软件工程学基础

龙连春

E-mail: longlc@bjut.edu.cn

办公室：基础楼310

电　话：6739-2760

手　机：13641365218

2006年8月22日

软件工程概述

软件（Software）

软件是“程序以及开发使用维护程序所需的所有文档”，由应用程序，系统程序，面向用户的文档，及面向开发者的文档四部分构成。

软件工程（Software Engineering）

软件工程是在克服60年代末所出现的“软件瓶颈”的过程中逐渐形成与发展的。是一门指导计算机软件系统开发和维护的工程学科，是一门新兴的边缘学科，主要研究如何应用软件开发的科学理论和工程技术来指导大型软件系统的开发。

软件的特点

1、软件是一种逻辑实体，不是具体的物理实体。

2、软件产品的生产主要是研制开发。

3、软件具有“复杂性”，其开发和运行常受到计算机系统的限制。

4、软件成本昂贵，其开发方式目前尚未完全摆脱手工生产方式。

5、软件不存在磨损和老化问题，但存在退化问题。

软件工程的产生和发展

软件工程的发展已经历了四个重要阶段：

1、第一代软件工程 — 传统的软件工程

2、第二代软件工程 — 对象工程

3、第三代软件工程 — 过程工程

4、第四代软件工程 — 构件工程

软件工程的产生和发展

软件工程的发展已经历了四个重要阶段：

1、第一代软件工程 — 传统的软件工程

2、第二代软件工程 —

3、第三代软件工程 — 过

4、第四代软件工程 — 构

60年代末到70年代为了克服“软件瓶颈”提出“软件工程”的名词，将软件开发纳入工程化的轨道，基本形成软件工程的概念、框架、技术和方法。称为传统的软件工程。

软件工程的产生和发展

软件工程的发展已经历了四个重要阶段：

1、第一代软件工程 — 传统的软件工程

2、第二代软件工程 — 对象工程

3、第三代软件工程 — 过程工程

4、第四代软件工程

80年代中到90年代，面向对象的方法与技术得到发展，研究的重点转移到面向对象的分析与设计，演化为一种完整的软件开发方法和系统的技术体系，称为对象工程。

软件工程的产生

软件工程的发展已经

1、第一代软件工程

2、第二代软件工程 — 对

3、第三代软件工程 — 过程工程

4、第四代软件工程 — 构件工程

80年代中开始，人们在软件开发的实践过程中认识到：提高软件生产率，保证软件质量的关键是“软件过程”，是软件开发和维护中的管理和支持能力，逐步形成软件过程工程。

软件工程的

软件工程的发展

1、第一代软件工

2、第二代软件工程 — 工程

3、第三代软件工程 — 过程工程

4、第四代软件工程 — 构件工程

90起年代，基于构件（**Component**）的开发方法取得重要进展，软件系统的开发可通过使用现成的可复用构件组装完成，而无需从头开始构造，以此达到提高效率和质量，降低成本的目的。称为构件工程。

软件工程的研究内容

软件工程是一门新兴的边缘学科,涉及的学科多，研究的范围广。归结起来软件工程研究的主要内容有以下几方面：

- 软件开发方法、技术
- 软件开发工具及环境

}软件开发技术

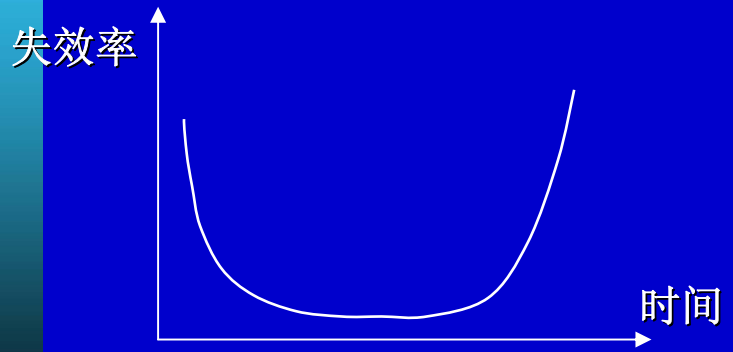
- 软件管理技术
- 软件规范

}软件管理技术

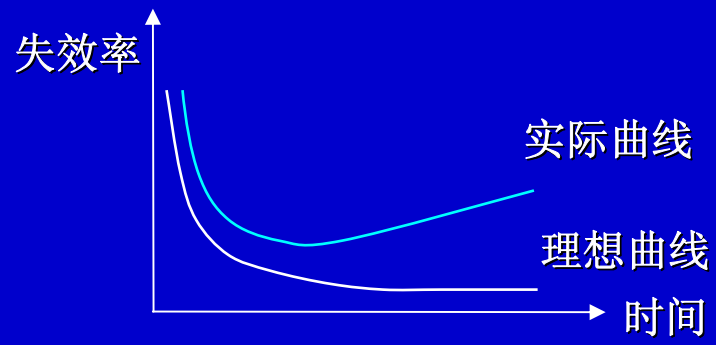
(1)软件开发技术（软件结构、开发方法、工具与软件工程环境、软件工程标准化）

(2)软件工程管理（质量管理，软件工程经济学：成本估算，计划安排）

软、硬件失效情况的对比



硬件失效率曲线



软件失效率曲线

硬件失效率曲线，是一U型曲线。软件失效率曲线，它没有U型曲线的右半翼。因为软件不存在磨损和老化问题，然而存在退化问题。其次，硬件的更新会对软件带来影响。

软件的分类

1、按照软件功能划分

系统软件 — 如操作系统、设备驱动程序等。

支撑软件（实用软件） — 协助用户开发的工具软件，如编辑程序、程序库、图形软件包等。

应用软件 — 如工程与科学计算软件、CAD/CAM软件、CAI软件、信息管理系统等。

2、按照软件规模划分

类 别	参加人数	研制期限	产品规模（源代码行）
微 型	1	1 –4 周	0.5K
小 型	1	1 – 6 月	1K – 2K
中 型	2 - 5	1 – 2 年	5 – 50 K
大 型	5 - 20	2 – 3 年	50 – 500 K
甚大型	100 - 1000	4 – 5 年	1M
极大型	2000 -5000	5 – 10 年	1M – 10 M

