1. 瀑布模型：

|  |  |
| --- | --- |
| 优点 | 缺点 |
| 可使得开发人员采用规范的方法，有组织有管理 | 开发过程不可逆装，否则代价太大 |
| 具有简单性，易向顾客做出解释 | 客户往往不能给出全面需求，会影响后面各阶段的工作 |
| 严格规定了每个阶段必须提交的文档 | 缺乏向管理员和开发人员提供相关指导，例如编码活动中发生需求变化时，随之带来的设计变化并没有在模型中强调 |
| 是很多复杂模型的基础 | 没有把软件看作是一个问题求解过程 |
| 有利于软件开发方法和工具的研究 | 只能通过文档了解产品，不经过实践的需求是不切实际的 |
|  | 并没有说明我们创建产品过程中所需的往返活动的任何特有信息 |

1. V模型（瀑布模型的变种）

|  |  |
| --- | --- |
| 优点 | 缺点 |
| 反映了软件从需求定义到实现于测试活动的关系，强调在整个软件项目生命周期中需要经历的若干开发与测试的对应级别 | 测试和开发活动也保持线性的前后关系,只有上一阶段完成 ,才开始下一阶段活动,因此,该模式不易支持迭代方式和不适应在开发过程中作变更调整。 |
| 体现“尽早地和不断地进行软件测试”原则 | 忽视了测试对需求分析,系统设计的验证，需求的满足情况一直到后期的验收测试才被验证 |
| 测试伴随整个软件开发周期,并且 测试对象不仅是程序,需求、设计同样需要进行测试(评审),测试与开发实行同步,从而有利于尽早发现软件潜在问题 |  |

1. 原型化模型

|  |  |
| --- | --- |
| 优点 | 缺点 |
| 开发人员和用户在“原型”上达成一致。这样一来，可以减少设计中的错误和开发中的风险，也减少了对用户培训的时间，而提高了系统的实用、正确性以及用户的满意程度 | 当告诉用户，还必须重新生产该产品时，用户是很难接受的。这往往给工程继续开展带来不利因素 |
| 缩短了开发周期，加快了工程进度 | 开发者为了使一个原型快速运行起来，往往在实现过程中采用这种手段 |
| 降低成本 | 不宜利用原型系统作为最终产品。采用原型模型开发系统，用户和开发者必须达成一致：原型被建造仅仅是用户用来定义需求，之后便部分或全部抛弃，最终的软件是要充分考虑了质量和可维护性等方面之后才被开发 |

1. 阶段化开发模型：增量和迭代

增量模型：

|  |  |
| --- | --- |
| 优点 | 缺点 |
| 整个产品被分为若干个构建逐步交付，用户可以看到系统开发的全过程 | 需要开放式架构，以便各构件逐步进入 |
| 主干部分优先交付可以得到充分的测试 | 需求难以在增量实现之前全部定义，因此增量与需求之间的准确映射难以以及所有增量的有效集成比较困难，容易退化成边做边改的模式，使软件过程的控制失去整体性 |
| 降低开发风险 |  |
| 将早期增量作为原型有助于后期需求的明确 |  |

迭代模型：

|  |  |
| --- | --- |
| 优点 | 缺点 |
| 第一个可交付版本的软件所需的成本与时间较小 | 如果对用户需求的变更没有整体的规划，可能会变化为"边做边开发"的模式 |
| 能够适应客户的需求变更，当需求变化时，只需要修改某一个组件即可 | 最终集成各个组件时，可能会出现集成失败的风险 |

1. 螺旋模型

|  |  |
| --- | --- |
| 优点 | 缺点 |
| 是风险驱动的 | 需要具备相当丰富的风险评估经验和专业知识，在风险较大的项目开发中，若未能及时标识风险，会造成重大损失 |
| 对可选方案和约束条件的强调有利于已有软件的重用，也有助于把软件质量作为软件开发的一个重要目标 | 过多的迭代次数会增加开发成本，延迟提交时间 |
| 减少了过多测试或测试不足所带来的风险 |  |
| 支持用户需求的动态变化有助于提高产品的适应能力 |  |
| 特别适合用于庞大、复杂并且具有高风险的系统 |  |

1. 敏捷模型

|  |  |
| --- | --- |
| 优点 | 缺点 |
| 用户很快可以看到一个基线架构版的产品 | 对于某些软件交付物，特别是大型软件交付物，很难在软件开发生命周期开始时评估所需的工作 |
| 敏捷注重市场快速反应能力，客户前期满意度高 | 缺乏对必要的设计和文件的重视 |
| 持续关注卓越的技术和良好的设计 | 如果项目人员流动大太，会增加项目维护难度 |
| 即使是最新的需求变化也受到欢迎 | 软件之前版本的可重现性、可回溯性较低 |
|  | 对于较大的项目,人员越多,面对面的有效沟通越困难。因此，该模型适用于小型项目的开发 |
|  | 只有高级程序员才能在开发过程中做出所需的决策。因此，除非与经验丰富的资源相结合，否则对于新手程序员来说，它没有立足之地 |