软件工程的基本概念

软件工程：是以工程化思想进行软件开发，以生产高质量和高效率的软件。

核心观点：把软件看作是一个工程产品。

Fritz Bauer的定义：软件工程是为了经济地获得能够在实际机器上有效运行的可靠软件而建立和使用的一系列完善的工程化原则。

软件工程基本原理：推迟实现的观点、逐步求精的观点、分解与抽象的观点、信息隐蔽观点、质量保证观点

软件过程是为了开发出软件产品，或者是为了完成软件工程项目而需要完成的有关软件工程的活动。通常使用生命周期模型简洁地描述软件过程

软件过程模型:就是把软件生命周期中各项开发活动的流程用一个合理的框架(开发模型)来规范描述。

瀑布模型

瀑布模型提供了软件开发的基本框架。

瀑布模型将软件生命周期划分为软件计划、需求分析和定义、软件设计、软件实现、软件测试、软件运行和维护这**6个阶段**，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而得名

它是一个软件开发架构，开发过程是通过一系列阶段顺序展开的。

每个阶段都会产生循环反馈

各个阶段产生的文档是维护软件产品时必不可少的，没有文档的软件几乎是不可能维护的。

瀑布模型特点:顺序性和依赖性、推迟实现、质量保证的观点、是一种线性模型、强调文档的作用。

面向对象过程模型

面向对象是一种的程序设计方法，或者说它是一种程序设计范型。

基本思想是使用对象，类，继承，封装，消息等基本概念来进行程序设计。

面向对象的**特征**：对象惟一性；分类性；继承性；多态性（多形性）。

面向对象的**要素**：

> 抽象：强调实体的本质、内在的属性，忽略一些无关紧要的属性。类实现了对象的数据（即状态）和行为的抽象，是对象的共性的抽象。
> 封装性：指所有软件部件内部都有明确的范围以及清楚的外部边界。
> 共享性

软件系统的开发方式

自行开发、联合开发、外包、购买

可行性研究

可行性研究的任务:用最小的代价在尽可能短的时间内确定问题是否能够解决。可行性研究的目的不是解决问题，而是确定问题是否值得解决。

经济可行性、技术可行性、操作可行性研究系统是否可行

可行性研究的输入是系统的一个框架描述和高层逻辑模型

输出是一份需求开发评价报告，对需求工程和系统开发是否值得做的具体建议和意见。
三个问题:

> 系统是否符合机构的总体要求?
> 系统是否可在现有技术条件、预算和时间限制内完成?
> 系统能否把已存在的其他系统集成?

设定优先级
设定的基本原则:时间投入、成本投入和获得效益(包括无形的社会性效益)是最主要的三个考核指标。
设定优先级的**因素**

> 目标系统能降低成本？何时、何地、如何降低？花费代价怎么样
> 目标系统能增加收入？何时、何地、如何增加？花费代价怎么样
> 目标系统能产生更多信息或更多的结果吗?
> 目标系统能为客户和企业自己提供更好的服务?
> 目标系统能在合理的时间内完成吗？目标系统能为企业提供多长的服务期限?
> 具备必要的人力、财力、技术、管理和资金资源吗?
> **注意事项**
> 系统分析员尽量采用有形的方式评价系统
> 无形成本/效益是影响设定级别的重要因素
> 设定优先级是一个综合权衡过程，没有绝对的、只有相对的。

结构化分析概述
结构化分析是一种适用于大型数据处理系统的、面向数据流的需求分析方法。
结构化分析方法是一种**传统的系统建模技术**，其过程是创建描述信息内容和**数据流的模型**，依据**功能**和**行为**对系统进行划分，并描述必须建立的系统要素。

> 结构化分析将系统**自顶向下逐层分解**，达到表达系统的目的，它采用**一组过程模型图形化地描述一个系统**的**逻辑模型**。
> 结构化需求分析**指导性原则**：理解问题、开发模型、描述需求、建立系统模型、确定需求优先级、验证需求
> 结构化分析方法是一种建模技术
> 基于计算机的系统是数据流和一系列的转换构成的
> 在模型的核心是**数据词典**，它描述了所有的在目标系统中使用的和生成的数据对象。围绕着这个核心的有三种图： **ERD、DFD、STD**

