

**Exame de Análise e Modelação de Sistemas – Recurso | 2019-01-31 15h00. Duração: 90min.**

NOME:

NR. MEC:

Questões de escolha múltipla: **responda na grelha**, assinalando uma opção por pergunta (pretende-se a opção verdadeira e, havendo várias que possam ser consideradas verdadeiras, pretende-se a mais específica para o enunciado dado); as não-respostas valem zero; **respostas erradas descontam**  $\frac{1}{4}$  da cotação; as respostas assinaladas de forma ambígua serão consideradas não-respostas.

Questões 22 e 23: responder no espaço vazio, no final do enunciado.

Grelha de respostas da escolha múltipla (perguntas 1 a 21):

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	
a)																						a)
b)																						b)
c)																						c)
d)																						d)
e)																						e)

**P1.**

Qual o papel característico do Analista numa equipa de desenvolvimento?

- a) É o representante dos interesses dos *stakeholders* do projeto, na definição dos requisitos do produto.
- b) É um profissional especializado no domínio do problema em que o projeto se situa e, por isso, compreende bem o funcionamento do negócio.
- c) **Analisa os processos da organização para identificar oportunidades de melhoria, e delineia o sistema de informação que as implementa.**
- d) Apresenta uma competência técnica acima da média, o que lhe permite fazer as escolhas tecnológicas da implementação.
- e) Assegura a gestão diária do projeto e o diálogo com o cliente.

**P2.**

Wieggers caracteriza a determinação de requisitos como um exigente desafio de interação humana. Porquê?

- a) A determinação de requisitos é propensa a equívocos e origina resultados incompletos.
- b) As conversas face-a-face são o método de determinação de requisitos por eleição.
- c) Os requisitos devem principalmente ser deduzidos de cenários de utilização.
- d) A descrição de situações concretas de interação pessoa/sistema (*user stories*) são decisivas para a construção de uma especificação de requisitos.
- e) **As tecnologias, por si só, não resolvem a necessidade de desenvolver uma cooperação eficiente com o cliente.**

**P3.**

O método *Unified Process* prevê quatro fases principais no desenvolvimento do projeto, cada qual com objetivos e atividades próprios. Neste contexto:

- a) A *Inception* inclui a análise detalhada dos casos de utilização, para mitigar os riscos de má compreensão das necessidades.

- b) **A *Elaboration* deve implementar as partes críticas necessárias para confirmar a viabilidade da arquitetura.**
- c) A *Elaboration* é facultativa na maior parte de projetos de implementação de aplicações para a Web.
- d) Na *Construction*, o analista deve detalhar o modelo da base de dados (se necessário).
- e) Na *Construction*, a solução deve ser instalada no ambiente de produção e aceite pelo cliente.

**P4.**

Os requisitos devem apresentar as características conhecidas por S.M.A.R.T. Identifique, na lista, um requisito funcional adequadamente formulado.

- a) “O sistema deve permitir a transferência de saldo entre contas do mesmo Investigador.”
- b) **“O utilizador deve poder consultar todos os seus dados pessoais, sempre que quiser.”**
- c) “Usar um ambiente Web, compatível com os *browsers* (produto e versão) que representam 85% do mercado.”
- d) “A renovação do aluguer tem de ser realizada até 24 horas antes de terminar o tempo já contratado.”
- e) “O Portal deve integrar com todos os sistemas de micro-pagamentos *online* existentes.”

**P5.**

Na terminologia dos projetos de desenvolvimento ágil, o que é a velocidade da equipa (numa iteração)?

- a) O número de submissões para o repositório partilhado (*commits*), por iteração.
- b) O número de histórias (*user stories*) completadas e aceites por iteração.
- c) O número de entregas (de incrementos à solução) feitas ao cliente, e aceites.
- d) **Os pontos acumulados das histórias implementadas e aceites, por iteração.**
- e) O número de critérios de aceitação verificados e aceites pelo cliente.

## P6.

Os sistemas de software podem ser caracterizados de acordo com diferentes perspetivas de modelação. Um modelo funcional...

