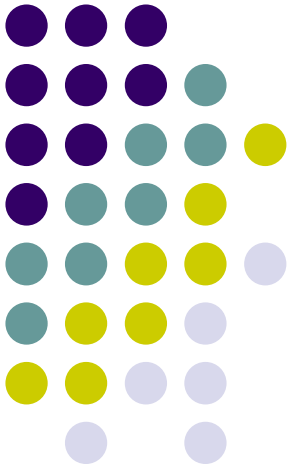
软件系统分析与设计

Lecture 01 Introduction



by Dr. Y.Yang

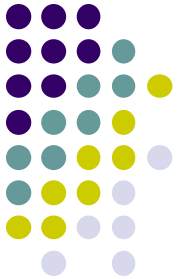
Associate Professor,

School of Computer Science & Technology,

Soochow University

yyang@suda.edu.cn

About the lecturer



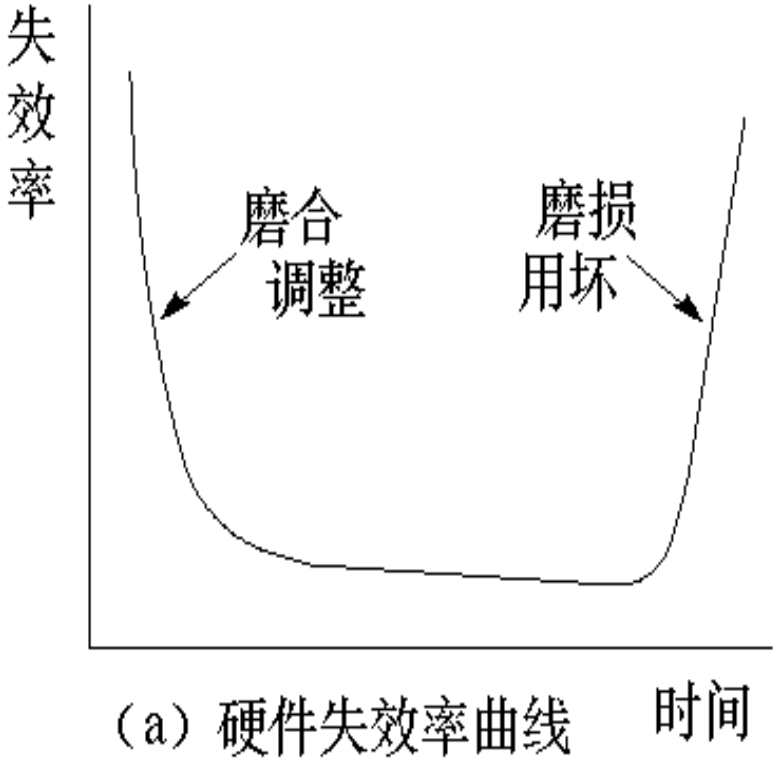
- yyang@suda.edu.cn
- 本部理工楼545

Today's Topics

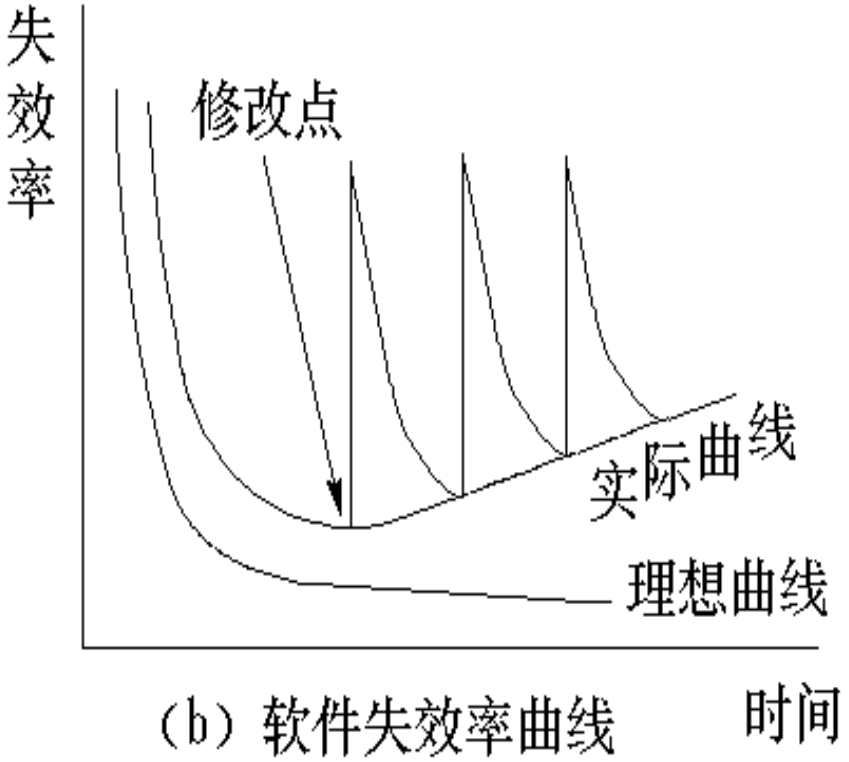
- 软件开发的基本方法
- 面向对象技术的基本概念
- 面向对象方法的特点和优点
- **OOA**的方法和步骤
- **OOD**的工作要点