3、按照软件工作方式划分

实时处理软件　交互式软件　批处理软件

4、按照软件服务对象的范围划分

项目软件 — 由客户委托开发的软件。

产品软件 — 由软件开发机构开发，提供给市场的。

软件工程过程（Software engineering process）：
是指在软件工具的支持下，所进行的一系列软件工程活动。

通常包括以下四类基本过程：

1、软件规格说明：规定软件的功能及其运行环境

2、软件开发：产生满足规格说明的软件

3、软件确认：确认软件能够完成客户提出的要求

4、软件演进：为满足客户的变更要求，软件必须在使用的过程中演进。

软件工程过程的特性：

1、可理解性　　2、可见性（过程的进展和结果可见）

3、可靠性　　4、可支持性（易于使用工具支持）

5、可维护性　　6、可接受性（为软件工程师接受）

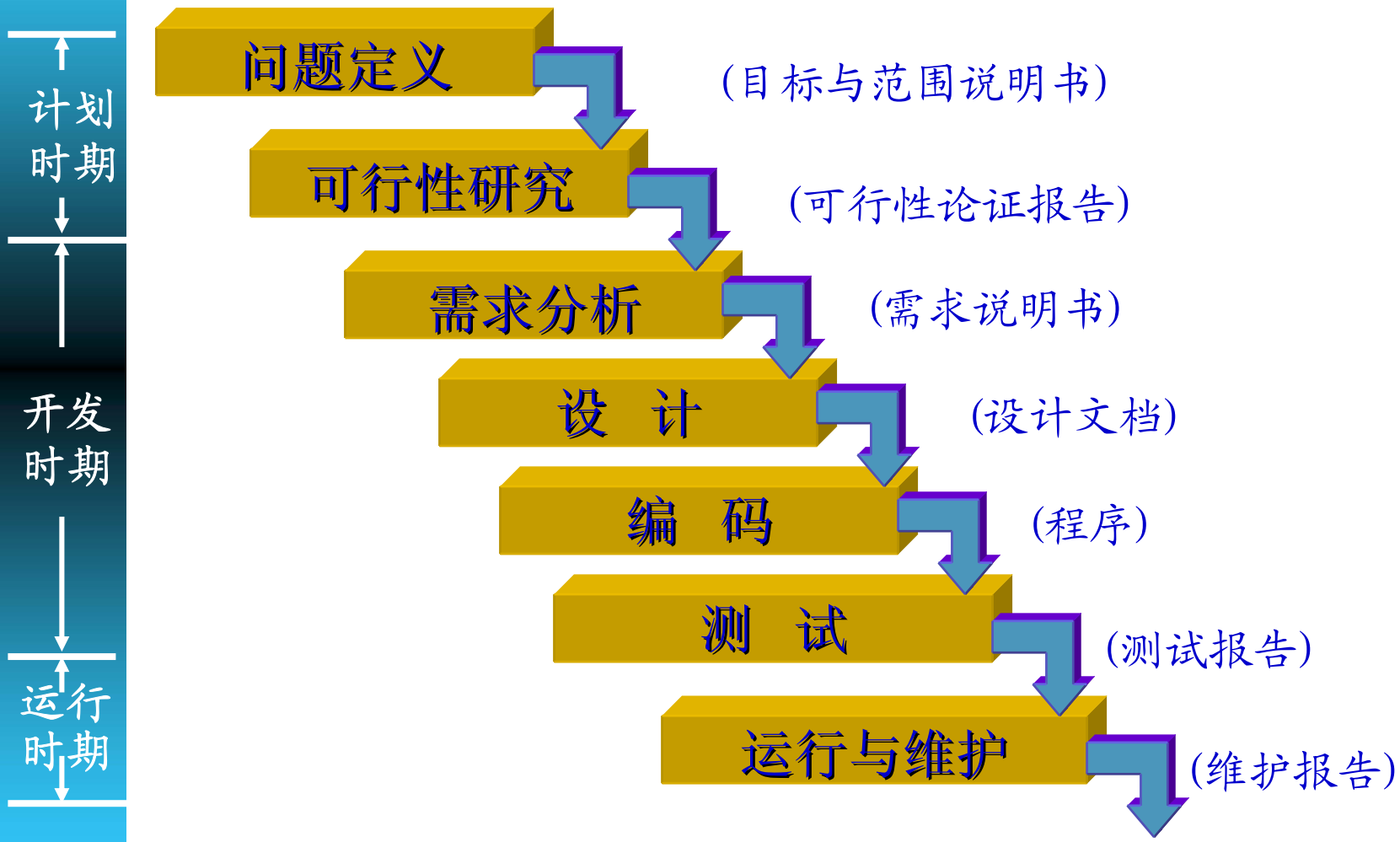
7、开发效率　　8、健壮性（抵御外部意外错误的能力）

软件生存期模型

软件生存周期模型是描述软件开发过程中各种活动如何执行的模型。

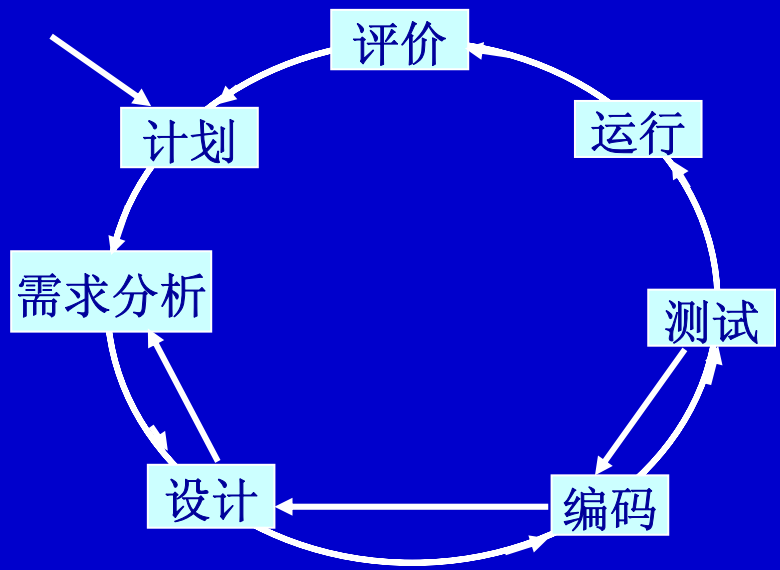
目前有若干软件生存期模型，各种模型有其不同的特点，并适用于不同的开发方法。

瀑布模型(waterfall model)

