学士学位论文

软件测试用例在线评判系统

学　　号：　20131002723

姓　　名：　王肖辉

学 科 专 业：　软件工程

指 导 教 师：　张剑波　副教授

培 养 单 位：　信息工程学院

二○一七年六月

学士学位论文原创性声明

本人郑重声明：本人所呈交的学士学位论文《软件测试用例在线评判系统》，是本人在指导老师的指导下，在中国地质大学（武汉）攻读学士学位期间独立进行研究工作所取得的成果。论文中除已注明部分外不包含他人已发表或撰写过的研究成果，本人所呈交的学士学位论文没有违反学术道德和学术规范，没有侵权行为，并愿意承担由此而产生的法律责任和法律后果。

学位论文作者签名：

日　期：　　年　　月　　日

摘　要

在现今的软件测试教学中，依然偏重于理论的教学，实践教学的缺乏是影响学生软件测试能力的主要因素。在软件工程中，软件测试的地位越来越重要，设计软件测试用例的优劣，对学生以后项目开发至关重要。目前软件测试实践还限制于设备环境，要完成测试，学生需要安装针对不同语言和不同测试类型的测试工具，且对于测试的结果的评判也限制于人工评判，由于测试结果分散在各自的环境中也不利于汇总分析。所以本论文希望设计一款系统，该系统能够接收用户提交的测试用例脚本，然后由测评机对测试用例进行分析并给出评价结果。

目前有许多的在线测评系统，但是大部分是用来判断代码程序的正确性，及代码执行过程中的信息参数。目前市场上有许多优秀的软件测试框架和工具，但是需要安装环境，且测试结果也不便于持久保存。所以本系统拟结合两者来实现一款软件测试用例在线评判系统，参照目前流行的在线判题系统的原理，使用 Web 开发技术完成开发。利用合适的软件测试工具作为测评机的开发基础，将软件测试工具整合到测评机中，实现对软件测试用例的测评，目前对测试用例的测评以覆盖率为标准，二者通过 ApacheMQ 发送异步消息进行通信。

为了实现扩展的方便，本系统采用 Web 端和测评机相互分离的策略。测评机对测评实现进行了抽象封装，扩展只需要实现测评接口，配置对应的测试环境。

本文的主要工作分三个部分：

（1）确定需求，明确架构。根据目前软件测试教学中存在问题以及测试工具和测试方法的不足，确定系统的需求，系统的核心需求是实现在线测评测试用例；根据需求明确系统的架构，一方面，本系统要实现一个在线测评的功能，所以系统采用 Web 开发满足在线提交管理；另一方面测评是系统的核心，所以将其与 Web 端分离开发。因此系统大体架构类似 B/S 结构：B 为 Web 端，S 为测评机。

（2）系统设计与实现。详细设计系统各功能模块的结构及实现流程、用户操作界面、数据库以及主要程序模块的实现方式。

（3）系统测试。根据需求设计测试方案完成测试并对测试结果进行分析，验证系统的实现情况。

关键词：在线评判；测评机；覆盖率；消息通信；Web 开发

Abstract

In the teaching of software testing in today, teacher still place extra emphasis on the theory of teaching.For students , the ability to design test cases is not effectively exercised. Software testing, as an important subject area in software engineering, is critical in practical project development. At present,the practice of software testing is also limited to the device environment.If students want to complete the test, students need to install a variety of test tools. The test results are also limited with the evaluation of artificial.The test results and their respective environment is also not conducive to statistics. Therefore,this paper hopes to design a system and to achieve the test case online evaluation.

There are many online evaluation systems, but most of systems are used to judge the correctness of the code program and get the parameters in the time of code execution. There are many frameworks and tools for software testing, but the installation environment is required and the test results are not easy to keep lasting. Therefore, this system intends to combine OJ and Web development to achieve an online software test case judgments.With referenceing to principle of the current popular OJ system and the application of Web development technology to complete the development of the Web. The appropriate software testing tools integrated into the evaluation machine to achieve the evaluation of software test cases.The current evaluation of the task is mainly on the coverage of the statistics. Both send an asynchronous message via ApacheMQ for communication each other.

In this system, the Web and the evaluation machine are separated from each other, so the expansion is more convenient. The web almost is no need to modify, and only need to modify the evaluation machine.The evaluation machine abstracts and encapsulates the evaluation implementation.The expansion only need to achieve evaluation interface and configure the corresponding test environment.

The main work of this paper is divided into three parts.

Identifying requirements and clarifying the structure. Determining the needs of the system according to the current problems in software testing and the lack of test tools and test methods.In general, the core needs of the system is to achieve online test and test cases; To design the system architecture according to the needs of the system.On the one hand, the system need to achieve an online evaluation of the function, so the system

uses Web development to meet the needs of the submission and management online; the other hand, the evaluation is the core of the system, so it will be separated from the Web side development. So the system architecture is similar to the B / S structure: the B is the Web side and S is the evaluation machine.

In the aspect of the design and implementation, I designed the structure and implementation of each function module of the system, the user interface, the database and the realization of the main program module in detail.

At the test stage,in order to verify whether the system to meet the needs,I design the test program to test the system according to the needs.After finishing the test,I analyze the results of the test and determine the completion of the system.

Keywords: Online Evaluation;Evaluation Machine;Coverage;Message Communication; Web Exploit

目　录

图清单

表清单

第一章 绪论

1.1 研究背景和意义

软件测试是软件工程范畴中一个重要学科[1]，为了强化培养学生在软件开发领域的实践、设计以及创新能力，必须加强软件测试的实践教学。

目前，在软件测试教学中实践教学依然是薄弱的环节，其一般流程为教师布置实践内容，学生要完成测试需要在各自的计算机上安装相应的测试软件以及搭建测试环境，对于测试的结果老师也不方便统计与分析。一方面，需要为每一台机器配置一套测试框架，造成了资源的浪费；另一方面，测试的结果分散在各自的测试环境中，很难统一管理，不利于软件测试实践教学。

如今，在线考试或者测试模式被广泛应用于职业选拔和教学中[2-4]，相比于传统的测试，在线测试具有低成本、高效率等优势。考虑到软件测试的重要性和目前存在的问题，本文拟采用在线测试的模式，研究实现一款软件测试案例在线测评系统，在线评判将有效的提高软件测试实践教学效率，降低教学成本，同时也为学生提供了方便，有利于他们软件测试的实践学习。

1.2 国内外研究现状

软件测试作为软件开发过程中的一个重要环节，其目的是验证软件是否正确、功能是否完成、软件是否安全可用以及软件的质量是否合格，所以软件测试至关重要。目前，软件测试大多采用自动化测试或人工测试的方式，软件测试用例的在线测评尚未兴起。

在线判题系统（Online Judge，OJ）起初是用于国际的程序设计大赛，使用 OJ 来进行判题和排名。现在各地高校纷纷建立自己的 OJ 系统，用于程序设计的训练和比赛。现在比较有名气的 OJ 如：北京大学（Peking University Online Judge，POJ）等，该类系统能够编译并执行代码，并用预先准备好的数据去验证代码的正确性，及执行过程中的参数（内存占用等），并返回验证结果。但尚未有针对软件测试的在线测评系统。

从软件测试国内使用现状上来看，目前主要的测试工具用来进行单元测试、集成测试、回归测试和性能测试。对于单元测试和集成测试而言，目前市场上有许多的测试工具，如 Java 的 Junit 和 TestNG，C++的 Google Test 等，通过这些工具包，搭建测试执行环境，完成测试工作。回归测试是对修改代码的再次验证，热门的回归测试工具有 stingWhiz 和 Sahi 等。主流的性能测试工具如负载测试的 LoadRunner 和 Web 测试的 Jmeter，通过模拟实际用户的操作行为进行性能监控。要使用这些测试工具完成测试不仅需要耗费安装机器资源，也需要花费成本学习如何搭建测试环境，如何运用测试工具。上述这些工具虽然能满足软件测试的需求，但用在教学实验平台方面会受到机房单台机器的硬件资源限制，不能模拟复杂的测试环境。从软件测试发展的趋势上，近些年软件测试从单机模式向云测试模式转变[5-6]。云测试是一种基于云计算技术的新型测试方案，它将软件测试或者自动化软件测试工具作为一种服务提供给使用者。云测试平台可以提供多种浏览器作为用户交互的客户端，通过网络接收用户编写的测试脚本，在测试环境中运行用户的测试脚本进行测试，最后再通过网络将测试结果传输给用户。

1.3 研究目标和内容

软件测试教学环节中，测试用例的设计是学生对软件测试技术掌握程度的直观反映。长期以来，对测试用例的执行和评判都是由教师和学生根据案例特点人工评判，容易出现漏判、误判、逻辑覆盖率低的情况。所以本文主要研究目标为，开发一套软件测试在线测评系统，该系统以在线判题系统为基础，并结合软件测试工具，借鉴在线评测系统的工作原理，采用 JAVA WEB 开发技术，实现软件测试用例在线评判。该系统要求设计一种计算机所能接受的测试用例格式，制定评分标准，具备常规 Web 系统的用户管理、系统配置等基本功能；能够自动解析并执行学生上传的测试用例，计算测试数据的覆盖率，以此作为测试用例的评价标准[7]，实时统计并给出评判分数，从而实现计算机自动判别测试用例的功能。

本论文的主要研究内容是在线测评系统的工作原理[8-10]，测评机的测评机制及系统的扩展。具体来说，包含三个部分：测评机与测评系统通信，测评机的实现和系统的扩展。测评机和测评系统的通信使用消息总线服务 Apache ActiveMQ 来实现。测评机是系统的核心，它实现了对测评任务的接收、测试用例的评判的执行、测评结果的统计处理反馈。测评机的实现需要结合软件测试工具以及覆盖率统计工具。系统的扩展性包含两个方面，一方面是对测试语言支持的扩展，另一

方面是对测评机的扩展，当用户量过多时可以增加测评机的数量来减轻负载。图 1.1 展示了系统的大体工作流程，教师可以上传测试案例，学生可以在线编写测试用例并提交，测评机分析提交，反馈测试用例分析结果，教师和学生可以查看结果。

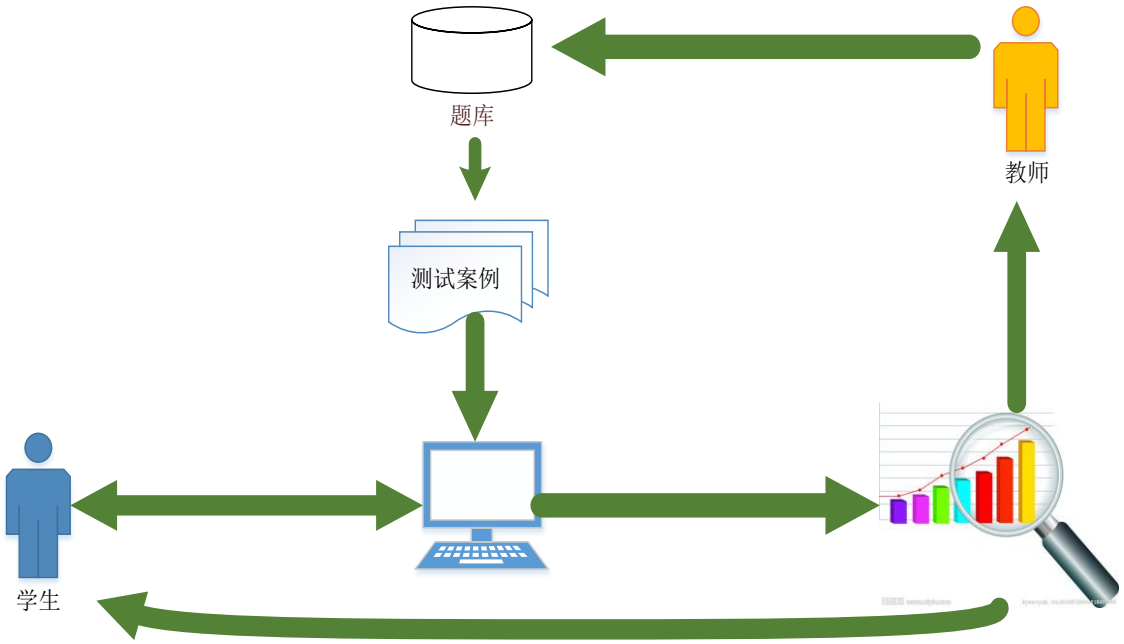

图 1.1 在线测评系统工作流程

1.4 论文的组织结构

本论文共分为六章，其组织结构与主要内容如下：

第一章 介绍本论文研究的主要目标、内容、现状和组织结构。

第二章 这一章介绍本系统实现过程中应用到的主要技术和框架，其中 Web 开发框架采用 SSM（Spring+SpingMVC+Mybatis），Web 端和测评机间的通信采用消息总线 Apache ActiveMQ，测试覆盖率统计使用 Jacoco（Java 覆盖率统计）和 nose（Python 覆盖率统计），测试代码的执行使用自动化工具 Ant，使用 jsoup 生成测试用例代码。

第三章 本章主要分析系统，明确需求，设计系统的架构包括逻辑架构涉及系统功能、物理架构涉及部署以及系统的功能结构。

第四章 本章主要是系统的详细设计，各功能模块的设计，主要模块包括：系统的管理模块、单元测试模块和系统维护模块，数据库设计，包含数据库 E-R 图及数据库表字典，操作界面设计、程序模块设计和安全性设计。

第五章 本章主要涉及系统的实现和系统的测试。介绍了系统实现的关键技术包含测试用例代码生成、Web 和测评机间的通信实现和测评机的实现，测试方面确定测试的环境、方案以及对测试结果的分析。