结构化分析模型
分析模型的**目的**是为基于计算机系统提供必须的信息、功能和行为域的说明。
模型是对系统某个方面的抽象，抛弃了具体细节，对系统中最突出的特征作简化。

分析模型的所有元素都可以直接映射到**设计模型**

结构化分析模型的组成

> 数据建模和对象描述
> 功能建模和数据流图
> 基本加工逻辑说明
> 行为建模（时序图）
> 数据词典

面向数据流的建模

面向数据流的建模是结构化需求分析方法之一
采用自顶向下逐层分解，描绘满足用户要求的软件模型（数据模型、功能模型、行为模型）
表示：

> 数据流图：描述系统处理过程
> 数据字典：模型中的数据信息集合
> 状态转换图：描述系统对内部或外部事件响应的行为模型

软件设计概述

软件设计阶段的基本目标是构造系统“**怎么做**”的模型描述。
“设计先于编码” 是软件工程“推迟实现”**基本原则。**
软件系统设计是把软件需求“变换”为用于构造软件的蓝图。
“输入”是需求分析各种模型元素
“输出”是软件设计模型和表示
软件设计的目标是对将要实现的软件系统的体系结构、系统的数据、系统模块间的接口，以及所采用的算法给出详尽的描述。

软件设计概念

总体设计，或称概要设计，或软件结构设计，或高层设计

> 分析需求规格说明
> 模块划分，形成具有预定功能的模块组成结构
> 表示出模块间的控制关系
> 给出模块之间的接口

软件详细设计，也称(模块),过程设计，或低层设计。

> 设计模块细节
> 确定模块所需的算法和数据结构等
> 编写所有代码

设计测试和复审

软件设计过程
软件结构设计过程
含**系统设计**和**结构设计**。系统设计确定系统的具体实现对象，结构设计确定软件的体系结构。**软件结构设计步骤**:

1. 设计供选择的方案
2. 选取合理的方案
3. 推荐最佳方案
4. 功能分解和设计软件结构
5. 数据库设计
6. 制定软件设计测试计划
7. 编制设计文档
8. 审查和复审

软件模块化设计

> **模块是一个独立命名的，拥有明确定义的输入、输出和特性的程序实体**。
> 把一个大型软件系统的全部功能，按照一定的原则合理地划分为若干个模块，每个模块完成一个特定子功能，所有的这些模块以某种结构形式组成一个整体，这就是软件的模块化设计
> **软件模块化设计可以简化软件的设计和实现，提高软件的可理解性和可测试性，并使软件更容易得到维护。**
> 分解、抽象、逐步求精、信息隐蔽和模块独立性，是软件模块化设计的指导思想。

软件模块化

> 软件系统的模块化是指整个软件被划分成若干单独命名和独立访问部分，称之为模块。每个模块完成一个子功能，把全部模块集成起来构成一个整体，可以完成指定的功能，满足用户的需求。
> 把问题 / 子问题的分解与软件开发中的系统 / 子系统或系统 / 模块对应起来，就能够把一个大而复杂的软件系统划分成易于理解的比较单纯的模块结构。
> 模块化可以使一个复杂的大型程序能被人的智力限制所管理，是软件应该具备的最重要的属性。
> 事实上，每个程序都相应地有一个最适当的模块数目，可使软件系统的开发成本最小。
> **模块划分的目的**: ①进行功能分解，把复杂的大的功能划分成简单的小的子功能，尽量降低每个模块的成本。②尽量使每个模块间的接口不能太多，太多会使接口成本增加。兼顾二者可取得最佳的划分状态，确保软件总成本最低。

