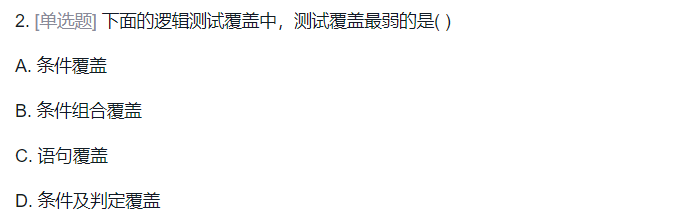
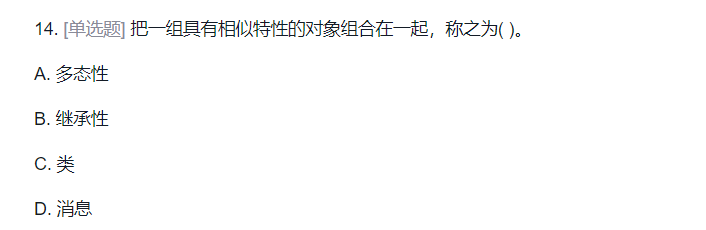
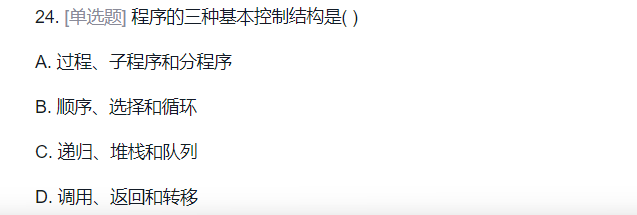
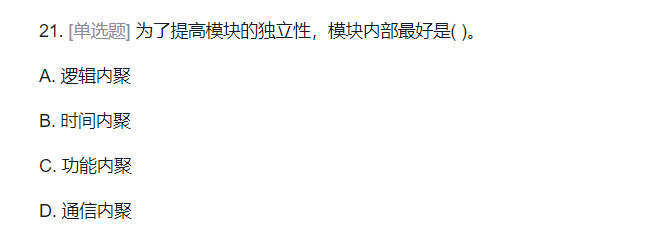
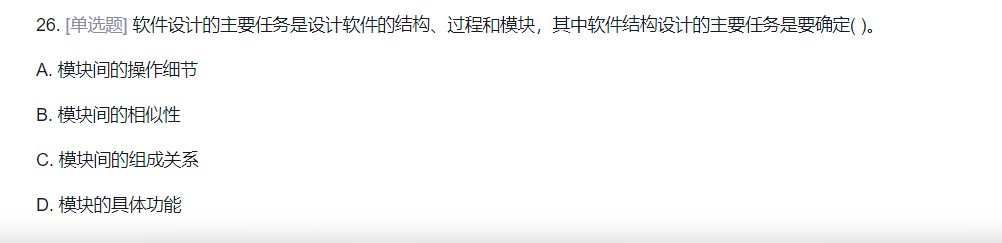
## 一

