简介和流程

第 1 讲

鲁赞娜-奇奇扬、乔恩-伯德、皮特-贝内特

助教亚历克斯-埃尔伍德、亚历克斯-科克里恩、卡斯帕-王

单位目的

本单元有助于将 这些知识包装成就业技能组合

为此，我们针对以下与行业相关的问题开展了工作：

- 可迁移的技能管理、协作、沟通
- 平台技能：应用先前的编程语言知识
- 可扩展性技能：工业规模的 "大规模 "开发

软件开发

但您听说过***软件危机***吗?

- 软件越来越大
- 软件越来越复杂
  - 复杂领域（如人类行为）
  - 系统依赖性（如能源和汽车）

- 上市时间比以往任何时候都短

项目失败

阿丽亚娜 5 飞行 501

火星气候轨道器

Therac-25 放射治疗

软件项目为何会失败?

- 糟糕的开发人员--不是主要问题!
- 超出预算
  - 要求不高
  - 要求过高
  - 不必要的要求
- 合同管理
- 最终用户培训

- 业务管理

软件工程

- 软件工程
  - 工程学：运用**经济有效的**方法解决实际问题

    用**科学知识**为**人类造物**
    - 具有成本效益的解决方案：流程和项目管理、合同...
    - 科学知识：建模、证明、测试、模拟、模式
    - 事物=软件

- 人：客户和最终用户

好吧，那我们*该*怎么做呢?

协作工具和技术

- UML 设计：绘制图表，就设计达成共识
- GitHub：共享文档，实现有效协作
- 看板：任务分配和进度监控
- 测试驱动开发：阻止其他人破坏你的代码！
- 鼓励使用其他技术：自由尝试和探索

每周主题

1. 简介 + 概述
2. 敏捷开发
3. 需求工程
4. 设计
5. 软件质量与测试
6. [阅读周]
7. 项目管理
8. 人机交互：定性分析
9. 人机交互：定量分析

10. [复活节假期：3周]
11. 软件工程高级专题
12. 模块关闭

小组工作

- 我们只能希望解决现实世界中的规模应用...

  ....if 我们通过团队合作实现最终目标

- 合作也是一项重要的可迁移技能

- 在任何非同小可的工业项目中，你都将在一个团队中工作（也许是暑期项目！）。

- 这并不容易--所以这一点至关重要
  现在就开始练习！

- 随机分配的 5 人（或 6 人）小组

"品尝棒 "隐喻

- 前面提到的所有技术都非常庞大和复杂
- 我们并不指望你们成为什么都懂的专家
- 目的是让您 "尝尝 "各种美食
- 这样，你至少可以说你已经试过了
- 把这个单元看作 "品尝棒 "的一种方法是
- 在小组工作中，个人可能在某些方面具有专长

软件开发生命周期流程：

要完成哪些任务，顺序如何?

