



## Aula 03

---

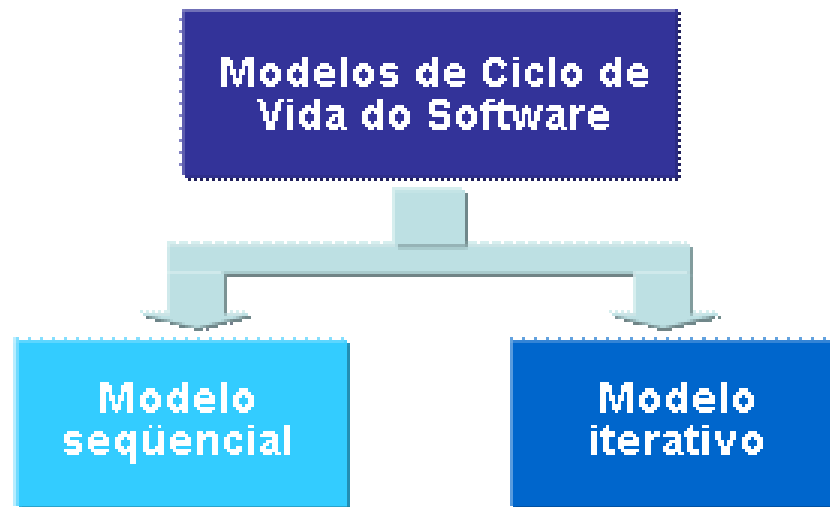
Análise de Sistemas Orientado a Objetos  
Prof. Me. Joseffe Barroso de Oliveira

UNIP - Universidade Paulista  
Análise e Desenvolvimento de Sistemas

# Modelos de Ciclo de Vida

- \* Na prática, são usados dois tipos de modelos de ciclo de vida de *software*, são eles:

- \* Sequencial
- \* Iterativo



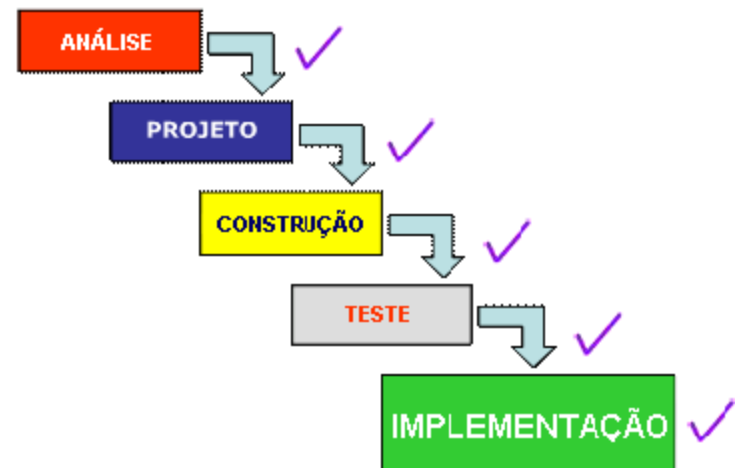
# Modelos de Ciclo de Vida “Cascata”

- \* Também conhecido como modelo sequencial, pode ser exibido de forma ilustrativa abaixo:



# Modelos de Ciclo de Vida “Cascata”

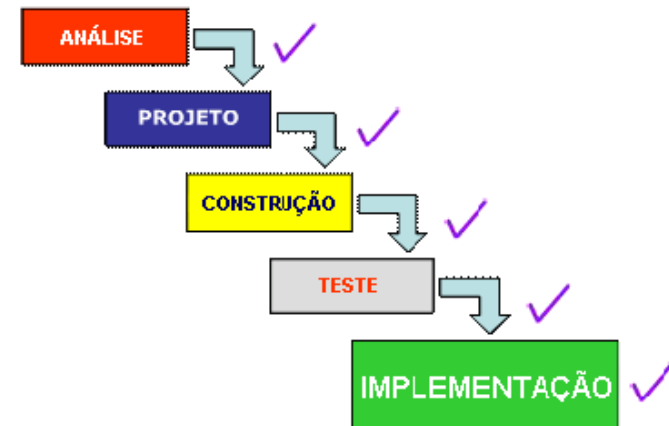
- \* Ele representa o processo de desenvolvimento como uma sequência de fases, exigindo que uma fase específica seja concluída antes da próxima ser iniciada.
- \* Devido ao reconhecimento de fases e sequenciamento, ele ajuda na finalização do contrato com referência a entrega e planos de pagamento.



# Modelos de Ciclo de Vida

## “Cascata”

- \* Na prática, é difícil usar este modelo como ele é, devido incerteza nos requisitos de *software*, que *a priori*, são difíceis de prever.
- \* Se um erro no entendimento dos requisitos for detectado durante a fase de codificação, o processo todo deverá ser reiniciado. Uma versão de trabalho do *software* não estará disponível até o final do ciclo de vida do projeto. Logo, a iteração dentro de uma fase e entre fases é uma necessidade.

