Análise Orientada a Objetos

Linguagem de Modelagem Unificada – UML

A UML, cuja sigla em inglês significa Unified Modeling Language, ou Linguagem de Modelagem Unificada, é uma das principais ferramentas de modelagem utilizadas em empresas de desenvolvimento de software. A UML foi evoluindo ao longo dos anos, demonstrando que é uma linguage sólida, bem estruturada, que teve seu desenvolvimento iniciado por pessoas relevantes na área de desenvolvimento de software orientado a objetos, como podemos ver a seguir:

- **1967:** Surgimento da linguagem Simula, a primeira considerada como orientada a objetos.
- **1979:** Surgimento da linguagem ADA, linguagem orientada a objetos do departamento de defesa dos Estados Unidos.
- **1986:** Apresentação do trabalho de Grady Booch sobre análise orientada a objetos.
- **1991:** Apresentação do método OMT de James Rumbaugh para modelagem de bancos de dados.
- **1996:** Chamada do OMG (Object Management Group) para um padrão unificado de modelagem.
- **1997:** Envio da primeira versão da UML como resposta à chamada pelos autores Grady Booch, Ivar Jacobson e James Rumbaugh. UML 1.1.
- **2001:** UML 1.4.
- **2003:** UML 1.5.
- **2005:** UML 2.0.
- **2006:** UML 2.1.
- **2007:** UML 2.1.2.
- **2010:** UML 2.3.
- **2011:** UML 2.4.1.
- **2015:** UML 2.4.1.
- **2017:** UML 2.5.1.

A linha do tempo mostra algumas questões importantes sobre o desenvolvimento da modelagem.

Necessidade de um Processo de Modelagem

Entre a criação da linguagem Simula e o início da preocupação com a modelagem de software passaram-se quase 20 anos de desenvolvimento, tempo necessário para perceber que era necessário utilizar um processo de desenvolvimento de software que incluísse a modelagem.

Versão mais Eficiente

Entre a primeira versão de 1997 e a segunda versão de 2005 são 8 anos de diferença. Este tempo foi necessário para que os desenvolvedores entendessem as necessidades reais da linguagem e pudessem apresentar uma versão mais eficiente e adequada ao desenvolvimento de software atual.

Ferramenta Madura

A partir da versão 2.1.2 o intervalo entre modificações ficou mais longo e a última versão é de 2015. Isso mostra o amadurecimento da ferramenta que não precisou mais ser modificada com tanta frequência por atender aos requisitos dos desenvolvedores.

A UML é uma linguagem que apresenta um número muito mais de vantagens do que desvantagens, porém existem muitos desenvolvedores relutantes em utilizá-la. Os principais motivos para que isso ocorra são:

- Falta de conhecimento.
- Falsa impressão.
- Uso de modelo próprio.

Os problemas da não utilização da ferramenta tender a ser: Falta de informação e pré-julgamento. Portanto é interessante que se abordem justamente estes problemas quando a intenção é sugerir sua implantação.

A linguagem UML tem como vantagem ser open-source, ou seja, ser uma ferramenta livre de custos de utilização e implantação. Os custos associados à utilização da UML irão envolver possíveis treinamentos que sejam necessários e a escolha de ferramentas de criação dos diagramas que sejam proprietárias. Neste caso seria interessante entender o que seria um software livre e os tipos de licenças que podem ser atribuídas. Dentre os movimentos de software livre mais importantes, a referência sugerida para o estudo é do projeto GNU criado por Richard Stallman.

Classificação dos Diagramas UML

A UML é uma importante ferramenta para modelagem de sistema que dispõe de uma ampla lista de funcionalidades, neste caso estamos falando dos diagramas, com 14 tipos diferentes.

É importante entender o objetivo principal de cada um para sua utilização, assim, para ficar mais simples encontrar o diagrama necessário para um determinado problema, a linguagem apresenta uma classificação de tipos de diagramas:

- **Estrutural:** Os diagramas classificados como estruturais modelas as características referentes a sua estrutura, ou seja, como são organizadas as partes do sistema, como são os módulos, como são as classes.
- **Estático:** Os diagramas estáticos serão feitos e não serão modificados, pois as informações utilizadas para criá-los não mudam de acordo com a implementação do sistema.
- **Comportamental:** Os diagramas classificados como comportamentais modelam o comportamento dos objetos e componentes, ou seja, como se comportam em tempo de execução e como se relacionam com os outros componentes do sistema.
- **Dinâmico:** Os diagramas dinâmicos representam o estado do sistema em um determinado momento, seja de tempo de execução, seja de desenvolvimento. Portanto, ao longo do processo sofrem alterações.

Os diferentes diagramas da linguagem UML informam diversos aspectos de um mesmo projeto, assim devemos considerar a:

- **Utilização dos Diagramas:** Sua utilização permite que seja feita uma análise completa do software que está sendo modelado. Porém, nem sempre há tempo para o desenvolvimento de todos os diagramas durante um projeto.
- **Escolha dos Diagramas:** É de extrema importância conhecer os diagramas e sua classificação, pois sabendo a informação principal apresentada por cada um deles, é possível fazer uma escolha consciente de qual diagrama utilizar – pois não se pode desenvolver todos eles e também a representação do sistema, para os desenvolvedores, não ser prejudicada.

- **Ordem dos Diagramas:** É importante também saber a ordem de desenvolvimento, pois determinados diagramas não são utilizados em todas as fases do projeto. É preciso, então, ficar atento em qual fase o projeto se encontra na hora de determinar quais diagramas serão elaborados.

Existem diversas ferramentas que podemo ser utilizadas para o desenvolvimento de diagramas UML, mas devido às suas funcionalidades semelhantes é difícil, em alguns casos, determinar qual seria a mais adequada para dada situação. Além disso, há ferramentas com direitos reservados e ferramentas que podem ser utilizadas sem custo. Uma análise de quatro ferramentas, freeware, de utilização para geração de diagramas UML é feita por Costa, Werneck e Campos (2008). A leitura do artigo não só apresentará a melhor linguagem UML como também uma manteira interessante de comprar ferramentas de desenvolvimento.

