

# Engenharia de Software

Prof. Alan Braz  
[alanbraz@gmail.com](mailto:alanbraz@gmail.com)

# Modelos de Processo de Software

Principais ciclos de  
vida de projeto para o  
desenvolvimento de  
software

***“Palavras arrumados de forma  
diferente tem um significado  
diferente e significados diferentes  
têm um efeito diferente.”***

*Blaise Pascal*



# Ciclos de Vida de Projeto



Como o cliente explicou o que queria

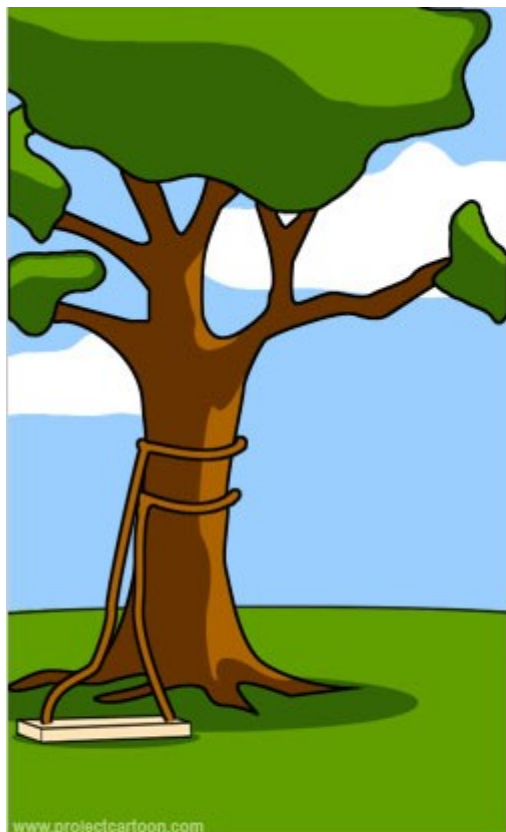


Como o Líder de Projeto Entendeu



Como o Analista Definiu

# Ciclos de Vida de Projeto



Como o programador  
escreveu

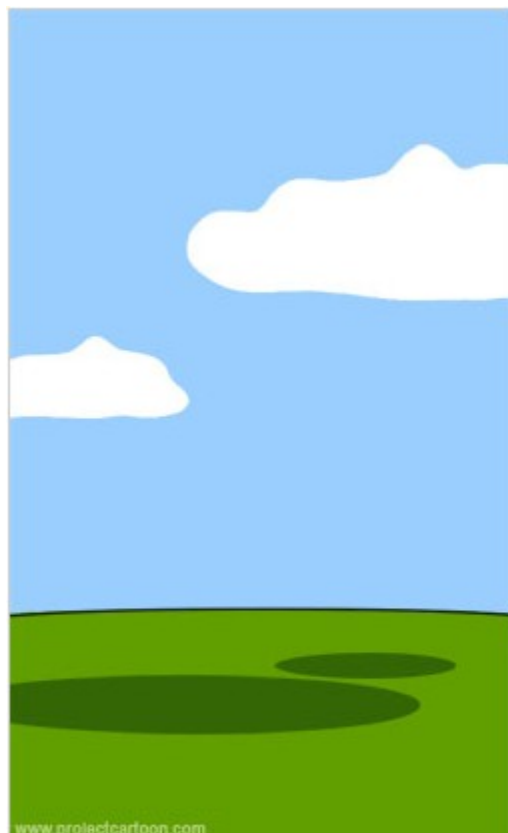


O que os testers receberam



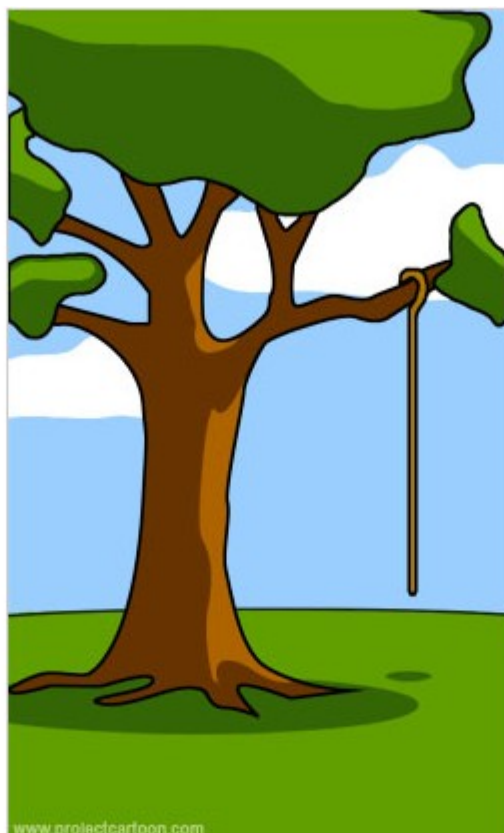