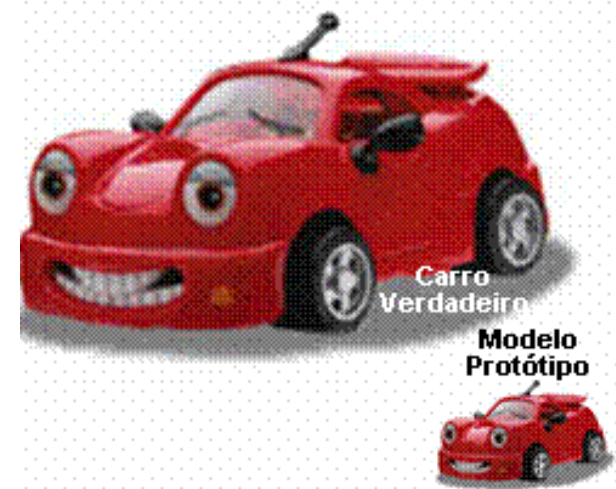


# Modelos de Ciclo de Vida

## “Prototipação”

- \* A prototipação é discutida na literatura como uma abordagem separada do desenvolvimento de *software*. Como o nome sugere, exige que uma versão de trabalho do *software* seja desenvolvida logo no início de projeto. Existem dois tipos de prototipagem, são eles:

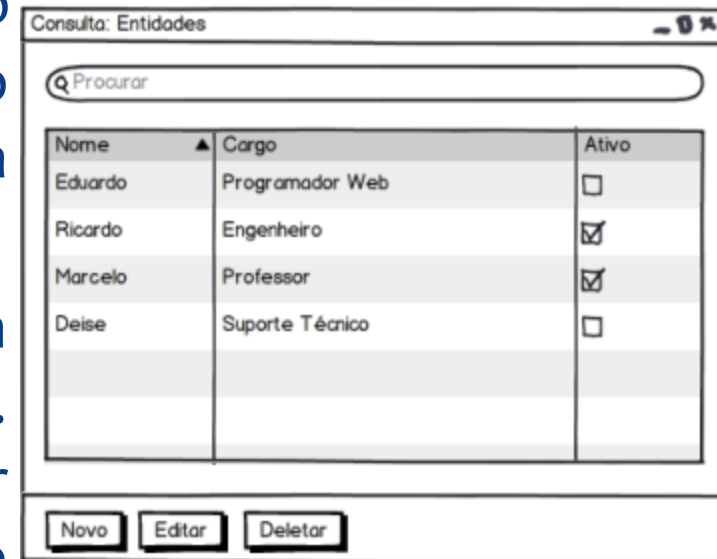
- \* Protótipo descartável
- \* Protótipo evolutivo



# Modelos de Ciclo de Vida

## “Prototipação”

- \* Finalmente, a principal vantagem dos protótipos está no fato de que o cliente consegue ter uma visão do produto logo no início do ciclo de vida do projeto.
- \* Como podemos ver, a prototipagem evolucionária é um modelo iterativo. Um modelo como esse pode ser caracterizado por fazer análise mínima, projeto, código, teste e por repetir o ciclo até a conclusão do produto.



Nome	Cargo	Ativo
Eduardo	Programador Web	<input type="checkbox"/>
Ricardo	Engenheiro	<input checked="" type="checkbox"/>
Marcelo	Professor	<input checked="" type="checkbox"/>
Deise	Suporte Técnico	<input type="checkbox"/>

# Modelos de Ciclo de Vida

## “Espiral”

- \* Barry Boehm sugeriu um modelo iterativo chamado Modelo Espiral. É como uma estrutura que precisa ser adaptada a projetos específicos.
- \* Ele permite a melhor combinação de várias abordagens e se concentra na eliminação antecipada de erros e alternativas inviáveis. Uma característica importante deste modelo é, no entanto, a ênfase em análise de risco.