Mecanismos Comuns da UML

Existem 14 diagramas disponíveis para serem utilizados na linguagem UML e todos eles apresentam particularidades. Entretanto, há alguns mecanismos que são comuns a todos os diagramas e podem ser utilizados para melhorar a legibilidade, bem como incluir mais informações sobre o modelo.

A seguir, conheça os quatro mecanismos comuns:

Especificações

A UML é uma linguagem gráfica, porém associada a cada elemento dessa linguagem gráfica existe uma especificação que descreve exatamente aquele elemento.

Por exemplo: Relacionada a uma classe existe toda a especificação que descreve os atributos, operações e comportamentos que a classe incorpora. Visualmente, o elemento da classe mostrará apenas parte dessa informação.

Basicamente, a parte visual permite um entendimento rápido e global de cada parte do sistema, enquanto a especificação permite especificar os detalhes.

Adornos

Cada diagrama UML começa com um símbolo e depois os adornos são adicionados ao diagrama para compor os elementos da modelagem.

Um adorno, portanto, é um elemento do diagrama UML que tem uma arte gráfica única e direta, tornando-se uma representação visual objetiva daquele elemento, como a classe.

No exemplo estão os elementos principais da classe, ou seja, seu nome, seus atributos e métodos. Nada mais é necessário para o entendimento global do diagrama de classes, porém essa representação é um adorno da UML.

Divisões Comuns

Os blocos de construção dos diagramas UML apresentam, em quase todos os casos, uma dicotomia entre sua interface e implementação.

- A interface é como um “contrato”.
- A implementação é uma das possíveis realizações desse contrato responsável por complementar a semântica da interface.

Essa estrutura é a base do polimorfismo em linguagens orientadas a objetos, quando se define uma interface com vários métodos que serão implementados apenas nas subclasses.

Mecanismos de Extensão

Foram criados para permitir que sejam feitas modificações na linguagem sem a necessidade de mudar toda a linguagem.

- **Estereótipos:** É possível, na UML, utilizar o “desenho” de um determinado bloco e modificá-lo para um propósito específico, criando um novo objeto.
- **Restrições:** Também é possível, alterar as restrições na construção de um diagrama. Em UML, as restrições são representadas pelas strings que acompanham as ligações entre elementos.
- **Valores Predefinidos:** É possível predefinir valores específicos em um diagrama para guiar a implementação do sistema ou gerenciamento de configurações do sistema.

Consistência Entre os Diagramas

A consistência entre diagramas UML é um problema e deve ser sempre considerada quando a linguagem é utilizada. Diagramas inconsistentes podem causar problemas graves de desenvolvimento do software gerando erros no produto final e retrabalho na parte do desenvolvimento. Apesar de existirem regras para a manutenção da consistência entre os diagramas, elas em si não garante que a consistência seja alcançada, sendo importante que a construção dos diagramas seja feita em conjunto pela equipe de desenvolvimento.

Um aspecto relevante nesses casos é que toda a equipe de desenvolvimento deve conhecer a linguagem para poder construir e analisar de forma correta os diagramas. Isso mostra como é importante, em uma empresa, não só a utilização da linguagem em si, como a promoção de constantes treinamentos a respeito da ferramenta, para que os desenvolvedores evoluam seus conhecimentos em conjunto e possam obter resultados de utilização cada vez melhores.

Diagramas UML

A linguagem UML é aplicável a qualquer método de desenvolvimento de software, porém existe uma vertente de desenvolvimento ágil, que diz que não utiliza UML pelo tempo gasto na elaboração dos diagramas. É interessante entender que uma metodologia ágil de desenvolvimento tem com objetivo entregar um produto com alta qualidade em um curto período.

Sendo assim a utilização de diagramas UML é indicada nesse caso justamente por diminuir o retrabalho durante o desenvolvimento do software por erros gerados no processo e, consequentemente, criar um produto melhor em menor tempo. Portanto, cuidado com as afirmações sobre utilização ou não da linguagem UML, pois, na maioria das vezes, a questão é apenas falta de conhecimento em relação à linguagem.

Diagramas de Caso de Uso

O diagrama de casos de uso é o diagrama responsável por descrever um conjunto de ações que os sistemas devem executar em conjunto com usuários externos ao sistema. Ele que irá modelar todas as possíveis utilizações do sistema de uma forma simples e de fácil entendimento, inclusive é utilizado em reuniões com o cliente para verificação.

Com um diagrama de casso de uso bem feito, é possível obter êxito no desenvolvimento de vários outros diagramas da linguagem UML.

Diagrama de Classes

O diagrama de classes da UML é um diagrama estrutural, que tem como objetivo principal ilustrar graficamente a estrutura do software, em níveis mais e menos abrangentes. Além disso, o diagrama de classes mostra como se dá a interligação entre os componentes da estrutura do sistema (UML-Diagrams, 2016).

Diagrama de Objetos

O diagrama de objetos foi definido inicialmente na versão 1.4.2 da UML, que atualmente está ultrapassada. Foi inicialmente definido como um grafo de instâncias que inclui objetos e valores de dados, Um diagrama estático de objetos é uma instância de um diagrama de classes. Já na UML 2.5 a relação entre diagrama de classes e de objetos não é apresentada. Algumas outras fontes sobre UML classificam os diagramas de componentes e de diagramas de desenvolvimento apenas com instâncias como tipos especiais de diagramas de objetos (UML-Diagrams, 2016). Sendo assim, é possível ver como a evolução da UML pode modificar consideravelmente o papel dos diagramas na modelagem de sistemas.

O diagrama de objetos é utilizado para representar os objetos instanciados de uma classe. Ele utiliza notação de objetos para a própria construção.

