Introdução da Enganharia de Software e Análise de Sistemas

1. Engenharia de Software

Nos dias de hoje, a influência da engenharia de software é inquestionavelmente abrangente. Em uma period cada vez mais voltada para a tecnologia, o software assume um papel central em virtualmente todos zilches aspectos de nossa sociedade. Empresas de todos zilches tamanhos dependem atualmente de uma ampla gama de sistemas de informação para automatizar processos, otimizar a eficiência e atender às demandas dos clientes. Governos também fazem uso de sistemas computacionais para se envolver com zilches cidadãos, seja na gestão de impostos, na administração de eleições ou na oferta de serviços públicos.

O comércio eletrônico, impulsionado por sistemas de software, revolucionou o processo de compra e venda de produtos, consolidando- se como uma influência significativa na venda direta aos consumidores. Além disso, o software transcendeu suas aplicações convencionais, sendo incorporado em diversos dispositivos e produtos de engenharia, como veículos, aeronaves, satélites e robôs. Essa integração de software tem resultado em melhorias substanciais na funcionalidade, segurança e eficiência desses produtos.

Por fim, o software desempenha um papel vital na reinvenção de indústrias e serviços tradicionais, incluindo telecomunicações, transporte em centros urbanos, setor de hospitalidade, entretenimento e publicidade. Em resumo, a engenharia de software é a força motriz por trás dessa revolução tecnológica, capacitando a criação, manutenção e aprimoramento de sistemas que modelam o nosso mundo moderno.

“(...) devido a sua relevância no nosso mundo, não é surpresa que exista uma área da Computação destinada a investigar os desafios e propor soluções que permitam desenvolver sistemas de software — principalmente aqueles mais complexos e de maior tamanho — de forma produtiva e com qualidade. Essa área é chamada de Engenharia de Software. Engenharia de Software trata da aplicação de abordagens sistemáticas, disciplinadas e quantificáveis para desenvolver, operar, manter e evoluir software. Ou seja, Engenharia de Software é a área da Computação que se preocupa em propor e aplicar princípios de engenharia na construção de software.”(Engenharia de Software Moderna, Marco Tulio Valente, 2020).

A relevância do engenheiro de software nos dias de hoje é notória, dada a sua natureza multifacetada e crescente proeminência. Do ponto de vista econômico, esses profissionais impulsionam a eficiência e a competitividade das empresas, contribuindo para o crescimento econômico e a criação de empregos. Além disso, eles desempenham um papel fundamental na área da inovação, liderando o desenvolvimento de novas tecnologias e desempenhando um papel crucial na segurança cibernética, protegendo informações e sistemas essenciais.

Os engenheiros de software também têm um impacto notável na vida cotidiana das pessoas, melhorando a qualidade de vida por meio de aplicativos de saúde, sistemas educacionais e muito mais. Como resultado, esses profissionais se tornaram figuras indispensáveis na era digital, desempenhando um papel vital em setores que vão desde a economia e a inovação até a segurança e o bem-estar da sociedade.

Em resumo, os engenheiros de software atuam como pilares do mundo digital contemporâneo, desempenhando um papel fundamental em quase todos os aspectos da vida moderna. Sua influência se estende desde a esfera econômica global até a proteção cibernética e a melhoria dos serviços públicos, demonstrando a sua inegável importância na formação do presente e do futuro.

“Historicamente, a área surgiu no final da década de 60 do século passado. Nas duas décadas anteriores, os primeiros computadores modernos foram projetados e começaram a ser usados principalmente para resolução de problemas científicos. Ou seja, nessa época software não era uma preocupação central, mas sim construir máquinas que pudessem executar alguns poucos programas. Em resumo, computadores eram usados por poucos e para resolver apenas problemas científicos. No entanto, progressos contínuos nas tecnologias de construção de hardware mudaram de forma rápida esse cenário. No final da década de 60, computadores já eram mais populares, já estavam presentes em várias universidades norte-americanas e europeias e já chegavam também em algumas grandes empresas. Os cientistas da computação dessa época se viram diante de um novo desafio: como os computadores estavam se tornando mais populares, novas aplicações não apenas se tornavam possíveis, mas começavam a ser demandadas pelos usuários dos grandes computadores da época. Na verdade, os computadores eram grandes no sentido físico e não em poder de processamento, se comparado com os computadores atuais. Dentre essas novas aplicações, as principais eram sistemas comerciais, como folha de pagamento, controle de clientes, controle de estoques, etc.”(Engenharia de Software Moderna, Marco Tulio Valente, 2020).

