1. (1,0 ponto) O que significa “domínio” no contexto dos sistemas de software? Por que é tão importante conhecê-lo para desenvolver software?
R: No contexto dos sistemas de software, o termo “domínio” refere-se ao conjunto de conhecimentos, regras e conceitos relacionados a uma área específica de negócio ou atividade em que o software será utilizado.

Exemplo: se estivermos desenvolvendo um software para uma empresa de finanças, é necessário ter conhecimento sobre conceitos financeiros, como contabilidade, impostos, investimentos e fluxo de caixa. Se o software for para uma empresa de logística, é importante ter conhecimentos sobre rotas de transporte, tipos de veículos, armazenamento de mercadorias, etc.

2. (1,0 ponto) Por que o modelo de domínio não é simplesmente a representação do conhecimento repassado pelos especialistas de domínio?
R: O conhecimento dos especialistas de domínio é importante para a criação do modelo de domínio, mas não é suficiente para produzir um modelo preciso e efetivo, pois pode ser subjetivo, incompleto e impreciso. É necessário validar e verificar esse conhecimento com outras fontes de informação para garantir a precisão do modelo de domínio.

3. (1,0 ponto) No exemplo do sistema bancário, apresentado em aula, foi mencionado que o sistema precisa conhecer o endereço do cliente, mas não deve se preocupar com a cor dos olhos dele. Dê um exemplo, não mencionado em aula, em que a decisão de incluir ou não uma informação neste sistema pode não ser tão óbvia.
R: Um exemplo em que a decisão de incluir ou não uma informação pode não ser tão óbvia é o caso de um sistema de gestão de recursos humanos que permite que os funcionários façam solicitações de licença médica. Nesse caso, uma informação importante a ser coletada é a razão da licença médica, que pode ajudar a empresa a identificar problemas de saúde comuns entre os funcionários e desenvolver medidas preventivas.

4. (1,0 ponto) O Domain-Driven Design (DDD) é um tipo de modelo de processo de desenvolvimento, como o modelo cascata ou o Extreme Programming, ou é uma abordagem que pode ser usada para auxiliar esses modelos de processo? Com quais atividades do processo de desenvolvimento de software o DDD se preocupa predominantemente?

R: O DDD é uma abordagem que pode ser usada para auxiliar diferentes modelos de processo de desenvolvimento de software, como o modelo cascata, o Extreme Programming (XP) e o Scrum.
O DDD se preocupa predominantemente com as atividades relacionadas à compreensão e modelagem do domínio do negócio em questão. Isso inclui a identificação dos conceitos do domínio, a definição de limites de contexto, a criação de modelos conceituais e a implementação de código que reflita esses modelos.

5. (3,0 pontos) Descreva como se daria o processo de construção do conhecimento para desenvolver um sistema de gerenciamento da rede externa de telefonia fixa de uma operadora. Para isso, utilize diagramas similares aos apresentados na aula. Demonstre como os diagramas vão se expandindo conforme se aprofunda no conhecimento do domínio. Sugestões para conhecer o domínio: https://wiki.sj.ifsc.edu.br/images/6/6d/Apostila_Rede_Externa_Prof._Saul_%282010%29.pdf
https://wiki.sj.ifsc.edu.br/images/1/1c/Introduçãoredeexterna.pdf (a partir do slide 10)

6. (1,0 ponto) Qual a importância da linguagem ubíqua na construção de um modelo de domínio? Quais problemas podem ocorrer quando não há uma linguagem comum durante essa construção?
R: A importância da linguagem ubíqua na construção de um modelo de domínio está relacionada ao fato de que ela permite uma comunicação clara e precisa entre todas as partes interessadas no projeto. Isso evita ambiguidades e garante que todos tenham uma compreensão comum dos conceitos, processos e regras de negócio envolvidos.

Quando não há uma linguagem comum durante a construção de um modelo de domínio, podem ocorrer diversos problemas. Por exemplo:

- Dificuldade de comunicação: sem uma linguagem compartilhada, as equipes de desenvolvimento e especialistas de domínio podem ter dificuldade em se comunicar, o que pode levar a mal-entendidos e atrasos no desenvolvimento do software.

- Ambiguidade: sem uma linguagem comum, pode haver diferentes interpretações dos conceitos de negócio, levando a diferentes implementações e inconsistências no software final.
- Desalinhamento entre o software e o negócio: se o modelo de domínio não refletir adequadamente o negócio, o software resultante pode não atender às necessidades dos usuários e dos clientes.

Portanto, a linguagem ubíqua é fundamental para garantir uma compreensão comum e clara dos conceitos de negócio envolvidos e para evitar problemas de comunicação, ambiguidade e desalinhamento entre o software e o negócio.

7. (1,0 ponto) Ao usar termos diferentes para um mesmo elemento ou conceito, ficamos sujeitos a vários problemas de comunicação, os quais se refletirão posteriormente no software em desenvolvimento. Por exemplo, em um sistema para gerenciar um hospital podemos chamar as pessoas que procuram seus serviços de “paciente” e de “cliente”, dependendo do que estamos descrevendo no sistema. Para o módulo de consultas médicas, a pessoa é um “paciente”, mas para o módulo de pagamentos, a pessoa poderia ser considerada um Instruções para entrega das listas de atividades: Meio de Entrega: Os arquivos das resoluções das listas de exercícios devem ser entregues exclusivamente no ambiente Moodle (http://eadcampus.spo.ifsp.edu.br). Não serão aceitos links para repositórios externos. Forma de Entrega: As respostas das questões devem ser entregues em um único arquivo PDF, cujo nome deverá conter a informação da aula, o nome e o sobrenome do aluno. Por exemplo: Aula2_MariaPereira.pdf. Prazo de Entrega: O prazo de entrega está definido na página da atividade no Moodle, lembrando que o sistema bloqueia o envio de arquivos após a data e horário indicados. Obs.: A resolução deste(s) exercício(s) deve ser feita de forma INDIVIDUAL. Listas de exercícios com uma ou mais respostas idênticas serão desconsideradas integralmente para efeitos de nota de participação. “cliente”. O ideal seria escolher um dos dois termos para nomeá-lo no modelo, ou adotar algum outro termo genérico. Cite um outro exemplo, similar a este, para outro sistema.

8. (1,0 ponto) Retomando o Exercício 5, crie uma linguagem ubíqua para o domínio de rede externa de telefonia fixa. Para isso, faça um pequeno

glossário com os oito termos que você acha mais importantes nesse domínio, considerando que eles representam conceitos-chave para desenvolver o sistema de gerenciamento da rede externa. Obs.: Embora, muitas vezes, os termos usados pelos especialistas de domínio acabem sendo escolhidos como parte da linguagem comum, os especialistas de software devem estar atentos para criar termos (ou propor termos mais adequados). Veja o termo “fix” proposto pelo desenvolvedor no exemplo da aula. Pense nisso ao criar o glossário!