## 什么是结构化分析方法

1 建模技术

2 目标系统是数据流和一些转换构成

3 核心是数据字典，它描述了目标系统中使用的和生产的数据对象，三种图：ERD，DFD，STD

## 结构化分析模型的组成

1 数据建模（ERD实体关系图）和对象描述（E-R中对象属性）

2 功能建模和数据流图（DFD数据流图）

3 基本加工逻辑说明（PSPEC加工规格说明：描述数据流图的每个功能）

4 行为建模（STD状态迁移图）（Statement Transfer Diagram）

5 数据字典（核心）



## 基本思想（自己）

抽象

忽略内部复杂性，只关注问题与外界的联系

分解（自顶向下逐层）

将问题分解为较小的问题，直到最底层问题足够简单为止

（控制复杂性的两个手段）

## 什么是功能建模（自己）

用抽象模型的概念，按照软件内部数据传递、变换的关系，自顶向下逐层分解，直到找到满足功能要求的所有可实现的软件为止。

## 行为建模（自己）

描述状态以及导致状态改变的事件来表示系统行为，每个状态代表系统的一个行为模式。

## 结构化分析的方法关注的是什么

关注的是过程

结构化分析方法是以过程为核心的，逻辑和数据可分离的，数据在逻辑的推动下变化的过程。数据推动逻辑变化，最终找到问题的解。

## 先有对象还是先有类

答案：先有对象

从哲学角度来看，是先有对象，后有类；先有子类，后有父类。是一种自底向上形成的体系。

## 抽象是有层次的

抽象层次树理论是OO（面向对象）中许多内容的理论基础。

如OO中的继承（父与子的关系）和多态

节点CNode非叶结点，也非根节点，CNode继承于父节点PNode，或者说PNode是CNode的泛化

把CNode看作任何一个Anode祖先节点，并当作Anode使用，叫做多态。

## 什么是耦合

我们在设计中是要降低耦合，而不是取消耦合。

## 泛化耦合

由于泛化（继承）关系的存在，在两个有祖孙、父子关系的类之间形成的一种逻辑关联。

## 聚合

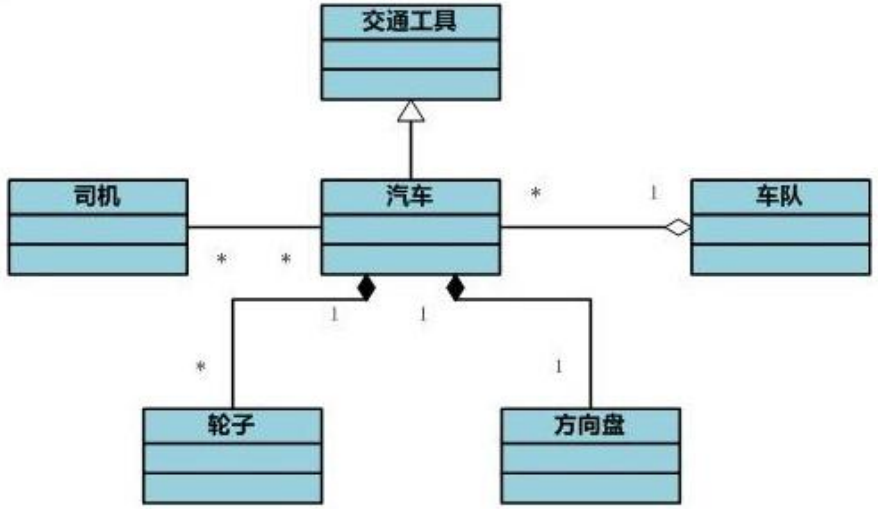
一种弱的拥有关系，体现了A可以包含B，但B不是A的一部分

## 组合（最强）

一种强的拥有关系，体现了严格的整体与部分关系，部分和整体具有一样的生命周期

## 依赖（最弱）

由于逻辑上相互协作，而形成的一种关系



汽车和交通工具是泛化耦合

轮子和方向盘组合于汽车

汽车聚合成车队

汽车和司机具有依赖关系

## OO面向对象

面向对象的方法是运用对象、类、继承、封装、聚合、消息传送、多态性等构造系统的软件的开发方法。

面向对象=对象+类+继承+通信

采用这四个概念开发的软件系统是面向对象的。

## 为什么采用面向对象的方法（自己）

1 面向对象的方法符合人们的认知规律

2 开发的系统容易维护，体系结构容易理解、扩展。

3 继承机制支持软件的复用

并没有摒弃面向过程的方法。

## 对象是什么

对象是一组属性以及这组属性上的操作的封装体

属性：通常是一些数据（有时也可以是另一个对象）只能通过该对象提供的操作来存取修改

操作：（也称方法或服务），规定对象的行为，表示对象所能提供的服务

封装：内部实现对用户隐蔽，使对象的使用者和生产者分离，对象的定义和实现分开

对象是由对象名（标识），属性，操作三部分组成。

## 继承（实现代码的重用，自己）

继承性的含义

