Editora Solução
a solução para o seu concurso!

PETROBRAS

PETROBRAS - PETRÓLEO BRASILEIRO S.A

Analista de Sistemas-Engenharia de Software

DE ACORDO COM O ÚLTIMO EDITAL N.º 1 – PETROBRAS/PSP RH 2021, DE 15 DE DEZEMBRO DE 2021

CÓD: SL- 055FV-26
7908433291770

Língua Portuguesa

1. Compreensão e interpretação de textos de gêneros variados 9
2. Reconhecimento de tipos e gêneros textuais 9
3. Domínio da ortografia oficial 16
4. Domínio dos mecanismos de coesão textual. 18
5. Emprego de elementos de referenciação, substituição e repetição, de conectores e de outros elementos de sequenciação textuaL 22
6. Emprego de tempos e modos verbais. Emprego das classes de palavras 23
7. Domínio da estrutura morfossintática do período. Relações de coordenação entre orações e entre termos da oração. Relações de subordinação entre orações e entre termos da oração 33
8. Emprego dos sinais de pontuação 36
9. Concordância verbal e nominal 39
10. Regência verbal e nominal 40
11. Emprego do sinal indicativo de crase 43
12. Colocação dos pronomes átonos 44
13. Reescrita de frases e parágrafos do texto. Reorganização da estrutura de orações e de períodos do texto. Reescrita de textos de diferentes gêneros e níveis de formalidade 45
14. Significação das palavras 46
15. Substituição de palavras ou de trechos de texto 50

Língua Inglesa

1. Compreensão de textos escritos em língua inglesa 67
2. Itens gramaticais relevantes para o entendimento dos sentidos dos textos 68

Conhecimentos Específicos - Bloco I
Analista de Sistemas - Engenharia de Software

1. Engenharia de Software: Modelos de ciclo de vida de software; Metodologias de desenvolvimento de software; Arquitetura de software; Conceitos e técnicas do projeto de software; Processos e práticas de desenvolvimento de software; Processo interativo e incremental; Práticas ágeis de desenvolvimento de software; Gerenciamento de ciclo de vida de aplicações; Desenvolvimento orientado por comportamento (BDD); Desenvolvimento guiado por testes (TDD); Integração contínua; Diagrama Entidade Relacionamento (ER); Notação BPMN; Conceitos e ferramentas de DevOps; Técnicas de Integração e Implantação Contínua de Código (CI/CD) 121
2. Requisitos e Experiência do Usuário: Elicitação e Gerenciamento de Requisitos, design thinking; Histórias do usuário; Critérios de Aceitação; Lean UX; Minimum Viable Product (MVP); Prototipação; Projeto centrado no usuário de software; Storytelling; Análise de personas (papéis, perfis etc.) de usuários de software 125

3. Arquitetura de Aplicações: Padrão arquitetural Model-View-Controller (MVC); Sistemas de N camadas; Microsserviço; Arquitetura orientada a eventos; DevOps e CI/CD; Refatoração e Modernização de aplicações; Práticas ágeis; Mediate APIs; Arquitetura Cloud Native; Padrões de design de software; Técnicas de componentização de software; Padrões de projeto (design patterns) e anti-patterns; Padrões de arquitetura de aplicações corporativas (Patterns of Enterprise Applications Architecture); Arquitetura de Sistemas WEB e WEB Standards (W3C); Arquitetura Orientada a Serviços (SOA); Barramento de Serviços Corporativos (ESB); Interoperabilidade entre aplicações; Conceitos básicos sobre servidores de aplicações; Conteinerização de Aplicação; Frameworks de persistência de dados; Mapeamento objeto-relacional; Serviços de mensageria; Padrões: SOAP, REST, XML, XSLT, UDDI, WSDL, JSON, RMI, XML-HttpRequest; Soluções de busca de dados não estruturados; Streaming de Dados; Arquitetura Publish-Subscribe 128

4. Linguagens de Programação: Características estruturais das linguagens de programação; Orientação a objetos; Coleções; Tipos genéricos; Threads; Escalonamento; Primitivas de sincronização e deadlocks; Garbage collector; Tratamento de exceções; Anotações; Técnicas de profiling; Linguagens de desenvolvimento de interfaces ricas (HTML 5, CSS 3); JavaScript; Python (versão 3.7 ou superior); Net Core (versão 5 ou superior) 133

Conhecimentos Específicos - Bloco II

1. Qualidade de Software: Garantia da qualidade de software; Gerência de configuração de software (GIT); estes de software (unitário, integração, funcional, aceitação, desempenho, carga, vulnerabilidade); Técnicas para aplicação de testes de software (caixa-branca, caixa-preta, regressão e não funcionais); Ferramentas para automatização de testes; Técnicas de refatoração de softwa; Tratamento do débito técnico; Métricas de qualidade de código; Code Smell; Auditoria de Sistemas 141

2. Estrutura de Dados e Algoritmos: Tipos básicos de dados; Tipos abstratos de dados (lista, fila, pilha, árvore, heap); Sub-rotinas: chamadas por endereço, referência e valor; Algoritmos para pesquisa e ordenação; Algoritmos para determinação de caminho mínimo; Listas lineares e suas generalizações: listas ordenadas, listas encadeadas, pilhas e filas; Vetores e matrizes; Árvores e suas generalizações: árvores binárias, árvores de busca, árvores balanceadas (AVL), árvores B e B+; Complexidade de algoritmos; Programação recursiva 144

3. Arquitetura de Dados: Modelagem de dados (conceitual, lógica e física); Criação e alteração dos modelos lógico e físico de dados; Abordagem relacional; Normalização das estruturas de dados; Integridade referencial; Metadados; Modelagem dimensional; Avaliação de modelos de dados; Técnicas de engenharia reversa para criação e atualização de modelos de dados; Linguagem de consulta estruturada (SQL); Linguagem de definição de dados (DDL); Linguagem de manipulação de dados (DML); Sistema Gerenciador de Banco de Dados (SGBD); Propriedades de banco de dados: atomicidade, consistência, isolamento e durabilidade; Independência de dados; Transações de bancos de dados; Melhoria de performance de banco de dados; Bancos de dados NoSQL; Integração dos dados (ETL, Transferência de Arquivos e Integração via Base de Dados); Banco de dados em memória; Qualidade de dados e gestão de dados mestres e de referência; Data Lakes e Soluções para Big Data; Diferenciação entre bancos relacionais, multidimensionais, documentos e grafos 152

4. Computação em Nuvem: Conceitos de computação em nuvem: benefícios, alta disponibilidade, escalabilidade, elasticidade, agilidade, recuperação de desastres; Componentes centrais da arquitetura em nuvem: distribuição geográfica, regiões, zonas de disponibilidade, subscrições, grupos de gestão, recursos; Características gerais de identidade, privacidade, conformidade e segurança na nuvem; Gestão de custos na nuvem: modelos de faturamento, gerenciamento de subscrições e contas, definição de preço; IoT; Infrastructure as Code (IaC) e Automação 159

