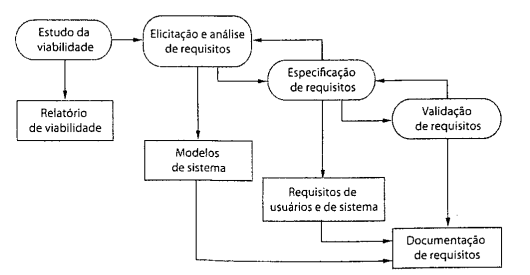
Aluno: Bruno Camargo Manso

1. O que é o estudo de viabilidade e em que fase do ciclo de vida ele deverá ser realizado?

Atividades em um processo são intercaladas com sequências de atividades técnicas. Dentre as diversas atividades exigidas diz respeito à viabilidade. O *estudo de viabilidade* está contido dentro das *atividades de processos,* que se dividem em quatro pontos básicos: especificação, desenvolvimento, validação e evolução .Assim, o *estudo de viabilidade* faz parte das especificações de um processo. É “*(...) uma estimativa acerca da possibilidade de se satisfazerem as necessidades do usuário identificado usando-se tecnologias atuais de software e hardware*.” (Sommerville, 2018). Tal estudo verifica a rentabilidade frente aos custos, no ponto de vista comercial. Gera também restrições orçamentárias, limitando a abrangência do projeto e consequentemente evitando rombos financeiros para empresa desenvolvedora. É rápido e têm um custo relativamente barato, e seu produto final foca na decisão de dar continuidade ou não no devido processo.



*Esquema acima retirado do livro de Ian Sommerville, Engenharia de Softwares, página 24.*

2. É possível realizar o planejamento do desenvolvimento de sistema antes de conhecer os seus requisitos? Por que?

Segundo Sommerville (2018), em muitos projetos, o fator mais importante está relacionado ao custo do projeto pois, certos requisitos podem ser futuramente financiados. Existem casos de empresas que buscam desenvolver um projeto porém não possuem documentos detalhados dos requisitos. Com o orçamento inicial em mãos pode, conforme surgem mais informações sobre o sistema, demandar demais requisitos, assim, se adaptar conforme novos requisitos são levantados.

3. Quais são as diferenças entre o desenvolvimento de um produto de *software* genérico e um desenvolvimento de um produto de *software* personalizado.

Em um *software* genérico é quase certo de que foi previamente personalizado, porém para uma outra empresa, outro cliente… Com isso, a necessidade de ‘remendos’ e novas implementações será certa. Isso acarretará no aumento de custo e consequentemente redução da viabilidade. Um *software* genérico também pode ter funções excessivas desnecessárias para o cliente, gerando problemas técnicos, baixa agilidade, dificuldade em manutenção e nas implementações futuras.

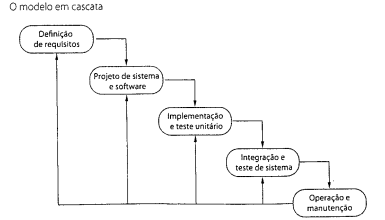
4. Sobre as dificuldades encontradas no desenvolvimento de *software*, explique o que vem a ser a conformidade, complexidade, alterabilidade e a invisibilidade.

Qualquer *Software* carrega em si problemas essenciais. Em 1986 Frederick P. Brooks, em seu livro lançado, faz uma metáfora entre um problema e um Lobisomem e, segundo ele, não existiria bala de prata, ou seja, a solução plena de um problema seria um utopismo. Segundo Brooks existem quatro tópicos em desenvolvimento de *software* que devem ser abordados, são eles:

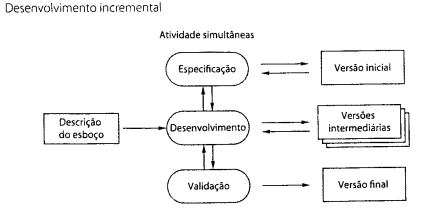
* **Conformidade:** A necessidade de adaptação de um *software* frente às leis que muitas vezes têm características plásticas e pouco definidas. Também deve se adaptar às instituições, diferentes interfaces que também possuem as mesmas características supracitadas, além da mutabilidade inevitável e intrínseca. Afinal, *softwares* são desenvolvidos por pessoas, e não por mitos gregos e semi-deuses.
* **Complexidade:** Costuma-se sintetizar ao máximo as funcionalidades de um *software* a fim de propiciar sua escalabilidade para desafios maiores, por exemplo reduzir funcionalidades duplicadas em apenas uma, ou a junção, não apenas das duplicadas, mas também múltiplas funcionalidades em apenas um só componente. Podemos pensar, em ponto de vista mais amplo, incluindo *softwares* porém indo mais além, na indústria aeroespacial que está sempre em busca de reduzir componentes, aliviar o peso, agregar peças, limpar códigos, otimização no consumo de energia etc.
* **Alterabilidade:** Considera-se sempre um produto passível de atualização, mesmo que sejam produtos finais consumidos pelas pessoas, apoiando ainda no exemplo aeroespacial, por mais difícil que seja o problema, existe uma solução. Como exemplo, o telescópio Hubble que após seu lançamento, ocorreu uma falha em sua lente e assim foi preciso trocá-las em uma missão, por astronautas no espaço. Ocorreram algumas missões com múltiplos objetivos. Além disso o Hubble foi lançado em 86, no entanto seu *software* não fora devidamente concluído a tempo e, apenas em 1990 o Hubble estava com seus *softwares* devidamente configurados. Isso demonstra que, desde uma simples atualização de um aplicativo de smartphone até um veículo espacial necessita dessa capacidade, arcando sempre com seus devidos custos, sendo eles pequenos, ou grandes.
* **Invisibilidade ou intangibilidade:** Para um usuário final um *software* deve se apresentar como uma ‘mágica’, deve mostrar sua interface gráfica porém distanciando tal usuário das linguagens, dos códigos e do funcionamento sistêmico do programa pois, na maioria dos casos, são completamente alheios às práticas computacionais mais avançadas.

5. Diferencie os modelos de processo Incremental/Iterativo e cascata.

O modelo em cascata foi introduzido por Royce(1970), neste, se encadeia uma fase a outra, segundo Sommerville(2018), é um exemplo de projeto dirigido a planos baseado em programação de todas atividades antes de se começar a trabalhar efetivamente nelas. As fases estão descritas como no exemplo abaixo:



O modelo incremental/iterativo, ainda segundo Sommerville(2018) é baseada na ideia de se desenvolver uma implementação inicial e expô-la aos comentários dos usuários sem de fato implementá-la no sistema do cliente, e assim, conforme a demanda, gerar novas versões que serão novamente expostas aos usuários até que um sistema adequado seja desenvolvido. Implementação significa poder utilizá-lo em ambientes reais. É um bom modelo para empresas que não suportam paralisações em seus processos e nem experimentações durante o processo. Abaixo um diagrama sobre suas etapas:



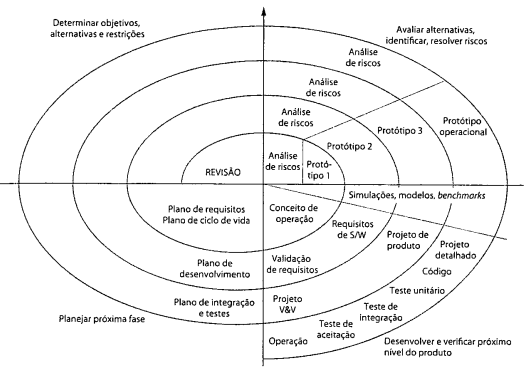
6. Explique o modelo de processo de prototipagem. Quando ele deve ser usado e quais as suas vantagens e desvantagens?

Um protótipo é a versão inicial de um sistema de *software*, demonstrando conceitos, tem a função de experimentar, procurar problemas e encontrar soluções mesmo que seja voltada para interface do usuário final. Essa capacidade ajuda a antecipar mudanças futuramente requisitadas. Possui características rápidas e iterativas de modo que os stakeholders consigam controlar efetivamente seus investimentos. A prototipação também pode ajudar na elicitação e na validação de requisitos. As desvantagens, segundo Sommerville(2018) é que pode ser impossível ajustar o protótipo para atender requisitos como: desempenho, proteção, robustez e confiabilidade que de repente fora foram ignorados no desenvolvimento do protótipo. Mudanças muito rápidas significam baixa qualidade de documentação, péssimo para manutenção a longo prazo. Mudanças durante o desenvolvimento poderão degradar a estrutura do sistema, aumentando o custo, reduzindo a viabilidade. E também pode existir um relaxamento nos padrões de qualidade organizacional ao longo do desenvolvimento do projeto