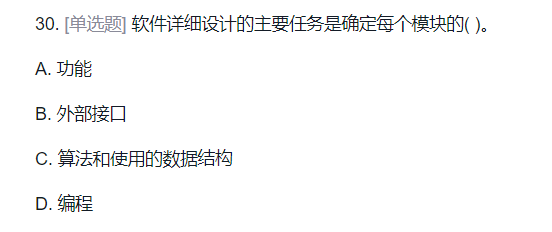


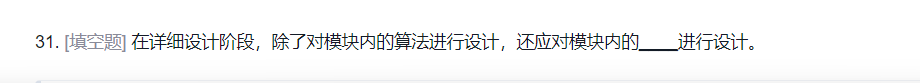


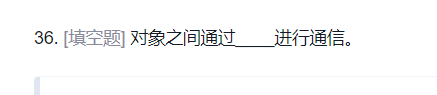


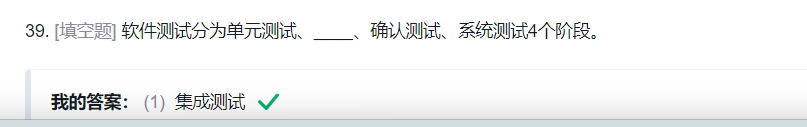
顺序、选择、循环



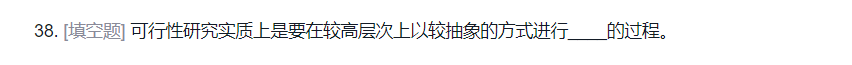




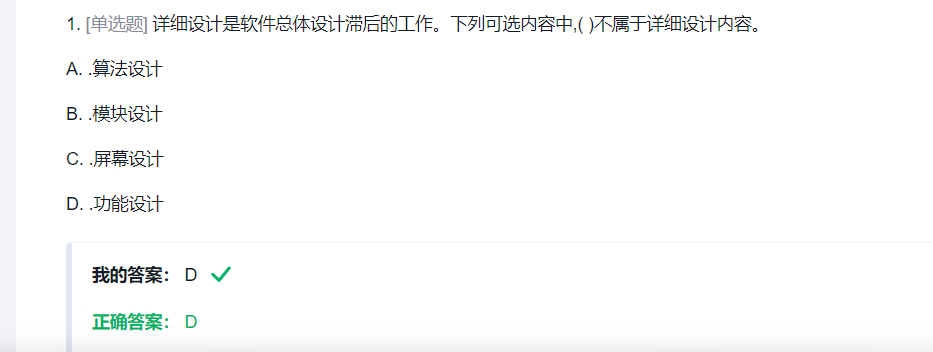




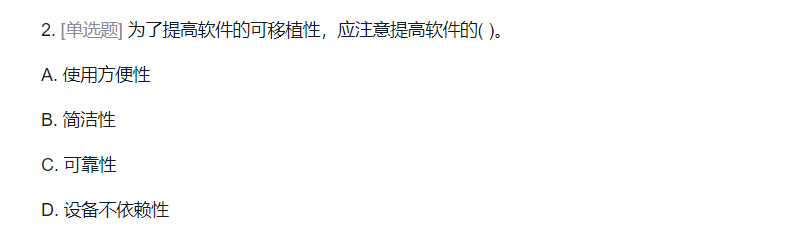