- Trata o sistema sob especificação como uma “caixa opaca”, especificando o comportamento observável “de fora”.
- Recorre a modelos de casos de utilização, mas sem detalhar os cenários, para evitar a especificação interna dos módulos na fase de Análise.
- Utiliza, principalmente, diagramas de sequência para caracterizar a troca de mensagens entre objetos que cooperam.
- Utiliza principalmente diagramas de classes, privilegiando a definição dos métodos (ou funções), em detrimento dos atributos (estado).
- Explica como é que um caso de utilização é realizado, isto é, a interação que é necessária entre as entidades de software para implementar o cenário de utilização.

## P7.

A arquitetura trata da tomada das grandes decisões técnicas em relação ao sistema a desenvolver. Um exemplo de uma decisão de arquitetura é:

- O desenho de classes de código que maximizam a coesão e minimizam a interdependência.
- A distribuição de atributos e métodos pelas classes.
- Os casos de utilização que são incluídos no âmbito do projeto.
- A tipologia de plataformas de utilização que devem ser suportadas (web, Android, etc.).
- A ferramenta CASE que a equipa vai adotar.

## P8.

O mesmo tipo de diagrama da UML pode ser usado para criar modelos com diferentes perspetivas de análise, em diferentes fases do SDLC, como por exemplo o \_\_\_\_, usado na fase de Análise para representar \_\_\_\_ e na fase de Desenho para representar \_\_\_\_.

- Diagrama de atividades/ os atores/ partições dos dados.
- Diagrama de objetos/ conceitos do domínio/ métodos de cada classe.
- Diagrama de pacotes/ a arquitetura / os componentes.
- Diagrama de classes / sistemas externos / a implementação interna.
- Diagrama de sequência / colaboração entre o sistema sob especificação e sistemas externos / colaboração entre objetos.

## P9.

É preciso documentar o protocolo de interação entre uma aplicação móvel e um dispositivo médico Bluetooth, de modo a clarificar as mensagens que devem ser trocadas ao longo do tempo. Que modelo UML pode ser utilizado para isso?

- Um diagrama de instalação, clarificando os nós que intervêm.
- Um diagrama de sequência, com ativações correspondentes aos intervenientes no protocolo.

- Um diagrama de atividades, para caracterizar as responsabilidades dos atores.
- Um diagrama de classes, em que o dispositivo deve estender a interface Bluetooth.
- O Diagrama de Casos de Utilização, para identificar os usos possíveis do sistema.

## P10.

Durante as atividades de implementação, o programador deve ter em conta os padrões de desenho de software (*software design patterns*). O que são os padrões de desenho?

- Soluções recomendadas para problemas recorrentes na programação por objetos.
- Orientações para distribuir corretamente a responsabilidade de instanciar objetos.
- Princípios para diminuir a interdependência entre objetos (*coupling*).
- Boas práticas para organizar visualmente os modelos UML em que há a modelação de entidades de software.
- Regras para avaliar a qualidade do desenho de uma implementação (por objetos).

## P11.

Um dos princípios “SOLID” afirma que as entidades de software devem estar abertas a extensões, mas fechadas para modificações. O que é isto significa, na prática?

- Uma classe, depois de implementada, não pode ser modificada.
- Construir classes coesas, que apresentam uma única responsabilidade.
- Para incrementar a funcionalidade de uma entidade, é preferível criar novo código, do que editar a implementação que já existe.
- Manter reduzido o número de classes que implementam uma interface.
- A criação de subclasses tem impacto na interação dos objetos já existentes e, por isso, deve ser evitada.

## P12.

As histórias do utilizador (*user stories*) podem ser usadas para montar uma estratégia de garantia de qualidade do software:

- Os cenários explorados nas histórias são usados para criar testes de aceitação sobre a web.
- As histórias identificam *personas* que podem validar (aceitar) a implementação.
- As histórias descrevem os objetivos que as *personas* pretendem realizar no sistema, utilizando o modelo “Sendo um...Quero [realizar]...De modo a...”.
- As histórias utilizam exemplos para descrever cenários de utilização, que constituem as condições de aceitação do incremento.
- As histórias incluem cenários concretos, representativos das situações de sucesso.

## P13.

Qual das seguintes sequências de passos deve ocorrer num processo de Integração Contínua?

