O Modelo espiral tem como sua principal característica o reconhecimento explícito do risco, por esse motivo, possui uma etapa baseada na avaliação e análise detalhada para a redução de riscos.

- [x] Verdadeiro
- [ ] Falso

Como os softwares são abstratos e intangíveis, existem restrições naturais que limitam o potencial de um software.

- [ ] Verdadeiro
- [x] Falso

Os requisitos para um sistema de software estabelecem o que o sistema deve fazer e definem as restrições sobre seu funcionamento e implementação.

- [x] Verdadeiro
- [ ] Fals

Softwares são programas de computador e documentação associada.

- [x] Verdadeiro
- [ ] Falso

Os Requisitos podem ser descritos como Requisitos de Usuário ou de Sistema, sendo a principal diferença a quantidade de detalhamento utilizada.

- [x] Verdadeiro
- [ ] Falso

A Engenharia de Software foca todos os aspectos da produção de software, desde os estágios iniciais da especificação do sistema até sua manutenção.

- [x] Verdadeiro
- [ ] Falso

A Engenharia de Software nasceu da necessidade de aperfeiçoamento dos sistemas computacionais em relação ao emprego correto do hardware e tem como foco a Teoria e os Fundamentos da Computação.

- [ ] Verdadeiro
- [x] Falso

Os Requisitos não funcionais podem ser classificados em: Requisitos de Produto, Requisitos Organizacionais e Requisitos Externos.

- [x] Verdadeiro
- [ ] Falso

A Sprint é a unidade Básica do Scrum. Ela é precedida por uma reunião de planejamento chamada Sprint Planning, na qual procura-se identificar quais tarefas do Product Backlog serão executadas. As tarefas identificadas compõem a Sprint Backlog.

- [x] Verdadeiro
- [ ] Falso

Os Requisitos podem ser classificados como Não Funcionais, quando descrevem o que o sistema deve fazer por meio de funcionalidades ou serviços ou como Funcionais, quando estão relacionados às propriedades emergentes do sistema, como usabilidade, confiabilidade, desempenho etc.

- [ ] Verdadeiro
- [x] Falso

O Modelo de Prototipação não pode ser aplicado quando os requisitos ainda não estão bem definidos, pois dificulta os interessados a compreender o que está sendo construído.

- Verdadeiro
- ✓ Falso

Um Processo de Software é uma prescrição rígida de atividades, ações e tarefas, que obrigatoriamente devem ser seguidas e executadas sempre da mesma forma, durante a produção de um software.

- Verdadeiro
- ✓ Falso

São exemplos de métodos ágeis: Scrum, XP, Lean e FDD.

- ✓ Verdadeiro
- Falso

O Modelo Cascata é um modelo predominantemente evolutivo, dado que esse modelo só pode ser empregado quando os requisitos não são bem compreendidos.

- Verdadeiro
- ✓ Falso

A Atividade de Engenharia de Requisitos é o processo de descobrir, analisar, verificar e documentar os requisitos, que são as descrições do que o sistema deve fazer, os serviços que deve oferecer e as restrições relacionadas ao seu funcionamento.

- ✓ Verdadeiro
- Falso

Sobre as quatro atividades fundamentais comuns a todos os Processos de Software, assinale o que for correto.

- [x] Na Especificação, se define o software a ser produzido e as restrições de sua operação.
- [ ] Na Evolução, o software é projetado e programado.
- [ ] No Desenvolvimento (Projeto e Implementação), o software é modificado para refletir a mudança de requisitos do cliente e do mercado.
- [x] Na Validação, o software é verificado para garantir que é o que o cliente quer.

São exemplos de abordagens de Modelos de Processo de Software: Modelo Cascata, Modelo Incremental e Modelo Espiral.

- [x] Verdadeiro
- [ ] Falso

As principais atividades da Engenharia de Software são: especificação de software, desenvolvimento de software, validação de software e evolução de software.

- [x] Verdadeiro
- [ ] Falso

✓ Forneça 3 exemplos (positivos e/ou negativos) que indiquem o impacto do software em nossa sociedade. *10/10

1. Softwares de edição de vídeo e imagem.
2. Softwares de comunicação como o Skype e o Zoom que permitem que as pessoas se comuniquem em tempo real.
3. A dependência das pessoas em aplicativos de redes sociais como o Facebook e o Instagram.

✓ Muitas aplicações modernas mudam frequentemente – antes de serem apresentadas ao usuário e depois da primeira versão ser colocada em uso. Sugira algumas maneiras de construir um software para impedir a deterioração decorrente de mudanças. *20/20

- Ter um planejamento estratégico.
- Usar uma metodologia ágil.
- Atualizar o software regularmente.
- Realizar testes.
- Acompanhar as tendências de mercado.

