Departamento de Mecânica Computacional
Faculdade de Engenharia Mecânica
Universidade Estadual de Campinas

Aula 6 - Principais documentos UML

Prof. Eurípedes Guilherme de Oliveira Nóbrega

ES770 - Laboratório de Sistemas Digitais
Segundo semestre - 2019

Engenharia de Controle e Automação

September 13, 2019

1 Introdução

Será usado o exemplo do projeto de uma máquina de vendas de bebidas quentes para introduzir de forma sucinta os principais conceitos dos seguintes diagramas:

- Casos de uso
- Atividade
- Comunicações
- Classes
- Sequência
- Máquina de estados

Os diagramas UML são genericamente classificados em dois tipos, estruturais e comportamentais (ou dinâmicos). Os primeiros apresentam a arquitetura de concepção da relação entre os elementos que o formam, enquanto os segundos relacionam-se com as operações que serão realizadas sobre os seus elementos através do fluxo de execução do programa.

Apesar de não ser uma sequência a ser seguida sempre, sugere-se que os diagramas sejam desenvolvidos na ordem em que serão apresentados, conforme a lista acima. Por outro lado, é também conveniente que o esforço para ser elaborado cada diagrama seja comedido, fazendo com que, a partir do caso de uso, todos os demais sejam rapidamente elaborados, no que será compreendido como o primeiro ciclo do projeto. O passo seguinte será rever todos os diagramas, modificando-os na medida em que mudanças foram realizadas nos diagramas seguintes. Esse será o segundo ciclo do projeto. Outros ciclos devem ser efetivados, até que o projeto esteja o mais bem elaborado possível, para então se iniciar a codificação.

A justificativa para esse procedimento cíclico é que um projeto de software é sempre complexo e multifacetado, por mais simples que pareça inicialmente. Desse modo, cada diagrama consiste em uma visão parcial do projeto, pode-se afirmar uma de suas faces, que dificilmente permite uma visão abrangente. Diversos aspectos do projeto apenas serão visíveis em um ou outro diagrama, ou pelo menos mais facilmente percebíveis. Assim, após um ciclo, principalmente o primeiro, várias modificações poderão ser incluídas em função da percepção do problema a partir dos demais pontos de vista proporcionados por cada diagrama. Isso não consiste de modo algum em uma falha, trata-se na verdade de eficiência no uso do tempo de projeto, evitando gastá-lo

em demasia para chegar de imediato ao melhor resultado possível para cada diagrama.

É conveniente notar que a metodologia conhecido como *programação por objetos* é atualmente adotada de forma geral também para o projeto de sistemas de computação, mesmo que a linguagem a ser usada na programação eventualmente não seja programável por objetos diretamente. Nesse caso, deve-se tentar adaptar a programação aos elementos que constarão nos diagramas UML em geral.

Finalmente, deve-se mencionar nessa introdução que esses diagramas são um número mínimo bem representativo das técnicas de projeto usando UML. Há diversos outros que bem podem ser usados para corresponder à complexidade do trabalho, e devem portanto serem igualmente estudados, na medida em que a necessidade seja encontrada.

2 Descrição do problema

Projetar uma máquina de venda de bebidas quentes que possa implementar os seguintes itens:

- Opção entre café, café com leite, chocolate;
- Opção de três tamanhos, curto, médio e longo;
- Opção entre sem açucar, pouco açucar e açucar normal;
- Pode iniciar com um token ou com uma seleção;
- Três preços, um para cada tamanho, um, dois ou três tokens.

Na sequência do texto, será usado o termo "sistema" significando o conjunto de programas de computador que será necessário para a solução do problema em questão. Note que a descrição acima é sumária. Alguns aspectos são essenciais para a sua consecução, e serão necessários serem incluídos no projeto. Outros foram propositadamente deixados de fora, e não precisarão ser incluídos. Eventualmente, uma nova especificação poderia vir a incluí-los. Para melhor compreensão, por exemplo, a alimentação da máquina com os respectivos insumos, é essencial para o seu funcionamento, mas foi deixada de lado e não precisaria ser incluída. Mas o processo simples de inicialização, apresentando uma tela inicial para um eventual cliente, deve fazer parte.

3 Diagrama de Caso de Uso

O objetivo dos diagramas de caso de usos é representar a interação entre os diversos atores que interagem com o sistema. Reporta-se fundamentalmente aos requisitos para o projeto apresentados na descrição do problema, focando na sua dinâmica. Apresenta uma visão externa e global do sistema, identificando os principais fatores que influenciam o seu comportamento.

Fazem uso dos seguintes elementos:

- Funcionalidades do sistema, representadas comumente através de um elemento elíptico;
- Atores que realizam as ações descritas nas especificações;
- Relacionamentos entre as funções e dessas com os atores;
- Um retângulo representando as fronteiras do sistema, assim delimitando os seus elementos internos e externos.

Deve-se notar que os módulos elípticos que representam as funções são também chamados de *casos de uso*.