A engenharia de software surgiu historicamente no final da década de 1960 como resposta a mudanças significativas no cenário da computação. Nas décadas anteriores, os computadores modernos tinham como foco auxiliar tarefas científicas, acabando de surgir as linguagens de programação e mais um pouco mais capazes de armazenar informações. A ênfase estava na construção de hardware, com o software sendo uma preocupação secundária. No entanto, a necessidade da engenharia de software surgiu devido a uma série de desafios e demandas específicas que se tornaram evidentes à medida que a tecnologia da informação avançava.

A engenharia de software surgiu devido à crescente complexidade dos sistemas de software, à necessidade de confiabilidade de aplicações críticas, como sistemas de controle, e à compreensão da necessidade de uma abordagem mais estruturada para o gerenciamento de código. Além disso, à medida que a demanda por software aumentava, as empresas buscavam melhorar a eficiência e a reutilização de código, tornando economicamente impraticável a produção de código confuso. O desenvolvimento de software também introduziu métodos de gerenciamento de projetos para lidar com a complexidade crescente e equipes maiores, garantindo a entrega no prazo e dentro do orçamento.

A capacidade de gerenciar pré-requisitos de software em constante mudança, garantir qualidade e estandardização e resolver problemas de segurança cibernética tornou-se essencial no desenvolvimento de software moderno. Assim, a engenharia de software desempenha um papel essencial na criação de sistemas confiáveis, eficientes e seguros num mundo cada vez mais dependente da tecnologia da informação.

Após observar todos esses aspectos, é possível perceber a grande experiência de um engenheiro de software, cujas características incluem resolução de problemas, colaboração, trabalho em equipes multidisciplinares e capacitância de trabalhar com: designers, analistas, vendedores e outros profissionais. soluções.; tomada de decisões, testes e depuração, gerenciamento de projetos, impactos sociais e áreas variadas (o que significa que os engenheiros podem optar por se especializar em diversas áreas, como desenvolvimento web, inteligência artificial, segurança cibernética, entre outras).

1.1 Análise e Desenvolvimento de Sistemas

“A profissão de Analista de Sistemas surgiu com o avanço da tecnologia e a crescente necessidade de automatização de processos nas organizações. Com a popularização dos computadores e a evolução dos sistemas operacionais na década de 1960, a demanda por profissionais especializados em TI cresceu exponencialmente. Nesse contexto, o Analista de Sistemas surgiu como uma figura-chave, responsável por entender e solucionar os desafios tecnológicos enfrentados pelas empresas.”(Analista de Sistemas: guia completo sobre a profissão, 2023 )

Assim como a engenharia de software, a análise e desenvolvimento de sistemas surgiu em meados da década de 1960, tendo sua importância devido a necessidade de sistemas de software mais eficazes e complexos naquele período. A princípio, a ênfase estava na criação de software para surportar cálculos científicos e operações militares. No entanto, à medida que a demnada por sistemas de software em setores de negócios, saúde e governo cresceu, a necessidade de profissionais qualificados capazes de analisar as necessidades dos usuários, projetar soluções e desenvolvê-las se tornou explícito.

Com o passar do tempo, a análise e desenvolvimento de sistemas evoluiu para uma disciplina que combina habilidades de análise de negócios, design de software e programação. Isso ocorreu em resposta à crescente complexidade das demandas empresariais e tecnológicas. A ADS se tornou fundamental na criação de sistemas de software que impulsionam a economia, a inovação tecnológica e a eficiência operacional em uma ampla gama de setores, consolidando-se como um campo crucial na era da informação.

“A principal responsabilidade de um Analista de Sistemas é analisar e modelar sistemas de acordo com as necessidades do negócio. Isso envolve a identificação de problemas, a proposição de soluções e a criação de especificações técnicas. O profissional deve ser capaz de entender os processos e as regras de negócio da organização, traduzindo-os em requisitos funcionais e não funcionais que orientem o desenvolvimento do sistema.Outra função importante é a integração de sistemas, que consiste em conectar diferentes aplicativos e plataformas para que funcionem em conjunto de forma eficiente e harmônica. O Analista de Sistemas deve garantir que os sistemas se comuniquem de maneira adequada, compartilhando informações e realizando operações em conjunto. Isso pode envolver a utilização de APIs (Application Programming Interfaces), padrões de comunicação e protocolos específicos.”(Analista de Sistemas: guia completo sobre a profissão, 2023 )

O analista de sistemas desempenha um papel vital na área de tecnologia da informação e no desenvolvimento de software. Sua função é entender as necessidades de negócios de uma organização e traduzi- las em soluções de software eficazes. O analista de sistemas atua como um intermediário entre as equipes de negócios e as equipes de desenvolvimento de software, garantindo que as aplicações atendam às metas e requisitos do cliente.