✓ Explique por que existem ideias fundamentais na engenharia de software que se aplicam a todos os tipos de sistemas. *20/20

Existem porque essas ideias fundamentais permitirão criar um software de alta qualidade, robusto e seguro. Essas ideias fundamentais são essenciais para assegurar que o software funcionará conforme o esperado, que será fácil de manter e atualizar, além de ser escalável e seguro. A aplicação desses princípios de engenharia de software ajuda a evitar erros caros, a manter a qualidade do software e aumentar a satisfação do usuário.

✓ Considere a afirmação a seguir: " Software não é apenas um programa (ou programas); ele inclui também a documentação ". Como ela se relaciona com o atributo essencial de um produto de software manutenibilidade? *20/20

Está relacionado porque sem a documentação clara e completa, os desenvolvedores podem achar difícil entender como o software funciona, quais funções ele realiza, quais são suas dependências e como diferentes partes do software estão integradas.

Além disso, a documentação completa também ajuda outros profissionais, como analistas de negócios, gerentes de projetos e testadores, a entender o software. Isso permite que eles contribuam para a manutenibilidade do software, participando da sua evolução, atualização e melhoria ao longo do tempo.

✓ À medida que o software invade todos os setores, riscos ao público (devido a programas com imperfeições) passam a ser uma preocupação cada vez maior. Pesquise sobre alguma situação da vida real, cuja falha de um programa de computador causou um grande dano (em termos econômicos ou humano) e comente a respeito. *10/10

Acidente com o Boeing 737 MAX em 2019. Nas aeronaves desse modelo, um sistema de estabilização foi introduzido para corrigir uma falha de design. Esse sistema, no entanto, continha uma falha de programação que podia fazer a aeronave entrar em mergulho inesperado. Na queda da aeronave da Lion Air no mar de Java em outubro de 2018, o software foi acionado erroneamente, fazendo com que o avião caísse rapidamente, matando todas as 189 pessoas a bordo. Poucos meses depois, um acidente semelhante com um voo da Ethiopian Airlines matou outras 157 pessoas.

A falha do software nesse caso foi catastrófica, causando a morte de 346 pessoas e interrompendo a operação das aeronaves 737 MAX em todo o mundo. A Boeing tem enfrentado várias ações judiciais e inquéritos desde então e tem trabalhado para corrigir o software. Isso mostra como as falhas do software podem ter consequências de grande alcance e altamente perigosas para o público.

É um acontecimento infeliz que veio para nos lembrar que os softwares que usamos todos os dias podem ter imperfeições que podem ser até fatais. Devemos aprender com os erros para tentar evitar ao máximo que acontecimentos assim se repitam, realizando testes ainda mais rigorosos em sistemas críticos.

✓ "Existem vários tipos diferentes de sistemas, e cada um requer ferramentas e técnicas de engenharia de software adequadas a seu desenvolvimento. Existem poucas, se houver alguma, técnicas específicas de projeto e implementação aplicáveis para todos os tipos de sistemas". Essa afirmação é VERDADEIRA ou FALSA? Justifique sua resposta e apresente um exemplo. *20/20

Verdadeira. Existem vários tipos diferentes de sistemas, e cada um precisa de diferentes abordagens e técnicas específicas para seu desenvolvimento.

Por exemplo, o desenvolvimento de sistemas embarcados podem exigir o uso de uma linguagem de programação de baixo nível, modelos de ciclo de vida de software diferentes e testes altamente rigorosos.

Enquanto que o desenvolvimento de sistemas de software corporativo pode exigir a adoção de metodologias ágeis, colaboração em equipe, modelagem de processos de negócios e testes automatizados.

- **Quais são as Principais Características?**

A metodologia RUP utiliza uma abordagem de orientação a objetos em sua concepção e é projetado e documentado utilizando o UML para ilustrar os processos. O principal objetivo é atender as necessidades dos usuários garantindo uma produção de software de alta qualidade que cumpra um cronograma e um orçamento previsíveis.

Assim, o RUP mostra como o sistema será construído na fase de implementação, gerando o modelo do projeto e, opcionalmente, o modelo de análise que é utilizado para garantir a robustez. O RUP define perfeitamente quem é responsável pelo que, como as coisas deverão ser feitas e quando devem ser realizadas, descrevendo todas as metas de desenvolvimento especificamente para que sejam alcançadas.

