

ANÁLISE E MODELAGEM DE SISTEMAS

Unidade 1

Introdução à Análise e Modelagem de Sistemas

Aula 1

Fundamentos da Análise e Modelagem de Sistemas

Este conteúdo é um vídeo!



Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Olá, estudante! Nesta videoaula, você conhecerá a jornada da Evolução de Software, mergulhando nos fundamentos do papel crucial desempenhado pelo analista de sistemas. Abordaremos também as nuances do projeto de software, destacando sua importância na criação de soluções eficientes. Este conhecimento é essencial para todos os profissionais da área, capacitando-os a compreender e adaptar-se às constantes transformações no cenário tecnológico. Prepare-se para aprimorar suas habilidades e impulsionar sua prática profissional.

[Clique aqui](#) para acessar os slides da sua videoaula.

Bons estudos!

Ponto de Partida

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

Os sistemas são conjuntos de componentes inter-relacionados, trabalhando juntos para coletar, processar, armazenar e disseminar informações. Eles são fundamentais para entender como as informações são gerenciadas e utilizadas nas organizações. As características essenciais dos

ANÁLISE E MODELAGEM DE SISTEMAS

sistemas incluem a interconexão de partes, a existência de um objetivo ou propósito, a capacidade de processamento de dados e informações, e a interação com o ambiente. Compreender esses conceitos básicos é crucial para a análise, o design e a implementação eficazes de sistemas de informação, pois oferecem a base para a identificação de problemas e a busca de soluções tecnológicas apropriadas.

A análise de sistemas envolve a investigação detalhada de sistemas existentes ou propostos para determinar as necessidades e objetivos do usuário e desenvolver soluções estratégicas. Esses princípios incluem entender as necessidades dos usuários, identificar e resolver problemas, desenhar soluções eficazes e eficientes, e garantir a adaptabilidade e escalabilidade do sistema. O conhecimento desses princípios é vital para desenvolver sistemas que não apenas atendam aos requisitos atuais, mas também sejam flexíveis o suficiente para se adaptar às necessidades futuras.

O analista de sistemas é um profissional crucial na ponte entre a tecnologia e as necessidades de negócios de uma organização. Eles são responsáveis por compreender os requisitos do negócio e traduzi-los em especificações técnicas para o desenvolvimento de sistemas de informação. Isso inclui a realização de análises de requisitos, a proposição de soluções, o acompanhamento do desenvolvimento do software e a garantia de que o produto final atenda às expectativas do cliente. O papel de um analista de sistemas é fundamental para garantir que os recursos de TI sejam usados de maneira eficiente e eficaz, contribuindo para o sucesso operacional e estratégico da organização.

Pensando nesses conceitos, suponha que você foi convidado a trabalhar como analista de sistemas em um 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?

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

Vamos Começar!

ANÁLISE E MODELAGEM DE SISTEMAS

Conceitos básicos de sistemas e suas características

Na era moderna, o uso de softwares se tornou extremamente difundido em vários aspectos da vida. Hoje em dia, é normal que as empresas exijam que seus empregados tenham competência em softwares específicos, ou providenciem treinamento para seu uso, visto que softwares são frequentemente empregados em atividades cotidianas.

Sommerville (2018) define softwares como "programas de computador acompanhados de documentação associada", e esclarece que eles podem ser criados para clientes específicos ou para um público mais amplo. Por outro lado, Pressman (2021) descreve softwares de computador como produtos desenvolvidos e mantidos a longo prazo por profissionais de Tecnologia da Informação (TI). Ele também destaca que softwares englobam qualquer forma de mídia eletrônica e são constituídos por três elementos principais:

- **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.

Deve-se destacar que o desenvolvimento do software não ocorreu instantaneamente; ele foi fruto de uma evolução contínua, cujo conhecimento é fundamental para todos os profissionais de Tecnologia da Informação. No Quadro 1, é apresentada uma linha do tempo que ilustra a história e a evolução do software, incluindo os eventos mais significativos que marcaram essa trajetória.

Quando?	O que?	Descrição
Década de 1940	Programa	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	Sistemas operacionais	Responsáveis pelo controle do hardware (realizavam a interface entre o homem e a máquina).
	Linguagens de programação	Surgem as linguagens de programação: COBOL, LISP, ALGOL, BASIC etc.

ANÁLISE E MODELAGEM DE SISTEMAS

Décadas de 1960 e 1970	Crise do software	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).
	Paradigmas de programação	Com novas linguagens sendo criadas, nessa época o conceito de orientação a objetos é criado.
	Sistemas operacionais mais eficientes	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	Computador desktop	A década foi marcada pelo surgimento da Microsoft e pela evolução dos computadores pessoais - desktop.
	Unix	O Unix (sistemas operacionais em rede) avançou pelo mundo.
	Evolução da internet	A internet começa a despontar como meio comunicação.
Década de 1990	Internet	A internet é amplamente utilizada.
	Linux	Surge o núcleo do futuro Linux.
	JAVA	Nasce a linguagem JAVA, que revoluciona o paradigma da orientação a objetos.
Década de 2000	Sistemas operacionais gráficos	Geração do Windows da Microsoft (Windows XP,

ANÁLISE E MODELAGEM DE SISTEMAS

Década de 2010 e 2020		Windows Vista, Windows 70) e versões gráficas do LINUX.
	Internet	Softwares que utilizam a web como plataforma de funcionamento.
	Computação em nuvem	Utilizada em larga escala por diversos seguimentos de empresas.
	Aplicativos para dispositivos móveis	Com a evolução dos dispositivos móveis os softwares para aplicativos tornaram-se essenciais.
	Inteligência Artificial	Utilização de algoritmos para a Deep Learning (aprendizagem profunda) e Machine Learning (aprendizado de máquina).

Quadro 1 | Cronologia histórica da evolução do software. Fonte: adaptado de Fonseca Filho (2007).

Conforme apresentado no Quadro 1, é notável que, com a evolução e maior acessibilidade do hardware e software, cresce a urgência de aprimorar os softwares. Paralelamente, as exigências dos consumidores evoluem com a disponibilidade de novas tecnologias. Vejamos alguns exemplos:

1. Nos anos 90, uma empresa geralmente precisava de um software e, possivelmente, de um site na internet para ganhar visibilidade. O site funcionava principalmente como um cartão de visitas virtual.
2. Na década de 2000, essa necessidade evoluiu para um software com capacidade de conexão à internet, permitindo oferecer mais recursos aos clientes.
3. Já em 2010, tornou-se essencial que a empresa dispusesse de um software, um site e um aplicativo móvel, facilitando o acesso a seus serviços em diferentes dispositivos e o armazenamento de dados na nuvem.
4. A partir de 2020, muitas empresas começaram a explorar o uso da Inteligência Artificial para otimizar seus processos de gestão.

Esses desenvolvimentos demonstram claramente que, à medida que novas tecnologias ganham popularidade, aumenta a demanda por software sofisticado para atender às necessidades em constante mudança da sociedade.

Alterações são uma constante no ciclo de vida de um software, ocorrendo durante seu desenvolvimento, na fase de entrega, e mesmo após ser entregue. Ajustes e correções frequentemente se fazem necessários, e muitas vezes surgem pedidos para adicionar novas

ANÁLISE E MODELAGEM DE SISTEMAS

funcionalidades, geralmente a pedido do cliente. O software, portanto, passa por várias manutenções devido a essas novas solicitações. Após múltiplas modificações, que podem introduzir novos desafios, pode surgir a necessidade de desenvolver um novo software.

Ao longo da vida útil de um software, ele passará por diversas mudanças, que podem introduzir novos erros e aumentar a taxa de defeitos. Esse fenômeno é ilustrado na Figura 1.

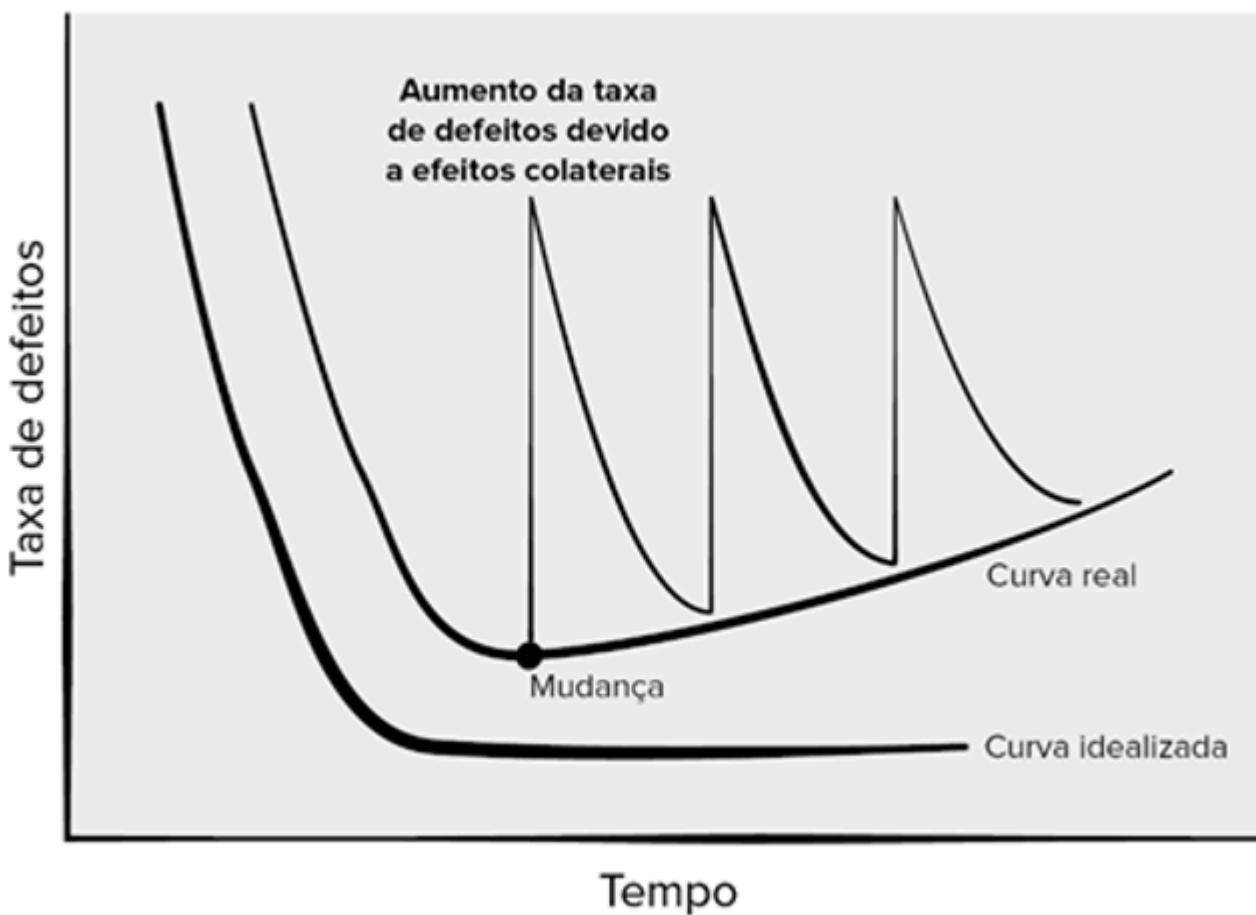


Figura 1 | Curva de defeitos de um software ao longo do tempo. Fonte: Pressman (2021).

A Figura 1 exibe a curva de defeitos em softwares (taxa de defeitos) ao decorrer do tempo. Esse gráfico inclui uma curva teórica mostrando que a taxa de defeitos é inicialmente alta no começo do desenvolvimento do software, mas diminui progressivamente até se estabilizar. Contudo, na realidade, esse padrão ideal raramente se verifica, pois frequentemente são exigidos acréscimos de novas funcionalidades (solicitadas pelo cliente) ou alterações para corrigir problemas detectados. Portanto, a curva real é introduzida, evidenciando que, conforme mudanças são implementadas no software, ele pode se tornar mais suscetível a falhas. Essas alterações podem resultar em novos erros ou defeitos, como indicado pelos picos no gráfico, que simbolizam o aumento nas taxas de defeitos. Muitas vezes, essas melhorias precisam ser feitas de forma rápida, levando a uma negligência no processo de documentação para acelerar as mudanças.

ANÁLISE E MODELAGEM DE SISTEMAS

Este ciclo de modificações pode resultar em um software mais propenso a falhas e ao surgimento de mais erros.

Diferentemente do hardware, que se desgasta fisicamente devido a fatores ambientais como altas temperaturas, poeira e vibrações, o software não sofre deterioração física. No entanto, os softwares podem se deteriorar devido às mudanças que são implementadas neles. É importante notar que a taxa de defeitos no hardware pode ser influenciada não apenas por fatores ambientais, mas também por picos de processamento exigidos pelo software. Isso ocorre quando os recursos de hardware (memória, processador, placa de vídeo) não são suficientes para suportar a execução de um software específico. Um exemplo disso são jogos que demandam placas de vídeo de alta capacidade computacional; eles podem apresentar falhas devido ao superaquecimento do processador e da placa de vídeo.

Siga em Frente...

Princípios da análise de sistemas

Os fundamentos da análise de sistemas residem na importância de conduzir estudos detalhados dos processos, a fim de identificar a solução mais adequada para o desenvolvimento de um sistema. De acordo com Roth, Dennis e Wixom (2014), a análise de sistemas é ancorada em métodos e técnicas de pesquisa e especificação, visando descobrir a solução mais eficaz para um problema ou necessidade computacional específica de uma área de negócios, baseando-se nas funcionalidades identificadas pelo analista de sistemas.

Em uma visão mais generalizada das fases que envolvem os processos da análise de sistema, destacam-se, conforme Pressman (2021):

- **Análise:** esta etapa envolve a realização de estudos focados na especificação do software. O objetivo é avaliar sua viabilidade, considerando o custo-benefício, estabelecer as funcionalidades necessárias do software, e definir o escopo, incluindo a alocação de recursos e a elaboração do orçamento. Os resultados obtidos nesta fase serão fundamentais para as etapas subsequentes.
- **Projeto:** nesta fase, o foco é na definição lógica do software. Envolve a elaboração dos layouts de telas e relatórios, a construção da estrutura de banco de dados e a criação de diagramas gráficos para o desenvolvimento do software.
- **Implementação:** esta etapa consiste na codificação do software, utilizando a linguagem de programação selecionada na fase de análise.
- **Testes:** com o objetivo de identificar erros, são conduzidos testes para verificar as funcionalidades dos componentes codificados.
- **Documentação:** esta fase envolve a elaboração de documentação para todos os processos e diagramas desenvolvidos em todas as etapas. Utiliza-se documentação padronizada (e adaptada por cada empresa de desenvolvimento) como meio de comunicação entre os

ANÁLISE E MODELAGEM DE SISTEMAS

envolvidos no desenvolvimento e como parte do contrato entre as partes interessadas na produção do software.

- **Manutenção:** após a implantação e início de operação do software, esta fase se dedica ao monitoramento, registro e correção de falhas, além de propor melhorias ou adicionar novas funcionalidades.

As fases descritas devem incluir um processo de homologação, que é a aprovação oficial do cliente, para validar os documentos produzidos. É essencial que o cliente compreenda que, uma vez aprovada uma etapa, quaisquer alterações posteriores afetarão os custos e poderão resultar em ajustes no cronograma, incluindo possíveis atrasos.

Sommerville (2018) e Pressmann (2021) destacam alguns princípios da análise de sistemas, são eles:

- É fundamental que a compreensão da informação relativa a um problema específico seja clara e compartilhada por todos os envolvidos no projeto.
- As funcionalidades do software devem ser inicialmente definidas e descritas de maneira ampla, refinando-se progressivamente até alcançar um nível mais detalhado.
- O comportamento do software deve ser ilustrado por meio de suas interações com o ambiente externo, incluindo usuários e outros sistemas.
- Os diagramas que ilustram as funções e comportamentos do software devem ser estruturados em camadas, decompondo um problema complexo em partes menores para facilitar a compreensão
- A análise deve começar com informações essenciais e avançar até os detalhes de implementação, sem se preocupar inicialmente com a codificação específica da solução; os detalhes de implementação definirão como a solução será efetivamente realizada.

O papel do analista de sistema

O analista de sistemas é o profissional encarregado de conduzir atividades de análise de sistemas, incluindo pesquisa, planejamento, coordenação de equipes de desenvolvimento e sugestão de soluções de software adequadas às necessidades de desenvolvimento ou resolução de problemas empresariais. As responsabilidades do analista de sistemas abrangem a concepção, implementação e implantação de software. A tarefa inicial é determinar as funcionalidades que o sistema deve ter, seguida pela compreensão e avaliação das necessidades e expectativas dos usuários do software, para que estas possam ser organizadas, especificadas e documentadas adequadamente (SOMMERVILLE, 2018).

O papel do analista de sistemas envolve desenvolver especificações, funcionalidades e transações, adaptando soluções para atender às demandas dos usuários. Essencialmente, esse profissional deve ter conhecimento sobre diversas áreas de negócio e, na ausência de domínio em algum tema específico, deve ser proativo em buscar informações sobre o campo de aplicação do software.

ANÁLISE E MODELAGEM DE SISTEMAS

Uma habilidade importante para o analista de sistemas é possuir uma perspectiva empresarial aguçada, contribuindo assim para os processos gerenciais na criação do software. As competências valiosas para um analista de sistemas incluem: estar atualizado com as tecnologias, ter organização e método, visão gerencial, excelente habilidade de relacionamento interpessoal, entre outras.

O analista de sistemas atua como um intermediário entre os desenvolvedores de software e os usuários finais. Sua responsabilidade é entender as necessidades e desejos dos usuários e determinar o que é tecnicamente factível para desenvolver. O papel do analista inclui coletar informações dos usuários, interpretar esses dados e comunicá-los aos programadores de uma maneira técnica. Isso garante que o software desenvolvido esteja alinhado com as expectativas tanto do cliente quanto dos usuários finais.

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

- Interagir com clientes e usuários para entender suas necessidades em relação ao software.
- Avaliar custos e determinar a viabilidade do projeto de software.
- Coletar informações essenciais entrevistando os usuários do software.
- Identificar dados e requisitos essenciais para análise e desenvolvimento de soluções.
- Desenvolver a modelagem do software.
- Guiar os desenvolvedores durante todo o processo de criação do software, incluindo aspectos lógicos e de interface gráfica.
- Conduzir e monitorar os testes do software.
- Gerenciar a documentação completa do software.
- Administrar mudanças que ocorram durante o projeto.
- Estabelecer padrões para o desenvolvimento do software.
- Assegurar que a qualidade do software atenda às expectativas do cliente.
- Executar monitoramento e auditorias para identificar possíveis falhas.
- Organizar e ministrar treinamentos para os usuários do software.
- Supervisionar a implantação do software, assegurando a integração e adaptação aos sistemas do cliente.
- Oferecer consultoria técnica para entender as necessidades dos clientes em diferentes áreas de negócio.
- Pesquisar novas tecnologias e fornecedores, e buscar aperfeiçoamento profissional para a equipe de desenvolvimento.

Vamos Exercitar?

Chegou o momento de resolver o desafio que lhe foi proposto no início da aula! Você está trabalhando em um 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,

ANÁLISE E MODELAGEM DE SISTEMAS

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 essa 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 acessem 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.

ANÁLISE E MODELAGEM DE 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 um relatório com as soluções encontradas.

Saiba mais

Quer entender mais sobre a profissão de um analista de sistema, leia o artigo:

SERASA EXPERIAN. [O que faz um analista de sistemas?](#) 2022.

Leia mais sobre a natureza e definição do software no livro *Engenharia de software*, em Capítulo 1, seção 1.1 e 1.2, disponível na Biblioteca Virtual:

PRESSMAN, R. S.; MAXIM, B. R. [Engenharia de software](#). Grupo A, 2021.

Quer compreender mais sobre a evolução de software? Leia o seguinte artigo: AZEVEDO. D. J. P. [Evolução de Software](#). Bate Byte. [s. d.]

Referências

FONSECA FILHO, C. **História da computação**: o caminho do pensamento e da tecnologia. Porto Alegre: EDIPUCRS, 2007.

PRESSMAN, Roger S. **Engenharia de software**: uma abordagem profissional. 9. ed. – Porto Alegre: AMGH, 2021.

SOMMERVILLE, Ian. **Engenharia de software**. 10. ed. São Paulo: Pearson, 2018.

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

Aula 2

Processos de Software

Processos de software

ANÁLISE E MODELAGEM DE SISTEMAS



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Bem-vindo à nossa videoaula! Neste encontro, exploraremos os intrincados Processos de Software e a importante Modelagem das Atividades, oferecendo insights valiosos para sua prática profissional. Compreenderemos a importância desses processos na construção de software eficiente e examinaremos como as tarefas moldam cada etapa. Além disso, mergulharemos no universo da Qualidade do Software, destacando sua relevância para assegurar resultados excepcionais. Prepare-se para aprimorar suas habilidades e elevar sua atuação no desenvolvimento de software!

[Clique aqui](#) para acessar os slides da sua videoaula.

Bons estudos!

Ponto de Partida

Na aula de hoje, vamos explorar os Processos de Software, um tema essencial para profissionais de tecnologia da informação, como desenvolvedores e analistas. Esses processos são mais do que apenas escrever código; eles envolvem uma série de atividades planejadas e executadas que são cruciais para desenvolver um software eficaz e confiável. Vamos cobrir desde os fundamentos da criação de software até a entrega ao cliente, enfatizando a importância de cada etapa do processo. Essa discussão incluirá as práticas comuns na indústria e como elas se aplicam a diferentes tipos de projetos de software, proporcionando uma visão clara da importância dos Processos de software no desenvolvimento tecnológico.

Em seguida, abordaremos as Tarefas e a Modelagem das atividades do processo de software. Essas tarefas são componentes cruciais no processo de desenvolvimento, e entender como elas se interligam é fundamental. A modelagem de atividades ajuda a visualizar o processo de desenvolvimento, identificar desafios e encontrar soluções eficazes. Um aspecto crucial que exploraremos é a Qualidade do software. A qualidade é um objetivo constante, integrado em cada fase do desenvolvimento, e não apenas uma etapa final. Vamos discutir estratégias para assegurar que o software atenda às necessidades do cliente e seja ao mesmo tempo robusto, seguro e eficiente. Esta aula é para aqueles que querem aprimorar seus conhecimentos e habilidades em engenharia de software, com foco na qualidade como um elemento fundamental dos seus projetos.

Pensando nesses conceitos, imagine que você é um Analista de sistemas em uma software house, você realiza diversas tarefas, atendendo diferentes clientes. Alguns clientes têm relatado

ANÁLISE E MODELAGEM DE SISTEMAS

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?

Vamos Começar!

Processo de software

De acordo com Sommerville (2018), um **processo de software** é um conjunto de atividades inter-relacionadas e resultados que conduzem à criação de um software. Pressman (2021) salienta que, na Engenharia de software, um processo não implica um método rígido de desenvolvimento, mas sim oferece uma abordagem flexível que permite à equipe de desenvolvimento selecionar processos alinhados com a filosofia da empresa, visando a qualidade do produto, cumprimento de prazos e redução de custos.

Sommerville (2018) também menciona que a metodologia organizada adotada pela Engenharia de software para a produção de software é conhecida como **processo de software**. Esse processo geralmente engloba várias atividades, incluindo especificação, design, implementação, validação, manutenção e evolução. Destaca-se os seguintes pontos sobre o processo de software:

- Estabelece um padrão para a criação de serviços e produtos.
- Facilita a repetição e reutilização de serviços e produtos, aproveitando componentes já desenvolvidos e padronizados.
- Preserva o conhecimento dentro da empresa, permitindo que novos membros continuem os processos estabelecidos.
- Define e orienta as atividades de um Projeto de Software.
- Especifica todo o processo ou partes do processo de desenvolvimento de software.
- Determina as tarefas a serem realizadas pela equipe e individualmente.
- Minimiza riscos e torna os resultados mais previsíveis.
- Fornece uma base comum de entendimento para a equipe de desenvolvimento, melhorando a comunicação.
- Pode ser usado como um modelo para outros projetos, aumentando a eficiência em novos projetos de software.

Conforme explicado por Engholm Jr. (2010), ao estabelecer processos de software, diversos parâmetros são determinados:

1. O evento que marca o início do processo.

ANÁLISE E MODELAGEM DE SISTEMAS

2. A matriz de responsabilidades atribuídas.
3. As atividades a serem realizadas, juntamente com a ordem em que devem ser executadas.
4. As entradas e saídas associadas a cada atividade.
5. As regras e políticas que devem ser seguidas durante a execução das atividades.
6. A infraestrutura necessária para a realização do processo.
7. Os resultados produzidos ao final de cada processo.

Além disso, a Figura 1 ilustra que, dentro de cada Processo de Software, várias atividades podem ocorrer, seja de forma sequencial ou em paralelo.

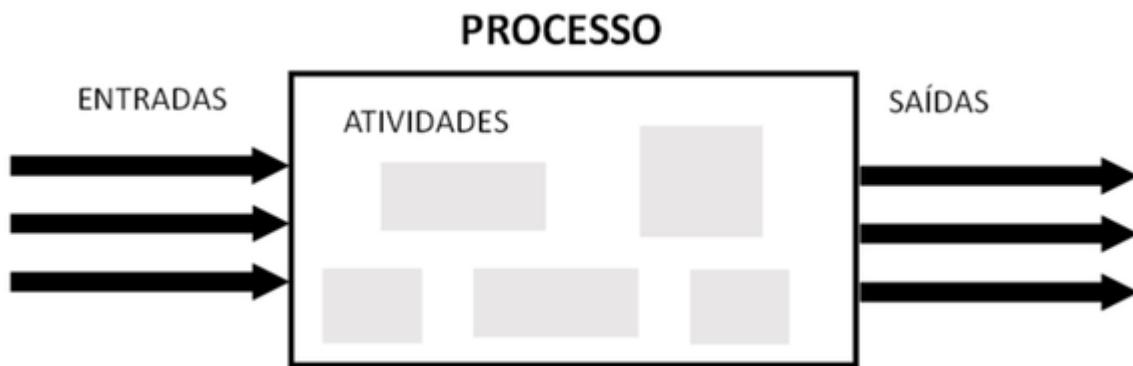


Figura 1 | Representação de um processo de software. Fonte: adaptada de Engholm Jr. (2010, p.43).

Como apresentado na Figura 1, um **processo de software** é caracterizado por várias entradas e saídas. Ele se compõe de uma sequência de atividades padronizadas, que são organizadas em diferentes fases. Nessas fases, ocorrem mudanças nas atividades realizadas. Em cada uma dessas fases, são estabelecidos elementos chave, como as responsabilidades (definindo quem fará o quê), os prazos de entrega, e as estratégias para alcançar os objetivos propostos. A Figura 2 apresenta um diagrama do processo de software, onde se nota que cada atividade metodológica é formada por uma série de ações no campo da engenharia de software. Estas ações são detalhadas por um conjunto de tarefas específicas, as quais determinam as tarefas de trabalho a serem concluídas, os artefatos de software a serem produzidos, os critérios de garantia de qualidade a serem seguidos e os marcos indicativos de progresso.

ANÁLISE E MODELAGEM DE SISTEMAS

PROCESSO DE SOFTWARE

METODOLOGIA DO PROCESSO DE SOFTWARE

ATIVIDADES DE APOIO

Atividade Metodológica Nº 1

Ação da Engenharia de Software nº 1.1

Conjunto
de tarefas

Tarefas de Trabalho
Artefatos
Qualidade
Controle do Projeto

Ação da Engenharia de Software nº 1.2

Conjunto
de tarefas

Tarefas de Trabalho
Artefatos
Qualidade
Controle do Projeto

Figura 2 | Exemplo de metodologia de processo de software. Fonte: adaptada de Pressman (2021).

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

ANÁLISE E MODELAGEM DE SISTEMAS

Software adotado pela empresa de desenvolvimento, todos utilizam uma Estrutura de Processo Genérico de Software, com atividades pré-estabelecidas.

Siga em Frente...

Tarefas e modelagem das atividades do processo de software

As etapas envolvidas em qualquer processo de software são cruciais para a criação de um produto de software finalizado e entregue ao cliente. Dentro de um Processo Genérico de Software, como descrito por Sommerville (2018), embora os processos possam variar entre diferentes projetos, existem quatro atividades fundamentais comuns a todos os projetos de desenvolvimento de software:

- **Especificação de software:** esta etapa envolve definir o escopo do projeto, incluindo suas funcionalidades e limitações.
- **Projeto e implementação de software:** esta fase abrange o desenho e a codificação (programação) do software, garantindo que ele atenda às especificações definidas.
- **Validação de software:** Nesta etapa, é realizada a verificação para assegurar que o software desenvolvido atende às necessidades e requisitos do cliente.
- **Evolução de software:** esta atividade implica em adaptar e atualizar o software para atender às demandas em mudança e às solicitações de melhorias feitas pelo cliente.

Dentro do **Processo genérico de software**, cada atividade é uma parte integrante da Engenharia de software. Segundo Pressman (2021), uma metodologia genérica em Engenharia de software engloba cinco atividades essenciais:

- **Comunicação:** essencial para compreender os objetivos do projeto, a comunicação entre os participantes é crucial para definir os requisitos e funcionalidades do software a ser desenvolvido.
- **Planejamento:** nesta fase, é elaborado um "mapa" ou plano de projeto, detalhando as tarefas técnicas, riscos, recursos, produtos a serem gerados e um cronograma de trabalho para acompanhar o progresso do desenvolvimento do software.
- **Modelagem:** esta atividade envolve a criação de modelos (como diagramas) para uma melhor compreensão das necessidades do software. Esses modelos são fundamentais tanto para a fase de codificação quanto para a validação do projeto pelos stakeholders.
- **Construção:** nesta etapa, ocorre a codificação do software (baseada nos modelos desenvolvidos anteriormente). Além disso, são realizados testes para validar os códigos de programação criados.
- **Entrega:** o software é entregue, seja parcial ou completamente, ao cliente, que realiza testes e fornece feedback. Durante esta fase, adaptações e correções são realizadas no software por um período previamente acordado entre as partes envolvidas.

ANÁLISE E MODELAGEM DE SISTEMAS

O desenvolvimento de software exige extenso planejamento, e cada projeto de software é único em sua natureza. Por exemplo, mesmo quando duas universidades solicitam softwares para controle acadêmico, é improvável que os dois softwares resultantes sejam idênticos. Como destaca Pressman (2021), diferentes projetos de software requerem diferentes conjuntos de tarefas e abordagens na modelagem das atividades do processo de software. Os analistas de sistemas selecionam essas tarefas com base nas necessidades específicas e nas características do projeto em questão.

Um conjunto de tarefas define as ações necessárias para atingir os objetivos específicos dentro do processo de desenvolvimento de software. Um exemplo clássico seria o desenvolvimento das interfaces de um sistema. O programador trabalha com base em um conjunto de requisitos para as telas do sistema. Após a conclusão da programação, a qualidade do trabalho do programador é avaliada com base nas especificações fornecidas para o desenvolvimento dessas interfaces. No Quadro 1, está apresentado um possível conjunto de tarefas na fase de Planejamento de um Software e os resultados esperados relativo a estas atividades.

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.

Quadro 1 | Conjunto de tarefas (atividades) na fase de Planejamento de um software. Fonte: adaptada de Werlich (2020, p. 31).

Qualidade do software

A presença de um **processo de software** em si não assegura automaticamente a qualidade do produto final, nem garante sua entrega dentro do prazo estipulado ou a conformidade total com as funcionalidades solicitadas pelo cliente. A qualidade do software está intrinsecamente ligada aos padrões de qualidade aplicados ao longo dos Processos de Software. Portanto, é crucial estabelecer procedimentos e padrões rigorosos para assegurar a qualidade em cada etapa do

ANÁLISE E MODELAGEM DE SISTEMAS

processo. Um dos benefícios da modelagem de processos é a oportunidade de revisão e busca por melhorias contínuas antes da conclusão e entrega do software.

É essencial avaliar o processo de software para verificar se ele atende a um conjunto de critérios fundamentais. O processo de software pode ser submetido a uma gama de critérios previamente estabelecidos, que contribuem para assegurar a integração e validação eficazes entre as atividades do processo. Pressman (2021) destaca várias metodologias para a avaliação e melhoria dos processos de software, incluindo SCAMPI, SPICE e ISO 9001:2000 aplicadas especificamente ao software.

A **metodologia SCAMPI** (*Standard CMMI Appraisal Method for Process Improvement*), que se alinha com o CMMI (*Capability Maturity Model Integration*), oferece um modelo de avaliação do processo em cinco etapas: início, diagnóstico, estabelecimento, atualização e aprendizado. Essa abordagem estabelece regras para garantir a objetividade nas avaliações, entre outros aspectos:

- Auxilia na coleta e compilação de evidências através de apresentações, documentação e entrevistas.
- Registra todas as evidências observadas em forma de anotações.
- Transforma as anotações em avaliações de conformidade ou não conformidade, em relação às práticas do CMMI.
- Transforma estas avaliações em observações preliminares.
- Confirma a validade das observações preliminares e as converte em observações finais.

A **abordagem SPICE** (ISO/IEC 15504) é um padrão internacional que estabelece critérios para a avaliação de processos de software. Seu objetivo é auxiliar as organizações na realização de avaliações objetivas da eficácia dos processos de software. Este método oferece uma estrutura para a avaliação de Processos de Software, que pode ser aplicada em organizações envolvidas na produção de software.

Por outro lado, a **abordagem ISO 9001:2000** para software, segundo Pressman (2021), é um padrão genérico que pode ser aplicado a qualquer organização que busque manter um padrão global de qualidade em seus produtos, sistemas ou serviços. Existem diversos modelos de referência que auxiliam na garantia da qualidade do software, incluindo ISO/IEC 9126, ISO 9000, ISO 9001 e ISO/IEC 12207. A ISO 9001 foca na garantia de qualidade em design, desenvolvimento e assistência técnica de projetos. Esta norma é especificamente aplicável ao desenvolvimento, fornecimento e manutenção de software, conforme detalhado na norma ISO 9000-3. Já a ISO/IEC 9126 detalha as características que definem um software de qualidade.

Os erros que surgem durante o processo de software podem ser gerenciados por meio de uma abordagem metodológica cuidadosa. É essencial que os Analistas de Sistemas se mantenham atualizados sobre as novas metodologias, avaliando-as e adotando-as no Processo de Software quando apropriado. O objetivo é desenvolver um software de alta qualidade, com o mínimo possível de erros e que atenda às expectativas do cliente.

ANÁLISE E MODELAGEM DE SISTEMAS

Vamos Exercitar?

Chegou o momento de resolver o desafio que lhe foi proposto no início da aula. 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:

- **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 melhoramento do software e inclusão de novos recursos.

É comum as atividades de Especificação que envolvem todo o processo de busca de informação sobre o software a ser construído funcionarem 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, 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.

ANÁLISE E MODELAGEM DE SISTEMAS

Saiba mais

Existe um modelo de referência de qualidade de software brasileiro, este é chamado de MPS.BR. Conheça mais sobre este modelo:

SOFTEX. [Modelos de referência](#). [s. d].

Leia mais sobre a modelos de processo e definição de atividade no livro *Engenharia de software*, em Capítulo 2, seção 2.1 e 2.2, disponível na Biblioteca Virtual:

PRESSMAN, R. S.; MAXIM, B. R. [Engenharia de software](#). Grupo A, 2021.

O livro *Engenharia de Software – Projetos e Processos*, Capítulos 1, 2 e 3 aprofundam bastante o tema estudado nesta aula.

FILHO, W. de P. P. [Engenharia de Software - Projetos e Processos](#) - Vol. 2. Grupo GEN, 2019

Referências

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

PRESSMAN, R. S. **Engenharia de software**: uma abordagem profissional. 9. ed. – Porto Alegre: AMGH, 2021.

SOMMERVILLE, I. **Engenharia de software**. 10. ed. São Paulo: Pearson, 2018.

WERLICH, C. **Análise e modelagem de sistemas**. Londrina: Editora e Distribuidora Educacional S.A., 2020.

Aula 3

Modelos de Processos de Software

Modelos de processos de software

Este conteúdo é um vídeo!

ANÁLISE E MODELAGEM DE SISTEMAS



Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Olá estudante! Nesta videoaula, você conhecerá o mundo dos modelos de processo, destacando o Scrum, XP e o Modelo Evolutivo. Compreender essas metodologias é crucial para aprimorar sua prática profissional. O Scrum otimiza a gestão de projetos, o XP foca em práticas ágeis, e o Modelo Evolutivo abraça a adaptabilidade. Juntos, esses conhecimentos capacitam você a moldar abordagens eficientes no desenvolvimento de software. Prepare-se para uma imersão prática e transformadora.

[Clique aqui](#) para acessar os slides da sua videoaula.

Bons estudos!

Ponto de Partida

O uso de smartphones vem revolucionando a maneira das pessoas utilizarem softwares. Existem APPs 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 aula, exploraremos diversos **modelos de processos de software** disponíveis no mercado. Alguns desses modelos são antigos, mas ainda são amplamente utilizados em vários projetos. A escolha do modelo adequado depende do tipo de software a ser desenvolvido e das expectativas do cliente. O objetivo de todos esses modelos é evitar o caos no desenvolvimento e estabelecer um fluxo de trabalho eficaz.

Imagine que você é um analista de sistemas trabalhando em uma software house e acaba de receber um novo cliente. Este cliente deseja desenvolver um software para sua loja de brinquedos online, que permita aos usuários comprar produtos. Além do site, é necessário criar um aplicativo para compras e aluguel de brinquedos, sendo que a opção de aluguel estará disponível apenas no app.

Agora, você enfrenta a decisão crucial: qual modelo de processo de software é o mais adequado para este novo projeto?

ANÁLISE E MODELAGEM DE SISTEMAS

Vamos Começar!

Os **Modelos de Processos de Software** são utilizados para gerenciar as atividades envolvidas no desenvolvimento de software. O objetivo de um modelo de processo de software é fornecer um framework estável, controlado e organizado para essas atividades. Geralmente representado de forma gráfica, um modelo de processo de software detalha os objetos e atividades que fazem parte do processo de desenvolvimento. Pfleeger (2004) aponta que há uma variedade de técnicas e ferramentas disponíveis para a Modelagem de Processos e que não existe um modelo universalmente aceito como o melhor. Cada empresa adota e adapta um modelo de processo de software de acordo com suas necessidades específicas e os requisitos de cada projeto de software.

Pressman (2021) enfatiza que um modelo de processo de software serve como um roteiro para as atividades de Engenharia de software, delineando o fluxo de atividades, ações e tarefas, bem como o grau de interação entre elas, os artefatos a serem produzidos e a organização do trabalho a ser realizado. Há vários modelos de processos de software, cada um com suas próprias características distintas. Alguns dos mais notáveis incluem **modelos de processos prescritivos**, **modelos de processos especializados** e **modelos de desenvolvimento ágil**.

Modelos de processos prescritivos

O **modelo de processo prescritivo**, também conhecido como **modelo de processos tradicionais**, é definido por um conjunto de elementos do processo, que incluem ações de engenharia de software, produtos resultantes do trabalho e mecanismos de garantia de qualidade e controle de mudanças em projetos de desenvolvimento de sistemas de software, conforme descrito por Pressman (2021). Este modelo estabelece as relações entre os elementos do processo com o objetivo de organizar e estruturar o desenvolvimento de um software de maneira ordenada.

Neste modelo, as tarefas são realizadas de forma sequencial e com diretrizes claras. Elas delineiam um conjunto de atividades metodológicas, ações, tarefas, artefatos, medidas de garantia de qualidade e mecanismos de controle de mudanças para cada projeto. Cada **Modelo de processo prescritivo** também define um **fluxo de processo** (ou **fluxo de trabalho**), descrevendo como os elementos do processo estão interconectados.

Dentre os modelos de processos prescritivos mais conhecidos estão o **modelo cascata**, o **modelo Incremental**, os **modelos evolucionários** (que incluem prototipação e espiral) e os **modelos concorrentes**.

O **modelo cascata**, também conhecido como **ciclo de vida clássico** de um sistema ou abordagem **Top-down**, caracteriza-se por sua abordagem sistemática e sequencial para os processos de desenvolvimento de software. Neste modelo, cada fase do projeto começa somente após a conclusão da fase anterior, enfatizando uma progressão linear e estruturada. A Figura 1 ilustraria

ANÁLISE E MODELAGEM DE SISTEMAS

este modelo, mostrando como ele flui de uma etapa para a próxima de maneira ordenada e previsível.

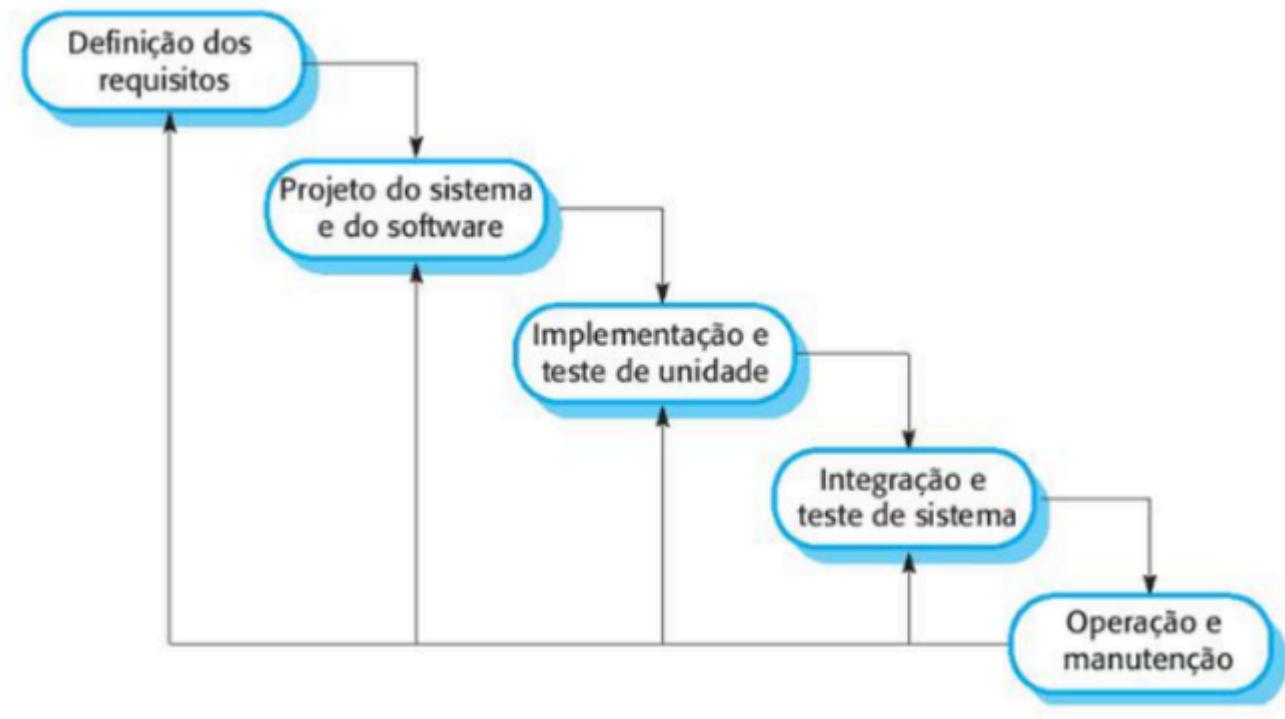


Figura 1 | Modelo cascata. Fonte: Sommerville (2018, p. 33).

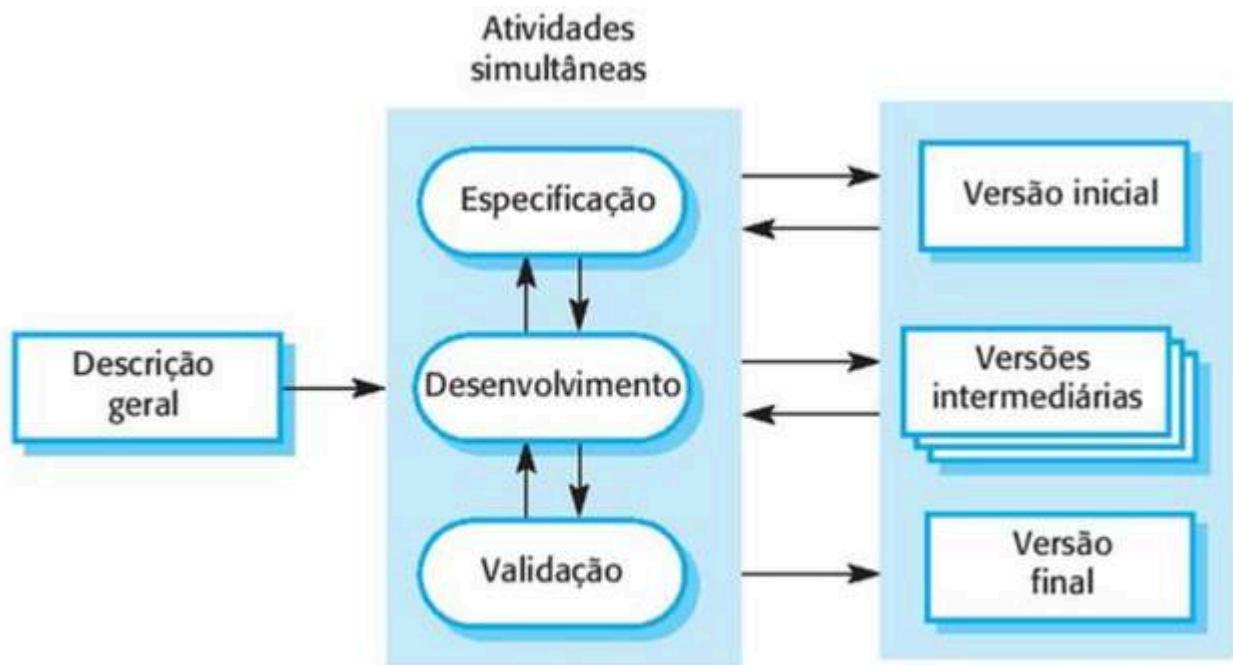
A Figura 1 destaca os estágios do modelo cascata com as cinco atividades fundamentais deste modelo. Vejamos a definição dessas cinco etapas segundo Sommerville (2018):

- **Definição de requisitos:** nesta etapa, as funcionalidades do sistema são definidas através de consultas aos usuários. A especificação do sistema é documentada, envolvendo a participação e a aprovação do cliente para garantir que as necessidades sejam atendidas.
- **Projeto do sistema de software:** esta fase inclui o planejamento da estrutura do sistema, designando recursos para o desenvolvimento do software. Aqui, são definidas a estrutura de dados, a arquitetura do software, interfaces gráficas, entre outros aspectos técnicos.
- **Implementação e teste de unidade:** durante a implementação, o software é codificado na linguagem de programação escolhida. Paralelamente, testes unitários são realizados em cada módulo ou parte do código para identificar e corrigir falhas de programação.
- **Integração e teste de sistemas:** após a codificação, todos os módulos do software são integrados e submetidos a testes conjuntos, preparando o software para entrega ao cliente.
- **Operação e manutenção:** com o software em uso, surgem correções e possíveis melhorias solicitadas pelos usuários. Nesta fase, são implementadas as correções e novos requisitos identificados como necessários pelo cliente, assegurando a funcionalidade contínua e aprimoramento do software.

ANÁLISE E MODELAGEM DE SISTEMAS

O modelo em cascata é o mais tradicional e simples dentre os modelos de processos de software, oferecendo uma estrutura clara para as atividades e servindo de base para outros modelos que surgiram posteriormente. Sua simplicidade e clareza tornam o gerenciamento mais fácil. No entanto, este modelo pode prolongar o tempo de desenvolvimento de um software, especialmente em projetos complexos, devido à sua natureza sequencial, onde atrasos em uma fase afetam as subsequentes. Como resultado, os clientes podem enfrentar longos períodos de espera antes de receberem o produto final, o que pode levar à insatisfação. Além disso, o modelo em cascata normalmente conta com uma única fase de especificação de requisitos. Uma alternativa para mitigar essas limitações é o uso do modelo incremental.

O **modelo incremental** é iterativo e constrói o software em etapas. A partir dos requisitos iniciais, pequenas versões do software são desenvolvidas e entregues ao cliente, expandindo-se progressivamente até a construção completa do sistema ideal. Neste modelo, cada versão representa um incremento, incluindo parte das funcionalidades requisitadas. Diferentemente do modelo em cascata, no modelo incremental o cliente recebe o software em várias entregas parciais ao longo do desenvolvimento. Essas entregas consistem em módulos que adicionam ou aprimoram as funcionalidades do sistema. Cada incremento é lançado como uma nova versão do software, culminando na versão final. A Figura 2 ilustra o modelo incremental, mostrando que as atividades de especificação, desenvolvimento e validação ocorrem de forma intercalada, não separadamente. Segundo Sommerville (2018), é essencial manter um feedback rápido entre as atividades simultâneas. Em cada incremento, o ciclo completo de desenvolvimento de software é realizado, do planejamento aos testes, resultando em um sistema funcional, mesmo que ainda incompleto em termos de requisitos.



ANÁLISE E MODELAGEM DE SISTEMAS

Figura 2 | Modelo de processo prescritivo: modelo incremental. Fonte: Sommerville (2018, p. 35).

O **modelo espiral**, apresentado na Figura 3, uma variante do **modelo evolutivo**, combina a natureza iterativa da prototipação com os elementos sistemáticos e controlados do modelo cascata, conforme descrito por Pressman (2016). Este modelo visa o desenvolvimento acelerado de versões do software, onde cada ciclo iterativo resulta em uma versão mais avançada e completa do produto.

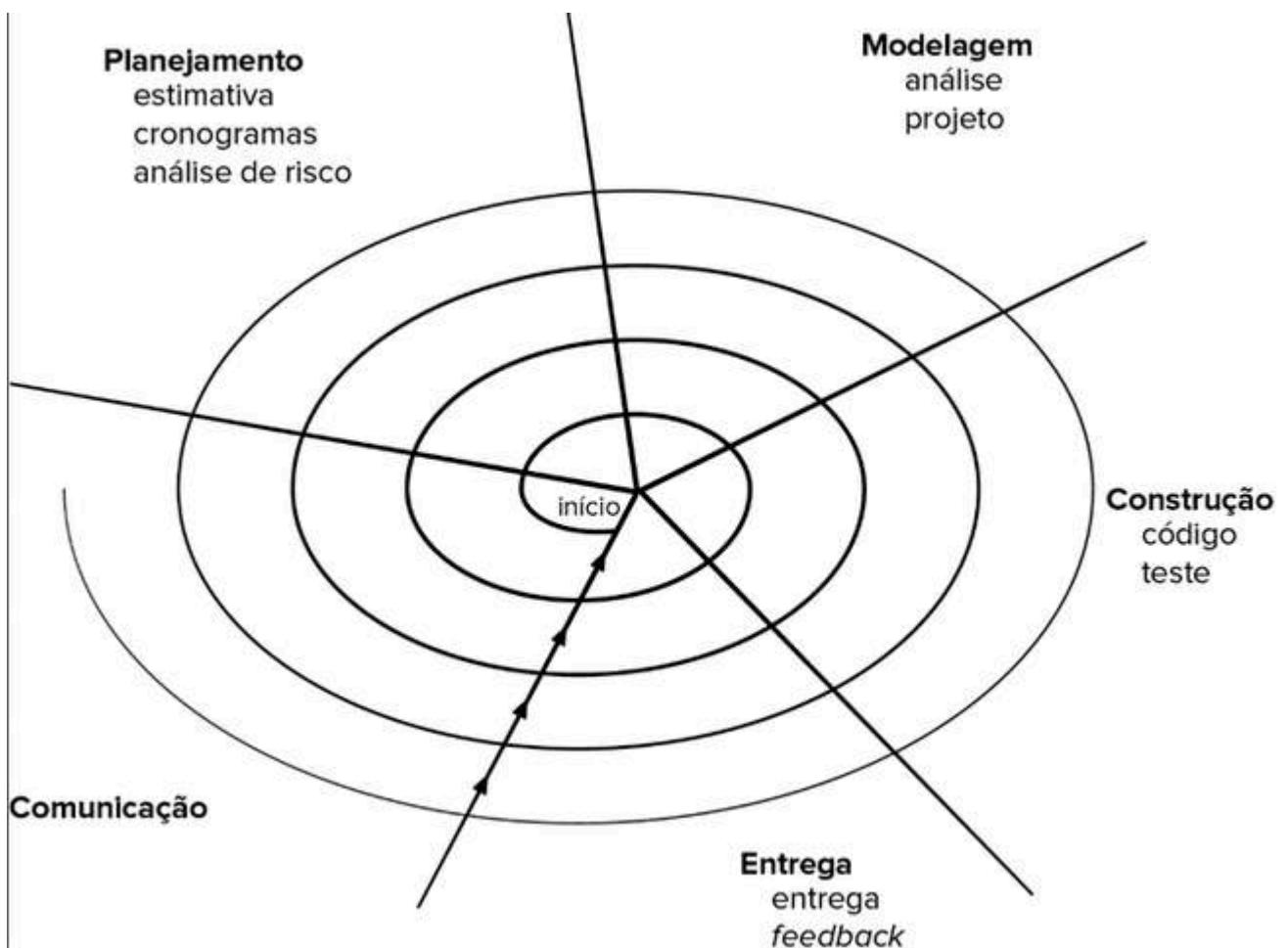


Figura 3 | Modelo de processo prescritivo: modelo evolucionário-espiral. Fonte: Pressman (2021).

Siga em Frente...

Modelo de processo especializado

Os **modelos de processos especializados** combinam elementos de um ou mais Modelos de Processos Prescritivos e são aplicados em situações que demandam uma abordagem de

ANÁLISE E MODELAGEM DE SISTEMAS

Engenharia de Software mais específica e detalhada. Pressman (2021) lista diversos exemplos desses modelos especializados:

- **Modelo baseado em componentes:** este modelo é frequentemente utilizado em projetos de software comercial, onde componentes de software pré-fabricados são empregados. Tais componentes, vendidos individualmente ou em conjunto, são projetados para reutilização em diversos projetos. Cada componente é uma unidade independente do software e pode ser substituído ou modificado conforme necessário.
- **Modelo de métodos formais:** este modelo envolve a criação de especificações matemáticas formais do software. Ele ajuda a identificar e eliminar problemas como ambiguidade, incompletude e inconsistência, fornecendo uma base sólida para a verificação de código e a detecção de erros que poderiam passar despercebidos.
- **Desenvolvimento de software orientado a aspectos:** esta abordagem metodológica define, especifica, projeta e constrói aspectos do software. O código é organizado de acordo com sua importância, com as classes orientadas a objetos sendo um exemplo. Os requisitos são modelados de maneira a abranger várias funcionalidades do sistema.
- **Modelo de processo unificado:** também conhecido como RUP (*Rational Unified Process*), este modelo combina características dos modelos prescritivos tradicionais com alguns princípios da metodologia Ágil. É considerado um modelo iterativo e incremental.
- **Modelos de processos pessoal e de equipe:** esses modelos focam no desenvolvimento de software como um esforço coletivo da equipe. No **modelo de processo de software pessoal**, a ênfase é na medição individual do trabalho produzido e na qualidade resultante. Já o **modelo de processo de software de equipe** visa criar uma equipe auto-organizada e autodirigida, com o objetivo de produzir software de alta qualidade.

Modelo de desenvolvimento ágil

Com a evolução constante das tecnologias, os processos de negócios têm exigido um desenvolvimento de software mais ágil e rápido. Diante disso, surgiu a Metodologia Ágil, uma abordagem que introduz maior flexibilidade e dinamismo no desenvolvimento de software.

A Metodologia Ágil foi desenvolvida para abordar certos desafios da Engenharia de software, oferecendo vantagens significativas. Para Sbrocco (2012), essa metodologia responde à necessidade de um desenvolvimento de software mais rápido, em resposta à demanda dos clientes por resultados rápidos e à natureza evolutiva dos aplicativos, que frequentemente incorporam novas funcionalidades. O Desenvolvimento Ágil enfatiza a flexibilidade e a rapidez na atenção às necessidades dos clientes. Outro ponto relevante deste método são as entregas frequentes, com ênfase na comunicação ativa e contínua entre as partes envolvidas, buscando satisfazer o cliente através de entregas incrementais. Os princípios fundamentais do Desenvolvimento Ágil:

- **Envolvimento ativo do cliente:** é crucial que o cliente participe ativamente no desenvolvimento do software, fornecendo requisitos e avaliando os incrementos desenvolvidos.

ANÁLISE E MODELAGEM DE SISTEMAS

- **Desenvolvimento incremental:** o software é desenvolvido em etapas, permitindo que o cliente forneça novos requisitos que serão incorporados em cada incremento subsequente.
- **Foco nas pessoas, não nos processos:** a equipe tem autonomia no desenvolvimento, maximizando o potencial dos desenvolvedores, sem a rigidez dos processos prescritivos.
- **Flexibilidade para mudanças:** o projeto é estruturado de forma a acomodar mudanças nos requisitos, adaptando-se às novas necessidades que possam surgir.
- **Simplicidade como chave:** a equipe deve evitar complexidades desnecessárias, adotando uma abordagem simples e eficiente em seu trabalho.

O processo de Desenvolvimento Ágil tem como objetivo principal diminuir significativamente a necessidade de documentação extensa. Isso torna o processo de desenvolvimento mais flexível e diminui a burocracia, comum em outros Modelos de Processos de Software. Nesse contexto, destacam-se dois Métodos Ágeis: XP (*Extreme Programming*) e Scrum.

No **Método XP**, a retroalimentação constante, o desenvolvimento incremental e a comunicação eficaz entre os envolvidos são aspectos centrais. Pressman (2021) salienta a necessidade de seguir quatro atividades metodológicas principais: Planejamento, Projeto, Codificação e Testes. O desenvolvimento do software segue um padrão, com trabalho em pares e a participação ativa de um representante do cliente no processo, esclarecendo dúvidas e integrando-se à equipe de desenvolvimento.

Por outro lado, a metodologia **Scrum** aplica um processo de desenvolvimento iterativo e incremental, sendo também aplicável à gestão de projetos. Este método estabelece um conjunto de práticas e regras de gestão visando o sucesso do projeto, como o trabalho em equipe e a comunicação eficaz. Pressman (2021) menciona que o Scrum compreende atividades metodológicas como Requisitos, Análise, Projeto, Evolução e Entrega, cada atividade ocorrem as seguintes tarefas principais:

- **Product Backlog:** esta é uma lista dinâmica de requisitos e funcionalidades do projeto, com prioridades que podem ser ajustadas a qualquer momento. O gerente do produto é responsável por adicionar, remover e atualizar as prioridades nessa lista.
- **Sprints:** estas são unidades de trabalho designadas para atingir objetivos específicos estabelecidos no Product Backlog. Cada Sprint é delimitado por um período fixo, conhecido como Time Box, que define os prazos de entrega.
- **Reunião de planejamento de sprint:** nessa reunião, o Product Owner define as prioridades dos itens do Product Backlog, e a equipe seleciona as tarefas que consegue implementar no próximo Sprint.
- **Daily Scrum:** são reuniões diárias e breves, normalmente com duração de 15 minutos, realizadas no início de cada dia de trabalho. Durante estas reuniões, cada membro da equipe responde a três perguntas chave: (i) O que foi realizado desde a última reunião?; (ii) Quais obstáculos foram encontrados?; (iii) O que está planejado até a próxima reunião?