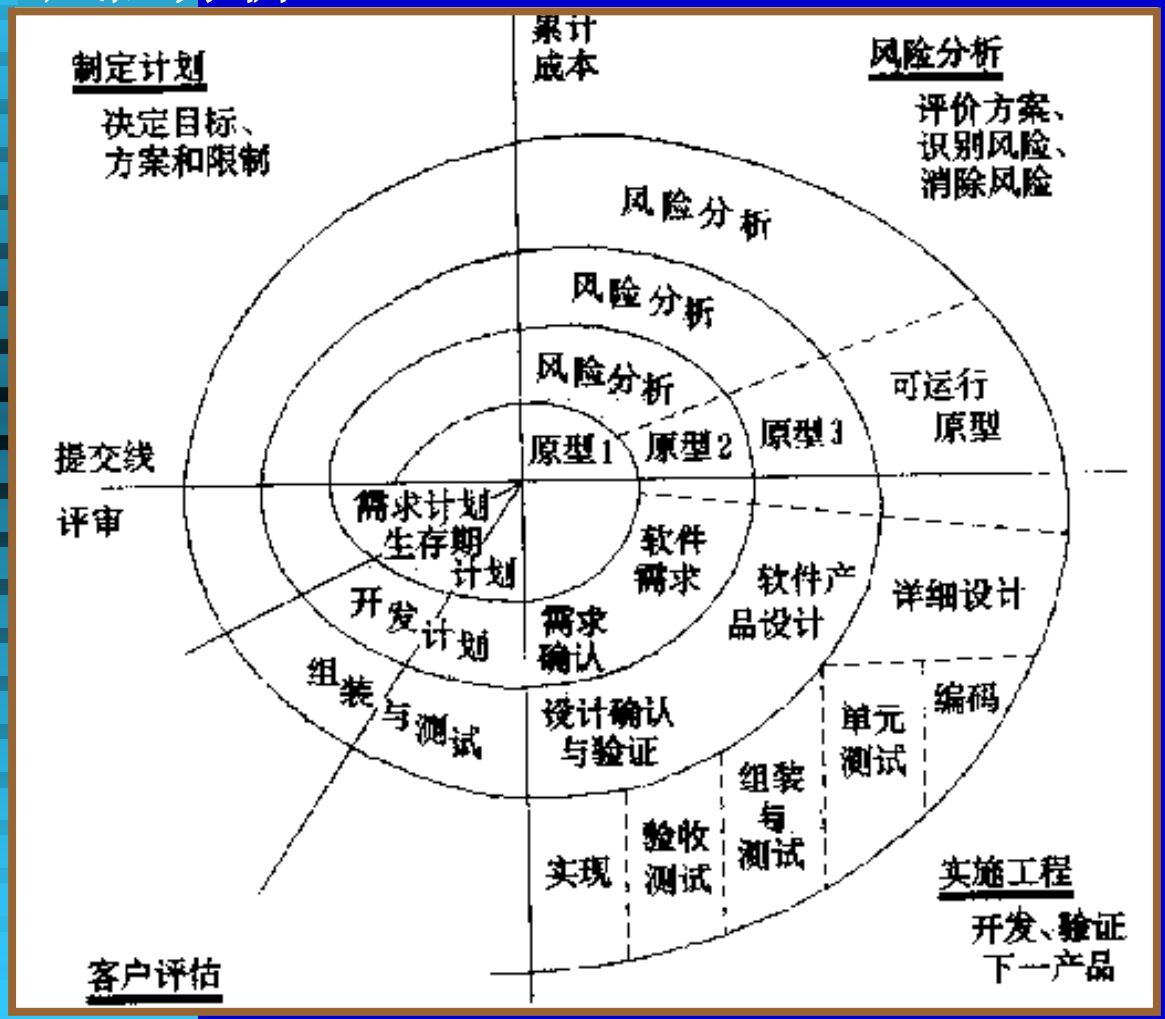


循环模型(loop model)

为了描述软件开发过程中可能的回溯，尤其是维护阶段往往要经历上述各个阶段，采用循环模型描述。



螺旋模型(spiral model) 对于大型软件，只开发一个原型往往达不到要求。螺旋模型将瀑布模型和增量模型结合起来，并加入了风险分析。



螺旋模型将开发过程分为几个螺旋周期，每个螺旋周期可分为4个工作步骤：

1. 确定目标、方案和限制条件；
2. 评估方案、标识风险和解决风险；
3. 开发确认产品；
4. 计划下一周期工作。

喷泉模型(fountain model)

该模型是由B.H.Sollers和J.M.Edwards于1990年提出的一种新的开发模型。主要用于采用对象技术的软件开发项目。它克服了瀑布模型不支持软件重用和多项开发活动集成的局限性，喷泉模型使开发过程具有迭代性和无间隙性。

其特点如下：

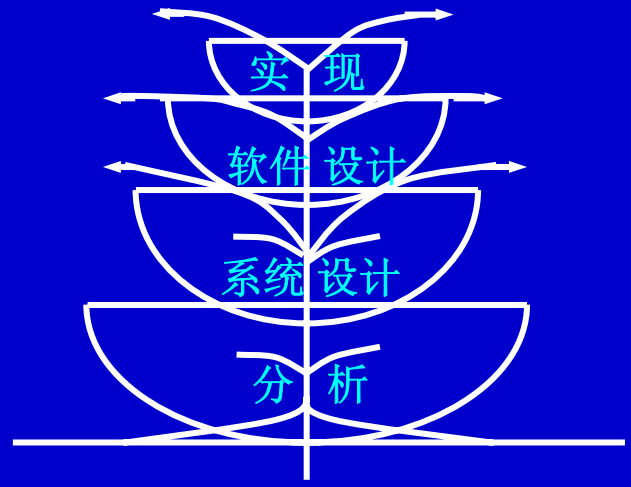
1、开发过程有分析、系统设计、软件设计和实现4个阶段。

2、各阶段相互重叠，它反映了软件过程并行性的特点。

3、以分析为基础，资源消耗成塔型。

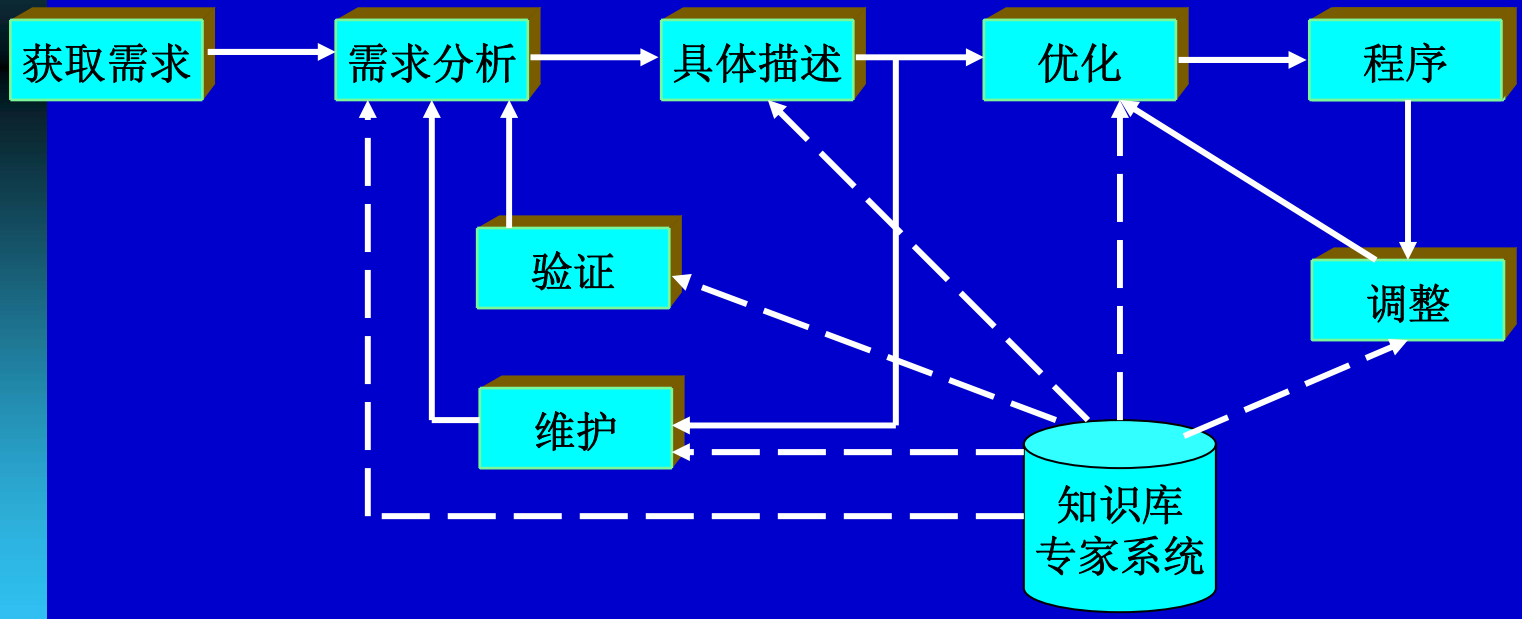
4、反映了软件过程迭代性的自然特性，从高层返回低层无资源消耗。

5、强调增量开发，整个过程是一个迭代的逐步提炼的过程。



智能模型(intelligent model)

