之一。尤其是“互联网+”概念的提出和实践，更是借由互联网的优势实现了和传统产业的融合升级，打破了行业壁垒，深入到千家万户和各行各业中，利用互联网便捷快速的信息化服务强有力地助力于人们的生产生活和日常工作，进而促进国民经济健康有序的发展。中国互联网络信息中心发布的第 45 次《中国互联网发展状态统计报告》显示¹，截至 2019 年 12 月，我国网站和网页数量分别高达 497 万和 2978 亿个，同时我国网民规模在 2020 年 3 月就已达到 9.04 亿人次，越来越多的用户和企业开始享受互联网的便利。

与此同时，随着 Web 应用的广泛使用，其质量和可用性也备受关注 [1]。一些低质量的 Web 应用，常常会因为应用缺陷、系统故障等原因导致用户的功能需求无法被满足或造成用户多次操作失败，从而极大地牺牲用户体验，面临用户流失和口碑下降的风险，严重情况下甚至会导致巨额经济损失和面临社会舆论的质疑和否定。一方面，这些缺陷可能来自于应用构建之初的设计隐患和编程疏忽，另一方面，需求的频繁变更，开发周期与成本的限制，也会带来缺陷注入和软件质量下降的风险 [2]。

软件测试，作为保障 Web 应用质量的重要手段和关键环节，开始承担越来越重要的责任。现有的主流 Web 应用测试手段，大部分还停留在依靠测试人员基于设计好的测试案例进行相应手工测试的阶段，渐渐不适应于当前快节奏的应用开发迭代进度，频繁的重复测试也会打击测试人员的积极性。基于测试脚本的自动化测试，解决了一部分测试效率问题，但也引入新的问题。测试脚本本身的构建会产生学习成本，人们日益变化增长的需求和系统变更，也会导致脚本需要频繁的维护。同时它作为一种回归测试，测试日志可读性低，评估指标单一，并不能完整有效发掘、描述和分析系统缺陷。所以如何探索一种有效的 Web 应用自动化测试解决方案，并能够对 Web 应用进行缺陷分析，生成测试报告，从而辅助测试人员更好地定位和修复缺陷，成为了我们接下来所要探究的主要问题和研究方向。

¹<http://cnnic.net.cn/hlwfzyj/hlwxbzg/>

针对这些问题，我们提出了 Web 应用自动化测试系统，针对复杂 Web 应用的执行和缺陷分析，在只提供少量配置的情况下，能够完成基于状态的 Web 应用自动化遍历执行和测试过程分析。本文就是系统后续分析服务流程的具体说明。服务基于在执行的过程中记录的待测系统相关控制台日志、请求日志、截图等多源“黑匣子”数据，利用聚类算法和特征检测等方法对其进行深度挖掘，生成可读性高可指导复现的缺陷报告，提高测试人员定位和修复缺陷的效率，规避故障风险。同时建立一套 Web 应用综合评价参考指标，实现对 Web 应用质量的多维度评估，对应用在性能、兼容性、稳定性和健壮性等方面进行参考性度量，帮助企业和开发人员更好地专项提升 Web 应用质量。

1.2 国内外研究现状

1.2.1 Web 应用自动化测试研究现状

Web 应用测试是保障 Web 应用质量的重要手段和关键环节，广义上的 Web 应用测试包括功能测试、负载测试、兼容性测试、安全测试等多种测试手段 [3]。Web 应用自动化测试，作为 Web 应用测试的一个分支，最开始是相较于传统手工测试手段而提出的一种黑盒测试方法，旨通过模拟用户操作，以能够自动执行、检查网站并验证 Web 应用的有效性。跟前者基于测试用例依次手动执行测试步骤并记录测试结果的过程不同，通过使用自动化测试工具或编写测试脚本代码的自动化测试，能够节省测试时间和人力成本，提高测试人员的测试效率，并支持重复执行，避免人的主观能动性对测试造成的影响。同时快速反馈的机制，也适用于当前快速迭代的敏捷开发的需要。

基于测试脚本的模拟执行，是现有的自动化测试主流方案。测试人员会根据前期设计的测试用例编写对应的测试脚本，交由 Selenium 及其变种等自动化测试框架转换和执行 [4] [5]，底层的 WebDriver 会负责对接和控制相应的浏览器环境，从而模拟真实的操作过程，进而大大提高测试效率。但同时，这种方案也存在一些问题。一方面，脚本的构建，需要进行场景设计、环境配置和节点获取等复杂的流程，相较于手工测试上手难度大幅上升，对测试人员的要求也较高；另一方面，Web 应用迭代周期短，在功能迭代变更的同时，很多时候会涉及到页面和操作流程的变化，进而会让测试脚本失效，因此需要专人定期维护，带来了维护成本。

针对这些问题，基于“录制-回放”和脚本组件化管理思路的自动化测试解决方案开始被提出。QTP、Selenium IDE²支持在 Chrome 和 FireFox 等浏览器上录制

²<http://seleniumhq.org>

真实操作,进而生成测试案例、测试脚本和进行回放测试。阿里的 UIRecorder³等软件还提供一端录制多端回放的测试功能,简化了脚本构建过程,让开发者以及质量保障人员更加高效去管理、设置、创建以及运行自动化测试。Katalon⁴、KylinTOP⁵等测试框架除了录制回放功能外,还通过高度可视化和可编辑化测试用例脚本,提供灵活的元素定位方式,消除了 Selenium 的技术复杂性,降低了测试难度和脚本维护成本。但是,录制和编辑脚本的过程仍是手动完成,且需要测试人员对 Web 应用的功能和业务足够熟悉。

云测试平台,则作为另一种方案,基于云测试,通过提供以测试即服务 TasS (Test as a Service) 的新型软件测试模式,完成相应测试环境、测试脚本的收录,进而完成 Web 应用的自动化测试。相较于手工测试和脚本测试,一方面保证效率,另一方面再次降低了使用难度和使用成本。Sauce Labs⁶作为 Web 云测试平台的典型代表,允许用户在 700 多种不同的浏览器平台,操作系统和设备组合云上运行测试,从而为使用 Selenium、JavaScript 等单元测试框架的自动化和手动测试提供了全面的测试基础架构,同时提供实时断点以便手动查找问题。

基于界面结构特性和状态机的自动化遍历也是 Web 自动化测试研究中的有效思路。与前几种自动化测试方案不同,利用爬网方式的自动化测试,并不会产生明确语义的测试用例,但是却能够快速遍历完成整个 Web 应用的访问,生成基于随机操作的测试用例,同时基于预定义的约束对整个应用的缺陷和异常情况进行检查,对崩溃、超时、破碎链接等情况进行报告。Benedikt 等人提出了 VeriWeb[6],使用 SmartProfiles 提取基于表单页面的候选输入值,能够通过搜寻器和检测器自动浏览多页网站的路径,查找到导航和页面错误等异常情况。Valentin Dallmeier 等人提出的 WebMate[7],能够在仅给定 URL 的情况下,自动探索 Web 应用功能,并自动生成测试数据和测试用例。Crawljax[8] 允许用户基于页面状态转换和操作事件图,通过触发事件和表单填写,完成 Web 应用程序的自动化抓取和基于不变量的测试,并通过重复检测算法避免状态爆炸 [9]。插件架构的设计也保证了其后续强大的可拓展性,F Ferrucci 等人就在 Crawljax 的基础上实现了应用的可访问性检查 [10]。

本文所述的 Web 应用自动化测试系统,就是基于最后一种自动化测试思路,通过优化 Crawljax 框架并实现多个插件,对自动化动态遍历 Web 应用过程获取的中间数据,如控制台日志、请求日志、截图等信息,进行关键的缺陷分析和多维评估,最终向用户提供测试报告。

³<https://uirecorder.com/>

⁴<https://www.katalon.com/>

⁵<http://www.70testing.com/>

⁶<https://saucelabs.com/>

1.2.2 软件测试缺陷分析研究现状

应用软件常见的缺陷来源，一种是来自用户主动上报，缺陷舆情监控等手段获取的非结构缺陷，另一种则是借由报错日志、截图等中间信息，经过程序分析后预警缺陷，并人工填写缺陷报告。目前也已经有一些工具来完成这些信息的采集和定位，如美团的 Logan 框架⁷，可在 Web、Android 等多端环境运行，并提供一整套的日志体系，包括日志的收集存储，上报分析以及可视化展示，最终完成应用的缺陷定位和分析。但是这两种信息渠道都存在相同的问题，即都属于被动触发，仍需要耗费大量时间来完成缺陷的复盘与报告。因此本系统希望能够通过自动化遍历，主动触发缺陷，并结合过程数据，如日志信息，来生成拥有完整缺陷上下文，且可复现的缺陷报告。

而在基于日志定位缺陷的研究领域上，很多算法和模型已经被提出，以用于自动异常检测、智能故障诊断等场景，根据日志训练数据集的标记与否，总体可分为有监督模型和无监督模型，常见的有监督异常检测模型包括逻辑回归、决策树和 SVM 等算法，无监督模型主要是日志聚类、不变量挖掘以及 PCA 等算法。具体地，Mike Chen 等人使用决策树进行故障诊断，能够从大量潜在缺陷原因定位真实原因，且误报率较低 [11]；Yinglung Liang 等人比较了 SVM 以及多种最近邻分类方法在分析日志和预测故障上的效果，发现定制的最近邻算法在覆盖范围和精度上更好 [12]；Min Du 等人推出了 DeepLog [13]，通过深度学习方法来对系统日志中进行异常检测和诊断；南京大学周志华教授则提出了孤立森林 [14]，通过隔离实例来进行异常的检测，且不依赖于任何距离和密度测量，线性的时间复杂度带来效率的巨大提升；Jian-Guang Lou 等人则提出一种非结构化日志分析技术 [15]，通过自动发现日志中的程序不变性，来自动检测日志中可能存在的异常。

聚类算法也是缺陷分析中可用的分析手段，常见的聚类算法主要分为五类，分别是划分法、层次聚类法、以及基于密度、网格和模型的聚类，适用于不同的分析场景。微软的 Qingwei Lin 等人就基于聚类算法，提出了 LogCluster 工具 [16]，用以简化日志缺陷诊断，定位应用缺陷。不同于后台日志，本文需要深度分析和缺陷检测的日志数据主要为控制台日志信息，缺乏大规模和规范化的数据集，以及统一普适的缺陷类别。结合前期自行收集的控制台日志数据集规模，同时综合系统技术方案、实现成本等多因素来看，基于密度的聚类方法，是最适合本项目中日志数据的分析处理方案。

⁷<https://github.com/Meituan-Dianping/Logan>

1.3 本文的主要工作

为解决手动测试效率问题和降低脚本测试复杂度,更好地发掘应用缺陷,保障 Web 应用的质量,本文主要阐述的分析服务,作为 Web 应用自动化测试系统的核心服务,在低成本高效率自动化遍历测试的基础上,旨在通过 Web 自动化测试执行产生的多元数据,构建 Web 应用缺陷体系,对待测应用进行相应的测试分析,从而为开发运维人员提供高效便捷的测试报告。本文的主要工作如下:

(1) 基于现状调研和前期分析,并考虑后期扩展的场景下,拆解了功能需求和职责,设计和实现了包括缺陷分析框架、控制台日志缺陷检测模块、请求日志缺陷检测模块、Web 应用多维评估等多个模块在内的分析服务,满足了用户轻量化的自动化测试和高可用测试报告需要。

(2) 设计和实现了缺陷分析框架,定义了整个缺陷检测的基本流程和接入标准,以任务场景为粒度,流水线模式对接上游服务,提高了应用的性能。同时支持其他数据源和缺陷检测技术的扩充,提高了应用的可拓展性。

(3) 设计和实现了控制台日志缺陷检测模块,一方面,针对不具备规模化特点的控制台日志,基于前期的日志搜集,通过聚类算法和人工审查,确定了 Web 应用缺陷知识库;另一方面,通过知识库完成对测试过程应用输出的控制日志的缺陷判定和上下文整合,实现高可用测试报告和缺陷的快速定位。

(4) 设计和实现了请求日志缺陷检测模块,对 Web 应用的网络请求过程进行建模还原,获取真实场景下的请求数据,对一些可能存在的异常情况进行检测,如断链、服务器内部错误等等,并产出相关性能数据服务于后续分析过程。

(5) 设计和实现了 Web 应用多维评估模块,从健壮性、稳定性、性能等多维度对 Web 应用进行深度解读,提供给用户除了缺陷报告外更多的应用信息和改进方向,帮助用户进一步提高应用质量。

结合上述的整个设计与实现方案,整个分析服务基于 SpringBoot 暴露给下游组件,并通过 RabbitMQ 实现异步化调用,保证了服务与服务之间的解耦。MongoDB 和 Redis 则做为数据存储层,保证了数据的可持久化与高效访问。同时本系统合理地使用缓存,结合功能场景应用设计模式,更是进一步提高了应用本身的质量属性。

1.4 本文的组织结构

本文主要分为六个章节,具体的组织结构如下:

第一章为引言部分,主要介绍了 Web 应用自动化测试系统分析服务的项目背景及意义,并从 Web 应用自动化测试和软件测试缺陷分析两个方向分析国内

外研究现状，最后简要介绍了 Web 应用自动化测试系统分析服务的主要工作和论文的组织结构。

第二章为相关技术部分，主要针对项目中涉及到的相关核心技术框架和算法，做进一步的介绍和分析。总体可划分为三大技术板块，即 Web 应用测试相关技术，包括 Selenium、Crawljax 框架等；服务端相关技术，如 Redis、MongoDB 等，以及日志挖掘相关算法，如聚类算法，Levenshtein 距离等。

第三章为系统需求分析与概要设计部分，一方面，对 Web 应用自动化测试系统进行整体的涉众分析和需求分析，明确了项目的功能需求和非功能需求，并通过用例图和系统用例描述等形式进行详细说明；另一方面，阐述了系统的整体架构，并对项目的分析服务部分进行概要设计，从服务架构与模块划分、4+1 视图、持久化模型等初步设计内容进行展开分析。

第四章为系统详细设计与实现部分，在前期概要设计的基础上，细化了缺陷分析框架模块、控制台日志缺陷聚类 and 判定模块、请求日志缺陷检测模块以及 Web 应用多维评估模块的设计和实现，并提供了核心部分的相关局部代码或伪代码。

第五章为系统测试和实验分析部分，一方面，全面测试了整个 Web 自动化测试系统，并验证了分析服务的主要功能执行过程，保障了系统的可用和可靠；另一方面，对现实中的 50 个真实网站，进行了系统实验，以评估测试分析结果的有效性。

第六章为总结和展望部分，主要总结了在本项目开发和撰写论文过程中的相关工作，分析了进行下一步工作的主要方向，并就 Web 应用自动化测试分析的未来前景做进一步展望。

第二章 相关技术

2.1 Web 应用测试相关技术

2.1.1 Selenium

Selenium 是一个完全开源、免费使用的 Web 应用自动化测试工具，通过 WebDriver 直接运行在真实浏览器或无头浏览器中，能够模拟真正的用户操作。其 API 简单易上手，支持功能测试、回归测试、兼容性测试等多种测试类型，兼容 Linux、Windows 等多个平台版本和包括 Chrome 和 Firefox 在内的所有主流浏览器类型，并支持分布式测试用例执行。成熟的社区和大量文档支持也是很多企业在测试选型时考虑它的主要因素。它目前主要包括以下几个组件：

(1) WebDriver: WebDriver 是一套操作页面的规范和协议。作为 Selenium 2.0 的主要技术，不同于 1.0 版本的 Selenium RC 通过嵌入 JavaScript 来控制浏览器，WebDriver 针对于不同浏览器不同的工作原理和内部执行实现，抽象了彼此之间的这些差异，通过使用同一套 API，并由浏览器供应商负责其浏览器的驱动程序实现，隐藏了浏览器内部细节和复杂性，使用户通过简易的代码就可以在不同浏览器环境上执行相同的复杂 workflows。同时支持 Java、Python、C 和 Ruby 等多种编程语言编写的自动化测试脚本，可以模拟最终用户的常见活动，如输入文本、下拉选择、鼠标点击和移动等，用以创建复杂的测试场景，并驱动相应浏览器完成操作。

(2) Selenium IDE: Selenium IDE 是用通过录制用户操作，来开发 Selenium 测试用例的工具，并通过 Chrome 和 Firefox 拓展程序来进行支持。它使用现有的 Selenium 命令，通过获取元素上下文参数，完成用户操作的录制和测试用例的管理，并支持测试脚本代码的导出。这不仅降低了测试脚本编写和管理的难度，同时也是一种学习 Selenium 语法的友好实践。

(3) Selenium Grid: Selenium Grid 允许用户在多机器多平台环境下运行测试用例，主要针对规模庞大测试案例集和兼容性测试需求的分布式测试提供拓展支持。它可以在主环境控制触发测试用例，并在触发测试用例时，由远程端环境自动执行对应的测试脚本。

尽管 Selenium 本身已经支持包括网站前端测试在内的多种测试类型，但其核心的浏览器代理库和透出的接口，也让它具有较好的二次开发和组件集成支撑基础，因此很多研究和测试工具都选用了 Selenium 作为底层实现。

2.1.2 Crawljax

Crawljax 是一个基于 Java 的开源自动化测试工具，可以自动化抓取和测试 Web 应用程序。它基于事件驱动的动态爬网引擎，能够通过触发事件和填充表单数据对任何基于 Ajax 的 Web 应用程序进行抓取，输出动态 DOM 状态及其基于事件转换的状态流图，为 Web 自动化测试奠定了强大的基础，如回归测试、基于不变性的测试以及缺陷分析等等，插件架构的设计也保证了其后续强大的可拓展性。

Crawljax 实现了一种算法，利用元素名称和属性约束标记了一组候选元素，并暴露于不同的事件类型下，如 click、mouseOver 等事件。对于每个候选元素，Crawljax 会自动触发该元素的对应事件类型，并将触发后的 DOM 树与触发前进行对比，用以确定是否改变当前的状态。如果检测到更改，则会创建一个新状态并添加到有限状态机的状态流图中，该事件则作为连接前序状态和当前新状态的边。最终通过递归调用，从而完成整个 Web 应用的遍历扫描。

默认情况下，Crawljax 会为输入字段提供随机的输入值，但也支持用户指定输入数据，以满足一些特定的输入约束并正常进行流程，如用户名和密码的输入。Crawljax 通过在新状态下扫描整个 DOM 树的表单元素，并根据输入属性和 HTML 结构来计算哈希码，并和对应的输入数据进行匹配。

同时 Crawljax 支持插件扩展，提供了包括 OnNewStatePlugin 在内的多种基础插件，分别对应执行过程中不同的阶段，如加载 URL 前，加载 URL 后和检测到应用新状态等。用户可以扩展这些插件，以实现自己的监控和约束需求。我们就是利用了 Crawljax 的这种插件架构，完成了测试执行中对应过程数据的采集。

2.1.3 Chrome Devtools Protocol

Chrome Devtools Protocol，又称 Chrome Remote Debugging Protocol，是基于 WebSocket 建立连接 Chrome 系浏览器内核和 DevTools 的快速数据通道，提供与浏览器的多个部分（例如页面，服务工作者和扩展程序）进行交互的 API，并允许第三方使用工具来对 Chrome、Chromium 和其他基于 BLink 的浏览器进行测试、检查、调试以及配置，如操作浏览器、获取网络信息等。

它把不同的操作划分成不同的域，如 DOM、Network 和 Page 等。每个域会负责不同的功能模块，定义了它相应支持的一系列命令和生成事件，并按照固定结构组织成序列化的 JSON 对象，以便第三方与页面建立通信时，组成相应的数据消息。例如，Network 域可以跟踪页面的网络活动，Page 域可以获取当前页面数据等等。

很多工具都使用了 Chrome Devtools Protocol 作为扩展和支撑, 包括 PhantomJS, Selenium 的 ChromeDriver 等等, 本质上都是建立在 Chrome 内核提供的 API 上调用。Crawljax 由于底层借助了 WebDriver 实现对浏览器的控制, 因此可以通过 ChromeDriver 来采集自动化遍历过程中以 Chrome Devtools Protocol 协议形式进行组织的性能日志数据。在本文中, 我们就是借助了这些日志和 Chrome Devtools Protocol 数据组织格式来完成了 Web 应用请求过程的重新建模和缺陷分析等活动。

2.2 服务端相关技术

2.2.1 Redis

Redis, 全称为 Remote Dictionary Server, 本质上是基于 C 语言编写的单线程高性能 Key-Value 内存数据库 [17], 并面向多种程序设计语言提供 API 支持, 是当前业界在考虑系统高并发、高可用和高可扩展等质量因素时常考虑使用的技术之一。作为一种非关系型数据库, 因为其基于内存存储和单线程执行的缘故, 跟传统数据库相比较起来, 它的性能非常高, 每秒可以完成 10 万次左右的读写操作, 常被用作高速缓存、数据库和消息中间件等。

不同于大多数其他非关系型数据库只支持字符串型数据 [18], Redis 支持包括 List、Hash、Map、Set 和 Sorted Set 在内的多种数据结构类型, 在 Redis5.0 版本, 还引入了 Stream 这一新的数据类型, 用以更为抽象地模拟日志数据结构。使用者可以根据其业务场景的不同需要选择使用不同的数据类型。常见地, 业界常用 Set 实现高性能 Tag 系统, 使用 Hash 计算页面访问量, 借助 Sorted Set 数据结构维护排行榜等等。

受限于物理内存限制, 为了 Redis 的内存空间使用完毕时保证新数据的引入, Redis 还支持对存入的数据设置失效时间, 并提供包括 allkeys-lru、volatile-lru 等在内的多种数据淘汰策略, 能够自动地清理失效数据, 以保持 Redis 的可用, 并将数据的退出维护与开发人员相解耦, 提高了开发效率。

另外, 不同于 memcached 在服务重启时会有数据丢失的风险, Redis 提供 RDB 和 AOF 这两种持久化方式, 将内存的数据保存在硬盘中, 进而保证数据的可持久性。其中 RDB 是一种全量保存策略, Redis 会按照一定的时间周期或频率, 定期将内存数据刷入磁盘, 以便下次读取文件进行加载; 而 AOF 是一种增量持久化策略, Redis 在进行操作时会不断地记录日志, 并在下次重启服务时, 重放日志操作来加载数据。

Redis 还支持主从模式下的数据备份和分布式集群，推出了高可用解决方案 Redis Sentinel（哨兵），用来保证 Redis 节点的故障发现和故障自动转移，并提供配置中心和客户端通知，保证了 Redis 的高可用。后续分布式集群解决方案 Redis Cluster 的实现，除了提高了 Redis 的可用性之外，更是保障了其在读写和容量等方面的水平扩展能力。

2.2.2 MongoDB

MongoDB 是一个由 C++ 语言编写的基于分布式文件存储的高性能数据库，能够为软件应用提供高可用和可扩展的数据存储解决方案，且易于使用和部署，常用于实时数据处理、缓存、大数据和其他需要高伸缩性的场景 [19]。

作为一个最接近关系型数据库的非关系型数据库，MongoDB 融合了关系型数据库的优点，并将数据库可分为多个集合，在集合内则面向文档结构来进行存储。文档由键值对组成，字段值可以包含其他数组，也可以嵌套其他文档以及文档数组，是一种十分松散并称之为 BSON 的类 JSON 格式。这种格式既保证了访问的高效，同时也保证了数据原有的组织形式，是最自然、最有效处理数据的方式，比传统的行列模型更加具有表现力和功能，使用上十分地灵活 [20]。

MongoDB 还提供了非常强大和丰富的查询语言，类似于大多数关系型数据库单表查询方法的实现，支持任意字段的过滤和排序操作，以及聚合等复杂查询，能够更好地搜索文档结构空间，帮助用户简单准确表达业务语义。同时，为了避免全表扫描，提高查询效率，MongoDB 也支持对数据建立索引，包括单字段索引、复合索引、多 Key 索引和文本索引等，我们可以根据使用场合的特点选择和使用最合适的索引类型。

MongoDB 支持多种高可用集群的搭建，包括 Replica Set、Sharding 和 Master-Slaver 等。其中，Master-Slaver 是最为简单的一种，即最常见的主从备份，现在官方已经不再推荐这种方案。Replica Set 机制则主要引入了新的仲裁节点。集群中会包含多份数据，以便在主节点宕机时，备节点能够继续提供数据服务。仲裁节点本身不存储数据，主要负责主节点宕机时备节点的选择。Sharding 机制类似于 Replica Set，不同的是，除了仲裁节点，Sharding 还引入了配置节点和路由节点，配置节点会负责和维护存储集群所有节点和分片数据的路由信息，路由节点则是服务的访问入口。

在本文中，我们使用 MongoDB 作为数据存储的主要中间件，并借助 Spring-Boot 框架来实现快速接入，希望在提高性能，满足数据存储和安全的基本要求上，能够对具有一定层次结构的测试报告、缺陷信息等数据进行友好管理和查询，简化数据组织的过程，并避免关系型数据库表连接带来的性能损耗。

2.2.3 RabbitMQ

RabbitMQ 是基于高级消息队列协议 (AMQP) [21] 架构设计, 通过 Erlang 语言编写的一个轻量级开源消息队列中间件, 并面向包括 Java、Python 等在内的多种程序设计语言提供客户端支持。它最开始服务于金融行业, 现已成为业界使用的主流消息队列之一, 常用使用场景包括异步调用、解耦以及填谷削峰等等, 具有较好的易用性、扩展性、数据一致性以及高可用性。

根据 AMQP 协议的设计, RabbitMQ 主要包括消息生产者 (Producer)、消息消费者 (Consumer)、消息交换机 (Exchange) 和消息的存储载体队列 (Queue) 这几个核心组件 [22]。其中, 消息生产者负责发布消息, 由交换机根据对应分发规则传递给绑定的消息队列, 消费者则负责从队列中订阅或主动拉取相应的消息进行消费。在这个过程中, Routing Key 主要负责配置路由规则, Binding 则会按照该路由规则绑定相应的 Exchange 和 Queue, 作用于 RabbitMQ 内部结构的组装和编排。同时, 由 ConnectionFactory 负责创建的 Connection, 会建立起生产者、消费者跟 RabbitMQ 之间的 Socket 连接, 为上层负责实际业务操作的 Channel 对象提供信息通道。

