Engenharia de Software

Unidade 05

Prof. Daniel Caixeta

Conteúdo programático

01 Modelagem de sistemas de *software*

1.1. Modelagem de sistemas de software: Conceitos iniciais.

1.1.1 Uma análise da modelagem de sistemas.

1.2. UML.

1.2.1. As visões do sistema.

1.3. Os diagramas UML.

1.4. Os mecanismos gerais.

1.4.1. Estereótipos.

1.4.2. Notas explicativas, Pacotes e Etiquetas valoradas.

1.5. Modelagem de Caso de Uso (MCU).

1.6. Modelagem de Classes de Análises (MCA).

1.6.1. As classes.

1.6.2. As associações entre as classes.

1.7. Modelagem de um sistema acadêmico (Exemplo).

1.7.1. Um estudo de caso [...]

1.7.2. Análises das Regras de Negócios.

1.7.3. Documentação das responsabilidades.

1.7.4. Glossário de conceitos.

BIBLIOGRAFIA

1.1. Modelagem de sistemas de *software*.

Conceitos iniciais [...]

1.1. Modelagem de sistemas de software: Conceitos iniciais

- Uma característica de sistemas de *software* é a complexidade de seu desenvolvimento, que aumenta à medida que o tamanho do sistema cresce. (BEZERRA, 2015).
- Para se ter uma ideia da complexidade envolvida na construção de alguns sistemas, pense no tempo e nos recursos materiais necessários para se construir uma casa de cachorro. Para essa tarefa, provavelmente você precisará de:
  - ✓ Algumas ripas de madeira;
  - ✓ Alguns pregos;
  - ✓ Uma caixa de ferramentas; e
  - ✓ Certa dose de amor pelo seu pet.

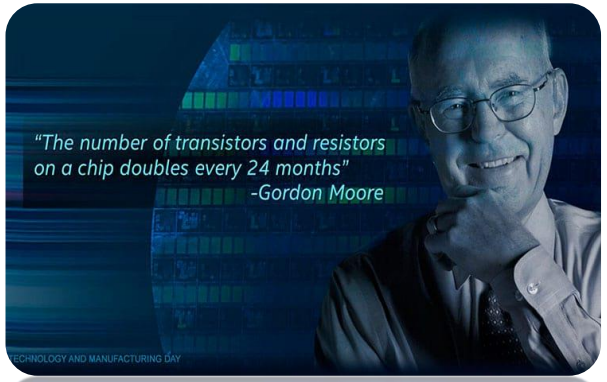
- Depois de alguns dias, a casa estaria pronta.
- Então, o que dizer da construção de uma casa para sua família?

- De acordo com Bezerra (2015), a construção de sistemas de *software* assim como a sua casa possuem graus de complexidades similares.
- Isso nos leva ao conceito de modelo, que de uma perspectiva mais ampla, pode ser visto como uma representação idealizada de um sistema a ser construído. *E.g*., maquetes de edifícios e de aviões e plantas de circuitos eletrônicos, são apenas alguns modelos representativos. (*ibidem*, grifo meu).
- São várias as razões para se utilizar modelos na construção de sistemas. Sendo algumas:
    1. Gerenciamento da complexidade.
    2. Comunicação entre as pessoas envolvidas.
    3. Redução dos custos no desenvolvimento.
    4. Previsão do comportamento futuro do sistema. Etc., ...

- Algumas questões importantes ainda segundo Bezerra (2015):
  - Esses desenhos são normalmente denominados *diagramas*. Portanto, um diagrama é uma apresentação de uma coleção de *elementos gráficos* que possuem significados predefinido.
  - Graças aos desenhos gráficos que modelam o sistema, os desenvolvedores têm uma representação concisa do sistema.
  - Embora diagramas consigam expressar diversas informações de forma gráfica, em diversos momentos há a necessidade de adicionar informações textuais, cujo objetivo é explicar e/ou definir certas partes desse diagrama.
  - Os diagrama e as informações textuais compõem a documentação do modelo.

1.1.1 Uma análise da modelagem de sistemas

- A Lei de Moore é bastante conhecida na área da computação. Embora seja conhecida como lei, foi apenas uma declaração feita em 1965 pelo engenheiro Gordon Moore, cofundador da Intel™. (BEZERRA, 2015).
- Esta “lei” estabelece que “a densidade de um transistor dobra em um período entre 18 e 24 meses”. [...].
- Portanto, a Lei de Moore implica em uma taxa de crescimento exponencial na capacidade de processamento dos computadores[1].



1. Desde que foi declarada, a Lei de Moore vem se verificando com uma precisão impressionante. No entanto, alguns especialistas, incluindo o próprio Moore, acreditam que a dobra da capacidade de processamento dos processadores a cada 18 meses não seja mais possível a partir de 2017. (Nota de rodapé, BEZERRA, 2015).

- Mas o que a “Lei” de Moore tem a ver com a modelagem de sistemas de *software*?
- Bezerra (2015) explica:

> O rápido crescimento da capacidade computacional das máquinas resultou na demanda por sistemas de *software* cada vez mais complexos, que tirassem proveito dessa nova capacidade. Por sua vez, o surgimento desses sistemas mais complexos resultou na necessidade de reavaliação da forma de desenvolver sistemas.

1.2. UML.

Unified Modeling Language / Linguagem Unificada de Modelagem.

1.2. UML - *Unified Modeling Language*/Linguagem de Modelagem Unificada

- A UML é uma linguagem-padrão para descrever/documentar projeto de *software*. (PRESSMAN, 2011). É usada para visualizar, especificar, construir e documentar os artefatos de um uma sistema de *software*-intensivo. (*ibidem apud* BOOCH; RUMBAUCH; JACOBSEN, 2005).
- Em outras palavras, assim como os arquitetos criam plantas e projetos para ser usados na construção, os arquitetos de *software* criam diagramas UML para ajudar na construção de sistemas de *softwares*.
- De acordo com Bezerra (2015), a UML teve muitos contribuintes, mas os principais autores no processo foram Grady Booch (*Booch Method*), James Rumbaugh (*OMT*) e Ivar Jacobson (*OOSE*), conhecidos como “os três amigos”.

cont. [...]

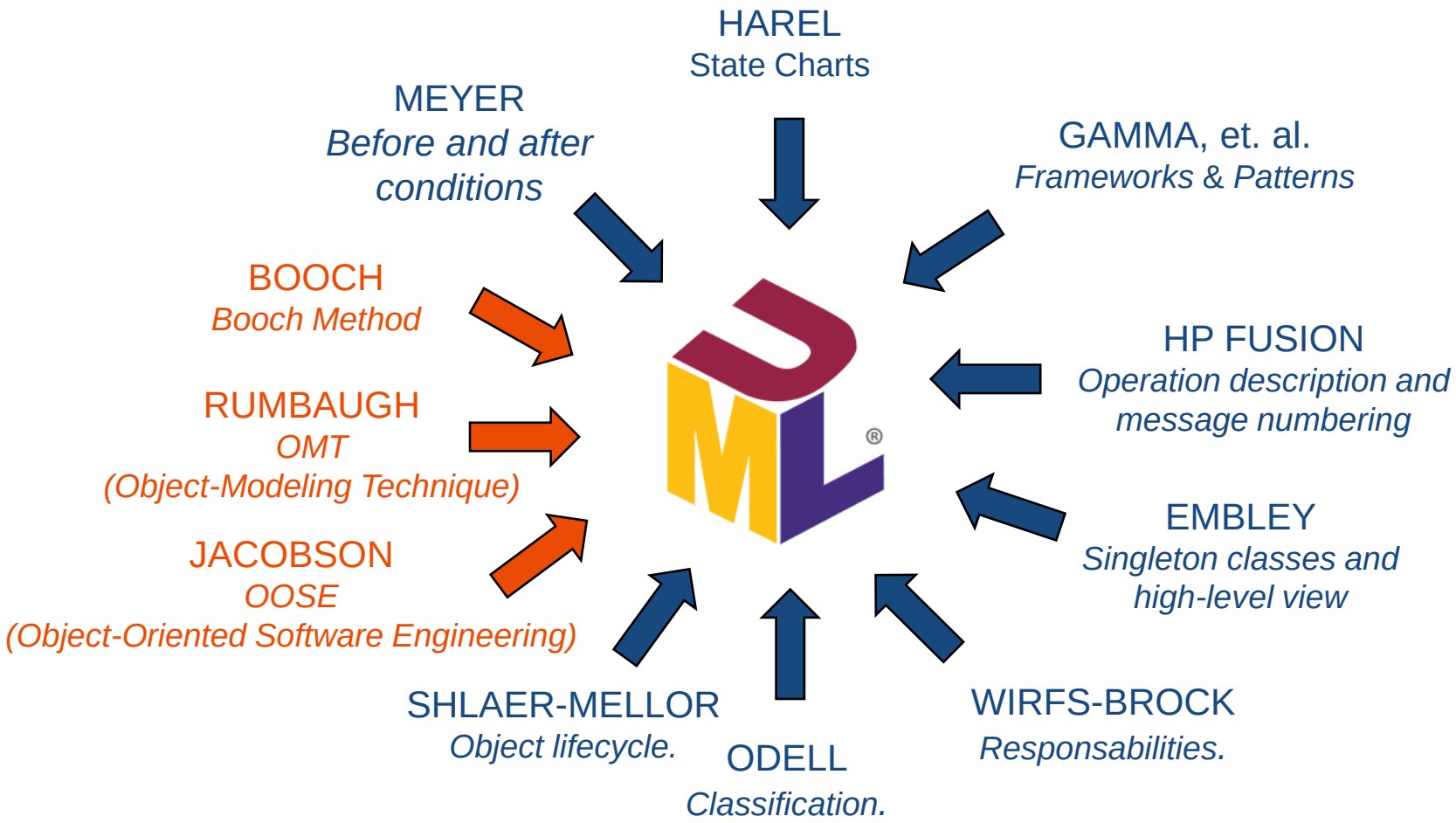


Figura 1. Técnicas e métodos que unificados formaram a UML.

- Os elementos gráficos possuem sintaxe e semântica. O primeiro corresponde à forma predeterminada de desenhar o elemento, já a segunda, define o que significa o elemento e com que objetivo ele deve ser utilizado. Considera-se que estes elementos são extensíveis, permitindo que a UML seja adaptada às características específicas de cada projeto de desenvolvimento. (BEZERRA, 2015).
- A UML é independente tanto de linguagens de programação quanto de processos de desenvolvimento. (*ibidem*).

2. A definição completa está na *Especificação da Linguagem de Modelagem Unificada* disponível gratuitamente no site da OMG (www.uml.org).

