瀑布模型：

- 需求分析->规格说明->设计检验->编码测试->综合测试->维护

瀑布模型的特点：

- 阶段间具有顺序性和依赖性
- 推迟实现的观点
- 质量保证的观点

瀑布模型的优点：

①可强迫开发人员采用规范的方法(如结构化技术)

②严格地规定了每个阶段必须提交的文档

③要求每个阶段交出的所有产品都必须经过质量保证小组的仔细验证

瀑布模型的缺点：

瀑布模型几乎完全依赖于书面的规格说明，很可能导致最终开发出的软件产品不能真正满足用户的需要。

结构化生命周期方法（SLC方法）

1、根据需求设计系统

要求在明确用户需求之前，不得进行下一阶段的工作。

其目的是：保证工作质量和以后各阶段开发的正确性，减少系统开发的盲目性。

2、严格按阶段进行

对生命周期的各个阶段雅阁划分，每个阶段有其明确的任务和目标，而各个阶段又可被分为若干工作和步骤。

其目的是：便与计划管理和控制，前阶段工作成果是后阶段工作的依据，基础扎实，不返工。

3、文档标准化和规范化

要求文档采用标准化、规范化、确定的格式和术语以及图形图表。

其目的是：保证通信内容的正确理解，使系统开发人员及用户有共同的语言。

4、分解和综合

将系统划分为相互联系又相对独立的子系统直至模块

其目的是：分解使复杂的系统简单化，便于设计和实施。综合使已实施的子系统成为完整的系统以体现系统的总体功能。

5、强调阶段成果审定和检验

阶段成果需得到用户、管理人员和专家认可。

其目的是：减少系统开发工作中的隐患。

内聚与耦合

内聚标志一个模块内各个元素彼此结合的紧密程度，它是信息隐蔽和局部化概念的自然扩展。内聚是从功能角度来度量模块内的联系，一个好的内聚模块应当恰好做一件事。它描述的是模块内的功能联系。

耦合是软件结构中各模块之间相互连接的一种度量，耦合强弱取决于模块间接口的复杂程度、进入或访问一个模块的点以及通过接口的数据。 程序讲究的是低耦合，高内聚。就是同一个模块内的各个元素之间要高度紧密，但是各个模块之间的相互依存度却要不那么紧密。

耦合可以分为以下几种，它们之间的耦合度由高到低排列如下：
（1） 内容耦合。当一个模块直接修改或操作另一个模块的数据时，或一个模块不通过正常入口而转入另一个模块时，这样的耦合被称为内容耦合。内容耦合是最高程度的耦合，应该避免使用之。
（2） 公共耦合。两个或两个以上的模块共同引用一个全局数据项，这种耦合被称为公共耦合。在具有大量公共耦合的结构中，确定究竟是哪个模块给全局变量赋了一个特定的值是十分困难的。
（3） 外部耦合 。一组模块都访问同一全局简单变量而不是同一全局数据结构，而且不是通过参数表传递该全局变量的信息，则称之为外部耦合。
（4） 控制耦合 。一个模块通过接口向另一个模块传递一个控制信号，接受信号的模块根据信号值而进行适当的动作，这种耦合被称为控制耦合。
（5） 标记耦合 。若一个模块A通过接口向两个模块B和C传递一个公共参数，那么称模块B和C之间存在一个标记耦合。
（6） 数据耦合。模块之间通过参数来传递数据，那么被称为数据耦合。数据耦合是最低的一种耦合形式，系统中一般都存在这种类型的耦合，因为为了完成一些有意义的功能，往往需要将某些模块的输出数据作为另一些模块的输入数据。
（7） 非直接耦合 。两个模块之间没有直接关系，它们之间的联系完全是通过主模块的控制和调用来实现的。
总结：耦合是影响软件复杂程度和设计质量的一个重要因素，在设计上我们应采用以下原则：如果模块间必须存在耦合，就尽量使用数据耦合，少用控制耦合，限制公共耦合的范围，尽量避免使用内容耦合。

CMM/CMMI将软件过程的成熟度分为5个等级,以下是5个等级的基本特征：

(1)初始级(initial)。工作无序，项目进行过程中常放弃当初的计划。管理无章法，缺乏健全的管理制度。开发项目成效不稳定，项目成功主要依靠项目负责人的经验和能力，他一但离去，工作秩序面目全非。
(2)可重复级(Repeatable)。管理制度化，建立了基本的管理制度和规程，管理工作有章可循。 初步实现标准化，开发工作比较好地按标准实施。 变更依法进行，做到

