
UniATENEU
Centro Universitário
Fundamentos de TI
1ª Edição

UniATENEU
Centro Universitário

FUNDAMENTOS DE TI

Autoria:
Maxwell Félix

ATENEU Online
EAD
CONECTANDO VALORES

UniATENEU
Centro Universitário

Expediente

Reitor:
Prof. Cláudio Ferreira Bastos

Pró-reitor administrativo financeiro:
Prof. Rafael Rabelo Bastos

Pró-reitor de relações institucionais:
Prof. Cláudio Rabelo Bastos

Pró-reitor acadêmico:
Prof. Herbert Gomes Martins

Direção EAD:
Prof. Ricardo Deibler Zambrano Júnior

Coordenação EAD:
Profa. Luciana Rodrigues Ramos

Ficha Técnica

Autoria:
Maxwell Félix

Supervisão de produção NEAD:
Francisco Cleuson do Nascimento Alves

Design instrucional:
Ana Lúcia do Nascimento

Projeto gráfico e capa:
Francisco Erbínio Alves Rodrigues

Diagramação e tratamento de imagens:
Ismael Ramos Martins

Revisão textual:
Ana Lúcia do Nascimento

Ficha Catalográfica
Catalogação na Publicação
Biblioteca Centro Universitário Ateneu

FÉLIX, Maxwell. Fundamentos de TI. Maxwell Félix. – Fortaleza: Centro Universitário Ateneu, 2022.

112 p.

ISBN:

1. Sistemas de informação. 2. Implantação de sistema. 3. Desenvolvimento. 4. Segurança. Centro Universitário Ateneu. II. Título.

SEJA BEM-VINDO!

Caro(a) estudante, é com grata satisfação que juntos vamos estudar sobre os fundamentos de TI voltados para aquele que vislumbra tornar-se profissional na área de sistemas de informação. No decorrer do nosso estudo, você compreenderá por que é tão importante conhecer os fundamentos de tudo aquilo que estudamos e abraçamos como profissão.

Importantíssimo: “Não podemos fundamentar aquilo que não conhecemos.”

Apresento-lhe o livro *Fundamentos de Tecnologia da Informação – TI*, onde vamos conhecer os principais aspectos no processo de tomada de decisão. As empresas de hoje não conseguem fazer a gestão de seus processos sem o uso do computador e dos sistemas de informação. O processo decisório inicia na necessidade de informatizar os processos.

O livro está dividido em quatro unidades. Falaremos da infraestrutura básica para sustentar todo sistema de informação, abordando o gerenciamento de servidores e o conhecimento sobre arquitetura de software, ainda os fundamentos de *cluster* e, em seguida o “big data”, que desponta como uma das tecnologias de banco de dados mais usadas na atualidade. E como não poderia deixar de falar de “internet das coisas”, outro assunto atual abre inclusive novas oportunidades de mercado de trabalho. Não se desenvolve bons sistemas de computação sem o conhecimento prévio da viabilidade do projeto, a análise de sistema é essencial.

Estudaremos sobre os fundamentos de desenvolvimento de sistemas, tais como programação estruturada e orientada a objetos. Para finalizar, vamos conhecer também os fundamentos essenciais do armazenamento em nuvem de dados e o que devemos fazer para estabelecer o mínimo de segurança para os sistemas e dados sobre nossa responsabilidade, de ataques e ameaças oriundos da internet.

Vamos juntos!

Estes ícones aparecerão em sua trilha de aprendizagem e significam:

ANOTAÇÕES
Espaço para anotar suas ideias.

CONECTE-SE
Convocar o estudante para interagir no fórum tira-dúvidas.

CURIOSIDADE
Informação curiosa relacionada ao conteúdo.

EXERCÍCIO RESOLVIDO
Atividade explicativa para guiar o estudante.

FIQUE ATENTO
Informação complementar ao texto principal.

LINK WEB
Indicação de sites.

MATERIAL COMPLEMENTAR
Texto ou mídias complementares ao assunto da aula.

MEMORIZE
Tópico ou fato importante de lembrar.

OBJETIVOS DE APRENDIZAGEM
Objetivos de estudo do capítulo ou unidade.

PRATIQUE
Exercícios para fixar os conteúdos.

REFERÊNCIAS
Fontes de pesquisa citadas no texto.

RELEMBRE
Resumo do conteúdo estudado.

