UML：OMT (James Rumbaugh) 、OOSE (Ivar Jacobson) 、Booch (Grady Booch)

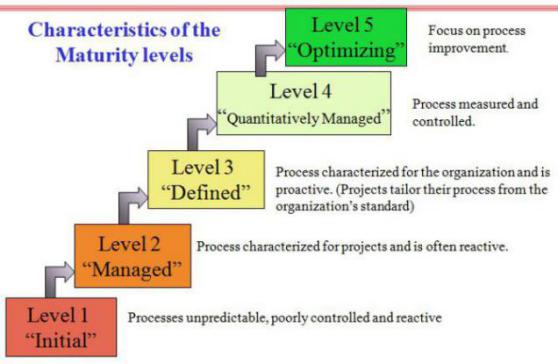
**软件危机**：软件的可靠性没有保障、维护费用不断上升、进度无法预测、成本增长无法控制、程序员无限度增加等，形成软件开发局面失控的状态。根源在于软件的复杂性与规模。

软件工程：应用计算机科学、数学及管理科学等原理，借鉴传统工程的原则和方法，来创建软件，从而达到提高质量、降低成本的目的。

**软件工程**是将系统的、规范的、可度量的途径用于软件的的定义，是指导计算机软件开发和维护的一门工程学科，包括技术和管理两个方面。

传统软件工程把**软件的生命周期**定义为6个阶段：问题定义(关于系统目标的定义、规模、经济效益分析系统的高层抽象逻辑模型)；需求分析(系统的逻辑模型：数据字典、数据流图、算法描述)；系统设计(可能的解法：系统流程图、系统层次图和结构图；编码规格说明、HIPO图、PDL)；编码(源程序规范、源程序、单元测试结果分析)；测试(集成和系统测试方案、测试结果分析)；运行与维护

**CMMI (Capability Maturity Model Integration) ——软件过程模型示例**

[](http://upload.wikimedia.org/wikipedia/commons/6/64/Characteristics_of_the_Maturity_levels.jpg)

**UML**:unified modeling language；**OMT**:object modeling technology

**OCL**（Object Constraint Language)**;OOAD**（Object Oriented Analysis And Design)；**FRIEND:**First Responder Interactive Emergency Navigational Database

**ODD**：The Object Design Document **;LOC**：length of code (代码行)

**COCOMO**：constructive cost model ;Application programmer’s interface (**API)**

**ISO ：** International Standard Organization **;OSI ：** Open System Interconnection

**OOSE：**object-oriented software engineering   **;RAD**：Requirement Analysis Document

**软件过程：**生产一个最终满足需求且达到工程目标的软件产品所需要的步骤（目标：standardization, predictability, productivity, high product quality, ability to plan time and budget requirements）软件过程评估（Process Assessment） CMMI、软件过程裁剪（Adaptable Model）（包含：沟通、策划、建模、构建、部署。）

**惯例过程模型**是在软件开发过程中的经验积累，它确定了软件开发全部过程、活动和任务的结构框架。

框架活动 vs 软件生命周期：沟通：需求获取策划：计划任务、估计风险、资源需求、进度计划 建模：分析、设计 构建：编码、测试 部署；交付、评测、反馈

**瀑布模型**优点：强调开发的阶段性；强调早期计划及需求调查；强调产品测试。

 缺点：瀑布模型过于依赖早期进行的唯一一次需求调查，不能适应需求的变化

瀑布模型是单一流程，开发中的经验教训不能反馈应用于本产品的过程。

**快速原型模型**优点：帮助导引出高质量的产品要求，尽早进行风险管理和配置管理，心理上，开发人员早日见到产品的雏型，是一种鼓舞。 使用户可以在新的一批功能开发测试后，立即参加验证，以便提供非常有价值的反馈

缺点：如果缺乏严格的过程管理的话，这个生命周期模型很可能退化为一种原始的无计划的“试－错－改”的循环模式。心理上，可能产生一种影响尽最大努力的想法，认为虽然不能完成全部功能，但还是造出了一个有部分功能的产品。 如果不加控制地让用户接触开发中尚未测试稳定的功能，可能对开发人员及用户都产生负面的影响。

