NÃO PODE FALTAR

TRANSIÇÃO DA ATIVIDADE DE ANÁLISE PARA PROJETO

Iolanda Cláudia Sanches Catarino

REFINAMENTO DOS ASPECTOS ESTÁTICOS E ESTRUTURAIS DO MODELO DE CLASSES

Complementação das classes com os tipos de dados dos atributos com as demais operações identificadas na especificação dos diagramas comportamentais e de interação, além da revisão dos relacionamentos das classes, apresentando, assim, uma nova versão do Modelo de Classes.

Fonte: Shutterstock.

Deseja ouvir este material?

Áudio disponível no material digital.

PRATICAR PARA APRENDER

Caro aluno, seja bem-vindo à seção sobre a transição da atividade de análise para o projeto!

Nesta seção, apresentamos a transição da análise ao projeto no desenvolvimento de sistemas orientado a objetos, considerando a adoção da Linguagem de Modelagem Unificada (UML – *Unified Modeling Language*) e considerando os princípios do modelo de processo de software Processo Unificado (PU – *Unified Process*) que surgiu para apoiar a UML, enfatizando o desenvolvimento dirigido a casos de uso, de forma iterativa e incremental, e fornecendo uma maneira sistemática e evolutiva de modelar os sistemas sob diferentes aspectos de análise e detalhamento.

Lembre-se de que a UML não faz grande distinção entre a modelagem das atividades de análise e projeto de um processo de desenvolvimento de software, sendo que a atividade de projeto é uma extensão refinada dos diagramas elaborados na análise com detalhes aplicados à implementação, e a análise e o projeto podem estar acontecendo simultaneamente. Essa definição de trabalhar com a modelagem simultânea ou até mesmo de apresentar uma única modelagem como sendo das atividades de análise e projeto com diferentes perspectivas de visão, é uma questão definida na metodologia de desenvolvimento do sistema, conforme cada equipe de desenvolvedores ou empresa prefere adotar.

Nas unidades anteriores foram abordadas todas as características das técnicas de modelagem da UML. Além disso, você compreendeu que a UML privilegia a descrição de um sistema em três perspectivas principais de visões, que são a estrutural, que enfatiza a visão estática do sistema; a funcional, que representa as funcionalidades do sistema; e a dinâmica, que representa a visão do comportamento do sistema em tempo de execução. Assim, de todas as técnicas de modelagem propostas pela UML – estruturais, comportamentais e de interação –, o importante é saber definir qual ou quais são as técnicas ideais a serem adotadas na modelagem de análise e de projeto, conforme o tipo e domínio do sistema.

Nesta seção você vai compreender os pontos essenciais do detalhamento dos aspectos estruturais do Diagrama de Classes, o qual pode ser apresentado como o Diagrama de Classes da atividade de projeto, como define-se o tipo de dados de cada atributo, listando todas as operações identificadas na modelagem dos diagramas comportamentais e de interação e fazendo a revisão dos relacionamentos.

Na modelagem de projeto, recomenda-se também apresentar uma nova versão dos diagramas estruturais e comportamentais que seja possível complementá-los com detalhes de padrões de projeto – *design patterns*, componentes de softwares, *frameworks* e das tecnologias que serão adotadas para implementação do sistema, assim definindo o projeto da arquitetura do sistema.

Outro aspecto importante a considerar na modelagem de um sistema orientado a objetos, na atividade de projeto, é a forma pela qual os objetos especificados na representação de classes serão persistidos, e uma das principais estratégias adotadas é o mapeamento de classes para um sistema de gerência de banco de dados relacional. O projeto de algoritmos para implementar o sistema e o projeto

de interface gráfica dos casos de uso que implicam a necessidade de interfaces, para darem o suporte de interação com os atores e usuários do sistema, também são atividades da modelagem de projeto.

Agora considere que você é o projetista de banco de dados, integrante da equipe de desenvolvedores responsável pelo sistema “Locação de Veículos”, e como parte da documentação referente a modelagem de dados você precisa apresentar o mapeamento das classes para tabelas do Banco de Dados Relacional (BDR) correspondente ao Modelo de Classes, composto de:

- Diagrama de Classes – md_Locacao_dc, ilustrado na Figura 4.31.
- Diagrama de Classes – md_Pagamento_dc, ilustrado na Figura 4.8 da seção 4.1.

Para especificação das tabelas, adote a notação “Nome da Tabela (coluna 1, coluna 2, coluna 3, coluna 4,... coluna n)”, representando, assim, o esquema completo do banco de dados.

Bom trabalho e ótimo estudo!

CONCEITO-CHAVE

Dependendo do domínio e da complexidade dos sistemas de software, é importante fornecer várias visões da modelagem orientada a objetos sob diferentes aspectos de análise e detalhamento, consistindo em uma única modelagem de análise e projeto ou apresentando modelagens distintas das atividades de análise e projeto. Conforme o modelo de processo de desenvolvimento de software Processo Unificado, no qual preconiza-se a iteração, a análise e projeto integram-se como uma atividade conjunta executada ao longo das fases do processo – Concepção, Elaboração, Construção e Transição. Contudo, adotando a UML para modelagem de sistemas orientados a objetos, não há uma regra de qual ou quais técnicas de modelagem estruturais ou comportamentais devem ser adotadas e em que momento devem ser aplicadas. Essa decisão e definição é de responsabilidade do analista de sistemas, em conformidade com a metodologia da empresa de desenvolvimento.

