

《软件工程》

2023年春

第五讲 需求分析

内容

一．软件需求概述

二．软件需求获取

三．软件需求分析



一、软件需求概述

内容

1. 软件需求和需求工程
2. 结构化需求分析方法学
3. 面向对象需求分析方法学
4. 需求工程的输出和评审

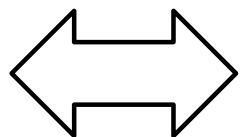
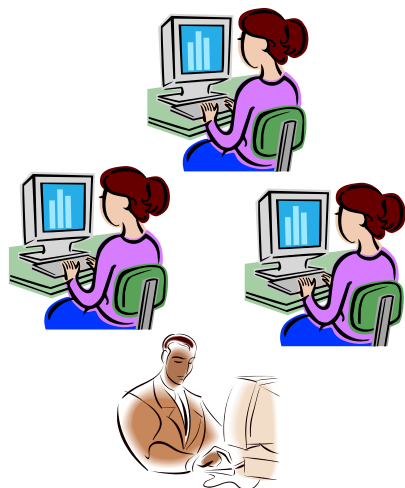


1.1 软件开发的本质（1/2）

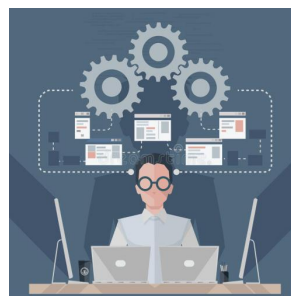
领域相关

技术相关

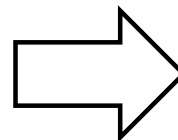
融合需求和技术



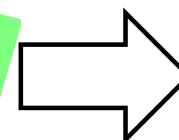
期望和
要求



软件工程师



解决
方案

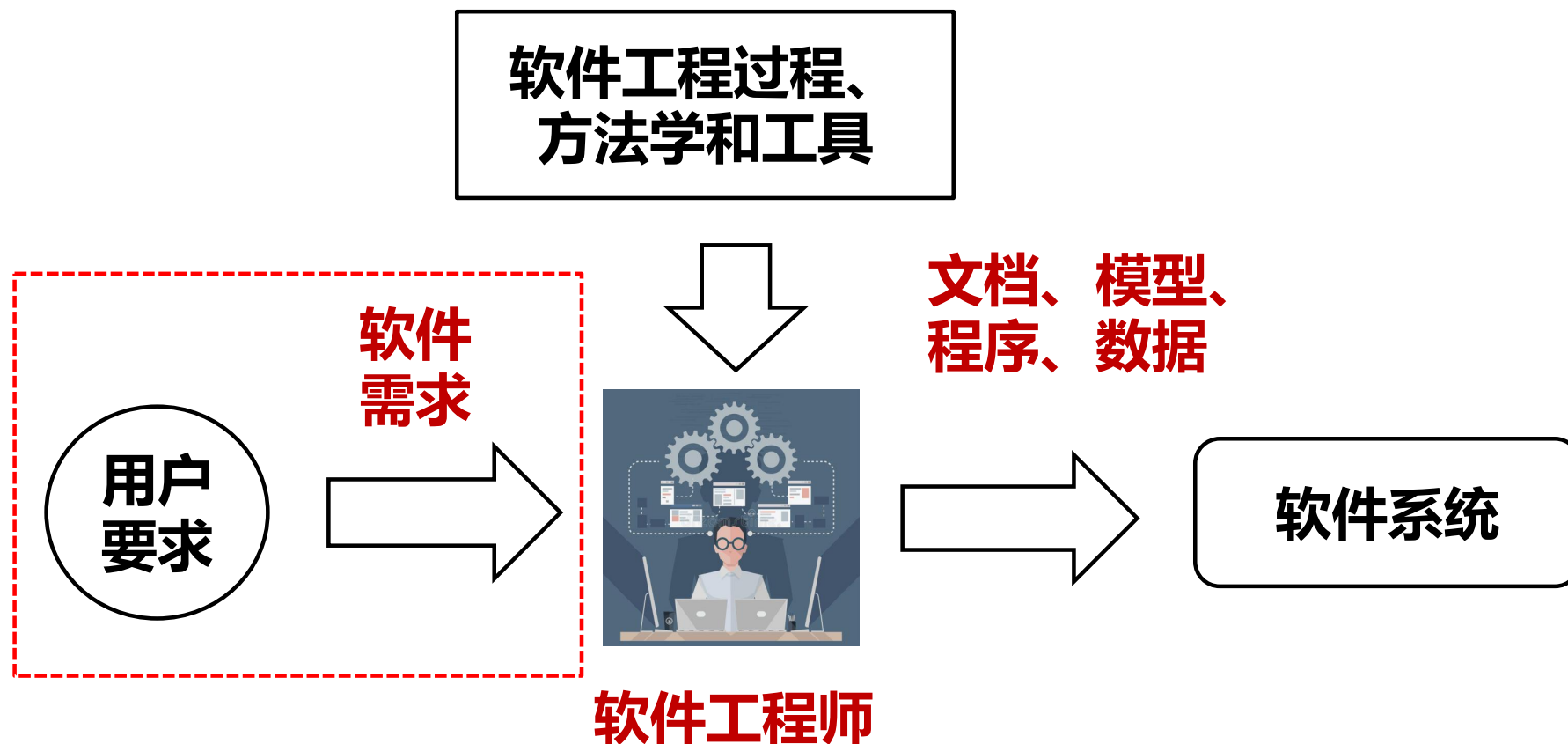


软件
产品

软件工程师

软件利益相关方

1.1 软件开发的本质（2/2）



开发软件系统的前提是要明确利益相关方的期望和要求，即软件需求

1.2 软件系统的利益相关方

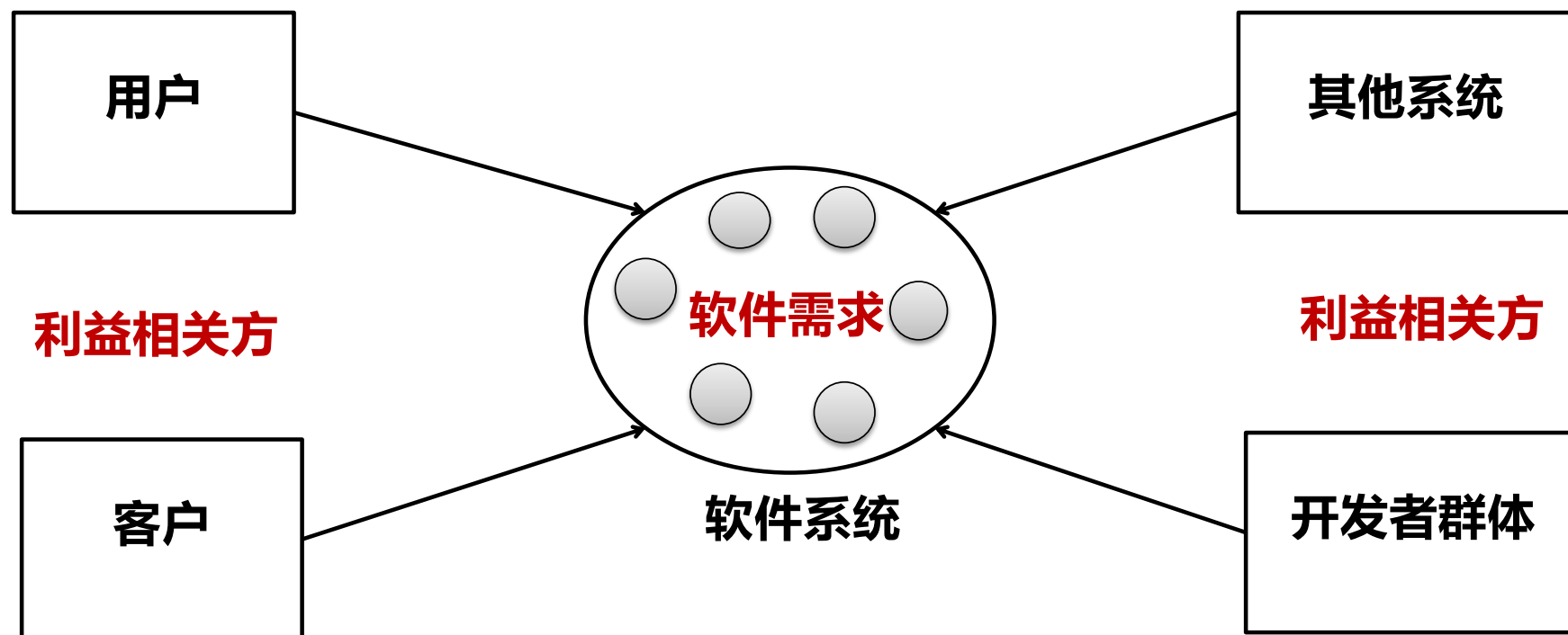
□何为利益相关方(stakeholder)

