Programação Extrema (XP)

Visão Geral

- Equipe Coesa
- Histórias de Usuário
- Ciclos Curtos e Planejamento de Iterações
- Testes de Aceitação
- Programação em Pares
- Desenvolvimento Guiado por Testes (TDD)
- Posse Coletiva
- Integração Contínua
- Ritmo Sustentável
- Ambiente Aberto
- Jogo de Planejamento

Importância da colaboração

- Equipe verdadeiramente integrada
  - Composta por clientes, gerentes e desenvolvedores
- Colaboração essencial
  - Todos compreendem os desafios
  - Trabalham juntos em soluções eficazes

Papel do cliente na equipe

Definição de Cliente

- Pode ser um analista de negócios, especialista em qualidade, marketing ou financiador do projeto

Importância da Presença do Cliente

- Cliente deve ser parte ativa da equipe
- Presença e acessibilidade são cruciais

Trabalho em Proximidade

- Idealmente, cliente trabalha na mesma sala dos desenvolvedores
- A distância compromete a comunicação e a coesão

Solução para Presença Física Inviável

- Nomear um representante com conhecimento e autoridade
- Representante atua como cliente substituto

Definição e uso de histórias de usuário

- Histórias de usuário como base
  - Unidade básica de planejamento
  - Representam conversas contínuas
- Anotações concisas
  - Facilitam comunicação entre cliente e desenvolvedores
- Evitar detalhamento precoce
  - Requisitos evoluem com o desenvolvimento do sistema

Estimativas e planejamento

- Estimativas de Esforço
  - Baseadas em conversas com o cliente
- Planejamento de Iterações
  - Utiliza as estimativas fornecidas
- Planejamento de Entregas
  - Guiado pelas estimativas de esforço

Duração e resultados das iterações

Operação em ciclos curtos

Iterações de aproximadamente duas semanas

Resultados de cada iteração

Software funcional

Demonstrado aos stakeholders

Feedback contínuo

Orçamento e escolha de histórias

- Definição do Orçamento
  - Baseado na produtividade da iteração anterior
- Escolha das Histórias
  - Cliente escolhe as histórias a serem implementadas
  - Respeito ao orçamento definido
- Proteção do Escopo
  - Escopo protegido contra mudanças durante a iteração
  - Assegura foco e estabilidade

