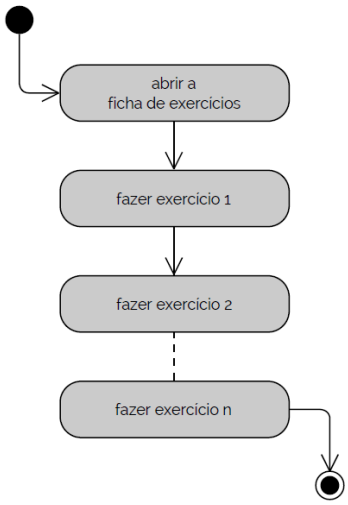
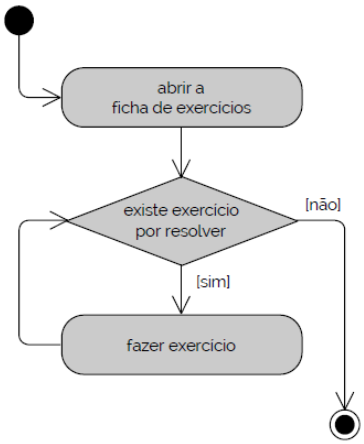
DIAGRAMA DE ATIVIDADES - COMPORTAMENTAL

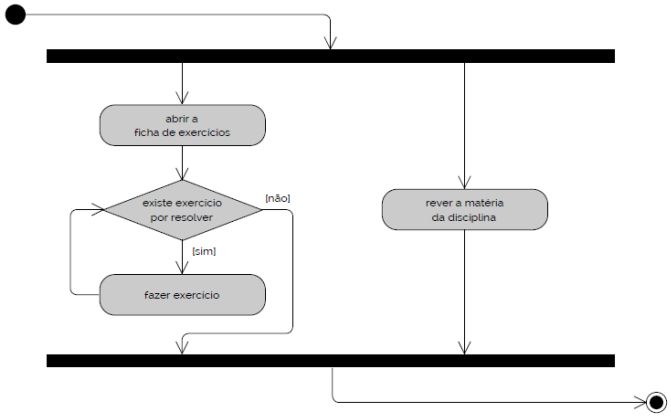
Temos inicialmente este diagrama de atividades, que inicia com uma entidade (neste caso Aluno) que abre uma ficha de exercícios (ação) e vai resolvendo os exercícios desde o 1 ao n (ações).



Porém, esta maneira de representar a ação do aluno durante a resolução da ficha pode ser melhorada. Isto é, podemos lhe atribuir uma ***condição*** que é representada por um nó de decisão e que, neste caso, irá questionar *se existem exercícios para resolver ou não*. Se sim, ele irá para a ação "resolver exercício" e, de seguida, regressa à condição anterior. Caso não haja mais exercícios por resolver, então o ciclo termina.

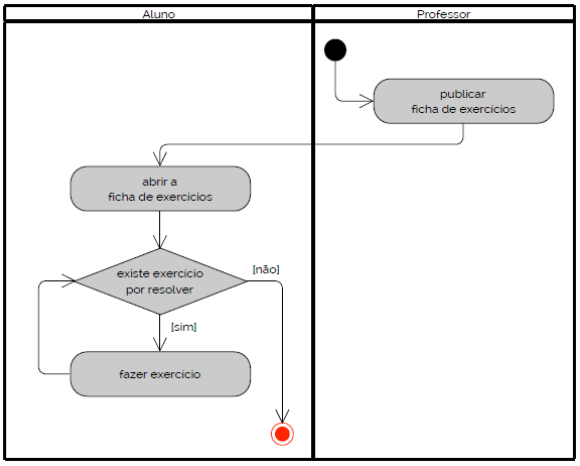


Porém, imaginemos que ao mesmo tempo que o aluno executa os exercícios, ele necessita de rever a matéria para os resolver. Então, neste caso, teremos de ter ***duas ações em simultâneo***. Este processo é possível e se criar com um ***nó de bifurcação***.

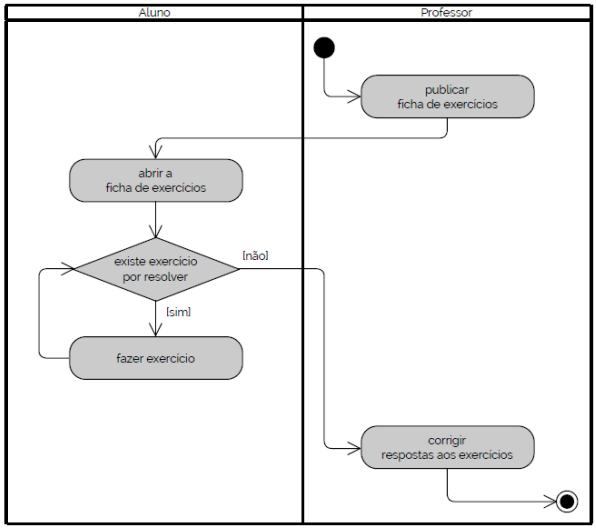


Por fim, analisamos o caso de que, para o aluno ter uma ficha de exercícios para resolver, então podemos suspeitar que a mesma ficha lhe foi fornecida por um professor.

Desta maneira, necessitamos de inserir uma ***nova entidade que inicia a ação***.



