



KLS

Análise e Modelagem de Sistemas

Análise e Modelagem de Sistemas

Claudia Werlich
Regina Fedozzi
Samuel Gonçalves da Silva

© 2020 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.

Presidência
Rodrigo Galindo

**Vice-Presidência de Produto, Gestão
e Expansão**
Julia Gonçalves

Vice-Presidência Acadêmica
Marcos Lemos

**Diretoria de Produção e
Responsabilidade Social**
Camilla Veiga

Gerência Editorial
Fernanda Miglioranza

Editoração Gráfica e Eletrônica
Renata Galdino
Luana Mercurio

Supervisão da Disciplina
Marcio Aparecido Artero

Revisão Técnica
Marcilyanne Moreira Gois
Marcio Aparecido Artero
Vanessa Cadan Scheffer

Imagens

Adaptadas de Shutterstock.

Todos os esforços foram empregados para localizar os detentores dos direitos autorais das imagens reproduzidas neste livro; qualquer eventual omissão será corrigida em futuras edições.

Conteúdo em websites

Os endereços de websites listados neste livro podem ser alterados ou desativados a qualquer momento pelos seus mantenedores. Sendo assim, a Editora não se responsabiliza pelo conteúdo de terceiros.

Dados Internacionais de Catalogação na Publicação (CIP)

Werlich, Cláudia

W489a Análise e modelagem de sistemas / Cláudia Werlich Regina Fedozzi, Samuel Gonçalves da Silva. - Londrina : Editora e Distribuidora Educacional S.A., 2020.
224 p.

ISBN 978-85-522-1683-4

1. Análise e Modelagem. 2. Sistemas. 3. Processos de Software.
I. Fedozzi, Regina. II. Silva, Samuel Gonçalves da. III. Título.

CDD 004

Jorge Eduardo de Almeida CRB-8/8753

2020

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

Introdução à engenharia de software e à análise de sistemas.....	7
--	---

Seção 1

Fundamentos da engenharia de software.....	9
--	---

Seção 2

O processo de software	26
------------------------------	----

Seção 3

Modelos de processos de software.....	37
---------------------------------------	----

Unidade 2

Processos de negócio para análise de sistemas.....	54
--	----

Seção 1

Fundamentos de processos de negócios.....	56
---	----

Seção 2

Modelagem de processos de negócio.....	71
--	----

Seção 3

Gerenciamento de processos de negócio.....	92
--	----

Unidade 3

Engenharia de requisitos	113
--------------------------------	-----

Seção 1

O processo de engenharia de requisitos	114
--	-----

Seção 2

Elicitação, especificação e validação de requisitos	132
---	-----

Seção 3

Modelagem de requisitos.....	148
------------------------------	-----

Unidade 4

Paradigma orientado a objetos	168
-------------------------------------	-----

Seção 1

Fundamentos da orientação a objetos	170
---	-----

Seção 2

Modelo do processo unificado.....	185
-----------------------------------	-----

Seção 3

Métodos orientados a objetos	203
------------------------------------	-----

Palavras do autor

A medida que a tecnologia evolui, acaba sendo cada vez mais incorporada ao nosso estilo de vida, alguns exemplos de inovações tecnológicas encontradas no nosso dia a dia são: computação em nuvem, realidades virtual e aumentada, reconhecimento facial, entre outras tantas. Porém, não podemos nos esquecer de uma inovação importante: o software.

A união de hardware e software proporciona eficiência e agilidade para diferentes aplicações, por exemplo, não adianta ter um celular de última geração sem funcionalidades e sem aplicativos (softwares) adequados. Dessa forma, a mão de obra especializada surge como uma necessidade da demanda por softwares eficientes e inovadores, o que faz surgir, por sua vez, a necessidade de técnicas para a criação, de forma eficaz, de novos sistemas. Nesse sentido, é necessário ter conhecimentos acerca de como e quais técnicas devem ser utilizadas para a Análise e Modelagem de Sistemas.

Assim, na Unidade 1 deste livro, você verá como o desenvolvimento de software evoluiu ao longo dos anos, a importância da engenharia de software para a evolução de processos de software, bem como o desenvolvimento ágil no processo de software. Além disso, você conhecerá o papel do analista de sistemas e os processos e modelos de software aplicados no desenvolvimento deste com o objetivo de reconhecer e identificar os processos de software e suas metodologias.

Na Unidade 2, você conhecerá os conceitos de processos de negócios, modelagem e gerenciamento de ferramentas de BPM unidos à gestão de negócios e à Tecnologia da Informação, com foco nos resultados e nas melhorias dos processos de negócios. E, desta forma, serão abordados os fundamentos de processos de negócio, a modelagem de processos de negócio e o gerenciamento de processos de negócio.

Na Unidade 3, serão apresentados o processo de engenharia de requisitos, a elicitação, a especificação e a validação de requisitos e a modelagem de requisitos. Você poderá aplicar e identificar os processos de engenharia de requisitos, assim como suas técnicas e análises para sistemas. Aprenderá também algumas técnicas para especificar, modelar e documentar os requisitos de um sistema.

Na Unidade 4, você irá aplicar e praticar os fundamentos e métodos orientados a objetos; conhecerá os fundamentos da orientação a objetos, o modelo do processo unificado com suas características e fases, a introdução ao Processo Unificado da Rational (*Rational Unified Process - RUP*)

e a introdução à Linguagem de Modelagem Unificada (*Unified Modeling Language* - UML).

A utilização de técnicas para a Análise e Modelagem de Sistemas traz segurança aos desenvolvedores, pois é uma garantia de que se está desenvolvendo o solicitado pelo cliente, além de ser uma forma de estabelecer padrões de qualidade para o desenvolvimento. Possibilita, também, a redução de erros no desenvolvimento de softwares, minimiza os custos orçamentários e até o tempo de desenvolvimento se as técnicas são aplicadas de maneira adequada.

Você está entrando agora no mundo do desenvolvimento de sistemas, que é um mundo de oportunidades! Sua visão sobre um software não será mais a mesma, pois este livro trará novas ideias e oportunidades sobre o desenvolvimento de sistemas.

Fique atento às suas ideias! Quem sabe você não seja um desenvolvedor de um aplicativo inovador? Boa jornada no universo da Análise e Modelagem de Sistemas!

Unidade 1

Claudia Werlich

Introdução à engenharia de software e à análise de sistemas

Convite ao estudo

Olá! Seja bem-vindo a esta unidade do livro de Análise e Modelagem de Sistemas.

A informática está revolucionando o mundo, o que tem aumentado a busca por profissionais qualificados para desenvolver novos sistemas. Dessa forma, novas oportunidades de trabalho aparecem a todo instante, e podemos considerar que este é um momento ímpar na era digital.

Como funcionário de uma empresa de desenvolvimento de software, você está ingressando no ramo da Análise de Sistemas e, com esse cargo, muitos desafios aparecerão diariamente. Seu primeiro desafio consiste em analisar o software de um possível cliente. A empresa de desenvolvimento precisa de novas ideias para sugerir ao cliente e, quem sabe, conseguir fechar um contrato de desenvolvimento. No segundo desafio, você precisará investigar o processo de software da empresa de desenvolvimento em que você trabalha e propor melhorias nos processos de desenvolvimento de software. Após os dois primeiros desafios, você deverá avaliar e propor um modelo de processo de software a partir da observação das características do software solicitado por um novo cliente.

Para vencer esses três desafios, nesta unidade você irá aprender a reconhecer e identificar processos de softwares e suas metodologias. Na Seção 1 você aprenderá sobre os fundamentos da engenharia de software e entenderá a natureza e a evolução do software, a prática da engenharia de software, os princípios da análise de sistemas e a importância do papel do analista de sistemas no desenvolvimento de um software. Na Seção 2 serão apresentados os processos de softwares, através de uma introdução ao processo de software, da estrutura de um processo genérico de software e das tarefas e modelagem das atividades do processo de software. Na Seção 3 serão introduzidos os modelos de processos de software com a integração e validação entre as atividades do processo de software, a introdução ao conceito de modelo de processo de software, os modelos de processos prescritivo, os

modelos de processos especializados e uma introdução ao desenvolvimento ágil e seus principais métodos.

Neste momento, é aconselhável uma reflexão sobre o seu potencial: você poderá se tornar um profissional bem-sucedido na área de desenvolvimento de sistemas e ou na área da análise de sistemas. E, para isso, algumas características importantes são necessárias: busca por conhecimentos e atenção às novas tecnologias do mercado. A fim de conseguir essas características, leia com atenção esta unidade, procure entrar em sites e fóruns de desenvolvimento e fique atento aos lançamentos de novas tecnologia do mercado (tanto em hardware quanto em software).

Invista no seu potencial!

Estude, boa aula!

Seção 1

Fundamentos da engenharia de software

Diálogo aberto

Caro aluno, seja bem-vindo à seção sobre fundamentos da engenharia de software.

Trabalhar na área de engenharia de software é sempre desafiador, pois um sistema nunca é igual ao outro e há constantes novidades avindas do emprego de tecnologias diferentes. Nesse universo são necessários organização e muito empenho para produzir um software de qualidade.

Pensando nisso, você foi convidado a trabalhar como analista de sistemas em uma *software house* (empresa de desenvolvimento de sistemas) e sua primeira missão é analisar um software de um possível cliente. A empresa de desenvolvimento precisa de novas ideias para sugerir ao cliente e, quem sabe, conseguir fechar um contrato de desenvolvimento.

O cliente possui uma rede de postos de coleta de exames laboratoriais no Sul do país. Sabe-se que o sistema utilizado atualmente, um desktop sem acesso à internet, já possui quase 15 anos de uso e nunca foi atualizado. Como a empresa pensa, agora, em atualizá-lo, sua missão será descobrir: é mais viável atualizar o sistema existente ou criar um novo? Quais as novas tecnologias que podem ser utilizadas em um novo sistema para uma empresa de exames laboratoriais? Quais processos deverão ser adotados para manter o software sempre atualizado? Qual o perfil dos profissionais envolvidos no processo da análise do sistema do cliente?

O primeiro passo para responder ao questionamento acima é conhecer os fundamentos da engenharia de software, por meio do estudo desta seção que conta com: a natureza e a evolução do software, a prática da engenharia de software, os princípios da análise de sistemas e o papel do analista de sistemas.

Crie uma apresentação com as soluções encontradas para gerar um debate sobre suas ideias.

Esta seção dará uma visão inicial sobre os processos de criação de um software e você perceberá que, após conhecer esses processos, seu pensamento ficará logicamente organizado e poderá entender e planejar futuros softwares de forma sistemática. Estude esta seção e resolva a situação-problema. Desafie-se!

Bons estudos!

Não pode faltar

Na sociedade atual, a utilização de qualquer tipo de software tomou grandes proporções. Atualmente, é comum o mercado de trabalho exigir que os profissionais tenham habilidades em algum software específico, ou então que as empresas treinem seus funcionários para a utilização dele, uma vez que geralmente utiliza-se algum tipo de software para a execução de tarefas diárias.

Sommerville (2011, p. 4) estabelece que os softwares são “programas de computadores com uma documentação associada e [que] os produtos de software podem ser desenvolvidos para um determinado cliente ou para um mercado mais generalizado”. Já Pressman (2016) afirma que um software de computador é um produto que profissionais da área da Tecnologia da Informação (TI) desenvolvem e para o qual darão suporte a longo prazo; além disso, abrange qualquer tipo de mídia eletrônica, consistindo na união de três elementos:

- Instruções: quando executadas, fornecem os atributos e funções de desempenhos desejados pelos usuários.
- Estruturas de dados: possibilitam aos programadores manipular as informações de forma mais adequada conforme a necessidade da aplicação.
- Documentação: é toda a informação descritiva do software, a qual detalha a operação de uso dos programas, diagramas de funcionalidades, etc.

É importante ressaltar que o software não “surgiu” do nada, houve um processo evolutivo que todos os profissionais da área de TI devem conhecer. No Quadro 1.1 pode ser observada uma cronologia da história da evolução do software e dos fatos mais impactantes que constituíram esse processo evolutivo.

Quadro 1.1 | Cronologia histórica da evolução do software

Quando?	O que?	Descrição
Década de 1940	<ul style="list-style-type: none"> Programa 	<ul style="list-style-type: none"> O programa era executável e tinha controle total sobre o computador, sendo que o software era responsável por todo o funcionamento tanto do hardware (Sistema Operacional) quanto das operações matemáticas que teria que fazer.
Décadas de 1950 e 1960	<ul style="list-style-type: none"> Sistemas Operacionais Linguagens de programação 	<ul style="list-style-type: none"> Responsáveis pelo controle do hardware (realizava a interface entre o homem e a máquina). Surgem as linguagens de programação: COBOL, LISP, ALGOL, BASIC, etc.
Décadas de 1960 e 1970	<ul style="list-style-type: none"> Crise do software Paradigmas de programação Sistemas Operacionais mais eficientes 	<ul style="list-style-type: none"> Houve um grande crescimento do número de sistemas e quase não havia planejamento e controle de processos de desenvolvimento de software. Devido ao desenvolvimento informal, havia atrasos e os custos superavam o orçamento estimado (surgindo a necessidade de criar métodos de engenharia de software). Com novas linguagens sendo criadas, nessa época o conceito de orientação a objetos é criado. O foco, no momento, era a programação desktop, de modo que foi quando surgiram o MS-DOS da Microsoft e o Macintosh da Apple.
Década de 1980	<ul style="list-style-type: none"> Computador desktop Unix Evolução da internet 	<ul style="list-style-type: none"> A década foi marcada pelo surgimento da Microsoft e pela evolução dos computadores pessoais – desktop. O Unix (Sistemas Operacionais em rede) avançaram pelo mundo. A internet começa a despontar como meio de comunicação.
Década de 1990	<ul style="list-style-type: none"> Internet Linux JAVA 	<ul style="list-style-type: none"> A internet é amplamente utilizada. Surge o núcleo do futuro Linux. Nasce a linguagem JAVA, que revoluciona o paradigma da orientação a objetos.
Década de 2000	<ul style="list-style-type: none"> Sistemas Operacionais gráficos Internet 	<ul style="list-style-type: none"> Geração do Windows da Microsoft (Windows XP, Windows Vista, Windows 70) e versões gráficas do LINUX. Softwares que utilizam a web como plataforma de funcionamento.
Década de 2010 e 2020	<ul style="list-style-type: none"> Computação em nuvem Aplicativos para dispositivos móveis Inteligência Artificial 	<ul style="list-style-type: none"> Utilizada em larga escala por diversos seguidores de empresas. Com a evolução dos dispositivos móveis os softwares para aplicativos tornaram-se essenciais. Utilização de algoritmos para a Deep Learning (Aprendizagem Profunda) e Machine Learning (Aprendizado de Máquina).

Fonte: adaptado de Fonseca Filho (2007).

Como pode ser observado no Quadro 1.1, à medida que novas formas de utilização do hardware e do software se tornam acessíveis à população, fica evidente a maior necessidade de melhorar os softwares; além disso, as necessidades dos clientes passam a mudar e vão se alterando conforme a tecnologia é disponibilizada, vejamos:

1. Na década de 1990, uma empresa precisava de um software e (talvez) de um site para que ficasse conhecida na internet; o site era utilizado como um cartão de visitas.
2. Na década de 2000, a mesma empresa precisava de um software com acesso à internet para disponibilizar recursos para seus clientes.
3. A partir de 2010, a empresa precisava de um software, um site e um aplicativo para que seus serviços pudessem ser acessados por diversos dispositivos e para que o armazenamento da base de dados estivesse na nuvem.
4. A partir de 2020, a mesma empresa já está pensando em utilizar a Inteligência Artificial para melhorar seus processos gerenciais.

Com os fatos supracitados, fica evidente que à medida que novas tecnologias se tornam populares, maiores são as necessidades da produção de software para atender as demandas da sociedade.

Assimile

Qual a diferença entre software e sistema?

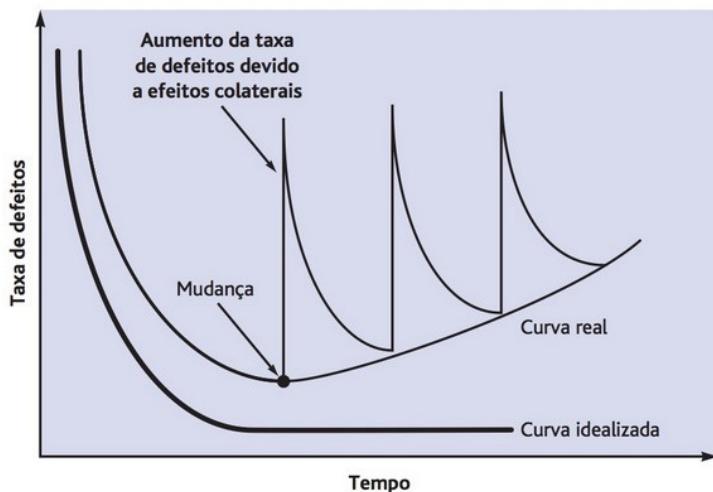
Conforme Pressman (2016), o termo software é definido como um programa de computador, uma sequência lógica de instruções a serem seguidas e/ou executadas na manipulação de um determinado conjunto de informações.

Sommerville (2011) destaca que sistema é um conjunto de componentes inter-relacionados que funcionam de forma unificada para atingir um certo objetivo. Essa definição de sistema é conhecida em algumas outras áreas, como a de engenharia de sistemas, sendo alguns exemplos: Sistema Operacional, sistema acadêmico e sistema bancário. Um sistema é formado por um determinado número de programas separados e por arquivos de configurações para eles, podendo incluir documentação específica para descrever a estrutura do sistema, documentação de usuário, etc. Um sistema possui um conjunto de partes que interagem entre si, visando a um objetivo em comum. É um conjunto de software, hardware e recursos humanos.

As mudanças sempre ocorrerão ao longo do tempo de criação e uso de um software: durante o desenvolvimento, na fase da entrega e depois de entregue. Sempre há necessidade de ajustes e correções ou ainda pode ocorrer a necessidade de incluir novas funcionalidades ao software, as quais são, muitas vezes, requisitadas pelo cliente. Além disso, o software passa por uma série de manutenções (isso acontece pois são realizadas novas solicitações do cliente) e, após realizar diversos ajustes (que podem levar a novos problemas), é possível que haja a necessidade de se criar um novo software.

Pressman (2016) destaca que, durante a vida útil de um determinado software, ele irá sofrer várias alterações, as quais podem gerar novos erros, fazendo com que a curva da taxa de defeito aumente. Esse efeito pode ser observado na Figura 1.1.

Figura 1.1 | Curva de defeitos de um software ao longo do tempo



Fonte: Pressman (2016, p. 6).

A Figura 1.1 ilustra a curva de defeitos de software (taxa de defeitos) ao longo do tempo. No gráfico, pode-se visualizar uma curva idealizada, na qual a taxa de defeitos é alta no início do desenvolvimento do software, mas que decresce com o tempo e, em seguida, estabiliza, tornando-se constante. Na prática, esse comportamento idealizado não ocorre, pois muitas vezes são necessários acréscimos de funcionalidades (a pedido do cliente) ou até mudanças no software para corrigi-lo de algum problema identificado. Nesse sentido, é apresentada a curva real, que mostra que, à medida que mudanças são introduzidas em um software, ele torna-se vulnerável, uma vez que tais mudanças podem ocasionar erros e/ou defeitos no software, como mostram

os picos no gráfico, que representam o aumento das taxas de defeitos. Muitas vezes as melhorias devem ser realizadas rapidamente e o processo da documentação é deixado de lado (para acelerar as mudanças). Todo o processo de mudança pode acarretar em um software mais frágil (com possibilidades de surgimento de mais erros).

Exemplificando

Qual a vida útil de um software? Se pensarmos na aplicabilidade de um produto, podemos dizer que ele pode se tornar obsoleto com o passar do tempo. Em se tratando de um software, sua perspectiva de usabilidade pode ser considerada longa, porém pode se tornar ultrapassado à medida que as mudanças não atendam mais as expectativas do usuário.

Pressman (2016) destaca que o software não se desgasta como ocorre com o hardware, o qual, por ser um componente físico, é suscetível e está exposto a fatores ambientais, tais como, altas temperaturas, poeira e trepidações. Todavia, softwares podem deteriorar-se com as mudanças implantadas. Importa destacar que a taxa de defeitos no hardware, além de fatores ambientais, pode ser acarretada devido a picos de processamento do software, uma vez que os recursos de hardware (memória, processador e placa de vídeo) podem não suportar a execução de um software. É o que ocorre, por exemplo, com jogos que requerem uma placa de vídeo de alto poder computacional; eles podem “travar” devido às altas temperaturas do processador e da placa de vídeo.

Sommerville (2011) destaca que as Leis de Lehman podem ser utilizadas para verificar a dinâmica da evolução dos softwares. O estudo foi realizado na década de 1980, entretanto ainda é aplicável nos dias atuais, pois as leis são genéricas e podem ser utilizadas em diferentes softwares que possuam os seguintes processos: tomada de decisão, planejamento, desenvolvimento e manutenção. As Leis de Lehman são, conforme citadas em Sommerville (2011):

- **Mudança contínua:** o software deve ser ininterruptamente adaptado, senão torna-se, aos poucos, cada vez menos eficaz.
- **Aumento da complexidade:** a cada mudança o software torna-se mais complexo. Deve-se tomar medidas para reduzir a complexidade de um software, mantendo a sua simplicidade.
- **Evolução (autorregulação):** o processo de evolução de um software é autoajustável e os atributos do sistema quase não mudam entre as versões.

- **Estabilidade organizacional:** a evolução do software tende a ser constante na sua taxa de desenvolvimento, independentemente dos recursos utilizados.
- **Conservação da familiaridade:** durante o tempo de uso do software em evolução, a taxa de mudanças tende a ser proporcional ao domínio que a equipe adquire, tornando-se constante.
- **Crescimento contínuo:** as funcionalidades do software tendem a aumentar conforme novas versões são desenvolvidas e entregues.
- **Declínio da qualidade:** caso não haja um processo de evolução do software, a qualidade deste cairá conforme os sistemas ficam desatualizados.
- **Sistema de *feedback*:** os processos de manutenção agregam sistemas de *feedback* em múltiplos níveis e *loops* para obter melhorias nos processos e no produto final.

Prática da engenharia de software

De acordo com Sommerville (2011), a prática da engenharia de software se faz necessária por dois fatores:

- Cada vez mais a sociedade (e os indivíduos que a compõem) estão dependentes dos sistemas de software e, neste contexto, devemos produzir softwares mais confiáveis e de forma mais econômica.
- A longo prazo é mais barato utilizar as técnicas da engenharia de software para evitar mudanças em algo que já tenha sido desenvolvido e principalmente que partes do software possam ser reutilizadas em outros sistemas.

Conforme Pressman (2016), a engenharia de software é uma tecnologia em quatro camadas que objetiva a disciplina, a adaptabilidade e a agilidade. As quatro camadas são:

- **Foco na qualidade:** é o objetivo final de toda a engenharia de software; toda a gestão de qualidade (das demais camadas) deve ser fundamentada em um comprometimento organizacional com a qualidade total de todas as etapas de desenvolvimento.
- **Processo:** é a base da engenharia de software; o processo é a ligação entre as demais camadas; é o que mantém as camadas de tecnologia coesas, possibilitando o desenvolvimento de forma racional (e dentro dos prazos preestabelecidos) e é a base para o controle e gerenciamento de projetos de software. É nesta camada que são produzidos

diversos artefatos (modelos, documentos, relatórios, formulários, etc.).

- **Métodos:** fornecem as informações técnicas para o desenvolvimento do software e envolvem uma grande quantidade de tarefas (comunicação, análise de requisitos, modelagem de projetos, codificação, testes e manutenção do software).
- **Ferramentas:** possibilitam um alicerce automatizado ou semi-automatizado para o processo e para os métodos; quando as ferramentas são integradas de modo que a informação criada possa ser usada por outra ferramenta, é estabelecido um suporte ao desenvolvimento de software, conhecido como engenharia de software auxiliado por computador.

A engenharia de software, segundo Sommerville (2011), preocupa-se com todos os aspectos de produção do software. Enquanto isso Pressman (2016) enfatiza que essa área engloba processos, métodos e ferramentas que possibilitam a construção de sistemas, incorporando cinco atividades específicas nesses processos: comunicação, planejamentos, modelagem, construção e entrega.

Pressman (2016) destaca sete grandes categorias de softwares e desafia a constante evolução das práticas empregadas na engenharia de software:

- **Software de sistema:** são conjuntos de programas com finalidade específica para auxiliar outros programas, por exemplo: compiladores, drivers, etc.
- **Software de aplicação:** programas independentes que têm a finalidade de resolver um problema específico de negócio, facilitando transações comerciais, decisões administrativas, e que podem ser: planilhas de cálculos, editores de textos, software exclusivo da empresa (criado e desenvolvido para a empresa).
- **Software de engenharia/científico:** programas que facilitam o entendimento e usam grandes quantidades de cálculos, geralmente utilizados em aplicações para meteorologia, astronomia, genética, etc.
- **Software embarcado:** programado de forma que só funciona para determinado produto (forno micro-ondas, geladeira, módulo de carros).
- **Software para linha de produtos:** projetado para ser utilizado por diversos clientes (com ou sem adaptações), concentrando-se em nichos de mercado, por exemplo: software contábil, software de controle de RH, etc.

- **Software de aplicações web/ aplicativos móveis:** foco em dispositivos com acesso à internet e voltados a navegadores e softwares residentes em dispositivos móveis: TV, celulares, tablets, etc.
- **Software de Inteligência Artificial:** utiliza algoritmos sofisticados para analisar e solucionar problemas complexos, inclui as seguintes áreas: robótica, reconhecimento de padrões (voz, imagens, sons), redes neurais, etc.

Um tipo de software que frequentemente é encontrado em várias instituições são os softwares legados, os quais, segundo Pressman (2016), são softwares antigos que continuam sendo utilizados, de modo que apresentam baixa qualidade, muita demora e funcionalidades que não são mais utilizadas pela empresa ou que são muito defasadas. Um software legado pode ter uma documentação deficitária ou inexistente, fazendo com que o processo de manutenção seja caro e muito demorado. O processo de evolução desse tipo de software é a criação de um novo com tecnologias mais recentes.

Assimile

A engenharia de software deve permitir que o desenvolvimento de um software seja eficiente, levando em consideração custo, qualidade e tempo de desenvolvimento. Pressman (2016) afirma que os componentes reutilizáveis devem ser projetados, adaptados e integrados em novos sistemas.

Princípios da análise de sistemas

Os princípios da análise de sistemas fundamentam-se na necessidade de realizar estudos de processos para encontrar a melhor solução para a criação de um sistema. Conforme Roth, Dennis e Wixom (2014), a análise de sistemas baseia-se em métodos e técnicas de investigação e em especificação para encontrar a melhor solução para algum problema ou necessidade computacional de determinada área de negócio, a partir das funcionalidades levantadas pelo analista de sistemas.

Em uma visão mais generalizada das fases que envolvem os processos da análise de sistema, destacam-se, conforme Pressman (2005):

- **Análise:** nesta fase são realizados estudos que objetivam a especificação do software, de modo a verificar a viabilidade (custo-benefício), definir as funcionalidades que o software deverá possuir e realizar o escopo, alocando recursos e realizando o orçamento do software. O resultado desta fase será utilizado nas próximas etapas.

- **Projeto:** nesta etapa há uma preocupação com a definição lógica do software, são elaborados os *layouts* de telas e relatórios e são criados a estrutura de banco de dados e os diagramas gráficos para o desenvolvimento do software.
- **Implementação:** nesta fase é realizada a codificação do software por meio de uma linguagem de programação (definida na fase de análise).
- **Testes:** objetivando a procura de erros, nesta fase, são realizados procedimentos de testes que verificam as funcionalidades dos itens codificados.
- **Documentação:** trata-se de documentar todos os processos (de todas as fases) e diagramas produzidos; são utilizados documentos padronizados (e personalizados por cada empresa de desenvolvimento), que servem como ferramenta de comunicação entre as pessoas envolvidas no desenvolvimento e também como parte de contrato entre as partes interessadas na produção do software.
- **Manutenção:** esta fase consiste em fazer o acompanhamento do software após ser implantado e entrar em funcionamento (durante um período), visando a registrar e corrigir falhas, propor melhorias ou incluir novas funcionalidades.

As fases apontadas por Pressman (2005), devem apresentar um processo de homologação (um aceite oficial do cliente) para validar os documentos gerados. O cliente deve estar ciente de que uma vez aprovada uma fase, caso seja alterada, haverá mudanças nos custos e implicará no dimensionamento do tempo (atrasos).

Engholm Jr. (2010) afirma que as empresas de desenvolvimento podem enfrentar diversos problemas ao não adotarem as técnicas da engenharia de software na análise de sistemas. Alguns deles são:

- Softwares com manutenção muito onerosa e demorada.
- Falta de padronização na documentação.
- Falta de controle em determinar a sequência do que deve ser realizado.
- Previsões imprecisas a respeito do prazo de entrega e do preço final do software desenvolvido.

Sommerville (2011) e Pressmann (2016) destacam alguns princípios da análise de sistemas, são eles:

- O domínio da informação de um determinado problema deve ser representado e entendido por todas as partes envolvidas.

- As funcionalidades de um software precisam ser definidas e descritas de forma genérica (inicialmente) até a forma mais genérica.
- O comportamento do software deve ser representado através das interações com o ambiente externo, ou seja, com usuários e/ou outros sistemas.
- Os diagramas que demonstram as funções e comportamentos do software devem ser divididos para revelar detalhes em forma de camadas, de modo que se decompõe um problema complexo em partes menores para facilitar sua compreensão.
- A tarefa de análise deve ir da informação essencial até os detalhes de implementação, sem a preocupação de como será realizada a codificação da solução; os detalhes sobre a implementação determinam como a solução será realizada.

O papel do analista de sistemas

O analista de sistemas possui um papel fundamental nos processos da engenharia de software conforme afirma Sommerville (2011). Esse profissional é o responsável por realizar atividades da análise de sistema como: pesquisas, planejamentos, coordenação de equipes de desenvolvimento e recomendação de alternativas de software de acordo com as necessidades de desenvolvimento ou de solução para problemas de negócios. O analista de sistemas possui como tarefas a criação, a implementação e a implantação de um software, de maneira que deve, primeiro, descobrir o que um sistema deverá fazer e depois entender e avaliar as necessidades e expectativas de cada usuário do software, a fim de que estas sejam organizadas, especificadas e documentadas.

Para Elmasri e Navathe (2011), o analista de sistema desenvolve as especificações, as funcionalidades e as transações customizando as soluções para atender as solicitações dos usuários. E, neste momento, o profissional desse ramo precisa conhecer um pouco de cada área de negócio e, caso não tenha o domínio necessário sobre algum tema, deve ter a proatividade de procurar o máximo de conhecimento sobre a área que o software irá abranger.

Uma característica importante do analista de sistemas é ter uma boa visão empresarial para ajudar nos processos gerenciais da produção do software. As habilidades desejáveis para um analista de sistema são: conhecimento tecnológico atualizado, organização e método, visão gerencial, ótimo relacionamento interpessoal, entre outras.

O analista de sistemas é a “ponte” entre os programadores e os usuários finais do software, sendo ele o responsável por interpretar os anseios dos

usuários e por saber o que é ou não viável para ser desenvolvido. Cabe ao analista de sistemas colher informações com os usuários que utilizarão o software, interpretar essas informações e repassá-las aos programadores de forma técnica para que seja criado um software que atenda as expectativas do cliente e dos usuários.

Durante o desenvolvimento de um software, o analista de sistemas possui as seguintes atribuições:

- Interagir com clientes e usuários finais do software.
- Analisar custos e verificar a viabilidade do projeto.
- Fazer o levantamento das informações através de entrevistas com usuários do software.
- Levantar os dados e os requisitos do software para analisar e propor soluções.
- Criar a modelagem do software.
- Orientar os programadores, acompanhando todo o desenvolvimento do software (tanto na parte lógica quanto na parte de interface gráfica).
- Acompanhar e executar testes.
- Preparar e acompanhar toda a documentação do software.
- Gerenciar eventuais mudanças no projeto.
- Determinar padrões de desenvolvimento.
- Garantir a qualidade final do software e que este esteja de acordo com o solicitado pelo cliente.
- Realizar o monitoramento e fazer auditorias, procurando eventuais falhas.
- Planejar e aplicar treinamentos para a utilização do software desenvolvido.
- Implantar o software desenvolvido, acompanhando o processo de adaptação e integração dos sistemas do cliente.
- Proporcionar consultoria técnica para identificar as necessidades dos clientes nas mais diversas áreas de negócio.
- Pesquisar novas tecnologias, fornecedores e, se for o caso, buscar por especializações para si e para a equipe de desenvolvimento.

Refita

Existem diversos cargos na área de TI, sendo comum a progressão deles dentro de uma empresa. Mas, será que um analista de sistemas precisa ser um ótimo programador? E um programador pode se tornar um analista de sistemas?

Nesta seção foi mostrado que um software precisa evoluir para não ficar obsoleto e foi apresentada a necessidade de incluir práticas de engenharia de software e princípios da análise de sistemas no desenvolvimento de um software. Além disso, vimos o papel do analista de sistemas em todo esse processo. Ser um analista de sistemas é ter uma carreira desafiadora, pois a cada novo projeto de Software há novas oportunidades de adquirir conhecimentos e de aperfeiçoar suas habilidades em solucionar problemas, gerenciar e coordenar equipes. Convido você a continuar seus estudos com os próximos tópicos desta unidade e a se aprofundar no fascinante mundo da criação de softwares.

Sem medo de errar

Você está trabalhando em uma *software house* e, como primeira missão nessa empresa, você precisa avaliar um software de exames laboratoriais a fim de recomendar novas tecnologias para ele, as quais, caso aprovadas, serão utilizadas. O software em questão é de uma empresa com diversas filiais no Sul do país que há 15 anos utiliza o mesmo software desktop (sem acesso à internet).

Vamos começar nossa análise avaliando se é mais viável atualizar o sistema existente ou criar um novo. O atual software que a empresa utiliza é limitado ao ambiente desktop, ou seja, não é um sistema projetado para funcionar com os recursos da internet. Esse é um importante ponto, pois, se a empresa possui filiais, é crucial que ela tenha um sistema central, o qual as filiais accessem via internet, mantendo todo o sistema integrado. Veja que, mesmo sem avaliar as funcionalidades desse sistema legado, já sabemos que não é viável usá-lo. Dessa forma, teremos que propor um novo sistema que suporte transações on-line.

Pois bem, sabendo que precisamos de um novo sistema, quais tecnologias podem ser utilizadas?

Para determinar as novas tecnologias a serem utilizadas em um novo sistema, deverá ser feito um trabalho de investigação para o qual recomenda-se os seguintes passos:

1. Visitar uma unidade da empresa para acompanhar o funcionamento do sistema.
2. Acompanhar o cadastramento da coleta de exames de um paciente a fim de observar o tempo gasto.
3. Verificar como é realizada a entrega dos resultados.
4. Descobrir qual a linguagem de programação utilizada e como é o funcionamento do banco de dados.

Após o trabalho investigativo, algumas tecnologias já podem ser apontadas:

- Criação de um site e de um aplicativo que permitam:
 - Marcar o exame.
 - Agendar a coleta em casa (caso o cliente assim deseje).
 - Acompanhar o andamento da análise laboratorial dos exames solicitados.
 - Visualizar os resultados dos exames.
 - Disponibilizar os resultados para que sejam impressos pelo paciente.
- Armazenamento do banco de dados em nuvem. Nesse caso, você deverá consultar os três grandes *players* (Amazon, Google e Microsoft) e fazer um levantamento de custos.
- Utilização da linguagem JAVA como sugestão de linguagem de programação para diminuir os custos para o cliente.

E qual deverá ser o perfil dos profissionais envolvidos no processo da análise do sistema do cliente? O ideal é ter, na equipe de desenvolvimento, analistas de sistemas com as seguintes habilidades: conhecimento tecnológico atualizado, organização e método, visão gerencial e ótimo relacionamento interpessoal. Um dos papéis do analista de sistemas é a atualização tecnológica constante, fator importante justamente nos casos em que um cliente pede ajuda para atualizar seus sistemas.

E como saber quais os processos que deverão ser adotados para manter o Software sempre atualizado? Imagine o seguinte cenário: após a entrega do software, foi verificado que nele poderiam ser utilizados códigos de barras e QR-Code. Na sua visão, o que precisa ser feito?

Acrescente itens à lista, crie uma apresentação com as soluções encontradas e as exponha aos seus colegas para gerar um debate sobre suas ideias.

Avançando na prática

Em busca de novas oportunidades de desenvolvimento

Um analista de sistemas sempre deve estar atento às novas oportunidades de desenvolvimento de novos sistemas. Muitas vezes uma empresa está “acomodada” com seus processos e é aí que esse profissional deve visualizar oportunidades de negócios para, assim, começar a empreender. Pesquise novas oportunidades de negócios; analise sites de livrarias, confeitarias, academias de ginástica, etc. Quais as vantagens que essas empresas podem ganhar com o desenvolvimento de um aplicativo? Crie uma lista de vantagens que a empresa pode ter com um aplicativo exclusivo.

Resolução da situação-problema

Para resolver essa questão, o primeiro passo é procurar por sites de empresas que podem virar aplicativos. Algumas das vantagens que ela pode ter com um aplicativo exclusivo podem ser:

- Maior visibilidade.
- Canal direto de vendas.
- Canal de relacionamento com o cliente.
- Valorização da marca com um aplicativo personalizado.
- Fidelização de seus clientes.

Agora é com você: acrescente mais vantagens à lista.

Pesquise clientes em potencial, deixe reservada uma relação de futuros clientes para que você possa oferecer consultorias como analista de sistemas.

Faça valer a pena

1. O analista de sistemas é responsável por pesquisar, desenvolver e melhorar vários tipos de softwares. Seu trabalho é analisar soluções sobre um determinado software e projetá-lo de acordo com as expectativas do cliente. Os projetos desenvolvidos podem ser: jogos eletrônicos, sites, aplicativos para celular, sistemas industriais, etc.

Analise as afirmativas a seguir sobre o papel do analista de sistemas em uma empresa de desenvolvimento de sistemas.

- I. Coordena e orienta os programadores, acompanhando todo o desenvolvimento do software.
- II. Gerencia eventuais mudanças no projeto, acompanhando os testes para validar as funcionalidades do software.
- III. Estabelece padrões de desenvolvimento e cria os projetos lógicos do software.
- IV. Coordena a implantação do software desenvolvido, acompanhando o processo de adaptação e integração dos sistemas do cliente.

Analise as alternativas a seguir e assinale a que apresenta as afirmativas corretas sobre o papel do analista de sistemas em uma empresa de desenvolvimento de sistemas.

- a. Apenas as afirmativas I e II estão corretas.
- b. Apenas as afirmativas I e III estão corretas.
- c. Apenas as afirmativas II, III e IV estão corretas.
- d. Apenas as afirmativas I, II e III estão corretas.
- e. As afirmativas I, II, III e IV estão corretas.

2. A tarefa de desenvolver softwares é complexa, exigindo métodos e técnicas para investigar, levantar e solucionar problemas dos usuários. Como são muitas atividades, a análise de sistemas ajuda com processos específicos (fases de desenvolvimento) para tornar as atividades mais gerenciáveis para o analista de sistemas.

Analise as afirmativas a seguir sobre das fases que envolvem os processos da análise de sistema.

- I. Projeto: nesta etapa há uma preocupação com a definição lógica do software; são elaborados os *layouts* de telas e relatórios; há a criação da estrutura de banco de dados e dos diagramas gráficos para o desenvolvimento do software.
- II. Análise: nesta fase são realizados estudos que objetivam a especificação do software, verificando a viabilidade (custo-benefício) e definindo as funcionalidades que o software deverá possuir.
- III. Documentação: trata-se de documentar todos os processos (todas as fases); serve como ferramenta de comunicação entre as pessoas

envolvidas no desenvolvimento e também é utilizada como parte de contrato entre as partes interessadas na produção do software.

- IV. Manutenção: nesta fase são realizados a codificação e os testes no código fonte do software, utilizando técnicas de refinamento e buscas unitárias para estabelecer a precisão lógica dos componentes do software.
- V. Implementação: esta fase consiste em fazer o acompanhamento do software após este ser implantado e entrar em funcionamento (durante um período), visando a registrar e corrigir falhas, propor melhorias ou incluir novas funcionalidades.

Analise as alternativas a seguir e assinale a que apresenta as afirmativas corretas sobre o papel do analista de sistema em uma empresa de desenvolvimento de sistemas.

- a. Apenas as afirmativas I, II, IV e V estão corretas.
- b. Apenas as afirmativas I, II e III estão corretas.
- c. Apenas as afirmativas II, III e V estão corretas.
- d. Apenas as afirmativas I, II e IV estão corretas.
- e. As afirmativas I, II, III, IV e V estão corretas.

3. Pressmann (2016) enfatiza que um software legado é um software antigo que continua sendo utilizado, porém apresenta baixa qualidade, demora e funcionalidades que não são mais utilizadas pela empresa ou que estão muito defasadas.

Assinale a alternativa correta sobre um software legado.

- a. Um software legado pode ter o seu código fonte totalmente reaproveitado, facilitando o processo de atualização para um software novo.
- b. Um software legado pode ter o seu código fonte parcialmente reaproveitado, facilitando o processo de atualização para um software novo.
- c. Um software legado possui a vantagem de ser atualizado com custos relativamente baixos em vista de um software novo.
- d. Um software legado deve ser mantido na empresa mesmo após a implantação de um novo software, pois não é possível um novo software substituir um software legado.
- e. Um software legado pode ter uma documentação deficitária ou inexistente, fazendo com que o processo de manutenção seja caro e muito demorado.

Seção 2

O processo de software

Diálogo aberto

Olá, seja bem-vindo!

O cotidiano de um Analista de Sistemas é bem agitado, são muitas tarefas que devem ser executadas: visita ao cliente, orientação aos programadores, pesquisas e, claro, muito planejamento. A organização do tempo deverá ser um exercício constante na vida profissional do Analista de Sistemas, caso contrário muitas tarefas ficarão esquecidas e, mais tarde (durante o desenvolvimento do software), esse esquecimento pode gerar uma série de prejuízos. Pensando neste aspecto, você receberá um desafio.

Como Analista de Sistemas em uma Software House, você realiza diversas tarefas, atendendo diferentes clientes. Alguns clientes têm relatado um incômodo, quanto ao tempo de entrega do produto final, o software. Para ajudar a empresa, você possui a missão de investigar o Processo de software da empresa e propor melhorias. Para isso, deverá se concentrar em responder os seguintes questionamentos: Quais as atividades principais de um Processo de Software? Como você poderia melhorar o Processo de construção de um software?

Para ajudar, nesta seção estaremos abordando as atividades do Processo de Software, que é uma série de atividades que estão relacionadas, tendo como resultado um produto (o software). Estaremos apresentando a Introdução ao Processo de Software, a Estrutura de um Processo Genérico de Software, as Tarefas e Modelagem das Atividades do Processo de Software e a

Integração e Validação entre as Atividades do Processo de Software.

Boa aula!

Não pode faltar

Um Processo de Software, conforme destaca Sommerville (2011), é um conjunto de atividades e resultados que estão relacionados, que levam à produção e ao resultado de um software desenvolvido. Pressman (2016) enfatiza que na Engenharia de Software um processo não é uma determinação rigorosa sobre como ele deve ser desenvolvido, ao contrário, é uma abordagem adaptável que possibilita à equipe de desenvolvimento escolher os processos que melhor se enquadram na filosofia da empresa (de

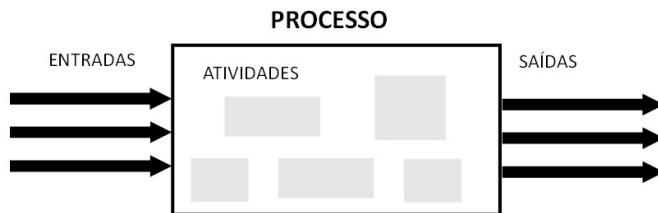
desenvolvimento) com o foco na qualidade do produto, no prazo de entrega e na redução de custos.

Sommerville (2011) afirma que a abordagem sistemática usada pela Engenharia de Software para produção de software é chamada Processo de Software. Um Processo de Software pode conter diversas atividades que normalmente são: especificação, projeto, implementação, validação, manutenção e evolução. Engholm Jr. (2010) relaciona os seguintes aspectos sobre o Processo de Software:

- Cria uma padronização na forma de gerar os serviços e produtos.
- Permite que sejam repetidos os serviços e produtos, reutilizando partes já produzidas e padronizadas.
- Retém o conhecimento na empresa, permitido que novos integrantes possam dar continuidade aos processos definidos previamente.
- Serve para definir e guiar as atividades de um Projeto de Software.
- Permite a especificação de todo o processo para o desenvolvimento do software (ou partes dele).
- Determina as tarefas que deverão ser executadas para a equipe e de forma individual.
- Reduz riscos, tornando os resultados mais previsíveis.
- Proporciona visões comuns para a equipe de desenvolvimento, facilitando a comunicação.
- Pode ser utilizado como um *template*, sendo utilizado em outros projetos, permitindo agilidade em novos projetos de software.

Na definição dos Processos de Software, segundo Engholm Jr. (2010), são definidas uma série de parâmetros: (i) evento que determina o início do processo; (ii) matriz de responsabilidades; (iii) atividades que serão executadas e suas sequencialidades; (iv) entradas e saídas de cada atividade; (v) regras e políticas a serem aplicadas na realização das atividades; (vi) infra-estrutura necessária; e (vii) resultado gerado na execução de cada processo. A Figura 1.2 demonstra graficamente que, em cada Processo de Software, podem ocorrer diversas atividades (sequenciais ou em paralelo).

Figura 1.2 | Representação de um Processo de Software

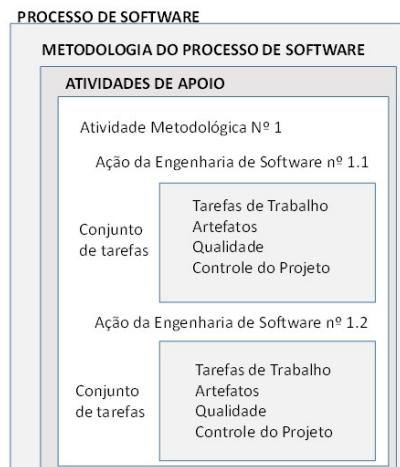


Fonte: adaptada de Engholm Jr. (2010, p.43).

Conforme a Figura 1.2, um Processo de Software possui inúmeras entradas e saídas. O processo se constitui em uma série de atividades que serão executadas de forma padronizada, agrupadas em fases (essas atividades mudam conforme há a troca de fase), sendo que, em cada fase, serão definidos: as responsabilidades (quem fará o quê), prazos de entrega e como o objetivo será alcançado. A Figura 1.3 ilustra um esquema para o processo de software. Podemos observar que:

cada atividade metodológica é composta por um conjunto de ações de engenharia de software. Cada ação é definida por um conjunto de tarefas, o qual identifica as tarefas de trabalho a ser completadas, os artefatos de software que serão produzidos, os fatores de garantia da qualidade que serão exigidos e os marcos utilizados para indicar progresso. (PRESSMAN, p. 31, 2016)

Figura 1.3 | Exemplo de Metodologia de Processo de Software



Fonte: adaptada de Pressman (2016, p.32).

Assimile

Um problema que atinge o desenvolvimento de um software é a troca de pessoal que naturalmente ocorre em qualquer situação. Engholm Jr. (2010) afirma que, estando os Processos de Software bem definidos, os conhecimentos produzidos em cada etapa (do Processo) estarão preservados, garantindo a sua continuidade.

O Processo de Software adotado em uma empresa pode ser completamente diferente de outra empresa, cada qual procura encontrar e estabelecer atividades que visam aumentar a qualidade e baixar o custo de produção do software produzido. Independente do modelo de Processos de Software adotado pela empresa de desenvolvimento, todos utilizam uma Estrutura de Processo Genérico de Software, com atividades pré-estabelecidas.

As atividades de um determinado Processo de Software constituem um conjunto mínimo para se obter um produto de software (o software finalizado e entregue ao cliente). Em um Processo Genérico de Software, os processos podem ser diferentes, mas podemos identificar quatro atividades fundamentais em toda a produção de software, conforme Sommerville (2011):

- **Especificação de software:** definição do que será desenvolvido, suas restrições e funcionalidades.
- **Projeto e Implementação de software:** projeto e desenvolvimento (programação) do software atendendo às especificações.
- **Validação de software:** verificação se o que foi construído atende as solicitações do cliente.
- **Evolução de software:** evolução do software para que acompanhe as alterações solicitadas pelo cliente.

Cada atividade do Processo Genérico de Software é composta por um conjunto de atividades da Engenharia de Software. Pressman (2016) afirma que uma metodologia genérica da Engenharia de Software é composta de cinco atividades, que são:

- **Comunicação:** com a intenção de entender os objetivos do projeto, a comunicação entre os envolvidos é a primeira ação primordial, para entender os requisitos (as funcionalidades) do software a ser realizado.
- **Planejamento:** é realizado um “mapa”, um plano de projeto do software a ser realizado, descrevendo as tarefas técnicas, os riscos, os recursos, os produtos resultantes e um cronograma de trabalho (para acompanhar o desenvolvimento do software).

- **Modelagem:** são criados modelos (diagramas) para melhor entendimento das necessidades do software, os modelos são utilizados para realizar a codificação do software e para validação das partes envolvidas no projeto.
- **Construção:** realização da codificação (baseada nos modelos criados anteriormente), nesta fase também são realizados os testes para validar os códigos de programação gerados.
- **Entrega:** o software é entregue parcialmente ou na sua totalidade, onde o cliente realiza testes e fornece um *feedback*; nesta fase são realizadas adaptações e correções no software por um determinado período (acordado entre as partes).

Pressman (2016) destaca que as atividades metodológicas (citadas anteriormente) devem ter uma série de tarefas que darão suporte no acompanhamento e controle do projeto, controlando os riscos, fazendo revisões técnicas, etc.. Entretanto, a sequencialidade ou não de cada atividade genérica pode ser dividida de acordo com o **Fluxo de Processo**. Segundo Pressman (2016), esse fluxo é usado para descrever como as atividades metodológicas de cada Processo são organizadas. Os Fluxos de Processos podem ser:

- **Fluxo de Processo Linear:** caracteriza-se por realizar em forma sequencial as cinco atividades metodológicas apontadas por Pressman (2016), inicializando pela Comunicação e terminando na fase da Entrega.
- **Fluxo de Processo Interativo:** possui como característica a repetição de uma ou mais atividades antes de avançar para a próxima atividade.
- **Fluxo de Processo Evolucionário:** as atividades são executadas de modo circular, cada ciclo envolve as cinco atividades, gerando uma versão mais completa (é um processo incremental).
- **Fluxo de Processo Paralelo:** as atividades são realizadas de forma paralela, onde duas ou mais atividades podem ser executadas simultaneamente, por exemplo, a atividade de Comunicação pode ocorrer em paralelo com a atividade de Análise.

Reflita

Os Fluxos de Processo de um Software conduzem a execução do software. Será que uma empresa desenvolvedora de software pode usar Fluxos de Processos diferentes para realizar o mesmo tipo de software, mas com clientes distintos?

O desenvolvimento de um software requer muito planejamento e um software nunca é igual ao outro. Uma universidade pode solicitar um software de controle Acadêmico, caso outra universidade solicite o mesmo tipo de software, dificilmente (para não afirmar quase que impossível) os dois softwares serão iguais. Pressman (2016) afirma que projetos diferentes exigem conjuntos de Tarefas e Modelagem das Atividades do Processo de Software diferenciados. Os Analistas de Sistemas determinam o conjunto de tarefas baseados nos problemas e nas características do projeto que será executado.

Um conjunto de tarefas determina o que deverá ser feito para alcançar os objetivos de uma determinada ação, dentro do Processo de Software. Um exemplo é a codificação das telas de um Sistema, o programador recebe um conjunto de requisitos que as telas deverão possuir, após a conclusão da programação, o trabalho do programador será avaliado conforme as especificações que ele recebeu para fazer as telas do Sistema.

Exemplificando

Observe um conjunto de tarefas (atividades) na fase de Planejamento de um Software:

ATIVIDADES DE PLANEJAMENTO DE UM SOFTWARE		
FASE	ATIVIDADES	RESULTADOS
• Planejamento	<ul style="list-style-type: none">• Levantamento de Requisitos.• Especificação dos Requisitos.• Estimativas de Prazos.• Estimativa de Recursos.	<ul style="list-style-type: none">• Documentação do Levantamento de Requisitos.• Documentação da Especificação de Requisitos.• Plano de ação para determinar os prazos.• Alocação de Recursos para criação do software.

Conforme Sommerville (2011) o Modelo de Processo de Software é uma descrição simplificada do Processo que especifica as atividades para o desenvolvimento, define os produtos de cada atividade, determina os papéis dos envolvidos no desenvolvimento, oferecendo um roteiro para a Engenharia de Software.

A existência de um Processo de Software não garante a qualidade do software e muito menos que o software será entregue no prazo combinado, também não é certo que as funcionalidades do software estarão de acordo com o que o cliente solicitou. A qualidade do software produzido é diretamente influenciada pelos padrões de qualidade impostos durante os Processos de software (durante a produção do software) sendo necessário estabelecer

procedimentos e padrões para garantir a qualidade dos Processos. Pfleeger (2004) destaca que uma das vantagens de fazer a Modelagem dos Processos é a possibilidade de examinar e procurar meios de aprimoramento antes de finalizar o produto (o software produzido).

O Processo de Software deve ser avaliado para certificar que ele atenda a um conjunto de critérios básicos. O Processo de Software pode passar por uma série de critérios pré-estabelecidos que ajudam a garantir a Integração e Validação entre as Atividades do Processo de Software. Pressman (2016) destaca uma série de abordagens de avaliação e aperfeiçoamento dos Processos de Software: (i) SCAMPI, (ii) CBA IPI, (iii) SPICE e (iv) ISO 9001:2000 para software.

A abordagem SCAMPI (Método Padrão CMMI de Avaliação para Aperfeiçoamento de Processos da CMMI - *Standard CMMI Appraisal Method for Process Improvement*) fornece um modelo de avaliação do Processo em cinco etapas – início, diagnóstico, estabelecimento, atualização e aprendizado. Define regras para assegurar a objetividade na classificação das avaliações, e ainda:

- Ajuda a coletar e reunir evidências por meio de apresentações, documentos e entrevistas.
- Transforma em anotações todas as evidências do que foi observado.
- Converte as anotações em declarações de acertos ou falhas quando comparadas às práticas do CMMI.
- Converte as declarações em descobertas preliminares.
- Obtém a validação das descobertas preliminares e transforma as descobertas preliminares em definitivas.

Assimile

CMMI (Modelo de Capacidade e Maturidade Integrado - *Capability Maturity Model Integration*) é um conjunto de práticas que orientam a implementação de uma série de atividades com o objetivo de alcançar uma meta pré-estabelecida, aumentando o amadurecimento organizacional e auxiliando a obter os resultados esperados pela área de TI.

Existem três modelos diferentes do CMMI:

- *CMMI for Development*: apresenta melhores práticas para desenvolver produtos e serviços.
- *CMMI for Acquisition*: mostra as melhores práticas para adquirir produtos e serviços.
- *CMMI for Services*: indica as melhores práticas para entregar serviços.