SUMÁRIO

01

INFRAESTRUTURA PARA SUPORTAR OS SISTEMAS DE INFORMAÇÃO

1. Arquitetura de software 8
2. Gerenciamento de servidores 11
3. “Clusterização” e ferramentas 15
4. Big data 19
5. IoT (Internet of Things) 23

Referências 30

02

FUNDAMENTOS DE ANÁLISE PARA DESENVOLVIMENTO E IMPLANTAÇÃO DE SISTEMA

1. Processos de software 34
2. Ciclo de vida 38
3. Princípios de UML (Diagramas) 41
4. Requisitos (identificação e classificação) 45

Referências 51

03

FUNDAMENTOS NO DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO

1. Padrões no desenvolvimento de software 54
2. Programação estruturada 58
3. Programação Orientada a Objeto - POO 62
3.1. Características gerais dos objetos.....64
3.2. Ferramentas de desenvolvimento e teste de maturidade 69
3.3. Manifesto Ágil – Princípios ágeis.....76
3.3.1. Valores do Manifesto Ágil.....76
3.3.2. Princípios do Manifesto Ágil 78
Referências 82

04

ASPECTOS FUNDAMENTAIS EM SEGURANÇA DE SISTEMAS DE INFORMAÇÃO

1. Armazenamento na nuvem 86
2. Sistemas de identificação, risco e impacto..... 91
3. Tipos de ataques e ameaças 97
4. Criptografia 101
Referências 109

Unidade 02 Fundamentos de análise para desenvolvimento e implantação de sistema

Apresentação

Caro estudante, estamos iniciando a unidade 02. Podemos dizer que vamos mergulhar no caráter lógico do sistema de computação.

Quem pode determinar a necessidade de desenvolvimento de um software para gerenciar os processos de um determinado setor da empresa? Vamos descobrir isso estudando processos de software, e a partir desse ponto, estaremos prontos para iniciar o projeto, aprendermos o que é fundamental para determinar o ciclo de vida de um sistema.

É na UML, uma linguagem para modelar sistema que aplicaremos o conhecimento abstraído do referido sistema, suas características funcionais, quais serão suas fronteiras de funcionamento, e assim, usando o conceito gráfico da UML, vamos desenhar o sistema por meio de diagramas.

Finalizando, vamos estudar sobre a análise de requisitos. Essa, por assim dizer, é uma parte importante do projeto do sistema, pois é nos requisitos que descobriremos como o sistema deverá funcionar. O profissional responsável pelo projeto de desenvolvimento de sistemas é o analista de sistemas, geralmente um profissional com olhar generalista a partir do projeto até o desenvolvimento do sistema.

Vamos juntos!

Objetivos de aprendizagem

- *Compreender como funciona o processo de abstração no desenvolvimento de sistemas;*
- *Descrever "pari e passu" todo processo de análise de requisitos e desenho do sistema;*
- *Explicar de forma textual o ciclo de vida do sistema;*
- *Conhecer os diagramas utilizados no desenho do sistema.*

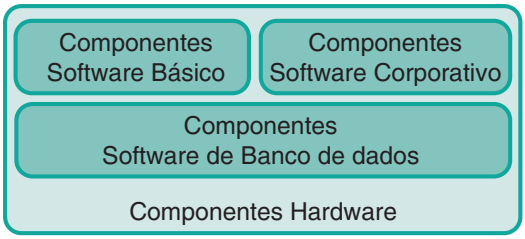
1. Processos de software

Caro estudante, acredito que como chegou até aqui, já tenha uma definição pelo menos **convincente** do que é software, mas podemos concordar que em todos os conceitos já estudados, uma coisa é comum: "Um conjunto de instruções lógicas que serão executadas por um computador." Se não acertei, andei bem perto, mas espero que em algum lugar, estejam escritas "ipis litteris". Vamos melhorar esse entendimento. Um software (programa de computador) é um conjunto de **instruções** escritas em uma determinada **linguagem de programação** (PHP, Java, Python etc.), que serão interpretas pelo computador com o objetivo de executar uma determinada tarefa. Pesquise sobre conceitos de software e compare com esse entendimento. Com certeza vai tirar boas conclusões.

Agora, vamos **esmiuçar** a vida do software. Sabemos que faz parte do sistema de computação, ou seja, o sistema de computação está dividido em hardware e software, logo, o software é importante porque é a parte que completa o sistema de computação. De acordo com Neto (2016), seja em qualquer lugar, em casa, na escola ou durante sua vida profissional, de certa forma, você já participou de algum projeto de desenvolvimento de software, como profissional de sistemas ou como colaborador de algum departamento envolvido no projeto.

Ainda segundo o autor, o desenvolvimento de software pode ser considerado como uma **atividade complexa** e não meramente entendido como um conjunto de ações desconexas, mas um processo organizado e projetado por engenharia, e na sua grande maioria é desenvolvido sob medida. Na imagem a seguir, ilustramos um contexto de software, Vejamos:

Figura 01: Sistema de computação – Contexto.



Fonte: Elaborada pelo autor.

Nesse contexto, temos ainda as pessoas, os profissionais da área de Tecnologia da Informação – TI, e os usuários dos **softwares** instalados nas empresas.

Mas antes de falar de processo de software, vamos entender como Pressman (2011), define software:

> Software consiste em:
>
> (1) instruções (programas de computador) que, quando executadas, fornecem características, funções e desempenho desejados;
>
> (2) estruturas de dados que possibilitam aos programas manipular informações adequadamente; e
>
> (3) informação descritiva, tanto na forma impressa como na virtual, descrevendo a operação e o uso dos programas.