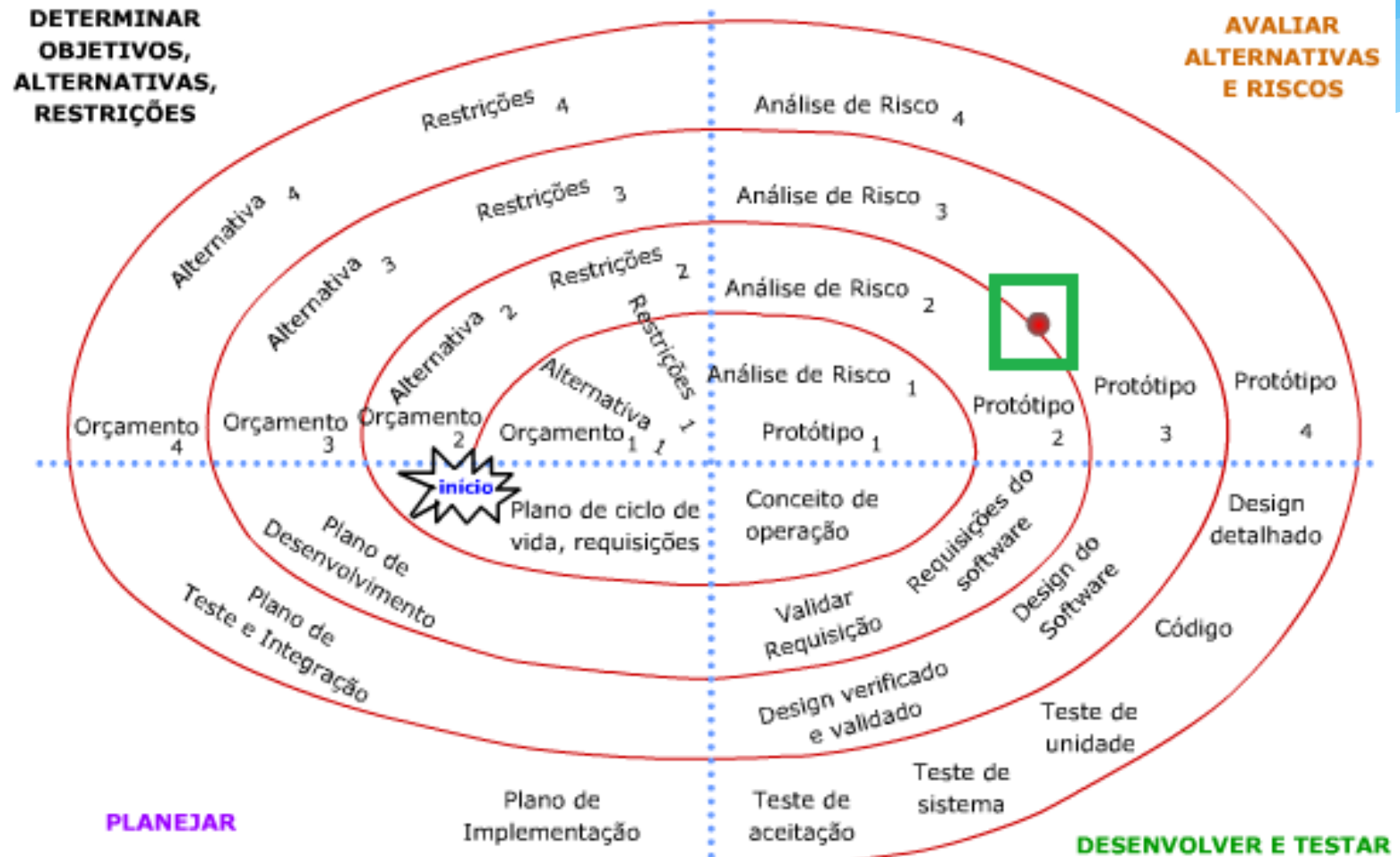
# Modelos de Ciclo de Vida

## “Espiral”

- \* Uma vez identificados os objetivos, alternativas e restrições de uma fase, os riscos envolvidos na sua execução são avaliados, resultando em uma decisão "ir, não ir".
- \* Para fins de avaliação, se pode usar prototipagem, simulações etc. Esse modelo é mais adequado para projetos que envolvam o desenvolvimento de novas tecnologias. Especialização em análise de risco é mais importante para esses projetos.

# Modelos de Ciclo de Vida

## “Espiral”



# Modelos de Ciclo de Vida

## “RUP”

- \* Entre os modelos modernos de processo, o Processo Racional Unificado (*Rational Unified Process* – RUP) desenvolvido pela Rational Corporation é digno de consideração. É um modelo iterativo que captura muitas das melhores práticas do moderno desenvolvimento de software.



# Modelos de Ciclo de Vida

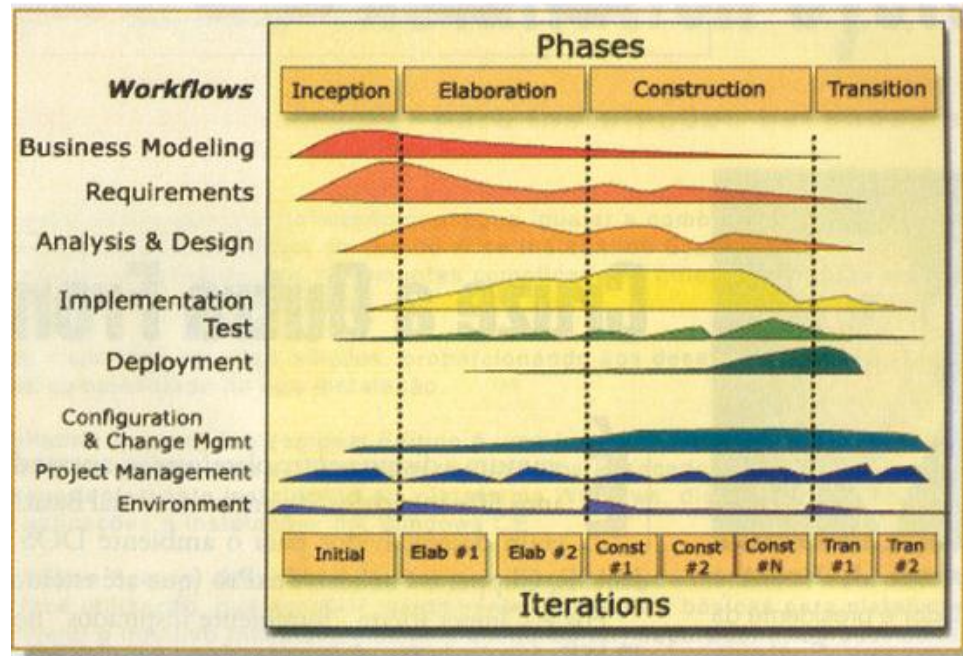
## “RUP”

- \* Cada vez mais os sistemas são complexos e precisam estar prontos em menos tempo. Mais do que isso, as necessidades mudam ao longo do tempo e a especificação de um sistema provavelmente será alterada durante seu desenvolvimento. Além disso, temos tecnologias novas (software e hardware) surgindo a cada dia. Algumas funcionam bem. Outras não. Visando atacar estes problemas, o RUP adota as seguintes premissas básicas:
- \* Uso de iterações para evitar o impacto de mudanças no projeto,
- \* Gerenciamento de mudanças e
- \* Abordagens dos pontos de maior risco o mais cedo possível.