Segundo Pressman (2016), a abordagem CBA IPI (CMM Based Appraisal for Internal Process Improvement - Avaliação para o Aperfeiçoamento do Processo Interno baseado na CMM (*Capability Maturity Model* - Modelo de Maturidade em Capacitação)), fornece uma técnica de diagnóstico para avaliar a maturidade relativa de uma organização de software. O método possui a capacidade de identificar pontos fortes e fracos dos processos e viabiliza a possibilidade de priorização das melhorias mais relevantes.

Pressman (2016) afirma que a abordagem SPICE (ISO/IEC15504) é um padrão que define um conjunto de requisitos para avaliação do Processo de Software, possui a finalidade de auxiliar as organizações no desenvolvimento de uma avaliação objetiva da eficácia de um processo de qualquer software. Este método fornece uma estrutura para a avaliação de Processos de Software e esta estrutura pode ser empregada nas organizações envolvidas na produção de um software.

A abordagem ISO 9001:2000 para software, conforme Pressman (2016), é um padrão genérico aplicável a qualquer organização que precise aplicar um padrão global de qualidade em seus produtos, sistemas ou serviços. Existem vários modelos de referências para ajudar a garantir a qualidade e que são aplicáveis a um software, alguns destes modelos são: ISO/IEC 9126, a ISO9000, a ISO9001, a ISO/IEC12207. A ISO9001 descreve um modelo de garantia de qualidade em projetos, instalações, desenvolvimento e assistência técnica. Esta norma pode ser aplicada especificamente na área de desenvolvimento, fornecimento e manutenção de software por meio dos roteiros descritos na norma ISO 9000-3. A ISO/IEC 9126 descreve as características de um software de qualidade.

Os erros que ocorrem durante o Processo de Software podem ser controlados utilizando uma abordagem metodológica. O Analista de Sistemas deve estar atento ao surgimento de novas metodologias, testá-las e, se forem apropriadas, utilizar durante o Processo de Software. O objetivo é criar um software com qualidade com o mínimo de erros e com a aprovação do cliente. Levando em consideração os preceitos apresentados nesta seção, convido você a conhecer os Modelos de Processos de Software que serão apresentados na próxima seção, dando continuidade ao Processo de Software.

Sem medo de errar

Chegou o momento de resolver o desafio que lhe foi proposto. O desenvolvimento de um software requer a mobilização de diversos profissionais, cada um contribuindo com sua função. Você, como Analista de Sistemas em uma Software House, recebeu como missão rever o processo de Software da

empresa e propor melhorias. Podemos começar essa investigação, fazendo um levantamento das principais atividades de um Processo de Software.

Geralmente as atividades de um Processo de Software estão divididas em: Especificação (Análise), Projeto, Implementação, Validação, Manutenção e Evolução.

Para exemplificar as atividades de um Processo de Software, veja algumas atividades:

- Na Especificação: são realizados o estudo de Viabilidade, a Elicitação de Requisitos, a Especificação de Requisitos e a Validação dos Requisitos.
- Projeto: são definidas as estruturas modulares do software, as *interfaces* gráficas e as estruturas de dados (do banco de dados).
- Implementação: tudo o que foi decidido nas atividades de Especificação e Projeto é passado para uma linguagem de programação e o banco de dados é criado.
- Validação: São realizados testes para validar tanto os códigos dos programas quanto a verificação dos requisitos (se o software atendeu aos requisitos impostos pelo cliente).
- Manutenção e Evolução: são atividades contínuas a fim de melhoria-mento do software e inclusão de novos recursos.

É comum às atividades de Especificação que envolvem todo o processo de busca de informação sobre o software a ser construído funcionar em paralelo às atividades de Projeto (assim que as funcionalidades do software forem levantadas, já é realizado o Projeto do Software), e como não podemos deixar os programadores desocupados, as atividades da Implementação (codificando o software) caminham em paralelo às atividades de Projeto (desde que aprovadas).

Agora que já sabemos quais as principais atividades, podemos pensar em como melhorar o processo de construção de um software. Para isso, pensando na qualidade final dos Processos de Software, é necessário estabelecer uma série de critérios de validação das atividades do Processo de Software, implantando um (ou mais) métodos de abordagem de avaliação e melhoria dos processos, podendo ser: SCAMPI, CBA IPI, SPICE e ISO 9001:2000.

Entretanto, você deverá fazer algumas pesquisas para ajudar a sua empresa. Pesquise na Internet:

1. Normas específicas da ISO 9001:2000. Faça um relatório indicando quais itens dessa norma podem ser adotados no Processo de Software.
2. Novas metodologias ágeis que podem acelerar o desenvolvimento.

Crie uma apresentação em slide para que possa mostrar os resultados para seus colegas.

Lembre-se que a pesquisa é um fator importante na carreira do Analista de Sistemas.

Bom trabalho!

Faça valer a pena

1. Pressman (2016) afirma que um Processo de desenvolvimento de software é um conjunto de atividades, ordenadas (de forma parcial), com a intenção de obter um produto de software. É considerado um dos principais mecanismos para se obter software de qualidade e cumprir os contratos de desenvolvimento firmados com o cliente.

Sobre os Processos de Software, analise as afirmativas a seguir.

- I. Os Processos de Software permitem que sejam repetidos os serviços e produtos, reutilizando partes já produzidas e padronizadas.
- II. Os Processos de Software servem para definir e guiar as atividades de um Projeto de Software.
- III. Os Processos de Software proporcionam visões comuns para a equipe de desenvolvimento, facilitando a comunicação.
- IV. Os Processos de Software determinam as tarefas que deverão ser executadas pelas equipes e de forma individualizada.

Analizando as afirmativas apresentadas, é correto o que se afirma em:

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

2. Sommerville (2011) descreve que a utilização de um Processo de Software é apontada como um fator essencial para o sucesso de empresas de desenvolvimento de software, ajudando no gerenciamento dos Processos de desenvolvimento.

Um Processo de Software pode ser entendido como:

- a. Uma tarefa específica utilizada no desenvolvimento de um software.
- b. Um conjunto estruturado de atividades exigidas para desenvolver um software.
- c. Uma metodologia de controle de agilidade, utilizada no desenvolvimento do software.
- d. Um diagrama utilizado para especificar o Fluxo de Processos no desenvolvimento do software.
- e. Um padrão de codificação, amplamente utilizado para reutilizar o código do software.

3. Pressman (2016) afirma que a Comunicação é um item fundamental no Processo de Software. Outros autores levam em consideração que a Comunicação está implícita em todos os Processos de desenvolvimento, pois existe a necessidade evidente de comunicação entre as partes envolvidas, começando no levantamento dos requisitos até a entrega do produto final.

Assinale a alternativa correta que apresenta as principais atividades que normalmente um Processo de Software possui.

- a. Especificação, implementação, validação, manutenção e evolução.
- b. Projeto, implementação, codificação, validação, manutenção e evolução.
- c. Implementação, validação, projeto, construção, manutenção e evolução.
- d. Especificação, projeto, implementação, validação, manutenção e evolução.
- e. Evolução, projeto, implementação, codificação, validação, manutenção e evolução.

Seção 3

Modelos de processos de software

Diálogo aberto

Olá, seja bem-vindo!

O uso de smartphones vem revolucionando a maneira das pessoas utilizarem Softwares. Existe APP para pagar contas, controlar gastos, localizar lugares e pessoas, e uma infinidade de outras utilidades. Nesse universo, dois grandes Sistemas Operacionais se destacam, o Android e o iOS. Você sabia que para desenvolver um Software (APP) nativo para esses dois sistemas é preciso utilizar diferentes linguagens de programação? Ou seja, para um mesmo APP nativo, são necessários dois projetos de desenvolvimento e duas equipes. Com tanto trabalho, como isso pode ser feito de maneira organizada, controlada e eficiente? Só há uma maneira! Usando um Modelo de Processo de Software que seja adequado ao projeto, à equipe e às expectativas do cliente.

Nesta seção serão apresentados vários Modelos de Processos de Software, disponíveis no mercado, sendo que alguns são bem antigos, mas, mesmo assim, são utilizados em vários projetos. Na verdade, tudo depende do tipo de Software a ser produzido, combinado com as expectativas do cliente. Todos os modelos possuem a finalidade de evitar o caos no desenvolvimento e estabelecer um Fluxo de Trabalho.

Você trabalha como Analista de Sistemas, em uma *software house* e recebeu um novo cliente, que deseja fazer um Software para sua loja de brinquedos *online*. O Software deverá permitir que os usuários comprem produtos da loja. Além do *site*, é necessário fazer um aplicativo, caso o usuário queira acessar a loja para comprar ou alugar brinquedos (o aluguel somente estará disponível na versão em aplicativo). Você deverá determinar: qual Modelo de Processo de Software será mais indicado para o mais novo projeto de Software?

Para realizar essa missão, teremos uma introdução ao conceito de Modelo de Processo de Software. Veremos os Modelos de Processos Prescritivos e os Modelos de Processos Especializados, além de uma introdução ao desenvolvimento ágil e seus principais métodos.

Se for necessário, pesquise mais sobre cada item, procurando imagens sobre cada Modelo de Processo.

Crie uma apresentação para compartilhar com seus colegas e trocar ideias sobre os Modelos de Processos de Software.

Boa jornada de estudos!

Não pode faltar

Um Processo de Software é constituído por várias atividades, cuja finalidade é ter como resultado um Produto de Software. Durante a fase de desenvolvimento de um Software existem muitas tarefas, que devem ser distribuídas entre os membros da equipe. Sommerville (2011) destaca que dentre as atividades do Processo de Software estão: o estudo da viabilidade, a análise dos requisitos, a especificação, a arquitetura de software, implementação (codificação), testes, documentação, suporte e treinamento e manutenção.

Um Processo de Software não precisa ter obrigatoriamente as atividades listadas por Sommerville (2011), além disso essas atividades podem ou não estar ordenadas, como afirma Pressman (2016). Existem atividades genéricas e que aparecem na maioria dos Processos de Software e que são apontadas por Sommerville (2011):

Análise e Especificação: são realizadas as definições sobre o Software (a ser produzido) e determinados seus requisitos (funcionalidades) e suas restrições.

Projeto: é realizada a alocação de recursos (Hardware e Software) e são identificadas e definidas as abstrações do funcionamento do Software.

Implementação e Teste Unitário: é todo o processo de codificação do Software (seu desenvolvimento realizado por Analistas e Programadores), é a fabricação do Software.

Integração e Verificação: é todo o processo de avaliação, correção e validação, considerando a qualidade final do Software.

Operação e Manutenção: nesta etapa são considerados todos os processos de alterações realizadas no Software (após ele estar em funcionamento).

Para o gerenciamento das atividades de Processo de Software são utilizados os **Modelos de Processos de Software**. Um Modelo de Processo de Software tem como objetivo propiciar estabilidade, controle e organização das atividades e é uma representação (geralmente gráfica) dos objetos e atividades envolvidas no Processo de Software. Pfleeger (2004) afirma que existem muitas técnicas e ferramentas para a Modelagem de Processos e não há um consenso em determinar o melhor modelo. Cada empresa adota seu Modelo de Processo de Software, realizando adaptações a cada Software produzido.

Pressman (2016) destaca que um Modelo de Processo de Software é um guia exclusivo para as atividades da Engenharia de Software, definindo um fluxo de todas as atividades, ações e tarefas, o nível de interação entre as atividades, os artefatos que serão produzidos e a organização do trabalho que deve ser realizado. Existem diversos Modelos de Processos de Softwares que possuem características diferentes. Destacam-se os seguintes: Modelos de Processos Prescritivos, Modelos de Processos Especializados e Modelos de Desenvolvimento Ágil.

Modelo de Processo Prescritivo

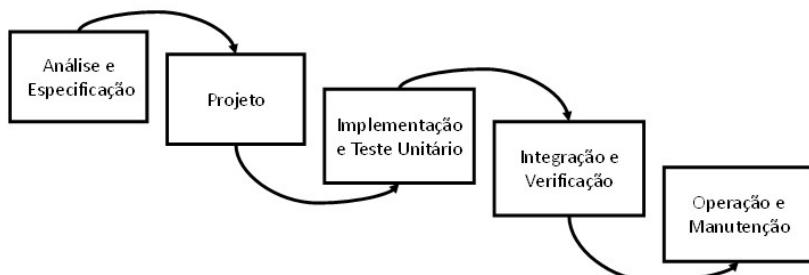
O Modelo de Processo Prescritivo (conhecido também como Modelos de Processos Tradicionais) consiste em um conjunto de elementos do processo, que podem ser ações de engenharia de software, produtos de trabalho e mecanismos que garantem a qualidade e o controle de mudanças nos projetos de desenvolvimento de um sistema de software (Pressman, 2016). Esse tipo de modelo prescreve os relacionamentos, ou seja, como os elementos dos processos são interligados, com a finalidade de estruturar e ordenar o desenvolvimento de um Software.

Conforme Pressman (2016), as tarefas ocorrem de forma sequencial, com diretrizes bem definidas, uma vez que indicam uma série de atividades metodológicas, ações, tarefas, artefatos, garantias de qualidade e mecanismos de controle de mudanças para cada projeto. Para cada Modelo de Processo Prescritivo também é indicado um Fluxo de Processo (ou Fluxo de Trabalho) ou seja, como esse conjunto de elementos do processo está interligado.

Alguns dos **Modelos de Processos Prescritivos** são os seguintes: **Modelo Cascata**, **Modelo Incremental**, **Modelos Evolucionários** (divididos em: Prototipação e Espiral) e **Modelos Concorrentes**.

O Modelo Cascata (conhecido como Ciclo de Vida Clássico de um Sistema ou abordagem *Top-down*) possui enfoque sistemático e sequencial dos Processos, cada fase é iniciada somente após a conclusão da fase anterior. A Figura 1.4 representa o Modelo Cascata.

Figura 1.4 | Modelo de Processo Prescritivo: Modelo Cascata



Fonte: elaborada pela autora (2020).

O Modelo Cascata, segundo Pressman (2016) é um dos mais antigos e ainda utilizados na Engenharia de Software. A Figura 1.4 destaca os estágios do Modelo Cascata com as cinco atividades fundamentais deste modelo. Vejamos a definição dessas cinco etapas segundo Sommerville (2011):

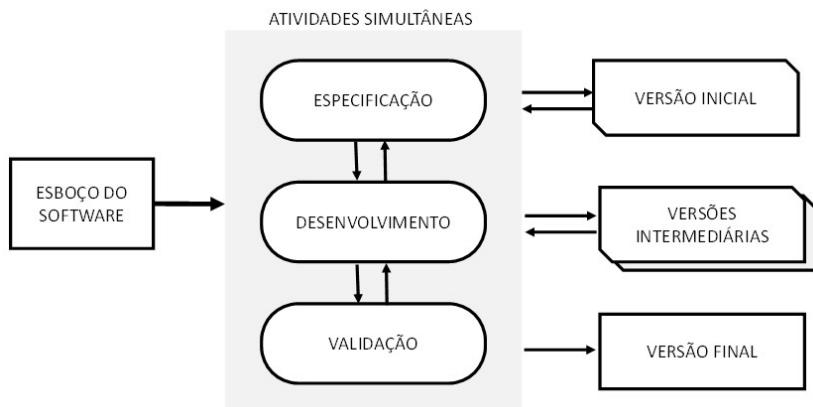
- **Análise e Especificação:** Através de consultas aos usuários do Sistema são estabelecidas as funcionalidades e é realizada a especificação (é realizada a documentação do que foi analisado) do Sistema, com participação e aprovação do cliente.
- **Projeto:** Envolve a abstração do Sistema, alocando recursos para criação do Software; nesta etapa é definida a estrutura de dados, a arquitetura do Software, as *interfaces* gráficas etc.
- **Implementação e teste de unidade:** Na fase de Implementação são realizados os programas (codificação na linguagem de programação escolhida), já o teste Unitário deve ser realizado em cada módulo (ou parte do código) para eliminar falhas de codificação.
- **Integração e Verificação:** Todas as partes (módulos) do Software são integradas e testadas para a entrega ao cliente.
- **Operação e Manutenção:** A partir da efetiva utilização do Software, são realizadas as correções solicitadas pelos usuários e ou implementados novos requisitos que o cliente tenha identificado como necessários.

O modelo em cascata é considerado o modelo mais tradicional e simples, com especificação das atividades de forma clara, além de ser uma base para modelos que surgiram posteriormente e de fácil gerenciamento. Todavia, ao adotar o Modelo em Cascata, o desenvolvimento de um Software pode se estender ao longo de meses, dependendo da complexidade do projeto, uma vez que as tarefas são realizadas de forma sequencial e o atraso em uma das etapas reflete nas demais. Um cliente pode ter que esperar diversos meses pela entrega do Software, o que pode gerar insatisfação com o produto final. Além disso, há apenas uma fase de especificação de requisitos. Diante desse cenário, uma possibilidade é adotar outro modelo de processo de software, o modelo incremental.

O Modelo Incremental é um modelo iterativo que visa, a partir de requisitos iniciais, criar pequenas versões do Software, que vão sendo entregues ao cliente, e posteriormente expandir o Software em novas versões até o sistema ideal ser totalmente construído, segundo Sommerville (2011). Nesse modelo, uma versão é um incremento. Cada incremento (ou versão) incorpora parte da funcionalidade requisitada pelo cliente. No modelo incremental ao invés do cliente receber o Software em uma única entrega, ele receberá

“pedaços” do Software (versões), a cada incremento, até que o Software seja desenvolvido por completo. Esses “pedaços” são módulos que acrescentam, ou melhoram as funcionalidades do sistema. Cada incremento (entrega) é lançado como uma nova versão, do software, até que se atinja a versão final. Na Figura 1.5, observe um esquema do Modelo Incremental. Veja que as atividades: Especificação, Desenvolvimento e Validação são realizadas de forma intercalada e não separada, e, conforme Sommerville (2011), deve ser realizado um rápido *feedback* entre as atividades simultâneas. Em cada versão (incremento) é realizado todo o ciclo de desenvolvimento de Software (do Planejamento aos Testes) e cada versão produzida é um Sistema funcional (que pode entrar em operação), mesmo ainda não estando completo (pode faltar vários requisitos).

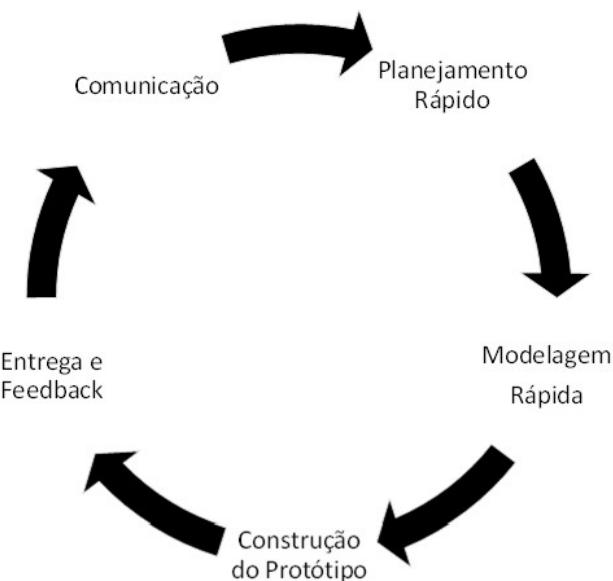
Figura 1.5 | Modelo de Processo Prescritivo: Modelo Incremental



Fonte: adaptada de Sommerville (2011, p. 22).

Outro modelo que pode ser adotado em um projeto é o Modelo de Processo Evolucionário. O Modelo de Processo Evolucionário produz uma versão cada vez mais completa do Software. Pressman (2016) afirma que esse tipo de modelo é iterativo e evolui ao longo do tempo, o que se alinha perfeitamente a um projeto do Software, pois os requisitos do negócio e do produto não são estáveis, eles mudam (evoluem) e o Software pode ser desenvolvido pensando nesta evolução. No Modelo de Processo Evolucionário aparecem dois Modelos: Prototipação e Espiral. A Figura 1.6 ilustra o esquema do Modelo Prototipação.

Figura 1.6 | Modelo de Processo Prescritivo: Modelo Evolucionário- Prototipação



Fonte: Pressman (2016, p. 45).

Na Figura 1.6 podemos observar as seguintes atividades: Comunicação, Planejamento Rápido, Modelagem Rápida, Construção do Protótipo e Entrega e Feedback. O Modelo de Prototipação começa a partir da Comunicação, identificando quais são os objetivos e as funcionalidades do Software. No Planejamento Rápido são determinados os requisitos que serão Modelados (e quais serão “deixados” para serem implementados mais tarde) e que já podem ser Modelados e Construídos. Após a Entrega, o cliente dá um Feedback e a equipe de desenvolvimento irá aprimorar os requisitos.

O modelo de Prototipação é útil para se apresentar uma versão inicial do software. Com essa versão inicial é possível fazer experimentações com usuários, testar funcionalidades, integração de componentes e sistemas, validar requisitos, dentre outras vantagens. É importante destacar que essa técnica pode ser utilizada em quaisquer partes do Processo de Software, pois a Prototipação ajuda no entendimento do que será construído para o cliente.

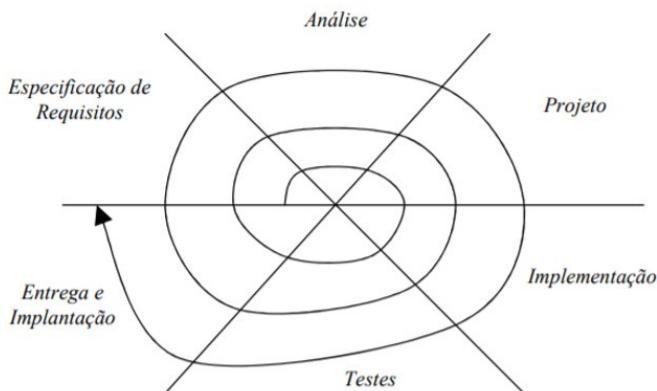
Assimile

O Modelo de Prototipagem tem seu início quando são definidos os requisitos do Software pelo cliente e o Analista de Sistemas. Após isso, é criado um protótipo do Software para que o cliente avalie e realize

testes de funcionalidades de modo a verificar se atende suas necessidades (requisitos), e servindo de orientação aos desenvolvedores. O cliente pode identificar alguma modificação, inclusão ou melhoria de alguma funcionalidade e solicitar os ajustes ao Analista de Sistemas. Ao final, esse protótipo será o produto (Software Final).

O Modelo Espiral (um tipo do Modelo Evolutivo) é iterativo como a prototipação, mas utiliza os aspectos sistemáticos e controlados do Modelo Cascata, conforme afirma Pressman (2016). O objetivo do Modelo Espiral é fornecer um rápido desenvolvimento de versão, que a cada ciclo possa gerar versões mais completas.

Figura 1.7 | Modelo de Processo Prescritivo: Modelo Evolucionário- Espiral



Fonte: Falbo (2012, p. 14).

Pressman (2016) afirma que os Modelos Prescritivos Concorrentes são representados graficamente por uma série de tarefas e técnicas maiores e estados associados a elas e que são utilizados como um paradigma para o desenvolvimento de aplicações Cliente/Servidor. Este Modelo permite que a equipe de desenvolvimento possa representar elementos concorrentes e iterativos de qualquer um dos Modelos de Processos de Software (integrando diferentes tipos de Modelos de Processos de Software). Os Modelos Concorrentes são utilizados em projetos que envolvem diferentes equipes de desenvolvimento e, conforme Pressman (2016), os planos de projeto devem ser considerados documentos vivos e a evolução de cada Processo deve ser avaliada com frequência e revisada levando em consideração as alterações. Os Modelos de Processos Concorrentes podem ser aplicados a diversos tipos de desenvolvimento de Softwares e, diferentemente do Modelo Cascata, ele

não segue uma sequência de atividades, mas estabelece uma rede de atividades que se movimentam de uma atividade para outra.

Refita

É fato que o surgimento de novas tecnologias mudam o comportamento da sociedade em usar os dispositivos eletrônicos. Será que os Modelos de Processos Prescritivos podem ser utilizados por uma empresa de desenvolvimento de Software? Ainda há espaço para esses Modelos?

Modelo de Processo Especializado

Os Modelos de Processos Especializados utilizam muitas das características de um ou mais Modelos de Processos Prescritivos e são utilizados quando existe a necessidade de uma abordagem mais especializada de Engenharia de Software. Conforme Pressman (2016), os Modelos de Processos Especializados são os seguintes:

- **Modelo Baseado em Componentes:** são utilizados em projetos de Software de Prateleira (Software de Linha), compreende aplicações de componentes de Software previamente empacotados (e vendidos em partes ou completo). São desenvolvidos para poderem ser reutilizados em outros projetos. Um componente é uma parte do independente do Software e pode ser trocado ou alterado.
- **Modelo de Métodos Formais:** compreende um conjunto de atividades que levam à especificação matemática formal do Software, fornecendo mecanismos para a descoberta e a eliminação de muitos problemas como a ambiguidade, incompletude e inconsistência. Servem como base para fazer a averiguação do código de programação com o objetivo de descobrir erros (que passariam despercebidos).
- **Desenvolvimento de Software Orientado a Aspecto:** fornece um processo e abordagem metodológica para definir, especificar, projetar e construir aspectos. O código do Software é separado de acordo com sua importância (classes orientadas a objetos é um exemplo de aspecto) e os requisitos são modelados transcendendo várias funcionalidades do Sistema.
- **Modelo de Processo Unificado:** (conhecida também como RUP – *Rational Unified Process*) aproveita as características dos Modelos de Processos tradicionais (Prescritivos), mas implementa alguns princípios da metodologia Ágil (abordada mais adiante nesta seção); é considerado um Modelo iterativo e incremental.

- **Modelos de Processos Pessoal e de Equipe:** o cerne do desenvolvimento de Software está diretamente ligado a toda equipe de desenvolvimento. No Modelo de Processo de Software Pessoal é enfatizada a medição pessoal do que foi produzido (do artefato gerado e a qualidade resultante). O Modelo de Processo de Software de Equipe objetiva a criação de uma equipe autodirigida, que se organiza por si mesma com a finalidade de produzir um Software com alto padrão de qualidade.

Modelo de Desenvolvimento Ágil

Com o rápido surgimento de novas tecnologias, o processo de negócios também foi atingido por essa velocidade, o que demanda maior velocidade no desenvolvimento de software. Nesse cenário surge uma nova forma de desenvolver Software através da Metodologia Ágil, que traz um formato mais flexível e dinâmico nos Processos de Softwares.

O Desenvolvimento Ágil procura resolver alguns problemas da Engenharia de Software, oferecendo benefícios importantes. As etapas de Levantamento de Requisitos, Análise e Projeto são muito demoradas e, de acordo com Pressman (2016), o Desenvolvimento Ágil é uma resposta ao rápido desenvolvimento de Software (os clientes querem ver resultados rapidamente) e Aplicativos (onde novas funcionalidades são agregadas, na medida que surgem novas ideias ou necessidades), tornando desta maneira o desenvolvimento mais flexível e atendendo as necessidades do cliente com mais rapidez.

Pressman (2016) afirma que o princípio do Desenvolvimento Ágil é focado nas entregas, priorizando também a comunicação entre os envolvidos de forma ativa e contínua para realizar entregas incrementais (procurando a satisfação do cliente). Sommerville (2011) destaca os seguintes princípios do Desenvolvimento Ágil:

- **Envolvimento do Cliente:** o cliente deve ter um forte envolvimento no processo de desenvolvimento do Software fornecendo os requisitos e avaliando o que foi desenvolvido.
- **Entrega Incremental:** o Software é desenvolvido com incrementos, o cliente indica os novos requisitos que devem ser acrescidos.
- **Pessoas e Não Processos:** a equipe possui liberdade de desenvolvimento, explorando ao máximo a capacidade dos desenvolvedores (não estão presos a Processos Prescritivos).
- **Aceitar as Mudanças:** os requisitos podem ser alterados, o projeto deve ser elaborado pensando nesta possibilidade.

- **Manter a Simplicidade:** a complexidade deve ser banida, a equipe deve trabalhar de forma simples e ativa.

Segundo Pressman (2016), o Processo de Desenvolvimento Ágil visa reduzir drasticamente a documentação, tornando o processo de desenvolvimento flexível e reduzindo a burocracia (presente em outros Modelos de Processos de Softwares). Nesse universo, dois Métodos Ágeis se destacam: **XP** (*Extreme Programming*) e **Scrum**.

No Método (ou metodologia) XP o *Feedback* é constante, a abordagem de desenvolvimento é incremental e a comunicação entre os envolvidos é primordial. Pressman (2016) destaca quatro atividades metodológicas que precisam ser seguidas: o Planejamento, o Projeto, a Codificação e os Testes; o desenvolvimento do Software deve ser padronizado e com o trabalho sendo realizado em pares e o cliente (um representante indicado) deve estar sempre presente para esclarecer dúvidas (ele faz parte da equipe de desenvolvimento).

A metodologia Scrum determina um processo de desenvolvimento iterativo e incremental, e pode ser utilizado em processos gerenciais. Esse método define um conjunto de regras e práticas de gestão, para alcançar o sucesso dos projetos como, por exemplo, o trabalho em equipe e comunicação melhorada. Pressman (2016) destaca que o Scrum possui as seguintes atividades metodológicas: Requisitos, Análise, Projeto, Evolução e Entrega; e em cada atividade ocorrem as seguintes tarefas principais:

- *Backlog*: lista com prioridades dos requisitos (das funcionalidades) do projeto, na qual um item pode ser adicionado ou eliminado a qualquer momento (essas são as alterações) e o gerente do produto deve registrar e atualizar as prioridades.
- *Sprints*: são unidades de trabalho para atingir um requisito (estabelecido no *Backlog*) e precisa ser ajustado dentro do *Time Box* (Janela de Tempo) para definir os prazos de entrega.
- Existe, ainda, uma reunião de planejamento na qual o *Product Owner* (dono do produto) prioriza os itens do *Product Backlog* e a equipe seleciona as atividades que ela será capaz de implementar durante o *Sprint* que se inicia.
- Reuniões *Scrum*: são reuniões breves, geralmente, de 15 minutos, chamadas *Daily Meeting* e realizadas diariamente (geralmente no início da manhã), nesta reunião são realizadas três perguntas chaves (para cada integrante da equipe): (i) O que você realizou desde a última reunião de equipe?; (ii) Quais foram os obstáculos encontrados?; (iii) O que planeja realizar até a próxima reunião?

Quando o projeto inicializa, são definidas as ideias e funcionalidades iniciais do produto a ser desenvolvido, estas ideias são chamadas Histórias e o conjunto de todas as Histórias forma o *Product Backlog*. No Método Scrum, geralmente as reuniões são conduzidas pelo *Scrum Master* (o líder da equipe) que conduz o processo e realiza avaliações das respostas de cada integrante da equipe, detectando de forma precoce eventuais problemas, como atrasos ou dificuldade de entendimento de algum requisito. Conforme Pressman (2016), no término do *Sprint* os requisitos são concluídos e o funcionamento é avaliado, melhorando o processo para a *Sprint* sequencial. Cada Sprint se encerra com um incremento ao produto (ou *Product Backlog*).

Existem ainda outros métodos de Desenvolvimento Ágil, Pressman (2016) lista os seguintes: Método de Desenvolvimento de Sistemas Dinâmicos (DSDM), Modelagem Ágil e Processo Unificado Ágil. Todos os métodos ágeis enfatizam a colaboração humana e a auto-organização como elementos chaves.

Todas os métodos de desenvolvimento Ágil são baseados no Manifesto Ágil, mas o que é isso? O Manifesto Ágil possui quatro valores fundamentais, exemplificando:

- Primeiro: indivíduos e suas interações são mais importantes que os processos e ferramentas.
- Segundo: software que funciona é mais importante que uma documentação vasta.
- Terceiro: A colaboração do cliente é mais importante que negociação de contratos.
- Quarto: responder às mudanças solicitadas é mais importante que seguir um plano.

Exemplificando

Na metodologia Scrum sempre é montado um quadro (board) para acompanhar as tarefas. Esse quadro pode ser adaptado para a realidade de cada equipe de desenvolvimento. Observe um exemplo de Quadro Scrum, na Figura 1.8.

Figura 1.8 | Quadro Scrum



Fonte: <https://www.shutterstock.com/pt/image-vector/scrum-agile-board-260061800>.

No quadro são inseridas as atividades a serem resolvidas:

- Stories: é a descrição das necessidades do cliente (sob o ponto de vista deste usuário). É o requisito explicado pelo próprio usuário.
- To Do (A Fazer): são as tarefas que deverão ser realizadas.
- In Progress (em andamento): são todas as tarefas que estão em andamento (mas não estão finalizadas).
- Testing: são os módulos que estão na fase de teste.
- Done (Pronto): é a relação do que já foi finalizado no Sistema.

O quadro do Scrum, com o passar do tempo, fica cheio, refletindo o fluxo de trabalho da equipe de desenvolvimento.

Nesta seção apresentamos os Modelos de Processos de Software e seus diferentes tipos: Modelos Prescritivos, Modelos Especializados e Modelos Ágeis. Todos os Modelos apresentados são amplamente utilizados nas empresas de desenvolvimento de Software. O que determinará qual o tipo de modelo a ser usado são dois fatores: o tipo de Software que deverá ser produzido e a experiência dos Analistas de Sistemas (envolvidos nos projetos).

É importante que você tenha conhecimento sobre esses Modelos de Processos de Software. Em uma entrevista de emprego na área de TI esse assunto é sempre abordado para: determinar o seu grau de conhecimento sobre esses modelos, visto que a grande maioria dos modelos apresentados destacam os fatores: comunicação e trabalho em equipe. Esses dois fatores são expostos como itens indispensáveis no desenvolvimento de um Software.

Vamos continuar buscando conhecimentos no mundo da Análise e Modelagem de Sistemas?

Continue seus estudos, oportunidades boas estão esperando por você!

Sem medo de errar

Você é um Analista de Sistemas e recebeu a missão de determinar qual Modelo de Processo de Software será mais indicado para o mais novo projeto de Software. A Software House, onde você trabalha, possui um novo cliente que deseja um Software para sua loja de brinquedos *online*. O Software deverá permitir que os clientes da loja comprem os produtos disponíveis no *site*. Além do *site*, o cliente deseja um aplicativo, caso o usuário queira acessar a loja para comprar ou alugar brinquedos (o aluguel somente estará disponível na versão em aplicativo).

Para responder qual Modelo de Processo de Software será mais indicado para o mais novo projeto de Software, antes devemos observar dois detalhes sobre o Software a ser desenvolvido: primeiro, o cliente deseja um site (um *e-commerce*) e, segundo, um aplicativo.

Precisaremos de uma equipe qualificada para realizar o desenvolvimento. A equipe deverá ser subdividida em duas partes: uma para o site e outra para o aplicativo. Há um porém: não foi mencionada qual plataforma deverá ser desenvolvida o aplicativo. Caso considerarmos os Sistemas Operacionais Android e iOS, deveremos ter uma equipe para cada Sistema Operacional.

Antes de propor um modelo de processo de software é preciso compreender a complexidade do problema e alinhar os prazos e valores com o cliente. Algumas perguntas que devem ser feitas antes da escolha, são:

1. O cliente tem pressa?
2. A equipe de desenvolvimento é grande o suficiente para trabalhar neste projeto?
3. A equipe domina toda a tecnologia envolvida para o desenvolvimento do site e do aplicativo?

Esse projeto é gigante, visto que o cliente deseja um *site* e um aplicativo; nesse caso, existe uma grande probabilidade de um modelo baseado nos Processos Ágeis ser o mais recomendável, entretanto, a participação do cliente é essencial para o sucesso desta metodologia. Um dos princípios dos Processos Ágeis é dividir o Software para entregá-lo em partes menores. O cliente não precisa esperar o site e os aplicativos ficarem totalmente prontos para ver o resultado final, mas pode participar ativamente de todo o processo de desenvolvimento.

Uma possível abordagem para esse projeto é adotar a metodologia Scrum. Nessa metodologia, entregas devem ser feitas a cada *Sprint*; dessa forma o cliente saberá o que vai receber e quando receberá. Os requisitos mais importantes podem ser entregues primeiro possibilitando que tanto o *site* quanto o aplicativo começem a operar. Outro ponto importante é a

construção do quadro que ajudará a organizar o cronograma e as equipes de trabalho. Veja no Quadro 1.2 um possível quadro que pode ser usado no projeto. Esse quadro apresenta algumas tarefas pertinentes ao desenvolvimento do site. Como podemos observar no *Backlog*, estão previstas as tarefas Gerar *Interfaces do Site* (conhecidas como *Wireframes*), Definir a Paleta de Cores do Site, Definir a Estrutura do Banco de Dados.

Quadro 1.2 | Exemplo de quadro Scrum para o projeto

ATIVIDADES DO BACKLOG			
ITENS DO BACKLOG	A FAZER	EM ANDAMENTO	PRONTO
Gerar Interfaces do Site (Wireframes)			
Definir Paleta de Cores			
Definir Estrutura do Banco de Dados			
.....			

Fonte: elaborado pela autora (2020).

Agora é com você! Tente propor outro modelo de desenvolvimento para o projeto. Faça uma lista de vantagens e desvantagens entre esse processo e a Metodologia Ágil. Pressman (2016) afirma que o melhor Processo de Software é aquele próximo à equipe de desenvolvimento, ou seja, um cenário ideal é propor um Processo de Software que se adapte às necessidades da equipe.

Boa jornada de estudos!

Faça valer a pena

1. Compreende um conjunto de atividades que levam à especificação matemática formal do Software, fornecendo mecanismos para a descoberta e a eliminação de muitos problemas, como a ambiguidade, incompletude e inconsistência. Servem como base para fazer a averiguação do código de programação com o objetivo de descobrir erros (que passariam despercebidos).

Assinale a alternativa correta referente ao nome deste Modelo de Processos.

- Modelo Prescritivo.
- Modelo de Processo Unificado.
- Modelo Scrum.
- Modelo Cascata.
- Modelo de Métodos Formais.

2. Pressman (2016) afirma que o Modelo Evolucionário é interativo e evolui ao passar o tempo. O projeto do Software evolui, pois os requisitos do negócio e do produto não são estáveis, eles mudam (evoluem) e o Software pode ser desenvolvido pensando nesta evolução.

Assinale a alternativa que demonstra dois tipos de Modelos Evolucionários.

- a. Scrum e XP.
- b. Cascata e Incremental.
- c. Espiral e Protótipo.
- d. RUP e Métodos Formais.
- e. Baseados em Componentes e Baseados em Objetos.

3. Os modelos de Processos Prescritivos possuem a finalidade de estruturar e ordenar o desenvolvimento de um Software e, conforme Pressman (2016), as tarefas ocorrem de forma sequencial, com diretrizes bem definidas, pois indicam uma série de atividades metodológicas, ações, tarefas, artefatos, garantias de qualidade e mecanismos de controle de mudanças para cada projeto.

Sobre os Processos Prescritivos, analise as afirmativas a seguir.

- I. Os Modelos Concorrentes são utilizados em projetos que envolvem diferentes equipes de desenvolvimento.
- II. O Modelo de Prototipagem tem seu início quando são definidos os requisitos do Software pelo cliente e o Analista de Sistemas.
- III. O Modelo de Processo Evolucionário produz uma versão cada vez mais completa do Software.
- IV. O Modelo Espiral é interativo como a prototipação, mas utiliza os aspectos sistemáticos e controlados do Modelo Incremental.

Analizando as afirmativas apresentadas, é correto o que se afirma em:

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

Referências

ELMASRI, R.; NAVATHE, S. B. **Sistemas de Banco de Dados.** 6. ed. [S. l.]: Pearson Addison Wesley, 2011.

ENGHOLM JR., H. **Engenharia de Software na Prática.** São Paulo: Novatec, 2010.

FALBO, R. A. **Engenharia de Requisitos:** Notas de Aula. Universidade Federal do Espírito Santo, Vitória, 2012.

FONSECA FILHO, C. **História da computação:** o caminho do pensamento e da tecnologia. Porto Alegre: EDIPUCRS, 2007. Disponível em: <http://www.pucrs.br/edipucrs/online/historiadacomputacao.pdf>. Acesso em: 31 jan. 2020.

PAULA FILHO, W. P. **Engenharia de software:** produtos. 4. ed. Rio de Janeiro: LTC, 2019.

PFLEEGER, S. L. **Engenharia de software:** teoria e prática. Tradução de Dino Franklin. Revisão técnica Ana Regina Cavalcanti da Rocha. 2. ed. São Paulo: Prentice Hall, 2004.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de software:** uma abordagem profissional. Tradução de João Eduardo Nóbrega Tortello. Revisão técnica: Reginaldo Arakaki, Julio Arakaki e Renato Manzan de Andrade. 8. ed. Porto Alegre: AMGH, 2016.

PRESSMAN, R. S. **Software engineering:** a practitioner's approach. 6. ed. Nova York: McGraw-Hill, 2005.

ROTH, R. M.; DENNIS, A.; WIXOM, B. H. **Análise e projeto de sistemas.** 5. ed. Rio de Janeiro: LTC, 2014

SOMMERVILLE, I. **Engenharia de software.** 9. ed. São Paulo: Pearson Prentice Hall, 2011.

TANENBAUM, A. S. **Sistemas operacionais modernos.** 2. ed. [S.l.]: Prentice Hall, 2003.

Unidade 2

Samuel Gonçalves da Silva

Processos de negócio para análise de sistemas

Convite ao estudo

Olá, aluno! Esta unidade tratará dos processos de negócios para análise de sistemas, ou seja, você terá contato com um conteúdo que permitirá criar um link entre as áreas e os processos de negócios com as demandas de análises de sistemas, ou seja, conectará os processos organizacionais à realidade de TI (Tecnologia da Informação) para que possa entregar resultados positivos. É importante ressaltar que para um bom desempenho na área de TI será necessário que os processos de negócios sejam claros, bem estruturados e documentados, de forma que qualquer pessoa da área consiga realizar as atividades propostas, além de evitar dificuldades nas decisões de TI ou até mesmo problemas futuros com tais decisões. Lembre-se de que quando falamos em projetos, grande parte dos problemas – não respeitar cronograma, estourar o orçamento planejado e não ter entregas dentro da qualidade desejada – estão vinculados aos processos mal desenhados, mal estruturados ou mal documentados, que acabam por gerar grande quantidade de mudanças nos projetos.

Nesta unidade você vai conhecer e aplicar as técnicas e ferramentas de modelagem e gerenciamento de processos de negócios, bem como sairá habilitado a utilizar os conceitos de processos de negócios, modelagem e gerenciamento de ferramentas de BPM.

Você está sendo contratado por uma empresa para atuar na área de TI, e para se manter no cargo deverá, é claro, entregar o seu melhor e atender aos requisitos da vaga. Você já possui o conhecimento, porém, precisa demonstrar de forma prática a aplicação destes conceitos e a utilização das ferramentas de forma adequada. Para isso você deverá, primeiro, identificar o principal processo de negócio da organização. Em um segundo momento, demonstrar a relação existente entre as diversas áreas organizacionais para se ter a visão completa do processo de negócio. Um terceiro ponto a ser explorado será classificar o processo de negócio com base nos padrões existentes e, por último, mostrar o impacto entre as diferentes visões (funcional e de processo).

Para um bom resultado ao longo desta unidade, você terá contato com os fundamentos de processo de negócio, isto é, compreenderá as informações básicas sobre o processo de negócios, entendendo suas áreas, bem como desenvolvendo uma visão funcional e de processos. Essas habilidades permitirão a você avançar para o entendimento de modelagem de processos de negócios, vislumbrando detalhes sobre como as coisas acontecem dentro da organização (fluxo de trabalho) e como elas são documentadas. Por fim, mas não menos importante, você vai compreender os aspectos envoltos ao gerenciamento de processos de negócios desenvolvendo uma visão da gestão de negócios, qual o papel e relevância das pessoas dentro deste contexto e quais ferramentas poderá utilizar, como utilizá-las de forma adequada, ou melhor, compreendendo a fundo os conceitos vinculados ao BPM (*Business Process Management*).

Veja, aluno, que temos um conteúdo relevante e bastante desafiador para a disciplina. Vamos ao estudo?

Seção 1

Fundamentos de processos de negócio

Diálogo aberto

Aluno, lembre-se de que o processo de negócio para análise de sistemas é uma atividade que surge da observação, parte para o estabelecimento de um plano de melhoria e por último realiza o monitoramento. Muitas vezes essa atividade é desenvolvida no nosso dia a dia de forma intuitiva, e um exemplo disso é quando mudamos um móvel ou até um componente da cozinha de lugar: visamos ganhar espaço, melhorar a utilização do que temos, deixar um item muito utilizado mais à mão, e por aí vai. A ideia de avaliar os processos de negócio tem o mesmo objetivo, e para fixarmos o aprendizado vamos ao primeiro desafio.

A empresa em questão é uma indústria alimentícia que precisa melhorar seus processos para a implementação de um novo software de gestão, e para isso você deverá auxiliar a gestora da empresa a construir os processos de negócios em uma visão funcional e de processos ponta a ponta. Dessa forma, será necessário compreender as áreas de negócios da empresa, e você deverá determinar sua classificação para que possa ter processos de negócios bem desenhados e que atendam, posteriormente, às demandas da área de TI.

Para atingir os objetivos do seu desafio, você foi informado que atualmente a organização conta com 38 colaboradores, sendo que 7 fazem parte da área administrativa (contas a pagar, contas a receber, faturamento, estoquista, compras, RH, gestor), 8 no comercial (gestor e consultores de vendas), 3 no marketing (gestor e analistas), 10 na produção (gestor e operadores de produção), 7 na logística (gestor, 3 motoristas e 3 ajudantes) e 3 na área de TI (gestor e analistas), cada área conta com um gestor. O objetivo da empresa é ter todos os processos mapeados e documentados de forma adequada para que a área de TI consiga ter entregas satisfatórias no menor tempo possível. A empresa conta com sua expertise para validar todos os processos junto aos responsáveis da área e estruturar o melhor cenário possível. Você deverá utilizar os conceitos desta seção para atender o solicitado e pensar sobre o funcionamento da organização, bem como as relações existentes dentro dela.

O entendimento sobre o segmento de atuação da organização é importante para definir os processos de negócio. Sempre que falamos de processos de negócio devemos levar em conta a relação com o cliente e pensarmos nos processos que contribuem para que esse seja atendido de forma satisfatória

ao longo de sua cadeia. A relação do cliente se inicia no ambiente externo, avança para o interno e novamente volta ao externo.

Como produto da execução das atividades propostas, você entregará à indústria alimentícia um relatório com todos os aspectos levantados por meio do estudo dos conceitos e das respostas geradas a partir das indagações realizadas por você e sua equipe para resolver essa etapa da jornada. Muitos desafios?

Não se esqueça que o desafio será vencido com maior facilidade a partir do seu empenho para adquirir os conhecimentos que serão tratados nesta seção. Conceitos esses que estão direcionados para o entendimento dos processos de negócio, como são estruturados e no que diferem de processos, bem como sua classificação e visão de negócio.

Aproveite esse momento de imersão nos estudos e mãos à obra.

Não pode faltar

Aluno, você já pensou em quantas áreas de negócios possui uma empresa? Claro que cada empresa tem uma estrutura, atividades e, por consequência, áreas diferentes, portanto, é fundamental analisar cada organização com bastante cautela para compreender e determinar quais áreas existem e como elas se relacionam.

O papel de integração da área TI às demais áreas de negócio é bastante significativo, pois somente desta forma será possível evitar as falhas no desenvolvimento de software, para que este não pareça desconectado do negócio. A área de TI, atualmente, contribui de forma estratégica para as organizações, e auxilia no desenho e gestão de processos adequados que permitem a entrega de soluções mais efetivas aos clientes.

Um cenário válido é a expertise da área de TI no desenho de processos de negócio, pois a tecnologia pode contribuir de forma efetiva para determinação de modelos de processos de negócio. Também é apto para otimizar, assim como promover redução dos custos, permitindo um posicionamento diferenciado da empresa e, por consequência, gerando vantagens competitivas, atendendo às demandas dos clientes. Um exemplo disso é o processo de vendas: conseguindo realizar um desenho mais adequado deste processo, contribuirá em mais agilidade de atendimento, entrega e na satisfação do cliente. Muitos outros processos podem ser tratados; financeiro, produtivo, entre outros.

As áreas de negócio são aquelas que têm por objetivo dar prosseguimento à missão organizacional, por meio de produção de bens ou serviços que

atenderão às necessidades do cliente externo. Tais atividades são determinadas como atividades essenciais, pois estão diretamente ligadas à atividade central (*core business*) da organização.

Quando falamos em áreas de negócios, não podemos nos esquecer do processo de negócio, mas antes é necessário lembrar do conceito de processo. Para Chiavenato (2014), processo trata uma sequência lógica e estruturada de tarefas que apresentam uma entrada (*input*) de diversos elementos, que são processados e geram saídas (*outputs*), conforme pode ser observado na Figura 2.1.

Figura 2.1 | Processo



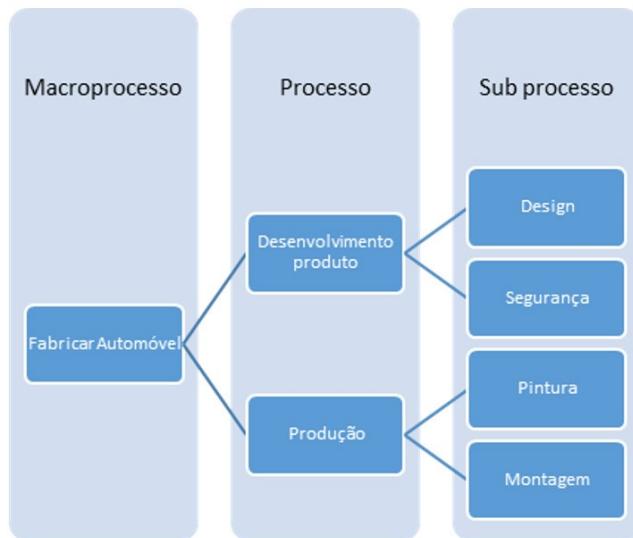
Fonte: adaptada de Chiavenato (2014).

Segundo Paim *et al.* (2009), o processo de negócio deve ser estruturado a partir da perspectiva de processos, que contempla *inputs* e *outputs*. Os *inputs*, entradas do processo, envolvem, basicamente, a área de recursos humanos (disponibilidade de pessoas), máquinas e equipamentos (desde um computador a um maquinário produtivo), materiais (insumos de produção ou não), recursos financeiros, informações (do cliente, do mercado, do estoque), procedimentos (como deverá ser executado). Por sua vez, os *outputs* representam o resultado gerado a partir do processamento de todos os recursos de entrada que têm por objetivo gerar um produto ou serviço.

Os processos ainda podem ser considerados simples ou complexos, como ligar uma televisão ou fazer uma televisão. A complexidade do processo infere diretamente no seu grau de dificuldade de modelagem e gerenciamento, sendo necessário realizar sua documentação adequada, que trata o detalhamento de como as tarefas e atividades devem ser executadas, a quem cabe a execução das tarefas para que o resultado esperado seja atingido. Vale ressaltar que um processo simples também deve ser documentado, pois favorece a padronização e pode gerar interferência em outros processos.

Em relação ao processo de decomposição, ele pode ser intitulado de macroprocesso, processo e subprocesso. A Figura 2.2 demonstra o exemplo de decomposição em que os macroprocessos tratam os processos mais abrangentes da organização, que são subdivididos em processos, que por sua vez são fracionados em subprocessos. Exemplo: uma montadora automobilística que possui a fabricação de veículo (macroprocesso), que por sua vez possui diversos processos, sendo um deles a produção, que tem como subprocessos pintura e montagem, entre outros.

Figura 2.2 | Decomposição de macroprocesso, processo e subprocesso



Fonte: elaborada pelo autor.

Para Brocke e Rosemann (2013), o processo de negócio representa a consolidação de atividades/tarefas que visam atingir um resultado que demonstre valor agregado ao cliente. Por meio de um bem ou serviço, ele realiza o sequenciamento de atividades/tarefas de forma lógica a fim de desenhar como o trabalho deverá ser executado.

Brocke e Rosemann (2013) enfatizam que os processos de negócios são classificados, conforme suas características, em processos primários, processos de suporte e processos de gerenciamento. Vale ressaltar que há interação entre eles.

Para Brocke e Rosemann (2013), os **processos primários** estão vinculados ao **core business** (negócio principal) da organização, ou seja, possuem

vínculo com as atividades essenciais; são aqueles que agregam valor ao cliente final e, por consequência, os resultados deles demonstram o grau de satisfação do cliente. Os processos primários ajudam a traduzir a missão da organização e podemos caracterizar como exemplos a produção dos produtos, a entrega ao cliente, o marketing, entre outras atividades voltadas à agregação de valor ao cliente.

Perceba que os processos primários são aqueles que podem iniciar e terminar fora da organização, pois estão diretamente relacionados com os clientes, ou seja, possuem vínculo com a entrega final (promessa realizada ao cliente).

Ainda segundo Brocke e Rosemann (2013), os **processos de suporte** suportam as ações dos processos primários e de gerenciamento, isto é, apoiam os dois outros grupos de processos. Auxiliam os processos primários a realizarem as entregas que agregam valor ao cliente, mas não entregam valor diretamente para o cliente final. Em outras palavras, agregam valor ao processo e não ao cliente. Uma boa execução dos processos de suporte contribui para que os processos primários sejam realizados da melhor forma possível e gerem impacto positivo no cliente. Basicamente deve primeiramente ocorrer um alinhamento organizacional para que todos os colaboradores estejam “remando” para a mesma direção, e posteriormente terem desenhos claro dos processos primários, de suporte e de gerenciamento. É importante compreender que o último tem como papel a melhoria continuada.

As áreas de recursos humanos, de tecnologia da informação, de contabilidade, de contas a pagar representam áreas de apoio aos processos primários, mas não estão diretamente conectados ao cliente final.

É de grande valia perceber que os processos de suporte devem ser avaliados dentro do contexto de cada organização, pois no caso de um escritório de contabilidade, o “produto” entregue é a contabilidade em si, então trata-se, neste caso, de um processo primário.

O **processo de gerenciamento** apresenta característica similar ao processo de suporte, pois funciona como um processo secundário. Ele é capaz de agregar valor ao processo primário e de suporte, entretanto, não agrava, diretamente, valor ao cliente final. Para Brocke e Rosemann (2013), o processo de gerenciamento está ligado ao monitoramento e controle das atividades organizacionais. Os processos de gerenciamento existem para que ocorra o acompanhamento dos resultados, consegue determinar se eles são satisfatórios ou não e, por consequência, determinam melhorias que indiretamente agregarão valor ao cliente, porém, por meio de melhoria nos processos e não em agregação de valor direta ao cliente. Tem como alvo fazer com que os objetivos organizacionais sejam atingidos. O monitoramento e

controle acontecem por meio de gerenciamento estratégico, gerenciamento de performance (desempenho), por exemplo, que servem para acompanhar, medir ou controlar o andamento dos processos primários e de suporte.

Falamos sobre controle, e este está ligado aos indicadores de desempenho que são estabelecidos por cada organização de acordo com suas atividades/tarefas. É relevante para o andamento do conteúdo que você tenha em mente que cada organização terá indicadores diferentes, portanto, fatores diferentes para serem monitorados e controlados.

Dessa forma fica clara a diferença entre as classificações de processos e como elas impactam os processos de negócios. A classificação serve, basicamente, para direcionar os esforços das empresas, pois quando há problemas em um processo primário o cliente “sente” de forma imediata, já quando o problema é em um processo de suporte ou de gerenciamento, o impacto no cliente, de maneira geral, é menor ou imperceptível.