Como a área de Marketing  
vendeu

# Ciclos de Vida de Projeto



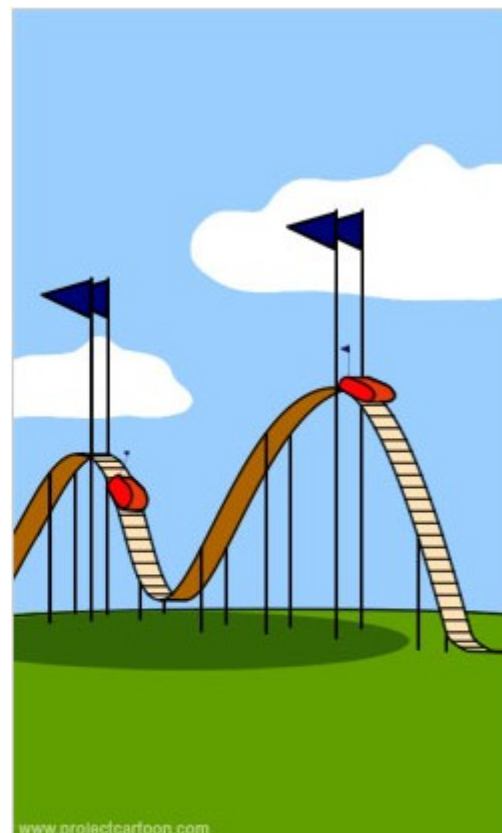
www.projectcartoon.com

Como o projeto foi  
documentado



www.projectcartoon.com

O que foi instalado

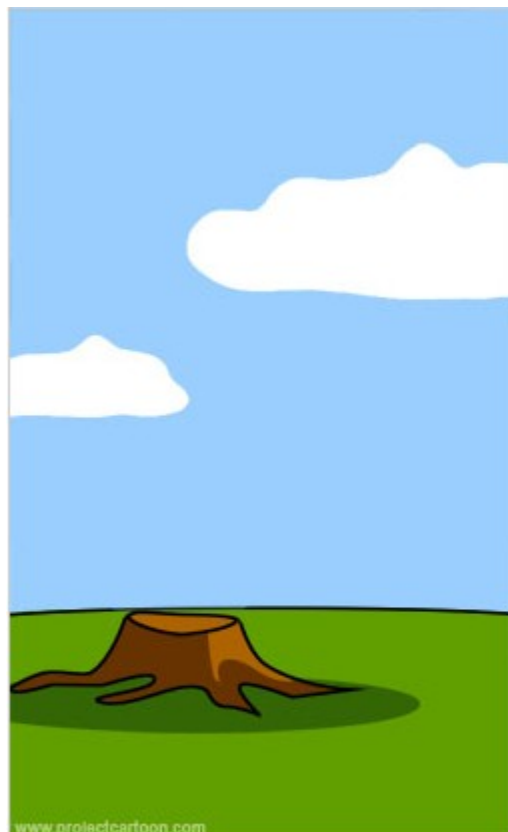


www.projectcartoon.com

Como o cliente foi cobrado



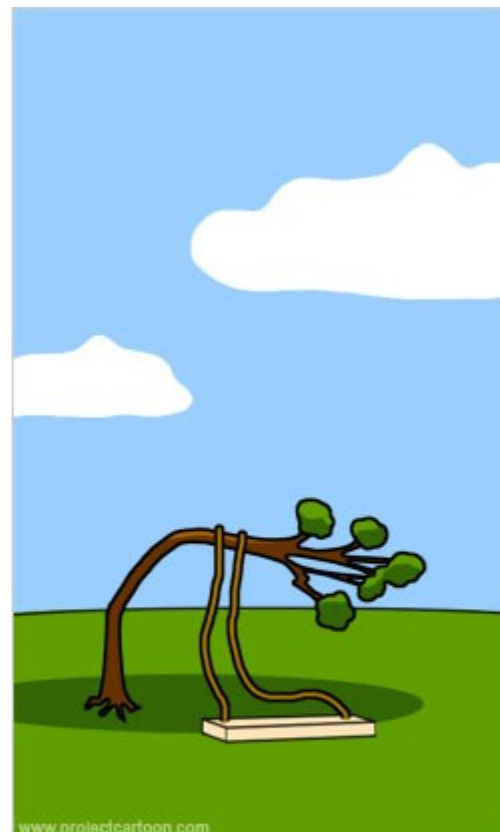
# Ciclos de Vida de Projeto



Como o Help Desk foi  
preparado para dar suporte



O que o Cliente realmente  
queria



A contingência para casos de  
Disaster Recovery

# Ciclos de Vida de Projeto



Como é a opção Open Source disponível no mercado



Como ela se comporta quando submetida a carga real



Como os Patches foram aplicados

[www.projectcartoon.com](http://www.projectcartoon.com)



# Desenvolvimento de Software

- Atividades e passos

- Modelos

- Disciplinas de suporte

- Ferramentas

## Software development process

### Activities and steps

Requirements • Specification  
Architecture • Design  
Implementation • Testing  
Deployment • Maintenance

### Models

Agile • Cleanroom • DSDM  
Iterative • RAD • RUP • Spiral  
Waterfall • XP • Scrum • Lean  
V-Model • FDD

### Supporting disciplines

Configuration management  
Documentation  
Quality assurance (SQA)  
Project management  
User experience design

### Tools

Compiler • Debugger • Profiler  
GUI designer  
Integrated development environment

