

Análise orientada a objetos I

Análise orientada a objetos I

Polyanna Pacheco Gomes
Everson Matias de Moraes
Marco Hisatomi

© 2018 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
Danielly Nunes Andrade Noé
Grasiele Aparecida Lourenço
Isabel Cristina Chagas Barbin
Lidiane Cristina Vivaldini Olo
Thatiane Cristina dos Santos de Carvalho Ribeiro

Revisora Técnica

Nathália dos Santos Silva

Editoração

Adilson Braga Fontes
André Augusto de Andrade Ramos
Leticia Bento Pieroni
Lidiane Cristina Vivaldini Olo

Dados Internacionais de Catalogação na Publicação (CIP)

F128a Fabris, Polyanna Pacheco Gomes
Análise orientada a objetos I / Polyanna Pacheco Gomes
Fabris, Everson Matias Moraes, Marco Hisatomi. – Londrina:
Editora e Distribuidora Educacional S.A., 2018.
128 p.

ISBN 978-85-522-0300-1

1. Programação orientada a objetos. 2. Análise de sistemas. I. Moraes, Everson Matias II. Hisatomi, Marco. III. Título.

CDD 005.12

2018
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 Visão da análise de sistemas	7
Seção 1 - Visão de sistema de informação	10
1.1 Sistema de informação	10
1.2 História da engenharia de software	12
1.3 Papel do analista de sistemas	13
Seção 2 - Princípios da modelagem de software	16
2.1 Modelos	16
2.2 Modelagem de software	16
2.3 Elementos textuais e os diagramas (documentação)	18
Seção 3 - Princípios do paradigma orientado a objetos	23
3.1 Introdução ao paradigma	23
3.2 Classe, objeto, atributo e método	24
3.3 Encapsulamento, polimorfismo e herança	26
Unidade 2 Modelos de processos e UML	35
Seção 1 - Introdução ao conceito de processos de software e seus modelos	38
Seção 2 - Processo unificado (PU): fases e atividades	42
Seção 3 - Metodologias ágeis	48
Seção 4 - Unified Modeling Language (UML)	60
Unidade 3 Visão da análise de sistemas	75
Seção 1 - Introdução ao caso de uso	78
1.1 Caso de uso	78
1.2 Identificando um caso de uso	82
Seção 2 - Diagrama de caso de uso	84
2.1 Cenário	85
2.2 Ator	85
2.3 Use case	87
2.4 Comunicação	88
Unidade 4 Modelagem de classes de análise	101
Seção 1 - Modelagem de classes de análise: conceito, componentes e notação do diagrama de classes	104
1.1 Modelagem de classes de análise	104
Seção 2 - Técnicas para identificação de classes e construção do diagrama de classes	117
2.1 Taxonomia de conceitos	117
2.2 Análise Textual de Abbot (ATA)	117
2.3 Construção do modelo de classes	120

Apresentação

Caro aluno, neste livro, apresentaremos conceitos importantes para o seu conhecimento, esteja você atuando ou não na área de desenvolvimento. Abordaremos alguns conceitos que, por meio de um único estudo de caso da construtora “More Feliz”, facilitarão a aplicação no dia a dia de todo o conteúdo de forma integrada.

Na Unidade 1, iniciaremos com a abordagem do papel do analista de sistemas e suas principais atividades, na sequência, apresentaremos os paradigmas da orientação a objeto, suas características e conceitos, abrindo um horizonte para a compreensão de alguns diagramas que serão mostrados nesta unidade e refinados nas unidades seguintes, independente do modelo de processo adotado.

Na Unidade 2, apresentaremos os principais modelos de processos existentes e utilizados por muitas empresas. Nesta abordagem, será possível identificar dois cenários que geram uma dúvida sobre qual modelo trabalhar, tradicional ou ágil? Essa dúvida será possível resolver com um estudo mais aprofundado, vivência no modelo e a aplicação da melhoria contínua. Não abordaremos no livro, mas cabe um estudo complementar em modelo híbrido, que é a utilização de práticas do tradicional e ágil. O maior objetivo nesta unidade, é fazer com que você saiba da existência desses modelos e consiga abstrair o que há de melhor em cada um. Vamos abordar também a UML (linguagem de modelagem unificada) que visa facilitar a compreensão entre os envolvidos do sistema que está sendo desenvolvido.

Já a Unidade 3 abordará com mais riqueza de detalhes o diagrama de caso de uso, seu conceito, como empregar a notação, como este diagrama pode contribuir durante o processo de desenvolvimento de software. E com base no estudo de caso, treinar sua aplicação.

E, por fim, na Unidade 4, trabalharemos com mais detalhes o diagrama de classe de análise, que é um dos diagramas da UML, considerado o coração do processo de modelagem de objetos, tendo como objetivo, a descrição do problema representado pelo sistema a ser desenvolvido em um nível alto de abstração.

Visão da análise de sistemas

Everson Matias de Moraes

Objetivos de aprendizagem

- Introduzir a visão da área de sistema de informação com foco na engenharia de software.
- Introduzir conceitos básicos do paradigma de orientação a objetos.

Seção 1 | Visão de sistema de informação

A Seção 1 aborda os princípios do sistema de informação com foco na engenharia de software, evidenciando sua história e o papel do analista de sistemas.

Seção 2 | Princípios da modelagem de software

A Seção 2 demonstra características de modelos de engenharia de software, exemplificando elementos da análise de sistemas estruturada e da orientada a objetos.

Seção 3 | Princípios do paradigma orientado a objetos

A Seção 3 descreve características básicas do paradigma orientado a objetos na área de engenharia de software, assim como a apresentação do estudo de caso da construtora fictícia "More Feliz".

Introdução à unidade

O estudo sobre métodos de análise de sistemas vem sendo realizado há vários anos, devido à necessidade de estruturação da informação com foco no modelo computacional, a partir de processos e rotinas que visam facilitar a organização e a otimização da informação. A grande importância para essa estruturação é o entendimento aprofundado sobre a base do negócio em questão, para a elaboração de um projeto de software. Nesse contexto, nasce a necessidade de modelos e ferramentas para a construção e a compreensão de processos independente de cada área, na qual surgem direcionamentos descritos por diferentes autores. Nesta unidade, a abordagem será sobre a visão de forma geral da análise de sistemas, como atividades e atribuições do analista de sistemas, histórico, características e conceitos de orientação a objetos e finalizando com a descrição de um estudo de caso da construtora fictícia "More Feliz".

Seção 1

Visão de sistema de informação

Introdução à seção

Nesta seção, serão abordados os princípios do sistema de informação com foco na engenharia de software, evidenciando sua história e o papel do analista de sistemas.

1.1 Sistema de informação

Segundo Bezerra (2007), um sistema de informação é definido por uma combinação de pessoas, dados, processos, interfaces, redes de comunicação e tecnologia, que interagem com o objetivo de dar suporte e melhorar o processo em organizações, assim como prover vantagens competitivas no mundo moderno. A partir dessa definição, podemos trabalhar com três visões, sendo:

- Processo.
- Tecnologia.
- Interação humano-computador.

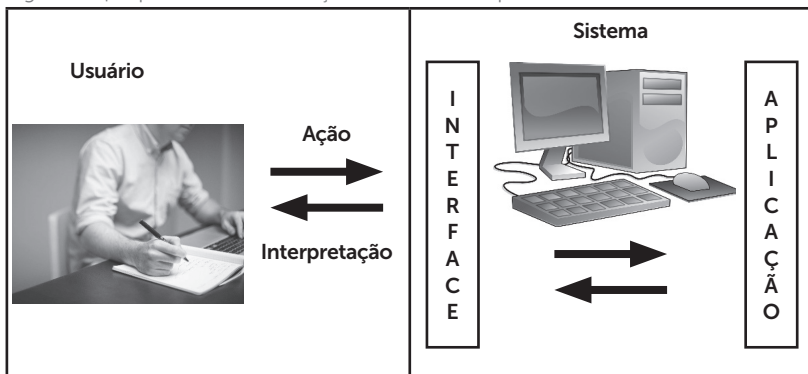
Na visão de processo, identificamos o funcionamento das rotinas com todos os elementos que formam um determinado negócio. Cada negócio em si, pode se tornar mais competitivo ou, simplesmente, ter seus processos automatizados. Nessa visão, a importância da análise de sistemas, está diretamente ligada ao entendimento de suas rotinas e, para isso, torna-se necessário o detalhamento formal de seus processos, contribuindo para um nível de abstração que torne possível uma automatização eficiente, ou seja, a identificação das principais partes necessárias para a estruturação do produto como um software.

Na parte de tecnologia, são envolvidos tanto o hardware como o software, que formam a base de um sistema computacional. Uma definição encontrada sobre tecnologia a define como um produto da ciência e da engenharia que envolve um conjunto de instrumentos, métodos e técnicas que visam à resolução de problemas

(SIGNIFICADOS, 2017). Dessa forma, para se obter uma análise eficiente de um determinado problema, é necessário o conhecimento prévio de toda a infraestrutura tecnológica que do escopo de abordagem do negócio envolvido.

Já em interação humano-computador, encontramos aspectos que vão além do aparato tecnológico. Por isso, além da eficiência na automatização dos processos que formam o software, o objetivo é construir sistemas com boa usabilidade, considerando a satisfação do usuário. Para essa satisfação, segundo Rocha e Baranauskas (2000), devem ser incluídos aspectos multidimensionais, tais como os cognitivos e os emocionais do usuário. Na Figura 1.1, ilustramos o processo de interação humano-computador, segundo Prates e Barbosa (2003, p. 246):

Figura 1.1 | O processo de interação humano-computador



Fonte: elaborada pelo autor.:

Portanto, para a realização de uma análise que atenda aos padrões atuais de desenvolvimento, é necessário ter como base metodologias que garantam um bom nível de abstração, considerando a complexidade do mundo real para a composição de um projeto de software de qualidade desejável e eficiente, tanto em processos como em sua usabilidade.

As metodologias atuais têm como base os conhecimentos adquiridos pelos profissionais denominados “analistas de sistemas” durante a história da evolução no campo da engenharia de software, por isso, dá-se a importância de um estudo sobre essa história, assim como o papel do analista.

1.2 História da engenharia de software

Um ponto importante desse contexto é conhecer a sua história, que será descrita a seguir.

Desde seus princípios, devido à necessidade de melhorar o seu dia a dia, as pessoas sempre buscam algum meio de automatizar seus processos diários, por exemplo, a criação da roda, que foi uma inovação tecnológica incrível, muito básica, mas de grande importância. Nesse caminho, temos inúmeros exemplos básicos, como a máquina a vapor, a eletricidade, o motor a explosão, o telefone, o rádio, entre outros. Logo após, iniciam inovações mais interessantes e complexas, pelo menos aos olhares atuais, como o avião, a válvula eletrônica, o avião a jato e chegando a um importante elemento do contexto direto ao estudo deste livro, sendo o computador e válvulas, em 1946.

Por volta da década de 1940, o homem inicia uma busca que parece não ter fim, que são as melhorias das rotinas e dos processos diários por meio da tecnologia computacional. Para isso, surge um campo que estudará métodos e maneiras para a elaboração de modelos computacionais, criando, assim, a área de análise de sistemas.

Para evidenciar essa importante história, um dos principais autores da área, o Prof. Barry Boehm, descreve a evolução da engenharia de software desde os anos 1950 até a modernidade, conforme a seguir (CREATIVANTE, 2011):

- Os anos 1950 foram marcados como a engenharia do hardware, devido ao entendimento da época prevalecer o contexto do hardware sobre o software, como: “produza software como se produz hardware”.
- Nos anos 1960, a caracterização foi em torno do artesanato do software. Inicia a importância da diferenciação de visão do software para o hardware, devido à facilidade de o software sofrer modificações em relação ao hardware e, consequentemente, surge o termo *codeandfix* (codifique e resolva).
- Já nos anos 1970, o contexto foi sobre os processos de formalidade e do waterfall, na qual se tem início ao enfoque nos processos em que a codificação era mais organizada, sendo precedida

pelo projeto com base na engenharia de requisitos, surgindo, assim, os métodos formais, que focavam na validação dos programas por prova matemática ou por cálculo de programação, e os métodos menos formais, que era a mistura de métodos técnicos com gerenciais.

- O foco dos anos 1980 era com as questões de produtividade e escalabilidade, sendo representada pelo número de iniciativas desenvolvidas para enfrentar os problemas enfrentados nos anos 1970.

- Para os anos 1990, o destaque foram os processos concorrentes versus os sequenciais, com a forte tendência dos métodos orientados a objetos, fortalecidos por avanços dos padrões de projetos, arquiteturas de software e as linguagens de descrição de arquiteturas. Nesse momento, nasce o desenvolvimento da UML (*Unified Modeling Language*) e se tem a expansão continuada da Internet, que fortalecem os métodos orientados a objetos quanto à importância do software em relação à concorrência de mercado. Como exemplos, outras questões também entram em destaque, como o *time-to-market*, a área de interação humano-computador e o desenvolvimento *open-source*.

- Nos anos 2000, a grande caracterização foi em torno da agilidade e do valor para a construção de um software, justificados pelo avanço nas mudanças das tecnologias e em organizações com suas fusões, aquisições e o surgimento das startups. Nasce a necessidade da abordagem em métodos ágeis para a gerência de projetos.

- Portanto, a partir de 2010, a visão foi principalmente permeada pelos fenômenos da conectividade global.

1.3 Papel do analista de sistemas

Atualmente, as atividades e as atribuições do analista de sistemas giram em torno de padrões de projetos, fundamentados em diversas metodologias com o apoio de ferramentas. Esse profissional tem como base o mundo dos negócios. Ao iniciar um projeto, é necessário que tenha a capacidade de abstrair os elementos essenciais do negócio a ser analisado, com o objetivo de concretizar um eficiente projeto de software.



Para realizar suas funções, o Analista de Sistemas deve entender não só do domínio do negócio da organização, mas também ter conhecimento dos aspectos mais complexos da computação. Nesse sentido, o Analista de Sistemas funciona como um tradutor, que mapeia informações entre duas “linguagens” diferentes: a dos especialistas de domínio e a dos profissionais técnicos da equipe de desenvolvimento. Uma característica importante que um analista de Sistemas deve ter é a capacidade de comunicação, tanto escrita como falada. Ele é um agente facilitador da comunicação entre os clientes e a equipe técnica. Muitas vezes, as capacidades de comunicar agilmente e de ter um bom relacionamento interpessoal são mais importantes para o analista do que o conhecimento tecnológico. Outra característica necessária a um analista é a ética profissional. Muitas vezes, o analista de sistemas está em contato com informações sigilosas e estratégicas dentro da organização na qual está trabalhando. Os analistas de Sistema têm acesso a informações como preços de custo de produtos, margens de lucro, algoritmos proprietários etc. (VALENTE, 2007, p. 38)



Questão para reflexão

Destaque três questões importantes com relação ao papel do analista de sistemas, para que justifique esta profissão no campo de negócio de sua realidade.



Para saber mais

Acesse o link: Disponível em: <<http://www.infobitsolucoes.com/antigos/Pos/Metodologia%20de%20Análise%20de%20Sistemas.pdf>>.
Acesso em: 27 set. 2017.

Atividades de aprendizagem

1. Uma definição encontrada sobre tecnologia a define como um produto da ciência e da engenharia que envolve um conjunto de _____, _____ e _____ que visam à resolução de problemas (SIGNIFICADOS, 2017).

Observe o texto e assinale a alternativa que preencha a lacuna de forma correta:

- a) Instrumentos, métodos e técnicas.
- b) Códigos, infraestrutura e comunicação.
- c) Códigos, infraestrutura e técnicas.
- d) Instrumentos, infraestrutura e comunicação.
- e) Códigos, métodos e comunicação.

2. Com relação à evolução da engenharia de software, identifique a década em que o destaque foram os processos concorrentes versus os sequenciais, com a forte tendência dos métodos orientados a objetos, fortalecidos por avanços dos padrões de projetos, arquiteturas de software e as linguagens de descrição de arquiteturas. Nesse momento, nasce o desenvolvimento da UML (*Unified Modeling Language*) e se tem a expansão continuada da Internet, que fortalecem os métodos orientados a objetos quanto à importância do software em relação à concorrência de mercado. Como exemplos, outras questões também entram em destaque, como o *time-to-market*, a área de interação humano-computador e o desenvolvimento open-source (CREATIVANTE, 2011).

- a) Década de 1950.
- b) Década de 1960.
- c) Década de 1970.
- d) Década de 1980.
- e) Década de 1990.

Seção 2

Princípios da modelagem de software

Introdução à seção

Nesta seção, serão descritas características de modelos de engenharia de software, exemplificando elementos da análise de sistemas estruturada e da orientada a objetos.

2.1 Modelos

Independentemente do tipo de projeto, devemos ter como base um planejamento estruturado em modelos que venham caracterizar a representação sistêmica dos processos que envolvem o negócio, garantindo, assim, a melhor abstração possível dos pontos fundamentais que formarão o resultado pretendido.

Nesse sentido, Bezerra (2007) destaca algumas razões para a construção de modelos:

- Gerenciamento da complexidade inerente ao desenvolvimento de software.
- Comunicação entre as pessoas envolvidas.
- Redução dos custos no desenvolvimento.
- Predição do comportamento futuro do sistema.

A partir desse contexto, nasce o termo modelagem de software, que será conceituado a partir do próximo item.

2.2 Modelagem de software

O conceito de modelagem de software está ligado à representação de forma estruturada e contextualizada das tarefas essenciais de um processo que comporão um projeto, o que contribuem muito para o entendimento real do escopo.

De acordo com Booch, Rumbaugh e Jacobson (2006), um modelo é uma simplificação da realidade e podem ser estruturais, com ênfase à organização do sistema, ou comportamentais, com ênfase à dinâmica do sistema, devendo ter como base os seguintes objetivos:

- Ajudar a visualizar o sistema como ele é ou como desejamos que seja.
- Permitir especificar a estrutura ou o comportamento de um sistema.
- Proporcionar um guia para a construção do sistema.
- Documentar as decisões tomadas.

Uma outra definição de Bezerra (2007) é que um modelo pode ser visto como uma representação idealizada de um sistema a ser construído e deve-se levar em conta algumas razões que os justifiquem, como:

- Gerenciamento da complexidade inerente ao desenvolvimento de software.
- Comunicação entre as pessoas envolvidas.
- Redução dos custos no desenvolvimento.
- Predição do comportamento futuro do sistema.

Dessa forma, a modelagem consiste no uso de notações gráficas e textuais com o objetivo de construir modelos que representam as partes essenciais de um sistema computacional, considerando-se diversas perspectivas (BEZERRA, 2007).

Nesse contexto, a modelagem pode ser de forma gráfica ou textual, por exemplo, alguns elementos da área de arquitetura: memorial descritivo, planta arquitetônica e a maquete de uma construção. No campo da computação, temos diversos autores com inúmeras formas de representação, incluindo elementos textuais e os gráficos, que na computação é comum serem chamados de diagramas. Os diagramas facilitam o entendimento de forma mais explícita dos

aspectos fundamentais para um bom desenvolvimento de software, contribuindo para a abstração de questões mais complexas, que tornam a visão do projetista mais refinada aos detalhes relevantes para a composição do projeto.

2.3 Elementos textuais e os diagramas (documentação)

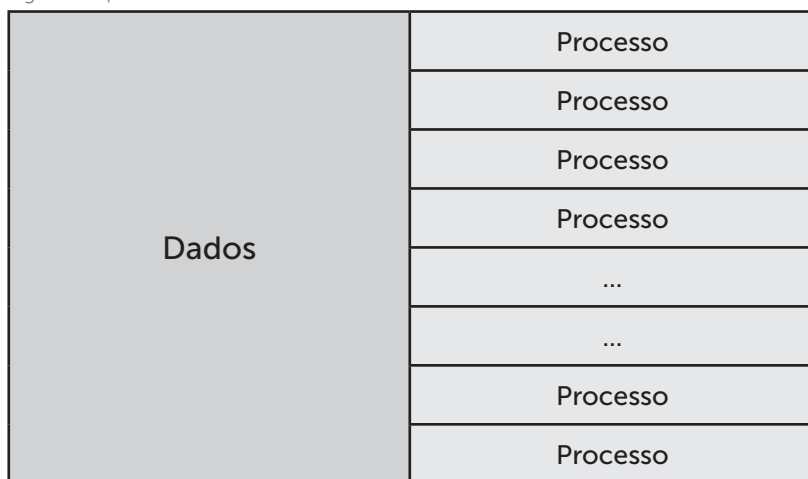
No decorrer da história da computação, foi notória a concepção de vários tipos de documentações para a descrição do processo de construção de um software. Dessa forma, vários autores criaram técnicas para o aperfeiçoamento de uma modelagem para garantir a eficiência da estruturação de um projeto. Essas técnicas têm como objetivo tornar explícita a representação da abstração de um processo, o que resulta na necessidade de elementos textuais e diagramas.

Para Bezerra (2007), os diagramas seguem padrões lógicos, em que a representação de uma coleção de elementos gráficos possui um significado predefinido, descrevendo uma representação concisa do sistema. Para apoio a essa representação, os modelos também são compostos de informações textuais. Assim, os diagramas associados à informação textual, formam a documentação de um modelo computacional.

Um fato importante a ser destacado neste ponto, é que na história da engenharia de software houve duas grandes divisões que envolvem paradigmas representativos de diferentes particularidades. A grande divisão gira em torno da análise estruturada e da análise orientada a objetos.

A análise estruturada tem como foco os processos com relação aos dados, na qual, seu detalhamento é *top-down*, do nível macro para os menores detalhes, conforme o exemplo da Figura 1.2.

Figura 1.2 | Base da análise de sistemas estruturada



Fonte: elaborada pelo autor.

Para a exemplificação de diagramas e informações contextuais da análise estruturada, apresentamos uma lista com itens documentacionais utilizados por diversos autores, como Gane e Sarson, Yourdon, Constantine e Page-Jones, que são descritas a seguir (IGLESIAS, 2017):

Modelo ambiental (contexto do sistema):

- Definição de objetos.
- Lista de eventos.
- Diagrama de contexto.

Modelo comportamental (ações do sistema):

- Diagrama de fluxos de dados (DFD).
- Dicionário de dados (DD).
- Diagrama de entidades e relacionamentos (DER).
- Especificação de processos (EP).
- Diagrama de transição de estados (DTE).