另外, Rabbit 支持多种消息分发模式, 包括生产者消费者一一对应的简单模式、一个生产者对应多个消费者, 不同消息被多个消费者并行进行唯一消费的工作队列模式、借助 “fanout” 交换器实现消息在多消费队列中进行广播的发布/订阅模式、借助 “direct” 交换器实现多队列按路由键分发消息的路由模式, 以及借助 “topic” 交换机进行通配符比较路由键的主题模式等等, 使用者可结合自己的业务场景和技术方案需要, 采用相应的消息分发模式。

同时为了确保消息的正常投递和正常消费, RabbitMQ 也提供了两种消息确认机制。一种是消息发送确认, 用户可以实现 ConfirmCallback 和 ReturnCallback 回调接口, 分别确认消息是否成功发送到交换机和是否成功发送到队列。另一种是消息接收确认, 用以确认消费者是否成功消费了队列中的消息, 可细分为三种具体的接收确认模式, 即不确认、手动确认和自动确认。

RabbitMQ 的消息默认存放在内存之中, 以保证使用上的高性能, 但同时也会带来数据的丢失风险, 因此 RabbitMQ 也提供持久化策略, 使其在牺牲性能的情况下, 保证故障恢复能力。当消息投递模式、交换机配置和队列配置均为持久化的情况下, RabbitMQ 会通过写日志的方式, 维护每一条未消费的消息。一旦出现故障, 它就会自动重建交换机、绑定和队列对象, 并通过重放日志来恢复相应的消息。在系统中, 我们主要是通过 RabbitMQ 来完成分析服务的异步调用等工作, 并通过流水线编排模式, 更好地提升系统的性能。

2.2.4 SpringBoot

SpringBoot 是 Spring 社区提供的一个开源框架，希望能够帮助使用者快速简便地搭建项目框架，并使相关的项目开发、配置和部署等阶段工作更为简单便捷，降低了开发的难度和复杂度，让开发人员能更好地专注于业务逻辑。

一方面，它继承自 Spring 框架，保留了 Spring 本身的优秀特质，如 Java 对象的生命周期管理，IOC 和 AOP 机制等等，为开发 Java 应用程序提供了基础架构支持。另一方面，它使用习惯优于配置的理念，相比 Spring 框架需要配置大量基于 XML 文件的固定模板，SpringBoot 会为其提供默认值，来促进项目的快速搭建与开发 [23]。同时支持开箱即用，通过注解，实现自动加载对象和生命周期管理，简化了整个项目的配置过程。

SpringBoot 还为用户定制了一系列的 starter 启动器，简化了我们对于 Maven 依赖项的配置加载过程，减少了依赖版本冲突，便于用户更好地集成第三方依赖和框架，如 Reids、MongoDB 等等，而不用专门去维护原先错综复杂的依赖关系 [24]。其次，SpringBoot 可以内嵌 Tomcat、Jetty 或者 Undertow 等 Web 容器，最终以 jar 包形式打包和部署整个项目，而无需配置额外的 Web 容器环境，简化了项目部署的流程和成本，从而实现快速发布。

此外，SpringBoot 也提供了一些常见的生产准备特性，如指标、外部化配置和健康监测等等，可以集成非常高效的监控插件，以监控各项组件的运行状况和性能。同时在配合 SpringCloud 提供的服务发现与服务注册，以及安全管控等方案的支撑下，SpringBoot 在微服务领域也十分受欢迎，常作为微服务架构的底层支持。本文的系统就是基于 SpringBoot 框架来提供服务的。

2.3 缺陷日志挖掘相关算法

2.3.1 缺陷日志聚类算法

聚类是一种广泛使用的无监督学习技术 [25]，在缺乏先验知识支撑和标记信息的情况下，可以将训练样本切分成不同的簇，使同簇之间数据样本的相似性和不同簇之间数据样本的差异性都尽可能高，从而帮助人们更好地揭露训练样本的结构、内在联系等等，以发掘数据中潜在的知识和模式，为进一步的数据挖掘提供基础，特别适合对未知数据集进行分析。

根据 Jiawei 等人提出的分类 [26]，目前聚类方法主要分为划分法、层次聚类法、基于密度的聚类、基于网格的聚类以及基于模型的聚类。其中，划分法需

要预先指定分类数量，代表算法有 K-means、K-medoids 等，简单快速，效率较高，但强依赖于预先设定的 K 值，对离群点和噪声点敏感 [27]；层次聚类法，包括凝聚型聚类和分裂型聚类，不需要提前设定分类数量，且可以发现类的层次关系，但计算相对复杂，极端情况会成链状结构 [28]；基于密度的聚类，典型算法有 DBSCAN，它同样不需要提前设置 K 值，且不受噪声点的影响，能发现任意形状和各种大小的类群 [29]；基于网格的聚类，常见算法有 STING，会将数据划分成有限单元的网格结构，并逐单元处理数据，易于增量实现和进行高维数据处理 [30]。基于模型的聚类，如 GMM 高斯混合模型、神经网络模型等，不同的模型设计和处理方法各有差异 [31]。

由于控制台日志缺乏大规模和先验知识支撑的数据集，且无法对缺陷类别数量做出精确化指定，在基于聚类成本和精度不做严格要求的情况下，我们最终选择了基于密度的聚类方法，通过实现 DBSCAN 算法，来对自动化采集的小规模控制台日志数据集进行聚类分析。作为密度聚类算法的典型代表 [32]，DBSCAN 的基本实现思路是，从任意节点开始，根据邻域半径和密度阈值，判定该节点是否为核心对象，并在为核心对象的情况下，找出该点所有密度可达的节点对象，进而形成一个聚类集群并标记当前类别，反之则开始下一轮节点选择，直到所有的节点都被访问过。根据这个思路，本文实现了 DBSCAN 算法，并基于日志的文本相似度反向构造了新的距离函数，希望能够直观地发掘应用可能存在的缺陷类别，构建初步的 Web 应用缺陷体系。

2.3.2 Levenshtein 距离

Levenshtein 距离，又称字符串编辑距离，是数学家 Vladimir Levenshtein 提出的一种衡量字符串差异的距离计算方式 [33]，在很多领域都有应用，如自然语言处理、拼写检查、抄袭侦测以及生物信息学的 DNA 分析等等。

它是指将一个字符串 A 转换成字符串 B 所需要的最少编辑操作次数，直接从字面上反映文本间的差异程度。其中，编辑操作包括：增加一个字符、删除一个字符和替换一个字符。通过 Levenshtein 距离，我们可以得到基于如下公式的文本相似度计算方式：

$$\text{Similarity} = 1 - \frac{\text{Levenshtein}_{AB}}{\max(L_A, L_B)} \quad (2.1)$$

其中，A、B 为待比较的两个字符串， Levenshtein_{AB} 代表 A、B 这两个字符串之间的编辑距离， L_A 、 L_B 则分别代表 A、B 字符串的长度，最终计算得到 A、B 之间值域为 [0, 1] 的文本相似度值。值越大，说明文本相似度就越高。鉴于控

制台缺陷日志的结构特点和长度限制，我们也采用了这种基于 Levenshtein 距离的文本相似度计算，来反向计算节点距离，服务于 DBSCAN 的聚类过程。

2.3.3 Jaccard 系数

Jaccard 系数，又称为 Jaccard 相似系数，常用于定量化比较两个有限样本集合之间的相似性和差异性，可通过计算两个集合元素之间的交集和并集的比值进行计算，值域为 $[0, 1]$ ，Jaccard 系数值越大，对应集合的样本相似度就越高。如以下公式所示：

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \quad (2.2)$$

其中，A, B 为任意两个样本集合。同时，Jaccard 系数也可以用于度量不同样本之间的文本相似度。由于控制台缺陷日志具有天然的英文空格分词特征，所以我们也选择了 Jaccard 系数来作为缺陷日志之间文本相似度和节点距离的计算策略。

2.3.4 轮廓系数

为了保证聚类这种非监督学习方式的效果，对聚类结果进行评估也是我们需要考虑的一大方向。根据先验知识的有无，评价指标可分为外部指标和内部指标两种 [34]，外部指标在评价过程中需要借助数据真实情况进行对比分析，内部指标不需要其他数据就可进行评估。常见的外部指标包括兰德系数 (ARI)、调整互信息 (AMI)、V-measure [35] 以及 FMI 等，内部指标则主要包括轮廓系数以及 Calinski-Harabaz 指数。其中，轮廓系数，是 Peter J. Rousseeuw 在 1986 年提出的一种内部度量指标 [36]，通过计算同簇之内聚度和不同簇之间的分离度，来反应聚类效果的好坏，可如以下公式所示：

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \quad (2.3)$$

其中， $s(i)$ 表示样本 i 的轮廓系数， $a(i)$ 表示样本到同簇其他节点的平均距离， $b(i)$ 则表示样本到其他簇平均距离的最小值。轮廓系数的值域为 $[-1, 1]$ ，其值越接近 1，说明聚类结果越合理。最终，所有样本的轮廓系数平均值，为整体聚类结果的轮廓系数值。鉴于当前控制台日志并没有先验知识，且后续会多次迭代聚类，因此我们选择采用轮廓系数来衡量我们整个聚类的效果，并确定最后分析使用的聚类结果数据。

2.4 本章小结

本章主要介绍了项目中所涉及和使用的相关技术和算法，从 Web 应用测试相关技术、服务端相关技术以及缺陷日志挖掘相关算法三部分进行展开。首先详细介绍了 Selenium、Crawljax 等工具以及 Chrome Dectools Protocol 协议，这是我们进行后续分析工作的概念和框架基础。其次，介绍了 Redis、MongoDB、RabbitMQ 以及 SpringBoot 等技术框架，它们是项目框架的构成要素，以满足项目实现需要，同时提高系统质量。最后，介绍了我们在实际进行缺陷分析时可能涉及的相关分析算法和指标，如聚类算法、轮廓系数等，它们是系统分析能力的有效保障。

第三章 系统需求分析与概要设计

3.1 Web 应用自动化测试系统整体概述

随着 Web 应用的广泛使用和功能的愈趋复杂，人们对 Web 应用的质量要求也随之提高，以手工测试和脚本测试为主流的 Web 应用测试被大量运用于应用开发生命周期中。但随着敏捷方法的推行和开发上线周期的缩短，传统的手工测试渐渐不能满足现阶段 Web 应用的测试需要。自动化脚本测试，也由于系统经常性变更的缘故，对测试人员的能力和测试脚本的维护都提出了较高的要求，同时测试日志可读性也不高，出现测试未通过情况仍需要开发人员后续排查具体缺陷。

为解决这些问题，我们设计并实现了本文主要提及的 Web 应用自动化测试系统，旨在针对 Web 应用，提供高效可靠的自动化测试和缺陷分析等功能。一方面，希望尽可能减轻测试人员的工作量，追求以轻量化的配置，甚至基于一个单一的测试应用入口 URL，就能完成 Web 应用的自动化遍历测试；另一方面，也能够提供自动化测试后相应的缺陷报告，自动分析 Web 应用可能存在的缺陷，以帮助开发人员快速定位和修复缺陷，提高测试效率。

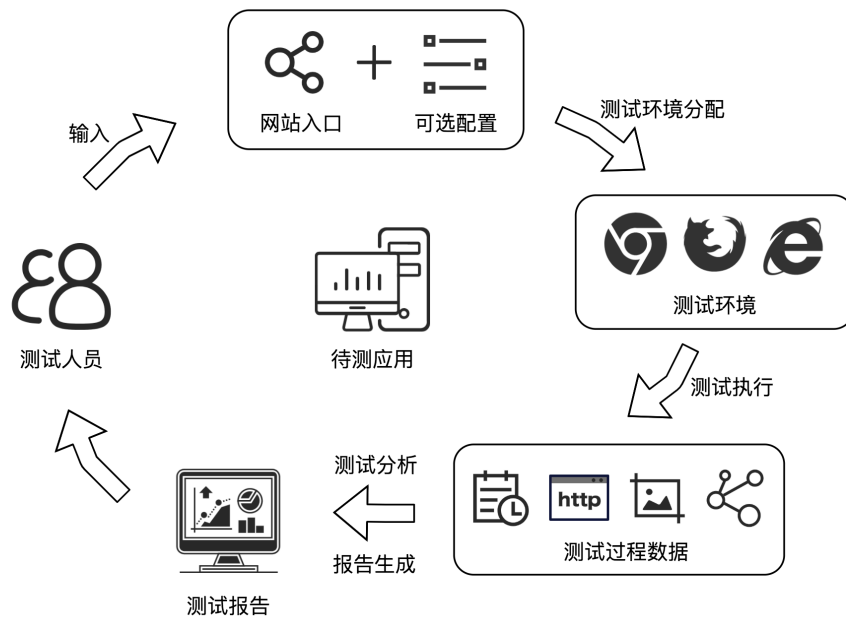


图 3.1: 系统工作流程

整个系统的工作流程如图 3.1 所示，测试人员通过提供待测网站的网站入口 URL，即可开启一项自动化测试任务。可选配置作为辅助和进阶操作，负责帮助用户在自定义使用场景下进行测试环境配置和权限配置等复杂操作。之后，测试任务会按照对应配置，分配对应任务操作系统和浏览器类别的测试环境，并在测试环境中完成对待测应用的自动化遍历测试，同时在测试过程采集相应的数据，如控制台日志、网络请求日志、页面截图以及操作事件流等。最终经过数据挖掘和数据建模，生成一份高可用、高可读和缺陷可复现的 Web 应用测试报告，并支持触发缺陷事件流对应测试脚本的生成与导出，以帮助测试人员更好更快更准的定位和修复缺陷，保障应用质量。

3.2 Web 应用自动化测试系统分析服务概述

分析服务，作为整个 Web 应用自动化测试系统的核心服务之一，简单来说，可以看做是一个基于多源自动化测试过程数据的 Web 应用测评器，旨在多方位评估 Web 应用，并发掘和分析其可能存在的缺陷和故障，如常见的断链、JS 类型错误、引用错误、服务器内部错误和资源请求失败等问题，结合缺陷对应页面状态中的上下文关联信息，如截图、操作序列等，构建出完整的缺陷模型，便于定位和复现缺陷。同时从健壮性、性能和稳定性等多个维度对 Web 应用进行切面总结和评估，以帮助专项提升应用质量。

在这个过程中，不同于周期记录和业界研究已久的服务端日志，在 Web 应用运行过程中输出的控制台日志和网络请求信息，虽然开发和运维人员也常常通过浏览器调试工具右键进行检查来发现缺陷，但并没有形成规模和定期采集的日志数据集，以及完全统一的缺陷体系。因此，在分析服务中，我们在前期调研和日常经验的基础上，确定了初步的缺陷种类和框架，再通过数据采集和数据挖掘等技术手段，对缺陷日志进行分类，以挖掘更多隐藏的缺陷。

其次，考虑到后续系统的功能扩展和能力增强，我们也针对后台日志等其他数据源的缺陷定义和检测算法提供扩展性支持，并满足后续新增评估维度的快速接入需要。

3.3 Web 应用自动化测试系统需求分析

需求分析是明确涉众和需求，并以规范化科学化的形式记录和分析需求，来确保软件是否达到预期的工程方法和工作过程 [37]，也是软件开发生命周期的核心组成步骤之一。下面本文将根据需求工程有关方法和对系统业务的分析结果，从涉众分析、功能性需求和非功能需求等多方面明确归纳系统的整体需求。

3.3.1 涉众分析

本系统主要聚焦于自动化测试 Web 应用的过程，因此涉众主要包括与 Web 应用直接相关的开发人员和测试人员，以及本系统的系统管理员。其各自具体的涉众特征与期望，如下表 3.1 所示，开发人员完成相应迭代阶段的 Web 应用产物，发布在相应的测试环境或线上环境后，即会通知测试人员进行测试。测试人员可通过本系统实现软件的自动化遍历测试，评结出高效和可读性高的测试报告，提供给开发人员修复缺陷。系统管理员，则作为网站的守护者，避免浪费系统测试资源和保障系统的稳健运行。

表 3.1: 系统涉众分析

| 涉众类别 | 涉众特征与期望 |
|-------|--|
| 开发人员 | 具有较强的软件技术实力和行业领域知识，熟悉常见的 Web 应用开发技术和主流框架，负责 Web 应用的功能实现和需求满足，是 Web 应用的构建者，并能在后续的测试发现缺陷缺陷的情况下，根据测试人员提供的测试报告，对大部分缺陷进行排查和修复。期望缺陷的主动发现和快速修复，规避线上故障的风险。 |
| 测试人员 | 具有一定的专业测试能力，熟悉各种计算机环境和常用软件测试工具，负责对迭代完成的应用进行详尽的测试，并填写测试报告，以供开发团队参考，同时对测试中发现的缺陷进行定位，与开发人员讨论缺陷解决方案，是软件质量的守护者。期望更方便快捷地对 Web 应用软件进行测试，减轻工作强度，更快更准地定位和复现产品缺陷，进而提高测试效率。 |
| 系统管理员 | 作为 Web 应用自动化测试系统的管理员，熟悉系统业务和功能，有丰富的维护、管理和分析信息系统经验，了解互联网法律法规，确保系统稳定健康的运营，同时对系统资源进行管理和维护，防止系统过载。期望能对测试任务进行较好的管理，避免测试资源浪费和恶意提交。 |

3.3.2 功能性需求

功能需求，作为一个软件产品的核心需求，是软件创造价值的基础。在这一环节中，除了常规的用户管理功能（如注册、登录、用户信息管理等）未详细进行展开外，我们根据项目背景和涉众分析的结果，从自动化测试任务管理和最终产出测试报告等方面出发，总结了如下表 3.2 所示的 Web 应用自动化测试系统功能需求列表。

在测试人员的业务流程中，验证身份后的测试人员首先要可以创建 Web 应用自动化测试任务，然后通过填写测试目标、添加测试约束、配置测试环境和可选的测试应用权限配置，完成测试任务的信息补全，使测试任务处于待审核状态。待测试任务审核通过后，测试人员可点击执行操作，通知系统按照前期配置执行测试任务的自动化遍历和测试分析，并随时查看当前任务状态。待测试完

成，测试人员就可以查看对应包括测试概况、缺陷报告以及多维评估等信息的测试报告，从而完成一次完整的测试流程。如果测试任务中途执行失败，测试人员还可以点击重测操作，重新对应用进行测试。

在系统管理员的业务流程中，验证身份后的管理员可以查看所有 Web 应用自动化测试任务的当前状态和任务详情，并对需要审核的测试任务，根据系统实际资源情况和任务详情，决定该测试任务是否通过审核，可进行后续的执行和分析服务。

表 3.2: 系统功能需求列表

| 需求编号 | 需求名称 | 需求描述 |
|------|-------------|--|
| RQ1 | 创建自动化测试任务 | 系统应该允许测试人员创建一个 Web 应用自动化测试任务，即输入需要测试的网站 URL，并结合测试需求可选性添加测试约束，如最长测试时间、最大页面状态数等。 |
| RQ2 | 配置测试环境 | 系统应该允许测试人员对测试任务进行环境配置，选择测试任务需要遍历执行的测试环境，包括测试平台类型、浏览器类型和浏览器数量。 |
| RQ3 | 测试任务权限配置 | 系统应该允许测试人员对测试任务进行配置来跳过 Web 应用的权限校验，包括配置权限验证请求和表单相关的参数等，以让系统能够遍历到后续的测试状态。 |
| RQ4 | 审核自动化测试任务 | 系统应该允许系统管理员对测试人员创建的 Web 自动化测试任务进行审核，判断是否允许发起该自动化测试任务。 |
| RQ5 | 查看自动化测试任务列表 | 系统应该允许测试人员查看当前账号下创建的所有自动化测试任务列表，并在当前各任务状态基础上显示相应后续操作以供用户执行。 |
| RQ6 | 启动自动化测试任务 | 系统应该允许测试人员对审核通过但还未执行或执行失败的自动化测试任务开启自动化测试，并监控测试任务进度。 |
| RQ7 | 查看测试报告 | 系统应该允许测试人员查看测试已执行完成的任务对应的测试报告，同时系统还应该提供离线报告下载功能。 |
| RQ8 | 查看缺陷列表 | 系统应该允许测试人员点击测试报告中的缺陷列表入口查看所有缺陷的列表，并提供列表筛选功能，筛选条件包括缺陷类别、严重等级等。 |
| RQ9 | 查看缺陷详情 | 系统应该允许测试人员点击缺陷列表中某个缺陷查看缺陷详情，并显示缺陷相关的上下文信息，包括问题路由，当前状态、缺陷类型、缺陷日志、页面截图以及进入该缺陷相关状态页面的操作步骤等内容。 |
| RQ10 | 查看多维评估信息 | 系统应该允许测试人员查看对应用的多维评估信息，能够获取当前 Web 应用在兼容性、稳定性、健壮性以及性能等维度上的建议和改进信息。 |
| RQ11 | 导出缺陷脚本 | 系统应该允许测试人员查看缺陷详情时，可下载进入该缺陷相关状态页面操作步骤对应的测试用例，即 Selenium 脚本，以方便后续的缺陷复现和测试案例收集。 |

3.3.3 非功能性需求

非功能需求，作为软件系统的质量属性和功能约束 [38]，既是系统的隐式需求，也是影响系统质量的重要因素，虽然跟用户没有直接的功能交互，但常常影响着系统的架构设计和技术选型。Web 应用自动化测试系统，作为一个面向专业用户的测试工具，对系统的可用性、可靠性和性能等方面均有较高的要求，经过仔细分析，其非功能需求主要如下表 3.3 所示。

一方面，系统应该具有高可用性、高可靠性和较好的性能，通过多种容灾备份技术和故障恢复机制等手段，来保证系统的功能正常使用和良好用户体验；另一方面，系统也要求具备良好的伸缩性和可扩展性，能够快速进行系统水平拓展，以及低成本功能变更和算法替换，以应对业务的迅速发展。同时，为了确保系统的数据安全和资源安全，系统的安全性也是我们需要考虑的要素。较好的易用性，则利于用户能在短期内熟悉本系统的相关操作。

表 3.3: 系统非功能需求列表

| 类别 | 需求描述 |
|------|---|
| 可用性 | 系统需要保障全年 99% 的可用性，当系统因自身缺陷或其他外部原因发生系统崩溃或服务不可用后，能通过负载均衡和主备切换等技术手段，在 5 分钟内快速恢复或切换到备用系统。 |
| 可靠性 | 系统应该具有较好的异常处理机制和数据备份体系，在系统出现异常和实例崩溃情况下，妥善保管和备份用户提交的测试任务数据和系统的分析数据。 |
| 性能 | 除测试任务自动化测试过程外，系统在正常网络场景下，应保证所有简单查询请求的时间控制在 200ms 以内，复杂查询和其他写请求性能要求可放宽至 500ms 以内。 |
| 可拓展性 | 系统应该采用低耦合的架构和模块设计，以较低的修改成本支撑系统后续功能的拓展，如缺陷种类和测试数据源的扩增，缺陷分类算法的替换等等。 |
| 安全性 | 系统应该确保系统的访问和数据安全，防止用户信息和被测试 Web 应用的相关数据泄露。 |
| 易用性 | 系统需要符合通用的人机交互规范，界面简洁且易于理解，布局合理，并提供给用户直观友好的信息提示和引导。 |
| 伸缩性 | 系统应该支持测试资源实例的快速扩展，以应对用户量和并发量的增长，实现高吞吐率和低延迟高性能。 |

3.3.4 系统用例图

根据上述系统功能性需求的分析结果，我们总结了如图3.2所示的系统用例图。本系统的主要用户为 Web 应用的测试人员和系统管理员，测试人员会负责 Web 应用自动化测试任务的大部分生命周期活动，包括任务发布、执行和结果查看，因此主要测试用例包括创建自动化测试任务、查看自动化测试任务列表、

启动自动化测试任务、查看测试报告、查看缺陷列表、查看缺陷详情以及查看多维评估数据。其中创建自动化测试任务用例还包括配置测试环境和配置应用权限功能，启动测试自动化测试任务则包括测试执行和测试分析功能，同时执行任务和重测任务功能扩展自启动自动化测试任务用例。

系统管理员则负责对测试人员新创建的自动化测试任务进行审核，只有审核通过的测试任务才能进入后续的测试流程。系统管理员拥有唯一的用例，即审核自动化测试任务。

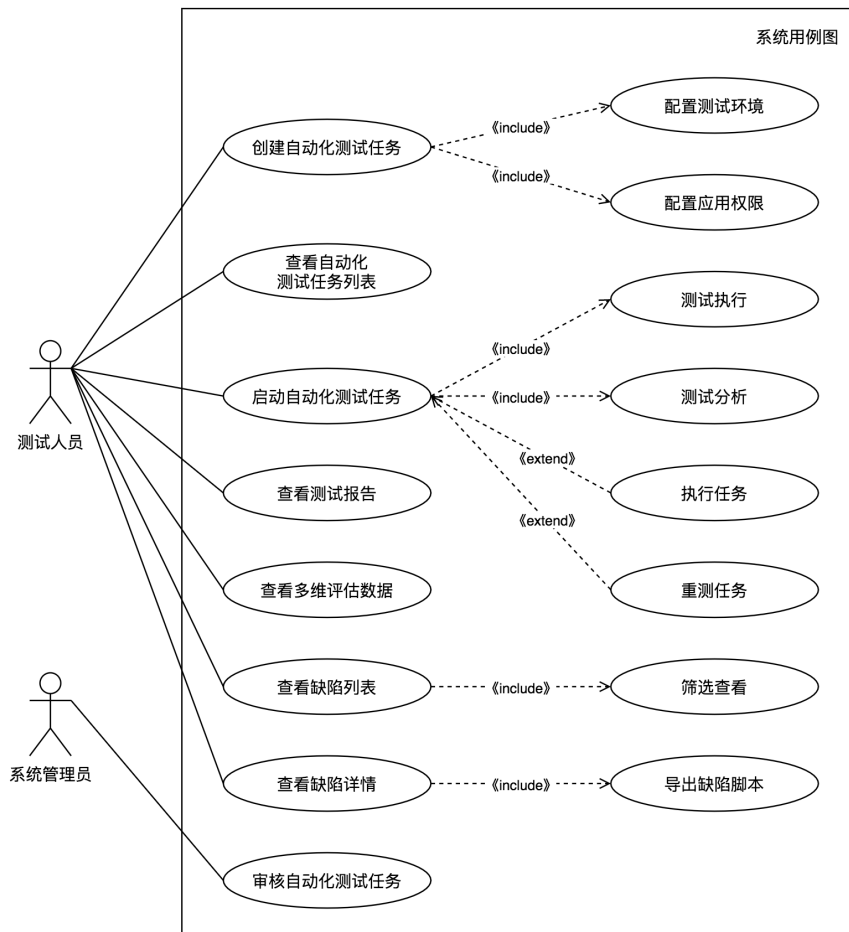


图 3.2: 系统用例图

3.3.5 系统用例描述

本小节是上述系统用例图中主要展示用例的详细阐述，将会介绍多个用例的正常流程、扩展流程、触发条件、前置条件和后置条件等信息，涉及用例跟功能需求的对应关系可如表 3.4所示。其中，囿于篇幅限制和突出重点需要，用例