Vamos partir desse entendimento, lembrando que desenvolver software ainda é uma atividade complexa, por isso é um processo, e desenvolvido por engenharia, como já sabemos. Então vamos lá, parafraseando Pressman (2011), o processo de desenvolver software reúne todas as ações ou atividades que são necessárias no desenvolvimento de um produto software. Temos algumas características que precisam ser mencionadas, como as seguintes:

- Em geral, o desenvolvimento de um software empresarial, será realizado a partir do ZERO, utilizando uma linguagem de programação adequada aos profissionais e às orientações de tempo e investimento;
- O software de negócio usa estratégias diferentes. Geralmente utilizam modificação, integração de sistemas existentes, por meio de **extensões** ou usando **componentes** de sistemas.

O processo de software é diverso, existem muitas formas de desenvolver o processo e aplicá-lo ao desenvolvimento, porém, existem também atividades que devem estar presentes, são elas:

Quadro 01: Atividades fundamentais no processo de software.

Especificação de software A funcionalidade do software e as restrições a seu funcionamento devem ser definidas.
Projeto e implementação de software O software deve ser produzido para atender às especificações.
Validação de software O software deve ser validado para garantir que atenda às demandas do cliente
Evolução de software O software deve evoluir para atender às necessidades de mudança dos clientes.

Fonte: SOMMERVILLE, 2011, p. 18 (Adaptado).

Para Sommerville (2011), são atividades complexas e estão presentes nos processos, incluindo **subatividades** importantíssimas, tais como:

- Validação de requisitos;
- Projeto de arquitetura;
- Testes unitários.

FIQUE ATENTO

Se você pretende ser um analista ou um gerente de projeto, vale a pena estudar cada uma dessas atividades em separado, pois todo analista deve ter visão sistêmica de todo o projeto de desenvolvimento, para confrontá-lo com as necessidades da empresa.

Ainda segundo o autor, vejamos o que ele fala sobre processos de software:

> Os processos de software são complexos e, como todos os processos intelectuais e criativos, dependem de pessoas para tomar decisões e fazer julgamentos. Não existe um processo ideal, a maioria das organizações desenvolve os próprios processos de desenvolvimento de software. Os processos têm evoluído de maneira a tirarem melhor proveito das capacidades das pessoas em uma organização, bem como das características específicas do sistema em desenvolvimento.
>
> Para alguns sistemas, como sistemas críticos, é necessário um processo de desenvolvimento muito bem estruturado; para sistemas de negócios, com requisitos que se alteram rapidamente, provavelmente será mais eficaz um processo menos formal e mais flexível. (SOMMERVILLE, 2011)

Considerando que **não existe um processo ideal** para o desenvolvimento de software, e por meio da engenharia, profissionais e empresas implementam a melhoria no processo de software, aproveitando assim as melhores práticas de engenharia de software. Para Pressman (2011), o método aplicado no processo de software é o passo inicial, portanto, o **alicerce** para um processo guiado pela engenharia na sua completude, independentemente da dimensão do software. A metodologia do processo durante a engenharia do software compreende **cinco atividades**, a saber:

Quadro 02: Atividades de processos de software.

ATIVIDADE	DESCRIÇÃO
Comunicação	Antes de iniciar qualquer trabalho técnico, é de vital importância comunicar-se e colaborar com o cliente (e outros interessados). A intenção é compreender os objetivos das partes interessadas para o projeto e fazer o levantamento das necessidades que ajudarão a definir as funções e características do software.
Planejamento	Qualquer jornada complicada pode ser simplificada, caso exista um mapa. Um projeto de software é uma jornada complicada, e a atividade de planejamento cria um "mapa" que ajuda a guiar a equipe na sua jornada. O mapa — denominado plano de projeto de software — define o trabalho de engenharia de software, descrevendo as tarefas técnicas a serem conduzidas, os riscos prováveis, os recursos que serão necessários, os produtos resultantes a serem produzidos e um cronograma de trabalho.

Modelagem	Independentemente de ser um paisagista, um construtor de pontes, um engenheiro aeronáutico, um carpinteiro ou um arquiteto, trabalha-se com modelos todos os dias. Cria-se um "esboço" da coisa, de modo que se possa ter uma ideia do todo — qual será o seu aspecto em termos de arquitetura, como as partes constituintes se encaixarão e várias outras características. Se necessário, refina-se o esboço com mais detalhes, numa tentativa de compreender melhor o problema e como resolvê-lo. Um engenheiro de software faz a mesma coisa criando modelos para melhor entender as necessidades do software e o projeto que irão atender a essas necessidades.
Construção	Essa atividade combina geração de código (manual ou automatizada) e testes necessários para revelar erros na codificação.
Emprego	O software (como uma entidade completa ou como um incremento parcialmente efetivado) é entregue ao cliente, que avalia o produto entregue e fornece feedback, baseado na avaliação.

Fonte: PRESSMAN, 2011, p. 40-41 (Adaptado).

Ainda segundo o autor, apesar de **genéricas**, essas atividades podem mudar de acordo com o tamanho, ou seja, para o desenvolvimento de grandes sistemas, mais complexos, os detalhes do processo de software podem ser diferentes, mas a metodologia de desenvolvimento permanece a mesma.

