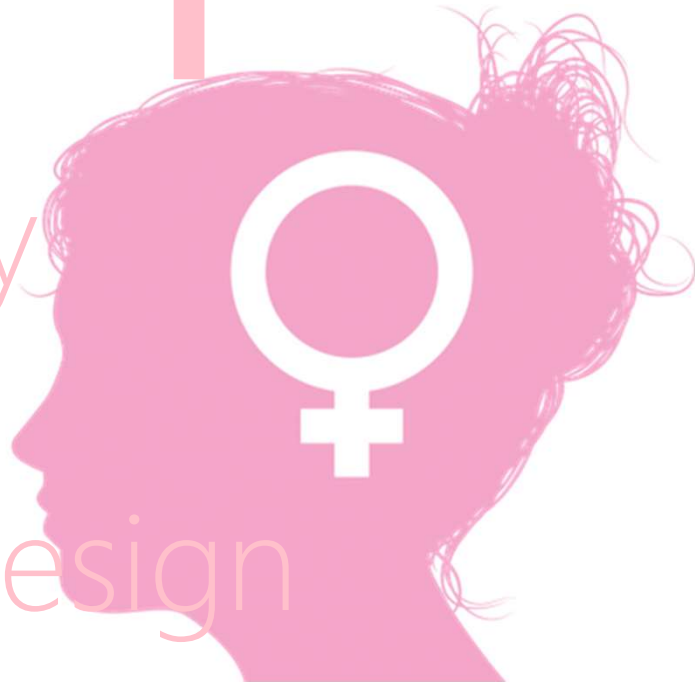
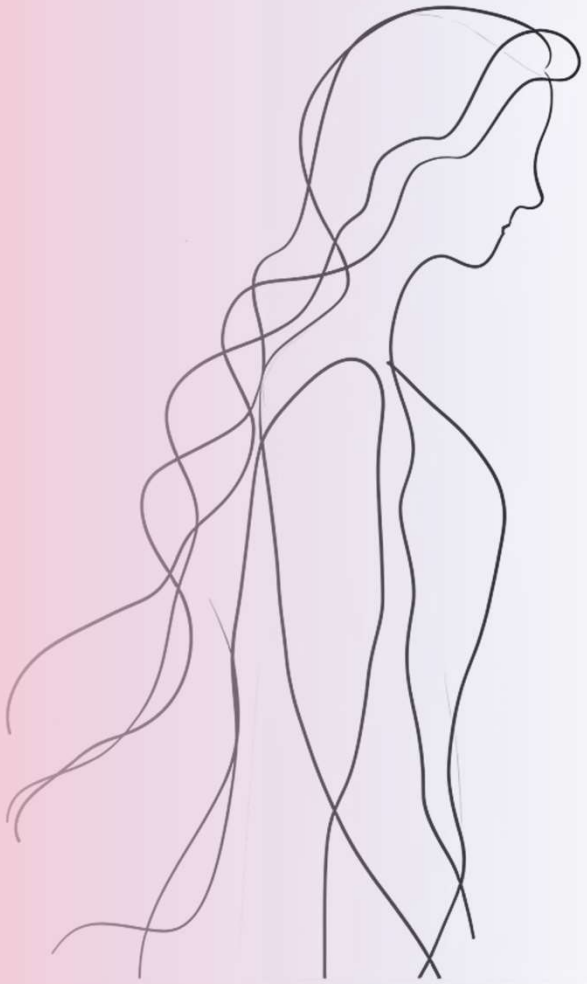
0304

Happy Girl's Day

14:10-16:00

System Analysis & Design





北京交通大學
BEIJING JIAOTONG UNIVERSITY

软件系统分析与设计
System Analysis & Design

M210007B [02]

Jingxin Su

Sunday, March 3, 2024

组队邀约

- 每支队伍 **3-4 人**，共同完成本课程的项目作业。
- 每支队伍选出一位队长，并且还需要起一个**很Coooooool的队名**。
- 作业内容**不会**涉及到代码实现部分
- 在线填写你的队伍信息

  https://docs.qq.com/sheet/DRE5zZHVUbFNTRHFj

项目要求

目标：

- 设计一个有创意的、原创性移动终端应用程序（App）
  - 项目涉及范围包括但不限于能源、健康医疗、环保、养老、智慧交通、智慧城市、智慧物流、与社会创新相关的物联网、食品安全、与社会服务相关的大数据/云服务等

