Análise e **Projetos** de **Sistemas**

Carla Daiane Zatti

Curitiba
2016

EDITORA FAEL

Ficha Catalográfica elaborada pela Fael. Bibliotecária – Cassiana Souza CRB9/1501

Z38a Zatti, Carla Daiane

Análise e projeto de sistemas / Carla Daiane Zatti. – Curitiba: Fael, 2016.

212 p.: il.

ISBN 978-85-60531-59-2

1. Tecnologia da informação 2. Análise de sistemas I. Título

CDD 004.21

FAEL

Direção de Produção	Fernando Santos de Moraes Sarmento
Coordenação Editorial	Raquel Andrade Lorenz
Revisão	Editora Coletânea
Projeto Gráfico	Sandro Niemicz
Capa	Vitor Bernardo Backes Lopes
Imagem da Capa	Shutterstock.com/Taiga/Kurdanfell
Arte-Final	Evelyn Caroline dos Santos Betim

Sumário

Carta ao Aluno

Prezado(a) aluno(a),

Quando falamos em Análise e Projeto de Sistemas, navegamos do humano ao tecnicista. Não há como considerarmos concepção e requisitos de sistemas, bem como nosso produto final, o software, sem humanizarmos o assunto. E, ao iniciarmos um processo de análise e projeto de sistemas, precisamos de muita sintonia e sinergia com nossos *stakeholders,* nossos "usuários". Assim, ao adentrarmos os assuntos de Análise e Requisitos, vislumbraremos ideias e conceitos abordados por outras áreas, tais como antropologia, sociologia, filosofia, ergonomia, entre tantas outras que fazem parte do currículo de ciências humanas.

Pensar em sistemas de informação requer ter em mente que estamos desenvolvendo sistemas para pessoas! O nível de qualidade de nosso sistema está diretamente ligado a requisitos bem desenhados. Isso não quer dizer que podemos relaxar na parte técnica, mas que grande parte do sucesso de nosso produto final será avaliado justamente por nossos *stakeholders*, que são os que mais sabem do que é necessário no sistema, porém muitas vezes não têm habilidades para desenvolver as próprias aplicações. É nesse contexto que entra o analista de sistemas, o profissional que permeia a área de ciências humanas e a área de computação, capaz de conceber e modelar sistemas que atendam às reais necessidades dos *stakeholders*.

Para trabalhar análise e projeto de sistemas, precisamos buscar alguns assuntos relacionados a Programação Orientada a Objetos, Engenharia de Software, Banco de Dados e Gestão de Projetos, infraestrutura de TI, hardware e sistemas operacionais. Ao desbravarmos tais assuntos, compreenderemos o quão tênue é a linha que separa o humano do computador e, para tanto, o quão importante será captarmos requisitos com maestria.

Uma vez compreendidas as barreiras de comunicação e requisitos, seguiremos no conhecimento de metodologias, modelos, técnicas e ferramentas que nos auxiliarão na elaboração e na construção de sistemas de informação. É sobre essa fascinante área multidisciplinar que estudaremos nesta disciplina.

1

Introdução à Análise e Projeto de Sistemas

Planejar um sistema implica rapidez de desenvolvimento, redução de custos, menos manutenção, mais produtividade e vantagem competitiva. Para estudar análise e projeto de sistemas, é recomendado o entendimento do que é um sistema.

Do grego, o termo sistema significa combinar, ajustar, formar um conjunto. Do latim *systema*, é um conjunto ordenado de diferentes partes, elementos, regras ou atividades intelectualmente organizadas e combinadas pelo homem ou pela natureza. Esses elementos são dependentes, relacionados, interagentes e coordenados entre si e, conjunta e regularmente, operam em um ambiente no desempenho de uma determinada função. Possuem um objetivo em comum, de modo a formar uma estrutura, um todo organizado, unitário e complexo. O sistema então é um conjunto de elementos relacionados, com atributos e funções especiais, que, armazenados juntos, de forma organizada e com um propósito comum, contribuem para o objeto. Produzem resultados específicos, sempre man-

tendo a harmonia interna face ao ambiente externo. Um sistema tem regras que regem o seu funcionamento. A interação entre os elementos é essencial para que seu conjunto possa ser considerado um sistema. Essa interação entre elementos é chamada de sinergia. A sinergia é o que possibilita que um sistema funcione adequadamente.

1.1 Sistema

O conceito de sistema é utilizado tanto para definir um conjunto de conceitos como objetos reais dotados de organização. É uma definição tão abrangente que acontece em várias disciplinas, como biologia, informática, administração; em questões diversas, como a operação de uma nave espacial; e em vários contextos, como sistema econômico, sistema computacional, sistema solar ou sistema digestivo. Há muitos tipos de sistemas, como os políticos, tecnológicos, biológicos e jurídicos. O que unifica todos esses exemplos é o fato de que cada um deles possui um conjunto de elementos inter-relacionados, e que é possível identificar alguma função desempenhada pelo sistema como um todo. Essa função se torna impossível na ausência de qualquer uma das partes. Por exemplo: um sistema computacional deve atender a uma determinada necessidade de processamento de informações de usuários; o sistema solar deve manter os planetas girando em torno do sol; o sistema digestivo deve incorporar ao corpo de um animal a energia e matéria contidas em alimentos.

Além da interação entre os elementos, um sistema deve ter metas para conseguir um objetivo. Todo sistema possui um objetivo geral a ser atingido.

Existem também princípios gerais dos sistemas:

a) quanto mais especializado for um sistema, menos capaz ele será de se adaptar a circunstâncias ou condições diferentes.

b) quanto maior for um sistema, maior o número de recursos que serão destinados ou dedicados à sua manutenção diária.

c) os sistemas sempre fazem parte de sistemas maiores e sempre podem ser divididos em sistemas menores.

d) os sistemas crescem.

1.1.1 Componentes dos sistemas

Um sistema também possui alguns componentes ou funções básicas e fundamentais.

a) Entrada (*input*)

Envolve a atividade de coleta e reunião de fontes de dados brutos ou primários de dentro da organização ou de seu ambiente externo, que entram no sistema para serem processados. São a energia e os insumos transformados pelo sistema.

Exemplos: formulários, informações, matéria-prima.

b) Processamento

É o processo de transformação usado pelo sistema para converter os dados de entrada bruta (insumos retirados do ambiente) em forma de saída (produtos para consumo próprio do sistema e para serem devolvidos ao ambiente), que seja mais útil, apropriada e desejada pelo usuário.

Exemplos: dados classificados e analisados, armazenamento.

c) Saída (*output*)

Envolve a etapa da transferência da informação ou de elementos produzidos por um processo de transformação ao seu destino final (pessoas ou atividades que a usarão). É o produto ou serviço resultante do processo de transformação do sistema. É a etapa que realmente interessa ao usuário do sistema.

Exemplos: gráficos, relatórios, bens materiais.

d) Realimentação ou retroalimentação (*feedback*)

Trata-se do retorno de informações sistemáticas, dados processados ou saída sobre algum aspecto ou desempenho do sistema, que possam ser utilizados para avaliar, monitorar e fazer ajustes ou modificações. Visa auxiliar a refinar ou corrigir os dados e atividades de entrada ou do processamento e melhorar seu desempenho.

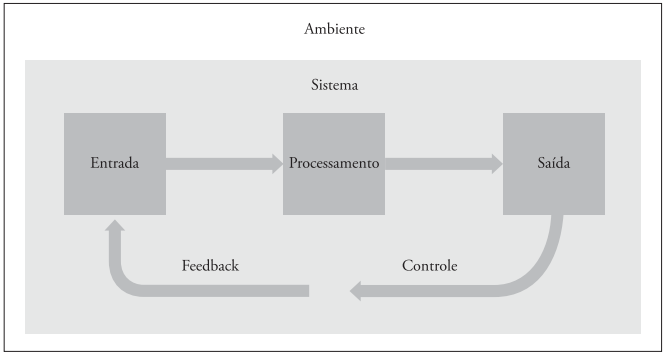
Exemplos: prazo de execução de atividades, erros de digitação.

e) Controle

Também conhecido por cibernética, envolve as atividades e processos de monitoramento e avaliação do feedback, para determinar se o sistema se dirige para a realização dos seus objetivos. Faz ainda a avaliação e os ajustes necessários aos componentes de entrada, processamento e saída do sistema, de modo a permitir as ações corretivas para garantir que seja alcançada a produção adequada.

Exemplos: testes de controle de qualidade, avaliação de desempenho, pesquisas.

Figura 1 - Sistema e seus componentes



Fonte: Elaborado pela autora. (2016)

A figura 1.1 mostra a relação entre os componentes do sistema, que acontece conforme descrito a seguir. Os sistemas recebem variáveis de entrada de dados, energia, material ou insumos. Então processam os insumos, convertendo-os em produtos, e geram uma ou mais variáveis de saída (informação, energia ou matéria), resultante do processo. Enquanto isso, o *feedback* retorna informações sistemáticas sobre algum aspecto do sistema, que possam ser utilizadas para o avaliar e monitorar, de modo a melhorar seu desempenho. Já o controle executa as atividades e os processos usados para avaliar as entradas, processamentos e saídas, de modo a permitir as ações corretivas.

1.1.2 Classificação dos sistemas

Os sistemas podem ser classificados quanto à sua constituição.

a) Sistema abstrato, conceitual ou ideal

Sistema composto por objetos não-físicos, como conceitos, planos, hipóteses e ideias, que muitas vezes só existem no pensamento das pessoas.

Exemplos: *software*, ideias.

b) Sistema concreto, físico, material ou real

Sistema composto por equipamentos, maquinaria e objetos, coisas ou entidades reais ou materiais.

Exemplos: *hardware*, objetos.

Os sistemas também podem ser classificados quanto à sua natureza.

a) Sistema aberto

Sistemas abertos se caracterizam por um intercâmbio ou interação de transações ou da organização com o ambiente. Conservam-se no mesmo estado (autorregulação), apesar de a matéria e a energia que o integram se renovarem constantemente. São completamente permeáveis à energia e à matéria. Possuem constante interação dual com o ambiente. Influenciam e são influenciados. Possuem capacidade de crescimento, mudança, adaptação ao ambiente externo para sobreviver (mudando seus produtos, técnicas e estrutura) e até autorreprodução sob certas condições ambientais. Competem com outros sistemas

Exemplos: ecossistemas, empresas.

b) Sistema fechado

Sistemas fechados possuem uma fronteira que permite troca de energia, mas não de matéria, entre eles e sua vizinhança. São sistemas que dispensam o contato com o ambiente. Analisam somente a organização interna, não visando o ambiente em que se encontram.

Exemplos: planetas, relógios.

c) Sistema isolado

Sistemas isolados são o conjunto de dois ou mais corpos isolados do ambiente externo. Não trocam nem matéria e nem energia com o ambiente, somente entre seus corpos, sendo delimitados por uma

fronteira completamente restritiva à troca de matéria, à variação de volume e ao calor. Nesses sistemas, a energia se conserva e a entropia nunca decresce.

Exemplos: o universo, garrafas térmicas.

Ainda, os sistemas podem ser classificados em:

a) sistema dinâmico

Sistemas dinâmicos são conjuntos de equações que geram uma regra fixa que descreve um ponto, um espaço geométrico. Dependem do tempo. As propriedades ou grandezas envolvem variáveis que evoluem no tempo, podendo também variar no espaço.

Exemplos: número de peixes existentes em um lago (ao longo do ano), veículos (movimento), preços de produtos.

b) sistema estático

Sistemas estáticos são a organização das normas que se relacionam entre si a partir de seus conteúdos e por meio de um sistema de sugestões lógicas. Nos sistemas estáticos, as propriedades descritivas do sistema não variam com o tempo, podendo variar no espaço.

Exemplos: vigas carregadas estaticamente, cômodos de um imóvel.

1.1.3 A teoria geral dos sistemas

O conceito de sistema é bem antigo, apesar de o termo sistema não ter sido inicialmente utilizado. Como consequência do avanço da tecnologia e da necessidade da abordagem dos sistemas, o termo sistemas vem se popularizando na sociedade moderna. Quando se percebeu que não era viável tratar as ciências por partes isoladas, a necessidade de se encontrar novos meios para realizar tarefas fez surgir novas profissões e empregos, até então desconhecidos, voltados ao enfoque sistêmico, com o objetivo de não somente realizar a tarefa pretendida, mas com o máximo de eficiência e menor custo possíveis. Essas novas áreas receberam nomes como: projeto de sistemas, análise de sistemas, engenharia de sistemas. Todas essas mudanças levam o período atual a ser associado a uma Segunda Revolução Industrial, pois os sistemas

estão presentes em todos os campos da ciência. Essa transformação ocorre na maneira de o homem pensar, que passa a enxergar grandes complexos (sistemas), reorientando assim o pensamento científico.

A Teoria Geral de Sistemas (TGS) tem como objetivo estudar a organização abstrata de fenômenos, independentemente de sua formação e configuração, e analisar a natureza dos sistemas e relação entre suas partes, bem como a relação entre eles em diferentes espaços. A ideia central é investigar todas as leis fundamentais e os princípios comuns a todas as entidades complexas e modelos que podem ser utilizados para a sua descrição, para o desenvolvimento de uma teoria de caráter geral. Assim, essa teoria pode ser aplicada a fenômenos bastante semelhantes que ocorrem em uma diversidade de campos específicos de conhecimento. Ela não busca solucionar problemas ou tentar soluções práticas, mas sim produzir formulações conceituais e referenciais que possam criar condições de aplicação na realidade empírica, e permitam a comunicação de diferentes áreas, como física, biologia e ciências sociais.

Os primeiros estudos sobre a abordagem orgânica surgiram em 1925. A teoria de sistemas foi desenvolvida em 1936 e proposta em 1937 pelo biólogo Ludwig von Bertalanffy e alcançou o seu auge de divulgação na década de 50. Os mais de trezentos trabalhos de Von Bertalanffy, baseados numa visão diferente do reducionismo científico até então aplicado pela ciência convencional, foram publicados entre 1950 e 1968. Sua ideia era a de que um organismo é um todo maior que a soma das suas partes. Ele dizia que:

> se as leis dos sistemas biológicos – que regem os processos como crescimento e adaptação – podem ser aplicados às áreas além da biologia; e se a lei da gravidade é igualmente aplicável às maçãs e aos planetas; e se a lei da probabilidade se aplica igualmente à genética e aos seguros de vida, então as leis dos sistemas biológicos, bem poderiam ser aplicáveis à psique humana, às instituições sociais, e ao conjunto global da ecosfera (DAVIDSON, 1983, p. 23).

Von Bertalanffy criticava o pensamento de que o mundo é dividido em diferentes áreas e recomendava o estudo dos sistemas de um modo global, envolvendo todas as suas interdependências. Para ele, cada um dos componentes, quando agrupados para constituir uma unidade funcional maior, desenvolvem qualidades que seus elementos não possuem quando isolados.

Ideias semelhantes à de Bertalanffy começaram a surgir em outros lugares, mostrando o início de uma nova tendência. Após a guerra, a TGS foi amplamente discutida entre físicos e em conferências. As analogias superficiais que mudavam as diferenças reais e conduziam a conclusões erradas foram o maior obstáculo para a aceitação da TGS. Entretanto, começou a ter uma aceitação maior por parte da comunidade científica à medida que as objeções feitas à teoria foram sendo derrubadas. Contribuiu com a aceitação da teoria o economista Kenneth Boulding, que escreveu uma carta a Bertalanffy, em 1953, na qual afirmava que ambos haviam chegado a uma conclusão muito semelhante quanto à TGS, embora partindo de campos científicos diferentes.

A principal característica da ciência moderna, a especialização, acaba dividindo a ciência em vários ramos e sub-ramos, prendendo o cientista em um universo privado, com pouca comunicação com outras áreas à sua volta. Pelo fato de não ter sido desenvolvida especificamente para uma única área de conhecimento, a TGS é muito importante, porque avalia a organização como um todo e não somente em departamentos ou setores. Ela está presente em todos os campos da ciência. Na biologia, por exemplo, é necessário estudar todo o sistema, e não somente as partes isoladas. Esse conceito também serve para outras áreas, pois uma mesma lei pode servir ao mesmo tempo para o campo da biologia quanto ao campo da matemática. A formulação de uma teoria sobre sistemas poderia fornecer modelos que servem para diversos âmbitos, inclusive no segmento das tecnologias e até mesmo nas ciências sociais. Deve-se tratar os fenômenos sociais contemporâneos como sendo sistemas, mesmo sabendo a complexidade das definições socioculturais dos povos atuais. Isso economizaria tempo e trabalho e aumentaria o progresso nos campos, pois pontos de vista semelhantes surgem em várias disciplinas da ciência, como também problemas que não são possíveis de analisar por partes isoladas. Em síntese, a TGS é um instrumento útil para entender a relação entre os sistemas, computacionais ou não, bem como as relações dentro de cada um deles. Também é útil para: fornecer modelos a serem utilizados em diferentes campos e transmitidos de uns para os outros; alcançar uma teoria "exata" nos campos não físicos da ciência; e conduzir à integração necessária na formação científica. Um exemplo é o corpo humano: um sistema formado de outros sistemas (sistema respiratório, digestivo, nervoso).

Existem propósitos básicos da TGS, expostos a seguir.

a) Uma nítida tendência geral para a integração nas várias ciências naturais e sociais.

b) A integração parece centralizar-se rumo a uma teoria geral dos sistemas.

c) A teoria de sistemas pode ser uma maneira importante e mais abrangente de estudar para alcançar uma teoria exata nos campos não físicos do conhecimento científico, especialmente as ciências sociais.

d) Essa teoria de sistemas, ao desenvolver princípios unificadores que atravessam verticalmente os universos particulares das diversas ciências individuais envolvidas, aproxima-nos do objetivo da unidade da ciência.

e) Isso pode conduzir a uma integração muito necessária na educação científica.

As premissas que embasam essa teoria são:

a) sistemas existem dentro de sistemas.

b) os sistemas são abertos.

c) as funções de um sistema dependem de sua estrutura.

Alguns conceitos são fundamentais para a TGS:

a) Entropia ou sinergia

Entropia ou sinergia é um conceito de unidade de medida de grandeza termodinâmica que é usada para mensurar o grau de desordem ou desorganização e irreversibilidade das partículas de um sistema físico ou termodinâmico, assim como a espontaneidade dos processos físicos que pode levar à falência de um sistema. É a parcela de energia que não pode mais ser transformada em trabalho a dada temperatura e pressão e o movimento natural que leva todas as coisas de volta à massa da Terra. O termo entropia é utilizado para explicar perdas irreversíveis. Exemplos: um corpo enterrado que é incorporado à Terra, um cubo de gelo que derrete e passa do estado sólido para o líquido. Quanto maior a desordem de um sistema, maior a sua entropia.

b) Sintropia, negentropia ou entropia negativa

Para que o sistema continue existindo, tem que desenvolver forças contrárias à entropia. Sintropia, negentropia ou entropia negativa é o contrário da entropia. É o nome dado à função de coordenação de energias que representa e mede o grau de ordem ou organização e previsibilidade das partículas existentes em um sistema. Ela tem por efeito diminuir a entropia ou o desgaste de energia e maximizar a sua utilização.

c) Homeostase

Homeostase ou homeostasia é uma das características fundamentais dos seres vivos. É um conjunto de fenômenos auto reguladores que apresentam uma situação físico-química característica e constante que interfere em um sistema aberto ou em alguns organismos. É a tendência a regular e permitir manter o estado de equilíbrio e o funcionamento correto e normal de suas variáveis essenciais ou do ambiente interno do organismo. É a condição de resistir a mudanças, dentro de determinados limites toleráveis e com relativa estabilidade. O organismo necessita dessa estabilidade para realizar suas funções adequadamente, de modo a manter uma condição estável mediante múltiplos ajustes de equilíbrio dinâmico. Esses ajustes são controlados por mecanismos de regulação inter-relacionados, para o equilíbrio e conservação de elementos fisiológicos e do metabolismo do corpo. Isso ocorre por meio de alguns mecanismos de regulação, mesmo diante de alterações impostas pelo meio ambiente. É o processo por meio do qual um organismo mantém as condições internas constantes necessárias para a vida, corrige desvios, elimina excessos, controla forças antagônicas.

d) Heterostase

Heterostase é um fenômeno contrário à homeostase. É o processo pelo qual um sistema pode sair de uma homeostase para outra homeostase bastante diferente.

e) Equifinalidade

Equifinalidade é o princípio ou teoria organizacional segundo a qual um sistema pode atingir um estado final igual com origem em

condições iniciais diferentes e por meio de diversas formas e meios de desenvolvimento. Isso porque não existe uma única maneira certa, mas sim várias alternativas dependendo do caso.

1.1.4 O pensamento sistêmico

O pensamento e o conhecimento humano têm enfrentado desafios e sido debatidos frente às rápidas mudanças técnicas e sociais que existem no ambiente no qual estão inseridos. Essas mudanças reivindicam uma nova visão de mundo que propõe superar a crise epistemológica e psicológica que se abate sobre a ciência, tecnologia, educação, cultura e sociedade. Essa crise é causada pelo excesso de racionalismo ocasionado pela fragmentação do conhecimento. Os avanços tecnológicos atuais causam uma grande desigualdade social em diversos países, principalmente os subdesenvolvidos. Nesse cenário, são necessárias novas maneiras de compreender e comunicar a realidade e enfrentar os problemas advindos dessas transformações.

O pensamento sistêmico é a disciplina mais importante para a organização que aprende, pois permite mudar os sistemas com maior eficácia e agir mais de acordo com os processos do mundo natural e econômico. É a criação de uma nova forma de abordagem da realidade, da análise e da solução de conflitos e da criação de soluções que os métodos mais lineares não conseguem. Possibilita uma visão sistêmica das pessoas, trabalho e organizações, pois considera as interações das partes com o todo. É a capacidade que uma pessoa adquire de avaliar os acontecimentos ao redor e suas possíveis implicações, a fim de criar uma solução única que possa contemplar as expectativas de todas as partes envolvidas. Surgiu no século XX, em contraposição ao pensamento reducionista-mecanicista ou cartesiano, que visava a fragmentação. Foi herdado dos filósofos da Revolução Científica do século XVII, que compreende o desenvolvimento humano sobre a perspectiva da complexidade. O pensamento sistêmico foi reformulado ao longo dos anos, tendo sua base epistemológica modificada.

O pensamento sistêmico não nega a racionalidade científica, mas acredita que ela não oferece parâmetros suficientes para o desenvolvimento humano e para a descrição do universo material. Por isso, deve ser desenvolvida em conjunto com a subjetividade das artes e das diversas tradições espirituais. Isto se deve à limitação do método científico e da análise quando aplicadas nos

estudos de física subatômica, biologia, medicina e ciências humanas. É visto como componente do paradigma emergente, que tem como representantes cientistas, pesquisadores, filósofos e intelectuais de vários campos. O pensamento sistêmico inclui a interdisciplinaridade e contrapõe o cartesianismo. Pensar de forma sistêmica exige uma nova maneira de olhar o mundo, o homem, e também exige uma mudança de postura por parte do cientista, que deve entender que o indivíduo não é o único responsável por ser portador de um sintoma. Um grande desafio para se manter o pensamento sistêmico nas empresas atuais é o alto índice de rotatividade de seus funcionários.

As disciplinas que fundamentam a base do pensamento sistêmico são:

a) **domínio pessoal** – quanto mais se expande a capacidade pessoal para obter os resultados desejados, maior a probabilidade de se criar um ambiente favorável ao engajamento das pessoas para o alcance de objetivos definidos.

b) **modelos mentais** – não há nada que não possa ser questionado, modificado, repensado ou reorganizado.

c) **visão compartilhada** – compartilhar uma visão e conquistar o engajamento de um grupo em relação ao futuro que se deseja criar é um desafio.

d) **aprendizado em equipe** – um dos maiores desafios de um líder é transformar aptidões coletivas e fazer com que as equipes desenvolvam capacidades maiores do que a soma dos talentos individuais.

1.1.5 Visão sistêmica

Com o crescimento contínuo da tecnologia, da concorrência e do público alvo cada vez mais exigente, faz-se necessário para a sobrevivência de toda organização a sua interação com meios externos e internos de atuação, na busca do sucesso. Muitas vezes, nos estudos ou em reuniões periódicas de organizações, ouve-se falar muito em análise global, em interação do todo e em visão sistêmica. Isso ocorre principalmente em processos de seleção de profissionais de nível superior e de executivos, no momento em que a visão sistêmica se torna um dos requisitos obrigatórios. Esses requisitos, também conhecidos como competências, devem ser atendidos para o exercício das

funções dos cargos que ocupam, para que o candidato avance na sua caminhada em busca da conquista daquela vaga para a qual está concorrendo. Ao falar em visão sistêmica, deve-se compreender que gerar informação é também a difundir. Os gestores precisam conhecer as variáveis que compõem a complexidade do negócio para uma melhor tomada de decisões. Não basta planejamento financeiro ou estratégico. As empresas como um todo, com pessoas e processos, precisam de diversas estratégias para atrair e fidelizar clientes e levar à sociedade padrões de alta qualidade.

A visão sistêmica consiste na habilidade ou capacidade de compreensão dos sistemas de acordo com a abordagem da teoria geral dos sistemas, ou seja, levar em consideração o conhecimento ou a visão do todo. Ela permite a análise global ou estudo das partes, bem como a identificação ou contexto da interação ou ligação entre estas partes ou a interferência no todo, o que faz com que várias forças, internas ou externas, atuem em um sistema em funcionamento. O objetivo é entender a influência das partes entre si e buscar excelência naquilo que diz respeito ao sistema. A visão sistêmica é formada a partir do conhecimento do conceito e das características dos sistemas. Está baseada no conceito de que o todo, resultante da junção das partes, é muito maior do que simplesmente a soma destas. Visão sistêmica é o olhar que permite enxergar de modo claro cada processo e cada negócio.

1.1.6 *Stakeholders*

O termo *stakeholder* foi criado por um filósofo chamado Robert Edward Freeman. A palavra vem do inglês *stake*, que significa interesse, participação, risco, enquanto *holder* significa aquele que possui. *Stakeholder* significa parte interessada ou interveniente ou público estratégico. É referente a qualquer pessoa, organização, grupo ou entidade que tenha legítimos interesses, que possa ser afetado, voluntária ou involuntariamente, ou que possua participação em um determinado projeto, processos ou desempenho de uma empresa, negócio ou indústria. Suas decisões e atuações podem afetar, direta ou indiretamente, essa mesma organização, onde há um objetivo específico de relacionamento. As pessoas e grupos mais importantes são designados para um planejamento estratégico ou plano de negócios, para trazer benefícios para ambas as partes. De maneira mais ampla, compreende o público de interesse e todos os envolvidos em um processo de uma organização. É uma palavra

em inglês muito utilizada em diversas áreas, como gestão de projetos, comunicação social, administração e tecnologia da informação. Os *stakeholders* são elementos essenciais ao planejamento estratégico de negócios. Ao entender sua importância, o responsável pelo planejamento consegue ter uma visão mais ampla de todos os envolvidos em um processo ou projeto e saber de que maneira eles podem contribuir para a otimização deste. Exemplos de *stakeholders*: patrocinadores, gestores, equipe, clientes, concorrentes, fornecedores, sindicatos, familiares dos membros da equipe.

Os *stakeholders* podem ser classificados em:

a) internos

São os *stakeholders* mais próximos da organização.

Exemplos: proprietários, trabalhadores, gestores.

b) externos

São os *stakeholders* externos à organização.

Exemplos: clientes, fornecedores, credores.

1.2 Organização

Organização é uma palavra originada do grego *organon*, que significa instrumento, utensílio, órgão ou aquilo com que se trabalha. No conceito tradicional, pode-se definir uma organização como o resultado do modo ou da forma em que se organiza um sistema para atingir os resultados pretendidos. É uma combinação de esforços individuais de duas ou mais pessoas ou elementos que realizam tarefas, em grupo ou individualmente, de forma conscientemente coordenada e controlada, com uma fronteira relativamente identificável. Esse conjunto atua em um determinado contexto ou ambiente e funciona numa base relativamente contínua, que tem por finalidade realizar propósitos coletivos e atingir um objetivo comum pré-determinado. Esse objetivo é alcançado com diversos meios e recursos disponíveis, liderados ou não por alguém com as funções da conhecida sigla usada em administração, POCCC, que representa as cinco funções de um administrador, descritas a seguir.

a) Planejar – estabelecer metas, objetivos e resultados futuros.

b) Organizar – definir como utilizar os recursos e estruturar a organização.

c) Controlar – acompanhar as atividades.

d) Coordenar – estabelecer prioridades e sequência de atividades.

e) Comandar – dirigir e liderar pessoas.

Uma organização é constituída pela soma de: pessoas, amparada pelas máquinas e outros equipamentos que facilitam o trabalho; recursos financeiros; entre outros. Por meio de uma organização, torna-se possível perseguir e alcançar objetivos que seriam inatingíveis para apenas uma pessoa.

São exemplos de organização: uma empresa, um clube, uma escola, um hospital, um evento. Em todos esses exemplos, o sentido de organização se baseia na forma como as pessoas se relacionam entre si na ordenação e distribuição dos diversos elementos envolvidos, com vista em uma mesma finalidade. São elementos diretamente associados a uma organização: clientes, fornecedores, concorrentes, comunicação social.

Convém reter alguns conceitos fundamentais para a adequada compreensão da definição de organização.

a) Recursos

Os recursos organizacionais representam todos os vários meios colocados à disposição das instituições ou organizações e necessários à realização das suas atividades ou tarefas, para que atinjam seus objetivos. São os bens ou serviços utilizados nas atividades organizacionais. Eles podem ter diferentes funções que representam o trabalho que fazem. Exemplos: as matérias-primas utilizadas nas produções, os serviços prestados pelas organizações, matérias, equipamentos, colaboradores. São divididos em: recursos físicos ou materiais, recursos financeiros, recursos humanos, recursos tecnológicos e recursos administrativos.

b) Objetivos

Os objetivos organizacionais representam as metas, resultados organizacionais, quantitativos e qualitativos; o fim desejado que a organização pretende atingir em prazo determinado, no contexto

de seu ambiente; o propósito que justifica toda a atividade desenvolvida ou até a própria existência da organização. Os objetivos são a razão de ser das organizações, para cumprir sua missão. Todas as organizações necessitam de um fim objetivo e devem determinar seus objetivos e definir as medidas e formas de atuação e de alocação de recursos.

c) Contexto

O contexto organizacional representa o que afeta as organizações, interna e externamente, o que pode influenciar significativamente em sua atuação e desempenho. Exemplos: contexto econômico, contexto tecnológico, contexto sociocultural.