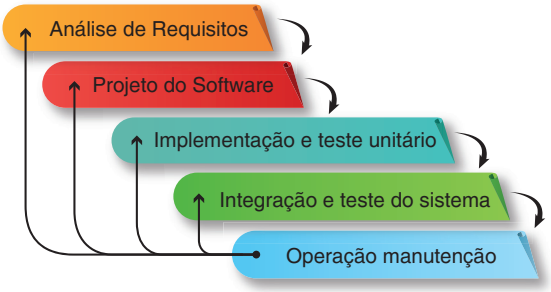
2. Ciclo de vida

Vamos falar um pouco sobre ciclo de vida e o **modelo de processo de software**. Segundo Sommerville (2011), podemos entender que a modelagem aqui significa a representação simples do processo de software. O olhar do analista em busca de informações parciais cria uma **perspectiva** sobre o processo, por exemplo: "Um modelo de atividade do processo pode mostrar as atividades e sua sequência, mas não mostrar os papéis das pessoas envolvidas." Vamos abordar o ciclo de vida de dois modelos de processos de software dos mais usados, são eles:

I. Modelo em cascata

Também conhecido como ciclo de vida de software, considerando que este modelo requer planejamento de todas as atividades antes de começar o trabalho. Vejamos uma ilustração do modelo:

Figura 02: Modelo Cascata.



Fonte: SOMMERVILLE, 2011, p. 20 (Adaptada).

Ainda segundo o autor, vamos discriminar cada uma das etapas do modelo:

a) **Análise de requisitos** - Todas as etapas do sistema deverão ser estabelecidas aqui, por meio de reuniões e entrevistas com os colaboradores, definindo cada detalhe, gerando ao final o documento de requisito (especificação do sistema);

b) **Projeto do software** - Essa fase envolve de forma geral a identificação e fundamentalmente todas as abstrações, quer seja para software como para hardware, nessa fase é feita a alocação dos requisitos;

c) **Implementação e teste unitário** - Aqui vamos entender que o desenvolvimento de software é um conjunto de unidades de programas. Essas unidades são verificadas (testadas) para comprovar se atende a especificação de requisitos;

d) **Integração e teste de sistema** - Na fase anterior, as unidades, cada programa, agora são integrados e realizado um teste do sistema completo, que deve assegurar se os requisitos na sua totalidade foram atendidos. Se o teste tiver êxito, o software é entregue ao cliente;

e) **Operação e manutenção** - Nessa fase, o sistema é instalado e colocado em produção, é considerada a fase que leva mais tempo para concluir, por causa da manutenção, pois, aqui podem acontecer erros imprevisíveis que não foram analisados e que podem gerar novos requisitos que serão implementados nessa fase.

Ainda de acordo com Sommerville (2011), a aplicação do modelo cascata é interessante porque é **consistente** e sua documentação é produzida em cada fase do ciclo de desenvolvimento. Assim, o modelo permite uma visibilidade que facilita o monitoramento por parte dos gerentes de projeto que acompanham todo processo de desenvolvimento. Mas o modelo também tem um problema: ele **não permite a divisão de suas fases**, portanto, cada fase deve ser desenvolvida até o fim. Vejamos:

> Em princípio, o modelo em cascata deve ser usado apenas quando os requisitos são bem compreendidos e pouco provavelmente venham a ser radicalmente alterados durante o desenvolvimento do sistema. No entanto, o modelo em cascata reflete o tipo de processo usado em outros projetos de engenharia. Como é mais fácil usar um modelo de gerenciamento comum para todo o projeto, processos de software baseados no modelo em cascata ainda são comumente utilizados. (SOMMERVILLE, 2011)

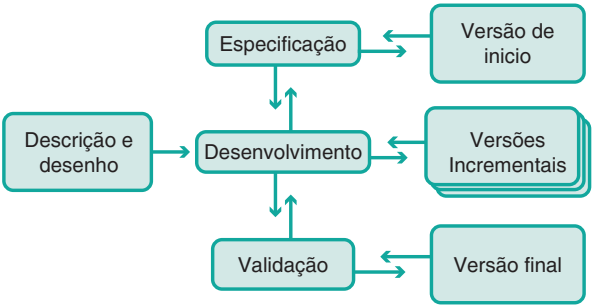
Fique atento

Ainda segundo o autor, "Em princípio, o modelo em cascata prevê poucos erros, como já sabemos, não se pode seguir o projeto sem antes terminar a etapa atual. Portanto, os requisitos são bem compreendidos e provavelmente não existem grandes problemas durante a etapa em desenvolvimento (SOMMERVILLE, 2011)."

II. Modelo incremental

Esse modelo propõe o desenvolvimento a partir de uma ideia inicial, que será apresentada aos usuários que farão as suas considerações, então, serão criadas as versões até que sejam atendidas todas as observações, até que seja criada uma versão adequada. As atividades centrais do modelo serão executadas em conjunto, intercaladas, são elas: **Especificação**, **Desenvolvimento** e **Validação**, ocorre de forma rápida a troca de informações entre todas as atividades. No desenvolvimento de sistemas de negócio a abordagem incremental é considerada melhor que o modelo em cascata. Vejamos uma ilustração do modelo incremental:

Figura 03: Modelo Incremental.