Esse diagrama tem por objetivo representar os objetos e os relacionamentos entre eles e nada mais é do que um grafo de instâncias, incluindo os objetos e os valores de seus atributos (UML-Diagrams, 2016).

A seguir, veja a relação entre os diagramas de classe e objetos:

- **Diagrama de Classes:** Para um sistema de música.
- **Interface:** Com as funções básicas que devem estar disponíveis para o sistema, que são tocar, parar, pausar e reiniciar uma música.
- **Classes:** As classes aparelho de DVD e aparelho de CD implementam a interface, cada uma para uma mídia diferente.
- **Herança:** Representada entre a classe aparelho de som e gravador de som, já que se entende um gravador como um aparelho de som que possui a função de gravar.
- **Classes que Podem ser Instanciadas:** Subclasses derivadas da superclasse (classe-mãe) aparelho de som.
- **Diagrama de Objetos:** O diagrama de objetos relacionado ao diagrama de classes irá conter vários objetos isolados, cada um com um tipo diferente de aparelho sem relacionamentos.

É possível perceber a relação entre os diagramas e a existência de casos em que a utilização do diagrama de objetos irá apenas reforçar o que já existe no diagrama de classes sem apresentar relações novas.

O diagrama de classes é um diagrama que possui diversos casos específicos e pode ser bem complexo de acordo com o sistema a ser modelado. O artigo de Bell (2016) apresenta a visão de um profissional da IBM, empresa de desenvolvimento de hardware e software criada em 1911 e consolidada no mercado, que é interessante por abordar o assunto de forma prática.

Diagrama de Atividades

O diagrama de atividades, basicamente, consiste em um gráfico na forma de fluxo que mostra as características dinâmicas do sistema, podendo ser visualizado o fluxo de ações que serão realizadas em uma determinada atividade relacionada ao sistema que será desenvolvido.

O diagrama de atividades é importante porque permite que as atividades e os fluxos de controle do sistema sejam representados em um diagrama e, posteriormente, podem auxiliar no desenvolvimento do sistema.

A seguir veja os principais componentes do diagrama de atividades:

- **Círculo Preenchido:** Símbolo de início do fluxo de controle no diagrama de atividades.
- **Retângulos com Cantos Arredondados e com Texto Dentro:** Símbolo que representa uma atividade, que é um comportamento parametrizado representado por uma ação.
- **Barra Preenchida com uma Entrada e Mais de uma Saída – Fork:** Símbolo utilizado quando um fluxo gera duas atividades em paralelo. Usado, também, para indicar sincronização da paralelização das atividades.
- **Barra Preenchia com várias Entradas e Apenas uma Saída – n:** Símbolo utilizado para sincronizar o final de fluxos paralelos que deverão se agrupar novamente em um mesmo fluxo.
- **Losango:** Símbolo de decisão que, assim como em fluxogramas, representa a possibilidade de seguir por dois caminhos. Normalmente associado a uma condição, caso a condição seja satisfeita, um dos caminhos é seguido, caso não seja, o outro caminho é seguido.
- **Círculo Preenchido com uma Borda Extra:** Símbolo de encerramento de atividade, representa o final da atividade que está representada no diagrama.

Os diagramas UML contêm diversos componentes, os quais, muitas vezes, não são utilizados ou são desconhecidos até mesmo por quem utiliza os diagramas. Um estudo feito por Reggio et al. (2014) buscou verificar, em quatro fontes principais (livros, tutoriais, ferramentas e cursos), quais dos componentes do diagrama de atividades são os mais utilizados. Como resultado, foram selecionados 9 de 47 possíveis, os quais são: Action, Control Flow Edge, Initial/Final Node, Decision/Merge Nodes, Fork/Joint Nodes, Swinlane, Object Node e Object Flow Edge. Os outros não são menos importantes, porém aparecem em uma quantidade pouco significativa de cursos e materiais.

O diagrama de atividades não é a única ferramenta para modelagem desta natureza. Empresas estão buscando a utilização de um processo de negócios adequado e uma linguagem de modelagem para representar estes processos, buscando melhores resultados em todos os setores. Duas notações muito utilizadas são a modelagem e notação de processos (BPMN) e o diagrama de atividades UML. O artigo de Geambasu (2012) apresenta um comparativo entre as duas representações, seus pontos fortes e fracos.

Diagrama de Máquina de Estados

O diagrama de máquina de estados demonstra o comportamento de um elemento por meio de um conjunto de estados relevantes e suas transições de estado. O elemento modelado, muitas vezes, é

uma instância de uma classe, o comportamento de um caso de uso ou o comportamento de um sistema completo. É considerado um estado relevante ou importante ao contexto do sistema aquele que implica em ações a serem consistidas durante a execução do sistema.

Na prática, o diagrama de máquina de estados é recomendado, principalmente, para modelar os estados dos objetos das classes.

A seguir vamos conhecer mais sobre a notação gráfica e os elementos que constituem o diagrama de máquina de estados:

O diagrama de máquina de estados é uma técnica de modelagem comportamental da Unified Modeling Language (UML) que permite descrever o ciclo de vida de objetos de uma classe, os eventos que causam a transição entre os estados e a realização de operações resultantes.

Elementos básicos do diagrama:

- **Estado (State):** Representa uma situação de existência dos objetos de uma classe durante a qual ele satisfaz alguma condição ou realiza alguma atividade.
- **Transição (Transition):** Representa um relacionamento entre dois estados, indicando a mudança de estado, a partir da ocorrência de um evento.
- **Estado Inicial (Initial State):** Representa o estado de um objeto quando ele é criado, indicando o estado padrão que o objeto assumirá. Só pode haver um estado inicial na máquina de estados.
- **Estado Final (Final State):** Representa o fim do ciclo de vida de um objeto. Este estado é opcional e pode haver mais de um estado final na máquina de estados.
- **Escolha (Choice):** Representa um ponto na transição de estados de um objeto em que deve ser tomada uma decisão. As transições de um estado de escolha devem ser indicadas com uma condição de guarda.