Diagramas:

- Diagrama de contexto.

- Diagrama de fluxos de dados.
- Diagrama entidade e relacionamento.
- Lista de eventos.
- Dicionário de dados.
- Tabela de decisão.
- Árvore de decisão.
- Diagrama de transição de estados.

O foco da análise orientada a objetos está em um elemento, o “objeto” como uma unidade autônoma, que contém seus próprios dados, que são manipulados pelos processos definidos para o objeto e que interage com outros objetos.

Figura 1.3 | Base da análise de sistemas orientada a objetos

Objeto (dados)	Processo
	Processo
Objeto (dados)	Processo
	Processo
...	...
	...
Objeto (dados)	Processo

Fonte: elaborada pelo autor.

Nesse caso, a exemplificação será em torno da UML, sendo uma linguagem de modelagem para projetos de sistemas orientados a objetos, e seus diagramas são divididos em estruturais e comportamentais:

Diagramas estruturais:

- Diagrama de classe.
- Diagrama de objeto.

- Diagrama de componentes.
- Diagrama de implantação.
- Diagrama de pacotes.
- Diagrama de estrutura.

Diagramas comportamentais:

- Diagrama de caso de uso (use case).
- Diagrama de máquina de estados.
- Diagrama de atividades.
- Diagrama de interação, que é dividido em:
- Diagrama de sequência.
- Diagrama geral interação.
- Diagrama de comunicação.
- Diagrama de tempo.



Questão para reflexão

Faça uma pesquisa sobre os diagramas de classe e o de fluxo de dados.



Para saber mais

Acesse o link: **UML: guia do usuário**

Disponível em: <https://books.google.com.br/books?id=ddWqxcDKGF8C&printsec=frontcover&dq=UML:+guia+do+usu%C3%A1rio&hl=en&sa=X&redir_esc=y#v=onepage&q=UML%3A%20guia%20do%20usu%C3%A1rio&f=false>. Acesso em: 27 set. 2017.

Atividades de aprendizagem

1. Identifique o único diagrama que faz parte da UML (*Unified Modeling Language*):

- Diagrama de fluxos de dados.
- Dicionário de dados.
- Diagrama de classe.
- Especificação de processos.
- Diagrama de transição de estados.

2. Identifique o único diagrama que faz parte do modelo comportamental:

- a) Diagrama de pacotes.
- b) Diagrama de implantação.
- c) Diagrama de classe.
- d) Diagrama de objeto.
- e) Diagrama de caso de uso.

Seção 3

Princípios do paradigma orientado a objetos

Introdução à seção

Nesta seção serão descritas características básicas do paradigma orientado a objetos na área de engenharia de software, assim como a apresentação do estudo de caso da construtora fictícia “More Feliz”.

3.1 Introdução ao paradigma

Bezerra (2007) relata que Alan Kay contextualizou uma analogia biológica para a comparação de como seria um software. Essa analogia permite descrever que cada “célula” interagiria com outras células através do envio de mensagens para realizar um procedimento comum, na qual, cada célula se comportaria como uma unidade autônoma. Com base nesse conceito, o objetivo foi construir um software a partir de agentes autônomos que interagissem entre si, originando os princípios da orientação a objetos:

- Qualquer coisa é um objeto.
- Objetos realizam tarefas através da requisição de serviços a outros objetos.
- Cada objeto pertence a uma determinada classe.
- Uma classe agrupa objetos similares.
- A classe é um repositório para comportamento associado ao objeto.
- Classes são organizadas em hierarquias.

O paradigma de orientação a objetos visualiza um software como uma coleção de agentes interconectados chamados de objetos, sendo que cada objeto é responsável por realizar sua tarefa e que de forma colaborativa resultam às funcionalidades de um sistema computacional (BEZERRA, 2007).

Um grande impacto na abordagem da orientação a objetos é trabalhar a modularização, proporcionando o aproveitamento de código, dessa forma, quando um modelo já é estruturado de forma particionada, pode se ter um melhor gerenciamento de sua complexidade. O processo de modularização do mundo real, com relação à resolução de um determinado problema, deve consistir em uma abstração que deriva um conjunto de componentes de software que seja fiel na representação do domínio desse problema. Por isso, é necessário o entendimento dos conceitos que regem a orientação a objetos.

3.2 Classe, objeto, atributo e método

Para um entendimento de conceitos de análise orientada a objetos, é importante entender que uma classe é uma abstração que define um tipo de objeto e que esse mesmo tipo possui seus atributos, que pelos métodos são definidas as ações que esse tipo será capaz de realizar.

Figura 1.4 | Conceitos de orientação a objetos

CLASSE	OBJETO
ATRIBUTO	MÉTODO

Fonte: elaborada pelo autor.

Bezerra (2007) explica que as coisas que formam o mundo real são denominadas objetos e que as pessoas os agrupam para entendê-los, nesse sentido, a descrição de um grupo de objetos é denominada classe de objetos. Assim, uma classe é a descrição dos atributos e serviços de um grupo de objetos.

Um objeto é denominado uma instância de uma classe, na qual, esse mesmo objeto possui propriedades que são seus atributos, e que identificam o estado do objeto. A partir de sua composição, o objeto

envia uma mensagem a outro objeto para que ele execute uma ação ou operação, dessa forma, uma mensagem é uma “chamada de um método” com o objetivo de realizar alguma tarefa.

Machado (2017) descreve que nesse modelo a abstração é muito importante por lidar com a representação de um objeto real. É necessário criar uma identidade única dentro do sistema para que não exista conflito, assim como identificar suas características essenciais, que em orientação a objetos, denomina-se propriedades, por exemplo, as propriedades de um objeto “pessoa” podem ser: tamanho, peso e idade. Assim, com base na identidade e nas propriedades do objeto, podem ser definidas suas ações, chamadas de métodos, por exemplo: “come”, “anda”.

Como o conceito de orientação a objetos está ligado à resolução de problemas diários de nossa realidade, um bom caminho para o seu entendimento é utilizar exemplos básicos que vivemos, como a seguir:

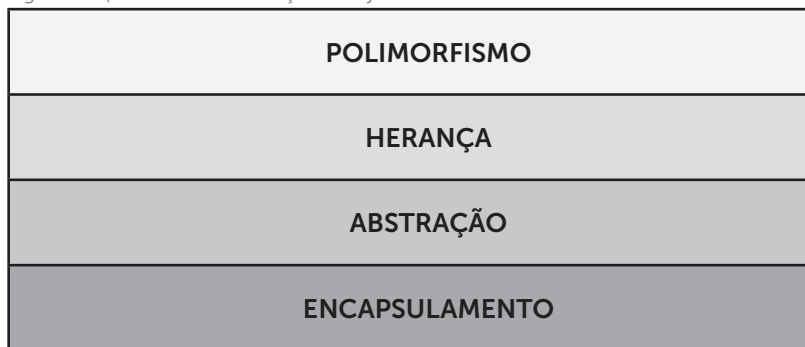
Uma pessoa atende a mensagens (requisições) para realizar um serviço. Por sua vez essa mesma pessoa envia mensagens a outras para que estas realizem outros serviços. Por que não aplicar essa mesma forma de pensar a modelagem de sistemas? Vejamos o exemplo: João quer comprar uma pizza, e por estar ocupado, pede a pizza por telefone. João informa ao atendente Luís seu nome, a pizza e o seu endereço. Por sua vez Luís avisa ao pizzaiolo Antonio qual a pizza a ser feita. Uma vez feita a pizza Luís chama Roberto, o motoboy, para entregá-la na casa do João. O objetivo de João foi atingido através da colaboração de diversos agentes, que são denominados objetos. Estes objetos foram: Luís, Antônio e Roberto. Embora todos tenham trabalhado em suas funções de forma individual, alcançaram o objetivo cooperando conjuntamente. O comportamento do pizzaiolo Antonio é o mesmo esperado de qualquer um de sua profissão (fazer pizza). Logo, Antonio é um objeto da classe Pizzaiolo. E todos os objetos pertencem às classes maiores por serem seres humanos, animais, mamíferos e assim por diante, formando as hierarquias. (VALENTE, 2007, p. 91)

”

3.3 Encapsulamento, polimorfismo e herança

Os pilares da orientação a objetos são: abstração, encapsulamento, herança e polimorfismo, mas, como já vimos conceitos relacionados à abstração, nesta seção, vamos conhecer algumas definições de encapsulamento, herança e polimorfismo, ilustrados na Figura 1.5.

Figura 1.5 | Pilares da orientação a objetos



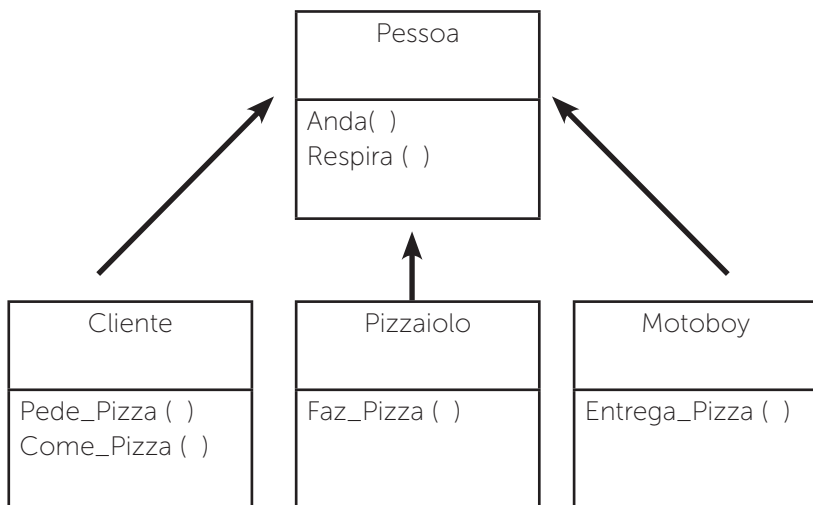
Fonte: elaborada pelo autor.

O encapsulamento é uma forma de limitar o acesso ao comportamento interno de um objeto, ao enviar uma solicitação a outro objeto para realizar alguma tarefa, simplesmente envia uma mensagem, dessa forma, o método que o objeto requisitado usa para realizar a tarefa não é transparente ao objeto requisitante, criando uma segurança pelo fato de esconder as propriedades, como se fosse uma espécie de “caixa preta” (BEZERRA, 2007).

Já o polimorfismo é a capacidade de objetos de classes diferentes responderem a mesma mensagem de diferentes maneiras, ou seja, as vezes é necessário que as ações para um mesmo método sejam diferentes, consistindo na alteração do funcionamento interno de um método herdado de um objeto pai.

Na herança, as classes semelhantes são agrupadas em níveis de hierarquias, herdando características de classes de níveis acima, conforme o exemplo da Figura 1.6.

Figura 1.6 | Exemplo de herança



Fonte: elaborada pelo autor.

De acordo com o contexto, é possível identificar que a análise orientada a objetos possibilita uma organização estrutural que contribui para o reaproveitamento de códigos, assim como da facilidade de se herdar atributos e comportamentos de seus objetos.

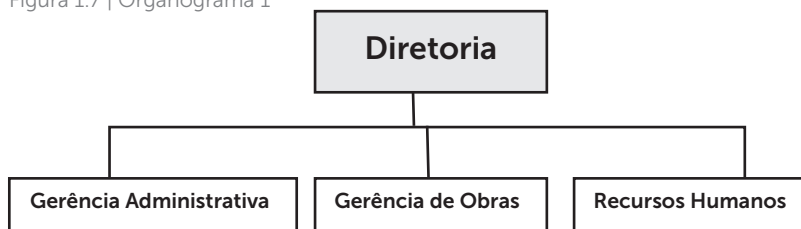
Para a continuação das futuras unidades, com aplicações práticas, a seguir ilustraremos um estudo de caso fictício, denominado Construtora “More Feliz”.

ESTUDO DE CASO CONSTRUTORA “MORE FELIZ”

A nossa história teve início em 1979, quando nosso fundador Luiz Bandeira iniciou um processo de construção de algumas pequenas casas, na cidade de Florianópolis, em Santa Catarina. Desde o começo, o objetivo foi executar um trabalho com qualidade, motivados pela convicção de que seria um bom caminho em rumo ao crescimento de forma sustentável.

Após sua fundação, a More Feliz foi crescendo aos poucos e conquistando o mercado regional com relação à construção de casas até o final da década de 1980, na qual, contava com aproximadamente 60 colaboradores. Nessa época, sua estrutura ainda era modesta, conforme o organograma 1 da Figura 1.7.

Figura 1.7 | Organograma 1

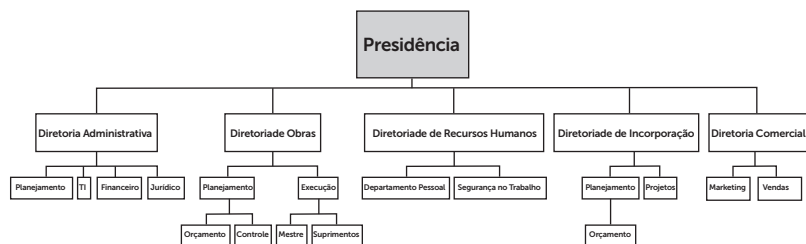


Fonte: elaborada pelo autor.

No início da década de 1990, inicia um projeto inovador para sua realidade, sendo a construção de edifícios residenciais. Dessa forma, a empresa se surpreendeu com seu gigantesco crescimento e houve a necessidade com urgência de sua estruturação na automatização de processos. No ano de 1993, seu fundador e presidente Luiz Bandeira, com sua equipe de gestores, decide contratar uma consultoria em tecnologia da informação para estruturar a primeira fase de informatização. Após o resultado do estudo preliminar do processo de informatização, a equipe gestora decidiu focar primeiramente em um sistema de gestão financeira, envolvendo a parte de compras, vendas, estoque e controle financeiro. Em 1998, houve uma nova estruturação de informatização, na qual, foram incluídas as seguintes áreas: recursos humanos, departamento pessoal, fiscal e contábil.

Atualmente, a empresa atua na área de construção civil, de comercialização e incorporação de imóveis residenciais e possui vários empreendimentos em desenvolvimento, como prédios residenciais, condomínios, loteamento e casas. Sua estrutura é formada por diversos departamentos, como recursos humanos, marketing, administrativo, obras, vendas, compras, com os mais diversos profissionais de exigência de sua área, sendo engenheiros, arquitetos, decoradores, pedreiros, assistentes, diretores, entre outros, totalizando aproximadamente 600 colaboradores e estruturado conforme o organograma 2 da Figura 1.8.

Figura 1.8 | Organograma 2



Fonte: elaborada pelo autor.

Devido ao setor ter apresentado um crescimento expressivo e o processo de canteiro de obras ser de difícil manuseio operacional administrativo, com o objetivo de alcançar dimensões com maior eficiência de controle e produtividade, nasce uma demanda com relação à informatização das rotinas e processos. A decisão da gestão da empresa junto aos consultores administrativos foi direcionar a equipe de TI no desenvolvimento inovador para automatizar o processo de manobras de controle das obras, atingindo suas rotinas de controle com relação aos recursos de terceiros, à alocação de equipamentos e ao planejamento e cronograma de obras correntes e futuras.



Questão para reflexão

Faça uma pesquisa sobre os diagramas de classe e o de fluxo de dados.



Para saber mais

Acesse o link: **Princípios de análise e projeto de sistemas com UML**

Disponível em: <https://books.google.com.br/books?id=elvJBwAAQBAJ&printsec=frontcover&dq=Princ%C3%ADpios+de+an%C3%A1lise+e+projeto+de+sistemas+com+UML&hl=en&sa=X&redir_esc=y#v=onepage&q=Princ%C3%ADpios%20de%20an%C3%A1lise%20e%20projeto%20de%20sistemas%20com%20UML&f=false>. Acesso em: 27 set. 2017.

Atividades de aprendizagem

1. _____ é uma forma de limitar o acesso ao comportamento interno de um objeto, ao enviar uma solicitação a outro objeto para realizar alguma tarefa, simplesmente envia uma mensagem, dessa forma, o método que o objeto requisitado usa para realizar a tarefa não é transparente ao objeto requisitante, criando uma segurança pelo fato de esconder as propriedades, como se fosse uma espécie de “caixa preta” (BEZERRA, 2007). Analise o texto e preencha a lacuna de forma correta:

- a) Abstração.
- b) Encapsulamento.
- c) Herança.
- d) Polimorfismo.
- e) Método.

2. _____ é a capacidade de objetos de classes diferentes responderem a mesma mensagem de diferentes maneiras, ou seja, às vezes é necessário que as ações para um mesmo método seja diferente, consistindo na alteração do funcionamento interno de um método herdado de um objeto pai.

Analise o texto e preencha a lacuna de forma correta:

- a) Abstração.
- b) Encapsulamento.
- c) Herança.
- d) Polimorfismo.
- e) Método.

Fique ligado

Esta unidade abordou as características da visão de sistemas de informação, história da engenharia de software e o papel do analista de sistemas, mas continue seus estudos com base nas referências aqui descritas.

Para concluir o estudo da unidade

Ao término desta unidade, você, aluno, teve a oportunidade de enriquecer seus conhecimentos com relação à visão de análise de sistemas no campo de engenharia de software, mas é importante que continue seus estudos utilizando os links e as bibliografias aqui apresentadas.

Atividades de aprendizagem da unidade

1. Na _____, as classes semelhantes são agrupadas em níveis de hierarquias, herdando características de classes de níveis acima. Analise o texto e preencha a lacuna de forma correta:

- a) Abstração.
- b) Encapsulamento.
- c) Herança.
- d) Polimorfismo.
- e) Método.

2. Identifique o único diagrama que faz parte da UML (*Unified Modeling Language*):

- a) Diagrama de caso de uso.
- b) Diagrama de fluxos de dados.
- c) Dicionário de dados.
- d) Especificação de processos.
- e) Diagrama de transição de estados.

3. Identifique o único diagrama que faz parte dos diagramas estruturais:

- a) Caso de uso.
- b) Máquina de estados.
- c) Atividades.
- d) Interação.
- e) Componentes.

4. Com relação à evolução da engenharia de software, identifique a década em que o foco eram as questões de produtividade e escalabilidade, sendo representada pelo número de iniciativas desenvolvidas para enfrentar os problemas enfrentados na década anterior:

- a) Década de 1950.
- b) Década de 1960.
- c) Década de 1970.
- d) Década de 1980.
- e) Década de 1990.

5. De acordo com os conceitos de orientação a objetos, identifique a alternativa que lista de forma correta os pilares da orientação a objetos:

- a) Abstração, encapsulamento, herança e polimorfismo.
- b) Use case, encapsulamento, herança e polimorfismo.
- c) Processos, interação, herança e polimorfismo.
- d) Abstração, encapsulamento, componentes e polimorfismo.
- e) Interação, composição, herança e polimorfismo.

Referências

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

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML: guia do usuário**. 2. ed. Rio de Janeiro: Elsevier, 2006.

CREATIVANTE. **A evolução da engenharia de software**. 2011. Disponível em: <<http://creativante.com/new/index.php/2013-02-03-19-36-05/2011/58-a-evolucao-da-engenharia-de-software>>. Acesso em: 6 ago. 2017.

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

LAUDON, K. C.; LAUDON, J. P. **Sistemas de informação gerenciais**. 11. ed. São Paulo: Pearson Prentice Hall, 2011.

MACHADO, H. **Os quatro pilares da programação orientada a objetos**. 2017. Disponível em: <<http://www.devmedia.com.br/os-4-pilares-da-programacao-orientada-a-objetos/9264>>. Acesso em: 13 ago. 2017.

PRATES, R. O.; BARBOSA, S. D. J. **Avaliação de interfaces de usuário: conceitos e métodos**. Anais do XXIII Congresso Nacional da Sociedade Brasileira de Computação. XXII Jornadas de Atualização em Informática (JAI). SBC'2003. Agosto de 2003.

ROCHA, H. V. da; BARANAUSKAS, M. C. C. **Design e avaliação de interfaces humano-computador**. São Paulo: IME-SP, 2000.

SIGNIFICADOS. Significados, conceitos e definições. 2017. Disponível em: <www.significados.com.br>. Acesso em: 12 ago. 2017.

VALENTE, C. **Metodologia de análise de sistemas**. 2007. Disponível em: <<http://www.infobitsolucoes.com/antigos/Pos/Metodologia%20de%20Análise%20de%20Sistemas.pdf>>. Acesso em: 5 ago. 2017.

Modelos de processos e UML

Polyanna P. Gomes Fabris

Objetivos de aprendizagem

- Compreender o que é um processo de software e seus principais modelos.
- Compreender as características do processo unificado e das metodologias ágeis.
- Compreender os conceitos e os contextos relacionados à UML.

Seção 1 | Introdução ao conceito de processos de software e seus modelos

Nesta seção, os conceitos relacionados aos processos e seus modelos são detalhados. As fundamentações teóricas destes conceitos têm como objetivo a clarificação para que as seções seguintes sejam compreendidas.

Seção 2 | Processo unificado: fases e atividades

Nesta seção, contextualizamos o processo unificado (UP) e seu conjunto de atividades para a transformação de requisito de usuário em um sistema de software, bem como suas características principais: iterativo e incremental, dirigido por casos de uso, centrado na arquitetura e focado no risco.

Seção 3 | Metodologias ágeis

A metodologia ágil aborda as melhorias do processo de desenvolvimento de software, de forma que seu desenvolvimento seja baseado na simplicidade, melhoria na comunicação e na interação dos envolvidos, aumento da qualidade e produtividade, valor agregado ao cliente. Nesta seção, focamos nosso estudo nas principais características das metodologias ágeis e descrevemos as duas principais: *Scrum* e *XP*.

Seção 4 | Unified Modeling Language (UML)

A notação utilizada pela UML é padronizada pela OMG (Object Management Group) e visa facilitar a compreensão de cada parte do sistema que está sendo modelado, por qualquer pessoa que tenha conhecimento sobre a linguagem. Entre as suas principais características, podemos citar: utilização de atores e caso de uso para demonstrar estruturas e fronteiras de uma aplicação, utilização de diagramas de classes com atributos, operações e relacionamento para representação da estrutura estática, entre outros. Nesta seção, abordaremos as técnicas de modelagens estruturais e comportamentais, detalhando-as e exemplificando-as.