单元测试、集成、系统、确认测试

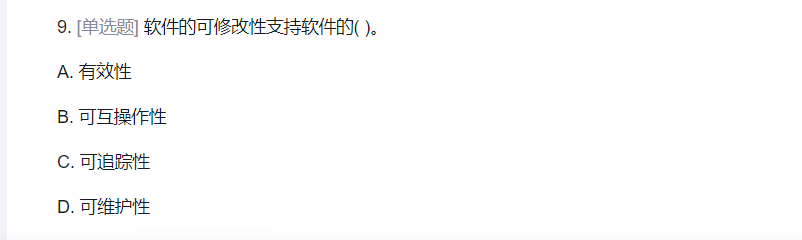


## 二

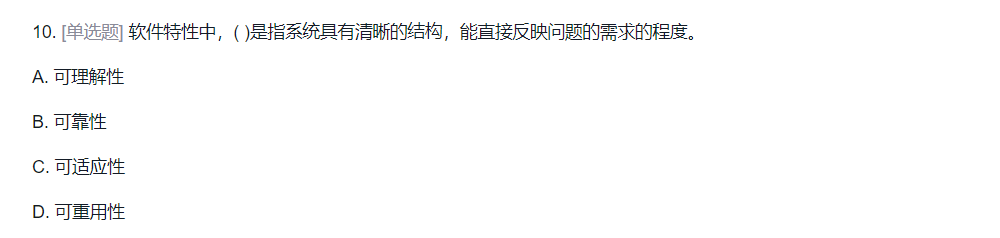


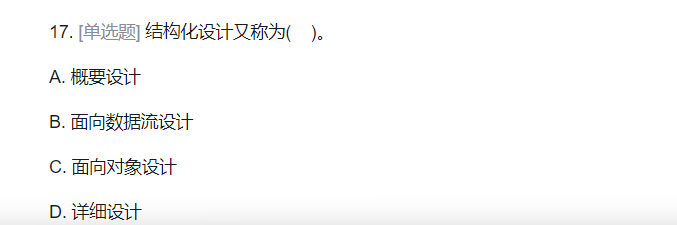
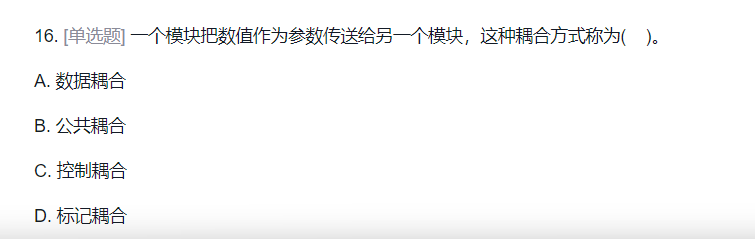
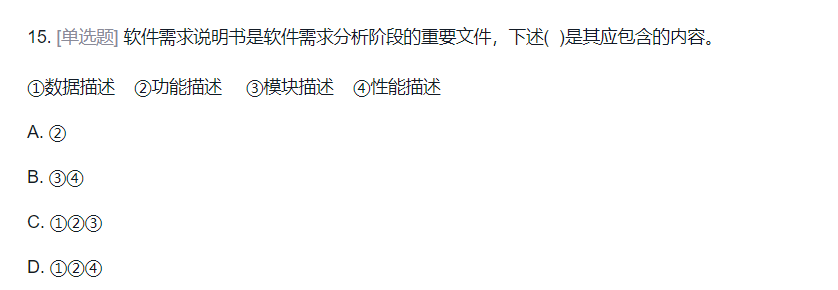
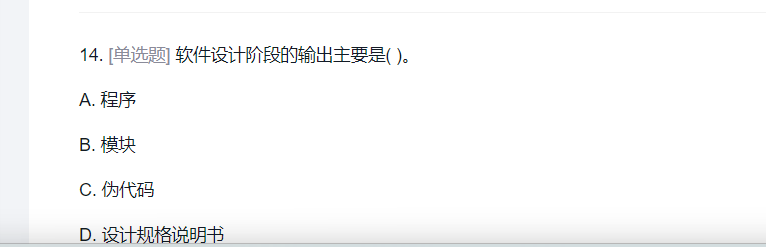
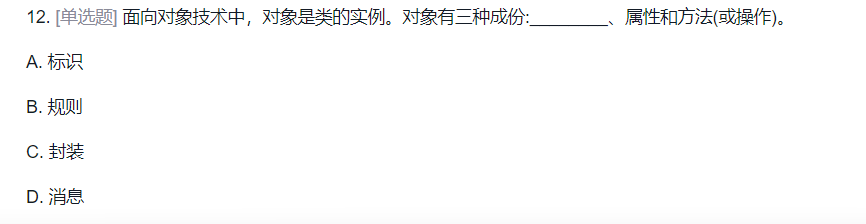
D



