2022141461100-欧阳楠-学期论文

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）是一套享誉全球的过程改进模型，旨在帮助组织提升其产品和服务的开发、采购及交付能力。CMMI 模型提供了两种表示方法：连续式和阶段式。其中，阶段式表示法，即层次成熟度模型，是更广为人知的一种，它将组织的成熟度划分为五个逐级递进的层次。这种结构化的框架为组织提供了一条清晰的、从混乱无序到持续优化的过程改进路径。

1.1 第一级：初始级

1.1.1 核心定位：

此级别是所有过程改进的起点，其特征是过程的不可预测性。

1.1.2 过程特征：

在初始级，组织通常没有提供一个稳定的环境来支持各项过程。过程通常是临时的、混乱的。项目的成功依赖于个别成员的能力、努力和奉献，而非依赖于一套行之有效的、已被证明的组织流程。当组织面临压力时，既有的流程往往会被放弃。因此，初始级组织的过程能力是不可预测的，常常导致成本超支、进度延迟和产品质量不达标的风险。

1.1.3 关键过程域：

初始级没有定义任何关键过程域。

1.2 第二级：已管理级

1.2.1 核心定位：

在项目层面建立基本的管理纪律，确保项目的需求、策划、跟踪和控制活动

得以实施。

1.2.2 过程特征：

在已管理级，项目能够在一个已定义和被管理的环境中执行。项目的策划和管理是基于成文的政策和流程。项目的需求、工作产品、过程均受到管理和控制。建立了过程基线，并确保其完整性。项目的状态和供应商交付情况对管理层是可见的。过程是计划驱动的。

1.2.3 关键过程域：

1.2.3.1 需求管理

宗旨：管理项目的需求，并确保项目计划和工作产品与这些需求保持一致。

核心实践概述：建立对需求的理解；获取对需求的承诺；管理需求的变更；维护需求的双向可追溯性（即需求与相关工作产品的链接）；识别项目工作与需求之间的不一致性并采取纠正措施。

1.2.3.2 项目策划

宗旨：建立并维护用以指导项目活动的计划。

核心实践概述：估算项目的范围；建立工作分解结构（WBS）；估算工作产品的属性和任务的工作量与成本；确定项目的生命周期模型；安排进度；策划并识别项目的风险；为数据管理、资源管理、干系人交互等活动制定计划。

1.2.3.3 项目监督与控制

宗旨：提供对项目实际进展的洞察，以便在项目绩效偏离计划时采取有效的纠正措施。

核心实践概述：对照计划来监督项目的各项参数（如进度、工作量、成本）；管理纠正措施直至关闭；管理和规避已识别的风险；定期向干系人汇报项目状态。

1.2.3.4 供方协议管理

宗旨：管理从项目外部供应商处采购的产品或服务的获取过程。

核心实践概述：确定采购类型；建立正式的供应商协议；监督协议的执行情况并确保供应商满足协议要求；验收已采购的产品并将其移交给项目。

1.2.3.5 度量与分析

宗旨：开发和维持一种度量能力，用以支持管理层的信息需求。

核心实践概述：根据已定义的管理信息需求来确定度量目标；详细规定度量方法、数据收集与存储机制；执行数据收集；分析度量数据并产出结果；向干系人沟通分析结果。

1.2.3.6 过程与产品质量保证

宗旨：为项目员工和管理层提供关于过程和工作产品的客观洞察。

核心实践概述：客观地对已定义的过程、标准和程序进行遵从性评估；客观地对指定的工作产品和服务进行评估；向相关人员沟通并确保不符合项得到解决；建立客观的质量保证记录。

1.2.3.7 配置管理

宗旨：建立并维护工作产品的完整性。

核心实践概述：识别需要进行配置管理的项目；建立配置管理系统；创建或发布基线（Baselines）；跟踪和控制对配置项的变更；建立配置管理记录，并对其配置项和基线的完整性进行审计。

1.3 第三级：已定义级

1.3.1 核心定位：

将项目级的管理流程进行提炼和综合，形成一套组织级的标准过程，并指导所有项目进行遵循和剪裁。

1.3.2 过程特征：

过程是主动的、已被充分定义的。组织建立了一套标准过程库，项目依据该库，通过剪裁指南，形成自己项目的、已定义的流程。管理和工程活动更加稳定和一致。

1.3.3 关键过程域： (在二级基础上增加)

1.3.3.1 需求开发

宗旨：产出并分析客户、产品及产品组件的需求。

核心实践概述：从干系人处收集需求；将干系人需求提炼为明确的、可验证的产品需求和产品组件需求；建立操作概念和场景；分析需求以确保其必要性和充分性；验证需求以确保其满足用户预期。