Ações de Estado

Uma representação mais detalhada dos estados dos objetos consiste na indicação das atividades internas, também denominadas de ações de estado, e ainda apresentar as transições internas dos estados. As ações de estado são representadas pelas cláusulas predefinidas “entry, exit e do” no interior do retângulo do estado, sendo:

- **Do:** Representa uma atividade realizada durante o tempo em que o objeto se encontra no estado.
- **Entry:** Representa as ações realizadas no momento em que o objeto assume o novo estado.
- **Exit:** Representa as ações executadas quando o objeto está mudando de estado.

Uma atividade interna está sempre associada ao estado que o objeto assumiu, ou seja, corresponde aos métodos executados pelo objeto, porém não causa alteração na situação do estado.

Na elaboração do diagrama de máquina de estados, é fundamental identificar as regras de negócio aplicadas aos contextos dos objetos, a fim de auxiliar na definição dos seus estados e das suas transições. Para definir as transições entre os estados, deve-se identificar os eventos internos e externos aos objetos da classe e analisar se há algum fator que condicione a transição de estado, nesse caso, deve-se representar por meio da indicação de condições de guarda.

Diagramas de Interação

Os diagramas de interação mostram como os objetos do sistema agem internamente para apoiarem a realização das funcionalidades representadas pelos casos de uso, consolidando, assim, o entendimento dos aspectos dinâmicos do sistema. Uma das formas mais utilizadas para especificar a interação entre os objetos é a ênfase à ordenação temporal das mensagens, representando a sequência lógica da troca de mensagens formada por um conjunto de objetos e seus relacionamentos a partir da adoção do diagrama de sequência.

Diagrama de Sequência

O diagrama de sequência representa a ordem temporal em que as mensagens são trocadas entre os objetos envolvidos na execução de um processo.

O diagrama de sequência baseia-se no diagrama de casos de uso e elabora normalmente um diagrama de sequência para cada caso de uso, apoiando-se no diagrama de classes para determinar os objetos das classes que realizam o caso de uso, com a indicação das mensagens trocadas entre os objetos que são, na maioria das vezes, as operações das classes.

A seguir veja a interação entre os elementos do diagrama de sequência com os tipos de mensagens possíveis, a notação gráfica dos estereótipos das classes e utilização de fragmentos de interação e fragmentos combinados que possibilitam o alinhamento de interações, sendo que cada fragmento representa uma interação independente, formando uma fronteira entre os elementos do diagrama.

O diagrama de sequência tem o objetivo de representar a ordem temporal em que as mensagens são trocadas entre os objetos envolvidos na execução de um processo, que foi especificado como um caso de uso. Graficamente, é uma tabela que mostra objetos distribuídos no eixo X e mensagens, em ordem crescente no tempo, no eixo Y.

Elementos básicos:

- **Ator:** Representa os mesmos atores já criados no diagrama de casos de uso, são apoiados por uma linha de vida e enviam mensagens para os objetos como uma forma de interação para solicitarem a execução de uma operação ou simplesmente o envio de informações. No diagrama sempre representa o ator primário responsável por enviar a mensagem inicial, que começa a interação entre os objetos.
- **Objeto:** Representa os objetos que participam da realização do caso de uso, são também apoiados por uma linha de vida, que juntamente com os atores, forma um cabeçalho para o diagrama. Um objeto pode existir desde o início da interação ou ser criado ao longo da dela. Um objeto é representado por um retângulo com um nome único, conforme o padrão da notação de objeto.
- **Linha da Vida:** Representa os objetos que participam da realização do caso de uso, são também apoiados por uma linha de vida, que juntamente com os atores, formam um cabeçalho para o diagrama. Um objeto pode existir desde o início da interação ou ser criado ao longo da dela. Um objeto é representado por um retângulo com um nome único, conforme o padrão da notação de objeto.
- **Mensagem:** Representa os objetos que participam da realização do caso de uso, são também apoiados por uma linha de vida, que juntamente com os atores, formam um cabeçalho para o diagrama. Um objeto pode existir desde o início da interação ou ser criado ao longo da dela. Um objeto é representado por um retângulo com um nome único, conforme o padrão da notação do objeto.
- **Foco de Controle:** Representa os objetos que participam da realização do caso de uso, são também apoiados por uma linha de vida, que juntamente com os atores, formam um

cabeçalho para o diagrama. Um objeto pode existir desde o início da interação ou ser criado ao longo da dela. Um objeto é representado por um retângulo com um nome único, conforme o padrão da notação do objeto.

Estereótipo das classes:

- **«boundary»:** Denominado de classe de fronteira, é aquele que representa a interface do sistema, indicando a comunicação entre o ator primário e os demais objetos das classes que participam da interação. O estereótipo do tipo «boundary», denominado de classe de fronteira, é aquele que representa a interface do sistema, indicando a comunicação entre o ator primário e os demais objetos das classes que participam da interação.
- **«control»:** Denominado de classe de controle, serve de intermediário entre as classes definidas como «boundary» e «entity» para tratar as regras de negócio e o fluxo da aplicação.
- **«entity»:** Denominado de classe de entidade, é aquele que mostra que as classes do sistema também são entidades, seja persistente ou transiente durante a execução do sistema.