- Entrega de código (*commit*) pelo programador, testes de aceitação, testes de integração.
- Entrega de código (*commit*), testes unitários automáticos, testes de aceitação automáticos, instalação em produção.
- Entrega de código (*commit*), resolução de dependências e compilação no ambiente de integração, execução dos testes automáticos, visão partilhada do estado da *build*.
- Deteção de alterações ao código na *workstation* do programador, execução de testes automáticos, instalação no ambiente de pré-produção.
- Testes à cabeça, deteção de novo código no repositório, correção automática dos erros.

## P14.

Relativamente ao Diagrama 1, o que se pode ver quanto aos atores modelados?

- A Base de Dados é parte do sistema sob especificação, logo não é um ator.
- A Base de Dados é um sistema, logo não é um ator.
- O Gestor dos espaços deve especializar o Utente, pois também pode beneficiar do aluguer de espaços.
- Faltam associações entre a Base de Dados e todos os casos de utilização (de alguma forma leem ou escrevem na BD).
- Não há problemas a assinalar; os atores são adequados.

## P15.

No Diagrama 1, as duas situações de <<include>> modeladas:

- Refletem a dependência temporal dos casos de utilização (e.g.: Procurar não pode ser feito antes do Adicionar)
- Estão mal aplicadas: o caso de utilização “incluído” não deve ter associação direta com atores.
- Servem para evidenciar a necessidade da intervenção dos vários atores naqueles cenários.
- Estão incompletas: faltam os respetivos pontos de extensão (“*extension points*”).
- Estão mal aplicadas: os cenários em causa têm objetivos diferentes e são autónomos entre si.

## P16.

Como é que diagramas do género do Diagrama 1 são utilizados ao longo do SDLC?

- Podem ser usados para detalhar/suplementar os conceitos identificados no modelo do domínio.
- Podem ser detalhados/suplementados com diagramas de sequência
- Podem ser substituídos por diagramas de atividades, em que há partições correspondentes aos atores.
- Os métodos ágeis de desenvolvimento privilegiam a comunicação sobre a documentação e não recomendam o uso dos casos de utilização.
- São usados apenas na fase de análise do sistema, para explorar requisitos funcionais.

## P17.

Considere o modelo do Diagrama 2:

- Um Equipamento encontra-se em uso em vários Complexos desportivos.
- Só as Entidades de um Município podem operar/gerir Piscinas.
- Um Utente pode, numa Reserva, incluir vários Equipamentos.
- Um Complexo Desportivo destina-se à prática de uma Modalidade desportiva.
- A Disponibilidade semanal (períodos de funcionamento) de um Equipamento é igual ao longo das várias semanas.

## P18.

Segundo o Diagrama 2, o que é que um Utente reserva?

- A utilização de um Complexo Desportivo, por um período de tempo.
- A utilização de Uma Piscina municipal, por um período de tempo.
- A utilização de um Equipamento desportivo, para realizar uma Modalidade bem definida.
- A utilização de um Equipamento desportivo, por um período de tempo.
- A utilização de um Equipamento desportivo, com um custo dependente da Modalidade desportiva.

## P19.

Que alterações ao Diagrama 2 seriam necessárias para que o modelo tivesse a capacidade expressiva para representar o requisito “O custo hora de um Equipamento depende da Modalidade que o vai usar”?

- Nenhuma (já é possível representar corretamente essa informação).
- O atributo custo, que existe em Equipamento, deve ser movido para a classe Modalidade.
- O atributo custo deve ser movido para uma classe de associação (entre Equipamento e Modalidade).
- Deve ser acrescentado um atributo em Equipamento para representar a Modalidade.
- O atributo custo em Equipamento pode ser descartado, porque é redundante com o que existe em Reserva de espaço.

## P20.

O Diagrama 3 representa o fluxo de trabalho orientado por Histórias (“*user stories*”), em que:

- A criação de Histórias leva à atualização do *Backlog*.
- A aplicação dos controlos automáticos de garantia de qualidade não aborta o circuito de tratamento da História.
- O Dono do produto pode rejeitar uma História, que retorna ao Programador.
- O circuito de tratamento da História não é cancelado por eventos externos.
- Todas as hipóteses anteriores estão corretas.

**P21.**

Em que ponto(s) da atividade modelada no Diagrama 3 seria de esperar que se utilizasse, como *input*, os resultados (entidades) do Diagrama 1?