也称为基于知识的软件开发模型，是知识工程与软件工程相结合的软件开发模型。其主要特点是必须建立知识库，并将模型本身、软件工程知识、特定领域知识放入知识库。具体描述可以使用形式功能规约，也可以使用知识处理语言描述等。其模型如图所示：



软件开发方法

软件开发的目标是要在规定的投资和时间内，开发出符合用户的需求，高质量的软件，为此需要有成功的开发方法。

软件工程方法包括三要素：方法、工具和过程。

- 方法：完成软件开发各项任务的技术方法；
- 工具：为方法的运用提供自动或者半自动的支撑环境；
- 过程：为开发高质量软件所规定的各项任务的工作步骤。

如何评价软件开发方法：

1、技术特征：层次性、抽象性、并行性、安全性、正确性等。

2、使用特征：易理解性、易修改性、可移植性、工具的支持、可复用性、任务范围、使用的频度等。

3、管理特征：易管理性；成本估算、配置管理等。

4、经济特征：各种效益与代价。

特别要注意：

由于软件与程序是不同的概念，软件开发方法与程序设计方法是两个不同的概念。

软件开发方法可以是针对局部的，也可以是针对全局的。软件工程方法，更加强调和重点研究的是需求分析与软件设计的开发方法。

软件开发方法可分为几大类：

- 面向过程的开发方法（传统的）
- 面向对象的开发方法
- 基于构件的开发方法

结构化开发方法（Structured Developing Method）

是一种面向数据流的开发方法，是现有的软件开发方法中最成熟，应用最广泛的方法，主要特点是快速，自然和方便。

结构化方法总的指导思想是自顶向下、逐步求精，以数据流，数据的封闭性准则来逐层分解的，它的基本原则是功能的分解与抽象。

结构化方法强调结构的合理性。提出了一组提高软件结构质量的准则，如功能的分解与抽象、模块独立性、信息屏蔽等。

原型化方法（Prototyping Method）

原型是软件开发过程中，软件的一个早期可运行的版本，它反映了最终系统的部分重要特性。

原型化方法的基本思想是花费少量代价建立一个可运行的系统，使用户及早获得学习的机会，原型化方法又称速成原型法（*Rapid Prototyping*）。强调的是软件开发人员与用户的不断交互，通过原型的演进不断适应用户任务改变的需求。将维护和修改阶段的工作尽早进行，使用户验收提前，从而使软件产品更加适用。

面向对象方法

OOSD（Object-Oriented Software Development）法这是80年代推出的一种全新的软件开发方法。非常实用而强有力，被誉为90年代软件的核心技术之一。

其基本思想是：对问题领域进行自然的分割，以更接近人类通常思维的方式建立问题领域的模型，以便对客观的信息实体进行结构和行为的模拟，从而使设计的软件更直接地表现问题的求解过程。面向对象的开发方法以对象作为最基本的元素，是分析和解决问题的核心。

软件复用技术

“软件重用”或“软件复用”（Software Reuse）是指将已有的软件成分用于构造新的软件系统。该技术是提高软件生产率和质量，降低成本的有效方法。

复用方式

- 复用程序：包括目标代码和源代码的复用，可通过连接(Link)、绑定(Binding)、包含(include)等功能支持及对象链接及嵌入(OLE)技术实现。
- 复用设计：设计结果比源程序的抽象级别高，因此复用受环境影响小。可以通过从现有系统中提取全部或者部分的设计构件，或者独立于具体应用开发设计构件。

可复用的构件

构件是指可以被明确标识的软件制品，可以是软件开发不同阶段的产品。

可复用构件是指可被其它系统复用，用于构成新系统的构件。

可复用构件的特性：

1. 独立性　解决相对独立的问题，与外界联系尽量少。
2. 完整性　既要包括完整的解决方案，还定义相应操作。
3. 通用性　在同类应用中具有一般性。
4. 可标识性　通过合适的命名，构件所解决的问题是可标识的。
5. 可适应性　适应环境变化。
6. 可靠性　对各个使用它的系统都具有高的可靠性。

软件工具及软件开发环境

CASE（计算机辅助软件工程）的两个阶段:

1、依赖于软件内生命周期各阶段的分散工具。

2、软件开发环境（Software Development Environment)

软件工程环境（Software Engineering Environment)

是包括方法、工具和管理等多种技术在内的综合系统，应具备以下特点:

① 紧密性（各种工具紧密配合工作）

② 坚定性（环境可自我保护，不受用户和系统影响，可实现非预见性的环境恢复）

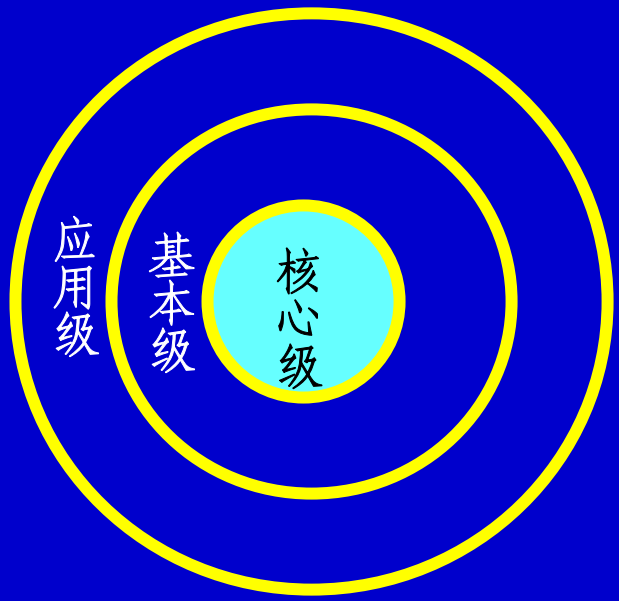
③ 可适应性（适应用户要求，环境中的工具可修改、增加、减少）

④ 可移植性（工具可移植）

典型的软件工程环境具有三级结构：

1. 核心级（核心工具组、数据库、通讯工具、运行支持功能、与硬件无关的移植接口）
2. 基本级（环境的用户工具，编译、编辑程序，作业控制语言的解释程序等）
3. 应用级（应用软件的开发工具）