Gerenciar processos se mostra como um pensamento de grande relevância para organização, pois acarreta em benefícios que gerarão impactos diretos no cliente. Para Valle, Oliveira e Braconi (2013), alguns dos benefícios gerados pelo gerenciamento de processos estão ligados a:

- Alinhamento dos processos com a estratégia organizacional.
- Melhoria da qualidade dos processos e dos produtos.
- Redução de custos por se desenvolver um olhar mais crítico.
- Muitos processos têm redução de sua complexidade e tornam-se mais simples, facilitando a interação entre as áreas.
- A melhor gestão sobre os processos permite readequação e redução de tempo em muitos casos.
- Processos não essenciais podem ser automatizados.
- Aumento do envolvimento e comprometimento dos *stakeholders* (partes interessadas).
- Melhor delegação de responsabilidades.
- Visão funcional e visão de processos.

Para De Sordi (2018), a visão funcional da organização está ligada à sua estrutura hierárquica, e isto quer dizer que trata um modelo de visualização vertical. Dessa forma, os processos são vistos por departamento e cada um gerencia um recurso específico de sua área.

Essa visão não trabalha a conectividade entre as áreas de negócios, portanto, cada área é percebida isoladamente como se não houvesse conexão com as demais áreas. Trata-se de um processo de isolamento, como se as engrenagens de um relógio não se tocassem.

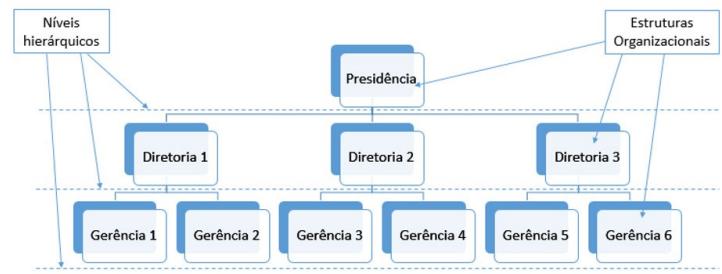
Segundo Paim *et al.* (2009), a abordagem funcional traz características de silos, isto é, cada atividade é realizada de forma isolada, com coordenação deficitária e desconhecimento de processos.

A visão em questão se demonstra com uma opção que gera baixa orientação para o mercado e, portanto, não consegue perceber as tendências e necessidades dos clientes. Os objetivos são vistos de forma isolada, cada departamento pensa apenas nos seus objetivos e não foca o objetivo global e, por consequência, os desempenhos são avaliados de forma departamental. O foco principal da visão está no desenvolvimento de competências funcionais na equipe, e isto quer dizer, que se privilegia uma visão restrita e não ampla. Como resultado, o reconhecimento se torna individualizado e restrito, sem avaliar se a atividade conseguiu agregar valor ao cliente ou ao processo que suporta. Ainda há características de orçamentos locais que não se comunicam aos demais orçamentos da companhia e não sofrem impacto pelo resultado global, mas apenas pelo local. A grande consequência de todos esses pontos é não existir uma preocupação pelos processos como um todo.

Assimile

A Figura 2.3 demonstra de forma bastante clara como é uma estrutura funcional. Também se torna simples de perceber que o fluxo de informação ocorre em um padrão hierárquico, ou seja, de cima para baixo e, portanto, não gera um processo de comunicação horizontal, mas apenas vertical. As diretorias e gerências funcionam de forma individualizada, tornando muito mais difícil atender às necessidades do cliente, pois a visão é hierarquizada e estrutural.

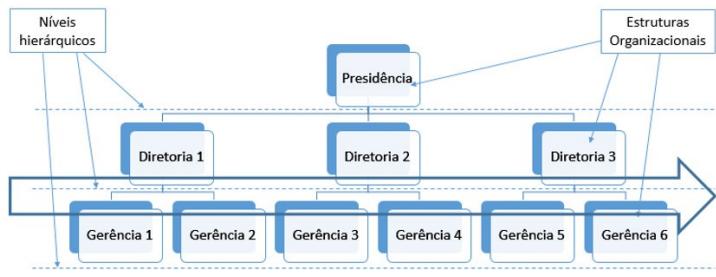
Figura 2.3 | Visão funcional



Fonte: elaborada pelo autor.

Já a Figura 2.4 mostra de forma bastante clara a integração horizontal existente entre as diretorias e gerências e, portanto, com maior possibilidade de atender às necessidades dos clientes de forma satisfatória.

Figura 2.4 | Visão de processos



Fonte: elaborada pelo autor.

Lembre-se: para a modelagem de sistema acontecer de forma adequada é necessária a integração entre as áreas.

A visão de processos ponta a ponta traz uma visão bastante ampla, pois trafega e visualiza a conexão entre todos os departamentos, isto é, em uma perspectiva horizontal. Envolve questões como tempo, custos, capacidade, qualidade, o que permite compreender a contribuição dada por cada parte para atender às necessidades do cliente. Permite uma visualização nos diferentes níveis e representa uma forma de agregar valor ao cliente.

A ABPMP (2013) descreve como processo ponta a ponta aquele que pode ser interfuncional, isto é, que vai além das funções departamentais, e conecta todos os departamentos que estão vinculados a um determinado processo. Pode ser ainda interorganizacional, que além de conectar os departamentos conecta elementos externos à organização.

Refletá

A visão de processo ponta a ponta possibilita ampliar a perspectiva organizacional para melhor atender às necessidades do cliente, pois esse é o objetivo final do processo.

O valor nada mais é do que a percepção do cliente com relação a um benefício recebido ao consumir um determinado produto ou serviço, portanto, trata-se de um elemento de extrema subjetividade, pois a percepção entre as pessoas é moldada de forma bastante diferente.

Dessa forma, de tempos em tempos as organizações devem rever suas estratégias para melhor atender às necessidades de seus clientes. É

fato que a personalização é fundamental para um melhor atendimento ao cliente. Reflita sobre as mudanças que a indústria automobilística passou. Henry Ford dizia que o cliente podia ter o carro da cor que quisesse, desde que fosse preto. Atualmente os automóveis não têm apenas cores diferentes, mas motorização, recursos como direção elétrica, computadores de bordo, entre tantos outros recursos. Tal cenário demonstra que as necessidades dos consumidores se modificaram e as organizações tiverem que redesenhar suas atividades, mudar seus processos para melhor atender tais necessidades. Avalie as mudanças que ocorreram em outros mercados.

E então, está preparado para compreender processos complexos (ponta a ponta) para desenhar softwares que atendam às demandas com maior qualidade?

Para Paim *et al* (2009), na visão por processos a estrutura organizacional é remodelada a fim de priorizar os processos como um eixo gerencial mais importante que o eixo funcional. Todas as decisões, bem como sistemas de informação, avaliação de desempenho, alocação de recursos financeiros, requisitos dos clientes, entre outros fatores são avaliados de forma conjunta, isto é, globalmente.

Exemplificando

Imagine que você não tem o conhecimento detalhado sobre processos de negócios e a visão de processos ponto a ponto e foi convidado para cuidar de um projeto para desenvolver um software de gestão para uma empresa de GLP (gás liquefeito de petróleo).

Será que você conseguiria desenvolver a ferramenta sem a determinação da integração entre as áreas organizacionais? Será que a informação de que a distribuidora de GLP compra o produto em litros e vende em quilos é uma informação relevante para o processo? Por se tratar de um produto que sofre alteração de massa em função da temperatura, é necessário ter um modelo de estoque diferenciado para a venda a granel?

Aluno, quando se fala em alteração de massa, isso quer dizer que a empresa pode comprar uma quantidade “X” do produto e vender uma quantidade “X + 1”.

Atualmente muitas distribuidoras possuem sistema de emissão de nota fiscal e boleto dentro do caminhão. Como integrar esse processo ao sistema e o como a legislação trata o assunto? Além disso, algumas delas gerenciam o abastecimento do cliente a distância, por meio de sistemas

via rádio integrados a redes de transmissão de dados, que permitem identificar o volume de GLP que o cliente possui em seu estoque.

Perceba, aluno, que as questões levantadas envolvem diversas áreas da empresa: fiscal, logística, TI, faturamento, estoque, contas a receber, contas a pagar, entre outros.

Aluno, o desenho de processos adequado determinará o sucesso do desenvolvimento de sua atividade e, portanto, é necessário que foque uma coleta de dados adequada para que possa atender aos requisitos do cliente de forma satisfatória. Atenção! Muitas vezes o processo é passado de forma superficial, portanto, busque aprofundamento na informação e o suporte de documentação.

Sem medo de errar

Caro aluno, a situação-problema desta seção traz o cenário da indústria de alimentos que possui 38 colaboradores, e você deverá ajudar a gestão da empresa bem como toda a sua equipe a desenvolver a visão de processos de negócio dentro da organização, além de determinar a classificação dos processos e sua visão funcional e de processos. Os 38 colaboradores estão distribuídos em 6 áreas organizacionais que se subdividem, portanto, aloque-os nas funções específicas de cada área de atuação, lembrando que a organização possui macroprocesso, processo e subprocesso.

Para ajudar a gestora você deverá, ainda, apropriar-se dos demais conceitos expostos nesta seção e construir o cenário organizacional, determinando de forma detalhada todos os processos que existem neste negócio. Não se limite ao conteúdo; busque informações sobre o segmento que podem complementar seus estudo e decisões, bem como estabelecer a classificação de cada um deles.

Como todo projeto, inicialmente vale realizar uma reunião com toda equipe para que ocorra um processo de conscientização sobre o seu papel e, ainda, sobre a importância do trabalho. Fora isso, você deverá navegar pela compreensão da relação entre os departamentos, portanto, levantarará questionamentos como:

- Quais são as áreas do negócio?
 - Administrativa (contas a pagar, contas a receber, faturamento, estoquista, compras, RH e gestor).
 - Comercial (gestor e consultores de vendas).

- Marketing (gestor e analistas).
- Produção (gestor e operadores de produção).
- Logística (gestor, 3 motoristas e 3 ajudantes)
- TI (gestor e analistas).
- Qual a estrutura hierárquica dessa organização?
 - Realizar a criação do organograma funcional da empresa, colocando no topo a gestora da empresa.
- Quais os processos existentes na organização?
 - Cada área de negócio possui processos específicos, portanto, relatar a importância do mapeamento de cada um deles.
 - Determinar em que ponto cada processo inicia e termina.
 - Estabelecer o processo ponta a ponta de cada um deles, ou seja, descreva por quais áreas o processo passa.
- Quais são os recursos necessários para que cada processo aconteça?
 - Quais máquinas e equipamentos são necessários.
 - Quais pessoas são necessárias.
 - Quais recursos financeiros são necessários.
 - Qual a infraestrutura necessária.
- Como cada um desses processos é classificado?
 - Para cada processo será necessário determinar se é um processo primário, de suporte ou de gerenciamento.

Lembre-se de que a determinação da classificação dos processos tem relação com agregação de valor para o cliente ou para outro processo.

Para facilitar seu trabalho, você pode construir uma tabela que contenha cada um dos processos, relacionando-os com colunas que tratem, área de início, área (s) intermediária (s) e área final do processo, se agrupa valor ao cliente ou não, e qual sua classificação. Dessa maneira conseguirá tabular de forma bastante organizada as informações desta empresa.

Apresentar um organograma funcional da empresa, um descriptivo detalhado de cada processo e uma tabela que contenha as informações indicadas anteriormente. Assim sendo, terá superado o desafio dos fundamentos de processos de negócio, estabelecendo compreensão sobre as áreas

de negócios, seus processos e a importância de estudá-las para o desenvolvimento da atividade de TI, e estará pronto para seguir em frente para mais um desafio!

Não perca tempo, mantenha seu foco, discuta com colegas e explore todo o conhecimento adquirido para realizar a atividade. Boa sorte!

Avançando na prática

E-commerce de eletroeletrônicos

Caro aluno, imagine que uma empresa atua com a venda de produtos eletroeletrônicos exclusivamente pela internet. Com o intuito de agilizar seu processo de atendimento ao cliente, reduzir seus custos operacionais e evitar falta de produtos, ela utiliza o *Cross Docking*.

Esse sistema, por manter o estoque do produto dentro de um armazém que não pertence ao *e-commerce*, ou seja, o estoque está nas mãos de um parceiro (fornecedor), permite maior agilidade ao atendimento ao consumidor e dinamismo à organização.

Todo o processo de separação, embalagem e envio da mercadoria passa a ser de responsabilidade do parceiro, porém, o atendimento ao cliente e a referência de satisfação do cliente está nas mãos do *e-commerce*. Perceba que é necessário o uso de tecnologia da informação para integrar as áreas da organização com as áreas do parceiro.

Você foi contratado para analisar o cenário organizacional e desenhar toda a estrutura de processos de negócios, e para isso deverá utilizar os conceitos envoltos às áreas de negócios, estrutura de processos, classificação de processos e tipos de visão. Imagine o funcionamento dessa organização e as variáveis que ela tem que lidar. Mão à obra!

Resolução da situação-problema

Para solucionar a problemática do *e-commerce* em questão será necessário, primeiro, ter a consciência que existe um processo principal que necessita de integração entre organizações (*e-commerce* e armazém parceiro). A partir desse elemento-chave você poderá estabelecer que o macroprocesso dessa organização pode ser definido em função de seu *core business* e desmembrado em processos e subprocessos, bem como sua estrutura funcional.

Posteriormente, você deverá detalhar o processo de venda para que possa compreender a complexidade e relações que ele gera, bem como determinar quais são os processos de suporte e de gerenciamento existentes nesta organização, e assim conseguirá realizar a classificação dos processos.

Realizando esses passos, você conseguirá explorar todos os aspectos envolvidos aos fundamentos de processos de negócio e entenderá como atuar no mapeamento de processos de negócios.

Faça valer a pena

1. Segundo Audy, Andrade e Cidral (2007), a expressão “análise de sistemas” tem duas conotações. Uma delas, em sentido amplo e que trata de uma aplicação do pensamento sistêmico na tentativa de solucionar problemas; a outra, em sentido mais restrito, está relacionada ao desenvolvimento de sistemas de informação com uso de computadores e a sistemas específicos. Em seu sentido amplo, a análise de sistemas pode ser tratada como sinônimo de pesquisa operacional, análise custo/benefício, análise operacional, etc. Quando essa expressão é associada a um sistema particular, como análise de cargos e salários, análise de organização e método, análise de custos, etc., já está acontecendo uma restrição em relação ao conceito original.

O texto mostra o conceito de análise de sistemas em seu sentido amplo e restrito. Considerando as informações do texto e realizando um paralelo com os conceitos estudados, assinale a alternativa correta.

- a. Assim como a análise de sistemas em seu sentido restrito, a visão de processos de negócios ponta a ponta traz uma perspectiva específica de um processo dentro de uma área de negócio.
- b. Assim como a análise de sistemas em seu sentido amplo, a visão funcional traz uma perspectiva específica de um processo dentro de uma área de negócio.
- c. Assim como a análise de sistemas em seu sentido restrito, a visão funcional traz uma perspectiva de um processo dentro de diversas áreas de negócio.
- d. Assim como a análise de sistemas em seu sentido amplo, a visão de processos de negócios ponta a ponta traz uma perspectiva de um processo dentro de diversas áreas de negócio.
- e. Assim como a análise de sistemas em seu sentido amplo, a visão funcional traz uma perspectiva de um processo dentro de diversas áreas de negócio.

2. O modelo de gestão fundamentado na divisão do trabalho funcional, centrada na especialização, tem limitações com relação à capacidade de coordenação do trabalho. Esse modelo revela-se restritivo para lidar com a realidade contemporânea, na qual a construção de organizações mais ágeis, integradas e flexíveis passa a ser uma condição importante para a atuação que sustente e aprimore o desempenho organizacional.

Considerando as ideias do texto, analise a seguinte situação hipotética.

Uma indústria que atua na fabricação e venda de enxovals atualmente realiza seus controles manualmente, e cada área cuida da sua informação. Acredita-se que a implantação de software de gestão possa oferecer a organização um modelo de gestão mais adequado às necessidades atuais do mercado, para que possa atingir seus objetivos organizacionais.

Avalie as asserções seguintes e a relação entre elas e, depois, assinale a alternativa correta.

- I. A implantação de um software de gestão permite que a empresa consiga otimizar os controles de cada uma das áreas organizacionais (financeiro, compras, estoque, entre outros).

PORQUE

- II. O software de gestão atende à demanda de alinhar os processos organizacionais em uma visão horizontal.
- a. As asserções I e II são proposições verdadeiras, e a II é uma justificativa da I.
 - b. As asserções I e II são proposições verdadeiras, mas a II não é uma justificativa da I.
 - c. A asserção I é uma proposição verdadeira, e a II é uma proposição falsa.
 - d. A asserção I é uma proposição falsa, e a II é uma proposição verdadeira.
 - e. As asserções I e II são proposições falsas.

3. Uma empresa atua com um sistema de gestão que integra as cotações e pedidos com o sistema de todos os seus fornecedores. Portanto, toda cotação é disparada automaticamente para três fornecedores com base nas últimas compras, e posteriormente uma delas é revertida em pedido.

Imagine que a partir de uma demanda gerada pela área de produção para atender a um cliente importante da organização, foi realizado o planeja-

mento da produção que, por consequência, fez com que o colaborador do departamento de compras realizasse uma cotação para suprir tal demanda. A cotação foi enviada automaticamente para três fornecedores, porém, devido a um problema no sistema, o colaborador não conseguiu o acesso às cotações e, por consequência, a efetivação do pedido.

Com base no texto-base, avalie as asserções a seguir e assinale a opção correta.

- a. O problema apresentado refere-se a um processo primário, pois está diretamente ligado ao atendimento da solicitação de um cliente.
- b. O problema apresentado refere-se a um processo de suporte, pois a atividade primária está funcionando de forma adequada.
- c. O problema apresentado refere-se a um processo de gerenciamento, pois a falta de gerenciamento trouxe problema para uma atividade primária.
- d. O problema apresentado refere-se a um processo de suporte, pois seu funcionamento inadequado está comprometendo o resultado do processo primário.
- e. O problema apresentado refere-se a um processo primário, pois seu funcionamento inadequado ocorre por uma falha no processo de gerenciamento.

Seção 2

Modelagem de processos de negócio

Diálogo aberto

Aluno, seja bem-vindo! Para esse desafio, você deverá compreender os conceitos vinculados à modelagem de processos de negócios, a notação BPMN (*Business Process Modeling Notation*), e seus elementos que são utilizados para formar o desenho de processos de negócio, a cadeia de valores, o fluxo de trabalho e documentação de processos de negócio.

Vale ressaltar que os elementos do BPMN tratam de uma evolução do fluxograma, isto é, permite um desenho mais aprimorado do processo, bem como gerar uma qualidade de informação mais interessante para o acompanhamento do dia a dia. O modelo de processo com base na notação BPMN vai além de informar o “trajeto” de uma atividade, ele agrupa informações e recursos detalhados ao processo, o que permite uma visão mais ampla e detalhada.

Dando continuidade à SP da indústria alimentícia de sorvetes, você deverá planejar com a gestora a modelagem de processos de negócios. Para isso, deverá realizar a apropriação dos conceitos envoltos na temática desta seção, isto é, faz-se necessário compreender de forma completa as etapas para a realização da modelagem de processo.

Neste momento, o enfoque principal da organização é avaliar e remodelar os processos de vendas da organização por ser o processo que mais corrabora para atender às necessidades dos clientes. A compreensão das relações externas (com fornecedores), dos processos internos e com os consumidores, também serão de grande valia para descrever o passo a passo para uma automatização de processos, gerando produtos, serviços e informações.

Faz-se necessário, ainda, criar o protocolo de documentação e padronização do processo com base nas regras estabelecidas. Assim, a empresa possuirá um processo melhor e mais estruturado que possibilitará melhor interpretação das ações necessárias por parte de todos os colaboradores e, por consequência, melhor alinhamento com as estratégias e objetivos organizacionais.

Para vencer o desafio você deverá ser capaz de demonstrar como os processos organizacionais se relacionam e de que forma contribuem para formação da cadeia de valor. Ainda vale lembrar que o mapeamento de processos precisa ser realizado de forma adequada, atendendo tanto as

necessidades organizacionais como as dos clientes, sendo otimizados e utilizando todos os elementos que compõem um processo de negócio.

Você deverá entregar à gestora da indústria um relatório que contenha a análise inicial do processo, a proposta de melhoria, bem como suas documentações geradas, indicando quais são os processos de suporte, os gerenciamentos envolvidos e a formação da cadeia de valor.

Organize tudo e mãos à obra, somente praticando conseguirá compreender efetivamente a aplicação dos conceitos.

Não perca tempo, boa aula!

Não pode faltar

Aluno, você já ouviu falar em modelagem de processos de negócio? Se você já ouviu, aproveite a oportunidade para afinar ainda mais seu conhecimento e, caso não tenha ouvido, vamos focar e descobrir tudo sobre esse universo!

Para tratar a modelagem de processos teremos que ampliar ainda mais a quantidade de conceitos já vistos. A modelagem de processos envolve habilidades e técnicas que fortalecem para entender e gerir os processos de negócios.

Vamos iniciar avaliando os termos de forma isolada. Segundo o dicionário Michaelis (2020), modelagem significa ato ou resultado de modelar e aplicada ao campo da informática trata da criação de modelos. Podemos entender esses modelos como representações em escala reduzida, isto é, a simplificação de algo real. Um exemplo é um esquema de um produto apresentado em manual que, a partir dele, conseguimos imaginar efetivamente o produto. O segundo conceito é o processo de negócio, que é traduzido como uma sequência de atividades executadas para atingir um objetivo (resultado) que agregue valor ao cliente.

A modelagem pode acontecer por meio de uma representação simples, que é composta por uma quantidade de elementos e áreas de negócio reduzidas, ou complexa, com uma grande quantidade dos mais variados elementos e com muitas áreas envolvidas. Os modelos podem ser matemáticos, gráficos, descriptivos ou uma combinação de alguns ou de todos, e são utilizados para organizar, aprender, prever, medir, explicar, verificar e controlar (ABPMP, 2013).

Muitos são os motivos para realizar o processo de modelagem, entre eles se destacam:

- Melhorar processos, isto é, avaliar e redesenhar processos visando melhor desempenho e atendendo melhor às demandas dos clientes internos/externos.
- Eliminar ou automatizar processos, ou seja, criar processos mais ágeis e eficazes que permitam custos reduzidos.
- Documentar processos, ou melhor, para que a organização possua informação uniforme e que todos os seus membros, por meio da documentação, possam compreender e realizar as tarefas ou atividades necessárias.

Para Valle e Oliveira (2013), existem diversas técnicas de modelagem, porém as mais difundidas são: BPMN (*Business Process Modeling Notation*), UML (*Unified Modeling Language*), IDEF (*Integrated DEFinition*) e EPC (*Event-driven Process Chain*). Algumas são utilizadas para fins específicos e outras trazem uma aplicação mais ampla.

Brocke e Rosemann (2013) afirmam que o BPMN atua com diagrama único BPD (*Business Process Diagram*) que permite desenhar os mais diversos tipos de modelagem de processo.

Já o UML, segundo Valle e Oliveira (2013), dá suporte ao desenvolvimento de softwares e, é uma linguagem de representação gráfica especificada, é independente da metodologia de modelagem de processos adotada, sendo apenas um conjunto de convenções de modelagem.

Ainda segundo Valle e Oliveira (2013), o IDEF permite a modelagem de requisitos para sistemas, as técnicas IDEF0 e IDEF3 são utilizadas para modelagem de processos de negócios. O IDEF0 tem como alvo realizar a modelagem de atividades e seus relacionamentos, não levando em conta questões funcionais ou de tempo, e permite decomposição funcional das atividades. O IDEF3 mostra como o processo opera e identifica os fluxos e aspectos de tempo entre os processos, visa detalhar como um sistema ou organização atuam.

Segundo a ABPMP (2013), o EPC visa a modelagem com base no controle de fluxo de atividades e suas dependências. Tem foco essencialmente para descrição de processos.

Independentemente da técnica, pode ser utilizada a abordagem *bottom up* (de baixo para cima) ou *bottom down* (de cima para baixo); a primeira a técnica parte do detalhamento de tarefas e atividade e depois se estabelece uma visão macro da empresa, isto é, caminha do nível mais baixo (micro) para o mais alto (macro) da organização. Na segunda técnica ocorre de

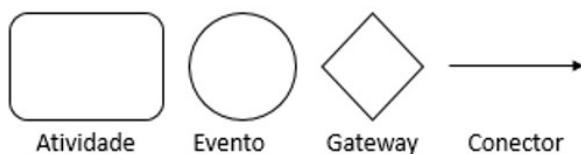
maneira inversa, primeiro se tem a visão macro (geral da organização) e posteriormente se atinge a visão do processo (tarefas e atividades).

Durante nossos estudos focaremos na técnica de modelagem realizada pelo BPMN (*Business Process Modeling Notation*).

O *Business Process Modeling Notation* (BPMN) pautou-se como uma técnica de fácil compreensão, segundo Valle e Oliveira (2013), pois atua com notações mais simples e que podem ser facilmente compreendidas, pode ser utilizado por todos os envolvidos nos processos de negócio e permite modelagem de todo o tipo de processo (compras, vendas, empréstimos, manutenção, distribuição, desenvolvimento de produtos ou serviços, entre outros).

Valle e Oliveira (2013), enfatizam que o BPMN se apresenta no formato de linhas paralelas e cada linha representa um papel diferente a ser desenvolvido na realização do trabalho. É composto por elementos básicos e específicos. São eles: atividade, evento, gateway e conector, e cada um deles possui seu ícone, conforme Figura 2.5.

Figura 2.5 | Elementos básicos do BPMN



Fonte: Valle e Oliveira (2013, p. 81).

A atividade nada mais é que o trabalho que será realizado e se subdivide em tarefa, subprocesso (colapsado ou expandido) e processo (Figura 2.6). Um exemplo de tarefa é “emitir o pedido”. Os eventos são ocorrências no processo que podem influenciar outros elementos e eventos na cadeia de processos. De alguma forma eles estão relacionados à linha do tempo dos acontecimentos, marcam o início e o término dos processos. Os gateways são elementos utilizados para controlar o fluxo de sequência e determinam decisões, bifurcações e uniões de caminhos, um exemplo é quando vamos utilizar o cartão de débito e o banco verifica: “cliente com saldo?”, e com base na resposta é tomada uma decisão. Os conectores são elementos de conexão, pois têm a capacidade de conectar os elementos (atividades, eventos, gateways), servem para ligar e demonstrar um caminho.

Figura 2.6 | Tipos de atividades



Fonte: Valle e Oliveira (2013, p. 82).

Para Valle e Oliveira (2013), a tarefa pode ter três marcadores: *Loop*, Instâncias Múltiplas e Compensação. O subprocesso colapsado é adicionado do símbolo “+” que indica outro nível de detalhes. Também podem ser utilizados marcadores de *Loop* (executa atividade até que a condição seja satisfeita), Instâncias Múltiplas (executa diversas atividades até que todas sejam satisfeitas), Compensação (serve para compensar atividade já executada do processo, isto é, desfaz a atividade) e Transacional (subprocesso com diversas atividades que devem ser completadas ou canceladas); já o subprocesso expandido contém um processo de negócio. O processo é um conjunto de objetos gráficos compostos por tarefas e subprocessos, isto é, ele não é representado por um único elemento, mas um grupo de deles.

Figura 2.7 | Subprocessos



Fonte: Valle e Oliveira (2013, p. 83).

A ABPMP (2013) enfatiza que o evento trata algo que ocorre durante o processo de negócio e afeta o fluxo do processo. Há três tipos de eventos: os de início (círculo com contorno claro), os intermediários (círculo duplo), que pode ser utilizado para enviar uma informação, e os de encerramento (círculo com contorno escuro), apresentados na Figura 2.8.

Figura 2.8 | Tipos de eventos



Fonte: Valle e Oliveira (2013, p. 84).

Segundo Valle e Oliveira (2013), todos os eventos apresentam uma indicação (representação gráfica) no centro do elemento. Nos eventos de início e intermediário essas representações significam os disparadores, e nos

eventos de fim são os resultados. A Figura 2.9 mostra os eventos de início nas duas colunas iniciais, intermediários nas duas colunas seguintes e de fim nas duas colunas finais.

Figura 2.9 | Eventos de início, intermediário e de fim



Fonte: adaptada de Valle e Oliveira (2013, p. 85-86).

Os gateways são filtros de decisão, eles separam e juntam os fluxos. Caso o fluxo não precise ser controlado não há a necessidade deste elemento.

Figura 2.10 | Tipos de gateways



Fonte: Valle e Oliveira (2013, p. 87).

O gateway exclusivo baseado em dados tem como caminhos possíveis “sim ou não” em resposta a uma pergunta, portanto trata uma decisão com escolha de apenas uma alternativa. Já o exclusivo baseado em evento depende de uma resposta externa ao processo para determinar o ponto de desvio. Exemplo: uma cotação é enviada a um cliente que pode responder (mensagem) com “sim” ou “não” e com base nesta resposta é determinado o caminho a ser tomado. É diferente do primeiro caso que se trata de algo interno à organização. Exemplo: tenho o produto que o cliente solicitou em estoque? A resposta “sim ou não” é imediata.

O último caso é o gateway inclusivo que depende de mais de uma condição para dar sequência na atividade, ou seja, ele não trabalha com “sim” e “não”, mas com a satisfação de duas ou mais condições para dar andamento na tarefa.

Segundo Valle e Oliveira (2013), os conectores servem para dar direção ao fluxo e podem ser divididos em três modalidades: sequência do fluxo, fluxo da mensagem e associação de elementos. Os conectores de sequência de fluxo determinam o caminho a ser realizado para que o processo seja finalizado, ou seja, indica tarefa a tarefa o que deve ser realizado. A direção de fluxo de mensagem possui aparência diferente para elucidar que se trata de fluxo de informação apenas e não de tarefa, e o último, associação de elementos, serve para conectar os elementos de artefatos ao diagrama.

Figura 2.11 | Tipos de conectores



Fonte: Valle e Oliveira (2013, p. 88).

BMPN usa ainda o conceito de *swinlanes* que serve para ajudar a dividir e organizar as atividades e é dividido em: *pool* (piscina) e *lane* (raia).

Para Brocke e Rosemann (2013), os *pools* devem ser utilizados quando se envolvem duas ou mais entidades de negócios ou atores determinando quem faz “o quê”. Já a *lane* é a separação das atividades associadas para um papel específico, isto é, são utilizadas para representar um ator do processo.

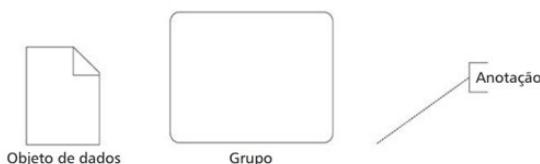
Figura 2.12 | Representação de Pool e Lane



Fonte: Valle e Oliveira (2013, p. 89).

Ainda é possível contar com os artefatos que são elementos que contribuem para que sejam mostradas informações além da estrutura básica do diagrama.

Figura 2.13 | Artefatos



Fonte: Valle e Oliveira (2013, p. 90).

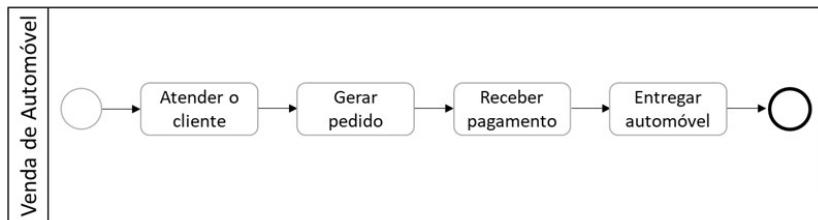
Um objeto de dados é utilizado para agregar informação ao processo, isto é, trata-se de um conjunto de informações referentes a uma atividade específica, por exemplo, a atividade “emitir pedido” é um documento que possui uma série de detalhes. Já o grupo serve para dar destaque a um grupo de atividades, ou seja, coloca em ênfase um grupo de atividades. A anotação traz comentários do processo que ajudam a entender a tarefa; mantendo o exemplo da atividade “emitir pedido” teríamos como anotação verificar impressora para que o pedido seja impresso.

A utilização de notação (representação gráfica) no processo de modelagem ajuda na comunicação, na conscientização dos processos decorrentes, permite a importação de processos entre diferentes ferramentas e gera aplicações a partir dos modelos.

Para a ABPMP (2013), o modelo de processo é composto por ícones que representam atividades, tarefas, decisões e podem conter informações. Esse modelo de processo pode ser representado por um diagrama, mapa ou modelo.

Ainda segundo a ABPMP (2013), o **diagrama** retrata apenas os principais elementos do fluxo, porém, não mostra detalhes menores relativos ao fluxo de trabalho. Ele contribui para entender as principais atividades do processo.

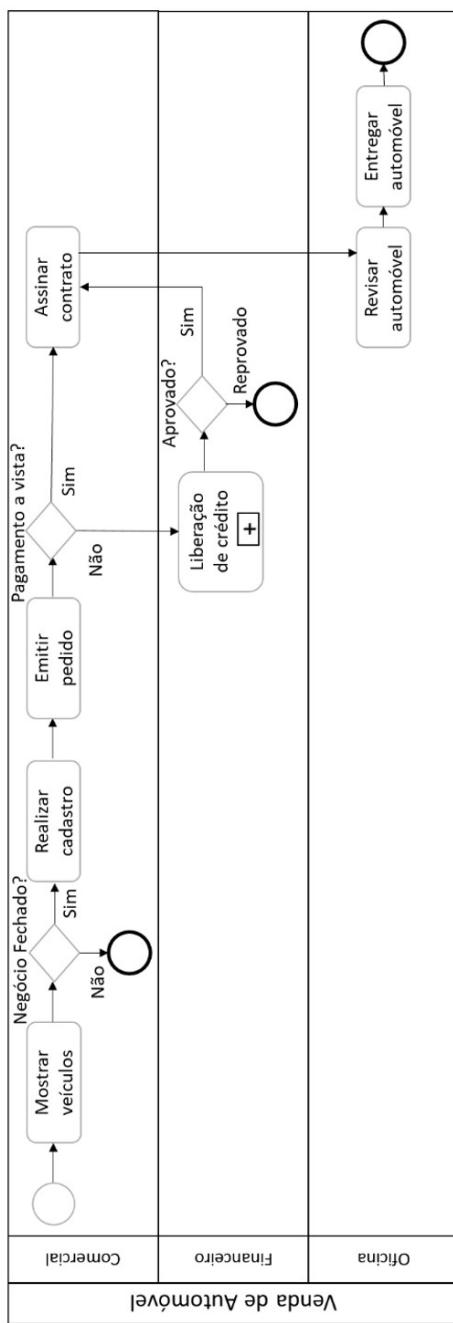
Figura 2.14 | Diagrama



Fonte: elaborada pelo autor.

Já o **mapa**, além do conteúdo do diagrama, agrupa mais detalhes acerca do processo, mostra os principais componentes do processo e apresenta maior precisão que um diagrama, e agrupa mais detalhes acerca dos atores, eventos e resultados.

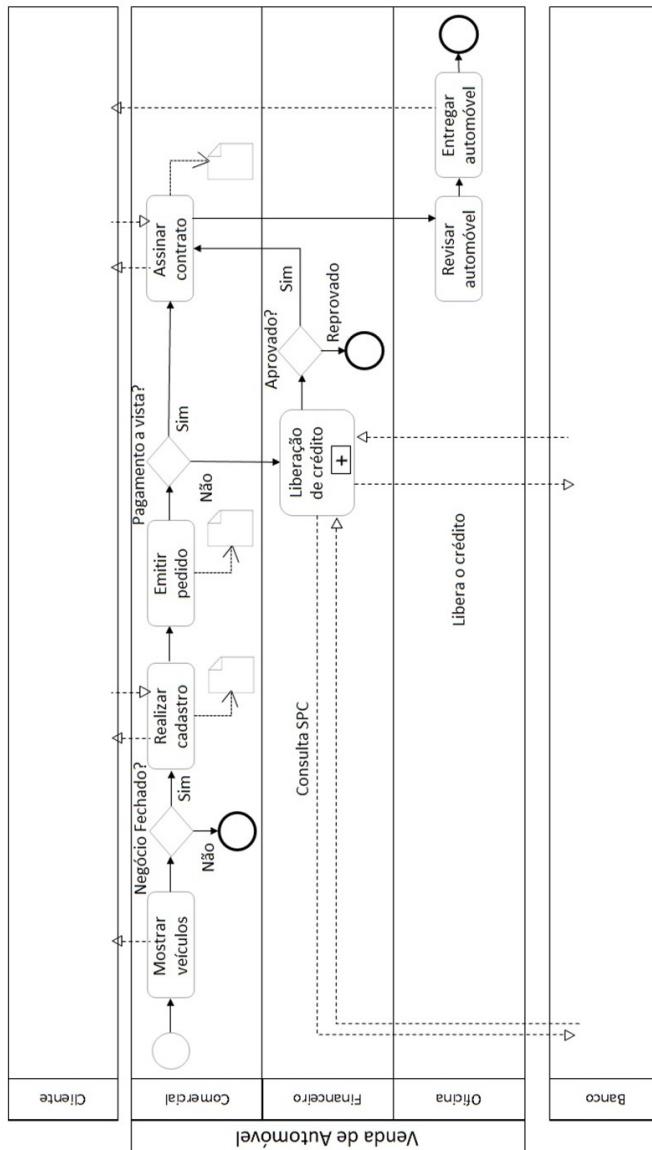
Figura 2.15 | Mapa



Fonte: elaborada pelo autor.

Por último, temos o **modelo** que é mais completo, segundo a ABPMP (2013), pois cria uma representação do negócio (atual ou futuro), de todos os envolvidos (pessoas, informação, instalações, automação, finanças e insumos) e dos fatores que afetam o comportamento do processo.

Figura 2.16 | Modelo



Fonte: elaborada pelo autor.

Segundo Valle e Oliveira (2013), o BPMN define e utiliza um único modelo de diagrama, trata-se do *Business Process Diagram* (BPD), ou o Diagrama de Processo de Negócio (DPN). Ele representa a saída gráfica de um modelo de processos no BPMN e é capaz de retratar diversos tipos de modelagem, nos quais serão apresentados os diversos elementos que formam o modelo.

A partir de todos os elementos que foram apresentados até aqui é possível realizar a modelagem de um processo de negócios. Não se esqueça que o BPMN nada mais é que uma notação evoluída de um fluxograma e que serão utilizados apenas os elementos que se fizerem necessários ao longo do processo de modelagem.

Vale ressaltar que o BPMN deve contribuir para o processo de alinhamento das estratégias organizacionais; Valle e Oliveira (2013) enfatizam que a cadeia de valores de Porter possui uma relação integrada com a classificação de processos, pois também traz uma perspectiva de processos primários e de suporte. Os autores ainda reforçam que a interface existente entre os processos na cadeia de valor é capaz de trazer vantagem competitiva para a organização.

De uma forma bastante resumida, a teoria desenvolvida por Michael Porter traz uma visão de que os processos e atividades devem agregar valor ao cliente e, por consequência, manter a organização em vantagem competitiva frente aos seus concorrentes.

A cadeia de valor demonstra, da esquerda para a direita, o fluxo dos processos que corroboram para agregar valor aos clientes e traz uma visão de macroprocesso, pois atua no ambiente corporativo. Para que a agregação de valor seja demonstrada são utilizadas diversas notações.

Os macroprocessos da cadeia de valor podem ser estratégicos, de negócio e de suporte. Cada qual possui seu papel, a estratégia tem como responsabilidade orientar todos os processos de negócios da empresa, agregando valor ao cliente e mantendo a sua margem. Os negócios são aqueles que geram valor aos clientes, pois têm conexão direta com os produtos e serviços oferecidos pela empresa. O suporte tem como papel orientar, controlar e planejar os recursos necessários aos processos de negócio.

Figura 2.17 | Cadeia de Valor de Michael Porter



Fonte: Brocke e Rosemann (2013, p. 45).

A Cadeia de Valor foi definida por Porter (1989) como um instrumento de diagnóstico de vantagem competitiva, de como criar e manter esta vantagem.

As empresas podem possuir uma ou mais cadeias de valor, que representam os processos centrais que definem a empresa, portanto há variação de empresa para empresa.

Ainda segundo Porter (1989), a cadeia de valor funciona como um meio para gerar vantagem competitiva e essa vantagem é vista como uma vantagem sustentável que permite que a organização se destaque frente a seus *players*. A cadeia de valor depende do alinhamento entre todas as áreas organizacionais para que ocorra viabilidade na realização de todos os processos com a maior eficácia possível.

Assimile

A cadeia de valor é um dos recursos que as organizações utilizam para que consigam manter a vantagem competitiva frente aos concorrentes. Por conta disso, muitas organizações têm cuidado para que cada etapa do seu processo seja um diferencial competitivo com o intuito de aumentar cada vez mais a agregação de valor ao cliente.

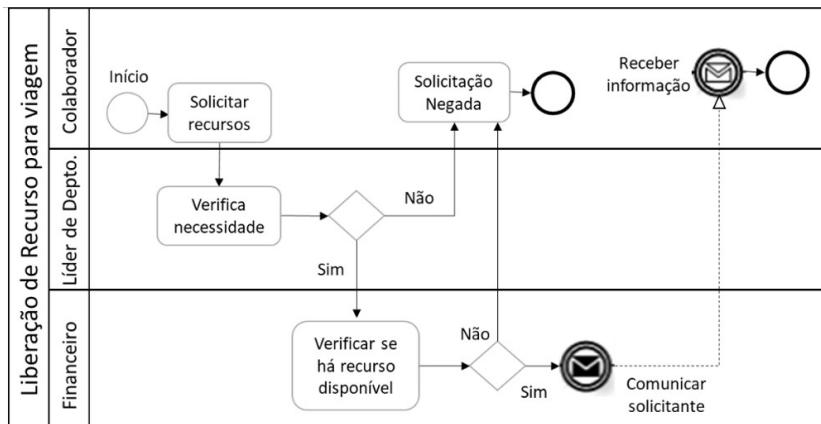
As atividades primárias ou principais são aquelas associadas à entrega do valor diretamente ao cliente final e as atividades de apoio são aquelas que contribuem para entrega de valor a outros processos.

O somatório dos esforços investidos nas atividades primárias e de apoio permitem que a organização mantenha seu diferencial competitivo e, por consequência, gera uma cadeia de valor que melhor atenda aos clientes e alavanquem a margem do negócio.

Apesar da ideia de a cadeia de valor ser voltada para organizações industriais, também é possível adaptar o modelo para empresas de serviços e comércios, basta adaptá-la à realidade de cada negócio. Lembre-se: a cadeia de valor é um elemento primordial para a análise e modelagem de processos de negócio, pois contribui para a compreensão dos objetivos organizacionais e definições de processos primários (principais), de apoio (suporte) e de gerenciamento.

Ao realizar a modelagem de processos de negócio dentro da cadeia de valores estabelecida pela organização é possível vislumbrar um fluxo de trabalho que entregará ao cliente o valor agregado necessário. O fluxo de trabalho nada mais é que a consolidação de atividades em uma área funcional com foco em eficiência e a modelagem mostrará o trabalho como um fluxo que descreve o relacionamento de cada atividade com as demais atividades executadas na área funcional (ABPMP, 2013).

Figura 2.18 | Exemplo de processo



Fonte: adaptado de Valle e Oliveira (2013).

Após a realização de mapeamento e modelagem dos processos será necessário gerar e disponibilizar a documentação necessária às áreas envolvidas em cada processo de negócio.

A documentação tem diversas utilidades, mas a principal é subsidiar a precisão das análises e embasamento dos resultados identificados. A documentação atenderá, sempre, a demanda de cada projeto, porém pode informar o motivo pelo qual o processo existe, para elucidar as interações existentes entre os subprocessos, para mostrar o fluxo de trabalho, identificar *gaps* de desempenho, motivo dos gaps, registro de coleta de dados e onde são coletados, entre muitos outros aspectos.

A ABPMP (2013) enfatiza que a documentação tem por objetivo permitir a compreensão do estado atual (*as is*), bem como subsidiar informações para o diagnóstico que permitam vislumbrar mudanças nos processos (*to be*).

Refita

Aluno, o processo de modelagem, como visto, envolve diversos elementos e, além do conhecimento do “como” o processo ocorre, também é necessário um conhecimento amplo sobre BPMN e seus recursos. Note que o processo exige, além dos elementos citados anteriormente, a dedicação de tempo. O tempo é elemento chave para um bom entendimento do processo (análise) e desenho do mesmo (modelagem).

Mas, lembrando que a modelagem traz uma visão sistematizada com propósito de gerar ganhos organizacionais, faz-se necessário documentar os processos de negócios.

Será que o processo de documentação é levado a sério da maneira que deveria? Qual o impacto de uma documentação adequada na melhoria contínua de um processo de negócio?

Pense como acontece no dia a dia, avalie na empresa que você trabalha ou trabalhou como ocorre o processo de documentação dos processos de negócio e a importância que as companhias dão para isso! Você acredita que as notações já constituem o processo de documentação?

A ABPMP (2013) reitera ainda que a documentação de análise permite elucidar uma visão geral do ambiente de negócios, para determinar o motivo pelo qual cada processo existe, registrar os processos mostrando suas interações e subprocessos, demonstrar o fluxo de trabalho (atividades realizadas dentro da área funcional), compreender os requisitos de medição de desempenho, determinar *gaps* (lacunas) de desempenho nos processos, motivos para que existam essas lacunas, compreensão de regras documentadas e não documentadas que afetam as atividades, identificação de tecnologia de informação utilizadas e em quais processos, onde os dados são coletados, armazenados e acessados, política de auditoria interna, oportunidade de melhoria e benefícios e riscos e seus impactos no processo.

Vale ressaltar que sempre que abordamos a análise do processo estamos falando do estado atual (*as is*) que tem por objetivo auxiliar na construção do cenário futuro (*to be*).

Exemplificando

A tecnologia da informação tem modificado, cada vez mais, a vida das organizações e de seus colaboradores. A indústria 4.0 já chegou, portanto os sistemas ciber-físicos, a internet das coisas e a computação em nuvem revolucionam a rotina das pessoas dentro de um contexto pessoal e profissional. Quando falamos em sistemas ciber-físicos e internet das coisas, ainda temos baixa utilização; imagine o momento em que pudermos explorar esses recursos integralmente, o quanto nossa vida poderá mudar. Talvez você esteja pensando: e o que isso tem a ver com o nosso conteúdo? A resposta é simples, ao atingirmos esse cenário a necessidade de mudança e readaptação das organizações será gigante e, por consequência, a necessidade de redesenhar processos de negócio será grande.

Diante de tamanha mudança com os sistemas ciber-físicos e a internet das coisas, você já imaginou como serão complexos os processos de negócios? O gerenciamento da cadeia fornecedor-cliente-consumidor será integrado em amplitude muito maior do que atualmente.

Só para se ter uma ideia, uma geladeira será capaz de identificar que um determinado item que está guardado nela vai acabar. Conectada a uma rede, dispara um processo de aquisição deste item, comunicando-se diretamente com um fornecedor que, na maioria das vezes, não é produtor, mas apenas um revendedor. Consegue imaginar como esse processo será amplo e envolverá uma quantidade de elementos enorme?

Agora, pense como você poderá integrar toda essa cadeia de processos sem se apropriar de ferramentas e conceitos que contribuam para isso? Saber onde está o cliente? Quem é ele? O que é importante para ele? Qual o melhor horário para atendê-lo?

Caro aluno, até o momento você teve contato com os aspectos da modelagem de processos de negócio. Busque mais informações, troque experiências com seus colegas, entenda como ocorre em sua empresa. Vincule conteúdos de outras disciplinas e lembre-se que sua jornada está apenas começando. Vamos adiante!

Sem medo de errar

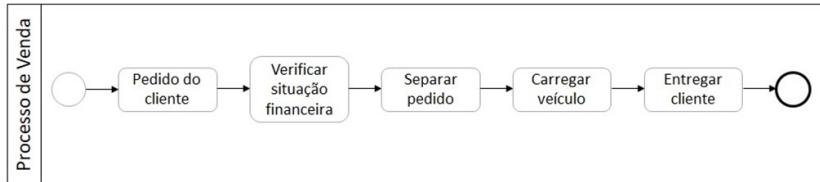
Aluno, você deverá contribuir para que a gestora da empresa tenha em mãos o desenho atual e futuro do processo de vendas, com o intuito de criar uma cadeia de valor, por meio da entrega de um produto/serviço que melhor atenda o cliente final da empresa. Vale relembrar que a indústria atua na área alimentícia, mais especificamente com produção de sorvetes. Sua atuação não é vinculada apenas à feitura dos produtos, mas também no processo de distribuição. A empresa conta com 38 colaboradores distribuídos em seis áreas diferentes (administrativa, comercial, marketing, produção, logística e TI).

Dentro do seu escopo de trabalho está não apenas a análise, mas também o redesenho do processo de vendas, pois se trata de um processo prioritário para a organização. Para isso, você utilizará os conteúdos abordados nesta seção e o levantamento realizado sobre o processo de vendas na primeira etapa da atividade para desenvolver o fluxo de trabalho e os processos de negócio. De uma forma mais detalhada, você deverá:

- 1º passo: criar o diagrama atual determinando o modelo “*as is*”.
- 2º passo: observar a execução das atividades do processo de vendas.
- 3º passo: determinar o responsável pelo processo.
- 4º passo: determinar as melhorias no processo com base no modelo “*to be*”.
- 5º passo: desenhar novo modelo.
- 6º passo: determinar a documentação dos processos.

Com base na abordagem *bottom up*, ou seja, de baixo para cima, foi possível identificar que o processo, atualmente, está desenhado da seguinte forma:

Figura 2.19 | Processo (*as is*)



Fonte: elaborada pelo autor.

Realizando uma investigação mais aprofundada, foi possível identificar que os departamentos envolvidos diretamente no processo de vendas são: comercial, expedição, faturamento, financeiro e logística. Indiretamente estão produção e TI. E o processo acontece da seguinte maneira:

1. Comercial recebe o pedido, liga para o financeiro para ter a liberação e o encaminha para a expedição.
2. A expedição realiza a separação do pedido, indicando se havia todos os itens disponíveis (falta de produto) e envia para o faturamento.
3. O faturamento emite nota fiscal/boleto e encaminha para a logística.
4. A logística efetua carregamento e sai para entrega.

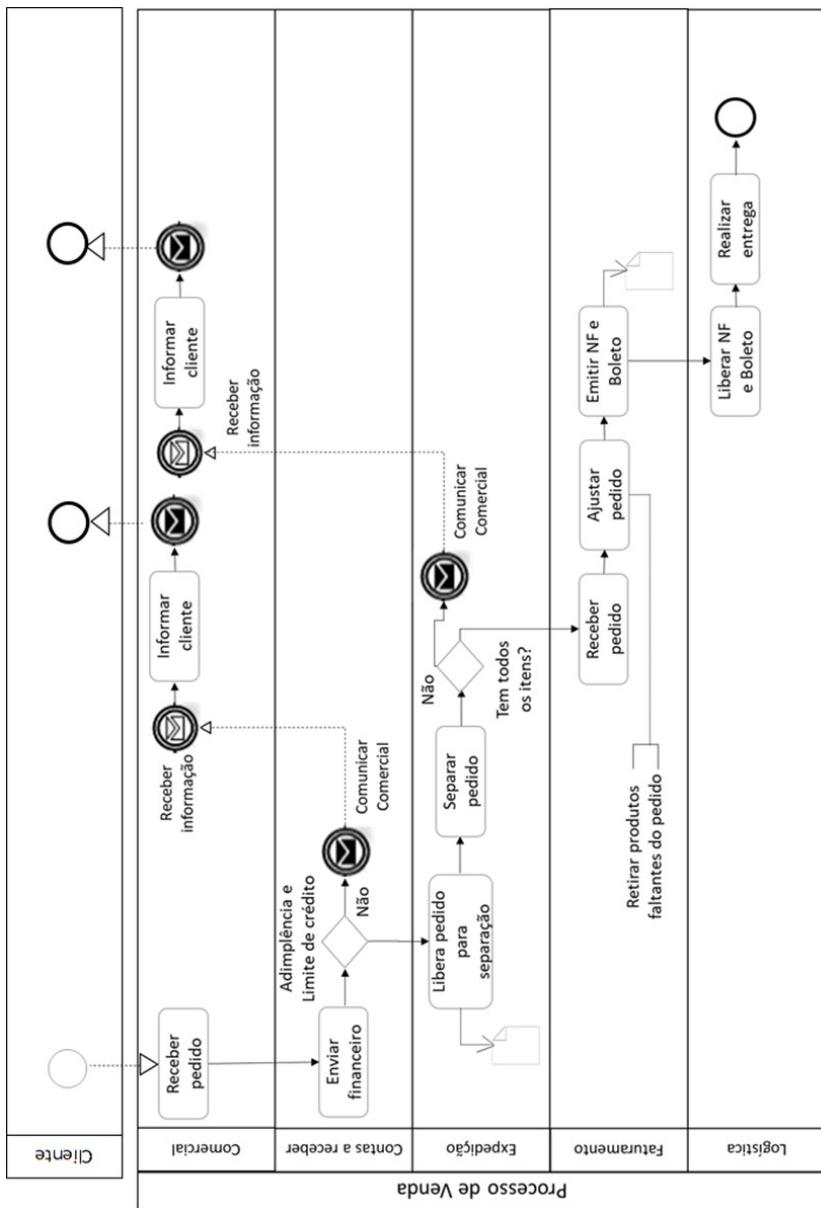
Observando o descritivo do processo, fica evidente que o dono do processo, apesar da interação entre as diversas áreas, deve ser o gestor comercial.

Com o seu conhecimento da área de TI é possível identificar diversas melhorias no processo, primeiro automatizar o processo de recebimento do pedido para que o comercial seja mais ágil e para que não faltem produtos no pedido dos clientes.

A partir dessa observação você foi capaz de melhorar o processo de negócio, agregando mais valor ao cliente, pois conterá mais detalhes e permitirá melhor gestão do processo.

O novo modelo de negócio se apresentará da seguinte forma:

Figura 2.20 | Modelo de Negócio (to be)



Fonte: elaborada pelo autor.

Além do desenho do processo é importante detalhar cada atividade, conforme segue:

1. O comercial recebe o pedido do cliente, verifica se o pedido está com preenchimento correto e encaminha para o departamento de contas a receber (via sistema).
2. O contas a receber, recebe o pedido, verifica se o cliente está adimplente e se o pedido não estoura o limite de crédito do cliente. Caso o pedido seja liberado, é enviado para expedição e, caso não seja, a informação é enviada ao comercial para que faça contato com o cliente.
3. A expedição inicia o processo de separação de produtos identificando se há em estoque os itens e as quantidades solicitadas pelo cliente e, caso faltem itens, é enviado um comunicado ao departamento comercial para que informe o cliente. O pedido é enviado para o faturamento.
4. O faturamento recebe o pedido, realiza os ajustes retirando os itens faltantes e transforma o pedido em nota fiscal. Realiza a emissão do boleto e encaminha os documentos para a logística.
5. A logística recebe as informações e efetua o carregamento do caminhão para que seja realizada a entrega aos clientes.

Mantenha o foco e aprofunde seus conhecimentos. Boa sorte!

Faça valer a pena

1. O propósito básico do BPMN é oferecer uma notação padrão para a modelagem de processos de negócio, de modo a superar as deficiências das outras técnicas de modelagem. O BPMN é originário de um acordo entre várias empresas de ferramentas de modelagem, que possuíam suas próprias notações, para utilizar uma mesma linguagem beneficiando o entendimento e treinamento do usuário final. O BPMN define e usa um único tipo de diagrama, chamado Diagrama de Processos de Negócio (DPN). Nesse diagrama, são dispostos os diversos elementos que formam o BPMN (VALLE; OLIVEIRA, 2013).

