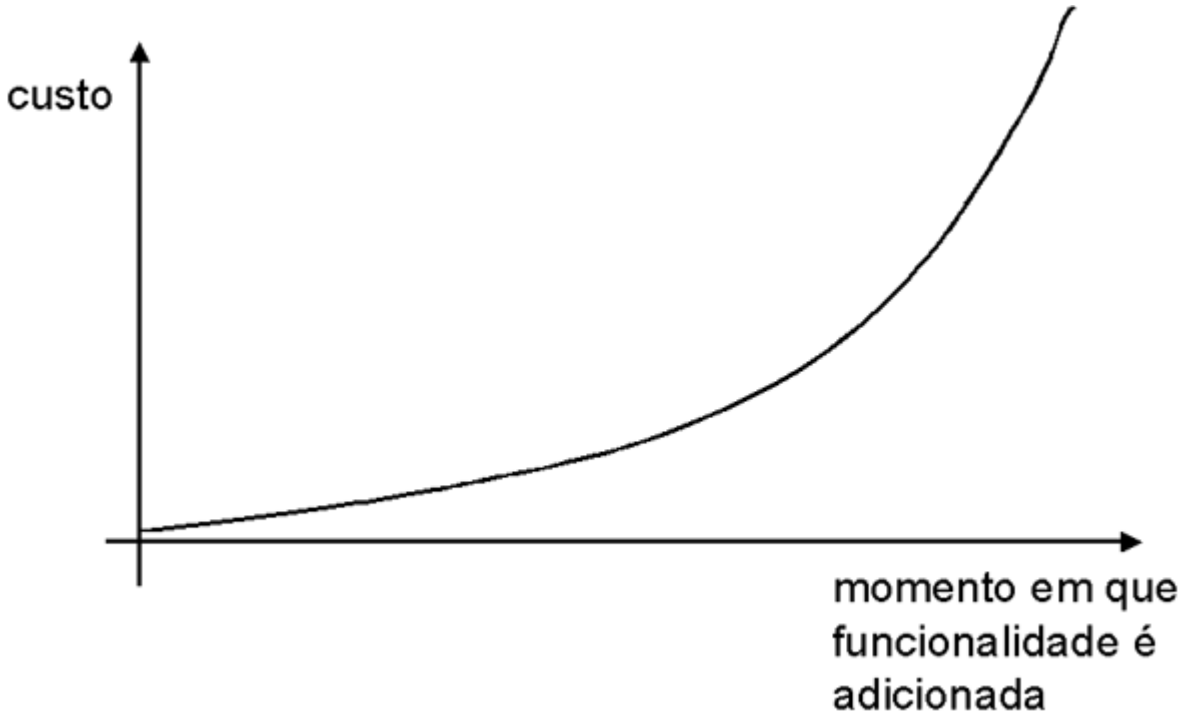
# METODOLOGIAS ÁGEIS

## O QUE É UMA METODOLOGIA ÁGIL?

DESENVOLVIMENTO AD-HOC DE SOFTWARE EM GERAL PRODUZ RESULTADOS MUITO RUINS. ESPECIALMENTE EM SISTEMAS GRANDES!

ENGENHARIAS TRADICIONAIS COLOCAM GRANDE ÊNFASE EM PROJETAR ANTES DE CONSTRUIR.

VISÃO TRADICIONAL DA ENGENHARIA DE SOFTWARE NOS MOSTRA QUE:




QUEREMOS AGORA PODER ALTERAR SOFTWARE:

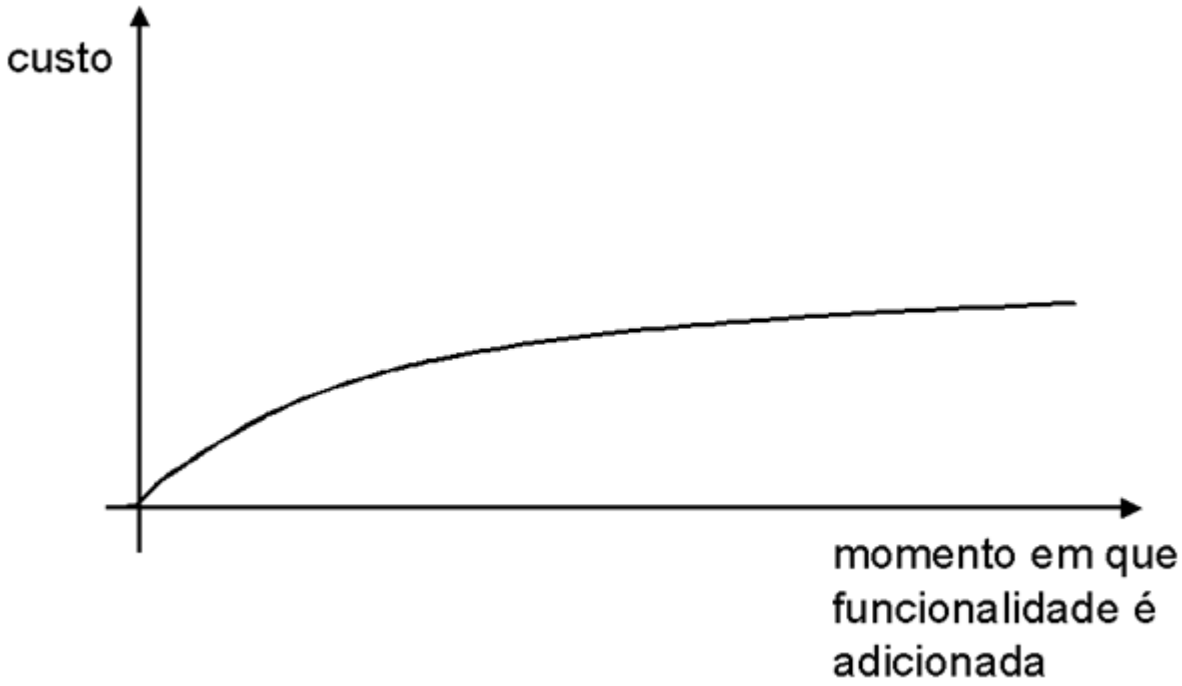
- NO INICIO DO PROJETO, NORMALMENTE NÃO SE SABE PRECISAMENTE O QUE SE QUER
- SOFTWARE EVOLUI PARA ATENDER AO NEGÓCIO
- SOFTWARE NUNCA FICA "PRONTO"

PRECISAMOS PARAR DE TENTAR EVITAR MUDANÇAS

MUDANÇAS SÃO UM ASPECTO INTRÍNSECO DA VIDA DO SOFTWARE

PRECISAMOS DE UMA METODOLOGIA DE DESENVOLVIMENTO QUE NOS PERMITA ALTERAR CONSTANTEMENTE O CÓDIGO SEM COMPROMETER SUA QUALIDADE

QUEREMOS ENXERGAR UMA VISÃO DIFERENTE DA ENGENHARIA DE SOFTWARE:




MÉTODOS ÁGEIS (AM) SÃO UMA COLEÇÃO DE METODOLOGIAS BASEADA NA PRÁTICA PARA MODELAGEM EFETIVA DE SISTEMAS BASEADOS EM SOFTWARE.

É UMA FILOSOFIA ONDE MUITAS METODOLOGIAS SE ENCAIXAM.

AS METODOLOGIAS ÁGEIS APLICAM UMA COLEÇÃO DE PRÁTICAS, GUIADAS POR PRINCÍPIOS E VALORES QUE PODEM SER APLICADOS POR PROFISSIONAIS DE SOFTWARE NO DIA A DIA.

NOS ÚLTIMOS 10 (DEZ) ANOS, METOLOGIAS ÁGEIS TEM SURGIDO. TEVE COMO PRINCIPAL MOTIVAÇÃO CRIAR ALTERNATIVAS PARA O MODELO DE DESENVOLVIMENTO EM CASCATA.

EM NOVEMBRO DE 2001 UM GRUPO DE DEZESSETE PESSOAS, REFERÊNCIAS MUNDIAIS EM DESENVOLVIMENTO DE SOFTWARE, SE REUNIU PARA ENTRE OUTRAS COISAS DISCUTIREM A VIRADA DA MESA!

FOI AÍ QUE SURGIU O MANIFESTO ÁGIL, QUE DEFINE OS SEGUINTES PRINCÍPIOS:

- INDIVÍDUOS E INTERAÇÕES SÃO MAIS IMPORTANTES QUE PROCESSOS E FERRAMENTAS.
- SOFTWARE FUNCIONANDO É MAIS IMPORTANTE DO QUE DOCUMENTAÇÃO COMPLETA E DETALHADA.
- COLABORAÇÃO COM O CLIENTE É MAIS IMPORTANTE DO QUE NEGOCIAÇÃO DE CONTRATOS.
- ADAPTAÇÃO A MUDANÇAS É MAIS IMPORTANTE DO QUE SEGUIR O PLANO INICIAL.