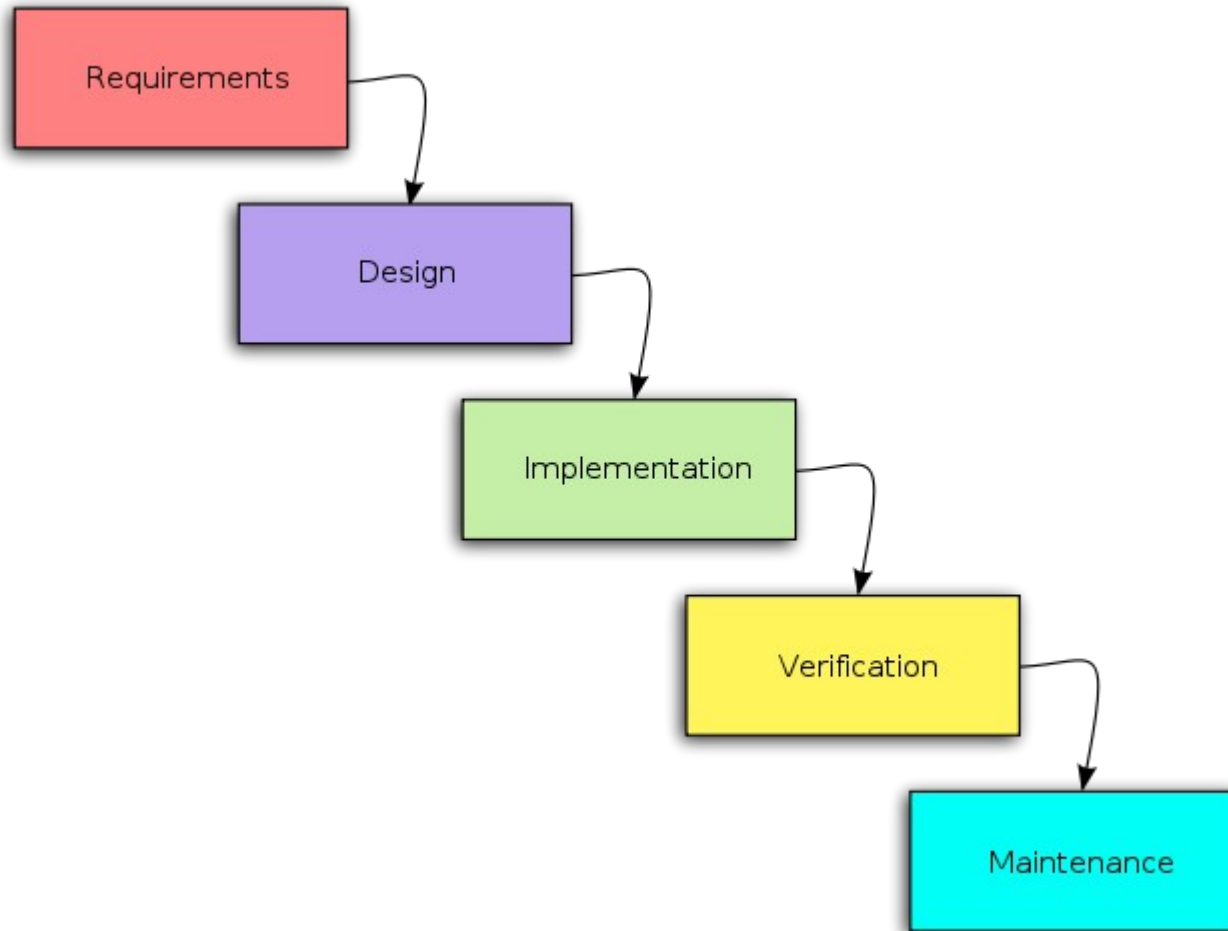
Fonte: [http://en.wikipedia.org/wiki/Software\\_development\\_process](http://en.wikipedia.org/wiki/Software_development_process)