表 3.4: 系统用例列表

| 用例编号 | 用例名称 | 功能需求编号 |
|------|-------------|-------------|
| UC1 | 创建自动化测试任务 | RQ1、RQ2、RQ3 |
| UC2 | 查看自动化测试任务列表 | RQ5 |
| UC3 | 启动自动化测试任务 | RQ6 |
| UC4 | 查看测试报告 | RQ7 |
| UC5 | 查看缺陷列表 | RQ8 |
| UC6 | 查看缺陷详情 | RQ9、R11 |
| UC7 | 查看多维评估数据 | RQ10 |
| UC8 | 审核自动化测试任务 | RQ4 |

UC2 和 UC3 相关的用例描述并未详细给出，分别对应自动化测试任务列表查看和筛选功能，以及具体任务的启动功能。

表 3.5: 创建自动化测试任务用例描述

| | |
|-------------|--|
| ID | UC1 |
| 用例描述 | 创建自动化测试任务 |
| 参与者 | 测试人员 |
| 触发条件 | 测试人员点击菜单项进入发布自动化测试任务页面 |
| 前置条件 | 测试人员必须已被识别和授权 |
| 后置条件 | 1. 系统管理员能在审核自动化测试任务页面看到该任务； 2. 测试人员能在查看自动化测试任务列表页面检索到该任务，并处于待审核状态。 |
| 优先级 | 高 |
| 正常流程 | <ol style="list-style-type: none"> 1. 测试人员进入任务发布页面，输入扫描目标网址，并点击上传； 2. 系统显示操作成功，并进入补充配置页面； 3. 测试人员输入通用配置信息，包括必填的任务目标，可选的任务限时、最大页面状态数和最大页面搜索深度参数； 4. 系统回显输入； 5. 测试人员在添加浏览器配置表单选择平台类型和浏览器类型，输入浏览器数量后点击保存配置； 6. 系统在当前页面显示添加的浏览器系列配置； 7. 测试人员在添加权限配置表单中输入事件类型、搜索方式、目标元素等信息，并点击保存配置； 8. 系统在当前页面显示添加的权限配置信息； 9. 测试人员点击提交测试按钮； 10. 系统提示提交成功，并进入提交完成页面。 |
| 拓展流程 | <ol style="list-style-type: none"> 1a. 测试人员输入的扫描目标校验未通过； <ol style="list-style-type: none"> 1. 系统提示不是一个有效的网址 1b. 测试人员选取某个历史测试目标； <ol style="list-style-type: none"> 1. 返回正常流程第 2 步。 |
| 特殊需求 | 系统需支持多组浏览器配置 |

UC1 创建自动化测试任务的用例描述如表3.5所示，测试人员登录系统后，进入任务发布页面即可进行新任务的信息填写，包括扫描目标、任务名称、浏览器系统配置和权限配置等信息，输入完成点击提交测试按钮后，测试任务即可创建成功，并处于“待审核”状态。其中，系统支持用户手动输入和选取历史测试目标两种扫描目标输入模式，并在输入的信息不符合校验规则时给出相应的错误提示。

表 3.6: 查看测试报告用例描述

| | |
|-------------|--|
| ID | UC4 |
| 用例描述 | 查看测试报告 |
| 参与者 | 测试人员 |
| 触发条件 | 测试人员点击查看测试报告操作 |
| 前置条件 | 1. 测试人员必须已被识别和授权； 2. 测试任务已测试完成。 |
| 后置条件 | 无 |
| 优先级 | 高 |
| 正常流程 | 1. 测试人员点击查看报告操作； 2. 系统跳转至当前应用的测试报告详情页，显示包括应用 URL、耗时、状态数和缺陷数在内的应用基本信息，提供按平台、缺陷类别、缺陷级别的缺陷分布饼图，以及具体的缺陷列表和多维评估信息跳转入口。 |
| 拓展流程 | 1a. 测试人员点击下载报告操作； 1. 系统将测试报告包下载到测试人员机器上，支持离线访问。 2a. 测试人员点击分布饼图上的某一统计维度类别； 1. 系统跳转至绑定维度筛选条件的缺陷列表页面。 |
| 特殊需求 | 无 |

UC4 查看测试报告的用例描述如表3.6所示，当测试任务在后台执行分析结束后，系统会更新当前任务为“执行完成”状态，并显示查看报告和下载报告操作。测试人员可以点击查看报告操作，在线查看对应的 Web 应用测试报告，或者点击下载报告操作，下载离线报告本地访问。其中，测试报告页面会包括应用 URL、耗时和状态数等应用基本信息、以饼图形式呈现的缺陷相关统计信息，以及具体的缺陷列表以及多维评估信息跳转入口。

UC5 查看缺陷列表的用例描述如表3.7所示，测试人员从测试报告页面点击缺陷列表选项卡或缺陷分布饼图中的某个统计维度类别图块，即可查看具体的缺陷列表，并支持从操作系统、缺陷类别和缺陷级别三个维度进行列表筛选。对于缺陷列表中的每一条缺陷，系统会显示缺陷所在应用页面状态、缺陷类型、缺陷级别、包含操作系统和浏览器在内的测试环境信息，以及缺陷详情的跳转入口等信息。

表 3.7: 查看缺陷列表用例描述

| | |
|-------------|---|
| ID | UC5 |
| 用例描述 | 查看缺陷列表 |
| 参与者 | 测试人员 |
| 触发条件 | 测试人员点击缺陷列表选项卡或饼图中的统计维度类别图块 |
| 前置条件 | 1. 测试人员必须已被识别和授权； 2. 测试任务已测试完成。 |
| 后置条件 | 无 |
| 优先级 | 中 |
| 正常流程 | 1. 测试人员进入查看缺陷列表页面； 2. 系统显示当前任务对应的缺陷列表信息，包括状态、缺陷类型、缺陷级别、操作系统、浏览器，以及详情跳转入口等信息； 3. 测试人员下拉选择某类操作系统、缺陷类别或缺陷级别； 4. 系统显示筛选后的缺陷列表。 |
| 拓展流程 | 无 |
| 特殊需求 | 相似缺陷的去重 |

UC6 查看缺陷详情的用例描述如表3.8所示，测试人员可以在缺陷列表页面选择某个缺陷点击查看详情，进入缺陷详情页面。系统会显示该缺陷的基本信息，包括缺陷日志、缺陷类别和严重等级等数据，以及相应的测试环境和缺陷上下文信息，包括前序状态、前序路由和操作步骤等数据。同时，系统会根据复现该缺陷的操作序列生成对应的 Selenium 测试脚本，用户可以点击脚本下载链接自行导出。

表 3.8: 查看缺陷详情用例描述

| | |
|-------------|--|
| ID | UC6 |
| 用例描述 | 查看缺陷详情 |
| 参与者 | 测试人员 |
| 触发条件 | 测试人员选择查看某个缺陷详情 |
| 前置条件 | 1. 测试人员必须已被识别和授权； 2. 测试任务已测试完成。 |
| 后置条件 | 无 |
| 优先级 | 高 |
| 正常流程 | 1. 测试软件点击缺陷列表中某个缺陷查看详情； 2. 系统显示该缺陷的详细内容，包括缺陷日志、类别、严重等级，以及包括截图、操作序列、当前状态和前序状态等在内的缺陷上下文信息； 3. 测试人员点击下载测试脚本链接； 4. 系统将对对应操作序列的 Selenium 脚本下载至测试人员机器上。 |
| 扩展流程 | 无 |
| 特殊需求 | 脚本能辅助可复现缺陷 |

UC7 查看多维评估数据的用例描述如表3.9所示，测试人员从测试报告页面可以点击多维评估选项卡，查看对应用的软件质量多维评估分析结果，主要包括从功能、性能、健壮性、兼容性和稳定性五个维度进行分析的量化雷达图，以及各自维度下的详细评估信息。

具体地，在功能维度下，系统会提供给测试人员在 Chrome 浏览器下的功能导航图，图中包括了遍历到的页面状态和操作事件流；在性能维度下，则会提示所有可能有性能问题的请求，并以列表呈现，每一条信息具体会包括请求链接、请求类型、请求方法，请求资源大小等等。在健壮性维度下，则会显示先前缺陷报告中具体的缺陷列表，详情包括缺陷类型、状态以及页面截图等等；在兼容性维度下，系统会显示各个不同测试环境下的成功操作路径图，以对比应用在不同环境下的可执行性；至于稳定性维度，系统则会显示在单一环境下应用无法继续执行操作的页面列表。

表 3.9: 查看多维评估数据

| | |
|-------------|---|
| ID | UC7 |
| 用例描述 | 查看多维评估数据 |
| 参与者 | 测试人员 |
| 触发条件 | 测试人员点击测试报告多维评估选项卡 |
| 前置条件 | 1. 测试人员必须已被识别和授权；2. 测试任务已测试完成。 |
| 后置条件 | 无 |
| 优先级 | 中 |
| 正常流程 | 1. 测试人员点击选项卡进入到多维评估页面； 2. 系统显示包含功能、性能、健壮性、兼容性和稳定性的多维评估雷达图； 3. 测试人员点击切换至功能评估详情，页面显示功能导航图，包括遍历到的页面状态和操作事件流； 4. 测试人员点击切换至性能评估详情，页面显示请求性能提示列表，包括请求链接、请求时间和请求类型等信息； 5. 测试人员点击切换至健壮性评估详情，页面显示缺陷列表； 6. 测试人员点击切换至兼容性评估详情，页面显示各环境测试路径图； 7. 测试人员点击切换至稳定性评估详情，页面显示稳定性问题页面列表。 |
| 拓展流程 | 5a, 7a. 测试人员点击查看某缺陷或稳定性问题页面截图； 1. 系统显示对应的页面截图。 |
| 特殊需求 | 无 |

UC8 审核自动化测试任务的用例描述如表3.10所示，系统管理员登录系统，进入任务审核页面后，会看到所有用户的测试任务列表。系统管理员可以选择某个具体任务进入其详情页面，并根据显示的待测应用和任务配置等信息进行测试任务的审核判断，执行审核通过或拒绝操作。

表 3.10: 审核自动化测试任务

| | |
|-------------|--|
| ID | UC8 |
| 用例描述 | 审核自动化测试任务 |
| 参与者 | 系统管理员 |
| 触发条件 | 有新建测试自动化测试任务提交到系统中 |
| 前置条件 | 系统管理员必须已被识别和授权 |
| 后置条件 | 任务对应测试人员能在列表查看页面检索到最新可执行状态的该任务 |
| 优先级 | 高 |
| 正常流程 | <ol style="list-style-type: none"> 1. 系统管理员点击进入任务审核页面； 2. 系统按时间倒序分页显示全部用户的自动化测试任务列表； 3. 系统管理员选择某个测试任务点击查看详情选项； 4. 系统跳转至任务详情页面； 5. 系统管理员点击审核通过按钮； 6. 测试任务更新为审核通过状态，并更新详情界面。 |
| 拓展流程 | <ol style="list-style-type: none"> 3a. 系统管理员点击拒绝按钮； <ol style="list-style-type: none"> 1. 测试任务更新为审核未通过状态，并更新详情界面。 |
| 特殊需求 | 无 |

3.4 Web 应用自动化测试系统整体架构设计

结合需求分析过程中明确的功能需求和非功能需求，经过服务拆分和技术选型，最终，我们得到了如图 3.3 所示的 Web 应用自动化测试系统整体架构设计图。系统整体框架自顶向下可拆解为展示交互层、任务调度层、测试执行层、测试分析与测试报告层，以及依赖的消息中间件和数据存储层。基于异步化调用和拉模型，层与层之间实现了高度解耦。

其中，展示交互层是用户和系统交互和操作的直接窗口，来自不同领域的测试人员用户可以在系统入口提交待测 Web 应用的基本信息和配置信息，如网站的测试入口 URL、测试环境和权限相关的配置，从而创建一个 Web 应用自动化测试任务。最终测试任务执行完成后，系统会基于 Node.js 的定制化模板，渲染返回的测试结果数据，将详细的测试报告信息展现给用户，并支持在线和离线多种访问形式。

任务调度层，主要负责对接展示交互层，提供相应请求业务逻辑的处理，负责任务的创建、查看和审核等操作，维护了整个测试任务的生命周期。当接收到展示交互层提交的测试任务后，它会根据输入配置设置，进行配置管理和任务的初始化，并通过消息中间件 Celery 分发到测试执行层进行后续操作，从而实现长任务的调度和状态监控，以避免 HTTP 请求超时重试带来的架构风险。

测试执行层，作为整个系统的关键部分之一，主要负责收到测试任务后，基于 Selenium Grid 实现的多测试环境和优化的 Crawljax 框架，实现 Web 应用的自

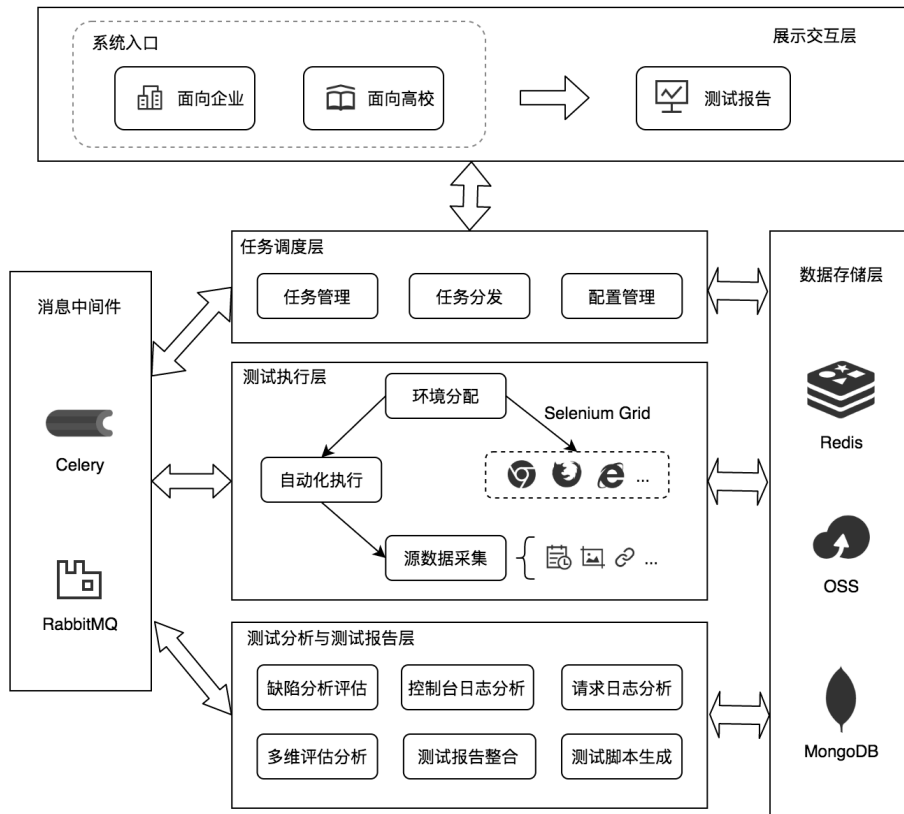


图 3.3: 系统整体架构设计图

自动化遍历执行。同时通过实现多个 Crawljax 插件，完成后续分析服务所需源数据的采集，如控制台日志、截图和相应请求日志等数据，并将这些数据存储到基于 Redis、OSS 和 MongoDB 等构建的异构数据存储层，通过 RabbitMQ 消息中间件触发下游的测试分析与测试报告层。

测试分析与测试报告层，又可以拆解为分析服务和报告生成服务。分析服务作为本文主要阐述的内容，主要是基于测试执行层采集的“黑匣子”数据，在接受到 RabbitMQ 传递的分析处理信号后，对这些多源异构数据进行分析 and 处理。整体职责上可以分为两个方向，一方面，它从无到有构建了一个 Web 应用缺陷体系，通过分析控制台日志和请求信息挖掘待测应用可能存在的缺陷，并进行分类和去重，最后交由报告生成服务产出对应的测试报告，另一方面，它从兼容性、健壮性和性能等多个维度，对应用进行多维评估，以更好地对 Web 应用进行分析。最后分析结果会存入数据存储层，供上游服务返回给用户。报告生成服务，主要是对分析服务产出的缺陷和评估数据进行整合分析，并进行可视化呈现，同时提供测试脚本的转换与导出。

3.5 Web 应用自动化测试系统分析服务概要设计

分析服务，是整个 Web 应用自动化测试系统的重要组成部分，重点关注于对待测 Web 应用的缺陷分析和多维评估方向，关系到最终系统生成测试报告的质量，因此概要设计的好坏对系统来说十分重要。本小节将重点叙述分析服务的概要设计过程，从内部架构、模块划分、4+1 视图以及持久化模型设计四方面进行展开分析。

3.5.1 架构设计与模块划分

分析服务内部的架构层次可如图 3.4 所示，我们将整个分析服务划分为缺陷分析框架、控制台日志缺陷检测模块、请求日志缺陷检测模块和 Web 应用多维评估模块四个部分，其中控制台日志缺陷检测模块，又可分为控制台日志缺陷聚类 and 判定两个子模块，对应整体系统架构中测试分析与测试报告层的主要工作。整个分析服务可以看做是基于缺陷分析框架的缺陷检测器集合和多维评估器，依赖于 RabbitMQ 消息中间件和由 Redis、OSS 以及 MongoDB 组成的异构数据存储层，对上游服务中获取的数据进行缺陷检测分类和软件质量多维评估，并支持其他缺陷检测模块的集成。结合获取到的“黑匣子”数据，我们当前可实现的缺陷检测器包括控制台日志缺陷检测模块和请求日志缺陷检测模块，后续如有后台日志、中间件日志等数据的引入，还可以实现对这些数据的缺陷检测和快速集成。

缺陷分析框架是整个分析服务的核心部件，负责其他组件的组织与集成，是系统扩展性的有效保障，可进一步细分场景接入、缺陷检测接入、缺陷整合去重和上下文管理四个子任务。在测试执行的不同场景阶段，RabbitMQ 作为消息中间件会发送对应的分析处理信号，由缺陷分析框架负责接入，并根据场景和数据源类型下发到不同的缺陷检测模块，最终汇集全部场景内缺陷并进行整合去重，同时存入数据存储层的 Redis 中，以便报告生成服务获取所有场景数据。

控制台日志缺陷检测模块是针对控制台日志进行缺陷检测的模块，由缺陷分析框架负责接入，对警告和严重级别的控制台日志进行清洗后，基于先期的缺陷聚类分析过程得到的缺陷信息知识库进行缺陷评估判定，并将结果返回到分析框架中进行后续整合操作。同时，新的严重级别日志，也将转化引入到缺陷聚类流程需要的源数据中，便于缺陷知识库的丰富。其主要步骤包括日志搜集和清洗、缺陷搜集、缺陷聚类和缺陷判定器构建等过程，可以拆解为控制台日志缺陷聚类和缺陷判定两个子模块。

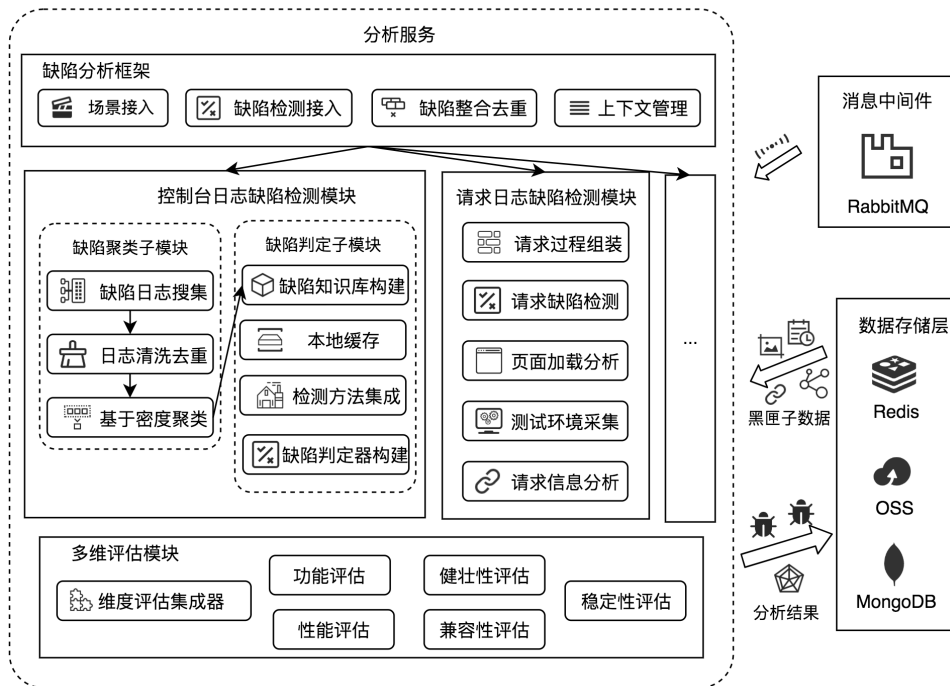


图 3.4: 分析服务架构设计图

请求日志缺陷检测模块是另一个接入的缺陷检测模块，负责请求过程组装、请求缺陷检测、页面加载分析、测试环境采集，以及请求信息分析等子任务，主要针对记录的请求日志，将原本以结构化记录的日志，参照 Chrome Devtools Protocol 规范重新建模成完整的请求过程，并针对异常情况，如请求响应状态码为 404 或 500 等情况，进行缺陷分析。同时，基于请求过程，完成相关性能数据的提取和存储，如请求响应时间等信息，便于后续多维评估模块的分析使用。

多维评估模块，是整个测试执行和测试分析过程的切面分析和维度总结，基于前期分析和产出的缺陷数据和性能数据，以及测试执行过程的操作事件流和状态信息等，从健壮性、性能、稳定性和兼容性以及功能五个维度对应用进行整体评估，为应用评价提供分析数据，包括参考性量化评分和各维度改进提示信息。同时，设计了维度评估集成器，支持新维度的分析引入。

整个服务基于 SpringBoot 框架进行构建，并依靠其 IOC 容器强大的特性，实现相关组件的引入和满足相关拓展性的需要。RabbitMQ 则为多场景接入实现提供异步调用解决方案，其消息确认机制，也进一步增强了服务的容灾能力。OSS 主要负责页面截图等大文件数据的存储维护，而 Redis 和 MongoDB 则分别提供了面向不同存储场景的短时和持久化数据存储需要。

3.5.2 4+1 视图

”4+1”视图模型是 Philippe Kruchten 教授提出的一种使用多个并发视图的软件架构描述模型 [39]，以从不同涉众的角度更好地刻画系统架构，主要视图包括开发视图、逻辑视图、进程视图、物理视图，以及对应数字“1”的场景视图。每个视图是整个系统的一个切面表达，结合起来才能完整的反映系统设计。其中场景视图，主要是用来识别业务需求和刻画业务场景，对应 UML 体系中的用例图，如上文图 3.2所示。

逻辑视图面向最终用户视角，描述了系统的功能需求，常用类图作为其表述的结构化载体，如图 3.5所示。我们将系统分解成一系列关键抽象，其中 SceneAnalysisService 负责场景分析接入和缺陷检测调度，负责支持整个分析服务的后续场景扩展。ConsoleChecker 和 PerformanceChecker 是按需使用的缺陷检测器，分别负责对控制台日志和请求日志的缺陷检测。SceneContext 负责整个场景中的上下文管理，BugList 提供了对缺陷列表的查询功能，BugDetail 负责维护缺陷详情和组装缺陷上下文，PageInfo 则维护了应用的页面信息。此外，ClusterService 提供了对控制台日志的聚类分析服务，构建了负责维护缺陷类别信息的缺陷知识库 BugKnowledge，AcceptanceScoreService 则负责提供对应用质量的多维评估功能，底下的 Calculator 是具体维度的评估实现。在整个过程中，RedisTemplate 和基于泛型 DAO 设计的 MongoRepository[40] 分别提供了完整的 Redis 和 MongoDB 数据库操作支持。

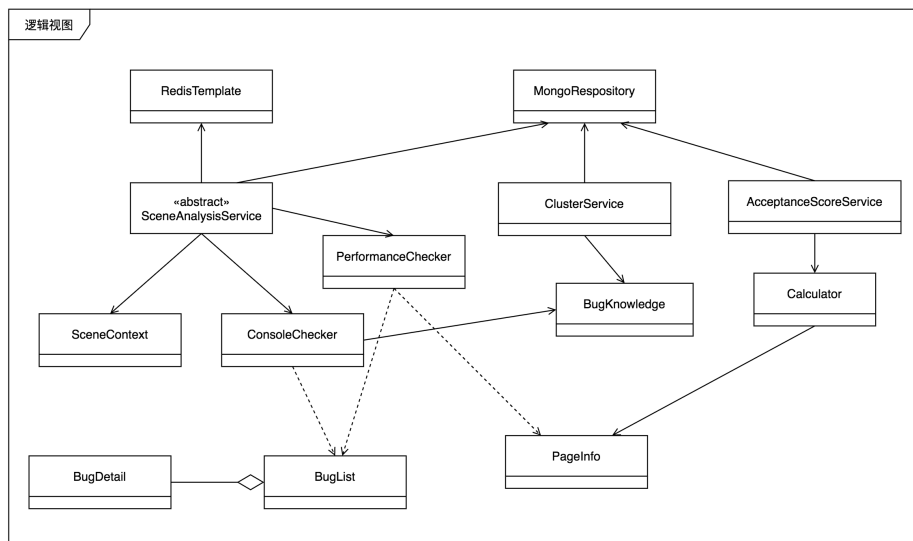


图 3.5: 逻辑视图

进程视图则关注于系统的并发和信息同步方向，面向系统设计和集成人员视角，主要描述系统进程和线程等主要部分之间的通信过程。分析服务的进程视图可如图 3.6 所示，执行服务进程会通过 RabbitMQ 消息队列发送不同的操作消息，进而实现分析服务的调用。分析服务主进程则会响应相应的 RabbitMQ 消息，从 Redis 和 MongoDB 数据中心分别获取相应的测试执行过程采集的中间数据，来调用自身的缺陷检测和多维评估对应的具体方法，以对待测应用进行完整的分析，并将结果数据存入 MongoDB 中，以供后续查询需要。其中，当新的未知缺陷日志引入后，分析服务主进程还可以启动异步线程来进行缺陷聚类分析，以进一步丰富缺陷知识库。