基线化，稳定可跟踪，新项目的计划和管理基于过去的实践经验，具有重复以前成功项目的环境和条件。
(3)已定义级(Defined)。开发过程，包括技术工作和管理工作，均已实现标准化、文档化。建立了完善的培训制度和专家评审制度，全部技术活动和管理活动均可控制，对项目进行中的过程、岗位和职责均有共同的理解 。
(4)已管理级(Managed)。产品和过程已建立了定量的质量目标。开发活动中的生产率和质量是可量度的。已建立过程数据库。已实现项目产品和过程的控制。可预测过程和产品质量趋势，如预测偏差，实现及时纠正。
(5)优化级(Optimizing)。可集中精力改进过程，采用新技术、新方法。拥有防止出现缺陷、识别薄弱环节以及加以改进的手段。可取得过程有效性的统计数据，并可据进行分析，从而得出最佳方法。

OMT方法有四个步骤，分别是分析、系统设计、对象设计和实现。OMT方法的每一个步骤都使用上述三种模型，每一个步骤对这三种模型不断地进行细化和扩充。
分析阶段基于问题和用户需求的描述，建立现实世界的模型。分析阶段的产物有问题描述、对象模型、动态模型和功能模型。
系统设计阶段结合问题域的知识和目标系统的体系结构(求解域)，将目标系统分解为子系统。
对象设计阶段基于分析模型和求解域中的体系结构等添加的实现细节，完成系统设计。主要产物包括细化的对象模型、细化的动态模型和细化的功能模型。
实现阶段将设计转换为特定的编程语言或硬件，同时保持可追踪性、灵活性和可扩展性。

白盒测试：
（1）语句覆盖
使程序中的每个可执行语句都能执行一次的测试用例
（2）判定覆盖（分支覆盖）
对于判断语句，在设计用例的时候，要设计判断语句结果为True和False的两种情况
（3）条件覆盖
设计用例时针对判断语句里面每个条件表达式true 和 false各取值一次，不考判断语句的计算结果
（4）判定条件覆盖（分支条件覆盖）
设计测试用例时，使得判断语句中每个条件表达式的所有可能结果至少出现一次，每个判断语句本身所有可能结果也至少出现一次。
（5）条件组合覆盖
设计测试用例时，使得每个判断语句中条件结果的所有可能组合至少出现一次

（6）路径覆盖

设计测试用例时，覆盖程序中所有可能的执行路径

优点：这种覆盖方法可以对程序进行彻底的测试用例覆盖，比前面讲的五种方法覆盖度都要高。

缺点：于路径覆盖需要对所有可能的路径进行测试（包括循环、条件组合、分支选择等），那么需要设计大量、复杂的测试用例，使得工作量呈指数级增长。路径覆盖虽然是一种比较强的覆盖，但未必考虑判断语句中条件表达式结果的组合，并不能代替条件覆盖和条件组合覆盖。

需求说明书
数据描述
功能需求
性能需求
运行需求
其它需求

概要设计说明书
总体设计
接口设计
数据结构设计
运行设计
出错处理设计
安全保密设计
维护设计

详细设计说明书
总体设计
程序描述

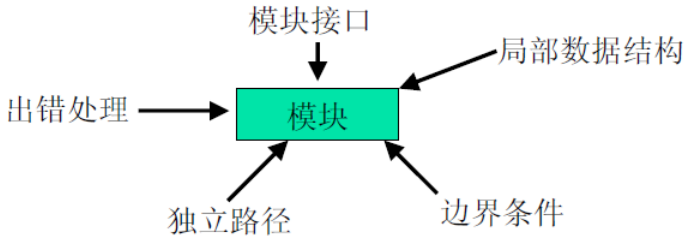
测试计划
计划
测试项目说明
评价

测试分析报告
测试计划执行情况
软件需求测试结论
评价

一、软件分析：是一个对用户的需求进行去粗取精、去伪存真、正确理解，然后把它用软件工程开发语言表达出来的过程,replica soccer jerseys。基本任务是和用户一起确定要解决的问题，建立软件的逻辑模型，编写需求规格说明书文档并最终得到用户的认可。
二、软件设计：主要任务就是将软件分解成模块使之能实现某个功能的数据和程序说明、可执行程序的程序单元。
三、软件编码：指把软件设计转换成计算机可以接受的程序，即写成以某一程序设计语言表示的"源程序清单"。
四、软件测试：目的是以较小的代价发现尽可能多的错误。要实现这个目标的关键在于设计一套出色的测试用例（测试数据和预期的输出结果组成了测试用例）。
五、软件维护：指在已完成对软件的研制（分析、设计、编码和测试）工作并交付使用以后，对软件产品所进行的一些软件工程的活动。根据软件运行的情况，对软件进行适当修改，以适应新的要求，以及纠正运行中发现的错误。