Introdução à unidade

Caro aluno, nesta unidade, apresentaremos o conceito de processo de software e os principais modelos existentes atualmente.

A Seção 1 contextualiza os conceitos e os modelos de processo e descreve o estudo de caso usado para nossos exemplos.

O processo unificado (PU) é representado por uma estrutura composta por um conjunto de atividades que podem ser personalizadas de acordo com a necessidade do cliente e considerando os aspectos do projeto. O PU faz uso extensivo da *Unified Modeling Language* (UML) e possui uma instância específica, o RUP (*Rational Unified Process*, traduzido para o português para *Processo Unificado Rational*), que é um produto comercializado pela empresa IBM. A Seção 2 mostra o detalhamento sobre o PU e sua instância.

A metodologia ágil surgiu com o objetivo de melhorar o processo de desenvolvimento de software, tornando-o mais simples, aumentando a comunicação e a interação entre os envolvidos, organizar a gestão do projeto e aspectos técnicos, aumentar a qualidade e a produtividade. Tratamos deste tema na Seção 3.

A *Unified Modeling Language* (UML) aborda aspectos relacionados à documentação, padronização e modelagem de sistemas a serem desenvolvidos. Suas principais atividades são: especificar, documentar, estruturar e fornecer uma visão lógica do software. Seus diagramas visam capturar diferentes visões: estrutural e comportamental. A Seção 4 aborda o detalhamento das técnicas de modelagem estrutural e comportamental.

Seção 1

Introdução ao conceito de processos de software e seus modelos

Introdução à seção

Caro aluno, esta é a primeira seção da presente unidade. Aqui abordaremos alguns conceitos relacionados ao processo: o que é processo ou o que é um modelo de processo. Um processo de software pode ser descrito como um conjunto de atividades inter-relacionadas ou interativas, que transforma insumos (entradas) em produtos (saídas) (SOFTEX, 2017).

Para Pressman (2011), o processo é um conjunto de atividades, ações e tarefas executadas durante a criação de algum produto de trabalho, em que a atividade tem um objetivo a ser atingido, uma ação envolve um conjunto de tarefas que resultam em um determinado artefato e uma tarefa tem um objetivo pequeno. Processo é uma abordagem adaptável, possibilitando que o trabalho seja realizado de forma que a equipe possa selecionar e escolher o conjunto de ações e tarefas que são apropriados. O processo de software incorpora um conjunto de cinco atividades estruturais, são elas: comunicação, planejamento, modelagem, construção e emprego.

Quando falamos em modelo de processo de software, nos referimos a uma representação, abstração dos objetos e atividades que estão envolvidas no processo de software. Eles oferecem uma abordagem mais abrangente e fácil de demonstrar a gestão do processo de software e a evolução do projeto. Dentre os principais modelos existentes no mercado, podemos citar: sequencial, iterativo e incremental, cascata, ágil etc.

Com o intuito de promover e compartilhar conhecimento, a presente unidade focará os estudos no processo unificado (PU), metodologias ágeis e UML. A seguir, descreveremos um estudo de caso, no qual nos basearemos para alguns exemplos no nosso texto. Tenha uma ótima leitura!

Estudo de caso: “CONSTRUTORA MORE FELIZ”

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

A nossa história teve início em 1979, quando nosso fundador Luiz Bandeira iniciou um processo de construção de algumas pequenas casas, na cidade de Florianópolis, em Santa Catarina. Desde o começo, o objetivo foi executar um trabalho com qualidade, motivados pela convicção de que seria um bom caminho em rumo ao crescimento de forma sustentável.

Após sua fundação, a More Feliz foi crescendo aos poucos e conquistando o mercado regional com relação à construção de casas até o final da década de 1980, na qual contava com aproximadamente 60 colaboradores.

No início da década de 1990, inicia um projeto inovador para sua realidade: a construção de edifícios residenciais. Dessa forma, a empresa se surpreendeu com seu gigantesco crescimento e houve a necessidade com urgência de sua estruturação na automatização de processos. No ano de 1993, seu fundador e presidente Luiz Bandeira com sua equipe de gestores, decidem contratar uma consultoria em tecnologia da informação para estruturar a primeira fase de informatização. Após o resultado do estudo preliminar do processo de informatização, a equipe gestora decidiu focar primeiramente em um sistema de gestão financeira, envolvendo a parte de compras, vendas, estoque e controle financeiro. Em 1998, houve uma nova estruturação de informatização, na qual foram incluídas as seguintes áreas: recursos humanos, departamento pessoal, fiscal e contábil.

Atualmente, a empresa atua na área de construção civil, comercialização e incorporação de imóveis residenciais, e possui vários empreendimentos em desenvolvimento, como prédios residenciais, condomínios, loteamento e casas. Sua estrutura é formada por diversos departamentos, como recursos humanos, marketing, administrativo, obras, vendas, compras, com os mais diversos profissionais de exigência de sua área, sendo engenheiros, arquitetos, decoradores, pedreiros, assistentes, diretores, entre outros, totalizando aproximadamente 600 colaboradores.

Devido ao setor ter apresentado um crescimento expressivo e o

processo de canteiro de obras ser de difícil manuseio operacional administrativo e, ainda, com o objetivo de alcançar dimensões com maior eficiência de controle e produtividade, nasce uma nova demanda com relação à informatização das rotinas e processos. A decisão da gestão da empresa junto aos consultores administrativos foi direcionar a equipe de TI no desenvolvimento inovador para automatizar o processo de manobras de controle das obras, atingindo suas rotinas de controle com relação aos recursos de terceiros, à alocação de equipamentos e ao planejamento e cronograma de obras correntes e futuras.

Para o presente estudo de caso, foram identificados alguns casos de uso. Deste modo, vamos apresentar mais detalhes dos UCs a seguir para fins didáticos:

- Manter imóveis.
- Manter funcionário (o funcionário que está responsável pelo imóvel).

Para saber mais

CMMI. **CMMI® para Desenvolvimento - Versão 1.2**. Disponível em: <http://www.sei.cmu.edu/library/assets/whitepapers/cmmi-dev_1-2_portuguese.pdf>. Acesso em: 14 jul. 2017.

PRESSMAN, R. **Engenharia de Software**. 7. ed. Porto Alegre: AMGH, 2011.

SOMMERVILLE, I. **Engenharia de Software**. São Paulo: Pearson Prentice Hall, 2011.



Questão para reflexão

Após a leitura do estudo de caso proposto, lançamos o seguinte desafio para você: reflita sobre o contexto em que a “More Feliz” está organizada, ao desenvolver um software que automatize as rotinas operacionais, qual é o modelo de processo que agrega valor às entregas?

Compartilhe sua opinião com seu professor.

Atividades de aprendizagem

1. Com base no conteúdo abordado, identificamos que processo é:

- I) Um conjunto de atividades, ações e tarefas executadas durante a criação de algum produto de trabalho.
- II) Um conjunto de atividades inter-relacionadas ou interativas, que transforma insumos (entradas) em produtos (saídas).
- III) Uma abordagem adaptável, possibilitando que o trabalho seja realizado de qualquer forma e que a equipe realize de forma flexível, ou seja, apenas quando julgar necessário.

Com base nessas afirmativas, assinale a alternativa correta:

- a) Apenas a afirmativa I está correta.
- b) As afirmativas I e II estão corretas.
- c) As afirmativas I e III estão corretas.
- d) As afirmativas II e III estão corretas.
- e) Apenas a afirmativa III está correta.

2. A empresa AgilSoft é uma empresa conceituada e tem buscado realizar melhorias em seu processo de desenvolvimento de software através de modelos de referência no âmbito nacional e internacional, mas ainda tem dificuldades de capacitar seus colaboradores na definição de processo. Qual exemplo básico do seu dia a dia você poderia dar às equipes da AgilSoft a respeito de processo?

Seção 2

Processo unificado (PU): fases e atividades

Introdução à seção

O processo de desenvolvimento de software tem como um dos objetivos principais a organização das atividades, a fim de que todos os cenários a serem desenvolvidos sejam organizados, problemas sejam facilmente identificados e resolvidos, bem como a entrega de artefatos com qualidade e que tenham valor ao cliente. De acordo com Pressman (2011), o processo unificado (PU) é um aproveitamento dos melhores recursos e características dos modelos tradicionais de desenvolvimento de software. Este modelo de processo tem sua base estruturada sob os componentes e os interconectados via interfaces, utiliza-se da UML para elaboração dos artefatos do sistema.

O PU reconhece a importância da comunicação e de métodos para descrição da visão do cliente sobre determinado sistema, enfatizando o papel da arquitetura de software e sugerindo um modelo iterativo e incremental. Podemos resumir o PU nas seguintes características principais:

- Dirigido a casos de uso.
- Centrado na arquitetura.
- Modelo iterativo e incremental.
- Focado nos riscos.

a. Dirigido a casos de uso

O caso de uso representa um processo que é compreendido sob o ponto de vista do usuário, para o PU, um determinado conjunto de casos de uso deve realizar a definição e o esgotamento de todas as funcionalidades do sistema (WAZLAWICK, 2013).

b. Centrado na arquitetura

O PU orienta que uma sólida arquitetura deve ser desenvolvida para

o sistema. A arquitetura pode ser entendida como um modelo que define a estrutura da informação, as operações e a organização em componentes. A cada ciclo iterativo, as funcionalidades devem ser incorporadas à arquitetura (WAZLAWICK, 2013).

c. Iterativo e incremental

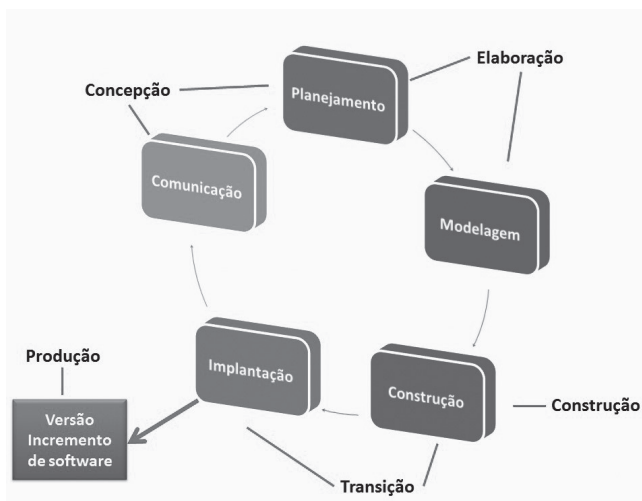
Para o PU, o desenvolvimento deve ser baseado em ciclos iterativos com uma duração predefinida e fixa, em que cada iteração realizará a incorporação das funcionalidades necessárias. Cada ciclo deve produzir um incremento no sistema, sendo resultado de um conjunto de tarefas realizadas pela equipe de desenvolvimento (WAZLAWICK, 2013).

d. Focado em riscos

A abordagem com foco em riscos é cada vez mais comum nos mais diversos modelos de desenvolvimento de sistemas. Ao usar esta abordagem, a imprevisibilidade sobre funcionalidades é reduzida, promovendo o entendimento e a aprendizagem do sistema, tratando-os primeiramente no ciclo de desenvolvimento corrente.

Para Pressman (2011), o PU é representado por cinco atividades metodológicas genéricas (comunicação, planejamento, modelagem, construção e implantação) que podem ser usadas para descrição de qualquer modelo de processo, já com relação às fases, é composto por concepção, elaboração, construção e transição. A imagem mostrada na sequência deste texto ilustra o processo unificado.

Figura 2.1| Processo unificado



Fonte: Pressman (2011).

a. Fase de concepção

O objetivo desta fase é a elaboração de uma visão mais abrangente do sistema. Nesta fase, são levantados os principais requisitos e a construção de um modelo conceitual preliminar é feito. Também, identificam-se os casos de uso de alto nível que implementam as funcionalidades solicitadas pelo cliente. Ainda como objetivo desta fase, temos o cálculo de esforço de desenvolvimento de casos de uso e a construção do plano de desenvolvimento. Quando necessário, podem existir implementações e testes, bem como elaboração de protótipos para redução de possíveis riscos ao projeto (WAZLAWICK, 2013).

Ao analisar o estudo de caso proposto na Seção 1, podemos concluir que para o projeto de desenvolvimento do software que fará a automatização de atividades e a integração entre os departamentos da construtora, teremos os seguintes objetivos a serem atingidos: escopo preliminar do sistema, versão preliminar dos requisitos, esforço e custos iniciais e identificação de riscos potenciais. Dentre os artefatos entregues ao final da fase, teremos: documento de visão, listagem dos riscos potenciais, plano de desenvolvimento e um glossário para entendimento comum do projeto.

b. Fase de elaboração

A fase de elaboração é composta por atividades de comunicação e modelagem de processos, refinando e expandindo os casos práticos preliminares, ampliando a representação da arquitetura e apresentando visões distintas de software: modelo de caso prático, modelo de requisitos, modelo de projeto, modelo de implementação e modelo de emprego (PRESSMAN, 2011).

Ao avançar o projeto de desenvolvimento para a construtora “More Feliz”, teremos o refinamento dos aspectos tratados na fase de concepção. Nesta fase, os objetivos atingidos serão: identificação dos riscos potenciais e definição de uma *baseline* da arquitetura. Os artefatos entregues pela equipe de desenvolvimento serão: conjunto de protótipos, modelagem de dados e modelagem do design.

c. Fase de construção

A fase de construção representa a estabilidade da arquitetura e os casos de uso mais complexos foram tratados. Deste modo, o foco das atividades das iterações será a geração de código e os testes de sistema para garantir a qualidade do que foi desenvolvido. É possível trabalhar com a automatização dos testes unitários (WAZLAWICK, 2013).

Quando o projeto de desenvolvimento do sistema do nosso estudo de caso estiver na fase de construção, a equipe atuante terá entregue o release do sistema, casos de teste e materiais de apoio ao usuário. Tais entregas resumem a qualidade do sistema e suas versões disponíveis.

d. Fase de transição

Esta é a fase final do processo, na qual o produto desenvolvido é colocado no ambiente para que os usuários possam acessá-lo.

Por fim, quando a equipe de desenvolvimento estiver na etapa de transição, os objetivos da fase estarão sendo concluídos: realização de testes com os usuários finais, versão beta do sistema, realização de treinamento e distribuição do sistema.



Questão para reflexão

Analise o seguinte cenário: ao desenvolver um projeto para a construtora “More Feliz”, a equipe atuante identificou a necessidade de ajustar o Processo Unificado para otimizar e melhorar as entregas. Na sua opinião, a adequação do processo unificado a uma necessidade específica da equipe de desenvolvimento pode ser realizada? Compartilhe sua opinião no portal do aluno.



Para saber mais

DEVMEDIA. **Introdução ao Processo Unificado**. [s.d.]. Disponível em: <<http://www.devmedia.com.br/introducao-ao-processo-unificado/3931>>. Acesso em: 13 ago. 2017.

IC. **Processo Unificado**. [s.d.]. Disponível em: <[http://www.ic.unicamp.br/~eliane/Cursos/MO409/Curso2004\(com%20MC746\)/Apresentacoes/UP.pdf](http://www.ic.unicamp.br/~eliane/Cursos/MO409/Curso2004(com%20MC746)/Apresentacoes/UP.pdf)>. Acesso em: 13 ago. 2017.

JACOBSON, I.; BOOCH, G.; RUMBAUGH, J. **The Unified Software Development Process**. Reading, MA.: Addison-Wesley, 1999.

SCOTT, K. **O Processo Unificado explicado**. Porto Alegre: Bookman, 2003.

Atividades de aprendizagem

1. Com base no processo unificado, assinale a alternativa correta:

- a) Ao finalizar a fase de concepção, teremos uma versão estável do software.
- b) A fase de elaboração possui uma arquitetura preliminar.
- c) Não é gerado um plano de desenvolvimento em nenhuma fase.
- d) A fase de construção, usualmente, possui mais ciclos que as demais.
- e) Na etapa de construção é obrigatório a “quebra” em *sprints*.

2. Ainda com base no processo unificado, analise os itens a seguir:

I- O processo unificado aborda um determinado conjunto de atividades para transformação de solicitações do usuário em um software, baseado apenas em testes de integração.

II- O processo unificado possui as seguintes características: centrado na arquitetura, orientado a casos de uso e focado em riscos.

III- O processo unificado possui as seguintes características: centrado em

uma linguagem, estrutura, arquitetura, orientado a casos de uso e focado em riscos de alto impacto.

Assinale a alternativa correta:

- a) Somente a I está correta.
- b) Somente a II está correta.
- c) Somente a III está correta.
- d) I e II estão corretas.
- e) I e III estão corretas.

3. Considerando a ordem de execução das fases do processo unificado, a alternativa que apresenta a ordem correta é:

- a) Concepção, transição, elaboração e construção.
- b) Construção, elaboração, transição e concepção.
- c) Concepção, requisitos, implementação e testes.
- d) Concepção, elaboração, construção e transição.
- e) Construção, concepção, transição e observação.

4. O processo unificado apresenta algumas características. Marque a alternativa que não representa o processo:

- a) Dirigido a casos de uso.
- b) Centrado na arquitetura.
- c) Modelo iterativo e incremental.
- d) Focado em execução de testes.
- e) Centrado na observação.

5. A fase _____ apresenta visões distintas de software: modelo de caso prático, modelo de requisitos, modelo de projeto, modelo de implementação e modelo de emprego. Marque a alternativa que preenche a lacuna de forma correta:

- a) Elaboração.
- b) Construção.
- c) Concepção.
- d) Transição.
- e) Execução.

Seção 3

Métodos ágeis

Introdução à seção

Para Pressman (2011) as metodologias ágeis foram resultados de um esforço com o objetivo de minimizar as fraquezas da engenharia de software convencional. Conforme descrito na literatura, existem doze princípios estabelecidos para a agilidade, Pressman (2011) cita-os:

- Satisfação do cliente com entregas de qualidade e contínuas.
- Acolhimento de mudanças.
- Entregas funcionais de software.
- Trabalho em conjunto entre as áreas envolvidas no projeto.
- Equipes motivadas.
- Comunicação efetiva.
- A principal medida de progresso é o software em funcionamento.
- Manutenção de um desenvolvimento sustentável.
- Atenção à excelência técnica.
- Simplicidade.
- Equipes auto-organizadas.
- Revisões do trabalho em intervalos regulares.

Para o *Agile Manifesto* (2017), estão sendo descobertas maneiras de melhorar o desenvolvimento de software. Estas novas maneiras resultam em valores:

- Indivíduos e interações mais que processos e ferramentas.
- Software em funcionamento mais que documentação abrangente.
- Colaboração com o cliente mais que negociação de contratos.
- Responder a mudanças mais que seguir um plano.

Quando falamos em engenharia de software, existem inúmeros modelos ágeis, para a presente unidade abordaremos os seguintes modelos:

- *Scrum*.
- *XP (Extreme Programming)*.

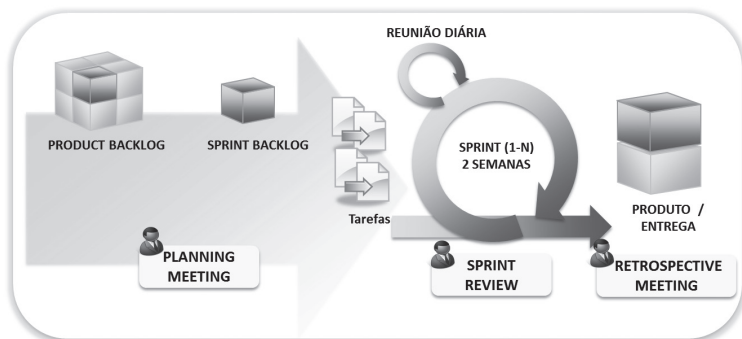
Scrum

Pham e Pham (2012) relata que o termo *Scrum* apareceu pela primeira vez em um artigo publicado por Hirotaka Takeuchi e Ikujiro Nonaka na Harvard Business Review de 1986, que descrevia uma abordagem holística, na qual equipes trabalhavam com um número reduzido de integrantes e exerciam papéis multifuncionais, similares a uma formação do Rugby, o Scrum.

Conforme descrito por Pressman (2011), os princípios do *Scrum* são consistentes com o manifesto ágil e são utilizados na orientação de atividades de desenvolvimento. O *Scrum* representa um framework que apoia a gestão de projeto e o desenvolvimento de produtos, composto por um conjunto de práticas muito utilizadas na área de desenvolvimento de software (PRIKLADNICKI; WILLI; MILANI, 2014).

O foco do *Scrum* está na utilização dessas práticas, que provaram ser eficazes para projetos que tenham características, como prazo de entregas apertados, requisitos críticos e mutáveis (PRESSMAN, 2011). Para atingir objetivos bem-sucedidos em um projeto, o *Scrum* adota uma estrutura iterativa e incremental e ao final de cada ciclo, são entregues incrementos do produto. Na imagem a seguir, mostraremos o ciclo de desenvolvimento do *Scrum*.

Figura 2.2 | Ciclo de desenvolvimento *Scrum*



Fonte: elaborada pelo autor.

Vamos analisar cada aspecto do ciclo de desenvolvimento do Scrum? Confira detalhamento.

a) Product Backlog

O ciclo de desenvolvimento inicia-se com o *Product Backlog*, coração do *Scrum*. Basicamente, representa uma lista de requisitos, necessidades do usuário ou itens que o cliente deseja. Os itens do Backlog são descritos usando terminologia do cliente e os chamamos de histórias. O responsável pelo *Product Backlog* é o *Product Owner* (PO).

Para um melhor entendimento sobre o que é uma história, pense no nosso estudo de caso da Seção 1, trata-se de uma construtora composta por inúmeros setores, incluindo a área de construção civil, certo? Então, suponhamos que você é o PO responsável desta área e criará a lista de requisitos que farão parte do *Product Backlog* que será desenvolvido. Esta lista deverá incluir itens que tragam valor ao negócio, que tenham identificação única (número autoincremental), descrição clara e objetiva para que todos que trabalhem no desenvolvimento tenham entendimento comum, grau de importância (itens com importância mais alta, serão priorizados), estimativa inicial (representação do tamanho em *stories points*), explicação de como demonstrar a história e notas (informações complementares). Ao gerar esta lista de itens, podemos ter o seguinte exemplo de *Product Backlog*:

Quadro 2.1 | Exemplo de Product Backlog

ID	Descrição	Importância	Estimativa inicial	Como demonstrar	Notas
01	Gerar custos dos recursos materiais e humanos de um determinado projeto de construção.	30	5 stories points	Ao clicar no menu de relatórios, o sistema deve mostrar a opção "Gerar relatório de custos materiais e humanos do projeto". A tela do relatório deve possibilitar a seleção do projeto XYZ, bem como o intervalo de tempo a ser considerado.	Gerar diagrama de sequência.