1.2.1. As visões do sistema

- A UML sugere que um sistema pode ser descrito em cinco visões interdependentes (BOOCH *et al.,* 2006 *apud* BEZERRA, 2015). Cada visão enfatiza aspectos variados do sistema. São elas:

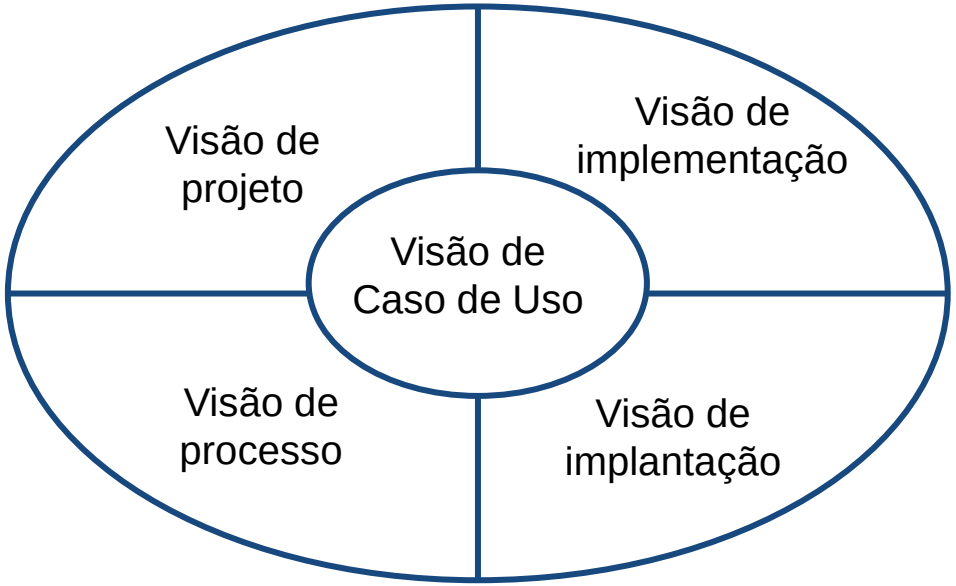


Figura 2. Visões (perspectivas) de um sistema de *software*.

- Visão de Casos de Uso: Descreve o sistema de um ponto de vista externo como um conjunto de interações com os agentes externos ao sistema.
- Visão de Projeto: Enfatiza as características do sistema, tanto estrutural quanto comportamental, às funcionalidades externamente visíveis do sistema.
- Visão de Implementação: Abrange o gerenciamento de versões do sistema, construídas pelo agrupamento de módulos (componentes) e subsistemas.
- Visão de Implantação: Corresponde à distribuição física do sistema em seus subsistemas e à conexão entre essas partes.
- Visão de Processo: Esta visão enfatiza as características de concorrência (paralelismo), sincronização e desempenho do sistema.

- Dependendo das características e da complexidade do sistema, nem todas as visões precisam ser construídas. (BEZERRA, 2015).
- Por exemplo, se o sistema tiver de ser instalado em um ambiente computacional de processador único, não há necessidade da visão de implantação. (*ibidem*).
- Outro exemplo: se o sistema for constituído em um único processo, a visão de processo é irrelevante. (*ibidem*).
- De forma geral, dependendo do sistema, as visões podem ser ordenadas por grau de relevância. (*ibidem*).

1.3. Diagramas UML.

Os modelos gráficos [...]

1.3. Diagramas UML

- Um processo de desenvolvimento que utilize a UML como linguagem de suporte à modelagem envolve a criação de diversos documentos.
- Esses documentos podem ser textuais ou gráficos.
- Na terminologia são denominados de artefatos de *software*, ou simplesmente artefatos.
- Esses artefatos compõem as visões do sistema.
- No total são 13 artefatos de acordo a versão UML 2.0. Na versão 2.5 são 16 artefatos. Para Bezerra (2015), na versão 2.0 os diagramas são mais que suficiente para a modelagem de sistemas.

cont. [...]

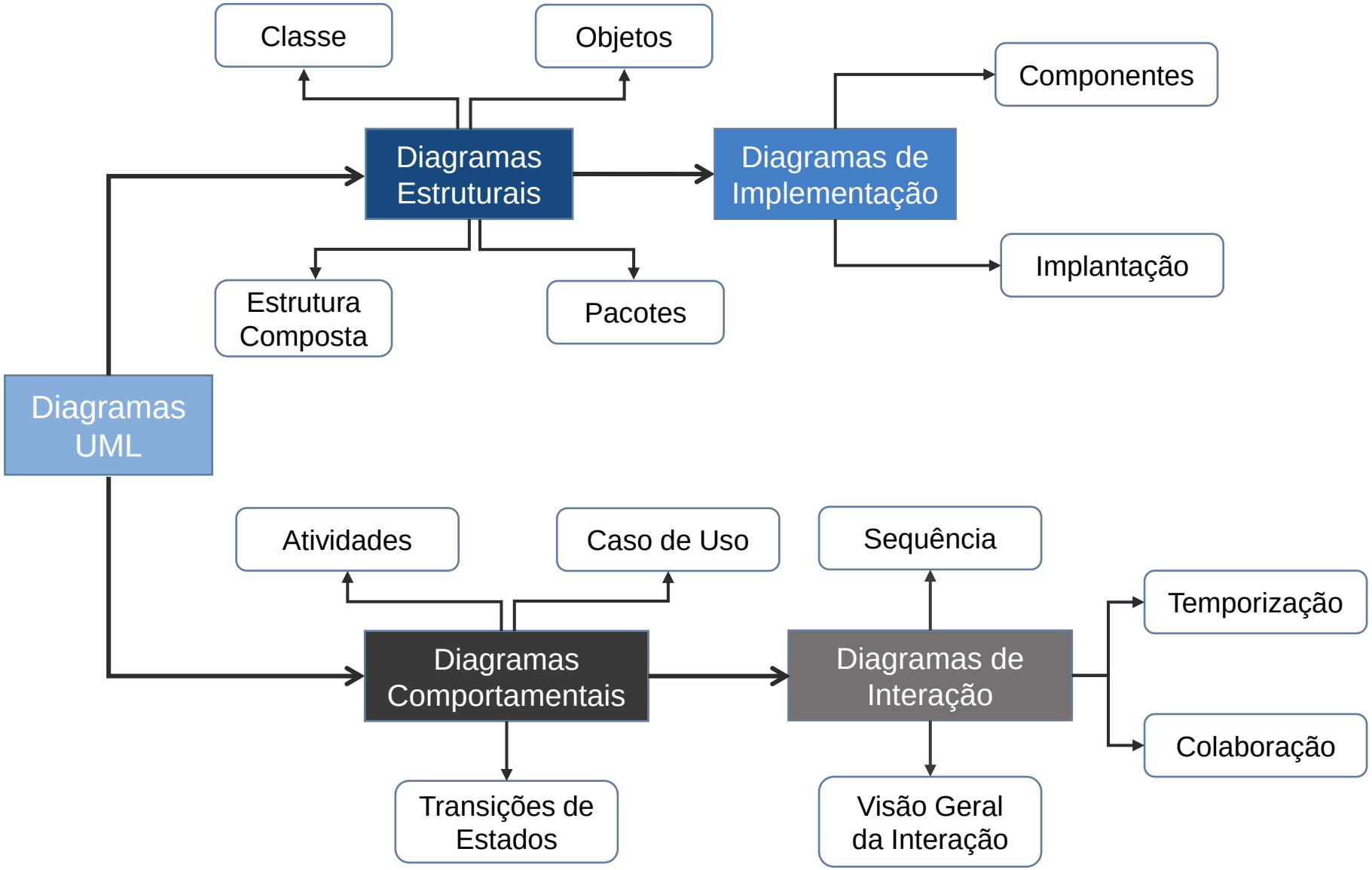


Figura 3. Diagramas definidos pela UML (versão 2.0).

1.4. Mecanismos gerais.

Os três grandes componentes [...]

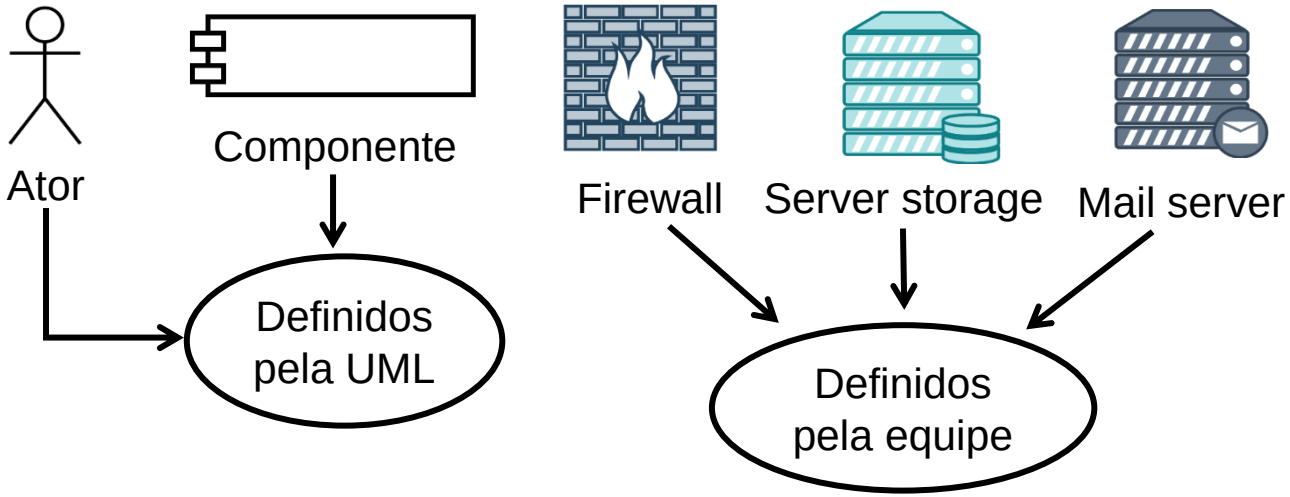
1.4. Introdução

- A UML consiste em três grandes componentes:
    1. Blocos de construção básicos;
    2. Regras associadas ao uso desse blocos; e
    3. Mecanismos de uso geral.
- Nesta seção faremos uma breve introdução sobre os mecanismos de uso geral utilizados em sua grande maioria nos diagramas da UML.

1.4.1. Estereótipos

- Segundo Bezerra (2015), estereótipos são mecanismos de uso geral, que é utilizado para estender o significado de determinados elementos em um diagrama.
- São divididos em dois tipos: predefinidos ou definidos.
- Outra classificação aplicada aos estereótipos é quanto à sua forma: gráficos (ou ícones) e textuais.

- Exemplos de estereótipos gráficos (ícones)



- Exemplos de estereótipos textuais (rótulos)

<< document >>

<< interface >>

<< control >>

e

<< entity >>

Figura 4. Exemplos de estereótipos gráficos da UML (versão 2.0).

1.4.2. Notas explicativas, Etiquetas valoradas (*Tagged values*) e Pacotes (*Packages*).

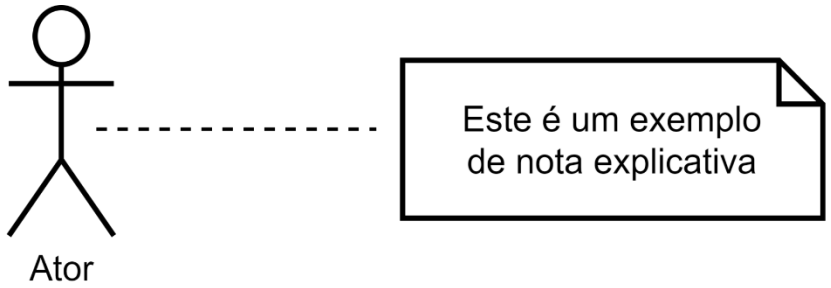


Figura 5. Exemplo de nota explicativa.

