METODOLOGIAS

São processos, práticas e regras orientando a forma de que vamos desenvolver o software

Envolvendo:

Para que usar essas metodologias:

* Maior controle sobre o projeto
* Redução de riscos
* Melhor comunicação entre cliente e equipe
* Padronização de processos

Metodologias clássicas: São baseados em planejamentos detalhados e execução sequencial

* Documentação extensa
* Pouca flexibilidade
* Segue caminho linear

Cascata(waterfall)

É dividida em várias fases e para outro ser iniciada depende da conclusão da anterior, sendo assim uma metodologia simples de entender mas com pouca flexibilidade sobre suas tarefas.

Modelo incremental

Ele trabalha em módulos, onde cada um deles possui fases de planejamento, execução e fim, e a entrega busca resultados funcionais.

Sua vantagem são as entregas parciais porem a integração desses módulos são mais complexas

Modelo espiral

Ele traz mais flexibilidade para o projeto, o processo é desenhado como uma espiral e cada volta dela representa uma iteração, sendo realizados planejamento, análise de riscos, desenvolvimento, teste e avaliação do cliente.

Espiral X Incremental

Modelo V

Ele é uma evolução do modelo cascata e serve para detectar uma detecção tardia de erros. Ele introduz uma correspondência direta entre cada etapa de desenvolvimento e uma etapa de teste associada.

Assim a verificação e validação acontecessem paralelos a etapa de desenvolvimento.

Conclusão: Não existe uma metodologia melhor do que a outra mas sim a que mais se encaixa no contexto proposto.

Metodologias Ágeis

Surgiram como resposta das metodologias tradicionais de desenvolvimento de software.

Caracterizada por ciclos curtos de desenvolvimento, feedback continuo, adaptabilidade a mudanças e entrega incremental de valor.

Manifesto Ágil: É um conjunto de valores e princípios para tornar o desenvolvimento de software mais flexível, colaborativo e centrado no cliente

Os principais valores envolvem:

* Indivíduos e interações
* Software funcionando
* Colaboração com o cliente

SCRUM

É uma estrutura ágil que organiza o trabalho em ciclos chamados de Sprints. Sendo divididos em 3 pilares principais: Papéis principais, Artefatos e eventos.

Papéis principais:  
\* Product Owner : define e prioriza os requisitos   
\* Scrum Master : facilita o processo e remove impedimentos   
\* Development team : Desenvolve o produto

Eventos: momentos regulares de planejamento, inspeção e adaptação  
\* Sprint Planning : Planejamento da sprint  
\* Sprint : ciclo de trabalho, geralmente 1 a 4 semanas   
\* Daily Scrum : reunião diária de sincronização  
\* Sprint review:  
\* Sprint retrospectiva:

Artefatos: Itens que registram e organizam o trabalho  
\* Product backlog: lista de tudo que precisa ser feito e priorizado  
\* Sprint Backlog: lista de tarefas selecionadas para sprint atual  
\* Incremento: resultado final da Sprint, pronto para uso

KANBAM  
Gerenciar seu fluxo de trabalho de forma mais visual, com quadros e colunas representando cada parte do processo

Princípios fundamentais:

* Visualizar o trabalho
* Limitar o trabalho em progresso
* Gerenciar o fluxo

EX (EXTREME PROGRAMMING)

Focada em qualidade do código e resposta rápida a mudanças, criada em contextos que ocorrem mudanças constantes.

Pincípios fundamentais:

* Comunicação consistente
* Feedback rápido sobre o software desenvolvido

Práticas seguidas por ela:

* Programação em par (Pair programming)
* Testes automatizados: testes são escritos antes do código
* Integração contínua: automatizar e testar o código várias vezes
* Refatoração : melhorar continuamente o código sem alterar a funcionalidade
* Propriedade coletiva: Qualquer desenvolvedor pode alterar qualquer parte