No início de um projeto, são estabelecidas as ideias e funcionalidades básicas do produto, conhecidas como Histórias. Essas Histórias, juntas, compõem o Product Backlog. No método

ANÁLISE E MODELAGEM DE SISTEMAS

Scrum, as reuniões são frequentemente lideradas pelo Scrum Master, que é responsável por orientar o processo e monitorar o progresso da equipe. O Scrum Master avalia as respostas de cada membro para identificar problemas precocemente, como atrasos ou dificuldades em compreender determinados requisitos. Ao final de cada Sprint, os requisitos definidos são concluídos e sua funcionalidade é avaliada, o que contribui para aperfeiçoar o processo para o próximo Sprint. Cada Sprint finaliza com um avanço no produto ou no Product Backlog.

Além do Scrum, existem outros métodos de Desenvolvimento Ágil. Pressman (2021) menciona métodos como o Desenvolvimento de Sistemas Dinâmicos (DSDM), Modelagem Ágil e o Processo Unificado Ágil. Esses métodos ágeis compartilham uma ênfase na colaboração humana e na auto-organização como componentes fundamentais.

Vamos Exercitar?

Chegou o momento de resolver o desafio que lhe foi proposto no início da aula! 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

ANÁLISE E MODELAGEM DE SISTEMAS

princípios da metodologia Ágil 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 comecem 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 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.

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			
...			

Quadro 1 | Exemplo de quadro Scrum para o projeto. Fonte: adaptada de Werlich (2020, p.50).

Saiba mais

Leia mais Modelos de Processos de software no livro *Engenharia de Software*, Capítulo 2 – Processos de software.

SOMMERVILLE, I. [Engenharia de software](#). 10. ed. São Paulo, SP: Pearson, 2018.

Gostou do tema Metodologias Ágeis? Quer saber mais sobre o assunto? O livro *Metodologias Ágeis* aborda esse assunto e apresenta outras metodologias. Acesse os capítulos de acordo com os temas estudados na aula.

SBROCCO, J. H. T. de C.; MACEDO, P. C. de. [Metodologias Ágeis - Engenharia de Software sob Medida](#). Editora Saraiva, 2012.

ANÁLISE E MODELAGEM DE SISTEMAS

Uma boa ferramenta para organizar as tarefas do método ágil Scrum, é o [Trello](#). Esta ferramenta pode ser empregue para criar quadros que representam Sprints, onde listas podem ser usadas para categorizar as tarefas em '*To Do*', '*In Progress*' e '*Done*'. Cada cartão pode representar uma história de usuário ou uma tarefa, e a equipe pode mover estes cartões pelas listas conforme avançam no Sprint.

Referências

PRESSMAN, R. S. **Engenharia de software: uma abordagem profissional.** 9. ed. – Porto Alegre: AMGH, 2021.

PFLEGER, S. L. **Engenharia de software: teoria e prática.** 2. ed. São Paulo: Prentice Hall, 2004.

SBROCCO, J. H. T. de C. **Metodologias ágeis:** engenharia de software sob medida.1. ed. -- São Paulo: Érica, 2012.

SOMMERVILLE, I. **Engenharia de software.** 10. ed. São Paulo: Pearson, 2018.

WERLICH, C. **Análise e modelagem de sistemas.** Londrina: Editora e Distribuidora Educacional S.A., 2020.

Aula 4

Processo Unificado

Processo unificado



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Olá estudante! Nesta videoaula, você conhecerá o Processo Unificado (PU), destacando suas variantes, como o RUP, e a ferramenta essencial para modelagem, a UML. Compreender esses conceitos é fundamental para aprimorar sua prática profissional, oferecendo uma abordagem robusta no desenvolvimento de software. O PU proporciona estruturação eficiente, o RUP amplia

ANÁLISE E MODELAGEM DE SISTEMAS

a adaptabilidade, e a UML potencializa a comunicação visual. Prepare-se para uma jornada enriquecedora que impactará positivamente sua atuação no universo da engenharia de software! [Clique aqui](#) para acessar os slides da sua videoaula.

Bons estudos!

Ponto de Partida

Nesta aula, focaremos no Processo Unificado, uma abordagem de desenvolvimento de software que se destaca por ser iterativa, incremental e centrada em arquitetura e casos de uso. O **processo unificado** é estruturado em quatro fases principais: Concepção, Elaboração, Construção e Transição. Cada fase tem objetivos específicos e contribui para o avanço progressivo do projeto de software. Ao longo da aula, exploraremos detalhadamente essas fases, enfatizando as atividades-chave e os artefatos gerados em cada uma delas. Além disso, discutiremos a importância dos casos de uso como um mecanismo fundamental para capturar requisitos e guiar o desenvolvimento. A abordagem iterativa e incremental permite avaliações e ajustes frequentes, facilitando o gerenciamento de riscos e melhorando a adaptabilidade do projeto às mudanças de requisitos.

A compreensão do **processo unificado** é crucial para os profissionais de tecnologia, especialmente aqueles envolvidos na gestão de projetos de software. Esta metodologia oferece um framework estruturado que ajuda as equipes a lidar com a complexidade do desenvolvimento de software, ao mesmo tempo em que fornece flexibilidade para se adaptar a mudanças e novos requisitos. O processo unificado apoia uma abordagem de desenvolvimento orientada a resultados, permitindo que as equipes identifiquem e resolvam problemas de maneira eficiente, minimizando riscos e garantindo a entrega de software de alta qualidade.

Imagine que você é um gerente de projeto em uma empresa de desenvolvimento de software. Seu novo projeto é criar um aplicativo de gerenciamento de tarefas pessoais. Utilizando os conceitos do **processo unificado**, elabore um plano inicial para o desenvolvimento deste software.

No seu planejamento, inclua uma descrição breve de cada uma das quatro fases do processo unificado (Concepção, Elaboração, Construção e Transição) aplicadas ao seu projeto. Especifique quais atividades seriam realizadas em cada fase e que tipo de artefatos (documentos, diagramas, código etc.) você espera produzir. Seu plano deve demonstrar uma compreensão clara do processo unificado e como ele pode ser aplicado para gerenciar efetivamente o ciclo de vida do desenvolvimento de software.

Vamos Começar!

Características do processo unificado

ANÁLISE E MODELAGEM DE SISTEMAS

O processo de software refere-se ao conjunto de métodos, técnicas e padrões empregados na produção de software. Dentro da ampla gama de abordagens existentes, o Processo Unificado (PU) se destaca como uma síntese de várias metodologias correlatas. Esta abordagem foi concebida e nomeada como Processo Unificado pelos renomados especialistas em software Jacobson, Booch e Rumbaugh em 2000, que combinaram suas experiências e conhecimentos para criar um modelo de desenvolvimento de software coeso e eficiente.

No processo unificado, um processo define quem está fazendo o quê, quando e como alcançar um determinado objetivo. No PU, os elementos do processo destacados referem-se a:

- Quem (papel) está fazendo.
- O que (artefato).
- Como (atividade).
- Quando (disciplina).

Um planejamento eficaz, fundamentado nos princípios da engenharia de software, é essencial para desenvolver um produto de software de alta qualidade, minimizando os riscos associados ao projeto, incluindo atrasos e custos excessivos. Assim como outros produtos, o software passa por um ciclo de vida que inclui várias fases, desde o início (concepção) até a maturidade e uso completo, e eventualmente chegando ao fim de sua utilidade, como demonstrado em um ciclo de vida típico ilustrado na Figura 1.



Figura 1 | 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. O ciclo de vida do produto engloba quatro fases principais: introdução (ou concepção), crescimento (que inclui o desenvolvimento e a implantação), maturidade e declínio. Por outro lado, o modelo do ciclo de vida de desenvolvimento refere-se à estrutura adotada para gerenciar o processo de criação de um software. Na engenharia de software, vários modelos são

ANÁLISE E MODELAGEM DE SISTEMAS

utilizados para estruturar o ciclo de vida de desenvolvimento, incluindo o modelo em cascata, espiral, prototipação, incremental e iterativo, entre outros. Estes modelos ajudam a organizar as várias etapas do desenvolvimento de software, desde a concepção até a entrega final.

O modelo em cascata, conhecido por ser um dos mais antigos, é estruturado em cinco etapas principais: (i) análise e levantamento de requisitos, (ii) projeto, (iii) desenvolvimento, (iv) teste e (v) implantação. No modelo incremental, o software é entregue ao cliente em módulos, cada um passando por todas as fases do modelo em cascata. Em contraste, o modelo iterativo ou evolutivo envolve a participação ativa do cliente na fase de análise, mesmo quando todos os requisitos e funcionalidades do software ainda não estão completamente definidos ou a relação entre as atividades não está clara. Inicialmente, realiza-se uma análise básica, seguida da implementação do software. Com o tempo, essa análise é aprimorada, levando à entrega de versões subsequentes e melhoradas do software.

Neste amplo leque de opções de metodologias, destaca-se o Processo Unificado (PU), o tema central desta discussão. O PU é um modelo adaptativo que se inspira no modelo incremental, focando na construção iterativa do software. Este modelo aprimora os processos tradicionais incrementais e iterativos, abordando e mitigando algumas de suas limitações. Por exemplo, no modelo iterativo, há o risco de o software nunca ser concluído devido a constantes solicitações de alterações por parte do cliente, e se a documentação não for bem gerida, o resultado pode ser um software desorganizado. No modelo incremental, a necessidade de completar cada parte integralmente antes de avançar pode ser um entrave. O Processo Unificado, proposto por Jacobson, Booch e Rumbaugh, em 2000, combina os aspectos dos modelos iterativo e incremental, permitindo que o software se desenvolva de forma robusta e coesa, evoluindo gradualmente para a maturidade do processo.

No Processo Unificado (PU), a Linguagem de Modelagem Unificada (UML) é uma ferramenta indispensável, fornecendo um meio eficiente e padronizado para a modelagem de sistemas de software. Como um conjunto de notações gráficas padronizadas, a UML é utilizada para a criação de representações visuais abrangentes dos sistemas, facilitando a visualização, especificação, construção e documentação dos componentes de software. Sua aplicação no PU, um framework adaptativo para o desenvolvimento de software, é fundamental para a comunicação eficaz entre desenvolvedores e demais stakeholders, assegurando que todos os elementos do sistema sejam entendidos e validados antes de sua implementação. Além disso, a UML desempenha um papel chave na organização do processo de desenvolvimento dentro do PU, permitindo a definição clara de requisitos, arquiteturas e componentes do software, bem como o acompanhamento do progresso do projeto através das fases iterativas e incrementais do PU.

O Processo Unificado (PU) foi desenvolvido após a criação da UML, que surgiu da combinação de três metodologias orientadas a objeto: Booch, OMT e OOSE, como destacado por Jacobson no prefácio do livro "*El Proceso Unificado de Desarrollo de Software*" (JACOBSON; BOOCH; RUMBAUGH, 2000). A UML rapidamente se tornou um padrão na indústria de desenvolvimento de software, mas por si só, não constituía um processo completo para o ciclo de desenvolvimento.

ANÁLISE E MODELAGEM DE SISTEMAS

Para preencher essa lacuna, a mesma equipe que desenvolveu a UML criou o Processo Unificado Racional (RUP), uma versão mais refinada e detalhada do PU. O RUP, um processo proprietário originalmente desenvolvido pela Rational Software (e mais tarde adquirido pela IBM), foi concebido pelos “três amigos”. Enquanto o RUP é mais detalhado e inclui disciplinas adicionais em comparação ao PU, este último é de domínio público e abrange o ciclo de desenvolvimento de software de forma mais ampla.

Tendo estabelecido uma compreensão inicial do Processo Unificado (PU) no contexto da engenharia de software, vamos agora aprofundar nosso conhecimento sobre esta metodologia significativa. Conforme descrito por Jacobson, Booch e Rumbaugh em 2000, o PU é caracterizado por três aspectos fundamentais: I. orientação por casos de uso, II. foco na arquitetura e III. natureza iterativa e incremental. A seguir, detalharemos cada um desses aspectos essenciais para entender melhor o PU.

- I. No Processo Unificado (PU), os casos de uso são elementos centrais, como explicado por Jacobson, Booch e Rumbaugh (2000). Eles são essenciais para entender as necessidades e desejos dos futuros usuários do sistema. É importante que os casos de uso se concentrem nas necessidades específicas de cada usuário, em vez de se limitarem apenas às funções do sistema. Assim, o modelo de casos de uso orienta as fases de projeto (design), implementação e testes do software, seguindo uma sequência lógica de fluxos de trabalho.
- II. O segundo aspecto do PU é o foco na arquitetura. Assim como na construção de um carro, onde se considera o design, a mecânica, o sistema elétrico e a aerodinâmica, a arquitetura do software proporciona uma visão geral do sistema. Essa visão abrangente deve incluir tanto as necessidades dos usuários quanto os objetivos estratégicos da empresa, fornecendo ao desenvolvedor um entendimento completo do sistema.
- III. O terceiro aspecto, iterativo e incremental, envolve dividir o projeto em subprojetos menores, ou iterações, onde cada uma resulta em um incremento do produto. Jacobson, Booch e Rumbaugh (2000) destacam que "as iterações se referem aos passos no fluxo de trabalho, e os incrementos, ao desenvolvimento do produto". O método iterativo no PU é bem controlado, o que diminui os riscos associados a custos crescentes e prazos estendidos, e acelera o desenvolvimento ao reconhecer que, especialmente em sistemas complexos, é improvável definir completamente os requisitos e necessidades dos usuários no início do projeto.

Ciclo de vida do processo unificado

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:** esta fase estabelece a visão geral do projeto, definindo o escopo e os requisitos iniciais. É o momento de determinar os objetivos principais e delinear o que o projeto pretende alcançar.

ANÁLISE E MODELAGEM DE SISTEMAS

- **Elaboração:** aqui, os requisitos e a arquitetura do sistema são detalhados e refinados. Esta fase também envolve uma análise aprofundada dos riscos e a elaboração de estimativas mais precisas para o desenvolvimento do projeto.
- **Construção:** esta é a fase de desenvolvimento ativo do sistema, onde a construção começa com os elementos mais simples e avança para os mais complexos. Também se inicia a preparação para a implantação do sistema.
- **Transição:** a última fase do ciclo é dedicada à implantação do sistema, momento em que o software é entregue. Esta fase envolve a finalização dos detalhes para garantir que o sistema esteja pronto para ser usado no ambiente de destino.

Observe, na Figura 2, 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).

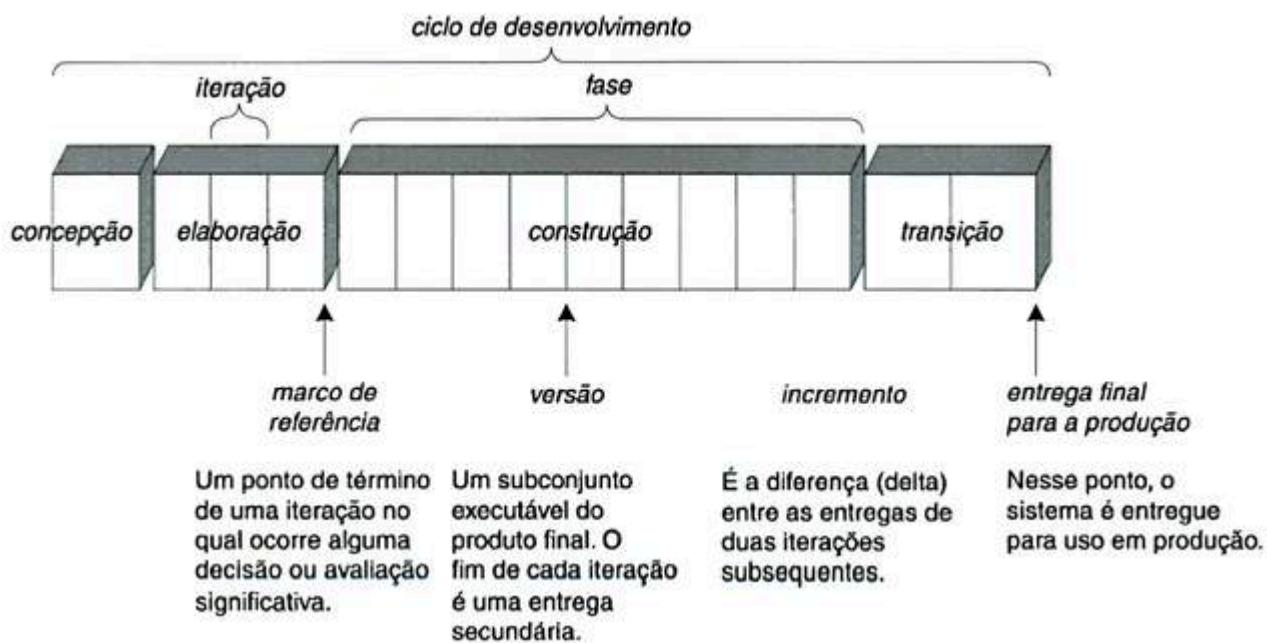


Figura 2 | Ciclo de vida do PU. Fonte: Larman (2007, p. 62)

O Processo Unificado (PU) estrutura-se em torno de quatro elementos fundamentais que definem o processo: o papel (quem), o artefato (o quê), a atividade (como) e a disciplina (quando):

- **Papel (trabalhador):** define as responsabilidades de cada pessoa envolvida no projeto, especificando as funções e tarefas de cada trabalhador em determinado momento. Um indivíduo pode assumir diferentes papéis e realizar diversas atividades ao longo do projeto.
- **Artefato:** representa qualquer produto gerado durante o processo de trabalho, que pode incluir código-fonte, esquemas de banco de dados, diagramas, modelos, entre outros. São os resultados tangíveis produzidos pelos trabalhadores.

ANÁLISE E MODELAGEM DE SISTEMAS

- **Atividade:** refere-se às tarefas executadas pelos trabalhadores com o objetivo de criar ou modificar um artefato. Cada atividade é um passo necessário na produção ou alteração de um artefato.
- **Disciplina (fluxo de trabalho):** envolve a organização das atividades em uma sequência lógica que leva a um resultado significativo. Os fluxos de trabalho são abordados sob três perspectivas: a dinâmica (relacionada ao tempo), a estática (focada nas atividades específicas) e as boas práticas. Dentro de cada perspectiva, existem várias disciplinas que orientam a execução do projeto.

Siga em Frente...

Fluxo de trabalho do processo unificado

As disciplinas fundamentais do Processo Unificado (PU), também conhecidas como fluxos de trabalho, incluem: (i) modelagem de negócios, (ii) requisitos, (iii) análise e projeto, (iv) implementação, e (v) teste e implantação. Estas disciplinas constituem o núcleo principal do nosso estudo. Além delas, existem disciplinas de suporte, como gerenciamento de projeto, gerenciamento de configurações e mudanças, e o ambiente. A Figura 3 ilustra como essas disciplinas são aplicadas durante as fases do ciclo de vida do desenvolvimento e em suas várias iterações. Cada iteração, que representa um minissistema, termina com a entrega de um incremento, momento em que se avalia o cumprimento dos objetivos e se fazem os ajustes necessários. Embora todas as disciplinas possam estar envolvidas em cada iteração, a quantidade de tempo dedicado a cada uma varia. Por exemplo, a Figura 3 mostra que nas primeiras iterações há um foco maior nas fases de concepção e elaboração, enquanto nas iterações finais, outras disciplinas recebem mais atenção, o que reflete a progressão natural no desenvolvimento de um sistema.

ANÁLISE E MODELAGEM DE SISTEMAS

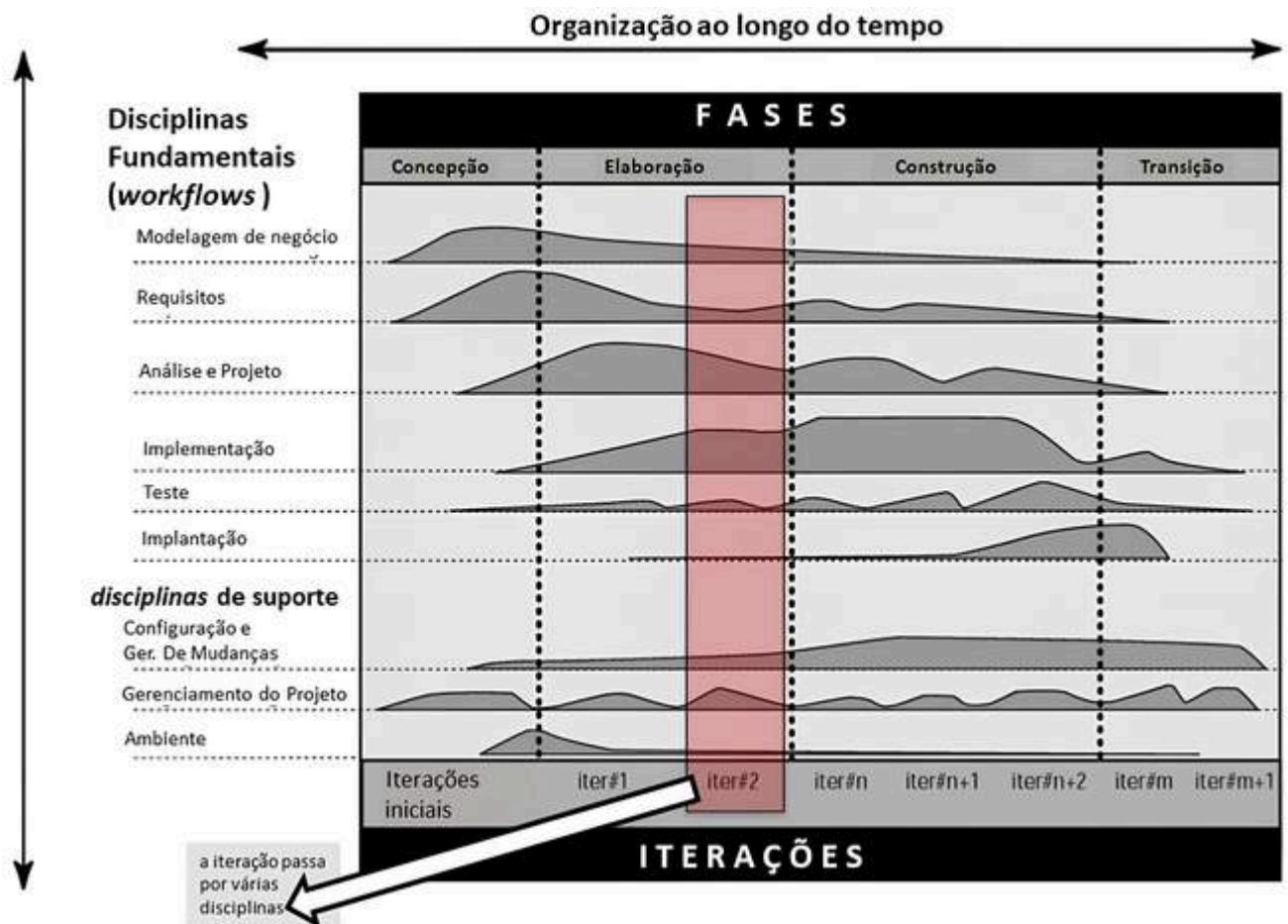


Figura 3 | Fluxo das iterações ao longo do tempo e das disciplinas. Fonte: IBM (2005).

Vamos agora analisar o que cada uma destas disciplinas contempla.

- Modelagem de negócio:** esta fase tem como objetivo documentar e analisar os objetivos de negócio envolvidos no projeto, com o intuito de compreender como o negócio deve suportar os processos associados. Embora muitos projetos possam optar por não realizar uma modelagem de negócios detalhada, esta etapa é fundamental para alinhar o projeto do sistema com as necessidades do negócio. Casos de uso de negócios são frequentemente empregados nesta fase para ajudar na documentação e análise.
- Requisitos:** esta fase visa definir claramente o que o sistema deve fazer. Ela se baseia principalmente em casos de uso para identificar os requisitos do sistema. Um aspecto importante nesta etapa é a identificação dos atores, que representam os usuários e quaisquer outros sistemas que interajam com o sistema em desenvolvimento. Conforme um artigo da IBM (2005, p. 4, tradução nossa) indica, é crucial reconhecer todos os atores envolvidos. O foco desta fase está nas funcionalidades do sistema, e é essencial que tanto os desenvolvedores quanto os clientes concordem com a descrição dos requisitos.
- Análise e Projeto:** estas fases são responsáveis pela criação dos modelos do sistema, que servirão de base para futuras implementações. Juntas, elas aprimoram a compreensão dos

ANÁLISE E MODELAGEM DE SISTEMAS

requisitos funcionais, estabelecidos nos casos de uso. Requisitos funcionais são declarações dos serviços que o sistema deve fornecer, de como ele deve reagir a entradas específicas e de como deve se comportar em determinadas situações. Larman (2005) destaca que a análise se concentra mais na investigação do problema e dos requisitos do que na solução propriamente dita, focando em como o sistema será usado e quais funções ele deve desempenhar, enquanto a implementação lida com o 'como fazer'. Na fase de projeto a atenção é voltada para a concepção da solução, isto é, a criação de modelos que atendam aos requisitos estabelecidos. Esta etapa é direcionada aos desenvolvedores, e não aos usuários finais. Ela envolve a definição clara de subsistemas, com a organização de classes em pacotes, delineando os componentes que serão implementados. Um aspecto crucial do projeto é seu foco na arquitetura do sistema, que implica em representar diversas visões ou abstrações das características mais significativas do sistema. Essa abordagem orientada para a arquitetura facilita a realização de mudanças futuras, especialmente quando os requisitos funcionais passam por alterações, garantindo assim maior flexibilidade e adaptabilidade ao longo do desenvolvimento do software.

4. **Implementação:** esta fase é dedicada à efetiva construção do software, ou seja, à programação propriamente dita. Aqui, o código é desenvolvido, configurando a parte prática do projeto, onde os desenvolvedores colocam a "mão na massa". A fase de implementação também inclui a preparação para os primeiros testes, conhecidos como testes beta.
5. **Testes:** nesta etapa, é crucial definir os procedimentos de teste a serem seguidos. Conforme o documento "RUP: boas práticas para o desenvolvimento de software" (IBM, 2005), os principais objetivos dos testes são verificar a interação entre objetos e componentes, assegurar que todos os requisitos foram implementados corretamente e identificar e resolver quaisquer defeitos antes da implantação do software.
6. **Implantação:** o foco do fluxo de trabalho de implantação é garantir a entrega bem-sucedida do produto aos usuários. Esta fase inclui a aceitação formal do software pelo cliente. As atividades envolvidas na implantação abrangem a distribuição, instalação e migração de dados, bem como a integração com softwares já existentes. Este estágio final assegura que o software esteja pronto para uso no ambiente destinado, atendendo às necessidades dos usuários finais.

As disciplinas adicionais, focadas em suporte, foram integradas no Processo Unificado Racional (RUP) para complementar as áreas não cobertas pelo PU. Estas incluem o gerenciamento de mudanças e configurações, o gerenciamento de projeto e o gerenciamento do ambiente de desenvolvimento. Essas disciplinas são essenciais para alcançar a maturidade do projeto, oferecendo estruturas e processos necessários para gerenciar eficientemente a evolução do projeto, assegurar a qualidade e facilitar a coordenação e execução das tarefas de desenvolvimento.

Vamos Exercitar?

Chegou o momento de resolver o desafio que lhe foi proposto no início da aula! O plano ilustra a aplicação prática do Processo Unificado no desenvolvimento de um aplicativo de gerenciamento

ANÁLISE E MODELAGEM DE SISTEMAS

de tarefas pessoais. Cada fase é abordada com atividades e artefatos específicos, demonstrando a abordagem iterativa e incremental do Processo Unificado. Esse planejamento enfatiza a importância da preparação cuidadosa e da resposta flexível às mudanças, elementos-chave para o sucesso no desenvolvimento de software.

1. Fase de Concepção

- **Escopo do projeto:** definir o objetivo do aplicativo de gerenciamento de tarefas, identificando as necessidades dos usuários e os benefícios esperados.
- **Requisitos iniciais:** levantar requisitos básicos, como a capacidade de criar, editar e organizar tarefas, definir lembretes e categorizar tarefas.
- **Casos de uso principais:** desenvolver casos de uso para funções críticas, como adicionar uma nova tarefa, definir prioridades e receber notificações.
- **Artefatos:** documento de visão, lista inicial de requisitos, casos de uso iniciais.

2. Fase de Elaboração

- **Arquitetura do sistema:** detalhar a arquitetura do software, escolhendo tecnologias apropriadas e definindo a estrutura de módulos e interfaces.
- **Plano de mitigação de riscos:** identificar riscos potenciais, como dependências tecnológicas e desafios de integração, propondo estratégias para mitigá-los.
- **Artefatos:** especificação de arquitetura, modelos de análise e projeto, plano de mitigação de riscos.

3. Fase de Construção

- **Desenvolvimento e teste:** organizar o desenvolvimento em iterações, com foco na implementação de funcionalidades conforme definido nos casos de uso. Realizar testes contínuos para garantir a qualidade do software.
- **Artefatos:** código fonte, testes automatizados, relatórios de teste.

4. Fase de Transição

- **Entrega e implantação:** preparar o software para lançamento, realizando testes beta com usuários finais e corrigindo quaisquer problemas identificados.
- **Treinamento e suporte:** desenvolver materiais de treinamento para usuários e estabelecer um plano de suporte pós-lançamento.
- **Planejamento de lançamento:** definir uma estratégia de lançamento, incluindo comunicação de marketing e cronograma de lançamento.
- **Artefatos:** versão final do software, manuais de usuário, plano de lançamento.

Saiba mais

ANÁLISE E MODELAGEM DE SISTEMAS

Leia mais sobre o Processo Unificado no livro *Engenharia de Software*, Capítulo 2, seção 2.5, disponível na Biblioteca Virtual.

PRESSMAN, R. S.; MAXIM, B. R. [Engenharia de software](#). Grupo A, 2021.

O Processo Unificado se baseia na Linguagem de Modelagem Unificada (UML). Para entender mais sobre essa linguagem, acesse o livro *UML Essencial*, Capítulo 2, disponível na Biblioteca Virtual.

FOWLER, M. [UML essencial](#). Grupo A, 2011.

Ouça o Podcast do Senai Play – Episódio 8 – Processo Unificado, e saiba mais sobre este assunto.

Referências

IBM. **Rational Unified Process**: Best practices for software development teams. From the developer Works archives. IBM Staff, jul. 2005.

JACOBSON, I.; BOOCH, G.; RUMBAUGH, J. **El Proceso Unificado de Desarrollo de Software**. 1. ed. rev. Madrid: Pearson Educación S.A., 2000.

LARMAN, C. **Utilizando UML e padrões**. 3. ed. São Paulo: Bookman, 2007.

REZENDE, D. A. **Engenharia de Software e sistemas de informações**. 2. ed. Rio de Janeiro: Brasport, 2002.

Aula 5

Encerramento da Unidade

Videoaula de Encerramento

ANÁLISE E MODELAGEM DE SISTEMAS



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Olá, estudante! Nesta videoaula, exploraremos os fundamentos essenciais para o analista de sistemas, mergulhando nos modelos de processo, desde a abordagem ágil até o detalhado Processo Unificado. Compreender esses conceitos é crucial para sua prática profissional, capacitando-o a conduzir análises mais precisas e eficientes. Do dinamismo dos processos ágeis à estrutura do Processo Unificado, esta aula oferecerá insights valiosos para aprimorar suas habilidades e destacar-se como um analista de sistemas exemplar.

[Clique aqui](#) para acessar os slides da sua videoaula.

Vamos começar!

Ponto de Chegada

Olá, estudante! Para desenvolver a competência desta Unidade, que é compreender os fundamentos da análise de sistemas e as características dos diversos tipos de processos de software, você deverá primeiramente conhecer os princípios da análise de sistemas.

A análise de sistemas é um processo que envolve estudos detalhados para identificar a solução ideal para o desenvolvimento de um sistema, baseando-se em métodos e técnicas de pesquisa e especificação. Este processo é dividido em várias fases, incluindo análise, projeto, implementação, testes, documentação e manutenção. Cada fase tem um papel específico, desde a avaliação de viabilidade e especificação de funcionalidades até a codificação, teste e documentação do software. Além disso, é importante incluir um processo de homologação para a aprovação oficial do cliente. Sommerville (2018) e Pressman (2021) ressaltam a importância da clareza e compartilhamento de informações, definição progressiva de funcionalidades, ilustração do comportamento do software, estruturação de diagramas em camadas para decompor problemas complexos, e detalhamento dos aspectos de implementação conforme o projeto avança.

Outro ponto importante para se compreender é o papel do analista de sistema. É um profissional responsável por liderar a análise de sistemas, incluindo atividades como pesquisa, planejamento, coordenação de equipes de desenvolvimento e recomendação de soluções de software adequadas às necessidades empresariais. Este papel abrange desde a concepção até a implementação e implantação de software, começando pela definição das funcionalidades do sistema e compreensão das necessidades dos usuários para organização, especificação e documentação adequadas. Este profissional deve ter conhecimentos abrangentes sobre várias

ANÁLISE E MODELAGEM DE SISTEMAS

áreas de negócios e estar disposto a se informar sobre campos específicos quando necessário. As habilidades importantes para um analista de sistemas incluem estar atualizado com as tecnologias, possuir organização, visão gerencial e excelentes habilidades interpessoais.

O analista de sistemas age como um intermediário entre os desenvolvedores e os usuários finais, coletando e interpretando informações dos usuários para comunicá-las aos desenvolvedores de forma técnica, garantindo que o software atenda às expectativas de clientes e usuários. As responsabilidades do analista de sistemas durante o desenvolvimento de software incluem interagir com clientes e usuários, avaliar a viabilidade do projeto, coletar informações, identificar requisitos, desenvolver modelagem do software, orientar desenvolvedores, realizar testes, gerenciar documentação, administrar mudanças no projeto, estabelecer padrões de desenvolvimento, garantir a qualidade do software, realizar monitoramento e auditorias, organizar treinamentos para usuários, supervisionar a implantação do software e oferecer consultoria técnica. O analista também deve se manter informado sobre novas tecnologias e buscar constante aperfeiçoamento profissional.

Um **Processo de software** é um conjunto de atividades e resultados inter-relacionados que levam à criação de um software. Segundo Sommerville (2018) e Pressman (2021), na Engenharia de software, esse processo não é uma metodologia rígida, mas sim uma abordagem flexível que permite à equipe de desenvolvimento escolher processos que se alinhem com a filosofia da empresa, buscando qualidade, cumprimento de prazos e redução de custos. O Processo de software geralmente inclui especificação, design, implementação, validação, manutenção e evolução. Esse processo estabelece padrões para a criação de serviços e produtos, facilita a repetição e reutilização de componentes, preserva conhecimento dentro da empresa, define atividades de projetos de software, especifica processos de desenvolvimento, determina tarefas, minimiza riscos, melhora a comunicação na equipe e pode ser usado como modelo para novos projetos.

Engholm Jr. (2010) explica que, ao estabelecer Processos de software, são determinados vários parâmetros, incluindo o evento que inicia o processo, a matriz de responsabilidades, as atividades a serem executadas em ordem, as entradas e saídas de cada atividade, as regras e políticas, a infraestrutura necessária e os resultados de cada processo. Em um Processo de Software, diversas atividades podem ocorrer de forma sequencial ou em paralelo, envolvendo mudanças nas atividades realizadas em cada fase e estabelecendo responsabilidades, prazos e estratégias para alcançar os objetivos. Mesmo variando entre empresas, todos os Processos de software utilizam uma estrutura genérica com atividades pré-estabelecidas.

As etapas fundamentais de qualquer Processo de software são essenciais para desenvolver e entregar um produto de software. De acordo com Sommerville (2018), em um Processo genérico de software, existem quatro atividades chave: Especificação, Projeto e Implementação, Validação e Evolução. A Especificação envolve definir o escopo e funcionalidades do projeto. O Projeto e Implementação abrangem o desenho e a programação do software. A Validação verifica se o software atende aos requisitos do cliente, e a Evolução trata de atualizações e melhorias do software.

ANÁLISE E MODELAGEM DE SISTEMAS

Pressman (2021) adiciona que uma metodologia genérica em Engenharia de software inclui cinco atividades essenciais: Comunicação, para compreender os objetivos do projeto; Planejamento, para detalhar tarefas técnicas e cronograma; Modelagem, que envolve criar modelos para entender as necessidades do software; Construção, que é a codificação e teste do software; e Entrega, onde o software é entregue e testado pelo cliente, com adaptações e correções feitas conforme necessário.

Cada projeto de software é único e requer diferentes conjuntos de tarefas e abordagens. Os analistas de sistemas determinam essas tarefas com base nas características e necessidades específicas do projeto. Por exemplo, na criação de interfaces de um sistema, o programador desenvolve as telas com base em requisitos específicos, e a qualidade do trabalho é avaliada conforme essas especificações.

A implementação de um Processo de software não garante automaticamente a qualidade do produto final, nem assegura que o software será entregue dentro do prazo ou que atenderá integralmente aos requisitos do cliente. A **qualidade do software** depende diretamente dos padrões de qualidade aplicados durante o processo. Estabelecer procedimentos e padrões rigorosos é fundamental para assegurar a qualidade em todas as etapas do processo. A **modelagem de processos** permite revisões e melhorias contínuas antes da finalização do software.

Existem diversos tipos de modelos de processos de software. O **modelo de processo prescritivo**, também conhecido como **modelo de processos tradicionais**, inclui um conjunto de elementos que englobam ações de engenharia de software, produtos de trabalho e mecanismos para garantir a qualidade e controlar mudanças em projetos de desenvolvimento de software. Este modelo orienta o desenvolvimento de software de forma estruturada e ordenada, com tarefas realizadas sequencialmente e diretrizes claras (SOMMERVILLE, 2018).

Dentre os modelos de processos prescritivos mais conhecidos estão o **modelo cascata**, **modelo incremental** e os **modelos evolucionários**, como prototipação e espiral. O **modelo cascata** é caracterizado por sua abordagem sistemática e sequencial, com cada fase iniciando após a conclusão da anterior. As etapas fundamentais deste modelo incluem definição de requisitos, projeto do sistema, implementação e teste de unidade, integração e teste de sistemas, e operação e manutenção. Este modelo é conhecido por sua simplicidade e clareza, facilitando o gerenciamento, mas pode prolongar o tempo de desenvolvimento devido à sua natureza sequencial.

O **modelo incremental**, por outro lado, é iterativo e desenvolve o software em etapas. Com base em requisitos iniciais, são criadas versões menores do software, entregues ao cliente de forma progressiva até a conclusão do sistema. Cada versão incremental adiciona ou melhora funcionalidades, oferecendo entregas parciais durante o desenvolvimento.

O **modelo espiral** combina a abordagem iterativa da prototipação com a estrutura sistemática do modelo cascata. Cada ciclo iterativo no modelo espiral resulta em uma versão mais completa do

ANÁLISE E MODELAGEM DE SISTEMAS

software, visando um desenvolvimento rápido e eficiente.

Esses modelos prescritivos são essenciais para organizar e estruturar o desenvolvimento de software, oferecendo diferentes abordagens dependendo das necessidades específicas do projeto.

Os **modelos de processos especializados** representam uma fusão de elementos de diferentes modelos de processos prescritivos, adaptados para situações que exigem abordagens mais específicas e detalhadas. Um exemplo é o **modelo baseado em componentes**, amplamente usado em projetos comerciais, onde se utilizam componentes de software pré-fabricados para reutilização. Outro modelo, o de **métodos formais**, concentra-se em criar especificações matemáticas formais para o software, facilitando a eliminação de ambiguidades e inconsistências e fortalecendo a verificação do código.

Além disso, o **desenvolvimento de software orientado a aspectos** organiza o código por importância, abordando diversos aspectos do sistema, como classes orientadas a objetos. O **modelo de processo unificado**, ou RUP, integra características dos modelos prescritivos com a metodologia Ágil, sendo iterativo e incremental. Por fim, há o **modelo de processos pessoal e de equipe**, que enfatiza o desenvolvimento de software como uma atividade coletiva, com foco na medição individual do trabalho no modelo pessoal, e na criação de equipes auto-organizadas e autodirigidas no modelo de equipe, visando a produção de software de alta qualidade.

A evolução constante das tecnologias e dos processos de negócios tem impulsionado a necessidade de um desenvolvimento de software mais ágil e rápido, levando ao surgimento da **Metodologia Ágil**. Esta abordagem traz maior flexibilidade e dinamismo ao processo de desenvolvimento de software, focando na rapidez e flexibilidade para atender às necessidades dos clientes. A Metodologia Ágil é caracterizada por envolver ativamente o cliente no desenvolvimento, adotar uma abordagem de desenvolvimento incremental, dar autonomia à equipe, ser flexível às mudanças e buscar simplicidade no trabalho.

A Metodologia Ágil visa reduzir a necessidade de documentação extensiva, tornando o processo de desenvolvimento mais flexível e menos burocrático. Dois métodos ágeis se destacam: XP (*Extreme Programming*) e Scrum. O método XP enfatiza a retroalimentação constante, desenvolvimento incremental e comunicação eficaz, seguindo atividades metodológicas como planejamento, projeto, codificação e testes, com a participação ativa de um representante do cliente (PRESSMAN, 2021). Já o Scrum é um processo de desenvolvimento iterativo e incremental, aplicável também à gestão de projetos, e incorpora práticas e regras de gestão para o sucesso do projeto. Inclui atividades como a definição de uma lista de requisitos (*Product Backlog*), sprints para atingir objetivos específicos, reuniões de planejamento e reuniões diárias (*Daily Scrum*) para monitorar o progresso.

Além destes, existem outros métodos de Desenvolvimento Ágil, como o Desenvolvimento de Sistemas Dinâmicos (DSDM), Modelagem Ágil e o Processo Unificado Ágil, todos compartilhando uma ênfase na colaboração humana e na auto-organização.

ANÁLISE E MODELAGEM DE SISTEMAS

O **Processo Unificado** (PU) representa uma abordagem adaptável no desenvolvimento de software, caracterizado por sua natureza iterativa e incremental. A intenção do PU é alinhar o processo de desenvolvimento com os objetivos específicos de cada projeto, mantendo ao tempo a flexibilidade e a eficiência. Uma das características distintas do PU é sua divisão do projeto em mini fases ou iterações, onde cada uma resulta em um incremento do software. Isso permite fazer ajustes contínuos com base no feedback e nas mudanças de necessidades do projeto (ENGHOLM, 2010). O PU é fortemente centrado em casos de uso para definir os requisitos funcionais, guiando o desenvolvimento pelas necessidades dos usuários finais para assegurar a relevância e utilidade do software. Além disso, a abordagem dá grande ênfase à arquitetura do software, garantindo que o sistema seja robusto, escalável e fácil de manter.

No que diz respeito às fases do PU, ele se divide em quatro etapas principais. A fase de concepção estabelece a visão e o escopo do projeto, focando em identificar requisitos básicos, riscos potenciais e a viabilidade do projeto. A fase de elaboração detalha os requisitos e define a arquitetura base do sistema, além de avaliar os riscos principais, sendo crucial para o planejamento detalhado do projeto. Na fase de construção, o software é desenvolvido e testado, seguindo o design e a arquitetura previamente definidos, resultando em um produto quase final ou versão beta. A fase final, transição, é onde o produto é entregue aos usuários finais, envolvendo a resolução de quaisquer problemas identificados, ajustes finais e, em alguns casos, treinamentos para os usuários e a manutenção do sistema.

Dentre as metodologias derivadas do PU, destaca-se o *Rational Unified Process* (RUP), desenvolvido pela Rational Software e mais tarde adquirido pela IBM. O RUP personaliza o PU para fornecer um processo mais estruturado e detalhado, adaptando-se a uma variedade de projetos de software (IBM, 2005).

O PU é especialmente benéfico em projetos grandes e complexos, onde os requisitos podem mudar com o tempo. A natureza iterativa e incremental do PU facilita a gestão de riscos, adaptação a mudanças e promove uma colaboração eficaz entre equipes de desenvolvimento e partes interessadas.

É Hora de Praticar!



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

ANÁLISE E MODELAGEM DE SISTEMAS

Scrum e Processo Unificado representam duas abordagens distintas, porém complementares, no vasto cenário do desenvolvimento de software. O Scrum, uma metodologia ágil, destaca-se por sua flexibilidade e capacidade de se adaptar a mudanças rápidas, dividindo o trabalho em iterações mensuráveis chamadas sprints. Por outro lado, o Processo Unificado oferece uma estrutura mais estruturada, dividindo o desenvolvimento em fases definidas, como concepção, elaboração, construção e transição. Embora essas abordagens possam parecer contrastantes à primeira vista, a combinação equilibrada de elementos do Scrum e do Processo Unificado pode resultar em uma metodologia híbrida que aproveita o dinamismo ágil e a solidez estruturada, oferecendo uma abordagem eficaz para projetos complexos. Nesse contexto, exploraremos como essas metodologias podem ser integradas para otimizar o desenvolvimento de software, equilibrando agilidade e estrutura para alcançar resultados sólidos e adaptáveis.

Sabendo disso, vamos imaginar que você é um consultor encarregado de liderar uma equipe de desenvolvimento em um projeto altamente complexo que exige inovação e adaptabilidade.

Proponha uma estratégia que combine elementos das metodologias ágeis, como o Scrum, com o processo unificado, enfatizando como essa abordagem híbrida pode gerenciar eficientemente mudanças constantes nos requisitos, garantindo ao mesmo tempo uma arquitetura sólida e escalabilidade para o sistema. Apresente um plano de implementação detalhado, considerando possíveis desafios e soluções, destacando como essa sinergia entre ágil e processo unificado pode ser uma vantagem competitiva no desenvolvimento de projetos de alta complexidade.

- Como o papel do analista de sistemas evolui diante das rápidas transformações tecnológicas e das demandas crescentes por inovação nas organizações?
- Qual é a importância de considerar as características específicas de um projeto de software ao escolher o processo de desenvolvimento, e como as necessidades e requisitos únicos podem influenciar a decisão entre metodologias tradicionais, como a cascata, e abordagens ágeis, como o Scrum?
- Como a abordagem do processo unificado na modelagem de software, ao desafiar a segmentação tradicional de fases, contribui para uma visão mais fluida e interconectada, influenciando positivamente na criação de sistemas mais adaptáveis e robustos?

Dê o Play!

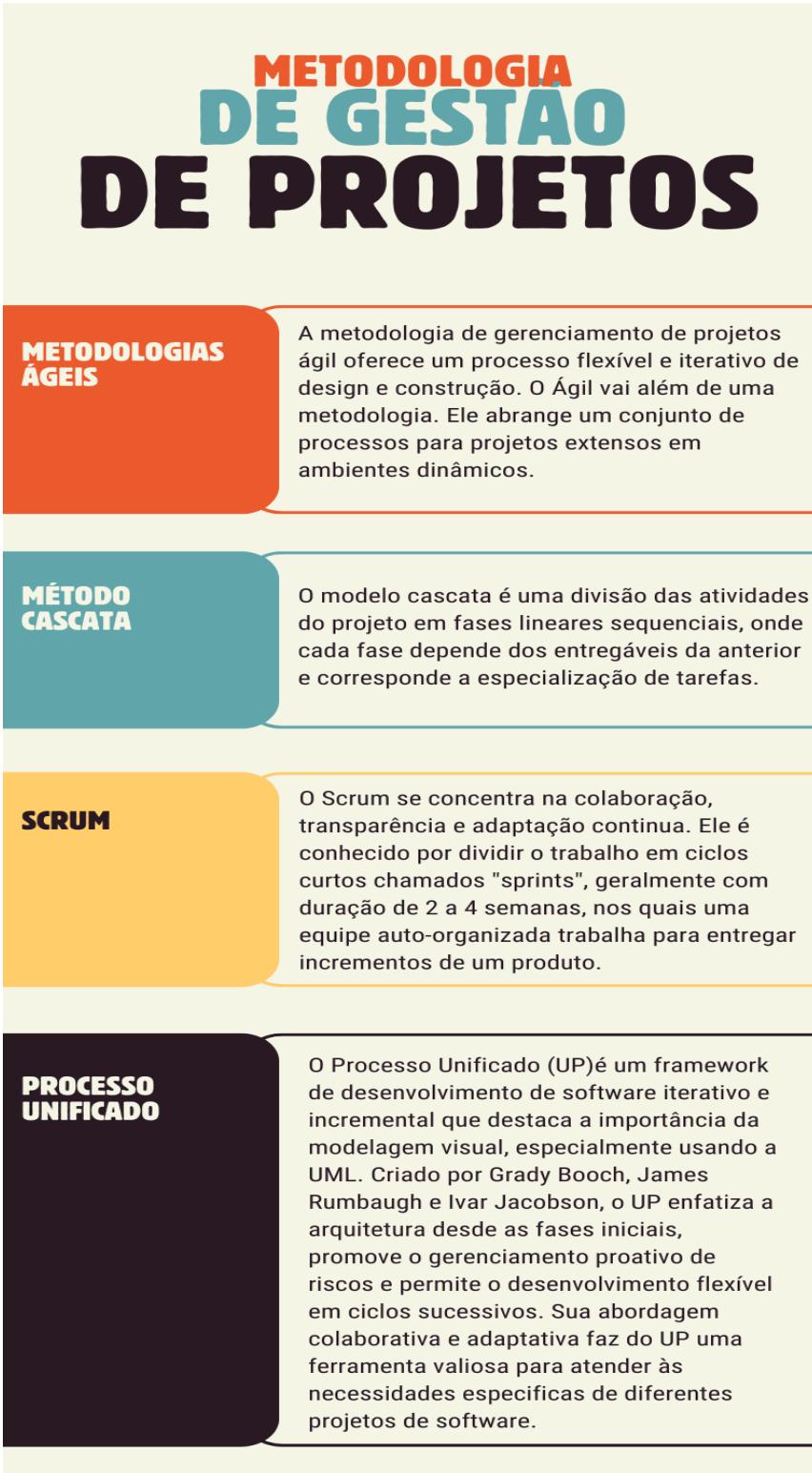
[Clique aqui](#) para acessar os slides do Dê o play!

Para implementar essa abordagem híbrida de Scrum com o processo unificado, o consultor poderia iniciar adotando sprints do Scrum para iterativamente desenvolver funcionalidades, enquanto, simultaneamente, a fase de Elaboração do processo unificado é utilizada para realizar análises detalhadas de requisitos e arquitetura. Durante os sprints, a equipe aplicaria princípios ágeis de flexibilidade e entrega contínua, enquanto a fase de Elaboração garantiria a estabilidade arquitetural e a integração consistente de novas funcionalidades. Desafios potenciais podem incluir a sincronização eficiente entre as práticas ágeis e tradicionais, que podem ser superados por uma comunicação clara e colaborativa entre as equipes envolvidas. A sinergia resultante entre Scrum e o processo unificado cria um ambiente propício para desenvolver projetos complexos de forma ágil e estruturada.

ANÁLISE E MODELAGEM DE SISTEMAS

Este infográfico oferece uma visão concisa e informativa sobre quatro abordagens fundamentais no desenvolvimento de software: Métodos Ágeis, Método cascata, Scrum e o Processo Unificado (PU). Ao explorar essas metodologias, o infográfico destaca as principais características e benefícios de cada uma. Descubra as práticas ágeis que promovem a flexibilidade e resposta rápida às mudanças, as iterações estruturadas do Scrum para gerenciamento de projetos, e a abordagem iterativa e focada em arquitetura do Processo Unificado.

ANÁLISE E MODELAGEM DE SISTEMAS



METODOLOGIA DE GESTÃO DE PROJETOS

METODOLOGIAS ÁGEIS

A metodologia de gerenciamento de projetos ágil oferece um processo flexível e iterativo de design e construção. O Ágil vai além de uma metodologia. Ele abrange um conjunto de processos para projetos extensos em ambientes dinâmicos.

MÉTODO CASCATA

O modelo cascata é uma divisão das atividades do projeto em fases lineares sequenciais, onde cada fase depende dos entregáveis da anterior e corresponde a especialização de tarefas.

SCRUM

O Scrum se concentra na colaboração, transparência e adaptação contínua. Ele é conhecido por dividir o trabalho em ciclos curtos chamados "sprints", geralmente com duração de 2 a 4 semanas, nos quais uma equipe auto-organizada trabalha para entregar incrementos de um produto.

PROCESSO UNIFICADO

O Processo Unificado (UP) é um framework de desenvolvimento de software iterativo e incremental que destaca a importância da modelagem visual, especialmente usando a UML. Criado por Grady Booch, James Rumbaugh e Ivar Jacobson, o UP enfatiza a arquitetura desde as fases iniciais, promove o gerenciamento proativo de riscos e permite o desenvolvimento flexível em ciclos sucessivos. Sua abordagem colaborativa e adaptativa faz do UP uma ferramenta valiosa para atender às necessidades específicas de diferentes projetos de software.

Figura | Infográfico - Abordagens fundamentais no desenvolvimento de software. Fonte: elaborada pela autora.

ANÁLISE E MODELAGEM DE SISTEMAS

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

IBM. **Rational Unified Process**: Best practices for software development teams. From the developer Works archives. IBM Staff, jul. 2005.

PRESSMAN, R. S. **Engenharia de software**: uma abordagem profissional. 9. ed. – Porto Alegre: AMGH, 2021.

SOMMERVILLE, I. **Engenharia de software**. 10. ed. São Paulo: Pearson, 2018.

WERLICH, C. **Análise e modelagem de sistemas**. Londrina: Editora e Distribuidora Educacional S.A., 2020.

Unidade 2

Processos de Negócio para Análise de Sistemas

Aula 1

Fundamentos de Processos de Negócios



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Olá, estudante! Nesta videoaula, você conhecerá a abordagem sobre Processos de negócio. Exploraremos desde o conceito fundamental até a classificação dos processos, proporcionando uma compreensão aprofundada. Com destaque para as visões funcional e ponta a ponta, esses conteúdos moldarão sua prática profissional, capacitando-o a analisar, otimizar e integrar eficientemente os processos empresariais. Esteja preparado para adquirir conhecimentos essenciais que ampliarão sua eficácia no cenário corporativo!

[Clique aqui](#) para acessar os slides da sua videoaula.

Bons estudos!

Ponto de Partida

ANÁLISE E MODELAGEM DE SISTEMAS

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.

Uma empresa do ramo de indústria alimentícia 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 da 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 a 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.

ANÁLISE E MODELAGEM DE SISTEMAS

Vamos Começar!

Conceito de processo de negócios

Você já refletiu sobre a diversidade de áreas que uma empresa abrange? É evidente que cada empresa possui uma estrutura única, atividades específicas e, por conseguinte, diversas áreas. Portanto, é crucial analisar cada organização com cuidado para compreender e identificar suas distintas áreas e como elas se inter-relacionam.

A integração da área de Tecnologia da Informação (TI) com as demais áreas de negócio desempenha um papel significativo. Essa integração é crucial para evitar falhas no desenvolvimento de software, garantindo que este esteja alinhado com os objetivos da empresa. A área de TI desempenha um papel estratégico nas organizações, contribuindo para o design e a gestão de processos que possibilitam a entrega de soluções mais eficazes aos clientes.

Um exemplo concreto é a experiência da área de TI no desenho de processos de negócio. A tecnologia desempenha um papel crucial na determinação de modelos de processos de negócio, otimizando e reduzindo custos. Isso permite que a empresa se posicione de maneira diferenciada e gere vantagens competitivas, atendendo às demandas dos clientes. Por exemplo, ao otimizar o processo de vendas, é possível aumentar a agilidade no atendimento, na entrega e na satisfação do cliente. Esse mesmo princípio pode ser aplicado a vários outros processos, como os financeiros e produtivos.

As áreas de negócio são aquelas cujo propósito é dar continuidade à missão organizacional, produzindo bens ou serviços que atendam às demandas do cliente externo. Estas atividades são consideradas essenciais, pois estão diretamente vinculadas à atividade central (core business) da organização.

Quando nos referimos às áreas de negócio, é imprescindível não negligenciar o processo de negócio, mas é necessário primeiro recordar o conceito de processo. Segundo Chiavenato (2014), um processo é uma sequência lógica e estruturada de tarefas que envolvem uma entrada (input) de vários elementos, os quais são processados para gerar saídas (outputs), conforme ilustrado na Figura 1.

ANÁLISE E MODELAGEM DE SISTEMAS



Figura 1 | Processo. Fonte: adaptada de Chiavenato (2014).

A estruturação do processo de negócio deve ser abordada a partir da perspectiva de processos, englobando inputs e outputs. Os inputs, que são as entradas do processo, incluem, essencialmente, aspectos relacionados à área de recursos humanos (como a disponibilidade de pessoas), máquinas e equipamentos (abrangendo desde computadores até maquinários produtivos), materiais (sejam insumos de produção ou outros), recursos financeiros, informações (provenientes do cliente, mercado, estoque), e procedimentos (definindo como a execução deve ocorrer). Por outro lado, os outputs representam os resultados gerados a partir do processamento de todos esses recursos de entrada, com o objetivo de criar um produto ou serviço (Paim, 2009).

Classificação de processos de negócio

Os processos podem ser categorizados como simples ou complexos, como no exemplo de ligar uma televisão ou fabricar uma televisão. A complexidade do processo tem impacto direto em sua dificuldade de modelagem e gerenciamento, exigindo uma documentação adequada que detalhe como as tarefas e atividades devem ser executadas, especificando quem é responsável pela execução das tarefas para alcançar o resultado desejado. É importante ressaltar que mesmo processos simples devem ser documentados, pois isso favorece a padronização e evita interferências em outros processos.

Quanto ao processo de decomposição, é possível categorizá-lo como macroprocesso, processo e subprocesso. A Figura 2 ilustra um exemplo de decomposição, em que os macroprocessos abordam as operações mais amplas da organização, sendo subdivididos em processos, os quais, por sua vez, são fragmentados em subprocessos. Por exemplo, uma montadora de automóveis pode ter a fabricação de veículos como macroprocesso, o qual contém vários processos,

ANÁLISE E MODELAGEM DE SISTEMAS

incluindo a produção, que, por sua vez, é desmembrada em subprocessos como pintura e montagem, entre outros.

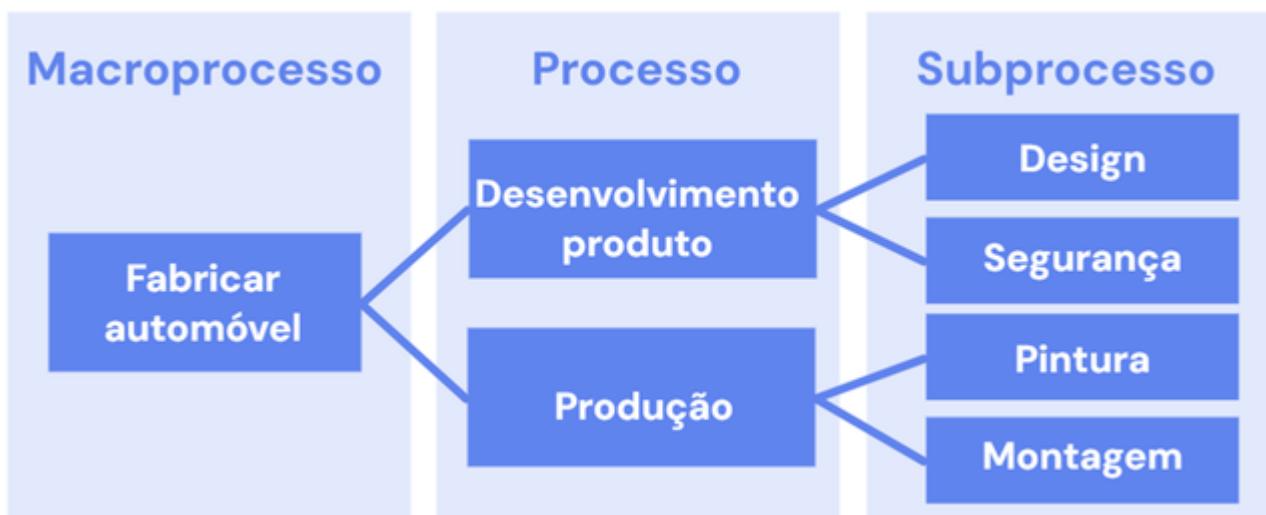


Figura 2 | Decomposição de macroprocesso, processo e subprocesso. Fonte: adaptada de Werlich (2020, p. 58).

