Aula 2
Fundamentos de Engenharia de Software

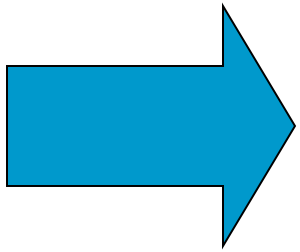
- Caracteristicas e Evolução do Software
- Mitos de Software
- Definição de Engenharia de Software
- Paradigmas Tradicionais
- Métodos Ageis

Introdução

- “O Software ultrapassou o Hardware como chave para o sucesso de muitos sistemas baseados em computador”

O Software é o que faz a diferença!

- *Completeza* da informação
- *user-friendlyness*
- *web-enhanced*
- inteligência
- funcionalidade
- compatibilidade
- suporte



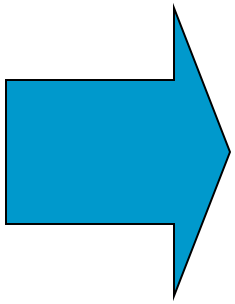
-
-
- Tornam 1 produto melhor que outro

A importância do Software

- Durante as 3 primeiras décadas da era do computador, o principal desafio era desenvolver um **HARDWARE** de baixo custo e alto desempenho.
- O hoje o desafio é melhorar a qualidade (e reduzir os custos) das soluções baseadas em **SOFTWARE**!

A evolução do Software

- Computação Industrial



3a. Onda

Parêntesis: Revolução Industrial Primeira Onda

- Ferro (Darby, 1709)
- Máquina a vapor:
  - Inventada (Newcomen, 1712)
  - Aperfeiçoada (WATT, 1766 - ’69 -’82)
- Mecanização da indústria têxtil:
  - Tear Mecânico (Kay, 1722)
  - Maquina de fiar (Hargreaves, 1764)
- Aspectos sociais, políticos e econo

Têxteis, Carvão e Ferro

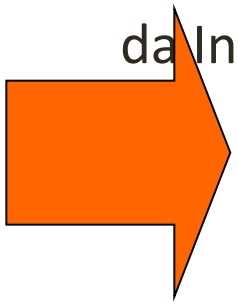
Parêntesis: Revolução Industrial Segunda Onda

- Aço (Bessemel, 1856 e 1885 - Liga)
- Locomotiva a Vapor (Rede de Transporte - 1830)
- Máquina de Costura (SINGER,1851)
- Motor a combustão interna:
  - Primeiro eficiente (OTTO, 1876)
  - Produção automobilística em massa (Daimler e Benz, 1896)
- Desemprego e fim da escravidão

Revolução Industrial: Terceira Ond

- Energia Nuclear (Fermi, 1942)
- Uso Industrial/Comercial da Eletricidade
- Computadores Eletrônicos (ENIAC 1946)
- Transistor (Shockley, et al., 1948)

Sociedade Industrial



Sociedade da Informação

transformação

Filosofando...

- A mudança de uma sociedade industrial para uma baseada na informação é uma *Radical Mudança Econômica*:
  - Material tem menos valor e Informação tem mais valor

- Antes: quanto menos pessoas possuísse algo, maior o valor.

- Hoje: quanto mais pessoas possuem algo, maior o valor.

Filosofando ... Exemplo!

- Cite as características dos sistemas operacionais que você conhece.
- Compare os sistemas:
  - Unix
  - Windows
  - MacOS

O Windows vende mais porque é mais fresquinho ou é mais fresquinho porque vende mais???

A evolução do software

- Software é dividida em 5 Eras:
  - Primeiros anos (1950) : foco no hardware, modo batch, sem documentação;
  - Segunda Era (1960) : interatividade com o usuario, multiusuarios, manutenção;
  - Terceira Era (1970) : redes, concorrência, sistemas distribuidos:
  - Quarta Era (1980) : orientação a objetos, sistemas especialistas, métodos formais;
  - Quinta Era (1990) : internet, mobilidade, sistemas hibridos

O que é Software?

- Definição - Software é:

1º - instruções (programas de computador) que, quando executadas, produzem a função e o desempenho desejados;

2º - estruturas de dados que permitem a manipulação das informações;

3º - documentos que descrevem a operação e uso dos programas.

Características do Software - 1

- O Software é desenvolvido ou projetado por engenharia, não manufaturado no sentido clássico:
  - Custos são concentrados no trabalho de engenharia.
  - Projetos não podem ser geridos como projetos de manufatura.
  - “Fábrica de Software!”

Características do Software - 2

- Software não desgasta!
  - Software não é sensível aos problemas ambientais que fazem com que o hardware se desgaste.
  - Toda falha indica erro de projeto ou implementação: manutenção do SW é mais complicada que a do HW.

Características do Software - 3

- A maioria dos softwares é feita sob medida e não montada a partir de componentes existentes.
- != Hardware.
- Situação esta mudando:
  - Orientação a objetos.
  - Reusabilidade é o “Santo Graal”(diminui custos e melhora projetos).

Aplicações de Software

- Software Básico
- Software de Tempo Real
- Software Comercial
- Software Científico ou de Engenharia
- Software Embutido
- Software de Computador Pessoal
- Software de Inteligência Artificial

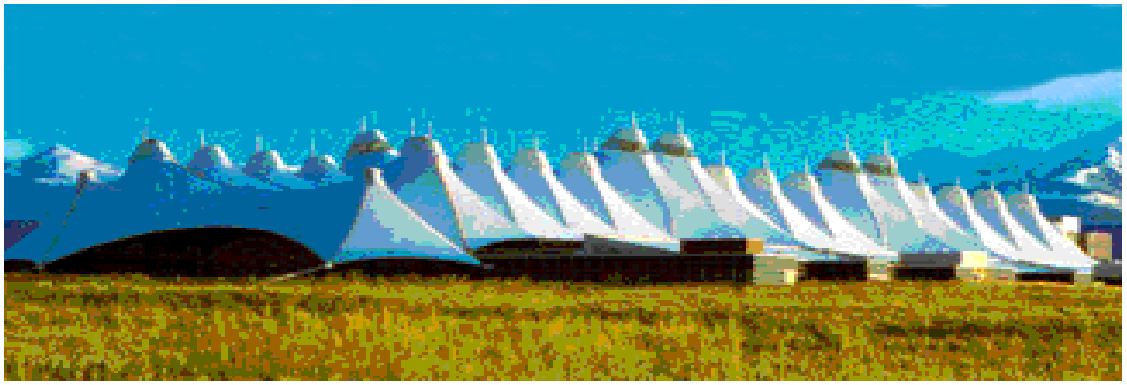
Uma Crise no horizonte

- A industria de Software tem tido uma “crise” que a acompanha há quase 30 anos:
  - Aflição Crônica != Crise
- Problemas não se limitam ao software que não funciona adequadamente, mas abrange:
  - desenvolvimento, testes, manutenção, suprimento, etc.