1.3.3.2 技术解决方案

宗旨：设计、开发和实现能满足需求的解决方案。

核心实践概述：选择产品组件的设计方案；开发详细设计；实现产品组件；建立产品支持文档。

1.3.3.3 产品集成

宗旨：组合产品组件，产生产品，并交付给客户。

核心实践概述：制定集成策略；确保接口的兼容性；组合产品组件并进行评估；打包和交付产品。

1.3.3.4 验证

宗旨：确保所选的工作产品满足其指定的需求。

核心实践概述：策划验证活动；实施同行评审（Peer Review）；对选定的工作产品执行验证，发现并纠正缺陷。

1.3.3.5 确认

宗旨：证实一个产品或产品组件在预期的使用环境中，满足其预期用途。

核心实践概述：策划确认活动；执行确认，发现并纠正问题。

1.3.3.6 组织级过程焦点

宗旨：策划、实施和部署组织范围内的过程改进。

核心实践概述：确定组织的过程改进机会；策划和协调过程改进活动的部署；部署组织的标准过程和相关资产。

1.3.3.7 组织级过程定义

宗旨：建立和维护一套可用的组织过程资产。

核心实践概述：建立组织的标准过程；建立生命周期模型描述；建立剪裁指南和标准；建立组织的度量信息库。

1.3.3.8 组织级培训

宗旨：发展员工的技能和知识，使其能够有效地执行其角色。

核心实践概述：识别组织的战略培训需求；提供必要的培训；建立培训记录。

1.3.3.9 集成项目管理

宗旨：依据一个集成的、已定义的过程来管理项目，这个过程是根据组织的标准过程剪裁而来的。

核心实践概述：使用项目已定义的过程来管理项目；协调和集成所有相关干系人的参与。

1.3.3.10 风险管理

宗旨：识别潜在的问题，以便在问题发生前采取措施应对。

核心实践概述：确定风险源和分类；识别并分析风险；实施风险规避或缓解计划。

1.3.3.11 决策分析与解决

宗旨：使用一个正式的评估过程，对备选方案进行分析，从而做出决策。

核心实践概述：建立决策分析的指导方针；评估备选方案；选择最优方案。

1.4 第四级：量化管理级

1.4.1 核心定位：

通过收集和分析过程性能数据，对过程和产品质量进行统计和量化管理。

1.4.2 过程特征：

组织和项目建立了量化的目标，并以此作为管理依据。过程性能是可预测的，管理者能基于统计数据识别和处理过程的特殊原因偏差，从而控制项目结果在可接受的量化范围之内。

1.4.3 关键过程域： (在二、三级基础上增加)

1.4.3.1 组织过程性能

宗旨：建立和维护对组织标准过程性能的量化理解，并提供过程性能数据、基线和模型。

核心实践概述：选择需要进行性能分析的过程；建立过程性能度量；建立过程性能基线（PPB）和过程性能模型（PPM）。

1.4.3.2 量化项目管理

宗旨：对项目已定义的过程进行量化管理，以达成项目既定的质量和过程性能目标。

核心实践概述：策划项目的量化管理活动；使用统计学方法监控所选子过程的性能，识别特殊原因偏差；管理项目的过程性能，使其能够达成质量和性能目标。

1.5 第五级：优化级

1.5.1 核心定位：

通过对过程共性原因偏差的分析和解决，以及部署创新的过程改进，实现持续的、主动的过程优化。

1.5.2 过程特征：

组织聚焦于持续的过程改进。组织能够理解并应对过程中的共性原因偏差，主动预防缺陷的发生。通过试点和评估，将增量的、创新的改进方法部署到整个组织，以提升组织整体的过程能力和绩效，使其与业务目标保持一致。

1.5.3 关键过程域： (在二、三、四级基础上增加)

1.5.3.1 组织绩效管理

宗旨：主动管理组织的绩效，以达成其业务目标。

核心实践概述：基于组织的业务目标，分析和选择待改进的绩效指标；部署并评估绩效改进方案。

1.5.3.2 因果分析与解决

宗旨：识别缺陷和其他问题的原因，并采取行动防止它们在未来重复发生。

核心实践概述：选择需要进行因果分析的缺陷或问题数据；利用结构化的方法分析并确定其根本原因；实施针对根本原因的纠正措施，并评估其效果。

二、 过往开发过程的软件过程成熟度评估

在回顾大三期间负责的“面向复杂监控环境的 UAV-Based-WRSN 充电调度算法研究” 大学生创新训练项目时，我以 CMMI 模型为基准，对项目的全生命周期进行了系统性反思。由于该项目在执行过程中主要由我独立承担，其运作模式与我个人的工作习惯和能力紧密交织，在深入剖析后，我发现该项目的过程管理特征高度契合 CMMI 一级的定义，呈现出典型的依赖个人能力驱动、过程随意且成果不确定性较高的特点。

整个项目推进过程中，从需求分析、技术方案选型、代码编写，到最终项目报告的撰写，每一个关键环节都由我独自完成。这种工作模式导致项目的核心知识与进度管控过度集中于个人，缺乏外部监督机制与客观的质量保障体系。一旦在项目执行过程中遭遇超出预期的技术难题或个人突发状况，项目进度与最终成果将面临巨大风险。