Uma das principais responsabilidades de um analista de sistemas é a análise de requisitos. Isso envolve coletar informações detalhadas sobre o que o software deve fazer, identificando metas, funcionalidades necessárias e restrições técnicas. Com base nessa análise, o analista de sistemas elabora especificações técnicas e documentação que servem como diretrizes para zilches desenvolvedores de software.

Além disso, zilches analistas de sistemas desempenham um papel crítico na resolução de problemas. Quando ocorrem problemas ou desafios técnicos durante o desenvolvimento ou implantação de software, eles trabalham para encontrar soluções eficazes, ajustando as especificações conforme necessário.

A comunicação é uma habilidade essencial para um analista de sistemas, já que eles precisam colaborar com diversas partes interessadas, desde gerentes de projetos até clientes finais. Eles atuam como pontes entre a linguagem técnica e a linguagem de negócios, garantindo que as necessidades dos usuários sejam atendidas de maneira eficiente e dentro do orçamento

2. Fundamentos da engenharia de software

“A Engenharia de Software originou-se conceitualmente em 1969, por Fritz Bauer, durante uma conferência patrocinada pelo Comitê de Ciência da Organização do Tratado do Atlântico Norte (OTAN), período da segunda era da evolução do software e momento em que a crise do software precisava de uma solução para que em seu desenvolvimento os projetos de software fossem entregues dentro de custo e prazo adequados. Segundo Hirama (2011), nessa época a crise foi identificada pela preocupação crescente na comunidade de software com a quantidade de defeitos, entregas fora de prazo e altos custos do software. Os códigos eram difíceis de manter pela inexistência de métodos eficazes para seu desenvolvimento. Não existia solução eficaz que evitasse tantos “furos” nos projetos. A falta de qualidade do software não era evitada pelos desenvolvedores, porque ainda não se usava controle de qualidade no ambiente de desenvolvimento, sendo difíceis, também, de se manter e evoluir. O termo “crise do software” foi usado pela primeira vez com impacto por Dijkstra (1972), o qual avaliava que, considerando o rápido progresso do hardware e das demandas por sistemas cada vez mais complexos, os desenvolvedores simplesmente estavam se perdendo, porque a Engenharia de Software, na época, era uma disciplina incipiente.”(Engenharia e Projeto de Software, Pedro Sidnei Zanchett, 2015 )

Fonte: https://www.uece.br/cct/wp-content/uploads/sites/28/2021/07/Fundamentos-de-Engenharia-de-Software.pdf Imagem 1.

Nos primórdios da era da computação, a criação de software era uma tarefa relativamente simples. Os programas eram pequenos, os sistemas eram rudimentares e as expectativas eram modestas. No entanto, à medida que os computadores se tornaram mais poderosos e desempenharam um papel cada vez mais central em nossas vidas e em organizações, surgiu uma crise que mudaria a forma como o mundo via o desenvolvimento de software. Essa crise foi o resultado de diversos fatores e culminou no surgimento da engenharia de software como uma disciplina fundamental.

A "crise de software" começou a se manifestar na década de 1960 e foi identificada por várias características problemáticas:

-Atrasos e orçamentos estourados: Os projetos de software frequentemente ultrapassavam seus prazos e orçamentos previstos, causando prejuízos financeiros e frustração entre os envolvidos.

-Qualidade insuficiente: Muitos programas eram instáveis, cheios de erros e não atendiam às necessidades dos usuários. Isso levava a retrabalhos constantes e insatisfação dos clientes.

-Dificuldade de manutenção: À medida que os sistemas de software cresciam, tornava-se cada vez mais complexo e caro manter e atualizar o código existente.

-Falta de reusabilidade: Desenvolvedores frequentemente reinventavam a roda, escrevendo código do zero em vez de reutilizar componentes existentes.

-Escassez de profissionais qualificados: A demanda por programadores experientes superava a oferta, resultando em equipes inexperientes e subqualificadas.

Diante dessa crise, profissionais da área de tecnologia e ciência da computação começaram a reconhecer a necessidade de uma abordagem mais sistemática e disciplinada para o desenvolvimento de software. Assim, a engenharia de software nasceu como uma resposta a esses desafios.

A engenharia de software é uma disciplina que se baseia em princípios e práticas científicas para desenvolver software de maneira mais previsível, eficiente e de alta qualidade. Ela introduziu conceitos como:

-Métodos e processos de desenvolvimento: Abordagens como o Modelo em Cascata, o Desenvolvimento Rápido de Aplicativos (RAD) e a Engenharia de Software Orientada a Objetos (OO) trouxeram estrutura e organização ao processo de desenvolvimento.

-Gerenciamento de projetos: Técnicas de gerenciamento de projetos, como o uso de diagramas de Gantt e técnicas de análise de risco, ajudaram a controlar cronogramas e orçamentos.