- ✓从软件系统中**受益**或与软件系统**相关**的**人、组织或者系统**
- ✓**受益**：使用、获益、盈利
- ✓**相关**：发生操作和交互、存在关联性

□软件利益相关方的表现形式

- ✓**用户**：最终使用软件的人
- ✓**客户**：从中获取利益的组织
- ✓**系统**：与待开发系统进行交互的系统
- ✓**开发者**：负责开发软件系统的人

软件需求及其利益相关方

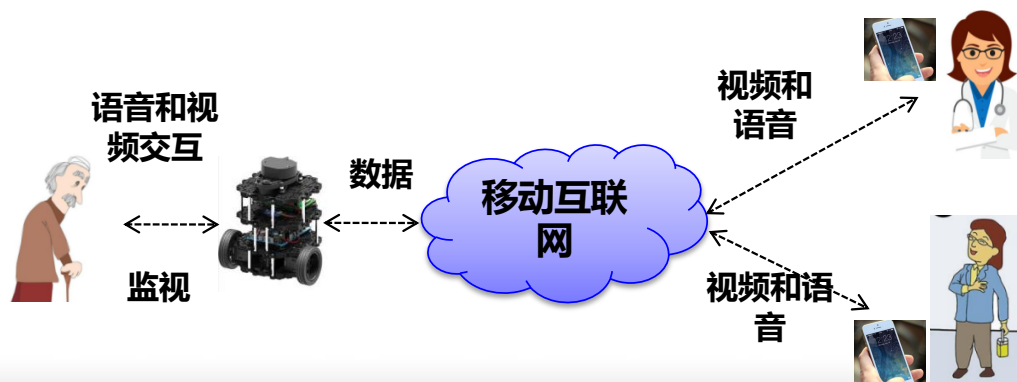


软件利益相关方会站在自身的角度对软件提出要求

软件案例: 空巢老人看护系统

□ 软件密集型信息系统，通过软件将自主机器人、智能手机等设备相结合，对家中独居老人进行监护

- ✓ 跟踪老人在家情况
- ✓ 老人与远端的家属进行语音和视频交互
- ✓ 发现和通告异常情况（如摔倒、突发疾病）
- ✓ 将老人在家状况（如图像和视频）和异常信息传送到远端家属或医生的智能手机上
- ✓ 通过语音进行呼叫和报警
- ✓ 提醒老人按时服药和保健
- ✓



示例：空巢老人看护软件的利益相关方

□用户

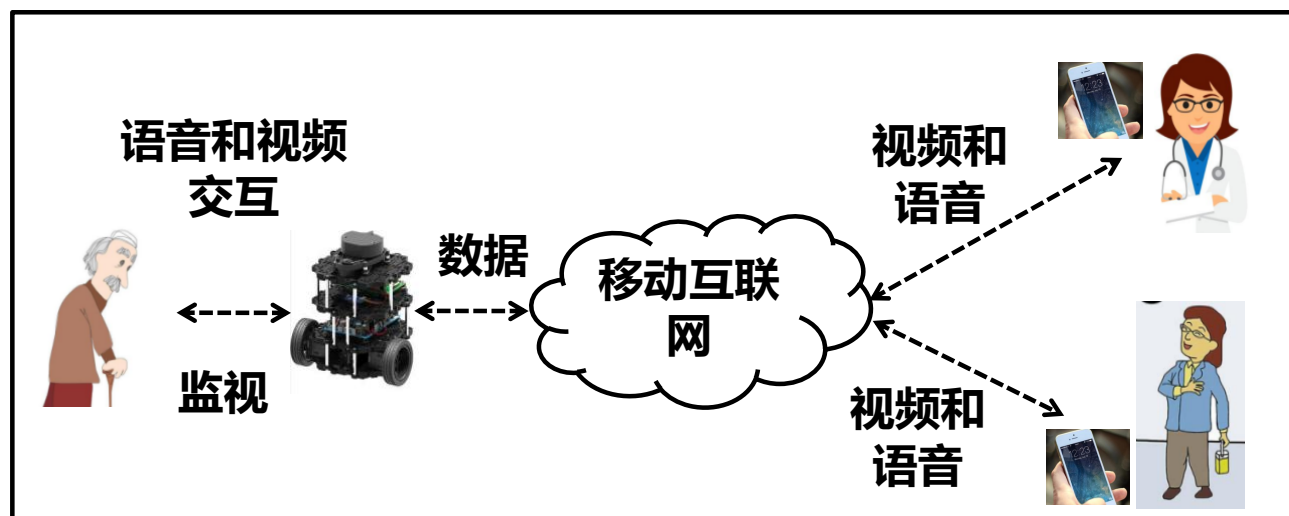
- ✓老人
- ✓家属
- ✓医生

□客户

- ✓投资方

□系统

- ✓机器人



1.3 何为软件需求(Software Requirement)?

- 从软件本身的角度，**软件需求**是指软件用于解决现实世界问题时所表现出的**功能和性能**等方面的要求
- 从软件利益相关方的角度，软件需求是指软件系统的**利益相关方**对软件系统的**功能和质量**，以及软件运行环境、交付进度等方面提出的**期望和要求**
- 软件需求刻画了软件系统能做什么（**What to do**），应表现出怎样的**行为**，需满足哪些方面的**条件和约束**等要求

软件需求的类别

□软件功能性需求(Functional)

- ✓能够完成的**功能**及在某些场景下可展现的**外部可见行为或效果**

□软件质量方面的需求(Quality)

- ✓**外部质量属性**，外部可展现的，用户、客户等会非常关心，如运行性能、可靠性、易用性等
- ✓**内部质量属性**，隐藏在内部的，软件开发工程师会非常关心，如可扩展性、可维护性、可理解性

□软件开发约束性需求(Constraint)

- ✓开发成本、交付进度、技术选型、遵循标准等方面提出的要求

示例：空巢老人看护软件的需求

□功能性需求

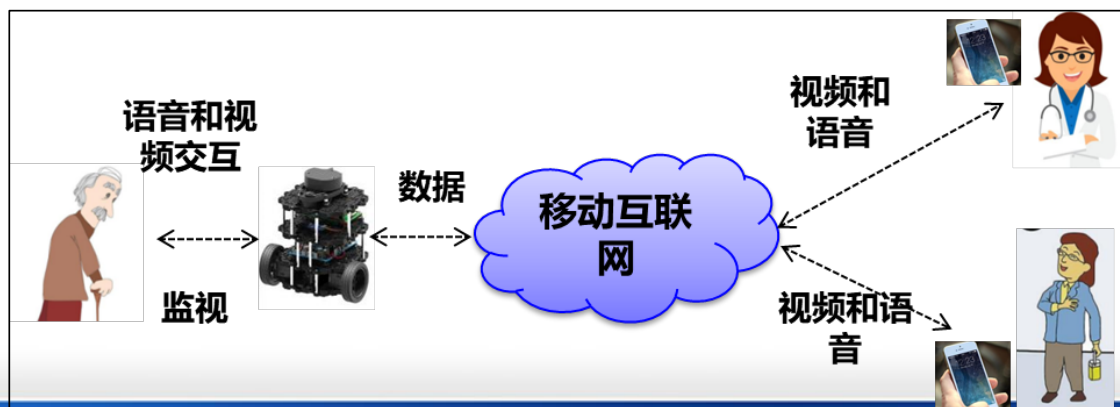
- ✓自主跟随老人、获取老人图像和视频信息、检测老人是否摔倒等

□质量方面的需求

- ✓始终保持在2米的安全距离，对机器人的控制在2秒内响应等

□约束性需求

- ✓成本不能超出50万元，要求半年内交付使用等等



这些需求都是谁提出来的？

软件需求的特点（1/2）

□隐式性

- ✓来自于利益相关方，它隐式存在
- ✓很难辨别，甚至会遗漏掉

□隐晦性

- ✓在利益相关方的潜意识之中，**不易于表达出来，难以获取**
- ✓所表达的软件需求存在**模糊性、歧义性、二义性**

□多源性

- ✓存在多个的利益相关方
- ✓存在相冲突和不一致的软件需求



软件需求的特点（2/2）

□易变性

- ✓用户对软件的期望和要求也会经常性地发生变化
- ✓在整个生命周期都会发生变化

□领域知识的相关性

- ✓软件需求的内涵与软件所在领域的知识息息相关
- ✓“12306”与铁路旅客服务领域相关

□价值不均性

- ✓不同的软件需求对于客户或用户而言所体现的价值是不一样的
- ✓主要和次要、核心和外围需求



如何从利益相关者获取完整、清晰和一致软件需求是一项挑战！

软件需求的质量要求（1/2）

□有价值（Valuable）

- ✓基于计算机软件的解决方案，有效提高问题解决的效率和质量，促进相关领域的业务创新

□正确（Right）

- ✓反映利益相关方的期望，不能曲解或误解他们的要求

□完整（Complete）

- ✓不能有遗漏或丢失

□无二义（Unambiguous）

- ✓软件需求的描述应该是清晰和准确的

软件需求的质量要求 (2/2)