Therac-25

- Equipamento de Radioterapia.
- Entre 1985 e 1987 se envolveu em 6 acidentes, causando mortes por overdoses de radiação.
- Software foi adaptado de uma antecessora, Therac-6:
  - falhas por falta de testes integrados
  - falta de documentação

Denver International Airport



- Custo do projeto: US$ 4.9 bilhões
  - 100 mil passageiros por dia
  - 1,200 vôos
  - 53 milhas quadradas
  - 94 portões de embarque e desembarque
  - 6 pistas de pouso / decolagem

Denver International Airport

- Erros no sistema automático de transporte de bagagens (*misloaded*, *misrouted*, *jammed*):
  - Atraso na abertura do aeroporto com custo total estimado em US$360 Milhões
- 86 milhões para consertar o sistema

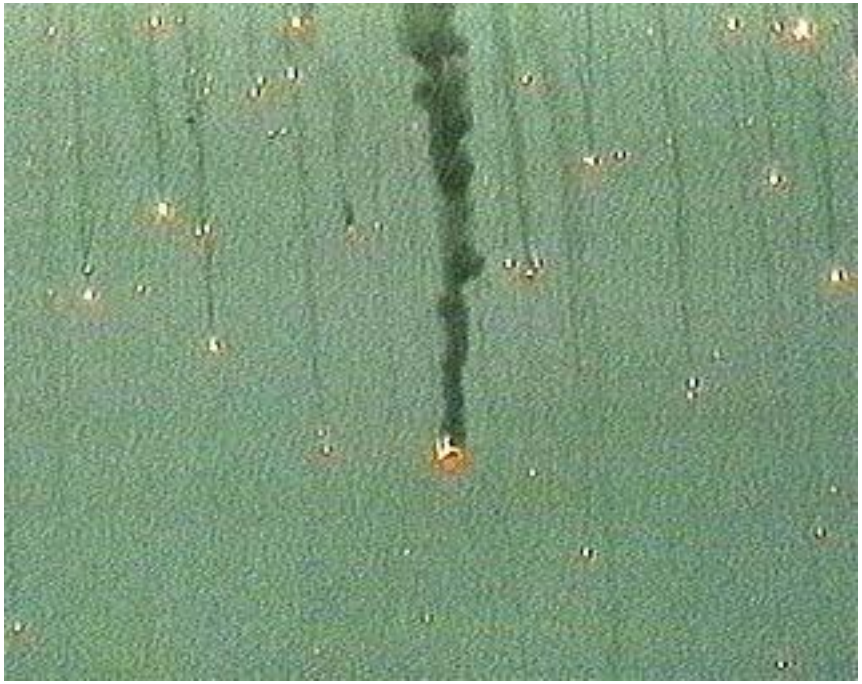
Ariane 5

Ariane 5

- Projeto da Agência Espacial Européia que custou:
  - 10 anos.
  - US$ 8 Bilhões.
- Capacidade 6 toneladas.
- Garante supremacia européia no espaço.

Vôo inaugural em 4/junho/1996

Resultado



- Explosão 40 segundos após a decolagem.
- Destruição do foguete e carga avaliada em US$ 500 milhões.

O que aconteceu? (I)

- Fato: o veículo detonou suas cargas explosivas de autodestruição e explodiu no ar. Por que?
- Porque ele estava se quebrando devido às forças aerodinâmicas. Mas por que?
- O foguete tinha perdido o controle de direção (atitude). Causa disso?
- Os computadores principal e back-up deram shut-down ao mesmo tempo.

O que aconteceu? (II)

- Por que o Shut-down? Ocorrera um *run time error* (out of range, overflow , ou outro) e ambos computadores se desligaram. De onde veio este erro?
- Um programa que convertia um valor em ponto flutuante para um inteiro de 16 bits recebeu como entrada um valor que estava fora da faixa permitida.

Especificamente:O que faltou?

- strict precondition 1:
- {
- Set."x"=FLPT and Set."y"=INT16
- **and -32768 <= x <= +32767**
- }
- program code:
- y := int(x);
- postcondition:
- {Set."x"=FLPT and Set."y"=INT16 and y=int(x)}

Ironia...

- O resultado desta conversão não era mais necessário após a decolagem...

Quais são os problemas?

- A sofisticação do software ultrapassou nossa capacidade de construção.
- Nossa capacidade de construir programas não acompanha a demanda por novos programas.
- Nossa capacidade de manter programas é ameaçada por projetos ruins.

Perguntas que Engenharia de Software quer responder:

- Porque demora tanto para concluir um projeto (não cumprimos prazos)?
- Porque custa tanto (uma ordem de magnitude a mais)?
- Porque não descobrimos os erros antes de entregar o software ao cliente?
- Porque temos dificuldade de medir o progresso enquanto o software está sendo desenvolvido?

Causas óbvias

- Não dedicamos tempo para coletar dados sobre o desenvolvimento do software - resulta em estimativas “a olho”.
- Comunicação entre o cliente e o desenvolvedor é muito fraca.
- Falta de testes sistemáticos e completos.

Causas menos óbvias

- O Software é desenvolvido ou projetado por engenharia, não manufaturado no sentido clássico (característica 1).
- Gerentes sem *background* em desenvolvimento de SW.
- Profissionais recebem pouco treinamento formal.
- Falta investimento (em ES).
- Falta métodos e automação.

Mitos do Software - Gerente

- Um manual oferece tudo que se precisa saber.
- Computadores de última geração solucionam problemas de desenvolvimento.
- Se estamos atrasados, basta adicionar programadores e tirar o atraso.

Mitos do Software - do Cliente

- Uma declaração geral é suficiente para começar a escrever programas.
- Mudanças podem ser facilmente acomodadas em um projeto

Mitos do Software - do Profissional

- Um programa está terminado ao funcionar.
- Quanto mais cedo escrever o código, mais rápido terminarei o programa.
- Só posso avaliar a qualidade de um programa em funcionamento.
- A única coisa a ser entregue em um projeto é o programa funcionando.

Engenharia de Software: Definição

- “Engenharia de Software é o estabelecimento e uso de sólidos princípios de engenharia para que se possa garantir a qualidade do sistema e a satisfação do usuario”
- É METODOLOGIA!