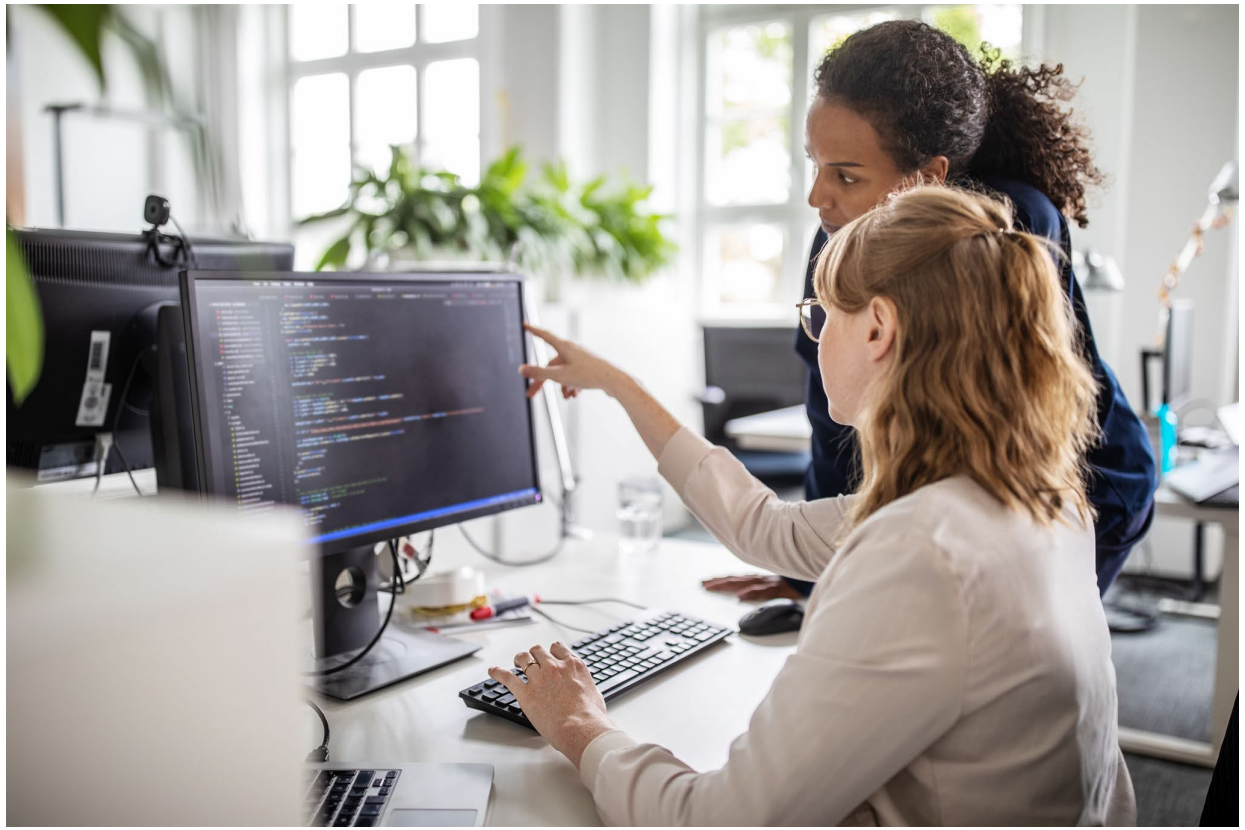
Planejamento de entregas

- Planejamento de entregas
  - Conjunto maior de histórias
  - Entregas após algumas iterações
- Plano flexível
  - Histórias podem ser incluídas
  - Histórias podem ser removidas
  - Histórias podem ser reordenadas
  - Conforme evolução do projeto

Testes de Aceitação

- Definição dos Critérios de Aceitação
  - Escritos em linguagem de fácil compreensão
  - Compreensíveis para todas as partes interessadas
- Validação Automática do Sistema
  - Testes validam automaticamente o comportamento do sistema
  - Garantem que o sistema se comporta conforme o esperado
- Documentação Viva do Sistema
  - Testes funcionam como documentação contínua
  - Atualizam-se conforme o sistema evolui

Dinâmica da programação em pares



- Escrita de código em dupla
  - Um desenvolvedor digita enquanto o outro revisa
  - Sugestões de melhorias são feitas constantemente
- Troca de papéis
  - Promove engajamento mútuo
  - Facilita aprendizado cruzado
  - Disseminação de conhecimento

Rotatividade e posse coletiva do código

Rotatividade de Duplas

Garantir que todos trabalhem com todos

Explorar diferentes partes do sistema

Fortalecimento da posse coletiva do código

Evitar dependência de especialistas isolados

Desenvolvimento Guiado por Testes (TDD)

- Escrita de Teste de Unidade
  - Teste de unidade é escrito antes do código funcional
  - Inicialmente, o teste falha
- Desenvolvimento do Código
  - Escreve-se o código mínimo necessário
  - Objetivo é fazer o teste passar
- Benefícios do Ciclo Rápido
  - Garante código testável
  - Promove modularidade
  - Assegura alta qualidade do código

Posse Coletiva

- Propriedade Compartilhada
  - Não há propriedade exclusiva sobre módulos
  - Qualquer par de desenvolvedores pode modificar qualquer parte do sistema
- Testes Automatizados
  - Viabilizam modificações seguras
- Programação em Pares
  - Facilita colaboração e revisão de código
- Integração Contínua
  - Permite integração frequente de alterações