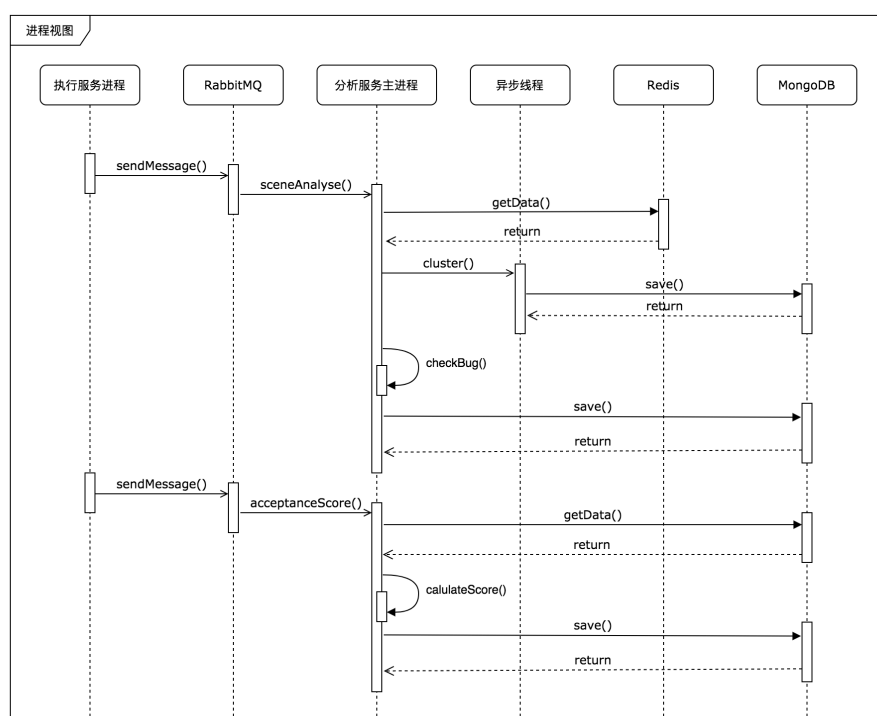


图 3.6: 进程视图

开发视图面向软件系统的开发人员、产品经理等视角，主要关注软件开发的静态组织结构，如程序包、引用类库等。分析服务涉及的开发视图可如 3.7 所示，在 UI 部分，一共包括 `template`、`css`、`js` 以及 `image` 四个子包，共同实现前端测试报告页面的展示。

在服务端部分，则采用分层的架构，自上而下包括了 `Controller` 包、`Service` 包、`Component` 包和 `Repository` 包，以及相应的 `Model` 包和 `Utils` 工具包。其中，`Controller` 包主要负责对接外部请求和来自 RabbitMQ 的场景消息，涵括了 `AnalysisListener`、`ReportListener` 以及 `ClusterController` 等具体类。`Service` 包是复杂业

务逻辑的构建器和直接服务提供商, 包括了 SceneAnalyseService、ClusterService、AcceptanceScoreService 等多个一级服务类。Component 包则是模块下各个细分高内聚业务的集合, 服务于 Service 包, 底下又细分为 checker 包、evaluate 包、cluster 包以及 calculator 包等多个子包。Repository 包提供了对接底层数据库的操作支持, 包括数据的存储和查询等。Model 包是对象实体的集合, 层下根据业务语义的需要划分为多个子包结构, 如 Context 上下文包, MongoDB 实体文档包、Request 请求数据包等等。Utils 包是工具类的集合, 提供了常用的 JAVA 工具类, 比如 JSONUtils, SpringUtils 等类。

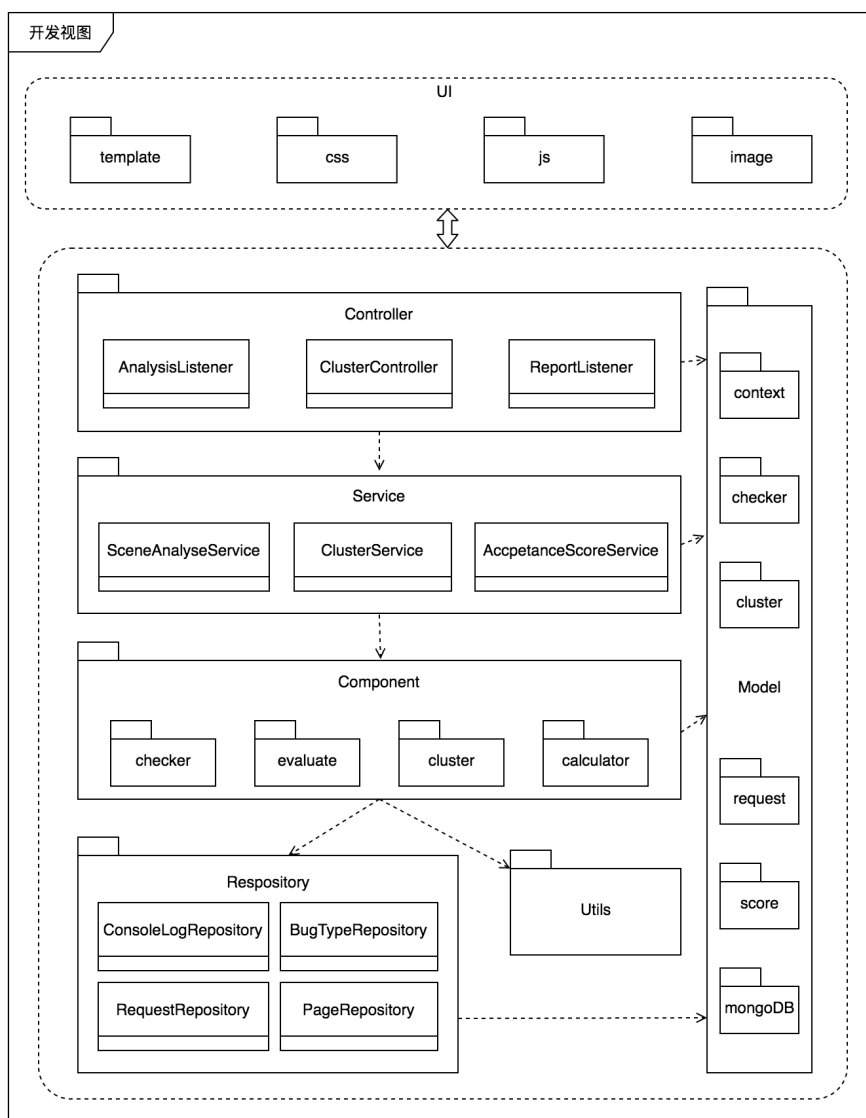


图 3.7: 开发视图

图 3.8是整个系统的部署视图，面向运维人员角度，重点关注软件的部署结构和网络通信策略，来配合软件应用的可靠性、可伸缩性等要求。用户通过浏览器发送 HTTP 请求来访问系统，由自动化测试平台服务器负责接收，并通过 HTTP 请求调度执行服务开始工作。执行服务和分析服务都通过 TCP 连接 RabbitMQ 消息队列服务器，绑定生产者-消费者关系，建立异步流水线调用机制。同时分析服务会通过 HTTP 连接，从 Redis、MongoDB 以及 OSS 在内的数据文件服务器中获取相应数据。这些数据服务器既可以单独部署，也可以随着应用规模和资源需求的增加，实现集群部署。

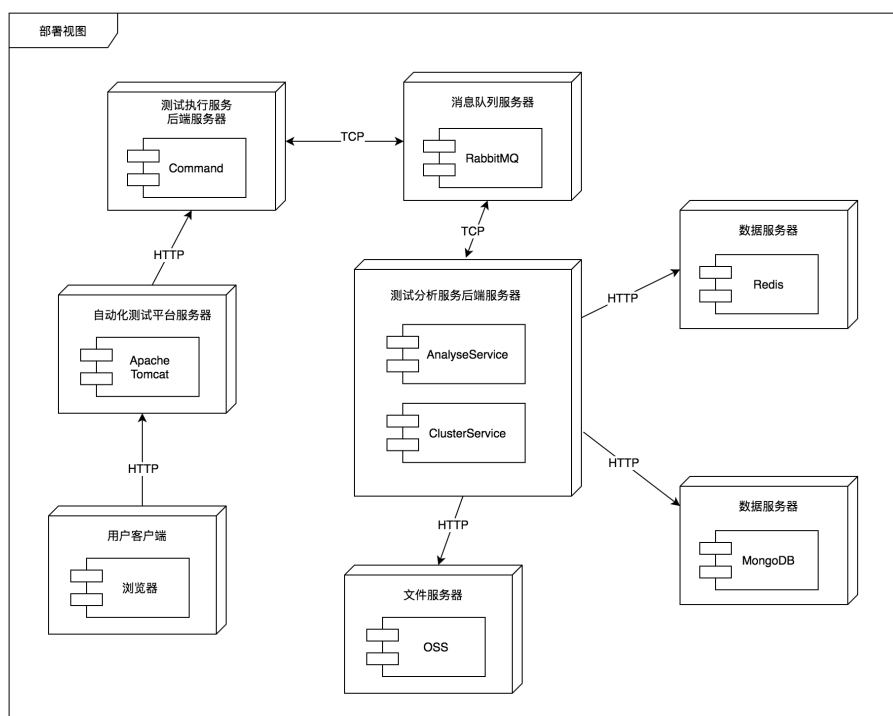


图 3.8: 部署视图

3.5.3 持久化模型设计

根据系统需求，分析服务需要对部分数据对象进行持久化处理，主要包括控制台严重日志记录、聚类分析结果、缺陷知识库，页面记录以及请求记录等。我们采用 MongoDB 作为数据存储的载体，利用其文档结构的特性，实现这些数据的存储和查询需要。

ConsoleLog 类保存了严重等级的控制台日志，包括日志的完整描述和输出页面，是测试执行过程中进行采集的原始数据，也是后续控制台缺陷日志聚类分析的数据源，表 3.11 列举了 ConsoleLog 类的详细说明。

表 3.11: ConsoleLog 类详情列表

| 字段 | 类型 | 含义 | 备注 |
|-----|--------|---------|-----------------|
| id | String | 日志记录 ID | 全局唯一的 ID 属性 |
| url | String | 输出日志页面 | 控制台日志输出时的页面链接 |
| log | String | 日志详情 | 完整的严重等级控制台日志字符串 |

ConsoleLogCluster 类保存了控制台日志缺陷聚类的结果数据，包括类别信息和该类别下的日志信息等，后续可用来进行缺陷的人工审查总结，来丰富缺陷知识库。表 3.12列举了 ConsoleLogCluster 类的详细说明。

表 3.12: ConsoleLogCluster 类详情列表

| 字段 | 类型 | 含义 | 备注 |
|-----------|--------------|---------|---------------------------|
| id | String | 类别 ID | 全局唯一的 ID 属性 |
| processId | String | 聚类任务 ID | 当前任务的唯一标识 |
| classId | Integer | 聚类类别 | 从 1 递增的类别号，特别地，-1 代表噪声点类别 |
| logList | List<String> | 类别下日志列表 | 归属于该类别下的缺陷日志集合 |

BugType 类保存了控制台日志缺陷聚类的最终沉淀结果，是控制台日志缺陷的知识库最小组成单元，包括缺陷种类、特征值、以及示例等信息，服务于后续的控制台日志缺陷判定流程。表 3.13列举了 BugType 类的详细说明。

表 3.13: BugType 类详情列表

| 字段 | 类型 | 含义 | 备注 |
|--------------|---------|-----------|--------------------------------|
| id | String | 缺陷 ID | 全局唯一的 ID 属性 |
| code | Integer | 缺陷所属大类 ID | 最上层的缺陷归类 |
| example | String | 缺陷示例 | 一个实际的该缺陷案例 |
| detectMethod | Integer | 检测方法 | 检测方法枚举值，如 1-特征检测，2-正则匹配，3-相似对比 |
| detectRule | String | 检测规则 | 跟缺陷绑定的检测方法使用的参数 |

PageInfo 类则保存了请求日志缺陷检测过程中沉淀的页面信息，用于后续的多维评估模块的稳定性分析，主要包括页面采集位置信息和页面信息，表 3.14是该类的详细说明。

RequestInfo 类保存了请求日志缺陷检测过程中沉淀的成功请求信息，后续服务于多维评估部分的性能评估，主要包括请求信息的部分参数和请求采集位置信息，表 3.15是该类的详细说明。

表 3.14: PageInfo 类详情列表

| 字段 | 类型 | 含义 | 备注 |
|----------------|---------|---------|--------------|
| taskId | String | 测试任务 ID | 全局唯一的测试任务标识 |
| partitionId | String | 子任务 Id | 绑定测试环境的子任务标识 |
| state | String | 测试状态 | 请求采集时的状态值 |
| url | String | 请求链接 | 采集的请求 URL |
| screenshotPath | String | 页面截图 | OSS 路径字符串 |
| errorPage | boolean | 是否为异常页面 | 根据响应状态码判断 |

表 3.15: RequestInfo 类详情列表

| 字段 | 类型 | 含义 | 备注 |
|--------------------------|------------------|---------|-----------------|
| taskId | String | 测试任务 ID | 全局唯一的测试任务标识 |
| partitionId | String | 子任务 Id | 绑定测试环境的子任务标识 |
| state | String | 测试状态 | 请求采集时的状态值 |
| url | String | 请求链接 | 采集的请求 URL |
| detail | RequestInfoExtra | 请求详情 | 嵌套文档对象 |
| detail.method | String | 请求方法 | 如 GET、POST 等 |
| detail.params | String | 请求参数 | 请求提交的参数，JSON 格式 |
| detail.type | String | 资源类型 | 如 Image、XHR 等 |
| detail.requestTime | Double | 请求时间 | 时间戳差值 |
| detail.encodedDataLength | Long | 响应字节数 | 单位为字节 |
| detail.description | String | 请求描述 | 请求的补充说明 |

3.6 本章小结

本章是系统需求分析和概要设计过程的文字表征，首先分别介绍了 Web 应用自动化测试系统的整体情况和分析服务的相关说明，然后对系统进行整体的需求分析工作，完成了系统的涉众识别，功能需求和非功能需求的分析归纳，并通过系统用例图和系统用例描述进行阐述。最后，阐述了系统的整体架构设计和分析服务部分的概要设计内容，包括服务内部的架构设计与模块划分，相应的 4+1 视图，以及涉及到的持久化模型设计，确定了分析服务的初步设计方案，为下一章的详细设计与代码实现奠定基础。

第四章 系统详细设计与具体实现

根据第三章的需求分析和概要设计结果，分析服务部分主要包括缺陷分析框架、控制台日志缺陷检测模块，请求日志缺陷检测模块以及 Web 应用多维评估模块。其中控制台日志缺陷检测模块，又可以拆解成控制台日志缺陷聚类 and 缺陷判定两个子模块。本章将对这些主要模块的详细设计和内部实现过程进行详细描述，并给出核心类图和部分关键代码。

4.1 Web 应用缺陷分析框架

Web 应用缺陷分析框架是整个分析服务的构建基础和设计规范。一方面，它以任务场景为粒度，将测试执行过程和缺陷分析过程相连接，这种流水线设计避免了服务的阶段式资源空闲和大量数据堆积后集中消费造成的性能抖动；另一方面，利用模板方法的设计模式 [41]，固定了整个缺陷分析的流程框架，规范了缺陷分析过程，同时支持后续其他数据源和缺陷检测算法的引入，进而提高整个应用的可拓展性。

4.1.1 框架设计

缺陷分析框架的整个设计如图 4.1 所示，结合执行服务的状态遍历特点，我们将整个测试执行分析过程切分成不同的场景，比如测试执行开始前（Before）、测试执行遍历到某个状态时（State）以及测试执行结束后（Finish）等，不同的任务场景下可获取的测试过程数据类型各异。

我们通过 Crawljax 的插件机制，监控整个遍历执行过程，让 RabbitMQ 在自动化测试任务进行到某个场景时发出相应的标志消息，异步化通知对应的 Listener 开始处理。接收到事件信号的 Listener，则会根据场景消息类型，获取并组装该场景中可采集到的测试过程数据，接入到不同的场景处理流程下。对应场景会利用这些差异化过程数据，通过集成的缺陷检测器集合，进行对待测 Web 应用的缺陷检测与分类。最终经过整合和去重的缺陷列表数据，会存入 Redis 数据中，以供后续的报告生成服务进行场景缺陷汇总和可视化。场景上下文则负责场景内差异化数据和相关分析状态的管理与维护，对后续多维评估模块可能涉及到的中间数据进行保存。

其中一个示例是，当执行服务遍历到 Web 应用的某个中间页面状态时，会通过 RabbitMQ 发出带有状态场景标志的消息，来通知相应的 Listener 组装该状

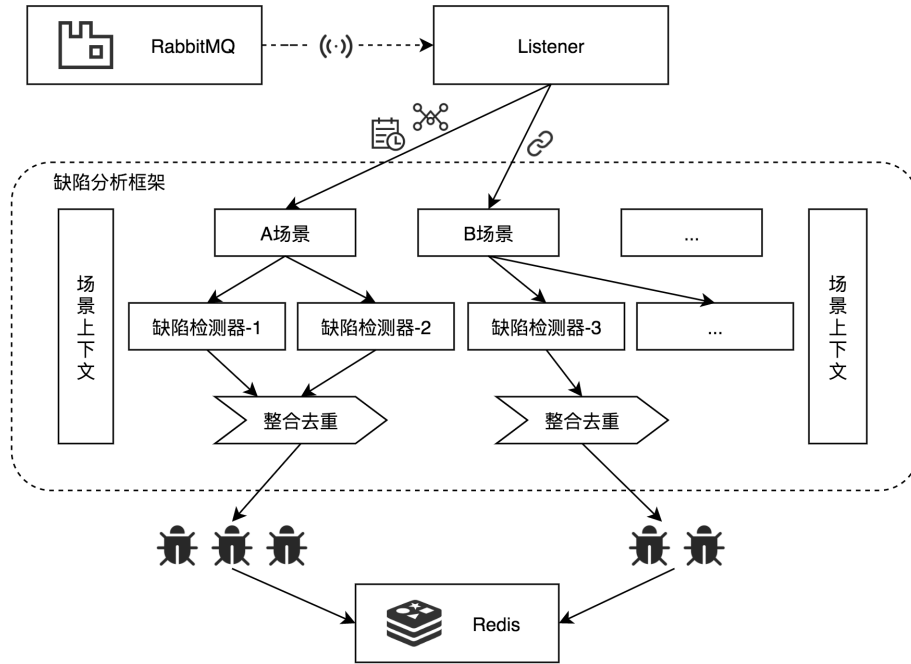


图 4.1: 缺陷分析框架设计图

态场景可获取的控制台日志、请求日志、截图等过程数据，并交付给其场景下的控制台日志缺陷检测和请求日志缺陷检测等模块，进行缺陷列表的构建和整合去重工作，以完成后续操作。

4.1.2 流程设计

图 4.2 展示了缺陷分析框架的示例系统顺序图。如图所示，当 Listener 接收到执行服务中 Crawljax 数据采集插件发送的 RabbitMQ 消息后，会根据场景信号调用 RedisTemplate 的 get 方法，获取和组织该场景下的产出数据，将其作为分析参数传递给 SceneAnalysisService 类进行场景下的缺陷分析。

作为缺陷分析框架的核心类，SceneAnalysisService 抽象类提供了 getSceneAnalysisList 公共方法，并在这个方法中编排了整个缺陷分析的流程，包括执行 initSceneContext 上下文初始化方法、不断调用继承自 AbstractChecker 抽象类的多个缺陷检测器的 checkBug 方法进行缺陷检测、执行自身的 doDistinctionOperation 和 doFinalOperation 方法，以及调用 MongoRepository 的 save 方法来进行相应数据的保存等等。其中，部分方法会根据不同场景需要，交由 SceneAnalysisService 子类实现或重写。最终，SceneAnalysisService 会补全缺陷上下文信息同时收集好缺陷列表，将数据返回给相应的 Listener，并由 Listener 调用 RedisTemplate 完成场景分析结果数据的储存。

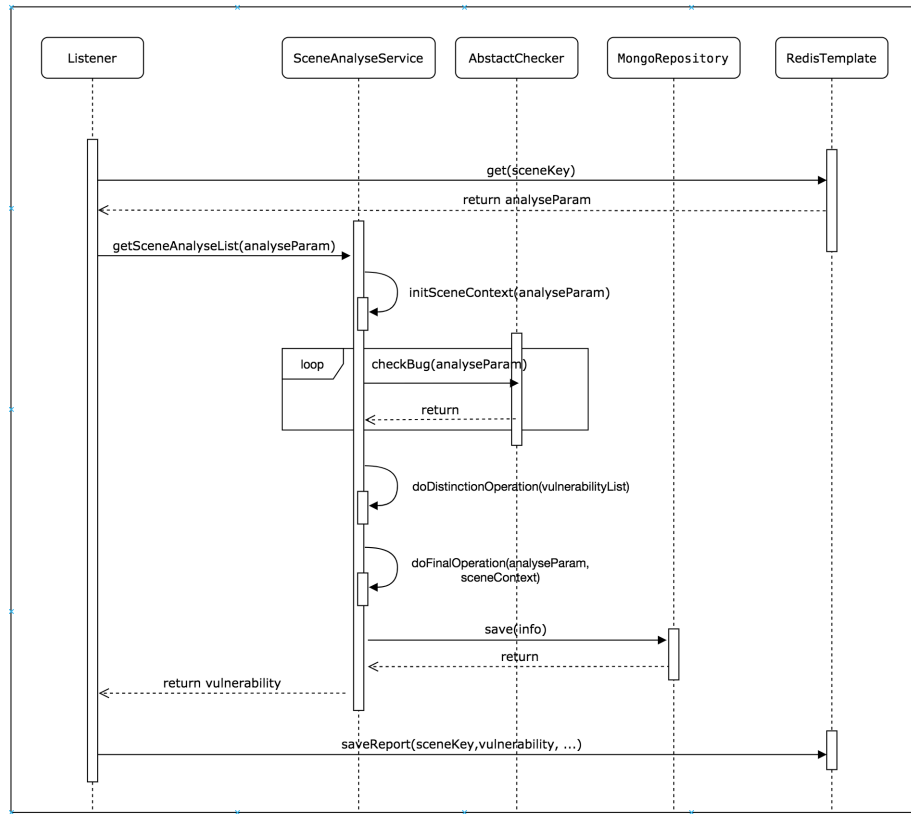


图 4.2: 缺陷分析框架系统顺序图

4.1.3 数据与类设计

缺陷分析框架的设计类图如 4.3 所示，我们使用了基于模板方法的设计模式，`SceneAnalysisService` 作为模板类，由 `getSceneAnalysisList` 方法定义了整个场景接入的流程骨架，并基于抽象类提供了多个抽象方法和公共实现方法，包括场景上下文的初始化、场景下的缺陷分析、对场景缺陷的整合去重以及场景分析完成时对上下文最后的处理流程等等。`StateSceneAnalysisService` 和 `FinishSceneAnalysisService` 是继承了 `SceneAnalysisService` 的具体实现类，通过实现父类的抽象方法，完成流程骨架中的自定义部分。

同时，通过泛型支持，参数化场景分析参数和场景上下文类型，结合上述的模板方法设计模式，实现了大规模的代码复用，以满足多场景接入的扩展要求 [42]。其中，我们将场景上下文具有的一些通用属性放在 `SceneContext` 类中，通过继承它来实现具体场景下的上下文对象，如 `FinalSceneContext`, `StateSceneContext` 等等。`AbstractChecker` 是一个抽象类，定义了缺陷分析器的基本类，通过继承其

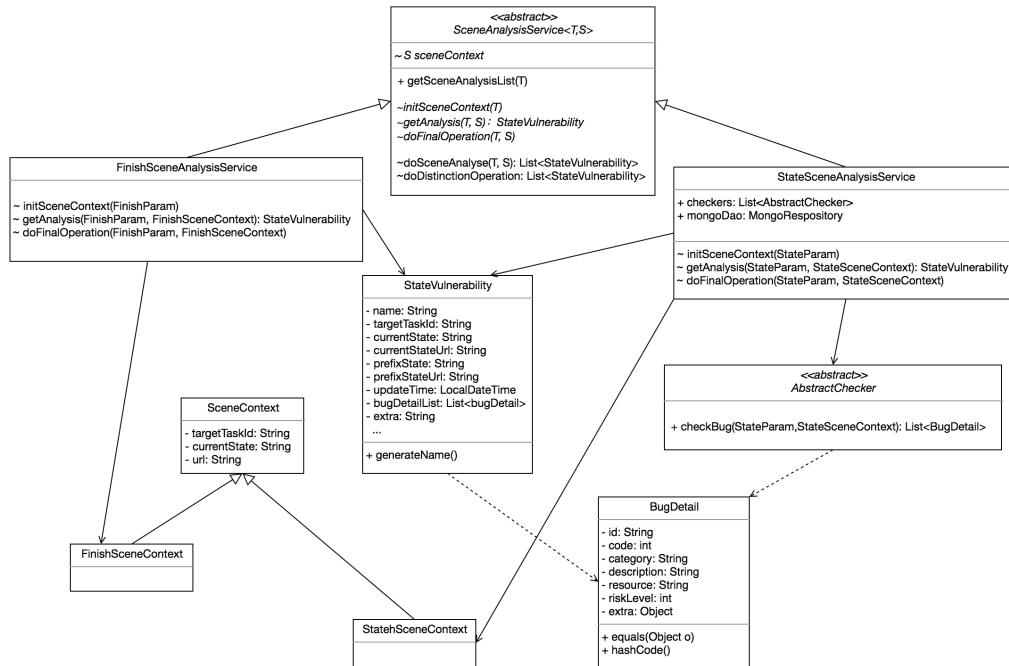


图 4.3: 缺陷分析框架设计类图

的多个子类，实现针对某个特定类型数据的缺陷分析，如控制台日志分析和请求日志分析等等。最终场景下的缺陷分析结果由 `StateVulnerability` 类来负责封装，包括了场景相关信息，以及具体的 `BugDetail` 缺陷列表，详细信息如 4.2 所示。

