《软件过程管理》课程作业

**课题名称:**  **软件过程管理**

学生姓名： 王鸿恺

学生学号: 2022141461163

指导教师： 毌攀良

**评阅成绩：**

评阅意见：

提交报告时间：2025年6月17日

目录

[1 CMMI的层次成熟度模型 3](#_Toc200849362)

[1.1模型简介 3](#_Toc200849363)

[1.2五个成熟度等级 3](#_Toc200849364)

[1.3关键过程域 4](#_Toc200849365)

[2过往开发经历 6](#_Toc200849366)

[2.1项目概述 6](#_Toc200849367)

[2.2技术路线图 8](#_Toc200849368)

[2.3系统设计 9](#_Toc200849369)

[2.3.1 系统功能设计 9](#_Toc200849370)

[2.3.2 软件架构设计 12](#_Toc200849371)

[2.4系统运行与测试 14](#_Toc200849372)

[2.4.1系统稳定性 14](#_Toc200849373)

[2.4.2系统测试报告 14](#_Toc200849374)

[2.4.3模型测试 14](#_Toc200849375)

[2.4.4系统性能 17](#_Toc200849376)

[3软件过程成熟度评估 18](#_Toc200849377)

[3.1评估方法与依据 18](#_Toc200849378)

[3.2评估结果分析 19](#_Toc200849379)

[4过程改进与改进计划 22](#_Toc200849380)

[4.1 当前成熟度级别与改进目标 22](#_Toc200849381)

[4.2 改进策略与措施 22](#_Toc200849382)

[4.3 实施步骤与时间安排 24](#_Toc200849383)

[4.4 风险管理与监控 25](#_Toc200849384)

# 1 CMMI的层次成熟度模型

## 1.1模型简介

能力成熟度模型集成（Capability Maturity Model Integration，CMMI）是由美国卡内基·梅隆大学软件工程研究所于 2000 年推出的过程改进框架，旨在帮助组织建立和完善其软件与系统工程过程。CMMI 强调以过程为中心，通过定义“什么”要做和“如何”做的最佳实践，来提升组织的过程能力和产品质量。CMMI 可应用于软件开发、系统工程、集成服务以及供应商管理等多个领域。