O importante é ter claro que a UML oferece uma forma sistemática e evolutiva de modelar os sistemas orientados a objetos em várias perspectivas de visão estática e dinâmica, com base nas suas técnicas de modelagem estruturais, comportamentais e de interação.

Na atividade de análise evolui-se a modelagem especificada na atividade de análise de requisitos, concentrando-se na solução do problema, a partir da abstração, identificação, definição e documentação do que o sistema deve fazer, considerando a visão lógica do negócio, independentemente das tecnologias que serão adotadas para implementação do sistema. Posteriormente, a análise especificada vai se transformando em projeto, com novos detalhes e refinamentos que definem os aspectos físicos do sistema e aplicadas as tecnologias que serão adotadas na implementação do sistema.

Segundo Bezerra (2014), no desenvolvimento de sistemas orientados a objetos, as mesmas técnicas de modelagem são utilizadas durante a análise e o projeto, caracterizando uma uniformidade na modelagem do sistema durante o desenvolvimento. No entanto, essa uniformidade de representação tem a desvantagem de tornar bem menos nítida a separação entre o que é especificado na análise e o que é feito no projeto. Dessa forma, pode-se dizer que a modelagem da análise vai se transformando na modelagem de projeto à medida que o desenvolvimento do sistema evolui.

MODELAGEM DE PROJETO

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. Assim, é recomendado:

1. 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.
2. Definir o estereótipo das classes, que são classes de fronteira (*<<boundary>>*), de controle (*<<control>>*) ou de entidade (*<<entity>>*).
3. Estabelecer o tipo de dados de cada atributo, correspondente à linguagem de programação que será adotada na implementação.
4. Detalhar as operações, listando todas as operações identificadas nos diagramas comportamentais e de interação.
5. 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.
6. 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.
7. 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.

> **LEMBRE-SE**
>
> Na representação dos atributos e operações de uma classe, indica-se também, à esquerda do nome desses atributos, o símbolo da visibilidade que determina o nível de acessibilidade de um atributo ou operação por outros objetos, do tipo: **privada** – representada por um símbolo de menos (-) e indica que somente os objetos da própria classe poderão enxergá-la; **pública** – representada por um símbolo de mais (+) e indica que qualquer objeto pode utilizar o objeto acessado; **protegida** – representada pelo símbolo de sustenido (#) e indica que além dos objetos da própria classe, os

objetos das subclasses também podem ter acesso ao objeto da superclasse; e de **pacote** – representada pelo símbolo de til (~) e indica que o atributo ou operação é visível por qualquer objeto do mesmo pacote.

MODELAGEM DE CLASSE DE PROJETO

As boas práticas da engenharia de software enfatizam a redução dos esforços e custos da produção, com a reutilização de software que pode ser obtida com a utilização de recursos como os componentes de softwares e os usuais *frameworks* para o desenvolvimento web e para o desenvolvimento de aplicações móveis, aplicados às diferentes tecnologias de desenvolvimento de software. Assim, uma vez adotados esses recursos que agilizam o desenvolvimento de software, eles devem ser integrados à modelagem de projeto, estabelecendo a arquitetura do software.

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 *design patterns*. Assim, podem surgir os relacionamentos do tipo realização e dependência. Vamos, então, revisar os conceitos que se referem a classes:

- **Realização:** relacionamento que modela a conexão existente entre uma interface e uma classe ou componente, no qual um dos elementos especifica um contrato de uso com o outro elemento. A representação gráfica do relacionamento de realização é indicada por uma linha tracejada com uma grande seta vazia, apontando para o elemento que especifica o contrato.
- **Dependência:** relacionamento de utilização entre classes, indicando que uma alteração na especificação de um elemento pode afetar outro elemento que a utilize. A representação gráfica do relacionamento de dependência é indicada por uma linha tracejada, apontando o item de que o outro depende.

ASSIMILE

Um **padrão de projeto** (*design pattern*) descreve uma estrutura lógica comum para criar um projeto orientado a objetos reutilizável, baseado em soluções práticas com mecanismos de cooperação entre componentes que são definidos para encontrar soluções para os problemas de projeto em um contexto específico. Um ***framework*** consiste em um conjunto de classes abstratas relacionadas, e por um modelo de extensibilidade das suas classes para as aplicações a serem construídas, proporcionando funcionalidades pré-fabricadas para agilizar o desenvolvimento do