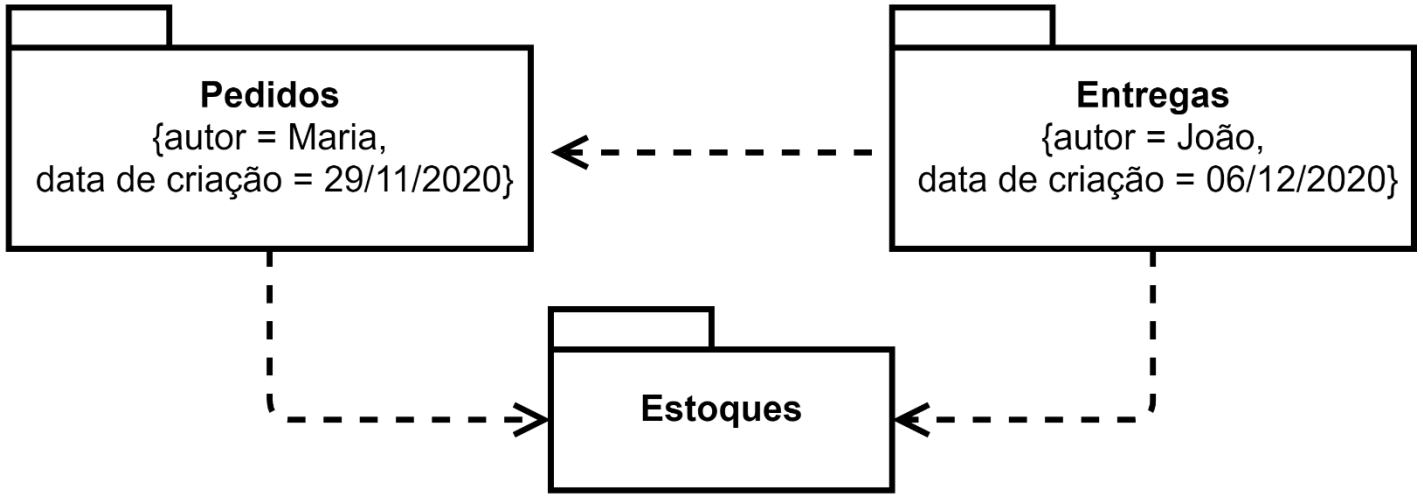


Figura 6. Exemplo de etiquetas valoradas.

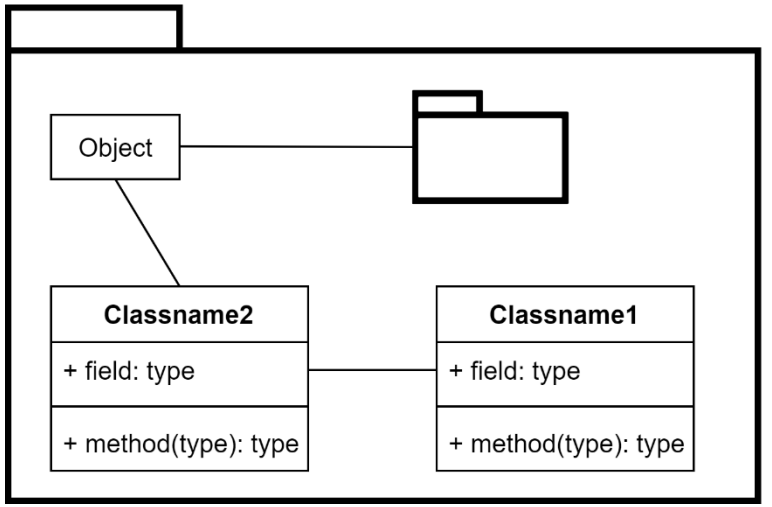


Figura 7. Exemplo de pacote(s).

1.5. Modelagem de Caso de Uso.

Conceitos e diagramas.

1.5.1. Modelagem de Caso de Uso

- O modelo de caso de uso (MCU) é uma representação das funcionalidades externamente observáveis do sistema e dos elementos externos ao sistema que interagem com ele. (BEZERRA, 2015). A ferramenta da UML utilizada nesta modelagem é o Diagrama de Casos de Uso.
- Por definição, um caso de uso (*use case*) é a especificação de uma sequência completa de interações entre um sistema e um ou mais agentes externos a esse sistema. Representa um relato de certa funcionalidade sem revelar a estrutura e o comportamento internos desse sistema. (BEZERRA, *ibidem*, grifo meu).
- As principais terminologias usadas são:
  - Ator: Elemento externo que interage com o sistema. Envia informações para processar e/ou recebe informações processadas. [...]. Pode relacionar-se com mais de um caso de uso.

- Relacionamento: Casos de usos e atores não existem sozinhos. Eles precisam se relacionar. A UML define os seguintes relacionamentos para o modelo de casos de uso: comunicação, inclusão, extensão e generalização.
    1. Comunicação: Associa o caso de uso ao ator. Interage e troca informações com o sistema caso de uso.

Figura 8. Exemplo de relacionamento comunicação.

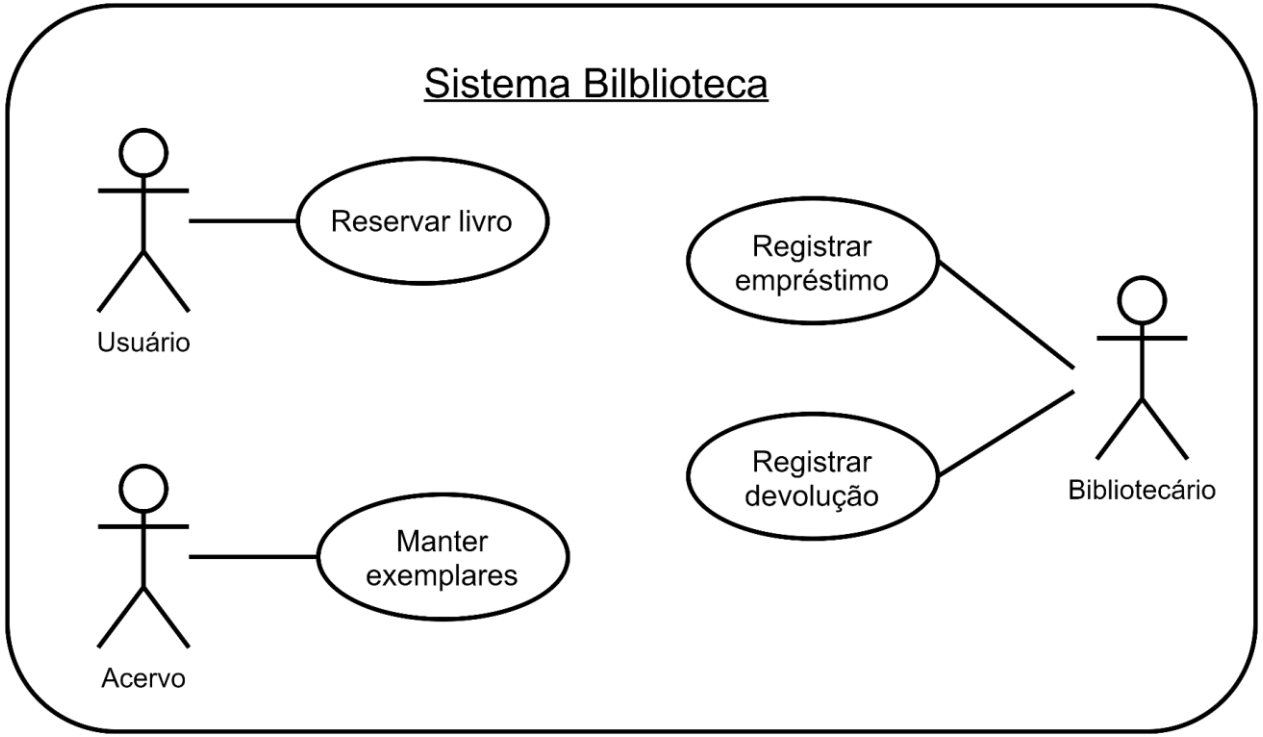


Figura 9. Exemplo de um caso de uso usando fronteira. Sistema biblioteca. (BEZERRA, 2015 *adaptado* CAIXETA, 2020).

2. Inclusão: Existente apenas entre casos de usos. Neste relacionamento existe o *inclusor* que é aquele que inclui, ou seja, provoca o *inclusor*, que recebe um comportamento do sistema.

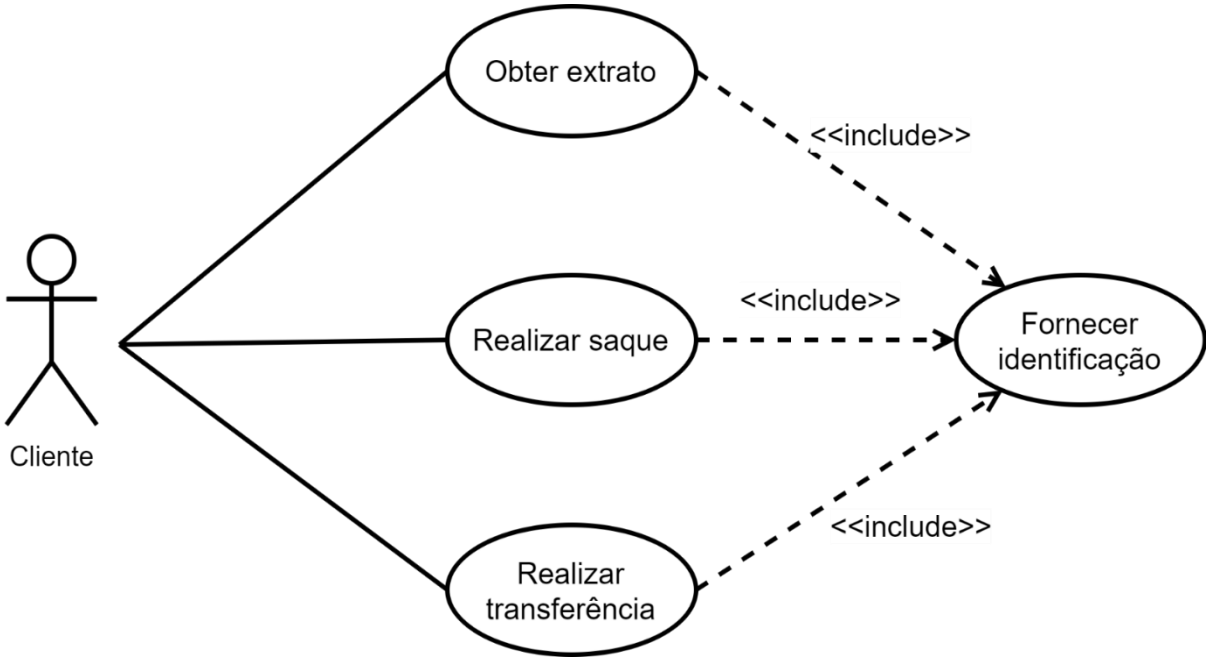


Figura 10. Exemplo de relacionamento inclusão (*include*). (BEZERRA, 2015, *adaptado* CAIXETA, 2020).

3. Extensão: Utilizado para modelar situações em que diferentes sequências de interações podem ser inseridas em um mesmo caso de uso. Cada sequência é um comportamento eventual. Ocorre sob certas condições ou com a escolha do ator.