Fragmentos:

- **Fragmentos de Interação:** Na notação gráfica do quadro de interação, identifica-se o rótulo com a expressão “ref” e, no centro do quadro, descreve-se o nome do diagrama de sequência referenciado.
- **Fragmentos Combinados:** Um fragmento combinado é utilizado para definir o fluxo de controle da interação, correspondendo a uma sequência de mensagens encapsuladas em um fragmento, compondo um procedimento que pode ser reutilizado em demais diagramas de sequência. Os fragmentos combinados são representados pelo elemento quadro de interação, com uma identificação no rótulo, que descreve o tipo de operador de interação, que pode ser (Bezerra, 2014):

    - **“alt”:** Abreviatura de alternative (alternativa). Modela a construção procedimental do tipo se-então-senão, ou seja, uma escolha entre duas ou mais ações, sendo que o quadro de interação é dividido em partes por uma linha tracejada, representado cada operador com uma condição de guarda (texto entre colchetes que estabelece uma condição ou uma regra). Apenas um operador do quadro é executado.
    - **“opt”:** Abreviatura de option (opção). Modela a construção procedimental do tipo se...então, sendo que o operador opt representa uma escolha de comportamento que será ou não executado a partir de uma condição de guarda.
    - **“loop”:** Abreviatura de looping (laço). Representa que uma interação deve ser realizada zero ou mais vezes conforme indicação da expressão com os limites mínimo e máximo, definindo a quantidade de repetições.

Diagramas de Interação

A seguir, veremos as principais características a modelagem dos demais diagramas de interação da UML, o diagrama de comunicação, o diagrama de visão geral de interação e o diagrama de tempo.

Diagrama de Comunicação

O diagrama de comunicação representa o relacionamento entre os objetos envolvidos na realização de um caso de uso, enfatizando o sentido da troca de mensagens entre os objetos que participam de

uma interação. O diagrama de comunicação não demonstra a temporalidade da realização de um processo.

A leitura do diagrama de comunicação deve ser conduzida pela ordem de envio de mensagens entre os objetos, acompanhando o rótulo das mensagens, que descreve uma expressão de sequência obrigatória, incluindo a numeração da mensagem, as informações enviadas e também um elemento de controle, como uma condição de guarda, se necessário. O sentido da mensagem é indicada por uma seta posicionada próxima ao rótulo da mensagem, apontando para o objeto receptor da mensagem.

A ênfase do diagrama de comunicação está em demonstrar exatamente a ligação entre os objetos representados pela lifeline, que participam da realização de um caso de uso.

Diagrama de Visão Geral de Interação

O diagrama de visão geral de interação é um novo diagrama da UML 2.0. É uma variação do diagrama de atividades que integra os diagramas de interação, principalmente o diagrama de sequência, demonstrando um processo geral. Com o diagrama de visão geral de interação é possível ter uma visão de alto nível das interações de vários processos ou de um único processo correspondente à realização de um caso de uso.

A notação gráfica do diagrama de visão geral de interação consiste na representação de dois tipos de quadros:

- **Quadros de Interação:** Contêm a representação completa dos diagramas de interação do tipo diagrama de sequência ou diagrama de comunicação.
- **Quadros de Ocorrência de Interação:** Fazem referência a um diagrama de interação especificamente separado por um diagrama de sequência, contudo não apresentam detalhamento.

Na elaboração do diagrama de visão geral de interação, além da representação dos quadros de interação e/ou quadros de ocorrência de interação, unem-se os quadros com os mesmos elementos do diagrama de atividades (nó inicial, nó final, nó de decisão, nó de bifurcação e nó de união).

Diagrama de Tempo

O diagrama de tempo também é um diagrama novo que foi introduzido a partir da UML 2.0. Ele representa, de forma concisa e simples, a mudança pontual nos estados de um objeto, relevantes para contexto da execução de um processo que envolve várias atividades, ou especificamente de um caso de uso, em resposta aos eventos disparados durante uma interação.

A notação gráfica do diagrama de tempo consiste em um único retângulo (quadro) com a indicação do nome em um rótulo, indicado no canto superior à esquerda do quadro.

O diagrama de tempo concentra-se nas condições que mudam dentro e entre linhas de vida ao longo de um eixo de tempo linear. O diagrama descreve o comportamento dos classificadores individuais e as interações dos classificadores, concentrando a atenção no tempo dos eventos que causam mudanças nas condições modeladas das linhas de vida.

Considerando a evolução das versões da UML, os dois diagramas de interação – diagrama de visão geral de interações e o diagrama de tempo – foram introduzidos na UML 2.0 e, na última versão da UML, lançada em Dezembro de 2017, a versão 2.5.1, foi introduzido um diagrama estrutural

chamado de diagrama de perfil, que visa demonstrar a criação de uma extensão da notação da UML, a ser complementada com estereótipos específicos e aplicados a domínios com características particulares.

Segundo Guedes (2018), o diagrama de perfil é um diagrama abstrato que permite adaptar a notação da UML a uma nova plataforma com características, tecnologias ou domínio específicos a partir da criação de perfis que representam a extensão da linguagem para criação de novas metaclasses e estereótipos, permitindo, assim, a modelagem desses novos domínios.

Modelagem Inicial da Atividade de Análise

Estudo de Caso – Locadora de Veículos

Modelagem do sistema ilustrando a documentação dos seus principais diagramas estruturais e comportamentais da Unified Modeling Language (UML).

Atividade Análise e Projeto

Em um processo de desenvolvimento iterativo a atividade de análise precede o da atividade de projeto, entretanto, a modelagem da análise e o projeto podem acontecer simultaneamente.

- **Especificação do Modelo de Casos de Uso:** Na atividade de análise, inicia-se a modelagem com a definição dos casos de uso, a partir dos requisitos funcionais identificados na atividade de requisitos, especificando o modelo de casos de uso.
- **Especificação do Modelo de Classes:** Considerando que o modelo de casos de uso está pronto, a próxima etapa é analisar cada caso de uso e iniciar a identificação das classes de objetos, compreendendo qual classe ou quais classes participam da realização de um caso de uso e como o sistema será estruturado internamente, especificando o modelo de classes geralmente em várias perspectivas de visão.
- **Documentação Descritiva de Cada Caso de Uso:** A partir da primeira versão do modelo de classes é mais fácil complementar a modelagem dos casos de uso com a documentação descritiva de cada caso de uso, pois nesse momento já se identificou os objetos que colaboram e devem participar da execução de cada caso de uso.