A estrutura de uma organização pode ser:

a) **formal** – organização planejada e estruturada que segue um regulamento interno.

b) **informal** – relações geradas espontaneamente entre as pessoas.

Não é possível entender as organizações como sistemas fechados. Ao considerar a organização como um sistema global, o foco da visão sistêmica passa a centrar-se na capacidade que um profissional tem de ver a organização ou empresa. Esse conhecimento da organização independe do seu tamanho, sendo necessário analisar o ambiente. É preciso estudar o conjunto de forças que possam exercer alguma influência sobre o funcionamento do sistema, e conhecer e entender sua dinâmica. Ainda, analisar como funcionam e se integram as forças atuantes na organização, como funcionam seus processos de obtenção, transformação e entrega de seus serviços, produtos e informações ao mercado e aos seus clientes. Os processos internos e como eles se relacionam com o ambiente externo também devem ser analisados. É necessário entender como circulam as informações veiculadas, desde seus pontos de origem até seus destinos. Possuir visão sistêmica é importante para todo profissional de uma organização e vital para os gestores, para identificar sua posição na estrutura empresarial. É interessante também compreender sua função como parte integrante de uma equipe na realização de suas atividades dentro da área em que trabalha, gerando ferramentas organizadas e provedoras de resultados. O

emprego da visão sistêmica nas organizações permite abrir caminhos para as decisões a fim de resolver problemas que anteriormente não existiam.

Síntese

Um sistema é um conjunto de diferentes elementos relacionados entre si que operam em um ambiente para desempenhar uma função com um objetivo em comum, de modo a formar uma estrutura organizada. Os sistemas possuem cinco componentes: entrada, processamento, saída, controle e *feedback*. Eles podem ser classificados em: abstrato/conceitual/ideal ou concreto/físico/material/real; aberto, fechado ou isolado; dinâmico ou estático.

A teoria geral dos sistemas estuda a natureza dos sistemas e a relação entre seus elementos, para permitir uma comunicação mais eficiente e um melhor entendimento entre sistemas de áreas distintas. Já o pensamento sistêmico permite a mudança de um sistema de acordo com os processos do ambiente. Visão sistêmica é a capacidade de ver o sistema como um todo.

Stakeholder é o nome dado a todo e qualquer envolvido em um sistema. Os *stakeholders* podem ser classificados em internos ou externos.

Uma organização é um conjunto de elementos que trabalham juntos ou individualmente com a finalidade de produzir determinado resultado em determinado ambiente. Em outras palavras, uma organização pode ser considerada um sistema.

Atividades

1. Sobre a teoria geral dos sistemas, marque a alternativa falsa.

a) A TGS surgiu dos trabalhos de Ludwig von Bertalanffy, quando se percebeu a inviabilidade de tratar as ciências por partes isoladas.

b) A TGS tem como objetivo estudar a natureza dos sistemas e a relação entre suas partes em diferentes espaços.

c) A TGS teve grande aceitação por todos desde o seu surgimento.

2. Sobre o pensamento sistêmico, marque a alternativa falsa.

a) É uma forma de abordagem da análise de conflitos e da criação de soluções que pode auxiliar na mudança dos sistemas de acordo com os processos do ambiente.

b) É a capacidade de avaliar os acontecimentos ao redor e suas possíveis implicações, a fim de criar uma solução única que possa contemplar as expectativas de todas as partes envolvidas.

c) É contra a racionalidade científica, pois acredita que ela não oferece parâmetros suficientes para o desenvolvimento humano e para a descrição do universo material.

3. Sobre visão sistêmica, marque a alternativa falsa.

a) Consiste na capacidade de compreensão dos sistemas de acordo com a abordagem da teoria geral dos sistemas.

b) Seu objetivo é entender a influência das partes entre si e buscar excelência naquilo que diz respeito ao sistema.

c) Está baseada no conceito de que o todo, resultante da junção das partes, é equivalente à soma destas.

4. O que são *stakeholders?*

a) São as partes interessadas em um sistema, ou seja, as pessoas que podem ser afetadas ou possuem participação em um determinado sistema.

b) São as pessoas e grupos mais importantes de um planejamento estratégico ou plano de negócios.

c) São as pessoas responsáveis por um planejamento estratégico ou plano de negócios.

2

Sistemas de Informação

No capítulo anterior, vimos o que é um sistema. Neste capítulo, estudaremos o que é um sistema de informação. Para isso, é necessário, primeiramente, entender o que é informação: a resultante do processamento, da manipulação e da organização de um conjunto de dados de tal forma que possibilite uma tomada de decisão e que represente uma modificação, quantitativa ou qualitativa, no conhecimento do sistema que a recebe.

2.1 Pirâmide DICS

Um dos grandes desafios da atual fase da Era da Informação é converter dados em conhecimento. Tecnologias estão sendo desenvolvidas na área de Ciência da Informação para dar significado e importância à grande quantidade de dados que são gerados diariamente.

Uma maneira interessante de representar a relação entre essas entidades é por meio da pirâmide DICS (Dados, Informação, Conhecimento e Sabedoria). Por ser simples, é frequentemente usada em sistemas de gestão de informação, gestão de conhecimento e educação corporativa.

À medida que vai se alcançando o topo da pirâmide, a complexidade e o valor dos elementos aumentam. Para ter sabedoria, é necessário ter conhecimento. Mas para ter conhecimento é necessário ter informação. E para ter informação é necessário ter dados.

É possível aplicar a pirâmide DICS aos sistemas de informação para identificar em qual estágio da pirâmide se encontra um sistema e analisar se o objetivo que se desejava realmente foi alcançado.

As camadas da pirâmide são:

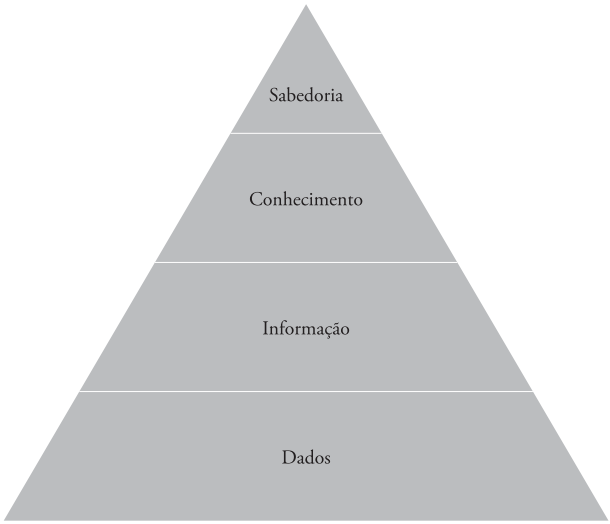
a) **dados** − são elementos discretos, estruturados, provenientes de coleta ou pesquisa. É o nível mais básico da pirâmide. Exemplos: números, símbolos, palavras.

b) **informação** − é a interpretação ou o processamento dos dados. Surge a partir da estruturação ou da organização de dados processados para um fim/contexto específico. Permite identificar "o quê". Exemplos: equações, significados, sentenças.

c) **conhecimento** − é composto por uma mescla de informação contextualizada, organizada ou aplicada, valores, experiências e regras. Permite identificar "como". Exemplos: teorias, soluções, livros.

d) **sabedoria** − é o estágio mais complexo de se definir e ocorre quando há a ressignificação dos outros níveis em combinações metalinguísticas. Permite identificar "por que". Exemplos: leis, princípios, paradigmas.

Figura 2.1 – Pirâmide DICS



Fonte: Elaborado pela autora (2016).

2.2 O que é um sistema de informação

Sistema de informação (SI) é a expressão derivada do conceito de sistema como atividade humana, utilizada para descrever um sistema cujo elemento principal é a informação. É uma entidade sociotécnica ou um conjunto organizado de elementos ou funções integradas voltadas para a transformação de dados em informação, seja automatizado (envolve a utilização de computadores), seja manual.

Esses elementos interagem entre si para processar a informação e divulgá-la de forma adequada em função dos objetivos ou dos processos de uma organização. Um SI pode ser constituído por pessoas, atividades, procedimentos, máquinas, métodos organizados e/ou recursos materiais em geral para coletar, armazenar, processar, disponibilizar e disseminar dados que representem informação relevante para o usuário, para o cliente e/ou para a organização. Possui objetivos específicos de tornar a informação acessível e útil para quem a deseje e possa utilizá-la.

Os SIs utilizam recursos de pessoas, hardware, software, dados e redes para executar atividades de entrada, processamento, saída, armazenamento e controle que convertam dados em informação. Todo SI que manipula dados e gera informação, usando ou não recursos de tecnologia em computadores, pode ser genericamente considerado como um SI que funciona, portanto, como suporte às ações e às decisões humanas e depende do contexto em que está inserido.

2.3 História e evolução

A história dos SIs está atrelada à evolução dos computadores. Antes da década de 1940 e da popularização dessas máquinas, as informações eram arquivadas em papéis e guardadas em pastas por um profissional denominado arquivista ou arquivador. Entre outras responsabilidades, sua função era organizar, registrar, catalogar e recuperar informações.

O fato de as informações serem gravadas em papéis, além de exigir grande esforço para manter os dados atualizados e de recuperá-los quando necessário, dificultava muito a realização de inventários, atividade que por esse motivo era executada raramente, porque demandava o trabalho de várias pessoas, já que os papéis não possibilitavam cruzamento e análise de dados, e também porque havia grande possibilidade de falhas humanas. Essa fase da história foi marcada pela simplicidade de dados, informações, métodos e técnicas, bem como pela limitação e pela ineficiência dos sistemas.

Após o surgimento dos computadores, a evolução histórica dos SIs pode ser dividida em cinco gerações. A primeira geração teve início na década de 1940, durante a Segunda Guerra Mundial, e foi caracterizada pelo surgimento dos computadores e dos sistemas operacionais. Os computadores eram máquinas enormes, que ocupavam áreas grandes, como salas e galpões, extremamente caros, utilizados cientificamente para fazer cálculos mais rápidos, e tinham vida útil muito curta.

Quando o primeiro computador eletrônico com fim comercial surgiu, a programação era em linguagem de máquina, e as informações eram salvas em papéis perfurados. Foi então que teve início a disputa entre o homem e a máquina, pois a cibernética começou a substituir a inteligência humana. Dos anos 1950 até a metade da década de 1960, a linguagem de programação dominante era o assembly, e os cálculos eram feitos em milionésimos de

segundo. A principal função dos sistemas de informação, que nada mais eram do que sistemas contábeis, era o processamento de dados. Houve então um aumento na comercialização dos computadores.

A segunda geração começou na década de 1960 e foi marcada pela tecnologia de integração de circuitos, que possibilitou processos simultâneos e computadores menores, chamados de minicomputadores, o que facilitou a comercialização para empresas, apesar do preço elevado. O enfoque dos sistemas dessa geração era gerencial.

Na terceira geração, caracterizada por novas evoluções da técnica de circuitos e pelo surgimento de linguagens orientadas, começaram a ser desenvolvidos os primeiros SIs gerenciais, cuja principal função eram relatórios administrativos. Havia grande demora na geração de algumas informações.

Nos anos 1970, na quarta geração, os computadores tiveram seu tamanho ainda mais reduzido, sendo chamados de microcomputadores. A procura por eles cresceu rapidamente devido ao custo mais baixo e à maior agilidade. Surgiram as linguagens de alto nível e foram possíveis a transmissão de dados e a troca de informações através de redes de computadores. A principal função dos SIs era o apoio à decisão.

A partir dos anos 1980, com o surgimento da inteligência artificial e da internet, teve início a quinta e atual geração, a era da interface gráfica. Surgiram os microcomputadores com alto desempenho e altíssima velocidade de processamento e SIs mais complexos, cujo principal objetivo era estratégico e a interatividade com o usuário final. A oferta de produtos de software aumentou significativamente, pois se percebeu que os SIs podiam influenciar rapidamente as tomadas de decisão e que os relatórios podiam ser gerados com maior rapidez e frequência.

Teve início a era do uso doméstico dos microcomputadores. No fim do século XX, a tecnologia da informação se transformou em uma ferramenta fundamental para qualquer organização, e a internet se tornou popular. Os SIs evoluíram para interfaces gráficas e utilização das últimas tecnologias disponíveis para desenvolvimento.

Atualmente, essa evolução continua com software voltado ao relacionamento com o cliente e os fornecedores, sendo que a principal função dos SIs visa ao comércio eletrônico. Com o surgimento de novas tecnologias em celulares, é possível ainda um fluxo de informação em tempo real.

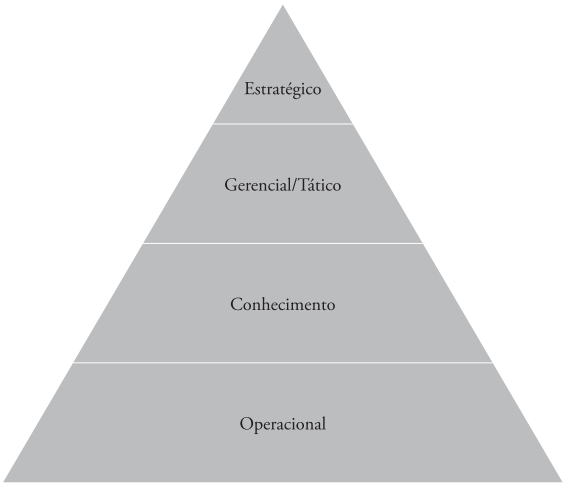
2.4 Classificação

Os SIs podem ser classificados quanto ao nível hierárquico, quanto à área funcional, quanto à abrangência e quanto à forma evolutiva, entre outros.

2.4.1 Nível hierárquico

Quanto ao nível hierárquico, os sistemas são classificados em apoio operacional, apoio ao conhecimento, apoio gerencial ou tático e apoio estratégico (figura 2.2).

Figura 2.2 – Níveis hierárquicos



Fonte: Elaborado pela autora (2016).

a) **Sistemas de apoio operacional:** auxiliam gerentes operacionais nas operações e nos processos. São formados por operações rotineiras e trabalham com grandes volumes de operações de entrada e saída. A principal meta dos sistemas desse nível é processar transações, controlar processos industriais, atualizar bancos de dados e apoiar comunicações. Exemplo: formulários de cadastro.

b) **Sistemas de apoio ao conhecimento:** auxiliam trabalhadores de dados e do conhecimento. Envolvem CAD (*computer aided design*, em inglês, ou desenho assistido por computador − DAC, em português), software office e gestão documental.

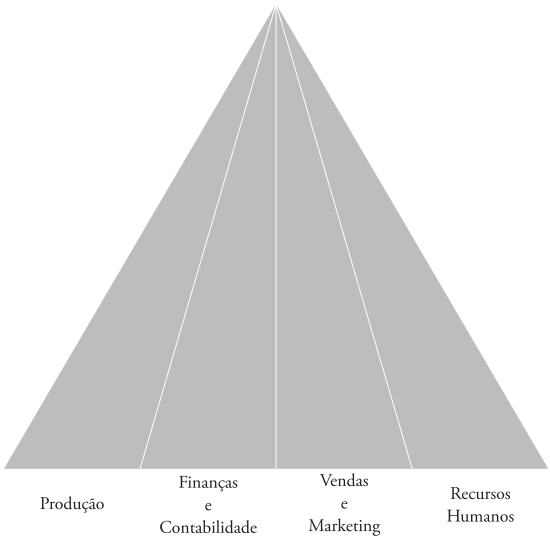
c) **Sistemas de apoio gerencial ou tático:** auxiliam gerentes de nível médio na tomada de decisão empresarial. Agrupam e sintetizam os dados das operações da organização e envolvem estatísticas de venda e de controle orçamental. São formados por operações de apoio na tomada de decisões e trabalham com informações agrupadas. Exemplo: relatórios.

d) **Sistemas de apoio estratégico:** auxiliam gerentes seniores nas estratégias para vantagem competitiva. Integram e sintetizam dados de fontes internas e externas à organização e envolvem produtos e tendências de custos. São formados por operações estratégicas. Tratam de tendências de longo prazo.

2.4.2 Área funcional

Quanto à área funcional, os sistemas são classificados em produção, finanças e contabilidade, vendas e marketing e recursos humanos (figura 2.3).

Figura 2.3 – Áreas funcionais



Fonte: Elaborado pela autora (2016).

a) **Produção:** fabricação, qualidade, manutenção.

b) Finanças e contabilidade: planejamento de recursos financeiros, captação de recursos financeiros, gestão dos recursos disponíveis, seguros, contábil.

c) Vendas e marketing: produtos, distribuição, promoção, preços.

d) Recursos humanos: planejamento, suprimento do quadro, gestão de recursos humanos, desenvolvimento de recursos humanos, pagamentos e recolhimentos, benefícios, obrigações sociais.

2.4.3 Abrangência

Quanto à abrangência, os sistemas são classificados em:

a) pessoais ou individuais − afetam um único usuário e têm como suporte físico geralmente um microcomputador. Auxiliam as atividades de um indivíduo da empresa cujo trabalho tem algumas características próprias e independentes do restante do funcionamento da organização. Têm um propósito muito específico e personalizado. Os sistemas pessoais são voltados à comunicação, à análise e à tomada de decisão e ao suporte ao registro e ao monitoramento de atividades. Fazem parte desse tipo os sistemas de coleta de dados que auxiliam os trabalhos de campo e os sistemas utilizados para melhorar a produtividade. Exemplo: ferramenta de edição de texto.

b) SIs de grupo ou departamento (*workgroup*) − afetam um grupo de usuários e estão ligados às atividades de um grupo de pessoas cujo trabalho tem aspectos comuns no que diz respeito ao cumprimento de um objetivo. Esses sistemas estão normalmente associados a um departamento da organização, servindo de suporte à coordenação das atividades e mantêm registro dos dados, fazendo acessos a informações de caráter global à organização. Podem ainda satisfazer solicitações diretas vindas de outros departamentos ou de uma hierarquia superior dentro da organização. Seu principal objetivo é facilitar o trabalho em grupo e as principais aplicações tratam de compartilhamento de hardware, da comunicação e das aplicações de controle de documentos e monitoramento de trabalho em grupo. Exemplo: servidor de e-mail.

c) **organizacionais ou corporativos** – afetam grande parte da organização e controlam o funcionamento global, coordenando as atividades de interação entre departamentos. Englobando os próprios sistemas locais dos departamentos, fornecem suporte a todas as divisões de uma organização, com o propósito de facilitar e controlar o fluxo de informações. Utilizam bancos de dados centralizados e compartilhados.

d) **interorganizacionais** – favorecem a comunicação entre duas organizações e têm como objetivo, principalmente, a troca eletrônica de dados, sistemas de acesso interorganizacionais, sistemas integrados interorganizacionais e redes de conhecimento. Referem-se à forma como as empresas em parceria geram as relações entre si e com os clientes e permitem a automatização de fluxos de informação entre organizações. Empresas que vendem produtos ou serviços semelhantes ou que precisam de ajuda de outras empresas para concluir a venda de um produto estão inegavelmente ligadas no mercado. Exemplo: extranet.

2.4.4 Forma evolutiva

Quanto à forma evolutiva, os sistemas são classificados em:

a) **manuais** – a maioria dos SIs começa de forma manual, para desenvolver o processo, e são exclusivamente manuais, como papel e lápis. Não são práticos e estão sujeitos a falhas, por isso costumam ser ineficientes.

b) **mecanizados** – são sistemas que utilizam os recursos da tecnologia da informação de forma mecânica, ou seja, sem valor agregado. São processos manuais transferidos para o computador sem acréscimo de funcionalidades. Sua maior vantagem é a agilidade em atividades repetitivas.

c) **informatizados** – são sistemas aprimorados em relação aos processos originais, utilizando os recursos da tecnologia da informação de forma inteligente e com valor agregado. Esse tipo de sistema integra basicamente três componentes: computadores (hardware),

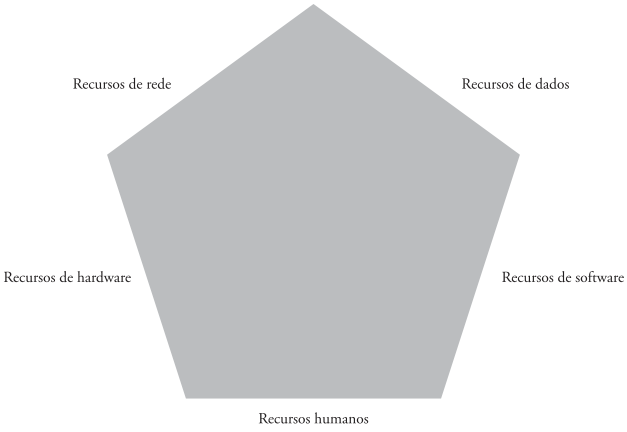
programas (software) e seres humanos (usuários). A vantagem de um SI informatizado é a facilidade de armazenamento e de recuperação de dados, a rapidez no processamento e, consequentemente, o fornecimento de forma rápida e efetiva das informações para a gerência. Mesmo assim, a informatização de um SI não garante a melhoria de sua efetividade, pois, se o sistema original já tiver defeitos, eles serão discriminados com a informatização.

d) automatizados – são sistemas que utilizam recursos mecânicos, pneumáticos, elétricos, eletrônicos e que trazem benefícios para a empresa, como agilidade, organização, praticidade, eficiência, entre outros. Exemplo: automação industrial.

2.5 Recursos

Um SI informatizado geralmente é composto por cinco recursos: humanos, hardware, software, dados e rede (figura 2.4).

Figura 2.4 – Recursos



Fonte: Elaborado pela autora (2016).

a) Recursos humanos: incluem os usuários finais (pessoas que utilizam ou administram um SI ou a informação que ele produz) e os especialistas em SI (profissionais que gerenciam, desenvolvem, dão suporte

e operam SIs). São componentes fundamentais, sendo o elemento mais importante em um SI baseado em computador, pois são quem faz tudo acontecer no tempo certo. Exemplos: gerente de projetos, analista de sistemas, analista de suporte, operador de sistema.

b) Recursos de hardware: consistem em todos os dispositivos físicos e equipamentos (máquinas e mídia) utilizados para executar as atividades de entrada (informar dados aos sistemas), processamento (transformar dados em informações úteis), saída (exibir informações resultantes do processamento) e armazenamento (reter os dados de entrada e/ou as informações processadas de forma permanente) de informações. Exemplos: computador, teclado, CPU (*central processing unit* ou unidade central de processamento), monitor de vídeo, HD (*hard disk* ou disco rígido).

c) Recursos de software: compreendem a parte lógica dos SIs com todos os conjuntos de programas e as instruções do processamento da informação (procedimentos) dadas ao computador e ao usuário. Permitem ao computador processar diversos aplicativos com rapidez, qualidade e baixo custo. Exemplos: sistema operacional, planilha eletrônica.

d) Recursos de dados: são meios físicos para armazenamento de dados e de software, uma coleção organizada de fatos e de informações. Os recursos de dados dos SIs normalmente são organizados em bancos de dados (coleção de registros e arquivos logicamente relacionados) e bases de conhecimento (que guardam conhecimento em uma multiplicidade de formas). Os recursos de dados são transformados por atividades de processamento de informação em uma diversidade de produtos de informação para os usuários finais. Devem ser efetivamente administrados para beneficiar todos os usuários finais de uma organização. O banco de dados é uma das partes mais valiosas de um SI baseado em computador. Exemplo: banco de dados Oracle.

e) Recursos de rede: redes de telecomunicações consistem em meios de transmissão e comunicação de dados entre computadores, processadores de comunicação e outros dispositivos interconectados

por mídia de comunicações em uma rede e controlados por software de comunicações. Os recursos de rede incluem mídia de comunicações e suporte de redes de comunicações (recursos de dados, pessoas, hardware e software que apoiam diretamente a operação e o uso de uma rede de comunicações). As redes de comunicações são um componente de recurso fundamental de todos os SIs. Exemplo: internet.

2.6 Ciclo de vida

O ciclo de vida de um SI informatizado é composto por várias etapas: especificação, análise, projeto, implementação, homologação, implantação e manutenção.

a) **Especificação:** é o levantamento de requisitos. Tem o intuito de identificar as necessidades do cliente e é responsabilidade principalmente do analista de requisitos.

b) **Análise:** é a documentação das funcionalidades. Tem o intuito de avaliar possíveis soluções e é responsabilidade principalmente do analista de sistemas.

c) **Projeto:** é a diagramação e o desenho da interface. Tem o intuito de avaliar tecnologias disponíveis e projetar as telas do sistema e é responsabilidade principalmente do arquiteto de software e do designer.

d) **Implementação:** é a codificação. Tem o intuito de desenvolver o sistema e é responsabilidade principalmente do programador.

e) **Homologação:** são os testes. Tem o intuito de testar as funcionalidades do sistema e é responsabilidade principalmente do analista de testes e do testador.

f) **Implantação:** é a instalação no ambiente do cliente. Tem o intuito de instalar o sistema para uso pelo cliente e é responsabilidade principalmente do programador, do analista de suporte e do administrador de banco de dados.

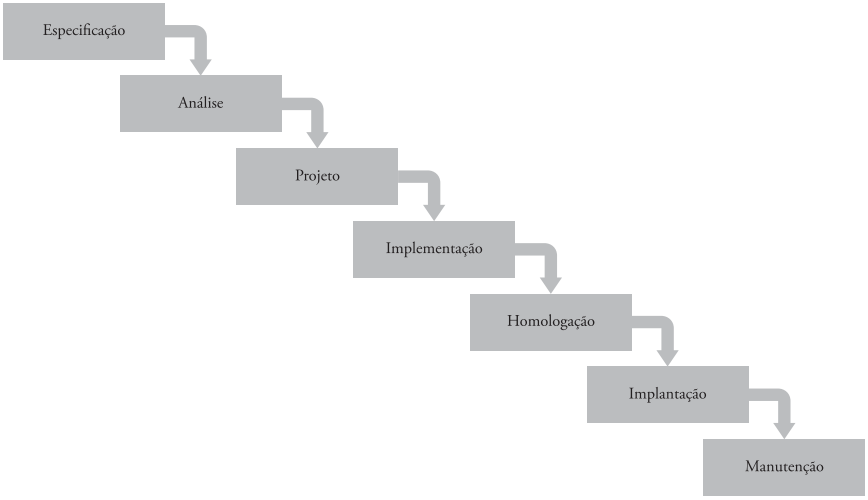
g) **Manutenção:** são atualizações de versão. Tem o intuito de corrigir eventuais erros e adicionar ou melhorar as funcionalidades do sistema e é responsabilidade principalmente do programador.

2.6.1 Processo de desenvolvimento

Existem vários processos de desenvolvimento de SIs. Os principais são: cascata, espiral, incremental e prototipagem.

a) **Modelo clássico ou em cascata:** o desenvolvimento do SI segue sempre a mesma sequência, na qual uma etapa é iniciada somente após a etapa anterior ter sido totalmente concluída, sendo que cada etapa é executada apenas uma vez. Assim, o produto de software é desenvolvido em apenas um ciclo e entregue somente após estar totalmente finalizado. As etapas desse modelo podem ser visualizadas na figura 2.5. Vantagens: redução do tempo de planejamento, facilidade de gerenciamento. Desvantagens: dificuldade de correção de erros, demora no prazo de funcionamento do sistema.

Figura 2.5 – Modelo clássico ou em cascata



Fonte: Elaborado pela autora (2016).

b) **Modelo em prototipagem ou prototipação:** o desenvolvimento do SI segue uma sequência não necessariamente na mesma ordem, sendo que cada etapa pode ser executada várias vezes. Assim, o produto de software é desenvolvido em vários ciclos e entregue somente após estar totalmente finalizado. Vantagens: visualização

do progresso, adiantamento do prazo de funcionamento do sistema. Desvantagens: falta de controle do tempo total do projeto, impossibilidade de cálculo da quantidade de ciclos.

c) **Modelo evolucionário (incremental e espiral):** o desenvolvimento do SI segue sempre a mesma sequência, na qual uma etapa é iniciada após a etapa anterior ter sido parcialmente concluída, sendo que cada etapa é executada várias vezes. Assim, o produto de software é desenvolvido em vários ciclos e entregue em versões, não estando totalmente finalizado a cada entrega. Vantagens: facilidade de execução dos testes, possibilidade de implementação de inclusões e alterações de requisitos. Desvantagens: dificuldade de integração entre as partes do sistema, dificuldade de negociação de prazos e custos.

2.7 Profissionais de informática

Há alguns anos havia o profissional de Tecnologia da Informação (TI), que era a pessoa responsável por fazer todo o trabalho visto. Com o passar do tempo, isso foi se modificando e gerando a setorização dos profissionais. Com isso, houve a especialização de grande parte dos trabalhadores nesse setor. Hoje, há pessoas trabalhando apenas com banco de dados ou apenas com programação, apenas com interfaces, enquanto outras testam as aplicações ou configuram e mantêm a infraestrutura.

Os principais cargos dos profissionais de informática são:

a) **administrador de banco de dados** – também chamado de DBA (*Data Base Administrator*), é o profissional responsável por criar, gerenciar e monitorar os bancos de dados corporativos nos ambientes de teste e homologação.

b) **administrador ou analista de redes** – é o profissional responsável por gerenciar e manter os equipamentos da rede local e remota, como instalar, configurar e manter recursos computacionais, ativos de rede e sistemas operacionais.

c) **analista de negócios** – é o profissional responsável por investigar os sistemas do negócio e seus processos e pelo alinhamento entre as

áreas de negócios e a área de TI, com o objetivo de propor melhorias e soluções.

d) **analista de requisitos** – é o profissional responsável por entrevistar os futuros usuários, para definir e especificar os requisitos do sistema de acordo com cada ação do usuário pela definição das necessidades deles.

e) **analista de segurança da informação** – é o profissional responsável pela análise, pelo projeto e pela manutenção do esquema de segurança da rede, incluindo a segurança de equipamentos, dados e SIs, para detectar vulnerabilidades.

f) **analista de sistemas** – é o profissional responsável pela sistematização de funções, que estuda processos computacionais com o objetivo de encontrar a melhor e mais racional forma de processar a informação, transformando problemas em soluções.

g) **analista de suporte** – é o profissional responsável pela seleção, pela implantação e pela manutenção da infraestrutura física de computadores (hardware) e de sistemas operacionais (software básico e de apoio), garantindo seu funcionamento.

h) **analista de testes** – é o profissional responsável por identificar e definir os testes necessários e monitorar a abrangência deles, para avaliar a qualidade geral obtida dos itens do teste-alvo conduzido em cada ciclo de teste.

i) **analista de usabilidade** – é o profissional responsável por aplicar os princípios de usabilidade para facilitar o entendimento do usuário e identificar problemas nas telas, resolvendo questões de experiência do usuário por meio de avaliação e testes.

j) **arquiteto de software** – é o profissional responsável por projetar uma solução compatível com os requisitos atuais da corporação empregadora, montando a melhor solução técnica para o projeto, para atender às expectativas do cliente.

k) **designer** – é o profissional responsável por elaborar projetos de design, sendo o designer digital o responsável por desenvolver

interfaces digitais interativas e atrativas, enquanto o web designer elabora o projeto estético e funcional dos websites.

l) **gerente de projetos** – é o profissional responsável por gerenciar e acompanhar a execução de projetos, levando em consideração escopo, metas, demandas, prazos, cronogramas e custos estabelecidos, atribuindo tarefas para a equipe.

m) **programador** – é o profissional responsável pela programação ou pela codificação e pelo desenvolvimento do sistema, escrevendo, montando e depurando o sistema desenvolvido pelos analistas e executando a manutenção dos sistemas já desenvolvidos.

n) **testador** – é o profissional responsável por testar os sistemas, avaliando a qualidade das aplicações dentro das normas estabelecidas, para garantir a qualidade e a eficiência do sistema que está sendo desenvolvido.

