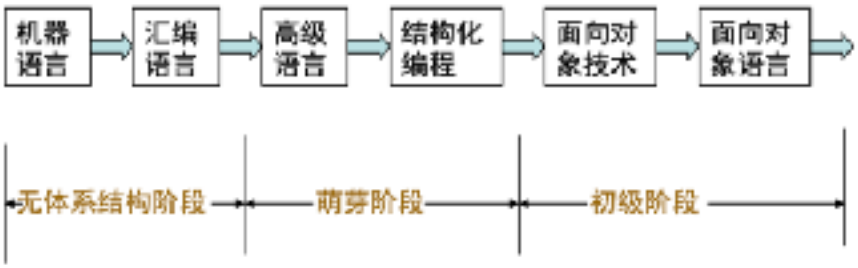
50%来自理论书，主要看ppt
30-40分要背，

开发过程，理解
问题定义 -> 需求分析 -> 软件设计 -> 软件编码 -> 软件测试 -> 维护升级

演化，别看
定义，要看
没有公认的统一定义原因：历史短，不完善；不同研究者理解不同；定义影响太大
• 版本1: Booch & Rumbaugh & Jacobson 定义
软件体系结构 = {组织，元素，子系统，风格}
• 版本2: Bass定义
软件体系结构是一个或多个结构，包括软件构件(component)、构件的外部可视属性 (property)和构件之间的关系 (relationship)。
• 版本3: Shaw定义
四种分类法
–结构模型 体系结构由构件、构件间连接及其他一些方面组成。
–框架模型 与结构模型类似，强调整个系统的结构。
–动态模型 强调系统的行为质量。
–过程模型 体系结构是一系列过程的结果。
• 版本4: Garlan & Shaw 定义
体系结构={构件，连接件，约束}
• 版本5: Perry & Wolf 定义
体系结构 = {元素，形式， 准则}
元素:
– 处理元素:负责完成数据加工
– 数据元素:作为被加工的信息对象
– 连接元素:用于将体系结构的不同部分组合连接起来
形式:
– 专有特性:用于限制体系结构元素的选择
– 关系: 用于限制体系结构元素组合的拓扑结构
• 版本6: Garlan & Perry 定义
体系结构: 是一个程序/系统各构件的结构、它们的相互 关系，以及进行设计的原则和指导方针。
• 版本7: Soni & Nord & Hofmeister 定义
体系结构包括 至少 4 种具体形态:
– 概念体系结构
– 模块互连体系结构
– 执行体系结构
– 代码体系结构
• 版本8: Boehm 模型
体系结构 = {构件，连接件，约束，不同人员的需求，准则}

发展阶段，好好背
• 一无体系结构”设计阶段[1946 ENIAC计算机问世]（软件开发汇编为主；编程时少数人的游戏；应用以军用和科学计算为主；规模小，没有考虑系统结构）

- 萌芽阶段[1970 - 1980]（结构化编程思想；出现结构化程序设计语言如PASCAL；标题已经成为系统开发中的明确概念）
  - 优点：应用走向商用、民用；软件规模数万数十万；软件开发成为职业，出现程序员
  - 困境：硬件更快，软件更胖，存在鸿沟；软件规模扩大导致质量下降；大量重复劳动，软件重用任重道远
- 初级阶段[80年代初 - 90年代初]（提出面向对象，出现支持软件开发的面向对象方法；面向对象语言出现如c++；软件以对象为基本元素，系统由离散对象构成；统一建模语言uml提出）
  - 优点：易于表述概念，交流；作为描述、分析和建立文档的手段，提高了软件的易读性、可维护性和可重用性；继承、封装、多态为软件重用提供可行手段
  - 困境：基于对象的重用属于代码重用，需要对类的内部设计、实现有清晰的认识；软件的升级、维护等需要重新编译、调试，无法实现动态升级；难以实现设计重用，缺乏描述体系结构的语言。
- 高级阶段 [90年代初 - 现在]（开发强调构建化技术和体系结构技术；提出COM、EJB、WebService；软件 = 构件 + 基于体系结构的构件组装；体系结构作为开发文档和中间产品，开始出现在软件开发过程中）
  - 优点：构件实现了可执行二进制代码的重用；构件的实现与实现语言无关；构件可以单独开发、变异、调试；基于构件的软件系统可以实现动态升级和维护；体系结构提供了设计重用的可能性，设计重用是比构件重用更加抽象、更加高级的重用



