第一章 软件工程概述

1.软件的定义：

软件 = 程序 + 数据 + 文档

注释：程序和数据是构造软件的基础，文档是软件质量的保证，也是保证软件更新及生命周期长短的必需品。

**2.软件危机：**

**定义：**

由于**落后的软件生产方式**无法满足**迅速增长的计算机软件需求**，从而导致**软件开发与维护过程中**出现一系列**严重问题的现象**。

**具体表现在：**

• 软件开发成本难以估算，无法制定合理的开发计划；

• 用户的需求无法确切表达；

• 软件质量存在问题；

• 软件的可维护性差；

• 缺乏文档资料；

**产生软件危机的原因：**

• 软件系统本身的复杂性；

• 软件开发的方法和技术不合理及不成熟；

**软件危机解决途径：**

**软件工程学**，即采用**工程化的方法**从事**软件系统的研究和维护**。

**3.软件工程**

**软件工程三要素：**方法、工具和过程。

• 方法：提供了“如何做”的技术

• 工具：提供了自动或半自动的软件支撑环境

• 过程：将软件工程的方法和工具综合起来以达到合理、及时地进行计算机软件开发的目的

**第二章 软件生命周期模型**

**1.软件生命周期  
定义：**指软件产品**从考虑其概念开始**，**到该软件产品不再使用为止**的整个时期，一般包括**概念阶段**、**分析与设计阶段**、**构造阶段**、**移交和运行阶段**等不同时期。

**六个基本步骤：**

• 制定计划 P

• 需求分析 D

• 软件设计 D

• 程序编码 D

• 软件测试 C

• 运行维护 A

**2.软件过程模型**从一个特定角度提出的**对软件过程的概括描述**，是对软件开发实际过程的抽象，包括**构成软件过程的各种活动**（Activities）、**软件工件**（artifacts）以及**参与角色**（Actors/Roles）等。

**3.软件生命周期模型**  
软件生命周期模型**是一个框架**，描述从**软件需求定义**直至**软件经使用后废弃为止**，

跨越整个生存期的软件开发、运行和维护**所实施的全部过程、活动和任务**，同时描述生命周期不同阶段产生的软件工件，明确活动的执行角色等。

**4.传统模型种类**  
•瀑布模型

•演化模型

•增量模型

•喷泉模型

•V模型和W模型

•螺旋模型

•构件组装模型

•快速应用开发模型

•原型方法

**瀑布模型：**

规定了**软件生命周期**提出的**六个基本工程活动**，并且规定了它们**自上而下、相互衔接**的固定次序，如同瀑布流水，逐级下落。**瀑布模型**将**软件生命周期**划分为**定义**阶段、**开发**阶段和**维护**阶段，在**定义**阶段部署了**计划**和**需求分析**活动；在**开发**阶段部署了**设计**，**编码**和**测试**活动，**维护**阶段部署了**运行**/**维护**行动。

**瀑布模型优点：**

**推迟了软件实现，强调在软件实现前必须进行分析和设计工作。**

但使用**瀑布模型**人们认识到，由于需求很难调研充分，所以**很难一次性**开发成功。

**演化模型**——**针对需求不是很明确的软件**：

**演化模型**提倡**两次开发**：

第一次是**试验开发**，得到**试验性的原型产品**，其目标只是在于**探索可行性**，**弄**

**清软件需求**；

第二次在**此基础上**获得较为满意的软件产品。

演化模型**适用范围**：

• 需求不清楚；

• 小型或中小型系统；

• 开发周期短

**5.新型软件生命周期模型**

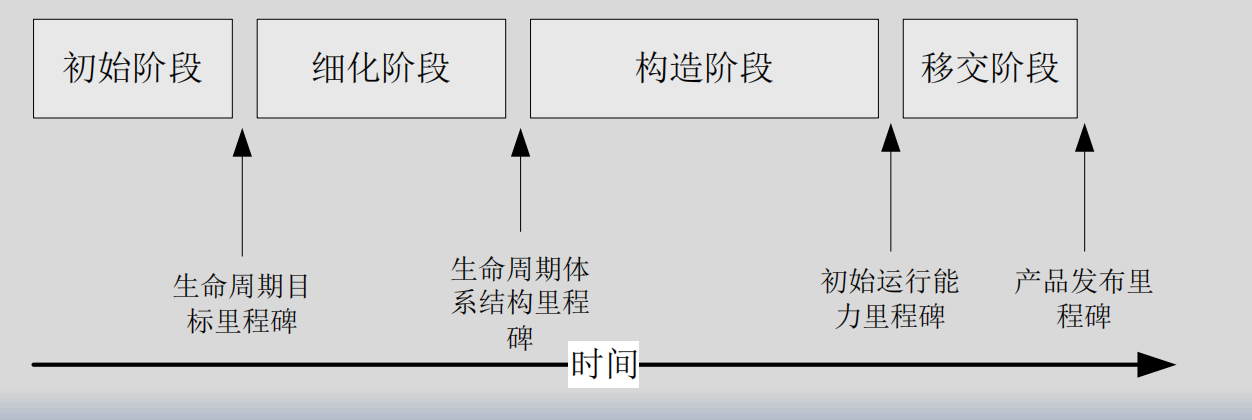
• RUP（Rational Unified Process）是由Rational公司（现被IBM公司收购）开发的一种软件工程过程框架，是一个基于面向对象的程序开发方法论 。