表 4.1: `StateVulnerability` 类详情列表

| 字段 | 类型 | 含义 | 备注 |
|------------------------------|-----------------|--------|-------------------|
| <code>targetTaskId</code> | String | 任务唯一编码 | 包含了任务 ID 和子任务 ID |
| <code>prefixState</code> | String | 前序场景 | 指到达该场景的上一个场景 |
| <code>prefixStateUrl</code> | String | 前序场景链接 | 上一个场景的标识路径 |
| <code>currentState</code> | String | 当前场景 | 当场场景标识 |
| <code>currentStateUrl</code> | String | 当前场景链接 | 当前场景的标识路径 |
| <code>screenshotPath</code> | String | 当前场景截图 | 存储在 OSS 中，此处为对应链接 |
| <code>bugDetailList</code> | List<BugDetail> | 缺陷列表 | 当前场景下检测出的缺陷列表 |
| <code>updateTime</code> | LocalDateTime | 更新时间 | 更新时间 |
| <code>extra</code> | Object | 扩展字段 | 用于可能会拓展的其他场景属性 |

其中，`BugDetail` 是本文中定义缺陷详情的数据结构，是最小粒度的缺陷描述和整合去重单位，详细描述如表 4.2 所示，包括了缺陷所属类别，具体的缺陷描述和风险等级等信息。

表 4.2: BugDetail 类详情列表

| 字段 | 类型 | 含义 | 备注 |
|-------------|---------|--------|------------------------|
| id | String | 缺陷编号 | 单测试任务下的唯一编号 |
| code | Integer | 缺陷类别代码 | 该缺陷所在类别代码，便于分类统计 |
| category | String | 缺陷类别名称 | 该缺陷所在类别名称 |
| description | String | 缺陷描述 | 关于缺陷的详细描述 |
| resource | String | 缺陷所在资源 | 一般是缺陷所在页面链接或缺陷请求链接 |
| riskLevel | Integer | 缺陷风险等级 | 包括建议、提示、一般、严重和致命五个风险等级 |
| extra | Object | 扩展字段 | 用于可能会拓展的其他缺陷属性 |

4.1.4 关键代码

如图 4.4所示为 SceneAnalysisService 模板类的核心代码。受篇幅限制，图中省略了部分常见的参数检查与流式处理代码，仅展示以下主要代码片段。

```
public abstract class SceneAnalysisService<T, S> {
    //场景上下文
    S sceneContext;
    //服务暴露接口：模板方法
    public List<StateVulnerability> getSceneAnalysisList(T analyseParam){
        //上下文初始化
        initSceneContext(analyseParam);
        //缺陷分析
        List<StateVulnerability> stateVulnerabilityList = doSceneAnalyse(analyseParam, sceneContext);
        //缺陷整合去重
        stateVulnerabilityList = doDistinctionOperation(stateVulnerabilityList);
        //上下文及汇总操作
        doFinalOperation(analyseParam, sceneContext);
        //返回结果
        return stateVulnerabilityList;
    }

    //抽象方法
    abstract StateVulnerability getAnalysis(T analyseParam, S sceneContext);
    abstract void initSceneContext(T analyseParam);
    abstract void doFinalOperation(T analyseParam, S sceneContext);
    //实现方法
    List<StateVulnerability> doSceneAnalyse(T sceneAnalyseParam, S sceneContext){
        ...
        StateVulnerability stateVulnerability = getAnalysis(sceneAnalyseParam, sceneContext);
    }
    List<StateVulnerability> doDistinctionOperation(List<StateVulnerability> stateVulnerabilityList) {
        .....
        bugDetailList.stream().distinct().collect(Collectors.toList());
    }
}
```

图 4.4: 场景分析模板类的部分代码

正如图中的声明代码所示，我们采用了基于泛型结合模板方法的设计思想，SceneAnalysisService 作为一个抽象类，定义了包括 initSceneContext、getAnalysis、doFinalOperation 在内的三个需子类实现的抽象方法以及实现了 doSceneAnalyse、doDistinctionOperation 两个公共方法，并将它们组织成一个完成的缺陷分析流程方法，即 getSceneAnalysisList 方法。在这个过程中，我们首先完成场景上下文的初始化，并交由具体的场景服务类实现场景下的缺陷分析工作，最后在通过重写 BugDetail 类的 equals 与 hashCode 方法实现场景缺陷的去重，并交由场景类完成最后的上下文数据处理工作。同时，场景分析的数据参数和场景上下文通过泛型提供，以最大程度地实现代码的复用和拓展支持。

4.2 控制台日志缺陷聚类子模块

控制台日志缺陷聚类子模块，是控制台日志缺陷检测模块其中的一个子模块，针对浏览器控制台输出日志，进行聚类分析和特征挖掘，尝试性发掘可能存在的缺陷类别，进而构建控制台日志相关的缺陷知识库，服务于后续的控制台日志缺陷判定子模块。

4.2.1 架构设计

控制台日志缺陷聚类子模块的整个架构设计如图 4.5 所示，我们在进行缺陷聚类前，先进行了控制台缺陷日志的搜集和清洗去重等前期工作，以完成聚类数据集的准备。同时抽象日志之间差异的度量方法，提供不同的距离算法，来进行缺陷聚类分析和聚类效果评估，以确定最佳聚类参数和聚类结果。最终，通过人工审查总结，对这些不同的日志簇进行缺陷总结和知识库录入，以辅助后续的缺陷判定流程。

一般来说，控制台日志包括多种级别的信息，如 Info、Warnings 和 Errors 等等，既可能来自于应用开发者自发的控制台输出，也可能是前端代码在执行时发出的警告和报错。为了避免过多无效信息的干扰，我们在 Web 应用自动化遍历执行过程，通过底层的 ChromeDriver，只采集了 Error 级别日志来进行聚类分析，可映射到 ChromeDriver 底层日志数据结构上的 SEVERE 级别日志。同时，为了避免重复采集和相同日志的多次输出，我们在聚类前，也通过 Java8 流式处理的 distinct 方法，对采集的控制台缺陷日志进行了去重处理。

至于聚类算法和距离算法，结合控制台缺陷日志模板化的特性，以及聚类成本和精度要求，最终我们选择了 DBSCAN 这种基于密度的聚类算法，以最经典的算法形式，在当前缺陷日志数据集规模小且缺乏先验知识的场景下，实现

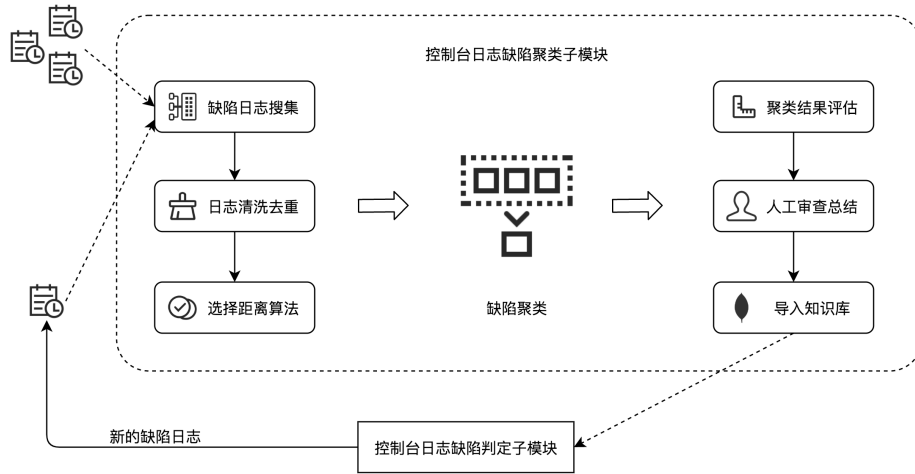


图 4.5: 控制台日志缺陷聚类子模块架构设计图

缺陷日志的类群发现。距离算法，我们则基于 Levenshtein 距离和 Jaccard 系数等算法，实现了不同日志字符串之间文本相似度的比较计算，并借助归一化的文本相似度，反向计算出它们的距离值，以适配于 DBSCAN 的聚类过程。

同时为了达到较好的聚类效果，我们可以针对不同参数，如邻域半径 ϵ 和密度阈值 MinPts 等，来进行多次实验，并借助相关评价指标来进行衡量。常见的评价指标，可根据数据集先验知识的有无，分为包括调整兰德系数 (ARI)、调整互信息 (AMI)、V-measure 以及 FMI 在内的外部指标和轮廓系数等内部指标。鉴于采集的缺陷日志数据集并没有明确的分类依据，以及考虑到后续新缺陷日志引入后的重新聚类，最终，我们采用轮廓系数来衡量我们整个聚类的效果，并确定最后分析使用的聚类结果数据。

最后，对这些归类后的日志集群和噪声点，通过搜索引擎确认和专家评审等人工审查手段，确认缺陷的实际风险和普遍性，并对符合标准的缺陷进行总结和特征入库，以供后续的控制台日志缺陷判定子模块使用。同时，若缺陷判定子模块发现了未识别的严重缺陷日志，还可以将其加入到缺陷日志收集流程，以丰富缺陷日志数据集，进行多次迭代聚类。

4.2.2 流程设计

图 4.6 展示了控制台日志缺陷聚类子模块的系统顺序图，描述了聚类分析的核心流程。当需要进行缺陷日志聚类分析时，`ClusterController` 类作为入口类，会调用 `ClusterService` 类的 `clusterConsoleLog` 方法执行实际的聚类流程。在这个流程中，首先 `ClusterService` 会调用 `ConsoleLogRepository` 类的 `findAll` 方法获取所

有采集的控制台缺陷日志数据,并通过调用 `preprocess` 方法对数据集进行预处理,包括对象转换、去重和结构拆解等步骤。然后执行 DBSCAN 类的 `performCluster` 方法进行基于密度的日志聚类。根据选择的距离计算方式, DBSCAN 会调用继承了 `AbstractDistance` 抽象类的某个具体距离算法类的 `calSimilarity` 方法,用于计算不同日志记录之间的远近关系,并通过调用自身的 `calSilhouetteCoefficient` 计算本次聚类结果的轮廓系数,用于评估聚类效果。最后返回按类聚集的结果哈希表,并调用 `ConsoleLogClusterRepository` 类的 `save` 方法保存聚类结果,以供下一步的人工审查。

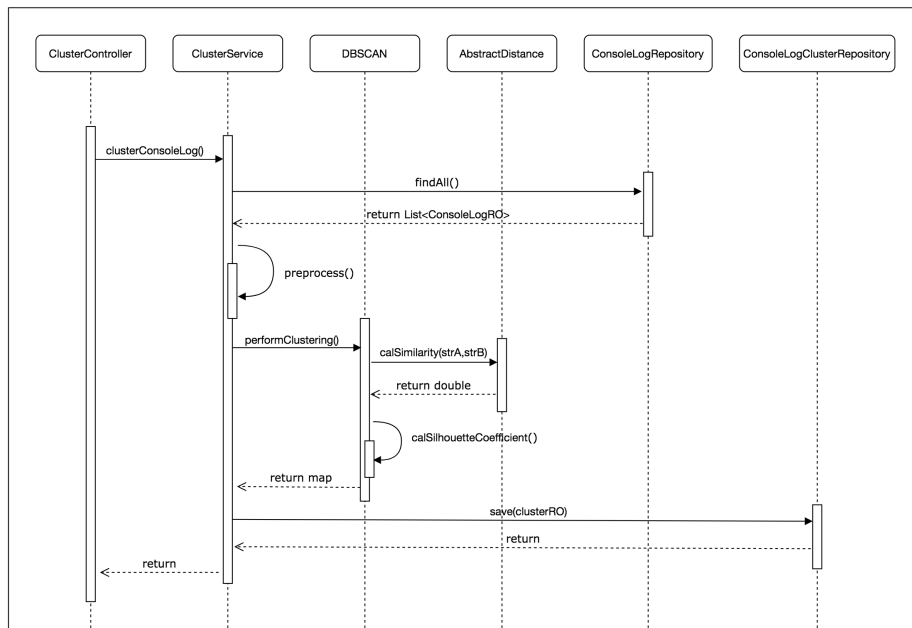


图 4.6: 控制台日志缺陷聚类子模块系统顺序图

4.2.3 数据与类设计

控制台日志缺陷聚类子模块的设计类图如 4.7 所示, `ClusterController` 和 `BugTypeController` 是缺陷聚类分析和缺陷知识库导入两个服务的请求入口, 由具体的 `ClusterService` 和 `BugTypeService` 负责具体业务逻辑的实现。 `ConsoleLogRepository`、`ConsoleLogClusterRepository` 以及 `BugTypeRepository` 都继承自 `MongoRepository`, 封装了对接 MongoDB 数据库的常见操作, 分别负责缺陷日志类 `ConsoleLog`、日志聚类结果类 `ConsoleClusterLog` 以及人工总结入库的缺陷类型类 `BugType` 这几个持久化文档类的数据管理, 以满足实际业务需要, 具体属性可参见第三章持久化模型部分, 这边就不重复赘述。至于 DBSCAN, 作为核心

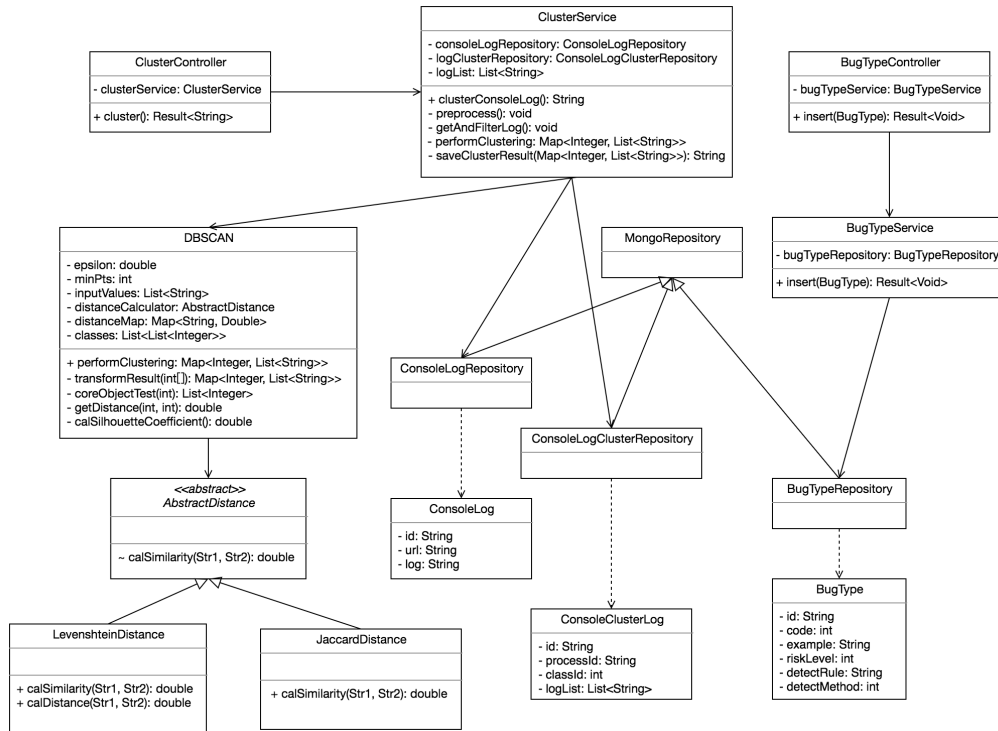


图 4.7: 控制台日志缺陷聚类子模块设计类图

的聚类算法实现类，和基于策略模式的 AbstractDistance、LevenshteinDistance、JaccardDistance 等类一起，完成了控制台日志的实际缺陷聚类过程。

4.2.4 关键代码

```

// 获取控制台缺陷日志并进行预处理
public void preprocess(){
    List<ConsoleLog> consoleLogList = consoleLogRepository.findAll();
    if (!CollectionUtils.isEmpty(consoleLogList)) {
        logger.info(String.format("console log origin size: %d", consoleLogList.size()));
        this.logList = consoleLogROList.stream()
            .map(ConsoleLogRO::getLog)
            .filter(Objects::nonNull)
            .distinct()
            .map(this::refactorLog)
            .collect(Collectors.toList());
        logger.info("console log first filter size: "+ logList.size());
    }
}

```

图 4.8: 控制台缺陷日志预处理部分代码

对数据集的预处理是聚类算法的主要步骤之一，以保证较好的聚类效果。本文在进行缺陷聚类前也对前期采集的 17835 条缺陷日志进行了一定的清洗去重等预处理操作，最终得到一个包含 2046 条数据的聚类数据集，其代码可如图 4.8 所示。正如代码所展现的那样，整个预处理过程，都基于 List 的流处理机制来进行组装，在获取 ConsoleLog 的整个数据集后，我们首先需要从 ConsoleLog 类中取出单独的缺陷日志信息，并通过 filter 和 distinct 操作进行空值处理和去重。同时，根据控制台日志的结构特点，我们通过 refactorLog 方法，将符合“缺陷对应代码位置 + 缺陷描述”规则的控制台日志信息，进行缺陷描述字段的择取，以避免应用之间各异的脚本位置信息对聚类效果造成影响，最终通过 collect 操作完成待聚类输入数据集的聚合。

```
public class DBSCAN {
    // 聚类方法
    public Map<Integer, List<String>> performClustering(){
        ... // 变量初始化+迭代选点，直到所有节点都被访问
        while(unvisitedList.size() != 0) {
            ... // 获取邻域样本，判断是否满足核心对象
            List<Integer> indices = judgeCoreObject(index);
            if(indices.size() >= minPts) {
                ... // 新簇生成，邻域搜索
                do {
                    ... //核心节点队列遍历
                }while(omega.size() != 0);
            } else { ... //噪声点处理 }
        }
        calSilhouetteCoefficient(); // 聚类评估
        return transformResult(types);
    }
    //距离计算
    public double getDistance(int firstIndex, int secondIndex){
        ... //生成 key，避免重复计算
        if(distanceMap.containsKey(key)){
            return distanceMap.get(key);
        }else{
            ...
            double distance = 1.0 - distance.calSimilarity(firstLog, secondLog);
            distanceMap.put(key, distance);
            return distance;
        }
    }
}
```

图 4.9: DBSCAN 聚类算法实现部分代码

图 4.9所示的代码则为具体的 DBSCAN 聚类算法实现，大部分代码逻辑依照 DBSCAN 的传统思路实现，即基于随机选择未访问样本，确定核心对象，并不断在邻域内寻找密度可达节点，使确定一个聚类集群，并继续迭代直到所有节点被访问过。略微不同的是，我们抽象了 `getDistance` 距离计算方法，根据注入的 `AbstractDistance` 实际对象的不同，执行相应的距离计算，并通过 `HashMap` 缓存了中间结果，避免后续计算轮廓系数时对距离的重复计算，提高性能。最终经过多次实验，确定了最后的聚类结果。图 4.10是其中的一个簇集示例，揭示了 JS 类型错误这类缺陷类型。

```
{
  "id": {
    "$oid": "5e89795c9f1755074695c3c8"
  },
  "processId": "38ad13e30a1e446c991eedb2ade8d404",
  "classId": 6,
  "logList": [
    "0:392 Uncaught TypeError: Cannot read property 'isArray' of undefined",
    "38:41 Uncaught TypeError: Cannot read property 'split' of undefined",
    "306:69 Uncaught TypeError: Cannot read property 'i' of null",
    "34:31 Uncaught TypeError: Cannot read property 'clientWidth' of null",
    "0:2107 Uncaught TypeError: Cannot read property 'length' of null",
    "5:10295 Uncaught TypeError: Cannot read property 'adType' of undefined",
    "0:3608 Uncaught TypeError: Cannot read property 'l' of null",
    "1:4304 Uncaught TypeError: Cannot read property 'l' of null",
    "2:40 Uncaught TypeError: Cannot set property 'href' of null",
    "57:40 Uncaught TypeError: Cannot read property 'style' of null",
  ]
}
```

图 4.10: 聚类结果的一个示例簇集

4.3 控制台日志缺陷判定子模块

控制台日志缺陷判定子模块，是控制台日志缺陷检测模块的另外一个子模块，基于聚类分析后总结的控制台日志缺陷库，结合控制台日志的模板化预警信息结构，针对实时获取的控制台日志，进行缺陷分析和判定，并由缺陷分析框架中的状态场景负责接入。

4.3.1 架构设计

控制台日志缺陷判定子模块的架构设计如图 4.11所示，当接收到待检测的日志后，我们可以通过前期总结的缺陷知识库，结合对应的检测方法，构造缺陷判定器，来判定该日志是否存在缺陷。这里我们运用了策略模式加工厂方法组合的设计模式，实现了检测方法的策略替换，并通过工厂方法直接根据反射产生检测对象，实现了进一步的解耦。当前我们提供了特征检查、正则匹配和相似对比三种检测方法，并支持其他检测方法的扩展。同时，鉴于缺陷知识库在整个测试任务周期内的持续使用需要，结合缺陷知识库的变更频率特点，我们也将知识库做了本地缓存处理，并通过合理设置过期时间，以提高整体的性能。

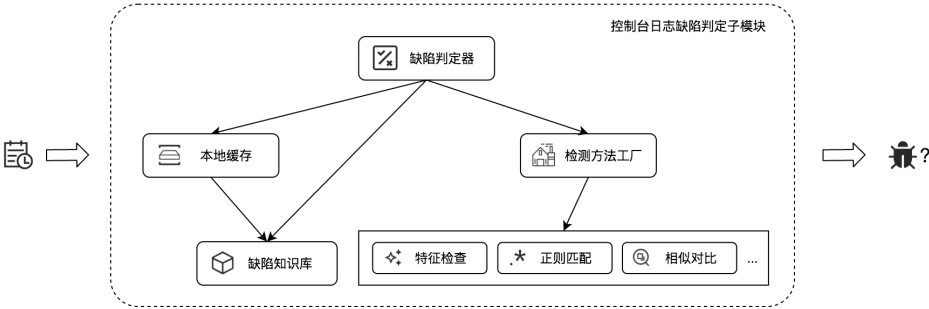


图 4.11: 控制台日志缺陷判定子模块架构设计图

4.3.2 流程设计

图 4.12是控制台日志缺陷判定子模块的系统顺序图，展示了对测试执行过程中采集的控制台日志进行缺陷发现和判定的整个系统调用过程。其中，ConsoleStateChecker 类,是整个子模块的服务入口,由上文所述的 Web 应用缺陷分析框架在状态场景下进行其所需数据的采集和调用接入。它继承自 AbstractChecker,是针对控制台输出日志的专用检测器，希望从控制台日志中检查应用可能存在的缺陷信息，以做下一步改进。

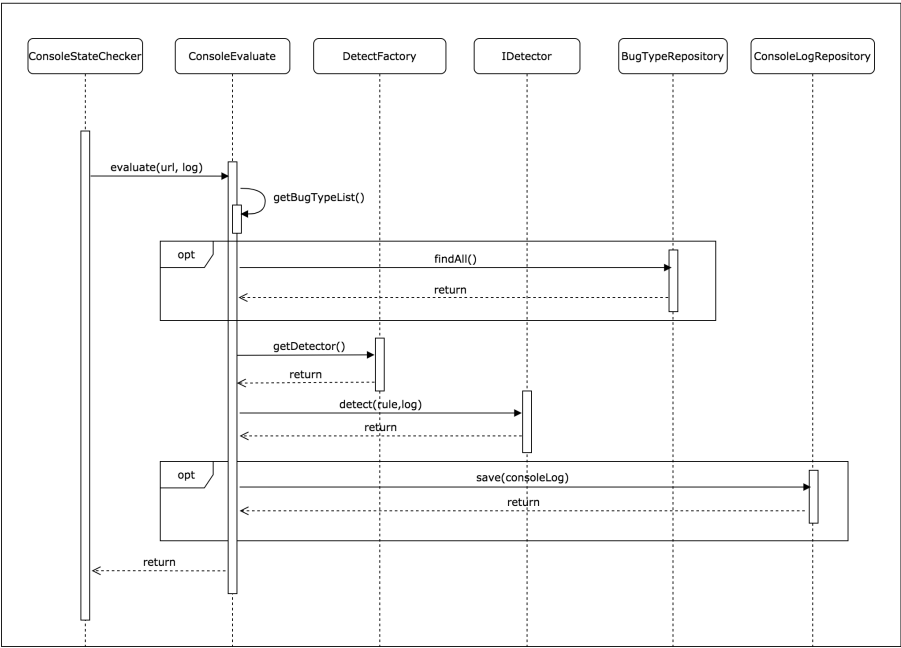


图 4.12: 控制台日志缺陷判定子模块系统顺序图

ConsoleStateChecker 负责管理当前场景下所有采集的控制台日志的流处理过程，实际会调用 ConsoleEvaluate 的 evaluate 方法来进行逐条分析，ConsoleEvaluate 则会调用自身的 getBugTypeList 方法从本地缓存中获取控制台日志缺陷知识库，如果本地缓存中没有该数据，则通过 BugTypeRepository 的 findAll 方法获取。在拿到缺陷知识库信息后，ConsoleEvaluate 会通过 DetectFactory 的 getDetector 方法，获取相应缺陷的检测器，并调用这个实现了 IDetector 接口的检测器的 detect 方法，判定当前日志是否预示着应用的某种缺陷。其中，若该条日志未被标记为某种缺陷，且在控制台日志级别中为 SEVERE 级别，则会调用 ConsoleLogRepository 的 save 方法，保存到控制台日志聚类分析的源数据集合中，以便丰富缺陷知识库。