02	Gerar solicitação de material para um projeto.	21	8	Ao clicar no menu de Projetos, abrir a opção "Solicitação de Material". Na tela apresentação, preencher todos os campos que deverão ser obrigatórios. Ao salvar a solicitação, ela deverá ficar com o status pendente e aparecerá na fila do departamento de Projetos. O departamento de Departamento analisará a solicitação e retornará à aprovação ou à reprovação. Caso a solicitação seja reprovada, retornará ao solicitante com a justificativa preenchida. Caso seja aprovada, será enviada ao Departamento de Compras.	Gerar diagrama de sequência.
----	--	----	---	---	------------------------------

Fonte: elaborada pelo autor.

b) Sprint Backlog e a Planning Meeting

Após a definição da primeira versão do *Product Backlog* (chamamos de primeira versão, pois é uma lista dinâmica que será atualizada de acordo com a necessidade do projeto). A equipe participante do projeto realiza a *Sprint Planning Meeting*, entre os participantes estão: *PO*, *Scrum Master* e o *Scrum Team*. Também, quando necessário, outros envolvidos no projeto poderão participar.

Ao realizar a cerimônia da *Sprint Planning Meeting*, o *PO* apresenta as funcionalidades e as prioriza para a equipe, assim, os objetivos da *Sprint* serão definidos. Neste momento, todas as dúvidas pertinentes às funcionalidades priorizadas serão verificadas com o *PO*. Com o entendimento das funcionalidades, a equipe realiza a quebra em tarefas de desenvolvimento. A priorização e a quebra em tarefas originarão o *Sprint Backlog*.

Podem-se elencar os tópicos a serem discutidos na reunião na seguinte ordem de prioridade:

- Definição do objetivo da *Sprint* e data para a realização da apresentação do produto gerado na *Sprint*.
- Listagem das histórias aceitas pelos participantes da *Sprint Planning Meeting* para a *Sprint* corrente.
- Preenchimento das estimativas de cada história da *Sprint*.
- Detalhamento de como as histórias da *Sprint* podem ser demonstradas.

- Cálculo da velocidade e recursos para execução da *Sprint*.
- Definição de hora e local para realização das cerimônias das reuniões diárias.
- Quebra das histórias em tarefas.

c) Execução da Sprint (1 – N)

Após a definição dos objetivos da *Sprint*, a equipe inicia o desenvolvimento das suas tarefas. Este desenvolvimento apresenta algumas características:

- Cerimônia das reuniões diárias: representa as reuniões de 15 minutos realizadas pela equipe, tais reuniões devem ser feitas em pé em frente ao quadro de tarefas para a *Sprint* corrente. Uma abordagem comum para estas reuniões é a resposta para três perguntas:

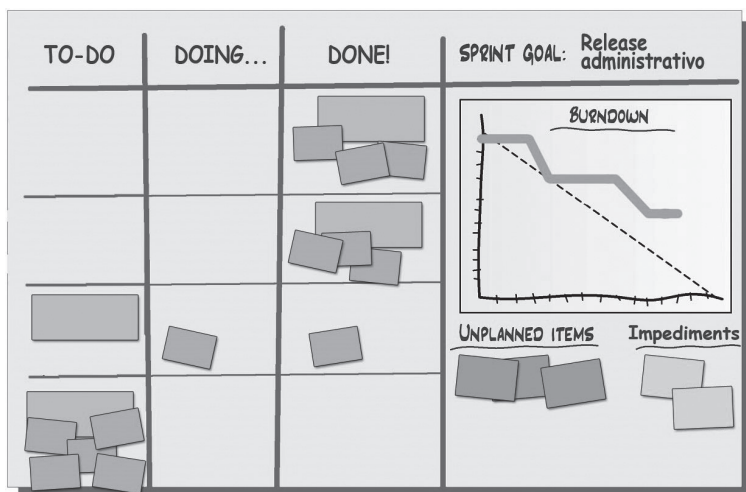
- o O que você fez?
- o O que você vai fazer?
- o Existem impedimentos que podem impactar no alcance dos objetivos da Sprint?

- Atualização do quadro de tarefas: durante a reunião diária, a equipe pode atualizar o quadro de tarefas. Este quadro pode ser eletrônico ou físico, as tarefas ou os impedimentos serão representados por post-its. Itens não planejados devem ser apontados no quadro. A discussão sobre possíveis problemas na Sprint deve ser tratada fora das reuniões diárias.

- Atualização do *burndown*: com a evolução da Sprint, a equipe deve atualizar o gráfico de evolução.

Um breve resumo da execução de uma Sprint é mostrado na figura a seguir:

Figura 2.3 | Resumo da execução de uma Sprint



Fonte: Agile Way (2009).

Explicação:

- Colunas *TO-DO* / *DOING* / *DONE*: evolução do andamento das tarefas, é como se fosse uma “fila”, para a qual todos sabem o que está em andamento e com quem, o que não está sendo executado no momento ou o que já foi finalizado.

- Coluna *SPRINT GOAL*: mostra o objetivo da Sprint, neste exemplo, é a entrega do *Release Administrativo*.

- *Burndown*: gráfico de progresso da Sprint para a entrega do *Release Administrativo*, eixo vertical representa a quantidade total de esforço para entrega da Sprint e o eixo horizontal representa o tempo restante para a entrega da Sprint.

- *Unplanned items*: itens que não foram planejados para a Sprint, porém, são necessários para entrega do *Release Administrativo*.

- *Impediments*: registro dos impedimentos que impactam na entrega da Sprint.

d) Sprint Review e Retrospective Meeting

Ao final de cada Sprint, é realizada a cerimônia *Sprint Review Meeting*. Nesta reunião, participam o PO, a equipe, *Scrum Master* e os demais envolvidos (quando necessário). O objetivo desta reunião é apresentar o resultado obtido na Sprint, demonstrando as funcionalidades desenvolvidas. Para uma reunião de *Sprint Review*,

atente-se aos seguintes pontos:

- Apresente o objetivo da Sprint. Caso existam participantes que não saibam sobre o produto a ser entregue, faça uma breve descrição sobre o assunto.
- Foque na demonstração do produto.
- A apresentação deve ser clara e objetiva.
- Quando possível, permita que os participantes da reunião façam testes no produto.
- Sobre correção/bugs, apenas cite, não entre em detalhes.

Outra cerimônia de suma importância para o Scrum é a *Sprint Retrospective*, ela acontece ao final da Sprint e tem como objetivo a verificação do que foi bom na Sprint, o que pode ser melhorado e estabelecer ações para melhorias. A diferença entre a Sprint Review e a Sprint Retrospective é que a primeira visa à adaptação no produto e, a segunda, visa à adaptação no processo de trabalho. Existem vários formatos para serem usados na *Sprint Retrospective*, mas, normalmente, podem ser feitos da seguinte forma:

- Alocação de três horas para realização da reunião.
- Contar com a participação do PO, Scrum Master e equipe.
- Evitar interrupções externas.
- Integrante da equipe é nomeado secretário para registro da reunião.
- Resumir a Sprint, eventos e decisões importantes.
- Cada integrante tem a oportunidade de apresentar a percepção sobre o que achou bom, o que acha que pode ser melhor e o que pode ser feito de diferente para a próxima Sprint.
- Comparação da velocidade estimada e realizada, identificando o motivo do desvio.
- Ao final, o Scrum Master resume as considerações apresentadas pelos integrantes da equipe sobre o que é necessário melhorar para a próxima Sprint.

XP (Extreme Programming)

Enquanto o Scrum é focado nas práticas de gestão e organização, o XP está mais relacionado à engenharia de software, levando ao extremo seu conjunto bem definido de boas práticas para que a equipe de desenvolvimento tenha condições de serem eficientes e eficazes nas mudanças que possam ocorrer no projeto. Nos tópicos descritos

a seguir, trataremos as três principais regras de funcionamento do XP: os valores, os princípios e o processo.

Valores XP

A base para XP está estabelecida e definida sob um conjunto de cinco valores: comunicação, simplicidade, feedback, coragem e respeito; cada valor torna-se um direcionador (PRESSMAN, 2011).

a. Comunicação: colaboração estreita entre os participantes do projeto, de modo que o entendimento seja comum, evitando documentações volumosas como meios de comunicação.

b. Simplicidade: projeção de necessidades imediatas, de forma simples e fácil de ser implementada.

c. Feedback: refere-se aos aspectos mais técnicos, o grau em que o software implementa o produto, procedimentos e seu comportamento durante a execução fornece o feedback, tanto para a equipe que desenvolve quanto para o cliente.

d. Coragem: refere-se à capacidade de respostas às mudanças, aprendizado com os erros, refazer códigos ruins, acreditar no feedback recebido e fornecer feedback.

e. Respeito: valor relacionado ao respeito entre equipe, cliente e, indiretamente, o software, destacando-se como reflexo dos valores anteriormente citados.

Princípios XP

Quando falamos nos princípios do XP, pode-se entender como o elo que conecta os valores às práticas: feedback rápido, simplicidade, incremento de mudanças, aceitação das mudanças e trabalho de qualidade.

Processo XP

Pressman (2011) descreve que XP emprega uma abordagem orientada a objetos como seu paradigma de desenvolvimento favorito, composta por um conjunto de regras e práticas que podem ser sintetizadas em quatro grupos de atividades metodológicas, são elas: planejamento, projeto, codificação e testes. A figura a seguir ilustra estes conjuntos de atividades e, na sequência, descreveremos cada um deles.

Figura 2.4 | Processo XP



Fonte: Pressman (2011).

a. **Planejamento:** é uma atividade conhecida também como jogo do planejamento. É iniciada com o levantamento de requisitos, possibilitando o entendimento do ambiente de negócios e permite a percepção sobre os resultados solicitados, fatores principais e funcionalidades. A partir desse levantamento, as estórias são definidas e detalhadas. São estimadas, agrupadas e distribuídas nas iterações a serem implementadas. Caso novas estórias sejam identificadas ao longo do projeto ou identificada a necessidade de alterações ou exclusões, o plano inicial é modificado.

b. **Projeto:** deve ser o mais simples possível e com soluções pontuais. Ao longo das iterações, o produto vai sendo incrementado e melhorado continuamente.

c. **Codificação:** após o desenvolvimento das estórias, a equipe de desenvolvimento trabalha na elaboração de testes de unidade que serão aplicados para cada estória a ser inclusa na versão corrente (gerada para a iteração). Após a elaboração dos testes, a equipe de desenvolvimento trabalha na implementação. No momento em que o código é gerado, XP sugere a aplicação de um conceito-chave, a programação em dupla, na qual dois integrantes da equipe devem

trabalhar na mesma estação de trabalho, a fim de resolver problemas em tempo real. Ao finalizar seu trabalho, o código desenvolvido é integrado ao trabalho dos demais.

d. **Testes:** os testes são elementos-chaves da abordagem XP e devem ser automatizados para que possam ser repetidos com frequência e com menor esforço. Como complemento, existem os testes de aceitação pelo cliente, os quais mantêm foco nas características e funcionalidades do sistema que são visíveis e podem ser revistas pelo cliente.



Questão para reflexão

O conhecido Manifesto Ágil representa a essência para os métodos ágeis, focado principalmente em desenvolvimento de software, tornando-o adaptável e aplicável.

Acesse o Manifesto Ágil que está disponível no link e o leia com atenção: Disponível em: <<http://www.manifestoagil.com.br/principios.html>>. Acesso em: 29 set. 2017. Após a leitura, analise o cenário: Carlos é gerente de projetos em uma grande empresa de desenvolvimento de software. Ele recebeu um novo projeto e pretende aplicar o Scrum. Em uma das reuniões com o cliente, Carlos disse que não poderá aceitar mudanças no projeto. Esta visão está de acordo com os princípios ágeis? Reflita sobre esta questão e compartilhe com seu professor.



Para saber mais

MANIFESTO ÁGIL. **Manifesto for Agile Software Development.** [s.d.]. Disponível em: <<http://agilemanifesto.org>>. Acesso em: 29 jul. 2017.

AGILE ALLIANCE BRAZIL. **Agile Alliance.** [s.d.]. Disponível em:

<<https://www.agilealliance.org/agilealliancebrazil/>>. Acesso em: 29 jul. 2017.

WARDEN, S.; SHORE, J. **The Art of Agile Development.** 1. ed. Sebastopol: O'Reilly & Assoc, 2007.

SMITH, G.; SIDKY, A. **Becoming Agile in an imperfect world.** Greenwich: Manning, 2009.

Atividades de aprendizagem

1. Analisando os valores do Manifesto Ágil, qual das opções a seguir não os reflete?

- a) Ferramentas e processos são mais importantes que indivíduos e interações.
- b) Software em funcionamento mais que documentação abrangente.
- c) Colaboração com o cliente mais que negociação de contratos.
- d) Responder a mudanças mais que seguir um plano.
- e) Indivíduos e interações mais que processos e ferramentas.

2. Nas metodologias ágeis (XP e Scrum), o produto é desenvolvido em partes, estas partes são trabalhadas em ciclos, variando entre uma a quatro semanas. Estes ciclos podem ser denominados:

- a) Sprint Planning Meeting.
- b) Reuniões diárias.
- c) Sprint Review.
- d) Iterações ou Sprints.
- e) Daily review meeting.

3. Abordagem orientada a objetos como seu paradigma de desenvolvimento favorito, composta por um conjunto de regras e práticas que podem ser sintetizadas em quatro grupos de atividades metodológicas, são elas: planejamento, projeto, codificação e testes. Esta descrição refere-se ao(à):

- a) Scrum.
- b) XP.
- c) Product Owner.
- d) Codificação.
- e) Processo Unificado.

4. A cerimônia do Scrum que tem como objetivo identificar as melhorias a serem aplicadas na Sprint seguinte é:

- a) Reunião diária.
- b) Sprint Planning Meeting.
- c) Sprint Retrospective.
- d) Backlog do Produto.
- e) Daily review meeting.

5. Analise as afirmativas a seguir:

I - XP possui um conjunto de cinco valores: comunicação, simplicidade, feedback, coragem e respeito.

II - Sprint Review tem como objetivo revisar o produto gerado na Sprint corrente.

III - O Product Owner não deve participar das cerimônias do Scrum.

Verifique qual alternativa está correta:

- a) I e III estão corretas.
- b) II e III estão corretas.
- c) Somente a III está correta.
- d) I e II estão corretas.
- e) Somente a II está correta.

Seção 4

Unified Modeling Language (UML)

Introdução à seção

Pressman (2011) relata que Grady Booch, Jim Rumbaugh e Ivar Jacobson foram responsáveis pelo desenvolvimento da UML na década de 1990, combinando um grupo de notações de modelagem concorrentes utilizadas pela indústria de software na época. Em 1997, a UML (1.0) foi apresentada ao OMG (*Object Management Group*). Posteriormente, a UML 1.0 sofreu revisões e tornou-se a UML 1.1. O atual padrão é a UML 2.0.

De acordo com Conallen (2003), a notação utilizada pela UML é padronizada pela OMG e facilita a compreensão de cada parte do sistema que está sendo modelado, por qualquer pessoa que tenha conhecimento sobre a linguagem. Portanto, é linguagem de modelagem indicada para especificar, visualizar, construir e documentar um sistema.

Dentre as suas características, podemos citar:

- Utilizar atores e casos de uso para mostrar estruturas e fronteiras de um sistema e suas principais funções e funcionalidades.
- Utilizar diagrama de classes, no qual demonstra atributos, operações e relacionamentos, para representar a estrutura estática.
- Representar a estrutura dinâmica com os diagramas de estado, sequência, colaboração e atividades.
- Revelar a arquitetura de implementação física (hardware ou pacote software) com os diagramas de componentes e implementação.
- Estender sua função por meio de estereótipos.

Com todas estas características, é possível a representação de partes do sistema com os vários diagramas, e a união destes representa o sistema como um todo. Os diagramas são recursos gráficos para a visualização de um sistema sob diferentes perspectivas e geralmente por itens e relacionamentos. Esses diagramas podem ser divididos em três categorias (MACORATTI, 2017):

- Diagramas estruturais: representação da construção de blocos de recursos de sistemas que não mudam com o tempo.
- Diagramas comportamentais: representação de como o processo responde às alterações ou como evolui ao longo do tempo.

Nos próximos tópicos, detalharemos os diagramas que compõem a UML e focaremos nas técnicas de modelagens comportamentais e estruturais.

Técnicas de modelagem comportamental

A categoria dos diagramas comportamentais tem como objetivo mostrar como o processo responde às alterações ou como é a sua evolução ao longo do tempo, representando uma modelagem dinâmica do sistema. Dentre os diagramas que a compõe estão: diagrama de atividades, sequência, casos de uso e estados.

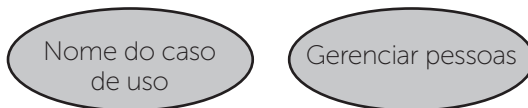
a. Diagrama de caso de uso

Os diagramas de caso de uso são responsáveis pela modelagem comportamental de um sistema, subsistema ou de uma classe, apresentando uma visão externa de como esses elementos podem ser utilizados no contexto. A seguir estão os três tipos de símbolos gráficos padronizados que compõem um caso de uso.

- **Caso de uso**

Um caso de uso captura e expressa a interação entre usuário do sistema (ator) e o próprio sistema. É escrito com o objetivo de mostrar o que o sistema deve fazer. A representação gráfica de um caso de uso é uma elipse com um nome que possa refletir o seu objetivo ou a sua finalidade.

Exemplos:



- **Ator**

O ator é um termo utilizado para representar um usuário do sistema ou outro sistema que se comunicará com o sistema em desenvolvimento. O modo gráfico de representação de um ator é uma figura com traços simples de uma pessoa, essa figura é acompanhada pelo nome que identifica o papel desempenhado no sistema.

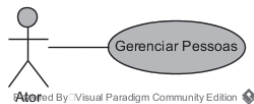
Exemplos:



- **Relacionamento**

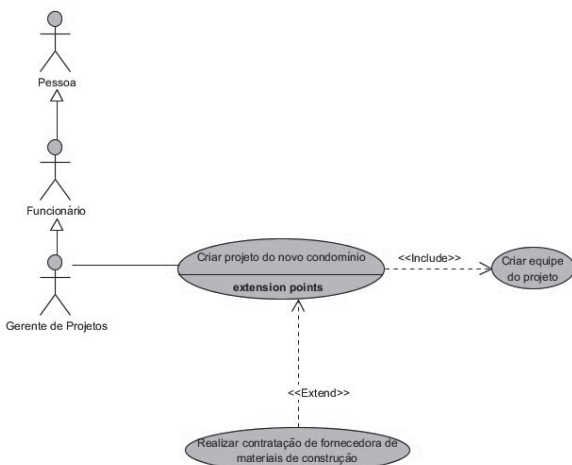
O relacionamento tem por objetivo expressar que um ator pode interagir com determinado caso de uso. No diagrama de caso de uso, podemos encontrar dois tipos principais de relacionamento: `<<extend>>` e `<<include>>`.

Exemplos:



Lembra do estudo de caso da Seção 1? Vamos colocá-lo em prática. Confira a seguir um exemplo de caso de uso completo.

Figura 2.5 | Exemplo de caso de uso - criar projeto do novo condomínio



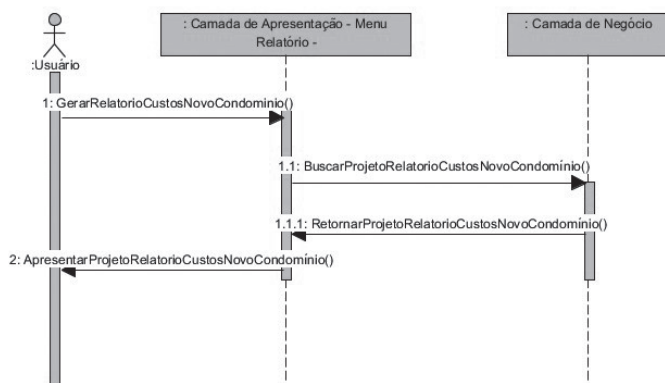
Fonte: elaborada pelo autor.

Observe nesta imagem que temos um ator que pode desempenhar vários papéis, o papel compartilhado em um caso de uso é mostrado pelo relacionamento de generalização. Neste exemplo, temos um gerente de projetos da equipe da construtora que precisa iniciar um novo projeto. Para a criação do projeto, é necessário que uma equipe seja alocada para executar as atividades, representada pelo relacionamento <<include>>. Também, se necessário, o gerente de projeto poderá realizar a contratação de uma determinada fornecedora de materiais e insumos necessários para a construção do novo condomínio.

b. Diagrama de sequência

Um diagrama de sequência descreve os aspectos dinâmicos de um sistema, mostrando o comportamento e a ordem temporal de seus objetos, e as comunicações necessárias entre objetos para a realização dos processos. Os diagramas de sequência são nomeados desta forma porque mostram ao longo de uma linha de tempo a sequência de comunicações entre objetos (MACORATTI, 2017). Geralmente, para cada caso de uso constrói-se um diagrama de sequência principal, apresentando as sequências habituais de comunicação entre objetos e diagramas complementares, descrevendo sequências alternativas e tratamento de erros. Ao falar de diagrama de sequência, utilizamos o termo cenário. Cenário pode ser entendido como uma forma de ocorrência de um caso de uso. Na figura a seguir, vemos um simples diagrama de sequência.

Figura 2.6 | Exemplo de diagrama de sequência



Fonte: elaborada pelo autor.