• RUP既是一种软件生命周期模型，又是一种支持面向对象软件开发的工具，它将软件开发过程要素和软件工件要素整合在统一的框架中 。

**RUP的四个主要阶段：**

RUP中的**软件生命周期**在时间上被分解为四个顺序的阶段：**初始阶段**（Inception）、**细化阶段**（Elaboration）、**构造阶段**（Construction）和**交付阶段**（Transition）。

每个**阶段结束**于一个**主要的里程碑**(Major Milestones)，并在**阶段结尾**执行一次评估以确定这个阶段的目标是否已经满足。如果评估结果令人满意的话，可以允许项目进入下一个阶段。



**6.敏捷及极限编程**• **敏捷建模**（Agile Modeling，AM）是由Scott W. Ambler从许多的软件开发过程实践中归纳总结出来的一些**敏捷建模价值观**、**原则和实践**等组成的，它是**快速软件开发**的一种思想代表，**具体的应用有极限编程**、SCRUM、水晶、净室开发等。

• 2001年敏捷联盟成立，其主要特点就是具有快速及灵活的响应变更的能力。

**第三章 软件需求分析**

**1.软件需求分析：**

**软件需求分析**的**对象**：**用户要求**。

**软件需求分析**的**任务**是：准确地**定义**新系统的**目标**，回答系统必须“做什么”的问题并**编制需求规格说明书**。

**需求分析**的**目标**：借助于当前（业务）系统的逻辑模型导出目标系统的逻辑模型，解决目标系统的“做什么”的问题。

**2.数据、功能及行为建模：**

数据模型：信息内容和关系；信息流；信息结构。

功能模型：对进入软件的信息和数据进行变换和处理的模块，它必须至少完成三个常见功能：输入、处理和输出。

行为模型：大多数软件对来自外界的事件做出反应，这种刺激／反应特征形成了行为模型的基础。行为模型创建了软件状态的表示，以及导致软件状态变化的事件的表示。

**3.需求类别**  
功能需求：列举出所开发软件在功能上应做什么，这是最主要的需求。