Engenharia de Software: Abrangência

- E.S. possui 3 elementos fundamentais:
  - métodos: “como fazer”
  - ferramentas: apoio automatizado aos métodos.
  - Procedimentos: elo de ligação entre os métodos e os procedimentos
- Existem diversos *Paradigmas de Engenharia de Software*:
  - abordagens que envolvem estes métodos, ferramentas e procedimentos

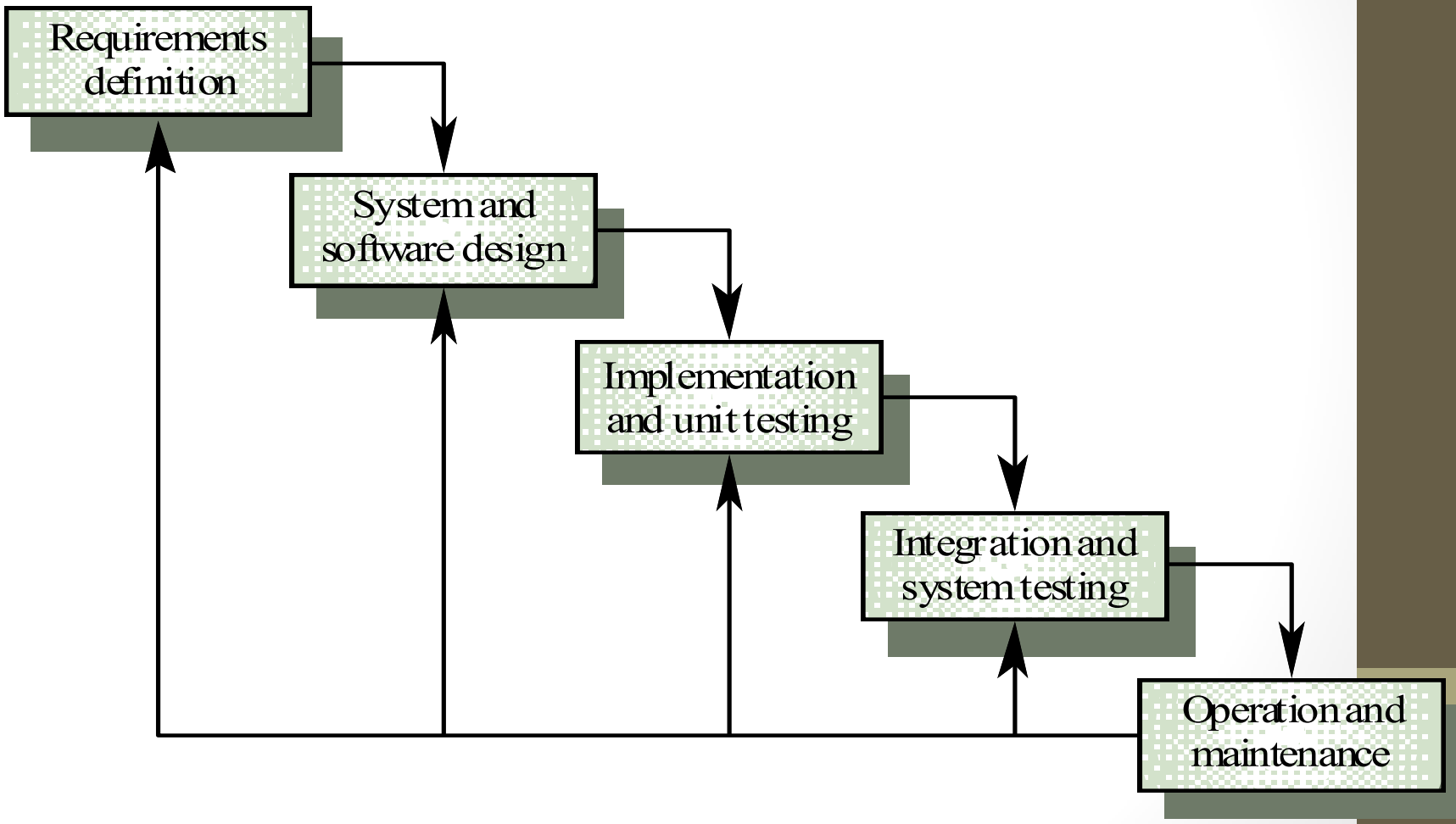
Paradigmas de Engenharia de Software

- Existem dezenas.
- 5 principais:
  - Ciclo de Vida Clássico (modelo Cascata)
  - RUP
  - Prototipagem
  - Espiral
  - Técnicas de Quarta Geração

Ciclo de Vida Clássico: modelo *Cascata (Waterfall)*

- Baseado em projetos de engenharia clássicos (não de Software) - 1970
- Fases:
  - Análise de requisitos
  - Definição
  - Projeto
  - Implementação
  - Integração e testes
  - Operação e manutenção

