# 软件过程管理课程学期论文

易虎 2022141461111

## 一、简述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（SEI）提出的，它是用于提升软件开发与维护能力的一整套过程改进模型。CMMI 的目标是帮助组织提升其开发过程的质量和效率，从而更好地交付符合需求的软件产品。CMMI 的核心是将软件过程划分为五个成熟度等级，每个等级代表组织在软件工程管理实践方面的成熟程度。  
  
1. 初始级（Initial）：这一阶段的组织通常缺乏系统化的软件过程，开发工作依赖个人英雄主义，项目成功率依赖个人经验与能力，缺乏可重复性和稳定性。风险管理、需求控制等环节尚未规范，过程容易混乱。  
  
2. 可管理级（Managed）：组织开始建立基本的项目管理流程，能够制定项目计划并对进度和成本进行一定程度的控制。项目团队能够对需求变更进行一定范围内的响应。过程仍依赖具体项目，但已经实现了部分标准化管理。  
  
3. 已定义级（Defined）：组织建立了统一的软件工程过程标准，并在各个项目中广泛应用。该阶段强调过程的标准化与文档化，鼓励知识复用，组织具有较强的可复制性与知识积累能力。项目管理不再局限于单个项目，而是提升至组织级。  
  
4. 量化管理级（Quantitatively Managed）：组织开始使用量化技术来分析和控制开发过程及产品质量。通过数据收集与分析，制定出科学的质量评估和风险预测机制。该阶段的特点是高度可控与可预测。  
  
5. 优化级（Optimizing）：强调持续改进与过程创新。组织不仅仅满足于稳定交付软件，而是主动识别过程瓶颈，通过引入新技术、新工具和改进方案，不断优化开发效率和软件质量。  
  
通过 CMMI 模型，软件企业和开发团队能够系统性地审视自身开发实践，明确改进目标和方向，是当前主流的软件过程改进框架之一。

## 二、评估自身软件过程成熟度

回顾我在本科阶段的主要开发实践，尤其是在本学期编程大作业“学生宿舍管理系统”的设计与实现过程中，我初步评估自身的软件过程成熟度处于 CMMI 的第二级——可管理级（Managed）。  
  
“学生宿舍管理系统”是一个面向高校后勤与学生事务的 Web 应用系统，包含宿舍分配、宿舍调整、楼管信息管理等功能模块。项目采用前后端分离架构，前端使用 Vue.js 开发，后端使用 Python Flask 进行 API 构建，并使用 MySQL 存储数据。我在其中承担了系统设计、部分后端编码与测试任务。  
  
我们的小组在开发过程中初步具备了可管理过程的特征：  
- 项目启动阶段制定了分工方案、开发计划与基本进度节点；  
- 使用 Git 进行版本控制，成员提交规范良好；  
- 项目按阶段进行需求梳理、功能实现与测试工作，最终成功部署演示版本。  
  
此外，在参与大创项目“高校健康信息填报系统”以及数据可视化课程项目“汽车行业市场分析平台”中，我也积累了一些管理经验，如需求评审会议、进度检查、可视化报表自动生成等机制的引入。  
  
但综合来看，我的软件过程管理仍存在以下不足：  
1. 缺乏标准化过程文档，如需求规格说明书、测试计划等；  
2. 模块之间协作依赖即时沟通，接口文档不规范；  
3. 单元测试覆盖率不高，部分功能测试依赖手动点击；  
4. 缺少量化度量，难以评估项目质量或成员贡献；  
5. 缺乏过程回顾与知识总结环节。  
  
因此，我的软件过程虽然具备基本的管理能力，但离“已定义级”的规范化仍有明显差距，需在流程、文档、度量、测试等方面系统改进。

## 三、过程改进建议与计划

为进一步提升自身在软件项目开发中的组织能力与过程控制能力，提升至 CMMI 第三级“已定义级”，我制定了以下过程改进策略和具体计划：  
  
1. 流程标准化与制度建设  
- 建立团队标准流程，包括需求分析、设计评审、代码规范、测试流程和上线流程；  
- 为每个阶段制定交付文档模板，如《需求说明书》《功能清单》《测试用例表》等；  
- 将流程制度嵌入到项目例会与任务分配中，确保规范得到执行。  
  
2. 统一编码与接口规范  
- 明确前后端命名规范（变量、函数、模块等）和目录结构；  
- 编写接口文档，使用 Swagger 或 Postman 等工具维护和共享；  
- 设置代码格式化工具（如 Black、Prettier）自动检查提交。  
  
3. 测试与质量控制体系  
- 制定测试流程和职责分配，测试覆盖功能逻辑、异常边界和接口数据；  
- 学习并引入 pytest、unittest 框架，提高单元测试覆盖率；  
- 使用覆盖率工具如 coverage.py，确保关键模块达到 80%以上覆盖率；  
- 定期进行代码评审，采用 SonarLint 等工具做静态质量检查。  
  
4. 度量与反馈机制  
- 设立项目数据记录表格，记录开发工时、Bug 数量、测试结果等指标；  
- 每周开展开发回顾会议，收集团队成员对过程的反馈与建议；  
- 梳理关键瓶颈与问题，逐步建立问题分类与解决方案知识库。  
  
5. 过程文档管理  
- 使用在线协作工具（如 Notion、Confluence）进行文档协作与版本管理；  
- 明确文档责任人，定期进行更新和归档；  
- 所有项目结束后输出完整项目报告和过程回顾文档。  
  
通过上述改进计划，我将逐步由“做中学”的个人模式转向“规范驱动”的组织化协作模式。

## 四、总结

通过本学期《软件过程管理》课程的学习与实践，我更加深入理解了软件开发不仅仅是编写代码，更是一套可控、可优化的工程流程。  
  
CMMI 模型为我们提供了过程改进的阶梯式框架，它强调组织在不同阶段应该关注的重点：从基础管理，到流程制度，再到量化控制与持续优化。对于我个人来说，当前仍处于管理型阶段，但已有提升空间和发展潜力。  
  
通过编程实践如“学生宿舍管理系统”，我清晰认识到目前过程管理中存在的不足，并通过调研、反思和计划制定出适合自己的改进路径。我相信，在今后的课程项目、科研训练和真实工程中，我将以规范流程为导向，不断打磨团队协作与技术质量能力，真正践行“过程决定成败，规范铸就质量”的工程理念。