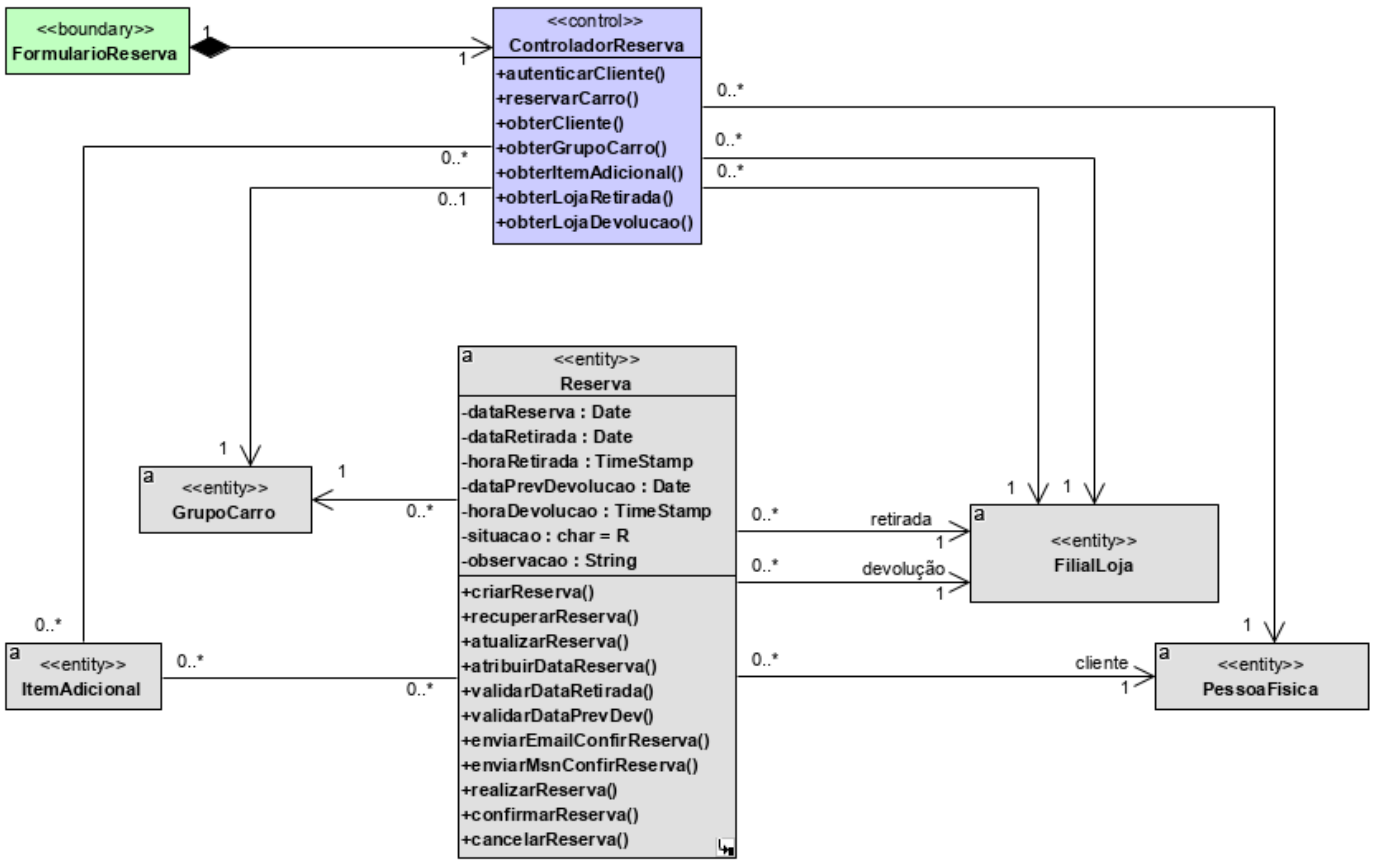
> software. Um **componente de software** é uma parte física e substituível de um software, aplicado a um domínio, ao qual se adapta e fornece a realização de serviços através de uma interface específica.

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. Alguns tipos de estereótipos de classe adotados no Diagrama de Classes de projeto são:

- **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.
- **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 movimentos do mouse ou o pressionamento de um botão, e retransmiti-lo para os objetos das classes de entidade que compõem o sistema.
- **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.

A Figura 4.30 ilustra um 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”, apresentando uma parte do refinamento da classe com os aspectos sugeridos anteriormente. Observe que, por simplificação, apenas o refinamento correspondente a uma classe foi exemplificado. Em uma situação real, todas as classes de análise devem ser consideradas.

