



Análise orientada a objetos II

Análise orientada a objetos II

Polyanna Pacheco Gomes Fabris
Iolanda Cláudia Sanches Catarino

© 2017 por Editora e Distribuidora Educacional S.A.
Todos os direitos reservados. Nenhuma parte desta publicação poderá ser reproduzida ou transmitida de qualquer modo ou por qualquer outro meio, eletrônico ou mecânico, incluindo fotocópia, gravação ou qualquer outro tipo de sistema de armazenamento e transmissão de informação, sem prévia autorização, por escrito, da Editora e Distribuidora Educacional S.A.

Presidente

Rodrigo Galindo

Vice-Presidente Acadêmico de Graduação

Mário Ghio Júnior

Conselho Acadêmico

Alberto S. Santana

Ana Lucia Jankovic Barduchi

Camila Cardoso Rotella

Cristiane Lisandra Danna

Danielly Nunes Andrade Noé

Emanuel Santana

Grasiele Aparecida Lourenço

Lidiane Cristina Vivaldini Olo

Paulo Heraldo Costa do Valle

Thatiane Cristina dos Santos de Carvalho Ribeiro

Revisão Técnica

Juliana Schiavetto Dauricio

Editorial

Adilson Braga Fontes

André Augusto de Andrade Ramos

Cristiane Lisandra Danna

Diogo Ribeiro Garcia

Emanuel Santana

Erick Silva Griep

Lidiane Cristina Vivaldini Olo

Dados Internacionais de Catalogação na Publicação (CIP)

F128a Fabris, Polyanna Pacheco Gomes
Análise orientada a objetos II / Polyanna Pacheco Gomes
Fabris, Iolanda Claudia Sanches Catarino. – Londrina :
Editora e Distribuidora Educacional S.A., 2017.
184 p.

ISBN 978-85-8482-910-1

1. Programação orientada a objeto. 2. Análise de sistemas. I. Catarino, Iolanda Claudia Sanches. II. Título.

CDD 005.1

2017

Editora e Distribuidora Educacional S.A.
Avenida Paris, 675 – Parque Residencial João Piza
CEP: 86041-100 – Londrina – PR
e-mail: editora.educacional@kroton.com.br
Homepage: <http://www.kroton.com.br/>

Sumário

Unidade 1 Diagramas estruturais da UML	9
Seção 1 - Visão geral da UML: conceitos, características e organização dos diagramas estruturais	13
1.1 Estudo de caso: controle de Eventos de Extensão universitária	14
Seção 2 - Diagrama de pacotes: conceitos, componentes, notação e construção	17
2.1 Como construir diagrama de pacote	18
2.1.1 Definição	18
2.1.2 Notação	18
2.2 Analisando alguns exemplos de diagrama de pacote do nosso estudo de caso	20
Seção 3 - Diagrama de classes: conceitos, componentes, notação e construção	23
3.1 Como construir um diagrama de classe	24
3.1.1 Conceituação geral	24
3.1.2 Afinal, o que são classes e como são feitos os relacionamentos entre elas?	25
3.2 Notação	28
3.3 Analisando alguns exemplos de diagrama de classe do nosso estudo de caso	29
Seção 4 - Diagrama de objetos: conceitos, componentes, notação e construção	35
4.1 Conceituação geral	35
4.2 Notação usada para construir um diagrama de objetos	35
4.2 Analisando alguns exemplos de diagrama de objetos do nosso estudo de caso	36
Seção 5 - Diagrama de estruturas compostas: conceitos, componentes, notação e construção	39
5.1 Organização de um diagrama de estrutura composta	39
5.1.1 Colaboração	40
5.1.2 Ocorrência de colaboração	40
5.1.3 Portas	40
5.1.4 Propriedades	40
5.1.5 Analisando alguns exemplos de diagrama de estrutura composta do nosso estudo de caso	41
Unidade 2 Diagramas comportamentais da UML	47
Seção 1 - Diagrama de Casos de Uso: conceitos, componentes, notação e construção	51
1.1 Como construir Diagrama de Casos de Uso	52
1.1.1 Definição	52
1.1.2 Notação	53
1.1.3 Exemplos do Diagrama de Casos de Uso	55

Seção 2 - Diagrama de Atividades: conceitos, componentes, notação e construção	65
2.1 Como construir Diagrama de Atividades	66
2.1.1 Definição	66
2.1.2 Notação	67
2.1.3 Exemplos do Diagrama de Atividades	69
Seção 3 - Diagrama de Máquina de Estados: conceitos, componentes, notação e construção	75
3.1 Como construir Diagrama de Máquina de Estados	76
3.1.1 Definição	76
3.1.2 Notação	77
3.1.3 Exemplos do Diagrama de Máquina de Estados	79
Unidade 3 Diagramas de interação da UML	91
Seção 1 - Introdução aos diagramas de interação UML	95
Seção 2 - Diagrama de Sequência: conceitos, componentes, notação e construção	101
Seção 3 - Diagrama de Colaboração: conceitos, componentes, notação e construção	111
Unidade 4 Consistência e interação entre os diagramas da UML	125
Seção 1 - Compreendendo a transição da modelagem entre as atividades	131
1.1 Processo Unificado	132
1.2 Detalhamento do Processo de Desenvolvimento de Software	135
Seção 2 - Modelagem da atividade de Análise de Requisitos com UML	143
2.1 Estudo de Caso – Agência de Estágios	144
2.2 Atividade de Análise de Requisitos	146
Seção 3 - Modelagem da atividade de Análise com UML	153
3.1 Diagrama de Casos de Uso	154
3.2 Diagrama de Classes	159
3.3 Diagrama de Objetos	164
3.4 Diagrama de Estruturas Compostas	165
3.5 Diagrama de Atividades	166
3.6 Diagrama de sequência	168
3.7 Diagrama de Máquina de Estados	171

Apresentação

Este livro contempla o estudo das principais técnicas de modelagem da *Unified Modeling Language* (UML) para documentar as atividades ou fases de Análise e Projeto de Modelos de Processos de Software, conforme o Paradigma Orientado a Objetos (POO).

Para facilitar e orientar a produção de software com qualidade e uma boa relação custo-benefício é essencial que o responsável pelo projeto de software defina a metodologia de desenvolvimento, que abrange o conjunto de métodos, técnicas, ferramentas, tecnologias e procedimentos. O método proporciona os detalhes de “como fazer” para construir o software, orientando-o e conduzindo-o através de suas técnicas de modelagem aplicadas às etapas do processo de desenvolvimento.

Na literatura, há vários métodos de desenvolvimento de software aplicados ao POO, destacando-se, o método de Coad & Yourdon (*Object-Oriented Analysis and Object-Oriented Design* - OOAD), de Grady Booch (*Booch Method*), Ivar Jacobson (*Object-Oriented Software Engineering* - OOSE) e James Rumbaugh (*Object Modeling Technique* - OMT). Os métodos abrangem as etapas (atividades ou fases) de desenvolvimento do software, sendo que para as etapas iniciais do processo de desenvolvimento – Análise de Requisitos, Análise e Projeto, os métodos apresentam e orientam a modelagem do software, através das técnicas de modelagem, que geralmente são especificadas no formato de diagramas. Um conjunto de diagramas constitui um modelo do sistema.

No início da década de 1990, os pesquisadores Grady Booch, Ivar Jacobson e James Rumbaugh uniram as melhores práticas de suas técnicas de modelagem e, com demais contribuições de alguns autores, construíram um padrão de referência para modelagem orientada a objetos, denominada Linguagem de Modelagem Unificada (UML). O lançamento oficial da UML como um padrão mundial foi em 1997, a partir do apoio de grandes empresas e da aprovação da *Object Management Group* (OMG - Agência Americana de Padrões). Desde o seu lançamento, várias atualizações foram feitas, sendo que a versão 2.0 foi apresentada em 2005 e atualmente a versão 2.5 foi lançada em 2015.

Algumas características da UML devem ser destacadas, entre elas: A UML não é uma metodologia, pois não diz quem deve fazer o quê, quando e como; a UML não está vinculada exclusivamente a um modelo de processo de desenvolvimento de software; a UML não está presa a uma etapa (fase ou atividade) do processo de desenvolvimento de software, como Análise de Requisitos, Análise, Projeto etc.; e a UML não é aplicada especificamente a uma linguagem de programação.

A UML é uma linguagem visual para modelar e especificar sistemas orientados a objetos, apoiando-se no desenvolvimento incremental, através de modelos que podem evoluir com a inclusão de novos detalhes e também sendo extensível, permitindo que a UML seja adaptada às peculiaridades de cada software. Cada elemento gráfico da UML possui uma sintaxe (forma predeterminada de desenhar o elemento) e uma semântica (significado e com o objetivo que ele dever ser utilizado).

A partir da versão 2.0, a UML contempla treze técnicas de modelagem gráficas e uma técnica de modelagem descritiva, privilegiando a descrição de um sistema segundo três perspectivas: estrutural, comportamental e de interação.

A perspectiva estrutural refere-se aos aspectos estáticos do software e das classes. Os diagramas são: Diagrama de Classes, Diagrama de Objetos, Diagrama de Componentes, Diagrama de Pacotes, Diagrama de Estrutura Composta e o Diagrama da Implantação.

A perspectiva comportamental refere-se à descrição e modelagem do aspecto dinâmico do software. Os diagramas são: Diagrama de Caso de Uso e sua documentação, Diagrama de Atividades e o Diagrama de Máquina de Estados.

A perspectiva de interação representa um subgrupo dos diagramas comportamentais, que mostram a interação entre os elementos da modelagem do software. Os diagramas são: Diagrama de Sequência, Diagrama de Comunicação, Diagrama de Interação Geral e o Diagrama de Tempo.

Nesse contexto, este livro está organizado em quatro unidades. A Unidade 1 descreve uma visão geral sobre os conceitos, características e organização dos diagramas da UML, e os conceitos, componentes, notação e construção dos Diagramas Estruturais

com exemplos baseados em um estudo de caso.

A Unidade 2 apresenta os conceitos, componentes, notação e construção dos Diagramas Comportamentais. A Unidade 3 contempla os conceitos, componentes, notação e construção dos principais Diagramas de Interação - Diagrama de Sequência e o Diagrama de Comunicação.

A Unidade 4 aborda a integração e consistência entre os principais Diagramas Estruturais, Comportamentais e de Interação, demonstrando a modelagem da atividade de Análise de um segundo estudo de caso, conforme as características do Processo Unificado.

O ciclo de vida do Processo Unificado é iterativo e incremental, abrangendo duas dimensões: dimensão temporal e dimensão de atividades (fluxos de trabalho). Na dimensão temporal, o processo é estruturado em quatro fases sucessivas, sendo elas: Concepção, Elaboração, Construção e Transição, e cada fase integra um conjunto de atividades interativas: Requisitos, Análise e Projeto, Implementação e Teste.

Diagramas estruturais da UML

Polyanna Pacheco Gomes Fabris

Objetivos de aprendizagem

- Entender o que são diagramas estruturais e suas características.
- Estudar os conceitos, as características e a organização dos diagramas de pacotes, objetos, classes e estruturas compostas.

Seção 1 | Visão geral da UML: conceitos, características e organização dos diagramas estruturais

Nesta seção, será apresentado um breve resumo sobre a UML e seus diagramas estruturais. Também, é descrito um estudo de caso proposto para a presente unidade.

Seção 2 | Diagrama de pacotes: conceitos, componentes, notação e construção

Nesta seção, apresentamos a definição do que é um diagrama de pacotes, como os elementos desse diagrama são organizados; também, são apresentados os detalhes que compõem a sua estrutura. Complementariamente, mostramos as notações usadas para representar graficamente o diagrama e usamos alguns exemplos a partir do estudo de caso proposto para esta unidade. Por fim, deixamos alguns links disponíveis para o aluno, a fim de ampliar o seu conhecimento no diagrama.

Seção 3 | Diagrama de classes: conceitos, componentes, notação e construção

Nesta seção, é possível obter o entendimento sobre o diagrama de classes e a sua importância para a atividade de análise, vemos que esse é o diagrama que representa o coração do processo de modelagem de

objetos. Apresentamos os conceitos relacionados à sua construção, bem como as estruturas, os relacionamentos e comportamentos. Um exemplo de diagrama de classe é ilustrado a partir do estudo de caso proposto para esta unidade. Ao final da seção, alguns links estão disponíveis para consulta e ampliação do conhecimento sobre o diagrama.

Seção 4 | Diagrama de objetos: conceitos, componentes, notação e construção

Nesta seção, nós explanamos o tema diagrama de objetos, suas características e como modelá-lo. Também, apresentamos exemplos de diagramas de objetos, construídos a partir do estudo de caso proposto para esta unidade.

Seção 5 | Diagrama de estruturas compostas: conceitos, componentes, notação e construção

Nesta seção, detalhamos sobre o tema diagrama de estrutura composta, suas características e como elas são modeladas. Alguns exemplos são apresentados e disponibilizamos alguns links.

Introdução à unidade

Caro(a) aluno(a), nesta unidade, apresentaremos os conceitos relacionados aos diagramas estruturais, quais são esses diagramas e quais são as características e os aspectos envolvidos na sua construção. Também, apresentamos uma visão geral da UML, focando nessa categoria de diagrama UML, demonstrando por meio de exemplos baseados em um estudo de caso. Essa visão está detalhada na Seção 1.

Os diagramas estruturais têm como objetivo a visualização, especificação, construção e documentação de aspectos estáticos do software. Dentre os diagramas que compõem essa categoria, elencamos os quatro mais importantes: diagrama de pacote, diagrama de classe, diagrama de objetos e estruturas compostas.

Um diagrama de pacote contém elementos sintáticos usados na modelagem orientada a objetos de um software e o relacionamento entre os elementos. Mais detalhes sobre esse diagrama estão na Seção 2.

Um diagrama de classe possibilita a visualização de um conjunto de classes, com o detalhamento de seus atributos, operações e relacionamentos. Mais detalhes sobre esse diagrama estão na Seção 3.

O diagrama de pacote é definido como uma variação do diagrama de classes, ao invés delas são mostradas as instâncias e ligações entre elas, descrevendo os objetos e seus relacionamentos. Mais detalhes sobre esse diagrama estão na Seção 4.

Por fim, temos o diagrama de estruturas compostas. Esse diagrama provê formas de definição das estruturas de elementos e mantém foco em seus detalhes, tanto na construção quanto nos relacionamentos internos. Mais detalhes sobre esse diagrama estão na Seção 5.

Cada diagrama acima citado abordará seus conceitos, componentes, notação, como construí-los e exemplificações usando o estudo de caso proposto pela instituição.

Seção 1

Visão geral da UML: conceitos, características e organização dos diagramas estruturais

Introdução à seção

Caro(a) aluno(a), esta é a primeira seção da presente unidade. Aqui, você será apresentado a uma visão geral sobre a UML, uma breve explanação sobre a categoria dos diagramas estruturais e um estudo de caso que é proposto para exemplificar a prática do uso da UML. Que o ânimo esteja com você. Vamos lá!

A UML (vem do inglês *Unified Modeling Language*) representa uma linguagem de modelagem unificada para a especificação, visualização, construção e documentação de características e aspectos que envolvem o software. Por especificação, pode-se entender como a clarificação do contexto e das características para que o software seja desenvolvido e implantado. No desenvolvimento, é necessário um padrão para que cada integrante da equipe tenha o mesmo entendimento e visualização da representação gráfica, bem como os aspectos representados. Durante a construção, a UML permite que seus modelos sejam mapeados para uma linguagem de programação específica, e vice-versa. Em relação à documentação, a UML suporta a criação e documentação de toda a análise gerada para o software desenvolvido.

Considerando os objetivos da linguagem e o contexto a ser aplicado, existem diferentes diagramas que a compõe e estes podem ser agrupados em categorias. De acordo com Melo (2002), essas categorias são descritas da seguinte maneira:

- Diagramas estruturais: responsáveis pelo tratamento de aspectos estruturais, sob o ponto de vista de um sistema ou de suas classes. Seu objetivo é a visualização, especificação, construção e documentação de aspectos estáticos do software;
- Diagramas comportamentais: responsáveis pela descrição e modelagem do aspecto dinâmico do software;

- Diagramas de interação: representa o subgrupo dos diagramas comportamentais, usamos para mostrar a interação entre elementos de uma modelagem e da aplicação.

Vimos acima que existem algumas categorias de diagramas UML, porém, a presente unidade focará apenas em uma categoria, **os diagramas estruturais**. Os diagramas que compõem essa categoria são: diagrama de classe, diagrama de objeto, diagrama de pacote e diagrama de estruturas compostas. Para gerar esses diagramas UML, existem diversas ferramentas disponíveis no mercado, nós selecionamos e usaremos a ferramenta **Visual Paradigm**. Para complementar o seu aprendizado, utilizaremos o estudo de caso a seguir, que orientará os exemplos citados na unidade: no Tópico 2.2, mostramos um exemplo de diagrama de pacote; no Tópico 3.3, temos o exemplo do diagrama de classe; já no Tópico 4.2, representamos um diagrama de objetos; por fim, no Tópico 5.1.5, o diagrama de estruturas compostas é representado. Agora, acompanhe o estudo de caso.

1.1 Estudo de caso: controle de Eventos de Extensão universitária

Leia com atenção o estudo de caso proposto, nossos exemplos de diagramas utilizarão dados aqui relacionados:

A Coordenadoria de Extensão de uma universidade precisa de um Sistema de Informação para controlar e gerenciar as inscrições on-line dos eventos de extensão que os cursos de graduação oferecem.

Os eventos de extensão que um curso de graduação promove são classificados em três tipos: Curso, Palestra ou Encontro (que pode ser uma feira, um workshop, um campeonato etc.). Todo evento precisa ser cadastrado com: nome do evento, professor responsável, carga horária, objetivo, público-alvo, período de inscrição, data do evento, horário, local, número de vagas, tipo (interno, externo ou misto), situação (agendado, sendo realizado, concluído ou cancelado), valor para comunidade interna (aluno

e funcionário) e valor para visitante. O evento do tipo curso deve conter também o nome do ministrante, que pode ser ou não um professor da própria instituição e número mínimo de inscritos para o curso ser realizado. O evento do tipo palestra deve conter também o nome do ministrante, que pode ser ou não um professor da própria instituição e síntese do currículo do palestrante. O evento do tipo encontro deve conter também uma descrição do encontro.

Um curso de graduação deve ser cadastrado com nome, carga horária total, turno, área e campus. Um curso de graduação pode ofertar nenhum ou vários eventos, e o mesmo evento deve ser ofertado por, no mínimo, um curso de graduação ou vários.

Uma pessoa que participa de um evento de extensão pode ser um aluno, funcionário da instituição (professor ou administrativo) ou uma pessoa da comunidade externa, denominada visitante. Todos os participantes devem ser cadastrados com seus dados pessoais (nome, endereço completo, data de nascimento, CPF, RG, sexo, filiação, telefone residencial, celular e endereço eletrônico). Um aluno deve estar vinculado a um curso de graduação. Um visitante deve conter também o nome de uma empresa que trabalha e/ou o nome de uma instituição que está matriculado. Um aluno pode ser um funcionário da instituição. Uma pessoa pode participar de muitos eventos, mediante a efetivação (pagamento) da inscrição.

A inscrição on-line de um participante para um evento deve conter um número, data da inscrição, valor da inscrição e situação (realizada, efetivada ou cancelada).

Baseando-se neste estudo de caso, propomos que você faça a construção de alguns exemplos de diagramas estruturais e compartilhe com seu professor. O que acha?

Para saber mais

GUEDES, G. T. A. **UML 2: guia prático**. São Paulo: Novatec, 2007.

MELO, A. C. **Desenvolvendo aplicações com UML 2.0 do conceitual à implementação**. 2. ed. Rio de Janeiro: Brasport, 2002.



Questão para reflexão

Analise a relação entre as categorias dos diagramas UML.
Compartilhe sua percepção sobre esta questão com o professor.

Atividades de aprendizagem

1. Observe as alternativas a seguir e assinale a que representa um exemplo de diagrama estrutural da UML:

- a) Diagrama de sequência.
- b) Diagrama de atividades.
- c) Diagrama de casos de uso.
- d) Diagrama de pacotes.
- e) Diagrama de comunicação.

2. Assinale a alternativa que representa as categorias dos diagramas UML:

- a) Estrutural, comportamental e integração.
- b) Estrutural, comportamental e interação.
- c) Estrutural, estratégico e integração.
- d) Estrutural, comportamental e estratégico.
- e) Nenhuma das alternativas anteriores.

Fique ligado

Nesta seção, apresentamos:

- Características da UML;
- As categorias de diagramas UML;
- Detalhes sobre diagramas estruturais: pacote, objeto, classes e estruturas compostas.

Como foi o estudo desta seção? Que tal compartilhar sua opinião com o professor?

Seção 2

Diagrama de pacotes: conceitos, componentes, notação e construção

Introdução à seção

Caro(a) aluno(a), nesta seção, apresentaremos a contextualização sobre os diagramas de pacote, as notações utilizadas para compor o diagrama e como construí-lo.

Nos mais diversos cenários, existe a necessidade de organizar elementos, normalmente, eles são organizados em coleções. Por exemplo, livros com temas relacionados são agrupados com outros similares em uma biblioteca ou livraria. No caso de uma loja de departamentos, esta possui um número de departamentos, cada departamento contém itens relacionados (DATHAN; RAMNATH, 2015).

Quando falamos em análise orientada a objetos, esse conceito não é diferente. À medida que o software se torna mais complexo, é importante garantir que os módulos estejam devidamente organizados. Esta organização será por meio de coleções exclusivas (DATHAN; RAMNATH, 2015).

Para Bezerra (2015), um pacote pode ser definido como um mecanismo de agrupamento acentuado pela UML. Por meio dos pacotes, elementos com a mesma semântica podem ser relacionados e aglutinados em um diagrama.

Assim, os diagramas de pacote são criados e customizados para ilustrar diferentes aspectos de estrutura de sistemas ou para representar a organização dos vários tipos de modelos utilizados para descrever um sistema, contendo modelos de casos de uso, diagramas de classes, diagramas de implantação etc. Dentre as características dos diagramas de pacotes estão, segundo Pender (2003):

- identificar os dados e as dependências funcionais entre partes de um sistema;
- identificar o particionamento de um sistema em subsistemas;
- identificar as fases em um projeto;
- utilitários separados de componentes específicos do sistema;
- identificar as camadas isoladas de uma arquitetura.

Para construir um diagrama de pacote, faz-se necessário o entendimento de todos os elementos que o compõe e qual o objetivo de cada um. Esses elementos são apresentados no próximo tópico.

2.1 Como construir diagrama de pacote

2.1.1 Definição

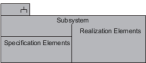







Por definição, nós podemos entender que um diagrama de pacote mostra o agrupamento e a organização de um modelo de elementos, mostrando a estrutura e a dependência entre sistemas, subsistemas ou módulos (DATHAN; RAMNATH, 2015).



A notação para um pacote é uma pasta com uma aba, cada pacote terá um nome. As ligações entre os pacotes representam relações de dependências, essas dependências são representadas pelos estereótipos (BEZERRA, 2015). No próximo tópico, exploraremos os elementos que compõem um diagrama de pacotes.

2.1.2 Notação

Para desenhar um diagrama de pacote, utilize as notações que mostramos no Quadro 1.1 a seguir. Elas estão ilustradas utilizando a ferramenta CASE Visual Paradigm.

Quadro 1.1 | Elementos do diagrama de pacote

Notação	Nome	Definição
	Subsystem	Um subsistema representa o limite do subsistema físico.
	Package	Um pacote é usado para agrupar elementos.
	Merge	Uma mesclagem de pacote é uma relação direta entre dois pacotes, que indica que o conteúdo dos dois pacotes deve ser combinado. Assemelha-se à generalização, adicionando as características do elemento-alvo para suas próprias características, resultando em um elemento que combina as características dos dois.
	Generalization	Uma generalização é uma relação taxonômica entre um classificador mais geral e um mais específico. Cada instância do classificador específico é também uma instância indireta do classificador geral. Deste modo, o classificador específico herdará as características do elemento que atua como "pai".
	Constraint	Uma condição ou restrição expressa em linguagem texto natural, ou em linguagem legível por máquina, com o propósito de declarar parte da semântica de um elemento.
	Realization	Realização é uma relação de abstração especializada entre dois conjuntos de elementos do modelo, um representando uma especificação e o outro representando uma implementação do cliente. A Realização pode ser usada para modelar o refinamento gradual, otimizações, transformações, modelos, síntese de modelos, composição de estruturas etc.
	Note	Uma nota permite que sejam anexadas várias observações aos elementos. Embora uma nota não possua força semântica, ela pode conter informações úteis para um modelador.
	Import	Uma importação de pacote é definida como uma relação direcionada que identifica um pacote cujos membros devem ser importados.

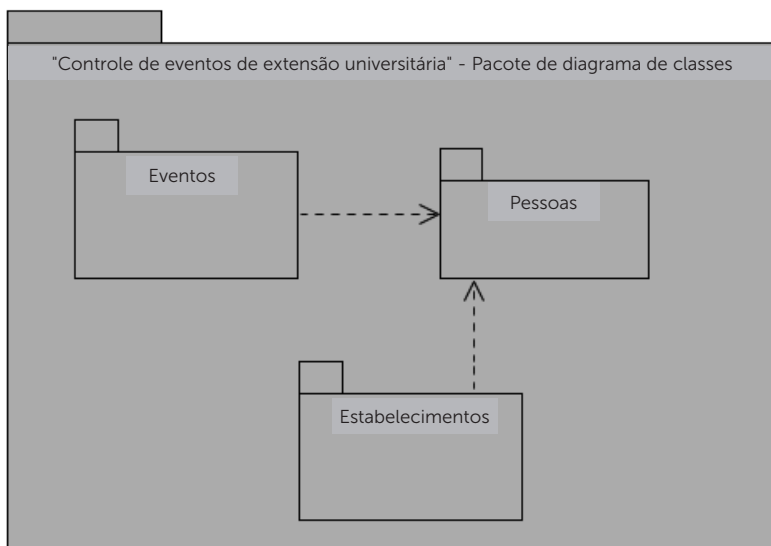
	Dependency	Estabelece um relacionamento de dependência de um ou mais elementos, requerendo características do elemento fornecedor.
	Access	O elemento é definido como a relação entre um pacote e o elemento "empacotado". O elemento "empacotado" é adicionado ao pacote importador.

Fonte: adaptado de Visual (2016).

2.2 Analisando alguns exemplos de diagrama de pacote do nosso estudo de caso

Nos tópicos anteriores, foram apresentados os conceitos sobre o diagrama de pacotes, qual a sua estrutura e qual notação usada para gerá-lo. Nesta seção, mostraremos um exemplo de diagrama de pacotes baseado no estudo de caso proposto na Seção 1, confira na Figura 1.1 a seguir.

Figura 1.1 | Exemplo de diagrama de pacotes



Fonte: elaborada pelo autor.

A Figura 1.1 mostra a organização do diagrama de pacote do estudo de caso “Controle de Eventos de Extensão Universitária”. Para construí-lo, consideramos que existia a necessidade de simplificar a modelagem UML. Foram considerados três pacotes principais: Pessoas (Funcionário, Aluno, Visitante, Palestrante e outros), Eventos (Evento, Inscrição, Curso Graduação e outros) e Estabelecimentos (Estabelecimentos, Instituição de Ensino, Empresa e outros). Assume-se a existência de dependências entre os pacotes Estabelecimentos e Eventos para o pacote Pessoas. Para entendimento do exemplo da Figura 1.1, consulte o Tópico 2.1.2.

Chegamos ao final da Seção 2.2, aqui mostramos as características e os exemplos de um diagrama de pacotes.



Questão para reflexão

De que forma os diagramas de pacote contribuem para o entendimento e a organização de um projeto de desenvolvimento de sistemas? Compartilhe a análise sobre esta questão com o seu professor.



Para saber mais

Livros

LARMAN, C. **Utilizando UML e padrões**. Brasil: Bookman Companhia, 2007.

CHONOLLES, M. J.; SCHARDT, J. A. **UML 2 for dummies**. New York: Wiley Pub, 2003.

Fique ligado

Nesta seção, foram apresentados os detalhes, notação e exemplo de um diagrama de pacote.

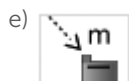
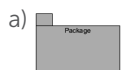
Como foi seu estudo desta seção? Compartilhe com o seu professor!

Atividades de aprendizagem

1. Analise as características a seguir e assinale a alternativa referente ao diagrama de pacotes:

- a) Organizar os casos de uso utilizando cenários.
- b) Organizar apenas diagramas de classe.
- c) Organizar apenas diagramas de sequência.
- d) Substituir os demais diagramas UML.
- e) Agrupar outros elementos do modelo, tais como as classes ou outros pacotes.

2. A notação usada para representar graficamente um pacote é:



Seção 3

Diagrama de classes: conceitos, componentes, notação e construção

Introdução à seção

De acordo com Pender (2003), o diagrama de classe é o coração do processo de modelagem de objetos. Ele representa a definição dos recursos essenciais para a operação do sistema. Por recursos, entende-se que representam pessoas, materiais, informações e comportamentos. Um diagrama de classe será composto por estruturas, relacionamentos e comportamentos.

Para Bezerra (2015), um diagrama de classe é usado na construção do modelo de classes, começando no nível de análise e estendendo até o nível de especificação.

As características do diagrama de classe são, segundo Pender (2003):

- definir os recursos essenciais de um sistema;
- definir os relacionamentos entre os recursos;
- gerar código;
- fornecer foco para todos os outros diagramas.

Conforme descrito por Guedes (2007), um diagrama de classes ainda poderá ser usado na modelagem do modelo lógico de um banco de dados, assemelhando-se ao Diagrama de Entidade-Relacionamento (diagrama usado em banco de dados). Contudo, é necessário reforçar que o diagrama de classes não é limitado a essa finalidade.

Nas próximas seções, apresentaremos os aspectos e as características que envolvem a construção de um diagrama de classes.

3.1 Como construir um diagrama de classe

3.1.1 Conceituação geral

Nesta seção, apresentaremos conceitos gerais que fazem parte da construção de um diagrama de classes, fornecendo informações complementares ao diagrama e permitindo que a análise seja mais completa. Dentre os conceitos gerais, restringindo apenas ao diagrama tratado aqui, esta seção abordará: visibilidade, multiplicidade, escopo, estereótipos, notas e restrições.

O primeiro conceito é a **visibilidade**. Identifica-se por quem uma propriedade, sendo ela atributo ou operação, será usada (MELO, 2002). Essa propriedade é definida por palavras-chave ou ícones. A seguir apresentamos uma breve descrição (MELO, 2002):

- **+** ou **public** (público): define que a propriedade pode ser vista e utilizada na classe para a qual foi declarada e em qualquer outro elemento externo;
- **#** ou **protected** (protegido): define que a propriedade pode ser vista e utilizada apenas na classe para a qual foi declarada e por classes descendentes;
- **-** ou **private** (privado): define que a propriedade pode ser vista e utilizada somente na classe em que foi declarada;
- **-** ou **package** (pacote): define que a propriedade pode ser vista e utilizada pelos elementos que façam parte do mesmo pacote.

Essa propriedade serve para a implementação do encapsulamento da estrutura interna da classe, sendo que somente as propriedades (realmente necessárias) ao exterior da classe devem ter a definição de visibilidade igual a **public** (BEZERRA, 2015).

O segundo conceito é a **multiplicidade**. Representa a quantidade de instâncias de um relacionamento (MELO, 2002). Provavelmente, trata-se da propriedade mais importante de uma associação (BEZERRA, 2015). A multiplicidade é representada por um subconjunto de um conjunto de números inteiros não negativos, numa sequência de intervalos inteiros e que são

separados por uma vírgula, representado da seguinte forma: limite-inferior...limite-superior (MELO, 2002).

O terceiro conceito é o **escopo**. Quando falamos de escopo, podemos tratá-lo de duas formas: escopo de instância e escopo de classe, em que, o primeiro define que cada objeto terá o seu próprio valor, e o último, valor comum de atributo para todos os objetos da classe (MELO, 2002).

O quarto conceito usado é o **estereótipo**. Podemos entendê-lo como mecanismo de extensibilidade que representa uma subclasse de um elemento que já existe com o mesmo formato, mas com objetivos diferentes e bem definidos. A representação gráfica desse elemento é "<< >>" (MELO, 2002), conhecido como *guillemets*.

O último conceito usado é **notas**. Representação textual, descrevendo o modelo com diversos tipos de informações (comentários, etiquetas, restrições e corpos de métodos) (MELO, 2002).

3.1.2. Afinal, o que são classes e como são feitos os relacionamentos entre elas?

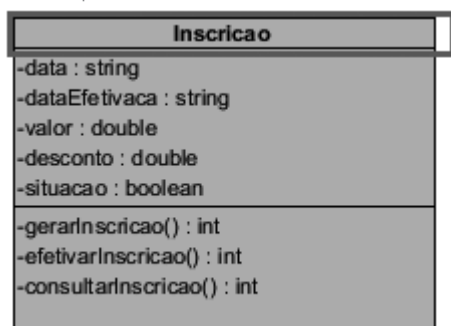
Conforme definido por Bezerra (2015), uma classe é ilustrada por uma "caixa" com três compartimentos: no primeiro, apresenta-se o nome da classe; no segundo, os atributos são declarados; por último, as operações são declaradas pelo objeto. Cada um desses compartimentos é separado por linhas horizontais (MELO, 2002). O refinamento das informações depende da importância para o projeto (MELO, 2002).

A seguir, apresentamos um breve detalhamento sobre cada compartimento de uma classe:

a. Compartimento do nome da classe: apresenta o nome da classe e outras propriedades, as quais são divididas em três seções: estereótipos, escritos em estilo normal dentro de *guillemets* (<<>>). Essa seção está localizada acima do nome da classe. Na sequência, está o nome da classe, com formato centralizado e negrito. Esta é a única seção obrigatória. A última seção é uma lista de *strings*, localizados abaixo do nome da classe, entre chaves (MELO, 2002).

Observe o primeiro compartimento destacado na Figura 1.2.

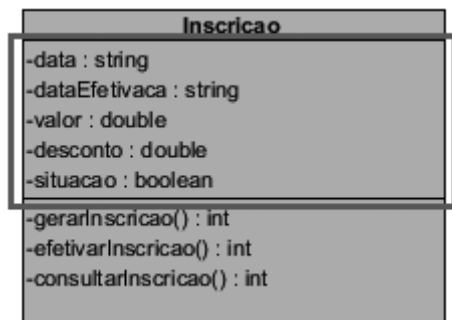
Figura 1.2 | Primeiro compartimento de uma classe



Fonte: elaborada pelo autor.

b. Atributos: representam a estrutura de uma classe. Os atributos não são definidos somente com seu nome e tipo de dado. Existem outras propriedades que podem ser acrescidas, tais como valor inicial, visibilidade e outras características. Na Figura 1.3, apresentamos um exemplo de atributos de uma classe. Foram criados atributos do tipo strings, double e boolean, porém existem outros tipos, tais como float, char ou long. A escolha de qual tipo usar dependerá do objetivo da modelagem a ser feita.

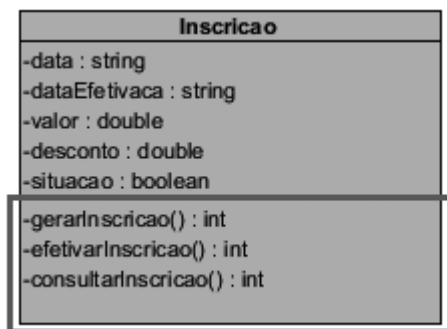
Figura 1.3 | Exemplos de atributos de uma classe



Fonte: elaborada pelo autor.

c. Operações ou métodos: uma operação pode ser descrita como uma função ou transformação que será aplicada a um ou mais objetos. A operação é composta por um nome, seus parâmetros e o tipo de retorno (caso exista). Na Figura 1.4, consta a exemplificação das operações da classe Inscrição.

Figura 1.4 | Exemplo de operações de uma classe



Fonte: elaborada pelo autor.

Em um diagrama de classes, existem colaborações entre as classes por meio de relacionamentos. Esses relacionamentos são: **associação**, **generalização**, **dependência**, **agregação** e **composição**. Cada um deles será descrito a seguir, e no Tópico 3.3 apresentamos exemplos.

O relacionamento de associação representa um relacionamento estrutural, indicando que um elemento contém ou está conectado a outro elemento. Ele é representado graficamente por meio de uma seta sólida que liga os elementos.

