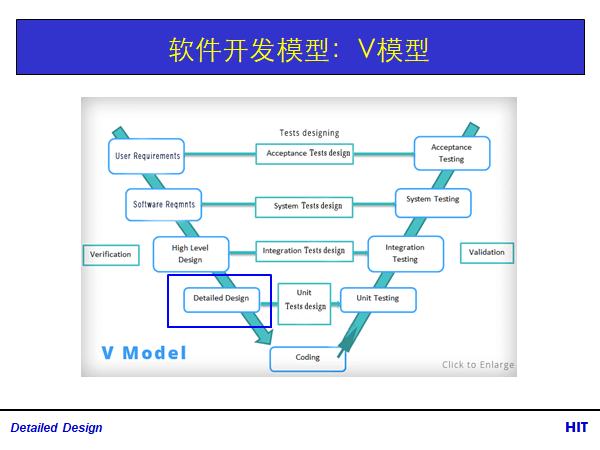
**第一章**

软件危机是指计算机软件开发维护过程中所遇到的一系列严重问题。

软件工程三要素：方法，工具，过程



软件工程基本原理#7

1. 用分阶段的生命周期严格管理。
2. 坚持进行阶段评审。
3. 实行严格的产品控制。
4. 采用现代程序设计技术。
5. 结果应能清楚的审查。
6. 开发小组人员应该少而精
7. 承认不断改变软件工程实践的重要性。

软件生命周期：软件定义，软件开发，软件维护。

软件定义：问题定义，可行性研究，需求分析。

软件开发：总体设计，详细设计，编码和单元测试，综合测试。

软件维护。

软件过程：为了获得高质量软件所需要完成的一系列任务框架。规定了完成各项任务的工作步骤。

瀑布模型：#3

1. 顺序性和依赖性。
2. 推迟实现的观点。
3. 质量保证的观点。

实际的瀑布模型带有反馈。 文档驱动。

快速原型模型：可看作是 不带反馈换的瀑布模型。

增量模型：一次完成一个功能。#2

1. 较短时间提供给用户部分工作的产品。
2. 给用户学习的时间，减少新软件的冲击。

螺旋模型：采用原型来降低风险，在每个阶段前都加上了风险分析。结合了瀑布模型和快速模型，是风险驱动的模型。

敏捷过程：#4

1. 个体和交互胜过过程和工具。
2. 可以工作的软件胜过面面俱到的文档。
3. 客户合作成果合同谈判。
4. 响应变化胜过遵循计划。

极限编程 通常有一名用户作为开发团队的成员。

**可行性研究 第二章**

可行性研究目的：用最小的代价在最短的时间里确定问题能否解决。目的不是解决问题，而是判断问题是否值得解决。

可行行方面：#3+X

1. 技术可行性
2. 经济可行性
3. 操作可行性
4. 法律可行性
5. 社会可行性

系统流程图：描绘数据在各个物理部件之间的流动。 描绘物理系统。

数据流图DFD + 数据字典DD：

描绘信息流和数据从输入到输出过程中的变换。是逻辑过程，不涉及物理部件。

数据流图和数据字典构成逻辑模型。

**需求分析 第三章：**准确地回答“系统应该做什么”

确定系统应该完成哪些工作，对目标系统提出轻完整、准确、清晰、具体的要求。

并且给出**软件需求规格说明书**。

。

针对结构化，要建立数据模型，行为模型，功能模型。

针对OO，建立对象模型，动态模型，功能模型。

与用户获取需求方法：#4

1. 访谈。
2. 面向数据流自顶向下逐步求精。
3. 建议的应用规格说明说明书。
4. 快速建立软件原型。

功能模型：DFD，层次图(H图)，IPO，HIPO。

数据模型：ER图（实体联系），层次方框图，数据规范化（范式）。

行为模型：状态转换图 和 OO的 动态模型使用的一样。

验证软件需求性4方面：

1. 一致性 2.完整性 3.现实性 4.有效性

**总体设计 第五章： “概要地说系统要**怎样做？” 也叫概要设计，初步设计。

总体设计的两大任务#2：

1. 划分系统的物理元素。
2. 设计软件结构，也就是确定系统中每个程序是由哪些模块组成，以及模块之间的相互关系。

总体设计两个阶段#2：

1. 系统设计阶段 2.结构设计阶段

总体设计的五大设计原理#5：

1. 模块化：将系统划分为独立命名、访问的模块，每个模块完成一个子功能，合起来是一个整体。
2. 抽象：抽出事物的本质而不考虑细节。
3. 逐步求精：为了集中精力解决主要问题，推迟对细节的考虑
4. 信息隐藏和局部化：模块内的信息对不需要他的模块不可以访问||关系密切的软件元素在物理上也放的彼此靠近。
5. 模块独立（是前四部的直接结果）