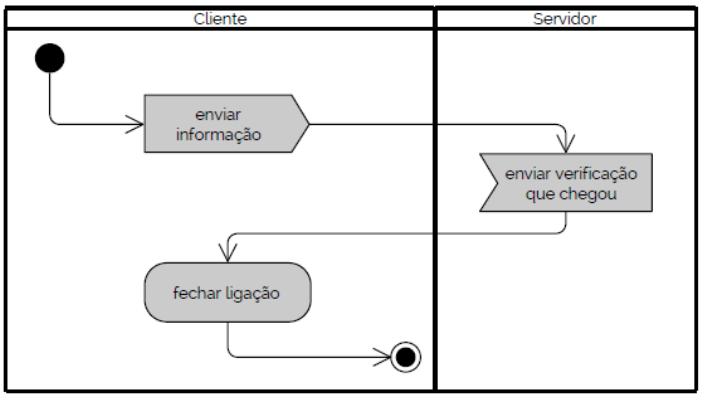
Porém, ***um processo só pode terminar na mesma entidade que a iniciou***. Desta maneira, a imagem anterior resulta na seguinte:



Neste caso que analisamos, percebemos que as ações em retângulos arredondados são feitas a partir de uma comunicação síncrona, uma vez que o passo seguinte depende inteiramente da conclusão desta. Mas e no caso de ocorrer uma comunição assincronia? Para entender

melhor este caso, por ***assincronismo*** pretende-se referir que para uma ação ser executada, não importa que haja uma resposta ao processo imediatamente a seguir a uma ação anterior e imediatamente antes de uma ação seguinte. Quando isto acontece, é importante saber designar dois papeis: quem envia uma mensagem e quem recebe e aceita uma mensagem. Para tal, criamos dois blocos – o ***bloco de envio*** e o ***bloco de aceitação***.

Imaginamos o caso de um cliente a comunicar com um servidor.



Inicialmente o cliente envia uma mensagem e o servidor, só quando receber a mensagem do primeiro, é que avança, por aceitação, para o envio de uma verificação de receção da mensagem inicial. Se esta troca fosse síncrona, então não poderíamos afirmar "só quando receber a mensagem do primeiro". Isto é, mal o cliente enviasse a mensagem, quer o servidor tenha recebido ou não esta mensagem, enviaria sempre uma verificação (possivelmente a verificar uma mensagem que nem existe). Após o envio de uma verificação, o cliente está modelado para terminar a ligação, quer o servidor tenha, ou não, recebido uma mensagem e, por conseguinte, enviado uma verificação de chegada.

Por fim, e para terminar os diagramas de atividades UML e ainda sobre o caso do Aluno, iremos introduzir o ***nó de ampulheta*** que implementa uma noção temporal, e permite que o processo seja estagnado por esta uma condição temporal.

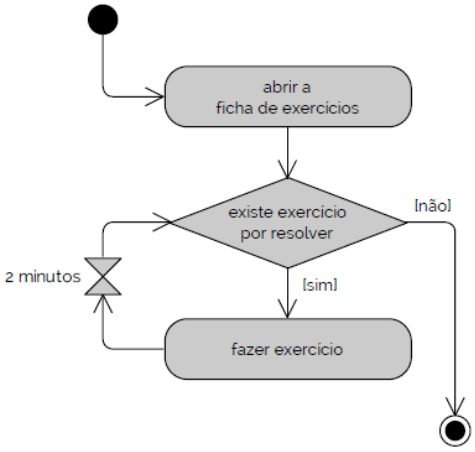
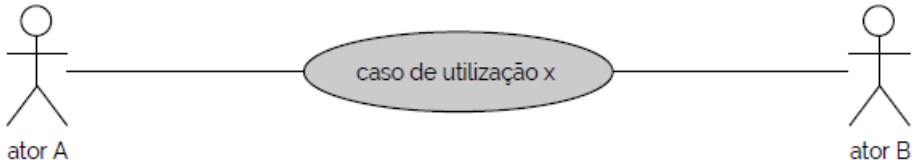
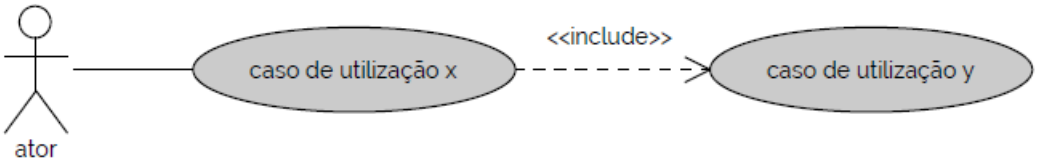


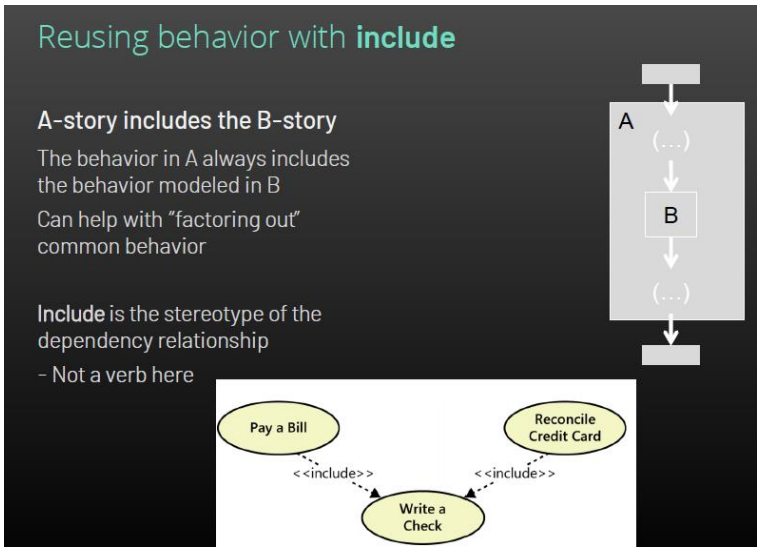
DIAGRAMA DE CASOS DE USO - COMPORTAMENTAL

Nestes diagramas é necessário que sempre haja atores. Atores intervêm nas ações do produto final e podemos ter mais de um ator a participar num caso de uso (e os mesmos têm de participar em pelo menos um caso de uso).