被继承的对象共享所在类的结构、操作和约束等语义特性

继承的传递性

多重继承

继承性的作用

使所建立的软件系统具有开放性

信息组织和分类的有效方法

提高代码重用率和可靠性，降低开发工作量

抽象类

用斜体表示，也可以在后面加{abstract}

单一继承

子类只有一个父类

多重继承

子类可有多个父类

Java不支持多重继承，可以用接口实现

C#不支持多重继承，可以用接口实现

## 多态性和多态绑定（自己）

多态性

是指同一个操作作用于不同对象上可以有不同解释，并产生不同执行结果。

例如

“画”，作用在“矩形”对象，则画出一个矩形，作用在“圆”对象上，则画出一个圆

动态绑定

在运行时，当一个对象请求服务时，要根据接收对象的具体情况将请求的操作与实现的方法进行连接，即动态绑定

## 永久对象（持久层的设计，如hibernate，自己）

永久对象

是指生存期可以超越程序的执行时间而永久存在的对象

## 对象的特点

对象的特点

对象是主动的

以数据为中心

属性和操作封装

独立处理自身的数据，通过消息传递进行通信，具有并行的性质

是OO软件的基本模块，低耦合，高内聚

能够唯一识别，行为丰富

## 面向对象分析

面向对象分析（自己）

就是抽取和整理用户需求并建立问题域精确模型的过程

面向对象需求分析过程和模型

需求描述 用例图，用例模板

静态模型（对象模型） 类图，之间的关系（核心）

动态模型（交互次序） 顺序图，协作图，活动图等，怎样耦合的（核心）

功能模型（数据交换） 数据流图

静态模型的构建方法

找出类——对象并进行筛选（动词：事件方法。名词：类的表示）

找出对象之间的关联

划分主题

找出对象的属性

用继承简化组织对象类

迭代并反复提炼模型

如何建立动态模型

编写脚本（系统某一执行期间内出现的一些列事件）

设计用户界面

画事件跟踪图（和序列图很像）

画状态图

审查动态模型

## 功能模型（自己）

为什么要功能建模

功能模型表示了数据之间的依赖关系，以及有关数据的处理功能，由一组数据流图组成

在建立了对象模型和动态模型之后再建立功能模型

功能模型说明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

由多个数据流图组成与结构化分析中的DFD相似

如何建立

画出基本的系统模型图

画出功能级的数据流图

描述处理框功能

## 结构化和OO在几方面的比较

从概念方面看

结构化是功能的集合，通过模块以及模块与模块之间的分层调用关系实现

OO是事物的集合，通过对象以及对象和对象之间的通讯联系实现

从构成方面看

结构化=过程+数据，以过程为中心

OO=（数据+相应操作）的封装以对象为中心

从运行控制方面看

结构化采用顺序处理方式，由过程驱动控制

OO采用交互式、并行处理方式，由消息驱动控制

从开发方面看

结构化的重点是设计

OO的重点是分析

从应用方面看

结构化适合数据类型简单的数值计算和数据统计管理软件的开发

OO适合大型复杂的人机交互软件和数据统计软件的开发

## 有可能是辨析题（自己）

OO的出现并不意味着结构化的消亡

分析阶段（分析模型）可能会和设计阶段（设计模型）有重合

对于不同的分析人员，OO的分析模型的描述可能不同

分析模型的建立需要多次反复迭代和细化

分析模型的质量直接影响后续阶段的质量和最终产品的成败