CMMI 简述、项目评估与过程改进

2022141461008 江浩睿

一、CMMI 层次成熟度模型简述

CMMI（Capability Maturity Model Integration）是一套用于提升组织过程能力的成熟度模型，其核心是五级成熟度模型，每个级别代表过程管理和改进能力的显著提升：

初始级（Level 1: Initial）：

特点：过程是临时的、混乱的。成功依赖个人能力和英雄主义。过程不可预测，难以重复。

状态：无稳定环境，项目常超预算、超时。

管理级（Level 2: Managed）：

特点：项目级过程受控。能对需求、计划、进度、质量进行管理。过程在项目层面可重复。

关键过程域（KPAs）：

需求管理，项目计划，项目监督与控制，供应商协议管理，度量和分析，过程和产品质量保证，配置管理。

定义级（Level 3: Defined）：

特点：组织拥有标准过程集，项目根据标准裁剪使用。过程在组织层面一致、可重复。强调过程资产积累。

关键过程域：

需求开发，技术解决方案，产品集成，验证，确认，组织过程焦点，组织过程定义，组织培训，集成项目管理，风险管理，决策分析和解决方案。

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

特点：使用统计技术管理过程和产品质量。设定量化目标，利用过程性能模型预测结果。

关键过程域：

组织过程性能，量化项目管理。

优化级（Level 5: Optimizing）：

特点：持续改进过程。识别过程弱点，创新技术，预防缺陷。追求增量式和创新式改进。

关键过程域：

组织绩效管理，因果分析和解决方案。

模型核心思想：通过逐步建立、制度化、量化和优化过程，提升组织能力、效率、质量和可预测性。

二、“校园集市”项目软件过程成熟度评估

现基于 CMMI 模型，开始评估“校园集市”软件的成熟度：

当前成熟度定位：**CMMI Level 1 (初始级) 向 Level 2 (管理级) 过渡**

Level 1 特征明显：

过程随意：需求常口头变更，缺乏正式记录和追踪。

计划薄弱：初期仅制定粗略甘特图，未细化任务分配和资源计划，导致开发后期赶工。

质量依赖个人：测试依赖开发者自测，未建立系统测试用例。

配置管理不足：虽使用 Git，但分支策略不够完善，曾出现代码覆盖问题。

无质量保证机制：未进行代码评审或定期质量检查。

Level 2 的初步尝试：

基础项目管理：制定了初始项目计划。

基本配置管理：使用 Git 进行版本控制。

部分需求记录：核心功能有用户故事描述。

评估结论："校园集市"项目具备一定管理意识，但在需求管理、项目监控、质量保证、配置管理的规范性和一致性上存在显著不足，处于 CMMI Level 1 并开始触及 Level 2 的部分实践，整体成熟度偏低。

三、过程改进建议与计划

需求管理 (Requirements Management)：

改进措施：

使用工具（如 GitHub Issues）记录所有用户需求（故事/用例）；

为每个需求条目建立唯一标识符；

创建简单需求跟踪矩阵（RTM），追踪需求来源、状态（待办/进行中/已完成/已测试/已验收）、实现代码/测试用例；

明确需求变更流程：提出，讨论评估，决策，更新需求文档和 RTM。

预期效果：需求清晰可追溯，变更受控，减少遗漏和误解。

项目计划与监控 (Project Planning & Monitoring & Control)：

改进措施：

详细计划：基于需求分解 WBS（工作分解结构），估算任务工时，明确负责人和截止日期；

工具：GitHub Projects 看板/Jira；

进度跟踪：每周例会检查任务板状态，识别阻塞项。使用燃尽图/燃起图可视

化进度偏差；

风险管理：建立简易风险登记册，定期评审；

状态报告：每周提交简洁进度报告（包括完成项、进行项、问题/风险、下周计划）。

预期效果：计划更可靠，进度透明可控，风险提前识别应对。

配置管理（Configuration Management）：

改进措施：

定义并遵守简单 Git 分支策略（如：main 分支保护，develop 分支集成）；

强制执行 Pull Request 机制：代码合并前需 PR，至少一人评审（关注功能实现、基本规范）；

明确定义版本基线规则；

确保关键工件（如需求文档、设计草图、测试用例）纳入版本库或共享文档管理。

预期效果：代码版本清晰可控，减少集成冲突，提升代码质量。