Os seguintes aspectos são importantes para a boa compreensão dos diagramas:

- Os atores devem ser identificados por nomes (substantivos) adequados, ilustrativos de seus papéis;
- As funções devem ser descritas pela ação que representam (verbos);
- As relações entre as funções devem ser de dois tipos: *inclusão* e *extensão*.

A relação de inclusão significa uma função que complementa necessariamente uma outra função, implicando em que ambas serão realizadas, ou seja quando a principal for executada, a incluída também o será.

A relação de extensão significa que a função complementar pode ou não ser executada, como é o caso por exemplo de opções em um menu. Mas deve-se notar que existem várias outras possibilidades de extensões de uma função.

É importante ressaltar que o diagrama de caso de uso é a representação mais abstrata do sistema, não devendo portanto entrar em detalhes desnecessários, e assim necessitando de outros diagramas para complementar o comportamento dinâmico do sistema.

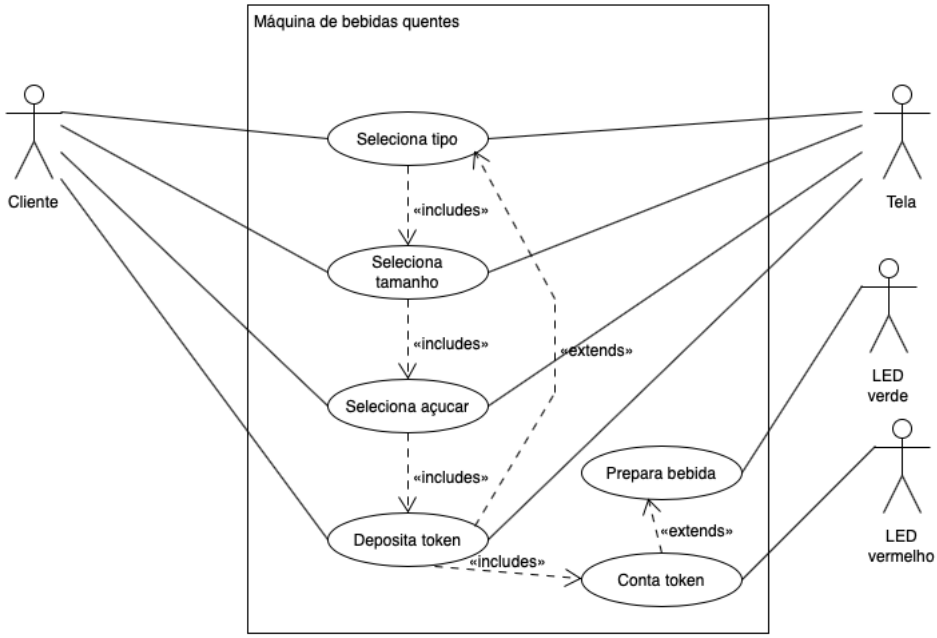

Figure 1: Exemplo de caso de uso

Nota-se na Figura 1 que o cliente da máquina de bebidas quentes realiza 4 ações para conseguir comprar uma bebida de sua escolha, selecionando as diversas opções e depositando uma moeda na máquina. Pode iniciar pela seleção do tipo do café, e em seguida realiza as opções seguintes incluídas na primeira. Se optar por começar pela deposição de um token, uma extensão dessa opção permite que realize a seguir a seleção da bebida, que inclui as opções de bebidas conforme já mencionado, mas também a deposição de moedas, de acordo com o preço especificado por suas opções. Tendo sido pago completamente, através da contagem dos tokens, é iniciada a preparação da bebida pela máquina. Para cada ação requerida pela máquina, uma mensagem é exposta na tela, e a preparação da bebida e sua conclusão são informadas ao cliente através dos LED's vermelho e depois o verde. Aparentemente, todas as especificações descritas nos requisitos foram atendidas.

4 Diagrama de Atividades

O diagrama de atividades é o segundo diagrama que descreve a dinâmica do sistema, agora focando no fluxo de ações, que representam as operações do sistema. Lembrando que o fluxo pode ser sequencial, onde uma ação precede

outra sucessivamente, podem ocorrer ramificações, com uma decisão levando o fluxo para uma dada sequência de ações alternativas, ou ainda concorrente, quando mais de uma ação ocorre de forma simultânea, o quê pode ser efetivado através de múltiplos *threads* ou por núcleos ou processadores diferentes.

As atividades são conduzidas através de mensagens que são enviadas entre os objetos do sistema, sendo que cada mensagem repassa o fluxo para o objeto destino, que pode executar completamente sua operação implícita na mensagem, retornando ao objeto que enviou a mensagem, ou ainda enviar uma nova mensagem a um outro objeto para dar continuidade ao fluxo do processo. Assim o diagrama deve representar um ponto de partida e um ou mais pontos de chegada, onde o ciclo da atividade se encerra. Pode ainda representar todo o sistema ou parte dele.

Apesar do fluxo representar uma passagem de mensagem, estas não deverão ser representadas diretamente no diagrama de atividades, o que será feito no diagrama de comunicações.

