Universidade Federal de Ouro Preto - Campus Ouro Preto

Lívia Stéffanny de Sousa - 20.1.4029

**BCC322 - Engenharia de Software I**

# LISTA I – INTRODUÇÃO À ENGENHARIA DE SOFTWARE

Ouro Preto
2022

Lívia Stéffanny de Sousa - 20.1.4029

# LISTA I – INTRODUÇÃO À ENGENHARIA DE SOFTWARE

.

Professor Orientador: Tiago G. S. Carneiro

Ouro Preto
2022

**1. Qual sua definição para o termo “Engenharia de Software”?**

**R:** Para mim, engenharia de software é uma área da ciência da computação que não se preocupa diretamente com determinados algoritmos para a resolução de um problema ‘x’, mas sim na utilização de tal. Ou seja, com o desenvolvimento da aplicação como um todo, como será organizada, implementada, todo o processo desde levantamento de requisitos até a entrega final, quais tecnologias serão usadas, compreender e atender a demanda da determinada aplicação.

**2. O que é um projeto segundo o PMBOK (Project Management Body of Knowledge)?**

**R:** PMBOK é um guia de melhores práticas em gerenciamento criado pelo PMI (Project Management Institute), uma instituição sem fins lucrativos que associa e certifica profissionais relacionados à área de projetos. Ele descreve de forma sistemática o trabalho que deve ser realizado durante o ciclo de vida de um projeto. Para o PMBOK, um projeto consiste em um empreendimento temporário, que tem um início, fim e recursos bem definidos, a fim de criar um produto, realizar um serviço ou chegar a um determinado resultado específico.

**3. O que é arquitetura de software?**

**R:** De acordo com o que vimos nas aulas expositivas, consiste na definição dos componentes de software e a forma em que ele se comunica com os outros. É uma representação abstrata de um determinado software. Onde, trata-se da relação dos principais componentes, a relação com outros softwares, como devem ser organizados, a estrutura geral do sistema. De uma maneira análoga, tem como exemplo um projeto arquitetônico de uma casa, onde você tem um desenho (projeção), abstrato, do produto final. Com isso essa documentação, registra as necessidades do cliente e decisões iniciais, contribui para reuso de componentes padrões e facilita a comunicação entre os stakeholders (envolvidos).

**4. O que é componente de software? O que é desenvolvimento baseado em componentes?**

**R:** É uma parte do sistema, composição, independente que pode ser substituída e com uma interface bem definida. Desenvolvimento baseado em componentes, é um desenvolvimento organizado por componentes independentes, onde cada um possui sua interface e funcionalidade específica.

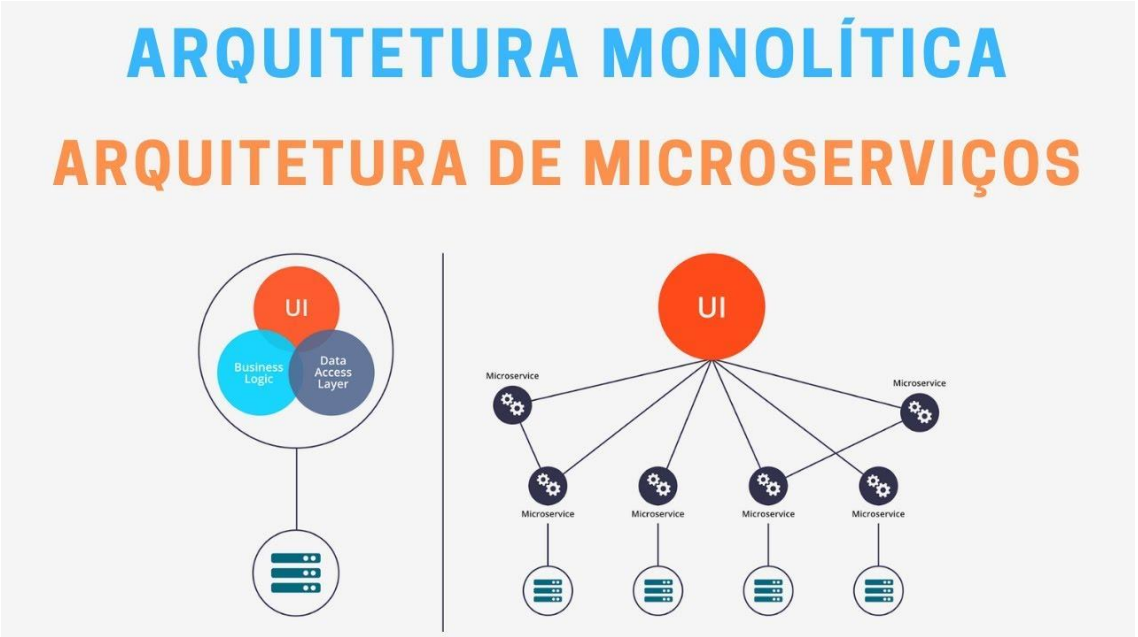
**5. Originalmente o UNIX possuía uma arquitetura monolítica, o Windows possui uma arquitetura cliente-servidor, a pilha de protocolos Ethernet possui uma arquitetura em camadas. Explique como essas arquiteturas são organizadas e como funcionam, utilize figuras. Que vantagens e desvantagens cada uma dessas arquiteturas possui?**