任务：

- 提交典型产品需求&设计文档
  - 产品需求文档（PRD）、软件需求规格说明书（SRS）、设计文档

一千个人眼中有一千个哈姆雷特

系统所有者的视角

学生可以申请课程；选课必须由学生提交申请

系统用户的视角

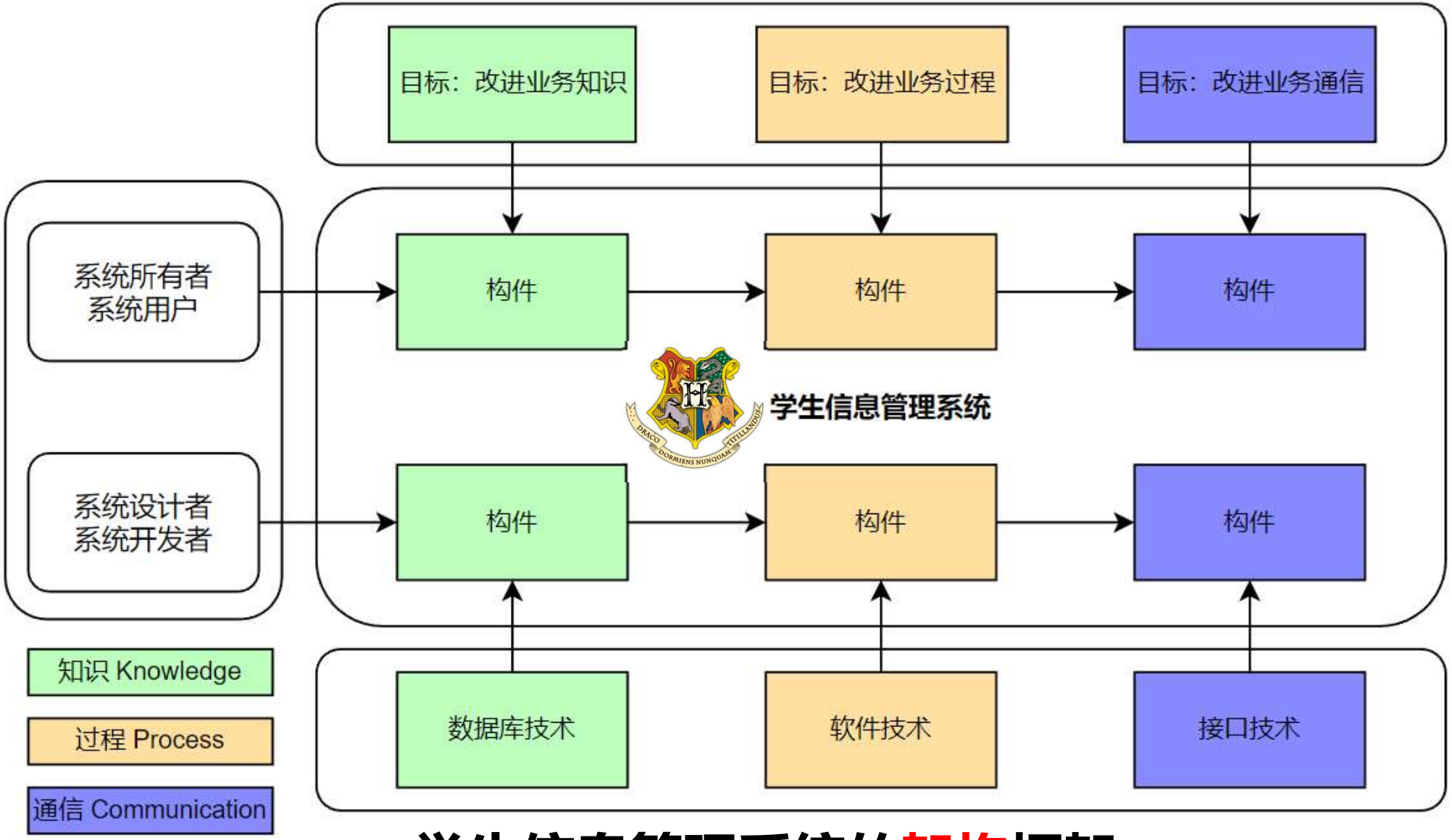
各专业学生可选择其相应的专业必修课程

系统设计者的视角

关系数据库中的 3 个表及其之间关系

系统开发者的视角

CREATE TABLE student（student_id int, ...）



学生信息管理系统的架构框架

“Where is the Life we have lost in living?
Where is the wisdom we have lost in knowledge?
Where is the knowledge we have lost in information?”
— T. S. Eliot 20th century English author 1888 - 1965

我们在生活中失去的生命在哪里?
我们在知识中失去的智慧在哪里?
我们在信息中丢失的知识在哪里?

DIKW Pyramid: Data, Information, Knowledge, and Wisdom

“知识”构件
Knowledge Building Blocks

系统所有者的视角

学生可以申请课程；选课必须由学生提交申请

系统用户的视角

各专业学生可选择其相应的专业必修课程

系统设计者的视角

关系数据库中的 3 个表及其之间关系

系统开发者的视角

CREATE TABLE student（student_id int, ...）

DIKW Pyramid: Data, Information, Knowledge, and Wisdom

“过程”构件
Process Building Blocks

系统所有者的视角

- 事件(Event)：学生提交课程申请
- 回应(Response)：学生在该课程中获得一个名额

系统用户的视角

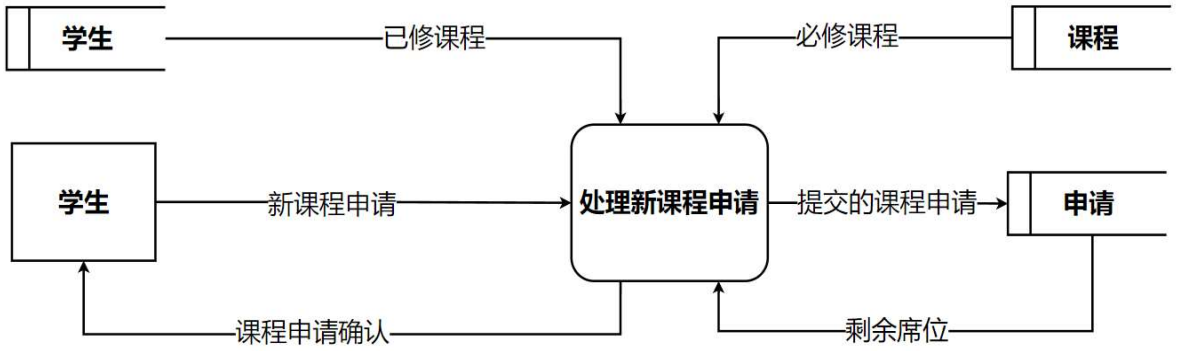
- 带有策略和规程约束的过程需求
  - 只有完成《初级魔咒》的学生才能申请《黑魔法防御课》

系统设计者的视角

- 软件规格说明文档

系统开发者的视角

- 程序设计语言表述“过程”



事件图-处理新的课程申请

“通信”构件
Communication Building Blocks

系统所有者的视角

- 谁可以看到什么?
- 系统会与其他信息系统交互吗?