Figura 4.30 | Recorte do Diagrama de Classes (Projeto) – md_Locacao_dc

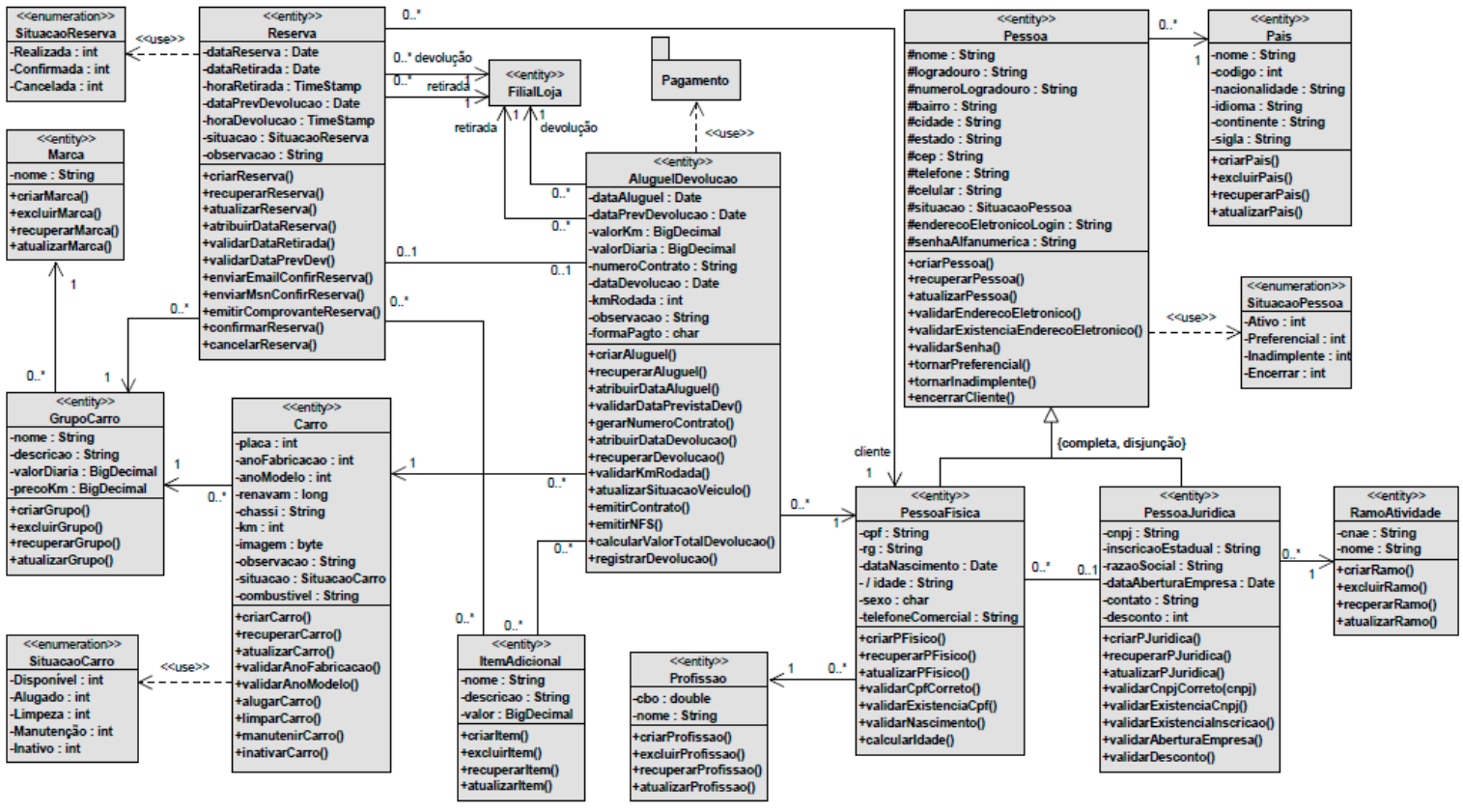


Fonte: elaborada pela autora.

O Diagrama de Classes da atividade de projeto complementa os elementos do diagrama, permitindo enriquecer os detalhes físicos desse diagrama, obtendo um modelo suficientemente completo e aproximando-o com o que será implementado.

A Figura 4.31 ilustra uma visão do Diagrama de Classes de projeto do pacote “md_Locacao_dc”, refinado apenas com a definição do tipo de dados dos atributos correspondentes à linguagem de programação Java; a indicação do estereótipo das classes do tipo entidade (<<entity>>); a representação das classes definidas com o estereótipo enumeração (<<enumeration>>) correspondentes aos valores do atributo situação das classes “Reserva”, “Carro” e “Pessoa”. Foi estabelecido o relacionamento do tipo dependência entre as classes enumeradas e as classes que as referenciam, contudo, não é obrigatório estabelecer esses relacionamentos de dependência entre essas classes e a indicação da navegabilidade nas associações binárias para definir a referência entre os objetos, sendo que a não indicação da navegabilidade representa a referência bidirecional entre os objetos associados.

Figura 4.31 | Diagrama de Classes (Projeto) – md_Locacao_dc



Fonte: captura de tela do software Visual Paradigm Community Edition elaborada pela autora.

No diagrama também foi indicada a restrição – também chamado de classificador do relacionamento de generalização. Os classificadores da generalização são utilizados para definir melhor a semântica das classes especializadas derivadas da classe genérica. Os classificadores da generalização são escritos entre chaves e próximos à seta de generalização. Os classificadores predefinidos para classes especializadas são do tipo:

- **Completo:** indica que qualquer instância da superclasse (abstrata) será uma instância de uma das subclasses especializadas.
- **Incompleto:** indica que pode existir um conjunto de objetos de novas subclasses não representadas, e uma instância de seu tipo pode existir apenas da própria superclasse (concreta).
- **Disjunção:** indica que qualquer instância da superclasse pode ser uma instância de apenas uma das subclasses, ou seja, exclusiva de uma subclasse.
- **Sobreposição:** indica que uma instância da superclasse pode pertencer a mais de uma subclasse.

Por padrão, da UML 2.0 até a versão 2.4.1, o classificador da generalização era do tipo “{incompleta, disjunção}”. Na UML 2.5, o padrão foi alterado para “{incompleta, sobreposição}”, assim, uma vez que o contexto do domínio do sistema for diferente desse padrão, então deve-se representar o classificador da generalização; senão, suprime-o. A especificação UML não determina como essa equivalência semântica será implementada e como a integridade entre os objetos será mantida a partir da indicação do classificador da generalização.