Assim, o diagrama de atividades é composto dos seguintes elementos:

- Elemento de partida
- Elementos de chegadas
- Operações
- Controles de fluxo

O elemento de partida é representado por um círculo negro, e o de chegada pelo mesmo círculo envolto por uma circunferência. As operações são representadas por retângulos, usualmente com cantos arredondados, e os elementos de controle de fluxo por barras ou losangos, podendo ocorrer ramificação, por decisão ou concorrência, ou de conjunção de fluxos. Nesse último caso, trata-se da junção entre fluxos alternativos, assim denotando um ponto de encontro entre as opções, ou também pode ser uma barreira de sincronização entre fluxos concorrentes. O losango representa fluxos alternativos enquanto as barras representam fluxos concorrentes.

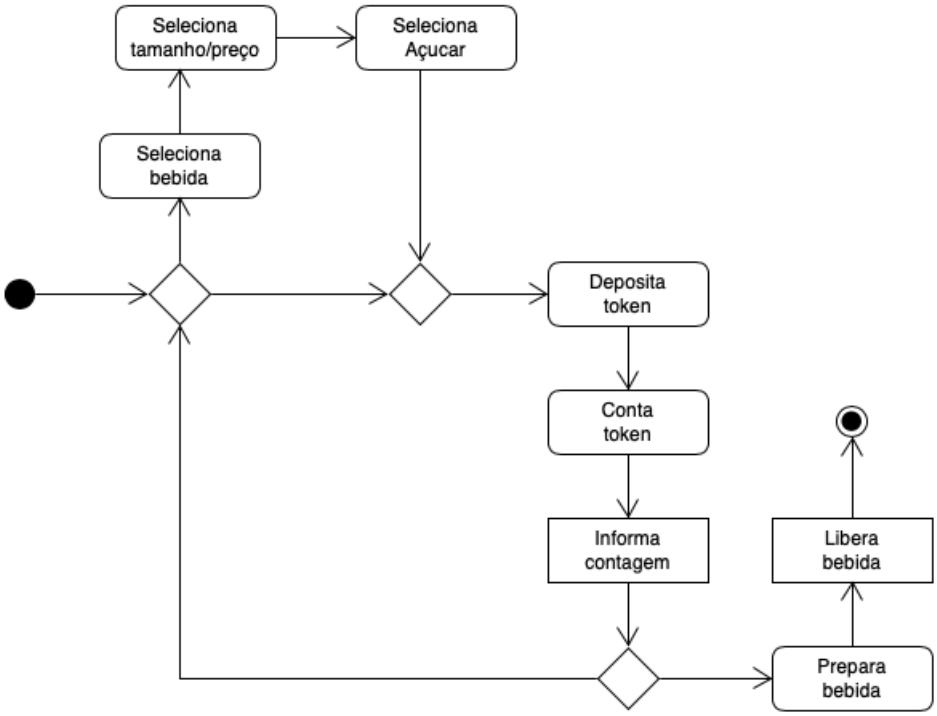

Figure 2: Diagrama de atividades do exemplo

Analisando o diagrama de atividades da Figura 2, pode-se notar que estão bem visíveis os pontos de início e de final das atividades. Partindo do início, há uma alternativa entre a seleção de bebidas ou o depósito de tokens. Se for selecionada a bebida, segue a seleção do tamanho, que implica no preço, e em seguida o nível de açucar, retornando ao fluxo em que vai se iniciar a deposição de tokens. Nessa primeira trajetória, o preço já é portanto conhecido, e o contador de tokens pode informar a contagem e se ainda falta algum token a ser inserido, o que gera o laço onde se pede o depósito de um novo token até completar o preço. A alternativa logo em seguida ao *InformaContagem* desvia à esquerda, a alternativa inicial desvia à direita e continua até o depósito de novo token. Caso tenha fechado o pagamento, o bloco mencionado informa e desvia agora à direita, iniciando a preparação da bebida selecionada e paga. Após concluir a preparação, ocorre a liberação da bebida e a conclusão das atividades. Note que não foi explicitada nesse diagrama nenhuma das ações que envolvem a emissão de mensagens na tela e o acionamento dos LED's. Isto fica implícito nos módulos apresentados, levando o cliente a tomar decisões. Poderia ser tornada explícita a emissão de mensagens, porém isso iria tornar o diagrama mais denso, e sem grandes efeitos no projeto. Em outros diagrama tais interações serão detalhadas.

5 Diagrama de Comunicações

O diagrama de comunicações é o terceiro diagrama que representa o comportamento do sistema, determinando as interações entre os objetos que o compõem, através das mensagens que são trocadas entre eles. Complementam dessa forma o diagrama de atividades, apesar de não estabelecer necessariamente a sequência de mensagens, sendo portanto uma representação estática. É também conhecido como diagrama de colaborações ou de interações.

Três elementos compõem os diagramas de comunicações:

- Quadro
- Objeto
- Mensagem

O quadro compreende o conjunto de mensagens que representa o sistema ou uma parte dele, correspondendo a um nome genérico escrito como rótulo de um retângulo (que nem sempre é desenhado). O diagrama pode então ser chamado de *quadro de mensagens*.