Conforme destacado por Brocke e Rosemann (2013), o processo de negócio é a consolidação de atividades e tarefas com o objetivo de alcançar um resultado que represente valor agregado ao cliente. Ele organiza sequencialmente essas atividades e tarefas de maneira lógica, delineando como o trabalho deve ser executado por meio de um bem ou serviço. Os mesmos autores enfatizam que os processos de negócios podem ser classificados, com base em suas características, como processos primários, processos de suporte e processos de gerenciamento. É importante ressaltar que existe interação entre esses tipos de processos.

Os processos primários estão intrinsecamente ligados ao core business (negócio principal) da organização, ou seja, estão diretamente associados às atividades essenciais. São aqueles que agregam valor ao cliente final, e, como resultado, seus desempenhos refletem o nível de satisfação do cliente. Os processos primários desempenham um papel crucial na tradução da missão da organização, englobando atividades como a produção de produtos, a entrega ao cliente, o marketing, entre outras iniciativas voltadas para a geração de valor ao cliente (BROCKE E ROSEMANN, 2013).

É importante notar que os processos primários são capazes de iniciar e encerrar fora da organização, uma vez que estão diretamente conectados aos clientes, estabelecendo uma relação com a entrega final (a promessa feita ao cliente).

Os processos de suporte desempenham um papel fundamental ao sustentar as operações dos processos primários e de gerenciamento, ou seja, oferecem apoio aos dois outros grupos de processos. Eles auxiliam os processos primários na realização das entregas que agregam valor ao cliente, embora não forneçam valor diretamente ao cliente final. Em outras palavras, esses

ANÁLISE E MODELAGEM DE SISTEMAS

processos acrescentam valor ao próprio processo, não ao cliente. Uma execução eficiente dos processos de suporte contribui para que os processos primários sejam executados da melhor forma possível, gerando um impacto positivo no cliente. Essa eficácia, por sua vez, demanda um alinhamento organizacional inicial, assegurando que todos os colaboradores estejam alinhados na mesma direção. Posteriormente, é essencial ter uma clara delinearção dos processos primários, de suporte e de gerenciamento (BROCKE E ROSEMANN, 2013).

Setores como recursos humanos, tecnologia da informação, contabilidade e contas a pagar são exemplos de áreas de apoio aos processos primários, mas não mantêm uma conexão direta com o cliente final.

É importante notar que a avaliação dos processos de suporte deve ser contextualizada de acordo com cada organização. Por exemplo, em um escritório de contabilidade, onde o "produto" entregue é a contabilidade em si, os processos de suporte podem ser considerados como processos primários nesse contexto específico.

O processo de gerenciamento compartilha características semelhantes com o processo de suporte, uma vez que opera como um processo secundário. Ele tem a capacidade de agregar valor tanto aos processos primários quanto aos de suporte, porém, não proporciona, diretamente, valor ao cliente final. De acordo com Brocke e Rosemann (2013), o processo de gerenciamento está vinculado à supervisão e controle das atividades organizacionais. Esses processos são essenciais para monitorar os resultados, avaliar sua satisfatoriedade e, consequentemente, identificar melhorias que indiretamente resultarão em valor agregado ao cliente, mas por meio da otimização dos processos e não pela entrega direta de valor ao cliente. Seu objetivo é assegurar o alcance dos objetivos organizacionais. O monitoramento e controle são realizados por meio de práticas como o gerenciamento estratégico e o gerenciamento de desempenho, que servem para acompanhar, medir ou controlar o desenvolvimento dos processos primários e de suporte.

Quando mencionamos controle, estamos nos referindo aos indicadores de desempenho estabelecidos por cada organização, adaptados às suas atividades e tarefas específicas. É crucial ter em mente que cada organização terá indicadores distintos, abrangendo diferentes fatores a 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.

Siga em Frente...

Visão funcional e visão de processos ponta a ponta

ANÁLISE E MODELAGEM DE SISTEMAS

Gerenciar processos se mostra como um pensamento de grande relevância para organização, pois acarreta benefícios que gerarão impactos diretos no cliente. Alguns dos benefícios gerados pelo gerenciamento de processos estão ligados a (VALLE, OLIVEIRA E BRACONI, 2013):

- 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.

A abordagem funcional da organização está associada à sua estrutura hierárquica, caracterizando-se por um modelo de visualização vertical. Nesse contexto, os processos são analisados por departamento, e cada departamento gerencia recursos específicos de sua área (DE SORDI, 2018).

Essa perspectiva não enfatiza a interconectividade entre as áreas de negócios, resultando em uma percepção isolada de cada área, como se não houvesse integração entre elas. Isso se traduz em um processo de isolamento, assemelhando-se à situação em que as engrenagens de um relógio não se entrelaçam.

A abordagem funcional apresenta características de silos, onde cada atividade é conduzida de maneira isolada, com coordenação deficiente e falta de conhecimento dos processos interdependentes.

Essa abordagem se revela como uma escolha que oferece uma baixa orientação para o mercado, tornando-se incapaz de identificar as tendências e necessidades dos clientes. Os objetivos são tratados de maneira isolada, com cada departamento concentrando-se exclusivamente em suas metas individuais, sem priorizar os objetivos globais. Como resultado, as avaliações de desempenho ocorrem de forma departamental. O enfoque principal dessa visão reside no desenvolvimento de competências funcionais na equipe, priorizando, assim, uma visão restrita em vez de abrangente. Isso resulta em reconhecimento individualizado e limitado, sem considerar se a atividade contribuiu para agregar valor ao cliente ou ao processo que sustenta. Além disso, há características de orçamentos locais que não se comunicam com os demais orçamentos da empresa e não são influenciados pelos resultados globais, apenas pelos resultados locais. A grande consequência de todos esses pontos é a falta de preocupação com os processos como um todo.

ANÁLISE E MODELAGEM DE SISTEMAS

A Figura 3 ilustra a configuração de uma estrutura funcional. Também fica evidente que o fluxo de informação segue um padrão hierárquico, ou seja, de cima para baixo, resultando em uma comunicação vertical, sem estabelecer processos de comunicação horizontais. As diretorias e gerências operam de maneira individualizada, o que torna mais desafiador atender às necessidades do cliente, dado que a visão é estruturada de forma hierárquica.

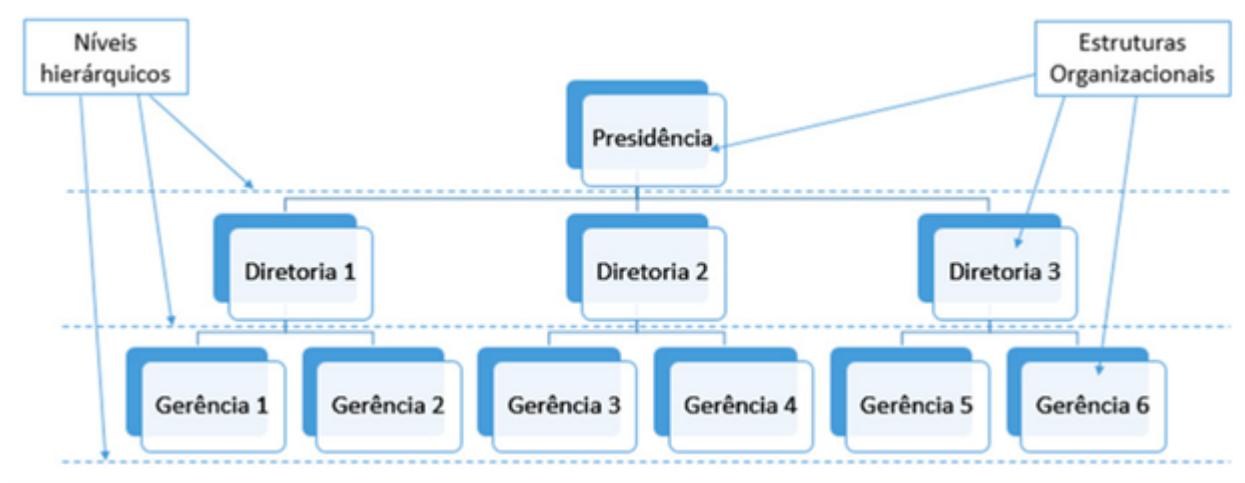


Figura 3 | Visão funcional. Fonte: Werlich (2020, p. 61).

Já a Figura 4 mostra a integração horizontal existente entre as diretorias e gerências e, portanto, com maior possibilidade de atender às necessidades dos clientes de forma satisfatória. Lembre-se: para a modelagem de sistema acontecer de forma adequada é necessária a integração entre as áreas (WERLICH, 2020).

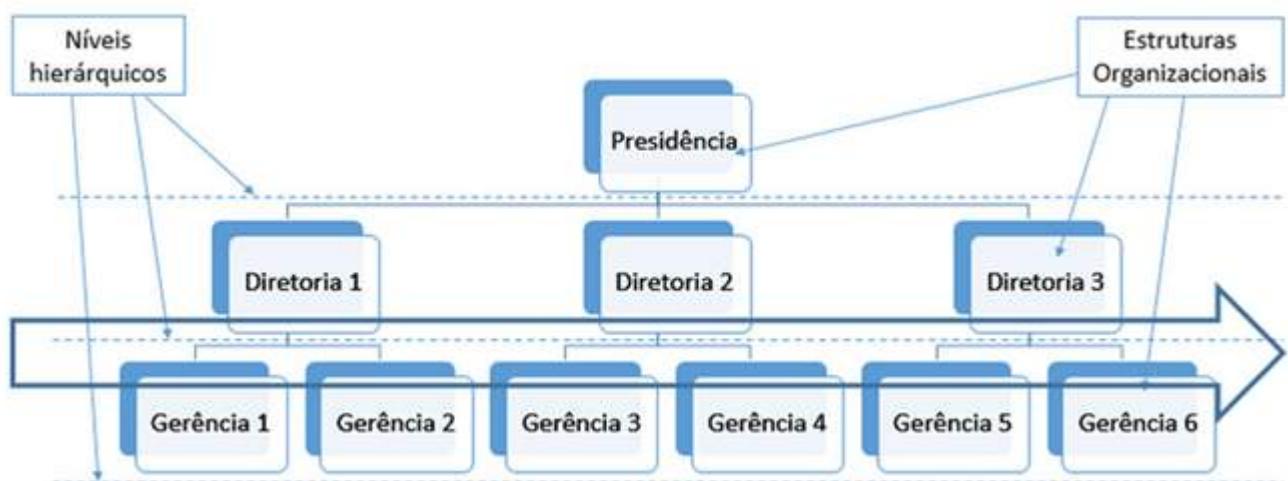


Figura 4 | Visão de processos. Fonte: Werlich (2020, p. 62).

A abordagem de processos ponta a ponta proporciona uma perspectiva abrangente ao percorrer e visualizar as conexões entre todos os departamentos, ou seja, em uma perspectiva horizontal.

ANÁLISE E MODELAGEM DE SISTEMAS

Ela engloba aspectos como tempo, custos, capacidade e qualidade, possibilitando uma compreensão da contribuição de cada parte para atender às necessidades do cliente. Essa abordagem viabiliza uma visualização em diversos níveis e representa uma maneira de agregar valor ao cliente.

Um processo ponta a ponta como aquele que pode ser interfuncional, ou seja, ultrapassa as fronteiras das funções departamentais, conectando todos os departamentos relacionados a um determinado processo. Além disso, pode ser interorganizacional, estabelecendo conexões não apenas entre departamentos, mas também com elementos externos à organização.

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.

A perspectiva por processos, a estrutura organizacional passa por uma reconfiguração para priorizar os processos como um eixo gerencial mais significativo do que o eixo funcional. Todas as decisões, assim como os sistemas de informação, avaliação de desempenho, alocação de recursos financeiros, requisitos dos clientes e outros fatores são avaliados de maneira integrada, ou seja, de forma global (WERLICH, 2020).

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.

Vamos Exercitar?

A situação-problema desta aula 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

ANÁLISE E MODELAGEM DE SISTEMAS

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.
- Realizar a criação do organograma funcional da empresa, colocando no topo a gestora da empresa.

ANÁLISE E MODELAGEM DE SISTEMAS

- 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 máquinas e equipamentos são necessários.
- Quais pessoas são necessárias.
- Quais recursos financeiros são necessários.
- Qual a infraestrutura necessária.
- 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!

Saiba mais

Para conhecer mais sobre a análise de modelagem de processos de negócios, acesse o livro *Análise e modelagem de processos de negócio: foco na notação BPMN*, disponível na Biblioteca Virtual.

VALLE, R.; OLIVEIRA, S. B. de. [Análise e modelagem de processos de negócio: foco na notação BPMN \(Business Process Modeling Notation\)](#). Grupo GEN, 2013.

Para saber mais sobre como funciona a Gestão por Processos, leia o artigo *Como funciona a Gestão por Processos? Entenda*.

SYDLE. [Como funciona a Gestão por Processos? Entenda](#). 2023.

ANÁLISE E MODELAGEM DE SISTEMAS

Para conhecer mais sobre o Gerenciamento de processos de negócios, acesse o **Guia BPM CBOK – Capítulo 2**, disponível na Biblioteca do Portal do Escritório de Processos do Instituto Federal de São Paulo:

BPM CBOK. Guia para o Gerenciamento de Processos de Negócio. Corpo Comum de Conhecimento. ABPMP BPM CBOK V3.0. 1a. Edição. 2013.

Referências

BROCKE, J. V.; ROSEMANN, M. **Manual de BPM**: gestão de processos de negócio. Porto Alegre: Bookman, 2013.

CHIAVENATO, I. **Administração**: teoria, processo e prática. 5. ed. Barueri: Manole, 2014.

DE SORDI, J. O. **Gestão de Processos**: uma abordagem da moderna administração. 5. ed. São Paulo: Saraiva, 2018.

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

WERLICH, C. **Análise e modelagem de sistemas**. Londrina: Editora e Distribuidora Educacional S.A., 2020.

Aula 2

Gerenciamento de Processos de Negócios

Gerenciamento de processos de negócios



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Olá estudante! Nesta videoaula, você irá conhecer o Gerenciamento de Processos de Negócio (BPM). Exploraremos desde a introdução aos fundamentos do BPM até a compreensão prática

ANÁLISE E MODELAGEM DE SISTEMAS

das BPMS (*Business Process Management Suite or System*). Além disso, ofereceremos uma visão abrangente das áreas de negócio, destacando a relevância desses conhecimentos para sua prática profissional. Prepare-se para mergulhar em conceitos essenciais que impulsionarão sua eficiência no universo empresarial!

[Clique aqui](#) para acessar os slides da sua videoaula.

Bons estudos!

Ponto de Partida

Caro estudante, nesta aula, 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.

Sobre esses assuntos, vamos imaginar uma empresa de logística, que atua na gestão logística de transporte de cargas. Com o crescimento do volume de negócios, a empresa enfrenta desafios na eficiência operacional e na integração de sistemas legados. Para superar esses obstáculos, a empresa decide implementar uma abordagem baseada em Arquitetura Orientada a Serviços (SOA) e Gerenciamento de Processos de Negócio (BPM).

Desafios:

- **Desconexão de sistemas:** a empresa possui sistemas legados que operam de forma isolada, dificultando a troca de informações entre departamentos.
- **Ineficiências operacionais:** os processos de gerenciamento de carga e rastreamento são complexos e enfrentam atrasos devido à falta de automação e padronização.
- **Falta de visibilidade:** a alta administração carece de visibilidade em tempo real sobre o desempenho operacional e os principais indicadores de desempenho (KPIs).

Encontre possíveis soluções para o desafio que a empresa enfrenta.

Vamos Começar!

Introdução ao Gerenciamento de Processos de Negócio (*Business Process Management - BPM*)

ANÁLISE E MODELAGEM DE SISTEMAS

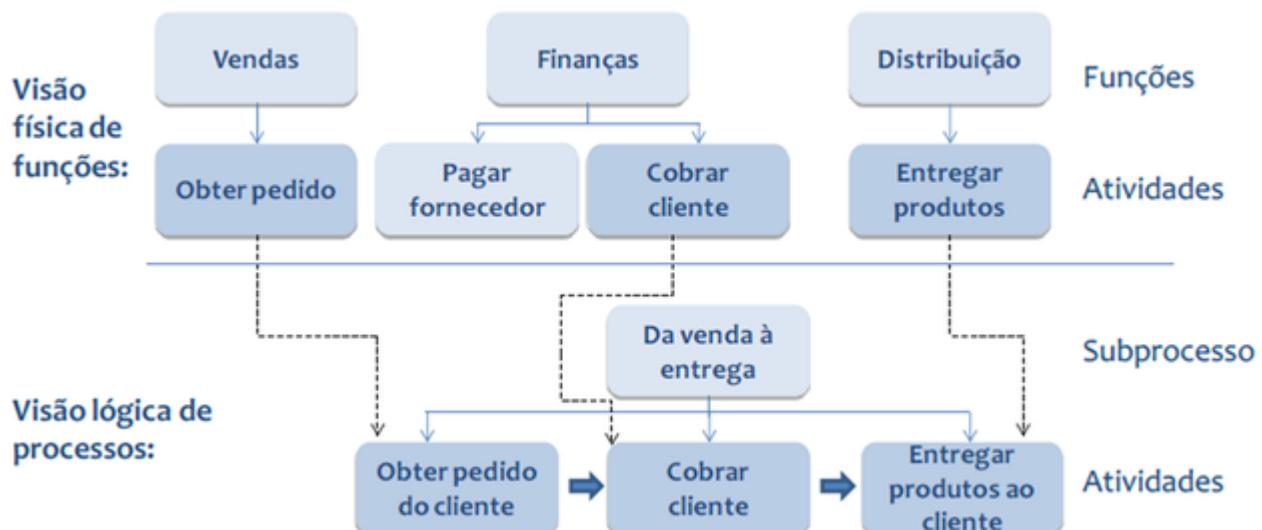
O gerenciamento representa uma atividade fundamental para todas as organizações e em todas as suas áreas, uma vez que possibilita a identificação de lacunas nos resultados e, a partir disso, a elaboração de planos de aprimoramento. Chiavenato (2014) destaca quatro funções administrativas essenciais: Planejamento, Organização, Direção e Controle. Segundo o autor, a última função está intrinsecamente ligada à gestão, e ele argumenta que não há uma organização com desempenho satisfatório sem a presença efetiva dessas quatro funções.

Chiavenato (2014) sublinha que controlar, monitorar ou gerenciar são termos intercambiáveis que desempenham uma função crucial, permitindo à organização entender precisamente o que está sendo realizado, como está sendo feito e quais são as dificuldades enfrentadas. Portanto, trata-se de uma atividade de extrema relevância para o funcionamento eficiente e eficaz da organização.

No BPM, que significa *Business Process Management* (Gerenciamento de Processos de Negócio), a abordagem aos processos organizacionais é distinta, pois a perspectiva é expandida. Existe uma compreensão abrangente de toda a cadeia envolvida na entrega de um produto ou serviço, diferenciando-se de uma visão verticalizada.

A perspectiva verticalizada limita a percepção da organização a uma visão de cima para baixo, onde as interações entre as áreas não são facilmente identificadas. O gerenciamento de processos de negócio assume relevância ao promover a integração entre as áreas, facilitando a comunicação, a troca de informações e o fortalecimento de uma visão horizontal.

De acordo com a ABPMP (2013), o BPM torna-se essencial para analisar o processo em um nível mais elevado do que a simples execução de tarefas. Somente ao enxergar integralmente o processo torna-se possível compreender as relações entre as áreas e subdividi-lo em subprocessos, os quais serão executados por diversas atividades (fluxos de trabalho) nas áreas funcionais. A Figura 1 ilustra a diferença entre a compreensão física das funções, representadas pelas atividades, e a visão lógica, representada pelos processos.



ANÁLISE E MODELAGEM DE SISTEMAS

Figura 1 | Analogia entre visão física e lógica dos processos. Fonte: ABPMP (2013, p. 34).

Observe que na Figura 1, a visão física retrata as áreas operando de maneira desintegrada, onde cada uma realiza suas atividades como se fossem isoladas, sem interdependência aparente. Por outro lado, na visão lógica, o foco está no processo, evidenciando a interconexão entre as atividades, uma vez que o resultado é influenciado por todas elas.

Nesse contexto, comprehende-se o BPM como a construção de processos ponta a ponta, viabilizando a integração das estratégias e objetivos das empresas. Para que essa abordagem funcione de maneira eficaz, é crucial compreender elementos como cultura organizacional, clima, estruturas, tecnologia e políticas. A governança dos processos, em que todas as ações seguem regras e diretrizes com o objetivo de alcançar os objetivos organizacionais, é essencial. A governança deve incorporar os conceitos de controle e prestação de contas, o que se mostra apropriado e extremamente necessário para a gestão de processos (ARAÚJO; GARCIA; MARTINES, 2017).

Os processos de negócio estão embasados em algumas metodologias de mudanças, conforme apresentado na Figura 2.

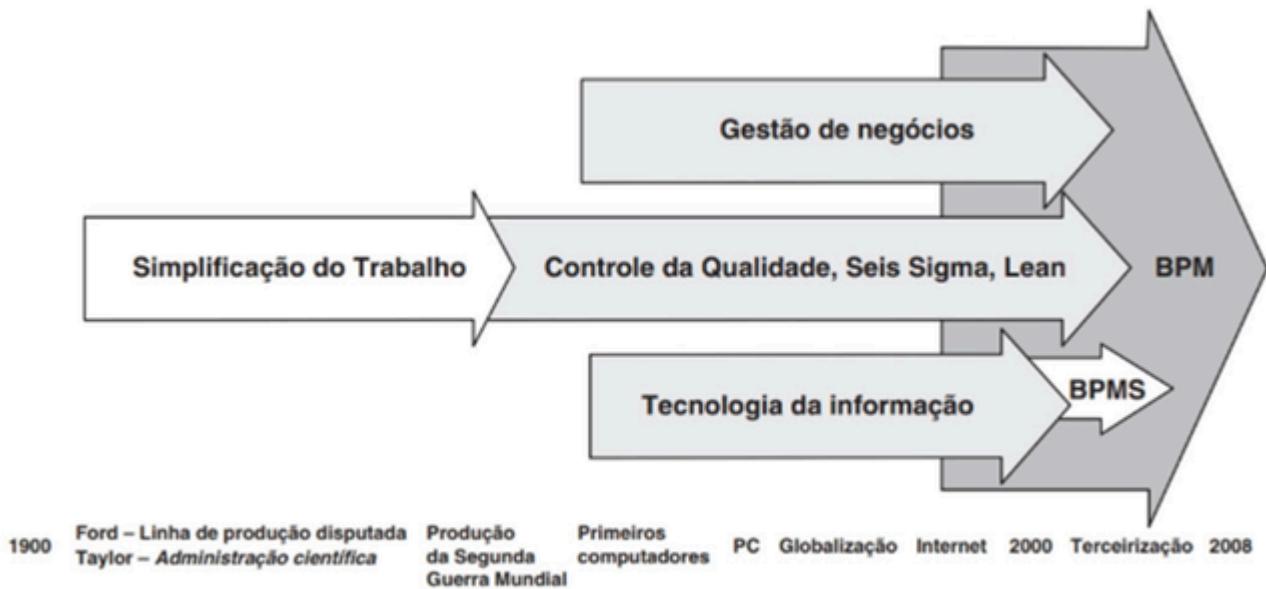


Figura 2 | Abordagens de mudanças dos processos de negócio. Fonte: Brocke e Rosemann (2013, p. 38).

A evolução da melhoria de processos, ilustrada na Figura 2, tem seu início com uma abordagem de simplificação do trabalho, progredindo para os controles de qualidade, metodologias como

ANÁLISE E MODELAGEM DE SISTEMAS

Seis Sigma e Lean, chegando até a gestão de negócios e tecnologia da informação. A convergência de todos esses elementos resulta no Gerenciamento de Processos de Negócio (BPM), e associado a ele está o BPMS (*Business Process Management Suite or System*), atuando como suporte tecnológico. O BPMS, traduzido como sistema de gerenciamento de processos de negócio, é um software que viabiliza o mapeamento, execução e monitoramento dos processos organizacionais.

BPMS (*Business Process Management Suite or System*)

O BPMS é uma ferramenta que desempenha a função de mapear, executar e monitorar os processos funcionais, buscando proporcionar uma visão abrangente de ponta a ponta. Em outras palavras, contribui para a automatização das ações e do fluxo de informações dentro dos processos. Conforme destacado por Araújo, Garcia e Martines (2017), o BPMS é considerado uma evolução em relação ao workflow, uma vez que tem a capacidade de integrar diversos fluxos de trabalho. Essa característica confere ao BPMS uma perspectiva mais ampla e a habilidade de se integrar a sistemas legados, ou seja, sistemas mais antigos que ainda estão em operação.

Segundo Valle e Oliveira (2013), é crucial que os processos de negócio estejam em sintonia com os objetivos e estratégias da organização. A Figura 3 ilustra a maneira pela qual esse alinhamento deve ser estabelecido na perspectiva do Gerenciamento de Processos de Negócio (BPM).

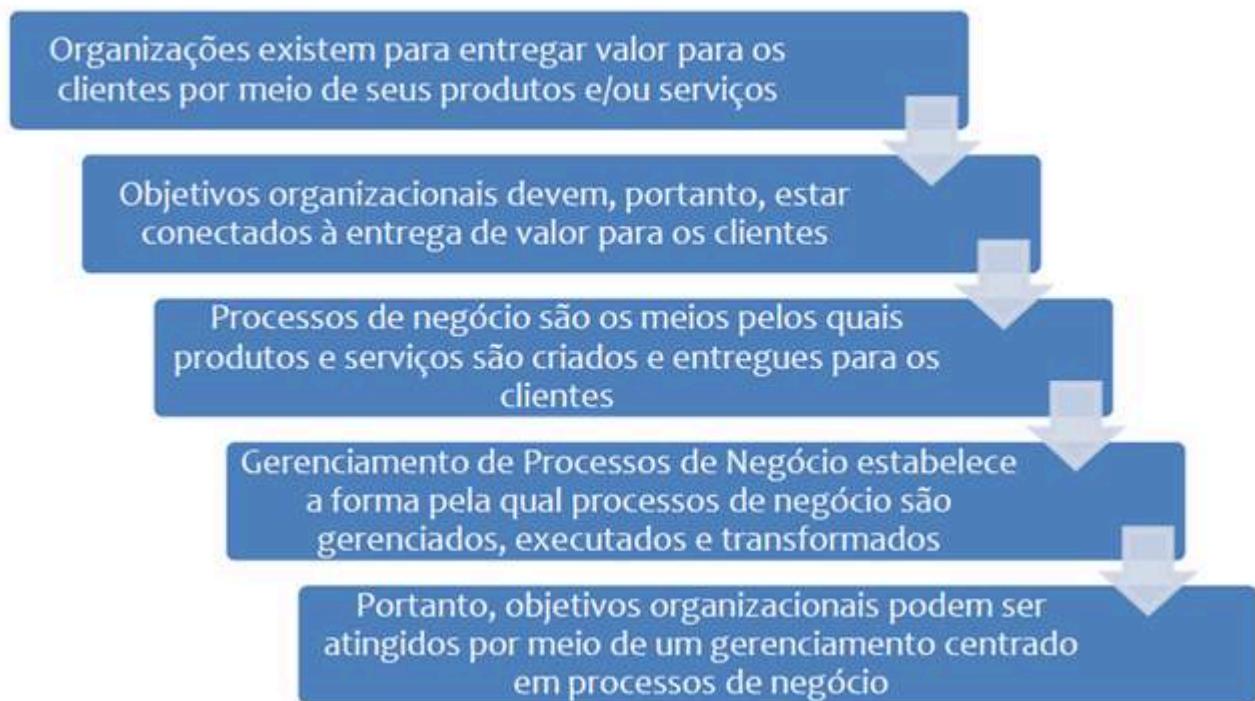


Figura 3 | BPM e a conexão com os objetivos estratégicos. Fonte: ABPMP (2013, p. 46).

ANÁLISE E MODELAGEM DE SISTEMAS

Na Figura 3, o desdobramento se inicia com a entrega de valor ao cliente, onde esse valor deve ser percebido por ele através de produtos/serviços. Esses produtos/serviços, por sua vez, necessitam estar alinhados aos objetivos globais da organização, ou seja, toda a empresa deve colaborar em direção a esses objetivos. Esses objetivos, por sua vez, são detalhados nos processos e em seu gerenciamento, orientados para a otimização da forma como as necessidades do cliente são atendidas, refletindo assim nos objetivos gerais da organização.

Siga em Frente...

Visão geral sobre as áreas de negócio

A visão de gestão de negócios está intrinsecamente relacionada à função administrativa, como indicado por Chiavenato (2014). Este autor destaca a importância de estabelecer padrões de desempenho mensuráveis, passíveis de comparação com os resultados reais por meio de monitoramento. Esse acompanhamento permite a tomada de medidas corretivas, se necessário, para alcançar os objetivos propostos. Chiavenato também enfatiza que o controle deve abranger todos os níveis organizacionais, dividindo-se em controles estratégicos, táticos e operacionais. Os controles estratégicos oferecem uma visão abrangente de longo prazo para a organização como um todo. Já os controles táticos são mais detalhados, de médio prazo, focando em perspectivas departamentais. Por fim, os controles operacionais são analíticos, de curto prazo, direcionados para tarefas e atividades específicas. Relacionando essa perspectiva com o conceito de BPM, percebe-se que o processo de controle também deve ocorrer em todos os níveis, mas, no contexto do BPM, o foco reside nas atividades e tarefas que compõem cada processo de negócio.

O planejamento do gerenciamento de processos de negócios está intrinsecamente ligado à compreensão do nível de maturidade em processos da organização. Isso porque a capacidade da empresa de compreender e gerenciar seus processos determinará a abordagem adotada no gerenciamento. A maturidade engloba não apenas a habilidade da empresa em compreender seus processos, mas também como ela interage com eles. O objetivo é implementar uma série de atividades que auxiliem a organização a alcançar metas predefinidas, contribuindo para a melhoria dos resultados na área de TI. Portanto, o CMMI (*Capability Maturity Model Integration*, ou Modelo Integrado de Capacidade de Maturidade em português) desempenha um papel crucial no contexto do BPM, pois colabora para um gerenciamento mais eficiente das atividades, resultando em produtos finais padronizados, com menor margem de erro e, consequentemente, 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.

ANÁLISE E MODELAGEM DE SISTEMAS

Conforme indicado pelo SEI (2010), com o propósito de consolidar todos os modelos de capacitação existentes, o conceito evoluiu para CMMI, que representa o Modelo de Maturidade da Capacitação Integrado. A transição de CMM para CMMI ocorreu entre 1999 e 2002, com o lançamento da versão 1.0 do CMMI em 2000, seguida pela versão 1.1 em 2002. A versão 1.2 foi iniciada em 2006, e a versão 1.3 foi lançada em 2010.

Segundo Couto (2007), a versão inicial, denominada 0.2 do CMMI, tinha como objetivo aprimorar os processos e produtos, reduzindo problemas de redundância ou falhas decorrentes da utilização de vários modelos distintos. As subsequentes versões do CMMI foram desenvolvidas para aperfeiçoar o modelo ao longo do tempo.

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 4, quanto maior o nível de maturidade melhor será a capacidade de gerenciamento do processo.

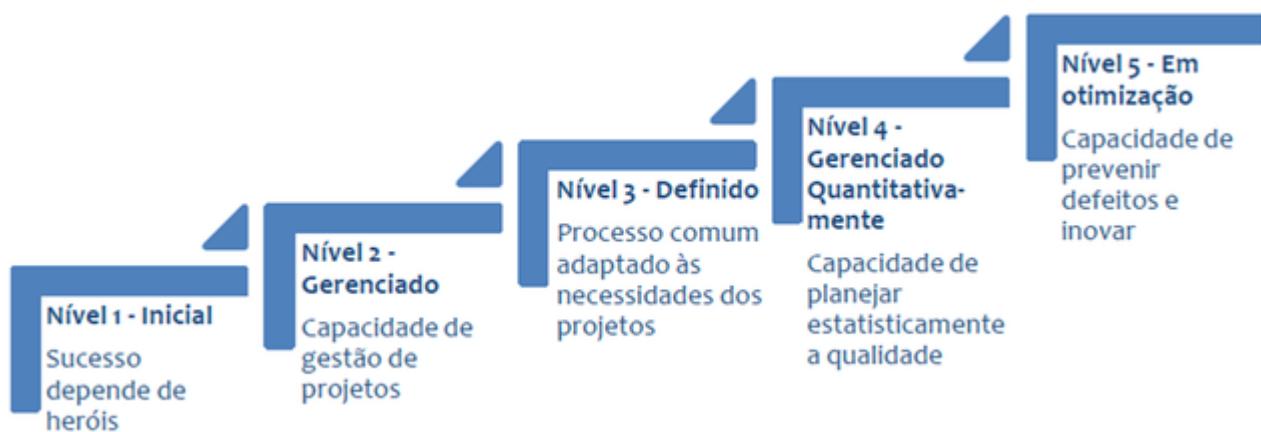


Figura 4: Níveis de maturidade CMMI. Fonte: ABPMP (2013, p. 218).

A Figura 4 ilustra os diferentes níveis de maturidade do CMMI. No primeiro nível, destaca-se que o sucesso depende de indivíduos excepcionais, pois não há padrões estabelecidos. No segundo nível, observa-se a implementação de planejamento, medição e controle dos processos. No

ANÁLISE E MODELAGEM DE SISTEMAS

terceiro nível, os processos são definidos e compreendidos pela empresa, com procedimentos padrão estabelecidos e a capacidade de serem aplicados de maneira previsível em outros projetos. No quarto nível, com o controle quantitativo, há a possibilidade de prever o desempenho. Por fim, no quinto nível, o foco concentra-se na melhoria contínua dos processos.

Até agora, abordamos principalmente métodos, técnicas, ferramentas e softwares; no entanto, para otimizar ao máximo os resultados dessas variáveis, é essencial considerar o papel das pessoas no processo de gerenciamento de processos de negócio. Um dos atores mais cruciais nesse contexto é o "*process owner*," que pode ser entendido como o proprietário do processo de negócios. De acordo com De Sordi (2018), os donos dos processos devem personificar o comprometimento que a organização tem com os processos de negócio. O autor destaca que poucos indivíduos na organização desempenham esse papel, uma vez que não se trata de um gestor funcional, mas sim de um proprietário do processo de negócio. Normalmente, esse proprietário é um gerente que mantém essa responsabilidade de forma contínua sobre o processo, ao contrário de um gerente de projetos, cuja responsabilidade se encerra em um determinado momento.

O gestor de processos desempenha algumas atribuições essenciais, sendo responsável por assegurar que todos os recursos necessários estejam disponíveis ao longo do processo de negócio. Além disso, ele realiza uma avaliação contínua dos resultados do processo, facilita o treinamento adequado de toda a equipe e, em colaboração com essa equipe, analisa, redefine e implementa as mudanças necessárias no processo (DE SORDI, 2018).

Embora o gestor de processos tenha diversas atribuições, é crucial destacar que as ações não são exclusivamente de sua responsabilidade. A perspectiva dos profissionais envolvidos no processo desempenhará um papel significativo na avaliação e implementação de mudanças que se tornem necessárias ao processo. Dessa forma, uma cultura participativa revela-se de grande relevância para o progresso eficaz do processo e para atender de maneira eficiente às demandas dos clientes.

As pessoas e a cultura organizacional exercem uma influência significativa na execução das ações requeridas para promover mudanças nas empresas. Portanto, é crucial que a cultura atue como um impulsionador de mudanças, em vez de promover a estagnação. O êxito dessas ações de mudança requer uma variedade de habilidades e competências, algumas associadas à liderança e outras à equipe. Sem dúvida, o líder desempenha um papel fundamental nesse processo (BROCKE E ROSEMANN, 2013).

Atualmente, a gestão de mudança é um tópico amplamente debatido nas organizações, sendo considerado um dos grandes desafios da administração contemporânea, conforme destacado por vários autores, como Bowditch e Buono (2017). Eles salientam que a mudança, por sua natureza, gera desconforto e impactos negativos nos processos de gestão organizacional.

Diante dessa perspectiva, é possível estabelecer uma conexão entre a mudança e o BPM, uma vez que o BPM requer uma disposição significativa para a mudança, pois seu objetivo é

ANÁLISE E MODELAGEM DE SISTEMAS

direcionar-se para a melhoria contínua dos processos de negócio. Essa característica precisa ser incorporada por todos os colaboradores da organização, facilitando assim a implementação de mudanças de maneira mais fluida.

No entanto, apesar da importância da mudança para o BPM, alterar padrões estabelecidos não é tão simples quanto pode parecer. Diversos fatores contribuem para a resistência e, por vezes, sabotagem às mudanças organizacionais. Elementos como o medo do desconhecido, a preferência pela zona de conforto, crenças arraigadas, pensamento divergente e até mesmo questões pessoais, como antipatia em relação à pessoa que propôs a ideia, podem prejudicar o processo de mudança.

Segundo Bowditch e Buono (2017), existem duas abordagens para a mudança cultural: a primeira consiste em persuadir as pessoas a "abraçarem a ideia", ou seja, compartilharem a mesma visão e perspectiva da situação. A segunda envolve a contratação de novos membros para compor a equipe, contudo, essa abordagem requer a incorporação de valores alinhados com a cultura já existente na equipe, além do investimento na troca e, por vezes, no desenvolvimento do conhecimento técnico moldado à organização. A mudança cultural, de acordo com Bowditch e Buono (2017, p. 208), é fundamentada em cinco pontos-chave:

- Alterar o comportamento dos membros da organização.
- Justificar a necessidade da mudança de comportamento.
- Comunicar mensagens culturais relacionadas à mudança.
- Recrutar e integrar novos membros.
- Desligar membros que não se adaptam.

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) destacam a importância de o responsável por processos de negócio possuir habilidades como um bom relacionamento com as áreas funcionais, uma comunicação eficaz com todos os membros da equipe, proficiência no uso de recursos de TI, capacidade de gerenciar equipes multifuncionais, aptidão para harmonizar as diversas atividades do processo, relacionamento efetivo com as organizações envolvidas, compreensão abrangente do processo de negócio, domínio total da entrega principal do processo e conhecimento sólido em modelagem e avaliação de resultados. No que diz respeito à equipe de processos de negócio, é crucial que ela tenha a habilidade de identificar e compreender problemas, bem como desenvolver soluções para desafios identificados e aproveitar oportunidades que surjam durante o processo.

O gestor, juntamente com sua equipe, requer qualificações técnicas e funcionais, contudo, é igualmente essencial possuir comportamentos que contribuam para o alcance dos objetivos.

ANÁLISE E MODELAGEM DE SISTEMAS

Entre esses comportamentos, destacam-se: manter o foco em metas e atividades que agreguem valor ao negócio, demonstrar flexibilidade para se adaptar às mudanças necessárias nos processos, cultivar uma mentalidade colaborativa e a partilha de informações, estar aberto a aceitar contribuições e feedback de outros membros, e possuir habilidades para utilizar eficientemente os recursos tecnológicos relacionados ao trabalho.

Diversos são os recursos tecnológicos, representados por softwares, que podem contribuir para o Gerenciamento de Processos de Negócio (BPM). Todos esses softwares compartilham o objetivo final de fornecer relatórios que possibilitem a identificação de cenários e, consequentemente, a formulação de planos de melhoria de processo. Em essência, eles devem apresentar dashboards, ou seja, painéis de desempenho, para os interessados. Importa salientar que existem softwares de BPM variando em complexidade, desde soluções simples até aquelas extremamente sofisticadas. Exemplos de softwares BPM incluem o Casewise, Aris Platform, WebSphere Business Modeler, Aqualogic BPM Studio, Visio, Bizagi Modeler, Bonita Open Solution e Comindware. Novas ferramentas frequentemente surgem para aprimorar a modelagem e o gerenciamento de processos. Um sistema BPMS, por sua vez, tem como finalidade modelar processos e fluxos de trabalho, estabelecer regras, simular operações de negócios, automatizar processos, bem como monitorar, controlar e acompanhar o desempenho das atividades.

Frequentemente, é essencial criar interfaces para garantir a disponibilidade dos dados necessários, mantendo-se em conformidade com as regras estabelecidas. Em organizações que adotam a Arquitetura Orientada a Serviços (SOA), o processo pode ser simplificado através do uso de adaptadores, que facilitam a definição da integração, e sistemas que facilitam a troca de dados entre as aplicações. De maneira geral, os processos de negócios são delineados pela notação BPMN quando se emprega um modelo de Sistema de Gerenciamento de Processos de Negócio (BPMS).

ANÁLISE E MODELAGEM DE SISTEMAS

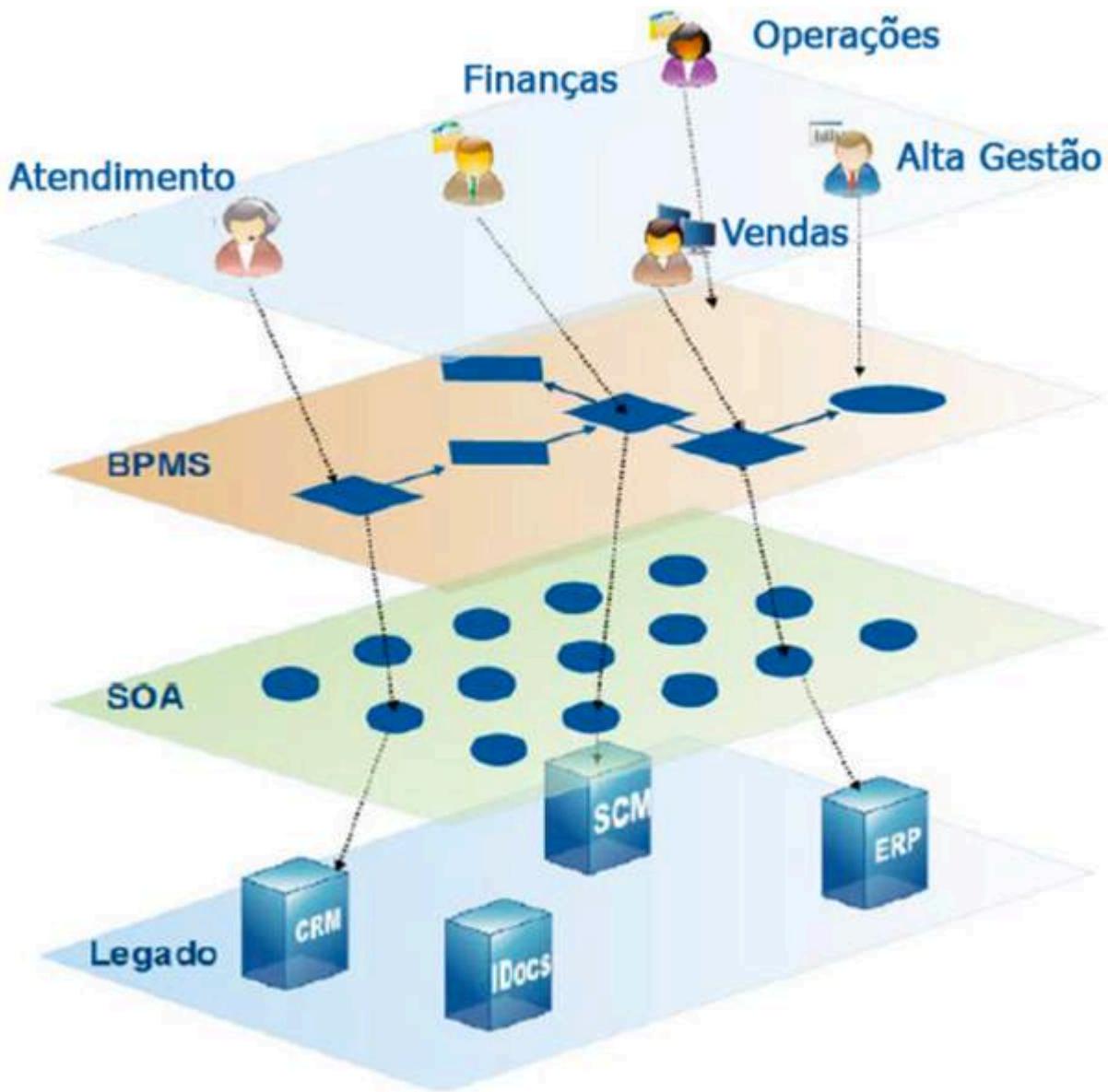


Figura 5 | Camada SOA para interagir com as funcionalidades legadas. Fonte: ABPMP (2013, p. 375).

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 (DE SORDI, 2018). 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.

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

ANÁLISE E MODELAGEM DE SISTEMAS

(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.

Vamos Exercitar?

A situação-problema proposta nesta aula traz o cenário da empresa de logística que decide adotar uma abordagem integrada de SOA e BPM para superar os desafios identificados.

Vamos à solução!

- **Desenvolvimento de serviços:** identificação e design de serviços para interligar os sistemas legados. Isso inclui serviços para rastreamento de carga, gerenciamento de estoque e comunicação entre diferentes áreas.
- **Automação de processos:** utilização de ferramentas BPM para modelar, otimizar e automatizar os processos de negócios. Isso abrange desde a solicitação de transporte até a entrega final, incorporando etapas de aprovação, verificação e escalonamento.
- **Implementação de APIs:** desenvolvimento de APIs para facilitar a comunicação entre os diversos sistemas internos e externos, promovendo a interoperabilidade.
- **Dashboard de KPIs:** criação de painéis de controle baseados em BPM, proporcionando uma visão unificada dos principais indicadores de desempenho, como tempo de entrega, eficiência operacional e satisfação do cliente.

Resultados esperados:

- **Melhoria na eficiência operacional:** redução de atrasos e erros nos processos logísticos, resultando em uma operação mais eficiente.
- **Integração de sistemas:** facilitação da troca de dados entre sistemas legados, eliminando silos de informação.
- **Visibilidade em tempo real:** a alta administração terá acesso a dashboards em tempo real, permitindo uma tomada de decisão mais informada.

ANÁLISE E MODELAGEM DE SISTEMAS

- **Adaptabilidade a mudanças:** a arquitetura SOA permite que a empresa se adapte mais facilmente a futuras mudanças, adicionando ou modificando serviços conforme necessário.
- **Avaliação:** ao final da implementação, a empresa avaliará os resultados com base na eficiência operacional, na satisfação do cliente e na capacidade de adaptação a novas demandas de mercado. O estudo de caso servirá como base para compreender o impacto positivo da integração de SOA e BPM no ambiente empresarial da empresa.

Saiba mais

Quer conhecer mais sobre a Arquitetura Orientada a Serviços?! Acesso o site indicado e conheça mais sobre este tema: *O que é arquitetura orientada a serviços?*

AWS. [O que é SOA \(arquitetura orientada a serviços\)?](#). [s. d.]

Para ler mais sobre BPMS acesse o seguinte artigo: *O que significa BPMS? Quais são seus componentes?*

DINIZ, B. [O que significa BPMS? Quais são seus componentes?](#). 2023.

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.

BOWDITCH, J. L.; BUONO, A. F. **Elementos do Comportamento Organizacional**. São Paulo: Cengage Learning, 2017.

BROCKE, J. V.; ROSEMANN, M. **Manual de BPM**: gestão de processos de negócio. Porto Alegre: Bookman, 2013.

CHIAVENATO, I. **Administração**: teoria, processo e prática. 5. ed. Barueri: Manole, 2014.

COUTO, A. B. **CMMI. Integração dos Modelos de Capacitação e Maturidade de Sistemas**. 1^a ed. Ciência Moderna. 2007.

ANÁLISE E MODELAGEM DE SISTEMAS

DE SORDI, J. O. **Gestão de Processos**: uma abordagem da moderna administração. 5. ed. São Paulo: Saraiva, 2018.

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

Aula 3

Modelagem de Processos de Negócios

Modelagem de processos de negócios

Este conteúdo é um vídeo!



Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Olá, estudante! Nesta videoaula, você conhecerá o universo da Modelagem de Processos de Negócios. Explore a introdução a essa prática, compreenda tanto a Arquitetura de Processos como a de Negócios, e aprofunde-se nos tipos de Notações de Modelagem de Negócios. Estes tópicos são essenciais para aprimorar sua prática profissional, capacitando-o a visualizar, analisar e otimizar eficientemente os processos empresariais. Esteja preparado para aprimorar suas habilidades e impulsionar sua eficácia no contexto corporativo.

[Clique aqui](#) para acessar os slides da sua videoaula.

Bons estudos!

Ponto de Partida

Engaje-se na emocionante jornada da modelagem de processos! Desenvolver esse conjunto de habilidades e técnicas não apenas permite compreender, mas também comunicar e gerenciar os elementos essenciais dos processos de negócio. Para organizações que reconhecem o valor significativo de seus processos, a modelagem torna-se uma peça central no quebra-cabeça do gerenciamento organizacional!

Durante esta aula, iremos explorar uma variedade de tópicos fundamentais relacionados à gestão eficiente de processos de negócios. Inicialmente, adentraremos o universo da Introdução

ANÁLISE E MODELAGEM DE SISTEMAS

à Modelagem de Processos de Negócios, proporcionando uma compreensão sólida dos conceitos-chave e das práticas essenciais nesse campo. Em seguida, exploraremos a Arquitetura de Processos e Arquitetura de Negócios, analisando como essas estruturas desempenham papéis cruciais na otimização e alinhamento dos processos organizacionais. Finalmente, dedicaremos tempo à exploração das Notações de Modelagem de Negócios, compreendendo os tipos de linguagens visuais para representar e comunicar eficientemente os processos de negócios. Este conjunto abrangente de temas busca fornecer uma base sólida para a compreensão e aplicação prática da gestão de processos, capacitando os participantes a navegar com sucesso no complexo ambiente empresarial.

Existem diversas linguagens de modelagens de processo, e a escolha dessas linguagens faz diferença. Por isso, como exercício prático, faça uma lista de algumas linguagens de modelagem e coloque as vantagens e desvantagens de cada linguagem.

Vamos Começar!

Introdução à modelagem de processos de negócios

A atividade de modelagem de processos requer um conjunto essencial de habilidades e técnicas, possibilitando compreender, comunicar e gerenciar os componentes dos processos de negócio. Para organizações conscientes do valor significativo de seus processos de negócio, a modelagem torna-se uma atividade central no gerenciamento organizacional.

A modelagem de processos de negócio abrange o conjunto de atividades destinadas a criar representações de processos existentes ou propostos. Essas representações podem oferecer uma visão abrangente do processo ou focalizar em partes específicas, como processos primários, de suporte ou de gerenciamento.

O objetivo da modelagem é desenvolver uma representação completa e precisa do processo, destacando seu funcionamento. O nível de detalhamento e o tipo de modelo adotados dependem das expectativas da iniciativa de modelagem. Em alguns casos, um diagrama simples pode ser suficiente, enquanto em outros, pode ser necessário um modelo mais abrangente e detalhado (ABPMP, 2013).

Um modelo é uma representação simplificada de um objeto, conceito ou atividade, podendo adotar formas matemáticas, gráficas, físicas, narrativas ou combinações desses tipos. Os modelos têm uma ampla variedade de aplicações nos contextos de negócios, desempenhando funções como (VALLE, R.; OLIVEIRA, 2013):

- Organização (estruturação)
- Descoberta (aprendizagem)
- Previsão (estimativas)

ANÁLISE E MODELAGEM DE SISTEMAS

- Medição (quantificação)
- Explicação (ensino, demonstração)
- Verificação (validação)
- Controle (estabelecimento de restrições e objetivos)

Os processos de negócio podem ser expressos por meio de modelagem em diferentes níveis de detalhe, desde uma visão abstrata contextual até uma visão minuciosa. Um modelo de processos de negócio completo geralmente abrange diversas perspectivas, atendendo a diferentes propósitos.

O conteúdo de um modelo de processo compreende diversos elementos que visam representar de maneira abrangente e detalhada todas as facetas de um determinado processo de negócio. Esses elementos são essenciais para proporcionar uma compreensão completa e eficaz do funcionamento do processo. Algumas das principais componentes presentes no conteúdo de um modelo de processo incluem (BROCKE, 2013):

1. **Atividades:** descrição das tarefas específicas ou operações realizadas no decorrer do processo. As atividades destacam as etapas cruciais para atingir os objetivos do processo.
2. **Fluxo de trabalho:** representação visual das sequências e interconexões entre as atividades do processo. O fluxo de trabalho mostra a ordem lógica das etapas e como a informação ou trabalho flui de uma atividade para outra.
3. **Recursos:** identificação dos elementos necessários para a execução das atividades, como pessoas, sistemas, dados ou equipamentos. Esses recursos são fundamentais para o desempenho eficiente do processo.
4. **Decisões:** inclusão de pontos de decisão ou ramificações no processo, indicando escolhas que influenciam o curso do fluxo de trabalho. Isso ajuda a modelar as condições para determinadas ações.
5. **Eventos:** indicação de eventos ou gatilhos que podem iniciar, interromper ou modificar o processo. Os eventos destacam momentos-chave que impactam o comportamento do processo.
6. **Tempo:** consideração dos aspectos temporais, como prazos, tempos de ciclo e quaisquer restrições temporais relevantes para as atividades do processo.
7. **Responsabilidades:** atribuição de responsabilidades específicas para as atividades, indicando quem é responsável por executar cada tarefa no processo.
8. **Objetivos:** clarificação dos objetivos gerais do processo, garantindo que todas as atividades estejam alinhadas com as metas organizacionais.

Ao integrar esses elementos, um modelo de processo robusto fornece uma representação abrangente que facilita a compreensão, análise, otimização e gestão eficaz dos processos de negócio dentro de uma organização. Vale ressaltar que nem todos os elementos citados anteriormente podem ser integrados a qualquer modelo, isso dependerá se faz parte dos elementos determinados pelo modelo.

ANÁLISE E MODELAGEM DE SISTEMAS

Os conceitos de diagrama de processo, mapa de processo e modelo de processos frequentemente são empregados de maneira intercambiável ou como termos sinônimos. No entanto, diagramas, mapas e modelos possuem distintos propósitos e aplicações. Na prática, diagrama, mapa e modelo representam diferentes estágios de desenvolvimento, cada um incorporando maior quantidade de informações e utilidade para o entendimento, análise e delineamento de processos. A Figura 1 apresenta essa diferença de modo visual.

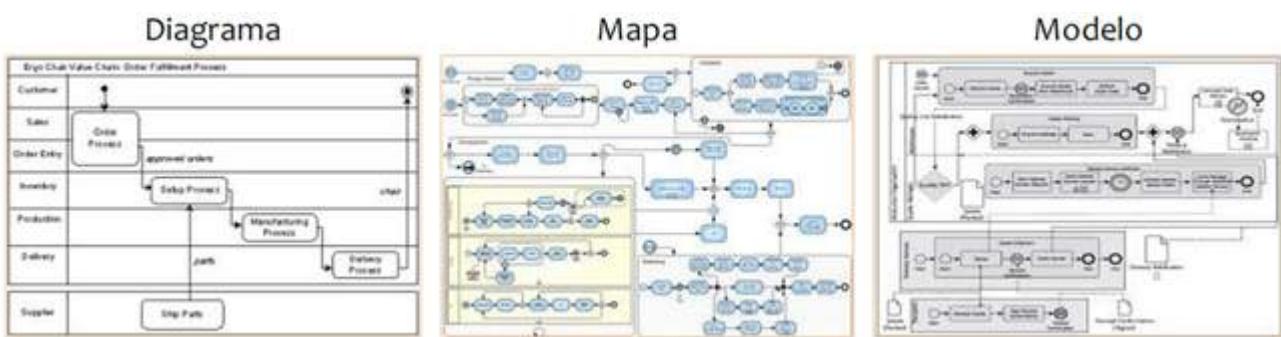


Figura 1 | Diagrama versus mapa versus modelo. Fonte: ABPMP (2013, p. 73).

Um diagrama apresenta os elementos principais de um fluxo de processo, simplificando detalhes menores para facilitar a compreensão dos fluxos de trabalho. Pode ser comparado a um mapa básico que ajuda a localizar um armazém, fornecendo uma visão simplificada ou exagerada de marcos geográficos e distâncias. Da mesma forma, um diagrama de processo auxilia na rápida identificação e compreensão das atividades principais do processo.

Um mapa oferece uma visão abrangente dos principais componentes do processo, acrescentando mais precisão do que um diagrama. Tende a incorporar maior detalhe sobre o processo e alguns relacionamentos cruciais com outros elementos, como atores, eventos e resultados.

Um modelo implica a representação de um estado específico do negócio (atual ou futuro) e dos recursos correspondentes, como pessoas, informações, instalações, automação, finanças e insumos. Por buscar uma representação mais precisa do funcionamento do que está sendo modelado, exige mais dados sobre o processo e os fatores que afetam seu comportamento.

Frequentemente, a modelagem é realizada por meio de ferramentas que oferecem capacidades de simulação e geração de relatórios úteis para analisar e compreender o processo. Ao examinar uma "ilustração" de negócio, a tabela a seguir pode ser útil para distinguir entre diagrama, mapa e modelo de processos.

Siga em Frente...

Arquitetura de processos e Arquitetura de negócios

ANÁLISE E MODELAGEM DE SISTEMAS

A distinção entre arquitetura de processos e arquitetura de negócio é frequentemente objeto de dúvida para aqueles envolvidos em modelagem. Arquitetos de negócio concentram-se em criar modelos de negócio, que operam em um nível elevado de abstração e tratam das capacidades de negócio - ou seja, a habilidade de realizar ou entregar algo. Esses modelos de arquitetura de negócio são, portanto, conceituais e lidam com o "O QUE" no negócio (ABPMP, 2013).

Por outro lado, modelos de arquitetura de processos abordam o "COMO" do negócio, definindo como um entregável, produto ou serviço é construído e entregue. Quando decompostos em maior detalhe, esses modelos de arquitetura de processos definem as atividades que uma empresa deve ser capaz de realizar. Cada atividade está transitivamente relacionada a uma determinada capacidade.

Os modelos de arquitetura de negócio relacionam as atividades necessárias e estão focados na eficácia, enquanto os modelos de arquitetura de processos concentram-se nas atividades físicas e na gestão delas, preocupando-se com a eficiência. Quando combinados, esses modelos permitem ir além do desenho de atividades para garantir que nenhum trabalho seja executado se não estiver relacionado à entrega de uma capacidade de negócio específica, assegurando efetividade.

A confusão entre esses dois tipos de modelo muitas vezes ocorre devido ao fato de que, em muitas organizações, arquitetos de negócio constroem modelos de arquitetura de processos. Embora ambas as disciplinas visem melhorar o negócio, elas têm propósitos diferentes. Essas disciplinas são complementares, não iguais nem concorrentes, e ambas são essenciais em qualquer nível de mudança, seja no âmbito de processos ou corporativo. No entanto, em algumas organizações, essa distinção não é clara, e os papéis e ferramentas associadas a cada grupo podem ser confundidos (ABPMP, 2013).

Notações de modelagem de negócios

Notação é um conjunto padronizado de símbolos e regras que determinam o significado desses símbolos.