Síntese

O resultado de um conjunto de dados organizados é chamado de informação. Um sistema de informação é um sistema cujo elemento principal é a informação.

A evolução dos sistemas de informação caminha com a história dos computadores e é dividida em cinco gerações: a primeira, nos anos 1940 e 1950, caracterizada pelo surgimento dos computadores; a segunda, nos anos 1960, caracterizada pela tecnologia de integração de circuitos; a terceira, ainda nos anos 1960, caracterizada pelo surgimento de linguagens orientadas; a quarta, nos anos 1970, caracterizada pelo surgimento das linguagens de alto nível; e a quinta, a partir dos anos 1980, caracterizada pelo surgimento da inteligência artificial e da internet.

Os sistemas podem ser classificados quanto ao nível hierárquico (sistemas de apoio operacional, de apoio ao conhecimento, de apoio gerencial ou tático, de apoio estratégico), à área funcional (produção, finanças e contabilidade, vendas e marketing, recursos humanos), à abrangência (pessoais ou individuais, de um grupo ou departamento, organizacionais ou corpo-

rativos, interorganizacionais), à forma evolutiva (manuais, mecanizados, informatizados, automatizados).

Os sistemas de informação informatizados possuem cinco recursos básicos: recursos humanos, de hardware, de software, de dados e de rede. O ciclo de vida de um sistema de informação passa por várias etapas: especificação, análise, projeto, implementação, homologação, implantação e manutenção. Essas etapas são executadas conforme os tipos de processos de desenvolvimento: cascata, espiral, incremental e prototipagem.

Atividades

1. Quando surgiram os computadores?

a) Na década de 1940, durante a Primeira Guerra Mundial.

b) Antes da década de 1940, durante a Segunda Guerra Mundial.

c) Na década de 1940, durante a Segunda Guerra Mundial.

2. Qual é a diferença entre SI de grupo e SI corporativo?

a) Os SIs de grupo auxiliam as atividades de um grupo qualquer de indivíduos, enquanto os SIs corporativos auxiliam as atividades dos indivíduos de uma empresa.

b) Os SIs de grupo auxiliam as atividades de um departamento de uma empresa, enquanto os SIs corporativos auxiliam as atividades de grande parte de uma empresa.

c) Os SIs de grupo auxiliam as atividades de um grupo de indivíduos de uma empresa, enquanto os SIs corporativos auxiliam as atividades de indivíduos de empresas diferentes.

3. Qual das alternativas a seguir lista exemplos de recursos humanos, de hardware, software, dados e rede, respectivamente?

a) Banco de dados, internet, sistema operacional, gerente de projetos, computador.

b) HD, operador de sistema, planilha eletrônica, BD Oracle, World Wide Web.

c) Usuário, CPU, ferramenta de edição de texto, BD SQL, internet.

4. Qual processo de desenvolvimento de software é caracterizado por suas etapas serem executadas apenas uma vez, sendo que cada etapa nunca é iniciada antes do término da etapa anterior?

a) Cascata

b) Prototipação

c) Evolucionário

3

Levantamento de requisitos

A atividade de desenvolvimento de software abrange muitas fases e tarefas que, independentemente da metodologia selecionada, acontecem para atingir sua finalidade: entregar, dentro do orçamento e do prazo estimados, um produto funcionando corretamente

Há algum tempo, os profissionais de informática vêm notando a importância da relação entre a área de Tecnologia da Informação (TI) e as áreas de negócios. Nesse sentido, o levantamento de requisitos é muito importante e deve enfatizar o que é realmente necessário para o cliente e para os usuários, para que o software seja implementado da maneira correta.

O mais importante não é a beleza do aplicativo, mas sim sua utilidade para o usuário. Conhecer uma linguagem de programação não é suficiente para programar sistemas, pois desenvolver um software não é somente desenvolver programas, codificar e escrever propostas. O desenvolvimento de software deve ser baseado em processos que gerenciem o projeto de forma efetiva, assegurando assim a qualidade do produto final.

Este capítulo aborda a importância do levantamento de requisitos, visto que o sucesso dos projetos de desenvolvimento de software está diretamente ligado à etapa de especificação de requisitos.

O levantamento de requisitos é o começo de todo o processo de desenvolvimento de sistemas, sendo a primeira etapa e também a principal função ou tarefa técnica que faz parte da engenharia de requisitos. Essa fase é responsável por conceituar os serviços que um sistema deve realizar, sua interface com os demais elementos e sob quais restrições deve funcionar, evidenciando requisitos bem claros. É um processo iterativo que abrange todas as etapas, como detecção, reconhecimento, conceito, estudo, escopo e elicitação das necessidades de negócio que novos sistemas ou suas alterações devem fornecer para a solução do problema especificado.

Abrange também atividades de gestão de requisitos, que auxiliam a criação de um documento de requisitos e a manutenção, o versionamento, as mudanças e a qualidade dos requisitos elícitos no decorrer do tempo, sendo um dos artefatos mais importantes do processo de desenvolvimento de software. Tem como objetivo oferecer a melhor condição para atender e satisfazer as necessidades ou os requisitos e a expectativa de um cliente e inclui as regras e os processos do negócio. Oferece melhorias e eficácia do início ao fim, garantindo assim o correto funcionamento do sistema.

Um trabalho consistente e bem feito de levantamento de requisitos, com o entendimento completo dos requisitos de um sistema de informação, é essencial para as tarefas seguintes de um projeto de sucesso. Quando a especificação de requisitos é ineficaz, ou nem existe, muitos projetos não cumprem o prazo estimado ou são cancelados. Além disso, a má realização da fase de elicitação de requisitos pode levar o projeto a custar muito mais do que o necessário.

Portanto, é praticamente certo que o projeto terá seu sucesso comprometido se não houver uma correta definição e gestão de requisitos do apli-

cativo, o que pode frustrar as expectativas e comprometer os objetivos e os planos do cliente. Por isso, para a especificação de requisitos, é fundamental entender exatamente o que é um requisito.

3.1 Requisito

Um requisito nada mais é do que uma especificação de uma característica ou condição que deve ser alcançada pelo sistema; uma capacidade ou propriedade que o sistema deve possuir ou realizar; assim como a restrição de operação. Requisitos são uma coleção de sentenças que devem estabelecer todos os aspectos significativos que um componente deve possuir para satisfazer um contrato. Eles devem descrever de modo claro, conciso e consistente, sem ambiguidades, o que o sistema proposto deve ser capaz de realizar para atingir seus objetivos.

Normalmente, é durante a etapa de elicitação que os requisitos são identificados e definidos, em sua maior parte, para definir o problema a ser solucionado e dar uma visão geral do sistema a ser desenvolvido a partir de um domínio de negócio. Domínio do negócio, do problema ou da aplicação é a área específica para a qual o produto de software é desenvolvido.

Os requisitos devem ter informações suficientes para possibilitar que os responsáveis pela implementação desenvolvam um produto de software que atenda os contratantes.

Os requisitos são divididos em:

a) **funcionais** − definem e descrevem explicitamente o que faz uma função, funcionalidade ou serviço de um sistema de software, seu componente ou parte dele; são conjuntos de entradas, seus comportamentos e suas saídas. Podem ser cálculos, detalhes técnicos, lógicas de trabalho, manipulação e processamento de dados e outras funcionalidades específicas que indicam o que um sistema pode fazer, registram como ele deve reagir a entradas específicas, como deve se comportar em determinadas situações e o que não deve fazer. Em outras palavras, os requisitos funcionais ou fundamentais são aqueles que fazem parte do sistema, como um relatório específico ou um campo de cadastro. Geralmente têm o objetivo

de agregar valor ao usuário ou auxiliar no trabalho que este produz e são implementados no próprio sistema, sendo o sistema caracterizado pela implementação desses requisitos. Recebem suporte dos requisitos não funcionais, que impõem restrições sobre o projeto ou a execução dele.

Exemplos: o sistema deve permitir a inclusão, a alteração e a remoção de usuários com os atributos nome e senha; cada projeto deve ser associado a um identificador único; [RF001] o sistema deve cadastrar veículos particulares (entrada).

Os requisitos funcionais podem ser divididos em:

I. evidentes − efetuados com conhecimento do usuário final do sistema, que está ciente de que a função está sendo executada. Exemplos: fazer login no sistema, cadastrar e atualizar pacientes, imprimir um mapa.

II. escondidos/ocultos − quando uma função está sendo realizada, mas é invisível ao usuário. Exemplos: manter log de indicadores, prover integração com outro sistema, registrar log de acesso ao sistema.

b) não funcionais − são aqueles que definem e descrevem propriedades, restrições e objetivos do sistema; não o que o sistema deve fazer, mas como ele deve fazer. Ao contrário dos requisitos funcionais, esses requisitos, também chamados de atributos de qualidade, restrições e objetivos, envolvem especificamente a parte técnica e estão relacionados não com as funcionalidades oferecidas, mas com o uso e com o estado do sistema. Sua finalidade é, muitas vezes, criar e impor restrições de projeto aos requisitos funcionais de serviço do produto de software a ser implementado antes e durante o processo de desenvolvimento. Podem ser mais críticos que os funcionais, visto que não é necessário, para o cliente, explicitar os não funcionais, mas os não funcionais devem ficar implícitos para o programador, porque são características mínimas de qualidade de software, de aspectos internos do sistema todo ou de partes do sistema. Fica, portanto, a critério do programador escolher satisfazer

esses requisitos ou não, mas, quando não são atendidos, tornam o sistema inútil.

Exemplos: o sistema deve suportar uma carga mínima de 115 usuários simultâneos sem degradação de desempenho em qualquer operação; usuários devem operar o sistema sem treinamento; o sistema só permitirá acesso aos dados com autorização mediante informação de senha.

Incluem atributos de qualidades globais para o produto, por exemplo em termos de:

I. **compatibilidade** – o aplicativo deve oferecer compatibilidade e suporte às versões atuais dos sistemas operacionais iOS, Android e Windows Phone, por exemplo.

II. **confiabilidade** – não mais que dez registros a cada milhão podem ser perdidos devido a falhas de software, por exemplo.

III. **desempenho** – ao submeter a solicitação, o resultado deve aparecer em, no máximo, 5 segundos, por exemplo.

IV. **disponibilidade** – o sistema deve estar disponível pelo menos 99,5% do tempo em dias úteis no horário comercial e pelo menos 99,9% do tempo após as 18h e nos fins de semana, por exemplo.

V. **interoperabilidade** – a automação deverá ser capaz de enviar comandos ao sistema de venda, por exemplo.

VI. **manutenção** – versões novas do sistema devem ser disponibilizadas para atualização regularmente a cada seis meses e correções de defeitos devem ser implementadas e disponibilizadas em até cinco dias úteis após o centésimo registro formal de reclamação de usuários, por exemplo.

VII. **portabilidade** – o produto deve ser desenvolvido de forma a possibilitar seu transporte para a versão mais recente do sistema operacional a ser atualizada, por exemplo.

VIII. **segurança** – apenas usuários que possuam senha devem ter acesso ao dispositivo, por exemplo.

IX. usabilidade – um novo usuário deve ser capaz de instalar uma aplicação no sistema operacional após não mais que 30 minutos de orientação, por exemplo.

3.2 Etapas da especificação de requisitos

A especificação de requisitos é uma etapa demorada e trabalhosa. Uma elicitação de requisitos é adequada quando há uma boa definição do projeto. Nessa fase do processo de desenvolvimento, o analista de requisitos, juntamente ao analista de negócios e ao analista de sistemas, identifica e analisa o que o usuário deseja ou acha que necessita e foca em compreender o negócio que a aplicação vai automatizar.

É função do analista de requisitos perguntar cada detalhe do negócio para obter o máximo de conhecimento do usuário ou do cliente e entender suas reais necessidades. É preciso que os envolvidos no projeto de software saibam o que realmente se espera do sistema a ser desenvolvido. É muito importante também que todos os envolvidos saibam igualmente o que o aplicativo não deve fazer.

Apesar de parecer óbvio, nem sempre fica claro para todos os envolvidos no projeto qual é a fronteira do software, o que determina objetivamente quais requisitos devem e quais não devem ser automatizados.

A fase de levantamento de requisitos pode ser dividida em cinco etapas:

a) compreensão/entendimento do domínio do negócio/aplicação/problema – nessa etapa, o analista de requisitos deve compreender o domínio do problema no qual a organização e o projeto se inserem.

b) elicitação dos requisitos – nessa etapa, o analista de requisitos deve se comunicar com as partes interessadas, usuários e clientes, e então identificar quais são os requisitos pretendidos para o sistema junto aos *stakeholders* por meio de uma ou de mais técnicas de levantamento de requisitos.

c) análise dos requisitos – nessa etapa, o analista de requisitos deve classificar os requisitos em grupos ou módulos, para facilitar uma

visão global, e ordená-los conforme a prioridade, com qualquer eventual contradição, ambiguidade ou conflito já resolvido.

d) documentação dos requisitos − nessa etapa, o analista de requisitos deve gerar um documento de especificação de requisitos padrão.

e) validação dos requisitos − nessa etapa, o analista de requisitos deve fazer uma validação de todos os requisitos junto aos *stakeholders*, para verificar a completude, a consistência e a validade dos requisitos.

Mesmo quando há um sistema funcionando, não se deve focá-lo, mas sim um sistema novo. Ao mesmo tempo, não se deve subestimar o sistema que já existe, pois assim haveria grandes chances de não satisfazer as necessidades ou as expectativas do cliente, visto que grande parte delas não é mencionada no levantamento. Um sistema já implementado e que funciona pode mostrar tudo aquilo que já opera bem e que deve continuar e também tudo o que o cliente não possui e gostaria de ter ou o que ele já tem e gostaria de melhorar.

3.3 Seleção dos *stakeholders*

A escolha das melhores fontes de informação utilizadas para montar uma matriz de requisitos para definir o escopo do projeto sempre é o início de um bom levantamento de requisitos.

É essencial definir todos os envolvidos, considerando suas necessidades, além de garantir que eles entendam as implicações do produto a ser desenvolvido. Envolver o cliente desde o começo do processo de desenvolvimento dá uma maior segurança de que o novo sistema atenderá às necessidades identificadas.

Quem pode fornecer informações são os usuários dos sistemas já existentes e do sistema a ser desenvolvido, os responsáveis pelos departamentos nos quais o sistema deve ser utilizado, os técnicos que estejam familiarizados com as tecnologias envolvidas no novo sistema e nos sistemas já existentes, os responsáveis pela manutenção do sistema a ser implementado e, de modo geral, todos aqueles que podem ter qualquer tipo de interação com o novo sistema ou que sejam por ele afetados.

Em qualquer sistema, pequeno, médio ou grande, geralmente há diferentes tipos de usuário final que possuem algum interesse nos requisitos

do sistema. Por isso, mesmo para um sistema relativamente simples, existem muitas perspectivas diferentes que devem ser levadas em consideração. O método VORD (*Viewpoint Oriented Requirements Definition* ou Definição de Requisitos Orientada a Ponto de Vista) foi projetado como um *framework* orientado a serviço para o levantamento de requisitos. A habilidade essencial da análise orientada a pontos de vista é identificar a existência desses diversos pontos de vista e oferecer um framework para identificar conflitos nos requisitos propostos por diferentes *stakeholders*. As abordagens ou análise orientadas a ponto de vista identificam e reconhecem essas possíveis perspectivas e as usam para estruturar e organizar o processo de especificação e os próprios requisitos, segundo uma hierarquia.

No levantamento de requisitos, não se deve ignorar o especialista do domínio ou do negócio, que é o profissional que possui experiência no ramo ou no nicho de mercado que o sistema deve atender em suas funcionalidades. Por exemplo: no sistema de uma loja, o especialista do domínio pode ser o administrador que foi vendedor por muitos anos; em um sistema de lançamento de notas escolares, o especialista pode ser o professor com mais tempo na escola; em um sistema de internet banking, pode ser um correntista que por algum motivo precisa acessar sua conta corrente diariamente.

Por outro lado, há mais uma questão a ser observada. Há sistemas implementados sobre processos errôneos, que o analista de requisitos pode ter tomado por base enquanto entrevistava um funcionário que executava uma atividade de forma inadequada.

3.4 Técnicas de elicitação de requisitos

Na etapa de levantamento de requisitos, o analista costuma se reunir com os clientes e/ou usuários do sistema para identificar as funções do produto de software a ser desenvolvido. Não há nenhuma técnica hábil para trabalhar toda a elicitação de requisitos com satisfação. Porém algumas ferramentas e técnicas são capazes de solucionar eficientemente parte do problema, e os analistas podem fazer uso de diversas delas para levantar os requisitos dos clientes.

A finalidade dessas técnicas é superar as dificuldades associadas a essa etapa. Não há uma técnica padrão para o processo de levantamento de requi-

sitos. Todas as metodologias de especificação de requisitos têm uma definição própria e suas respectivas vantagens e desvantagens a serem levadas em conta, que podem ser usadas pelo analista juntamente a outras, e nenhuma delas é completa dadas as inúmeras variáveis de complexidade. Para atingir um levantamento de requisitos mais preciso, é essencial conhecer várias técnicas para saber qual delas utilizar em cada situação. Assim, dependendo das características do projeto, o uso de uma técnica isoladamente ou de mais técnicas em conjunto ajuda a melhorar a qualidade do levantamento de requisitos.

A seguir estão as principais técnicas de elicitação de requisitos.

a) **Análise de observação** – a técnica se resume em visitar o local em foco com a finalidade de observar os usuários em seu ambiente de trabalho enquanto executam suas atividades. Pode ser utilizada para ratificar os resultados de uma entrevista, obter informações conforme o dia a dia das transações e a realização dos processos diários do local e identificar a documentação que deve ser estudada. A presença do analista não deve interferir na execução das tarefas do usuário, mas é necessário que todos os processos observados sejam anotados.

b) ***Brainstorming*** – é uma metodologia para a geração de ideias que consiste em uma ou várias reuniões cujo objetivo é uma apresentação do problema/necessidade. Possibilita a sugestão e o debate das ideias de um grupo específico de pessoas para listar assim as melhores soluções. Sua principal finalidade é fazer o grupo expor seu conhecimento e sua criatividade, encorajando os participantes a juntar ou melhorar as ideias dos outros. É preciso que todas as ideias fiquem disponíveis para todos os participantes e que nenhuma ideia seja desprezada ou ignorada.

c) **Entrevista com *stakeholder*** – é talvez uma das formas de comunicação tradicionais mais simples entre, no mínimo, duas pessoas. É muito eficaz e bastante utilizada com a finalidade de coletar informações e gera bons resultados na etapa inicial de coleta de dados. É uma reunião do projeto requerido, em que se sugere entrevistar apenas uma pessoa por vez. O entrevistador deve dar espaço ao entrevistado para que apresente suas ideias e esclareça suas necessidades logo no começo. Um plano de entrevista pode ser preciso

para que não exista dispersão do assunto principal e para que a entrevista não demore, deixando o usuário exausto e, consequentemente, não gerando bons resultados. Após a entrevista, deve ser verificado se o que foi registrado pelo analista está em conformidade com a necessidade do entrevistado. Outra questão a se verificar é se o usuário não mudou de ideia e se compreendeu a notação ou representação gráfica das informações. Essas entrevistas devem levantar requisitos ainda não bem definidos conforme o escopo do projeto e requisitos que possam ser contraditórios; o custo inicial é um fator na decisão de quem será entrevistado.

d) Etnografia/Estudo etnográfico − a etnografia é uma técnica de observação e análise de componente social das tarefas desempenhadas em dada organização. Pode ser usada para produzir uma compreensão completa e detalhada dos requisitos sociais e organizacionais e também pode ser usada para compreender a política organizacional e a cultura de trabalho. Sua finalidade é se ambientar com o sistema e sua história e auxiliar na descoberta de requisitos de sistema implícitos que representem os processos reais. Nessa técnica, o analista entra no ambiente de trabalho em que o sistema deve ser usado − essa observação pode ser usada de forma combinada com registros de áudio ou vídeo.

e) Grupo focal − é um grupo de discussão informal de tamanho reduzido, com o objetivo de coletar informações qualitativas. As pessoas são convidadas para participar de um debate sobre determinado assunto.

f) Questionário − perguntas organizadas com o objetivo de levantar dados para uma pesquisa ou um estudo e gerar conhecimento sobre opiniões acerca das questões, cujas respostas são fornecidas pelo informante sem a orientação direta do pesquisador. Essa técnica pode ser a menos complexa, mas é muito eficaz em uma etapa inicial de coleta de dados e é interessante quando existe uma grande quantia de informantes para oferecer as mesmas informações. Recomenda-se utilizar essa técnica quando existem vários grupos de pessoas que podem estar em vários lugares distintos do país. Nesse caso, como não seria prático entrevistar todos os usuários em todos os lugares, as pesquisas específicas de acompanhamento

devem ser criadas com perguntas direcionadas por escrito às pessoas. São autoaplicáveis, pois o próprio participante as responde. Existem diversos tipos de questionários, entre eles múltipla escolha, lista de verificação e perguntas com espaços em branco. O questionário deve ser elaborado com perguntas simples, claras e concisas, com espaço suficiente para as respostas subjetivas e as perguntas de assuntos específicos agrupadas com um título especial. Para ressaltar a importância da pesquisa para a organização, uma carta explicativa deve acompanhar o questionário.

g) **Rápida prototipação/prototipagem/prototipificação** – é muito utilizada no estágio inicial do projeto, quando os *stakeholders* são incapazes de expressar seus requisitos ou quando não têm nenhuma experiência com o sistema. Consiste em uma prévia da interface da versão inicial do sistema, com base em requisitos ainda pouco definidos. Possibilita e auxilia os *stakeholders* a amadurecer uma forte visão ou noção de como deve ser a aparência do sistema que ainda não foi desenvolvido. Sua finalidade é analisar aspectos críticos dos requisitos de uma aplicação, desenvolvendo de forma rápida um pequeno subconjunto de funções desse produto. Lida com a validação dos requisitos e sua representação de forma compreensível a stakeholders distintos. Protótipos são recomendados para analisar as opções de telas do produto de software, problemas de integração com outros produtos e possibilidade de satisfação dos requisitos de desempenho. Os leitores podem apontar os reais requisitos e fluxos de trabalho do produto através da visualização antecipada das telas, levando a poucas mudanças posteriores e, consequentemente, reduzindo consideravelmente o custo total. *Wireframes* é o nome dado aos protótipos representados em diagramas. Nesse tipo de abordagem, não são desenhadas todas as funções.

h) **Revisão/Estudo de documentação/Análise de conteúdo** – constituem em geral uma abordagem mais simples e são uma das modalidades mais comuns de obtenção de dados sobre a situação atual do sistema: a leitura, o estudo e a reutilização de documentação de diferentes naturezas. Objetivam a identificação dos requisitos a serem implementados no sistema a ser desenvolvido. Há uma grande variedade de documentos e fontes de informação que pode

ser estudada e usada, por exemplo: estrutura organizacional da empresa, manuais de procedimentos, padrões de mercado, manuais de projeto, leis, diagramas, manuais de usuário, relatórios de pesquisas de mercado, glossário de termos de negócio, entre outros. Essa técnica é normalmente usada com outras técnicas de levantamento de requisitos.

i) **Sessão JAD/RAD** – é uma metodologia criada pela IBM para promover cooperação, entendimento e trabalho em grupo entre os usuários e os desenvolvedores. É baseada em sessões de dinâmica de grupo e workshops nos quais *stakeholders* e analistas de requisitos se reúnem para debater as funcionalidades desejadas do produto de software. Tem como finalidade envolver todos os *stakeholders* importantes no processo de levantamento através de reuniões estruturadas e com objetivo bem definido. O JAD inclui os objetivos do sistema e a criação da interface e de relatórios, definindo o ponto de vista dos usuários sobre o produto e ajudando na elaboração de uma visão compartilhada do que o sistema deve ser. Depende diretamente do nível de comprometimento dos *stakeholders*, bem como do líder das sessões JAD. Na sessão, há sete tipos de participantes: coordenador, moderador ou líder da sessão; secretário; analista de requisitos; patrocinador ou executor; representantes dos usuários; representantes de produtos de software; especialista.

j) **Workshop de requisitos** – trata-se de uma técnica de elicitação em grupo usada através de uma reunião estruturada. Devem participar do grupo uma equipe de analistas e os *stakeholders* que melhor representem a organização e o contexto em que o sistema deve ser utilizado, para assim identificar um conjunto de requisitos bem definidos. A interação entre todos os *stakeholders* no workshop deve ser encorajada por um moderador neutro que pode ser utilizado para manter o processo em foco. Sua função é conduzir o workshop e encorajar o debate entre os elementos presentes e sua postura deve ser de mediador e observador. A finalidade do workshop é promover o trabalho em equipe. Em alguns casos, o workshop de requisitos deve ser feito em ambientes controlados, para que os

participantes não fiquem dispersos. Após os workshops, devem ser produzidos documentos que representem os requisitos.

3.5 Documento de especificação de requisitos

Na fase de levantamento de requisitos, um documento de especificação de requisitos padronizado para obtenção de informações dos requisitos de software deve ser elaborado como resultado. Uma definição dos requisitos do usuário e uma especificação dos requisitos que o sistema deve possuir devem estar nesse documento, que aparece como um consenso entre a equipe de desenvolvimento e o cliente.

Esse documento de requisitos de software é a documentação oficial que orienta as tarefas seguintes, atribuídas aos desenvolvedores do sistema, e permanece como um parâmetro para validações. Torna-se um ponto de referência para medir o tempo gasto e os recursos necessários para realizar as mudanças solicitadas durante o desenvolvimento, visto que há requisitos que mudam durante o projeto.

Uma vez identificados e registrados os requisitos do cliente, o analista de sistemas e o arquiteto de software podem analisar e projetar a solução.

A seguir estão alguns exemplos de requisitos documentados.

Quadro 3.1 - Requisito de um sistema de venda

1.01	Identificação do Requisito: **001**
1.02	Domínio da Aplicação: Registro de logs Responsável pela informação: Maria dos Santos / Área: gerência / Data: 11.04.16
1.03	Qualificação Funcional: (2) (1) operacional (2) gerencial (3) estratégico
1.04	Área de Origem: (1) (1) interna (2) externa (3) ordem legal
1.05	Universo de Abrangência da Fonte de Informação: (t) (t) total (e) estimada
1.06	Quantidade Total: (1) Estimada (_) (1) 1-30 (2) 31-100 (3) >100

1.07	Descrição do Requisito:
	A Empresa quer melhorar o controle de informações de transações realizadas na loja
1.08	Problema Identificado:
	Não se faz o controle de transações realizadas na loja
1.09	Produto:
	Informações quantitativas e qualitativas das transações realizadas na loja
1.10	Aplicação:
	Controlar as informações das transações realizadas na loja
1.11	Atributos:
	Contador de transações mantido em log
1.12	Restrições:
	Limitações da tecnologia
1.13	Preferências:
	Fácil recuperação das informações
1.14	Expectativas:
	Ter informação em tempo adequado à necessidade de suporte técnico

Fonte: Elaborado pela autora (2016).

Quadro 3.2 - Requisito do sistema de uma biblioteca empresarial

RF-02	
Nome:	**Cadastrar Obra**
Descrição:	O sistema deve inserir uma nova obra no seu banco de dados.
Atores:	Bibliotecário.
Prioridade:	Essencial
Requisitos não funcionais associados:	RNF/USA-12 RNF/USA-14 RNF/DES-01

RF-02	
Nome:	**Cadastrar Obra**
Entradas e pré-condições:	• Ter efetuado o login no sistema • Entradas: Nome, Tipo, Descrição
Saídas e pós-condições:	Obra cadastrada no sistema com um cód. definido
Fluxos de eventos	
Fluxo principal:	O usuário deve informar nome, tipo e descrição da obra. Após essa etapa o usuário terá cadastrado uma nova obra no banco de dados do sistema.
Fluxo secundário:	O usuário pode esquecer alguma(s) dessas informações. Nesse caso, será mostrada uma mensagem informando qual(is) campo(s) ficou(aram) sem o seu devido preenchimento.

Fonte: Elaborado pela autora (2016).

Quadro 3.3 - Requisito de um smartphone

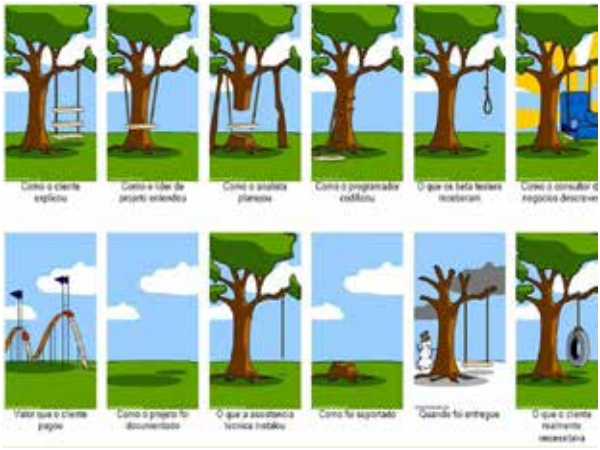
Nome	Quantidade de contas possíveis **(RN03)**
Descrição	Um aparelho/dispositivo não pode ter mais de seis contas de usuários cadastradas.
Fonte	Gerente do projeto
Histórico	Data de identificação: 11/04/2016

Fonte: Elaborado pela autora (2016).

