标题：软件工程

来源：李爱萍、崔东华、李东生.软件工程.人民邮电出版社，2014 （微信读书）

编辑日期：2020.0.08

适用对象：

先验知识：

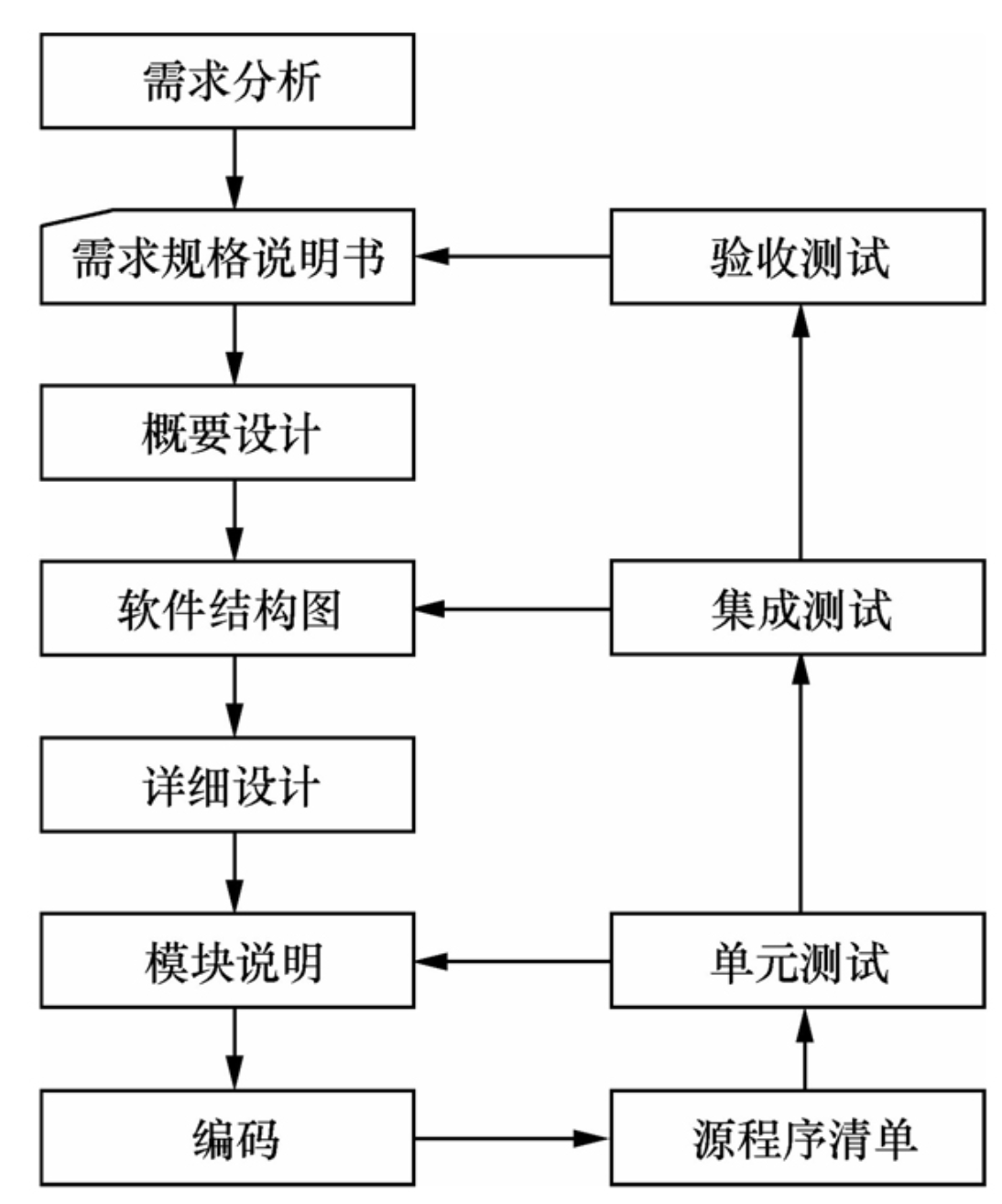
概念产生：1968年，NATO会议首次提出”软件工程“的概念。

目录：

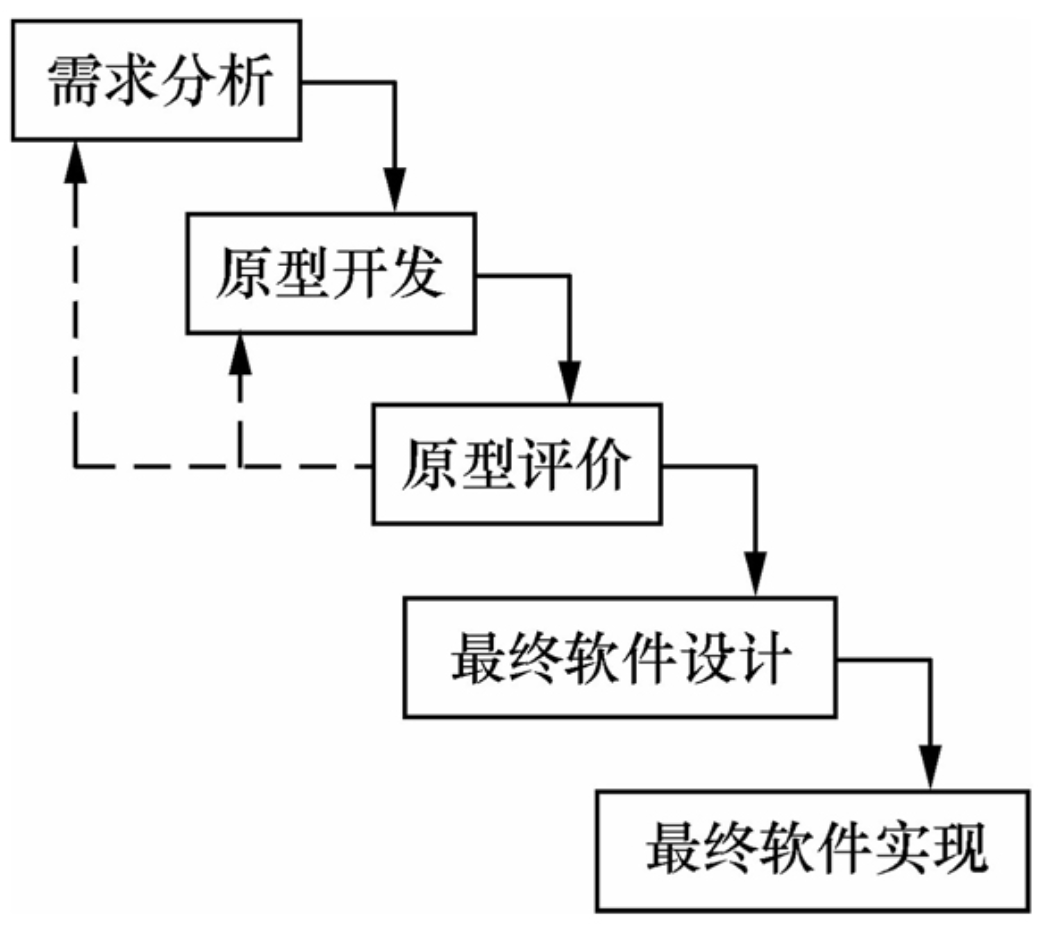
正文：

# 第一章 背景及意义

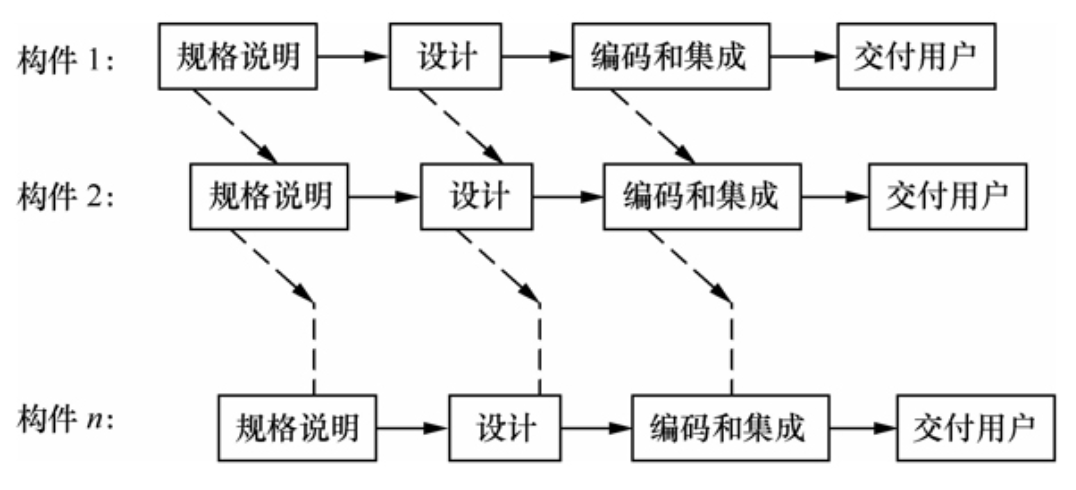
* 软件工程背景：软件危机
* 软件工程的原理：应用计算机科学、工程学、管理学及数学的原则、方法来创建软件
* 软件工程的功能：指导软件开发、质量控制、开发过程管理
* 软件的组成：程序、数据、开发、使用、维护所需要的所有文档的完整集合。IEEE中软件的软件的定义：计算机程序、方法、规则、相关文档以及在计算机上运行时所必必需的数据，其中方法和规则通常在文档中说明并在程序中实现
* 软件开发的文档规范：《计算机软件开发规范》、《计算机软件需求说明编制指南》、《计算机软件测试文件编制规范》、《计算机软件配置管理计划规范》
* 软件的特点：运行时为，为了适应软件、硬件、环境以及需求的变化，需要进行修改和完善，同时会引入新的错误使得软件退化。大部分产品为定制的，生产效率低
* 软件分类：
  + 从计算机角度分类：a)系统软件：管理、控制和维护计算机及外设，并提供计算机用户界面的软件，如操作系统、编译器、数据库管理系统、网络软件等；b)应用软件：解决某一领域问题的软件，如通信软件、计算机辅助教学软件等
  + 从软件用途分类：a)服务类软件：面向用户，提供各种服务的软件，包括软件开发工具和常用的库函数等；b)维护类软件：面向计算机维护，包括错误诊断、检测、测试、调试等软件；c)操作管理软件：面向计算机操作和管理，包括操作系统、网络通信系统、计算机管理软件等
* 软件危机的表现形式：开发成本和研制精度的误差；无法满足用户需求；软件质量保障技术，包括审查、复审和测试不够完善；可维护性差，缺少公认的规范和完善规范的文档；生产效率等