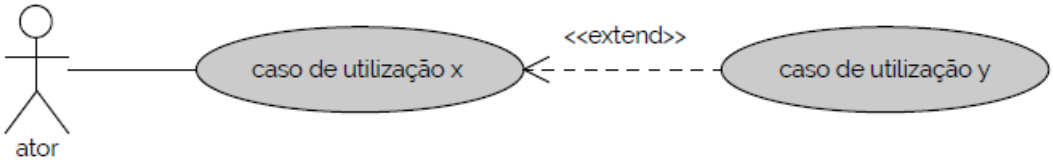


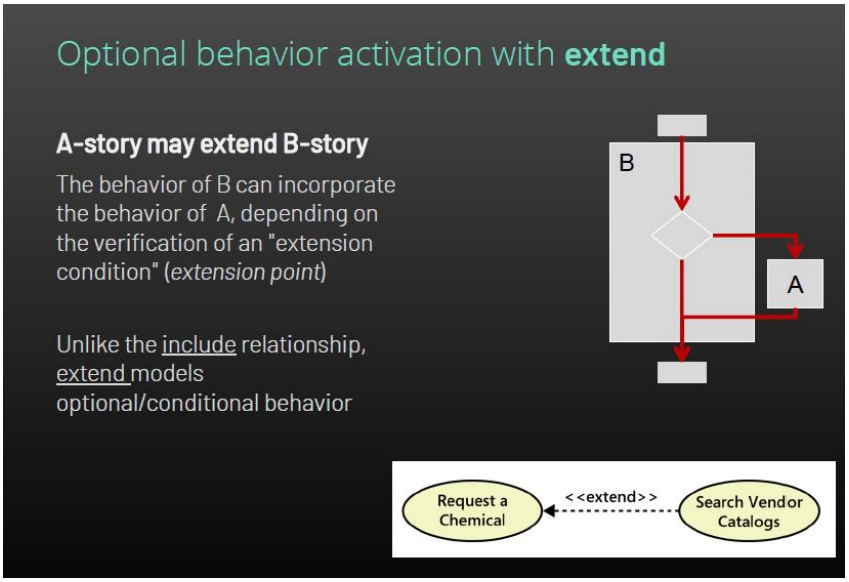
Para além disso, é possível que um caso dependa de outro e nesses casos existe a relação de ***include*** que, no caso da imagem em baixo, significa que o caso x inclui o caso y e, por isso, x incorpora y e, por outras palavras, dependendo do comportamento de x este delega y.

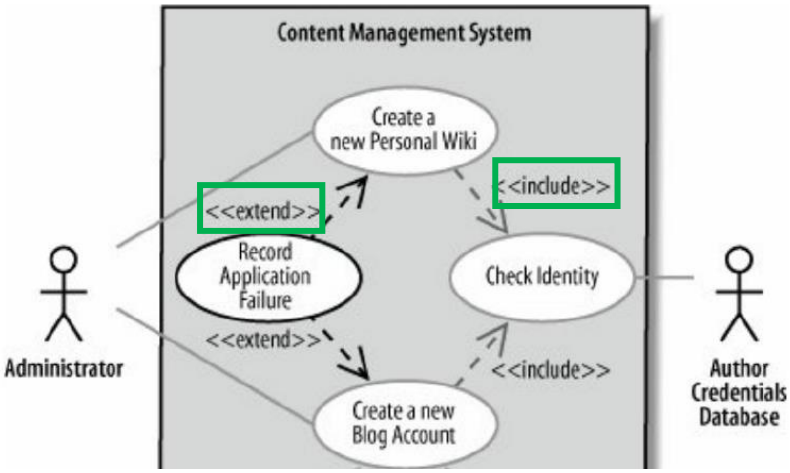




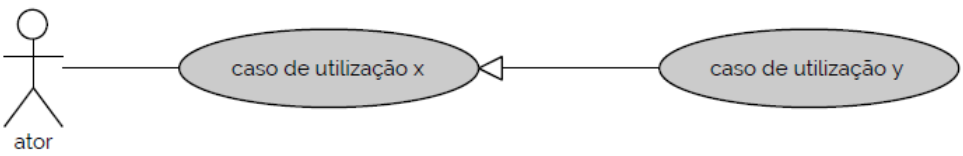
Temos também o caso de um caso de uso estender outro. Por exemplo, se o caso de y incorpora um caso x, então podemos afirmar que y pode ser estendido por x.