体系结构意义，尽量背

- 有利于系统分析：可以进一步抽象问题；它是软件系统相关各方交流的平台
- 有利于软件开发：它是系统实现的基本约束；决定了开发和维护项目的组织结构；有利于控制软件质量
- 有利于软件复用：构件复用、软件子系统复用、设计复用
- 有利于软件系统的演化：局部的；非局部的；体系结构级的

体系结构的目标，上课没提，爱看不看

- 便于维护和升级，因而应该是模块化的
- 设计应该是便于移植的(移植比重新设计花费要小的多)
- 设计应该具有适用性
- 设计过程应该收到理性的控制
- 设计应该表现出概念的完整性（内在结构；外在表现）

构建概念，理解

- 定义：构件是具有一定功能、可明确辨识的数据单位或计算单元
- 特点：语义完整；语法正确；可重用；封装性；独立性

• 特征信息：计算功能；额外功能特性；结构特性；家族特性

常见构建，7个要背

模块，对象，过滤器，过程，数据文件，数据库，文档

构建分类，别看

构建开发，读读

– 从现有构件库中获取符合要求的构件，直接使用或者经过简单修改，得到可重用的构件。
– 通过遗产工程，将具有潜在重用价值的构件提取出来
– 从市场购买现成的商品软件或者第三方厂商开发的中间件
– 基于COM、CORBA、WebService等技术规范，自行开发所需的构件。