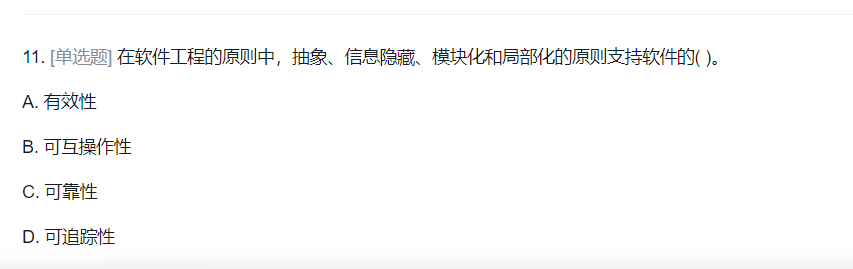


可维护性

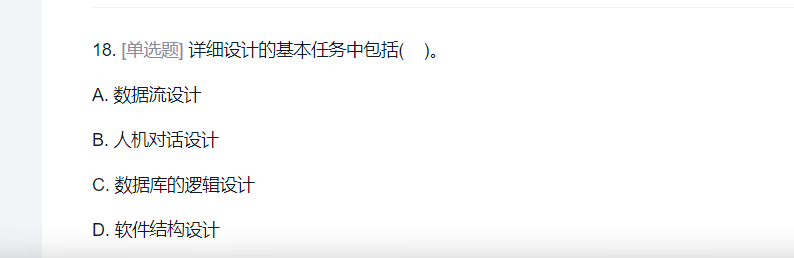


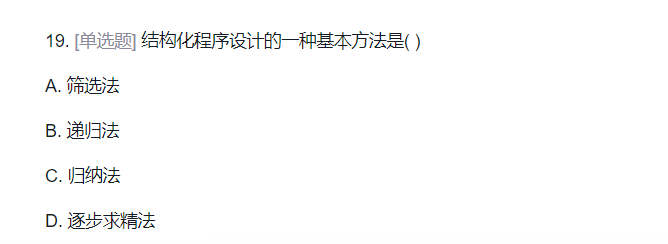


B

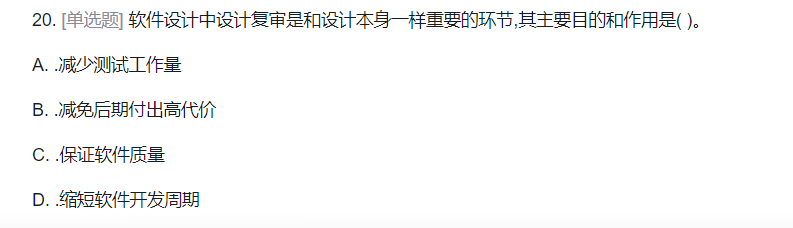


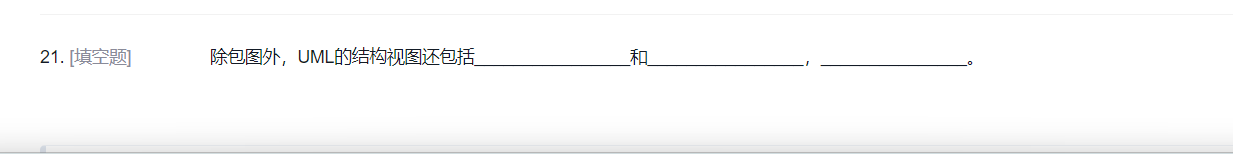
可靠性



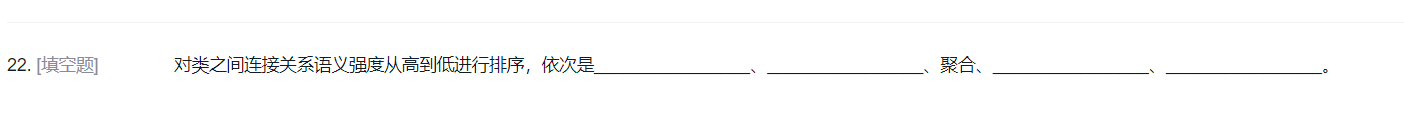


D

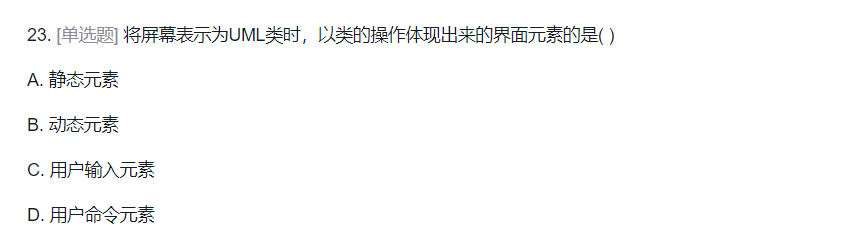


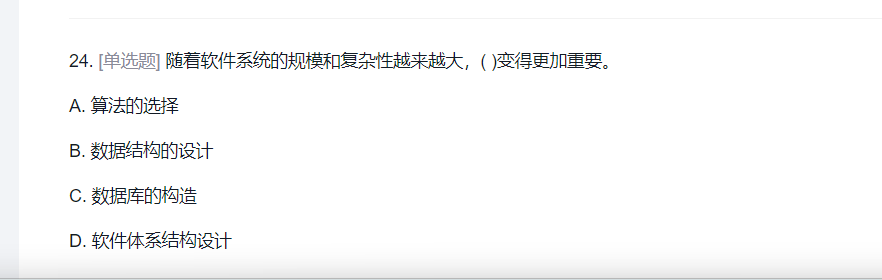


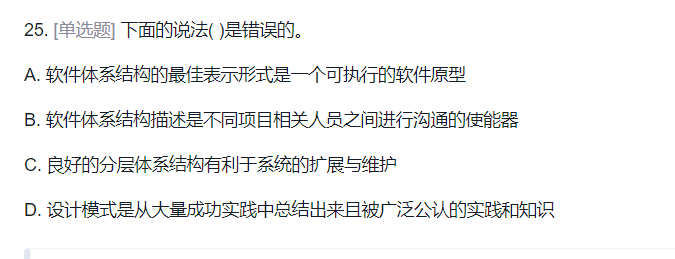
包图、类图、构件图、对象图

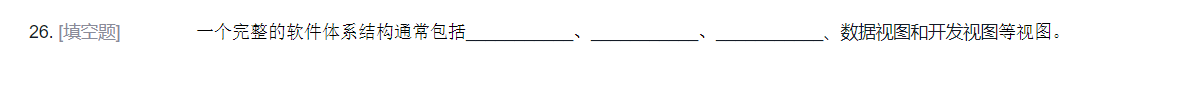


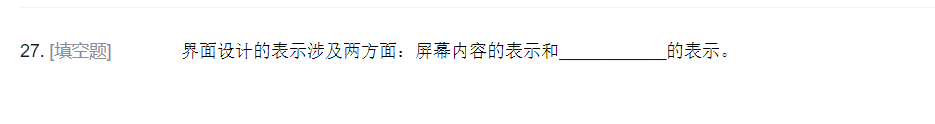
组合、聚合、继承、关联、依赖



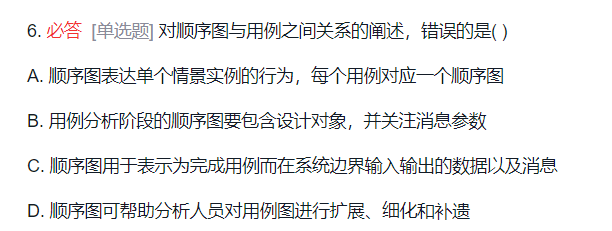


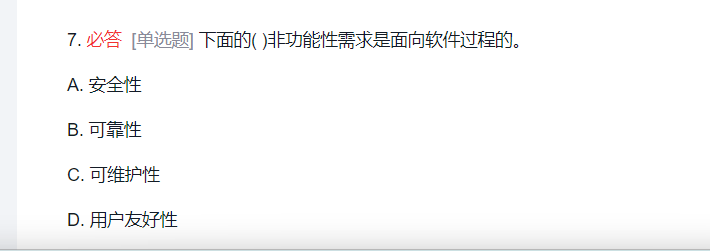




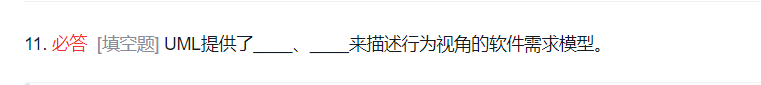


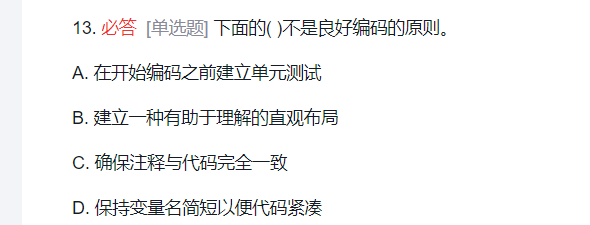
## 三

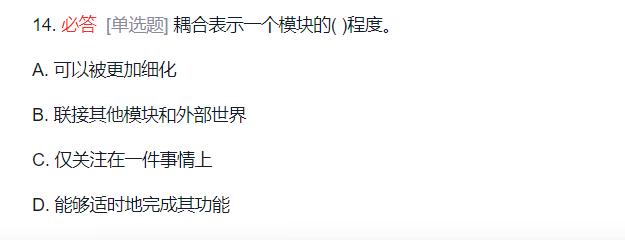
B

