**《软件过程管理》学期论文**

**一．CMMI层次成熟度模型概述**

CMMI（Capability Maturity Model Integration，能力成熟度模型集成）是由美国软件工程研究所（SEI）开发的过程改进框架，用于评估和提升组织在软件开发、系统工程和服务管理等领域的过程能力。该模型将组织的过程成熟度划分为五个渐进式层级，每个层级代表不同的过程控制水平和优化能力。

Level 1：初始级（Initial）

过程处于无序状态，依赖个人能力。项目常出现进度延误、成本超支和质量失控，缺乏稳定性和可预测性。组织无标准化流程，成功不可重复。

Level 2：已管理级（Managed）

建立基础项目管理实践，包括需求管理、项目计划、配置管理。能跟踪成本与进度，实现项目的可重复成功，但过程仍以项目为单位，未形成组织级标准。

Level 3：已定义级（Defined）

形成组织级标准化过程（OSSP），所有项目采用统一流程框架并允许定制。过程文档化，强调需求开发、技术解决方案等工程实践，质量可控性显著提升。

Level 4：量化管理级（Quantitatively Managed）

基于数据分析驱动决策。使用统计过程控制（SPC）技术管理过程变异，设定量化目标（如缺陷密度≤0.5/千行代码），实现过程性能的预测与优化。

Level 5：优化级（Optimizing）

聚焦持续改进与创新。通过根因分析、技术革新主动预防缺陷，过程能力在量化反馈中螺旋上升，形成自适应改进机制。

表：CMMI各级别核心特征对比

| ****成熟度级别**** | ****过程状态**** | ****管理特点**** | ****关键能力**** |
| --- | --- | --- | --- |
| ****Level 1 初始级**** | 无序、临时性 | 依赖个人英雄主义 | 无标准化，不可预测 |
| ****Level 2 已管理级**** | 项目级可控 | 基础项目管理 | 项目可重复成功 |
| ****Level 3 已定义级**** | 组织级标准化 | 过程文档化定制 | 质量稳定性高 |
| ****Level 4 量化管理级**** | 数据驱动 | 统计过程控制 | 过程性能可预测 |
| ****Level 5 优化级**** | 持续优化 | 创新与预防 | 自适应改进 |

**二．个人开发项目的过程成熟度评估**

在“旅游路线智能推荐”项目（2024年实训）中，我们团队开发了一款酒店查询、路线推荐的小程序。基于CMMI模型评估，该项目成熟度处于Level 1向Level 2过渡阶段，具体表现为：

2.1 Level 1特征体现

需求管理无序：初期仅通过一次会议收集需求，未建立跟踪矩阵。客户在开发中途新增“自定义路线”模块，导致后端架构重构，延期2周。

开发过程混乱：无编码规范，成员随意选用技术栈（如混合使用MySQL与MongoDB），集成时出现数据兼容性问题。

质量保障缺失：测试仅依靠临时的“全员点击测试”，未设计测试用例。上线后因并发漏洞导致预约系统崩溃，紧急修复3次。

2.2 Level 2的初步实践

基础计划控制：使用甘特图制定里程碑计划，每周例会跟踪进度，但未量化偏差（如仅口头报告“进度正常”）。

配置管理尝试：用Git管理代码，但分支策略混乱（多人直接修改main分支），曾因冲突丢失一天工作量。

该项目最终交付的功能完整，但技术债务高昂：缺陷密度达1.2个/千行代码（行业推荐值≤0.5），维护成本超开发成本的30%。评估表明，过程成熟度不足直接导致效率与质量损失。

**三．基于CMMI的改进方案**

针对当前痛点，设计分阶段改进计划，目标在1年内达到Level 3（已定义级），并为量化管理奠定基础。

3.1 过渡至Level 2：规范化管理

需求管控：

采用需求跟踪矩阵（RTM），链接用户需求-设计-测试用例。需求变更需经小组评审并更新RTM，控制范围蔓延。

项目监督：

引入量化指标：进度偏差（SV）、缺陷解决周期。例如设定阈值：SV≥10%时触发预警。

配置管理：

实施GitFlow标准化分支策略：feature分支开发 → release分支测试 → main分支发布，每日自动构建验证。

3.2 提升至Level 3：组织级标准化

过程资产库建设：

编制《前端开发规范》《API设计手册》，要求所有项目复用组件库（如表单校验工具），减少重复编码错误。

工程过程强化：

需求开发：原型设计阶段嵌入用户场景测试（如使用FigJam收集反馈）。

技术解决方案：架构决策记录（ADR）模板化，关键设计需3人评审。

测试管理：分层测试策略（单元测试覆盖率≥70%，API自动化测试覆盖率≥90%）。

知识传递机制：

每月举办“缺陷复盘会”，分析TOP3缺陷根因并更新检查表，避免重复问题。

3.3 Level 4~5的预备实践

度量体系设计：

定义核心指标：需求稳定性指数（＜20%变更）、缺陷逃逸率（＜5%），使用SonarQube自动采集。

改进实验机制：

在非关键项目中试点创新（如低代码平台构建管理后台），评估效率提升效果后推广。

**五．总结**

CMMI框架的系统性揭示：从无序操作到量化优化需逐级进化。本次评估暴露个人项目在需求管理、质量保障的薄弱点，亟需建立组织级标准与量化机制。改进计划以工具规范化→过程标准化→数据资产化为路径，目标不仅为认证升级，更为形成可复用的工程能力。在软件开发日益复杂的背景下，过程成熟度将成为团队响应速度与交付质量的核心杠杆。