耦合#7：

1. 非直接耦合
2. 数据耦合：参数传简单数据类型
3. 特征耦合：参数穿数据结构
4. 控制耦合：传递控制信息
5. 外部耦合：共同访问全局变量
6. 公共耦合：共同访问数据结构
7. 内容耦合：访问另外模块

内聚#7：

1. 偶然性内聚：无直接关系
2. 逻辑性内聚：如if else
3. 时间性内聚：同一时间执行，仅此而已
4. 过程性内聚：按顺序执行
5. 通讯型内聚：各部分访问相同数据
6. 顺序性内聚：顺序执行，且前一个的输出是后一个输入
7. 功能性内聚：只完成一个功能。

总体设计中的启发规则#7：

1. 改进软件结构，提高模块独立性
2. 模块规模适中
3. 深度，宽度，扇入，扇出应该适当
4. 模块的作用域应该在控制欲之内
5. 力争降低接口模块的复杂度
6. 设计单入口单出口模块
7. 模块的功能应该可预测

描绘软件的结构工具：

层次图：表示调用关系。

HIPO：层次图，且每个方框加了IPO图与之对应，除第一层以外要加编号。

结构图：空心圆传信息，实心圆穿控制，有选择和循环调用。

DFD-SC是面向数据流设计方法 Jackson是面向数据结构的设计方法

**详细设计 第六章** 确定应该如何具体的实现所要求的系统，不编码，是“蓝图“。

1. 确定算法
2. 编写数据结构
3. 确定各模块的接口细节和用户界面。

人机界面设计问题#4

1. 系统响应时间
2. 用户帮助设施
3. 出错信息
4. 处理命令交互

过程设计工具：

程序流程图，判定树，判定表

结构化程序定义：代码块仅通过顺序选择循环三种结构，且只有一个入口和一个出口。

面向数据结构的设计方法 Jackson方法（数据结构导出程序结构）

**第七章** 实现 编码、测试统称实现

测试横跨生命周期的两部分。

测试目的：发现程序之中的错误。

测试步骤：

1. 模块测试（单元测试）
2. 子系统测试（集成测试）
3. 系统测试
4. 验收测试

集成测试方法：自顶向下，自底向上，混合。

优缺点（顶向下）：无需驱动程序，较早验证系统功能，较早发现上层接口错误。

缺点：需要存根程序（桩），下层错误发现晚。

确认测试（验收测试）：验证软件的有效性，以需求规格说明书为标准。用户主导。采用黑盒测试。

白盒测试：已知产品内部结构，检测是否按规格说明书正常运行。常用方法为逻辑覆盖和路径覆盖。

黑盒测试：直到产品功能，不知道内部结构，测试其功能能否正确使用。常用方法为等价类划分，边界值。

回归测试：系统加入了新模块后程序发生变化，用于保证由于调试或其他原因引起的错误不会导之非预期的软件行为或额外错误。

软件配置复查：保证软件配置所有成分齐全。

Beta测试：用户在场，开发者不在场，在不受控的真实环境，用户将问题记录下来，反馈给开发者。

独立路径：至少引入程序的一个新处理语句集合或一个新条件的路径||独立路径至少包含一条在定义该路径之前不曾用过的边。

**软件维护 第八章：**软件交互使用后，改正错误或满足新需求的过程。

维护的分类#4：

1. 改正型维护
2. 完善性维护
3. 适应性维护
4. 预防性维护

**第九章 面向对象方法学引论**

面向对象四个要点#4：

1. 万物皆对象
2. 相同类型的对象可以抽象成类
3. 子类和父类之间有继承关系，派生类具有父类的全部属性和服务。
4. 对象间通过消息传递相互联系。

对象模型，动态模型，功能模型的关系

* **对象模型定义了做事情的实体,描述了动态模型、功能模型所操作的数据结构。对象模型中的操作对应于动态模型中事件和功能模型中的函数。**
* **动态模型描述了对象的控制结构，它明确规定了什么时候(即在何种状态下接受了什么事件触发)做。**
* **功能模型指明了系统应该“做什么”，由数据流图和用例图组成，描述了对象模型中操作的含义、动态模型中动作的意义以及对象模型中约束的意义。**