## 1.2五个成熟度等级

CMMI 层次成熟度模型将组织过程能力划分为五个成熟度等级，如下：

1. 第一级：初始级

过程无序，依赖个人英雄主义，难以复制成功经验。

1. 第二级：已管理级

建立基本项目管理流程，如需求管理、项目规划、项目监控等。

重点在“按计划执行”，保证项目在预定成本、进度范围内完成。

1. 第三级：已定义级

全组织层面制定并部署了标准化过程，各项目在此基础上可进行适当调整。

过程资产库和过程定义文件被正式维护。

1. 第四级：已量化管理级

引入统计与量化管理手段，对关键过程和产品质量属性进行度量与分析。

通过统计控制图等技术，监控过程绩效、识别变异原因并持续改进。

1. 第五级：优化级

组织持续以定量数据为基础，开展过程创新与优化改进。

聚焦于预防性过程改进与先进技术采纳，如自动化、模型驱动开发等。

## 1.3关键过程域

每个成熟度等级都包含若干关键过程域（Process Areas），组织需在该等级完整建立并有效运行对应的关键过程域实践，方可达到该成熟度，如下所示：

1. 已管理级

* 需求管理
* 项目规划
* 项目监控与控制
* 供应商协议管理
* 配置管理

1. 已定义级

* 过程定义
* 组织培训
* 风险管理
* 决策分析与解决

1. 已量化管理级

* 量化项目管理
* 组织过程绩效

1. 优化级

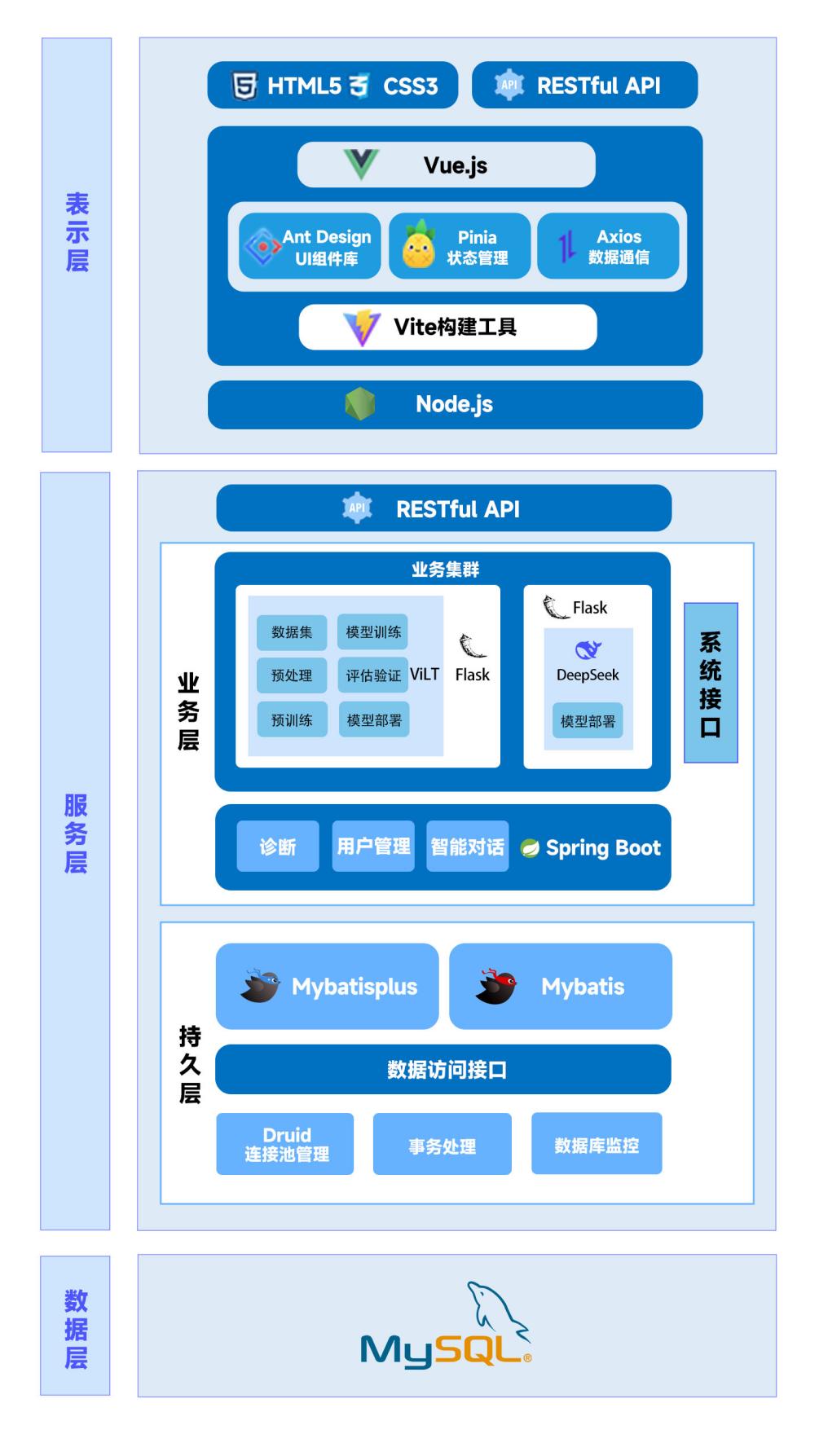
* 组织创新与部署
* 因果分析与解决

# 2过往开发经历

## 2.1项目概述

人工关节假体感染（Prosthetic Joint Infection, PJI）是关节置换术后产生的一种严重并发症，常导致患者需要反复清创、假体移除、延长住院时间，并可能引发患肢短缩和畸形。这不仅增加了患者的痛苦，也显著提高了医疗成本。每年全国范围内有超过90万例人工关节置换手术，这些患者术后复诊的需求非常高。然而，目前临床上缺乏特异性诊断方法，传统的诊断方式存在依赖医生主观判断、诊断效率低、准确度不高等问题。因此，建立一个有效的术后感染诊断平台具有重要意义。 本团队依托某著名三甲医院的病例数据，并以国家诊疗指南为基准，经过严密运算和论证，自主研发了一种基于多模态融合技术的术后感染智能诊断系统，以“数据+AI”为核心技术。