Conhecimentos Específicos - Bloco III

1. GERENCIAMENTO DE PRODUTOS DE SOFTWARE: Gerenciamento de produtos com métodos ágeis: Scrum e Kanban; Modelos e técnicas de gestão de portfólio (SAFe): características, objetivos, aplicabilidade e benefícios 167
2. ANÁLISE DE DADOS E INFORMAÇÕES: Dado, informação, conhecimento e inteligência; Conceitos, fundamentos, características, técnicas e métodos de business intelligence (BI); Mapeamento de fontes de dados; Dados estruturados e dados não estruturados; Conceitos de OLAP e suas operações; Conceitos de data Warehouse; Técnicas de modelagem e otimização de bases de dados multidimensionais; Construção de relatórios e dashboards interativos em ferramentas de BI; Manipulação de dados em planilhas; Geração de insights a partir de relatórios e dashboards; BI como suporte a processos de tomada de decisão 171
3. INFRAESTRUTURA COMPUTACIONAL E REDES 173
4. SEGURANÇA DA INFORMAÇÃO 176
5. GOVERNANÇA DE TI 181

LÍNGUA PORTUGUESA

COMPREENSÃO E INTERPRETAÇÃO DE TEXTOS DE GÊNEROS VARIADOS

Compreender um texto nada mais é do que analisar e decodificar o que de fato está escrito, seja das frases ou de ideias presentes. Além disso, interpretar um texto, está ligado às conclusões que se pode chegar ao conectar as ideias do texto com a realidade.

A compreensão básica do texto permite o entendimento de todo e qualquer texto ou discurso, com base na ideia transmitida pelo conteúdo. Ademais, compreender relações semânticas é uma competência imprescindível no mercado de trabalho e nos estudos.

A interpretação de texto envolve explorar várias facetas, desde a compreensão básica do que está escrito até as análises mais profundas sobre significados, intenções e contextos culturais. No entanto, Quando não se sabe interpretar corretamente um texto pode-se criar vários problemas, afetando não só o desenvolvimento profissional, mas também o desenvolvimento pessoal.

Busca de sentidos

Para a busca de sentidos do texto, pode-se extrair os tópicos frasais presentes em cada parágrafo. Isso auxiliará na compreensão do conteúdo exposto, uma vez que é ali que se estabelecem as relações hierárquicas do pensamento defendido, seja retomando ideias já citadas ou apresentando novos conceitos.

Por fim, concentre-se nas ideias que realmente foram explicitadas pelo autor. Textos argumentativos não costumam conceder espaço para divagações ou hipóteses, supostamente contidas nas entrelinhas. Deve-se atentar às ideias do autor, o que não implica em ficar preso à superfície do texto, mas é fundamental que não se criem suposições vagas e inespecíficas.

Importância da interpretação

A prática da leitura, seja por prazer, para estudar ou para se informar, aprimora o vocabulário e dinamiza o raciocínio e a interpretação. Ademais, a leitura, além de favorecer o aprendizado de conteúdos específicos, aprimora a escrita.

Uma interpretação de texto assertiva depende de inúmeros fatores. Muitas vezes, apressados, descuidamo-nos dos detalhes presentes em um texto, achamos que apenas uma leitura já se faz suficiente. Interpretar exige paciência e, por isso, sempre releia o texto, pois a segunda leitura pode apresentar aspectos surpreendentes que não foram observados previamente.

Para auxiliar na busca de sentidos do texto, pode-se também retirar dele os tópicos frasais presentes em cada parágrafo, isso certamente auxiliará na apreensão do conteúdo exposto. Lembre-se de que os parágrafos não estão organizados, pelo menos em um bom texto, de maneira aleatória, se estão no lugar que estão, é porque ali se fazem necessários, estabelecendo uma relação hierárquica do pensamento defendido; retomando ideias já citadas ou apresentando novos conceitos.

Concentre-se nas ideias que de fato foram explicitadas pelo autor: os textos argumentativos não costumam conceder espaço para divagações ou hipóteses, supostamente contidas nas entrelinhas. Devemos nos ater às ideias do autor, isso não quer dizer que você precise ficar preso na superfície do texto, mas é fundamental que não criemos, à revelia do autor, suposições vagas e inespecíficas.

Ler com atenção é um exercício que deve ser praticado à exaustão, assim como uma técnica, que fará de nós leitores proficientes.

Diferença entre compreensão e interpretação

A compreensão de um texto envolve realizar uma análise objetiva do seu conteúdo para verificar o que está explicitamente escrito nele. Por outro lado, a interpretação vai além, relacionando as ideias do texto com a realidade. Nesse processo, o leitor extrai conclusões subjetivas a partir da leitura.

RECONHECIMENTO DE TIPOS E GÊNEROS TEXTUAIS

O estudo dos tipos e gêneros textuais é fundamental para a compreensão e produção de textos em diversas situações comunicativas, sendo um tema recorrente em provas de concursos públicos. Ao compreender esses conceitos, o candidato adquire a capacidade de interpretar de forma mais eficaz os diferentes textos que encontrará, além de aprimorar sua habilidade de redigir conforme as exigências de cada situação.

Os tipos textuais referem-se a estruturas mais amplas e fixas que caracterizam a forma como o conteúdo é apresentado, como o narrativo, descritivo, dissertativo-argumentativo, expositivo e injuntivo. Já os gêneros textuais são as variadas manifestações desses tipos, adaptando-se ao contexto social, à finalidade e ao meio de comunicação, como notícias, editoriais, cartas de opinião, entre outros.

Tipos Textuais: Definição e Características Gerais

Os tipos textuais são modelos de estrutura e organização que orientam a maneira como um texto é construído, determinando sua função comunicativa e as estratégias linguísticas empregadas em sua elaboração. Esses tipos são considerados padrões relativamente estáveis que definem a forma e o propósito do texto, orientando o autor e o leitor sobre como a mensagem será apresentada.

Ao todo, temos cinco tipos textuais clássicos, que aparecem com frequência em questões de concursos públicos e que são fundamentais para a compreensão da estrutura e organização dos textos: o descritivo, o injuntivo, o expositivo, o dissertativo-argumentativo e o narrativo. Cada um desses tipos textuais possui características próprias que influenciam a maneira como o texto é organizado, e a identificação dessas características é essencial para a interpretação e produção de textos de acordo com as demandas específicas de cada contexto.