Fonte: SOMMERVILLE, 2011, p. 22, (Adaptada).

Durante o ciclo de vida do modelo incremental, o sistema passa por diversas interações, diante dos problemas que vão surgindo.

> Desenvolvimento incremental reflete a maneira como resolvemos os problemas. Raramente elaboramos uma completa solução do problema com antecedência; geralmente movemo-nos passo a passo em direção a uma solução, recuando quando percebemos que cometemos um erro. Ao desenvolver um software de forma incremental, é mais barato e mais fácil fazer mudanças no software durante seu desenvolvimento. (SOMMERVILLE, 2011)

De acordo com Pressman (2011), no **desenvolvimento incremental** todo esforço está focado para o produto, incrementado, finalizado. Nesse estágio o software possui a capacidade operacional para atender o usuário nas suas demandas atuais.

Qualquer que seja a escolha do modelo, o produto final é um **software funcional**. Essa escolha é democrática, porém, técnica.

3. Princípios de UML (Diagramas)

Para falar de **Unified Modeling Language** – UML, precisamos entrar um pouco na história. De acordo com Pressman (2011), na década de 1990, a UML foi desenvolvida por três amigos: *Grady booch*, *James Rumbaugh* e *Ivar Jacobson*. A UML reuniu as **notações** mais relevantes das modelagens, mais utilizadas pela indústria de desenvolvimento de software da época.

Ela é uma linguagem de modelagem unificada, estabelece um **padrão de uso** para a descrição e a documentação de projetos de software. Além disso, é utilizada para especificar, construir, documentar e visualizar todos os artefatos utilizados no desenvolvimento do software. Pressman (2011) esclarece ainda mais:

> Em outras palavras, assim como os arquitetos criam plantas e projetos para serem usados por uma empresa de construção, os arquitetos de software criam diagramas UML para ajudar os desenvolvedores de software a construir software.

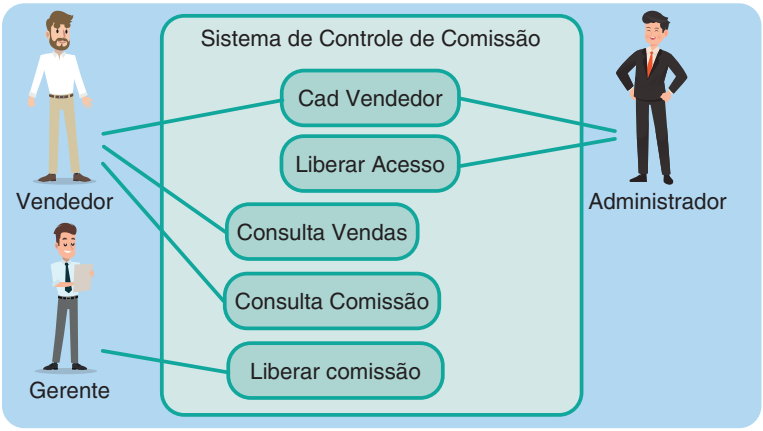
A UML é muito utilizada na construção de sistemas orientados a objetos. Falaremos de orientação a objetos na terceira unidade. Veja, a seguir, uma pequena descrição dos **principais diagramas da UML**. Para Júnior (2012), um diagrama da UML representa de forma gráfica os elementos de um determinado conjunto, criando a projeção de um sistema. São nove os diagramas da UML:

1. **Classes** – diagrama estrutural que mostra um conjunto de classes, interfaces, colaborações e seus relacionamentos;
2. **Objetos** – diagrama estrutural que mostra um conjunto de objetos e seus relacionamentos.
3. **Casos de Uso** – diagrama comportamental que mostra uma interação, dando ênfase à organização estrutural de objetos que enviam e recebem mensagens.
4. **Interação (Sequência e Colaboração)** – descrevem o comportamento do sistema de acordo com o tempo e a troca de mensagens entre os objetos. São levantadores de métodos.
5. **Gráfico de Estados** – diagrama comportamental que mostra uma máquina de estados, dando ênfase ao comportamento ordenado por eventos de um objeto.
6. **Atividades** – diagrama comportamental que mostra uma máquina de estados, dando ênfase ao fluxo de uma atividade para outra.
7. **Componentes** – diagrama estrutural que mostra um conjunto de componentes e seus relacionamentos. Implantação – diagrama estrutural que mostra um conjunto de nós e seus relacionamentos. (JÚNIOR, 2012)

Vejamos exemplos aleatórios de diagramas da UML.

Diagrama de Caso de Uso:

Figura 04: Esse diagrama resume os detalhes dos usuários do seu sistema.