Integração Contínua

- Integração de código várias vezes ao dia
  - Garantia de execução bem-sucedida de todos os testes antes da integração
- Uso de sistemas de controle de versão sem bloqueio
  - Facilita o trabalho colaborativo sem interrupções
- Prática de registros frequentes
  - Evita conflitos complexos de mesclagem

Ritmo Sustentável

Proibição de horas extras contínuas

Evita sobrecarga de trabalho

Promove um ambiente de trabalho saudável

Planejamento de trabalho sustentável

Ritmo de trabalho consistente

Garantia de produtividade ao longo do tempo

Ambiente Aberto

Configuração do Espaço

- Estações de trabalho compartilhadas
- Ambiente físico aberto

Vantagens

- Favorece a comunicação
- Facilita a colaboração
- Resolução rápida de problemas

Impacto na Produtividade

- Estudos indicam aumento significativo
- Possibilidade de dobrar a produtividade

Jogo de Planejamento

Colaboração entre Clientes e Desenvolvedores

- Clientes e desenvolvedores trabalham juntos no planejamento
- Foco na comunicação e entendimento mútuo

Priorização de Histórias pelo Valor de Negócio

- Clientes determinam a importância das histórias
- Histórias são priorizadas com base no valor que agregam ao negócio

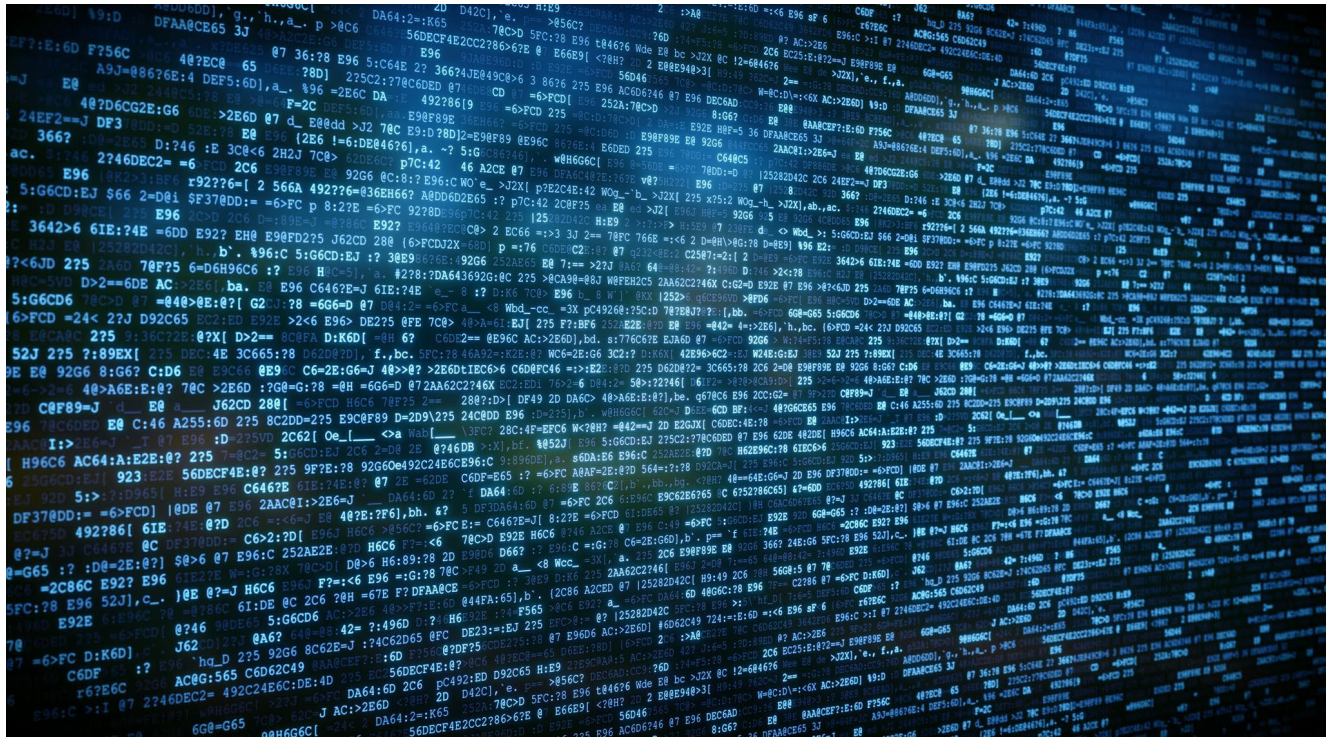
Estimativa de Custo de Implementação

- Desenvolvedores avaliam o esforço necessário para implementar cada história
- Estimativas ajudam no planejamento e alocação de recursos

Alinhamento entre Valor e Esforço

- Escopo de cada iteração é definido pelo equilíbrio entre valor e esforço