系统用户的视角

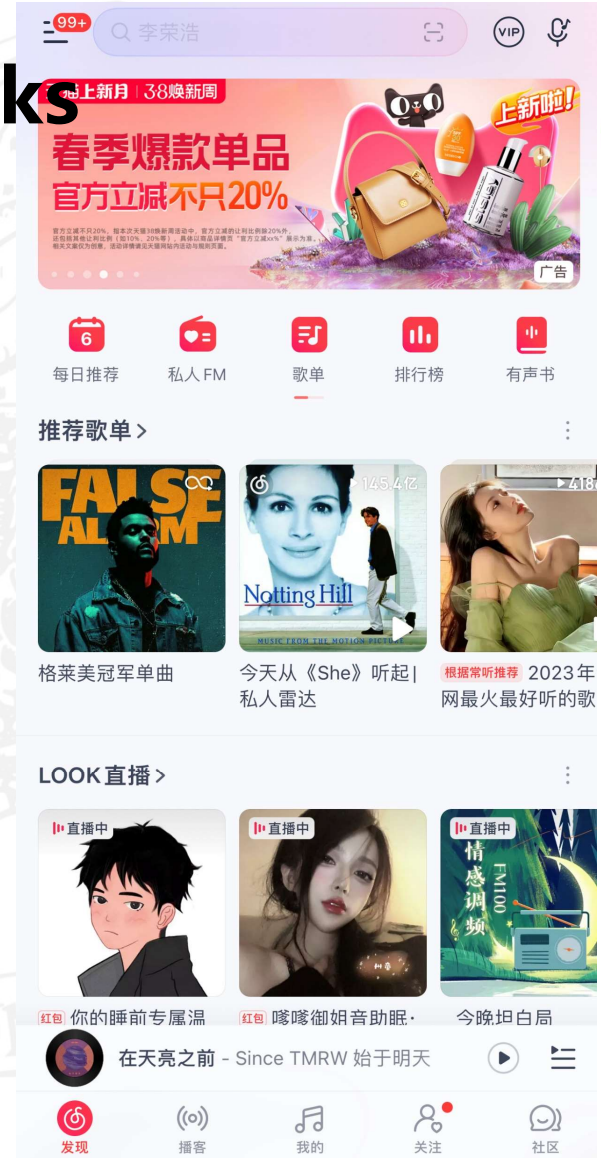
- 详细的输入输出定义

系统设计者的视角

- 接口规格

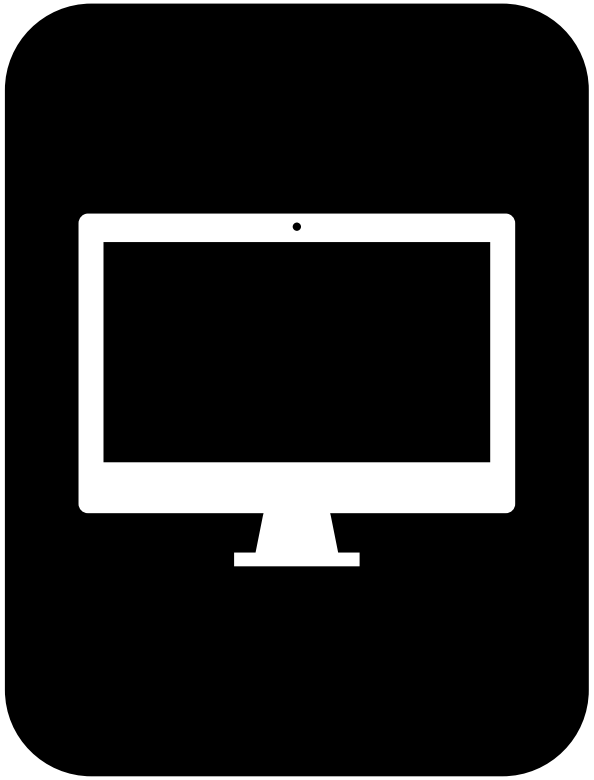
系统开发者的视角

- 接口层代码

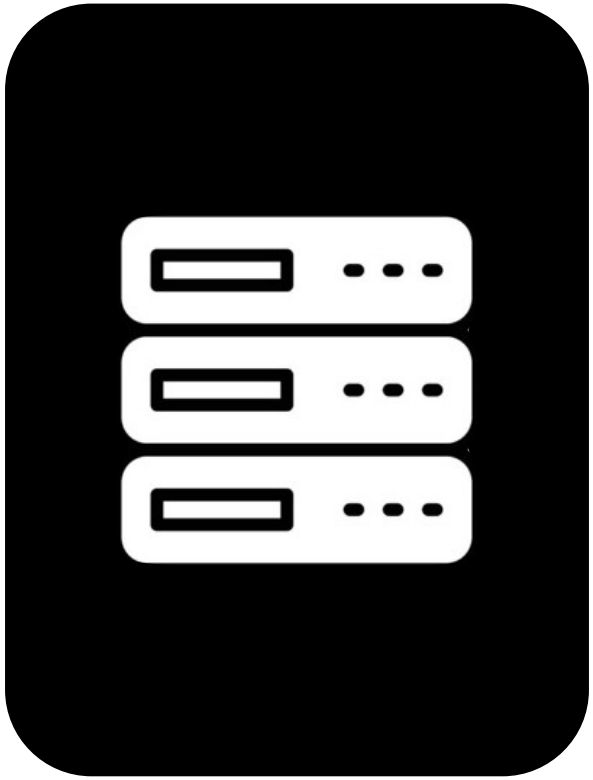


三层架构
Three-Tier Architecture

表现层
Presentation Tier



应用层
Logic Tier



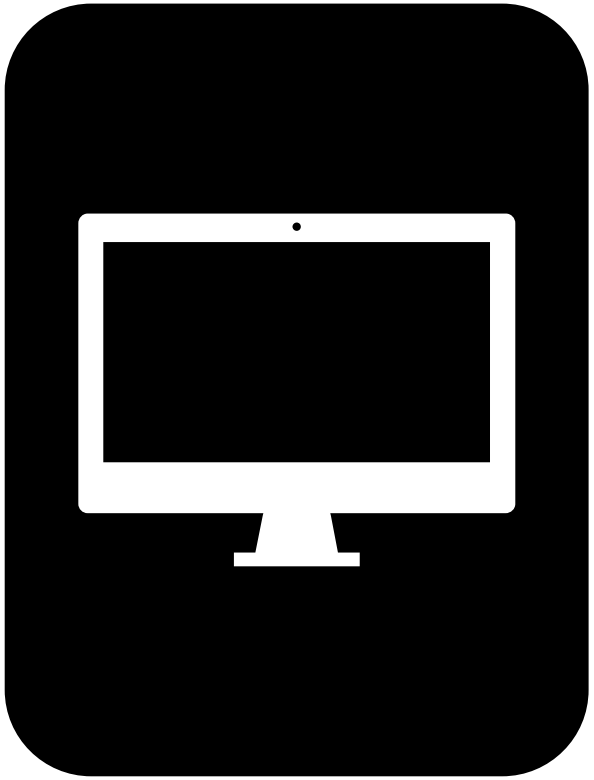
持久层
Data Tier

表现层

- 是应用程序的用户界面和通信层
- 用户在此与应用程序进行交互
- 主要目的是向用户显示信息并从收集信息
- 根据平台以各种不同语言编写
- Web 表示层通常使用 HTML、CSS 和 JavaScript 开发

表现层

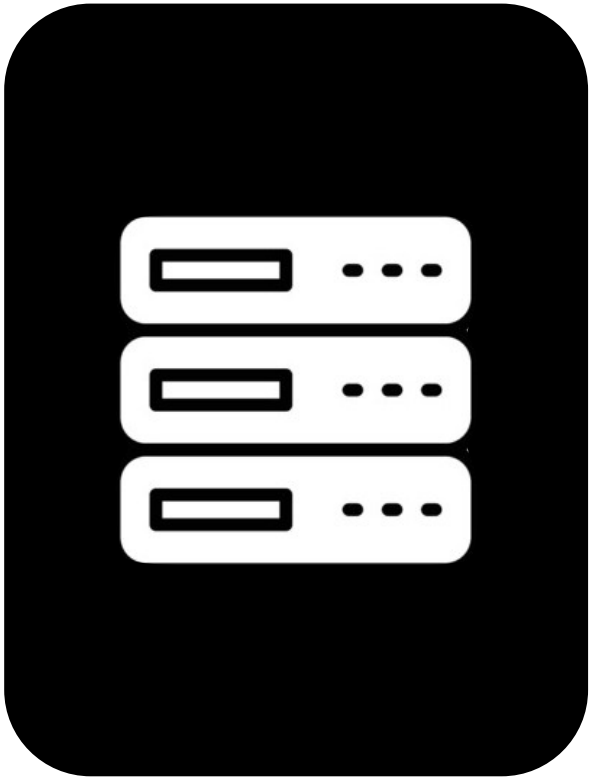
Presentation Tier



应用层

- 也称**逻辑层**或**中间层**，是应用程序的核心
- 通过业务逻辑来处理表现层中收集的信息
- 还可以添加、删除或修改持久层中的数据
- 通常使用 Python、Java或 Ruby等进行开发，并使用 API 调用与数据层通信