# Modelos de Ciclo de Vida

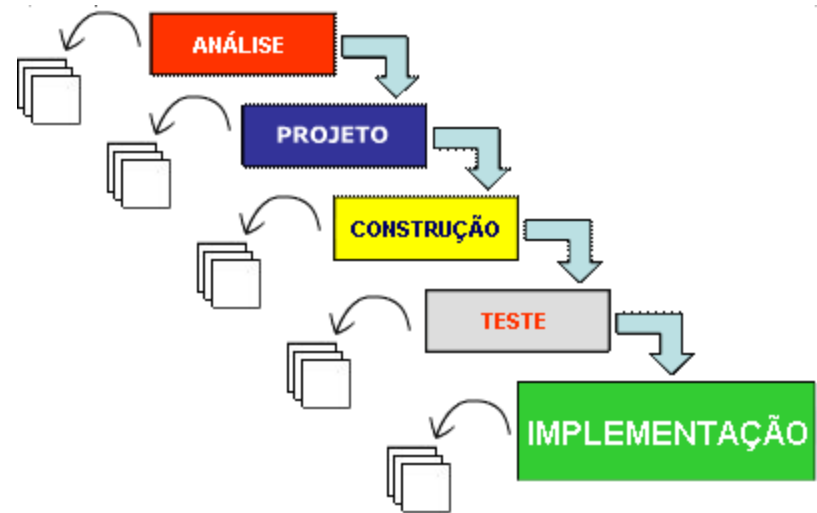
## “RUP – Estrutura”

- \* Inception - Entendimento da necessidade e visão do projeto;
- \* Elaboration - Especificação e abordagem dos pontos de maior risco;
- \* Construction - Desenvolvimento principal do sistema;
- \* Transition - Ajustes, implantação e transferência de propriedade do sistema.



# Modelos de Ciclo de Vida “Metodologias Ágeis”

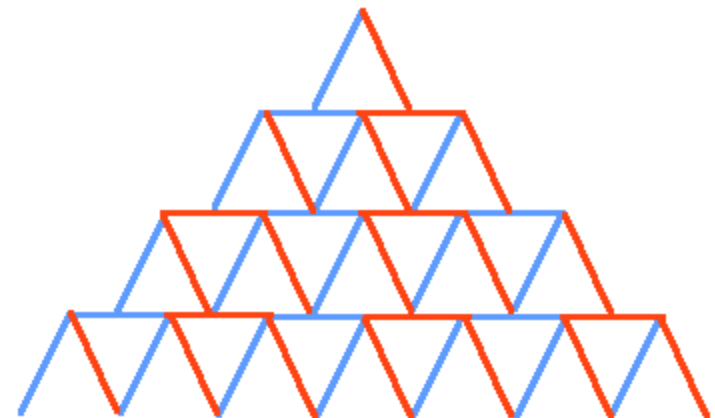
- \* Todas as metodologias descritas anteriormente são baseadas na premissa de que qualquer processo de desenvolvimento de *software* deve ser previsível e repetível. Uma das críticas contra essas metodologias é que:
- \* Há ênfase sobre procedimentos e preparação da documentação.
- \* São consideradas pesadas ou rigorosas.
- \* Enfatizam excessivamente a estrutura.



# Modelos de Ciclo de Vida

## “Metodologias Ágeis”

- \* Deste modo, as metodologias ágeis defendem o princípio da “construção curta, construção frequente”, ou seja, o projeto é dividido em subprojetos e cada subprojeto é desenvolvido e integrado ao sistema já entregue.
- \* Assim, o cliente recebe constantemente sistemas úteis e utilizáveis. Os subprojetos são escolhidos para que tenham ciclos de entrega curtos, geralmente da ordem de 3 a 4 semanas. A equipe de desenvolvimento também recebe *feedback* constante.

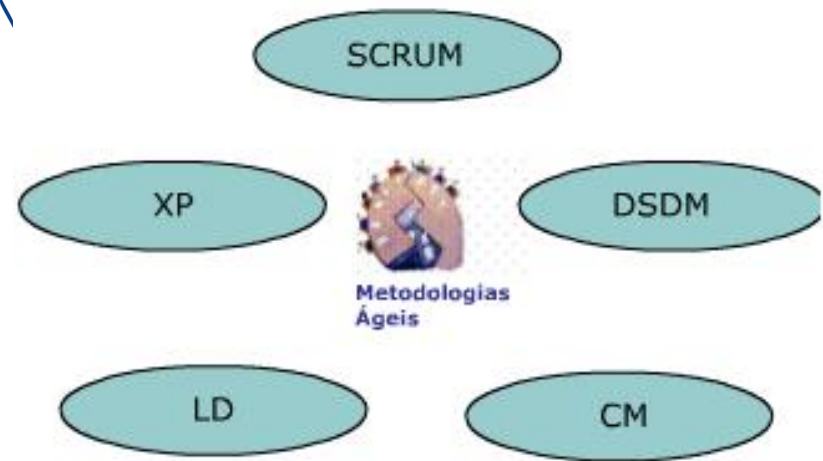


Construindo camada por camada

# Modelos de Ciclo de Vida

## “Metodologias Ágeis”

- \* Uma série de metodologias ágeis foram propostas. As mais populares entre elas são:
- \* SCRUM
- \* MÉTODO DE DESENVOLVIMENTO DE SISTEMAS DINÂMICOS (*DYNAMIC SYSTEMS DEVELOPMENT METHOD – DSDM*)
- \* MÉTODOS CRYSTAL
- \* DESENVOLVIMENTO VOLTADO A RECURSO
- \* DESENVOLVIMENTO ENXUTO (*LEAN DEVELOPMENT – LD*)
- \* PROGRAMAÇÃO EXTREME (*EXTREME PROGRAMMING – XP*).