4.3.3 数据与类设计

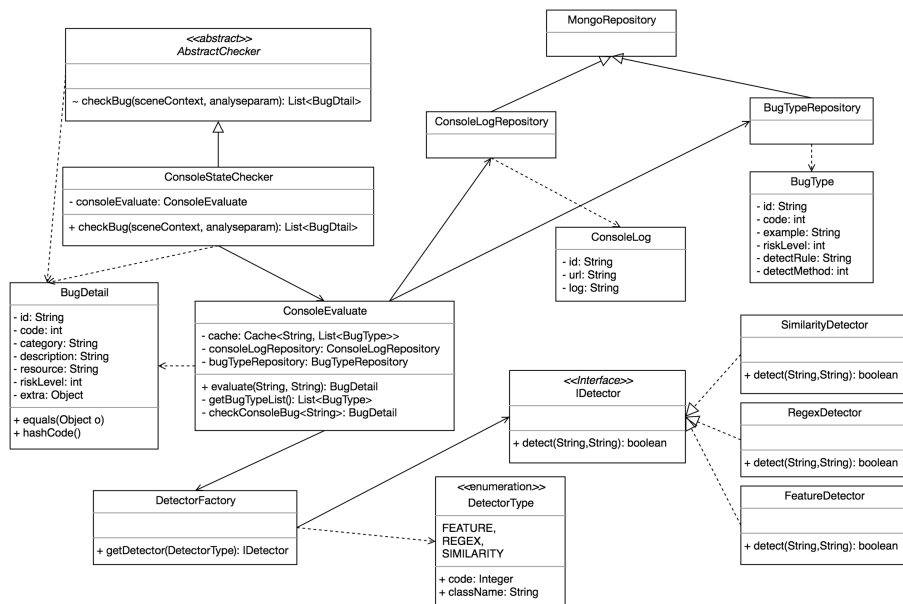


图 4.13: 控制台日志缺陷判定子模块设计类图

控制台日志缺陷判定子模块的设计类图如 4.13 所示，ConsoleStateChecker 继承自 AbstractChecker，实现了 checkBug 方法，是对控制台日志进行缺陷检测的最上层服务提供商，并由具体的 ConsoleEvaluate 类实现对单条日志的缺陷判定，BugDetail 则作为缺陷详情，是整个分析过程被传递的信息中介，具体属性可参见表 4.2。FeatureDetector、RegexDetector、SimilarityDetector 类都实现了 IDetector 接口类，提供包括特征检测、正则匹配以及基于相似度进行比较的多种判定方法，并通过 DetectFactory 工厂类和 DetectorType 枚举映射类，为 ConsoleEvaluate 类，

实现相关检测策略的获取与使用。ConsoleLogRepository 和 BugTypeRepository 类则分别提供了对 ConsoleLog 和 BugType 文档类的数据操作支持。

4.3.4 关键代码

图 4.14展示了缺陷判定子模块中部分最为核心的代码，摘录自 ConsoleEvaluate 类。一方面，我们借助 Google Guava 库提供的缓存机制，实现了对控制台日志缺陷知识库的本地缓存。保证线程安全的同时，也结合缺陷知识库低频定期更新的现状，设置了相应的最大容量限制和失效时间等参数，保证了系统的内存使用安全和知识库数据的更新生效。另一方面，通过策略模式结合工厂方法设计模式的设计，以及 Java 反射机制，定义了缺陷判定策略家族，并将对应类的创建工作交给 DetectorFactory 工厂类，完全解耦了 ConsoleEvaluate 类和实际使用的判定方法类之间的关系，避免了单策略模式下策略暴露的修改风险，同时也更好地支持了后续新的判定方法的引入和扩充。

```
@Component
public class ConsoleEvaluate {
    //缺陷知识库本地缓存
    Cache<String, List<BugTypeRO>> cache = CacheBuilder.newBuilder()
        .maximumSize(10) // 设置缓存的最大容量
        .expireAfterWrite(2, TimeUnit.HOURS) // 设置缓存在写入两小时后失效
        .build();

    private List<BugTypeRO> getBugTypeList(){
        List<BugTypeRO> bugTypeList = cache.getIfPresent(CACHE_KEY_CONSOLE);
        if(bugTypeList == null){
            bugTypeList = bugTypeRepository.findAll();
            cache.put(CACHE_KEY_CONSOLE, bugTypeList);
        }
        return bugTypeList;
    }
    //基于工厂方法调用相应缺陷检测策略
    private BugDetail checkConsoleBug(String msgStr) {
        List<BugTypeRO> bugTypeList = getBugTypeList();
        ... // 参数检测等操作
        for(BugTypeRO bugTypeRO: bugTypeList){
            DetectorType type = DetectorType.getByCode(bugTypeRO.getDetectMethod());
            IDetector detector = DetectFactory.getDetector(type);
            ...
            boolean hasBug = detector.detect(bugTypeRO.getDetectRule(), msgStr);
            if(hasBug){
                ... //生成 BugDetail
                return bugDetail;
            }
        }
        ... // 后续处理
    }
}
```

图 4.14: 控制台日志缺陷判定相关部分代码

4.4 请求日志缺陷检测模块

请求日志缺陷检测模块，是状态场景下接入的另一个重要缺陷检测器，针对通过 Crawljax 插件收集的 HTTP 请求日志碎片，来完成对请求过程的重建，进而对 Web 应用的网络请求状况进行缺陷分析和相关数据析出工作，服务于后续的多维评估模块。

4.4.1 架构设计

整个请求日志缺陷检测模块的架构设计如图 4.15 所示，主要包括请求过程组装和请求缺陷判定两个核心工作步骤，以及从请求组装到缺陷判定过程中提供的相应请求信息管理、页面信息管理和测试环境采集等辅助操作，产出数据可服务于多维评估模块的后续分析过程。

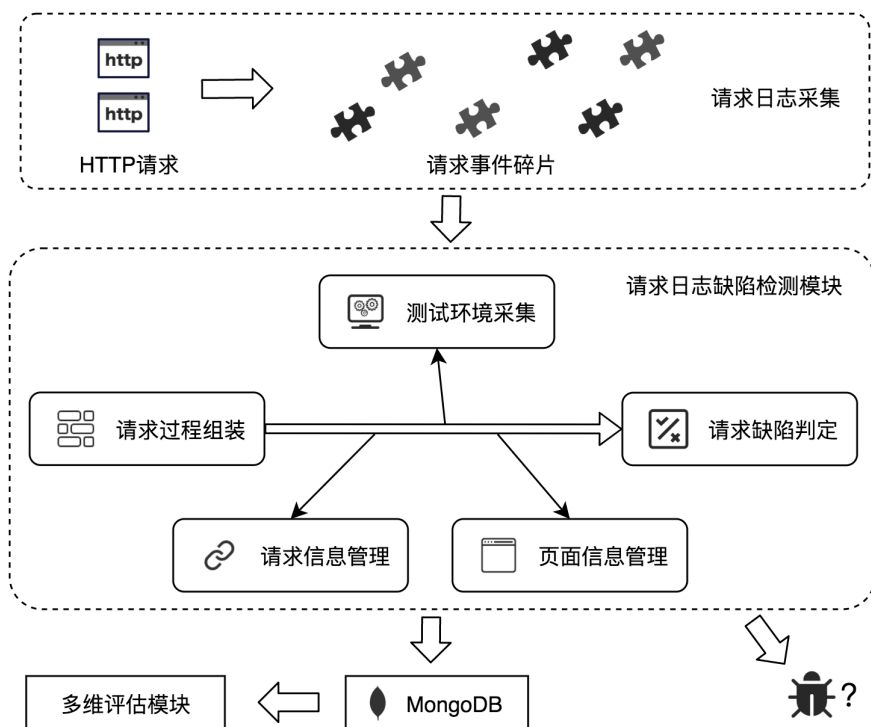


图 4.15: 请求日志缺陷检测模块架构设计图

不同于控制台日志的信息字符串结构，执行服务部分通过 ChromeDriver 采集到的请求日志，是按照 Chrome Devtools Protocol 规范进行组织的 LogEntry 对象所对应 JSON 字符串，涉及请求有效内容的部分以子 JSON 字符串的形式，放

在该 LogEntry 对象的 message 属性中，并通过 requestId 和 method 属性，确定日志的信息类型。图 4.16 是采集请求日志的一个示例，包裹了如图所示的 requestId 在 Network.loadingFinished 事件下相应的数据信息。

```

    {
      "level": {
        "localizedName": "信息",
        "name": "INFO",
        "resourceBundleName": "sun.util.logging.resources.logging"
      },
      "message": {
        "message": {
          "method": "Network.loadingFinished",
          "params": {
            "encodedDataLength": 0,
            "requestId": "D401E4349D75867191B385D84A1E6BD5",
            "shouldReportCorbBlocking": false,
            "timestamp": 115971.066164,
            "webview": "8495E76426A401E8CF7B0220E759B8F7"
          }
        },
        "timestamp": 1574326920457
      }
    }
  
```

图 4.16: 请求日志原始格式的一个示例

根据 Chrome Devtools Protocol 的机制，跟网络请求过程核心相关的主要是 Network 中的 requestWillBeSent、responseReceived、loadingFinished 以及 loadingFailed 四个事件，分别对应每个 http 请求发送前、首次接收到 http 响应时、请求加载完成时以及请求加载失败时对应的回调操作和数据信息，其中，大部分完整的请求过程都会包括 requestWillBeSent、responseReceived 以及 loadingFinished 这三个阶段，除非请求加载失败，或者是某些特定的资源类型请求，如 Font 等。因此，我们可以基于全局唯一的 requestId，完成单请求过程的组装和建模。

至于请求日志的缺陷判定，不同于控制台日志缺乏相应缺陷体系和先验知识，请求过程本身的构建特点就可以帮助我们有效识别缺陷。一方面，loadingFailed 事件提供了对请求失败事件的信息提示，另一方面，成功请求响应体数据的 HTTP 响应状态码属性，本身就是一个请求分类机制。因此，我们总结了请求资源加载失败、页面或操作断链、服务端错误等多种请求日志缺陷类别，并利用上述判断机制，实现了对请求日志的缺陷分析。

测试环境采集，则是对缺陷的信息补充操作，通过分析获取请求体数据中的 User-Agent 信息，提取日志产生时的测试环境信息，进一步丰富缺陷报告中的上下文信息。请求信息管理和页面信息管理操作，则是在缺陷分析的基础上，过滤出成功请求，实现对请求数据的信息摘要和请求分析，析出响应时间和异常页面等中间数据，服务于后续的多维评估模块。

4.4.2 流程设计

图 4.17 展示了请求日志缺陷检测模块的系统顺序图，详细刻画了对请求日志进行缺陷检测和信息摘要的整个系统调用过程。类似于 ConsoleStateChecker 类，继承自 AbstractChecker 的 RequestStateChecker，是请求日志检测器的入口，负责对接状态场景下的请求日志输入，通过调用自身的 unionRequestLogs 方法，根

据 requestId 的唯一性，完成请求数据的过程组装和构建，并调用具体 RequestEvaluate 类的 evaluate 方法，实现对这些请求过程的具体缺陷分析和信息总结。RequestEvaluate 类在执行 evaluate 方法时，会循环调用重载的另一个 evaluate 方法，逐个对请求过程进行分析，以判断是否存在缺陷。analyseTestEnvironment 方法和 RiskEvaluate 的 evaluateRiskLevel 方法的调用，则分别负责测试环境信息的采集和缺陷严重等级的判断，以补充缺陷上下文信息。其中，若请求成功或为页面请求时，则可分别通过调用 RequestRepository 和 PageInfoRepository 的 save 方法进行相应请求和页面信息的持久化存储。

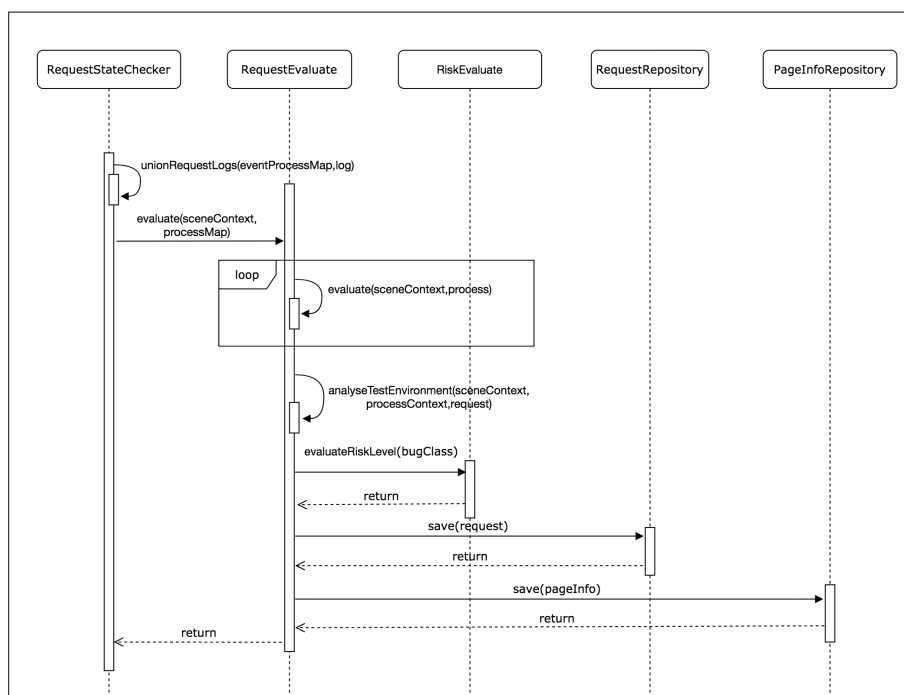


图 4.17: 请求日志缺陷检测模块系统顺序图

4.4.3 数据与类设计

请求日志缺陷检测模块的设计类图如 4.18所示，RequestStateChecker 继承自 AbstractChecker，实现了 checkBug 方法，是对请求日志进行缺陷检测的最上层服务提供商，实现了请求过程建模和组装工作，由 RequestEventProcess 类负责维护请求过程。并调用具体的 RequestEvaluate 类实现对请求过程的缺陷判定和信息收集，由 BugDetail 类负责维护缺陷信息，RiskEvaluate 和 BugClass 类负责补充缺陷信息，ProcessContext 上下文类则提供了整个过程中的上下文属性支持。RequestRepository 和 PageInfoRepository 类则分别提供了对 Request 请

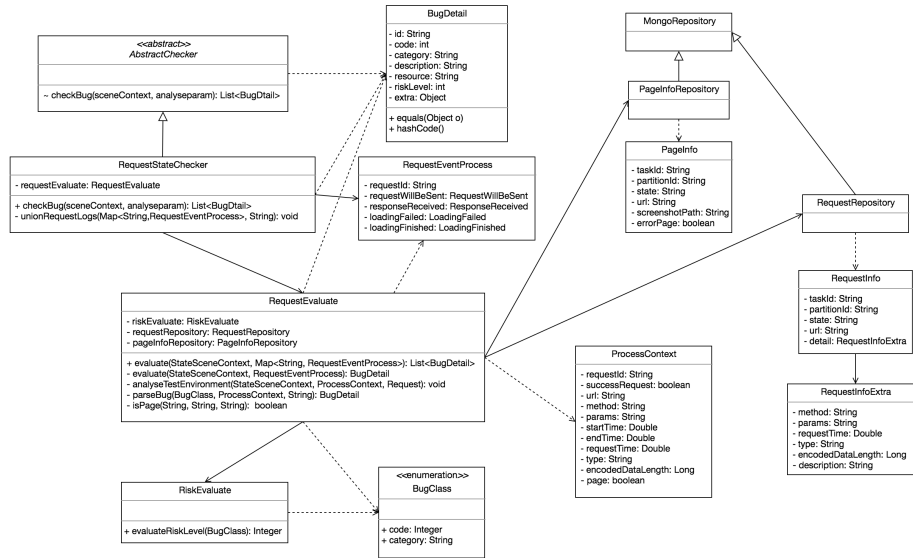


图 4.18: 请求日志缺陷检测模块设计类图

求数据和 PageInfo 页面数据文档类的数据库操作支持，Request 和 PageInfo 的详细信息可参见概要设计中持久化模型部分。其中，ProcessContext 类，是在单个 RequestEventProcess 请求过程分析过程中使用的上下文类，提供了请求数据部分关键属性和相应状态参数，辅助于缺陷检测和信息收集工作流程，详细描述可如表 4.3 所示。

表 4.3: ProcessContext 类详情列表

| 字段 | 类型 | 含义 | 备注 |
|-------------------|---------|---------|---------------------------|
| url | String | 请求链接 | 不唯一 |
| requestId | String | 请求 ID | 一个 HTTP 请求的唯一标识 |
| method | String | 请求方法 | 如 GET、POST 等 |
| params | String | 请求参数 | 请求提交的参数，JSON 格式 |
| type | String | 资源类型 | 如 Image、XHR 等 |
| startTime | Double | 请求开始时间 | 时间戳格式 |
| endTime | Double | 请求结束时间 | 时间戳格式 |
| requestTime | Double | 请求时间 | 根据 startTime 和 endTime 计算 |
| encodedDataLength | Long | 响应字节数 | 单位为字节 |
| successRequest | boolean | 请求是否成功 | 根据请求流程的完整性和响应状态码判断 |
| page | boolean | 是否为页面请求 | 通过类型判断 |

RequestEventProcess 类则是我们定义的单个请求模型类，用以重建单个请求从发送到接收整个过程的事件信息结构，详细信息可如 4.4 所示，包含了唯一标识的请求 ID 等信息。

表 4.4: RequestEventProcess 类详情列表

| 字段 | 类型 | 含义 | 备注 |
|-------------------|-------------------|-----------|-------------------------------|
| requestId | String | 请求 ID | 一个 HTTP 请求的唯一标识 |
| requestWillBeSent | RequestWillBeSent | 请求发送前数据对象 | 日志 method 为 requestWillBeSent |
| responseReceived | ResponseReceived | 接收响应时数据对象 | 日志 method 为 responseReceived |
| loadingFailed | LoadingFailed | 加载失败时数据对象 | 日志 method 为 loadingFailed |
| loadingFinished | LoadingFinished | 加载完成时数据对象 | 日志 method 为 loadingFinished |

4.4.4 关键代码

图 4.19展示了请求日志缺陷检测模块中组装请求过程的部分代码，摘录自 RequestStateChecker 类。我们通过 requestId 的唯一性，构建用于保存请求过程的 Map 对象，并将其作为 Map 对象中的 key 值，value 值则为该请求过程的模型类，即 RequestEventProcess 类。同时逐条解析日志，根据 method 属性的不同，通过 fastjson 库，将日志转化到不同的事件数据对象中，并在过程模型类中设置相应属性值，完成请求过程的构建。

```
private void unionRequestLogs(Map<String, RequestEventProcess> eventProcessMap, String log) {
    try {
        ... // 获取日志实体内容
        String methodStr = (String) JSONUtils.getKeyJsonQuery(msgStr, KEY_METHOD); // 获取事件和对象
        JSONObject jsonObject = (JSONObject) JSONUtils.getKeyJsonQuery(msgStr, KEY_PARAMS);
        String requestId = jsonObject.getString(KEY_REQUEST_ID); // 获取请求 ID
        //获取请求过程
        RequestEventProcess process;
        if (!eventProcessMap.containsKey(requestId)) {
            process = new RequestEventProcess();
        } else { process = eventProcessMap.get(requestId);}
        EventType eventType = EventType.getTypeByMethod(methodStr); // 根据事件分类
        switch (eventType) {
            case REQUEST_WILL_BE_SENT:
                RequestWillBeSent first = JSON.toJavaObject(jsonObject, RequestWillBeSent.class);
                process.setRequestWillBeSent(first); break;
            case RESPONSE_RECEIVED:
                ResponseReceived second = JSON.toJavaObject(jsonObject, ResponseReceived.class);
                process.setResponseReceived(second); break;
            case LOADING_FAILED:
                LoadingFailed third = JSON.toJavaObject(jsonObject, LoadingFailed.class);
                process.setLoadingFailed(third); break;
            case LOADING_FINISHED:
                LoadingFinished fourth = JSON.toJavaObject(jsonObject, LoadingFinished.class);
                process.setLoadingFinished(fourth); break;
            case OTHERS:
                // ...
        }
        eventProcessMap.put(requestId, process);
    } catch (Exception ignored) { ... } //异常处理
}
```

图 4.19: 请求日志过程构建部分代码

图 4.20 则展示了对单个请求进行缺陷检测的部分代码，摘录自 `RequestStateChecker` 类。根据不同请求具有的事件数据的不同，按对象类型进行缺陷检测或者请求关键属性的采集。首先，通过分析 `RequestWillBeSent` 对象，我们可以获取相应的 `Request` 参数，更新请求上下文的相关字段，并实现测试环境的信息采集。其次，当出现 `LoadingFailed` 对象时，我们就可以对这个请求缺陷建模，完成对加载失败缺陷的跟踪。如果成功接收到 `ResponseReceived` 对象的话，我们还可以进行针对响应体数据的分析，根据状态码，确定该请求是否存在缺陷，这里我们只考虑 4XX 和 5XX 等响应码对应的缺陷，并更新上下文。`LoadingFinished` 对象，则提供了对时间戳和响应字节码属性的数据支撑。最后，基于请求上下文对象数据值，按规则执行加载时间、存储请求和页面数据的相关操作。

```
private BugDetail evaluate(StateSceneContext sceneContext, RequestEventProcess process) {
    ...
    ProcessContext processContext = new ProcessContext();
    try {
        //发送请求分析
        RequestWillBeSent requestWillBeSent = process.getRequestWillBeSent();
        ...
        analyseTestEnvironment(sceneContext, processContext, request); //评估测试环境并更新请求
        processContext.setStartTime(requestWillBeSent.getTimestamp()); //上下文管理
        ...
        boolean isPage = isPage(requestUrl, contextUrl, type); // 判断是否为页面
        // 加载失败分析
        LoadingFailed loadingFailed = process.getLoadingFailed();
        detail = evaluate(processContext, loadingFailed);
        // 返回内容分析
        ResponseReceived responseReceived = process.getResponseReceived();
        detail = evaluate(processContext, responseReceived);
        // 加载完成分析
        LoadingFinished loadingFinished = process.getLoadingFinished();
        ...
        calRequestTime(); //加载时间处理
        requestRepository.save(requestInfo); //存储请求
        savePage(sceneContext, isError); //存储页面
        ...
    } catch (Exception ignored) { ... //异常处理 }
}
```

图 4.20: 单个请求缺陷检测部分代码

4.5 软件质量多维评估模块

软件质量多维评估模块，主要基于前期测试执行和缺陷分析等工作根据收集的中间数据产物，负责从多个维度对 Web 应用进行切面分析，并提供给用户对应维度下的下钻数据和可能的改进建议，以及参考的维度量化评分，是对 Web 缺陷分析工作的补充支持。当前，系统提供对性能、兼容性、健壮性、稳定性以及功能维度下的分析支持，并支持其他维度的扩增接入。

4.5.1 架构设计与维度分析

整个多维评估的架构设计图如图 4.21 所示，在前期的执行和分析工作中，我们已经收集了测试过程中的一些基础数据，如请求信息，操作信息，缺陷信息和页面信息等，并交由 MongoDB 和 Redis 进行维护。当相关的测试执行和缺陷分析工作结束后，系统会调用多维评估模块，来进行对这些数据的维度拆解和量化评分，最终将拆解的数据和维度评分，交由报告生成服务进行渲染和前端展示。在这个过程中，维度评估集成器负责维度任务的组装和扩展支持，并交由对应的维度评估组件实现各自维度的信息组织和评分计算，如性能评估、健壮性评估、稳定性评估等。

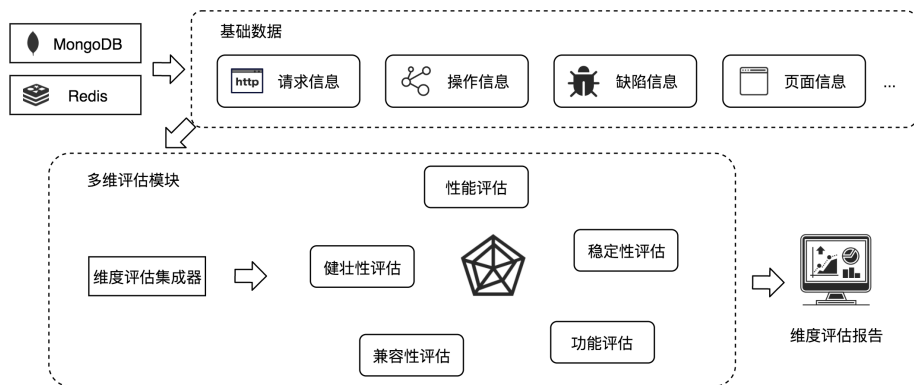


图 4.21: 软件质量多维评估模块架构设计图

其中，性能评估，负责衡量 Web 应用的性能表现，常见的评价指标主要包括请求响应时间、吞吐量，以及并发数等 [43]。鉴于系统目前并未支持负载测试，所以我们主要是针对请求响应时间这一指标进行深度分析。根据业界经验，我们分别给静态资源请求和异步调用请求设置了对应的性能阈值，对中间收集的全部成功请求数据进行超时判定，并向用户展示可能存在性能问题的请求，同时按静态资源和异步调用两类请求加权计算未超时请求占总体请求的比例作为性能评分。

兼容性评估，则是来评价 Web 应用对不同平台环境的运行兼容，常见的评估方向包括行为不一致性、结构不一致性，以及内容不一致性 [44]。这边我们主要探讨 Web 应用在不同环境下的行为不一致性。基于有限状态机的自动遍历执行，我们生成并展示了应用在不同测试环境下的测试路径图。其中，每一个节点代表一个应用状态，每一条边则代表一项操作。我们通过逐个对比同状态下的操作序列，来分析不同环境下状态页面的可达性，进而确认对应功能是否失效，