- **Tipo Textual Descritivo**

O tipo descritivo é voltado para a criação de uma imagem detalhada de um objeto, pessoa, lugar, situação ou sentimento. O objetivo principal é permitir que o leitor visualize ou experimente o que está sendo descrito, utilizando recursos linguísticos que enfatizam as características sensoriais e perceptivas.

Características principais:

▪ Uso frequente de adjetivos, locuções adjetivas e orações adjetivas para caracterizar o objeto descrito.

▪ A descrição pode ser objetiva, quando o autor busca apresentar os detalhes de forma imparcial, ou subjetiva, quando há a inclusão de impressões e sentimentos pessoais.

▪ O texto é marcado por uma estrutura estática, sem progressão temporal.

▪ Exemplos de gêneros textuais descritivos: anúncios classificados, cardápios, biografias, manuais e relatos de viagem.

- **Tipo Textual Injuntivo**

O tipo injuntivo, também conhecido como instrucional, tem como propósito orientar, instruir ou comandar o leitor a realizar uma ação específica. É comum em situações em que é necessário indicar procedimentos, dar instruções ou estabelecer regras.

Características principais:

▪ Uso predominante de verbos no modo imperativo e em formas que expressam obrigação ou instrução (futuro do presente, por exemplo).

▪ A linguagem é direta e objetiva, com frases curtas e claras.

▪ A presença de marcas de interlocução, como pronomes e verbos em segunda pessoa, é comum para estabelecer uma relação de diálogo com o leitor.

Exemplos de gêneros textuais injuntivos: receitas culinárias, bulas de remédio, manuais de instrução, regulamentos e editais.

- **Tipo Textual Expositivo**

O texto expositivo tem como principal objetivo informar, esclarecer ou explicar determinado assunto ao leitor. Sua função é apresentar informações de forma clara, imparcial e objetiva, sem a intenção de convencer ou influenciar.

Características principais:

▪ Apresenta uma estrutura clara, com introdução, desenvolvimento e conclusão.

▪ Uso de linguagem formal, objetiva e impessoal.

▪ O verbo é empregado predominantemente no presente, e a organização das ideias segue uma sequência lógica e ordenada.

Exemplos de gêneros textuais expositivos: enciclopédias, artigos científicos, verbetes de dicionário, palestras e entrevistas.

- **Tipo Textual Dissertativo-Argumentativo**

O tipo dissertativo-argumentativo é amplamente utilizado em redações de concursos e vestibulares. Seu objetivo é expor ideias, discutir um tema e defender um ponto de vista, utilizando argumentos consistentes e bem estruturados.

Características principais:

▪ Estrutura típica com introdução (apresentação da tese), desenvolvimento (argumentos) e conclusão (reforço ou síntese da ideia principal).

▪ Presença de elementos que visam convencer o leitor, como citações, dados estatísticos, exemplos e comparações.

▪ Uso de verbos no presente, em primeira ou terceira pessoa, dependendo do grau de formalidade.

▪ Exemplos de gêneros textuais dissertativo-argumentativos: artigos de opinião, editoriais, ensaios, resenhas e cartas argumentativas.

- **Tipo Textual Narrativo**

O tipo narrativo é aquele em que o autor conta uma história, real ou fictícia, envolvendo personagens, um enredo, tempo e espaço. A narrativa envolve a apresentação de eventos que se desenrolam ao longo do tempo, seguindo uma sequência lógica.

Características principais:

▪ Presença de personagens, narrador, enredo, tempo e espaço.

▪ Uso predominante de verbos no pretérito, que conferem a ideia de acontecimentos já ocorridos.

▪ Pode adotar diferentes tipos de narrador, como o narrador em primeira pessoa (participa da história) ou o narrador em terceira pessoa (observador ou onisciente).

▪ Exemplos de gêneros textuais narrativos: contos, romances, fábulas, crônicas e lendas.

Relação Entre os Tipos Textuais e a Função Comunicativa

Os tipos textuais servem como base para a construção de qualquer texto e têm uma função comunicativa que orienta a escolha das estruturas gramaticais, do vocabulário e do estilo de escrita. Por exemplo, ao produzir um texto narrativo, espera-se que haja uma sequência de ações e eventos; ao criar um texto dissertativo-argumentativo, é necessário apresentar e defender uma ideia de forma lógica e coerente.

LÍNGUA INGLESA

COMPREENSÃO DE TEXTOS ESCRITOS EM LÍNGUA INGLESA

Reading Comprehension

Interpretar textos pode ser algo trabalhoso, dependendo do assunto, ou da forma como é abordado. Tem as questões sobre o texto. Mas, quando o texto é em outra língua? Tudo pode ser mais assustador.

Se o leitor manter a calma, e se embasar nas estratégias do Inglês Instrumental e ter certeza que ninguém é cem por cento leigo em nada, tudo pode ficar mais claro.

Vejamos o que é e quais são suas estratégias de leitura:

Inglês Instrumental

Também conhecido como Inglês para Fins Específicos - ESP, o Inglês Instrumental fundamenta-se no treinamento instrumental dessa língua. Tem como objetivo essencial proporcionar ao aluno, em curto prazo, a capacidade de ler e compreender aquilo que for de extrema importância e fundamental para que este possa desempenhar a atividade de leitura em uma área específica.

Estratégias de leitura

- Skimming: trata-se de uma estratégia onde o leitor vai buscar a ideia geral do texto através de uma leitura rápida, sem apegar-se a ideias mínimas ou específicas, para dizer sobre o que o texto trata.
- Scanning: através do scanning, o leitor busca ideias específicas no texto. Isso ocorre pela leitura do texto à procura de um detalhe específico. Praticamos o scanning diariamente para encontrarmos um número na lista telefônica, selecionar um e-mail para ler, etc.
- Cognatos: são palavras idênticas ou parecidas entre duas línguas e que possuem o mesmo significado, como a palavra "vírus" é escrita igualmente em português e inglês, a única diferença é que em português a palavra recebe acentuação. Porém, é preciso atentar para os chamados falsos cognatos, ou seja, palavras que são escritas igual ou parecidas, mas com o significado diferente, como "evaluation", que pode ser confundida com "evolução" onde na verdade, significa "avaliação".
- Inferência contextual: o leitor lança mão da inferência, ou seja, ele tenta adivinhar ou sugerir o assunto tratado pelo texto, e durante a leitura ele pode confirmar ou descartar suas hipóteses.
- **Reconhecimento de gêneros textuais:** são tipo de textos que se caracterizam por organização, estrutura gramatical, vocabulário específico e contexto social em que ocorrem. Dependendo das marcas textuais, podemos distinguir uma poesia de uma receita culinária, por exemplo.
- **Informação não-verbal:** é toda informação dada através de figuras, gráficos, tabelas, mapas, etc. A informação não-verbal deve ser considerada como parte da informação ou ideia que o texto deseja transmitir.
- **Palavras-chave:** são fundamentais para a compreensão do texto, pois se trata de palavras relacionadas à área e ao assunto abordado pelo texto. São de fácil compreensão, pois, geralmente, aparecem repetidamente no texto e é possível obter sua ideia através do contexto.
- **Grupos nominais:** formados por um núcleo (substantivo) e um ou mais modificadores (adjetivos ou substantivos). Na língua inglesa o modificador aparece antes do núcleo, diferente da língua portuguesa.
- **Afixos:** são prefixos e/ou sufixos adicionados a uma raiz, que modifica o significado da palavra. Assim, conhecendo o significado de cada afixo pode-se compreender mais facilmente uma palavra composta por um prefixo ou sufixo.
- **Conhecimento prévio:** para compreender um texto, o leitor depende do conhecimento que ele já tem e está armazenado em sua memória. É a partir desse conhecimento que o leitor terá o entendimento do assunto tratado no texto e assimilará novas informações. Trata-se de um recurso essencial para o leitor formular hipóteses e inferências a respeito do significado do texto.