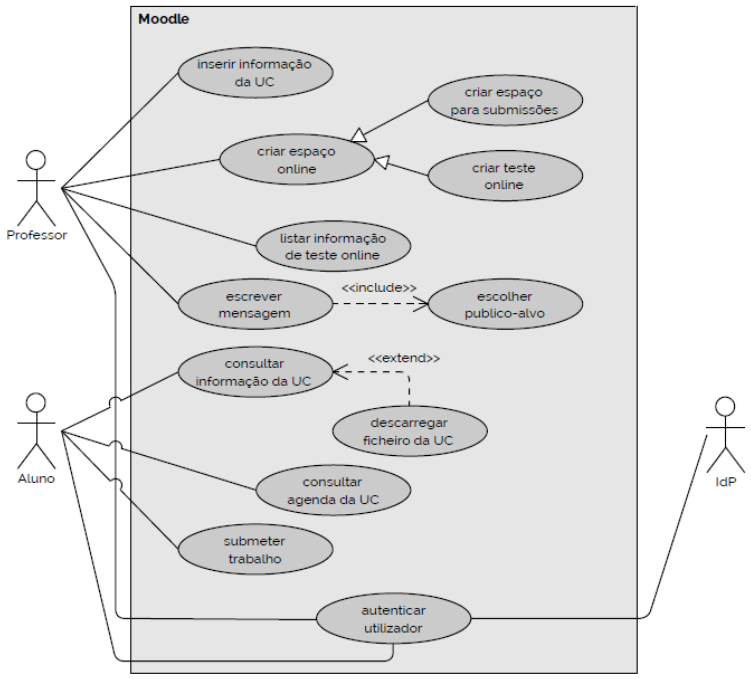






Uma das mais valias nos diagramas UML é a ***generalização***. Se um caso de uso y partilha semântica e procedimentos com um caso x, então podemos gerar uma generalização.





Todos os casos de utilização, contrariamente aos atores, estão inseridos numa só área, esta a que damos o nome de **sistema**. Se pensarmos bem, os atores não são parte do sistema, mas antes seus intervenientes. Esta é a razão pela qual estes não estão dentro da área do sistema com a nomenclatura “Moodle”.

Caso de Utilização	**criar um espaço para submissão**
descrição	O Professor cria novos trabalhos diretamente na página da disciplina, escolhendo um dos tipos disponivel e configurando o período e modo de entrega. O trabalho fica disponivel para os alunos e aceitam-se entregas no período indicado.
fluxo típico	**1.** Autenticar-se no sistema **2.** Selecionar a página da disciplina pretendida **3.** Ativar o modo de edição **4.** Adicionar o Trabalho diretamente na página de entrada **5.** Configurar os parâmetros da entrega **6.** Confirmar as alterações
fluxos alternativos	**FA1:** Sistema central de autenticação indisponivel. **FA2:** Em vez de criar um novo trabalho de raiz, o Professor pretende reusar a configuração de um já existente.
aspetos em aberto	- Um curso de semestres passados pode aceitar a criação de novos trabalhos? - Se o browser não aceitar HTML5 qual é a etsratégia alternativa?

Acoplamento:

Dizer que é importante aplicar um bom ***acoplamento*** nas várias partes de um projeto é afirmar que um projeto, no fim, não terá pontas soltas, isto é, que nenhuma das partes foi esquecida ao longo de um *processo de desenvolvimento*. Esta ideia pode provir da força que uma classe tem perante a sua ligação com outras, numa perspetiva mais orientada aos objetos.

Coesão:

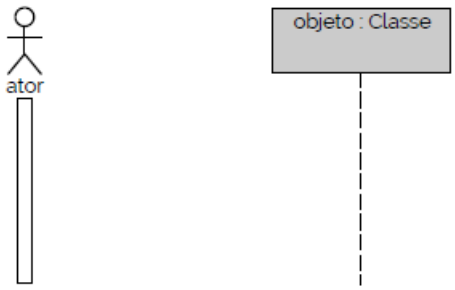
Quando referimos que um determinado componente é coeso com o projeto significa que o componente consegue obedecer às necessidades que o projeto tem, tal como às suas respon-

sabilidades: se um componente foi desenhado para produzir *x* e, ao funcionar produzir *x*, podemos dizer que este é coeso, contrariamente ao produzir outra coisa diferente de *x*. Falando em responsabilidades significa também que um componente só terá uma funcionalidade.

Tanto a coesão como o acoplamento são dois critérios importantes na análise de um ***diagrama de sequências***.

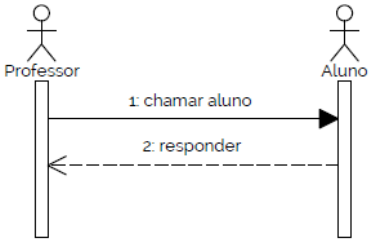
DIAGRAMA DE SEQUÊNCIAS - COMPORTAMENTAL

Nos casos em que não pretendemos referir o papel de um ator, mas antes de uma partícula do nosso sistema, representamos o objeto que possui uma linha temporal, através de um retângulo com uma legenda do tipo objeto : Classe.

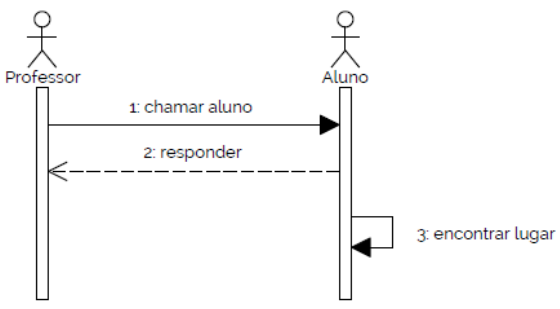


O início de um diagrama de sequências pode ser proveniente de uma mensagem exterior ou pode ser interno. Se for uma mensagem externa denomina-se de ***porta*** (transmissão de mensagem entre o nosso diagrama e algo que não esteja contemplado no diagrama, e vice-versa).

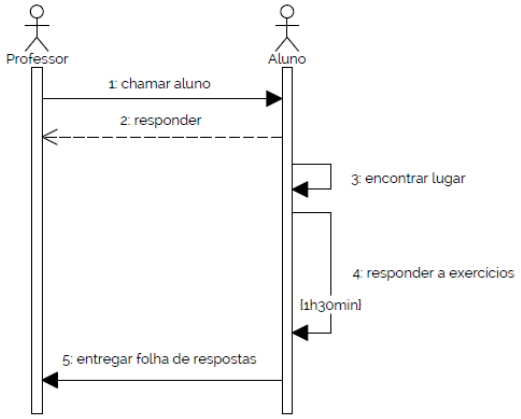
Imaginemos o caso de que um Aluno vai para a universidade num dia de avaliação e sai quando o exame terminar. Antes de entrar para realizar o teste, o Aluno espera pela chamada do Professor para que a presença do mesmo possa ser confirmada.