从而刻画应用的行为不一致性。最后根据待测应用在不同测试环境下的兼容状态占比，加权计算得到应用的兼容性评分。

健壮性评估，主要是用来衡量一个应用在各种缺陷故障条件下的生存和恢复能力。鉴于自动化遍历过程无法持续监控应用状态，所以无法给出常见的健壮性评估指标。这里我们根据分析到的应用缺陷信息，借助缺陷严重等级和缺陷数量，来进行缺陷分的累加和单状态下平均缺陷分的计算，并通过扣分制反向刻画应用遇到缺陷故障时的容错能力，同时为用户提供简化的缺陷列表概览。

稳定性评估，是对同一平台环境的稳健运行状态的有效分析，探讨影响运行状态的主要场景，如页面无法访问等情况。这里我们根据收集到的应用页面信息，计算非异常页面占总页面的比例，作为稳定性评分，同时展示对应异常页面列表，作为稳定性改进提示。

至于功能评估，目前主要是分析应用构建的基本工作量和状态变化情况，来对应用的复杂程度进行初步刻画。我们向用户提供了应用各个状态之间的功能导航图。同时基于状态数、可交互操作数，以及运行时间等指标，加权计算单位时间内的应用复杂度，作为该 Web 应用的功能得分。后续，我们可结合具体的网站开发需求，引入功能验证点更准确地来刻画应用的功能满足情况。

4.5.2 数据与类设计

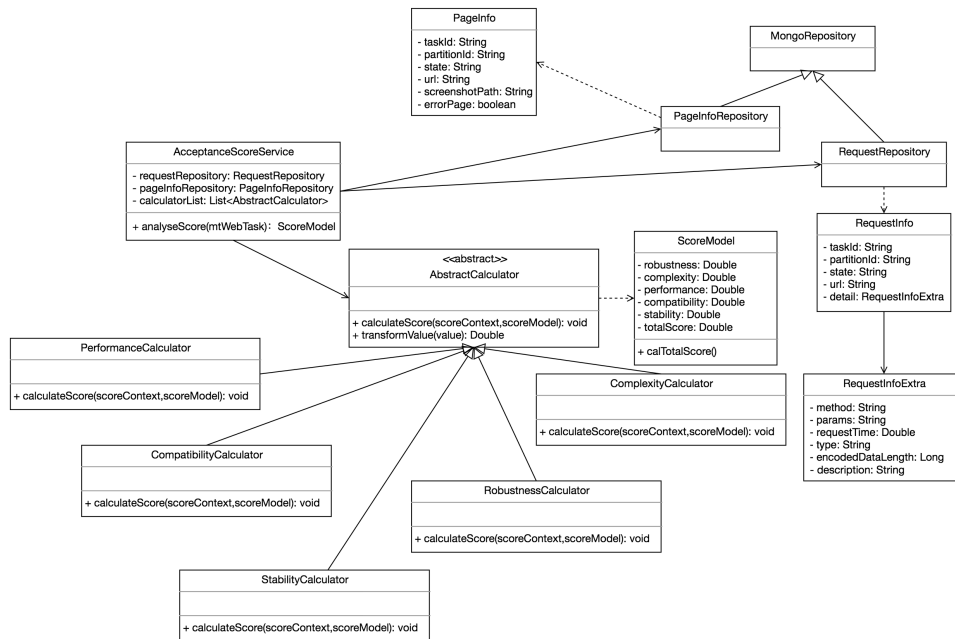


图 4.22: 软件质量多维评估模块设计类图

软件质量多维评估模块的设计类图如 4.22 所示，AcceptanceScoreService 是整个多维评估的分析入口，在测试执行过程和前序缺陷分析工作结束后，由 ReportListener 进行调用。AbstractCalculator 是模块评估类的抽象父类，定义了维度检测的集成规范，支持后续其他维度的按需引入，并暴露了 calculateScore 方法以供上游组件调用使用。

CompatibilityCalculator、ComplexityCalculator、StabilityCalculator、PerformanceCalculator、RobustnessCalculator 等类均继承自 AbstractCalculator，分别实现了对兼容性、功能、稳定性、性能以及健壮性等维度的评估服务，并借助 ScoreModel 提供多维评估上下文管理。RequestRepository 和 PageInfoRepository 类则分别提供了对 Request 请求数据和 PageInfo 页面数据文档类的数据操作支持。

4.5.3 关键代码与界面截图

```
public void calculateScore(ScoreContext scoreContext, ScoreModel scoreModel) {
    ... //参数检查与处理
    // 按测试环境聚合操作路径
    Map<String, List<StateNodeRO>>> partitionGroup = stateNode.stream()
        .collect(Collectors.groupingBy(StateNodeRO::getPartitionId));
    //单状态逐一比较，key 为状态标识，value 为调用链
    Map<String, List<List<TestNode>>>> singleStateList = Maps.newHashMap();
    //对照组
    Map<String, List<TestNode>>> standardStateList = Maps.newHashMap();
    //对照策略
    partitionDelegate.delegate(scoreContext, singleStateList, standardStateList, partitionGroup);
    ...
    //首页打开情况分析
    double indexScore = analyseIndexState();
    //一致性状态分析
    for(String key: standardStateList.keySet()){
        List<TestNode> stdStateList = standardStateList.get(key);
        List<List<TestNode>>> judgeStateList = singleStateList.get(key);
        //对比状态一致性
        intimateCount += compare(stdStateList, judgeStateList);
    }
    ...
    //计算一致状态比列
    double score = indexScore * indexWeight + rateWeight * intimateCount / stateCount / environmentCount;
    scoreModel.setCompatibility(transformValue(score));
    ...//保存数据
}
```

图 4.23: 兼容性评估相关部分代码

不同维度评估，分析的数据和评估计算方式都各有差异，这里我们以兼容性评估的实现为例，解释兼容性维度评估的实现过程。图 4.23 展示了对兼容性评估的部分代码，摘录自 CompatibilityCalculator 类，实现了 AbstractCalculator 的 calculateScore 方法，通过 SpringBoot 的 @Order 注解实现维度检测器的按序集成。在完成参数检查与处理后，我们通过流处理机制和分组聚合操作，实现按测

试环境的操作路径聚合。然后完成对照组和一般组的数据容器的初始化，并根据对照策略进行数据容器的数据装载，以供后续分析。当前对照策略按照状态数的值进行对照组选择。接着，完成首页打开情况分析和一致性状态分析过程，分别得到兼容性的基础分和一致性状态比例，最后按照设置的权重计算兼容性总分，并保存报告中兼容性页面渲染所需数据。

图 4.24所示是对某个在线计算器网站进行自动化测试后的兼容性评估详情页面，展示了该网站在多个测试环境下的测试路径，包括了 Chrome、Firefox 等浏览器环境，并提供了每个状态路径的单独查看功能。用户可以通过比较相同状态操作路径之间的差异，对比分析待测应用在不同测试环境下的表现。

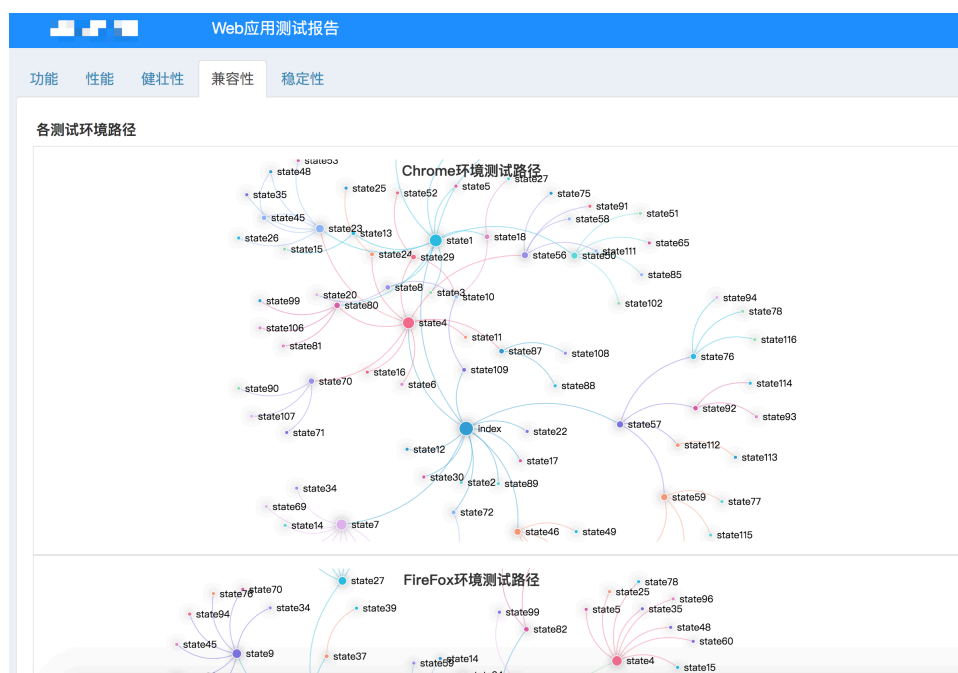


图 4.24: 某应用兼容性维度评估页面

4.6 本章小结

本章是本文的核心内容，展示了分析服务几个模块的详细设计和代码实现过程，借助系统顺序图、类图、代码片段等展现形式，通过架构分析、流程分析、数据和类设计等手段完成了缺陷分析框架、控制台日志缺陷聚类子模块、控制台日志缺陷判定子模块、请求日志缺陷检测模块以及多维评估模块的设计思路 and 实现解读，并提供了核心部分代码的截图和相关说明。

第五章 系统测试与实验分析

为进一步确认和验证 Web 应用自动化测试系统的可用性和有效性，本文也对系统进行了相关测试和实验设计。具体地，我们通过使用多组测试用例来验证系统业务流程的可用性，并基于 50 个真实线上网站对系统的缺陷发现和报告可用性进行实验研究。

5.1 系统测试

5.1.1 测试目标与测试环境

本文的测试主要面向系统功能的正确性和完整性来展开，基于部署的真实生产环境，对系统承诺的目标功能，验证其是否正确实现，能否满足用户和业务的期望，以及最终产品是否符合规格等等。具体地，主要包括对创建、查看、启动和停止 Web 应用自动化测试任务，审核自动化测试任务，查看测试报告与缺陷详情，查看维度分析等功能的测试。我们可通过设计和执行这些功能的测试用例来完成相应的测试过程。

表 5.1: 测试环境说明

| 服务器 | 环境基础配置 | 部署服务说明 |
|-------|---|--|
| 用户计算机 | 配置: 4 核 16G 系统: MacOS 10.15.3 | Chrome 浏览器: 79.0.3945.88 |
| 阿里云-A | 配置: 2 核 4G JAVA 版本: 1.8.0.242 系统: Ubuntu 16.04.6 LTS | 测试任务管理: 维护测试任务的生命周期 |
| 阿里云-B | 配置: 8 核 16G JAVA 版本: 1.8.0.242 系统: Ubuntu 16.04.6 LTS | Redis 数据库: 3.0.6 MongoDB 数据库: 4.2.2 RabbitMQ 消息队列: 3.8.2 Selenium 测试框架: 4.2.2 分析服务: 对测试数据的分析挖掘 |
| 阿里云-C | 配置: 8 核 16G 系统: Windows Server 2016 | Selenium 测试框架: 4.2.2 浏览器集群: 包括 Chrome、FireFox、IE 等浏览器 |

如表 5.1 所示是系统部署和测试的相关环境说明，包括相关硬件配置和部分服务的版本信息。其中，考虑现实因素，本系统的相关服务主要部署在三台阿里云服务器。A 服务器为 2 核 4G 的低配 Linux 服务器，主要负责对接用户操作，进行测试任务生命周期管理，后续随着用户数量和访问次数的增加可提升相关

配置；B 服务器由于负责部署核心的分析服务，以及承载了较多中间件服务，如 Redis、MongoDB、RabbitMQ 等，所以采用了 8 核 16G 的高配置 Linux 服务器；C 服务器由于需要提供真实的多浏览器测试环境，所以也采用较高配置的 8 核 16G 服务器，不同的是，其系统暂采用 Windows Server，来提供对 IE 浏览器的额外支持。后续随着测试资源的增加，可实现多平台多浏览器集群测试环境。用户端则使用个人设备，通过 Chrome 浏览器访问系统。

5.1.2 功能测试

功能测试，是对产品的功能进行逐项验证，检查是否满足用户需求的一种黑盒测试手段。本小节，将在第三章需求分析部分产出功能需求的基础上，借助等价类划分、边界值等功能测试方法，对系统提供的功能进行相应测试用例的设计和测试执行，进而完成对系统功能可用性和完整性的检测。其中，测试用例主要是对测试功能场景和操作步骤的高度刻画，是按照编码、测试说明、测试操作步骤、预期结果和测试结果等分项进行表述的结构化产物。

表 5.2: 创建自动化测试任务相关测试用例

| ID | 测试说明 | 操作/输入 | 预期结果 | 测试结果 |
|---------|------------|--------------------------------------|-------------------|------|
| TC1-1-1 | 输入测试目标 | 测试人员输入某个待测应用网址，并点击上传 | 系统显示操作成功，进入补充配置页面 | 通过 |
| TC1-1-2 | 输入错误测试目标 | 输入非网址错误字段，并点击上传 | 系统提示不是一个有效网址 | 通过 |
| TC1-1-2 | 选取历史测试目标 | 点击选取测试过的测试目标 | 系统进入该任务的补充配置页面 | 通过 |
| TC1-2-1 | 输入通用配置 | 输入任务目标，任务限时、最大页面状态数以及最大页面搜索深度参数，点击保存 | 系统显示输入数值 | 通过 |
| TC1-2-2 | 必填通用配置未输入 | 未输入任务目标点击保存 | 系统提示有必填项未输入 | 通过 |
| TC1-3-1 | 输入浏览器配置 | 点击添加浏览器配置后选择平台类型和浏览器类型，并输入浏览器数量，点击保存 | 系统显示添加的浏览器配置 | 通过 |
| TC1-3-2 | 必填浏览器配置未输入 | 未输入平台类型就点击保存 | 系统提示有必填项未输入 | 通过 |
| TC1-4-1 | 输入权限配置 | 点击添加权限配置后输入事件类型、搜索方式、目标元素等信息，点击保存 | 系统显示添加的权限配置 | 通过 |
| TC1-4-2 | 必填权限配置未输入 | 未输入事件类型就点击保存 | 系统提示有必填项未输入 | 通过 |
| TC1-5-1 | 提交测试任务 | 点击提交测试 | 系统显示提交成功 | 通过 |

表 5.2是针对 UC1 功能需求描述的功能进行展开的测试用例设计, 主要用于测试 Web 应用自动化测试任务的创建过程, 包括了选择测试目标、输入通用配置、输入浏览器配置以及输入权限配置等测试项。该测试用例主要验证创建 Web 自动化测试任务功能的正常流程能否正确运行, 以及在用户输入不符合规范的时候, 系统能否及时给予相应的错误提示。最后全部通过的测试结果表明, 该功能符合预期需求。

表 5.3: 审核自动化测试任务相关测试用例

| ID | 测试说明 | 操作/输入 | 预期结果 | 测试结果 |
|---------|--------|---------------|-------------------------|------|
| TC2-1-1 | 查看审核任务 | 点击菜单栏进入任务审核页面 | 系统按时间倒序分页显示全部用户的自动化测试任务 | 通过 |
| TC2-2-1 | 查看任务详情 | 点击某个具体任务详情按钮 | 系统跳转至该任务详情页面 | 通过 |
| TC2-3-1 | 通过审核 | 点击审核通过按钮 | 系统更新测试任务状态为“审核通过” | 通过 |
| TC2-3-2 | 不通过审核 | 点击拒绝按钮 | 系统更新测试任务状态为“审核未通过” | 通过 |

表 5.3是针对 UC8 功能需求描述的功能进行展开的测试用例设计, 主要用于测试管理员用户审核 Web 应用自动化测试任务的操作过程, 包括了测试任务的查看、通过审核以及不通过审核等测试项。需注意的一点是, 在执行本测试前需准备两个待审核任务以支持审核操作。最后全部通过的测试结果表明, 该功能符合预期需求。

表 5.4: 查看自动化测试任务相关测试用例

| ID | 测试说明 | 操作/输入 | 预期结果 | 测试结果 |
|---------|------------|-----------------------|------------------------|------|
| TC3-1-1 | 查看审核任务 | 测试人员点击菜单栏进入任务列表页面 | 系统按时间倒序分页显示该用户的自动化测试任务 | 通过 |
| TC3-2-1 | 模糊搜索任务-存在 | 输入存在的任务名模糊值 | 系统显示匹配到的测试任务列表 | 通过 |
| TC3-2-2 | 模糊搜索任务-不存在 | 输入不存在的任务名模糊值 | 系统显示暂无匹配测试任务列表 | 通过 |
| TC3-3-1 | 测试进度筛选-存在 | 选择某个存在任务的测试进度, 如审核通过 | 系统显示对应测试进度的任务列表 | 通过 |
| TC3-3-2 | 测试进度筛选-不存在 | 选择某个不存在任务的测试进度, 如执行失败 | 系统显示对应测试进度的任务列表 | 通过 |

表 5.4是针对 UC2 功能需求描述的功能进行展开的测试用例设计, 主要用于对测试人员查看 Web 应用自动化测试任务的操作过程进行测试, 包括了测试

任务列表的查看、任务名模糊搜索以及按测试进度筛选等测试项，除了正常流程外，还检验系统对筛选后空列表的显示支持。最后全部通过的测试结果表明，该功能符合预期需求。

表 5.5: 启动自动化测试任务相关测试用例

| ID | 测试说明 | 操作/输入 | 预期结果 | 测试结果 |
|---------|--------|--------------------|----------------|------|
| TC4-1-1 | 执行测试任务 | 测试人员选择可执行任务并点击执行操作 | 系统更新当前任务状态为执行中 | 通过 |
| TC4-2-1 | 停止测试任务 | 点击停止按钮停止正在执行的任务 | 系统更新当前任务状态为已停止 | 通过 |

表 5.5是针对 UC3 功能需求描述的功能进行展开的测试用例设计，主要用于对测试人员执行 Web 应用自动化测试任务的操作过程进行测试，包括了对应测试任务的执行和停止测试项，在进行测试前，需提前准备一个已通过审核的测试任务以供操作。最后全部通过的测试结果表明，该功能符合预期需求。

表 5.6: 测试报告相关测试用例

| ID | 测试说明 | 操作/输入 | 预期结果 | 测试结果 |
|---------|--------|----------------------|---|------|
| TC5-1-1 | 查看测试报告 | 测试人员点击某个执行完成任务查看报告操作 | 系统显示对应应用的测试报告详情页，包括应用 URL、扫描状态数等基本信息和按平台、权限列表、缺陷级别维度统计的分布饼等信息 | 通过 |
| TC5-2-1 | 查看缺陷列表 | 点击进入查看缺陷列表页面 | 系统显示当前任务对应的缺陷列表信息，每一项包括状态、缺陷类型、缺陷级别、操作系统等信息 | 通过 |
| TC5-3-1 | 筛选缺陷列表 | 下拉选择某类操作系统、缺陷类别或缺陷级别 | 系统显示筛选后的缺陷列表 | 通过 |
| TC5-4-1 | 查看缺陷详情 | 点击列表中某个缺陷查看详情 | 系统显示该缺陷的详细内容，如日志、类别等缺陷基本信息和包括截图、操作序列以及当前状态等的缺陷上下文信息 | 通过 |
| TC5-5-1 | 导出测试脚本 | 点击下载测试脚本 | 系统将对对应操作序列的 Selenium 脚本下载到本机 | 通过 |
| TC5-6-1 | 下载测试报告 | 点击下载报告操作 | 系统将离线报告下载到本机，解压后查看，内容一致 | 通过 |

表 5.6是针对 UC4-UC6 功能需求描述的功能进行展开的测试用例设计，围绕测试报告相关的功能进行测试，包括了对应测试报告的查看和下载，缺陷列表和缺陷详情信息的查看、测试脚本导出等测试项。同时在进行测试前，也需

提前准备一个已执行完成的测试任务以供操作。最后全部通过的测试结果表明，该功能符合预期需求。

表 5.7: 维度评估相关测试用例

| ID | 测试说明 | 操作/输入 | 预期结果 | 测试结果 |
|---------|-----------|-------------------|----------------------------------|------|
| TC6-1-1 | 查看多维评估评分 | 测试人员点击选项卡进入多维评估页面 | 系统显示包括功能、性能、健壮性、兼容性和稳定性的多维评估雷达图 | 通过 |
| TC6-2-1 | 查看功能维度信息 | 点击功能维度进入功能评估详情 | 系统显示功能导航图，包括页面状态和操作事件流 | 通过 |
| TC6-3-1 | 查看性能维度信息 | 点击性能维度进入性能评估详情 | 系统显示请求性能提示列表，包括请求链接、请求时间和请求类型等信息 | 通过 |
| TC6-4-1 | 查看健壮性维度信息 | 点击健壮性维度进入健壮性评估详情 | 系统显示相应的缺陷列表 | 通过 |
| TC6-5-1 | 查看兼容性维度信息 | 点击兼容性维度进入兼容性评估详情 | 系统显示各环境下的测试路径图 | 通过 |
| TC6-6-1 | 查看稳定性维度信息 | 点击稳定性维度进入稳定性评估详情 | 系统显示稳定性页面列表 | 通过 |

表 5.7是针对 UC7 功能需求描述的功能进行展开的测试用例设计，主要用于对测试报告中维度评估相关功能进行测试，包括了对应维度报告以及各维度信息的查看等测试项，选用 TC5 中测试任务作为待操作对象。最后全部通过的测试结果表明，该功能符合预期需求。

5.2 实验分析

在验证了从创建 Web 应用自动化测试任务到生成测试报告的整个功能流程的正确性之后，我们仍需对产出测试报告的质量进行实验分析和评估，以探究由分析服务产出数据组成的该测试报告是否具有实际意义的参考价值，具体包括该测试报告是否能够有效发掘 Web 应用缺陷，发掘的 Web 应用缺陷是否真实存在，缺陷分类是否合理准确以及基于测试报告是否能够简单复现缺陷等问题。

5.2.1 实验对象

为了解答这些问题，我们选取了如图 5.1所示的 50 个涵盖了多种业务领域和技术类型的线上网站作为实验对象，来开展我们的系统评估实验。一方面，我们选择了来自计算机、教育、新闻、生活等多个常见领域的网页应用，来检验系统对不同的知识域应用的支持能力，另一方面，从网站开发技术的角度，在选择领域内应用时，按照单页、多页以及多页 + 动态加载这几种技术类型进行细分

挑选，以测试对包括 Vue、Angular、React 以及 LayUI 等在内的多前端框架的兼容支持。

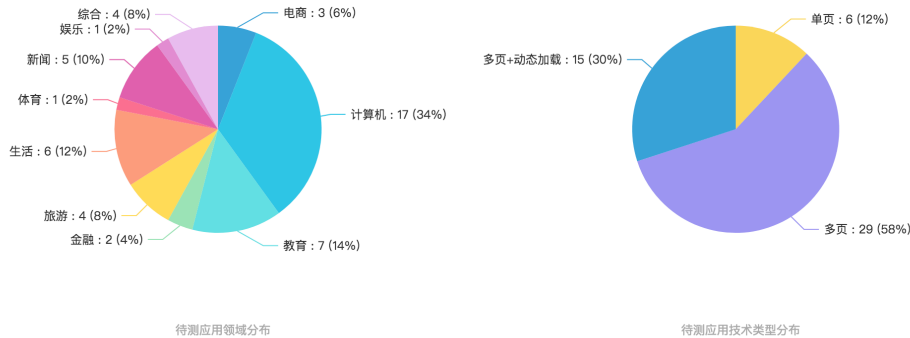


图 5.1: 待测应用集领域和技术类型分布图

其中，单页和多页分别代表不同的网站应用模式。单页应用是指在不需要重新加载页面的情况下，能够通过动态重写当前页面实现与用户的交互过程，是一种大前端体系，对应用前端开发要求较高，因此很多采用单页架构的待测应用都来自计算机领域，如这边单页类别的 6 个待测应用。多页应用则是传统的网站开发方案，即每一次页面跳转回都会返回新的 HTML 文档，需要频繁刷新页面，页面加载较慢，但 SEO 效果好，一些新闻和门户网站都比较偏好采用这种完全的多页模式。多页加动态加载是单页模式和多页模式的中间产物，一方面，基于多页模式降低开发难度，另一方面，利用 AJAX 动态加载避免页面的频繁刷新，很多电商、互联网企业网站目前都是这种技术模式。

5.2.2 实验设计

本文的实验步骤，主要包括两个步骤。首先第一步，主要是在系统里对上述实验对象进行自动化测试，并生成测试报告。类似于上文的功能测试流程，对于每个待测应用，我们都会依次进行相应测试任务的创建、配置、审批以及执行操作。同时，为了保证测试的一致性，避免其他因素的影响，我们在配置阶段，统一为所有的测试任务定义了最大测试时间为 3600 秒以及最大页面搜索为 3 的公共约束，以及包含 Chrome 和 Firefox 浏览器在内的相同测试环境配置。

第二步则是针对产出测试报告的人工审查和复盘操作，通过对照测试报告提供的缺陷列表和缺陷详情，统计测试任务相关的评估指标，包括测试是否成功执行、耗时、扫描缺陷数以及扫描状态数等，并在这些数据的基础上，通过人工抽样检查和实际执行测试脚本，统计缺陷的复现率和分类正确率、以及测试

脚本的执行成功率等。其中，缺陷复现率是指检测出的缺陷真实存在且能被人工复现的比率，以衡量系统缺陷发现的准确性；缺陷的分类准确率是指系统检测出的缺陷能被准确分类到对应缺陷类型的比率，以衡量系统缺陷判定的准确性；用例执行成功率，则是指系统生成测试脚本执行成功的比率，用于评估系统辅助缺陷定位的有效性。

5.2.3 实验结果

实验的分析结果可如图 5.2 和图 5.3 所示，分别对应实验步骤的第一步和第二步的实验结果。受限于篇幅限制，这边我们并未详细给出全部实验对象的测试结果，而是按照总体情况和技术类别，统计相关状态数和缺陷数的平均值，如图 5.2 所示。总体来看，在约束条件下，平均每个待测应用会搜索到 89 个页面状态，并发现 83 个应用缺陷。其中大部分缺陷为资源加载失败、请求超时等低风险缺陷，页面操作断链和服务端错误等较为严重的缺陷也偶有发生，但总体出现频率较低。同时通过观察图可以发现，随着网站应用模式从多页偏向单页，相应的网页状态数会逐渐减少，但缺陷数则会增加。我们猜想，单页模式下技术复杂度的提高，可能会带来了更多的缺陷引入。