□可行 (Feasible)

- ✓在技术、经济等方面应该是可行的

□一致 (Consistent)

- ✓不应存在冲突

□可追踪 (Traceable)

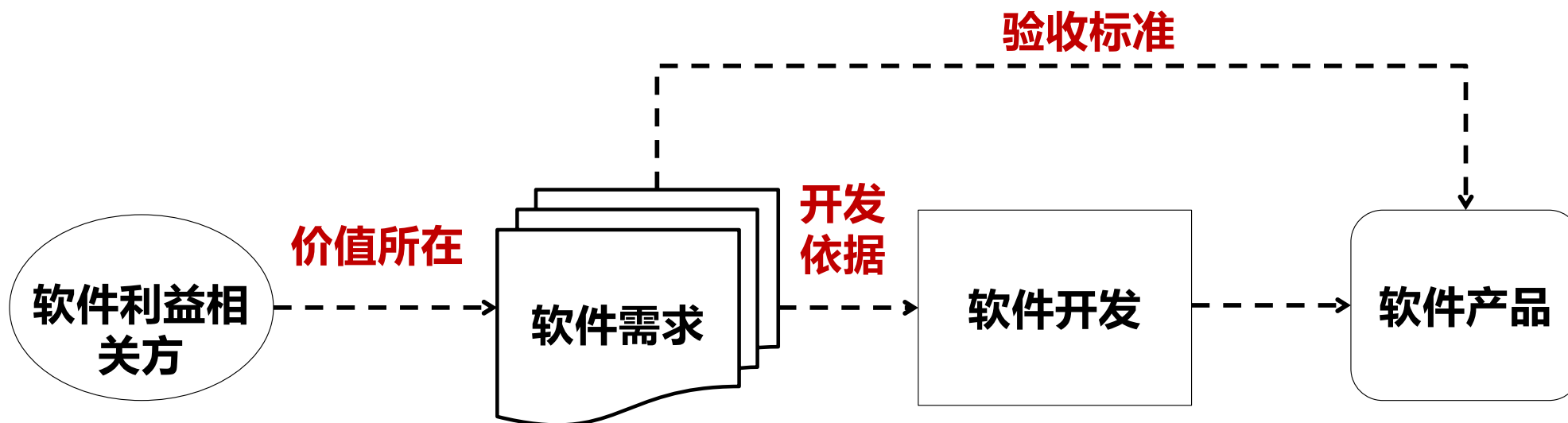
- ✓可追踪到其源头

□可验证 (Verifiable)

- ✓可找到某种方式来检验软件需求是否在软件系统中得到实现

软件需求的重要性

- 软件的**价值所在**
- 软件开发的**基础和前提**
- 软件验收的**标准和依据**



思考和讨论

□如果软件需求存在以下问题会给软件开发带来什么样的挑战，产生怎样的后果？

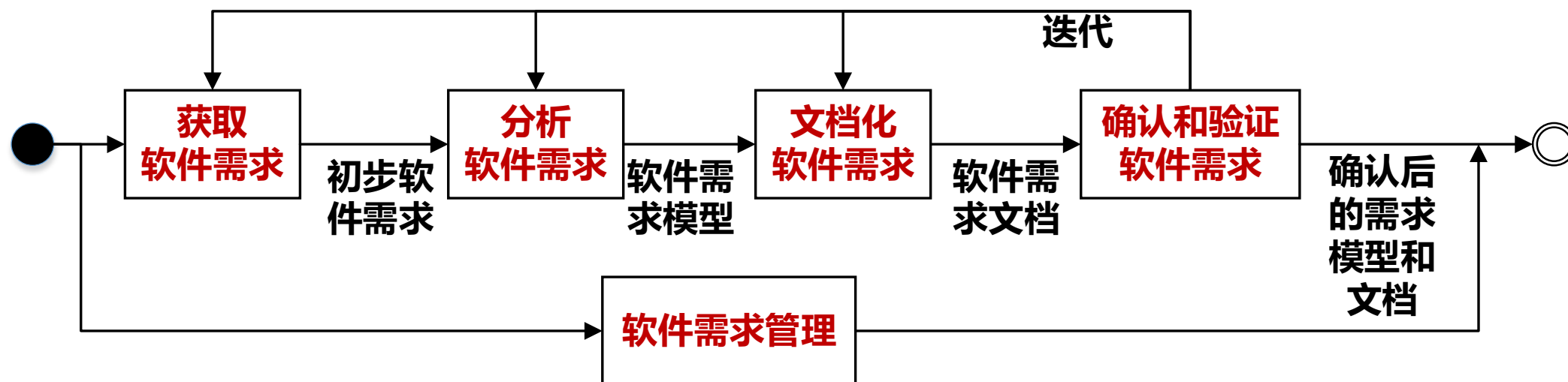
- ✓提不出有价值的软件需求
- ✓不清晰的软件需求：没有说清楚
- ✓不完整的软件需求：漏掉了重要的软件需求
- ✓不一致的软件需求：对同一个需求项有不同的表述



1.4 何为需求工程？

□用**工程**的理念和方法来指导软件需求实践，它提供了一系列的**过程、策略、方法学和工具**，帮助需求工程师加强对业务或领域问题及其环境的**理解**，**获取和分析**软件需求，指导软件需求的**文档化和评审**，以尽可能获得**准确、一致和完整**的软件需求，产生软件需求的相关软件制品

需求工程的一般性过程



需求工程的特点

□知识密集型工作，需要交叉多学科的知识

- ✓ “12306” 软件系统就需要充分掌握铁路旅行服务的相关知识，包括火车调度、车次安排、票务服务等等

□多方共同参与

- ✓ 软件需求的获得需要多方人员的共同参与，包括不同类别的用户、客户、领域和业务专家、各类开发者、质量保证人员等等

□需求获取的多种形式和源头

- ✓ 获取、构思、创作等，要采用多种形式和手段

□持续迭代和逐步推进

- ✓ 贯穿于软件整个生命周期，如 “12306” APP软件

需求工程的方法学（1/3）- 抽象

□如何理解和抽象软件需求？

- ✓软件需求本质是什么？应采用什么样的抽象来刻画软件需求？

□20世纪70年代结构化数据和数据流抽象

- ✓功能表现为数据处理，要理解软件功能需求，就要知道软件具有哪些数据以及要对这些数据进行什么样处理
- ✓提出了一组抽象来表征软件的功能性需求，包括数据流、数据字典、数据加工或处理、数据源等，设计了数据流图建模语言

□20世纪90年代面向对象抽象

- ✓软件功能表现为系统中的对象所展示的行为，要理解软件功能需求，就要知道软件有哪些对象、有什么样行为、交互和协作
- ✓设计面向对象建模语言UML，产生了面向对象需求分析方法学