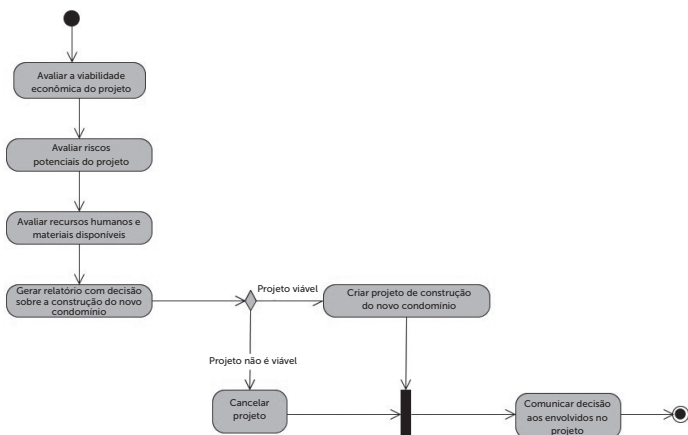
Neste diagrama de sequência, exemplificamos uma operação realizada por um determinado usuário. Este usuário faz a requisição de um relatório de custos do projeto para o novo condomínio e realiza uma interação com a camada de apresentação do sistema. Ao clicar em gerar novo relatório, a camada de negócio é chamada e retorna os resultados do relatório solicitado.

c. Diagrama de atividades

O diagrama de atividades modela o fluxo sequencial das atividades geradas por uma operação ou por um agente externo ao sistema. Este diagrama é modelado como um fluxograma. Desta forma, mostra um processo de negócios ou um processo de software como um fluxo de trabalho por meio de uma série de ações. Pessoas, computadores ou componentes de software podem executar essas ações (MSDN, 2017).

No exemplo de diagrama de atividades a seguir, demonstraremos uma análise que é realizada antes do início dos projetos na construtora. Sempre que existe um potencial projeto para a empresa, o gerente de projetos realiza uma série de análises, a fim de verificar se o projeto atende aos requisitos mínimos para ser trabalhado ou precisará ser cancelado. Nesta análise, são considerados aspectos técnicos e econômicos, viabilidade de execução, fatores de riscos e disponibilidade de recursos. Após realizar esta análise, o gerente de projetos gera um relatório com o parecer sobre o projeto. Com este relatório, temos a decisão se o projeto continuará ou será cancelado. Em ambas as decisões, o resultado deve ser compartilhado com os envolvidos.

Figura 2.7 | Exemplo de diagrama de atividades



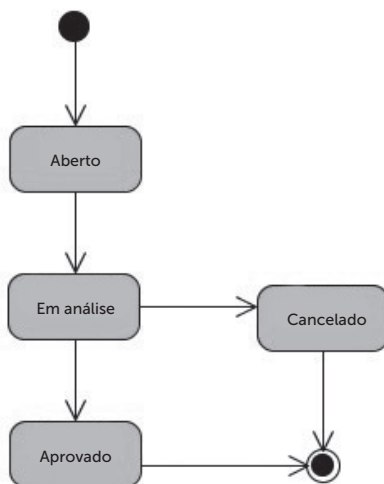
Fonte: elaborada pelo autor.

d. Diagrama de estados

O comportamento de uma classe de objetos é representado através de um diagrama de transição de estado, que apresenta o ciclo de vida de uma classe, os eventos que causam a transição de estado para o outro e as ações resultantes da mudança de estado.

A imagem a seguir representa o diagrama de estados para a pré-análise feita para os projetos de novos condomínios.

Figura 2.8 | Exemplo de diagrama de estados



Fonte: elaborada pelo autor.

Técnicas de modelagem estrutural

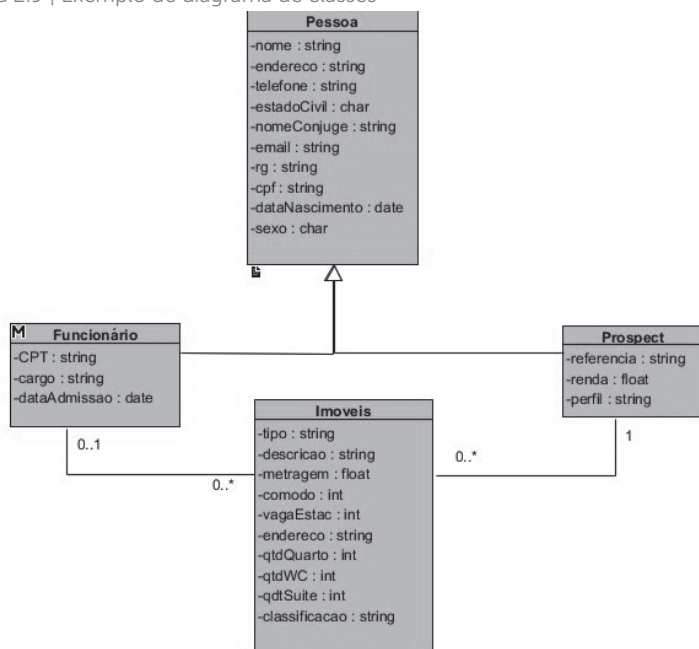
A categoria dos diagramas estruturais representa a construção de blocos de recursos de sistemas que não mudam com o tempo, representando a estrutura estática do sistema. Nesta categoria estão: diagrama de classes, diagrama de objetos, estrutura composta, implantação, componentes e pacotes.

a. Diagrama de classes

O diagrama de classes tem como objetivo modelar a visão estática do sistema, mostrando um conjunto de classes, interfaces, colaborações e os seus relacionamentos (CONALLEN, 2003).

A representação de um exemplo de diagrama de classes é mostrada na imagem a seguir.

Figura 2.9 | Exemplo de diagrama de classes



Fonte: elaborada pelo autor.

b. Diagrama de objetos

O diagrama de objetos pode ser visto como uma variação do diagrama de classes, pois representará as instâncias e as ligações entre instâncias de uma classe. Na sua estrutura, descreve um determinado conjunto de objetos e como eles se relacionam entre si.

c. Diagrama de estrutura composta

O diagrama de estrutura composta mostra a estrutura interna dos classificadores estruturados usando peças, portas e conectores. Um classificador estruturado define a implementação de um classificador e pode incluir uma classe, um componente ou um nó de implementação (IBM, 2017).

d. Diagrama de componente

Este diagrama representa a modelagem de software baseada em componentes, indicando cada componente e seu relacionamento.

Mostra cada artefato do qual o componente é feito, tais como código, fonte, biblioteca de programação, organização de tabelas de bancos de dados ou interfaces.

e. Diagrama de implantação

O diagrama de implantação aborda a configuração e os aspectos de arquitetura relacionados a um determinado sistema, mostrando a ligação entre seus componentes, tanto na visão de hardware quando de software.

f. Diagrama de pacotes

Pode-se 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).



Questão para reflexão

Após o estudo de todas as seções desta unidade, você considera possível executar as atividades de um projeto de desenvolvimento de software usando todas as metodologias apresentadas, de forma que uma seja complemento da outra e não haja sobrecarga de esforço? Compartilhe sua análise no portal do aluno.



Para saber mais

IBM. **An introduction to the Unified Modeling Language**. [s.d.]. Disponível em: <<https://www.ibm.com/developerworks/rational/library/769.html>>. Acesso em: 13 ago. 2017.

UML. **UML**. Disponível em: <<http://www.uml.org/>>. [s.d.]. Acesso em: 13 ago. 2017.

ABPMP. **BPM CBOK**. 1. ed. Brasil: Association of Business Process Management, 2013.

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

Atividades de aprendizagem

1. Analise as descrições a seguir:

- I- Descrevem grupos de objetos que colaboram através de comunicação.
- II- Tem como objetivo modelar a visão estática do sistema, mostrando um conjunto de classes, interfaces, colaborações e os seus relacionamentos.
- III- São responsáveis pela modelagem comportamental de um sistema, subsistema ou de uma classe, apresentando uma visão externa de como esses elementos podem ser utilizados no contexto.

É possível afirmar que as descrições acima estão relacionadas, respectivamente, ao:

- a) Diagrama de colaboração, diagrama de classe e diagrama de caso de uso.
- b) Diagrama de caso de uso, diagrama de objetos e diagrama de estado.
- c) Diagrama de sequência, diagrama de caso de uso e diagrama de classe.
- d) Diagrama de caso de uso, diagrama de estado e diagrama de sequência.
- e) Diagrama de colaboração, diagrama de interação e diagrama de classe

2. O diagrama de _____ possui alguns conceitos aplicados durante a modelagem de sistemas, alguns destes conceitos são: representação de estereótipos, definição de escopos, relacionamentos de multiplicidade. Qual das opções a seguir preencherá corretamente a lacuna?

- a) Caso de uso.
- b) Sequência.
- c) Classes.
- d) Pacotes.
- e) Estado.

3. Dentre os diagramas listados nas alternativas a seguir, qual deles NÃO pertence à categoria de diagramas comportamentais:

- a) Diagrama de caso de uso.
- b) Diagrama de sequência.
- c) Diagrama de classes.
- d) Diagrama de atividades.
- e) Diagrama de pacote.

4. Analise as afirmativas a seguir:

I- Um caso de uso captura e expressa a interação entre usuário do sistema (ator) e o próprio sistema.

II- O ator é um termo utilizado para representar um usuário do sistema ou outro sistema que se comunicará com o sistema em desenvolvimento.

III- Os casos de uso devem ser modelados apenas na etapa de iniciação de um projeto.

Qual opção apresenta uma visão incorreta sobre os elementos de um caso de uso:

- a) A I está incorreta.
- b) A I e a II estão incorretas.
- c) A II está incorreta.
- d) As opções I e II apresentam visão correta sobre os elementos de um caso de uso.
- e) As opções I, II e III apresentam visão correta sobre os elementos de um caso de uso.

5. O diagrama UML que tem como objetivo a representação do comportamento interno de um objeto, sistema ou subsistema é o:

- a) Diagrama de classe.
- b) Diagrama de caso de uso.
- c) Diagrama de colaboração.
- d) Diagrama de objetos.
- e) Nenhuma das alternativas anteriores.

Fique ligado

A Seção 1 apresentou a introdução sobre os conceitos e os modelos de processo, e descreveu um estudo de caso. A partir de agora, entraremos nos detalhes sobre processo unificado, UML e metodologias ágeis.

Como foi visto na Seção 2, O PU está estruturado sob um conjunto de técnicas e procedimento que são necessários para a transformação dos requisitos solicitados pelo cliente em um produto final, o software. É um modelo bem estruturado e definido, cuja modelagem permite uma visão clara do que precisa ser feito, bem como define as

responsabilidades de cada papel.

Quando foi tratado, na Seção 3, o tema desenvolvimento de software, vemos que a literatura apresenta uma enorme quantidade de metodologias que são apresentadas como boas práticas. No entanto, devemos ter a ciência de que cada projeto apresenta características peculiares e analisá-las é um exercício que deve ser feito pela equipe que atua com desenvolvimento de sistemas. Ao utilizar uma metodologia ágil, tanto a equipe que atenderá ao projeto quanto o cliente devem estar alinhados e unidos em prol de cada evento proposto, a fim de que no final, todos possam colher os frutos de um produto correto e com qualidade. Dentre as metodologias ágeis disponíveis estão o Scrum e o XP, que na prática são complementares. Visto que o primeiro trata aspectos relacionados à gestão e o XP foca nas características mais técnicas.

Na última seção, foram apresentados os conceitos relacionados à UML, sua evolução ao longo do tempo e seus diagramas.

- Características: conjunto de método para diagrama e notações, descrição de relacionamentos laterais e “pai-filho”, o bloco de símbolo varia conforme o tipo de diagrama e é utilizado para escrever sistemas.
- Objetivos da linguagem: elaboração de casos de uso, elaboração de requisitos de sistemas e os fluxos de processo podem ser representados em um nível mais detalhado.
- Vantagens: linguagem difundida e bem estabelecida, utilizada por inúmeras organizações e existem muitas referências bibliográficas disponíveis.
- Desvantagens: desenhado para modelagem de aplicações. Modelagem de processo de negócio é uma utilização secundária e a notação pode variar de ferramenta para ferramenta.

Para concluir o estudo da unidade

Caro aluno, chegamos ao fim desta unidade, esperamos que ao longo de cada seção você tenha alcançado sua meta de aprendizagem. Cada seção focou em um tópico específico: na Seção 2, vimos o processo unificado, o conjunto de atividades e seu detalhamento; na Seção 3 foram destacadas as características da metodologia ágil; na Seção 4, abordamos aspectos relacionados à UML.

Para complementar sua aprendizagem, disponibilizamos e recomendamos links, vídeos e livros ao longo das seções, veja em “Para saber mais” e consulte nossas “Referências bibliográficas”.

Atividades de aprendizagem da unidade

1. Quando estamos falando do processo unificado, é possível afirmar que:

- a) Ao finalizar a fase de concepção, o projeto é concluído e a equipe é realocada em outro projeto.
- b) Na fase de elaboração você deve elaborar diagrama de classes.
- c) Não existem ciclos de desenvolvimento.
- d) Na fase de construção podem ser realizados testes unitários.
- e) A transição é a fase final do processo onde os requisitos serão levantados com os envolvidos.

2. Considerando as fases do processo unificado, a alternativa que não relaciona uma das fases é:

- a) Transição.
- b) Elaboração.
- c) Testes unitários.
- d) Concepção.
- e) Construção.

3. _____ apresenta as funcionalidades e as prioriza para a equipe, assim, os objetivos da Sprint serão definidos. Neste momento, todas as dúvidas pertinentes às funcionalidades são priorizadas. Assinale a alternativa que mostra o papel responsável por esta tarefa:

- a) Scrum master.
- b) Equipe de desenvolvimento.
- c) PO.
- d) Diretor executivo.
- e) Gerente de Desenvolvimento.

4. Uma abordagem comum para estas reuniões é a resposta para três perguntas:

- O que você fez?
- O que você vai fazer?
- Existem impedimentos que podem impactar no alcance dos objetivos da Sprint?

Esta abordagem é usada em qual cerimônia do Scrum?

- a) Reuniões diárias.
- b) Reuniões de retrospectiva.
- c) Reuniões de planejamento.
- d) Reuniões de revisão.
- e) Reuniões de retrospectiva de escopo.

5. Analise as afirmações sobre os elementos que compõem o diagrama de casos de uso:

I - O ator é um termo utilizado para representar um usuário do sistema ou outro sistema que se comunicará com o sistema em desenvolvimento.

II - Um caso de uso captura e expressa a interação entre usuário do sistema (ator) e o próprio sistema.

III- O relacionamento tem por objetivo expressar que um ator pode interagir com determinado caso de uso.

É possível afirmar que:

- a) Apenas a I está correta.
- b) Apenas a II está correta.
- c) II e III estão corretas.
- d) I e III estão corretas.
- e) Todas as afirmações estão corretas.

Referências

AGILE MANIFESTO. **Manifesto for Agile Software Development**. [s.d.]. Disponível em: <<http://agilemanifesto.org>>. Acesso em: 29 jul. 2017.

AGILE WAY. **Gráfico Burndown** - Sugestão de uso. 2009. Disponível em: <<http://agileway.com.br/wp-content/uploads/2009/08/taskboard-burndown.jpg>>. Acesso em: 13 ago. 2017.

CONALLEN, J. **Desenvolvendo aplicações Web com UML**. 2. ed. Rio de Janeiro: Campus, 2003.

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

IBM. **Diagramas de estrutura composta**. 2017. Disponível em: <https://www.ibm.com/support/knowledgecenter/pt-br/SS5JSH_8.5.0/com.ibm.xttools.modeler.doc/topics/ccompstruc.html>. Acesso em: 13 ago. 2017.

MACORATTI. **UML: conceitos básicos**. 2017. Disponível em: <http://www.macoratti.net/vb_uml2.htm>. Acesso em: 5 ago. 2017.

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

MSDN. **Diagramas de atividade UML: referência**. 2017. Disponível em: <<http://msdn.microsoft.com/pt-br/library/dd409360.aspx>>. Acesso em: 13 ago. 2017.

PHAM, A.; PHAM, P. **Scrum em ação**. São Paulo: Novatec, 2012.

PRESSMAN, R. **Engenharia de software**. 7. ed. Porto Alegre: AMGH, 2011.

PRIKLADNICKI, R.; WILLI, R.; MILANI, F. **Métodos ágeis para o desenvolvimento de software**. Porto Alegre: Bookman, 2014.

SOFTTEX. **Guia geral MPS de software**. 2017. Disponível em: <http://www.softex.br/wp-content/uploads/2013/07/MPS.BR_Guia_Geral_Software_2012-c-ISBN-1.pdf>. Acesso em: 13 ago. 2017.

SOMMERVILLE, I. **Engenharia de software**. São Paulo: Pearson Prentice Hall, 2011.

SOUZA, L. M. de. Método ágil XP (Extreme programming). **Revista Eletrônica FIA**, on-line, São Paulo, v. 3, n. 3, jul-dez, 2007. Disponível em: <http://intranet.fainam.edu.br/aceso_site/fia/academos/revista3/6.pdf>. Acesso em: 13 ago. 2017. ISSN 1809-3604.

WAZLAWICK, R. S. **Engenharia de software: conceitos e práticas**. 1. ed. Rio de Janeiro: Campus, 2013.

Modelagem de caso de uso

Marco Ikuro Hisatomi

Objetivos de aprendizagem

Você sabe por que devemos modelar um sistema? Como seria programar um sistema sem o levantamento dos dados? Um bom sistema é aquele que auxilia a gestão do negócio do usuário, mas como desenvolver um bom sistema sem a parte de modelagem?

Caro aluno, você será levado a estudar e a conhecer conceitos importantes do caso de uso e do diagrama de caso de uso, previsto na UML.

A análise de sistemas é uma atividade onde se realizam estudos de métodos com a finalidade de encontrar o melhor caminho para que a informação possa ser processada, fazendo a informação se transformar em algo de valor para a organização/instituição, auxiliando nas tomadas de decisão. Para auxiliar o analista de sistemas nesta fase, existem algumas metodologias e diagramas.

Lembrando que a UML é uma linguagem de modelagem unificada, onde se definem dispositivos que nos ajudam na missão de projetar e documentar sistemas orientados a objetos. Os modelos da UML estão compostos em modelagem estrutural com os diagramas de classe, pacote e objeto; modelagem comportamental com os diagramas de caso de uso, sequência, comunicação, atividades, estados e modelagem arquitetural com os diagramas de componentes e implementação.

Seção 1 | Introdução ao caso de uso

Na Seção 1, ao conhecer as regras de negócio e as funcionalidades a que que o sistema deverá atender, uma ou várias histórias podem ser descritas para

demonstrar, em um descritivo, o que ficou entendido. Esta seção apresentará como escrever estas histórias em um formato de caso de uso, sendo um grande aliado para melhorar a comunicação entre os membros da equipe de desenvolvimento de sistema.

Seção 2 | Diagrama de caso de uso

Dando continuidade na técnica de documentação do levantamento do sistema, na Seção 2, teremos o diagrama de caso de uso. Para o melhor entendimento das interações que existem entre os atores e as funcionalidades, elas serão nomeadas (para facilitar a rastreabilidade), inclusive possibilitando uma leitura padronizada e, para as dependências entre uma funcionalidade e outra, faz-se uso do diagrama de caso de uso. Esta seção descreverá os principais componentes e os padrões utilizados nesta linguagem para documentação de caso de uso.

Introdução à unidade

Caro aluno, iniciaremos nosso aprendizado de caso de uso da UML, o que complementará o seu conhecimento enquanto analista de sistemas no processo de desenvolvimento de software.

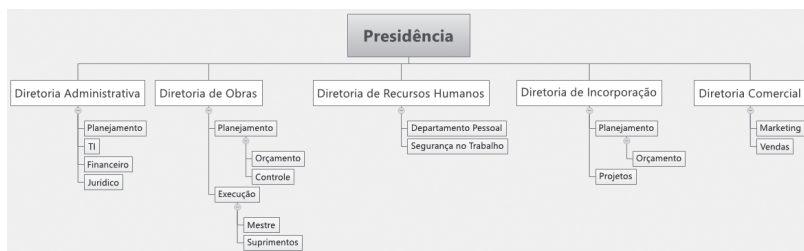
A UML é uma linguagem de modelagem unificada, usada para visualização, especificação, construção e documentação de sistemas complexos. Em 1994, quando Rumbaugh e Booch fizeram a unificação dos métodos Booch e OMT, surgiu a UML.

O caso de uso é uma técnica usada para melhorar a compreensão dos requisitos, ele é composto por descrições narrativas dos processos e o detalhamento das funcionalidades, requisitos, regras de negócio do sistema a ser desenvolvido.

O diagrama de caso de uso representa a funcionalidade proposta para um software que será desenvolvido, com as interações com os seus atores; é uma ferramenta para a documentação do levantamento dos requisitos funcionais do sistema.

Nesta unidade, abordaremos sobre o caso de uso e o diagrama de caso de uso, veremos que eles descrevem as funcionalidades essenciais do sistema, identificando as interações com os usuários. Para auxiliar o aprendizado, utilizaremos o estudo de caso da construtora “More Feliz”, ou seja, daremos continuidade ao estudo de caso utilizado nas unidades anteriores, conforme apresentado no organograma, na Figura 3.1:

Figura 3.1 | Organograma da More Feliz



Fonte: elaborada pelo autor.

Seção 1

Introdução ao caso de uso

Introdução à seção

Caro aluno, esta seção abordará o caso de uso como uma forma simples e direta para documentar as funcionalidades a serem desenvolvidas em um sistema. Veremos que ele é definido por suas funcionalidades, regras de negócio e visão do usuário.

Como já mencionado, o caso de uso é formado por histórias do assunto (objeto de estudo e da proposta) do desenvolvimento do sistema, para melhor entendimento da parte que estão levantando e, para ser validado, com a outra parte (usuários dos sistemas).

Assim, entende-se que a partir destas histórias, considera-se que os requisitos (entre funcionalidades e regras de negócio) ficam completos e compreendidos para dar início ao projeto do futuro sistema a ser desenvolvido. Essa é uma ótima ferramenta usada durante o levantamento de requisitos funcionais do sistema, identificando os atores e os processos pelos quais eles são os responsáveis.

Tenha uma ótima leitura!

1.1. Caso de uso

É um documento narrativo, em que é descrita uma sequência de eventos entre um ator e um processo do sistema. Veremos no decorrer desta seção quem pode ser o ator e como identificar um caso de uso. Podemos repetir o que já foi dito antes, que casos de uso são histórias ou casos de utilização do sistema, que demonstram através da narrativa estruturada de tal forma que todos os membros envolvidos no desenvolvimento possam entender as aplicabilidades das funções que o sistema deverá atender.