7. No modelo de processo espiral cada volta da espiral é dividida em quatro setores, cite e explique cada um deles.

O modelo espiral de de processo de *software* de Boehm é dividido em quatro setores básicos:

* Desenvolver e verificar próximo nível do produto;
* Avaliar alternativas, identificar, resolver riscos;
* Determinar objetivos, alternativas e restrições;
* Planejar próxima fase;



8. A perspectiva prática sobre o RUP descreve boas práticas da engenharia de *software* que são recomendadas para o uso no desenvolvimento de sistemas. Quais são as seis boas práticas recomendadas?

O modelo Rational Unified Process (RUP) recomenda as seguintes boas práticas:

* Desenvolver iterativamente;
* Gerenciar requisitos;
* Usar arquiteturas baseadas em componentes;
* Modelar o *software* visualmente;
* Verificar a qualidade do *software*;
* Controlar mudanças no *software*;

9. Scrum é considerado um método ágil geral, mas, ao invés das abordagens técnicas tradicionais específicas da engenharia de *software*, seu foco está no gerenciamento do desenvolvimento iterativo. Ele não prescreve o uso de práticas de programação, como programação em pares e desenvolvimento test-first, mas pode ser utilizado com abordagens ágeis mais técnicas, como XP, para fornecer um framework de gerenciamento

do projeto. O ciclo Sprint, do scrum, pode ser resumido em quatro passos; assinale-os.

a) Avaliar; Desenvolver; Revisar; e, Entregar.

**b) Avaliar; Selecionar; Desenvolver; e, Revisar.**

c) Apresentar; Desenvolver; Revisar; e, Entregar.

d) Apresentar; Selecionar; Desenvolver; e, Revisar.

10. Uma equipe Scrum elaborou uma lista de tarefas que se comprometeu a entregar em

um período definido. A lista de tarefas foi extraída de um produto com base em

prioridades pré-definidas por um profissional.

Um Programador fez as seguintes perguntas:

− Qual é a lista de tarefas?

− Quem define as prioridades?

− Qual a base de extração da lista de tarefas?

− Qual é o período previsto de entrega das tarefas listadas?

As respostas corretas, em sua respectiva ordem, são

**a) Product Backlog − Product Owner − Sprint Backlog − Sprint.**

b) Sprint Backlog − Product Owner − Product Backlog − Sprint.

c) Sprint Backlog − Product Backlog − Product Owner − Sprint.

d) Sprint − Product Backlog − Product Owner − Sprint Backlog.

e) Product Owner − Product Backlog − Sprint Backlog − Sprint.

11. Sobre os princípios do método de desenvolvimento Scrum, que são consistentes com o

manifesto ágil, julgue as seguintes afirmativas e assinale a alternativa correta.

I - Testes e documentação constantes são realizados à medida que o produto é

construído. V

II - O processo produz frequentes incrementos de software que podem ser

inspecionados, ajustados, testados, documentados e expandidos. V

III - O trabalho de desenvolvimento e o pessoal que o realiza é dividido em partições

claras, de baixo acoplamento, ou em pacotes.

a) Apenas as afirmativas I e II são corretas.

b) Apenas as afirmativas I e III são corretas.

c) Apenas as afirmativas II e III são corretas.

**d) Todas as afirmativas são corretas.**

e) Nenhuma das afirmativas é correta.

12. Está sendo implementado o XP (eXtreme Programming) em uma equipe de TI. Para

tanto, está sendo colocada a seguinte série de práticas específicas da metodologia XP em

análise:

I. Programação Pareada (Pair Programming).

II. Fases pequenas (Small Releases).

III. Refatoração (Refactoring).

IV. Jogo de Planejamento (Planning Game).

Com base no seu conhecimento sobre a metodologia citada acima, suas práticas

específicas estão corretamente relacionadas nos itens:

a) I, II e III, apenas

b) I, II e IV, apenas

c) II, III e IV, apenas

d) I, III e IV, apenas

**e) I, II, III e IV**