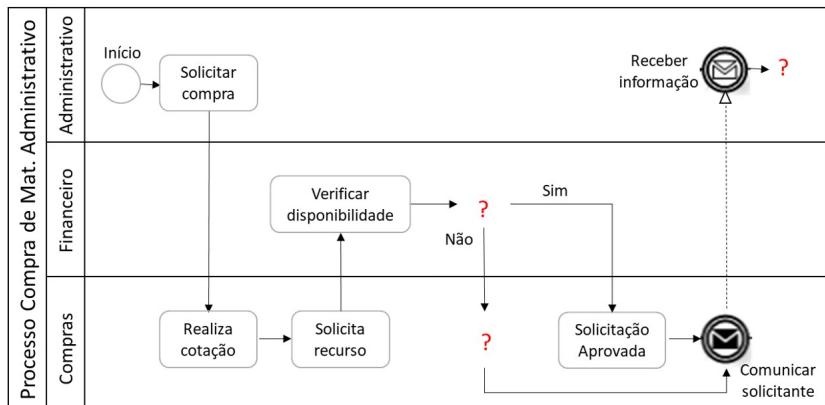
Considerando as informações do texto-base, assinale a alternativa que indica os elementos básicos que formam o BPMN:

- a. Atividades, *gateways*, *swinlanes* e artefatos.
- b. Atividades, eventos, *gateways* e conectores.
- c. Atividades, eventos, *gateways* e *swinlanes*.
- d. Atividades, conectores, *gateways* e *swinlanes*.
- e. Atividades, *gateways*, conectores e artefatos.

2. Para Brocke e Rosemann (2013), a notação de modelagem possibilita a criação de processos de negócio ponta a ponta e é concebida para abranger várias atividades de modelagem restritas a esses processos. Os elementos estruturais de BPMN permitirão que o observador seja capaz de diferenciar seções de um diagrama de BPMN utilizando grupos, pools (piscinas) ou raias. Os tipos básicos de submodelo encontrados em um modelo de BPMN podem ser processos de negócio privados (internos), processos abstratos (públicos) e processos de colaboração (globais).

Avalie a Figura 2.21 e responda quais são os elementos faltantes para que a modelagem seja finalizada.

Figura 2.21 | Modelo de Processo de Negócio



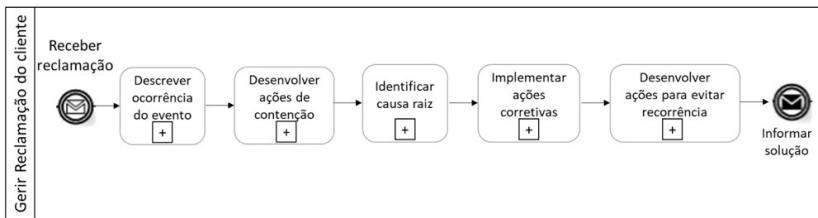
Fonte: elaborada pelo autor.

Assinale a alternativa que apresenta os elementos faltantes para que a modelagem seja finalizada.

- Raia administrativo – conector; raia financeiro – atividade; raia compras – evento intermediário.
- Raia administrativo – gateway; raia financeiro – atividade; raia compras – gateway.
- Raia administrativo – evento de fim; raia financeiro – gateway; raia compras – atividade.
- Raia administrativo – artefato; raia financeiro – gateway; raia compras – gateway.
- Raia administrativo – gateway; raia financeiro – conector; raia compras – artefato.

3. *Business Process Model and Notation* é um padrão criado pela *Business Process Management Initiative* (BPMI), incorporado ao *Object Management Group* (OMG) grupo que estabelece padrões para sistemas de informação. A aceitação do BPMN tem crescido sob várias perspectivas com sua inclusão nas principais ferramentas de modelagem. Essa notação apresenta um conjunto robusto de símbolos para modelagem de diferentes aspectos de processos de negócio. Como na maioria das notações, os símbolos descrevem relacionamentos claramente definidos, tais como fluxo de atividades e ordem de precedência.

Figura 2.22 | Fluxo de processo



Fonte: elaborada com base em ABPMP (2013).

Analisando o texto-base e a imagem, assinale a alternativa que contém os elementos que demonstram se tratar de um fluxo em alto nível:

- A imagem demonstra um fluxo em alto nível, pois é composta por diversos subprocessos colapsados
- A imagem demonstra um fluxo em baixo nível, pois se inicia e termina com um evento de mensagem.
- A imagem demonstra um fluxo em alto nível, pois é composta por uma “lane” que gera um “pool”.
- A imagem demonstra um fluxo em baixo nível, pois possui apenas cinco atividades seguidas, sem inserção de gateway.
- A imagem demonstra um fluxo simples, pois possui apenas uma “lane” que não é consolidada em um “pool”.

Seção 3

Gerenciamento de processos de negócio

Diálogo aberto

Caro aluno, nesta seção você entenderá como monitorar e controlar processos de negócio dentro das organizações e quais ferramentas (softwares) você poderá utilizar para realizar esse processo.

Será de grande relevância relacionar os conhecimentos sobre a classificação de processos de negócios, o entendimento de processo ponta a ponta, o desenho do fluxo de trabalho, a criação de diagramas, mapas ou modelos e a documentação do processo de negócio; que permitirão um bom entendimento e uma boa execução das atividades.

Retomando a situação da indústria alimentícia que atua especificamente na área de sorvetes, você deverá trabalhar o gerenciamento de processos de negócios (BPM) para ajudar a equipe da empresa a desenvolver um melhor processo de gestão. Será relevante aprofundar seu conhecimento no que diz respeito à visão de gestão de negócios, compreender as questões relacionadas à estratégia organizacional, o papel das pessoas dentro do gerenciamento de processos de negócios, bem como as ferramentas do BPM (*Business Process Management*).

É fundamental que seja desenhado um processo de monitoramento para que as ações e objetivos organizacionais sejam atingidos e, também, para que as melhorias e correções sejam realizadas de maneira eficaz. Seu engajamento na compreensão e solução do cenário apresentado deve passar por todos os aspectos do ciclo de gestão de processos, sendo fundamental lembrar que o gerenciamento dos processos acontece após a modelagem.

O gerenciamento de processos de negócio é uma atividade de extrema relevância, pois é ele que permite a criação de um processo de melhoria contínua. A criação de KPIs (*Key Performance Indicator*) por área permite o monitoramento permanente de cada área que faz parte do processo de negócio e, por isso, é importante lembrar que os indicadores de desempenho das áreas devem contribuir para a melhor execução do processo como um todo, isto é, todas as etapas estão ligadas e se relacionam.

Ao final da atividade, você deverá entregar um plano de gerenciamento de processos de negócio para ser implantado e acompanhado pela organização, a fim de que os objetivos organizacionais sejam atingidos. Esse plano

deve abordar a determinação dos KPIs envolvidos no processo de vendas da organização, separados por área utilizando o método SMART.

Mantenha-se focado que os resultados serão atingidos. Ótimo estudo!

Não pode faltar

O gerenciamento é uma atividade importante para todas as organizações e em todas as suas áreas, pois é essa atividade que permite a identificação de *gaps* (lacunas) de resultados e, a partir daí, o desenvolvimento de planos de melhoria.

Chiavenato (2014) enfatiza que são quatro as funções administrativas: planejamento, organização, direção e controle. A última está relacionada ao gerenciamento, e para o autor não há organização com desempenho satisfatório sem que existam essas quatro funções.

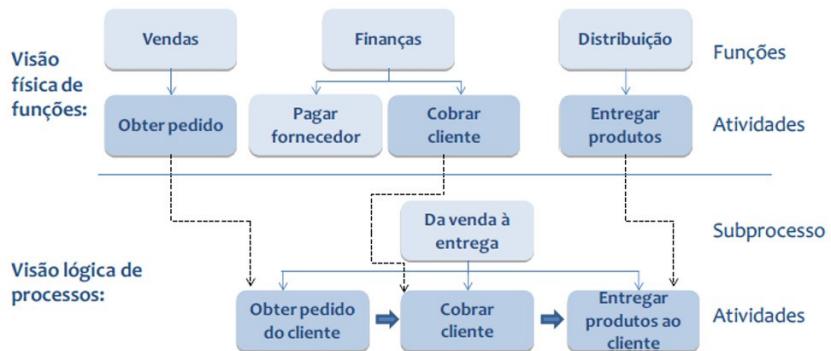
Para Chiavenato (2014) controlar – ou monitorar ou gerenciar – é uma função que permite à organização saber exatamente o que está fazendo, como está fazendo e quais as suas dificuldades e, por isso, trata-se de uma atividade tão relevante.

Valle e Oliveira (2013) destacam que com o BPM – do inglês *Business Process Management* (Gerenciamento de Processos de Negócio ou Gestão de Processos de Negócio) – os processos organizacionais são vistos de uma forma diferente, pois a visão é ampliada, há uma percepção de toda a cadeia envolvida para entregar um produto ou serviço e não se trata de uma visão verticalizada.

A visão verticalizada permite que a organização seja percebida apenas de cima para baixo, e isto quer dizer que a relação entre as áreas não é percebida. O gerenciamento de processos de negócios se torna uma matéria relevante, pois vislumbra a integração entre as áreas, a sua comunicação, a troca de informações e os tipos de relacionamento existentes, fortalecendo uma visão horizontal.

Para a *Association Of Business Process Management Professionals International* – ABPMP (2013), o BPM é necessário para olhar para o processo de um nível mais elevado do que o de execução do trabalho, pois apenas vendo o todo do processo será possível entender a relação entre as áreas e subdividi-lo por subprocessos que serão executados por diversas atividades (fluxos de trabalho) dentro das áreas funcionais. A Figura 2.23 mostra a diferença de entendimento da visão física das funções, representada pelas atividades, e a visão lógica, representada pelos processos.

Figura 2.23 | Analogia entre visão física e lógica dos processos



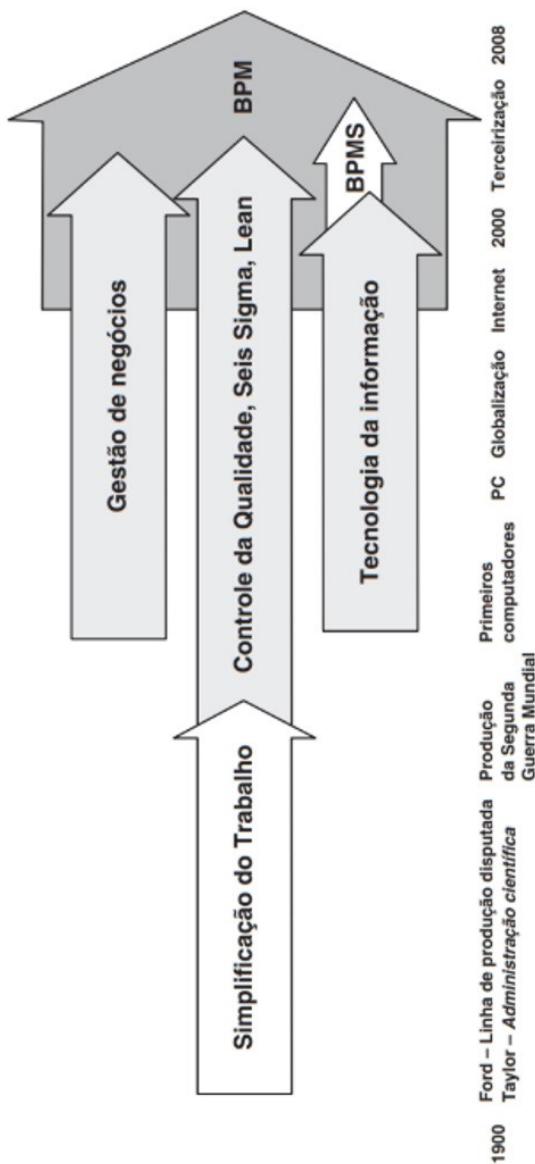
Fonte: ABPMP (2013, p. 34).

Note que a Figura 2.23 demonstra em sua visão física as áreas atuando de forma desintegrada, ou seja, cada qual realizando a sua atividade como se as atividades fossem isoladas (como se não houvesse interdependência). Já na visão lógica o enfoque está direcionado ao processo e, portanto, mostra a interconexão existente entre as atividades, pois o que importa é o resultado final, que depende de todas as atividades.

Então, é possível entender o BPM como a construção de processos ponta a ponta, que permite a integração das estratégias e objetivos das empresas. Para que funcione de forma adequada, elementos como cultura, clima, estruturas, tecnologia, políticas devem ser compreendidos para que ocorra a governança dos processos, isto é, todas as ações são direcionadas pelas mesmas regras e diretrizes com o intuito de se atingir os objetivos organizacionais. A governança deve incorporar a ideia de controle e da prestação de contas, o que para a gestão dos processos torna-se perfeitamente adequado e extremamente necessário (ARAÚJO; GARCIA; MARTINES, 2017).

Para Brocke e Rosemann (2013), os processos de negócio estão embasados em algumas metodologias de mudanças, conforme apresentado na Figura 2.24.

Figura 2.24 | Abordagens de mudanças dos processos de negócio



Fonte: Brocke e Rosemann (2013, p. 38).

A Figura 2.24 mostra o processo de evolução da melhoria dos processos, iniciando por uma visão de simplificação do trabalho, evoluindo para os

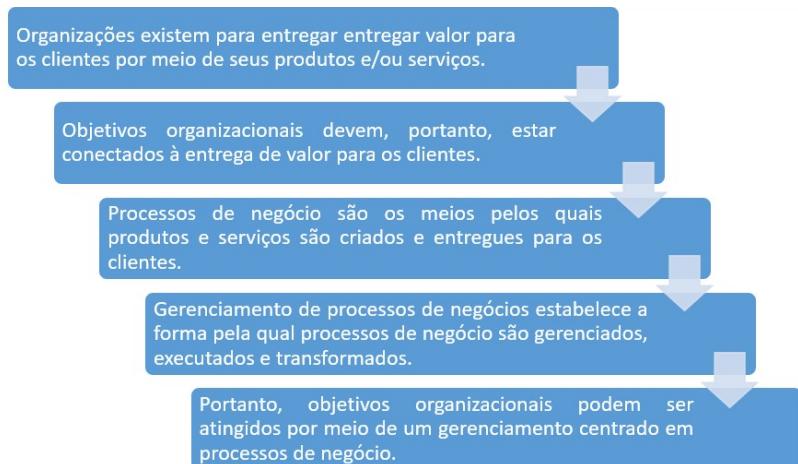
controles de qualidade, seis sigma, *lean*, gestão de negócios e tecnologia da informação. A soma de todos esses elementos culmina no BPM, e vinculado a ele temos o BPMS como suporte tecnológico (BPMS: *Business Process Management Suite or System*, que no português significa sistema de gerenciamento de processos de negócio, ou seja, sistema (software) que permite a realização do mapeamento, execução e monitoramento dos processos organizacionais).

O BPMS é uma ferramenta que permite mapear, executar e monitorar os processos funcionais com o intuito de dar-lhes uma visão de processo ponta a ponta, ou seja, contribuem para a automatização das ações e do fluxo de informações existentes nos processos.

Para Araújo, Garcia e Martines (2017), o BPMS é considerado uma evolução do *workflow* (fluxo de trabalho), pois é capaz de integrar diversos *workflows*. Por conta disso, o BPMS traz uma visão muito mais ampla e permite que ocorra integração com sistemas legados (sistemas antigos que permanecem em operação).

Para Valle e Oliveira (2013), os processos de negócio devem estar alinhados com os objetivos e estratégias organizacionais. A Figura 2.25 demonstra como tal alinhamento deve existir na visão do BPM.

Figura 2.25 | BPM e a conexão com os objetivos estratégicos



Fonte: ABPMP (2013, p. 46).

Perceba na Figura 2.25 que o desdobramento ocorre a partir da entrega de valor ao cliente (valor que deve ser percebido por ele) por meio de produtos/serviços, que devem ser alinhados aos objetivos organizacionais,

ou seja, a empresa toda deve trabalhar em prol dos objetivos. Os objetivos são desdobrados em processos e seu gerenciamento, que visam à melhor forma de atender as necessidades do cliente e, por consequência, aos objetivos organizacionais.

A visão de gestão de negócios tem uma relação íntima com a função administrativa, que segundo Chiavenato (2014) estabelece padrões de desempenho que sejam mensuráveis e que possam ser comparados com os resultados reais por meio de monitoramento para que, se necessário, sejam tomadas medidas corretivas com o intuito de atingir os objetivos propostos. Ainda segundo o autor, o controle deve abranger todos os níveis organizacionais e se divide em controles estratégicos, táticos e operacionais. Os **controles estratégicos** trazem uma visão mais genérica, de longo prazo e abordam a organização como um todo. Os **controles táticos** são mais detalhados, de médio prazo e abordam a organização em uma perspectiva departamental, e por último os **controles operacionais** são analíticos, de curto prazo e voltados às tarefas e atividades. Criando um vínculo entre a visão do autor e o conceito de BPM, é possível determinar que no caso do BPM o processo de controle também deve ocorrer em todos os níveis, porém, como o foco é o processo, o olhar deve ser mais voltado para as atividades/tarefas que compõem cada processo de negócio.

Para a ABPMP (2013), o planejamento do gerenciamento de processos de negócios está vinculado à compreensão da organização de seu nível de maturidade em processos, pois a capacidade da empresa de entender e gerenciar seus processos determinará a forma como o gerenciamento acontecerá.

A maturidade então comprehende tanto a capacidade que a empresa tem de compreender seus processos e como deles interagem. Tem como meta implementar uma série de atividades que ajudarão a organização a atingir metas preestabelecidas, corroborando para estabelecer melhores resultados na área de TI. Portanto, o CMMI (*Capability Maturity Model Integration*, ou, no português, Modelo Integrado de Capacidade de Maturidade) é extremamente relevante para o BPM, pois contribui para o melhor gerenciamento de atividades e, por consequência, o produto final é padronizado, com menor possibilidade de erros, gerando satisfação do cliente.

No ano de 1986 o desenvolvimento do modelo de maturidade de processos teve seu início e sua primeira versão foi lançada em 1991. Em 1993 a versão 1.1 foi liberada com alguns ajustes.

O SEI (*Software Engineering Institute*) é o responsável pela criação do CMM, que é a descrição dos elementos-chave de um processo de software eficaz. O CMM é baseado em cinco níveis de maturidade, com o intuito das empresas de software evoluírem seu processo.

Segundo o SEI (2010), com o objetivo de integralizar todos os modelos de capacitação que surgiram, o conceito evoluiu para CMMI, ou seja, Modelo de Maturidade da Capacitação Integrado. A evolução de CMM para CMMI se deu no período de 1999-2002: em 2000 foi lançada a versão 1.0 do CMMI, a 1.1 foi lançada 2002, em 2006 a versão 1.2 foi iniciada, e em 2010, a versão 1.3.

Para Couto (2007), a versão introdutória, chamada de 0.2 do CMMI, tinha por objetivo melhorar os processos e produtos e diminuir problemas de redundância ou falhas que a utilização de vários modelos diferentes ocasionava. As demais versões foram aperfeiçoando o modelo.

A versão 1.2 do CMMI é composta por até vinte e cinco áreas de processos, bem como seus objetivos e práticas. E suas vinte e cinco áreas são divididas em quatro grupos:

- Gerenciamento de processos.
- Gerenciamento de projetos.
- Engenharia.
- Apoio.

Na versão 1.3 do CMMI a estrutura geral do modelo foi mantida. Apenas algumas diferenças são encontradas: simplificação das práticas genéricas, revisão de glossário, métodos mais ágeis, foco na satisfação do cliente, visando mais atributos de qualidade.

Conforme demonstra a Figura 2.26, quanto maior o nível de maturidade melhor será a capacidade de gerenciamento do processo.

Figura 2.26 | Níveis de Maturidade CMMI



Fonte: ABPMP (2013, p. 218).

Como podemos perceber, a Figura 2.26 demonstra os níveis de maturidade do CMMI. No primeiro nível é dito que o sucesso depende de heróis,

pois não existem padrões; no segundo nível há planejamento, medição e controle dos processos; no nível três os processos são definidos e compreendidos pela empresa com procedimentos padrão estabelecidos e com previsibilidade de aplicação em outros projetos; no quatro, com o controle quantitativo há possibilidade de prever o desempenho; e no último nível, o foco está na melhoria continuada dos processos.

Até o momento falamos basicamente em métodos, técnicas, ferramentas e softwares, porém, para que todas essas variáveis entreguem o máximo de resultado possível, dependemos de pessoas e, neste momento, vamos compreender um pouco mais sobre o papel dos indivíduos no processo de gerenciamento de processos de negócio.

Uma das figuras mais importantes dentro da gestão de processos de negócios é o *process owner*, que nada mais é que o dono do processo de negócios. De Sordi (2018) enfatiza que os donos dos processos devem representar o comprometimento que a própria organização tem com os processos de negócio. Ainda para o autor, poucas pessoas na organização têm esse papel, pois não se trata de um gestor funcional, mas sim de um dono de processo de negócio. Esse dono, normalmente, é um gerente que tem essa responsabilidade permanente sobre o processo, diferentemente de um gerente de projetos, cuja responsabilidade termina em dado momento.

De Sordi (2018) enfatiza que o gestor de processos tem algumas atribuições principais, determinadas para garantir que todos os recursos necessários sejam disponibilizados ao longo do processo de negócio, além de avaliar de forma contínua os resultados do processo, viabilizar treinamento adequado de toda equipe e, em conjunto com a equipe, avaliar, redefinir e implementar mudanças que se façam necessárias no processo.

Como indicado pelo autor, diversas são as atribuições do gestor de processos, porém, as ações não dependem exclusivamente dele. A visão dos profissionais envolvidos no processo contribuirá de forma significativa para a avaliação e a implementação de mudanças que se fizerem necessárias ao processo, portanto, uma cultura participativa terá grande relevância para o bom andamento do processo e para o bom atendimento das demandas dos clientes.

Para Brocke e Rosemann (2013), as pessoas e a cultura organizacional trazem um grande impacto para a realização das ações necessárias de mudanças nas empresas e, portanto, será necessário que a cultura seja um motor propulsor de mudança, e não de congelamento. Diversas habilidades e competências são necessárias para o sucesso dessas ações (mudanças); algumas vinculadas à liderança e outras à equipe; mas, sem sombra de dúvida, o líder tem um papel fundamental.

Atualmente, um dos pontos de maior debate das organizações é a gestão de mudança, um fator apontado por diversos autores como um dos grandes desafios da gestão contemporânea.

Bowditch e Buono (2017) enfatizam que a mudança gera desconforto e traz impactos negativos aos processos de gestão organizacionais.

A partir da afirmação desses autores, é possível estabelecermos uma relação sobre a mudança e o BPM, pois para o BPM a disponibilidade para a mudança é extremamente necessária, já que a ideia é voltar-se para a melhoria contínua dos processos de negócio. Essa característica deve ser “enraizada” em todos os colaboradores da organização, para que o processo de mudança ocorra de forma mais simplificada.

Entretanto, mudar não é tão fácil quanto parece; existem diversos fatores que geram resistência e muitas vezes sabotagem às mudanças organizacionais. O medo do desconhecido, o fato de estar na zona de conforto, crenças, pensamento divergente, não gostar da pessoa que deu a ideia são alguns dos fatores que podem prejudicar o processo de mudança.

Para Bowditch e Buono (2017) há duas formas de a mudança cultural acontecer: a primeira é fazendo com que as pessoas “comprem a ideia”, isto é, passem a partilhar do mesmo pensamento e ver a situação pelo mesmo ângulo. A segunda envolve o recrutamento de novas pessoas para o compor a equipe, porém, este caso envolve o enraizamento de alguns valores que a equipe atual pode ter desenvolvido e o investimento na troca e, às vezes, o desenvolvimento do conhecimento técnico modelado à organização. A mudança acontece com base em cinco pontos-chave, segundo Bowdicht e Buono (2017, p. 208):

- Mudar o comportamento dos membros da organização.
- Justificar a mudança de comportamento necessária.
- Comunicar mensagens culturais sobre a mudança.
- Contratar e socializar novos integrantes.
- Remover integrantes que não se adaptam.

Segundo Brocke e Rosemann (2013), no geral os métodos BPM não têm uma abordagem de gestão de mudanças e isso se dá porque são desenvolvidos com foco em conteúdo e não em mudança de comportamento. O gestor de processo também atua como gestor de mudanças e/ou gestor de conflitos, pois apesar de gerenciar um processo, a organização ainda continua tendo uma estrutura funcional, o que gera, muitas vezes, conflitos de interesse.

Brocke e Rosemann (2013) enfatizam ainda que é necessário que esse indivíduo tenha bom relacionamento com as áreas funcionais, boa comunicação com todos os membros da equipe e saiba utilizar os recursos de TI, além de habilidade em gerenciar equipes multifuncionais, capacidade de harmonizar as diversas atividades que compõem o processo, bom relacionamento junto às organizações que fazem parte do processo, conhecimento pleno do processo de negócio que é dono, domínio total da principal entrega que o processo de negócio deve ter e conhecimento do processo de modelagem e de avaliação dos resultados.

Já a equipe de processos de negócio deve ter a capacidade de identificar e compreender o problema e criar soluções para problemas identificados, bem como aproveitar oportunidades que surgiem no processo.

Assimile

O BPMS e os sistemas legados

Os sistemas BPMS surgem com o intuito de suportar as operações da gestão por processos nas empresas. Esses sistemas têm diversas proposições que envolvem a interação de usuários de negócio na sua implantação e operação, somadas às evoluções tecnológicas que geram redução de programação, integração de sistemas legados, facilidade para desenvolver indicadores e alterar regras de negócio.

O BPMS proporciona um novo nível de automação por meio da criação e execução de aplicações que combinam lógica mostrada nos modelos de negócio com regras e dados conectados às atividades. Essa capacidade de definir e gerar aplicação de um negócio a partir de modelos e regras permite que o BPMS ofereça um gerenciamento avançado de fluxo de trabalho e reportes da situação do fluxo melhorado.

Isso direciona o BPMS a um papel de controle na orquestração de qualquer processo. Como tal, pode ser responsável por conectar processamento de fluxos de trabalho a sistemas legados usando o que for necessário (telas/funcionalidade), controlar dados usados dentro do trabalho que está sendo realizado e, em seguida, processar e entregar dados para serem armazenados. Os sistemas BPMS oferecerem muitas vantagens, mas as principais são:

- Velocidade na modelagem e geração de aplicação.
- Flexibilidade por meio da interação rápida.
- Qualidade por meio da capacidade de exteriorizar regras e testá-las.

Com tais características, o BPMS gera menor investimento por parte das organizações por substituição de sistemas legados, já que é possível realizar integração com eles.

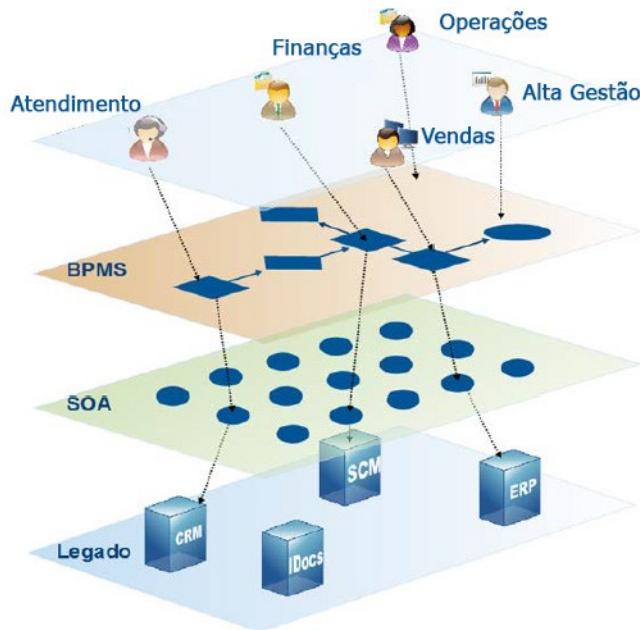
Segundo Brocke e Rosemann (2013), o gestor, bem como sua equipe, precisam de qualificações técnicas e funcionais, porém, são necessários também comportamentos que contribuem para o atingimento de objetivo. Alguns desses comportamentos são: manter o foco nos objetivos e atividades que agregam valor ao negócio, ter flexibilidade para se adequar aos processos de mudanças necessários, gostar de contribuir com o todo e compartilhar informações, estar apto a aceitar as contribuições e apontamentos realizados por outras pessoas, e saber utilizar os recursos tecnológicos vinculados ao trabalho.

Muitos são os recursos tecnológicos (softwares) que podem contribuir para o BPM e todos eles têm como objetivo final entregar relatórios que permitam a identificação de cenários e, por consequência, determinar um plano de melhoria de processo, ou seja, devem apresentar *dashboards* (painéis de desempenho) para os interessados. Vale lembrar que existem softwares simples e extremamente arrojados (complexos). São exemplos de softwares BPM o Casewise, Aris Platform, WebSphere Business Modeler, Aqualogic BPM Studio, Visio, Bizagi Modeler, Bonita Open Solution, Comindware. Frequentemente surgem novas ferramentas que podem ser incorporadas na modelagem e gerenciamento de processos.

Um sistema BPMS tem como finalidade modelar processo e fluxo de trabalho, definir regras, simular operações de negócios, automatizar processos, acompanhar o desempenho, monitorar e controlar as atividades.

Normalmente é necessário estabelecer interfaces para que os dados necessários sejam fornecidos dentro das regras estabelecidas. Em empresas que utilizam SOA (*Service-Oriented Architecture* ou Arquitetura Orientada a Serviços), o processo pode ser simplificado por meio de adaptadores, que ajudam a definir a integração, e os sistemas, que fornecem troca de dados entre as aplicações. No geral, os processos de negócios são construídos pela notação BPMN quando se usa um modelo BPMS.

Figura 2.27 | Camada SOA para interagir com as funcionalidades legadas



Fonte: ABPMP (2013, p. 375).

O modelo conceitual do BPMS valoriza os investimentos já realizados em softwares pelas organizações envolvidas com o processo de negócio, diferentemente da estratégia da reengenharia de uma década atrás, que apregoava o descarte e a substituição dos sistemas de informação legados pelo sistema ERP (DESORDI, 2018).

Para Araújo, Garcia e Martines (2017), o BPMS se mostra como um sistema que permite à organização ter um processo de mapeamento, modelagem e controle dos processos de negócio de uma forma mais ampla e integrada.

Refita

Aluno, o gerenciamento de processo tem o papel de avaliar se as ações estão caminhando conforme planejado, isto é, compara o resultado esperado com o realizado, e a partir daí determina o desempenho do processo. É notório que um único processo pode conter diversos indicadores de desempenho, pois o resultado final depende da soma de esforços entre todas as atividades que compõem o processo. O geren-

ciamento de processos envolve a utilização adequada dos recursos, e um dos recursos que tem extrema relevância para o processo são as pessoas. As pessoas e a cultura que formam a organização podem contribuir de forma exponencial para o atingimento dos objetivos propostos nos processos de negócio, entretanto, como visto anteriormente, o processo de mudança da cultura, quando necessário, e o processo de mudanças das pessoas são complexos e muitas vezes difíceis. Você já parou para pensar o quanto é difícil mudar? Em como as pessoas, de uma maneira geral, são resistentes à mudança? Se as pessoas não mudarem, faz sentido utilizar as melhores ferramentas de modelagem e gerenciamento de processos de negócio? Debata com um colega e levem para a realidade do dia a dia o quanto é difícil o processo de mudança organizacional! Cada dia mais a gestão de pessoas se torna elemento de destaque para as organizações. As organizações que melhor utilizam esses recursos saem na frente e criam vantagem competitiva.

Lembre-se de que os elementos técnicos e tecnológicos, depois de compreendidos e implementados, são fáceis de administrar, porém, é necessário administrar os elementos comportamentais. Você está pronto para isso?

Ao falarmos em Sistemas de Gerenciamento de Processos não podemos deixar de abordar a temática metas, pois são elas que darão sentido ao processo de gerenciamento. O estabelecimento de metas deve levar em conta o método SMART – acrônimo de *Specific* (específico), *Measurable* (mensurável), *Attainable* (alcançável), *Relevant* (relevante) e *Timely* (temporal). Esse método foi criado por Peter Drucker e tem como objetivo, no momento da construção das metas, realizar questionamentos usando o significado de cada letra. Segundo ele, as metas devem ser específicas, portanto, deve ter clareza nos seus objetivos e ações; também é importante que sejam mensuráveis, isto é, possíveis de serem medidas, e alcançáveis, ou seja, devem estar dentro da realidade. Além disso, precisam ser relevantes para a organização, pois não se deve perder tempo com o que não é importante e, por último, devem ser temporais, isto quer dizer que deve ser possível apurar seus resultados dentro de um período de tempo. Ao modelar um processo de negócio e pensar no seu processo de controle, Gerenciamento de Processos de Negócio, é necessário que se faça o questionamento indicado sugerido por Drucker para cada KPI planejado. É preciso que os indicadores de desempenho sejam claros, estejam ligados aos objetivos organizacionais e tenham método de apuração definido. O método não foca indicadores simples de serem medidos ou gerenciados, mas sim, a importância de serem medidos e gerenciáveis.

Exemplificando

Falando de e-commerce de produtos eletroeletrônicos, quais KPIs ele pode possuir para sua avaliação de desempenho?

- Qualidade da entrega?
- Tempo de resposta às dúvidas de clientes?
- Tempo que o cliente navega na página da empresa?

Perceba que os questionamentos realizados envolvem áreas diversas e integrações diversas, como os correios (externo), atendimento ao cliente (interno), *Google Analytics* (externo) para entender o tráfego no site da empresa.

Observe a quantidade de variáveis, cenários, ferramentas e conhecimentos necessários para fazer com que o gerenciamento de processos de negócio seja eficaz.

Caro aluno, vimos nesta seção as características do gerenciamento de processos de negócio, como criar KPIs e medi-los, a relevância do papel das pessoas neste processo, bem como a importância de utilizar ferramentas corretas no processo de modelagem e gerenciamento de processos. Agora falta muito pouco para concluir esta unidade. Mantenha o foco, expanda sua pesquisa e boa sorte!

Sem medo de errar

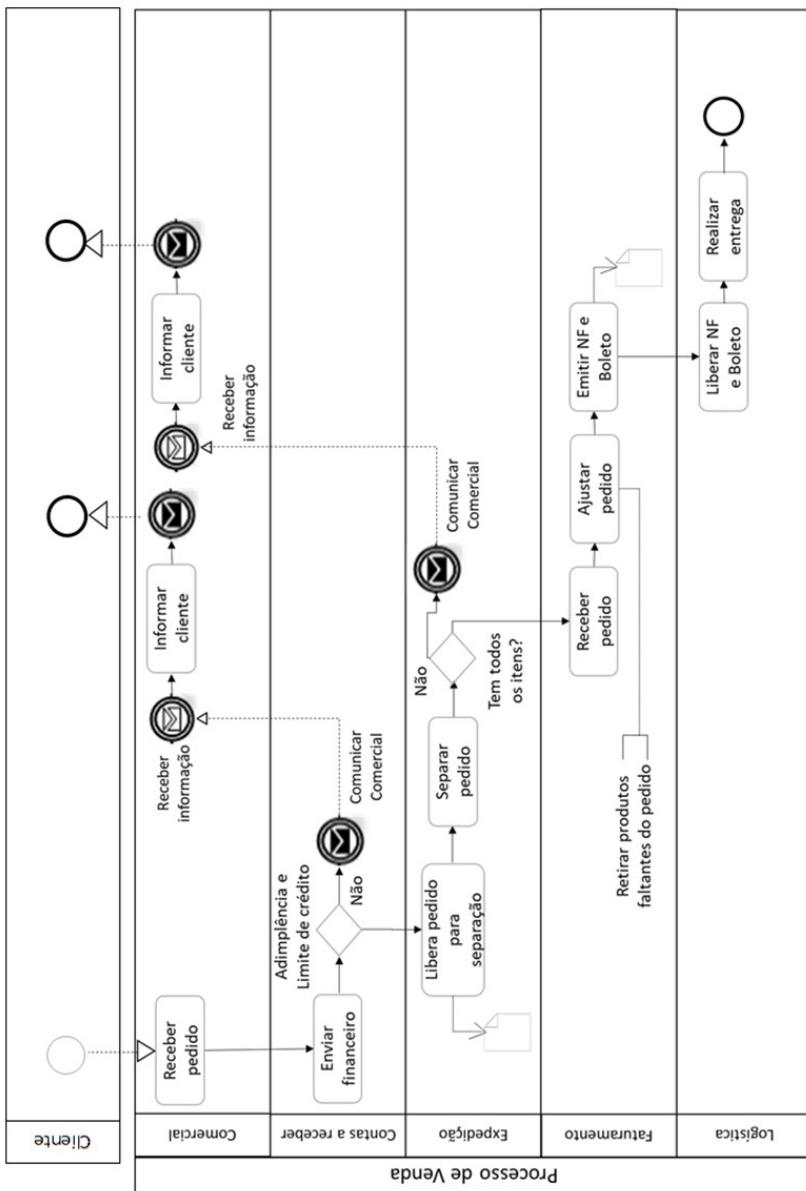
Aluno, chegou a hora de resolver o desafio que a indústria de sorvetes enfrenta. Sua demanda é desenvolver um plano de gerenciamento de processos de negócio para ser implantado e acompanhado pela organização para que os objetivos sejam atingidos.

A princípio, você deverá pernecer pelos seguintes pontos:

1. Ter em mãos a modelagem detalhada do processo de negócio de vendas.
2. Determinar indicadores de desempenho para as tarefas, atividades e, por consequência, processos de negócio.
3. Verificar, utilizando o método SMART, se todos os KPIs criados atendem às regras do método; caso não atendam, reavaliar.
4. Determinar o período de tempo em que os indicadores deverão ser avaliados.

Inicialmente, será necessário apresentar o desenho de processo como demonstrado na Figura 2.28.

Figura 2.28 | Modelo de Negócio (to be)



Fonte: elaborada pelo autor.

O estabelecimento de KPIs para o processo visa à entrega do melhor produto/serviço ao cliente e, portanto, será necessário controlar:

- Tempo de recebimento do pedido.
- Tempo de aprovação/liberação do pedido.
- Falta de produtos em estoque (para melhorar o processo de produção).
- Agilidade na comunicação com o cliente por parte do comercial.
- Tempo de separação e carregamento do pedido.
- Disponibilidade de veículo para carregamento.
- Cumprimento do prazo de entrega.
- Envio de nota fiscal e boleto como acompanhamento de carga.

Note que grande parte dos indicadores estão vinculados ao tempo de execução da atividade, mas também há indicador de comunicação com o cliente e retroalimentação do sistema, para melhoria contínua do processo produtivo.

É de suma importância utilizar o método SMART para validar cada um dos indicadores, portanto, para estar certo de que os indicadores são viáveis, deve-se lembrar de que precisa ser específico, mensurável, atingível, relevante e apurável dentro de um espaço de tempo.

O controle do tempo será realizado pelo processo evolutivo do pedido dentro do sistema, ou seja, com base nos registros gerados de transferência de pedido de uma área para outra, de forma automatizada.

Os indicadores serão avaliados diariamente para que as falhas no processo possam ser identificadas e corrigidas de forma adequada.

Muito bem! Analise as soluções e proponha novos desafios para essa demanda!

Faça valer a pena

1. O gerenciamento de processos de negócio depende da comparação dos resultados planejados e dos resultados conquistas, isto é, do estabelecimento de metas. E no que diz respeito a metas, Peter Drucker sugere o método SMART para que ocorra um processo de desenho de metas inteligentes.

Considerando as informações apresentadas, determine os papéis do método SMART, analisando as afirmativas a seguir:

- I. Criar metas que sejam quantificáveis.
- II. Criar metas importantes para a organização.

- III. Criar metas que tenham prazos estabelecidos para serem apuradas.
- IV. Criar metas que sejam simples.
- V. Criar metas possíveis de serem atingidas.

Considerando o contexto apresentado, é correto o que se afirma apenas em:

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

2. Segundo a ABPMP (2013), o Gerenciamento de Processos de Negócio (BPM – *Business Process Management*) é uma disciplina gerencial que integra estratégias e objetivos de uma organização com expectativas e necessidades de clientes, por meio do foco em processos ponta a ponta. BPM engloba estratégias, objetivos, cultura, estruturas organizacionais, papéis, políticas, métodos e tecnologias para analisar, desenhar, implementar, gerenciar desempenho, transformar e estabelecer a governança de processos.

Considerando o gerenciamento de processos de negócios, avalie as seguintes asserções e a relação proposta entre elas.

- I. A BPM auxilia no estabelecimento de princípios e práticas que permitem às organizações serem mais eficientes e eficazes na execução de seus processos de negócios.

PORQUE

- II. As pessoas são elementos-chave na execução das tarefas e na adesão às mudanças necessárias à implantação da BPM.

Com base no exposto, podemos afirmar que:

- a. As asserções I e II são proposições verdadeiras, e a II é uma justificativa da I.
- b. As asserções I e II são proposições verdadeiras, mas a II não é uma justificativa da I.
- c. A asserção I é uma proposição verdadeira, e a II é uma proposição falsa.

- d. A asserção I é uma proposição falsa, e a II é uma proposição verdadeira.
- e. As asserções I e II são proposições falsas.

3. Segundo Brocke e Rosemann (2013), o BPMS é utilizado principalmente para dois tipos de propósito: criar descrições de processos (com relação às atividades que os compõem), que podem ser empregadas para apoiar iniciativas de análise, simulação e desenho de processos, e gerar códigos executáveis que suportam o desempenho do processo, com a automação de determinados passos do processo, integração com sistemas e bases de dados utilizados pelo processo e gestão do fluxo de documentos e outros formulários que passam pelo processo.

O *workflow* é tratado por muito como um sinônimo do BPMS. Imagine que uma empresa utiliza *workflow* para automatizar os fluxos de trabalho e necessita de total integração entre as áreas de negócio envolvidas nos processos e troca de informações entre os diversos processos que são interligados, e dispõe de sistema legado.

Com base no texto-base, avalie as asserções a seguir e assinale a opção correta.

- a. É necessário implantar o BPMS, pois ele é capaz de integrar e controlar os diversos processos de negócio e permite a conexão com sistemas legados.
- b. É necessário utilizar o BPMN como forma de modelar os fluxos de trabalho e integrar os processos de negócio.
- c. O *workflow* é capaz de atender toda a demanda da empresa, pois basta trabalhar com múltiplas piscinas (pools) para integrar os processos.
- d. O BPMS não é adequado para o caso, pois muitas vezes necessita de integração por meio de uma camada SOA (*Service-Oriented Architecture*).
- e. Tanto o BPMS como o *workflow* são capazes de atender às necessidades da empresa no que diz respeito à integração dos processos de negócio.

Referências

- ABPMP. Association Of Business Process Management Professionals International. **BPM CBOK**: guia para o gerenciamento de processos de negócio corpo comum de conhecimento v. 3.0. Brasília: ABPMP Brasil, 2013.
- ARAÚJO, L. C. G.; GARCIA, A. A.; MARTINES, S. **Gestão de Processos**: melhores resultados e excelência organizacional. 2. ed. São Paulo: Atlas, 2017.
- AUDY, J. N.; ANDRADE, G.; CIDRAL, A. **Fundamentos de sistemas de informação**. Porto Alegre: Bookman, 2007. Disponível em: <https://integrada.minhabiloteca.com.br/#/books/9788577801305/cfi/0!/4/2@100:0.00>. Acesso em: 28 dez. 2019.
- BOWDITCH, J. L.; BUONO, A. F. **Elementos do Comportamento Organizacional**. São Paulo: Cengage Learning, 2017.
- BRIAGA, M. Minimizando Riscos na Implantação de BPM (Business Process Management). **Revista Estratégica**, São Paulo, vol. 10, n. 2, 2010, p. 75-89. Disponível em: <https://doi.org/10.20985/1980-5160.2016.v11n2.699>. Acesso em: 17 jan. 2020.
- BROCKE, J. V.; ROSEMANN, M. **Manual de BPM**: gestão de processos de negócio. Porto Alegre: Bookman, 2013.
- CAPELLI, C. Transparéncia de processos organizacionais. In: II Simpósio Internacional de Transparéncia em Negócios, 2008. **Anais** [...]. Rio de Janeiro: PUC. Disponível em: https://www.researchgate.net/profile/Claudia_Cappelli/publication/242668813_TRANSPARENCIA_DE_PROCESSOS_ORGANIZACIONAIS/links/548effad0cf2d1800d861e2c/TRANSPARENCIA-DE-PROCESSOS-ORGANIZACIONAIS.pdf. Acesso em: 28 dez. 2019.
- CHIAVENATO, I. **Administração**: teoria, processo e prática. 5. ed. Barueri: Manole, 2014.
- COSTA, C. A. A.; TONOLLI JR., E. J.; OLIVEIRA, J. R. Avaliação do BPM e BPMS no setor de manutenção de uma IES. **Revista Eletrônica Sistemas & Gestão**, Caxias do Sul, v. 11, n. 2, 2016, p. 133-149. Disponível em: <https://doi.org/10.20985/1980-5160.2016.v11n2.699>. Acesso em: 17 jan. 2020.
- COUTO, A. B. **CMMI**: Integração dos Modelos de Capacitação e Maturidade de Sistemas. Rio de Janeiro: Ciência Moderna, 2007.
- DE SORDI, J. O. **Gestão de Processos**: uma abordagem da moderna administração. 5. ed. São Paulo: Saraiva, 2018.
- DRUCKER, P. **The Practice of Management**. Norte America: Butterworkth Heinemann, 2007.
- GONCALVES, J. E. L. As empresas são grandes coleções de processos. **Revista de Administração de Empresas**, São Paulo, v. 40, n. 1, p. 6-9, mar. 2000. Disponível em: <http://dx.doi.org/10.1590/S0034-75902000000100002>. Acesso em: 28 dez. 2019.

GRANATO, L. Estes 17 cargos na área de tecnologia vão bombar em 2019. **Revista Exame**, 11 dez. 2018. Disponível em: <https://exame.abril.com.br/carreira/17-cargos-na-area-de-tecnologia-que-vao-bombar-em-2019/>. Acesso em: 28 dez. 2019.

MACHADO, P. **Análise comparativa entre ferramentas gratuitas de gestão e automação de processos (BPMS)**. 2017. 82 p. Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação) – Universidade do Sul de Santa Catarina, Palhoça, 2017. Disponível em: <http://www.riuni.unisul.br/handle/12345/3912>. Acesso em: 17 jan. 2020.

MICHAELIS. Moderno Dicionário da Língua Portuguesa. Disponível em: <http://michaelis.uol.com.br/moderno/portugues/index.php>. Acesso em: 10 jan. 2020.

PAIM, R. et al. **Gestão de processos:** pensar, agir e aprender. Porto Alegre: Bookman, 2009. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788577805327/cfi/2!/4/4@0.00:59.4>. Acesso em: 28 dez. 2019.

PORTER, M. E. **Vantagem competitiva:** Criando e sustentando um desempenho superior. Rio de Janeiro: Campus, 1989.

SEI. Software Engineering Process Management Program. **CMMI® for Development**. Version 1.3. Carnegie Mellon University, nov. 2010. Disponível em: https://resources.sei.cmu.edu/asset_files/TechnicalReport/2010_005_001_15287.pdf. Acesso em: 1 fev. 2020.

TESSARI, R. **Gestão de processos de negócio:** um estudo de caso da BPMN em uma empresa do setor moveleiro. Dissertação (Mestrado em Administração) – Universidade de Caxias do Sul, Caxias do Sul, 2008. Disponível em: <https://repositorio.ucs.br/handle/11338/428>. Acesso em: 10 jan. 2020.

VALLE, R.; OLIVEIRA, S. B.; BRACONI, J. Descrevendo os processos de sua organização. VALLE, R.; OLIVEIRA, S. B. de. In: **Análise e modelagem de processos de negócio:** foco na notação BPMN. São Paulo: Atlas, 2013.

VALLE, R.; OLIVEIRA, S. B. de. **Análise e modelagem de processos de negócio:** foco na notação BPMN. São Paulo: Atlas, 2013.

Unidade 3

Claudia Werlich

Engenharia de requisitos

Convite ao estudo

Olá, seja bem-vindo!

Você está prestes a entrar numa área fascinante da Informática: a engenharia de requisitos, cujos processos envolvem várias atividades que são responsabilidade do analista de sistemas. Dessa forma, os programadores precisam dominar cada vez mais essa área, uma vez que podem participar ativamente do processo de projeto e desenvolvimento de um sistema. Não serão raras as oportunidades que você terá para projetar um software: mesmo trabalhando para uma empresa de TI, nada impede que você desenvolva seus próprios sistemas. Por isso, compreender os processos da engenharia de requisitos irá ajudá-lo nessa tarefa. Esses processos serão apresentados, exemplificados e discutidos nesta unidade.

Na primeira seção, seu desafio será realizar o levantamento dos requisitos para o desenvolvimento de um software para um grande cliente que cultiva e comercializa plantas ornamentais. Na segunda seção, você determinará como será feita a coleta de informações, para obter o máximo de informação sobre o software a ser desenvolvido para o controle de venda de plantas ornamentais. Na terceira seção, o desafio final será criar uma Documentação da Especificação de Requisitos (DER) para o software a ser desenvolvido.

Para o desenvolvimento dos seus estudos, você aprenderá, na primeira seção, os processos da Engenharia de Requisitos, os conceitos, os tipos de requisitos e as atividades que envolvem esses processos. Na segunda seção, veremos a importância da elicitação, especificação e validação de requisitos, juntamente da sua negociação e monitoramento. Já na última seção trabalharemos com a Modelagem de Requisitos, demonstrando as suas técnicas e toda a documentação da especificação da elicitação de requisitos, aplicando a modelagem de Requisitos na Análise de Sistemas.

Bons estudos!

Seção 1

O processo de engenharia de requisitos

Diálogo aberto

Caro aluno, seja bem-vindo à seção sobre o processo de engenharia de requisitos.

Um dos pontos que merecem destaque na elaboração do projeto de software é que, na engenharia de requisitos, qualquer erro ou esquecimento pode impactar no resultado final do produto desenvolvido. Essa é uma fase muito importante, pois dela depende o sucesso do software desenvolvido.

Ser um profissional de sucesso na área de Tecnologia da informação (TI) está relacionado a saber interpretar o que o cliente (nossa usuária final) deseja e, principalmente, saber do que o cliente nem sabe que precisa. Parece confuso, mas existem técnicas que ajudam no cumprimento desta fase primordial do desenvolvimento de um software e que o deixarão apto a conhecer e aplicar as técnicas para especificação, validação e modelagem de requisitos.

O desafio desta seção é aplicar e identificar os processos de engenharia de requisitos, assim como suas técnicas e análises, para atender à demanda de um cliente, que diz respeito ao desenvolvimento de um software para uma empresa que vende produtos e plantas ornamentais. Porém, para realizar essa tarefa, você necessita conhecer os conceitos e as técnicas desta seção.

Você trabalha em uma empresa de desenvolvimento de softwares e, por ser bem detalhista, você ficou encarregado do levantamento de requisitos para a produção de um software para um grande cliente que cultiva e comercializa plantas ornamentais. A fim de conhecer bem as rotinas da empresa, serão necessárias várias visitas, pois, dessa forma, além de conhecer o fluxo de trabalho da companhia, você poderá investigar como funcionam as rotinas de trabalho que, muitas vezes, não são contadas pelos usuários do sistema.

O cliente necessita de um software para atender à crescente demanda por plantas ornamentais, pois todo o processo, com exceção ao realizado pela máquina de cartão de crédito, é feito manualmente e resulta em anotações em muitos cadernos. A empresa possui como meta expandir os negócios através de franquias, mas, para isso, será necessário informatizar todo o processo de compra de plantas ornamentais e realizar um controle das encomendas realizadas por arquitetos e paisagistas. A empresa atende ao público em geral, de

modo que consumidores podem comprar (e/ou encomendar) diversos tipos de plantas.

Você, como analista de sistemas, precisa auxiliar o cliente no progresso dessa informatização. Para resolver o problema de falta de conhecimento sobre as rotinas de trabalho da empresa, você foi designado a elaborar uma apresentação com os seguintes questionamentos: o que precisará ser feito? Quais requisitos funcionais podem ser apontados nesta primeira etapa? Determine também requisitos não funcionais que contribuem para atender às expectativas do cliente, classificando-os em “requisitos funcionais” e “não funcionais”.

O primeiro passo para responder as questões acima é estudar esta seção, que contém conhecimentos acerca de procedimentos da engenharia de requisitos, de seus conceitos e tipos.

Bom trabalho e ótimos estudos!

Não pode faltar

Um dos maiores problemas no desenvolvimento de um software é estabelecer quanto tempo ele levará para ser concluído. Outro desafio é “garantir que o software desenvolvido para o cliente seja exatamente como ele deseja”. Tais dificuldades podem ser superadas com a engenharia de requisitos.

Como destaca Pressman (2016), essa área da Informática diz respeito a um conjunto de atividades que colaboram para a produção e a manutenção de um documento, também conhecido como documento de requisitos, o qual possui como principal meta a especificação do que deve ser implementado (antes de começar a desenvolver o software ou sistema). Entretanto, é comum empresas começarem a desenvolver partes do software antes mesmo de finalizar o levantamento de seus requisitos.

Conforme Sommerville (2011), engenharia de requisitos é o processo de descrever todas as funcionalidades que um sistema deve possuir, bem como descrever todos os serviços e as restrições de seu funcionamento, refletindo diretamente as determinações dos clientes (usuários do software projetado) e contribuindo, de forma significativa, para o desenvolvimento de um produto final com qualidade. O objetivo da engenharia de requisitos, para Pressman (2016), é proporcionar a todos os envolvidos no desenvolvimento do sistema uma mesma compreensão por escrito do problema; para isso é utilizada uma série de elementos (artefatos) que garante a qualidade do que foi especificado.

Mas, o que é o requisito de um sistema?

Há algum tempo, segundo Pfleeger (2004) e Robertson e Robertson (2006), requisito de sistema era definido como uma função que o software deveria ter ou uma qualidade que ele deveria apresentar. Atualmente, além da função e da qualidade, os requisitos de um sistema abrangem especificações dos serviços que ele deve fornecer, restrições que deve possuir, características gerais e restrições que devem ser atendidas no seu processo de desenvolvimento, como destaca Falbo (2012).

Refilita

Definir os requisitos de um sistema são fundamentais para o sucesso de um software. Será que existem requisitos que, se não forem prontamente atendidos, podem colocar em risco o sucesso do programa ao ponto de o cliente não utilizar o software desenvolvido?

Dando sequência a nossa tratativa sobre a engenharia de requisitos, suponha que você precise criar um jogo on-line de perguntas e respostas para treinar seus conhecimentos sobre o ENADE. Nessa situação, podemos elencar alguns exemplos de requisitos:

1. O jogador deverá realizar um cadastro antes de jogar, criando um apelido, senha, e-mail e escolher um avatar.
2. O jogo deverá ter várias fases, apresentando graus de dificuldade.
3. O jogador deverá escolher uma ou mais áreas de conhecimento.
4. As questões deverão ser classificadas em vários níveis de dificuldade.

Assimile

Os requisitos podem evoluir e podem ser modificados no decorrer do desenvolvimento do software, eles não são estáticos e sofrem atualizações constantes; além disso, devem ser documentados para fins de controle. Paula Filho (2019) destaca os seguintes fatores que contribuem para a evolução dos requisitos:

1. Descobrimento de defeitos e inconformidades nos requisitos originais.
2. Falta de detalhamento nos requisitos originalmente elencados.
3. Modificações incontornáveis no projeto (por exemplo, alterações de legislação, moedas, impostos).