单元测试

内容：根据详细设计说明书、程序清单，采用白盒、黑盒测试的测试用例，对模块进行检查，主要包括以下五部分



集成测试

又称组装测试，联合测试
将所有模块按设计要求组装成系统
一步到位与增量集成

一次到位：

又称为非增式组装，一次性组装或大爆炸集成，这种集成将所有单元在一起编译并进行一次性测试。

缺点：

一次成功的可能性不大

当发现缺陷时，没有多少线索能够用来帮助确定缺陷位置。

增式组装：

逐步组装成较大的系统，边组装边测试

自顶向下的增殖方式

自底向上的增殖方式

混合增殖方式

确认测试

在传统软件和面向对象软件间没有明显差别。

始于集成测试的结束

验证软件的功能和性能及其它特性是否与用户要求一致

α测试和β测试

系统测试是基于实际应用环境对计算机系统的一种多方位的测试，每一种测试都具有不同的目的，但所有的测试都是为了检验各个系统成分能否正确集成到一起并且是否能完成预定的功能。

原型化方法也称为：快速原型法，简称原型法。是根据用户初步需求来利用原型工具快速的建立一个系统模型展示给用户，在此基础上于用户交流。最终实现用户需求的信息系统快速开发的方法。

原型法的特点：

1、可以使系统开发的周期缩短、成本和风险降低、速度加快。获得较高的综合开发效益。

2、原型法是以用户为中心来开发系统的，用户参与的程度大大提高，开发的系统符合用户的需求，因而增加了用户的满意度，提高了系统开发的成功率。

3、由于用户参与了系统开发的全过程，对系统的功能和结构容易理解和接收，有利于系统的移交，有利于系统的运行和维护。

原型法的优点主要在于能够有效的确认用户的需求，适用于那些需求不明确的系统开发。对于分析层面难度大、技术层面不大的系统时候采用原型法开发。而对于技术层面的困难远远大于分析层面的系统，则不宜用原型法。

一、软件测试的目的

1）软件测试是为了发现错误而执行程序的过程。

2）测试是为了证明程序有错，而不是证明程序无错。（发现错误不是唯一目的）

3）一个好的测试用例在于它发现至今未发现的错误。

4）一个成功的测试是发现了至今未发现的错误的测试。

注意：

1、测试并不仅仅是为了要找出错误。通过分析错误产生的原因和错误的分布特征。可以帮助项目管理者发现当前所采用的软件过程的缺陷，以便改进。同时，通过分析也能帮助我们设计出有针对性的检测方法，改善测试的有效性。

2、没有发现错误的测试也是有价值的，完整的测试是评定测试质量的一种方法。详细而严谨的可靠性增长模型可以证明这一点。例如Bev Littlewood发现一个经过测试而正常运行了n个小时的系统有继续正常运行n个小时的概率。