A descrição do caso de uso pode ser informal, típica ou detalhada. Cada tipo de descrição empregada requer níveis diferentes de entendimento e, assim, maior será a precisão do entendimento do caso de uso:

- **Informal:** possui o nome do caso de uso e uma descrição de sua funcionalidade.

- Típica: identifica o ator que inicia o caso de uso, apresenta a descrição do fluxo normal e alternativo.
- Detalhada: identifica o ator que inicia o caso de uso, informa o objetivo, as condições de disparo (*triggers*), apresenta a descrição do fluxo normal e alternativo.

Segue um exemplo de caso de uso, de como procede o registro para o pagamento de uma compra de material escolar.

Quadro 3.1 | Exemplo de caso de uso

Caso de uso	Atores	Descrição
Comprar material	<ul style="list-style-type: none"> • Cliente • Fornecedor 	O cliente chega ao caixa com os itens a pagar e o fornecedor registra os itens para pagamento.

Fonte: elaborado pelo autor.

Na sequência, o desenvolvimento do caso de uso, a partir de um processo básico (podendo ser considerado o processo primário), em que será aprofundado na continuidade do levantamento de requisitos do sistema, poderíamos, por exemplo, criar o caso de uso “Tipo Pagamento”, no qual estabelece o valor do item comprado mediante cada tipo de pagamento e, assim, ter sucessivo aprofundamento das regras de negócios, para atender às funcionalidades da organização. Dessa forma, é necessário o grau de necessidades a que o sistema atenderá para ter o sucesso no desenvolvimento do software.

Vale lembrar que a UML não especifica um formato para a descrição do caso de uso, mas é adequado fazer a narração de forma que fique clara, e que não gere interpretações ambíguas, errôneas ou incompletas.

Outro fator importante: um caso de uso representa a interação entre um ator e um sistema, ou seja, é uma parte de um todo em um processo de desenvolvimento de sistemas, como “cadastrar imóvel” ou “criar solicitação”. Também, não é necessária uma ordem nas sequências de ações, das decisões ou das estruturas de objetos, pois eles constituem os elementos que estruturam todas as etapas do processo de software.

Um caso de uso é isolado? Vejamos, cada caso de uso terá uma descrição de sua funcionalidade dentro do sistema a ser desenvolvido;

porém, esse pode 'usar' ou 'estender' a funcionalidade de outro caso de uso. Ao longo do refinamento que se faz necessário para o detalhamento das regras de negócio ou do procedimento do usuário final, cada caso de uso pode se estender através de outro caso de uso. Portanto, um pode ter continuidade por caso de uso, principalmente quando se utiliza as condições de gatilho.

Temos alguns tipos de caso de uso, como: primário, secundário e opcional. Um caso de uso primário representa os objetivos do autor, ele reflete os processos principais que estão sendo automatizados, geralmente acionados pelo ator. Já o caso de uso secundário seria aquele que não traz benefício direto ao ator, mas é necessário para que o sistema funcione de forma satisfatória, descreve os processos menos importantes. Os casos de uso opcionais são aqueles que podem ser incluídos em versões futuras do software.

Percebe-se que os casos de uso podem ser descritos com vários níveis de detalhamento e comprometimento no projeto, o mesmo caso de uso pode ser escrito com formato e detalhamento diferentes. Para descrever um caso de uso, temos o formato de alto nível e o formato expandido. Cabe ao analista de sistemas identificar e priorizar os casos de uso que representam os processos de negócio da empresa.

Questão para reflexão

Pense em algo do nosso dia a dia e gere a descrição do caso de uso para ela. Imagine criar um caso de uso "compra no mercado". Como seria?

Quando vamos ao mercado, geralmente deixamos o carro no estacionamento, escolhemos os produtos, podemos interagir com o pessoal da padaria e/ou açougue, depois vamos ao caixa efetuar o pagamento. Então, vamos completar o quadro a seguir. Lembre-se de que temos que ver os assuntos principais:

Caso de uso	Atores	Descrição

Temos a certeza de que foi fácil construir esse caso de uso: compra no mercado.

Agora, praticaremos com o nosso estudo de caso da construtora “More Feliz”. Como vimos, a “More Feliz” é uma construtora e seu sistema abrangerá diversos departamentos.

Verificaremos alguns casos de uso que poderemos ter:

Quadro 3.2 | Estudo de caso da construtora More Feliz

Caso de uso	Ator(es)	Descrição
Manter imóveis	Assistente Adm.	Os imóveis para venda têm seu cadastro atualizado pelos assistentes administrativos, eles informam todos os detalhes, como tipo (casa, apartamento, comercial), metragem, quantidade de cômodos, quantidade de vagas na garagem (no caso de o imóvel ser em edifício).
Solicitar material	Engenheiro Arquiteto	Antes de iniciar a construção da obra, o engenheiro encaminha a solicitação do material a ser usado para o departamento de orçamento.
Manter estoque	Assistente Adm.	A empresa trabalha com grandes construções, e cada uma tem seu estoque local de material, e fica um assistente com a missão de dar “baixa” ou “acrescentar” o material. Ele também pode acionar outras construções para solicitar material de urgência.
Mostrar imóveis	Consultor de vendas	A construtora possui um centro para a venda de seus imóveis, com consultores treinados, que atendem o cliente, mostram o portfólio e informam sobre as opções de pagamento.
Manter vendas	Consultor de vendas	O cliente tem várias opções de pagamento para a aquisição do imóvel, e na venda, o consultor finaliza os documentos e faz o acompanhamento até a quitação do imóvel. Ao finalizar a venda, o sistema já atualiza o imóvel, colocando que não está mais disponível para venda.

Manter funcionário	Rh	O departamento de Rh cuida da contratação de todos os funcionários da empresa. Faz o controle de férias, horas extras e verifica a necessidade de treinamentos.
Controlar orçamento	Departamento de orçamento	Todo o material solicitado pelo engenheiro, o departamento de orçamento realiza a cotação de preços, finaliza a compra e repassa para as obras.
Realizar MKT	Departamento de marketing	O departamento de marketing faz a propaganda dos novos lançamentos da construtora, eles devem realizar pesquisas de público-alvo, elaborar estratégias de marketing e fazer a cotação do custo destas estratégias.

Fonte: elaborado pelo autor.

Aqui, você percebe que foram informados alguns casos de uso da construtora. Volte ao texto inicial, veja o organograma para entender a extensão de aprofundamento da análise e do projeto a ser construído. Agora é a sua vez! Investigue mais casos de uso para o nosso estudo de caso. É recomendado que se faça a análise de cada departamento, veja como é o processo de cada um e como os departamentos se relacionam.

Para saber mais

Um software é usado para cadastrar dados e informações, mas seu principal objetivo é elaborar algo que agregue valor para a organização no qual ele será implementado.

1.2. Identificando um caso de uso

Como observamos, um caso de uso é uma descrição de um processo, geralmente com uma certa complexidade e que possui várias partes. Um erro que pode ocorrer, é colocar uma operação como sendo um caso de uso, por exemplo, empregando a descrição da “compra no mercado”, usando o item “imprimir nota fiscal”; este não seria um caso de uso e sim uma operação no caso de uso “comprar itens”. A “compra no mercado” possui interações mais amplas, sem entrar no detalhe, no nível de operações.

Então, como identificar os casos de uso do sistema? Isso envolve

uma percepção dos processos e de seus detalhamentos. Dentre várias alternativas, podemos citar duas largamente utilizadas: o *brainstorming* (no quadro “Para saber mais”) e a revisão de documentos (manuais e instruções normativas da organização) que especificam os processos e os procedimentos da organização.

Podemos realizar o levantamento junto às pessoas autorizadas dos casos de usos baseados em atores, nesse caso, com aqueles que interagem com o sistema, e depois identificar os processos que eles iniciam ou que participam.

Outra forma de identificação de casos de uso é baseada em eventos, nesse caso, identificar os eventos aos quais o sistema deve interagir e, em sequência, relacionar os atores e o caso de uso.

É importante que todas as funcionalidades (requisitos e regras de negócio) do sistema sejam identificadas e descritas nos casos de uso. Ao documentar as funcionalidades, é possível que elas sejam rastreáveis até a implementação e o teste do software, lembrando que a rastreabilidade de requisitos é uma opção para aumentar a qualidade do processo de desenvolvimento de sistema com qualidade.

Para saber mais

Brainstorming é uma expressão inglesa formada pela união das palavras “*brain*” (cérebro/ ideias) e “*storm*” (tempestade). Então, podemos dizer que este termo significa uma “tempestade de ideias”. Para complementar o conceito desse recurso, acesse os links a seguir. Disponível em: <<https://marketingdeconteudo.com/brainstorming/>>; <<https://blog.evernote.com/ptbr/2017/02/20/tecnicas-efetivas-brainstorming/>>. Acesso em: 26 out. 2017.

Atividades de aprendizagem da seção

1. Como seria o início do desenvolvimento de um sistema, sem antes, ter uma percepção correta das necessidades (de requisitos e regras de negócio)?
2. Seria possível alcançar o resultado com sucesso sem que toda a equipe de desenvolvimento de sistema tenha o entendimento do todo e do detalhamento dos usuários do futuro software?

Seção 2

Diagrama de caso de uso

Introdução à seção

O diagrama de caso de uso mostra um conjunto de casos de uso, atores e a relação entre eles. Essa relação é identificada pelo uso de linhas de comunicação entre atores e casos de uso, que indicam o fluxo de informação. Sua finalidade é apresentar quais são os atores do sistema e a sua missão, sendo as responsabilidades a que o sistema a ser desenvolvido deverá atender.

Para representar o processo completo utilizado em um sistema, o diagrama de caso de uso passa a ser um documento narrativo representando uma sequência de eventos a ser executado por um ator (JACOBSON, 1995). Esse diagrama exerce um papel importante na análise de sistemas, sendo este o principal diagrama a ser usado no diálogo entre usuários e requisitos de sistema.

O diagrama de caso de uso é composto por quatro partes: cenário, ator, use case e comunicação (JACOBSON, 1995). Eles representam, através de um nível alto de abstração, uma visão externa do sistema, mostrando através de imagens os atores, o caso de uso e os seus relacionamentos.

Figura 3.2 | Exemplo de diagrama de caso de uso



Fonte: elaborada pelo autor.

Para saber mais

O detalhamento de cada caso de uso dependerá do risco ou da complexidade que ele possui, ou seja, quanto maior o risco, mais detalhes seu caso de uso deverá possuir; quanto maior a complexidade, maior o aprofundamento dos procedimentos e das regras de negócio.

2.1 Cenário

No cenário estão descritas as funções do sistema sob o ponto de vista do usuário. É recomendado que se concentre nas funcionalidades principais do sistema e, depois, acrescente as funcionalidades secundárias, quando necessário. O cenário pode ser identificado por fluxo normal, alternativos e de exceção.

O fluxo normal ou principal é um modo “*default*” (padrão) que o ator utiliza a funcionalidade do sistema, ou seja, é o que ele fará primariamente sempre que desempenhar a sua atividade rotineira. Pode-se dizer que tem como objetivo principal o uso da funcionalidade. Esse fluxo sempre existirá na modelagem, já os fluxos alternativos e de exceção são opcionais, mas em casos complexos é importante que estejam presentes.

O fluxo alternativo representa escolhas que o usuário pode realizar durante a execução de uma funcionalidade. Elas podem alterar o comportamento da funcionalidade.

O fluxo de exceção (ou de erro) para alguns analistas são considerados *bugs* ou seja, um erro ou defeito. Seria uma exceção do sistema que pode gerar erro. Para entender melhor os tipos de fluxos, imagine que está caminhando por uma avenida, este trajeto seria seu fluxo normal, mas ao chegar à esquina você pode escolher entre continuar seu fluxo normal ou virar à direita ou à esquerda, esses seriam seus fluxos alternativos. Durante sua caminhada, bem à frente está um buraco e acaba caindo nele, este é o seu fluxo de exceção.

2.2 Ator

O ator pode ser qualquer pessoa, departamento, dispositivo ou máquina que especifica um trabalho a ser executado, interagindo com o sistema para desenvolver uma atividade significativa. Exemplo: secretária, aluno, departamento de compras, estoque, sistema integrador, impressora, entre outros.

O ator deve ser externo ao sistema, e ter relação para os casos de uso, componentes ou classes. Ele é representado por um boneco (*stick man*) com linhas finas (um padrão da linguagem UML).

Figura 3.3 | Representação de um ator



Fonte: elaborada pelo autor.

Importante: cada ator, seja humano ou máquina, deve ser representado separadamente, ou seja, não representar mais de uma missão para o mesmo ator. Exemplo: vamos ao nosso estudo de caso, em um cenário onde o arquiteto e o engenheiro interagem no sistema, cada um com uma missão, portanto, cada um terá sua representação.

Figura 3.4 | Representação correta de caso de uso



Fonte: elaborada pelo autor.

Os atores são representados pelo papel que exercem no caso de uso. É sugerido que esse papel inicie com letra maiúscula, ou seguir sempre um padrão de escrita, para ser fácil sua identificação no texto de caso de uso. Lembrando que o ator pode ser um papel desenvolvido por pessoas e também pode ser um sistema, um dispositivo eletrônico e/ou um dispositivo mecânico.

Os atores podem ser primários, quando iniciam a sequência de interação de um caso de uso ou secundários, estes supervisionam, mantêm e/ou auxiliam na utilização do sistema.

Para saber mais

Para encontrar o ator no sistema, devemos identificar onde inicia o processo e qual o seu destino. Podemos usar algumas perguntas, como: quem utiliza o sistema? Quem o mantém? Quem inicia o processo? Quem finaliza o processo? Quem disponibiliza a informação no sistema? Quem recebe a informação?

Diante dos conceitos abordados, avançaremos utilizando o nosso estudo de caso, a construtora “More Feliz”. Vamos identificar alguns atores? Para isso, responda a algumas questões:

- Quais são os envolvidos diretos em uma construção civil?
- Quando a construtora vai realizar uma obra, é necessário ter material em estoque, então, quem cuida deste material?
- Como é a compra deste material?
- Quem recepciona os clientes? Quem são os clientes? Somente pessoa física? Pode ser empresa?
- Quem faz o faturamento?

Já temos alguns atores aparecendo por aqui! Vamos colocá-los em nosso diagrama. Você reparou que quanto mais perguntas fazemos sobre a funcionalidade do sistema, mais detalhes teremos dele?

Figura 3.5 | Possíveis atores do sistema da construtora “More Feliz”



Fonte: elaborada pelo autor.

2.3 Use case

É a identificação de um conjunto de ações efetuadas por um sistema, representa qualquer interação entre um ator e o sistema. O use case é representado por uma elipse, com o nome do caso de uso dentro ou abaixo dessa notação.

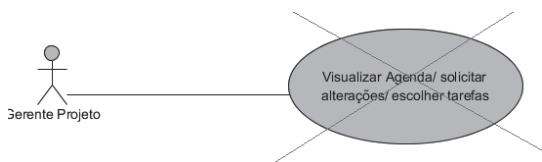
Figura 3.6 | Representação de um caso de uso



Fonte: elaborada pelo autor.

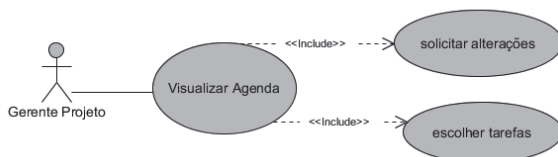
Importante: não representar mais de uma funcionalidade no mesmo use case. Ex.: um gerente de projeto precisa visualizar a situação do projeto e solicitar alterações e/ou escolher a tarefa para executar. Veja as duas figuras a seguir, uma demonstrando como não pode ser desenhado e a outra como deve ser representado.

Figura 3.7 | Representação incorreta do caso de uso



Fonte: elaborada pelo autor.

Figura 3.8 | Representação correta do caso de uso



Fonte: elaborada por

2.4 Comunicação

A comunicação é representada por um segmento de reta ligando ator e use case (ou use cases), sendo que um ator pode estar relacionado a outros atores ou vários use cases em um mesmo diagrama.

Alguns tipos de comunicação são associação, generalização e dependência.

2.4.1 Associação

A associação é a comunicação entre o ator e o caso de uso, ela é representada por uma linha sem seta, pois não representa fluxo de informação, esta pode ser: inclusão (include) e extensão (extend).

O include é usado quando um use case precisa ou é composto por outro use case. É uma relação direta entre dois use cases, informando que o comportamento do use case incluído é inserido no use case include, assim, sempre que o primeiro acontecer, obrigatoriamente o incluído ocorrerá.

O extend representa um caso de uso ampliado a partir de um caso de uso, ele é opcional, sendo que um caso de uso pode utilizar o outro. O comportamento do caso de uso extensor pode ou não ser inserido no caso de uso estendido.

Figura 3.9 | Exemplos de include e extend



Fonte: elaborada pelo autor.

2.4.2 Generalização

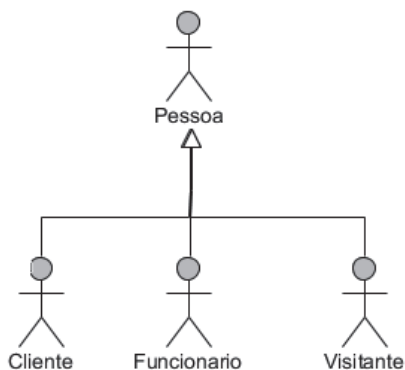
Herança entre atores

O termo herança ou generalização passa a ter o mesmo significado, apenas diferenciando em função do sentido de leitura.

Podemos utilizar herança entre os atores, na UML, quando um ator possui os mesmos papéis de outro ator. Exemplo: em nosso estudo de caso da “More Feliz”, temos vários grupos de pessoas: funcionários, clientes e visitantes, sendo que alguns dados são comuns para todos, porém, cada grupo de pessoas terá a sua particularidade.

O relacionamento de generalização ou herança ocorre quando mais de um ator se comunica com o mesmo conjunto de use case. Um ator “filho” pode se comunicar com os use cases que o ator “pai” se comunica.

Figura 3.10 | Exemplo de generalização usando atores

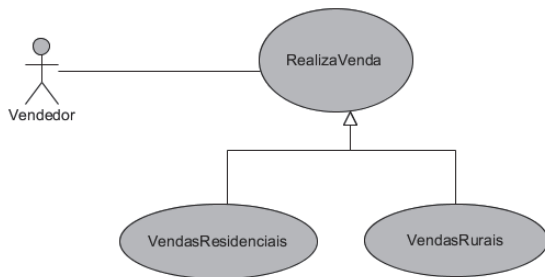


Fonte: elaborada pelo autor.

Herança entre use cases

Assim como ocorre a herança entre atores, podem ocorrer herança entre use cases, neste caso, o use case “filho” herda do “pai” seu comportamento e significado. Imagine uma empresa de engenharia onde temos a venda de imóveis, temos os contratos para imóveis residenciais que são diferentes dos contratos de imóveis rurais.

Figura 3.11 | Exemplo de generalização usando caso de uso



Fonte: elaborada pelo autor.

Para saber mais

O diagrama de caso de uso tem como objetivo definir as principais funcionalidades do sistema e sua interação com os usuários, representando como o software deverá atender aos atores (futuros usuários do sistema).

Para construir um diagrama de caso de uso, podemos seguir os passos:

- 1 Iniciar um levantamento junto ao usuário que utilizará o sistema para identificar os atores e o caso de uso.
- 2 Criar uma lista dos possíveis atores.
- 3 Criar uma lista dos possíveis casos de uso.
- 4 Repassar ambas as listas, verificando se não tem duplicidade ou falta de algum item.
- 5 Fazer o relacionamento dos atores com seus casos de uso.
- 6 Novamente junto ao usuário, realizar um novo levantamento para entender como o sistema deve atender aos casos de uso.

Dicas para construção de diagrama de caso de uso

Para se construir um diagrama de caso de uso, é interessante colocar os casos de uso mais importantes ou mais críticos para o sistema. Coloque verbos nos nomes dos casos de usos, isso ajudará no entendimento.

Para ajudar na identificação, faça uma lista com todos os nomes dos casos de usos. É relevante informar todos os requisitos de um caso de uso.

Evite um número elevado de casos de uso em um mesmo diagrama, verifique os itens mais importantes do sistema e separe em fragmentos. Ao separar em diagramas por afinidades de funcionalidades ou regras de negócios (de um mesmo processo), facilitará o entendimento e a construção do sistema.

Use palavras que identifiquem o caso de uso, exemplo: ManterEstoque, GerenciarVendas, assim por diante.

Para saber mais

Podemos dizer que um software é bem-sucedido quando, na entrega, o usuário final não encontra divergência entre o que solicitou e o que foi desenvolvido.

Retornaremos ao estudo de caso da nossa construtora “More Feliz”, volte ao texto inicial, veja que temos uma estrutura de departamentos e o levantamento tem que ser baseado nesta estrutura. Em alguns momentos deste capítulo iniciamos os descritivos de caso de uso, agora, verificaremos como ficarão os diagramas. Repare que a descrição do caso de uso tem que estar de acordo com o diagrama.

Iniciaremos pela venda dos imóveis. Como vimos, temos um consultor de vendas que auxilia nesse processo, ele busca no sistema os imóveis já cadastrados pelo assistente administrativo. Tanto o consultor de vendas quanto o assistente administrativo têm que estar no cadastro de RH. Quando concluída a venda, o próprio sistema atualiza a situação do imóvel.

