****

**SICHUAN UNIVERSITY**

****

**题 目** **基于CMMI模型的软件过程成熟度评估与改进**

**学 院 软件学院**

**学生姓名 张可心**

**专 业 软件工程**

**学 号 2022141461181**

**指导教师 毌攀良**

**2025年6月18日**

基于CMMI模型的软件过程成熟度评估与改进

2022级 软件工程 张可心 2022141461181

(四川大学软件学院 四川省成都市 610065)

(\*993206750@qq.com)

1. 本文围绕 CMMI（能力成熟度模型集成）中的五个成熟度等级，简要介绍了模型的核心概念与层次特征，并结合本人在过往课程编程大作业、科研竞赛项目等开发实践中的真实经验，对项目过程进行了成熟度自评。评估结果显示当前软件开发过程大致处于 CMMI Level 2（已管理级），具备一定的项目管理能力和协作基础，但在流程标准化、测试体系建设与度量机制等方面仍有较大提升空间。为此，本文制定了一套切实可行的过程改进计划，旨在通过文档规范化、任务管理制度化、测试自动化等措施，逐步向 CMMI Level 3（已定义级）迈进，为未来项目开发提供更有力的过程支持与质量保障。
   1. CMMI 软件过程 项目管理 代码规范
2. **CMMI的层次成熟度模型**

CMMI（Capability Maturity Model Integration，能力成熟度模型集成）是由美国卡内基梅隆大学软件工程研究所提出的一种软件过程改进模型，主要用于评估和提升组织在软件工程过程中的能力水平。

CMMI 采用分级模型，将过程成熟度划分为以下五个层次：

**1.初始级（Level 1：Initial）：**软件过程不可预测、无序，几乎完全依赖于个人英雄主义。成功更多依赖于个别开发者的能力而非系统化流程，常见于没有过程管理经验的开发团队。

**2.已管理级（Level 2：Managed）：**团队开始建立基本的项目管理机制，如进度控制、需求跟踪、配置管理等。流程部分制度化，具有一定的可重复性，但依然不够规范。

**3.已定义级（Level 3：Defined）：**软件过程已标准化、文档化并推广到整个组织。各项目在执行时参考组织定义的标准流程，同时可灵活裁剪。

**4.量化管理级（Level 4：Quantitatively Managed）：**团队不仅遵循标准流程，还能通过度量数据进行量化管理。例如，通过数据监控项目质量、进度、风险，实现可预测性与持续优化。

**5.优化级（Level 5：Optimizing）：**团队以持续改进为导向，利用前面积累的量化经验，系统性识别过程瓶颈，快速响应变化，实现自适应能力与创新。

CMMI 的核心价值，在于它不仅仅评估过程本身的规范性，更重要的是帮助团队构建一套科学、可控、可优化的软件开发机制。

1. **个人项目成熟度评估**
   1. **成熟度定位**

在本科专业课程的学习过程中，我参与了多个开发实践，包括：

**1.课程编程大作业**：如分组开发的前端展示平台，尝试用 Python 构图 + 前端渲染的解耦结构；

**2.创新创业项目**：如“数据驱动的外汇风险信号模型”比赛项目，涉及多模块协同、自动信号生成、后端建模与前端展示；

**3.课程研究性汇报**：例如以石头剪刀布中的博弈分析为主题的运筹学研究，涉及策略建模与结构性表达；

**4.学术写作项目：**如论文《从996到远程办公：IT行业工作模式的范式革命》，在文献管理与协作流程中也反映出一定的软件工程方法意识。

以数据驱动的外汇风险信号模型项目为例：

|  |  |  |
| --- | --- | --- |
| **过程域** | **实践表现** | **成熟度等级** |
| 需求管理 (REQM) | 建立需求跟踪矩阵（功能点→测试用例） | Level 2 |
| 项目计划 (PP) | 使用甘特图管理迭代周期 | Level 2 |
| 技术方案 (TS) | 渐进式融合架构设计文档化 | Level 3 |
| 验证 (VER) | 自动化回测框架（JUnit+回测引擎） | Level 2 |
| 配置管理 (CM) | Git分支策略混乱（未定义发布流程） | Level 1 |