第六章 本章主要阐明本系统存在的一些不足，及改进设想，同时对本系统做一个小结。

第二章 技术支持

2.1 开发平台及框架

本系统以 Eclipse MARS.2 作为开发工具，以 Spring+SpringMVC+Mybatis 作为 Web 开发框架[11]。

Spring 是一个轻量级的 Java 开发框架，它起初主要是为了简化 Java EE 的企业级应用开发，与过去重量级的企业级应用开发相比，Spring 框架更加简单灵活。它所提供的面向切面编程和控制反转是 Spring 框架的核心功能。面向切面编程（Aspect Oriented Programming，AOP）实现了公共功能的模块化处理，提高了代码的可重用性。控制反转（Inversion of Control，IOC）是对象构建的一种策略，不直接进行对象的构建，而是通过注入依赖关系，框架通过依赖关系构建所需要的对象。

SpringMVC 作为一个请求驱动的 Web 框架，其处理控制器的实现策略，与其他的请求驱动的 Web 框架在总体思路上是相似的，通过分离流程控制逻辑与具体的 Web 请求处理逻辑。DispatcherServlet 它负责接收并处理所有的 Web 请求，只不过针对具体的处理逻辑，它会委托给它的下一级控制器去实现即 Controller。SpringMVC 处理请求流程如下图 2.1 所示：

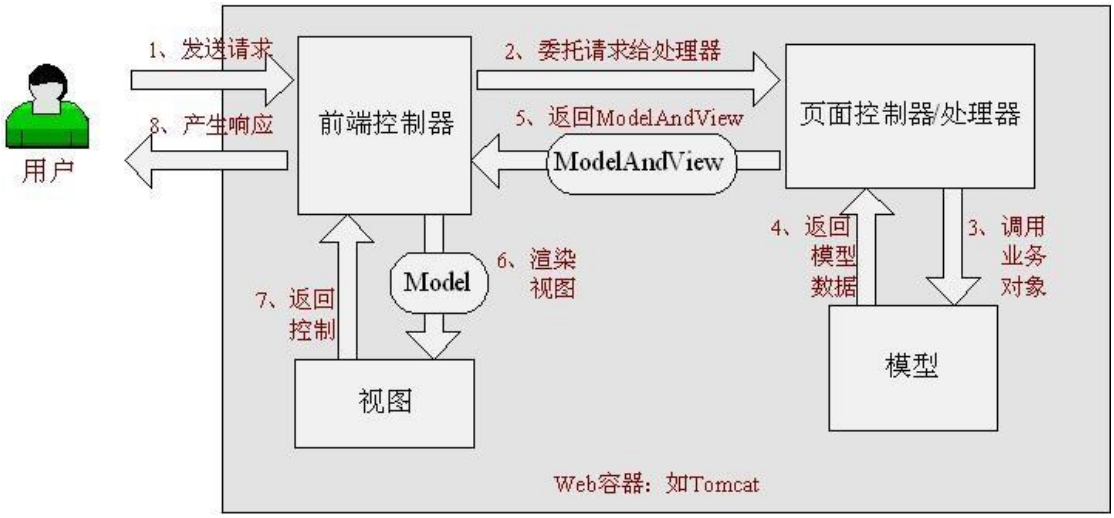

图 2.1 Spring Web MVC 处理请求流程

MyBatis 是一个持久化层框架，它隐藏了繁琐的配置和 JDBC 的代码，可以通过 xml 或注解来实现数据与 Mapper 接口之间的映射。

2.2 消息通信 ActiveMQ

ActiveMQ 是一种支持 JMS 规范的中间件。它提供了一种松散耦合的灵活的应用程序集成机制，程序彼此之间不需要直接通信，而是通过消息中间件来作为中介者进行通信。ActiveMQ 支持两种消息传递模型，点对点模型和发布/订阅模型。

点对点模型，此消息传递模型中，队列是目标。消息首先被传送至队列目标，然后根据队列传送策略，从该队列将消息传送至向此队列进行注册的某一个消费者，一次只传送一条消息。可以向队列目标发送消息的生产者的数量没有限制，也可以有多个消费者，但每条消息只能发送至一个消费者并由一个消费者成功使用。如果没有已经向队列目标注册的消费者，队列将保留它收到的消息，并在某个消费者向该队列进行注册时将消息传送给该消费者。

发布/订阅模型，此消息传递模型中，主题是目标。消息首先被传送至主题目标，然后传送至所有已订阅此主题的活动消费者。可以向主题目标发送消息的生产者的数量没有限制，并且每个消息可以发送至任意数量的订阅消费者。主题目标也支持持久订阅的概念。持久订阅表示消费者已向主题目标进行注册，但在消息传送时此消费者可以处于非活动状态。当此消费者再次处于活动状态时，它将接收此信息。如果没有已经向主题目标注册的消费者，主题不保留其接收到的消息，除非有非活动消费者注册了持久订阅。

本系统应用 Spring 集成的 ActiveMQ，使用 Spring JMS 实现异步发收消息。考虑到在线测评系统测评任务只需要有一个测评机来处理的需求，以及 Active 消息模型的特点，本系统采用 ActiveMQ 的点对点消息传递模型，其涉及的主要 API 如下表 2.1 所示：

表 2.1 spring 集成 ActiveMQ 主要 API

org.apache.activemq.ActiveMQConnectionFactory
org.springframework.jms.connection.CachingConnectionFactory
org.apache.activemq.command.ActiveMQQueue
org.springframework.jms.core.JmsTemplate
org.springframework.jms.listener.SimpleMessageListenerContainer

(1) ActiveMQConnectionFactory：此类以 broker URL 为构造参数，使用工厂模

式创建连接工厂（ConnectionFactory）对象，调用 ConnectionFactory 的 createConnection 方法即可创建连接。

(2) CachingConnectionFactory：Spring 中发送消息的核心是 JmsTemplate，然而 Jmstemplate 的问题是在每次调用时都要打开/关闭 session 和 producter，效率很低，所以自 spring2.5.3 版本后，Spring 提供了 CachingConnectionFactory，此类扩自 SingleConnectionFactory，主要用于提供缓存 JMS 资源功能。具体包括 MessageProducer、MessageConsumer 和 Session 的缓存功能。

(3) ActiveMQQueue：消息队列，ActiveMQ 消息传递模型的一种实现方式，接收来自生产者发送的消息到消息队列中，一次只能一个消费这接收消息，且接收后此消息将会从消息队列中删除，即使没有消费者消息也不会丢失。

(4) JmsTemplate：此类封装了消息连接、关闭等一些繁琐的操作，只需要传递一些属性如连接工厂 connectionFactory、消息目的地 Destination 等，就可以获取会话并进行收发消息，这样开发者就可以只关注收发消息的开发工作。

(5) SimpleMessageListenerContainer：为了接收 JMS 消息，Spring 提供消息监听容器的概念，这些容器可以绑定去接受到达某一目的地的消息，不同的容器其配置参数信息略有不同，SimpleMessageListenerContainer 需要三个参数，连接工厂、目的地和自定义消息接收者（实现了 javax.jms.MessageListener）。

消息流程图如下图 2.2 所示，JMS 发送消息到 ActiveMQ Broker 的消息队列，消费者向 Broker 提交注册，Broker 会按照一定的策略将队列中的消息分发给注册的消费者，消费者监听到有消息到达时，就会接收消息并进行处理。

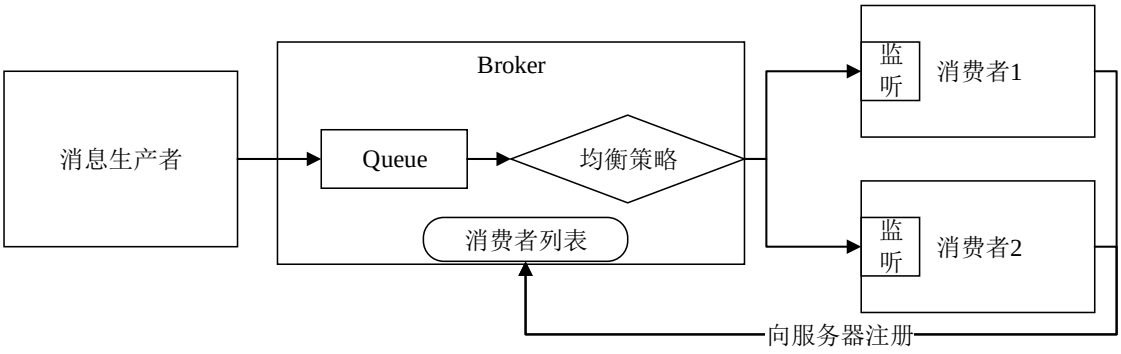

图 2.2 JMS 消息流程图

2.3 覆盖率统计

2.3.1 Java 覆盖率统计

目前市场上对 Java 覆盖率统计工具的一般流程如下图 2.3 所示。

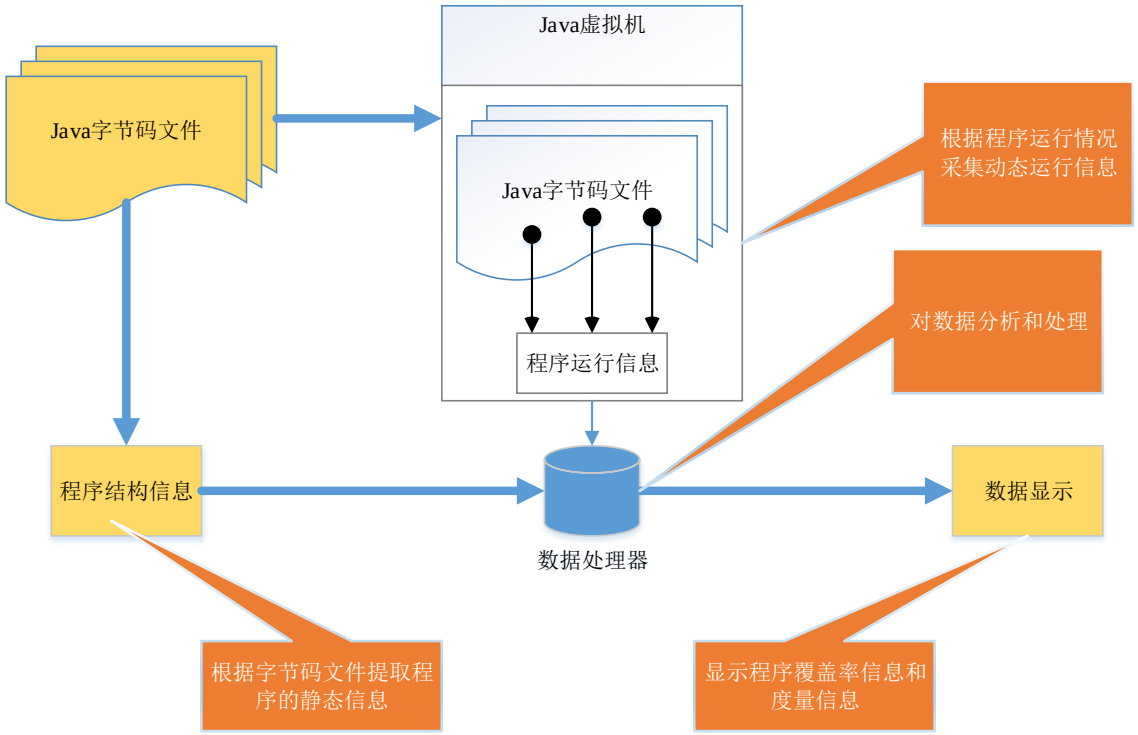

图 2.3 覆盖率统计工具流程

(1) 对 Java 字节码进行插桩，On-The-Fly 和 Offine 两种方式；

(2) 执行测试用例，收集程序执行轨迹信息，将其 dump 到内存；

(3) 数据处理器结合程序执行轨迹信息和代码结构信息分析生成代码覆盖率报告；

(4) 将代码覆盖率报告图形化展示出来，如 html、xml 等文件格式。

目前 Java 常用覆盖率工具 Jacoco、Emma 和 Cobertura，其各自的特点如下表 2.2 所示：

表 2.2 Java 覆盖率工具对比

工具	Jacoco	Emma	Cobertura
原理	使用ASM修改字节码	可以修改 Jar 文件、class 文件字节码文件	基于 Jcoverage。基于 ASM 框架对 class 插桩
覆盖粒度	方法、类、行、分支、指令、圈	行、块、方法、类	行、分支
插桩	on-the-fly 和 offline	on-the-fly 和 offline	offline
缺点		不支持 JDK8	关闭服务器才能获取覆盖率报告

性能	快	较快	较快

通过对比三种覆盖率工具，Jacoco[12]不仅仅支持的粒度多，而且速记快，所以本系统对 java 语言覆盖率统计使用 Jacoco。它的使用方式比较灵活，既可以通过 Eclipse 插件的形式使用，也可以通过嵌入到 maven 或 ant 中使用，本系统是嵌入到 ant 中来执行的。目前 Jacoco 支持不同标准的覆盖率统计,其中有常用的分支覆盖，还有指令级别的覆盖、行覆盖、方法类覆盖。

(1) 指令覆盖指单个 java 二进制代码指令是否执行，是 Jacoco 最小计数单位。

(2) 分支覆盖确定一个方法里面执行和不执行的分支数量。

(3) 行覆盖指示一行代码是否被执行。

(4) 方法覆盖是指一个方法中是否至少有一条指令被执行。

(5) 类覆盖即一个类中是否至少有一个方法被执行

Jacoco 支持两种插桩模式 On-The-Fly 和 Offline。

(1) On-The-Fly：JVM 中通过-javaagent 参数指定特定的 jar 文件启动 Instrumentation 的代理程序，代理程序在每装载一个 class 文件前判断是否已经转换修改了该文件，如果没有则需要将探针插入 class 文件中，代码覆盖率就可以在 JVM 执行代码的时候实时获取。

(2) Offline：在测试前先对文件进行插桩，然后生成插过桩的 class 或 jar 包，测试插过桩的 class 和 jar 包后，会生成动态覆盖信息到文件，最后统一对覆盖信息进行处理，并生成报告。

On-The-Fly 模式优点在于无需修改源代码，可以在系统不停机的情况下，实时收集代码覆盖率信息。Offine 模式优点在于系统启动不需要额外开启代理，但是只能在系统停机的情况下才能获取代码覆盖率。