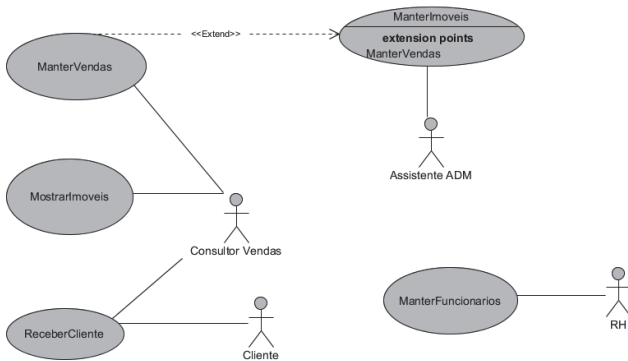
Quadro 3.3 | Descrição do caso de uso

Caso de uso	Ator(es)	Descrição
Manter imóveis	Assistente Adm.	Os imóveis para venda têm seu cadastro atualizado pelos assistentes administrativos, eles informam todos os detalhes, como tipo (casa, apartamento, comercial), metragem, quantidade de cômodos, quantidade de vagas na garagem (no caso de o imóvel ser em edifício).
Mostrar imóveis	Consultor de vendas	A construtora possui um centro para a venda de seus imóveis, com consultores treinados, que atendem o cliente, mostram o portfólio e informam sobre as opções de pagamento.
Manter vendas	Consultor de vendas	O cliente tem várias opções de pagamento para a aquisição do imóvel, e na venda, o consultor finaliza os documentos e faz o acompanhamento até a quitação do imóvel. Ao finalizar a venda, o sistema já atualiza o imóvel, colocando que não está mais disponível para venda.
Manter funcionário	Rh	O departamento de Rh cuida da contratação de todos os funcionários da empresa. Faz o controle de férias, horas extras e verifica a necessidade de treinamentos.

Fonte: elaborado pelo autor.

Vamos continuar com outro departamento da construtora?

Figura 3.12 | Diagrama de caso de uso para venda de imóveis



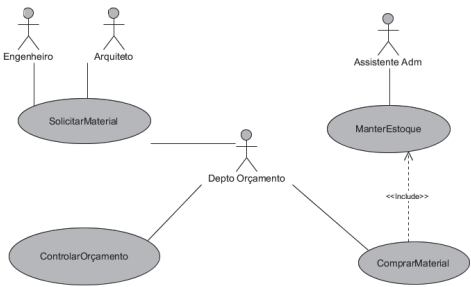
Fonte: elaborada pelo autor.

Quadro 3.4 | Descrição do caso de uso

Caso de uso	Ator(es)	Descrição
Solicitar material	Engenheiro Arquiteto	Antes de iniciar a construção da obra, o engenheiro encaminha a solicitação do material a ser usado para o departamento de orçamentos.
Manter estoque	Assistente Adm.	A empresa trabalha com grandes construções e cada uma tem seu estoque local de material, e fica um assistente com a missão de dar "baixa" ou "acrescentar" o material. Ele também pode acionar outras construções para solicitar material de urgência.
Controlar orçamento	Departamento de orçamento	Todo o material solicitado pelo engenheiro, o departamento de orçamento realiza a cotação de preços, finaliza a compra e repassa para as obras.
Realizar Mkt	Departamento de marketing	O departamento de marketing faz a propaganda dos novos lançamentos da construtora, eles devem realizar pesquisas de público-alvo, elaborar estratégias de marketing e fazer a cotação do custo destas estratégias
Comprar material	Departamento de orçamento	Após realizar a cotação do material, o departamento de orçamento analisa a melhor opção para concluir a compra. Ao chegar a mercadoria comprada, é lançada no estoque.

Fonte: elaborado pelo autor.

Figura 3.13 | Diagrama de caso de uso de compra de material



Fonte: elaborada pelo autor.

Agora chegou a sua vez. Com os atores e o caso de uso que levantou, complemente seus diagramas.

Questão para reflexão

Você acredita que consegue elaborar um diagrama de caso de uso completo e correto sem conhecer a situação real, as regras de negócio da organização?

Atividades de aprendizagem da seção

1. Identifique os atores e os casos de uso do cenário a seguir:

“Para ter acesso ao interior da biblioteca, o aluno deverá estar com a matrícula ativa, o que deve ser informado pela secretária. A bibliotecária deverá cadastrar os livros, informando sua situação. Ao realizar um empréstimo, a bibliotecária deve checar se o aluno está com livros em atraso”.

2. Quais são os objetivos do diagrama de caso de uso?

3. Analise a frase a seguir, informe se é verdadeira ou falsa e justifique:

“O objetivo do diagrama de caso de uso é identificar as operações e os atributos dos envolvidos no processo de criação de um software”.

o tipo de imóvel desejado, o consultor mostra as opções disponíveis no portfólio da construtora”.

Para as questões 4, 5 e 6, utilize o texto a seguir:

“Quando o cliente chega à construtora, a recepcionista o recebe e pergunta em que pode ajudar. Se o cliente deseja conhecer os imóveis que a empresa tem para vender, ela chama um consultor de vendas, que verifica qual tipo de imóvel o cliente deseja. A empresa trabalha com imóveis residenciais (casas e apartamentos) e imóveis comerciais. Após o cliente especificar o tipo de imóvel desejado, o consultor mostra as opções disponíveis no portfólio da construtora”.

4. Marque a alternativa mais correta com relação aos atores:

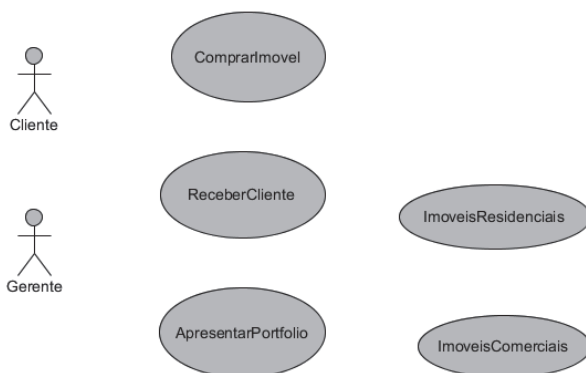
A. () Portfólio e cliente.

B. () Portfólio, cliente e recepcionista.

C. () Cliente, consultor de vendas e recepcionista.

D. () Portfólio, cliente e consultor de vendas.

5. Os atores a seguir se relacionam com algum caso de uso? Caso seja sim, qual?



Fonte: elaborada pelo autor.

Fique ligado

O diagrama de casos de uso auxilia a comunicação entre os analistas e o cliente; ele descreve os cenários que mostram as funcionalidades do sistema usando a visão do usuário. Seu cliente deverá ver nesse diagrama as principais funcionalidades de seu sistema.

Para concluir o estudo da unidade

Vimos, neste capítulo, o diagrama de caso de uso, seus componentes e notações e que seu objetivo é representar graficamente a relação entre atores e casos de uso do sistema. Existem várias ferramentas que auxiliam na criação do diagrama de caso de uso, nos exemplos deste capítulo, usamos a ferramenta Visual Paradigm. Vimos também que o caso de uso descreve uma sequência de ações e transações que interagem com o ator. Estudamos que é possível a generalização entre atores e caso de uso.

Observamos que o diagrama de caso de uso é importante para auxiliar a comunicação com o usuário do sistema. Para seu desenvolvimento, temos que inicialmente encontrar os atores, o caso de uso e fazer seu relacionamento.

Fizemos exercícios na prática com o levantamento inicial no estudo de caso da construtora “More Feliz”. Vimos que quanto mais perguntas fazemos, mais detalhes teremos do sistema.

Até o próximo capítulo!

Atividades de aprendizagem da unidade

Vamos continuar com nossa prática. Aqui estão alguns exercícios que já foram aplicados em concursos. Mãos à obra!

1. O tipo de relacionamento entre atores nos diagramas de casos de uso na linguagem de modelagem unificada (UML), é conhecido como: (Prova ano: 2015, banca: UERJ, órgão: UERJ. Prova: analista de sistemas - desenvolvimento)

- A) Inclusão.
- B) Herança.
- C) Extensão.
- D) Comunicação.
- E) Extend.

2. Sobre o relacionamento de generalização entre classes, analise as três afirmações a seguir: (Ano: 2013, banca: FUNRIO, órgão: INSS. Prova: analista - tecnologia da informação)

I – Uma classe é uma generalização de outra classe se toda instância desta última for também uma instância da primeira.

II – Herança múltipla é a situação em que uma classe pode ter mais de uma superclasse.

III – Subclasses de uma classe abstrata também podem ser abstratas, formando uma hierarquia de classes abstratas.

Assinale a alternativa correta:

- A) Somente a I está correta.
- B) A I e a II estão corretas.
- C) A I e a III estão corretas.
- D) A II e a III estão corretas.
- E) Todas estão corretas.

3. Na modelagem de casos de uso, em sistemas complexos, em vez de representar todos os casos de uso em um único diagrama, pode-se adotar a abordagem de criar vários diagramas, de acordo com as necessidades de visualização. Analise as três opções a seguir: (Ano: 2013, banca: FUNRIO, órgão: INSS. Prova: analista - tecnologia da informação)

I – Diagrama exibindo um caso de uso e seus relacionamentos.

II – Diagrama exibindo todos os casos de uso para um ator.

III – Diagrama exibindo todos os casos de uso a serem implementados em um ciclo de desenvolvimento.

Qual(is) afirmativa(s) pode(m) ser considerada(s) modelo(s) de particionamento do diagrama de casos de uso?

A) Apenas a I.

B) A I e a II.

C) A I e a III.

D) A II e a III.

E) Todas.

4. São características da análise estruturada e da análise orientada a objetos, respectivamente: (Ano: 2012, banca: FCC, órgão: TST. Prova: analista judiciário - analista de sistemas)

A) A organização do código-fonte em pacotes e o uso de diagrama de classes.

B) Programas elaborados com o uso de funções e determinação do dicionário de dados.

C) O uso de diagramas de sequência e o uso do diagrama de contexto.

D) A modelagem do fluxo de dados e a abstração de conceitos do mundo real.

E) A técnica de encapsulamento e a extensão de classes com a aplicação de herança.

5. Na UML 2.0, é um exemplo de diagrama estrutural: (Ano: 2009, banca: TJPR, órgão: TJ-PR. Prova: informática - analista de sistemas)

A) Diagrama de transição de estados.

B) Diagrama de caso de uso.

C) Diagrama de classes.

D) Diagrama de atividade.

E) Diagrama de sequência.

6. Diagramas são os meios utilizados para a visualização dos blocos de construção da UML, utilizando representações gráficas de um conjunto de elementos que permitem visualizar o sistema sob diferentes perspectivas. Na UML 2.0, os diagramas são divididos em três categorias: (Concurso: TJ-PR - 2009 - Analista | Prova: TJ-PR - 2009 - TJ-PR - analista de sistemas)

- A) Estruturais, temporais e de instalação.
- B) Estruturais, comportamentais e de interação.
- C) Caso de uso, classes e atividade.
- D) Sequência, tempo e classe.
- E) Temporais, estruturais e de instalação.

Referências

- BEZERRA, E. **Princípios da análise e projeto de sistemas com UML**. Rio de Janeiro: Elsevier, 2007.
- BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **Unified modeling language user guide**. 2. ed. Addison-Wesley Object Technology Series, 2005.
- CHEN, P. **Active conceptual modeling of learning**: Next Generation Learning-Based System Development. com Leah Y. Wong (Eds.). Springer. 2007.
- FOWLER, M.; SCOTT, K. **UML Distilled** - A Brief Guide to the standard Object Modeling Language. Addison Wesley Longman, 2002.
- GUÉDES, G. T. **UML 2**: uma abordagem prática. 2. ed. São Paulo: Novatec, 2011.
- JACOBSON, I. Object-oriented Software Engineering. Addison Wesley Longman, 1995.
- LARMAN, C. **Utilizando UML e padrões**. 3. ed. Porto Alegre: Bookman, 2007.
- RUMBAUGH, J. et al. Modelagem e projetos baseados em objetos. Rio de Janeiro: Campus, 1994.
- VISUAL. **Package diagram**. [s.d.]. Disponível em: <<https://www.visual-paradigm.com/VPGallery/diagrams/Package.html>>. Acesso em: 10 jul. 2017.

Modelagem de classes de análise

Polyanna P. Gomes Fabris

Objetivos de aprendizagem

- Compreender o que é modelagem de classes de análise e como os diagramas de classe são estruturados.
- Entender quais são as técnicas para identificação de classes e a construção do diagrama de classes.

Seção 1 | Modelagem de classes de análise: conceito, componentes e notação do diagrama de classes

Nesta seção será apresentado o modelo de classes de análise, construído para representar o aspecto estrutural estático de uma colaboração. Este modelo permite a compreensão de como o sistema está estruturado internamente. Seu objetivo é a descrição do problema representado pelo sistema a ser desenvolvido em um nível alto de abstração (BEZERRA, 2015). A representação gráfica do aspecto estrutural estático é o diagrama de classes. Todos os detalhes para a construção desse diagrama também serão apresentados nesta seção.

Seção 2 | Técnicas para identificação de classes e construção do diagrama de classes

Na Seção 2 apresentaremos algumas técnicas de identificação de classes e demonstraremos como construir um diagrama de classes.

Introdução à unidade

Caro aluno, nesta unidade apresentaremos os aspectos relacionados à modelagem de classes de análise.

O modelo de classes de análise tem como objetivo descrever o problema representado pelo sistema a ser desenvolvido em um nível alto de abstração, permitindo que a estrutura interna do sistema seja compreendida. Para representar esse modelo, temos o diagrama de classes e sua descrição textual associada. A evolução do modelo ocorre conforme a evolução do sistema que está sendo desenvolvido (BEZERRA, 2015). A Seção 1 abordará os conceitos relacionados ao modelo de classes de análise.

Para modelar um sistema orientado a objeto, precisamos entender que a sua composição é feita por uma coleção de objetos que são colaborativos para realizar as tarefas do sistema. Dessa forma, ao descrever o que é identificar as classes, entende-se que o processo visa saber quais são os objetos que fazem parte do sistema. A identificação de classes pode ser dividida em duas atividades: classes candidatas e aplicação de princípios para eliminação das classes candidatas (BEZERRA, 2015). A Seção 2 apresentará os detalhes da identificação de classes. Complementarmente, apresentaremos a construção do diagrama de classes.

Seção 1

Modelagem de classes de análise: conceito, componentes e notação do diagrama de classes

Introdução à seção

Caro aluno, esta é a primeira seção da presente unidade. Entende-se que os diagramas UML têm como objetivo apresentar as múltiplas visões dos sistemas, contemplando os mais diversos aspectos, nas quais cada diagrama será um complemento do outro e promoverá a redução de falhas durante a fase de desenvolvimento (GUEDES, 2007). Ao realizar a modelagem dos diagramas UML, esta permitirá que sejam mapeados para uma linguagem de programação específica ou vice-versa. Com relação à documentação, a UML suporta a criação e a documentação de toda a análise gerada para o software desenvolvido.

Quando falamos na categoria estrutural, percebemos que existem diversas formas de modelagem. Dentre essas formas, estão as classes de análise. A presente unidade foca seus estudos na apresentação da modelagem das classes de análise, apresentando os detalhes que compõem sua estrutura, o detalhamento das principais funções e as técnicas de modelagem. Para exemplificar os elementos que compõem as classes de análise, usaremos um estudo de caso que está descrito logo na sequência do nosso texto. Tenha uma boa leitura!

1.1 Modelagem de classes de análise

Bezerra (2015) afirma que uma determinada funcionalidade externa de um sistema orientado a objetos é fornecida por meio de colaboração entre os objetos. Essa colaboração pode ser visualizada considerando dois aspectos.

O primeiro aspecto é o dinâmico, no qual são descritas as trocas de mensagens entre cada objeto e sua reação aos eventos que ocorrem no sistema. O segundo aspecto é o estrutural estático, o qual permite a compreensão de como está a estrutura interna do sistema para que as estruturas externas possam ser desenvolvidas, apresentando a interação entre os objetos no decorrer do tempo e a estrutura de classes de

objetos e seus relacionamentos são representados (BEZERRA, 2015). Desse modo, entende-se que o modelo de classes é composto pelo diagrama de classes e da sua descrição textual relacionada. Vamos conhecer um pouco sobre essa estrutura?

1.1.1 Diagrama de classes

De acordo com Guedes (2007), o diagrama de classes é o mais utilizado entre os diagramas UML. Dentre seus principais objetivos, pode-se citar como principal, a visualização das classes utilizadas pelo sistema e como é o relacionamento entre si. O diagrama de classes possui uma visão estática, a qual é representada por classes organizadas e com uma estrutura lógica definida. Ao modelar esse diagrama, cada classe é abstraída com a sua lista de atributos e métodos.

Para Bezerra (2015), o diagrama de classes é usado na construção do modelo de classes, iniciando no nível de análise até o nível da especificação, sendo considerado o mais rico em termos de notação. Nos próximos tópicos, serão descritos os elementos utilizados para construir o modelo de classes.

Dentre as características de um diagrama de classes, podemos citar (PENDER, 2003):

- Definir os recursos essenciais de um sistema;
- Definir os relacionamentos entre os recursos;
- Gerar código;
- Servir de base para a construção de outros diagramas.

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

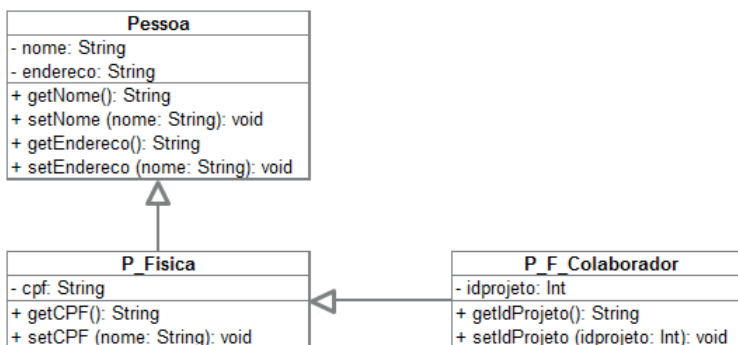
Para que você possa entender melhor como é composta a estrutura do diagrama de classes, detalharemos os conceitos de classes e seus relacionamentos.

a. Classes

De uma forma simples, vamos pensar onde estão as classes no contexto apresentado no nosso estudo de caso “**Construtora More Feliz**”, abordada nas unidades anteriores. Imagine que você está trabalhando na construtora “More Feliz”, todos os dias você interagirá com clientes, fornecedores ou colaboradores, e estes podem representar pessoas físicas ou jurídicas. Você pode perceber que todos possuem algumas características que são similares, tais como: nome, endereço, número de telefone, entre outros. Ao analisar estes grupos (subclasses, você verá este conceito na sequência do livro), você percebe que eles pertencem a um grupo superior, pessoas (superclasses, conceito abordado na sequência). Nesse contexto, tanto clientes, fornecedores ou funcionários representam classes, assim como pessoas também serão uma classe. A organização hierárquica dessas classes e seus relacionamentos serão abordados na próxima seção.

Observe a figura a seguir. Veremos os conceitos aplicados.

Figura 4.1 | Exemplo de classes



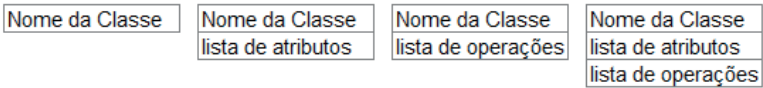
Fonte: elaborada pela autora.

A classe possui a representação gráfica de um retângulo ou “caixa”, composto por três compartimentos (no máximo). O primeiro compartimento mostra o nome da classe. No segundo, os atributos são declarados. Por fim, no terceiro, são declaradas as operações (BEZERRA, 2015). Cada ocorrência de uma classe será chamada objeto ou instância.

De acordo com Pender (2003), o diagrama de classes é o coração do processo de modelagem de objetos. Esse diagrama representa

a definição dos recursos essenciais para a operação do sistema. Por recursos, entende-se que representam pessoas, materiais, informações e comportamentos. Observe a estrutura do diagrama de classes na Figura 4.2

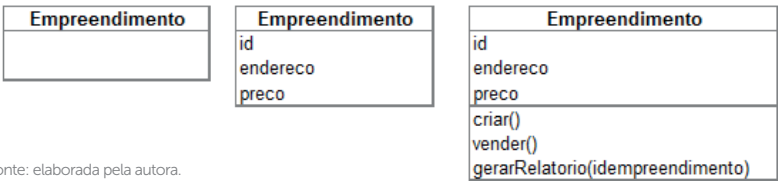
Figura 4.2 | Possíveis notações de uma classe



Fonte: Bezerra (2015).

Cada atributo de uma determinada classe corresponderá a uma descrição de dados que são armazenados pelos objetos de uma classe. Já as operações representam as ações que os objetos podem realizar. Para exemplificar o diagrama de classes, observe a figura a seguir:

Figura 4.3 | Exemplo de diagrama de classes

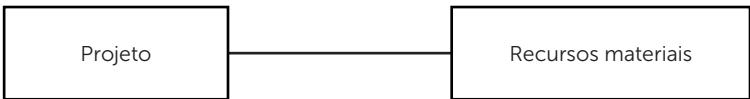


Fonte: elaborada pela autora.

b. Associações

Podemos descrever a associação como um relacionamento que conecta duas ou mais classes, de forma que a colaboração entre as instâncias de classes seja demonstrada durante a execução do sistema. Para representar graficamente a associação, nós usaremos um segmento de reta, conectando as classes as quais os objetos relacionados pertencem (BEZERRA, 2015).

Figura 4.4 | Exemplo de associação entre objetos

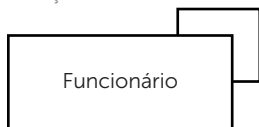


Fonte: elaborada pela autora.

A associação pode ser unária (reflexiva), binária ou n-ária:

- A **associação unária** é conhecida como autoassociação, na qual a classe é associada a ela própria. Por exemplo, a construtora “More Feliz” possui um quadro de funcionários. Cada funcionário possui um gestor responsável, que também pertence ao quadro de funcionários, assim, teremos uma associação da classe Funcionário com ela própria.

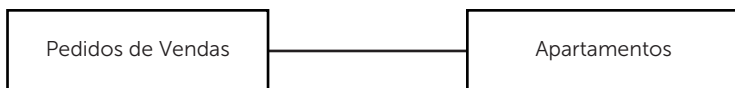
Figura 4.5 | Exemplo de associação unária



Fonte: elaborada pela autora.

- A **associação binária** conecta duas classes (MELO, 2002). Por exemplo, a construtora “More Feliz” possui o controle de vendas dos apartamentos, os pedidos de vendas da construtora serão relacionados aos apartamentos construídos.

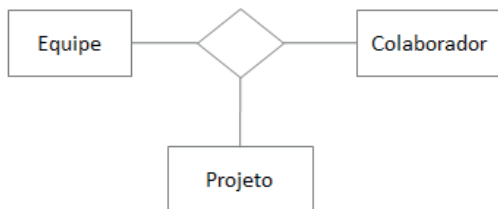
Figura 4.6 | Exemplo de associação binária



Fonte: elaborada pela autora.