PODEMOS ENCARAR O O MANIFESTO ÁGIL COMO UMA INQUIETAÇÃO MOTIVADA POR:

- REAÇÃO ÀS METODOLOGIAS PESADAS
- MENOS ORIENTAÇÃO AO DOCUMENTO MAIS ORIENTAÇÃO AO CÓDIGO
- MÉTODOS CLAROS E ADAPTÁVEIS
- PROCESSOS ÁGEIS: MINIMAMENTE CARREGADOS COM ATIVIDADES QUE CONSOMEM E NÃO PRODUZEM NENHUM GANHO VISÍVEL

É IMPORTANTE ENTERDERMOS QUE METODOLOGIAS ÁGEIS,

- SÃO UMA ATITUDE, NÃO UM PROCESSO PRESCRITIVO.
- É UM SUPLEMENTO AOS MÉTODOS EXISTENTES, ELE NÃO É UMA METODOLOGIA COMPLETA.
- É UMA FORMA EFETIVA DE SE TRABALHAR EM CONJUNTO PARA ATINGIR AS NECESSIDADES DAS PARTES INTERESSADAS NO PROJETO.
- É UMA COISA QUE FUNCIONA NA PRÁTICA, NÃO É TEORIA ACADÊMICA.
- NÃO É UM ATAQUE À DOCUMENTAÇÃO, PELO CONTRÁRIO ACONSELHA A CRIAÇÃO DE DOCUMENTOS QUE TEM VALOR.
- UTILIZAM O MODELO ITERATIVO E INCREMENTAL.
- O CLIENTE É FAZ PARTE DA EQUIPE DE DESENVOLVIMENTO.

# APRESENTAÇÃO DE ALGUMAS METODOLOGIAS ÁGEIS

## EXTREME PROGRAMMING - XP

XP É UMA ABORDAGEM DELIBERADA E DISCIPLINADA PARA DESENVOLVIMENTO DE SOFTWARE.

CRIADA POR KENT BAECK EM 1996 DURANTE O PROJETO DAIMLER-CHRYSLER.

O SUCESSO DE XP ADVÉM DA INTENSA SATISFAÇÃO DO CLIENTE. CLIENTE SATISFEITO É O MELHOR INDICATIVO DE SUCESSO DE UM PROJETO.

ESTA METODOLOGIA FOI CRIADA PARA PRODUZIR O SOFTWARE QUE O CLIENTE PRECISA QUANDO ELE É NECESSÁRIO.

XP ENCORAJA OS DESENVOLVEDORES A ATENDER AS REQUISIÇÕES DE MUDANÇAS DOS REQUISITOS DO SOFTWARE, NO MOMENTO EM ISTO ACONTECE.

EM XP ALGUNS PRINCÍPIOS TRADUZEM O ESPÍRITO DA METODOLOCIA E DEVEM SEREM RIGOROSAMENTE SEGUIDOS E PLANEJADOS.

- SIMPLICIDADE
- COMUNICAÇÃO
- FEEDBACK
- CORAGEM

## SIMPLICIDADE

- TENTE SEMPRE DESENVOLVER A SOLUÇÃO MAIS SIMPLES POSSÍVEL.
- MUITOS PROJETOS PERDEM MUITO TEMPO QUANDO OS DESENVOLVEDORES DESTINAM MUITO TEMPO DESENVOLVENDO UMA SOLUÇÃO GENÉRICA.
- A SOLUÇÃO DEVE RESPONDER SIMPLESMENTE UM REQUISITO DO USUÁRIO.
- ALGUMAS FUNCIONALIDADES PODEM NUNCA VIR A SEREM UTILIZADAS.

## COMUNICAÇÃO

- CANAL ABERTO DE COMUNIÇÃO ENTRE A EQUIPE DE DESENVOLVIMENTO E COM OS USUÁRIOS.
- A COMUNICAÇÃO É CHAVE PARA O SUCESSO.

## FEEDBACK

- FEEDBACK POSSIBILITA QUE O SOFTWARE EVOLVA
- "PERGUNTE AO SOFTWARE, NÃO A UM DOCUMENTO"
- FEEDBACK PRECISA SER:
  - CEDO (PRA GENTE DESCOBRIR LOGO SE ESTÁ FAZENDO A COISA CORRETA)
  - CONCRETO (FEEDBACK ORIUNDO DO CÓDIGO)
  - CONSTANTE (O CICLO DE DESENVOLVIMENTO TEM QUE SER CURTO)

## CORAGEM

- COLOCAR O CLIENTE A PAR DO QUE TÁ ACONTECENDO
- ACREDITAR NA CAPACIDADE DE RESPONDER A MUDANÇAS
- APRENDER COM OS ERROS
- ACREDITAR NO FEEDBACK (NÃO NA "TEORIA")
- JOGAR PRA GANHAR (NÃO PRA TER UMA DESCULPA)
- FAZER O QUE PRECISA SER FEITO
  - JOGAR FORA CÓDIGO RUIM

O PROJETO É INCIADO EM XP COM O LEVANTAMENTO DAS "ESTÓRIAS DOS USUÁRIOS".