二、软件测试的原则

1）应当把“尽早地不断地进行软件测试“作为软件开发者的座右铭。

2）测试用例应由测试数据和与之对应的预期输出结果这两部分组成。

3）程序员应避免检查自己的程序。

4）在设计测试用例时，应当包括合理的输入条件和不合理的输入条件。

5）充分注意测试中的群集现象。

6）严格执行测试计划，排除测试的随意性。

7）应当对每一个测试结果做全面的检查。

8）妥善保存测试计划、测试用例、出错统计和最终分析报告，为维护提供方便。

三、软件测试的流程

立项阶段–需求阶段----设计阶段----编码和单元测试阶段----集成测试阶段----系统测试阶段-----验收测试阶段-----结项总结阶段

补充：

根据不同的测试阶段，测试可以分为单元测试、集成测试、系统测试和验收测试。

体现了测试由小到大、又内至外、循序渐进的测试过程和分而治之的思想。

单元测试的粒度最小，一般由开发小组采用白盒方式来测试，主要测试单元是否符合“设计”。

集成测试界于单元测试和系统测试之间，起到“桥梁作用”，一般由开发小组采用白盒加黑盒的方式来测试，既验证“设计”，又验证“需求”。

系统测试的粒度最大，一般由独立测试小组采用黑盒方式来测试，主要测试系统是否符合“需求规格说明书”。

验收测试与系统测试相似，主要区别是测试人员不同，验收测试由用户执行。

黑盒测试不考虑程序内部结构和逻辑结构，主要是用来测试系统的功能是否满

足需求规格说明书。一般会有一个输入值，一个输入值，和期望值做比较。

白盒测试主要应用在单元测试阶段，主要是对代码级的测试，针对程序内部逻辑结构，测试手段有：语句覆盖、判定覆盖、条件覆盖、路径覆盖、条件组合覆盖

集成测试主要用来测试模块与模块之间的接口，同时还要测试一些主要业务功能。

系统测试是在经过以上各阶段测试确认之后，把系统完整地模拟客户环境来进行的测试。

获得值分析是项目费用管理中常用的一种费用控制方法。
获得值（Earned Value，也译为盈余值，挣值）可以较为客观地显示出项目计划工作量和实际工作量之间的偏差，确定项目费用是否按计划执行。
BCWS：在规定的时间内所有计划执行活动的已批准预算费用总和称为计划工作预算成本（费用控制和分析的基准）
ACWP：已完成工作实际成本
BCWP：已完成工作预算成本
项目的费用偏差CV＝BCWP－ACWP
项目的进度偏差SV＝BCWP－BCWS
费用执行情况指数 CPI＝BCWP/ACWP
进度执行情况指数SPI＝BCWP/BCWS
预测完工费用＝总预算费用 / CPI

产品负责人、Scrum Master、Scrum团队
Scrum是迭代式增量软件开发过程，通常用于敏捷软件开发。
Scrum包括了一系列实践和预定义角色的过程骨架。
Scrum中的主要角色包括同项目经理类似的Scrum主管角色负责维护过程和任务，产品负责人代表利益所有者，开发团队包括了所有开发人员。虽然Scrum是为管理软件开发项目而开发的，它同样可以用于运行软件维护团队，或者作为计划管理方法：Scrum of Scrums.

1. 软件规模估算
基于问题的估算（LOC估算，FP估算）
基于过程的估算
基于用例的估算

基于代码行的估计，功能点的估计
面向对象估计方法

面向对象项目的估算
1. 使用工作量分解、FP分析和任何其他适用于传统应用的方法进行估算
2. 使用面向对象的分析模型建立用例并确定用例数。
3. 由分析模型确定关键类（分析类）的数量
4. 对应用的界面类型进行归类，确定支持类的数量
5. 将类的总数（关键类+支持类）乘以每个类的平均工作单元数。Lorenz和Kidd建议每个类的平均单元数是15—20人.日
6. 将用例数乘以每个用例的平均工作单元数，对基本类的估算做交叉检查。

极限编程实践
完整团队
计划游戏
客户测试
简单设计
结对编程
测试驱动开发
改进设计
持续集成
集体代码所有权
编码标准
隐喻
可持续的速度

- 增量模型；
  以迭代方式运用瀑布模型

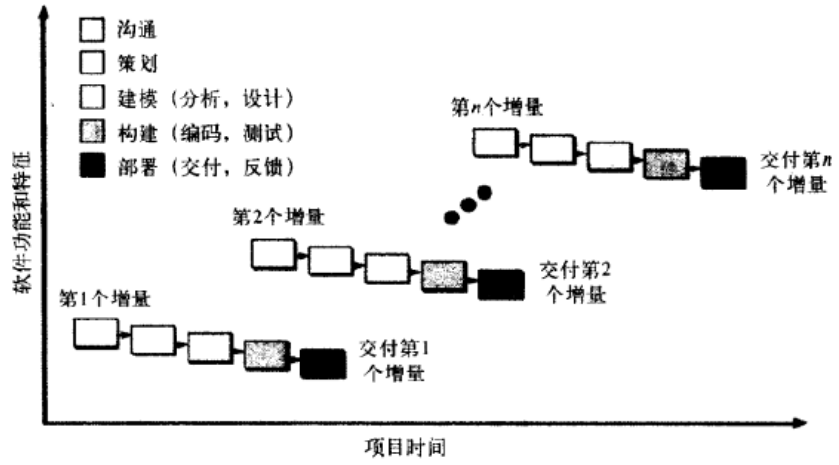


图3-2 增量模型

特点：

- 一般来讲，最重要的增量放在前面。
- 每次交付的增量产品都是可用的。
- 适合于功能可以划分，而且时间不紧迫的情况。
- 可以规避一定的风险。如有些技术还不稳定，将这部分放到后边。