- A **associação n-ária** contém mais de duas classes por relacionamento e são conectadas por um elemento na forma de losango. Pense no seguinte exemplo, a construtora “More Feliz” trabalha com projetos, um engenheiro civil trabalha em uma equipe de projetos em um determinado projeto. Assim, uma equipe de projetos possui engenheiros civis diferentes a cada projeto executado.

Figura 4.7 | Exemplo de associação N-ária



Fonte: elaborada pela autora.

Para representar uma associação, é importante observar as suas características: multiplicidade, nome, direção da leitura e seus papéis:

- **Nome da associação:** o nome deve ser descrito próximo à linha de representação da associação, porém, não devemos adicioná-lo próximo às extremidades. Ao adicionar o nome da associação, ele poderá ser acompanhado de um triângulo preenchido, indicando a direção em que o nome deve ser lido.

Figura 4.8 | Exemplo de nome de associação



Fonte: elaborada pela autora.

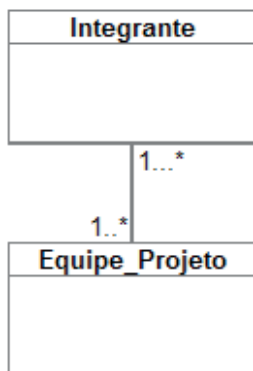
- **Multiplicidade:** representa a quantidade de instâncias de um relacionamento (MELO, 2002). Provavelmente, trata-se da propriedade mais importante de uma associação (BEZERRA, 2015). A multiplicidade é representada por um subconjunto de um conjunto de números inteiros não negativos, em uma sequência de intervalos inteiros e que são separados por uma vírgula, representado da seguinte forma: limite-inferior...limite-superior (MELO, 2002). A simbologia usada para representar a multiplicidade está descrita na tabela a seguir.

Tabela 4.1 | Simbologia para representar multiplicidades

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

Fonte: adaptada de Bezerra (2015).

Figura 4.9 | Exemplo de multiplicidade



Fonte: elaborada pela autora.

- **Papéis:** também chamado de nome de fim da associação, é colocado nas extremidades do caminho da associação, indicando o papel representado pela classe na associação.

Figura 4.10 | Exemplo de papéis em uma associação



Fonte: elaborada pela autora.

Quando falamos de associação, existem dois tipos de relacionamentos em uma categoria específica em toda parte. Esse tipo de relacionamento ocorre entre dois objetos, indicando que um objeto está contido em outro. Na UML, existem dois tipos de relacionamento nessa categoria: agregação e composição.

- **Agregação:** representa uma propriedade fraca, já que uma determinada classe, considerada a “parte”, pode estar contida em outras agregações. As agregações podem ser assimétricas, em que um objeto A é parte do objeto B, o objeto B não pode ser parte do objeto A. Elas propagam o comportamento que se aplica a um todo, de forma automática, ou seja, aplica-se às suas partes (BEZERRA, 2015). Para representar a agregação, usaremos o elemento no formato de um losango na extremidade da classe.

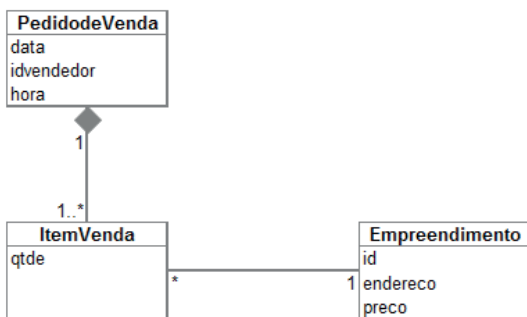
Figura 4.11 | Exemplo de agregação



Fonte: elaborada pela autora.

- **Composição:** basicamente, as características da composição são similares às de agregação, existindo o relacionamento de objeto, “todo”, e outro, objeto “parte”. No entanto, na agregação existe a independência de “vida” dos objetos, enquanto na composição, o objeto “todo” é composto pelo objeto “parte”, de modo que o objeto “parte” não tem sentido sem que haja o objeto “todo”.

Figura 4.12 | Exemplo de composição

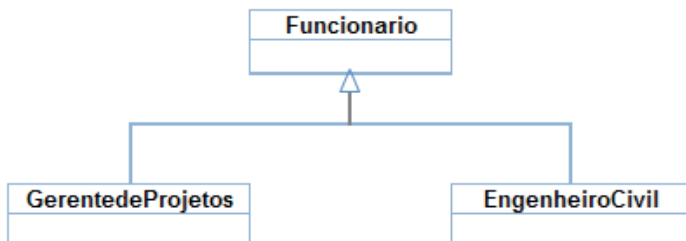


Fonte: elaborada pela autora.

c. Generalizações e especializações

Quando criamos um modelo de classes, precisamos pensar nos relacionamentos entre elas. Esses relacionamentos representam aspectos de generalidade ou especificidade entre os elementos envolvidos. A generalização refere-se ao relacionamento entre um elemento geral e um específico. Esse relacionamento é representado por uma seta vazada e vazia, direcionada do elemento específico para o geral. Para exemplificar o diagrama de classes com o relacionamento da generalização, temos a superclasse **Funcionário** e as subclasses **Gerente de Projetos** e **Engenheiro Civil**.

Figura 4.13 | Exemplo de generalização



Fonte: elaborada pela autora.

As relações do tipo generalizações são usadas para demonstrar as relações de herança. A herança permite que as subclasses herdem o estado e o comportamento das suas superclasses, as subclasses podem ter seus próprios estados e comportamentos e alterar os métodos herdados. Além disso, a herança permite que os códigos possam ser reutilizados e os frameworks sejam definidos (SILVA; VIDEIRA, 2001).

d. Visibilidade












Outro aspecto importante no diagrama de classes é a visibilidade, em que se identifica por quem uma propriedade, sendo o atributo ou a operação, será usada (MELO, 2002). Essa propriedade é definida por palavras-chaves ou ícones. A seguir, apresentaremos 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.

e. Notação

Para desenhar um diagrama de classes, utilize as notações que mostraremos nas imagens a seguir. Estas notações estão ilustradas utilizando a ferramenta CASE Visual Paradigma Professional V14.

Tabela 4.2 | Notações de um diagrama de classe

Notação	Nome	Definição
	Abstraction	Relaciona dois elementos ou conjuntos de elementos que representam níveis 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 pela autora.

Questão para reflexão

Faça a leitura do estudo de caso. Após, identifique três classes, construa uma visão parcial de um diagrama de classes e mostre o relacionamento do tipo composição. Quais foram as principais percepções ao elaborar este diagrama? Compartilhe com o seu professor.

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

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

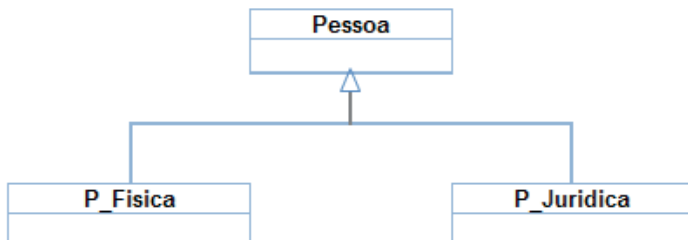
IBM. **Relacionamentos em diagramas de classe**. [s.d.]. Disponível em: https://www.ibm.com/support/knowledgecenter/pt-br/SS8PJ7_9.6.1/com.ibm.xtools.modeler.doc/topics/crelsme_clssd.html. Acesso em: 1 set. 2017.

AGILE MODELING. **UML 2 Class Diagrams: an agile introduction**. [s.d.]. Disponível em: <http://www.agilemodeling.com/artifacts/classDiagram.htm>. Acesso em: 1 set. 2017.

COURSERA. **Lecture 31 - Diagrama de Classes UML: classe, associação e multiplicidade**. [s.d.]. Disponível em: <https://pt.coursera.org/learn/orientacao-a-objetos-com-java/lecture/7adrt/diagrama-de-classes-uml-classe-associacao-e-multiplicidade>. Acesso em: 1 set. 2017.

Atividades de aprendizagem

1. Analise a figura a seguir:



O tipo de relacionamento ilustrado é:

- a) Generalização.
- b) Dependência.
- c) Especialização.
- d) Agregação.
- e) Instanciação.

2. Com relação ao diagrama de classes, analise as características a seguir:

- I - Definir os recursos essenciais de um sistema.
- II - Definir os relacionamentos entre os recursos.
- III - Gerar código.
- IV - Não fornece foco para todos os outros diagramas.

Marque a alternativa que contém as informações corretas sobre as características:

- a. I e II estão corretas.
- b. Apenas a III está correta.
- c. I, II e III estão corretas.
- d. Apenas a IV está correta.
- e. Apenas a I está correta.

3. Leia as proposições a seguir:

- I - Associações: descrever a associação como um relacionamento que conecta duas ou mais classes, de forma que a colaboração entre as instâncias de classe seja demonstrada durante a execução do sistema.
- II - Multiplicidade: representa a quantidade de instâncias possíveis naquele relacionamento.
- III - Agregação: representa uma propriedade forte, já que uma determinada classe, considerada o "todo", pode estar contida em outras agregações.

Sobre a descrição desses conceitos, é possível afirmar que:

- a. Apenas a I está correta.
- b. Apenas a II está correta.
- c. Apenas a III está correta.
- d. I e II estão corretas.
- e. II e III estão corretas.

4. Analise a imagem a seguir:



Essa imagem apresenta a representação gráfica da associação. Baseando-se no estudo que você fez nesta unidade, qual elemento foi adicionado de forma incorreta?

- a. Multiplicidade.
- b. Nome da classe.
- c. Ordem da representação das classes.

- d. Localização do nome da associação.
- e. Não existem elementos adicionados errados.

5. Ao construir um diagrama de classes, é necessário entender a sua estrutura. Assinale a alternativa que NÃO apresenta esses elementos:

- a. Métodos, classes e relacionamentos de dependência.
- b. Classes, entidades, processos de negócio.
- c. Classes, subclasses e superclasses.
- d. Visibilidade, estereótipos e relacionamento de dependência.
- e. Classe, atributo e parâmetro para cada método.

Fique ligado

Nesta seção apresentamos o contexto relacionado à modelagem de classes de análise. Para entender esse modelo, é necessário conhecer os aspectos relacionados ao diagrama de classes. O objetivo foi abordar que esse diagrama representa o coração do processo de modelagem de objetos. Ainda, nesta seção, apresentamos os conceitos relacionados à sua construção, bem como as estruturas, os relacionamentos e os comportamentos. Foram apresentadas algumas representações gráficas do modelo. Ao final do texto, disponibilizamos alguns materiais complementares. Como foi o estudo desta seção? Compartilhe com o seu professor!

Seção 2

Técnicas para identificação de classes e construção do diagrama de classes

Introdução à seção

Quando falamos em desenvolvimento de um software orientado a objetos, é necessário entender que ele é composto por uma coleção de objetos colaborativos na realização de tarefas neste sistema. Ao realizar a modelagem das classes, são realizadas duas atividades: classes candidatas e aplicação de princípios para eliminação das classes candidatas que sejam desnecessárias (BEZERRA, 2015).

Dentre as técnicas usadas para identificação de classes, estão: taxonomia de conceitos, análise textual de Abbot e Análise de Robustez.

2.1 Taxonomia de conceitos

Conforme descrito por Bezerra (2015), uma das abordagens utilizadas para identificação das classes é a taxonomia de conceitos:

- Conceitos concretos: por exemplo, edifícios, salas de aula, carros, entre outros;
- Papéis desempenhados por seres humanos: professores, alunos, empregados, clientes, entre outros;
- Eventos: reuniões, pedidos ou aulas;
- Lugares: escritórios, salas de aula ou filiais;
- Organizações: departamentos, projetos etc.;
- Conceitos abstratos: reservas, vendas ou inscrições.

2.2 Análise Textual de Abbot (ATA)

No ano de 1983, uma técnica de identificação de classes foi proposta por Russell Abbott, a Análise Textual de Abbott (ATA), que utiliza as mais diferentes fontes de informação sobre o sistema, tais como (BEZERRA, 2015) documento de requisitos, modelos do negócio ou glossários, conhecimento sobre o domínio. Para cada documento citado, os nomes que aparecem devem ser destacados. Na sequência,

os termos que são sinônimos são removidos. Os termos que sobraram serão enquadrados em umas das situações a seguir (BEZERRA, 2015)

- o termo torna-se uma classe candidata;
- o termo torna-se um atributo;
- o termo não apresenta relevância com relação aos requisitos do sistema (BEZERRA, 2015).

Nessa técnica, Abbott aplica sua proposta na identificação de operações e associações. Para identificar, a sugestão é destacar os verbos no texto, por exemplo, calcular, confirmar ou comprar (BEZERRA, 2015).

De acordo com Bezerra (2015), uma das vantagens da técnica Abbott é a sua simplicidade no destaque dos termos para detecção de componentes do modelo de classes. No entanto, mesmo apresentando essa abordagem simples, possui uma desvantagem relacionada à dependência de completude do documento usado como fonte. Também podem ser identificadas classes candidatas que não gerarão classes. Para complementar essa técnica, podem ser usadas outras técnicas, tais como as descritas na sequência, para identificar novas classes.

2.2.1 Análise de Robustez

Uma outra técnica para identificação é a Análise de Robustez. Essa técnica foi proposta por Ivar Jacobson e, posteriormente, adotada pelo processo de desenvolvimento denominado ICONIX (BEZERRA, 2015). Nessa técnica, entendemos que os objetos que compõem o sistema podem ser divididos em categorias: objetos de fronteira, objetos de controle e objetos de entidade. Denominamos essa categorização de BCE (lembrete para seus nomes originais das categorias: boundary, control e entity).

Os estereótipos usados para representar graficamente as categorias são:

Figura 4.14 | Representação gráfica BCE



Fonte: elaborada pela autora.

a. Objetos de fronteira (traduzido do inglês boundary)

Estes objetos realizam a comunicação do sistema com os atores, permitindo que o sistema interaja com seu ambiente. Existem três tipos de classes de fronteira:

- as que realizam a interface com o usuário;
- as que realizam a interface com os sistemas externos;
- as que realizam a comunicação com os dispositivos atrelados ao sistema.

Para exemplificar, analise a seguinte situação. A construtora "More Feliz" possui um cliente que realizará a compra de um empreendimento. Ao realizar a compra, é emitido um pedido de venda que passará pelo processo de confirmação. Esse pedido será lançado no sistema de faturamento da empresa. Para que o sistema realize a comunicação entre o cliente e o sistema de faturamento, é necessária a criação de um objeto que realize a comunicação. Assim, é criada uma classe: Pedido-Venda-Faturamento.

b. Objetos de controle (traduzido do inglês control)

O objeto de controle é conhecido também como controlador, tem como objetivo ser uma ponte de comunicação entre os objetos de fronteira e os objetos de entidade. É responsável pela coordenação da execução de algumas funcionalidades específicas do sistema. Normalmente, essa parte do sistema corresponde a um caso de uso, decidindo o que o sistema deve fazer quando ocorre um evento externo relevante. Dentre as responsabilidades de objetos de controle, estão:

- Realizar monitorações, a fim de responder a eventos externos ao sistema;
- Coordenar a realização de um caso de uso por meio do envio de mensagens a objetos de fronteira e a objetos de entidade;

- criar associações entre alguns objetos de entidade;
- manter valores acumulados ou derivados durante a realização de um caso de uso;
- manter o estado da realização do caso de uso (BEZERRA, 2015).

c. Objetos de entidade (traduzido do inglês *entity*)

Representam um conceito encontrado no domínio do problema, servem como repositório para alguma informação manipulada pelo sistema. Também, nessas classes, são alocadas as responsabilidades do sistema, no que diz respeito à lógica do negócio (BEZERRA, 2015).

Dentre as responsabilidades típicas desses objetos, estão:

- Informar valores de seus atributos a objetos requisitantes;
- Realizar cálculos ou impor restrições relativas às regras do negócio;
- criar e destruir objetos-parte (BEZERRA, 2015).

Pode-se concluir que a BCE atua como uma “receita de bolo” para identificação de objetos participantes da realização de um caso de uso. Essa técnica indica que para cada caso de uso devem ser aplicadas as seguintes regras:

- adicionar um objeto de fronteira para cada ator participante do caso de uso;
- adicionar um objeto de controle para o caso de uso, pois um caso de uso representa um determinado processo do negócio (BEZERRA, 2015).

2.3 Construção do modelo de classes

Com a atividade de identificar as classes e as atribuições de responsabilidades, o responsável pela modelagem deve realizar a verificação da consistência entre as classes para eliminação das inconsistências e redundâncias. Ao final do mapeamento, é gerado um diagrama de classes que apresenta uma estrutura estática relativa a todas as classes que foram identificadas anteriormente como participantes da realização de um ou mais casos de uso (BEZERRA, 2015).

Nos tópicos a seguir, apresentaremos a descrição do mapeamento

de responsabilidade e colaboradores, considerando os elementos do diagrama de classes (BEZERRA, 2015):

- a. propriedades: propriedade é um nome genérico que se aplica aos membros de uma classe, tanto atributos, quanto operações;
- b. associações: para criar associações, verifique os colaboradores de uma classe. Para definir as associações, a classe em questão é o todo e o seu colaborador corresponde à parte;
- c. organização da documentação: a partir do momento que as classes e suas responsabilidade e colaboradores foram identificados, os elementos devem ser organizados em um ou mais diagramas de classes e documentados. O conjunto de todos os diagramas de classes identificadas na análise de um sistema com sua documentação, corresponde ao **modelo de classes de análise**.

2.3.1 Como modelar um diagrama de classes

Ao iniciar a etapa de especificação dos requisitos que comporão o sistema, o responsável pela modelagem deve preocupar-se com os seguintes pontos:

- Identificação das classes que compõem o sistema;
- Identificação dos atributos de cada classe do modelo;
- Realização de análise detalhada para identificar possíveis relacionamentos que estão mapeados como atributos;
- Identificação de algumas operações que já são inicialmente conhecidas;
- Análise de classes que possuem características semelhantes, realizando remodelagem com relacionamentos de herança;
- Detalhamento dos atributos.



Questão para reflexão

A partir da análise das técnicas de identificação de classes, verifique se as técnicas podem ser combinadas entre si. Descreva essa análise e compartilhe com o seu professor.

RUP. **Diretrizes**: classe de análise. [s.d.]. Disponível em: <http://www.funpar.ufpr.br:8080/rup/process/modguide/md_acls2.htm>. Acesso em: 1 set. 2017.

Seção 5.4 - Técnicas para identificação de classes. Disponível no livro abaixo:

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

Atividades de aprendizagem

1. Para a construção de um diagrama de classes, temos várias boas práticas que podem ser adotadas para que ele esboce a estrutura estática do sistema a ser desenvolvido. Analise como os relacionamentos entre as classes de um digrama apoiam no entendimento de como será o sistema.
2. Nesta unidade, você aprendeu as técnicas de identificação de classes. As técnicas são aplicáveis no dia a dia para desenvolvimento de software?

Para concluir o estudo da unidade

Nesta unidade foram apresentadas duas seções que abordaram o contexto relacionado à modelagem de classe de análise. A primeira seção mostrou que para representar o modelo de classe, temos o diagrama de classe, permitindo que a estrutura do sistema seja compreendida. A segunda seção mostrou as técnicas para identificação de classes e como é sua composição. Para complementar sua aprendizagem, disponibilizamos e recomendamos links e livros ao longo das seções, veja em *Para saber mais* e consulte nossas referências.

Atividades de aprendizagem da unidade

1. Analise as afirmativas a seguir sobre a categorização BCE.

I - Realizar cálculos ou impor restrições relativas às regras do negócio.

II - As que realizam a interface com o usuário.

III - Criar associações entre alguns objetos de entidade.

Assinale a alternativa que apresenta a sequência correta de categorias de objetos:

a) Entidade, fronteira e controle.

b) Fronteira, controle e entidade.

c) Controle, entidade e fronteira.

d) Controle, controle e entidade.

e) Todas as descrições referem-se à fronteira.

2. Ao citarmos conceitos concretos, papéis, eventos, lugares, organizações ou conceitos abstratos, fazemos referência à qual técnica de identificação de classes?

a) Análise Textual de Abbott.

b) Taxonomia de conceitos.

c) Diagrama de classes.

d) Análise de Robustez.

e) A descrição não está fazendo referência à técnica alguma de identificação de classes.

3. Uma das vantagens da técnica _____ é a sua simplicidade no destaque dos termos para detecção de componentes do modelo de classes. A técnica de identificação de classes que preenche a lacuna da descrição é:

a) Análise Textual de Abbott.

b) Taxonomia de conceitos.

c) Diagrama de classes.

d) Análise de Robustez.

e) A descrição não está fazendo referência à técnica alguma de identificação de classes.

4. Na técnica _____, entendemos que os objetos que compõem o sistema podem ser divididos em categorias: objetos de fronteira, objetos de controle e objetos de entidade.

- a) Taxonomia de conceitos.
- b) Classes.
- c) Análise Textual de Abbott.
- d) Análise de Robustez.
- e) A descrição não está fazendo referência à técnica alguma de identificação de classes.

5. Analise a descrição: propriedade é um nome genérico que se aplica aos membros de uma classe, tanto atributos, quanto operações. Essa característica está relacionada à qual aspecto da modelagem de classes de análise?

- a) Construção do modelo de classes.
- b) Criação de uma fronteira.
- c) Criação relacionada à generalização de uma classe.
- d) Criação de um objeto.
- e) Criação de uma entidade.

Referências

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

DATHAN, B.; RAMNATH, S. **object-Oriented Analysis, Design and Implementation: an integrated approach**. 2. ed. St. Cloud, MN: Springer, 2015.

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.

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

SILVA, A.; VIDEIRA, C. **UML, metodologias e ferramentas CASE**. 2001. Disponível em: <http://www.cesarkallas.net/arquivos/livros/informatica/UML_Metodologias_e_Ferramentas_CASE_portugues_.pdf>. Acesso em: 28 set. 2017.

VISUAL. **Package diagram**. [s.d.]. Disponível em: <<https://www.visual-paradigm.com/VPGallery/diagrams/Package.html>>. Acesso em: 28 set. 2017.

ISBN 978-85-522-0300-1



9 788552 203001 >