1. Explique por que software profissional não é apenas os programas que são desenvolvidos para o cliente.
   O software pode conter, além de programas, configurações de outros softwares (interfaces de comunicação, interfaces gráficas, etc de terceiros), softwares completamente replicados no software central (código copiado no software desenvolvido) e toda sua documentação e manutenção.

2. Qual a diferença mais importante entre o desenvolvimento de um produto genérico de software e o desenvolvimento de software sob demanda? O que isso pode significar na prática para usuários de produtos de software genérico?
- O sotware genérico é distribuído no mercado por parte dos desenvolvedores. O sofware sob demanda é feito para um cliente específico que poderá decidir se será distribuído no mercado ou ficará restrito ao uso do cliente.
- Significa que terão de pedir suporte e dar feedbacks diretamente aos desenvolvedores (ou à empresa que desenvolveu o software) e esperar pelas atualizações.

3. Quais são os quatro atributos importantes que todo software profissional deve possuir? Sugira outros quatro atributos que, às vezes, podem ser significantes.
- Manutenibilidade, confiança e proteção, eficiência e aceitabilidade.
- Segurança, usabilidade, adequabilidade e desempenho.

4. Explique por que existem ideias fundamentais na engenharia de software que se aplicam a todos os tipos de sistemas.
   Porque softwares precisam ser fáceis de serem consertados e rapidamente melhorados para diminuir os custos com suporte e para torná-lo sempre atualizado e otimizado.

5. Explique como o uso universal da Internet mudou os sistemas de software.
   Os softwares começaram a dar suporte a sistemas que funcionam na internet em aplicativos ou browser. Também trouxe a necessidade de implementar novas camadas de implementação para se comunicarem com protocolos de internet de forma eficiente. Também facilitou na distribuição do próprio software e de correções e otimizações nele.

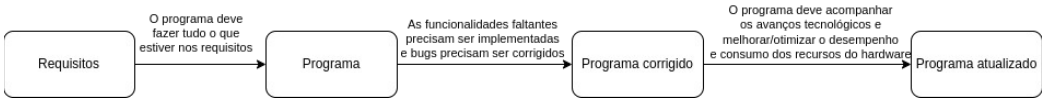
6. Justificando sua resposta com base no tipo de sistema a ser desenvolvido, sugira o modelo genérico de processo de software mais adequado para ser usado como base para a gerência do desenvolvimento dos sistemas a seguir:
- Um sistema para controlar o antibloqueio de frenagem de um carro.
  Embarcado.
- Um sistema de realidade virtual para dar apoio à manutenção de software.
  Aplicativo ou de uso pessoal.
- Um sistema de contabilidade para uma universidade, que substitua um sistema já existente.
  Científico e de engenharia.
- Um sistema interativo de planejamento de viagens que ajude os usuários a planejar viagens com menor impacto ambiental.
  Aplicativo ou de uso pessoal.

7. Explique por que o desenvolvimento incremental é o método mais eficaz para o desenvolvimento de sistemas de software de negócios. Por que esse modelo é menos adequado para a engenharia de sistemas de tempo real?

- Porque os requisitos vão mudando e feedbacks vão chegando, necessitando uma constante revisão e modificação do software.
- Não é adequado para sistemas de tempo real por terem requisitos bem definidos, não dinâmicos e por geralmente serem empregados em sistemas críticos que acabariam perdendo o controle do cumprimento do tempo real ao longo dos incrementos e também dificuldade para validar o sistema ao longo das versões.

8. Descreva as principais atividades do processo de projeto de software e as saídas dessas atividades. Usando um diagrama, mostre as possíveis relações entre as saídas dessas atividades.

- Especificação: define o que o programa deve fazer. A saída são os requisitos.
- Desenvolvimento: construção (implementação) do programa. A saída é o programa.
- Verificação e validação: testar o programa para solucionar possíveis erros e conferir com o cliente, se o programa faz tudo que ele pediu. A saída é o programa corrigido.
- Manutenção e evolução: atualizar o programa para acompanhar os avanços tecnológicos. A saída é o programa atualizado.



9. Explique por que, em sistemas complexos, as mudanças são inevitáveis. Exemplifique as atividades de processo de software que ajudam a prever as mudanças e fazer com que o software seja desenvolvido mais tolerante a mudanças (desconsidere prototipação e entrega incremental.

- O cliente nem sempre contará tudo que vai precisar no programa. Algumas coisas ele só vai achar quando começar a sair as versões do programa. Por exemplo: “Essa tela poderia ter isso aqui, também” ou “Isso aqui seria melhor se tivesse isso ou aquilo”.
- O modelo espiral repete os passos (comunicação, planejamento, modelagem, construção e implantação) até entregar o programa final. Na comunicação, os desenvolvedores e clientes irão conversar sobre o que o programa fará ou o que foi feito e falta fazer; no planejamento, os desenvolvedores analisarão qual plano seguir na implementação, o que vão fazer primeiro, a ordem das coisas a serem feitas; na modelagem, os desenvolvedores irão montar uma estrutura de alta abstração para entender melhor o programa que será feito e como será organizado; na construção, os desenvolvedores irão fazer o programa, escrevendo o código; e na implantação, os desenvolvedores entregam ao cliente, o que foi feito.