# Ciclos de Vida de Projeto

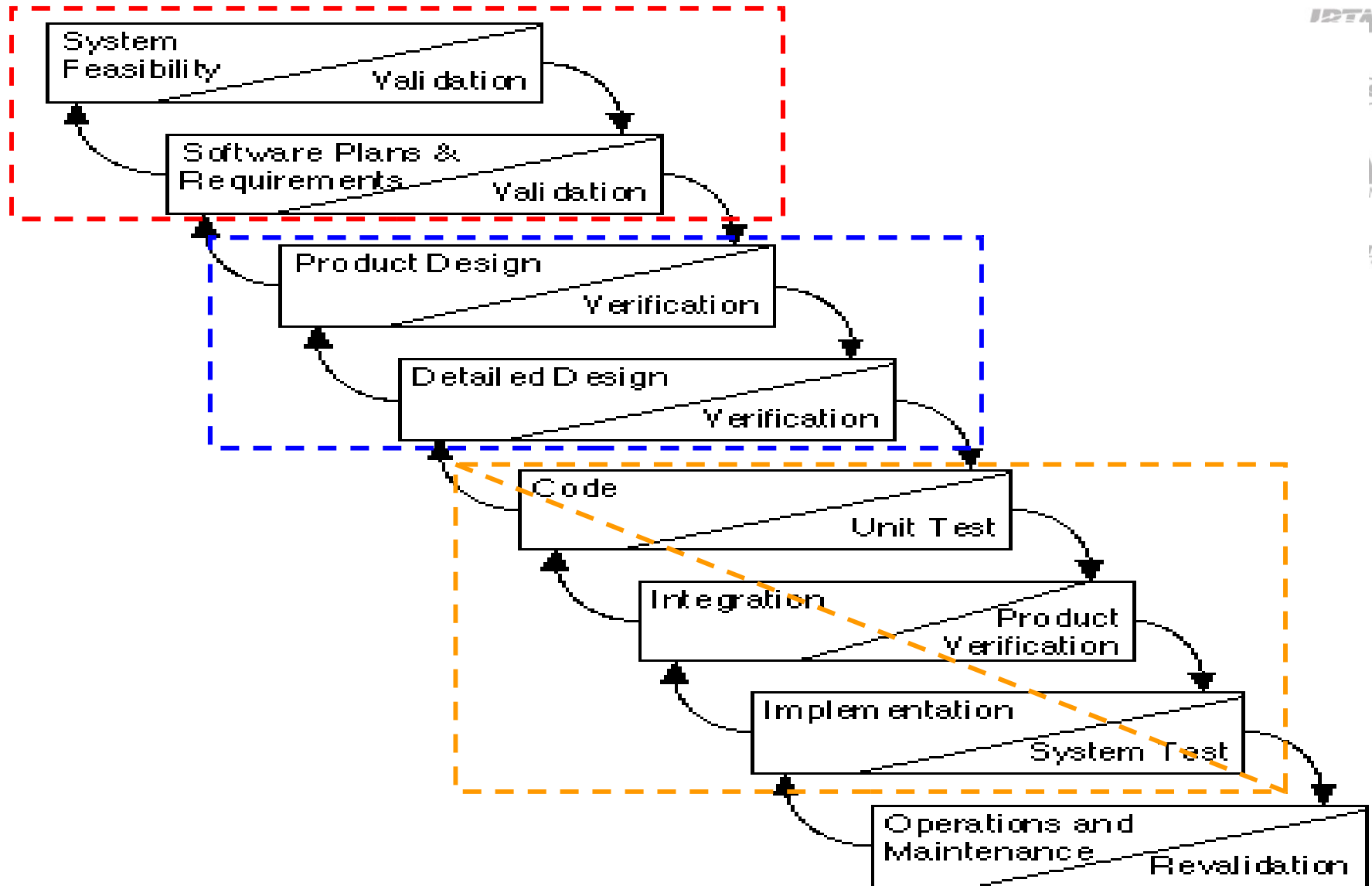
• Alguns dos ciclos de vida de projeto mais conhecidos são:

- Cascata Simples (Waterfall)
- Espiral / Win-Win Espiral
- RUP
- Ágil / Scrum

# Waterfall Model



# Modelo Cascata Simples - Waterfall



# Modelo Cascata Simples - Waterfall



- A 1ª descrição formal do modelo foi um artigo publicado em 1970 por **Winston W. Royce**
- Uma fase é executada de cada vez
- Cada fase termina com uma:
  - Verificação – “**fizemos certo o software?**”
  - Validação – “**fizemos o software certo?**”
- O produto passa por cada fase seqüencialmente e produz uma *baseline* incremental
- Ocorrem iterações em cada fase até que a verificação e validação sejam satisfeitas
- Problemas:
  - Iteratividade do ciclo nem sempre é visível
  - Ênfase no *finish-to-start* dos produtos às vezes aumenta riscos



# Modelo Cascata Simples - Waterfall

- No modelo original de Royce foram descritas as 7 fases abaixo:

1. Requisitos do Sistema
2. Requisitos do Software
3. Análise
4. *Design*
5. Codificação
6. Testes e debugging
7. Operação

# Waterfall

- Argumentos favoráveis:

- Planejamento rigoroso
  - Redução de risco (?)
- Avanço seguro
- Documentação robusta
- Abordagem simples
  - Disciplinada
  - Estruturada
- “*Milestones*” bem definidos

