# 基于CMMI的软件过程改进实践

——以《Sky-Take-Out外卖平台》开发为例

​**​摘要：​**​  
作为软件工程专业大三学生，我在《程序设计实践》课程中与小组成员合作开发了一个外卖平台Sky-Take-Out。本文以该项目为研究对象，系统阐述CMMI成熟度模型理论，结合开发过程中遇到的需求频繁变更、测试覆盖率不足等问题，评估当前过程成熟度等级，并提出针对性改进方案。通过引入需求跟踪矩阵与自动化测试框架，项目迭代效率提升40%，为同类学生项目提供可借鉴的实践经验。

​**​关键词**：CMMI；软件过程改进；敏捷开发；学生项目

## 一、CMMI成熟度模型理论框架

CMMI（能力成熟度模型集成）作为软件工程领域的经典模型，通过五级成熟度描述组织能力的演进路径。在参与中国计算机学会组织的CMMI专题培训后，我深刻认识到该模型对学生项目的重要指导意义。

初始级是大多数学生项目的起点，正如我们在项目初期遇到的困境：需求网页仅用微信群聊记录，导致开发后期因教师要求增加"商家评分功能"而返工两周；测试环节依赖人工逐行检查，上线后仍发现12处逻辑漏洞。这种无序状态印证了CMMI对初始级的定义——"过程不可预测且缺乏控制"。

已管理级是当前项目的实际水平。我们使用Git进行版本控制，通过飞书网页记录任务分配，每周例会跟踪进度。但需求网页存在版本冲突，接口规范未统一，这正对应CMMI强调的"建立基本项目管理规范"。

## 二、项目开发过程成熟度评估

在为期16周的开发周期中，团队经历了典型的学生项目成长轨迹。项目初期采用瀑布模型，需求分析阶段耗时3周完成网页编写，但开发过程中教师三次调整验收标准，暴露出需求管理薄弱的问题。

质量控制环节的短板尤为明显。使用SonarQube进行代码扫描时，发现代码异味占比达23%，单元测试覆盖率仅覆盖核心业务模块的65%。这导致测试阶段耗费大量时间排查低级错误，与CMMI指出的"缺乏量化管理"特征高度吻合。

配置管理方面，虽然使用Docker实现环境隔离，但数据库迁移脚本未纳入版本控制，导致测试环境与生产环境出现3次配置差异问题。这种"部分实践缺失"的状态，正是CMMI对已管理级的典型描述。

## 三、过程改进计划与实施

针对评估中发现的问题，我主导制定了分阶段改进方案。在需求管理环节，借鉴CMMI推荐的"需求跟踪矩阵"，将教师需求网页拆解为36个具体任务，每个任务标注优先级与验收标准。例如"订单状态实时推送"需求，明确要求通过WebSocket实现且延迟不超过500ms。

测试体系重构是改进重点。我们基于Selenium搭建自动化测试框架，针对用户登录、下单支付等核心流程编写28个测试用例。在持续集成阶段，每次代码提交自动触发测试，失败用例通过企业微信通知责任人。这一改进使缺陷发现前移，后期测试周期缩短40%。

在过程资产沉淀方面，团队创建了包含12份模板的知识库。其中《接口设计规范》明确规定RESTful API的版本控制策略，避免出现"同一接口不同版本共存"的混乱。通过定期开展代码评审会，将代码规范遵循率从68%提升至92%。

## 四、改进成效与经验总结

经过三个月的持续改进，项目取得显著成效。缺陷密度从2.1个/千行代码降至0.7个，需求变更响应时间缩短50%。更关键的是，团队形成了"编写网页-执行改进-验证效果"的闭环意识。

这次实践让我深刻体会到CMMI的实践价值。在后续的开发实践中，我们计划引入量化管理工具，建立缺陷预测模型。同时将探索敏捷与瀑布模型的混合应用，在需求快速变化与过程规范之间寻求平衡点。

学生项目的改进往往受限于时间与资源，但通过聚焦关键过程域、建立最小可行改进方案，完全可以在有限条件下实现能力跃升。这段经历不仅加深了我对软件工程理论的理解，更培养了解决复杂工程问题的系统思维。