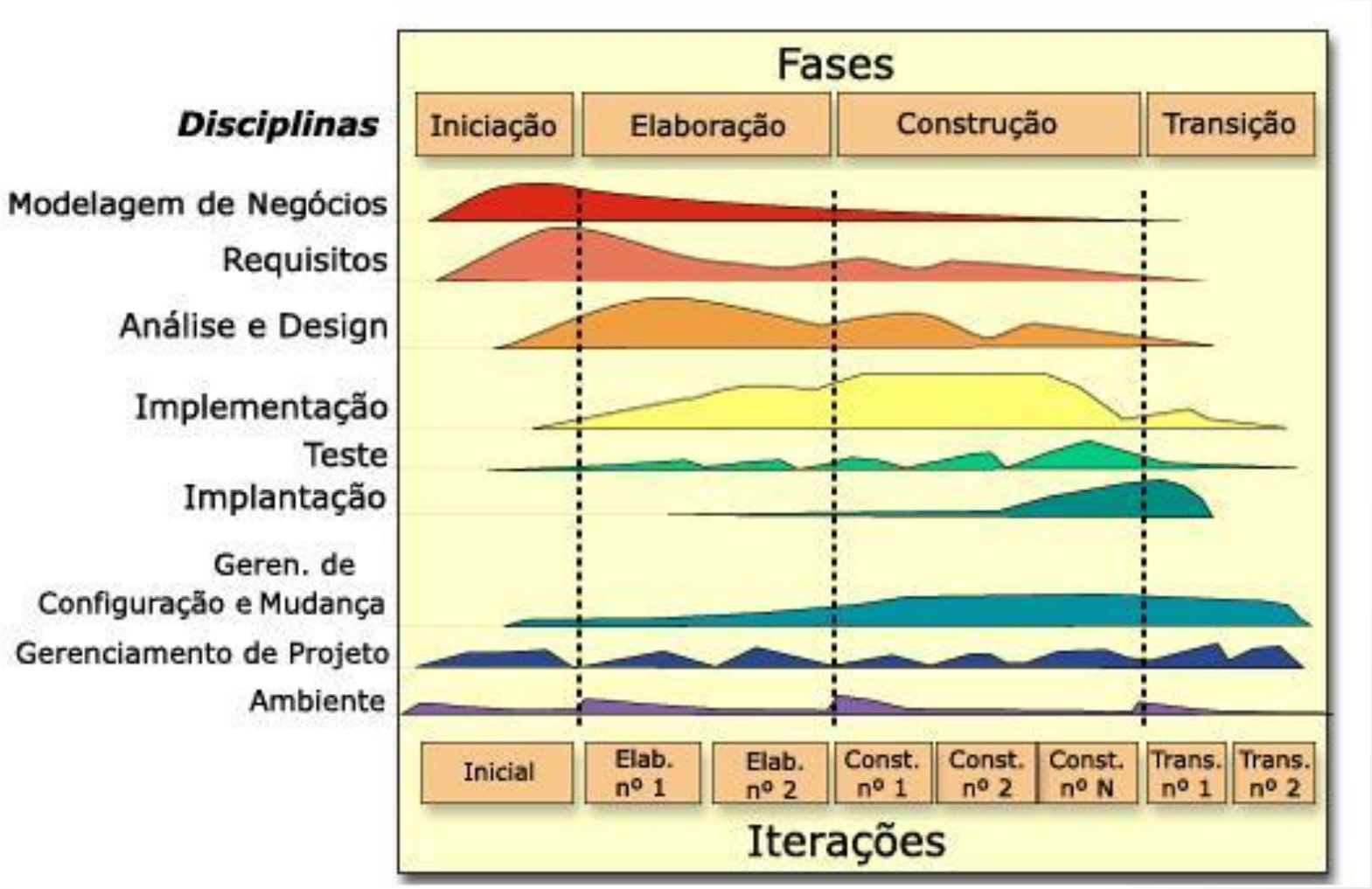
Ciclo de Vida Clássico (II)



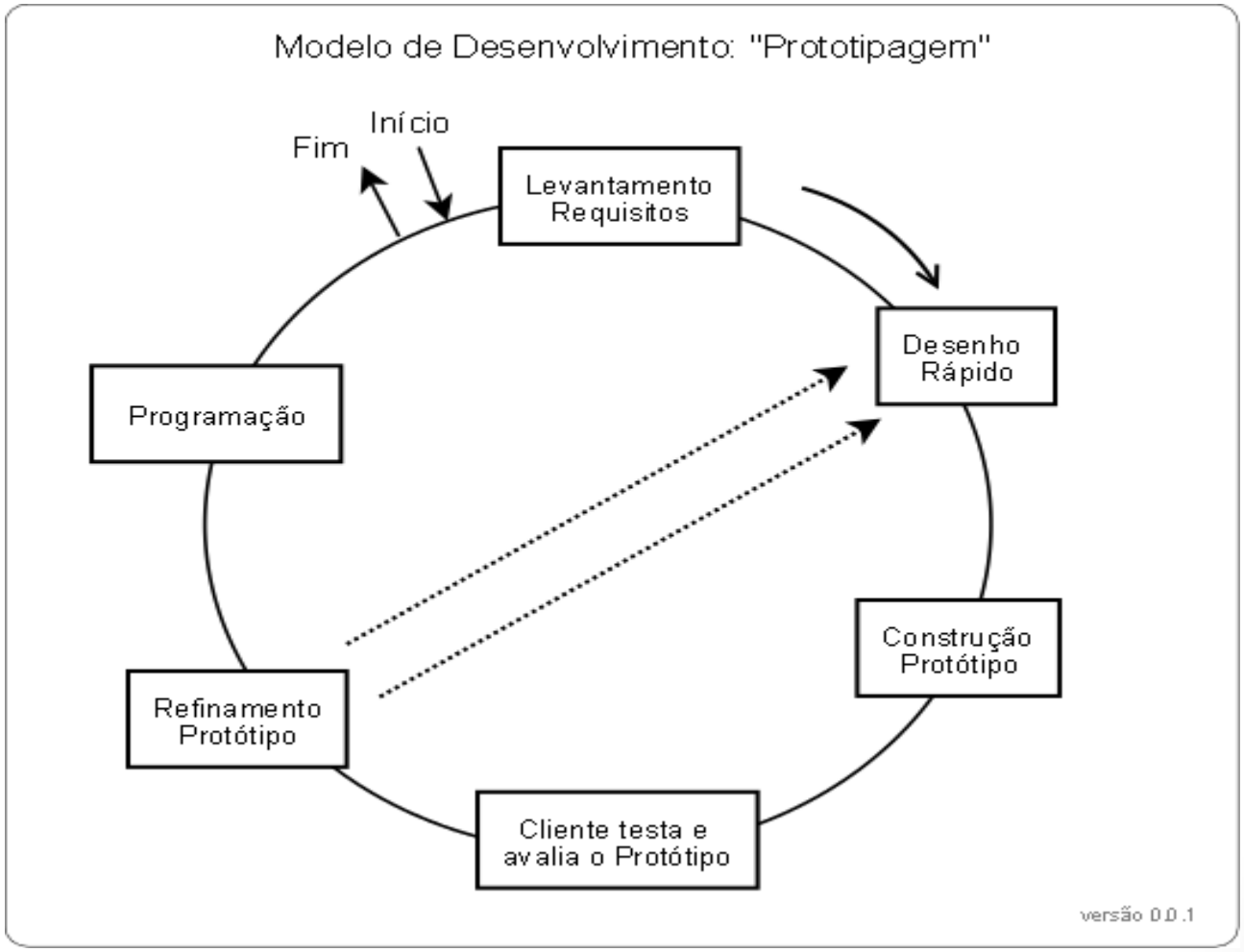
Ciclo de Vida Clássico (III)

- Problemas:
  - projetos reais não seguem um fluxo seqüencial: dificuldade de acomodar mudanças depois de iniciado.
  - Dificuldade de declaração de todas as exigências pelo cliente.
  - Paciência até a primeira versão!