典型的软件工程环境



软件设计与编码

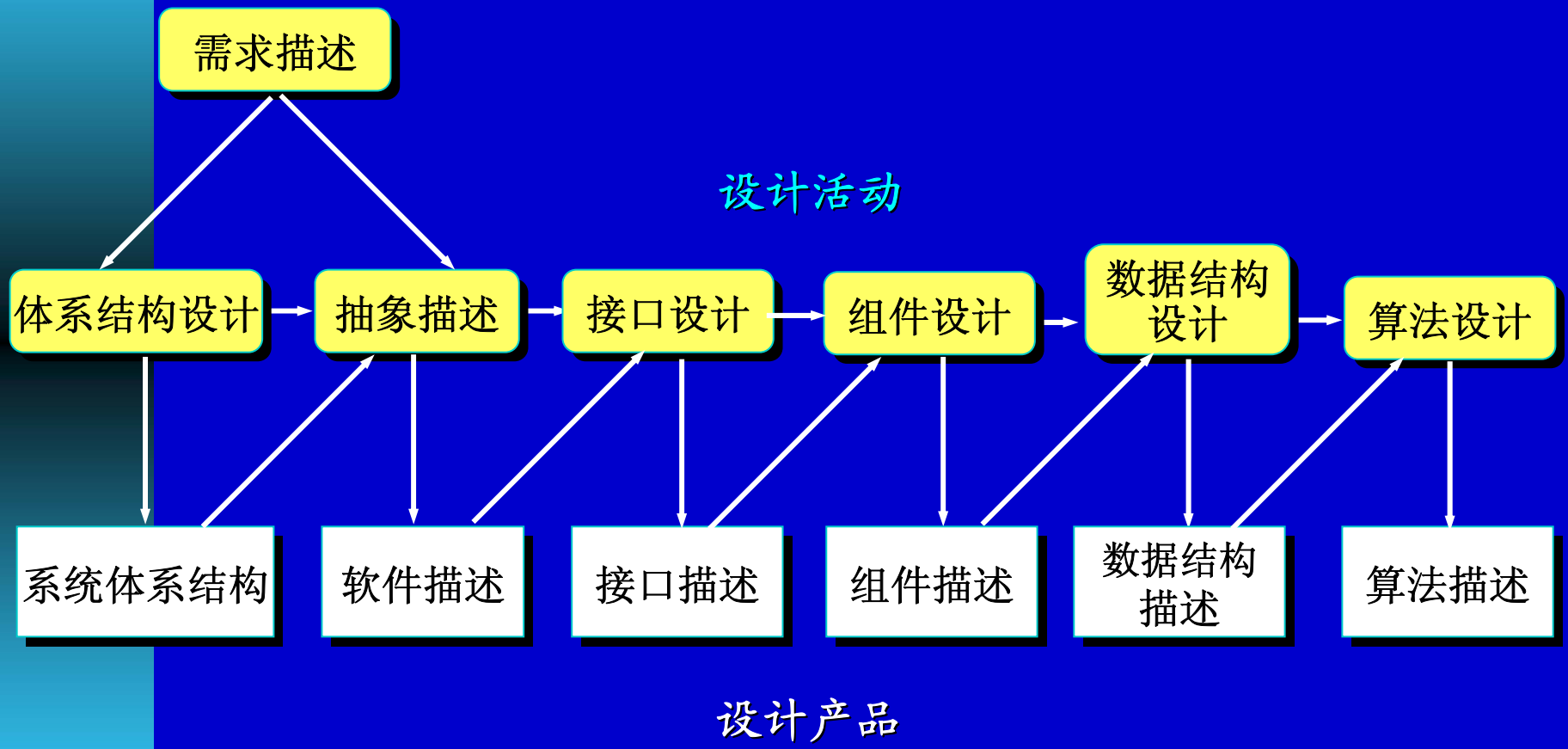
软件设计阶段的任务与目标

软件设计是对实现软件的结构、系统的数据、系统组件之间的接口以及所用算法的描述。

软件设计是软件开发的关键步骤，直接影响软件的质量。

在软件需求分析阶段已经完全弄清楚了软件的各种需求，较好地解决了所开发的软件“做什么”的问题，并已在软件需求说明书和数据要求说明书中详尽和充分地阐明了这些需求以后，下一步就要着手实现软件的需求，即软件设计阶段要解决“怎么做”的问题。

设计过程的一般模型


需求描述
设计活动
体系结构设计
抽象描述
接口设计
组件设计
数据结构设计
算法设计
系统体系结构
软件描述
接口描述
组件描述
数据结构描述
算法描述
设计产品

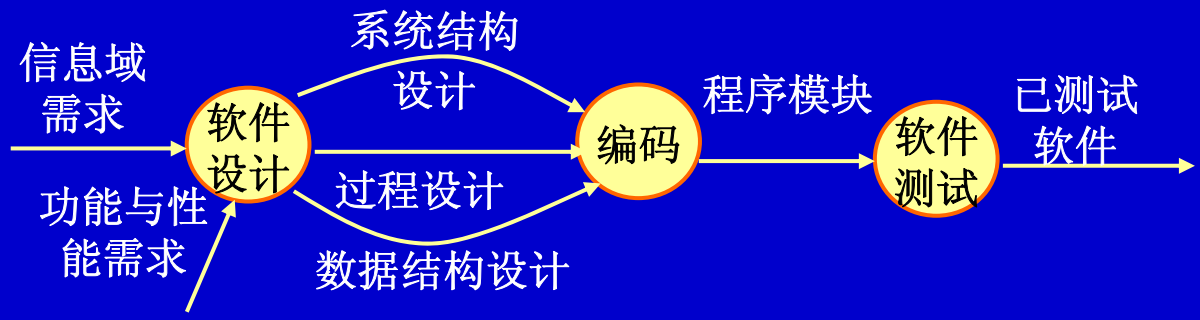
软件设计在开发阶段中的重要性

软件设计阶段要解决“如何做”的问题。这显然是整个软件开发过程的核心问题，所有的开发工作都将根据设计的方案进行。系统的总体结构在该阶段决定，因此软件的总体设计决定了系统的质量。