REFLITA

O relacionamento do tipo generalização estabelecido entre classes do Diagrama de Classes indica o princípio de herança entre classes genéricas e classes especializadas, o qual representa a propriedade pela qual uma classe pode herdar atributos e operações de uma classe que generaliza as características e comportamentos comuns de um grupo de objetos. Assim, no Diagrama de Classes refinado para a atividade de projeto é obrigatório estabelecer o relacionamento de generalização entre classes?

REFINAMENTO DOS ASPECTOS COMPORTAMENTAIS

A modelagem de projeto ainda pode ser complementada com demais os diagramas comportamentais e de interação da UML que não foram especificados na análise, ou os diagramas de análise podem ser refinados com detalhes que serão aplicados à implementação.

Segundo Bezerra (2014, p. 177), "[...] embora o estudo dos aspectos dinâmicos do sistema já comece na etapa de análise, é na fase de projeto que esse estudo se concretiza e onde se realiza o detalhamento das colaborações nas quais cada classe participa".

Na modelagem comportamental de projeto com a UML, é recomendável evoluir o Modelo de Casos de Uso detalhando todos os relacionamentos de inclusão (<<Include>>) e extensão (<<Extend>>) entre os casos de uso, bem como consistir as operações listadas nas classes de objetos com as mensagens que executam operações indicadas nos Diagramas de Sequência e com as ações definidas nos Diagramas de Atividades. É importante também revisar as ações de estados representadas pelas cláusulas predefinidas "*entry*, *exit* e *do*" em cada estado representado nos Diagramas de Máquina de Estados, sendo que cada ação de estado deve indicar uma operação nas respectivas classes de objetos. Assim, a modelagem comportamental deve começar na atividade de análise para representar os aspectos dinâmicos do sistema e posteriormente ser detalhada como modelagem de projeto. Além disso, a modelagem de projeto inclui a definição do projeto de algoritmos para posterior implementação das funcionalidades do sistema, compreendendo os métodos de codificação de algoritmos em consonância com as soluções desejadas, sendo que podemos utilizar o Diagrama de Atividades da UML para especificar os algoritmos. E ainda, pode-se elaborar o projeto das interfaces gráficas (por exemplo, formulários/telas

e relatórios) correspondentes aos casos de uso que implicam uma interface amigável com alta usabilidade e facilidade de operação, seguindo os princípios básicos da Interação Humano-Computador (IHC).

De uma forma geral, Bezerra (2014) sintetiza as seguintes características da evolução da documentação de análise para projeto:

- Refinamento dos aspectos estáticos e estruturais da modelagem do sistema.
- Detalhamento dos aspectos dinâmicos da modelagem do sistema.
- Detalhamento da arquitetura do sistema, tomando-se por base a decomposição lógica e física do sistema.
- Definição dos mecanismos de armazenamento dos dados manipulados pelo sistema.
- Definição dos algoritmos a serem utilizados na implementação do sistema.
- Elaboração do projeto da interface gráfica das funcionalidades do sistema.

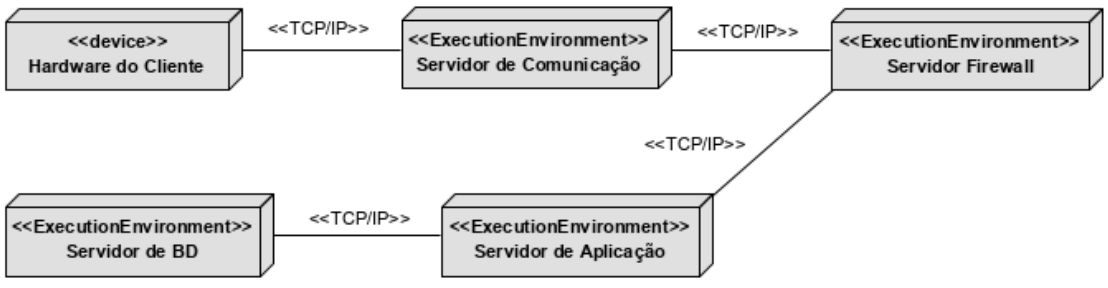
> **REFLITA**
>
> A definição da multiplicidade em uma associação estabelecida entre classes depende de pressupostos e das regras de negócio referentes ao domínio do sistema. Assim, a multiplicidade determina o número mínimo e máximo de objetos envolvidos em uma associação. Dessa forma, no Diagrama de Classes refinado para a atividade de projeto devem ser revisadas também as multiplicidades já definidas na atividade de análise.

Por fim, toda documentação da atividade de projeto do sistema deve ser estabelecida de acordo com as definições da metodologia da empresa de desenvolvimento, atendendo às características e particularidades do domínio do sistema de software.

> **EXEMPLIFICANDO**
>
> Como parte da documentação de projeto, pode-se utilizar a técnica de modelagem estrutural da UML, o Diagrama de Implantação, para ilustrar a organização da arquitetura física do sistema, a partir da representação de nós e suas ligações físicas. Um nó pode representar um item de hardware do sistema ou um outro dispositivo integrado ao sistema e os ambientes de execução que integram a arquitetura física do sistema. A Figura 4.32 ilustra um exemplo do Diagrama de Implantação, demonstrando a associação

entre o nó do tipo dispositivo do “Hardware do Cliente” que representa o acesso do cliente à web para fazer seu cadastro ou uma reserva, por exemplo, com os nós do tipo ambiente de execução “Servidor de Comunicação”, “Servidor Firewall”, “Servidor de Aplicação” e o “Servidor de BD”.