Observe que os requisitos demonstrados estão escritos de modo que tanto o cliente e os desenvolvedores possam ter um entendimento claro e preciso do que o software deverá realmente fazer. Não convém criar os enunciados dos requisitos muito extensos e com muita subjetividade. Eles devem ser objetivos e consistentes, permitindo o entendimento do que será realizado por todas as partes envolvidas, o Quadro 1 apresenta as qualificações que os requisitos devem possuir.

Quadro 3.1 | Qualificação dos requisitos

QUALIFICAÇÃO	DESCRÍÇÃO
Exatidão	Todo requisito precisa ser um requisito do produto a ser desenvolvido.
Precisão	Todo requisito possui apenas uma única interpretação, aceita tanto pelos desenvolvedores quanto pelos clientes (usuários).
Completude	Todo requisito reflete as decisões de especificação que foram acordadas entre as partes envolvidas.
Consistência	Não pode haver conflitos entre os requisitos e qualquer um dos seus subconjuntos.
Priorização	Todo requisito é rotulado de acordo com a sua importância, estabilidade e complexidade.
Verificabilidade	Demonstra a conformidade do produto final com cada requisito elencado.
Modificabilidade	O requisito deve ter estrutura e estilo que permitam mudança de maneira fácil, completa e consistente.
Rastreabilidade	Permite a determinação dos antecedentes e das implicações de todos os requisitos.

Fonte: adaptado de Paula Filho (2019).

Destacar um fator no levantamento de um requisito é determinar a sua prioridade, a qual possuirá denominações diferenciadas de acordo com a literatura consultada, podendo cada empresa adotar a sua terminologia. Os requisitos são classificados como: essencial, importante ou desejável, conforme afirma Pfleeger (2004).

Um requisito classificado como **essencial** é de extrema importância (é fundamental) para o software ser executado, de maneira que sem esse requisito o programa ficará incompleto. O sistema não poderá ser implantado ou

concluído caso um requisito essencial não esteja totalmente realizado. Para exemplificar o requisito essencial, podemos pensar em um aluno que somente poderá acessar a disciplina de seu curso se estiver devidamente matriculado nela. Nesse caso, é evidente que o acesso do aluno às disciplinas ofertadas em seu curso deverão ser restritivas, isto é, ele somente poderá acessar se estiver matriculado, fator essencial na funcionalidade do software.

Um requisito classificado como **importante** é aquele que não é essencial para que o software seja implantado, ou seja, ele deve ser realizado, mas pode ficar em segundo plano; são requisitos desejáveis, porém não imprescindíveis. Pode-se considerar um exemplo de requisito importante a senha dos usuários do sistema acadêmico que deve ser criptografada. A criptografia, nesse caso, é importante, mas, caso não seja desenvolvida, ainda assim o sistema poderá ser implantado e depois atualizado com as modificações sobre a criptografia no acesso ao sistema.

Um requisito classificado como **desejável** é aquele que não é imprescindível para o software estar concluído, é algo opcional que pode ou não ser realizado e até pode ser eliminado no decorrer do desenvolvimento do sistema. Um exemplo disso é a contagem de tempo de cada acesso realizado pelo aluno no sistema acadêmico, o que serve para determinar por quanto tempo cada usuário utilizou-o durante o semestre. Esse é um requisito que, sendo classificado como deseável, pode eventualmente nem ser desenvolvido caso haja atrasos e seja necessária a implantação do software.

Assimile

O que é um escopo do projeto? Segundo o Guia PMBOK®, o escopo do projeto descreve de forma clara o que deverá estar incluso no projeto a ser desenvolvido e estabelece os limites das operações que serão realizadas juntamente com os recursos a serem utilizados. Definir o escopo do projeto é primordial para determinar seu orçamento e seu cronograma.

As prioridades de cada requisito devem ser determinadas durante a definição do escopo do projeto, entretanto vale ressaltar que essas prioridades podem mudar com a evolução do sistema.

Exemplificando

Os requisitos de um sistema são as especificações das propriedades que deve ter, das características do que deve fazer, de como deve se comportar e das suas restrições. Os requisitos descrevem o que o sistema deverá fazer e o que ele não deverá fazer, sem mencionar o

modo como será feito. Por isso, é importante escrever o enunciado de um requisito de forma correta.

Observe um exemplo incorreto de escrita:

“Uma fase do jogo deve ser apresentada ao jogador, entrando no arquivo: escolha de fase. Em seguida o jogador deverá realizar todas as etapas da fase; cada etapa deve ser apresentada ao jogador de forma que ele não possa escolher outra fase; assim que o jogador terminar a etapa, uma nova etapa deverá ser mostrada com dificuldade mais elevada”.

Uma forma mais abreviada e correta do mesmo requisito:

“O jogo deverá ter várias fases que apresentem graus de dificuldades de acordo com a evolução do jogador”.

Na engenharia de requisitos é importante fazer uma distinção dos diferentes tipos de descrições dos requisitos conforme afirma Sommerville (2011). Eles podem ser classificados como:

- **Requisitos de usuário:** descrevem os requisitos funcionais e os não funcionais do sistema, utilizando uma linguagem acessível aos usuários do sistema (clientes), e demonstram as funções e as restrições que o sistema deverá possuir. Como resultado será produzido um documento que não contém detalhamento técnico do sistema e possui como finalidade a comunicação entre os desenvolvedores e clientes.
- **Requisitos de sistema:** especificam detalhes do sistema (apoiados nos requisitos de usuários), resultando em um documento que pode servir como parte do contrato entre os envolvidos no desenvolvimento do sistema. É o ponto que marca a fase inicial de um projeto.

Assimile

Muitas vezes pode haver confusão entre os termos “requisitos” e “regra de negócios”. Os dois termos se referem a alguma necessidade do sistema, entretanto possuem finalidades distintas. Regras de negócios são restrições imperativas para o negócio existir e tratam sempre do contexto do sistema. A diferença básica entre os dois termos é que um requisito menciona **o que o sistema realizará** e a regra de negócio menciona **como o sistema realizará**. Os requisitos aparecem como ações claras e objetivas, sendo um pedido ou uma necessidade do sistema. Na regra de negócio sempre existe algum tipo de condição expressa por termos como: se, quando, somente se, é obrigatório testar, sempre que, etc.

Determinar o tipo de requisito em um sistema pode ser uma tarefa extenuante, e um requisito pode ser classificado inicialmente como de um tipo e, no decorrer no projeto, pode sofrer alterações e receber outra classificação, podendo gerar, ainda, uma série de novos requisitos.

Segundo Sommerville (2011) e Pressman (2016), os requisitos de um sistema podem ser classificados como **requisitos funcionais**, **requisitos não funcionais** e **requisitos de domínio**:

- Requisitos funcionais: determinam de forma clara e precisa as funcionalidades específicas do que o sistema deve ou não realizar.
- Requisitos não funcionais: estabelecem restrições sobre as funcionalidades do sistema, por meio do estabelecimento de padrões específicos de desenvolvimento, plataforma, tempos de resposta e restrições de acessos; estão relacionados às qualidades que o sistema deverá apresentar.
- Requisitos de domínio: determinam as características do domínio do sistema, refletindo as características do sistema; podem estabelecer restrições aos requisitos funcionais ou podem indicar cálculos específicos sobre determinado requisito.

Os **requisitos funcionais** determinam os objetivos específicos do sistema, ou seja, o que ele deve possuir ao final de seu desenvolvimento, segundo afirma Sommerville (2011). Esse tipo de requisito deverá conter todas as funções e informações fornecidas pelo cliente antes da construção do software. Usamos a sigla RF (requisito funcional) seguida de uma numeração para indicar o requisito. Os exemplos de requisitos funcionais de um sistema de controle acadêmico hipotético são apresentados a seguir:

- [RF0001] – O sistema deve manter os dados pessoais e acadêmicos dos alunos.
- [RF0002] – O sistema deve permitir que o aluno faça a matrícula por disciplina.
- [RF0003] – O sistema deverá manter os dados (pessoais e profissionais) dos professores, especialmente dos seguintes atributos: CPF, RG, nome, endereço (completo), data de nascimento, telefones para contato, e-mail (pessoal e corporativo), nacionalidade, data de admissão, data de demissão, valor da hora-aula, carga horária.
- [RF0004] – O sistema deve permitir a visualização das notas cadastradas.

- [RF0005] – O professor poderá cadastrar as notas no sistema de acordo com o calendário acadêmico (previamente cadastrado na plataforma).

O grau de detalhamento de um requisito ajuda na comunicação com o cliente no momento de tirar dúvidas sobre o sistema (estabelecendo o que realmente deverá ser realizado). Auxilia também na hora de programar o requisito, uma vez que o programador irá se guiar pelo requisito ao programar a funcionalidade do sistema. Dessa forma, é essencial um detalhamento do que deverá ser realizado (não deixando margens para dúvidas). Observe que no requisito funcional [RF0003] há um detalhamento maior em relação ao [RF0001]. O detalhamento é importante nessa fase, porém deve-se ter em mente que cada requisito listado deverá ser específico, ou seja, se estamos falando de informações dos alunos, não podemos, no mesmo requisito, tratar dos dados dos professores.

Os **requisitos não funcionais** possuem a sigla RFN e especificam critérios para qualificar os funcionais. Esse tipo de requisito aponta para questões relacionadas à qualidade, performance, confiabilidade, usabilidade, implementação, etc. Um requisito não funcional pode gerar uma grande quantidade de requisitos funcionais. Observe alguns exemplos de um sistema acadêmico hipotético:

- [RNF0001] – O tempo de espera do aluno para visualizar as notas, não poderá exceder os sete segundos.
- [RNF0002] – O sistema deverá ser implementado utilizando a linguagem de programação JAVA.
- [RNF0003] – As notas só poderão ser lançadas por profissionais da empresa com o perfil de professor.
- [RNF0004] – Todas as cores do sistema deverão obedecer ao padrão de cores da instituição: laranja (RGB: 255,140,0), cinza (RGB: 211,211,211) e branco (RGB: 248,248,255).
- [RNF0005] – Todos os relatórios devem ser gerados com as cores do sistema, logomarca da instituição, com data e hora da geração; devem ser gerados no formato PDF.

Uma recomendação importante para saber se a especificação do requisito é um requisito não funcional é observar se o assunto tratado pode ser mensurado (velocidade, tempo, linguagens, versões); caso seja, possivelmente será um requisito não funcional, conforme Paula Filho (2019). Observe o Quadro 3.2 com métricas que auxiliam na identificação e especificação de requisitos não funcionais.

Quadro 3.2 | Métricas de requisitos não funcionais

Propriedade	Métrica
Velocidade	Quantidade de transações realizadas por segundo. Tempo de resposta ou atualização do sistema.
Tamanho	Megabytes ou gigabytes ocupados. Quantidade de RAM. Quantidade de espaço em disco ou nuvem.
Usabilidade	Tempo de treinamento (horas necessárias). Quantidade de telas de ajuda.
Confiabilidade	Tempo médio para falhar. Probabilidade de indisponibilidade. Taxa de ocorrência de falhas. Disponibilidade.
Robustez	Percentual de eventos que causam falhas e o tempo necessário para o sistema se reestabelecer. Determinar as possibilidades de que os dados sejam corrompidos.
Portabilidade	Quantidade de adaptações para o sistema funcionar em diferentes plataformas (Sistema Operacional).

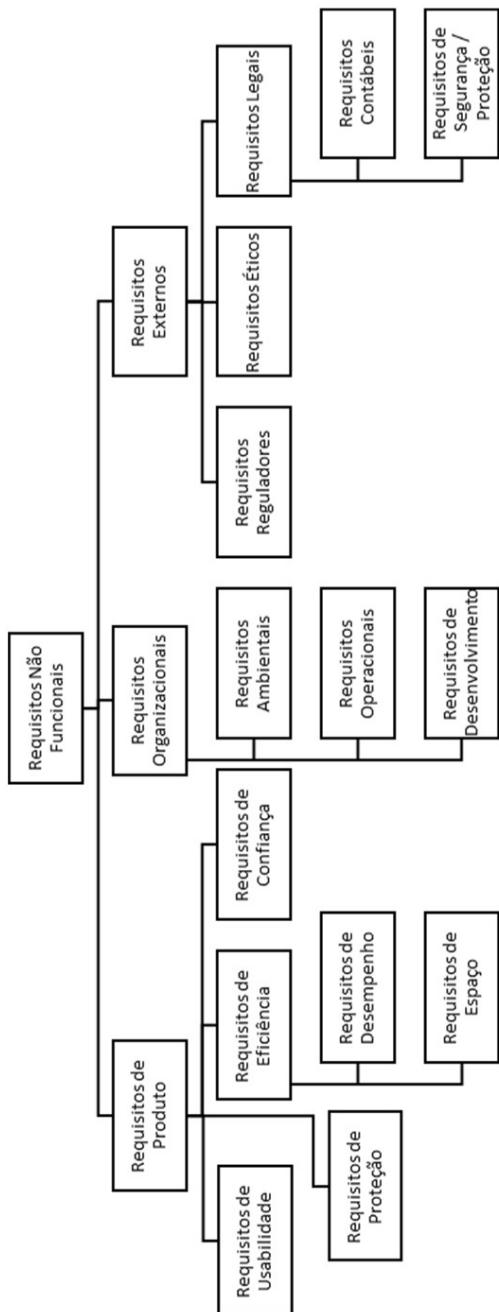
Fonte: adaptado de Sommerville (2011).

Os requisitos não funcionais possuem uma classificação gerada a partir de três grandes áreas de requisitos:

1. Requisitos de produto: especificam ou restringem o comportamento do sistema e podem determinar a linguagem de programação a ser utilizada.
2. Requisitos organizacionais: são derivados das políticas e procedimentos da empresa, do cliente e do desenvolvedor.
3. Requisitos externos: abrangem todos os fatores externos ao sistema e que podem influenciar no desenvolvimento do sistema sem desobedecer a legislação vigente.

A classificação completa dos requisitos não funcionais pode ser observada na Figura 3.1.

Figura 3.1 | Classificação dos requisitos não funcionais



Fonte: adaptada de Sommerville (2011).

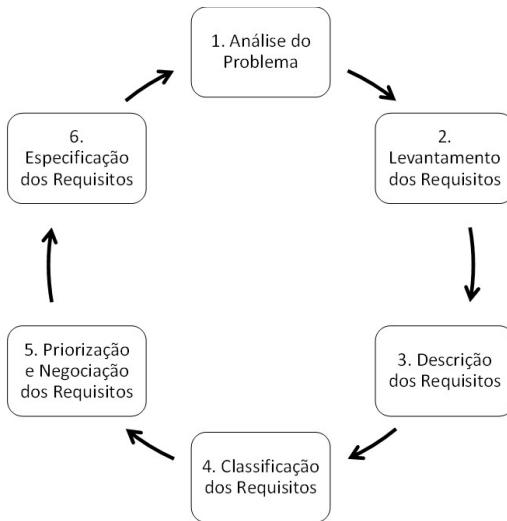
Refita

Erros e esquecimentos podem existir em quaisquer softwares; mas, qual dos tipos de requisitos, funcional ou não funcional, causaria danos irreparáveis no sistema? Você saberia apontar um determinado requisito que, se fosse ignorado, tornaria o sistema simplesmente inútil para o cliente?

Conforme aponta Sommerville (2011), os requisitos de domínio descrevem as características e estabelecem ressalvas aos requisitos funcionais, indicando, por exemplo, um cálculo obrigatório para o requisito ser validado. Eles podem ser novos requisitos funcionais ou alguma restrição (complementar) de algum requisito funcional. Um exemplo de requisito de domínio seria a seguinte situação: o aluno será considerado aprovado se obtiver, no mínimo, 3000 pontos nas avaliações semestrais e 4000 pontos nas atividades complementares de cada disciplina. Observe o que está sendo determinado: pontuação mínima das avaliações deve ser igual a 3000 pontos. E pontuação mínima deve ser igual a 4000 pontos. As duas condições deverão ser verdadeiras para que o aluno esteja aprovado no sistema.

De acordo com Pressman (2016), as atividades do processo da engenharia de requisitos envolvem a coleta de informações sobre o software (sistema) a ser realizado, a análise do problema e, em seguida, a descrição dos requisitos, classificação e priorização, sendo que logo após isso é gerada a especificação desses requisitos. Tais processos podem ser melhor visualizados na Figura 3.2.

Figura 3.2 | Atividades dos processos de engenharia de requisitos



Fonte: adaptada de Pfleeger (2004) e Sommerville (2011).

Todas as atividades que englobam a engenharia de requisitos possuem como finalidade a produção de um documento de requisitos, como afirmam Sommerville (2011) e Pressman (2016), englobando as seguintes etapas:

1. **Concepção:** determina-se o escopo geral do sistema e todos os envolvidos.
2. **Elicitação:** faz-se o levantamento dos requisitos funcionais e não funcionais, utilizando técnicas como entrevistas, observação e pesquisa.
3. **Elaboração:** detalha-se cada requisito (levantado e escrito em linguagem natural) para transformá-los em modelos conceituais (UML – Linguagem Unificada de Modelagem), eliminando erros, esquecimentos, duplicidades e inconsistências.
4. **Negociação:** identificam-se os requisitos conflitantes, sendo realizada uma negociação entre as partes envolvidas para modificar e/ou eliminar o requisito.
5. **Especificação:** transformam-se os requisitos ao se observar a visão do sistema (do desenvolvimento da solução).
6. **Validação:** realiza-se a homologação dos requisitos pelos usuários envolvidos, verificando se todos os requisitos foram atendidos (na visão do cliente).
7. **Gerenciamento:** é a verificação constante (no decorrer dos processos) de que os requisitos estão de acordo com o escopo do projeto e a garantia da rastreabilidade, que é o gerenciamento das eventuais modificações que um requisito pode sofrer.

Essa etapa permeia todo o ciclo de vida do produto e consiste em dois aspectos fundamentais. Várias atividades envolvem o processo de engenharia de requisitos. A partir de uma análise inicial do problema do cliente, na qual é realizada a compressão do domínio do problema, realiza-se a fase do levantamento dos requisitos, atividade que dará suporte a todas as etapas da engenharia de requisitos. Após isso, realiza-se a classificação, a priorização e a documentação dos requisitos. A participação do cliente e dos usuários finais do sistema é fundamental para a validação dos requisitos levantados. Deve-se também, inserir atividades de controle de qualidade em todas as etapas para validar os requisitos e garantir a qualidade do que está sendo projetado. E, como tudo é dinâmico em um projeto de software, os requisitos podem sofrer alterações, sendo necessário um processo de gerenciamento para validar as evoluções dos requisitos funcionais e não funcionais a fim de manter o controle dessa evolução.

Resumidamente, os requisitos funcionais explicitam as funções, os comportamentos e as propriedades de entrada e saída de um sistema, e os requisitos não funcionais referem-se aos atributos (qualidades) do sistema.

Os requisitos são a essência de qualquer software. Antes de sair desenvolvendo algum sistema, é necessário criar uma lista de funcionalidades e características que ele deve possuir (esses são os requisitos iniciais), mas não paramos nessa etapa. Por isso, convido você a se aprofundar ainda mais na engenharia de requisitos. Veremos ainda que é preciso eliciar e especificar os requisitos, além de negociar, monitorar e validar os requisitos funcionais e não funcionais.

Bons estudos!

Sem medo de errar

Retomando a situação-problema, você foi escolhido para fazer parte da equipe de analistas cuja meta é elencar os requisitos de um software para um cliente, dono de uma empresa de plantas ornamentais que possui como objetivo expandir os negócios através de franquias. Para isso, será necessário informatizar todo o processo de compra de plantas ornamentais dos fornecedores, controlar as encomendas realizadas por arquitetos e paisagistas e, como a empresa também fornece o serviço de decoração paisagista (doméstico e empresarial), é necessário também um controle desses processos. A empresa atende ao público em geral, de modo que consumidores podem comprar (e/ou encomendar) diversos tipos de plantas. Assim sendo, alguns questionamentos foram realizados e precisam ser respondidos.

Para resolver o problema de falta de conhecimento sobre as rotinas de trabalho da empresa, o que você precisará fazer?

Para solucionar este problema, o primeiro passo é visitar o cliente e conversar com os usuários que utilizarão o software. Desta forma, poderemos conhecer a rotina da empresa, conversando com as pessoas, ficando atentos às atividades que elas realizam, sendo curiosos e perguntando sempre que houver dúvidas. Dica: anotar tudo e, se possível, recolher cópias de documentos que são utilizados.

Durante as visitas, foram realizadas várias entrevistas e foram coletados vários documentos, como listas de anotações de pedidos aos fornecedores, encomendas de clientes e a lista de plantas disponíveis para a venda.

Quais requisitos funcionais podem ser apontados nesta primeira etapa?

Aqui é preciso determinar requisitos não funcionais que contribuem para atender às expectativas do cliente, classificando-os em requisitos funcionais e não funcionais.

Após as visitas realizadas, pode-se listar alguns requisitos funcionais:

- [RF0001] – O sistema deverá manter (incluir, consultar, alterar e excluir) os dados dos clientes: nome, endereço, CPF ou CNPJ, e-mail, telefones.
- [RF0002] – O sistema deve gerar um relatório das plantas disponíveis para venda.
- [RF0003] – O sistema deve permitir que sejam cadastrados os pedidos de encomenda de plantas e produtos da empresa.
- [RF0004] – O sistema deve autorizar a inclusão de pedidos de jardinagem, permitindo a escolha das plantas e demais produtos.
- [RF0005] – O sistema deverá manter o cadastro de todos os produtos da empresa.
- [RF0006] – O sistema deverá manter as informações botânicas das plantas.
- [RF0007] – O sistema deverá emitir relatórios sobre as encomendas, situação do estoque, trabalhos paisagísticos em andamento e finalizados.
- [RF0008] – O sistema deverá manter o cadastro de todos os fornecedores.
- [RF0009] – O sistema deverá emitir um relatório das plantas que podem ser plantadas, sendo estas classificadas por época do ano, tipos de terrenos e tipos de construção.
- [RF0010] – O sistema deverá manter um cadastro dos usuários do sistema, classificando-os como: vendedor, administrador e supervisor.

Pode-se elencar os seguintes requisitos não funcionais:

- [RNF0001] – O sistema deverá ser realizado na linguagem de programação JAVA.
- [RNF0002] – O tempo de espera dos relatórios não poderá ultrapassar os seis segundos.
- [RNF0003] – As telas do sistema devem ter cores claras (tom pastel) e ícones grandes.

- [RNF0004] – O sistema deverá utilizar o banco de dados MySQL.
- [RNF0005] – Apenas usuários classificados como supervisor e administrador poderão gerar os relatórios sobre as encomendas.

Observe que os requisitos funcionais sempre aparecem em maior quantidade do que os requisitos não funcionais. Isso é natural, até porque qualquer software deverá ter muitas funcionalidades e todas devem ser descritas de forma individualizada. Foram listados alguns requisitos, mas será que você não consegue identificar mais alguns?

Avançando na prática

Classificação de requisitos não funcionais

Um cinema localizado em uma cidade do interior precisa informatizar seus serviços, porém, como não tem muitas condições financeiras, não conseguirá comprar licença de software. O espaço conta com duas salas pequenas, que podem ser alugadas para passar filmes, e seus funcionários são pessoas acima de 60 anos. Uma das preocupações do cliente é que não haja filas de espera na hora da compra de bilhete. Existe a intenção de disponibilizar, na internet, a possibilidade de reservar as salas de cinemas para clientes previamente cadastrados por meio de *login* e senha.

Com a descrição do sistema do cinema, liste os requisitos não funcionais e classifique-os quanto ao seu tipo.

Resolução da situação-problema

Para resolver essa situação-problema iremos separar os tipos de requisitos não funcionais encontrados:

Usabilidade:

[RNF0001] – O sistema deverá ser responsivo, pois será utilizado em ambiente Web.

[RNF0002] – Os ícones e as letras devem ter fonte de tamanho 16.

Proteção:

[RNF0003] – Autenticação de usuários antes de efetuar uma reserva.

Desempenho:

[RNF0004] – O tempo de espera para a impressão do ingresso do cinema não poderá passar dos seis segundos.

[RNF0005] – O sistema deverá operar mesmo se estiver sem acesso à internet, possibilitando a venda de ingressos.

Desenvolvimento:

[RNF0006] – O sistema deverá ser feito em JAVA e com banco de dados MySQL.

Agora é com você: liste mais alguns requisitos não funcionais e liste também os requisitos funcionais a fim de realizar o software para o cinema.

Faça valer a pena

1. Os requisitos funcionais são a base para criar as estimativas de preço e de tempo (quanto tempo levará o desenvolvimento do software), do processo de modelagem (pois tudo que será realizado dependerá dos requisitos) e de implementação (será programado o que for solicitado nos requisitos). Em suma, os requisitos servem de base para todo o ciclo de vida de um software.

Os requisitos de um software são fundamentais para determinar seu sucesso, visto que eles atendem:

- a. As necessidades dos analistas de sistemas, que precisam cumprir metas de entregas aos clientes, gerando menos expectativa de espera pelo software desenvolvido.
- b. As qualidades do software que foram inseridas nos requisitos funcionais e que devem ser desenvolvidas pelos programadores.
- c. Aos objetivos pelos quais foi desenvolvido o software, de modo que deve estar totalmente de acordo com as solicitações dos clientes.
- d. As expectativas dos programadores em um desenvolvimento de acordo com as funcionalidades propostas exclusivamente pelos analistas de sistemas.
- e. Aos escopos propostos pelos clientes, os quais fundamentam toda a base dos requisitos não funcionais propostos pelos analistas de sistemas e pelos programadores.

2. Os requisitos servem de guia para todas as atividades do projeto e do desenvolvimento de um sistema e são escritos em linguagem natural para que todos os envolvidos entendam de forma clara e objetiva o que será desenvolvido.

Analise as afirmativas a seguir sobre os requisitos:

- I. Gerenciar pedidos, manter clientes e emitir relatórios mensais são exemplos de requisitos funcionais.
- II. O analista de sistemas é o único que deve identificar e criar uma lista de requisitos do sistema.
- III. Durante a atividade de obtenção ou detalhamento de requisitos, seja em entrevista ou observação do processo, o profissional deve estar atento para identificar, nas entrelinhas, as regras de negócio ocultas.
- IV. Requisito é algo que o usuário solicita explicitamente sobre algum detalhe do sistema.

Com base no contexto apresentado sobre os requisitos de um sistema, é correto o que se afirma em:

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

3. Na engenharia de requisitos existem vários processos que devem ser realizados para testar se os requisitos apontados estão de acordo com as necessidades do cliente. Uma das etapas fundamentais é classificá-los em requisitos funcionais e não funcionais.

Sobre os requisitos não funcionais de software, analise as afirmativas:

- I. São medidas pautadas em comportamentos do sistema durante a execução das funcionalidades.
- II. Podem ser classificados como requisitos não funcionais: usabilidade, eficiência, confiança e desenvolvimento.
- III. Definem os parâmetros de funcionamento do sistema que trarão ao usuário uma melhor experiência no seu uso, porém não são direta-

mente acionados por ele, já que, às vezes, esses requisitos estão implícitos.

Analisando cuidadosamente as afirmativas apresentadas, é correto o que se afirma em:

- a. II e III, apenas.
- b. I e II, apenas.
- c. I e III, apenas.
- d. III, apenas.
- e. I, II e III.

Seção 2

Elicitação, especificação e validação de requisitos

Diálogo aberto

Olá, seja bem-vindo!

O processo de desenvolvimento de um software é muito dinâmico, tudo pode mudar rapidamente. Um cliente pode mudar de ideia e esperar que o software a ser desenvolvido tenha características diferentes do início do desenvolvimento. Precisamos deixar o cliente satisfeito, porém, nós, como desenvolvedores, precisamos garantir que os requisitos sejam executados conforme o contratado.

Você sendo detalhista em requisitos de software ficou encarregado do levantamento dos requisitos para o desenvolvimento de um software para um cliente que cultiva e comercializa plantas ornamentais. Foram realizadas diversas visitas ao cliente e foram identificados vários requisitos funcionais e não funcionais:

Requisitos funcionais:

[RF0001] – O sistema deverá manter (incluir, consultar, alterar e excluir) os dados dos clientes: nome, endereço, CPF ou CNPJ, e-mail, telefones.

[RF0002] – O sistema deve gerar um relatório das Plantas disponíveis para venda.

[RF0003] – O sistema deve permitir que sejam cadastrados os pedidos de encomenda de plantas e produtos da empresa.

[RF0004] – O sistema deve permitir a inclusão de pedidos de jardinagem, permitindo a escolha das plantas e demais produtos.

[RF0005] – O sistema deverá manter o cadastro de todos os produtos da empresa.

[RF0006] – O sistema deverá manter as informações botânicas das plantas.

[RF0007] – O sistema deverá emitir relatórios sobre as encomendas, situação do estoque, trabalhos paisagísticos em andamento e finalizados.

[RF0008] – O sistema deverá manter o cadastro de todos os fornecedores.

[RF0009] – O sistema deverá emitir um relatório das plantas que podem ser plantadas classificadas por época do ano, tipos de terrenos e tipos de construção.

[RF0010] – O sistema deverá manter um cadastro dos usuários do sistema, classificando-os como: Vendedor, Administrador e Supervisor.

Requisitos não funcionais:

[RNF0001] – O sistema deverá ser realizado na linguagem de programação JAVA.

[RNF0002] – O tempo de espera dos relatórios não poderá ultrapassar os 6 segundos.

[RNF0003] – As telas do sistema devem apresentar cores claras (tom pastel) e ícones grandes.

[RNF0004] – O sistema deverá utilizar o banco de dados MySQL.

[RNF0005] – Apenas usuários classificados como Supervisor e Administrador poderão gerar os relatórios sobre as encomendas.

Seu desafio é determinar como será feita a coleta de dados para obter o máximo de informações sobre o software a ser desenvolvido para o controle de venda de plantas ornamentais. Quais os passos para realizar a elicitação dos requisitos? De que forma os requisitos elencados (os citados e os novos que você inseriu) podem ser especificados?

Elabore uma tabela com os requisitos encontrados (e supralistados) e adicione novos requisitos funcionais e não funcionais. Prepare uma apresentação para demonstrar e debater com o cliente sobre os requisitos encontrados.

Para ajudar a solucionar a situação-problema, convido a estudar esta seção, que trata da especificação dos requisitos de um software, com o objetivo principal de demonstrar os processos de: elicitação de requisitos, especificação de requisitos, negociação e monitoramento de requisitos e validação de requisitos.

Um analista de sistemas precisa se aprofundar nos processos da Engenharia de Software. Cada dia é necessário um aprendizado novo, agregando novos conhecimentos e aperfeiçoando as técnicas utilizadas. Esse processo de crescimento profissional é desafiador e recompensador.

Vamos ao trabalho?

Bons estudos!

Não pode faltar

Engenharia de Requisitos determina todo o processo de definição dos requisitos de um sistema e tudo começa através da elicitação de requisitos. Segundo consta no dicionário Aurélio (1999) “elicitar” significa “extrair” ou “extrair algo de”. Na Engenharia de Requisitos, a elicitação de requisitos é descobrir (extrair de algo ou alguém) o máximo de informações para estabelecer os requisitos de determinado sistema, sendo essa uma das primeiras etapas da Engenharia de Requisitos, conforme definem Pressmann e Maxim (2016).

O analista de sistemas não faz a elicitação de requisitos sozinho; esse processo envolve diversas pessoas – todos os envolvidos –, conhecidas como *stakeholders*. De acordo com Pressman e Maxim (2016) a elicitação de requisitos combina elementos para solucionar algum problema e para isso é necessária uma abordagem colaborativa dos envolvidos. Sommerville (2011) destaca que os *stakeholders* são as partes envolvidas (as que têm interesse) no projeto e podem ser compostas de: analista de sistemas, gerente de projetos, programadores, cliente (o contratante), usuários finais do sistema.

No processo de elicitação de requisitos é fundamental ter visões diferentes das funcionalidades do sistema e os *stakeholders* devem fazer parte da elicitação para verificar se não há requisitos contraditórios (que caso não sejam resolvidos antes do desenvolvimento, podem ocasionar problemas na hora da entrega do produto desenvolvido). A partir dos problemas e das necessidades dos *stakeholders*, a elicitação de requisitos visa realizar as seguintes tarefas, conforme afirmam Pressmann e Maxim (2016):

- Especificar o domínio do problema do sistema.
- Verificar as possibilidades de reutilização de alguma solução já realizada.
- Identificar os *stakeholders* diretamente envolvidos pelo sistema.
- Elicitar e qualificar os requisitos do sistema.
- Análise dos requisitos elicitados.
- Validação dos requisitos elicitados.

A Engenharia de Requisitos é um processo que compreende todas as atividades que contribuem para a produção de um documento de requisitos e sua manutenção ao longo do tempo. Os procedimentos básicos de levantamento e análise de requisitos de um sistema, propostos por Sommerville (2011), contêm as seguintes tarefas:

1. **Concepção ou compreensão do domínio:** é estabelecer um entendimento básico sobre o problema a ser resolvido. Todos os *stakeholders* devem ter a compreensão exata do que será realizado e os limites (do que será e do que não) que serão realizados.
2. **Coleta de requisitos e classificação dos requisitos (elicitação):** são todas as atividades para conseguir elencar o máximo de requisitos dos *stakeholders*. Os requisitos são classificados em funcionais ou não funcionais.
3. **Negociação dos requisitos:** nesse processo (já com os requisitos listados) é fundamental determinar se há requisitos em conflito com outros, por exemplo:
 - RF0025 – O cliente precisa ter uma senha para entrar no sistema.
 - RF0078 – É desnecessário que um cliente precise de senha para entrar no sistema.

Observe que o requisito RF0078 é totalmente contrário ao requisito RF0025, e o que fazer? Provavelmente na coleta dos requisitos um *stakeholder* solicitou acesso com senha para os clientes, porém outra pessoa envolvida com o projeto sinalizou que era desnecessário o uso de senha. O procedimento para resolver esse tipo de impasse é chamar as partes envolvidas para resolver este conflito, realizando as seguintes análises:

- Verificação de requisitos: todos os requisitos são analisados e verificados se estão de acordo com as exigências dos *stakeholders*.
 - Definição das prioridades: é a determinação dos requisitos mais importantes e que merecem total atenção para o sucesso do projeto.
4. **Validação dos requisitos:** é realizada uma verificação geral dos requisitos, proveniente de um “termo de aceite” em que todas as partes envolvidas (os *stakeholders* do sistema projetado) concordam e validam os requisitos.
 5. **Documentação:** resultado final da Engenharia de Requisitos, quando é gerada uma documentação com todas as especificações dos requisitos funcionais e requisitos não funcionais, é o principal meio de comunicação entre o analista de sistemas ou engenheiro de software e o cliente.

A Figura 3.3 apresenta o processo de levantamento e análise de requisitos. Observe que a partir da compreensão do domínio do que será efetivamente realizado no sistema, é realizada a coleta e a classificação de requisitos, com a finalidade de determinar o que realmente será realizado no sistema,

classificando para melhor controle. A negociação serve para estabelecer os ajustes necessários e ajudar a determinar o que será prioritário no desenvolvimento. A especificação é o início da documentação dos requisitos, e na validação é realizada uma checagem geral de todos os requisitos, tendo como objetivo a documentação de requisitos.

Figura 3.3 | Levantamento e análise de requisitos



Fonte: elaborada pela autora.

O levantamento e análise de requisitos, proposto por Sommerville (2011), é um processo de validação continuada. A Figura 3.3 demonstra essa sequência dos processos, entretanto, basta alguma alteração em um requisito para que o ciclo deve ser repetido, de forma iterativa e contínua, até a finalização do projeto.

O processo de elicitação de requisitos pode ser diferente entre as empresas, em que cada empresa dispõe de seus métodos e processos de elicitação. Segundo Sommerville (2011), podemos destacar como atividades do processo de elicitação de requisitos:

- 1. Descoberta de requisitos:** com participação ativa dos *stakeholders*, é um processo interativo, e os usuários finais do sistema devem entrar com sua expertise para ajudar na coleta dos requisitos do sistema.
- 2. Classificação e organização de requisitos:** é realizado um agrupamento dos requisitos para descobrir se eles repetem ou se fazem parte de subsistemas.
- 3. Priorização e negociação de requisitos:** juntamente com os *stakeholders* é estabelecida a priorização de cada requisito (essencial, importante ou desejável), servindo para definir etapas mais prioritárias do desenvolvimento do sistema.
- 4. Especificação de requisitos:** nessa etapa é realizada a documentação dos requisitos, e esses documentos podem ser formais e informais.

A elicitação de requisitos tem por objetivo conseguir o máximo de requisitos do sistema a ser desenvolvido, destacando as seguintes técnicas:

- Pesquisa:** envolve a observação de como funciona a rotina dos processos do sistema e de outros softwares utilizados, visando

procurar requisitos que não foram explicitamente solicitados. Envolve também a pesquisa pela tecnologia solicitada.

- **Entrevista:** geralmente é guiada por um questionário para saber as necessidades que o sistema deverá suprir. Nessa etapa é importante saber ouvir e marcar o máximo de informações obtidas.
- **Reuniões:** usando técnicas como o brainstorming para descobrir requisitos que ainda não foram determinados e resolver requisitos conflitantes que apareceram nas entrevistas.
- **Documentos:** coleta de documentos que possam auxiliar na clareza das funcionalidades do sistema, como: relatórios, planilhas, papéis de controle, cadernos de anotações, etc.
- **Etnografia:** é a observação e análise de como os usuários finais realmente trabalham (diferente do que venham a dizer), gerando requisitos importantes para o sistema.

Refita

Estabelecer uma boa comunicação na elicitação de requisitos é fundamental para se ter sucesso. Como podemos determinar uma boa comunicação entre todas as partes envolvidas, estabelecendo um entendimento comum? Como diminuir os problemas de comunicação entre todos os *stakeholders*?

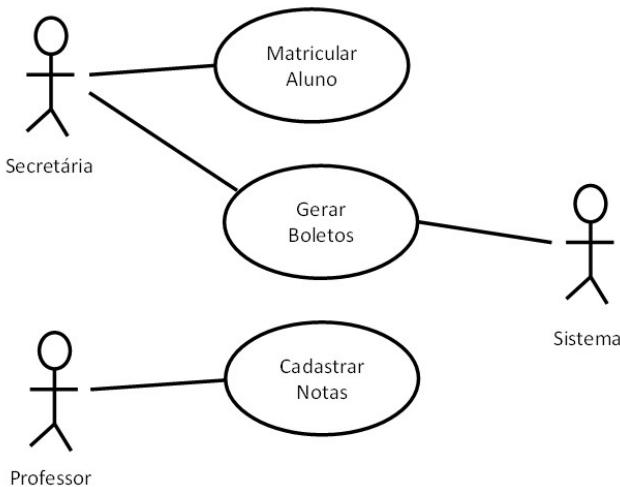
Sommerville (2011) enfatiza que a especificação de requisitos é a sistematização e abstração do que o software deverá realizar a partir de descrições claras e objetivas. Essa sistematização procura caracterizar o problema a ser solucionado e como resultado, e então é gerado um documento de especificação de requisitos. Pfleeger (2004) afirma que a melhor forma de começar a especificação de requisitos é usando a forma hierárquica, começando dos atributos gerais do sistema, passando por subníveis até chegar aos atributos mais específicos.

A especificação de requisitos é o meio de comunicação entre o analista de sistemas e os programadores que desenvolverão o software. É preciso especificar os requisitos de forma que não haja duplicidade de interpretações, pois o programador utilizará a especificação gerada para programar exatamente o que estará na especificação. A especificação de requisitos descreve todas as funcionalidades e suas restrições dos requisitos funcionais e dos requisitos não funcionais (geralmente em tabelas ou documentos específicos) e pode utilizar diagramas de caso de uso para ajudar na comunicação ou ainda fazer uso da prototipagem.

Os casos de uso são diagramas que compõem a Linguagem Unificada de Modelagem, conhecida como UML (*Unified Modeling Language*). Usada de forma simplificada, é uma ótima ferramenta de comunicação nas atividades do desenvolvimento, conforme afirma Sommerville (2011).

Na Figura 3.4 pode-se observar um diagrama de caso de uso que identifica de forma simples os atores que fazem parte de um sistema acadêmico genérico. De acordo com Engholm Jr (2010), os atores podem ser encontrados ao se procurar por usuários (ou grupos de usuários) que vão interagir com o sistema. Na Figura 3.4, os atores de um sistema acadêmico genérico são: a Secretaria, o Professor e o Sistema (nesse exemplo, o ator sistema está fazendo referências independentes de checagem e envio do boleto, entre-tanto, quando aparecer um ator Sistema, poderá ser referência a um outro sistema como: um sistema financeiro, um sistema de RH, etc. As linhas fazem a ligação com cada processo representado pela elipse.

Figura 3.4 | Casos de uso de um sistema acadêmico genérico



Fonte: elaborada pela autora.

A prototipagem, como afirma Paula Filho (2019), é a criação de uma versão menor do sistema a ser desenvolvido e tem como princípio a verificação de custo-benefício, em que a experiência do usuário é uma parte fundamental do desenvolvimento do protótipo.

Paula Filho (2019) destaca algumas vantagens que podem ser listadas na técnica da prototipagem: (i) possibilita a descoberta de problemas com antecedência; (ii) permite a verificação que os requisitos do software

satisfazem às necessidades dos clientes; (iii) melhora a comunicação com o cliente ao apresentar o progresso do software; (iv) possibilita a realização de testes para verificar a aceitação do software por parte do cliente, junto com o feedback do cliente.

Na técnica da prototipagem é possível prever vários problemas, mensurando melhor o tempo para resolve-los. Pfleeger (2004) descreve que a prototipagem pode ser classificada em três abordagens:

- **Protótipo descartável:** é desenvolvido para se aprender sobre o problema a ser resolvido e se explorar mais sobre a viabilidade da solução, não é utilizado posteriormente (só no nível exploratório) e é descartado.
- **Protótipo evolutivo:** é criado quando os clientes estão em dúvida sobre alguma parte do software (geralmente a interface gráfica). Esse tipo de protótipo é utilizado como parte do software final.
- **Prototipagem rápida:** são construídas partes do software para verificar a viabilidade dos requisitos. Nessa abordagem o objetivo é verificar se a realização do projeto é viável.

Exemplificando

A prototipagem é muito utilizada em desenvolvimento de aplicativos móveis (apps). Com um protótipo é possível fazer testes com os usuários para determinar as funcionalidades do aplicativo, juntamente com a interface gráfica. Os benefícios da prototipagem de aplicativos (e de softwares mais complexos) são:

- Testes das funcionalidades para verificar o que e de que forma precisa ser ajustado.
- Testes da usabilidade para verificar o que precisa ser melhorado na visão do cliente.
- Re却bimento de feedbacks do cliente, apontando o que gostou e o que não gostou.
- Redução de riscos, podendo-se testar a aceitação do aplicativo pelo público, disponibilizando funções básicas, diminuindo o investimento.
- Menor investimento, criando versões.

A negociação de requisitos tem como finalidade, conforme destacam Pressman e Maxim (2016), desenvolver um plano de projeto que atenda às demandas dos envolvidos (*stakeholders*) e, ao mesmo tempo, analisa as restrições no que diz respeito ao orçamento, pessoal, tecnologia e/ou tempo, impostas à equipe de desenvolvimento do software.

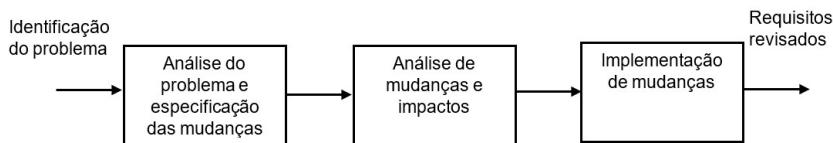
O monitoramento de requisitos é um processo que consiste em garantir que o escopo do software desenvolvido seja realizado. A cada alteração (em um ou mais requisitos) deve-se garantir a rastreabilidade das alterações, utilizando alguma ferramenta de controle, por exemplo: (i) determinar um status do requisito (proposto, em progresso, em alteração, adiado, excluído, aprovado, etc.); e (ii) criar uma matriz de rastreabilidade, para facilitar o gerenciamento dos requisitos, sendo que nessa matriz deverão constar todos os requisitos, suas dependências (quais requisitos dependem do requisitos em questão), o status do requisito, quem alterou o requisito, quem aprovou o requisito e, principalmente, as datas que esses fatos ocorreram.

Assimile

Pressman e Maxim (2016) definem rastreabilidade como sendo um termo da Engenharia de Software que se refere ao mapeamento documentados entre os artefatos projetados, por exemplo, entre os requisitos e os casos de uso. É necessário que, ao se alterar um requisito, a documentação gerada também seja alterada.

Toda mudança (por menor que seja) em um requisito deverá ser muito bem gerenciada para evitar duplicidade. Na Figura 3.5 é possível observar a gestão de mudança de requisitos, que é um processo que visa garantir a rastreabilidade das mudanças durante o processo de desenvolvimento do software, realizando análises de impacto das mudanças propostas para evidenciar sua viabilidade técnico-financeira, como destaca Sommerville (2011). Muitas vezes a pressa de alterar uma funcionalidade no sistema em desenvolvimento pode comprometer a gestão de mudanças de requisitos, pois é realizada a mudança e nem sempre é alterada a documentação dos requisitos, causando inconsistência e deixando a documentação dos requisitos defasadas.

Figura 3.5 | Gerenciamento de mudança de requisitos



Fonte: adaptada de Sommerville (2011).

O processo de validação dos requisitos determina que a especificação é consistente com a definição dos requisitos, assegurando que os requisitos propostos atenderão às necessidades impostas pelo cliente, de acordo com Pfleeger (2004).

Sommerville (2011) destaca que durante o processo de validação de requisitos podem existir diferentes tipos de verificação:

- Validação: verificar se os requisitos contemplam os objetivos do projeto.
- Consistência: garantir que um requisito não entre em conflito com outro requisito.
- Completude: deve haver requisitos que garantam as funcionalidades esperadas pelos usuários do sistema.
- Realismo: ter certeza que a tecnologia utilizada atenda as demandas do sistema projetado.
- Ambiguidade: o requisito não pode ter mais de uma interpretação.
- Verificável: o requisito deve ser passível de verificação através de testes.
- Rastreabilidade: cada requisito deve ter origem clara e bem definida.

O objetivo da validação de requisitos é encontrar erros nos requisitos documentados. Pressman e Maxim (2016) destacam que exemplos típicos de erros nos sistemas são inconsistências, contradições, duplidade, ambiguidades, incompletudes e imprecisões. Algumas perguntas podem ser feitas para validar os requisitos, segundo demonstram Pressman e Maxim (2016):

- Os requisitos estão de acordo com objetivos globais do sistema?
- O requisito foi bem especificado, fornecendo detalhes apropriados de forma clara e concisa?
- O requisito é realmente necessário? Está com a sua classificação correta?
- O requisito não está em conflito com outro requisito?
- O modelo de requisito reflete adequadamente a informação, comportamento e funcionalidade do sistema?

O grau das perguntas de Pressman e Maxim (2016) pode ser mais elaborado e aprofundado, conforme a dimensão do sistema a ser desenvolvido. Muitas vezes será necessária mais investigação, mais perguntas para determinar se os requisitos estão ou não corretos. Destaca-se que o fundamental na validação de requisitos é a eliminação de erros que podem ocasionar atrasos e aumentos de custos (tanto de produção do software quanto para o cliente).

Uma ferramenta que pode auxiliar na garantia da qualidade da validação de requisitos é checklist, uma lista de perguntas elaborada e que servirá para analisar cada requisito do sistema. Essa técnica visa: (i) a descoberta de erros

em vários níveis: função, lógica, implementação; (ii) a verificação se o sistema possui os requisitos especificados; (iii) a garantia de que o software desenvolvido foi implementado de acordo com padrões previamente impostos.

Exemplificando

A melhor maneira de manter a qualidade dos requisitos começa no processo de especificação dos requisitos, ajudando tanto nos processos de monitoração dos requisitos quanto nos processos de validação dos requisitos. Para se fazer uma boa especificação você deverá:

1. Separar as funcionalidades de implementação do sistema, visão do usuário final e outra para os programadores (orientada ao processo de desenvolvimento – geralmente detalhada com casos de uso).
2. A especificação do requisito deverá ser completa, ter precisão e ser consistente.
3. Cada especificação deverá estar dentro do escopo do projeto.
4. A especificação deve ser realista, tanto na capacidade técnica dos desenvolvedores quanto na tecnologia a ser empregada na solução.
5. Atenção ao enunciado do requisito, pois um enunciado mal escrito pode ocultar funcionalidades do sistema.
6. Uso de frases curtas, evitando ambiguidades e termos vagos.

Nesta seção, você estudou os conceitos sobre a elicitação, especificação, negociação e monitoramento e validação dos requisitos de um sistema. Os conceitos aprendidos darão suporte para a próxima etapa da Engenharia de Requisitos, que é o processo de documentação dos requisitos.

Sem medo de errar

O desafio desta seção é determinar como será feita a coleta de informações, para obter o máximo de informação sobre o software a ser desenvolvido para o controle de venda de plantas ornamentais. A meta agora é determinar como será realizada a coleta de dados para obter o máximo de informação sobre o software a ser desenvolvido. Quais os passos para realizar a elicitação dos requisitos? De que forma os requisitos elencados (os citados e os novos que você inseriu) podem ser especificados?

Para resolver essa situação-problema, primeiramente devemos observar os requisitos funcionais e os requisitos não funcionais elencados:

Requisitos funcionais:

[RF0001] – O sistema deverá manter (incluir, consultar, alterar e excluir) os dados dos clientes: nome, endereço, CPF ou CNPJ, e-mail, telefones.

[RF0002] – O sistema deve gerar um relatório das plantas disponíveis para venda.

[RF0003] – O sistema deve permitir que sejam cadastrados os pedidos de encomenda de plantas e produtos da empresa.

[RF0004] – O sistema deve permitir a inclusão de pedidos de jardinagem, permitindo a escolha das plantas e demais produtos.

[RF0005] – O sistema deverá manter o cadastro de todos os produtos da empresa.

[RF0006] – O sistema deverá manter as informações botânicas das plantas.

[RF0007] – O sistema deverá emitir relatórios sobre as encomendas, situação do estoque, trabalhos paisagísticos em andamento e finalizados.

[RF0008] – O sistema deverá manter o cadastro de todos os fornecedores.

[RF0009] – O sistema deverá emitir um relatório das plantas que podem ser plantadas classificadas por época do ano, tipos de terrenos e tipos de construção.

[RF0010] – O sistema deverá manter um cadastro dos usuários do sistema, classificando-os como: Vendedor, Administrador e Supervisor.

Requisitos Não Funcionais:

[RNF0001] – O sistema deverá ser realizado na linguagem de programação JAVA.

[RNF0002] – O tempo de espera dos relatórios não poderá ultrapassar os 6 segundos.

[RNF0003] – As telas do sistema devem apresentar cores claras (tom pastel) e ícones grandes.

[RNF0004] – O sistema deverá utilizar o banco de dados MySQL.

[RNF0005] – Apenas usuários classificados como Supervisor e Administrador poderão gerar os relatórios sobre as encomendas.

Dando continuidade:

1º Como será feita a coleta de dados para obter o máximo de informação sobre o software a ser desenvolvido e conseguir realizar a elicitação dos requisitos?

Para responder essa pergunta, primeiro teremos que analisar os requisitos elencados após as visitas realizadas no cliente e responder:

- Serão somente esses requisitos? Provavelmente não, pode haver uma lista maior de requisitos. Observe que não foi informado nada a respeito dos trabalhos paisagísticos realizados pela empresa. Será necessário guardar informações sobre esse assunto?
- As informações dos requisitos são suficientes? Provavelmente, a resposta também será não. Observe que foram mencionados nos requisitos anteriores: fornecedores, funcionários, plantas, encomendas, porém não há informação que complete o requisito. Procure por mais informações sobre os usuários que utilizarão o software.

A coleta deverá envolver: (i) **pesquisa**: examinando com os usuários envolvidos para obter mais informações sobre o que falta em cada requisito; (ii) **observação**: mesmo que algo não é solicitado explicitamente, você deverá observar para que não haja contradições entre o que é feito e o que dito; e (iii) **entrevistas**: converse informalmente com os futuros usuários procurando saber o que realmente é importante ter no sistema a ser desenvolvido e, caso necessário, providencie entrevistas estruturadas (já com as perguntas pré-realizadas).

2º Como especificar os novos requisitos funcionais e não funcionais elencados?

Você precisará adotar técnicas para especificar os requisitos. Analise as seguintes dicas:

1. Estimule a participação dos *stakeholders*.
2. Escreva os requisitos de forma simples e objetiva.
3. Evite o excesso de detalhamento nos requisitos, pois isso dificulta a leitura e a compreensão.
4. Tenha como foco resolver o problema do cliente.

Exemplo de uma boa especificação: [RF0001] – O sistema deverá manter (incluir, consultar, alterar e excluir) os dados dos clientes: nome, endereço, CPF ou CNPJ, e-mail, telefones.

Mais um exemplo: [RF0001] – O sistema deverá manter (incluir, consultar, alterar e excluir) os dados dos clientes: nome, endereço, CEP, CPF ou CNPJ, Inscrição Estadual, e-mail, telefone residencial, telefone comercial, celular, cidade, data de nascimento.

É uma boa prática criar tabelas separadas com os tipos de requisitos: uma para os requisitos funcionais e outra para os requisitos não funcionais.

Para os requisitos não funcionais é importante estabelecer a classificação do requisito.

Observe uma sugestão de tabela para os requisitos não funcionais:

Quadro 3.3 | Requisitos não funcionais

Identificador	Descrição	Classificação
RNF0001	O sistema deverá ser realizado na linguagem de programação JAVA.	Implementação
RNF0002	O tempo de espera dos relatórios não poderá ultrapassar os 6 segundos.	Desempenho
RNF0003	As telas do sistema devem apresentar cores claras (tom pastel) e ícones grandes.	Usabilidade
RNF0004	O sistema deverá utilizar o banco de dados MySQL.	Implementação
RNF0005	Apenas usuários classificados como Supervisor e Administrador poderão gerar os relatórios sobre as encomendas.	Segurança

Fonte: elaborado pela autora.

Agora é com você. Crie novos requisitos funcionais e não funcionais e analise. Prepare uma apresentação para demonstrar e debater com o cliente os requisitos encontrados.

Faça valer a pena

1. Sommerville (2011) destaca que os *stakeholders* são as partes envolvidas em um projeto, são as pessoas que têm algum interesse no sistema a ser desenvolvido. Para conseguir obter os requisitos de um sistema, os analistas de sistemas devem utilizar técnicas para obter o máximo de requisitos. Podemos citar alguns exemplos dessas técnicas utilizadas: (i) pesquisa: envolve a observação de como funciona a rotina dos processos do sistema; (ii) entrevista: geralmente é guiada por um questionário para saber as necessidades que o sistema deverá suprir; (iii) reuniões: uso de técnicas como o brainstorming para descobrir requisitos que ainda não foram determinados e resolver requisitos conflitantes; (iv) documentos: coleta de documentos que auxiliem na clareza das funcionalidades do sistema; (v) etnografia: é a observação e análise de como os usuários finais realmente trabalham.

Assinale a alternativa que apresenta corretamente em qual dos processos da Engenharia de Requisitos são utilizadas as técnicas apresentadas.

- a. Validação dos requisitos.
- b. Gerenciamento de requisitos.
- c. Implementação de requisitos.
- d. Prototipagem de requisitos
- e. Elicitação de requisitos.

2. Pressman e Maxim (2016) destacam que a negociação de requisitos tem como finalidade desenvolver um plano de projeto que atenda às demandas dos *stakeholders* do projeto e, ao mesmo tempo, reflita as restrições no que diz respeito ao orçamento, pessoal, tecnologia e/ou tempo, impostas à equipe de desenvolvimento do software.