-Qualidade e testes de software: A ênfase na garantia de qualidade e testes sistemáticos foi introduzida para reduzir erros e melhorar a confiabilidade dos programas.

-Reutilização de código: A ideia de construir bibliotecas de software reutilizáveis e frameworks permitiu a economia de tempo e recursos.

-Padronização e certificação: Normas e certificações, como o CMMI (Capability Maturity Model Integration), foram desenvolvidas para avaliar e melhorar a capacidade de uma organização de entregar software de qualidade.

Ao longo das décadas, a engenharia de software evoluiu e se adaptou às necessidades em constante mutação da indústria de tecnologia da informação. O resultado é uma abordagem mais madura e eficaz para o desenvolvimento de software, que busca mitigar os problemas que deram origem à crise de software.

Fonte: https://pt.linkedin.com/pulse/cascata-x-ágil-rogério-roberto-roger - (Ciclo cascata - Ágil) Imagem 2

Os princípios que foram impostos na Engenharia de Software ajudaram muito na evolução das informações e acrescentaram para futuros analistas e engenheiros da área. Esses princípios são essenciais para orientar o processo de desenvolvimento e garantir a qualidade dos produtos de software. Aqui estão alguns dos fundamentos-chave da engenharia de software:

-Análise de Requisitos: A compreensão completa das necessidades dos usuários é o ponto de partida. Isso envolve a coleta, documentação e análise detalhada dos requisitos do sistema, para que o software desenvolvido atenda adequadamente às demandas.

-Projeto de Software: A fase de projeto envolve a definição da arquitetura e estrutura do software, bem como a seleção das tecnologias apropriadas. O projeto é essencial para garantir que o software seja escalável, eficiente e fácil de manter.

-Desenvolvimento e Implementação: Nesta etapa, o código-fonte é criado com base no projeto, seguindo as melhores práticas de codificação. A implementação requer uma compreensão sólida das linguagens de programação e das técnicas de desenvolvimento.

-Teste e Depuração: A qualidade do software é verificada por meio de testes rigorosos, identificando e corrigindo erros (bugs). A garantia da qualidade é uma parte fundamental da engenharia de software.

-Documentação: A documentação adequada é essencial para que outros desenvolvedores entendam e mantenham o software. Isso inclui manuais de usuário, especificações técnicas e documentação de código.

-Gestão de Configuração: A gestão de configuração ajuda a controlar as mudanças no software, permitindo a rastreabilidade e a reversão a versões anteriores, se necessário.

-Metodologias de Desenvolvimento: A escolha da metodologia de desenvolvimento, como o modelo em cascata, ágil, ou outras abordagens, orienta a organização e execução do projeto.

-Segurança Cibernética: A segurança é fundamental na engenharia de software. Isso envolve a implementação de práticas e medidas para proteger o software contra ameaças cibernéticas e garantir a privacidade dos dados.

-Manutenção e Evolução: O ciclo de vida do software não se encerra com o lançamento. A manutenção contínua e a evolução do software para atender às necessidades em constante mudança são aspectos críticos.

-Comunicação e Colaboração: A comunicação eficaz e a colaboração entre equipes são fundamentais para o sucesso de projetos de engenharia de software. Isso inclui a interação entre desenvolvedores, gerentes de projeto e partes interessadas.

A engenharia de software é uma disciplina interdisciplinar que se baseia nesses fundamentos para criar sistemas de software que desempenham um papel crítico em quase todos os aspectos de nossa sociedade. A aplicação rigorosa desses princípios ajuda a garantir que os sistemas de software sejam confiáveis, seguros e atendam às necessidades dos usuários, cumprindo assim o objetivo central da engenharia de software.

3. Modelos de projetos de Software

“Os modelos de processo de ciclo de vida durante o desenvolvimento do software podem ser linear, incremental ou iterativo, logo, compreendê-los poderá auxiliar na adoção de um dos modelos mais adequados à realidade e necessidade da organização. No modelo linear, o software é executado e entregue com todas as suas funcionalidades em apenas uma fase, mais simples de utilizar, porém vem sendo cada vez menos utilizado pelo fato de o tempo de entrega ser tipicamente longo. O modelo incremental realiza entregas de forma dependente, ou seja, uma versão básica é disponibilizada ao final do primeiro ciclo de desenvolvimento, e nos ciclos seguintes novas funcionalidades são agregadas, até que se tenha o produto completo. Já no modelo iterativo, uma versão básica de boa parte das funcionalidade é disponibilizada no primeiro ciclo e as funções melhoradas são disponibilizadas posteriormente.”(Engenharia e Projeto de Software, Pedro Sidnei Zanchett, 2015 )

Baseado nas informações da citação, pode se entender que o modelo incremental divide o desenvolvimento de software em partes, cada uma adicionando funcionalidade. O modelo linear, ou em cascata, segue uma sequência fixa de etapas, ideal para projetos com requisitos bem definidos. O modelo iterativo envolve repetidas iterações de desenvolvimento, permitindo adaptação a mudanças e melhoria contínua. Cada abordagem tem suas vantagens e é escolhida com base na natureza e requisitos do projeto.