几个重要的SE概念

- 1.软件危机和软件工程

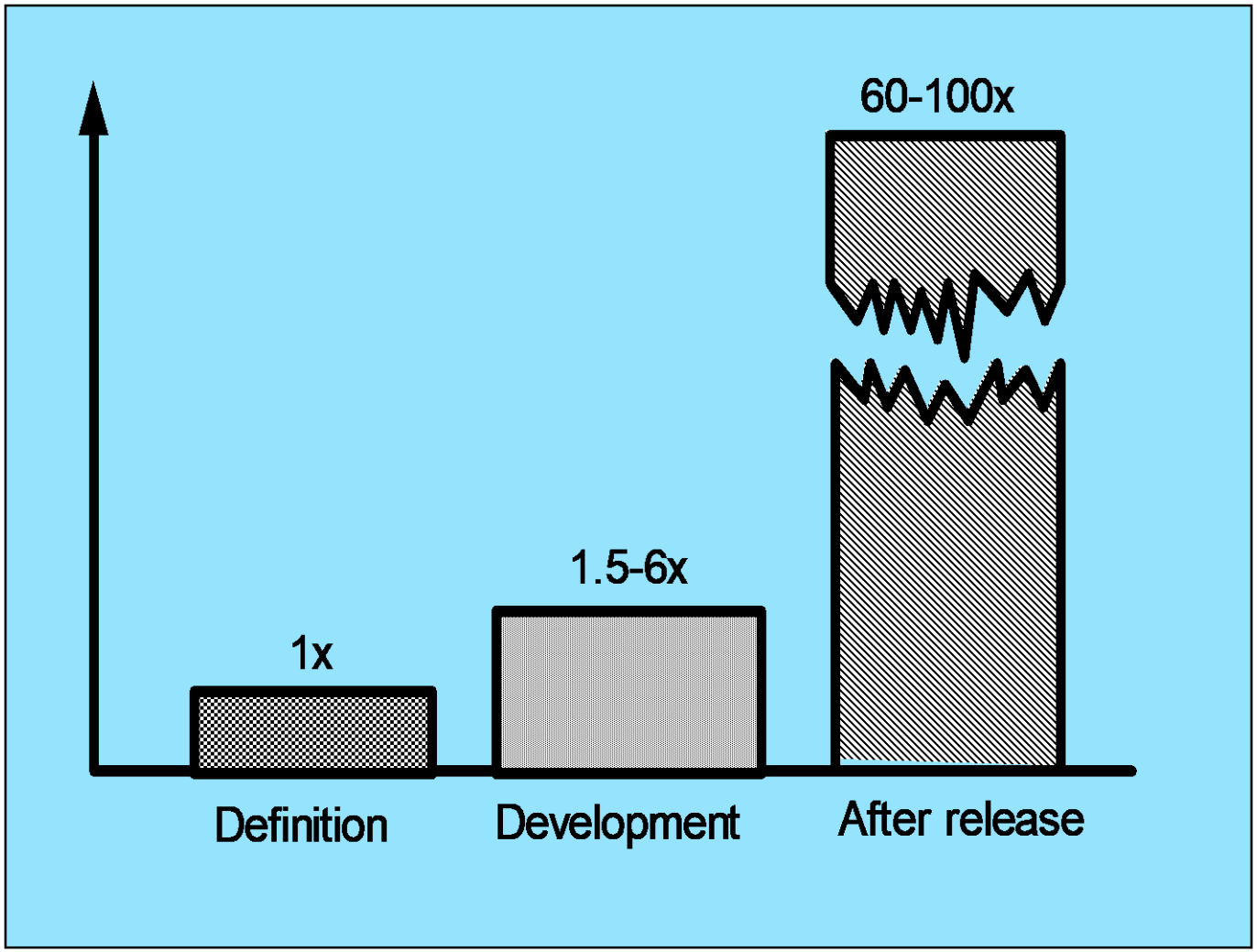


(a) 硬件失效率曲线



(b) 软件失效率曲线

The Cost of Change



软件危机的主要特征

- 软件开发周期大大超过规定日期;
- 软件开发成本严重超标;
- 软件质量难于保证;
- 软件难于维护。
- 原因
  - 软件本身的复杂性
  - 软件产品的特殊性
  - 人们认识的局限性

软件产品的复杂性（一）

- 例如，Windows 95有1000万行代码，Windows 2000有5000万行代码

Exchange2000和 Windows2000开发人员结构

	Exchange 2000	Windows 2000
项目经理	25人	约250人
开发人员	140人	约1700人
测试人员	350人	约3200人

软件产品的复杂性（二）

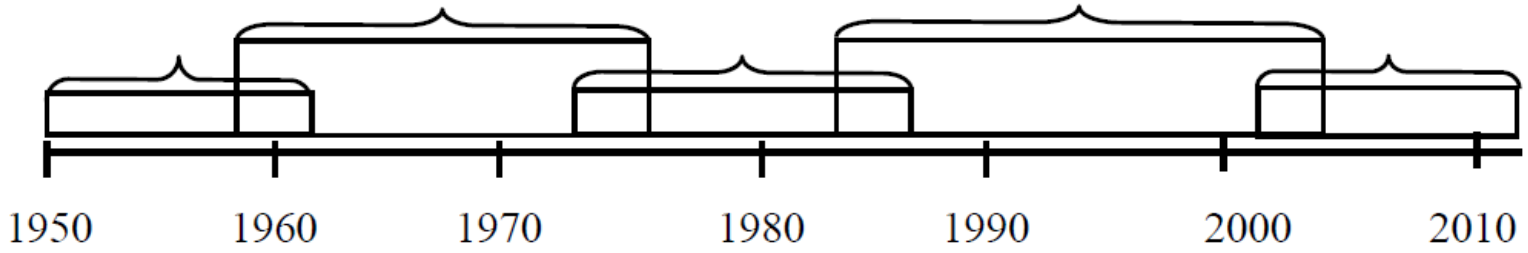
早期
- 面向批处理
- 有限的分布
- 自定义软件

第二阶段
- 多用户
- 实时
- 数据库
- 软件产品

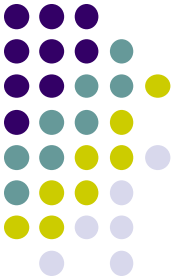
第三阶段
- 分布式系统
- 嵌入“智能”
- 低成本硬件
- 消费者的影响

第四阶段
- 强大的桌面系统
- 面向对象技术
- 专家系统
- 人工神经网络
- 并行计算
- 网络计算机

第五阶段
- 大数据
- 云计算
- 物联网



Definition of SE



- **Fritz Bauer(1968) in NATO：**

  “建立并使用完善的工程化原则，以较经济的手段获得能在实际机器上有效运行的可靠软件的一系列方法。”