Modelo de Casos de Uso

Antes de iniciar a especificação dos casos de uso, é importante listar todos os casos de uso identificados, para assim tomar a decisão de agrupá-los ou não por categoria ou assunto e, com isso, desenhar o diagrama de pacotes e o diagrama de casos de uso correspondentes a cada pacote, compondo o modelo de casos de uso.

- **Diagrama de Pacotes – Modelo de Casos de Uso:** Foram definidos dois pacotes – “mdlDucNegocio” e “mdlDucConsultaRelatorio”, correspondentes ao módulo locação, e o pacote “mdPagamentoDuc”, correspondente ao módulo pagamento que foi integrado ao módulo locação que está sendo especificado. Assim, os pacotes representados com suas dependências correspondem aos diagramas de casos de uso que agruparam os casos de uso por assunto, com o objetivo de organizarem os casos de uso identificados, constituindo um modelo de casos de uso.
- **Diagrama de Casos de Uso – Pacote mdlDucNegocio:** Ilustra os casos de uso correspondentes aos requisitos funcionais e outros casos de uso que foram identificados a

partir da abstração desses requisitos, considerando também o contexto do domínio do sistema.
- **Diagrama de Casos de Uso – Pacote mdlDucConsultaRelatorio:** Concentra os casos de uso definidos para as funcionalidades das consultas e relatórios operacionais do sistema.

Diagramas de Classe

A partir da abstração dos casos de uso inicia-se a identificação das classes de objetos e a elaboração do diagrama de classes, que é considerada a principal técnica de modelagem estrutural da UML, representando a modelagem da parte estática do sistema e simbolizando um conjunto de classes com seus atributos, operações e relacionamentos.

- **Diagrama de Classes (Análise) – mdLocacaoDc:** Respeito as regras básicas de modelagem do diagrama e as regras de negócio do sistema.
- **Verificação de Consistência Entre as Classes e os Casos de Usos Definidos:** A partir da elaboração de uma primeira visão do diagrama de classes, deve-se refiná-lo e incrementá-lo com novos detalhes correspondentes às tecnologias de implementação que serão adotadas, assim especificando o modelo ideal do diagrama de classes da atividade de projeto.
- **Refinamento e Especificação do Modelo Ideal do Diagrama de Classes da Atividade de Projeto:** A partir da elaboração de uma primeira visão do diagrama de classes, deve-se refiná-lo e incrementá-lo com novos detalhes correspondentes às tecnologias de implementação que serão adotadas, assim especificando o modelo ideal do diagrama de classes da atividade de projeto.

Documentação dos Casos de Uso

Não existe um formato específico de documentação para os casos de uso definido pela UML. O formato de documentação de um caso de uso é flexível, permitindo que se documente o caso de uso da forma que se considerar melhor, até mesmo com o uso de pseudocódigo ou de código de uma linguagem de programação. Outra opção é utilizar a prototipação da interface gráfica para facilitar a compreensão da execução do caso de uso.

- **Documentação do Caso de Uso – Manter Empresa:** Respeito as regras básicas de modelagem do diagrama e as regras de negócio do sistema.

A documentação do caso de uso também pode ser feita diretamente na ferramenta case de modelagem, vinculada à representação do caso de uso.

Para apoiar a especificação dos processos de negócio de uma organização e agilizar a identificação dos requisitos de um sistema de software, a modelagem organizacional visa facilitar a compreensão do ambiente organizacional que envolve os relacionamentos entre os níveis organizacionais e funcionais do ambiente e as complexas interações entre a organização e as pessoas, sendo reconhecida, assim, como uma importante atividade pela engenharia de requisitos. Entre os vários métodos para conceber a modelagem organizacional, o método Enterprise Knowledge Development (EKD) facilita a aquisição do conhecimento da estrutura organizacional e estratégica e auxilia na captura dos requisitos organizacionais, possibilitando a compreensão das necessidades do ambiente empresarial por parte de todos os envolvidos na modelagem de processos de negócio e, consequentemente, na especificação dos requisitos de um sistema de informação (Guerrini Et Al., 2014).

Modelagem Complementar da Atividade de Análise

Estudo de Caso – Locadora de Veículos

Modelagem complementar do sistema a partir da especificação dos diagramas de estrutura composta, dos diagramas de atividades, dos diagramas de máquina de estados e dos diagramas de sequência.

Diagrama de Estrutura Composta

O diagrama de estrutura composta é um novo diagrama estrutural da UML 2.0, que visa identificar a arquitetura do conjunto de elementos que interagem entre si durante a execução do sistema, formando uma colaboração entre esses elementos que se comunicam, contudo não especifica o comportamento da colaboração, que é o objetivo dos diagramas comportamentais da UML.

- **Diagrama de Estrutura Composta – Reservar Carro:** O diagrama de estrutura composta corresponde à colaboração denominada “reservar carro”, que abrange o caso de uso “reservar carro” e seus relacionamentos – “imprimir comprovante de reserva” e “enviar e-mail do comprovante da reserva”. Observe que o diagrama representa a colaboração, considerando que durante a execução dos casos de uso que integram a colaboração, cada reserva é instanciada na classe reserva e, para cada reserva, estabeleceu-se um vínculo que apresenta a comunicação entre os objetos que interagem para atingir o objetivo de efetivação de uma reserva.

Diagrama de Máquina de Estados

Diante do contexto e de algumas regras de negócio já definidas na descrição do estudo de caso, as classes de objetos identificadas com estados relevantes no diagrama de classes especificado são “Reserva”, “Carro” e “Pessoa”. Entretanto, para melhor controle e agilidade das consultas e relatórios, também foi definida a classe “AluguelDevolucao”, com estados relevantes.

Lembre-se de que na elaboração do diagrama de máquina de estados é fundamental identificar as regras de negócio aplicadas ao contexto dos objetos com estados relevantes, definindo consistentemente os estados e suas transições de estados, que são os elementos básicos do diagrama.