Jacoco 使用 On-The-Fly 插桩模式执行流程为：Jacoco 作为 Java 代理运行，它负责在运行测试时对字节码进行检测。在执行测试时，Jacoco 会渗透到每一行指令并且显示哪些行被执行。Jacoco 接收来自 JVM 工具接口中的事件，使用 ASM（一个通用的 Java 字节码操作和分析框架）进行代码检测，收集覆盖率数据。流程如下图 2.4 所示：

图 2.4 Jacoco 覆盖率统计

使用 Jacoco 统计 Java 单元测试覆盖率报告如下图 2.5 所示：

MathUtil

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Methods
MathUtil()		0%		n/a	1	1	1	1
max(int, int, int)		100%		100%	0	4	0	1
Total	3 of 20	85%	0 of 6	100%	1	5	1	2

图 2.5 Jacoco 覆盖率统计报告

2.3.2 Python 覆盖率统计

Nose 作为一个知名的 Python 单元测试框架[13]，优于 UnitTest 。Nose 可以自动识别继承于 UnitTest.TestCase 的单元测试，并且也可以测试非继承于 UnitTest.TestCase 的单元测试。Nose 为编写单元测试提供了全面的 API。Nose 也可以通过内置插件实现代码覆盖、文档测试、错误查找等等，同时也可以自定义开发插件来满足测试需求。

Nose 可以通过—with-coverage 选项实现单元测试覆盖率的统计，而实际上是调用了 Python 的 coverage.py 模块。覆盖数据的采集由 coverage/collector.py 中的 Collector 类和 PyTracer（CTracer）类合作完成的。其核心是利用了 Python 虚拟机的 trace 机制，trace 是 Python 程序调试机制的核心，可以干预其上执行的程序。Trace 通常会启动各种钩子（Hook），会严重拖慢被 trace 的程序的运行速度，所以 coverage.py 模块在进行覆盖数据采集的实现上提供了一个 C 语言版本的实现 Ctracer 类，有效提高执行效率。

本系统对于 Python 语言的测试用例覆盖率统计使用 nose。使用 nose 的 cover 插件统计 Python 单元测试覆盖率结果如下图 2.6 所示：

Coverage report: 60%

Module	statements	missing	excluded	branches	partial	coverage
MaxMath.py	9	3	0	6	1	60%
Total	**9**	**3**	**0**	**6**	**1**	**60%**

coverage.py v4.4b1, created at 2017-05-07 09:35

图 2.6 nose-coverage 覆盖率统计结果

2.4 其他技术

2.4.1 Ant

Ant 作为一个跨平台的构件工具，它能够完成工程的自动化构建和部署等功能。Ant 的构件文件使用 XML 格式。构件文件有两个主要元素，project 是构件文件的基本元素；基本任务执行单元写在 target 中。在本系统中 Java 测试的编译、执行都是通过构建文件来执行，Java 测试覆盖率统计 Jacoco 也是通过集成到构建文件来执行的。

2.4.2 Jsoup

Jsoup 作为一个 Java 版本的 Html 解析工具包[14]，可以以 URL、文件以及字符串作为输入解析出 DOM。可以通过 DOM 或选择器进行标签或元素的操作。本系统中对于 HTML 形式覆盖率结果的解析由 Jsoup 来完成，使用 Jsoup 将覆盖率结果以 List 对象的形式存储。

2.4.3 Beetl

Beetl 作为新一代 Java 模板引擎的典范[15]，相对于其他 Java 模板引擎，具有功能齐全，语法直观,性能超高，以及编写的模板容易维护等特点。使得开发和维护模板有很好的体验。是新一代的模板引擎。总得来说，它的特性如下：

1、内置方法完善：Beetl 提供日期、json 以及数字类型转化等常用的内置方法，还提供了字符串、数组、正则表达式以及 Spring 等相关的函数。

2、功能完备：作为主流模板引擎，Beetl 具有相当多的功能和其他模板引擎不具备的功能。适用于各种应用场景，从对响应速度有很高要求的大网站到功能繁

多的 CMS 管理系统都适合。Beetl 本身还具有很多独特功能来完成模板编写和维护，这是其他模板引擎所不具有的。

3、非常简单：类似 Javascript 语法和习俗，只要半小时就能通过半学半猜完全掌握用法。拒绝其他模板引擎那种非人性化的语法和习俗。同时也能支持 html 标签，使得开发 CMS 系统比较容易。

4、超高的性能：Beetl 远超过主流 Java 模板引擎性能(引擎性能 5-6 倍与 freemaker，2 倍于 JSP。参考附录)，而且消耗较低的 CPU。

5、易于整合：Beetl 能很容易的与各种 web 框架整合，如 Spring MVC，Jfinal、Struts、Nutz、Jodd、Servlet 等。

6、支持模板单独开发和测试，即在 MVC 架构中，即使没有 M 和 C 部分，也能开发和测试模板。

7、扩展和个性化：Beetl 支持自定义方法，格式化函数，虚拟属性，标签，和 HTML 标签。 同时 Beetl 也支持自定义占位符和控制语句起始符号也支持使用者完全可以打造适合自己的工具包。

2.5 本章小结

这一章节主要介绍系统开发过程中所应用到的主要技术，包含 Web 端技术、覆盖率工具、通信技术以及其他相关工具。Web 方面介绍了 Web 开发框架 Spring、SpringMVC 以及 Mybatis，展示了 Web 请求到界面渲染的基本流程。覆盖率工具方面介绍了 Java 和 Python 的覆盖工具，Java 常用覆盖率工具及其覆盖率统计的流程，并具体介绍了 Jacoco 的覆盖粒度、覆盖统计的原理以及两种插桩模式；Python 的覆盖率统计工具 nose 的使用以及 nose 收集覆盖数据的原理。消息通讯方面详细介绍了 ActiveMQ 以及其两种消息传递模式，Spring 集成 ActiveMQ 通过 JSM 收发消息的主要 API 以及其流程。此外还介绍了一些辅助工具构件工具 ant、html 解析工具以及代码生成工 Beetle。

第三章 系统需求分析和结构设计

3.1 需求分析

3.1.1 系统目标

软件测试教学环节中，测试用例的设计是学生对软件测试技术掌握程度的直观反映。长期以来，对测试用例的执行和评判都是由教师和学生根据案例特点人工评判，容易出现漏判、误判、覆盖率低的情况。如果有一款能借助计算机实现测试用例自动评判的软件系统，将能显著改善教学效果。鉴于此，本系统将借鉴在线测评系统的工作原理，实现一款软件测试用例在线评判系统。此系统能够接收来自 Web 端学生提交的测试用例脚本，然后结合测试案例对测试用例进行覆盖率分析，对于覆盖率分析结果能够进行二次处理并且能够反馈给学生。此外作为 Web 系统还应当拥有一般系统的功能如能够登录注册、系统的一些设置及管理等等。

3.1.2 功能需求

通过对系统目标的分析，系统的功能性需求主要有以下几点：

(1) 单元测试。学生可以根据条件筛选测试案例，提交测试用例，查看提交记录，查看所有提交记录，修改个人资料等。获取学生提交的测试用例，生成测试用例代码。执行测试生成测试覆盖率报告，解析测试报告，持久化测试结果信息。向用户反馈测试消息及结果。

(2) 系统管理。用户信息管理，涉及到用户的登录及注册，用户的增删改查，用户权限的管理。系统本身的一些属性的管理，如系统的版本，是否允许注册，支持语言等等。

(3) 系统维护。本部分功能涉及测试相关的元素的维护，包含测试案例、用户提交的添加、删除、修改等操作。还有对不同语言测试工具的管理，对测评机的维护等。

3.1.3 系统非功能性需求

(1) 高可用性

在系统运行过程中可能出现一些特殊情况如服务器宕机、断电等，导致测评机无法正常工作，这将影响测评任务的正常进行，为了保证系统测评能够正常进行，系统应当确保当某一个测评机挂掉以后，能够由备用的测评机去完成测评任务。所以系统提供多个测评机，当某一个测评机停止工作后，其他测评机任务依然能够由其他测评机来完成，且对于多个测评机任务分发是均衡的即消费者负载均衡。

(2) 可扩展性

本系统是针对单元测试用例的在线评判，而不应该仅仅是针对某一种语言的单元测试用例的在线评判，所以应当对语言有较高的扩展性，扩展应当尽量少了改动系统原代码，而是以模块的形式添加新语言的扩展及移除。

3.2 系统结构设计

3.2.1 逻辑架构

从逻辑结构上来看分为四个层次：

(1) 用户展现层

本层主要为用户提供操作界面，用户目前有三种：一为学生，学生视图提供学生操作的界面，包括案例的选择页面，测试用例的提交页面，提交的查看页面等等；二为教师，教师视图提供了教师操作的界面，教师所提交的测试案例的管理，优秀测试用例的查看等；三为管理员，管理员视图提供管理员的操作界面，包括学生所拥有操作页面，测试案例管理页面，用户的管理页面，提交用例管理页面，系统的基本设置页面和测试工具配置页面等等。

(2) 控制层

这一层次主要实现了用户请求的分发，控制层接收来自展现层的用户请求，针对不同请求，DispatcherServlet 会寻找匹配@RequestMapping 值的 Controller 去处理对应的请求。

(3) 服务层

本层次主要为控制层提供服务。服务主要分三大模块：系统管理模块，本模块主要包含用户登录注册功能、用户的管理（添加、删除、修改、查询）和用户

权限管理等等；系统维护模块，本模块主要包含测试案例的管理、用例提交的管理、测试工具管理和系统设置等等；单元测试模块，本模块主要包含测试用例提交请求处理，测试结果反馈等。

(4) 数据层

数据层提供对系统相关数据的持久化存储和访问，主要包含三个方面的数据：测试案例数据，存储教师提交的测试案例，包含了测试案例相关的属性；用例提交，存储学生提交的测试用例信息，包含了测试案例的信息、用例、测评结果等相关信息。用户数据，存储学生和教师用户相关信息。

(5) 测评机

测评机为执行学生提交测试用例的核心，当测评机的消息接受收到 web 端传来的测试任务创建消息后，创建测试任务并执行，为避免同步线程安全问题，此处采用 synchronized 来解决，保证一次只执行一个测评任务。待任务执行完毕后，使用解析工具将执行结果解析成 Java 的 Map 对象并将结果持久化到用例提交数据库中，然后发送测试任务执行完毕的消息给 Web 端。同时对提交的测试用例做筛选排序处理，将优秀的测试用例持久化到数据库。系统的整体逻辑架构如图 3.1 所示：

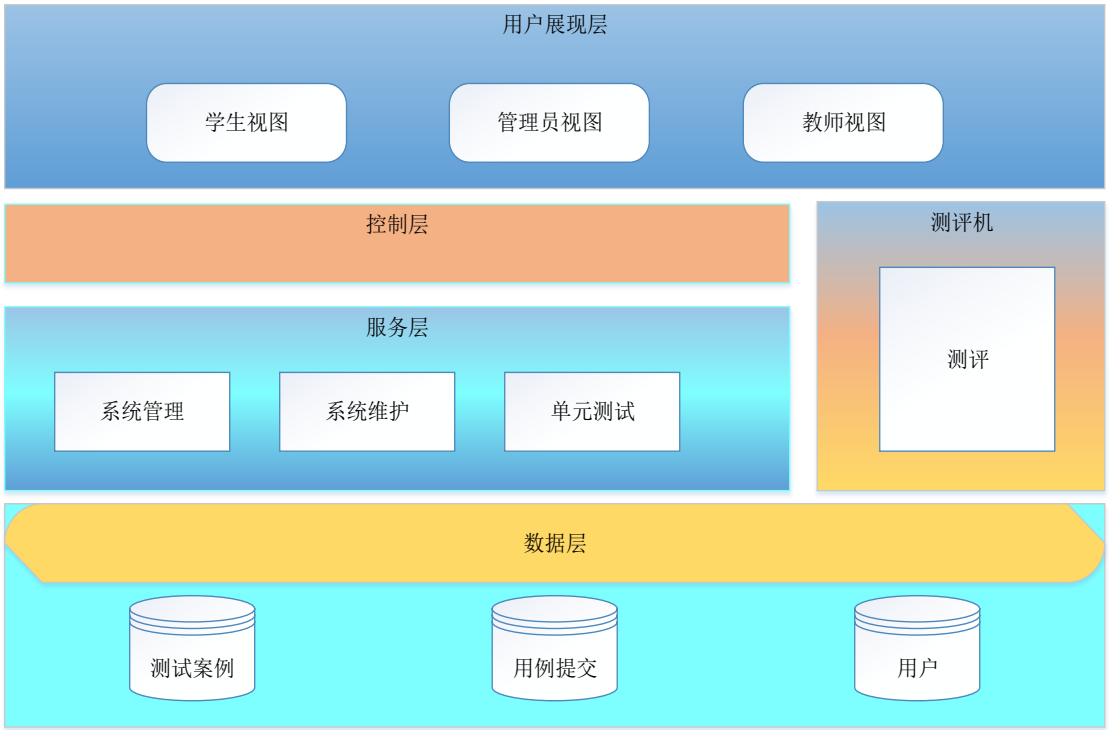

图 3.1 系统逻辑架构图

3.2.2 物理架构

本系统分两部分，一为 Web，此部分提供了用户操作界面的构建以及用户提交请求的处理，是整个项目的前台；二为测评机，此是测评系统的核心部分，负责完成测评任务的处理及反馈工作。部署上二者相互独立，通过 ActiveMQ 来进行通信，二者共同完成测评工作。为了提高系统的可用性，在部署上采用多个测评机，针对 Web 端的测评任务请求，ActiveMQ 会均衡的分配给多个测评任务消费者即测评机。其物理架构图如下图 3.2 所示：