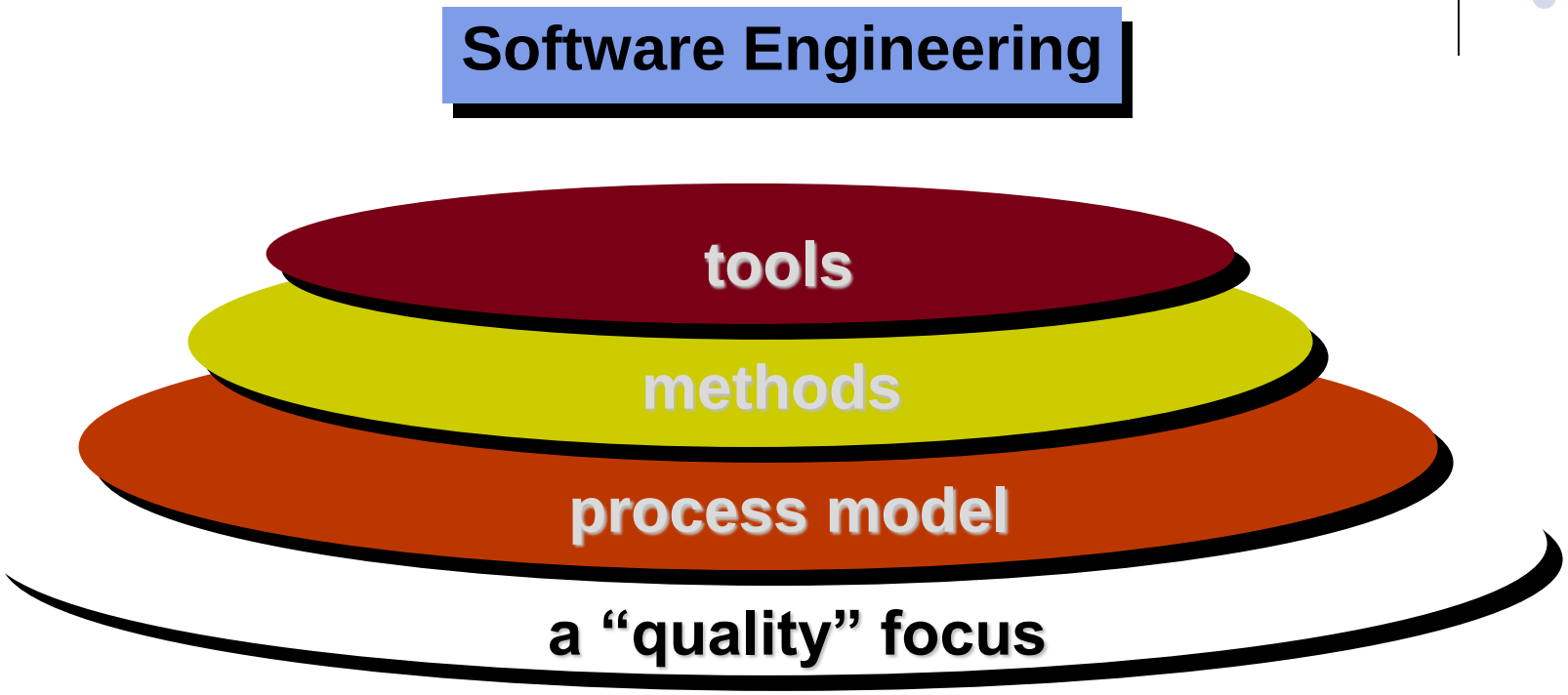
- **IEEE(1983)：**

  “软件工程是开发、运行、维护和修复软件的系统方法。”

- **IEEE(1993)：**

  “将系统化的、规范的、可度量的方法应用于软件的开发、运行和维护的过程，即将工程化应用于软件中。”

A Layered Technology



几个重要的SE概念

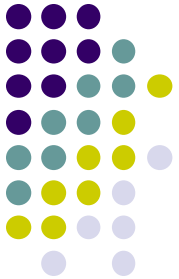
2.软件开发方法

*软件工程的方法*提供了建造软件在技术上需要“如何做”。方法覆盖了一系列的任务：需求分析、设计、编程、测试和支持。软件工程方法依赖于一组基本原则，这些原则控制了每一个技术区域且包含建模活动和其他描述技术。

良好软件属性

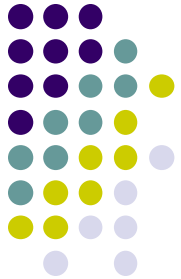
ISO 9126

功能性	可靠性	可用性	效率	可维护性	可移植性
可适应性	成熟度	可理解性	时间行为	可分析性	自适应性
准确性	容错	可学习性	资源使用	可变更性	一致性
可操作性	可恢复性	可操作性		稳定性	可替换性
安全性				可测试性	



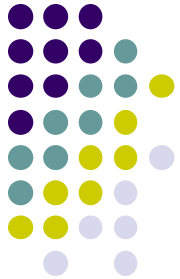
一是从管理的角度，希望实现软件开发过程的工程化：

- “瀑布式”生命周期模型：分析—设计—编码—测试—维护
- 快速原型法、螺旋模型、喷泉模型等对“瀑布式”生命周期模型进行补充。



- 二是侧重于对软件开发过程中分析、设计方法的研究。诞生了许多程序设计范型。

程序设计范型



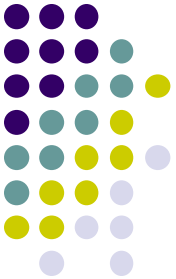
- 范型指的是一种通用的一般化的关于现实世界的模型，这个模型提供了观察和研究现实世界的一个视图。换句话说，范型就是被大家共享的概念集。

- 程序设计范型是关于计算机系统的思考方法，是分类一组语言的抽象特征的集合，它体现了一类语言的主要特点，这些特点能用来支持应用域所希望的设计风格。

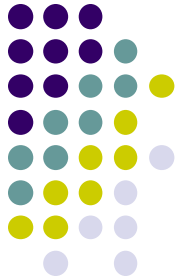
编程范例的种类

- 面向过程　算法
- 面向对象　类和对象
- 面向逻辑　目标，通常以谓词演算表示
- 面向规则　**if-then**规则
- 面向约束　不变量的关系

计算机世界中的抽象层次



- **XO (X?一Oriented)：最高的抽象层次**
- **OO (对象、类)**
- **PO (过程、函数、变量)**
- **变量、运算、表达式、语句**
- **字节 (4位、8位、16位、32位、64位)**
- **二进制串0101011110001：最低的抽象层次**