该系统的主要功能模块包括智能诊断、病例管理和智能对话。 在智能诊断模块中，医生可以输入患者的实验室检查数据和X光片，系统通过多模态数据融合技术和深度神经网络模型进行分析，快速提供诊断结果。这一过程显著提高了诊断的效率和准确性。在病例管理模块中，系统允许医生对患者的病例及过往诊断结果进行系统化管理，医生可以方便地查看、添加、修改和删除病例信息，确保病历资料完整且易于查阅。智能对话模块则为医生提供了PJI诊断和治疗的建议，并帮助医生了解最新的医学前沿知识。 本系统不仅在诊断效率和准确性方面表现出色，在性能、扩展性、维护性和安全性方面也达到了高标准。我们采用前后端分离架构，分为表示层、业务层、持久层和数据层，每一层的构建都符合软件工程设计原则——高内聚低耦合，接口采用RESTful API风格。这使得系统在临床应用中具有显著优势，能够为医生提供可靠的工具，提升医疗服务质量和效率，减轻患者的经济负担和痛苦。 项目的核心技术是ViLT多模态模型的训练、优化与部署。ViLT是一种基于Transformer的视觉和语言处理架构，能够显著改善运行时间和参数效率，即使在图像处理中规避深度卷积，仍能保持优良性能。它的性能远优于传统卷积模型，将联合诊断的准确率从传统方法的85%提升至95%。在项目开发过程中，我们取得了多项突破性进展，包括专利授权、软件著作权授权。这些成就标志着我们项目的先进性和高价值。未来，本团队将纳入更多符合标准的研究对象作为模型优化的数据集，并不断进行试点应用，为社会和人民带来福祉。

## 2.2技术路线图

****

## 2.3系统设计

### 2.3.1 系统功能设计

该系统主要包括诊断模块、智能对话模块、用户模块和管理模块四个功能模块，各模块的功能如表2.3\_1所示：

表2.3\_1 系统功能模块表

|  |  |  |
| --- | --- | --- |
| 模块 | 功能描述 | 备注 |
| 诊断模块 | 用户输入实验室检查数据及X光片，经ViLT模型处理后输出诊断结果 | 实验室检查数据即C反应蛋白、红细胞沉降率、组织学检查等信息。结果为阳性或阴性。 |
| 智能对话模块 | 用户与大语言模型进行对话 | 可获取PJI相关诊断信息 |
| 用户模块 | 用户登录/注册；  用户账号、密码、手机号等信息管理 |  |
| 管理模块 | 用户对过往病例进行管理 | 包括病人信息及诊断结果 |