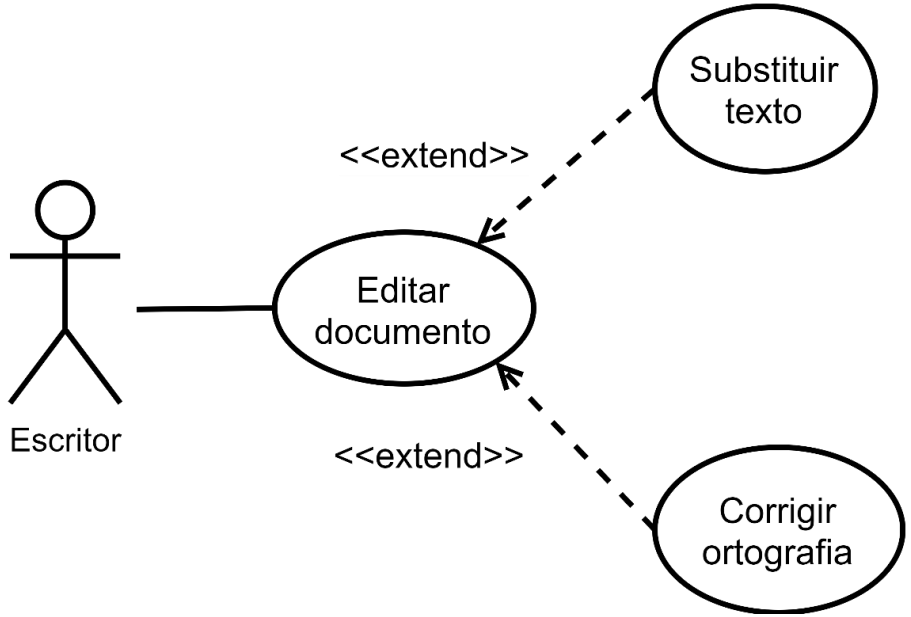


Figura 11. Exemplo de relacionamento extensão (*extend)*. (BEZERRA, 2015 *adaptado* CAIXETA, 2020).

4. Generalização: Existe entre dois casos de uso ou entre dois atores. Permite que um caso de uso (ou um ator) herde características de outro, mais genérico, este último normalmente chamado de caso de uso (ator) base. O caso de uso (ator) herdeiro pode especializar o comportamento do caso de uso (ator) base.

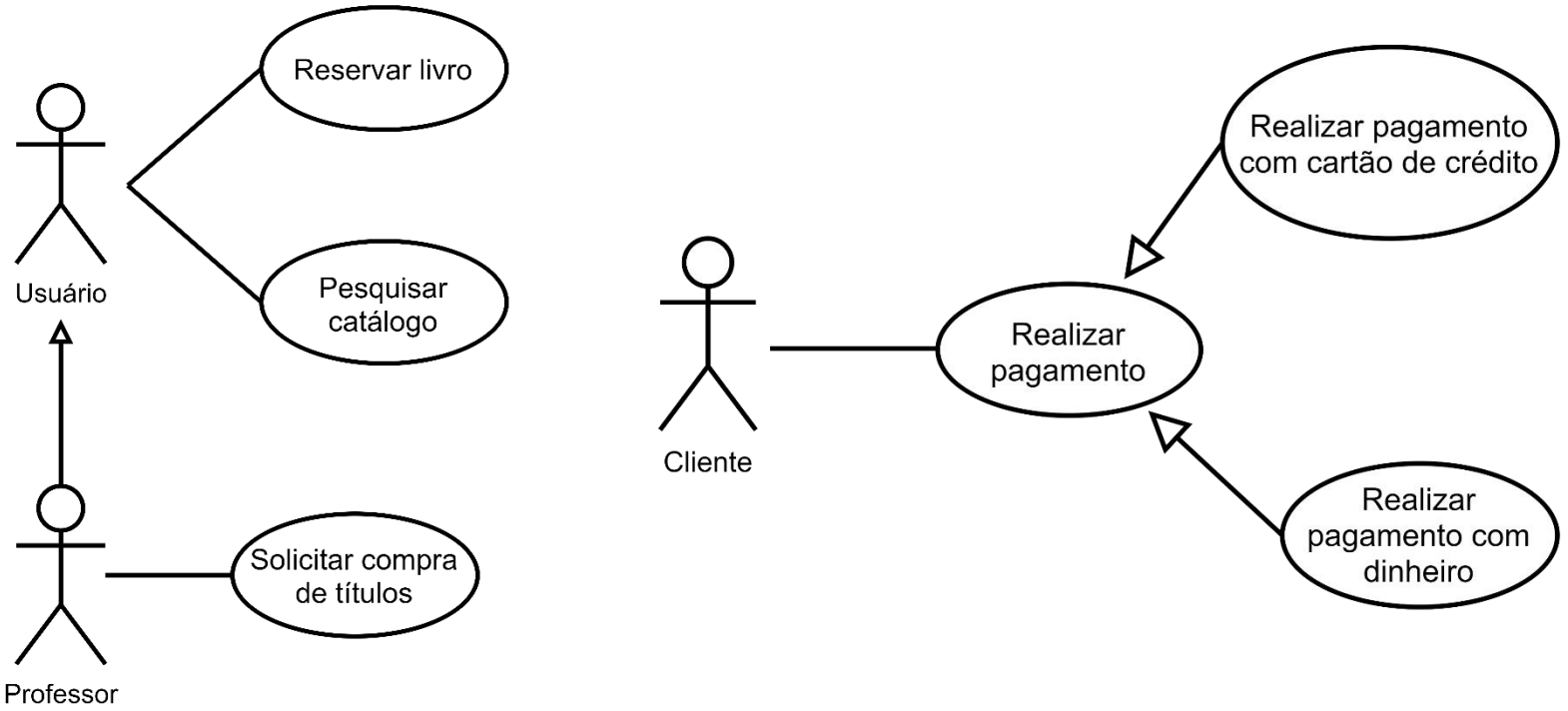


Figura 12. Exemplo de relacionamento de generalização (heranças). A figura à esquerda representa um sistema de biblioteca, já à direita, um sistema bancário em operação. (BEZERRA, 2015 *adaptado* CAIXETA, 2020).

- Um ponto importante está nas possibilidades de relacionamento existente entre os elementos de caso de uso. (BEZERRA, 2015).

Elementos	Relacionamentos			
	Comunicação	Extensão	Inclusão	Generalização
Caso de uso e caso de uso		X	X	X
Ator e ator				X
Caso de uso e ator	X			

- De acordo com Pressman (2011), diagramas de casos de uso, são bons auxílios para assegurar a inclusão de toda a funcionalidade do sistema. Mas tenha em mente que a maior contribuição para os processos de desenvolvimento de *software* é a descrição textual de cada caso e não o diagrama geral.

1.6. Modelagem de Classe de Análises.

Conceitos e diagramas.

1.6. Modelagem de Classes de Análises (MCA)

- Externamente ao sistema, os atores visualizam resultados [...]. Internamente, os objetos colaboram uns com os outros para produzir os resultados visíveis de fora. (BEZERRA, 2015).
- Essa colaboração pode ser vista sob o aspecto *dinâmico* e o *estrutural estático*. Do primeiro, descreve-se as trocas de mensagens e reações a eventos que ocorrem no sistema. Já do segundo, permite-se compreender como o sistema está estruturado internamente para que as funcionalidades externamente visíveis sejam produzidas. (BEZERRA, 2015).

1.6.1. As classes

- Uma classe é uma descrição da abstração de um conjunto de objetos com características similares.
- Pode ser definida também como uma descrição das propriedades ou estados possíveis, bem como os comportamentos ou ações aplicáveis aos objetos.
- A Figura 13 apresenta a construção gráfica de uma classe em O.O.

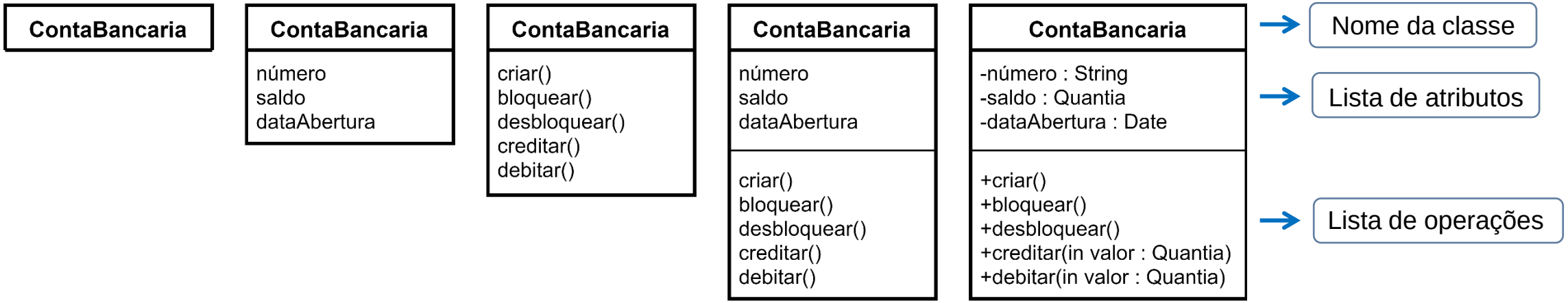


Figura 13. Exemplos de representação de uma mesma classe ContaBancaria em diferentes graus de abstração. (BEZERRA, 2015).

1.6.2. As associações entre classes

- Um ponto importante a respeito de objetos de um sistema é o fato de que eles podem se relacionar uns com os outros.
- A existência de um relacionamento entre dois objetos possibilita a troca de mensagens entre os mesmos. Portanto, em última análise, relacionamentos entre objetos permitem que eles colaborem entre si para produzir as funcionalidades do sistema.
- Esse tipo de relacionamento denominamos associação. Alguns exemplos:

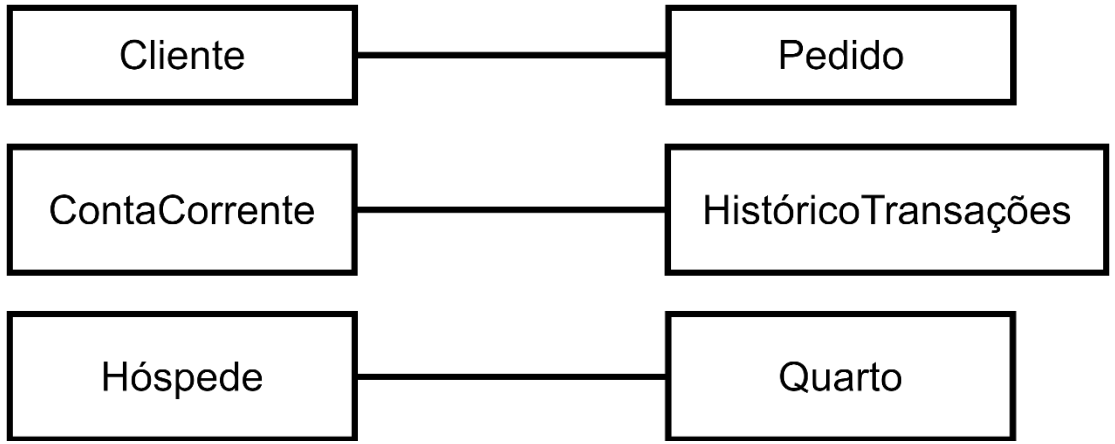


Figura 14. Exemplos de associações entre os objetos. (BEZERRA, 2015).

- As associações possuem diversas características importantes: multiplicidades, nome, direção de leitura, papéis, tipo de participação e conectividade. Nas próximas seções, descreveremos algumas dessas características.

1. Multiplicidades

São os limites de associações tanto inferior como superior entre as classes.

Nome	Simbologia
Apenas Um	1
Zero ou Muitos	0...*
Um ou Muitos	1...*
Zero ou Um	0...1
Intervalo Específico	1i...1s

Figura 15. Exemplos do uso de multiplicidade.

- Existem infinitas possibilidades de associação entre objetos. No entanto, essas associações podem ser agrupadas em apenas três tipos: “muitos para muitos” (N:N), “um para muitos” (1:N) e “um para um” (1:1). Denomina-se conectividade o tipo de associação entre duas classes. A conectividade da associação entre duas classes depende dos símbolos de multiplicidade que são utilizados na associação.

Conectividade	Multiplicidade de um extremo	Multiplicidade de outro extremo
Um para Um	0..1 ou 1	0..1 ou 1
Um para Muitos	0..1 ou 1	* ou 1..* ou 0..*
Muitos para Muitos	* ou 1..* ou 0..*	* ou 1..* ou 0..*

- Na próxima página, serão apresentados três exemplos de conectividade possíveis. Os exemplos são de Bezerra (2015).