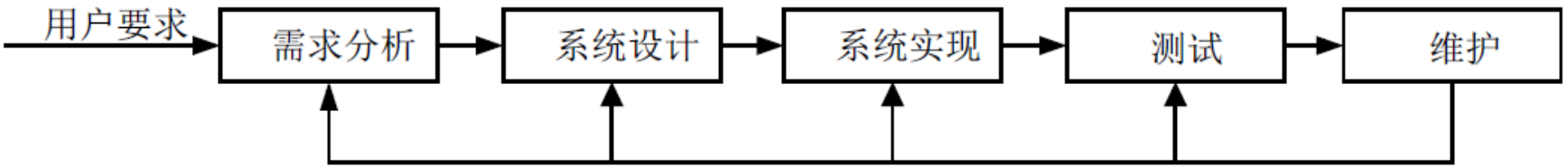


- 软件工程的发展历史就是人们不断追求更高的抽象、封装和模块化的历史，面向对象技术当然不会是历史的终结。

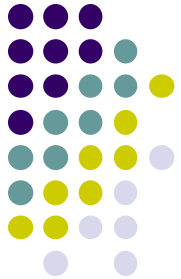
几个重要的SE概念

3. 软件生存周期

- 软件定义
- 软件开发
- 软件使用与维护



软件定义阶段



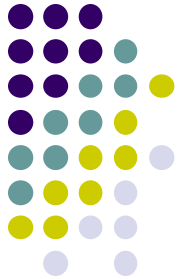
- 软件系统的可行性研究

  经济、技术、法律可行性研究和开发方案的选择性研究

- 项目需求分析

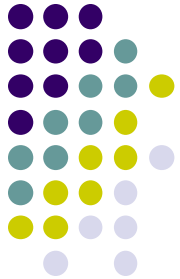
  提交需求规格说明书

软件开发阶段



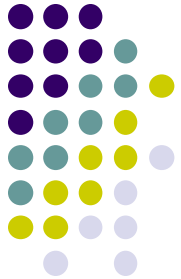
- 概要设计
- 详细设计
- 实现
- 组装测试
- 确认测试

软件使用、维护和更新换代阶段



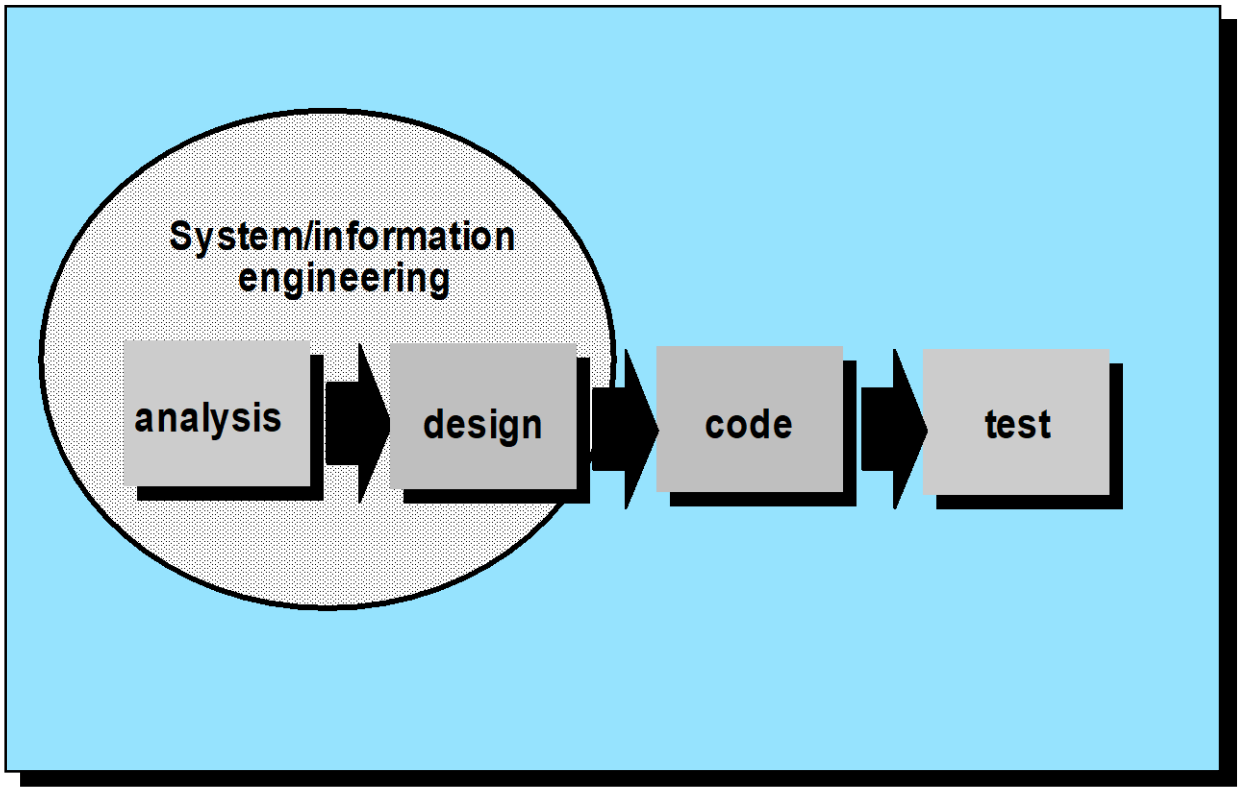