3.6 Principais dificuldades e problemas

Existem algumas dificuldades típicas associadas à especificação incorreta do que o sistema deve fazer e que são muito antigas. As figuras a seguir são muito utilizadas como referência em várias fontes da literatura para comparar o processo de desenvolvimento de software ao processo de instalação de um balanço em uma árvore. Elas ilustram a dificuldade do processo de desenvolvimento de um produto, software ou não, a partir de uma solicitação do cliente, bem como a necessidade de um gerenciamento de projetos.

Figura 3.1 - Projeto balanço



Fonte: www.projectcartoon.com/CC BY 3.0

Cada figura do conjunto demonstra um processo ou uma fase do desenvolvimento de software e o conjunto consegue demonstrar o que realmente ocorre em um projeto. A análise do balanço é feita de forma completa no início do ciclo e diversos problemas acontecem no decorrer do projeto tanto para os desenvolvedores quanto para os clientes.

Especificação, análise e projeto de sistemas são processos que envolvem:

a) levantamento de informações sobre necessidades específicas do negócio da empresa;

b) estudo, organização e ilustração dessas necessidades;

c) elaboração da solução a ser utilizada no desenvolvimento do sistema.

Esse procedimento é semelhante ao processo de construção de qualquer produto, desde um livro até um imóvel. Porém, é na fase de especificação de requisitos que ocorre a maior parte dos erros, pois a falta de experiência dos clientes ou dos usuários faz que eles nem sempre tenham claras quais funcionalidades o produto de software deve ter. Além disso, ao falar em projeto novo, os desenvolvedores tendem a pensar em customizações do sistema, em

efeitos visuais ou em tecnologias novas que não são prioritárias ou agregam pouco ou nenhum valor para a aplicação ou o usuário final.

Falhas no entendimento de um sistema ocorrem devido a falhas em seus eventos. A comunicação entre os *stakeholders* deve ser a mais excelente possível, a fim de que o projeto seja homologado a contento pelo cliente. Os problemas que podem inibir a obtenção dos requisitos podem ocorrer devido a uma provável falta de *feedback*, com duas prováveis causas:

a) **falta de clareza do usuário** − o usuário principal do sistema é o responsável por mostrar ao analista, de maneira clara, a quais requisitos o sistema deve atender. Porém, ele pode não saber o que deseja que o sistema faça, ou até saber, mas não conseguir transmitir para o analista, apesar de estar convencido de que expressou claramente seus desejos, ou ainda mude de ideia sobre o que quer.

b) **não entendimento do analista** − o analista é o responsável por identificar e analisar os requisitos esperados pelo usuário. Ele deve documentar de forma clara seu trabalho, para que os consumidores saibam identificar esses requisitos. Porém, o pessoal técnico pode não entender o problema ou não perguntar exatamente o que o cliente quer, acreditando estar em perfeito acordo até que o produto final seja entregue.

Uma tentativa de solucionar os problemas de comunicação tem sido o emprego de analistas de negócio. As técnicas introduzidas em 1990, como prototipação, UML, Casos de Uso e desenvolvimento ágil de software, foram também introduzidas como soluções para os problemas apresentados.

Síntese

A primeira etapa do processo de desenvolvimento de software é o levantamento de requisitos. Essa fase é diretamente proporcional ao sucesso de um sistema. Requisitos são características que um sistema deve ter ou realizar, os quais podem ser divididos em funcionais e não funcionais. Um requisito funcional é uma função explícita de um sistema e pode ser evidente ou oculto. Já um requisito não funcional é uma propriedade ou restrição do produto de software. Entre os requisitos não funcionais estão compatibilidade, con-

fiabilidade, desempenho, disponibilidade, interoperabilidade, manutenção, portabilidade, segurança e usabilidade.

A etapa de especificação de requisitos pode ser dividida em compreensão/entendimento do domínio do negócio/aplicação/problema, elicitação dos requisitos, análise dos requisitos, documentação dos requisitos e validação dos requisitos.

A seleção dos *stakeholders* é muito importante, visto que há vários tipos de interessados no sistema a ser desenvolvido. Assim como existem os usuários experientes, cuja ajuda no levantamento de requisitos é válida, também existem os profissionais que executam os processos de forma inadequada e podem prejudicar a elicitação dos requisitos.

Entre as principais técnicas de elicitação de requisitos estão análise de observação, brainstorming, entrevista com *stakeholder*, etnografia, grupo focal, questionário, rápida prototipação, revisão de documentação, sessão JAD e workshop de requisitos.

O documento de especificação de requisitos deve conter a definição dos requisitos do sistema a ser desenvolvido. Entre as principais dificuldades e problemas dessa fase de especificação de requisitos, pode-se citar a falta de clareza do usuário e o não entendimento do analista.

Atividades

1. É um requisito evidente:

a) o telefone móvel deve utilizar a plataforma iOS.

b) efetuar login.

c) criptografar senha.

2. "Verificar acesso" é um requisito de:

a) confiabilidade.

b) disponibilidade.

c) segurança.

3. A técnica de elicitação de requisitos na qual o analista visita o local em foco com a finalidade de observar os usuários em seu ambiente de trabalho enquanto eles executam suas atividades é chamada de:

a) análise de observação.

b) estudo etnográfico.

c) sessão JAD.

4. A técnica de entrevista com *stakeholder* consiste em:

a) uma reunião do projeto requerido, em que se sugere entrevistar apenas uma pessoa por vez.

b) um grupo de discussão informal de tamanho reduzido, com o objetivo de coletar informações qualitativas.

c) sessões de dinâmica de grupo e workshops nos quais *stakeholders* e analistas de requisitos se reúnem para debater as funcionalidades desejadas do produto de software.

4

Análise de Sistemas

Análise de sistemas é uma tarefa na qual o problema é detectado, compreendido e modelado, e os requisitos e o modelo conceitual são detalhados. Assim, no trabalho de análise de sistemas é feita a divisão de um inteiro em seus componentes ou partes integrantes. Os sistemas de informação, com suas características e componentes, são mais bem entendidos quando divididos em partes menores, para conhecer melhor sua natureza, funcionalidades, relacionamentos, circunstâncias. Durante a análise, é realizado também um procedimento minucioso, além de pesquisas, que geralmente dão ênfase à detecção dos problemas complexos.

O princípio dos estudos da análise de sistemas abrange a compreensão ou o entendimento do que é um sistema na realidade dos sistemas computacionais, que é o objeto de estudo deste material. A finalidade deste capítulo é apresentar os conceitos básicos da análise e da modelagem de sistemas e a importância dessas práticas para os projetos de software.

Os analistas de sistemas observam os vários sistemas existentes entre hardware, software e o usuário final e criam novos sistemas, geralmente grandes e complexos, que funcionam em equipamentos utilizados por pessoas capacitadas e treinadas em processos operacionais padronizados e dotadas de conhecimentos para executar sua função. Os comportamentos e aplicações desses produtos são desenvolvidos a partir de soluções padronizadas e transcritas de maneira que o computador possa executá-las.

O analista procura entender o problema em amplitude, mas não necessariamente em profundidade, e definir a proposta de uma solução geral. Para isso, busca separar o problema em pequenas partes para facilitar o trabalho de análise, cuja meta é realizar pesquisas de processos, para então refinar, organizar e validar os requisitos, em termos de um modelo conceitual do problema. Também é finalidade da análise reconhecer e definir o que o sistema deve realizar, bem como detectar a melhor maneira racional para padronizar, minimizar a redundância, evitar a ambiguidade e reduzir a manutenção corretiva das especificações do sistema. Assim, os requisitos podem ser utilizados como base para o planejamento e acompanhamento refinados da implementação do sistema para que a informação possa ser processada.

A análise também tem como objetivo definir as funcionalidades de processamento de informações requisitadas por cada tarefa do produto, entrada, processamento, saída, armazenamento e controle. Ela visa refinar e registrar os processos de negócio para sua informatização e elaborar como resultado uma especificação completa de tudo que o sistema de informação deve fazer. Também modela o problema e se resume a todas as tarefas necessárias realizadas com ou para o conhecimento do cliente e para compreender o domínio do problema.

A análise trabalha em alto nível o modo como uma possível solução pode ser elaborada para satisfazer os requisitos, acabando por documentar uma especificação, mas sempre da perspectiva do usuário. Ela trabalha apenas com os objetos do domínio do problema, não com detalhes de programação. Um modelo de análise se conecta um pouco com a solução, pois as informações relevantes elicitadas nessa etapa são aquelas que determinam os processos de planejamento e que podem impactar na seleção das tecnologias. Elas podem e devem ser discutidas e aprovadas pelo cliente, especialmente no que se refere a alguns tipos de interações que devem acontecer na interface do usuário, entre outros.

As atribuições na análise de sistemas recaem sobre a análise propriamente dita e a administração de sistemas computacionais. É uma tarefa árdua, pois parte da estruturação, implantação e manutenção de aplicativos é de responsabilidade do analista. A qualidade do processo de análise é importante porque o custo de um erro de concepção que não é resolvido na fase de análise tende a crescer consideravelmente à medida que o desenvolvimento do sistema se aproxima do fim.

4.1 Métodos de análise

No decorrer do tempo, por causa das demandas, emergiram diversas propostas de métodos e técnicas como tentativa de solucionar os problemas. Os métodos de desenvolvimento de sistemas são fundamentais para a criação de um sistema que satisfaça completamente as necessidades e os requisitos definidos pelo cliente.

A metodologia que um analista utiliza para o desenvolvimento de um sistema pode ser compreendida como um caminho a ser percorrido em fases, e algumas delas podem ser trabalhadas simultaneamente. As atividades são realizadas por meio de técnicas, que são processos parametrizados e sistemáticos. Pelo fato de existirem diversos métodos para o desenvolvimento de sistemas e por ser uma tarefa de construção, executada por pessoas, sempre deve ser considerado o estudo de novas maneiras de deixar o método escolhido mais eficiente e eficaz.

A seleção do método apropriado para o trabalho dos desenvolvedores pode contribuir para o sucesso do projeto. Portanto, é necessário estudar quais são as principais metodologias de desenvolvimento de software.

4.1.1 Análise tradicional

Até 1965, os computadores de grande porte eram classificados como de segunda geração. Os sistemas existentes eram ferramentas simples e restritas e não tinham documentação. Não existia formação profissional para o desenvolvimento de sistemas.

A partir de 1965, já na terceira geração, houve aumento considerável na quantidade de usuários de sistemas. A documentação era basicamente um

documento, que representava a parte física da aplicação informatizada e que apenas o profissional que o elaborou conseguia entender. A formação profissional era precária. A abordagem da análise tradicional era quase somente funcional, orientada às funções da aplicação.

As principais dificuldades da análise tradicional eram o uso excessivo de termos técnicos na comunicação com o usuário, a falta de ferramentas de apoio e o fato de as mudanças normais requeridas pelo sistema serem maiores nas aplicações comerciais. E, conforme já mencionado, a documentação, quando existia, era manuscrita, sem padronização. Além disso, a manipulação dos processos era difícil.

4.1.2 Análise estruturada

A necessidade de compatibilidade com o projeto estruturado fez com que se criasse também a análise estruturada.

Uma união de elementos ou partes é chamada de estrutura. O método de análise estruturada consiste na utilização conjugada de técnicas e ferramentas para a criação de um modelo lógico para um sistema de informação gerencial. Cria-se uma nova e melhor especificação e um conceito de sistemas que utiliza técnicas gráficas para tornar mais fácil a comunicação entre o usuário, os analistas de sistemas e os projetistas. Essas representações gráficas são constituídas de símbolos que possibilitam elaborar modelos com o objetivo de achar um quadro ou solução clara, geral e única para o sistema. O conceito fundamental da análise estruturada tem como finalidade providenciar uma abordagem sistemática, etapa por etapa, para solucionar problemas. O método representa o fluxo e o conteúdo das informações usadas pelo sistema e particiona este em divisões funcionais ou ambientais e comportamentais. A essência do que deve ser desenvolvido é conectar os componentes do sistema para satisfazer às verdadeiras necessidades de quem dele necessita. As abordagens do método de análise estruturada são feitas por meio de processos e dados.

O método estruturado é um dos processos de análise de sistemas que mais chama a atenção dos profissionais de informática. Porém a análise estruturada deve ser utilizada somente para problemas pequenos e simples. Ela é de difícil utilização devido aos problemas de comunicação, às mudanças nos

requisitos do sistema e às técnicas de avaliação inapropriadas. Em sistemas maiores e mais complexos, recomenda-se que ela seja utilizada somente para proporcionar uma visão do sistema em alto nível. Entre as dificuldades da análise estruturada estão: comunicação, custo e documentação. Já entre os benefícios ou qualidades desse tipo de análise, destacam-se: modelos gráficos, divisão da especificação e interação com usuários.

A ênfase da análise estruturada é dada ao ponto de vista das funções, destacando os processos, sem modelar o comportamento temporal ou os relacionamentos complexos de dados. Os modelos da análise estruturada são o modelo essencial e o modelo de implementação do usuário.

a) Modelo essencial

É o modelo do sistema que especifica os requisitos verdadeiros ou essenciais.

I. Modelo ambiental: define o ambiente no qual o sistema se situa, descreve o contexto e as fronteiras do sistema, bem como as suas relações com o ambiente externo.

II. Modelo comportamental: define o comportamento interno do sistema, que descreve as ações que o sistema deve executar para reagir da melhor maneira possível aos eventos identificados no modelo ambiental.

b) Modelo de implementação do usuário

É o modelo que se baseia nos requisitos de implementação que são estabelecidos pelo usuário.

4.1.3 Análise essencial

Um dos métodos mais usados hoje é a análise essencial, também chamada de análise estruturada moderna. Ela pode ser considerada uma evolução de sucesso e também um refinamento da análise estruturada e seus métodos antecessores no desenvolvimento de sistemas.

Em 1984, Mc Menamim e Palmer apresentaram a nomenclatura *essential analysis* (análise essencial) para refletir a introdução dos novos conceitos que estavam sendo integrados à análise estruturada clássica. A análise essen-

cial é uma abordagem nova para especificar sistemas de informação utilizando levantamento e modelagem dos requisitos verdadeiros do sistema. Requisitos verdadeiros ou lógicos são características ou habilidades de que um sistema deve dispor para atender ao seu objetivo, independentemente da maneira como o sistema deve ser construído; são componentes do fluxo de informações necessários ao negócio da organização. Já requisito falso é aquele sem o qual o sistema consegue atender ao seu objetivo, ou seja, não é necessário para o objetivo do sistema. O conjunto completo de requisitos verdadeiros é chamado de essência do sistema ou requisitos essenciais, e a análise essencial é a metodologia que orienta a análise de sistemas à essência do negócio para o qual se destina. Assim, antes que um sistema seja construído, é preciso saber qual a sua verdadeira essência.

O passo inicial (muito importante, antes mesmo da aplicação da metodologia da análise essencial) é realizado pela compreensão e por uma definição precisa do domínio do negócio. É necessário entender o que o usuário está pedindo e o que se espera do produto de software a ser construído, em princípio dando atenção aos aspectos mais essenciais relativos à questão. Para iniciar a especificação do sistema, deve-se pensar nos eventos, acontecimentos ou mudanças no ambiente ou no mundo externo do sistema que afetam ou requerem uma resposta ou reação do sistema. É por meio dos eventos que se juntam os dados e as funções do sistema. O conjunto de ações executadas e resultados provenientes do sistema em reação ao acontecimento de um evento ou à realização de uma função denomina-se resposta, e a maneira como o evento age sobre o sistema é um estímulo, ativador de uma função. Com o conhecimento sobre o que se quer solucionar com o desenvolvimento do sistema, utiliza-se o método da análise essencial.

Após definir a abrangência do que deve ser realizado, outro passo importante é realizar grande, rigoroso, profundo e detalhado levantamento de dados englobando o conteúdo que se deve automatizar. O analista deve conhecer todos os eventos e dados essenciais referentes ao domínio do problema ao encerrar essa etapa.

A análise essencial propõe o princípio da divisão, que nada mais é do que a separação do sistema em um conjunto de eventos ou problemas menores para solucionar o problema central, pois assim é mais fácil de entendê-los.

Essa metodologia sugere que a especificação do sistema seja mostrada em três perspectivas, que se complementam: processos ou funções, dados e controle.

a) Modelagem de processos ou funcional: apresenta a perspectiva dos processos de transformação dos dados.
b) Modelagem de dados: procura especificar, a partir dos casos essenciais relacionados ao domínio de conhecimento estudado, a perspectiva que representa os dados que necessitam ser guardados para satisfazer a todas as necessidades de informações do sistema. Possibilita também organizar esses dados em estruturas bem definidas e delimitar as regras de dependência e restrições entre eles, construindo um modelo expresso por uma representação descritiva e diagramática.
c) Modelagem de controle: representa a perspectiva dos controles e tem uma função importante no caso de sistemas em tempo real.

Na análise essencial há dois níveis: essencial e de implementação, e cada um é representado por um modelo. A análise essencial é iniciada pelo modelo essencial.

4.1.3.1 Modelo essencial

O princípio básico do modelo essencial é descrever e mostrar o software em um grau ou nível de abstração de modo totalmente independente de limitações de tecnologia. O modelo essencial parte do princípio de que os sistemas existem independentemente dos equipamentos e são construídos objetivando a uma oportunidade de negócio. Ele possibilita uma solução ideal ao problema, sem depender das soluções de tecnologia da informação usadas em seu desenvolvimento. Além disso, não permite se influenciar por questões provenientes das restrições, que podem antecipadamente colocar alguma limitação à solução estudada.

Com isso, convém dizer que o modelo essencial é um modelo ideal, e que, na execução da análise essencial, a especificação de um sistema deve ser iniciada com a compreensão e a descrição dos requisitos verdadeiros que o usuário está pedindo. Também se deve definir quais requisitos o sistema deve satisfazer, cogitando a existência de uma tecnologia perfeita, sem se preocupar com a maneira como ele pode ou deve ser programado, bem como suas finalidades, e pela detecção dos eventos que impactam nele. Essa definição

deve ser compreendida como princípio da abstração, em que se cogita uma tecnologia ideal, sem restrições, que possibilita solucionar o problema por meio da divisão dos aspectos que estão relacionados a determinada realidade. O objetivo do princípio da abstração é representar esses aspectos de maneira simplificada e geral, em que não há falhas, defeitos ou erros de sistema. Para a tecnologia ideal, os custos, o consumo e o desgaste dos equipamentos são nulos, a capacidade de armazenamento de dados do sistema e a velocidade dos processadores são infinitas, e o tempo de acesso aos dados é instantâneo. Todas as atividades que fazem parte do objetivo declarado do sistema e que o sistema teria que realizar mesmo se fosse desenvolvido com tecnologia perfeita são chamadas de atividades essenciais ou fundamentais. É elaborado um modelo essencial do sistema a ser implementado, não contemplando as necessidades físicas, criado a partir dos esforços que se concentram na identificação das funções lógicas requisitadas para o software a ser produzido. O problema existente é estudado, mas não é modelado.

Não é relevante saber se a programação de um sistema deve ser manual ou automatizada, tampouco que tipo de hardware ou software deve ser utilizado antes de ele ser codificado.

Basicamente duas etapas ou modelos formam o modelo essencial.

a) **Modelo ambiental:** de uma perspectiva externa, determina e descreve as fronteiras, a interface e o relacionamento entre o sistema a ser desenvolvido e o resto do mundo exterior ou o meio ambiente no qual o sistema está situado. Apresenta a finalidade do sistema, o que é interno e o que é externo e que informações chegam ao sistema vindas do mundo exterior e vice-versa. Também representa os eventos do ambiente externo ao sistema aos quais este deve reagir e a interação do sistema com os elementos externos a ele.

b) **Modelo comportamental:** de uma perspectiva interna, determina e descreve o modelo de dados sobre o qual o sistema deve agir frente aos eventos e estímulos previstos do ambiente modelado no modelo ambiental. Apresenta o comportamento dos componentes internos de que o sistema deve dispor e todos os procedimentos que devem fazer parte do sistema. Também repre-

senta as ações que o sistema, como um conjunto de componentes inter-relacionados, deve realizar para se relacionar e interagir interna e apropriadamente.

4.1.3.2 Modelo de implementação

O modelo de implementação mostra o sistema em um grau ou nível de abstração totalmente dependente das limitações da tecnologia, sendo uma variação do modelo essencial que diz respeito à implementação do sistema.

O modelo de implementação tem como objetivo elaborar um esquema para determinar a maneira de implementação do sistema em um ambiente técnico dado a partir de suas especificações conceituais e dos requisitos para ele determinados e a interface com o usuário, bem como envolve assuntos relacionados ao uso do sistema pelo usuário.

4.1.4 Análise orientada a objetos

A orientação a objetos (OO) é uma forma de analisar os problemas usando modelos com base em conceitos do mundo real. Teve início no final dos anos 1960 e se estabilizou no final dos anos 1980, como uma tentativa de reduzir e resolver com baixo custo os problemas encontrados e até então persistentes no desenvolvimento e na manutenção de sistemas complexos. É um paradigma de análise, projeto e implementação de produtos de software com base na composição e interação entre elementos de software denominados objetos.

A análise orientada a objetos (OOA, *object oriented analysis*) é um processo de desenvolvimento de sistemas de software que estuda os conceitos do negócio usando objetos do mundo real. É um método de análise que usa a noção de objetos que se relacionam uns com os outros e, por meio dessa interação, executam tarefas computacionais.

A OOA é baseada em conceitos que o ser humano aprende na infância, como objetos e suas características, com a finalidade de definir as classes que são importantes para o problema a ser solucionado. Um sistema é uma solução sistêmica para um processo de negócio formado por objetos que se relacionam, com características próprias, representadas por atributos e operações.

O elemento principal da OO é o objeto, pois o mundo real é constituído de objetos que interagem entre si. O ser humano aprende, desde criança, a pensar de maneira orientada a objetos. Um objeto é identificado por um substantivo, por exemplo: caixa, balão, guarda-roupa, bolo, osso, flor, criança. É o conceito de uma entidade real ou abstrata ou de um elemento do mundo real que representa um conceito existente na realidade humana. As pessoas aprendem a classificar os objetos, constituindo conjuntos de objetos com características e comportamentos similares. Um objeto representa uma entidade de natureza física, conceitual ou de software:

a) entidade física – exemplo: carro, cliente, pessoa;

b) entidade conceitual – exemplo, diagrama, modelo, teoria;

c) entidade de software – exemplo, barra de componente, combobox, item de menu.

Os objetos são formados por uma estrutura, com componentes ou atributos que os caracterizam, e comportamentos, que são as ações que atuam sobre os objetos. Os componentes de um objeto estão descritos a seguir.

a) Identidade

É o nome do objeto. Cada objeto tem um nome único que não pode ser usado por nenhum outro objeto.

Exemplos: personagem Joaquim, João, Ana; cliente Mônica, Ana, Silva; professor Mário, Mônica, Araújo.

b) Atributo

São os valores dos dados de propriedades, características ou peculiaridades que caracterizam os objetos. Um atributo pode sofrer alterações a partir do comportamento do objeto ou ao longo do tempo e costuma variar de objeto para objeto.

Exemplos: uma Pessoa tem altura, cor da pele, quantidade de filhos; um Carro tem cor, placa, defeito mecânico; uma Venda tem número, data, valor total.

c) Método

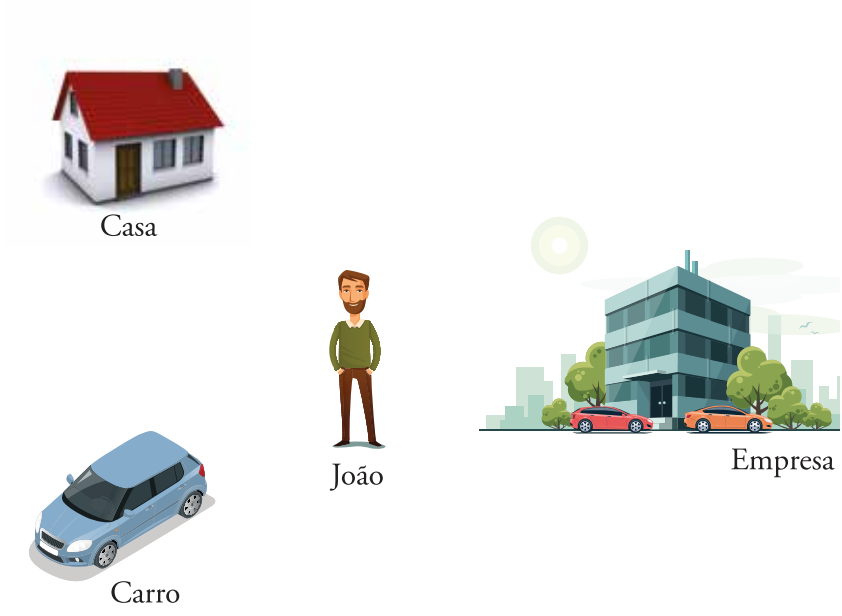
Também denominado comportamentos e operações, representa a forma como um conjunto de comportamentos pré-definidos são

executados por uma classe. Por meio dele o objeto reage a estímulos. Os métodos podem receber parâmetros e retornar valores.

Exemplos: um Carro pode transportar Pessoa, acelerar, frear; um Cliente pode consultar Produto, fornecer Endereço, incluir Pedido; um Usuário pode digitar, imprimir, fazer Login.

O desenvolvimento de um sistema de software OO é estruturado por um conjunto de objetos do domínio do problema. A OO auxilia no entendimento do mundo real e facilita a atividade de programação em computador, pois, num sistema de software, cada objeto executa atividades específicas, e as tarefas computacionais são executadas pela interação entre esses objetos. Dessa forma, as alterações se resumem a apenas modificações nos objetos, sem precisar modificar funções e dados. Por exemplo, João, sua casa, seu carro e a empresa onde trabalha fazem parte do mundo real.

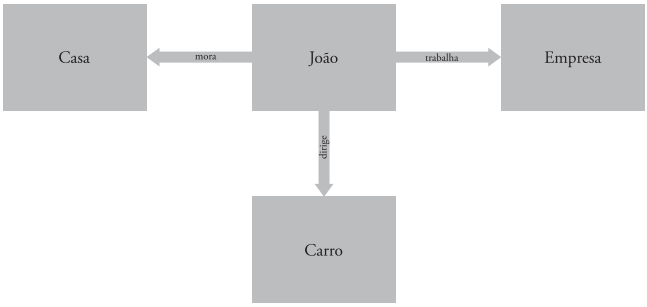
Figura 4.1 – Mundo real



Fonte: Elaborado pela autora (2016).

Esses objetos do mundo real são representados na modelagem.

Figura 4.2 - Modelagem



Fonte: Elaborado pela autora (2016).

Três conceitos são relevantes para os objetos.

a) Encapsulamento: é o armazenamento dos atributos e métodos de um objeto nesse mesmo objeto.

b) Visibilidade: é o nível de acessibilidade de um atributo ou método. Existem três tipos de visibilidade:

   I. pública – os objetos de quaisquer classes conseguem acessar o atributo ou método. É indicada pelo sinal (+);

   II. privada – apenas a classe possuidora do método ou do atributo consegue acessá-lo. É indicada pelo sinal (–);

   III. protegida – apenas a classe e as subclasses possuidoras do método ou do atributo conseguem acessá-lo. É indicada pelo sinal (#).

c) Mensagem: é a comunicação entre os objetos que acontece pela execução das operações.

Um objeto é também denominado instância ou ocorrência de uma classe, que é o projeto ou representação de um conjunto ou categoria de objetos semelhantes. Objetos da mesma classe são do mesmo tipo, visto que seguem a mesma especificação e têm atributos e estados similares e comportamentos e relacionamentos parecidos com os de outros objetos. Por exemplo,

uma classe Cliente contém os objetos João, Maria, Pedro, Renan e Thiago. A estrutura de um software é constituída de classes.

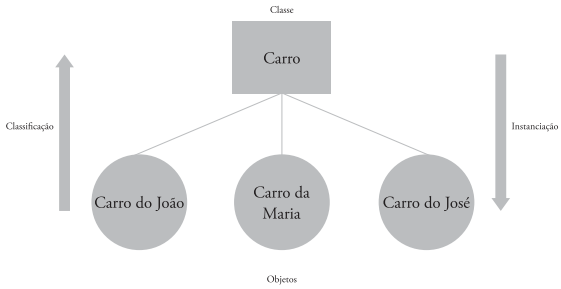
A modelagem conceitual das classes e objetos envolve dois conceitos importantes: abstração e representação.

a) Abstração

Tem a finalidade de observar a realidade e capturar a estrutura do negócio e consiste na seleção de alguns aspectos de domínio do problema a ser modelado para abstrair objetos. As operações de abstração estão descritas a seguir.

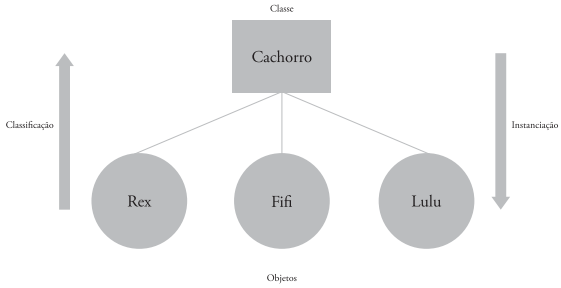
I. Classificação e instanciação: é a categorização dos objetos em grupos e/ou classes, com base em propriedades comuns. Exemplos a seguir.

Figura 4.3 – Exemplo 1 de Classificação e Instanciação



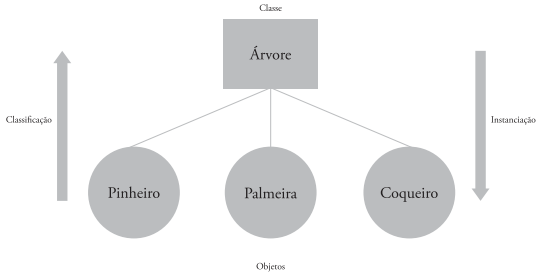
Fonte: Elaborado pela autora (2016).

Figura 4.4 – Exemplo 2 de Classificação e Instanciação



Fonte: Elaborado pela autora (2016).

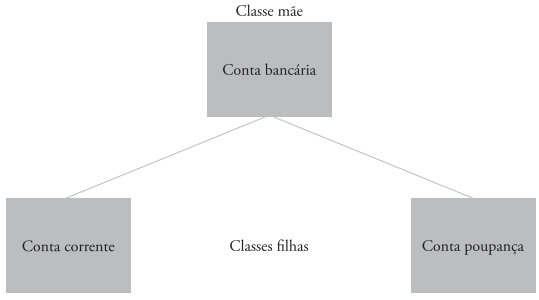
Figura 4.5 – Exemplo 3 de Classificação e Instanciação



Fonte: Elaborado pela autora (2016).