需求工程的方法学（2/3）- 建模

□如何刻画和建立软件需求模型？

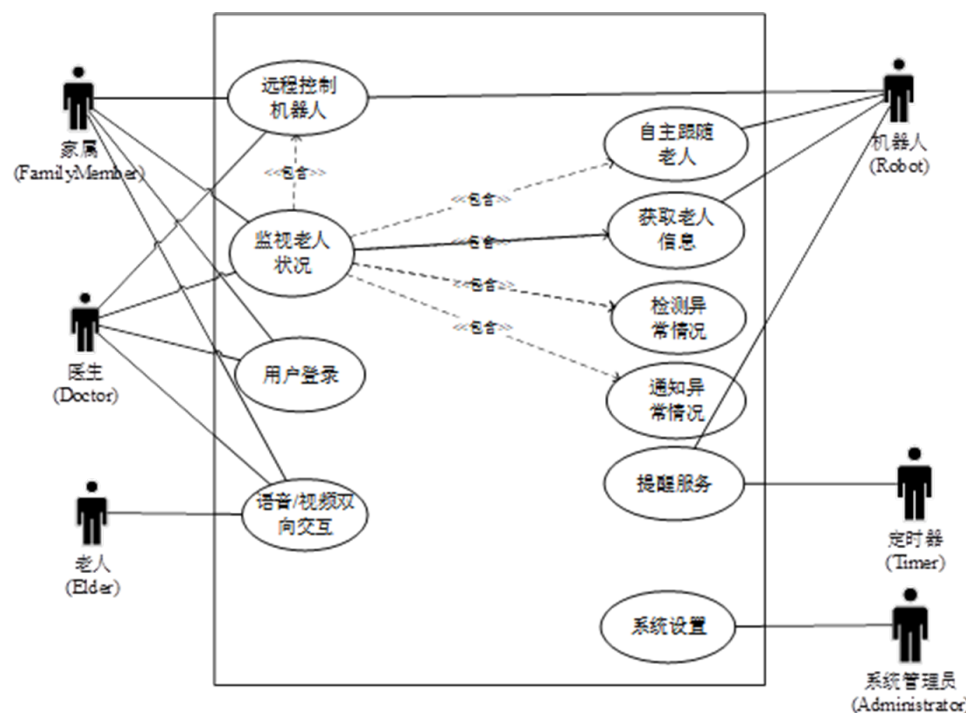
- ✓清晰地表达软件需求，目的是理解和交流

□采用自然语言或结构化自然语言

- ✓存在描述不直观、二义性和模糊性等问题

□图形化的需求建模语言

- ✓直观、易于理解
- ✓数据流图、UML图



需求工程的方法学（3/3）- 分析

□如何精化和分析软件需求？

- ✓循序渐进地获得软件需求细节，进而逐步得到详细的软件需求

□提供策略和手段

- ✓指导一步步地精化和分析软件需求
- ✓建立准确和一致的软件需求模型
- ✓防止漏掉重要的软件需求
- ✓发现并解决其中的问题和存在的缺陷，以保证软件需求的质量

软件需求工程师

□负责需求工程的各项工作

- ✓与用户和客户的沟通、导出和构思软件需求、协商需求问题或解决冲突、建立软件需求模型、撰写软件需求文档等

□须具备多方面知识、技能和素质，应既是**专才**，也是**通才**

- ✓软件工程、需求工程、业务领域的知识，如“12306”软件

□组织、沟通和协调

- ✓与软件的客户、用户，开展讨论、交流和评审

□语言表达

- ✓清晰地表达需求，准确地刻画内涵，直观地建立模型

□创新能力

内容

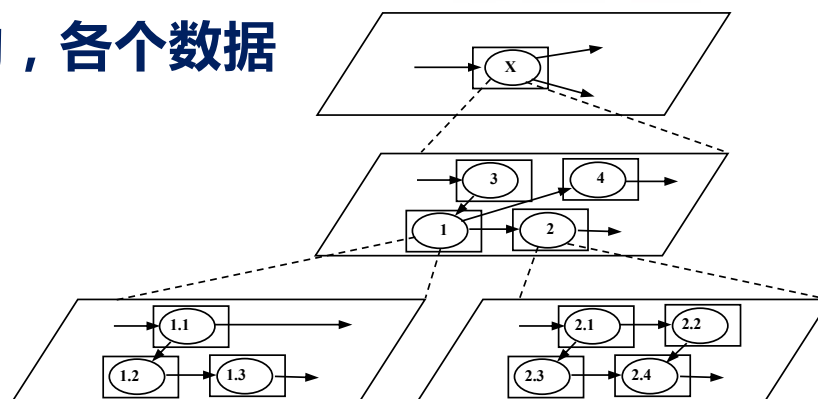
1. 软件需求和需求工程
- 2. 结构化需求分析方法学**
- 3. 面向对象需求分析方法学**
- 4. 需求工程的输出和评审**



2.1 基本思想

□ 数据主义认为，整个宇宙都是由**数据流构成的**，各个数据处理系统构成人类的社会。

□ **抽象与自顶向下的逐层分解**
(控制复杂性的两个基本手段)



- ✓ **抽象**：在考虑某一复杂问题的解决方案时，可以给出许多抽象级。从作为整体的软件系统开始，向下精化……，每一抽象层次上忽略其内部复杂性，只关注于其与外界的联系（输入输出）。
- ✓ **分解**：将较复杂问题不断分解为较小的问题，随着分解层次的增加，抽象的级别越来越低，越到底层越接近问题的解（算法和数据结构）

2.2 数据流图

□ Data Flow Diagram(简称DFD)：描述输入数据流到输出数据流的变换(即加工)过程，用于对系统的功能建模

□ 基本元素包括：

→ 数据流(data flow)：由一组固定成分的数据组成，代表数据的流动方向

○ 加工(process)：描述了输入数据流到输出数据流的变换，即将输入数据流加工成输出数据流

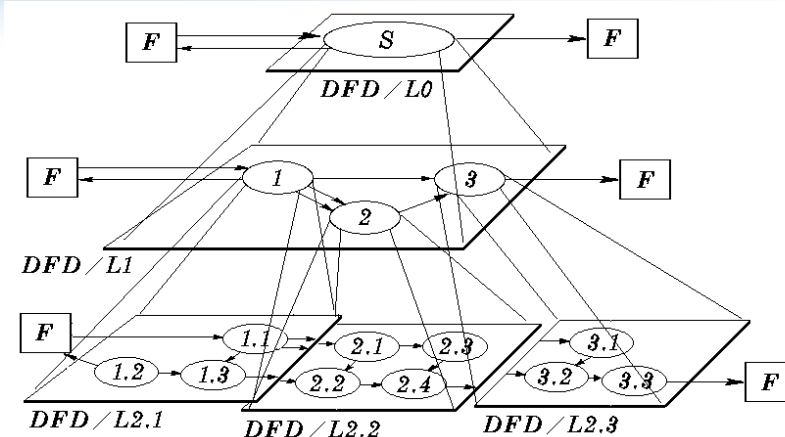
≡ 文件(file)：使用文件、数据库等保存某些数据结果供以后使用

□ 源或宿(source or sink)：数据来源去向
存在于软件系统之外的人员或组织，表示软件系统输入数据的来源和输出数据的去向，因此也称为源点和终点

2.1 结构化需求分析方法

画分层数据流图的步骤

1. 画系统的输入和输出
2. 画系统内部
3. 画加工内部
4. 根据自顶向下逐层分解的思想，重复第3步，继续分解加工，直至每个尚未分解的加工都足够简单(即不必再分解)