A generalização refere-se ao relacionamento entre um elemento geral e um específico. Esse relacionamento é representado por uma seta fechada e vazia, direcionada do elemento específico para o geral. Ao falarmos dos conceitos de subclasses e superclasses, trabalhamos um conceito chamado herança. Quando uma subclasse é derivada de uma superclasse, ela herda todos os seus atributos e métodos. A principal vantagem de uso da herança é que, ao declarar os atributos e métodos, e logo após realizar a derivação para uma subclasse, não será necessário redeclarar os atributos e métodos que foram definidos inicialmente, a subclasse herdará todas as características de sua superclasse de forma automática. Assim, permitirá que os códigos já prontos possam ser reutilizados e as futuras manutenções sejam facilitadas (GUEDES, 2007).

O conceito aplicado ao relacionamento do tipo dependência é definido como dependência entre os elementos do diagrama, qualquer mudança em um elemento, o outro elemento sofrerá

o impacto. Quando falamos em agregação, pode-se entender como um caso particular de associação, usado para representar um relacionamento do tipo “todo-parte”.

3.2 Notação

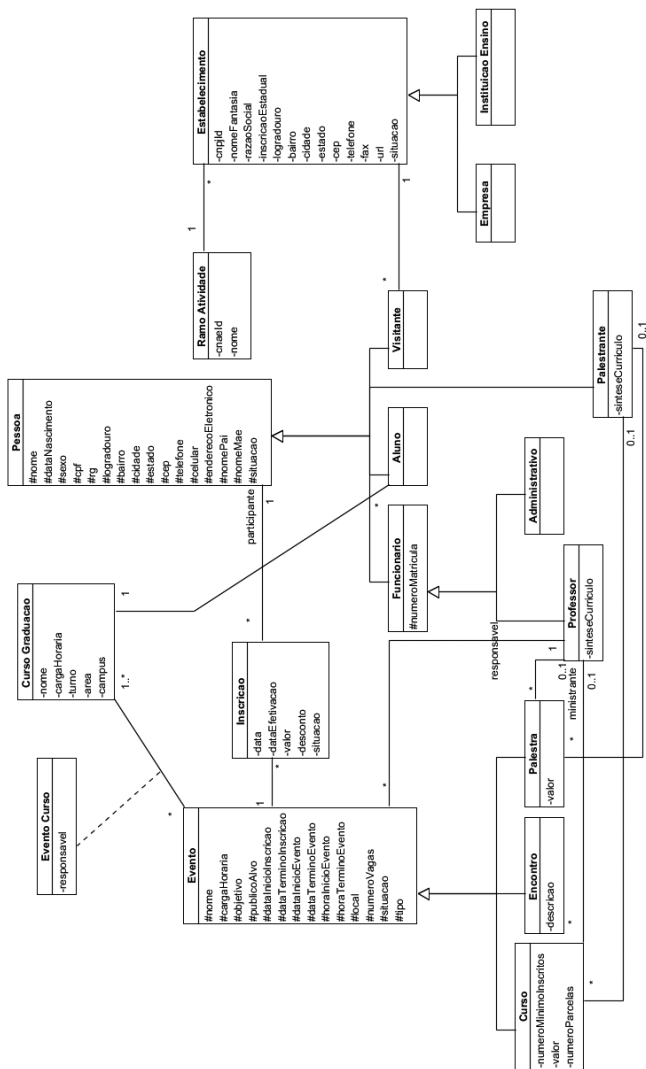
Para desenhar um diagrama de classes, utilize as notações que mostramos nas imagens a seguir. Elas estão ilustradas utilizando a ferramenta CASE Visual Paradigm.

Quadro 1.2 | Notações de um diagrama de classe

Notação	Nome	Definição
	Abstraction	Relaciona dois elementos ou conjuntos de elementos que representam níveis conceituais de abstração ou de diferentes pontos de vista.
	Aggregation	Relacionamento do todo com a parte ou vice-versa.
	Association Class	Relacionamento entre dois tipos de instâncias.
	Class	Representação gráfica da classe.
	Generalization	Relacionamento do tipo generalização entre classes.
	Usage	Relacionamento em que um elemento requer outro elemento.
	N-ary Association	Usado para representar relacionamentos complexos entre três ou mais elementos.
	Association	Associação simples entre elementos.
	Dependency	Relacionamento entre elementos no qual um é dependente do outro.
	Colaboration	Representação da colaboração entre entidades.
	Note	Adicionar informações relevantes para entendimento do diagrama.
	Constraint	Representação de uma condição ou restrição.

Fonte: elaborada pelo autor.

3.3 Analisando alguns exemplos de diagrama de classe do nosso estudo de caso



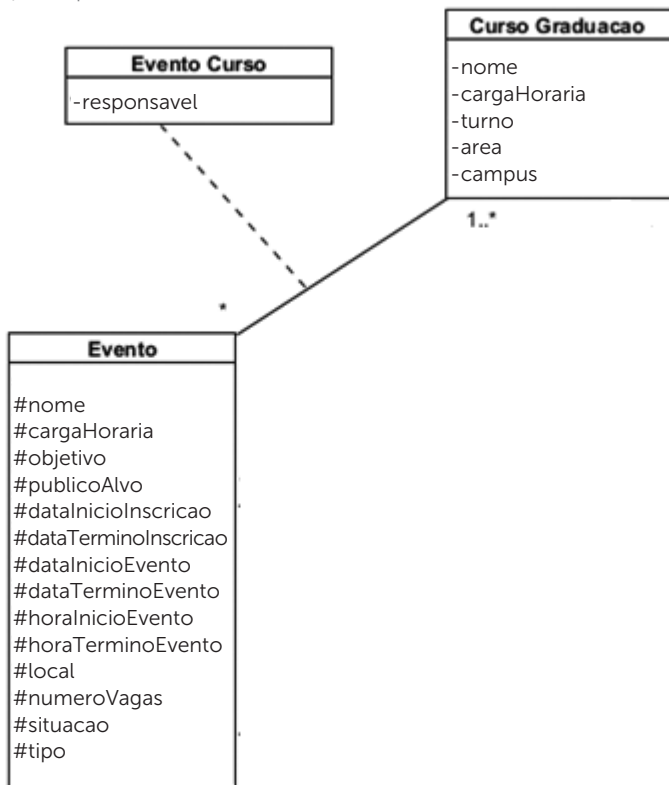
Ao observar a imagem acima, nós podemos identificar algumas características do diagrama de classe, tais como os listados a seguir.

Um dos primeiros pontos observados é em relação à multiplicidade, identificamos que o diagrama desenhado conta com alguns tipos de multiplicidade:

- [0...1]: indicando que os objetos das classes associadas não necessitam, obrigatoriamente, ser relacionados, porém, caso haja relacionamento, indica que somente uma instância da classe se relaciona com as instâncias da outra classe. Considerando nosso diagrama de classes acima, podemos ver essa multiplicidade na classe Palestrante;
- [*]: este asterisco refere-se a “muitos”, indicando que muitos objetos da classe estão envolvidos no relacionamento. No diagrama acima, vemos essa multiplicidade na classe Eventos e seu relacionamento com a classe Curso_Graduacao;
- [1...*]: existe, no mínimo, um objeto e, no máximo, muitos objetos envolvidos. Como ocorre no relacionamento entre as classes Evento e Inscrição.

O segundo ponto a ser observado é a existência de uma classe associativa “Evento_Curso” representada na imagem parcial do diagrama de classes. Confira:

Figura 1.6 | Exemplo de classe associativa



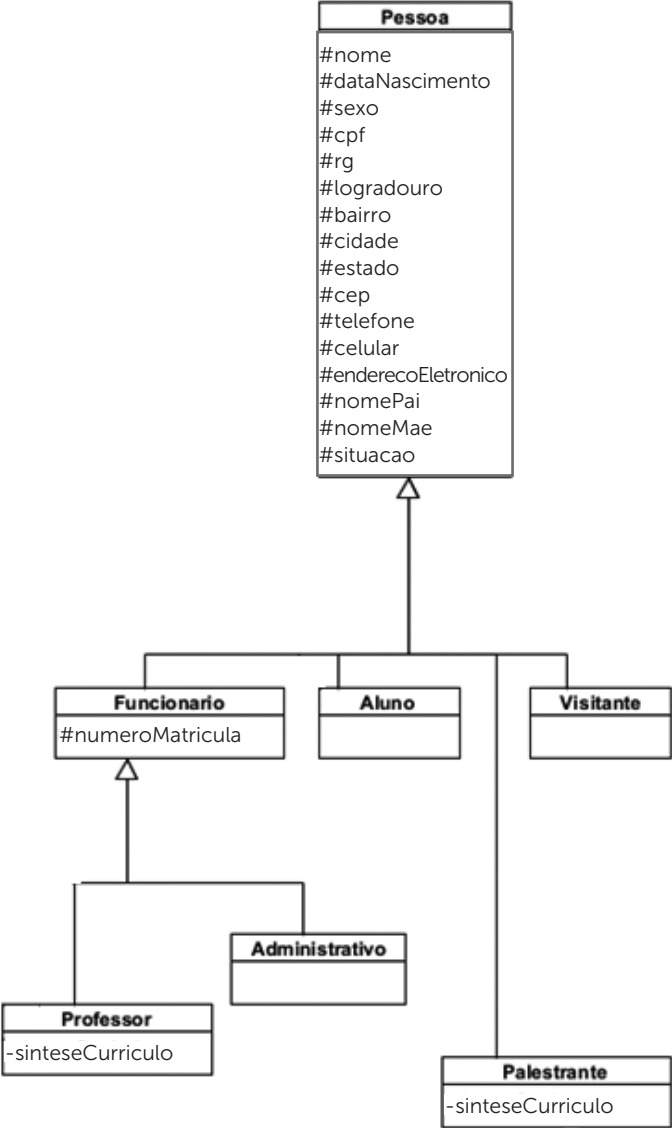
Fonte: elaborada pelo autor.

Nesta imagem, a instância da classe Evento pode se relacionar com muitas instâncias da classe Curso_Graduação e uma instância da classe Curso_Graduação pode se relacionar com muitas instâncias da classe Evento. Desse modo, como existe a multiplicidade de "muitos" nas extremidades de cada uma das classes, é necessário criar uma nova classe para armazenar os atributos decorrentes dessa associação. Os atributos considerados chave das classes Evento e Curso_Graduação serão transmitidos transparentemente pela associação.

Outro conceito no diagrama de classe mostrado na Figura 1.5 é a generalização. Para uma melhor visualização, particionamos a imagem para apresentar o relacionamento citado. Na imagem, é

possível notar a seta fechada, vazada e com corpo sólido, direcionando as classes mais específicas para as classes mais genéricas.

Figura 1.7 | Relacionamento generalização



Fonte: elaborada pelo autor.

Nesta imagem, no relacionamento entre as classes, podemos identificar que:

- A classe Pessoa é uma **superclasse** das **subclasses** Funcionário, Aluno, Palestrante e Visitante.
- E a classe Funcionário, mesmo sendo uma **subclasse**, é também uma **superclasse** do relacionamento entre as classes Professor e Administrativo, e estes, por sua vez, são **subclasses**.

As superclasses e subclasses também são conhecidas como classe pai e classe filha, respectivamente, nas quais as filhas herdarão as características da classe pai.

Aqui terminamos a Seção 3, as características dos diagramas de classe foram apresentadas, exemplificamos algumas delas usando o nosso estudo de caso da Seção 1 e, nos itens a seguir, complementamos seu estudo com algumas sugestões de tema.



Questão para reflexão

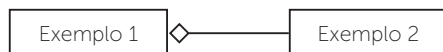
Qual é a importância dos relacionamentos em um diagrama de classes? Compartilhe sua opinião sobre este tema com o professor.

Atividades de aprendizagem

1. A alternativa que **não** relaciona os elementos que fazem parte da composição de um diagrama de classe é:

- a) Métodos, classes e relacionamentos de dependência.
- b) Classes, atributos e parâmetros de métodos.
- c) Classes, subclasses e superclasses.
- d) Visibilidade, relacionamento de dependência e estereótipos.
- e) Classes, representação do mapa de dados e escopo.

2. Analise a figura a seguir:



O tipo de relacionamento ilustrado é:

- a) Generalização.
- b) Dependência.

- c) Especialização.
- d) Agregação.
- e) Instanciação.

Fique ligado

Nesta seção, foram apresentados:

- detalhamento das características de um diagrama de classes;
- notação usada para construção de um diagrama de classes;
- exemplificação do diagrama de classe usando o estudo de caso proposto na Seção 1.

Como foi seu estudo desta seção? Compartilhe com o seu professor!

Seção 4

Diagrama de objetos: conceitos, componentes, notação e construção

Introdução à seção

4.1 Conceituação geral












De acordo com Bezerra (2015), o diagrama de objetos pode ser visto como uma instância do diagrama de classes, do mesmo modo que um objeto é uma instância de uma classe, exibindo uma “fotografia” do sistema em um dado momento. Essa “fotografia” apresentará as ligações e iterações entre os objetos.

A representação gráfica do objeto em um diagrama é um retângulo dividido em dois compartimentos. A parte superior do compartimento recebe a identificação do objeto, e a parte inferior receberá os atributos definidos na classe do objeto (BEZERRA, 2015). O objeto deve ser identificado de forma sublinhada e com letra inicial maiúscula.

4.2 Notação usada para construir um diagrama de objetos

Para desenhar um diagrama de objetos, utilize as notações que mostramos nas imagens a seguir. Elas estão ilustradas utilizando a ferramenta CASE Visual Paradigm.

Quadro 1.3 | Notação para os diagramas de objetos

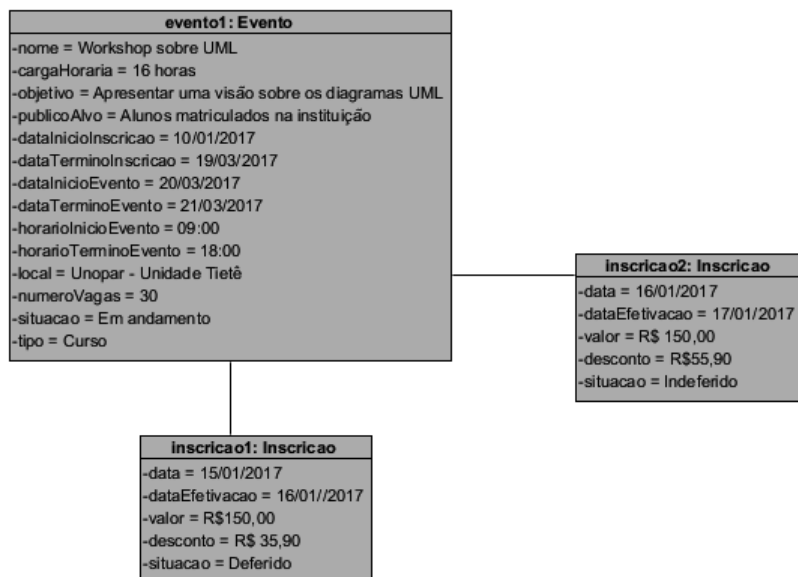
Notação	Nome	Definição
	Instance Specification	Representa as instâncias de objetos a serem modeladas.
	Associação	Associação entre as instâncias de objetos.
	Class	Representa um conjunto de objetos que serão modelados.
	Port	Representa pontos de interação distinta entre classificadores e ferramentas.
	Dependency	Relacionamento de dependência.
	Generalization	Relacionamento de generalização.
 Note  Anchor  Constraint  Containment	Note	Adicionar informações relevantes ao diagrama modelado.
	Constraint	Representa restrições ou condições para o diagrama modelado.

Fonte: Visual (2017).

4.2 Analisando alguns exemplos de diagrama de objetos do nosso estudo de caso

Para exemplificarmos um objeto, criamos a imagem a seguir. O objeto mapeado refere-se à inscrição 1, veja que o nome do evento está sublinhado no primeiro compartimento do retângulo, já na parte inferior temos os atributos desse objeto com seus respectivos valores. Ao analisarmos a modelagem do diagrama de classes, é possível concluir que podemos ter muitos objetos instanciados, ou seja, podem existir muitas inscrições para um determinado evento. Vale a pena ressaltar que Pessoa, Estabelecimento e Evento são classes abstratas, assim, não podem existir instâncias diretas delas.

Figura 1.8 | Exemplo de objeto



Fonte: elaborada pelo autor.



Questão para reflexão

Qual a contribuição dos diagramas de objetos para entendimento do software a ser desenvolvido? Compartilhe sua análise com o professor!

Atividades de aprendizagem

1. Ao criar um diagrama de objetos, devemos respeitar a notação representada na alternativa:
 - a) nome_objeto = nome_classe.
 - b) nome_objeto (nome_classe).
 - c) nome_objeto > nome_classe.
 - d) nome_objeto: nome_classe.
 - e) nome_objeto:> nome_classe.
2. Assinale a alternativa que descreve os compartimentos de um objeto:
 - a) Compartimentos para o nome e atributos.
 - b) Compartimentos para o nome, atributos e métodos.

- c) Compartimento para atributos e métodos.
- d) Compartimento para métodos apenas.
- e) Não existem compartimentos em um objeto.

Fique ligado

Esta seção abordou as características dos diagramas de objetos, apresentou sua notação para representação gráfica e expôs exemplo de diagrama de objetos usando como base o nosso estudo de caso da Seção 1. Como foi seu estudo desta seção? Que tal compartilhar sua opinião com o professor?

Seção 5

Diagrama de estruturas compostas: conceitos, componentes, notação e construção

Introdução à seção

O diagrama de estrutura composta representa a estrutura interna de classificadores estruturados por meio de peças, portas e conectores que definem a implementação, a fim de apresentar detalhes internos de um classificador e descrever comportamentos necessários para a execução. Suas características são similares a uma classe, porém, quando falamos do diagrama de estruturas compostas, nos referimos apenas a uma parte e não ao todo (representado pela classe) (IBM, 2017).

Conforme descrito por Guedes (2007), um diagrama de estrutura composta é utilizado para a modelagem de colaborações. Essa colaboração representa uma visão de um conjunto de entidades cooperativas interpretadas por instâncias que cooperam entre si, com o objetivo de executar procedimentos específicos. Estrutura é um termo referente à composição de elementos que se conectam, os quais representam instâncias executadas para atender determinado objetivo (GUEDES, 2007). Nos próximos tópicos, apresentaremos detalhes sobre a estrutura do diagrama de estruturas compostas.

5.1 Organização de um diagrama de estrutura composta

Já citamos acima que um diagrama de estrutura composta se assemelha ao diagrama de classe, entretanto, apresenta uma visão estática da estrutura de classes, objetivando a modelagem de colaborações (GUEDES, 2007). Nos tópicos a seguir, apresentamos um detalhamento sobre a estrutura do diagrama de estruturas compostas.

5.1.1 Colaboração

Guedes (2007) descreve que a colaboração é representada como um tipo de classificador e define um conjunto de entidades cooperativas que são interpretadas por instâncias, as quais representarão o papel de entidade, assim como um conjunto de conectores que realizam a comunicação entre os participantes.

O objetivo principal de uma colaboração é a explicação de como é o funcionamento de um sistema, assim, mostrando apenas aspectos essenciais. A colaboração é graficamente representada por meio de uma elipse tracejada com o seu descritivo (GUEDES, 2007).

5.1.2 Ocorrência de colaboração

Para Guedes (2007), a ocorrência é a aplicação do padrão descrito por uma colaboração a uma situação específica que envolve as classes ou instâncias executoras de papéis específicos da colaboração. A notação para representar uma ocorrência de colaboração é a mesma utilizada na denominação de um objeto: nome da ocorrência, na sequência dois-pontos (:) e o nome da colaboração.

5.1.3 Portas

Uma porta é uma característica estrutural de um classificador que representa um ponto de interação entre um classificador e seu ambiente ou classificador e sua parte interna. A porta é conectada às propriedades de um classificador por conectores para representar os serviços que o classificador requer dele. Para representá-la, usam-se quadrados sobrepostos à borda de um classificador ou de suas partes internas (GUEDES, 2007).

5.1.4 Propriedades

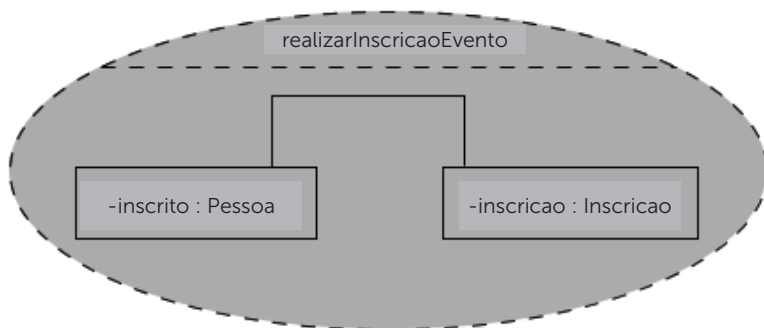
Pode-se entender que uma propriedade representa um conjunto de instâncias internas, que são incluídas pela instância de um classificador contêiner. A partir do momento que uma instância de um classificador é gerada, o conjunto de instâncias

correspondentes às suas propriedades pode ser criado. A propriedade específica que determinado conjunto de instâncias pode existir.

5.1.5 Analisando alguns exemplos de diagrama de estrutura composta do nosso estudo de caso

Para exemplificar um diagrama de estrutura composta, apresentamos a Figura 1.9. Nela figura, uma instância da classe Pessoa interpretando o papel de inscrito realiza uma interação com a instância da classe Inscrição, que interpreta o papel inscrição. Desse modo, uma nova inscrição pode ser feita por uma pessoa.

Figura 1.9 | Exemplo de diagrama de estrutura composta



Fonte: elaborada pelo autor.



Questão para reflexão

Em sua opinião, os diagramas de estruturas compostas devem ser usados em todas as modelagens de sistemas? Que tal compartilhar sua análise com o professor?

Atividades de aprendizagem

1. Analise as afirmativas a seguir:

I - A notação para representar uma ocorrência de colaboração é a mesma utilizada na denominação de um objeto: nome da ocorrência, na sequência dois-pontos (:) e o nome da colaboração.

II - Uma porta é uma característica estrutural de um classificador que representa um ponto de interação entre um classificador e seu ambiente

ou classificador e sua parte interna.

III - A colaboração é graficamente representada por meio de losango tracejado com o seu descritivo.

Sobre as afirmativas:

- a) Somente as afirmativas I e III estão corretas.
- b) Somente as afirmativas II e III estão corretas.
- c) Somente as afirmativas I e II estão corretas.
- d) As três afirmativas estão incorretas.
- e) As três afirmativas estão corretas.

2. Em um diagrama de estrutura composta, podemos encontrar os elementos descritos na alternativa:

- a) Colaboração, ocorrência de colaboração, portas e propriedades.
- b) Colaboração, integração, portas e propriedades.
- c) Colaboração, integração, portas e métodos.
- d) Colaboração, integração, métodos e classes.
- e) Colaboração, integração, métodos e portas.

Fique ligado

Na Unidade 1, mostramos uma visão sobre os diagramas UML, especificamente a categoria dos diagramas estruturais. Em cada seção, tratamos um tema específico: Seção 1 – conceitos, organização e características dos diagramas estruturais e o estudo de caso proposto para a unidade; na Seção 2, iniciamos o detalhamento dos diagramas, o primeiro diagrama foi o de pacotes, sendo possível entender detalhes da sua construção; na Seção 3, tivemos um dos diagramas mais importantes da UML, diagrama de classes, sobre o qual foi possível acompanhar sua estrutura, relacionamentos e comportamentos; na Seção 4, o diagrama de objetos foi detalhado; por fim, o diagrama de estruturas compostas foi conceituado e informações relevantes à sua construção foram apresentadas.

Em todas as seções, tivemos a inclusão de exemplos e material de apoio para complemento da aprendizagem.

Para concluir o estudo da unidade

Aqui finalizamos esta unidade, esperamos que você tenha alcançado a sua meta inicial de estudo, que era entender os diagramas estruturais, seus conceitos, suas características e organização. Para complementar sua aprendizagem, disponibilizamos referências em cada unidade.

Atividades de aprendizagem da unidade

1. Analise as características a seguir e assinale a alternativa que descreve uma característica do diagrama de classe:

- a) Identificar os dados e as dependências funcionais entre partes de um sistema.
- b) Definir os recursos essenciais de um sistema.
- c) Exibindo uma “fotografia” do sistema em um dado momento.
- d) É utilizado para a modelagem de colaborações.

2. Esse diagrama representa a definição dos recursos essenciais para a operação do sistema. Por recursos, entende-se que representam pessoas, materiais, informações e comportamentos. Essa descrição refere-se ao:

- a) Diagrama de classes.
- b) Diagrama de pacotes.
- c) Diagrama de objetos.
- d) Diagrama de estruturas compostas.

3. Observe as afirmativas relacionadas a seguir:

I - public (público): define que a propriedade pode ser vista e utilizada na classe para a qual foi declarada e em qualquer outro elemento externo;

II - protected (protegido): define que a propriedade pode ser vista e utilizada apenas na classe para a qual foi declarada e por classes descendentes;

III - private (privado): define que a propriedade pode ser vista e utilizada em todas as classes do projeto;

IV - package (pacote): define que a propriedade pode ser vista e utilizada pelos elementos que façam parte do mesmo pacote.

Assinale a alternativa que apresenta uma afirmação falsa:

- a) I.
- b) II.
- c) III.
- d) IV.

4. Analise a seguinte descrição de um diagrama: a representação gráfica do objeto em um diagrama é um retângulo dividido em dois compartimentos. Essa descrição é referente ao:

- a) Diagrama de classes.
- b) Diagrama de pacotes.
- c) Diagrama de objetos.
- d) Diagrama de estruturas compostas.

5. Os conceitos de multiplicidade, escopo e estereótipo são aplicáveis a qual diagrama?

- a) Diagrama de classes.
- b) Diagrama de pacotes.
- c) Diagrama de objetos.
- d) Diagrama de estruturas compostas.

Referências

BEZERRA, E. **Princípios de análise e projeto de sistemas com UML**. 3. ed. Rio de Janeiro: Campus, 2015.

DATHAN, B; RAMNATH, S. **Object-oriented analysis, design and implementation: an integrated approach**. 2. ed. St. Cloud: Springer, 2015.

GUEDES, G. T. A. **UML 2: guia prático**. São Paulo: Novatec, 2007.

IBM. **Diagramas de estrutura composta**. Disponível em: http://www.ibm.com/support/knowledgecenter/pt-br/SS8PJ7_8.0.4/com.ibm.xttools.modeler.doc/topics/ccompstruc.html. Acesso em: 17 jan. 2017.

MELO, A. C. **Desenvolvendo aplicações com UML 2.0 do conceitual à implementação**. 2. ed. Rio de Janeiro: Brasport, 2002.

PENDER, T. **UML Bible**. Indianapolis: John Wiley & Sons, 2003.

VISUAL. **Package diagram**. Disponível em: <https://www.visual-paradigm.com/VPGallery/diagrams/Package.html>. Acesso em: 11 abr. 2017.

Diagramas comportamentais da UML

Iolanda Cláudia Sanches Catarino

Objetivos de aprendizagem

- Estudar os conceitos, elementos, notação e exemplos do Diagrama de Casos de Uso.
- Estudar os conceitos, elementos, notação e exemplos do Diagrama de Atividades.
- Estudar os conceitos, elementos, notação e exemplos do Diagrama de Máquina de Estados.

Seção 1 | Diagrama de Casos de Uso: conceitos, componentes, notação e construção

Nesta seção, apresentamos a definição do que é um Diagrama de Casos de Uso (*Use Cases*) e como os elementos deste diagrama são organizados. Também, são apresentados os detalhes que compõem a sua estrutura. Complementarmente, mostramos as notações usadas para representar graficamente o diagrama e usamos alguns exemplos a partir do estudo de caso proposto para esta unidade. Por fim, deixamos alguns links disponíveis para o aluno, a fim de ampliar o seu conhecimento sobre o diagrama.

Seção 2 | Diagrama de Atividades: conceitos, componentes, notação e construção

Nesta seção, é possível obter o entendimento sobre o Diagrama de Atividades e a sua aplicabilidade como diagrama dinâmico da UML. Apresentamos os conceitos relacionados à sua construção, bem como os seus elementos e notação gráfica. Um exemplo de Diagrama de Atividades é ilustrado a partir do estudo de caso proposto para esta unidade. Ao final da seção, alguns links também estão disponíveis para consulta e ampliação do conhecimento sobre o diagrama.

Seção 3 | Diagrama de Máquina de Estados: conceitos, componentes, notação e construção

Nesta seção, explanaremos o Diagrama de Máquina de Estados, seus elementos e notação gráfica e como modelá-lo. Também, apresentamos exemplos de Diagrama de Máquina de Estados construídos a partir do estudo de caso proposto para esta unidade.

Introdução à unidade

Caro(a) aluno(a), nesta unidade, apresentaremos os conceitos relacionados a três diagramas comportamentais, seus objetivos, características, elementos e aspectos envolvidos na sua construção, considerando as regras básicas de notação gráfica da UML. Também, apresentamos uma visão da sua aplicabilidade na modelagem de análise de sistemas, demonstrando exemplos baseados no estudo de caso descrito na unidade anterior.

Os diagramas comportamentais enfatizam o comportamento dinâmico do sistema, ou seja, demonstram as funcionalidades ou os serviços do sistema na sua perspectiva de execução, bem como a manipulação dos objetos do sistema. A partir da UML 2.0, as técnicas de modelagem comportamentais são classificadas em:

- Diagrama de Casos de Uso (*Use Cases*): representa a interação entre Casos de Uso, Atores e seus relacionamentos.
- Documentação de Casos de Uso: descreve a execução dos *Use Cases* de forma narrativa, enfatizando os eventos que são disparados durante a execução de um caso de uso, sendo que o grau de detalhamento da narrativa pode variar.
- Diagrama de Atividades: representa um conjunto de ações que devem ser percorridas para a conclusão de atividades de uma funcionalidade ou até de um processo completo.
- Diagrama de Máquina de Estados: demonstra o comportamento de um objeto através de um conjunto de estados e suas transições em um determinado instante de tempo de execução do sistema.

Na unidade anterior, apresentamos os principais diagramas estruturais, geralmente adotados na modelagem da atividade de Análise, considerando, principalmente, o modelo de desenvolvimento de software - Processo Unificado.

Nesta unidade, vamos explorar três diagramas comportamentais, sendo o Diagrama de Casos de Uso, o Diagrama de Atividades e o Diagrama de Máquina de Estados. A Documentação de Caso de Uso pode ser considerada uma técnica de modelagem textual utilizada de apoio ao Diagrama de Casos de Uso. Vários autores não a mencionam

como uma técnica de modelagem porque se referem ao Diagrama de Casos de Uso como Modelo de Casos de Uso, considerando que, uma vez desenhado um caso de uso, é necessário documentá-lo para relatar o seu funcionamento, pois em um Diagrama de Casos de Uso é ilustrado apenas o nome do caso de uso.

Um Diagrama de Casos de Uso contempla as funcionalidades ou os serviços do sistema e os elementos externos ao sistema que interagem com ele. É o diagrama mais abstrato, flexível e informal da UML, sendo utilizado no início da modelagem do sistema, na atividade de análise, embora venha a ser consultado e, possivelmente, modificado durante todo o processo de engenharia do software. Mais detalhes sobre esse diagrama estão na Seção 1.

Um Diagrama de Atividades descreve os passos a serem percorridos para a conclusão de uma atividade específica, muitas vezes representada por um método com certo grau de complexidade, podendo ser utilizado para modelar um processo completo. Mais detalhes sobre esse diagrama estão na Seção 2.

Por fim, o Diagrama de Máquina de Estados. Esse diagrama demonstra o comportamento de um elemento, através de um conjunto de transições de estado realizadas entre os estados dos objetos de uma classe, de um caso de uso ou mesmo de um subsistema ou sistema completo. Mais detalhes sobre esse diagrama estão na Seção 3.

Cada diagrama acima citado aborda seus conceitos, componentes, notação, como construí-los e exemplificações, usando o estudo de "Controle de Eventos de Extensão Universitária", apresentado na unidade anterior.

Seção 1

Diagrama de Casos de Uso: conceitos, componentes, notação e construção

Introdução à seção

Caro(a) aluno(a), nesta seção, apresentaremos a contextualização sobre o Diagrama de Casos de Uso (*Use Cases*), as notações utilizadas para compor o diagrama e como construí-lo.

Considerando as principais atividades ou fases dos processos de desenvolvimento dos Modelos de Engenharia de Software, as atividades iniciais exigem maior esforço e interação do Analista de Sistemas com os usuários envolvidos nos processos de negócio. Para melhorar a compreensão e aplicabilidade do domínio de um sistema, ou seja, a identificação e familiarização do contexto para o qual será provida uma solução computacional, é importante conhecer e delimitar a fronteira do sistema a partir da compreensão dos fatos e processos, regras de negócios envolvidas nos processos, conceitos específicos do negócio, responsabilidades dos agentes envolvidos nos processos, entre outros fatores, para que, assim, o Analista de Sistemas possa garantir que o software a ser desenvolvido atenda às reais necessidades da organização.

O Diagrama de Casos de Uso pode auxiliar na modelagem de negócio do sistema, bem como na modelagem conceitual das atividades de Análise de Requisitos, Análise e Projeto, ajudando a especificar, visualizar e documentar as características e os serviços do sistema, evoluindo e sendo representado por visões. Na atividade de Análise, o Diagrama de Casos de Uso representa um refinamento dos requisitos funcionais do sistema, tendo um papel central na modelagem do comportamento de um sistema.

A técnica de modelagem de casos de uso foi idealizada pelo conceituado cientista da computação, o sueco Ivar Jacobson, na década de 1970. Em 1992, Jacobson lançou seu método - *Object-Oriented Software Engineering* (OOSE) -, que se caracteriza, principalmente, por utilizar casos de uso para descrever o sistema. Posteriormente, Jacobson se uniu a Grady Booch e a James

Rumbaugh, incorporando a notação do Diagrama de Casos de Uso à UML, tornando essa técnica cada vez mais popular para representar os requisitos funcionais de um software, devido à sua notação gráfica simples e à sua documentação descrita em linguagem natural, o que facilita a comunicação entre a equipe técnica e os usuários do domínio do sistema (BEZERRA, 2007).

1.1 Como construir Diagrama de Casos de Uso

1.1.1 Definição

Segundo Guedes (2008), o Diagrama de Casos de Uso demonstra o comportamento externo do sistema, procurando apresentar o sistema através de uma perspectiva do usuário, demonstrando as funções e os serviços oferecidos e quais usuários podem utilizar cada serviço. Para Booch, Rumbaugh e Jacobson (2006, p. 232), o uso de um diagrama de Caso de Uso: “pode aplicar diagramas de casos de uso para visualizar o comportamento de um sistema, subsistema ou classe, para que os usuários possam entender como utilizar esse elemento e os desenvolvedores possam implementá-lo”.

Segundo Bezerra (2007, p. 43), o Modelo de Casos de Uso (MCU):



É uma representação das funcionalidades externamente observáveis do sistema e dos elementos externos ao sistema que interagem com ele. O MCU é um modelo de análise que representa um refinamento dos requisitos funcionais do sistema em desenvolvimento. A ferramenta da UML utilizada na modelagem de casos de uso é o diagrama de casos de uso.




Na elaboração do Diagrama de Casos de Uso, os casos de usos podem ser categorizados, organizados e agrupados em pacotes, representando um Diagrama de Pacotes para todos os casos de uso de um sistema, sendo necessário, posteriormente, elaborar um Diagrama de Casos de Uso correspondente a cada pacote, assim, formando um Modelo de Casos de Uso.


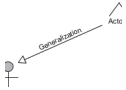
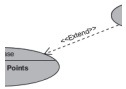
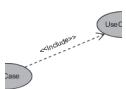
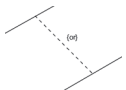
Um Diagrama de Casos de Uso é representado pelos elementos: Atores, Casos de Uso e Relacionamentos. Esses elementos são apresentados na próxima seção.


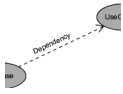
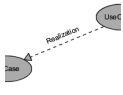

1.1.2 Notação

Para desenhar um Diagrama de Casos de Uso, utilize a notação dos elementos ilustrados no Quadro 2.1. Os elementos foram representados utilizando a ferramenta CASE *Visual Paradigm*.