Por exemplo, assim como a notação musical emprega símbolos universalmente reconhecidos para notas e claves, a notação de modelagem de processos de negócio incorpora ícones (figuras) e conectores que facilitam a representação dos relacionamentos entre diversos componentes dos processos de negócio (ABPMP, 2013). Existem vários padrões de notação de modelagem, e escolher a abordagem mais adequada dentre as opções disponíveis pode ser desafiador. Contudo, optar por uma metodologia que siga normas e convenções bem estabelecidas oferece diversas vantagens, tais como a utilização de um conjunto comum de símbolos, linguagem e técnicas que facilitam a comunicação entre as pessoas, consistência no formato e significado dos modelos resultantes, além da capacidade de importar, exportar e gerar aplicações a partir desses modelos de processos. Serão apresentadas breves descrições de algumas das notações de modelagem mais comumente encontradas.

ANÁLISE E MODELAGEM DE SISTEMAS

O **BPMN (Business Process Model and Notation)** é uma notação gráfica padronizada internacionalmente, desenvolvida para representar visualmente os processos de negócios em organizações. Essa notação fornece um conjunto de símbolos e regras que permitem aos analistas e profissionais de BPM descrever, analisar e otimizar processos de forma clara e compreensível. O BPMN utiliza uma abordagem intuitiva e visual, representando as etapas do processo, as decisões, os eventos e as interações entre diferentes participantes. Essa notação facilita a comunicação entre as partes interessadas, promovendo uma compreensão consistente dos processos em toda a organização. O BPMN é amplamente adotado como uma ferramenta eficaz para modelagem de processos de negócios, contribuindo para a melhoria da eficiência operacional e para o alinhamento estratégico das atividades organizacionais (VALLE, R.; OLIVEIRA, 2013).

O **fluxograma** é uma representação visual que utiliza símbolos gráficos para descrever a sequência de atividades em um processo. Essa ferramenta é amplamente empregada na modelagem e documentação de processos, proporcionando uma visão clara e sistemática das etapas envolvidas. Os símbolos utilizados no fluxograma representam diferentes elementos, como operações, decisões, pontos de início e fim, facilitando a compreensão tanto para profissionais especializados quanto para aqueles que não têm conhecimento técnico aprofundado. O fluxograma é uma valiosa ferramenta na identificação de gargalos, ineficiências e oportunidades de melhoria em processos, contribuindo para uma gestão mais eficaz e uma comunicação eficiente entre os membros da equipe e partes interessadas.

O **Event-driven Process Chain (EPC)** é uma notação de modelagem de processos que oferece uma representação gráfica para descrever e analisar os processos de negócios em uma organização. Desenvolvido por August-Wilhelm Scheer, o EPC se destaca por sua abordagem centrada em eventos, onde os eventos desempenham um papel fundamental na definição das atividades e conexões no processo. Os eventos representam o início ou o término de uma atividade, sendo conectados por funções, processos e controles. Essa notação é eficaz na visualização das relações causais entre os eventos e nas condições sob as quais as atividades são executadas. Comumente utilizada em conjunto com metodologias de gestão de processos, o EPC é uma ferramenta valiosa para mapear, analisar e otimizar os processos organizacionais, proporcionando uma compreensão clara das interações e fluxos de trabalho.

A **Unified Modeling Language (UML)** é uma linguagem padronizada de modelagem visual amplamente utilizada na engenharia de software para representar sistemas complexos. Desenvolvida no final da década de 1990, a UML fornece uma notação gráfica abrangente que permite aos desenvolvedores, analistas e outros stakeholders visualizar, entender e comunicar eficientemente as diferentes partes de um sistema. A UML utiliza diversos diagramas, como diagramas de classes, diagramas de sequência, diagramas de casos de uso, entre outros, para representar aspectos específicos do sistema. Essa linguagem ajuda a capturar os requisitos, projetar a arquitetura, modelar a interação entre componentes e documentar o software ao longo do ciclo de vida do desenvolvimento. A UML tornou-se uma ferramenta essencial na engenharia de software, facilitando a comunicação entre equipes e promovendo uma compreensão mais clara e abrangente dos sistemas complexos.

ANÁLISE E MODELAGEM DE SISTEMAS

Vamos Exercitar?

Vamos à apresentação da lista das linguagens de modelagem de processos existentes, cada uma com suas características específicas. Aqui estão algumas delas, juntamente com suas vantagens e desvantagens:

1. BPMN (Business Process Model and Notation)

- **Vantagens:** padrão amplamente adotado, notação visual fácil de entender, suporte para diferentes níveis de detalhe, permite a modelagem de processos de ponta a ponta.
- **Desvantagens:** pode se tornar complexo para processos muito detalhados, limitado em representar certos aspectos organizacionais.

2. UML (Unified Modeling Language) Activity Diagrams

- **Vantagens:** amplamente conhecido na engenharia de software, integração com outros diagramas UML, suporte a diversas notações.
- **Desvantagens:** menos específico para processos de negócios do que BPMN, pode ser mais técnico.

3. EPC (Event-driven Process Chain)

- **Vantagens:** foco em eventos que acionam processos, representação clara de controle e dados, integração com sistemas de informação.
- **Desvantagens:** menos adotado globalmente do que BPMN, pode não ser tão intuitivo para não especialistas.

4. IDEF (Integrated DEFinition)

- **Vantagens:** pode ser aplicado em diversas disciplinas, incluindo processos de negócios, modelagem clara de dados e atividades.
- **Desvantagens:** menos difundido do que BPMN ou UML, curva de aprendizado pode ser íngreme.

5. CMMN (Case Management Model and Notation)

- **Vantagens:** Especializado em modelagem de casos, gestão adaptativa de processos, suporte a situações não estruturadas.
- **Desvantagens:** Menos utilizado em comparação com BPMN, pode ser mais complexo para cenários simples.

6. Flowchart

ANÁLISE E MODELAGEM DE SISTEMAS

- ***Vantagens:*** simplicidade, amplamente compreendido, adequado para processos simples.
- **Desvantagens:** menos poderoso para processos complexos, falta de padronização em comparação com BPMN.

Cada linguagem tem seu lugar dependendo das necessidades específicas do projeto e do público-alvo. A escolha deve considerar a clareza da representação, a familiaridade da equipe e a adequação ao contexto organizacional.

Saiba mais

Para conhecer mais sobre o mundo do BPMN, acesse o livro: *Análise e modelagem de processos de negócio: foco na notação BPMN*, disponível na Biblioteca Virtual.

]VALLE, R.; OLIVEIRA, S. B. de. [Análise e modelagem de processos de negócio: foco na notação BPMN \(Business Process Modeling Notation\)](#). Grupo GEN, 2013.

Existem ferramentas online para fazer a modelagem de negócios, o Visual Paradigma Online é uma dessas ferramentas.

VISUAL PARADIGMA ONLINE. [Ferramenta de Diagrama BPMN](#). [s. d.]

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.

BROCKE, J. V.; ROSEMANN, M. **Manual de BPM:** gestão de processos de negócio. Porto Alegre: Bookman, 2013.

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

Aula 4

Business Process Model and Notation (BPMN)

Business Process Model and Notation (BPMN)

ANÁLISE E MODELAGEM DE SISTEMAS



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Olá, estudante! Nesta videoaula, você conhecerá o universo do BPMN (*Business Process Model and Notation*). Exploraremos a introdução ao BPMN, seus elementos essenciais e forneceremos exemplos práticos. Estes conteúdos são fundamentais para sua prática profissional, capacitando-o a mapear e otimizar processos de negócios de maneira clara e eficiente. Prepare-se para dominar uma ferramenta essencial na gestão de processos e aprimorar suas habilidades no ambiente empresarial.

[Clique aqui](#) para acessar os slides da sua videoaula.

Bons estudos!

Ponto de Partida

Para esta aula, é fundamental compreender os conceitos relacionados à modelagem de processos de negócios, à notação BPMN (*Business Process Modeling Notation*) e aos seus elementos utilizados na construção de desenhos de processos de negócio, cadeias de valores, fluxos de trabalho e na documentação de processos de negócios.

É importante destacar que os elementos do BPMN representam uma evolução do fluxograma, proporcionando um desenho mais refinado do processo e gerando informações de maior qualidade para o acompanhamento das atividades cotidianas. O modelo de processo baseado na notação BPMN vai além de simplesmente indicar o "caminho" de uma atividade, incorporando informações e recursos detalhados ao processo, o que possibilita uma visão mais abrangente e minuciosa.

Para você treinar essa modelagem de processos, você deve criar um modelo de negócio para a especificação a seguir:

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 faltarem itens, é enviado um

ANÁLISE E MODELAGEM DE SISTEMAS

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.

Vamos Começar!

Introdução ao BPMN

A modelagem de processos requer a ampliação dos conceitos previamente apresentados. Esse campo envolve habilidades e técnicas essenciais para compreender e administrar os processos de negócios de maneira eficaz. Vamos começar examinando os termos individualmente. Pode-se definir modelagem como o ato ou resultado de modelar, sendo aplicada na informática à criação de modelos. Esses modelos podem ser entendidos como representações em escala reduzida, ou seja, simplificações de algo real. Por exemplo, um esquema de produto apresentado em um manual nos permite visualizar efetivamente o produto. O segundo conceito, processo de negócio, é definido como uma sequência de atividades executadas para alcançar um objetivo (resultado) que agregue valor ao cliente.

A modelagem pode se manifestar em uma representação simples, caracterizada por uma quantidade reduzida de elementos e áreas de negócio, ou em uma forma complexa, incorporando uma grande variedade de elementos e envolvendo várias áreas. Os modelos podem adotar formas matemáticas, gráficas, descritivas ou uma combinação de algumas ou todas essas abordagens. São empregados para fins de organização, aprendizado, previsão, medição, explicação, verificação e controle (ABPMP, 2013).

O aprimoramento de processos implica na avaliação e reconfiguração, visando um desempenho mais eficiente e atendimento otimizado às demandas de clientes internos e externos. A eliminação ou automação de processos busca a criação de procedimentos ágeis e eficazes, resultando em custos reduzidos. Além disso, a documentação de processos é essencial para assegurar que a organização possua informações consistentes, permitindo que todos os membros comprehendam e executem as tarefas ou atividades necessárias.

Business Process Model and Notation (BPMN) é um padrão desenvolvido pela *Business Process Management Initiative* (BPMI) e adotado pelo *Object Management Group* (OMG), um consórcio que estabelece padrões para sistemas de informação. A aceitação do BPMN tem experimentado um aumento significativo, especialmente com sua integração nas principais ferramentas de modelagem. Essa notação oferece um conjunto abrangente de símbolos para representar diversos aspectos de processos de negócio. Como em muitas notações, esses símbolos

ANÁLISE E MODELAGEM DE SISTEMAS

delinham relacionamentos claramente definidos, abrangendo desde o fluxo de atividades até a ordem de precedência (ABPMP, 2013).

Dentro da BPMN, as raias dividem o modelo em diversas linhas paralelas, onde cada raias representa um papel desempenhado por um ator na execução do trabalho. O fluxo do trabalho ocorre de uma atividade para outra, seguindo o trajeto estabelecido de raias a raias. A construção dos modelos em BPMN deve ser orientada por padrões corporativos, especialmente se a perspectiva de longo prazo for criar um modelo integrado de negócios para a organização. Esses padrões fornecem diretrizes sobre quando e como as raias são definidas (representando papéis), a decomposição das atividades, a coleta de dados durante a modelagem, entre outros aspectos.

A BPMN possui ícones organizados em conjuntos descritivos e analíticos para atender a diversas necessidades de utilização. Sua notação permite indicar eventos de início, intermediários e fim, fluxo de atividades e mensagens, comunicação intranegócio e colaboração internegócio. Essa notação é útil para apresentar modelos de processos a diferentes públicos, simular processos de negócio com um motor de processo e gerar aplicações em BPMS a partir dos modelos de processos. Suas vantagens incluem um amplo uso e entendimento em muitas organizações, versatilidade para modelar diversas situações de processo e suporte por ferramentas BPMS. No entanto, suas desvantagens incluem a necessidade de treinamento e experiência para usar o conjunto completo de símbolos, dificuldade na visualização do relacionamento entre diferentes níveis de um processo e a possibilidade de diferentes ferramentas serem necessárias para apoiar subconjuntos específicos da notação. A origem na tecnologia da informação pode também inibir seu uso por pessoal de negócios (ABPMP, 2013).

Elementos do BPMN

O BPMN é representado por linhas paralelas, onde cada linha indica um papel distinto a ser desempenhado na execução do trabalho. Este formato é composto por elementos básicos e específicos, que incluem atividade, evento, gateway e conector. Cada um desses elementos é identificado por um ícone específico, conforme mostrado na Figura 1.



Figura 1 | Elementos básicos do BPMN. Fonte: Valle e Oliveira (2013, p. 81).

A atividade representa o trabalho a ser realizado, desdobrando-se em tarefa, subprocesso (colapsado ou expandido) e processo (Figura 2). Por exemplo, uma tarefa pode ser "emitir o pedido". Os eventos são ocorrências no processo que podem influenciar outros elementos e eventos na cadeia de processos, marcando o início e o término dos processos de alguma forma

ANÁLISE E MODELAGEM DE SISTEMAS

em relação à linha do tempo dos acontecimentos. Os gateways são utilizados para controlar o fluxo de sequência e tomar decisões, bifurcações e uniões de caminhos, como o exemplo de verificar o saldo do cliente ao usar o cartão de débito. Os conectores, por sua vez, são elementos de conexão que ligam atividades, eventos e gateways, proporcionando a demonstração de um caminho.

A tarefa pode ter três marcadores: Loop, Instâncias Múltiplas e Compensação, esses símbolos, estão apresentados na Figura 2.

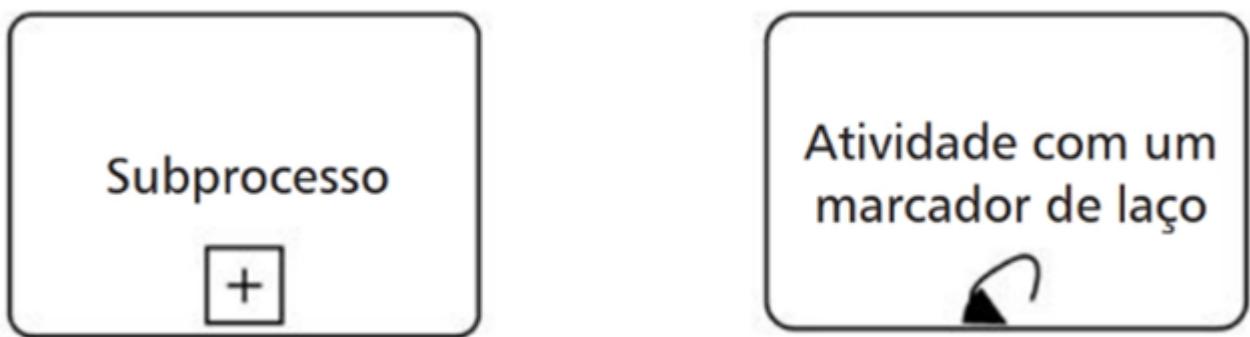


Figura 2 | Tipos de atividades. Fonte: Valle e Oliveira (2013, p. 82).

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 deles. Esses símbolos estão apresentados na Figura 3.

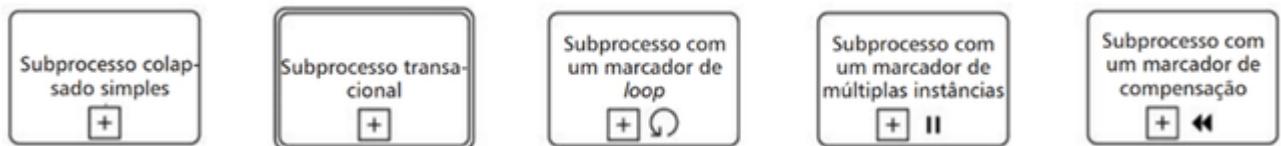
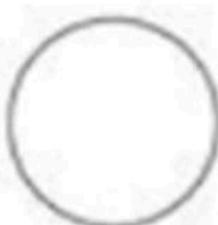


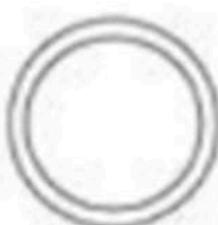
Figura 3 | Subprocessos. Fonte: Valle e Oliveira (2013, p. 83).

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 4.

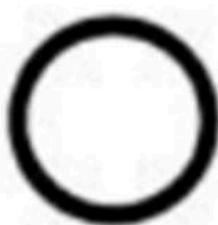
ANÁLISE E MODELAGEM DE SISTEMAS



Evento de início



Evento intermediário



Evento de fim

Figura 4 | Tipos de eventos. Fonte: Valle e Oliveira (2013, p. 84).

ANÁLISE E MODELAGEM DE SISTEMAS

Cada evento exibe uma representação gráfica no centro do elemento. Nos eventos de início e intermediários, essas representações indicam os disparadores, enquanto nos eventos de fim representam os resultados. A Figura 5 ilustra os eventos de início nas duas primeiras colunas, eventos intermediários nas duas colunas seguintes e eventos de fim nas duas últimas colunas.



Figura 5 | Eventos de início, intermediário e de fim. Fonte: adaptada de Valle e Oliveira (2013, p. 84 e 85).

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. Os tipos de gateway estão apresentados na Figura 6.



Figura 6 | 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.

ANÁLISE E MODELAGEM DE SISTEMAS

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.

Os conectores têm a função de orientar o fluxo e são classificados em três categorias: sequência de fluxo, fluxo de mensagem e associação de elementos. Os conectores de sequência de fluxo delineiam a rota a ser seguida para a conclusão do processo, indicando passo a passo as tarefas a serem realizadas. Já os conectores de fluxo de mensagem apresentam uma aparência distinta para evidenciar que se trata de transferência de informação, não envolvendo tarefas. Por fim, a associação de elementos conecta artefatos ao diagrama, proporcionando uma ligação entre os elementos. Na Figura 7, está apresentado esses conectores.

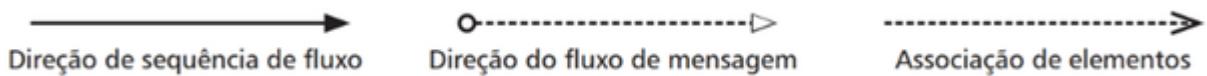


Figura 7 | Tipos de conectores. Fonte: Valle e Oliveira (2013, p. 88).

BMPN usa ainda o conceito de swimlanes 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. A Figura 8 apresenta esses elementos.



Figura 8 | 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. A Figura 9, apresenta os tipos de artefatos.

ANÁLISE E MODELAGEM DE SISTEMAS



Figura 9 | 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.

Siga em Frente...

Exemplos de BPMN

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.

O diagrama representa exclusivamente os elementos principais do fluxo, deixando de exibir detalhes menores relacionados ao fluxo de trabalho. Contudo, ele é útil para compreender as atividades fundamentais do processo. Um exemplo de diagrama está representado na Figura 10.

ANÁLISE E MODELAGEM DE SISTEMAS

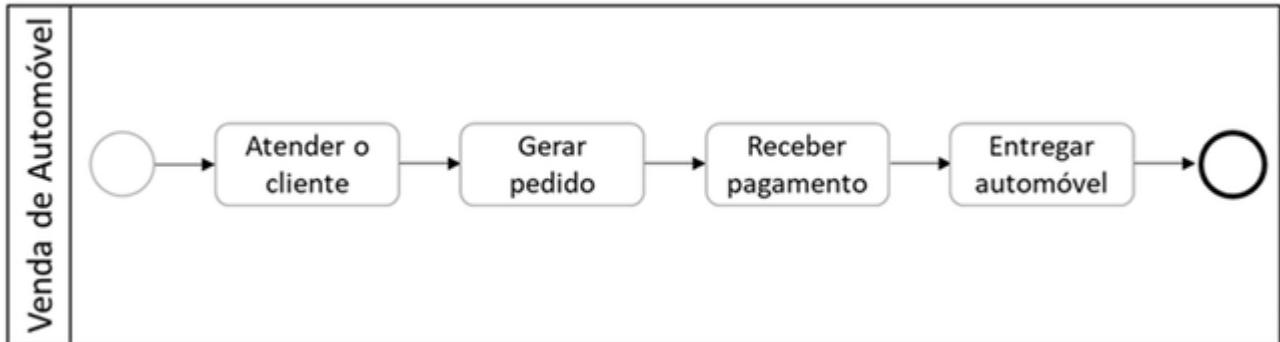


Figura 10 | Diagrama. Fonte: Werlich (2020, p. 77).

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. A Figura 11 apresenta um mapa.

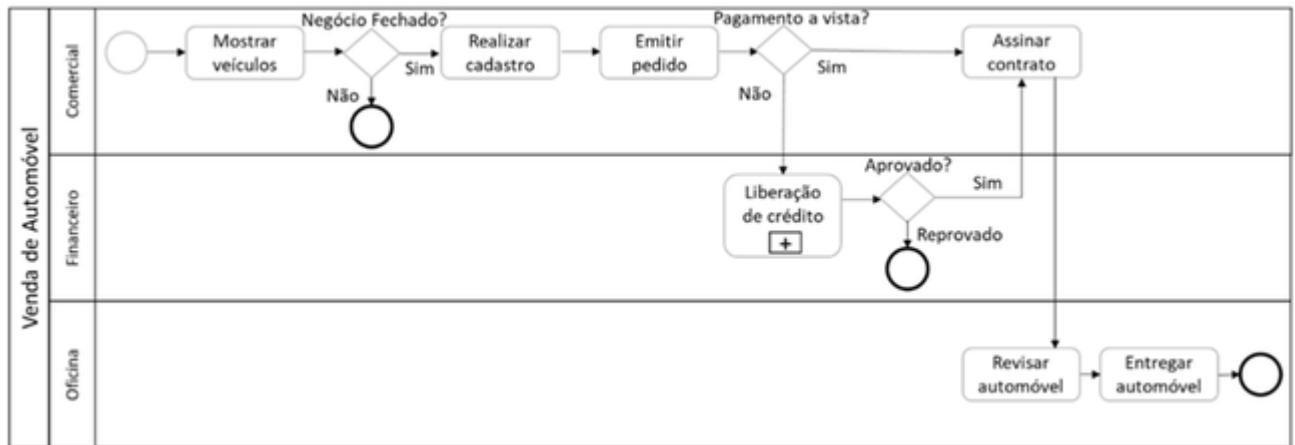


Figura 11 | Mapa. Fonte: Werlich (2020, p. 78).

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. Essa representação de um modelo está exposto na Figura 12.

ANÁLISE E MODELAGEM DE SISTEMAS

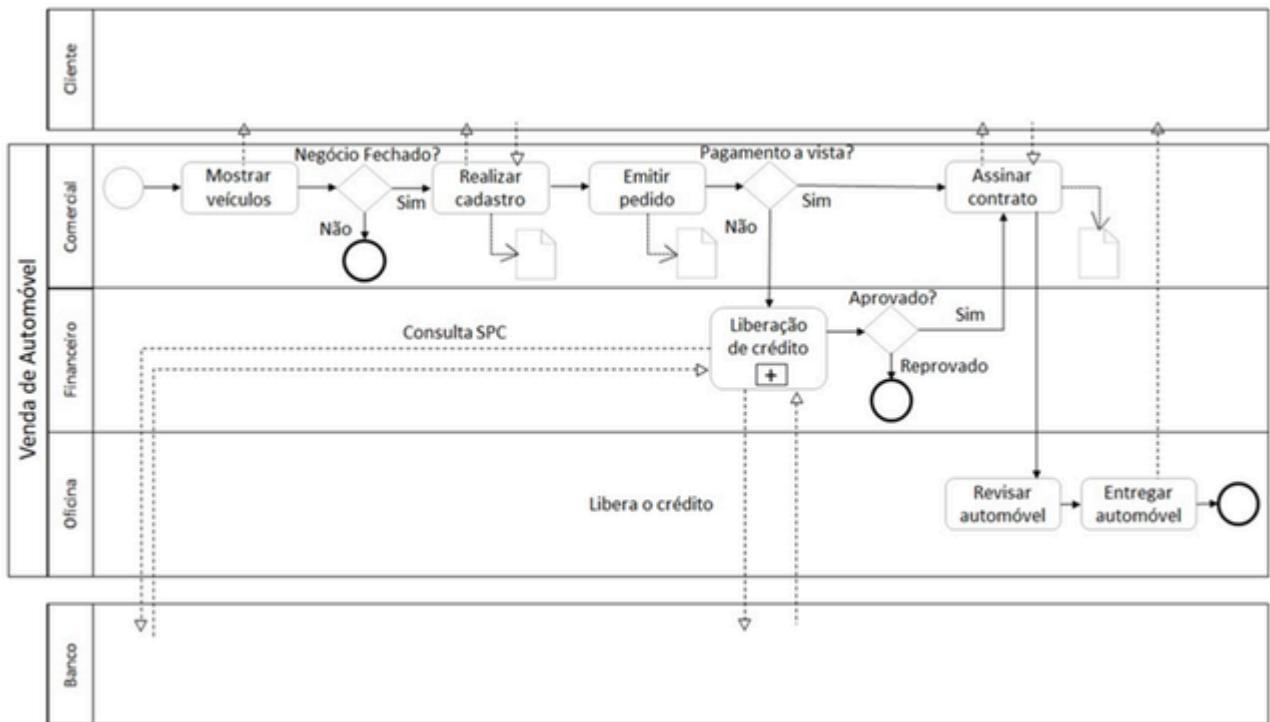


Figura 12 | Modelo. Fonte: Werlich (2020, p. 79).

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.

Vamos Exercitar?

A modelagem utilizando a BPMN é uma abordagem essencial para representar visualmente os processos de negócios de uma organização. A notação oferece um conjunto de símbolos padronizados que permite a criação de diagramas comprehensíveis e consistentes, facilitando a comunicação entre diferentes stakeholders. Através da BPMN, é possível descrever não apenas o fluxo sequencial de atividades, mas também capturar informações detalhadas sobre eventos, decisões, responsabilidades e interações entre processos. Essa metodologia vai além do simples

ANÁLISE E MODELAGEM DE SISTEMAS

desenho de um caminho linear, proporcionando uma visão abrangente e detalhada do funcionamento interno da empresa. Ao utilizar a BPMN na modelagem de processos, as organizações podem aprimorar a compreensão, análise e otimização de suas operações, promovendo uma gestão mais eficiente e alinhada com os objetivos estratégicos.

Um modelo de negócios possível para o problema proposto no início da aula está apresentado a seguir:

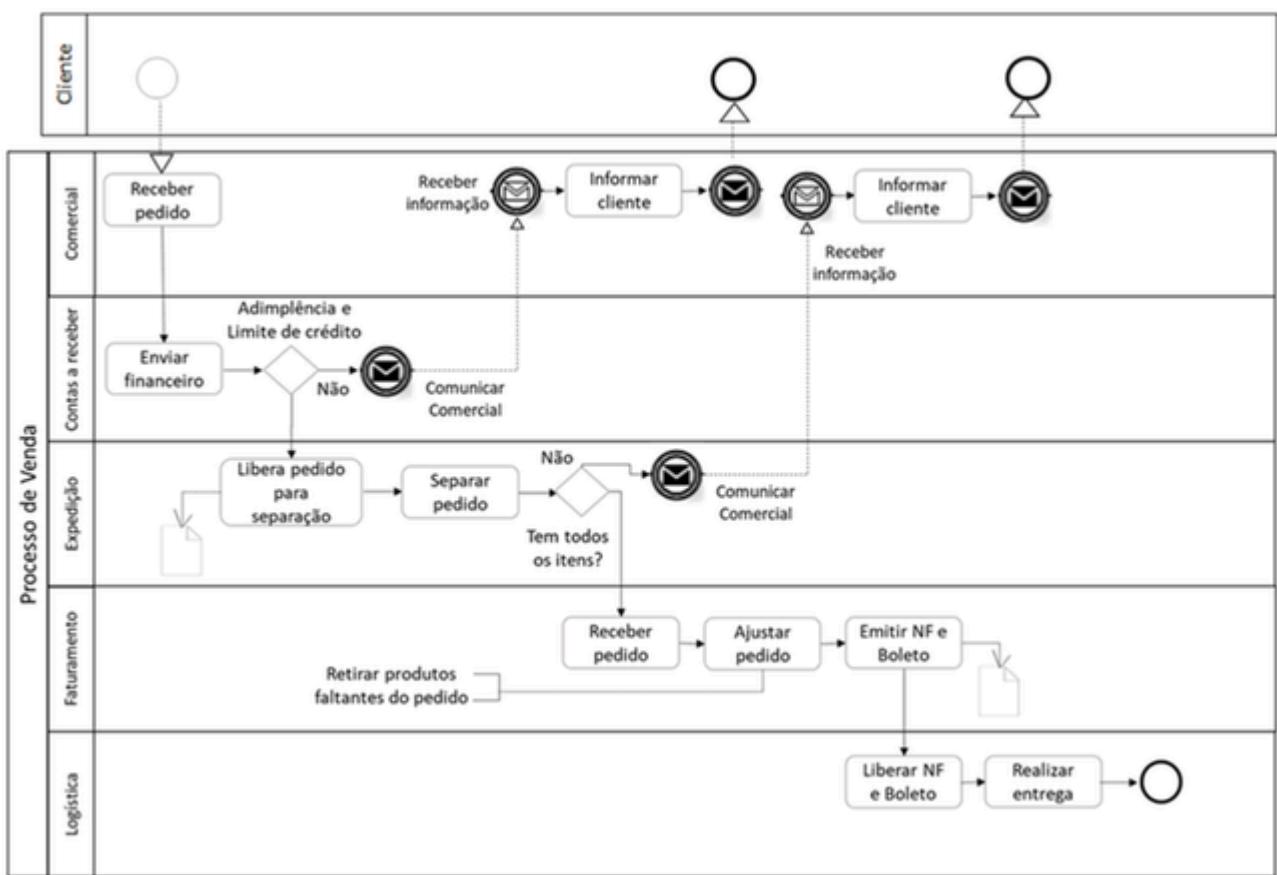


Figura 13 | Modelo de Negócios. Fonte: Werlich (2020, p. 87).

Saiba mais

A Bizagi é uma poderosa ferramenta de automação de processos de negócios que oferece uma abordagem intuitiva e visual para modelagem, execução e otimização de fluxos de trabalho empresariais.

BIZAGI. [Bizagi Modeler](#). [s. d.]

ANÁLISE E MODELAGEM DE SISTEMAS

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.

BROCKE, J. V.; ROSEMANN, M. **Manual de BPM:** gestão de processos de negócio. Porto Alegre: Bookman, 2013.

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

WERLICH, C. **Análise e modelagem de sistemas.** Londrina: Editora e Distribuidora Educacional S.A., 2020.

Aula 5

Encerramento da Unidade

Videoaula de Encerramento



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Olá, estudante! Nesta videoaula, você relembrará os principais pontos dos Processos de negócios. Exploraremos desde os conceitos fundamentais até a abordagem prática do BPM (*Business Process Management*) e da notação BPMN. Esses conteúdos são cruciais para aprimorar sua prática profissional, capacitando-o a entender, modelar e otimizar processos empresariais de forma eficiente. Esteja preparado para uma imersão que transformará sua visão e habilidades no gerenciamento de negócios!

[Clique aqui](#) para acessar os slides da sua videoaula.

Bons estudos!

ANÁLISE E MODELAGEM DE SISTEMAS

Ponto de Chegada

Olá, estudante! Para desenvolver a competência desta Unidade, que é aplicar os conhecimentos de processos de negócios e desenvolver diagramas utilizando o BPMN, devemos conhecer os principais conceitos dos processos de negócios. O processo de negócio é uma sequência de atividades inter-relacionadas e coordenadas que visam alcançar um objetivo específico dentro de uma organização. Ele representa a maneira pela qual as tarefas, recursos, informações e decisões são organizados e executados para produzir um resultado desejado. Os processos de negócio são fundamentais para o funcionamento eficiente de uma empresa, pois proporcionam uma abordagem estruturada para a execução de atividades, promovendo a eficiência operacional.

Além de conhecer os conceitos, é importante compreender as classificações de processos de negócio. Esses processos podem ser classificados em primários, de suporte e de gerenciamento, com interações entre eles. Os processos primários, essenciais ao core business, agregam valor ao cliente final, refletindo em sua satisfação. Já os processos de suporte oferecem apoio aos primários, contribuindo para a eficiência global e exigindo alinhamento organizacional. Por fim, os processos de gerenciamento supervisionam e controlam as atividades organizacionais, otimizando os processos e indiretamente agregando valor ao cliente.

Os processos primários iniciam e encerram fora da organização, estabelecendo uma relação direta com o cliente, enquanto os processos de suporte, como recursos humanos e tecnologia da informação, desempenham um papel crucial ao sustentar as operações. A avaliação dos processos de suporte deve ser contextualizada conforme a natureza da organização, reconhecendo que, em alguns contextos, como em um escritório de contabilidade, eles podem ser considerados processos primários.

Os processos de gerenciamento compartilham características com os de suporte, não proporcionando valor direto ao cliente, mas contribuindo para o alcance dos objetivos organizacionais. O monitoramento e controle desses processos são realizados por meio de práticas como o gerenciamento estratégico e de desempenho, utilizando indicadores específicos de cada organização. A classificação dos processos é crucial para direcionar os esforços das empresas, pois problemas nos processos primários impactam imediatamente o cliente, enquanto problemas nos processos de suporte ou gerenciamento têm um impacto geralmente menor ou imperceptível.

Gerenciar processos é fundamental para uma organização, trazendo benefícios que impactam diretamente os clientes (VALLE; OLIVEIRA; BRACONI, 2013). Esses benefícios incluem o alinhamento dos processos com a estratégia organizacional, melhoria da qualidade, redução de custos, simplificação e automação de processos, além do aumento do envolvimento das partes interessadas e uma melhor delegação de responsabilidades. A abordagem funcional, por outro lado, caracterizada por uma estrutura hierárquica, analisa os processos por departamento, resultando em uma visão isolada e uma falta de interconectividade entre as áreas. Isso leva a

ANÁLISE E MODELAGEM DE SISTEMAS

uma gestão deficiente, orientação limitada para o mercado e objetivos departamentais isolados, não priorizando os objetivos globais.

A visão funcional cria silos organizacionais, onde as atividades são conduzidas de forma isolada, prejudicando a coordenação e a compreensão dos processos interdependentes. A integração horizontal apresenta uma alternativa, promovendo uma abordagem de processos ponta a ponta. Essa abordagem proporciona uma visão abrangente e interfuncional, reconhecendo as interconexões entre os departamentos e contribuindo para atender às necessidades do cliente de maneira mais eficaz. A ênfase em processos ponta a ponta representa uma reconfiguração na estrutura organizacional, priorizando os processos como eixo gerencial principal em relação à abordagem funcional tradicional.

A perspectiva por processos implica uma reavaliação contínua para atender às necessidades em constante evolução dos clientes. A indústria automobilística serve como exemplo, passando de uma abordagem de produção em massa para uma oferta personalizada, evidenciando a necessidade de adaptação dos processos às mudanças no mercado. A coleta detalhada de dados e uma análise aprofundada são cruciais para o sucesso na gestão de processos, garantindo que as atividades estejam alinhadas com os requisitos do cliente de maneira satisfatória.

A gestão é uma atividade crucial para todas as organizações, abrangendo planejamento, organização, direção e controle, conforme destacado por Chiavenato (2014). A função de controle, especificamente, está intrinsecamente ligada à gestão e é fundamental para assegurar o desempenho satisfatório da organização. No contexto do Gerenciamento de Processos de Negócio (BPM), a abordagem expande a visão organizacional, promovendo uma compreensão holística da cadeia de entrega de produtos ou serviços e superando limitações da perspectiva verticalizada.

O BPM, conforme definido pela ABPMP (2013), é essencial para analisar os processos em um nível mais elevado, permitindo a compreensão das relações entre as áreas e a subdivisão em subprocessos. A visão física destaca a operação desintegrada das áreas, enquanto a visão lógica enfoca o processo, evidenciando a interconexão entre as atividades e a influência coletiva no resultado. O BPM é percebido como a construção de processos ponta a ponta, integrando estratégias e objetivos empresariais.

A eficácia dessa abordagem depende da compreensão de elementos como cultura organizacional, clima, estruturas, tecnologia e políticas. A governança dos processos, incorporando controle e prestação de contas, é essencial para alcançar os objetivos organizacionais (ARAÚJO; GARCIA; MARTINES, 2017).

A evolução da melhoria de processos abrange desde a simplificação do trabalho até metodologias avançadas como Seis Sigma e Lean, culminando no Gerenciamento de Processos de Negócio (BPM). Associado a isso está o BPMS, um sistema tecnológico de suporte que possibilita o mapeamento, execução e monitoramento dos processos organizacionais. Essa

ANÁLISE E MODELAGEM DE SISTEMAS

convergência destaca a importância do BPM na gestão organizacional e sua adaptação contínua às mudanças nas necessidades do cliente e no ambiente de negócios (Brocke e Rosemann, 2013).

Outro ponto importante é entender o BPMS que desempenha um papel fundamental ao mapear, executar e monitorar os processos funcionais, proporcionando uma visão completa e automatizando ações e fluxos de informações. De acordo com Araújo, Garcia e Martines (2017), o BPMS é uma evolução em relação ao workflow, integrando diversos fluxos de trabalho e adaptando-se a sistemas legados. A integração eficiente dos processos é crucial, como destacado por Valle e Oliveira (2013), para garantir que estejam alinhados aos objetivos e estratégias organizacionais.

A gestão de negócios, segundo Chiavenato (2014), está intimamente ligada à administração e destaca a importância do estabelecimento de padrões de desempenho mensuráveis. O controle, abrangendo níveis estratégicos, táticos e operacionais, é crucial para alcançar metas, e no contexto do BPM, foca nas atividades de cada processo de negócio. O planejamento do gerenciamento de processos está relacionado ao nível de maturidade da organização, e o CMMI desempenha um papel crucial nesse contexto, promovendo a eficiência do gerenciamento das atividades.

O CMMI, desenvolvido pelo SEI, baseia-se em cinco níveis de maturidade, indicando que quanto maior o nível, melhor a capacidade de gerenciamento do processo. O gestor de processos, ou "*process owner*," desempenha um papel crucial, sendo responsável por garantir recursos, avaliar resultados, facilitar o treinamento e implementar mudanças necessárias. A cultura participativa e a gestão de mudanças são fundamentais para o progresso eficaz do processo.

A conexão entre mudança e BPM é destacada, enfatizando a necessidade de uma disposição significativa para mudanças. No entanto, a resistência é comum devido a fatores como medo do desconhecido e preferência pela zona de conforto. Abordagens para mudança cultural incluem persuasão e contratação alinhada à cultura existente. A gestão de mudanças é vital, mas muitas vezes os métodos BPM não a incorporam efetivamente.

Os gestores de processos devem possuir habilidades técnicas, relacionamento eficaz, proficiência em TI, capacidade de gerenciar equipes multifuncionais e compreensão abrangente do processo de negócio. A tecnologia desempenha um papel crucial, com softwares BPM e BPMS, sendo essenciais para modelagem, automação e controle de processos. A definição de metas, seguindo o método SMART, é vital, pois proporciona sentido ao gerenciamento de processos, vinculando indicadores de desempenho aos objetivos organizacionais.

Um ponto importante para alcançarmos a competência é saber aplicar a modelagem nos processos de negócios. Logo, compreender sobre o que é modelar e os tipos de modelagem de processos.

ANÁLISE E MODELAGEM DE SISTEMAS

A modelagem de processos é uma atividade essencial no gerenciamento organizacional, exigindo habilidades e técnicas para compreender, comunicar e gerenciar componentes dos processos de negócio. Ela envolve a criação de representações, oferecendo visões abrangentes ou focadas em partes específicas dos processos. O objetivo é desenvolver uma representação completa e precisa do processo, com o nível de detalhamento dependendo das expectativas da iniciativa de modelagem.

Os modelos, representações simplificadas de objetos ou atividades, têm diversas aplicações nos negócios, desempenhando funções como organização, descoberta, previsão, medição, explicação, verificação e controle. Os processos podem ser expressos em diferentes níveis de detalhe, de uma visão abstrata contextual a uma visão minuciosa, abrangendo diversas perspectivas.

O conteúdo de um modelo de processo inclui elementos como atividades, fluxo de trabalho, recursos, decisões, eventos, tempo, responsabilidades e objetivos. Esses elementos proporcionam uma compreensão completa do funcionamento do processo, facilitando a análise, otimização e gestão eficaz. Os termos diagrama, mapa e modelo são frequentemente usados de maneira intercambiável, mas representam estágios distintos de desenvolvimento, cada um com maior quantidade de informações e utilidade.

Um diagrama simplifica detalhes para facilitar a compreensão dos fluxos de trabalho, semelhante a um mapa básico que fornece uma visão simplificada. Um mapa oferece uma visão mais precisa e detalhada do processo, incorporando informações sobre relacionamentos cruciais. Já um modelo representa um estado específico do negócio, exigindo mais dados sobre o processo e os fatores que afetam seu comportamento. Ferramentas de modelagem oferecem capacidades de simulação e geração de relatórios para análise aprofundada do processo.

A distinção entre arquitetura de processos e arquitetura de negócio é muitas vezes fonte de confusão na modelagem. Os arquitetos de negócio se concentram em criar modelos conceituais que abordam "O QUE" no negócio, tratando das capacidades de negócio em um nível elevado de abstração. Por outro lado, os modelos de arquitetura de processos se dedicam ao "COMO" do negócio, detalhando como produtos, serviços ou entregáveis são construídos e entregues, relacionados a atividades específicas. Embora ambas as disciplinas busquem melhorar o negócio, os modelos de arquitetura de negócio focam na eficácia, enquanto os de arquitetura de processos se preocupam com a eficiência. A clareza na distinção entre esses modelos é crucial para garantir que atividades estejam alinhadas à entrega de capacidades de negócio específicas, promovendo efetividade organizacional.

Um termo importante que temos que conhecer é notação. A notação, em modelagem de processos de negócio, refere-se a um conjunto padronizado de símbolos e regras que define o significado desses símbolos. A escolha de uma metodologia com normas estabelecidas, como o BPMN (*Business Process Model and Notation*), proporciona vantagens como comunicação eficiente, consistência nos modelos e facilidade de importação/exportação. O BPMN utiliza uma abordagem visual para representar processos de forma comprehensível, sendo amplamente

ANÁLISE E MODELAGEM DE SISTEMAS

adorado para aprimorar a eficiência operacional e alinhar atividades estratégicas. Além disso, outras notações, como fluxogramas, *Event-driven Process Chain* (EPC) e *Unified Modeling Language* (UML), desempenham papéis específicos na modelagem, oferecendo visualizações claras das etapas do processo, análise centrada em eventos e representação abrangente de sistemas complexos na engenharia de software, respectivamente. Cada uma dessas notações contribui para a compreensão, análise e otimização de processos em diferentes contextos organizacionais.

Para a modelagem de processos de negócios, essa disciplina opta pela notação BPMN. O BPMN é um padrão desenvolvido pela *Business Process Management Initiative* (BPMI) e amplamente aceito pelo *Object Management Group* (OMG), destacando-se pela sua integração nas principais ferramentas de modelagem. Essa notação oferece uma gama abrangente de símbolos que representam vários aspectos dos processos de negócio, incluindo fluxo de atividades e ordem de precedência. Dentro do BPMN, as raias organizam o modelo em linhas paralelas, representando diferentes papéis desempenhados por atores na execução do trabalho, seguindo um trajeto estabelecido. É crucial orientar a construção dos modelos por padrões corporativos para criar uma perspectiva de longo prazo de modelos integrados de negócios.

A notação BPMN utiliza ícones descritivos e analíticos para atender diversas necessidades, permitindo indicar eventos, fluxo de atividades, comunicação intra e interorganizacional. Embora tenha ampla aceitação e versatilidade para modelar diversas situações de processo, apresentando vantagens como suporte por ferramentas BPMS, ela também apresenta desafios, como a necessidade de treinamento para usar todos os símbolos, dificuldade na visualização de relacionamentos em diferentes níveis de um processo e a possível necessidade de diferentes ferramentas para suportar subconjuntos específicos da notação. A origem na tecnologia da informação pode ser um obstáculo para o uso por profissionais de negócios.

A notação BPMN (*Business Process Model and Notation*) é composta por diversos elementos que desempenham papéis cruciais na representação gráfica de processos de negócio. Entre esses elementos, destacam-se os eventos, que abrangem desde pontos de início até eventos intermediários e de fim. As atividades são representadas por tarefas e subprocessos, sendo essenciais para descrever unidades de trabalho. Gateways são utilizados para indicar decisões exclusivas ou paralelismo nas sequências de atividades.

Conectores, como fluxos de sequência e associações, delineiam a ordem das atividades e conectam informações adicionais aos elementos do diagrama. Raias, como pools e lanes, fornecem uma estrutura organizacional, representando diferentes participantes ou departamentos envolvidos no processo. Artefatos, como objetos de dados e anotações, adicionam camadas de informação contextual ao diagrama.

Os conectores de mensagem, como fluxos de mensagem, são cruciais para indicar a troca de mensagens entre diferentes participantes ou pools, facilitando a compreensão das interações. Essa diversidade de elementos na notação BPMN oferece flexibilidade para modelar uma ampla

ANÁLISE E MODELAGEM DE SISTEMAS

gama de processos de negócio, proporcionando uma representação visual clara e abrangente que promove a compreensão e a comunicação eficaz entre profissionais de negócios e de TI.

Em resumo, o Gerenciamento de Processos de Negócio e a notação associada, BPMN, destacam-se como ferramentas para organizações contemporâneas que buscam aprimorar operações e alinhar estratégias. Ao incorporar essas práticas, as empresas não só enxergam, analisam e aprimoram seus processos, mas também aprimoram a comunicação entre as partes interessadas. A modelagem de processos com a notação BPMN fornece uma representação visual eficaz, capacitando profissionais a entender complexidades, identificar oportunidades de melhoria e impulsionar a eficiência operacional. Com a crescente aceitação e adoção dessas abordagens, fica claro que BPM e BPMN não são apenas ideias, mas sim impulsionadores para organizações que buscam se adaptar e prosperar em um ambiente empresarial dinâmico e competitivo.

É Hora de Praticar!



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

A atividade proposta neste encerramento consiste na elaboração de um modelo BPMN voltado para a "Solicitação de recursos de viagem", integrando os departamentos Financeiro, Líder de departamento e Colaboradores. Essa iniciativa visa estabelecer uma representação visual clara e abrangente do processo, desde a requisição inicial até a aprovação final e o subsequente tratamento financeiro. A modelagem incorporará elementos essenciais, como pontos de decisão, atribuições de responsabilidade e eventos críticos, visando melhorar não apenas a eficiência operacional, mas também a comunicação entre os diversos participantes envolvidos. O modelo resultante será submetido à validação dos stakeholders pertinentes, sendo refinado conforme necessário para garantir uma implementação eficaz, acompanhada da devida documentação e treinamento para a correta execução do novo processo de solicitação de recursos para viagem.

Atividade proposta: desenvolver um modelo BPMN para a Solicitação de Recursos de Viagem, envolvendo os setores Financeiro, Líder de Departamento e Colaboradores.

- Inclua os elementos necessários para representar claramente o fluxo do processo, desde a solicitação até a aprovação e processamento financeiro.
- Destaque pontos de decisão, responsabilidades específicas de cada participante e eventos que possam influenciar o andamento do processo.

ANÁLISE E MODELAGEM DE SISTEMAS

- Ao final, valide o modelo com stakeholders relevantes, ajuste conforme necessário e esteja pronto para a implementação, documentação e treinamento associados ao novo processo.

O objetivo é criar uma representação visual que melhore a eficiência operacional e a comunicação entre os envolvidos nesse processo específico de solicitação de recursos para viagem.

- Como o Gerenciamento de Processos de Negócio (BPM) contribui para a eficiência e inovação nas organizações, promovendo a adaptação contínua e a maximização do valor entregue aos clientes e partes interessadas?
- Como a Modelagem de Gerenciamento de Processos de Negócio influencia a compreensão, análise e aprimoramento contínuo dos processos organizacionais, contribuindo para uma gestão mais eficiente e alinhada aos objetivos estratégicos?
- Como a escolha da melhor linguagem de Modelagem de Gerenciamento de Processos de Negócio pode influenciar a clareza, eficácia na comunicação e na capacidade de análise das operações organizacionais, considerando a diversidade de opções disponíveis e as particularidades de cada contexto empresarial?

Dê o Play!

[Clique aqui](#) para acessar os slides do Dê o play!

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. Sabendo disso, uma possível solução para a atividade proposta é:

ANÁLISE E MODELAGEM DE SISTEMAS

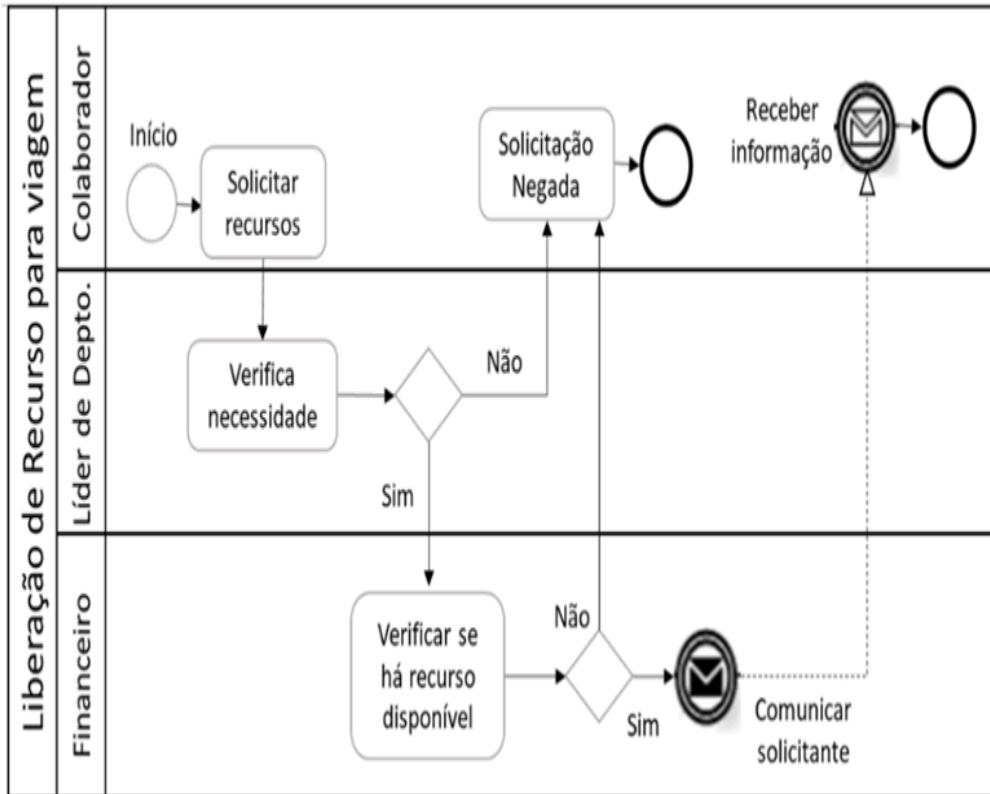


Figura 1 | Modelo de negócios. Fonte: elaborada pela autora.
Lembre-se dependendo da interpretação, pode haver variações entre os modelos.

Desvende as siglas importantes da modelagem de negócios com o infográfico. Entenda sem rodeios o que BPM, BPMN e BPMS representam, e ganhe clareza na linguagem da gestão empresarial.

ANÁLISE E MODELAGEM DE SISTEMAS

GUIA DE SIGLAS

**B
P
M**

Abordagem estratégica que visa aprimorar a eficiência, agilidade e eficácia dos processos organizacionais, impulsionando a excelência operacional e a adaptação contínua às mudanças no ambiente de negócios.

**B
M
P
N**

Linguagem visual poderosa que facilita a compreensão, análise e otimização dos processos organizacionais, proporcionando uma representação clara e padronizada para impulsionar a eficiência e a inovação.

**B
P
M
S**

Conjunto de ferramentas e softwares integrados que apoiam a gestão e a execução de processos de negócio em uma organização. Essas suites ou sistemas oferecem funcionalidades abrangentes, como modelagem de processos, automação, monitoramento.

**S
O
A**

Abordagem que promove a criação e o uso de serviços independentes e interoperáveis, proporcionando flexibilidade e eficiência na integração de sistemas e no desenvolvimento de aplicações empresariais.

Figura | Guia de siglas. Fonte: elaborada pela autora.

ANÁLISE E MODELAGEM DE SISTEMAS

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.

CHIAVENATO, I. **Administração:** teoria, processo e prática. 5. ed. Barueri: Manole, 2014.

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

Unidade 3

Engenharia de Requisitos

Aula 1

Introdução à Engenharia de Requisitos

Introdução à engenharia de requisitos



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Bem-vindo à videoaula dedicada aos Requisitos de software, abordando os conceitos de Requisitos funcionais e Não funcionais. Esses fundamentos são essenciais para sua prática profissional, pois guiam o desenvolvimento de software, garantindo que as necessidades dos usuários sejam atendidas e que o sistema seja eficiente e confiável. Dominar esses conteúdos é crucial para o sucesso na análise, design e implementação de sistemas de software robustos e eficazes. Prepare-se para fortalecer suas habilidades e se destacar no mercado!

[Clique aqui](#) para acessar os slides da sua videoaula.

Bons estudos!

Ponto de Partida

ANÁLISE E MODELAGEM DE SISTEMAS

Um aspecto crucial na elaboração de um projeto de software é a engenharia de requisitos, em que qualquer erro ou omissão pode afetar significativamente o resultado do produto desenvolvido. Esta fase é de extrema importância, pois o sucesso do software depende dela.

O sucesso de um profissional de Tecnologia da Informação (TI) está intimamente ligado à capacidade de compreender as necessidades do cliente (nossa usuário final) e, ainda mais importante, identificar aquilo que o cliente nem mesmo tem consciência de que precisa. Embora possa parecer complexo, existem técnicas que auxiliam na execução dessa etapa fundamental do desenvolvimento de software, capacitando o profissional a compreender e aplicar métodos para especificação, validação e modelagem de requisitos.

O desafio desta aula 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 que serão apresentados.

Para isso, vamos imaginar que 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 aula, que contém conhecimentos acerca de procedimentos da engenharia de requisitos, de seus conceitos e tipos.

Vamos Começar!

ANÁLISE E MODELAGEM DE SISTEMAS

Conceitos e fundamentos sobre requisitos

Uma das principais dificuldades no desenvolvimento de software é estimar o tempo necessário para sua conclusão. Além disso, garantir que o produto final atenda exatamente às necessidades do cliente é outro desafio crucial. No entanto, esses obstáculos podem ser superados por meio da prática da engenharia de requisitos.

De acordo com Pressman (2021), a engenharia de requisitos engloba um conjunto de atividades que visam produzir e manter um documento, conhecido como documento de requisitos, cujo principal objetivo é especificar o que deve ser implementado antes do início do desenvolvimento do software ou sistema. No entanto, é comum que as empresas comecem a desenvolver partes do software antes mesmo de concluir o levantamento de requisitos.

Sommerville (2018) define a engenharia de requisitos como o processo de descrever todas as funcionalidades que um sistema deve possuir, além de detalhar os serviços e as restrições de funcionamento, refletindo as necessidades dos clientes (usuários do software) e contribuindo para o desenvolvimento de um produto de qualidade. O objetivo principal da engenharia de requisitos é garantir que todas as partes envolvidas no desenvolvimento do sistema tenham uma compreensão compartilhada por escrito do problema. Para isso, são utilizados diversos artefatos que asseguram a qualidade das especificações fornecidas.

Mas, o que é o requisito de um sistema?

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 (Pfleeger, 2004).

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 para um concurso. 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.

Note que os requisitos são redigidos de forma que tanto o cliente quanto os desenvolvedores possam compreender claramente o que o software deve realizar. É importante evitar enunciados muito extensos ou subjetivos; eles devem ser concisos e consistentes, garantindo que todas as

ANÁLISE E MODELAGEM DE SISTEMAS

partes envolvidas entendam claramente o que será entregue. O Quadro 1 lista as qualificações que os requisitos devem atender.

QUALIFICAÇÃO	DESCRIÇÃO
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.

Quadro 1 | Qualificação dos requisitos. Fonte: adaptado de Paula Filho (2019, [s. p]).

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 suma importância para o funcionamento do software, sendo vital para sua completude. Sem esse requisito, o programa ficaria incompleto, impossibilitando sua implantação ou conclusão. Para ilustrar, podemos considerar o exemplo de um aluno que só pode acessar uma disciplina se estiver devidamente matriculado nela. Nesse caso, é evidente que o acesso às disciplinas é restrito e essencial para a funcionalidade do software.

ANÁLISE E MODELAGEM DE SISTEMAS

Um requisito classificado como **importante** é aquele que não é crucial para a implantação do software, podendo ser realizado em segundo plano. Embora desejável, não é indispensável. Por exemplo, a criptografia das senhas dos usuários em um sistema acadêmico é importante, mas, se não for implementada inicialmente, o sistema ainda pode ser implantado e atualizado posteriormente com essa funcionalidade.

Já um requisito classificado como **desejável** não é essencial para a conclusão do software; é opcional e pode até ser descartado durante o desenvolvimento. Um exemplo seria a contagem de tempo de acesso dos alunos em um sistema acadêmico, utilizado para determinar a duração do uso do sistema por cada usuário durante o semestre. Esse requisito, classificado como desejável, pode ser deixado de lado caso haja atrasos no desenvolvimento e seja necessário priorizar a implantação do software.

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.

Siga em Frente...

Tipos de Requisitos

Na engenharia de requisitos é importante fazer uma distinção dos diferentes tipos de descrições dos requisitos conforme afirma Sommerville (2018). Eles podem ser classificados como:

- **Requisitos do usuário:** estes delineiam tanto os requisitos funcionais quanto os não funcionais do sistema, utilizando uma linguagem compreensível para os usuários finais (clientes). Eles destacam as funcionalidades e as restrições que o sistema deve apresentar. O resultado é a criação de um documento que não se aprofunda em detalhes técnicos do sistema, mas sim facilita a comunicação entre os desenvolvedores e os clientes.
- **Requisitos do sistema:** estes especificam os detalhes do sistema, baseados nos requisitos do usuário, resultando em um documento que pode servir como parte do contrato entre todas as partes envolvidas no desenvolvimento do sistema. Este estágio marca o início do projeto.

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.

De acordo com Sommerville (2018) e Pressman (2021), os requisitos de um sistema podem ser agrupados em três categorias principais: requisitos funcionais, requisitos não funcionais e requisitos de domínio:

ANÁLISE E MODELAGEM DE SISTEMAS

- **Requisitos funcionais:** estes especificam de forma clara e precisa as funcionalidades específicas que o sistema deve executar ou não. Eles descrevem as ações que o sistema deve realizar em resposta a entradas específicas do usuário ou de outros sistemas.
- **Requisitos não funcionais:** estes impõem restrições sobre as funcionalidades do sistema, definindo padrões específicos de desenvolvimento, plataforma, tempos de resposta e restrições de acesso. Eles estão relacionados às qualidades que o sistema deve apresentar, como desempenho, segurança, usabilidade e escalabilidade.
- **Requisitos de domínio:** estes determinam as características do domínio do sistema, refletindo as particularidades do ambiente em que o sistema será utilizado. Eles podem impor restrições aos requisitos funcionais ou fornecer cálculos específicos sobre determinados requisitos com base no contexto do domínio do sistema.