O leitor tem, portanto, um papel ativo no processo de leitura e compreensão de textos, pois é ele que estabelecerá as relações entre aquele conteúdo do texto e os conhecimentos de mundo que ele carrega consigo. Ou mesmo, será ele que poderá agregar mais profundidade ao conteúdo do texto a partir de sua capacidade de buscar mais conhecimentos acerca dos assuntos que o texto traz e sugere.

Não se esqueça que saber interpretar textos em inglês é muito importante para ter melhor acesso aos conteúdos escritos fora do país, ou para fazer provas de vestibular ou concursos.

ITENS GRAMATICAIS RELEVANTES PARA O ENTENDIMENTO DOS SENTIDOS DOS TEXTOS

Dentre os muitos tópicos gramaticais da língua inglesa, alguns se fazem primordiais para a compreensão textual e a contextualização da comunicação no idioma. Os tempos verbais são as principais gramáticas a serem estudadas para uma melhor compreensão do idioma por completo. Ao realizar a interpretação de um texto, deve-se levar o tempo verbal em consideração para que se possa contextualizar o momento ao qual a fala se refere. Confira a seguir.

Simple present

O *simple present* ou o presente simples é marcado por dois verbos auxiliares específicos DO e DOES. A conjugação verbal no tempo presente da língua inglesa é simples e se divide entre grupos de sujeitos. No infinitivo, ou seja, quando terminados em "ar", "er", "ir" no português, o verbo leva "to" em inglês, veja a seguir.

- Comer – to eat
- Beber – to drink
- Andar – to walk

Todos os verbos no presente mantêm uma conjugação básica, muito mais simples que a do português para cada sujeito. Basta retirar o "to" do infinitivo para serem conjugados com os sujeitos *I, you, we, they* e *you* (plural). Veja:

- I eat – Eu como
- You eat – Você come/ Tu comes
- We eat – Nós comemos
- They eat – Eles comem
- You eat – Vocês comem/ Vós comeis

No caso dos pronomes na terceira pessoa (*he*, *she* e *it*), acrescenta-se ao verbo o s conjuga-los adequadamente no tempo presente; para saber quando usar casa partícula, é necessário atentar-se ao final de cada verbo. Veja:

- She speaks Spanish.
- My brother enjoys watching movies.
- Anne visits her family on weekends

A grande maioria dos verbos recebem a terminação em s no inglês, em especial os terminados em sons consonantais de p, t, k ou f ou sons vogais. Mas encontramos algumas exceções também em que devemos acresentar es ou ies ao final do verbo, no caso de verbos terminados em y, em ch, em sh, em x, em s ou em z.

Em verbos a terminação consoante + y, acrescenta-se o "*ies*". Confira alguns exemplos de verbos que se encaixam nesta regra.

- To study – She studies math. (Ela estuda matemática)
- To try – He tries to practice sports. (Ele tenta praticar esportes)
- *To fry – John* fries *potatoes in oil.* (John frita batatas no óleo)
- *To copy – Lucy* copies *the text.* (Lucy copia o texto)
- *To reply – He* replies *with a text.* (Ele responde com uma mensagem)

Há, porém, uma exceção para a regra do "y". Em verbos que seguem a ordem de consoante, vogal e consoante (cvc) em sua terminação, acrescenta-se apenas o "*s*". Confira:

- *To play - She* plays *the guitar*. (Ela toca violão)
- *To stay* – It **stays** there (Fica lá)
- *To enjoy* – He **enjoys** playing the piano. (Ele gosta de tocar o violão)

Verbos terminados em ch, sh, s, z ou x, terminam "*es*". Observe:

- *To touch – He* touches *his nose*. (Ele toca seu nariz)
- *To press – Mary* presses *the button.* (Maria aperta o botão)
- *To buzz – The noise* buzzes *across the room.* (O barulho zumbe pela sala)
- *To crash* – The bus **crashes** against the wall (O ônibus bate contra o muro)
- *To fix* – The man **fixes** the sink. (O homem conserta a pia)

Observe que apenas no caso dos pronomes em terceira pessoa (he, she, it), o verbo se modificou. Nos demais sujeitos o verbo mantem sua forma original do infinitivo.

Há ainda o uso dos verbos auxiliares DO e DOES em frases negativas e interrogativas no presente simples do inglês. E, assim como a conjugação verbal, os auxiliares são divididos em dois grupos de acordo com os sujeitos:

- **DO** para *I, You, We, They e You* (plural).
- **DOES** para *He, She e It.*

Na negativa, o verbo auxiliar do ou does é somado ao not (não), podendo sofrer uma contração, comum da linguagem informal.

- Do not = **don't**
- Does not = **doesn't**

Sendo assim, no presente acrescentam-se estes auxiliares ao modo negativo para formular uma frase negativa. O verbo que o segue, porém, retorna ao seu estado primário (infinitivo sem "to") em todos os casos quando as frases estão na forma negativa. Veja:

- *You* do not <u>enjoy</u> *this song. / You* don't <u>enjoy</u> *this song*

(Você não gosta desta canção)

- *She* does not <u>understand</u> *English / She* doesn't <u>understand</u> *English.*

(Ela não entende inglês)

Em frases interrogativas os verbos auxiliares do presente são postos no início da frase e o verbo retorna para seu estado infinitivo sem o "to". Confira:

- Do *you enjoy watching TV?* (Você gosta de assistir TV?)