Quadro 2.1 | Elementos do Diagrama de Casos de Uso

Notação	Nome	Definição
	<i>System</i>	Sistema representa a modelagem da fronteira/contexto do sistema, sendo que os atores são desenhados do lado de fora e os casos de uso são desenhados dentro do retângulo, indicando uma ideia visual clara da fronteira do sistema. A fronteira do sistema é representada por retângulo grande.
	<i>Actor</i>	Ator representa qualquer elemento externo ao sistema que interage com ele. Os Atores representam os papéis desempenhados por pessoas, hardware, dispositivo ou outro sistema que pode utilizar ou interagir com as funcionalidades do sistema. Um Ator primário é responsável em fornecer os dados para que ocorra a execução do serviço. Os Atores são representados por símbolos de "bonecos magros", identificados com um nome que representa qual o papel que o Ator assume no contexto do sistema. Cada Ator deve representar um único papel.
	<i>Use Case</i>	Caso de Uso (Use Case) representa um relato de uso de uma funcionalidade do sistema, sem revelar a estrutura e o comportamento interno desse sistema. Cada funcionalidade deve ser representada, individualmente, como um Caso de Uso. Os Casos de Uso são representados por uma elipse, contendo uma breve descrição dentro do seu símbolo que identifica qual serviço o Caso de Uso assume. Recomenda-se nomear um Caso de Uso com verbo no infinitivo mais substantivo(s). Exemplo: Cadastrar Aluno ou Manter Aluno, Realizar Evento, Gerar Relatório de Eventos.

	<p><i>Association</i></p>	<p>Associação representa um relacionamento de comunicação entre ator e caso de uso, indicando uma interação com o sistema. A associação também pode ser estabelecida entre um Caso de Uso e outros Casos de Uso.</p> <p>Os relacionamentos possíveis são: associação, generalização, extensão e inclusão.</p>
	<p><i>Generalization</i></p>	<p>Generalização é um tipo de relacionamento que representa o reuso de comportamento existente entre casos de uso ou entre atores. Esse relacionamento permite que casos de uso ou Atores herdem características de um caso de uso ou Ator mais genérico.</p> <p>A Generalização de Atores é uma representação abstrata de papéis que eles assumem como função no sistema.</p>
	<p><i>Extend</i></p>	<p>Extensão é um tipo de relacionamento de dependência existente entre casos de uso utilizado para modelar situações em que diferentes sequências de interações podem ser inseridas em um mesmo caso de uso. O caso de uso estendido é uma descrição completa de uma sequência de interações, com significado em si mesmo.</p> <p>O relacionamento de dependência <<extend>> é representado por uma seta que parte do Caso de Uso "estendido" para o Caso de Uso "base".</p>
	<p><i>Include</i></p>	<p>Inclusão é um tipo de relacionamento existente somente entre casos de uso utilizados para representar o mesmo mecanismo de definição de rotinas de linguagens de programação. A inclusão indica uma obrigatoriedade com outro Caso de Uso, sendo que a execução do primeiro obriga também a execução do segundo.</p> <p>O relacionamento de dependência <<include>> é representado por uma seta que parte do Caso de Uso "base" para o Caso de Uso "incluído".</p>
	<p><i>Constration</i></p>	<p>Condição ou restrição representada em linguagem natural no formato de texto, com o propósito de declarar parte da semântica de um elemento.</p>

	<i>Collaboration</i>	Colaboração representa um conjunto de instâncias cooperando entre si para realizar uma tarefa conjunta ou um conjunto de tarefas. As colaborações são representadas no Diagrama de Estruturas Compostas.
	<i>Dependency</i>	Dependências são relacionamentos de utilização entre casos de uso, classes, pacotes ou anotações. Esse relacionamento indica que um único ou um conjunto de elementos do modelo requer outros elementos do modelo para sua especificação ou implementação.
	<i>Realization</i>	Realização representa uma relação de abstração especializada entre dois conjuntos de elementos do modelo, um representando uma especificação do elemento fornecedor e o outro representando uma implementação do elemento cliente. A Realização pode ser usada para modelar refinamento gradual, otimizações, transformações, modelos, síntese de modelos, composição de estruturas etc.
	<i>Note</i>	Nota ou comentário é utilizado para representar observações aos elementos de um diagrama. Um comentário não contém força semântica, mas pode conter informações úteis para os desenvolvedores.

Fonte: adaptado de <https://www.visual-paradigm.com/VPGallery/diagrams/UseCase.html>. Acesso em: 2 fev. 2017.

1.1.3 Exemplos do Diagrama de Casos de Uso

Nas seções anteriores, foram apresentados os conceitos sobre o Diagrama de Casos de Uso, qual a sua estrutura e qual notação usada para elaborá-lo. Nesta seção, mostraremos alguns exemplos do Diagrama de Casos de Uso baseado no estudo de caso proposto “Controle de Eventos de Extensão Universitária”.

A partir da descrição do estudo de caso, identificamos os principais requisitos funcionais relacionados no Quadro 2.2. Alguns requisitos funcionais estão explícitos na descrição, sendo de fácil identificação. Os requisitos RF8, RF9 e RF10 foram identificados a partir da análise e abstração do contexto da aplicação.

Quadro 2.2 | Elementos do Diagrama de Casos de Uso

Nº	Requisito	Descrição
RF1	Cadastro de eventos	O sistema deve prover um cadastro dos eventos de extensão classificados em curso, palestra ou encontro.
RF2	Cadastro de professor	O sistema deve prover um cadastro dos professores responsáveis pelos eventos ou ministrante de um curso ou palestra.
RF3	Cadastro de curso de graduação	O sistema deve prover um cadastro dos cursos de graduação que promovem eventos de extensão.
RF4	Cadastro de participantes	O sistema deve prover um cadastro dos participantes dos eventos classificados em aluno, funcionário ou visitante.
RF5	Cadastro de empresas	O sistema deve prover um cadastro das empresas que os visitantes trabalham.
RF6	Cadastro de instituição de ensino	O sistema deve prover um cadastro das instituições de ensino que os visitantes estão matriculados.
RF7	Registro de inscrição do evento	O sistema deve prover um controle de registro de inscrição online dos eventos de extensão que os participantes realizam via Web.
RF8	Gerar comprovante de inscrição do evento	O sistema deve prover a geração automática de comprovante das inscrições realizadas via Web.
RF9	Emitir comprovante de inscrição do evento	O sistema deve prover a emissão de um comprovante das inscrições realizadas via Web.
RF10	Consulta das inscrições do evento	O sistema deve prover um controle para os participantes consultarem a situação das inscrições dos eventos ou até mesmo cancelar uma inscrição.

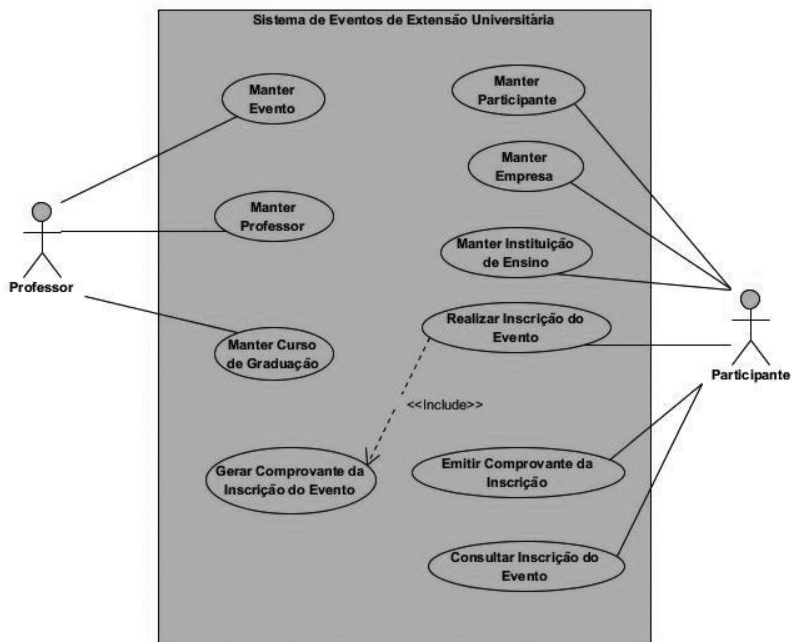
Fonte: elaborado pela autora.

A partir da identificação dos requisitos funcionais listados no Quadro 2.2, apresentamos a seguir o Diagrama de Casos de Uso na visão da atividade de Análise.

A Figura 2.1 mostra o Diagrama de Casos de Uso representado com a fronteira do sistema, ilustrando um caso de uso para cada requisito funcional. Foram definidos os atores primários – Professor e Participante –, que interagem com os casos de uso, sendo

eles os responsáveis em fornecerem os dados para que o caso de uso seja executado. O caso de uso “Gerar Comprovante da Inscrição do Evento” já foi representado com um relacionamento de inclusão (<<Include>>) a partir do caso de uso base “Realizar Inscrição do Evento”, porque o sistema gera automaticamente um comprovante de inscrição a partir de toda inscrição realizada, ou seja, uma obrigatoriedade do sistema.

Figura 2.1 | Diagrama de Casos de Uso



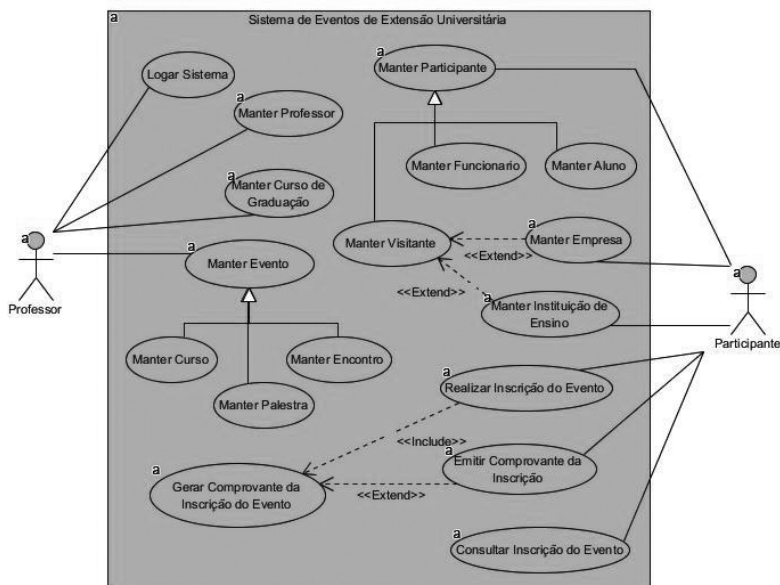
Fonte: elaborada pela autora.

A Figura 2.2 mostra uma segunda visão do Diagrama de Casos de Uso com mais detalhes, o qual poderia ser apresentado como o Diagrama de Casos de Uso na visão da atividade de Projeto ou simplesmente uma evolução do primeiro Diagrama de Casos de Uso, a partir de uma melhor abstração do contexto do sistema. É natural que a modelagem do sistema evolua a partir da validação dos diagramas com os usuários, o que caracteriza as interações e os incrementos do software.

Analisando o caso de uso “Manter Evento”, sendo um evento do tipo curso, palestra ou encontro, esses eventos têm algumas características específicas, então é melhor representar o caso de uso “Manter Evento” com uma generalização. A mesma situação acontece com o caso de uso “Manter Participante”. Ainda para o caso de uso especializado “Manter Visitante” foram estabelecidos dois relacionamentos de extensão (<<Extend>>) com os casos de uso “Manter Empresa” e “Manter Instituição de Ensino”, considerando que, no momento do cadastro de um visitante, ele possa não localizar o cadastro da empresa em que trabalha ou a instituição de ensino que está matriculado, assim, definindo um relacionamento de extensão (<<Extend>>), é possível já chamar o caso de uso “Manter Empresa” ou “Manter Instituição de Ensino” para realizar o novo cadastro deles, não sendo uma obrigatoriedade, facilitando a usabilidade do sistema.

Também, foi estabelecido o relacionamento de extensão (<<Extend>>) entre os casos de uso “Gerar Comprovante da Inscrição do Evento” e o “Emitir Comprovante da Inscrição”, considerando que, a partir da geração do comprovante, o participante já pode emití-lo (impressão), se quiser.

Figura 2.2 | Diagrama de Casos de Uso com Relacionamentos



Fonte: elaborada pela autora.

Após a criação do Diagrama de Casos de Uso, é necessário elaborar a documentação dos Casos de Uso com a descrição deles. Não existe um formato específico de documentação para 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 de forma narrativa, através do uso de pseudocódigo ou até com o código de uma linguagem de programação.

A seguir é exemplificada a documentação do Caso de Uso “Manter Curso de Graduação” no formato numerado descritivo em cenários, especificando-os em Cenário Principal e Cenário(s) Alternativo(s). O cenário principal apresenta uma descrição de uma tarefa que represente o mundo perfeito, sem exceções, e o cenário alternativo relata qualquer situação que represente uma exceção (condição) do cenário principal.

Exemplo de Descrição de Caso de Uso:

Caso de Uso: **Manter Curso de Graduação**

Cenário Principal:

1. Professor solicita cadastro do curso de graduação.
2. Sistema exibe o cadastro de cursos de graduação.
3. Professor informa os dados do curso de graduação.
4. Sistema verifica que curso de graduação não está cadastrado.
5. Sistema verifica que a carga horária total do curso de graduação é válida.
6. Professor confirma o cadastro do curso de graduação.
7. Sistema registra o curso de graduação.
8. Sistema emite Mensagem, informando que o curso de graduação foi cadastrado com sucesso.
9. Sistema encerra o caso de uso.

Cenário Alternativo 4:

- 4. Sistema verifica que já existe o curso de graduação cadastrado.
- 4.1 Sistema recupera dados do curso de graduação cadastrado.
- 4.2 Sistema permite alteração ou exclusão do curso de graduação.
- 4.3 Professor escolhe a opção de alteração.
- 4.4 Professor informa dados (nome, carga horária, turno, área e campus) a serem alterados.
- 4.5 Professor confirma alteração dos dados.
- 4.6 Sistema atualiza dados do curso de graduação.
- 4.7 Sistema encerra o caso de uso.

Cenário Alternativo 4.3:

- 4.3. Professor escolhe a opção de exclusão.
- 4.3.1 Sistema verifica que o curso de graduação não está associado a outro objeto.
- 4.3.2. Sistema emite Mensagem, confirmando a exclusão do curso de graduação.
- 4.3.3. Professor confirma a exclusão do curso de graduação.
- 4.3.4. Sistema exclui o curso.
- 4.3.5. Sistema encerra o caso de uso.

Cenário Alternativo 5:

- 5. Sistema verifica que a carga horária total do curso de graduação não é válida.

5.1 Sistema emite Mensagem, informando que a carga horária do curso não é válida.

5.2 Sistema permite informar nova carga horária do curso de graduação.



Questão para reflexão

Entre as principais técnicas de modelagem da UML, o Diagrama de Casos de Uso e o Diagrama de Classes são considerados os principais diagramas que devem ser adotados na modelagem da atividade de Análise de um sistema, sendo o Diagrama de Casos de Uso um diagrama de fácil visualização e leitura para validar os requisitos funcionais do sistema junto aos usuários.

Qual é a relação do Diagrama de Casos de Uso com o Diagrama de Classes? Como é possível manter essa relação entre os diagramas?



Para saber mais

Para conhecer mais sobre a técnica de modelagem do Diagrama de Casos de Uso, acesse o material indicado a seguir:

Livros:

- BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas com UML**. 2. ed. Rio de Janeiro: Elsevier, 2007.
- BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML: guia do usuário**. 2. ed. Rio de Janeiro: Elsevier, 2006.
- JACOBSON, Ivar; NG, Pan-Wei. **Aspect-oriented software development with use cases**. Addison-Wesley Object Technology Series, 2004.

Fique ligado

Nesta seção, introduzimos o Diagrama de Casos de Uso com sua notação e orientações para modelagem inicial da atividade de Análise de um sistema de software, principalmente para modelagem de sistemas de informação. É importante lembrar que, além do diagrama, é necessário fazer a documentação de cada caso de uso representado no diagrama, independente do formato a ser adotado para descrever exatamente a funcionalidade e o objetivo do caso de uso, pois a UML não define um formato único para documentar os casos de uso. Para facilitar a descrição dos casos de uso, sugerimos desenhar o protótipo da interface gráfica correspondente ao caso de uso, isso ajudará, principalmente, a validar os casos de uso com os usuários do sistema.

Atividades de aprendizagem

1. Considerando as técnicas de modelagem da *Unified Modeling Language* (UML), O Diagrama de Casos de Uso (*Use Cases*) é um diagrama comportamental, utilizado para representar os serviços ou as funcionalidades que o sistema disponibilizará para os usuários.

Sobre o Diagrama de Casos de Uso, julgue as sentenças a seguir:

I. Os casos de uso são utilizados para representar os requisitos funcionais do sistema, ou seja, referem-se aos serviços, às tarefas ou funcionalidades identificadas como necessários ao software e que podem ser utilizados de alguma maneira pelos atores que interagem com o sistema.

II. Cada Caso de Uso é representado por uma elipse, contendo uma breve descrição dentro do seu símbolo, que identifica qual serviço o Caso de Uso assume.

III. Um ator pode ser qualquer elemento externo que interaja com o software, como uma pessoa, um departamento de uma empresa, um outro sistema ou um dispositivo eletrônico.

IV. Um Diagrama de Casos de Uso é representado pelos elementos: Atores, Casos de Uso, Objetos, Atributos e Relacionamentos.

Estão corretos os itens:

- a) I e II.
- b) II e III.
- c) III e IV.
- d) I, II e III.
- e) I, II, III e IV.

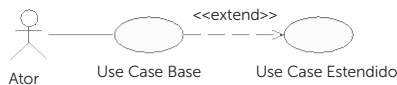
2. No Diagrama de Casos de Uso (*Use Cases*), a associação é um tipo de relacionamento entre os Atores que interagem com o sistema. A associação pode ser estabelecida entre o Ator e Caso de Uso, ou entre um Caso de Uso e outros Casos de Uso. Uma associação estabelecida entre um Ator e Casos de Uso representa que o Ator se utiliza, de alguma maneira, da função representada pelo Caso de Uso. Entre os Casos de Uso, pode-se estabelecer os relacionamentos e Extensão (<<extend>>) ou Inclusão (<<include>>).

Sobre os relacionamentos entre Casos de Uso de um Diagrama de Casos de Uso, julgue as sentenças a seguir:

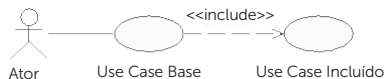
I. Um relacionamento estendido é utilizado para a modelagem da parte de um Caso de Uso que o usuário poderá considerar como um comportamento obrigatório do sistema.

II. O Caso de Uso incluído nunca permanece isolado, mas é apenas instanciado como parte de alguma base maior que o inclui.

III. A notação a seguir representa um relacionamento estendido (estereótipo <<extend>>) que parte do *use case* base para o *use case* estendido.



IV. A notação a seguir representa um relacionamento de inclusão (<<include>>) que parte do *use case* base para o *use case* incluído.



Estão corretos os itens:

- a) I e II.
- b) II e III.
- c) II e IV.
- d) I, II e III.
- e) I, II, III e IV.

Seção 2

Diagrama de Atividades: conceitos, componentes, notação e construção

Introdução à seção

Caro(a) aluno(a), nesta seção, apresentaremos a contextualização sobre o Diagrama de Atividades, os seus elementos e a notação utilizada para compor o diagrama e como construí-lo.

Considerando que os casos de uso foram identificados, é importante evoluir com a modelagem dos processos de negócio do sistema, para uma melhor compreensão do comportamento e funcionamento dos serviços do sistema, ou seja, como determinados casos de uso se relacionam no decorrer do tempo de execução do sistema.

O Diagrama de Atividades pode ser utilizado para modelar uma sequência de atividades, que pode ser um método ou um algoritmo, ou mesmo um processo completo. As atividades podem descrever as operações definidas para os objetos de uma classe, sendo que uma atividade é composta por um conjunto de ações, ou seja, os passos necessários para que a atividade seja concluída. O Diagrama de Atividades também pode representar interações entre objetos.

Segundo Bezerra (2007, p. 308), “os elementos de um diagrama de atividades podem ser divididos em dois grupos: os que são utilizados para representar fluxos de controle sequenciais e os que são utilizados para representar fluxos de controle paralelos”.

Na elaboração do Diagrama de Atividades, o diagrama deve ter um Nó Inicial e pode ter vários ou nenhum Nó Final, o que significa que o processo ou procedimento modelado é cíclico. Um Fluxo de Controle liga uma ação ou atividade a outro, representando o término de um passo e o início do outro. Um Nó de Decisão possui uma única transição de entrada e várias transições de saída, sendo que, para cada transição de saída, deve ser indicada uma condição de guarda.

Em um Diagrama de Atividades utilizado para representar fluxos de controle paralelos, pode haver dois ou mais fluxos de controle sendo executados simultaneamente. Para essa representação, deve-se utilizar as barras de sincronização do tipo bifurcação (*fork*) ou barra de junção (*join*).

O Diagrama de Atividades também pode ser representado com o uso de raias, analogamente, como as de de natação (tradução para *swimlanes*), que dividem o diagrama com suas atividades ou ações. Essa representação, normalmente, é utilizada para representar os processos de negócios que têm a interação entre vários agentes do processo.

2.1 Como construir Diagrama de Atividades

2.1.1 Definição

Um Diagrama de Atividade pode representar o funcionamento de um software, um processo de negócios ou uma funcionalidade do software como um fluxo de trabalho por meio de um conjunto de ações.

Segundo Bezerra (2007, p. 307), o Diagrama de Atividades “pode ser visto como uma extensão dos fluxogramas. Além de possuir toda a semântica existente em um fluxograma, o diagrama de atividade possui notação para representar ações concorrentes, juntamente com a sua sincronização”.

Conforme Booch, Rumbaugh e Jacobson (2006, p. 257), um diagrama de atividades:







Mostra o fluxo de uma atividade para outra. Uma atividade é uma execução em andamento não atômica em uma máquina de estados. As atividades efetivamente resultam em alguma ação, formada pelas computações executáveis atômicas que resultam em uma mudança de estado do sistema ou o retorno de um valor. As ações abrangem a chamada a outras operações, enviando um sinal, criando ou destruindo um objeto ou alguma computação pura, como o cálculo de uma expressão.




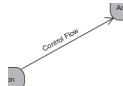

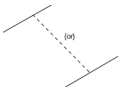

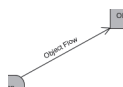

Os elementos de um Diagrama de Atividades podem ser divididos para representar fluxos de controle sequenciais (ou simples) e fluxos de controle paralelos (ou simultâneos). Os elementos comuns de um Diagrama de Atividades são: Atividades, Ações, Nó Inicial, Nó Final, Fluxo de Controle, Nó de Decisão, Condição de Guarda e Bifurcação (Nó Fork ou Junção). Esses elementos são apresentados na próxima seção.


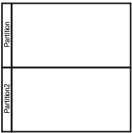



2.1.2 Notação

Para desenhar um Diagrama de Atividades, utilize a notação dos elementos ilustrados no quadro a seguir. Os elementos foram representados utilizando a ferramenta CASE *Visual Paradigm*.

Quadro 2.3 | Elementos do Diagrama de Atividades

Notação	Nome	Definição
	<i>Activity</i>	Atividade representa o nível mais alto do comportamento em um diagrama de atividades. As atividades contêm diversos nós de atividade e linhas de atividade que representam a sequência de tarefas em um fluxo de trabalho que resulta em um comportamento.
	<i>Activity Parameter Node</i>	Nós de Parâmetro de Atividade são nós de objeto no início e no fim de fluxos que fornecem um meio para aceitar entradas para uma atividade e fornecer saídas da atividade, através dos parâmetros de atividade. Usado quando a atividade representada pelo diagrama é chamada de outra atividade, ou quando o diagrama descreve uma operação ou função.
	<i>Action</i>	Ação representa uma única etapa dentro de uma atividade, isto é, uma que não é mais decomposta dentro da atividade, sendo de execução imediata. As ações em uma atividade começam quando todas as condições de entrada são atendidas.
	<i>Input Pin</i>	Pino de entrada de uma ação. Pode-se incluir pinos de entrada e de saída para especificar valores transmitidos de e para a ação quando ela é iniciada. Representa dados que uma ação pode receber quando ele é executado.

	<i>Output Pin</i>	Pino de saída de uma ação. Pode-se incluir pinos de entrada e de saída para especificar valores transmitidos de e para a ação quando ela é iniciada. Representa os dados que uma ação gera quando ele é executado.
	<i>Initial Node</i>	Nó Inicial representa o início da Atividade, sendo representado por um círculo preenchido. Indica a primeira ação ou ações na atividade. O Diagrama de Atividades deve ter um estado inicial obrigatoriamente.
	<i>Activity Final Node</i>	Nó Final representa o fim de uma Atividade, é representado por um círculo preenchido dentro de um círculo vazio. Um Diagrama de Atividades pode ter um ou mais estados finais.
	<i>Control Flow</i>	Fluxo de Controle representa um conector que liga dois nós, enviando sinais de controle. É representado por uma reta contendo uma seta, podendo conter uma descrição, uma condição de guarda ou uma restrição.
	<i>Decision Node</i>	Nó de decisão representa uma escolha entre dois ou mais fluxos. Em geral, um Nó de Decisão é acompanhado por Condições de Guarda, que determinam a condição para que um fluxo possa ser escolhido. Tem uma entrada e duas ou mais saídas. Os fluxos de controle de um Nó de decisão devem ser indicados com condições de guarda.
	<i>Constraint</i>	Condição ou restrição representada em linguagem natural no formato de texto, com o propósito de declarar parte da semântica de um elemento.
	<i>Object Node</i>	Nó de Objeto representa uma instância de uma classe, que pode estar disponível em um determinado ponto de Atividade. São representados como um retângulo.
	<i>Object Flow</i>	Fluxo de Objeto representa um conector que pode possuir objetos ou dados passando através dele, entre o Nó de Ação e o Nó de Objeto. O Fluxo de Objetos pode ser utilizado para modificar o estado de um objeto, definindo um valor para um de seus atributos ou mesmo instanciando o objeto.
	<i>Fork Node</i>	Nó Fork ocorre quando há uma transição de entrada e várias transições de saída. Significa que uma atividade chegou neste ponto e foi subdividida em mais de uma atividade.

	<i>Join Node</i>	Nó de Junção ocorre quando é necessário sincronizar atividades. Significa que mais de uma atividade chegou em um mesmo ponto e criou-se uma nova atividade.
	<i>Swimlane</i>	Partições de Atividade permitem representar o fluxo de um processo que passa por diversos setores ou departamentos de uma empresa, ou mesmo um processo que é manipulado por diversos atores. As Partições de Atividade são formadas por retângulos representando divisões que identificam as zonas de influência de um determinado setor sobre um determinado processo.
	<i>Send Signal Action</i>	Enviar Sinal de Ação representa uma ação que envia uma mensagem ou um sinal para outra atividade ou a um thread simultâneo na mesma atividade. O tipo e o conteúdo da mensagem são indicados pelo título da ação, ou especificados nos comentários adicionais. A ação pode enviar dados de sinal, que pode ser passado para a ação em um fluxo de objeto, ou para um ponto de entrada.
	<i>Accept Event Action</i>	Aceitar Ação de Evento representa uma ação que aguarda uma mensagem ou um sinal antes de continuar a ação. O tipo de mensagem que pode receber a ação é indicado pelo título ou especificado nos comentários adicionais. Se a ação não tiver nenhum fluxo de controle de entrada, ele produz um token sempre que ele recebe uma mensagem. A ação pode receber dados de sinal, que pode ser transmitido em um objeto fluxo ou em um ponto de saída.
	<i>Note</i>	Nota ou comentário é utilizado para representar observações aos elementos de um diagrama. Um comentário não contém força semântica, mas pode conter informações úteis para os desenvolvedores.

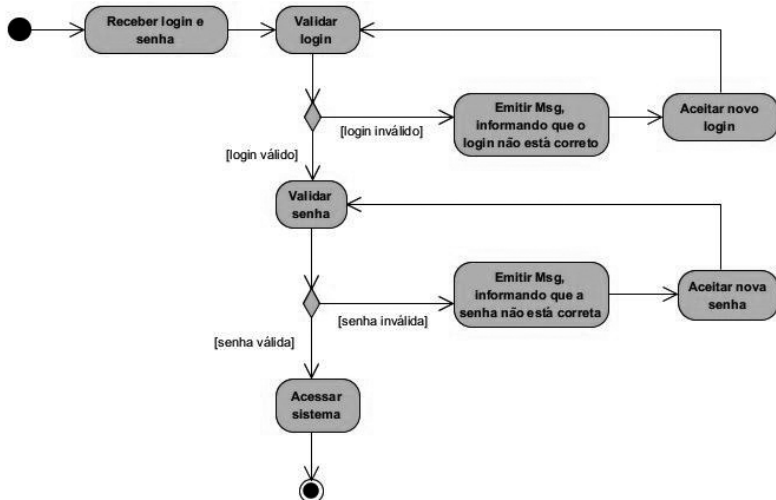
Fonte: adaptado de: <https://www.visual-paradigm.com/VPGallery/diagrams/UseCase.html>. Acesso em: 2 fev. 2017.

2.1.3 Exemplos do Diagrama de Atividades

Nas seções anteriores, foram apresentados os conceitos sobre o Diagrama de Atividades, qual a sua estrutura e qual notação é usada para elaborá-lo. Nesta seção, mostraremos alguns exemplos do Diagrama de Atividades baseado no estudo de caso proposto.

A Figura 2.3 ilustra o Diagrama de Atividades correspondente ao caso de uso “Logar Sistema” em fluxo de controle sequencial. Observe na representação do diagrama que em todo “Nó de Decisão” deve-se indicar, obrigatoriamente, as condições de guarda que determinam a condição para que um fluxo possa ser escolhido.

Figura 2.3 | Diagrama de Atividades – Logar Sistema

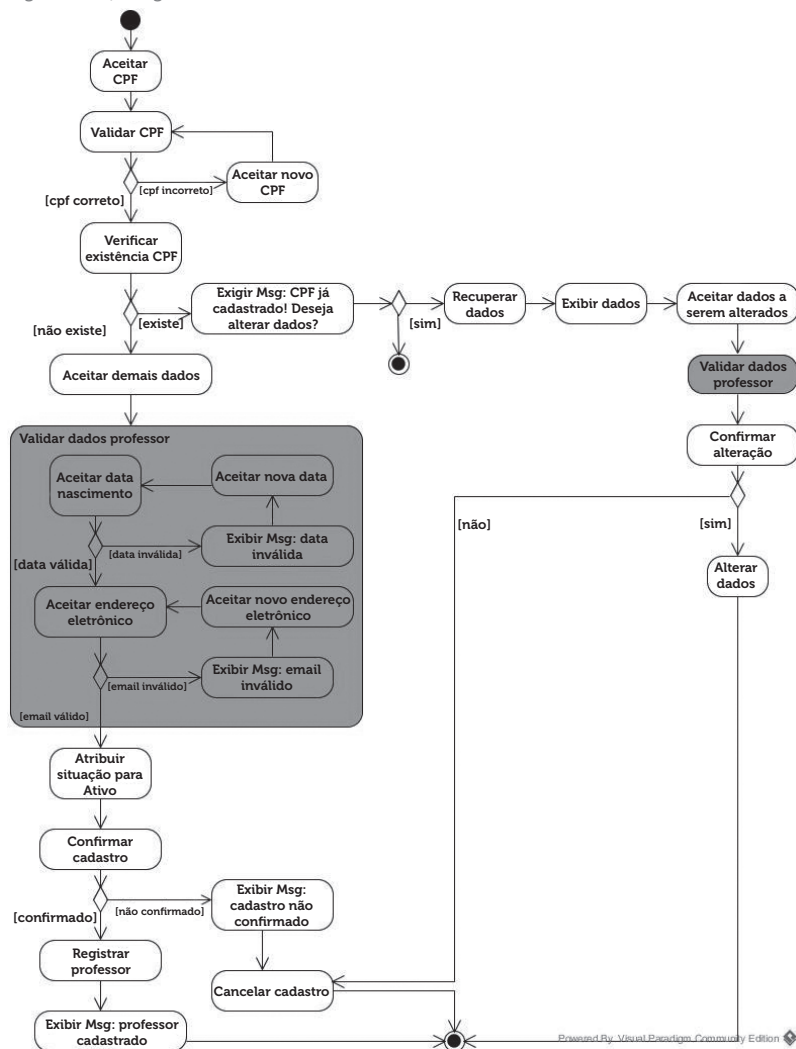


Fonte: elaborada pela autora.

A Figura 2.4 ilustra o Diagrama de Atividades correspondente ao caso de uso “Manter Professor” também em fluxo de controle sequencial, representado em duas partes em função da extensão do diagrama, sendo que algumas validações foram agrupadas em uma Atividade para, posteriormente, ser reutilizada.

O elemento *Atividade* (“Validar dados professor”) ilustrado no diagrama é composto de várias ações. Decidiu-se definir esta validação como uma Atividade porque ela também será executada caso a condição de guarda “se existir CPF” for escolhida para tratar uma alteração de dados.

Figura 2.4 | Diagrama de Atividades – Manter Professor



Fonte: elaborada pela autora.

❓ Questão para reflexão

A partir da modelagem dos casos de uso identificam-se as funcionalidades do sistema, ou seja, os requisitos funcionais do sistema. O Diagrama de Casos de Uso ilustra o nome das funcionalidades, e a Documentação dos Casos de Uso relata a sua descrição em uma sequência lógica de execução em cenários

principais e alternativos, facilitando a compreensão do objetivo do caso de uso. Avançando na modelagem do sistema, é necessário documentar a parte dinâmica do sistema utilizando-se dos diagramas comportamentais, entre eles o Diagrama de Atividades.

Como é possível manter a relação entre o Diagrama de Atividades com o Diagrama de Casos de Uso e o Diagrama de Classes?

Para saber mais

Para conhecer mais sobre a técnica de modelagem do Diagrama de Atividades, acesse o material indicado a seguir:

Livros:

- BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas com UML**. 2. ed. Rio de Janeiro: Elsevier, 2007.
- BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML: guia do usuário**. 2. ed. Rio de Janeiro: Elsevier, 2006.

Fique ligado

Nesta seção, introduzimos o Diagrama de Atividades com sua notação e orientações para especificar o diagrama e adotá-lo como técnica de modelagem da atividade de Análise, iniciando assim a modelagem comportamental do software. O Diagrama de Atividades pode ser adotado para representar os processos de negócio do sistema como um todo, o qual sugerimos adotá-lo para essa finalidade na etapa inicial do processo de desenvolvimento, na atividade de Análise de Requisitos. Na atividade de Análise, indicamos adotá-lo para complementar a documentação dos casos de uso com uma técnica de modelagem gráfica que pode ser vista como uma extensão de fluxogramas, facilitando a compreensão do objetivo dos casos de uso. Entretanto, ao contrário de fluxogramas, o Diagramas de Atividades da UML suporta diversos outros recursos, tais como as partições e os nós do tipo *Fork*, *Join* e de Objeto, que permitem uma modelagem bem mais rica do que simplesmente um fluxograma.

Atividades de aprendizagem

1. A *Unified Modeling Language* (UML) apoia o desenvolvimento incremental, sendo que os modelos podem evoluir com a inclusão de novos detalhes. A UML abrange um conjunto de técnicas de modelagem, classificadas em estruturais e comportamentais.

Sobre a técnica de modelagem Diagrama de Atividades, julgue as sentenças a seguir:

I. O Diagrama de Atividades pode representar os componentes do sistema quando este for ser implementado em termos de módulos de código-fonte, bibliotecas e componentes de software.

II. O Diagrama de Atividades descreve os passos a serem percorridos para a conclusão de uma atividade específica, no entanto, pode adotá-lo para modelar um processo completo.

III. Os elementos de um Diagrama de Atividades podem ser divididos para representar fluxos de controle sequenciais e fluxos de controle paralelos.

IV. Os principais elementos da notação do Diagrama de Atividades são: Objeto, Relacionamentos, Atividade e Ator.

Estão corretos os itens:

- a) I e II.
- b) II e III.
- c) II e IV.
- d) I, II e III.
- e) I, II, III e IV.

2. O Diagrama de Atividades representa um conjunto de ações que devem ser percorridas para a conclusão de atividades de uma funcionalidade ou até de um processo completo. As Atividades podem descrever os métodos correspondentes às operações definidas para os objetos de uma classe.

Assinale a alternativa correta que indica os principais elementos de um Diagrama de Atividades.