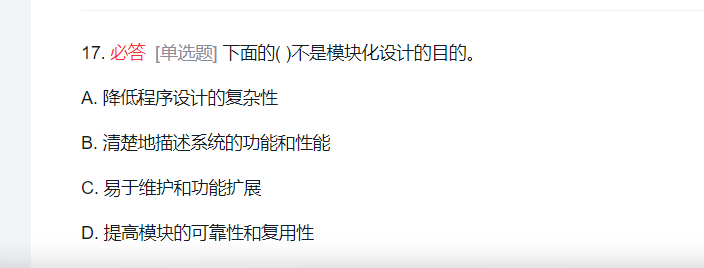


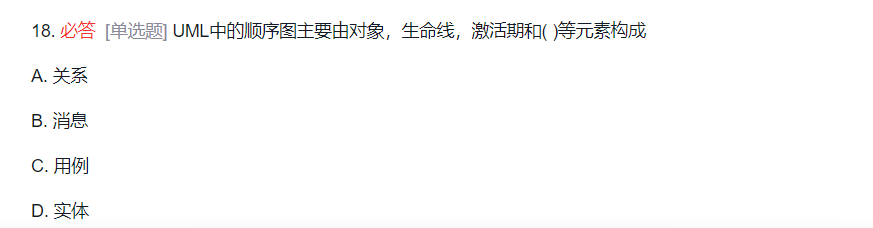
C

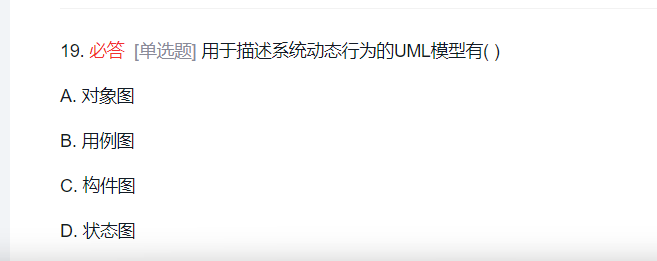


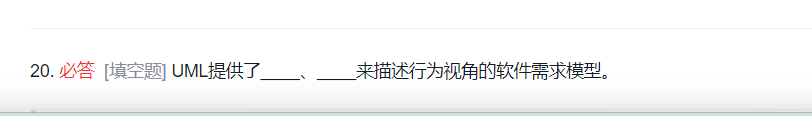












状态图和交互图

瀑布模型的生命周期通常包括以下八个阶段：

### 1. 问题定义

- \*\*目标\*\*：明确软件项目的目标和范围。确定要解决的问题是什么，以及软件系统需要满足哪些需求。

- \*\*关键活动\*\*：与利益相关者（如客户、用户、管理层等）沟通，收集初步的需求信息，了解业务场景和期望的软件功能。

### 2. 可行性研究

- \*\*目标\*\*：评估项目在技术、经济和操作上的可行性。确定是否有足够的资源和技术能力来开发该软件，以及开发该软件是否具有经济价值。

- \*\*关键活动\*\*：进行技术调研，分析现有技术能否满足项目需求；进行成本 - 效益分析，评估项目的投资回报率；考虑操作层面的问题，如软件的运行环境、维护成本等。

### 3. 需求分析

- \*\*目标\*\*：详细地收集、分析和定义软件的功能和非功能需求。确保开发团队准确理解用户的需求和期望。

- \*\*关键活动\*\*：与用户和利益相关者进行深入访谈和调研，采用问卷调查、现场观察等方法收集需求；对收集到的需求进行分类、整理和细化，形成详细的需求规格说明书。

### 4. 总体设计

- \*\*目标\*\*：设计软件的总体架构和模块划分。确定软件系统的主要组成部分及其相互关系。