注：每个层次画在独立的数据流图中，加工个数可大致控制在“7加减2”的范围中

分层数据流图示例——空巢老人看护系统

□ 确定源或宿

- ✓ 老人、家属、医生、机器人和时钟
- ✓ 除了时钟外，都既是源又是宿

□ 顶层图唯一的加工

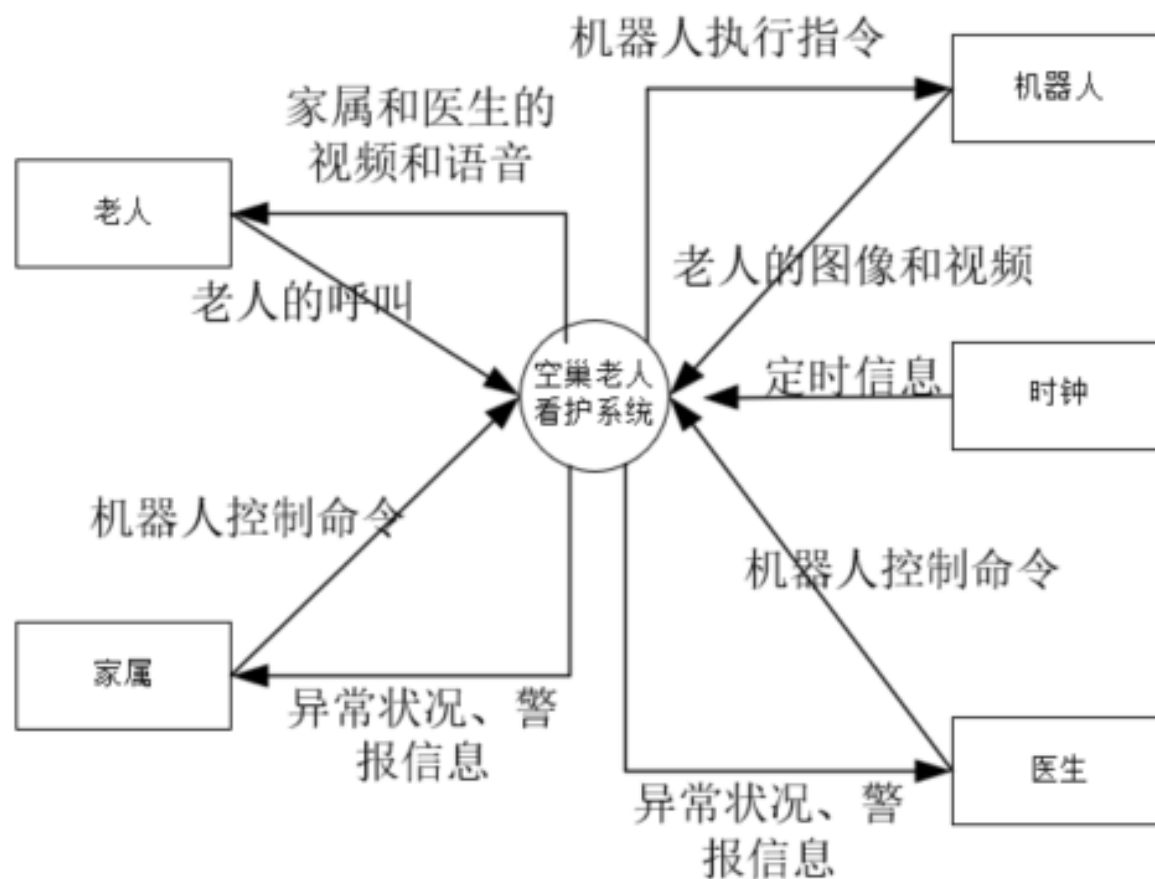
- ✓ 软件系统(空巢老人看护系统)

□ 确定数据流：系统的输入/输出信息

- ✓ 输入数据流：老人的呼叫、家属和医生的控制命令、老人的图像和视频、定时器
- ✓ 输出数据流：家属和医生的语音和视频、异常状态和警告信息、机器人执行指令

□ 顶层图通常没有文件

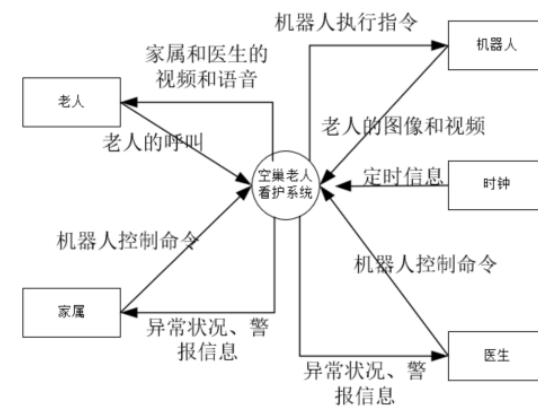
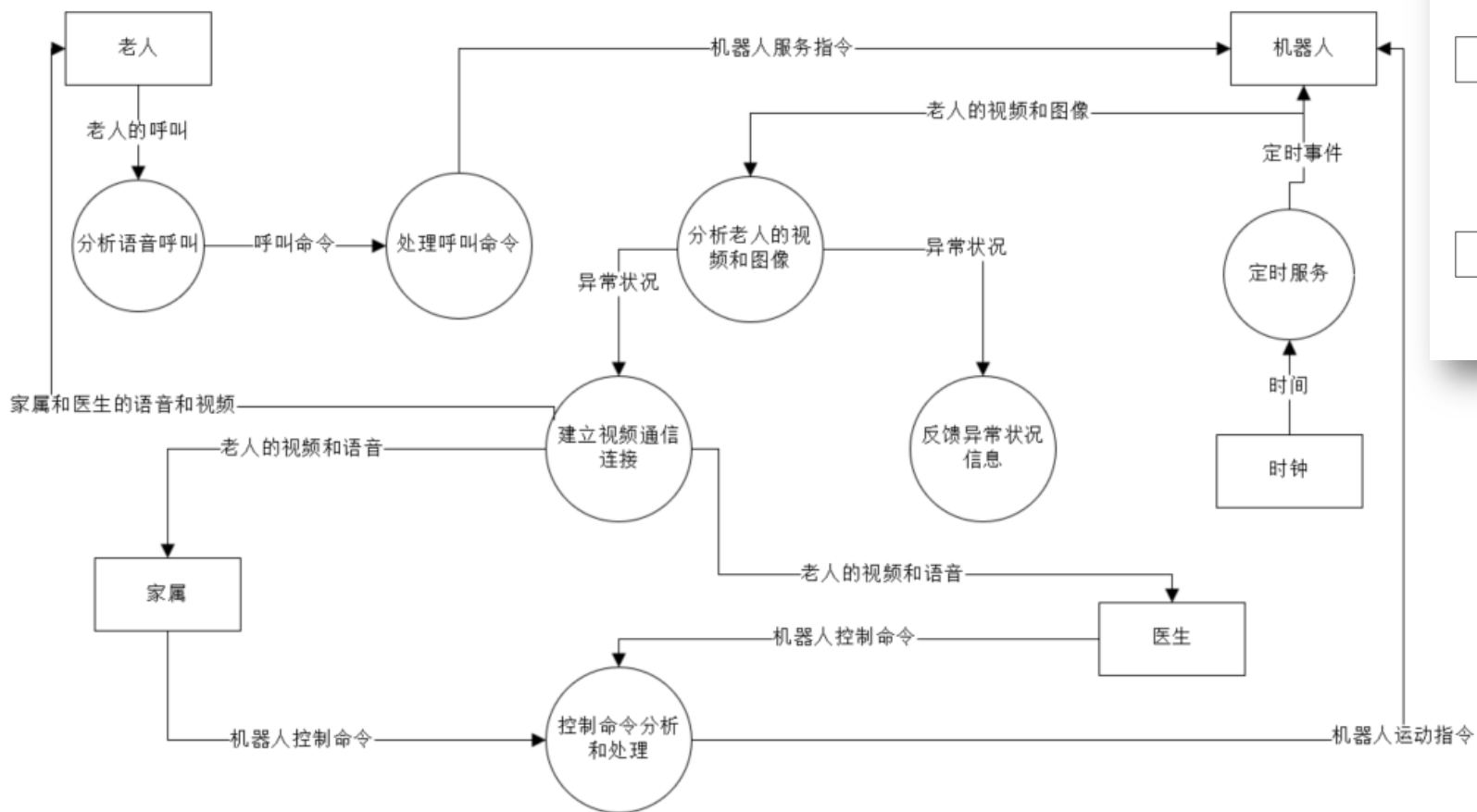
系统的输入输出（顶层图）



精化DFD

- 如果转换所提供的功能粒度大，内部所包含的数据和处理等要素多，相互之间关系松散，那么需要对该转换进行精化。
- 策略：自顶向下，逐步求精，确定系统的需求
- 注意点：
 - ✓ 父图和子图的信息平衡
 - ✓ 如果精化得到的DFD中的转换功能单一、粒度小，就无须继续对其做进一步的精化。
 - ✓ 检查是否存在冲突、遗漏

精化DFD



内容

1. 软件需求和需求工程
2. 结构化需求分析方法学
- 3. 面向对象需求分析方法学**
- 4. 需求工程的输出和评审**



2.1 基本思想（1/2）

□现实世界（应用问题）还是计算机世界（软件系统），它们都是由多样化的**对象所构成的**，每个对象都有其**状态并可提供功能和服务**，不同对象之间通过交互来开展**协作**来实现功能和提供服务

□示例：“空巢老人看护软件”

✓机器人就是一个对象，它可处于不同的状态（如空闲、运行、故障等），并可提供诸如获取视频和图像信息、播放语音、向前运动、先后运动等一系列的功能

2.1 基本思想 (2/2)