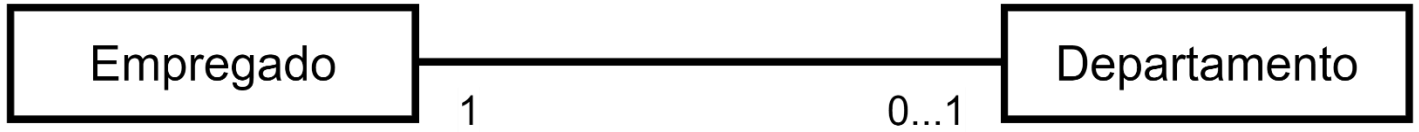


Figura 16. Um para um ou 1:1: Indica que um empregado pode gerenciar um e somente um departamento. Um departamento, por sua vez, possui um único gerente.

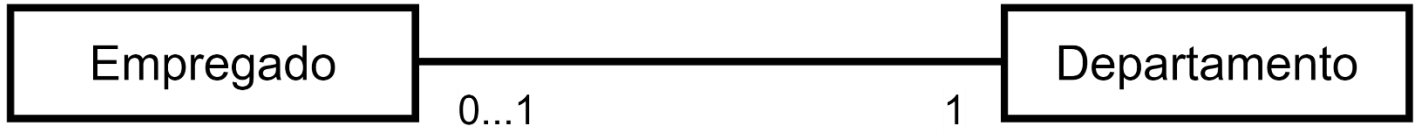


Figura 17. Um para muitos ou 1:N: Indica que um empregado está lotado em um único departamento, mas um departamento pode ter diversos empregados.

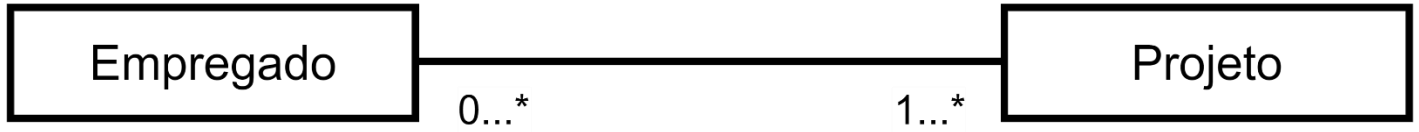


Figura 18. Muitos para muitos ou N:N: Indica que um projeto pode ter diversos empregados. Além disso, um empregado pode trabalhar em diversos projetos.

2. Nome de associação, direção de leitura e papéis

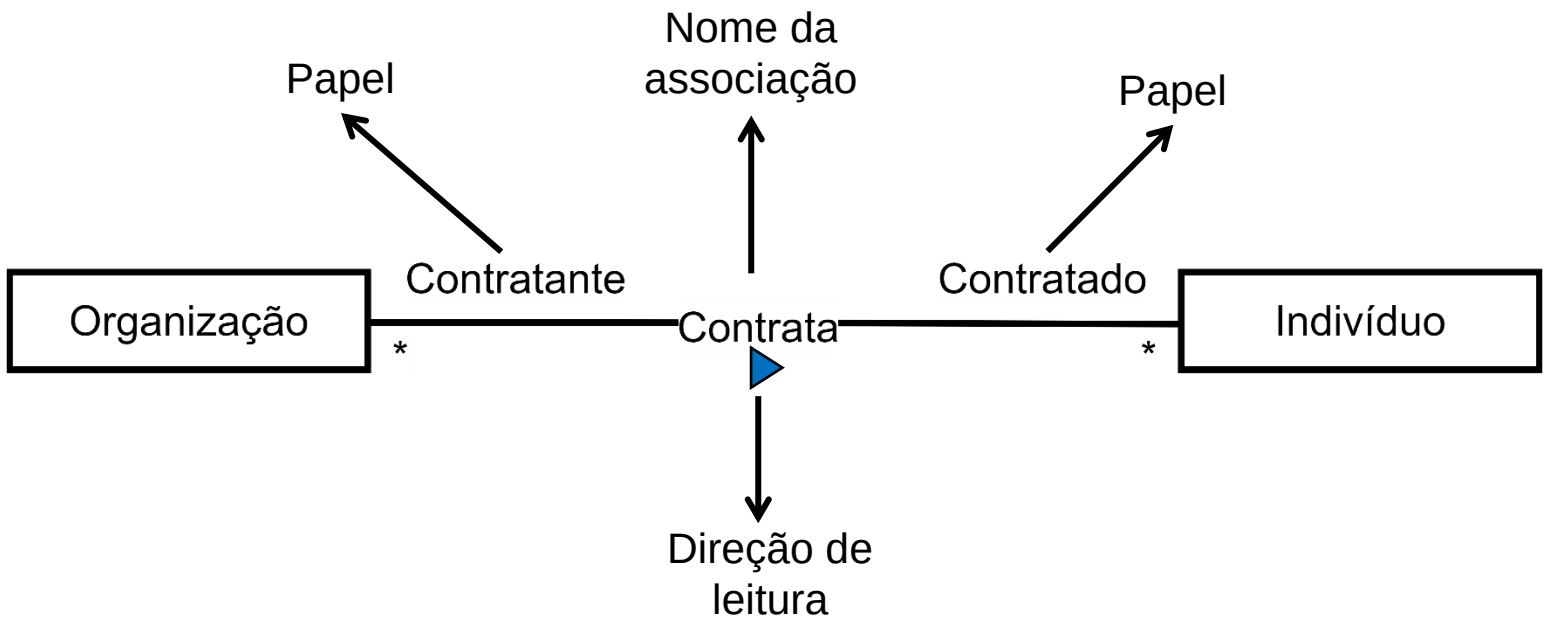


Figura 19. Exemplo de utilização de nome de associação, direção de leitura e de papéis. (BEZERRA, 2015).

3. Classes associativas

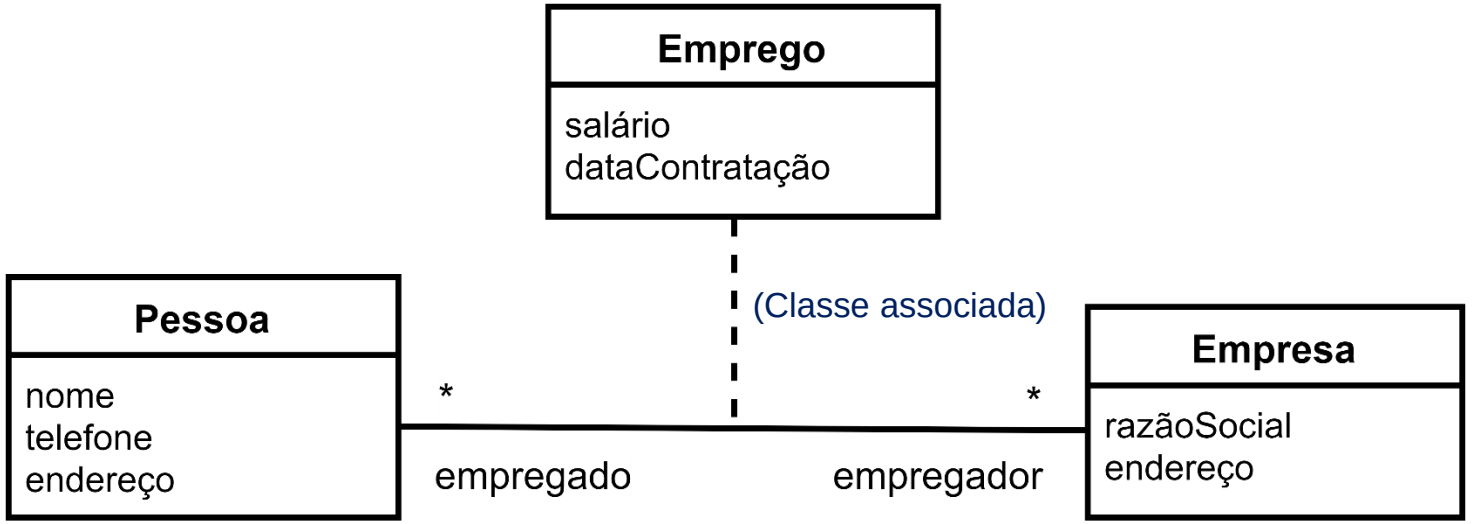


Figura 20. Exemplo de uma classe associativa. (BEZERRA, 2015).

4. Agregações e composições

- De todos os significados diferentes que uma associação pode ter, há uma categoria especial de significados que representa relações todo-parte. (BEZERRA, 2015).
- Esse tipo de relação entre dois objetos indica que um deles está contido no outro, ou um objeto contém o outro. Por exemplo:

$$A \subset B \quad \text{ou} \quad B \supset A$$

- A UML define dois tipos de relacionamentos todo-parte, a agregação e a composição, que são casos especiais da associação.
- Todas as características válidas anteriormente, vale também para esses dois relacionamentos.

- Bezerra (2015), apresenta algumas características particulares que as diferem das associações simples. São elas:
  - ✓ Agregações/composições são *assimétricas*, no sentido de que, se um objeto A é parte de um objeto B, o objeto B não pode ser parte do objeto A.
  - ✓ Agregações/composições propagam comportamento, de forma que um comportamento que se aplica a um todo automaticamente se aplica também às suas partes.
  - ✓ Nas agregações/composições, as partes são normalmente criadas e destruídas pelo todo. Na classe do objeto todo, são definidas operações para adicionar e remover as partes.

cont. [...]

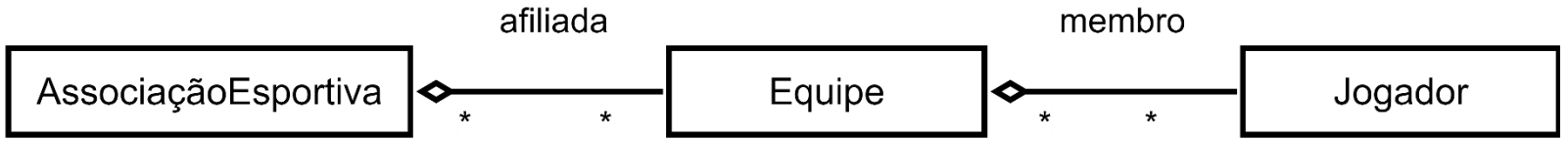


Figura 21. Exemplo de agregação. (BEZERRA, 2015).

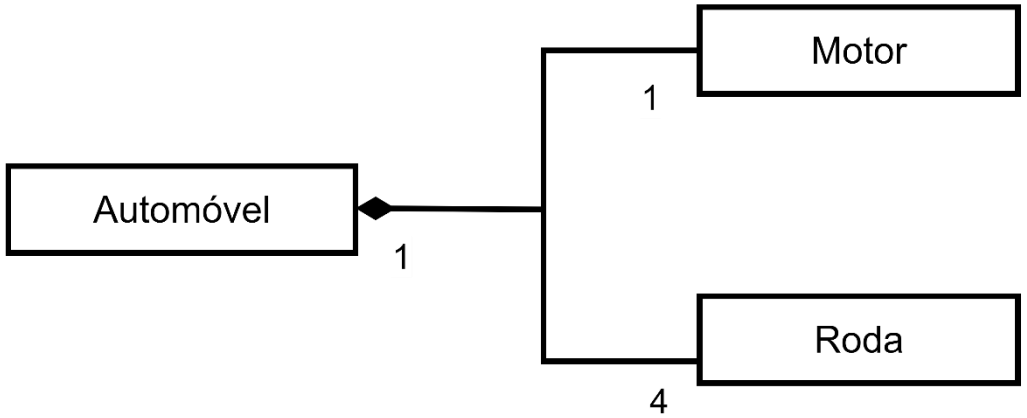


Figura 22. Exemplo de composição. (BEZERRA, 2015).

