## 一、CMMI 层次成熟度模型简述

CMMI（Capability Maturity Model Integration，能力成熟度模型集成）是由卡内基梅隆大学开发的一套软件过程改进框架，旨在帮助组织提升过程能力和项目管理水平。其成熟度模型分为五个层次，从低到高依次为：

**1. 初始级（Level 1: Initial）**

特征：过程无序，依赖个人能力而非规范流程。项目成功主要依赖核心成员的经验和努力，缺乏标准化的管理和文档，进度、质量和成本难以预测，风险控制能力薄弱。

关键过程域：无明确定义的过程，管理活动随机且被动。

**2. 管理级（Level 2: Managed）**

特征：建立基本的项目管理流程，如需求管理、项目计划、配置管理和质量保证。过程可重复，通过跟踪和监控确保项目按计划执行，依赖已定义的规则和工具管理项目，但流程尚未在组织层面标准化。

关键过程域：需求管理、项目计划、项目监控与控制、供应商管理、度量与分析、过程和产品质量保证。

**3. 定义级（Level 3: Defined）**

特征：组织级标准化流程形成，建立了统一的过程资产（如模板、指南、工具），项目执行基于已定义的标准流程，并根据项目特点进行裁剪。强调跨部门协作和过程改进，注重文档化和培训。

关键过程域：需求开发、技术解决方案、产品集成、验证、确认、组织过程定义、组织过程焦点、组织培训、集成项目管理。

**4. 量化管理级（Level 4: Quantitatively Managed）**

特征：通过量化数据对过程和产品进行管理，建立过程性能模型和度量体系，利用统计方法分析偏差并优化。项目目标可通过量化指标衡量，过程能力可预测和控制。

关键过程域：组织过程性能、量化项目管理。

**5. 优化级（Level 5: Optimizing）**

特征：组织具备持续改进的能力，通过创新和新技术引入，主动识别过程中的不足并优化。建立了缺陷预防和过程改进机制，注重长期效率和质量提升。

关键过程域：缺陷预防、技术改进管理、组织革新与部署。

## 二、过往开发过程的软件成熟度评估

在软件项目管理课程的 “医疗管理系统项目” 中，我作为开发者参与了需求分析、架构设计、编码实现和测试部署全流程。我认为我处于管理级,以下从 CMMI 维度分析成熟度：

1. 初始级特征残留（部分环节仍存不足）

需求管理无序：在 项目初期，需求仅通过口头沟通记录，未形成正式文档。用户后期提出 “增加权限管理模块” 时，因需求变更未受控，导致设计反复调整，延期 2 周。

2. 管理级特征显现（具备基础项目管理能力）

基础流程建立：使用 Git 进行版本控制，通过分支管理（主分支 / 开发分支 / 修复分支）避免代码冲突；制定了简单的测试计划，包括单元测试（使用 Google Test 框架）和功能测试，覆盖核心模块 70% 的用例；编写了需求规格说明书和设计文档，明确模块划分和接口定义。

量化管理萌芽：大创项目中尝试用甘特图规划进度，设置里程碑节点；记录缺陷日志，统计模块缺陷密度，但未形成系统性度量体系。

3. 已定义级能力缺失（组织级标准和跨团队协作不足）

流程未标准化：两个项目采用不同的编码规范，文档模板不统一，新人上手需额外学习成本.

过程资产缺失：项目结束后未沉淀可复用的组件，后续项目重复开发类似功能。

## 三、过程改进计划：从 “管理级” 迈向 “定义级”

#### （一）改进目标

建立组织级标准化流程，覆盖需求、设计、开发、测试全生命周期；

降低对个人能力的依赖，提升团队协作效率；

积累过程资产，为后续项目提供可复用的模板和工具。

#### （二）分阶段改进方案

###### 阶段一：流程定义与标准化（第 1-3 个月）

1. 制定组织级过程资产

核心流程文档：

编写《需求管理规范》，明确需求获取（用户访谈 / 用例图）、分析（需求优先级排序）、变更控制（需经评审委员会审批）的流程；

发布《编码规范手册》，统一 C++/Java 编码风格（如命名规则、注释规范、异常处理机制），参考 Google 代码规范并结合项目特点定制；

制定《文档模板库》，包括《需求规格说明书》《架构设计文档》《测试报告》模板，明确各文档的必填项和格式要求。

工具链统一：

采用 Jira 进行项目管理，定义标准工作流（需求提出→设计评审→开发→测试→发布）；

集成 Confluence 作为文档管理平台，要求所有项目文档实时同步，禁止本地存储孤本文档。

2. 建立跨职能协作机制

设立 “需求评审委员会”（含开发、测试、产品经理），所有需求变更需通过评审会审批，避免单一角色决策；

定义 “接口契约文档” 模板，要求前后端 / 模块间协作时必须提前签署接口文档（包含参数定义、返回格式、错误码说明），通过 Swagger 工具生成可视化接口文档。

###### 阶段二：过程实施与培训（第 4-6 个月）

1. 试点项目落地流程

选择新的课程设计项目作为试点，强制应用标准化流程：

需求阶段：通过用户故事卡（User Story）梳理功能，使用 MoSCoW 法则（Must/Should/Could/Won’t）排序；

设计阶段：采用 UML 建模（类图、时序图），架构设计需通过团队评审（重点检查可扩展性和安全性）；

开发阶段：实行 “结对编程 + 代码评审”，每次提交代码前需通过至少 1 人评审，使用 GitLab 的 Merge Request 流程；

测试阶段：制定覆盖度目标（单元测试≥80%，集成测试覆盖所有接口），引入持续集成（CI）工具（如 Jenkins），每次代码提交自动触发单元测试。

2. 团队培训与能力建设

组织 CMMI 基础培训，确保团队理解各成熟度级别要求和流程价值；

开展工具使用培训（Jira/Confluence/Git 协作），定期举办 “最佳实践分享会”，邀请试点项目成员分享经验。

###### 阶段三：过程度量与优化（第 7-12 个月）

1. 建立度量体系:每月生成《过程性能报告》，分析指标趋势，例如：若需求变更率超标，追溯是否因需求分析阶段用户参与不足导致。

2. 过程资产沉淀与复用

建立 “组件库” 和 “缺陷库”：组件库收录可复用模块（如数据库连接池、日志工具、加密组件），附带使用说明和单元测试用例；缺陷库记录历史项目中的典型缺陷（如空指针解引用、SQL 注入漏洞），作为新成员培训案例。

推行 “项目后评审”（Post-Implementation Review）：每个项目结束后，召开总结会，识别可改进点并更新流程文档。