**螺旋模型**(引入了“**风险驱动**”的思想，适用于大规模的内部开发项目)

制定计划 风险分析 工程实施 客户评估**优点**：强调严格的全过程风险管理。强调各开发阶段的质量。提供机会检讨项目是否有价值继续下去。 **缺点**：必须引入非常严格的风险识别，风险分析和风险控制，这对风险管理的技能水平提出了很高的要求。也需要人员，资金和时间的大量投入。

**基于构件的开发模型优点**：构件组装模型导致了软件的复用，提高了软件开发的效率，面向对象技术是软件工程的构件组装模型的基础；构件可由一方定义其规格说明，被另一方实现，然后供给第三方使用；构件组装模型允许多个项目同时开发，降低了费用，提高了可维护性；可实现分步提交软件产品。**缺点**：可重用性和软件高效性不易协调。缺乏通用的组装结构标准，而自定义的组装结构标准引入较大的风险。需要精干的有经验的分析和开发人员，一般的开发人员插不上手。客户的满意度低。

**RUP (Unified Process)**：RUP的时间轴被分解为**四个顺序的阶段**：

初始阶段(Inception)为系统建立业务案例并确定项目的边界。 细化阶段(Elaboration)分析问题领域，建立健全的体系结构基础，编制项目计划，淘汰项目中最高风险的元素。

构造阶段(Construction)所有剩余的构件和应用程序功能被开发并集成为产品，所有的功能被详细测试。 交付阶段(Transition)确保软件对最终用户是可用的。**优点**：

提高了团队生产力，在迭代的开发过程、需求管理、基于组件的体系结构、可视化软件建模、验证软件质量及控制软件变更等方面，针对所有关键的开发活动为每个开发成员提供了必要的准则、模板和工具指导，并确保全体成员共享相同的知识基础。它建立了简洁和清晰的过程结构，为开发过程提供较大的通用性。 **缺点**：RUP在理论上，是比较理想的，但在实际应用上，还需要更多的工具的支持和普及推广工作。

* **高内聚低耦合(**high cohesion & low coupling)模块的内聚性最高的是功能内聚

程序模块优化的启发式规则：通过模块的合并和分解，降低模块的耦合度，提高模块的内聚性  C）将模块的作用范围限制在模块的控制范围之内 D）降低模块之间接口的复杂性，避免“病态连接”

**软件的可行性研究**：Ａ）法律可行性 Ｂ）技术可行性 Ｃ）经济可行性

**面向对象的分析方法**主要是建立：对象模型、动态模型、功能模型

**程序设计语言的选择**：将来能否占主导地位   B）可重用性  C）有良好的开发工具支持

**结构化分析方法**：数据流图 判定表和判定树 系统流程图 用例图不是

**软件危机**表现:  
• 软件生产效率低下,软件成本高: 软件开发成本和进度的估计常常很不准确。  
• 软件产品质量差: 用户对“已完成的”软件系统不满意的现象经常发生。  
• 软件产品难以(不能)维护: 软件通常没有适当的文档资料

软件工程的发展阶段:传统SE、面向对象、软件过程工程、基于构件的软件开发

面向对象：从事物的组成部件及每个部件的属性、功能来认识事物。

用例图用于描述系统与外部系统及用户之间的交互

数据字典的内容：处理逻辑 数据结构 外部实体

**简述面向对象设计过程中软件重用的定义、范围和主要技术。**

重用是指同一事物不作修改或稍加改动就多次重复使用。（2分） ①软件重用可以分为3个层次：知识重用、方法和标准的重用和软件成分的重用。 ②软件成分的重用可以进一步分为3个级别：代码重用、设计结果重用、分析结果重用。 类构件是比较理想的软件重用技术，其重用方式有：实例重用、继承重用和多态重用（3分）