系统结构设计定义软件系统的整体结构，是软件开发的核心步骤。在设计步骤中，建立软件主要成份之间的关系。

过程设计则是把结构成份转换成软件的过程性描述。

在编码步骤中，根据这种过程性描述，生成源程序代码，然后通过测试，最终得到完整有效的软件。



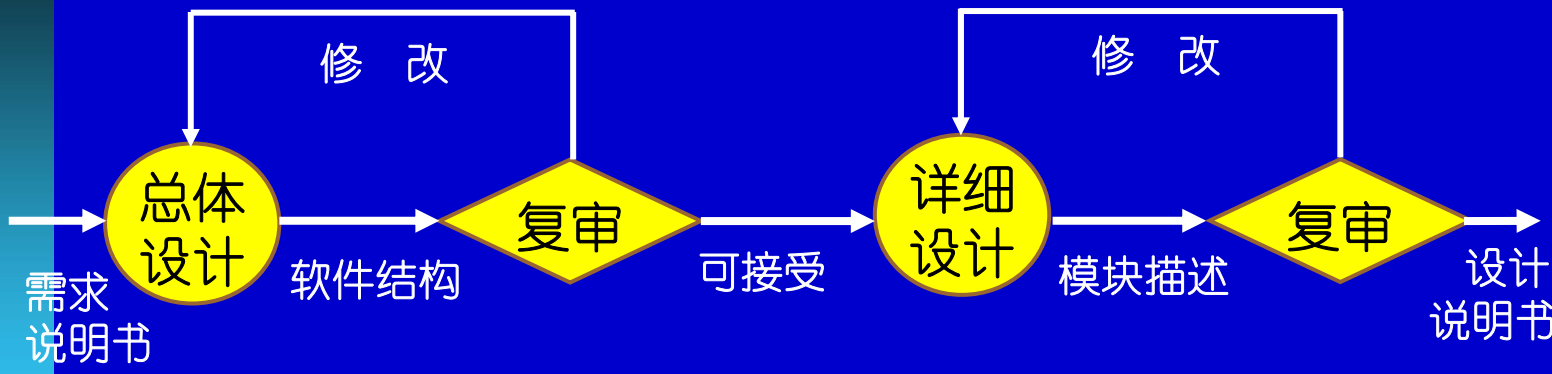
开发阶段信息流

软件设计阶段的任务

一、软件设计的任务

从工程管理的角度来看，软件设计分两步完成；分为总体设计（概要设计）和详细设计两个阶段。其工作流程如图所示。

首先做概要设计，将软件需求转化为数据结构和软件的系统结构。然后是详细设计，即过程设计。通过对结构表示进行细化，得到软件详细的数据结构和算法。



软件设计工作流程

因此，软件设计阶段的任务可分为三部分：

1、划分模块，确定软件结构

开发方法不同，确定软件结构的方法也不同。

一般包括确定系统的软件结构，分解模块 ，确定系统的模块层次关系。

2、确定系统的数据结构

数据结构的建立对于信息系统而言尤为重要。要确定数据的类型，组织、存取方式，相关程度及处理方式等。

3、设计用户界面

作为人机接口的用户界面起着越来越重要的作用，它直接影响到软件的价值与寿命。

二、软件设计的目标

在设计阶段应达到的目标是：

提高可靠性；提高可维护性；提高可理解性；提高效率。

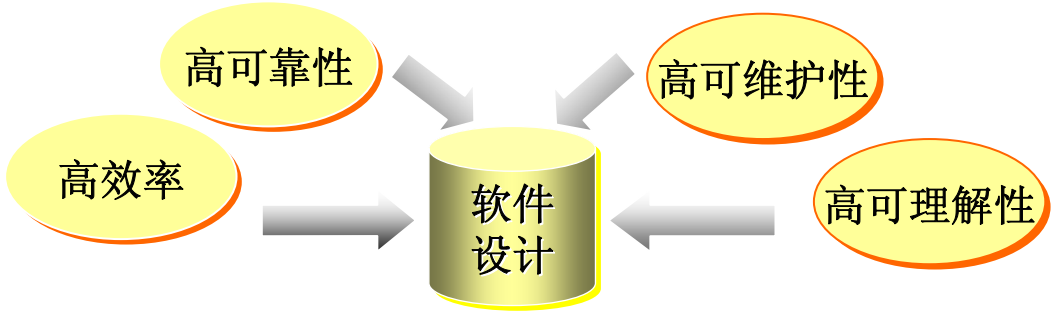
衡量该目标的准则：

1、软件实体有明显的层次结构，利于软件元素间控制。

2、软件实体应该是模块化的，模块具有独立功能。

3、软件实体与环境的界面清晰。

4、设计规格说明清晰、简洁、完整和无二义性。



软件结构与软件结构图

软件结构是软件模块之间关系的表示，它决定了整个系统的结构，也确定了系统的质量。模块之间的关系可有多种，但都可以归结为一种层次关系。

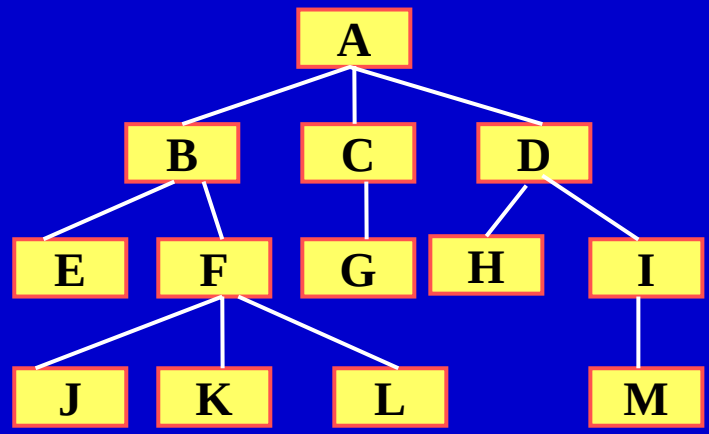
软件结构图是总体设计阶段的主要描述工具，它描述了构成系统的基本元素——模块及模块之间的调用关系，模块之间的数据传递关系。

软件结构的基本概念

软件结构表示软件系统的构成，是软件模块间关系的表示，下图则为软件结构示意图。下面先介绍几个相关的的概念。

一、模块（Module）

模块是程序对象有名字的集合。例如，过程、函数、子程序、宏等，是构成软件系统结构的基本元素。

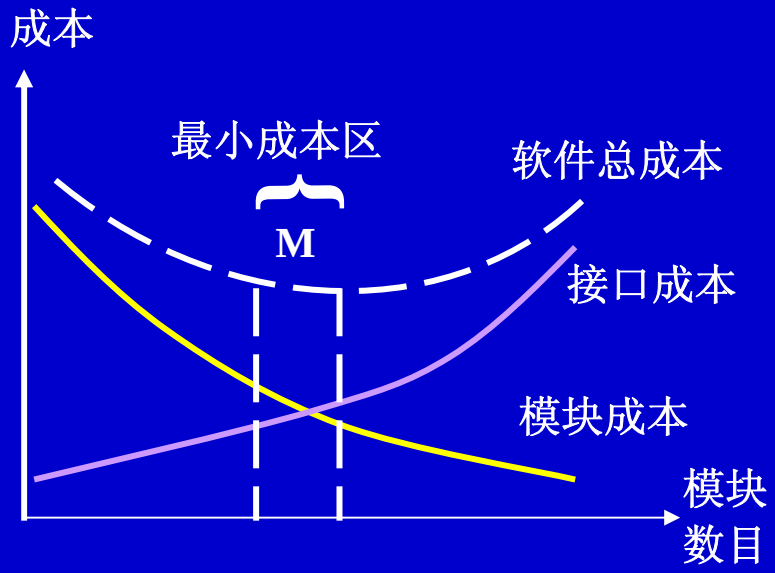


软件结构图

二、模块化

模块化就是将系统划分为若干个模块，每个模块完成一个子功能。模块化的目的是将系统“分而治之”，模块化能够降低问题的复杂性，使软件结构清晰，易阅读、易理解，易于测试和调试，因而也有助于提高软件的可靠性。

模块化降低软件复杂度



模块化与软件成本的关系

并非模块分得越小越好，因为模块之间接口的复杂度和工作量增加。显然，每个软件系统都有一个最佳模块数M。注意选择分解的最佳模块数。右上图描述了模块化与软件成本的关系。

模块的独立性（*module independence*）

模块化方法已经为所有工程领域所接受。模块化的主要优点是，模块化设计降低了软件系统的复杂性，使得系统容易修改，同时使得系统各个部分可以并行开发，从而提高了软件的生产率。