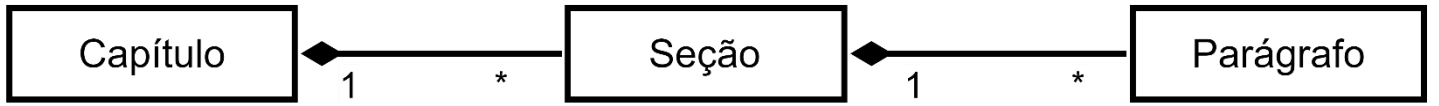


Figura 23. Exemplo de hierarquia de composição. (BEZERRA, 2015).

5. Generalizações e Especializações

- Além de relacionamentos entre objetos, o modelo de classes também pode representar relacionamentos entre classes. Esses últimos denotam relações de generalidade ou especificidade entre as classes envolvidas. (BEZERRA, 2015).
- É também conhecido como herança, ou simplesmente gen./espec. Isso porque são dois pontos de vista do mesmo relacionamento: dadas duas classes A e B, se A é uma generalização de B, então B é uma especialização de A.

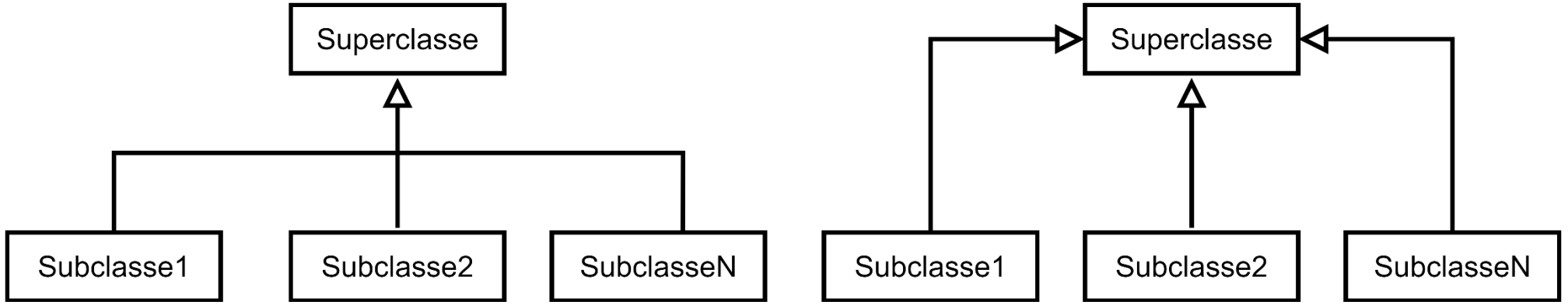


Figura 24. Representações alternativas para o relacionamento de generalização. (BEZERRA, 2015).

cont. [...]

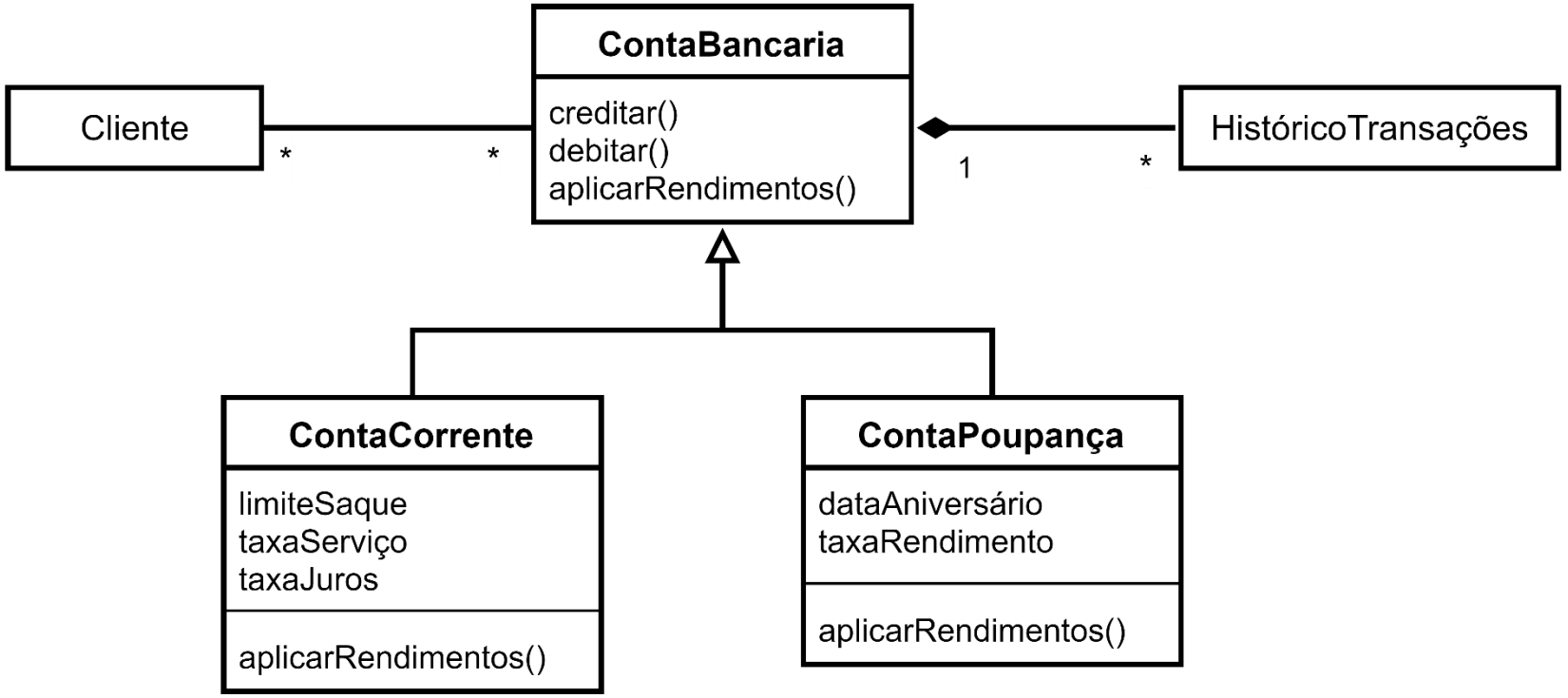


Figura 25. Conformidade das subclasses à superclasses no relacionamento de generalização. (BEZERRA, 2015).

1.7. Modelagem de um sistema acadêmico (Exemplo).

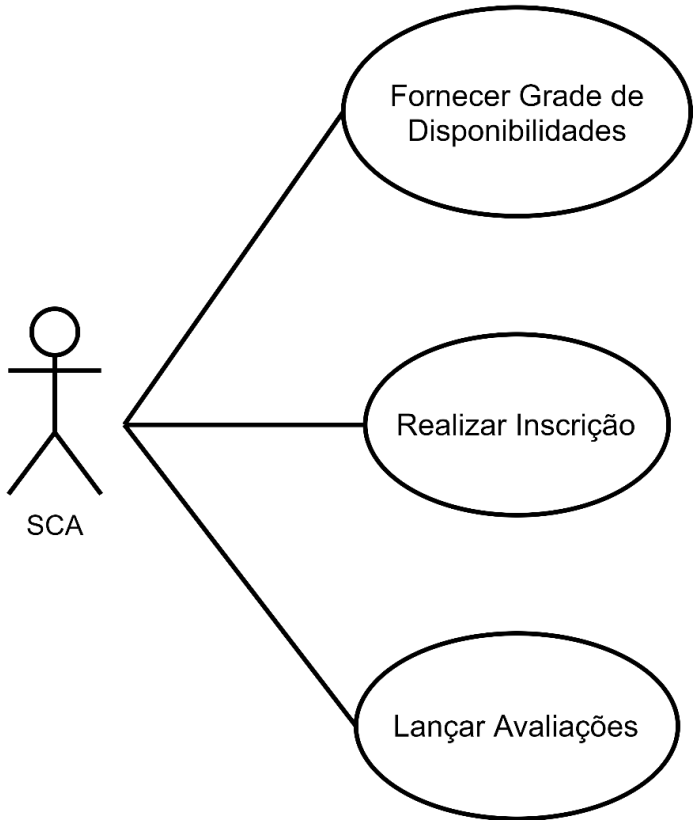
Diagrama Caso de Uso + Classes.

1.7.1. Um estudo de caso [...]

- De acordo com Bezerra (2015), em um desenvolvimento dirigido a casos de uso, após a descrição dos casos essenciais, é possível realizar a identificação das classes. [...]. Após a identificação, essas são refinadas para retirar inconsistências e redundâncias e, finalmente, são documentadas, e dá-se o início à construção do diagrama de classes.
- Nesta seção apresentaremos um exemplo de construção de artefatos que farão parte da composição inicial da arquitetura de um sistema de informação para cadastro acadêmico. Observa-se que esta arquitetura representa um sistema real, comumente usado pelas IES.
- Os casos de usos que serão construídos são:

1. Fornecer Grades de Disponibilidade;
2. Realizar Inscrição;
3. Lançar Avaliações.

a) SCA: Sistema de Cadastro Acadêmico



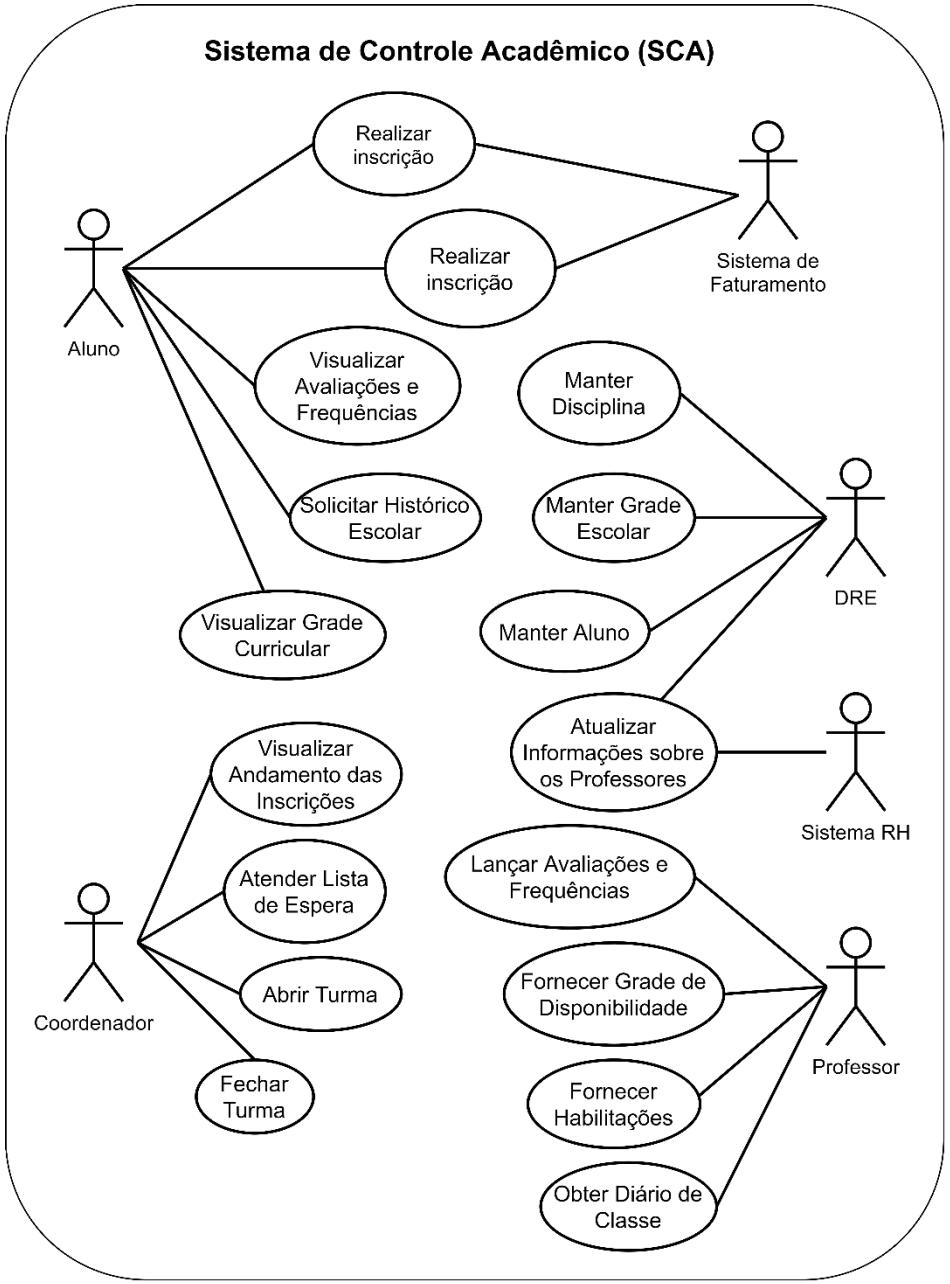


Figura 26. Exemplos de dois casos de uso, um básico (esquerda), e o outro global (direita) de um SCA. (BEZERRA, 2015).

b) SCA: Diagrama de Classe do Caso de Uso - Fornecer Grade de Disponibilidade

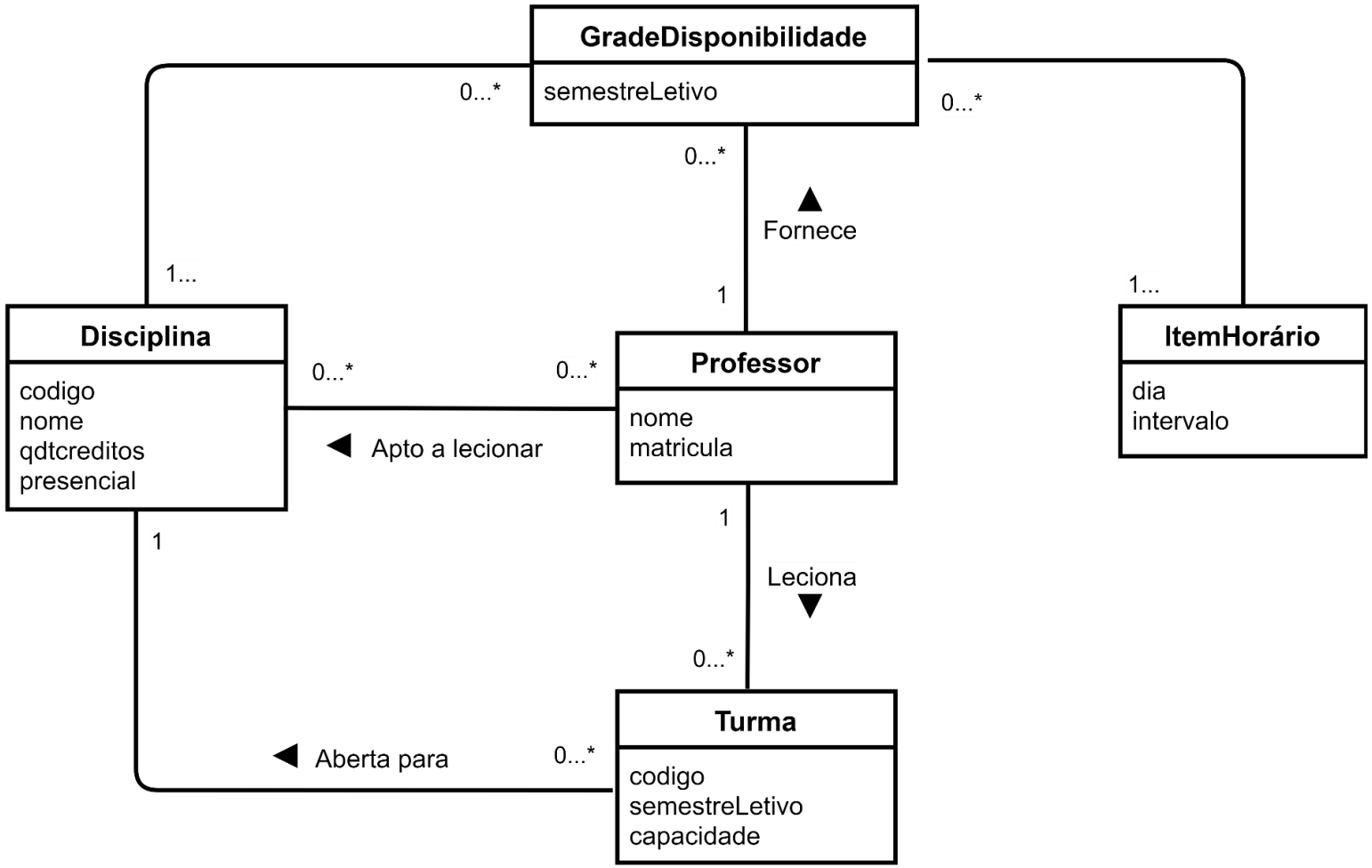


Figura 27. Diagrama de classe do caso de uso Fornecer Grade de Disponibilidade (Arquitetura versão 1). (BEZERRA, 2015).

c) SCA: Diagrama de Classe do Caso de Uso - Realizar Inscrição

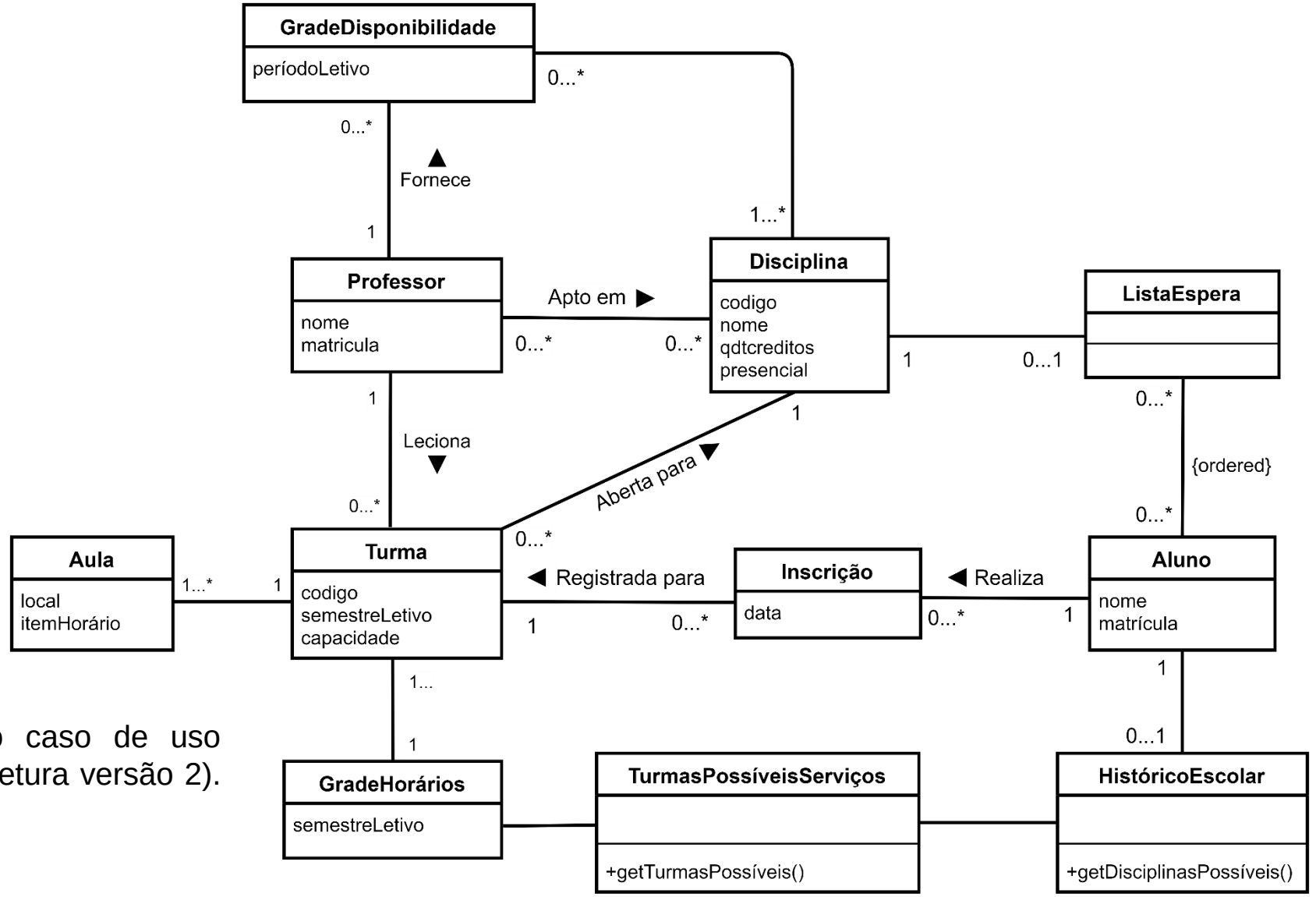


Figura 28. Diagrama de classe do caso de uso Realizar Inscrição (Arquitetura versão 2). (BEZERRA, 2015).