组织和组装不要

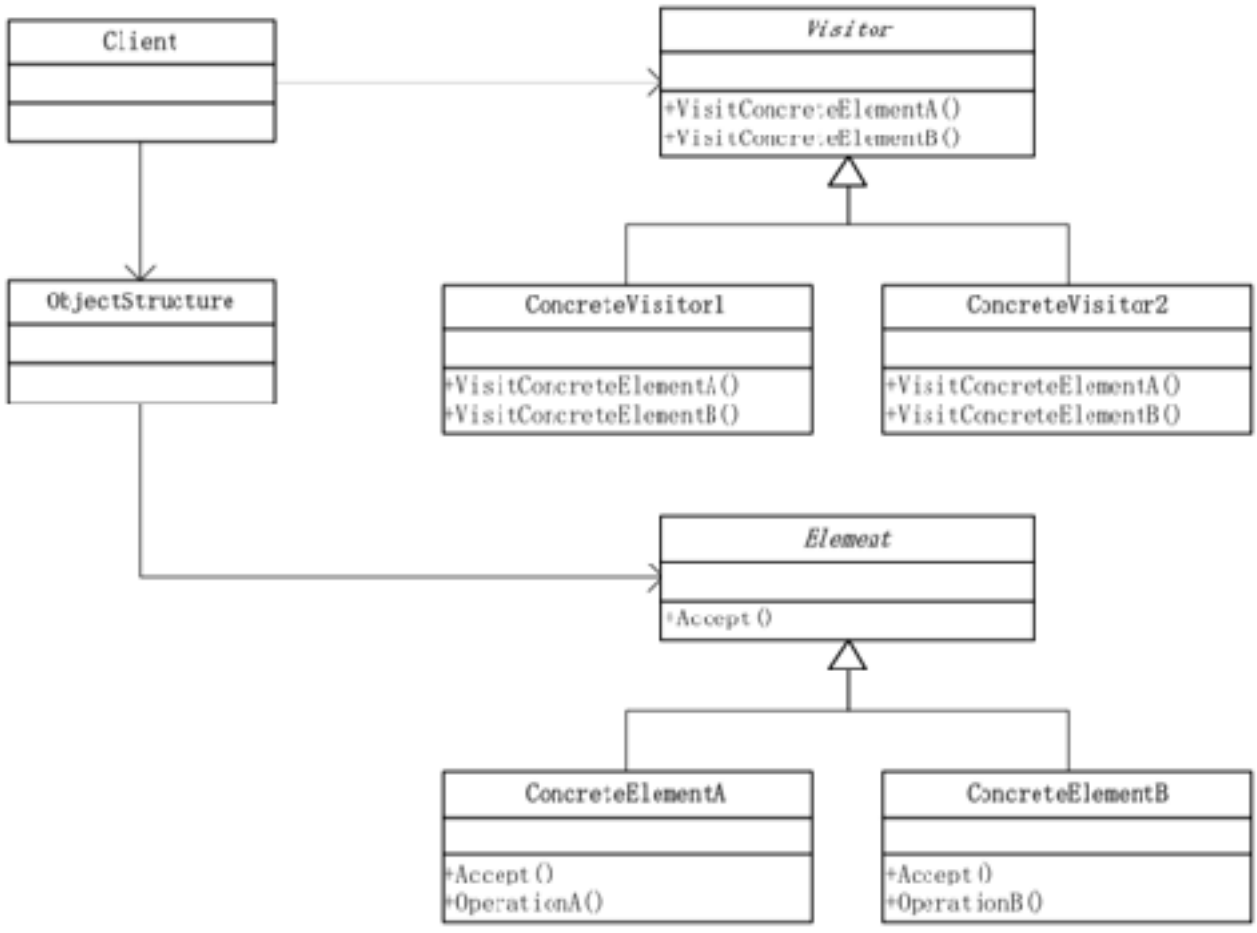
构建的例子，不看

设计模式，visitor，仔细看ppt

设计模式四个基本要素，要背

– 模式名称(Pattern Name): 一个助记名，它用一两个词来描述模式的问题、解决方案和效果。
– 问题(Problem): 描述了应该在何时使用模式。
– 解决方案(Solution) 描述了设计的组成部分，它们之间的相互关系和各自的职责和协作方式。
– 效果(Consequence)：描述了模式应用的效果及使用模式应该权衡的问题。

visitor设计思想



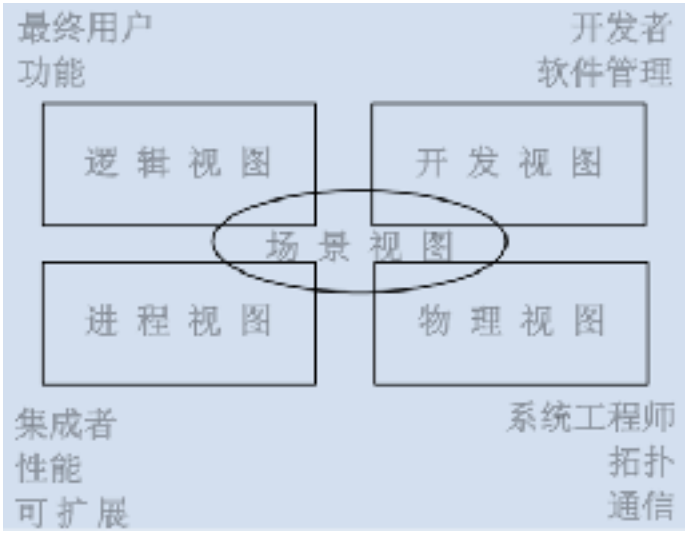
定义：表示一个作用于某对象结 构中的各元素的操作。它使我们可以在不改变各元素 的类的前提下定义作用于这些元素的新操作。访问者 模式是一种对象行为型模式。
优点：访问者模式使得增加新的操作变得很容易；访问者模式将有关的行为集中到一个访问者对象中， 而不是分散到一个个的节点类中；访问者模式可以跨过几个类的等级结构访问属于不同的等级结构的成员类。
缺点：破坏封装（访问者模式要求访问者对象访问并调用每一个节点对象的操作，这隐含了一个对所有节点对象的要求:它们必须暴露一些自己的操作和内部状态。不然，访问者的访问就变得没有意义。由于访问者对象自己会积累访问操作所需的状态，从而使这些状态不再存储在节点对象中，这也是破坏封装的）；增加新的节点类变得很困难（每增加一个新的节点都意味着要在抽象访问者角色中增加一个新的抽象操作，并在每一个具体访问者类中增加相应的具体操作）
使用场景：定义对象结构的类很少改变，但经常需要在此结构上定义新的操作；对象需要添加很多不同的并且不相关的操作，而我们想避免让这些操作“污染”这些对象的类。访问者模式使得我们可以将相关的操作集中起来定义在一个类中。当该对象结构被很多应用共享时，用访问者模式让每个应用仅包含需要用到的操作。
层次结构：访问者层次结构，提供了抽象访问者和具体访问者；元素层次结构，提供了抽象元素和具体元素。