Os requisitos funcionais delineiam os objetivos específicos do sistema, ou seja, as características que ele deve possuir ao término de seu desenvolvimento. Esses requisitos devem englobar todas as funcionalidades e informações fornecidas pelo cliente antes da implementação do software. Para identificá-los, utilizamos a sigla RF (requisito funcional) seguida de uma numeração (Sommerville, 2018). Abaixo estão exemplos de requisitos funcionais para um sistema de controle acadêmico fictício:

- [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 nível de detalhamento de um requisito desempenha um papel crucial na comunicação com o cliente, ajudando a esclarecer o que precisa ser realizado no sistema. Além disso, facilita a programação da funcionalidade correspondente, pois fornece uma orientação clara ao programador. Portanto, um detalhamento minucioso é essencial para eliminar ambiguidades e garantir que todos compreendam claramente o que está sendo solicitado.

É importante notar que o requisito funcional [RF0003] possui um nível de detalhamento mais elevado em comparação com [RF0001]. Embora o detalhamento seja fundamental nessa fase, cada requisito deve ser específico. Por exemplo, se estamos discutindo informações dos alunos em um requisito, não devemos incluir dados dos professores no mesmo requisito.

Os requisitos não funcionais são identificados pela sigla RFN e estabelecem critérios para qualificar os requisitos funcionais. Esses requisitos abordam aspectos relacionados à qualidade, desempenho, confiabilidade, usabilidade, implementação, entre outros. É importante ressaltar que um único requisito não funcional pode resultar em vários requisitos funcionais. A seguir,

ANÁLISE E MODELAGEM DE SISTEMAS

apresentamos alguns exemplos de requisitos não funcionais para um sistema acadêmico fictício:

- [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 2 com métricas que auxiliam na identificação e especificação 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).

Quadro 2 | Métricas de requisitos não funcionais. Fonte: adaptado de Sommerville (2018, p. 94).

ANÁLISE E MODELAGEM DE SISTEMAS

Os requisitos não funcionais são categorizados com base em três grandes áreas de requisitos:

1. **Requisitos do produto:** estes especificam ou restringem o comportamento do sistema e podem incluir determinações sobre a linguagem de programação a ser utilizada.
2. **Requisitos organizacionais:** estes são derivados das políticas e procedimentos da empresa, do cliente e do desenvolvedor.
3. **Requisitos externos:** estes englobam todos os fatores externos ao sistema que podem influenciar no seu desenvolvimento, desde que estejam em conformidade com a legislação vigente.

A classificação completa dos requisitos não funcionais pode ser visualizada na Figura 1.



Figura 1 | Classificação dos requisitos não funcionais. Fonte: adaptada de Sommerville (2018, p. 92).

Os requisitos de domínio descrevem características específicas e estabelecem condições adicionais aos requisitos funcionais, indicando, por exemplo, critérios obrigatórios para a validação de um requisito. Eles podem consistir em novos requisitos funcionais ou em complementos que impõem restrições adicionais aos requisitos existentes.

Um exemplo de requisito de domínio seria o seguinte cenário: um aluno será considerado aprovado se alcançar, no mínimo, 3000 pontos nas avaliações semestrais e 4000 pontos nas atividades complementares de cada disciplina. Este requisito estabelece duas condições: a pontuação mínima nas avaliações deve ser de 3000 pontos e a pontuação mínima nas atividades

ANÁLISE E MODELAGEM DE SISTEMAS

complementares deve ser de 4000 pontos. Ambas as condições devem ser cumpridas para que o aluno seja considerado aprovado no sistema.

Papel da engenharia de requisitos no ciclo de vida do projeto

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 (Pressman, 2021). Tais processos podem ser melhor visualizados na Figura 2.

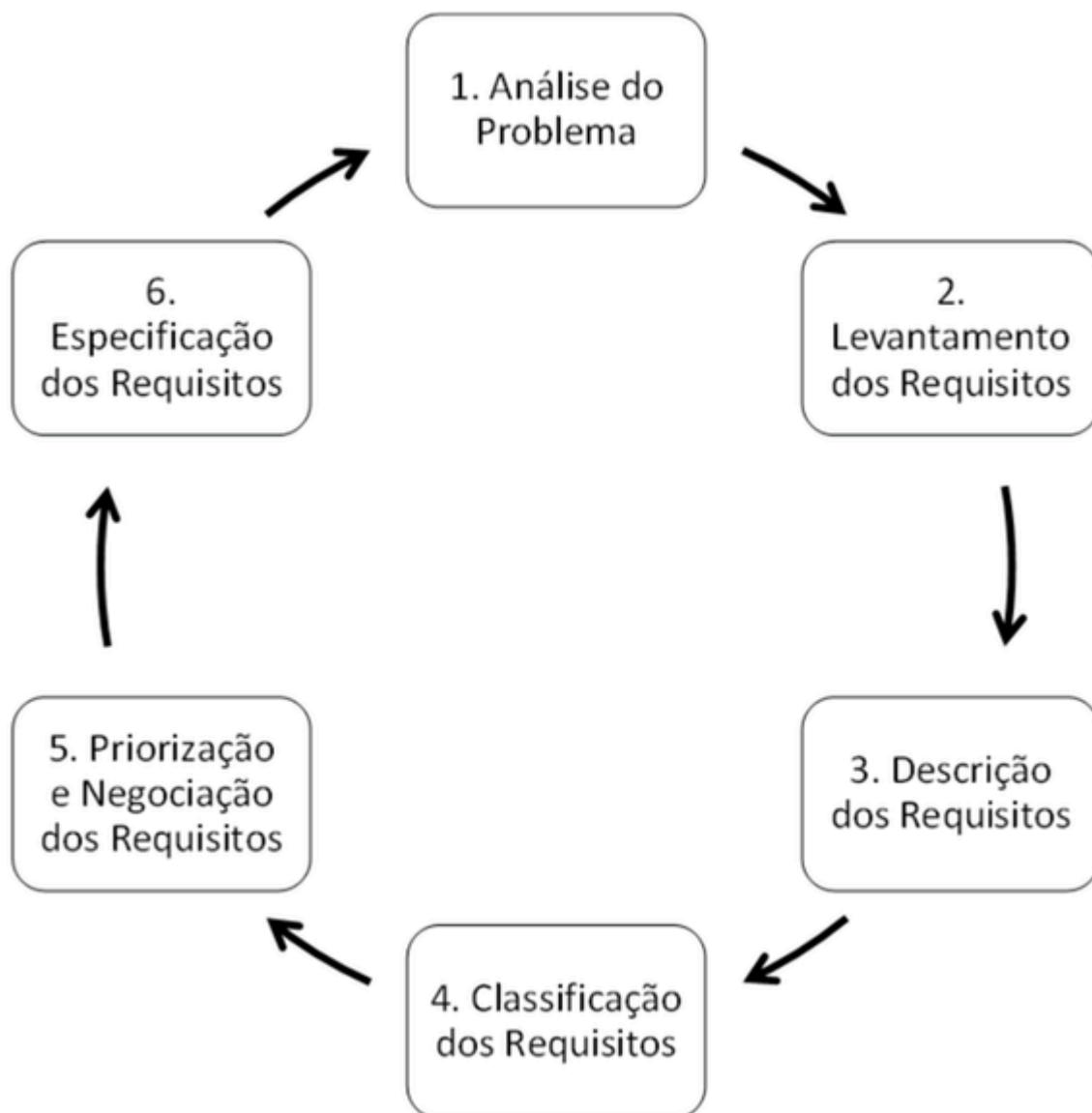


Figura 2 | Atividades dos processos de engenharia de requisitos. Fonte: adaptada de Pfleeger (2004) e Sommerville (2018).

ANÁLISE E MODELAGEM DE SISTEMAS

Todas as atividades relacionadas à engenharia de requisitos têm como objetivo principal a elaboração de um documento de requisitos, como destacado por Sommerville (2018) e Pressman (2021). Estas atividades abrangem as seguintes etapas:

1. **Concepção:** define-se o escopo geral do sistema e identificam-se todos os stakeholders envolvidos.
2. **Elicitação:** realiza-se a coleta de requisitos funcionais e não funcionais, utilizando técnicas como entrevistas, observação e pesquisa.
3. **Elaboração:** detalha-se cada requisito inicialmente expresso em linguagem natural, transformando-os em modelos conceituais, muitas vezes utilizando a Linguagem Unificada de Modelagem (UML), com o objetivo de eliminar erros, omissões, duplicações e inconsistências.
4. **Negociação:** identificam-se e tratam-se requisitos conflitantes, realizando-se negociações entre as partes envolvidas para ajustar e/ou eliminar tais requisitos.
5. **Especificação:** transformam-se os requisitos em uma visão mais detalhada do sistema, considerando o desenvolvimento da solução.
6. **Validação:** realiza-se a validação dos requisitos pelos usuários relevantes, verificando se todas as necessidades foram atendidas, sob a perspectiva do cliente.
7. **Gerenciamento:** consiste na verificação contínua, ao longo do processo, da conformidade dos requisitos com o escopo do projeto, bem como na garantia da rastreabilidade, que envolve o controle de possíveis alterações que os requisitos possam 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 delineiam as funcionalidades, os comportamentos e as características de entrada e saída de um sistema, enquanto os requisitos não funcionais se referem aos atributos e qualidades do sistema.

Os requisitos são o cerne de qualquer software. Antes de iniciar o desenvolvimento de um sistema, é crucial criar uma lista de funcionalidades e características que ele deve possuir - esses são os requisitos iniciais. No entanto, o processo não termina aí. Convido você a explorar ainda mais a engenharia de requisitos.

ANÁLISE E MODELAGEM DE SISTEMAS

Vamos Exercitar?

Retomando a situação-problema do início da aula, 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.

ANÁLISE E MODELAGEM DE SISTEMAS

- [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 ultra- passar 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?

Saiba mais

O livro de Pressman oferece insights valiosos sobre a Engenharia de Requisitos, portanto acesse o Capítulo 7.1 do livro *Engenharia de Software – Pressman* e leia mais sobre o tema:

PRESSMAN, R. S.; MAXIM, B. R. [Engenharia de software](#). Grupo A, 2021.

Para saber mais sobre os requisitos funcionais e não funcionais, leia o Capítulo 4.1 do livro *Engenharia de Software – Sommerville*:

SOMMERVILLE, I. [Engenharia de software](#). 10. ed. São Paulo, SP: Pearson, 2018.

Acesse o artigo *Requisitos de Software: Um Guia Abrangente para o Desenvolvimento de Sistemas de Sucesso* para entender mais sobre a importância de requisitos, os tipos, processos e melhores práticas para o desenvolvimento de sistemas de sucesso.

ALMEIDA, B. [Requisitos de Software: Um Guia Abrangente para o Desenvolvimento de Sistemas de Sucesso](#). Data Perspective. 2023.

Referências

PAULA FILHO, W. P. [Engenharia de software: produtos](#). 4. ed. Rio de Janeiro: LTC, 2019.

ANÁLISE E MODELAGEM DE SISTEMAS

PFLEEGER, S. L. **Engenharia de Software**: teoria e prática. 2. ed. São Paulo: Prentice Hall, 2004.

PRESSMAN, R. S. **Engenharia de software**: uma abordagem profissional. 9. ed. – Porto Alegre: AMGH, 2021.

SOMMERVILLE, I. **Engenharia de software**. 10. ed. São Paulo: Pearson, 2018.

Aula 2

Elicitação e Análise de Requisitos

Elicitação e análise de requisitos



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Bem-vindo à videoaula essencial sobre Identificação, Elicitação e Validação de Requisitos. Estes pilares da Engenharia de Requisitos são importantes para sua prática profissional, fornecendo as bases para entender, coletar e garantir a qualidade dos requisitos de software. Dominar esses conceitos é fundamental para o sucesso na análise e desenvolvimento de sistemas de software que atendam às necessidades dos clientes. Prepare-se para adquirir habilidades valiosas e impulsionar sua carreira na área de desenvolvimento de software.

[Clique aqui](#) para acessar os slides da sua videoaula.

Bons estudos!

Ponto de Partida

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.

ANÁLISE E MODELAGEM DE SISTEMAS

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.

Vamos Começar!

ANÁLISE E MODELAGEM DE SISTEMAS

Técnicas e abordagens para identificar, coletar e compreender os requisitos junto aos stakeholders

Na engenharia de requisitos, a elicitação de requisitos consiste em obter o máximo de informações, seja por meio de extrair dados de fontes ou de interações com indivíduos relevantes, a fim de estabelecer os requisitos de um sistema específico. Essa etapa é fundamental e uma das primeiras na disciplina de Engenharia de Requisitos, conforme delineado por Pressmann (2021).

O processo de elicitação de requisitos não é realizado isoladamente pelo analista de sistemas; ele envolve a colaboração de várias pessoas, conhecidas como stakeholders, que incluem todos os interessados no projeto. Segundo Pressman (2021), a elicitação de requisitos envolve a combinação de elementos para resolver problemas, requerendo uma abordagem colaborativa dos stakeholders. Sommerville (2018) ressalta que os stakeholders são as partes interessadas no projeto, que podem incluir analistas de sistemas, gerentes de projetos, programadores, clientes (contratantes) e usuários finais do sistema.

No processo de elicitação de requisitos, é crucial ter perspectivas diversas sobre as funcionalidades do sistema, e os stakeholders devem participar da elicitação para garantir a ausência de requisitos contraditórios. Estes, se não forem resolvidos antes do desenvolvimento, podem acarretar problemas durante a entrega do produto final. A partir das questões e necessidades dos stakeholders, a elicitação de requisitos visa realizar as seguintes tarefas (PRESSMANN, 2021):

- Especificar o escopo do problema do sistema.
- Avaliar as oportunidades de reutilização de soluções existentes.
- Identificar os stakeholders diretamente envolvidos com o sistema.
- Elicitar e qualificar os requisitos do sistema.
- Analisar os requisitos elicitados.
- Validar os requisitos elicitados.

A engenharia de requisitos abrange todas as etapas que colaboram para a elaboração de um documento de requisitos e sua subsequente manutenção ao longo do ciclo de vida do projeto. Os procedimentos fundamentais de levantamento e análise de requisitos de um sistema incluem as seguintes atividades (SOMMERVILLE, 2018):

1. **Concepção ou entendimento do domínio:** consiste em estabelecer uma compreensão básica do problema a ser abordado. É essencial que todos os stakeholders tenham uma compreensão clara do que será realizado, incluindo os limites do que será e do que não será realizado.
2. **Coleta e classificação de requisitos (elicitação):** envolve todas as atividades destinadas a identificar o máximo de requisitos dos stakeholders. Os requisitos são então classificados em funcionais e não funcionais.

ANÁLISE E MODELAGEM DE SISTEMAS

3. Negociação de requisitos: neste estágio, após a lista de requisitos ter sido compilada, é crucial determinar se há conflitos entre eles. Por exemplo:

- RF0025 - O cliente deve ter uma senha para acessar o sistema.
- RF0078 - Não é necessário que um cliente tenha uma senha para acessar o sistema.

Observe que o requisito RF0078 contradiz completamente o requisito RF0025. Como lidar com essa situação? Provavelmente, durante a coleta de requisitos, um stakeholder solicitou acesso com senha para os clientes, enquanto outra pessoa envolvida no projeto indicou que o uso de senha era desnecessário. O procedimento para resolver esse impasse envolve convocar as partes envolvidas para resolver o conflito, realizando as seguintes análises:

- Verificação de requisitos: todos os requisitos são analisados para garantir que estejam alinhados com as necessidades dos stakeholders.
- Definição de prioridades: é feita a determinação dos requisitos mais importantes, que requerem total atenção para o sucesso do projeto.

4. Validação dos requisitos: uma verificação abrangente dos requisitos é realizada, resultando em um "termo de aceitação" no qual todas as partes envolvidas (stakeholders do sistema projetado) concordam e validam os requisitos.

5. Documentação: este é o resultado da Engenharia de Requisitos, no qual é gerado um documento contendo todas as especificações dos requisitos funcionais e não funcionais. Esse documento serve como o principal meio de comunicação entre o analista de sistemas ou engenheiro de software e o cliente.

A Figura 1 ilustra o processo de levantamento e análise de requisitos. É evidente que a partir da compreensão do domínio do que será efetivamente realizado no sistema, ocorre a coleta e classificação de requisitos, com o propósito de determinar o que será implementado no sistema, facilitando sua organização para melhor controle. A negociação entra em cena para estabelecer os ajustes necessários e auxiliar na definição das prioridades durante o desenvolvimento. A especificação marca o início da documentação dos requisitos, enquanto a validação consiste em uma verificação geral de todos os requisitos, com o objetivo final de documentar os requisitos de forma precisa.



Figura 1 | Levantamento e análise de requisitos. Fonte: Werlich (2020, p. 133).

ANÁLISE E MODELAGEM DE SISTEMAS

O levantamento e a análise de requisitos, proposto por Sommerville (2018), é um processo de validação continuada. A Figura 1 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

Elicitação de requisitos

O processo de elicitação de requisitos pode variar entre as empresas, pois cada uma possui seus próprios métodos e procedimentos de elicitação. De acordo com Sommerville (2018), as atividades do processo de elicitação de requisitos incluem:

1. **Descoberta de requisitos:** uma atividade interativa com a participação ativa dos stakeholders, onde os usuários finais do sistema contribuem com sua experiência para auxiliar na coleta dos requisitos do sistema.
2. **Classificação e organização de requisitos:** consiste em agrupar os requisitos para identificar se há repetições ou se eles pertencem a subsistemas específicos.
3. **Priorização e negociação de requisitos:** em colaboração com os stakeholders, estabelece-se a prioridade de cada requisito (essencial, importante ou desejável), ajudando a definir as etapas mais críticas do desenvolvimento do sistema.
4. **Especificação de requisitos:** nesta fase, os requisitos são documentados, podendo ser em formatos formais ou 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:** tipicamente conduzida com um questionário para identificar as necessidades que o sistema deve atender. Nesta fase, é crucial praticar a escuta ativa e registrar o máximo de informações obtidas.
- **Reuniões:** utilizando técnicas como o brainstorming para identificar requisitos que ainda não foram determinados e resolver conflitos de requisitos surgidos durante as entrevistas.
- **Documentos:** coleta de documentos que possam esclarecer as funcionalidades do sistema, tais como relatórios, planilhas, registros em papel, cadernos de anotações, entre outros.
- **Etnografia:** observação e análise do verdadeiro modo de trabalho dos usuários finais (em contraste com o que eles possam expressar), resultando na identificação de requisitos significativos para o sistema.

ANÁLISE E MODELAGEM DE SISTEMAS

A especificação de requisitos consiste na formalização e abstração das funcionalidades que o software deve executar, utilizando descrições claras e objetivas. Esse processo de sistematização busca caracterizar o problema a ser resolvido, resultando na elaboração de um documento de especificação de requisitos. De acordo com Pfleeger (2004), a melhor abordagem para iniciar a especificação de requisitos é adotar uma estrutura hierárquica, começando pelos atributos gerais do sistema e, em seguida, detalhando progressivamente em subníveis até alcançar os atributos mais específicos.

A especificação de requisitos serve como o canal de comunicação entre o analista de sistemas e os programadores responsáveis pelo desenvolvimento do software. É crucial que os requisitos sejam especificados de maneira que não haja margem para interpretações divergentes, uma vez que o programador utilizará essa especificação como base para implementar exatamente o que está descrito. A especificação de requisitos abrange todas as funcionalidades e suas respectivas restrições, tanto funcionais quanto não funcionais, geralmente apresentadas em tabelas ou documentos específicos, podendo também incluir diagramas de casos de uso para facilitar a compreensão ou a utilização de prototipagem.

Os casos de uso são diagramas que compõem a Linguagem Unificada de Modelagem, conhecida como UML (*Unified Modeling Language*), são uma técnica fundamental na engenharia de requisitos, usada para capturar os requisitos funcionais de um sistema de forma clara e comprehensível. Eles descrevem interações entre atores (usuários ou sistemas externos) e o sistema em questão, apresentando cenários detalhados de como o sistema será utilizado para alcançar determinados objetivos. Esses casos de uso ajudam a definir o escopo do sistema, identificar requisitos específicos e fornecer uma visão geral das principais funcionalidades esperadas. Além disso, são uma ferramenta valiosa para comunicação entre stakeholders, permitindo que todos compreendam e validem os requisitos do sistema de forma eficaz.

De acordo com Paula Filho (2019), a prototipagem consiste na criação de uma versão simplificada do sistema a ser desenvolvido, baseada no princípio da verificação do custo-benefício, em que a experiência do usuário desempenha um papel crucial no processo de prototipagem. Além disso, o mesmo autor destaca várias vantagens associadas à técnica de prototipagem: (i) possibilita a identificação precoce de problemas; (ii) permite verificar se os requisitos do software atendem às necessidades dos clientes; (iii) aprimora a comunicação com o cliente, apresentando o progresso do software; (iv) facilita a realização de testes para avaliar a aceitação do software pelo cliente, juntamente com o feedback fornecido pelo cliente.

Na técnica da prototipagem, é possível antecipar vários problemas, permitindo uma melhor estimativa do tempo necessário para resolvê-los. Pfleeger (2004) categoriza a prototipagem em três abordagens distintas:

- **Protótipo descartável:** desenvolvido com o propósito de explorar o problema a ser solucionado e avaliar a viabilidade da solução. Este tipo de protótipo não é utilizado posteriormente, sendo descartado após a fase exploratória.

ANÁLISE E MODELAGEM DE SISTEMAS

- **Protótipo evolutivo:** criado quando os clientes têm dúvidas sobre alguma parte do software, geralmente relacionada à interface gráfica. Esse tipo de protótipo é integrado ao software final.
- **Prototipagem rápida:** construção de partes do software para avaliar a viabilidade dos requisitos. Nesta abordagem, o objetivo é determinar se a implementação do projeto é viável.

A negociação de requisitos tem como finalidade, conforme destacam Pressman (2021), 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.

Siga em Frente...

Monitoramento e validação de requisitos

O monitoramento de requisitos é um processo essencial para assegurar que o escopo do software desenvolvido seja alcançado. A cada modificação em um ou mais requisitos, é fundamental garantir a rastreabilidade das alterações, utilizando ferramentas de controle adequadas. Por exemplo, é necessário determinar o status de cada requisito (proposto, em progresso, em revisão, adiado, excluído, aprovado, etc.) e criar uma matriz de rastreabilidade para facilitar o gerenciamento dos requisitos. Essa matriz deve incluir todos os requisitos, suas dependências, o status atual, os responsáveis pela modificação e aprovação dos requisitos, e, especialmente, as datas relevantes de cada evento ocorrido.

Toda mudança, por mínima que seja, em um requisito deve ser cuidadosamente gerenciada para evitar duplicações. Na Figura 2, é possível visualizar o processo de gestão de mudanças de requisitos, que visa garantir a rastreabilidade das alterações durante o desenvolvimento do software. Este processo inclui a análise de impacto das mudanças propostas para determinar sua viabilidade técnica e financeira (SOMMERVILLE, 2018). Muitas vezes, a pressa em alterar uma funcionalidade no sistema em desenvolvimento pode comprometer a gestão de mudanças de requisitos, pois a modificação é realizada sem que a documentação dos requisitos seja devidamente atualizada, resultando em inconsistências e tornando a documentação dos requisitos desatualizada.

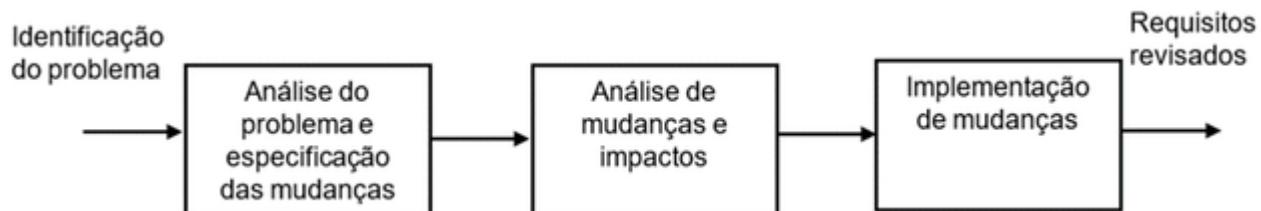


Figura 2 | Gerenciamento de mudança de requisitos. Fonte: adaptada de Sommerville (2018, p. 116).

ANÁLISE E MODELAGEM DE SISTEMAS

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).

Durante o processo de validação de requisitos podem existir diferentes tipos de verificação (SOMMERVILLE, 2018):

- **Validade:** 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 de 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 propósito da validação de requisitos é identificar falhas nos requisitos documentados.

Pressman (2018) ressaltam que erros comuns nos sistemas incluem inconsistências, contradições, duplicações, ambiguidades, omissões e imprecisões. Para validar os requisitos, algumas perguntas fundamentais podem ser feitas:

- 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 nível de detalhamento das perguntas pode ser ampliado e aprofundado de acordo com a complexidade do sistema a ser desenvolvido. Em muitos casos, será necessário realizar uma investigação mais aprofundada e formular mais perguntas para determinar a precisão dos requisitos. É importante ressaltar que o principal objetivo da validação de requisitos é a detecção e eliminação de erros que possam resultar em atrasos e aumento de custos, tanto durante a produção do software quanto para o cliente.

Um recurso que pode contribuir para garantir a qualidade da validação de requisitos é o checklist, uma lista de perguntas elaborada para analisar cada requisito do sistema. Essa técnica tem como objetivos: (i) identificar erros em diversos níveis, como função, lógica e implementação; (ii) verificar se o sistema atende aos requisitos especificados; (iii) assegurar que o software desenvolvido foi implementado conforme os padrões previamente estabelecidos.

ANÁLISE E MODELAGEM DE SISTEMAS

Vamos Exercitar?

O desafio desta aula é determinar como será feita a coleta de informações, a fim de 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?

1. 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).

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

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

- Estimule a participação dos stakeholders.
- Escreva os requisitos de forma simples e objetiva.
- Evite o excesso de detalhamento nos requisitos, pois isso dificulta a leitura e a compreensão.
- 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.

ANÁLISE E MODELAGEM DE SISTEMAS

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:

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

Quadro 1 | Requisitos não funcionais. Fonte: Werlich (2020, p. 143).

Saiba mais

O livro de Pressman oferece insights valiosos sobre o Levantamento de Requisitos, portanto acesse o Capítulo 7.3 do livro *Engenharia de Software – Pressman* e leia mais sobre o tema:

Pressman, R. S.; MAXIM B. R. [Engenharia de software](#). (9th edição). Grupo A, 2021.

ANÁLISE E MODELAGEM DE SISTEMAS

Para saber mais sobre a Elicitação de Requisitos, leia o Capítulo 4.3 de *Engenharia de Software - Sommerville*:

SOMMERVILLE, I. [Engenharia de software](#). 10. ed. São Paulo, SP: Pearson, 2018.

Quer saber quais são as técnicas utilizadas para a Elicitação de Requisitos?! Leia o artigo *Técnicas de Elicitação de Requisitos*:

ACERT. [Técnicas de Elicitação de Requisitos](#). 2023.

Referências

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

PFLEGER, S. L. **Engenharia de Software**: teoria e prática. 2. ed. São Paulo: Prentice Hall, 2004.

PRESSMAN, R. S. **Engenharia de software**: uma abordagem profissional. 9. ed. – Porto Alegre: AMGH, 2021.

SOMMERVILLE, I. **Engenharia de software**. 10. ed. São Paulo: Pearson, 2018.

WERLICH, C. **Análise e modelagem de sistemas**. Londrina: Editora e Distribuidora Educacional S.A., 2020.

Aula 3

Modelagem de Requisitos

Modelagem de requisitos



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

ANÁLISE E MODELAGEM DE SISTEMAS

Olá, estudante! Nesta videoaula, você conhecerá a Modelagem e Especificação de Requisitos, com foco em Diagramas. Estes conteúdos são fundamentais para sua prática profissional, capacitando-o a traduzir as necessidades dos clientes em requisitos claros e comprehensíveis. Dominar a modelagem e especificação de requisitos, incluindo o uso de diagramas, é essencial para o desenvolvimento de sistemas de software precisos e alinhados às expectativas dos usuários. Prepare-se para adquirir habilidades valiosas que impulsionarão sua carreira na Engenharia de requisitos.

[Clique aqui](#) para acessar os slides da sua videoaula.

Bons estudos!

Ponto de Partida

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.

Pensando nessa documentação, vamos apresentar métodos de documentar os requisitos, seja por meio de linguagem natural ou através de modelos especificados e próprios para essa atividade. Para que você exerça esse seu lado de modelagem de requisitos, vamos imaginar que você está trabalhando como analista de sistemas e faz parte de uma equipe responsável pela engenharia de requisitos do software. Você documentará os seguintes requisitos:

- [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 (incluir, consultar, alterar e excluir) os dados dos clientes: nome, endereço, CPF ou CNPJ, e-mail, telefones.
- [RF0011] - O sistema deve gerar um relatório das Plantas disponíveis para venda.

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 elição dos requisitos?

Vamos Começar!

Especificação de requisitos funcionais e não funcionais

ANÁLISE E MODELAGEM DE SISTEMAS

A modelagem de requisitos tem como principal objetivo concentrar-se no "o que será feito" e não no "como será feito". Conforme afirmado por Pressman (2021), o modelo de requisitos deve alcançar três objetivos fundamentais:

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

Para compreender completamente os requisitos, é essencial notar que não apenas descrevem o fluxo de informações que entram e saem do sistema, bem como a transformação dos dados dentro do sistema, mas também delineiam todas as restrições relacionadas ao seu comportamento e desempenho, como afirmado por Pfleeger (2004). Os requisitos permitem: (i) explicar, na perspectiva dos desenvolvedores, como os clientes desejam que o sistema funcione; (ii) definir as funcionalidades e atributos que o sistema deve possuir; (iii) informar à equipe de testes o que deve ser validado junto ao cliente.

Observa-se no Quadro 1 uma série de perguntas que devem ser feitas sobre cada requisito, visando promover o entendimento tanto do cliente quanto dos desenvolvedores, garantindo a qualidade dos requisitos levantados.

PERGUNTA	DESCRÍÇÃO
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

ANÁLISE E MODELAGEM DE SISTEMAS

funções desnecessárias (e consequentemente perda de tempo no desenvolvimento).

Quadro 1 | Características dos requisitos. Fonte: adaptado de Pfleeger (2004, [s. p.]).

Existem diversas técnicas de modelagem de requisitos, a primeira, conforme Sommerville (2018), 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).

A especificação de requisitos envolve a documentação dos requisitos do usuário e do sistema em um documento formal. Em teoria, esses requisitos devem ser claros, inequívocos, facilmente comprehensíveis, completos e consistentes, embora seja praticamente impossível alcançar essas condições idealmente. Os stakeholders tendem a interpretar os requisitos de maneiras diversas, frequentemente resultando em conflitos e incoerências. Os requisitos de usuário são geralmente redigidos em linguagem natural e podem ser complementados por diagramas e tabelas apropriados no documento de requisitos. Por outro lado, os requisitos do sistema também podem ser expressos em linguagem natural, mas outras notações, como formulários, gráficos ou modelos matemáticos, podem ser utilizadas. Quadro 2 resume as diferentes notações possíveis para escrever requisitos de sistema.

NOTAÇÃO	Descrição
Sentenças em linguagem natural	Os requisitos são escritos usando frases numeradas em linguagem natural. Cada frase deve expressar um requisito.
Linguagem natural estruturada	Os requisitos são escritos em linguagem natural em um formulário ou template. Cada campo fornece informações sobre um aspecto do requisito.
Notações gráficas	Modelos gráficos, suplementados por anotações em texto, são utilizados para definir os requisitos funcionais do sistema. São utilizados com frequência os

ANÁLISE E MODELAGEM DE SISTEMAS

	diagramas de casos de uso e de sequência da UML.
Especificações matemáticas	Essas notações se baseiam em conceitos matemáticos como as máquinas de estados finitos ou conjuntos. Embora essas especificações inequívocas possam reduzir a ambiguidade em um documento de requisitos, a maioria dos clientes não comprehende uma especificação formal.

Quadro 2 | Notações para escrever requisitos do sistema. Fonte: adaptado de Sommerville (2018, p. 104).

Uso de linguagem natural

Desde os anos 1950, a linguagem natural tem sido empregada na redação de requisitos de software. É uma forma expressiva, intuitiva e universal de comunicação. No entanto, também é intrinsecamente vaga e ambígua, sua interpretação variando conforme a experiência do leitor. Apesar das diversas propostas de alternativas para escrever requisitos, nenhuma delas foi amplamente adotada, e a linguagem natural continuará sendo o meio predominante de especificar requisitos de sistema e software.

Para mitigar possíveis mal-entendidos ao redigir requisitos em linguagem natural, é recomendável seguir estas diretrizes simples (SOMMERVILLE, 2018):

1. Estabelecer um formato padrão e garantir sua adoção em todas as definições de requisitos. Padronizar o formato reduz a probabilidade de omissões e facilita a verificação dos requisitos. Sempre que viável, é sugerido expressar o requisito em uma ou duas frases em linguagem natural.
2. Utilizar a linguagem de maneira consistente para distinguir entre requisitos obrigatórios e desejáveis. Os requisitos obrigatórios são aqueles que o sistema deve obrigatoriamente suportar e geralmente são expressos com o termo "deve". Já os requisitos desejáveis não são essenciais e são indicados pelo termo "pode".
3. Destacar partes importantes do requisito usando realces de texto, como negrito, itálico ou cor.
4. Evitar presumir que os leitores tenham conhecimento técnico em engenharia de software. Termos como "arquitetura" e "módulo" podem ser facilmente mal interpretados. Sempre que possível, evitar o uso de jargões, abreviações e acrônimos.

ANÁLISE E MODELAGEM DE SISTEMAS

5. Associar um racional a cada requisito de usuário sempre que possível. O racional deve explicar por que o requisito foi incluído e quem o propôs (a origem do requisito), facilitando a identificação de quem contatar caso o requisito precise ser alterado. O racional dos requisitos é particularmente útil em momentos de mudança, ajudando a discernir quais alterações seriam indesejáveis.

A linguagem natural estruturada é uma forma de redigir os requisitos do sistema de maneira padronizada, em oposição ao texto livre. Essa abordagem mantém a expressividade e a clareza da linguagem natural, porém impõe uma certa uniformidade à especificação. As notações que adotam a linguagem estruturada utilizam modelos para descrever os requisitos do sistema. Essa descrição pode fazer uso de construções da linguagem de programação para apresentar alternativas e iterações, possibilitando destacar elementos-chave por meio de sombreamento ou diferentes fontes.

Para adotar uma abordagem estruturada na especificação de requisitos de sistema, é necessário estabelecer um ou mais modelos para os requisitos e representá-los como formulários estruturados. A especificação pode ser organizada em torno dos objetos manipulados pelo sistema, das funções executadas por ele ou dos eventos processados (SOMMERVILLE, 2018).

A adoção de especificações estruturadas remove algumas das dificuldades encontradas na redação de requisitos em linguagem natural. A variabilidade na especificação é minimizada e os requisitos são organizados de maneira mais eficiente. No entanto, em certas situações, pode ser desafiador expressar os requisitos de forma clara e inequívoca (SOMMERVILLE, 2018).

Siga em Frente...

Uso de modelos e diagramas

Uma técnica de modelagem de requisitos empregada na fase de Elicitação de requisitos é a técnica REMO (sigla em inglês de *Requirements Elicitation oriented by business process MOdeling*). Esta técnica possibilita a integração da modelagem de processos de negócios, utilizando a notação BPMN (*Business Process Model and Notation* - Modelo e Notação de Processos de Negócio), com a Elicitação de requisitos, conforme mencionado por Vieira (2012).

A técnica REMO possibilita a extração de requisitos a partir dos diagramas de processos de negócios, apoiada por um conjunto de heurísticas. Vieira (2012) descreve que o método de aplicação da técnica REMO comprehende duas fases distintas. Na primeira fase, o foco está no entendimento do contexto para explorar o domínio do problema do software que será desenvolvido. Nessa etapa, é elaborado um documento pelo Analista de Sistemas, contendo informações cruciais para compreender o domínio do software, incluindo problemas e necessidades identificados, papéis envolvidos nos processos, recursos necessários e

ANÁLISE E MODELAGEM DE SISTEMAS

disponíveis, e diagramas de processos de negócios. Já na segunda fase, a atenção se volta para os requisitos, onde ocorre a extração e descrição dos requisitos do sistema.

Observe na Figura 1 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 1, 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).

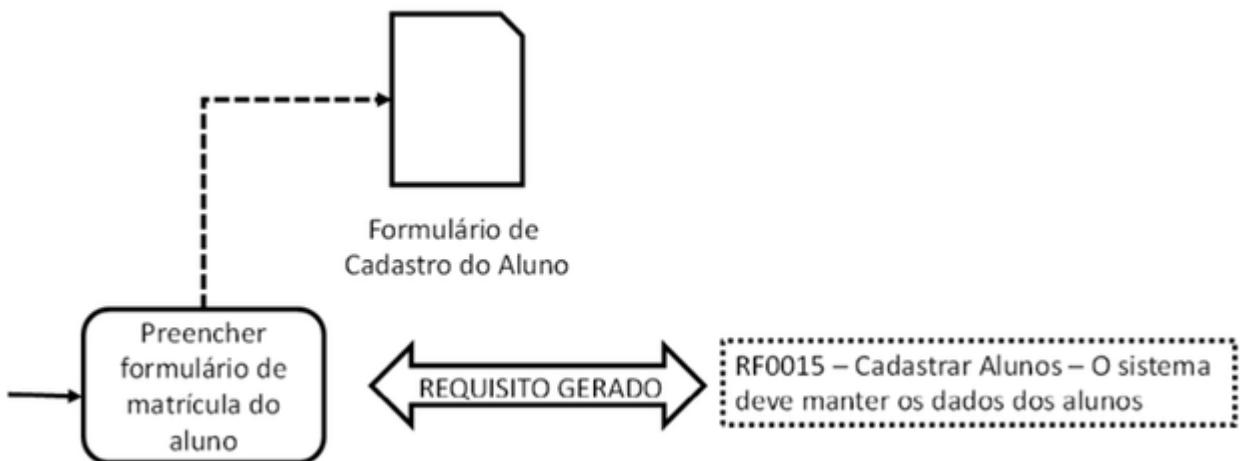


Figura 1 | Requisito gerado a partir da modelagem de processos de negócio. Fonte: Werlich (2020, p. 150).

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. Na Figura 2 são ilustrados três 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.

ANÁLISE E MODELAGEM DE SISTEMAS

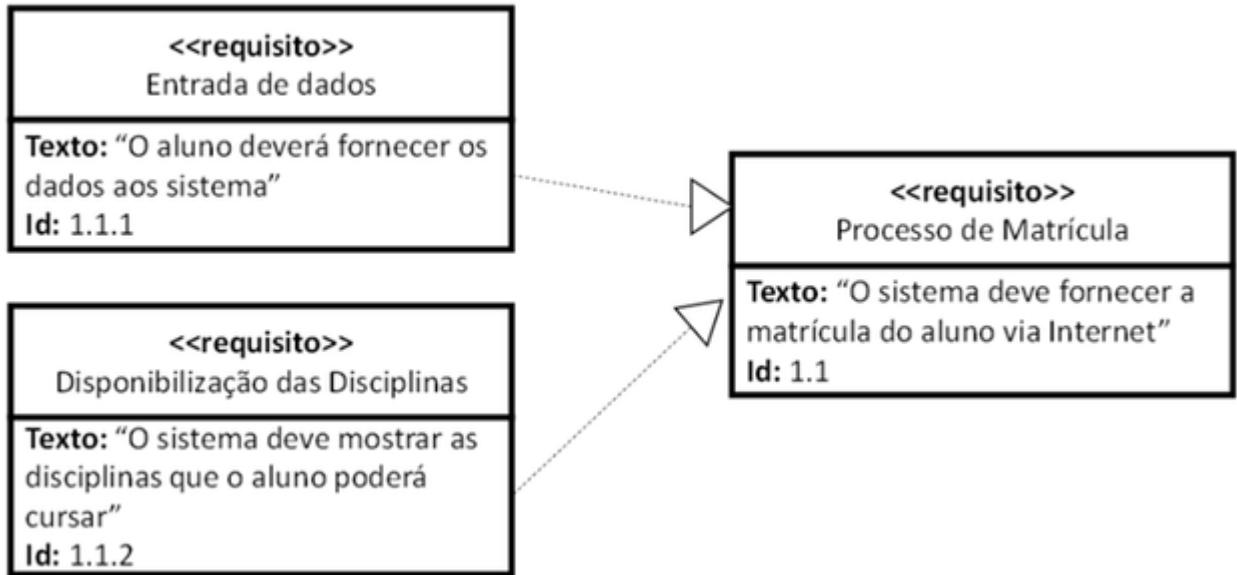


Figura 2: Diagrama de requisitos. Fonte: Werlich (2020, p. 151).

A UML é amplamente empregada na modelagem de requisitos. A Linguagem UML (Linguagem de Modelagem Unificada) oferece uma variedade de diagramas que podem ser utilizados de forma isolada, representando aspectos específicos, ou combinados para compor modelos mais abrangentes. Por exemplo, o **caso de uso** é um dos diagramas mais utilizados para modelar requisitos, juntamente com o **diagrama de sequência** e o **diagrama de atividades**. Cada empresa pode adotar uma ou várias técnicas de modelagem de requisitos para produzir a documentação necessária ao final do processo de modelagem.

O processo de modelagem de requisitos na análise de sistemas pode ser considerado uma conexão entre as fases de elicitação e especificação de requisitos e a fase de modelagem de projetos do sistema. Uma alternativa adicional para especificar requisitos é através de **diagramas de caso de uso**. Pressman (2021) destaca que um **cenário de uso**, ou especificação do **caso de uso**, detalha o **caso de uso**, auxiliando na modelagem do requisito específico. Este detalhamento serve como principal meio de comunicação entre a equipe de desenvolvimento, ou seja, entre o analista de sistemas responsável pela modelagem e o programador encarregado de implementar o requisito modelado.

O diagrama de caso de uso desempenha um papel fundamental na construção de software orientado a objetos utilizando a UML. Estes diagramas acompanham o software desde sua concepção até sua conclusão. Além disso, eles servem como um meio de comunicação eficaz entre o analista de sistemas e os programadores, pois detalham precisamente o que precisa ser implementado e codificado.

Na Figura 3 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,

ANÁLISE E MODELAGEM DE SISTEMAS

e envolve dois atores que “alimentarão” os processos com o fornecimento ou o recebimento de informações.

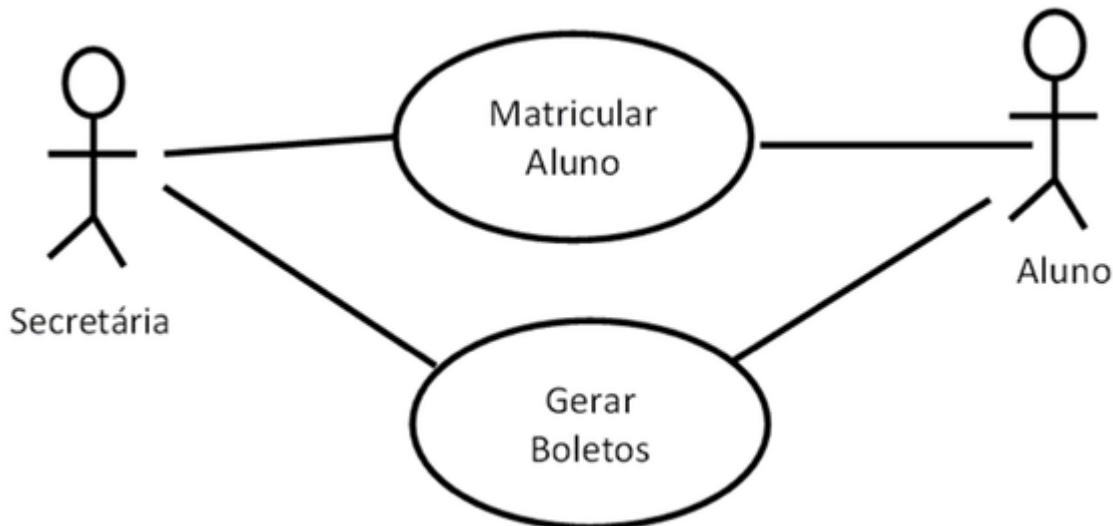


Figura 3 | Exemplo de casos de uso. Fonte: Werlich (2020, p. 158).

Cada caso de uso representa uma determinada função ou tarefa do sistema, segundo Sommerville (2018), 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, com a especificação do caso de uso Matricular Aluno.

Nome do caso de uso	Matricular Aluno
Ator 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

ANÁLISE E MODELAGEM DE SISTEMAS

1. O aluno informa a matrícula e senha para se logar 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 1.1 O aluno poderá redefinir a sua senha.	Ações do Sistema 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).

Quadro 3 | Especificação do caso de uso: Matricular Aluno. Fonte: adaptado de Werlich (2020, p. 159).

A finalidade da especificação ou descrição de um caso de uso é fornecer informações sobre quais atores (sejam pessoas ou sistemas) interagem com as funcionalidades específicas que estão sendo modeladas no sistema. Não existe um modelo padrão de especificação, pois este é totalmente adaptável às necessidades das empresas. No entanto, é recomendável que seja redigido de forma simplificada para facilitar a compreensão.

Vamos Exercitar?

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

ANÁLISE E MODELAGEM DE SISTEMAS

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 é criar uma Documentação da Especificação de Requisitos e, para isso, primeiramente resgataremos os Requisitos Funcionais:

- [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 (incluir, consultar, alterar e excluir) os dados dos clientes: nome, endereço, CPF ou CNPJ, e-mail, telefones.
- [RF0011] - O sistema deve gerar um relatório das Plantas disponíveis para venda.

Será possível usar uma padronização para documentar os requisitos? Existe alguma vantagem ao utilizar essa técnica? Sim, podemos utilizar o padrão de modelagem apresentado a seguir, no Quadro 4 ou utilizar outro apresentado nesta aula.

Identificador:			
Nome:			
Módulo:			
Data de criação:	Autor:		
Data última alteração:	Autor:		
Versão:	Prioridade:		
Descrição:			

Quadro 4 | Padrão de modelagem. Fonte: adaptado de Werlich (2020, p. 160).

E como ficaria a documentação do Requisito Funcional: [RF0008] – O sistema deverá manter o cadastro de todos os fornecedores? Observe o quadro 5 que ilustra a documentação básica do requisito RF008.

Identificador:	FR0008		
Nome:	O sistema deverá manter o cadastro de todos os fornecedores		
Módulo:	N/A		
Data de 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		

ANÁLISE E MODELAGEM DE SISTEMAS

dados que deverão ser informados são: Nome fantasia, Razão social, CNPJ, Inscrição Estadual, Endereço completo, Cidade, Estado, E-mail.

Quadro 5 | Documentação básica do requisito RF008. Fonte: adaptado de Werlich (2020, p. 160).

Utilize o modelo de modelagem usado anteriormente para documentar os demais 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.

Saiba mais

O livro de Pressman oferece insights valiosos sobre a Modelagem de Requisitos, portanto acesse o Capítulo 8 do livro *Engenharia de Software – Pressman* e leia mais sobre o tema:

Pressman, R. S.; MAXIM B. R. [Engenharia de software](#). (9th edição). Grupo A, 2021.

Para saber mais sobre a Especificação de Requisitos, leia o Capítulo 4.4 de *Engenharia de Software – Sommerville*:

SOMMERVILLE, I. [Engenharia de software](#). 10. ed. São Paulo, SP: Pearson, 2018.

Para entender mais sobre a Modelagem de Requisitos, utilizando Casos de Uso, leia o Capítulo 5 do livro *Engenharia de requisitos – Reinehr*:

REINEHR, S. [Engenharia de requisitos](#). Grupo A, 2020.

Referências

PFLEEGER, S. L. **Engenharia de Software**: teoria e prática. 2. ed. São Paulo: Prentice Hall, 2004.

PRESSMAN, Roger S. **Engenharia de software**: uma abordagem profissional. 9. ed. – Porto Alegre: AMGH, 2021.

SOMMERVILLE, Ian. **Engenharia de software**. 10. ed. São Paulo: Pearson, 2018.

VIEIRA, S. R. C. **Remo**: uma técnica de elicitação de requisitos orientada pela modelagem de processos de negócios. 2012.

ANÁLISE E MODELAGEM DE SISTEMAS

WERLICH, C. **Análise e modelagem de sistemas**. Londrina: Editora e Distribuidora Educacional S.A., 2020.

Aula 4

Validação e modificações nos Requisitos



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Bem-vindo à videoaula dedicada a Validação, Mudanças e Rastreabilidade de Requisitos. Esses conteúdos são fundamentais para sua prática profissional na engenharia de requisitos. A validação assegura que os requisitos atendam às necessidades dos clientes, enquanto o gerenciamento de mudanças garante a adaptabilidade do sistema. Além disso, a rastreabilidade permite acompanhar e verificar a evolução dos requisitos ao longo do ciclo de vida do projeto. Prepare-se para adquirir habilidades essenciais para o sucesso na análise e desenvolvimento de sistemas de software.

[Clique aqui](#) para acessar os slides da sua videoaula.

Bons estudos!

Ponto de Partida

Nesta aula, iremos abordar três aspectos cruciais no desenvolvimento de software: validação de requisitos, gestão de mudanças nos requisitos e rastreabilidade de requisitos. Primeiramente, nos concentraremos na validação de requisitos, visando garantir que esses requisitos estejam alinhados com as necessidades dos usuários e do negócio. Usaremos técnicas como revisões sistemáticas e prototipagem para identificar e corrigir problemas antes do desenvolvimento do produto final.

Em seguida, exploraremos a gestão de mudanças nos requisitos, aprendendo a lidar com alterações imprevistas durante o processo de desenvolvimento. Implementaremos processos claros para solicitar, analisar e implementar essas mudanças de maneira eficiente, com técnicas

ANÁLISE E MODELAGEM DE SISTEMAS

para avaliar seu impacto e comunicá-las de forma eficaz. Por último, adentraremos no conceito de rastreabilidade de requisitos, que nos permite rastrear o desenvolvimento dos requisitos desde sua concepção até sua implementação.

Para exercitar esses temas, vamos analisar o seguinte estudo de caso: uma loja está desenvolvendo um novo sistema de comércio eletrônico para substituir seu sistema legado. O objetivo é criar uma plataforma mais robusta e escalável para lidar com o crescimento do negócio e proporcionar uma melhor experiência ao cliente. O projeto envolve uma equipe multidisciplinar de desenvolvedores, designers, testadores e analistas de negócios.

Um dos principais desafios enfrentados pela equipe de desenvolvimento é lidar eficientemente com mudanças nos requisitos do sistema. Como o mercado de comércio eletrônico é altamente dinâmico, novos requisitos e alterações nos requisitos existentes são esperados ao longo do ciclo de vida do projeto. No entanto, a falta de um processo formal para gerenciar essas mudanças pode levar a atrasos, custos adicionais e, em última instância, a um produto final que não atende às expectativas do cliente. Proponha soluções que possam ajudar neste desafio.

Vamos Começar!

Validação de requisitos

A validação de requisitos é um processo crucial para garantir que os requisitos definidos correspondam verdadeiramente ao sistema desejado pelo cliente. Este processo se destaca durante a elicitação e análise, concentrando-se na identificação de possíveis problemas. Sua importância é primordial, pois erros nos requisitos podem resultar em custos significativos de retrabalho, tanto durante o desenvolvimento quanto após a implementação do sistema (PRESSMAN, 2021).

O custo de corrigir problemas nos requisitos, por meio de alterações no sistema, geralmente é consideravelmente maior do que corrigir erros de projeto ou de código. Uma mudança nos requisitos frequentemente implica em ajustes no projeto e na implementação do sistema, muitas vezes exigindo um reinício do processo.

Durante a validação de requisitos, é essencial realizar diferentes tipos de verificações nos requisitos documentados, incluindo (SOMMERVILLE, 2018):

- Validade:** verificar se os requisitos refletem as reais necessidades dos usuários do sistema, considerando possíveis mudanças nas circunstâncias desde a elicitação inicial.
- Consistência:** garantir que os requisitos no documento não entrem em conflito entre si, evitando restrições contraditórias ou descrições diferentes para a mesma função do sistema.

ANÁLISE E MODELAGEM DE SISTEMAS

3. **Completude:** certificar-se de que o documento de requisitos abrange todas as funções e restrições desejadas pelo usuário do sistema.
4. **Realismo:** avaliar se os requisitos podem ser implementados dentro do orçamento proposto, considerando as tecnologias existentes e o cronograma de desenvolvimento.
5. **Verificabilidade:** para diminuir o potencial de conflito entre o cliente e o contratante, os requisitos do sistema sempre devem ser escritos de modo que sejam verificáveis. Isso significa ser capaz de escrever um conjunto de testes que possam demonstrar que o sistema entregue satisfaz cada um dos requisitos especificados.

Diversas técnicas de validação de requisitos podem ser empregadas para garantir a qualidade dos requisitos de um sistema:

1. **Revisões de requisitos:** uma equipe de revisores examina os requisitos em busca de erros e inconsistências de forma sistemática.
2. **Prototipação:** desenvolvimento de um modelo executável do sistema para que os usuários finais e clientes possam experimentá-lo, fornecendo feedback para possíveis mudanças nos requisitos.
3. **Geração de casos de teste:** os requisitos devem ser testáveis, e o desenvolvimento de casos de teste revela problemas potenciais nos requisitos. Se um teste é difícil de conceber, pode indicar que os requisitos são complexos e precisam ser reconsiderados.

A validação de requisitos é um desafio, pois é difícil demonstrar que um conjunto de requisitos atende verdadeiramente às necessidades dos usuários. Muitas vezes, problemas nos requisitos só são descobertos após o início da implementação do sistema, exigindo ajustes mesmo após o consenso inicial no documento de requisitos.

Gestão de mudanças nos requisitos

Os requisitos de sistemas de software grandes estão sujeitos a mudanças frequentes devido à natureza dos problemas que esses sistemas abordam, muitas vezes complexos e difíceis de definir completamente. Como esses problemas são intrinsecamente desafiadores, os requisitos de software tendem a ser incompletos, pois não é possível uma definição abrangente. Durante o processo de desenvolvimento de software, a compreensão do problema pelos stakeholders está em constante evolução, o que exige uma adaptação contínua dos requisitos do sistema para refletir essa compreensão em mudança.

Após a instalação do sistema, é comum surgirem novos requisitos devido a erros, mudanças no ambiente de negócios e tecnológico, e às diferentes necessidades dos stakeholders. Os ambientes de negócios e tecnológicos estão em constante evolução, exigindo adaptações no sistema. Os clientes e usuários finais podem impor requisitos conflitantes, e a diversidade de stakeholders pode levar a prioridades divergentes, exigindo uma conciliação. O gerenciamento de requisitos é crucial para acompanhar as mudanças e garantir que os requisitos sejam priorizados e vinculados adequadamente. Enquanto os processos ágeis de desenvolvimento podem lidar

ANÁLISE E MODELAGEM DE SISTEMAS

com mudanças de requisitos de forma mais flexível, é importante garantir uma abordagem sistemática para gerenciar as alterações e manter a coerência entre os requisitos do sistema.

O gerenciamento mudança de requisitos é fundamental para avaliar a viabilidade e o impacto das mudanças propostas após a aprovação do documento de requisitos de um sistema. Um processo formal de gerenciamento de mudança garante que todas as propostas sejam tratadas de forma consistente e controlada. Este processo envolve três etapas principais, que estão apresentadas na Figura 1.

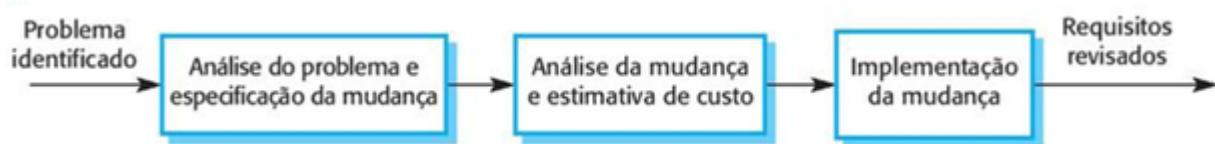


Figura 1 | Gerenciamento de mudança dos requisitos. Fonte: Sommerville (2018, p. 116).

O detalhamento do processo de gerenciamento de mudanças em requisitos segue três etapas fundamentais (SOMMERVILLE, 2018):

- 1. Análise do problema e especificação da mudança:** inicia-se com a identificação de um problema de requisito ou uma proposta de mudança específica. Durante essa fase, o problema ou proposta é analisado para determinar sua validade, e o requisitante da mudança pode fornecer uma proposta mais específica ou optar por abandonar a solicitação.
- 2. Análise da mudança e estimativa de custo:** o impacto da mudança proposta é avaliado com base na rastreabilidade das informações e no conhecimento geral dos requisitos do sistema. Uma estimativa de custo é feita, considerando as modificações necessárias nos documentos de requisitos, no projeto e na implementação do sistema. Após a análise, uma decisão é tomada sobre a execução da mudança.
- 3. Implementação da mudança:** os documentos de requisitos, e se aplicável, de projeto e implementação do sistema, são ajustados conforme necessário. É importante organizar o documento de requisitos de forma a facilitar futuras mudanças, minimizando referências externas e tornando as seções modulares para permitir alterações sem afetar outras partes do documento.

Quando surge a necessidade de implementar um novo requisito com urgência, há uma tentação de modificar o sistema primeiro e, em seguida, ajustar retrospectivamente o documento de requisitos. No entanto, isso frequentemente resulta em uma discrepância entre a especificação dos requisitos e a implementação do sistema. Após as mudanças serem feitas, é fácil esquecer de atualizar o documento de requisitos para refletir essas alterações. Embora em certas circunstâncias seja necessário realizar mudanças emergenciais no sistema, é crucial atualizar o documento de requisitos o mais rápido possível para incluir os requisitos revisados.

ANÁLISE E MODELAGEM DE SISTEMAS

Siga em Frente...

Rastreabilidade de requisitos

Para realizar uma análise de impacto de mudança de maneira eficaz, é crucial estabelecer e compreender as conexões entre os requisitos, bem como entre esses requisitos e outros elementos do sistema. Essas conexões, conhecidas como rastreabilidade de requisitos, são fundamentais para embasar essa análise. A rastreabilidade implica rastrear a origem de um requisito, o que é conhecido como rastreabilidade para trás (*backward traceability*), bem como acompanhar suas ramificações, denominadas rastreabilidade para frente (*forward traceability*). Essa rastreabilidade vertical é essencial. Além disso, há a rastreabilidade entre os próprios requisitos, que identifica as interdependências entre eles, denominada rastreabilidade horizontal. Geralmente, a rastreabilidade é implementada por meio de ferramentas de mapeamento de grafos ou matrizes bidimensionais.