- CADA ESTÓRIA É ESCRITA PELO USUÁRIO E CONSISTE DE UM OU ALGUNS PARÁGRAFOS DE UM TEXTO NARRATIVO/DESCRITIVO. NÃO É TEXTO TÉCNICO.

- O PROPÓSITO DA ESTÓRIA NÃO É DEFINIR TODA A FUNCIONALIDADE DE UM CENÁRIO, MAS SIM, ESTIMAR COMO SERÁ A COMPLEXIDADE DE PARTE DO SISTEMA EM QUANTO TEMPO ISSO SERÁ DESENVOLVIDO.
- TODOS OS DEMAIS DETALHES DA ESTÓRIA SERÃO ESCLARECIDOS COM O CLIENTE, IMEDIATAMENTE AO INÍCIO DO DESENVOLVIMENTO.

O PRÓXIMO PASSO É O PLANEJAMENTO DO *RELEASE*, ONDE SERÁ DEFINIDO QUAIS ESTÓRIAS DEVERÃO SER DESENVOLVIDADAS EM QUAIS *RELEASES*.

CADA *RELEASE* CONSISTE DE UM NÚMERO DE ITERAÇÕES. CADA ITERAÇÃO TERÁ UM CONJUNTO DE ESTÓRIAS IMPLEMENTADAS. EM CADA ITERAÇÃO,:

- PLANEJE, PARA QUE VOCÊ SEMPRE FAÇA A COISA MAIS IMPORTANTE AINDA A FAZER
- CODIFIQUE, SENÃO O SOFTWARE NÃO SAI
- TESTE, SENÃO VOCÊ NÃO SABE SE ESTÁ FUNCIONANDO
- REFATORE, SENÃO O CÓDIGO VAI FICAR TÃO RUIM QUE SERÁ IMPOSSÍVEL DAR MANUTENÇÃO
- ESCUTE, PARA QUE VOCÊ SAIBA QUAL É O PROBLEMA A RESOLVER

O REFATORAMENTO E OS TESTES AUTOMÁTICOS SÃO ATIVIDADES FUNDAMENTAIS EM XP.

### REFATORAMENTO

- REFATORAR É MELHORAR O CÓDIGO SEM ALTERAR SUA FUNCIONALIDADE
- ANTES DE UMA MUDANÇA, VOCÊ REFATORA O

CÓDIGO PARA QUE A MUDANÇA SEJA SIMPLES DE FAZER

- REFATORAÇÃO CONTINUA POSSIBILITA MANTER UM DESIGN LEGAL, MESMO COM MUDANÇAS FREQÜENTES

## TESTES AUTOMÁTICOS

- TESTES AUTOMÁTICOS SÃO PARTE DO SOFTWARE
  - SE VOCÊ TEM SOMENTE A FUNCIONALIDADE, SEU SOFTWARE ESTÁ INCOMPLETO
- TESTES PERMITEM QUE VOCÊ REFATORE SEM MEDO DE QUEBRAR O CÓDIGO
- TESTES REPRESENTAM UMA "REDUNDÂNCIA LÓGICA" QUE VOCÊ ADICIONA AO CÓDIGO
- ESCREVENDO TESTES ANTES DA FUNCIONALIDADE, VOCÊ CLAREIA DÚVIDAS SOBRE O QUE O SOFTWARE DEVE FAZER

ALGUMAS PRÁTICAS IMPORTANTES EM XP:

- PROJETO MAIS SIMPLES POSSÍVEL
- PROGRAMAÇÃO EM PARES
- PROPRIEDADE COLETIVA DO CÓDIGO
- CLIENTE SEMPRE DISPONÍVEL
- ESTÓRIAS DO USUÁRIO
- PLANEJAMENTO DO RELEASE

## PROJETO MAIS SIMPLES POSSÍVEL

- PROJETOS FLEXÍVEIS SÃO UMA DEFESA CONTRA MUDANÇAS IMPREVISTAS NO SOFTWARE
- PORÉM, PROJETOS FLEXÍVEIS TAMBÉM TÊM

CUSTOS

- TEMPO PARA DESENVOLVIMENTO E MANUTENÇÃO
- O CÓDIGO FICA MAIS COMPLEXO
- MUITA VEZES A FLEXIBILIDADE NÃO É UTILIZADA NUNCA

