实验四 XP开发方法，DevOps，活动图

实验目的：

1. 了解XP开发方法

2. 了解DevOps

3. 理解项目活动图

实验内容：

1. 阅读XP开发方法文档，理解XP过程工作模型

2. 阅读DevOps文档，了解DevOps

3. 活动图练习

书上练习题2,3（p97-98) 的软件开发项目活动图，找出关键路径。

小组讨论，针对自己项目中的工作进行工作活动分解，分工进行各自合理的工作进度估算，最后汇总绘出项目活动图，找出关键路径。

将小组工作文档、进度等文档保存到每个小组选定的协作开发平台上

极限编程（eXtreme Programming，简称XP）是一种流行的软件开发方法论，重点放在提升开发过程的质量和响应性，面对客户的需求变化。它属于敏捷软件开发的范畴，旨在通过实施一系列最佳实践，来缩短开发周期，提高生产效率，并且尽可能降低成本。

1. XP工作模型的核心价值

XP工作模型基于四个核心价值观：沟通（Communication）、简单性（Simplicity）、反馈（Feedback）、勇气（Courage）。这些价值观在实践中具体体现为：

1. 沟通: 团队成员之间频繁沟通，确保信息的透明和准确传递。项目的各方应当经常会面交谈，包括开发人员、客户以及其他利益相关者。
2. 简单性: 在设计和实现功能时选用最简单的方法。这并不意味着能够工作的最简答案，而是最合适的简单解决方案，尽量减少将来修改的成本。
3. 反馈: 通过频繁测试和客户评审来获得反馈，持续地检查工作是否满足需求。
4. 勇气: 在面对变化和挑战时，团队需要勇气坚持正确的做事方式。
5. XP实践

为了将这些价值观转化为实际行动，XP提出了一系列具体的实践，包括：

1. 整体团队（Whole Team）: 项目涉及利益相关者，如客户、开发人员和测试人员，全程参与。
2. 规划游戏（Planning Game）: 定期的会议，用于创建和优先排序用户故事（用户需求的简洁描述），并计划即将到来的迭代。
3. 小版本发布（Small Releases）: 快速、频繁地将增量改进的软件版本提供给客户，从而获得反馈并适应变化。
4. 客户现场（On-Site Customer）: 一个或多个真实用户的持续参与，为团队提供即时的需求澄清和决策，确保产品方向一致性。
5. 简单设计（Simple Design）: 任何时候，系统的设计都尽可能保持简洁，没有多余的部分。
6. 测试驱动开发（Test-Driven Development, TDD）: 开发人员编写测试来定义代码应当如何正确执行，然后才写出让测试通过的代码。
7. 持续集成（Continuous Integration）: 频繁地集成所有开发人员的工作，并尽可能早地发现集成错误。
8. 成对编程（Pair Programming）: 两名开发人员共用一台计算机进行编程，一个负责编写代码，另一个负责审阅代码。
9. 集体代码所有权（Collective Code Ownership）: 代码由整个团队共同拥有，任何团队成员都可以改进代码。
10. 可持续节奏（Sustainable Pace）: XP鼓励避免过度工作，保持长期的工作节奏。
11. XP的角色

XP工作模型中定义了几个关键角色，包括：

1. 客户（Customer）: 提供需求，进行计划游戏会议，并为发布作最终确认。
2. 程序员（Programmers）: 实现功能、提供测试并进行成对编程。
3. 教练（Coach）: 确保团队遵循XP实践，有效地促进团队成长。
4. 追踪者（Tracker）: 跟踪项目进度，并协助管理变化。
5. 工作空间和工具

XP团队通常在一个开放的工作空间（称为“牛棚”）中工作，以促进沟通和协作。使用的工具包括：

持续集成工具: 支持代码的频繁集成和版本控制。

重构工具: 帮助开发者安全地进行代码重构。

测试框架: 支持自动化测试的编写和执行。

1. 核心实践

XP的核心实践包括：

持续集成: 频繁地将代码变更集成到主代码库中。

集体代码所有权: 团队成员共同负责代码的维护和改进。

无情重构: 持续改进代码结构，以提高可读性和可维护性。

编程对: 两名开发者共同在一台电脑上工作，以提高代码质量和知识共享。

单元测试: 编写测试代码以验证代码单元的正确性。

功能测试: 客户定义的测试用例，用于验证系统满足需求。

XP模型鼓励透明和反馈，从而使得开发过程更加适应性强和可预测。重要的是，XP不是一套固定不变的规则，而是一套经过不断调整和改进的指南，适应团队和项目的特定需求。通过尊重以上的价值和实践，XP工作模型能够有效地提升软件开发质量，加强团队合作，同时灵活地适应变化。