符号执行
状态定义：每个状态记录执行过程中的一条程序路径.
状态内容：pc计数器；符号(符号变量)；具体变量和具体值；约束(Constraint)

KLEE运行流程：src —(编译连接)—> bin —(符号执行)—>{用例，约束，属性} —(分析)—>结果

第二种设计架构：不记录状态；符号执行与具体执行同时进行；每次先具体执行一条路径, 再由符号执行求 解出另外路径的输入, 来执行另外一条路径 ..... 直到所有路径都被执行到为止.

视图4+1模型



- 逻辑视图
  - 强调系统的功能性需求
  - 使用类、对象来表示。从问题域出发，采取面向对象分析方法，按照抽象、封装、继承的原则，获得代表问题抽象的对象和类

- 进程视图
  - 关注系统的非功能性需求，如性能，可用性。强调系统的并发性、分布性、容错，以及逻辑视图的类如何对应到进程
  - 进程是由一系列的任务构成的执行单元软件被分解成为众多的任务，这些任务分成两种:主任务和辅任务。主任务针对软件的主要功能，辅任务是为了完成主任务而引入的，如缓存、延时等
  - 主任务之间的通信方式包括基于消息的 同步/异步、远程过程调用、事件广播等。
  - 辅任务之间通信主要通过共享内存等方式
- 开发视图
  - 强调在开发环境下，如何将软件划分成为 不同的模块或者子系统，以便分配给程序员或团队开发。
  - 通常这样的子系统都是以层次方式组织，每一层通过接口给上一层提供服务。
  - 开发视图遵循的三大原则:分割、编组、 可视化。
  - 设计开发视图的时候主要考虑到以下因素：开发的难易程度、软件管理、重用和通用性、工具及开发语言的限制。
- 物理视图
  - 如何将软件映射到不同的物理硬件上去。
- 场景视图
  - 把前4个视图串联在一起的纽带
  - 场景是发现软件体系结构元素的驱动者。
  - 场景是体系结构设计结束后验证和演 示的最好角色，不管是书面形式还是作为测试体系结构原型的起始点。

四个架构
- 逻辑架构：层、子系统；类；接口；协作关系
- 开发架构：程序包（含SDK和框架）；文件组织结构；编译依赖关系；目标单元
- 运行架构：进程；线程；主动类；通信方式
- 物理架构：安装单元；节点；网络；基础设施选型

后面就看ppt吧～

visitor设计思想：应用场景，部分……
雇员的例子不看，编译器的例子仔细看
优缺点不要看漏
while语言不看
后面分析、代码重在理解

视图4+1模型，好好看
各个视图给什么人用，识别图是什么视图
模型文档不要了，迭代算法不要了
四个架构理解就好
5种视图比较，理解
视图和uml今年不考
用例，风格这章不考

符号执行

符号状态的部分
算法理解
klee实现不考
符号执行的内存结构看看
第二种设计架构，看看理解

tansorflow
cnn和rnn不考
读例子，看数据流
操作过一下，会话/设备看一下，理解

By 15吴静琦