对原型的基本要求包括：

* 体现主要的功能；
* 提供基本的界面风格；
* 展示比较模糊的部分，以便于确认或进一步明确，防患于未然。
* 原型最好是可运行的，至少在各主要功能模块之间能够建立相互连接。

基线(Baseline)由一组配置项组成，这些配置项构成了一个相对稳定的逻辑实体，是一组经过正式审查并且达成一致的范围或工作产品。基线中的配置项被“冻结”了，不能再被任何人随意修改。基线通常对应于开发过程中的里程碑，一个产品可以有多个基线，也可以只有一个基线。产品的测试版本可以作为一个基线。

- 掌握人机界面设计的黄金规则
  1. 置于用户的控制之下
  2. 减少用户的记忆负担
  3. 保持界面一致

- 掌握功能点的计算方法
  - FP=总计数值×［0.65＋0.01×ΣFi］
  - 其中“总计数值”是从上页图中得到的所有条目的总和。
  - F_i (i=1 到14)是基于对下面的问题的回答而得到的“复杂度调整值”(0 到5)。
  - 等式中的常数和信息域值的加权因子是根据经验确定的。

五个最基本的过程框架活动是沟通、 策划、建模、构建和 部署

2.软件工程的框架是?
答:沟通，策划，建模，构建，部署
通用过程框架通用过程框架可适用于绝大多数的软件项目，该框架由沟通、策划、建模、构造和部署5个通用框架活动组成。

(1)沟通。这项框架活动包含系统分析员与客户之间大量的交流和协作，还包括需求获取以及其他相关活动。

(2)策划。策划活动协助软件开发团队定义全局目标，并为后续的软件工程工作制定计划。策划活动包括一系列管理和技术实践，如描述需要执行的技术任务、可能的风险、资源需求、工作产品和工作进度计划。

(3) 建模。建模的目的是为了更好地理解需要构建的实体。在软件工程中，要创建两类模型:分析模型和设计模型。分析模型通过描述软件的信息域、功能域及行为域来表达客户的需求;设计模型描述软件架构、用户界面及构件细节，从而帮助开发者高效地开发软件。

(4)构造。构造活动包括一系 列构件组装、编码和测试任务，从而为向客户和最终用户交付可运行软件做好准备。

(5)部署。部署活动是将软件(全部或者完成的部分)交付给用户，用户对其进行评测并给出反馈意见。部署活动包括三个动作:交付、支持和反馈。

等价类划分

- 使用这一方法时，完全不考虑程序的内部结构，只依据程序的规格说明来设计测试用例。
- 等价类划分方法把所有可能的输入数据，即程序的输入域划分成若干部分，然后从每一部分中选取少数有代表性的数据做为测试用例。

划分等价类
①有效等价类：是指对于程序的规格说明来说，是合理的，有意义的输入数据构成的集合。
②无效等价类：是指对于程序的规格说明来说，是不合理的，无意义的输入数据构成的集合。

划分等价类的原则：
(1) 如果输入条件规定了取值范围，或值的个数，则可以确立一个有效等价类和两个无效等价类。

(2) 如果输入条件规定了输入值的集合，或者是规定了“必须如何”的条件，这时可确立一个有效等价类和一个无效等价类。

(3) 如果输入条件是一个布尔量，则可以确定一个有效等价类和一个无效等价类。

(4) 如果规定了输入数据的一组值，而且程序要对每个输入值分别进行处理。这时可为每一个输入值确立一个有效等价类，此外针对这组值确立一个无效等价类，它是所有不允许的输入值的集合。

(5) 如果规定了输入数据必须遵守的规则，则可以确立一个有效等价类（符合规则）和若干个无效等价类（从不同角度违反规则）。

测试用例设计：尽可能多地覆盖尚未被覆盖的有效等价类；重复这一步骤直到所有有效类都被覆盖为止。设计一个新方案以覆盖一个且仅一个尚未被覆盖的无效等价类；重复这一步骤直到所有无效类都被覆盖为止。

1，先确定等价类别
2，找出有效等价类和无效等价类
3，边界值找好，尽可能多的找的会有重复的数据
4，有效等价类尽可能条件符合的归一起不要重复
5，无效等价类单独写开
6，写好测试用例
7，执行测试用例

黑盒测试的测试用例设计

- 等价类划分
- 边界值分析
- 错误推测法
- 因果图