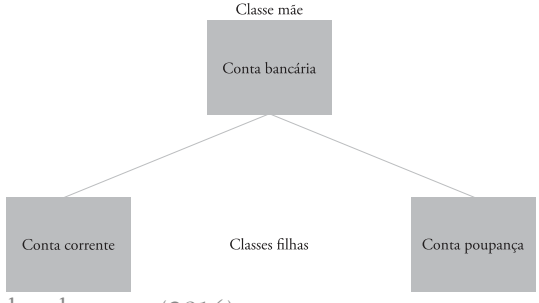
II. Herança: é a divisão de uma categoria genérica em mais categorias especializadas que satisfazem todas as propriedades da categoria mãe. Exemplos a seguir.

Figura 4.6 – Exemplo 1 de Herança



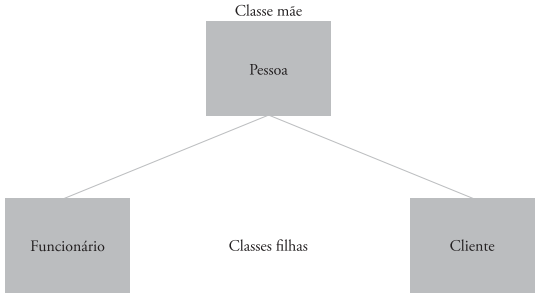
Fonte: Elaborado pela autora (2016).

Figura 4.7 – Exemplo 2 de Herança



Fonte: Elaborado pela autora (2016).

Figura 4.8 – Exemplo 3 de Herança



Fonte: Elaborado pela autora (2016).

b) Representação

São convenções de representação das classes por meio de retângulos com três divisões: nome da classe, atributos pertencentes à classe e métodos da classe. Exemplos a seguir.

Quadro 4.1 – Exemplo 1 de Representação

Cliente
+Codigo +Nome +CPF -Endereço -Cidade -Telefone -QtdPontos
+devolveNome() +fala() +imprime()

Fonte: Elaborado pela autora (2016).

Quadro 4.2 – Exemplo 2 de Representação

```
Cachorro
-nomeCachorro: string
-sexo: string
-raça: string
-cor: string
-dataDeNascimento: date
-idade: int
+setNome()
```

Fonte: Elaborado pela autora (2016).

Quadro 4.3 – Exemplo 3 de Representação

```
Pessoa
-cpf : string
-rg : string
-nome : string
-data_nascimento : datetime
-sexo : boolean
-endereço : string
+falar()
+escrever()
+andar()
+correr()
+parar()
+sentar()
+comer()
+dormir()
acordar()
```

Fonte: Elaborado pela autora (2016).

Em resumo, a OOA consiste na identificação de objetos e classes de um domínio de negócio, para a criação de um modelo descritivo contendo informações do projeto pela descrição de casos de uso – baseada na identificação dos cenários de como atores interagem com o produto a ser construído –, que serão estudados no próximo capítulo. As classes e os objetos, bem como os atributos e operações para cada objeto do sistema, além das estruturas e hierarquias das classes, precisam ser identificados e especificados. Por fim, os relacionamentos entre os objetos devem ser apresentados com a construção de um modelo objeto-relacionamento.

A OOA é utilizada em projetos grandes, complexos e críticos, com requisitos indefinidos, vagos, incompletos ou inconsistentes, além de sistemas novos. Também é recomendada para equipes com especialidades diversas.

Síntese

Análise de sistemas é a tarefa na qual o problema é detectado, compreendido e modelado, e os requisitos e o modelo conceitual são detalhados. Também é feita a divisão do domínio em partes, para melhor entendimento do problema.

Quatro métodos de análise de sistemas marcaram a história dos sistemas de informação computacionais. A análise tradicional não tinha padrão, e sua documentação, quando existia, era apenas um documento básico com informações do projeto. Já a análise estruturada, ainda hoje utilizada, tem foco nas funções do sistema, dando ênfase aos processos do negócio. Sua abordagem é sistemática, com a finalidade de resolver os problemas etapa por etapa.

A análise essencial, também muito utilizada hoje, tem foco na essência do sistema, que é o conjunto dos requisitos verdadeiros, independentemente da tecnologia utilizada na implementação, bem como seus limites e restrições. E a análise orientada a objetos, como o próprio nome diz, dá ênfase aos objetos do domínio, que são representados por substantivos do mundo real, têm atributos (características) e métodos (comportamentos) e são relacionados entre si. Geralmente são classificados em categorias, chamadas classes.

Atividades

1. Quais os métodos de análise mais utilizados hoje?

a) Análise tradicional, análise estruturada e análise essencial

b) Análise estruturada, análise essencial e análise orientada a objetos

c) Análise tradicional, análise estruturada e análise orientada a objetos

2. Qual o principal elemento da análise essencial?

a) A abordagem baseada em processos e dados

b) O conjunto de requisitos verdadeiros

c) A abstração de conceitos utilizados no mundo real

3. Marque I para identidade, A para atributo e M para método.

( ) Pessoa	( ) Funcionário	( ) Saldo	( ) Cor
( ) Nome	( ) Trabalhar	( ) Carro	( ) Excluir
( ) CPF	( ) Cachorro	( ) Comer	( ) Cliente

4. "Pastor alemão, pequinês e buldogue são cães" é um exemplo de:

a) instanciação

b) herança

c) visibilidade

5

Ferramentas de apoio à Análise de Sistemas

A maior parte do trabalho do Analista de Sistemas abrange a modelagem do sistema que o usuário deseja. Os modelos de Análise de Sistemas são representações abstratas do que se torna uma junção de hardware e software. Eles focam aquilo que o sistema deve realizar, o que possibilita uma comunicação mais direta com o usuário, pois assim é possível ler e entender um modelo, ainda que não haja conhecimento para elaborar um.

Ferramentas simples, porém muito eficazes, foram produzidas nas últimas décadas para ajudar na análise de dados. Nem sempre é preciso usar mais do que uma ferramenta de modelação, mas, quando combinadas, geram bons resultados.

As ferramentas básicas da atividade de descrição da análise de dados de um sistema são o diagrama de contexto, o diagrama de fluxo de dados, o diagrama de entidade e relacionamento e o dicionário de dados.

5.1 Diagrama de contexto

O diagrama de contexto (DC) é um diagrama de fluxo de dados (DFD) de alto nível que representa todo o sistema como um único processo. É constituído de fluxos de dados que apresentam as interfaces entre o sistema de negócios e sua relação com as entidades externas do ambiente. Seu desenho deve ser facilmente compreendido tanto pelo usuário quanto pelos profissionais de Tecnologia da Informação (TI). O DC delimita, até certo ponto, quais tarefas são executadas pelo sistema e quais não são.

5.1.1 Componentes

Os elementos do DC são:

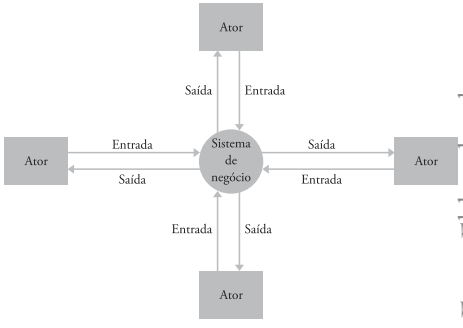
a) sistema de negócio − também conhecido como processo de alto nível, é representado por um círculo. Deve haver apenas um processo no diagrama, que representa e recebe o nome do próprio sistema. Exemplos: sistema de negócio; sistema PDV; o sistema.

b) atores − também chamados de entidades externas, são representados por retângulos e fornecem a entrada e recebem a saída. Podem existir diversas entidades externas que interagem com o sistema, mas elas não devem interagir entre si. Exemplos: alunos; professores; clientes.

c) entradas e saídas − são representadas por setas retas ou curvas que indicam o sentido do fluxo dos dados. Exemplos: pedido_reserva; dados-empréstimo; Cupom_Fiscal.

A figura 5.1 mostra relações possíveis entre esses elementos em um DC genérico.

Figura 5.1 - Modelo de diagrama de contexto

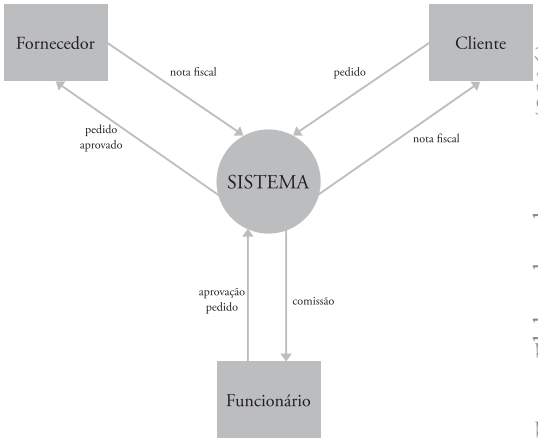


Fonte: Elaborado pela autora (2016).

5.1.2 Exemplos

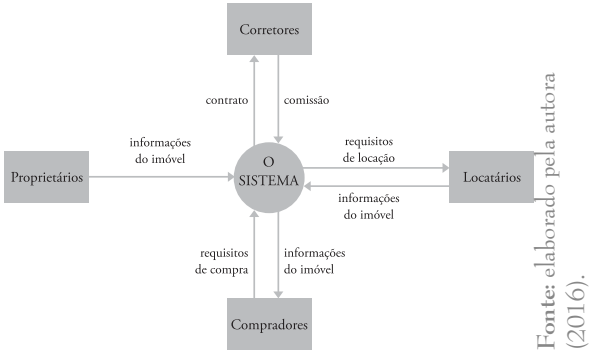
As figuras a seguir representam alguns exemplos de DC, sendo a Figura 5.2 a representação genérica do DC do sistema de uma empresa e a Figura 5.3 o DC do sistema de uma imobiliária.

Figura 5.2 - Diagrama de contexto de empresa genérica



Fonte: Elaborado pela autora (2016).

Figura 5.3 - Diagrama de contexto de uma imobiliária



Fonte: elaborado pela autora (2016).

5.2 Diagrama de fluxo de dados

Também chamado de diagrama de bolhas, modelo de processo, diagrama funcional, modelo funcional e diagrama de fluxo de trabalho, o diagrama de

fluxo de dados (DFD) é a principal técnica de modelagem funcional e a ferramenta de demonstração central mais utilizada por Analistas de Sistemas para documentar a fase de análise estruturada do ciclo de desenvolvimento de novos sistemas de informação.

O DFD é um diagrama dos processos do sistema e dos dados que ligam esses processos, descrevendo o fluxo de informação e a estrutura do sistema a ser desenvolvido. Esses diagramas facilitam o entendimento do que acontece com os dados durante a execução do sistema, fornecendo uma visão estruturada das funções, chamada de fluxo de dados.

Eles representam o que o sistema faz, que tipos de informação devem entrar e sair do sistema, de onde os dados devem vir, para onde devem ir e onde devem ser armazenados. O DFD também pode descrever processos computadorizados e não computadorizados.

5.2.1 Componentes

Os elementos do DFD são:

a) processos − também conhecidos como bolha, função ou transformação, representam transformações de fluxo(s) de dados de entrada em fluxo(s) de dados de saída, mostrando como uma ou mais entradas são convertidas em saídas. O processo é graficamente representado por um círculo, por uma figura oval ou por um retângulo com os cantos arredondados. O nome do processo deve descrever o que ele faz e pode ser composto por uma frase constituída de um verbo e um objeto. Exemplos: receber pedido; diagnosticar paciente; desinterditar.

b) fluxos de dados − os fluxos normalmente representam caminhos por onde passam os dados em movimento, transportados entre os elementos do DFD. Os tipos de fluxo são:

   I. fluxo externo − entre entidade externa e processo.

II. fluxo interno – entre dois processos.

III. fluxo de acesso à memória – entre processo e depósito.

IV. fluxo de erro ou rejeição – para fora de um processo.

O fluxo é graficamente representado por uma seta ou um vetor que entra ou sai de um processo e indica o destino do dado. O sentido do fluxo indica entrada, saída ou diálogo. Cada fluxo deve ter um nome único que identifica e representa as informações transportadas. Esses nomes devem fazer parte do dicionário de dados. Exemplos: solicitação; login e senha; pedido.

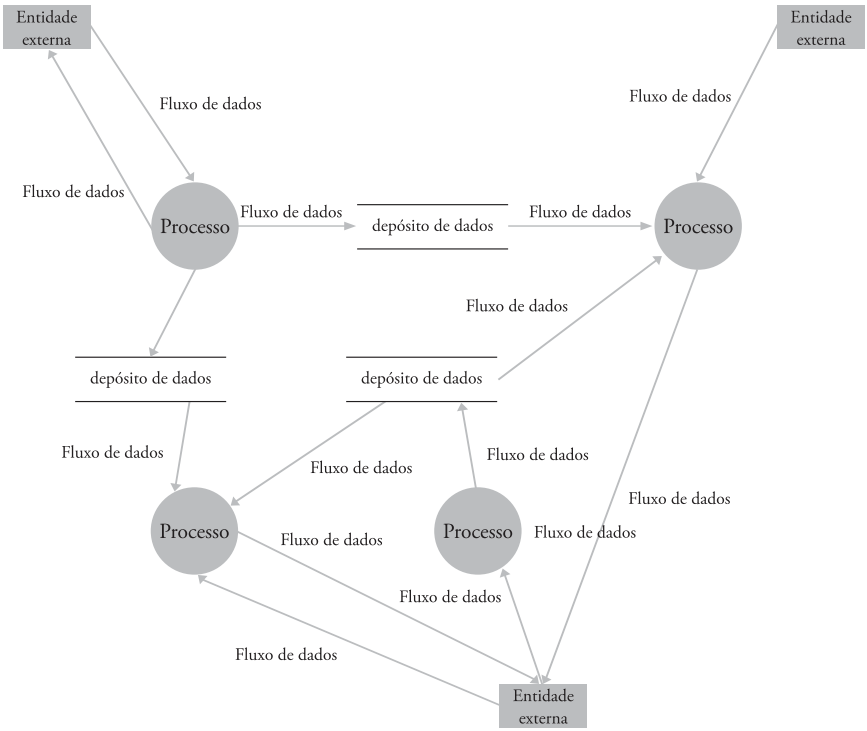
c) arquivos/depósitos de dados – armazenamento de dados utilizado para modelar e representar uma coleção de dados em repouso para acesso e/ou atualização futura através de um processo. Nem sempre um depósito de dados é um arquivo ou Sistema de Gerenciamento de Banco de Dados (SGBD), visto que pode representar também formas não computadorizadas. O nome para identificar o depósito pode ser o plural do nome dos pacotes transportados pelos fluxos para dentro e para fora do depósito. Exemplos: estimativas; clientes; usuário DB.

d) entidades externas/componentes terminadores – entidades externas são as fontes e os destinatários dos dados que entram e que saem e com os quais o sistema se comunica e funcionam sempre como origem e destino desses dados. Representam coisas, pessoas, grupos de pessoas, organização externa, setor dentro de uma empresa ou outro sistema externo ao sistema em desenvolvimento que recebe e/ou envia dados ao sistema. Os fluxos que ligam as entidades aos processos do sistema representam a interface entre o sistema e o mundo externo. Qualquer relacionamento existente entre entidades externas não deve ser mostrado no DFD. Sua nomenclatura pode ser utilizada no plural quando representar um grupo de pessoas ou deve conter o nome do setor ou da organização externa e a

palavra *sistema* quando for um sistema. Exemplos: usuários; Assistência Técnica; sistema de compras.

A figura 5.4 mostra relações possíveis entre estes elementos em um DFD genérico.

Figura 5.4 - Modelo de diagrama de fluxo de dados

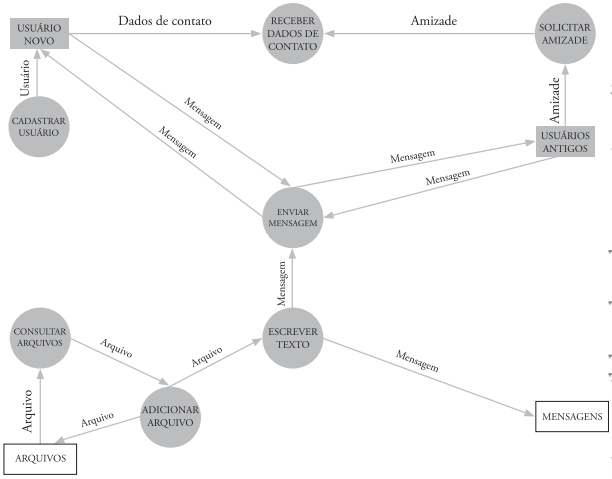


Fonte: Elaborado pela autora (2016).

5.2.2 Exemplos

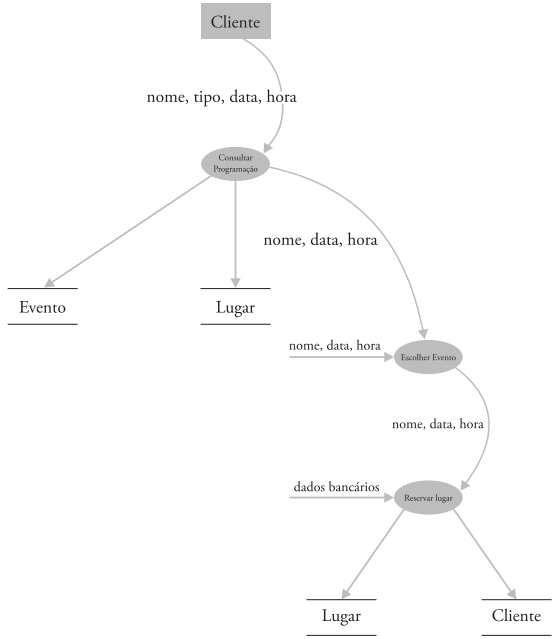
As figuras a seguir representam alguns exemplos de DFD; a figura 5.5 mostra um DFD de um programa de conversação instantânea e a Figura 5.6, o DFD de um sistema de venda de ingressos.

Figura 5.5 - Diagrama de fluxo de dados de um programa de conversação instantânea



Fonte: Elaborado pela autora (2016).

Figura 5.6 - Diagrama de fluxo de dados de um sistema de venda de ingressos



Fonte: Elaborado pela autora (2016).

5.3 Diagrama de entidade e relacionamento

O diagrama de entidade e relacionamento (DER) é muito utilizado na modelagem de banco de dados e representa o modelo de dados de um sistema para facilitar ao projetista a construção do modelo de dados. É a principal ferramenta e a principal representação gráfica do Modelo de Entidades e Relacionamentos (MER), pois mostra os arquivos como entidades e a ligação entre elas como relacionamentos.

5.3.1 Componentes

Os elementos do DER são:

a) entidades – tabelas ou conjuntos de objetos do mundo real dos quais se deseja manter informações no banco de dados. São representadas no diagrama por um retângulo e possuem nome e atributos.

II. Nome: substantivo concreto ou abstrato que representa a função de uma entidade dentro do domínio. Exemplos: cliente; agente; equipamento.

III. Atributo: característica relacionada a cada ocorrência de uma entidade ou de um relacionamento que pode ser mostrado para melhorar o entendimento do diagrama.

- Chave primária: é o atributo que representa um valor único da entidade a que pertence dentro do domínio, não podendo repetir seu valor na tabela, por exemplo: o ID de uma publicação; o CPF de um cliente; o código de um produto.
- Chave estrangeira: é o atributo ligado a uma chave primária de outra entidade, por exemplo: o nome do departamento para o qual um empregado trabalha; a matrícula do empregado responsável pelos seus dependentes; o CodMatricula do aluno de uma turma.
- Atributo descritivo: é o atributo que não é chave primária nem estrangeira, por exemplo: o nome de um empregado; a rua onde mora um cliente; o telefone de uma pessoa.

- Os atributos podem ser classificados em:
- simples – um único atributo que define uma característica da entidade, por exemplo: a nacionalidade de alguém; um convênio ou plano de saúde; a idade de uma pessoa.
- composto – vários atributos usados para definir uma informação, por exemplo: um endereço (contém logradouro, número, complemento); um nome (contém primeiro nome e sobrenome); uma moradia (contém localidade, rua, número).

As entidades podem ser classificadas como físicas ou lógicas, de acordo com sua existência no mundo real.

I. Entidade física: são objetos concretos, existentes e visíveis no mundo real. Exemplos: proprietário; produto; professor.

II. Entidade lógica: são objetos abstratos (não físicos) no mundo real. Exemplos: compra; consulta; categoria.

Também podem ser classificadas segundo o motivo de sua existência:

I. Entidade forte – sua existência não depende da existência de outras entidades. Exemplos: cliente; aluno; pessoa.

II. Entidade fraca – sua existência depende da existência de outra(s) entidade(s). Exemplos: inadimplente (depende da entidade paciente no sistema de um consultório); lista de compras (depende da entidade produto); pedido (depende das entidades cliente e produto).

III. Entidade associativa – é uma entidade intermediária cuja identificação é formada pelas chaves primárias de outras entidades. Exemplos: Hospedagem_Servico (associação entre as entidades hospedagem e serviço); Emprestimo-Cliente (associação entre as entidades empréstimo e cliente); utilização (associação entre as entidades pessoa e computador).

b) relacionamentos – um relacionamento é uma associação entre duas ou mais entidades, que demonstra como funciona sua interação e

também a forma como os atributos também podem interagir. É representado por um losango e por linhas que ligam as entidades relacionadas. Exemplos: um aluno está matriculado em uma turma; uma pessoa_fisica é uma pessoa; um produto possui fornecedor.

Podem ser classificados de três formas:

I. relacionamento 1..1 (um para um) − cada uma das duas entidades envolvidas referencia obrigatoriamente apenas uma unidade da outra. Exemplos: um curso é coordenado por apenas um coordenador que coordena somente um curso; um usuário é uma pessoa única; um contador possui apenas um perfil que é desse contador.

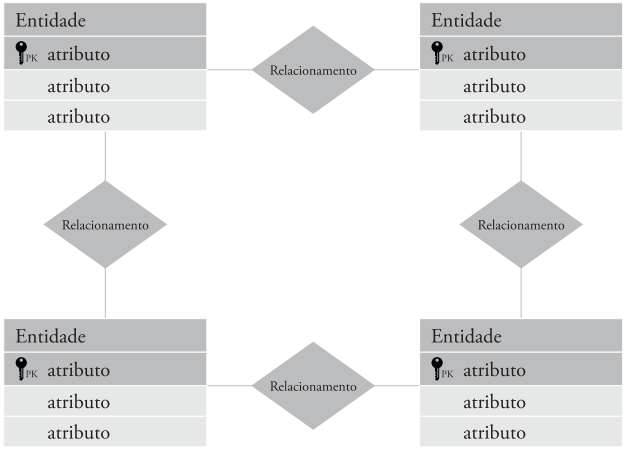
II. relacionamento 1..n ou 1..* (um para muitos) − uma das entidades envolvidas pode referenciar várias unidades da outra, que só pode estar ligada a uma unidade daquela. Exemplos: um hospital é formado por vários ambulatórios que formam somente esse único hospital; um cliente efetua muitas vendas, mas uma venda só é registrada exclusivamente a um único cliente; um proprietário pode possuir vários imóveis que são apenas desse único proprietário.

III. relacionamento n..n ou *..* (muitos para muitos) − cada entidade do relacionamento pode referenciar múltiplas unidades da outra. Exemplos: um médico consulta vários pacientes que podem ser consultados por vários outros médicos; um cliente aluga vários jogos que podem ser alugados por vários outros clientes; um professor leciona para várias turmas que também possuem vários outros professores.

Notações mais modernas passaram a aplicar o formato mais utilizado na *Unified Modeling Language* (UML − Linguagem de Modelagem Unificada, em português), em que os atributos já aparecem listados na própria entidade, pois isso torna o diagrama mais limpo e fácil de ser lido.

A figura 5.7 mostra relações possíveis entre estes elementos em um DER genérico.

Figura 5.7 - Modelo de diagrama de entidade e relacionamento

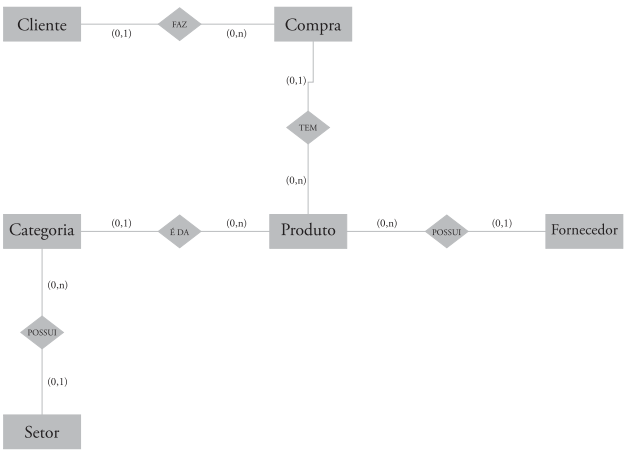


Fonte: Elaborado pela autora (2016).

5.3.2 Exemplos

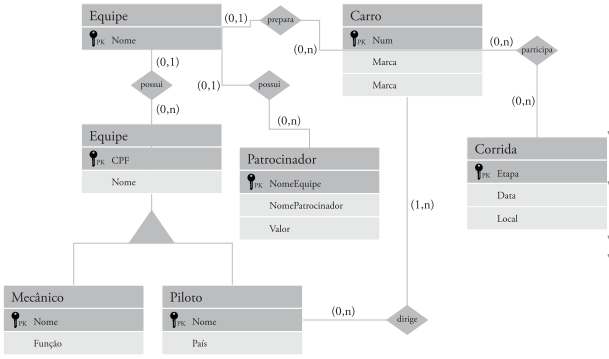
As figuras a seguir mostram alguns exemplos de DER; a Figura 5.8 representa o DER de um sistema de venda, e a Figura 5.9 mostra o DER do sistema de um campeonato de automobilismo.

Figura 5.8 - Modelo de diagrama de entidade e relacionamento de um sistema de vendas



Fonte: Elaborado pela autora (2016).

Figura 5.9 - Modelo de diagrama de entidade e relacionamento de um campeonato de automobilismo



Fonte: Elaborado pela autora (2016).

5.4 Dicionário de dados

Durante a fase de análise, é criado também um documento chamado dicionário de dados (DD), que contém a especificação de todos os objetos existentes no DER. Um DD é um grupo de tabelas ou uma base de dados propriamente dita que contém todos os fluxos de entrada e de saída e os depósitos de dados e descreve o significado, as características lógicas e as representações de todos os elementos usados em um sistema.

5.4.1 Componentes

Os elementos do DD são:

a) entidade − descreve o nome da entidade definida no DER e pode ser uma pessoa, um objeto ou um lugar que é considerado como objeto. Exemplos: cliente (entidade que armazena clientes); alunos (entidade que armazena alunos); Tb_Venda (entidade que armazena vendas).

b) atributo − os atributos são as características da entidade. Exemplos: PES_CPF (CPFs de pessoas); PED_CODIGO (códigos de pedidos); fk01_idTecnico (identificadores de técnicos).

c) classe − as classes podem ser:

I. simples – um atributo normal. Exemplos: ENVIOPDF (envio de arquivo do tipo PDF); ME_VALOR (valor de uma mesa); S_DATE (uma data qualquer).

II. composta – um atributo que pode ser dividido em outros atributos. Exemplos: at01_nomeGerente (nomes podem ser divididos em primeiro nome, nome do meio, sobrenome); Address (endereços podem ser divididos em logradouro, número, complemento); CA_Telefone (telefones podem ser divididos em DDI, DDD, número).

III. multivalorada – um atributo cujo valor pode não ser único. Exemplos: telefone (um cliente pode ter mais de um telefone); e-mail (um cliente pode ter mais de um e-mail); PES_ENDERECO (uma pessoa pode ter mais de um endereço).

IV. determinante – um atributo que é usado como chave. Exemplos: idEndereco (identificador de um endereço); Id (identificador de algo); cod_form (código de um formulário).

d) domínio – é o tipo do valor do atributo.

e) tamanho – é a quantidade de caracteres necessários para armazenar o conteúdo.

f) descrição – é opcional e usada para descrever o que é cada atributo. Exemplos: ID_album é a chave de identificação do álbum; Nome_fornecedor é o atributo que representa o nome do fornecedor; DATAF é a data de fim.

Quadro 5.1 - Modelo de dicionário de dados

Entidade				
Atributo	**Classe**	**Domínio**	**Tamanho**	**Descrição**
Atributo 1				
Atributo 2				
Atributo 3				

Fonte: Elaborado pela autora (2016).

5.4.2 Exemplos (PESO 2)

Os quadros a seguir representam alguns exemplos de DD; o Quadro 5.2 mostra a entidade COUNTRIES de um sistema qualquer, e o Quadro 5.3 mostra a entidade pessoa de um sistema qualquer.

Quadro 5.2 - Dicionário de dados da entidade COUNTRIES

Nome	Descrição	Tipo de Dados	Valor Padrão
Codigopais	Codigopais	Autonumeração	
Nome	Nome	Texto	

Fonte: Elaborado pela autora (2016).

Quadro 5.3 - Dicionário de dados da entidade pessoa de um sistema

TABELA: PESSOA					
	CAMPO	**DESCRIÇÃO**	**TIPO**	**TAM**	**DEC**
PK	PES_CODIGO	Código da Pessoa	INTEIRO	6	-
FK	MAE_ CODIGO	Código da Mãe	INTEIRO	6	-
	PES_NOME	Nome da Pessoa	CADEIA DE CARACTERES	100	-
	PES_DATA-NASCIMENTO	Data de Nascimento da Pessoa	DATA	-	-
	PES_DATAFALECIMENTO	Data de Falecimento da Pessoa	DATA	-	-

PK – Primary Key (Chave Primária)

FK – Foreign Key (Chave Estrangeira)

Fonte: Elaborado pela autora (2016).

5.4.3 Ocorrências

Cada entrada no DD é formada por um identificador e sua respectiva descrição, a qual inclui significado, conteúdo, valores e unidades permitidas e chave primária de cada entrada.

Para descrever de forma precisa e concisa cada componente de dados, utiliza-se um conjunto de símbolos simples, como mostra o Quadro 5.4.