As mensagens são trocadas entre objetos do sistema, representados por retângulos (posivelmente de canto arredondados), onde o nome nele inscrito identifica claramente o objeto respectivo. Note que um objeto corresponde a uma instância de uma dada classe, e portanto os nomes seguem a norma de representar como "nome-do-objeto:nome-da-classe", onde eventualmente o objeto por ser parte de uma lista ou sequência, devendo nesse caso explicitar o seu índice. Não havendo necessidade de se identificar completamente a instância, pode-se usar a notação ":nome-da-classe".

A mensagem propriamente dita são as linhas ou setas que conectam os objetos, determinando a orientação emissor/receptor, e o seu título sobreposto ou próximo à seta. Pode ter um conjunto de números e letras antes do título indicando o sequenciamento (similar a seções ou capítulos de textos), com o símbolo ":" separando-os.

Pode ainda permitir a representação de concorrência, iteração ou condicionamento, porém, visando manter a simplicidade, tais detalhes comportamentais podem ser omitidos e melhor representados nos diagramas de sequência. Mensagens que são enviadas ao próprio objetos são representadas por setas que retornam ao objeto.

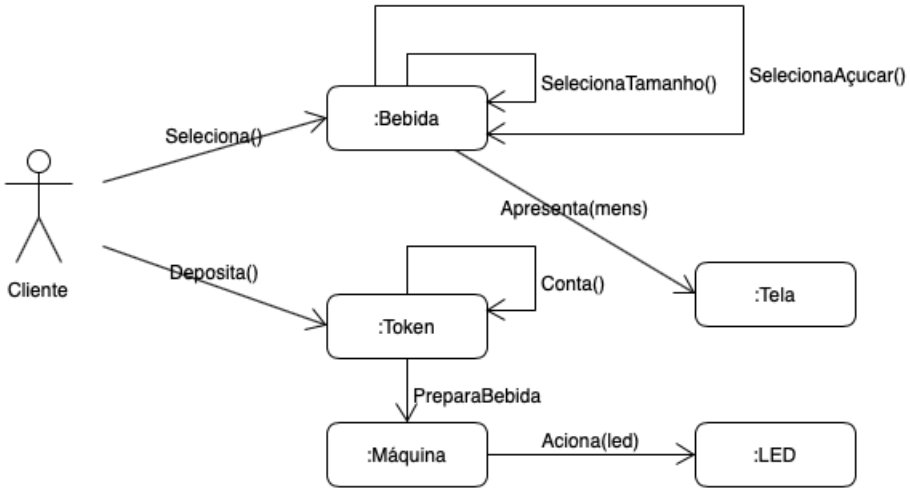

Figure 3: Diagrama de comunicações do exemplo

Foi usado no diagrama o símbolo do cliente como ator para iniciar o procedimento, com as duas alternativas já conhecidas. Note que o ator não envia mensagens diretamente, a sim através da interface homem/máquina, no caso o acionamento de três teclas ou a deposição de um token. Por simplicidade, optou-se por não representar nesse momento esses dois elementos do sistema, que se encontrariam entre o ator e os dois blocos que recebem a mensagem atual que inicia o procedimento. No entanto aqui já foram incluídas as mensagens que são enviadas à tela e aos LED's. Sã0 identificadas portanto 5 classes, Bebida, Token, Máquina, Tela e LED. Veja que devido aos dois pontos tratam-se no caso de instâncias dessas classes, que não precisam de fato de um nome. A instância de Bebida recebe uma seleção do seu tipo, inicialmente, e envia duas mensagens a si própria para selecionar o tamanho e a quantidade de açucar, enviando mensagens para tela para tanto. Tendo concluída a seleção da bebida, a mensagem a ser exibida vai pedir o depósito de um token. Assim o cliente deposita iniciando a contagem, também como uma automensagem. Observe que não foi explicitado o envio de mensagens entre o objeto :Token e a :Tela. Isso é na verdade uma falha, porque já que foi incluída a mensagem da :Bebida para a :Tela, também o segundo caso deveria ter ficado claro no desenho. Desse modo, poderiam ser cobrados novos tokens e ao final do pagamento informado que o valor adequado foi atingido. Ao ser concluído o pagamento a :Máquina inicia a preparação da bebida, tendo acionado o LED vermelho antes. Ao concluir a preparação, uma segunda mensagem deve ser enviada, agora para acionar o LED verde. Também poderia ter uma mensagem para a :Tela para informar

que a bebida está pronta e deve ser retirada. Todos essas mensagens servem para esclarecer a relação entre a instância da Tela e demais classes que serão necessárias ao projeto.

Deve-se ainda notar quais são as instâncias que deverão ser criadas, em quê momento, e quando elas devem ser destruídas porque não terão mais utilidades. Note finalmente que se o cliente iniciar pela deposição de um token, todo o procedimento continua viável de acordo como o esquema das mensagens, apenas alterando a ordem em que são emitidas. Por essa razão, nem sempre é conveniente colocar a ordem de forma explícita nesse diagrama.