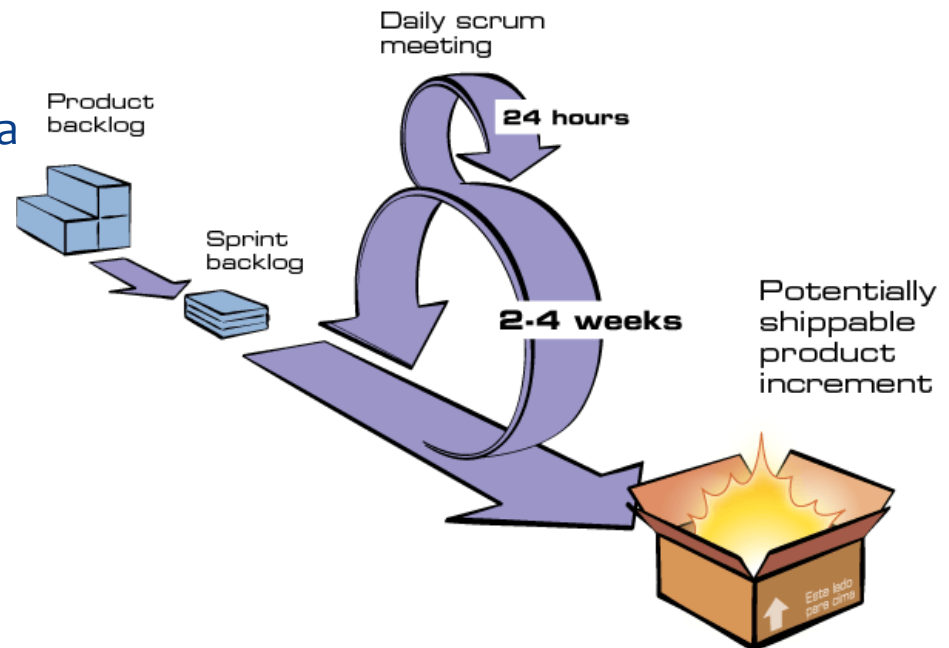




# Modelos de Ciclo de Vida

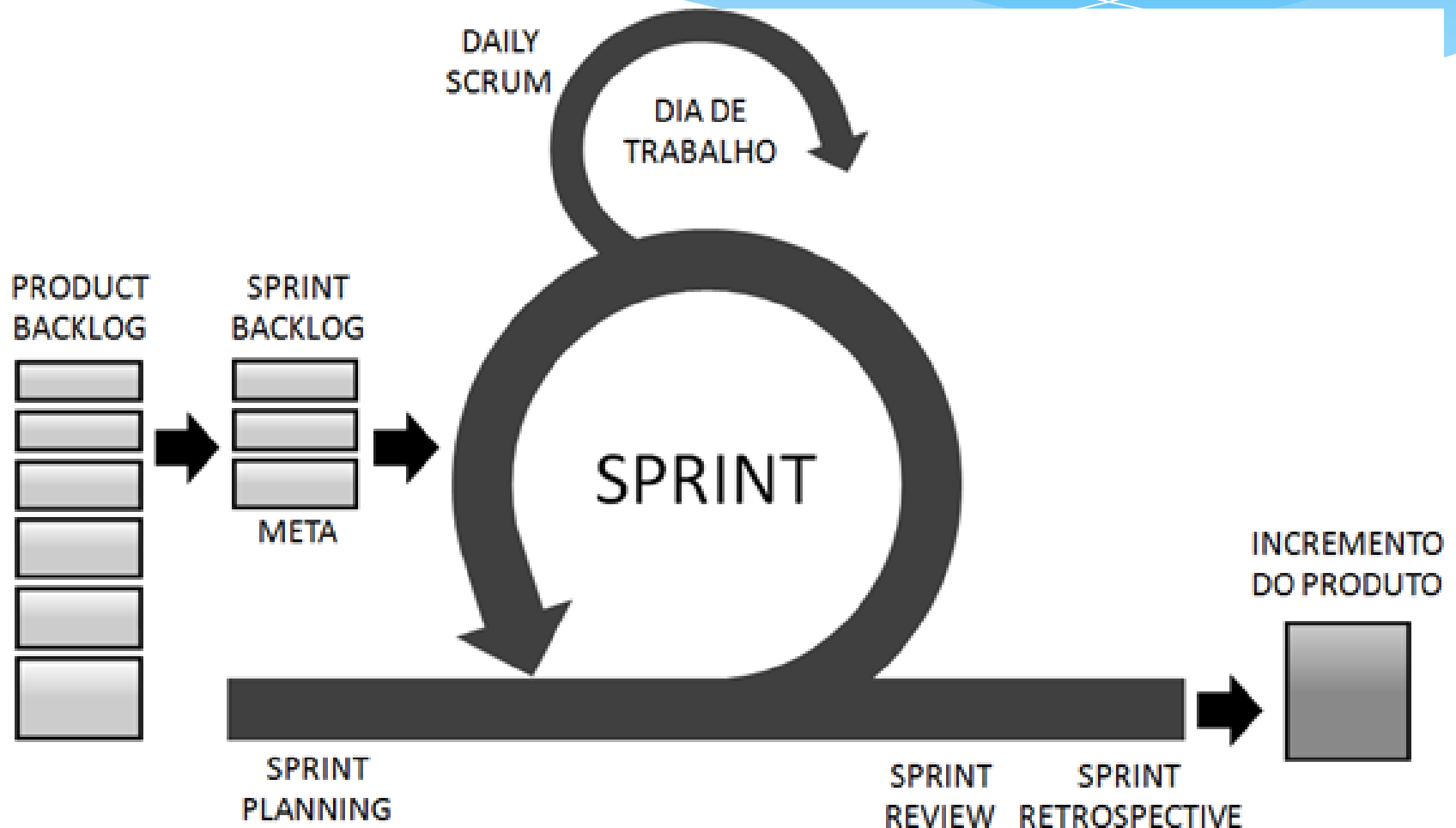
## “Scrum”