- a) Nó Inicial e Final, Nó de Ação, Fluxo de Controle e Nó de Decisão.
- b) Nó Inicial e Final, Nó de Ação, Associação e Objeto.
- c) Nó Inicial e Final, Fluxo de Controle, Objeto de Ação e Vínculo.
- d) Nó de Ação, Fluxo de Controle, Nó de Decisão e Objeto de Ação.
- e) Nó de Ação, Nó de Decisão, Objeto de Ação e Objeto de Controle.

Seção 3

Diagrama de Máquina de Estados: conceitos, componentes, notação e construção

Introdução à seção

Caro(a) aluno(a), nesta seção, apresentaremos a contextualização sobre o Diagrama de Máquina de Estados, os seus elementos e a notação utilizada para compor o diagrama e como construí-lo.

O Diagrama de Máquina de Estados também consiste na modelagem dinâmica do sistema, descrevendo o comportamento dos objetos durante a execução do sistema. A melhor indicação de uso do Diagrama de Máquina de Estados é para modelar o comportamento dos objetos das classes que possuem estados representativos, o qual é afetado e modificado pelos diferentes estados, consequentes dos eventos disparados durante a execução dos casos de uso do sistema.

O Diagrama de Máquina de Estados demonstra o comportamento de um elemento através de um conjunto de transições de estado. O elemento modelado, muitas vezes, é uma instância de uma classe, ou o comportamento de um Caso de Uso, ou mesmo o comportamento de um sistema completo (GUEDES, 2008).

O Diagrama de Máquina de Estados é composto, basicamente, pelos elementos Estado e Transição de Estados. Um estado representa a abstração de uma forma de apresentação dos objetos de uma classe em um determinado instante de tempo. Na definição de Booch, Rumbaugh e Jacobson (2006, p. 290), um estado é “uma condição ou situação na vida de um objeto durante a qual o objeto satisfaz alguma condição, realiza alguma atividade ou aguarda um evento. Um objeto permanece em um estado por uma quantidade finita de tempo”. A transição de estado representa a mudança de estado de um objeto como resposta à chegada de um evento. As transições podem possuir condições de guarda e descrições, se isso for considerado necessário.

Os eventos, assim como os estados, dependem da abstração

aplicada especificamente a um contexto. Eventos são os acontecimentos que fazem os objetos mudarem de estado. Na definição de Rumbaugh (1994, p. 114-115), um evento é “um estímulo individual de um objeto para outro; é algo que acontece em certo momento. Um evento é uma transmissão ou informação unidirecional de um objeto para outro”.

3.1 Como construir Diagrama de Máquina de Estados

3.1.1 Definição

De acordo com Bezerra (2007, p. 287), o Diagrama de Máquina de Estados “permite descrever o ciclo de vida de objetos de uma classe, os eventos que causam a transição de um estado para outro e a realização de operações resultantes”. E ainda, para complementar, segundo Booch, Rumbaugh e Jacobson (2006, p. 288), “uma Máquina de Estados é um comportamento que especifica as sequências de estados pelas quais um objeto passa durante seu tempo de vida em resposta a eventos, juntamente com suas respostas e esses eventos”.

Na elaboração do Diagrama de Máquina de Estados, é importante identificar e especificar as regras de negócio aplicadas ao contexto dos objetos da classe representada. As regras de negócio auxiliam na definição dos estados e das transições entre os estados.

Para identificar os estados de um objeto, deve-se analisar os possíveis valores de seus atributos e as ligações que ele pode realizar com outros objetos. Para definir as transições, deve-se identificar os eventos que podem gerá-las e analisar se há algum fator que condicione o disparo da transição, representando através das condições de guarda (BEZERRA, 2007).

Na representação de um Diagrama de Máquina de Estados, é indicado, além do nome do estado, especificar ações (de entrada ou saída) e atividades a serem executadas quando uma transição é disparada e o objeto assumir um determinado estado, utilizando-se das 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. Atividades internas do tipo *Do* também são chamadas de Atividades de Estado.
- *Entry*: representa as ações realizadas no momento em que o objeto assume o Estado em questão.
- *Exit*: representa as ações executadas antes de o objeto mudar de Estado.


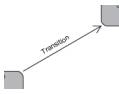

Segundo Guedes (2008), uma ação está associada a uma transição, ou seja, quando o objeto assume um novo estado, ou quando o objeto está mudando de estado, ocasionando uma simples atribuição de um valor a um atributo ou a geração de uma saída. Uma atividade está sempre associada a um estado, ou seja, correspondente aos métodos executados pelo objeto, decorrentes de um evento.






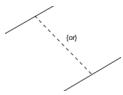



Um Diagrama de Máquina de Estados é representado, basicamente, pelos elementos: Estados, Transições de Estados, Estado Inicial e Estado Final. Esses elementos são apresentados na próxima seção.




3.1.2 Notação

Para desenhar um Diagrama de Máquina de Estados, utilize a notação dos elementos ilustrados no Quadro 2.4. Os elementos foram representados utilizando a ferramenta CASE *Visual Paradigm*.

Quadro 2.4 | Elementos do Diagrama de Máquina de Estados

Notação	Nome	Definição
	<i>State</i>	Estado representa uma situação na vida de um objeto durante a qual ele satisfaz alguma condição ou realiza alguma atividade.
	<i>Transition</i>	Transição representa uma associação entre os estados, com uma seta apontando para um dos estados.
	<i>Initial Pseudo State</i>	Estado inicial representa o estado de um objeto quando ele é criado. Só pode haver um estado inicial em um diagrama de máquina de estados.

	<i>Final State</i>	Estado final representa o fim do ciclo de vida de um objeto. Este estado é opcional e pode haver mais de um estado final em um diagrama de máquina de estados.
	<i>Choice</i>	Estado de escolha representa ponto na transição de estados de um objeto em que deve ser tomada uma decisão, a partir da qual um determinado estado será ou não gerado. As transições de um estado de escolha devem ser indicadas com as condições de guarda.
	<i>Fork</i>	Barra de sincronização com bifurcação representa a ocorrência de estados paralelos, causados por transições concorrentes. Essa barra indica que dois ou mais processos paralelos estejam sincronizados em um determinado momento do processo.
	<i>Join</i>	Barra de sincronização com junção representa a ocorrência de estados paralelos, causados por transições concorrentes. Essa barra indica o momento em que dois ou mais subprocessos se uniram em um único Processo.
	<i>Submachine State</i>	Estado de submáquina indica que um estado contém internamente dois ou mais estados, também chamado estado composto ou concorrente. São utilizados para detalhar um estado principal, porém seus estados não são representados no diagrama.
	<i>Constraint</i>	Condição ou restrição representada em linguagem natural, no formato de texto, com o propósito de declarar parte da semântica de um elemento.
	<i>Entry Point</i>	Ponto de entrada indica a entrada de uma máquina de estado ou estado composto.
	<i>Exit Point</i>	Ponto de saída indica a saída de uma máquina de estado ou estado composto.
	<i>Junction</i>	Vértices de junção são vértices semânticos livres que são usados para encadear várias transições. Eles são usados para construir caminhos de transição compostos entre estados.
	<i>Shallow History</i>	Histórico Superficial representa uma transição para o estado de histórico superficial em um estado composto, chamando o estado mais recente que estava ativo, ou seja, o histórico superficial guarda informações sobre o estado atual, mas não sobre os seus subestados.

	<i>Deep History</i>	Estado Profundo representa a configuração ativa mais recente do estado composto que contém diretamente esse pseudoestado, por exemplo, a configuração de estado que estava ativa quando o estado composto foi encerrado pela última vez, ou seja, o histórico profundo guarda informações sobre o estado atual e os seus subestados.
	<i>Terminate</i>	Terminador indica que a execução da máquina de estado por meio de seu objeto de contexto é terminada. A máquina de estado não sai de nenhum estado, nem executa quaisquer ações de saída que não sejam as associadas à transição que conduz ao pseudoestado de terminação.
	<i>Note</i>	Nota ou comentário é utilizado para representar observações aos elementos de um diagrama. Um comentário não contém força semântica, mas pode conter informações úteis para os desenvolvedores.

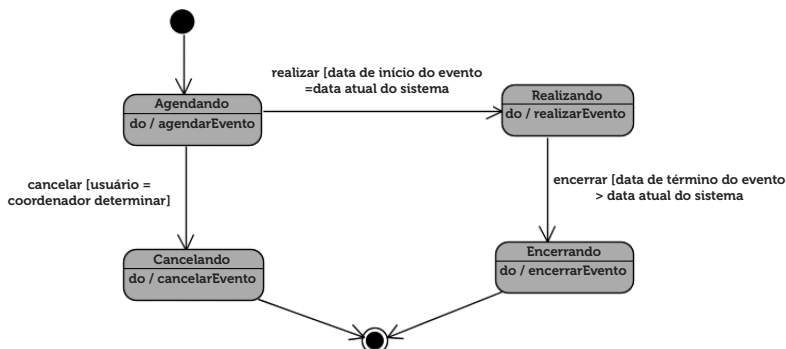
Fonte: adaptado de: <https://www.visual-paradigm.com/VPGallery/diagrams/UseCase.html>. Acesso em: 2 fev. 2017.

3.1.3 Exemplos do Diagrama de Máquina de Estados

Nas seções anteriores, foram apresentados os conceitos sobre o Diagrama de Máquina de Estados, qual a sua estrutura e qual notação usada para elaborá-lo. Nesta seção, mostraremos alguns exemplos do Diagrama de Máquina de Estados baseado no estudo de caso proposto.

A Figura 2.5 ilustra o Diagrama de Máquina de Estados correspondente à classe “Evento”. Observe que os estados não possuem ações de entrada ou saída, apenas a própria atividade interna (cláusula “Do”) correspondente a cada estado que os objetos assumem durante a execução do sistema. Também, foi indicada nas transições de estados, junto aos eventos, uma condição de guarda, a qual cada estado só assumirá a partir da condição de guarda verdadeira, sendo necessário tratar essas condições na implementação da classe “Evento” posteriormente.

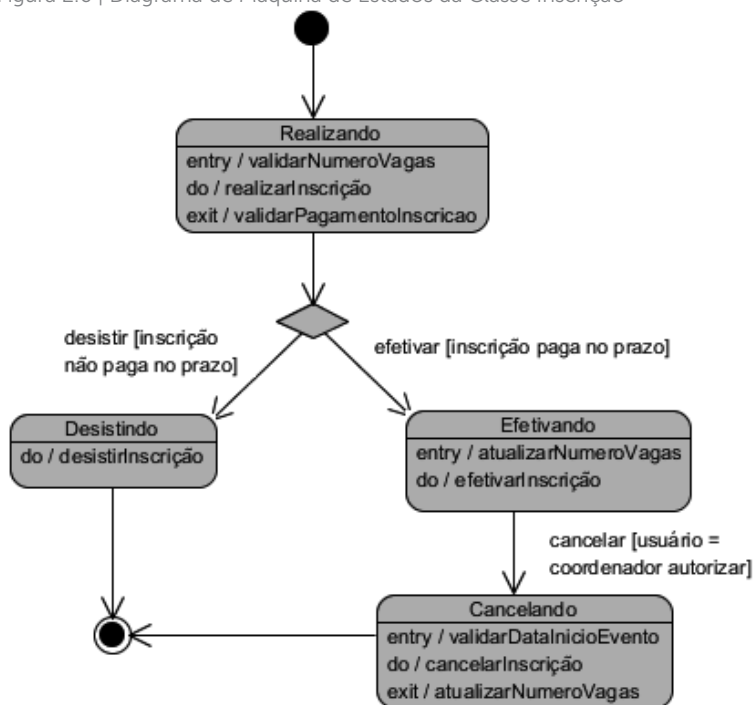
Figura 2.5 | Diagrama de Máquina de Estados da Classe Evento



Fonte: elaborada pela autora.

A Figura 2.6 a seguir ilustra o Diagrama de Máquina de Estados correspondente à classe "Inscrição". Observe que os estados possuem a atividade interna correspondente a cada estado que os objetos assumem durante a execução do sistema, indicado pela cláusula "Do". Também, foi indicado nas transições de estados, junto aos eventos, uma condição de guarda, a qual cada estado só assumirá a partir da condição de guarda verdadeira (se verdadeiro). O estado inicial "Realizando" possui a ação de entrada (cláusula "Entry") "validarNumeroVagas" para verificar se, no momento da realização de uma nova inscrição, o evento ainda possui vaga disponível e, ao sair (cláusula "Exit") do estado para assumir os estados "Desistindo" ou "Efetivando", o sistema deve verificar se as inscrições realizadas foram pagas no prazo determinado, conforme regras de negócio definidas previamente. Considerando que as inscrições pagas no prazo assumem o estado "Efetivando", o sistema deverá atualizar o número de vagas do evento, conforme indicado pela cláusula de entrada "atualizarNumeroVagas". Ainda, definiu-se a transição de estado "cancelar", uma inscrição efetivada sob a condição de autorização do coordenador do evento, assim, indicou-se a cláusula de entrada "validarDataInicioEvento" para consistir que um cancelamento de inscrição só seja permitido caso o evento ainda não tenha iniciado e, posteriormente, a cláusula de saída "atualizarNumeroVagas" do evento para estornar uma vaga que estava preenchida anteriormente.

Figura 2.6 | Diagrama de Máquina de Estados da Classe Inscrição



Fonte: elaborada pela autora.



Questão para reflexão

O Diagrama de Máquina de Estados demonstra o comportamento de um elemento através de um conjunto de estados e transições de estado. O elemento modelado, muitas vezes, é um objeto, ou o comportamento de um caso de uso, ou mesmo o comportamento de um sistema completo, porém muitos autores orientam adotar o Diagrama de Máquina de Estados para modelar os estados dos objetos das classes que possuem estados significantes.

Com base em que os estados e as transições de estados dos objetos de uma classe são definidos? O que implica diretamente a definição de uma transição de estado entre os estados do objeto?

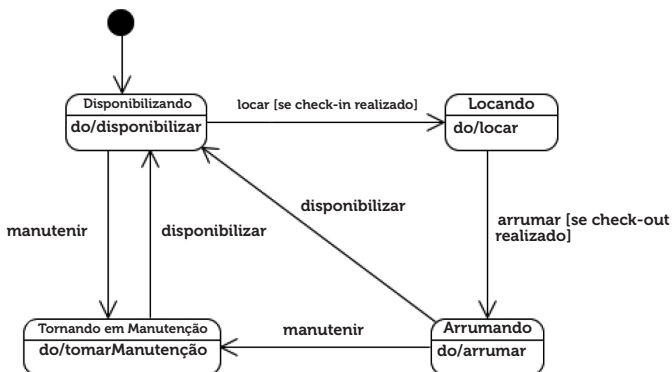
Para conhecer mais sobre a técnica de modelagem do Diagrama de Máquina de Estados, acesse o material indicado a seguir:

Livros:

- BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML: guia do usuário**. 2. ed. Rio de Janeiro: Elsevier, 2006.
- GUEDES, Gilleanes T. A. **UML: uma abordagem prática**. 3. ed. São Paulo: Novatec, 2008.

Atividades de aprendizagem

1. Os profissionais e pesquisadores da área de computação vêm buscando, aplicando e evoluindo práticas e conceitos que fundamentem a análise e o projeto de sistemas de informação orientados a objetos. Assim, surge a *Unified Modeling Language* (UML), uma linguagem flexível, extensível e repleta de recursos que propõe um esquema de representação gráfica mais amplamente utilizado para a modelagem de sistemas orientados a objetos, ou seja, a Linguagem de Modelagem Unificada. O diagrama a seguir representa uma técnica de modelagem Comportamental da UML.



Fonte: elaborada pela autora.

Assinale a alternativa que indica o Diagrama Comportamental representado acima.

- Diagrama de Estruturas Compostas.
- Diagrama de Atividades.

- c) Diagrama de Sequência.
- d) Diagrama de Objetos.
- e) Diagrama de Máquina de Estados.

2. O Diagrama de Máquina de Estados demonstra o comportamento de um elemento através de um conjunto de transições de estado. Conforme Guedes (2008), o elemento modelado, muitas vezes, é uma instância de uma classe, ou o comportamento de um Caso de Uso, ou mesmo o comportamento de um sistema completo.

(Fonte: GUEDES, Gilleanes T. A. **UML: uma abordagem prática**. 3. ed. São Paulo: Novatec, 2008).

Assinale a alternativa que indica a definição do elemento Estado de um Diagrama de Máquina de Estados.

- a) Um estado representa um objeto de uma classe, porém os objetos não apresentam operações, somente atributos, e estes armazenam os valores possuídos pelos objetos em uma determinada situação. Um objeto pode passar por diversos estados dentro de um mesmo processo.
- b) Um estado representa a situação em que um objeto se encontra em um determinado momento durante o período em que este participa de um processo. Um estado representa a abstração de uma forma de apresentação dos objetos de uma classe em um determinado instante de tempo, sendo que um objeto pode passar por diversos estados dentro de um mesmo processo.
- c) Um estado representa uma visão de um conjunto de entidades cooperativas interpretadas por instâncias que cooperam entre si para executar uma função específica. Um objeto pode passar por diversos estados dentro de um mesmo processo.
- d) Um estado representa os componentes de software com sua estrutura e conexão entre os componentes. Os estados de um Diagrama de Máquina de Estados possuem vínculos entre si, representando instâncias das associações entre as classes representadas no Diagrama de Classes.
- e) Um estado representa a ordem temporal em que as mensagens são trocadas entre os objetos envolvidos em um determinado processo. Um objeto pode passar por diversos estados dentro de um mesmo processo, contendo, obrigatoriamente, um estado inicial e um estado final.

Fique ligado

Na Unidade 2, mostramos uma visão sobre os diagramas UML, especificamente a categoria dos diagramas comportamentais que especificam a modelagem dinâmica do software. Em cada seção, tratamos um tema específico: na Seção 1, iniciamos o estudo do Diagrama de Casos de Uso descrevendo seus elementos, notação e exemplos. Na Seção 2, apresentamos o estudo do Diagrama de Atividades, importante diagrama comportamental que pode ser aplicado na representação de um processo de negócio, ou do funcionamento de um caso de uso. Por fim, o Diagrama de Máquina de Estados foi apresentado com seus elementos, notação e exemplos referentes a sua construção. Para todas as seções, tivemos a inclusão de exemplos e material de apoio para complemento da aprendizagem.

Para concluir o estudo da unidade

Aqui finalizamos esta unidade, esperamos que tenha alcançado a sua meta inicial de estudo, que era entender os diagramas comportamentais, seus conceitos, suas características e organização. Para complementar sua aprendizagem, consulte nossas referências bibliográficas.

Atividades de aprendizagem da unidade

1. O Processo Unificado (PU) foi criado para apoiar o desenvolvimento orientado a objetos com a Linguagem de Modelagem Unificada (*Unified Modeling Language* – UML), fornecendo uma forma sistemática de especificar sistemas de softwares para diferentes domínios e tamanhos de projetos.

Considerando as características das fases e atividades do Processo Unificado, assinale a alternativa correta.

a) Os ciclos de desenvolvimento são organizados em quatro fases sucessivas - Concepção, Elaboração, Construção e Transição; e cada fase integra um conjunto de atividades interativas - Requisitos, Análise e Projeto, Implementação e Testes.

- b) Os ciclos de desenvolvimento são organizados em quatro fases sucessivas - Requisitos, Análise e Projeto, Implementação e Testes; e cada fase integra um conjunto de atividades interativas - Concepção, Elaboração, Construção e Transição.
- c) Na fase de Concepção, delimita-se o escopo do sistema, define-se os requisitos funcionais e não funcionais e desenvolvem-se protótipos do sistema para validação dos usuários.
- d) Na fase de Elaboração, define-se a ideia geral do negócio do sistema e a delimitação do escopo do projeto, para obter um desenvolvimento bem fundamentado nos requisitos do usuário.
- e) Na fase de Transição, concentra-se a implementação e os testes das funcionalidades, através do desenvolvimento iterativo e incremental do sistema.

2. A *Unified Modeling Language* (UML) apresenta um conjunto de técnicas de modelagem gráficas, integrando vários elementos (objetos, classes, atributos etc.) do paradigma orientado a objetos. A UML se apoia no desenvolvimento incremental e interativo através de modelos (um conjunto de diagramas) que podem evoluir com a inclusão de novos detalhes. Considerando as técnicas de modelagem COMPORTAMENTAIS, relacione o nome das técnicas de modelagem (diagramas) com suas definições.

Técnicas de Modelagem:

- 1 - Diagrama de Casos de Uso.
- 2 - Documentação de Casos de Uso.
- 3 - Diagrama de Atividades.
- 4 - Diagrama de Máquina de Estados.
- 5 - Diagrama de Sequência.
- 6 - Diagrama de Comunicação.
- 7 - Diagrama de Interação Geral.
- 8 - Diagrama de Tempo.

Definições:

- A. Representa um conjunto de ações que devem ser percorridas para a conclusão de atividades de uma funcionalidade, ou até de um processo completo.
- B. Representa a interação entre Casos de Uso, Atores e seus relacionamentos. Os casos de usos representam os serviços do sistema.
- C. Representa diversos tipos de diagramas de interação para demonstrar um processo geral ou um fluxo de trabalho.

D. Representa a mudança no estado ou condição de uma instância de uma classe, enfatizando as mudanças de estado de um objeto ao longo do tempo.

E. Descreve a execução do Caso de Uso de forma narrativa textual, sendo que o grau de detalhamento pode variar.

F. Representa o inter-relacionamento entre os objetos envolvidos em um processo, representando os objetos, seus vínculos e quais mensagens são trocadas entre si durante um processo.

G. Demonstra o comportamento de um objeto através de um conjunto de estados e suas transições em um determinado instante de tempo de execução do sistema.

H. Representa a ordem temporal em que as mensagens são trocadas entre os objetos envolvidos em um determinado processo.

Assinale a relação correta:

a) 1 – B; 2 – E; 3 – A; 4 – G; 5 – H; 6 – F; 7 – C; 8 – D.

b) 1 – A; 2 – B; 3 – C; 4 – D; 5 – E; 6 – F; 7 – G; 8 – H.

c) 1 – A; 2 – D; 3 – H; 4 – F; 5 – G; 6 – E; 7 – B; 8 – C.

d) 1 – H; 2 – G; 3 – F; 4 – E; 5 – D; 6 – C; 7 – B; 8 – A.

e) 1 – C; 2 – F; 3 – B; 4 – H; 5 – A; 6 – G; 7 – D; 8 – E.

3. A *Unified Modeling Language* (UML) 2.0 abrange treze técnicas de modelagem, classificadas em estruturais e comportamentais. As técnicas estruturais enfatizam a estrutura dos elementos estáticos, a partir da identificação dos objetos. As técnicas de modelagem comportamentais enfatizam o comportamento dinâmico e a interação entre os elementos do sistema. Para representar os requisitos funcionais de um sistema, pode-se utilizar as técnicas de modelagem da UML.

Assinale a alternativa correta que indica o diagrama mais adequado para essa finalidade.

a) Diagrama de Classes.

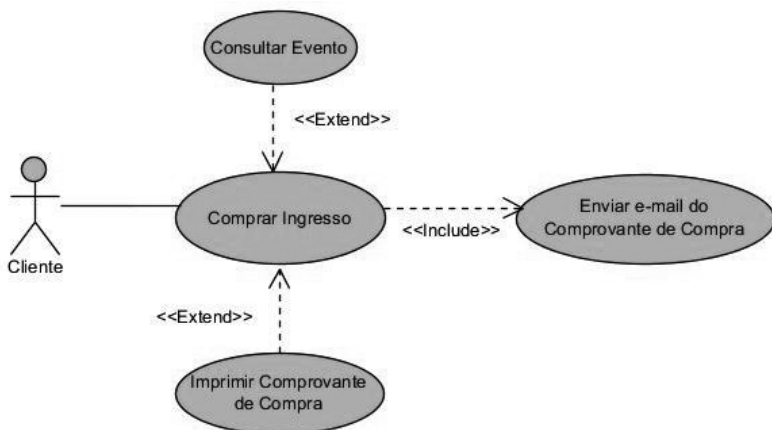
b) Diagrama de Objetos.

c) Diagrama de Casos de Uso.

d) Diagrama de Atividades.

e) Diagrama de Sequência.

4. Na *Unified Modeling Language* (UML), o Diagrama de Casos de Uso proporciona uma forma de representar a aplicação segundo a perspectiva do usuário. Considere o Diagrama de Casos de Uso para o domínio de uma loja virtual de venda de ingressos online para eventos, apresentado na figura a seguir:



Sobre os relacionamentos indicados no diagrama, analise os itens a seguir:

- I. O relacionamento <<Include>> entre os casos de uso “Comprar Ingresso” e “Enviar Email do Comprovante de Compra” representa um caminho obrigatório de execução de funções da aplicação.
- II. O relacionamento <<Include>> entre os casos de uso “Comprar Ingresso” e “Enviar Email do Comprovante de Compra” representa um caminho NÃO obrigatório de execução de funções da aplicação.
- III. O caso de uso “Consultar Evento” e “Imprimir Comprovante de Compra” representam um caminho não obrigatório de execução, ou seja, a partir do caso de uso “Comprar Ingresso” pode-se executar os casos de uso estendidos.
- IV. Os relacionamentos especiais <<Include>> e <<Extends>> são exclusivos para casos de uso.

Estão corretos os itens:

- a) I e II.
- b) II e III.
- c) I, II e III.
- d) I, III e IV
- e) I, II, III e IV.

5. Um Diagrama de Atividades representa uma sequência de atividades, sendo que as atividades podem descrever os métodos correspondentes às operações definidas para os objetos de uma classe. Uma atividade é composta por um conjunto de ações, ou seja, os passos necessários para que a atividade seja concluída. Sobre os elementos básicos de um Diagrama de Atividades, julgue os itens a seguir:

I. Nó Inicial e Final: representam o início e o fim da Atividade.

II. Nó de Ação: é o elemento mais básico de uma Atividade. Um Nó de Ação representa um passo, uma etapa que deve ser executada em uma Atividade. Um Nó de Ação não pode ser decomposto.

III. Fluxo de Controle: é um conector que liga dois nós, enviando sinais de controle. Pode conter uma descrição, uma condição de guarda ou uma restrição.

IV. Nó de Decisão: usado para representar uma escolha entre dois ou mais fluxos. É acompanhado por Condições de Guarda que determinam a condição para que um fluxo possa ser escolhido.

Estão corretos os itens:

a) I e II.

b) II e III.

c) III e IV.

d) I, II e III.

e) I, II, III e IV.

Referências

BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas com UML**. 2. ed. Rio de Janeiro: Elsevier, 2007.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML: guia do usuário**. 2. ed. Rio de Janeiro: Elsevier, 2006.

DATHAN, B. RAMNATH, S. **Object-Oriented Analysis, Design and Implementation: An Integrated Approach**. 2.ed. St. Cloud, MN: Springer, 2015.

GUEDES, Gilleanes T. A. **UML: uma abordagem prática**. 3. ed. São Paulo: Novatec, 2008.

JACOBSON, Ivar; NG, Pan-Wei. **Aspect-oriented software development with use cases**. Addison-Wesley Object Technology Series, 2004.

RUMBAUGH, James et al. **Modelagem e projetos baseados em objetos**. Rio de Janeiro: Campus, 1994.

VISUAL PARADIGM. **VP Galery – UML MODELING**. Disponível em: <<https://www.visual-paradigm.com/VPGallery/diagrams/index.html>>. Acesso em: 2 fev. 2017.

Diagramas de interação da UML

Polyanna P. Gomes Fabris

Objetivos de aprendizagem

Caro(a) aluno(a), você será levado a estudar e conhecer conceitos importantes dos diagramas de interação da UML.

A UML é uma linguagem de modelagem unificada. Em 1995, Ivar Jacobson, no livro *Object-orientad Software Engineering*, aborda sobre o processo unificado, baseado em casos de uso e seus diagramas que dão embasamento para a construção do software. Esses processos são centrados em uma arquitetura, diagrama de caso de uso (referente à visão e aos seus módulos), diagramas de componentes e implantação, diagramas de interação, sendo estes divididos em diagrama de sequência e diagrama de colaboração.

Nesta unidade, vamos aprofundar nossos conhecimentos nos diagramas de interação, diagrama de sequência e diagrama de colaboração.

Seção 1 | Introdução aos diagramas de interação UML

Caro(a) aluno(a), nesta seção, vamos abordar sobre diagrama de interação da UML, que tem uma visão geral de diversas comunicações entre objetos. Veremos que como exemplos de diagrama de interação temos os diagramas de sequência e diagrama de colaboração, e abordaremos a semelhança entre ambos. Tenha uma boa leitura!

Seção 2 | Diagrama de Sequência: conceitos, componentes, notação e construção

Caro(a) aluno(a), nesta seção, você conhecerá o diagrama de sequência, o qual tem como objetivo mostrar a troca de mensagens

entre diversos objetos, em uma situação específica e delimitada no tempo. Vamos verificar também seus conceitos, componentes, notação e construção. Tenha uma boa leitura!

Seção 3 | Diagrama de Colaboração: conceitos, componentes, notação e construção

Caro(a) aluno(a), nesta seção, você conhecerá o diagrama de colaboração, que assim como o diagrama de sequência, representa a troca de mensagens entre objetos, porém, no diagrama de colaboração, o tempo não é modelado explicitamente, uma vez que a ordem das mensagens é definida através de enumeração. Vamos verificar seus conceitos, componentes, notação e construção. Tenha uma boa leitura!

Introdução à unidade

Caro(a) aluno(a), vamos iniciar nosso aprendizado pelos diagramas de interação da UML.

Na Unidade 1, Seção 1, apresentamos a UML, e como complemento podemos dizer também que ela é uma linguagem de modelagem unificada que pode ser considerada um padrão mundial de modelagem para sistemas, possuindo diagramas que mostram a função de cada indivíduo no processo de desenvolvimento do software. Podemos utilizar a UML independente da linguagem de programação escolhida. As pessoas podem achar que a UML é um processo, quando, na verdade, ela é uma forma de comunicação que um determinado processo pode utilizar. Nesta unidade, vamos focar nossa atenção nos diagramas de interação.

Dentro dos diagramas de interação temos o diagrama de sequência, que trataremos na Seção 2, que representa o fluxo de processos (mensagens) em um programa computacional, e o diagrama de colaboração, da Seção 3, que mostra a interação de objetos e seus relacionamentos, informando as mensagens que podem ser trocadas entre eles. Esses diagramas são isomórficos, ou seja, são similares.

Uma questão que podemos inserir aqui é: “Devo utilizar o diagrama de sequência ou de colaboração?”

Vamos analisar a importância de ambos e você verá que essa é uma questão pessoal, alguns analistas usam os dois, outros optam por um ou outro. Durante esta unidade, você terá o conhecimento necessário para tomar essa decisão.

Vamos conhecer um pouco mais sobre esses diagramas.

Seção 1

Introdução aos diagramas de interação UML

Introdução à seção

Caro(a) aluno(a), nesta seção, abordaremos o diagrama de interação da UML, que tem uma visão geral de diversas comunicações entre objetos e faz parte da categorização dos diagramas estruturais.

Durante o levantamento de requisitos, fazemos alguns questionamentos, como:

- De que forma os objetos interagem em determinado caso de uso?
- Quais informações devem ser enviadas em uma mensagem de um objeto para o outro? E em que ordem?

Os diagramas de interação respondem a essas perguntas, pois eles servem para representar o funcionamento interno do sistema para que o ator consiga fazer a realização de um caso de uso. Bezerra (2015, p.1) relata que “em consequência do crescimento da importância da informação, surgiu a necessidade de gerenciar informações de uma forma adequada e eficiente”. Atualmente, as empresas têm a necessidade do sistema, mas para que este seja confiável, o analista precisa utilizar de metodologias adequadas para compreender a necessidade de seus clientes.

Segundo Bezerra (2015, p. 227), “Somente após a construção de diagramas de interação para os cenários de um caso de uso, pode-se ter certeza de que todas as responsabilidades que os objetos devem cumprir foram identificadas”. Os diagramas de interação têm como objetivo obter informações adicionais e auxiliar a aprimorar outros modelos, como o diagrama de classe, verificando, por exemplo, quais são as operações de uma classe e quais são os objetos que participam do caso de uso.

A construção dos diagramas de interação consolida o entendimento sobre os aspectos dinâmicos do sistema. Ele representa como o sistema age internamente para que um ator atinja seu objetivo na conclusão do caso de uso. Geralmente, a modelagem de um sistema tem vários

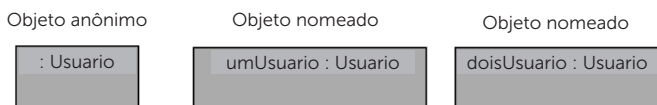
diagramas de interação, e esse conjunto, com todos os diagramas de interação do sistema, constitui o seu modelo de interações.

São fornecidos aos desenvolvedores do software uma visão entre os objetos e as mensagens trocadas na realização do caso de uso. A mensagem é um conceito básico da interação entre objetos, ela indica a solicitação de um objeto emissor a um objeto receptor para que este execute uma determinada ação. A mensagem deve possuir informação suficiente para que a operação do objeto receptor possa executar a comunicação inicial dos objetos, conforme mostra a Figura 3.4, e também de forma mais completa onde o objeto está executando uma ação, como exposto na Figura 3.14, na Seção 2.

Para os diagramas de interação, os objetos são representados usando a mesma notação do diagrama de objetos, conforme apresentamos na Unidade 1, Seção 4: **"nomeObjeto: NomeClasse"** ou **"nome_objeto: Nome_Classe"**. A UML orienta a definir uma notação dos diagramas criados e mantê-la para melhor entendimento dos envolvidos.

Esses objetos podem ser representados como anônimos ou nomeados, dependendo da situação. Na Figura 3.1, utilizando a ferramenta *Visual Paradigm Community Edition*, representamos um objeto da classe "Usuario", mas ele se apresenta de forma anônima, ou seja, para esse objeto não foi dado um nome. Já nas duas instâncias, representando objeto nomeado, podemos verificar que temos o "umUsuario" e "doisUsuario" representando os objetos ou as instâncias da classe "Usuário".

Figura 3.1 | Representação de objeto anônimo e nomeado



Fonte: elaborada pela autora.

Para os objetos, podemos utilizar também o termo multiobjeto, que seria uma coleção de objetos de uma mesma classe (BEZERRA, 2015). Ele pode representar a quantidade de instâncias de um relacionamento (MELO, 2002), que é a associação de conectividade um para muitos, conforme descrito na Seção 3, Unidade 1, ou pode representar uma lista (temporária ou não) de objetos.

Veja na Tabela 3.1 alguns exemplos de cardinalidade baseados no exemplo das classes evento e inscrição.

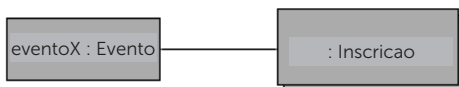
Tabela 3.1 | Exemplo de Cardinalidade

Multiplcidade (quantidade mínima e máxima)	Significado	Exemplo
0..1	zero ou um	Um evento pode ter nenhuma ou 1 inscrição
1	somente um	Um evento pode ter uma e somente uma inscrição
0..*	zero ou muitos	Um evento pode ter nenhuma ou muitas inscrições
*	muitos	Um evento pode ter muitas inscrições
1..*	um ou muitos	Um evento pode ter uma ou muitas inscrições
1..12	um ou no máximo 12	Um evento pode ter uma ou no máximo 12 inscrições

Fonte: elaborada pela autora.

O multiobjeto é representado na UML através de dois retângulos superpostos, sendo que seu nome fica no retângulo que está por cima e segue a mesma nomenclatura utilizada para objetos, conforme a Figura 3.2.

Figura 3.2 | Exemplo de multiobjeto

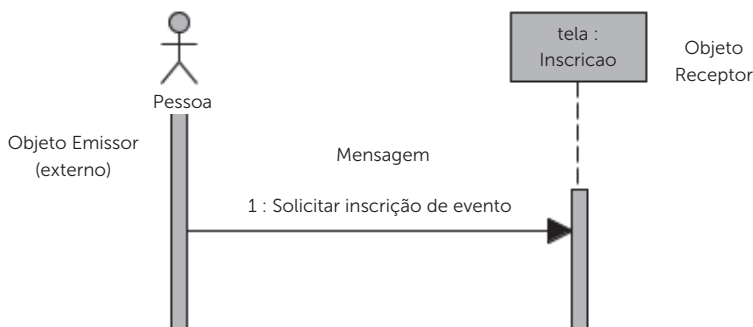


Fonte: elaborada pela autora.