* 软件危机的解决途径：改变软件就是程序的认知错误；使用完善并不断改进的技术和方法；良好的组织和配合、严密的管理；使用好的软件工具
* 软件工程的目标：降低软件开发成本；满足用户需求的全部软件功能、性能、易用性、可重用性、可移植性、可靠性；降低维护成本
* 软件工程的研究内容：
  + 用分阶段的生存周期计划严格管理，包括项目概要计划、里程碑计划、项目控制计划、产品控制计划、验证计划、运行维护计划
  + 坚持进行阶段评审，评审失败时不能进入下一阶段，以便及时发现错误并改正，提高开发效率
  + 实行严格的产品控制，适应需求的变化
  + 采用现代程序设计技术，使用面向对象的技术等
  + 提高软件可见性，根据软件产品的总目标及完成期限，规定开发人员的责任和各阶段产品标准，使得每个阶段的所有结果能够被清晰审查
  + 开发小组人员小而精
* 软件生存周期：
  + 计划时期
    - 问题定义：调查用户需求，提出系统目标与范文的说明，并通过用户审查和认可
    - 可行性研究：清楚问题的性质、目标、规模、解决方法、技术可行性、经济效益等。可做一次简化的、抽象的需求分析和概要设计
  + 开发时期
    - 需求分析：功能需求、性能需求、环境约束、外部接口等
    - 软件设计：概要设计：将需求转化为体系结构，从需求规格说明书导出为软件结构图，确定模块之间的关系，并设计数据库结构；详细设计：对每个模块进行精确的、结构化的过程描述，确定功能需要的算法和数据结构，并用相应的详细设计工具表示出来
    - 编码：将模块的过程性描述翻译为计算机可接受的源程序，并保证结构好，清晰易读
  + 运行时期
    - 测试：保证软件质量的重要手段。可细分为单元即模块测试、集成即组装测试、验收即确认测试。
    - 维护：发现潜在错误并诊断和改正
* 软件开发模型：包含计划、开发和维护三类活动
  + 瀑布模型：即生存周期模型或线性顺序模型，以文档驱动。具有阶段顺序性和依赖性
    - 只有分析员能够做出准确的需求分析时，才能得到预期的正确结果
    - 缺乏灵活性，无法解决需求你不明确或不准确的问题