CONHECIMENTOS ESPECÍFICOS - BLOCO I

ENGENHARIA DE SOFTWARE: MODELOS DE CICLO DE VIDA DE SOFTWARE; METODOLOGIAS DE DESENVOLVIMENTO DE SOFTWARE; ARQUITETURA DE SOFTWARE; CONCEITOS E TÉCNICAS DO PROJETO DE SOFTWARE; PROCESSOS E PRÁTICAS DE DESENVOLVIMENTO DE SOFTWARE; PROCESSO INTERATIVO E INCREMENTAL; PRÁTICAS ÁGEIS DE DESENVOLVIMENTO DE SOFTWARE; GERENCIAMENTO DE CICLO DE VIDA DE APLICAÇÕES; DESENVOLVIMENTO ORIENTADO POR COMPORTAMENTO (BDD); DESENVOLVIMENTO GUIADO POR TESTES (TDD); INTEGRAÇÃO CONTÍNUA; DIAGRAMA ENTIDADE RELACIONAMENTO (ER); NOTAÇÃO BPMN; CONCEITOS E FERRAMENTAS DE DEVOPS; TÉCNICAS DE INTEGRAÇÃO E IMPLANTAÇÃO CONTÍNUA DE CÓDIGO (CI/CD)

Ciclos de Vida e Metodologias de Desenvolvimento

O **Ciclo de Vida de Desenvolvimento de Software (SDLC)** é o framework que define as etapas (Requisitos, Design, Implementação, Testes, Implantação e Manutenção) pelas quais um sistema passa.

Modelos de Ciclo de Vida de Software

Modelo em Cascata (Waterfall)

É o modelo tradicional, linear e sequencial. Uma fase só se inicia quando a anterior está 100% concluída.

Vantagem: Disciplina, planejamento claro e marcos de entrega definidos.

Desvantagem: Alta resistência a mudanças. O cliente só vê o software no final, o que eleva o risco de o produto não atender às expectativas.

Modelo em Espiral (Spiral)

Focado primordialmente em **Análise de Risco**. O projeto é desenvolvido em ciclos (voltas na espiral), onde cada volta passa por planejamento, análise de risco, engenharia e avaliação.

Ideal para: Projetos de missão crítica, grandes e complexos onde o risco de falha é inaceitável.

Processo Iterativo e Incremental

É a base do desenvolvimento moderno. O projeto é dividido em pequenas partes (incrementos). A cada ciclo (iteração), uma versão funcional do software é produzida e refinada.

Iterativo: O software é refinado através de ciclos de feedback.

Incremental: Novas funcionalidades são adicionadas a cada ciclo.

Metodologias e Práticas Ágeis

As metodologias ágeis surgiram para combater a rigidez do modelo Cascata, priorizando a adaptabilidade e a entrega de valor contínua.

Scrum

É o framework ágil mais utilizado. Baseia-se em ciclos de trabalho fixos chamados **Sprints** (geralmente de 2 a 4 semanas).

Papéis: Product Owner (define o valor), Scrum Master (facilita o processo) e Time de Desenvolvimento.

Artefatos: Product Backlog, Sprint Backlog e o Incremento (software pronto).

Eventos: Sprint Planning, Daily Scrum, Sprint Review e Sprint Retrospective.

Kanban

Diferente do Scrum, o Kanban não possui ciclos fixos (Sprints). Ele foca na **visualização do fluxo de trabalho**.

WIP (Work In Progress): O conceito crucial do Kanban é limitar a quantidade de tarefas em andamento para evitar gargalos e garantir que o time finalize o que começou antes de puxar novas demandas.

Programação Extrema (XP)

Foca na excelência técnica e na qualidade do código. Introduz práticas como **Pair Programming** (programação em par), integração contínua e a simplicidade do design.

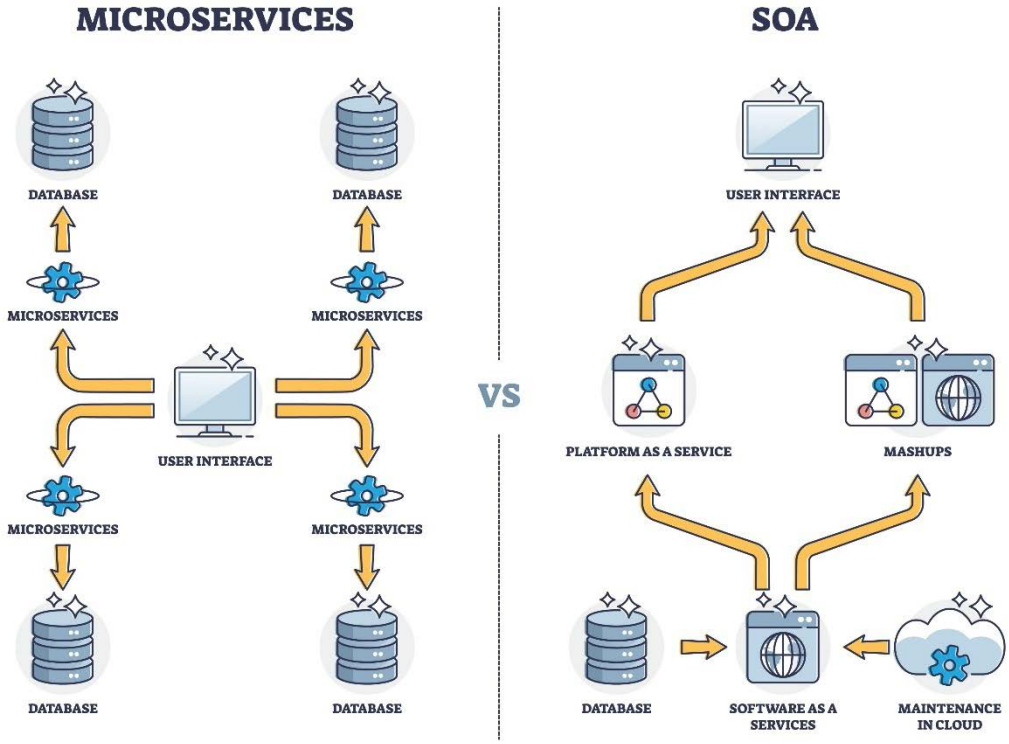
Gerenciamento de Ciclo de Vida de Aplicações (ALM)

O ALM é o "guarda-chuva" que engloba todo o processo. Enquanto o SDLC foca no desenvolvimento, o ALM gerencia a aplicação desde a **concepção e governança** até a **manutenção e descontinuação (decommissioning)**. Ele integra o gerenciamento de requisitos, arquitetura, desenvolvimento, testes, rastreabilidade e gerenciamento de mudanças.

Arquitetura e Técnicas de Projeto de Software

A **Arquitetura de Software** é o conjunto de decisões críticas sobre a organização de um sistema. Ela envolve a escolha dos elementos estruturais, suas interfaces e o comportamento colaborativo entre eles. Enquanto o *Design* (projeto) se preocupa com detalhes de implementação de componentes individuais, a *Arquitetura* foca no "blueprint" global e nas restrições que guiarão todo o desenvolvimento.