应用层
Logic Tier



持久层

- 也称为**数据层**、**数据访问层**
- 用于存储和管理应用程序所处理的信息
- 可以是关系数据库管理系统，例如 MySQL、Oracle 或 Microsoft SQL Server
- 也可以是 NoSQL 数据库服务器，如 MongoDB

/-2 软件开发过程模型
Software Development Process Model

软件工程三要素

三要素：方法、工具和过程

- ✓ 软件工程方法为软件开发提供了“如何做”的技术
- ✓ 软件工程工具为软件工程方法提供了自动的或半自动的软件支撑环境
- ✓ 软件工程过程是为了获得高质量的软件所需要完成的一系列任务框架，它规定了完成各项任务的工作步骤

软件工程过程定义了：方法使用的顺序；要求交付的文档资料；为保证质量和适应变化所需要的管理；软件开发各个阶段完成的里程碑

软件工程方法

软件工程方法(Method) 为构建软件技术提供技术上的解决方法（如何做）

包括沟通、需求分析、设计建模、程序构造、测试和技术支持

软件工程方法依赖于一组基本原则，这些原则涵盖了软件工程的所有技术领域，包括建模活动和其他描述性技术等

1. 结构化程序设计方法
2. 模块化程序设计方法
3. 面向对象程序设计方法

软件工程工具

软件工程工具（Tool）为过程和方法提供自动化或是半自动化的支持，这些工具可以集成起来，使得一个工具产生的信息可被另外一个工具使用

- 文档编写
- 分析设计
- 版本控制
- CASE (Computer-Assisted Systems Engineering)
- IDE (Integrated Development Environment)
- ADE (Application Development Environment)

软件过程

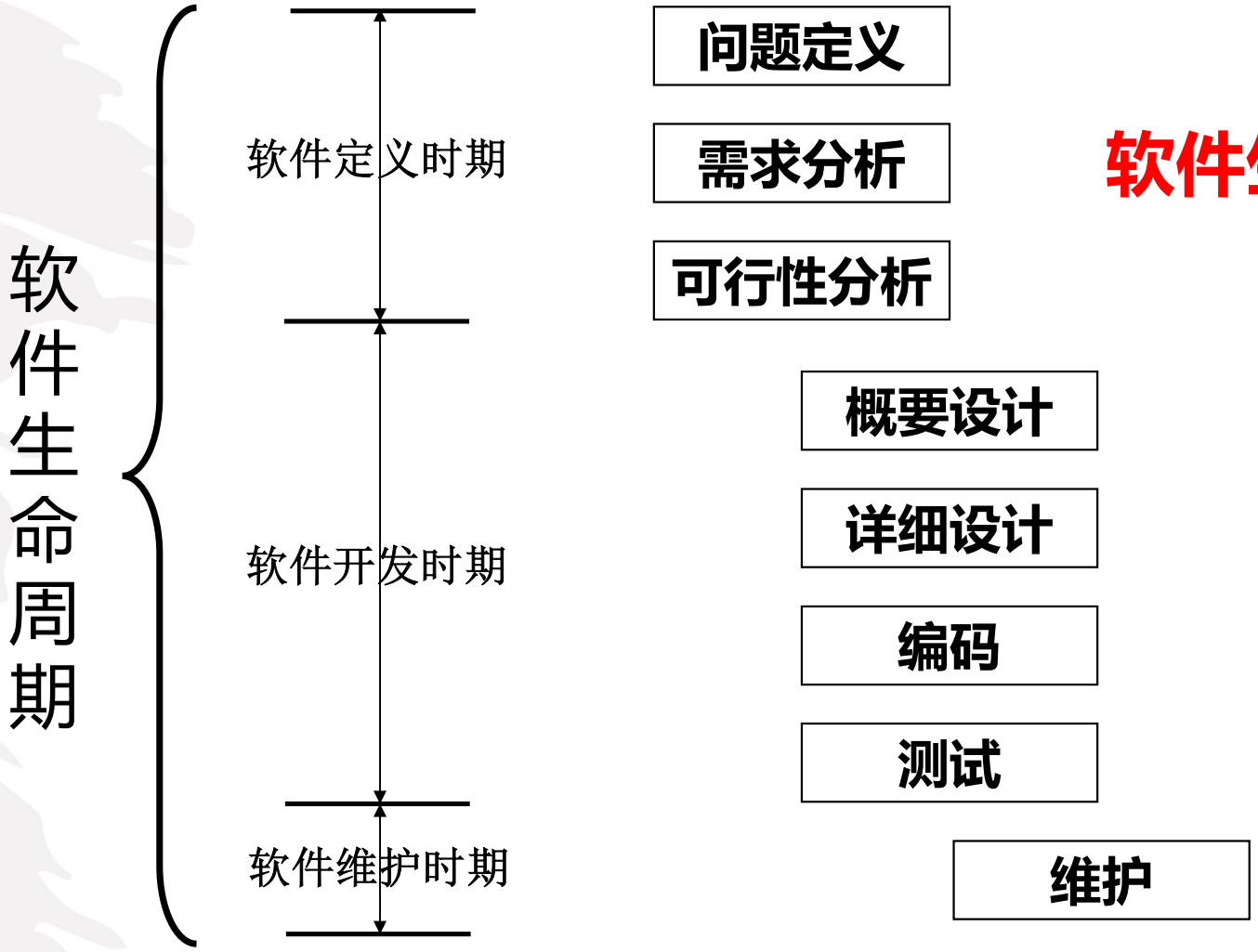
软件过程是工作产品构建时所执行的一系列**活动**、**动作**和**任务**的集合**活动**(Activity)主要实现宽泛的目标（如与参与者进行沟通），与应用领域、项目大小、结果复杂性或者实施软件工程的重要程度没有关系

动作(Action)包含主要工作产品生产过程中的一系列任务

任务(Task)关注小而明确的目标，能够生产实际产品（如构建一个单元测试）

过程**不是**对如何构建软件的严格规定，而是一种可适应性的调整方法，以便于工作人员（软件团队）可以挑选合适的工作动作和任务集合。其目标通常是及时、高质量地交付软件，以满足软件项目资助方和最终用户的需求

软件生命周期和开发过程



软件危机的典型表现

1. 对软件开发成本和进度的估计常常很不准确
2. 用户对“已完成的”软件系统不满意的现象经常发生
3. 软件产品的质量往往靠不住
4. 软件常常是不可维护的
5. 软件通常没有适当的文档资料
6. 软件成本在计算机系统总成本中所占的比例逐年上升
7. 软件开发生产率提高的速度，既跟不上硬件的发展速度，也远远跟不上计算机应用迅速普及深入的趋势

软件危机发生的主要原因

1. 缺乏软件开发的经验和有关软件开发数据的积累
2. 设计开发人员与用户的交流存在障碍
3. 软件开发过程不规范
4. 随着软件规模的增大，其复杂性往往会呈指数级升高
5. 缺少有效的软件评测手段