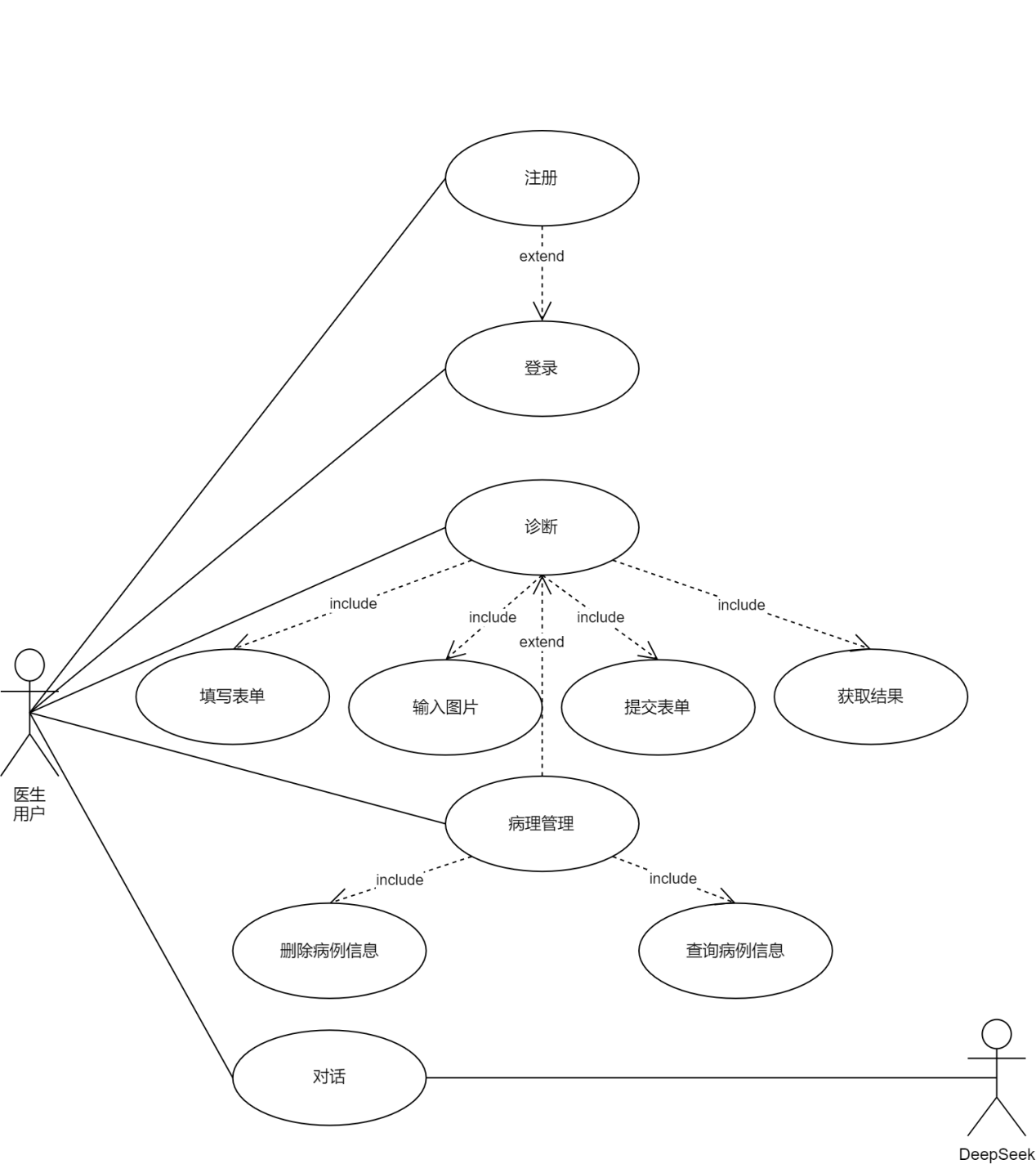


图2.3.2\_1 用例图

智能诊断：

医生可以输入患者的实验室检查数据和X光片，系统通过多模态数据融合技术和深度神经网络模型进行分析，快速提供诊断结果。这一功能有效提升了PJI诊断的准确性和效率，帮助医生做出更快速、精准的判断。

病例管理：

系统允许医生对患者的病例及过往诊断结果进行管理。医生可以方便地查看、添加、修改和删除病例信息，确保患者的病历资料完整且易于查阅。通过系统化的管理，医生可以更好地跟踪患者的治疗进程和恢复情况。

智能对话：

系统集成了智能对话模块，医生可以通过该模块获取PJI诊断和治疗的建议，了解最新的医学前沿知识。智能对话模块利用自然语言处理技术，为医生提供便捷的互动方式和专业的医学信息支持，提升诊断和治疗的质量。

系统不仅提供了高效、精准的PJI诊断服务，还在性能、扩展性、维护性和安全性方面达到了高标准。这使得系统在临床应用中具有显著的优势，能够为医生提供可靠的工具，提升医疗服务的质量和效率，减少患者的经济负担和痛苦。

### 2.3.2 软件架构设计

在开发过程中，将系统后端划分为通用模块、用户模块、管理模块、智能对话模块、诊断模块以及API模块，同时系统划分为表示层、服务层和数据访问层三层。

其中，API模块单独抽象，负责向前端提供可供调用的API接口；用户模块、管理模块、智能对话模块、诊断模块各自负责处理四个模块的业务逻辑，通用模块则负责进行配置、用户鉴权、错误处理、外部接口封装、图片传输等各功能模块共用的功能实现。