- Para escrever a “user story”.
- Para entregar a implementação.
- Para aceitar a “user story”.
- Todas as alíneas anteriores.
- Na verdade, a atividade modelada no Diagrama 3 não recorre às entidades modeladas no Diagrama 1.

**P22.[questão de desenvolvimento]**

“As empresas agora operam num ambiente que está a mudar de forma incrivelmente rápida. Novos produtos aparecem e desaparecem, as leis e regulamentos mudam, as empresas fundem-se e reestruturam-se. Os novos pacotes de software têm de ser rapidamente concebidos, implementados e entregues. Não há tempo para processos de engenharia de requisitos prolongados. O desenvolvimento começa logo que uma visão para o software está disponível; os requisitos emergem e são clarificados durante o processo de desenvolvimento.” In: Sommerville, I. (2005). Integrated requirements engineering: A tutorial. *IEEE software*, 22(1), 16-23.

A citação de alguma forma passa a ideia que o desenvolvimento de software precisa de se adaptar ao novo ritmo (acelerado) com que acontecem os negócios.

O que pode ser feito para adequar as metodologias da engenharia de software ao “novo” contexto?

**P23. [questão de desenvolvimento]**

Considere o trecho de código seguinte, em Java, com omissões.

- Apresente um diagrama de classes para visualizar a informação estrutural que se pode depreender deste código.
- Apresente um diagrama de sequência para representar a interação entre objetos que ocorre quando é invocado o método `WeatherRepository#refreshForecastIfNeeded`.

```

9  /**
10   * Manages the requests to get the weather forecast. If the local data
11   * is still recent, no remote requests are made. Otherwise, the IPMA's
12   * API is invoked.
13   */
14  public class WeatherRepository {
15      private static final String BASE_URL = "https://api.ipma.pt/open-data/";
16      private static final int REFRESH_PERIOD = 300;
17
18      private static RemoteWeatherAPI apiService = new IpmaApiClient( BASE_URL);
19      private MyDatabase localDb;
20
21      public void refreshForecastIfNeeded( int placeId, Date day) {
22          boolean goodLocalData = localDb.hasUpdatedForecast(placeId, day, REFRESH_PERIOD);
23          if (! goodLocalData) {
24              WeatherForecast forecast = null;
25              try {
26                  forecast = apiService.getForecastForPlace( placeId);
27                  if (forecast != null) {
28                      localDb.save(forecast);
29                  }
30              } catch (IOException e) { e.printStackTrace(); }
31          }
32      }
33  }

```

## Folha de Diagramas

UML Paradigm Standard (Universidade de Aveiro)

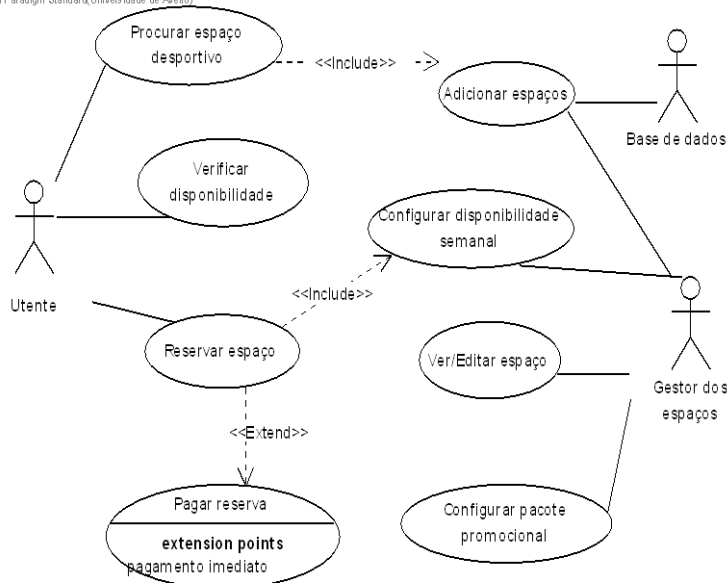


Diagrama 1- Cenários de utilização associados a um sistema de reserva de espaços desportivos.