- 软件使用：“软件问题报告”
- 软件维护
- 系统更新换代

开发过程模型

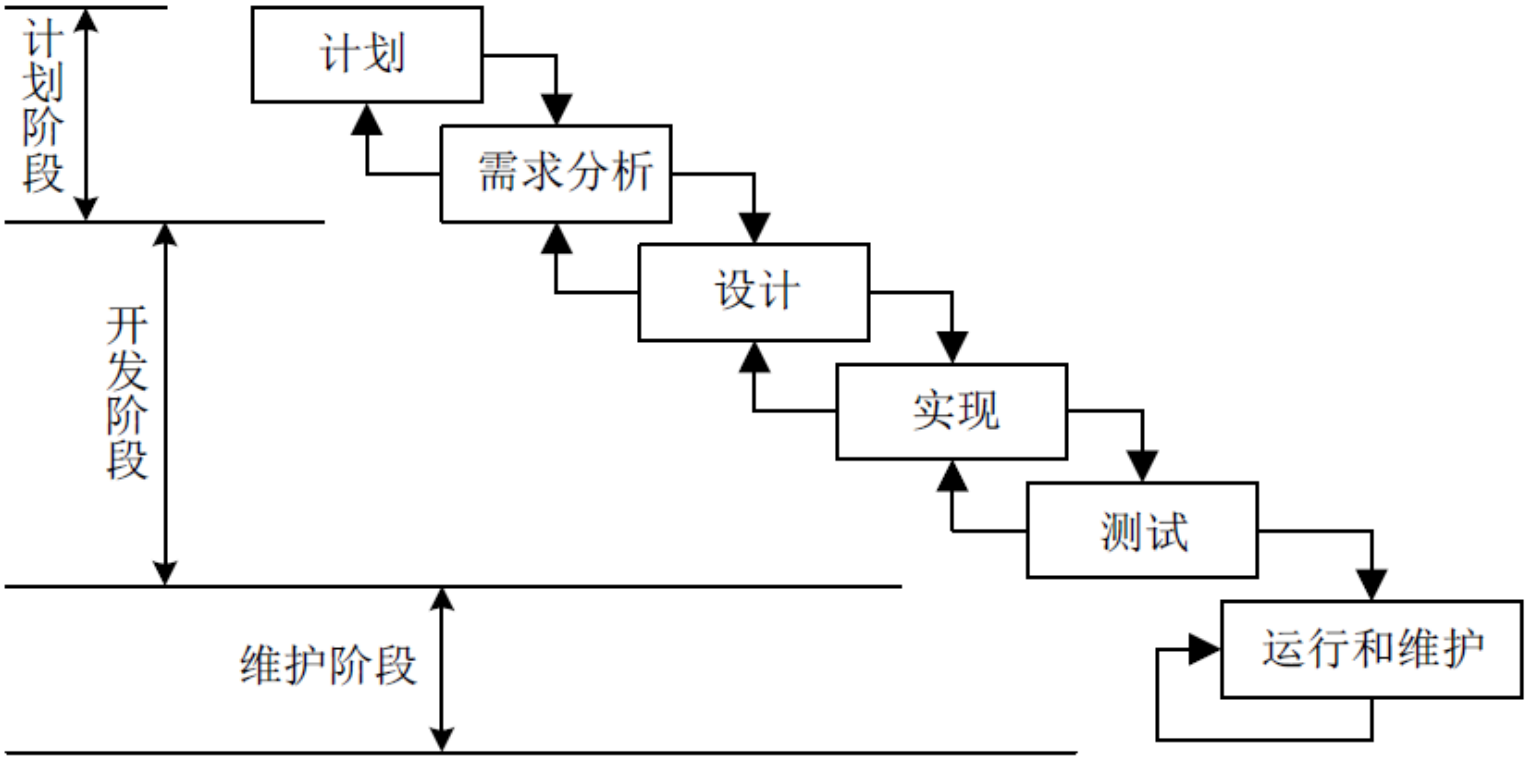


- 瀑布模型（线性模型）
- 增量模型
- 螺旋模型

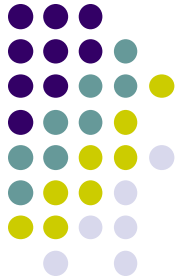
The Linear Model



瀑布模型

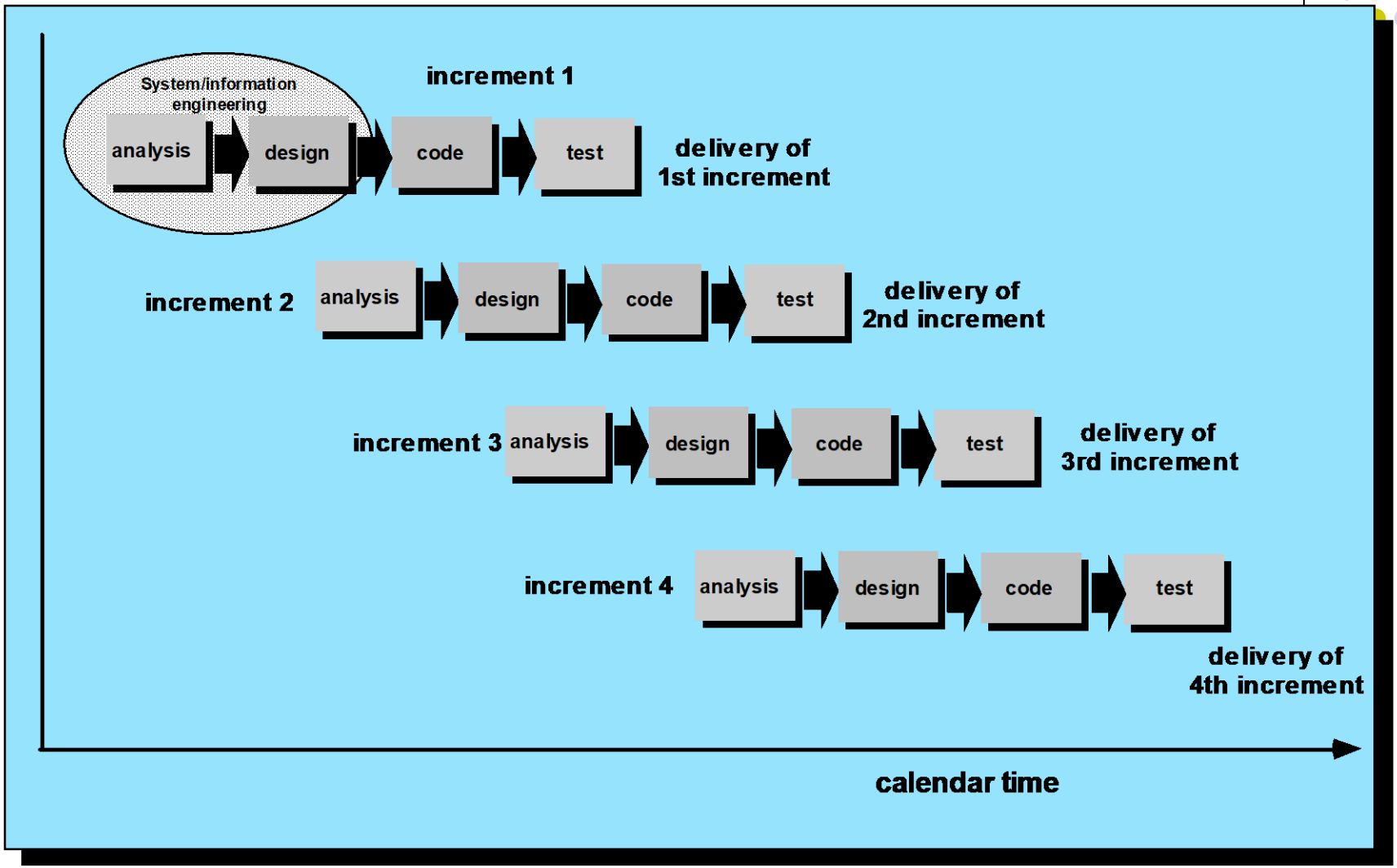


瀑布模型开发软件的特点

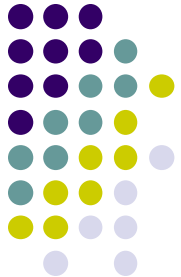


- 各项活动按自上而下，相互衔接的固定次序，如同瀑布逐级下落。每项活动均处于一个质量环（输入-处理-输出-评审）中。
- 阶段间具有顺序性和依赖性。
- 推迟实现的观点。
- 每个阶段必须完成规定的文档；每个阶段结束前完成文档审查。

The Incremental Model

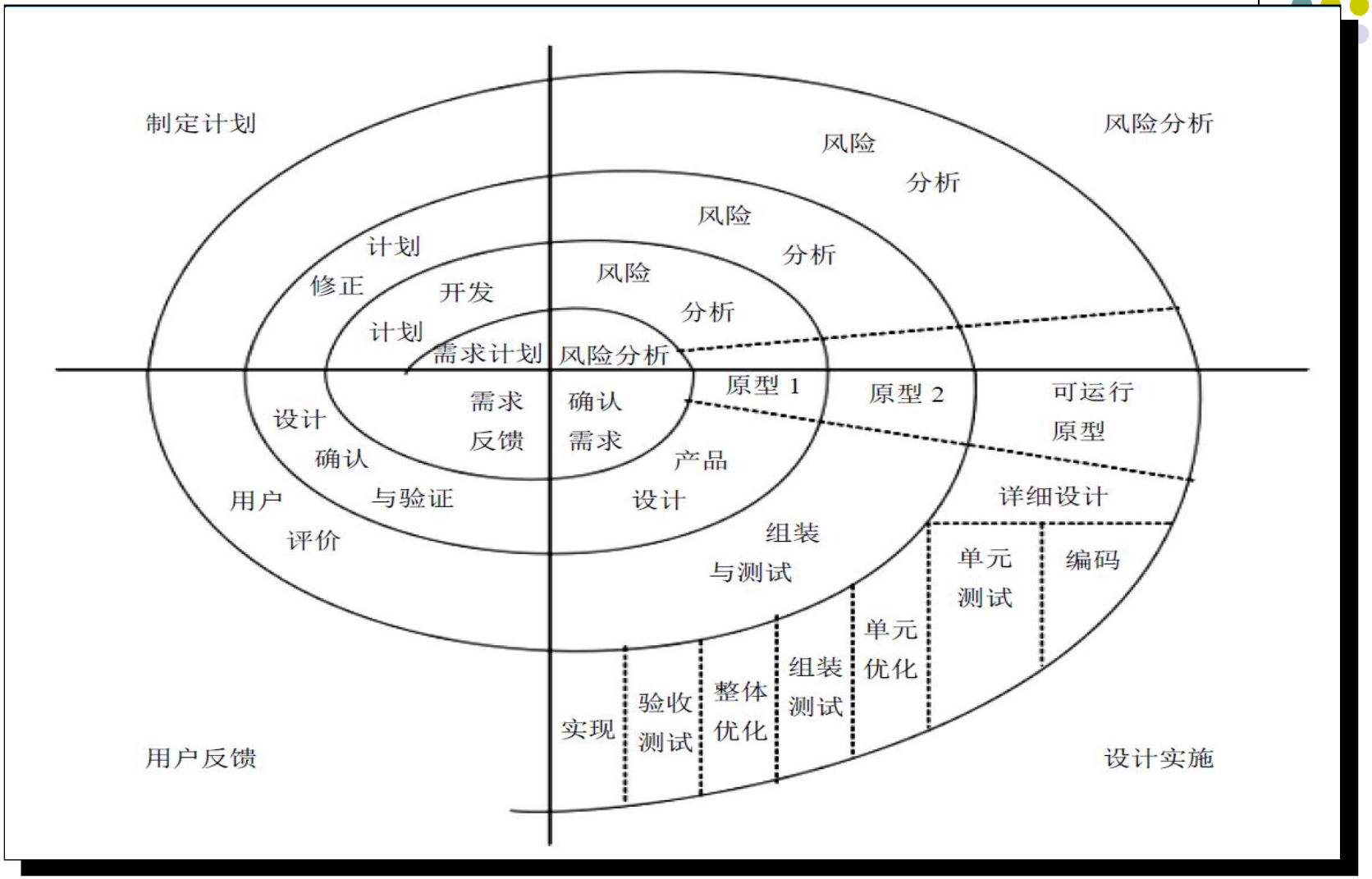


增量模型



- 增量模型是迭代和演进的过程。
- 增量模型把软件产品分解成一系列的增量构件，在增量开发迭代中逐步加入。
- 每个构件由多个相互作用的模块构成，并且能够完成特定的功能。
- 早先完成的增量可以为后期的增量提供服务。
- 增量开发方法的新演进版本叫做“极限程序设计（eXtreme Programming）”。

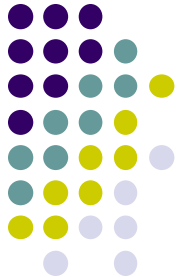
An Evolutionary (Spiral) Model



螺旋模型

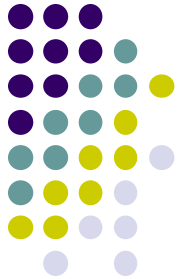
- 螺旋模型将瀑布模型与演化模型结合起来，并且加入两种模型均忽略了的风险分析。

- 螺旋模型沿着螺线旋转，自内向外每旋转一圈便开发出更完善的一个新版本。
  - **制定计划** 确定软件目标，选定实施方案，弄清项目开发的限制条件；
  - **风险分析** 分析所选方案，考虑如何识别和消除风险；
  - **实施工程** 实施软件开发
  - **客户评估** 评价开发，提出修正建议。