**Figure 1 外汇风险信号系统成熟度定位**

综合来看，我的项目开发经验尚处于学习与实践融合阶段，我认为整体成熟度大致可以评估为 CMMI Level 2（已管理级）。

* 1. **典型Level 2实践表现及不足之处**

1. **实践表现**

在团队协作中注重文档编写与代码版本控制（如使用 Git 进行分支管理与协同开发），能够合理分工并制定基本的进度规划。此外，具备一定的风险识别能力，例如在外汇风险信号建模项目中，中期审查时能够及时发现系统结构混乱的问题，并果断选择推翻重构，体现出初步的过程控制意识。

同时，我也开始尝试将流程进行标准化管理，例如将项目团队明确划分为数据、建模、前端三个功能模块，并使用 Markdown 文档记录接口信息与分工安排，以提升沟通效率与一致性。在配置管理方面，努力将开发环境、依赖项、测试数据等进行清晰标注，从而减少“跑不起来”的问题，有效降低了协作障碍与重复劳动的风险。

1. **不足之处**

1）缺乏正式的流程定义文档：流程多基于临时沟通与即时协商，缺少标准化的任务流与决策机制；

2）缺少过程度量机制：没有用量化数据（如开发效率、bug密度、测试覆盖率）来评估过程效果；

3）测试流程不成熟：多数项目测试不成体系，依赖人工测试或边写边调，未形成自动化机制；

4）人员替换或加入缺乏流程引导：新成员接入时需花费大量口头传授时间，说明知识沉淀不充分。

1. **过程改进措施与计划**
   1. **关键改进措施**
2. **构建组织级过程资产库（Level 3核心）**

在Confluence平台构建统一知识库，系统化归档两类核心资产，标准化ADR模板（记录关键决策如DeBERTa选型依据）和典型错误案例库（如订单簿解析内存泄漏的零拷贝解决方案）。

同步开发Docker环境脚本（封装Python+Torch+Kafka的标准化训练环境），三者在资产复用、故障规避、环境一致性三个维度形成闭环，使后续项目技术复现效率提升80%，环境配置时间压缩至5分钟。

**2.制度化评审机制**

推行代码审查"3+1"规则，即：

审查人 ≥ 3人（含领域专家1人）

缺陷发现率 > 15个/千行代码 → 触发架构重构

**3.需求变更量化控制（Level 4预备）**

定义变更稳定性指标：，控制CSI ≥ 85%（迭代后期禁止重大变更）。

**4.缺陷预测预防（Level 5基础）**

建立历史缺陷数据库训练预测模型。

例如：基于代码复杂度的缺陷预测

def predict\_defect(file):

cyclomatic = calculate\_cc(file) # 圈复杂度

churn = get\_code\_churn(file) # 代码变更频次

return 0.8 \* cyclomatic + 0.2 \* churn # 权重可调

* 1. **阶段规划（以三个月为周期）**

**第1个月：文档规范与流程建设**

编写标准开发流程、接口文档模板、PR模板

团队内部培训一次规范使用方法

**第2个月：测试机制引入**

为当前项目添加至少20%代码的单元测试

探索CI流程集成，如GitHub Actions自动测试

**第3个月：过程回顾与优化**

汇总流程中出现的问题，组织小型复盘会议

根据反馈优化流程文档与工具使用方式

1. **结语**

虽然当前在软件过程方面尚处于起步阶段，但通过借鉴 CMMI 的成熟度模型，我能够更清晰地识别出团队在实践中存在的问题与成长空间。软件过程的成熟并非一朝一夕可以完成，它需要从意识开始逐步构建，结合实践不断调整和优化。希望通过此次评估与规划，能够为未来的项目积累更稳固的过程基础，也为后续走向更高成熟度打好根基。