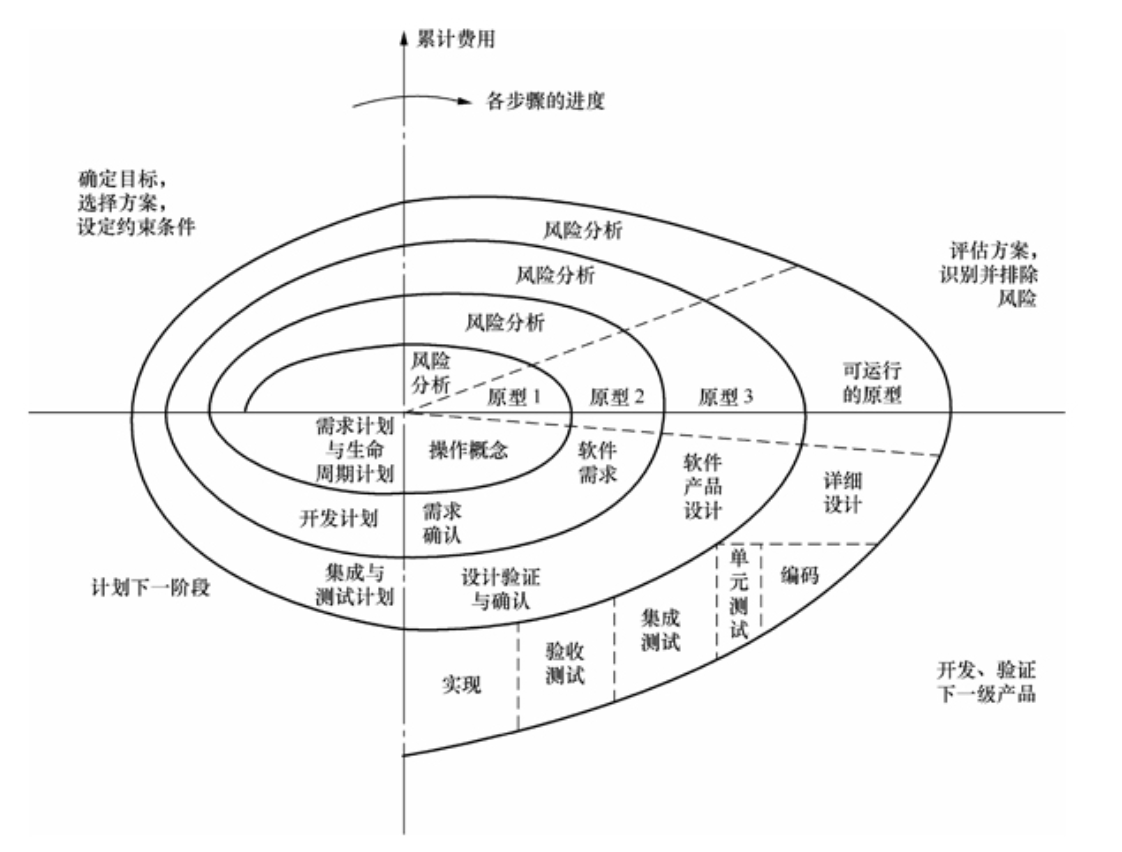
* + 快速原型模型：先建立能够反映用户需求的原型系统，让用户试用，了解未来目标系统的概貌。