RUP


Fases
Disciplinas
Iniciação
Elaboração
Construção
Transição
Modelagem de Negócios
Requisitos
Análise e Design
Implementação
Teste
Implantação
Geren. de Configuração e Mudança
Gerenciamento de Projeto
Ambiente
Inicial
Elab. nº 1
Elab. nº 2
Const. nº 1
Const. nº 2
Const. nº N
Trans. nº 1
Trans. nº 2
Iterações

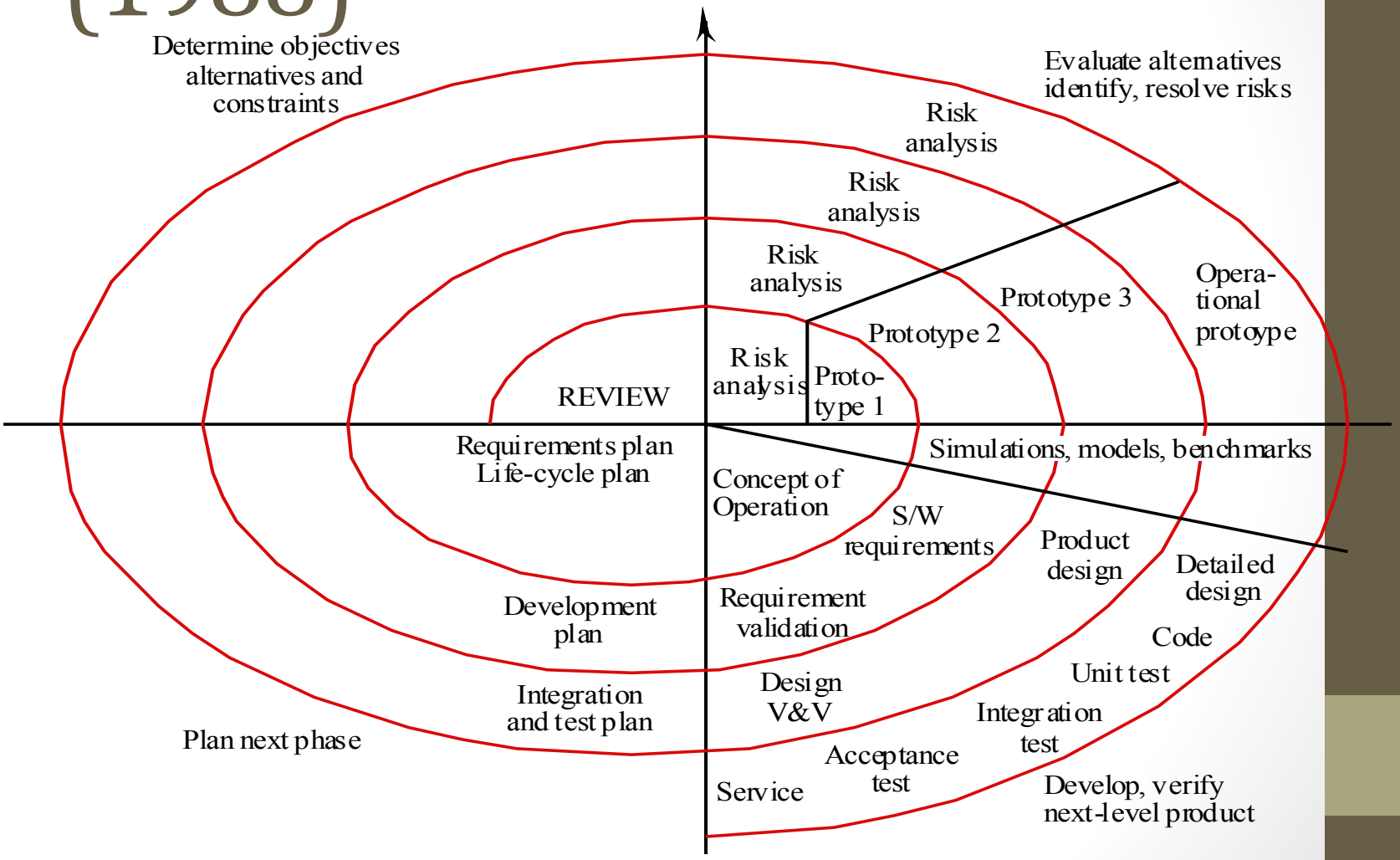
Modelo de Prototipagem



Modelo de Prototipagem

- Problemas:
  - Fluxo sequencial continuo, pois o cliente revisa uma unica vez.
  - O cliente pode achar que o prototipo já é o sistema.
  - Devido ao tempo, o programador esquece « lixo » no codigo.

Modelo Espiral de Boehm (1988)