学生信息管理系统

Student Information System

产品需求提出
The Request

为了提高霍格沃茨学院学生相关管理的效率

- **支持远程协作**
- **流程自动化**

核心功能应该包括

系统开发方法
How - System Development Methodologies

- 瀑布模型 (Waterfall Model)
- 快速原型模型 (Rapid Prototype Model)
- 增量模型 (Incremental Model)
- 螺旋模型 (Spiral Model)
- 统一过程 (RUP, Rational Unified Process)
- 敏捷过程 (Agile Process Model)
- 极限编程 (XP, eXtreme Programming)
- 迭代式增量软件开发过程 (Scrum)
- …

系统开发的基本原则
Underlying Principles for Systems Development

- 让系统用户参与
- 使用一套问题解决步骤
- 确立开发阶段和开发活动
- 在开发过程中记录文档
- 建立标准
- 管理过程和项目
- 将信息系统作为重要的投资看待
- 不必害怕取消或返工
- 分而治之
- 设计系统时应考虑到增长和变化

软件开发过程中的典型文档

软件需求规格说明书：描述将要开发的软件做什么

项目计划：描述将要完成的任务及其顺序，并估计所需要的时间及工作量

软件测试计划：描述如何测试软件，使之确保软件应实现规定的功能，并达到预期的性能

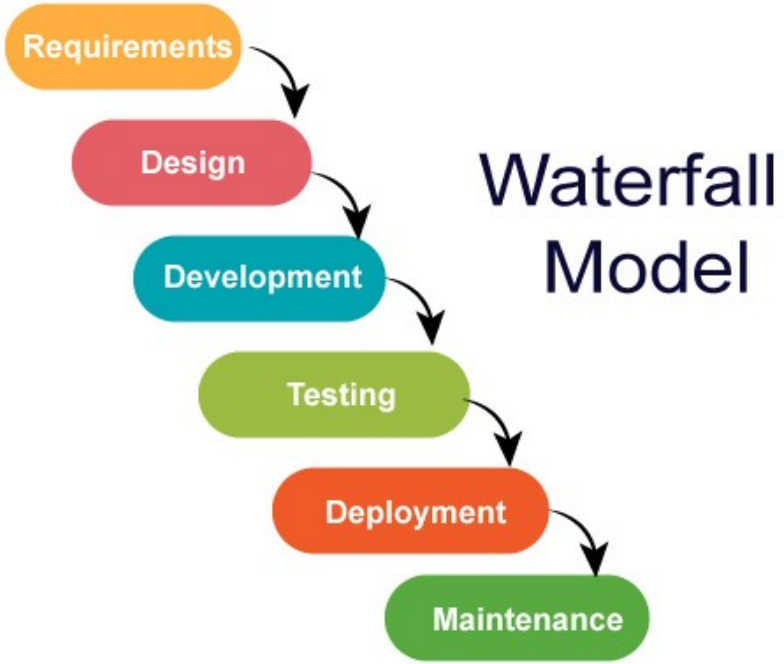
软件设计说明书：描述软件的结构，包括概要设计及详细设计

用户手册：描述如何使用软件。

系统开发方法
How - System Development Methodologies

- 瀑布模型 (Waterfall Model)
- 快速原型模型 (Rapid Prototype Model)
- 增量模型 (Incremental Model)
- 螺旋模型 (Spiral Model)
- 统一过程 (RUP, Rational Unified Process)
- 敏捷过程 (Agile Process Model)
- 极限编程 (XP, eXtreme Programming)
- 迭代式增量软件开发过程 (Scrum)
- …

瀑布模型 Waterfall Model



阶段间具有顺序性和依赖性：

1. **必须等前一阶段的工作完成之后，才能开始后一阶段的工作**
2. **前一阶段的输出文档就是后一阶段的输入文档**

瀑布模型的特点其一

推迟实现：

1. 瀑布模型在编码之前设置了系统分析和系统设计阶段，分析与设计阶段的基本任务是主要考虑目标系统的逻辑模型，不涉及软件的物理实现
2. 清楚地区分逻辑设计与物理设计，尽可能推迟程序的物理实现，是按照瀑布模型开发软件的一条重要的指导思想