6 Diagrama de Classes

Os diagramas de classes representam a estrutura de um dado sistema através de seus elementos principais, onde as classes são descrições abstratas desses objetos, representando os sujeitos que realizam ações ou que as sofrem. Uma classe compreende os seguintes elementos:

- Nome
- Atributos
- Operações

O nome é a própria definição classe, correspondendo assim a um substantivo nas orações (dos requisitos do projeto). Os atributos são os seus parâmetros descritivos, e as operações são as transformações e ações que esses parâmetros receberão. Para tanto, a cada operação deverá corresponder um módulo de programação, conhecido como método, que vai realizar a transformação respectiva. Sendo uma descrição abstrata, na execução dos programas serão criadas instâncias das classes para as quais serão atribuidos valores específico a seus parâmetros, através das operações que constam na classe.

As classes são representadas por retângulos que podem ser divididos em três partes, compreendo respectivamente seu nome, seus atributos e suas operações. Deve-se lembrar que normalmente ao usar a palavra objeto, está-se referindo a uma instância de uma dada classe, apesar que uma classe qualquer também é em si um objeto, uma instância da metaclasse *Classe*. As classes são assim uma definição à qual todos os seus objetos devem seguir. Em muitos casos, para uma dada classe, existe apenas um único objeto, uma instância necessária para que possa fazer parte do fluxo de execução.

Uma classe pode ter as seguintes relações com uma outra classe:

- Agregação
- Composição
- Generalização
- Realização
- Associação
- Associação direta
- Associação reflexiva
- Multiplicidade

Uma classe pode ser especializada em subclasses, que são também classes, mas que mantêm uma relação de dependência para com a chamada classe-mãe. É o caso da generalização. É representada por uma seta sólida apontando para a classe mãe. A realização é quando uma classe implementa um método definido em outra classe. É representada por uma seta de linha tracejada apontando para a classe que implementa o método. Ao se destruir (eliminar da memória) um objeto da classe mãe, os objetos da classe filha também são destruídos automaticamente. Todas as classes possuem um construtor, executado ao se criar uma instância qualquer, que não precisa ser representado nos diagramas.

Uma classe também pode ter por atributo uma outra classe, através da relação de composição. Significa que se uma dada instância da classe for destruída, a composição também o será. É representada por uma linha com um losango cheio na classe que contém a outra. Já na agregação, a destruição não ocorre com a classe agregada. É representada por uma conexão com um losango vazio na classe contenedora. A interação entre classes assim relacionadas se dá através da troca de mensagens, o quê também é o caso para classes que não se relacionam mas que podem colaborar para atingir um dado fim.

Uma associação simples é a conexão lógica entre duas classes mais leve, que não possue direção na linha que as conecta. Já a associação direta estabelece a relação de que uma classe contém outra, representada por uma seta sólida apontando para a classe contida. Associação reflexiva é quando múltiplas funções são acionadas dentro da mesma classe, sendo representada por uma linha em laço.

A multiplicidade ocorre quando existe cardinalidade entre duas classes, ou seja, múltiplas instâncias da classe contida podem fazer parte da contenedora.

Nesse caso, a linha que as conecta reflete essa repetição com números escritos próximos à classse múltipla.