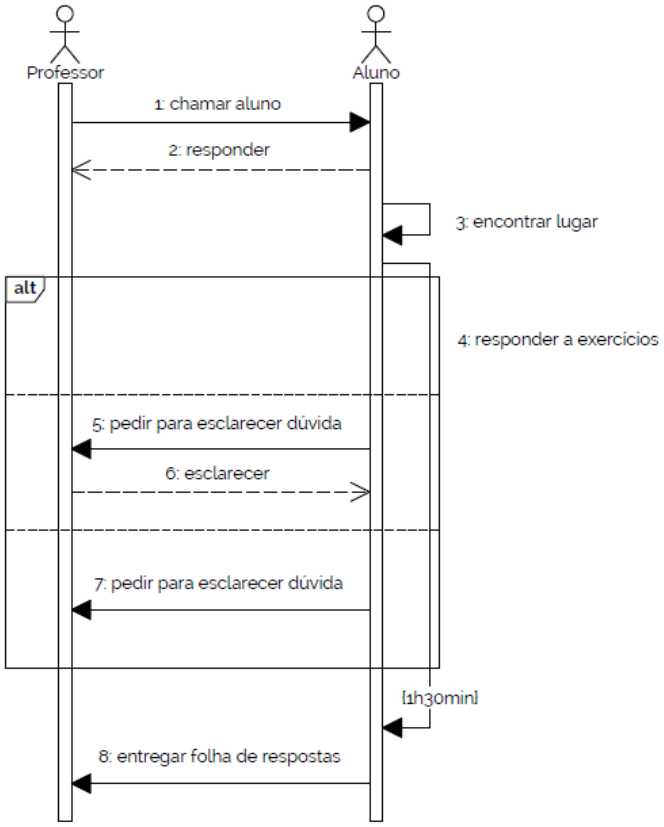
Depois de ser chamado, o Aluno vai à procura de um sítio para se sentar na sala.



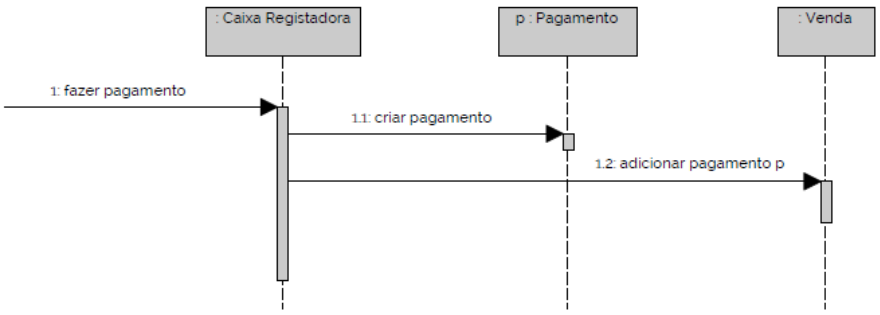
Assim que este escolhe um lugar, o exame é lhe entregue e este começa a resolver os exercícios. Quando terminar de os fazer, então entrega a sua avaliação concluída.



Embora este diagrama seja correto para descrever a comunicação simples entre o Aluno e o Professor, durante o exame, o Aluno poderá querer questionar o professor para esclarecer uma dúvida à qual o Professor poderá querer responder ou não. Para isso, demarca-se uma ***região alternativa*** onde se criam secções com o número de áreas (operandos) necessários para distinguir os fluxos diferentes. Se for realizado um ciclo, ao invés de demarcar uma região alternativa, deve-se demarcar uma ***região de loop,*** à qual não se pode contar com vários operandos.



Em termos de coesão é importante frisar, agora que já conhecemos os diagramas de sequência, que tudo depende das responsabilidades dos módulos.



Neste exemplo, o grau de coesão é muito baixo porque o mesmo módulo tem duas responsabilidades que são distintas e sem relação entre si.

O melhor seria separa as várias responsabilidades por cada módulo existente.

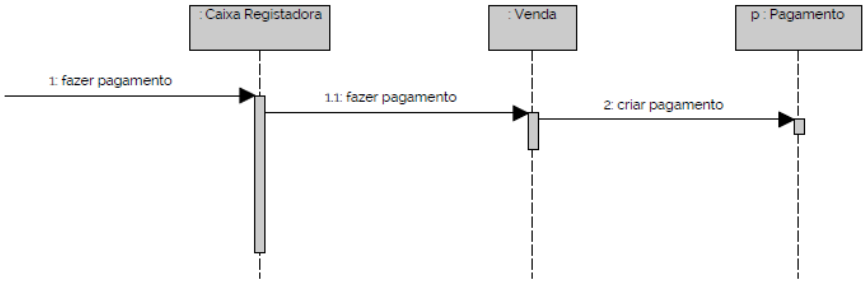


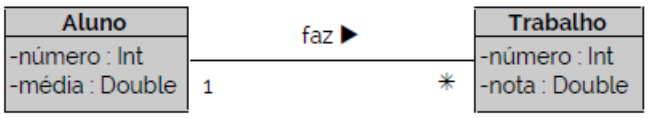
DIAGRAMA DE CLASSES - ESTRUTURAL

Num diagrama de classes, o elemento mais atómico de todos, e onde iremos basear o nosso estudo, é a **classe**. Uma classe pode ser vista como um tipo de dados/objetos, que é definida através de um conjunto de atributos e um conjunto de métodos. Representamos as classes por um retângulo tri-partido, onde a parte de cima é composta pelo nome da classe, de seguida são os atributos e, por fim, os métodos. Os atributos ***privados*** são identificados por um caracter '-', já os métodos ***públicos*** são identificados por um '+'.