质量保证：

1. 每个阶段都必须完成规定的文档，没有交出合格的文档就是没有完成该阶段的任务
2. 每个阶段结束前都要对所完成的文档进行评审，以便尽早发现问题，改正错误

瀑布模型的特点其二

强迫开发人员采用规范化的方法

严格地规定了每个阶段必须提交的文档

要求每个阶段交出的所有产品都必须是经过验证的

瀑布模型几乎完全依赖于书面的规格说明，很可能导致最终开发出的软件产品不能真正满足用户的需要。

如果需求规格说明与用户需求之间有差异，就会发生这种情况

瀑布模型只适用于项目开始时需求已确定的情况

什么时候选择瀑布模型

不能充分理解客户需求或客户需求可能发生变化

系统太大、太复杂、不能一次性做完所有的事情

拟采用的技术迅速发生变化

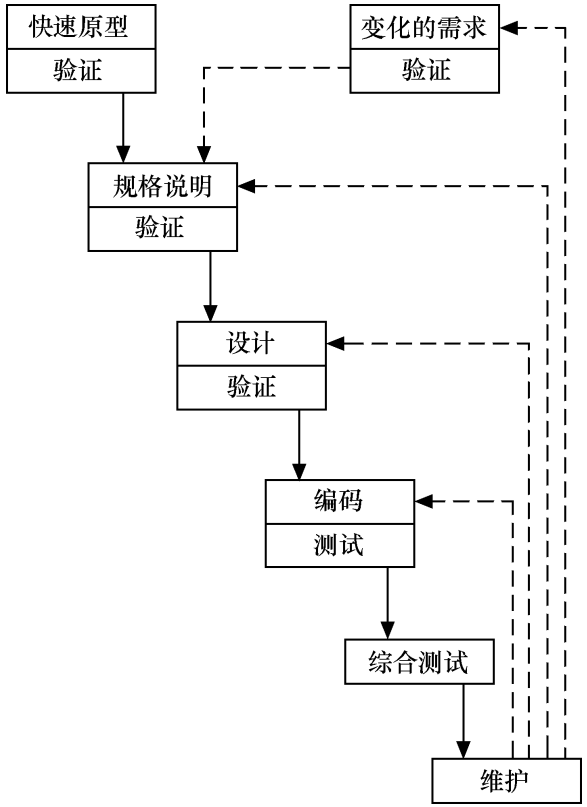
提供的资源有限

无法利用各个开发阶段的某一中间产品

系统的所有功能、性能要求客户可以一次性准确交付

是首次开发的新系统并且淘汰全部老系统

快速原型模型 Rapid Prototype Model



快速原型模型的特点

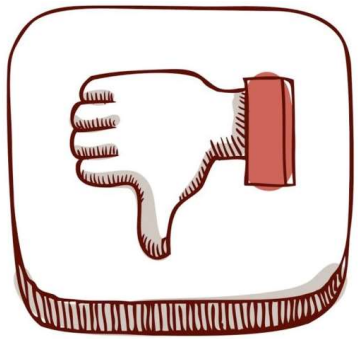
“快速”。尽可能快地建造出原型系统，以加速软件开发过程节约开发成本

原型系统已经通过与用户的交互得到验证

产生的规格说明文档能够正确地描述用户需求

软件产品的开发基本上是按线性顺序进行

规格说明文档正确地描述了用户需求，在开发过程的后续阶段不会进行较大的返工

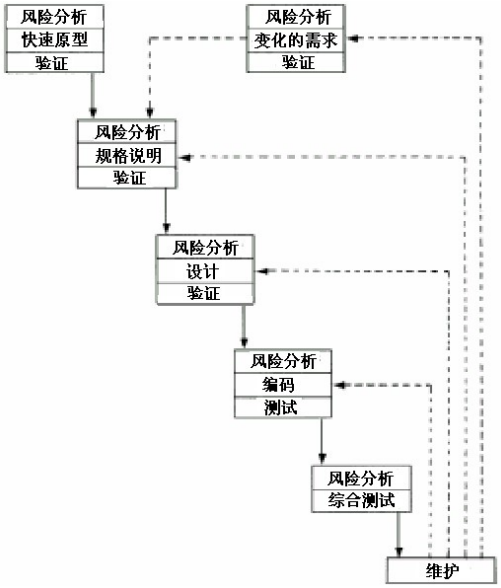


快速建立起来的系统结构加上连续的修改可能会导致产品质量低下

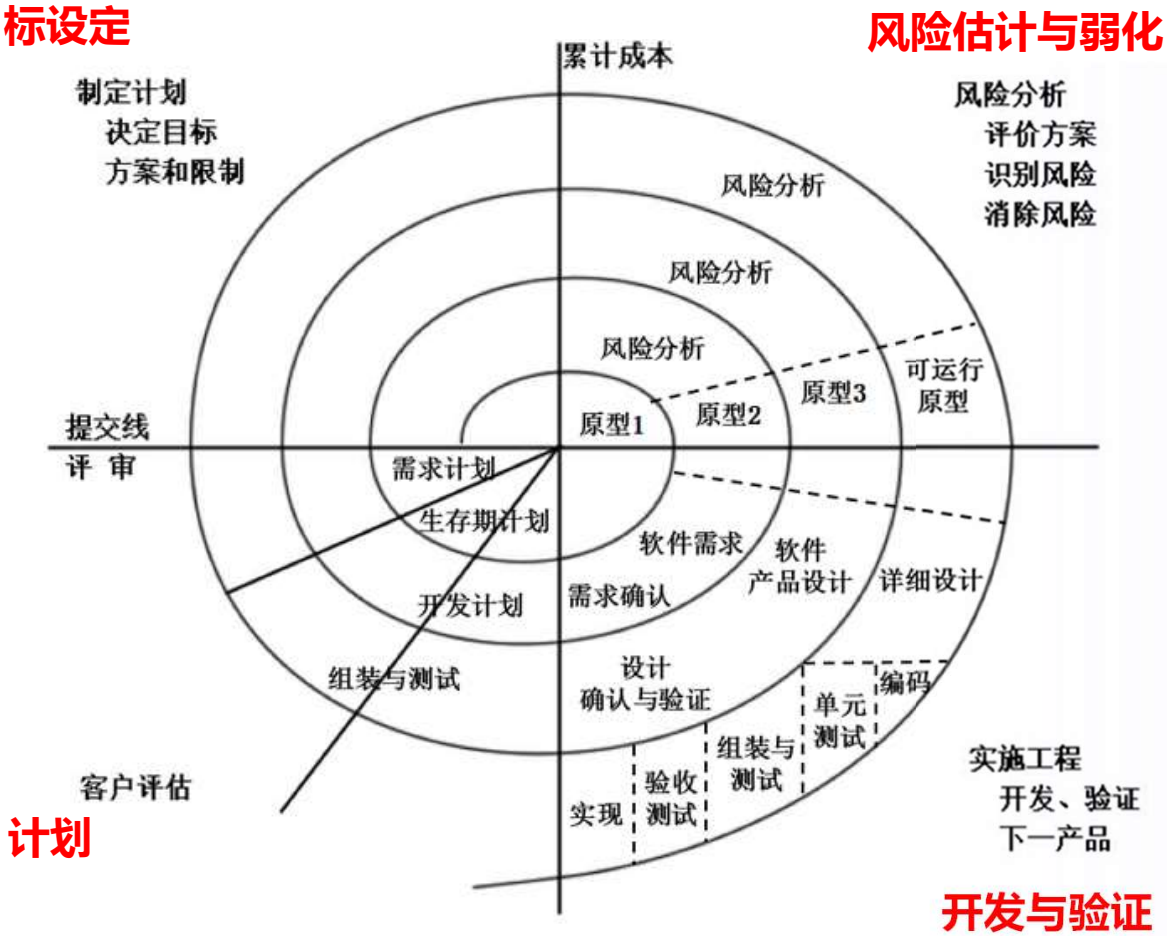
螺旋模型 Spiral Model

基本思想：

- **使用原型及其他方法来尽量降低风险**
- **将瀑布模型与快速原型模型结合起来，加入风险分析**