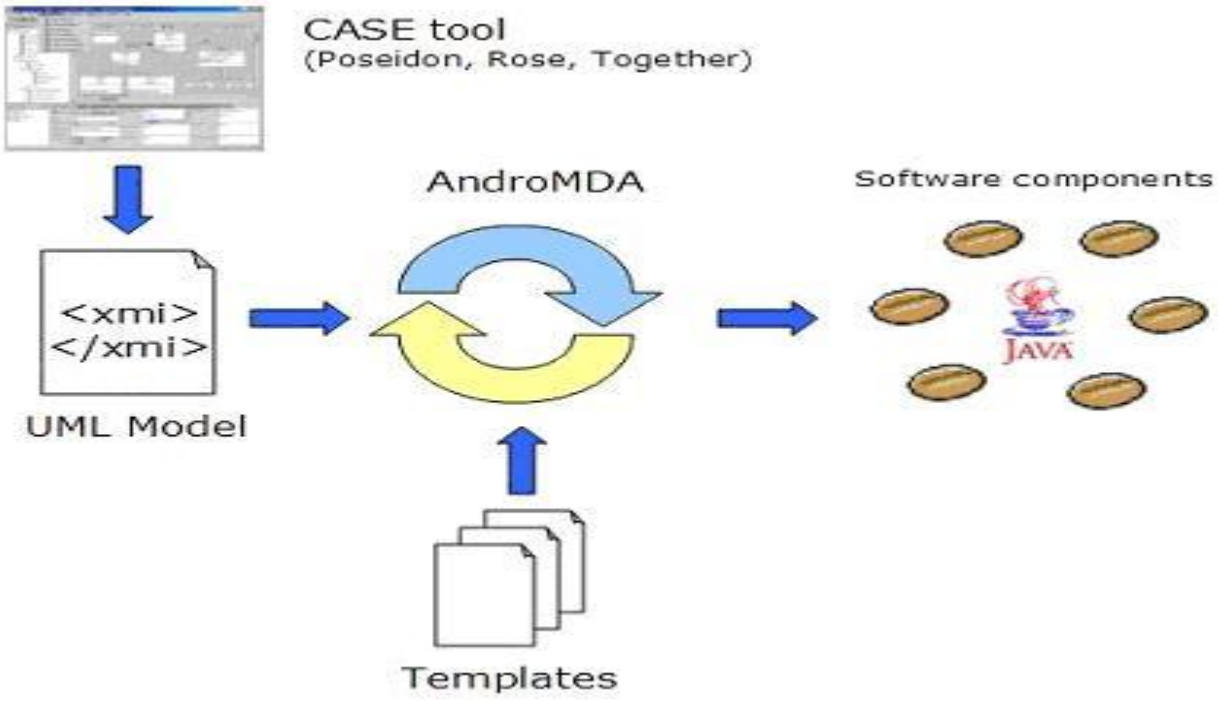
Fases do modelo Espiral

- Definição dos objetivos
  - Especificação dos objetivos específicos desta fase.
- Análise dos riscos
  - Identificação e solução dos principais riscos
- Desenvolvimento e validação
- Planejamento
  - O projeto é revisto e se define planos para a próxima “volta da espiral”

Modelo Espiral

- Problemas:
  - Descobrir o fim.
  - Quanto mais ciclos, mais irritado o cliente fica.

Desenvolvimento Dirigido por Modelos



Desenvolvimento Dirigido por Modelos

- Problemas:
  - Codigo ineficiente.
  - Ferramentas não são mais faceis que programar.
  - Manutenção do software.

Métodos Formais

MACHINE Elevador

SEES Types

VARIABLES elev, dir, andar_dest, por_stat, est_elev

INVARIANT

elev <: ELEVADOR &
dir : elev-->DIREC &
andar_dest : elev-->ANDAR &
por_stat : elev-->P_STATUS &
est_elev : elev-->ELEV_ESTADO

INITIALISATION

ANY ll WHERE
ll <: ELEVADOR &
ll /= {}
THEN
elev := ll /* || */
/* dir := ll * {dir_sobe} /* || */
/* andar_dest := ll * {terreo} || */
/* por_stat := ll * {p_fechado} || */
/* est_elev := ll * {pronto} */
END

DEFINITIONS

ANDAR == {terreo,ultimo}

OPERATIONS

elevador_desce(ll) =
PRE
  ll : elev
THEN
  dir(ll) := desce
END;

elevador_sobe(ll) =
PRE
  ll : elev
THEN
  dir(ll) := sobe
END;

elevador_setdestino(ll,fl) =
PRE
  ll : elev &
  fl : ANDAR
THEN
  andar_dest(ll) := fl
END;

fl <-- elevador_getDestino(ll) =
PRE
  ll : elev
THEN
  fl := andar_dest(ll)
END;

Métodos Formais

- Problemas:
  - O cliente não entende a modelagem.
  - O tempo de modelagem é maior.

Métodos Ageis

- Reunião entre 17 gurus da comunidade de desenvolvimento;
- Realizada entre os dias 11 e 13 de fevereiro de 2001
- Estação de esqui nas montanhas de Utah, Estados Unidos.
- Lider principal: Kent Bech

Manifesto Agil

- **Indivíduos e interações** => mais importantes que *processos e ferramentas.*
- **Software funcionando** => mais importante do que documentação completa e detalhada.
- **Colaboração com o cliente** => mais importante do que negociação de contratos.
- **Adaptação a mudanças** => mais importante do que seguir o plano inicial.

Metodologias ágeis

- **XP (eXtreme Programming)**
- **DSDM (Dynamic Systems Development Method)**
- **ASD (Adaptive Software Development)**
- **SCRUM**
- **FDD (Feature-driven development)**

XP (eXtreme Programming)

- Projeto C3 (Chrysler) - Kent Beck, Ward Cunningham and Ron Jeffries (1996)
  - http://www.xprogramming.org
- Valores:
  - Comunicação
  - Simplicidade
  - Feedback
  - Coragem
- Práticas Principais:
  - Pair Programming, Refactoring, Simple Design, Test-driven development, User Stories, CRC
  - Coding Standard, Continuous Integration
  - Customer tests, Small Releases

DSDM (Dynamic Systems Development Method)

Método Dinâmico de Desenvolvimento de Sistemas

- Proprietária do consórcio DSDM (Reino Unido, 1994)
  - http://www.dsdm.org/
- Ciclo:
  - Estudo de viabilidade
  - Estudo do negócio (workshops)
  - 3 ciclos em paralelo, entrelaçados
    - Ciclo do modelo funcional -> análise e protótipos
    - Ciclo de design e build -> engenharia do produto
    - Ciclo de implementação -> implantação operacional
- Princípios:
  - Iterações fixas (2-6 semanas)
  - Releases freqüentes
  - Qualidade total
  - Adaptabilidade a mudanças de requisitos

DSDM

- Progenitor do XP
- Framework para desenvolvimento rápido de aplicações (RAD)
- Fixa tempo e recursos ajustando a quantia de funcionalidades – Principio de Pareto
- Pequenas equipes
- Suporta mudanças nos requisitos durante o ciclo de vida

DSDM

- Usuário sempre envolvido
- Equipe do DSDM autorizada a tomar decisões
- Foco na freqüente entrega de produtos
- Adaptação ao negócio é o critério para entregas

“Construa o produto certo antes de você construí-lo corretamente”

- Desenvolvimento iterativo e incremental
- Mudanças são reversíveis utilizando pequenas iterações
- Requisitos são acompanhados em alto nível
- Testes integrados ao ciclo de vida

ASD (Adaptive Software Development)