- \* Estrutura de gerenciamento de projeto.
- \* Divide o desenvolvimento em ciclos curtos chamados de “Sprint” nos quais um conjunto específico de recursos é fornecido.
- \* Defende reuniões diárias de equipe para coordenação e integração.



# Modelos de Ciclo de Vida

## “Scrum”



# Modelos de Ciclo de Vida

## “Scrum”

- \* **Product Backlog** – É uma lista organizada que contém tudo que o produto deverá ter. Sua ordenação e coerência é mantida pelo Product Owner. O backlog do produto é dinâmico e deve evoluir de acordo com a evolução do produto em si, para que se adeque ao novo formato e tenha a utilidade apropriada;
- \* **Product Owner** – O dono do produto é a pessoa responsável por gerenciar o backlog do produto. Ele acrescenta valor ao produto e ao trabalho do time de desenvolvimento. É o principal responsável por manter contato com a equipe de desenvolvedores e afirmar quais são os requisitos necessários no product backlog;
- \* **Defect Backlog** – Representa tarefas defeituosas, que por algum motivo não funcionaram. Em outras palavras, são tarefas com bugs.

# Modelos de Ciclo de Vida

## “Scrum”

- \* **Scrum Master** – Quem desempenha este papel deve garantir o progresso do projeto do produto, mantendo a comunicação com a equipe, monitorando o trabalho feito e organizando reuniões. Além disso deve garantir que cada membro envolvido no projeto tenha as ferramentas necessárias para executar seu próprio trabalho;
- \* **RoadMap do produto** – É um plano feito pelo Product Owner, que demonstra como se espera que o produto evolua ao longo do tempo;
- \* **Daily Scrum (Scrum diariamente)** – São reuniões diárias que o grupo se compromete a participar. As reuniões são feitas em pé e a ideia por trás disso é não desperdiçar tempo, então as reuniões são curtas. No Daily Scrum é muito comum que o tema abordado seja o andamento e colaboração de cada participante no projeto;

# Modelos de Ciclo de Vida

## “Scrum”

- \* **Sprint backlog ou Sprint**– Todas as atividades do projeto Scrum se encontram divididas em Sprints, que são ciclos de tarefas;
- \* **Sprint review** – É uma reunião informal, onde é feita uma revisão, sempre executada ao final de cada Sprint para avaliar o que foi feito e, caso necessário, fazer modificações no Product Backlog;
- \* **Sprint retrospective** – Acontece após o fechamento de uma Sprint com o intuito de analisar pontos positivos e negativos do que foi realizado;

# Modelos de Ciclo de Vida

## “Scrum”

- \* **Product planning** – Reunião que visa discutir e planejar os trabalhos que serão realizados nas Sprints. O conceito de time-box (Caixa de tempo) também é discutido. Time-boxes são determinações de tempo para fazer um trabalho. O tempo máximo que uma time-box pode receber é de oito horas, que pode ser aplicado à reuniões ou aos Sprints;
- \* **Release Planning** – Forma "enxuta" do backlog do produto. Os requisitos do backlog são ordenados por prioridade para depois serem divididos entre os Sprints;
- \* **Burndown chart** – É um gráfico que assegura que os Sprints estão sendo cumpridos dentro do prazo previsto. É muito importante para o time ter conhecimento do andamento do projeto e fazer ajustes, caso seja necessário.

# Modelos de Ciclo de Vida

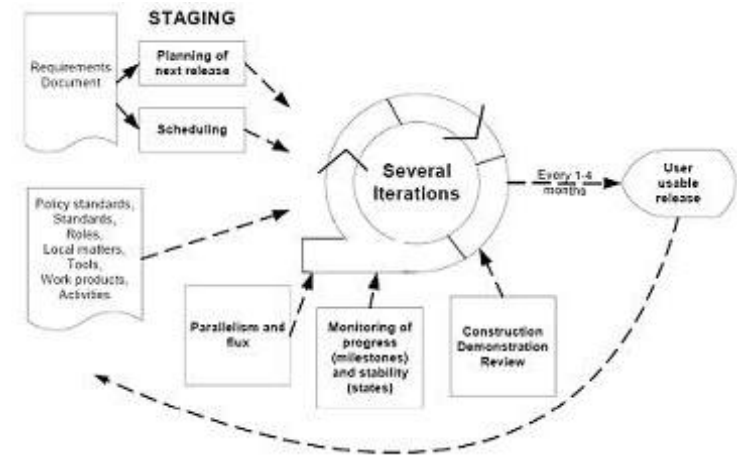
## “DSDM”

- \* Metodologia de Desenvolvimento de Sistemas Dinâmicos (do inglês Dynamic Systems Development Method - DSDM) é uma metodologia de desenvolvimento de software originalmente baseada em "Desenvolvimento Rápido de Aplicação" (RAD). DSDM é uma metodologia de desenvolvimento iterativo e incremental que enfatiza o envolvimento constante do usuário.
- \* Seu objetivo é entregar softwares no tempo e com custo estimados através do controle e ajuste de requisitos ao longo do desenvolvimento. DSDM é um dos modelos de Metodologia Ágil de desenvolvimento de software, e seu formato é propriedade da Agile Alliance.