系统各模块的调用关系如图2.3.2.1\_1所示：

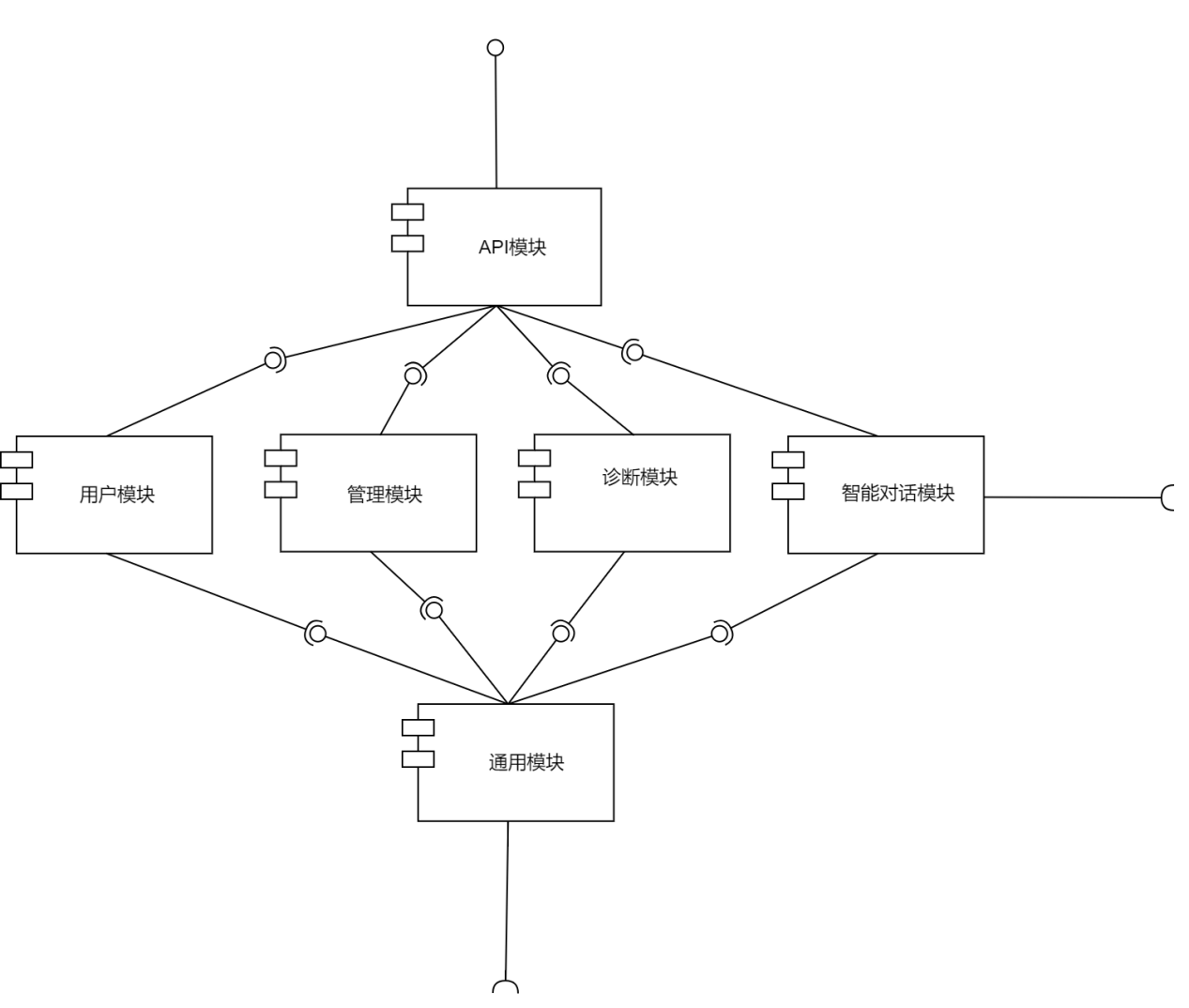


图2.3.2.1\_1 后端构件图

系统的部署主要包括客户端和服务端两个部分。其中，客户端部署形式为用户浏览器。服务端的部署主要包括Spring Boot服务器、CDN服务器、数据库服务器。

当用户向后端请求网页静态资源时，CDN服务器保存的网页静态资源将直接传输至用户，这样减少后端服务器的负载压力。系统所需的数据持久化储存在Mysql数据库服务器中。