Figura 4.32 | Diagrama de Implantação – Sistema “Locação de Veículos”



Fonte: elaborada pela autora.

Ver anotações 0

PERSISTÊNCIA DE OBJETOS PARA O MODELO RELACIONAL

Os princípios básicos do paradigma orientado a objetos e do modelo relacional são distintos, considerando que as tecnologias de orientação a objetos se baseiam no princípio do encapsulamento, em que os objetos são abstrações de um comportamento. No modelo relacional, os elementos correspondem a dados no formato tabular que utilizam um Sistema Gerenciador de Banco de Dados Relacional (SGBDR). Dessa forma, outro aspecto importante na modelagem de projeto é relativo ao mecanismo de armazenamento persistente de dados correspondentes ao mapeamento de classes de objetos 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.

A maioria das ferramentas CASE de modelagem orientada a objetos fornece a funcionalidade de mapeamento automático para geração de um esquema relacional, a partir do modelo de classes e da definição do SGBDR a ser utilizado. No entanto, é importante que você tenha conhecimento dos procedimentos existentes a serem adotados para a conferência ou elaboração do mapeamento. Dessa forma, vamos conhecer algumas regras fundamentais que refletem em termos práticos para fazer o mapeamento de objetos para o modelo relacional.

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.

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):

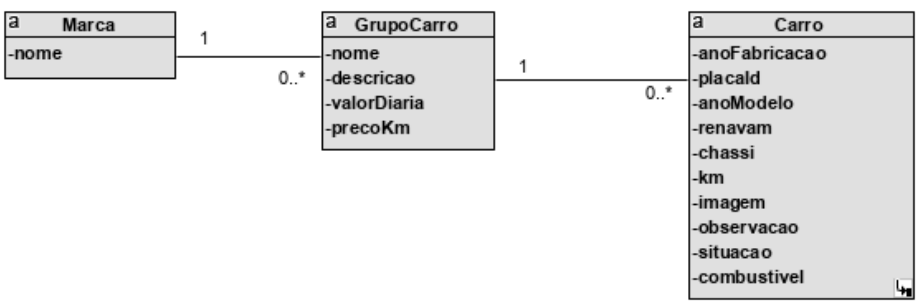
Nome da Tabela (coluna 1, coluna 2, coluna 3, coluna 4,... coluna n)

- 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.

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, conforme exemplo ilustrado na Figura 4.33.

Figura 4.33 | Recorte do Diagrama de Classes com Associação Binária (um-para-muitos)



Fonte: elaborada pela autora.

Mapeamento:

Marca (marcaId, nome).

GrupoCarro (grupoCarroId, nome, descricao, valorDiaria, precoKm, marcaId).

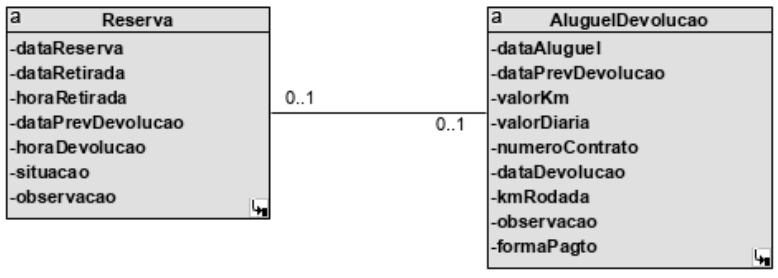
Carro (carroId, anoFabricacao, placa, anoModelo, renavam, chassi, km, imagem, observacao, situacao, combustivel, grupoCarroId).

Para associação binária com multiplicidade um-para-um, pode-se mapear as classes cada uma em uma tabela ou unir os atributos das duas classes em uma única tabela. Essa decisão depende das preferências do projetista de banco de

dados em termos da extensibilidade, do número de tabelas e de desempenho.

A Figura 4.34 ilustra o mapeamento sendo mantido em tabelas separadas. No exemplo ilustrado, não foram representadas as outras classes que se associam com ambas as classes, que seriam outras chaves estrangeiras. Por isso, no final de cada tabela correspondente às classes, foi incluída a sigla "demais*FK*". Essa representação também foi adotada em outros exemplos a seguir.

Figura 4.34 | Recorte do Diagrama de Classes com Associação Binária (um-para-um)



Fonte: elaborada pela autora.

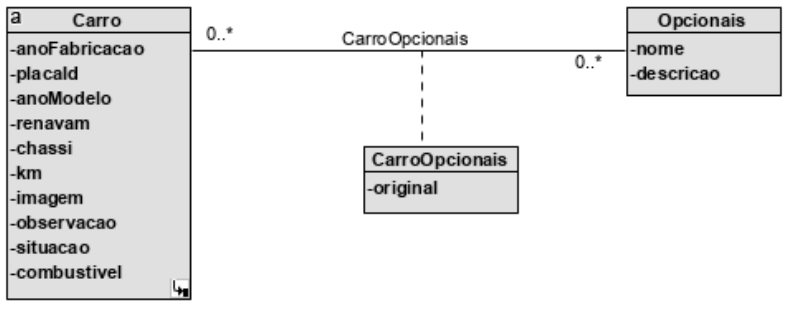
Mapeamento:

Reserva (reservaId, dataReserva, dataRetirada, horaRetirada, dataPrevDevolucao, situacao, observacao, demais*FK).*

AluguelDevolucao(aluguelDevolucaoId, dataAluguel, dataPrevDevolucao, valorKm, valorDiaria, numeroContrato, dataDevolucao, kmRodada, observacao, formaPagto, reservaId, demais*FK).*

Mapeamento de classe associativa: para as classes relacionadas com associação de classe associativa, mapeia-se cada classe em uma tabela, conforme exemplo ilustrado na Figura 4.35.