□面向对象软件工程提供对象、类、属性、操作、消息、继承等**概念**来抽象表示现实世界的应用，分析其**软件需求特征**，建立起**软件需求模型**，描述**软件需求**

- ✓基于类、包、关联等概念来分析应用系统的**构成**
- ✓借助类的方法等概念来描述对象所具有的**行为**
- ✓利用对象间的**消息传递**等概念来分析多个不同对象如何通过**协作**来实现应用功能的。

□面向对象需求分析方法学还提供了**可视化的建模语言**，帮助需求工程师建立多视点的软件需求模型

- ✓如用例模型、交互模型、分析类模型等等

2.2 面向对象的核心概念

- **对象 (Object)**
- **类 (Class)**
- **继承 (Inheritance)**
- **多态 (Polymorphism)**
- **覆盖 (Override)**
- **重载(Overload)**
- **消息 (Message)**
- **聚合 (Aggregation)和组合(Composition)**

对象(Object)

□个体或者事物抽象表示

- ✓现实世界和计算机世界
- ✓应用领域对象、软件实体

对象是具体、有意义的、
存在的实体

□对象的属性和操作

- ✓属性(Attribute)：对象的性质，其值定义了对象状态
- ✓操作(Operation)：也称方法，对象行为，表示对象提供的服务

□示例

- ✓用对象表示应用领域的一个事物（如NAO机器人），也可以用
它来表示在计算机软件中的某个运行元素或单元（如运行实例）

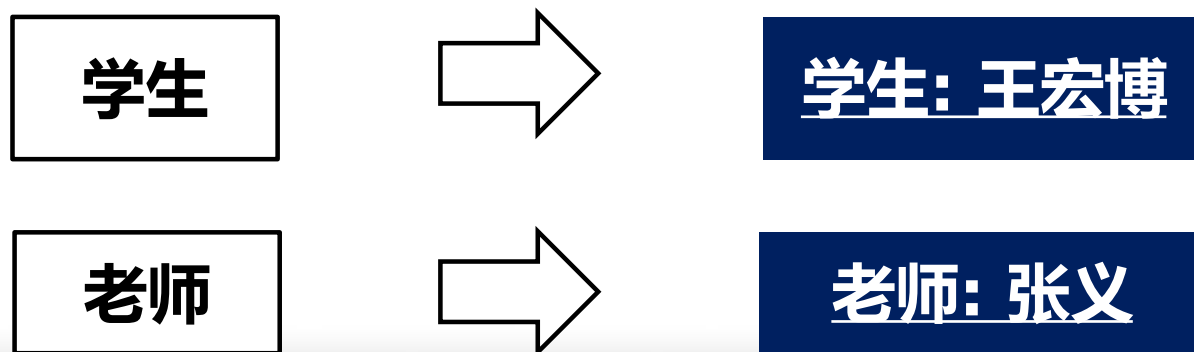
类(Class)

□类是对一组具有相同特征对象的抽象

- ✓分类、组织机制，将具有相同特征的对象组织为一类
- ✓封装了属性和操作

□对象与类的关系

- ✓对象是类的**实例**，类是创建对象的**模板**
- ✓类是**静态的抽象**；对象是**动态、可运行的实体**



消息(Message)

□消息传递是实现对象间通讯和协作的基本手段

- ✓一个对象向另一个对象发送消息来请求其服务

□消息描述

- ✓接收对象名、操作名和参数: `received-obj.msg-name(para.)`

□消息类型

- ✓**同步消息**：请求者需要等待响应者的处理结果
- ✓**异步消息**：请求者发出消息后继续工作，无需等待

学生:王宏博

`registerCourse(course-name)`

课程注册中心

继承(Inheritance)

□表示类与类间的一般与特殊关系

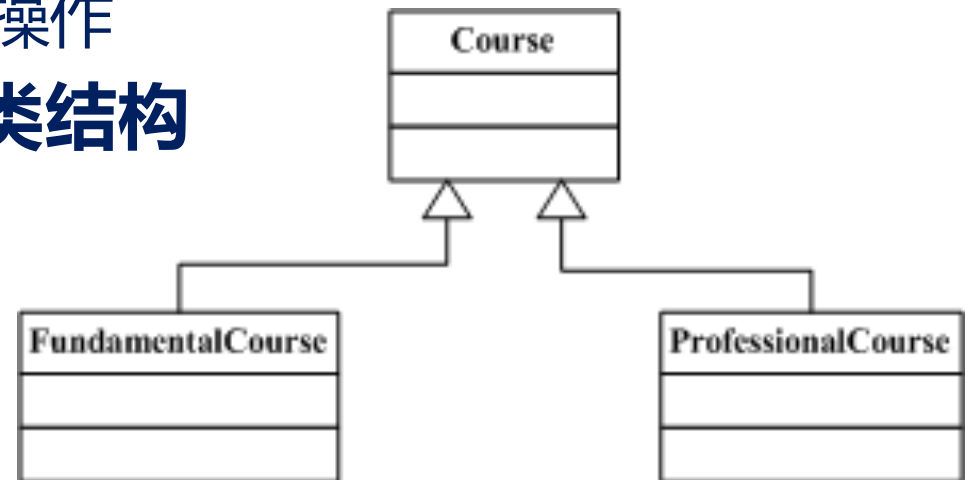
- ✓模拟现实世界类之间的遗传关系

□子（特殊）类可共享父（一般）类的属性和操作

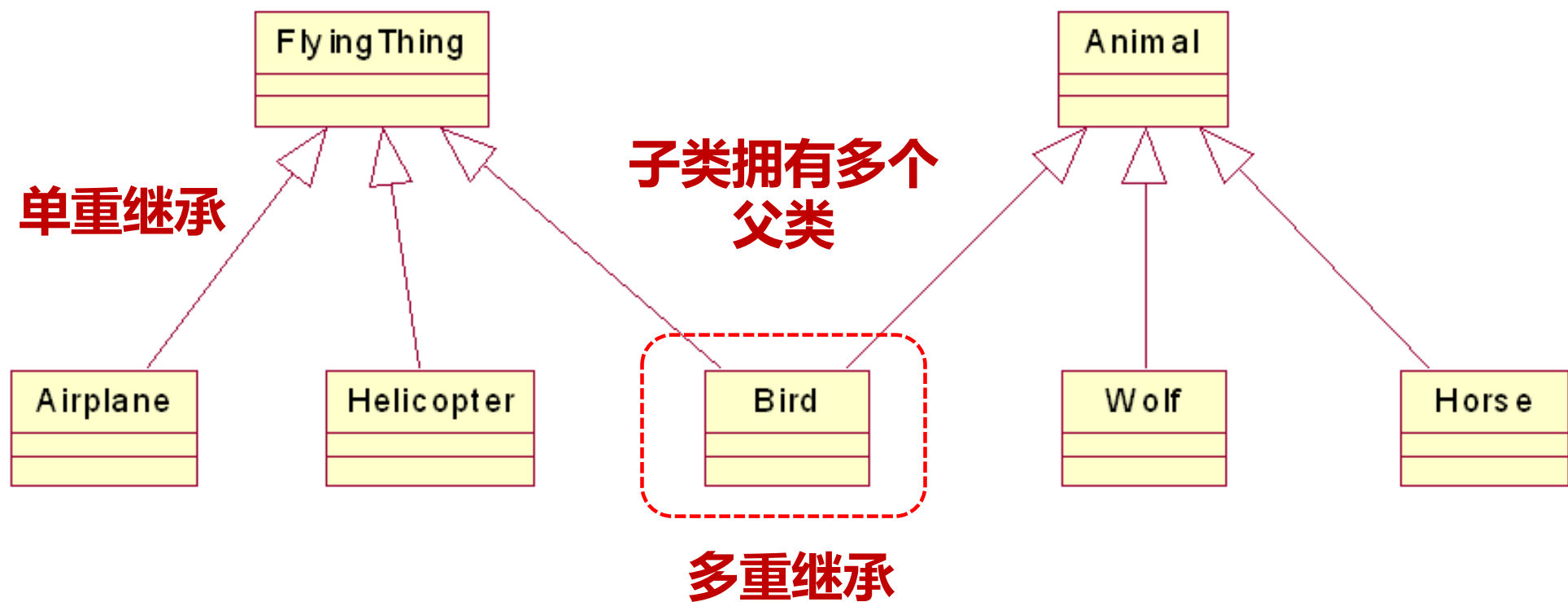
- ✓刻画类间的内在联系以及对属性和操作的共享
- ✓子类也可以有自己的独特属性和操作