软件开发任务

- 需求分析
- 规划
- 设计：高度和细节
- 发展
- 测试
- 部署

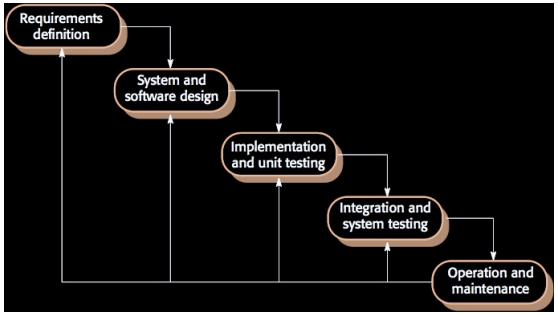
- 运行和维护

这些待办事项以不同的顺序组合在一起，构成了

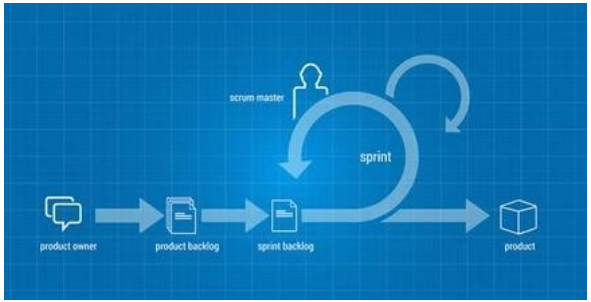
不同的软件开发生命周期流程

软件开发生命周期示例

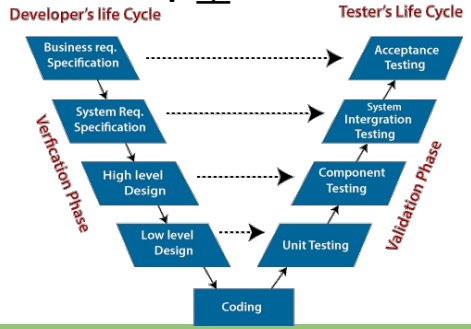
瀑布式



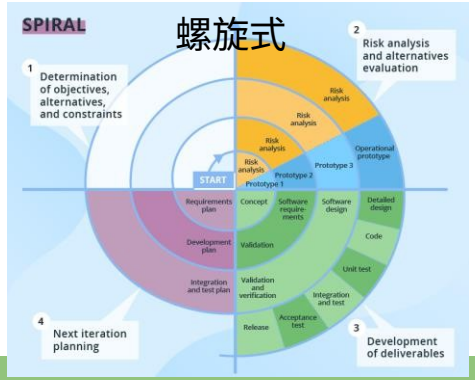
敏捷



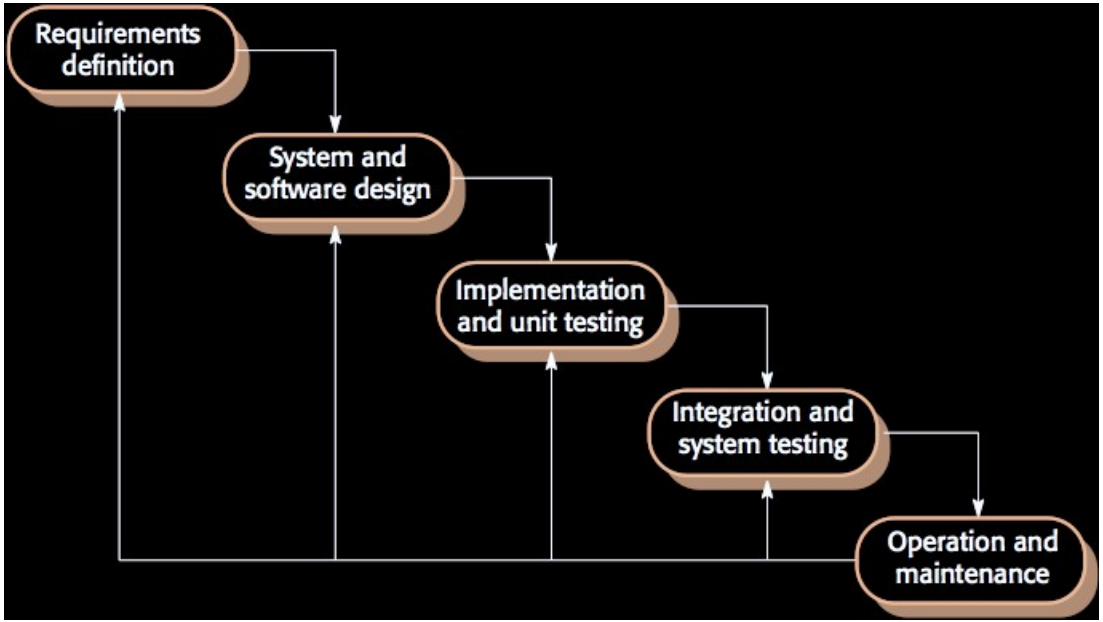
V 型



螺旋式



瀑布式软件开发生命周期



要求：实施什么？

- 系统应该做什么？
- 用户的需求是什么？
- 东道组织的需求是什么？
- 现有系统的互操作性需求是什么？

我们需要考虑功能要素：系统

该怎么做?

还有非功能性要素：系统的质量属性

操作，如安全性、易用性、响应时间等。

设计：如何构建软件?

- 我们应该
  创建?
- 这些要素是如何构成一个系统的?

为什么要设计?

- 有助于决定在何处提出要求
- 系统各部分将如何互动

- 团队成员分工

实施情况

不仅是编程，还有...

- 并行工作和代码共享
- 应用程序接口分区和 "防火墙
- 版本、集成和配置管理
- 开发环境和自动构建

- 自动测试
- 文件和培训材料

核查和验证

- 验证：检查软件/服务是否符合要求、限制和规定。(你做对了吗？）
  - 证明系统符合规格要求
- 验证：这是否符合客户/利益相关者的需求?
  ("你们建造的是正确的东西吗？")
  - 证明系统满足用户需求
- 可以通过验证，但未能通过验证：如果按照规范建造，但规范并未满足用户的需求。

- 涉及检查、审查、评估和测试

瀑布式 SDLC：优缺点

- 简单易懂
- 每个阶段都有明确的结果和过程审查
- 发展阶段逐一进行
- 非常适合要求明确并已达成一致的项目
- 轻松确定关键点开发周期
- 易于对任务进行分类和优先排序
- 软件只有在最后阶段结束
- 高风险和不确定性
- 由于以下原因，错过了复杂性决定的相互依存性
  - 不适合长期项目需求将发生变化的领域
- 该阶段的进展还处于发展阶段，很难衡量

- 整合工作在最后才进行，无法提前发现问题

案例研究：胰岛素泵控制系统

胰岛素泵是一种模拟胰腺（一种内脏器官）运行的医疗系统。控制该系统的软件是一个嵌入式系统，它从传感器收集信息，并控制泵向用户输送受控剂量的胰岛素。

糖尿病患者使用该系统。

糖尿病是一种比较常见的疾病，是指人体胰腺无法产生足够数量的一种叫做胰岛素的激素。胰岛素能代谢血液中的葡萄糖（糖）。糖尿病的传统治疗方法是定期注射基因工程胰岛素。糖尿病患者使用外部测量仪测量血糖水平，然后计算出应该注射的胰岛素剂量。

这种治疗方法的问题在于，所需的胰岛素水平不仅取决于血糖水平，还取决于最后一次注射胰岛素的时间。这可能导致血糖水平非常低（如果胰岛素过多）或血糖水平非常高（如果胰岛素过少）。低血糖在短期内是一种较为严重的情况，因为它会导致暂时性的大脑功能障碍，最终导致昏迷和死亡。但从长远来看，持续的高血糖会导致眼睛损伤、肾脏损伤和心脏问题。

目前在开发微型传感器方面取得的进展意味着现在有可能开发出自动胰岛素输送系统。这些系统能监测血糖水平，并在需要时提供适当剂量的胰岛素。类似的胰岛素输送系统已经用于医院的治疗。

患者。许多糖尿病患者都希望在身体上永久安装这种

系统。

要做的事两人一组

- 讨论为什么瀑布式流程可以很好地适用于本案例研究的 2 个原因，以及为什么它不适用的 2 个原因（10 分钟）
- 课堂讨论（5 分钟）

评论

- 软件工程是什么?
- 什么是软件开发生命周期?
- 您能说出任何软件开发生命周期吗?
- 瀑布式 SDLC 有哪些特点?