Figura 4.35| Recorte do Diagrama de Classes com Classe Associativa



Fonte: elaborada pela autora.

Mapeamento:

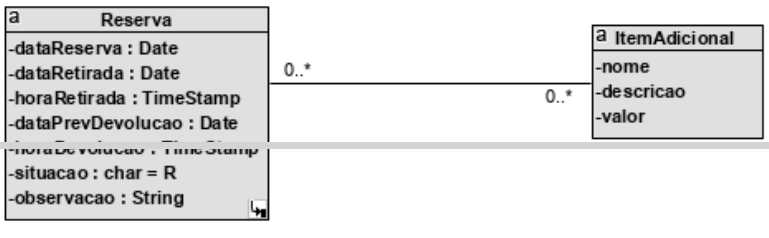
Carro (carroId, anoFabricacao, placa, anoModelo, renavam, chassi, km, imagem, observacao, situacao, combustivel, grupoCarroId).

Opcionais (OpcionaisId, nome, descricao).

CarroOpcionais (carroId, OpcionaisId, original).

Para classes relacionadas com multiplicidade muitos-para-muitos, cria-se a terceira tabela com apenas a chave primária composta, conforme mostra a Figura 4.36.

Figura 4.36 | Recorte do Diagrama de Classes com Associação Binária (muitos-para-muitos)



Fonte: elaborada pela autora.

Mapeamento:

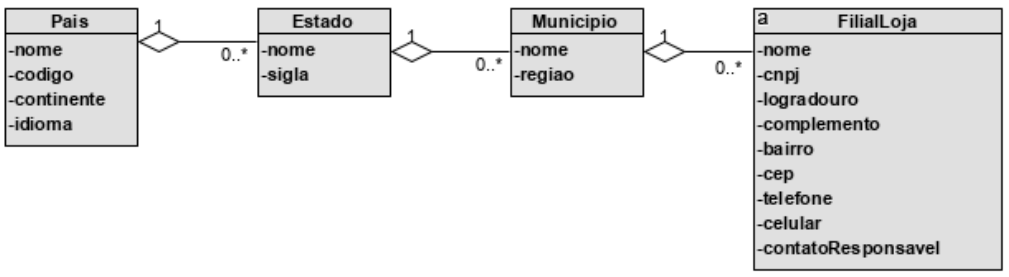
Reserva (reservaId, dataReserva, dataRetirada, horaRetirada, dataPrevDevolucao, situacao, observacao, demais*FK).*

ItemAdicional (itemAdicionalId, nome, descricao, valor).

ReservaItemAdicional (reservaId, itemAdicionalId).

- **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”, conforme é mostrado na Figura 4.37, ou seja, mesma forma de mapear classes com associação binária.

Figura 4.37 | Recorte do Diagrama de Classes com Agregação



Fonte: elaborada pela autora.

Mapeamento:

Pais (paisId, nome, código, continente, idioma).

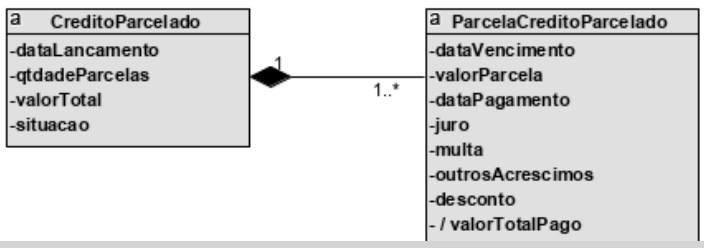
Estado (estadoId, nome, sigla, paisId).

Municipio (municipioId, nome, regiao, estadoId).

FilialLoja (filialLojaId, nome, cnpj, logradouro, complemento, bairro, cep, telefone, celular, contatoResponsavel, municipioId).

- **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”, conforme o exemplo ilustrado na Figura 4.38.

Figura 4.38 | Recorte do Diagrama de Classes com Composição



Fonte: elaborada pela autora.

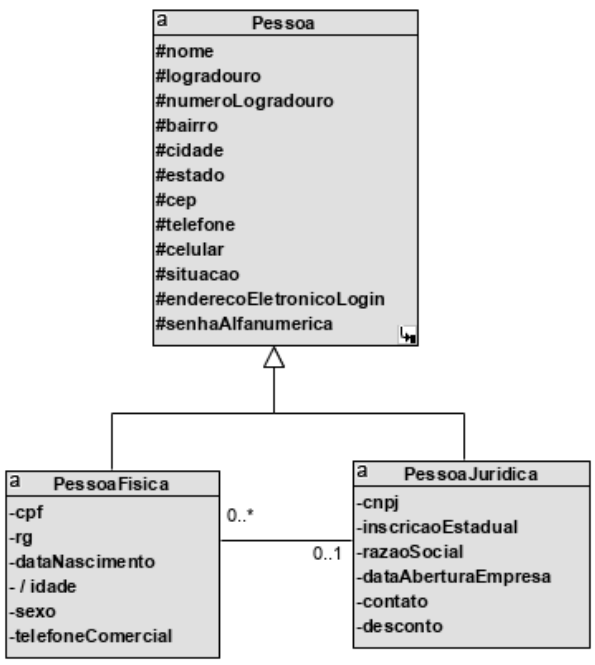
Mapeamento:

CreditoParcelado (creditoParceladoId, dataLancamento, qtdadeParcelas, valorTotal, situacao, demais*FK).*

ParcelaCreditoParcelado (creditoParceladoId, parcelaCreditoParceladoId, dataVencimento, valorParcela, dataPagamento, juro, multa, outrosAcrescimos, desconto).

- **Mapeamento de generalização:** existem três abordagens para o mapeamento do relacionamento do tipo generalização em tabelas. A abordagem normalmente 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, conforme o recorte ilustrado na Figura 4.39.

Figura 4.39 | Recorte do Diagrama de Classes com Generalização



Fonte: elaborada pela autora.

Mapeamento – Alternativa 1:

Pessoa (pessoaId, nome, logradouro, numeroLogradouro, bairro, cidade, estado, cep, telefone, celular, situacao, enderecoEletronicoLogin, senhaAlfanumerica, tipoPessoa [PF/PJ], demais*FK).*

PessoaFisica (pessoaId, cpf, dataNascimento, sexo, telefoneComercial, pessoaIdJ, demais*FK).*

PessoaJuridica (pessoaId, cnpj, inscricaoEstadual, razaoSocial, dataAberturaEmpresa, contato, desconto, demais*FK)*

A segunda e a terceira abordagens são consideradas alternativas de mapeamento de generalização. Segundo Rumbaugh (1997, p. 506), " [...] elas são motivadas pelo desejo de eliminar a navegação de superclasse para subclasse, melhorando o desempenho". A segunda abordagem define a eliminação da tabela correspondente à superclasse e mapeia-se uma tabela correspondente a cada subclasse, reproduzindo todos os atributos da superclasse em cada tabela da superclasse.

Mapeamento – Alternativa 2:

PessoaFisica (pessoaIdF, nome, logradouro, numeroLogradouro, bairro, cidade, estado, cep, telefone, celular, situacao, enderecoEletronicoLogin, senhaAlfanumerica, cpf, dataNascimento, sexo, telefoneComercial, pessoaIdJ, demais*FK).*

PessoaJuridica (pessoaIdJ, nome, logradouro, numeroLogradouro, bairro, cidade, estado, cep, telefone, celular, situação, enderecoEletronicoLogin, senhaAlfanumerica, cnpj, inscricaoEstadual, razaoSocial, dataAberturaEmpresa, contato, desconto, demais*FK).*

A terceira abordagem define a criação de uma única tabela correspondente à superclasse, unindo todos os atributos das subclasses ao nível da superclasse.

Mapeamento – Alternativa 3:

Pessoa (pessoaId, nome, logradouro, numeroLogradouro, bairro, cidade, estado, cep, telefone, celular, situacao, enderecoEletronicoLogin, senhaAlfanumerica, tipoPessoa [PF/PJ], cpf, dataNascimento, sexo, telefoneComercial, pessoaIdJ, cnpj, inscricaoEstadual, razaoSocial, dataAberturaEmpresa, contato, desconto, demais*FK).*

Para garantir a consistência da modelagem de análise e projeto de um software, bem como a evolução da modelagem, é importante utilizar todos os recursos das ferramentas CASE de modelagem e o recurso de gerenciamento de versões ou visões dos modelos especificados, para, assim, facilitar a leitura e interpretação do conjunto de diagramas estáticos e dinâmicos que compõem a documentação de um sistema de software.

REFERÊNCIAS

BEZERRA, E. **Princípios de análise e projeto de sistemas com UML**. 3. ed. Rio de Janeiro: Elsevier, 2014.

OLIVEIRA, M. M. A. *et al.* Um Estudo comparativo entre banco de dados orientado a objetos, banco de dados relacionais e framework para mapeamento objeto/relacional, no contexto de uma aplicação web. **HOLOS**, [*s. l.*], v. 31, n. 1, p. 182–198, 2015. DOI 10.15628/holos.2015.1153. Disponível em: Biblioteca Virtual – EBSCO HOST. https://bit.ly/3bBE3BI. Acesso em: 17 jun. 2020.

PRESSMAN, R.; MAXIM, B. **Engenharia de software**. 8. ed. Porto Alegre: AMGH, 2016

RODRIGUES DE OLIVEIRA, L. *et al.* Desenvolvimento e Avaliação de um Perfil UML para Modelagem de Jogos Educacionais Digitais. **Revista Brasileira de Informática na Educação**, [*s. l.*], v. 26, n. 2, p. 124–143, 2018. DOI 10.5753/RBIE.2018.26.02.124. Disponível em: Biblioteca Virtual – EBSCO HOST. https://bit.ly/3bBE3BI. Acesso em: 17 jun. 2020.

RUMBAUGH, J. *et al.* **Modelagem e projetos baseados em objetos**. Rio de Janeiro: Campus, 1997.