Observando a Figura 3.2, usando a notação do diagrama de colaboração, que será mais detalhado na Seção 3 desta unidade, podemos dizer que a instância “eventoX” da classe “Evento” tem um relacionamento “0 (zero) ou muitos” com o objeto “Inscricao”, ou seja, para representar que um evento pode ter (ou tem) 0 (zero) ou muitas inscrições. Como exemplo de instância de inscrição, usando a notação do diagrama de objetos, podemos apresentar “inscricao1:Inscricao”, “inscricao2:Inscricao”, e assim por diante.

A Figura 3.3 é um exemplo de interação entre os objetos, usando a notação do diagrama de sequência, apresentado com mais detalhes na Seção 2 desta unidade. Neste diagrama, o objeto emissor é o ator (Aluno) que envia uma mensagem para o objeto receptor da classe Inscrição, representado com uma interface (tela) para a realização de inscrição de um evento, conforme mostrado em nosso estudo de caso.

Figura 3.3 | Envio de mensagem



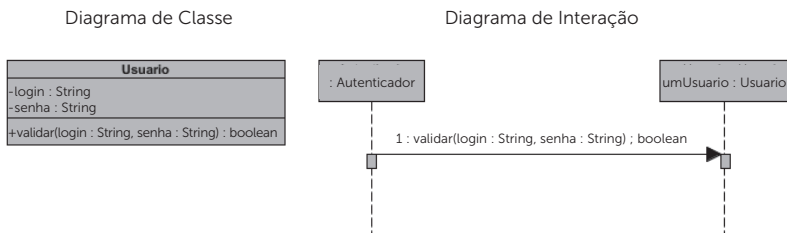
Fonte: elaborada pela autora.

Para começar a modelar os diagramas de interação, é importante ter o diagrama de caso de uso e o diagrama de classe modelado, pois existe a rastreabilidade entre a “classe e o caso de uso”, e as mensagens trocadas entre os objetos são, na maioria das vezes, as operações da classe. E durante a criação de um diagrama de sequência, por exemplo, podemos identificar operações não identificadas na criação do diagrama de classe, ou seja, os diagramas da UML se completam.

No exemplo a seguir, na Figura 3.4, temos uma classe “Usuario” representada com seus atributos e uma operação “validar()”, observe como a classe e sua operação estão representadas em um diagrama de interação, neste caso, o diagrama de sequência.

Quando um objeto da classe “Autenticador” se relaciona com um objeto da classe “Usuario”, realizam uma troca de mensagem utilizando operação “validar()”, passando os parâmetros (login e senha) e mostrando a representação de um retorno (boolean).

Figura 3.4| Exemplo de diagramas de interação e de classe

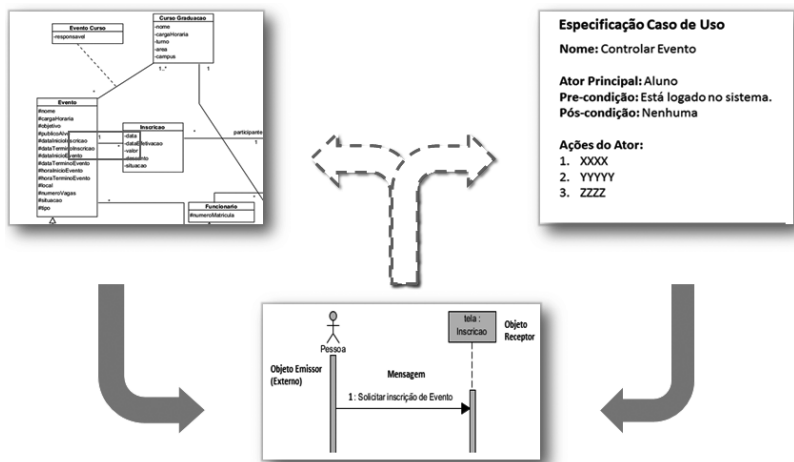


Fonte: A autora (2017).

Dicas para construção de diagramas de interação

Para se construir um diagrama de interação precisamos do diagrama de caso de uso com o detalhamento de seus cenários e, também, do diagrama de classe com seus objetos e relacionamentos (BEZERRA, 2015). Por meio dos diagramas de interação é possível apoiar no refinamento do diagrama de classe e caso de uso (UC). Em alguns casos, durante a construção de um diagrama de interação, identificamos novas classes, assim como novos atributos, operações, associações e, também, o refinamento de algum cenário de UC. Caso seja identificada de fato a necessidade de refinamento, o analista deve voltar aos demais diagramas e refinar os itens. Veja na Figura 3.5 a rastreabilidade entre os modelos.

3.5 | Rastreabilidade entre os modelos



Fonte: elaborada pela autora.



Questão para reflexão

Conseguiu identificar a importância em utilizar a UML e onde seus diagramas se completam?

E tem mais, a UML é uma linguagem conhecida mundialmente, ou seja, onde quer que você venha trabalhar, se o padrão adotado for UML, todos estarão falando a mesma língua. É claro que cada empresa ou equipe definirá seu padrão de trabalho, mas, quando

amparada por uma linguagem de modelagem unificada, torna-se mais fácil a criação desses modelos.

Atividades de aprendizagem

- 1.** Utilizando o caso de uso de eventos de extensão, determine um ator e uma classe para realizar a interação entre eles.
- 2.** Analise a frase a seguir, informe se é verdadeira ou falsa e justifique. “O objetivo do diagrama de interação é identificar as mensagens e responsabilidades (operações e atributos) dos envolvidos no processo de criação de um software”.

Fique ligado

Nesta seção, foram apresentados:

- informações sobre o diagrama de interação, bem como uma visão geral dos diagramas de sequência e colaboração;
- exemplificação do diagrama de sequência;
- um apanhado sobre a rastreabilidade entre os diagramas de caso de uso, classe e diagrama de sequência.

Como foi o estudo desta seção? Compartilhe com o seu professor e colegas!

Seção 2

Diagrama de Sequência: conceitos, componentes, notação e construção

Introdução à seção

Caro(a) aluno(a), nesta seção, você conhecerá o diagrama de sequência, seus conceitos, componentes, notação e construção com base no estudo de caso já abordado desde a unidade 1.

O diagrama de sequência representa uma visão temporal, em que as mensagens são trocadas entre os objetos de um determinado caso de uso, sendo responsável por verificar o fluxo de envio e recebimento para que uma ação seja executada. Neste momento, serve como rastreabilidade do diagrama de classe.

Segundo Guedes (2011, p. 192):

Este diagrama se baseia no diagrama de caso de uso, havendo normalmente um diagrama de sequência para cada caso de uso declarado, uma vez que um caso de uso, em geral refere-se a um processo disparado por um ator.

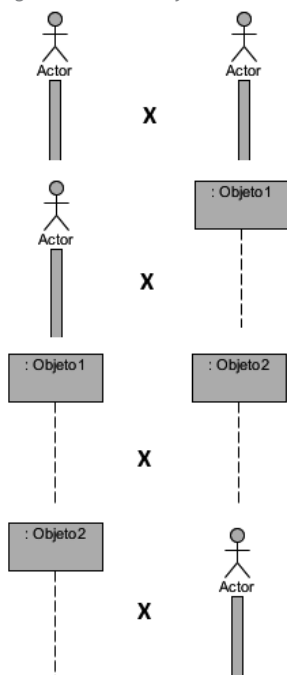


Quanto aos seus cenários, utiliza as operações definidas no diagrama de classe, sendo uma forma de validação deles.

Ele é formado por mensagens, atores, objetos, linha de vida e foco de controle. As mensagens são representadas por setas e são lidas de cima para baixo, assim como transcorre o tempo. Vamos abordar um pouco mais sobre cada um desses elementos?!

1. Mensagem: são as solicitações que um objeto envia para o outro com o objetivo de executar uma ação, ou seja, para fazer uma requisição a um objeto é necessário “enviar uma mensagem” para ele. Veja na Figura 3.6 as possíveis trocas de mensagens.

Figura 3.6 | Troca de mensagens entre os objetos



Fonte: elaborada pela autora.

1.1. Mensagens síncronas: a mensagem pode ser síncrona quando o emissor fica bloqueado até o receptor aceitar e tratar a mensagem. Na Figura 3.7, o "Ator" enviará uma mensagem para o objeto "objeto: NomeClasse" e este envia outra mensagem ao objeto "objeto2:NomeClasse2", que executará sua operação "operacao01()". Como exemplo, podemos utilizar um fogão automático, no qual enviamos a mensagem para acender a "boca de um fogão" e ficamos aguardando o retorno do acendimento. E quando o acendimento acontece, podemos entender como a execução da ação dele e um retorno do envio de mensagem.

Exemplos de mensagens:

a. Uma simples mensagem enviada do ator para uma interface representando sua solicitação representamos com texto básico, "mensagem": **"Solicitar inscrição do evento"** ou **"Consultar inscrição"**.

b. Uma mensagem que é uma operação de uma classe é representada com a descrição da operação mais um par de parênteses no final "nomeOperacao()": mostrarEvento() ou consultarInscricao(), e como essa operação tem procedência do diagrama de classe, usará a mesma notação.

c. Uma mensagem que é uma operação de uma classe, assim como o item "b", e que precisa representar a necessidade de um retorno ou não.

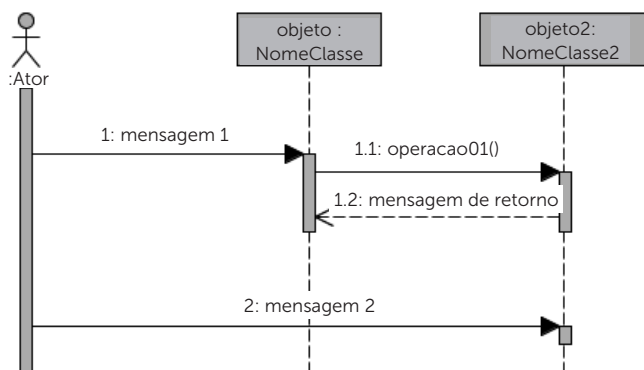
i. **Exemplo 1:** "nomeOperacao():void", nesse caso, o "void" é a palavra reservada que representa que não terá retorno.

ii. **Exemplo 2:** "nomeOperacao():String", nesse caso, o tipo de dados "String" representa que, ao executar essa operação, teremos como retorno uma "String".

d. Uma mensagem que é uma operação de uma classe, assim como o item "b", e que precisa representar uma mensagem com passagem de parâmetro utilizamos "nomeOperacao(atributo: tipo de dado)": consultarInscricao(cpf:String).

1.1.1. Aproveitando o exemplo, também temos representada na Figura 3.7 a **mensagem de retorno**, que é uma mensagem que um objeto envia ao outro em resposta à mensagem recebida após a execução de uma ação. Na figura, podemos ler "mensagem de retorno", representada pela linha tracejada.

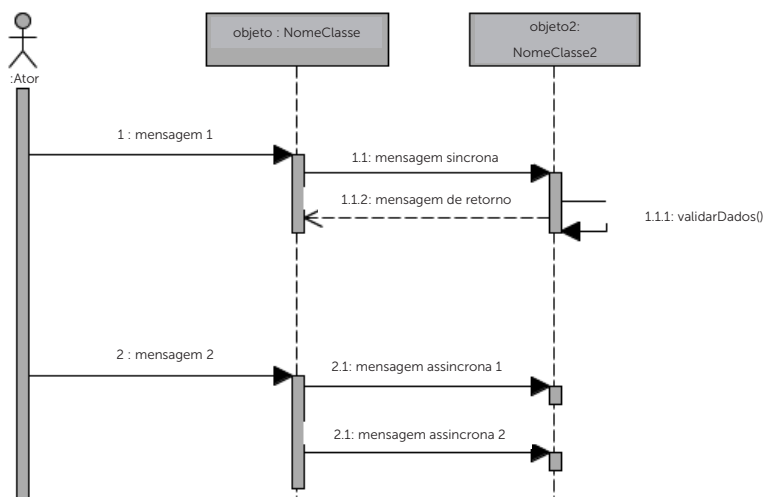
Figura 3.7 | Troca de mensagens síncronas



Fonte: elaborada pela autora.

1.2. Mensagens assíncronas: uma mensagem instantânea ou assíncrona é quando o emissor continua a emitir mensagem, não existindo dependência. No exemplo da Figura 3.8, temos as mensagens “mensagem assíncrona 1” e “mensagem assíncrona 2”, as quais são enviadas do “objeto: NomeClasse” que é o emissor para o “objeto2: NomeClasse2”, o qual é o receptor e não aguardará uma resposta. Como exemplo mais prático de mensagem assíncrona temos o envio de e-mail, em que mensagens são disparadas do emissor para o receptor diversas vezes e o processo não fica interrompido no aguardo de respostas. A maioria das ferramentas CASE representa a mensagem assíncrona com uma seta de meia ponta ou uma seta de “pé de galinha”.

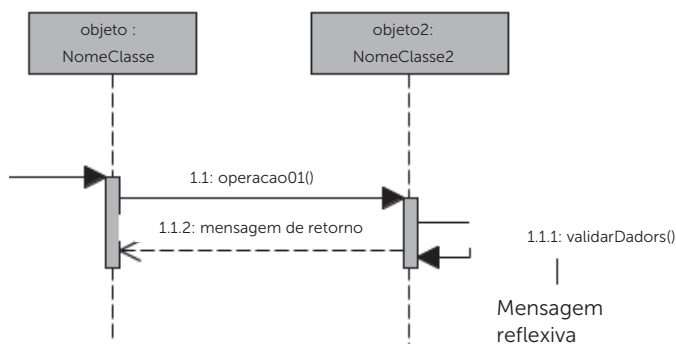
Figura 3.8 | Troca de mensagens assíncronas



Fonte: elaborada pela autora.

1.3. Mensagem reflexiva ou automensagem: o objeto remetente da mensagem é também o receptor. Na Figura 3.8, no “objeto2” da classe “NomeClasse2”, foi representada a validação no objeto da classe “objeto2”, com operação “validarDados()”, fazendo a validação no próprio “objeto2”. Em muitos casos, é utilizado para representar “validação de CPF”, “validação de usuário e senha”, exibir uma nova página etc.

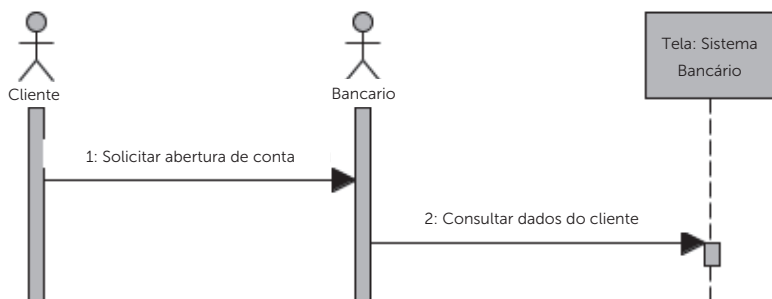
Figura 3.9 | Exemplo de mensagem reflexiva



Fonte: elaborada pela autora.

2. Atores: são os mesmos atores que participam da realização do caso de uso, conforme apresentado na Unidade 2 [verificar Seção 2], em que o ator envia a mensagem para objeto como uma forma de interação para solicitar a execução de uma determinada ação. No exemplo da Figura 3.10, temos um ator trocando mensagem com outro ator e também realizando interação com interface do sistema bancário. Esse exemplo mostra que o ator “Cliente” não tem acesso diretamente à interface do banco, sendo necessário fazer uma requisição para o ator, “Bancário”, que tem acesso à interface do sistema bancário.

Figura 3.10 | Troca de mensagem entre atores (Sistema Bancário)

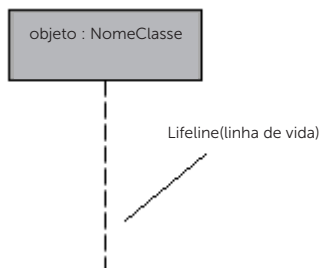


Fonte: elaborada pela autora.

3. Objeto: no diagrama de sequência, a notação do objeto se dá conforme apresentado na Unidade 1, Seção 4, e também nesta unidade, Seção 1, Figura 3.1.

4. Linha de vida (lifeline): representa a existência de um objeto em um tempo particular, ou seja, indica quando um objeto foi instanciado. É representada por barras verticais tracejadas, nas quais em sua cabeça, teremos o objeto. Algumas ferramentas representam o ator também acima da linha de vida.

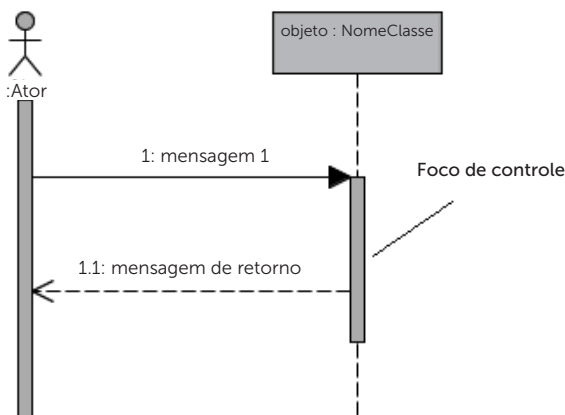
Figura 3.11 | Linha de Vida



Fonte: elaborada pela autora.

5. Foco de controle: é um retângulo sobre a linha de vida que mostra o período de tempo durante o qual um objeto está executando uma ação, diretamente ou com um procedimento subordinado. E pode aparecer diversas vezes ao longo da linha de vida.

Figura 3.12 | Foco de controle



Fonte: elaborada pela autora.



Questão para reflexão

Qual exemplo do dia a dia você poderia utilizar para representar uma mensagem síncrona e assíncrona?



Para saber mais

Na situação de um chat de mensagem, no qual existe um emissor e um receptor, como podemos classificar essa troca de mensagem (síncrona ou assíncrona)?

Agora que você já conheceu a notação do diagrama de sequência, vamos utilizar nosso estudo de caso “Controle de Eventos de Extensão Universitária” e modelar parte dele?

Para isso, vamos precisar:

1) De um ator, para fazer a interação com os objetos da classe, neste caso, definimos o ator “Participante”.

2) Utilizar as classes “Evento”, “Inscrição” e “Pessoa”.

3.13 | Diagrama de Classe do estudo de caso “Controle de Eventos de Extensão Universitária”

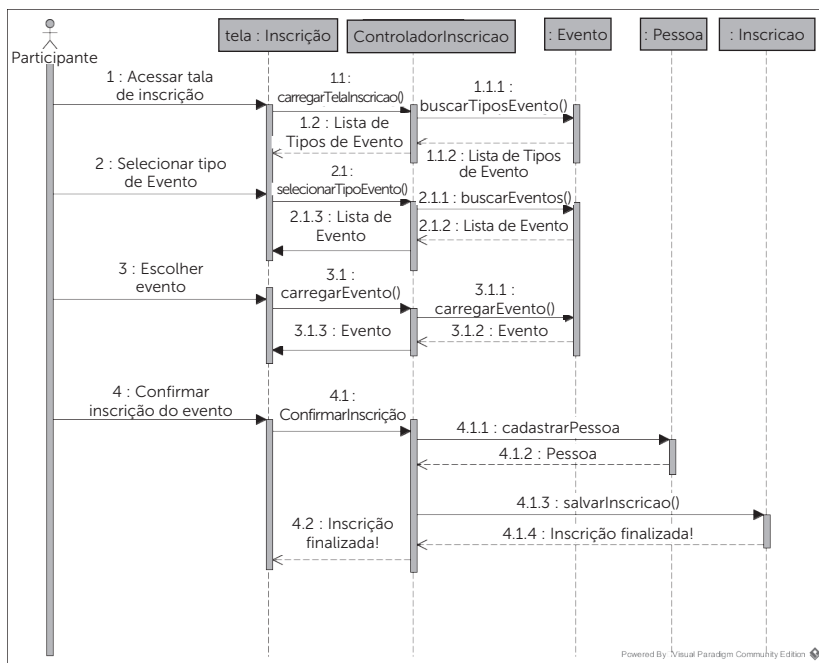


Fonte: elaborada pela autora.

3) De uma representação de interface da tela de inscrição.

4) De uma classe controladora chamada “ControladorInscricao”, na qual trataremos de todas as regras de negócio e do fluxo da aplicação.

3.14 | Diagrama de sequência do estudo de caso “Controle de Eventos de Extensão Universitária”



Fonte: elaborada pela autora.

No diagrama da Figura 3.14, o ator externo é o “Participante” que inicia a comunicação na tela de “Inscrição”, e mensagens são trocadas entre os objetos com o objetivo de realizar a inscrição de um evento.

Para cada objeto representado podemos perceber que se iniciou uma “linha de vida” e, quando o objeto começa a interagir, utilizamos o foco de controle, que possui suas mensagens de forma numeradas e ordenadas, representando a sequência das trocas de mensagens.

Para saber mais

Você sabia que no diagrama de sequência também é possível trabalhar com fragmentos combinados ou fragmentos de interação? Esses fragmentos possibilitam o alinhamento de interações, ou seja, cada fragmento representa uma interação independente (GUEDES,

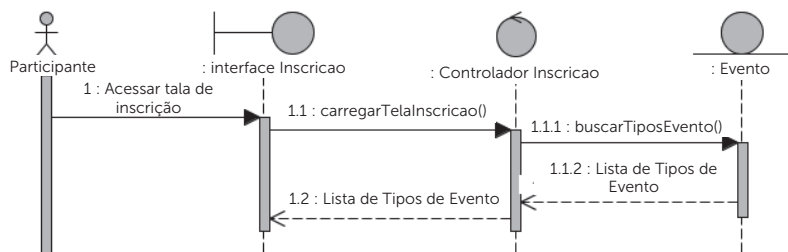
2011). Dentro de um diagrama de sequência podemos ter nenhum ou muitos fragmentos. A orientação da UML é que seja colocado apenas se de fato contribuir para o entendimento do diagrama, caso contrário deixará o diagrama poluído.

Para facilitar o entendimento, podemos modelar o diagrama de sequência utilizando alguns estereótipos do diagrama de classe, tais como: <<boundary>>, <<control>> e <<entity>> (GUEDES, 2011).

- O <<boundary>> é utilizado para representar a interface do sistema, sendo a comunicação entre os atores externos e as demais classes.
- O <<entity>> serve para representar que as classes do sistema também são entidades, seja persistente ou transiente.
- O <<control>> é utilizado para servir de intermédio entre as interfaces <<boundary>> e as <<entity>>. Nelas tratamos das regras de negócio e do fluxo da aplicação.

Na Figura 3.15, faremos a mesma representação da Figura 3.14, da mensagem "1 a 1.2". Nela utilizaremos os estereótipos mostrados anteriormente.

Figura 3.15 | Diagrama de sequência com estereótipo



Fonte: elaborada pela autora.

Atividades de aprendizagem

1. Observando nosso estudo de caso “Controle de Eventos de Extensão Universitária” e o diagrama de sequência da Figura 3.14, represente os demais itens 2, 3 e 4 com os estereótipos abordados, assim como modelamos na Figura 3.15.

2. Na construção do diagrama de sequência, caso seja identificada uma nova operação que não estava no diagrama de classe ou que não há necessidade em ter determinada operação no diagrama de classe, como devemos proceder?

Fique ligado

Nesta seção, foram apresentados:

- informações sobre o diagrama de sequência;
- as possíveis trocas de mensagens;
- a organização do diagrama, quanto ao uso do ator, objetos, linha de vida, foco de controle e mensagens;
- estereótipos do diagrama de classe aplicados também no diagrama de sequência.

Como foi o estudo desta seção? Compartilhe com o seu professor!

Seção 3

Diagrama de Colaboração: conceitos, componentes, notação e construção

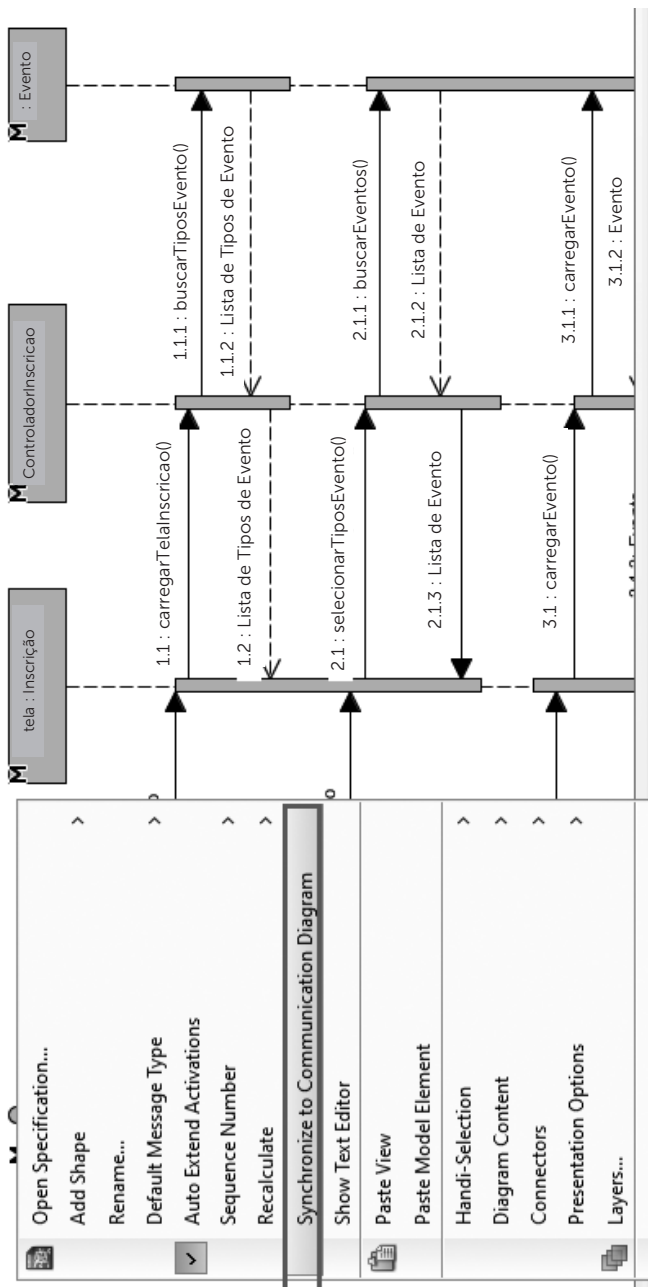
Introdução à seção

Caro(a) aluno(a), nesta seção, você conhecerá o diagrama de colaboração, que assim como o diagrama de sequência, representa a troca de mensagens entre objetos, porém, no diagrama de colaboração o tempo não é modelado explicitamente, uma vez que a ordem das mensagens é definida através de enumeração. Vamos verificar seus conceitos, componentes, notação e construção.

O diagrama de colaboração, após a versão 1.5 da UML, ficou conhecido como diagrama de comunicação. Esse diagrama é uma alternativa ao diagrama de sequência. Nele, representamos as ações por números e no diagrama de sequência por linhas verticais.

Ele tem foco nas mensagens enviadas entre objetos que estão relacionados e precisam ter consistência com o diagrama de classe. Utiliza a mesma notação do diagrama de sequência para os atores, objetos, mensagens, inclusive os estereótipos abordados na Figura 3.15. Por meio da ferramenta CASE, abordada no livro *Visual Paradigm*, é possível fazer a geração do diagrama de colaboração a partir do diagrama de sequência, conforme a Figura 3.16, basta clicar com o botão direito do mouse no cenário do diagrama de sequência, clicar na opção “Sincronize to Communication Diagram” e o arquivo será criado.

Figura 3.16 | Diagrama de sequência gerando diagrama de colaboração



Fonte: elaborada pela autora.

Para representar o fluxo de atividades desse diagrama, existem duas formas:

- Simples, representado por números inteiros, por exemplo, 1,2,3.
- Composta, representada por valores em casas decimais, por exemplo 1.1, 2.2.2.

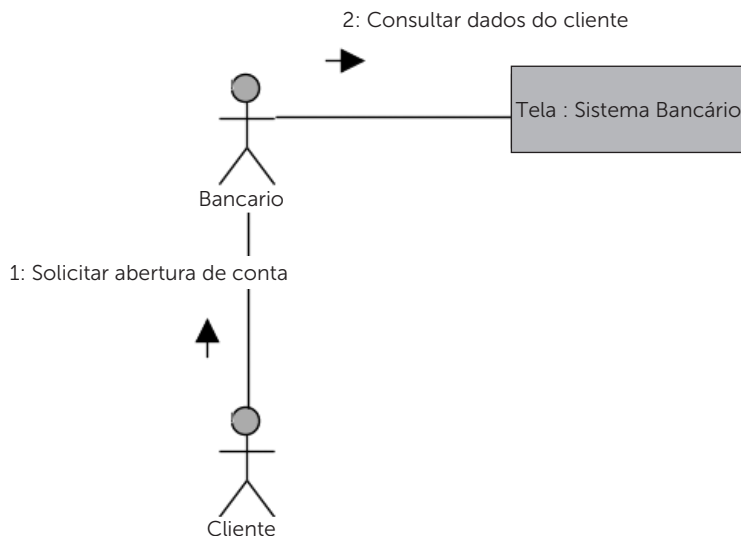
O diagrama de colaboração mostra a interação do usuário com o software em determinado ponto. Ele pode ser considerado um diagrama simples, pois não possui fragmentos combinados, assim como o diagrama de sequência. Utilizamos esse diagrama quando precisamos ver o comportamento de vários objetos dentro de um caso de uso, observando a conexão entre eles.

Podemos dizer que o principal objetivo dele é o vínculo, ou seja, são as ligações existentes entre os objetos envolvidos no processo. Esse vínculo é representado por uma linha unindo os objetos.

Ao compararmos o diagrama de colaboração com o diagrama de sequência, temos como vantagem que, normalmente, permite construir desenhos mais legíveis, porém sua desvantagem é que não há como saber a ordem de envio das mensagens, a não ser pelas expressões de sequência. A direção de envio da mensagem é indicada por uma seta próxima ao seu rótulo.

Ele é formado por objetos (representados por retângulos), interações entre objetos (linhas ligando objetos) e mensagens (textos e setas). O retorno é representado por uma seta no formato de “pé de galinha”, porém não tracejado.

Figura 3.17 | Diagrama de Colaboração do diagrama de Sequência da Figura 3.10 (Sistema bancário)

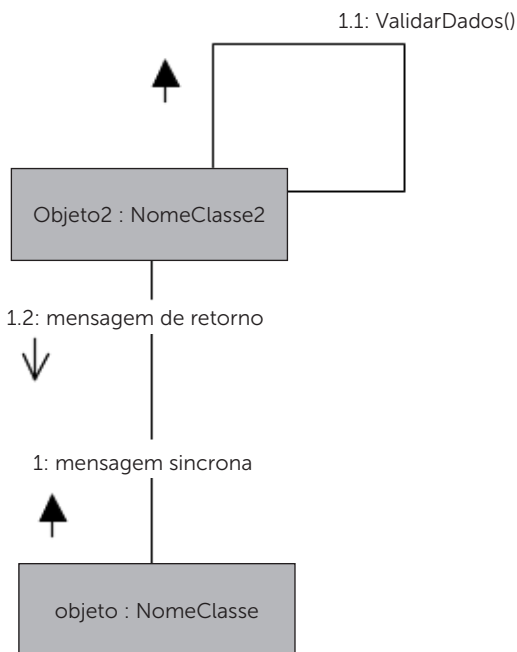


Fonte: elaborada pela autora.

Nesta Figura 3.17, o ator “Cliente” inicia a comunicação com outro “Bancario” e, nessa comunicação, é solicitada a abertura de uma conta ainda na interface de um sistema, ou seja, não representa um disparo de uma operação.

Assim como no diagrama de sequência, nesse diagrama também pode ocorrer de um objeto enviar uma mensagem para si mesmo, esse fenômeno tem o nome de autochamada ou mensagem reflexiva. Ele indica que o objeto daquela determinada tarefa deve iniciar e completar o serviço solicitado.

Figura 3.18 | Diagrama de colaboração do diagrama de sequência da Figura 3.9 (Mensagem reflexiva)



Fonte: elaborada pela autora.

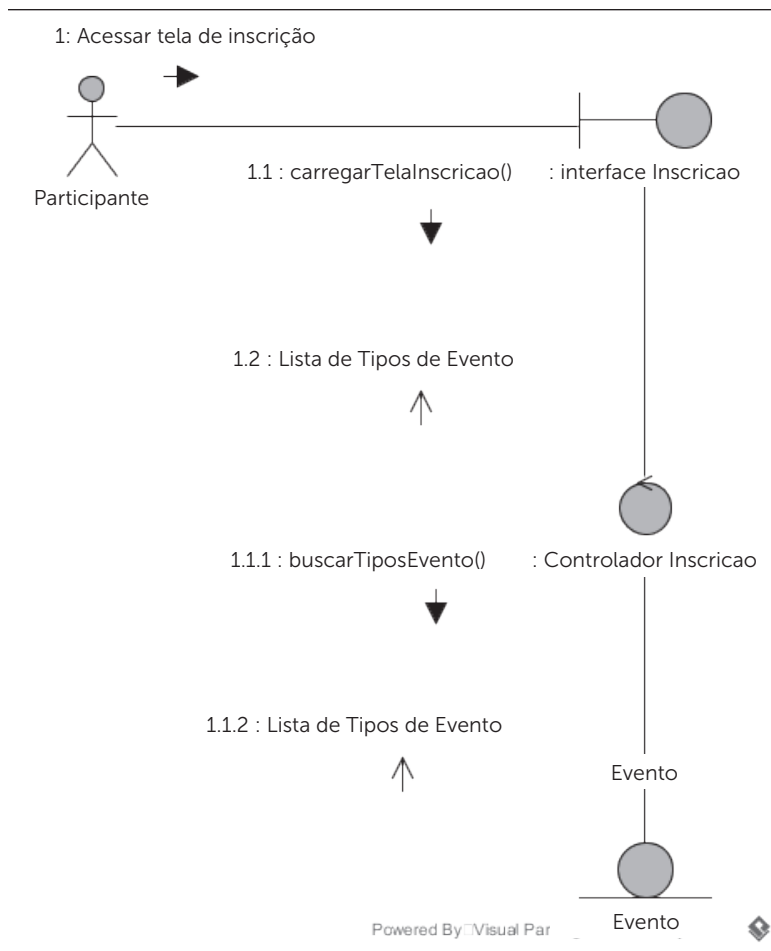
Podemos citar como diferença entre esse diagrama e o diagrama de sequência o fato de que aquele seria mais indicado no desenvolvimento do projeto, já este seria mais indicado no início da análise.

O diagrama de colaboração fornece uma visão geral do cenário, ajudando a identificar as interações entre objetos, e tem ênfase na estrutura, a ordem no tempo está implícita e o relacionamento entre os objetos envolvidos é mais visual.

Já o diagrama de sequência mostra o cenário dando ênfase na ordem cronológica da troca de mensagens entre os envolvidos e tem ênfase na progressão. A sequência do tempo é visual, já o relacionamento entre os objetos envolvidos está implícito.

Agora, com base em nosso estudo de caso “Controle de Eventos de Extensão Universitária” e no diagrama de sequência da Figura 3.15, representando de forma simplificada alguns estereótipos, vamos criar o diagrama de colaboração.