Analise as afirmativas a seguir sobre a negociação de requisitos.

- I. *Stakeholders* do projeto são os analistas de sistemas, clientes e pessoas diretamente envolvidas ou com interesses no desenvolvimento do software.
- II. É na negociação de requisitos que são identificados os conflitos entre os requisitos.
- III. É na negociação de requisitos que são efetivadas a priorização de requisitos.

IV. A negociação de requisitos nem sempre é obrigatória, e cabe somente ao analista de sistemas a decisão do que será ou não realizado e priorizado.

Analise as alternativas sobre os requisitos de um sistema e marque a alternativa correta.

- a. Apenas as afirmativas I e II estão corretas.
- b. Apenas as afirmativas I e III estão corretas.
- c. Apenas as afirmativas II, III e IV estão corretas.
- d. Apenas as afirmativas I, II e III estão corretas.
- e. As afirmativas I, II, III e IV estão corretas.

3. O monitoramento de requisitos é um processo que consiste em garantir que o escopo do software desenvolvido seja realizado; é a monitoração contínua dos processos da Engenharia de Software. A cada alteração (em um ou mais requisitos) deve-se garantir a rastreabilidade das alterações.

Assinale a alternativa correta sobre a rastreabilidade de um requisito.

- a. A rastreabilidade de um requisito pode ser definida no documento final, resultante de todo o monitoramento dos requisitos.
- b. A rastreabilidade de um requisito é a elicitação de todos os requisitos com a ajuda de todos os *stakeholders* do projeto.
- c. A rastreabilidade de um requisito pode ser definida como a habilidade de acompanhar e descrever o ciclo de vida de um requisito.
- d. A rastreabilidade de um requisito é um processo unitário, realizado no final do processo de monitoramento de requisitos, visando determinar a prioridade do requisito.
- e. A rastreabilidade de um requisito são etapas opcionais que podem ou não ser realizadas quando há alguma modificação ou conflito de requisitos.

Seção 3

Modelagem de requisitos

Diálogo aberto

Você já reparou que estamos cada vez mais atolados com muitas informações? A cada dia aparecem mais informações que devemos administrar. Como esse excesso de informação deve ser administrado em uma empresa de desenvolvimento de software? É necessário um gerenciamento eficaz das informações recolhidas para o desenvolvimento de um software. Em um desenvolvimento de software, tudo precisa ser documentado, organizado e validado com o cliente, para que ambas as partes concordem com o que será desenvolvido.

Chegamos à etapa final desta unidade, o desafio é documentar o processo de modelagem dos requisitos encontrados.

Você está trabalhando como Analista de Sistemas e faz parte de uma equipe responsável pela Engenharia de Requisitos do Software. Você documentará os requisitos para o sistema da empresa de plantas ornamentais. Após realizar várias visitas, ter conversado com o cliente e outros usuários do sistema, vários requisitos funcionais e não funcionais do software foram levantados, mas não houve uma documentação apropriada para essas ações.

Agora, o desafio final é criar uma Documentação da Especificação de Requisitos. Será possível usar uma padronização para documentar os requisitos? Existe alguma vantagem ao utilizar essa técnica? Você consideraria importante documentar a especificação e a elicitação dos requisitos?

É de fundamental relevância que você estude esta seção. Ela o ajudará na resolução das questões do desafio final, pois apresenta técnicas para a modelagem de requisitos que devem ser aplicadas pelos analistas de sistemas.

Crie um documento final com toda a documentação dos requisitos encontrados, crie uma apresentação para que seja apreciado e validado com o cliente.

Bom desafio!

Não pode faltar

A Modelagem de Requisitos tem como meta fundamental o foco “no que será feito” e não em “como será realizado” e, conforme, afirma Pressman (2016), o modelo de requisitos deve atingir três objetivos principais:

- Primeiro: descrever o que o cliente solicitou.
- Segundo: desenvolver uma base para a criação do Projeto de Software.
- Terceiro: produzir um conjunto de requisitos que possa ser validado, assim que o software estiver pronto.

A fim de concluir o entendimento sobre os requisitos, podemos destacar que não só descrevem o fluxo de informação que entra e sai de um sistema e a transformação dos dados no sistema, mas descrevem também todas as restrições quanto ao seu comportamento e desempenho, conforme afirma Pfleeger (2004). Os requisitos permitem: (i) a explicação (na visão dos desenvolvedores), o entendimento de como os clientes querem que o sistema funcione; (ii) informar as funcionalidades e atributos que o sistema deverá possuir; (iii) informam a equipe de testes aquilo que deverá ser validado com o cliente.

Observe no Quadro 3.4 as perguntas que devem ser realizadas sobre cada requisito, com o objetivo do entendimento do cliente e dos desenvolvedores, uma forma de garantir a qualidade do requisito levantado.

Quadro 3.4 | Características dos requisitos

Pergunta	Descrição
Os requisitos estão corretos?	Todos os envolvidos devem verificar se os requisitos estão corretos.
Os requisitos estão consistentes?	Procurar inconsistência de informação (um requisito para uma determinada ação e outro requisito desfaz essa necessidade).
Os requisitos estão completos?	Verificar a existência de lacunas nos requisitos, com o máximo de informação sobre o que será realizado, como será realizado e as alternativas que podem haver.
Os requisitos são realistas?	Verificar se o que está sendo solicitado é realmente possível de ser realizado.
O requisito descreve algo necessário para o cliente?	O requisito deverá focar no que o sistema deverá realizar, evitando funções desnecessárias (e consequentemente perda de tempo no desenvolvimento).

Os requisitos podem ser validados?	A partir de testes planejados é verificada a validade do requisito.
Os requisitos podem ser rastreados?	Todo requisito deverá ser rastreado, caso haja necessidade. Agrupamentos de requisitos pelas suas funcionalidades é uma opção de facilidade no rastreio.

Fonte: adaptado de Pfleeger (2004, p. 118).

Assim como em nossas vidas ocorrem diversas mudanças, em um software isso não é diferente. Mudanças na empresa, mudanças em rotinas, mudanças técnicas, mudanças na lei, podem afetar diretamente o software e consequentemente seus requisitos. Sommerville (2011) enfatiza que o gerenciamento de mudanças de requisitos é um Processo de Gerenciamento e Controle das mudanças. O Gerenciamento de Requisitos é um modelo sistemático para localização, documentação, organização e rastreamento dos requisitos de um sistema. É necessário um acompanhamento de toda e qualquer alteração nos requisitos e desta forma ter um controle das suas mudanças.

Existem diversas Técnicas de Modelagem de Requisitos, a primeira, conforme Sommerville (2011), é fazer uma separação entre os Requisitos Funcionais e Requisitos Não Funcionais, determinando um equilíbrio entre ambos (lembre-se que os Requisitos Funcionais dependem dos Requisitos Não Funcionais). O ideal é agrupar os requisitos conforme os seus objetivos, suas prioridades e seus tipos. Após o agrupamento dos requisitos funcionais e não funcionais, devemos agrupar todos os Requisitos Funcionais com prioridade Essencial (sem esse tipo de Requisito, o software não estará apto a funcionar).

Exemplificando

Suponha que o requisito a ser escrito deve solicitar os dados do professor para serem cadastrados no sistema. Observe:

- Exemplo do que NÃO ESCREVER: cadastro dos dados dos professores no sistema.
- Exemplo DO QUE DEVE SER ESCRITO: cadastrar os dados dos professores no sistema, como: nome, carga horária, e-mail, celular.

O tempo verbal do infinitivo (no caso, *cadastrar*) demonstra uma obrigação, mas que necessita ser expressa como uma ação a ser efetivada.

De acordo com Pressman (2016), a Elicitação de Requisitos (também conhecida como Levantamento de Requisitos) procura identificar o problema a ser resolvido e todo pessoal envolvido (*stakeholders*), procurando combinar a solução dos problemas encontrados, com a negociação (do que será realizado) e finalizando com a especificação dos requisitos. A documentação a ser produzida na fase da Elicitação de Requisitos pode ser, conforme Pressman (2016):

- Listas de funcionalidades: identificadas em entrevistas individuais e ou em reuniões de grupos.
- Casos de Uso: com o auxílio da UML podemos exemplificar ações do sistema.
- Cenários de Uso: é uma descrição narrativa textual, em linguagem natural (sem termos técnicos) que descreve uma determinada situação de uso do sistema.

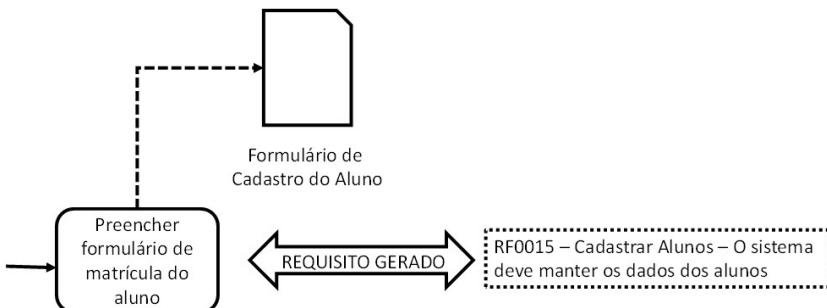
Uma técnica de Modelagem de Requisitos, utilizada na fase de Elicitação de Requisitos, é a técnica REMO (sigla em inglês de *Requirements Elicitation oriented by business process Modeling*) e que permite integrar a modelagem de processos de negócios (no desenvolvimento do sistema) usando a notação BPMN (*Business Process Model and Notation* - Modelo e Notação de Processos de Negócio) com a Elicitação de Requisitos, como afirma Vieira (2012).

A técnica REMO permite que a extração de requisitos seja retirada dos diagramas de processos de negócios, apoiados por um conjunto de heurísticas (métodos de investigação motivado na aproximação progressiva de um determinado problema). Vieira (2012) descreve que o método de aplicação da técnica REMO é composto por duas fases: a primeira fase foca no entendimento do contexto para conhecer o domínio do problema do software (que será produzido). Deverá ser elaborado um documento (feito pelo Analista de Sistemas) contendo informações importantes para entender o domínio do software e que são: problemas e necessidades; papéis envolvidos nos processos; recursos necessários e disponíveis e diagramas de processos de negócios. A segunda fase se concentra nos requisitos, na extração e descrição dos requisitos do sistema.

Observe na Figura 3.6 como um requisito funcional pode ser gerado a partir da Modelagem de Processos de Negócios. Primeiramente, foi identificada uma atividade (Preencher formulário de matrícula do aluno) e, então, é realizado o seguinte questionamento: “Essa atividade pode ser um requisito?”,

caso afirmativo temos um requisito encontrado a partir da Modelagem de Processos de Negócios. No exemplo da Figura 3.6 o requisito gerado foi o RF0015 – Cadastrar Alunos. O sistema deve manter os dados dos alunos. A palavra “manter” no requisito significa: cadastrar, consultar, alterar e excluir (evitando, assim, a escrita do mesmo requisito mudando os verbos – que representam o objetivo do requisito).

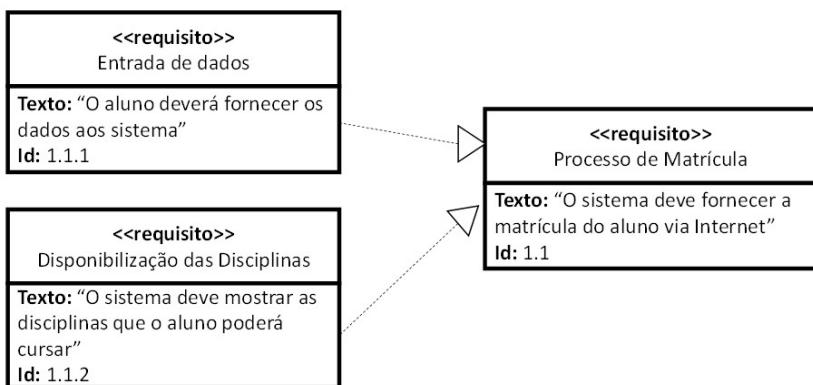
Figura 3.6 | Requisito Gerado a partir da Modelagem de Processos de Negócio



Fonte: elaborada pela autora.

Existem outras técnicas de Modelagem de Requisitos, como a Linguagem de Modelagem SysML (*Systems Modeling Language*), que é *open source* (código aberto), derivada da linguagem UML e especificada pelo *Object Management Group*, SysML suporta as seguintes atividades: especificação, análise, design, verificação e validação de sistemas. Nesta modelagem podemos reutilizar vários diagramas da UML, e foi criado o Diagrama de Requisitos, que possui como principal vantagem a demonstração dos requisitos e suas relações. O diagrama de requisitos do SysML é baseado em textos e os relacionamentos entre os requisitos (nos retângulos). O requisito “Processo de Matrícula” possui duas dependências, os requisitos: “Entrada de dados” e “Disponibilização das Disciplinas” precisam ser elaborados após a abertura do requisito “Processo de Matrícula”. Importa destacar que o diagrama pode ficar muito grande, dependendo da quantidade de requisitos. Nesse sentido, ainda é usual a utilização de tabelas para descrever os requisitos.

Figura 3.7 | Diagrama de Requisitos



Fonte: elaborada pela autora.

A UML é amplamente utilizada na modelagem de requisitos. A Linguagem UML (Linguagem de Modelagem Unificada - que será demonstrada na próxima unidade) possui muitos diagramas que podem ser utilizados de forma isolada (um diagrama para representar algo específico) ou podemos usar um dos vários diagramas que compõem a UML, por exemplo: Caso de Uso (o mais utilizado para modelar os requisitos), Diagrama de Sequência, Diagrama de Atividades. Cada empresa pode adotar uma ou várias técnicas de Modelagem de Requisitos, com a finalidade de produzir uma documentação ao final da modelagem.

A documentação da Elicitação de Requisitos registra os principais tópicos que dizem respeito diretamente ao que o sistema deverá realizar e determinar em quais condições as soluções serão realizadas, como afirma Sommerville (2011). O documento gerado pode ser de forma sequencial (em forma de lista ou tópicos), onde os requisitos são descritos sem muita complexidade, porém de forma clara, objetiva e completa. As lacunas desse documento (se existirem e ou na medida que forem encontradas) serão preenchidas durante a fase de Especificação de Requisitos.

Figura 3.8 | Documento de Elicitação de Requisitos

<u>Sistema Acadêmico Genérico – Documento de Elicitação de Requisitos</u>	
Requisitos Funcionais	
1.	Registrar todos os dados dos alunos.
2.	Matricular o aluno por disciplina.
3.	Cadastro de todos os professores e suas especializações.
4.	Distribuir as disciplinas para os professores, respeitando suas especializações.
5.	Calcular a frequência do aluno.
6.	Emitir relatório da frequência do aluno por disciplina.
7.	Emitir relatório das notas dos alunos por disciplina.
8.

Fonte: elaborada pela autora.

O detalhamento de cada requisito pode ser necessário caso evidencie dúvidas. Na Figura 3.8, observe o segundo Requisito Funcional: “2. Matricular o aluno por disciplina”; esse requisito requer um melhor entendimento, sendo necessária nova documentação. Na Figura 3.9, temos o detalhamento deste requisito.

Figura 3.9 | Detalhamento do Requisito do Documento de Elicitação de Requisitos

<u>Documento de Elicitação de Requisitos – Detalhamento</u>	
2. Matricular o aluno por disciplina.	
Descrição:	O aluno poderá escolher a disciplina que irá cursar (desde que a disciplina seja ofertada no semestre). O sistema deverá verificar se o aluno estiver no primeiro semestre da faculdade, se sim ele precisará se matricular em todas as disciplinas ofertadas; caso contrário (se não estiver no primeiro semestre) ele poderá se matricular nas disciplinas ofertadas (com o limite de seis disciplinas por semestre).
Fontes:	Secretaria da faculdade e Coordenador do Curso de TI.
Informações de Entrada:	O aluno precisar informar sua matrícula, seu curso e quais disciplinas deseja cursar (se não for aluno do primeiro semestre).
Informações de Saída:	Será emitido um comprovante de matrícula, que poderá ser salvo e ou impresso pelo aluno.

Fonte: elaborada pela autora.

Reflita

Requisitos ocultos são funções realizadas pelo sistema e, na maioria das vezes, o usuário não tem conhecimento sobre a sua existência. Geralmente, são referentes a cálculos realizados no software, atualizações do sistema realizadas pela equipe de desenvolvimento. Mas, você saberia elencar três requisitos ocultos referentes ao Sistema Acadêmico Genérico?

Os documentos exemplificativos da Figura 3.10 e da Figura 3.11 são esqueletos que podem (e devem) ser adaptados conforme a necessidade da equipe que está fazendo a Elicitação de Requisitos. Entretanto, vale a pena ressaltar que a documentação produzida nesta fase começa nos registros das reuniões com todo o pessoal envolvido na Elicitação de Requisitos e começa a ser elaborado o Documento de Visão do Sistema (exemplo genérico na Figura 3.10). Segundo Vazquez (2016), o Documento de Visão tem como finalidade identificar as restrições e fornecer uma visão geral do sistema que se pretende desenvolver, sendo uma ferramenta auxiliar no controle de comunicação e mudanças do projeto.

Figura 3.10 | Documento de Visão de um Sistema

Data:	Autor:	Revisão:
Sumário		
1.0 – Introdução		
1.1 Objetivo do documento		
1.2 Escopo		
1.3 Abreviaturas, Siglas		
2.0 – Contexto		
2.1 Declaração do Problema		
3.0 – Lista de Stakeholders		
3.1 Primários		
3.2 Secundários		
4.0 – Lista de Requisitos		
4.1 Requisitos Funcionais		
4.2 Requisitos Não Funcionais		
5.0 – Lista dos Riscos		
6.0 – Lista das Restrições		
6.1 Software		
6.2 Hardware		
6.3 Ambiente e Tecnologia		

Fonte: elaborada pela autora.

Na Documentação da Especificação de Requisitos, de acordo com Pressman (2016), os Requisitos Funcionais e os Requisitos Não Funcionais são documentados; nesta etapa podem ser utilizados Diagramas de Casos de Uso (UML) e realizados protótipos de parte do sistema. Na documentação deverão ser descritos todos os passos das funcionalidades e das restrições do requisito. O documento da Especificação de Requisitos deve seguir alguns padrões, conforme Sommerville (2011):

- Linguagem natural sem terminologias muito técnicas, facilitando a leitura de todos os envolvidos.
- Cada requisito deve ter um identificador único (facilitando a rastreabilidade), nos Requisitos Funcionais é utilizado a sigla RF, já nos Requisitos Não Funcionais é utilizado a sigla RNF, todos os requisitos devem ter um número acompanhando a sigla.
- Os Requisitos Funcionais devem estar separados dos Requisitos Não Funcionais, em listas separadas para facilitar a visualização e

compreensão, é importante que os requisitos estejam agrupados conforme seus objetivos específicos.

A documentação gerada da Especificação de Requisitos não segue um padrão pré-estabelecido e as empresas adaptam os formulários da documentação de acordo com suas necessidades internas de desenvolvimento. Uma configuração básica da documentação da Especificação de Requisitos pode ser vista na Figura 3.11.

Figura 3.11 | Configuração Básica da Documentação da Especificação de Requisitos

Identificador:			
Nome:			
Módulo:			
Data Criação:		Autor:	
Data Última Alteração:		Autor:	
Versão:		Prioridade:	
Descrição:			

Fonte: elaborada pela autora.

Os componentes básicos utilizados na Figura 3.11 da Documentação da Especificação de Requisitos possuem as seguintes características:

- Identificador: campo que identifica o requisito e que não deverá se repetir, deve sempre ter o sufixo RF ou RNF antes de uma sequência numérica.
- Nome: para identificar o requisito, deve ser curto para não confundir com a descrição.
- Módulo: identifica a qual módulo (parte do sistema) o requisito pertence.
- Data da criação: a data em que o requisito foi criado (especificado pela primeira vez).
- Data da modificação: a data em que o requisito especificado foi modificado.
- Autor: nome de quem criou e ou modificou o requisito.
- Versão: o número da versão do requisito (podem haver várias versões) inicializa com a Versão 1.0.
- Prioridade: determina se o requisito é: essencial, importante ou desejável.

- Descrição: descrição textual de forma bem detalhada sobre as funcionalidades do requisito (observe que, a partir desta descrição, vários diagramas como Casos de Uso podem ser gerados).

Assimile

Em um sistema temos mais Requisitos Funcionais ou mais Requisitos Não Funcionais? Pelas funcionalidades do sistema certamente haverá uma lista maior de Requisitos Funcionais do que Requisitos Não Funcionais. Deve-se ter uma atenção na especificação dos Requisitos Não Funcionais, visto que eles geralmente afetam a qualidade de entrega do produto final (no quesito de usabilidade, segurança, tecnologia empregada). Deve-se estar atento na hora da especificação destes tipos de requisitos, pois nem sempre os usuários (cliente e usuários finais) sabem se expressar sobre esses requisitos. As consequências de falhas de especificação podem arruinar o projeto inteiro.

Um exemplo do Documento da Especificação de Requisitos pode ser visualizado no Quadro 3.5.

Quadro 3.5 | Documentação da Especificação do Requisito RF0015

Identificador:	RF0015		
Nome:	Matricular o aluno por disciplina.		
Módulo:	Módulo Matrículas		
Data Criação:	02/12/2019	Autor:	Lucybelo Artys
Data Última Alteração:	N/A	Autor:	N/A
Versão:	1.0	Prioridade:	Essencial
Descrição:	O aluno deverá se logar no sistema para poder realizar a matrícula do semestre. O sistema deverá verificar se não há pendências financeiras para que o aluno possa se matricular. O sistema deverá verificar se o aluno estiver no primeiro semestre da faculdade, se sim ele precisará se matricular em todas as disciplinas ofertadas. O sistema deverá mostrar as disciplinas ofertadas no semestre (de acordo com curso que o aluno esteja matriculado). O aluno poderá escolher a disciplina que irá cursar (desde que a disciplina seja ofertada no semestre). O aluno poderá se matricular nas disciplinas ofertadas no semestre (com o limite de seis disciplinas por semestre).		

Fonte: elaborado pela autora.

Pressman (2016) enfatiza que a Modelagem de Requisitos aplicada à Análise de Sistemas define vários aspectos do problema a ser resolvido, resultando na especificação de detalhes operacionais do software. Nesta etapa, o requisito é mais detalhado, sendo ampliado em relação à especificação original e são utilizados vários modelos que auxiliam no projeto de software e fornecem meios para verificar a qualidade do que está sendo construído, esses modelos podem ser:

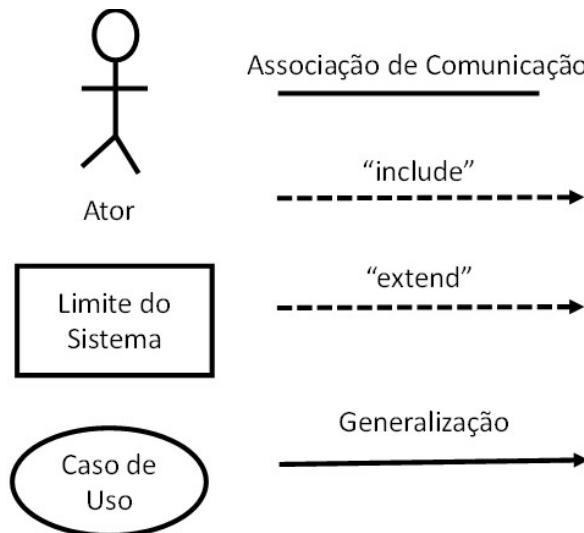
- Modelos baseados em cenários de requisitos, com visões diferenciadas dos atores do sistema.
- Modelos de classes orientadas a objeto (UML) utilizando os atributos e métodos para verificação dos requisitos do sistema.
- Modelos comportamentais e baseado em padrões para verificar como o sistema se comporta com acontecimentos externos ao sistema.
- Modelos de dados que representam o domínio de informação para o problema.
- Modelos de fluxos dos elementos funcionais do sistema e como eles transformam os dados na medida em que são utilizados.

O processo de Modelagem de Requisitos aplicada à Análise de Sistemas pode ser considerado uma ligação entre toda a fase de Elicitação e Especificação de Requisitos com a fase de Modelo de Projetos do sistema.

Outra forma de especificar requisitos é a partir de Diagramas de Caso de Uso. Pressman (2016) afirma que um Cenário de Uso (ou especificação do Caso de Uso) é um detalhamento do Caso de Uso. Esse detalhamento ajuda a modelar o requisito especificado e será a principal forma de comunicação entre a equipe de desenvolvimento, ou seja, entre o Analista de Sistemas (que fez a modelagem) e o Programador (que programará o requisito modelado).

Conforme Medeiros (2008), o Diagrama de Caso de Uso é a parte mais importante da construção de um software orientado a objetos usando a UML (Linguagem de Modelagem Unificada), esses diagramas acompanham o software desde sua inicialização até a finalização. O Diagrama de Caso de Uso é uma forma de comunicação entre o Analista de Sistemas e os Programadores, pois os diagramas detalham o que precisará ser implementado (codificado). Na Figura 3.12 os componentes do Diagrama de Caso de Uso podem ser observados.

Figura 3.12 | Componentes de Casos de Uso



Fonte: elaborada pela autora.

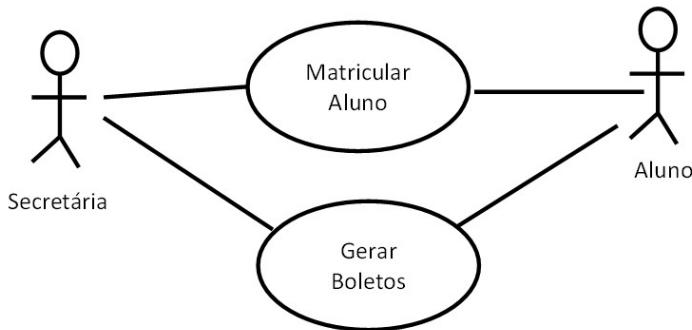
A Figura 3.12 demonstra os componentes do Diagrama de Casos de Uso, cada componente possui as seguintes características:

- **Autor** (boneco): os Atores não fazem parte do sistema e representam algo (pode ser um outro sistema) ou alguém que interage com o sistema. Um Ator pode somente fornecer informações para o sistema ou somente receber informações do sistema, ou ainda, fornecer e receber informações para o sistema. Num diagrama podem haver vários Atores.
- **Limite do Sistema** (retângulo): representam o limite do diagrama.
- **Caso de Uso** (elipse): descrevem as funcionalidades (os requisitos) do sistema, são as transações executadas no sistema. Cada Caso de Uso (cada elipse) deve ser detalhada e especificada com orientações sobre suas funcionalidades. Um Caso de Uso pode interagir com um ou mais Atores e com outros Casos de Uso.
- **Associação de Comunicação** (seta simples): é o relacionamento (a ligação) entre os Atores e outros Casos de Uso.
- **“Include”** (seta com linha tracejada): esse relacionamento mostra que o tipo de relacionamento entre dois Casos de Uso implica na obrigatoriedade da execução do Caso de Uso que está sendo incluído.

- Muitos casos de uso podem compartilhar pedaços de pequenas funcionalidades. A “Extend” (seta com linha tracejada): esse relacionamento é usado para mostrar um comportamento opcional.
- Generalização (linha com seta): representam a herança entre componentes (atores ou casos de uso).

Na Figura 3.13 é demonstrado um modelo de diagramas de Casos de Uso com dois processos (cada processo ou a elipse é chamada de Caso de Uso): (i) Matricular o Aluno e (ii) Gerar Boletos, e envolve dois atores que “alimentarão” os processos com o fornecimento ou o recebimento de informações.

Figura 3.13 | Exemplo de Casos de Uso



Fonte: elaborada pela autora.

Cada Caso de Uso representa uma determinada função ou tarefa do sistema, segundo Sommerville (2011), que envolve a interação externa com o sistema e pode ser um cenário simples que ajuda no entendimento do sistema (ou de parte dele). É na especificação (ou descrição) de cada Caso de Uso que o processo de Modelagem de Requisito é ampliado. Observe o Quadro 3.6, com a especificação do Caso de Uso Matricular Aluno.

Quadro 3.6 | Especificação do Caso de Uso: Matricular Aluno

Nome do caso de Uso		Matricular Aluno
Autor Principal		Aluno
Atores Secundários		Secretária
Resumo		Este caso de uso tem por objetivo detalhar o processo de matrícula do aluno no sistema.
Pré-condições		1. O aluno deverá estar matriculado no sistema. 2. Não poderá haver nenhuma pendência financeira no sistema.
Fluxo Principal		
Ações do Ator		Ações do Sistema
1. O aluno informa a matrícula e senha para se <i>logar</i> no sistema. 4. O aluno poderá escolher a disciplina que cursará num total de 6 disciplinas por semestre. 5. O aluno deverá finalizar a escolha da disciplina apertando o botão FINALIZAR. 6. O aluno poderá escolher entre imprimir ou salvar o comprovante de matrícula.		2. O sistema deverá verificar a matrícula e validar a senha do aluno. 3. Deverão ser apresentadas as disciplinas que o aluno poderá cursar. 6. Um comprovante de matrícula deverá ser gerado em formato PDF.
Fluxos Alternativos		
Ações do Ator		Ações do Sistema
1.1 O aluno poderá redefinir a sua senha.		2.1 – Caso o aluno não tenha senha o sistema deverá permitir o cadastramento da senha ou a sua redefinição. 3.1 – Deverá ser verificado o semestre atual do aluno, se for o 1º semestre do aluno, ele deverá escolher todas as disciplinas. 3.2 – Deverá ser verificada a quantidade de disciplinas a serem escolhidas (limite dever ser igual a 6).

Fonte: elaborado pela autora.

A especificação ou descrição de um Caso de Uso possui como objetivo informar quais os atores (pessoas ou sistemas relacionados com o sistema) interagem com as funcionalidades específicas que estão sendo modeladas. Não há um modelo padrão de especificação, é totalmente adaptável às necessidades das empresas, mas é aconselhável que seja escrito de forma simplificado para facilitar seu entendimento.

Chegamos ao final desta seção e você pode notar que existe grande quantidade de formulários e diagramas que ajudam no processo de Modelagem de Requisitos. Você está iniciando o processo de conhecimento de Análise e Modelagem de Sistemas e a prática ajudará a reconhecer a necessidade desta documentação, você perceberá que a palavra adaptação será muito utilizada, pois você poderá adaptar os modelos de acordo com a necessidade de modelagem ou de acordo com o sistema que será desenvolvido para o sistema. Convidado você a continuar no mundo da modelagem de sistemas.

Siga em frente, bons estudos.

Sem medo de errar

Para relembrar, você está trabalhando como analista de sistemas e faz parte de uma equipe responsável pela Engenharia de Requisitos do Software. Você documentará os requisitos para o sistema da empresa de plantas ornamentais. Após realizar várias visitas, conversado com o cliente e outros usuários do sistema, vários requisitos funcionais e não funcionais do software foram levantados, mas não houve uma documentação apropriada para essas ações.

O desafio final é criar uma Documentação da Especificação de Requisitos e, para isso, primeiramente resgataremos os Requisitos Funcionais já relacionados na situação-problema anterior; são eles:

[RF0001] – O sistema deverá manter (incluir, consultar, alterar e excluir) os dados dos clientes: nome, endereço, CPF ou CNPJ, e-mail, telefones.

[RF0002] – O sistema deve gerar um relatório das Plantas disponíveis para venda.

[RF0003] – O sistema deve permitir que seja cadastrado os pedidos de encomenda de plantas e produtos da empresa.

[RF0004] – O sistema deve permitir a inclusão de pedidos de jardinagem, permitindo a escolha das plantas e demais produtos.

[RF0005] – O sistema deverá manter o cadastro de todos os produtos da empresa.

[RF0006] – O sistema deverá manter as informações botânicas das plantas.

[RF0007] – O sistema deverá emitir relatórios sobre as encomendas, situação do estoque, trabalhos paisagísticos em andamento e finalizados.

[RF0008] – O sistema deverá manter o cadastro de todos os fornecedores.

[RF0009] – O sistema deverá emitir um relatório das plantas que podem ser plantadas classificadas por época do ano, tipos de terrenos e tipos de construção.

[RF0010] – O sistema deverá manter um cadastro dos usuários do sistema, classificando-os como: Vendedor, Administrador e Supervisor.

Será possível usar uma padronização para documentar os requisitos? Existe alguma vantagem ao utilizar essa técnica? Sim, podemos utilizar o padrão de modelagem apresentado nesta seção, conforme o modelo da Figura 3.11.

Identificador:			
Nome:			
Módulo:			
Data Criação:		Autor:	
Data Última Alteração:		Autor:	
Versão:		Prioridade:	
Descrição:			

E como ficaria a documentação do Requisito Funcional: [RF0008] – O sistema deverá manter o cadastro de todos os fornecedores? Observe a figura que ilustra a Documentação básica do requisito RF008.

Identificador:	RF0008		
Nome:	O sistema deverá manter o cadastro de todos os fornecedores		
Módulo:	N/A		
Data Criação:	05/11/2019	Autor:	Claudia W.
Data Última Alteração:	N/A	Autor:	N/A
Versão:	1.0	Prioridade:	Essencial
Descrição:	Todo fornecedor deverá ser cadastrado antes de ser efetuada uma encomenda ou compra. Todos os dados deverão ser cadastrados pelo usuário do sistema. Os dados que deverão ser informados são: Nome fantasia, Razão Social, CNPJ, Inscrição Estadual, Endereço completo, Cidade, Estado, E-mail.		

Utilize o modelo da Figura 3.11 para documentar os requisitos do sistema de plantas ornamentais. Cada requisito deverá ter o seu próprio formulário. Não esqueça que a identificação dever ser única. Caso alguma informação não seja preenchida no formulário, insira a sigla N/A (Não se Aplica), não deixe a lacuna em branco. Toda a versão deverá começar do número 1. Procure ser detalhista no item da descrição do requisito, essa descrição servirá de base para a construção do **Caso de Uso desse requisito**.

Você consideraria importante documentar a especificação e a elicitação dos requisitos?

Com certeza é importante, por mais trabalhoso que seja! É através da documentação produzida nas etapas de elicitação e especificação que poderemos nos certificar que nada foi esquecido (ou quase nada, pois sempre há contratemplos), evitando retrabalhos. A elicitação dos requisitos possui a vantagem de identificar todos os fatos que constituem os requisitos

do software a ser produzido, trazendo um melhor entendimento do que será desenvolvido.

Agora é com você! Faça as documentações dos requisitos funcionais e apresente para seus colegas em sala de aula. A apresentação é muito importante. Você, como analista de sistemas, precisará falar em público diversas vezes, aproveite essa oportunidade para treinar e também solicite opiniões de seus colegas sobre a documentação dos requisitos. Será que alguém achou um modelo melhor para documentar os requisitos?

Crie um documento final com toda a documentação dos requisitos encontrados, crie uma apresentação para que seja apresentado e validado com o cliente.

Faça valer a pena

1. Na Documentação da Especificação de Requisitos, de acordo com Pressman (2016), os Requisitos Funcionais e os Requisitos Não Funcionais são documentados, podem ser utilizados Diagramas de Casos de Uso (UML) e realizados protótipos de parte do sistema.

Na documentação da Especificação de Requisitos deverão ser descritos:

- a. Todos os nomes dos envolvidos no desenvolvimento do projeto.
- b. Somente os Requisitos Não Funcionais, conforme prevê o documento da Especificação.
- c. Todos os passos das funcionalidades e das restrições do requisito.
- d. Somente os diagramas para auxiliar na comunicação com os usuários envolvidos no projeto.
- e. Todos os Requisitos Funcionais, os Requisitos Não Funcionais não precisam ser especificados.

2. Não há regras rígidas sobre a criação de um Documento de Especificação de Requisitos, entretanto, Sommerville (2011) afirma que um documento de Especificação de Requisitos deve seguir alguns padrões para auxiliar na compreensão do que está descrito na especificação.

Analise as afirmativas a seguir sobre os padrões a serem seguidos na Especificação de Requisitos.

- I. Separação entre os Requisitos Funcionais e os Requisitos Não Funcionais.

- II. Linguagem natural para facilitar a leitura de todos os envolvidos.
- III. Uso de identificador único para cada requisito.
- IV. Uso de jargões técnicos para um melhor entendimento da equipe de desenvolvimento.

É correto o que se afirma em:

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

3. Sommerville (2011) destaca que a Documentação da Elicitação de Requisitos registra os principais tópicos que dizem respeito diretamente ao que o sistema deverá realizar e determinar em quais condições as soluções serão realizadas. As lacunas desse documento (se existirem e na medida que forem encontradas) serão preenchidas durante a fase de Especificação de Requisitos.

Marque a alternativa que apresenta como deve ser gerado o Documento da Elicitação de Requisitos.

- a. O documento gerado pode ser de forma sequencial em forma de lista ou tópicos, onde os requisitos são descritos com muita complexidade e profundidade, sendo esse o documento essencial para os programadores.
- b. O documento gerado pode ser de forma sequencial em forma de lista ou tópicos, onde os requisitos são descritos sem muita complexidade, porém de forma clara, objetiva e completa.
- c. O documento gerado pode ser de forma gráfica, onde os requisitos são descritos através de diagramas, sem muita complexidade, porém de fácil entendimento.
- d. O documento gerado pode ser de forma sequencial em tópicos com o auxílio de gráficos ilustrativos, onde os requisitos são descritos de forma completa para o entendimento dos *stakeholders*.
- e. O documento gerado pode ser de forma aleatória, onde os requisitos são descritos através de especificações técnicas (com as Regras de Negócio) para o uso exclusivo dos programadores.

Referências

AURÉLIO, B. H. F. **Elicitar**. In: Novo dicionário Aurélio da língua portuguesa. 3. ed. rev. aum. Rio de Janeiro: Nova Fronteira, 1999.

ENGHOLM JR., H. **Engenharia de Software na Prática**. São Paulo: Novatec, 2010.

FALBO, R. A. **Engenharia de Requisitos**: Notas de Aula. Universidade Federal do Espírito Santo, Vitória, 2012.

HAUSE, M. et al. The SysML modelling language. In: Fifteenth European Systems Engineering Conference. 2006. p. 1-12.

MEDEIROS, E. S. **Desenvolvendo software com UML 2.0**: definitivo. 4 reimpr. São Paulo: Pearson Makron Books, 2008.

NASCIMENTO, N. M.; VIVACQUA, A. S.; SILVA, M. F. Uma proposta de integração de ER Ágil com IHC. In: PESQUISAS EM ANDAMENTO - SIMPÓSIO BRASILEIRO DE SISTEMAS COLABORATIVOS (SBSC), 15., 2019, Rio de Janeiro. **Anais** [...]. Porto Alegre: Sociedade Brasileira de Computação, set. 2019. p. 111-116. Disponível em: https://sol.sbc.org.br/index.php/sbsc_estendido/article/view/8361/8258. Acesso em: 7 dez. 2019.

PAULA FILHO, W. P. **Engenharia de software**: produtos. 4. ed. Rio de Janeiro: LTC, 2019.

PFLEEGER, S. L. **Engenharia de Software**: teoria e prática. 2. ed. São Paulo: Prentice Hall, 2004.

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

PROJECT MANAGEMENT INSTITUTE (PMI). **Guia PMBOK**: Um Guia para o Conjunto de Conhecimentos em Gerenciamento de Projetos. 6. ed. Pennsylvania: PMI, 2017.

ROBERTSON, J.; ROBERTSON, S. **Mastering the requirements Process**. 2. ed. [S.l.]: Addison-Wesley Professional, 2006.

SOMMERVILLE, I. **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

VAZQUEZ, S.; SIMÕES, G. **Engenharia de Requisitos**: Software Orientado ao Negócio. [S.l.]: Brasport, 2016.

VIEIRA, S. R. C. **Remo**: uma técnica de elicitação de requisitos orientada pela modelagem de processos de negócios. 2012. 129 f. Dissertação (Mestrado em Informática) – Universidade Federal do Amazonas (UFAM), Manaus, 2012.

Unidade 4

Regina Fedozzi

Paradigma orientado a objetos

Convite ao estudo

Caro aluno, a cada dia os softwares tornam se mais complexos e com uma grande quantidade de funcionalidades. Surge, então, aos desenvolvedores, a necessidade de se organizarem para agilizar sua construção e, ao mesmo tempo, gerar sistemas com qualidade. Assim, é importante realizar a modelagem do sistema antes de iniciar o seu desenvolvimento, isto é, analisar quais as funções eles deverão executar, o que os clientes esperam do sistema, bem como de que forma desenvolveremos o software para atender todas as funcionalidades requisitadas pelo cliente. É importante conhecer como trabalhar com o time de desenvolvimento para agilizar a construção do sistema e gerar um produto com qualidade. Para isso, foram desenvolvidos processos e uma linguagem de modelagem que ajuda os desenvolvedores e suas equipes nesta tarefa.

Nesta unidade, você vai estudar sobre estes processos, e também conhecerá a linguagem de modelagem.

O objetivo desta unidade é mostrar a você a importância de como modelar um software usando o processo unificado iterativo e incremental utilizando UML (*Unified Modeling Language* – Linguagem de Modelagem Unificada). A modelagem de software é uma fase importante de ser realizada, e existem algumas formas (padrões) para fazer isso. Nesta unidade daremos ênfase ao paradigma de orientação a objetos, o padrão mais utilizado na engenharia de software para modelagem de sistemas.

Vamos começar ajudando a elaborar um projeto de software para uma organização não governamental (ONG) que apoia o desenvolvimento de valores e o empreendedorismo. A proposta é auxiliar uma ONG em um projeto denominado “Empreendedores de Sucesso”. Sua participação no projeto será ajudar com os seus conhecimentos adquiridos sobre a engenharia de software e com os conceitos de orientação a objetos, processos unificados (PU) e UML (*Unified Modeling Language*) introduzidos nesta seção. A partir disso, propor um projeto de software para uma fábrica de bolos que a ONG pretende implantar.

Inicialmente, você explicará os conceitos fundamentais aos novos empreendedores (Seção 4.1) para que vocês, nas etapas seguintes, possam ter uma boa comunicação e assim desenvolvam a modelagem dos processos que envolvem todas as etapas da produção da fábrica de bolos (Seção 4.2). Na última etapa, você estará apto para elaborar os diagramas que auxiliarão a conclusão do projeto, os diagramas UML (Seção 4.3).

Para prepará-lo para essas atividades, na Seção 4.1 será apresentado a você o paradigma e os conceitos principais de orientação a objetos. Já na Seção 4.2, serão apresentadas as características do processo unificado (PU), bem como suas fases e atividades. Por fim, na Seção 4.3 vamos aprender como desenvolver os principais diagramas da UML.

Em cada uma das seções desta unidade você poderá praticar esses conhecimentos ao modelar o software para a fábrica de bolos do projeto “Empreendedores de Sucesso” e, assim, vivenciar uma experiência valiosa para o seu início profissional nesta carreira de analista de sistemas. Preparado?

Seção 1

Fundamentos da orientação a objetos

Diálogo aberto

Caro aluno, nesta seção vamos tratar dos fundamentos e dos alicerces da orientação a objeto. Quando buscamos solução para um problema percebemos que há várias formas de pensar como resolvê-lo: se a solução será pensada de uma forma sequencial, como uma receita de bolo, ou procurarmos solucionar o problema subdividindo-o em subproblemas (pequenas partes). Os dois padrões de pensamento nos permitem alcançar o objetivo final, a solução do problema, porém podem seguir diferentes abordagens, ou seja, paradigmas diferentes. Segundo Tucker e Noonan (2010), pensamos em um paradigma como um padrão de pensamento que guia um conjunto de atividades relacionadas.

Assim, convidamos você a se preparar para pensar de forma diferente, analisar as situações de forma analítica, sistemática e principalmente observar que um problema complexo pode ser subdividido em pequenas partes para obtenção de soluções mais simples, garantindo que as correções sejam mais rápidas e eficazes.

E, para que você possa modelar um caso do mundo real para o modelo computacional utilizando os conceitos de orientação a objeto, convidamos você a participar do projeto desenvolvido por uma ONG denominado “Empreendedores de Sucesso”, em que boa parte dos participantes do projeto sabem fazer bolos. Assim, pensaram em montar uma fábrica de bolos, com diversos tipos de bolos, alguns simples e outros especiais, além de bolos de festas. A grande dificuldade dos empreendedores está na definição dos padrões.

Eles perceberam que os bolos têm características semelhantes, porém estão com dificuldades para identificar todos os ingredientes e processos que serão necessários em cada etapa de produção e comercialização. A equipe está preocupada, pois como será um processo em que várias pessoas estarão envolvidas e precisam de um padrão, estão com dificuldades para compreender como isso funcionará, e não conseguem entender como podem reutilizar as receitas, se os bolos são diferentes. Assim, você deverá elaborar uma apresentação, a mais dinâmica e visual possível, para explicar aos profissionais envolvidos os conceitos de orientação a objeto associados à fábrica de bolo, denominando os quatro pilares da orientação a objeto, para que eles

possam entender que há padrões entre os bolos de cada linha, sejam os simples, especiais ou de festa. Use sua criatividade.

Para ajudá-lo nessa empreitada, nesta seção estudaremos os fundamentos do paradigma de orientação a objeto e os seus princípios de como solucionar problemas olhando para cada situação e aprender a visualizar padrões. Neste caminho você será guiado pelos conceitos fundamentais de orientação a objetos, tais como: abstração (objeto), classe, herança e polimorfismo.

Preparado para estudar e conhecer um dos paradigmas mais utilizados na análise e no desenvolvimento?

Bons estudo!

Não pode faltar

É muito comum você ouvir frases como: “Este é um novo paradigma” ou “A mudança de paradigma fez toda a diferença”.

Mas, afinal: o que é um paradigma?

Na engenharia de software, consideramos um paradigma como um modelo que já foi testado e segue alguns princípios para a resolução de um problema computacional. Há uma grande vantagem em seguir um modelo, pois facilita o desenvolvimento e a compreensão da solução encontrada.

Segundo Tucker e Noolan, (2010, p. 3) “[...] um paradigma de programação é um padrão de resolução de problemas que se relaciona a um determinado gênero de programas e linguagens”. Conforme podemos observar, padrão e modelo são termos utilizados como sinônimos, e um modelo nada mais é que uma representação do mundo real.

Historicamente, a busca por padrões foi um processo evolutivo na construção de programas e sistemas. No clássico artigo *Go To Statement Considered Harmful* (Vá para [go to] comando considerado danoso) de 1968, Dijkstra tentou deixar claro a importância de utilizar uma estrutura adequada para a construção de algoritmos, com o objetivo de melhorar os padrões e processos de programação e facilitar o entendimento pelos programadores quando eles realizassem uma manutenção (alteração). Essa análise de Dijkstra levou os cientistas a buscarem um padrão para o desenvolvimento de software, um paradigma. Neste período – entre 1968 e 1969 – ocorreu o que se chamou de “crise do software”.

Na década de 1970, cientistas como Niklaus Wirth, Chris Gane e Tom DeMarco sedimentaram o paradigma estruturado que passou a ser utilizado pelos desenvolvedores. Na época, o padrão estruturado era adequado

às linguagens existentes. Porém, alguns fatores surgiram, tais como: o surgimento de ambientes gráficos, como o Windows; o crescimento dos sistemas, que tornaram-se cada vez maiores e mais complexos; a crescente necessidade de integração entre sistemas; a solicitação, dia após dia, de melhorias para serem incrementadas nas aplicações existentes. Diante desses fatores, o padrão estruturado se mostrou ineficiente para algumas aplicações. Assim, um novo paradigma foi desenvolvido à orientação a objetos.

O paradigma orientado a objeto tornou-se muito utilizado a partir de 1997, quando foi criada uma linguagem unificada de modelagem, a UML (*Unified Modeling Language*). Apesar de o paradigma orientado a objeto ter “explodido” após o padrão UML ser criado, esse paradigma surgiu em 1966, com a linguagem SIMULA, a qual introduziu o conceito de objeto. Posteriormente, em 1980, surgiu a linguagem Smalltalk 80, e o conceito de objetos tornou-se mais amplo e com maior importância. A Smalltalk 80 aborda todos os itens como objetos, aproximando o paradigma como objetos físicos do mundo real. De acordo com Sebesta (2006), ambas linguagens suportavam recursos de orientação a objetos, tais como, abstração, herança e vinculação dinâmica. Em 1983 surge a linguagem C *with Classes* que, posteriormente, em 1986, foi chamada de C++.

Com o paradigma orientado a objeto surgiu não só um novo padrão para o desenvolvimento de software, mas também uma nova forma de pensar como modelar os problemas do mundo real. Alan Kay, um dos idealizadores da orientação a objetos, relata seu insight sobre essa forma de pensar no seu artigo *The Early History of Smalltalk* (A origem da SmallTalk):

O princípio básico do projeto recursivo é fazer com que as peças tenham o mesmo poder que o “todo”. Naquele momento, pensei no todo como o computador inteiro e me perguntei por que alguém iria querer dividi-lo em coisas mais fracas, chamadas estruturas e procedimentos de dados. Por que não o dividir em pequenos computadores? [...], mas não em dezenas. Por que não milhares deles, cada um simulando uma estrutura útil? (KAY,1993, [s.p.], tradução nossa)

Na sua nova forma de pensar, Alan Kay partiu da ideia que se os seres vivos são compostos de células independentes, e essas células estão integradas, trocando mensagens entre si, logo os computadores também podem funcionar seguindo o mesmo princípio. Alan estendeu esse conceito para a linguagem de programação que estava desenvolvendo, a Smalltalk. E

percebeu que poderia criar pequenos objetos autônomos, com características e ações próprias, e integrá-los para um objetivo final. A partir da linguagem Smalltalk, posteriormente surgiram outras duas importantes linguagens de programação orientada a objetos, a Eiffel e a Java (Félix, 2016). A linguagem Java, além da orientação a objetos, tinha como principais características a portabilidade e uma grande quantidade de bibliotecas.

Dessa forma, surgiram outros importantes conceitos básicos de orientação a objetos. Entre os conceitos e princípios básicos de POO (Programação Orientada a Objetos) destacaremos:

1. A **abstração** é a ideia central do paradigma orientado a objetos. No processo de abstração, nos referimos a um **objeto** (qualquer item do mundo real como, casa, bolo, carro, sanduíche, boleto, contrato etc.) sem nos preocuparmos com detalhes, como cor, tamanho, código e validade, entre outros. Muitas vezes, associamos a abstração à classificação do objeto. Assim, a modelagem do objeto inicia pela abstração, que é quando reconhecemos os objetos. Suponha que você ouviu o termo cadeira: você pensa na ideia de como é uma cadeira, e isso é uma abstração.
2. A **classe** é a representação da abstração; é o momento em que você define as características que toda cadeira deverá ter e quais ações que ela poderá fazer.
3. As denominações técnicas para as características são **atributos**, e as ações ou comportamentos chamamos de **métodos**.

Assim, a nossa ideia de cadeira se concretiza quando construímos a classe Cadeira e definimos seus atributos, por exemplo: nossa classe Cadeira tem pés, rodinhas, encosto, assento e cor – esses são seus atributos. E ela move o encosto, eleva o assento e anda – esses são seus métodos. Sobre a classe, ainda podemos afirmar que é algo abstrato, mas tornou-se um uma definição do projeto de uma cadeira. Quando uma cadeira é construída a partir desta classe, então temos um objeto do tipo Cadeira, ou seja, um objeto da classe cadeira.

4. O **objeto** é a materialização de forma única de uma classe, e a esta materialização damos o nome de instância da classe. Portanto, um objeto é uma instância de uma classe. Os objetos são únicos e distinguem-se pelos valores dos seus atributos e ações (comportamentos), apesar de pertencerem a uma mesma classe.
5. Assim, uma **classe** é a representação de um conjunto de objetos; em outras palavras, é a representação da abstração do objeto com suas características e comportamentos. Ao construirmos a *cadeiraCinza* a

partir da classe Cadeira que definimos no item (3), temos um objeto do tipo Cadeira, com cor cinza, rodinhas, pés, encosto e assento, que move o encosto e pode andar. A *cadeiraCinza* é única. Em um programa, temos a classe Cadeira, e ao instanciarmos a *cadeiraCinza* a partir da classe Cadeira, ela terá um espaço na memória do computador, no qual serão armazenados seus atributos e métodos e um endereço único que a identifica, isto é, o objeto é a materialização de um exemplar do tipo da classe que o define.

Em UML (*Unified Modeling Language*) utiliza-se uma representação simples para representar uma classe. Um retângulo, dividido em três partes: nome da classe, atributos e métodos.

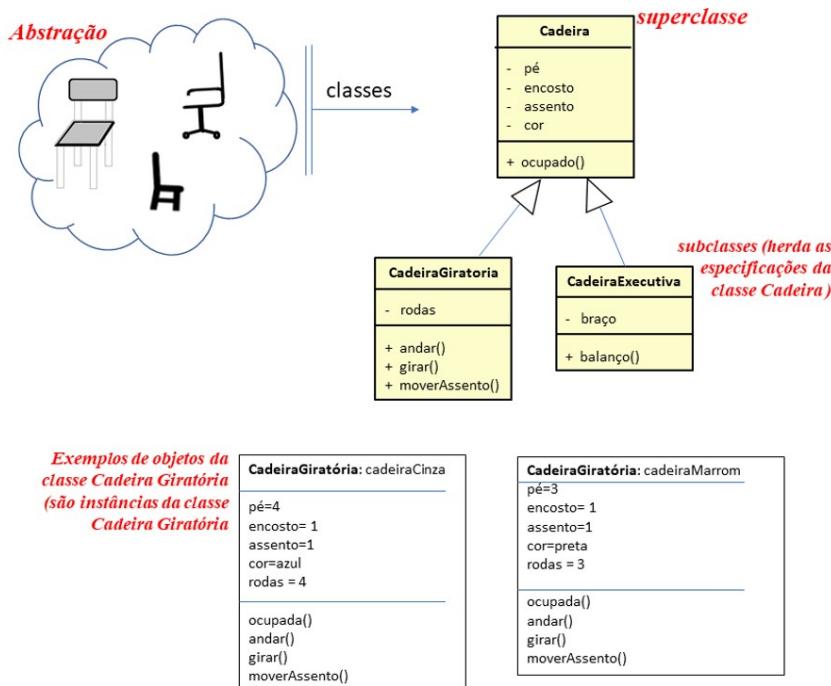
6. A **herança** “permite criar novas classes a partir de classes já existentes, sem duplicar nenhum código” (REZENDE, 2002, p. 214).

No processo de abstração podemos definir classes abrangentes, e durante o processo de modelagem serão refinadas e construídas subclasses que poderão herdar as características e comportamentos da classe genérica. A classe genérica denominamos de **superclasse**, e a classe que herda as características da superclasse chamamos de **subclasse**, conforme a Figura 4.1.

É importante ressaltar que a subclasse pode acrescentar novas características e comportamentos e alterar as já existentes, pois a subclasse é uma nova classe. Essas alterações têm efeito somente na nova classe especificada.

Acompanhando nosso exemplo da classe Cadeira, poderíamos criar uma classe genérica Cadeira apenas com os atributos encosto, assento e pés – essa seria nossa superclasse, ou classe pai. E, se aproveitarmos as características desta superclasse Cadeira e construirmos uma nova classe *CadeiraGiratória*, acrescentando a ela o atributo rodas e os métodos andar, girar, mover assento, para indicarmos que a classe *CadeiraGiratória* herda as características da classe Cadeira usariamos uma seta com a ponta vazada da classe *CadeiraGiratória* apontando para a classe Cadeira, como pode ser visto na Figura 4.1. A classe *CadeiraGiratória* é uma subclass da classe Cadeira, o que chamamos de classe filha. Outro exemplo de herança que pode ser visto na Figura 4.1 é a classe *CadeiraExecutiva*, a qual herda as características da classe Cadeira e adiciona o atributo braço e o método balanço.