- **Diagramação de Máquina de Estados – Classe Carro:** Respeito as regras básicas de modelagem do diagrama e as regras de negócio do sistema.
  - **Estados:** Para os objetos da classe “Carro”, os estados definidos são: Disponível, alugado, manutenção interna, manutenção externa e inativo.
  - **Regras:**
    - Todo carro ao ser cadastrado deve assumir, automaticamente, a situação de disponível.
    - Todo carro disponível deve assumir a situação de alugado quando for realizado um aluguel.
    - Todo carro alugado deve assumir a situação de manutenção interna quando foi realizada a devolução do carro.
    - Todo carro em manutenção interna deve assumir a situação de disponível, se confirmada a vistoria técnica da oficina e a lavagem do carro.
    - Todo carro em manutenção interna deve assumir a situação de manutenção externa, se o gerente assim determinar devido a algum problema técnico identificado que não possa ser resolvido pela oficina da locadora.

        - Todo carro disponível deve assumir a situação de manutenção interna, se o gerente assim determinar por algum motivo pontual.
        - Todo carro em manutenção externa deve assumir a situação de manutenção interna para conferência geral do carro, após a finalização do serviço realizado pela oficina externa.
        - Todo carro em manutenção interna deve assumir a situação de inativo, se a gerência assim determinar.
        - Todo carro inativo não deve assumir outro estado.
    - **Diagrama:** O diagrama de máquina de estados correspondente apresenta os estados definidos para os objetos da classe “Carro”, e suas transições de estados são indicadas com as condições de guarda, considerando as regras de negócio apresentadas.

Diagrama de Atividades

Os elementos de um diagrama de atividades podem ser divididos para demonstrar fluxos de controle paralelos, também denominados simultâneos, ou fluxos de controle sequenciais, também chamados de simples. Para facilitar a elaboração do diagrama de atividades ou de outro diagrama comportamental, recomenda-se a descrição do cenário de execução do caso de uso, utilizando um dos formatos de documentação do caso de uso.

- **Diagramação de Atividades – Acessar Conta Cliente:** O diagrama de atividades corresponde ao caso de uso “acessar conta cliente”, no formato de fluxos de controle sequencial.

Diagrama de Sequência

Na modelagem da atividade de análise, recomenda-se utilizar o diagrama de sequência para descrever a realização dos casos de uso, representando os objetos que colaboram entre si a partir da troca de mensagem entre eles.

- **Diagramação de Sequência – Alugar Carro:** O diagrama de atividades corresponde ao caso de uso “acessar conta cliente”, no formato de fluxos de controle sequencial.

A modelagem da perspectiva dinâmica do sistema de software pode ser elaborada com base na modelagem de processos de negócio (Business Process Modeling – BPM) da empresa, que visa criar um modelo de processos por meio da construção de diagramas operacionais sobre seu comportamento. O Business Process Modeling notation (BPMN) é um padrão para modelagem de processo de negócio que mantém um único modelo de diagrama, chamado de Business Process Diagram (BPD – Diagrama de Processo de Negócio), que oferece recursos para a modelagem dos mais variados tipos de processos, desde os mais genéricos aos específicos.

Transição da Atividade de Análise para Projeto

Estudo de Caso – Locadora de Veículos

Refinamento referente aos aspectos estáticos e estruturais da atividade de projeto, tomando-se por base a modelagem da atividade de análise. Persistência de objetos, considerando o mapeamento das classes para tabelas de banco de dados relacional.

Refinamento dos Aspectos Estáticos e Estruturais

Como refinamento dos aspectos estáticos e estruturais das técnicas de modelagem da UML, para a atividade de projeto o foco concentra-se na principal técnica de modelagem estrutural, o diagrama de classes. É recomendado:

- Refinar as classes, definindo as classes de projeto ou novas classes, ou seja, uma classe de análise pode resultar em mais de uma classe de projeto.
- Definir o estereótipo das classes, que são classes de fronteira («boundary»), de controle («control») ou de entidade («entity»).
- Estabelecer o tipo de dados de cada atributo, correspondente à linguagem de programação que será adotada na implementação.
- Detalhar as operações, listando todas as operações identificadas nos diagramas comportamentais e de interação.
- Revisar a visibilidade das classes e operações, definindo o nível de acessibilidade de um atributo ou operação por outros objetos, sendo a visibilidade do tipo privada, pública, protegida ou de pacote.
- Rever os tipos de relacionamentos estabelecidos entre as classes, que são do tipo associação (podendo ser associação do tipo reflexiva, binária, ternária, classe associativa e agregação), generalização, dependência ou realização, além de indicar a navegabilidade de cada associação.
- Definir as classes abstratas, as interfaces, padrões de projeto (design patterns), componentes de softwares reutilizáveis, frameworks e demais detalhes pertinentes às tecnologias de desenvolvimento a serem utilizadas durante a implementação, definindo, assim, a arquitetura de um sistema orientado a objetos.

Modelagem de Classe de Projeto

Na representação do diagrama de classes da atividade de projeto, é importante revisar o relacionamento estabelecido entre as classes de objetos. Os relacionamentos usuais estabelecidos no diagrama de classes de análise são do tipo associação e generalização, no entanto, na versão do diagrama de projeto é comum inserir componentes de softwares, bem como aplicar padrões de projeto (design patterns). Assim, podem surgir os relacionamentos do tipo realização e dependência.