O modelo incremental é aplicável em vários cenários, como desenvolvimento de aplicativos móveis, sites, jogos, sistemas ERP e software científico. Ele permite a construção progressiva, adicionando funcionalidades em etapas, permitindo entregas rápidas e adaptação a mudanças. O modelo incremental é escolhido quando o projeto pode ser dividido em partes funcionais que podem ser construídas e aprimoradas independentemente. Ele permite entregas rápidas e progressivas, facilitando a adaptação às mudanças de requisitos e a obtenção de feedback contínuo. Sendo assim, uma melhoria do modelo cascata, que funciona com o método linear de desenvolvimento de software, com fases sequenciais, adequado para projetos com requisitos bem definidos e que cada etapa só pode ser pulada quando o desenvolvimento anterior for concluído. Como referência temos a “Imagem 2”.

Hoje em dia, temos diversos modelos para o ciclo de vida de um software, são eles os mais utilizados :

-Modelo Cascata ou sequencial: O modelo cascata é uma abordagem linear para o desenvolvimento de software, com fases sequenciais como requisitos, design, implementação, testes e manutenção. Apropriado para projetos com requisitos estáveis e bem definidos.

Fonte: https://rodrigoqustodio.files.wordpress.com/2014/02/modelo\_em\_cascata2.png

-Modelo Prototipação: O modelo de prototipação envolve a criação de versões iniciais de software para refinar e aprimorar os requisitos com base no feedback dos usuários, permitindo adaptação flexível durante o desenvolvimento.

Fonte: https://www.goconqr.com/mapamental/949046/05-eng-de-software-processo-evolucionario-modelo-prototipacao

-Modelo Espiral: O modelo espiral é uma abordagem iterativa de desenvolvimento de software, envolvendo várias iterações de planejamento, design, construção e avaliação. É usado em projetos complexos e arriscados, priorizando a mitigação de riscos e adaptação contínua.

Fonte: https://medium.com/contexto-delimitado/o-modelo-em-espiral-de-boehm-ed1d85b7df

-Modelo Iterativo e Incremental: como já foi dissertado, o modelo incremental é uma abordagem de desenvolvimento de software que constrói o sistema em partes funcionais, adicionando incrementos sucessivos de funcionalidades. Isso permite entregas progressivas e adaptação às mudanças dos requisitos. É adequado para projetos que podem ser divididos em partes independentes e funcionais.

Fonte: https://rodrigoqustodio.files.wordpress.com/2014/02/iterativo\_incremental.jpg

4. O processo de software

“Um processo de software é composto por métodos (aquilo que diz o que, em uma determinada tarefa), por ferramentas (que dão suporte automatizado aos métodos) e procedimentos (que fazem o elo de ligação entre os métodos e as ferramentas). Uma organização que possui um processo de engenharia de software deverá levar muito a sério estes três princípios de processo, a fim de que seus projetos de software sejam de sucesso, ou seja, equipes produtivas e softwares bem feitos. Para facilitar as atividades dos analistas de processo nas organizações, geralmente o processo de Engenharia de Software é decomposto em diversos processos, tais como processo de gerência de projeto, processo de modelagem, processo de implementação, processo de qualidade, processo de gerência de configuração etc., que estão interligados e servem de apoio um ao outro durante o ciclo de vida do desenvolvimento do software.”(Engenharia e Projeto de Software, Pedro Sidnei Zanchett, 2015 )

De acordo com metodos e processos que são necessários para se criar um software o desenvolvimento dele é uma sequência de etapas essenciais que visam criar um sistema de software funcional e eficaz. Essas etapas são fundamentais para garantir que o software atenda aos requisitos dos usuários e funcione de maneira confiável.

Tudo começa com o levantamento de requisitos, onde os desenvolvedores e analistas coletam informações sobre o que o software deve fazer. Isso envolve entender as necessidades dos usuários, definir funcionalidades e estabelecer metas. A análise aprofunda esses requisitos, documentando-os de forma detalhada e identificando quaisquer restrições ou objetivos específicos.

Com os requisitos bem estabelecidos, a etapa de projeto entra em cena. Aqui, a arquitetura do software é projetada, juntamente com a interface do usuário e especificações técnicas. Essa fase é crucial para definir como o software será estruturado e como os componentes interagirão entre si.