Quadro 5.4 - Símbolos do DD

SÍMBOLO	SIGNIFICADO
=	Composição: é composto por
+	Concatenação: E
( )	Opção: enquadram componentes opcionais
{ }	Iteração: enquadram componentes cavalares, que podem se repetir em nenhuma ou mais ocorrências
[ ]	Seleção: enquadram componentes que são utilizados como escolha em uma das alternativas
| ou ,	Separação de alternativas: separa componentes alternativos enquadrados por [ ]
* *	Comentário: enquadram comentários
@ ou sublinhado	Chave de identificação: identificador da chave primária em um depósito
“ ”	Valor discreto

Fonte: Elaborado pela autora (2016).

Exemplos:

N_produto = *Número de identificação de produto da loja* {dígito}

NomeCliente = titulo_de_cortesia + (primeiro_nome) + ultimo_nome

Endereço-cliente = Endereço-residentcial + (endereço-comercial)

No DD, todos os dados do DFD e todos os objetos do DER devem estar definidos.

Síntese

As ferramentas básicas da atividade de descrição da análise de dados de um sistema são o diagrama de contexto (DC), o diagrama de fluxo de dados (DFD), o dicionário de dados (DD) e o diagrama de entidade e relacionamento (DER).

O DC é um diagrama que representa o sistema como um todo, composto por sistema, atores/entidades e dados (entradas e saídas). Já o DFD é um diagrama que representa o fluxo dos dados pelo sistema de forma mais detalhada. É composto por processos, entidades externas/componentes terminadores, fluxos de dados e arquivos/depósitos de dados.

O DER, por sua vez, é um diagrama que representa a modelagem do banco de dados, composto por entidades e relacionamentos. E o DD é um documento que especifica os dados do DER e do DFD, composto por entidades que contêm atributos como classe, domínio, tamanho e descrição.

Atividades

1. Quais são os principais diagramas da análise de sistemas?

a) DC, DFD e DER

b) DFD, DER e DD

c) DC, DFD e DD

2. O que é um diagrama de fluxo de dados?

a) É um diagrama de alto nível, que representa todo o sistema como um único processo.

b) É um diagrama dos processos do sistema e dos dados que ligam esses processos, descrevendo o fluxo de informação e a estrutura do sistema a ser desenvolvido.

c) É um diagrama que representa o modelo de dados de um sistema para facilitar ao projetista do banco de dados a construção do modelo de dados.

3. Quais são os elementos componentes do DFD?

a) Sistema, entidades e dados

b) Processos, entidades externas, fluxo de dados e depósitos de dados

c) Entidades e relacionamentos

4. O que é um diagrama de entidade e relacionamento?

a) É um diagrama de alto nível, que representa todo o sistema como um único processo.

b) É um diagrama dos processos do sistema e dos dados que ligam esses processos, descrevendo o fluxo de informação e a estrutura do sistema a ser desenvolvido.

c) É um diagrama que representa o modelo de dados de um sistema para facilitar ao projetista do banco de dados a construção do modelo de dados.

6

Linguagem de Modelagem Unificada

Grandes aplicações de negócios que realizam as atividades da funcionalidade principal da empresa e a mantêm em funcionamento precisam ter uma estrutura que possibilite segurança e execução sob condições específicas. É preciso projetar essa estrutura, que é também chamada de arquitetura, de maneira muito clara. Assim, os programadores que fazem a manutenção no código conseguem, eficientemente, encontrar e corrigir um erro percebido muito tempo após os programadores do código original terem sido alocados em outros projetos.

Apesar de a funcionalidade principal ser o núcleo essencial do negócio, ela não deve ser a única desenvolvida para ser executada sem erros. As aplicações precisam ser desenhadas para todas as áreas que envolvam o negócio, seja ele grande, seja pequeno. Uma arquitetura bem projetada traz benefícios, pois é uma maneira de trabalhar a complexidade conforme o porte do programa. Ela também possibilita a reutilização de código, portanto, é o melhor momento para estruturar um programa como um conjunto de elementos ou componentes independentes.

Algumas vezes, o negócio possui uma biblioteca de modelos de componentes, sendo que cada um deles representa uma implementação guardada em uma biblioteca de módulos de código. Quando outra aplicação precisar usar determinada funcionalidade, no momento da programação, pode-se eficientemente importar seu módulo de código da biblioteca para dentro da aplicação.

Este capítulo estuda os fundamentos da UML (*Unified Modeling Language* – Linguagem de Modelagem Unificada, em português), mas não detalhadamente, somente como uma introdução.

A atividade do desenvolvimento de aplicações de software antes da codificação é chamada de modelagem, uma parte fundamental e útil para grandes, médios ou pequenos projetos de software. A modelagem está para o desenvolvimento de software assim como plantas baixas, mapas e maquetes estão para a construção de um imóvel. Por meio desses modelos, os responsáveis pelo sucesso de um projeto de desenvolvimento de software conseguem garantir que a função do negócio seja perfeita, que os requisitos do usuário final sejam atendidos e que o projeto de software englobe esses requisitos e quaisquer características antes que a programação do código deixe as alterações difíceis ou caras de serem realizadas.

Pesquisas constatam que grandes projetos de aplicações apresentam uma grande possibilidade de falha, sendo que é mais provável que um grande software não atinja todos os requisitos dentro do prazo e do orçamento do que obtenha total sucesso.

UML é uma linguagem padrão para modelagem, orientada a objetos e que possibilita representar um sistema de software de maneira padronizada. Entretanto, não é um método de desenvolvimento.

Apesar de possibilitar a representação do sistema através de modelos orientados a objetos, a UML não mostra que tipo de trabalho precisa ser realizado, visto que não possui um procedimento que considere as atividades que precisam ser executadas. Ela não é uma metodologia de desenvolvimento, o que implica que não define a ordem do que deve ser feito nem a maneira como o software deve ser projetado.

A criação de um dicionário de dados completo é fundamental para registrar todas as tarefas realizadas, para assim detalhar todos os requisitos funcionais do sistema. A UML é muito utilizada na elaboração de modelos

de software, pois, de maneira geral, ajuda a entender o desenho e a relação entre os objetos. Ela também possibilita aos desenvolvedores conseguir ver os produtos do trabalho em diagramas padronizados, e sua representação gráfica também especifica a semântica dos diagramas.

Apesar de o RUP (*Rational Unified Process* – Processo Unificado da Rational, em português) ter sido integralmente produzido usando a UML, ela é uma representação independente de procedimentos. Além de prover a tecnologia necessária para suportar a prática de engenharia de software orientada a objetos, a UML pode ser a linguagem de modelagem padrão para modelar sistemas de software. Um conjunto de técnicas de notação gráfica para a elaboração de modelos visuais de sistemas é usado através da junção das melhores técnicas de modelagem de dados, negócios, objetos e componentes. É uma linguagem de modelagem unificada, popular e muito usada.

A UML se originou da fusão das melhores práticas de engenharia bem-sucedidas na modelagem de sistemas grandes e complexos. Ela foi gerada a partir da junção dos conceitos dos métodos BOOTCH, OMT (*Object-Modeling Technique*) e OOSE (Engenharia de software orientada a objetos), combinando-os em uma única linguagem de modelagem comum e amplamente usada. Os primeiros passos para a elaboração da UML tiveram origem em 1994, no momento em que o cientista da computação James Rumbaugh se juntou a Grady Booch na Rational Software. Visando combinar os métodos Booch e OMT, após um ano de trabalho, foi divulgada, em 1995, a primeira versão do Unified Process (Processo Unificado). Logo após, Ivar Jacobson se associou à Rational, e o escopo do projeto da UML foi complementado para abranger o método OOSE. Surgiu, assim, em 1996, outra versão da UML, que ainda não é um padrão da indústria, mas tem o objetivo de se tornar um padrão para o OMG (*Object Management Group*), que visa à criação de uma linguagem de modelagem de software. Vários gestores querem ajudar a criar esse padrão, e pretende-se usar a UML como a linguagem de modelagem padrão para modelar sistemas.

A finalidade da UML é descrever o que, como, quando e por que deve ser feito. Em outras palavras, trata-se da especificação, da documentação e da estruturação para visualização lógica do desenvolvimento completo de um sistema de informação. Com o objetivo de extrair todas as visualizações e características do sistema, a UML tem um conjunto de diagramas utilizados simultaneamente, os quais podem ser divididos em estruturais e comportamentais.

6.1 Diagramas

Um diagrama de UML é a representação gráfica da informação de um modelo de UML, mas o modelo é independente do diagrama.

A UML define 13 tipos de diagramas, divididos em duas categorias: seis tipos de diagramas estruturais, que representam a estrutura de aplicação estática, e quatro comportamentais, que representam tipos gerais de comportamento. Desses últimos, o diagrama de interação pode ser subdividido em quatro diagramas que representam diferentes aspectos de interações.

a) Diagramas estruturais:
   I. Diagrama de classe
   II. Diagrama de objeto
   III. Diagrama de componente
   IV. Diagrama de estrutura
   V. Diagrama de pacote
   VI. Diagrama de implantação

b) Diagramas comportamentais:
   I. Diagrama de caso de uso
   II. Diagrama de atividade
   III. Diagrama de máquina de estados
   IV. Diagramas de interação:
      × Diagrama de sequência
      × Diagrama de comunicação
      × Diagrama de tempo
      × Diagrama geral de interação

Neste capítulo, será dada ênfase a três diagramas, que são os mais estudados nas universidades e mais utilizados nas empresas pelas equipes de desenvolvimento de software: o diagrama de classe, o diagrama de caso de uso e o diagrama de sequência. Assim, contempla-se o estudo de um diagrama de cada tipo citado acima.

6.1.1 Diagrama de classe

Fora do sistema, os usuários visualizam resultados de cálculos, relatórios produzidos, requisições solicitadas, entre outros. Dentro do sistema, os objetos colaboram uns com os outros para produzir os resultados esperados pelos usuários.

Um diagrama de classe é o diagrama central da modelagem orientada a objetos. É uma representação com o objetivo de definir e descrever as informações da estrutura usada pelo aplicativo. Não faz referência a qualquer implementação específica, mas mostra os relacionamentos de um conjunto de todas as classes que o sistema necessita possuir. Essas classes servem de modelo para os vários tipos de objetos do sistema e podem ser implementadas de várias maneiras. O diagrama de classe apresenta como as classes interagem entre si e qual é a responsabilidade de cada uma delas na realização das operações solicitadas pelos atores. É a base para a construção de outros diagramas, como o de sequência.

Os elementos de um diagrama de classe são:

a) classes − uma classe é um elemento abstrato que representa um conjunto definido de objetos, sendo cada objeto um exemplo de determinado grupo que compartilha características comportamentais ou estruturais. A classe contém a especificação do objeto, as características (atributos) e as ações ou os comportamentos (métodos). Graficamente, as classes são representadas por retângulos incluindo nome, atributos e métodos.

   I. Nome: o padrão mais comum adotado para nomear as classes é aquele em que todos os nomes de classes são substantivos singulares sempre iniciados com letra maiúscula. Exemplos: Esporte; Paciente; Banco.

   II. Atributo: representa o conjunto de características dos objetos da classe, como visibilidade ou nível de encapsulamento, nome, tipo de dados, multiplicidade, valor inicial e propriedade. Exemplos: – codigo : int, + description : string, – dt_ criacao : String.

      × Visibilidade: pode ser público (visível em qualquer classe de qualquer pacote), protegido (visível para classes do mesmo pacote), privado (visível somente para classe) ou

pacote (acessado pelo relacionamento da classe com a classe externa).

- Nome: demonstra as características do objeto.
- Tipo de dados: pode ser String, boolean, int, float, double, Date, entre outros.

III. Método: representa o conjunto de operações que a classe fornece, com visibilidade ou nível de encapsulamento, nome, lista de parâmetros e tipo de retorno. Exemplos: + CriarAtividade() : void, + Apresentar relatório(), # set(text : String) : *boolean*.

- Visibilidade: pode ser público (visível em qualquer classe de qualquer pacote), protegido (visível para classes do mesmo pacote) ou privado (visível somente para classe).
- Nome: deve expressar a ação que realiza e não deve possuir espaços nem começar com dígitos.
- Lista de parâmetros: devem vir entre parênteses e separados por vírgula.
- Tipo de retorno: informa que tipo de dado o método deve retornar após sua execução.

b) relacionamentos – são associações entre as classes, com nome, multiplicidades e navegabilidade. Exemplos: um cliente faz um pedido de alguns livros; uma loja usa um catálogo de produtos que contém as especificações de alguns produtos; um gestor organiza um evento com uma atividade realizada por um ministrante e assistida por alguns congressistas. Os relacionamentos possuem:

I. nome – descrição dada ao relacionamento.

II. navegabilidade – indicada por uma seta no fim do relacionamento.

III. tipo – associação (relacionamento entre objetos de classes diferentes), generalização (relacionamento entre itens gerais e específicos) ou dependência (relacionamento em que a alteração de um objeto pode afetar outro objeto).

IV. papéis – desempenhados por classes em um relacionamento.

Um diagrama de classes pode oferecer três perspectivas, cada uma para um tipo de observador diferente. São elas:

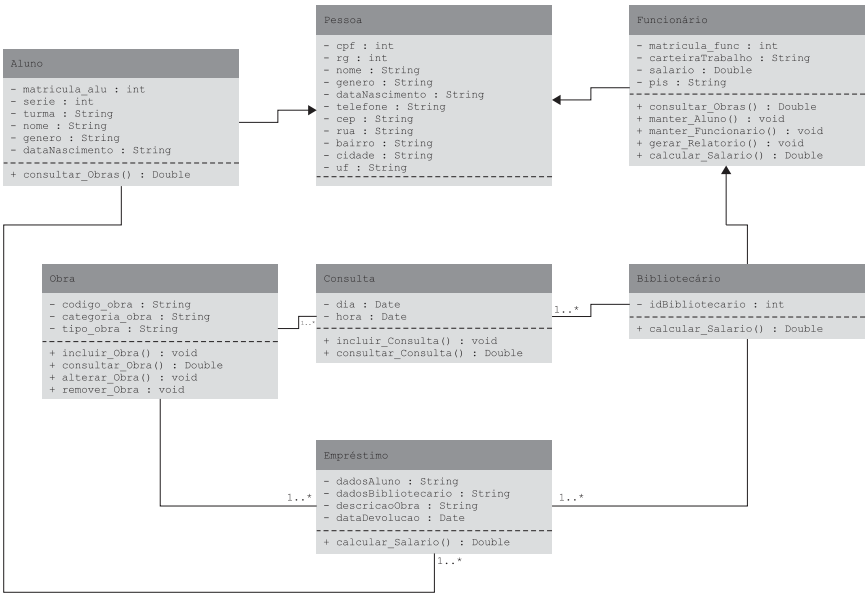
a) conceitual − aborda os conceitos do domínio em estudo. Essa perspectiva é destinada ao cliente.

b) especificação − representa as principais interfaces da arquitetura, nos principais métodos. Essa perspectiva é destinada aos gerentes de projeto.

c) implementação − tem foco em vários detalhes de implementação. Essa perspectiva é destinada ao time de desenvolvimento.

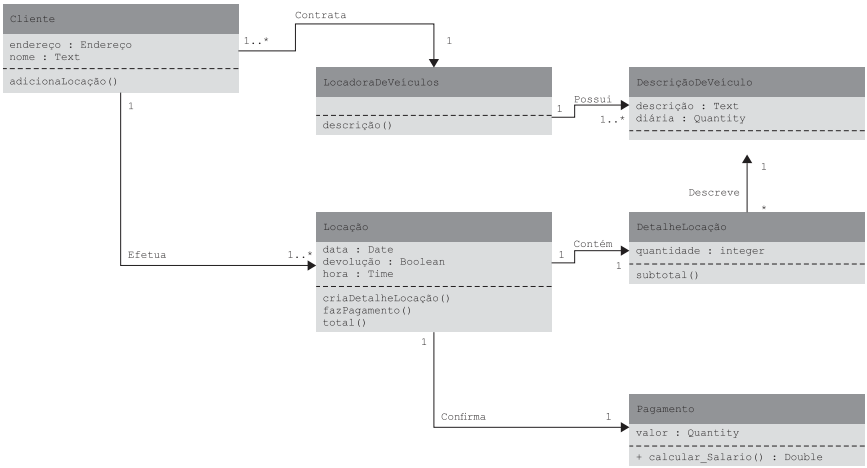
As figuras a seguir mostram alguns exemplos de diagramas de classes. A figura 6.1 representa o diagrama de classe do sistema de uma biblioteca escolar e a Figura 6.2 mostra o diagrama de classe do sistema de uma locadora de veículos; já a Figura 6.3 apresenta o diagrama de classe do sistema de serviços de manutenção de uma empresa.

Figura 6.1 - Diagrama de classe do sistema de uma biblioteca escolar



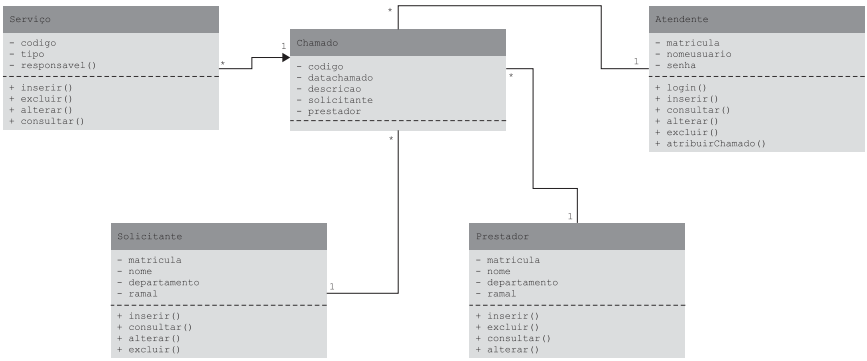
Fonte: Elaborado pela autora (2016).

Figura 6.2 - Diagrama de classe do sistema de uma locadora de veículos



Fonte: Elaborado pela autora (2016).

Figura 6.3 - Diagrama de classe do sistema de serviços de manutenção de uma empresa



Fonte: Elaborado pela autora (2016).

6.1.2 Diagrama de caso de uso

É possível definir um diagrama de caso de uso como um tipo de documento classificador de narrativas de texto que descreve e representa uma unidade funcional coerente fornecida pelo sistema ou subsistema. É uma classe

manifestada por sequências de eventos e mensagens intercambiáveis de interação entre um sistema e um ou mais atores que o utilizam para completar um processo.

Pelo fato de darem uma visão externa do sistema, os casos de uso são muito utilizados para descobrir e registrar requisitos funcionais, visto que descrevem o que o sistema faz. Apesar de não especificarem como isso deve ser feito, precisam ser capazes de comunicar a funcionalidade e o comportamento do sistema para o cliente. A descrição de um caso de uso deve conter basicamente o fluxo de eventos. Há vários modelos que podem ser seguidos, entre eles alguns exemplos mostrados nos quadros 6.1 e 6.2.

Quadro 6.1 - Modelo de caso de uso 1

Este caso de uso começa quando "(Nome do caso de uso)" pede que o sistema (faça algo).

1. O sistema (faz algo). Se (condição), então (algo acontece), e o caso de uso termina.

2. O sistema (faz algo). Se (condição), o sistema (faz algo).

Fonte: Elaborado pela autora (2016).

Quadro 6.2 - Modelo de caso de uso 2

UC000 – Nome do caso de uso	
Objetivo:	Este UC tem como objetivo...
Requisitos:	(RFs contemplados no UC)
Atores:	(Atores do UC)
Prioridade:	(Alta, normal, baixa...)
Pré-condições:	(Pré-condições para a execução do UC)
Frequência de uso:	(Frequência de uso do UC)
Campos:	(Campos da tela)
Fluxo principal:	1. O usuário (faz algo); 2. O sistema (faz algo); 3. O fluxo é encerrado.

UC000 – Nome do caso de uso	
Fluxo alternativo:	A1 – (Nome do fluxo alternativo): 1. O usuário (faz algo); 2. O sistema (faz algo); 3. OUC retorna ao passo 2 do fluxo principal.
Fluxo de exceção:	E1 – (Nome do fluxo de exceção): 1. O sistema (faz algo); 2. OUC retorna ao passo 2 do fluxo principal.
Validações:	(Parâmetros para validação)
Protótipo das telas:	(Imagens dos protótipos)

Fonte: Elaborado pela autora (2016).

O diagrama de caso de uso é um diagrama da UML cuja finalidade é representar um requisito do sistema a ser informatizado e ajudar na comunicação entre os analistas e o cliente. A descrição e os diagramas de casos de uso são uma técnica de modelagem de requisitos e descrevem o que um sistema deve fazer do ponto de vista do usuário. Também registram a interação dessas funcionalidades com os usuários do mesmo sistema.

Esses diagramas podem ser representados por uma elipse que contém, em seu interior, o nome do caso de uso, que trata somente dos requisitos funcionais de um sistema. Um diagrama de caso de uso não mostra os detalhes técnicos dos casos de uso que dizem como o sistema deve funcionar, apenas resume algumas das associações entre casos de uso, atores e sistemas. De fato, o diagrama não apresenta a prioridade de execução das etapas para alcançar os objetivos de cada caso de uso. Além disso, outros requisitos, como qualidade de serviço e restrições de implementação, devem ser apresentados em outros documentos, bem como a arquitetura e os detalhes internos.

Um diagrama de caso de uso descreve um cenário que apresenta as funcionalidades do sistema do ponto de vista do usuário, que deve visualizar no diagrama as principais funções de seu sistema.

Os diagramas de casos de uso são representados por quatro elementos ou componentes:

a) atores − elementos que interagem com o sistema, são usuários ou tipos de usuários do sistema e podem ser uma classe de organização, um usuário humano, um dispositivo ou componente de software externo que interage com o sistema ou outro sistema computacional. O ator representa os papéis desempenhados por um usuário, outro sistema que interage com o assunto ou elementos externos ao sistema. Um ator precisa ser externo ao sistema e é representado por um boneco e um rótulo com o nome do ator. Exemplos: Usuário; Cliente; Secretário.

b) casos de uso − definem e representam uma especificação de um conjunto de grandes funções, tarefas, funcionalidades ou ações do sistema executadas por um ou mais atores ou por um sistema em busca de uma meta específica. Casos de uso contêm um resultado observável e são iniciados por um ator ou por outro caso de uso. Um caso de uso é representado por uma elipse e um rótulo com o nome do caso de uso dentro ou abaixo e está associado com o ator que o realizar. Exemplos: Pedir Comida; Abrir Projeto; Iniciar Sessão.

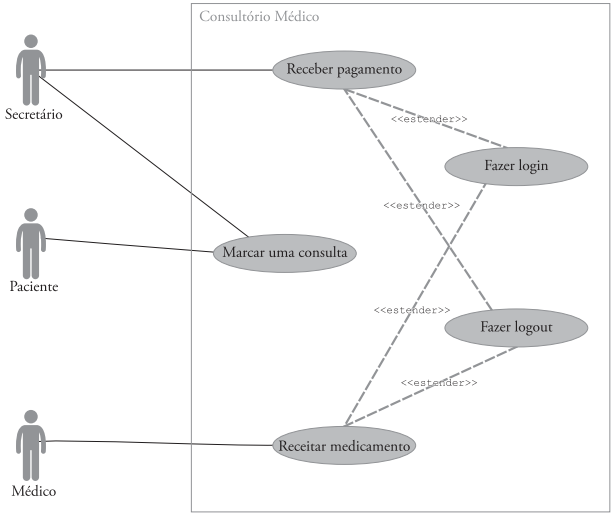
c) relacionamentos/comunicação − é o que associa um ator com um caso de uso e auxilia na descrição dos casos de uso.

d) cenário/sistema − elemento opcional, pode ser tudo o que está sendo desenvolvido: um componente de software pequeno, cujos atores são apenas outros componentes de software; um aplicativo completo; ou ainda um grande conjunto distribuído dos aplicativos implantados em vários computadores e dispositivos. Também é a sequência de eventos que acontecem quando o usuário interage com o sistema e serve para delimitar a área de atuação do sistema. É representado por um retângulo que envolve os casos de uso que compõem o sistema. O nome do sistema fica localizado dentro do retângulo. Exemplos: Subsistema Gestão de Pedidos; Caixa Automático; Jantar.

A seguir estão alguns exemplos de diagramas de caso de uso. A figura 6.4 mostra o diagrama de caso de uso do sistema de um consultório médico, a

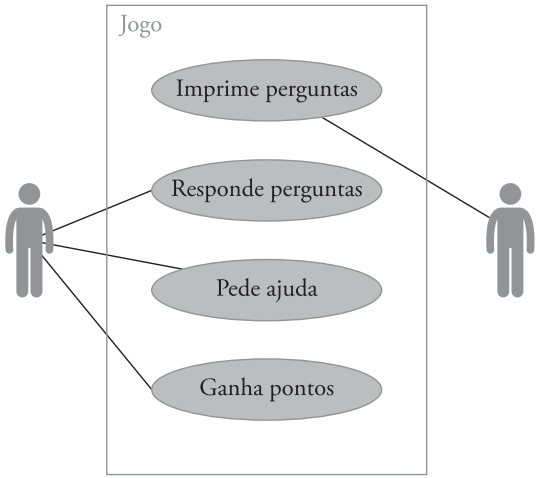
figura 6.5 representa o diagrama de caso de uso de um jogo, e a figura 6.6 traz o diagrama de caso de uso de um sistema de matrícula escolar.

Figura 6.4 - Diagrama de caso de uso do sistema de um consultório médico



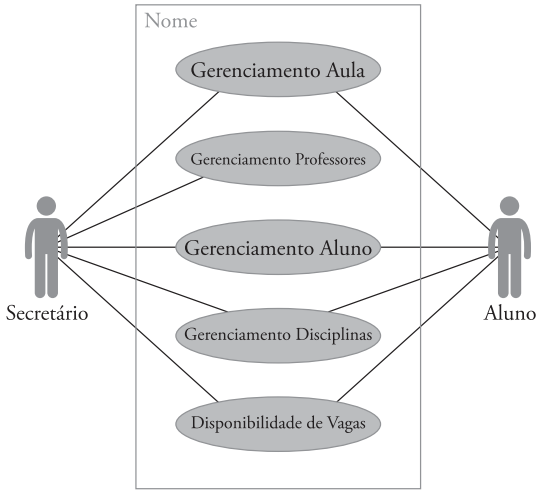
Fonte: Elaborado pela autora (2016).

Figura 6.5 - Diagrama de caso de uso de um jogo



Fonte: Elaborado pela autora (2016).

Figura 6.6 - Diagrama de caso de uso de um sistema de matrícula escolar



Fonte: Elaborado pela autora (2016).

6.1.3 Diagrama de sequência

Diagrama de sequência (de mensagens) consiste em um diagrama usado em UML. Tem o objetivo de estabelecer os objetos que interagem e seus relacionamentos e interações dentro de um contexto ou cenário. Também visa representar uma sequência de processos, operações ou métodos no decorrer do tempo. O diagrama de sequência representa principalmente como os grupos de objetos colaboram com algum comportamento do contexto de um caso de uso ao longo do tempo a partir das mensagens que são trocadas entre os objetos. Ele descreve de uma forma simples e lógica a sequência global do comportamento de vários objetos dentro de um contexto.

É construído a partir da escolha de um caso de uso do diagrama de casos de uso, que define o papel do sistema. Ele identifica os objetos que fazem parte da interação, inclusive o objeto que começa a interação, e as mensagens passadas entre esses objetos no caso de uso. Define como o software deve realizar seu papel pela sequência de operações e mensagens e dá ênfase à ordenação ou à perspectiva temporal em que as mensagens são trocadas entre os

objetos de um sistema para a realização de uma operação em um programa de computador.

Os elementos básicos em um diagrama de sequência são:

a) atores − são entidades ou participantes externos ao sistema que interagem com o sistema e solicitam serviços. Normalmente, o ator primário é o responsável pelo início do processo, por enviar a mensagem que inicia a interação entre os objetos tratada pelo diagrama de sequência.

b) objetos − caracterizam as instâncias das classes representadas no processo, simbolizados por retângulos no topo do diagrama. Os objetos têm nomenclatura padrão, com a sintaxe: nome_do_objeto:Sua_Classe, sendo o nome do objeto em minúsculo e o nome da classe em maiúsculo. Esses dois nomes devem ser separados por dois pontos (:). Exemplos: josé:Usuário, Cliente:Tela.

c) mensagens − objetos interagem por meio da troca de mensagens. A sintaxe para as mensagens é: sincronização condição sequência ":" retorno ":=" nomemensagem (parâmetro: tipoparâmetro) tiporetorno. Exemplos: 1:Acessa o grupo(), executarComando(), 3: Exibir Tela para Selecionar Período.

   Uma mensagem pode invocar uma operação sobre um objeto *(call)*, representar o retorno de um valor para o objeto que chamou a operação *(return)*, criar um objeto *(create)* ou eliminar um objeto *(destroy)*. As mensagens podem ser síncronas, assíncronas ou de retorno.

   I. Síncronas: o objeto emissor ou remetente fica bloqueado e aguardando a conclusão do processamento da mensagem até o objeto receptor ou destino recebê-la, tratá-la e respondê-la antes de continuar seu fluxo de execução. São representadas pela seta →.

   II. Assíncronas: o objeto emissor ou de origem continua seu processamento, emitindo mensagens independentemente do tratamento da mensagem feita no objeto destino, pois não exige

uma resposta antes de continuar e não há dependências. São representadas pela seta →.

III. De retorno: são mensagens que retornam para um participante que está aguardando o retorno de uma chamada anterior. São opcionais e podem indicar respostas ao autor e para objetos. São representadas pela seta ←.

d) linhas de vida − São linhas verticais que representam a sequência de eventos que ocorrem durante uma interação e durante o tempo de vida dos objetos do processo.

e) criação e destruição de objetos:

I. criação de objeto − indica que o objeto está executando algo. É representada por mensagem dirigida à própria caixa que representa o objeto. A mensagem de criação pode ser a sintaxe: <<create>>.

II. destruição de objeto − é representada por um X no fim da linha de vida do objeto. A mensagem de destruição pode ter a sintaxe: <<destroy>>. Pode ocorrer na recepção de mensagem ou no retorno de chamada. Um objeto pode se autodestruir.

f) iterações − uma mensagem pode ser enviada repetidas vezes.