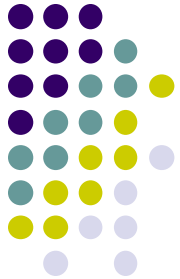
- 在开发过程中，项目经理需要根据项目实际情况和开发人员情况，灵活选择、组合不同的开发过程模型，甚至可以发明新的开发模型，以提高开发效率，保证软件质量，切忌教条主义。

结构化方法的基本概念



- 结构是指系统内各组成要素之间的相互联系、相互作用的框架。
- 结构化方法也称面向过程的方法或传统软件工程开发方法，由E.Yourdon和L.L.Constantine于1978年提出。
- 特点是自顶向下地分析与设计，逐步求精。
- 一组提高软件结构合理性的准则，如分解和抽象、模块的独立性、信息隐蔽等。

面向对象方法的基本概念：



1. 什么是面向对象

- Coad-Yourdon给出了一个定义：

  面向对象 = 对象 + 类 + 继承 + 通信

- 如果一个软件系统是使用这样 4 个概念设计和实现的，则认为这个软件系统是面向对象的。

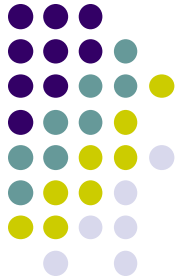
- 一个面向对象的程序的每一成份应是**对象**，计算是通过新的**对象的建立**和**对象之间的通信**来执行的。

面向对象方法的基本概念：

2.面向对象的特点

- **抽象性**：对象的数据抽象和行为抽象；
- **封装性**：信息隐蔽（两个视图）；
- **共享性**：
  - 同一类中所有实例共享数据结构和行为特征；
  - 同一应用中所有实例通过继承共享数据结构和行为特征；
  - 不同应用中所有实例通过复用共享数据结构和行为特征

面向对象方法的基本概念：
3. 对象 (object)

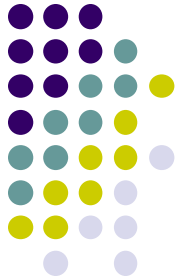


- 对象是系统中用来描述客观事物的一个实体，是构成系统的一个基本单位，由一组属性和一组对属性进行操作的服务组成。

- 属性一般只能通过执行对象的操作来改变。

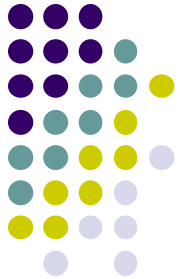
- 操作又称为方法或服务，它描述了对象执行的功能，若通过消息传递，还可以为其他对象使用。

面向对象的基本概念：4. 类(class)



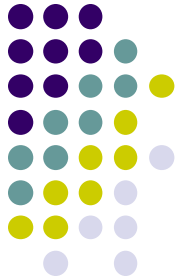
- 把具有相同特征（属性）和行为（操作）的对象归在一起就形成了类(class) 。
- 类的定义包括一组数据属性和在数据上的一组合法操作。
- 在一个类中，每个对象都是类的实例(Instance)，它们都可使用类中的函数。
- 类定义了各个实例所共有的结构，使用类的构造函数，可以在创建该类的实例时初始化这个实例的状态（实例变量）。

面向对象的基本概念：5. 消息（Message）



- 消息是一个实例与另一个实例之间传递的信息，要求该实例执行类中定义的某个操作。
- 消息的使用类似于函数调用，消息中指定了某一个实例，一个操作名和一个参数表 (可能是空的)。
- 接收消息的实例执行消息中指定的操作，并将形式参数与参数表中相应的值结合起来。

面向对象的基本概念：6. 继承 (Inheritance)

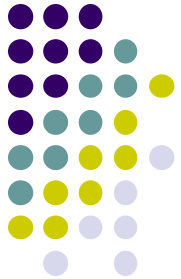


- 如果某几个类之间具有共性的东西(信息结构和行为)，抽取出来放在一个一般化类（泛化类）中，而将各个类的特有的东西放在特殊化类（特化类）中分别描述，则可建立起特化类对泛化类的继承。

- 继承是使用已有的类定义做为基础建立新类的定义的技术。

- 已有的类可当做基类来引用，则新类相应地可当做派生类来引用。

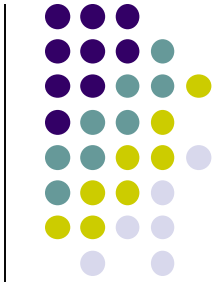
面向对象的基本概念：



7. 多态性和动态绑定

- 对象互相通信，即一个对象发消息给另一个对象，执行某些行为或又发消息给另外的对象，从而执行系统的功能。

- 发送消息的对象可能不知道另一个对象的类型是什么。



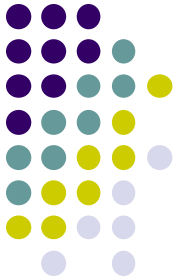
- 例如在C语言程序中使用命令 ClearInt ( ) 时要区分该命令适合一个整数，还是一个整数数组。但在C++情形，ClearInt( ) 对两者都适用，它自己判断对象是哪个。

- 这就是多态性(Polymorphism)。它意味着**一个操作在不同类中可以有不同的实现方式**。如ClearInt( ) 针对消息对象是 int array 还是int，其实现是不同的。

面向对象的开发过程

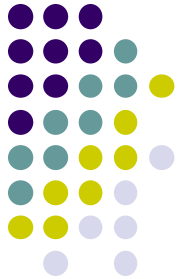
- 系统需求分析阶段
- 系统分析阶段
  - 对象静态模型
  - 对象动态模型
  - 对象功能模型
- 系统设计阶段
  - 系统设计
  - 对象设计
  - 模式设计
- 系统实现、测试、维护阶段