“模块”，又称“构件”，一般指用一个名字可调用的一段程序。它一般具有如下三个基本属性：

(1)功能　即指该模块实现什么功能，做什么事情。必须注意：模块功能，应是该模块本身的功能加上它所调用的所有子模块的功能。

(2)逻辑　即描述模块内部怎么做。

(3)状态　即该模块使用时的环境和条件。

所谓模块的独立性，是指软件系统中每个模块只涉及软件要求的具体的子功能，而和软件系统中其他模块的接口是简单的。即功能专一，模块之间无过多的相互作用的模块。

这种类型的模块可以并行开发，模块独立性越强，开发越容易。独立性强的模块，还能减少

模块独立性的度量标准是两个定性准则：

耦合性　用于描述模块之间联系的紧密程度。

内聚性　用于描述模块内部联系的紧密程度。

模块独立性比较强的模块应该是具有高内聚性和的低耦合度。

结构化设计（SD）方法

结构化设计方法（*Structured Design*，SD ）是结构化开发方法的核心，主要完成软件系统的总体结构设计。

SD法的设计步骤

由于软件具有两类特征：

软件〈 层次性 ： 反映软件整体的性质（结构图）

过程性 ： 反映局部的性质（框图）

1、总体设计

解决系统的模块结构，即分解模块，确定系统模块的层次结构。任务:

① 划分模块

② 确定模块功能

③ 确定模块间调用关系

④ 确定模块间界面

文档：模块结构图及其模块功能说明。

2、详细设计

对模块图中每个模块的过程进行描述，常用的描述的方式有：伪代码，

流程图，N-S图，PAD图等。

SD法的设计总则

“降低块间联系，提高块内联系”

按照“降低块间联系，提高块内联系”的设计总则进行修改，完善系统的模块图，写出模块的功能说明。

具体应从以下方面改进:

1）尽可能建立功能模块

功能模块具有最强的内聚性，应满足信息屏蔽原则:

一个模块内所包含的信息（过程和数据）对不需要这些信息的模块是不能访问的（黑盒）。

功能模块的组成:

执行某项任务的部分

出错处理部分

返回结束标志

2）消除重复功能

若两模块含有重复的部分，应设法将重复的功能消去。重复部分有完全相同和部分相同的情况，在确定重复部分及实施方案时，一定要谨慎小心。

3）模块的大小适当

模块大小指其篇幅，一般模块大小约50-100行为宜。

4）模块的扇入扇出数不宜太多

一个模块调用其他模块的个数，称为该模块的扇出。模块的扇出不宜过大，一般认为不要超出 7 个。

一个模块被其他模块调用的个数，称为该模块的扇入。扇入越大，除服务性模块外，模块的扇入扇出数不宜太多。否则块间联系增加。

详细设计

一、任务

详细设计阶段的任务是开发一个可以直接转换为程序的软件表示，即对系统中每个模块的内部过程进行设计和描述。

二、常用的描述方法工具

1、流程图

2、结构化流程图（N-S图）

3、PAD图—问题分析图

用户界面应具有的特性

1、可使用性

① 使用简单

② 用户界面中所用术语的标准化和一致性

③ 具有HELP功能

④ 快速的系统响应和低的系统成本

⑤ 具有容错能力

2、灵活性

① 考虑用户的特点、能力、知识水平。

② 提供不同的系统响应信息。

③ 提供根据用户需求制定和修改界面。

3、界面的复杂性与可靠性

复杂性—界面规模及组织的复杂程度。应该愈简单愈好。

可靠性—指无故障使用的时间间隔。用户界面应该能够保证用户正确、可靠地使用系统，及程序、数据的安全。

用户界面设计的任务

这部分工作应该与软件需求分析同步进行。包括以下内容:

1、用户特性分析 — 用户模型

了解所有用户的技能和经验，针对用户能力设计或更改界面。从以下方面分析:

用户类型—通常分为：外行型、初学型、熟练型、专家型。

用户特性度量—与用户使用模式和用户群体能力有关。

包括：用户使用频度、用户用机能力、用户的知识、思维能力等。

2、用户界面的任务分析 —任务模型

是对系统内部活动的分解，不仅要进行功能分解，还要包括与人相关的活动。每个加工即一个功能或任务。

3、确定用户界面类型

用户界面的基本类型

用户界面设计的类型主要有问题描述语言，数据表格、图形与图标、菜单、对话框及窗口等。每一种类型都有不同的特点和性能。讨论以下类型：菜单、图象、对话框和窗口。

1、菜单（menu）

按照显示方式

正文菜单、图标菜单、正文和图标混合菜单，如：开始菜单。

按屏幕位置和操作风格

固定位置、浮动位置（弹出）、下拉式、嵌入式

用户界面的基本类型

2、图　象

在用户界面中，加入丰富多彩的画面，将能够更加行象地为用户提供有用的信息，以达到可视化的目的。主要的处理操作有：图象的隐蔽和再现、屏幕滚动和图案显示、动画等。

3、对话框

对话框是在需要时，显示在屏幕上一个矩形区域内的图形和正文信息。通过对话，实现系统和用户之间的通信。

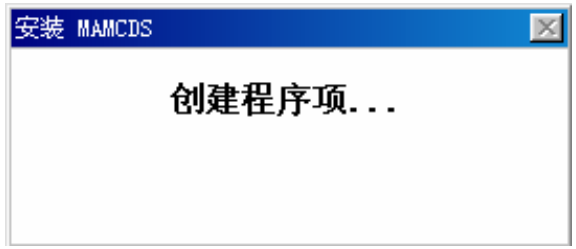
对话框显示的方式与弹出式菜单类似，即瞬时弹出。同时，系统还应将对话框所覆盖的原图象进行保存，以便在对话结束后能立即恢复。

有三种对话形式：

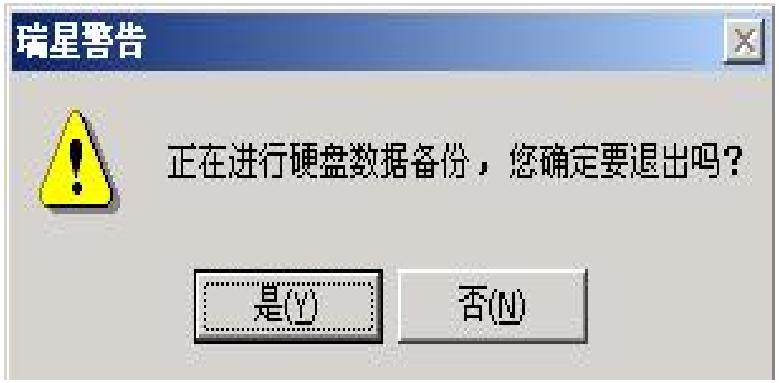
必须回答式

无需回答式

警告式



无需回答式对话框



必须回答式对话框

警告式对话框

4、窗口（window）

图形学中称为视图区（Viewport），视为虚拟屏幕。一个实用窗口，可包含部件：

菜单区（menu bar）　　图标区（icon bar）

标题区（title bar）　　移动区（move bar）

大小区（size bar）　　退出区（quit bar）

用户工作区（user’s work bar）

横向滚动区（horizontal scroll bar）

纵向滚动区（vertical scroll bar）

数据I/O界面设计

数据I/O界面，是系统的重要组成部分。主要从输入速度和减少出错率考虑。

1、尽量减少输入工作量

- 对相同内容输入设置默认值
- 自动填入
- 列表选择或点击选择

2、输入屏幕与输入格式匹配

即屏幕显示按照数据使用频率、重要性、次序等组织。

3、数据输入的一般规则

- 确定输入
- 交互动作
- 确定删除
- 提供反馈

软件测试

一、软件测试的目的和重要性

因为开发工作的前期不可避免地会引入错误，测试的目的是为了发现和改正错误，这对于某些涉及人的生命安全或重要的军事、经济目标的项目显得尤其重要。

- 1963年美国飞往火星的火箭爆炸，原因是FORTRAN程序：DO 5 I=1，3
- 误写为：DO 5 I=1. 3 损失1000万美元。
- 1967年苏联“联盟一号”宇宙飞船返回时因忽略一个小数点，在进入大气层时打不开降落伞而烧毁。