模块的独立性

> 模块的独立性是模块化、抽象、信息隐蔽等概念的直接结果，也是判断模块化结构是否合理的标准。

> 模块独立性是指开发具有独立功能而和其他模块没有过多关联的模块。
> 模块独立性两大优点：

- 独立的模块由于分解了功能，简化了接口，使得软件比较容易开发；
- 独立的模块比较容易测试和维护。

> 模块独立性由两个定性标准度量：

- 耦合是模块之间的互相连接的紧密程度的度量
- 内聚是模块功能强度(一个模块内部各个元素彼此结合的紧密程度)的度量。

> 模块独立性愈高，则块内联系越强，块间联系越弱。

概要设计

> 概要设计也称**总体设计**，确定软件的结构以及各组成成分(子系统或模块)之间的相互关系。
> 概要设计的主要任务是：

- 将系统划分成模块；
- 决定每个模块的功能；
- 决定模块的调用关系；
- 决定模块的界面，即模块间传递的数据。

> 概要设计的实现方式：通过数据流图来确定系统的结构图，并且对这些结构图进行分析和细化。
> 在概要设计阶段，结构化设计主要采用**面向数据流的设计方法**。

详细设计

> 详细设计就是在概要设计的基础上决定如何具体实现各模块的内部细节，直到对系统中的每个模块给出足够详细的过程描述。
> **在编码实现阶段可以完全按照详细设计的细节过程来映射到代码，最终实现整个系统。**
> 一般使用结构化程序设计工具来描述

面向对象概念

> 抽象性：强调实体的本质、内在的属性，忽略一些无关紧要的属性。类实现了对象的数据（即状态）和行为的抽象，是对象的共性的抽象。
> 封装性：指所有软件部件内部都有明确的范围以及清楚的外部边界。每个软件部件都有友好的界面接口，软件部件的内部实现与外部可访问性分离。
> 共享性：面向对象技术在不同级别上促进了共享。

面向对象模型

- 面向对象模型的三种主流形式——按照产生顺序排

> 对象模型：定义“做什么”的实体。它可表达系统的数据或对数据的处理，**它是数据流和语义数据模型的结合**。
> 动态模型：规定在何种状态下，接受什么事件的触发而“做什么”。它表示瞬间的、行为化的系统“控制”性质，并规定了对象模型中对象的合法变化序列。
> 功能模型：指明系统应该“做什么”。它直接反映用户对目标系统的需求

对象模型

> 对象模型表示静态的、结构化系统的“数据”性质。描述的是系统一种**静态结构**，是对模拟客观世界实体的对象，以及对象彼此间的关系的**映射**。
> 对象模型的基本组成形式 = **类**(包括其属性和行为) + **对象**(类的实例) + **类或对象之间关系**。
> 类名是**一类对象的抽象命名**，其命名是否恰当对系统的可理解性影响相当大。
> 对象模型还必须表示类/对象之间的结构关系。类/对象之间的关系一般可概括为**关联、归纳/继承（泛化）、组合（聚集）三类。**

动态模型

> **动态模型表示瞬间的、行为化的系统“控制”性质**，它规定了对象模型中对象的合法变化序列。也可以说，动态模型是基于共享而相互联系的一组状态集合。
> 对象运行周期中的阶段就是对象的状态,对象状态是对对象属性的一种抽象。
> 对象之间相互触发/作用的行为（称为事件），引起一系列的状态变化。
> 事件是引起**对象状态转换的控制信息。事件没有持续时间,是瞬间完成的。**
> 对象对事件的响应，取决于接受该触发的对象当时所处的状态，其响应包括改变自己的状态，或者是形成一个新的触发行为（事件）。
> **动态模型描绘了对象的状态**，触发状态转换的事件，以及对象行为（对事件的响应）。

软件测试的定义
用户的角度：普遍希望通过软件测试暴露软件中隐藏的错误和缺陷，以考虑是否可接受该产品。
软件开发者的角度：希望测试成为表明软件产品中不存在错误的过程，验证该软件已正确地实现了用户的要求，确立人们对软件质量的信心。
软件测试的定义
(1) 测试是程序的执行过程，目的在于发现错误；
(2) 一个好的测试用例在于能发现至今未发现的错误；
(3) 一个成功的测试是发现了至今未发现的错误的测试。