□借助继承可形成系统的层次化类结构

- ✓示例：课程、公共课、专业课



单重继承和多重继承



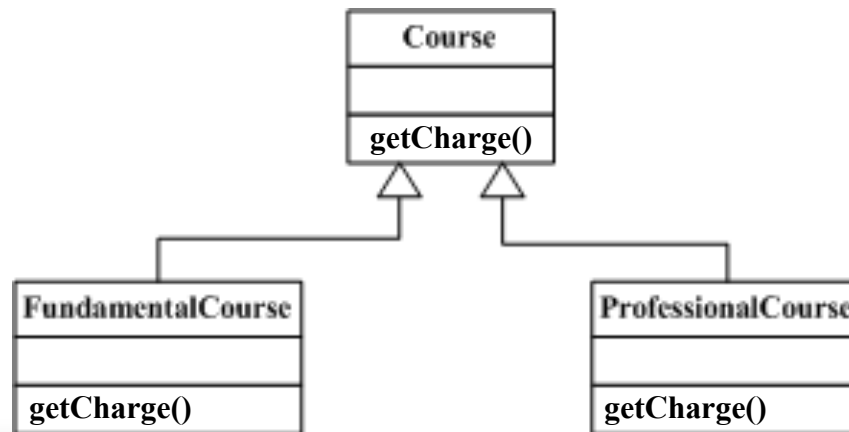
多态(Polymorphism)

□操作的**外部接口定义形式相同**，但是**内部实现方式不一样**

- ✓同一个操作作用于不同对象上可有不同解释，并产生不同结果
- ✓课程、公共课、专业课三个类的“计费”操作实现不同

□多态的执行方式

- ✓从具体子类开始，沿继承结构向上找，直至发现为止



覆盖(Override)

□ 子类**增加或重新定义**所继承的属性或方法，从而用新定义属性和方法来**覆盖**所继承的、来自父类中的属性或方法

```
public class A {  
    String name;  
    public String getValue() {  
        return "Value is:" + name;  
    }  
}  
  
public class B extends A {  
    String address;  
    public String getValue() {  
        return "Value is:" + address;  
    }  
}
```

子类B重新定义所继承的方法**getValue**

重载(Overload)

- 一个类中有多个同名操作，但它们在操作数或操作数类型上有区别，系统根据实参引用不同方法

```
Public class A {  
    int age;  
    String name;  
  
    public void setValue(int agePara){  
        age = agePara;  
    }  
  
    public void setValue(String namePara){  
        name = namePara;  
    }  
}
```

聚合(Aggregation)和组合(Composition)

□共性

- ✓均描述**整体-部分关系**，部分类对象是整体类对象的组成部分

□差别

- ✓**聚合**：部分类对象可以是多个整体类对象的组成部分，即部分类对象可以为多个整体类对象**所共享**
- ✓**组合**：部分类对象只能位于某个整体类对象之中，一旦整体类对象消亡，其中部分类对象也无法生存。从设计和实现的角度上看，整体类必须具备完整的管理部分类生命周期的职责。

□示例

- ✓**聚合关系**：老师与大学，老师可以兼职
- ✓**组合关系**：校长与大学，校长不可兼职

思考和讨论

□如何基于面向对象的概念来抽象和描述软件需求？

✓对象、类、消息传递等

□它们能够描述所有形式的软件需求吗？

✓质量需求、软件开发约束需求



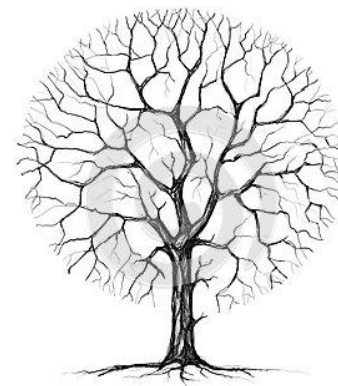
基于图的模型表示及优势

□表示方法

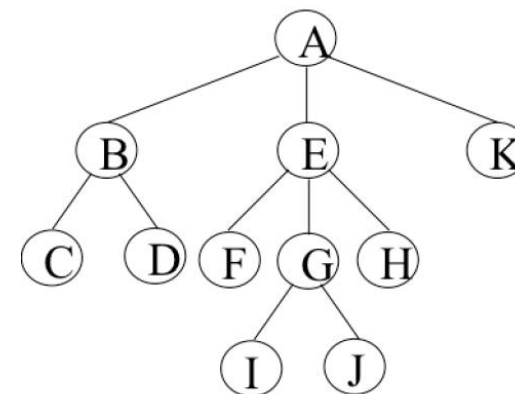
- ✓形式的符号化表示 $\langle N, E \rangle$
- ✓自然语言的表示

□图表示特点和优势

- ✓**直观**，一目了然
- ✓可以从不同方面来描述，体现**多视点**
- ✓抽象，**便于发现问题**



dreamstime.com



2.3 面向对象建模语言

□概念

- ✓基于**面向对象的概念和抽象**，提供图形化的**图符**，用来**表示**软件系统的一种语言

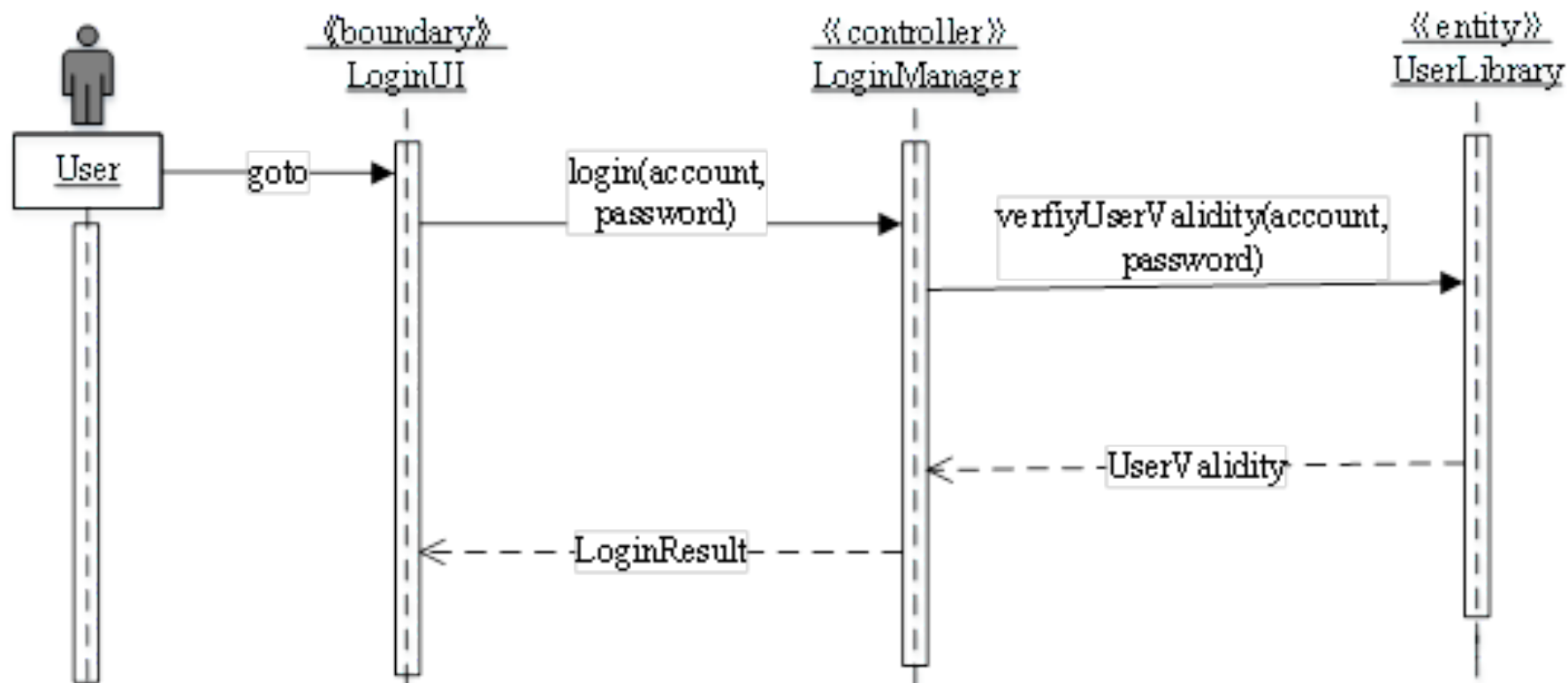
□目的

- ✓**用于建模**：绘制和描述软件系统模型(分析模型和设计模型)
- ✓**支持交流**：便于开发人员之间的交流、沟通和讨论

□组成

- ✓**语法**：图形化的符号表示
- ✓**语义**：形式或半形式的语义
- ✓**语用**：如何使用语言来建立模型、提供策略和原则

示例：面向对象建模语言的使用



顺序图：表示对象之间的消息传递和相互协作

面向对象建模语言的发展历程

□1980s末-1990s初出现大量面向对象建模语言

- ✓数量几十种之多
- ✓代表性：Booch方法、OMT方法和OOSE方法
- ✓各有千秋，却又有很多类似之处，往往让使用者无所适从

□UML的初衷

- ✓需要吸收不同建模语言的优点
- ✓寻求一种概念清晰、表达能力丰富、适用范围广泛的面向对象的建模语言

UML: Unified Modeling Language

□ Unified(**统一**)