Um dos principais pilares do RUP é o conceito de best practices (melhores práticas), que são regras/práticas que visam reduzir o risco (existente em qualquer projeto de software) e tornar o desenvolvimento mais eficiente. O RUP define seis best practices, sendo elas:

- Desenvolver iterativamente;
- Gerenciar requerimentos;
- Utilizar arquiteturas baseadas em componentes;
- Modelar visualmente;
- Verificação contínua de qualidade;
- Controle de mudanças;

- **Quais suas Etapas e Funcionamento?**

O RUP organiza o desenvolvimento de software em quatro fases:

➔ **Concepção**

É nesse momento que é elaborado o planejamento do projeto com os stakeholders. A etapa é realizada em um período de tempo curto e abrange as tarefas de comunicação com o cliente e planejamento. A partir disso, é feito um plano de projeto avaliando os possíveis riscos, as estimativas de custo e prazos, estabelecendo as prioridades, levantamento dos requisitos do sistema e preliminarmente analisá-lo. Ela orienta a equipe a analisar a viabilidade do projeto e como começar a definir os primeiros passos. Assim, são examinados os objetivos para se decidir sobre a continuidade do desenvolvimento. Utilizando esse conceito temos uma metodologia chamada Lean Inception.

➔ **Elaboração**

Abrange a Modelagem do modelo genérico do processo. O objetivo desta fase é analisar de forma mais detalhada a análise do domínio do problema, revisando os riscos que o projeto pode sofrer e a arquitetura do projeto começa a ter sua forma básica. Busca o levantamento de casos, documentação, estudos base, ou seja, modelos para orientar o projeto. Isso ocorre para guiar qual será a melhor forma de acordo com as premissas das partes interessadas. Após todo esse conhecimento, é elaborado um plano do projeto, com todas as características e especificidades, da forma mais detalhada possível.

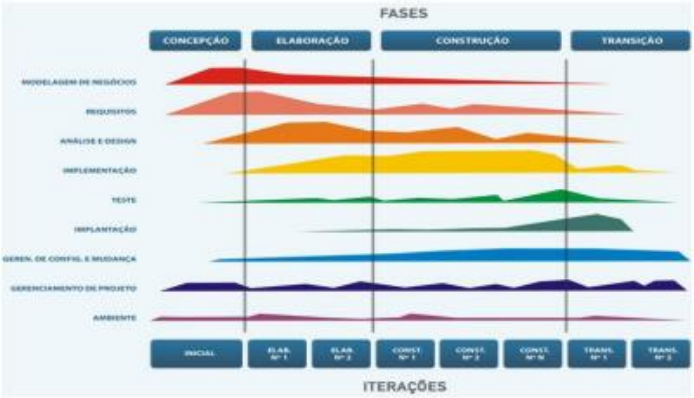
➔ **Construção**

É nesse momento que é feita a construção do projeto. Desenvolve ou Adquire os componentes de Software. O principal objetivo desta fase é a construção do sistema de software, com foco no desenvolvimento de componentes e outros recursos do sistema. É na fase de Construção que a maior parte de codificação ocorre. Além disso, é nessa fase que são realizados os primeiros testes para que a base inicial esteja preparada para a etapa de *transição*.

Abrange a entrega do software ao usuário e a fase de testes. O objetivo desta fase é disponibilizar o sistema, tornando-o disponível e compreendido pelo usuário final. As atividades desta fase incluem o treinamento dos usuários finais e a realização de testes da versão beta do sistema visando garantir que o mesmo possua o nível adequado de qualidade.

Depois de todos os testes realizados e com o objeto pronto, é a hora de disponibilizá-lo para o usuário final, ou seja, a entrega do projeto. Além da entrega, nessa fase inclui a realização dos treinamentos e garantir que o objeto final solucione todos os problemas das partes interessadas.

Visto todas as fases que compõem um projeto utilizando a metodologia RUP, é importante ressaltar que nos desenvolvimentos dessas atividades toda a equipe precisa ser orientada a algumas práticas e a executar os artefatos de forma alinhada.



- **Vantagens e Desvantagens?**

Vantagens:

- Iterativo e Incremental: Cada fase do projeto é dividida em pequenas iterações.
- Orientado a Objetos: Se concentra na modelagem de objetos do mundo real em vez de processos.
- Comunicação: Enfatiza a comunicação entre os membros da equipe e os stakeholders.

Desvantagens:

- Documentação Excessiva: Isso pode aumentar o tempo e o custo do projeto.
- Abordagem Estruturada: Pode não se adaptar bem a projetos que requerem mais flexibilidade e agilidade.
- Dificuldade em Manter Atualizado: Difícil de manter atualizado à medida que novas tecnologias e práticas surgem

- **Como pode ser relacionado com o Desenvolvimento Ágil?**

Tanto o RUP quanto o Desenvolvimento Ágil são baseados em abordagens iterativas e incrementais. Ambos enfatizam a importância de envolver os stakeholders do projeto e trabalhar em equipe. Além disso, o RUP e o Desenvolvimento Ágil podem ser combinados para criar uma abordagem híbrida que aproveita o melhor dos dois mundos.

Por exemplo, é possível usar as práticas de planejamento e gerenciamento de projeto do RUP, juntamente com as técnicas de desenvolvimento ágil, como testes contínuos, desenvolvimento orientado a testes e programação em pares.