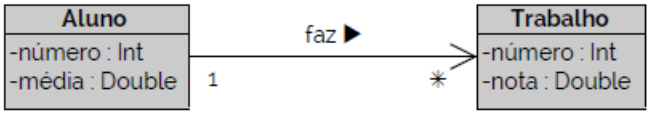
Uma classe pode ter várias relações com outras classes. Por exemplo, se tivermos duas classes Aluno e Trabalho, podemos ter uma associação referente ao fazer: o Aluno faz Trabalhos e os

Trabalhos são feitos por alunos. Note-se que para além da associação, também falamos em termos de número, isto é, aplicamos ***multiplicidade*** a esta relação.

Na relação entre as classes Aluno e Trabalho em baixo referidas, temos uma direção aplicada. Isto é, não significa que o Trabalho não se relaciona com Aluno, mas existe apenas para *simplificar a leitura* do diagrama.



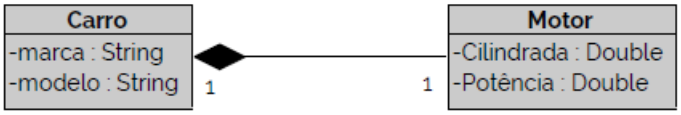
Além disso, esta maneira de demonstrar direção é diferente da seguinte, que revela que apenas existe uma direção unilateral entre a relação destas classes, sendo ela: Aluno faz Trabalhos.



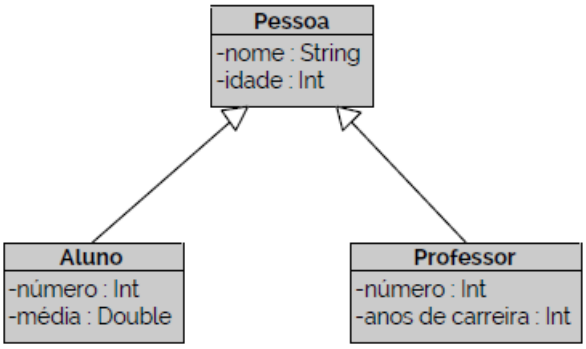
Iremos agora analisar: agregações, composições e generalizações.

No caso de *agregações*, temos que uma classe A agrega uma classe B se e só se B é parte de A. Exemplo: sendo um Motor parte de um Carro, então podemos dizer que Carro tem Motor. Representa-se agregações desta maneira:

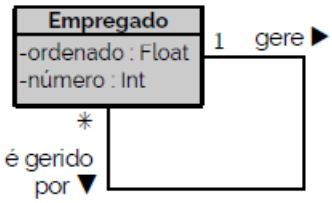
Já a *composição* diz que sendo B parte de A, A não existe sem pelo menos uma instância de B, o que é, também, o caso de "Carro tem Motor", dado que Carro não existe sem Motor. Representa-se composições desta maneira:



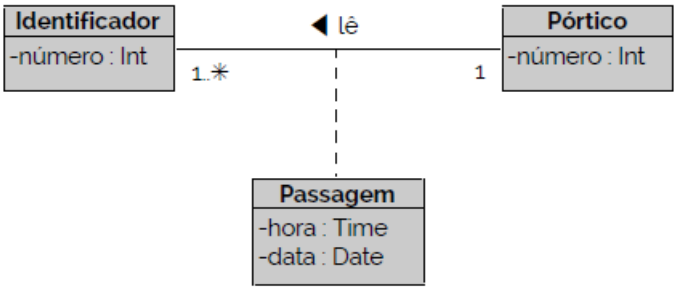
Por último, temos a *generalização* que toma partido da relação de ***herança***.



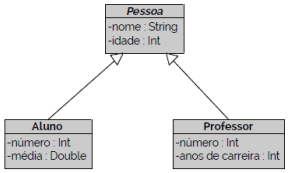
Além disto que analisamos, é possível também que uma classe tenha relações com ela própria, quase de forma recursiva, isto é, reflexiva. Por exemplo, um Empregado pode gerir por outro Empregado:



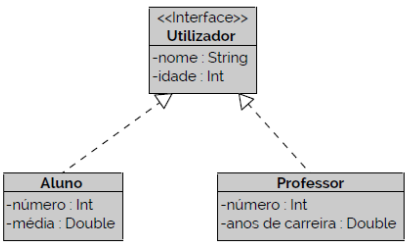
Com associações podemos também ter ***classes de associação.*** Isto é, classes que provêm da relação entre duas classes.

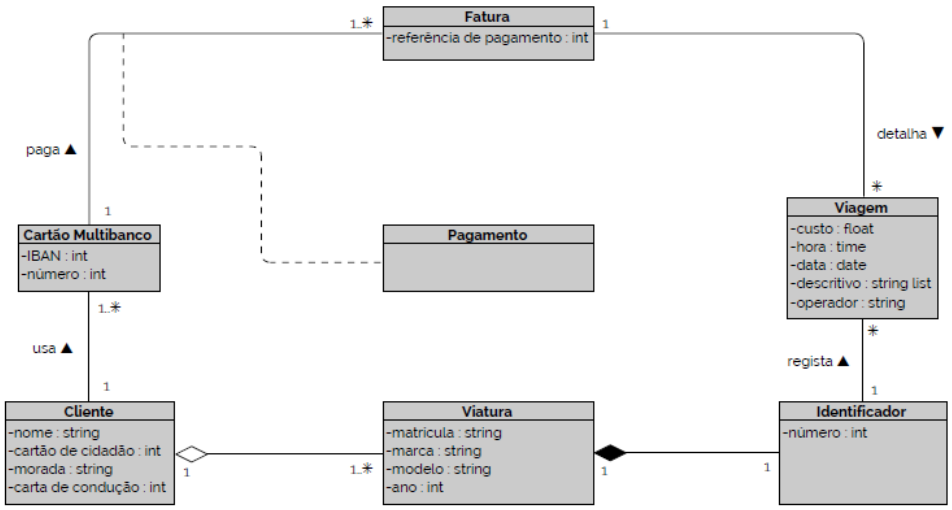


Uma classe abstrata é representada na formatação itálica do nome da classe.



Uma interface tem o esterótipo <<interface>> e são representadas pelas setas que vemos em baixo.





<>----------<>

QUESTIONS EXAME 2019:

O método *Unified Process* prevê quatro fases principais no desenvolvimento do projeto, cada qual com um objetivo a atingir (*milestone*) para se poder avançar, por esta ordem:

1/ Decisão de avançar ou parar o projeto; 2/ Arquitetura técnica definida e validada; 3/ Funcionalidades da primeira versão do produto implementadas; 4/ Solução instalada e aceite pelo cliente.

Uma das principais razões para se utilizar métodos ágeis de desenvolvimento, em detrimentos dos métodos sequenciais, é a diminuição do risco do projeto. Que prática é decisiva para a mitigação do risco:

A entrega frequente de valor diminui a probabilidade de eventuais divergências na perceção dos requisitos.

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

"A resposta com a autorização por parte do sistema de micro-pagamentos não pode demorar mais de 2 segundos."

Os casos de utilização e os requisitos funcionais descrevem um sistema segundo uma perspetiva de "caixa opaca". Qual a relação que se aplica entre estas duas técnicas:

Os casos de utilização captam os requisitos funcionais de forma contextualizada, nas descrições estruturadas dos cenários de interação.

Os conceitos de Ator e Persona são usados na análise de sistemas para descrever cenários de uso de um sistema.

A Persona é uma instância/concretização de um Ator.

A arquitetura trata da tomada das grandes decisões técnicas em relação ao sistema a desenvolver, tendo em conta os atributos de qualidade pretendidos. Um exemplo de um assunto/decisão de arquitetura é:

Especificar os cenários de interoperação com sistemas externos e as tecnologias selecionadas para os implementar.
Definir estratégias de distribuição de carga para garantir a disponibilidade do sistema em utilização contínua, com 1000 sessões simultâneas.
Definir os mecanismos técnicos para separar a informação pessoal da informação operacional, para dar resposta aos requisitos previstos no RGPD.
A tipologia de plataformas de utilização que devem ser suportadas (*web*, Android, etc.).

Os padrões de desenho fornecem soluções conhecidas para problemas de programação comuns. Alguns exemplos de padrões relacionados com a criação de objetos (*creational*) são:

Abstract Factory, Factory Method, Singleton.

Um dos princípios "SOLID" para o desenho por objetos, preconiza a segregação das interfaces. Como é que esse princípio deve ser usado pelo programador?

Criar interfaces específicas, evitando que os clientes dependam de interfaces que não utilizam.

No cerne da abordagem TDD (*test-driven development)*, está um ciclo de trabalho com um fluxo característico. Como se desenrola o trabalho do programador?

Adicionar um pequeno teste; executar os testes e verificar que o novo está a falhar; implementar as alterações suficientes para o teste passar; executar os testes e verificar que o novo passa; rever o código para o tornar mais claro.

Uma forma comum de alinhar as histórias do utilizador (*user stories)* e os métodos de garantia de qualidade do software é:

As histórias são suplementadas com a descrição de cenários concretos, que estabelecem as condições de aceitação do incremento.

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

Entrega de código (*commit*), resolução de dependências e compilação no ambiente de integração, execução dos testes automáticos, disponibilização de *feedback* quanto ao estado da *build*.

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

Os pontos acomulados das histórias implementadas, por iteração.

O modelo do domínio é construído na Análise, para mapear os conceitos do universo de discurso.

Os conceitos identificados no modelo do domínio serão candidatos naturais a constituir classes de programação, numa linguagem por objetos.

Qual o papel caraterístico do Analista numa equipa de desenvolvimento?

Analisa os processos da organização para identificar oportunidades de melhoria, e delineia o sistema de informação que as implementa.

O Diagrama 1 utiliza o conceito de estereótipo da UML (*stereotype*) na relação "extend".

Configurar módulos do projeto" e "configurar a equipa" são formas opcionais de complementar o fluxo do caso de utilização "Configurar projeto"

Relativamente ao modelo representado no Diagrama 1:

O "Gestor de Projeto" também é um "Membro da Equipa".

Nos elementos modelados no Diagrama 1 há um que está desajustado e carece de revisão. Trata-se de:

O caso de utilização "Notificações de progresso" não evidencia nenhum fluxo.

Considere a capacidade expressiva do Diagrama 2:

Um cliente pode ter um anúncio afixado em diferentes Outdoors.

Que alterações seria necessário fazer ao Diagrama 2 para captar o requisito: "Um funcionário faz a inspeção de Outdoors da sua zona de atuação."

Deve ser introduzida uma classe Zona, associando-a a Funcionário e a Outdoor.

Que alterações seria necessário fazer ao Diagrama 2 para captar o requisito: "Um funcionário faz a inspeção de Outdoors da sua zona de atuação."

Deve ser introduzida uma classe Zona, associando-a a Funcionário e a Outdoor.

Relativamente ao Diagrama 2:

O custo de um anúncio depende do tipo de papel.

Relativamente ao Diagrama 3:

Pode ser um subdiagrama associado a uma classe Perfil ("User account")

O Diagrama 3 descreve o ciclo de vida de um perfil de utilizador num site de vendas on-line.

Todas os perfis podem ser Encerrados ("Closed").

Relativamente ao tipo de diagrama ilustrado no Diagrama 3 e à sua relação com outros diagramas da UML:

Os estados representados podem ser mostrados num diagrama de atividades, como estados de uma entidade de dados (associada a um *object flow*).

<>----------<>

QUESTIONS EXAME 2019 RECURSO:

Wiegers carateriza a determinação de requisitos como um exigente desafio de interação humana. Porquê?

Analisa os processos da organização para identificar oportunidades de melhoria, e delineia o sistema de informação que as implementa.

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

A *Elaboration* deve implementar as partes críticas necessárias para confirmar a viabilidade da arquitetura.

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

"O utilizador deve poder consultar todos os seus dados pessoais, sempre que quiser."

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

Utiliza, principalmente, diagramas de sequência para caraterizar a troca de mensagens entre objetos que cooperam.

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 objetos/ conceitos do domínio/ métodos de cada classe.

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 sequência, com ativações correspondentes aos intervenientes no protocolo.

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.

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?

Para incrementar a funcionalidade de uma entidade, é preferível criar novo código, do que editar a implementação que já existe.

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

As histórias utilizam exemplos para descrever cenários de utilização, que constituem as condições de aceitação do incremento.

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

A Base de Dados é parte do sistema sob especificação, logo não é um ator.

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)

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

São usados apenas na fase de análise do sistema, para explorar requisitos funcionais.

Considere o modelo do Diagrama 2:

Um Utente pode, numa Reserva, incluir vários Equipamentos.

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

A utilização de um Equipamento desportivo, por um período de tempo.

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"?

O atributo custo deve ser movido para uma classe de associação (entre Equipamento e Modalidade).

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.

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".

<>----------<>

QUESTIONS EXAME 2020 A:

A expressão "entrega contínua" (*Continuous Delivery*) é usada para caraterizar a prática em que…

O software é construído de forma a que possa ser lançado para produção a qualquer momento.

Como se pode argumentar que a utilização de práticas de escrita de testes à-priori (TDD) não só não atrasa, como pode acelerar o desenvolvimento, no médio prazo?

O TDD revela os erros mais cedo, e é mais fácil de localizar e corrigir os erros no software, quanto mais perto do momento em que foram introduzidos.

A metáfora da "pirâmide dos testes" transmite a ideia de que:

Existem diferentes classes de teste de software, que variam em quantidade e quanto aos seus objetivos.

O V-Model descreve uma forma de integrar os procedimentos de garantia de qualidade com o processo de desenvolvimento.

O *framework* de gestão de equipas Scrum prevê a definição de alguns papéis na equipa.

Product Owner representa os interesses dos *stakeholders* e tem um papel preponderante na definição de prioridades.

Qual a relação entre as classes usadas no modelo do domínio (na análise) e as classes usadas no código (desenho e implementação)?

As classes concetuais do modelo do domínio inspiram a definição das classes do código (classes candidatas, nomes, atributos candidatos, etc.).

O modelo comportamental de um sistema inclui as realizações dos casos de utilização (*use-case realizations*), que são:

Um resultado do desenho, em que se desenvolve a interação necessária entre objetos e consequente distribuição de responsabilidade, para implementar o caso de utilização;

As narrativas dos casos de utilização suportam diferentes atividades do processo de desenvolvimento, mas **não servem** para:

Indicar a arquitetura necessária para satisfazer as ações dos atores.

Tanto os casos de utilização como as histórias (*user stories*) fazem parte dos recursos da análise de sistemas. Qual das seguintes propriedades é caraterística dos casos de utilização (e não da história)?

Descreve como é que o ator se imagina a interagir com o sistema para realizar os seus objetivos.

O Analista documenta requisitos, mas também o ambiente do negócio. Identifique um exemplo de uma regra do negócio ("Business rule"), no contexto de um sistema para pagamento de refeições, numa cafetaria.

Para os utilizadores que se apresentam sem cartão, a refeição tem um custo suplementar de 1EUR.

Assinale o conjunto que corresponde a uma coleção de técnicas de levantamento de requisitos estudadas.

Entrevistas, *workshops*, questionários, análise documenta, observação no local, "focus group".

Relativamente ao Diagrama 1:

Quando o dentista cria um diagnóstico, pode consultar exames de imagiologia que existam.

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

Podem ser detalhados/suplementados com diagramas de sequência

---

Em relação ao Diagrama 2:

O serviço "ProductSearch" é implementado pelo subsistema "WebStore".

Em relação ao Diagrama 2:

É uma vista de arquitetura que mostra a forma como a solução está dividida em componentes, e as interfaces entre eles.

Relativamente ao Diagrama 3, é possível inferir que:

A execução das invocações dentro do fragmento "alt" é condicional.

Como é que os diagramas do género do Diagrama 3 são utilizados num projeto?

O analista prepara estes diagramas para mostrar a realização dos casos de utilização.

Este tipo de diagramas pode ser usado para clarificar a lógica da integração com um sistema externo.

<>----------<>