- COMO MUDANÇA É BARATA EM XP, VAMOS MANTER O PROJETO MAIS SIMPLES POSSÍVEL, MODIFICANDO-O QUANDO FOR NECESSÁRIO SUPORTAR MAIS FUNCIONALIDADE
- O MELHOR PROJETO É AQUELE QUE:
  - RODA TODOS OS TESTES
  - NÃO CONTÉM DUPLICAÇÃO DE FUNCIONALIDADE
  - DEIXA CLARO AS DECISÕES DE DESIGN IMPORTANTES
  - TEM O MENOR NÚMERO POSSÍVEL DE CLASSES E MÉTODOS
- O MELHOR PROJETO NÃO É AQUELE:
  - MAIS FLEXÍVEL (COM MAIS "GANCHOS")
  - MAIS ABSTRATO
  - QUE RESISTIRÁ AO TEMPO

## PROGRAMAÇÃO EM PARES

- SE REVISÃO DE CÓDIGO É LEGAL, VAMOS FAZÊ-LA O TEMPO TODO
- EM XP, PROGRAMAÇÃO É FEITA EM PARES
- PARES MUDAM COM RELATIVA RAPIDEZ (EM DIAS)

- PROGRAMAÇÃO EM PARES FAVORECE COMUNICAÇÃO E APRENDIZADO
- MAS, VOCÊ PRECISA ESTABELECER UM PADRÃO DE CODIFICAÇÃO
- HÁ CASOS DE REDUÇÃO NO TEMPO DE DESENVOLVIMENTO COM PROGRAMAÇÃO EM PARES

## PROPRIEDADE COLETIVA DO CÓDIGO

- DESENVOLVIMENTO COM OBJETOS LEVA A ALTERAÇÕES POR TODO O CÓDIGO
- COORDENAR ALTERAÇÕES TOMA TEMPO E GERA RESISTÊNCIAS NO "DONO" DO CÓDIGO
- EM XP, "NÃO SE COORDENA", SIMPLESMENTE FAZ-SE O QUE PRECISA SER FEITO
- MAS INTEGRA-SE FREQÜENTEMENTE
- NO MÁXIMO, UMA VEZ POR DIA
- TODOS SÃO RESPONSÁVEL POR TODO O CÓDIGO
- QUALQUER UM QUE VÊ UMA OPORTUNIDADE DE ADICIONAR VALOR AO CÓDIGO, DEVO FAZÊ-LO
  - MANTENDO EM VISTA AS PRIORIDADES DO CLIENTE
  - MANTENDO O DESIGN MAIS SIMPLES POSSÍVEL
- TESTES PROTEGEM A FUNCIONALIDADE
- PADRÃO DE CODIFICAÇÃO EVITA A "GUERRA DOS PARÊNTESES"

## CLIENTE SEMPRE DISPONÍVEL

- UM CLIENTE (USUÁRIO DA APLICAÇÃO) DEVE TRABALHAR COM O TIME PARA ESCLARECER

DÚVIDAS E ESTABELECER PEQUENAS PRIORIDADES

- É MUITO CARO COLOCAR UM CLIENTE A DISPOSIÇÃO DO DESENVOLVIMENTO?
- TALVEZ ENTÃO NÃO VALHA A PENA FAZER O SISTEMA...

## ESTÓRIAS DO USUÁRIO

- USUÁRIOS ESCREVEM ESTÓRIAS DESCREVENDO A FUNCIONALIDADE QUE QUEREM
- DESENVOLVEDORES ESTIMAM O TEMPO NECESSÁRIO PARA IMPLEMENTAR CADA ESTÓRIA
- UM RELEASE É UM CONJUNTO DE ESTÓRIAS QUE SÃO DISPONIBILIZADOS SIMULTANEAMENTE
- AS ESTÓRIAS MAIS IMPORTANTES E/OU MAIS DIFÍCEIS TEM PRIORIDADE

## PLANEJAMENTO DO RELEASE

O CLIENTE DECIDE:

- ESCOPO
- PRIORIDADE
- COMPOSIÇÃO DO RELEASE
- DATA DO RELEASE

OS PROGRAMADORES DECIDEM

- ESTIMATIVAS
- CONSEQUÊNCIAS
- PROCESSO
- PLANEJAMENTO INTRA-RELEASE (O MAIS ARRISCADO PRIMEIRO)

- XP PRECONIZA RELEASES PEQUENOS E FREQÜENTES (A CADA 2-3 MESES)
- AS QUATRO DIMENSÕES DO DESENVOLVIMENTO DE SOFTWARE SÃO CUSTO, TEMPO, QUALIDADE E ESCOPO
- XP TENTA MANTER ESCOPO COMO VARIÁVEL LIVRE
- RELEASES SÃO DIVIDIDAS EM ITERAÇÕES DE 2-3 SEMANAS