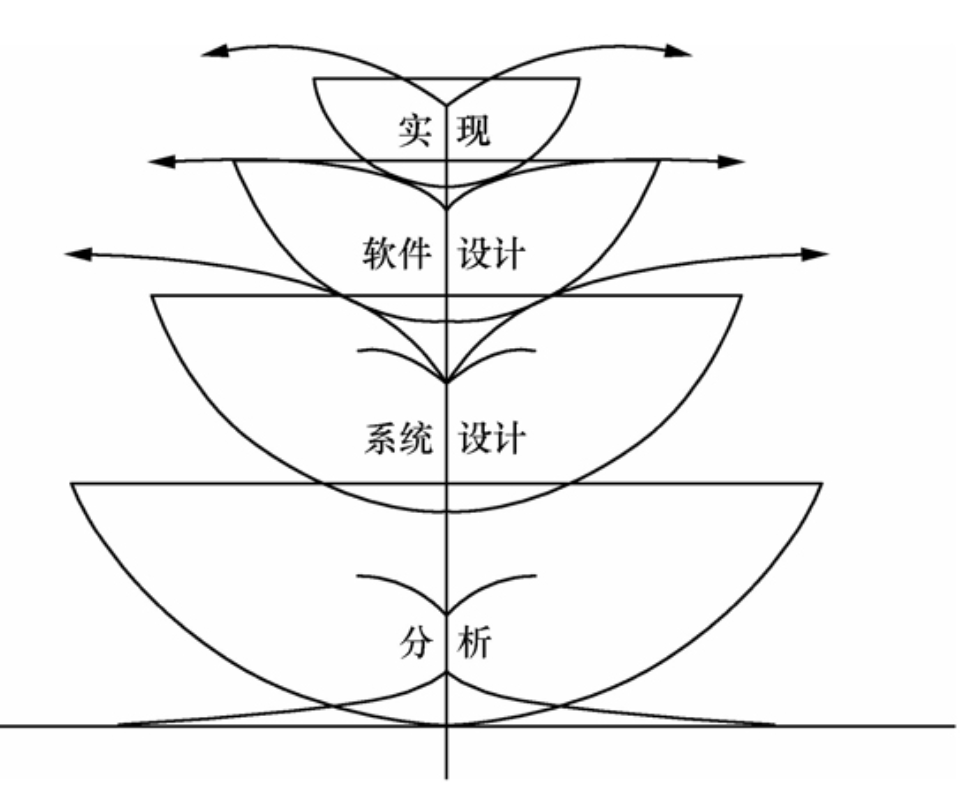
* + 增量模型：非整体开发模型：组装



* + 螺旋模型：结合瀑布模型与增量模型，引入风险分析，使用风险驱动；迭代模型，没迭代一次，螺旋线就前进一周；适用于大规模、高风险的软件项目



* + 喷泉模型：面向对象。以用户需求为动力，以对象作为驱动的模型，克服了瀑布模型不支持软件重用和多项开发活动基础的局限性质



* + 构件集成模型：（开源库？）将食物封装成包含数据和加工该数据操作的对象，并抽象成类；经过一定的设计和实现的类可称为构件，可以在不同的系统中复用；提高可重用性和可靠性
  + 转换模型：
  + 净室模型
* 传统的软件工程：
  + 采用结构化分析、结构化设计和结构化实现
  + 将软件生存周期分为若干阶段
  + 每个阶段的开始和结束都有严格标准，前一阶段结束的标准是后一阶段工作开始的标准
  + 每个阶段结束前，必须严格正式地进行严格的技术审查和管理复审
* 面向对象的软件工程
  + 在开发早期定义一系列面向问题领域的对象，建立对象模型
  + 整个开发阶段统一使用这些对象
  + 并不过分充实和扩展对象模型
  + 开发阶段模糊，开发过程逐步求精，开发活动反复迭代
  + 不再以功能划分导向，以对象作为整个问题分析的中心，围绕对象展开系统的分析与设计
  + 同样把软件开发划分为分析、设计、编码、测试等阶段，但各阶段工作不同。一般在需求分析阶段确定类与对象，设计阶段完成对象内部的详细设计
* 面向对象的软件开发过程：
  + 需求分析与对象抽取
  + 对象详细设计
  + 面向对象的编码和测试

# 第三章 软件需求分析

3.1 软件需求分析的任务

（1）系统的综合需求

1. 功能需求：需要提供的服务与功能
2. 性能需求：技术性能指标，包括存储容量、运行时间等
3. 环境需求：所需要的软、硬件（如机形、外设、操作系统和数据库管理系统等）
4. 接口要求：与环境通信的格式，包括用户接口需求，软件接口需求，意见接口需求，通信接口需求
5. 用户界面需求：人机交互方式，输入输出数据格式等
6. 其他需求：可靠性、安全性、保密性、约束、可移植性和可维护性等

（2）数据需求

建立数据模型，通常为实体联系图。对复杂的数据结构则利用图形工具辅助描绘，如层次方框图和Warnier图等

（3）建立逻辑模型

根据上述两项的需求结果，进行一致性的分析检查，确定系统构成和主要成分，并建立系统的逻辑模型，如数据流图，数据字典及处理方法等。

（4）规格说明书

明确需求与要求。

（5）需求分析评审

发现需求分析的错误和缺陷，修改开发计划

3.2 需求分析的步骤

（1）需求获取：调查研究

对目标系统的运行环境、功能与用户取得一致意见。

常用方法包括：客户访谈、建立联合分析小组、问题分析与确认。并且可通过快速建立软件模型来获取需求。

（2）需求提炼：分析建模

根据用户信息建立目标系统的分析模型，如数据流图、实体联系图、控制流图、状态转换图、用例图、类对象关系及行为图等。

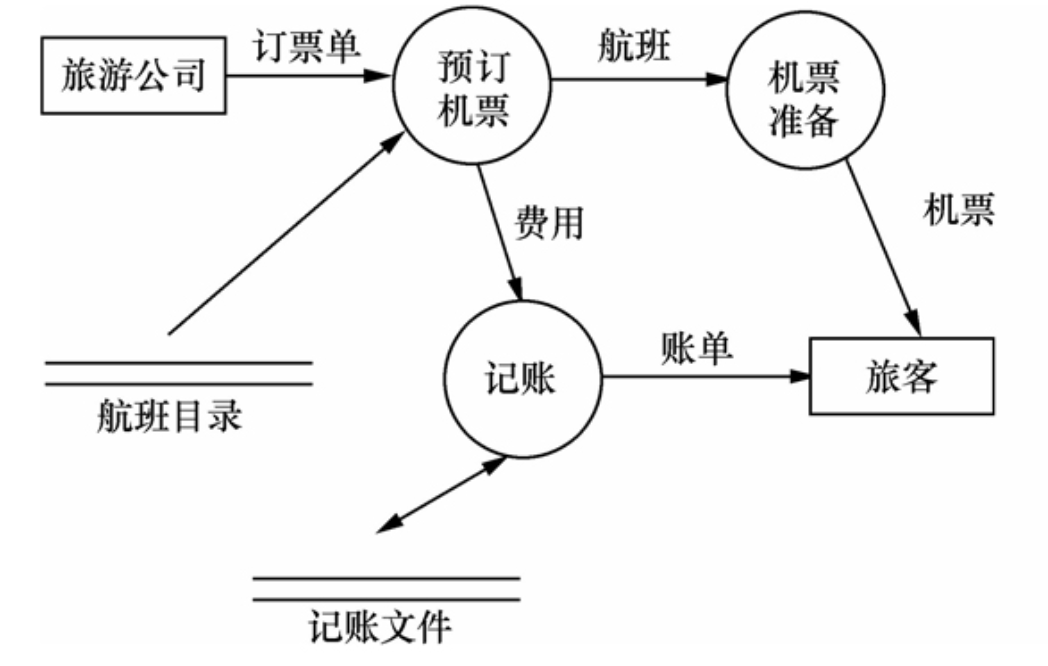
需求分析的常用方法包括：

1. 功能分解方法：含功能、子功能、功能接口三要素。自顶向下生成功能树。
2. 结构化分析方法：面向数据流，从问题空间到某种标表示的映射。功能由数据流图表示，并由数据流图和数据字典构成系统的逻辑模型
3. 信息建模方法：从数据角度来对现实世界建立模型。常用面向对象的实体联系图，有实体、属性和联系构成，但是数据不封闭，没有继承性和消息传递机制支持模型。
4. 面向对象方法：关键在于识别、定义问题域内的类与对象，分析关系并根据操作规则和内存性质建立模型