性能需求：给出所开发软件的技术性能指标，尤其是系统的实时性和其他时间要

求，如响应时间、处理时间、消息传送时间等；资源配置要求，精确度，数据处

理量等要求。

环境需求：是对软件系统运行时所处环境的要求。

• 在硬件方面，采用什么机型、有什么外部设备、数据通信接口等等。

• 在软件方面，采用什么支持系统运行的系统软件（指操作系统、数据库管理系统等）。

• 在使用方面，需要使用部门在制度上、操作人员的技术水平上应具备什么样的条件等等。

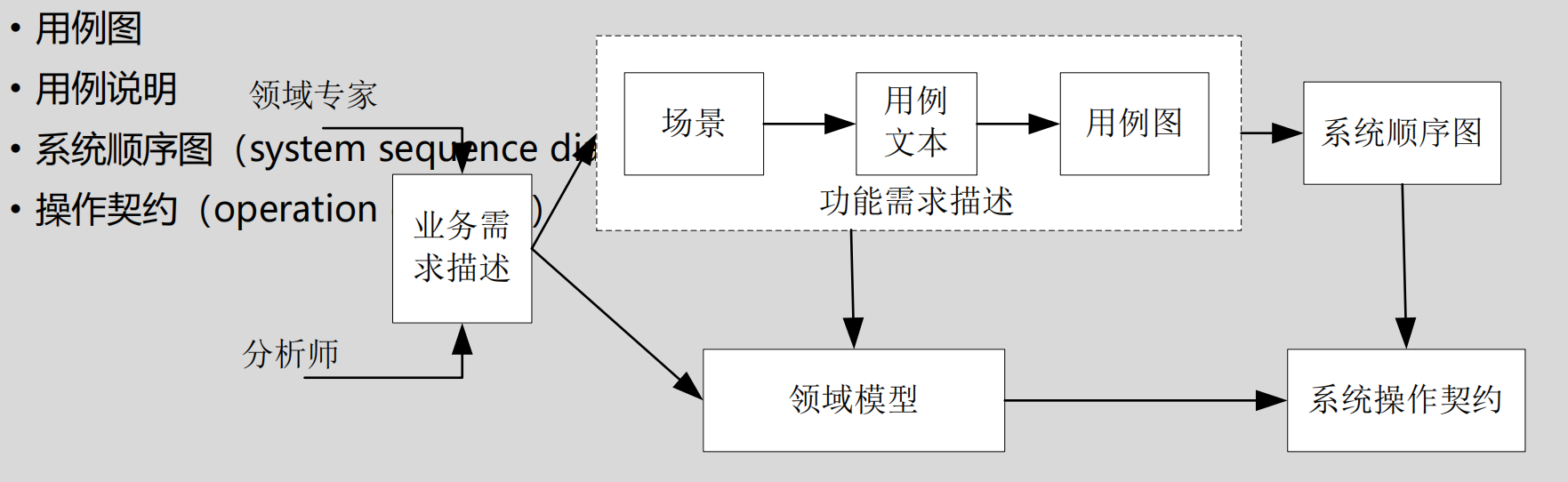
**第四章 面向对象需求分析方法**

**1.面向对象的需求分析建模**

**面向对象分析方法**中的**需求分析**包含两个模型：**领域模型**和**用例模型**。

**领域模型**表示了**需求分析**阶段“**当前系统**”**逻辑模型**的**静态结构**及**业务流程**。

**用例模型**是“**目标系统**”的**逻辑模型**，定义了“**目标系统**”做什么的需求。由以下四个部分组成：**用例图，用例说明，系统顺序图，操作契约**



2.领域模型  
定义：

针对某一特定领域内**概念类**或者**对象**的抽象可视化表示。

主要用于概括地描述**业务背景**及重要的**业务流程**，并通过UML的类图和活动图进行展示，帮助软件开发人员在短时间内了解业务。

• 业务背景：**可由需求定义或者用例说明中具有代表业务概念或者业务对象的词汇获得，这些词汇可统称为“概念类”；并通过能够代表关系的词汇建立概念类之间的关系，表示成能够代表业务知识结构的类图；**

• 业务流程：一般由角色及其执行的活动（活动及任务节点）构成，活动的输出一般有数据对象和传给另一个活动的消息组成，建议使用**UML的活动图**进行描述。

**创建领域模型的步骤：**

• 第1步，找出**当前需求**中的**候选概念类**；

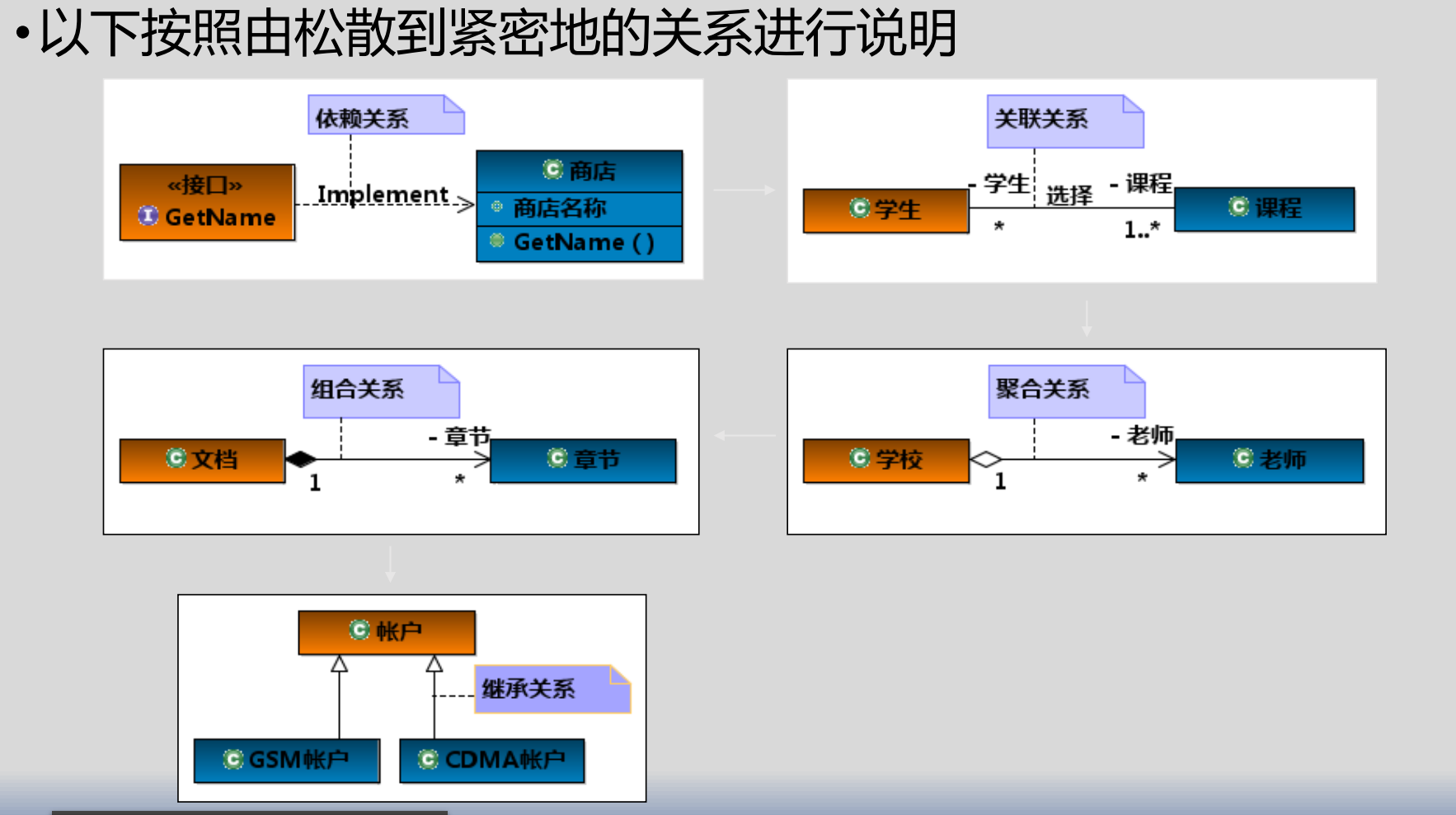
• 第2步，在**领域模型**中**描述这些概念类**。用**问题域**中的**词汇**对**概念类**进行**命名**，