系统的物理部署如图2.3.2.2\_1所示：

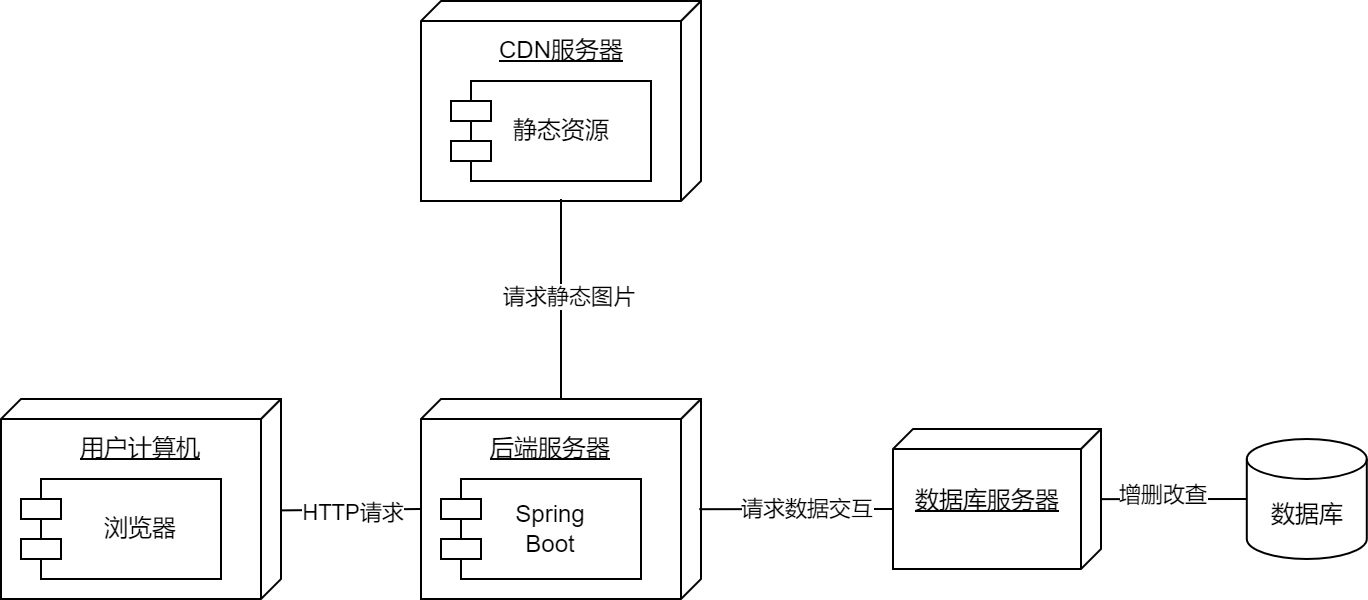


图2.3.2.2\_1 部署图

## 2.4系统运行与测试

### 2.4.1系统稳定性

自2024年1月上线以来，系统已稳定运行超过6个月。期间无重大故障发生，系统的平均无故障运行时间（MTBF）超过1000小时，证明了系统的高稳定性。

### 2.4.2系统测试报告

系统的功能测试主要是对术后诊断平台的各个功能模块和实现的功能目标进行测试，系统各个功能模块的测试用例主要包括单独输入X光医学图像是否可以输出结果的测试、单独输入临床医学数据的测试、API接口有效性的测试、网站界面的测试和算法模型诊断结果可视化测试。

### 2.4.3模型测试

模型测试如下表所示：

|  |  |  |
| --- | --- | --- |
| 测试内容 | 输入条件 | 结果 |
| 单独输入 X光医学图像  C反应蛋白与红细胞沉淀率为空 | 上传医学图像  不输入C反应蛋白与红细胞沉淀率 | 不能给出诊断结果 |
| 不输入X光医学图像  输入C反应蛋白与红细胞沉淀率 | 不上传医学图像  C 反应蛋白：  7034 ug/L  红细胞沉淀率：14mm/h | 不能给出诊断结果 |
| 输入X光医学图像  输入C反应蛋白  输入红细胞沉淀率  输入一、二阶段培养阳性  输入是否存在窦道  输入组织学阳性 | 上传医学图像  C反应蛋白：6984 ug/L  红细胞沉淀率：14mm/L  一、二阶段培养阳性：否  是否存在窦道：否  组织学阳性：否 | 给出诊断结果为：阴性 |

平台测试如下表所示：

|  |  |  |
| --- | --- | --- |
| 测试内容 | 输入条件 | 结果 |
| API 接口有效性的测试 | 网站是否成功调用算法模型的 API | 成功调用正常返回结果 |
| 首页开始诊断按钮 | 点击开始诊断 | 跳转到诊断界面 |
| 关于界面了解更多按钮 | 点击了解更多 | 弹出更多信息界面 |
| 更多信息界面退出 | 点击右上角按钮 | 退回到了解更多按钮 |
| 管理界面搜索功能 | 输入姓名，表格筛选指定条件输入 | 根据搜索条件  成功渲染表格 |
| 管理界面查看历史信息 | 点击查看按钮 | 弹出信息详情框 |
| 上传图片功能 | 在诊断界面点击上传 X 光影像部分，选择上传图片 | 调用上传功能  成功上传图并可预览 |
| 病理数据输入 | 输入 C 反应蛋白、红细胞沉淀率等输入数据 | 成功输入数据 |
| 生成结果测试 | 上传医学图像、输入临床文本数据后点击生成结果 | 汇总输入信息  并给出诊断结果 |