Qual é a importância da rastreabilidade? Essa técnica nos permite o identificar os seguintes pontos com relação aos requisitos:

- **Identificação de requisitos ausentes:** consiste em buscar requisitos de negócios não rastreados até requisitos de usuário e requisitos de usuário não vinculados a nenhum requisito funcional. Essa abordagem permite detectar lacunas que podem ter sido negligenciadas durante o processo de definição dos requisitos.
- **Eliminação de requisitos desnecessários:** envolve a busca por requisitos funcionais que não possuem rastreabilidade até os requisitos de usuário e de negócios, indicando sua possível desnecessidade.
- **Certificação e conformidade:** a rastreabilidade fornece informações valiosas durante processos de certificação de produtos críticos para a segurança, permitindo demonstrar a implementação de todos os requisitos, embora não garanta sua correta implementação. Além disso, é útil para demonstrar a inclusão e tratamento de requisitos relacionados à conformidade regulatória, especialmente em setores como saúde e finanças.
- **Análise de impacto de mudanças:** a ausência de informações de rastreabilidade aumenta o risco de ignorar elementos do sistema que seriam afetados por adições, remoções ou modificações de requisitos específicos.
- **Manutenção de software:** a rastreabilidade facilita a capacidade de efetuar mudanças de forma precisa e completa durante a manutenção do software. À medida que políticas corporativas ou regulamentações governamentais evoluem, sistemas de software frequentemente precisam ser atualizados. Uma tabela que indica onde cada regra de negócio relevante foi abordada nos requisitos funcionais, nos projetos e no código simplifica a implementação das mudanças necessárias. É possível identificar relações de causa e efeito de falhas, ajudando a determinar quais componentes precisam ser alterados e fornecendo suporte para estimativas de mudança.
- **Executar testes:** quando um teste falha, as ligações entre os testes, os requisitos e o código apontam os desenvolvedores para as áreas prováveis a serem examinadas para o defeito.

ANÁLISE E MODELAGEM DE SISTEMAS

A rastreabilidade apoia, portanto, as correções de defeitos encontrados em testes.

Manter a rastreabilidade de todos os elementos de um projeto é desafiador e pode ser dispendioso. É essencial priorizar e manter apenas os relacionamentos críticos para o desenvolvimento e manutenção do software. Isso requer uma decisão consciente por parte do time de desenvolvimento e stakeholders, semelhante à gerência de configuração. Itens como origem dos requisitos, requisitos críticos para o negócio, impacto na arquitetura e testes, bem como código e casos de teste cruciais para o sistema, são candidatos a serem rastreados. Essa abordagem ajuda a minimizar o risco de análises de impacto imprecisas e garante uma visão mais clara sobre a abrangência das mudanças no software.

A representação mais comum da rastreabilidade de requisitos é no formato de matriz, que pode ser implementada usando-se planilha eletrônica ou ferramenta automatizada. Uma implementação eficiente de uma rastreabilidade mais robusta deve ser realizada por meio de ferramentas automatizadas. Existem ferramentas de modelagem UML, por exemplo, que já oferecem formas de registrar e manter a rastreabilidade dos requisitos e dos demais artefatos do ciclo de desenvolvimento de software.

A Figura 2 apresenta um modelo de matriz bidimensional que representa a rastreabilidade de requisitos.

ANÁLISE E MODELAGEM DE SISTEMAS

	Requisitos funcionais		Requisitos não funcionais			Casos de Uso		Casos de Teste		Módulos					
	RF01	(...)	RF20	RNF1	RNF2	RNF3	UC1	(...)	UC12	CT1.1	(...)	CT20.1	M1	(...)	M54
Gerente de Vendas			X			X									
Diretor de Vendas	X														
Vendedor															
OE1	X														
OE2															
OE3			X												
RF01					X	X				X					X
(...)															
RF20						X		X			X	X			
RNF1															
RNF2															
RNF3															
UC1															
(...)															
UC12															
CT1.1															
(...)															
CT20.1															
M1															
(...)															
M54															

Figura 2 | Exemplo de matriz de rastreabilidade de requisitos usando planilha eletrônica. Fonte: Reinehr (2020, p. 79).

Na primeira coluna, listamos todos os elementos que poderão ser rastreados entre si. Nesse exemplo, listamos os stakeholders, os objetivos estratégicos, os requisitos funcionais, os requisitos não funcionais, os casos de uso, os casos de teste e os módulos de código.

Na primeira linha listamos os mesmos elementos. Note que não foram listados na linha 1 os stakeholders e nem os objetivos estratégicos, pois não temos interesse em rastrear o relacionamento entre eles, mas sim com os requisitos que eles geraram (funcionais e não funcionais) (REINEHR, 2020).

Poderíamos ter mapeado todos os relacionamentos do Gerente de vendas com os elementos que foram desdobrados a partir do RF20, mas isso iria poluir desnecessariamente a planilha. Então assumimos que apenas nos interessa mapear quais foram os requisitos solicitados por cada stakeholder, ou seja, o único mapeamento possível na nossa planilha é entre o stakeholder solicitante e os requisitos funcionais e não funcionais (únicas células com o fundo branco). Raciocínio similar foi utilizado para a representação dos objetivos estratégicos. Note que a interseção dos stakeholders e dos objetivos estratégicos com os elementos Casos de Uso, Casos de Teste e Módulos foram suprimidos (células em cinza). Podemos observar que estão

ANÁLISE E MODELAGEM DE SISTEMAS

marcados com um “X” todos os relacionamentos entre os elementos. A parte inferior da tabela está em cinza, denotando que são relacionamentos já representados na parte superior (REINEHR, 2020).

Em destaque verde temos a representação dos relacionamentos existentes com o requisito funcional RF20. Vemos que ele foi solicitado pelo Gerente de vendas e que se relaciona com o objetivo estratégico OE3. Esta é a chamada rastreabilidade para trás, ou seja, para a origem do requisito. Logo em seguida temos a rastreabilidade para frente, ou seja, quais foram os artefatos que foram gerados a partir do requisito funcional RF20. Temos então o caso de uso UC12, o caso de teste CT20.1 e o módulo implementado M1. Temos mais um relacionamento representado, que é o relacionamento com o requisito não funcional RNF3. Esta é a rastreabilidade horizontal entre elementos do mesmo tipo (no caso, requisitos) (REINEHR, 2020).

Manter a rastreabilidade é importante para o software, especialmente durante a manutenção. É essencial encontrar um equilíbrio na quantidade de informações rastreadas para garantir sua atualização constante. Pior do que não ter uma informação, nesse caso, é ter uma informação incompleta ou incorreta.

Vamos Exercitar?

A equipe de gerenciamento de projetos de uma loja decidiu implementar um sistema de gerenciamento de mudanças de requisitos e rastreabilidade. Eles optaram por utilizar uma ferramenta de gerenciamento de requisitos que permitisse a captura, documentação e rastreamento de todos os requisitos do sistema, bem como suas interdependências.

Passos implementados:

- **Identificação de requisitos:** a equipe trabalhou em estreita colaboração com os stakeholders para identificar e documentar todos os requisitos do sistema, incluindo requisitos funcionais, não funcionais e de negócios.
- **Priorização de requisitos:** os requisitos foram priorizados com base na sua importância para o sucesso do projeto e seu impacto no cronograma e no orçamento.
- **Estabelecimento de um processo de mudança:** foi definido um processo formal para solicitação, avaliação, aprovação e implementação de mudanças nos requisitos. Isso incluiu a designação de um comitê de controle de mudanças responsável por revisar e aprovar todas as solicitações de mudança.
- **Rastreabilidade de requisitos:** a ferramenta de gerenciamento de requisitos foi configurada para rastrear as relações entre os requisitos, permitindo à equipe entender como uma mudança em um requisito afeta outros requisitos e componentes do sistema.

Resultados:

ANÁLISE E MODELAGEM DE SISTEMAS

A implementação do sistema de gerenciamento de mudanças de requisitos e rastreabilidade teve vários benefícios significativos:

- **Redução de atrasos:** a equipe conseguiu lidar com mudanças nos requisitos de forma mais eficiente, evitando atrasos no cronograma do projeto.
- **Controle de custos:** o processo de controle de mudanças ajudou a evitar custos adicionais associados a mudanças de escopo não gerenciadas.
- **Maior satisfação do cliente:** a capacidade de rastrear requisitos e suas interdependências resultou em um produto final que atendeu melhor às expectativas do cliente.
- **Melhoria da qualidade:** o sistema de rastreabilidade permitiu à equipe identificar e resolver mais facilmente inconsistências e conflitos nos requisitos, resultando em um produto final de maior qualidade.

Conclusão:

A implementação de um sistema de gerenciamento de mudanças de requisitos e rastreabilidade foi fundamental para o sucesso do projeto de desenvolvimento de software loja. A abordagem estruturada para lidar com mudanças nos requisitos ajudou a equipe a manter o controle sobre o escopo do projeto, evitar atrasos e custos adicionais, e fornecer um produto final que atendeu às expectativas do cliente. Este caso destaca a importância de uma abordagem sistemática para gerenciar mudanças em projetos de desenvolvimento de software.

Saiba mais

Quer saber mais sobre o Gerenciamento e controle de mudanças nos requisitos? Acesse o artigo *Gerenciamento e controle de mudanças de requisitos* e leia mais sobre o tema:

DEVMEDIA. [Gerenciamento e controle de mudanças de requisitos](#). [s. d.].

Para saber mais sobre as Mudanças em requisitos, leia o Capítulo 4.6 do *livro Engenharia de Software* - Sommerville.

SOMMERVILLE, I. [Engenharia de software](#). 10. ed. São Paulo, SP: Pearson, 2018

Para entender mais sobre a Rastreabilidade de Requisitos, leia o Capítulo 13 do livro *Engenharia de requisitos* – Reinehr.

REINEHR, S. [Engenharia de requisitos](#). Grupo A, 2020.

Referências

ANÁLISE E MODELAGEM DE SISTEMAS

PRESSMAN, R. S. **Engenharia de software**: uma abordagem profissional. 9. ed. – Porto Alegre: AMGH, 2021.

REINEHR, S. **Engenharia de requisitos**. [Digite o Local da Editora]: Grupo A, 2020.

SOMMERVILLE, I. **Engenharia de software**. 10. ed. São Paulo: Pearson, 2018.

Aula 5

Encerramento da Unidade

Videoaula de Encerramento



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Olá, estudante! Nesta videoaula, você conhecerá a Engenharia de requisitos! Nesta aula, vamos explorar os fundamentos dos tipos de requisitos e a importância da modelagem. Esses conteúdos são essenciais para sua prática profissional, pois garantem uma compreensão sólida das necessidades dos clientes e a entrega eficaz de soluções de software. Prepare-se para adquirir habilidades valiosas que impulsionarão sua carreira na área de desenvolvimento de software.

[Clique aqui](#) para acessar os slides da sua videoaula.

Bons estudos!

Ponto de Chegada

Olá, estudante! Para desenvolver a competência desta Unidade, que é identificar os tipos de requisitos e suas características para a modelagem de sistemas, você deverá primeiramente conhecer os fundamentos da Engenharia de requisitos. A engenharia de requisitos abrange a descrição detalhada das funcionalidades, serviços e restrições de um sistema, refletindo as necessidades dos usuários e contribuindo para um produto de qualidade. Os requisitos de um sistema vão além de funções e qualidades, incluindo especificações de serviços, restrições e características gerais. Exemplos concretos de requisitos para um jogo de perguntas e respostas

ANÁLISE E MODELAGEM DE SISTEMAS

ilustram a importância de redigir requisitos de forma clara e concisa, garantindo compreensão mútua entre cliente e desenvolvedores.

Na elaboração dos requisitos, é essencial determinar suas prioridades, que podem ser classificadas como essencial, importante ou desejável. Requisitos essenciais são vitais para a completude do software, enquanto os importantes são secundários, mas ainda relevantes, e os desejáveis são opcionais. Essa categorização permite priorizar o desenvolvimento e atualização do software de acordo com as necessidades do projeto, embora as prioridades possam mudar ao longo da evolução do sistema.

Na engenharia de requisitos, é crucial distinguir entre os tipos de descrições de requisitos, conforme afirmado por Sommerville (2018). Os requisitos do usuário delineiam as necessidades dos clientes de forma comprehensível, abordando tanto os requisitos funcionais quanto os não funcionais do sistema. Por outro lado, os requisitos do sistema especificam os detalhes técnicos do sistema com base nos requisitos do usuário, podendo servir como parte de um contrato entre as partes envolvidas no desenvolvimento.

A categorização dos requisitos em um sistema, conforme descrito por Sommerville (2018) e Pressman (2021), engloba três principais categorias: requisitos funcionais, não funcionais e de domínio. Os requisitos funcionais definem as funcionalidades específicas que o sistema deve executar, enquanto os não funcionais estabelecem restrições e padrões relacionados ao desempenho, segurança e usabilidade. Já os requisitos de domínio descrevem as características específicas do ambiente em que o sistema será utilizado, impondo condições adicionais aos requisitos funcionais.

A elaboração detalhada dos requisitos desempenha um papel crucial na comunicação com o cliente e na orientação do desenvolvimento do sistema. Os requisitos funcionais são identificados por uma sigla específica seguida de uma numeração, enquanto os não funcionais são categorizados e mensurados de acordo com diferentes métricas. Os requisitos de domínio complementam os requisitos funcionais, estabelecendo critérios específicos para validar determinadas funcionalidades do sistema.

Outro ponto fundamental para alcançar a competência da unidade, é entender o conjunto de atividades da engenharia de requisitos, que envolvem desde a coleta inicial de informações sobre o sistema a ser desenvolvido até a validação dos requisitos pelo usuário final. Essas atividades, conforme descritas por Pressman (2021) e Sommerville (2018), incluem concepção, elicitação, elaboração, negociação, especificação, validação e gerenciamento de requisitos. Cada etapa tem como objetivo primordial a elaboração de um documento de requisitos detalhado e preciso, que servirá como base para o desenvolvimento do sistema.

O processo de engenharia de requisitos compreende uma série de etapas interligadas. Desde a definição do escopo geral do sistema até o gerenciamento contínuo dos requisitos ao longo do ciclo de vida do produto, todas as atividades visam garantir a compreensão das necessidades do cliente e a conformidade dos requisitos com o escopo do projeto.

ANÁLISE E MODELAGEM DE SISTEMAS

Em suma, os requisitos funcionais delineiam as funcionalidades e características específicas do sistema, enquanto os requisitos não funcionais estabelecem atributos e qualidades do sistema, como desempenho e segurança. Esses requisitos são essenciais para orientar o desenvolvimento de software e garantir a entrega de um produto que atenda às expectativas do cliente e dos usuários finais. O processo de engenharia de requisitos é dinâmico e envolve a participação ativa do cliente e dos usuários ao longo de todo o ciclo de desenvolvimento do sistema.

Na engenharia de requisitos, a elicitação de requisitos desempenha um papel central na definição das necessidades de um sistema específico, envolvendo a coleta e análise de informações a partir de diversas fontes e interações com stakeholders relevantes, conforme destacado por Pressman (2021). Essa fase requer a colaboração de várias partes interessadas, incluindo analistas de sistemas, gerentes de projetos e usuários finais do sistema, como mencionado por Sommerville (2018), visando resolver problemas de forma colaborativa.

A elicitação de requisitos busca estabelecer uma compreensão clara do escopo do sistema, identificar oportunidades de reutilização de soluções existentes e eliciar requisitos funcionais e não funcionais. A partir daí, ocorre a negociação e validação dos requisitos, garantindo que não haja conflitos entre eles e que atendam às necessidades dos stakeholders. O resultado é a documentação detalhada dos requisitos, que serve como principal meio de comunicação entre os desenvolvedores e o cliente.

O processo de levantamento e análise de requisitos é uma sequência contínua e iterativa de atividades que visam validar as necessidades do sistema. A figura apresentada demonstra a progressão desse processo, desde a compreensão do domínio do problema até a documentação final dos requisitos. A validação contínua dos requisitos é essencial, pois qualquer alteração requer revisão e ajustes, garantindo que o sistema atenda efetivamente às expectativas do cliente e usuários finais (SOMMERVILLE, 2018).

O processo de elicitação de requisitos pode variar entre as empresas, envolvendo atividades como descoberta, classificação, priorização e especificação de requisitos. Essa fase é colaborativa e interativa, com a participação ativa dos stakeholders na identificação das necessidades do sistema, utilizando técnicas como pesquisa, entrevistas, reuniões e análise documental para obter uma visão abrangente do problema a ser resolvido.

A especificação de requisitos consiste na formalização das funcionalidades que o sistema deve executar, seguindo uma abordagem hierárquica para detalhar progressivamente os requisitos. Essa etapa é fundamental para criar um documento claro e objetivo que servirá como base para o desenvolvimento do software, garantindo que os requisitos sejam compreendidos de forma unívoca pelos desenvolvedores e stakeholders (PFLEGER, 2004).

Além disso, a prototipagem é uma técnica valiosa na engenharia de requisitos, permitindo a criação de versões simplificadas do sistema para avaliar sua viabilidade e identificar problemas precocemente. A negociação de requisitos também é essencial para desenvolver um plano de

ANÁLISE E MODELAGEM DE SISTEMAS

projeto que atenda às demandas dos stakeholders, considerando as restrições de orçamento, pessoal, tecnologia e tempo impostas à equipe de desenvolvimento do software (PAULA FILHO, 2019).

O monitoramento de requisitos é crucial para garantir que o escopo do software seja atendido, exigindo a rastreabilidade das alterações através de ferramentas de controle apropriadas. Ao modificar um requisito, é essencial atribuir um status apropriado e manter uma matriz de rastreabilidade atualizada, incluindo todas as informações relevantes sobre cada requisito. A gestão de mudanças de requisitos visa analisar o impacto das alterações propostas para determinar sua viabilidade técnica e financeira, evitando inconsistências decorrentes de documentação desatualizada.

A validação de requisitos é fundamental para garantir que a especificação seja consistente com as necessidades do cliente. Durante esse processo, diferentes tipos de verificação, como validade, consistência, completude e verificabilidade, são realizados para identificar falhas nos requisitos documentados. O propósito é eliminar erros comuns, como inconsistências, contradições e ambiguidades, visando evitar atrasos e aumentos de custos durante o desenvolvimento do software (PFLEGER, 2004).

Um recurso útil para a validação de requisitos é o uso de checklists, listas de perguntas elaboradas para analisar cada requisito do sistema. Essa técnica visa identificar erros em vários níveis e garantir que o software desenvolvido atenda aos padrões estabelecidos, contribuindo para a qualidade do processo de validação de requisitos.

Outro ponto importante é a Modelagem de requisitos, que se concentra no "o que será feito" em vez do "como será feito", com o objetivo de descrever as necessidades do cliente, fornecer uma base para o desenvolvimento do software e produzir requisitos válidos (PRESSMAN, 2021). Para compreender completamente os requisitos, é crucial considerar não apenas o fluxo de informações no sistema, mas também as transformações de dados e todas as restrições de comportamento e desempenho. As técnicas de modelagem de requisitos incluem a separação entre requisitos funcionais e não funcionais, agrupamento dos requisitos por prioridade e a especificação detalhada em documentos formais.

A redação dos requisitos em linguagem natural é uma prática comum, mas pode levar a interpretações diversas e inconsistências. Para mitigar isso, são sugeridas diretrizes como estabelecer um formato padrão, destacar partes importantes e associar um racional a cada requisito, conforme recomendado por Sommerville (2018). Além disso, a linguagem estruturada e o uso de modelos e diagramas, como os fornecidos pela Linguagem de Modelagem Unificada (UML) e SysML, são úteis para organizar os requisitos de forma mais clara e eficiente.

Diversas técnicas de modelagem de requisitos são empregadas, incluindo a técnica REMO para extração de requisitos a partir de diagramas de processos de negócios e a utilização de diagramas de caso de uso na UML para detalhar as interações entre os atores e o sistema. A

ANÁLISE E MODELAGEM DE SISTEMAS

especificação de cada caso de uso é essencial para comunicar as funcionalidades específicas a serem implementadas, geralmente redigida de forma simplificada para facilitar a compreensão.

A validação de requisitos é um passo importante para garantir a correspondência entre os requisitos definidos e o sistema desejado pelo cliente. Realizada durante a elicitação e análise, visa identificar problemas que poderiam resultar em custos significativos de retrabalho durante o desenvolvimento ou após a implementação do sistema (PRESSMAN, 2021). Durante esse processo, são realizadas diversas verificações nos requisitos documentados, incluindo validade, consistência, completude, realismo e verificabilidade (SOMMERVILLE, 2018).

Para assegurar a qualidade dos requisitos, são empregadas diferentes técnicas de validação. As revisões de requisitos envolvem uma equipe de revisores que examina sistematicamente os requisitos em busca de erros e inconsistências. A prototipagem permite o desenvolvimento de um modelo executável do sistema para que os usuários finais possam fornecer feedback, enquanto a geração de casos de teste revela problemas potenciais nos requisitos ao torná-los testáveis (PRESSMAN, 2021).

No entanto, a validação de requisitos é um desafio, pois é difícil demonstrar que um conjunto de requisitos atende verdadeiramente às necessidades dos usuários. Muitas vezes, problemas só são descobertos após o início da implementação do sistema, exigindo ajustes mesmo após o consenso inicial no documento de requisitos.

Os sistemas de software de grande porte enfrentam desafios recorrentes devido à complexidade inherente dos problemas que abordam, muitas vezes resultando em requisitos incompletos devido à dificuldade em defini-los integralmente. Ao longo do processo de desenvolvimento de software, a compreensão do problema pelos stakeholders está em constante evolução, exigindo uma adaptação contínua dos requisitos do sistema para refletir essa mudança de compreensão. Após a implementação do sistema, novos requisitos podem surgir devido a erros, mudanças no ambiente de negócios e tecnológico, bem como diferentes necessidades dos stakeholders, demandando uma gestão eficaz de requisitos para acompanhar essas mudanças e garantir a coesão entre os requisitos do sistema.

O gerenciamento de mudança de requisitos desempenha um papel crucial na avaliação da viabilidade e do impacto das mudanças propostas após a aprovação do documento de requisitos de um sistema. Esse processo inclui a análise do problema e a especificação da mudança, a avaliação do impacto e a estimativa de custo da mudança, e a subsequente implementação da mudança. É imperativo atualizar o documento de requisitos após as mudanças serem realizadas para manter a coesão entre a especificação dos requisitos e a implementação do sistema.

A rastreabilidade de requisitos é essencial para a análise de impacto das mudanças e para a manutenção do software, permitindo a identificação de requisitos ausentes, a eliminação de requisitos desnecessários, a garantia de certificação e conformidade, a análise do impacto das mudanças, a facilitação da manutenção do software e a execução de testes. Embora manter a rastreabilidade seja desafiador, é fundamental para garantir uma compreensão clara do escopo

ANÁLISE E MODELAGEM DE SISTEMAS

das mudanças no software. Uma forma comum de representar a rastreabilidade é através de uma matriz bidimensional, que pode ser implementada utilizando uma planilha eletrônica ou uma ferramenta automatizada, assegurando a atualização contínua das informações rastreadas.

Em suma, a engenharia de requisitos é um processo essencial e contínuo durante todo o desenvolvimento de software, assegurando a compreensão e tradução precisa das necessidades dos stakeholders em requisitos claros. Ao fornecer uma estrutura para várias etapas, como elicitação, análise, documentação, validação e gerenciamento de requisitos, ela desempenha um papel fundamental na criação de sistemas de software bem-sucedidos que atendem às metas do cliente e do usuário final. Além disso, destaca-se a importância da adaptação contínua dos requisitos para acompanhar as mudanças no ambiente empresarial e tecnológico, garantindo a pertinência e eficácia do sistema ao longo do tempo. Em resumo, a engenharia de requisitos é uma disciplina dinâmica e crucial que fomenta a colaboração entre todas as partes envolvidas, resultando em soluções de software que agregam valor e satisfazem as necessidades em constante evolução do mercado.

É Hora de Praticar!



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Esta atividade tem como objetivo compreender a diferença entre requisitos funcionais e estruturais (também conhecidos como requisitos não funcionais) em projetos de desenvolvimento de software. Você vai praticar a classificação de requisitos para um projeto de software fictício, aprimorando sua capacidade de identificar e diferenciar esses dois tipos de requisitos.

Projeto de software fictício: Sistema de reservas de hotéis on-line

O sistema permite que os usuários busquem hotéis, vejam detalhes dos quartos, comparem preços, façam reservas e paguem online. Ele também deve oferecer uma interface administrativa para os hotéis cadastrarem e atualizarem informações sobre quartos, preços e disponibilidade.

Lista de requisitos para classificação:

- O sistema deve permitir que os usuários criem uma conta usando seu endereço de e-mail.
- O sistema deve criptografar as senhas dos usuários antes de armazená-las no banco de dados.
- O tempo de resposta para buscas de hotéis não deve exceder 2 segundos.

ANÁLISE E MODELAGEM DE SISTEMAS

- O sistema deve permitir que os usuários filtrem hotéis por localização, preço e disponibilidade.
- O sistema deve enviar um e-mail de confirmação após a conclusão de uma reserva.
- O sistema deve ser acessível em dispositivos móveis e desktops.
- O sistema deve manter um registro de todas as reservas feitas, acessível pelos administradores do hotel.
- O sistema deve garantir uma disponibilidade de 99,9%.
- O sistema deve oferecer suporte a múltiplos idiomas.
- O sistema deve permitir que os administradores do hotel gerenciem as informações de seus hotéis.

Tarefa:

- **Classificação:** você deve classificar os requisitos acima em "Funcionais" ou "Estruturais".
- **Discussão:** após a classificação, apresente a razão por trás da sua classificação, destacando a importância de cada tipo de requisito para o sucesso do projeto.
- 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?
- 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?
- Como a modelagem de software pode influenciar a qualidade e a eficácia dos sistemas desenvolvidos, além de proporcionar uma compreensão mais profunda das necessidades dos usuários e dos requisitos do projeto?

Dê o Play!

[Clique aqui](#) para acessar os slides do Dê o play!

Para o projeto de software fictício "**Sistema de reservas de hotéis on-line**" apresentado na atividade, aqui está o gabarito sugerido para a classificação dos requisitos em funcionais e estruturais:

Requisitos funcionais

1. **O sistema deve permitir que os usuários criem uma conta usando seu endereço de e-mail.**
 - Descreve uma funcionalidade específica relacionada à interação do usuário com o sistema.
2. **O sistema deve permitir que os usuários filtrem hotéis por localização, preço e disponibilidade.**
 - Detalha uma funcionalidade do sistema que permite aos usuários realizar buscas baseadas em critérios específicos.

ANÁLISE E MODELAGEM DE SISTEMAS

3. O sistema deve enviar um e-mail de confirmação após a conclusão de uma reserva.

- Especifica o comportamento do sistema de enviar notificações automáticas para os usuários.

4. O sistema deve manter um registro de todas as reservas feitas, acessível pelos administradores do hotel.

- Indica uma funcionalidade do sistema relacionada à gestão de informações de reservas.

5. O sistema deve permitir que os administradores do hotel gerenciem as informações de seus hotéis.

- Descreve uma funcionalidade administrativa do sistema para os provedores de serviço (Hotéis).

Requisitos estruturais (não funcionais)

1. O sistema deve criptografar as senhas dos usuários antes de armazená-las no banco de dados.

- Relacionado à segurança e proteção de dados, não é uma funcionalidade direta do ponto de vista do usuário.

2. O tempo de resposta para buscas de hotéis não deve exceder 2 segundos.

- Especifica um critério de desempenho que o sistema deve cumprir.

3. O sistema deve ser acessível em dispositivos móveis e desktops.

- Define um requisito de compatibilidade e acessibilidade do sistema.

4. O sistema deve garantir uma disponibilidade de 99,9%.

- Estabelece um critério operacional relacionado à disponibilidade do sistema.

5. O sistema deve oferecer suporte a múltiplos idiomas.

- Indica um requisito de internacionalização, relacionado à usabilidade e acessibilidade, mas não é uma funcionalidade específica do ponto de vista da lógica de negócios.

Essa classificação ajuda a destacar a diferença entre o que o sistema faz (requisitos funcionais) e como o sistema deve ser (requisitos estruturais). Os requisitos funcionais estão diretamente relacionados às ações que o sistema deve ser capaz de executar, enquanto os requisitos estruturais descrevem propriedades gerais e critérios que o sistema deve atender para garantir qualidade, segurança, desempenho e usabilidade.

ANÁLISE E MODELAGEM DE SISTEMAS

Este infográfico oferece uma visão sobre a modelagem de requisitos, um processo crucial na Engenharia de Software. Ao longo deste recurso visual, exploramos os diferentes aspectos da modelagem de requisitos, abrangendo as técnicas mais utilizadas, compreendendo a importância da linguagem natural na redação de requisitos e como os diagramas, como o diagrama de casos de uso e a notação SysML, podem ser empregados para visualizar e comunicar eficazmente os requisitos do sistema.

ANÁLISE E MODELAGEM DE SISTEMAS

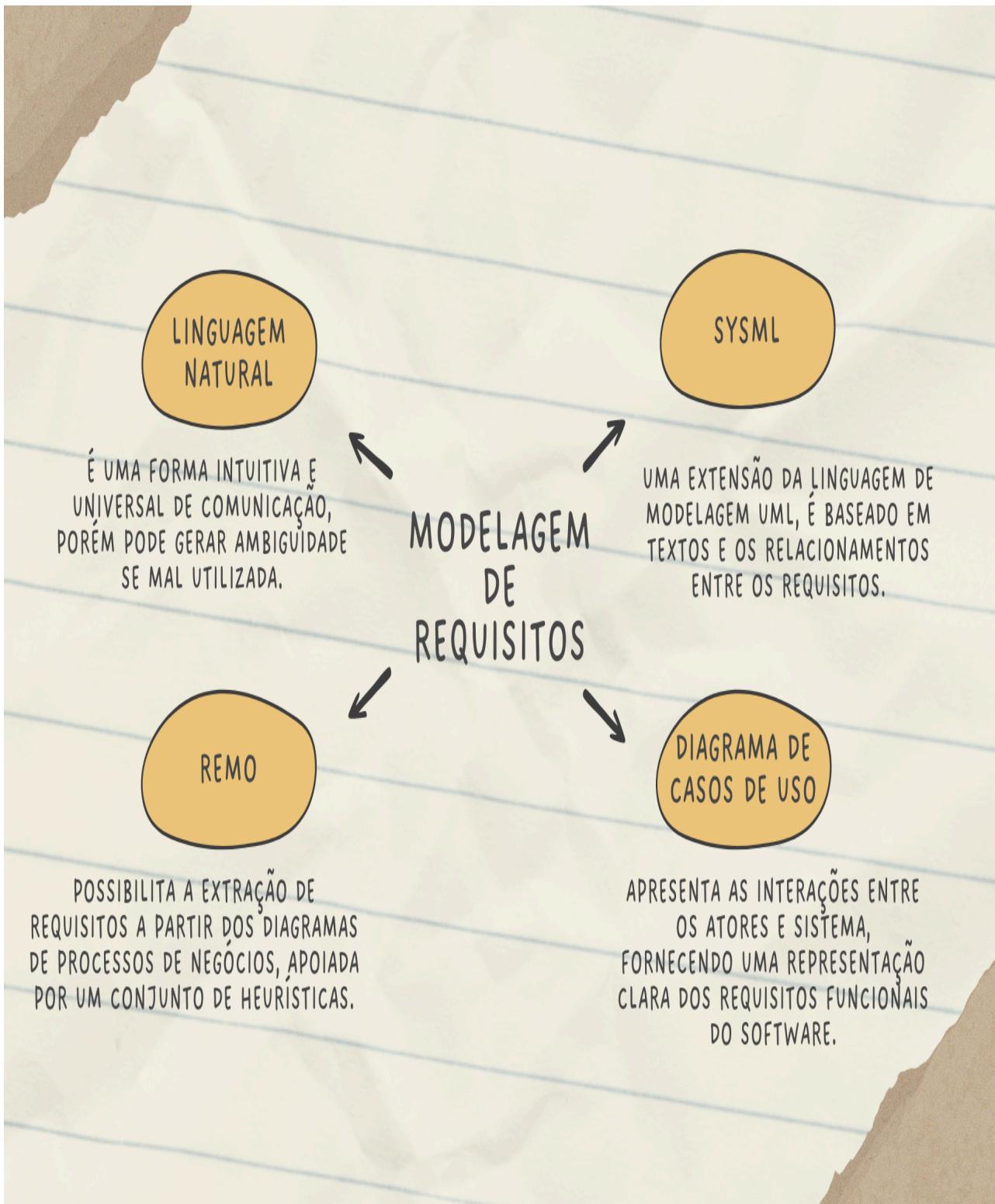


Figura | Modelagem de requisitos. Fonte: elaborada pela autora.

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

ANÁLISE E MODELAGEM DE SISTEMAS

PFLEGER, S. L. **Engenharia de Software**: teoria e prática. 2. ed. São Paulo: Prentice Hall, 2004.
PRESSMAN, R S. **Engenharia de software**: uma abordagem profissional. 9. ed. – Porto Alegre: AMGH, 2021.
SOMMERVILLE, I. **Engenharia de software**. 10. ed. São Paulo: Pearson, 2018.

Unidade 4

Modelagem de Sistemas

Aula 1

Linguagem de Modelagem Unificada - UML

Linguagem de Modelagem Unificada - UM



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Olá, estudante! Nesta videoaula, você conhecerá os fundamentos da UML (*Unified Modeling Language*), com foco em Diagramas Estruturais e Comportamentais. Estes conteúdos são essenciais para sua prática profissional, capacitando-o a visualizar e comunicar de forma eficaz a arquitetura e o comportamento de sistemas de software. Dominar a UML e seus diagramas permite uma análise mais precisa e uma comunicação mais clara entre os membros da equipe de desenvolvimento. Prepare-se para adquirir habilidades valiosas e impulsionar sua carreira.

[Clique aqui](#) para acessar os slides da sua videoaula.

Bons estudos!

Ponto de Partida

Nesta aula, exploraremos uma linguagem essencial para a modelagem de sistemas de software, a UML(*Unified Modeling Language*). Começaremos discutindo a importância da UML como uma ferramenta padronizada e visualmente intuitiva para representar conceitos complexos de

ANÁLISE E MODELAGEM DE SISTEMAS

engenharia de software. Em seguida, abordaremos os diagramas estruturais, como o diagrama de classes e o diagrama de componentes, que são cruciais para compreender a organização estática de um sistema, incluindo suas classes, interfaces e relacionamentos. Estes diagramas são fundamentais durante as fases de design e arquitetura do software, ajudando os desenvolvedores a visualizar e comunicar a estrutura do sistema de forma clara e precisa.

Além disso, exploramos os diagramas comportamentais, como o diagrama de sequência e o diagrama de atividade, que oferecem insights valiosos sobre o fluxo dinâmico de interações e eventos em um sistema de software. Esses diagramas são essenciais para entender como os diferentes componentes do sistema interagem entre si ao longo do tempo, permitindo aos desenvolvedores modelar e analisar o comportamento funcional do software. Compreender os diagramas comportamentais é crucial para garantir que o sistema atenda aos requisitos funcionais e de desempenho esperados, além de facilitar a detecção e resolução de potenciais problemas de design.

Pensando nesses diagramas, vamos imaginar a seguinte situação, a empresa XYZ está planejando desenvolver um novo sistema de gerenciamento de projetos online chamado "ProjectMaster". Este sistema permitirá que os usuários criem projetos, atribuam tarefas, acompanhem o progresso, gerenciem recursos, colaborem em tempo real e gerem relatórios de desempenho. "ProjectMaster" será um sistema complexo com várias interações entre usuários e componentes do sistema, bem como uma estrutura de banco de dados abrangente para manter informações sobre projetos, tarefas, usuários e relatórios.

Como atividade, você deve refletir sobre o problema descrito e determinar quais diagramas UML seriam mais úteis para planejar e modelar o sistema "ProjectMaster". Sua tarefa é escolher ao menos três tipos de diagramas UML e justificar como cada um contribuirá para o desenvolvimento do sistema.

Para cada diagrama UML escolhido, você deve fornecer:

- 1. Nome do diagrama UML:** especifique qual diagrama você está escolhendo.
- 2. Propósito do diagrama:** descreva qual aspecto do sistema o diagrama ajudará a modelar.
- 3. Justificativa:** explique por que você acredita que esse diagrama é uma boa escolha para o problema descrito e como ele pode beneficiar a equipe de desenvolvimento do "ProjectMaster".

Vamos Começar!

Fundamentos da UML

A Linguagem de Modelagem Unificada (*Unified Modeling Language - UML*) é uma linguagem visual projetada para modelar software baseado no paradigma de orientação a objetos. Ela é

ANÁLISE E MODELAGEM DE SISTEMAS

uma linguagem de modelagem de propósito geral, aplicável a uma ampla variedade de domínios de aplicação. Nos últimos anos, a UML emergiu como a linguagem-padrão de modelagem adotada internacionalmente pela indústria de engenharia de software.

É importante ressaltar que a UML não é uma linguagem de programação; em vez disso, é uma linguagem de modelagem, uma notação destinada a auxiliar engenheiros de software na definição das características do sistema. Isso inclui requisitos, comportamento, estrutura lógica, dinâmica de processos e até mesmo requisitos físicos relacionados ao hardware no qual o sistema será implantado. Essas características podem ser definidas por meio da UML antes do início do desenvolvimento do software (FOWLER, 2005).

Adicionalmente, é importante ressaltar que a UML não está intrinsecamente ligada a um processo específico de desenvolvimento de software. Ela é completamente independente e pode ser utilizada por uma variedade de processos de desenvolvimento diferentes, ou até mesmo de maneiras personalizadas, de acordo com a preferência do engenheiro.

Por que é necessário modelar um software? Muitos profissionais argumentam que conseguem discernir todas as necessidades de um sistema de informação de forma intuitiva e sempre operaram dessa maneira. Da mesma forma, por que projetar uma casa? Um pedreiro experiente não seria capaz de construí-la sem um projeto detalhado? Embora essas afirmações possam ser verdadeiras em alguns casos, a questão é muito mais complexa e abrange uma série de fatores, incluindo levantamento e análise de requisitos, prototipagem, escala do projeto, complexidade, prazos, custos, documentação, manutenção e reutilização, entre outros (GUEDES, 2011).

A diferença entre construir uma pequena casa e erigir um prédio de vários andares é marcante. Para edificar um prédio, é imprescindível elaborar um projeto minucioso, com cálculos precisos e corretos. Além disso, é necessário estimar custos, determinar o tempo de construção, avaliar a mão de obra necessária, especificar a quantidade de materiais, escolher o local de construção e muito mais. Grandes projetos não podem ser improvisados, e mesmo a maioria dos projetos pequenos demanda uma abordagem estruturada.

Na realidade, qualquer sistema, por mais simples que seja, deve ser modelado antes de sua implementação, principalmente porque os sistemas de informação tendem a crescer em tamanho, complexidade e alcance. Muitos profissionais consideram os sistemas de informação como "vivos" porque estão sempre sujeitos a mudanças. Na verdade, a designação mais precisa seria "dinâmicos", pois estão constantemente evoluindo. Essas mudanças são motivadas por diversos fatores, como solicitações de clientes por modificações ou melhorias, mudanças no mercado que exigem adaptações nas estratégias das empresas e alterações na legislação que requerem ajustes nos sistemas.

Portanto, é essencial que os sistemas de informação possuam documentação detalhada, precisa e atualizada para facilitar sua manutenção sem introduzir novos erros. A modelagem é uma maneira eficaz de documentar um sistema, mas não se limita a isso. Além da documentação, a modelagem oferece diversas outras vantagens (GUEDES, 2011).

ANÁLISE E MODELAGEM DE SISTEMAS

A modelagem de software envolve a criação de modelos de software, mas o que exatamente é um modelo de software? Um modelo de software representa uma perspectiva de um sistema físico; é uma representação abstrata do sistema com um propósito específico, como descrever aspectos estruturais ou comportamentais do software. Esse propósito determina o que deve ser incorporado ao modelo e o que é considerado irrelevante.

Portanto, um modelo abrange completamente os aspectos do sistema físico que são relevantes para o propósito do modelo, no nível apropriado de detalhamento. Por exemplo, um modelo de casos de uso oferecerá uma visão dos requisitos essenciais do sistema, identificando as funcionalidades do software e os usuários que poderão interagir com elas, sem se preocupar em detalhar aspectos adicionais. Enquanto isso, um modelo conceitual identifica as classes relacionadas ao domínio do problema, sem entrar em detalhes sobre seus métodos. Por outro lado, um modelo de domínio amplia o modelo conceitual, incluindo informações relacionadas à solução do problema, como os métodos necessários para essa solução (GUEDES, 2011).

Atualmente, a versão mais recente da UML, conhecida como UML 2.5.1, oferece 13 diagramas diferentes para uso na modelagem de software (PRESSMAN, 2021). Com uma quantidade significativa de diagramas disponíveis, os desenvolvedores têm à disposição uma poderosa ferramenta para representar e comunicar diferentes aspectos de seus sistemas de software.

Os diagramas da UML podem ser categorizados em dois grandes grupos: os diagramas UML estruturais e os diagramas UML comportamentais. Além disso, existem os diagramas de interação, que geralmente se enquadram no grupo dos diagramas comportamentais. Essa distinção permite uma visualização do sistema a partir de diferentes perspectivas. A Figura 1 apresenta essa classificação e os diagramas que fazem parte de cada grupo.

ANÁLISE E MODELAGEM DE SISTEMAS

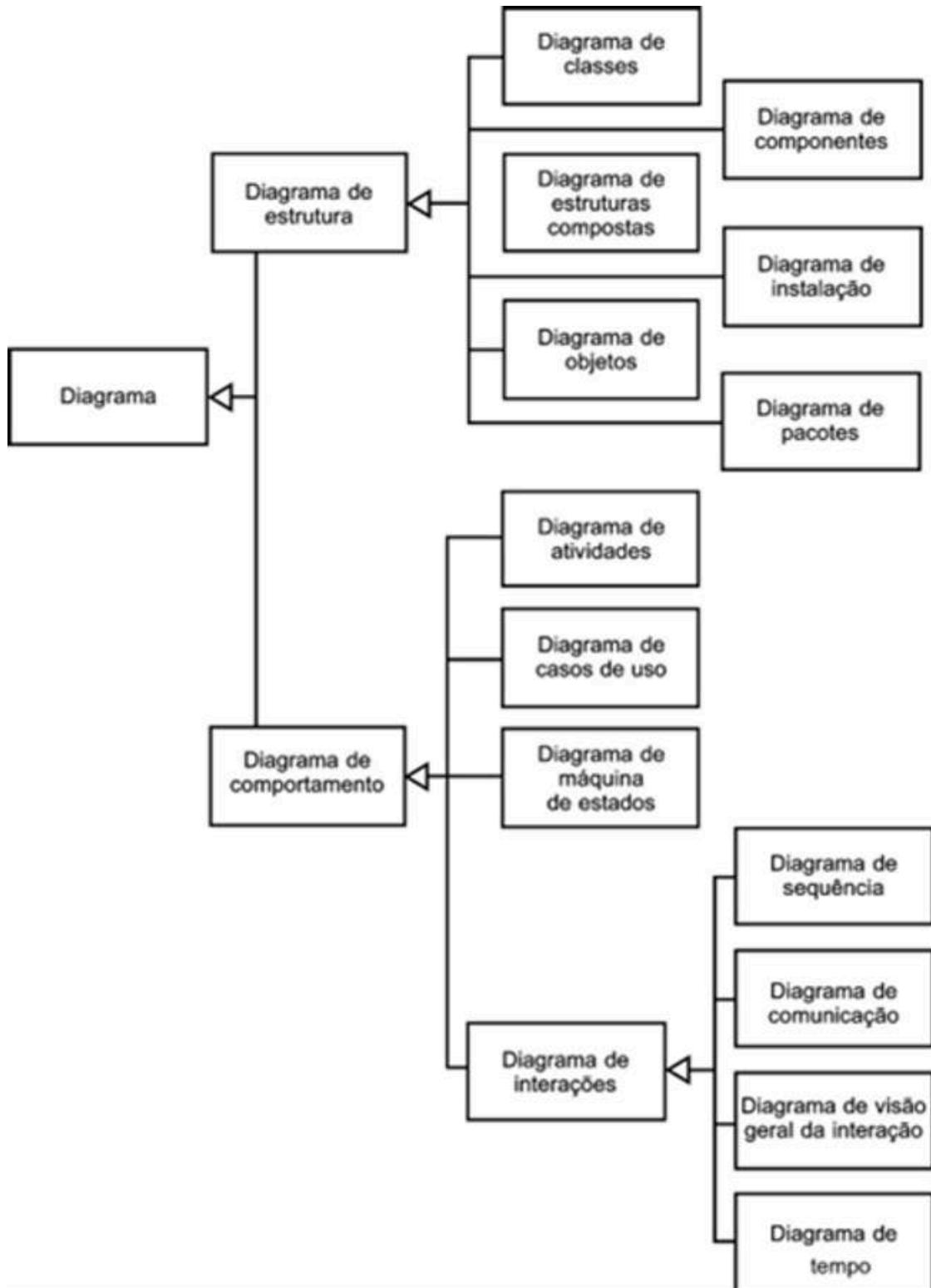


Figura 1 | Classificação dos tipos de diagrama da UML. Fonte: adaptada de Fowler (2005, [s. p.]).

ANÁLISE E MODELAGEM DE SISTEMAS

Embora os manuais da linguagem UML não prescrevam uma ordem específica para a criação e utilização dos diagramas em um fluxo de desenvolvimento de software, geralmente os diagramas são apresentados em uma sequência didática. Essa sequência segue uma lógica baseada na complexidade dos diagramas, onde os mais simples são abordados primeiro para facilitar a compreensão do processo de criação. Segundo essa abordagem, começamos de maneira simplificada com os diagramas estruturais e, em seguida, avançamos para os diagramas comportamentais, incluindo uma subdivisão para os diagramas de interação, que serão também explorados.

Siga em Frente...

Diagramas estruturais

Os diagramas estruturais revelam a organização de um sistema em suas partes constituintes, componentes e as interconexões entre esses elementos. Esses diagramas são frequentemente vinculados à modelagem estática, já que descrevem a estrutura do sistema em si. Geralmente, os diagramas estruturais são concebidos durante a fase inicial do projeto de arquitetura do sistema, representando conceitos significativos como abstrações, considerações de implementação e características do mundo real (FOWLER, 2005).

Os diagramas estruturais são seis, descritos a seguir, iniciando com o diagrama que provavelmente é um dos mais utilizados até por desenvolvedores que não estão totalmente familiarizados com os conceitos da UML: o diagrama de classes.

O **diagrama de classes** é amplamente reconhecido como um dos elementos mais essenciais e frequentemente utilizado na UML (GUEDES, 2011). Funciona como uma base para a maioria dos outros diagramas, fornecendo uma representação estruturada das classes empregadas pelo sistema. Este diagrama delineia os atributos e métodos associados a cada classe, além de especificar os relacionamentos e interações entre elas.

O **diagrama de pacotes** tem como objetivo representar os subsistemas ou submódulos incluídos em um sistema, identificando suas partes constituintes. Pode ser empregado de forma autônoma ou em conjunto com outros diagramas. Além disso, é útil para ilustrar a arquitetura de uma linguagem, como no caso da UML, e para definir as camadas de um software ou processo de desenvolvimento (PRESSMAN, 2021).

O **diagrama de componentes** está intimamente ligado à linguagem de programação escolhida para o desenvolvimento do sistema modelado. Ele visualiza os componentes do sistema no momento da implementação, representando-os como módulos de código-fonte, bibliotecas, formulários, arquivos de ajuda, entre outros (FOWLER, 2005). Este diagrama determina a estrutura e a interação dos componentes, essenciais para o funcionamento adequado do sistema.

ANÁLISE E MODELAGEM DE SISTEMAS

O **diagrama de instalação** descreve a estrutura de hardware e software necessária para a correta execução do software em desenvolvimento. No diagrama é possível relacionar quais componentes do software desenvolvido serão executados em cada um dos nós de hardware descritos.

O **diagrama de objetos** está intimamente relacionado ao diagrama de classes, servindo como um complemento essencial e dependente deste último (GUEDES, 2011). Ele oferece uma perspectiva dos valores contidos pelos objetos representados no diagrama de classes em um instante específico durante a execução de um processo de software.

O **diagrama de estrutura composta** oferece uma representação detalhada da organização interna de um classificador, como uma classe ou componente, delineando suas partes constituintes, comunicação e colaboração entre elas. Além disso, é empregado para descrever uma colaboração na qual um conjunto de instâncias trabalha em conjunto para realizar uma determinada tarefa.

Diagramas comportamentais

Os diagramas comportamentais têm como finalidade representar o fluxo de informações e eventos ao longo do tempo no sistema. Em outras palavras, eles ilustram a resposta do sistema a eventos do ambiente, exibindo o comportamento dinâmico dos objetos e como o sistema reage a ações específicas ou eventos.

Os **diagramas de interação** estão incluídos no conjunto de diagramas comportamentais e são utilizados para modelar as interações dentro do sistema ou entre os seus componentes.

O **diagrama de casos de uso** é amplamente reconhecido como o diagrama mais abrangente e informal da UML, frequentemente empregado nas etapas de levantamento e análise de requisitos do sistema. Contudo, ele continua a ser uma referência ao longo de todo o processo de modelagem e pode servir como base para outros diagramas. Este diagrama utiliza uma linguagem simples e de fácil compreensão para proporcionar aos usuários uma visão geral do comportamento esperado do sistema (PRESSMAN, 2021). Seu objetivo é identificar os atores (sejam usuários, outros sistemas ou hardware especializado) que interagirão de alguma forma com o software, bem como os serviços ou funcionalidades oferecidas pelo sistema aos atores, denominados casos de uso neste contexto.

O **diagrama de atividade** tem como objetivo descrever os passos necessários para a conclusão de uma determinada atividade, que pode variar desde um método com certo nível de complexidade até um algoritmo ou até mesmo um processo completo. Sua principal preocupação reside na representação do fluxo de controle durante a execução da atividade.

O **diagrama de visão geral de interação** é uma derivação do diagrama de atividade, projetado para oferecer uma perspectiva global dentro de um sistema ou processo de negócios. Esta

ANÁLISE E MODELAGEM DE SISTEMAS

representação foi introduzida na UML 2.

O **diagrama de sequência** é uma representação comportamental que se concentra na ordem temporal das mensagens trocadas entre os objetos envolvidos em um determinado processo. Tipicamente, ele se baseia em um caso de uso definido pelo mesmo nome e utiliza o diagrama de classes para identificar os objetos das classes envolvidas no processo. Este diagrama costuma identificar o evento que desencadeia o processo modelado, assim como o ator responsável por esse evento. Ele delinea como o processo deve se desdobrar e ser concluído, através da invocação de métodos por meio de mensagens enviadas entre os objetos.

O **diagrama de máquina de estados** ilustra o comportamento de um elemento através de um conjunto finito de transições de estado, representando essencialmente uma máquina de estados. Além de descrever o comportamento de uma parte do sistema, quando é denominado máquina de estado comportamental, ele também pode expressar o protocolo de uso de uma parte do sistema ao identificar uma máquina de estado de protocolo. Similar ao diagrama de sequência, o diagrama de máquina de estados pode ser baseado em um caso de uso, mas também é capaz de rastrear os estados de outros elementos, como uma instância de uma classe.

O **diagrama de comunicação** está estreitamente ligado ao diagrama de sequência, sendo, na verdade, um complemento deste. As informações representadas no diagrama de comunicação muitas vezes são semelhantes às apresentadas no diagrama de sequência, porém com uma abordagem distinta. Enquanto o diagrama de sequência se concentra na ordem temporal das interações, o diagrama de comunicação destaca como os elementos estão conectados e quais mensagens são trocadas entre eles durante o processo.

O **diagrama de tempo** representa a evolução do estado ou condição de uma instância de classe ou seu papel ao longo de um período específico. Ele é frequentemente empregado para ilustrar a alteração no estado de um objeto ao longo do tempo em resposta a eventos externos (GUEDES, 2011).

Vamos Exercitar?

Esta atividade visa desenvolver a capacidade de tomada de decisão do estudante sobre quais diagramas UML são mais adequados para modelar diferentes aspectos de um sistema de software. Com isso, você deve selecionar e justificar o uso de diagramas UML específicos com base em um cenário de problema fornecido no início da aula.

Um exemplo de resposta é:

- **Diagrama UML:** diagrama de casos de uso
- **Propósito do diagrama:** representar as funcionalidades do sistema a partir da perspectiva do usuário, mostrando como diferentes atores interagem com o sistema.

ANÁLISE E MODELAGEM DE SISTEMAS

- **Justificativa:** esse diagrama é crucial para entender as necessidades dos usuários e garantir que o "ProjectMaster" ofereça as funcionalidades necessárias para gerenciamento de projetos. Ele ajuda a definir os requisitos do sistema de uma maneira que todos os stakeholders possam entender.

Saiba mais

O livro *UML Essencial* apresenta os principais conceitos da UML, além de trazer diversos exemplos de diagramas.

FOWLER, Martin. [UML essencial](#). 3.ed. – Porto Alegre: Bookman, 2005.

A UML é uma linguagem de modelagem que suporta o paradigma Orientado a Objetos. No livro *Análise e Design Orientados a Objetos para Sistemas de Informação* você pode ver diversos exemplos utilizando a UML para a POO.

WAZLAWICK, R. S. [Análise e Design Orientados a Objetos para Sistemas de Informação: Modelagem com UML, OCL e IFML](#). Grupo GEN, 2014.

Referências

FOWLER, M. **UML essencial**: um breve guia para a linguagem-padrão de modelagem de objetos. 3.ed. – Porto Alegre: Bookman, 2005.

GUEDES, Gilleanes T. **A UML 2**: uma abordagem prática. 2. ed. - São Paulo: Novatec Editora, 2011.

PRESSMAN, Roger S. **Engenharia de software**: uma abordagem profissional. 9. ed. – Porto Alegre: AMGH, 2021.

Aula 2

Diagrama de Casos de Uso

Diagrama de casos de uso

Este conteúdo é um vídeo!

ANÁLISE E MODELAGEM DE SISTEMAS



Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Nesta videoaula, abordaremos a introdução ao Diagrama de casos de uso, seus elementos e um exemplo prático de sua aplicação. Esses conteúdos são fundamentais para sua prática profissional na Engenharia de software, pois permitem a compreensão e a comunicação eficaz dos requisitos de um sistema. Dominar o Diagrama de casos de uso capacita o profissional a modelar e analisar os comportamentos esperados do software, garantindo sua eficiência e adequação às necessidades dos usuários. Prepare-se para adquirir conhecimentos essenciais e elevar sua competência na área!

[Clique aqui](#) para acessar os slides da sua videoaula.

Bons estudos!

Ponto de Partida

Nesta aula, exploraremos um dos aspectos fundamentais da modelagem de sistemas de software: o Diagrama de casos de uso. Este diagrama é uma ferramenta poderosa da Linguagem de Modelagem Unificada (UML), essencial para capturar as funcionalidades de um sistema do ponto de vista dos usuários. Ao entender como criar e interpretar um diagrama de casos de uso, você estará equipado para visualizar o escopo de um projeto, detalhar interações entre o sistema e seus usuários e identificar requisitos críticos de uma maneira organizada e eficiente. Estudaremos atores, casos de uso, relações de inclusão e extensão. Todos os componentes que juntos fornecem uma visão clara das expectativas dos usuários e dos requisitos funcionais do sistema.

Compreender o diagrama de casos de uso é mais do que um exercício acadêmico; é uma habilidade prática que melhora a comunicação entre desenvolvedores, analistas de sistemas, stakeholders e usuários finais. Essa compreensão assegura que o desenvolvimento do software esteja alinhado com as necessidades reais dos usuários, evitando mal-entendidos e retrabalhos dispendiosos.

Para exercitar o que será aprendido nesta aula, elabore um diagrama de casos de uso da seguinte situação: "O sistema de pedidos da empresa oferece uma série de funcionalidades para facilitar e gerenciar a forma como os clientes realizam pedidos. Os clientes podem fazer pedidos de três maneiras diferentes: por telefone, on-line através do website, ou usando o aplicativo móvel. Quando um pedido é realizado, independente do método, ele é gerenciado por um responsável pelos pedidos, que tem a tarefa de assegurar que o pedido seja processado corretamente. Para os pedidos on-line, existe a possibilidade de extensão para fazer pedidos utilizando o aplicativo móvel. Além disso, antes de um pedido ser confirmado, o gerente do sistema deve verificar os dados do pedido, uma etapa incluída no processo de realização de

ANÁLISE E MODELAGEM DE SISTEMAS

pedido. O sistema também permite o cadastro de novos usuários, que é uma funcionalidade acessível pelo cliente e inclui um passo adicional onde o gerente deve verificar os dados fornecidos pelo novo usuário."

Vamos Começar!

Introdução ao diagrama de casos de uso

O diagrama de casos de uso tem como objetivo simplificar a compreensão do comportamento externo do sistema, mostrando suas funcionalidades de forma acessível a qualquer pessoa e destacando a perspectiva do usuário. É o diagrama mais abstrato e flexível da UML, sendo comumente utilizado nas fases iniciais de modelagem do sistema, especialmente durante o levantamento e análise de requisitos. No entanto, ele continua sendo consultado e ajustado ao longo do processo de engenharia, servindo de base para outros diagramas.

Este diagrama oferece uma visão panorâmica das funcionalidades que o sistema deve disponibilizar aos usuários, sem entrar em detalhes sobre sua implementação. Ele é fundamental para identificar e compreender os requisitos do sistema, ajudando a especificar, visualizar e documentar as características, funções e serviços desejados pelos usuários. Além disso, ele ajuda a identificar os diferentes tipos de usuários que interagirão com o sistema, os papéis que desempenharão e as funções que cada usuário poderá solicitar.

Devido à sua linguagem informal e capacidade de proporcionar uma visão geral do comportamento do sistema a ser desenvolvido, o diagrama de casos de uso é uma ferramenta valiosa para ser apresentada durante as primeiras reuniões com os clientes. Ele serve como uma forma de ilustrar o funcionamento do sistema de informação, tornando mais fácil para os usuários compreenderem e identificarem possíveis lacunas na especificação, garantindo assim que os requisitos do sistema tenham sido entendidos corretamente. Recomenda-se fortemente apresentar o diagrama de casos de uso juntamente com um protótipo, pois isso permite que um complemente o outro, oferecendo uma visão mais abrangente e concreta do sistema em desenvolvimento.

O primeiro passo ao desenvolver um caso de uso é estabelecer o conjunto de "atores" envolvidos na narrativa. Os atores são as diversas entidades (sejam pessoas ou dispositivos) que interagem com o sistema ou produto dentro do contexto das funcionalidades e comportamentos a serem descritos. Eles representam os papéis que estas entidades desempenham enquanto o sistema está em operação. Em termos mais formais, um ator é qualquer entidade externa ao sistema ou produto que se comunica com ele e possui uma ou mais metas específicas ao usar o sistema.

É importante observar que "ator" e "usuário" não são sinônimos. Enquanto um usuário típico pode assumir diversos papéis ao interagir com um sistema, um ator representa uma classe de entidades externas (que podem ser pessoas, mas não necessariamente) que desempenham

ANÁLISE E MODELAGEM DE SISTEMAS

apenas um papel dentro do contexto de um caso de uso específico. Por exemplo, consideremos um usuário que utiliza um software para configurar os sensores de alarme em um edifício virtual. Após analisar os requisitos do sistema, o software requer quatro modos distintos de interação: modo de posicionamento, modo de teste, modo de monitoramento e modo de diagnóstico. Assim, podemos definir quatro atores: o editor, o testador, o monitorador e o diagnosticador. Em alguns casos, uma única pessoa pode desempenhar todos esses papéis, enquanto em outros, diferentes pessoas podem assumir cada papel de ator.