**第十章 面向对象分析**

对象模型五个层次：

1. 主题层
2. 类与对象层
3. 结构层
4. 属性层
5. 服务层

[功能模型](https://www.baidu.com/s?wd=%E5%8A%9F%E8%83%BD%E6%A8%A1%E5%9E%8B&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)定义“做什么”，状态模型定义“何时做”，对象模型定义“谁对谁做”

建立对象模型五个步骤#5

1. 确定类与对象
2. 确定关联
3. 划分主题（可没有）
4. 确定属性
5. 确定继承关系

**十一章 面向对象设计**

软件设计四方面

1. 架构设计（物理架构 逻辑架构）
2. 数据结构设计
3. 模块设计
4. 接口设计

架构的组成部分

1. 构建 2.连接件 3.拓扑结构 4.约束 5.质量

软件架构的四大思想

1. 分层
2. 异步
3. 缓存
4. 分布式

经典软件架构风格

1. 调用返回
2. 数据仓库
3. 层次风格 C/S， B/S

* 领域模型（即对象模型）专注于问题领域的现实世界对象(模拟现实世界)。
* 职责驱动设计的核心思想：在对一个系统进行分析设计的时候，应当以职责为中心，根据职责分配行为。

Grasp#9：

1. 信息专家模式
2. 创建者模式
3. 低耦合
4. 高内聚
5. 控制器 GUI和底层逻辑分开，设一个控制器接受用户请求
6. 多态模式
7. 纯虚构模式 如A，B，C都要访问数据库，并且ABC有关系，在底层多加一个负责访问数据库的类。
8. 间接模式
9. 受保护变化：将不稳定的类设置一个稳定接口。

**十二章 面向对象实现**

1. 单元测试：最小单位是类和对象，不能单独测试某个操作。传统的单元测试关注模块算法细节和接口间数据流动，而OO单元测试关注封装在类中的操作和类的状态行为。
2. 集成测试：面向对象们没有层次控制结构。#2

**基于线程，基于使用**

1. 基于线程：集成对应系统的一个输入或者事件所需要的一组类，对每个线程分别测试，应用回归测试保证没有副作用。
2. 基于使用：测试几乎不适用服务器的独立类构造系统，在独立类测完之后测试使用独立类的依赖类，直至测完。
3. 确认测试：和传统一样，集中检查用户可见的动作和用户可识别的输出。从动态模型和描述系统行为的脚本（用例描述）可以到处测试用例，也可以用黑盒测试。

传统测试方法在OO测试#2

1. 白盒测试：用于测试类级别的
2. 黑盒测试：用于集成测试，确认测试。

单元测试分：

1. 随机测试
2. 划分测试
3. 基于故障的测试

集成测试：

针对类间协作进行测试

1. 多类测试

1) 对每个客户/UI类，为其生成一系列测试序列来覆盖所有操作。

2) 对这些操作所生成的每个消息，确定接收消息的协作类/服务器对象中的对应操作。

3）把每个2）的操作，结合进测试序列中。对每个产生消息的操作，继续2）

1. 动态模型导出测试用例

系统测试

基于情景（场景）的测试是OO系统测试的主流方法。

* 用例场景要通过描述流经用例的路径来确定，这个流经过程要从用例开始到结束遍历其中所有基本流和备选流。

**十三章**

能力成熟度模型：用来评价软件机构软件过程能力成熟度的模型

1. 初始级：没有健全的软件工程管理制度
2. 可重复级：有了基本的项目管理过程（过程模型），可以追踪成本，进度，功能，质量。
3. 已定义级：定义了完整的软件过程（过程模型），已文档化，标准化
4. 已管理级：项目的活动是可度量的、
5. 优化级：软件过程是可优化的

软件配置管理：

1. 软件配置项：程序，数据，文档。统称为软件配置，每一项叫软件配置项。
2. 基线：是一个软件配置管理概念。是经过了正式复审的软件配置项，只有通过正式的变化控制才能改变它，可以作为下一步开发的基础。