## 学期论文

一、CMMI层次成熟度模型概述

CMMI（Capability Maturity Model Integration） 是一种衡量组织软件开发过程成熟度的框架，分为5个层级：

**初始级（Level 1: Initial）**

过程随意且不可预测，依赖个人能力。

无标准化流程，项目易超期、超预算。

**可重复级（Level 2: Managed）**

建立基本项目管理（计划、跟踪、配置管理）。

能复用成功经验，但缺乏组织级规范。

**已定义级（Level 3: Defined）**

组织级标准化流程，文档化且可裁剪。

强调过程资产库（如模板、指南）。

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

基于数据的决策（如缺陷率、生产率）。

过程性能可预测，质量目标可量化。

**优化级（Level 5: Optimizing）**

持续改进流程，主动预防问题。

技术创新驱动效率提升。

二、个人项目成熟度评估

基于我上学期做的项目“徐霞客游记旅行路线可视化”的实践，结合CMMI模型评估如下：

1、按level1的标准来看，初始级特征显著

首先，过程比较随意：

项目启动时未定义清晰的需求范围。例如，初版视图试图融合路线、时间、景点介绍等多维度数据，导致信息过载。在助教反馈后才被迫拆分为四个独立视图，造成多部分设计返工。

项目推进过程基本是“走一步看一步”，依赖个人探索（如边学边改Vue3/D3.js）。

其次，存在技术债务积累的情况：

直接复用学长基于 Vue2 的代码（Page 9），未评估 Vue3 的兼容成本。重构时遭遇生命周期钩子变更（beforeDestroy → beforeUnmount）、Composition API 适配等问题（Page 10），额外消耗开发时间，并且，冗余代码未清理（如无用逻辑块），导致维护困难。

并且，数据收集仅通过口头指导（Page 9），未形成文档规范，导致景点分类统计出现 “自然景观/人文景观”标准不一致

2、部分达到可重复级（Level 2）

基础管理实践：

存在任务分工结构化的尝试：

明确组长统筹、数据收集、视图实现等角色，确保数据采集与开发并行。

存在风险管理雏形：

利用外部资源降低风险（通过阅读《徐霞客游记》原文来收集数据时，提前参考博客梳理徐霞客游历各地的脉络，以便定位原文，减少数据收集工作量）。

配置管理的尝试：

静态数据统一存储：省份地图使用 GeoJSON 文件，景点介绍嵌入代码，避免后端服务依赖。

3、未达更高级别

缺乏组织级流程（Level 3）：无共享过程资产库（如视图设计规范）。

无量化管理（Level 4）：未统计代码缺陷率、任务完成周期等指标。

综合评估：成熟度介于 Level 1~Level 2，核心问题为过程随意性、技术债务、缺乏量化跟踪。

三、改进计划

第一步：打好基础，让项目“可重复”

这次项目能完成，很大程度上是靠组员的努力和老师的指导，但很多环节比较随意。下次项目一开始就要把基础工作做扎实：

**1、需求管理要更清晰：**

（1）用工具记需求： 不要光靠口头或群聊记录需求。学着用像腾讯文档、飞书文档或者简单的Excel表格，建一个“需求清单”。把老师、助教、我们自己想到的功能点、修改意见都清清楚楚地列进去，标明谁提的、什么时候提的、优先级高低、最后决定做不做。比如这次助教说“视图有点少”，我们就该立刻记下来，后面新增四个视图的决策也记录在案，这样就不会漏掉或者忘记为什么改了。

（2）及时记录需求变更： 需求变更是常事（就像我们中途加了四个视图），但每次变更的原因、内容和影响范围要简单记下来。这样以后复盘或者代码出问题时，能快速知道当时是怎么想的。

**2、规范代码管理：**

（1）Git不能只用来交作业： 这次我们用了Git，但可能更多是为了最后提交。下次要真正用好它。定个简单的规矩：比如建一个main分支放稳定可运行的代码，一个dev分支做日常开发，每个新功能（比如做甘特图）开一个自己的feature-xxx分支。做完并测试好了，再合并回dev。

（2）每次提交代码时，强制自己写清楚这次提交干了什么。不能只写“更新”或“修复bug”。比如写成：“修复：点击景点标签时详细介绍弹窗不显示的bug (#任务ID) ” 或者 “新增：完成省份旅行时间段甘特图视图 (#任务ID)”。这样以后查问题或者看历史记录就一目了然。

**3、定期跟踪进度：**

定期开会： 即使组员都很负责，定期（比如每周一次，线上也行）快速同步一下进度也很必要。用个共享的在线表格或者看板（Trello、飞书项目模板都行），列出这周要做什么、谁负责、做到哪了、有没有卡住的地方（比如数据收集慢了）。这样能及时发现风险，像这次如果数据收集遇到困难，早点发现就能一起想办法。

第二步：尝试“量化”和自动化

**1、简单记录开发数据：**

（1）记一下开发时间： 下次做视图或者核心功能时，简单记录一下大概花了多少个小时。比如“完成山水名胜关联图：前端迁移和调试 - 约25小时”。不用很精确，有个大概感觉就行。积累几次后，就能预估类似功能要多久，安排计划更靠谱，也能知道哪些部分特别耗时（比如和D3.js死磕）。

（2）记录一下遇到的Bug： 项目快结束时，回顾一下总共解决了多少个比较明显的Bug？主要集中在哪个模块（比如地图交互、数据加载）？这能帮我们下次重点防范这些“重灾区”。

**2、引入基础自动化测试：**

这次调试基本靠手动点页面，效率低。下次可以尝试学一下Jest（一个流行的JavaScript测试框架），给项目里那些独立的、核心的工具函数写点简单的单元测试。比如，写一个函数把“万历四十四年二月初三”转换成标准日期格式，那就写个测试，输入这个字符串，看输出是不是正确的1616-02-03（假设的）。