几种典型的面向对象方法



- Booch方法
- Jacobson方法
- Coad－Yourdon方法
- James Rumbaugh的OMT方法
- RUP*

几个重要的SE概念：软件工程工具

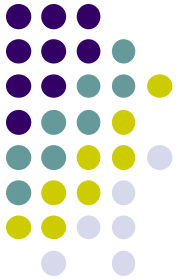


- 软件工程工具泛指软件开发全过程中使用的各种程序系统。具体来讲就是开发人员在系统分析、设计、编程、测试等过程中应用的一系列辅助软件工具。

- 软件工具通常由工具、工具接口和工具用户接口3部分组成。

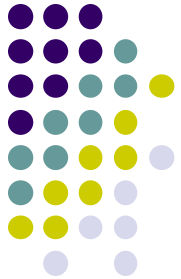
- 软件工具能在软件开发各个阶段帮助开发者控制开发中的复杂性，提高工作质量和效率。

CASE



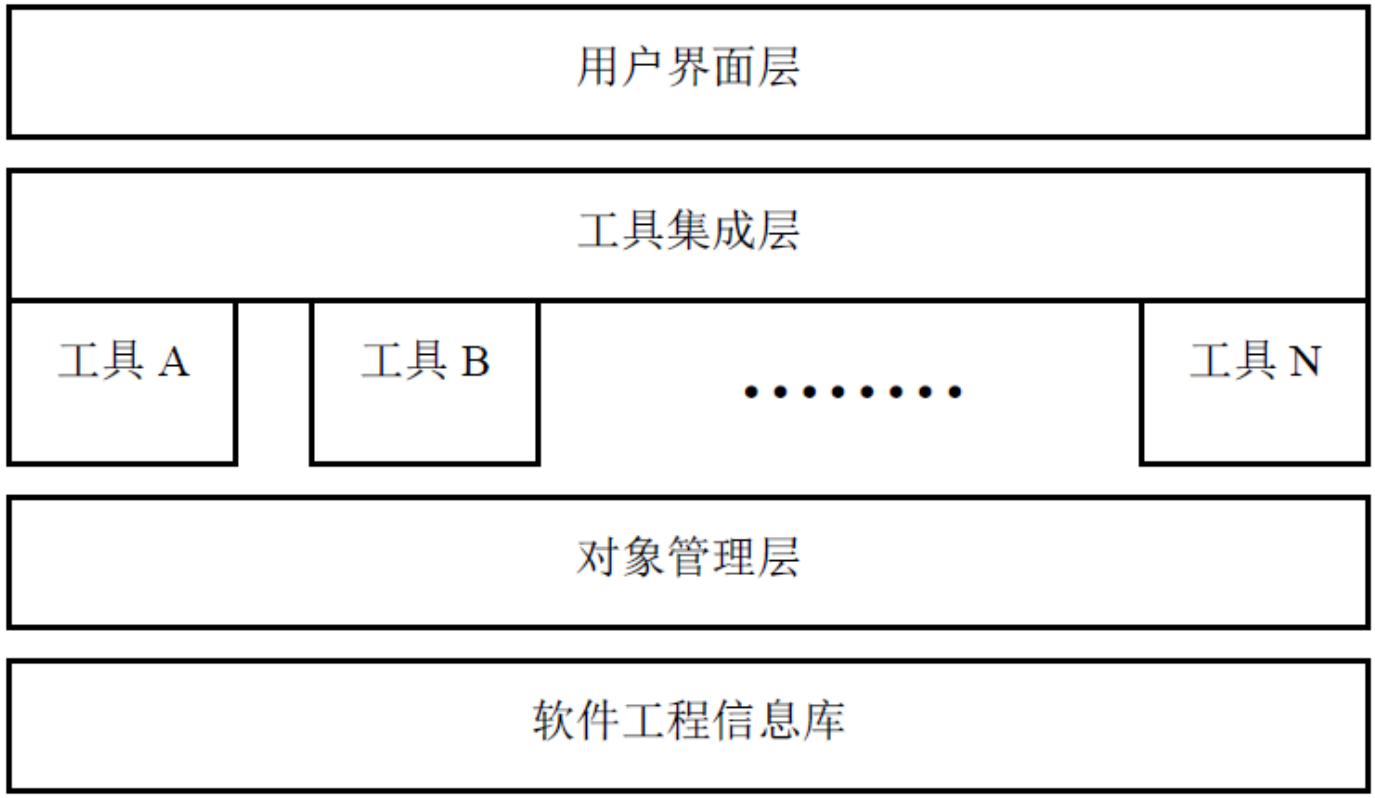
- 计算机辅助软件工程（Computer-Aided Software Engineering，CASE），原指支持管理信息系统开发的、由各种计算机辅助软件和工具组成的大型综合性软件开发环境；
- 已转化为支撑整套独立的方法论。注意CASE用作方法支持，其有效性需要通过“集成”达到；
- 一般的CASE环境需要网络的支持，允许若干软件工程师在这个环境中同时使用相同或不同的软件工具相互通信协同工作。

CASE的分类

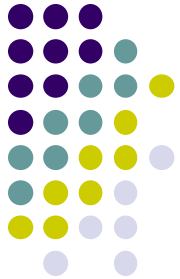


- 事务系统规划工具（Business System Planning Tools）
- 项目管理工具（Project Management Tools）
- 支撑工具（Support Tools）
- 分析与设计工具（Analysis and Design Tools）
- 程序设计工具（Programming Tools）
- 测试与分析工具
- 原型建造工具（Prototyping Tools）
- 维护工具（Maintenance Tools）
- 框架工具（Frame Tools）

集成化的CASE开发环境

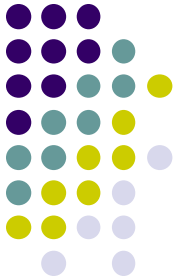


Conclusion



- 软件工程产生的背景
- 软件工程的定义
- 软件工程学中的三个核心元素：开发过程模型、方法和工具。
- 线性模型、增量模型、螺旋模型
- 结构化方法和面向对象方法
- 软件工程工具**CASE**的分类和集成化**CASE**环境。

软件系统分析与设计



- 分析和设计在软件工程中的重要性体现在其发生在软件系统的定义阶段。
- 需求分析阶段为整个软件开发项目的顺利进展和成功打下良好的基础；
- 设计阶段则为软件程序的编写打下良好的基础。
- 这两个阶段充分做好的话，可以从根本上减少整个软件开发过程中耗费的时间及相应的开发成本；反之，如果对系统的需求分析阶段重视，所开发的软件无法准确反映用户的需求甚至发生错误，所带来的损失将是不可估量的。