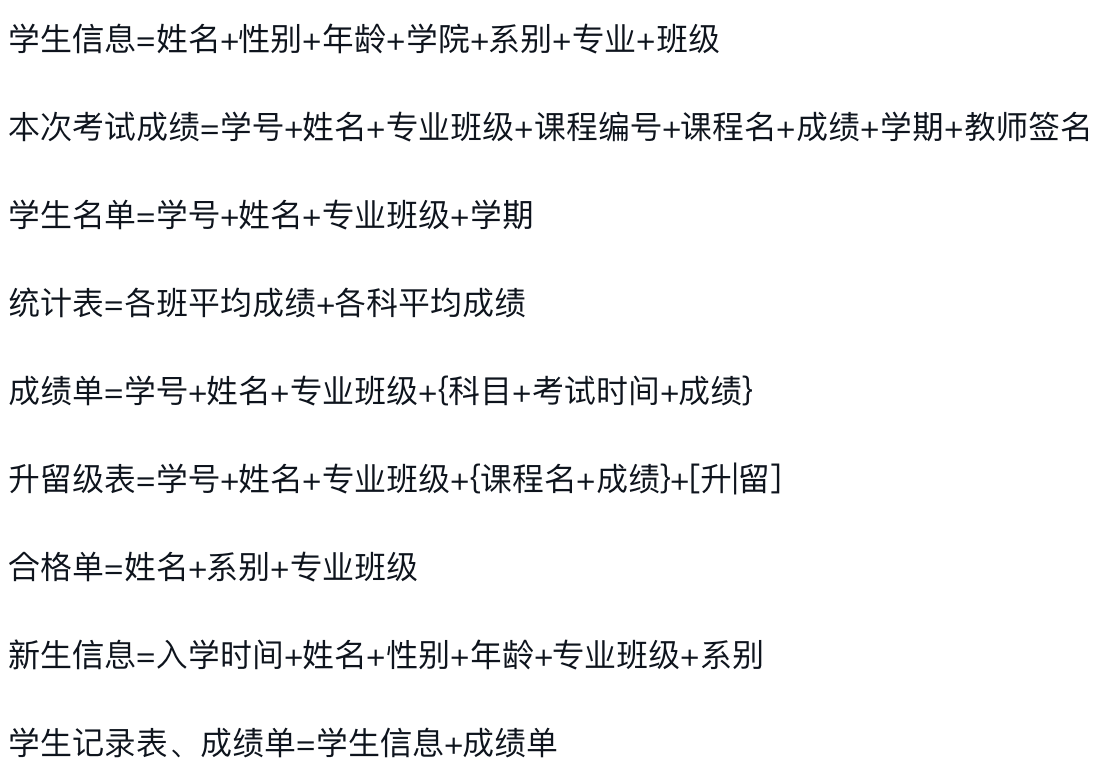
结构化分析方法（Structured Analysis）使用数据流图、数据字典、结构化语言、判定树和判定表等建立节后华说明书的目标文档（SRS）。

结构化分析方法可分为非形式化、版形式化和形式话三类。

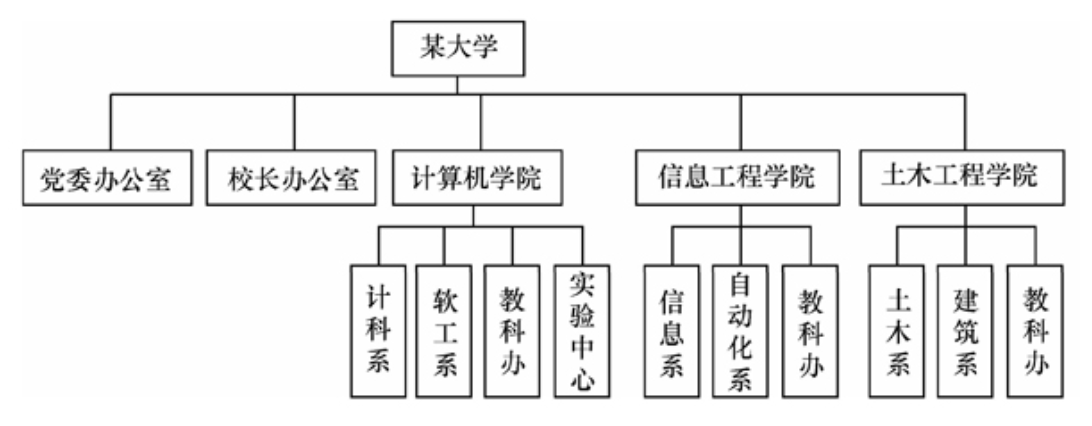
1. 非形式化方法：用自然语言描述需求规格说明
2. 半形式化：用数据流图或实体-联系图建立模型
   1. 数据流图：（详细设计见书）



