《软件过程与管理》学期论文

学号：2022141461129　　姓名：张雅秋

一、 CMMI 层次成熟度模型

CMMI（Capability Maturity Model Integration，能力成熟度模型集成）是软件工程领域的国际标准模型，用于评估和改进组织的软件开发、服务和管理过程。它是在 CMM（Capability Maturity Model，能力成熟度模型）的基础上发展而来，由美国卡耐基梅隆大学软件工程研究所制定，广泛应用于全球的软件企业、政府机构和大型组织。

CMMI 将组织的过程能力分为 5 个级别，每个级别代表不同的过程成熟度和管理能力：

- **初始级(initial)：**工作无序，项目进行过程中常放弃当初的计划。管理无章法，缺乏健全的管理制度。开发项目成效不稳定，项目成功主要依靠项目负责人的经验和能力，他一旦离去，工作秩序面目全非。
- **可重复级(Repeatable)：**管理制度化，建立了基本的管理制度和规程，管理工作有章可循。初步实现标准化，开发工作比较好地按标准实施。变更依法进行，做到基线化，稳定可跟踪，新项目的计划和管理基于过去的实践经验，具有复现以前成功项目的环境和条件。
- **已定义级(Defined)：**开发过程，包括技术工作和管理工作，均已实现标准化、文档化。建立了完善的培训制度和专家评审制度，全部技术活动和管理活动均可控制，对项目进行中的过程、岗位和职责均有共同的理解。
- **已管理级(Managed)：**产品和过程已建立了定量的质量目标。开发活动中的生产率和质量是可量度的。已建立过程数据库。已实现项目产品和过程的控制。可预测过程和产品质量趋势，如预测偏差，及时纠正。
- **优化级(Optimizing)：**可通过采用新技术、新方法，集中精力改进过程。具备防缺陷、识别薄弱环节以及改进的手段。可取得过程有效性的统计数据，并可据此进行分析，从而得出最佳方法。

二、 软件成熟度分析

软件介绍

我的大二下学期的数据库课程编程大作业是公共图书馆管理系统，两个人一组，开发周期是两周。

公共图书馆管理系统是一个用于管理图书馆日常运营的软件系统。其主要功能包括图书的库存管理、读者借阅与归还服务等，分为用户端和后台管理端。该系统旨在提高图书馆的工作效率，提供更好的读者服务体验。

我们使用的数据库管理工具为 Navicate，关系型数据库管理系统为 SQLite，开发平台为 QT。 其中我负责数据库设计、前端界面设计，另一个同学负责数据库设计、后端功能实现。

成熟度分析

在开发公共图书馆管理系统的过程中，我们的团队表现出的软件过程成熟度大致处于 CMMI 初始级。由于这是一个课程大作业，项目规模小周期短，我们的开发过程较为随意，缺乏系统化的流程管理和质量控制措施。

虽然我们进行了简单的任务拆分，但这种分工更多是基于个人兴趣而非系统分析，且没有形成正式的文档记录或可追溯的任务分配方案。在需求管理方面，我们通过几次临时讨论就确定了系统功能，没有建立完整的需求文档，也没有对需求进行优先级排序或变更管理。在开发实施阶段，我们的编码过程完全依赖个人编程习惯，没有统一的编码规范，也没有进行代码审查，这使得代码质量依赖个人能力。前端界面的开发也缺乏原型设计，而是在编码过程中不断调整。最后的测试部分，我们仅进行了基本的系统功能测试。

项目管理方面，整个过程我们没有使用任何专业的项目管理工具，进度跟踪完全依靠口头沟通。虽然最终完成了系统的基本功能，但这种高度依赖个人能力的开发模式使得项目质量存在较大不确定性。这种开发方式在小型课程项目中尚可应付，但在更复杂的实际项目中会带来严重风险。

三、 改进计划

针对我们开发公共图书馆管理系统时暴露出的过程管理问题，结合课程项目的实际约束（两人团队、两周周期），可以制定一套切实可行的渐进式改进方案。

首先应该在项目启动阶段投入更多时间进行规范化准备，虽然不需要像商业项目那样编写冗长的文档，但至少要建立一份简明的需求清单和功能规格说明。我们可以用 1-2 个小时进行头脑风暴，使用 Markdown 格式记录核心功能点、用户角色和关键业务流程，比如明确区分读者和管理员的操作权限，列出必须实现

的借阅、归还、查询等基本功能，并确定哪些是核心功能哪些是可选功能。这样的小型需求文档既不会耗费太多时间，又能为后续开发提供明确依据。

在技术实施层面，我们需要改进开发流程但不过度工程化。数据库设计应该先花 1-2 小时绘制简化的 ER 图，可以使用在线工具如 draw.io 快速完成，而不是直接跳转到 SQL 实现。前端开发应当先制作低保真原型，可以用纸笔草图或 QT Designer 快速勾勒界面布局，确保双方对 UI 交互达成共识后再编码。代码管理方面，虽然两人协作对 Git 要求不高，但至少要规范提交信息并保持每日同步。测试环节可以采取"开发即测试"的策略，每完成一个功能模块就立即进行交叉测试，即我测试搭档的后端接口，她测试我的前端交互，这样能及早发现问题。

项目管理可以引入轻量级但有效的管控措施。比如使用飞书软件制作简易的项目看板，列出待办、进行中和已完成的任务，每天花 10 分钟同步进度。风险管理的重点应该放在技术难点预判上，比如提前确认 QT 与 SQLite 的连接方式、分页查询的实现方案等可能卡壳的环节。

考虑到这个软件项目为课程作业，规模较小、开发周期较短，所以我认为所有改进措施的时间投入都应该控制在 2 小时以内，确保在两周周期内可完成。通过这些有针对性的调整，可以在不大幅增加工作量的情况下，显著提升开发过程的可控性和产出质量。