# Modelos de Ciclo de Vida “Crystal”

- \* Conjunto de metodologias configuráveis.
- \* Elas focam nos aspectos de desenvolvimento das pessoas.
- \* A configuração é executada com base no tamanho, urgência e objetivos do projeto.
- \* Alguns dos nomes usados para as metodologias são Claro (*Clear*), Amarelo (*Yellow*), Laranja (*Orange*), *Orange web*, Vermelho (*Red*) etc.

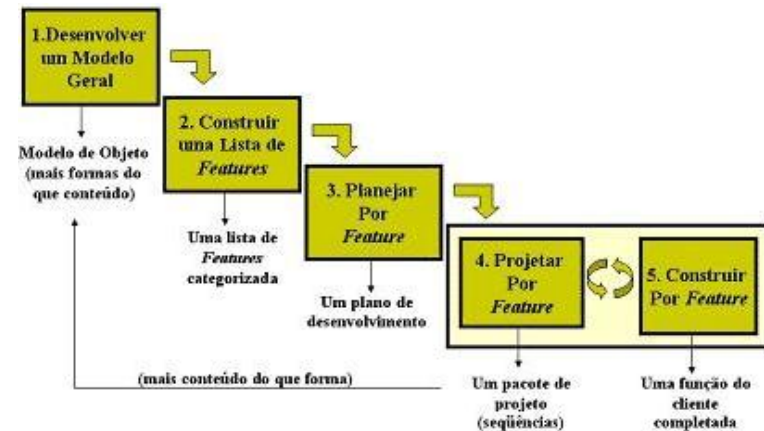




# Modelos de Ciclo de Vida

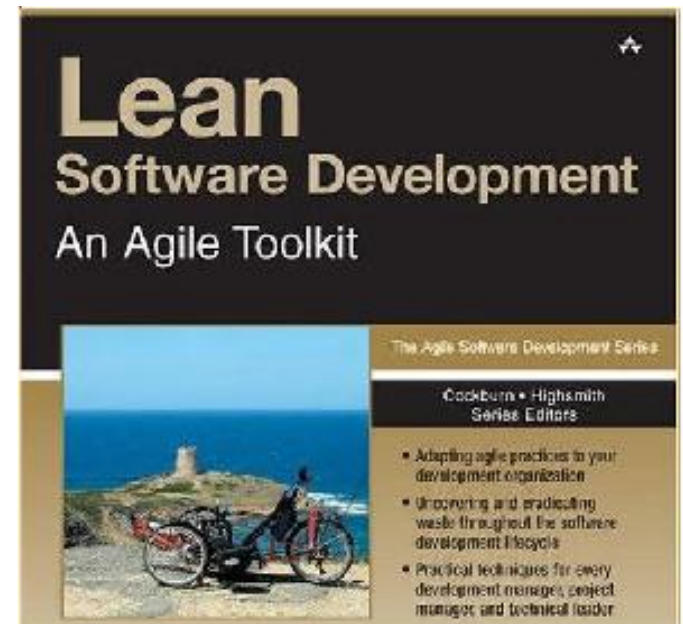
## “FDD”

- \* **Feature Driven Development** (Desenvolvimento Guiado por Funcionalidades) é uma metodologia ágil para gerenciamento e desenvolvimento de software.
- \* O desenvolvimento é voltado à funcionalidade. É definido como sendo mais apropriado a projetos iniciais, à atualização de código existente, à criação de uma segunda versão, ou ainda à substituição de um sistema inteiro em partes. Diferentemente de outros métodos ágeis, o FDD possui características específicas para desenvolver sistemas críticos.



# Modelos de Ciclo de Vida “Lean”

- \* **DESENVOLVIMENTO ENXUTO (LEAN DEVELOPMENT - LD):** Esta metodologia deriva de princípios de produção enxuta, a reestruturação da indústria manufatureira automobilística japonesa que ocorreu na década de 80. Ela se baseia nos seguintes princípios do pensamento enxuto:
  - \* Eliminar desperdícios
  - \* Ampliar o aprendizado
  - \* Decidir o mais tarde possível
  - \* Entregar o mais rápido possível
  - \* Capacitar a equipe
  - \* Construir a integridade
  - \* Enxergar o todo



# Modelos de Ciclo de Vida

## “XP”

- \* PROGRAMAÇÃO EXTREME (EXTREME PROGRAMMING - XP): Essa metodologia é provavelmente a mais popular entre as metodologias ágeis. Ela se baseia em três princípios importantes: testar primeiro, reestruturar continuamente e programar em par.



## Projeto de Programação Extreme



# Modelos de Ciclo de Vida

## “XP”

- \* Um dos conceitos mais importantes popularizados pela Programação Extreme (XP) é a programação em par. O código é sempre desenvolvido em pares. Enquanto uma pessoa insere o código, a outra revisa.



# Modelos de Ciclo de Vida

- \* O site [agilealliance.com](http://agilealliance.com) é dedicado a promover metodologias de desenvolvimento de software ágeis.