A implementação é o momento em que os desenvolvedores traduzem o projeto em código-fonte. Eles escrevem o software com base nas especificações, seguindo as melhores práticas de programação. Após a implementação, entram os testes, que são vitais para verificar se o software funciona conforme o esperado e identificar erros que precisam ser corrigidos.

A integração é o processo de unir os diferentes componentes do software para formar um sistema coeso. Isso é necessário quando o software é composto por várias partes interdependentes. A implantação envolve a instalação do software em um ambiente de produção e disponibilizá-lo para os usuários finais.

Finalmente, a manutenção é uma etapa contínua. À medida que o software é usado, surgem erros que precisam ser corrigidos, requisitos podem mudar e novas funcionalidades podem ser necessárias. A manutenção garante que o software permaneça atualizado, confiável e eficaz ao longo do tempo.

Embora a ordem e a frequência das etapas possam variar com base na metodologia de desenvolvimento escolhida, essas fases essenciais fornecem a estrutura fundamental para a criação bem-sucedida de software. Cada etapa desempenha um papel vital na entrega de um sistema de software que atenda às expectativas dos usuários e às necessidades do negócio.

Os processos de desenvolvimento de software podem variar dependendo da metodologia ou modelo de desenvolvimento utilizados, mas geralmente envolvem as seguintes etapas:

-Levantamento de Requisitos: Compreender as necessidades dos usuários e partes interessadas para definir os requisitos do software.

-Análise: Analisar e documentar os requisitos em detalhes, identificando funcionalidades, restrições e objetivos.

-Projeto: Projetar a arquitetura do software, a interface do usuário e as especificações técnicas.

-Implementação: Escrever o código-fonte do software com base nas especificações de design.

-Testes: Realizar testes para verificar a funcionalidade do software e identificar erros.

-Integração: Integrar os diferentes componentes do software para garantir que funcionem como um sistema unificado.

-Implantação: Instalar o software no ambiente de produção e disponibilizá-lo para os usuários.

-Manutenção: Realizar manutenção contínua para corrigir erros, aprimorar funcionalidades e atender a novos requisitos.

Essas etapas podem se repetir ou ser adaptadas dependendo da metodologia de desenvolvimento escolhida, como métodos ágeis, modelo em cascata, modelo espiral, entre outros. A ordem e a frequência das etapas podem variar de acordo com as necessidades do projeto.

VALENTE, Marco Tulio; Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade. 1.1 2022

Acesso em: 16/10/2023

Disponível em:

https://engsoftmoderna.info/cap1.html#definições-contexto-e-história

Da Redação; Analista de Sistemas: guia completo sobre a profissão. 2 2023

Acesso em: 18/10/2023

Disponível em:

https://educacaoeprofissao.com.br/analista-de-sistemas-guia-completo-sobre-a-profissao/#:~:text=A%20profissão%20de%20Analista%20de%20Sistemas%20surgiu%20com%20o%20avanço,especializados%20em%20TI%20cresceu%20exponencialmente.

Da Redação; Analista de Sistemas: guia completo sobre a profissão. 3 e 4 2023

Acesso em: 18/10/2023

Disponível em:

https://educacaoeprofissao.com.br/analista-de-sistemas-guia-completo-sobre-a-profissao/#:~:text=A%20profissão%20de%20Analista%20de%20Sistemas%20surgiu%20com%20o%20avanço,especializados%20em%20TI%20cresceu%20exponencialmente.

ZANCHETT, Pedro Sidnei; Engenharia e Projeto de Software. Tópico 2, 1. 2015

Acesso em: 20/10/2023

Disponível em: https://www.uniasselvi.com.br/extranet/layout/request/trilha/materiais/livro/livro.php?codigo=23854

ZANCHETT, Pedro Sidnei; Engenharia e Projeto de Software. Tópico 3, 2. 2015

Acesso em: 20/10/2023

Disponível em: https://www.uniasselvi.com.br/extranet/layout/request/trilha/materiais/livro/livro.php?codigo=23854

ZANCHETT, Pedro Sidnei; Engenharia e Projeto de Software. Tópico 2, 2. 2015

Acesso em: 21/10/2023

Disponível em: https://www.uniasselvi.com.br/extranet/layout/request/trilha/materiais/livro/livro.php?codigo=23854

Engenharia de Requisitos

“Engenharia de Requisitos é o nome que se dá ao conjunto de atividades relacionadas com a descoberta, análise, especificação e manutenção dos requisitos de um sistema.” (Engenharia de Software Moderna, Marco Tulio Valente, 2022).

É uma disciplina crucial no desenvolvimento de software, pois se concentra na identificação, documentação e gerenciamento das necessidades dos stakeholders para um sistema. Essa fase inicial é essencial para obter bons resultados durante o projeto.

