

CES-28 Exame - 2017

Com consulta - individual - com computador - 3h

Obs.:

1. Para a resolução da prova, podem ser usados sites de ajuda online para procurar exemplos da sintaxe para os testes, e o próprio material da aula com pdfs, exemplos de código e labs, inclusive o seu código, mas sem usar código de outros alunos.
2. Questões com itens diversos, favor identificar claramente (inserir comentário) pela letra que representa o item, para que se saiba precisamente a que item corresponde a resposta dada!
3. Só precisa implementar deve ser usado o Eclipse, onde essas questões ou itens serão indicados com o rótulo **[IMPLEMENTAÇÃO]**! Para as outras questões, você pode usar o Eclipse caso se sinta mais confortável digitando os exemplos, mas não precisa de um código completo, executando. Basta incluir trechos de código no texto da resposta.
4. **Submissão -> VIA TIDIA e GITHUB**
 - a. Código completo e funcional da questão **[IMPLEMENTAÇÃO]** - deve ser gerado um arquivo zip ou rar contendo o projeto eclipse, assim como todas as bibliotecas;
 - b. Arquivo PDF com respostas, código incluso no texto para as outras questões. Use os números das questões para identificá-las.
 - c. **Arquivo texto contendo nome completo e repositório GITHUB.**
5. No caso de diagramas, vale usar qualquer editor de diagrama, e vale também desenhar no papel, tirar foto, e **incluir a foto no pdf dentro da resposta, não como anexo separado**. Atenção: use linhas grossas, garanta que a foto é legível!!!!

UNMANNED AIRCRAFT SYSTEM (UAS) TRAFFIC MANAGEMENT (UTM)

A inserção de aeronaves remotamente pilotadas (ARP), ou como elas são mais conhecidas - drones, em espaços aéreos urbanos é uma realidade que estará presente em um futuro muito próximo. Várias empresas ao redor do mundo vêm buscando forçar os órgãos reguladores, responsáveis pelo gerenciamento e controle do espaço aéreo, a se posicionarem e apresentarem quais os rumos que a normatização irá tomar, visando dessa forma, garantir o entendimento de quais são as tecnologias candidatas, bem quais são os critérios de operação e padrões de segurança a serem adotados.

Uma vez que operações de pequenos drones já são bastante comuns de serem vistos nas cidades, principalmente em missões de transmissão de imagens reais de grandes acontecimentos, se faz necessário caracterizar essa porção do espaço aéreo, definindo, por exemplo, quais os critérios de acesso a esse espaço, buscando evitar que colisões entre aeronaves, danos à pessoas e patrimônio, bem como garantir isonomia entre empresas que pretendem desenvolver novos planos de negócio baseado no uso dessa tecnologia. Para isso, os países têm trabalhado na construção de um



novo espaço aéreo e será utilizado apenas por aeronaves não pilotadas (ou remotamente pilotadas) – o Unmanned Traffic Management (UTM). O UTM tem por objetivo identificar serviços, funções, responsabilidades, arquitetura de informações, protocolos de troca de dados, funções de software, infraestrutura e requisitos de desempenho para permitir o gerenciamento de operações de aeronaves remotamente pilotadas não controladas em espaços aéreos de baixa altitude (< 500 Ft). Ou seja, o UTM é um ecossistema de "gerenciamento de tráfego" para operações não controladas que é independente do sistema convencional de controle de tráfego aéreo (ATM), mas que é complementar ao mesmo.

Um dos problemas relacionados a sistemas UTM é como prover consciência situacional a todos atores em tempo real, uma vez que muitas vezes eventos diversos e dinâmicos precisam ser compartilhados, por terem grande impacto no desenvolvimento da operação.

Basicamente, essa operação possui os seguintes atores, conforme apresentado na figura abaixo:

- a) Drone (ARP): plataforma aérea que possui uma inteligência embarcada que é capaz de detectar obstáculos e outras aeronaves, assim **como evitar que a aeronave venha a colidir com os mesmos (função *avoid-collision*)**. Ela envia de 10 em 10 seg, via CLOUD, suas informações de posicionamento (***self status***) para a UTM-CTR. É importante que cada drone possui uma GCS associada e este possui um link de comunicação ponto-a-ponto (***c2 msg***) – **NÃO CONSIDERAR ESSE CANAL PARA EFEITO DESSA PROVA**.
- b) GCS: estação de controle de um ARP específico, conforme legislação pertinente, que pode **interferir através de um piloto habilitado no que o drone está a realizar** (mudar altura, prumo, etc.), através do link de comando e controle que ele tem com o drone (***c2 msg***) – **NÃO CONSIDERAR ESSE CANAL PARA EFEITO DESSA PROVA**. Essa estação recebe informações diretamente do drone que ela controla, assim como envia informações para o mesmo (***c2 msg***) – **NÃO CONSIDERAR ESSE CANAL PARA EFEITO DESSA PROVA**. É importante mencionar que a GCS possui informações do seu ambiente (consciência situacional), ou seja, ela possui um sistema de mapas, que permite com que ela realize sua navegação, que contém informações de posicionamento de outras aeronaves (raio de 50 km), bem como informações meteorológicas, entre outras, que são enviadas de 20 em 20 seg da UTM-CTR (via CLOUD) (***sa info***).
- c) UTM-CTR: sistema de controle e vigilância de tudo que ocorre em um espaço UTM. É ela que recebe via CLOUD todas as informações oriundas dos ARPS (***all infos***), bem como informações meteorológicos, trata essas informações (compila) e gera um mapa síntese de informações (mapa de consciência situacional), que apresenta as informações necessárias para o funcionamento das GCSs em sua área de controle. Para isso, ela envia de 20 em 20seg esse mapa síntese (***sa info***) para as GCS. Uma informação importante, é que esse serviço deve ser único, ou seja, não pode ter mais de um rodando por vez.