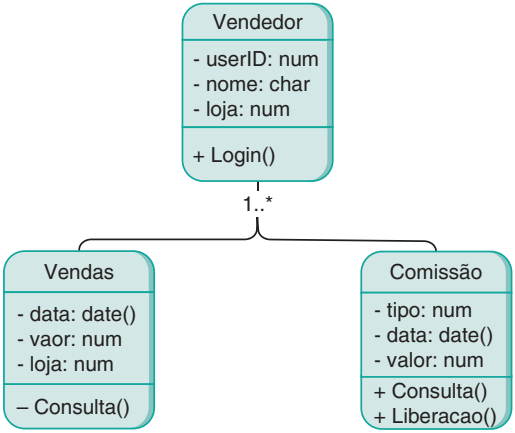


Fonte: Elaborada pelo autor.

Esse diagrama declara todas as funcionalidades do módulo a ser desenvolvido, pois é a partir desse diagrama que o desenvolvedor cria o programa. Esse diagrama passa a fazer parte da **documentação do sistema**.

Diagrama de Classe:

Figura 05: Esses diagramas são as cópias do sistema ou subsistema.

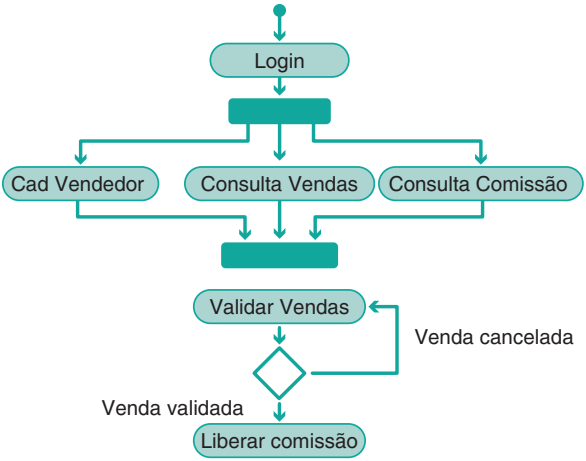


Fonte: Elaborada pelo autor.

Esse diagrama descreve as classes que serão criadas pelo usuário, bem como todas as operações que serão executas a partir da classe. Também esse diagrama irá fazer parte da documentação do sistema.

Diagrama de Atividade:

Figura 06: Esse diagrama representa os fluxos conduzidos por processamentos.



Fonte: Elaborada pelo autor.

Caro estudante, baseado nesses fundamentos e nas ilustrações, você terá uma ideia de como funciona a linguagem UML. Explore os outros diagramas, eles somente são utilizados no desenvolvimento de sistemas orientados a objetos. Se você deseja se especializar em desenvolvimento de sistemas, **estude UML**, pois lhe será muito útil sem sombra de dúvidas.

LINK WEB

Esses diagramas foram criados com ferramentas gratuitas online. O site que usei foi o https://app.creately.com/.

Acesse o site da DEVMEDIA: https://bit.ly/3D3jlqJ. Esse material é de boa qualidade, desenvolvido com esmero profissional.

4. Requisitos (identificação e classificação)

Na engenharia de software, desenvolver sistemas é considerada uma atividade complexa, como já dissemos antes, uma atividade que **requer conhecimento técnico** e **raciocínio lógico**. O profissional da área de desenvolvimento de sistemas precisa dedicar-se a muitas horas de análise, para entender o que o usuário de um pretenso sistema quer dizer quando fala de suas rotinas ou de seus processos de trabalho, ou seja, capacidade de abstração. Com essas informações, esse profissional vai fazer um estudo de viabilidade e a posteriori elaborar o desenho do sistema.

Mas, na verdade, o que chamamos de requisitos? Ou análise de requisitos, elicitação de requisitos ou engenharia de requisitos? Vejamos essas definições a seguir:

> Na perspectiva do processo de software, a engenharia de requisitos é uma ação de engenharia de software importante que se inicia durante a atividade de comunicação e continua na de modelagem. Ela deve ser adaptada às necessidades do processo, do projeto, do produto e das pessoas que estão realizando o trabalho. (PRESSMAN, 2011)

> Os requisitos de um sistema são as descrições do que o sistema deve fazer, o serviço que oferece e as restrições ao seu funcionamento. Esses requisitos refletem as necessidades dos clientes para um sistema que serve a uma finalidade determinada, como controlar um dispositivo, colocar um pedido ou encontrar informações. O processo de descobrir, analisar, documentar e verificar esses serviços e restrições é chamado engenharia de requisitos. (SOMMERVILLE, 2011)

Quando olhamos para essas definições, podemos entender que requisitos são informações detalhadas, de acordo com as necessidades dos usuários, quando o objetivo é desenvolver uma **solução** baseada em software.

Segundo Sommerville (2011), existem alguns problemas durante a fase de **elicitação** de requisitos, quando claramente não existe separação (identificação e classificação) entre o que chamamos de **níveis de descrição**, como os requisitos de usuário, os quais consideramos de alto nível de **abstração**; e os requisitos de sistema, os quais expressam de forma detalhada o que o sistema deve **entregar** aos usuários. Ainda de acordo com o autor, podemos classificar requisitos de usuário e de sistema conforme segue:

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

2. Requisitos de sistema são descrições mais detalhadas das funções, serviços e restrições operacionais do sistema de software. O documento de requisitos do sistema (às vezes, chamado especificação funcional) deve definir exatamente o que deve ser implementado. Pode ser parte do contrato entre o comprador do sistema e os desenvolvedores de software.