Figura 4.1 | POO – abstração, classe, herança e instanciação



Fonte: elaborada pela autora.

7. O **encapsulamento** consiste na junção de partes isoladas de um programa, e essas partes podem ser acessadas separadamente. Na POO, o encapsulamento tem capacidade de tornar a visibilidade das informações e os detalhes da implementação dos métodos de uma classe oculta ou restrita. Em outras palavras, ocultar do usuário da classe como ela faz uma determinada ação ou como os dados são representados. Segundo Rezende “[...] é o processo de dissimulação de todos os detalhes de um objeto que não contribuem para suas características essenciais” (REZENDE, 2002, p. 213).

Um exemplo de aplicação do conceito de encapsulamento pode ser visto quando você usa o caixa eletrônico (um objeto). Você apenas informa sua identificação e senha para acesso, e os serviços são liberados para seu uso. Você não sabe como o caixa fez a validação, onde estava armazenado sua identificação, como ele descriptografou sua senha etc. Isso é um exemplo de encapsulamento, e você pode ter

acesso aos dados encapsulados pela classe apenas por meio de mensagens com o objeto: você solicita e ele responde.

No diagrama de classe usamos o símbolo “-“ (um traço) para indicar que o atributo ou o método está encapsulado. Isto significa que somente o objeto da classe pode modificá-los ou acessá-los. Já o símbolo “+” (adição) indica que o atributo ou método é visível por outros objetos, ou seja, é público. Exemplo desse diagrama você pode observar na Figura 4.1.

Refita

Se você construir uma classe *Conta_bancaria* com os seguintes atributos:

Identificação, Nome, Saldo

E os métodos para *deposito()*, *retirada()* e *verSaldo()*.

Você definiu que o método *deposito()* que sempre que alguém (um outro objeto) informar um valor para *deposito()*, ele deve somar o valor no saldo.

Até aqui, tudo certo, porém você não encapsulou a classe *Conta_bancaria*. **O que pode acontecer?**

Por exemplo, se o atributo saldo não estiver encapsulado, outro objeto do tipo *Conta_bancaria* poderá alterar seu saldo externamente, e realizar uma operação inadequada.

Se existir um objeto *conta_bancaria CONTA_DO_JOAO* e outro objeto *conta_bancaria, CONTA_DO_PEDRO*, e o objeto *conta_bancaria* não for encapsulado, o objeto *CONTA_DO_JOAO* pode alterar o saldo do objeto *CONTA_DO_PEDRO*.

Como podemos fazer com que isso não ocorra?

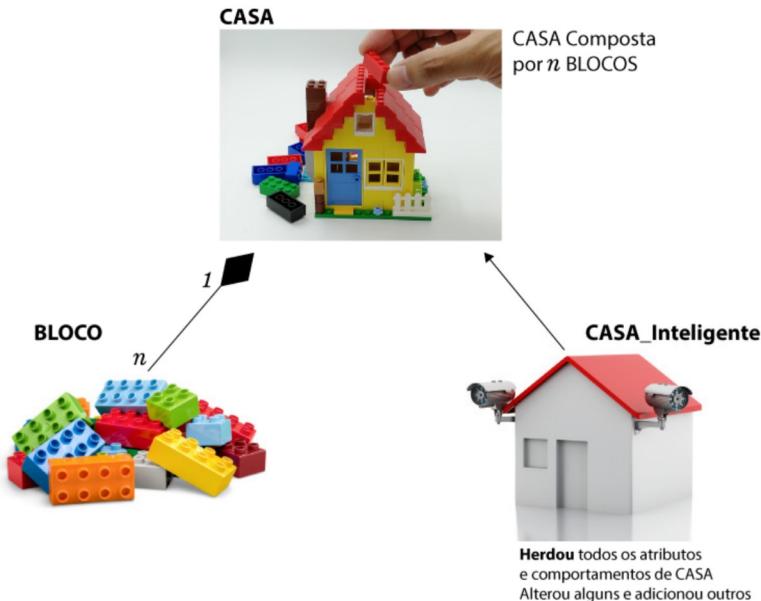
8. O **polimorfismo** “[...] significa que a mesma operação [método] pode atuar de modos diversos em classes diferentes” (REZENDE, 2002, p. 214). O polimorfismo é uma característica da POO que dá às classes flexibilidade na construção da solução e no reuso do código.

Em uma classe, um método (operação) pode aparecer diversas vezes, pois pode ter comportamentos diferentes nas subclasses.

Exemplificando

Se quando criança você brincou de blocos de montar, certamente você aplicou os conceitos (recursos) do paradigma de orientação a objetos. Veja a Figura 4.2.

Figura 4.2 | Princípios elementares de orientação a objetos



Fonte: elaborada pela autora.

Podemos considerar que o BLOCO é o objeto básico. Alguém teve uma ideia, abstraiu e projetou o bloco com suas **características** tais como tamanho, cor, material. Neste conceito, podemos dizer que a **abstração** é uma concepção do objeto, chamada de **classe**. Seguindo a linha de raciocínio, uma **instância da classe**, que será a materialização do projeto concebido, será chamada de **objeto** e, assim, **as características serão os atributos**.

Ainda em relação à Figura 4.2, a classe CASA é composta de diversos objetos BLOCO, isto é, uma casa é formada por n blocos. Temos uma associação entre CASA e BLOCO, e estes objetos interagem entre si.

Avançando mais um pouco, podemos criar uma nova **classe**, por exemplo, **Casa_Inteligente**, aproveitando a classe já concebida CASA e acrescentando novas características (**atributos**), como câmeras e sensores de presença.

A nova classe ***Casa_Inteligente*** herda todos os atributos e comportamentos da classe CASA, o que permite o reuso da classe pai CASA e agiliza a concepção da nova classe, que será mais complexa, porém sem que o princípio da atomicidade seja perdido. Se mudarmos a cor dos blocos da *Casa_inteligente*, ela muda de cor.

Agora que você já conheceu os principais conceitos e princípios de POO, será mais fácil perceber as vantagens deste paradigma, pois ele envolve a análise, o projeto e a implementação.

A POO (Programação Orientada a Objeto) aplica os conceitos de OO no desenvolvimento do código. Já a A/POO (Análise e Projeto Orientado a Objeto) aplica os conceitos de OO na análise e na elaboração do projeto, que são fases que antecedem a programação. Um instrumento de OO utilizado na análise é o caso de uso, e no projeto a UML (Linguagem de Modelagem Unificada). Sommerville explica que um projeto que venha a utilizar o paradigma orientado a objeto deve aplicar em todo o seu processo de desenvolvimento esta estratégia, atentando para as práticas utilizadas no projeto e na programação (SOMMEVILLE, 2009).-

“Quando construídos corretamente, sistemas orientados a objetos são flexíveis a mudanças, possuem estruturas bem conhecidas e proveem a oportunidade de criar e implementar componentes com alto grau de reutilização” (RAMOS, 2017, [s.p.]).

Assim, podemos identificar as seguintes vantagens na Orientação a Objetos:

- Reutilização de código.
- Utilização de um único padrão conceitual para a análise, o projeto e a implementação.
- O tempo de desenvolvimento do software é mais rápido.
- A construção de sistemas mais complexos é simplificado pelo fato de cada objeto ser simples, fácil de testar e integrar ao demais objetos. Este fator também alonga o ciclo de vida do software, pois as horas demandadas para o desenvolvimento do software são gastas na manutenção e no incremento de novas funcionalidades, e o conceito de objeto e os mesmos interagirem entre si, facilitando implementar novos recursos; logo o tempo de vida das aplicações é mais longo.
- Pode ter os custos de desenvolvimento reduzidos.

Em contrapartida, são poucas as desvantagens no paradigma orientado a objetos:

- Requer desenvolvedores que conheçam este paradigma.
- A modelagem requer uma atenção maior que a análise estruturada.
- Requer uma documentação minuciosa (detalhada).

Assimile

Classe: representação de um conjunto de objetos. Em outras palavras, é a representação da abstração do objeto com suas características e comportamentos.

Objeto: é a materialização da classe, qualquer coisa do mundo real.

Herança: aplicamos o conceito de herança quando elaboramos uma classe que herda as características e operações de outra classe.

Polimorfismo: quando objetos de um mesmo tipo de classe apresentam comportamentos diferentes. Dois carros são objetos da classe carro, porém um usa embreagem para mudar as marchas e outro não.

Encapsulamento: isola os atributos da classe e suas ações, garantindo a integridade e a ocultação dos dados e ações. O relacionamento entre as classes passa a ser feito por mensagens. Por exemplo: no objeto controle remoto da televisão, você pressiona o comando e ele transmite a mensagem para o aparelho de televisão, que executa o pedido, mas você não precisa ter conhecimento de como isso é feito.

Caro aluno, a orientação a objeto tornou se um paradigma muito utilizado no desenvolvimento de software e você pôde ter contato com os pilares deste paradigma: abstração, classe, objeto, herança, encapsulamento e polimorfismo. Agora é o momento de você começar a praticar a abstração e aplicar todos estes conceitos.

Sem medo de errar

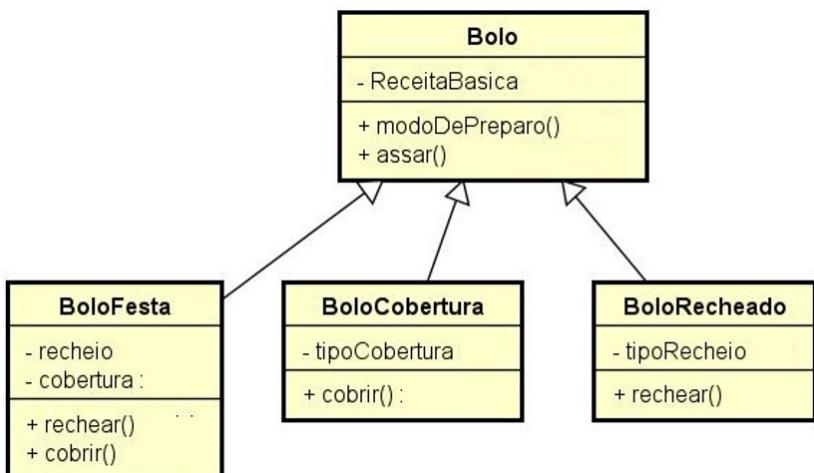
Você foi convidado para ajudar a elaborar um projeto de software para uma organização não governamental (ONG), o projeto “Empreendedores de sucesso”, que necessita de um software para uma fábrica de bolo. Sua participação no projeto, nesta primeira fase, será ajudar os empreendedores do projeto a entenderem o que é modelagem orientada a objetos, e por meio de sua explicação dos conceitos fundamentais, mostrar que existem padrões nos processos de funcionamento da fábrica.

Inicialmente simule uma reunião com os usuários que atuarão na fábrica. Você também pode pesquisar na internet sobre franquias deste tipo de comércio e estudar quais são os produtos e serviços oferecidos neste negócio, ler sobre algumas receitas de bolos, tudo para coletar dados. De posse destas informações será o momento de abstrair, definir os objetos que serão necessários e aplicar os conceitos estudados.

De acordo com o exposto pela ONG, eles pretendem ter bolos de vários tipos, tais como simples, com cobertura e de festa. Eles dispõem de uma receita padrão, algumas coberturas e recheios, que podem ser usadas em vários bolos. De posse destas informações, podemos definir as principais classes com base nestes dados:

Solução 1 - básica:

Figura 4.3 | Solução 1- Básica

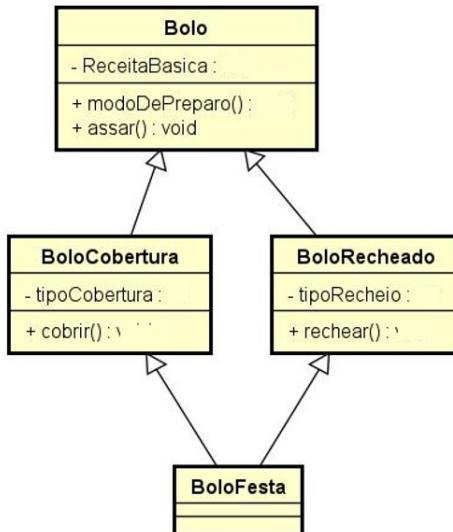


Fonte: elaborada pela autora.

Observe o desenho das classes. Há alguma coisa repetida? Será que podemos melhorar esse projeto? Reutilizar alguma coisa?

Se você respondeu sim, é porque você percebeu que o *BoloFesta* pode herdar as características de duas outras classes. Muito bem, você descobriu a chave da orientação a objeto, a reutilização de códigos já existentes e, mais ainda, o que acabamos de fazer é um processo de iterativo; repetimos a análise e observamos que é possível refinar. Veja a solução 2 – uso de herança múltipla:

Figura 4.4 | Solução 2 – Uso de herança múltipla



Fonte: elaborada pela autora.

Nas duas soluções apresentadas foram aplicados os conceitos de herança, sendo que na solução 2 a classe *BoloFesta* utilizou herança múltipla, isto é, herdou características de duas superclasses.

Caso você queira refinar mais ainda, você pode pensar em ter *modoDePreparo()* diferente para algum tipo de ingrediente específico, e neste caso aplicar o conceito de polimorfismos, mas vamos deixar este refinamento para as próximas seções, quando estudarmos UML e PU.

Para ilustrar os conceitos teóricos as suas clientes do projeto, você pode ilustrar os conceitos de objeto, atributos, métodos e herança, mostrando alguns trechos dos vídeos sobre receitas de bolos que utilizam na prática o conceito de orientação a objeto na produção culinária em escala. Orientação a objeto não é somente aplicável a software, é uma forma de modelar soluções.

No canal da Chef Léo Oliveira você pode conferir esses exemplos nos seguintes vídeos:

Faça 10 tipos de bolos caseiro com apenas uma massa – parte 1 (de 23 min a 25 min 41s) (OLIVEIRA, 2019).

Faça 10 tipos de bolos caseiro com apenas uma massa – parte 2 (do início a 9 min). (OLIVEIRA, 2019).

Cada “saquinho” da massa básica é a receita básica, um tipo de dado que está encapsulado na classe Bolo.

Faça valer a pena

1. Analise o texto:

Considere o exemplo [...] a seguir: suponha que criamos um programa de simulação do movimento de vários tipos de animais para um estudo biológico. As classes **Peixe**, **Anfíbio** e **Pássaro** representam os três tipos de animais sob investigação. Imagine que cada uma dessas classes estende a **superclasse Animal**, que contém um método *mover* e mantém a localização atual de um animal como coordenadas x-y. Cada **subclasse** implementa o método *mover*. [...] Cada tipo específico de Animal responde à mensagem de maneira única – um **Peixe** poderia nadar um metro, um **Anfíbio** poderia pular um metro e meio, e um **Pássaro** poderia voar três metros. O Programa emite (isto é, *mover*) para cada objeto de animal genericamente, mas cada objeto sabe modificar suas coordenadas x-y apropriadamente. [...] A mesma mensagem (nesse caso, *mover*) enviada a uma variedade de objetos tem ‘muitas formas’ de resultados. (DEITEL, 2010, p. 305)

O exemplo apresentado aplica vários conceitos fundamentais de orientação de objeto.

- a. Peixe, Anfíbio e Pássaro são exemplos de instâncias da classe Animal.
- b. A classe Peixe herda a classe Animal, por isso ela é chamada de superclasse.
- c. A subclasse Animal assim é chamada pois tem um relacionamento de herança com as Classes Peixe, Anfíbio e Pássaro.
- d. O método *mover* existe na Animal e foi reescrito nas suas subclasses Peixe, Anfíbio e Pássaro, resultando em uma variedade de resultados, daí o que representa um erro de programação.
- e. O conceito de herança permite criarmos subclasses que herdam as características das superclasses e promover a extensibilidade, ao modificar determinado(s) método(s) da superclasse diretamente nas subclasses, gerando, assim, comportamentos polimórficos.

- 2.** A “crise do software” em 1968–1969 provocou o surgimento de modelos para o desenvolvimento de software. Vários métodos de desenvolvimento foram criados na engenharia de softwares, e alguns paradigmas foram aplicados na programação.

Um paradigma é uma forma de fazer algo.

De acordo com as informações apresentadas no quadro a seguir, faça a associação dos Paradigmas contidos na Coluna A com suas respectivas Formas de Pensar na Coluna B.

COLUNA A	COLUNA B
I. Estruturado	1. Utilizamos abstrações para modelar os objetos que se relacionam entre si.
II. Orientado a objetos	2. Devemos nos preocupar com o objetivo e com os dados recebidos.
III. Funcional	3. Pensamos em como resolver o problema passo a passo.

Assinale a alternativa correta:

- a. I – 1; II – 2 ; III – 3.
- b. I – 2; II – 1; III – 3.
- c. I – 3; II – 2; III – 1.
- d. I – 3; II – 1; III – 2.
- e. I – 2; II – 3; III – 1.

- 3.** Considere os seguintes exemplos:

- 1. Há algumas pessoas em uma fila, aguardando atendimento por um caixa em uma loja de departamentos.
- 2. Assim que um caixa fica disponível, a primeira pessoa da fila avança para o caixa livre. As pessoas da fila são atendidas por ordem de chegada; a primeira a entrar é a primeira a sair.
- 3. Em uma padaria, há uma linha de produção de salgados. Para evitar perdas, quando um novo salgado fica pronto ele é colocado atrás do último salgado que está no escaninho, de tal modo que o salgado retirado primeiro é o mais antigo.

Cada um dos exemplos apresentados representa uma situação específica, porém é possível encontrar uma descrição genérica que funcione nos dois exemplos.

Exemplo: nos dois casos temos uma fila – primeiro que entra, primeiro que sai – independentemente se é uma pessoa ou um salgado.

Para chegar a essa descrição genérica, qual conceito de orientação a objeto foi aplicado? Assinale a alternativa correta:

- a. Objeto.
- b. Classe.
- c. Abstração.
- d. Atributo.
- e. Método.

Seção 2

Modelo do processo unificado

Diálogo aberto

Caro aluno, o Processo Unificado (PU) surgiu para melhorar o desenvolvimento de softwares com foco na A/POO (Análise e Projeto Orientados a Objetos). Este modelo de desenvolvimento de software é iterativo e adaptativo, permitindo produzir um sistema de grande porte como se fossem vários pequenos sistemas, o que diminui o risco do projeto.

Segundo Rezende, não há um modo certo de projetar um sistema, pois as variantes são muitas: tamanho, tecnologia, complexidade, habilidades, experiência da equipe e infraestrutura. Nessa busca, muitos processos foram apresentados na literatura, com suas vantagens e desvantagens (REZENDE, 2002).

Na seção anterior você conheceu sobre o paradigma de orientação a objetos e seus principais conceitos. Agora você caminhará mais um pouco no processo de desenvolvimento de software orientado a objeto e conhecerá o Processo Unificado (PU). O Processo Unificado foi um marco do desenvolvimento de software e foi desenvolvido com o objetivo de garantir a produção de software de alta qualidade. Nessa seção convidamos você a mergulhar no estudo do Processo Unificado e aprofundar seus conhecimentos sobre orientação a objetos, já que este paradigma continua a ser o mais observado na engenharia de software.

Depois de ter recebido o convite para participar do projeto “Empreendedores de sucesso” e ter desenvolvido a primeira etapa, você viu como foi interessante explicar os princípios de orientação a objetos e, ao mesmo tempo, perceber como estes conceitos ajudam a modelar os problemas, permitindo ter uma visão do funcionamento das diversas fases que comporão o projeto para a fábrica de bolo. Provavelmente você sentiu falta de alguns instrumentos para organização ou identificação dos objetos, pessoas e atividades que envolverão o projeto. Muito bem, então agora é o momento.

Nesta segunda etapa da elaboração do projeto para o desenvolvimento do software para a fábrica de bolos do programa “Empreendedores de sucesso” da ONG, você elaborará os casos de uso para a fase inicial da modelagem dos processos envolvidos na fábrica e, com os conhecimentos adquiridos sobre as características do processo unificado (PU), você deverá identificar as fases e atividades de cada processo. Além disso, você poderá aplicar o método interativo e incremental para refinar o desenvolvimento. Ao final,

você deverá apresentar ao cliente, no formato de relatório, os casos de uso para aprovação ou melhorias.

Pronto para adquirir estes conhecimentos e enfrentar mais um desafio na busca de capacitação, para poder participar de uma grande equipe de desenvolvimento de software?

Não pode faltar

Um processo de software diz respeito à maneira como produzimos software, ou seja, qual a metodologia, quais técnicas e padrões vamos adotar ao longo do processo. Dentro desse universo de possibilidades, o processo unificado (PU) representa a união de certas metodologias similares que “os três amigos”, Jacobson, Booch e Rumbaugh (2000) justificadamente chamaram de Processo Unificado (PU).

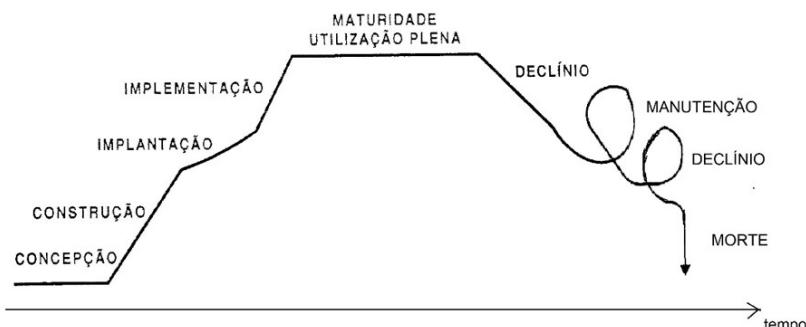
No processo unificado “Um processo define *quem* está fazendo *o quê*, *quando* e *como* alcançar um determinado objetivo” (JACOBSON; BOOCH; RUMBAUGH, 1999 *apud* PRESSMAN, 2011, p. 40, grifos do original). No PU, os elementos do processo destacados referem-se a:

- Quem (papel) está fazendo.
- O quê (artefato).
- Como (atividade).
- Quando (disciplina).

Em breve, entenderemos melhor cada um desses elementos.

Um bom planejamento, baseado nos elementos da engenharia de software, permite construir um produto, o software, com qualidade e reduz os riscos do projeto, seja em relação a atraso ou custo. O software, como qualquer produto, tem um ciclo de vida, por exemplo, como ilustrado na Figura 4.5, na qual podemos observar um ciclo com começo (concepção), meio (maturidade e utilização plena) e um fim.

Figura 4.5 – Representação gráfica do ciclo de vida do software



Fonte: adaptada de Rezende (2002, p. 49).

É importante que você não confunda ciclo de vida do produto com modelo do ciclo de vida de desenvolvimento. Ciclo de vida do produto, segundo Kotler (2018), consiste em 4 fases: introdução (*concepção*), crescimento (*desenvolvimento e implantação*), maturidade e declínio. Já modelo do ciclo de vida de desenvolvimento é a arquitetura do processo. Em engenharia de software existem alguns modelos que podem ser usados para organizar o ciclo de vida de desenvolvimento do software, por exemplo, modelo em cascata, em espiral, de prototipação, incremental, iterativo, dentre outros.

O modelo em cascata, o mais antigo, consiste em 5 fases: (i) análise e levantamento de requisitos, (ii) projeto, (iii) desenvolvimento, (iv) teste e (v) implantação. O modelo incremental é entregue em partes (módulos) ao cliente, sendo que cada módulo passa por todas as fases do modelo em cascata. Já no modelo iterativo, também chamado evolutivo, o cliente participa ativamente da análise, porém ainda não sabe exatamente especificar todos os requisitos, todas as funcionalidades que o software deve ter ou como as atividades se relacionam. Nesse caso, uma análise inicial e superficial é feita e o software implementado. Posteriormente, a análise é refinada e um novo software melhorado é entregue.

Dentro desse universo de possibilidades, encontra-se o modelo que é nosso assunto central: o Processo Unificado (PU). O PU é um modelo adaptativo, baseado no modelo incremental e que visa a construção iterativa de um software. Podemos dizer que o modelo PU aperfeiçoou o tradicional processo incremental e o iterativo, eliminando algumas desvantagens dos dois processos. Por exemplo, no modelo iterativo, às vezes, o software não termina, pois o cliente está sempre solicitando alterações e, se o processo de documentação não for adequado, o software vira uma colcha de retalhos. No caso do incremental, cada parte tem de ser concluída integralmente para se

passar à próxima. Já com o PU proposto por Jacobson, Booch e Rumbaugh (2000) os processos iterativos e incremental caminham em paralelo, permitindo que o software vá se tornando robusto a cada refinamento, num caminho para a maturidade do processo.

O PU surgiu depois da UML (Unified Modeling Language), que se originou a partir de três métodos (Booch, OMT e OOSE), todos orientados a objeto, conforme apresentado por Jacobson no prefacio do livro *El Proceso Unificado de Desarrollo de Software* (JACOBSON; BOOCH; RUMBAUGH, 2000). A UML passou a ser utilizada como uma norma pelas empresas de desenvolvimento, mas não era um processo completo para ser seguido no ciclo de desenvolvimento. A partir dessa necessidade, os mesmos participantes da equipe de desenvolvimento sistematizaram o Processo Unificado Racional (RUP) que é uma especialização, com refinamento detalhado do processo unificado. O RUP é um processo proprietário, originalmente desenvolvido na empresa RATIONAL SOFTWARE, pelos “três amigos”, e posteriormente adquirido pela IBM. Em termos gerais, podemos dizer que o PU é um processo sem algumas disciplinas presentes no RUP, porém é de uso público e contempla o ciclo de desenvolvimento do software.

Agora que já temos uma noção do PU dentro da engenharia de software, vamos conhecer melhor essa importante tecnologia. Segundo Jacobson, Booch e Rumbaugh (2000), o PU é definido por três aspectos chaves: I. **dirigido por caso de uso**, II. **centrado na arquitetura**, III. **iterativo e incremental**. Vamos explicar cada um desses aspectos chaves.

- I. O caso de uso é o fio condutor do PU, como afirmam Jacobson, Booch e Rumbaugh (2000). Nesta fase devemos conhecer **o quê** os futuros usuários necessitam e desejam. Devemos nos atentar para que os casos de usos respondam o que cada usuário necessita e não apenas as funções que o sistema precisa ter. Desta maneira o modelo de caso de uso guiará o desenvolvimento no seu projeto (**design**), na implementação e nos testes, avançando através de uma série de fluxo de trabalho (**workflow**).
- II. O segundo aspecto, centrado na arquitetura, visa dar ao engenheiro de software uma visão abrangente do sistema, como no caso do projeto de um carro, no qual há o design, a mecânica, o projeto elétrico, aerodinâmico etc. A arquitetura do software também deverá prover ao desenvolvedor estas visões generalizadas, tais como as necessidades dos usuários e objetivos estratégicos da empresa.

Reflita

O desenvolvimento pelo ciclo de vida do processo unificado (PU) é:
DINÂMICO, INCREMENTAL e ITERATIVO.

Você está aplicando o PU no desenvolvendo de um software, qual aspecto você desenvolverá primeiro?

Este é um típico caso do “ovo e da galinha”.

Ao pensar no sistema você inicialmente precisa identificar as características mais importantes, deixando os detalhes para depois; por outro lado, quais as necessidades mais importantes do sistema para cada usuário e quem participa delas?

Você deve ter em mente que estes dois aspectos devem se encaixar e saber que os casos de uso de chaves constituem as funções fundamentais do sistema. Segundo Jacobson, Booch e Rumbaugh (2000), 5% a 10% dos casos de uso representam estas funções.

Uma boa dica é começar a arquitetura pela plataforma e compreender os casos de uso mais abrangente. À medida que os casos de uso se especificam e vão ganhando maturidade, descobrem-se mais detalhes da arquitetura.

Este processo continua até que possamos considerar a arquitetura estável.

Esse é um exemplo clássico do aspecto iterativo e incremental em paralelo do PU.

III. O terceiro aspecto do PU é o iterativo e incremental. A iteração consiste em dividir o projeto em subprojetos menores e o resultado de uma iteração produz um incremento. Segundo Jacobson, Booch e Rumbaugh (2000, p. 7), “as iterações fazem referência aos passos no fluxo de trabalho e os incrementos ao crescimento do produto”. O processo iterativo no PU é controlado. Isso reduz os riscos de aumento de custos com incrementos específicos, os prazos são melhor controlados e também acelera o ritmo de desenvolvimento, pois reconhece um aspecto ignorado: dificilmente em um sistema complexo, no início, é possível definir totalmente os requisitos e necessidades do usuário.

Assimile

INTERAÇÃO ≠ ITERAÇÃO

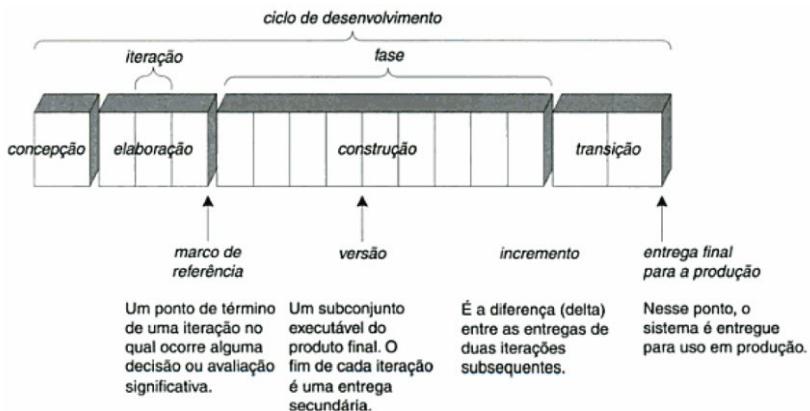
Não confunda interação que significa relacionar-se, que é o ato de interagir com outros elementos, com **Iteração** que é uma **repetição** de ações sucessivas sobre determinadas atividades ou objetos, e o resultado é um novo objeto resultado da iteração anterior.

O ciclo de vida do PU é uma série de repetições ao longo da vida do sistema, sendo que cada ciclo completo resulta em uma versão do software, por sua vez cada ciclo é composto por 4 fases:

- **Concepção:** definirá a visão geral do projeto, o escopo e os requisitos iniciais.
- **Elaboração:** é uma visão mais refinada dos requisitos e da arquitetura, análise de riscos e estimativas.
- **Construção:** é o momento de desenvolvimento do sistema, começando pelos elementos mais fáceis, e inicia-se a preparação para a implantação.
- **Transição:** é a fase de implantação do sistema, ou seja, a entrega.

Observe na Figura 4.6 que o ciclo de desenvolvimento (concepção + elaboração + construção + transição) termina com a entrega de uma versão do sistema, pronta para ser implementada em produção. Todo esse ciclo é composto de muitas iterações, segundo LARMAN (2007).

Figura 4.6 | Ciclo de vida do PU



Fonte: Larman (2007, p. 62)

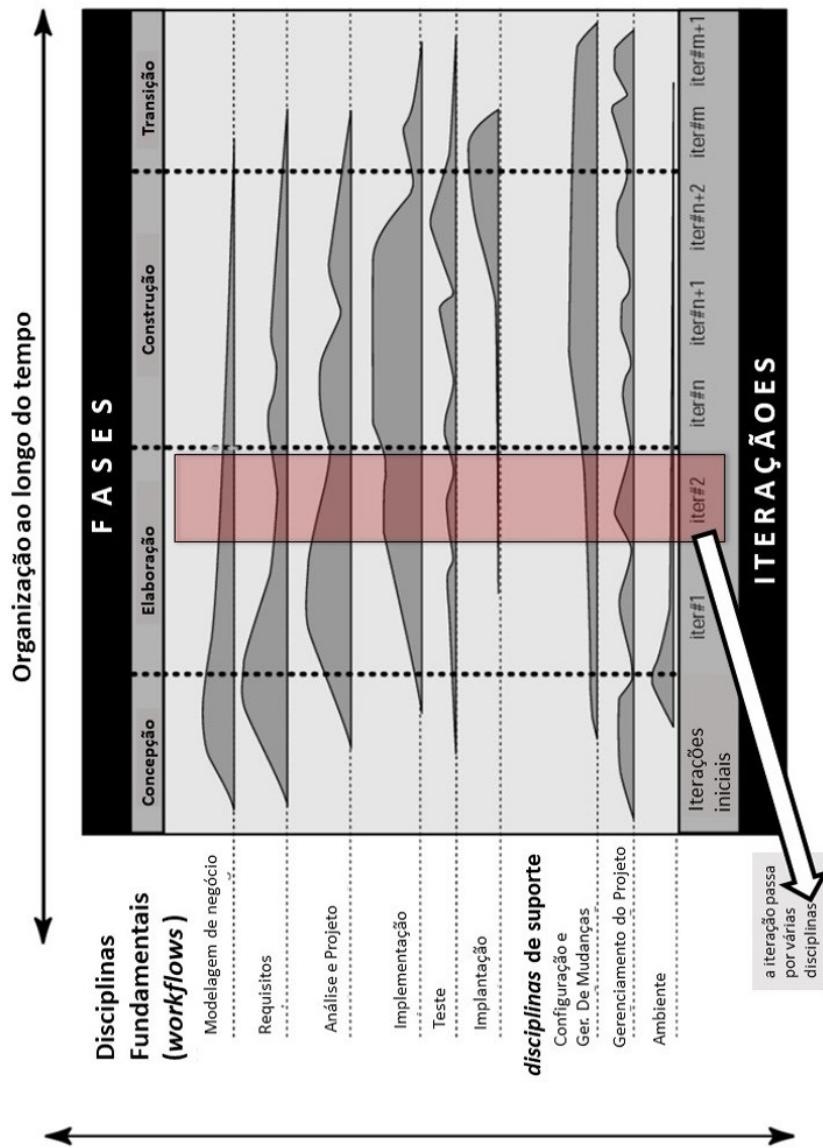
O PU descreve um processo por quatro elementos básicos: papel (**quem**), artefato (**o quê**), atividade (**como**) e disciplina (**quando**):

- **Papel (worker, trabalhador):** é a identificação das responsabilidades de cada indivíduo (quem faz o quê), o papel do trabalhador naquele momento, o ator da cena. Ao longo do processo, um *worker* pode ter várias responsabilidades e desenvolver uma série de atividades.

- **Artefato:** é o termo utilizado para identificar qualquer produto de trabalho, seja código, esquema de banco de dados, diagramas, modelos etc. É o produto que o *worker* gera.
- **Atividade:** é a tarefa executada pelo *worker* com o objetivo de produzir ou alterar um artefato.
- **Disciplina** (*workflow* – fluxo de trabalho): é a sequência de atividades que gera um resultado significativo. Os fluxos estão associados a três perspectivas: dinâmica (tempo), estática (atividades) e boas práticas. Em cada uma encontramos um conjunto de disciplinas.

Segundo Jacobson, Booch e Rumbaugh (2000), as disciplinas fundamentais, também chamadas fluxo de trabalho do PU são: (i) **modelagem de negócio**, (ii) **requisitos**, (iii) **análise & projeto**, (iv) **implementação**, (v) **teste e implantação**. Este núcleo principal será o nosso foco. Mas, também há as disciplinas de suporte: gerenciamento do projeto, gerenciamento de configurações & mudanças e ambiente. Na Figura 4.7 você pode observar como as disciplinas são desenvolvidas ao longo das fases do ciclo de vida do desenvolvimento e nas diversas iterações (minissistemas). Cada ponto de iteração é um marco no PU e finaliza com a entrega de um incremento. Neste ponto é avaliado se os objetivos foram alcançados e, se necessário, ajustes são realizados. Embora para cada iteração todas as disciplinas podem estar presentes, o tempo dedicado a cada uma muda de iteração para iteração. Por exemplo, na Figura 4.7 podemos perceber que na primeira iteração temos mais tempo dedicado às fases concepção e elaboração. Já nas últimas iterações, outras disciplinas ganham mais dedicação. O que é natural no desenvolvimento de um sistema.

Figura 4.7 | Fluxo das iterações ao longo do tempo e das disciplinas



Fonte: IBM (2005, p. 3, tradução nossa).

Vamos agora analisar o que cada uma destas disciplinas contempla.

- I. Modelagem de negócio:** o objetivo é documentar quais os objetivos de negócio estão envolvidos no processo e são analisados, buscando entender como o negócio deve apoiar os processos associados.

Muitos projetos podem optar por não fazer modelagem de negócios.

Os casos de uso de negócios são utilizados nessa fase.

- II. Requisitos:** Tem por objetivo descrever o que o sistema deve fazer. Tem como base os casos de uso para identificar os requisitos. Nessa fase surge um importante elemento na engenharia de software, os atores. Segundo artigo publicado pela IBM: “Para isso, devem ser identificados os atores envolvidos, todos os quais representam os usuários e qualquer outro sistema que venha a interagir com o sistema em desenvolvimento” (IBM, 2005, p. 4, tradução nossa). O foco está nas funcionalidades do sistema e tanto desenvolvedores como o cliente deverão concordar com a descrição dos requisitos.

	<p>Caso de uso na prática</p> <p>A definição dos casos de uso consiste em escrever texto. Um formulário para nortear os principais conceitos a serem levantados é importante.</p> <p>Para consultar um exemplo disso, acesse o QR Code ou o link: http://cm-kls-content.s3.amazonaws.com/ebook/embed/qr-code/2020-1/analise_modelagem_sistemas/u4/s2/lkls201_u4s2_ana_mod_sis_QRCode1.docx.</p>
---	---

- III. Análise e projeto:** estas disciplinas geram os modelos do sistema objetivando as implementações futuras. Em conjunto, essas fases melhoram a compreensão dos requisitos funcionais, definido nos casos de uso. Vale lembrar que “os requisitos funcionais são declarações de serviços que o sistema deve fornecer, como o sistema deve reagir a entradas específicas e como o sistema deve se comportar em determinadas situações” (SOMMERVILLE, 2007, p. 80).

Segundo Larman, “a análise enfatiza a investigação do problema e dos requisitos, em vez de uma solução” (LARMAN, 2005, p. 34), ou seja, como será usado e quais serão as suas funções, deixando para a implementação o modo como fazer isso.

O projeto dá ênfase à solução conceitual, ou seja, cria os modelos para satisfazer os requisitos. O projeto é para o desenvolvedor e não para o usuário. Consiste em subsistemas bem definidos, com classes estruturadas em pacotes e deverá representar os componentes a serem implementados. Vale lembrar que um projeto focado na arquitetura

representará uma série de visões do sistema, ou seja, abstrações das características mais importantes, facilitando futuras mudanças quando os requisitos funcionais sofrerem alterações.

Refita

Análise e projetos são úteis quando?

Quando você inicia o desenvolvimento de um sistema, é muito importante ter clareza do problema para o qual você irá criar a solução

Fazer a coisa certa está totalmente conectado com entender o problema que queremos resolver, quais são as pessoas e objetos impactados por esse problema, como essas pessoas estão tentando resolver atualmente e quanto deste problema queremos resolver.

Esta compreensão gera clareza e isso vem bem antes de criar a solução, esta é a fase da análise. Nesta fase não importa muito como vamos resolver e sim a clareza sobre qual problema vamos resolver. *Logo, comece fazendo a coisa certa: uma boa análise do problema.*

Encontrado a solução certa, a partir da análise correta, será o momento de fazer as coisas corretamente, isto é, o projeto. O projeto envolve conhecimento técnico, processos e metodologias adequadas. Portanto é necessário um domínio adequado destes conhecimentos, preparação para fazer um projeto correto, isto é, fazer certo.

Se você permite questionar constantemente, se é a coisa certa a ser feita e aprende nesse processo a buscar resultados, você passa a aceitar que o aprendizado é importante, que você pode mudar de ideia ou adaptar o que está fazendo, pois o que importa é resolver o problema. Essa é uma das propostas do PU iterativo-incremental, reavaliar e evoluir.

Portanto, análise e projeto são úteis, pois permitem saber se:

- Você fez a coisa certa (*análise*).
- Se você fez certo a coisa (*projeto*).

IV. Implementação: o seu foco está direcionado à construção do software, o desenvolvimento do código, a famosa “mão na massa”. Também prepara os primeiros testes, chamados *beta teste*.

V. Testes: nesta etapa deve ser descrito quais os procedimentos a serem avaliados.

Segundo as recomendações do documento *RUP: boas práticas para o desenvolvimento de software* (IBM,2005), os objetivos do teste são: verificar a interação entre objetos e componentes, verificar se todos os requisitos foram implementados corretamente e identificar e garantir que os defeitos sejam resolvidos antes da implantação do software.

VI. Implantação: o objetivo do fluxo de trabalho da implantação é entregar o produto com êxito para os usuários, e envolve a aceitação formal do software pelo cliente.

Entre as atividades do fluxo de trabalho da implantação temos a distribuição, instalação, migração dos dados e de software já existente.

As demais disciplinas estão associadas ao suporte e foram incluídas no Processo Unificado Racional (RUP) para suprir disciplinas não contempladas pelo PU, são elas: gerenciamento de mudanças e configurações, gerenciamento de projeto e ambiente. Basicamente visam a maturidade do projeto.

Como exposto nesta seção, vimos que o PU foi um marco na engenharia de software, e voltado para o **paradigma orientado a objeto**, sendo uma característica importante o fato de ser iterativo e incremental, assim o usuário não mais aguarda a conclusão total do software para iniciar o uso do sistema e, ao término do desenvolvimento, praticamente não há erros. O cuidado a tomar tanto no PU quanto no RUP é a com a atualização da documentação e a definição de marcos precisos para cada iteração.

Nesta seção você pôde conhecer a importância de um processo organizado, documentado e sistematizado para todo o ciclo de vida e desenvolvimento de um projeto de software, em específico o processo unificado (PU) e, ao ter contato com a história da evolução do PU, observar que as diferenças entre o processo unificado racional (RUP) e o PU estão em que o RUP cobre as disciplinas de suporte, voltadas para a maturidade do produto, e é um framework proprietário da IBM nos dias atuais.

Você conheceu as fases do PU e suas disciplinas (fluxo de trabalho). Note que esse processo é muito indicado a grandes projetos nos quais há o envolvimento de vários desenvolvedores na equipe.

Os conhecimentos adquiridos nesta seção, você poderá aplicá-los ao participar de equipes de desenvolvimento de software.

Sem medo de errar

Agora é hora de pensar em tudo que foi estudado e começar a organizar o processo que melhor irá se adaptar para o projeto da fábrica de bolo. Nessa etapa do desafio, você deve elaborar os casos de uso para a fase inicial da modelagem dos processos, com base no seu conhecimento sobre o Processo Unificado (PU).

Quais pontos podem e devem ser levantados nessa fase inicial da modelagem dos processos? É preciso ter em mente que para desenvolver um

caso de uso é preciso conhecimento do negócio e do problema a ser resolvido. Uma boa dica é começar a arquitetura do sistema pela plataforma e compreender os casos de uso mais abrangente. À medida que os casos de uso se especificam e vão ganhando maturidade, descobre-se mais detalhes. Dificilmente, no início do projeto de um sistema complexo, consegue-se definir totalmente os requisitos e necessidades do usuário.

Para nos guiar nesse desenvolvimento, vamos começar retomando os principais pontos que devem ser seguidos no PU. O caso de uso é o fio condutor do PU e você deverá identificar em cada um, quatro etapas:

1. Quem participa do caso de uso, O PAPEL de cada usuário, suas responsabilidades.
2. O que será gerado pelo processo, OS ARTEFATOS. São as funcionalidades do caso de uso.
3. Como serão realizadas as ATIVIDADES para gerar os artefatos
4. Quando serão realizadas as atividades, é a identificação precisa da sequência de passos para realizar cada atividade.

Para a fase inicial do projeto da fábrica de bolo, já conseguimos identificar os seguintes papéis de usuários: cliente, fornecedor, vendedor, sistema e-commerce, sistema ERP, funcionário da produção e gerente.

Também já conseguimos identificar os seguintes artefatos: relatório das entrevistas (DOC_ENTREVISTAS.docx), receitas dos bolos (DOC_RECEITAS.docx) e relação das permissões dos usuários versus funções do sistema (DOC_PERMISSOES.docx).

Quanto às atividades para gerar os artefatos, podemos identificar: cadastrar os usuários, cadastrar e manter as receitas de bolo, registrar/associar as permissões dos usuários com as atividades.

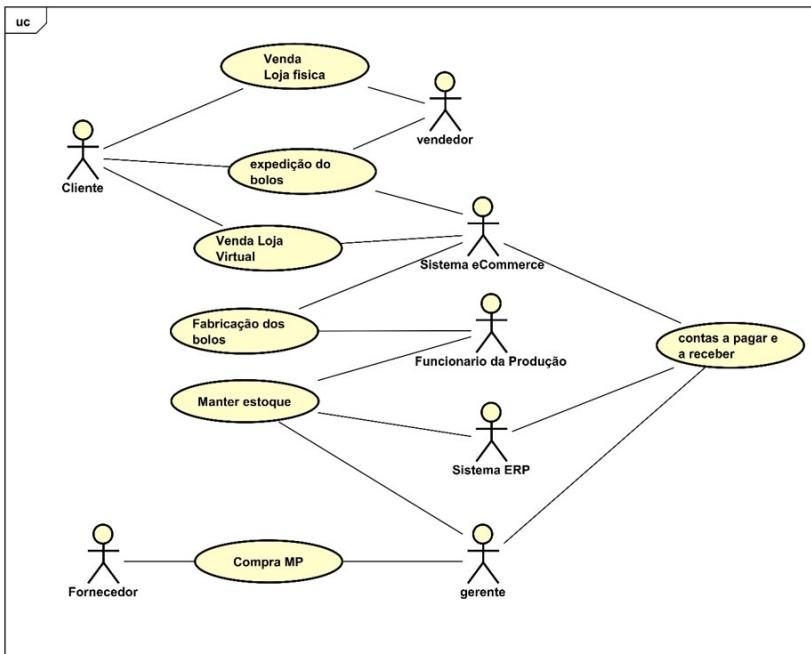
Como esboço inicial, podemos ter a seguinte sequência de atividades: solicitação de compra, separação do produto, expedição do produto.

Agora que já identificamos os principais elementos dessa fase inicial, seguindo as quatro etapas da PU, vale lembrar que para o PU funcionar é importante definir marcos para cada processo iterativo e incremental. Cada processo iterativo é um ciclo de vida completo no PU, como a fase da concepção, elaboração, construção e transição. Isto facilitará muito a documentação do seu trabalho, mesmo que você não venha a participar das iterações seguintes tudo estará documentado.

A Figura 4.8 apresenta uma proposta para o diagrama de caso de uso (UML). Nessa Figura, podemos observar os atores: fornecedor, cliente,

vendedor, sistema de e-commerce, funcionário da produção, sistema ERP e gerente. Veja que temos tanto pessoas físicas, como sistêmicas para os atores. De acordo com a Figura 4.8, a venda ao cliente pode ser feita pela loja física ou virtual, o vendedor é responsável pela expedição dos produtos adquiridos tanto via loja física como virtual. Já quando o produto é expedido o sistema de e-commerce gerar informações sobre contas a receber. O gerente é responsável pela solicitação de compra aos fornecedores e registrar as entradas dos produtos no estoque, bem como o registro das contas a pagar.

Figura 4.8 | Proposta de diagrama de caso de uso (UML)



Fonte: elaborada pela autora.

Essa primeira visão do sistema, apresentada na Figura 4.8, permite nortear tanto o desenvolvedor como o cliente, para as decisões que definirão a arquitetura, os investimentos e prioridades do projeto de desenvolvimento. Por exemplo, o ator ERP apresentado no caso de uso deverá ser um sistema legado, isto é, desenvolvido internamente na ONG, ou deverá ser um produto de terceiros? Caso a opção seja por um produto de terceiros, como será a integração com o e-commerce? Veja que essa é uma decisão estratégica e caberá ao profissional responsável apresentar as opções ao cliente para o

ajudar na tomada de decisão em conformidade com os objetivos estratégicos da empresa, tais como recurso financeiro, recursos humanos, fator tempo, segurança, necessidade de disponibilidade do sistema etc.

Como você pode ver, elaborar um projeto de sistemas não é uma tarefa trivial, é preciso ter dedicação, foco e conhecer os processos para oferecer ao cliente a melhor opção.

	A documentação de um projeto é parte essencial da modelagem. Para consultar o documento completo elaborado para essa fase inicial da modelagem dos processos, com base no PU, acesse o QR Code ou link: http://cm-cls-content.s3.amazonaws.com/ebook/embed/qr-code/2020-1/analise_modelagem_sistemas/u4/s2/lcls201_u4s2_mod_sis_QRCode2.docx
---	--

Faça valer a pena

1. “O Rational Unified Process (RUP) fornece a cada membro da equipe as diretrizes, os modelos e ferramentas necessários para que toda a equipe aproveite ao máximo, entre outras, as seguintes práticas recomendadas:

1. Desenvolver software iterativamente.
2. Gerenciar requisitos.
3. Usar arquiteturas baseadas em componentes.
4. Modelar visualmente o software.
5. Verificar a qualidade do software.
6. Controlar alterações no software” (IBM, 2005, p. 1, tradução nossa).

Com o objetivo de alcançar as boas práticas recomendadas para um bom desenvolvimento de um software, assinale a alternativa que apresenta corretamente o fluxo de trabalho proposto no PU.

- a. Concepção, elaboração, construção e transição.
- b. Disciplina, requisitos, implementação, teste e entrega.
- c. Modelagem de negócio, análise & projeto, implementação, teste e entrega.
- d. Modelagem de negócio, requisitos, análise & projeto, implementação, teste e implantação.
- e. Requisitos, análise & projeto, implantação, teste e implementação se necessário.

2.

Um processo de desenvolvimento de software descreve uma abordagem para a construção, implantação e, possivelmente, a manutenção de um software. O Processo Unificado (PU) surgiu como um processo iterativo popular para o desenvolvimento de software, visando à construção de sistemas orientados a objetos. Em particular, o Processo Unificado da Rational ou RUP (de Rational Unified Process), um refinamento detalhado do PU, é muito adotado. (LARMAN, 2007, p. 46)

Com relação ao PU, analise as afirmações a seguir:

- I. PU é um processo somente interativo.
- II. PU é flexível e pode ser aplicado sem que seja necessário construir todos os artefatos indicados no PU.
- III. PU é centrado em casos de uso e na arquitetura.

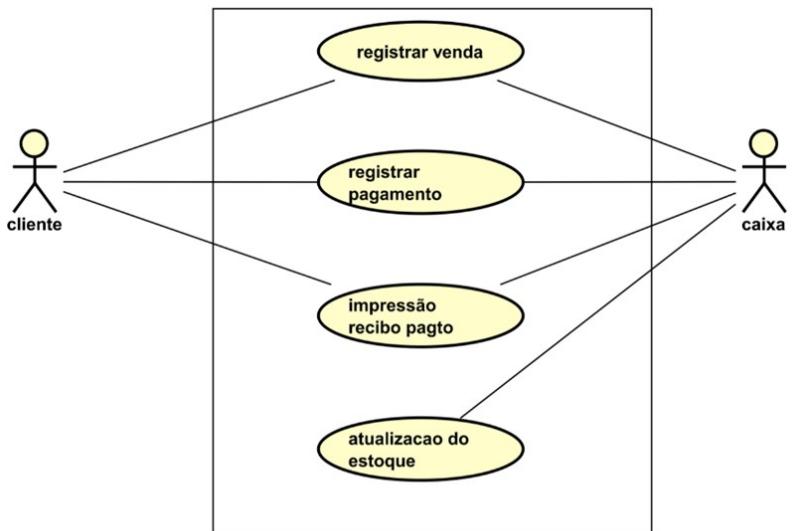
É correto apenas o que se afirma em:

- a. I e III, apenas.
- b. I, II e III.
- c. II, apenas.
- d. II e III, apenas.
- e. I e II, apenas.

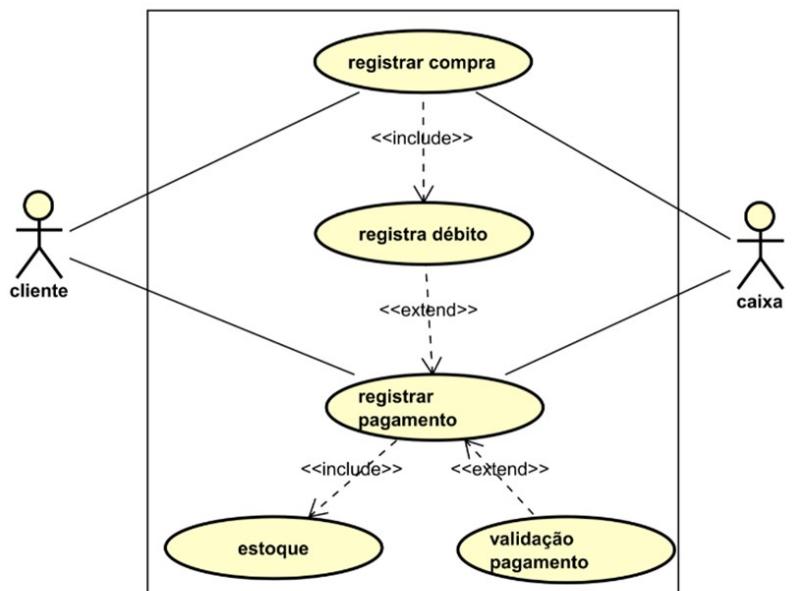
3. Um cliente vai ao caixa para o pagamento dos produtos que deseja comprar. O caixa registra no sistema os produtos comprados; após o sistema mostrar o total da compra, o cliente informa os dados para pagamento, que serão validados e registrados no sistema. O sistema atualiza o estoque e o cliente recebe o comprovante de pagamento com o detalhe das compras. O cliente sai com os produtos adquiridos.

Com base na sequência proposta pelo texto, assinale a alternativa que apresenta corretamente a figura que a descreve:

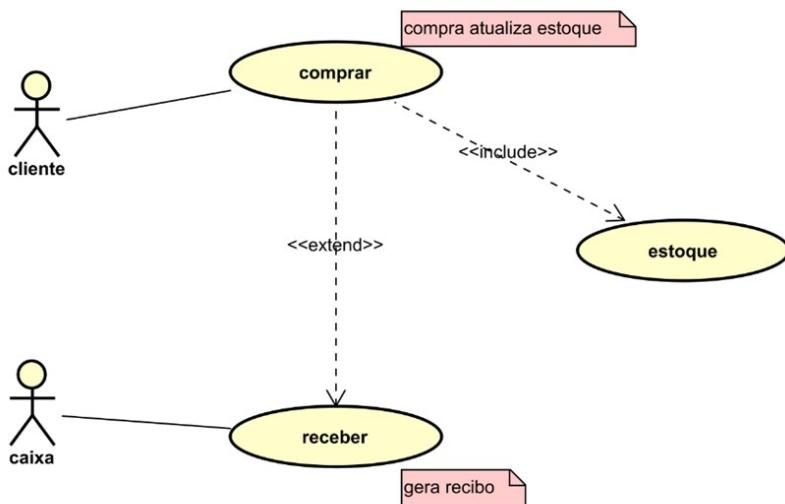
a.