- UMA ITERAÇÃO ALCANÇA ALGUM OBJETIVO (TIPICAMENTE A ADIÇÃO DE NOVA FUNCIONALIDADE)
- NADA É FEITO QUE NÃO SEJA IMEDIATAMENTE ÚTIL E NECESSÁRIO PARA NÃO IMPACTAR OS PRAZOS DE DESENVOLVIMENTO

PROBLEMAS EM XP

- CONSIDERAR TESTES COMO PARTE NORMAL DO PROCESSO DE DESENVOLVIMENTO
- SEMPRE FAZER A COISA MAIS SIMPLES
- ADMITIR QUE VOCÊ NÃO SABE
- COLABORAR
- VENCER RESISTÊNCIA NAS PESSOAS
- TIMES GRANDES
- SITUAÇÕES EM QUE VOCÊ NÃO PODE MUDAR LIVREMENTE O CÓDIGO

## FEATURE DRIVEN DEVELOPMENT - FDD

FEATURE DRIVEN DEVELOPMENT É UMA METODOLOGIA ÁGIL CRIADA POR JEFF DE LUCA E PETER CODE. FOI RECONHECIDA EM 1997.

FDD  POSSUI REQUISITOS MAIS FORMAIS E MAIS PASSOS QUE XP, ALÉM DE POSSUIR UM MECANISMO MAIS PRECISO PARA ACOMPANHAMENTO DO PROJETO.

O DESENVOLVIMENTO BASEADO EM FDD CONSISTE DE DOIS ESTÁGIOS PRINCIPAIS:

- DESCOBRIR UMA LISTA DE "*FEATURES*" A SEREM

IMPLEMENTADAS.

- IMPLEMENTAÇÃO BASEADA EM "*FEATURES*".

DESCOBRIR A LISTA DE "*FEATURES*" É SEM DÚVIDA O PROCESSO MAIS CRÍTICO. É A CHAVE PARA O SUCESSO DO PROJETO!

A QUALIDADE COM A QUAL SE IDENTIFICA A LISTA DE "*FEATURES*" DEFINE QUÃO PRECISO O PROJETO SERÁ CONTROLADO, QUÃO EXTENSÍVEL E MANUTENIBILIDADE O CÓDIGO TERÁ.

REQUER PARTICIPAÇÃO INTEGRAL DO CLIENTE JUNTO À EQUIPE DE DESENVOLVIMENTO.

COMO A IDENTIFICAÇÃO DA LISTA DE "*FEATURES*" DERIVA-SE UM DIAGRAMA DE CLASSES DE ANÁLISE.

AS RESPONSABILIDADES DAS CLASSES DEVEM EXPRESSAR UM LINGUAJAR COMUM ENTRE DEVESENVOLVEDORES E USUÁRIOS.

POR EXEMPLO: IMAGINEM UM SISTEMA DE COMPRAS ON-LINE ONDE O CLIENTE LOGA NO SISTEMA PARA COMPRAR ALGO.

- O DIAGRAMA DE CLASSES DE CONTER CLASSES COMO: *CARRINHO DE COMPRAS, CLIENTE E ITEM*.
- O RESULTADO DA LISTA DE "*FEATURES*" DEVE INCLUIR:
  - CRIAR UM NOVO *CARRINHO DE COMPRAS* PARA O *CLIENTE*;
  - ADICIONAR UM NOVO *ITEM* NO *CARRINHO DE COMPRAS*;
  - LISTAR TODOS *ITENS* DO *CARRINHO DE COMPRAS*;

- CALCULAR O PREÇO TOTAL DOS *ITENS* DO *CARRINHO DE COMRAS*.

A LISTA DE "*FEATURES*" DEVE REPRESENTAR ALGO PARA O USUÁRIO, REFLETINDO DIRETAMENTE A FUNCIONALIDADE QUE SERÁ DISPONIBILIZADA NA APLICAÇÃO.

A LISTA DE "*FEATURES*" TAMBÉM DEVE CONTER UNIDADES DE TRABALHO PARA OS DESENVOLVEDORES, ONDE TODA "*FEATURE*" É PEQUENA O SUFICIENTE PARA QUE O SEU DESENVOLVIMENTO SEJA FEITO EM PEQUENAS ITERAÇÕES.

A IMPLEMENTAÇÃO COMEÇA ATRAVÉS DO AGRUPAMENTE DE UM CONJUNTO DE "*FEATURES*" RELACIONADAS EM UM PACOTE DE TRABALHO.

