Licenciaturas em Engenharia Informática

Modelação e Design

AULAS LABORATORIAIS – 2017 / 2018

FICHA 7 - DESENHO DE ARQUITETURA

Exemplo de Desenho

Considere que, após efetuar a devida análise a um caso de uso "Regista Venda", foram obtidos os seguintes diagramas de sequência de sistema (DSS – Figura 1) e do modelo de domínio (MD – Figura 2).

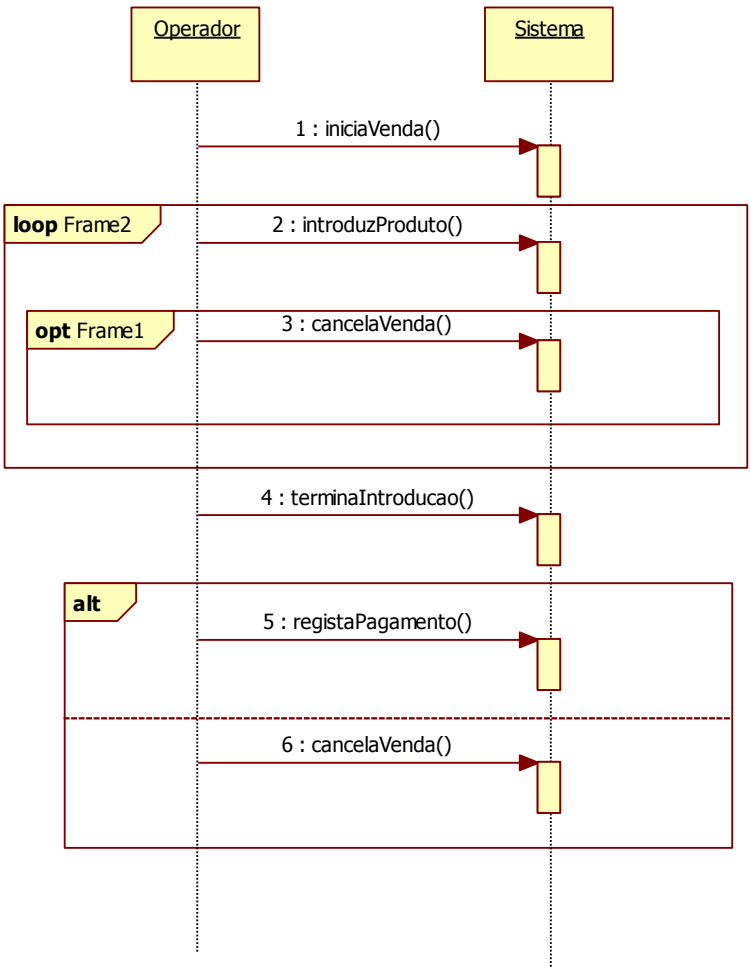

Figura 1 - Diagrama de Sequência de Sistema - Caso de Uso "Regista Venda".

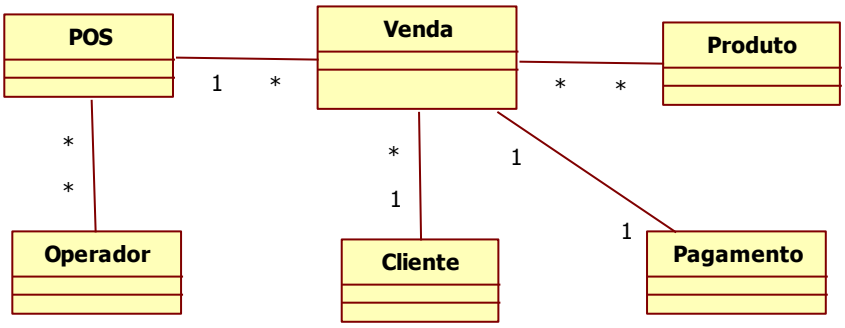

Figura 2 - Modelo do Domínio - Caso de Uso "Regista Venda".

De que forma deve continuar o desenvolvimento?

Tendo sido estabelecidas as mensagens que circulam entre o ator e o sistema e tendo sido identificados os conceitos mais importantes, torna-se necessário efetuar o desenho (design) do software. O propósito da atividade de desenho é obter uma descrição da estrutura e do comportamento do software, normalmente através da criação de um conjunto de diagramas adequados (especialmente, diagramas de classes e diagramas de sequência).

Como obter o desenho do sistema?

A estratégia geral para desenhar o sistema pode ser descrita, sumariamente, na seguinte forma:

1. As mensagens identificadas no DSS representam estímulos que elicitam uma resposta do sistema. Sendo assim, a resposta a essa mensagem deve ser considerada como uma responsabilidade a distribuir por um conjunto de classes e de métodos adequados.
2. A atribuição de responsabilidades é feita identificando os métodos que as implementam, quais os dados relevantes e em que classes é que estes métodos devem ser colocados.
3. A qualidade da solução encontrada deve ser avaliada, tendo em consideração os princípios básicos da análise orientada a objetos:
    a. Os dados devem encontrar-se próximos dos métodos que os manipulam.
    b. As classes devem ter elevada coesão. Ou seja, não devem acumular (muitas) responsabilidades diversificadas.
    c. Deve existir um baixo acoplamento entre as diversas classes. Ou seja, uma classe não deve depender de muitas outras classes.

Estes critérios de avaliação entram por vezes em conflito (por exemplo, uma solução que verifique o primeiro critério pode não verificar o segundo critério), pelo que, por vezes, pode ser necessário aceitar certos compromissos.

As classes de desenho devem ser, sempre que possível, inspiradas nas classes identificadas no modelo do domínio. No entanto, podem ser criadas outras classes que não se encontram no modelo do domínio. Normalmente, também é necessário criar o modelo de dados (para assegurar a persistência dos dados relevantes, numa base de dados).

Exemplo

Considerem-se os diagramas apresentados. A mensagem iniciaVenda() é despoletada quando o operador pretende dar inicio ao registo de uma nova venda. Por agora, a ênfase é colocada no desenho da parte do sistema que é responsável pela resposta a esta mensagem.

Um extrato, relevante da descrição do *caso de uso*, é:

1. O operador escolhe a opção que dá início a uma nova venda.
2. O sistema apresenta uma mensagem de boas vindas.

Numa especificação suplementar (por exemplo, glossário), sabe-se que cada venda tem um código identificador único e está associada à informação dos produtos vendidos e à forma de pagamento.

Portanto, quando o utilizador pretende dar início a uma nova venda, é necessário inicializar uma nova venda, vazia, com um identificador único e apresentar uma mensagem adequada. Assim, deve-se identificar quais os dados envolvidos nesta operação e qual é (ou são) a(s) classe(s) que os contém. Neste caso, é necessário armazenar a informação relativa à venda que é iniciada.

Como tal, é preciso determinar a classe que será responsável por isto. Como é que se escolhe? Vai-se usar o seguinte processo:

1. Se, porventura, já existir no modelo de desenho uma classe adequada, aproveita-se essa classe.
2. Caso contrário, olha-se para o MD e verifica-se se alguma das classes, lá representadas, é uma candidata adequada para aí se incluírem os novos métodos e os dados.
3. Caso nenhuma classe do MD pareça adequada,… pode ser sugerida (definida) uma nova classe que o seja!

Neste caso, como esta é a primeira mensagem, que se está a concretizar, não existe uma "classe candidata" que já defina esta operação. Por esse motivo, vai ser necessário introduzir uma nova classe, no modelo de desenho.

Olhando para o modelo do domínio, percebe-se que a classe POS pode ser considerada uma candidata adequada, dado que, semanticamente, parece fazer sentido "iniciar uma venda no POS". Tendo isso em mente, vai-se criar uma versão inicial do desenho, da fracção do sistema, responsável por dar a resposta. Isto é feito construindo os diagramas de sequência e os diagramas de classes, adequados.

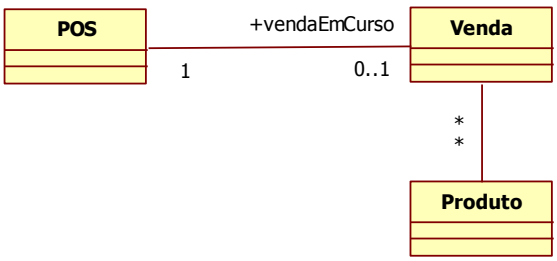

Figura 3 - Diagrama de Classes (Desenho) - versão inicial.

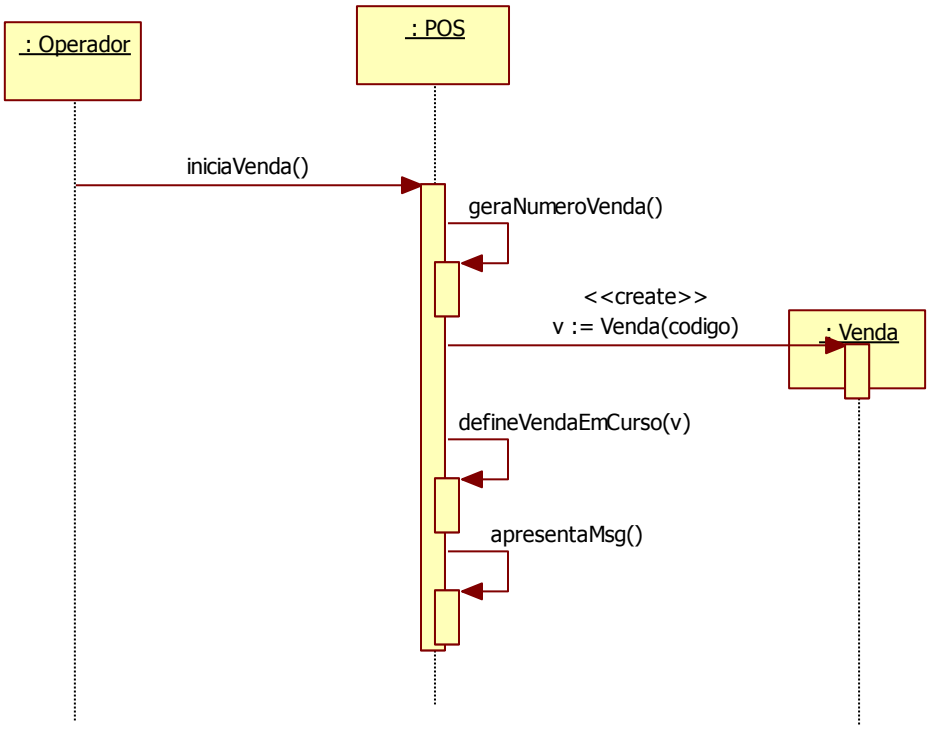

Figura 4 - Diagrama de Sequência (Desenho) - versão inicial.

Os dois diagramas, em articulação, permitem dar uma ideia concreta das classes necessárias e da colaboração entre estas. Paralelamente, vai-se também poder analisar a solução de uma forma crítica, tendo em consideração os critérios indicados (forte coesão, …). Tendo em atenção esses critérios, percebe-se que:

> Um aspeto negativo é que a classe POS é responsável, neste momento, por criar a nova venda corrente, gerar o código único e apresentar a mensagem.

Assim, pode-se concluir que a classe POS tem demasiadas responsabilidades atribuídas. Isto é desaconselhável pois dificulta a sua manutenção (torna-a mais difícil de compreender, de modificar, …). Então, para resolver este problema, vai-se reorganizar a estrutura de forma a eliminar essa desvantagem. Isso será conseguido através da deslocação de algumas das responsabilidades identificadas para outras classes (eventualmente, para novas classes).

Considere a geração do identificador único, seguindo o processo descrito:

1. Olhando para as classes, atualmente existentes no modelo de desenho, pode parecer que a classe Venda é uma candidata razoável para gerar o identificador único. Não sendo uma má opção, há que ter em atenção, no entanto, que a classe Venda refere-se apenas a uma única venda, enquanto a geração de um identificador único diz respeito a TODAS as vendas (anteriormente) geradas. Sendo assim, pode-se optar por atribuir essa responsabilidade a uma nova classe, criada especificamente para esse propósito.
2. Tal como anteriormente, vai-se olhar para o MD para encontrar uma classe adequada. No entanto, neste caso, não parece haver uma alternativa razoável à classe POS.
3. Sendo assim, é necessário criar uma nova classe "artificial", que deverá ficar responsável pela criação do "número único de venda": GeradorIdentificador.

Por um processo análogo, pode-se concluir que se pode definir uma classe "InterfaceVenda" que ficará responsável por receber o *input* do utilizador e apresentar a mensagem. Fazendo estas modificações, fica-se com o novo desenho (Figura 5 e Figura 6):

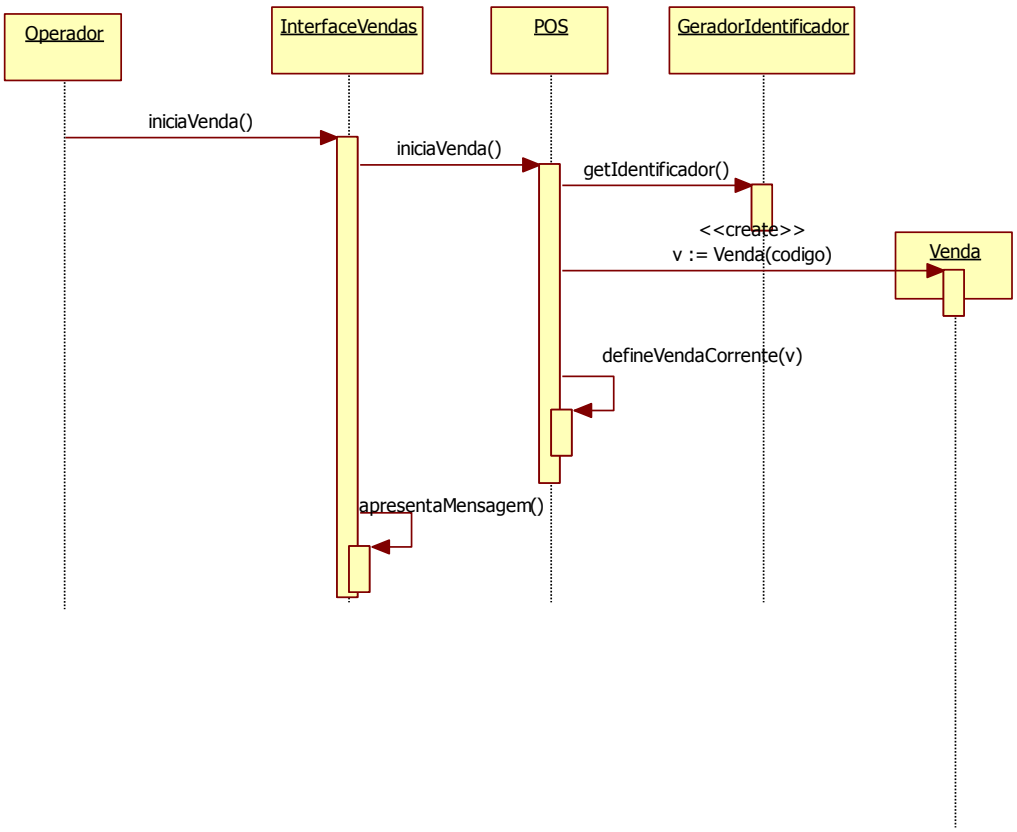

Figura 5 - Diagrama de Sequência (Desenho) – nova versão.

Com o diagrama de classes correspondente:

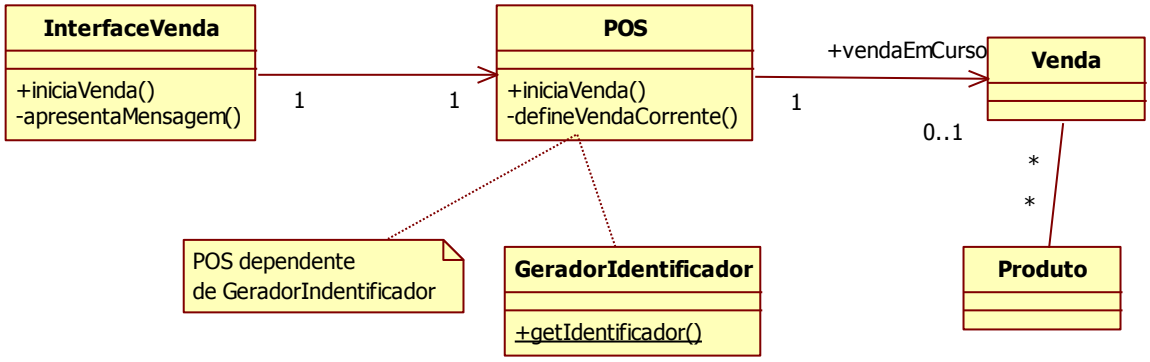

Figura 6 - Diagrama de Classes (Desenho) – nova versão.

Note-se que o método getIdentificador é estático. Para além de serem incluídas as novas classes, o diagrama de classes também exibe a seguinte informação:

- Os métodos são apresentados nas classes adequadas.
- De acordo com o comportamento descrito no diagrama de sequência, pode-se já identificar a visibilidade dos métodos, bem como a navegabilidade entre as classes InterfaceVenda, POS e Venda. Naturalmente, esta informação tem natureza provisória e poderá ser atualizada, caso o desenho das outras partes do sistema venha a revelar, por exemplo, que um determinado método necessita de ter a sua visibilidade modificada.

De acordo com os critérios enunciados, a nova solução é preferível, relativamente, à solução inicial. Caso ainda não seja satisfatória, podem-se efetuar novas iterações até que a nova solução seja considerada satisfatória. O processo de desenho prosseguiria, por exemplo, repetindo-se o processo para a mensagem seguinte (introduzProduto). O resultado final resulta da integração dos resultados obtidos nas diversas iterações.

OBSERVAÇÕES FINAIS

A solução encontrada tem um aspeto comum à maioria das aplicações: a interface é dependente e separada da lógica e dos dados da aplicação mas, o inverso, não se verifica. Apesar de não se ter partido inicialmente dessa solução, rapidamente se lá chega. Este tipo de situação é relativamente comum e deu origem ao estudo de *padrões de desenho de software*, que sistematizam problemas típicos e as respetivas soluções.

Naturalmente, não é obrigatório considerar apenas uma mensagem de cada vez que se cria o desenho do sistema. Se for mais fácil, ou lógico, trabalhar considerando um cenário mais

abrangente, isso pode ser feito. No entanto, uma forma de lidar com a complexidade é dividir um problema complexo (o desenho do sistema completo) em vários sub-problemas mais simples (o desenho dos elementos necessários à resposta de uma única mensagem).

Observe-se que a função do modelo do domínio consiste em dotar-nos de uma nomenclatura adequada para as classes que precisamos de criar, enquanto a função do diagrama de sequência de sistema é identificar as mensagens às quais o sistema deve ser capaz de desenvolver.

O desenho das classes do sistema não deve ser feito independentemente, mas sim em paralelo com o desenho da colaboração, entre as mesmas. Caso isso não seja feito, incorre-se no risco de serem projetadas "soluções" demasiado ingénuas, normalmente adaptadas de forma mais ou menos direta do modelo do domínio, o que é francamente desadequado, ao problema em questão.

Por fim, nada obriga a que uma mensagem de sistema corresponda apenas a uma única mensagem (ou método) no diagrama de desenho. Por exemplo, uma mensagem de sistema *introduzLogin(username, password)* pode ser decomposta numa sequência de duas mensagens (por exemplo, *introduzUsername(username)* e *introduzPassword(password)*, caso isso se revele conveniente.

EXERCÍCIOS

1. Propõe mais alguma modificação ao desenho apresentado? Se sim, justifique, tendo em consideração os *princípios de qualidade* enunciados no texto.
2. Considere a mensagem de sistema introduzProduto(codigoProduto, numero). Efectue o desenho do *software* correspondente ao processamento dessa mensagem.