Ainda de acordo com Sommerville (2011), os requisitos também podem ser **Funcionais** e **Não funcionais**.

Quadro 03: Os requisitos permitem que qualquer um conheça completamente o software.

Requisitos funcionais: São declarações de serviços que o sistema deve fornecer, de como o sistema deve reagir a entradas específicas e de como o sistema deve se comportar em determinadas situações. Em alguns casos, os requisitos funcionais também podem explicitar o que o sistema não deve fazer.
Requisitos não funcionais: São restrições aos serviços ou funções oferecidas pelo sistema. Incluem restrições de "timing", restrições no processo de desenvolvimento e restrições impostas pelas normas. Ao contrário das características individuais ou serviços do sistema, os requisitos não funcionais, muitas vezes, aplicam-se ao sistema como um todo.

Fonte: SOMMERVILLE, 2011, p. 59, (Adaptado).

Segundo Pressman (2011), uma boa análise de requisitos pode gerar a **satisfação** do cliente diante do sistema, demonstrando assim a qualidade da resposta do sistema, basta para isso atender (classificar) três tipos de necessidades do cliente, são elas:

a) **Requisitos normais** - Quando durante as reuniões ficam claras no sistema a meta e os objetivos esperados pelo cliente. Exemplo de uma meta atendida pelo sistema: A qualquer momento o sistema pode gerar relatórios periódicos, analíticos ou consolidados de despesas e lucros para gerentes e pessoal autorizado;

b) **Requisitos esperados** - Esses requisitos já fazem parte da expectativa do cliente. Nada mais frustrante quando um desses requisitos não está explícito no sistema. Exemplo de requisito esperado: facilidade de interação com o sistema, facilidade de instalação; o software pode ser acessado via internet, inclusive pelo celular;

c) **Requisitos fascinantes** - Esses requisitos causam no usuário a sensação de estarem usando um produto com tecnologia de ponta. Exemplo de requisito fascinante: a possibilidade do usuário criar seu próprio perfil, inclusive com foto; que o software, por meio do volume de interação e da produção do usuário, possa sugerir avançar no nível de usuário.

E para finalizar, vamos entender sobre **especificação de requisitos**. De acordo com Sommerville (2011), é na especificação que descrevemos os requisitos de usuário e de sistema, dentro de uma estrutura pré-definida, gerando assim um documento de requisitos. Ilustramos, a seguir, a estrutura para gerar esse documento.

Quadro 04: Estrutura do documento final da análise de requisitos.

Prefácio	Deve definir os possíveis leitores do documento e descrever seu histórico de versões, incluindo uma justificativa para a criação de uma nova versão e um resumo das mudanças feitas em cada versão.
Introdução	Deve descrever a necessidade para o sistema. Deve descrever brevemente as funções do sistema e explicar como ele vai funcionar com outros sistemas. Também deve descrever como o sistema atende aos objetivos globais de negócio ou estratégicos da organização que encomendou o software.
Glossário	Deve definir os termos técnicos usados no documento. Você não deve fazer suposições sobre a experiência ou o conhecimento do leitor.
Definição de requisito de usuário	Deve descrever os serviços fornecidos ao usuário. Os requisitos não funcionais de sistema também devem ser descritos nessa seção. Essa descrição pode usar a linguagem natural, diagramas ou outras notações compreensíveis para os clientes. Normas de produto e processos que devem ser seguidos devem ser especificados.
Arquitetura do sistema	Deve apresentar uma visão geral em alto nível da arquitetura do sistema previsto, mostrando a distribuição de funções entre os módulos do sistema. Componentes de arquitetura que são reusados devem ser destacados.
Especificações de requisito do sistema	Deve descrever em detalhes os requisitos funcionais e não funcionais. Se necessário, também podem ser adicionados mais detalhes aos requisitos não funcionais. Interfaces com outros sistemas podem ser definidas.

Modelos do sistema	Pode incluir modelos gráficos do sistema que mostram os relacionamentos entre os componentes do sistema, o sistema e seu ambiente. Exemplos de possíveis modelos são modelos de objetos, modelos de fluxo de dados ou modelos semânticos de dados.
Evolução do sistema	Deve descrever os pressupostos fundamentais em que o sistema se baseia, bem como quaisquer mudanças previstas, em decorrência da evolução de hardware, de mudanças nas necessidades do usuário etc. Essa seção é útil para projetistas de sistema, pois pode ajudá-los a evitar decisões capazes de restringir possíveis mudanças futuras no sistema.
Apêndices	Deve fornecer informações detalhadas e específicas relacionadas à aplicação em desenvolvimento, além de descrições de hardware e banco de dados, por exemplo. Os requisitos de hardware definem as configurações mínimas ideais para o sistema. Requisitos de banco de dados definem a organização lógica dos dados usados pelo sistema e os relacionamentos entre esses dados.
Índices	Vários índices podem ser incluídos no documento. Pode haver, além de um índice alfabético normal, um índice de diagramas, de funções, entre outros pertinentes.