UM PACOTE DE TRABALHO DEVE SER COMPLETAMENTE DESENVOLVIDO EM UMA ITERAÇÃO, NORMALMENTE DE 1 A 3 SEMANAS.

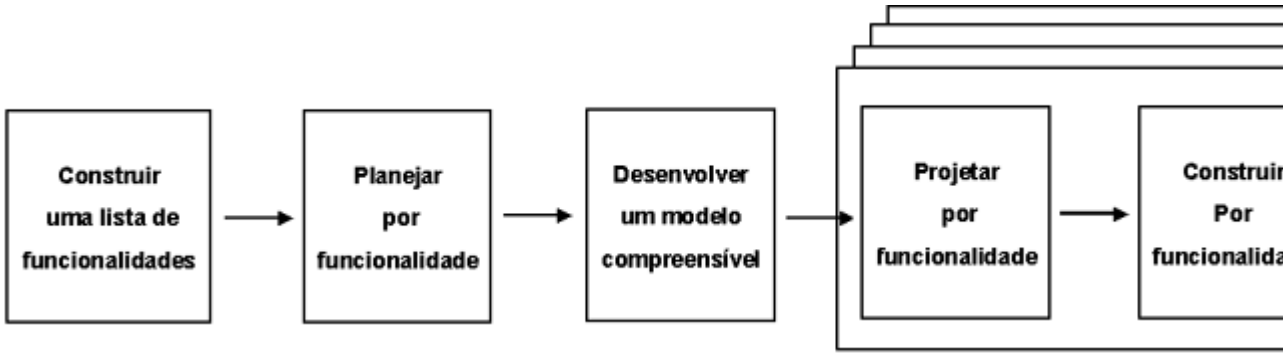
UM PACOTE DE TRABALHO DESENVOLVIDO DEVE APRESENTAR UMA PARTE DO SOFTWARE ONDE O USUÁRIO PODE UTILIZÁ-LO.

CADA ITERAÇÃO INCLUI:

- KICK-OFF MEETING PARA UM PACOTE DE TRABALHO: DETALHES DAS "*FEATURES*" INCLUÍDAS DEVEM SER ESCLARECIDOS.
- PROJETO: CLASSES/MÉTODOS E DOCUMENTAÇÃO NECESSÁRIAS SÃO CRIADAS.
- REVISÃO DE PROJETO: UM PROJETISTA EXPERIENTE AVALIA O PROJETO FEITO, ANALISANDO-O OU REJEITANDO-O.

- DESENVOLVIMENTO: IMPLEMTAÇÃO E TESTES DE UNIDADE SÃO CRIADOS.
- REVISÃO DE CÓDIGO: É UMA ESPÉCIE DE PROGRAMAÇÃO EM PARES.
- FECHAMENTO DO RELEASE: AS "*FEATURES*" DESENVOLVIDAS SÃO LIBERADAS EM UM BUILD.

O FDD POSSUI 5 PRINCIPAIS PROCESSOS:




## SCRUM

SCRUM É UM PROCESSO PARA CONSTRUIR SOFTWARE INCREMENTALMENTE EM AMBIENTES COMPLEXOS, ONDE OS REQUISITOS NÃO SÃO CLAROS OU MUDAM COM MUITA FREQÜÊNCIA.

O OBJETIVO DO SCRUM É FORNECER UM PROCESSO CONVENIENTE PARA PROJETOS E DESENVOLVIMENTO ORIENTADO A OBJETOS.

A METODOLOGIA É BASEADA EM PRINCÍPIOS SEMELHANTES AOS DE XP: EQUIPES PEQUENAS, REQUISITOS POUCO ESTÁVEIS OU DESCONHECIDOS, E ITERAÇÕES CURTAS PARA PROMOVER VISIBILIDADE PARA O DESENVOLVIMENTO.

SCRUM DIVIDE O DESENVOLVIMENTO EM SPRINTS DE 30 DIAS. EQUIPES PEQUENAS, DE ATÉ 7 PESSOAS, SÃO FORMADAS POR PROJETISTAS, PROGRAMADORES, ENGENHEIROS E GERENTES DE