Padrões Arquiteturais e Evolução



A escolha do padrão arquitetural impacta diretamente a capacidade de deploy e a performance do sistema. Historicamente, o modelo **Monolítico** (onde toda a aplicação reside em uma única unidade de execução) era o padrão pela sua simplicidade inicial de desenvolvimento e teste. No entanto, com a necessidade de escalabilidade horizontal e ciclos de entrega ultra-rápidos, surgiram os **Microserviços**. Neste modelo, a aplicação é decomposta em serviços pequenos, independentes e que se comunicam via rede (geralmente REST ou mensageria), permitindo que diferentes partes do sistema sejam atualizadas sem afetar o todo.

Arquitetura em Camadas (Layered): Organiza o código em níveis de responsabilidade (Apresentação, Negócio, Persistência), facilitando a separação de interesses.

Arquitetura Hexagonal (Ports and Adapters): Isola a lógica de negócio central de influências externas (bancos de dados, APIs de terceiros), permitindo que o núcleo do software seja testado de forma isolada e independente de tecnologias de infraestrutura.

Serverless e Event-Driven: Focam na execução de funções disparadas por eventos específicos, otimizando o custo computacional e a resposta a estímulos em tempo real.

Conceitos e Técnicas Fundamentais do Projeto

O projeto de software eficaz utiliza princípios que visam reduzir a complexidade cognitiva do código. Dois pilares fundamentais são o **Acoplamento** e a **Coesão**. O acoplamento mede o nível de dependência entre dois módulos; o ideal é que seja **baixo**, para que a mudança em um não quebre o outro. Já a coesão mede o quanto as responsabilidades dentro de um único módulo estão relacionadas; buscamos uma **alta** coesão, garantindo que cada componente faça “apenas uma coisa e a faça bem”.

CONHECIMENTOS ESPECÍFICOS - BLOCO II

QUALIDADE DE SOFTWARE: GARANTIA DA QUALIDADE DE SOFTWARE; GERÊNCIA DE CONFIGURAÇÃO DE SOFTWARE (GIT); ESTES DE SOFTWARE (UNITÁRIO, INTEGRAÇÃO, FUNCIONAL, ACEITAÇÃO, DESEMPENHO, CARGA, VULNERABILIDADE); TÉCNICAS PARA APLICAÇÃO DE TESTES DE SOFTWARE (CAIXA-BRANCA, CAIXA-PRETA, REGRESSÃO E NÃO FUNCIONAIS); FERRAMENTAS PARA AUTOMATIZAÇÃO DE TESTES; TÉCNICAS DE REFATORAÇÃO DE SOFTWA; TRATAMENTO DO DÉBITO TÉCNICO; MÉTRICAS DE QUALIDADE DE CÓDIGO; CODE SMELL; AUDITORIA DE SISTEMAS

A qualidade de software é um dos principais fatores que determinam o sucesso de um sistema. Um software de qualidade não é apenas aquele que "funciona", mas aquele que funciona corretamente, é seguro, eficiente, fácil de manter e atende às necessidades do usuário.

No desenvolvimento moderno, garantir qualidade não é uma etapa final, mas um processo contínuo que envolve planejamento, testes, revisão de código, organização de versões e boas práticas técnicas. Este capítulo apresenta os principais conceitos relacionados à qualidade de software, de forma clara e introdutória, permitindo que o aluno compreenda os fundamentos que sustentam sistemas confiáveis e bem estruturados.

Garantia da Qualidade de Software

A Garantia da Qualidade de Software (Quality Assurance – QA) é o conjunto de atividades planejadas para assegurar que o software atenda aos padrões estabelecidos.

Ela não se limita à fase de testes. Envolve todo o ciclo de desenvolvimento, desde o levantamento de requisitos até a manutenção.

Entre os principais objetivos da garantia da qualidade, destacam-se:

Prevenir defeitos antes que eles ocorram.

Garantir conformidade com padrões e requisitos.

Melhorar continuamente os processos de desenvolvimento.

Reduzir riscos e retrabalho.

A qualidade é construída ao longo do processo, e não apenas verificada no final.

Gerência de Configuração de Software (GIT)

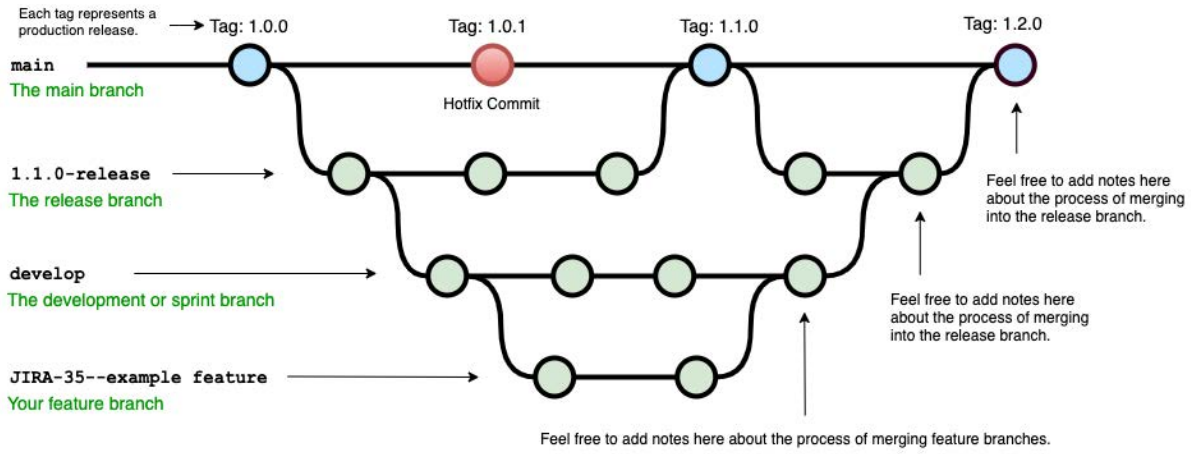
A Gerência de Configuração de Software é responsável por controlar e organizar as mudanças realizadas em um sistema ao longo do tempo.

Uma das ferramentas mais utilizadas para esse controle é o Git, sistema de versionamento que permite registrar alterações no código, manter histórico de versões e trabalhar em equipe de forma organizada.