**软件架构**包括构件、连接件和约束三种基本元素，以及端口和角色两种元素。构件作为一个封装的实体，仅通过其接口结构与外部环境交互，而构件的接口由一组端口组成，每个端口表示了构件和和外部环境的交互点。连接件的接口由一组角色组成，每个角色定义了相关交互的参与者。

repository仓库——数据库管理系统、编译器、软件开发环境;cs结构——web浏览器系统;peer to peer对等——实时聊天系统

**软件架构建模方法**：基于非规范的图形表示；基于UML；基于形式化；基于UML形式化；基于文本

**软件架构模型意义**

软件架构建模是对架构设计决策的具象化和文档化。

软件架构建模的意义在于它能够将软件架构某些关键或关注的方面剥离出来，使用统一的图形、文档和数据进行描述，达到直观、便捷地理解、分析和交流。

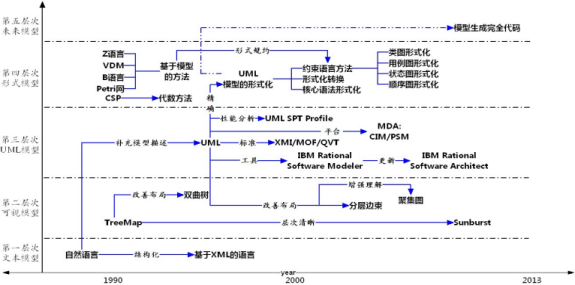
**UML不同视角描述软件架构**：逻辑视图可以采用UML用例图来实现；开发视图使用UML的类图、对象图和组件图来表示模块，用包来表示子系统，利用连接表示模块或子系统之间的关联；过程视图可以采用UML的状态图、顺序图和活动图来实现

；物理视图定义了功能单元的分布状况，描述用于执行用例和保存数据的业务地点，可以使用UML的配置图来实现**UML对软件架构建模的优点**：通用的模型表示法、统一的标准，便于理解和交流；支持多视图结构，能够从不同角度来刻画软件架构，可以有效地用于分析、设计和实现过程；有效利用模型操作工具(支持UML 的工具集)，可缩短开发周期， 提高开发效率; 统一的交叉引用( cross-referencing ) 模型信息的方法， 有利于维护开发元素的可处理性， 避免错误的产生。**缺点：**UML 虽然可以对软件架构进行较好的描述，但它只是针对特定的面向对象的架构，比如对架构缺少形式化的支持，使用UML建模存在着一些问题：对架构的构造性建模能力不强，还缺乏对架构风格和显式连接件的直接支持；虽然UML使用交互图、状态图和活动图描述系统行为，但语义的精确性不足;使用UML多视图建模产生信息冗余和不一致；对架构的建模只能到达非形式化的层次，不能保证软件开发过程的可靠性，不能充分表现软件架构的本质。

**用UML对架构进行建模的三种途径**：将UML看作是一种软件架构描述语言直接对架构建模；通过扩展机制约束UML的元模型以支持软件架构建模的需要；对UML的元模型进行扩充。

**软件架构风格**（software architecture style）是描述某一特定应用领域中系统组织方式的惯用模式，作为“可复用的组织模式和习语”，为设计人员的交流提供了公共的术语空间，促进了设计复用与代码复用。

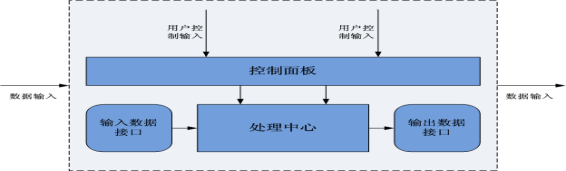
**使用架构风格的优势：**它极大地促进了设计的重用性和代码的重用性，并且使得系统的组织结构易被理解；即使没有给出具体实现细节，依然可以通过“客户/服务器”架构风格大致推测出系统的组成结构和工作方式；使用标准的架构风格可较好地支持系统内部的互操作性以及针对特定风格的分析