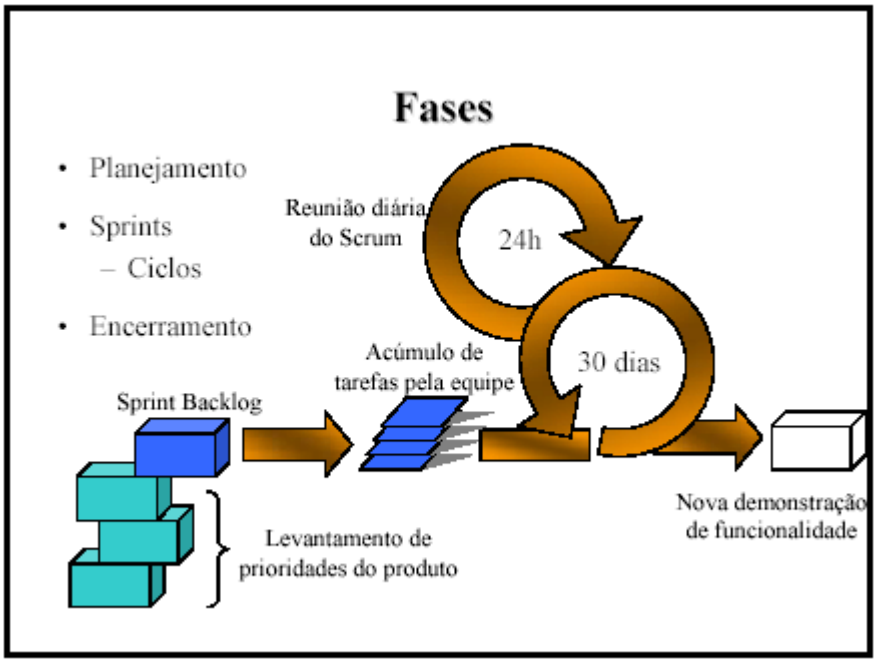
QUALIDADE.

AS EQUIPES TRABALHAM EM CIMA DE FUNCIONALIDADE (OS REQUISITOS, EM OUTRAS PALAVRAS) DEFINIDAS NO INÍCIO DE CADA SPRINT. A EQUIPE TODA É RESPONSÁVEL PELO DESENVOLVIMENTO DESTA FUNCIONALIDADE.

TODO DIA, É FEITA UMA REUNIÃO DE 15 MINUTOS ONDE O TIME EXPÕES À GERÊNCIA O QUE SERÁ FEITO NO PRÓXIMO DIA, E NESTAS REUNIÕES OS GERENTES PODEM LEVANTAR OS FATORES DE IMPEDIMENTO, E O PROGRESSO GERAL DO DESENVOLVIMENTO.

TODOS RESPONDEM ÀS PERGUNTAS:

- O QUE VOCÊ REALIZOU DESDE A ÚLTIMA REUNIÃO?
- QUAIS PROBLEMAS VOCÊ ENFRENTOU?
- EM QUE VOCÊ TRABALHARÁ ATÉ A PRÓXIMA REUNIÃO?




## ALGUMAS CONCLUSÕES SOBRE XP, FDD E SCRUM

AS TRÊS SÃO METODOLOGIAS QUE UTILIZAM O MODELO ITERATIVO PARA EVITAR OS PRINCIPAIS "GARGALOS" DO MODELO EM CASCATA;

AS TRÊS SÃO METODOLOGIAS ÁGEIS, MAS O XP É CONSIDERADO MUITO MAIS "LEVE" PORQUE NÃO PRODUZ MUITOS ARTEFATOS;

XP É MELHOR PARA PROJETOS QUE POSSUEM MUDANÇAS FREQUENTES OU "POBRE" LEVANTAMENTO DE REQUISITOS;

A ESCALABILIDADE DO FDD É MELHOR. EXISTE UMA HIERARQUIA DE PROCESSOS QUE PERMITE ITERAÇÕES TANTO COM EQUIPES PEQUENAS QUANTO GRANDES;

FDD OFERECE UM MELHOR ACOMPANHAMENTO GERENCIAL DO PROJETO;

AS TRÊS NECESSITAM DE DISCIPLINA, O QUE NÃO EXCLUI A NECESSIDADE DE UM BOM GERENTE DE PROJETOS;

SCRUM FORNECE UM MECANISMO DE INFORMAÇÃO DE STATUS QUE É ATUALIZADO CONTINUAMENTE;

SCRUM E XP SÃO COMPLEMENTARES POIS SCRUM PROVÊ PRÁTICAS ÁGEIS DE GERENCIAMENTO ENQUANTO XP ESTÁ MAIS PREOCUPADO COM A PRODUÇÃO DE CÓDIGO;

AS TRÊS OFERECEM TÉCNICAS BASTANTE ÚTEIS E QUE PODEM SER APLICADAS EM OUTRAS METODOLOGIAS.

## ALGUMAS OUTRAS METODOLOGIAS ÁGEIS:

- CRYSTAL/CLEAR
- DYNAMIC SYSTEMS DEVELOPMENT METHOD

(DSDM)

- ADAPTIVE SOFTWARE DEVELOPMENT (ASD)

METODOLOGIAS ÁGEIS - programa próxima anterior