# 《软件过程管理》学期论文

黄天予 2022141461050

# 简述CMMI的层次成熟度模型

CMMI是一套过程改进框架，它旨在帮助组织改进其软件开发、系统工程、IT服务管理、采购等过程，从而提高产品和服务的质量，降低成本，并按时交付。CMMI提供了一个清晰的路径，指导组织从不成熟、混乱的开发过程逐步走向成熟、可预测、高效的过程。

以下是CMMI阶段式成熟度模型的五个层次：

## 成熟度级别1：初始级

特征：处于初始级的组织，其软件开发过程通常是混乱的、临时的、不可预测的。项目的成功高度依赖于个人英雄主义和少数有经验的开发人员。没有明确定义的标准流程，或者即使有，也往往不被遵循。

问题表现：项目经常超出预算、延期交付，产品质量不稳定，缺陷率高。当关键人员离职时，项目可能会陷入停滞。组织对项目的进度和质量缺乏有效的控制，通常在危机发生后才采取被动应对措施。

过程域示例：在此级别，几乎没有可识别的、可重复的过程域。

## 成熟度级别2：已管理级

特征达到已管理级的组织，其项目管理过程开始变得有计划、可执行、可测量和可控制。项目在项目层面（而非组织层面）对需求、计划、监控、配置管理等方面有了基本的管理。项目经理能够对项目的范围、成本和进度进行初步的估算和跟踪。

改进重点：关注于特定项目的基本管理实践。虽然过程可能不是组织级的标准化，但每个项目都遵循了基本的管理纪律。

过程域示例：

需求管理：建立并维护客户和产品需求的一致性。

项目规划：制定项目计划，包括范围、资源、进度和风险。

项目监控与控制：跟踪项目进展，与计划进行比较，并采取纠正措施。

配置管理：建立和维护产品、服务和组件的完整性。

测量与分析：收集和分析数据以支持决策。

过程和产品质量保证：客观地评估过程和工作产品是否符合既定标准。

## 成熟度级别3：已定义级

特征：已定义级的组织，其软件开发过程是标准化的、文档化的，并在整个组织范围内得到理解和应用。组织拥有一个标准的过程库，项目可以根据其特定需求对这些标准过程进行剪裁和定制。过程的执行不再依赖于个人，而是成为组织资产。

改进重点：从项目层面提升到组织层面，强调过程的标准化、一致性和可重用性。组织开始关注过程的效率和有效性。

过程域示例：除了级别2的所有PA外，还包括：

组织过程焦点：规划、实施和部署组织过程改进。

组织过程定义：建立和维护组织过程资产。

集成项目管理：管理项目和相关利益相关者的集成过程。

风险管理：识别、分析和缓解项目风险。

决策分析与解决：采用系统化的方法进行决策。

组织培训：培养组织人员的技能和知识。

需求开发：产生和分析客户、产品和组件需求。

技术解决方案：设计和实现解决方案。

产品集成：将产品组件组装成集成产品。

验证：确保工作产品满足其指定的需求。

确认：确保产品满足其预期用途。

## 成熟度级别4：量化管理级

特征：达到定量管理级的组织，其已定义的过程是可测量的，并且通过统计和其他定量技术进行控制。组织为关键过程建立了性能基线和模型，能够预测过程的性能，并根据定量目标进行管理。

改进重点：关注过程性能的量化管理和统计控制。通过数据分析来识别过程中的变异，并采取措施减少这些变异，从而提高过程的稳定性和可预测性。

过程域示例：除了级别3的所有PA外，还包括：

组织过程性能：建立和维护组织过程性能的定量理解。

定量项目管理：定量管理项目以实现质量和过程性能目标。

## 成熟度级别5：优化级

特征：优化级的组织致力于持续改进其过程性能。他们通过识别和解决过程中的根本原因缺陷，以及引入创新技术和方法来不断优化过程。组织能够主动预防缺陷，并对过程进行持续的、增量的改进。

改进重点：关注于持续的过程改进和创新。组织能够识别并部署过程和技术改进，以响应不断变化的业务需求和目标。

过程域示例：除了级别4的所有PA外，还包括：

组织性能管理：管理组织性能以实现业务目标。

因果分析与解决：识别缺陷和问题的根本原因，并采取措施预防其再次发生。

# 评估软件过程成熟度

我选择评估之前参加中国大学生游戏开发创作大赛（CUSGA）时完成的游戏项目，该项目为基于Unity平台的2.5D视角地牢闯关游戏，团队由8名成员组成，包含四位程序开发人员、一位音效师、三位在项目中期才招募到的美术人员。

根据CMMI的层次成熟度模型，我对我们团队在该游戏项目中的软件过程成熟度评估为初始级。尽管在项目初期我们尝试进行了一些时间和任务的初步安排，但实际的开发过程却呈现出高度的混乱、不可预测和被动响应的特点。

具体评估原因如下：

## 缺乏明确的需求管理

项目开始时，我们有一个大致的游戏设计文档，但它更多是概念性的，缺乏对具体功能、交互逻辑和非功能性需求的详细定义。

在开发过程中，需求经常发生口头上的变更，且没有正式的记录或评审流程。这导致不同成员对同一功能可能有不同的理解，最终体现在代码中，造成了大量的逻辑冲突和返工。

## 项目规划与监控的不足

虽然我们最初有简单的任务分配和时间表，但这些计划很快就被实际开发中的各种突发情况打乱。

我们没有有效的机制来跟踪每个任务的实际进度，也没有定期进行正式的进度会议或代码审查。经常出现一个人因为自己的需求需要直接去修改他人代码，导致代码相互交错混乱的现象。

## 质量保证薄弱

我们没有系统性的测试计划，也没有编写单元测试或集成测试。测试主要依赖于开发人员的自我测试和简单的功能验证，缺乏独立的质量检查环节。当发现bug时，往往是在使用过程中偶然发现，然后被动地进行修复，而非主动预防。

## 风险管理缺失

在项目初期，我们没有充分评估外部资源的可用性和依赖性，也没有制定相应的风险应对计划。当项目进行中美术资源无法到位时，整个项目进度受到严重阻碍，甚至部分功能代码因缺乏素材而无法测试或集成。

# 提出改进计划

## 建立基础的需求管理流程

编写游戏设计文档或功能列表，明确核心玩法与功能。团队内部评审文档，确保理解一致。建立简单的需求变更流程，变更需经讨论确认并更新文档。以需求文档覆盖率作衡量，减少因需求不明确导致的返工率。

## 实施基本的项目规划与监控

将功能分解为可执行任务并分配负责人，采用短周期迭代规划。使用任务管理工具跟踪任务状态。不时召开例会，同步进度与障碍。

## 引入基础的质量保证活动

定义任务“完成”标准。开发人员自测模块功能。迭代结束时进行集成测试。鼓励团队内部进行非正式代码审查。

## 初步的风险识别与应对

项目初期和迭代开始时进行风险的规划，对主要风险制定简单应对措施如备用资源，定期回顾风险状态。