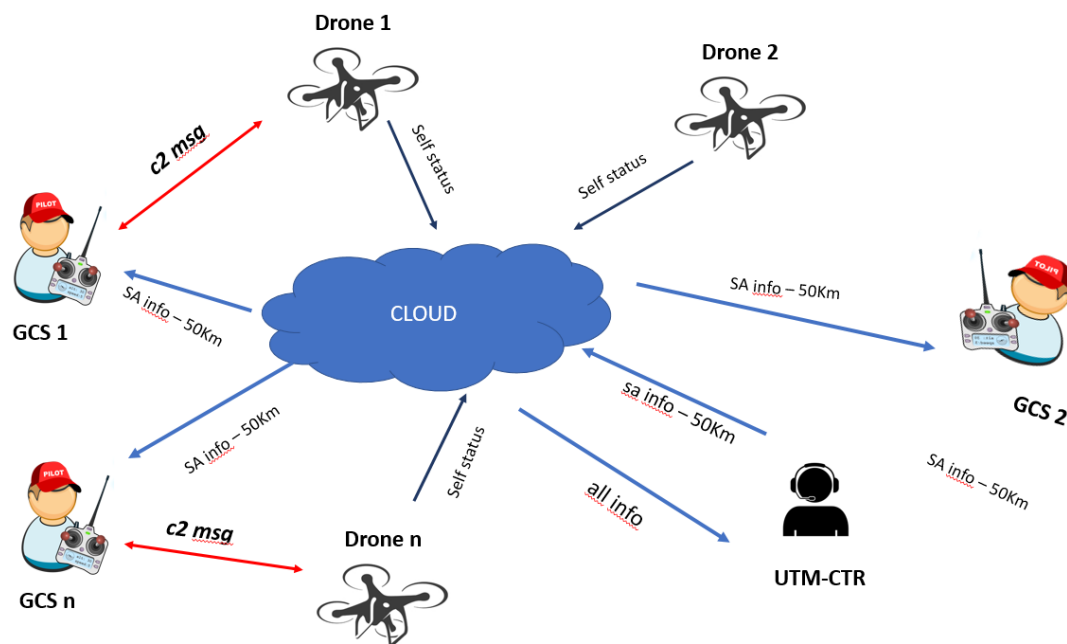


Figura A: Representação dos atores e suas relações.

PARTE II – QUESTÕES

- 1) **[1.5 PTOS]** De posse do problema apresentado na Parte I, **mapeie 03 requisitos funcionais**, os escrevendo por via de histórias de usuários (**user stories**), atentando para que os mesmos atendam para possuir as características do mnemônico INVEST (Independente, Negociável, Valiosa, Estimável, Pequena (Small) e Testável).
 - a. **Observação:** Caso falte alguma das informações requeridas para a construção de histórias de usuários, use sua imaginação para complementar a mesma.
 - b. As histórias dos usuários devem ser descritas no Word (Libre Office), não devendo ser usado nenhum sistema como Taiga, etc.
1. Como dono do drone, quero que a inteligência embarcada saiba evitar colisões para que o drone não precise de um piloto humano para não ocorrer acidente.
Critério de aceitação: simular voo com mais de um drone em situações de colisão sem acidentes.
2. Como dono do drone, quero que ele envie mensagens de posição a cada 10 segundos para que atenda às normas regulatórias governamentais.
Critério de aceitação: simular o envio de mensagens e verificar se o tempo entre cada envio não é discrepante mais do que 1 segundo.
3. Como GCS, devo poder alterar a altura do drone quando piloto habilitado para que seja possível controlar a sua rota.
Critério de aceitação: simular com o piloto habilitado se o GCS consegue mudar a altitude do drone.

- 2) **[0.5 PTOS]** Apresente (explique), como suas palavras, em cada uma das histórias de usuários os atributos contidos em INVEST são obtidos.

