一、CMMI的层次成熟度模型简述

CMMI（Capability Maturity Model Integration）是一种过程改进框架，用于评估和提升组织的软件开发和管理能力。其成熟度模型分为5个等级，每个等级代表组织在过程管理和优化方面的不同能力水平。

初始级（Level 1）：过程混乱不可控，依赖个人能力，项目结果难以预测。

可重复级（Level 2）：建立基本项目管理，能跟踪进度和成本，类似项目可复制成功经验。

已定义级（Level 3）：组织级标准化流程落地，项目基于最佳实践执行，质量更稳定。

量化管理级（Level 4）：数据驱动决策，用统计方法控制质量，过程可预测和优化。

优化级（Level 5）：持续改进流程，主动引入创新技术，减少浪费并提升效率。

核心演进是从混乱→可管理→标准化→数据驱动→持续优化。

1. 开发经历评估

在已经过去的本科三年里，我通过团队合作或个人完成，参与过各种课程项目以及实训项目的开发，包括：智慧小区管理系统、火车票订票系统、食堂在线点单小程序、数据库管理系统...

结合CMMI层次成熟度评估，我对于这些项目开发经历的等级评估为Level2即可重复级。因为我们已经能完成团队协作开发（如火车票系统、智慧小区管理），有基本的项目管理（任务分配、版本控制如Git），还有可复用部分代码或经验（如多个系统都用了Spring Boot+MySQL）。因此是符合可重复级（level2）的标准的，但由于缺乏标准化流程：不同项目可能用了不同技术栈或开发规范；较为依赖临时决策：需求变更或问题解决较随意（如数据库设计反复修改）；质量保障不足：测试可能依赖人工，缺乏自动化（如单元测试覆盖率低）等三点原因，还不足以向level3靠近，还需改进。

1. 改进计划

1. 建立组织级标准（关键：可复用的最佳实践）

·技术栈统一：

选择固定技术组合（如Spring Boot+Vue+MySQL），减少学习成本。

制定《项目启动模板》（含目录结构、代码规范、API设计）。

·开发流程标准化：

使用Git Flow分支管理，规范Commit Message（如feat: 用户登录功能）。

引入轻量级Scrum（每日站会+迭代回顾）。

2. 强化需求与设计管理

·需求分析模板化：

强制编写用户故事（User Story）或用例图（如食堂订单系统的"用户下单"流程）。

使用Swagger或YAPI管理API文档。

·设计评审：

数据库设计需团队评审（避免智慧小区系统的ER图后期大改）。

3. 提升质量保障能力

·自动化测试：

单元测试覆盖率≥60%（用JaCoCo统计），关键逻辑100%覆盖。

集成测试（如Postman自动化接口测试）。

·代码审查：

使用SonarQube扫描代码坏味道，合并请求需1人Review。

4. 过程资产沉淀

建立知识库：

记录常见问题（如微信支付集成步骤）、技术决策（选型理由）。

项目复盘模板（哪些做得好/不好，如食堂订单系统的并发问题）。

四、具体改进（以"智慧小区管理系统"改进为例）

·需求阶段：

用Axure画原型，输出《需求规格说明书》（含物业、业主角色权限）。

通过用例图、流程图、活动图进行系统设计，保证逻辑通畅，功能涵盖全面。

·开发阶段：

按Git Flow分支开发，每日同步进度。

代码必须通过Checkstyle检查（如命名规范getUserById()）。

团队开发需要定期进行组会沟通，同步开发进度与开发逻辑，保证系统各分支一致性。

·测试阶段：

落地使用，结合使用反馈进行测试与后期升级。

五、总结

在本科阶段的各类开发项目（如火车票订票系统、智慧小区管理系统、食堂点单小程序等）中，我初步体验了团队协作和项目管理的基本模式，符合CMMI 2级（可重复级）的特征：能通过任务分工和版本控制完成开发，但依赖临时决策，缺乏标准化流程和系统化的质量保障。

通过《软件过程与管理》课程的学习，我深刻认识到：软件工程不仅是编写代码，更是通过规范化的过程管理（需求、设计、开发、测试、交付）降低风险、提升效率和质量。

未来在更专业的团队开发中，我将重点关注建立团队级的标准（如代码规范、API设计模板、自动化测试流程），减少随意性；通过代码覆盖率、缺陷密度等指标量化质量，而非凭感觉优化；利用迭代回顾和根因分析持续优化流程。

这门课程让我意识到：优秀的软件=优秀的技术×优秀的过程。未来的职业道路上，我将不仅提升个人编码能力，更致力于推动团队的工程实践成熟度，向高效、可控、可预测的软件开发迈进。