螺线上的每一个循环可划分为4个象限，分别表达了4个方面的活动



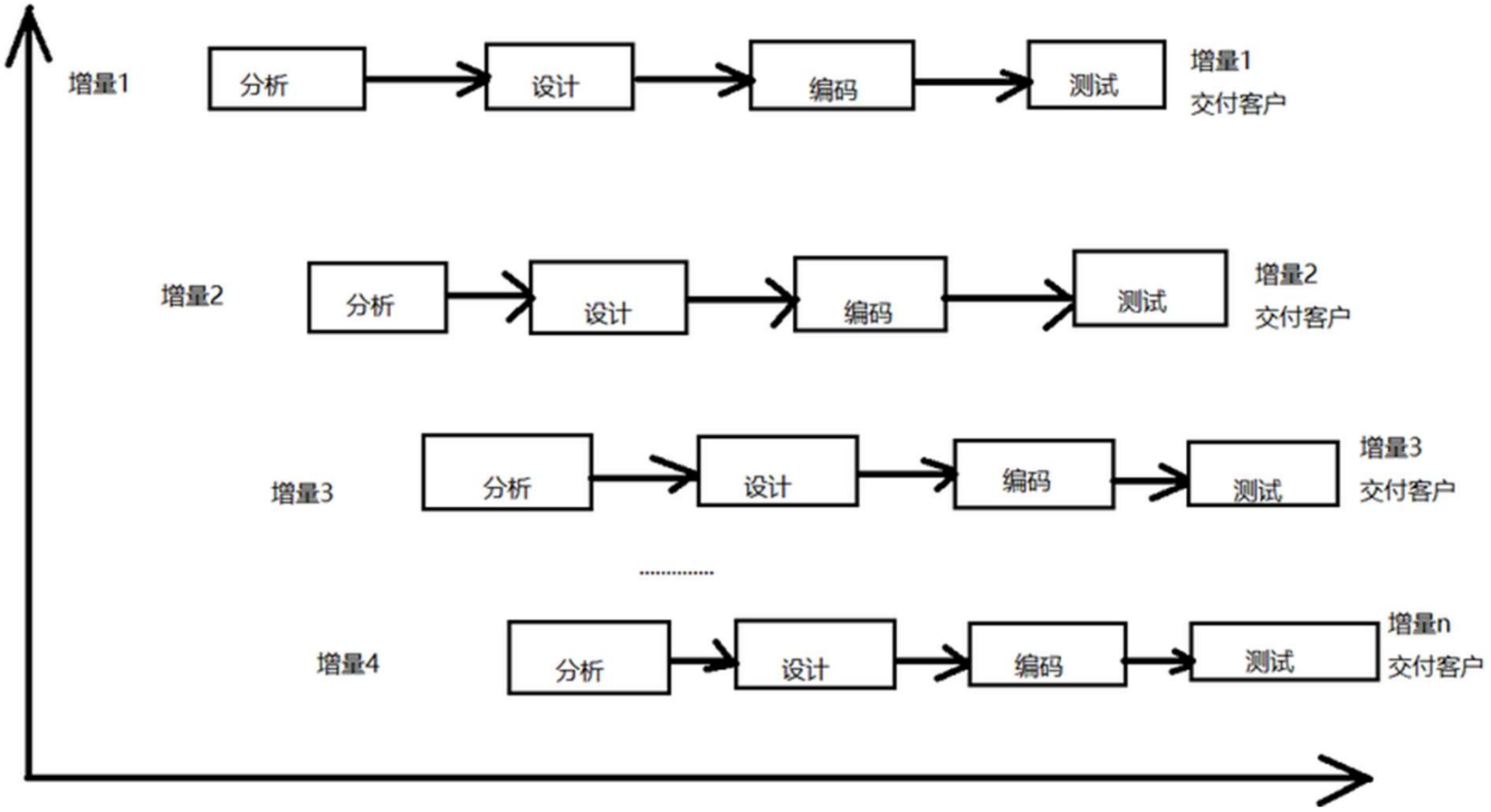
简化的螺旋模型

螺旋模型的特点

1. 对可选方案和约束条件的强调有利于已有软件的重用，也有助于把软件质量作为软件开发的一个重要目标
2. 减少了过多测试或测试不足所带来的风险
3. 在螺旋模型中维护只是模型的另一个周期，因而在维护和开发之间并没有本质区别

“风险驱动”。除非软件开发人员具有丰富的风险评估经验和这方面的专门知识，否则将出现真正的风险：当项目实际上正在走向灾难时，开发人员可能还以为一切正常

增量模型 Incremental Model



增量模型的特点

能在较短时间内向用户提交可一些有用的产品功能

用户有较充裕的时间学习和适应新产品

项目失败的风险较低

优先级最高的服务首先交付，然后再将其他增量构件逐次集成进来

“自相矛盾”。一方面要求把软件看做一个整体，另一方面又要求把软件看做构件序列，每个构件本质上都独立于另一个构件

什么时候选择增量模型

需要尽短的时间内得到系统基本功能的演示或使用

各版本都有中间阶段产品可提供使用

系统可以被自然地分割成渐增模式

开发人员与资金可以逐步增加

不能充分理解客户需求或客户需求有可能迅速发生变化

事先拟采用的技术迅速发生变化

客户突然提出一些新的功能需求

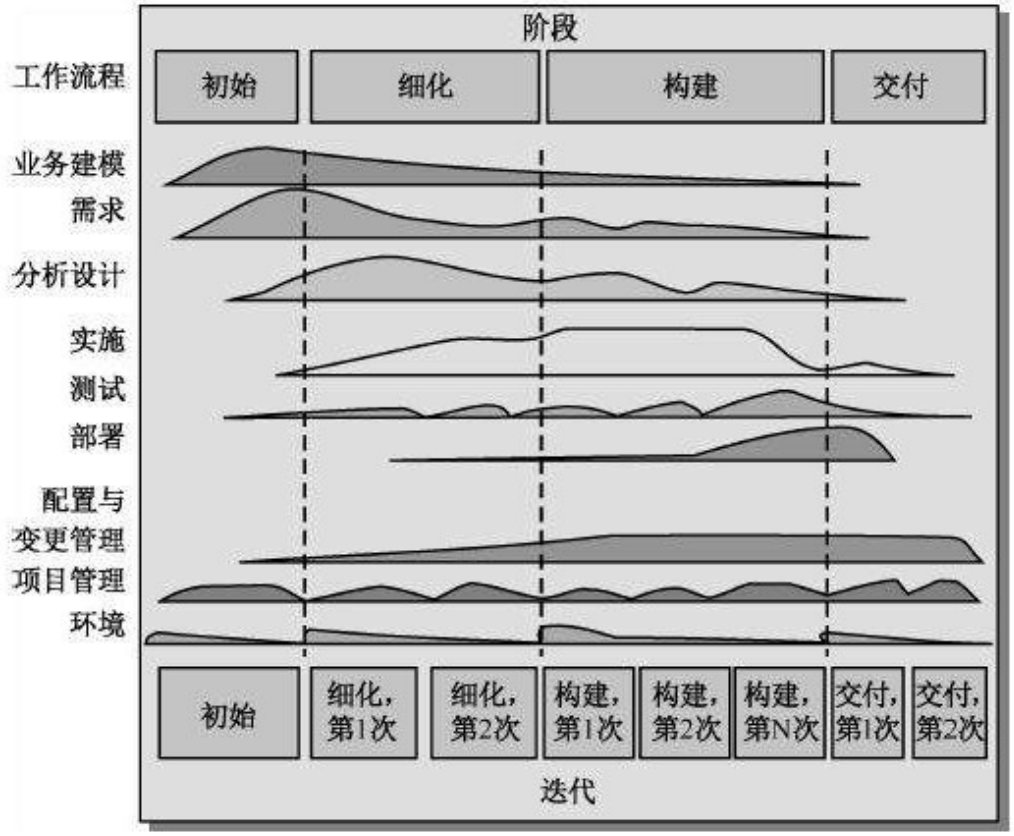
长时期内仅有有限的资源保证（开发人员和资金）

统一过程模型 Rational Unified Process, RUP

统一过程是用UML进行面向对象软件工程的框架

统一过程的阶段

1. **初始阶段：关注项目计划和风险评估**
2. **细化阶段：细化初始需求、优先级、业务用例以及制定项目管理计划**
3. **构建阶段：建立系统的第一个具有操作性的版本，以能够交付给客户进行测试的版本结束（测试版本）**
4. **交付阶段：以发布完整的系统而终止，确保系统真正满足客户需求**