- Objetivo é maximizar o valor entregue em cada iteração

Projeto Simples



- Solução Simples
  - Atende às necessidades atuais
  - Infraestrutura e complexidade adicionadas apenas quando justificadas
- Princípio YAGNI
  - Você não vai precisar disso
  - Evita duplicação de código
- Abstrações e Refatoração
  - Incentiva abstrações
  - Refatoração contínua

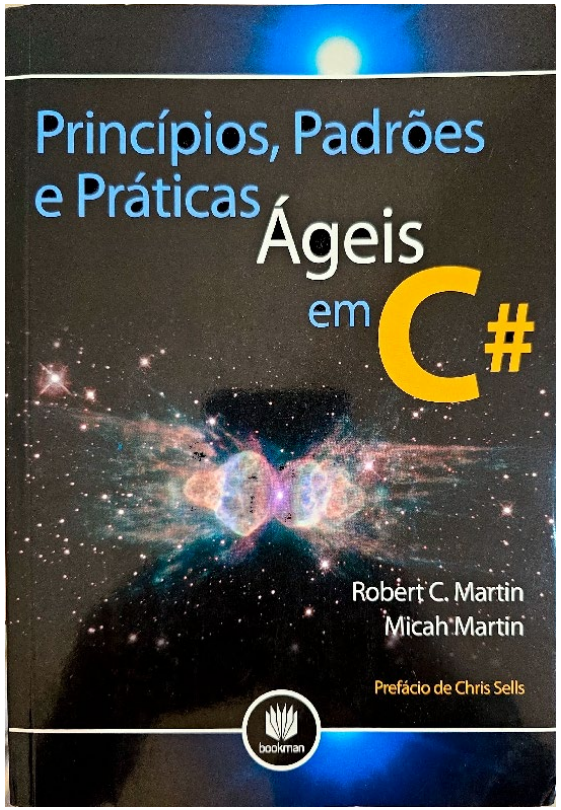
Refatoração

- Melhoria do Design Interno
  - Refatoração melhora a estrutura interna do sistema
  - Não altera o comportamento externo
- Frequência da Refatoração
  - Deve ser realizada frequentemente
  - Garante código limpo e compreensível
- Facilidade de Modificação
  - Refatoração facilita futuras modificações
- Validação da Estabilidade
  - Testes são executados após cada refatoração
  - Assegura que o sistema permanece estável

Metáfora

- Visão compartilhada
  - Guia o design e a nomeação dos elementos
  - Fornece uma analogia compreensível
- Alinhamento dos desenvolvedores
  - Estrutura do sistema
  - Funcionamento do sistema
- Prática intangível
  - Coesão ao projeto
  - Clareza ao projeto

Conclusão



- Colaboração Intensa
  - Promove trabalho em equipe
  - Facilita comunicação contínua
- Qualidade Técnica
  - Enfatiza boas práticas de codificação
  - Garantia de software robusto
- Adaptabilidade
  - Flexível conforme o contexto do projeto
  - Permite ajustes conforme necessidades
- Processo Disciplinado
  - Transforma desenvolvimento de software
  - Leve e eficaz