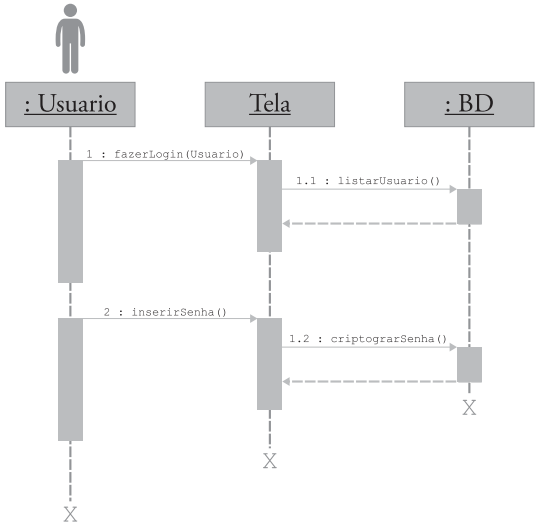
Um diagrama de sequência é representado da seguinte maneira:

a) linhas verticais − representam a linha ou o tempo de vida de um objeto, preenchidas por barras verticais que indicam o período de existência de um objeto – a parte inferior da barra é marcada por um X que representa o desaparecimento desse objeto.

b) linhas horizontais ou diagonais − representam mensagens trocadas entre os objetos. Contêm rótulos com o nome da mensagem, sempre posicionados acima da linha, sendo que, se o rótulo estiver entre colchetes, trata-se de uma condição. As linhas tracejadas são mensagens de retorno. As linhas horizontais contêm setas indicando o sentido da mensagem, e o formato da ponta da seta indica o tipo de mensagem (síncrona ou assíncrona).

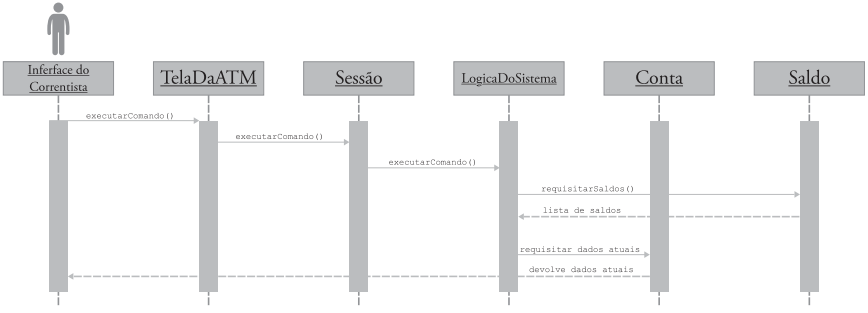
A seguir estão alguns exemplos de diagramas de sequência. A figura 6.7 mostra o diagrama de sequência da atividade de login em um sistema qualquer, a figura 6.8 representa o diagrama de sequência da atividade de gerar extrato em um sistema bancário, e a figura 6.9 traz o diagrama de sequência do sistema de um estacionamento.

Figura 6.7 - Diagrama de sequência da atividade de login em um sistema qualquer



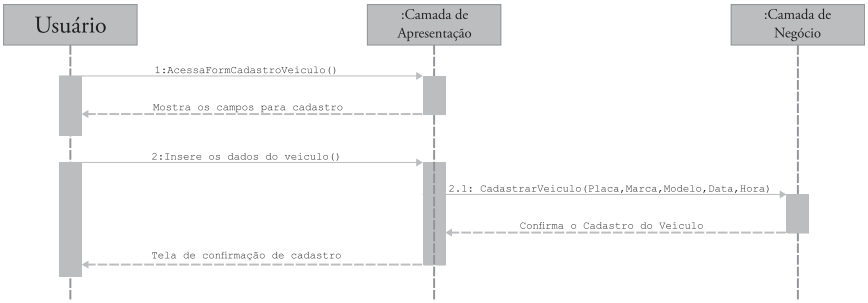
Fonte: Elaborado pela autora (2016).

Figura 6.8 - Diagrama de sequência da atividade para gerar extrato em um sistema bancário



Fonte: Elaborado pela autora (2016).

Figura 6.9 - Diagrama de sequência do sistema de um estacionamento



Fonte: Elaborado pela autora (2016).

Síntese

A UML é uma linguagem padrão para modelagem orientada a objetos que possibilita representar um sistema de software de maneira padronizada. Ela foi gerada a partir da junção dos conceitos dos métodos BOOTCH, OMT e OOSE, combinando-os em uma única linguagem de modelagem comum e amplamente usada. A finalidade da UML é descrever o que, como, quando e por que deve ser feito.

A UML define 13 tipos de diagramas, divididos em duas categorias: estruturais e comportamentais. Os diagramas mais estudados nas universidades e mais utilizados nas empresas pelas equipes de desenvolvimento de software são o diagrama de classe, o diagrama de caso de uso e o diagrama de sequência.

Um diagrama de classe é uma representação com o objetivo de definir e descrever as informações da estrutura usada pelo aplicativo. Já um diagrama de caso de uso descreve um cenário que apresenta as funcionalidades do sistema do ponto de vista do usuário, que deve visualizar no diagrama de caso de uso as principais funções de seu sistema. E um diagrama de sequência tem o objetivo de estabelecer os objetos que interagem e seus relacionamentos e interações dentro de um contexto ou cenário.

Atividades

1. Quais são os diagramas estruturais?

a) Diagrama de classe, de objeto, de componente, de estrutura, de pacote e de implantação

b) Diagrama de caso de uso, de atividade, de máquina de estados e de interação

c) Diagrama de sequência, de comunicação, de tempo e geral de interação

2. Quais são os elementos do diagrama de classe?

a) Classes e relacionamentos

b) Atores, casos de uso, relacionamentos e cenário

c) Atores, objetos, mensagens, linhas de vida

3. O que é um diagrama de caso de uso?

a) É uma representação com o objetivo de definir e descrever as informações da estrutura usada pelo aplicativo.

b) É um diagrama da UML cuja finalidade é representar um requisito do sistema a ser informatizado e ajudar na comunicação entre os analistas e o cliente.

c) É um diagrama que representa uma sequência de processos, operações ou métodos no decorrer do tempo.

4. O que é um diagrama de sequência?

a) É uma representação com o objetivo de definir e descrever as informações da estrutura usada pelo aplicativo.

b) É um diagrama da UML cuja finalidade é representar um requisito do sistema a ser informatizado e ajudar na comunicação entre os analistas e o cliente.

c) É um diagrama que representa uma sequência de processos, operações ou métodos no decorrer do tempo.

7

Projeto de Sistemas

O computador é uma das máquinas mais complexas já inventadas pelo homem, e os sistemas de software são ainda mais complexos. Portanto, projetar um sistema é uma tarefa complexa, pois significa necessariamente melhorar a velocidade de desenvolvimento, reduzir custos, reduzir o número e a complexidade das manutenções, aumentar a produtividade e ter vantagem competitiva. E o nível de complexidade dessa tarefa é diretamente proporcional ao tamanho do produto de software a ser projetado. Além disso, há o agravante das mudanças, geralmente oriundas da necessidade de alteração de funções do sistema. Este capítulo tem como objetivo definir o que é projeto de sistemas, comparando-o com a análise de sistemas.

Um projeto é uma ideia ou um empreendimento a ser executado ou realizado no futuro e dentro de determinado esquema. Um projeto de software possui muito em comum com um projeto de engenharia e está relacionado às ações a serem realizadas para atingir os objetivos levantados na análise. Ele integra a parte técnica do processo de desenvolvimento de software e é realizado independentemente do modelo de ciclo de vida executado. O projeto tem como objetivo incorporar a tecnologia aos requisitos para projetar o sistema a ser desenvolvido.

Projetar sistemas de software é definir como os requisitos devem ser implementados na forma de estruturas de software. Os requisitos do produto de software devem ser mapeados em soluções de tecnologia ou de software para o desenvolvimento do software. O manual do sistema a ser desenvolvido geralmente é esboçado na fase de projeto.

7.1 Diferenças entre análise e projeto

A análise se concentra na questão "o quê?". É a modelagem do problema e consiste em todas as atividades necessárias para entender o domínio do problema ou buscar descrever o que o sistema deve fazer. Ela dá ênfase para encontrar e descrever objetos no domínio do problema e é uma atividade de investigação da informação que é produzida com discussão e para aprovação pelo cliente. Não há preocupação com a implementação, visto que a fase de análise parte do pressuposto de que existe uma tecnologia perfeita. Por exemplo, supõe-se que a capacidade de processamento e de armazenamento seja ilimitada, a velocidade seja instantânea, o custo não exista e não haja falhas. Por isso, o analista deve conseguir abstrair informações sem importância para o sistema e mostrar o que realmente o sistema deve fazer e indicar o que não deve fazer.

Já o projeto aborda a questão "como?". Está mais próximo da implementação, é a modelagem da solução, consiste em todas as atividades necessárias para criar uma possível solução e preocupa-se em como a solução do sistema pode executar determinada tarefa. Ele dá ênfase para achar objetos lógicos de software e é uma atividade que resulta em informação que interessa apenas ao programador. Há uma preocupação com a arquitetura a ser utilizada, visto que a fase de projeto é a modelagem de como o sistema deve ser implementado já com os requisitos tecnológicos ou não funcionais. Por isso, o projetista deve conseguir mostrar como as partes do sistema devem ser construídas e como elas podem interagir.

Diferentemente da análise propriamente dita, o projeto é a modelagem do sistema de software considerando os requisitos tecnológicos, também chamados de não funcionais. Enquanto a análise define o "o quê", descobrindo o que o cliente quer, o projeto trabalha o "como", propondo uma solução para o cliente. O projeto de sistemas é um processo que envolve a elaboração

da solução a ser utilizada no desenvolvimento de um sistema, ou seja, ele dá ênfase para propor uma solução para o problema de forma a atender aos requisitos especificados durante a análise para projetar o que será construído na implementação. Deve contemplar todos os requisitos explícitos definidos pela análise e também os requisitos implícitos desejados pelo cliente.

O objetivo da especificação de projeto é incorporar a tecnologia aos requisitos essenciais do cliente, concretizando os recursos necessários, como tempo, investimento e tecnologias adicionais necessárias para o desenvolvimento de um sistema com qualidade. Segundo Pressman (2005), o "projeto é a única maneira com a qual se consegue traduzir com precisão os requisitos do cliente em um produto de software ou sistema finalizado". Para isso, é preciso ter conhecimento das tecnologias disponíveis nos ambientes de desenvolvimento e de produção. Seus artefatos devem ser compreendidos pelos programadores, tanto os responsáveis pela codificação quanto os responsáveis pela manutenção do sistema, e também pelos testadores. Um manual preliminar do sistema é desenvolvido como artefato dessa fase. Resumindo, o projeto nada mais é do que uma extensão ou um complemento da análise, pois define como os requisitos devem ser implementados na forma da estrutura interna de um sistema.

Não há, entretanto, definição que separe a análise do projeto. A análise sempre acaba por envolver um pouco do projeto, pois o cliente pode e deve discutir alguns tipos de interações que ocorrerão principalmente no que diz respeito à interface do usuário, entre outras pequenas discussões da solução, além da aprovação do modelo de análise. Além disso, alguns diagramas de análise e da UML (*Unified Modeling Language* − Linguagem de Modelagem Unificada, português) podem ser desenvolvidos na fase de projeto, como o diagrama de classe, o de sequência, o de estados, o de atividade, o de implantação e o de entidade e relacionamento.

7.2 Diretrizes e princípios de projeto

Existem alguns princípios e algumas diretrizes de qualidade de um projeto de software consideradas relevantes, algumas delas definidas por Pressman (2005).

- Um projeto deve ser modular.

- Um projeto deve conter representações distintas para dados, arquitetura, interfaces e componentes.
- Um projeto deve levar a componentes que possuam características de independência funcional.
- Um projeto deve levar a interfaces que reduzam a complexidade das conexões entre os componentes e o ambiente externo.
- O projeto deve ser rastreável para o modelo de análise.
- A arquitetura do sistema a ser construído sempre deve ser representada em sua totalidade.
- O projeto dos dados é tão importante quanto o projeto das funções de processamento.
- As interfaces (internas e externas) devem ser projetadas com cuidado, exibindo uniformidade.
- A interface com o usuário deve estar sintonizada e integrada com as necessidades do usuário final.
- O projeto, ao nível dos componentes, deve ser funcionalmente independente.
- Os componentes devem ser reutilizados, revisados e fracamente acoplados entre eles e com o ambiente externo, a fim de minimizar erros semânticos e a distância conceitual entre o software e o mundo real.
- As representações (modelos) de projeto devem ter orientações refinadas para serem entendidas facilmente por quem for construir cada detalhe.
- O projeto deve ser desenvolvido iterativamente.
- Diferentes visões do sistema devem ser providas.
- O projeto deve ser estruturado para acomodar mudanças e circunstâncias não usuais.
- O projeto deve apresentar nível de abstração superior ao código-fonte.

- O projeto deve ser passível de avaliação da qualidade.

7.3 Conceitos de projeto

Existem alguns conceitos básicos de projeto que também devem ser analisados.

a) Abstração: utiliza um conjunto selecionado de conceitos e regras de forma a focar aspectos específicos de interesse em um sistema. Quanto mais alto for o nível de abstração, mais próxima do ambiente do problema estará a linguagem; quanto mais baixo for o nível de abstração, mais computacional será a linguagem. Um bom projeto deve considerar vários níveis de abstração, e, à medida que se avança no processo de projeto, o nível de abstração deve ser reduzido.

b) Independência funcional: decorre da modularização, da ocultação de informações e dos níveis de abstração, obtida pelo desenvolvimento de módulos com finalidade única e alguma interação entre outros módulos. Seus objetivos são interfaces simples entre os módulos, evitando efeitos secundários e permitindo a reutilização dos módulos. Pode ser avaliada por meio de dois critérios de qualidade:

   I. Coesão − é a unidade de medida da força funcional entre os módulos. Em outras palavras, é a limitação do elo do módulo, ou seja, o quanto uma classe encapsula atributos e operações relacionadas entre si.

   II. Acoplamento − é a medida do grau de interdependência entre os módulos, ou seja, o grau com o qual as classes estão conectadas entre si. Módulos independentes podem ser testados e mantidos com mais facilidade.

c) Modularidade: é uma estratégia baseada na construção de produtos complexos a partir da divisão e da estruturação do sistema ou do software em partes distintas chamadas de subsistemas, módulos ou componentes. Estes são projetados e desenvolvidos individualmente, mas se mantêm integralmente interdependentes, em sua operação, para atender aos requisitos do problema. Um projeto

de programa modular facilita seu desenvolvimento e seu gerenciamento por meio da divisão em módulos, porque a complexidade é reduzida e há realização de atividades em paralelo, já que os módulos são desenvolvidos simultaneamente. Consequentemente, o sistema se torna mais legível e acaba tendo melhor manutenção, facilitando a mudança, e melhor desempenho.

d) Ocultação de informações: sugere que os módulos ou os componentes sejam caracterizados pelas decisões de projeto que cada módulo esconde dos demais. Cada módulo deve ser especificado de forma que suas informações sejam inacessíveis pelos outros módulos. Assim, os módulos interdependentes compartilham apenas informações necessárias.

e) Refinamento: é um processo de elaboração. Um programa é desenvolvido pelo refinamento sucessivo de níveis de detalhes procedimentais. Começa-se por uma função definida em alto nível de abstração, descrevendo a função conceitualmente, sem informações sobre o funcionamento interno dessa função.

7.4 Qualidade do projeto

A finalidade dessa etapa do projeto é adicionar os requisitos não funcionais ao sistema. Dessa maneira, o projetista deve atentar para os critérios de qualidade aos quais o sistema deve atender. A norma ISO/IEC 25010 define alguns critérios de qualidade:

a) funcionalidade − no processo de desenvolvimento de sistemas, a funcionalidade é definida como a capacidade de execução de determinada tarefa, comportamento, ação ou algo passível de execução. É tudo aquilo que um produto pode fazer e para o qual possa ser visualizado um início e um fim. Testar a funcionalidade quer necessariamente dizer que é preciso garantir que o produto funcione exatamente como foi especificado.

b) confiabilidade − em geral, a confiabilidade está intuitivamente associada a uma medida da capacidade, de probabilidade ou de garantia de um sistema, um item, uma instalação, um equipamento, um

dispositivo, um produto ou um serviço para desempenhar uma função. É a execução satisfatória e adequada de funcionalidades sistêmicas para manter seu funcionamento ou uma missão sem falhas. É dar como resposta, de forma adequada, aquilo que se espera sob certas condições ambientais específicas de uso e pré-estabelecidas em circunstâncias de rotina. Também em circunstâncias hostis e inesperadas, é atender requisitos não funcionais a seu propósito de acordo com determinadas especificações, como previsto no projeto. Deve ter uma duração de intervalo de tempo pré-determinada.

c) usabilidade − na interação humano-computador e na ciência da computação, a usabilidade nada mais é do que a implementação de recursos e de um atributo de qualidade dos produtos focando o usuário final. É também um termo usado para se referir à eficiência do design e para definir e descrever o conjunto de métodos destinados à mensuração e à melhoria do grau de facilidade de uso e de aprendizado. É também a simplicidade, a rapidez, a eficiência e a qualidade com que as pessoas ou o usuário consegue aprender a utilizar ou interagir com determinada interface, programa de computador, website ou ferramenta. Objetiva que seja possível realizar uma tarefa específica e importante sem perder a interação de suas funcionalidades com o sistema. Se um produto ou uma interface for fácil de utilizar, terá capacidade de satisfazer as necessidades do usuário de forma simples e eficiente, e o usuário terá maior produtividade, aprenderá mais rápido, memorizará as operações e cometerá menos erros. O termo, sinônimo de facilidade de uso, também é utilizado em contexto de produtos como aparelhos eletrônicos, em áreas da comunicação e em produtos de transferência de conhecimento, como manuais, documentos e ajuda on-line.

d) eficiência − em qualidade, eficiência geralmente consiste em uma particularidade, uma capacidade ou uma produtividade de algo que realiza corretamente suas funções, suas operações, suas tarefas ou seus trabalhos. Refere-se à relação entre atingir a eficácia dos resultados pré-determinados obtidos de modo eficaz, o melhor rendimento e o mínimo possível de erros ou de perda dos recursos utilizados.

e) manutenibilidade − em engenharia de software, manutenibilidade é o termo usado para definir uma característica, um aspecto ou um atributo da qualidade de software ou hardware que caracteriza um projeto de sistema, um produto de software ou um componente e possibilita que seja mantido ou modificado através de manutenções. Refere-se a certa facilidade, precisão, segurança, economia, viabilidade, tempo e frequência de manutenção. É também a adaptação de um software ou de determinado item na modificação ou na execução de ações de manutenção relacionadas a esse sistema, produto ou aparelho. Visa à correção de defeitos, melhorias, adaptação para uma mudança no ambiente operacional, adequação a novos requisitos ou a um ambiente novo e aumento da suportabilidade.

f) portabilidade − em informática, a portabilidade de um programa de computador, de um componente de hardware ou de software, de suas aplicações, de seus elementos ou de sua linguagem de programação é utilizada para determinar sua capacidade de ser transferida de ambiente. É a característica que permite que seja executada em quaisquer arquiteturas, sistemas, tipos de computadores, sistemas operacionais e outras plataformas além daquela de origem. É também a expectativa de ocorrência de falhas ou erros no sistema ou em algum de seus dispositivos.

7.5 Etapas do projeto

O projeto de um sistema pode ser dividido em duas partes: projeto arquitetural e projeto detalhado.

7.5.1 Projeto arquitetural, preliminar ou de alto nível

Tudo o que o ser humano produz, desde livros até imóveis, precisa de um projeto arquitetural, que precede a fase da construção e define como o produto a ser desenvolvido pode e deve ser dividido, para que suas partes ou seus componentes interajam entre si.

O projeto arquitetural transforma os requisitos do sistema em uma arquitetura de software e estruturas de dados, juntando as duas para definir interfaces que permitam que os dados transitem pelo software.

A finalidade do projeto preliminar é elaborar uma estrutura modular do programa e representar as relações de controle entre os módulos. Nessa etapa, são criados um esboço do manual do sistema e um documento com alguns requisitos de teste. Deve-se registrar o desenho de alto nível para as interfaces e as bases de dados. A interação entre as fases de projeto e a análise arquitetural permite o detalhamento da documentação da arquitetura do sistema a ser implementado.

A partir do projeto arquitetural, são avaliadas as tecnologias, suas características, suas limitações e suas restrições. É nesse momento que todos os aspectos tecnológicos devem ser observados, incluindo dispositivos de hardware e ferramentas de software. Portanto, durante o projeto arquitetural, o projetista ou o arquiteto de software deve tomar algumas decisões estratégicas que podem levar a consequências profundas. Ele deve estudar a melhor arquitetura, já que são possíveis várias configurações, para desenvolver o projeto levando em consideração os requisitos arquiteturais e os cenários de uso. Além disso, também devem ser consideradas as metas de qualidade desejadas para o sistema a ser desenvolvido.

A divisão do projeto em partes menores, a interação entre essas partes, a possibilidade de reuso de componentes no caso de projeto de sistemas e o atendimento aos requisitos não funcionais de qualidade, entre outros, são exemplos de decisões estratégicas a serem tomadas. O projeto arquitetural é uma fase iterativa do processo de desenvolvimento de software sobre a qual o projetista ou o arquiteto de software precisa tomar essas decisões, que podem ser reconsideradas durante a implementação do sistema de software.

O projeto de alto nível deve gerar o projeto da arquitetura do software.

7.5.1.1 Projeto da arquitetura do software

O projeto arquitetural é a etapa que produz a estrutura interna de um produto de software, também conhecida como arquitetura de software. Essa estrutura é a definição, a organização e a documentação do conjunto de elementos arquiteturais ou dos componentes de software do sistema, suas propriedades externas e seus relacionamentos com outros programas. Ela permite o particionamento do problema principal em problemas menores para fazerem corresponder os requisitos aos subsistemas e componentes.

O conceito de arquitetura de software se originou na década de 1960, em um trabalho de pesquisa do cientista da computação Edsger Dijkstra e posteriormente com um trabalho do também cientista David Parnas, e se popularizou na década de 1990. Do grego *arkhé*, que significa "chefe" ou "mestre", e *tékton*, que significa "trabalhador" ou "construtor", ou *tekhne*, que significa "arte" ou "habilidade", arquitetura é a arte de projetar e construir; o termo é usado para designar processo e produto.

A arquitetura de software é uma representação da informação contida na organização fundamental da estrutura de interface entre o problema do negócio e a solução técnica que define o que é sistema em termos de conjunto de componentes computacionais. É composta de elementos de software e seus relacionamentos com o ambiente e deve satisfazer os requisitos do produto de software e facilitar o entendimento por parte do interessado. Abrange a definição e a descrição de elementos de arquitetura usados na construção dos sistemas, além de interações, padrões e restrições desses componentes.

A arquitetura dá garantia à integridade e à consistência do sistema como um todo. É centrada na ideia da redução da complexidade através da abstração e da separação de interesses. Quem trabalha com desenvolvimento de software deve saber que a única constante no processo de desenvolvimento é a mudança. A documentação da arquitetura do software facilita a comunicação entre os *stakeholders*, registra as decisões iniciais acerca do projeto de alto nível e permite o reuso do projeto de componentes e padrões entre projetos.

No contexto da arquitetura de software, normalmente a programação estruturada de sistemas de pequeno porte tem estruturas de controle. A abstração e a modularização de sistemas de médio porte têm encapsulamento e ocultação de informações, e os componentes e conectores de sistemas de grande porte têm estilos arquiteturais. Projetos simples podem ser realizados com pouca modelagem, pouco projeto, pouca especialização para desenvolver e com ferramentas e processos simples. Projetos maiores e mais complexos exigem mais modelagem e mais projeto, com ferramentas mais poderosas, processos bem definidos e alta especialização para o desenvolvimento. Além disso, questões arquiteturais abrangem organização e estrutura geral de controle, protocolos de comunicação, sincronização, alocação de funcionalidade a componentes e seleção de alternativas de projeto.

Uma boa arquitetura de software, bem definida e consistente desempenha um papel fundamental para o gerenciamento dessa complexidade. É necessária para a redução do tempo e do custo de desenvolvimento e manutenção do software, tornando assim eficiente a implementação do projeto.

A arquitetura de software é normalmente organizada em visões.

a) Visão funcional ou lógica − são os principais mecanismos de projeto, utilizados no fluxo de trabalho de análise e design. Contém e descreve as classes de design e os elementos de projeto mais importantes, sua organização em pacotes e subsistemas e a organização desses pacotes e subsistemas em camadas.

b) Visão de código − contém arquivos e diretórios.

c) Visão de desenvolvimento ou estrutural − capta a organização interna dos componentes de software, normalmente como são mantidos em um ambiente de desenvolvimento ou uma ferramenta de gerenciamento de configuração.

d) Visão de concorrência ou de processo − trata a divisão do sistema em processos e processadores.

e) Visão física ou evolutiva − descreve o mapeamento do software para o hardware e reflete a natureza distribuída do sistema.

f) Visão de ação do usuário.

7.5.2 Projeto detalhado ou de baixo nível

O projeto detalhado referencia os objetos individuais e suas interações e tem a finalidade de refinar e detalhar a descrição procedimental e das estruturas de dados dos elementos mais básicos da arquitetura do software. No projeto detalhado, são modelados os relacionamentos e são atribuídas as responsabilidades entre os objetos de cada módulo com a finalidade de executar suas funções. Também são produzidos o projeto da interface com o usuário, o projeto de banco de dados e o projeto dos algoritmos, bem como alguns diagramas de UML e são documentados o desenho de cada componente, o desenho das interfaces e o desenho das bases de dados. As necessidades de

concorrência são levantadas, o tratamento de falhas é considerado, o formato de saída é detalhado e os objetos para tabelas são mapeados.

O projeto de baixo nível deve produzir o projeto de dados, o projeto de interfaces e o projeto procedimental.

7.5.2.1 Projeto de dados

O projeto de dados tem como finalidade projetar a estrutura dos dados necessários para implementar o software através da transformação das informações obtidas durante a fase de análise.

A maneira como os dados do sistema devem ser armazenados é fundamental. E existem vários modelos para isso:

a) modelo plano/tabular − modelo baseado em matrizes simples, consiste em planilhas eletrônicas, formulários, tabelas, relatórios.

b) modelo hierárquico − modelo no qual as tabelas são ligadas em uma estrutura similar a uma árvore invertida, com os dados classificados hierarquicamente. Os únicos tipos de relacionamento possíveis são um para um e um para muitos, nunca muitos para um nem muitos para muitos; por esse motivo, muitas vezes esse modelo não pode ser usado. É composto por:

   I. registro − é um conjunto de campos com valores que são informações sobre uma entidade. Exemplos: a Cecília de Mauá que ganha R$ 800,00; uma calça tamanho P; um empregado engenheiro mensalista.

   II. relacionamento − é o relacionamento do tipo pai-filho. Exemplos: o departamento pessoal possui os empregados Silva, Souza e Santos; um bloco é uma peça de motor, que, por sua vez, é uma peça; P, M e G são tamanhos de camiseta.

c) modelo em rede − modelo derivado do modelo hierárquico, tem tabelas que são usadas simultaneamente e ligadas em uma estrutura similar a uma rede. Esse modelo permite relacionamentos dos tipos muitos para um e muitos para muitos. É composto por:

   I. registro − é um conjunto de campos com valores que são informações sobre uma entidade. Exemplos: o departamento

cujo código é 28 é o departamento financeiro; o empregado número 32 se chama Sílvio, ganha R$ 950,00 e pertence ao departamento técnico; o estudante número 8 tirou nota A.

II. relacionamento − é o relacionamento do tipo pai-filho. Exemplos: os empregados números 29 e 47 trabalham no departamento financeiro; um cliente aluga um automóvel de certa marca e certo modelo; uma empresa possui filiais que possuem funcionários.

d) modelo relacional − modelo que representa os conjuntos de dados em tabelas de valores que se relacionam entre si, podendo ser derivado do modelo de entidade e relacionamento. É o modelo mais utilizado. É composto por:

I. relação − é cada tabela de valores do banco de dados estruturada em linhas e colunas. Equivale às classes do diagrama de classe. Exemplos: Localidade; Costumers; tblProduto.

II. grau da relação − é o número de colunas de uma tabela. Exemplos: seis (id, name, address, city, state, zip); três (Codigo, Nome, DataNascimento); cinco (Id, Meios, Valor, Prazo, Nome).

III. linha − representa um registro de uma tabela, um objeto de uma classe. Exemplos: F1 é o id de um registro da tabela Filmes; José da Silva é o nome de um registro da tabela Cliente; Centro é o bairro de um registro da tabela Livro.

IV. coluna − representa um dado dos registros de uma tabela, um atributo dos objetos de uma classe. Exemplos: Data operação; Nome Artístico; Código.

V. célula − é um dado de um registro de uma tabela, o valor de um atributo dos objetos de uma classe. Exemplos: “65” é o código do 65º registro da tabela Pessoa; “João” é o nome do 10º registro da tabela Clientes; “06/05/2016” é a data entrada do 478º registro da tabela Processos.

VI. chave primária − é o dado único que identifica cada registro de uma tabela, o valor do atributo que identifica cada objeto de

uma classe. Exemplos: o código de um fornecedor; o código de um cliente; o identificador de um filme.

VII. chave estrangeira − é o dado que identifica um registro de outra tabela, o valor do atributo que identifica um objeto de outra classe. Exemplos: o código de um item de um produto; o identificador de um médico de uma consulta; o número de uma parcela de uma nota fiscal.

VIII. ligação − é equivalente ao relacionamento entre as entidades do diagrama de entidade e relacionamento. Exemplos: produtos pertencem a uma categoria; um empregado atende clientes; um pedido contém itens de pedido.

e) modelo orientado a objetos − modelo no qual as informações são armazenadas como objetos em uma estrutura similar a classes. É um modelo muito utilizado em orientação a objetos. É composto por:

I. objeto − uma entidade do mundo real. Exemplos: uma pessoa conhecida; o cachorro de alguém; a Maria.

II. classe − um conjunto de objetos similares. Exemplos: pessoas; pedidos de clientes; animais.

III. atributos − características dos objetos. Exemplos: o nome de uma pessoa; a altura de uma árvore; a profissão de um cliente.

7.5.2.2 Projeto de interfaces

O projeto de interfaces descreve como o software deve se comunicar interna e externamente como outros sistemas e com seus usuários. São estabelecidos mecanismos de interação e layout para o diálogo homem-máquina, por exemplo, janelas e relatórios. Envolve tecnologias para as interfaces gráficas por meio da prototipagem e o estudo dos usuários, como eles recebem as informações do sistema e também como interagem com ele. O projeto de interfaces deve contemplar, além dos aspectos tecnológicos, o estudo das pessoas.

O designer, profissional responsável pela prototipação da interface gráfica do sistema, deve conhecer o perfil dos usuários e as atividades que eles executam. Entre suas atribuições estão a descrição das atividades a serem realizadas por meio das funcionalidades do sistema, a definição do perfil dos

usuários, a garantia da usabilidade da interface, a construção dos protótipos e a avaliação do resultado.