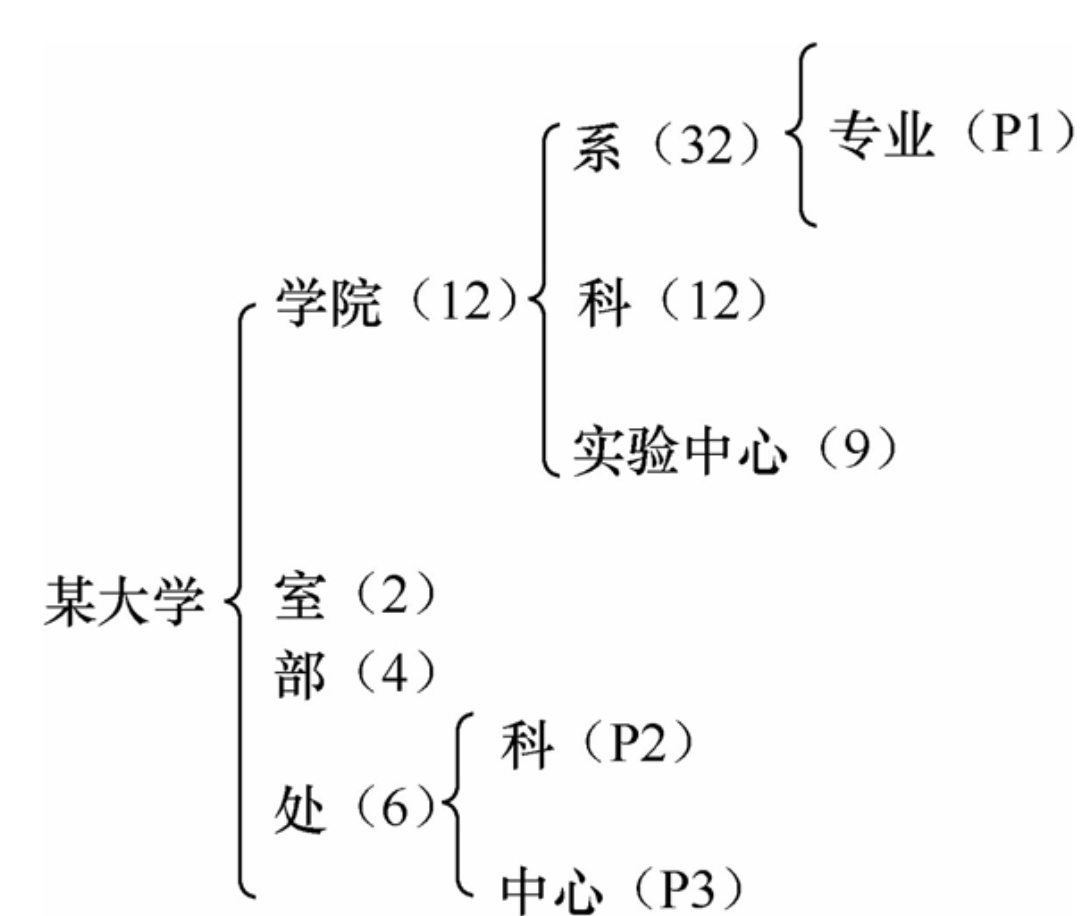
* 1. 数据字典：包括数据流条目、数据存储条目、数据项、加工条目



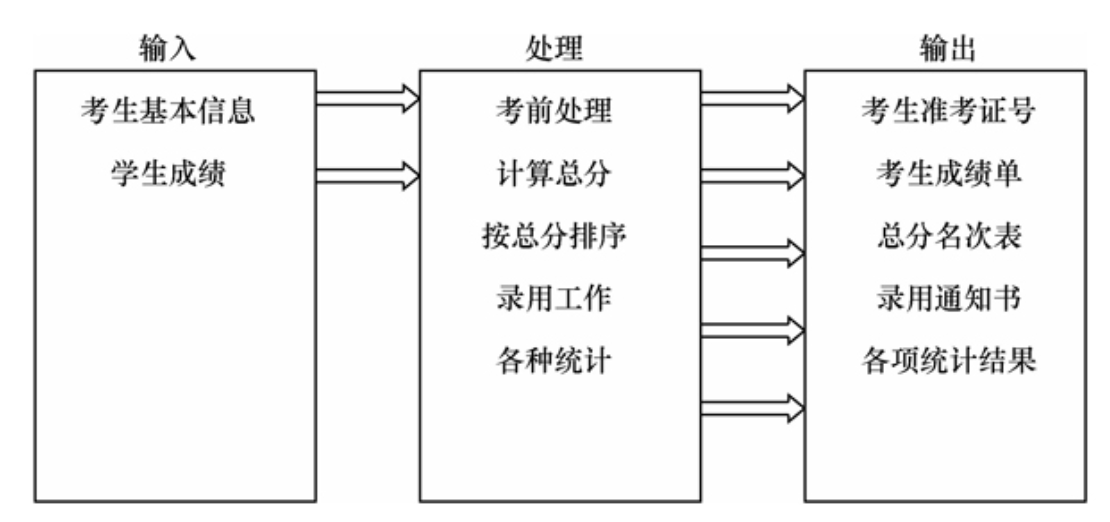
* 1. 层次框图



* 1. Warnier(维纳)图



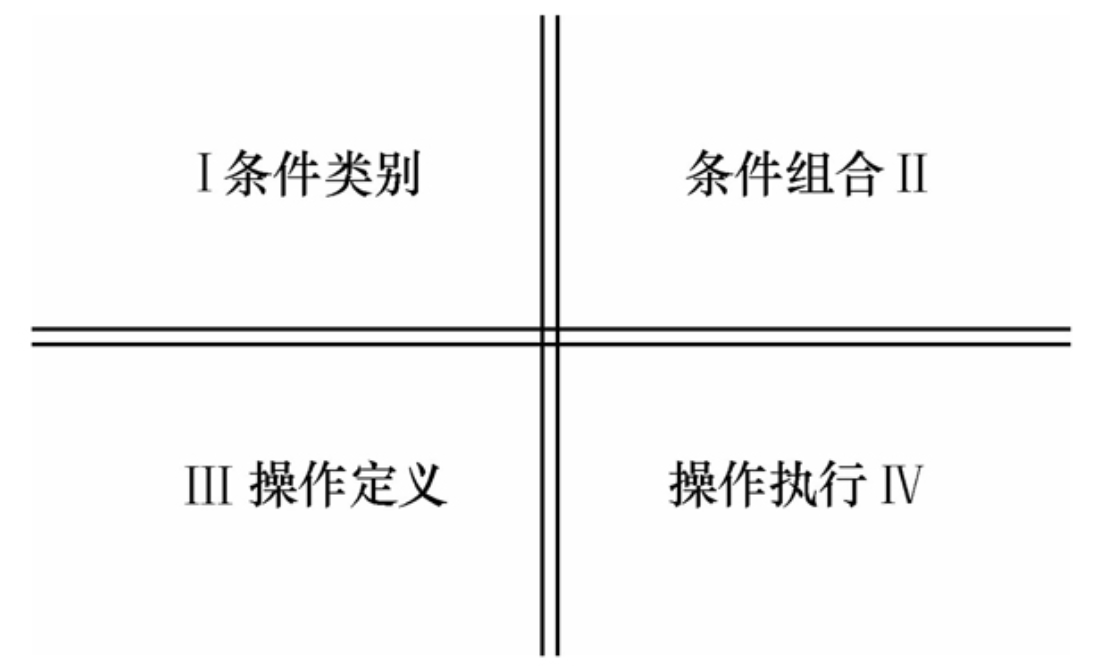
* 1. IPO（Input-Preocess-Output）图



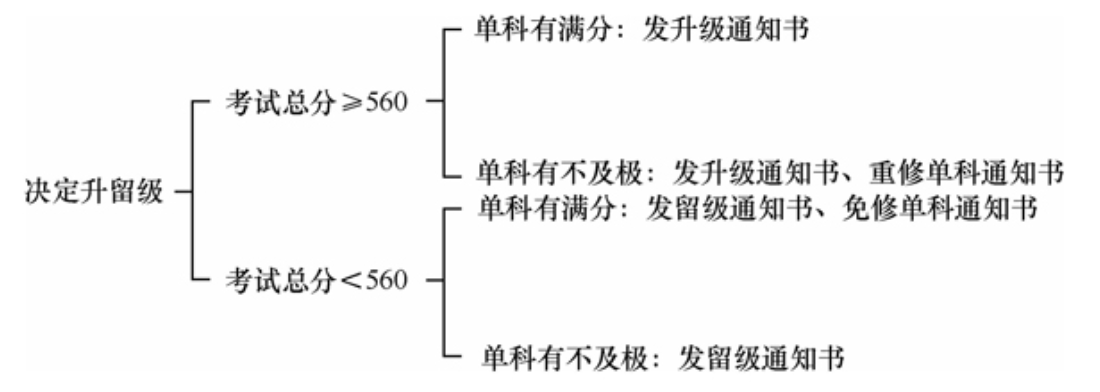
1. 形式化：基于数字的技术（即数学基础）描述系统性质

加工逻辑描述：即对输出数据流和输入数据流的逻辑关系

1. 结构化语言：自然语言结合结构化的形式，清晰易读且简单易懂
2. 判定表：



1. 判定树：



数据库内容需求分析：

1. 信息需求：从数据库中获取的信息的内容和性质
2. 处理需求：软件系统完成的功能以及处理时间、方式等需求
3. 使用需求：数据库运行的安全性、完整性、一致性等限制。查询方式、输入输出格式、响应速度、故障恢复等

建立局部和全局ER图

（3）需求描述：编写SRS

根据标准模板编写SRS模板，如IEEE 830-1998或GB 9385等

主要内容包括：

1，引言

1.1 编写目的，面向对象等

1.2 项目背景：委托单位、开发单位、主管部门，与其他系统的关系等

1.3 定义：专门术语的定义与缩写词的原文

1.4 参考资料

2 任务概述

2.1 目标

2.2 运行环境

2.3 条件与约束

3 数据描述

3.1 静态数据

3.2 动态数据：输出输出数据

3.3 数据库描述

3.4 数据字典

3.5 数据采集

4 功能需求

4.1 功能划分

4.2 功能描述

5 性能需求

5.1 数据精确度

5.2 时间特性：响应时间、更新处理时间、数据转换与传输时间、运行时间

5.3 适用性：操作方式、运行环境、软件结构、开发计划、扩展性等

6 运行需求

6.1 用户界面：屏幕格式、报表格式、菜单格式、输入输出时间等

6.2 硬件接口

6.3 软件接口

6.4 故障处理

7 其他需求：可用性、安全保密性、可维护性、可移植性等