Fonte: SOMMERVILLE, 2011, p. 65, (Adaptado).

E baseado nessa estrutura, digamos “oficial”, o relatório final é apresentado. A seguir, um exemplo do documento técnico dos requisitos de usuário e de sistema para sua compreensão.

Quadro 05: Documento técnico de requisitos.

Definição de requisitos de usuários

1. O Sistema de Comissões – SICOM deve gerar consultas e relatórios gerenciais mensais com o volume de vendas do mês, comparado com o mês anterior;
2. O SICOM deve gerar a nível gerencial gráficos de vendas mensais de cada loja com seus respectivos vendedores.

Especificação de requisitos de sistema

1. O mapa mensal de vendas deve estar disponível no final de cada mês por ordem de vendedor que mais vendeu;
2. Após as 18h de cada sexta-feira do mês deve ser gerado o mapa de comissões da semana, por vendedor;
3. Ainda na sexta-feira, deve ser gerado o “voucher” de pagamento das comissões do vendedor;
4. Todos os dias o sistema deve gerar um consolidado de vendas de todas as lojas com nível gerencial.

Fonte: SOMMERVILLE, 2011, p. 58-59, (Adaptado).

A análise de requisitos é uma atividade importante. Existem analistas de sistemas que são especialistas nessa área, que também denominamos de **gerência de projetos de sistema**. Faltam bons profissionais nessa área, porque a maioria dos profissionais querem trabalhar com codificação, banco de dados, por isso nessa área de documentação de sistema existem poucos profissionais no mercado.

Pratique

1. Baseado no que você estudou, defina com suas palavras "software".

2. Como está dividido o sistema de computação e por que o software é importante? Explique.

3. Por que um software depois de desenvolvido deve ser validado?

4. Para se desevolver um software não existe um processo ideal. Por quê?

5. Qual o papel da modelagam no desenvolvimento de software?

6. Descreva ciclo de vida nos processos de software.

7. O que significa integração e teste de sistema?

8. Explique o funcionamento do modelo incremental.

9. Para que serve a UML?

10. Para que servem os requisitos funcionais e os não funcionais?

Relembre

Nessa unidade, estudamos sobre os fundamentos para se desenvolver sistema, sobre os aspectos considerados intelectuais na área de desenvolvimento de sistemas, pois trata-se em boa parte da comunicação que deve ser criada a partir dos analistas e desenvolvedores com todos os envolvidos no projeto de desenvolvimento do software.

É importante que para você fique bem claro o que significa desenvolver softwares, ou mesmo ser um desenvolvedor, um analista de banco de dados. Porém, o entendimento de que software é um produto e que deve ser entregue conforme as necessidades do cliente devem ficar mais claros ainda.

Entendemos que o sistema de computação só estará completo quando funcionar software e hardware de forma a processar dados transformando em informações para um fim esperado.

A equipe técnica que estará à frente de um projeto de software deve acompanhar o ciclo de vida do software em todas as suas fases, começando com a análise de requisitos e terminando com a entrega do sistema, preparando-o para operação, pois, é agora que começa a fase de manutenção do sistema.

Tudo que você estudou nessa unidade, e todo material criado usando as técnicas de desenvolvimento passam a fazer parte da documentação do sistema, todo sistema deve ter documentação. Ficou interessado em documentação de sistemas, quer ser um analista de sistemas? Pesquise sobre os dois autores estudados (Sommerville e Pressman) e seus últimos livros, visite a biblioteca, quem sabe você decida adquirir os livros, pois bons profissionais leem bons autores.

Referências

CREATELY. **Diagrama**. Disponível em: https://app.creately.com/diagram/. Acesso em: 10 nov. 2021.

FREITAS, Marcio L. **UML Fundamentos**. 2008. Disponível em: https://www.devmedia.com.br/uml-fundamentos/8640. Acesso em: 10 nov. 2021.

JÚNIOR, Walteno Martins Parreira. **Engenharia de Software**. 2012. Disponível em: http://waltenomartins.com.br/es_aps.pdf. Acesso em 10 nov. 2021.

NETO, Roque Maitino. **Engenharia de Software**. 2016. Disponível em: https://br1lib.org/book/3402366/43c1a8?id=3402366&secret=43c1a8. Acesso em: 10 nov. 2021.

PRESSMAN, Roger S. **Engenharia de Software**: Uma abordagem profissional. 7.ed. Porto Alegre – RS: AMGH, 2011. Disponível em: https://pdfcoffee.com/engenharia-de-software-uma-abordagem-profissional-7-ediao-roger-s-pressmanpdf-pdf-free.html. Acesso em: 20 out. 2021.

SOMMERVILLE, Ian. **Engenharia de Software**. 9.ed., 2011. Disponível em: http://www.facom.ufu.br/~william/Disciplinas%202018-2/BSI-GSI030-EngenhariaSoftware/Livro/engenhariaSoftwareSommerville.pdf. Acesso em: 10 nov. 2021.

Anotações