

**《软件过程管理》学期论文**



|  |  |
| --- | --- |
| 学 院： | 软件学院 |
| 专 业： | 软件工程 |
| 学生姓名： | 尹力 |
| 学 号： | 2022141461007 |
| 指导老师： | 毌攀良 |
| 评阅意见： |  |
|  |  |
|  |  |
|  |  |

二零二五 年 六 月 七 日

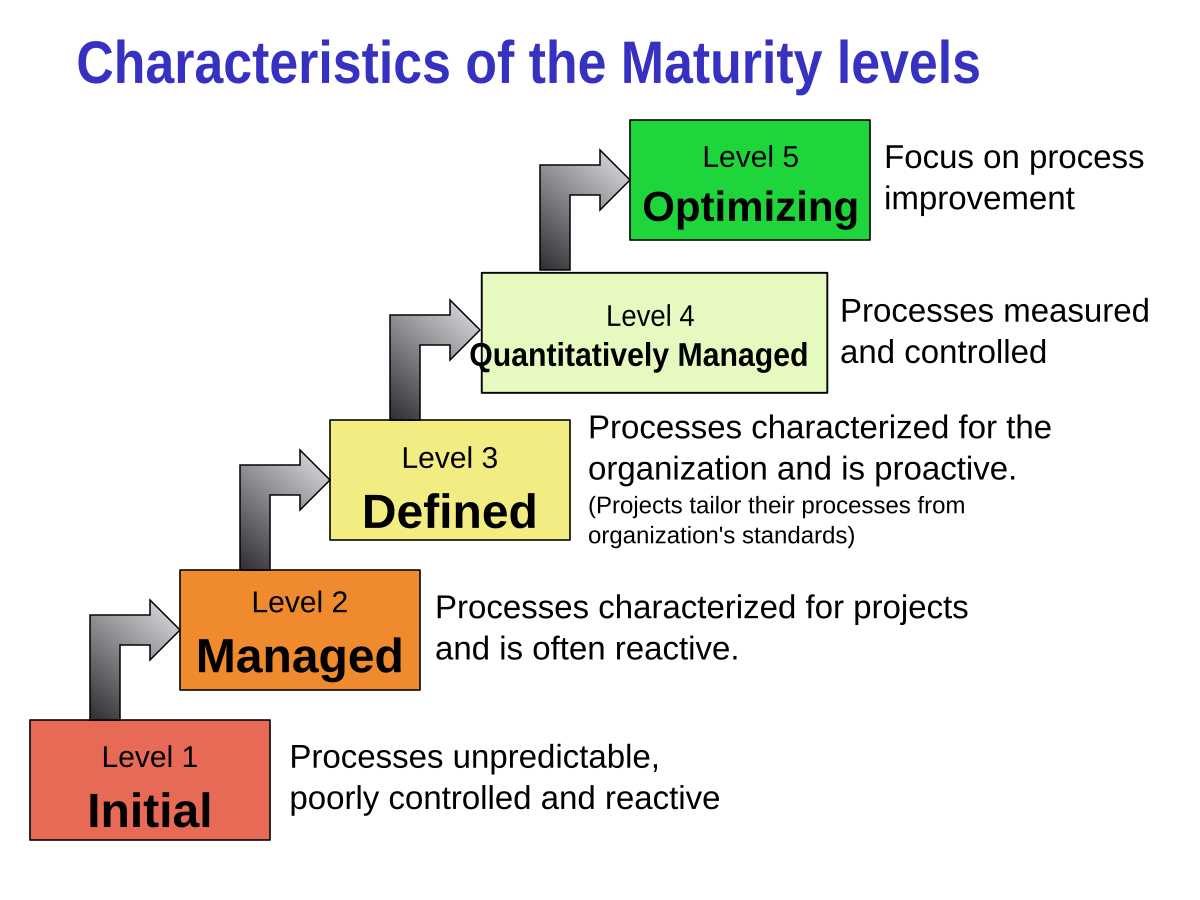
**基于CMMI模型的软件过程评估与改进研究**

**一、引言**

随着软件工程的日益成熟，企业和团队对软件开发过程的规范性、可控性和高效性提出了更高的要求。CMMI（Capability Maturity Model Integration，能力成熟度模型集成）作为当前国际上广泛应用的软件过程改进模型之一，为组织提供了一套系统的方法来评估和提升其软件过程能力。本文将简要介绍 CMMI 的层次成熟度模型，并结合我在大学期间参与的多个开发项目，分析其当前所处的软件过程成熟度水平，并据此制定相应的改进计划。

**二、CMMI 层次成熟度模型简述**

CMMI 是由美国卡内基梅隆大学软件工程研究所（SEI）提出的一个模型体系，主要用于指导软件过程改进。CMMI 将软件过程的成熟度分为五个等级，每一级都建立在前一级的基础上，逐步引导组织从无序到规范再到优化。具体如下：