Elementos do diagrama de casos de uso

Atores

O diagrama de casos de uso se concentra em dois elementos principais: atores e casos de uso. Os atores representam os papéis desempenhados pelos vários usuários que podem utilizar os serviços e funções do sistema de alguma forma. Em alguns casos, um ator pode representar hardware especializado ou até mesmo outro software que interaja com o sistema, como em um sistema integrado. Portanto, um ator pode ser qualquer entidade externa que interaja com o software. Os atores são representados por símbolos de "bonecos magros", acompanhados de uma breve descrição abaixo de seu símbolo, identificando o papel que o ator desempenha no contexto do diagrama. A Figura 1 apresenta a representação de três possíveis atores de um sistema.

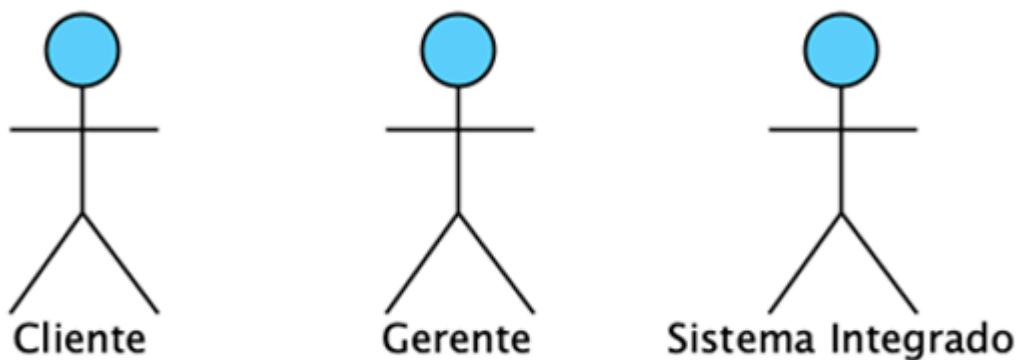


Figura 1 | Atores. Fonte: elaborado pela autora.

Nesse exemplo, os atores Cliente e Gerente representam usuários normais, enquanto o ator Sistema Integrado representa um software que interage de alguma forma com o sistema.

ANÁLISE E MODELAGEM DE SISTEMAS

Casos de uso

Os casos de uso servem para capturar os requisitos do sistema, ou seja, englobam os serviços, tarefas ou funcionalidades identificadas como essenciais para o software e que podem ser utilizados pelos atores que interagem com o sistema. Eles são empregados para expressar e documentar os comportamentos desejados das funções do sistema. Podem ser divididos em casos de uso primários e secundários. Um caso de uso primário refere-se a um processo crucial que se concentra em um dos requisitos funcionais do software, como efetuar um saque ou gerar um extrato em um sistema bancário. Por outro lado, um caso de uso secundário aborda um processo periférico, como a manutenção de um cadastro (GUEDES, 2011).

Em algumas situações, um caso de uso pode estar associado a um formulário do sistema, embora isso não seja uma regra absoluta, uma vez que uma funcionalidade do sistema, ou seja, um caso de uso, pode abranger vários formulários ou até mesmo consistir em uma pequena opção de interface, como um botão, por exemplo. Os casos de uso são representados por elipses contendo um texto que descreve a funcionalidade a que se refere. Não há um limite específico para o texto dentro da elipse, mas geralmente a descrição de um caso de uso é concisa. A Figura 2 exemplifica um caso de uso, representando a funcionalidade de matricular aluno.



ANÁLISE E MODELAGEM DE SISTEMAS

Figura 2 | Exemplo de casos de uso. Fonte: elaborada pela autora.

Os casos de uso costumam ser documentados, fornecendo instruções em linhas gerais de como será seu funcionamento, quais atividades deverão ser executadas, qual evento forçará sua execução, quais atores poderão utilizá-los e quais suas possíveis restrições, entre outras.

A documentação de um caso de uso geralmente inclui informações básicas sobre sua função, os atores envolvidos, as etapas necessárias para sua execução por parte do ator e do sistema, os parâmetros necessários e quaisquer restrições ou validações aplicáveis. No entanto, não há um formato fixo para a documentação de casos de uso definido pela UML, o que reflete a própria flexibilidade do diagrama. Assim, o formato da documentação pode variar amplamente, permitindo diferentes abordagens, incluindo o uso de pseudocódigo ou até mesmo código de programação real. No entanto, é importante considerar que o objetivo principal do diagrama de casos de uso é utilizar uma linguagem simples e acessível, comprehensível até mesmo para usuários não técnicos.

Por essa razão, é recomendável evitar o uso de pseudocódigo na documentação de casos de uso, uma vez que isso é mais apropriado para outros tipos de diagramas, como o de atividade. Além disso, os casos de uso também podem ser documentados usando outros tipos de diagramas, como o de sequência, de estado ou de atividade (GUEDES, 2011). O Quadro 1 oferece uma sugestão de como documentar o caso de uso representado na Figura 2.

Nome do caso de uso		Matricular Aluno
Ator 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 logar 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.		2. O sistema deverá verificar a matrícula e validar a senha do aluno. 3. Deverão ser apresentadas as

ANÁLISE E MODELAGEM DE SISTEMAS

6. O aluno poderá escolher entre imprimir ou salvar o comprovante de matrícula.

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.	<p>2.1 - Caso o aluno não tenha senha o sistema deverá permitir o cadastramento da senha ou a sua redefinição.</p> <p>3.1 - Deverá ser verificado o semestre atual do aluno, se for o 1º semestre do aluno, ele deverá escolher todas as disciplinas.</p> <p>3.2 - Deverá ser verificada a quantidade de disciplinas a serem escolhidas (limite dever ser igual a 6).</p>

Quadro 1 | Especificação do caso de uso: Matricular Aluno. Fonte: adaptado de Werlich (2020, p. 159).

Associação

As associações representam as interações ou conexões entre os atores dentro do diagrama, entre os atores e os casos de uso, ou entre os próprios casos de uso. Esses relacionamentos entre os casos de uso recebem nomes específicos, como inclusão, extensão e generalização (FOWLER, 2015). Uma associação entre um ator e um caso de uso indica que o ator está de alguma forma envolvido na utilização da funcionalidade representada pelo caso de uso, seja solicitando a execução dessa função, seja recebendo o resultado produzido por ela em resposta a uma ação de outro ator.

ANÁLISE E MODELAGEM DE SISTEMAS

Essa associação entre um ator e um caso de uso é denotada por uma linha conectando o ator ao caso de uso. Em certos casos, as extremidades dessa linha podem ter setas, indicando a direção do fluxo de informações: se elas são fornecidas pelo ator ao caso de uso, se são transmitidas pelo caso de uso ao ator ou se ocorre uma troca de informações bidirecional (nesse caso, a linha não possui setas, indicando que as informações fluem em ambas as direções). As setas também esclarecem quem inicia a comunicação. Além disso, uma associação pode conter uma descrição própria para esclarecer a natureza das informações transmitidas ou para atribuir um nome à associação, quando necessário. A Figura 3 apresenta uma associação entre ator e caso de uso.

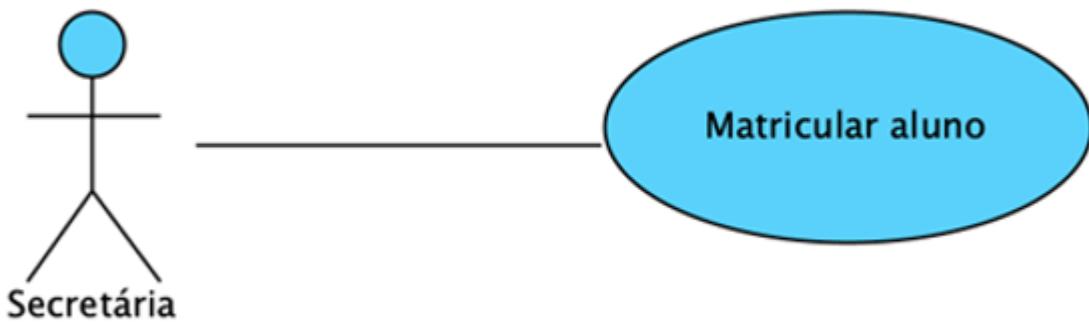


Figura 3 | Associação entre ator e um caso de uso. Fonte: elaborada pela autora.

Ao examinarmos o exemplo ilustrado pela Figura 3, percebemos que o ator Secretária utiliza, de alguma forma, a funcionalidade de Matricular aluno, representada pelo caso de uso, e que a informação referente a esse processo trafega nas duas direções.

Generalização/Especialização

O relacionamento de generalização/especialização é uma forma de associação entre casos de uso em que dois ou mais casos de uso compartilham características semelhantes, mas apresentam pequenas diferenças entre si. Nesse cenário, costuma-se criar um caso de uso genérico que descreve as características comuns a todos os casos de uso envolvidos, e então relacioná-lo aos casos de uso específicos, cuja documentação incluirá apenas as características distintas de cada um.

Dessa forma, torna-se desnecessário repetir a mesma documentação para todos os casos de uso, uma vez que todas as características do caso de uso genérico são herdadas pelos casos de uso especializados. Além disso, os casos de uso especializados também herdam quaisquer

ANÁLISE E MODELAGEM DE SISTEMAS

associações de inclusão ou extensão do caso de uso genérico, bem como as associações com os atores que utilizam o caso de uso genérico (GUEDES, 2011).

O relacionamento de generalização/especialização é representado por uma linha com uma seta mais espessa, que indica o caso de uso genérico (ponta da seta) e os casos de uso especializados (extremidade oposta da seta, apontando para o caso de uso genérico). Um exemplo desse tipo de relacionamento pode ser observado na Figura 4.

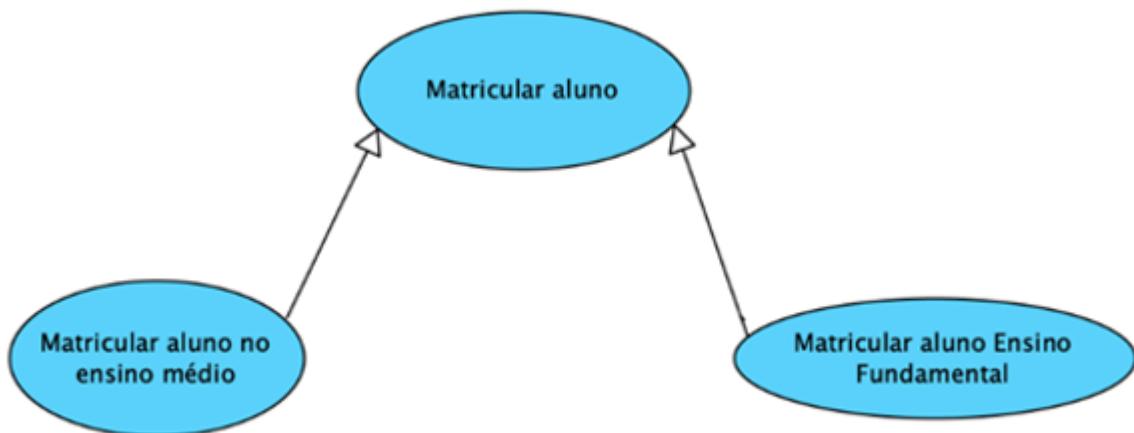


Figura 4 | Generalização/Especificação. Fonte: elaborada pela autora.

Inclusão

A associação de inclusão é frequentemente aplicada quando há uma rotina, situação ou cenário comum a múltiplos casos de uso. Nesse contexto, a documentação dessa rotina é centralizada em um caso de uso específico para que outros casos de uso possam utilizar esse serviço, evitando a redundância na descrição de uma mesma sequência de passos em vários casos de uso. As associações de inclusão indicam uma dependência obrigatória, ou seja, quando um caso de uso possui uma associação de inclusão com outro, a execução do primeiro implica automaticamente a execução do segundo. Essa relação de inclusão pode ser comparada à chamada de uma sub-rotina ou função, um conceito comum em diversas linguagens de programação (GUEDES, 2011).

Uma associação de inclusão é representada por uma linha tracejada, contendo uma seta em uma das extremidades, que aponta para o caso de uso que está sendo incluído pelo caso de uso posicionado na outra extremidade da linha. Essas associações costumam incluir um estereótipo que contém o texto "include" entre dois sinais de menor (<) e dois de maior (>). Os estereótipos são utilizados para destacar componentes ou associações, atribuindo-lhes características

ANÁLISE E MODELAGEM DE SISTEMAS

especiais em relação aos demais. Um exemplo de associação de inclusão pode ser observado na Figura 5.

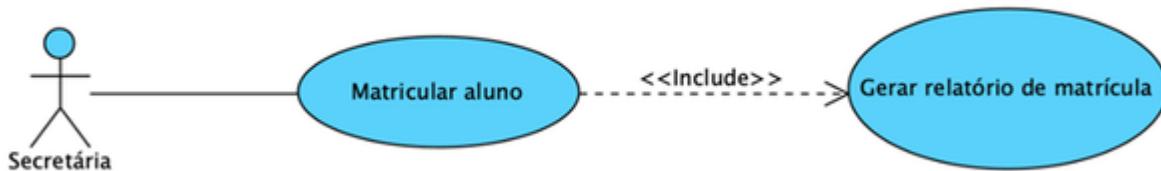


Figura 5 | Inclusão (Include). Fonte: elaborada pela autora.

Ao analisarmos a Figura 5, percebemos que sempre que ocorrer a matrícula de um aluno, o sistema deve gerar um relatório referente a essa matrícula.

Extensão

As associações de extensão são empregadas para descrever cenários opcionais dentro de um caso de uso. Os casos de uso estendidos descrevem situações que ocorrerão apenas sob circunstâncias específicas, dependendo se determinada condição for satisfeita ou não. Assim, as associações de extensão indicam a necessidade de uma avaliação para determinar se é necessário ou não executar o caso de uso estendido. Os relacionamentos de extensão representam eventos que não ocorrem sempre, embora isso não signifique que sejam raros. As associações de extensão possuem uma representação muito semelhante às associações de inclusão, sendo também representadas por uma linha tracejada. A diferença está na direção da seta, que aponta para o caso de uso que aciona a extensão, e na presença de um estereótipo contendo o texto "extend" em vez de "include" (GUEDES, 2011). Para ilustrar o conceito de extensão, apresentamos um exemplo na Figura 6.

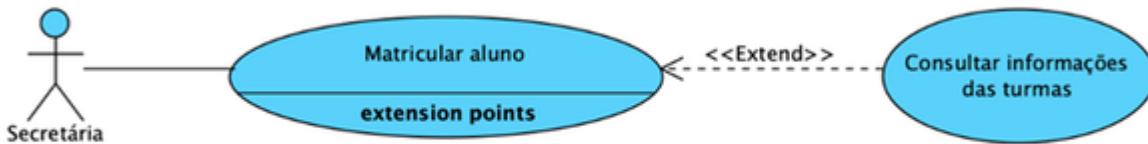


Figura 6 | Extensão (Extend). Fonte: elaborada pela autora.

No caso de uso anterior, pode-se notar que matricular aluno e consultar informações das turmas são casos de uso e existe uma relação de extensão entre eles. Isso demonstra que consultar

ANÁLISE E MODELAGEM DE SISTEMAS

informações de turmas, poderá ser chamado a partir do caso de uso matricular aluno, porém isso não é obrigatório.

Siga em Frente...

Exemplos de utilização do diagrama

Para exemplificar os elementos estudados, vamos ver o seguinte exemplo apresentado na Figura 7, que apresenta um diagrama de casos de uso para o sistema bancário.

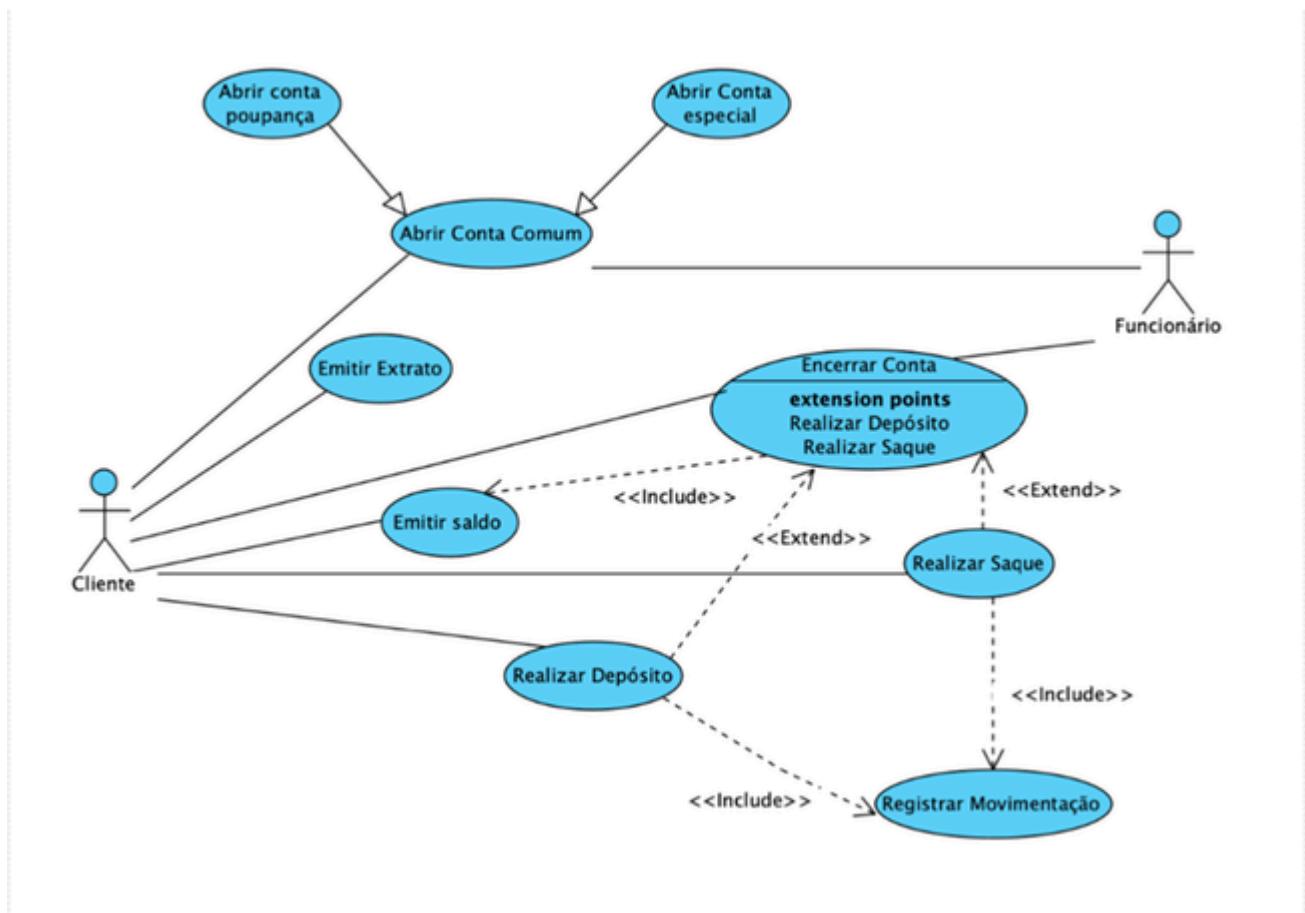


Figura 7 | Diagrama de casos de uso de um sistema bancário. Fonte: adaptada Guedes (2011).

A Figura 7 mostra o diagrama de casos de uso relacionado a um sistema de controle bancário. Este sistema possibilita que os clientes realizem a abertura e o encerramento de contas, bem como efetuem depósitos ou saques e solicitem saldos ou extratos. As quatro últimas funcionalidades mencionadas podem ser utilizadas diretamente pelo cliente por meio de um caixa eletrônico. No entanto, para abrir ou encerrar uma conta, o cliente precisa interagir com um

ANÁLISE E MODELAGEM DE SISTEMAS

funcionário do banco, que também pode executar algumas operações de manutenção em seu cadastro, como registrá-lo ou alterar seus dados.

No sistema representado, um cliente, representado como um ator, pode solicitar a abertura de uma conta comum, especial ou poupança. Como os processos de abertura para cada tipo de conta são muito semelhantes entre si, decidiu-se usar o caso de uso "Abrir Conta Comum" como uma generalização, com os outros dois tipos de abertura como especializações, detalhando as características específicas em sua documentação.

Uma associação de extensão existe entre o caso de uso "Abrir Conta Comum" (herdada também por suas especializações) e o "Manter Clientes". Embora o caso de uso "Manter Clientes" possa ser utilizado independentemente pelos funcionários do banco, a criação de uma conta normalmente implica o registro do novo cliente ou a atualização de seus dados. No entanto, como isso nem sempre é necessário, esse caso de uso não é uma inclusão e só será executado se o cliente não estiver registrado ou precisar atualizar seus dados.

Há também uma associação de inclusão entre o caso de uso "Abrir Conta Comum" (igualmente herdado por suas especializações) e o "Realizar Depósito", pois é necessário depositar algum valor no momento da abertura da conta. Sempre que uma conta é aberta, as instruções contidas no caso de uso "Realizar Depósito" são executadas.

Um cliente pode querer encerrar uma conta, mas antes disso, é necessário verificar o saldo para determinar se é necessário devolver dinheiro ao cliente ou se ele precisa depositar dinheiro para encerrar a conta. Essa verificação de saldo é obrigatória e é representada por uma associação de inclusão entre os casos de uso "Encerrar Conta" e "Emitir Saldo".

Se o saldo for positivo, um saque é realizado por meio do caso de uso "Realizar Saque". Se o saldo for negativo, um depósito é feito através do caso de uso "Realizar Depósito". Como não é possível prever se será necessário sacar ou depositar, as associações entre esses casos de uso são representadas por extensões.

Uma última associação de extensão existe entre o caso de uso "Encerrar Conta" e o "Manter Cliente" devido à possibilidade de a conta a ser encerrada ser a única conta mantida pelo cliente. Nesse caso, o registro do cliente é mantido, identificando-o como inativo, pois, uma vez cliente, seu registro não pode ser excluído do sistema.

Além disso, há os casos de uso "Emitir Saldo" e "Emitir Extrato", que são serviços simples utilizados diretamente pelo cliente por meio de um caixa eletrônico, sem a intermediação de um funcionário. Esses casos de uso não requerem interações com outros casos de uso do diagrama, embora o caso de uso "Emitir Saldo" seja utilizado pelo caso de uso "Encerrar Conta".

Por fim, abordamos os casos de uso "Realizar Saque" e "Realizar Depósito", os quais já foram mencionados anteriormente, pois também podem ser solicitados pelo caso de uso "Encerrar Conta". Normalmente, no entanto, esses serviços, assim como os casos de uso "Saldo" e

ANÁLISE E MODELAGEM DE SISTEMAS

"Extrato", são utilizados diretamente pelo cliente. No entanto, para garantir o registro adequado das transações, ambos os casos de uso obrigatoriamente envolvem o caso de uso "Registrar Movimento", que registra qualquer saque ou depósito realizados em uma conta para fins de histórico bancário. Devido à sua obrigatoriedade, a associação entre os casos de uso "Realizar Saque" e "Realizar Depósito" com o caso de uso "Registrar Movimento" precisa ser uma associação de inclusão.

O exemplo do sistema de controle bancário apresentado é relativamente complexo, envolvendo diversos tipos de associações. Entretanto, é importante ressaltar que o diagrama de casos de uso é geralmente concebido para oferecer uma visão externa simplificada do sistema ao usuário, caracterizando-se muitas vezes por sua simplicidade. Portanto, sempre que possível, deve-se evitar desenvolver diagramas de casos de uso excessivamente complexos. Ao longo deste capítulo e nas soluções dos exercícios propostos, serão apresentados novos exemplos, alguns mais simples que este (GUEDES, 2011).

Vamos Exercitar?

Nesta atividade, você deve desenvolver um diagrama de casos de uso para um sistema de pedidos de compras. Na Figura 8, está apresentado uma possível solução para o sistema proposto.

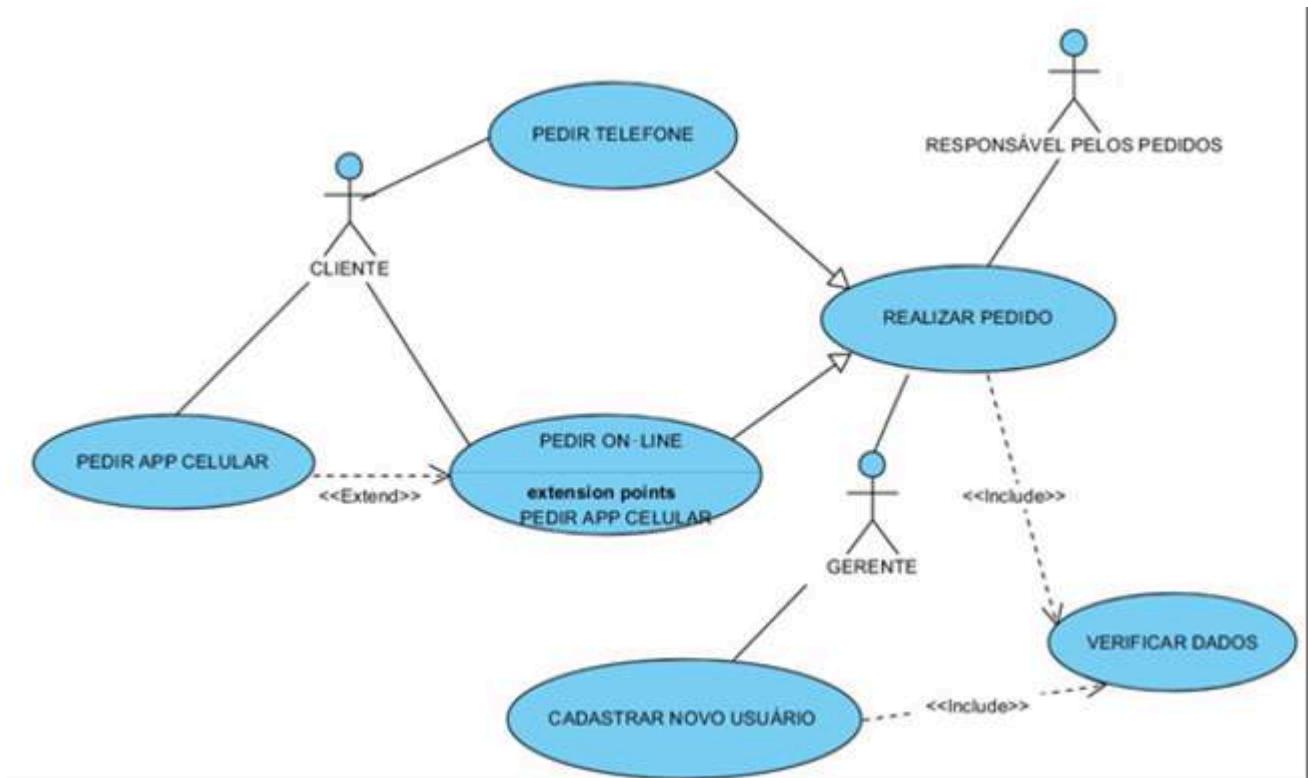


Figura 8 | Diagrama de casos de uso sistemas de pedidos. Fonte: elaborada pela autora.

ANÁLISE E MODELAGEM DE SISTEMAS

Saiba mais

O Visual Paradigm Online oferece uma plataforma intuitiva e acessível para criar diagramas UML com facilidade, ajudando os usuários a visualizarem e comunicarem eficientemente suas ideias e projetos.

VISUALPARADIGM ONLINE. [Modelos de diagramas](#). [s. d.]

O livro *UML Essencial* apresenta os principais conceitos do diagrama de casos de uso no Capítulo 9.

FOWLER, M. [UML essencial](#). 3.ed. – Porto Alegre: Bookman, 2005.

O livro *Engenharia de Software* - Pressman aborda o diagrama de casos de uso diversas vezes durante o seu livro, mas ele traz no Apêndice 1, a descrição deste diagrama e exemplos de aplicações.

PRESSMAN, R. S.; MAXIM, B. R. [Engenharia de software](#). Grupo A, 2021.

Referências

FOWLER, M. **UML essencial**: um breve guia para a linguagem-padrão de modelagem de objetos. 3.ed. – Porto Alegre: Bookman, 2005.

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

PRESSMAN, R. S. **Engenharia de software**: uma abordagem profissional. 9. ed. – Porto Alegre: AMGH, 2021.

WERLICH, C. **Análise e modelagem de sistemas**. Londrina: Editora e Distribuidora Educacional S.A., 2020.

Aula 3

Diagrama de Atividades

Diagrama de atividades

ANÁLISE E MODELAGEM DE SISTEMAS



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Nesta videoaula, exploraremos a introdução ao Diagrama de atividades, seus elementos essenciais e exemplos práticos de sua aplicação. Esses conteúdos são essenciais para sua prática profissional, permitindo a visualização e análise dos processos e fluxos de trabalho de um sistema. Ao compreender e dominar o Diagrama de atividades, você estará capacitado a modelar e otimizar os processos de negócios, sistemas de software e fluxos de trabalho, impulsionando a eficiência e a qualidade dos projetos. Prepare-se para mergulhar em conceitos fundamentais e ampliar suas habilidades nesta área!

[Clique aqui](#) para acessar os slides da sua videoaula.

Bons estudos!

Ponto de Partida

Nesta aula, vamos nos aprofundar nos Diagrama de atividades, uma parte integral da Linguagem de Modelagem Unificada (UML) que nos permite mapear processos e fluxos de trabalho dentro de sistemas complexos. Este diagrama é uma ferramenta valiosa que fornece uma visão dinâmica das sequências de ações, decisões e paralelismo em operações de negócios e sistemas de software. A importância dos Diagramas de Atividades vai além da simples documentação; eles são fundamentais para a compreensão e a otimização de processos. Ao visualizar as etapas de um processo, podemos identificar gargalos, redundâncias e oportunidades para melhorias, garantindo que o design do sistema seja tanto eficiente quanto eficaz.

Durante a aula, examinaremos os elementos que compõem o diagrama de atividades, como nós de atividades, transições, decisões, ramificações e junções, assim como pontos de sincronização. Estes elementos trabalham juntos para fornecer um mapa claro de como um processo é realizado, permitindo que todos os envolvidos no desenvolvimento do sistema tenham a mesma compreensão de como as coisas devem funcionar. Com o conhecimento prático desses conceitos, você estará bem equipado para enfrentar os desafios do design de processos em seus projetos futuros.

Para colocar em prática o que veremos em aula, pense na seguinte situação: "o processo de preparação de um bolo é um esforço colaborativo que envolve várias etapas coordenadas entre Jennie, Mary e Helen. Inicialmente, Mary é responsável por procurar a receita, enquanto Jennie e Helen preparam os ingredientes necessários. Jennie mistura os ingredientes secos e Mary mistura os ingredientes molhados. Uma vez que os ingredientes secos e molhados estão

ANÁLISE E MODELAGEM DE SISTEMAS

prontos, Jennie mistura tudo. Paralelamente, Helen é encarregada de aquecer o forno. Após a mistura estar pronta e o forno aquecido, a massa é levada para assar. A etapa de assar inclui uma verificação de conclusão, onde o bolo é testado para ver se está pronto. Se não estiver pronto, o bolo continua a assar; se estiver pronto, ele é retirado do forno por Mary, concluindo o processo de preparação do bolo”.

Vamos Começar!

Introdução ao diagrama de atividades

O diagrama de atividades, anteriormente, era considerado uma variante do antigo diagrama de gráfico de estados, agora chamado de diagrama de máquina de estados na versão atual da UML. A partir da versão 2.0 da UML, esse diagrama foi elevado à categoria de um diagrama totalmente independente, deixando de se basear em máquinas de estados e passando a se basear em redes de Petri (GUEDES, 2011).

A modelagem de atividade enfatiza a sequência e as condições para coordenar comportamentos de baixo nível. Assim, o diagrama de atividade destaca-se como o diagrama mais focado no nível de algoritmo da UML e provavelmente um dos mais detalhados. Esse diagrama compartilha muitas semelhanças com os antigos fluxogramas utilizados para desenvolver a lógica de programação e determinar o fluxo de controle de um algoritmo.

Este diagrama é empregado, conforme o próprio nome sugere, para modelar atividades, que podem ser um método ou um algoritmo, ou mesmo um processo completo. As atividades podem descrever computação procedural, sendo os métodos correspondentes às operações sobre classes nesse contexto. Elas também podem ser aplicadas à modelagem organizacional para engenharia de processos de negócios e fluxo de trabalho. Por fim, as atividades podem ser usadas na modelagem de sistemas de informação para especificar processos ao nível do sistema (GUEDES, 2011). Uma atividade é composta por um conjunto de ações, ou seja, os passos necessários para que a atividade seja concluída.

Um diagrama de atividade pode modelar mais de uma atividade. Embora uma atividade sempre contenha ações, não é obrigatório que elas estejam representadas dentro da própria atividade, podendo ser feita referência a uma atividade já modelada ou para economizar espaço no diagrama. Além disso, o diagrama de atividade pode ser usado para representar dois tipos de fluxo: de controle e de objetos, como será explorado a seguir.

Elementos do diagrama de atividades

Em geral, a criação dos diagramas de atividades é realizada do topo para o final e, normalmente, possui os seguintes elementos principais:

ANÁLISE E MODELAGEM DE SISTEMAS

- Estados iniciais e finais.
- Atividades e nós de ação
- Fluxo de controle
- Nós de Decisões
- Fork e Join
- Swimlanes (partições)

Agora, iremos definir todos os elementos listados anteriormente.

Estados iniciais e finais

Todo diagrama possui os estados iniciais e finais em sua notação, conforme notação ilustrada na Figura 1. Em um diagrama de atividades, pode-se ter um símbolo inicial e um símbolo final.

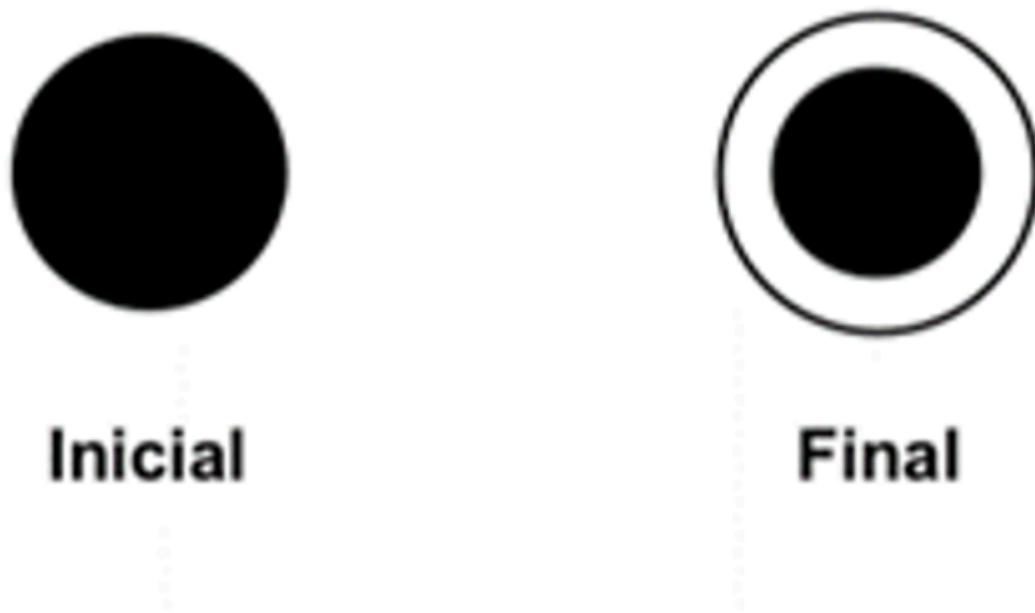


Figura 1 | Estado inicial e final. Fonte: elaborada pela autora.

O nó inicial é usado para representar o início do fluxo quando a atividade é invocada. É representado por um círculo preenchido. Já o nó final é utilizado para representar o fim de um fluxo de uma atividade. É representado por um círculo preenchido dentro de um círculo vazio.

Atividades e nós de ação

ANÁLISE E MODELAGEM DE SISTEMAS

Uma atividade define a coordenação das execuções de comportamentos subordinados utilizando um modelo de fluxo de controle e dados. Esses comportamentos subordinados podem ser acionados quando outros comportamentos no modelo terminam sua execução, quando objetos e dados se tornam disponíveis, ou quando eventos externos ocorrem (GUEDES, 2011). Uma atividade é representada por um retângulo grande com bordas arredondadas, conforme ilustrado na Figura 2.



Emitir Saldo

Figura 2 | Exemplo de atividade. Fonte: elaborada pela autora.

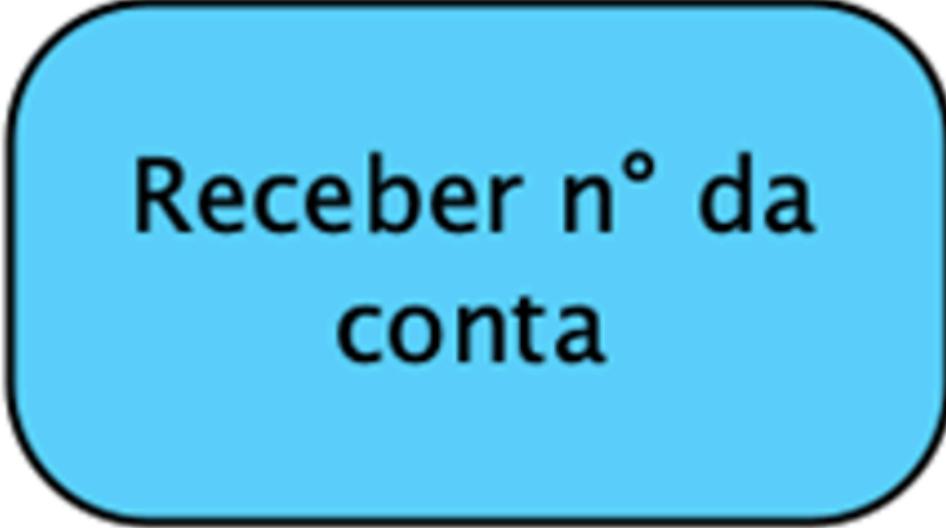
Este exemplo retrata a atividade relacionada ao processo de emissão de saldo de uma conta, que faz parte do sistema de controle bancário

As atividades podem incluir ações de diversos tipos, como:

- Execução de funções primitivas, como operações aritméticas.
- Chamadas de comportamento, como a execução de outras atividades.
- Ações de comunicação, como o envio de sinais.
- Manipulação de objetos, como a leitura ou gravação de atributos, ou até mesmo a criação ou destruição de objetos.

Já o nó de ação são os elementos fundamentais de uma atividade. Eles representam passos ou etapas que devem ser executados dentro de uma atividade. Um nó de ação é considerado atômico, ou seja, não pode ser decomposto em partes menores. Ele é simbolizado por um pequeno retângulo com bordas arredondadas, semelhante a uma atividade, porém em tamanho reduzido (Guedes, 2011). Um exemplo de nó de ação é apresentado na Figura 3.

ANÁLISE E MODELAGEM DE SISTEMAS



Receber n° da conta

Figura 3 | Exemplo de ação. Fonte: elaborada pela autora.

Fluxo de controle

O fluxo de controle é uma conexão que une dois nós, transmitindo sinais de controle entre eles. Ele é representado por uma linha com uma seta que aponta para o novo nó, partindo do nó anterior. Esta linha pode conter uma descrição, uma condição de guarda ou uma restrição, conhecida neste diagrama como peso (*weight*), que especifica, por exemplo, o número mínimo de sinais que devem ser transmitidos pelo fluxo (GUEDES, 2011).

Um sinal (token) pode conter valores de controle, objetos ou dados. No entanto, os objetos e dados só podem ser transmitidos através de um fluxo de objetos. Um exemplo de fluxo de controle é apresentado na Figura 4.

ANÁLISE E MODELAGEM DE SISTEMAS

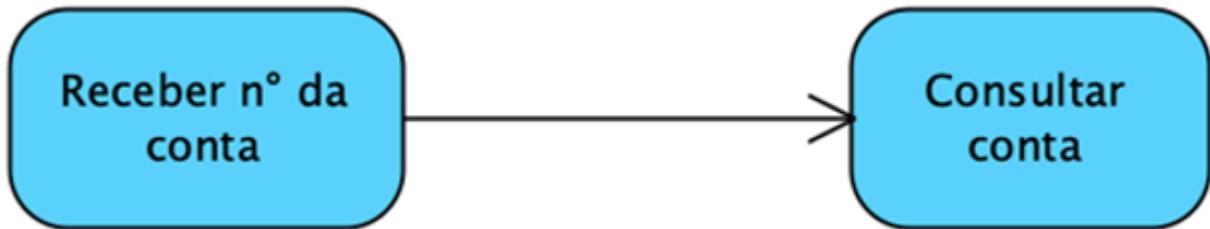


Figura 4 | Exemplo de fluxo de controle. Fonte: elaborada pela autora.

Neste exemplo, há dois nós de ação. O primeiro nó de ação representa a entrada do número de uma conta, enquanto o segundo representa a consulta da conta fornecida. É importante observar que a transição entre uma ação e outra é indicada pela linha do fluxo de controle que conecta os dois nós de ação. Além disso, a seta no fluxo de controle indica a ordem em que as ações serão executadas.

Nó de decisões

Um nó de decisão é também um nó de controle, utilizado para representar uma escolha entre dois ou mais fluxos possíveis, em que um dos fluxos será escolhido em detrimento dos outros (FOWLER, 2005). Normalmente, um nó de decisão é acompanhado por condições de guarda, que são expressas entre colchetes e determinam as condições para a seleção de um determinado caminho. Além disso, um nó de decisão pode ser usado para reunir fluxos divergentes originados de um nó de decisão anterior, nesse caso, sendo denominado nó de junção. A Figura 5 ilustra um exemplo de nó de decisão.

ANÁLISE E MODELAGEM DE SISTEMAS

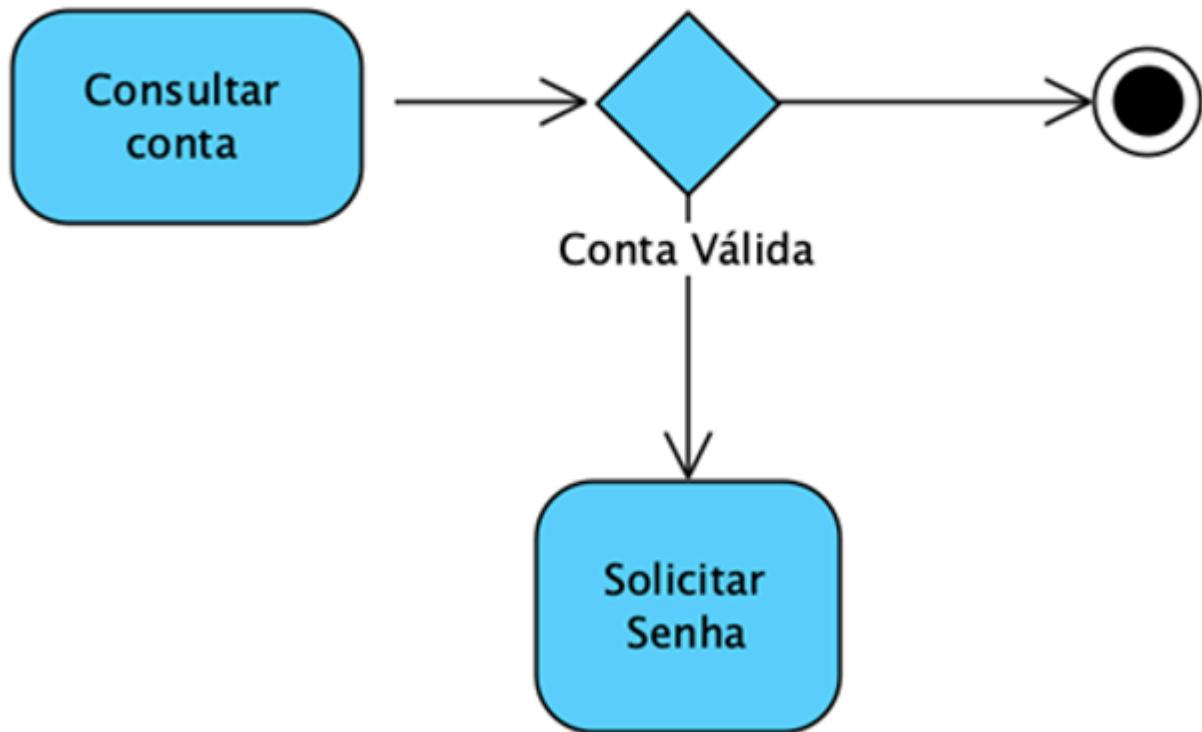


Figura 5 | Exemplo de nó de decisão. Fonte: elaborada pela autora.

Neste exemplo, continuamos a atividade que representa o processo de emissão de saldo, onde, após a consulta da conta, uma decisão deve ser tomada, representada por um losango. Se a conta for inválida, o processo é encerrado; caso contrário, é solicitada a senha. As condições para essa decisão são definidas por meio de condições de guarda, apresentadas entre colchetes e posicionadas sobre os dois fluxos alternativos.

Fork e Join

Um *fork* indica a separação de atividades em duas ou mais atividades concorrentes. Visualmente, é representado por uma barra horizontal preta com uma seta apontando para ela e duas ou mais setas saindo dela. Cada seta de saída representa um fluxo de controle que pode ser executado simultaneamente com os fluxos associados às outras setas que também saem do fork (PRESSMAN, 2021). Essas atividades concorrentes podem ser executadas em um único computador, usando diferentes threads, ou até mesmo em diferentes computadores.

Uma junção (*join*) é utilizada para sincronizar fluxos de controle concorrentes. Visualmente, é representada por uma barra preta horizontal com duas ou mais setas chegando e uma seta saindo. O fluxo de controle representado pela seta que sai não pode iniciar a execução até que todos os outros fluxos representados pelas setas que chegam tenham sido completados.

ANÁLISE E MODELAGEM DE SISTEMAS

A Figura 6 apresenta esses elementos.

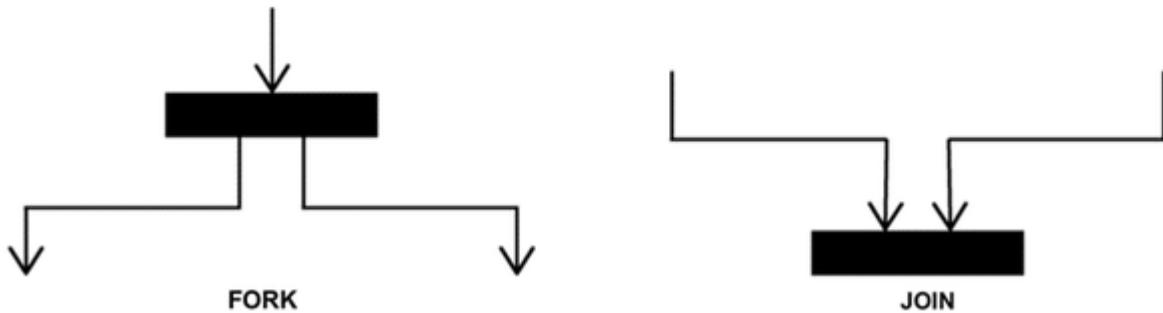


Figura 6 | Fork e Join. Fonte: elaborada pela autora.

Swimlanes

Swimlanes são divisões horizontais ou verticais em um diagrama de atividades que ajudam a visualizar a responsabilidade e a interação entre diferentes participantes ou entidades envolvidas em um processo. Cada *swimlane* representa uma unidade organizacional, um papel ou uma entidade distinta que desempenha um conjunto específico de atividades dentro do processo. Essa abordagem ajuda a tornar mais claro quem é responsável por cada etapa do processo e como as informações fluem entre as partes envolvidas. As *swimlanes* são uma ferramenta útil para modelar e analisar processos complexos, permitindo uma compreensão mais clara das interações entre os diferentes componentes do sistema (PRESSMAN, 2021). A Figura 7 apresenta um exemplo de *Swimlanes*.

ANÁLISE E MODELAGEM DE SISTEMAS

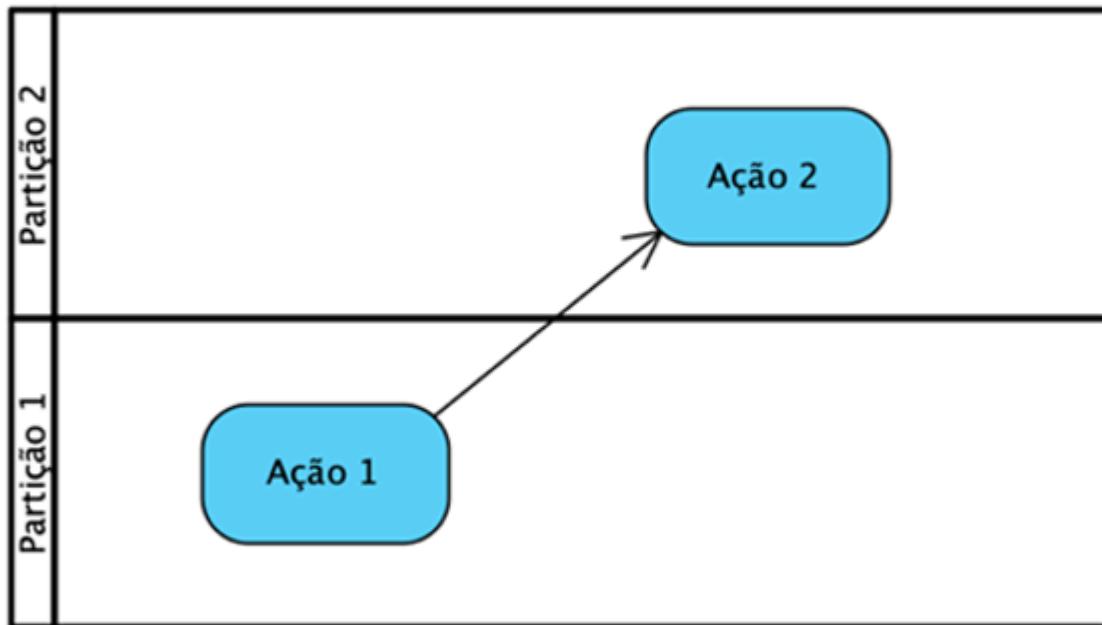


Figura 7 | Exemplo de Swimlanes. Fonte: elaborado pela autora, 2023.

Siga em Frente...

Exemplos de utilização do diagrama

Para exemplificar os elementos estudados, vamos ver o seguinte exemplo apresentado na Figura 8, que apresenta um diagrama de atividades que apresenta um processo de retirada de dinheiro em um caixa eletrônico.

ANÁLISE E MODELAGEM DE SISTEMAS

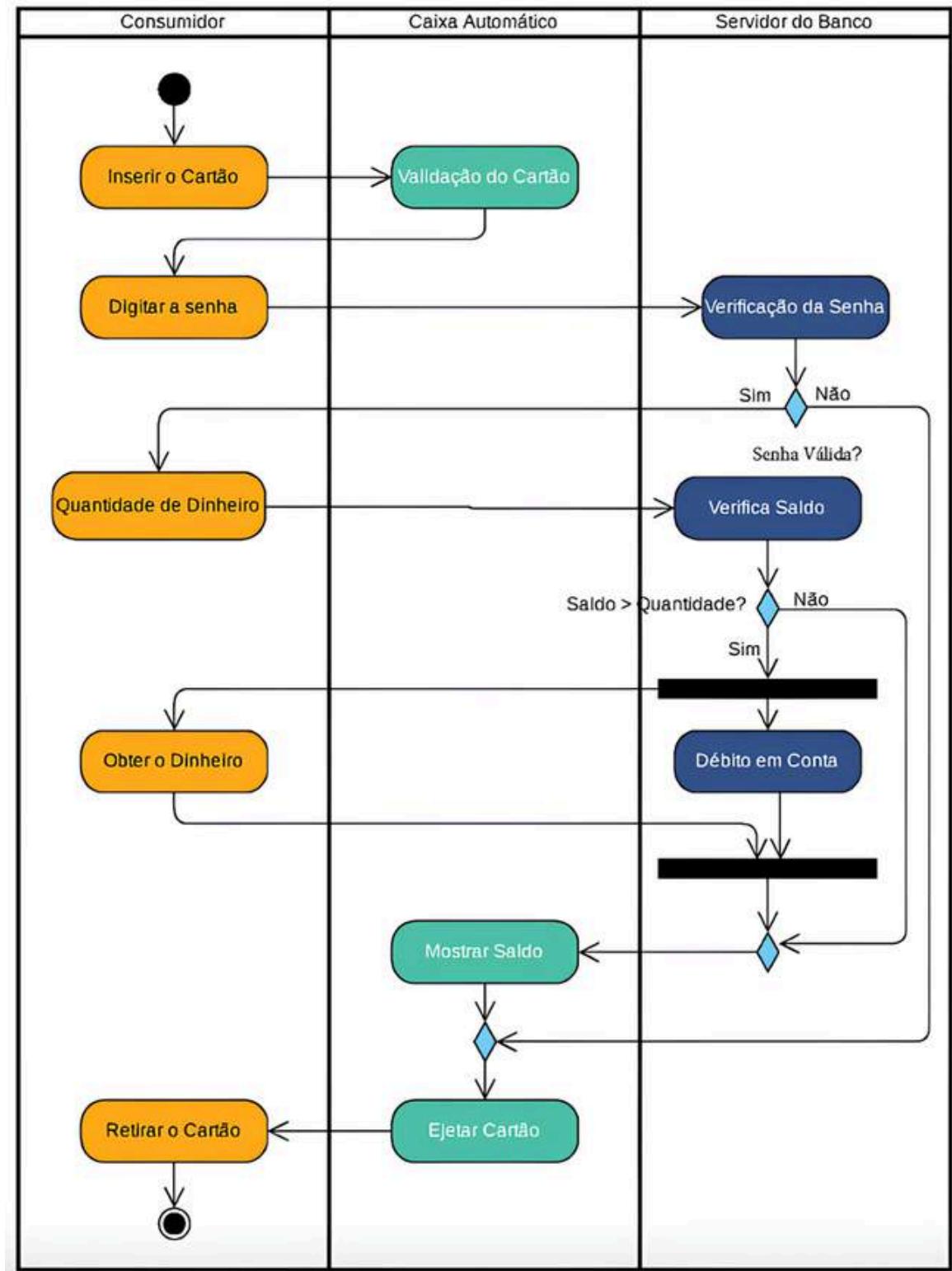


Figura 8 | Exemplo de diagrama de atividades que apresenta um processo de retirada de dinheiro em um caixa eletrônico.
Fonte: elaborada pela autora.

ANÁLISE E MODELAGEM DE SISTEMAS

Baseado no diagrama de atividades apresentado, o enunciado apropriado poderia ser o seguinte:

O diagrama descreve o processo de retirada de dinheiro em um caixa eletrônico a partir da interação entre o Consumidor, o Caixa Automático e o Servidor do Banco. Inicialmente, o consumidor insere o cartão no caixa automático, que então valida o cartão com o servidor do banco. Após a validação, o consumidor digita a senha, que é verificada pelo servidor do banco. Se a senha estiver correta, o sistema verifica o saldo da conta. Caso o saldo seja suficiente para a quantidade de dinheiro solicitada, o servidor do banco procede com o débito na conta do consumidor e o caixa automático disponibiliza o dinheiro. Em seguida, o consumidor pode optar por mostrar o saldo restante na tela do caixa automático. Por fim, o consumidor retira o cartão quando o caixa automático o ejeta, concluindo a transação. Se em algum ponto do processo a senha digitada estiver incorreta ou o saldo for insuficiente, o caixa automático não procederá com a retirada do dinheiro e ejeta o cartão, também finalizando a sessão.

Vamos Exercitar?

Nesta atividade, o você deve desenvolver um diagrama de atividades que represente uma preparação de um bolo, conforme apresentada no início da aula. Na Figura 9, está apresentado uma possível solução para o problema proposto.

ANÁLISE E MODELAGEM DE SISTEMAS

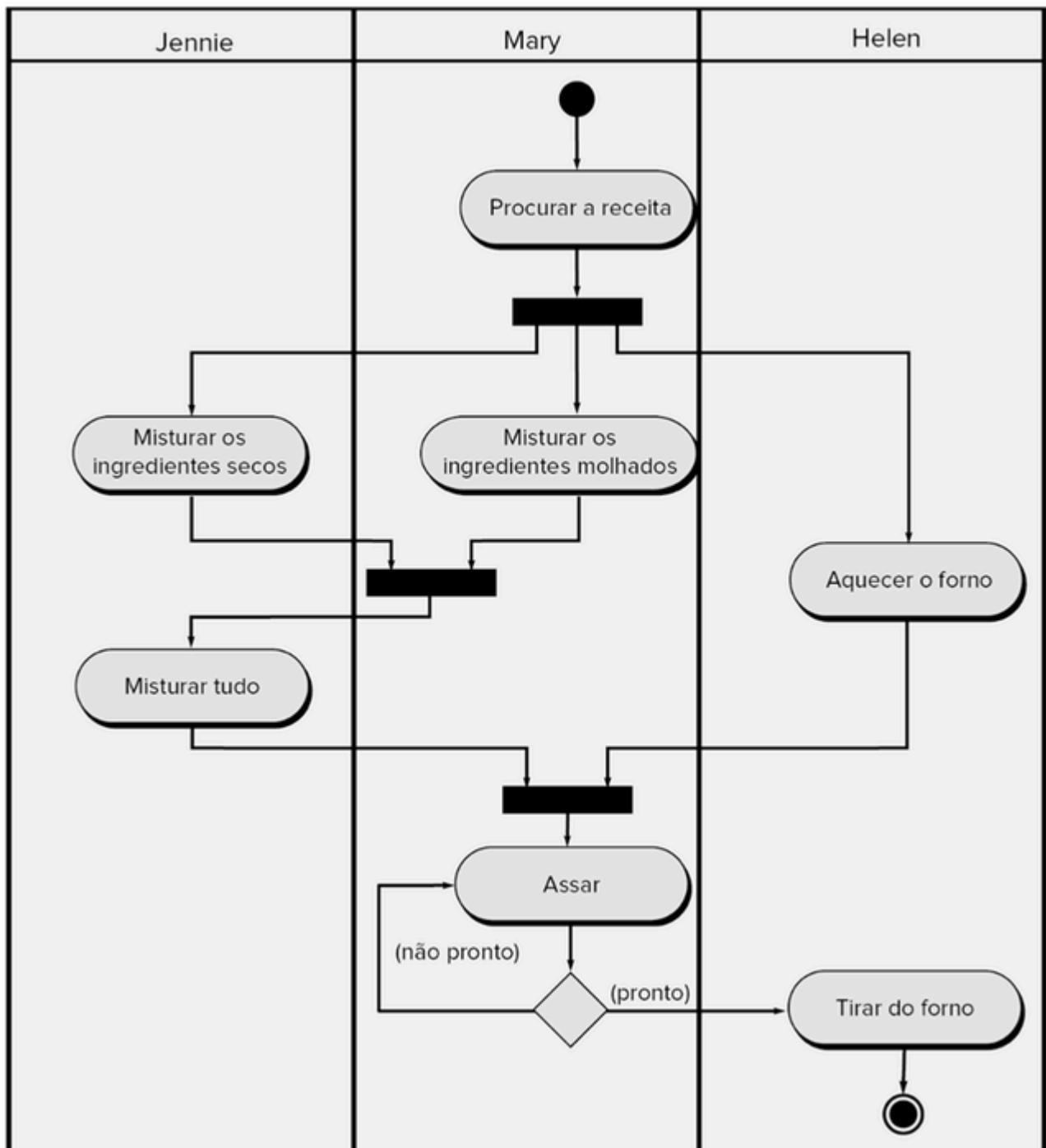


Figura 9 | Diagrama de atividades do processo de fazer um bolo. Fonte: Pressman (2021).