将与当前需求无关的概念类排除在外。

• 第3步，在**概念类之间**添加必要的**关联**来记录那些需要保存记忆的关系，概念之间的关系用**关联、继承、组合/聚合**来表示。

• 第4步，在**概念类**中添加用来实现需求的**必要属性**。

**类的关系——依赖、关联、组合、聚合、继承：**



3.用例模型  
用例模型由以下四个部分组成：

• 用例图；

• 用例说明；

• 系统顺序图（system sequence diagram，option）；

• 操作契约（operation contract，option）；

**用例图：**

用例图由三个基本元素组成：

• Actor：称为角色或者参与者，表示使用系统的对象，代表角色的不一定是人，

也可以是组织、系统或设备；

• Use\_case：称为用例，描述角色如何使用系统功能实现需求目标的一组成功场

景和一系列失败场景的集合；

• Association：表示角色与用例之间的关系，以及用例和子用例之间的关系；

**基本用例与子用例**：

**基本用例**：**与角色直接相关的用例**，表示系统的功能需求；

**子用例**：通过场景描述分析归纳出的用例，也表示了系统的功能，但**这些用例与**

**角色无直接关系**，**而与基本用例存在关联关系**；

• **包含子用例**：多个基本用例中的某个与角色交互的场景具有相同的操作，且这些场景都是基本用例中必须执行的步骤，可以将其抽取出来作为基本用例的子用例；

• **扩展子用例**：（多个）基本用例中的某些场景存在相同的条件判断的情况，可以将其抽取出来作为基本用例的子用例；

**系统顺序图：**

系统顺序图中“一般”只需要三个UML的符号元素

• 角色；

• 代表软件系统的对象，一般使用system或者系统命名；

• 角色与system之间的交互信息，简称**消息**或**操作**；

**操作契约**

**系统操作**：处理系统事件的操作，也称为系统事件；

**操作契约**是为**系统操作**而定义的，参考领域模型中业务对象接收到相同的系统事件后，执行必须的业务处理时各业务对象的状态以及系统操作执行的结果，以便软件设计时进行参考。模板如下表所示：

|  |  |
| --- | --- |
| 操作： | 操作以及参数的名称 |
| 交叉引用： | （可选择）可能发生此操作的用例 |
| 前置条件： | 执行该操作之前系统或领域模型对象的状态 |
| 后置条件： | 操作完成后领域模型中对象的状态：  1、对象的创建和删除；  2、对象之间“关联”的建立或消除；  3、对象属性值的修改； |

**创建操作契约：**

• 根据系统顺序图识别进入到系统内的所有系统事件，即操作；

• 针对每一个系统操作结合对应的用例领域模型，找到与此操作相关的概念类对象；

• 对那些相对复杂以及用例描述中不清楚的那些系统操作按照以下内容描述并确定对象的状态变化，即后置条件；

• 对象实例创建和删除。

• 对象属性修改。

• 对象关联形成和断开。

**第五章 结构化需求分析方法**

**1.数据建模**