**R:** Monolítica é onde tem um único arquivo executável com todos os componentes, a comunicação a dados, regra de negócio, interface do usuário e etc., juntos e compilados. Há uma grande vantagem em velocidade, pois estando tudo junto facilita a comunicação entre os componentes. Mas, por outro lado, a grande desvantagem é que se um componente apresentar algum problema, a aplicação toda irá falhar em conjunto.

**6. Quais são os princípios da arquitetura de Microsserviços? Apresente uma figura e a explique.**

**R:** A ideia principal da arquitetura de microsserviços é o desenvolvimento de uma única aplicação desmembrada em vários microsserviços. Onde cada micro serviço é desenvolvido em torno de um conjunto de regras de negócio específico, e é implementado de forma independente. Diferente da arquitetura monolítica que possui a aplicação como um único bloco, um único serviço. Na imagem abaixo podemos observar o que foi dito acima. Nessa imagem o exemplo

de microserviçoes ainda mostra uma grande vantagem de sua arquitetura que é trabalhar com várias bases de dados de uma maneira menos complexa.




**7. Qual a principal diferença entre as seguintes arquiteturas de software: biblioteca de funções e framework (arcabouço)? Dê exemplo de software largamente conhecidos que possuam essas arquiteturas. Quando uma arquitetura é preferível à outra?**

**R:** Uma arquitetura de biblioteca de funções, se caracteriza por uma biblioteca de funções que fazem parte de um determinado sistema ou criada pelo desenvolvedor, usuário. Uma arquitetura de framework, é baseada em uma rotina de projeto, são códigos comuns existentes em vários projetos provendo uma funcionalidade genérica. Também, podendo ter funcionalidade específicas. IDE - Integrated Development Environment, ambiente de desenvolvimento integrado, é um software para desenvolvimento de software que utiliza-se tais arquiteturas. Se uma arquitetura é mais adequada que a outra, vai depender do objetivo do projeto. Se deseja fazer um programa específico que calcula a quantidade de material que vai precisar para revestir uma casa, uma arquitetura de biblioteca atende. De outra maneira, deseja-se fazer uma aplicação com esses mesmos problemas, onde deseja fazer a medição através de um aplicativo que já realiza um orçamento com uma empresa e por ele mesmo também finalizo a compra, uma arquitetura de framework é mais adequada.

**8. Defina o conceito de API – (Application Programming Interface).**

**R:** É uma aplicação que pode ser implementada a um projeto de software, que integra sistemas, realizando tarefas. De uma maneira mais técnica, seria uma função, ou um código, que realiza tarefas e que integra com algum serviço, fortemente tipado seguindo o conceito de interface. O conceito de interface, de uma função, tem a ver com a estrutura e parâmetros, após definidos terão a obrigatoriedade de serem respeitados quando a API for requisitada. API do GoogleMaps por exemplo, é uma API Rest onde suas requisições são feitas via protocolo https, muito conhecida entre os desenvolvedores, realiza busca de endereços, realiza rotas, calcula áreas, entre outras funcionalidades. Integrada a um projeto, é possível realizar essas tarefas sem a necessidade de o desenvolvedor criar um algoritmo do zero, basta instalar a API, aprender a usar com base em sua documentação e adequá-la de acordo com as necessidades.

**9. Defina os seguintes conceitos: (a) Fraco Acoplamento e (b) Alta Coesão.**

**R:** Acoplamento se diz respeito a dependência entre os componentes em uma determinada aplicação. Ou seja, um acoplamento forte quer dizer que essa relação de dependência entre componentes é forte, é necessário muito cuidado ao realizar uma manutenção em componente para que não comprometa outro. De maneira análoga, fraco acoplamento a relação de dependência é fraca. O conceito de coesão está diretamente ligado ao princípio de responsabilidade única, onde um componente com alta coesão possui apenas uma única responsabilidade. Sendo assim, pensando na qualidade e na manutenção, espera-se de um, bom, software fraco acoplamento e alta coesão.