Example Git Branching Diagrams

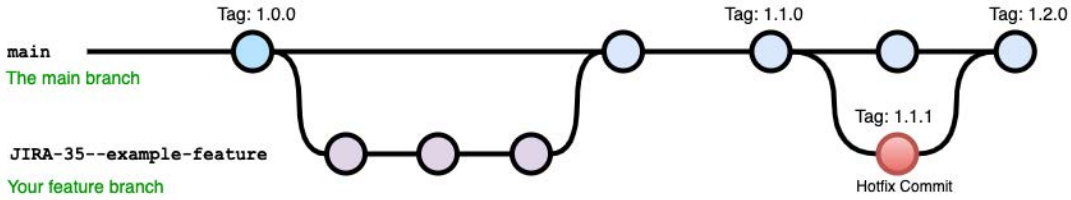
Example diagram for a workflow similar to "Git-flow" :

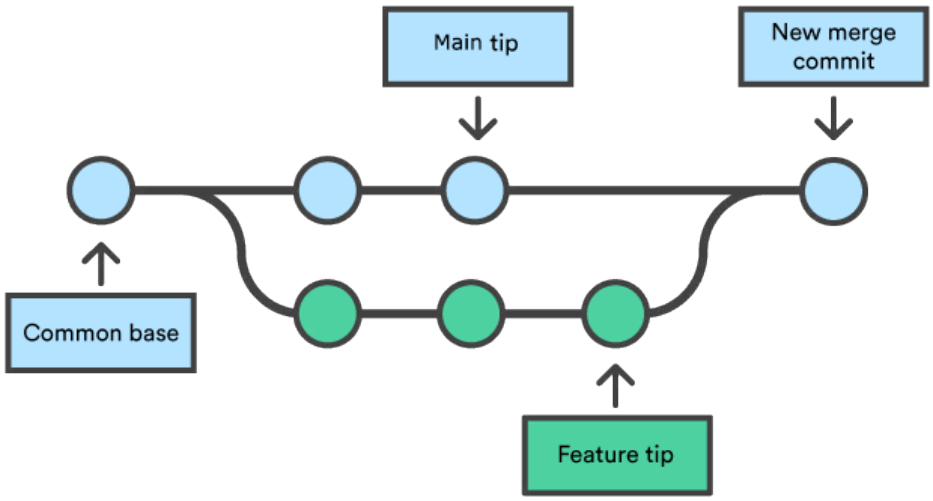
See: https://nvie.com/posts/a-successful-git-branching-model/



Example diagram for a workflow with a simpler branching model:

See: https://gist.github.com/jbenet/ee6c9ac48068889b0912 or https://www.endoflineblog.com/oneflow-a-git-branching-model-and-workflow





CONHECIMENTOS ESPECÍFICOS - BLOCO III

GERENCIAMENTO DE PRODUTOS DE SOFTWARE: GERENCIAMENTO DE PRODUTOS COM MÉTODOS ÁGEIS: SCRUM E KANBAN; MODELOS E TÉCNICAS DE GESTÃO DE PORTFÓLIO (SAFE): CARACTERÍSTICAS, OBJETIVOS, APLICABILIDADE E BENEFÍCIOS

AGILIDADE NO DNA DO PRODUTO: SCRUM E KANBAN

Historicamente, o software era desenvolvido no modelo "Cascata" (*Waterfall*), onde cada etapa (requisitos, design, código, testes) ocorria sequencialmente. O problema? Quando o software chegava ao cliente, meses depois, o mercado já havia mudado. Os métodos ágeis surgiram para quebrar esse ciclo, focando em entregas pequenas, frequentes e baseadas no feedback real dos usuários.

O Framework Scrum: Ritmo e Disciplina

O **Scrum** é um framework iterativo e incremental. Ele divide o trabalho em blocos de tempo fixos chamados **Sprints** (geralmente de 2 a 4 semanas). O coração do Scrum bate na previsibilidade e na melhoria contínua, estruturando-se em três pilares: transparência, inspeção e adaptação.

Papéis Fundamentais: O **Product Owner (PO)** define *o que* será feito, priorizando o valor de negócio no *Product Backlog*. O **Scrum Master** atua como um facilitador, removendo impedimentos e garantindo que o time siga os valores ágeis. Os **Desenvolvedores** são o time multidisciplinar que transforma ideias em software funcional.

A Dinâmica das Cerimônias: Tudo começa no *Sprint Planning*, onde o time decide o que cabe na próxima Sprint. Diariamente, a *Daily Scrum* alinha o progresso em 15 minutos. Ao final, a *Sprint Review* demonstra o que foi construído para os interessados, e a *Sprint Retrospective* analisa como o time pode trabalhar melhor no próximo ciclo.

O Método Kanban: Visualização e Fluxo Contínuo

Enquanto o Scrum é focado em "caixas de tempo" (Sprints), o **Kanban** é focado no **fluxo contínuo**. Originado no sistema de produção da Toyota, sua aplicação na TI foca em visualizar o trabalho e limitar a quantidade de tarefas em andamento para evitar gargalos.

O Quadro Kanban: O trabalho é representado visualmente em colunas (ex: *To Do, Doing, Testing, Done*). Isso permite que qualquer pessoa identifique instantaneamente onde o fluxo está travado.

WIP (Work in Progress): O segredo do Kanban é limitar o número de itens em cada coluna. Se o limite de "Testing" é 2 e já existem 2 tarefas lá, ninguém pode mover nada novo para essa coluna até que algo seja concluído. Isso força o time a focar em "terminar o que começou" em vez de "começar coisas novas", reduzindo o desperdício e aumentando a velocidade de entrega (*throughput*).

Scrum vs. Kanban: Qual escolher?

A escolha entre os dois depende da natureza do produto. O **Scrum** é excelente para produtos que precisam de um roteiro estruturado e onde o time pode se comprometer com metas de curto prazo. Já o **Kanban** brilha em ambientes de suporte, manutenção ou fluxos de trabalho muito dinâmicos, onde as prioridades mudam a cada hora e o objetivo é manter a "esteira" de produção sempre rodando sem interrupções artificiais.

Gestão de Portfólio e o Desafio da Escala (SAFe)

O gerenciamento de um único produto com um time de sete pessoas é um desafio tático; gerenciar um portfólio de dez produtos interdependentes com cinquenta times é um desafio estratégico. O **SAFe (Scaled Agile Framework)** surge como a resposta do mercado para organizações que precisam de governança, sincronização e previsibilidade sem perder a agilidade conquistada na base.

O que é o SAFe? Sincronizando o "Trem" Ágil

O SAFe é uma base de conhecimento de princípios e práticas comprovadas para aplicar Lean, Agile e DevOps em larga escala. Ele introduz o conceito de **Agile Release Train (ART)**, ou o "Trem de Liberação Ágil". Imagine que cada "vagão" desse trem é um time Scrum ou Kanban. Para que o produto final chegue à estação (o mercado) com sucesso, todos os vagões precisam rodar nos mesmos trilhos, na mesma velocidade e com a mesma direção.

Características e Objetivos: O foco do SAFe não é apenas "fazer software rápido", mas sim o **Alinhamento** e a **Transparência**. Ele busca garantir que o que o desenvolvedor está codificando hoje esteja diretamente ligado aos objetivos estratégicos que o CEO definiu para o ano.