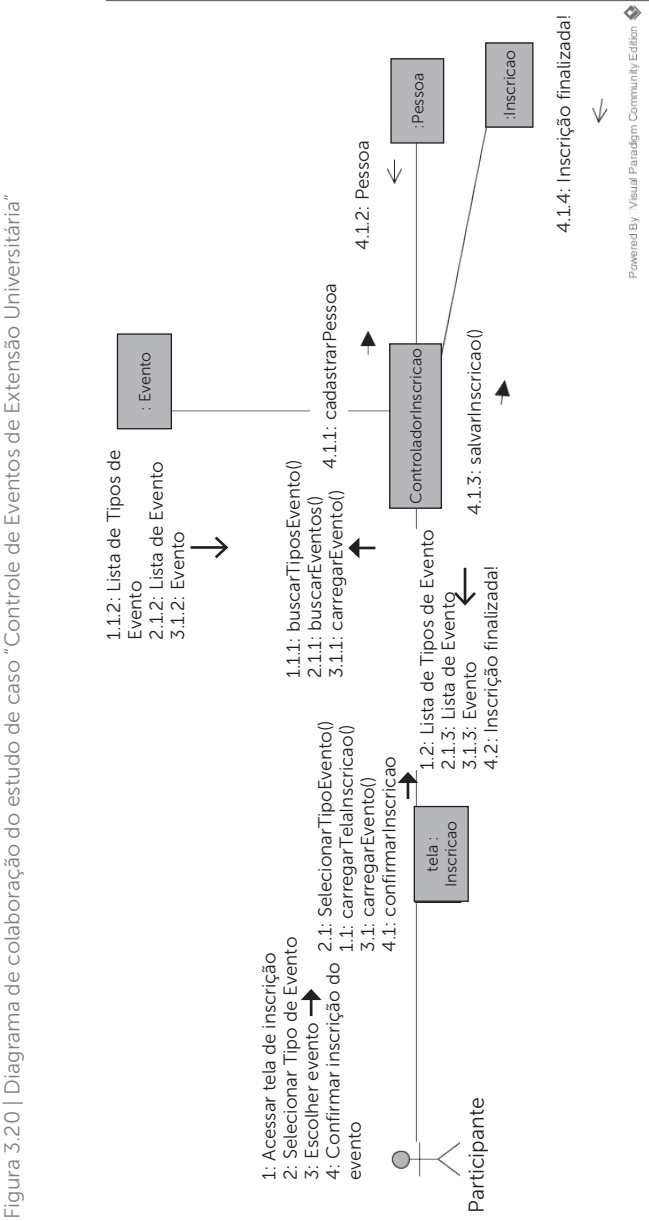
Figura 3.19 | Diagrama de colaboração do diagrama de sequência da Figura 3.15



Fonte: elaborada pela autora.

Como é possível observar, os estereótipos representados nesse diagrama também terão a mesma notação da Figura 3.15 dentro da notação do diagrama de colaboração.

Veja, agora, na Figura 3.20, a representação completa do diagrama de colaboração do estudo de caso, conforme o diagrama da Figura 3.14.



Fonte: elaborada pela autora.



Questão para reflexão

Com base no que você aprendeu até o momento, acredita que em todos os projetos seria necessário usar todos os diagramas da UML? Quais critérios você usaria para priorizar os diagramas?

Atividades de aprendizagem

1. O diagrama de comunicação pode ser considerado um diagrama de interação. Por quê?

- a) Mostra a troca de mensagens entre os objetos.
- b) Mostra a troca de mensagens entre o ator com o caso de uso.
- c) Mostra a troca de mensagens entre os diagramas de sequência e de comunicação.
- d) Mostra a interação entre todos os diagramas da UML.
- e) Mostra a interação entre todos os diagramas estruturais e comportamentais.

2. O diagrama de colaboração teve o nome alterado na versão 1.5 da UML para:

- a) Diagrama de interatividade.
- b) Diagrama de organização.
- c) Diagrama de empregabilidade.
- d) Diagrama de comunicação.
- e) Diagrama de estado.

Fique ligado

Nesta seção, foram apresentados:

- Informações sobre o diagrama de sequência novamente, porém, com a interação com o diagrama de comunicação.
- Como podemos representar o mesmo diagrama com visões diferentes.
- Como fazer a conversão (sincronização) do diagrama de sequência para o diagrama de colaboração utilizando a ferramenta Visual Paradigm.

Como foi seu estudo desta seção? Compartilhe com o seu professor!

Para concluir o estudo da unidade

Vimos, nesta unidade, os diagramas de interação, como diagramas de sequência e colaboração, os quais representam troca de mensagens entre objetos. No início da unidade, colocamos uma questão sobre qual desses diagramas utilizar. Vimos, durante a leitura, que ambos têm sua importância, eles nos mostram a interação dos envolvidos no sistema. Observamos que eles devem estar em concordância com outros diagramas, como o de classe e de caso de uso, e cabe ao analista optar pela utilização desses diagramas.

Podemos dizer que o diagrama de sequência é melhor usado quando a ênfase está no decorrer do tempo, já o diagrama de colaboração quando a ênfase estiver no contexto do sistema.

Utilizamos o diagrama de interação quando queremos observar o comportamento de vários objetos dentro de um mesmo caso de uso, ele nos mostra a comunicação entre os objetos.

Os diagramas de interação modelam os aspectos dinâmicos do sistema, eles contêm os objetos, vínculos e mensagens.

Analisamos o diagrama de sequência, que tem ênfase na ordem das mensagens trocadas pelos objetos, e possui como elementos os objetos, as mensagens e a linha da vida.

Vimos que o diagrama de colaboração mostra a interação entre os objetos e seus vínculos, além de ter uma visão sobre um conjunto de elementos relacionados para um propósito específico.

Até o próximo capítulo!



Questão para reflexão

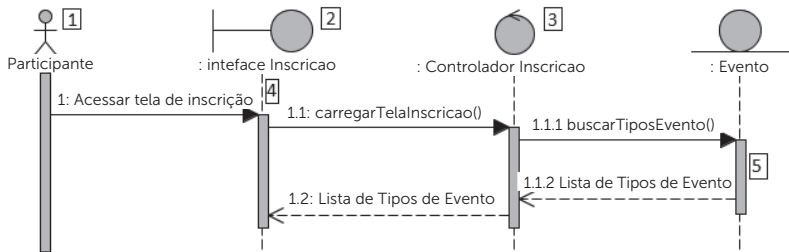
Com base nos diagramas da UML, os analistas conseguem representar o escopo do projeto e auxiliam os programadores na implementação dele de forma mais rápida e confiável. Por que será que mesmo assim os projetos ultrapassam prazos e custos?

Atividades de aprendizagem da unidade

1. O diagrama de sequência é responsável por:

- Verificar somente o fluxo de envio de mensagens que são realizadas pelos objetos para que uma ação seja executada.
- Verificar o fluxo de envio e recebimento de mensagens que são realizadas pelos objetos para que uma ação não seja executada.
- Verificar o fluxo de envio e recebimento de mensagens que são realizadas pelos objetos para que uma ação seja executada.
- Descrever os tipos de objetos do sistema e seu relacionamentos.
- Modelar a arquitetura física de um sistema.

2. Com base na Figura 3.15, que representa o diagrama de sequência do estudo de caso "Controle de Eventos de Extensão Universitária", alguns componentes dessa figura foram numerados, conforme imagem a seguir:



Análise as alternativas, em que cada numeração tem a informação do que ela representa.

I - 1= Ator, 2=boundary (representa a classe controladora), 5=linha de vida.

II - 1= Ator, 3 = entity (representa a tela do sistema que permitirá a comunicação visual com o usuário), 5=foco de controle.

III - 3 = entity (representa a tela do sistema que permitirá a comunicação visual com o usuário), 4=linha de vida, 5=foco de controle.

IV - 1=Ator, 4=linha de vida, 5=foco de controle.

Assinale a alternativa correta:

- Apenas o item I está correto.
- Apenas os itens II e III estão corretos.
- Apenas os itens II e IV estão corretos.
- Apenas o item III está correto.
- Apenas o item IV está correto.

3. Qual é o nome dos estereótipos da classe apresentados nesta Unidade 3 nos diagramas de sequência e colaboração?

- a) Apenas <<boundary>>.
- b) Apenas <<entity>>.
- c) Apenas <<controladoria>>.
- d) Apenas <<boundary>> e <<controladoria>>.
- e) <<boundary>>, <<entity>>, <<control>>.

4. Na Figura 3.14, temos a representação do diagrama de sequência do estudo de caso “Controle de Eventos de Extensão Universitária”. Nele temos a classe controladora “ControladorInscricao” e o objeto da classe “:Evento”. Com base nesse relacionamento, quais são as operações identificadas?

- a) buscarTipoEvento(), Lista de Tipos de Evento, buscarEventos(), Lista de eventos, carregarEvento(), Evento.
- b) Apenas buscarTipoEvento(), Lista de Tipos de Evento, buscarEventos() e carregarEvento().
- c) Apenas buscarTipoEvento() e Lista de Tipos de Evento, Lista de eventos, Evento.
- d) Apenas Lista de Tipos de Evento, Lista de eventos e Evento.
- e) Apenas buscarTipoEvento(), buscarEventos(), carregarEvento().

5. Na Figura 3.14, temos a representação do diagrama de sequência do estudo de caso “Controle de Eventos de Extensão Universitária”. Nele temos a classe controladora “ControladorInscricao” e o objeto da classe “:Evento”. Com base nesse relacionamento, quais são as mensagens de retorno identificadas?

- a) buscarTipoEvento(), Lista de Tipos de Evento, buscarEventos() e Lista de eventos.
- b) Apenas buscarTipoEvento(), Lista de Tipos de Evento e buscarEventos().
- c) Apenas buscarTipoEvento() e Lista de Tipos de Evento e Lista de eventos.
- d) Apenas Lista de Tipos de Evento e Lista de eventos e Evento.
- e) Apenas buscarTipoEvento(), buscarEventos() e carregarEvento().

Referências

BEZERRA, E. **UML** - Princípios de Análise e Projeto de Sistemas. 3. ed. Rio de Janeiro: Campus, 2015.

CHEN, Peter. **Active conceptual modeling of learning**: Next Generation Learning-Base System Development. com Leah Y. Wong (Eds.). Berlin, Germany: Ed. Verlag-Springer, 2007.

FOWLER, Martin; SCOTT, Kendal. **UML essencial**. Porto Alegre: Bookman, 2000.

GUEDES, G.T. **UML 2**: Uma abordagem prática. 2. ed. São Paulo: Novatec Editora, 2011.

MELO, A. C. **Desenvolvendo Aplicações com UML 2.0 do Conceitual à Implementação**. 2. ed. Rio de Janeiro: Brasport, 2002.

JACOBSON, Ivar. **Object-orientad Software Engineering**. Redwood City, CA, USA: Addison Wesley Longman, 1995.

LARMAN, Craig. **Utilizando UML e padrões**. 3. ed. Porto Alegre: Bookman, 2007

RUMBAUGH, James et al. **Modelagem e projetos baseados em objetos**. Rio de Janeiro: Editora Campus, 1994.

Consistência e interação entre os diagramas da UML

Iolanda Cláudia Sanches Catarino

Objetivos de aprendizagem

- Compreender a transição da modelagem entre as atividades do processo de desenvolvimento do software.
- Compreender a modelagem de análise de requisitos com os diagramas da UML.
- Compreender a modelagem de análise com os diagramas da UML.

Seção 1 | Compreendendo a transição da modelagem entre as atividades

Nesta seção, é possível obter o entendimento sobre a transição da modelagem entre as atividades do processo de desenvolvimento do software, a partir dos principais diagramas estruturais, comportamentais e de interação da UML, conforme as atividades de Análise de Requisitos, Análise e Projeto do Processo Unificado. Ao final da seção, indicamos alguns links para o aluno, a fim de ampliar o seu conhecimento na modelagem de análise e projeto de sistemas orientado a objetos com UML.

Seção 2 | Modelagem da atividade de Análise de Requisitos com UML

Nesta seção, é possível compreender a modelagem da atividade de Análise de Requisitos de um estudo de caso proposto, considerando a adoção do Diagrama de Atividades para representar os processos de negócio e seus fluxos, e do Diagrama de Casos de Uso para especificar os requisitos funcionais identificados nessa etapa. Ao final da seção, alguns links também estão disponíveis para consulta e ampliação do conhecimento sobre a atividade de Análise de Requisitos.

Seção 3 | Modelagem da atividade de Análise com UML

Nesta seção, apresentamos a modelagem da atividade de Análise do estudo de caso proposto, começando com a modelagem dinâmica inicial do sistema, a partir da especificação do Diagrama de Casos de Uso em duas visões, sendo a primeira a representação dos casos de uso base do sistema; e a segunda, a representação dos casos de uso do sistema com seus principais relacionamentos. Na sequência, é possível compreender a modelagem estática do sistema, a partir da especificação do Diagrama de Classes, Diagrama de Objetos e Diagrama de Estruturas Compostas. Por fim, a modelagem dinâmica complementar de análise, a partir da exemplificação do Diagrama de Atividades, Diagrama de Máquina de Estados e o Diagrama de Sequência. Ao final da seção, alguns links também estão disponíveis para consulta e ampliação do conhecimento sobre os diagramas trabalhados.

Introdução à unidade

Caro(a) aluno(a), nesta unidade apresentamos uma visão sobre a consistência e interação entre os diagramas estruturais, comportamentais e de interação da UML aplicados à modelagem das atividades de Análise de Requisitos, Análise e Projeto durante o desenvolvimento de um sistema de software, considerando o Processo Unificado entre os Modelos de Engenharia de Software.

Cada processo de desenvolvimento de software regido pela Engenharia de Software tem suas particularidades em relação às características e ao modo de encadear a realização das atividades, fases ou etapas. Entretanto, os vários processos de desenvolvimento abrangem as atividades básicas de Análise de Requisitos, Análise, Projeto, Implementação, Testes e Implantação, consistindo a modelagem do software nas atividades iniciais do processo - Análise de Requisitos, Análise e Projeto -, no qual a consistência e interação da especificação da modelagem do software assegurará à eficiência e eficácia do sistema para os usuários e para a organização.

A atividade de Análise de Requisitos (também conhecida como elicitación de requisitos, levantamento de dados etc.), corresponde à etapa de compreensão do problema, domínio e contexto aplicado ao desenvolvimento do sistema, com o objetivo de que os usuários e desenvolvedores tenham a mesma compreensão do problema a ser resolvido e das necessidades dos usuários que farão uso do novo sistema. As necessidades dos usuários são denominadas de requisitos identificados a partir do domínio. O domínio do sistema, também chamado de domínio do problema ou domínio do negócio, corresponde à parte de mundo real, área de conhecimento, nível organizacional, área funcional da empresa ou segmento para que o sistema será desenvolvido, o qual define e delimita o contexto da fronteira do sistema.

A atividade de Análise consiste em abstrair, definir e documentar o que o sistema deve fazer, considerando uma visão lógica do negócio, independente das tecnologias que serão adotadas para implementação do sistema e demais detalhes de desenvolvimento. Nesta etapa, o analista de sistemas define uma estratégia de

solução para o problema, realizando um estudo detalhado dos requisitos identificados na atividade de Análise de Requisitos ou Levantamento de Dados, bem como analisa a necessidade de dividir o sistema em partes (módulos), para assim definir as suas funcionalidades e interações.

A atividade de Projeto consiste em documentar como o sistema deve fazer para atender os requisitos definidos na atividade anterior, considerando os recursos tecnológicos existentes, ou seja, definem-se os aspectos físicos do sistema e dependentes de implementação, porém consistentes com a atividade de análise. Exemplos de aspectos tecnológicos a serem considerados nesta etapa, conforme será adotado na implementação do sistema: definição da arquitetura do sistema, linguagens de programação, sistema gerenciador de banco de dados, padrão de interface gráfica e demais padrões de programação (*design patterns*) condizentes com a metodologia que a empresa adota, ou seja, a equipe de profissionais desenvolvedores de software.

A atividade de Implementação corresponde à etapa de codificação do sistema, ou seja, ocorre a tradução da especificação obtida na atividade de projeto em código executável, mediante o uso de uma ou mais linguagens de programação. Para implementação das funcionalidades do sistema, pode-se também reutilizar componentes de software, bibliotecas de classes e adotar *frameworks* e *design patterns* para agilizar a atividade. A reutilização é um princípio essencial na área de Engenharia de Software para garantir o aumento da produtividade e, com isso, reduzir esforços e custos durante o processo de desenvolvimento de softwares. Os componentes são partes operacionais de software desenvolvidas de forma a desempenhar totalmente suas funções, devendo ser empacotados a fim de prover conjuntos de serviços acessíveis apenas através de uma interface bem definida. Um *framework* proporciona funcionalidade pré-fabricada e extensibilidade das suas classes para as aplicações a serem construídas. O desenvolvedor precisa saber como usá-lo, não requerendo conhecimento detalhado de como foi projetado.

A atividade de Testes corresponde à execução de testes com as funcionalidades do software, com os componentes e com o software para verificar e validar o nível de confiabilidade do

programa desenvolvido, com o objetivo de assegurar e garantir que o software está pronto para ser entregue e implantado. Os processos de verificação e validação devem abranger técnicas de inspeções e de testes de software para consistir os requisitos funcionais e não funcionais do sistema.

A atividade de Implantação corresponde à etapa de entrega do software, na qual o sistema é preparado, distribuído e instalado de forma planejada, no ambiente do usuário, sendo todo o processo orientado e acompanhado pela equipe técnica responsável pelo processo de implantação.

Diante do contexto das atividades básicas de um processo de desenvolvimento, nesta unidade, vamos explorar a modelagem de Análise de Requisitos e Análise de um estudo de caso, apresentando a especificação das principais técnicas de modelagem da UML, aplicadas a cada atividade, destacando a consistência e interação entre os diagramas utilizados.

Iniciamos a Seção 4.1 reforçando alguns conceitos sobre o Processo Unificado para destacarmos os pontos essenciais da evolução e detalhamento dos diagramas da UML durante a execução das atividades perante as fases.

Na sequência, na Seção 4.2, apresentamos a modelagem da atividade de Análise de Requisitos do estudo de caso proposto, considerando a adoção do Diagrama de Atividades para representar os processos de negócio e seus fluxos, e do Diagrama de Casos de Uso para especificar os requisitos funcionais identificados nessa atividade.

Por fim, na Seção 4.3, apresentamos a modelagem estática e dinâmica da atividade de Análise, a partir da especificação do Diagrama de Casos de Uso, do Diagrama de Classes, do Diagrama de Objetos, do Diagrama de Estruturas Compostas, do Diagrama de Atividades, do Diagrama de Máquina de Estados e do Diagrama de Sequência.

Seção 1

Compreendendo a transição da modelagem entre as atividades

Introdução à seção

Caro(a) aluno(a), nesta seção, apresentaremos uma visão geral sobre as fases e atividades do Processo Unificado, destacando os pontos essenciais da evolução e detalhamento dos diagramas da UML para garantir a consistência e interação da modelagem do sistema com os requisitos implementados.

A consistência e interação entre os diagramas que integram a documentação do sistema devem ser asseguradas conforme ocorre a evolução da modelagem do software, a partir do desenvolvimento de ciclos de vida incremental e iterativo de um sistema.

Com a abordagem de desenvolvimento de software de forma incremental e iterativa, é importante que se definam mecanismos para dividir os requisitos do sistema em partes e alocá-los em um ciclo de desenvolvimento, atentando-se ao grau de importância e risco atribuído a cada requisito. Segundo Bezerra (2007, p. 37), “o desenvolvimento evolui em versões, ao longo da construção incremental e iterativa de novas funcionalidades até que o sistema completo esteja construído”.

A UML não se limita a um modelo de processo de desenvolvimento da Engenharia de Software e nem se vincula a uma etapa do processo de desenvolvimento, mas se apoia no desenvolvimento incremental, através de modelos que evoluem com a inclusão de novos detalhes, aplicando um refinamento na modelagem do sistema (CATARINO, 2012).

Vários autores destacam vantagens e desvantagens nessa abordagem de desenvolvimento incremental e iterativo. Entre as vantagens, destaca-se a participação ativa dos usuários, interagindo no processo de verificação e validação dos requisitos, e que os riscos do projeto diminuem e podem ser mais bem gerenciados, priorizando a implementação dos requisitos mais arriscados. Entre

as desvantagens, destaca-se o grau demasiado de expectativa gerada pelos usuários que acompanham o processo de verificação e validação dos requisitos, e uma maior atenção e dedicação do gerente do projeto em gerenciar vários ciclos de desenvolvimento de um sistema durante a sua produção.

1.1 Processo Unificado

A organização geral do ciclo de vida de processo incremental e iterativo consiste nas dimensões temporal e de atividades (ou de fluxos de trabalho). A dimensão temporal é estruturada em fases, sendo que em cada fase ocorre uma ou mais iterações, com duração de, em média, duas a seis semanas, e ao final de cada iteração é produzido um incremento que pode ser liberado para os usuários ou não. A dimensão de atividades compreende as realizações durante a iteração de uma fase (BEZERRA, 2007).

Segundo Booch, Rumbaugh e Jacobson (2000, p. 34), as características de um processo iterativo e incremental referem-se a:



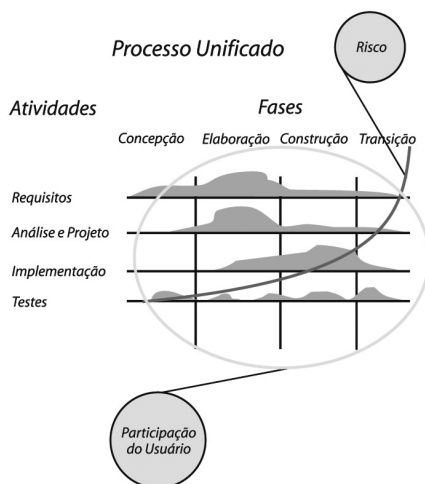
Um processo iterativo é aquele que envolve o gerenciamento de sequências de versões executáveis. Um processo incremental é aquele que envolve a integração contínua da arquitetura do sistema para a produção dessas versões, de maneira que cada nova versão incorpora os aprimoramentos incrementais em relação às demais. Em conjunto, um processo iterativo e incremental é orientado a riscos, ou seja, cada nova versão tem como foco atacar e reduzir os riscos mais significativos para o sucesso do projeto.

Seguindo a abordagem de desenvolvimento incremental e iterativo, surgiu o *Unified Process* (BOOCH; RUMBAUGH; JACOBSON, 2000), identificado como Processo Unificado, que apoia o desenvolvimento de softwares orientados a objetos, fornecendo uma forma sistemática e evolutiva de modelar sistemas de softwares com as técnicas de modelagem da UML. Segundo Catarino (2012, p. 10), o Processo Unificado:

Consiste na repetição de uma série de ciclos durante o processo de desenvolvimento de um sistema, permitindo uma gerência mais efetiva de projetos complexos. Cada ciclo é concluído com uma versão do produto pronta para distribuição e é subdividido em quatro fases sucessivas identificadas como: Concepção, Elaboração, Construção e Transição; sendo que cada fase, por sua vez, é subdividida em iterações que passam pelas seguintes atividades do processo: Requisitos, Análise e Projeto, Implementação e Testes.

As fases representam os aspectos dinâmicos do processo desenvolvimento e tratam a dimensão do tempo de execução de cada fase, ou seja, o intervalo de tempo decorrido entre dois pontos do processo quando um conjunto bem-sucedido de objetivos é alcançado, produzindo os artefatos do software (modelagem do software especificada no formato de diagramas), que são concluídos e, assim, decisões são tomadas para prosseguir para a próxima fase. As atividades são executadas de forma incremental e evolutiva, representando os aspectos estáticos do processo. A ênfase sobre as várias atividades muda em cada fase do processo ao longo do tempo, como mostra a Figura 4.1 a seguir.

Figura 4.1 | Processo Unificado



Fonte: Medeiros (2004, p. 16).

A fase de *Concepção* inicia-se com a definição do domínio do problema, descrevendo a delimitação do escopo e contexto do projeto e a elaboração de um planejamento para desenvolver o sistema. É feita a identificação dos atores que irão interagir com o sistema, definindo a natureza dessa interação em uma perspectiva de alto nível, destacando os principais requisitos funcionais do sistema, bem como a identificação dos requisitos não funcionais do sistema e o estudo sobre a continuidade e viabilidade do desenvolvimento do projeto (CATARINO, 2012).

Na fase de *Elaboração*, define-se, analisa e especifica os requisitos funcionais do sistema, descrevendo e enfatizando o comportamento dinâmico dos requisitos, além de estabelecer a arquitetura e os mecanismos para tratar aspectos abrangentes do sistema na atividade de Projeto, conforme o domínio do problema. Segundo Booch, Rumbaugh e Jacobson (2000, p. 445), “no final da fase de elaboração, você examina o escopo e os objetivos detalhados do sistema, a escolha de arquitetura e a solução para os principais riscos, além de decidir se deve prosseguir com a construção”.

Na fase de *Construção*, desenvolve-se, de maneira iterativa e incremental, um produto completo do software, concluindo a implementação, verificação e validação do software. Decide-se também se o software, os ambientes e usuários estão prontos para se tornarem operacionais (BOOCH; RUMBAUGH; JACOBSON, 2000).

A fase de *Transição* refere-se à entrega do software aos usuários do sistema, muitas vezes disponibiliza-se uma primeira versão e, depois, a partir da indicação de ajustes e adequações identificadas pelos usuários, é necessário corrigir alguns problemas para, enfim, disponibilizar a versão de produção do sistema.

Em cada fase do Processo Unificado, diferentes artefatos de software são produzidos ou refinados à medida que as iterações do processo são realizadas em função da evolução do desenvolvimento do software.

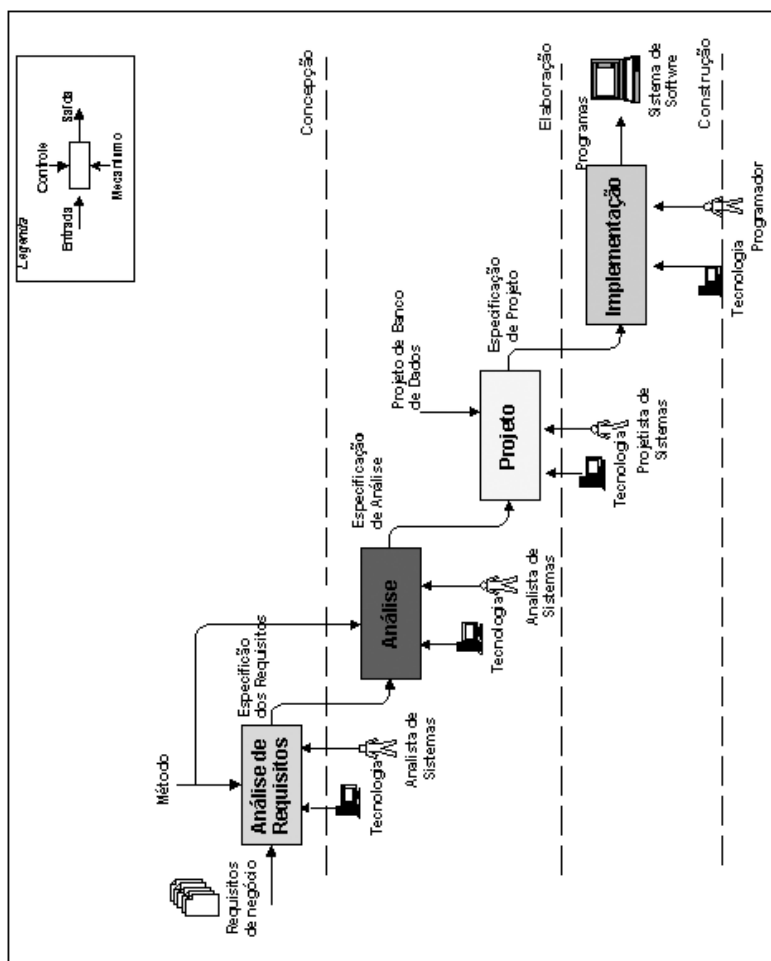
1.2 Detalhamento do Processo de Desenvolvimento de Software

Desenvolver sistemas complexos de software são tarefas que requerem a visualização desses sistemas sob várias perspectivas. Considerando o ciclo de desenvolvimento incremental e iterativo do Processo Unificado, os fluxos de trabalho de cada atividade indicam variações de graus do foco ao longo do tempo. Conforme ilustra a Figura 4.2, na fase de Elaboração, evidencia-se a realização das atividades de Análise e Projeto, sendo que os artefatos construídos na atividade de Análise (técnicas de modelagem especificadas) são utilizados como entrada para iniciar a atividade de Projeto.

No desenvolvimento de sistemas orientados a objetos utilizando-se da UML, há uma quantidade significativa de técnicas de modelagem a serem adotadas em cada atividade, consistindo na modelagem do sistema. A modelagem de análise vai se transformando em projeto à medida que o desenvolvimento evolui, com isso, o modelo de casos de uso e o modelo de classes de análise, por exemplo, são dois artefatos construídos na análise e refinados no projeto posteriormente.

Em um processo de desenvolvimento iterativo, a análise precede o projeto, entretanto, a análise e o projeto podem acontecer simultaneamente. Segundo Bezerra (2007, p. 176), “os modelos especificados na análise esclarecem o problema a ser resolvido. No entanto, as perspectivas do sistema fornecidas por esses modelos não são suficientes para se ter uma visão completa do sistema para que a implementação comece”. Assim, o objetivo do projeto é definir alternativas que atendam aos requisitos funcionais, respeitando as restrições indicadas pelos requisitos não funcionais, de forma que os artefatos de software representem um alto nível de detalhamento que ampare a implementação do software.

Figura 4.2 | Ciclo de Desenvolvimento Incremental



Fonte: adaptada de Catarino (2012).

Analisando as treze técnicas de modelagem da UML e em concordância com vários autores da UML, em especial Bezerra (2007), recomendamos que a modelagem da atividade de Análise seja especificada a partir do Diagrama de Casos de Uso, representando-o apenas com os casos de uso base e na sequência seja identificada as classes de análise a partir da abstração dos casos de uso, elaborando o Diagrama de Classes sem se preocupar com os aspectos de linguagens de programação, considerando uma pequena parte dos aspectos dinâmicos do sistema. Com a

primeira versão do Diagrama de Classes, sugerimos elaborar a documentação dos casos de uso (formato descrito, numerado ou tabular), pois nesse momento já ficou mais fácil identificar os objetos que colaboram e participam durante a execução dos casos de uso.

Ainda como modelagem da atividade de Análise, sugerimos adotar o Diagrama de Estruturas Compostas, para representar a colaboração interna de classes que participam de uma funcionalidade do sistema; o Diagrama de Atividades, para especificar os casos de uso; e o Diagrama de Máquina de Estados, para representar as classes que possuem estados relevantes.

Na concepção de Bezerra (2207, 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”. Assim, na modelagem da atividade de Projeto, recomendamos evoluir o Diagrama de Casos de Uso detalhando-o com os relacionamentos de inclusão (<<Include>>) e extensão (<<Extend>>) entre os casos de uso.

No modelo de classes, deve-se refiná-las, definindo as classes de projeto ou novas classes (uma classe de análise pode resultar em várias classes de projeto) com o tipo de dados de cada atributo, correspondente à linguagem de programação que será adotada na implementação; ainda, detalhar as operações, estereotipar as classes, rever seus relacionamentos, definir classes abstratas, interfaces, padrões de projeto (*design patterns*) e demais detalhes pertinentes ao contexto do sistema e as tecnologias de desenvolvimento, obtendo um modelo suficientemente completo para que a implementação das classes possa ser feita a partir dele.

A modelagem da atividade de Projeto ainda é complementada com os diagramas comportamentais da UML, como o Diagrama de Sequência, que representa a sequência de eventos que ocorrem em um determinado processo, identificando quais métodos devem ser disparados entre os atores e objetos envolvidos e em que ordem (GUEDES, 2008). Outro aspecto importante na modelagem de projeto é relativo ao mapeamento de classes de objetos persistentes para tabelas de um sistema gerenciador de banco

de dados relacional, caso este seja utilizado como mecanismo de armazenamento persistente de dados para um sistema de software orientado a objetos. 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.

Ainda, na modelagem da atividade de Projeto, deve-se determinar o projeto da arquitetura a ser utilizada durante a implementação do sistema, definindo, por exemplo, subsistemas, interfaces, camadas de software, dependência entre subsistemas e componentes reutilizáveis, como *frameworks* e bibliotecas (BEZERRA, 2007). Outro aspecto é o projeto de interface gráfica com o usuário, pois as associações entre casos de uso e atores implicam a necessidade de interfaces gráficas (formulários, relatórios etc.) para que o usuário interaja com o sistema com utilidade, usabilidade e comunicabilidade, seguindo os princípios da Interação Humano-Computador (IHC).

Mesmo que a análise e o projeto trabalhem sobre o mesmo modelo, essas atividades se distinguem pela quantidade de detalhes que são incluídos nesse modelo. Bezerra (2007, p. 176) sintetiza as seguintes características da evolução da documentação de análise para projeto:



- a) Detalhamento dos aspectos dinâmicos do sistema;
- b) Refinamento dos aspectos estáticos e estruturais do sistema;
- c) Detalhamento da arquitetura do sistema;
- d) Definição das estratégias para armazenamento, gerenciamento e persistência dos dados manipulados pelo sistema;
- e) Realização do projeto da interface gráfica com o usuário;
- f) Definição dos algoritmos a serem utilizados na implementação.

Considerando o detalhamento dos aspectos dinâmicos na modelagem de projeto do sistema, é importante identificar a modelagem especificada em ferramentas CASE por versões ou visões, para assim facilitar a leitura e interpretação dos diagramas

de forma que seja possível consistir e validar a evolução da modelagem da atividade de Análise para a atividade de Projeto.



Questão para reflexão

Na UML, os agrupamentos que organizam um modelo (conjunto de diagramas) são chamados de pacotes. O Diagrama de Pacotes tem por objetivo mostrar a estrutura e dependência entre sistemas, subsistemas ou módulos, podendo ser utilizado de maneira independente ou associado a outros diagramas.

Para distinguir a evolução da modelagem entre as atividades de Análise e Projeto, pode-se adotar o Diagrama de Pacotes?



Para saber mais

Para conhecer mais sobre o processo incremental e iterativo e sobre a evolução da modelagem de análise para projeto, acesse o material indicado a seguir:

Livros:

- BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas com UML**. 2. ed. Rio de Janeiro: Elsevier, 2007.
- BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML: guia do usuário**. 2. ed. Rio de Janeiro: Elsevier, 2006.
- GUEDES, Gilleanes T. A. **UML: uma abordagem prática**. 3. ed. São Paulo: Novatec, 2008.

Fique ligado

Nesta seção, introduzimos a transição da modelagem orientada a objetos entre as atividades do processo de desenvolvimento, utilizando UML. Considerando os diversos elementos apresentados, é possível elaborar um primeiro estágio do Diagrama de Classes, aplicando e respeitando as regras de negócio do domínio do problema, porém priorizando os aspectos estáticos dos objetos do sistema e, posteriormente, avançando na modelagem da atividade de Projeto, deve-se refiná-lo, priorizando os aspectos dinâmicos dos objetos do sistema.

Atividades de aprendizagem

1. O produto final do processo de engenharia de requisitos deve ser um documento de especificação dos requisitos do sistema, que descreve, de forma estruturada, todos os requisitos que o software deverá possuir e que, de forma explícita, deve celebrar o compromisso mútuo de compreensão entre o usuário e o responsável por descrever os requisitos.

Sobre a classificação dos requisitos, analise os itens:

I. Os requisitos funcionais são declarações de todos os serviços e restrições que o software deve possuir, mediante as necessidades do cliente ou usuário.

II. A declaração de um requisito funcional deve ser realizada do ponto de vista do usuário, e deve determinar o que o software deve ou não fazer, não importando como será feito.

III. Os requisitos não funcionais são restrições sobre os serviços ou funções oferecidas pelo sistema. São exemplos de requisitos não funcionais: tempo de resposta, escalabilidade, usabilidade, padrões e segurança.

IV. Um requisito não funcional pode estar vinculado a um requisito funcional ou pode-se aplicar ao sistema como um todo.

Estão corretos os itens:

- a) I e II.
- b) II e III.
- c) III e IV.
- d) I, II e III.
- e) I, II, III e IV.

2. Antes de desenvolver qualquer sistema, é necessário entender o que ele irá realizar e como irá suportar as metas individuais ou de negócios. Isso envolve compreender o domínio da aplicação, o sistema operacional, as funcionalidades específicas requeridas pelos usuários e as características essenciais, como performance, segurança e dependabilidade (SOMMERVILLE, 2011).

Considerando a classificação dos requisitos, segundo Sommerville (2011), complete as lacunas dos seguintes parágrafos:

Os _____ são restrições sobre os serviços ou funções oferecidos pelo sistema. Os _____ são declarações de todos os serviços e restrições que o software deve possuir, mediante as necessidades do cliente ou usuário.

Assinale a alternativa correta que indica a sequência dos termos:

- a) Requisitos não funcionais. Requisitos de domínio.
- b) Requisitos não funcionais. Requisitos funcionais.
- c) Requisitos funcionais. Requisitos não funcionais.
- d) Requisitos de domínio. Requisitos funcionais.
- e) Requisitos de domínio. Requisitos não funcionais.

Seção 2

Modelagem da atividade de Análise de Requisitos com UML

Introdução à seção

Caro(a) aluno(a), nesta seção, apresentaremos a modelagem da atividade de Análise de Requisitos de um estudo de caso proposto, considerando a adoção do Diagrama de Atividades para representar os processos de negócio e seus fluxos, e do Diagrama de Casos de Uso para especificar os requisitos funcionais identificados nessa etapa.

