

**SICHUAN UNIVERSITY**

****

**题 目** **软件过程管理期末论文**

**学 院 软件学院**

**学生姓名 何依琳**

**专 业 软件工程**

**学 号 2022141461105 年级 2022**

**指导教师 毌攀良**

**2025年6月1日**

# 一、CMMI模型

CMMI是一种用于评估和改进组织过程能力的模型。它通过定义一系列最佳实践，帮助组织提高软件、硬件、系统工程等领域的开发和管理能力。CMMI 的层次成熟度模型将组织的过程能力分为五个成熟度等级，每个等级代表了组织在过程管理方面的能力水平。以下是 CMMI 的五个成熟度等级的简述：

## 1. 初始级

特点：这是最低的成熟度等级，组织的过程通常是混乱和无序的。项目成功主要依赖于个人的能力和经验，而不是系统的流程。

表现：项目进度和质量难以预测，容易出现延误和质量问题。过程缺乏标准化和文档化，难以复用和改进。

改进方向：组织需要开始建立基本的过程规范，减少对个人的依赖，逐步引入标准化的流程。

## 2. 已管理级

特点：在这个级别，组织开始对过程进行管理和控制。项目的过程被定义和文档化，项目团队能够按照既定的流程执行任务。

表现：项目进度和质量可以通过过程数据进行监控和管理，项目团队能够更好地应对风险和变更。过程的执行结果可以被记录和评估。

改进方向：组织需要进一步优化过程，确保过程的稳定性和可重复性，减少过程中的变异性。

## 3. 已定义级

特点：组织的过程被标准化、文档化，并在整个组织范围内统一。项目团队能够按照标准化的过程执行任务，过程的改进成为组织的共同目标。

表现：项目团队能够高效地执行标准化的过程，过程的执行结果具有较高的可预测性和一致性。组织能够通过过程改进提高效率和质量。

改进方向：组织需要进一步量化过程性能，通过数据驱动的决策来优化过程。

## 4. 量化管理级

特点：在这个级别，组织通过量化的方法对过程进行管理和优化。过程性能被量化为可测量的目标，并通过统计过程控制（SPC）等方法进行监控和改进。

表现：组织能够通过量化数据预测项目的结果，及时发现和纠正过程中的偏差。过程的性能得到持续改进，项目质量更高，成本更低。

改进方向：组织需要进一步优化过程性能，通过持续改进的方法提高过程的效率和质量。

## 5. 优化级

特点：这是最高的成熟度等级，组织的过程不仅被量化管理，还能够通过持续改进实现优化。组织能够快速适应变化，不断创新，以满足客户的高要求。

表现：组织的过程具有高度的灵活性和适应性，能够快速响应市场和技术的变化。过程的性能达到最优，项目质量和效率处于行业领先水平。

改进方向：组织需要持续关注行业动态和技术创新，通过持续改进和创新保持竞争优势。

# 二、个人项目CMMI分析

本项目为一个基于 OpenCV 的人脸识别系统，由本人在五天的实训期间独立完成。系统主要功能包括人脸检测、微笑检测、人脸数据库管理（添加、删除、查询用户信息）以及通过摄像头或本地文件进行图像处理等。项目使用 C++ 语言，结合 Qt 框架进行界面开发，涉及多线程、信号与槽等技术。

CMMI（能力成熟度模型集成）将软件组织的过程能力分为五个成熟度等级，以下从每个等级的关键过程域（KPA）出发，对本项目进行评估。

## 1. 初始级

本人在五天内完成项目，时间紧迫，可能导致部分功能实现不够完善，如人脸匹配的准确率可能有待提高，部分界面交互不够友好。

代码中部分注释缺失或不够清晰，如 camera.cpp 中 openCamera 函数捕获帧失败后的处理逻辑，未详细说明为何要使用递归调用；项目缺少需求文档、设计文档等。

## 2. 已管理级

项目中部分组件有简单的头文件（如 camera.h、checkhandle.h 等）对类的功能和接口进行说明，但整体项目缺乏详细的过程定义文档，如开发流程、测试流程等。本人在开发过程中可能对项目进度有一定的自我监控，但缺乏系统的进度跟踪和质量评估机制，如未明确设定里程碑节点及其对应的任务完成情况。

## 3. 已定义级

项目中不同模块的开发风格和规范存在差异，如部分类的成员变量命名不够规范，有的使用下划线分隔，有的直接拼接；函数的参数命名在不同地方也不一致，未遵循统一的命名规范。

## 4.量化管理级

项目中未明确设定可量化的质量目标和性能指标，如人脸检测的准确率、响应时间等，无法通过量化指标来评估项目是否达到预期目标。

缺乏对过程性能数据的收集和分析，如未记录代码缺陷率、测试覆盖率等数据，无法通过数据驱动的方式对过程进行优化。

## 5. 优化级

本人在项目开发过程中主要关注功能的实现，缺乏对新技术的探索和应用，未主动对项目过程进行持续改进和创新。

项目在五天内完成，未充分考虑后续功能扩展和需求变更的情况，代码的可扩展性和可维护性有待提高，如新增一种人脸检测算法或增加语音提示功能时，可能需要大量修改现有代码。

# 三、修改建议

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 改进领域 | 任务描述 | 负责人 | 时间安排 | 预期成果 |
| 过程定义与标准化 | 制定详细的过程定义文档，涵盖需求分析、设计、编码、测试、部署等各阶段。 | 项目开发者 | 项目复盘阶段（1周内完成初稿） | 形成完整的开发流程文档，明确各阶段任务、交付物和责任人。 |
| 过程定义与标准化 | 参考通用编程规范（如 | Google | C++ | 风格指南），制定统一的代码规范，并对现有代码进行规范调整。 |
| 文档与注释 | 编写需求文档、设计文档、用户手册等，详细说明项目功能、架构设计、使用方法等。 | 项目开发者 | 项目复盘阶段完成需求和设计文档初稿，项目交付前完成用户手册 | 提供完整的项目文档，便于新接手人员快速理解和使用项目。 |
| 文档与注释 | 对现有代码进行注释补充，尤其是复杂逻辑和关键函数，确保每段代码都有清晰的注释说明。 | 项目开发者 | 后续代码优化阶段（1周内完成重点模块注释） | 提高代码可维护性，减少后续开发和维护难度。 |
| 项目管理与监控 | 使用项目管理工具（如 | Trello、Jira | 等）记录项目进度，设定里程碑节点并跟踪任务完成情况。 | 项目开发者 |
| 项目管理与监控 | 引入代码质量管理工具（如 | SonarQube | 等）和测试框架（如 | Google |
| 功能优化与扩展 | 优化人脸匹配算法，提高匹配准确率；考虑引入更先进的算法（如基于深度学习的算法）进行对比和改进。 | 项目开发者 | 后续功能优化阶段（3周内完成初步优化） | 人脸匹配准确率显著提高，用户体验提升。 |
| 功能优化与扩展 | 采用模块化开发方式，将功能划分为独立的模块，降低模块间的耦合度；预留接口以便后续功能扩展。 | 项目开发者 | 后续功能开发阶段（2周内完成模块化重构） | 代码的可扩展性和可维护性增强，便于后续功能扩展和需求变更。 |