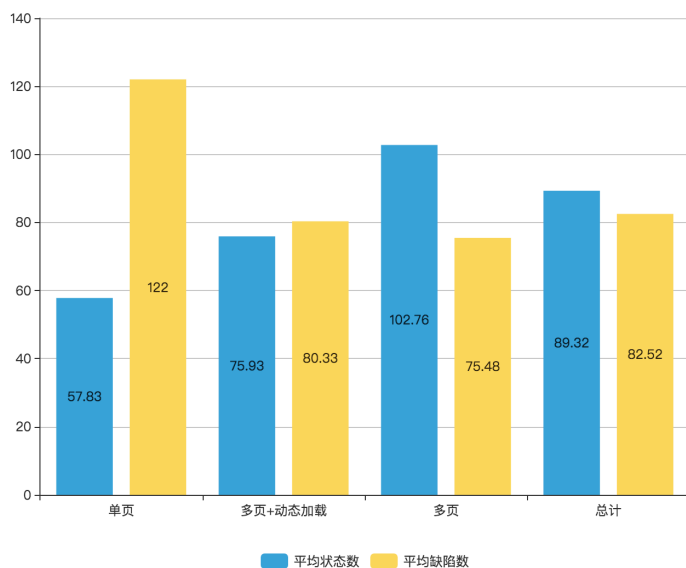


图 5.2: 实验对象自动化测试结果统计

图 5.3 则展示了第二步测试报告人工审查的统计结果。我们基于各个实验对象测试产生的缺陷列表，随机挑选了 632 个缺陷进行复现测试、分类审查以及测试脚本执行，最终的结果数据如图所示。在缺陷的可复现能力上，有 71.2% 的缺陷可以直接复现，18.2% 的缺陷则为概率复现，主要是资源加载失败和资源加

载时间过长等缺陷，推测可能与测试环境网络情况不稳定有关。还有 10.6% 的缺陷无法复现，主要集中在资源加载失败上，深度调研后，我们推测这可能与动态参数拼接的资源链接有关。在缺陷的准确分类情况上，97.47% 的缺陷都能被准确分类，2.53% 的未成功分类缺陷原因则来自知识库缺失的新缺陷的引入。在测试脚本的正确执行上，执行成功率为 99.84%，唯一失败的测试脚本问题来自 frame 窗口的未正常切换。综上结果表明，系统具有较好的缺陷发现和分类能力，且提供的测试脚本能有效辅助用户定位缺陷。

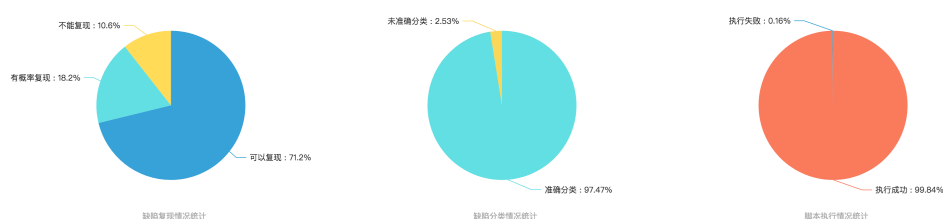


图 5.3: 测试报告人工审查结果统计

5.2.4 案例解读



图 5.4: 某订餐网站测试报告概况

图 5.4是上述实验其中一个实验对象的测试结果，展示了系统对某点餐应用的整体测试状况。系统在 20 分钟内一共搜索到该点餐应用的 90 个页面状态和 190 个缺陷数。其中，大部分缺陷问题集中在资源加载失败缺陷上，同时也具有资源加载时间过长、JS 引用错误、JS 类型错误以及页面或操作断链等缺陷。接下来，我们以该结果为例，解读系统测试报告在缺陷发现和定位方面的作用。



图 5.5: 页面操作断链缺陷详情页面

图 5.5所示为该测试任务的一个缺陷示例，展示了系统捕获页面或操作断链这类缺陷的示例情况。具体地，该缺陷展示了一次输入提交后系统出错并跳转至错误页面的场景，报告页面显示了该缺陷的具体详情和相关上下文信息，以及触发该缺陷的操作步骤和对应测试用例的下载地址。一方面，我们可以既通过页面截图和缺陷日志描述，直接获取该缺陷的表现，明确缺陷的产生过程，另一方面，我们也可以借由操作步骤提示和提供测试用例，快速定位到缺陷发生的实际场景，实现缺陷的定位，辅助用户修复缺陷。

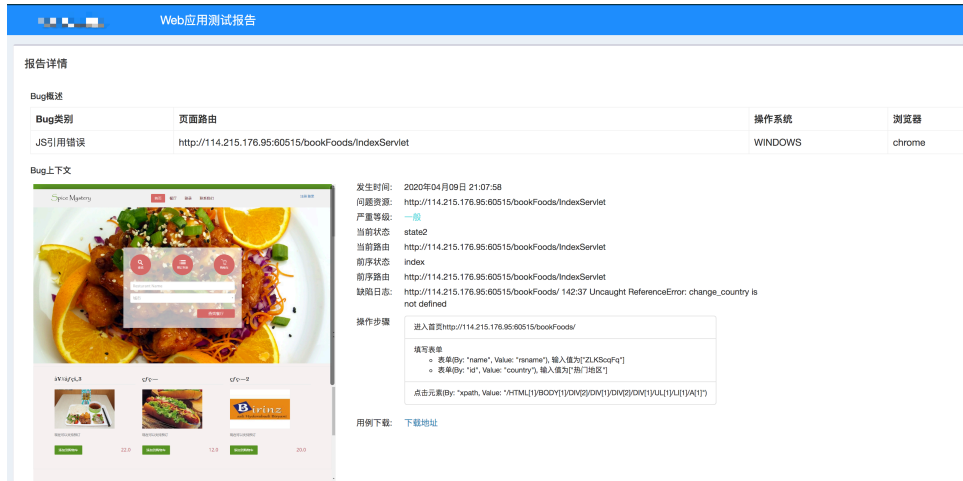


图 5.6: JS 引用错误缺陷详情页面

图 5.6是该测试任务的另一个缺陷示例，展示了一个点击选择城市操作触发的 JS 引用错误缺陷。通过查看缺陷详情和检查对应应用页面，我们发现，正常情况下，该操作会调用 `change-country` 方法，从而实现相应的监听功能，比如根

据筛选条件自动刷新餐厅数据。但在应用中该方法并未定义，所以会产生报错，应用也偏离了预期的功能设计。该缺陷详情预示了这一过程，有助于开发人员及时地修复该问题。

5.3 本章小结

本章主要对 Web 应用自动化测试系统进行功能测试和实验分析。其中，功能测试，主要基于第三章需求分析获取的功能需求和用例场景，构造了多组测试用例，验证系统是否满足用户需求。实验分析，则主要针对测试报告的可用性和有效性，进行实验设计和后续分析工作。实验表明，系统能够有效发掘页面操作断链、资源加载失败以及 JS 脚本错误等多种 Web 应用缺陷，同时具有较高的缺陷复现率、系统缺陷分类正确率以及导出测试脚本执行正确率，能够为用户检测待测 Web 应用相关缺陷，并提供指导缺陷定位的高可用测试报告。

第六章 总结和展望

6.1 总结

Web 应用的便捷性和广泛使用带来软件的高质量要求，迭代式敏捷开发则对 Web 测试提出了更高的要求。为解决传统手工测试的低效率和测试脚本的维护，丰富单一的测试结果，进行缺陷的有效定位和辅助复现缺陷，我们设计并实现了 Web 应用自动化测试系统，在精简输入的基础上，通过自动化遍历测试，完成缺陷的主动触发和分析报告。

本文阐述的分析服务是其中的核心服务之一，主要负责对自动化遍历测试下获取的过程产物，如控制台日志、请求日志，以及操作序列等数据，进行缺陷分析和上下文构建工作，帮助用户更好地识别、定位和修复缺陷。同时从性能、健壮性、稳定性等多个维度分析待测应用，帮助用户专项提升应用质量。本文的主要工作包括：

首先，调研了当前学术界和工业界针对 Web 应用自动化测试和日志缺陷检测相关的研究现状，并受到相关工作的启发，明确了现有实现手段的技术可行性。同时，介绍了项目设计和开发阶段中涉及到的技术框架、算法理论以及相应数学概念等等，奠定了后续的架构技术栈和实现思路基础。

其次，借助用例图、系统用例描述和建模方法，明确了系统的涉众、功能需求和非功能需求。同时通过架构设计、4+1 视图以及持久化模型等方法，阐述了系统架构和分析服务的概要设计过程，并通过模块划分，将服务拆解为 Web 应用缺陷分析框架、控制台日志缺陷聚类子模块、控制台日志缺陷判定子模块、请求日志缺陷检测模块以及多维评估模块，并通过架构设计、流程设计、数据与类设计，实现了核心的缺陷接入和缺陷检测功能。

最后对系统进行测试和实验分析，在系统的部署级环境上，完成了对系统的功能测试和基于 50 个真实网站的应用实验分析，验证了系统的缺陷发掘能力、缺陷分类正确性，同时评估了相应缺陷的可复现概率，验证了系统和产出测试报告的可靠性和有效性。

6.2 下一步展望

尽管当前我们的 Web 应用自动化测试系统已经部署完成，支持大多数网站的测试，并能发掘一定数量和类型的 Web 应用缺陷，但它仍然存在以下不足和可以改进的地方：

首先，现阶段采集的控制台缺陷日志数量较少，数据规模不大，因此采用基于密度的聚类算法，进行缺陷类别的挖掘发现。虽然聚类效果差强人意，但可能存在幸存者偏差，且并不能完整地概括可能的缺陷种类，具有漏报误报的风险。因此，随着后续数据集规模扩大和数据标记工作的开展，我们还可以借助神经网络，SVM 等有监督学习方式，完成缺陷的判断与分类，以及相关参数和阈值的动态设置。

其次，目前系统进行缺陷分析的数据来源主要是控制台日志和请求日志，后续可以引入更多的数据源，如后台实时日志、中间件日志以及应用源码等，实现缺陷上下文的信息补全，进一步完善 Web 应用缺陷体系，同时，也可以接入更多的缺陷检测手段，如图像识别，安全扫描，源码分析等，进一步发掘应用可能存在的缺陷，并助力于多维度的评估需要。

最后，系统目前的多维评估模块还处于迭代开发的基础阶段，是基于现有过程数据和中间记录，得到的参考性切面分析结果。随着后续数据类型和相关资料的引入和增加，在维度数量和维度深度分析方向上，还有很大的扩展和分析前景，比如借助项目现有页面截图功能，引入图像识别对比应用在不同测试环境下的页面差异，优化现有的兼容性维度评估方案，集成相关开源工具，在提供项目源码的基础上，通过静态分析和 CheckStyle 对照等手段，评估应用的可维护性维度等。

参考文献

- [1] G. A. D. Lucca, A. R. Fasolino, F. Faralli, U. de Carlini, Testing web applications, in: 18th International Conference on Software Maintenance (ICSM 2002), Maintaining Distributed Heterogeneous Systems, 3-6 October 2002, Montreal, Quebec, Canada, IEEE Computer Society, 2002, pp. 310–319.
URL <https://doi.org/10.1109/ICSM.2002.1167787>
- [2] G. Schermann, J. Cito, P. Leitner, H. C. Gall, Towards quality gates in continuous delivery and deployment, in: 24th IEEE International Conference on Program Comprehension, ICPC 2016, Austin, TX, USA, May 16-17, 2016, IEEE Computer Society, 2016, pp. 1–4.
URL <https://doi.org/10.1109/ICPC.2016.7503737>
- [3] 许蕾, 徐宝文, 陈振强, Web 测试综述, 计算机科学 (3) (2003) 100–104.
- [4] A. Bruns, A. Kornstädt, D. Wichmann, Web application tests with selenium, IEEE Software 26 (5) (2009) 88–91.
URL <https://doi.org/10.1109/MS.2009.144>
- [5] F. Wang, W. Du, A test automation framework based on WEB, in: H. Miao, R. Y. Lee, H. Zeng, J. Baik (Eds.), 2012 IEEE/ACIS 11th International Conference on Computer and Information Science, Shanghai, China, May 30 - June 1, 2012, IEEE Computer Society, 2012, pp. 683–687.
URL <https://doi.org/10.1109/ICIS.2012.21>
- [6] M. Benedikt, J. Freire, P. Godefroid, Veriweb: Automatically testing dynamic web sites, in: In Proceedings of 11th International World Wide Web Conference (WWW'02), Citeseer, 2002.
- [7] V. Dallmeier, B. Pohl, M. Burger, M. Mirolid, A. Zeller, Webmate: Web application test generation in the real world, in: Seventh IEEE International Conference on Software Testing, Verification and Validation, ICST 2014 Workshops Proceedings, March 31 - April 4, 2014, Cleveland, Ohio, USA, IEEE Computer Society, 2014, pp. 413–418.
URL <https://doi.org/10.1109/ICSTW.2014.65>

-
- [8] A. Mesbah, A. van Deursen, Invariant-based automatic testing of AJAX user interfaces, in: 31st International Conference on Software Engineering, ICSE 2009, May 16-24, 2009, Vancouver, Canada, Proceedings, IEEE, 2009, pp. 210–220.
URL <https://doi.org/10.1109/ICSE.2009.5070522>
- [9] E. van Eyk, W. van Leeuwen, M. A. Larson, F. Hermans, Performance of near-duplicate detection algorithms for crawljax (2014).
- [10] F. Ferrucci, F. Sarro, D. Ronca, S. Abrahao, A crawljax based approach to exploit traditional accessibility evaluation tools for ajax applications, in: Information technology and innovation trends in organizations, Springer, 2011, pp. 255–262.
- [11] M. Y. Chen, A. X. Zheng, J. Lloyd, M. I. Jordan, E. A. Brewer, Failure diagnosis using decision trees, in: 1st International Conference on Autonomic Computing (ICAC 2004), 17-19 May 2004, New York, NY, USA, IEEE Computer Society, 2004, pp. 36–43.
URL <http://doi.ieeecomputersociety.org/10.1109/ICAC.2004.31>
- [12] Y. Liang, Y. Zhang, H. Xiong, R. K. Sahoo, Failure prediction in IBM bluegene/l event logs, in: Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007), October 28-31, 2007, Omaha, Nebraska, USA, IEEE Computer Society, 2007, pp. 583–588.
URL <https://doi.org/10.1109/ICDM.2007.46>
- [13] M. Du, F. Li, G. Zheng, V. Srikumar, Deeplog: Anomaly detection and diagnosis from system logs through deep learning, in: B. M. Thuraisingham, D. Evans, T. Malkin, D. Xu (Eds.), Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017, ACM, 2017, pp. 1285–1298.
URL <https://doi.org/10.1145/3133956.3134015>
- [14] F. T. Liu, K. M. Ting, Z. Zhou, Isolation forest, in: Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy, IEEE Computer Society, 2008, pp. 413–422.
URL <https://doi.org/10.1109/ICDM.2008.17>

- [15] J. Lou, Q. Fu, S. Yang, Y. Xu, J. Li, Mining invariants from console logs for system problem detection, in: P. Barham, T. Roscoe (Eds.), 2010 USENIX Annual Technical Conference, Boston, MA, USA, June 23-25, 2010, USENIX Association, 2010.
- [16] Q. Lin, H. Zhang, J. Lou, Y. Zhang, X. Chen, Log clustering based problem identification for online service systems, in: L. K. Dillon, W. Visser, L. Williams (Eds.), Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14-22, 2016 - Companion Volume, ACM, 2016, pp. 102–111.
URL <https://doi.org/10.1145/2889160.2889232>
- [17] J. L. Carlson, Redis in action, Manning Publications Co., 2013.
- [18] W. Cao, S. Sahin, L. Liu, X. Bao, Evaluation and analysis of in-memory key-value systems, in: C. Pu, G. C. Fox, E. Damiani (Eds.), 2016 IEEE International Congress on Big Data, San Francisco, CA, USA, June 27 - July 2, 2016, IEEE Computer Society, 2016, pp. 26–33.
URL <https://doi.org/10.1109/BigDataCongress.2016.13>
- [19] A. Rahartomo, R. F. Aji, Y. Ruldeviyani, The application of big data using mongodb: Case study with scele fasilkom UI forum data, in: International Workshop on Big Data and Information Security, IWBIS 2016, Jakarta, Indonesia, October 18-19, 2016, IEEE, 2016, pp. 51–56.
URL <https://doi.org/10.1109/IWBIS.2016.7872889>
- [20] K. Banker, MongoDB in action, Manning Publications Co., 2011.
- [21] S. Vinoski, Advanced message queuing protocol, IEEE Internet Comput. 10 (6) (2006) 87–89.
URL <https://doi.org/10.1109/MIC.2006.116>
- [22] M. Rostanski, K. Grochla, A. Seman, Evaluation of highly available and fault-tolerant middleware clustered architectures using rabbitmq, in: M. Ganzha, L. A. Maciaszek, M. Paprzycki (Eds.), Proceedings of the 2014 Federated Conference on Computer Science and Information Systems, Warsaw, Poland, September 7-10, 2014, Vol. 2 of Annals of Computer Science and Information Systems, 2014,

- pp. 879–884.
URL <https://doi.org/10.15439/2014F48>
- [23] H. Suryotrisongko, D. P. Jayanto, A. Tjahyanto, Design and development of back-end application for public complaint systems using microservice spring boot, *Procedia Computer Science* 124 (2017) 736–743.
- [24] F. Gutierrez, *Pro Spring Boot*, Springer, 2016.
- [25] 孙吉贵, 刘杰, 赵连宇, 聚类算法研究, *软件学报* 19 (1) (2008) 48–61.
- [26] J. Han, M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann, 2000.
- [27] J. M. Peña, J. A. Lozano, P. Larrañaga, An empirical comparison of four initialization methods for the k-means algorithm, *Pattern Recognit. Lett.* 20 (10) (1999) 1027–1040.
URL [https://doi.org/10.1016/S0167-8655\(99\)00069-0](https://doi.org/10.1016/S0167-8655(99)00069-0)
- [28] S. Patel, S. Sihmar, A. Jatain, A study of hierarchical clustering algorithms, in: *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, IEEE, 2015, pp. 537–541.
- [29] H. Kriegel, P. Kröger, J. Sander, A. Zimek, *Density-based clustering*, Wiley Interdiscip. Rev. Data Min. Knowl. Discov. 1 (3) (2011) 231–240.
URL <https://doi.org/10.1002/widm.30>
- [30] 伍育红, 聚类算法综述, *计算机科学* 42 (6A) (2015) 491–499.
- [31] C. Fraley, A. E. Raftery, Enhanced model-based clustering, density estimation, and discriminant analysis software: MCLUST, *J. Classification* 20 (2) (2003) 263–286.
URL <https://doi.org/10.1007/s00357-003-0015-3>
- [32] M. Ester, H. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: E. Simoudis, J. Han, U. M. Fayyad (Eds.), *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, Portland, Oregon, USA, AAAI Press, 1996, pp. 226–231.
URL <http://www.aaai.org/Library/KDD/1996/kdd96-037.php>

- [33] V. I. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals, in: Soviet physics doklady, Vol. 10, 1966, pp. 707–710.
- [34] Y. Liu, Z. Li, H. Xiong, X. Gao, J. Wu, Understanding of internal clustering validation measures, in: G. I. Webb, B. Liu, C. Zhang, D. Gunopulos, X. Wu (Eds.), ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010, IEEE Computer Society, 2010, pp. 911–916.
URL <https://doi.org/10.1109/ICDM.2010.35>
- [35] A. Rosenberg, J. Hirschberg, V-measure: A conditional entropy-based external cluster evaluation measure, in: J. Eisner (Ed.), EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic, ACL, 2007, pp. 410–420.
URL <https://www.aclweb.org/anthology/D07-1043/>
- [36] P. J. Rousseeuw, A. Leroy, Robust Regression and Outlier Detection, Wiley Series in Probability and Statistics, Wiley, 1987.
URL <https://doi.org/10.1002/0471725382>
- [37] B. Nuseibeh, S. M. Easterbrook, Requirements engineering: a roadmap, in: A. Finkelstein (Ed.), 22nd International Conference on Software Engineering, Future of Software Engineering Track, ICSE 2000, Limerick Ireland, June 4-11, 2000, ACM, 2000, pp. 35–46.
URL <https://doi.org/10.1145/336512.336523>
- [38] A. K. Bharadwaj, T. R. G. Nair, Mapping general system characteristics to non-functional requirements, in: 2009 IEEE International Advance Computing Conference, 2009, pp. 1634–1638.
- [39] P. Kruchten, The 4+1 view model of architecture, IEEE Software 12 (6) (1995) 42–50.
URL <https://doi.org/10.1109/52.469759>
- [40] 孟晨, 赵春亮, 张建国, et al., 泛型 dao 模式在 java web 开发中的应用, 计算机应用与软件 (2012 年 01) (2012) 175–177+.
- [41] E. Gamma, Design patterns: elements of reusable object-oriented software, Pearson Education India, 1995.

- [42] 陈叶旺, 余金山, 泛型编程与设计模式, 计算机科学 (4) 257–261.
- [43] Q. Wu, Y. Wang, Performance testing and optimization of j2ee-based web applications, in: 2010 Second International Workshop on Education Technology and Computer Science, Vol. 2, IEEE, 2010, pp. 681–683.
- [44] S. R. Choudhary, M. R. Prasad, A. Orso, X-PERT: accurate identification of cross-browser issues in web applications, in: D. Notkin, B. H. C. Cheng, K. Pohl (Eds.), 35th International Conference on Software Engineering, ICSE '13, San Francisco, CA, USA, May 18-26, 2013, IEEE Computer Society, 2013, pp. 702–711.
URL <https://doi.org/10.1109/ICSE.2013.6606616>

简历与科研成果

基本情况 周赛，男，汉族，1997 年 1 月出生，湖南省汨罗市人。

教育背景

2018.9 ~ 2020.6 南京大学软件学院 硕士

2014.9 ~ 2018.6 南京大学软件学院 本科

参与项目

1. 国家自然科学基金项目：基于可理解信息融合的人机协同移动应用测试研究 (61802171) , 2019-2021
2. 中央高校基本科研业务费专项资金资助项目：基于群智协同的众包测试技术 (14380021), 2020-2020

致 谢

随着毕业论文进入尾声，转眼间自己的研究生生活也慢慢来到了终点。仿佛上次的告别还在眼前，这次就要正式和母校说再见了。这两年来，除了学习上的耕耘和成长，也经历了很多温暖的美好时光，收获了很多来自老师和同学们的鼓励和帮助。正好借助现在这个机会，让我表达心中的感谢之情。

首先我要感谢我的导师陈振宇老师，谢谢陈老师在学习上、技术上以及生活上等各方面给予我的支持，尤其感谢陈老师在设计阶段时的耐心指导和悉心帮助。从一开始立项时的方向确定，再到项目设计和实施阶段全程的进度跟踪和意见反馈，陈老师一直不辞辛苦地在帮助我们一步一步构建和完善项目。同时也特别感谢实验室的房春荣老师、黄勇老师以及徐剑锋老师，感谢他们一直以来对我们几个的关心和专业建议，我也从他们身上学习到很多。

其次我要感谢我的队友，尹子越和张晨剑同学。罗马不是一天建成的，我们共同经历了项目从无到有，从问题到解决方案的过程。虽然这期间，也面临过很多困难和挑战，但相互帮助、互相鼓励到最后战胜困难的过程真的很美好。

同时我还要感谢实验室的田元汉学长、袁阳阳、门铎、韩奇、徐文远、薛晓波、孙加辉、郭超以及其他有幸相遇的同学们，谢谢他们不厌其烦地被我请教，在毕业设计过程和生活中，都给予了我非常重要的帮助。然后我也要感谢亲爱的南京大学，让我度过了我最美好的这六年，遇到了很多优秀的老师和同学，完成了很多自己以前从来没有想过的尝试。未来，“励学敦行，诚朴雄伟”的校训将时刻叮嘱我奋勇向前，不惧艰险。

最后我要感谢我的父母和家人，在今年这个特殊时期，一直在背后默默支撑我，保护我，给了我向前跑的最大动力，我永远爱你们~同时谢谢参加论文评审和答辩的所有专家老师们，向你们的专业和付出致敬。

未来的征途是星辰大海，愿我们一起努力!

版权与原创性说明

任何收存和保管本论文的单位和个人，未经作者本人授权，不得将本论文转借他人并复印、抄录、拍照或以任何方式传播，否则，引起有碍作者著作权权益的问题，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：周赛
日期：2020 年 5 月 28 日

《学位论文出版授权书》

本人完全同意《中国优秀博硕士学位论文全文数据库出版章程》(以下简称“章程”),愿意将本人的学位论文提交“中国学术期刊(光盘版)电子杂志社”在《中国博士学位论文全文数据库》、《中国优秀硕士学位论文全文数据库》中全文发表。

《中国博士学位论文全文数据库》、《中国优秀硕士学位论文全文数据库》可以以电子、网络及其他数字媒体形式公开出版,并同意编入《中国知识资源总库》,在《中国博硕士学位论文评价数据库》中使用和在互联网上传播,同意按“章程”规定享受相关权益。

作者签名: 周赛

2020 年 5 月 28 日

| | | | | | |
|----------|--|------|------|------|------|
| 论文题名 | Web 应用自动化测试系统分析服务的设计与实现 | | | | |
| 研究生学号 | MF1832270 | 所在院系 | 软件学院 | 学位年度 | 2020 |
| 论文级别 | <input type="checkbox"/> 学术学位硕士 <input checked="" type="checkbox"/> 专业学位硕士 <input type="checkbox"/> 学术学位博士 <input type="checkbox"/> 专业学位博士 (请在方框内画钩) | | | | |
| 作者 Email | zhouss.wel@gmail.com | | | | |
| 导师姓名 | 陈振宇 | | | | |

论文涉密情况:

☒ 不保密

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