O domínio de um sistema ou domínio do problema refere-se ao contexto para o qual é provida uma solução de uma determinada área de conhecimento ou segmento, sendo que o processo de reconhecimento do domínio caracteriza a definição de seu escopo e fronteira. O conteúdo de um domínio compreende fatos, regras, necessidades, conceitos, agentes etc.

Considerando um modelo de engenharia de software, como o Processo Unificado, a atividade de Análise de Requisitos, ou simplesmente Requisitos, visa identificar, analisar, definir e especificar os requisitos do sistema. Os requisitos de um sistema são as descrições do que o sistema deve fazer, os serviços a oferecer e as restrições a seu funcionamento (SOMMERVILLE, 2011).

Alguns estudos indicam que, para garantir a identificação dos requisitos de um sistema de software, é fundamental iniciar a compreensão do domínio do sistema com a modelagem organizacional, o qual tem o propósito de identificar, analisar, definir e especificar os objetivos e processos da organização.

A modelagem organizacional tem sido amplamente discutida e praticada. Refere-se a um processo que envolve a especificação de técnicas de modelagem, no formato de modelos, que representam o ambiente empresarial e seus componentes para auxiliar o entendimento dos aspectos políticos, sociais, técnicos, jurídicos e econômicos da organização, e com isso auxiliar e facilitar o levantamento de requisitos para iniciar o desenvolvimento de

um sistema de software, porque a transição entre negócios e tecnologias de informação tem sido o grande problema das organizações e da engenharia de sistemas.

O método *Enterprise Knowledge Development* (EKD) é uma abordagem para a modelagem organizacional que facilita a aquisição do conhecimento da estrutura organizacional e estratégica, visando descrever, a partir de visões, os objetivos, a política, a estrutura da organização, as questões de gestão de pessoas e as questões técnicas relacionadas aos objetos ou serviços da organização, além de auxiliar na identificação dos requisitos organizacionais, para melhorar a compreensão do domínio e a interação entre os usuários no desenvolvimento de sistemas de software, principalmente os sistemas de informação (CATARINO, 2012).

A Engenharia de Requisitos (*Requirements Engineering*) preocupa-se com o que deve ser feito, ou seja, a compreensão do problema, e não como fazer, considerando o domínio do sistema. A Engenharia de Requisitos é o processo de descobrir, analisar, documentar e verificar os serviços e restrições (SOMMERVILLE, 2011).

Esta seção não tem o objetivo de descrever a teoria sobre modelagem organizacional e o processo da engenharia de requisitos, porém contextualizamos esses conteúdos para evidenciar a importância da identificação dos requisitos funcionais e não funcionais de sistemas complexos de software, considerando as boas práticas de engenharia de software.

2.1 Estudo de Caso – Agência de Estágios

Para complementar o seu aprendizado, utilizaremos o estudo de caso descrito a seguir, o qual orientará na exemplificação dos diagramas da UML representados na unidade.

O estudo de caso – Agência de Estágios – refere-se a um sistema de informação simples, com poucas funcionalidades, mas a partir dele será possível aplicar as principais técnicas de modelagem da UML para documentar as atividades de Requisitos e Análise do Processo Unificado, e com isso consolidar a sua compreensão e o seu aprendizado sobre as técnicas de modelagem da UML.

Na descrição do estudo de caso, é possível identificar vários requisitos funcionais que estão explícitos na descrição, ou seja, as funcionalidades que o sistema deverá prover para os usuários. Alguns requisitos funcionais já descrevem as suas regras de negócio, por exemplo, para o requisito funcional “Cadastro de Candidato” estão evidentes algumas regras de negócio, sendo: um candidato tem que ser estudante com matrícula ativa em uma instituição de ensino; um candidato pode participar de várias entrevistas etc.

Descrição do estudo de caso – Agência de Estágios

Deseja-se desenvolver um sistema para uma agência de estágios que atua no ramo de seleção e contratação de candidatos para as empresas conveniadas. A agência precisa manter os cadastros de: candidato, empresa, instituição de ensino, entrevista e contrato.

O candidato é a pessoa que se cadastra na agência à procura de uma vaga de estágio. Um candidato pode fazer o seu cadastro na agência desde que seja estudante com vínculo ativo (matrícula) em uma instituição de ensino. Ele pode participar de várias entrevistas. Um candidato é descrito por: CPF, nome, endereço completo, idade, telefone, sexo, filiação, escolaridade, endereço eletrônico e instituição de ensino atual. O candidato também informa os dados do seu currículo.

A empresa é a pessoa jurídica que se cadastra na agência, ofertando vagas de estágio para estudantes. Uma empresa pode ofertar várias vagas de estágio. Uma empresa é descrita por: CNPJ, nome fantasia, razão social, inscrição estadual, ramo de atividade, endereço completo, telefone, fax, endereço eletrônico e contato. A vaga é retratada por uma descrição, cargo, quantidade, requisitos, salário, horário e período.

A agência cadastra todas as instituições de ensino médio e superior do município e da região. Uma instituição de ensino é cadastrada por: CNPJ, nome fantasia, razão social, endereço completo, telefone, fax, endereço eletrônico e contato.

A cada entrevista realizada com um candidato deve-se registrar: a data de realização, dados da empresa, dados da vaga, descrição da entrevista, nome do funcionário que realizou a entrevista, observação e situação (agendada, cancelada, realizada, aprovado ou reprovado). Para cada vaga pode-se realizar várias entrevistas.

Algumas entrevistas geram a contratação (contrato), envolvendo os dados: nº do contrato, candidato, empresa, cargo, data de início, data de término, carga horária semanal, horário e salário. Os contratos de estágio são emitidos e encaminhados para o candidato aprovado e para a empresa concedente do estágio.

O sistema deve disponibilizar a opção de um candidato realizar o seu cadastro e se inscrever para uma vaga via Web, e o sistema deve disponibilizar consultas e relatórios relacionados aos cadastros.

Fonte: elaborado pela autora (2017).

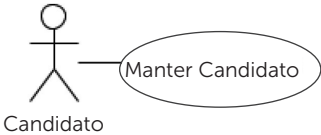
2.2 Atividade de Análise de Requisitos

A atividade de Requisitos, Análise de Requisitos ou também chamada de Elicitação de Requisitos, entre outros sinônimos, refere-se à primeira atividade da fase de Concepção, conforme o Processo Unificado. Sommerville (2011) classifica previamente os requisitos de um sistema em requisitos de usuário e requisitos de sistema, conforme exemplifica a Figura 4.3, sendo:

a) Requisitos de usuário: são declarações em uma linguagem natural com complemento de diagramas ou não, de quais serviços o sistema deverá fornecer a seus usuários e as restrições com as quais este deve operar;

b) Requisitos de sistema: são descrições mais detalhadas das funções, serviços e restrições operacionais do sistema de software, sendo que o documento de requisito de sistema (especificação funcional) deve definir exatamente o que deve ser implementado.

Figura 4.3 | Exemplo de Representação de Requisito de Usuário e de Sistema

Requisito de Usuário	Requisito de Sistema
 <p>Candidato</p>	<ol style="list-style-type: none"> 1. Candidato solicita seu cadastro. 2. Sistema exibe o cadastro de candidatos. 3. Candidato informa seus dados pessoais. 4. Sistema verifica que não existe candidato cadastrado. 5. Candidato informa demais dados. 6. Candidato confirma o seu cadastro. 7. Sistema registra o candidato.

Fonte: elaborada pela autora.

Os requisitos de sistema são classificados em dois tipos:

a) Requisitos Funcionais (RF): são declarações de serviços que o sistema deve fornecer, de como o sistema deve reagir a entradas específicas e de como o sistema deve se comportar em determinadas situações, ou seja, descrevem as funcionalidades do sistema.

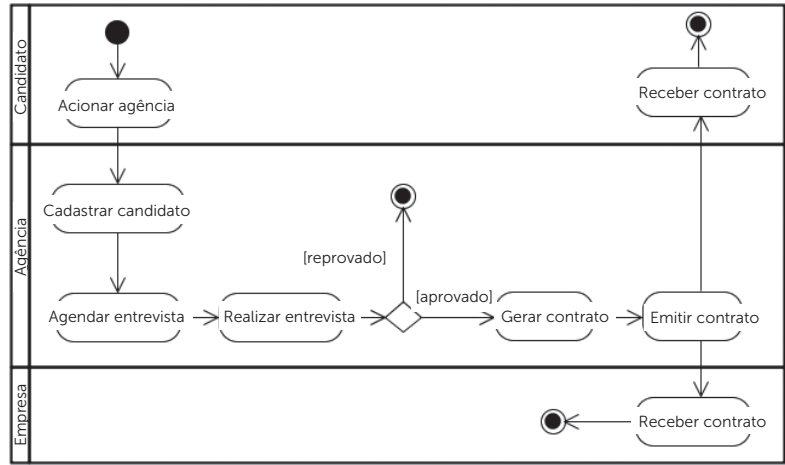
b) Requisitos Não Funcionais (RNF): expressam restrições aos serviços ou funções, ou qualidades específicas que o software deve atender. As restrições referem-se ao tempo, ao processo de desenvolvimento e, muitas vezes, são impostas pelas normas organizacionais, surgindo por meio das necessidades dos usuários, devido a restrições de orçamentos, políticas organizacionais etc. Um requisito não funcional pode estar vinculado a um requisito funcional ou pode-se aplicar ao sistema como um todo.

Das técnicas de modelagem da UML, podemos adotar na modelagem dos requisitos funcionais o Diagrama de Casos de Uso, o qual, posteriormente, guiará o processo de desenvolvimento; o Diagrama de Atividades, para representar o comportamento do sistema; e o Diagrama de Sequência, para especificar o cenário de cada funcionalidade identificada como requisito funcional.

Para iniciar a modelagem dos requisitos do estudo de caso, adotaremos o Diagrama de Atividades em compartimentos para compreender o domínio e contexto do sistema, representando o fluxo de atividades dos principais processos de negócio que demonstram o comportamento do sistema.

A Figura 4.4 ilustra o Diagrama de Atividades correspondente ao principal processo de negócio do estudo de caso, “Realizar Entrevista”, no formato de *swimlanes* (raias de natação), representando as atividades do processo de negócio distribuídas pelos vários agentes (Candidato, Agência e Empresa) que realizam as atividades. Cada compartimento contém atividades que são realizadas por um agente específico.

Figura 4.4 | Diagrama de atividades do processo realizar entrevista



Fonte: elaborada pela autora.

A partir da descrição do estudo de caso e da representação do processo “Realizar Entrevista” especificado através do Diagrama de Atividades, identificamos os principais requisitos funcionais relacionados no Quadro 4.1. No quadro, foram listados apenas os requisitos funcionais referentes aos cadastros ou controles. As consultas e os relatórios não foram listados, mas serão representados no Diagrama de Casos de Uso, na modelagem da próxima atividade.

Quadro 4.1 | Requisitos funcionais de cadastros

Nº	Requisito	Descrição
RF1	Cadastro de candidato	O sistema deve prover um cadastro de candidatos. Um candidato pode se cadastrar via Web também.
RF2	Cadastro de currículo de candidato	O sistema deve prover um cadastro do currículo dos candidatos.
RF3	Cadastro de empresa	O sistema deve prover um cadastro das empresas conveniadas com a agência de estágio.
RF4	Cadastro de vaga de estágio	O sistema deve prover um cadastro das vagas de estágio que as empresas ofertam.
RF5	Cadastro de instituição de ensino	O sistema deve prover um cadastro das instituições de ensino que os candidatos estão vinculados.

RF6	Cadastro de entrevista	O sistema deve prover um cadastro de entrevistas realizadas com os candidatos, referente às vagas de estágio. Cada entrevista deve ser gerenciada pela situação (agendada, cancelada, realizada, aprovado ou reprovado).
RF7	Geração do contrato de estágio	O sistema deve gerar um contrato de estágio para as entrevistas aprovadas.
RF8	Emissão do contrato de estágio	O sistema deve emitir cópias do contrato de estágio para o candidato e para a empresa.
RF9	Cadastro de inscrição para vaga	O sistema deve prover um cadastro de inscrição para vaga de estágio, via Web.

Fonte: elaborada pela autora.



Questão para reflexão

O Diagrama de Casos de Uso é um diagrama de fácil visualização e leitura para validar os requisitos funcionais do sistema junto aos usuários. O Diagrama de Casos de Uso é uma técnica de modelagem adotada para especificar as funcionalidades do sistema, documentando-as, principalmente, nas atividades de Análise de Requisitos e Análise, porém, pode-se evoluir com essa modelagem até a atividade de Projeto.

Existe distinção na notação dos elementos de um Diagrama de Casos de Uso da atividade de Análise de Requisitos, Análise e Projeto?



Para saber mais

Para conhecer mais sobre a modelagem organizacional e a engenharia de requisitos, acesse o material indicado a seguir:

Livros:

- SOMMERVILLE, Ian. **Engenharia de software**. 9. ed. São Paulo: Pearson, 2011.

Fique ligado

Nesta seção, introduzimos a modelagem da atividade de Análise de Requisitos, utilizando algumas técnicas de modelagem da UML. A modelagem inicial de um sistema deve ser cuidadosamente especificada em consonância com o domínio do sistema para garantir que os requisitos funcionais de um sistema sejam implementados com consistência e possam ser validados com os usuários do sistema, por meio de documentos de fácil leitura, interpretação e compreensão.

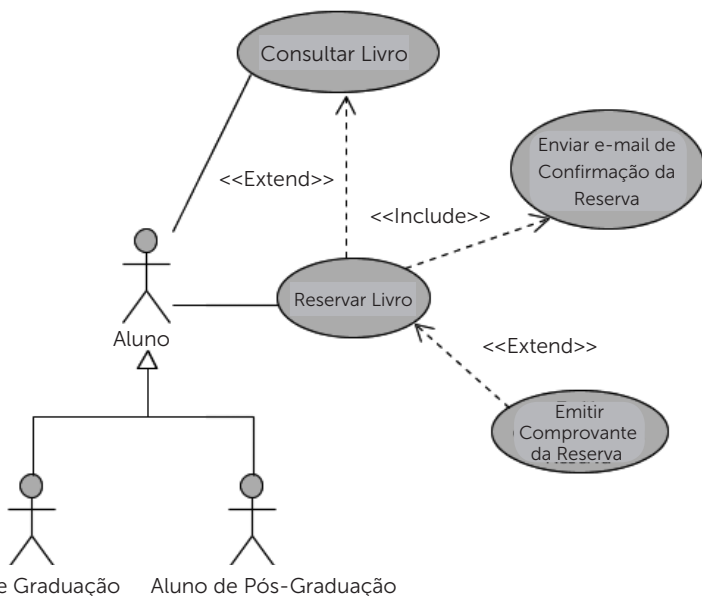
Atividades de aprendizagem

1. A partir da *Unified Modeling Language* (UML) 2.0, as treze técnicas de modelagem são classificadas em estruturais e comportamentais, sendo o Diagrama de Casos de Uso a principal técnica de modelagem comportamental para iniciar a modelagem do sistema.

Assinale a alternativa correta que apresenta os objetivos do Diagrama de Casos de Uso.

- a) É um diagrama que representa os objetos com seus estados e transições de estados.
- b) É um diagrama flexível com poucos elementos de notação para representar as colaborações entre os objetos do sistema.
- c) É um diagrama que representa a organização do sistema em pacotes para categorizar ou modularizar o sistema.
- d) É um diagrama abstrato e flexível com poucos elementos de notação, que representa a interação entre os elementos Ator e Casos de Uso.
- e) É uma técnica de modelagem estática com poucos elementos de notação para representar os requisitos não funcionais do sistema.

2. Na *Unified Modeling Language* (UML), o Diagrama de Casos de Uso proporciona uma forma de representar a aplicação segundo a perspectiva do usuário. Considere o Diagrama de Casos de Uso para um sistema de biblioteca de uma instituição de ensino, apresentado na figura a seguir:



Sobre os relacionamentos indicados no diagrama, analise os itens a seguir:

- I. Os relacionamentos especiais <<Include>> e <<Extends>> são exclusivos para casos de uso.
- II. A utilização de diferentes perfis de Aluno (atores: Aluno de Graduação e Aluno de Pós-Graduação) é representada através de um tipo de relacionamento especial chamado "Dependência", o qual pode ser aplicado tanto a casos de uso como entre atores.
- III. O relacionamento <<Include>> entre os casos de uso "Reservar Livro" e "Enviar e-Mail de Confirmação da Reserva" representa um caminho obrigatório de execução de funções da aplicação.
- IV. Os casos de uso "Consultar Livro" e "Reservar Livro" representam um caminho não obrigatório de execução, ou seja, a partir do caso de uso "Consultar Livro" pode-se executar o caso de uso estendido "Reservar Livro".
- V. O caso de uso "Reservar Livro" e "Emitir Comprovante da Reserva" representam um caminho não obrigatório de execução, ou seja, a partir do caso de uso "Emitir Comprovante da Reserva" pode-se executar o caso de uso estendido "Reservar Livro".

Estão corretos os itens:

- a) I, II e III.
- b) II, III e IV.
- c) I, III e IV.

d) III, IV e V.

e) I, II, III, IV e V.

Seção 3

Modelagem da atividade de Análise com UML

Introdução à seção

Caro(a) aluno(a), nesta seção, apresentaremos a modelagem da atividade de Análise do estudo de caso proposto, considerando a adoção do Diagrama de Casos de Uso, do Diagrama de Classes, do Diagrama de Atividades e do Diagrama de Sequência correspondente a casos de uso, e do Diagrama de Máquina de Estados.

A atividade de Análise consiste em modelar as funcionalidades do sistema, definindo o que o sistema deve fazer independente das tecnologias que serão adotadas no desenvolvimento do software. A ênfase está em iniciar a modelagem dinâmica do software a partir da identificação das funcionalidades do sistema e especificá-las com o Modelo de Casos de Uso.

O Diagrama de Casos de Uso representa as funcionalidades do sistema, ou seja, os requisitos funcionais do sistema, e os elementos externos ao sistema que interagem com ele, identificados como atores. É o diagrama mais abstrato, flexível e informal da UML, sendo utilizado no início da modelagem do sistema, contudo o diagrama será consultado e consistido com as demais técnicas de modelagem dinâmicas e, se necessário, será atualizado durante o processo de engenharia do software (CATARINO, 2012).

Após a criação do Diagrama de Casos de Uso, conforme orientação da UML, recomenda-se complementar o Modelo de Casos de Uso com uma documentação, geralmente adotamos o formato descritivo em cenários ou fluxos – principal e alternativos. O formato de documentação de um Caso de Uso é flexível por meio de uma linguagem simples, permitindo que se documente o Caso de Uso com descrições no formato contínuo, tabular ou numerado, até mesmo através do uso de pseudocódigo (CATARINO, 2012).

Considerando que o Diagrama de Casos de Uso está concluído, a próxima etapa é analisar os Casos de Uso para iniciar o trabalho de identificação das classes de objetos, o qual se faz necessário

compreender como o sistema está estruturado internamente para que as funcionalidades sejam executadas. As classes de objetos, muitas vezes, podem ser identificadas também como substantivos nas descrições do domínio do problema, ou recomenda-se adotar técnicas auxiliares para identificação das classes. Assim, elabora-se uma primeira visão do Diagrama de Classes.

O Diagrama de Classes representa a modelagem da parte estática do sistema, representando um conjunto de classes com seus atributos, operações e relacionamentos. Após a identificação das classes e atributos, deve-se verificar a consistência entre as classes e os casos de usos já definidos previamente, e assim observar a necessidade de novas classes ou eliminar incoerências e redundâncias. 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, especificando o modelo ideal do Diagrama de Classes da atividade de Projeto.

Na sequência, sugerimos continuar a modelagem comportamental do sistema, atentando-se em consistir a evolução da modelagem de análise sempre a partir dos casos de uso e das características e do comportamento das classes.

3.1 Diagrama de Casos de Uso

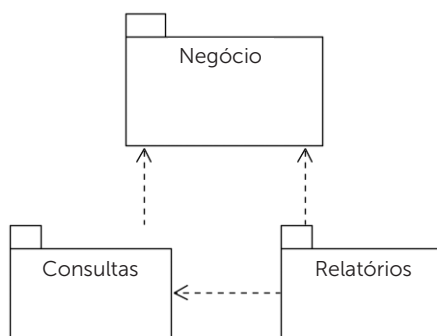
A partir da listagem dos requisitos funcionais na seção anterior, analisamos os requisitos identificados, abstraímos as regras de negócio do sistema e preferimos estruturar os casos de uso em três pacotes. Na UML, os agrupamentos que organizam um modelo (conjunto de diagramas) são chamados de pacotes. O Diagrama de Pacotes tem por objetivo representar os subsistemas ou módulos do sistema de forma a determinar as partes que o compõem. Demonstra como os elementos estão organizados nos pacotes e as dependências que existem entre os elementos e os próprios pacotes (GUEDES, 2008).

Um Pacote pode se subdividir em outros pacotes, devendo ser representativo e único. Relacionamentos de Dependência devem ser estabelecidos entre os pacotes para representar as dependências entre os pacotes, indicando que o elemento dependente necessita, de alguma forma, do elemento do qual depende (CATARINO, 2012).

Sugerimos, antes de iniciar um Diagrama de Casos de Uso, listar todos os casos de uso do sistema em uma tabela, formando uma lista de casos de uso, para assim identificar todos os casos de uso e tomar a decisão de desenhar um único Diagrama de Casos de Uso ou desenhar vários Diagramas de Casos de Uso, compondo um Modelo de Casos de Uso e, com isso, elaborar o Diagrama de Pacotes dos Casos de Uso, no qual cada pacote terá um Diagrama de Casos de Uso correspondente.

A Figura 4.5 exemplifica o Diagrama de Pacotes dos casos de uso. Foram definidos três pacotes - Negócio, Consultas e Relatórios – para organizarem os casos de uso previamente identificados, constituindo um modelo de casos de uso com três Diagramas de Casos de Uso.

Figura 4.5 | Diagrama de Pacotes dos Casos de Uso



Fonte: elaborada pela autora.

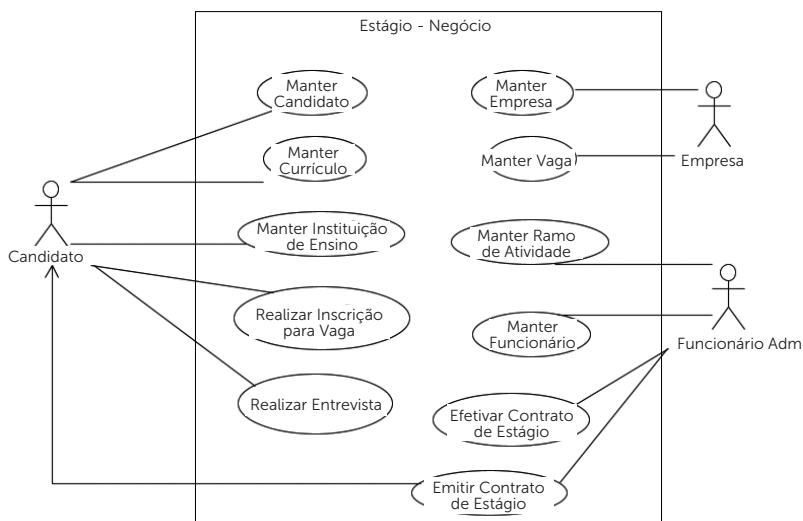
A Figura 4.5 mostra o Diagrama de Casos de Uso representado com a fronteira do sistema, ilustrando os casos de uso correspondentes aos requisitos funcionais e outros, considerando a abstração do contexto.

Foram definidos os atores primários – Candidato, Empresa e Funcionário Administrativo –, os quais interagem com os casos de uso, ou seja, o analista de sistemas deve identificar quais as fontes de dados e quais são os destinos das informações geradas pelo sistema. Além de fazer essa identificação inicial, o analista de sistema deve continuar a pensar sobre os atores quando define os casos de uso, pois nessa atividade podem aparecer outros atores ainda não identificados.

É importante também estudar cada requisito funcional identificado na atividade anterior para tomar a decisão na definição dos casos de uso, pois um requisito funcional pode ser representado por dois ou mais casos de uso, e vice-versa, sobre o qual recomendamos pensar no objetivo de cada caso de uso, considerando a usabilidade do sistema para os usuários, pois os casos de uso representam os processos de negócio automatizados pelo software desenvolvido.

O Diagrama de Casos de Uso deve dar suporte à parte escrita do modelo, fornecendo uma visão de alto nível do sistema. Posteriormente, recomenda-se elaborar a documentação de cada caso uso, porém, como a UML não define uma estruturação específica a ser utilizada na descrição dos casos de uso, recomendamos adotar um único padrão para documentar todos os casos de uso (CATARINO, 2012).

Figura 4.6 | Diagrama de Casos de Uso – Pacote Negócio



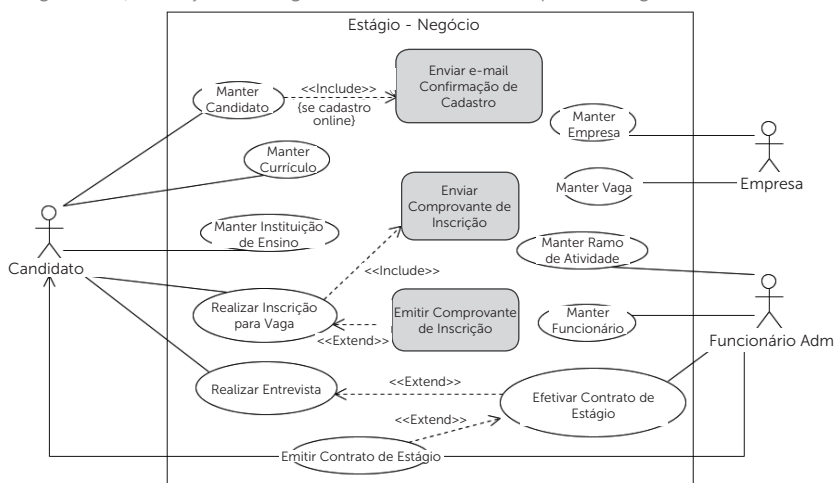
Fonte: elaborada pela autora.

A Figura 4.7 mostra uma segunda visão do Diagrama de Casos de Uso, compreendendo os casos de uso base (casos de uso associados a atores) do diagrama anterior e novos casos de uso que foram definidos a partir de uma análise do contexto do sistema, ou a partir de regras de negócio (políticas, condições ou

restrições que devem ser consideradas na execução dos processos) estabelecidas pela empresa.

Alguns relacionamentos entre os casos de uso já foram definidos porque completam o papel da funcionalidade, considerando a sua execução. Observa-se essa situação no caso de uso “Realizar Inscrição para Vaga”, no qual, a partir da inscrição para uma vaga de estágio que um candidato realiza, o candidato receberá (representado pelo relacionamento de inclusão <<Include>>) um comprovante da inscrição efetuada por e-mail, automaticamente, e se for de interesse do candidato, ele poderá (representado pelo relacionamento de extensão <<Extend>>) emitir o comprovante da inscrição efetuada após a confirmação de cadastro da inscrição.

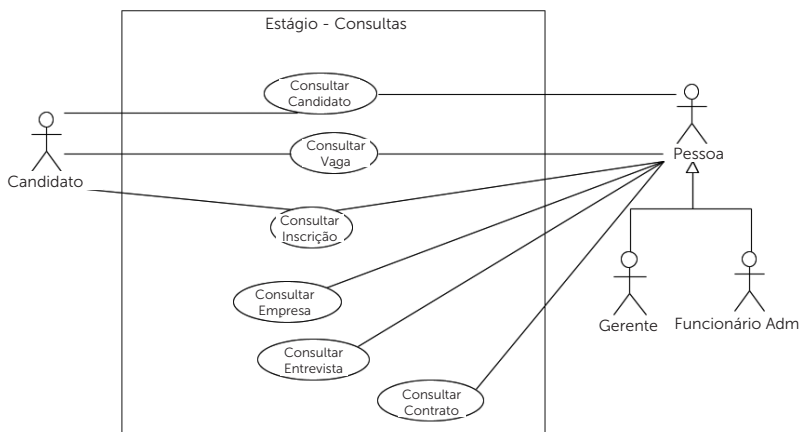
Figura 4.7 | Evolução do diagrama de casos de uso – pacote negócios



Fonte: elaborada pela autora.

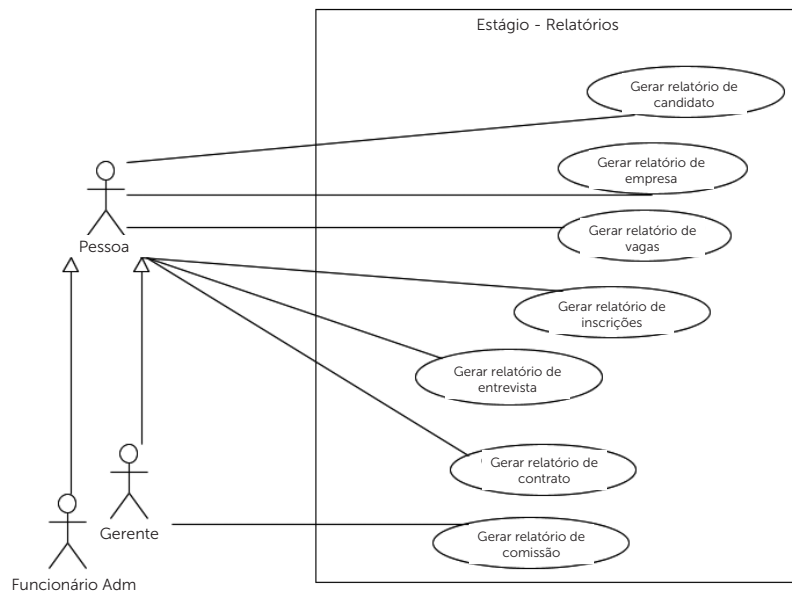
As Figuras 4.8 e 4.9 representam os Diagramas de Casos de Uso referentes às consultas e aos relatórios. Perceba que alguns casos de uso estão associados ao ator genérico, o qual indica que os atores especializados têm acesso a todos os casos de uso associados ao ator genérico.

Figura 4.8 | Diagrama de casos de uso – pacote consultas



Fonte: elaborada pela autora.

Figura 4.9 | Diagrama de casos de uso – pacote relatórios



Fonte: elaborada pela autora.

As Figuras 4.8 e 4.9 ilustraram os Diagramas de Casos de Uso referentes aos pacotes de consultas e relatórios, representando uma divisão por categorização de casos de uso. Essa representação é uma sugestão de dividir o Diagrama de Casos de Uso em um Modelo de Casos de Uso, ou seja, um conjunto de diagramas que facilita a leitura e compreensão das funcionalidades do sistema.

3.2 Diagrama de Classes

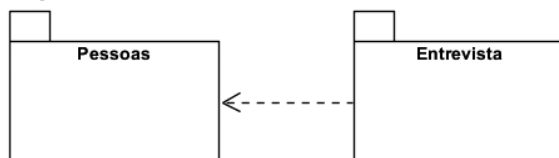
A partir da abstração dos casos de uso, inicia-se a identificação das classes de objetos e a elaboração do Diagrama de Classes, que é considerado a principal técnica de modelagem estrutural da UML.

O objetivo do Diagrama de Classes é permitir a visualização das classes utilizadas pelo sistema e como estas se relacionam. Esse diagrama apresenta uma visão estática de como as classes estão organizadas, preocupando-se em definir sua estrutura lógica (GUEDES, 2008).

Recomendamos identificar as classes analisando cada caso de uso e pensando na sua manipulação ou manutenção em tempo de execução. Por exemplo, se foi definido o caso de uso “Manter Candidato”, é porque o sistema deve permitir a manipulação dos objetos do tipo candidatos com as operações básicas de inclusão, alteração, exclusão (ou controlar o status do objeto, ativo ou não) ou pesquisa, assim é necessário definir uma classe para esses objetos com os atributos e as operações pertinentes ao contexto do domínio do sistema. Entretanto, pode acontecer de um ou mais casos de uso manipularem objetos da mesma classe, e vice-versa.

Considerando o contexto do estudo de caso, decidimos organizar as classes em dois pacotes – Pessoas e Entrevista –, conforme ilustra o Diagrama de Pacotes na Figura 4.10.

Figura 4.10 | Diagrama de Pacotes do Modelo de Classes



Fonte: elaborada pela autora.

Ainda pode-se utilizar de técnicas para identificação das classes, entre elas destacamos:

a) **Análise Textual de Abbott (1983):** utilizam-se diversas fontes de informação sobre o sistema: documento de requisitos, modelo de negócio, glossários, conhecimentos sobre o domínio etc. Para cada um desses documentos, os nomes (substantivos e adjetivos) são destacados e os sinônimos são removidos. Cada termo remanescente pode se tornar uma classe candidata, um atributo ou ser descartado. Os verbos são destacados para identificar as operações de uma classe e das associações entre classes. Verbos com sentido de “ação” (calcular, confirmar, cancelar, comprar, fechar, estimar, depositar etc.) são operações em potencial; verbos com sentido de “ter” são agregações ou composições em potencial; verbos com sentido de “ser” são generalizações em potencial; e os demais verbos são associações.

b) **Análise dos Casos de Uso:** preconizada pelo processo de desenvolvimento RUP, seguindo os passos:

- Faça a descrição de cada caso de uso.
- Para cada caso de uso, identifique classes a partir do comportamento do caso de uso.
- Para cada classe de análise, descreva suas responsabilidades, atributos e associações.
- Unifique as classes de análise identificadas em um ou mais diagramas de classes.

c) **Cartões CRC (Classes, Responsabilidades e Colaboradores):** propostos em 1989 por Kent Beck e Ward Cunningham, neles o comportamento de um objeto é definido de tal forma que ele possa cumprir com suas responsabilidades, utilizando-se de cartão CRC, no formato de tabela, para representar as classes identificadas.

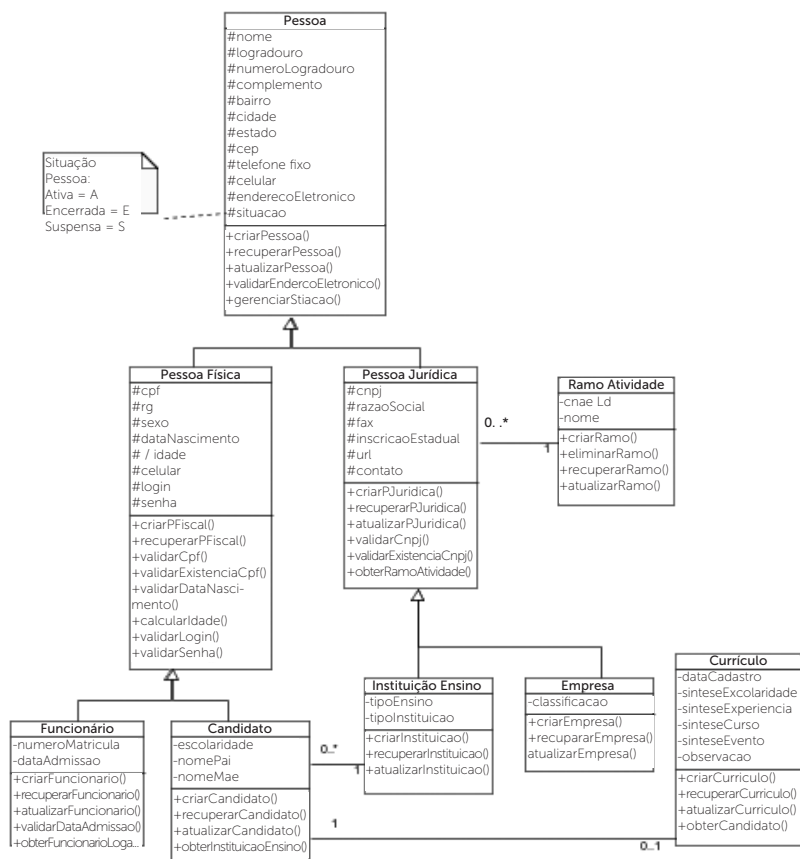
d) **Taxonomia de Conceitos,** identificando e analisando os conceitos conforme:

- Conceitos concretos, como edifícios, carros, salas de aula etc.
- Papéis desempenhados por seres humanos, como professores, alunos, empregados, clientes etc.
- Eventos que representam ocorrências em um determinado período de data e horário, por exemplo, reuniões, aulas, aterrisagens etc.
- Lugares que representam áreas reservadas para pessoas, como, escritórios, filiais, salas de aula etc.
- Organizações que demonstram coleções de pessoas ou de recursos, por exemplo, departamentos, projetos, campanhas, turmas etc.
- Conceitos abstratos referentes aos princípios ou às ideias não tangíveis, como, reservas, vendas, inscrições etc.