二、软件测试的特点

- 1、软件测试的开销大
- 按照Boehm的统计，软件测试的开销大约占总成本的30%-50%。例如：APPOLLO登月计划，80%的经费用于软件测试。

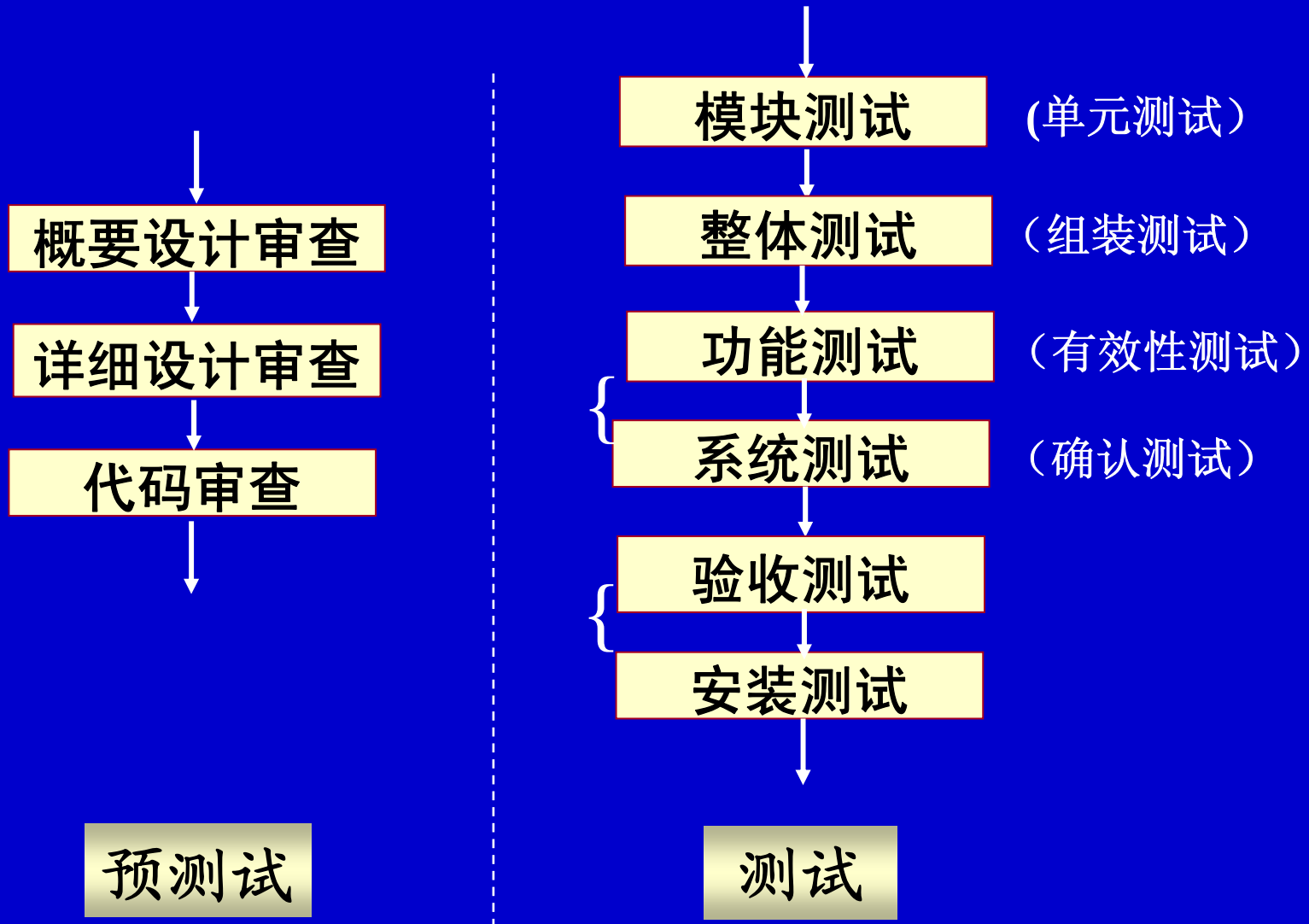
- 2、不能进行“穷举”测试
- 只有将所有可能的情况都测试到，才有可能检查出所有的错误。但这是不可能的：

- 3、软件测试难度大
- 根据上述分析，既然不能进行“穷举”测试，又要查出尽可能多的错误，软件测试工作的难度大。

三、软件测试的基本原则

- 1、尽量不由程序设计者进行测试。
- 2、关键是注重测试用例的选择。
- 输入数据的组成（输入数据、预期的输出结果）
- 既有合理输入数据，也有不合理的输入数据。
- 用例既能检查应完成的任务，也能够检查不应该完成的任务。
- 长期保存测试用例。

四、测试的基本步骤


概要设计审查
详细设计审查
代码审查
预测试
模块测试
(单元测试)
整体测试
(组装测试)
功能测试
(有效性测试)
系统测试
(确认测试)
验收测试
安装测试
测试

软件测试方法

一、静态分析方法

- 指以人工的、非形式化的方法对程序进行分析和测试。

二、动态测试方法

- 通过选择适当的测试用例，执行程序。

软件测试的步骤

一、模块测试（Module Testing）

二、组装测试（Integration Testing）

也称为联合测试或集成测试，重点测试模块的接口部分，需设计测试过程使用的驱动模块或桩模块。

三、确认测试（validation testing）

又称为有效性测试或功能测试。其任务是验证系统的功能、性能等特性是否符合需求规格说明。

四、系统测试（system testing ）

- 将经过确认测试的软件，与计算机硬件、外设、
- 支持软件等一起，在实际运行环境下测试。

五、验收测试（acceptance testing）

验收测试是以用户为主的测试。

软件维护

软件维护是指软件系统交付使用以后，为了改正错误或满足新的需求而修改软件的过程。按照不同的维护目的，维护工作可分成4类。

- 完善性维护（*Perfective Maintenance*）

扩充原有系统的功能，提高原有系统的性能，满足用户的实际需要。

- 纠错性维护（*Corrective Maintenance*）

对在测试阶段未能发现的，在软件投入使用后才逐渐暴露出来的错误的测试、诊断、定位、纠错以及验证、修改的回归测试过程。

- 适应性维护（*Adaptive Maintenance*）

  要使运行的软件能适应运行环境的变动而修改软件的过程。

- 预防性维护（*Preventive Maintenance*）

  为了进一步改善软件的可靠性和易维护性，或者为将来的维护奠定更好的基础而对软件进行修改。