> 以最少的时间和人力，系统地找出软件中潜在的各种错误和缺陷。
> 测试附带收获：能够证明软件功能和性能与需求说明相符合

> 测试结果数据为可靠性分析提供了依据。
> 测试不能表明软件中不存在错误，只能说明软件中存在错误。

软件测试准则

> 软件测试三个最佳实践**:尽早测试、连续测试、自动化测试**
> **所有的测试都应该能追溯到用户需求。**
> **应该远在测试开始之前就制定出测试计划。**
> 严格执行测试计划，排除测试的随意性。
> 测试用例应由**测试输入数据**和对应的**预期输出结果**这两部分组成。
> 程序员应避免检查自己的程序。**为了达到最佳的测试效果，应该由独立的第三方从事测试工作**
> 在设计测试用例时，应包括合理的输入条件和不合理的输入条件。
> **穷举错误是不可能的，因此，测试只能证明程序中有错误，而不能证明程序中没有错误。**

白盒测试
测试规划:根据程序的内部结构，如语句的控制结构，模块间的控制结构以及内部数据结构等进行测试
优点:能够对程序内部的特定部分进行覆盖测试
缺点:无法检验程序的外部特性、无法对未实现规格说明的程序内部欠缺部分进行测试
方法举例:逻辑覆盖：语句覆盖、判定覆盖、条件覆盖、判定-条件覆盖
、基本路径覆盖、循环路径测试

黑盒测试
测试规划：根据用户的规格说明，即针对命令、信息、报表等用户界面及体现它们的输入数据与输出数据之间的对应关系，特别是针对功能进行测试
优点:能站在用户的立场上进行测试
缺点:不能测试程序内部特定部分，如果规格说明有误，则无法发现
方法举例:等价类划分、边界值分析、因果图法测试

黑盒测试技术

> 黑盒测试是根据程序组件的规格说明测试软件功能的方法，所以也称为功能测试。被测对象作为一个黑盒子，它的功能行为只能通过研究其输入和输出来确定，又称为输入/输出接口测试
> **设计黑盒测试用例的原则**
> 对于有输入的所有功能，既要用有效的输入来测试，也要用无效的输入来测试。
> 经过菜单调用的所有功能都应该被测试，包括通过同一个菜单调用的组合功能也要测试。
> 设计的测试用例数量，能够达到合理测试所需的“最少”（减少测试成本）。
> 设计的测试用例，不仅能够告知有没有错误，而且能够告知某些类型的错误存在或不存在（提高测试效率）

软件项目管理概述-软件项目管理

项目管理的目的

> 按照系统的观点，使用现代项目管理的科学理念和方法进行控制，以较小的代价，获得较大的收益。
> 软件项目管理的目的是为了按照预定的进度、费用等要求，成功地组织与实施软件的工程化生产，完成软件（产品）的开发和维护任务。

项目管理的特点

> 项目管理是一项比较复杂的工作
> 项目管理具有创造性
> 项目管理的对象是项目或被当作项目来处理的作业
> 项目负责人在项目管理中起着非常重要的作用
> 项目管理需要集权领导和建立专门的项目组织
> 项目管理的方法、工具和手段具有先进性、开放性

软件项目管理的任务

> **软件开发管理的任务**：制定文档；预计需要的资源；费用估算；安排工作任务和计划；定期做评审；质量保证管理；开发总结报告；处理意外情况等。
> **软件测试管理的任务**:制定测试计划；测试分析并报告；编制用户手册。
> **项目运行管理的任务**:人员的组织与管理；设备和资料管理；财政预算与支出管理；作业时间管理。
> **项目后评价管理的任务**:技术水平与先进性评价；经济与社会效益分析；系统的内在质量评价；系统的推广使用价值评价；系统的不足之处与改进意见等。