Desenvolvimento Adaptável de Software

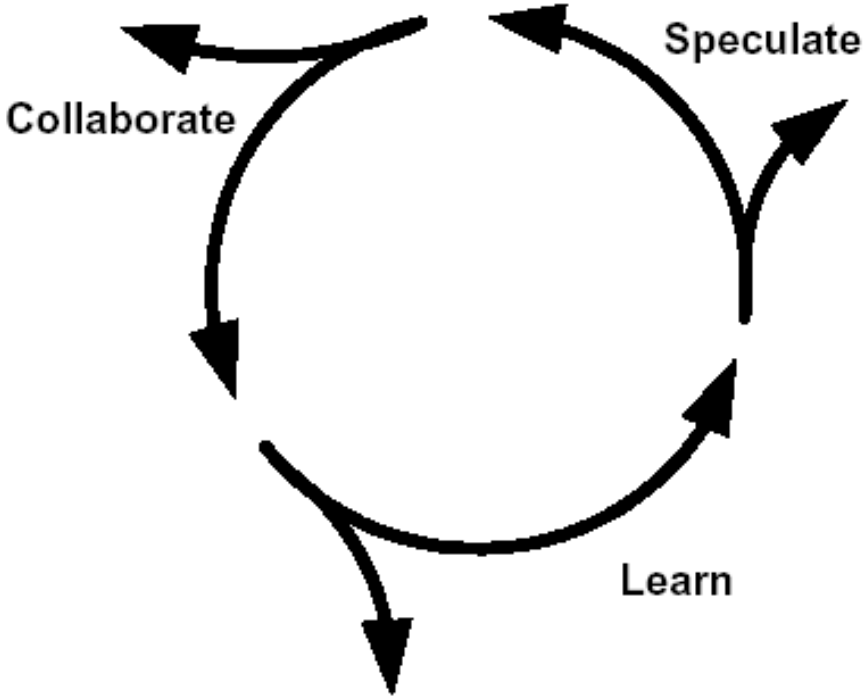
- Jim Highsmith (1997)
  - http://www.adaptivesd.com/
- Sistemas complexos => Resultados imprevisíveis
- Ciclo:

→ Especulação → Colaboração→Aprendizado

- Abordagem:
  - Do it *wrong* the first time: erre cedo, corrija cedo, não potencialize mal-entendidos.
  - *Good enough quality*: melhor compromisso entre dimensões de qualidade (extrínseca e intrínseca) para os recursos disponíveis.
  - Mecânica: RAD (rapid application development), sessões JAD (joint application development) com o cliente.

ASD

Ciclos de 3 fases



ASD

- Iterativo e incremental
- Sistemas grandes e complexos
- Framework para evitar o caos
- Cliente sempre presente:

Desenvolvimento de aplicações em conjunto (Joint Application development – JAD)

ASD

- **Especular**
  - Fixa prazos e objetivos
  - Define um plano baseado em componentes
- **Colaborar**
  - Construção concorrente de vários componentes
- **Aprender**
  - Repetitivas revisões de qualidade e foco na demostranção das funcionalidades desenvolvidas (Learning loop)
  - Presença do cliente e especialistas do domínio

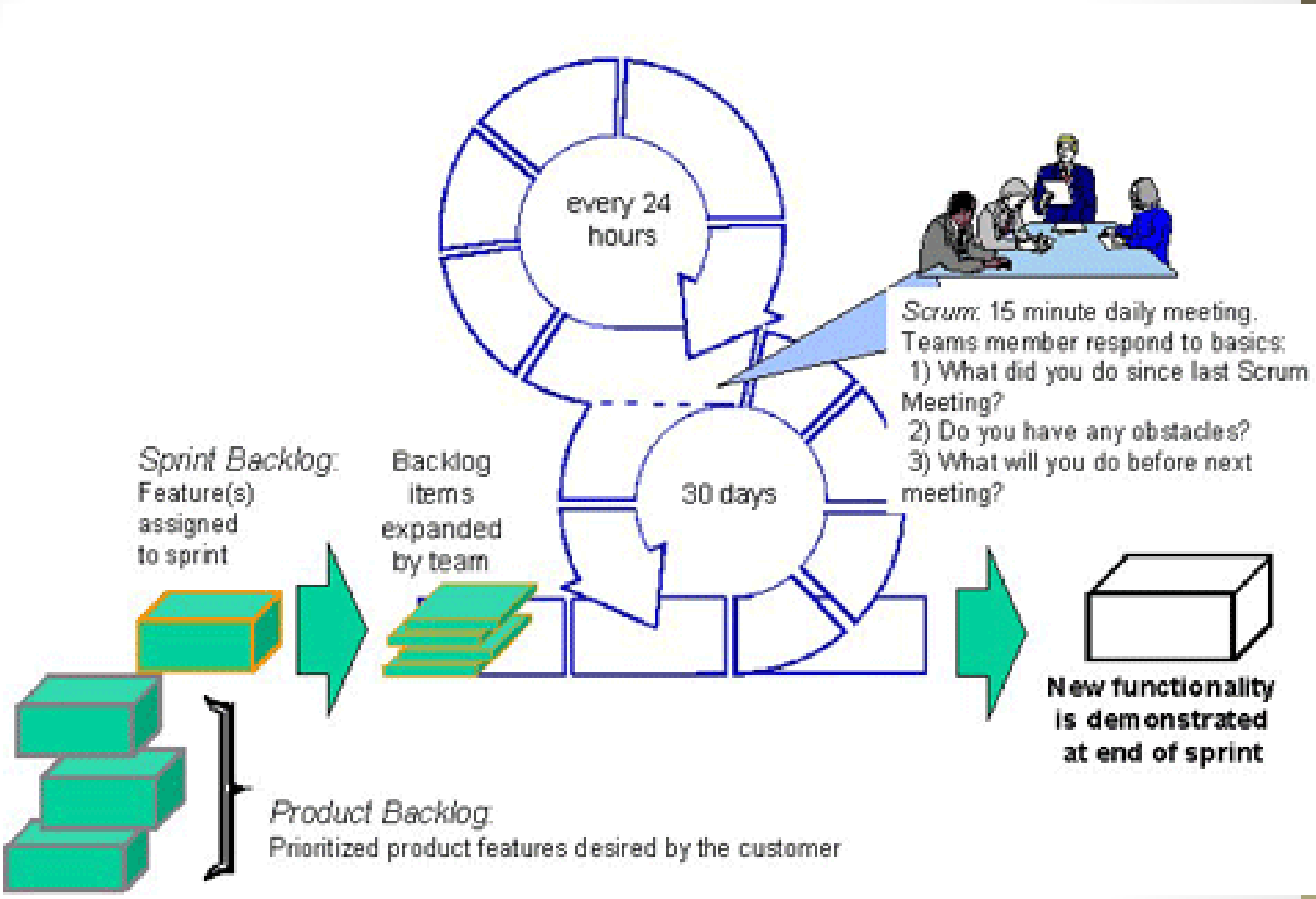
Os ciclos duram de 4 a 8 semanas

ASD

- **Orientado a missões**
  - Atividades são justificadas através de uma missão, que pode mudar ao longo do projeto
- **Baseado em componentes**
  - Construir o sistema em pequenos pedaços
- **Iterativo**
  - Desenvolvimento em cascata (Waterfall) só funciona em ambientes bem definidos e compreendidos.
  - foco em refazer do que fazer corretamente já na primeira vez.
- **Prazos pré-fixados**
- **Tolerância a mudanças (Change-tolerant)**
  - As mudanças são freqüentes
  - É sempre melhor estar pronto a adaptá-las do que controlá-las
  - Constante avaliação de quais componentes podem mudar
- **Orientado a riscos (Risk driver)**
  - Itens de alto risco são desenvolvidos primeiro