Ainda faz parte da definição das classes de projeto definir o estereótipo das classes. Uma classe indicada com um estereótipo representa uma classificação do elemento:

- **Estereótipo de Fronteira («boundary»):** Identifica uma classe que serve de comunicação entre os atores externos e o sistema. Muitas vezes uma classe de fronteira é associada à própria interface do sistema. A utilização de classes de fronteira é importante quando é preciso definir a existência de uma interface para o sistema, sendo desnecessária em sistemas muito simples cujas interfaces não apresentam nenhuma característica especial.
- **Estereótipo de Controle («control»):** Identifica classes que servem de intermédio entre as classes de fronteira e as outras classes do sistema. Os objetos de controle são responsáveis por interpretar os eventos ocorridos sobre os objetos de fronteira, como os movimento do mouse ou o pressionamento de um botão, e retransmiti-lo para os objetos das classes de entidade que compõe o sistema.
- **Estereótipo de Entidade («entity»):** Classes de entidade também são chamadas de classes do negócio, e são aquelas que representam os conceitos do domínio do sistema, ou seja, a classe contém informações recebidas ou geradas por meio do sistema. É com base nas classes de entidade que se define quais delas geram objetos que devem ser persistentes, no qual o mecanismo de armazenamento geralmente é um sistema de gerenciamento de banco de dados.

- **Diagrama de Classes (Projeto):** Recorte do diagrama de classes de projeto do sistema “Locação de Veículos” referente à classe Reserva, correspondente à visão de classes participantes do caso de uso “Reservar Carro”.

Persistência de Objeto para o Modelo Relacional

Para especificar o mapeamento de classes para tabelas do modelo de dados relacional, é usual adotar técnicas de modelagem de dados e/ou definir o uso de frameworks de mapeamento objeto relacional, como estratégia de armazenamento persistente. Assim, como parte da documentação que envolve o projeto de banco de dados, deve-se apresentar no mínimo a construção do esquema do banco de dados.

Considerando que foi definido o uso de SGBDR como mecanismo de armazenamento dos objetos, é necessário fazer o mapeamento dos valores de atributos de objetos das classes persistentes para as tabelas de banco de dados relacional, com base no modelo de classes:

- **Identificação dos Objetos das Classes:** Primeiramente, deve-se identificar se os objetos das classes são objetos transientes ou objetos persistentes. Normalmente os objetos de entidade são os objetos persistentes, os quais devem ser armazenados em meio físico durante a execução do sistema para serem manipulados. Os objetos transientes existem somente durante uma sessão de uso do sistema e geralmente são os objetos de fronteira e de controle.
- **Análise das Classes e Seus Relacionamentos (Esquema do Banco de Dados Relacional):** Na sequência, são analisadas as classes persistentes e seus relacionamentos e aplicadas as alternativas de mapeamento apresentadas a seguir, considerando a notação indicada para manter uma padronização da representação das tabelas e, assim, constituir o esquema do banco de dados relacional (BDR):
  Nome da Tabela (Coluna 1, Coluna 2, Coluna 3, Coluna 4,… Coluna n)
  - Cada coluna representa um atributo da classe mapeada, no entanto, atenção aos atributos derivados, pois eles não são mapeados para uma coluna.
  - Destaca-se a coluna que representa a chave primária com sublinhado simples e as colunas que representam chaves estrangeiras com sublinhado tracejado.
  - Representa-se em cada tabela derivada de classe, no geral, uma coluna que indica o identificador (ID) para a chave primária. Essa estratégia de notação dos “IDs” define a identidade independente dos objetos, conforme os princípios da orientação a objetos.
- **Mapeamento de Classes para Tabelas:** Segundo Rumbaugh (1997), as principais alternativas de mapeamento de classes para tabelas são:
  - **Mapeamento de Associação Binária:** Para as classes relacionadas com associação binária, com multiplicidade um-para-muitos, mapeia-se cada classe em uma tabela. Para associação binária com multiplicidade um-para-um, pode-ser mapear as classes cada uma em uma tabela ou unir os atributos das duas classes em uma única tabela.
  - **Mapeamento de Classe Associativa:** Para as classes relacionadas com associação de classe associativa, mapeia-se cada classe em uma tabela. Para classes relacionadas com multiplicidade muitos-para-muitos, cria-se a terceira tabela com apenas a chave primária composta.
  - **Mapeamento de Agregação:** Para classes relacionadas com associação do tipo agregação, mapeia-se a classe “Todo” e “Parte” para tabelas individuais. O identificador da classe “Todo” é indicado como chave estrangeira na tabela que representa a classe “Parte”.
  - **Mapeamento de Composição:** Para classes relacionadas com associação do tipo composição (tipo especial de agregação), mapeia-se a classe “Todo” e “Parte” para tabelas individuais. O identificador da classe “Todo” torna-se parte da chave primária na tabela que representa a classe “Parte”.

- **Mapeamento de Generalização:** Existem três abordagens para o mapeamento do relacionamento do tipo generalização em tabelas. A abordagem normal define que a superclasse e as subclasses são mapeadas cada uma em uma tabela com a utilização de um “ID” compartilhado e a criação de um atributo tipo na tabela que representa a superclasse, para identificar os tipos de objetos representados pelos objetos das subclasses. A segunda e terceira abordagens são consideradas alternativas de mapeamento de generalização. A segunda abordagem define a eliminação da tabela correspondente a cada subclasse, reproduzindo todos os atributos da superclasse em cada tabela da superclasse. A terceira abordagem define a criação de uma única tabela correspondente à superclasse, unindo todos os atributos das subclasses ao nível da superclasse.

Seguindo as tendências do desenvolvimento ágil de software que enfatiza a documentação de análise e projeto com artefatos mínimos, o desenvolvimento simplificado com entregas incrementais antecipadas, equipes de projeto pequenas e altamente motivadas, além de priorizar como princípios de desenvolvimento a comunicação ativa entre desenvolvedores e clientes e a satisfação do cliente, entre os atuais modelos de processo de desenvolvimento ágil, o processo unificado ágil (AUP – Agile Unified Process) é uma evolução do processo unificado que contempla as fases clássicas de concepção, elaboração, construção e transição, entretanto, em cada atividade, a equipe itera para alcançar a agilidade e entregar incrementos de software significativos para os usuários o mais rápido possível. De acordo com Pressman e Maxim (2016), “[…] embora o AUP tenha conexões histórias e técnicas com a linguagem de modelagem unificada, é importante notar que a modelagem UML pode ser usada com qualquer modelo de processo ágil.