**Level 1：初始级（Initial）**

该阶段的软件过程是不稳定的、不可预测的，往往依赖于个人的能力和英雄主义精神，缺乏标准化和规范。项目成功主要靠经验丰富的个体，存在很高的失败风险。

**Level 2：已管理级（Managed）**

组织建立了基本的项目管理机制，如进度控制、需求管理等，能够确保项目按计划进行。过程仍然是项目层面进行，组织层面尚未形成统一规范。

**Level 3：已定义级（Defined）**

组织建立了统一的软件过程标准，并在所有项目中推广使用。项目会根据组织级标准进行裁剪，过程具有一定的可重复性和可度量性。

**Level 4：量化管理级（Quantitatively Managed）**

组织开始系统地收集过程执行中的各类数据，并通过数据分析对过程进行控制与优化。项目质量和绩效受到量化的监控与改进。

**Level 5：优化级（Optimizing）**

组织具备持续改进的能力，能够基于过程数据进行根因分析，并主动发现问题，推动组织级别的技术革新和过程优化。

**三、个人开发经验的软件过程成熟度评估**

**项目背景**

在大学期间，我参与过多项软件开发任务，包括《软件项目管理》课程的结课项目--《springboot代码生成器制作工具》、大创课题--《基于深度学习的车道检测方法研究》。这些项目虽然完成度不错，但从过程的角度来看，仍存在明显的提升空间。以下从五个维度对我在《springboot代码生成器制作工具》这个项目的开发过程进行评估。

**评估结果：CMMI Level 2（已管理级）**

**1. 项目计划管理**

在多个项目中，通常会制定一个初步的开发计划，例如里程碑时间节点、模块划分和分工。然而，计划多为临时制定，且缺乏系统工具支持（如 Jira、甘特图等），计划执行过程中容易出现脱节或延误。

**2. 需求管理**

我们能够在开发前与用户或指导老师进行基本的需求沟通，并形成简单的需求说明文档。然而，缺乏需求变更管理机制，需求常在开发中期发生变化，影响整体进度。

**3. 配置管理**

团队通常会使用 Git 进行代码版本控制，具备基本的分支管理和合并规范。但对于数据库脚本、配置文件等资源的管理较为薄弱，常存在不同成员本地环境不一致的问题。

**4. 质量保证**

过程中会编写单元测试，但整体测试覆盖率低，主要以手动测试为主，缺少系统的测试计划和测试用例。也未设置持续集成或自动化测试流程。

**5. 过程记录与复盘**

项目结束后，团队会进行简单的经验总结和代码整理，但缺乏正式的回顾会议和过程度量数据，难以推动持续优化。

综上，我的开发实践虽然已经具备一定的项目管理意识，但在统一规范、过程定义、数据驱动等方面仍较为欠缺，整体处于 CMMI 的第二级水平。

**四、过程改进方向与计划**

为了逐步提高软件开发过程的成熟度，向 CMMI 第三级迈进，我制定了如下过程改进目标与计划：

**改进目标：达到 CMMI Level 3（已定义级）**

**改进计划：**

一、构建统一的开发流程文档制定通用的项目开发流程模板，包括需求分析 → 设计 → 编码 → 测试 → 发布 → 回顾六个阶段，每个阶段明确所需产出（如需求文档、设计图、测试报告）。使用 Notion 或 Typora 进行过程记录与文档编写。

二、建立需求评审与变更管理机制在开发前召开需求评审会议；开发过程中通过 GitHub Issues记录和追踪需求变更。每个功能模块开发前必须确认需求冻结，并设置需求变更审批流程。

三、加强质量管理与测试流程，统一编写单元测试用例，覆盖核心逻辑。设计测试用例文档，涵盖功能测试、边界值测试等。引入自动化测试工具（如 Vitest、Jest）进行持续测试。并且要求测试覆盖率不低于 60%。

四、标准化配置管理流程，建立开发环境统一规范文档。所有配置文件、数据库脚本等纳入 Git 版本控制。使用 .env 管理环境变量，分离配置与代码。

五、建立正式的项目回顾与过程度量机制，每个项目结束后组织一次回顾会议，记录过程问题、改进建议。统计并记录项目关键数据，如实际开发工时、缺陷数量、需求变更次数。积累经验库，供后续项目参考。

**五、结语**

CMMI 作为一套科学、系统的过程改进模型，不仅适用于大型软件企业，对于我们个人或学生团队来说，同样具有重要的指导意义。通过本文的分析，我认识到自己的软件开发实践在组织性、标准化和过程可控性方面仍有较大改进空间。在今后的开发实践中，我将持续优化个人开发流程，借助工具和文档管理，努力向更高的软件过程成熟度迈进，为将来进入企业或带领开发团队奠定坚实基础。