**管道过滤器风格**(传统编译器)**优点**：

（1）由于每个组件行为不受其他组件的影响，整个系统的行为易于理解。（2）管道-过滤器风格支持功能模块的复用（3）基于管道-过滤器风格的系统具有较强的可维护性和可扩展性。 （4）支持一些特定的分析，如吞吐量计算和死锁检测等。（5）管道-过滤器风格具有并发性；**缺点**：（1）管道-过滤器风格往往导致系统处理过程的成批操作。

（2）根据实际设计的需要，设计者也需要对数据传输进行特定的处理（如为了防止数据泄漏而采取加密等手段，或者使用了底层公共命名），导致过滤器必须对输入、输出管道中的数据流进行解析或反解析，增加了过滤器具体实现的复杂性。（3）交互式处理能力弱：管道-过滤器模型适于数据流的处理和变换，不适合为与用户交互频繁的系统建模。下图为改进



**Main program/subroutines组件为主程序和子程序**，连接件为调用-返回机制，拓扑结构为层次化结构。**优点：** （1）具有很高的数据访问效率，因为计算共享同一个存储区。（2）不同的计算功能被划分在不同的模块中**。缺点**：该方案在处理变更的能力上有许多严重的缺陷：（1）对数据存储格式的变化将会影响几乎所有的模块。（2）对处理流程的改变与系统功能的增强也很难适应，依赖于控制模块内部的调用次序。（3）这种分解方案难以支持有效的复用。

**OO风格优点**：1对象隐藏了其实现细节，所以可以在不影响其它对象的情况下改变对象的实现，不仅使得对象的使用变得简单、方便，而且具有很高的安全性和可靠性。2设计者可将一些数据存取操作的问题分解成一些交互的代理程序的集合。**缺点**：

当一个对象和其它对象通过过程调用等方式进行交互时，必须知道其它对象的标识。无论何时改变对象的标识，都必须修改所有显式调用它的其它对象，并消除由此带来的一些副作用。

**层次化风格(TCP/IP分层通信协议)优点**:（1）支持基于可增加抽象层的设计，允许将一个复杂问题分解成一个增量步骤序列的实现。（2）支持扩展。每一层的改变最多只影响相邻层。3）支持重用。只要给相邻层提供相同的接口，它允许系统中同一层的不同实现相互交换使用。**缺点：**（1）不是所有系统都容易用这种模式来构建；（2）定义一个合适的抽象层次可能会非常困难，特别是对于标准化的层次模型。

**事件驱动风格**（ Win32 GUI程序）优点:（1）事件声明者不需要知道哪些组件会影响事件，组件之间关联较弱。2）提高软件复用能力。只要在系统事件中注册组件的过程，就可以将该组件集成到系统中。（3）系统便于升级。只要组件名和事件中所注册的过程名保持不变，原有组件就可以被新组件替代。缺点:

（1）组件放弃了对计算的控制权，完全由系统来决定。（2）存在数据交换问题。（3）该风格中，正确性验证成为一个问题。

**黑板系统风格组成**：（1）知识源（2）黑板数据结构（3）控制器 **优点：**（1）便于多客户共享大量数据，他们不关心数据何时有的、谁提供的、怎样提供的。（2）既便于添加新的作为知识源代理的应用程序，也便于扩展共享的黑板数据结构。（3）知识源可重用。4）支持容错性和健壮性。**缺点：**（1）不同的知识源代理对于共享数据结构要达成一致，而且，这也造成对黑板数据结构的修改较为困难——要考虑到各个代理的调用。（2）需要一定的同步/加锁机制保证数据结构的整性和一致性，增大了系统复杂度。