尽管项目文件夹中包含进度甘特图等看似规范的文档，但这些文档在实际开发过程中并未真正发挥管理效能，更多是为满足大创项目结题验收要求而准备。项目初期制定的需求规格与实现流程文档，在后续开发过程中，随着研究的深入和算法的优化迭代，并未及时更新。每当有新的想法或发现更优解决方案时，我会直接在代码中进行修改和实现，却忽视了对相关文档的同步更新，使得这些文档逐渐与实际开发情况脱节，沦为“沉睡的文档”。

项目初期制定的甘特图也存在类似问题。虽然在项目启动阶段，我依据预期目标规划了详细的时间节点和任务安排，但在实际开发过程中，我并未严格按照甘特图进行进度监控。这份甘特图更像是一份提交给评审专家的静态展示材料，而非指导日常工作的动态工具。这也正是 CMMI 一级与二级的显著差异所在：二级过程强调通过持续更新和维护的“活文档” 来进行项目管理，而我在该项目中的文档管理方式显然未能达到这一要求。

项目中配置管理的缺失更是其处于 CMMI 一级水平的直观体现。在代码管理方面，我未使用任何专业的版本控制工具，如 Git。代码文件的命名规则仅基于个人理解，功能实现也缺乏模块化设计，往往通过单个脚本实现多项功能。在项目功能扩展或错误修复时，我采用复制文件并添加自定义后缀的方式区分不同版本，这种管理方式不仅难以追溯代码变更历史，还增加了引入新错误的风险。项目的不同版本更多是开发过程中随机探索和灵感迸发的产物，而非经过系统性规划和严格管控的迭代成果。

三、 基于现有成熟度的过程改进计划

基于上述对项目过程管理的全面反思，我深刻认识到，若要提升软件开发能力，必须从当前相对混乱的 CMMI 一级状态，逐步向更具规范性和可重复性的 CMMI 二级管理模式迈进。为此，我制定了一套分阶段、有重点的改进计划。

首要任务是建立规范的配置管理体系。我将摒弃以往通过文件复制管理版本的原始方式，全面引入 Git 进行代码和文档管理。具体而言，我会为每个项目创建专属的 Git 仓库，将所有代码文件、技术文档等纳入版本控制范畴。无论是修复微小的代码漏洞，还是新增复杂的功能模块，每一次变更都将通过规范的“commit 操作进行记录，并添加详细的提交说明。对于项目中的重要里程碑，如核心算法的完成，我将使用 Git 的 “tag”功能进行标记，形成稳定可靠的版本基线，彻底改变过去随意打包归档的管理方式。规范的配置管理不仅能确保项目资产的安全性，还能为后续的开发和维护提供完整的历史追溯依据。

其次，我将着力激活项目计划和需求文档，使其真正成为指导项目推进和监控的有效工具。以现有的甘特图为基础，我计划将其融入每周的工作规划与总结流程。通过建立“个人工作日志”，每周对照甘特图详细记录实际工作进展、遇到的问题以及与原计划的偏差情况。这份工作日志将成为我自我监督和调整项目进度的重要依据。

同时，项目初期的《需求文档》也将被赋予新的活力。在后续研究过程中，每当产生新的功能需求或对原有设计提出修改建议时，我会首先在文档的“变更记录” 部分详细说明变更内容、原因及影响范围，经过审慎思考和评估后，再进行实际的代码实现。这一流程的建立将有效避免需求管理的随意性，逐步形成规范的需求变更管理机制。

最后，我将在项目开发过程中引入初步的质量保证和度量分析机制。鉴于项目主要由个人独立完成，传统的同行评审模式难以实施，但自我评审同样能发挥重要作用。我计划制定一份详细的代码提交前 “自检清单”，涵盖代码注释完整性、变量命名规范性、边界条件处理等关键检查项。在每次提交重要代码前，严格对照清单进行全面检查，以此降低代码缺陷率。在度量分析方面，我将在工作日志中详细记录每项主要任务的实际耗时，并在项目结束后，将实际用时与甘特图中的计划时间进行对比分析。通过这种方式，不断积累项目工作量估算经验，提高未来项目计划的准确性和可靠性。

四、 结论

软件过程改进是一个长期的、持续优化的过程。通过本次基于 CMM 模型的

深度复盘，我不仅清晰认识到过往项目在过程管理中的不足，也明确了未来的改进方向。这次评估并非对自身能力的否定，而是一次难得的学习和成长契机。通过严格落实上述改进计划，尤其是在配置管理、文档激活、质量保证和度量分析等方面的具体举措，我将逐步摆脱依赖个人经验和灵感的“作坊式”开发模式，向更具规范性、可重复性和可预测性的 “工程化” 开发模式转型。这不仅有助于提升未来项目的成功率和交付质量，更为我从学生开发者向专业软件工程师的角色转变奠定坚实基础。