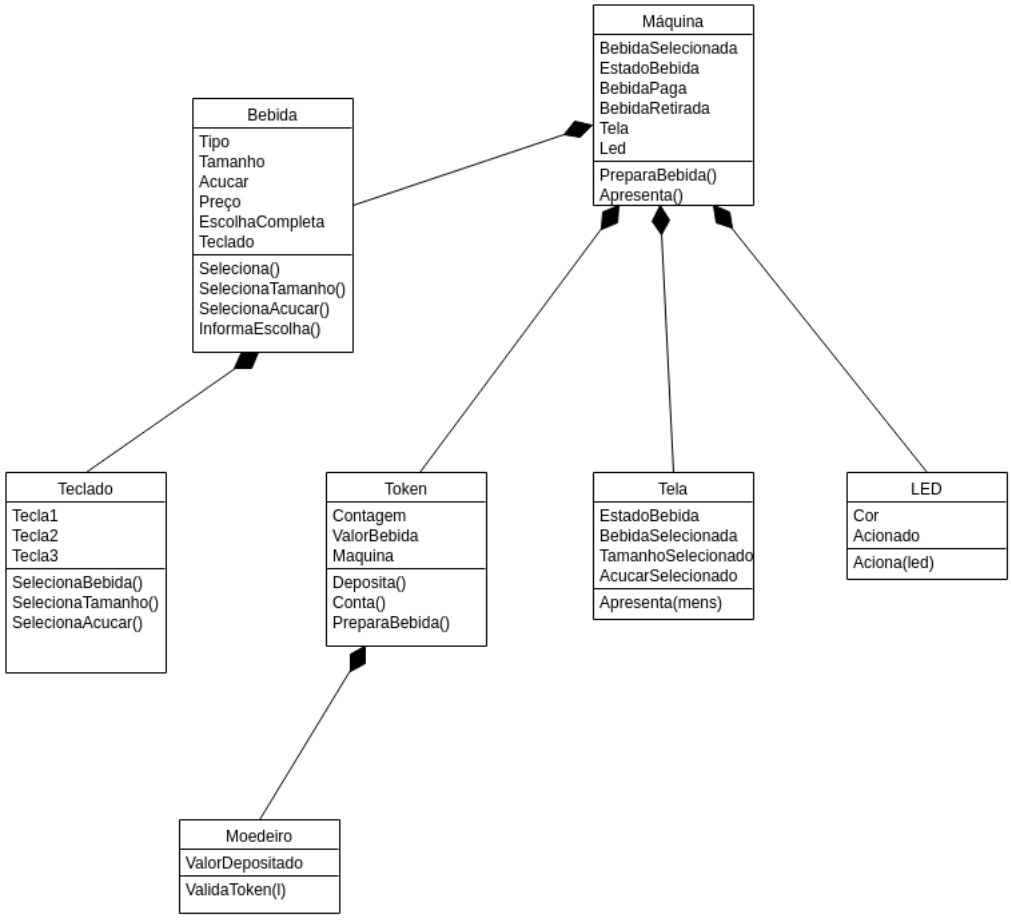

Figure 4: Diagrama de classes do exemplo

Observe que a classe Máquina é a estrutura principal, centralizando as demais. Isso porque ela é diretamente responsável por identificar a bebida desejada pelo cliente, conferir o pagamento, preparar e entregar a bebida. Vamos verificar como isso seria possível através da estrutura de classes apresentada, e se ela estaria correta ou deveria ser modificada. Note que nem todas as funções devem ser implementadas diretamente por essa classe, podendo serem usadas as relações acima explicadas para se atingir o seu objetivo central que é o de vender bebidas quentes.

Assim, note que agora existem duas novas classes além das que constam no diagrama de mensagens, que são: o Teclado, com três teclas, já que são três as opções possíveis em todos os casos; e o Moedeiro, responsável por receber e

validar cada token. Assim, parece que o número as operações das classes estão corretamente identificadas. Uma das ações da Máquina é identificar a seleção de bebida desejada pelo cliente, com as vinte e sete opções correspondentes. Veja que apenas uma instância da classe Máquina é necessária, que é a própria máquina, uma vez que trata-se de apenas uma. Essa deve existir durante todo o tempo de operação da máquina, sendo portanto criada em sua inicialização e destruída ao ser desligada. Da mesma forma, as instâncias do Teclado, do Moedeiro, da Tela e da classe LED são únicas e podem permanecer vivas durante toda a operação. Já as instâncias da Bebida e de Token devem ser criadas a cada processo de venda de uma dada bebida, e destruídas após isso. Eventualmente apenas um registro da venda deve ser mantido para se ter a contabilidade da máquina, mas isso está fora do escopo do nosso problema. Portanto cinco instâncias são permanentes e duas são circunstanciais.

Para identificar a bebida desejada, a :Máquina, após inicializada, deve apresentar uma mensagem inicial na tela informando as opções do cliente, e passar então a monitorar o :Teclado e o :Moedeiro. Dependendo de por onde o cliente começar o procedimento, será inicialmente instanciada uma bebida e selecionadas todas as opções através de mensagens na tela e do :Teclado. Em seguida a :Máquina inicia o procedimento de pagamento, instanciando a classe Token, que vai monitorar diretamente o :Moedeiro. Se o cliente iniciar pelo :Moedeiro, este aciona a classe Token que será instanciada e enviará mensagem à :Máquina para a identificação da bebida. Em seguida, se necessário a solicitação de novos tokens será efetivada pela classe. Tendo sido concluídas a seleção de bebidas e o pagamento, a :Máquina prepara e serve a bebida, enquanto controla as mensagens respectivas enviadas ao cliente.

Há vários aspectos a considerar aqui. Inicialmente, pode-se notar que as classes Bebida e Token relacionam-se por agregação com a Máquina, uma vez que suas instâncias serão destruídas por mensagens explícitas a cada venda. Já as demais terão o tempo de vida igual ao da :Máquina, podendo ser composição. Mas apenas Tela e LED são seus componentes diretos. As outras duas estão associadas respectivamente Teclado à Bebida e Moedeiro à Token. Nesse caso elas não podem ser uma composição, uma vez que suas instâncias não serão eliminadas se as duas principais o forem. Isto não corresponde ao que está representado no diagrama.

Existe ainda uma decisão a ser tomada, em relação às duas classes componentes da Máquina. Normalmente, elas apenas seriam acessadas através da classe mãe. Porém pode ser mais simples que as classes Bebida e Token pudessem enviar mensagens a Tela e a LED. Isso levaria a uma associação direta entre essas classes, ou a uma associação simples. Isso pode ser deixado para se averiguar posteriormente, com o andamento do projeto.

Finalmente, deve-se averiguar se todas as mensagens que serão trocadas estão corretamente explicitadas nas classes correspondentes, bem como suas operações.

7 Diagrama de Sequência

Trata-se de um diagrama comportamental que apresenta a sequência que deve ser seguida para a implementação do sistema ou de uma parte dele. Ou seja, qual a interação entre um grupo de objetos que cooperam para atingir um dado fim. Apresenta assim os detalhes de um caso de uso, representando os objetos envolvidos e a sequência das operações respectivas.