**面向服务SOA架构**风格：服务请求者、提供者、注册中心、发布、发现、绑定调用

Orthogonal softwara:**正交软件架构**：组织层、线索的组件

**软件质量可以定义**为：在一定程度上应用有效的软件过程，创造有用的产品，为生产者和使用者提供明显的价值。**架构质量评估：**（评估系统体系结构和重要属性）

**重要属性：可修改性、可用性、性能、可测试性、易用性、安全性架构质量**

**架构质量评估原因：**软件体系结构的好坏关系到软件产品的好坏，软件产品的好坏关系到软件公司的发展。通过评估能了解系统的体系结构和重要属性(质量属性)，能够屏蔽风险，带来诸多收益 。

**基于调查问卷或检查表的评估方式**系列可以应用到各种体系结构评估的相关问题，缺：由于评估的结果很大程度上来自评估人员的主观推断，因此不同的评估人员可能会产生不同甚至截然相反的结果，而且评估人员对领域的熟悉程度、是否具有丰富的相关经验也成为评估结果是否正确的重要因素。

**基于场景评估方式：**由SEI首先提出并应用在体系结构权衡分析方法（ATAM）和软件体系结构分析方法（SAAM）中。这种分析软件体系结构对场景也就是对系统的使用或修改活动的支持程度，从而判断该体系结构对这一场景所代表的质量需求的满足程度。

**优点：** 这一评估方式(基于场景)考虑到了包括系统的开发人员、维护人员、最终用户、管理人员、测试人员等在内的所有与系统相关的人员对质量的要求;基于场景的评估方式涉及到的基本活动包括确定应用领域的功能和软件体系结构的结构之间的映射，设计用于体现待评估质量属性的场景以及分析软件体系结构对场景的支持程度。**缺点**：不同的应用系统对同一质量属性的理解可能不同，因此基于场景的评估方式是特定于领域的。这一评估方式的实施者一方面需要有丰富的领域知识以对某以质量需求设计出合理的场景，另一方面，必须对待评估的软件体系结构有一定的了解以准确判断它是否支持场景描述的一系列活动。

**基于度量的评估：**度量是指为软件产品的某一属性所赋予的数值，如代码行数、方法调用层数、构件个数等。

**ATAM: Architecture Tradeoff Analysis Method（体系结构权衡分析方法）**

**敏感点影响一个质量属性、权衡点影响多个质量属性**

**敏感点（**Sensitivity point）是一个或多个构件的特征；可以使设计师搞清楚实现质量目标时应该注意什么。**权衡点Tradeoff point）**是影响多个质量属性的特征，是多个质量属性的敏感点，权衡点需要进行权衡。步骤：

1）陈述，包括通过它进行的信息交流

2）调查与分析，包括对照体系结构方法评估关键质量属性需求：在这一步骤中，体系结构风险、非风险、敏感点和权衡点被识别出来3）测试，包括对照所有相关人员的需求检验最新结果4）形成报告，包括陈述ATAM的结果（结果的表述：包括方法、场景、特定属性的问题、效用树、有风险决策、无风险决策、敏感点和权衡点

**软件项目管理：**项目经理和项目组织的努力，运用系统理论的方法对项目及其资源进行计划、组织、协调、控制，旨在实现项目的特定目标的管理方法体系**，基本内容**：项目定义、项目计划、项目执行、项目控制、项目结束**关注点：**人员、产品（范围包括：目标和范围、可选的解决方案、技术或管理的约束等 ）**过程：**软件过程框架、**项目：**科学的方法及工具科学的方法及工具，有计划，可控制

管理过程：. 软件项目管理过程软件项目启动（——确定产品的范围）建立项目组织（组织不同类型的项目组成员共同协作完成软件项目）、项目计划（估算、进度安排、风险管理）跟踪与控制（跟踪：其核心在于反映项目变化、提供相关信息的报告，控制：并对项目的未来走向进行预测，再此基础上进行项目的各种调整）

**确定产品范围A work breakdown structure （WBS）**，书写格式：树形结构；目录结构（产品分解：将一个复杂问题分解成若干个小问题过程分解：根据软件开发过程中的活动(分析、设计、编码、测试等)进行分解）**工作包**（**Work Packages）**