（4）需求验证

分析员与用户一起对需求分析结果进行严格的审查、验证

# 第四章 软件总体设计

4.1 总体设计的目标和任务

软件体系结构设计准则：

1. 普遍适用，并描述各种风格的软件体系结构
2. 在一定的时间内保持稳定。确保接口一致

软件模块设计准则：

1. 降低模块之间的耦合性，提高模块的内聚性
2. 模块结构的深度（模块层次）、宽度（同一层次中模块树）、扇出（对其他模块的直接调用，平均3~4）和扇入应适当
3. 模块的作用范围（模块本身及直接或间接从属于它的模块集合）应该在控制范围内
4. 模块接口设计简单，以降低复杂程度和冗余度
5. 设计功能可预测并能得到验证的模块
6. 适当划分模块规模，并保持其独立性

4.3 软件设计的概念和原理

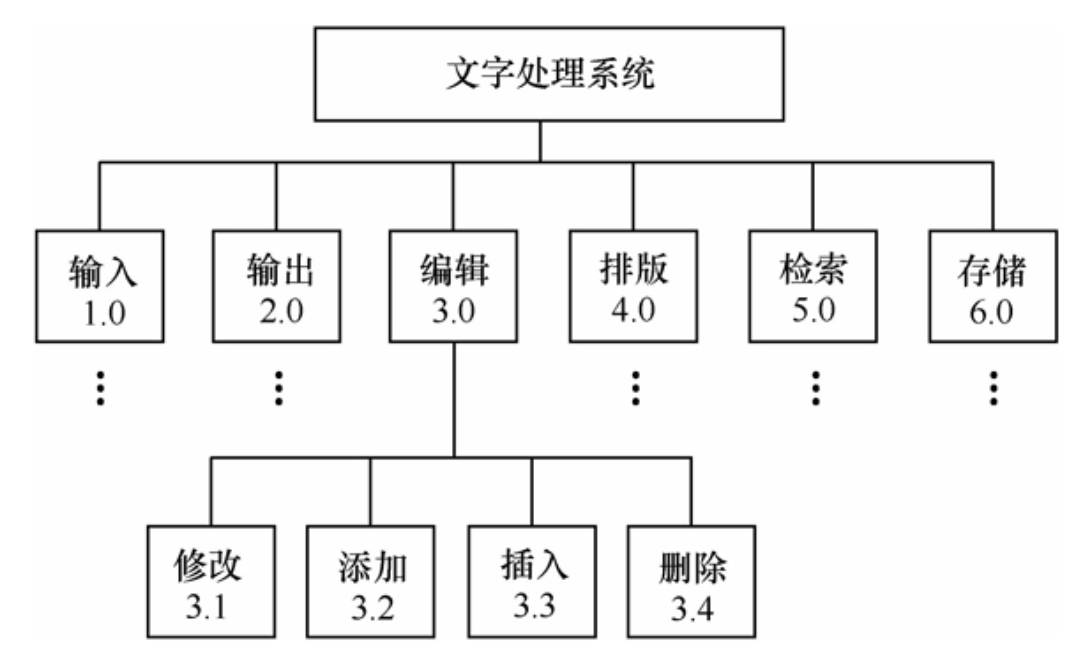
1. 模块和模块化
2. 抽象
3. 信息隐蔽和局部化
4. 模块独立性及其度量
   1. 耦合：模块间的联系程度
      1. 无直接耦合：不传递任何信息
      2. 数据耦合：存在调用关系，信息以参数形式传递
      3. 标记耦合：传递数据结构
      4. 控制耦合：模块调用
      5. 公共环境耦合：两个或多个模块共用一个数据环境，包括全局变量、内存的公共覆盖区，公共环境耦合的复杂程度随着耦合模块个数的增加而显著增加，如果一个模块只是给公共环境送数据，而另一个模块只是从公共环境取数据，这是数据耦合的一种方式，是比较低的耦合；如果两个模块都既往公共环境送数据，又从里面取数据，这种耦合比较高，介于数据耦合和控制耦合之间，如果两个模块共享的数据很多，都通过参数传递可能很不方便，这时可以利用这种耦合
      6. 内容耦合：一个模块访问另一个模块的内部数据，不通过正常入口而转入另一个模块内部，属于最差的耦合
   2. 内聚：模块内各元素之间的联系
      1. 偶然内聚：代码间无联系
      2. 逻辑内聚：代码逻辑功能相同
      3. 时间内聚：需要同时执行的动作
      4. 通信内聚：使用相同的输入或输出数据
      5. 顺序内聚：处理元素顺序执行
      6. 功能内聚：共同完成单一功能

