1. **Engenharia de Software:** A Engenharia de Software é uma disciplina que envolve a aplicação de princípios de engenharia para o desenvolvimento de software. Ela abrange todas as atividades necessárias para especificar, projetar, implementar, testar e manter sistemas de software de forma eficiente e confiável.

2. **Projeto (PMBOK):** De acordo com o PMBOK, um projeto é um esforço temporário empreendido para criar um produto, serviço ou resultado único. Um projeto é caracterizado por ter um início e um fim definidos, um escopo bem definido, recursos alocados e uma série de atividades planejadas para alcançar seus objetivos.

3. **Arquitetura de Software:** A arquitetura de software se refere à estrutura organizacional de um sistema de software, incluindo componentes, relacionamentos, padrões e princípios que orientam o design e a implementação. Ela define como os diversos componentes de um sistema interagem e se relacionam para atender aos requisitos do sistema.

4. **Componente de Software e Desenvolvimento Baseado em Componentes:** Um componente de software é uma unidade modular e reutilizável que desempenha uma função específica em um sistema de software. O desenvolvimento baseado em componentes é uma abordagem que enfatiza a criação e reutilização de componentes de software em vez de escrever código do zero. Os componentes podem ser combinados para criar sistemas mais complexos.

5. **Arquiteturas de Sistemas:**

- **UNIX (Arquitetura Monolítica):** No UNIX, o sistema operacional é composto por um único núcleo (kernel) que interage diretamente com o hardware, executando todos os serviços do sistema. Isso pode resultar em eficiência, mas também em falta de isolamento entre os serviços.

- **Windows (Arquitetura Cliente-Servidor):** O Windows utiliza uma arquitetura cliente-servidor, onde o núcleo do sistema atua como um servidor que fornece serviços a aplicativos (clientes). Isso permite uma melhor separação de responsabilidades e segurança.

- **Pilha de Protocolos Ethernet (Arquitetura em Camadas):** A pilha de protocolos Ethernet é organizada em camadas, como a arquitetura OSI, onde cada camada fornece serviços para a camada superior. Isso permite a modularidade e a interoperabilidade de sistemas.

Vantagens e Desvantagens:

- Arquitetura Monolítica: Vantagem - eficiência; Desvantagem - falta de flexibilidade.

- Arquitetura Cliente-Servidor: Vantagem - escalabilidade e segurança; Desvantagem - complexidade.

- Arquitetura em Camadas: Vantagem - modularidade e interoperabilidade; Desvantagem - sobrecarga.

6. **Princípios da Arquitetura de Microsserviços:** Os princípios da arquitetura de microsserviços incluem a divisão de um sistema em serviços independentes, a comunicação via API, a escalabilidade independente de serviços, a implantação independente e a responsabilidade única.

7. **Biblioteca de Funções vs. Framework:**

- Uma biblioteca de funções é um conjunto de funções reutilizáveis que podem ser usadas em um programa. Um exemplo é a biblioteca padrão do Python.

- Um framework (arcabouço) é uma estrutura que fornece diretrizes e estrutura para o desenvolvimento de aplicativos. Exemplos incluem o framework web Django e o framework de desenvolvimento de jogos Unity.

A escolha entre biblioteca e framework depende do contexto. As bibliotecas oferecem maior flexibilidade, enquanto os frameworks fornecem estrutura e convenções que podem acelerar o desenvolvimento.

8. **API (Application Programming Interface):** Uma API é um conjunto de regras e protocolos que permite que diferentes componentes de software interajam uns com os outros. Ela define as formas e os métodos pelos quais os aplicativos podem solicitar e compartilhar dados e funcionalidades.

9. **a) Fraco Acoplamento:** Fraco acoplamento refere-se à redução da dependência entre os componentes de um sistema. Componentes fracamente acoplados são independentes uns dos outros, o que torna mais fácil alterar ou substituir um componente sem afetar os outros.

b) Alta Coesão: Alta coesão indica que as responsabilidades de um componente são bem definidas e relacionadas, ou seja, um componente deve se concentrar em realizar uma única função ou tarefa de maneira coesa.

10. **Separação de Interface e Implementação:** A separação da interface (API) da implementação é uma prática recomendada para promover o encapsulamento e o fraco acoplamento. Isso permite que os detalhes internos de um componente sejam alterados sem afetar os componentes que o utilizam, desde que a interface permaneça a mesma. Isso facilita a manutenção e a evolução do software.

11. **Reuso de Código:** O reuso de código envolve o uso de componentes de software existentes em vez de escrever novo código do zero. Isso economiza tempo e recursos. Vantagens incluem economia de tempo, redução de erros e padronização. Desvantagens podem incluir complexidade de integração e dependência de componentes externos.

12. **Fases no Desenvolvimento de Software:**

- **Análise de Requisitos:** Compreensão dos requisitos do sistema.
- **Projeto:** Design da arquitetura e estrutura do sistema.
- **Implementação:** Codificação do sistema.
- **Testes:** Verificação e validação do software.
- **Implantação:** Lançamento do software para uso.
- **Manutenção:** Atualizações e correções ao longo do tempo.

13. **Verificação vs. Validação de Software:**

- **Verificação:** É o processo de avaliar se o software atende às especificações e requisitos. Envolve revisões, inspeções e testes para garantir que o código foi implementado corretamente.
- **Validação:** É o processo de avaliar se o software atende às necessidades e expectativas do usuário final. Envolve testes para garantir que o software atende aos objetivos do cliente.

14. **Níveis de Teste de Software:**

- **a) Teste Unitário:** Testa unidades individuais de código, como funções ou métodos.
- **b) Teste Funcional:** Testa se

o software atende aos requisitos funcionais.

- **c) Teste de Integração:** Testa a interação entre componentes do sistema.
- **d) Teste Sistêmico:** Testa o sistema como um todo, incluindo a integração de módulos.
- **e) Teste de Aceitação:** Verifica se o software atende aos critérios de aceitação do cliente.

15. **Tipos de Teste de Software:**

- **a) Teste Caixa Branca:** Envolve o teste do código-fonte e da lógica interna do software.
- **b) Teste Caixa Preta:** Envolve testar o software com base em suas entradas e saídas, sem conhecimento interno do código.

- **c) Teste Caixa Cinza:** Combina elementos de teste caixa branca e caixa preta, ou seja, conhece parte da estrutura interna do software.