Saiba mais

ANÁLISE E MODELAGEM DE SISTEMAS

O livro *UML Essencial* apresenta no Capítulo 11 os principais conceitos do diagrama de atividades, além de trazer diversos exemplos deste diagrama.

FOWLER, M. [UML essencial](#). Grupo A, 2011.

Quer entender mais sobre a UML sobre como aplicar o diagrama de atividades? O livro *Utilizando UML e Padrões* traz essas explicações e exemplos no Capítulo 27.

LARMAN, C. [Utilizando UML e padrões](#). 3.ed. – Porto Alegre: Bookman, 2005.

Referências

FOWLER, M. **UML essencial**: um breve guia para a linguagem-padrão de modelagem de objetos. 3.ed. – Porto Alegre: Bookman, 2005.

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

PRESSMAN, R. S. **Engenharia de software**: uma abordagem profissional. 9. ed. – Porto Alegre: AMGH, 2021.

Aula 4

Diagrama de Classes

Diagrama de classes



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Nesta videoaula, adentraremos ao fascinante mundo do paradigma Orientado a Objetos, explorando os elementos fundamentais do Diagrama de classes e exemplos práticos de sua aplicação. Estes conteúdos são essenciais para sua prática profissional na Engenharia de Software, permitindo a modelagem e representação visual das estruturas de dados e relacionamentos entre objetos em um sistema. Ao dominar o Diagrama de classes, você estará

ANÁLISE E MODELAGEM DE SISTEMAS

preparado para projetar e desenvolver sistemas de software mais robustos e flexíveis. Prepare-se para ampliar seus conhecimentos e habilidades nesta área empolgante!

[Clique aqui](#) para acessar os slides da sua videoaula.

Bons estudos!

Ponto de Partida

Nesta aula, vamos nos concentrar na introdução ao paradigma orientado a objetos, um pilar fundamental na programação moderna e design de software. Vamos dissecar os elementos do Diagrama de Classes, uma ferramenta essencial da UML que nos ajuda a visualizar a estrutura e o design de um sistema de software.

Abordaremos os conceitos-chave da orientação a objetos, como classes, objetos, atributos, métodos e as relações entre eles, como herança, associação, agregação e composição. Esses conceitos não são apenas teóricos; eles são a base para a criação de código real e funcional que você irá escrever e interagir no seu dia a dia como desenvolvedor.

Ao analisar exemplos práticos de utilização do Diagrama de classes, você começará a ver como esses conceitos se traduzem em aplicações do mundo real. Esta compreensão é crítica, pois permite que você estruture seus projetos de software de maneira lógica e sustentável.

Para colocar em prática o que foi aprendido, desenvolva um diagrama de classes para o seguinte problema: “o sistema de vendas de veículos é projetado para capturar as relações entre carros, clientes, vendas e vendedores. Cada “Carro” é definido por atributos como placa, ano, marca, modelo e origem. Os “Clientes” são identificados por nome, telefone, endereço e CPF. As vendas, representadas pela classe “Venda”, são caracterizadas por um número de venda, data, valor e tipo. A classe “Venda” é especializada em duas formas de pagamento: “A vista” e “A_prazo”. A venda à vista inclui informações sobre o valor do desconto, o banco e o tipo, enquanto a venda a prazo detalha a quantidade e o valor das parcelas, além de conter um atributo genérico. Cada venda está associada a um único “Cliente” e um “Vendedor”, mas tanto clientes quanto vendedores podem estar ligados a várias vendas. Os vendedores são simplesmente caracterizados por um número e um nome”.

Esta aula é um investimento na sua capacidade de pensar e projetar sistematicamente, uma habilidade que será aplicada em qualquer situação de desenvolvimento de software. Então, vamos mergulhar de cabeça e começar a construir uma base sólida no paradigma orientado a objetos.

Vamos Começar!

Introdução ao paradigma orientado a objetos

ANÁLISE E MODELAGEM DE SISTEMAS

Frequentemente, você pode encontrar expressões como "Este é um novo paradigma" ou "A mudança de paradigma fez toda a diferença". Mas afinal, o que é um paradigma?

Um paradigma é considerado como um modelo previamente testado que segue princípios específicos para resolver problemas computacionais. Seguir um modelo traz uma vantagem significativa, pois facilita o desenvolvimento e a compreensão da solução encontrada.

O paradigma orientado a objetos ganhou ampla adoção a partir de 1997 com o surgimento da Linguagem de Modelagem Unificada (UML). No entanto, sua origem remonta a 1966 com a linguagem SIMULA, que introduziu o conceito de objeto. Em seguida, em 1980, a Linguagem Smalltalk 80 expandiu e aprimorou esse conceito, tratando todos os itens como objetos, o que aproximou o paradigma dos objetos físicos do mundo real. Tanto a Smalltalk 80 quanto a SIMULA suportavam recursos de orientação a objetos, como abstração, herança e vinculação dinâmica. Em 1983, surgiu a Linguagem C *with Classes*, que posteriormente, em 1986, evoluiu para o C++ (FOWLER, 2005).

Com o advento do paradigma orientado a objetos, não apenas emergiu um novo padrão para o desenvolvimento de software, mas também uma nova forma de pensar como modelar os problemas do mundo real.

A orientação a objetos então é um paradigma de programação que organiza o software em torno de objetos e suas interações. Cada objeto é uma instância de uma classe, que define seu comportamento e estrutura. Essas interações entre objetos são realizadas através de trocas de mensagens. Entre os conceitos e princípios básicos de POO (Programação Orientada a Objetos) destacaremos (GUEDES, 2011):

1. A essência do paradigma orientado a objetos reside na **abstração**. Na prática da abstração, referimo-nos a um objeto (um item do mundo real, como casa, bolo, carro, sanduíche, boleto, contrato, etc.) sem nos preocuparmos com detalhes específicos, como cor, tamanho, código ou validade, entre outros. Frequentemente, associamos a abstração à categorização do objeto. Portanto, o processo de modelagem de um objeto começa pela abstração, que ocorre quando reconhecemos os objetos. Por exemplo, ao ouvir o termo "cadeira", concebemos uma ideia geral do que é uma cadeira, e isso é um ato de abstração.
2. Uma **classe** representa essa abstração; é o ponto em que se definem as características que todas as cadeiras devem possuir e as ações que elas podem realizar.
3. As características técnicas dentro de uma classe são denominadas **atributos**, enquanto as ações ou comportamentos são chamados de **métodos**. Dessa forma, nossa concepção de uma cadeira se materializa quando criamos a classe "Cadeira" e especificamos seus atributos, por exemplo: a classe "Cadeira" pode ter atributos como pés, rodinhas, encosto, assento e cor. Além disso, ela pode executar ações como mover o encosto, elevar o assento e deslocar-se. Essas ações são seus métodos. Em relação à classe, ela é conceitualmente abstrata, mas se torna uma definição do projeto de uma cadeira. Quando uma cadeira é instanciada a partir dessa classe,

ANÁLISE E MODELAGEM DE SISTEMAS

obtemos um objeto do tipo "Cadeira", ou seja, um objeto que segue as características e comportamentos definidos pela classe "Cadeira".

4. O **objeto** é a representação concreta e única de uma classe, e a essa representação específica damos o nome de instância da classe. Portanto, um objeto é uma instância única de uma classe. Embora pertençam à mesma classe, os objetos são distintos uns dos outros devido aos valores de seus atributos e comportamentos (ações).

5. Dessa forma, uma **classe** é a abstração de um conjunto de objetos; em outras palavras, é a representação abstrata do objeto com seus atributos e comportamentos. Ao criarmos uma cadeira chamada "cadeiraCinza" a partir da classe "Cadeira" definida no item (3), obtemos um objeto do tipo "Cadeira" com cor cinza, rodinhas, pés, encosto e assento, capaz de mover o encosto e deslocar-se. A cadeiraCinza é única. Em um programa, a classe "Cadeira" existe e, ao instanciarmos a "cadeiraCinza" a partir dessa classe, ela ocupará um espaço na memória do computador, onde serão armazenados seus atributos e métodos, além de receber um endereço único que a identifica. Em resumo, o objeto é a realização concreta de um exemplar do tipo de classe que o define.

Em UML, 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 classes a partir de classes já existentes, sem duplicar nenhum código. Durante o processo de abstração, podemos definir classes amplas, que serão refinadas e estendidas durante a modelagem para formar subclasses. Essas **subclasses** podem herdar as características e comportamentos da classe mais geral, conhecida como **superclasse**. Por sua vez, a classe que herda essas características é denominada subclass, como ilustrado na Figura 1.

É importante destacar que a subclass pode adicionar novos atributos e comportamentos, além de modificar os existentes, pois ela é uma nova entidade. Tais modificações afetam exclusivamente a nova classe especificada.

Retomando o exemplo da classe "Cadeira", poderíamos criar uma classe genérica "Cadeira" contendo apenas os atributos "encosto", "assento" e "pés" – esta seria nossa superclasse, ou classe pai. Em seguida, aproveitando as características dessa superclasse "Cadeira", poderíamos construir uma nova classe "CadeiraGiratória", acrescentando a ela o atributo "rodas" e os métodos "andar", "girar" e "mover assento". Para indicar que a classe "CadeiraGiratória" herda as características da classe "Cadeira", usamos uma seta com a ponta vazada apontando para a classe "CadeiraGiratória", conforme mostrado na Figura 4.1. A classe "CadeiraGiratória" é uma subclass da classe "Cadeira", também conhecida como classe filha.

Outro exemplo de herança apresentado na Figura 1 é a classe "CadeiraExecutiva", que herda as características da classe "Cadeira" e adiciona o atributo "braço" e o método "balanço".

ANÁLISE E MODELAGEM DE SISTEMAS

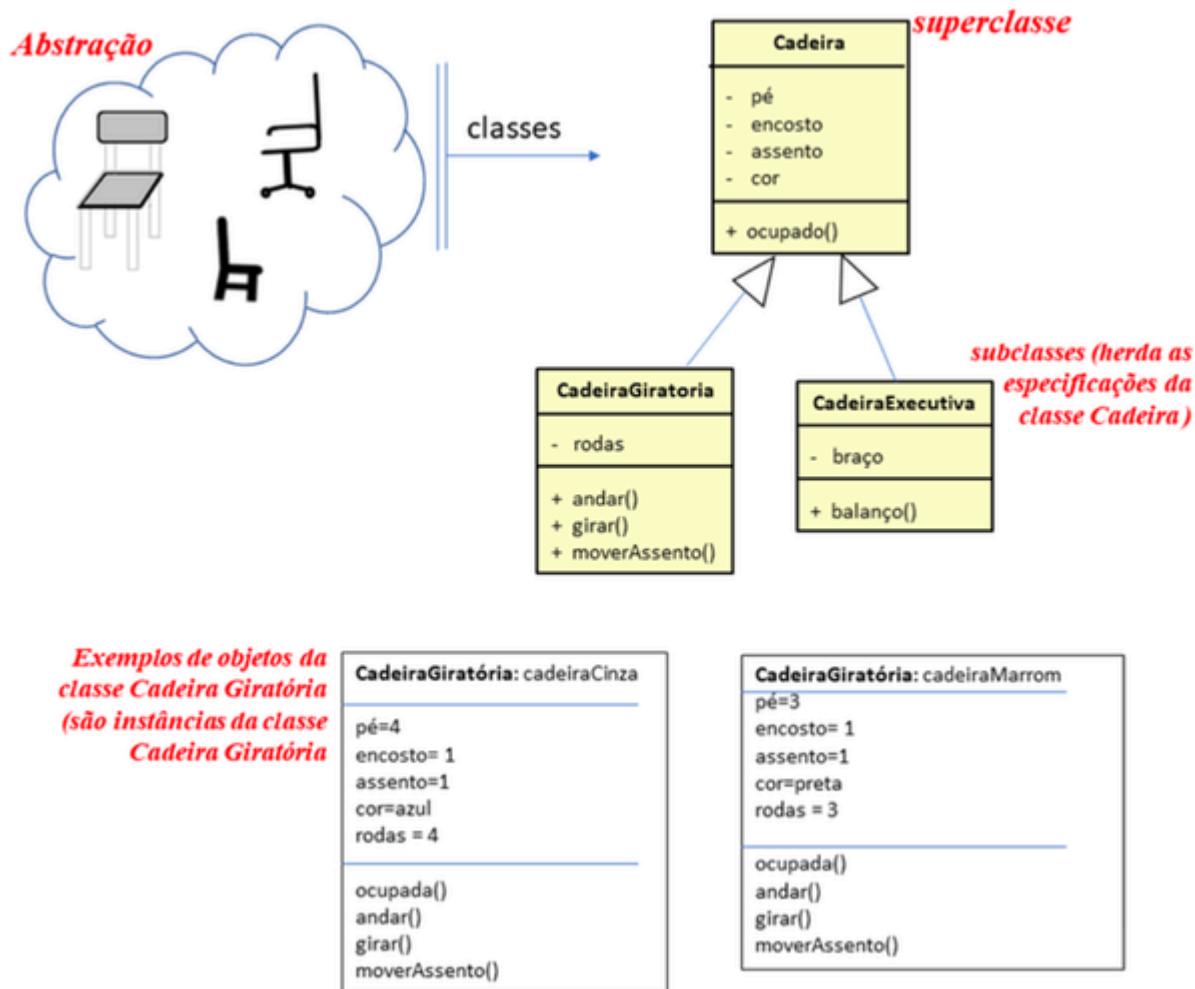


Figura 1 | POO – abstração, classe, herança e instanciação. Fonte: Werlich (2020, p. 172)

7. O **encapsulamento** é a prática de unir partes isoladas de um programa, permitindo que essas partes sejam acessadas separadamente. Na Programação Orientada a Objetos (POO), o encapsulamento tem a capacidade de ocultar ou restringir a visibilidade das informações e os detalhes da implementação dos métodos de uma classe. Em outras palavras, ele esconde do usuário da classe como uma determinada ação é realizada ou como os dados são representados. É o processo de dissimulação de todos os detalhes de um objeto que não contribuem para suas características essenciais (WERLICH, 2020).

Um exemplo prático do conceito de encapsulamento pode ser observado ao utilizar um caixa eletrônico (um objeto). Você simplesmente fornece sua identificação e senha para acesso, e os serviços são liberados para utilização. Você não precisa entender como o caixa eletrônico valida suas informações, onde elas estão armazenadas ou como a senha é descriptografada. Isso exemplifica o encapsulamento, no qual o usuário interage com o objeto sem precisar compreender os processos internos.

ANÁLISE E MODELAGEM DE SISTEMAS

No diagrama de classe, usamos o símbolo “-” (um traço) para indicar que o atributo ou o método está encapsulado. Isso significa que somente o objeto da classe pode modificá-los ou acessá-los. Por outro lado, o símbolo “+” (adição) indica que o atributo ou método é visível para outros objetos, ou seja, é público. Um exemplo desse diagrama pode ser observado na Figura 1.

8. O **polimorfismo** significa que a mesma operação [método] pode atuar de modos diversos em classes diferentes. Essa característica da Programação Orientada a Objetos (POO) confere flexibilidade às classes na construção de soluções e no reuso do código. Dentro de uma classe, um método (ou operação) pode ser definido várias vezes, pois pode ter comportamentos distintos nas subclasses.

Agora que você está familiarizado com os principais conceitos e princípios da POO, será mais fácil perceber as vantagens desse paradigma, pois ele abrange a análise, o projeto e a implementação.

A POO aplica os conceitos da Orientação a Objetos (OO) no desenvolvimento do código. Por outro lado, a Análise e Projeto Orientados a Objetos (A/POO) aplicam os conceitos da OO na análise e na elaboração do projeto, etapas que precedem a programação. Durante a análise, um instrumento utilizado é o caso de uso, enquanto no projeto é empregada a UML (Linguagem de Modelagem Unificada). Segundo Sommerville (2018), um projeto que adote o paradigma orientado a objetos deve empregar essa estratégia em todo o seu processo de desenvolvimento, observando as práticas tanto no projeto quanto na programação.

Dessa forma, podemos elencar as seguintes vantagens da Orientação a Objetos:

- Reutilização de código.
- Utilização de um único modelo conceitual para análise, projeto e implementação.
- Aceleração do tempo de desenvolvimento do software.
- Simplificação da construção de sistemas complexos, pois cada objeto é simples, fácil de testar e integrar com outros objetos. Isso prolonga o ciclo de vida do software, já que as horas dedicadas ao desenvolvimento são gastas principalmente em manutenção e adição de novas funcionalidades. O conceito de objeto e sua interação simplificam a implementação de novos recursos, aumentando assim a longevidade das aplicações.
- Potencial redução dos custos de desenvolvimento.

Por outro lado, as desvantagens do paradigma orientado a objetos são relativamente poucas:

- Exige desenvolvedores familiarizados com esse paradigma.
- A modelagem requer maior atenção em comparação com a análise estruturada.
- Necessita de uma documentação minuciosa e detalhada.

Siga em Frente...

ANÁLISE E MODELAGEM DE SISTEMAS

Elementos do diagrama de classes

O diagrama de classes é uma das ferramentas mais importantes e amplamente utilizadas na Linguagem de Modelagem Unificada (UML). Seu foco principal é possibilitar a visualização das classes que compõem o sistema, juntamente com seus atributos e métodos correspondentes. Além disso, o diagrama destaca as relações entre essas classes, mostrando como elas se complementam e trocam informações entre si. Este diagrama oferece uma visão estática da organização das classes, concentrando-se na definição da estrutura lógica das mesmas. Também serve como base para a construção de muitos outros diagramas na UML. Essencialmente, o diagrama de classes é composto pelas próprias classes e pelas associações que existem entre elas, ou seja, pelos relacionamentos entre as classes.

Alguns métodos de desenvolvimento de software, como o Processo Unificado, recomendam o uso do diagrama de classes desde a fase de análise. Nesta etapa, um modelo conceitual é elaborado para representar as informações necessárias ao software. Neste modelo, o foco está em retratar apenas as informações que o software precisará, incluindo classes, seus atributos e as associações entre elas (PRESSMAN, 2021). Detalhes como os métodos que as classes irão conter não são modelados nesta fase, pois fazem parte da implementação do software. Somente na fase de projeto é que o modelo conceitual do diagrama de classes é utilizado para produzir o modelo de domínio, que aborda diretamente a solução do problema. Os métodos necessários às classes são identificados a partir da modelagem de diagramas de interação, como o diagrama de sequência, que será abordado nos próximos capítulos.

A representação da classe é um retângulo dividido em três partes: o nome da classe; os atributos e; os métodos, como mostra a Figura 2.

ANÁLISE E MODELAGEM DE SISTEMAS

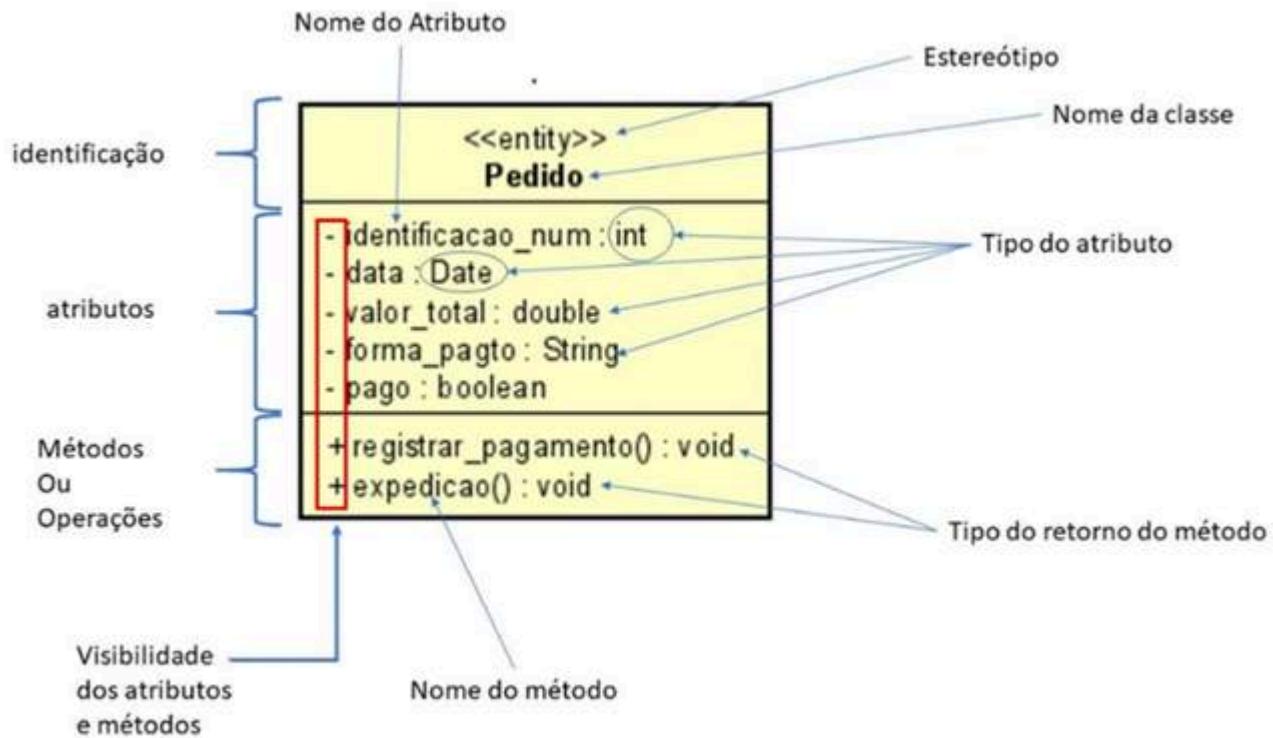


Figura 2 | Representação de uma classe em UML. Fonte: Werlich (2020, p. 206).

Os estereótipos são um recurso de extensibilidade amplamente empregado na UML, especialmente nos diagramas de classes. Sua finalidade é categorizar elementos do diagrama que compartilham características semelhantes. Os estereótipos podem ser definidos pelo desenvolvedor ou já estar predefinidos na linguagem. A UML possui 50 estereótipos predefinidos, conforme a especificação OMG® (2017), sendo que três dos mais comuns são (GUEDES, 2011):

- **<<entity>>**: identifica classes de persistência, responsáveis por armazenar dados recebidos pelo sistema. Além da notação textual, <<entity>> pode ser representada graficamente por um símbolo, como a classe "Produto" na Figura 3.
- **<<boundary>>**: identifica uma classe de fronteira, que atua na comunicação entre atores externos e o sistema. Sua representação gráfica pode ser vista na Figura 3 pela interface "Sistema_da_Loja".
- **<<control>>**: representa classes responsáveis por implementar regras de negócio. Na Figura 3, é exemplificado pela classe "controlador_Sistema_da_Loja".

As classes estabelecem diferentes tipos de relacionamentos entre si, conforme especificado na UML (GUEDES, 2011):

1. **Dependência**: este é o tipo de relacionamento mais fraco, representando uma relação semântica entre duas classes. Uma alteração na classe independente pode afetar a classe dependente. Por exemplo, na Figura 3, há uma dependência entre as classes "Produto" e

ANÁLISE E MODELAGEM DE SISTEMAS

"Receita". Embora existam independentemente, uma mudança na classe "Receita" pode impactar a classe "Produto".

2. **Associação**: este é o tipo mais comum de relacionamento, indicando que a classe A tem algum tipo de relação com a classe B. É um relacionamento genérico.
3. **Agregação**: este tipo de relacionamento é mais específico, onde a classe filha pode existir independentemente da classe pai. Na Figura 3, por exemplo, há uma agregação entre a classe "Produto" (pai) e a classe "Cobertura" (filha). Isso significa que, se um objeto do tipo "Produto" deixar de existir, o objeto do tipo "Cobertura" continuará a existir.
4. **Composição**: este é um tipo específico de associação em que, se o objeto da classe pai for destruído, não faz sentido a existência do outro objeto associado à classe filha. Por exemplo, entre as classes "Pedido" e "Item_de_Pedido", temos um relacionamento de composição, onde os "Itens_de_Pedido" são partes integrantes do "Pedido". Se excluirmos o "Pedido", os "Itens_de_Pedido" também devem ser excluídos. Este relacionamento é representado na Figura 3
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 3.
6. **Multiplicidade**: indica quantas instâncias dos objetos estão envolvidos na associação, como está mostrado na Figura 3.

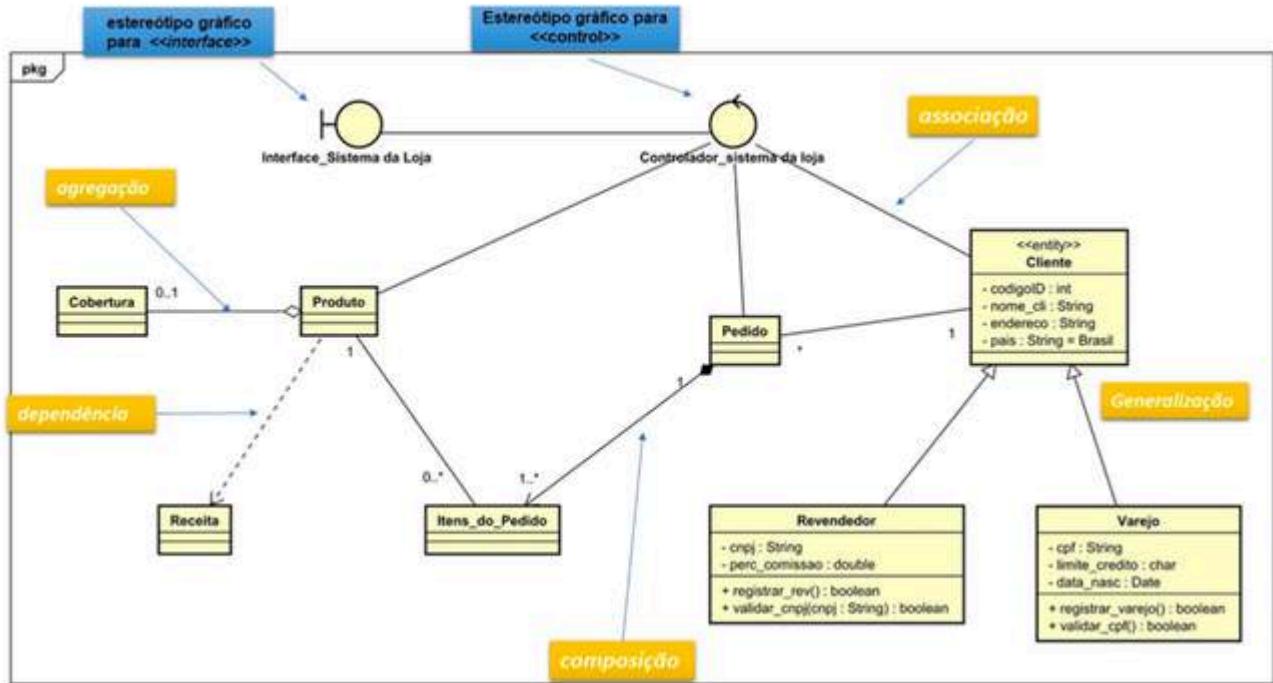


Figura 3 | Diagrama de classe representando a venda de bolos com as indicações dos tipos de relacionamentos entre as classes. Fonte: Werlich (2020, p. 208).

Exemplo de utilização do diagrama

ANÁLISE E MODELAGEM DE SISTEMAS

Para exemplificar e explicar os elementos de um diagrama de classes, analise a Figura 4, que apresenta uma modelagem de um sistema de pedidos.

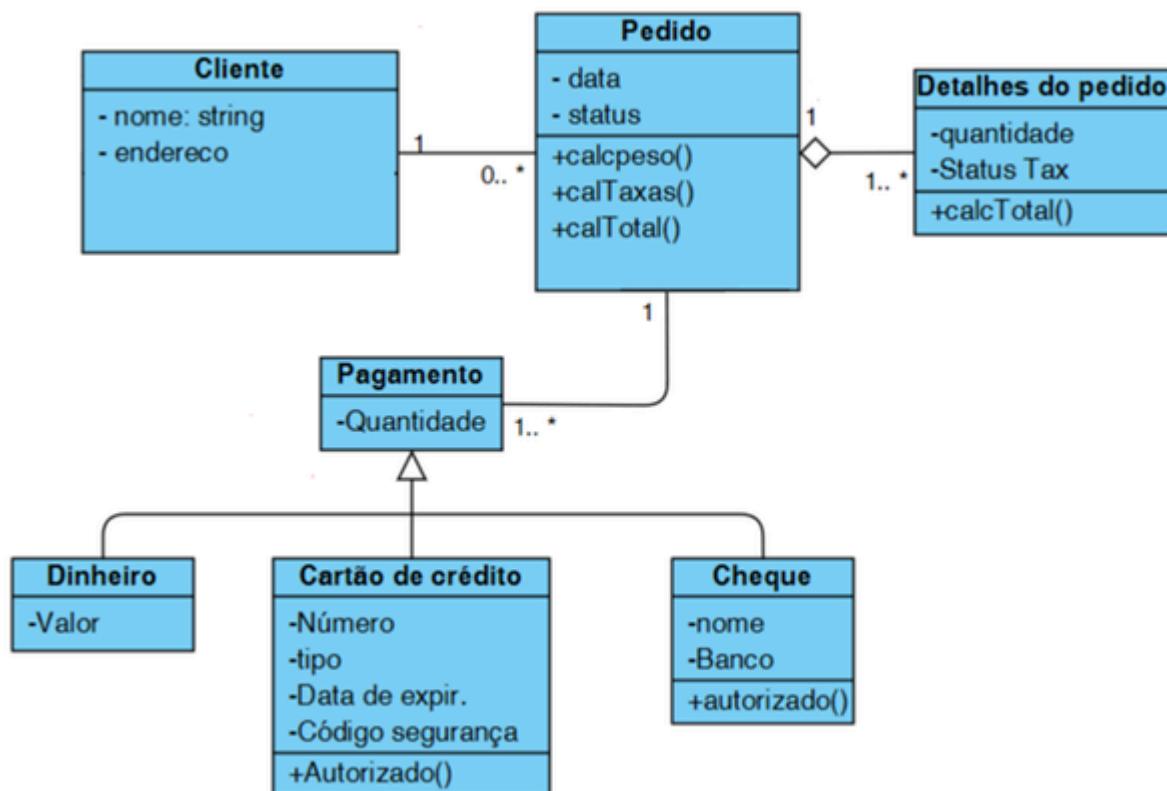


Figura 4 | Diagrama de classes. Fonte: elaborada pela autora.

O diagrama de classes apresentado descreve um sistema de pedidos com foco na interação entre clientes, pedidos e formas de pagamento. No sistema, cada “Cliente”, identificado por nome e endereço, pode fazer vários “Pedidos”. Cada “Pedido” tem uma data, um status e está associado a um ou mais “Detalhes do pedido”, que descrevem a quantidade de itens pedidos e o status fiscal. Os pedidos têm métodos para calcular peso, impostos e o total do pedido.

Para realizar o pagamento de um pedido, são oferecidas múltiplas formas de pagamento, representadas pela classe “Pagamento”, que é uma classe generalizada para “Dinheiro”, “Cartão de crédito” e “Cheque”. O pagamento em “Dinheiro” é simples, tendo apenas o valor como atributo. O “Cartão de crédito” inclui detalhes como número, tipo, data de expiração e código de segurança, e possui uma operação para verificar se a transação está autorizada. Da mesma forma, “Cheque” tem atributos para nome e banco e uma operação para validação. O diagrama ilustra como diferentes formas de pagamento são integradas ao processo de finalização de pedidos no sistema.

ANÁLISE E MODELAGEM DE SISTEMAS

Vamos Exercitar?

Este diagrama destaca a estrutura de dados que fundamenta o sistema de vendas veículos, conforme apresentado no início da aula, que permite que desenvolvedores e analistas compreendam as relações e atributos necessários para implementar as funcionalidades desejadas no sistema de gerenciamento de vendas de veículos.

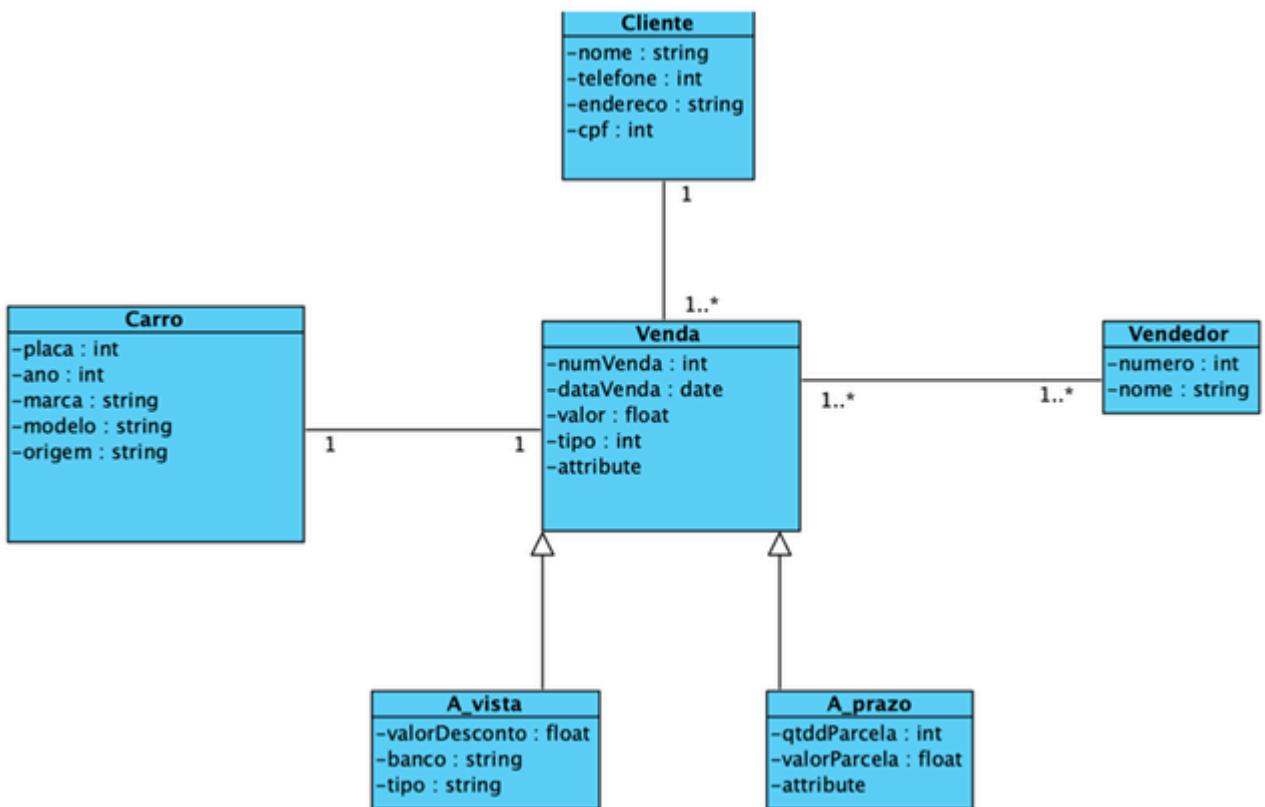


Figura 5 | Diagrama de classes para o sistema de venda de veículos. Fonte: elaborada pela autora.

Saiba mais

O Capítulo 6 do livro *Análise e Design Orientados a Objetos para Sistemas de Informação* apresenta os fundamentos dos diagramas de classes e relaciona isso com exemplos em programação.

WAZLAWICK, R. S. [Análise e Design Orientados a Objetos para Sistemas de Informação: Modelagem com UML, OCL e IFML](#). Grupo GEN, 2014.

O livro *UML Essencial* apresenta os principais conceitos do diagrama de classes no Capítulo 5.

ANÁLISE E MODELAGEM DE SISTEMAS

FOWLER, M. [UML essencial](#). 3.ed. – Porto Alegre: Bookman, 2005.

Quer entender mais sobre o diagrama de classes? O livro *Utilizando UML e Padrões* traz esses padrões de maneira clara e simplificada no Capítulo 16.

LARMAN, C. [Utilizando UML e padrões](#). Grupo A, 2011.

Referências

FOWLER, M. **UML essencial**: um breve guia para a linguagem-padrão de modelagem de objetos. 3.ed. – Porto Alegre: Bookman, 2005.

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

PRESSMAN, R. S. **Engenharia de software**: uma abordagem profissional. 9. ed. – Porto Alegre: AMGH, 2021.

SOMMERVILLE, I. **Engenharia de software**. 10. ed. São Paulo: Pearson, 2018.

WERLICH, C. **Análise e modelagem de sistemas**. Londrina: Editora e Distribuidora Educacional S.A., 2020.

Aula 5

Encerramento da Unidade

Videoaula de Encerramento



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Nesta videoaula, exploraremos os princípios da UML, com foco nos Diagramas de Casos de Uso e Atividades. Além disso, abordaremos a Modelagem de Sistemas e o Diagrama de Classes. Esses conteúdos são cruciais para a prática profissional, permitindo a visualização, análise e

ANÁLISE E MODELAGEM DE SISTEMAS

comunicação eficaz dos sistemas. Dominar essas técnicas de modelagem capacita os profissionais a projetar e desenvolver sistemas de alta qualidade. Prepare-se para aprofundar seus conhecimentos e habilidades!

[Clique aqui](#) para acessar os slides da sua videoaula.

Bons estudos!

Ponto de Chegada

Olá, estudante! Para desenvolver a competência desta Unidade, que é elaborar diagramas de acordo com as especificações da UML, você deverá primeiramente conhecer os conceitos fundamentais da Linguagem de Modelagem Unificada, conhecida como UML. Esta é uma linguagem visual que se tornou padrão na indústria de engenharia de software para modelar sistemas baseados em orientação a objetos. Diferentemente de uma linguagem de programação, a UML é uma notação para auxiliar engenheiros de software a definir as características do sistema, abrangendo desde requisitos até aspectos físicos do hardware onde o sistema será implantado. A UML é flexível, podendo ser adaptada a diversos processos de desenvolvimento de software, conforme a preferência do engenheiro (FOWLER, 2005).

Modelar software é essencial devido à sua complexidade e à constante evolução dos sistemas de informação. A modelagem não se limita à documentação, mas também oferece vantagens como facilitar a compreensão do sistema, estimar custos e identificar mudanças necessárias ao longo do tempo. Um modelo de software representa uma perspectiva abstrata do sistema, detalhando aspectos estruturais e comportamentais conforme o propósito do modelo, seja identificar requisitos essenciais, definir classes ou especificar métodos para resolver problemas.

A UML 2.5.1 oferece uma variedade de diagramas para modelagem de software, divididos em estruturais e comportamentais, permitindo aos desenvolvedores representarem e comunicarem diferentes aspectos dos sistemas de software (PRESSMAN, 2021). Embora não haja uma ordem prescrita para a criação e utilização dos diagramas, geralmente eles seguem uma sequência didática, começando pelos mais simples, como os diagramas estruturais, e avançando para os comportamentais, incluindo os de interação, para facilitar o processo de modelagem e compreensão do sistema.

Os diagramas estruturais da UML revelam a organização e interconexões de um sistema em suas partes constituintes. Eles fornecem uma visão estática da estrutura do sistema e são fundamentais na fase inicial do projeto de arquitetura. Entre os seis tipos de diagramas estruturais, o diagrama de classes se destaca como um dos mais essenciais, delineando atributos, métodos e relacionamentos entre as classes. Outros diagramas, como o de pacotes, componentes e instalação, oferecem representações específicas das partes do sistema, incluindo subsistemas, componentes de código e requisitos de hardware e software.

Os diagramas estruturais da UML, como o de classes, pacotes, componentes e instalação, entre outros, proporcionam uma visão estática e detalhada da organização e interconexões de um

ANÁLISE E MODELAGEM DE SISTEMAS

sistema, sendo cruciais na fase inicial do projeto de arquitetura. Enquanto o diagrama de classes delineia as características e relacionamentos das classes do sistema, o diagrama de componentes visualiza os módulos de código-fonte e bibliotecas. Já o diagrama de instalação descreve a configuração necessária de hardware e software para a execução do sistema, oferecendo uma visão abrangente da infraestrutura necessária (FOWLER, 2005).

Já os diagramas comportamentais da UML visam representar o fluxo de informações e eventos ao longo do tempo em um sistema, evidenciando o comportamento dinâmico dos objetos e suas respostas a ações específicas ou eventos. Dentro desse conjunto, os diagramas de interação, como o diagrama de sequência e o diagrama de comunicação, modelam interações dentro do sistema ou entre seus componentes, enquanto o diagrama de casos de uso destaca os atores e as funcionalidades do sistema de forma mais abrangente e informal, servindo como base para outros diagramas (GUEDES, 2011).

Esses tipos de diagramas oferecem uma visão dinâmica do sistema, incluindo representações detalhadas das interações e do fluxo de controle durante a execução de atividades. Enquanto o diagrama de atividade descreve os passos necessários para concluir uma atividade específica, o diagrama de máquina de estados ilustra o comportamento de um elemento através de transições de estado. Além disso, o diagrama de tempo destaca a evolução do estado de uma instância ao longo do tempo em resposta a eventos externos, fornecendo uma compreensão mais abrangente do comportamento do sistema.

Outro ponto importante para alcançar a competência da unidade é conhecer e saber aplicar o diagrama de casos de uso. Esse diagrama é uma ferramenta fundamental para compreender as funcionalidades do sistema de forma abstrata e acessível, destacando a perspectiva do usuário e simplificando a compreensão do comportamento externo do sistema. Ele é amplamente utilizado nas fases iniciais de modelagem do sistema, especialmente durante o levantamento e análise de requisitos, servindo como base para outros diagramas ao longo do processo de engenharia. Ao oferecer uma visão panorâmica das funcionalidades desejadas pelos usuários sem entrar em detalhes de implementação, o diagrama de casos de uso ajuda a identificar, especificar e documentar os requisitos do sistema, além de auxiliar na identificação dos diferentes tipos de usuários e suas interações com o sistema.

O diagrama de casos de uso da UML é composto por elementos essenciais que capturam os requisitos e interações do sistema. Os atores representam os papéis desempenhados pelos usuários e entidades externas que interagem com o sistema, enquanto os casos de uso descrevem as funcionalidades ou serviços que o sistema oferece aos atores. Os atores podem ser usuários comuns, hardware especializado ou outros sistemas integrados, representados por símbolos de "bonecos magros" no diagrama (GUEDES, 2011). Os casos de uso, por sua vez, identificam e documentam as diferentes atividades essenciais do sistema, divididos em primários e secundários, e representados por elipses com uma descrição concisa dentro.

Além dos atores e casos de uso, as associações são fundamentais para representar as interações no diagrama de casos de uso. Essas associações incluem relacionamentos de

ANÁLISE E MODELAGEM DE SISTEMAS

inclusão, extensão e generalização/especialização. A inclusão é usada para descrever rotinas comuns a múltiplos casos de uso, evitando a redundância na documentação, enquanto a extensão trata de cenários opcionais que ocorrem sob circunstâncias específicas. A generalização/especialização relaciona casos de uso genéricos a casos de uso especializados, permitindo herdar características comuns e associações entre eles.

Outro diagrama importante da UML é o diagrama de atividades. Este diagrama anteriormente era considerado uma variação do diagrama de gráfico de estados. O diagrama de atividade na UML foi promovido à sua própria categoria a partir da versão 2.0, deixando de se basear em máquinas de estados para se fundamentar em redes de Petri. Esse diagrama destaca-se por sua ênfase na sequência e nas condições para coordenar comportamentos de baixo nível, tornando-se um dos mais detalhados e focados em nível de algoritmo na UML (PRESSMAN, 2021). Comumente associado aos antigos fluxogramas, o diagrama de atividade é utilizado para modelar atividades que podem variar de métodos e algoritmos a processos completos, sendo aplicável tanto para computação procedural quanto para modelagem organizacional e sistemas de informação.

Cada atividade, que pode ser um método, algoritmo ou processo, é composta por um conjunto de ações representando os passos necessários para sua conclusão. Embora uma atividade deva conter ações, estas não precisam estar todas representadas dentro da própria atividade no diagrama, permitindo referências a atividades já modeladas ou economia de espaço. Além disso, o diagrama de atividade é capaz de modelar dois tipos de fluxo: o de controle, que descreve a ordem das ações, e o de objetos, que representa como os objetos são passados entre as ações durante a execução da atividade.

A criação dos diagramas de atividades geralmente segue uma abordagem de cima para baixo e envolve vários elementos principais, incluindo estados iniciais e finais, atividades, nós de ação, fluxo de controle, nós de decisão, *forks*, *joins*, e *swimlanes*. Os estados iniciais e finais marcam o início e o término do fluxo de uma atividade, representados por símbolos específicos. As atividades representam as etapas ou processos no diagrama, enquanto os nós de ação são os elementos fundamentais dentro de uma atividade, cada um representando uma etapa a ser executada. O fluxo de controle é estabelecido por conexões entre os nós, indicando a ordem de execução das ações, podendo conter descrições ou condições (GUEDES, 2011).

Os nós de decisão são utilizados para escolher entre diferentes caminhos de fluxo, com base em condições especificadas, e podem também ser usados para reunir fluxos divergentes. *Forks* indicam a separação de atividades em múltiplos fluxos concorrentes, enquanto *joins* sincronizam esses fluxos, garantindo que a execução prossiga somente quando todos os caminhos concorrentes forem completados. As *swimlanes*, por sua vez, são divisões horizontais ou verticais que ajudam a visualizar a responsabilidade e a interação entre diferentes participantes ou entidades envolvidas em um processo, tornando mais clara a compreensão das interações e fluxos de informação. Esses elementos combinados proporcionam uma representação detalhada e comprehensível do comportamento e da lógica de um processo ou sistema.

ANÁLISE E MODELAGEM DE SISTEMAS

O diagrama mais utilizado da UML é o diagrama de classes. Para compreender esse diagrama é indispensável conhecer o paradigma orientado a objetos. Um paradigma é um modelo testado que segue princípios específicos para resolver problemas computacionais, simplificando o desenvolvimento e a compreensão das soluções encontradas. O paradigma orientado a objetos, amplamente adotado desde o surgimento da UML em 1997, organiza o software em torno de objetos e suas interações, cada objeto sendo uma instância de uma classe que define sua estrutura e comportamento. A abstração, a herança, o encapsulamento e o polimorfismo são conceitos fundamentais desse paradigma, proporcionando uma nova maneira de abordar problemas do mundo real e facilitando o desenvolvimento de software.

Na Programação Orientada a Objetos (POO), a abstração é essencial, representada por classes que definem características e comportamentos de objetos, enquanto o encapsulamento oculta detalhes de implementação, permitindo que partes isoladas do programa sejam acessadas separadamente. A herança possibilita a criação de classes a partir de outras já existentes, enquanto o polimorfismo confere flexibilidade às classes ao permitir que uma mesma operação tenha comportamentos distintos em diferentes classes. Adotar o paradigma orientado a objetos não só estabelece um novo padrão para o desenvolvimento de software, mas também muda a forma como problemas são modelados, abrangendo análise, projeto e implementação de maneira integrada, embora exija atenção à modelagem e documentação detalhada.

O diagrama de classes na UML é uma ferramenta essencial para visualizar a estrutura estática de um sistema de software, destacando as classes, seus atributos e métodos, bem como as relações entre elas. Utilizado desde a fase de análise até o projeto, o diagrama de classes concentra-se na representação conceitual das informações necessárias ao software, sendo a base para a construção de outros diagramas na UML. As classes são representadas por retângulos divididos em três partes: nome da classe, atributos e métodos, enquanto os relacionamentos entre classes, como dependência, associação, agregação, composição e herança, são definidos para mostrar como as classes interagem entre si.

Além disso, os estereótipos são amplamente empregados para categorizar elementos do diagrama de classes, como classes de persistência, fronteira e controle. Os tipos de relacionamentos entre as classes, como dependência, associação, agregação, composição, generalização/especialização e multiplicidade, são especificados para indicar a natureza das interações entre as classes. Por meio desses elementos e relacionamentos, o diagrama de classes oferece uma representação clara e abrangente da estrutura e das interações do sistema de software, facilitando o desenvolvimento e a compreensão do software durante todo o ciclo de vida do projeto (GUEDES, 2011).

A compreensão e aplicação da Linguagem de Modelagem Unificada (UML) e seus diagramas, como casos de uso, atividades e classes, é fundamental para o desenvolvimento eficaz e eficiente de software. Essas ferramentas oferecem uma representação visual clara e concisa dos requisitos do sistema, das interações entre os usuários e o software, das atividades necessárias para alcançar determinados objetivos e da estrutura lógica do software em si. Ao aplicar adequadamente os conceitos e técnicas da UML, os desenvolvedores podem comunicar de

ANÁLISE E MODELAGEM DE SISTEMAS

forma mais eficaz as necessidades do cliente, planejar e projetar sistemas de software de maneira abrangente e construir soluções que atendam às expectativas e requisitos do usuário final.

É Hora de Praticar!



Este conteúdo é um vídeo!

Para assistir este conteúdo é necessário que você acesse o AVA pelo computador ou pelo aplicativo. Você pode baixar os vídeos direto no aplicativo para assistir mesmo sem conexão à internet.

Os diagramas de casos de uso são uma ferramenta essencial na modelagem de sistemas, pois ajudam a capturar os requisitos do sistema de forma clara e concisa. Eles descrevem como os usuários interagem com o sistema em diferentes cenários, identificando suas necessidades e objetivos. Sabendo disso, leia o enunciado a seguir e elabore um diagrama de casos de uso.

O diagrama de casos que será desenvolvido deve ilustrar o fluxo de operações em um sistema de gerenciamento de vendas. O sistema envolve três atores principais: Gerente Comercial, Gerente e Vendedor, cada um com responsabilidades específicas e interações com o sistema de contabilidade. O objetivo do sistema é facilitar e registrar as atividades comerciais, garantindo que os processos de negócio sejam conduzidos de maneira eficiente e controlada.

O Gerente Comercial é responsável por “Estabelecer Limites” para as operações de vendas, assegurando que haja um controle sobre as margens comerciais e os riscos associados. Ele também desempenha a função de “Analizar Riscos” para avaliar potenciais problemas que possam afetar as transações comerciais.

O gerente tem a tarefa de “Fechar Preço” nas negociações, o que é um passo crítico no processo de venda. Esta ação inclui a avaliação do negócio de modo obrigatório.

O Vendedor é o ator que realiza a “Atualização de Contas” no Sistema de Contabilidade e também “Registrar Negócio”, o que indica que o vendedor interage diretamente com o sistema para garantir que todas as informações de vendas sejam devidamente documentadas e contabilizadas.

O Sistema de Contabilidade é um ator secundário que serve como um repositório para registrar as transações e manter as contas atualizadas. Este sistema é essencial para manter a integridade dos dados financeiros e para o funcionamento geral do processo de vendas.

- Considerando o investimento significativo de tempo e recursos que as empresas dedicam à modelagem de software, como você avalia o papel da UML nesse processo?
- É possível que a complexidade inerente à UML possa, em certos contextos, aumentar a carga de trabalho ao invés de simplificá-la?

ANÁLISE E MODELAGEM DE SISTEMAS

- Qual é a importância de selecionar os diagramas adequados da UML durante o processo de desenvolvimento de software?
- Como as empresas podem tomar decisões informadas ao escolher os diagramas mais apropriados para seus projetos específicos, levando em consideração as necessidades do cliente, a complexidade do sistema e os recursos disponíveis?

Dê o Play!

[Clique aqui](#) para acessar os slides do Dê o play!

Uma possível solução para o exercício está apresentada a seguir:

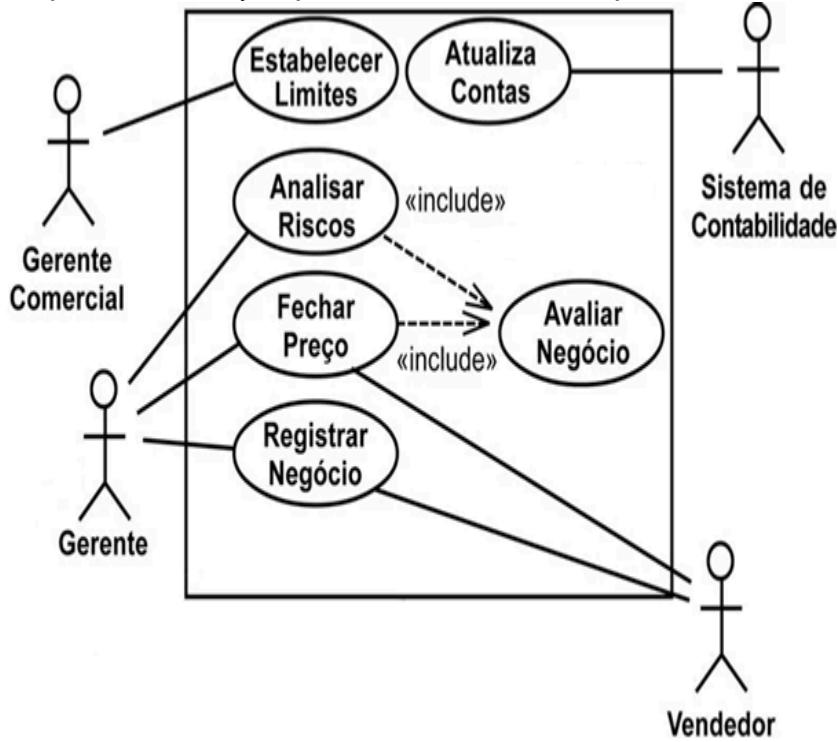
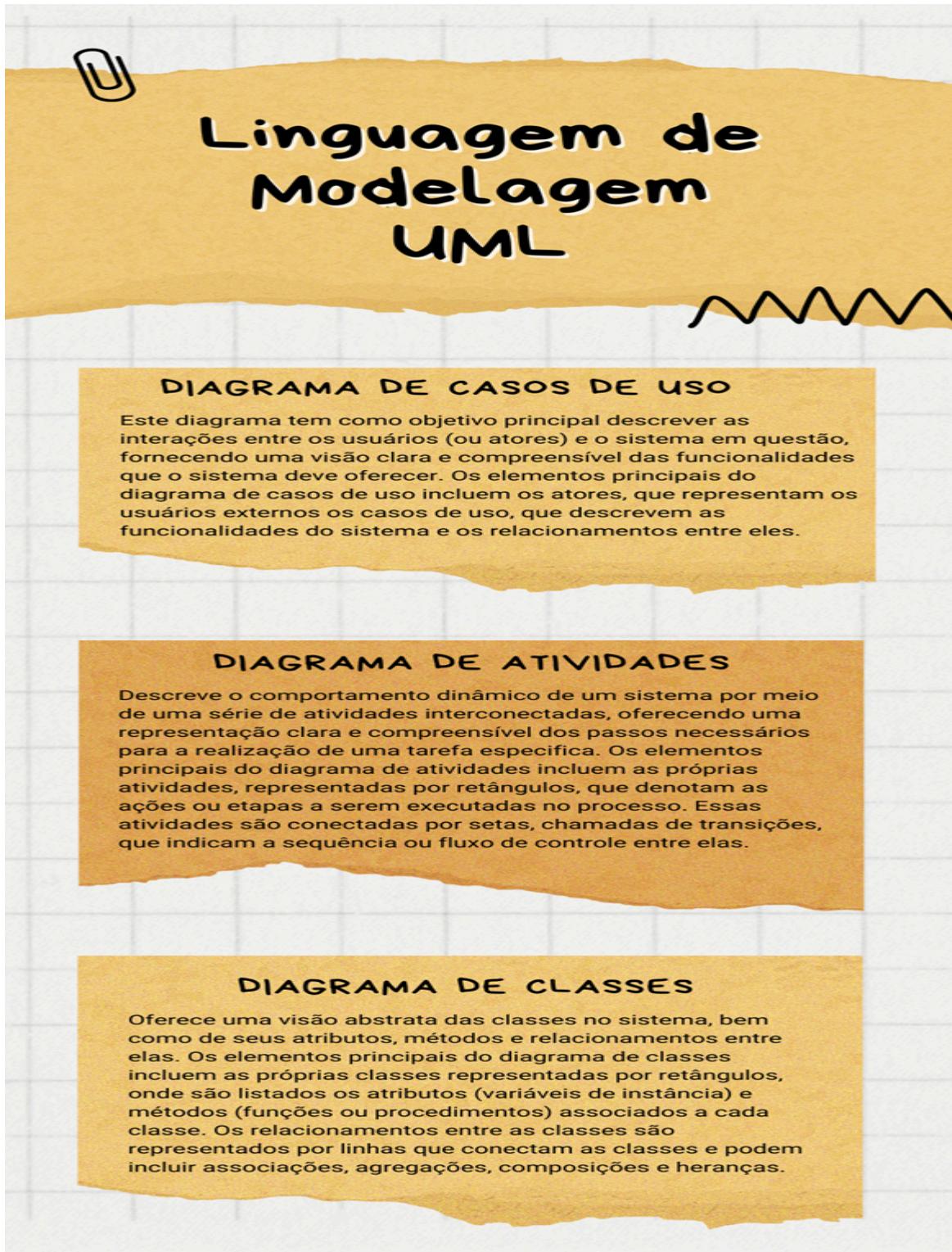


Figura 1 | Resolução. Fonte: Fowler (2005, [s. p.]).

Desvende a essência da modelagem de sistemas com a UML. Este infográfico apresenta a descrição dos diagramas de casos de uso, a dinâmica dos processos com diagramas de atividades, e a estrutura sólida do sistema com diagramas de classes.

ANÁLISE E MODELAGEM DE SISTEMAS



Linguagem de Modelagem UML

DIAGRAMA DE CASOS DE USO

Este diagrama tem como objetivo principal descrever as interações entre os usuários (ou atores) e o sistema em questão, fornecendo uma visão clara e compreensível das funcionalidades que o sistema deve oferecer. Os elementos principais do diagrama de casos de uso incluem os atores, que representam os usuários externos os casos de uso, que descrevem as funcionalidades do sistema e os relacionamentos entre eles.

DIAGRAMA DE ATIVIDADES

Descreve o comportamento dinâmico de um sistema por meio de uma série de atividades interconectadas, oferecendo uma representação clara e compreensível dos passos necessários para a realização de uma tarefa específica. Os elementos principais do diagrama de atividades incluem as próprias atividades, representadas por retângulos, que denotam as ações ou etapas a serem executadas no processo. Essas atividades são conectadas por setas, chamadas de transições, que indicam a sequência ou fluxo de controle entre elas.

DIAGRAMA DE CLASSES

Oferece uma visão abstrata das classes no sistema, bem como de seus atributos, métodos e relacionamentos entre elas. Os elementos principais do diagrama de classes incluem as próprias classes representadas por retângulos, onde são listados os atributos (variáveis de instância) e métodos (funções ou procedimentos) associados a cada classe. Os relacionamentos entre as classes são representados por linhas que conectam as classes e podem incluir associações, agregações, composições e heranças.

Figura | Linguagem de Modelagem UML. Fonte: elaborada pela autora.

GUEDES, G. T. A. UML 2: uma abordagem prática. 2. ed. - São Paulo: Novatec Editora, 2011.

ANÁLISE E MODELAGEM DE SISTEMAS

FOWLER, M. **UML essencial**: um breve guia para a linguagem-padrão de modelagem de objetos. 3.ed. – Porto Alegre: Bookman, 2005.

PRESSMAN, R. S. **Engenharia de software**: uma abordagem profissional. 9. ed. – Porto Alegre: AMGH, 2021.