São usados os seguintes símbolos:

- Ator
- Objeto
- Caixa de ativação
- Linha da vida
- Iteração
- Opção
- Alternativa
- Sincronismo
- Assincronismo
- Retorno
- Criação
- Final de vida

O ator é o mesmo símbolo do caso de uso e pode ser usado para dar início a um processo, refletindo a dinâmica descrita no caso de uso.

O objeto são as instâncias das classes que são representados por retângulos em paralelo, de onde saem linhas verticais que representam a vida dos métodos que são acionados pelas mensagens que ocorrem entre eles. A cada mensagem recebida, uma caixa de ativação é criada na linha da vida do objeto respectivo. Um objeto pode iniciar um processo, desse modo é criado uma caixa

de ativação desse método inicial. Ou um ator pode ser usado para representar o início do processo, enviando uma mensagem a partir de sua linha da vida, sem a caixa de ativação respectiva, que deve corresponder apenas a um módulo de computação sendo executado. Ainda, é possível iniciar um processo de um ponto indeterminado, representado por um pequeno círculo negro. Pode ser o caso por exemplo de uma falha de hardware.

Desse modo, uma caixa de ativação envia uma mensagem, de forma síncrona (uma seta de ponta cheia) ou assíncrona (uma seta de ponta aberta). No primeiro caso, o emissor da mensagem fica aguardando o retorno respectivo, enquanto no segundo o processo continua. O retorno da mensagem é representado por uma seta aberta tracejada.

Os casos de iteração, opção ou alternativa, são representados por um retângulo com rótulo que contem todas as caixas de ativação envolvidas. Para as alternativas, o retângulo contem linhas horizontais que as separam.

A mensagem de criação, representada por uma seta tracejada aberta, é enviada para instanciar um novo objeto de uma dada classe.

O símbolo de final de vida é usado para determinar a destruição explícita do objeto respectivo.

Finalmente, uma mensagem pode ser enviada ao próprio objeto.

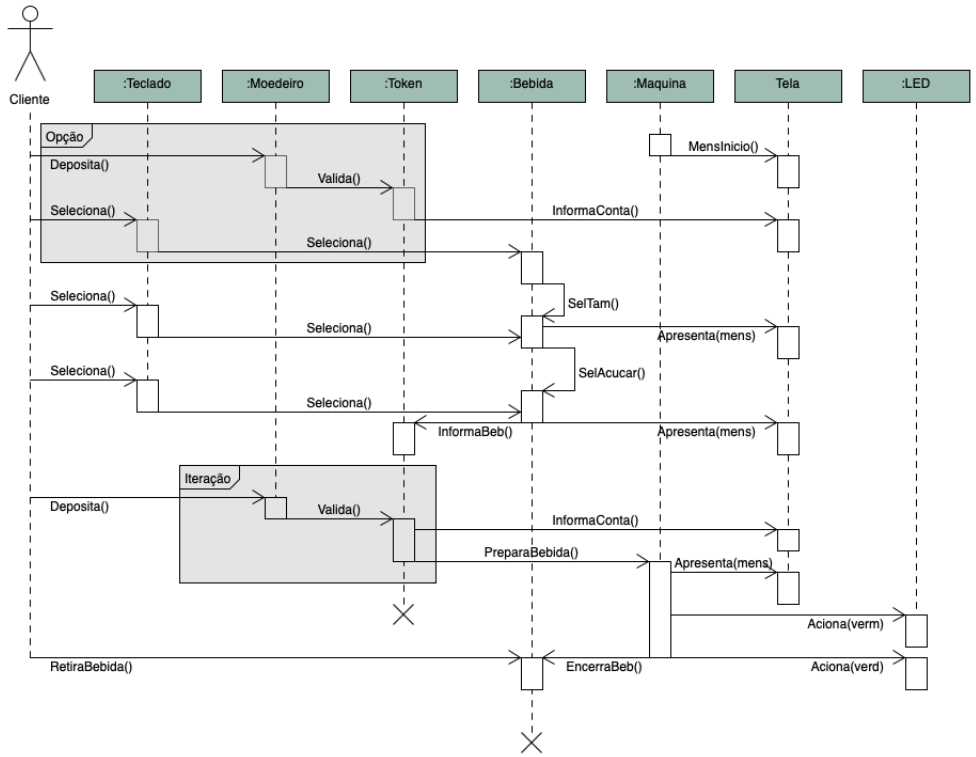

Figure 5: Diagrama de sequência do exemplo

Vamos verificar, fazendo uma leitura do diagrama, se todo o sequenciamento de operações está corretamente descrito no diagrama de sequência, onde o tempo de execução está finalmente sendo considerado. Note que isso é feito pela altura das caixas de ativação e a posição de suas mensagens.