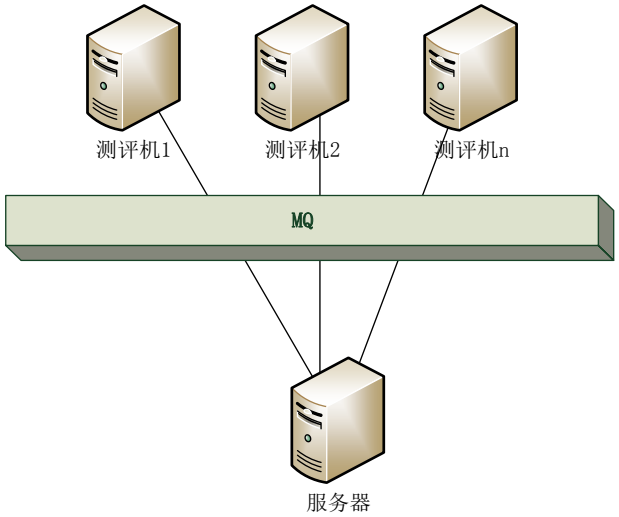

图 3.2 系统物理架构图

3.2.3 功能结构

大体介绍系统的功能结构。系统管理主要是用户管理和系统设置，单元测试是系统的主要功能，实现测试用例的测评分析，系统维护是测试相关数据的管理。系统功能结构如下图 3.3 所示：

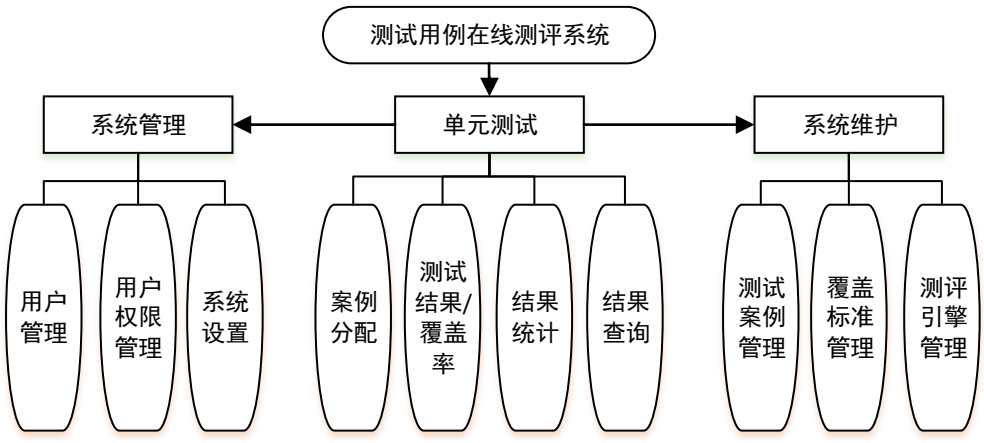

图 3.3 系统功能结构

3.3 本章小结

本章主要介绍了系统的需求和设计。确定了实现在线测评测试用例的核心目标，也由此明确了系统的需求；从逻辑视图、物理视图角度分析设计系统的架构，功能结构图展示了系统设计的功能目标。总的来说这一部分是从需求和设计角度来为系统的开发提供全方位的指导。

第四章 系统详细设计

4.1 系统功能模块设计

4.1.1 系统管理模块

1、模块结构

图 4.1 系统管理模块结构

2、模块类图

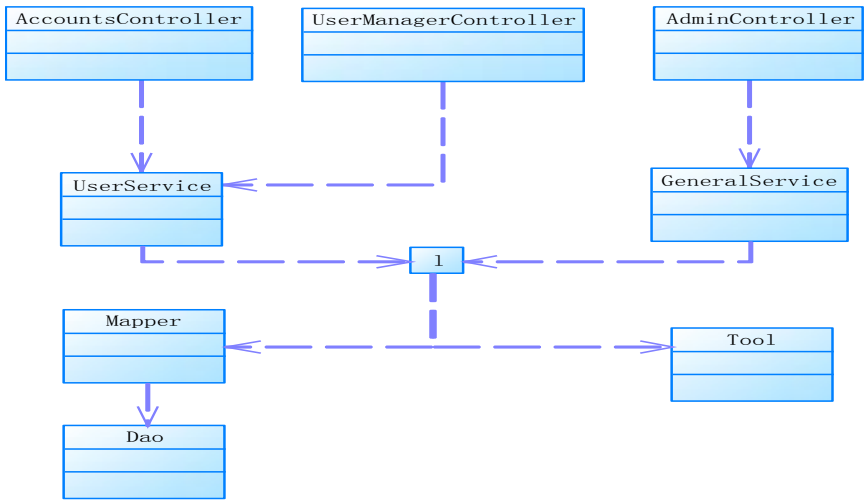

图 4.2 系统管理模块类图

AccountsController 控制器：用来处理用户的登录、注册以及修改个人资料等请求。

UserManagerController 控制器：用来处理用户管理的请求，用户的添加、删除、修改、更新。

AdminController 控制器：用来处理系统设置的请求，支持语言的修改，系统是否允许注册等。

UserService：提供用户相关的数据操作服务。

GeneralService：提供系统一般设置的数据服务。

Mapper、Dao、Tool：为 Service 提供调用接口以及辅助工具。

3、模块功能

主要实现用户信息管理和系统的设置。

(1) 用户的注册、登录

用户输入验证信息，通过验证则注册或登录成功，未通过则显示错误信息，用户重新操作。其流程图如下图 4.3 所示：

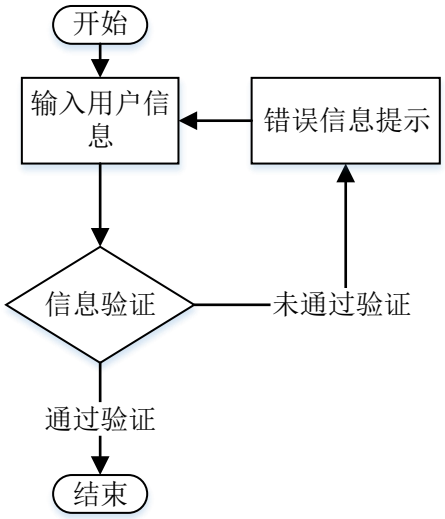

图 4.3 用户登录、注册流程图

(2) 用户管理

实现对用户信息的添加、删除、修改、查询操作，同时也实现了操作过程中的参数校验。

(3) 用户权限管理

根据角色的不同将请求提交到不同的控制器，由控制器显示不同的操作界面或是返回不同用户角色属性到页面上。对于测试提交有哪些人能够查看设计的测试用例实现通过角色属性进行判断，只有教师、管理员和提交者本身能够看到。

(4) 系统设置

系统的版本信息设置；系统注册功能的开关设置，在某些特殊时期，管理员可以关闭系统的注册功能，禁止用户注册；管理支持的语言等等。

4.1.2 系统维护模块

1、模块结构

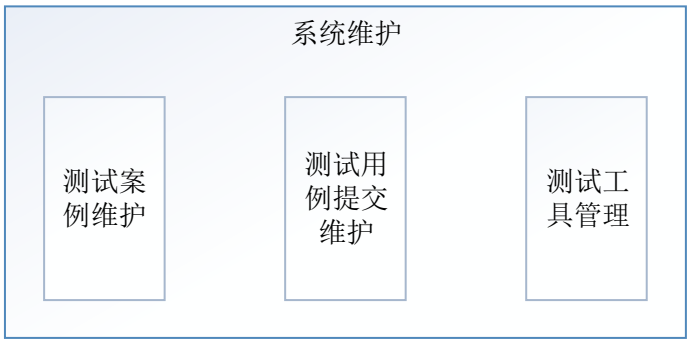

图 4.4 系统维护模块结构图

2、模块类图

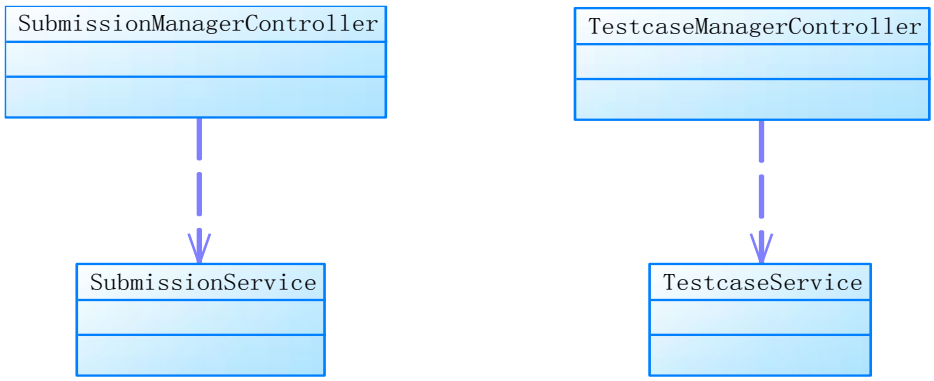

图 4.5 系统维护模块类图

SubmissionManagerController 控制器：测试用例提交管理请求处理器。

TestcaseManagerController 控制器：测试案例管理请求处理器。

SubmissionService：提供测试用例提交数据管理服务。

TestcaseService：提供测试案例数据管理服务。

3、模块功能

本模块主要实现测试用例测评相关数据的维护工作，包含测试案例的维护、测试用例提交的维护和覆盖标准的维护。

4.1.3 单元测试模块

1、 模块结构

图 4.6 单元测试模块结构图

2、 类图

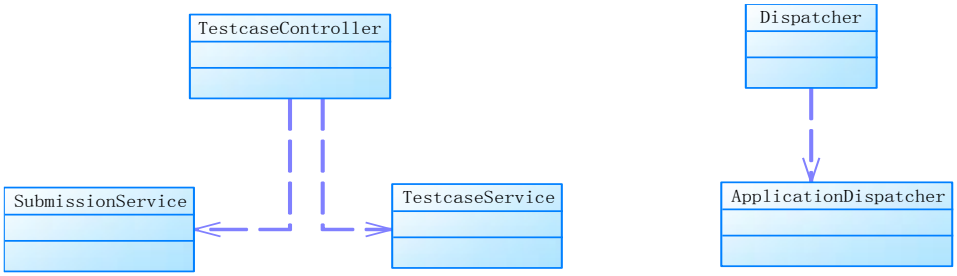

图 4.7 单元测试模块类图

TestcaseController 控制器：接收用户的测试用例提交请求。

SubmissionService：提供提交信息服务。

TestcaseService：提供测试案例信息服务。

Dispatcher: 测评调度器，完成测评任务的创建与执行。

ApplicationDispatcher:消息处理器，处理测评过程中消息,如测试相关文件创建完成等。

3、 模块功能

单元测试模块主要负责测评的工作，从用户选择测试案例，提交测试用例，交由测评机进行测试、覆盖率统计、及测试结果处理（优秀测试用例的筛选，结果消息的反馈），用户查询所有测试用例提交信息等等。

4.2 用户界面设计

4.2.1 设计原则

一个成功的应用系统，不仅拥有健全的功能还要有良好的操作体验。系统最终的使用者是用户，所以一个好的系统还应当有一个友好的操作界面。在设计时应当遵循一定的界面设计原则：

1、用户原则：界面设计首先要确定用户类型，针对用户的特点来设计。例如节日营销网站，页面应当是色彩斑斓的，而一个信息管理系统，应当简洁朴素的。

2、简单原则：尽量减少用户的记忆负担，一个完整的任务，仅仅需要两三步操作就可以完成，而不需要用户去记忆繁琐的操作步骤。

3、直接操作：在任何一个界面，用户都能以最少的点击到达另一个想要进入的界面。或许添加连接可以到达这一目的，但过多的连接、按钮会影响用户对界面功能把握，所以要把握好这个度。

4.2.2 界面设计

1、主页面

主页面为系统的首页，此页面是用户操作的起点，在此可以登录、注册、查看问题以及查看提交。界面如下图 4.8 所示：

图 4.8 主界面

2、登录及注册界面

用户用来完成登录和注册，且二者之间可以通过一个连接进行相互跳转，方便已有账号用户直接到达登录页面，没有账号用户快速到达注册页面。界面如下图 4.9 和 4.10 所示：

图 4.9 登录界面

图 4.10 注册界面

3、测试案例界面

测试案例展示界面，主体部分展示测试案例的条目，用户可以通过搜索或直接点击分类类型快速定位到所希望的测试案例。界面如下图 4.11 所示：

图 4.11 测试案例列表

4、提交测试用例界面

通过一个遮盖层向用户提供测试用例的编写和提交，这样可以避免用户进行其他不必要的操作，简化用户的提交操作。

图 4.12 提交测试用例

5、结果反馈界面

结果展示界面向用户反馈了测评结果，分三部分：最上端向用户展示了测试用例测评的总体信息；中间向用户展示了测评过程的日志记录，便于用户查看详细的测评过程。最下端向用户展示了自己编写的测试用例，便于用户回忆自己提交的测试用例。