UML Paradigm Standard (Universidade de Aveiro)

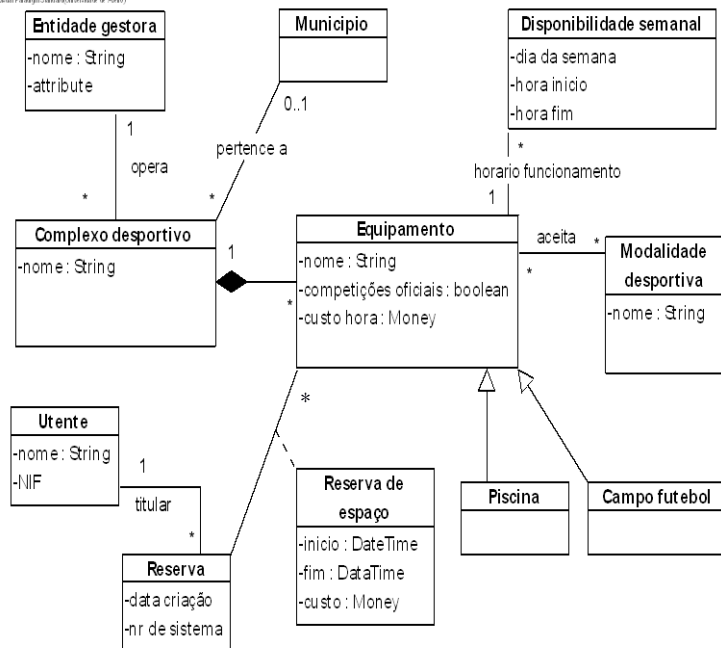


Diagrama 2 – Representação parcial dos conceitos associadas à gestão de alugueres de espaços desportivos.

Visual Paradigm Standard (Universidade de Aveiro)

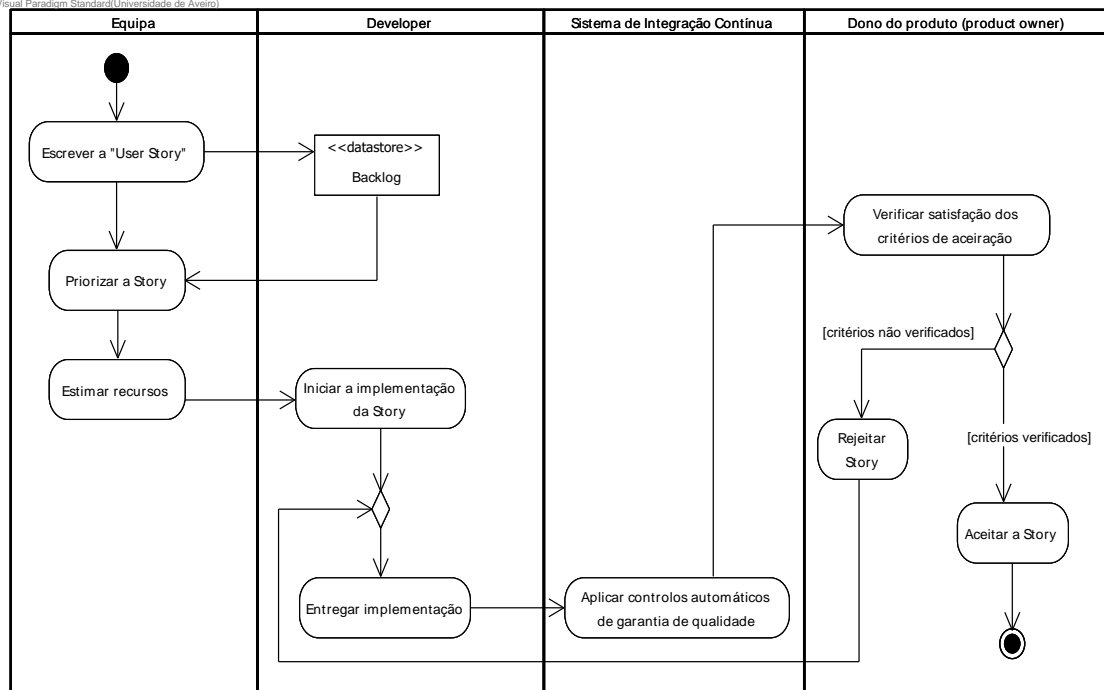


Diagrama 3- Processo de trabalho numa equipa de desenvolvimento, recorrendo a “user stories”.

### **Questões de desenvolvimento**

NOME:

NR. MEC:



