Resumo Gerenciamento de Software

Gerenciamento de Software

- Envolve o planejamento de processos, pessoas e eventos.
- Possui 4 P's que o influencia: Pessoas, Produto, Processos e Projeto.
- Possui 3 pilares básicos: Custo, escopo, prazo com base na qualidade.

PMI – Gerenciamento de Projetos

- Entidade internacional sem fins lucrativos que congrega profissionais da área de gestão de projetos.
- Objetivo: promover e difundir a gestão de projetos pelo mundo.
- Criou o PMBOK.

PMBOK

- Guia de padronização dos termos.
- Boas práticas.
- Resume os processos de gerência em: iniciação, planejamento, execução, controle e finalização.
- Áreas de conhecimento que compõe: qualidade, recursos humanos, escopo, tempo, custo, integração, comunicação, aquisições e riscos.

Agile Project Management

- Agilidade e flexibilidade nas comunicações e desenvolvimento de software a curto período de tempo.
- Gerenciamento ágil de projetos: projetos simples, ágeis e interativos.
- 12 princípios: satisfazer o cliente com entrega adiantada; aceitar alterações do cliente; entrega de software em funcionamento com frequência; equipe de devs e comercial trabalhando em equipe; equipe motivada; conversa aberta; medida do progresso é o software funcionando; desenvolvimento sustentável; excelência técnica aumenta agilidade; simplicidade; equipes auto-organizáveis e avaliação da equipe em intervalos regulares.
- Práticas do gerenciamento:
  - **Pessoas:** bem treinadas e motivadas;
  - **Equipes:** possui alguns paradigmas. Fechado – autoridade tradicional; Randômico – estrutura vaga e depende da iniciativa de cada membro; Aberto – fechado + randômico; Sincronizado – organiza os membros a trabalharem partes do problema.
  - **Modelagem:** ajuda a visualizar o sistema, especificar estrutura, serve de guia e documenta.
  - **Diagramas:** representa um conjunto de informações.
  - **Cronograma:** distribui o esforço estimado e aloca esforço para tarefas específicas.

- **Métricas:** medição para obter controle. Prazo, tempo, segurança, qualidade, custos, confiabilidade e recursos.

Modelos de Desenvolvimento de Software

1) Processo Unificado (UP)
   - Metodologia de gerencia de projeto que usa UML.
   - Desenvolvimento iterativo.
   - Arquitetura baseada em componentes.
   - Pode ser usado com outras metodologias.
   - Fases:
     - **Concepção:** Comunicação com cliente atividade de planejamento.
     - **Elaboração**: Comunicação e modelagem do processo.
     - **Construção:** Construção do sistema.
     - **Transição:** Transferência do sistema ao usuário para teste beta + relatório de feedback do usuário.
     - **Produção:** Monitoramento do software e suporte, foco nos defeitos e mudanças.
   - Disciplinas: Modelagem de Negócios; Requisitos; Análise e Designer; Implementação; Testes; Implantação; Ambiente; Configuração e Gerência de mudança; Gerenciamento de Projeto.

2) Extreme Programming (XP)
   - Orientada a objetos.
   - Projeto é conduzido com base nos requisitos que se modificam rapidamente.
   - Atender o cliente com mais rapidez, qualidade e de forma simples.
   - Equipe até 12 devs.
   - Modelo incremental.
   - Desenvolvimento guiado por testes.
   - Flexível e adaptativa.
   - Valores e princípios: comunicação, feedback, simplicidade, coragem e respeito.
   - Praticas da XP: Padrão de código de desenvolvimento, design simples, cliente presente, jogo de planejamento, stand up meeting, programação em pares, refatoração, desenvolvimento orientado por testes, código coletivo, metáfora, ritmo sustentável, integração contínua, releases curtos.
   - Testes conhecidos como Test-First ou Test-Last. Test-First: cada nova história de usuário é escrito um teste, onde é executado apenas a nova funcionalidade. O novo código é aceito se passar pelo teste. Test-Last: teste escrito depois da implementação do código.
   - Usa o TDD.

3) Feature Driven Development (FDD)
   - Metodologia ágil robusta.
   - Orientada a funcionalidades.

- Construída com pequenos blocos de funcionalidades que chamamos de features.
- Enfatiza qualidade.
- Modelar o domínio do problema antes do início do projeto.
- Rastreabilidade e relatórios mais precisos.
- Orientada a objetos.
- Iterativo.
- Envolvidos no projeto: apreciam, pois provê os resultados significados mais cedo e acompanha a evolução do projeto.
- Devs: apreciam, pois há regras e técnicas + fáceis de entender e resultados + rápidos.
- Processos:
  - Concepção e planejamento: 1 a 2 semanas.
  - Construção: forma iterativa. 2 semanas.
- Etapas: Desenvolver um modelo abrangente(DMA); Construir a Lista de Features(CLF); Planejar por Features(PPF); Detalhar por features(DPF); Construir por features(CPF).

4) Dynamic Systems Development Methodology (DSDM)
   - Centrada em estabelecer recursos e o tempo fixo.
   - Iterativa e incremental.
   - Prototipagem.
   - Timeboxing: encapsulamento do tempo reservado para desenvolvimento das funcionalidades.
   - Ajusta suas funcionalidades de maneira a atender os prazos.
   - 9 princípios: Participação ativa de usuários e stakeholders, abordagem cooperativa e compartilhada, equipes com poder de decisão, entregas contínuas, desenvol. Iterativo e incremental, feedback, possíveis alterações durante o desenvolvimento devem ser reversíveis, fixar requisitos essenciais e teste em todo ciclo de vida.
   - Fases: Estudo da viabilidade, estudo de negócio, modelo de iteração funcional, projeto e construção de iteração e implementação.
   - Equipe: os papéis podem ser acumulados entre os envolvidos. Podendo existir várias equipes pequenas, sendo que em cada uma de duas pessoas deve existir pelo menos um usuário e um colaborador.

5) Iconix Process
   - Modelagem dirigida por casos de uso.
   - Objetivo: estudar e comunicar o comportamento do sistema sob ponto de vista do usuário final.
   - Usa 3 diagramas da UML: casos de uso, classe e sequência.
   - Usa também o diagrama de robustez: híbrido do diagrama de classes e atividades.
   - Iterativo e incremental. Possui 3 estereótipos: fronteria(boundary), entidade(entity) e controle(control).

- Minimiza paralisia de análise.
- Rastreamento da análise à implementação.
- Metodologia prática e simples.
- Dividida em 2 visões:
  - Dinâmica: aspectos comportamentais do software. Interação do usuário com o sistema. Artefatos: Diag. De caso de uso, diag. De robustez e sequência.
  - Estática: aspectos estruturais. Sem interação com o usuário. Artefatos: Modelo de domínio e diagrama de classe.
- Estrutura: Análise de requisitos, análise e desenho preliminar, desenho, implementação.

6) SCRUM
   - Construção de software incrementalmente em ambientes complexos.
   - Requisitos não são claros ou que muda com frequência.
   - Não pode ocorrer nenhuma mudança durante uma Sprint.
   - O produto é projetado, codificado e teste durante uma Sprint.
   - Possui 3 papéis: Product Owner, Scrummaster e Team.
   - Cerimônias: Daily meeting ou daily scrum, sprint review, sprint planning, sprint retrospective.
   - Artefatos gerados das cerimônias: product backlog, sprint backlog, burndown chart.
   - Princípio básico: os clientes podem mudar de ideia durante o desenvolvimento do projeto.

   - Papéis:
     - **Product Owner**: representa o cliente. Moderador entre o cliente e o team. Responsável por: levantar requisitos, colaborar com o scrum máster e team para planejar sprints, guiar equipe, acompanhar o desenvolvimento e decidir datas de lançamento.
     - **Scrum Master**: Papel de facilitador. Protege o team do product owner. Guardião do processo do scrum. Responsável por: guiar e treinar o team, organizer a retrospective de sprint, garantir a colaboração entre as partes interessadas e o team.
     - **Team**: é quem desenvolve o projeto. Responsável por: desenvolver o projeto e demonstrar os resultados dos sprints para o product owner, realizar estimativas, transformar o product backlog em tarefas, apresentar o produto ao cliente, deve ser auto-organizável e multidisciplinar.

   - Cerimônias:
     - **Sprint Planning:** primeira reunião. Todos participam. Máximo de 8 horas. Product Owner planeja e elabora as prioridades com base na história do usuário e equipe de dev define Sprint backlog.

- **Daily Meeting**: Participa team e scrum master. No máximo 15 mins. Respondem as seguintes perguntas: O que você fez ontem? O que você fará hoje? Quais obstáculos que impedem seu trabalho?
- **Sprint Review**: Reunião de balanço sobre tudo o que foi feito na Sprint. Time mostra o resultado da Sprint ao Product Owner e convidados. Máx. 4 horas. Ao final da Sprint.
- **Sprint Retrospective:** Verifica o que foi bom e o que pode melhorar. Participa o team e o scrum máster, o product owner pode participar. Após a Sprint review. Máx. 3 horas.

- Artefatos:
  - **Product Backlog**: contém os itens que devem ser desenvolvidos durante o projeto. Lista de funcionalidade. Descritos de forma simples e fácil.
  - **Sprint Backlog**: Proveniente do planejamento da Sprint. Tarefas que devem ser desenvolvidas durante uma Sprint ou iteração.
  - **Burndown Chart**: mostra o esforço restante para a conclusão da iteração, bem como mostrar o quão próximo ou distante o time está de atingir a meta.

7) <u>Openup – Processo Unificado Aberto</u>
   - Iterativa e incremental.
   - Qualidade do RUP, agilidade do XP e melhores práticas de gerenciamento do SCRUM.
   - Foca no desenvolvimento colaborativo.
   - Valorização da equipe.
   - 4 grandes áreas: comunicação e colaboração, objetivo, solução e gerência.
   - Disciplinas: Análise e projeto, gerência de configuração e mudança, implementação, gerência de projeto, requisitos e test.
   - 7 papéis:
     - **Stakerholder:** se comunica e colabora com o management e com analistas.
     - **Analista:** coleta infos com o stakerholder e tem foco na intenção.
     - **Arquiteto:** responsável pela arquitetura, design e codificação do projeto. Comunica com management e devs. Foco na intenção.
     - **Devs:** escrever códigos, testes e integração de componentes. Foco é solução.
     - **Testador:** teste de código. Comunica com analista e devs.
     - **Management:** liderar, planejar, coordenar os stakes e time.
     - **Qualquer**: qualquer membro pode realizar tarefas gerais.
   - 4 princípios: igualar prioridades para maximizar benefícios aos stakeholders, colaboração constante, foco na arquitetura para redução de riscos, evolução contínua com feedbacks.
   - Ciclo de vida: em todo ele temos o artefato Plano de Projeto que possui todas as datas de atualizações e fases a serem seguidas até o final. As iterações mantém a equipe focada na entrega de valores para o cliente.

8) Metodologia Crystal

- Foco na comunicação e interações entre as pessoas.
- Mais objetivo.
- Código genérico comum que atende a diferentes tipos e tamanhos de projetos.
- Foco no talento e habilidades dos envolvidos.
- Dividida em cores:
    - **Clear** – 1 a 6 devs. Falha: Perdem dinheiro, mas recuperam facilmente.
    - **Yellow** – 7 a 20 devs. Falha: Perdem dinheiro discretamente.
    - **Orange-** 21 a 40 devs. Falha: Perdem dinheiro substancialmente.
    - **Red-** 41 a 100 devs. Falha: Há perda substancial de dinheiro e possivelmente vidas humanas.
- Pode ser classificado em: criticidade e tamanho da equipe.
- Princípios: Trabalho direto com o cliente, maior complexidade tem maior custo, equipes maiores pedem metodologia diferenciada, muita cerimônia gera maior criticidade, comunicação eficiente, habilidade em lidar com pessoas e eficiência do desenvolvimento.
- Propriedades: Entregas frequentes, melhoria reflexiva, comunicação intensa, segurança pessoal, foco, acesso fácil a usuários experientes e integração contínua.
- Equipe: stakeholder, coordenador de projeto, analista de negócio, usuário stakeholder, designer, programador, testador e redator.
- Ciclo de vida:
    - Encenação: planejamento do próximo incremento do sistema.
    - Construção, demonstração e revisão: editado e revisado os objetivos do incremento.
    - Monitoramento do processo: monitorado levando em consideração o progresso e estabilidade da equipe.
    - Paralelismo e fluxo: diferentes equipes operando.
    - Inspeções de usuários: inspeções feitas por usuário a cada incremento.
    - Workshops refletivos: reuniões antes e depois das iterações.
    - Local matters: procedimentos aplicados.
    - Produtos de trabalho;
    - Padrões: de código, convenção de produto.
    - Ferramentas.

9) Test Driven Development (TDD)

- Derivada da XP.
- Antecipar a identificação e a correção de erros durante o desenvolvimento.
- Pequenas iterações que começam pela implementação de um caso de teste.
- Baseia em escrever testes automatizados para a funcionalidade entes de ser implementada.
- Ciclo red-green-refactor.
- Devem seguir o modelo FIRST:
    - **F**ast: testes rápidos em apenas uma unidade.

- **I**solated: testes unitários isolados.
- **R**epeated: teste de repetição para avaliar o comportamento.
- **S**elf-verifying: para verificar se passou ou deu falha.
- **T**imely: um por unidade.

10) Lean Software Development (LD)

- Foca na eliminação de desperdício, excelência na qualidade e aumento de velocidade de processo.
- Fornecer algo com valor aos clientes, custos mais baixos e melhoria constante do processo.
- 7 princípios: Eliminar desperdícios, amplificar aprendizado, decidir o mais tarde possível, entregar o mais rápido possível, dê poder a equipe, construir com integridade e ver o todo.

11) KANBAN

- Ferramenta visual que ajuda no acompanhamento do fluxo de trabalho e controle do WIP (work in progress).
- Método de gestão de mudanças.
- Sem muitas regras.
- Princípios: visualizar fluxo de trabalho, acompanhar o software do início ao fim, visualizar e limitar a quantidade de trabalho em casa fase, lembrar das políticas que estão sendo seguidas, medir e gerenciar fluxo de trabalho e identificar melhorias que podem ser feitas.
- Auxilia a equipe a assimilar e controlar o progresso das tarefas de forma visual.
- Usa quadro em branco e post-it.
- Ao término de cada tarefa o papel é empurrado para a etapa seguinte.

Metodologias Tradicionais X Metodologias Ágeis

Aspecto	Metodologia Tradicional	Metodologia Ágil
Objetivo	Orientado por atividades e foco nos processos	Orientado por produto e foco nas pessoas
Projeto	Estáveis e inflexíveis	Adaptável
Tamanho	Qualquer tamanho	Pequenos, mas pode ser usado em projetos de maior porte
Gerente de Projeto	Controle total	Papel de facilitador ou coordenador
Equipe	Atuação com papéis claros e bem definidos em todas as atividades	Atuação colaborativa em todas as atividades
Cliente	Participa nas fases iniciais de levantamento de requisito	Parte integrante da equipe
Planejamento	Detalhado e os envolvidos não participam	Curto e com participação de todos envolvidos
Tamanho da equipe	Grande número de envolvidos e estão em locais distantes	Reduzido e trabalham no mesmo local

PS: Softwares Críticos!

Softwares críticos, cuja precisão, risco e confiabilidade são decisivos, não podemos dispensar as recomendações que são impostas pelas regras e normas das metodologias tradicionais, principalmente em relação a documentação.

Estimativa para Desenvolvimento Ágil

- Fazemos antes de iniciarmos um projeto.
- Estimativa do trabalho, recursos necessários e período de tempo para concluir o projeto.
- Estabelecer cronograma.
- Depende de vários fatores que devem ser analisados: complexidade do projeto, tamanho e grau de incerteza estrutural.
- A estimativa é feita através de decomposição: Cada cenário de usuário é separado para fins de estimativa. O cenário é decomposto em várias tarefas a serem desenvolvidas. Cada tarefa e estimada separadamente. Estimativas de cada tarefa são somadas. Estimativas de todos os cenários são somados para desenvolver a estimativa para o incremento.
- Propósito: Garantir o número de cenários a incluir no incremento esteja de acordo com os recursos disponíveis e estabelecer uma base para a alocação de esforço a medida que o incremento é desenvolvido.
- Estimativa de custo deve se basear na funcionalidade do software.

Gerenciamento de Risco em Projetos Ágeis

- Existem riscos que sempre se transformam em perdas.
- Três componentes básicos: O fato ou evento que caracterizam o possível risco, probabilidade de que o fato ou evento realmente venha acontecer, impacto financeiro se caso o fato ou evento tenha acontecido.
- Tipos de riscos:
    - **Riscos de Projeto:** Ameaçam o plano do projeto.
    - **Riscos Técnicos:** Ameaçam a qualidade e a data de entrega.
    - **Riscos de Negócio:** Ameaçam a viabilidade do software, o projeto ou produto.