图 4.13 测评结果反馈界面

6、系统设置界面

此界面包含了当前用户的提交统计信息以及用户的账户信息，根据用户角色的不同显示第三项，教师显示提交案例管理、管理员显示系统管理、学生无此项。

图 4.14 设置界面

7、系统管理界面首页

管理首页向管理员展示了系统的大体概况，用户情况、测试案例情况、提交状况以及测评机信息。

图 4.15 管理员管理首页

8、测试案例管理页面

此界面向管理员展示了测试案例列表，提供了批量删除，按条件查询，翻页，详情等操作按钮或连接。

图 4.16 测试案例管理

9、教师管理测试案例界面

向教师展示了自己所提交测试案例的情况，并提供了其对测试案例的添加、查看及编辑操作。

图 4.17 教师管理页面

10、其他界面设计

其他界面设计风格与上述界面类似，受篇幅有限，不再列出。

4.3 数据库设计

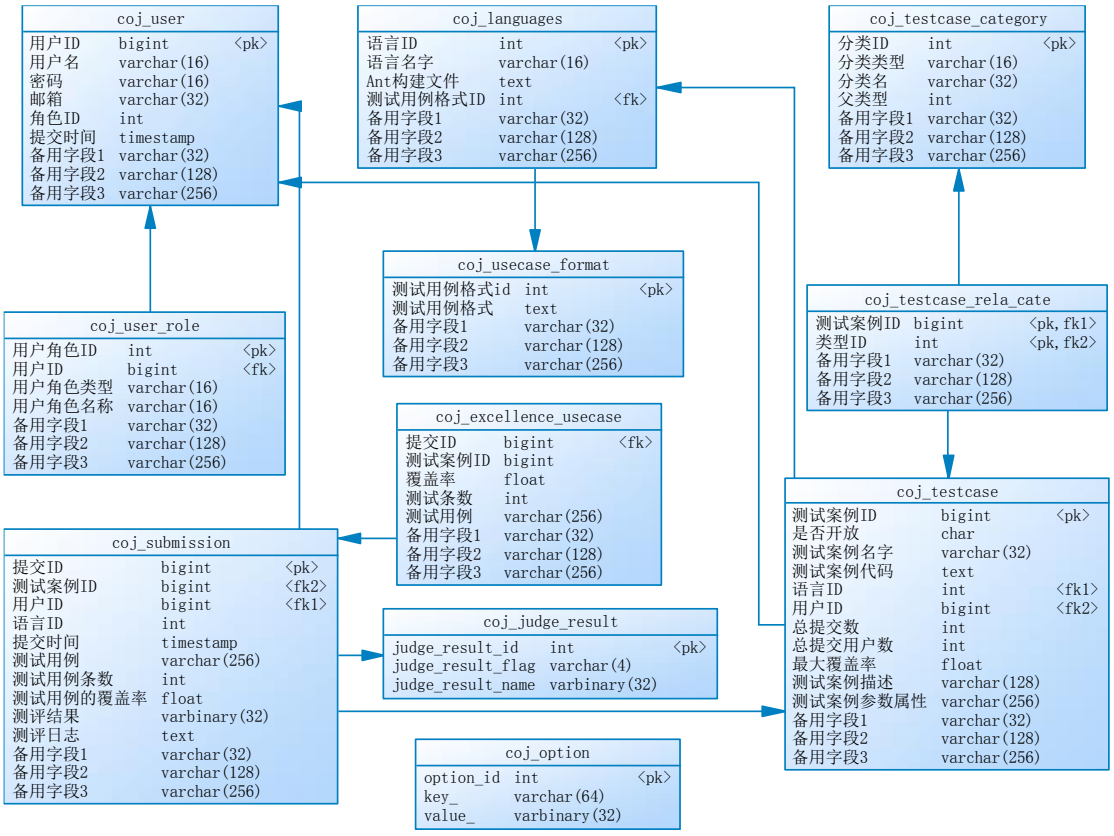

图 4.18 系统数据库结构

本系统总共涉及 14 张表，每张表的作用如下表 4.1 所示：

表 4.1 系统表功能

表 **Name**	**Code**	**Comment**
测试案例表		
coj_testcase	coj_testcase	测试案例表
coj_testcase_category	coj_testcase_category	测试案例类型
coj_testcase_rela_cate	coj_testcase_rela_cate	测试案例和类型关系表
用户表		
coj_user_role	coj_user_role	用户角色类型分类
coj_user	coj_user	记录用户信息
提交用例表		
coj_submission	coj_submission	提交测试案例

coj_excellence_usecase	coj_excellence_usecase	优秀的测试用例设计
数据字典		
coj_usecase_format	coj_usecase_format	测试用例编写格式
coj_language	coj_language	语言
coj_judge_result	coj_judge_result	测评结果
coj_option	coj_option	系统属性

4.4 系统开发结构设计

4.3.1 技术结构

图 4.19 展示了 Web 端开发中要使用的技术、图 4.20 展示了测评机开发所使用的技术。本系统的静态包组织结构如下表 4.2；开发视图从技术及组织结构的角度介绍了系统的开发结构。

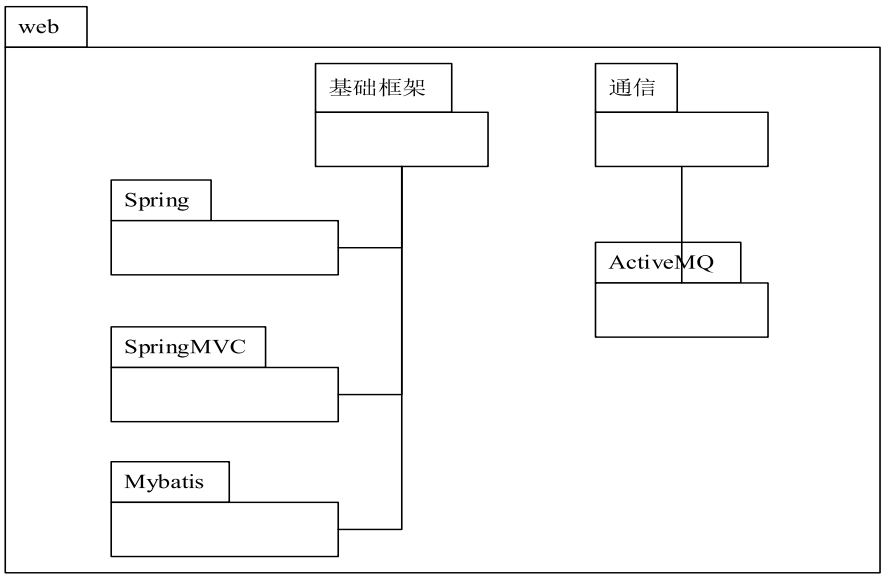

图 4.19 Web 相关技术

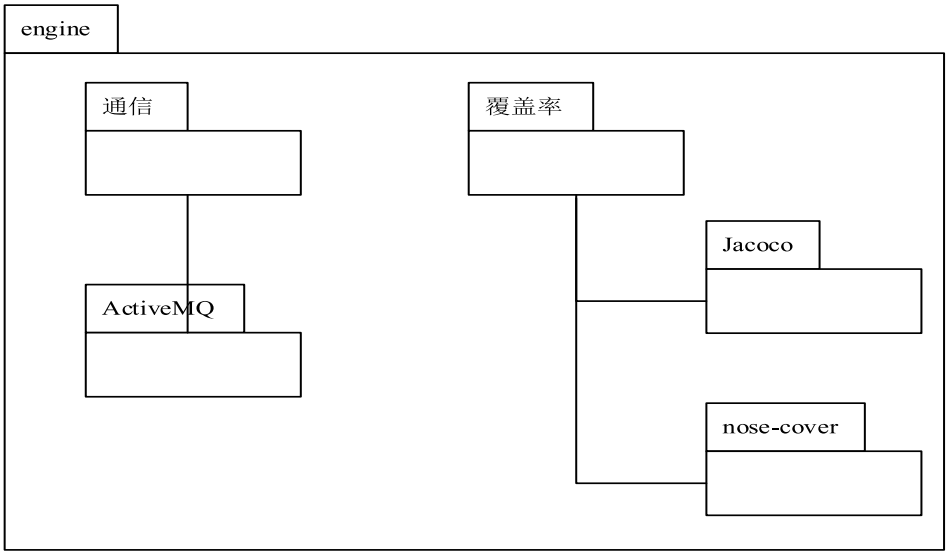

图 4.20 测评机相关技术

4.3.2 静态包结构

表 4.2 项目开发包结构

项目名	包名	说明
caseonline.engine		测评机
	caseonline.judger.engine.application	入口
	caseonline.judger.engine.core	测评核心实现
	caseonline.judger.engine.exception	异常
	caseonline.judger.engine.mapper	数据接口
	caseonline.judger.engine.messenger	消息处理
	caseonline.judger.engine.model	数据实体
	caseonline.judger.engine.util	工具
caseonline.web		
	caseonline.judger.web.aspect	切面
	caseonline.judger.web.controller	控制
	caseonline.judger.web.exception	异常
	caseonline.judger.web.mapper	数据接口
	caseonline.judger.web.messenger	消息处理
	caseonline.judger.web.model	数据接口
	caseonline.judger.web.service	数据服务接口
	caseonline.judger.web.util	工具

4.5 程序模块设计

测评模块设计。

1、测评模块类图

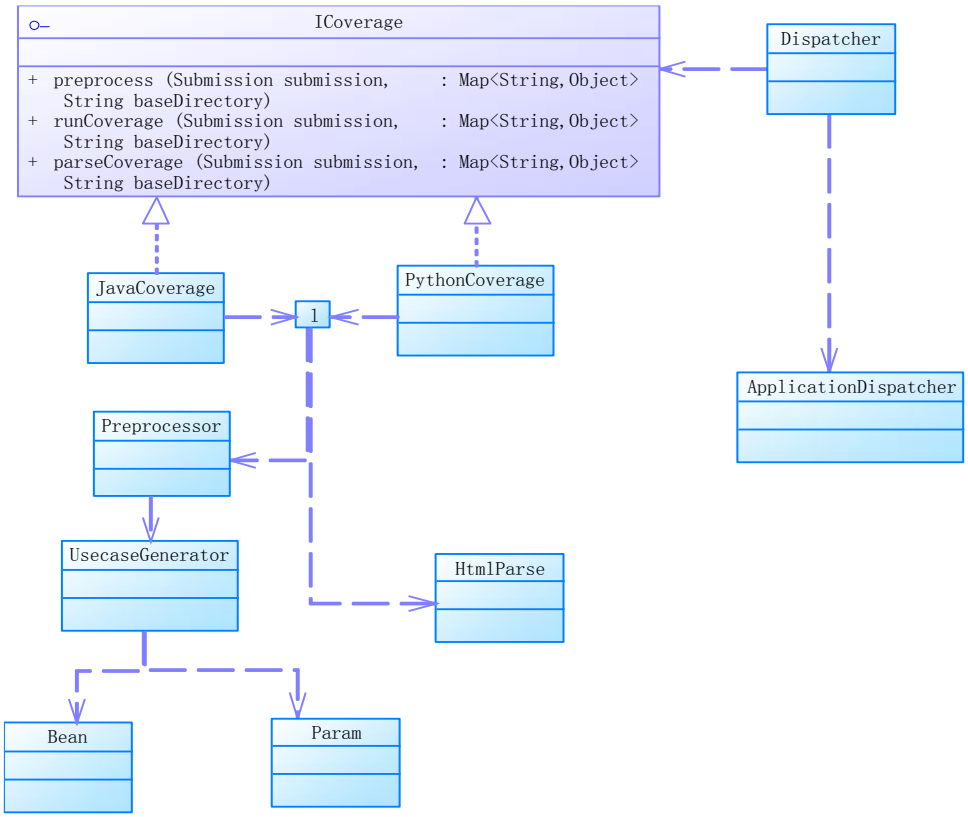

图 4.21 测评程序模块类图

测评程序模块在设计上，将整个测评过程抽象为三步：测评前的准备、测评和测评后的结果分析。为语言扩展的方便，定义统一接口 ICoverage，其中定义三个函数，来完成对应工作。JavaCoverage 和 PythonCoverage 是对 java 和 Python 语言覆盖率分析过程的实现，测试前期准备工作工具类 Preprocessor 实现测试案例，测试用例文件的创建工作，UsecaseGenerator 类实现根据测试用例脚本生成测试用例代码，Bean 和 Param 是 UsecaseGenerator 的辅助类。 覆盖率结果解析工具 HtmlParse 实现对测试用例覆盖率报告的解析。要实现某一语言的测试用例覆盖率统计扩展，需要实现 ICoverage 接口，实现接口中测评的三个方法。

2、测评模块流程图

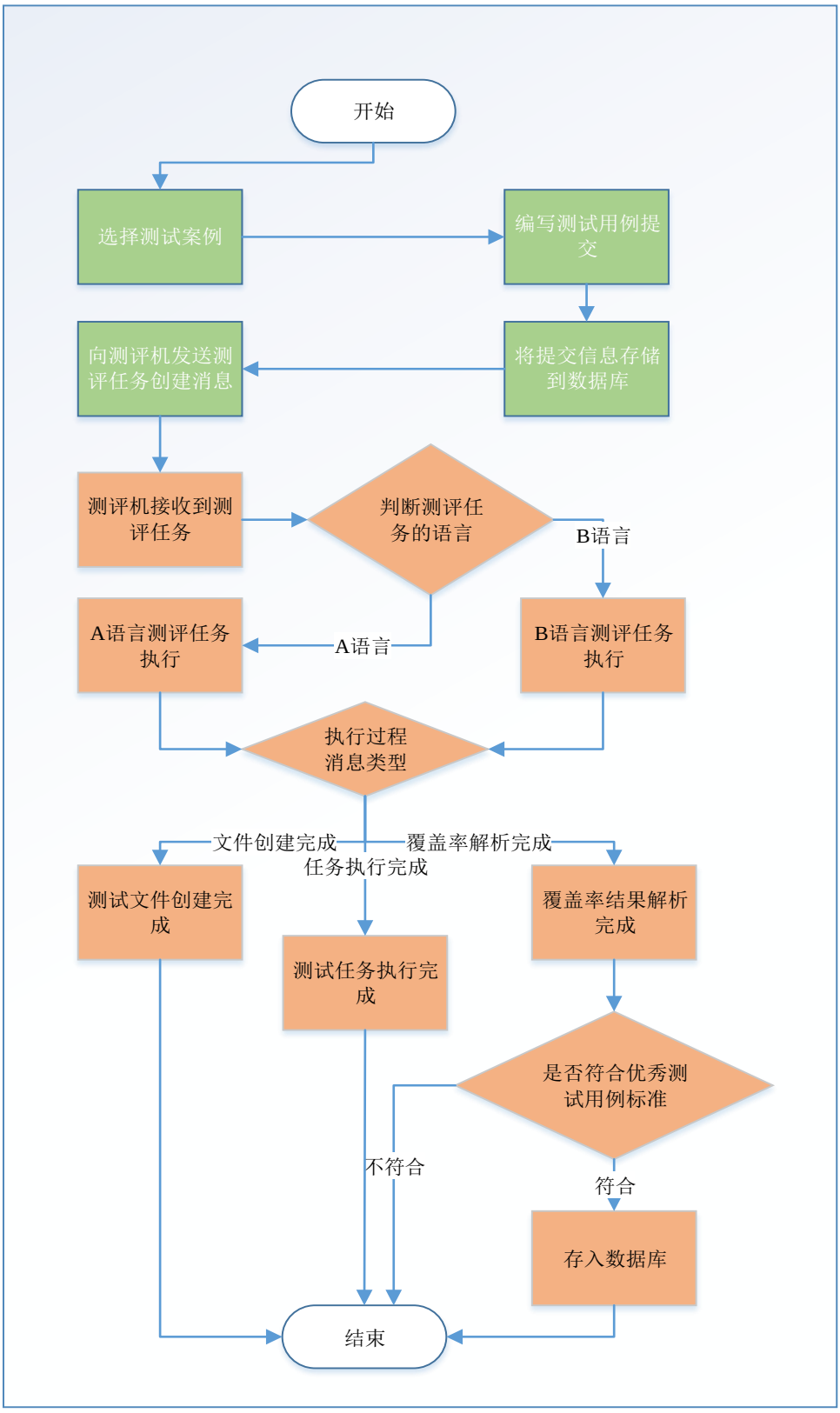

图 4.22 测评模块流程图

4.6 安全设计

4.5.1 用户权限划分

目前系统提供了三种用户类型，管理员、教师、学生。管理员拥有最高权限，可以管理及维护测试案例，测试用例提交等等；教师拥有二级权限，可以管理维护自己提交的测试案例查看优秀的测试用例等；学生拥有最低权限，查看测试案例、测试用例提交和自己提交的用例脚本等。目前的设计方案为根据用户角色类型的不同，其可见操作界面不同，处理请求的控制器 Controller 映射不同，返回的不同的页面且控制器会把角色类型参数传递到页面上，页面在浏览器中解释的时候会根据角色参数的不同解释不同的页面元素。