As histórias são testáveis devido ao critério de aceitação, elas são independentes umas das outras, algumas podem ser quebradas em outras menores como a implementação da I.A. e outras como o envio da mensagem são pequenas o bastante para serem executadas. O bom das histórias é isso poder estimar mais facilmente o seu custo e assim estimar o seu valor ou perceber a falta de informação para executar algumas e a necessidade de negociar melhor esses requisitos com o cliente.

- 3) **[1.0 PTOS]** Perceba que na descrição do UTM-CTR informa que ele deve ser implementado de forma única, ou seja, não pode existir mais de uma instância dele rodando por vez. Qual o padrão de projeto que resolve essa questão? Explique.

Design Pattern que evita que duas ou mais instâncias de uma classe existam ao mesmo tempo é o Singleton. Portanto, considerando que uma classe irá ser implementada para representar o UTM-CTR, se aplicado corretamente o DP Singleton garante-se que somente uma instância dele está rodando por vez.

- 4) **[2.0 PTOS]** O problema apresentado na Parte I, claramente nos remete a uma série de padrões aprendidos no ano de 2017 em CES 28. Entretanto, de forma bem explícita podemos ver o padrão arquitetural MVP (*Model View Presenter*) e o Observer. Usando o problema, construa um diagrama de classe que apresente uma solução para o problema.

- a. Obs1: Lembre-se de que uma das vantagens do MVP é seu baixo acoplamento, bem como uma separação clara de interesses. Dessa forma, atente para definir que um conceito é uma classe (concreta / abstrata) ou uma interface.
- b. Obs2: Identifique de forma clara, como os padrões se apresentam em seu diagrama de classe.

Pela Figura A, percebe-se que o problema pode ser modelo como o GCS e o UTM sendo as views, a CLOUD sendo o Presenter e os drones sendo o model, pensando em MVP. Porém, a Figura A está mais para um MVC, pois existe aresta de model para view.

Não desenhamos a Classe MAin no diagrama.

- 5) **[IMPLEMENTAÇÃO]** De posse do diagrama do exercício 4, usando linguagem Java implemente o mesmo. Construa uma classe principal (Main), que demonstre o correto funcionamento dos padrões usados em sua solução. Por exemplo, no caso do MVP deve ser apresentadas que as responsabilidades entre as entidades da arquitetura (model, view, Presenter) funcionam adequadamente.

- a. Obs1: Atente que não é necessário implementar nenhuma interface gráfica, onde na demonstração, deve-se imprimir na console, as mensagens de forma a entender o funcionamento do código gerado.
- b. Obs2: Para efeito do exemplo devem ser instanciados ao menos 3 instâncias de GCS, consequentemente 03 instâncias de Drones.

- c. Obs3: A solução deve tratar o problema apresentado na questão 3 (instância única do UTM-CTR).
- d. Salve todo o código gerado num pacote utm_v0.

CORREÇÃO:

i. [2.0 PTOS] – IMPLEMENTAR CORRETAMENTE A ARQUITETURA MVP(Ok feito!)

As view implementam uma IView que é uma interface que estende o Observer do Java.util.

UTM e GCS são views.

UTM é Singleton.

De 2 em 2 segundos o UTM e o GCS enviam o mapa de informações. Aqui o mapa é simples só a posição dos drones.

Presenter seria o CLOUD. Presenter é Observable de view e Observer de Model.

Models são os drones. Drones são Observable de Presenter.

De 1 em 1 segundo o Model pede pro Presenter avisar o UTM de sua posição. Presenter informa o UTM a posição do Model.

ii. [1.0 PTOS] – IMPLEMENTAR UMA CLASSE MAIN QUE DEMONSTRE O MVP (Ok feito!)

iii. [1.0 PTOS] – TRATAR O PROBLEMA DA QUESTÃO 5.C(Fizemos o singleton)

- 6) **[IMPLEMENTAÇÃO]** Transforme seu código, de forma a substituir sua classe principal (Main) por testes projetados em JUNIT e MOCKITO.
- a. Obs1: Lembre-se um teste, uma funcionalidade.
 - b. Obs2: Os nomes dos testes, devem lembrar o que está sendo testado.
 - c. Obs3: Em cada teste, deve-se ter um comentário (JAVADOC) do que está se fazendo.
 - d. Salve todo o código gerado num pacote utm_v1.

Modificou-se as classes para que não aparecesse mensagens no console.

Nem que as threads que rodavam para enviar as mensagens de tempos em tempos ficassem rodando.

Testou-se alguns métodos das classes utilizando-se o Mockito. Alguns testes de estado e alguns testes de comportamento.

CORREÇÃO:

i. [1.0 PTOS] – USO CORRETO DO MOCKITO (NÃO USAR CLASSES FUNCIONAIS, QUANDO SE PODE MOCAR)

ii. [2.0 PTOS] – ESTRATÉGIA CORRETA DE TESTES -> DEMONSTRAR EM TESTES SEPARADOS O MODEL, O VIEW, O CONTROLLER

iii. [1.0 PTOS] – DEMONSTRAR EM TESTES A SOLUÇÃO DO PROBLEMA DA QUESTÃO 5.C