WBS结构的叶子节点，即任务分解的最小单位，不可再划分的子任务（不在wbs中的则不在项目中）

**成本估算的常用方法：**

1）基于已经完成的类似项目进行估算，这是一种常用的也是有效的估算方法2）基于规模的估算：基于LOC；基于FP的估算3）基于经验估算模型的估算： COCOMO模型

**基于规模的估算：**项目计划人员为每个功能或信息域确定一个估算值的范围：乐观值opt，可能值m，悲观值pess；计算各个功能或信息域的规模的期望值S=(Sopt +4Sm + Spess)/6；应用历史生产率数据估算项目的成本

**功能点估算：FP**基于软件信息域的特征(可直接测量)和软件复杂性进行规模估算

**功能点度量方法步骤：**计算信息域特征的值CT；计算复杂度调整值；计算功能点FP

**- 项目进度管理1.**Defining Task Sets2.Sequencing Activities3.Scheduling Steps4.Using Gantt charts (甘特图) for scheduling5.Schedule Tracking

**构建项目任务网络（Project network diagrams）**

**Arrow Diagramming Method (ADM)（AOA）**

**Precedence Diagramming Method (PDM)**

**任务网络的优点:**Show precedence (优先关系) well;Ability to identify critical path （关键路径分析）;Ability to perform “what if” analysis

**任务网络的缺点：**Default model assumes resources are unlimited;we need to incorporate Resource Dependencies ourselves when determining the “real” Critical Path;Difficult to follow on large projects

**- 风险管理**（风险;在给定情况下和特定时间内，那些可能发生的结果与预期结果之间的差异，差异越大，风险越大）（项目、技术、商业风险）

**主动风险管理**的过程：风险识别（已知的和可预测的风险）、风险分析（风险预测：

评价每种风险发生的可能性或概率以及当该风险发生时所导致的后果、风险预测任务：建立一个尺度，以反映风险发生的可能性描述风险的后果、估算风险对项目及产品的影响、标注风险预测的整体精确度以免产生误解 建立风险表：综合考虑风险发生的概率和风险所产生的影响，对风险表排序 风险评估，定义中截线）、风险规划、风险监控

**风险规划(RMMM)**的主要目的就是辅助项目团队制定处理风险的策略（风险管理就是识别评估风险，建立、选择、管理和解决风险的可选方案和组织方法）

**有效的风险规划必须考虑：**风险避免Risk Mitigation）风险监测 Risk Monitoring）风险管理及应急计划（Risk Management Plan）对于每个中截线以上的风险，都应制定RMMM

**Open closed liskov subsititution dependency inversion interface segration**

**软件度量：**度量的作用是为了有效地采用定量的方式来进行管理Measure(noun 测度)：为产品或过程的某些属性的程度、数量、维数、容量或大小提供量化的指示。

Metric(度量)：一个系统、构件或过程具有给定属性的量化测量程度

Indicator（指标）：是一个度量或度量的组合，

**度量对象：产品、过程、项目**

**敏捷方法**

**敏捷开发的基本理念：**强调个体和互动比强调过程和工具更好、可运行的软件、

强调与客户合作、强调响应变化

**敏捷开发在实践**中表现为一种迭代、增 量和持续集成的开发方法。迭代反映了项目的开发节奏，是一个多周 期的开发过程。 增量说明了项目的实际进展，整个项目就 是由很多增量构成的。 持续集成反映了集成增量的过程是持续进 行的、

与软件架构出发点同：都是一个权衡的过程、发目的都是为了提高软 件开发效率、提高软件质量、降低软件成 本，将开发团队的价值最大化。

**特点：**轻详设，种子架构+详细架构，把传统的详设分散到整个过程中，提高效率减少风险、团队设计、简单设计（表达方式的简 单化和现实抽象的简单化）

优秀的**敏捷软件架构的设计过程**一般同 时包含**规划式设计和演进式设**计，具体 体现为初始阶段设计和迭代过程中的设 计