4.5.2 数据的一致性

本系统数据库设计中并未使用外键，在相关联的数据表进行添加和删除操作时，往往会出现不一致导致出错情况。所以为了避免这种情况，在进行删除操作时，同时考虑将其相关联的表项进行删除操作。

4.7 本章小结

这一章中详细介绍了系统的设计，从各个方面展示了系统的实现的方案。功能模块设计展示了系统的模块划分、各模块的功能和模块的出入参数和处理流程；数据库设计展示了系统的数据库基础，确定了系统的数据存储结构；用户界面设计展示了系统的门面，确定系统界面设计的简介明了的整体风格。程序模块设计主要涉及到一些程序代码的设计，以提高代码的可重用性；安全性设计主要考虑两个方面，一是用户权限，保障系统数据的安全性，避免一般用户对系统造成破坏。二是数据库，由于相关表之间没有设计外键，没有自动级联删除，所以需要通过代码来实现级联删除，保证数据的一致性，避免系统获取数据异常。

第五章 系统实现和测试

5.1 系统整体实现

系统整体上采用 B/S 结构实现，使用 Spring+SpringMVC+Mybatis 开发框架来实现，MySQL 作为数据库，Tomcat8.0 作为服务器，MQ 作为消息服务。使用 bootstrap 作为页面开发框架。整体实现结构如下图 5.1 所示：

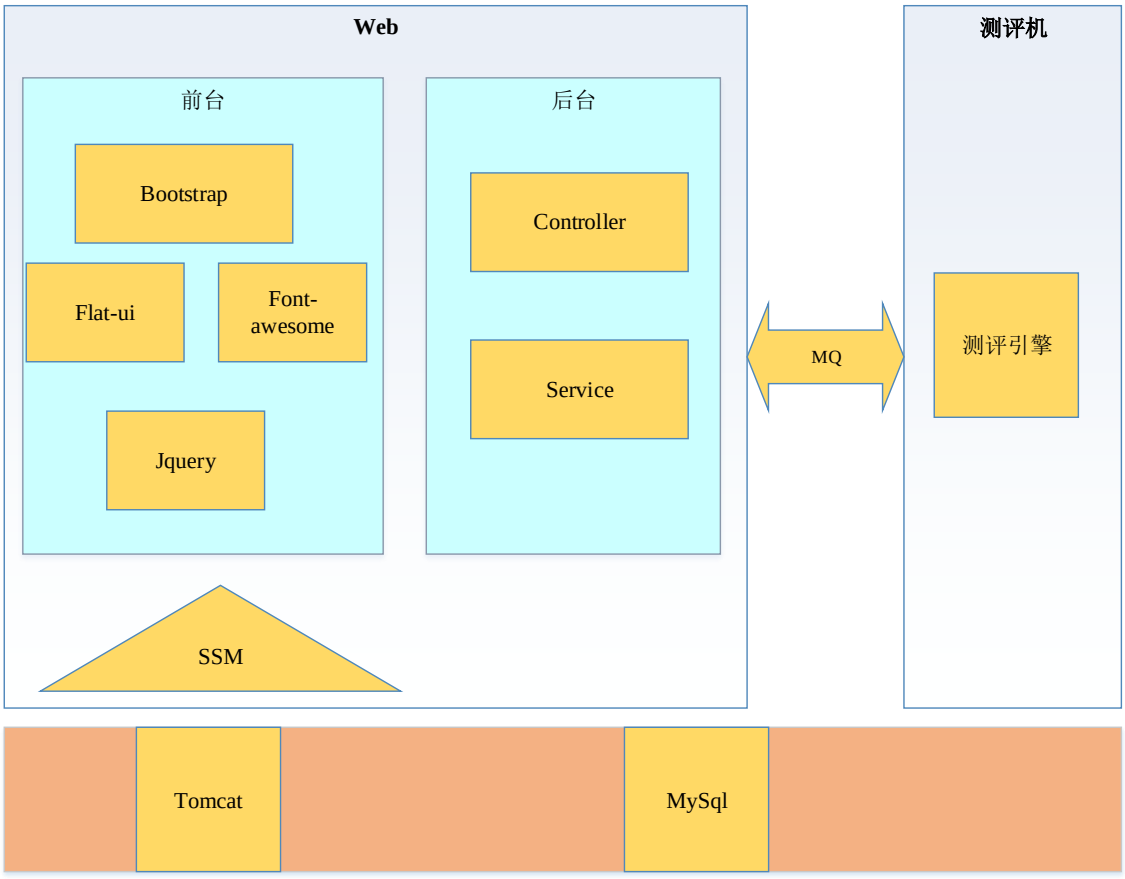

图 5.1 系统整体实现

5.2 关键技术实现

5.2.1 代码生成

测试用例提交时，由于编辑框没有代码提示，所以学生直接写代码不方便，另外直接编写测试用例代码也容易出错，考虑上述两个方面的原因，实际上提交的是测试脚本而不是测试代码，所以在执行测试时，需要根据脚本数据生成测试代码，即以测试用例脚本数据以及对应测试案例作为输入参数，以测试用例代码文件为输出结果。在代码生成工具的选择上，通过对比目前主流模板引擎 Freemarker 和 Velocity，Beetle 性能以及操作上都要优于它们，所以代码生成采用 Beetle 模板引擎，具体实现流程如下图 5.2 所示：

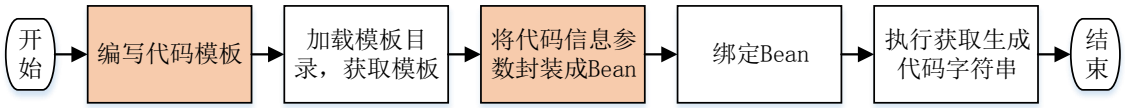

图 5.2 使用 Beetl 模板生成代码流程

使用 Beetle 生成代码模板的过程中，代码模板的编写和把信息参数封装成 Bean 是关键步骤，且二者相互影响的。如本系统中 java 测试用例代码的生成，在封装 Bean 时，将测试用例封装成 Bean 的 List<Param> paramInfo 属性，在编写代码模板时需要利用循环来生成每一测试用例对应的方法。下图 5.3 展示了模板中循环生成测试方法的部分。

```
  <%
  var i=0;
  for(param in bean.paramInfo){
            i++;
  %>
  @Test
  public void test_${param.methodName}${i}() {
  <%
      var date=json(param.params);
      var len=strutil.length(date);
      var p=strutil.subStringTo(date,1,len-1);
    %>
       assertEquals(${param.expectValue},
${bean.testClassName}.${param.methodName}(${p}));
  }
                                <%}%>
```

图 5.3 Java 测试用例代码生成模板

5.2.2 消息通信

为了提高系统的可用性，防止测评时出现问题导致系统死机的情况，本系统将测评机作为一个独立的整体。测评机与其他部分的通信通过 MQ 消息服务器来实现，具体消息通讯如下图 5-4 所示：

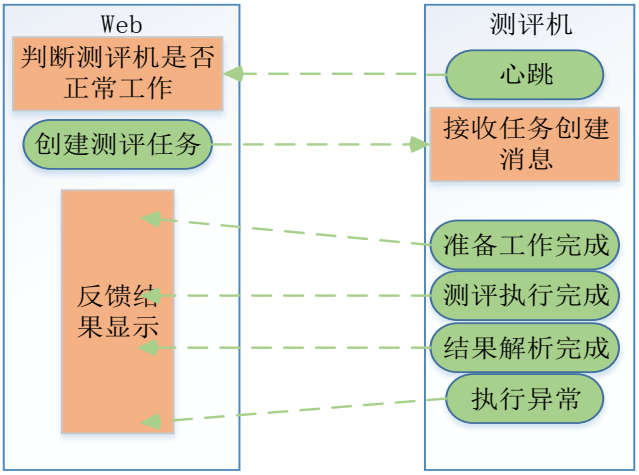

图 5.4 Web 端与测评机消息通讯

学生提交测试用例后，首先将测试用例相关属性信息存入数据库中，然后向测评机发送提交创建消息，测评机收到消息后，首先根据提交 ID 获取提交用例信息，然后创建测评任务（此部分为了避免线程同步问题，采用 synchronized 来保障一次只有一个测评任务在执行），执行测试用例的测评，完成测评后，将测评日志信息和结果存入数据库，同时将测评进度和结果信息打包成消息反馈给 Web 端，向用户展示测评结果。关键代码实现如下图 5.5 所示：

存入数据库	submissionMapper.createSubmission(submission)
发送提交消息到测评机	mapMessage.put("submissionId",submissionId); messageSender.sendMessage(mapMessage);
测评机收到提交消息	if (event.equals("SubmissionCreated")) { LOGGER.info("收到任务"); newSubmissionHandler(mapMessage);}
更新提交数据并发送消息给Web	submissionMapper.updateSubmission(submission);mapMessage.put("coverageInfo",coverageInfo) messageSender.sendMessage(mapMessage);

图 5.5 测评机消息通信实现

5.2.3 测评机

测评机流程

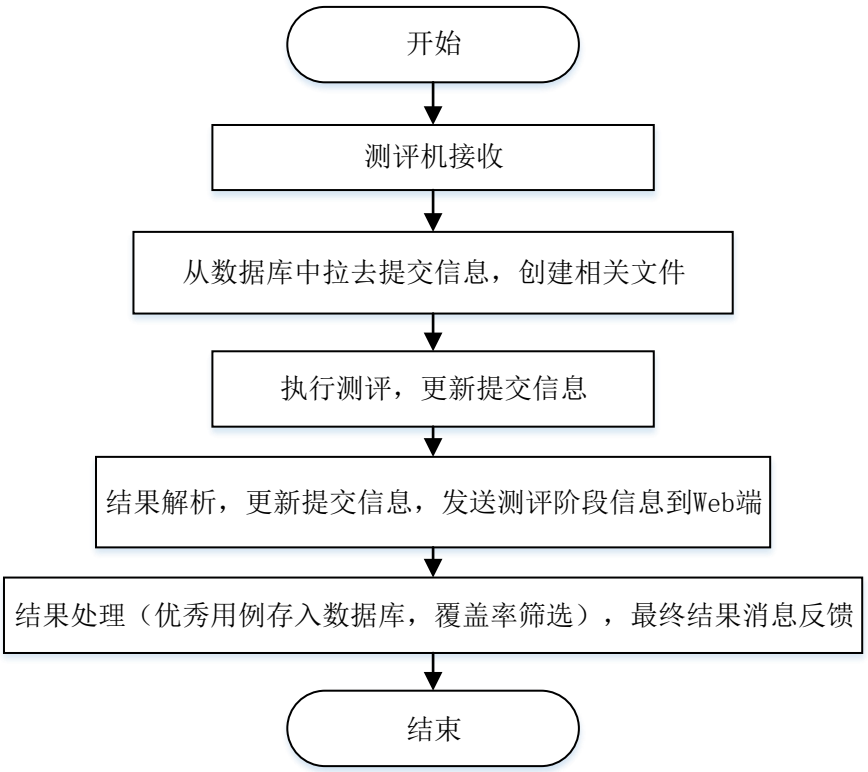

图 5.6 测评机执行流程

测评机是在线测评系统的核心，其主要功能就是接收测评请求、创建测评任务（执行前准备、执行测评任务、覆盖率解析）、结果处理及反馈。接收测评请求的实现是通过 MQ 消息服务器来实现；测评任务的创建，测评请求消息体包含由提交 ID，创建测评任务时可以根据提交 ID 来获取提交的测试用例以及对应的测试案例信息，根据获取的信息完成相应的文件创建等前期准备工作，执行测评任务通过 Java 的 Running.exec()来执行相应语言的测评指令，执行完成之后会生成对应的覆盖率报告（针对目前支持的语言 Java 和 Python，来说报告是 html 形式），通过 html 解析工具（Jsoup）完成覆盖率报告的解析；测评完成后将测评的过程日志、结果存入数据库，根据测评的覆盖率选出有优秀的测试用例存入数据库中。

在测评的过程中每完成一个阶段都会给 Web 端发送相应的消息，告知测评的进度。由于测评机是独立的，所以可以同时部署多个测评机，以此来提高系统的效率，减轻测评机的负载。其实现关键代码如下图 5.7 所示：