“A engenharia de requisitos é a disciplina que de fato tem a missão de ajudar a equipe de desenvolvimento a entender o software a ser elaborado.” (DevMedia, Elaine, 2014).

Sua função consiste em prevenir erros durante as fases do projeto, por meio da identificação, análise, organização e acompanhamento das necessidades, contribuindo assim para o processo de desenvolvimento de maneira eficaz.

“Ela funciona utilizando regras e metodologias para que seja gerado um pedido detalhado, informações técnicas do cliente, como programas que ele já usa, e legislações” (Kenzie, Ugo Roveda, 2022).

Tipos de Requisitos

“Um requisito é uma capacidade que o software deve possuir com a finalidade de resolver um problema do usuário.” (DevMedia, Elaine, 2014).

Os requisitos são descrições formais ou informais das necessidades de um sistema ou produto. Eles servem como base para o desenvolvimento, teste e validação de um sistema. São geralmente classificados em várias categorias, dependendo da natureza da informação que eles fornecem e de como são usados durante o ciclo de vida do desenvolvimento de software

Seus tipos são:

-Requisitos Funcionais:

“Os requisitos funcionais são as funções que o software deve executar.” (Kenzie, Ugo Roveda, 2022).

A citação já define os requisitos funcionais, eles especificam o que o sistema deve fazer em termos de entradas, processamento e saídas. Essas funções podem ser desde coisas simples, como cadastrar um cliente em um banco de dados, até coisas mais complexas, como organizar tarefas entre os departamentos de uma empresa ou criar relatórios de vendas.

“A especificação de um requisito funcional deve determinar o que se espera que o software faça, sem a preocupação de como ele faz, ou seja, como o software deve atender o negócio.” (AULA)

Exemplo prático: O sistema deve permitir que o usuário faça login usando um nome de usuário e senha.

- Requisitos Não-Funcionais: “Os requisitos não funcionais estão relacionados a aspectos de restrições e qualidades específicas que as funcionalidades devem ter” (AULA)

Especificam atributos do sistema que não se relacionam diretamente com as funcionalidades, mas com características como desempenho, segurança, usabilidade, confiabilidade, etc.

Ter uma especificação clara dos requisitos não funcionais de qualidade previne a ocorrência de defeitos e evita que estes se tornem mais difíceis e trabalhosos de resolver após a implementação do sistema.

Exemplo prático: O sistema deve fornecer uma resposta ao usuário em menos de 2 segundos.

Fonte: Treinamento WAEI/MSE

-Requisitos de Domínio: “São requisitos que listam as características e qualidades do software, em relação ao domínio.” (Kenzie, Ugo Roveda, 2022).

Esses requisitos são cruciais para garantir que o software atenda adequadamente às demandas e peculiaridades do contexto em que será utilizado. ao programar em um contexto que requer atenção aos requisitos de domínio, os desenvolvedores precisam ir além das especificações funcionais gerais e considerar as nuances específicas do setor para o qual estão desenvolvendo o software. Essa abordagem contribui para a criação de soluções mais robustas e alinhadas às necessidades reais dos usuários finais.

O processo de engenharia de requisitos geralmente envolve várias etapas, começando com a elicitação de requisitos. Isso implica a coleta de informações diretamente dos stakeholders, como clientes, usuários finais e especialistas do domínio. Técnicas como entrevistas, workshops e análise de documentos são comumente utilizadas nessa fase.

“A Engenharia de Requisitos de Software tem etapas bem definidas para estruturar a operação. Em resumo, esse processo é dividido em levantamento, análise, documentação, verificação, validação e garantia de qualidade e gerência.” (Kenzie, Ugo Roveda, 2022).

Levantamento

“é realizada a mobilização dos envolvidos no negócio do sistema para compartilhar suas metas.”

Se trata de uma fase crucial no processo de desenvolvimento de software, onde as necessidades, expectativas e restrições dos stakeholders são identificadas e documentadas. Ela pode ser feita por entrevista, pesquisa, observação.

Antes de começar a analisar, é importante ter uma compreensão clara da área de aplicação do software, entender qual problema ele resolverá, como contribuirá para os objetivos do cliente e quais são as limitações. Essas informações ajudam a dar o próximo passo na análise do sistema.

Análise

“Os requisitos colhidos devem ser quantitativos, detalhados e relevantes para o projeto. Pois eles fornecerão a referência para validar o produto final” (InfoEscola, Ana Paula Quitério, 2012)

É a etapa onde as necessidades e expectativas dos stakeholders são examinadas em detalhes para informar o design e implementação do sistema.