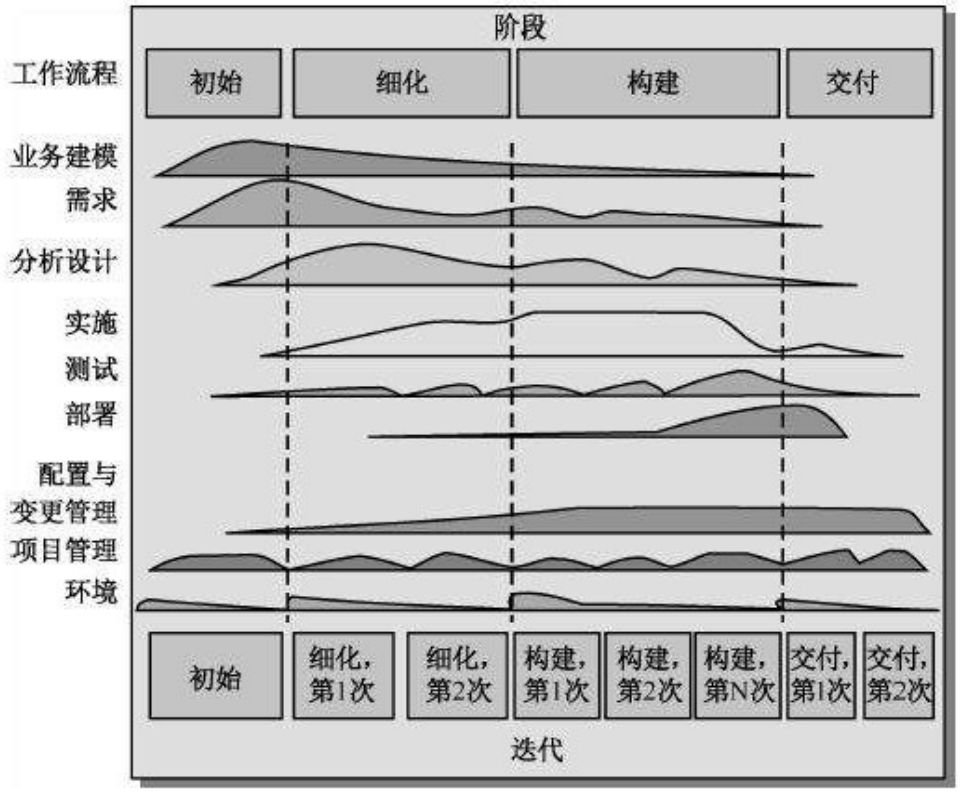


统一过程模型 Rational Unified Process, RUP

统一过程是用UML进行面向对象软件工程的框架

统一过程的核心过程工作流

1. **业务建模：用业务用例为业务过程建立模型**
2. **需求：明确系统的功能需求和非功能需求（约束）**
3. **分析和设计：分析和细化需求，建立分析模型和设计模型**
4. **实现：用分层的方式组织代码的结构，用构件的形式来实现类，对构件进行单元测试，将构件集成到可执行的系统中**
5. **测试：执行集成测试。验证对象之间的交互、是否所有的构件都集成了、是否正确实现了所有需求、查错并改正**
6. **部署：制作软件的外部版本、软件打包、分发、为用户提供帮助和支持**



“敏捷软件开发”宣言：

- 个体和交互 胜过 过程和工具
- 可工作软件 胜过 宽泛的文档
- 客户合作 胜过 合同谈判
- 响应变化 胜过 遵循计划

敏捷过程模型 Agile Process Model

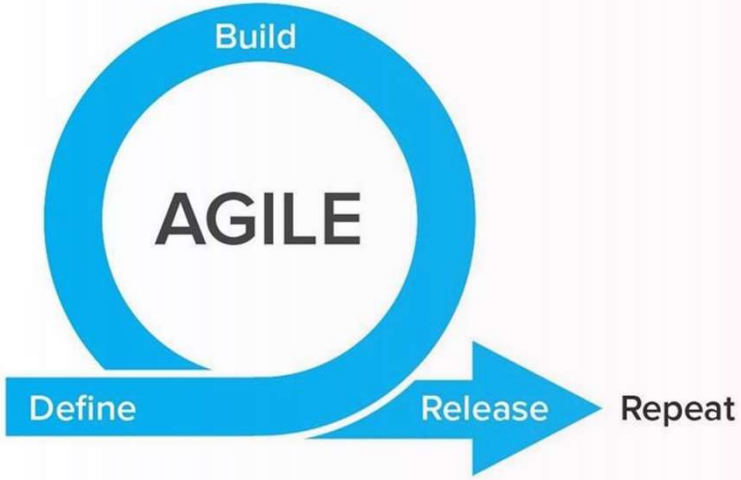
- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

— The four values of the Agile Manifesto

敏捷过程模型 Agile Process Model

“敏捷软件开发”宣言：

- **个体和交互**
  - 围绕有积极性的个人构建项目。给他们提供所需的环境和支持，并且信任他们能够完成工作
  - 在团队内部，最富有效果和效率的信息传递方法是面对面交谈
- **可工作软件**
  - 经常交付可运行软件，交付的时间间隔越短越好
- **客户合作**
  - 最优先、尽早、持续交付有价值的软件来使客户满意
- **响应变化**
  - 即使在开发的后期，也欢迎需求变更



敏捷过程模型的特点

基于价值驱动交付，**更快交付价值**
确保考虑并接纳每个人的意见，**更高的团队满意度**
每个迭代回顾会议进行分析、讨论、总结，**持续改进**
根据市场不断调整需求范围、变更以及优先级，**更大的灵活性**

很难进行准确的资源规划
很难准确的定义“轻量的“或必要的文档
很难把握整体产品的一致性

极限编程（eXtreme Programming, XP）

使用最广泛的敏捷过程

包含了策划、设计、编码和测试4个框架活动的规则和实践

策划：

细分功能需求(用户故事)，分组实现需求

设计：

保持简洁，碰到困难立即建立这部分的可执行原型

编码：

结对编程，两个人面对同一台计算机共同开发代码

测试：

所建立的单元测试使用一个可以自动实施的框架

学生信息管理系统的开发
Apply to Student Information System

为了提高霍格沃茨学院学生相关管理的效率

- 支持远程协作
- 流程自动化

系统开发方法的选择?

- 关注数据的输入、输出、存储
- 关注业务问题的特点

核心功能应该包括

PIECES 框架

Performance 性能	改进**性能**的需要
Information 信息	改进**信息**和数据的需要
Economics 经济	改进**经济**、控制成本或增加收益的需要
Control 控制	改进**控制**或安全的需要
Efficiency 效率	改进人与过程的**效率**的需要
Service 服务	改进对客户、供应商、合作伙伴、雇员等的**服务**需要

本节内容

Readings

《软件工程概论》

- 第1章 软件与软件工程的概念
- 第2章 软件生存期模型

《系统分析与设计方法》

- 第1章 系统分析和设计方法的环境
- 第2章 信息系统构件
- 第3章 信息系统开发

关键词：系统开发过程；软件危机；瀑布模型；快速原型模型；螺旋模型；统一过程；敏捷过程；