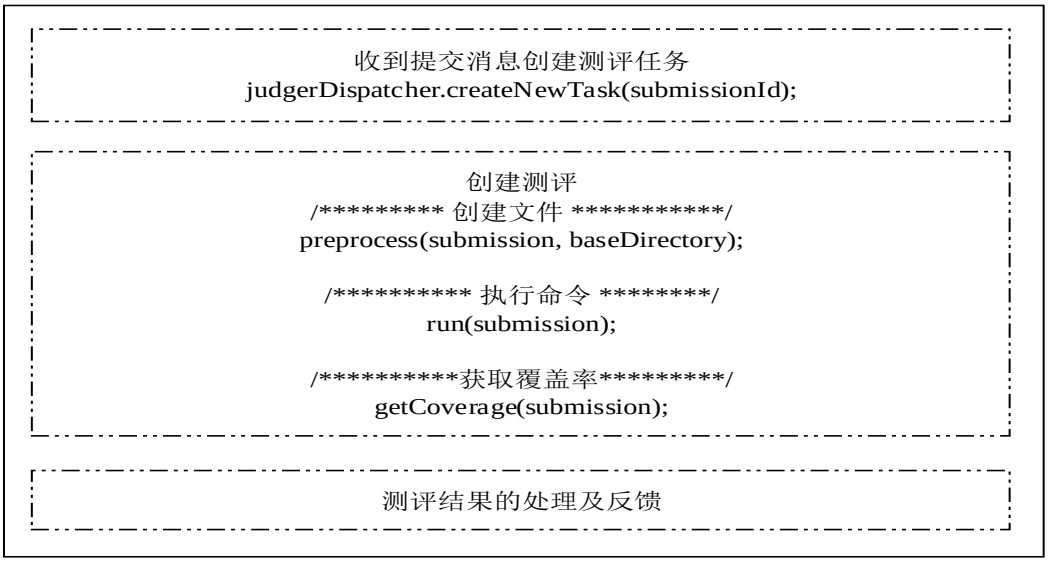

图 5.7 测评机执行

5.2.4 获取实时测评结果

获取最新的测评进度及测评结果，有两种方式实现，一种方式是通过轮询的方式拉取数据，这种方式通过周期性的调用 Ajax 请求，来获取最新数据；另一种方式是通过服务器推送数据，SSE（Server Send Event）和 Websockets 能够很方便的实现服务器端推送数据。Websockets 的接口实现的是客户端和服务器端之间的双向通信，对应的开销也会更大。而对于本系统场景，只需要单向的服务器端推送即可，所以使用 SSE 来实现更为合适，另外日志的显示采用轮询的方式获取。流程如下图 5.8 所示：

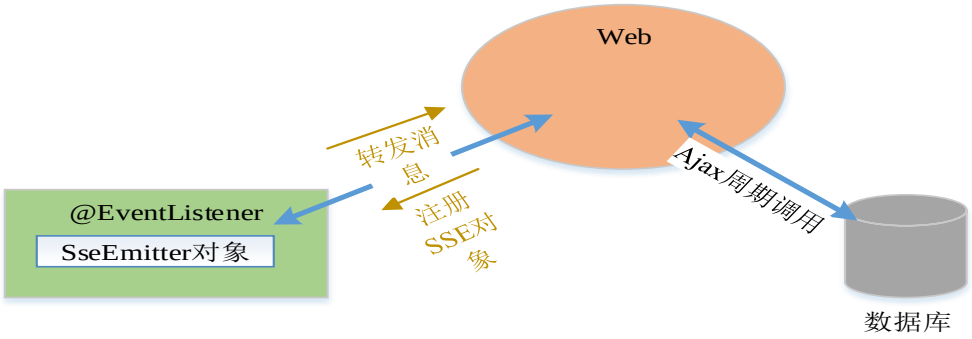

图 5.8 实时测评信息获取流程

SSE 实现关键代码如下图 5.9 和图 5-10 所示：

(1) 客户端实现

```
var subscriptionUrl = '<c:url value="/submission/getRealTimeJudgeResult.action?submissionId=${submission.submissionId}" />',
        source        = new EventSource(subscriptionUrl),
        lastMessage   = '';

    source.onmessage   = function(e) {
      var message     = e['data'];
    …
    }
```

图 5.9 SSE 客户端实现关键代码

(2) 服务器端实现

```
                                    转发测评机发送来的消息
    @EventListener
    public void submissionEventHadnler(SubmissionEvent event)。。{
            String message = event.getMessage();
            。。。
            SseEmitter sseEmitter = sseEmitters.get(submissionId);
            Map<String, Object> mapMessage = new HashMap<String, Object>();
            mapMessage.put("message", message);
            。。。
            sseEmitter.send(mapMessage);
            if (isCompleted) {
                    sseEmitter.complete();
                    removeSseEmitters(submissionId);
            }}
```

图 5.10 SSE 服务器端实现关键代码

5.3 系统测试

5.3.1 测试环境与测试方案

1、测试环境

表 5.1 系统测试环境

系统	Window7
JDK	1.8
Python	3.6
数据库	mysql5.5
服务器	apache-tomcat-8.5
构件工具	apache-ant-1.10
消息服务器	apache-activemq-5.6
测试工具	Junit4

2、测试方案：

系统测试大体分两方面进行测试：

(1) 接口测试

描述：测试数据层的数据操作接口。

流程：

第一步：测试参数环境配置。

第二步：编写测试类。

第三步：执行分析执行结果。

(2) 功能测试

(a) 系统的核心功能测试

描述：测试用例的测评，主要测试系统的测评功能，能否正确分析测试用例，获取测评结果。

流程：

第一步：选择测试案例，根据类型或名称筛选测试案例。

第二步：编写测试用例并提交，查看测评结果及测评日志。

第三步：查看提交列表。

(b) 系统相关实体的管理功能测试

描述：测试数据库实体的添加、修改、删除，查询。

流程：

第一步：输入操作实体参数。

第二步：执行操作。

第三步：结果对照。

(c) 错误处理测试

描述：测试程序对错误输入参数，错误操作的处理。

流程：

第一步：分析可能出错的操作。

第二步：执行错误操作。

第三步：查看错误处理是否与设想的一致。

5.3.2 系统测试数据与流程

1、数据准备

(a) 数据库

通过 sql 脚本完成数据库的初始化，涉及的数据表如下：

表 5.2 测试需要初始化的表

名称	**说明**
coj_user_role	角色
coj_user	用户
coj_usecase_format	用例格式
coj_testcase_category	案例类型
coj_option	系统属性
coj_language	语言
coj_judge_result	测评结果

(b) 测试用例

表 5.3 测试用例数据

测试用例分组	**组 1**	**组 2**	**组 3**
用例	3,1,2,3 3,3,2,1 0,0,0,0 1,0,0,1	3,1,2,3 3,1,3,2 3,3,2,1 0,0,0,0 1,0,1,0 3,2,1,3	0,0,0,0