- \*\*关键活动\*\*：根据需求规格说明书，设计软件的高层架构，包括选择合适的软件架构风格（如分层架构、客户端 - 服务器架构等）；将系统划分为不同的模块，并定义模块之间的接口。

### 5. 详细设计

- \*\*目标\*\*：对总体设计中的各个模块进行详细设计，包括算法设计、数据结构设计、数据库设计等。

- \*\*关键活动\*\*：为每个模块设计具体的算法和数据结构；进行数据库设计，包括数据库的模式、表结构、关系等；确定模块内部的详细处理流程和操作逻辑。

### 6. 编码和单元测试

- \*\*目标\*\*：根据详细设计进行编码实现，并对每个模块进行单元测试，确保模块的功能正确性。

- \*\*关键活动\*\*：程序员按照详细设计文档编写代码；对编写好的代码进行单元测试，使用测试用例验证模块的功能是否符合设计要求，发现并修复模块内部的缺陷。

### 7. 综合测试

- \*\*目标\*\*：将各个经过单元测试的模块集成在一起，进行集成测试、系统测试和验收测试，确保整个软件系统的功能和性能符合要求。

- \*\*关键活动\*\*：进行集成测试，检查模块之间的接口是否正确，模块集成后是否能正常工作；进行系统测试，从整体上对软件系统的功能、性能、可靠性等方面进行测试；最后进行验收测试，由用户或客户参与，验证软件是否满足业务需求。

### 8. 维护

- \*\*目标\*\*：在软件交付后，对软件进行维护，包括修复缺陷、优化性能、满足新的需求等。

- \*\*关键活动\*\*：处理用户反馈的问题，对软件进行缺陷修复；根据用户新的需求对软件进行修改和扩展；对软件的性能进行监控和优化，确保软件长期稳定运行。

瀑布模型是一种线性的、顺序的软件开发模型，前一个阶段完成后才进入下一个阶段。它强调文档的完整性和准确性，适合需求明确、稳定的项目。