### 2.4.4系统性能

系统在日常运行中表现出色，平均响应时间为500毫秒，最大响应时间不超过800毫秒。系统能够支持同时在线的用户数达到200人，每日处理的数据量超过5GB，充分展示了其强大的并发处理能力和数据处理能力。

# 3软件过程成熟度评估

## 3.1评估方法与依据

本次对“多模态 PJI 智能诊断系统”项目的软件过程成熟度评估，采用 CMMI‑DEV 2.0 推荐的 SCAMPI‑C 自评流程，并结合定性访谈与定量度量两方面数据。具体如下：

**过程域选取：**

Level 2（已管理级）PA：需求管理（REQM）、项目规划（PP）、项目监控与控制（PMC）、配置管理（CM）、供应商协议管理（SAM）。

Level 3（已定义级）PA：组织过程定义（OPD）、组织培训（OT）、风险管理（RSKM）、决策分析与解决（DAR）。

**数据采集方式：**

文档审查：审阅立项需求说明、系统架构设计文档、功能模块设计表（表 2.3\_1）、测试报告、部署与运维记录等；

访谈问卷：对项目负责人、后端开发与测试人员进行结构化自评，覆盖每个 PA 的 Specific Goal 与 Generic Practice；

**定量指标：**

稳定性：MTBF ≈ 1 000 h；

性能：平均响应时间 ≈ 500 ms，最大 ≤ 800 ms；并发支持 200 用户，日处理 > 5 GB；

测试覆盖：功能测试用例涵盖诊断、对话、管理、API 四大模块；

异常率：上线半年内无重大故障，日均异常恢复时间 < 30 min。

**打分规则:**

每个 PA 下属的 SG/GP 按 “已实施（S）”、“部分实施（P）”、“未实施（N）”评级；

PA 整体覆盖率按 S:P:N 的比例计算；

仅当所有 PA 均评为 S 才可认定达成该成熟度等级。

## 3.2评估结果分析

| **过程域** | **SG/GP 覆盖** | **定量指标结果** | **评级** |
| --- | --- | --- | --- |
| **需求管理 (REQM)** | S:5 / P:0 / N:0 | 需求文档齐全，功能表（表 2.3\_1）完整 | 已实施 |
| **项目规划 (PP)** | S:2 / P:2 / N:1 | 规划文档存在，里程碑与实际进度偏差 ≈ 12% | 部分实施 |
| **项目监控与控制 (PMC)** | S:1 / P:3 / N:1 | 通过日志监控响应与 MTBF，但缺少定期审议 | 部分实施 |
| **配置管理 (CM)** | S:4 / P:1 / N:0 | Git+CI/CD 完整，API 接口版本可追溯 | 已实施 |
| **供应商协议管理 (SAM)** | S:0 / P:0 / N:3 | 未对 DeepSeek 等第三方服务签订正式 SLA | 未实施 |
| **组织过程定义 (OPD)** | S:1 / P:2 / N:2 | 有系统架构与模块设计文档，但缺少流程标准化手册 | 部分实施 |
| **组织培训 (OT)** | S:0 / P:1 / N:2 | 团队内部有一次框架和模型分享，无正式培训计划 | 部分实施 |
| **风险管理 (RSKM)** | S:2 / P:1 / N:2 | 识别了性能与安全风险，记录存在但未定期复审 | 部分实施 |
| **决策分析与解决 (DAR)** | S:0 / P:1 / N:2 | 核心技术选型（ViLT vs. 传统 CNN）有讨论记录 | 部分实施 |