d) SCA: Diagrama de Classe do Caso de Uso - Lançar Avaliações

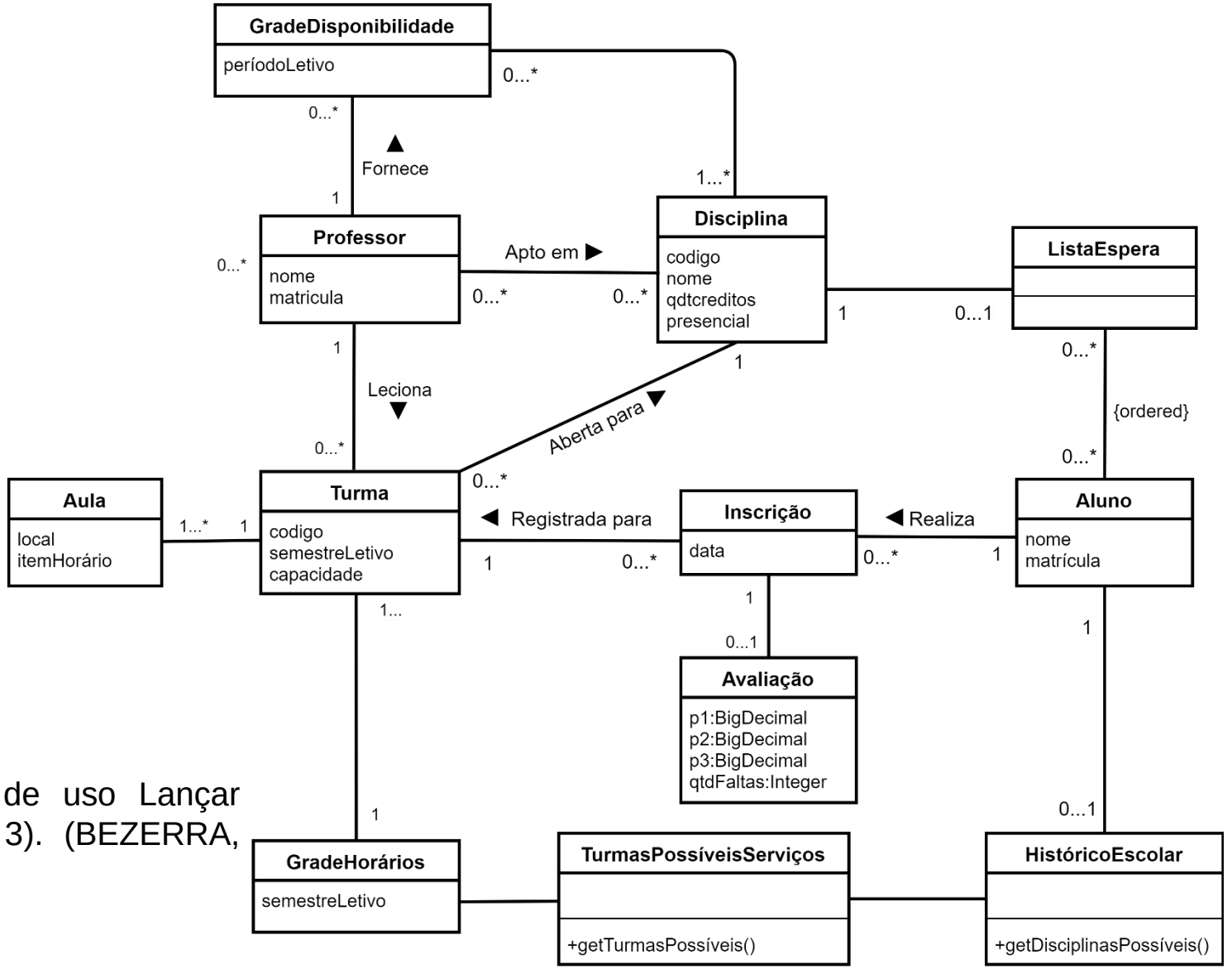


Figura 29. Diagrama de classe do caso de uso Lançar Avaliações (Arquitetura versão 3). (BEZERRA, 2015).

1.7.2. Análises das Regras de Negócios

- Segundo Bezerra (2015), durante a criação do MCU e do MCA torna-se necessário também analisar o modelo de regras de negócio (MRN).
- É recomendável que os modeladores sempre dêem uma repassada no MRN à procura de informações que irão gerar alterações nos modelos. (*ibidem*).
- Abaixo apresentamos para cada regra de negócio do SCA seu efeito no modelo de classes de análise inicial. (*ibidem*).

RN	Comportamento
RN00	O sistema deve detectar “choques de horários” nas inscrições realizadas pelo(a) discente no semestre letivo.
RN01	A classe Aluno deve estar associada às inscrições. Cada inscrição está associada à uma turma que, por sua vez, está associada a uma disciplina, que possui uma quantidade de créditos. Uma vez fixada as inscrições, é possível determinar a soma dos créditos correspondentes.
RN02	Implica em conhecer a capacidade máxima da turma. O atributo é da classe Turma, além disso, a turma possui referências para as inscrições por meio da associação Registrada para.

cont. [...]

RN	Comportamento
RN03	Atribui responsabilidade à classe Disciplina, que correspondente à necessidade de saber que disciplinas servem como pré-requisitos (caso existam). O elemento de modelagem criado é a auto associação entre objetos da classe Disciplina.
RN04	Nessa regra faz-se necessário que o sistema tenha conhecimento a respeito de que disciplinas um professor está apto a lecionar. Para isso, foi criada a associação de nome Apto a lecionar entre as classes Disciplina e Professor.
RN05	Implica na responsabilidade de saber qual a situação de um(a) discente inscrito em uma turma. A classe para satisfazer essa responsabilidade é Avaliação, posto que está associada tanto a Turma quanto a Aluno.
RN06	Implica na responsabilidade do sistema de conhecer a quantidade de reprovações de um(a) aluno(a). Do ponto de vista estrutural, é possível saber se o aluno A foi reprovado em uma turma T: basta determinar o objeto inscrição correspondente ao par (A,T) e verificar a situação do objeto Avaliação associado. No modelo de caso de uso inicial, não foi considerado a funcionalidade cancelamento de matrícula por um(a) aluno(a). Temos aqui, portanto, um exemplo em que a modelagem de classes gera informações para o modelador refinar este artefato. Essa situação ocorre comumente em análise de sistema.
RN07	Essa regra resultou na utilização da restrição {ordered} atrelada à associação entre Aluno ListaEspera.

1.7.3. Documentação das responsabilidades

- Abaixo apresentemos possíveis formas de documentação de responsabilidades de análise para algumas classes do SCA.

Disciplinas	
1.	Conhecer seus pré-requisitos.
2.	Conhecer seu código, nome e quantidade de créditos.

ListaEspera	
1.	Conhecer a disciplina para a qual foi criada.
2.	Manter os alunos em espera pela abertura de vagas.

Aluno	
1.	Conhecer seu número de registro e nome.
2.	Conhecer as disciplinas que já cursou.

Turma	
1.	Conhecer o seu código e situação.
2.	Conhecer a disciplina para a qual é ofertada.

HistoricoEscolar	
1.	Permitir trancamentos, aprovações e reprovações.
2.	Conhecer disciplinas não cursadas pelo(a) aluno(a).
3.	Conhecer já e não cursadas pelo(a) aluno(a).

InscreverUmAluno	
1.	Conhecer o semestre letivo.
2.	Conhecer dias, horários e salas de aula.
3.	Conhecer a quantidade máxima de aluno(a)s da turma.
4.	Conhecer o professor.

1.7.4. Glossário de conceitos

- Durante a modelagem de classes, os termos relevantes do domínio do problema (*i.e.*, os correspondentes às classes identificadas) devem ser definidos em um glossário. (BEZERRA, 2015). No caso do SCA, a definição inicial é apresentada a seguir:

1. Aluno: Representa um(a) discente, que se inscreve em turmas.
2. Avaliacao: Representa notas e frequência obtidas por um(a) aluno(a) em uma turma. A avaliação é lançada por um professor(a) e reflete o aproveitamento do(a) aluno(a) na turma correspondente.
3. Aula: Representa o acontecimento de um dos encontros semanais em alguma turma. Toda aula acontece em um local e está associada a um item de horário (ItemHorario).
4. DiarioClasse: Corresponde a um formulário que contém os nomes do(a)s aluno(a)s inscritos em determinada turma. Nesse formulário, o(a) professor(a) lança as avaliações (notas e quantidade de faltas) de cada aluno(a).
5. Disciplina: É um componente da grade curricular. Pode possuir *pré-requisitos*, que também são disciplinas.
6. GradeDisponibilidades: Representa as informações fornecidas por um(a) professor(a) para indicar em que dias e horários está disponível para lecionar, assim como quais disciplinas está disponível para lecionar. Uma grade de disponibilidade é definida para um determinado SemestreLetivo [...]

e composta pelos dias da semana (e os respectivos intervalos de tempo) nos quais um(a) professor(a) está disponível para lecionar. Representa também as disciplinas que o(a) professor(a) estará disponível para lecionar nesse mesmo semestre letivo. As informações de grades de disponibilidade de um determinado período letivo são usadas para montar a grade de horários daquele período.

7. GradeHorarios: Todas as turmas ofertadas para um determinado semestre letivo (com seus horários, locais de aula e professores responsáveis) compõem a grade de horários daquele semestre. A cada período letivo, é definida uma nova grade de horários. A grade de horários de um período letivo é necessária para permitir a inscrição de aluno(a)s em turmas daquele período.
8. GradeCurricular: Corresponde a um conjunto de disciplinas mais pré-requisitos. De tempos em tempos, a instituição de ensino atualiza a grade curricular do curso. Nessa atualização novas disciplinas podem ser criadas, assim como disciplinas existentes podem ser removidas da grade curricular. É também possível que pré-requisitos sejam redefinidos em uma alteração da grade curricular. Uma grade curricular entra em vigência (passa a valer) em um determinado semestre letivo e sai de vigência em outro semestre letivo.
9. Intervalo: Corresponde a um intervalo de tempo. Formado por um horário de início e um horário de término. Serve para delimitar a duração da ocorrência com precisão de minutos. Por exemplo, o intervalo de tempo (18h20, 21h50) possui duração igual a 210 minutos.

10. ItemHorario: É a unidade mínima de alocação de aulas para uma turma em uma semana. Um item de horário é formado por um dia da semana e por um intervalo de tempo de duração fixa (*e.g.*, 50 minutos). As aulas de uma turma ocorrem em um ou mais itens de horário semanais, que podem ser contíguos ou não. Por exemplo, a tabela apresentada a seguir contém 17 itens de horário, cada um deles com duração de 50 minutos e distribuídos em turnos (manhã, tarde ou noite).

ID	Manhã	Tarde	Noite
1º	07:00 - 07:50	12:40 - 13:30	18:20 - 19:10
2º	07:55 - 08:45	13:35 - 14:25	19:10 - 20:00
3º	08:50 - 09:40	14:30 - 15:20	20:00 - 20:50
4º	09:55 - 10:45	15:35 - 16:25	21:00 - 21:50
5º	10:50 - 11:40	16:30 - 17:20	21:50 - 22:40
6º	11:45 - 12:35	16:30 - 17:20	

11. HistoricoEscolar: Um histórico escolar está associado a um(a) aluno(a) e representa o registro do desempenho acadêmico desse(a) aluno(a) nas diferentes turmas em que ele(a) participou. É responsabilidade dessa classe registrar as aprovações e reprovações obtidas pelo(a) aluno(a). É também sua responsabilidade manter o conhecimento acerca de quais disciplinas da grade curricular o(a) aluno(a) já venceu. Diz-se que um(a) aluno(a) venceu uma disciplina se ele(a) cursou com aprovação alguma turma ofertada para essa disciplina.

12. ListaEspera: Representa uma fila de aluno(a)s que estão esperando a abertura de uma Turma para determinada Disciplina.
13. Local: Representa um dos locais que podem ser utilizados para as aulas de uma ou mais turmas.
14. Inscricao: Representa o registro da inscrição de um(a) aluno(a) em alguma turma. Uma inscrição é posteriormente associada a uma avaliação (Avaliacao).
15. Professor: Representa o indivíduo que leciona as disciplinas ofertadas por meio de turmas.
16. SemestreLetivo: Representa um semestre letivo. Por exemplo, 2023.1 ou 2023.2.
17. Turma: Representa uma turma existente em algum semestre letivo para realizar as atividades de determinada disciplina. Uma turma é a contrapartida temporal de uma (e apenas uma) disciplina da grade curricular.

BIBLIOGRAFIA

BEZERRA, E. Princípios de análise e projetos de Sistema com UML. 3ª ed., Rio de Janeiro : Elsevier, 2015.

BOOCH, G., J. RUMBAUGH, and I. JACOBSEN. The Unified Modeling Language User Guide. 2d ed., Addison-Wesley, 2005.

PRESSMAN, R. S. Engenharia de Software: Uma abordagem profissional. 7ª edição. Dados eletrônicos. Porto Alegre: AMGH-McGrawHill, 2011.

SOMMERVILLE, I. Engenharia de Software. 10ª edição. São Paulo: Pearson Prentice Hall, 2018.

WAZLAWICK, R. S. Engenharia de software: conceitos e práticas. Rio de Janeiro: Elsevier, 2013.

Obrigado!

Engenharia de Software