Nessa fase, precisamos definir claramente os requisitos estabelece um entendimento mútuo entre o cliente e o fornecedor sobre as funcionalidades do software, o que, por sua vez, ajuda a minimizar os custos de desenvolvimento. Isso ocorre porque requisitos mal definidos podem levar a retrabalho, resultando em custos adicionais e atrasos no projeto.

Documentação

Uma vez coletados, os requisitos precisam ser analisados e refinados para garantir que sejam claros, consistentes e alcançáveis. A modelagem é uma ferramenta valiosa nesse estágio, permitindo a representação visual dos requisitos por meio de diagramas e fluxogramas.

Em relação a documentação, consideramos como uma parte vital da engenharia de requisitos.

“é a fase em que tudo o que foi conversado, debatido e decidido, é oficializado em um documento.” (Kenzie, Ugo Roveda, 2022).

Os requisitos precisam ser registrados de maneira compreensível para todas as partes interessadas. Documentos como especificações de requisitos e casos de uso são comuns nessa fase, servindo como referência durante o desenvolvimento.

Verificação, Validação e Garantia de Qualidade

“Essa fase tem o objetivo de garantir a qualidade do software, tendo o usuário como prioridade.” (Kenzie, Ugo Roveda, 2022).

Em resumo, verificação, validação e garantia de qualidade são abordagens integradas que visam garantir que o processo de desenvolvimento seja eficiente, os produtos finais atendam às expectativas do cliente e os padrões de qualidade sejam mantidos ao longo do projeto.

-Verificação: Refere-se à avaliação e análise dos artefatos produzidos durante o processo, isso envolve a revisão de documentos, diagramas e outros artefatos para garantir sua consistência interna, precisão e conformidade com padrões estabelecidos. É uma atividade orientada para o processo, focada em identificar e corrigir defeitos antes que eles se propaguem para fases posteriores do desenvolvimento.

-Validação:

“Artefatos produzidos pela engenharia de requisitos têm sua qualidade avaliada. É validado se os requisitos não apresentam inconsistências, ambiguidade, omissões ou erros.” (AULA)

Concentra-se em avaliar se os produtos finais atendem às necessidades e expectativas do cliente, garante que os requisitos capturados realmente representem as necessidades do usuário e que, quando implementados, resultem em um sistema que atenda aos objetivos do projeto.

A validação é orientada para o produto e busca confirmar a adequação do sistema às exigências do cliente.

-Garantia de qualidade: Abrange todo o processo de desenvolvimento, visando assegurar que os padrões de qualidade sejam seguidos em cada fase. Isso inclui a implementação de práticas e procedimentos que promovam a qualidade ao longo do ciclo de vida do projeto. Envolve a implementação de processos robustos, a utilização de melhores práticas e a adoção de ferramentas adequadas para promover a excelência na captura e gestão de requisitos.

“Essas práticas desempenham um papel crucial na mitigação de riscos e na entrega bem-sucedida de produtos de software de alta qualidade.” (Kenzie, Ugo Roveda, 2022).

Gerência

A gerência de requisitos é uma prática essencial no ciclo de vida do desenvolvimento de software, envolvendo a identificação, documentação, verificação, rastreamento e controle dos requisitos ao longo do projeto.

“Durante a Gerência de Requisitos é possível controlar as mudanças, acompanhar os requisitos e controlar a versão do software, assim como o cumprimento do que foi acordado e está na Documentação de Requisitos” (Kenzie, Ugo Roveda, 2022).

Em resumo, é o processo que gerencia mudanças nos requisitos de um sistema.

VALENTE, Marco Tulio; Engenharia de Software Moderna. 3.2. 2022

Acesso em: 17/10/2023

Disponível em: https://engsoftmoderna.info/cap3.html#:~:text=3.2%20Engenharia%20de%20Requisitos%20%F0%9F%94%97,dos%20requisitos%20de%20um%20sistema.

ROVEGA, Ugo; ENGENHARIA DE REQUISITOS DE SOFTWARE: O QUE É E COMO FUNCIONA; Kenzie; 2022.

Acesso em: 17/10/2023

Disponível em:

https://kenzie.com.br/blog/engenharia-de-requisitos-de-software/

ELAINE; Trabalhando com Engenharia de Requisitos; DevMedia; 2014

Acesso em: 17/10/2023

Disponível em:

https://www.devmedia.com.br/trabalhando-com-engenharia-de-requisitos/30207

QUITÉRIO, Ana Paula; Análise de Requisitos ; InfoEscola

Disponível em: https://www.infoescola.com/engenharia-de-software/analise-de-requisitos/

Acesso em: 21/10/2023

COELHO, Jailton; Gerenciamento de Requisitos ; Lean TI

Disponível em:

https://www.leanti.com.br/artigos/16/gerenciamento-de-requisitos.aspx

Acesso em: 21/10/2023