# 《FuturesLab期货实验室》

# 项目软件过程成熟度与改进计划

## 一、 CMMI的层次成熟度模型简述

能力成熟度模型集成（Capability Maturity Model Integration, CMMI）是一个帮助组织改进其开发过程的成熟框架。简单来说， CMMI 将软件过程的能力划分为五个逐步提升的成熟度等级：

1. 初始级（Initial Level 1）：在这一级别，过程通常是混乱和无序的。组织的成功依赖于个别成员的能力和英雄主义，而非标准化的流程。项目成果和质量难以预测，项目计划和预算常常超支。
2. 管理级（Managed Level 2）：在这一级别，组织已经建立了基本的项目管理规程，能够对项目的成本、进度和功能进行跟踪。需求管理、项目策划、项目监控与控制、配置管理等过程已经建立。过程是“在项目级别上”被计划和执行的，但仍可能是被动响应式的，不同项目间的过程可能差异很大。
3. 定义级（Defined Level 3）：在这一级别，组织已经将有效的软件工程和管理过程标准化，并汇编成一套“组织级的标准过程”。所有项目都基于这套标准过程进行剪裁，形成自己明确定义的过程。组织更加主动，能够从更广阔的视角来理解和改进过程。
4. 定量管理级（Quantitatively Managed Level 4）：在这一级别，组织为软件过程和产品质量建立了定量的目标。通过收集和分析详细的过程性能数据，使用统计和其他定量技术对过程进行控制，使得过程的性能和产出是可预测的。
5. 优化级（Optimizing Level 5）：在这一级别，组织能够通过对现有过程的量化反馈和试点新方法、新技术的创新手段，持续地改进过程性能。其核心是理解并消除常见问题的根源，实现持续的、积极的过程改进。

## 二、《FuturesLab期货实验室》项目简述

我选择的是我这学期选修软件项目管理课程的课程设计：《FuturesLab期货实验室》。

期货交易作为一种复杂的金融投资工具，需要投资者具备深厚的专业知识和丰富的实践经验。我们的项目旨在开发一个轻量化、教学导向的期货模拟交易网站—《FuturesLab期货实验室》，帮助用户快速掌握期货交易的基本知识和操作技能。通过提供实时行情展示、模拟交易、风险管理、实时咨询、数据统计和报表以及社区交流等功能模块，为投资者提供全面、高效且智能化的投资决策支持工具。

## 三、《FuturesLab期货实验室》软件过程成熟度评估

评估结果：综合来看，我们项目的过程成熟度大概在CMMI 1级和2级之间。 虽然已经做到了2级（管理级）要求的很多事情，但执行起来不够稳定，效果和计划的有差别，所以我觉得可以叫它 “不稳定的2级”。

依据如下：

1. 超越了1级（初始级）：我们的团队并非处于完全混乱的状态。我们有明确的《项目计划书》、《需求分析报告》和《详细设计说明书》等，定义了项目目标、范围、交付成果和风险。我们的团队有清晰的组织结构和人员分工。开发过程采用了Scrum敏捷模型，划分了5个冲刺周期，并计划了每周会议。这些都表明该项目已经脱离了完全依赖个人能力的初始级阶段。
2. 具备2级（管理级）的关键特征：
3. 项目策划：我们制定了详细的工作流程和总体计划，包括甘特图、任务分解和明确的里程碑。
4. 需求管理：我们撰写了详尽的《需求分析报告》，定义了用户角色、功能需求和非功能需求，并使用了UML用例图进行建模。
5. 项目监控与控制：我们使用PingCode和Jira进行任务跟踪，通过燃尽图监控进度，并设立了风险登记和应对策略。
6. 配置管理：我们使用了Git进行版本控制，并制定了明确的分支策略（main/dev/feature）。所有关键文档也被纳入管理。
7. 过程不稳定（不是稳定的2级）：
8. 监控与控制不足：尽管有计划，但我们项目最终的执行结果暴露了控制力不足。我们的项目最终还存在以下问题：“风险管理模块未进行开发”，“仍有部分功能性BUG未能修复”，以及“系统整体性能较差”等。这表明，我们虽然能够监控到问题，但未能有效控制项目偏差，导致最终交付成果与计划存在差距。
9. 许多应对措施是“事后”的。例如，我们的进度控制计划中规定“任务延迟＞2天时重新分配资源”，这是一种被动的补救措施，而非主动的风险规避。对“忘记密码”等安全漏洞的修复也是在测试发现后才定位到后端逻辑缺失。
10. 未完全遵循过程：我们的计划中提到了多项质量控制目标，如单元测试覆盖率≥80%，并使用SonarQube、OWASP ZAP等工具。但最终数据库设计存在冗余，这表明代码审查（Peer Review）等质量保证活动没有严格执行或效果不佳。

## **四、软件过程改进计划**

为了使我们团队的项目能够达到一个“稳定的2级”水平，并为迈向3级（定义级）打下基础，我提出以下改进计划：

1. 建立正式的需求变更控制流程
2. 问题：项目存在需求变更风险，且最终有核心功能模块被放弃开发，这表明需求管理在项目后期可能失控。
3. 改进计划：

* 规范变更流程：任何对基线需求（冲刺开始时确定的需求）的变更，都必须提交《需求变更申请》并与团队成员商量。
* 影响分析：团队成员对变更申请进行评估，分析其对项目进度、成本和技术架构的潜在影响。
* 正式批准：只有经过团队成员批准的变更才能进入开发待办列表。这能将需求变更从“随意发生”变为“受控过程”，巩固2级的需求管理能力。

1. 强化质量保证（QA）体系
2. 问题：项目交付时仍有多个已知BUG 和性能问题，说明质量控制活动未能有效阻止缺陷流入最终产品。
3. 改进计划：

* 强制代码审查：将其作为一项纪律来执行。所有feature分支在合并到dev分支前，必须经过至少一名其他团队成员的审查，并通过GitHub的Pull Request机制进行。这是3级（定义级）中“同行评审”过程域的要求。
* 定义明确的“完成标准”：一个任务的“完成”不仅仅是代码写完，而必须同时满足：代码已提交并通过代码审查；相关的单元测试已编写并通过；在集成环境中测试通过；相关文档（如接口文档）已更新。这将防止“功能已实现但不可用”的情况。
* 分析缺陷根源：对于测试阶段发现的每一个中高级别的BUG，我们需要花少量时间进行讨论，定位其产生的根本原因（如需求理解错误、技术方案缺陷、编码疏忽等），并记录在《风险应对日志》中。这是5级（优化级）的缺陷预防的要求。

1. 实现基础的量化管理
2. 问题：当前的项目监控主要依赖主观判断和任务状态跟踪，缺乏客观数据支撑，导致对项目健康状况的判断不准确。
3. 改进计划：

* 度量关键指标：除了燃尽图，引入并持续跟踪以下几个简单指标：
* 周缺陷发现率和周缺陷修复率：如果发现率持续高于修复率，是项目质量失控的强烈信号。
* 单元测试覆盖率：利用工具（如JaCoCo）自动生成报告，确保覆盖率（可设定一个更实际的目标，如60%）不下降。
* 团队速度：在前两个冲刺后，统计团队平均每个冲刺能完成的故事点/任务数，并以此为依据进行后续冲刺的计划，使计划更贴近现实。
* 建立项目“仪表盘”：每周利用腾讯文档等协作工具，将上述指标汇总成一个简单的仪表盘，在周会上进行审阅。这是向4级（定量管理级）的初步探索。