O procedimento é iniciado pelo cliente, com as duas opções, depósito do token no :Moedeiro ou selecionar pelo :Teclado o tipo da bebida. Note que a :Máquina já terá enviado a MensInicio(mens), para aguardar a ação do cliente. Se for depositado o token, o :Modedeiro o valida e aciona o :Token para começar a contagem, e informa a conta e solicita a seleção do tipo na :Tela. Tendo feito essa primeira seleção, as outras duas seguem através de duas outras mensagens que essa envia para a tela. Note que há uma discrepância de tempo e falta uma outra mensagem entre a bebida e a tela. A primeira para a seleção do açucar e a segunda para informar que a bebida foi completamente selecionada. Uma outra mensagem é enviada pelo :Token, informando o estado do pagamento, e podendo sofrer iteração até completá-lo. Veja que a sequência será a mesma qualquer que seja opção de início. Ao concluir o pagamento, há a mensagem enviada pelo :Token para a máquina que inicia a preparação da bebida e avisa o cliente pela tela e aciona o LED

vermelho, e em seguida a destruição da instância do :Token. Ao concluir a preparação, o LED verde é acionado também está faltando a mensagem da máquina para a tela informando a bebida pronta. Quando o cliente retira a bebida a instância respectiva é destruída.

Note-se pelo diagrama acima que as mensagens foram passadas diretamente para a tela pelas classes envolvidas, evitando assim a passagem pela classe Máquina.

8 Diagrama de Estados

O diagrama de máquina de estados é um diagrama estrutural que representa a natureza dinâmica de um sistema, considerando os diversos estados que estabelecem sua condiçao de operação, o que identifica como o sistema reage a cada estímulo externo recebido. Tendo sido determinados os diversos estados que descrevem a operação do sistema, o diagrama de estados descreve como o fluxo de controle transita entre eles, porém sem o detalhamento das funçoes que são executadas associadas a cada estado. Ou seja, trata-se de um diagrama transverso ao diagrama de classes.

Para elaborar um diagrama de estados são necessárias a identificação do objetos do sistema, de seus estados e dos eventos que podem causar a transição entre os estados. Uma máquina de estados é um conceito geral de projeto de um sistema qualquer, e possue sempre um estado inicial por onde o processo começa, e um estado final onde o processo é encerrado. A representação desses é a mesma usada em outros diagramas. Deve contar ainda com uma representação dos estados, normalmente um retângulo com os cantos arredondados, e com setas que indicam a transição entre estados, onde descreve-se o evento que propiciou a transição. Faz uso também do elemento de decisão, representado pelo losango, bem como pode incorporar o uso de barras para a criação de concorrência ou de sincronização entre estados. Também deve-se ressaltar a autotransição, ocorrida de um estado para ele mesmo. Finalmente, podem ocorrer o uso de *guardas* em uma transição (condições necessárias para que a transição possa ser efetivada), e a execução de operações que ocorrem na entrada em um estado, na saída de um estado, ou ainda na própria transição entre dois estados.

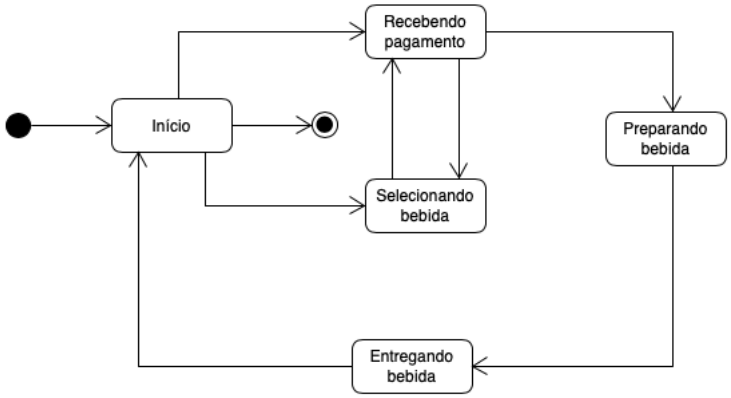

Figure 6: Diagrama da máquina de estados do exemplo

O diagrama da máquina de estados é bem simples,apresentando apenas quatro estados, além da partida e do encerramento. Partindo, o que corresponde à ligação da máquina, o estado de "Início" procede à toda inicialização dos parâmetros e criação das instâncias permanentes, que apenas serão destruídas antes do encerramento. Poderíamos ter então o estado de encerramento, mas esse papel pode ser feito pelo de início. A inicialização termina pela mensagem inicial onde a máquina aguarda o primeiro cliente. Assim, dependendo da ação do cliente, o próximo estado pode ser o ""Recebendo pagamento" ou o "Selecionando bebida", sendo que há as necessárias transições entre esses dois estados. Ao concluir o pagamento, a máquina passa para o estado de "Preparando bebida", cujos detalhes estão fora do escopo do trabalho. Ao terminar a bebida, passa para o estado "Entregando bebida" e daí retorna ao início. Note que cada estado é reponsável por ações de entrada e saída do estado pelo menos, o que inclui as mensagens para a tela, que não estão explicitadas.

Deve perceber que a máquina de estados é uma forma de implementação, e cada estado pode conter instâncias de uma ou mais classes. Trata-se na verdade de uma forma racional de agrupar as funções que são desempenhadas pelas instâncias do problema.

De modo geral, ao chegar nesse ponto, deve-se refazer os diagramas para corrigir as falhas eventuais e, feito isso, iniciar a implementação.