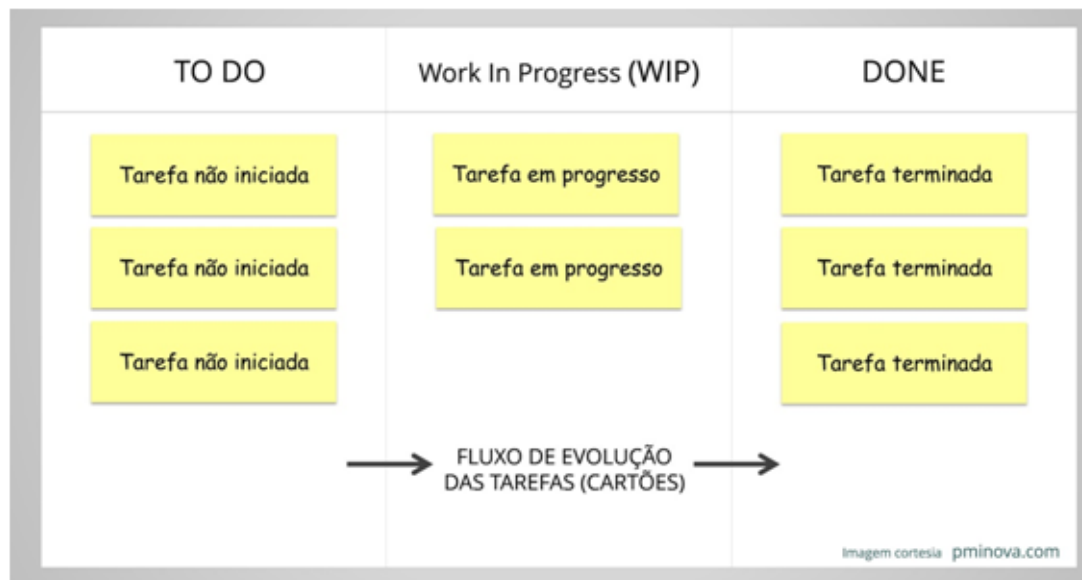
***Agile Alliance***

Satisfazer o cliente através da entrega antecipada e contínua de software de valor

# Modelos de Ciclo de Vida

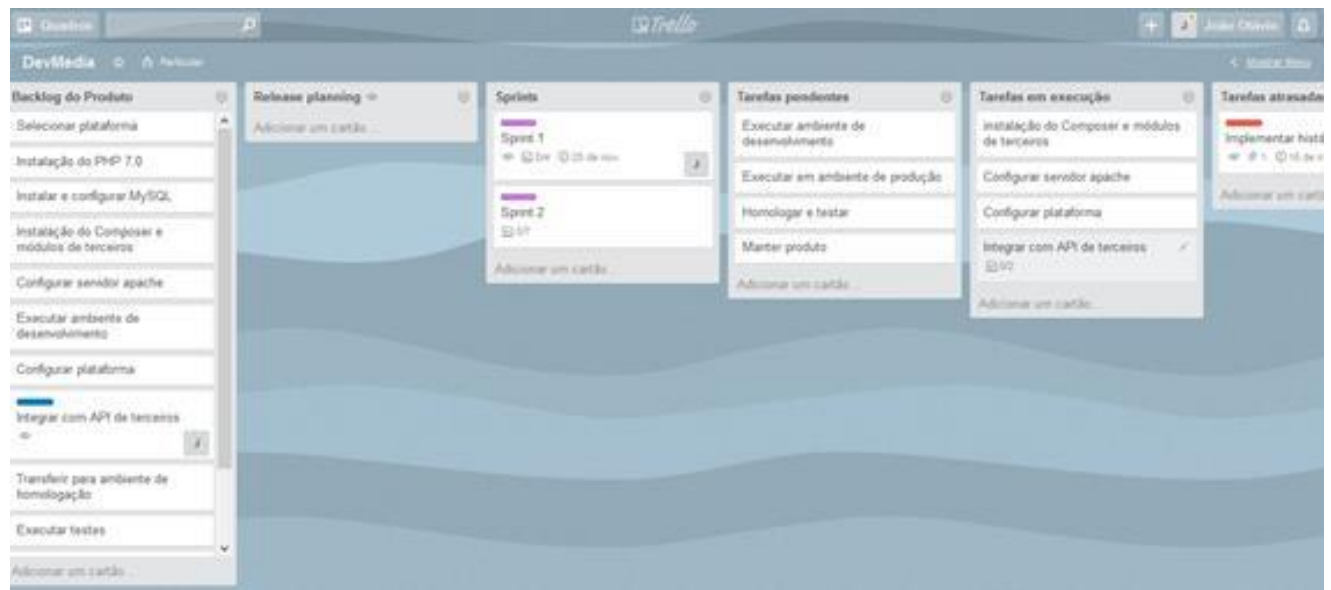
## “Kanban”

- \* É um sistema, geralmente representado por um quadro, mas também organizado através de software ou até mesmo uma folha de papel, onde cartões que representam o trabalho seguem um fluxo pré-estabelecido de estágios. Na medida em que o trabalho vai evoluindo, os cartões vão mudando de estágio, e sempre que um novo trabalho é identificado, um novo cartão é criado.



# Modelos de Ciclo de Vida

## “A Ferramenta Trello”



# Modelos de Ciclo de Vida

## “A Ferramenta Trello”