SCRUM

- Jeff Sutherland, Ken Schwaber (1993)
  - http://www.controlchaos.com/
- Sprints de 30 dias
  - Estabilizar requisitos em cada iteração
- Scrum (reunião de status) diária (15 min)
  - Guia o desenvolvimento daquele dia
- Foco em gerência e tracking
  - Pode ser combinado com métodos mais prescritivos (ex: XP@scrum)

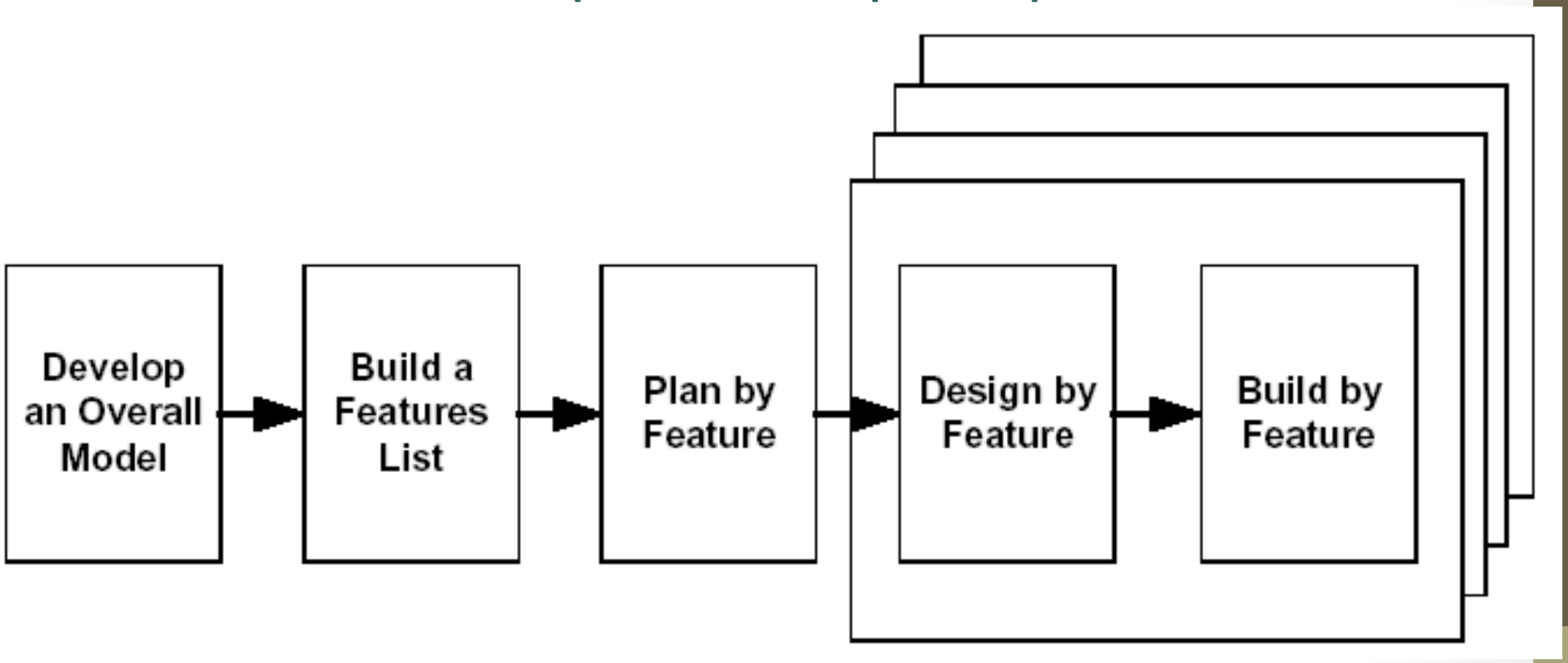

every 24 hours
30 days
Scrum: 15 minute daily meeting.
Teams member respond to basics:
1) What did you do since last Scrum Meeting?
2) Do you have any obstacles?
3) What will you do before next meeting?
Sprint Backlog:
Feature(s) assigned to sprint
Backlog items expanded by team
New functionality is demonstrated at end of sprint
Product Backlog:
Prioritized product features desired by the customer

FDD (Feature-driven development)

Desenvolvimento Orientado a Funcionalidades

- Jeff DeLuca, Peter Coad, Stephen Palmer - 2002
  - http://thecoadletter.com/download/fddguide/
- 5 processos:
  - 1- Modelo geral (arquitetura)
  - 2 -Lista de funcionalidades:
    - Levanta requisitos para todo o projeto
  - 3 – Planejamento por funcionalidades:
    - Define escopo de cada iteração (quais funcionalidades)
    - Forma times para desenvolver cada funcionalidade.
  - (A cada iteração):
    - 4 – Projeto por Funcionalidades
    - 5 - Construção por Funcionalidades

- O FDD consiste de 5 processos principais:



FDD – Boas Práticas

- **Modelagem de objetos de domínio**
  - Exploração e explicação do problema do domínio
  - Resulta em um framework
- **Desenvolver por funcionalidade**
  - Desenvolvimento e acompanhamento do progresso através de da lista de funcionalidades.
- **Proprietários de classes individuais**
  - Cada classe possui um único desenvolvedor responsável
- **Equipe de funcionalidades**
  - Formação de equipes pequenas e dinâmicas.
  - Inspeção (Inspection)
  - Uso dos melhores métodos conhecidos de detecção de erros.
- **Releases freqüentes**
  - Garantir que existe um sistema sempre disponível e demonstrável.
- **Administração de Configuração**
  - Habilita acompanhamento do histórico do código-fonte.

Próxima aula: Seminário

- **XP (eXtreme Programming)**
- **DSDM (Dynamic Systems Development Method)**
- **ASD (Adaptive Software Development)**
- **SCRUM**
- **FDD (Feature-driven development)**
- **PSP(Personal Software Process)**
- **TSP(Team Software Process)**
- **KANBAN**

Pesquisem

- A metodologia
- Ferramentas de suporte à metodologia
- Cases de sucesso
- Artigos e pesquisas relacionadas ao assunto
- Quais os pontos pesquisados pela academia nessa metodologia? Vejam isso em congressos e conferências.