10. Explique por que os sistemas desenvolvidos como protótipos normalmente não devem ser usados como sistemas de produção.
    Porque os protótipos têm o objetivo de entender melhor o programa a ser construído. É um negócio não funcional, apenas visual/ilustrativo.

11. Explique por que o modelo em espiral de Boehm é um modelo adaptável, que apoia tanto as atividades de prevenção de mudanças quanto as de tolerância a mudanças. Na prática, esse modelo não tem sido amplamente usado. Sugira as possíveis razões para isso.

- O modelo espiral é um modelo cascata iterativo. A prevenção de mudanças ocorre ao seguir um fluxo definido como no modelo cascata, iniciando com a definição dos requisitos e terminando na entrega do produto (nesse caso, parte do produto). A tolerância a mudanças é feita durante as iterações, onde cada seção da espiral se repete (seria, novamente, da definição de requisitos até à entrega).
- Por ter uma parte do modelo cascata, o escopo do modelo espiral diminui, pois parte do processo de desenvolvimento passa por um modelo não ágil, mais demorado, o que pode atrasar o cronograma.

12. Explique por que, para as empresas, a entrega rápida e implantação de novos sistemas frequentemente é mais importante do que a funcionalidade detalhada desses sistemas.
    Para mostrar ao cliente, algo “concreto”/funcional. Desta forma, ele consegue avaliar se era isso mesmo que ele queria, o que muitas vezes não seria possível de ser avaliado em um grande documento, mesmo sendo o mais detalhado possível.

13. Explique como os princípios básicos dos métodos ágeis levam ao desenvolvimento e implantação de software acelerados.
    O foco no código ao invés do projeto, leva ao desenvolvimento de funcionalidades do programa final que, a partir daí, leva à definição de seus respectivos requisitos e documentações, todos feitos juntos e de forma dinâmica. Então atualiza-se o código, junto com os requisitos e documentações.

14. Quando você não recomendaria o uso de um método ágil para o desenvolvimento de um sistema de software?
    Quando o software ou a equipe é grande. Quando os clientes exigem algo mais elaborado e bem documentado.

15. Extreme Programming expressa os requisitos dos usuários como histórias, com cada história escrita em um cartão. Discuta as vantagens e desvantagens dessa abordagem para a descrição de requisitos.
- Facilita muito o entendimento do sistema e a programação das partes do programa de forma mais clara e rápida. Os requisitos estão claramente definidos nessas histórias e as prioridades são facilmente definidas em quais partes do programa devem ser feitas primeiro.
- Por serem compactas, as histórias abstraem muito os dados técnicos de cada requisito, abrindo margem para várias adaptações no código que podem prejudicar o funcionamento do programa.

16. Sugira quatro razões pelas quais a taxa de produtividade de programadores que trabalham em pares pode ser mais que a metade da taxa de produtividade de dois programadores que trabalham individualmente.
    a. Um programador monitora o desempenho do outro, apontando possíveis erros que o outro não enxerga;
    b. Os dois se suportam intelectualmente, diminuindo a carga de estresse em somente um, consequentemente diminuindo o cansaço;
    c. Ambos trabalham suas ideias até chegarem em uma solução ótima, muito mais eficiente que cada um tinha anteriormente;
    d. Dependendo da ferramenta, os dois podem trabalhar no mesmo arquivo, em paralelo, aumentando a velocidade de desenvolvimento.

17. Tem-se sugerido que um dos problemas de se ter um usuário participando de uma equipe de desenvolvimento de software é que eles ‘se tornam nativos’, ou seja, adotam a perspectiva da equipe de desenvolvimento e perdem de vista as necessidades de seus colegas usuários. Sugira três maneiras de evitar esse problema e discuta as vantagens e desvantagens de cada abordagem.

- Não entrar nos detalhes do processo de desenvolvimento do programa: o usuário não precisa saber dos mínimos detalhes, como se ele fosse virar um programador. Basta saber dos níveis mais abstratos do sistema. Ele pode perder algumas partes sobre o funcionamento de certas coisas, mas tudo depende de um equilíbrio.
- Não chamá-lo frequentemente: participando de todas as reuniões da equipe de desenvolvimento pode fazer o usuário ter muitas informações desnecessárias para ele, o que acaba alterando sua percepção do programa. É necessário saber em quais reuniões é necessário chamá-lo, de forma que ele não fique nem muito por dentro do projeto, nem desinformado sobre seu progresso.
- Realizar outras formas de interação com o usuário: mostrar ao usuário, as telas que foram feitas com suas respectivas funcionalidades seria o suficiente para deixá-lo informado. Deixar o usuário mexer no sistema para ele se adaptar ao estilo do programa que está sendo desenvolvido e sugerir outras coisas.