- ✓ 提取不同方法中最好建模技术，如OMT(James Rumbaugh)，Booch method(Grady Booch)和OOSE(Ivar Jacobson)
- ✓ 采用**统一、标准化**的表示方式

□ Modeling(**建模**)

- ✓ 对现实系统和软件系统进行**可视化建模**
- ✓ 建立系统模型

□ Language(**语言**)

- ✓ **图形化语言**：语法、语义和语用
- ✓ 包括规则，约束 扩展机制



www.omg.org

OMG组织致力于对象技术标准化工作

UML用途

- 用来**可视化**(visualize)、**描述**(specify)、**构造**(construct)和**文档化**(document)软件密集型系统的各种产品
- 支持不同人员之间的**交流**(Communication)

UML®

Unified Modeling Language

A specification defining a graphical language for visualizing, specifying, constructing, and documenting the artifacts of distributed object systems.

Title: Unified Modeling Language
Acronym: UML®
Version: 2.5.1
Document Status: formal ①
Publication Date: December 2017
Categories: **Modeling** **Software Engineering**
IPR Mode ① RF-Limited ①



Visualizing



Documenting



Business Modeling



Specifying



Constructing



Communications

多视点建模 (1/2)

□ 结构视点 (Structural View)

✓ 用于描述系统的构成

- ✓ UML提供了**包图** (Package Diagram)、**类图** (Class Diagram)、**对象图** (Object Diagram) 和**构件图** (Component Diagram) , 从不同的抽象层次来表示系统的静态组织及结构

□ 行为视点 (Behavioral View)

✓ 刻画系统的行为

- ✓ UML提供了**交互图** (Interaction Diagram)、**状态图** (Statechart Diagram) 与**活动图** (Activity Diagram) , 以从不同侧面刻画系统的动态行为。

多视点建模 (2/2)

□部署视点 (Deployment View)

- ✓刻画目标软件系统的软件制品及其运行环境
- ✓UML提供了部署图 (Deployment Diagram) 来描述软件系统的部署模型

□用例视点 (Use Case View)

- ✓刻画系统的功能
- ✓UML提供了用例图 (Use Case Diagram) 以描述系统的用例及其与外部执行者之间的关系。

UML的视点及图

视点	图 (diagram)	说明
结构	包图 (package diagram)	从包层面描述系统的静态结构
	类图 (class diagram)	从类层面描述系统的静态结构
	对象图 (object diagram)	从对象层面描述系统的静态结构
	构件图(component diagram)	描述系统中构件及其依赖关系
行为	状态图(statechart diagram)	描述状态的变迁
	活动图(activity diagram)	描述系统活动的实施
	通信图(communication diagram)	描述对象间的消息传递与协作
	顺序图(sequence diagram)	描述对象间的消息传递与协作
部署	部署图 (deployment diagram)	描述系统中工件在物理运行环境中的部署情况
用例	用例图 (use case diagram)	从外部用户角度描述系统功能

2.4 面向对象需求分析步骤

□明确问题边界，获取软件需求，建立用例模型

- ✓理解系统边界，识别系统利益相关方，导出或构思软件需求，绘制出软件用例图，建立软件用例模型

□开展用例分析，精化软件需求，建立分析模型

- ✓分析用例，从而精化软件需求，建立起用例的交互模型，并依此导出系统的分析类图

□汇总需求模型，撰写需求文档，评审软件需求

- ✓汇总不同视点、不同抽象层次的需求模型，撰写软件需求文档，对软件需求模型和文档进行评审，以确保它们的质量

面向对象需求分析方法的优势和特色

□自然建模

- ✓面向对象提供了一系列更加贴近现实世界（而非计算机世界）的概念和抽象来描述软件需求

□统一的概念和抽象

- ✓提供统一的概念和抽象，方便用户和软件开发人员用同一个概念模型来理解问题、分析问题和解决问题
- ✓无需采用模型转换的方式，而是采用不断精化模型的方法来开展软件开发，从而简化软件开发的复杂度

需求工程的CASE工具

- **需求文档撰写工具**，如借助于Microsoft Office、WPS
- **需求建模工具**，如利用Microsoft Visio、Rational Rose、StarUML
- **软件原型开发工具**，如Mockplus、Axure RP Pro、UIDesigner
- **需求分析和管理的专用工具**，如IBM Rational RequisitePro
- **配置管理工具和平台**，如Git、Github、Gitlab、PVCS、Microsoft SourceSafe等

内容

1. 软件需求和需求工程
2. 结构化需求分析方法学
3. 面向对象需求分析方法学
- 4. 需求工程的输出和评审**



3.1 需求工程的输出软件需求制品

□ 软件需求模型

- ✓ 抽象和直观地表示软件需求

□ 软件需求文档

- ✓ 完整和详尽地记录软件需求

□ 软件原型

- ✓ 直观地展示软件需求

软件需求文档的内容

- ① 系统和文档概述
- ② 软件功能性需求
- ③ 软件质量方面的需求
- ④ 软件开发的约束性需求
- ⑤ 软件需求的优先级

3.2 软件需求缺陷

- **需求缺失**，即漏掉了一些重要的软件需求
- **需求描述不正确**，对软件需求的理解存在偏差
- **需求描述不准确**，软件需求的表述与用户的要求不一致
- **软件需求有冲突、不一致**
- **软件需求不可行**，存在可行性问题
- **软件需求不详尽**，没有提供足够详细的信息

软件需求确认和验证

□软件需求确认

- ✓ **站在用户和客户的角度，确保软件需求的正确性**，通常采用需求评审（Review）、原型确认等方式。
- ✓ 例如，12306 App软件的开发者可邀请一些旅客，作为软件的用户代表，评审12306 App软件的需求文档以及所开发的软件原型，逐条确认各项软件需求的合法性和正确性
- ✓ 基于原型的确认是一种常用且有效的方式

□软件需求验证

- ✓ **判断软件需求文档和模型是否准确地刻画了用户和客户的要求**，后续的软件设计制品、程序代码等是否正确地实现了软件需求

软件需求变更管理

□软件需求的变更管理

- ✓ **多变性和易变性**引起的，明确哪些方面的需求发生了变化、反应在软件需求模型和文档的哪些部分、导致软件需求模型和文档的版本发生了什么样变化等

□软件需求的追溯管理

- ✓ 开展**溯源追踪**，掌握清楚是谁提出需求变更、为什么要进行变更等内容，以判别需求变更的合法性；评估需求变更的影响域，基于对需求变更的理解，分析需求变更会对哪些软件制品会产生什么样的影响；评估需求变更对软件项目开发带来的影响

□软件需求的配置管理

- ✓ 形成软件需求基线

小结

□ 软件需求

- ✓ 来自于**软件利益相关者**，表现为**多种形式**，具有**多变易变**特点

□ 需求工程

- ✓ 基于工程的手段来支持需求的**获取、分析、建模和文档化**

□ 结构化需求分析方法学

□ 面向对象需求分析方法学

- ✓ 基本思想：系统中的**对象及其展现的功能、行为和协作**
- ✓ 基本概念：对象、类、消息传递等
- ✓ 建模语言：UML、视角、图

□ 需求输出，**模型、文档和原型**