**总体成熟度水平：**

Level 2：五个 Level 2 PA 中，只有 REQM 与 CM 达到“已实施”，其余均为“部分实施”或“未实施”，尚未达到 Level 2 整体要求。

Level 3：四个 Level 3 PA 均未完全落实，距离 Level 3 要求尚有较大差距。

**定量绩效对比：**

稳定性（MTBF 1 000 h）：优于行业基准（典型 ≥ 500 h），体现良好运营维护；

性能（平均 500 ms）：接近 SLA ≤ 500 ms 目标，偶有峰值延迟至 800 ms；

测试覆盖：功能测试用例全面，但缺少自动化回归测试，故障恢复仍依赖人工干预。

**主要改进方向：**

完善项目监控：引入定期评审与仪表盘（Dashboard），提升 PMC 的数据透明度与决策效率；

强化风险复审：定期评估已识别风险并执行缓解措施（RSKM），形成闭环；

决策记录规范化：对重大技术与架构决策使用 DAR 模板，收集多方案评估数据。

# 4过程改进与改进计划

## 4.1 当前成熟度级别与改进目标

当前成熟度：基于自评结果，本项目的软件过程处于 Level 1（初始级）与 Level 2（已管理级）之间，已落实的 PA 有：

Level 2：需求管理（REQM）、配置管理（CM）

Level 3：无任何 PA 完全达标

**改进目标：**

短期（3 个月内）：全面达成 Level 2 要求，使所有五个 Level 2 PA 达到“已实施”状态；

中期（6 个月内）：在 Level 2 基础上，启动 Level 3 PA 的落地，尤其是组织过程定义（OPD）和组织培训（OT）；

长期（12 个月内）：实现 Level 3 完整覆盖，并开始向 Level 4（已量化管理级）过渡。

## 4.2 改进策略与措施

**项目规划与进度管理：**

采用简单的甘特图工具（如 Excel 或线上开源模板）制定关键里程碑（需求评审、模型训练、系统测试）；

每周召开一次小组例会，汇报各自进度和遇到的问题，并将会议纪要归档。

**项目监控与反馈：**

在 Git 仓库中约定分支策略与提交规范，使用 Issue 功能跟踪任务完成情况；

利用持续集成（GitHub Actions 等免费服务）自动运行基础单元测试，确保新代码不破坏已有功能。

**供应商与第三方服务管理：**

对 DeepSeek API 的使用，整理接口文档并与团队共享；

在 README 中记录调用示例和常见异常处理方法，确保后续使用者能快速上手。

**过程文档化与共享：**

撰写简明的《项目开发流程说明》，包括需求提交、代码评审、测试发布等步骤；

建立项目 Wiki 或共享文档库，分类保存设计文档、测试报告、部署步骤。

**团队培训与知识沉淀：**

每两周安排一次技术分享，由小伙伴轮流讲解自己负责的模块或遇到的技术难题；

汇总培训要点，形成“学期项目成长手册”，方便新成员快速了解项目背景与技术栈。

**风险识别与决策记录：**

在项目 Wiki 中设立“风险与决策”页面，列出可能的技术、进度或资源风险，并记录应对方案；

对于重大技术选型（如 ViLT 模型 vs. 传统 CNN），简单比较性能与实现难度并记入文档。

## 4.3 实施步骤与时间安排

| **步骤** | **时间** | **负责人** | **产出物** |
| --- | --- | --- | --- |
| 制定项目流程说明 | 第1周 | 项目负责人 | 《项目开发流程说明》 |
| 设置甘特图与里程碑 | 第1周 | 全员 | 甘特图文档 |
| 配置 CI 单元测试流程 | 第2周 | 开发负责人 | GitHub Actions 测试脚本 |
| 建立 Wiki 文档库 | 第2–3周 | 文档负责人 | 项目 Wiki 结构与首批文档 |
| 深度学习模块决策记录 | 第3周 | 算法负责人 | “风险与决策”页面示例 |
| 每两周技术分享 | 第1–12周 | 全员轮流 | 分享 PPT 或 Markdown 笔记 |
| 小范围自评与调整 | 第8周 | 评估小组 | 自评报告与改进建议 |
| 项目总结与经验手册编写 | 学期末 (第16周) | 全员 | 《学期项目成长手册》 |

## 4.4 风险管理与监控

周报与例会：通过周报和例会及时发现进度偏差与技术瓶颈，确保小问题不积累成大风险；

自动测试覆盖：持续跟踪单元测试通过率，设定最低 80% 的覆盖目标；

文档更新：每次流程或接口变更后，确保对应文档在两天内更新；

阶段性回顾：第 8 周和学期末各召集一次“过程复盘”，评估流程执行情况并优化改进措施。