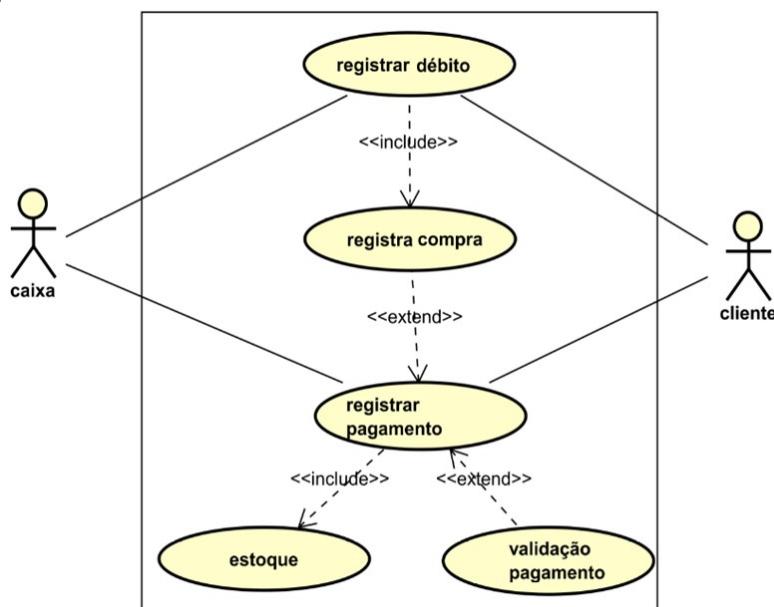
b.



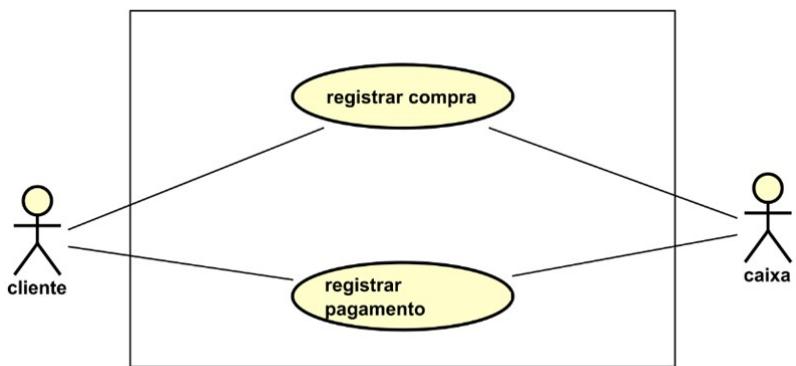
C.



d.



e.



Seção 3

Métodos orientados a objetos

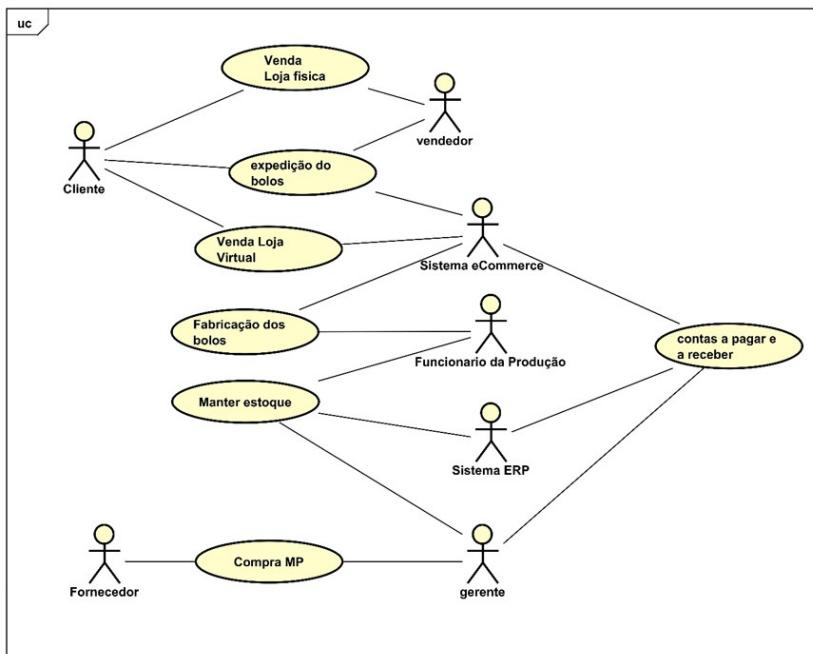
Diálogo aberto

Caro aluno, nesta seção estudaremos a linguagem que unificou os métodos para modelagem de software orientado a objetos, a UML (Linguagem Unificada de Modelagem ou *Unified Modeling Language*). Essa linguagem permitiu aos desenvolvedores e engenheiros de software um padrão único para elaborar a modelagem de programas e projetos de desenvolvimento.

Iniciamos esta unidade com o estudo do paradigma orientado a objeto, que revolucionou a forma de pensar o desenvolvimento do código. Porém, a análise estruturada com seus DFDs (diagramas de fluxo de dados) e outras ferramentas não atendiam às novas necessidades da A/POO (Análise e Projeto Orientados a Objetos). O processo unificado (PU) e o RUP (Processo Unificado Racional) vieram ao encontro dessa necessidade, apresentando um método bastante robusto para acompanhar todo o ciclo de desenvolvimento e da vida de um software. Agora será o momento de ter contato com os diagramas propostos pela UML, que trouxeram para engenharia de software o instrumental para modelar os programas e projetos de acordo com o paradigma orientado a objeto. Além de conhecer os principais diagramas de UML, você também terá uma breve introdução do que vêm a ser métodos ágeis e quais os mais utilizados atualmente.

Para que você possa aplicar na prática os conhecimentos com os quais terá contato nesta seção, você poderá unir a metodologia de processo unificado (PU) e a UML (*Unified Modeling Language*) na continuidade da elaboração do projeto de software para a fábrica de bolos do projeto “Empreendedores de Sucesso”. Nessa etapa, você deverá escolher os diagramas necessários para modelar o sistema, a partir do esboço do caso de uso inicial apresentado na Figura 4.8, e elaborar os refinamentos necessários, desenvolvendo os diagramas UML que forem essenciais para conclusão e apresentação do projeto ao cliente.

Figura 4.8 | Caso de uso geral Projeto Fábrica de Bolo



Fonte: elaborada pela autora.

Vale lembrar que atualmente você faz parte do projeto com outros programadores e analistas, porém, em breve, outros voluntários da ONG participarão do projeto e outros sairão, inclusive você. Assim, manter uma documentação precisa e atualizada desse projeto de software será muitíssimo importante para garantir sua qualidade e manutenção.

Preparado para desbravar esses novos conhecimentos e aplicá-los na construção dos diagramas UML para modelagem desse software?

Não pode faltar

Com a introdução do paradigma orientado a objeto, surgiu a necessidade de métodos específicos voltados para análise e projetos orientados a objetos (A/POO). Assim, surgiram vários métodos para modelar sistemas orientados a objetos. Todavia, o mercado ansiava por uma norma única para facilitar a modelagem e a manutenção. Segundo Jacobson, Booch e Rumbaugh (2000), no início da década de 1990 o mercado estava dividido entre estes métodos, então, a empresa Rational Software decidiu reunir Grady Booch, inventor

da metodologia Booch Methodology, James Rumbaugh, da empresa OMT, e Ivar Jacobson da Rational. Em conjunto, eles desenvolveram uma linguagem para modelagem de sistemas que foi nomeada de *Unified Modeling Language* – UML, a qual passou a ser utilizada na maioria das empresas por suas equipes de desenvolvimento. A especificação 1.1 da UML foi disponibilizada em 1997. Surgia, assim, um padrão para modelagem orientada a objetos. A Figura 4.9, disponível no QR Code, apresenta algumas contribuições dos diversos métodos para a especificação das primeiras versões da UML.

	Para ver a evolução da UML, comparativamente, com os métodos que serviram de base para o seu desenvolvimento (Figura 4.9) e a evolução das suas versões ao longo da sua existência (Figura 4.10), acesse o QR Code ou o link: http://cm-cls-content.s3.amazonaws.com/ebook/embed/qr-code/2020-1/analise_modelagem_sistemas/u4/s3/lkcls201_u4s3_ana_mod_sis_QRCode1.docx
---	---

A UML faz uso de uma linguagem gráfica, o que nos permite visualizar com mais facilidade os objetos e suas interações (relacionamentos), bem como construir, especificar e documentar os artefatos gerados por um software.

Assimile

Basicamente a UML é composta de diagramas. Veja a definição de Fowler para UML:

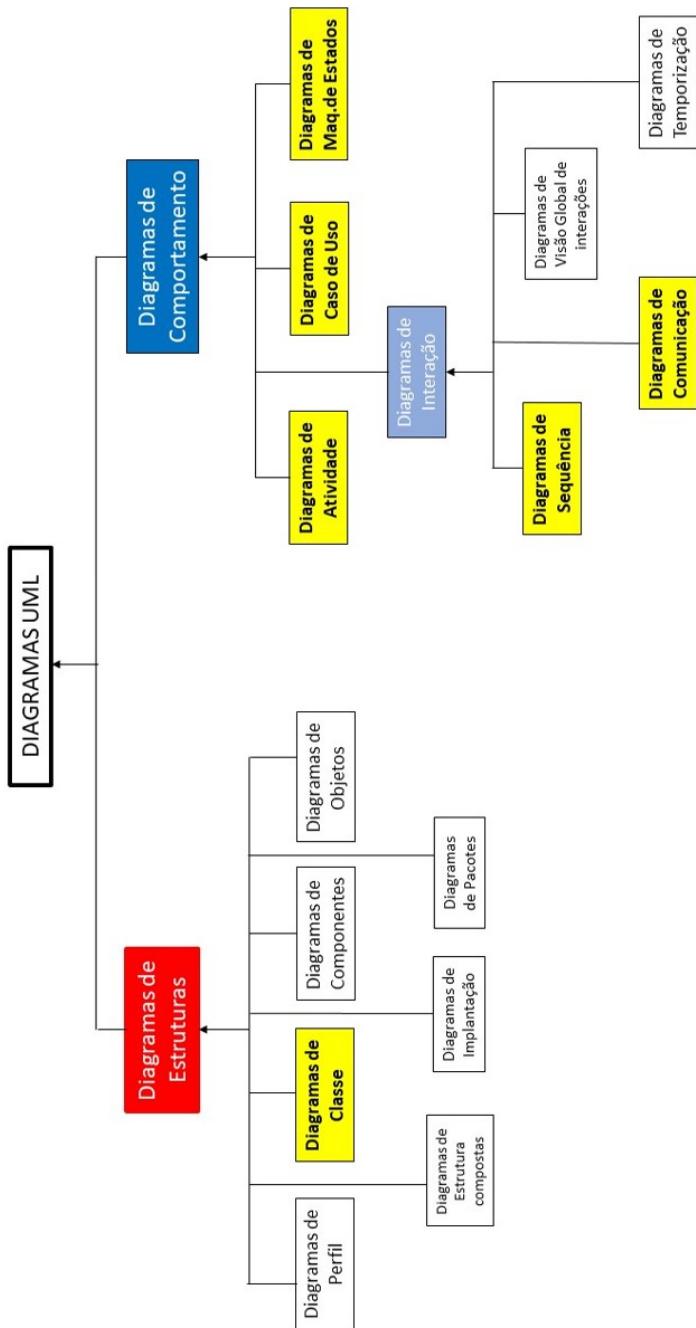
UML (Unified Modeling Language) é uma família de notações gráficas, apoiada por um metamodelo único, que ajuda na descrição e no projeto de sistemas de software, particularmente daqueles construídos utilizando o estilo orientado a objetos (OO). (FOWLER, 2005, p. 25)

As especificações da UML são controladas pela OMG (*Object Management Group*), um consórcio internacional sem fins lucrativos. A Figura 4.10, disponível no quadro do QR Code, apresenta a evolução das versões da UML. Algumas mudanças significativas ocorreram da versão 1.x para a 2.0; por exemplo, o diagrama de atividade sofreu uma revisão significativa, o diagrama de colaboração passou a ser denominado de diagrama de comunicação. Desde a primeira versão já se passaram vinte anos. A versão atual é a 2.5.1, a qual apresentaremos a você e passaremos a denominar daqui para frente simplesmente de UML 2.5. Segundo Cris Kobryn, 20% da UML ajuda você a fazer 80% do seu trabalho (2005 *apud* FOWLER, 2005, p. 7). Portanto, é importante dedicar uma atenção a esses 20% fundamentais para o domínio da UML e, especialmente, para os diagramas marcados na Figura 4.11.

A versão 2.5 apresenta 14 diagramas subdivididos em duas categorias: **diagramas de estrutura** e **diagramas de comportamento**. Veja que na Figura 4.11 os diagramas de estrutura representam as estruturas estáticas do sistema por meio de objetos, relações e atributos. Eles proporcionam uma visão da arquitetura e os aspectos funcionais globais do sistema, além de conceitos de implementação. Seu foco não são os detalhes nem o tempo, pois seu objetivo é mostrar como os objetos se relacionam. Já os diagramas de comportamento representam os aspectos dinâmicos do sistema, que são as mudanças que ocorrem no sistema com o passar do tempo, por meio da comunicação entre os objetos e das mudanças de estados internos e/ou externos no sistema. Entre os diagramas de comportamento há uma subdivisão, denominada diagrama de interação, em que encontramos quatro diagramas. Na Figura 4.11 destacamos alguns diagramas, os quais estudaremos nesta seção.

A existência de tantos diagramas tem como objetivo permitir visões múltiplas do sistema a ser modelado, como afirma Guedes (2011). Vamos conhecer um pouco mais sobre alguns dos diagramas mais importantes da UML, destacados na Figura 4.11:

Figura 4.11 | Classificação dos diagramas UML 2.5



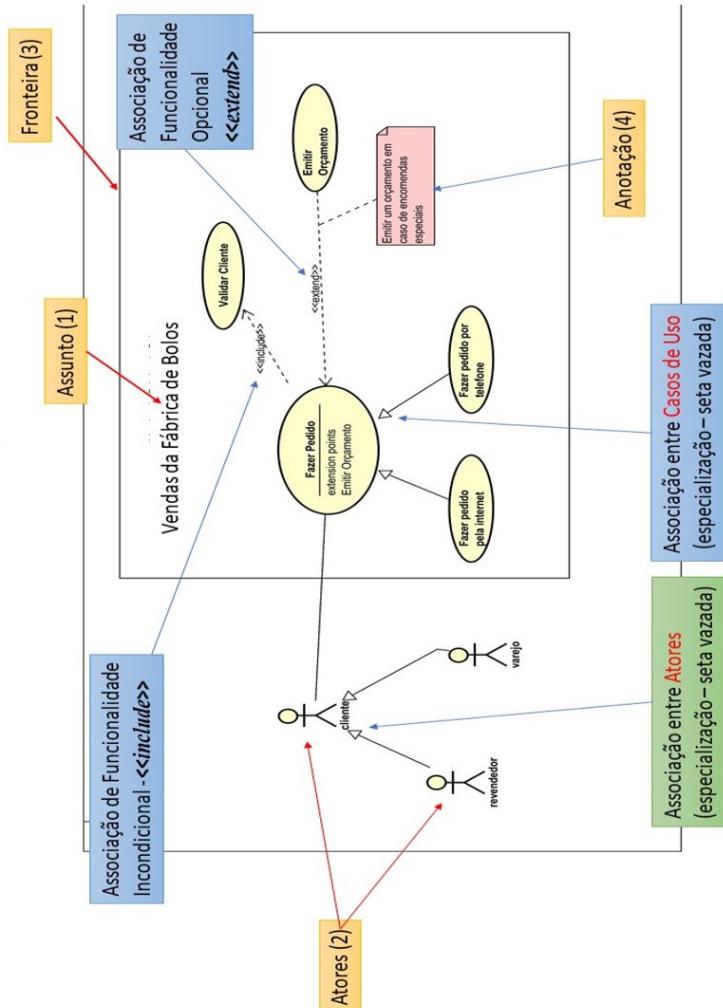
Fonte: adaptada de OMG® (2017).

1. Diagrama de caso de uso

Esse diagrama é bastante utilizado. O diagrama de caso de uso fornece uma visão geral dos objetivos que os usuários (os atores) desejam alcançar utilizando o sistema.

Os elementos mais importantes são os atores, os relacionamentos e o fluxo de eventos. A Figura 4.12 apresenta os principais elementos do diagrama de caso de uso.

Figura 4.12 | Elementos básicos do diagrama de caso de uso



Fonte: captura de tela da ferramenta Astah elaborada pela autora.

Lembre-se

Para entender o diagrama de classe é importante você relembrar os conceitos fundamentais de orientação a objeto.

Uma **classe** é uma representação abstrata de um objeto.

Um **atributo** é uma característica de uma classe.

Um **método** é um comportamento de uma classe.

Relembando: a classe cliente tem atributos como nome, endereço, telefone, etc., e comportamentos como fazer pedido, efetuar pagamento, aprovar orçamento.

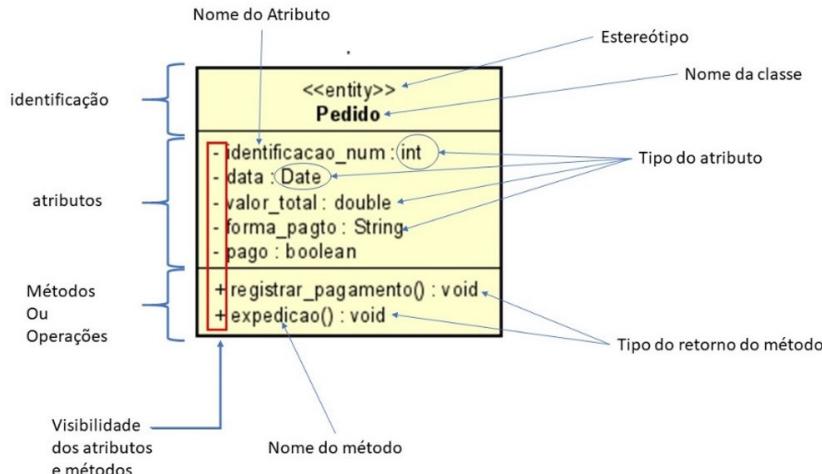
E o “Pedro” é um objeto do tipo cliente.

2. Diagrama de classe

O diagrama de classe, segundo Guedes (2011, p. 31), “[...] define a estrutura das classes utilizadas pelo sistema, determinando os atributos e métodos que cada classe tem, além de estabelecer como as classes se relacionam e trocam informações entre si”.

A representação da classe é um retângulo dividido em três partes: o nome da classe; os atributos e; os métodos, como mostra a Figura 4.13.

Figura 4.13 | Representação de uma classe em UML



Fonte: captura de tela da ferramenta Astah elaborada pela autora.

Um mecanismo de extensibilidade da UML bastante utilizado nos diagramas de classe são os estereótipos. A finalidade de um estereótipo é permitir classificar elementos do diagrama que tenham algo em comum

entre si. Os estereótipos podem ser definidos pelo desenvolvedor ou predefinidos. Os estereótipos predefinidos são nativos na linguagem. A UML dispõe de 50 estereótipos predefinidos, segundo a especificação OMG® (2017), mas os estereótipos predefinidos mais comuns são três:

- <<**entity**>> Estereótipo identidade, que identifica classes de persistências. Essas classes armazenam dados recebidos pelo sistema. Além da notação textual, <<**entity**>>, também pode ser representada pelo símbolo indicado na Figura 4.14 pela classe Produto.
- <<**boundary**>> Estereótipo fronteira, o qual identifica uma classe de fronteira. Essas classes servem de comunicação entre atores externos e o sistema. Sua representação gráfica está indicada na Figura 4.14 pela *interface_Sistema_da_Loja*.
- <<**control**>> Estereótipo de controle. Esse estereótipo geralmente é formado por classes que indicam regras de negócio. Sua representação gráfica é apresentada na Figura 4.14 pelo *controlador_Sistema_da_loja*.

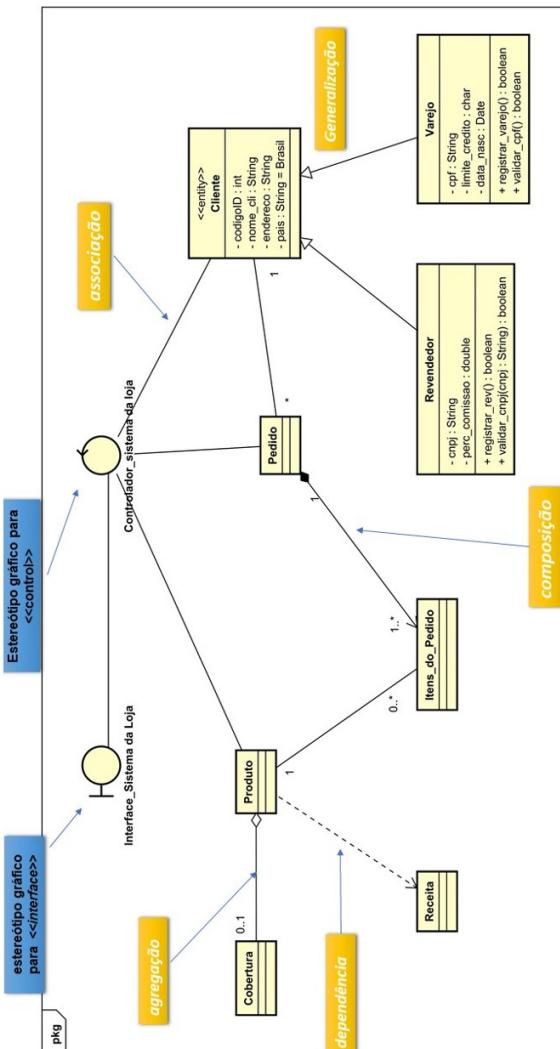
As classes relacionam-se entre si, e os tipos de relacionamentos possíveis especificados na UML (OMG®, 2017) são:

1. **Dependência:** é o tipo de relacionamento mais fraco entre duas classes, chamado de relação semântica entre duas classes, na qual uma alteração na classe independente pode afetar a classe dependente. Veja o exemplo na Figura 4.14, em que há um relacionamento do tipo dependência entre as classes Produto e a classe Receita. Ambas existem independentemente, porém uma alteração na classe Receita poderá afetar a classe Produto.
2. **Associação:** este é o tipo de relacionamento mais comum, e indica que a classe A tem uma relação com a classe B. É um relacionamento genérico.
3. **Agregação:** é uma associação específica, em que a classe filha pode existir independentemente da classe pai. É o caso do relacionamento de agregação mostrado na Figura 4.14 entre a classe Produto (pai) e a classe Cobertura (filha), em que, se um objeto do tipo Produto deixa de existir, o objeto do tipo Cobertura (filha) continua a existir.
4. **Composição:** é uma associação específica em que, se o objeto da classe pai é destruído o outro objeto associado, não fará sentido a existência da classe filha. Exemplo: entre as classes Pedido e Item_de_Pedido temos um relacionamento de composição, pois os Itens_de_Pedido são partes do Pedido. Se excluirmos o Pedido, o Item_de_Pedido também deve ser excluído. Veja a representação deste relacionamento na Figura 4.14.

5. **Generalização/especialização:** indica que a classe filha herda as características da classe pai, também conhecida como especialização da classe. As classes Revendedor e Varejo são classes filhas da classe Cliente, conforme mostrado na Figura 4.14.

6. **Multiplicidade:** indica quantas instâncias dos objetos estão envolvidos na associação, como está mostrado na Figura 4.14.

Figura 4.14 | Diagrama de classe representando a venda de bolos com as indicações dos tipos de relacionamentos entre as classes



Fonte: captura de tela da ferramenta Astah elaborada pela autora.

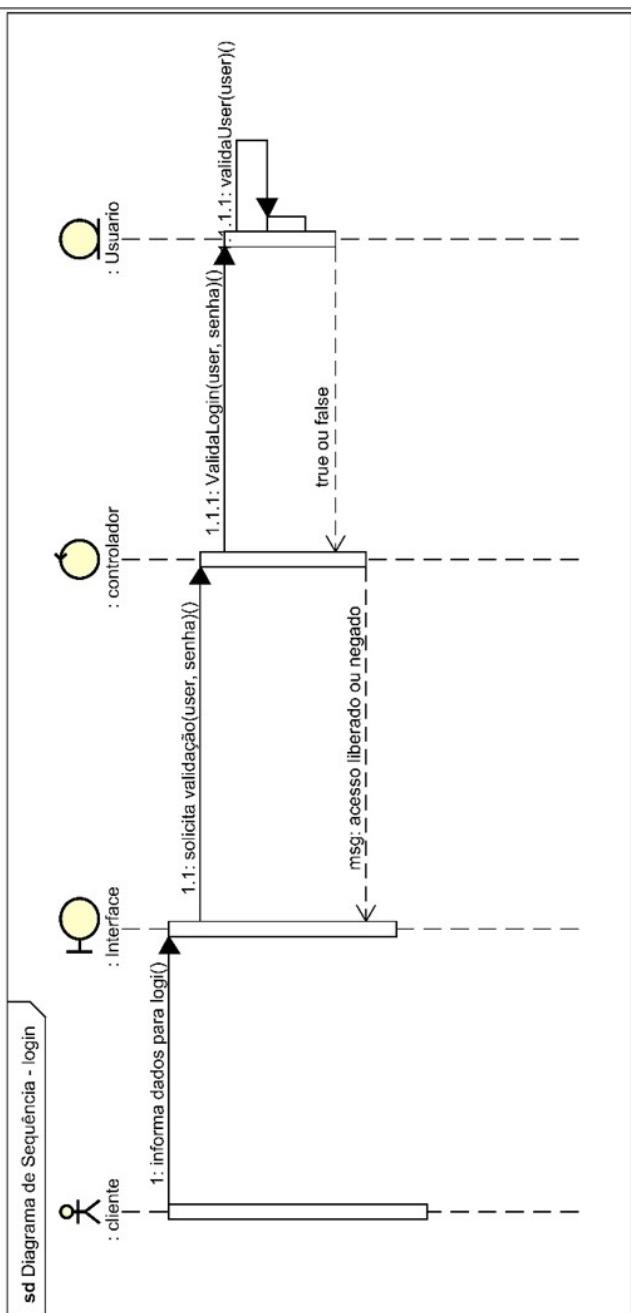
3. Diagrama de sequência

Entre os diagramas de interação, esse é o mais utilizado. O diagrama de sequência mostra a interação entre os participantes do cenário ao longo da vida, a qual é mostrada verticalmente e na ordem de cima para baixo. Esse diagrama é muito intuitivo e quase não requer muitas explicações. A Figura 4.15 mostra um diagrama de sequência.

Na Figura 4.15, os objetos participantes da interação são organizados na horizontal e a seguir pode ser visto a linha da vida, em que cada linha tem seu foco de controle. No exemplo apresentado, o ator cliente acessa o sistema e fornece os dados para efetuar login na interface, a interface aciona o controlador que, por sua vez, aciona uma instância da classe usuário para validar o acesso. O objeto usuário retorna se a validação foi aceita ao controlador, que por sua vez envia à interface, e o acesso é liberado ao cliente.

O diagrama de sequência apresenta, como o próprio nome diz, a sequência dos passos realizados pelos objetos.

Figura 4.15 | Diagrama de sequência



Fonte: captura de tela da ferramenta Astah elaborada pela autora.

Exemplificando

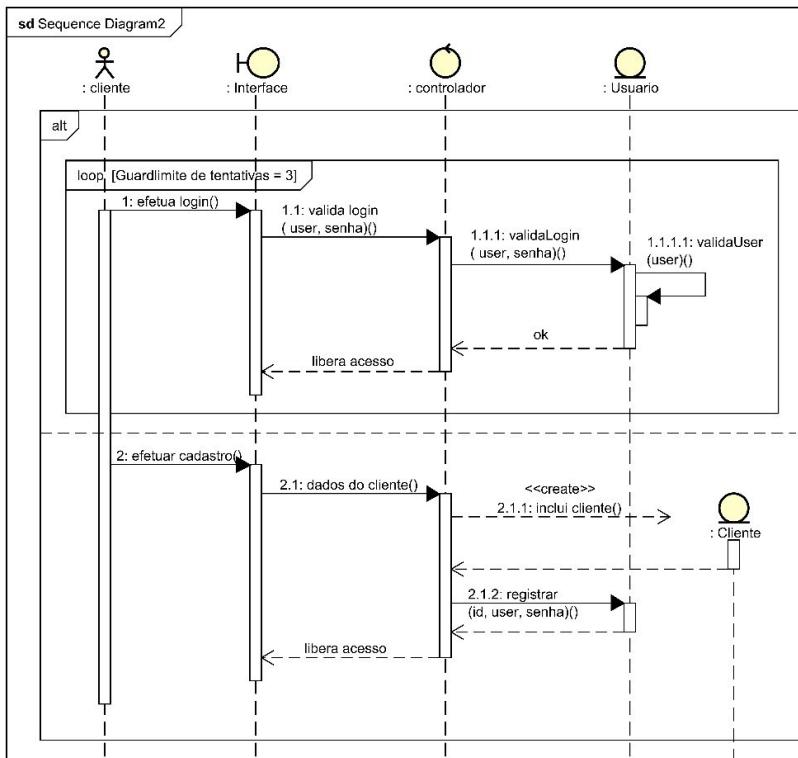
No diagrama de sequência podemos inserir elementos para indicar repetição e operação opcional.

Vamos supor que o cliente possa efetuar o login apenas três vezes.

Nesse caso, basta envolvermos os elementos da Figura 4.15 por um retângulo com a indicação do termo **loop** e uma mensagem indicativa desta condição, como pode ser visto na Figura 4.16.

Outro evento que podemos tratar é se o cliente desejar fazer login mas ainda não está cadastrado, possa efetuar esse cadastro e se logar no sistema. Assim, podemos incluir no diagrama esse evento, que é opcional. Para expressar isso no diagrama de sequência basta, nesse caso, envolver todos os processos por outro retângulo, com a indicação **alt**, a qual indica operação alternativa, e a linha tracejada horizontalmente separa as duas opções. Veja na Figura 4.16.

Figura 4.16 | Diagrama de sequência com operadores de interação



Fonte: captura de tela da ferramenta Astah elaborada pela autora.

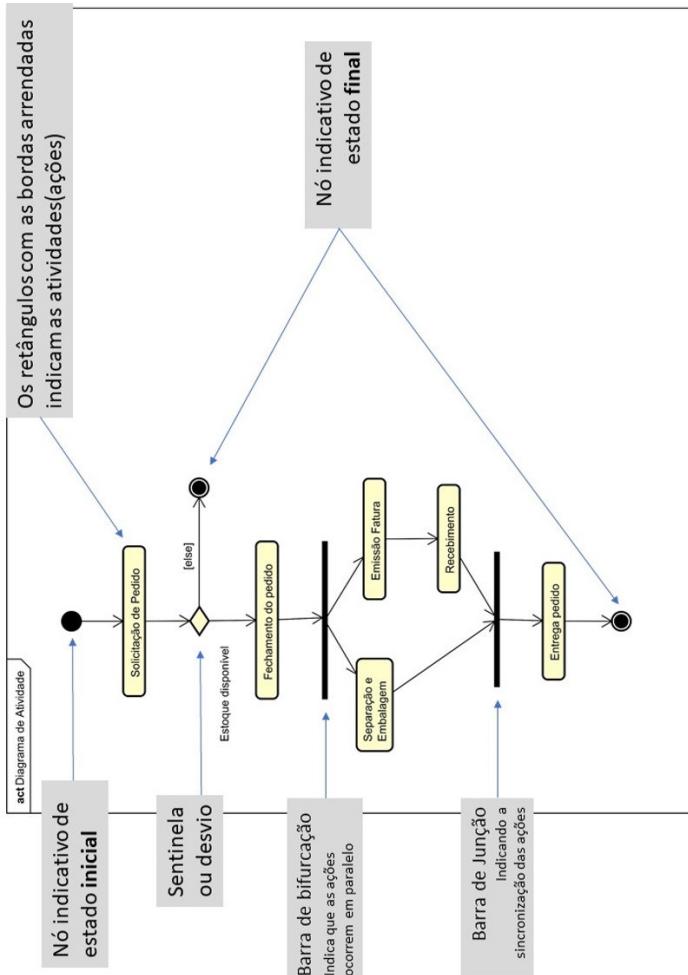
4. Diagrama de atividade

O diagrama de atividade tem por objetivo descrever os passos que devem ser seguidos para a execução de uma determinada atividade.

Esse diagrama assemelha-se muito com as funções de um fluxograma, exceto pelo fato que o diagrama de atividades pode representar atividades em paralelo.

Os elementos básicos do diagrama de atividade são: ações (atividades), sentinela (desvios), estados inicial e final, barra de bifurcação e barra de junção. Esses elementos podem ser vistos na Figura 4.17.

Figura 4.17 | Diagrama de atividades



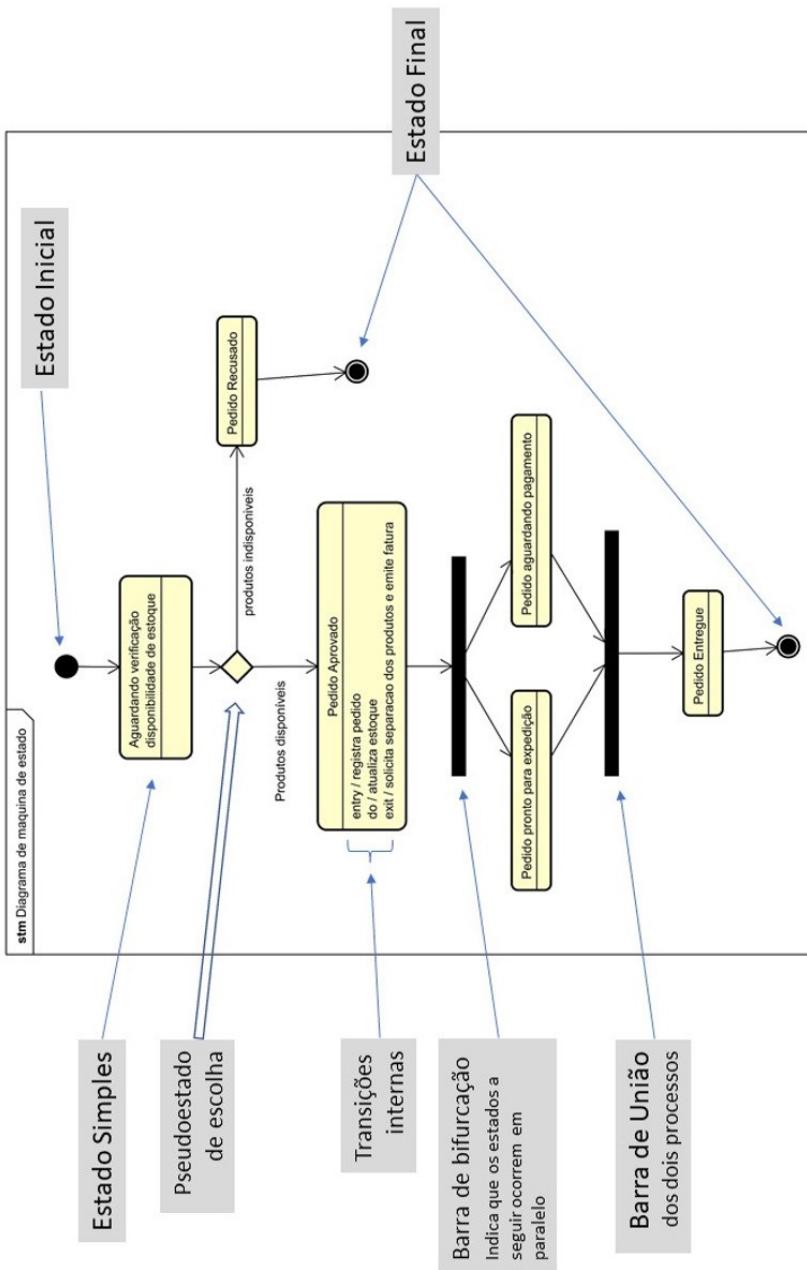
Fonte: captura de tela da ferramenta Astah elaborada pela autora.

5. Diagrama de máquina de estados

O diagrama de máquina de estados é mais um diagrama de comportamento da UML. Esse diagrama visa mostrar a transição de um estado a outro dos objetos do sistema. Na versão 1.x da UML, o diagrama de atividades era um caso específico desse diagrama. Assim, é possível observar muita semelhança entre eles na Figura 4.18, na qual se vê que os símbolos de estado inicial, estado final, barra de bifurcação e pseudoestado (desvio) são iguais aos utilizados no diagrama de atividade. Uma diferença está no símbolo de estado que apresenta transições internas, as quais existem apenas no diagrama de máquina de estados. As transições internas de estado são três, a saber:

- **Entry:** indica ações internas do estado executada pelo objeto ao assumir um estado.
- **Do:** são ações internas do estado executadas durante o período que o objeto se encontra em um estado.
- **Exit:** neste caso, são ações executadas pelo objeto quando ele sai de um estado.

Figura 4.18 | Diagrama de máquina de estados



Fonte: captura de tela da ferramenta Astah elaborada pela autora.

Refita

Você percebeu que existem muitos diagramas?

Por onde começar?

É necessário fazer todos?

Acompanhe esse raciocínio: um fundamento essencial do PU é a **iteração**. Assim, sempre que necessário, retornamos a uma atividade já realizada e a refinamos à medida que avançamos no desenvolvimento da A/POO (Análise e Projeto Orientado a Objeto). Isso é o que a caracteriza o processo iterativo.

Continuando nosso raciocínio, vale lembrar que a UML é uma linguagem que foi criada com um dos objetivos de facilitar esse processo, e com 20% dos diagramas UML é possível elaborar quase 80% dos softwares, o que representa três diagramas. Quais seriam os três diagramas mais utilizados no processo de modelagem? Todos são estáticos?

Para concluir esta seção, vamos conhecer um pouco mais sobre o desenvolvimento ágil. Em fevereiro de 2001, 17 profissionais que praticavam métodos ágeis elaboraram uma declaração de valores e princípios essenciais para o desenvolvimento de software, a qual passou a ser conhecida como Manifesto Ágil (BEEDLE *et al.*, 2001).

Alguns métodos ágeis são focados em responder rapidamente à entrega do sistema; outros são voltados ao reuso e principalmente utilização de componentes, e há métodos ágeis direcionados especificamente para o desenvolvimento de sistemas críticos.

Atualmente o eXtreme Programming e o Scrum estão entre os métodos ágeis mais utilizados no mercado. Ambos seguem o mesmo conjunto de princípios apresentados no Quadro 4.1 por Sommerville (2007).

Quadro 4.1 | Princípios dos métodos ágeis

PRINCÍPIOS	DESCRIÇÃO
Envolvimento do cliente	Clientes devem ser profundamente envolvidos no processo de desenvolvimento. Seu papel é fornecer e priorizar novos requisitos do sistema e avaliar as iterações do sistema.
Entrega incremental	O software é desenvolvido em incrementos e o cliente especifica os requisitos a serem incluídos em cada incremento.
Pessoas, não processo	As habilidades da equipe de desenvolvimento devem ser reconhecidas e exploradas. Os membros da equipe devem desenvolver suas próprias maneiras de trabalhar sem processos prescritivos.
Aceite às mudanças	Tenha em mente que os requisitos do sistema vão mudar, por isso projete o sistema para acomodar essas mudanças.

Manutenção da simplicidade	Concentre-se na simplicidade do software que está sendo desenvolvido e do processo de desenvolvimento. Sempre que possível, trabalhe ativamente para eliminar a complexidade do sistema.
-----------------------------------	--

Fonte: Sommerville (2007, p. 263).

O método Extreme Programming (XP) exige uma abordagem “extrema” para o processo iterativo. O envolvimento do cliente é parte do processo, é adaptável a mudanças, busca a simplicidade do software e seu foco é no valor do negócio. Uma característica interessante deste método, por ser colaborativo (equipe), é que o desenvolvimento do código é feito aos pares. Isso permite que um programador mais experiente atue junto com outro não tão experiente e vá instruindo-o, ou, em casos mais críticos, quando dois pensam juntos, a solução vem mais rápido e fácil. Assim, o resultado esperado de agilizar as entregas é alcançado.

O outro método ágil bastante usado é o Scrum, e como já falamos, segue os mesmos princípios expostos no Quadro 4.1. O método Scrum dá alguns nomes específicos para algumas disciplinas, por exemplo: *sprint*, que funciona como os marcos das iterações do PU, só que o tempo de cada *sprint* é relativamente curto e diariamente a equipe avalia os resultados. Ao final de um *sprint* deve ser entregue um produto ao cliente (minissistema).

Nesta seção você pôde conhecer alguns diagramas UML e a importância de ter um bom conhecimento em pelo menos três diagramas. Também percebeu que os métodos ágeis foram uma evolução na engenharia de software, aprimorando e adaptando as necessidades do PU. Conheceu um pouco sobre a história do surgimento dos métodos ágeis e pôde ver, nos exemplos dos diagramas, alguns casos das situações-problema que terá que desenvolver e refinar.

Pronto para aplicar os conhecimentos adquiridos nesta disciplina no desenvolvimento do seu próximo projeto, futuro desenvolvedor?

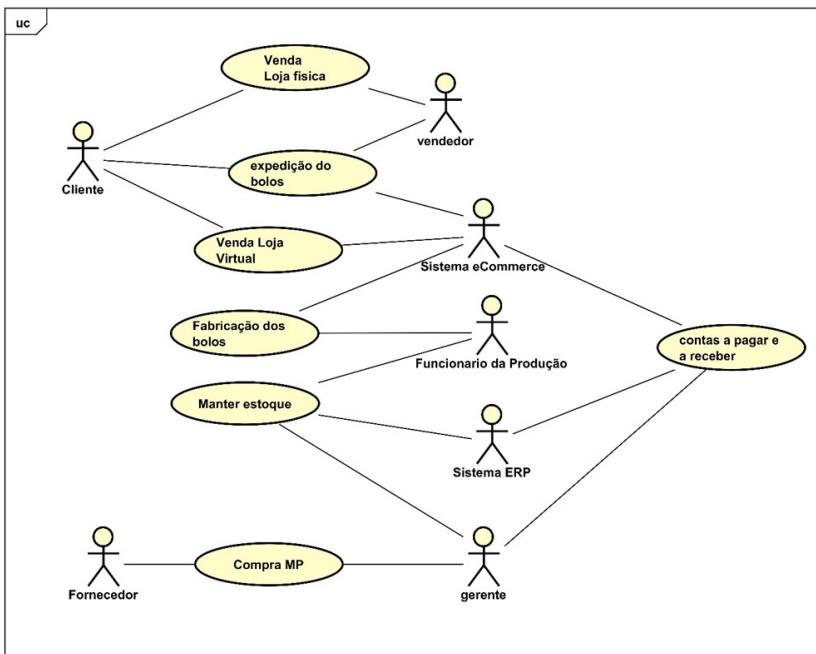
Sem medo de errar

Caro aluno, chegamos ao momento da modelagem do projeto da fábrica de bolos. Seguindo as disciplinas do processo unificado (PU), concluiremos a primeira iteração antes de começar a fase de implementação do código.

Com os conhecimentos de UML adquiridos nesta seção, será possível modelar o sistema utilizando alguns diagramas para responder: “o que o sistema deve fazer”, “como deverá fazer” e “quem irá fazer”.

Vamos começar analisando o diagrama de caso de uso geral, elaborado a partir das primeiras entrevistas com o cliente, o qual está apresentado na Figura 4.8, além de ajustá-lo nesta etapa e de acordo com os padrões UML.

Figura 4.8 | Diagrama de caso de uso geral da fábrica de bolo



Fonte: elaborada pela autora.

Para “o que o sistema irá fazer”, vamos desenvolver e refinar o diagrama de caso de uso apresentado na Figura 4.8 e criar outros se necessários. Já para o “como” será feito, temos algumas opções:

- O diagrama de máquina de estados pode ser usado para transição entre os objetos.
- O diagrama de atividades, que em alguns casos permite um detalhamento do passo a passo a ser seguido em uma determinada atividade.

Escolha um deles, se achar que algum caso de uso deve ser detalhado.

A solução que desenvolveremos utiliza apenas os diagramas de caso de uso, diagrama de sequência e o diagrama de classe que dá suporte tanto ao diagrama de caso de uso como o diagrama de sequência, pois atendem bem às necessidades desse projeto.

Para essa primeira iteração, após uma reunião com o cliente ficou definido que a primeira fase a ser implantada deve ser o caso das vendas, e ficou ajustado que todo o processo poderá ser feito pelo sistema da loja virtual.

Quando a venda for no balcão, o próprio funcionário registrará a venda na loja virtual, com uma indicação que está sendo feito pelo vendedor. Essas vendas deverão ter prioridade na expedição.

	Com base nas decisões da reunião com o cliente, foram elaborados o diagrama de caso de uso específico para as vendas de bolo, o diagrama de classe e o diagrama de sequência. Para consultá-los, acesse o QR Code ou o link: http://cm-cls-content.s3.amazonaws.com/ebook/embed/qr-code/2020-1/analise_modelagem_sistemas/u4/s3/lcls201_u4s3_mod_sis_QRCode2.docx
--	--

Agora que você já desenvolveu uma fase completa do PU, incluindo as disciplinas de modelagem de negócio, requisitos, análise e projeto, aplicando assim todos os conhecimentos adquiridos nesta unidade, é o momento de pensar em se capacitar para a próxima fase: a implementação. Assim, é o momento de pensar em conhecer um pouco de linguagem de programação.

Mas, antes disso, por que não praticar um pouco mais dos diagramas de UML vistos nesta seção? Fica aqui a sugestão: o que acha de rever os diagramas de caso de uso, classe e sequência sobre fazer o pedido mostrado nesta seção?

Faça valer a pena

1. A Linguagem de Modelagem Unificada (UML) tem 14 tipos de diagramas, subdivididos em estruturais e comportamentais.

Analise as assertivas a seguir sobre o diagrama comportamental de caso de uso:

- I. Descrevem estados de cada evento a ser executado pelo sistema.
- II. Descrevem a sequência de passos iniciadas por um ator.
- III. As associações de funcionalidades incondicionais são as <<extend>>.
- IV. As associações de funcionalidades opcionais são as <<include>>.

É correto o que se afirma apenas em:

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

2. A versão 1.0 da UML foi disponibilizada em 1997 e tornou-se um padrão para a modelagem de sistemas orientados a objetos. Nestas duas décadas de sua existência, novas versões surgiram e com elas novos diagramas foram incorporados para facilitar a modelagem de diversos tipos situações. Alguns diagramas são parecidos, mas apresentam finalidades diferentes. Há um diagrama que é utilizado para descrever algoritmos, processos de negócios e fluxo de trabalho e que desempenha um papel semelhante aos fluxogramas, porém diferencia-se deste por apresentar uma notação específica para comportamentos paralelos.

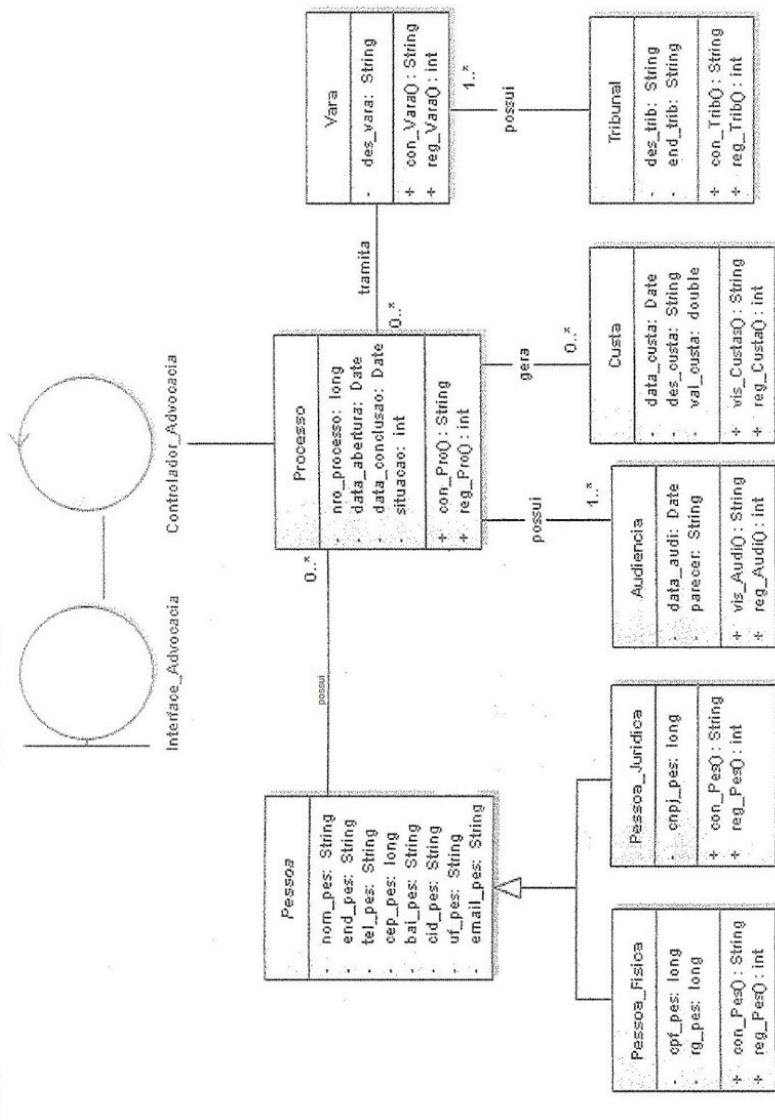
O diagrama referido no texto é:

- a. Máquina de estados.
- b. Sequência.
- c. Colaboração.
- d. Atividades.
- e. Classes.

3. O diagrama de classe está no ranking dos diagramas UML como o mais utilizado. Há muitos elementos nesse diagrama além das classes, tais como estereótipos, vários tipos de relacionamentos e multiplicidade.

O diagrama de classe a seguir refere-se ao modelo de domínio de um sistema de advocacia, apresentado por Guedes (2011).

Figura | Diagrama de classes para sistema de controle de advocacia



classe Sistema de Controle de Advocacia - Modelo de Domínio

Fonte: Guedes (2011, p. 152).

Com relação aos relacionamentos do diagrama de classe apresentado, é correto o que se afirma em:

- a. Todos os relacionamentos são do tipo associação.
- b. Os relacionamentos são de generalização e de associação.
- c. Os relacionamentos entre a classe Processo e as classes Audiência e Cesta estão errados, pois deveriam ser de composição.
- d. Um processo tramita em várias Vara ou em nenhuma.
- e. O relacionamento entre a classe Processo e a classe Vara deveria ser de dependência.

Referências

BEEDLE, M. et al. Manifesto Ágil. [S.l.], 2001. Disponível em: <http://agilemanifesto.org/>. Acesso em: 17 dez. 2019.

Boston: Addison-Wesley, 1999.

CMMI INSTITUTE. Página inicial. 2020. Disponível em: <https://cmmiinstitute.com/>. Acesso em: 12 dez. 2019.

D'ÁVILA, M. Sucessos e falhas em projeto de TI. **Blog do Márcio**, 2010. Disponível em: <http://blog.mhavila.com.br/2010/06/17/sucessos-e-falhas-em-projetos-de-ti/>. Acesso em: 12 dez. 2019.

DEITEL, H. M. Java: como programar. Tradução: Edson Furmarkiewicz. 8. ed. São Paulo: Pearson, 2010.

DIJKSTRA, E. W. Go To Statement Considered Harmful. (Reprinted from) Communications of the ACM, v. 11, n. 3, mar. 1968, p. 147-148. Association for Computing Machinery Inc., 1968. Disponível em: <https://web.archive.org/web/20070703050443/http://www.acm.org/classics/oct95/>. Acesso em: 30 nov. 2019.

FÉLIX, R. Programação orientada a objetos. São Paulo: Pearson Education do Brasil, 2016.

FOWLER, M. **UML Essencial**: um breve guia para a linguagem de modelagem de objetos. Tradução: João Tortello. 3. ed. Porto Alegre, Bookman, 2005.

GOLDBERG, A. Smalltalk-80: the language and its implementation. Xerox Corporation, 1983.

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

IBM. **Rational Unified Process: Best practices for software development teams**. From the developer Works archives. IBM Staff, jul. 2005. Disponível em: https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf. Acesso em: 7 dez. 2019.

JACOBSON, I.; BOOCHE, G.; RUMBAUGH, J. **El Proceso Unificado de Desarrollo de Software**. 1. ed. rev. Madrid: Pearson Educación S.A., 2000.

JACOBSON, I.; BOOCHE, G.; RUMBAUGH, K. **The unified software development process**.

KAY, A. The Early History of Smalltalk. Gagne.homedns.org. ACM 1993 (HOPL-II/4/93/MA), US. Disponível em: <http://gagne.homedns.org/~tgagne/contrib/EarlyHistoryST.html>. Acesso em: 15 dez. 2019.

KOTLER, P.; KELLER, K.L. **Administração de Marketing**. Tradução: Sonia M. Yamamoto. 15. ed. São Paulo: Pearson Education do Brasil, 2018.

KRUCHTEN, P. B. The 4+1 View Model of architecture. **IEEE Software**, v. 12, n. 6, p. 42-50, nov. 1995.

LARMAN, C. **Utilizando UML e padrões**. 3. ed. São Paulo: Bookman, 2007.

OLIVEIRA, L. Faça 10 tipos de bolos caseiros com apenas uma massa! Parte 1. 1 vídeo (25 min. 41 seg.). Canal Chef Léo Oliveira. Disponível em: <https://www.youtube.com/watch?v=COxdVr-TvbM0>. Acesso em: 2 dez. 2019.

OLIVEIRA, L. Faça 10 tipos de bolos caseiros com apenas uma massa! Parte 2. 1 vídeo (31 min. 27 seg.). Canal Chef Léo Oliveira. Disponível em: https://www.youtube.com/watch?v=jh2Ogn8_zc. Acesso em: 2 dez. 2019.

OMG® – OBJECT MANAGEMENT GROUP. **OMG® Unified Modeling Language®**. Specification version 2.5.1, dec. 2017. Disponível em: <https://www.omg.org/spec/UML/2.5.1/> PDF. Acesso em: 6 dez. 2019.

PRESSMAN, R. S. **Engenharia de software [recurso eletrônico]**: uma abordagem profissional. Tradução: Ariovaldo Griesi. Rev. técnica: Reginaldo Akakaki, Julio Akakai, Renato Manzan de Andrade. 7. ed. Dados eletrônicos. Porto Alegre: AMGH, 2011.

RAMOS, D. C. A. Conceitos de Orientação a Objetos: Curso. Programa de TIC – Tecnologia Informação e Comunicação do CCUEC - Unicamp, Campinas-SP. 10 mar. 2017. Disponível em: <http://ftp.unicamp.br/pub/apoio/treinamentos/linguagens/POO.pdf>. Acesso em: 2 dez. 2019.

REZENDE, D. A. **Engenharia de Software e sistemas de informações**. 2. ed. Rio de Janeiro: Brasport, 2002.

SEBESTA, R. W. Conceitos de linguagens de programação. 5. ed. São Paulo: Bookman, 2006.

SINTES, T. Aprenda Programação Orientada a Objetos em 21 dias. Tradução: João Eduardo N. Tortello. São Paulo: Pearson Education do Brasil, 2002. Disponível em: <https://plataforma.bvirtual.com.br/Leitor/Publicacao/8/pdf>. Acesso em: 3 dez. 2019.

SOFTEX. **MpsBr**. Melhoria do Processo de Software Brasileiro. Disponível em: <https://softex.br/mpsbr/>. Acesso em: 12 dez. 2019.

SOMMERVILLE, I. **Engenharia de software**. 8. ed. São Paulo: Pearson - Addison Wesley, 2007. Disponível em: <https://plataforma.bvirtual.com.br/Leitor/Publicacao/276/pdf>. Acesso em: 3 dez. 2019.

STANDISH GROUP. Página inicial. 2020. Disponível em: https://www.standishgroup.com/sample_research. Acesso em: 12 dez. 2019.

TUCKER, A. B.; NOONAN, R. E. Linguagens de programação: princípios e paradigmas. 2. ed. Porto Alegre: AMGH, 2010.

ISBN 978-05-522-1683-4



9 780552 216834 >