4.4 数据库设计

1. 三级模式：外模式、模式、内模式
2. 逻辑设计原则：概念结构（ER图）-一般数据模型（关系、网状、层次）-DBMS支持的数据模型-优化的数据模型
3. 关系数据库规范化：1NF⊃2NF⊃3NF⊃BCNF⊃4NF⊃5NF

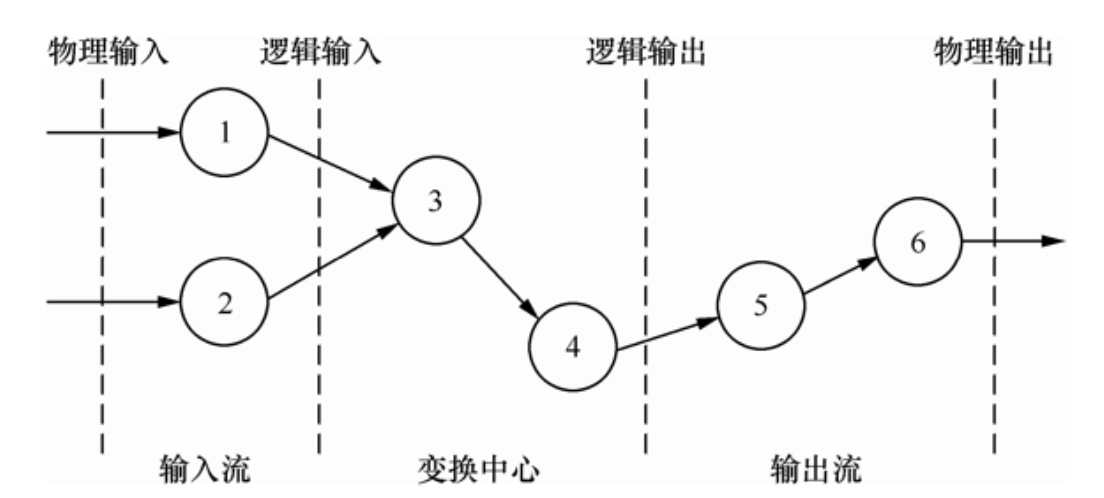
4.5 软件结构设计的图形工具

1. 软件结构图：模块、模块的调用关系
2. 层次图
3. HIPO图：带编号的层次图

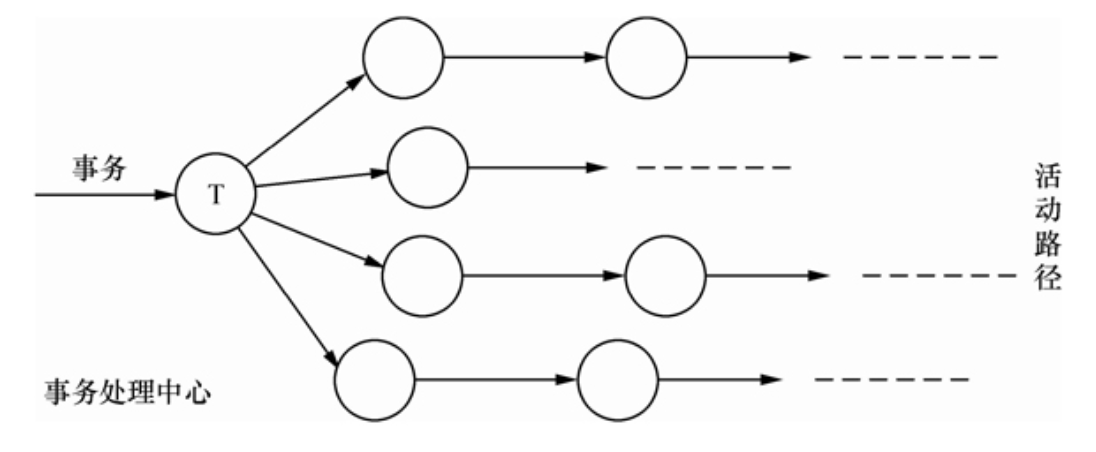


4.6 结构化设计方法

1. 数据流图及其类型
   1. 变换型数据流图



* 1. 事务型数据流图



1. 结构化设计方法的步骤
   1. 复审数据流图
   2. 确定数据流图类型
   3. 分解上层模块，设计中下层模块结构
   4. 根据软件结构设计准则对软件结构求精并改进
   5. 导出接口描述和全程数据结构
   6. 复审
2. 变换型数据流的分析设计
   1. 确定逻辑输入和逻辑输出的边界，找出变换中心
   2. 设计软件结构的顶层和第一层
   3. 设计中下层模块
   4. 对初始结构图优化
3. 事务型数据流图的分析设计