Jakob Nielsen, considerado "o pai da usabilidade", elaborou uma lista com dez heurísticas para avaliar a usabilidade de sistemas:

a) visibilidade do estado do sistema − o sistema deve sempre manter os usuários informados sobre o que está acontecendo por meio de feedback apropriado e em prazo razoável.

b) compatibilidade entre sistema e o mundo real − o sistema deve falar a linguagem dos usuários com palavras, frases e conceitos familiares a eles, em vez de termos orientados ao sistema. Deve-se seguir convenções do mundo real, fazendo que a informação apareça em uma ordem lógica e natural.

c) liberdade e controle do usuário − os usuários frequentemente escolhem funções do sistema por engano e precisam de uma "saída de emergência" claramente marcada para deixar o estado indesejado sem ter de passar por um diálogo extenso. Deve-se dar suporte ao desfazer e refazer ações.

d) consistência e padrões − usuários não devem ter de adivinhar quais palavras, situações ou ações diferentes significam a mesma coisa. Deve-se seguir as convenções da plataforma.

e) prevenção de erros − melhor do que boas mensagens de erro é um design cuidadoso que em primeiro lugar previne a ocorrência de um problema. Também deve-se eliminar condições passíveis de erro ou verificar a existência delas e apresentá-las aos usuários com uma opção de confirmação antes que eles realizem uma ação.

f) reconhecimento em vez de memorização − deve-se minimizar a carga de memória do usuário deixando visíveis objetos, ações e opções. O usuário não deve precisar lembrar de informações de uma parte do diálogo para outra. Instruções de uso do sistema devem ficar visíveis ou facilmente acessíveis sempre que necessário.

g) flexibilidade e eficiência de uso − atalhos, nem sempre vistos por um usuário iniciante, podem frequentemente acelerar a interação para um usuário experiente, assim o sistema pode atender a ambos

os usuários, com ou sem familiaridade. Deve-se permitir que usuários se adaptem a ações frequentes.

h) design minimalista e estético − diálogos não devem conter informação irrelevante ou raramente necessária. Toda informação extra em um diálogo compete com a informação relevante e diminui sua visibilidade relativa.

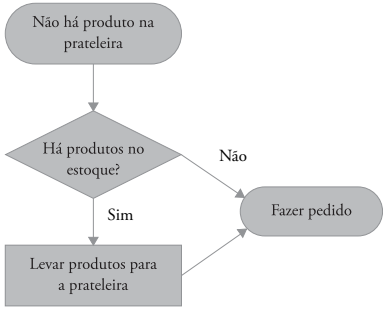
i) ajuda a usuários para reconhecer, diagnosticar e se recuperar de erros − mensagens de erro devem ser expressas em linguagem sucinta, não em códigos, indicar precisamente o problema e sugerir uma solução.

j) ajuda e documentação − mesmo que seja melhor se o sistema puder ser usado sem documentação, pode ser necessário fornecer ajuda e documentação. Qualquer informação deve ser fácil de ser pesquisada e focada na tarefa do usuário, listar passos concretos para ser cumprida e não ser muito grande.

7.5.2.3 Projeto procedimental

O projeto procedimental refina e transforma os componentes estruturais em uma descrição procedimental detalhada da arquitetura do software. São definidos os detalhes dos algoritmos que devem ser utilizados para implementar o programa. Uma boa maneira de representar um algoritmo é utilizando um fluxograma, que nada mais é do que uma representação gráfica do fluxo de controle, ou seja, do passo a passo do sistema de acordo com as tomadas de decisão. A figura 7.1 é um exemplo de um fluxograma básico de um sistema de compras.

Figura 7.1 - Fluxograma básico de site de compras



Fonte: Elaborado pela autora (2016).

Síntese (PESO 1)

A fase de projeto de um sistema é uma extensão da fase de análise e tem por objeto propor uma solução para o cliente. O projeto considera os requisitos não funcionais e as tecnologias disponíveis para o desenvolvimento do sistema e define como os requisitos devem ser implementados na estrutura interna, também chamada de arquitetura do software. Existem algumas diretrizes de qualidade e alguns princípios de projeto que devem ser seguidos para que o desenvolvimento do software tenha mais chance de sucesso.

A fase de projeto pode ser dividida em duas etapas, sendo a primeira o projeto preliminar ou arquitetural, também chamado de projeto de alto nível. Nessa etapa, é desenvolvido o projeto da arquitetura do software, cujo objetivo é criar uma estrutura a ser particionada para possibilitar a construção do sistema. A segunda etapa é chamada de projeto detalhado ou de baixo nível e pode ser subdividida em três etapas: projeto de dados, projeto de interfaces e projeto procedimental. O projeto de dados é o desenvolvimento do banco de dados por meio de um modelo de dados. Existem vários modelos, sendo os mais utilizados o relacional, que se assemelha ao diagrama de entidade e relacionamento, e o orientado a objetos, mais usado para a programação orientada a objetos. Já o projeto de interfaces é a prototipação das telas do sistema.

Para o desenho da interface humano-computador, deve-se seguir as recomendações de usabilidade. E o projeto procedimental é a estruturação de um fluxograma do sistema conforme as tomadas de decisão. Esse fluxograma facilita o entendimento de como o sistema deve se comportar de acordo com as ações do usuário e também com situações que podem ocorrer no sistema.

Atividades

1. O que é modularidade?

a) É um conjunto selecionado de conceitos e regras de forma a focar aspectos específicos de interesse em um sistema.

b) É uma estratégia baseada na construção de produtos complexos a partir da divisão e da estruturação do sistema ou do software em partes distintas chamadas de subsistemas, módulos ou componentes.

c) É um processo de elaboração no qual um programa é desenvolvido pelo refinamento sucessivo de níveis de detalhes procedimentais.

2. O que é usabilidade?

a) É uma medida da capacidade, da probabilidade ou da garantia de um sistema, um item, uma instalação, um equipamento, um dispositivo, um produto ou um serviço para desempenhar uma função.

b) É a implementação de recursos e de um atributo de qualidade dos produtos focando o usuário final.

c) É uma característica, um aspecto ou um atributo da qualidade de software ou de hardware que caracteriza um projeto de sistema, produto de software ou componente.

3. O que é projeto de dados?

a) É o projeto da estrutura dos dados necessários para implementar o software através da transformação das informações obtidas durante a fase de análise.

b) É o projeto que descreve como o software deve se comunicar interna e externamente como outros sistemas e com seus usuários.

c) É o projeto que refina e transforma os componentes estruturais em uma descrição procedimental detalhada da arquitetura do software.

4. O que é projeto procedimental?

a) É o projeto da estrutura dos dados necessários para implementar o software através da transformação das informações obtidas durante a fase de análise.

b) É o projeto que descreve como o software deve se comunicar interna e externamente como outros sistemas e com seus usuários.

c) É o projeto que refina e transforma os componentes estruturais em uma descrição procedimental detalhada da arquitetura do software.

8

Projeto Lógico e Físico

O PROJETO LÓGICO é o procedimento de projetar a arquitetura lógica do sistema, atividade que abrange um estudo do ambiente de aplicação e dos tipos de estruturas lógicas disponíveis no sistema. Hoje em dia, não existem muitas ferramentas para ajudar no procedimento de projeto lógico, o que faz com que o profissional normalmente tenha de levar em consideração seu conhecimento e experiência.

O PROJETO FÍSICO, bem como seu processo, será estudado com mais detalhes neste capítulo, por ser um assunto relevante. Aspectos do projeto físico que impactam no desempenho de uma aplicação também serão discutidos.

8.1 Projeto lógico

O projeto lógico foi criado para satisfazer a todos os requisitos especificados pelo cliente, especialmente no que se refere à alta disponibilidade, tolerância a falhas, segurança, continuidade do serviço e desempenho. Originou-se a partir das melhores tecnologias, soluções e práticas, como armazenamento centralizado, *backup* altamente confiável, infraestrutura redundante e virtualização dos recursos.

Um projeto lógico é um esquema lógico ou elaboração de um modelo interno, isto é, são recomendações sobre arquitetura e produtos para o gerenciamento e o esclarecimento sobre o motivo das diversas decisões tomadas, associando essas decisões às metas do cliente. Também é conhecido como modelagem lógica e pode ser definido como a criação da documentação do projeto, a conceituação do que o projeto de sistemas de informação deve fazer e a representação da modelagem conceitual em um modelo de banco de dados. Ele descreve como as informações são organizadas internamente, sem detalhar a estrutura de armazenamento físico, quais dados devem ser guardados e como devem estar relacionados. É criado a partir do mapeamento do modelo conceitual. Um projeto lógico, independente de implementação, é realizado para desenvolver um projeto que poderia ser implementado em diferentes plataformas, como hardware, linguagem de programação e Sistema de Gerenciamento de Banco de Dados (SGBD), devendo refinar a solução, os produtos e as integrações sistêmicas. Nessa etapa, devem ser elaborados os modelos internos do sistema. Às vezes, se a plataforma é conhecida quando se inicia o projeto, não é necessário existir a etapa de projeto lógico.

8.1.1 O modelo lógico

O modelo lógico, por exemplo um modelo relacional, orientado a objetos ou outros, leva em consideração algumas limitações, implementa recursos como adequação de padrão e nomenclatura e descreve um modelo criado no estágio anterior para o sistema. Nessa hora, o conhecimento das características do sistema a ser desenvolvido é essencial para um projeto bem sucedido, pois deve considerar o trabalho do arquiteto de software, do analista de sistemas e do administrador de banco de dados. Essa atividade leva em consideração os exemplos de modelagem elaborados no modelo conceitual. Finalmente,

o resultado de um projeto lógico é um esquema parecido com o modelo conceitual, porém mais detalhado, e não apenas definições. Nessa etapa, a finalidade é a especificação refinada dos componentes do software em nível lógico. O propósito é transformar o modelo conceitual em um modelo lógico que identifica como o sistema deve ser implementado. Além disso, deve tratar da especificação refinada dos processos externos ao computador, como:

a) captação das informações;

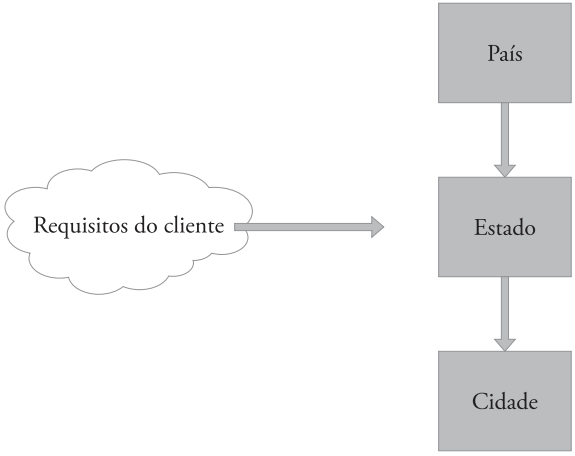
b) preparo e envio para processamento;

c) crítica e correções;

d) distribuição das saídas.

Concluindo, o projeto lógico enfatiza a estruturação de dados para o sistema. A figura 8.1 ilustra a modelagem lógica do sistema.

Figura 8.1 – Modelagem lógica.



Fonte: Elaborado pela autora (2016).

A etapa do projeto lógico, quando executado antes da etapa do projeto físico, pode vir a aumentar a possibilidade de atender às finalidades de adaptabilidade e desempenho de um cliente.

Um modelo começa apenas depois da elaboração do modelo conceitual, que dá origem a uma das abordagens possíveis da tecnologia.

O modelo lógico, criado a partir da aplicação de regras sobre um modelo conceitual que representa um nível mais direcionado aos desenvolvedores, descreve as estruturas que devem estar presentes no sistema. Porém, deve estar em conformidade com as possibilidades da abordagem.

As alterações no esquema lógico não resultam em modificações nos esquemas externos. Assim, não é necessária a reescrita dos aplicativos.

8.1.2 Etapas do projeto lógico

As subfases e atividades realizadas na etapa do projeto lógico estão listadas a seguir.

a) Modelagem de dados

   I. Detalhamento do modelo de informações empresariais, organizacionais ou institucionais.

   II. Modelo lógico normalizado – é a descrição da lógica das informações, refinando a lógica de cada informação do sistema proposto e descrevendo como essas informações devem ser criadas e disponibilizadas aos devidos *stakeholders*. Após a criação do diagrama de entidade e relacionamento, as técnicas de normalização são executadas, com a finalidade de evitar que o modelo de dados fique repetitivo. Das entidades identificadas, as já implementadas, informatizadas ou processadas devem ser marcadas para evitar que se repitam.

   III. Descrição de entidades e seus atributos – o dicionário de dados deve conter todas as entidades identificadas, seus atributos e volumes desses atributos.

b) Modelagem de processos

   I. Diagrama de fluxo de dados – o DFD finalizado na etapa de análise do sistema de informação apresenta os particionamentos sucessivos identificados, os procedimentos ou funcionalidades e respectivos fluxos de dados do *software*. Também apresenta a visão macro até os menores níveis de refinamento, de forma gráfica. Nesse estágio, os depósitos de dados do DFD devem representar entidades normalizadas.

II. Descrição dos processos – o dicionário de dados deve descrever os processos e apresentar o processamento dos fluxos de dados de entrada em saída.

III. Composição dos fluxos de dados – o dicionário de dados deve conter os fluxos de dados para que seus elementos fiquem registrados.

IV. Definição de entradas e saídas das informações – os objetivos, o conteúdo e o volume, entre outros, devem ser identificados para cada formulário de entrada, relatório, tela e outros meios. As telas e os relatórios do sistema de informação a ser desenvolvido devem ser projetados.

c) Definição de tecnologia de base para o projeto físico

As configurações a serem utilizadas para *hardware*, *software*, sistemas de telecomunicações, gestão de dados e informações devem ser descritas.

d) Elaboração de plano logístico e de contingência

Materiais, móveis, instalações elétricas, pessoal, obras civis e demais infraestruturas necessárias para o *software* devem ser descritos.

e) Controle de segurança

Os controles, manuais ou automatizados, do analista ou da empresa e do cliente a serem executados e mantidos para operação normal do software, inclusive procedimentos de reinício para paradas anormais, devem ser identificados.

f) Controle de qualidade da fase

O planejamento e execução da revisão para o controle de qualidade do *software* nesse estágio devem ser realizados, considerando os processos e os critérios de revisão da análise do produto.

g) Análise de custos, benefícios, riscos e viabilidade

h) Planejamento das fases seguintes

Fases de elaboração do projeto físico e do projeto de implantação.

i) Aprovação do projeto lógico

É um termo de compromisso que descreve a validação do acordo dos requisitos de qualidade, produtividade e efetividade do projeto de software.

O modelo lógico descreve a estrutura que deve estar no sistema conforme as possibilidades que são permitidas pela abordagem. Essa descrição tem como resultado um esquema lógico sob a perspectiva de uma das abordagens citadas, por meio da aplicação de uma técnica de modelagem com restrições de abordagem.

8.1.3 Documento do projeto lógico

O artefato da etapa do projeto lógico é um documento que pode ser denominado *Manual do Software – Parte I – Projeto Lógico*, apresentado ao usuário para leitura e aprovação. Esse documento deve conter as seguintes informações:

a) nome do *software;*

b) sigla do *software;*

c) versão;

d) nome da empresa;

e) cidade;

f) mês e ano;

g) empresa analista;

h) relação das pessoas da equipe técnica envolvidas:

I. coordenador;

II. gerente do projeto;

III. consultor;

IV. analista;

V. programador;

VI. apoio.

Esse documento pode ter a seguinte estrutura:

a) capa;

b) folha de rosto;

c) sumário;

d) introdução – objetivo e estrutura do documento;

e) modelagem de dados – diagrama de entidade e relacionamento (DER);

f) modelagem de processos:

   I. diagrama de fluxo de dados (DFD) – com todos os níveis, desde o diagrama de contexto, representando a solução proposta do problema;

   II. descrição de processos:

      × nome do processo;

      × referência (DFD);

      × descrição do processo;

g) dicionário de dados:

   I. entidades;

   II. atributos;

   III. fluxo de dados;

h) definição de entradas/saídas:

   I. formulários de entrada – cada formulário deve ter:

      × nome;

      × finalidade;

      × conteúdo;

      × frequência/volumes;

   II. relatórios – cada relatório deve ter:

- nome;
- finalidade;
- conteúdo;
- número de vias;
- destinatários;
- frequência/volumes;

III. telas – cada tela deve ter:

- nome;
- finalidade;
- conteúdo;

i) controle de segurança – é a descrição do controle, finalidade e método para implementação dos procedimentos de controle, sejam eles manuais ou automatizados;

j) termo de aprovação da fase – é o termo que o usuário aprova referente à etapa do projeto lógico.

8.2 Projeto físico

A etapa do projeto físico é o último estágio, no qual as estruturas de armazenamento internas, organizações de arquivo, índices, caminhos de acesso e parâmetros físicos do projeto para os arquivos da base de dados são definidos. Nessa parte final do projeto de sistemas, deve ser usada a linguagem de definição de dados para a sua disponibilização no dicionário de dados. O estágio de projeto físico inclui a escolha de índices, particionamentos e *clustering* de dados, bem como o projeto e a programação das transações de banco de dados referentes às especificações de alto nível.

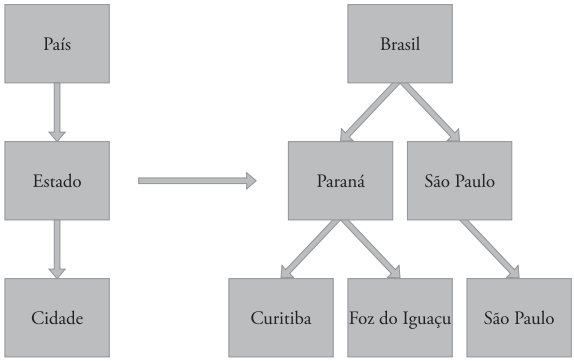
A questão a ser colocada é como fazer o projeto do modelo para mostrar os dados correspondentes a partir de um modelo conceitual. Os dados disponíveis em um modelo são valores, não podendo ser utilizados apontadores ou índices de linhas.

O método de projeto lógico diminui a quantidade de dependências que necessitam ser estudadas, facilitando assim a técnica de projeto de grandes sistemas. Como consequência disso, tem-se a combinação das fases de modelagem de dados conceitual e a junção da visão à técnica tradicional de projeto. Essas fases têm por finalidade uma representação muito precisa do mundo real. O projeto físico tem como objetivo aperfeiçoar o desempenho da melhor maneira possível.

O processo de escolher uma estrutura física para determinada estrutura lógica é chamado de projeto físico. Há várias estruturas físicas possíveis em um sistema, cada uma delas com vantagens e desvantagens. Algumas são simples de programar, outras nem tanto. Com isso, deve-se entender que nenhuma estrutura física é totalmente perfeita.

A figura 8.2 ilustra a modelagem física do sistema, tendo como base a figura 8.1, que representa a modelagem lógica.

Figura 8.2 – Modelagem física



Fonte: Elaborado pela autora (2016).

Projeto físico é a programação do sistema e inclui a criação de *scripts,* modelos físicos, estratégias de armazenamento, *backup*, tecnologias, dispositivos, informação de preços, condições físicas e dimensionamento do ambiente. É a codificação do projeto no ambiente da organização, descreve os dados do nível mais baixo ou interno e trabalha as questões de implementação do sistema, inclusive uma implementação específica para um SGBD especificado. É também a transformação do modelo lógico em um esquema

físico de acordo com as ferramentas e linguagens selecionada. Ele determina como o projeto lógico deve ser fisicamente armazenado, envolvendo a definição do espaço necessário em disco, da periodicidade dos *backups,* do volume de alteração dos dados e da quantidade e perfil dos usuários que devem ter acesso aos dados. Ademais, ele também descreve os métodos de acesso. Nesse estágio, deve ser feita uma análise com a finalidade de otimizar o desempenho pela identificação dos procedimentos mais críticos.

O arquiteto de *software* e o analista de sistemas devem dar suporte a essa etapa. O projeto físico ou implementação específica é realizado para desenvolver um projeto especificado para a plataforma selecionada. Inclui o desenvolvimento de programas, em *software* e manuais, e seus respectivos testes e controle de qualidade, bem como o *layout* físico final das entradas e das saídas do banco de dados. Essa etapa identifica detalhes técnicos da programação do sistema – por exemplo, a linguagem de programação, entre outros. O SGBD a ser usado também é tratado nessa etapa do projeto. Na construção do projeto físico, converte-se o protótipo e a especificação em código de programação. Após essa conversão, são executados alguns testes para validar a qualidade das aplicações criadas nesse estágio. Para cada item criado, deve ser executada uma bateria de testes com finalidade de encontrar não conformidades nas especificações, nas regras de negócios e funcionalidades e em relação às tecnologias usadas. A principal finalidade da etapa é assegurar a criação de todos os itens conforme estabelecido pelo analista de sistemas, e a aplicação deve ser capaz de passar pela homologação do cliente. Finalmente, o cliente deve estar com a primeira versão do sistema. Baseando-se no projeto lógico, esse estágio tem como propósito refinar os componentes do *software* em nível físico. O projeto físico é uma tarefa na qual o objetivo é construir a estruturação adequada com um bom desempenho. Para um esquema conceitual, existem muitas alternativas de projeto físico.

Geralmente, especifica-se como parte dos requisitos de desempenho do sistema limites médios sobre os parâmetros. Utilizam-se técnicas analíticas ou experimentais, que podem conter protótipo e simulação, para fazer uma estimativa dos valores sob diferentes decisões de projeto físico e determinar se elas satisfazem aos requisitos de *performance* especificados.

8.2.1 O modelo físico

O modelo físico depende do modelo de dados e do SGBD. Deve ser criado a partir do modelo lógico e descrever as estruturas físicas. O conhecimento do modelo físico de implantação das estruturas pode ser necessário.

No modelo físico, é realizada a modelagem física do modelo do sistema. São consideradas as limitações exigidas pelas tecnologias selecionadas e deve ser construído sempre tendo como base os exemplos de modelagem de dados produzidos no modelo lógico. O sucesso dessa etapa é diretamente proporcional à avaliação correta da etapa anterior. O modelo é amplamente detalhado neste estágio, influenciando assim o desempenho do sistema, mas sem impactar na sua função.

As estruturas físicas são projetadas conforme os requisitos de processamento e uso dos recursos computacionais. Esse modelo refina a análise dos métodos de acesso ao SGBD para a criação dos índices necessários para cada dado contido nos modelos conceitual e lógico.

8.2.2 Etapas do projeto físico

As subfases ou atividades executadas nessa etapa estão listadas a seguir.

a) Revisão do projeto lógico

   É o complemento, refinamento e especificação do modelo de dados, reestruturação dos dados, eliminação de redundâncias, análise das dependências funcionais, finalização do dicionário de dados, elaboração do modelo de dados, normalização dos depósitos de dados. É também a especificação do modelo de processos, identificação e reestruturação dos processos ou tarefas do sistema.

b) Projeto físico da base de dados

   É o projeto da estrutura física da base de dados, organização das entidades e seus atributos, de modo a atender, com eficácia, os aspectos de desempenho, facilidades de uso, utilização de espaço no meio físico, integridade, potencial de crescimento, flexibilidade,

privacidade e integração com outras bases de dados, atentando para as restrições do *software* a ser usado.

c) Projeto da estrutura do *software*

I. Diagrama: é a criação do diagrama estruturado do *software* a partir do projeto lógico. Deve apresentar sua estrutura hierárquica em módulos e as informações trocadas entre eles. Devem ser vistos no diagrama, além dos procedimentos lógicos, os módulos de controle e segurança necessários para o *software*.

II. Definição de programas: é a descrição de cada programa em termos de objetivo e procedimentos básicos, bem como a descrição dos módulos executados.

d) Projeto de comunicação

I. Telas: é o projeto do diagrama hierárquico e suas respectivas telas baseadas no diagrama estruturado do *software*. É a utilização de um desenho ou uma ferramenta de *software*, para caracterizar o formato dos campos utilizando máscaras com a seguinte notação:

- A – alfabético;
- 9 – numérico;
- X – alfanumérico;
- Z – número com supressão de zeros à esquerda.

Depois de avaliar a configuração do *hardware*, deve-se analisar a utilização de telas de ajuda, para guiar o usuário na utilização do sistema.

II. Formulários: é a elaboração do desenho dos formulários de entrada baseado na especificação do projeto lógico. Se o formulário de origem satisfizer aos requisitos do projeto, ele deve ser usado e anexado à documentação.

III. Relatórios: é a elaboração do desenho dos relatórios emitidos pelo *software* baseado na especificação do projeto lógico. É a utilização de um desenho ou ferramenta de *software* para

caracterizar o formato dos campos utilizando máscaras com a seguinte notação:

- A – alfabético;
- 9 – numérico;
- X – alfanumérico;
- Z – número com supressão de zeros à esquerda.

e) Definição de arquitetura e plano de segurança

É a definição dos arquivos físicos e métodos de acesso do *software* e dos procedimentos do plano de segurança das informações ou *backup*.

f) Construção do sistema de informação

É a finalização das entradas e saídas do sistema, a análise das linguagens de programação, a execução do sistema ou implementação do *software* e o desenvolvimento de programas paralelos.

g) Finalização do sistema de informação

É a elaboração de testes dos módulos e dos programas para arquivar os resultados. É a definição de fluxos e procedimentos operacionais e o complemento da documentação.

h) Definição de estratégia de projeto de implantação

É o esboço do projeto de implantação, o planejamento de treinamento e a elaboração do plano de conversão.

i) Controle de qualidade da fase

É o planejamento e realização da revisão prevista para o controle da qualidade do produto da etapa, considerando os processos e os critérios de revisão do projeto estruturado do *software*.

j) Aprovação do projeto físico

É a avaliação da qualidade, a organização de informações, a revisão de estágios anteriores, a elaboração do parecer e do termo de compromisso e a reunião e a apresentação.

8.2.3 Documento do projeto físico

O artefato da etapa do projeto físico é um documento que pode ser denominado *Manual do Software – Parte II – Projeto Físico*, que deve conter a especificação técnica completa do software. O documento deve apresentar as seguintes informações:

a) nome da empresa;

b) nome do *software;*

c) sigla do *software;*

d) versão;

e) relação das pessoas da equipe técnica envolvidas:

I. coordenador;

II. gerente do projeto;

III. consultor;

IV. analista;

V. programador;

VI. apoio.

O documento deve ter a seguinte estrutura:

a) capa;

b) sumário;

c) introdução – objetivo e estrutura do documento;

d) projeto físico da base de dados – listar os arquivos/elementos e suas características físicas:

I. nome:

- nome especificado no projeto físico;
- nome do arquivo dentro dos programas;

II. finalidade – descrição das informações guardadas nos arquivos;

III. modelo:

- nome do campo – nome criado conforme os padrões das normas de documentação de módulos;
- tipo do campo – indicação se o campo é numérico, alfanumérico ou alfabético;
- tamanho do campo – quantidade de caracteres que o campo deve aceitar;
- descrição do campo – nome dos elementos de dados de acordo com o dicionário de dados;

IV. organização – indicação do tipo de organização do arquivo; se organização indexada, deve conter uma explicação dos arquivos de índices associados e chaves de acesso:

- nome do arquivo de índice – nomes dos arquivos de índice da base de dados principal;
- nome do campo – campos que devem ser chaves de acesso à base de dados;

V. matriz arquivo/entidade;

e) projeto de comunicação:

I. telas;

II. formulários;

III. relatórios;

os modelos criados nesse estágio, no projeto de comunicação, devem ser usados na documentação do manual de uso do sistema;

f) projeto da estrutura do *software*:

IV. diagrama estruturado do software, representação do último nível de empacotamento;

V. definição de programas – a definição de cada programa deve conter:

- objetivo;

- lista dos módulos executados;
- lista dos arquivos de entrada e saída;
- lista das telas editadas;
- lista dos relatórios;
- descrição dos módulos;

VI. matriz programa/arquivo;

VII. matriz programa/módulo.

Depois que os projetos lógico e físico forem concluídos, pode-se implementar o sistema de banco de dados.

Síntese

O projeto lógico de sistemas é o procedimento de projetar a arquitetura lógica para o sistema, atividade que abrange um estudo do ambiente de aplicação e dos tipos de estruturas lógicas disponíveis. Foi criado para satisfazer a todos os requisitos especificados pelo cliente. As etapas de um projeto lógico são: modelagem de dados, modelagem de processos, definição de tecnologia de base para o projeto físico, elaboração de plano logístico e de contingência, controle de segurança, controle de qualidade da fase, análise de custos, benefícios, riscos e viabilidade, planejamento das fases seguintes e aprovação do projeto.

O projeto físico é o estágio no qual as estruturas de armazenamento internas, organizações de arquivo, índices, caminhos de acesso e parâmetros físicos do projeto para os arquivos da base de dados são definidos. Ele é a seleção das estruturas específicas, para assim atingir um bom desempenho. É a programação do sistema e inclui a criação de *scripts*, modelos físicos, estratégias de armazenamento, *backup*, tecnologias, dispositivos, informação de preços, condições físicas e dimensionamento do ambiente. É a transformação de um modelo lógico em um modelo físico. As etapas de um projeto físico são: revisão do projeto lógico, projeto físico da base de dados, projeto da estrutura do *software*, projeto de comunicação, definição de arquitetura e plano de segurança, construção do sistema de informação, finalização do

sistema de informação, definição de estratégia de projeto de implantação, controle de qualidade da fase e aprovação do projeto físico.

Atividades

1. O modelo lógico consiste em:

a) descrever um modelo criado a partir do modelo conceitual para o sistema.

b) descrever como as informações são organizadas internamente e a estrutura que pode ser processada pelo sistema, sem detalhar a estrutura física.

c) descrever um projeto que poderia ser implementado em diferentes plataformas, como *hardware*, linguagem de programação e SGBD.

2. A primeira e a última atividade do projeto lógico são, respectivamente:

a) modelagem de dados e modelagem de processos.

b) modelagem de processos e aprovação do projeto lógico.

c) modelagem de dados e aprovação do projeto lógico.

3. O que é modelo físico?

a) É a descrição de um modelo criado a partir do modelo conceitual para o sistema.

b) É a organização lógica das estruturas de armazenamento de dados e dos índices de acesso.

c) É a organização física das estruturas de armazenamento de dados e dos índices de acesso.

4. Qual o artefato da etapa do projeto lógico?

a) Manual de uso completo do *software*.

b) Documento do projeto físico.

c) Termo de aprovação do projeto físico.