**10. O desenvolvedor de software é aconselhado a sempre separar a interface de um programa (API) da sua implementação. Por que?**

**R:** Um dos principais motivos é a organização e a segurança. Pois, quando você separa suas classes, estruturas, protótipos de funções, regras de negócios, das implementações os códigos ficaram mais organizados contribuindo para usabilidade e manutenção. Aliado a isso, seu arquivo de implementação uma vez separado é possível ser privado.

**11. O que significa reuso de código? Quais as vantagens e desvantagens? Por que é importante?**

**R:** É uma maneira de reutilizar algum código, já implementado em algum outro lugar que realize alguma tarefa parecida ou igual à qual se deseja. A principal vantagem é o ganho em produtividade principalmente se o código estiver bem documentado. A principal desvantagem é má interpretação, dificuldades de integração e má documentação. Sua maior importância dentro de uma empresa é a padronização, e um ganho em produtividade.

**12. Quais são as fases no desenvolvimento de um projeto de software? Quais atividades são realizadas em cada fase?**

**R:** Fase de requisitos, é a fase inicial do projeto onde se determina reconhecer as necessidades do cliente, compreender o problema em questão, qual a finalidade, recursos disponíveis, requisitos funcionais, não funcionais e etc. De uma maneira geral, é o levantamento de requisitos e reconhecimento das principais necessidades.

Fase de projetos, é a fase em que o engenheiro de software identifica os requisitos do sistema e determina qual recurso tecnológico utilizará, projeta as documentações. O diagrama UML - Unified Modeling Language, linguagem de modelagem, é um dos principais recursos e o mais utilizado pelos desenvolvedores.

Fase de implementação, é a fase onde com toda documentação em mãos, é realizado o desenvolvimento do software, propriamente dito, pelos envolvidos.

Fase de testes, é a fase, como o nome diz, realizar testes com a aplicação. Muito importante dessa fase é se colocar no lugar do cliente e usuários.

Fase de produção, é a parte final onde realiza a entrega do produto, ou onde se disponibiliza a aplicação e seus derivados: como algum manual, termos, dicas, tudo para o cliente utilizar o que lhe foi concedido.

**13. Qual a diferença entre verificação e validação de software?**

**R:** Verificação de software está relacionado a análise dos requisitos funcionais e não funcionais a fim de certificá-los. Já a validação é a certificação propriamente dita, de que o sistema atende as necessidades solicitadas. Requisitos funcionais são as funcionalidades do software, as quais

deverá realizar. Os requisitos não funcionais de um software estão ligados ao uso. Como o sistema fará para realizar suas funções, qual será o desempenho, as restrições e os atributos de qualidade, são alguns exemplos de requisitos não funcionais.

**14. Defina cada um dos seguintes níveis de teste de software: (a) teste unitário, (b) teste funcional, (c) teste de integração, (d) teste sistêmico e (e) teste de aceitação.**

**R:** Teste unitário, é um teste realizado em uma determinada função ou método. Preocupando com o funcionamento, suas entradas e saídas.

Teste de integração, é o teste feito entre os componentes.

Teste funcional, é um teste realizado na aplicação preocupando com o comportamento durante a execução e suas saídas. Muito conhecido como caixa-preta.

Teste sistémico, é um teste mais voltado para performance e instalação. Preocupado com a integração entre software, hardware e sistemas operacionais.

Teste de aceitação, é o teste final onde espera-se que a aplicação funcione corretamente, correspondendo às necessidades e as entradas específicas.

**15. O Que é: (a) teste caixa branca, (b) teste caixa preta e (c) teste caixa cinza**

**R:** Teste de caixa-branca é uma técnica de teste que usa a perspectiva interna do sistema para modelar os casos de teste. No teste de software, a perspectiva interna significa basicamente o código fonte. Teste de caixa-preta é um teste de software para verificar a saída dos dados usando entradas de vários tipos. Tais entradas não são escolhidas conforme a estrutura do programa. Quanto mais entradas são fornecidas, mais rico será o teste. O teste de caixa cinza é um processo para depurar aplicativos de software, fazendo uma entrada pelo front-end e verificando os dados no back-end.