As Figuras 4.11 e 4.12 representam os diagramas do Modelo de Classes de análise correspondente aos pacotes “Pessoas e Entrevista”, respeitando as regras básicas de modelagem do diagrama e as regras de negócio do sistema.

Uma Classe é representada por um retângulo com, no máximo, três partes. Na primeira parte (de cima para baixo), é exibido o nome da Classe. Por convenção, o nome é apresentado no singular e com as palavras compostas começando por letra maiúscula. Na segunda parte, são declarados os atributos e, na terceira parte, são declaradas as operações.

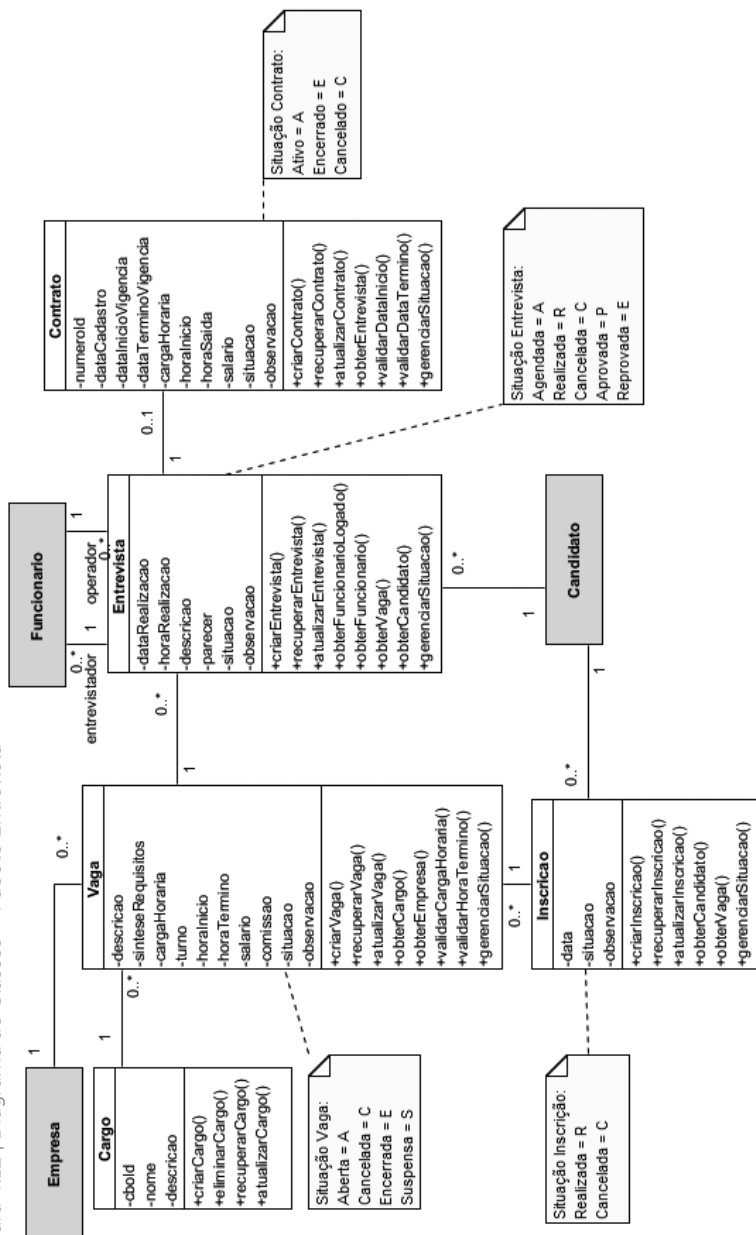
Figura 4.11 | Diagrama de Classes – Pacote Pessoas



Fonte: elaborada pela autora.

O nome de um atributo deve ser declarado por um substantivo, tipicamente, em letra minúscula e, para palavras compostas, usa-se concatená-las, sendo que, a partir da segunda palavra, inicia-se com letra maiúscula, por exemplo, *dataNascimento*, *razaoSocial*, *cargaHoraria* etc. O nome de uma operação deve ser declarado por um verbo, usando a mesma convenção de letras minúsculas e maiúsculas dos atributos, por exemplo, *criarVaga*, *recuperarCargo* etc.

Figura 4.12 | Diagrama de Classes – Pacote Entrevista



Fonte: Elaborado pela autora (2017).

Na UML, os modos pelos quais os itens podem estar conectados a outros, isto é, logicamente ou fisicamente, são modelados como relacionamentos, que permitem compartilhar informações e colaboram para a execução dos processos pelo sistema (GUEDES, 2008).

Existem quatro tipos de relacionamentos:

a) Associações: são relacionamentos estruturais entre instâncias. Tipos de associações: Unária (autoassociação ou reflexiva), binária, ternária, classe associativa e agregação.

b) Generalizações: conectam classes generalizadas a outras mais especializadas, o que é conhecido como relacionamento Generalização e Especialização.

c) Dependências: são relacionamentos de utilização entre casos de uso, classes, pacotes e anotações.

d) Realizações: são relacionamentos que especificam um contrato de execução entre classes e interfaces.

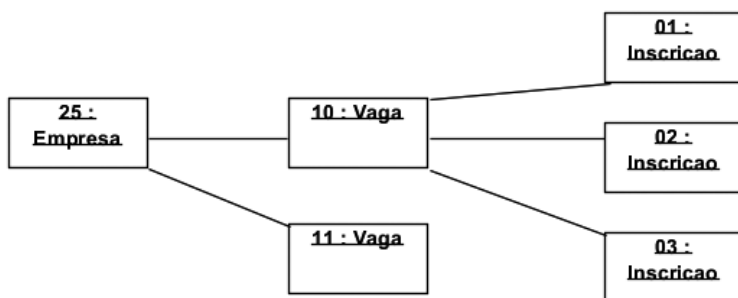
Em cada extremidade da associação deve ser definida a multiplicidade da associação. A multiplicidade depende de pressupostos e de como são definidas as fronteiras de um problema, conforme as regras de negócio.

Posteriormente, o Modelo de Classes deve ser refinado com detalhes das tecnologias que serão adotadas na implementação do sistema, reapresentando outra visão do Diagrama de Classes correspondente à atividade de Projeto.

3.3 Diagrama de Objetos

O Diagrama de Objetos é uma instância do Diagrama de Classes. Cada classe mostra seu objeto em um determinado ponto de tempo. É relevante adotá-lo para domínios complexos de software para facilitar a compreensão do sistema. A Figura 4.13 demonstra um exemplo de Diagrama de Objetos para ilustrar e esclarecer certos aspectos de um Diagrama de Classes. O Diagrama de Objetos é raramente utilizado na prática, porém a sua elaboração justifica-se em situações de difícil compreensão do domínio e contexto do sistema.

Figura 4.13 | Diagrama de Objetos

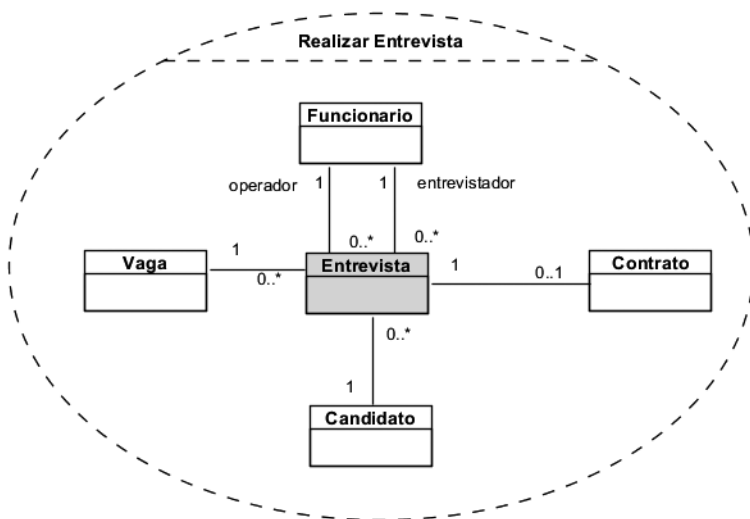


Fonte: elaborada pela autora.

3.4 Diagrama de Estruturas Compostas

Na sequência, recomenda-se documentar a atividade de Análise com o Diagrama de Estruturas Compostas, que é utilizado para representar Colorações, ilustrando o relacionamento entre os elementos participantes para executar uma função. Uma Coloração descreve uma visão de um conjunto de entidades cooperativas interpretadas por instâncias que cooperam entre si para executar uma função específica (GUEDES, 2008). A Figura 4.14 exemplifica um Diagrama de Estruturas Compostas do caso de uso “Realizar Entrevista”.

Figura 4.14 | Diagrama de estruturas compostas da colaboração realizar entrevista



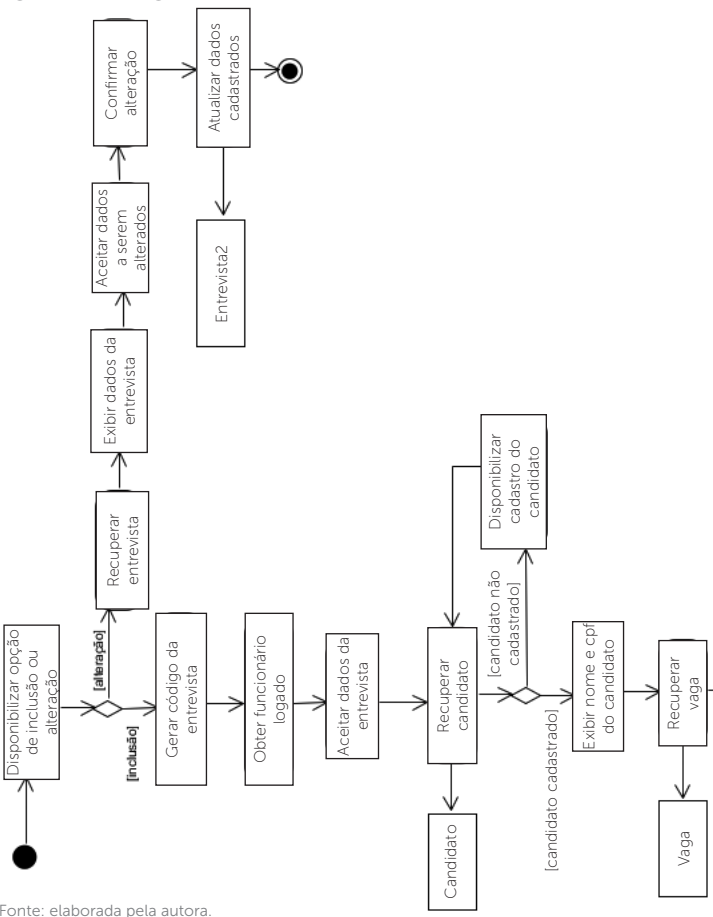
Fonte: elaborada pela autora.

3.5 Diagrama de Atividades

Ainda na atividade de Análise, recomendamos especificar os casos de uso com o Diagrama de Atividades. Um Diagrama de Atividades descreve uma sequência de atividades decompostas em ações, dispostas em fluxo vertical ou horizontal. Uma atividade é composta por um conjunto de ações, ou seja, os passos necessários para que as atividades sejam concluídas, representando a execução do caso de uso (CATARINO, 2012).

As Figuras 4.15 e 4.16 representam um Diagrama de Atividades em duas partes, correspondente ao caso de uso “Realizar Entrevista”.

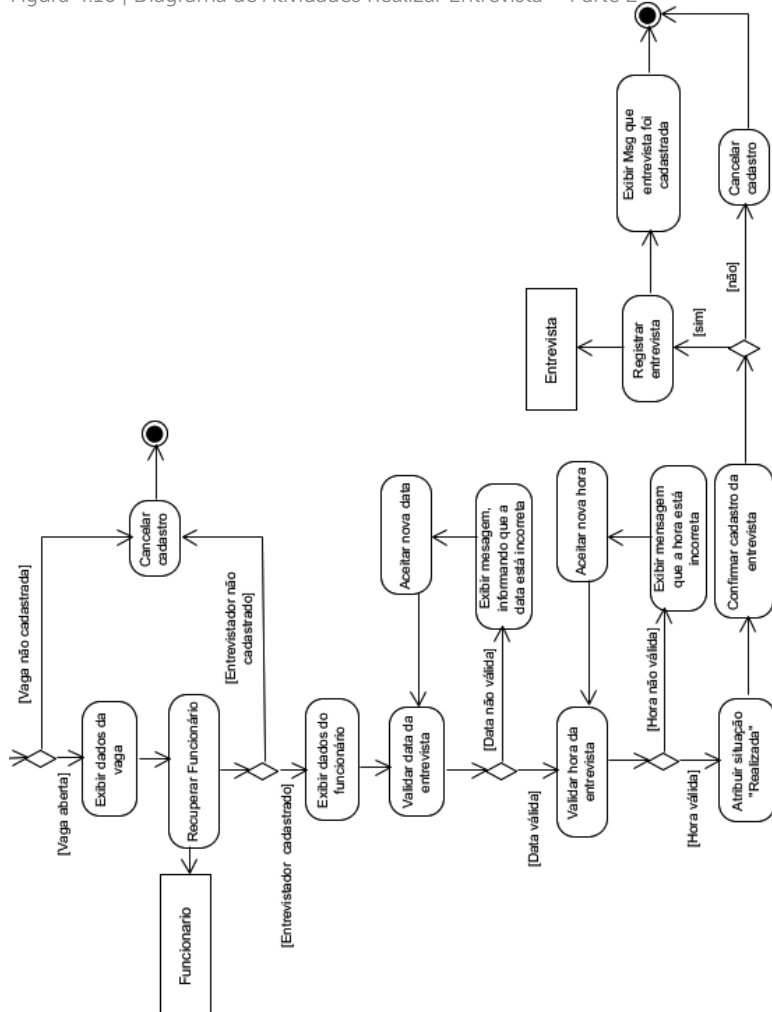
Figura 4.15 | Diagrama de atividades realizar entrevista – Parte 1



Fonte: elaborada pela autora.

O Diagrama de Atividades correspondente ao caso de uso “Realizar Entrevista”, neste exemplo, está ilustrado em duas figuras apenas por uma questão de tamanho do diagrama, pois para representá-lo em um único diagrama ficaria ilegível. Observe que neste exemplo foi adotada a representação de fluxo de controle simples, pois nenhuma ação ocorre simultaneamente, não sendo necessário adotar elementos que representassem fluxos simultâneos (nó de bifurcação, nó de junção etc.).

Figura 4.16 | Diagrama de Atividades Realizar Entrevista – Parte 2



Fonte: elaborada pela autora.

As Figuras 4.15 e 4.16 representaram o Diagrama de Atividades correspondente ao caso de uso “Realizar Entrevista”, no qual todas as ações devem ser posteriormente consistidas com as operações dos objetos entrevista, representadas no Diagrama de Classes da atividade de Projeto, pois o Diagrama de Classes da atividade de Análise, ilustrado nas Figuras 4.11 e 4.12, demonstra as principais operações identificadas no momento da análise. A sequência de ações representadas no fluxo vertical corresponde à execução de um cenário principal do caso de uso, já as ações representadas no fluxo horizontal correspondem à execução dos cenários alternativos do caso de uso.

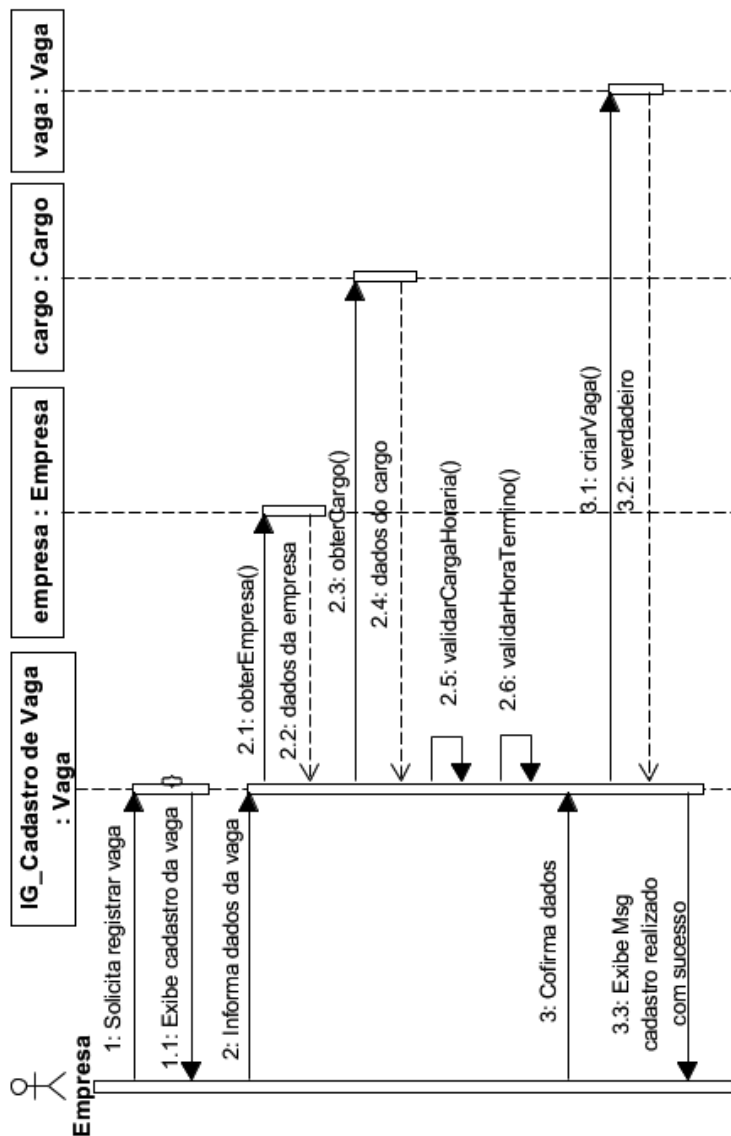
3.6 Diagrama de sequência

O Diagrama de Sequência representa a sequência de eventos que ocorrem em um determinado processo, identificando quais métodos devem ser disparados entre os atores e objetos envolvidos e em que ordem (GUEDES, 2008). O Diagrama de Sequência baseia-se no Diagrama de Casos de Uso e no Diagrama de Classes para representar os objetos que colaboram durante a execução de um caso de uso.

Os atores são os mesmos do Diagrama de Casos de Uso e possuem a mesma representação, porém com uma linha de vida. Os objetos representam as instâncias das classes envolvidas no processo. Os objetos do Diagrama de Sequências também possuem uma Linha de Vida e um objeto pode existir desde o início do processo ou ser criado durante a execução dele.

A Figura 4.17 representa um exemplo do Diagrama de Sequência (fluxo principal) da atividade de Análise correspondente ao caso de uso “Manter Vaga”.

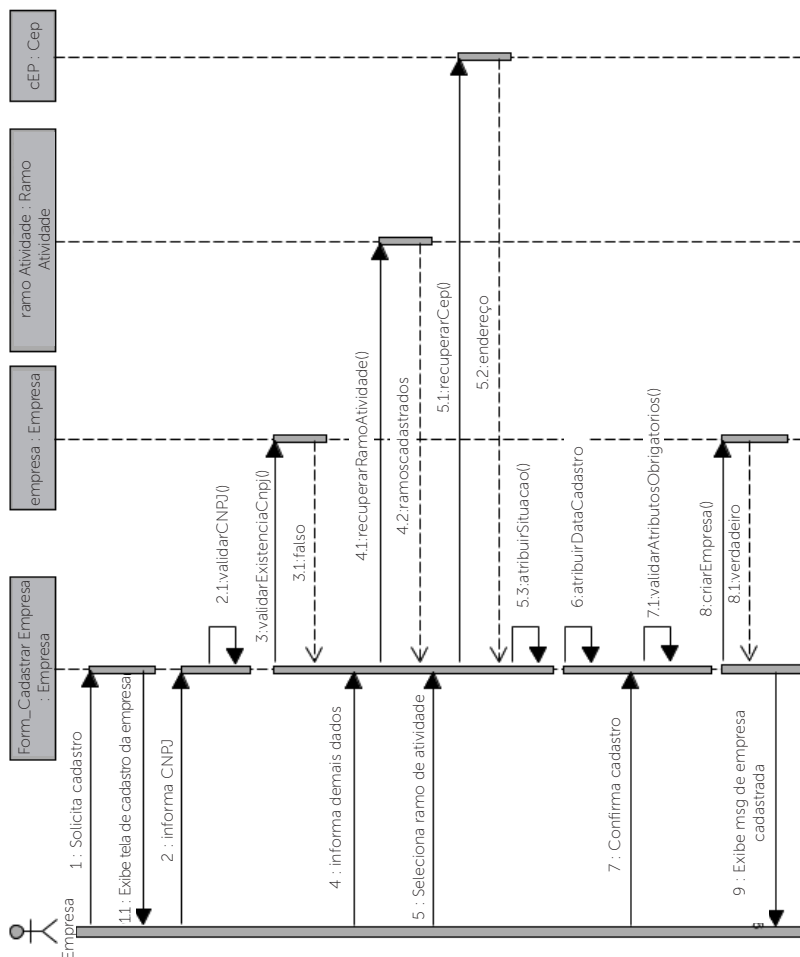
Figura 4.17 | Diagrama de sequência – manter vaga



Fonte: elaborada pela autora.

A Figura 4.18 representa um exemplo do Diagrama de Sequência (fluxo principal) de análise, correspondente ao caso de uso “Manter Empresa”. Geralmente, o Diagrama de Sequência é adotado com mais frequência, na atividade de Projeto, para detalhar as interações entre os objetos na ordem temporal em que elas acontecem com os detalhes de implementação.

Figura 4.18 | Diagrama de sequência – manter empresa



Fonte: Elaborado pela autora (2017).

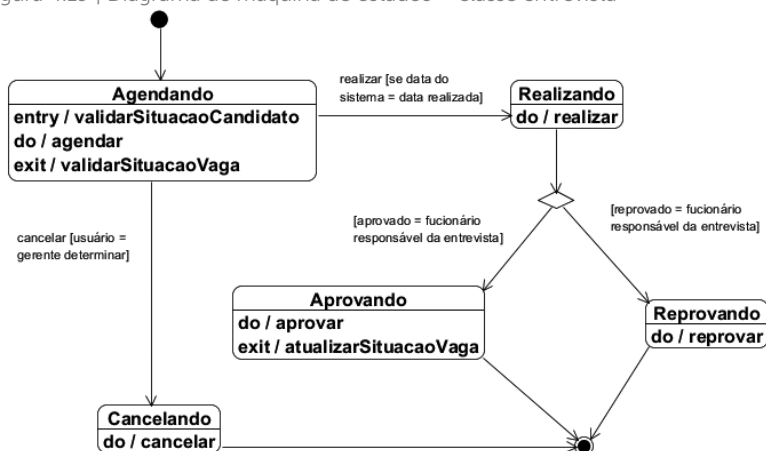
Os diagramas ilustrados nas Figuras 4.17 e 4.18 correspondem a Diagramas de Sequência da atividade de Análise, contudo os diagramas devem ser refinados com detalhes de notação (estereótipos para objetos, fragmentos, ocorrências de interação, portões, fragmentos combinados com operador etc.) em conformidade com a arquitetura do sistema e padrões de implementação.

3.7 Diagrama de Máquina de Estados

O Diagrama de Máquina de Estados representa as alterações de estados de uma instância de uma classe, de um Caso de Uso ou mesmo de um sistema completo. Segundo Bezerra (2007, p. 287), o Diagrama de Máquina de Estados “permite descrever o ciclo de vida de objetos de uma classe, sendo que os eventos causam a transição de um estado para outro e a realização de operações resultantes”. Um Estado representa a abstração de uma forma de apresentação dos objetos de uma classe em um determinado instante de tempo.

A Figura 4.19 representa um exemplo do Diagrama de Máquina de Estados correspondente à classe “Entrevista”. As transições e condições de guarda dos estados foram definidas a partir de um estudo e análise das regras de negócio, para assegurar a consistência dos objetos da classe “Entrevista” aplicadas ao contexto.

Figura 4.19 | Diagrama de máquina de estados – classe entrevista



Fonte: elaborada pela autora.

O diagrama ilustrado na Figura 4.19 retrata os estados definidos para os objetos da classe “Entrevista” e suas transições de estados indicadas com condições de guarda. Nesse diagrama, toda entrevista, ao ser registrada, assume o estado inicial “Agendando”, sendo que a ação de entrada é executada “validarSituacaoCandidato”, pois uma entrevista só pode ser agendada para um candidato com matrícula ativa; e a ação de saída “validarSituacaoVaga” também é executada para consistir se a entrevista agendada se refere a uma vaga aberta. Uma vez que a data do sistema seja igual à data de realização da entrevista, então ocorre a transição para o estado “Realizando” e, posteriormente, dependerá de uma decisão do funcionário que entrevistou o candidato para analisar se a entrevista foi aprovada ou não, atualizando a situação da entrevista correspondente à decisão. Por fim, para entrevistas aprovadas, é disparada uma ação de saída “atualizarSituacaoVaga” para atualizar a situação do objeto “Vaga” para vaga preenchida ou encerrada.

Posteriormente, na implementação do sistema, os estados significantes definidos para os objetos de uma classe, bem como suas transições, condições de guarda e ações de entrada e saída devem ser fielmente implementadas para garantir a consistência e integridade do sistema com sua modelagem.



Questão para reflexão

A UML não faz grande distinção entre a notação da atividade de Análise e da atividade de Projeto. A atividade de Projeto é uma extensão dos diagramas criados na Análise com um grau de detalhamento para iniciar a próxima atividade, a implementação das funcionalidades do software. Os diagramas criados durante a Análise são refinados com mais detalhes, apresentando várias visões da especificação do sistema. O Diagrama de Casos de Uso e o Diagrama de Classes são os dois principais diagramas que são adotados na atividade de Análise (CATARINO, 2012).

Além dos Diagramas de Casos de Uso e de Classes, qual(is) diagrama(s) da UML você considera essencial(is) para documentar cada atividade do processo de desenvolvimento do sistema?

Para conhecer mais sobre a modelagem orientada a objetos de análise e projeto com UML, acesse o material indicado a seguir:

Livros:

BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas com UML**. 2. ed. Rio de Janeiro: Elsevier, 2007.

GUEDES, Gilleanes T. A. **UML: uma abordagem prática**. 3. ed. São Paulo: Novatec, 2008.

MEDEIROS, Ernani Sales de. **Desenvolvendo software com UML 2.0: definitivo**. São Paulo: Pearson, 2006.

PENDER, TOM. **UML Bible**. Indianapolis, Indiana: John Wiley & Sons, 2003.

Atividades de aprendizagem

1. Os diagramas estruturais têm como objetivo a visualização, especificação, construção e documentação de aspecto estáticos do software.

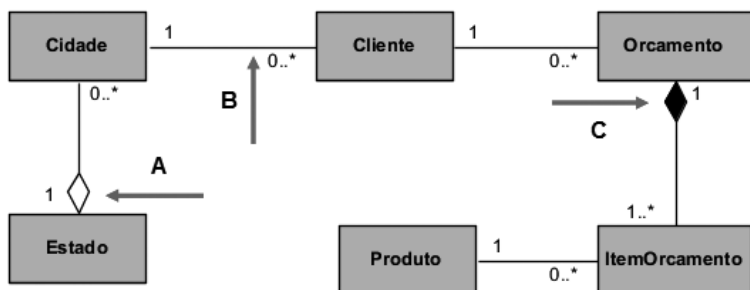
Considerando as técnicas de modelagem estruturais da UML, complete as lacunas dos seguintes parágrafos:

À medida que o software se torna mais complexo, é importante garantir que os módulos estejam devidamente organizados por meio de coleções exclusivas, assim utiliza-se do _____ para representar o agrupamento de elementos com a mesma semântica. O _____ pode ser visto como uma instância do _____, do mesmo modo que um objeto é uma instância de uma classe.

Assinale a alternativa correta que indica a sequência dos termos que preenchem as lacunas:

- a) Diagrama de Casos de Uso. Diagrama de Classes. Diagrama de Objetos.
- b) Diagrama de Casos de Uso. Diagrama de Classes. Diagrama de Pacotes.
- c) Diagrama de Pacotes. Diagrama de Casos de Uso. Diagrama de Objetos.
- d) Diagrama de Pacotes. Diagrama de Classes. Diagrama de Objetos.
- e) Diagrama de Pacotes. Diagrama de Objetos. Diagrama de Classes.

2. Analise o Diagrama de Classes a seguir, observando as associações estabelecidas entre as classes.



Considerando a representação das associações “A, B e C” estabelecidas entre classes, assinale a alternativa correta que indica o nome das associações.

- Agregação, associação binária e composição.
- Agregação, classe associativa e generalização.
- Agregação, associação unária e composição.
- Composição, associação binária e agregação.
- Composição, associação unária e generalização.

Fique ligado

Na Unidade 4, mostramos um exemplo da modelagem de um estudo de caso de fácil compreensão para reforçar o seu aprendizado sobre as principais técnicas de modelagem da UML, entre elas o Diagrama de Casos de Uso e o Diagrama de Classes, aplicadas, principalmente, no desenvolvimento de sistemas de informação. A UML não faz grande distinção entre a notação da atividade de Análise e da atividade de Projeto, assim a atividade de Projeto é uma extensão dos diagramas criados na Análise para prepará-los para a implementação do software.

Para concluir o estudo da unidade

Aqui finalizamos esta unidade, esperamos que tenha alcançado a sua meta inicial de estudo, que era compreender a modelagem estática e dinâmica, considerando as principais técnicas de modelagem da UML para visualizar, especificar, construir e documentar sistemas complexos de software. Para complementar sua aprendizagem, consulte nossas Referências.

Atividades de aprendizagem da unidade

1. A *Unified Modeling Language* (UML) 2.0 abrange treze técnicas de modelagem, classificadas em estruturais e comportamentais. O Diagrama de Classes é a principal técnica de modelagem estrutural da UML, composto, basicamente, pelos elementos classes e relacionamentos com suas multiplicidades.

Sobre relacionamentos do Diagrama de Classes, julgue as sentenças a seguir:

I. Os relacionamentos são classificados em: Associação, Generalização, Dependência e Realização.

II. São tipos de associação: binária, ternária, classe associativa e agregação.

III. Associação do tipo Agregação demonstra que as informações de um objeto identificado como o objeto-todo precisam ser complementadas pelas informações contidas em um ou mais objetos de outra classe, identificados como objeto-partes, não existindo uma ligação forte entre as duas.

IV. Composição é uma variação da Agregação, na qual é apresentado um vínculo mais forte entre os objetos-todo e os objetos-partes, demonstrando que os objetos-partes são associados a um único objeto-todo.

V. O relacionamento de Generalização representa uma hierarquia entre a classe genérica e as classes especializadas.

Estão corretos os itens:

- a) I, II e III.
- b) II, III e IV.
- c) I, III e V.
- d) I, II, III e IV.
- e) I, II, III, IV e V.

2. O Diagrama de Classes é a principal técnica de modelagem estrutural, representando a modelagem da parte estática do sistema com um conjunto de classes, atributos, operações e relacionamentos. Assinale a alternativa correta que apresenta os objetivos do Diagrama de Classes.

- a) É um diagrama que representa os componentes de software com sua estrutura e conexão entre os componentes.
- b) É um diagrama que representa a organização do sistema, podendo ser utilizado de maneira independente ou associado com outros diagramas.
- c) É um diagrama abstrato e flexível com poucos elementos de notação para representar os requisitos não funcionais do sistema.
- d) É um diagrama que representa a modelagem da parte estática do sistema, representando um conjunto de classes com seus atributos, operações e relacionamentos.
- e) É um diagrama abstrato e flexível com poucos elementos de notação, que representa a interação entre os elementos Ator e Casos Uso.

3. Conforme Guedes (2008), na *Unified Modeling Language* (UML), os modos pelos quais os itens podem estar conectados a outros, isto é, logicamente ou fisicamente, são modelados como relacionamentos, que permitem compartilhar informações e colaboram para a execução dos processos pelo sistema.

Fonte: GUEDES, Gilleanes T. A. *UML: uma abordagem prática*. 3. ed. São Paulo: Novatec, 2008.

Sobre os relacionamentos da UML, analise e julgue as sentenças a seguir:

- I. Um paciente realiza várias consultas médicas.
- II. Uma consulta médica é realizada por um médico.
- III. Existem dois tipos de médicos: clínico e cirurgião.
- IV. Uma guia de exames é composta por vários itens de exames.

Assinale a alternativa que melhor se adequaria ao modelar os tipos de relacionamentos descritos acima.

- a) Associação binária, associação binária, composição e agregação.
- b) Associação binária, associação binária, generalização e composição.
- c) Composição, agregação, generalização e associação binária.
- d) Composição, agregação, associação binária e generalização.
- e) Composição, associação binária, associação binária e generalização.

4. Conforme a classificação das técnicas de modelagem da *Unified Modeling Language* (UML), o Diagrama de Casos de Uso é uma técnica de modelagem comportamental.

Considerando as características do Diagrama de Casos de Uso, indique "V" para os itens verdadeiros e "F" para os itens falsos.

1. () Apresenta uma notação simples, ilustrando as colaborações que concentram os casos de uso com suas interações com as classes.
2. () É que representa os objetos com seus estados significantes e as transições entre os estados.
3. () Os casos de uso são utilizados para capturar os requisitos não funcionais do sistema, sendo usados para definir a arquitetura e o desempenho pretendidos para cada caso de uso.
4. () Demonstra o comportamento de um objeto através de um conjunto de estados e suas transições em um determinado instante de tempo de execução do sistema.

Assinale a alternativa que indica a sequência correta.

- a) 1 – V; 2 – V; 3 – V, 4 – V.
- b) 1 – F; 2 – F; 3 – F, 4 – F.
- c) 1 – F; 2 – F; 3 – F, 4 – V.
- d) 1 – F; 2 – V; 3 – V, 4 – V.
- e) 1 – V; 2 – F; 3 – V, 4 – F.

5. Na modelagem orientada a objetos com UML, adota-se o Diagrama de Classes como sendo a principal técnica de modelagem estática para especificar a atividade de Análise. Posteriormente, na atividade de Projeto, refina-se o Diagrama de Classes com detalhes em conformidade com as tecnologias que serão utilizadas para implementar o software.

Sobre o Diagrama de Classes, julgue as sentenças a seguir:

- I. Uma Classe é representada por uma elipse com um nome que identifica a sua funcionalidade.
- II. Os relacionamentos utilizados no Diagrama de Classes são apenas associação e dependência.
- III. O nome de um atributo é declarado por um substantivo, usando a convenção de letras minúsculas e maiúsculas, e para palavras compostas usa-se concatená-las, sendo que a partir da segunda palavra inicia-se com letra maiúscula, por exemplo, "horaTermينو".
- IV. O nome de uma operação é declarado por um verbo, usando a convenção de letras minúsculas e maiúsculas, por exemplo, "recuperarAluno".

Estão corretos os itens:

- a) I e II.
- b) II e III.
- c) III e IV.
- d) I, III e IV.
- e) I, II, III e IV.

Referências

BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas com UML**. 2. ed. Rio de Janeiro: Elsevier, 2007.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML: guia do usuário**. Rio de Janeiro: Elsevier, 2000.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML: guia do usuário**. 2. ed. Rio de Janeiro: Elsevier, 2006.

CATARINO, Iolanda Claudia Sanches. **Análise de Sistemas**. Webaula unidades 1 e 2 (Especialização em Tecnologias para Aplicações Web) – Universidade Norte do Paraná: Londrina, 2012.

GUEDES, Gilleanes T. A. **UML: uma abordagem prática**. 3. ed. São Paulo: Novatec, 2008.

MEDEIROS, Ernani Sales de. **Desenvolvendo software com UML 2.0: definitivo**. São Paulo: Pearson, 2004.

PENDER, TOM. **UML Bible**. Indianapolis, Indiana: John Wiley & Sons, 2003.

SOMMERVILLE, Ian. **Engenharia de software**. 9. ed. São Paulo: Pearson, 2011.

Anotações

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

Anotações

This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

Anotações

[illegible]

Anotações

This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

Anotações

[illegible]

Anotações

This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

ISBN 978-85-8482-910-1



9 788584 829101 >