Cadência e Sincronização: Diferente do Scrum individual, onde cada time pode ter sua própria duração de Sprint, no SAFe todos os times do "trem" operam na mesma cadência. Eles planejam juntos a cada 8 a 12 semanas em um evento massivo chamado **PI Planning (Program Increment Planning)**.

Níveis de Gestão e Aplicabilidade

O framework é modular, permitindo que a empresa escolha o nível de complexidade necessário:

Essential SAFe: O nível básico, focado na coordenação de vários times que trabalham no mesmo produto (o ART).

Portfolio SAFe: O nível mais alto, onde a diretoria decide onde investir o dinheiro da empresa. Aqui, a gestão não é sobre "tarefas", mas sobre **Épicos de Negócio** e fluxos de valor. O objetivo é garantir que o orçamento de TI esteja sendo gasto nas iniciativas que trarão o maior retorno sobre o investimento (ROI).

Benefícios e a Entrega de Valor

A grande vantagem do SAFe é a redução do "ruído" organizacional. Em empresas grandes e tradicionais, é comum que times trabalhem em funcionalidades que ninguém vai usar ou que entram em conflito com o trabalho de outro departamento.

Visibilidade: O SAFe torna as dependências visíveis. Se o Time A precisa de uma API do Time B para entregar sua parte, isso é mapeado visualmente meses antes, evitando atrasos críticos na data de lançamento.

Time-to-Market: Ao focar no fluxo de valor ponta a ponta (do conceito ao dinheiro), o framework ajuda a eliminar desperdícios burocráticos, permitindo que grandes corporações respondam a ataques de *startups* com agilidade competitiva.

Técnicas de Gestão de Portfólio e Métricas Ágeis

Ter uma lista de desejos (Backlog) infinita é fácil; o desafio real da TI é que a capacidade de desenvolvimento é sempre finita. A gestão de portfólio atua como o filtro estratégico que decide quais iniciativas receberão investimento, baseando-se no valor de entrega e no custo do atraso.

Priorização de Valor: A Técnica do WSJF

No modelo tradicional, quem gritava mais alto ou tinha o cargo mais alto (o famoso *HiPPO - Highest Paid Person's Opinion*) decidia a prioridade. No gerenciamento ágil de portfólio, utilizamos o **WSJF (Weighted Shortest Job First)**.

A Lógica do WSJF: Esta técnica calcula o "Custo do Atraso" (*Cost of Delay*) dividido pelo "Tamanho do Trabalho".

O Objetivo: Identificar tarefas que entregam um valor gigantesco em um tempo relativamente curto. Se uma funcionalidade pode trazer um milhão de reais em lucro e leva apenas duas semanas para ser feita, ela "fura a fila" de projetos de seis meses que trazem o mesmo retorno. O WSJF foca na economia do fluxo: o que nos dá o maior retorno com o menor esforço imediato?

Gestão de Dependências: O Mapa do Caminho

Em sistemas complexos (como um aplicativo de banco), o Time de Pagamentos não consegue entregar nada se o Time de Segurança não liberar o acesso aos dados. Gerenciar o portfólio é, em grande parte, gerenciar essas **interdependências**.

Program Board: No SAFe e em outras metodologias de escala, utiliza-se um quadro visual onde fios (físicos ou digitais) conectam as tarefas de diferentes times. Se o Time A atrasar, o "fio" mostra imediatamente que o Time B e o Time C também serão impactados. Isso permite que o Gestor de Produto antecipe crises e renegocie prazos antes que o problema chegue ao cliente final.

Agilidade no DNA do Produto: Scrum e Kanban

Historicamente, o software era desenvolvido no modelo "Cascata" (Waterfall), onde cada etapa (requisitos, design, código, testes) ocorria sequencialmente. O problema? Quando o software chegava ao cliente, meses depois, o mercado já havia mudado. Os métodos ágeis surgiram para quebrar esse ciclo, focando em entregas pequenas, frequentes e baseadas no feedback real dos usuários.

O Framework Scrum: Ritmo e Disciplina

O Scrum é um framework iterativo e incremental. Ele divide o trabalho em blocos de tempo fixos chamados Sprints (geralmente de 2 a 4 semanas). O coração do Scrum bate na previsibilidade e na melhoria contínua, estruturando-se em três pilares: transparência, inspeção e adaptação.

Papéis Fundamentais: O Product Owner (PO) define o que será feito, priorizando o valor de negócio no Product Backlog. O Scrum Master atua como um facilitador, removendo impedimentos e garantindo que o time siga os valores ágeis. Os Desenvolvedores são o time multidisciplinar que transforma ideias em software funcional.

A Dinâmica das Cerimônias: Tudo começa no Sprint Planning, onde o time decide o que cabe na próxima Sprint. Diariamente, a Daily Scrum alinha o progresso em 15 minutos. Ao final, a Sprint Review demonstra o que foi construído para os interessados, e a Sprint Retrospective analisa como o time pode trabalhar melhor no próximo ciclo.

O Método Kanban: Visualização e Fluxo Contínuo

Enquanto o Scrum é focado em "caixas de tempo" (Sprints), o Kanban é focado no fluxo contínuo. Originado no sistema de produção da Toyota, sua aplicação na TI foca em visualizar o trabalho e limitar a quantidade de tarefas em andamento para evitar gargalos.

O Quadro Kanban: O trabalho é representado visualmente em colunas (ex: To Do, Doing, Testing, Done). Isso permite que qualquer pessoa identifique instantaneamente onde o fluxo está travado.

WIP (Work in Progress): O segredo do Kanban é limitar o número de itens em cada coluna. Se o limite de "Testing" é 2 e já existem 2 tarefas lá, ninguém pode mover nada novo para essa coluna até que algo seja concluído. Isso força o time a focar em "terminar o que começou" em vez de "começar coisas novas", reduzindo o desperdício e aumentando a velocidade de entrega (throughput).

Scrum vs. Kanban: Qual escolher?

A escolha entre os dois depende da natureza do produto. O Scrum é excelente para produtos que precisam de um roteiro estruturado e onde o time pode se comprometer com metas de curto prazo. Já o Kanban brilha em ambientes de suporte, manutenção ou fluxos de trabalho muito dinâmicos, onde as prioridades mudam a cada hora e o objetivo é manter a "esteira" de produção sempre rodando sem interrupções artificiais.

Gestão de Portfólio e o Desafio da Escala (SAFe)

O gerenciamento de um único produto com um time de sete pessoas é um desafio tático; gerenciar um portfólio de dez produtos interdependentes com cinquenta times é um desafio estratégico. O SAFe (Scaled Agile Framework) surge como a resposta do mercado para organizações que precisam de governança, sincronização e previsibilidade sem perder a agilidade conquistada na base.