**软件过程管理学期论文**

**CMMI层次成熟度模型将组织的过程成熟度分为五个等级：**

初始级（Level 1）：过程无序，项目成功靠运气和英雄行为，项目易失败，产品质量不稳定，缺乏管理和控制。

管理级（Level2）：项目过程可管理，有基本项目管理过程和要求，能对项目进度、成本和质量进行监控，确保项目按计划执行。

定义级（Level3）：组织有标准过程，项目依此定制过程并执行，过程文档化，项目组能共享组织的资源和经验，提高生产效率。

量化管理级（Level4）：过程性能被量化管理，基于数据分析对过程进行优化和调整，提高过程的可预测性和稳定性。

优化级（Level5）：持续改进过程，对新思想和新技术保持开放，主动发现和解决问题，提升组织竞争力。

**过往编程项目评估：**

以一次程编程大作业的图书馆管理系统举例：代码没有使用版本控制系统（如Git）进行管理，无法追踪代码的历史版本；缺乏详细的开发文档，包括需求文档、设计文档、测试文档等，不利于后续维护；没有任何自动化测试代码，测试覆盖率低，无法保证代码质量和功能的正确性；开发过程不够规范，没有明确的需求分析、设计、编码、测试和部署阶段。因此该课程编程大作业处于CMMI中的初始级，其余做过的课程编程大作业也都处于初始级这个级别。

**改进计划：**

为了提高该项目的软件过程成熟度，可采用以下办法：

1. 实现版本控制：可以通过git来实现版本控制，从而方便地对代码的每一次变动进行记录和追溯。通过创建不同的分支，能够毫无顾虑地尝试新的功能或进行代码重构，即便这些尝试最终未能达到预期效果，也不会对项目的主分支造成任何影响。此外，如果项目后续成为团队协作的项目的话，Git 能够轻松应对多人协作场景，确保代码合并过程有条不紊。
2. 编写开发文档：可以在项目目录下专门开辟一个用于存放文档的文件夹，将项目开发过程中的各类文档以 Markdown 格式撰写并存放于此。从项目初期的需求分析，到设计阶段的架构规划，再到开发过程中的功能说明以及最后的测试报告，都能通过 Markdown 文档进行详尽的记录和阐述。这些文档不仅能够为开发者自身提供清晰的开发思路和历史记录，还能在需要向他人展示项目细节或进行项目交接时，提供全面且易于理解的参考资料。
3. 提高测试覆盖率：选取适合项目技术栈的测试框架，并依据项目功能模块编写相应的测试用例。通过模拟各种正常及异常的使用场景，全面验证系统功能的正确性和健壮性。每次对代码进行修改或新增功能后，重新运行测试用例，查看测试结果，依据反馈及时调整代码和测试用例，确保两者之间的高度匹配。逐步提升测试覆盖率有助于及早发现潜在的代码缺陷和逻辑漏洞，为项目的长期维护和发展奠定坚实基础。
4. 规范开发流程：如果后续该项目演变为了团队协作的项目，便需要规范开发流程。可以采用敏捷开发方法，将项目分解为多个迭代周期，每个周期内完成特定功能模块的开发。借助Jira或Trello等工具对任务进行分配、跟踪与管理，确保每个团队成员明确自身职责与任务进度。在开发过程中，需遵循严格的代码审查机制，利用Pull Request（PR）功能对代码变更进行审查，团队成员应充分沟通、协作，及时解决代码中的潜在问题与缺陷。同时，确保代码质量与项目稳定性。定期召开团队会议，以便及时沟通项目进展、解决遇到的问题并总结经验教训，从而不断提升团队协作效率与项目开发质量。