2、测试流程

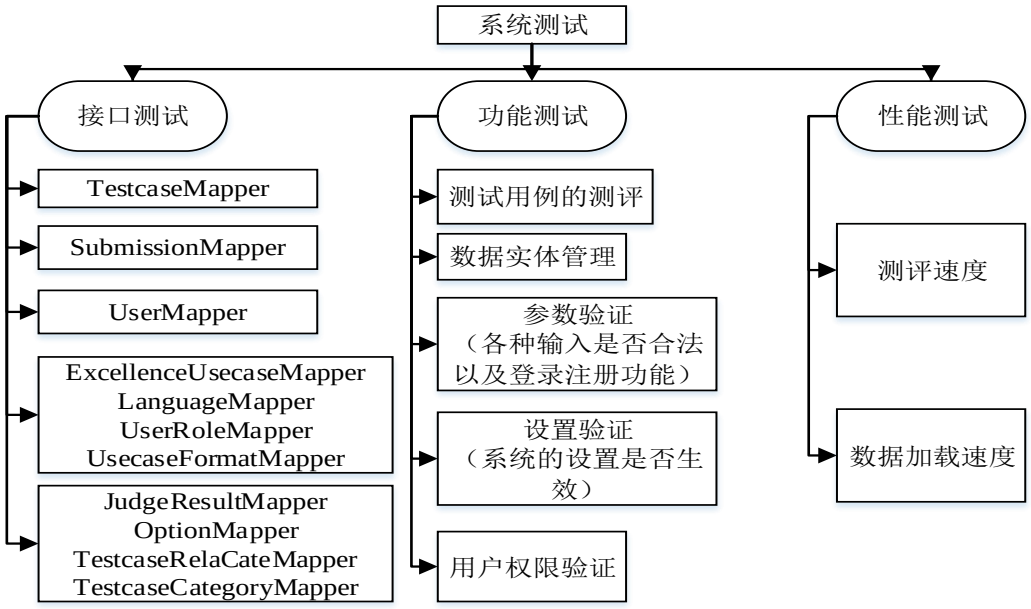

图 5.11 系统测试流程

受篇幅限制，本文中仅展示部分测试过程。

(1) 接口测试

(a) 测试案例接口（TestcaseMapper）

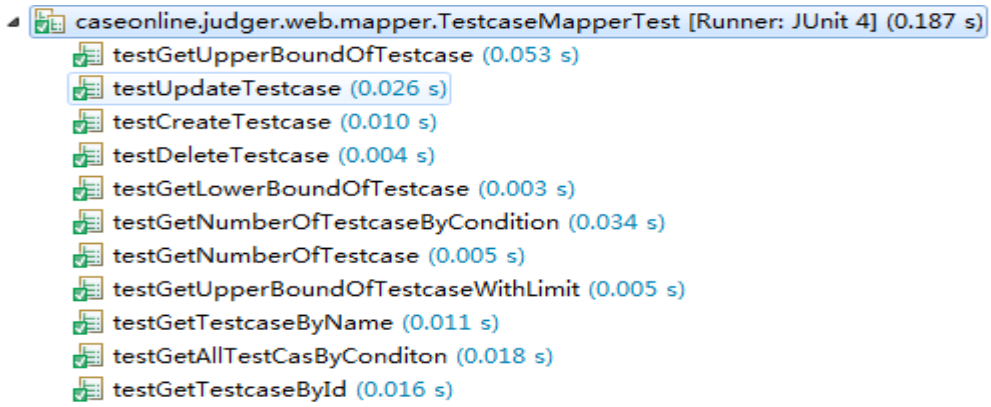

图 5.12 测试案例接口测试结果

(b) 提交接口（SubmissionMapper）

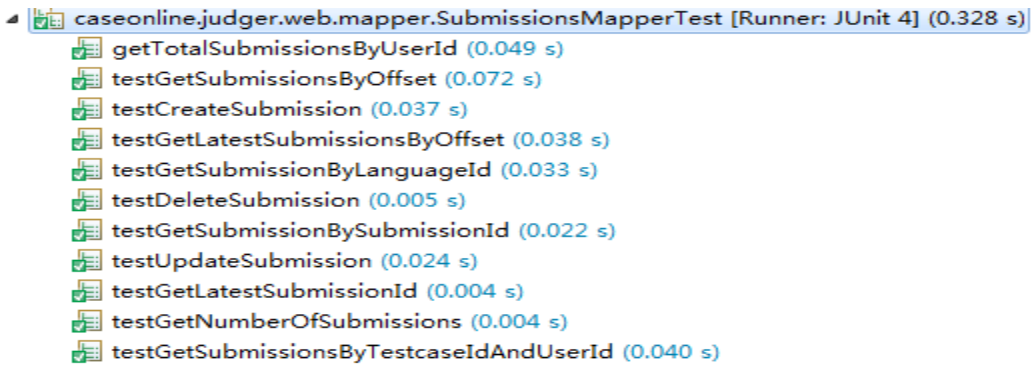

图 5.13 提交接口测试结果

(2) 功能测试

(a) 测试用例测评

图 5.14 提交测试用例

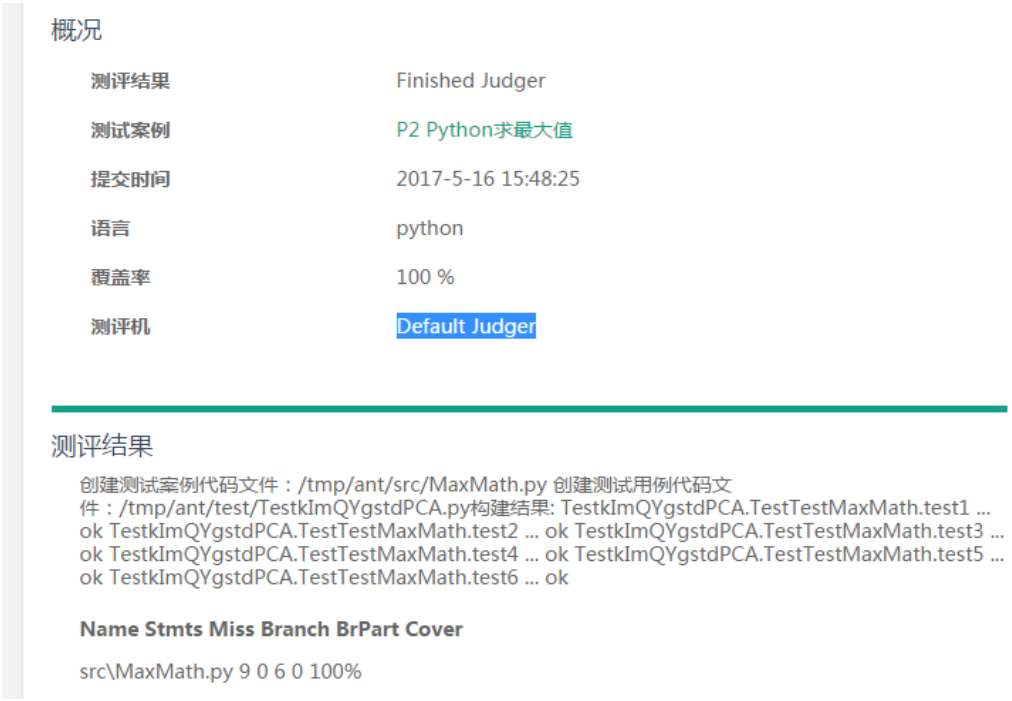

图 5.15 测评结果

(b) 添加测试案例

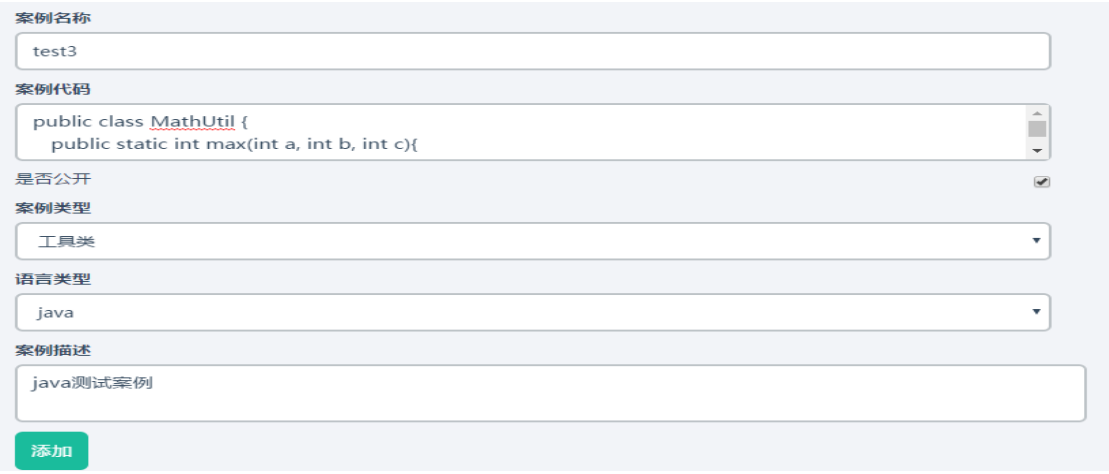

图 5.16 添加测试案例

图 5.17 添加成功

(c) 登录和注册信息验证

图 5.18 登录和注册验证

(d) 案例提交验证

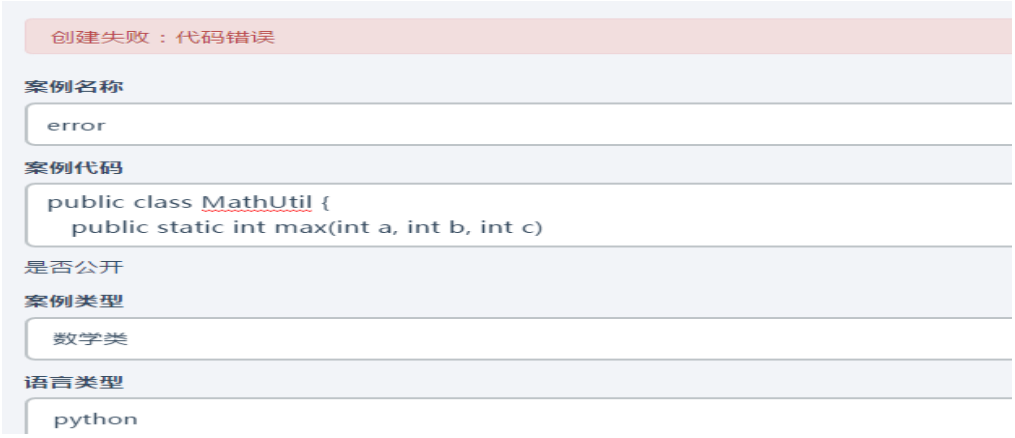

图 5.19 测试案例提交验证

(e) 关闭注册功能

图 5.20 关闭注册功能

(f) 未登录用户无法提交

图 5.21 未登录用户无法提交

(g) 教师只能管理自己提交的案例

图 5.22 教师管理本身提交案例

(3) 性能测试

(a) 通过日志查看测评每个阶段的时间

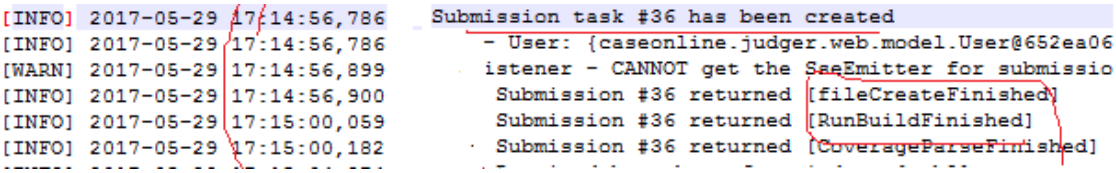

图 5.23 测评日志

(b) 在数据量较大时查看实体列表加载速度

在不同数据量的情况下，体验数据列表的加载速度。

5.3.3 系统测试结果与分析

1、功能测试结果

表 5.4 功能测试结果

测试功能点	测试结果
测试案例的查询筛选	正常
测试用例的提交获取测评结果	正常
测试用例提交的查询筛选	正常
用户的管理（添加、删除、修改、查询）	正常
测试案例的管理（添加、删除、修改、查询）	正常

测试用例提交（查询、删除）	正常
系统设置	正常
用户操作限制	正常
验证（测试案例代码、用户注册信息等）	正常
提交用例排名	正常
用户的登录注册	正常
测试用例检查验证	正常

2、数据接口测试结果

表 5.5 数据接口测试接口

接口	说明	测试结果
ExcellenceUsecaseMapper	优秀测试用例提交接口	正常
JudgeResultMapper	测评结果接口	正常
LanguageMapper	支持语言接口	正常
OptionMapper	系统属性接口	正常
SubmissionMapper	测试用例提交接口	正常
TestcaseCategoryMapper	测试案例类型接口	正常
TestcaseMapper	测试案例接口	正常
TestcaseRelaCateMapper	测试案例类型关系接口	正常
UsecaseFormatMapper	测试用例格式接口	正常
UserMapper	用户接口	正常
UserRoleMapper	用户角色接口	正常

3、性能测试结果

(1) 测评速度

(a) 测试结果

表 5.6 测评时间统计数据

阶段 / 用例		任务创建	文件准备	执行	解析
Java	组 1	18:53:40.676	18:53:41.708	18:53:45.248	18:53:45.500
	组 2	18:54:17.220	18:54:17.357	18:54:21.234	18:54:21.317
	组 3	18:54:30.935	18:54:31.087	18:54:34.573	18:54:34.684
Python	组 1	19:02:23.197	19:02:23.267	19:02:24.183	19:02:24.229
	组 2	19:02:53.887	19:02:53.969	19:02:55.205	19:02:55.337
	组 3	19:03:06.332	19:03:06.407	19:03:09.224	19:03:09.276

(b) 分析数据

表 5.7 测评时间分析数据

阶段 / 用例		创建文件耗时	执行耗时	解析耗时	总耗时
Java	组 1	0:00:01.032	0:00:03.540	0:00:00.252	**0:00:04.824**
	组 2	0:00:00.137	0:00:03.877	0:00:00.083	**0:00:04.097**
	组 3	0:00:00.152	0:00:03.486	0:00:00.111	**0:00:03.749**
	每阶段耗时	**0:00:01.321**	**0:00:10.903**	**0:00:00.446**	**0:00:12.670**
Python	组 1	0:00:00.070	0:00:00.916	0:00:00.046	**0:00:01.032**
	组 2	0:00:00.082	0:00:01.236	0:00:00.132	**0:00:01.450**
	组 3	0:00:00.075	0:00:02.817	0:00:00.052	**0:00:02.944**
	每阶段耗时	**0:00:00.227**	**0:00:04.969**	**0:00:00.230**	**0:00:05.426**

从分析数据表中可以看出，测评过程中的主要耗时点在测评的执行上，且就 Java 和 Python 而言，Java 语言的测试用例测评速度要慢于 Python。从执行的过程来看，Java 执行测评时是通过 Ant 构件来执行的，构件文件中的任务包含编译、打包、单元测试、覆盖率执行、打印单元测试报告和打印覆盖率报告。但是测评任务只需要一份覆盖率报告，所以可以删除不必要的环节如打包和打印单元测试报告，以此来提高测评执行的速度。

(2) 页面加载列表数据

当数据量大时感觉页面加载速度变慢，一种优化措施是，对于查询较多的数据表，添加 Mybatis 缓存，加快数据查询速度。

5.4 本章小结

在这一章中主要讲解了系统的整体实现以及系统中关键技术点的实现方式，包括如何根据测试用例数据代码的生成、Web 是如何与测评机进行消息通信的以及测评机实现细节。然后介绍了系统的测试，包含了系统测试环境与测试方案，测试需要的数据环境，简要的测试过程展示，最后是对测试过程中的结果进行统计与分析。就当前测试，系统基本上可以正常工作，实现所设计的功能。但是测试上难免有所疏漏，或许系统还有一些没有测试出的 Bug，这就有待以后使用过程中发现。

第六章 总结和展望

6.1 总结

本论文主要针对目前软件测试实践教学环节存在的问题，设计开发一款在线测评测试用例的系统。本系统结合在线判题系统的原理、整合软件测试工具以及使用 Web 技术开发。在线判题系统基本原理就是通过 Web 页面选择试题提交，具体对题提交答案的判定由判题引擎来完成。本文主要涉及两个方面的开发探究，一是测评机的开发，一是 Web 端的开发。

在测评机开发方面，目前确定的评判标准是单元测试的覆盖率，单元测试覆盖率统计工具有 Java 语言的 Jacoco，python 语言的 nose-cover 等等，这些工具以源码及单元测试类作为输入参数，以覆盖率报告作为输出结果。所以测评机需要做的就是：根据测试任务请求创建文件，然后使用指令执行覆盖率统计工具，得出覆盖率结果。同时考虑到测评机对支持语言的扩展性，将测评的工作抽象封装成对应的三步，文件创建、执行、结果解析。

在 Web 开发方面，采用 SSM 框架开发，系统包含四个层次：控制层 XXXController 用来处理具体的用户请求。服务层 XXXService 为控制层提供调用方法，实现对下层数据处理的封装。数据层 XXXMapper 为服务层提供了数据操作接口，实现了数据的基本操作。用户展现层即页面提供用户的可操作界面。

通过论文研究，得出以下几点结论：

(1) 在线测评可以有效的降低软件测试实践成本，给学生和老师提供了方便。学生不必去关心测试环境，只需要全心设计测试用例；教师也可以收集测试结果，对测试成绩分析。

(2) 模块化开发有效的提高开发的效率。此模块的概念小到一个功能模块如本系统的系统管理模块，大到一个系统如本系统的 Web 端。这种模块话开发，彼此之间分离，只需要关注本模块的开发，这在团队开发过程中尤为重要。例如本系统中的 Web 端和测评机，二者相互独立，以 MQ 作为第三方进行通信，这样在测评机的扩展上也是非常方便。

6.2 展望

本论文针对目前软件测试实践教学环节中存在的问题，设计开发了一款能够在线测评测试用例的系统，但尚存在一些不足，归纳如下：

1、测评方面，目前系统可以完成单元测试测试用例的覆盖率的测评，受覆盖率工具的限制，支持的覆盖率标准较少，如 Python 的 nose-cover 仅仅支持分支覆盖率的统计，所以可以参考覆盖率工具实现原理[16]，自己设计实现一种单元测试覆盖率统计工具。

2、在支持语言扩展方面，目前需要手动去配置新语言单元测试覆盖率统计环境，且需要改动测评机代码，一种理想的扩展方案是，将新语言的测评实现进行封装做成一个插件，直接在系统管理页面上配置实现扩展，或者是将测评实现打包，通过页面添加扩展时，向测评机发送下载包的指令，将新语言的测评包下载到服务器，并关联到测评机，以实现对新语言的支持。

3、此外此系统使用于单元测试，可以在此基础上进行开发，实现对性能测试的支持。

致谢

本系统设计过程中，得到了许多的帮助。很是感谢张剑波老师的指导，从系统设计到论文的编写，张老师都给出了不少的建议。另外张老师严格的教学方式和耐心的指导以及做事的严谨性对我的提高有不少帮助。还要感谢研究夏灯城和赵加奥学长在系统设计以及论文修改上的帮助。

另外还要感谢我身边的朋友和同学们，感谢他们为我提供了一个良好的学习环境，当然还要感谢自己的坚持和努力。

参考文献

[1] 王秀.谈软件工程中软件测试的重要性及方法[J]. 天津成人高等学校联合学报,2004,(02):76-78+81.

[2] 张盈. 在线考试评判系统的设计与实现[D].四川大学,2005.

[3] 卢依宁. 基于静态分析的 Java 单元测试教学反馈系统[D].南京大学,2016.

[4] 王迎. 自动化测试服务平台的设计与实现[D].北京交通大学,2015.

[5] 荀江萍. 浅谈基于云计算的软件测试[J]. 无线互联科技,2014,(05):45.

[6] 熊娇. 新型软件测试技术研究与实现[D].电子科技大学,2008.

[7] 焦安涛. 测试用例综合评价模型的研究[D].昆明理工大学,2009.

[8] 黄晓华,沈健,常晋义,张明新. 基于 Online Judge 与 HTML 批注技术的实验教学平台设计[J]. 计算机与现代化,2014,(11):117-121.

[9] 王腾,姚丹霖.Online Judge 系统的设计开发[J]. 计算机应用与软件,2006,23(12):129-130+137.

[10] 何迎生,罗强. Online Judge 评判内核的设计与实现[J]. 吉首大学学报(自然科学版),2010,31(06):37-39.

[11] SSM 框架笔记 http://www.kancloud.cn/digest/lunaticssm/122056[OL]

[12] JaCoCo：分析单元测试覆盖率的利器 https://www.ibm.com/developerworks/cn/java/j-lo-jacoco/[OL]

[13] nose introduction http://pythontesting.net/framework/nose/nose-introduction/[OL]

[14] Jsoup 中文文档 http://www.open-open.com/jsoup/parsing-a-document.htm[OL]

[15] Beetl 中文文档 http://ibeetl.com/guide/#beetl[OL]

[16] 魏光新,苏丽. 逻辑覆盖测试工具的设计与实现[J]. 计算机工程与应用,2000,(05):106-109.