# Waterfall

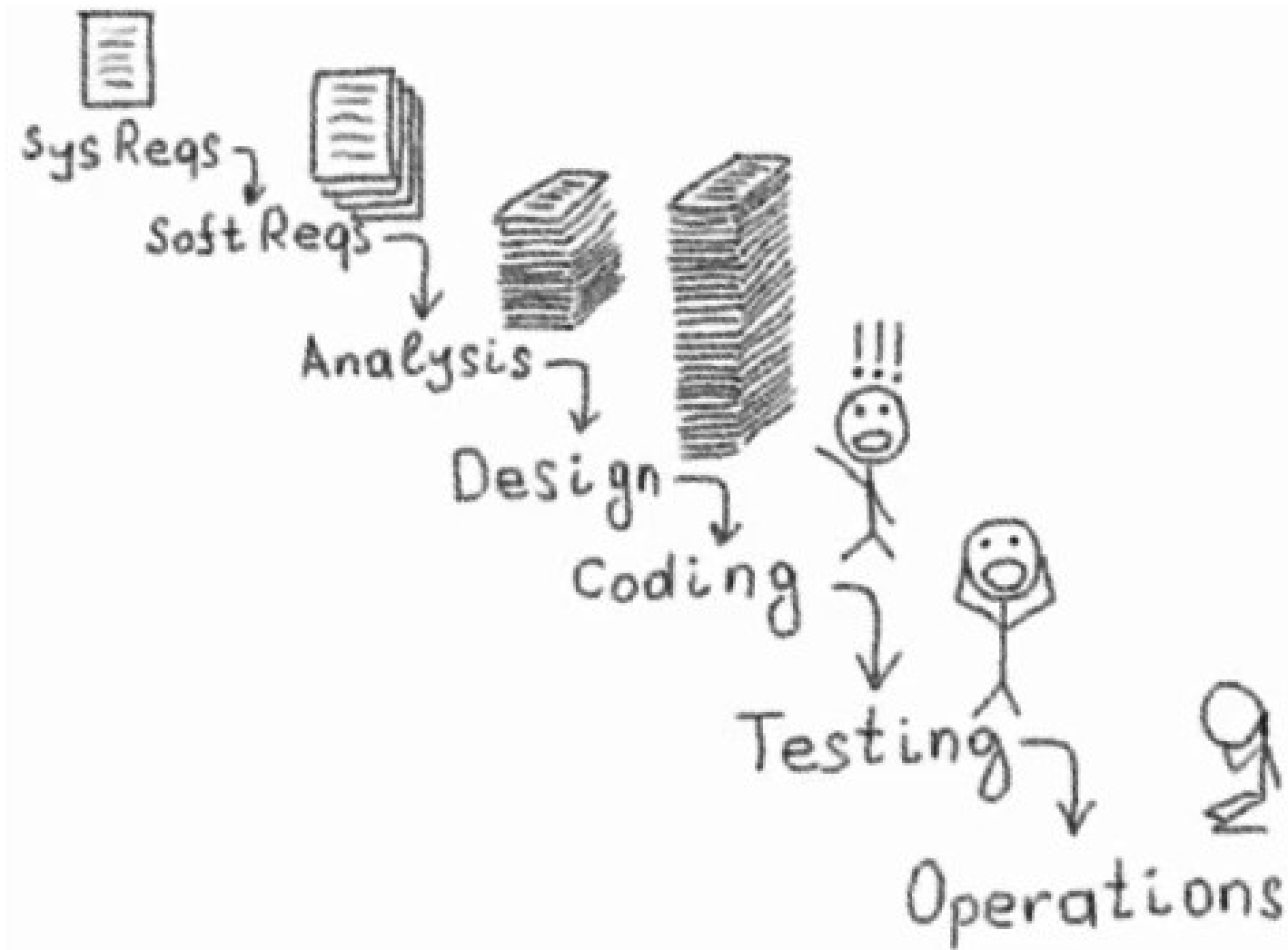
## •Críticas:

- Na prática é impossível
- Mudança dos requisitos do cliente
- Refém dos “*wicked problems*” (perversos)

*“Muitos dos detalhes [do sistema] só se tornam conhecidos por nós durante o progresso na implementação [do sistema]. Algumas das coisas que aprendemos invalida nosso projeto e temos de recuar.”*

David Parnas - “A Rational Design Process: How and Why to Fake It”

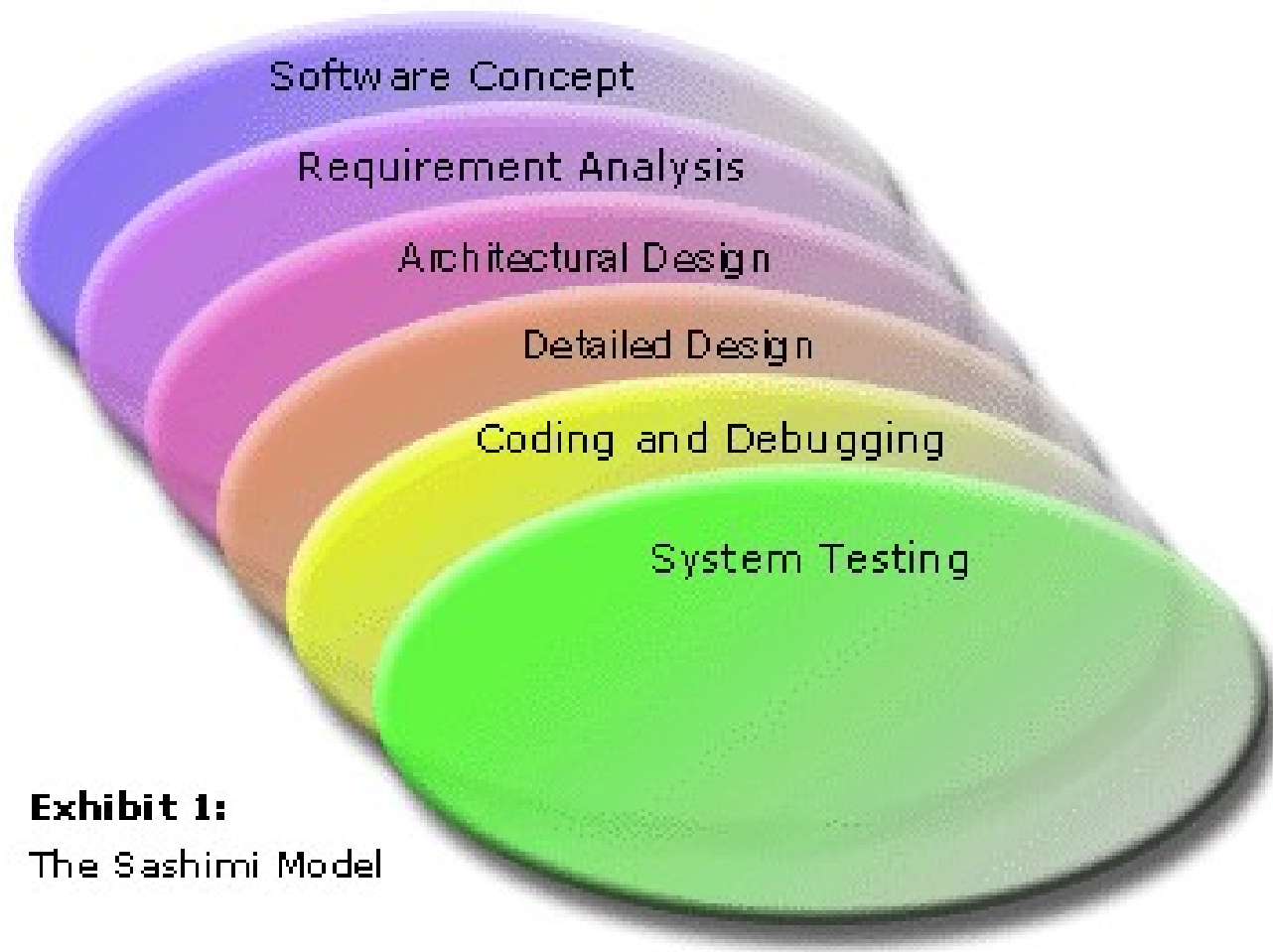
- Modelos derivados: The Sashimi Model



The Rise And Fall Of Waterfall

<http://www.youtube.com/watch?v=X1c2--sP3o0>

# Sashimi Model

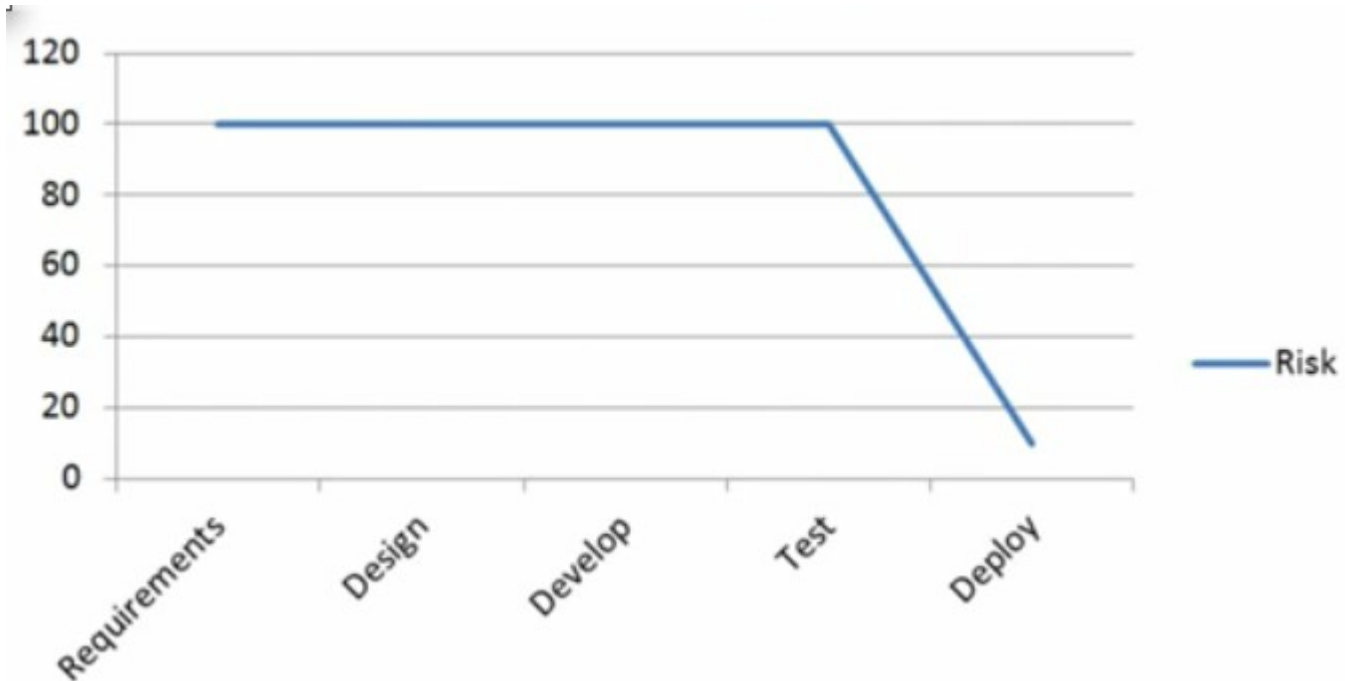


**Exhibit 1:**

The Sashimi Model



# Risco no ciclo de vida Cascata



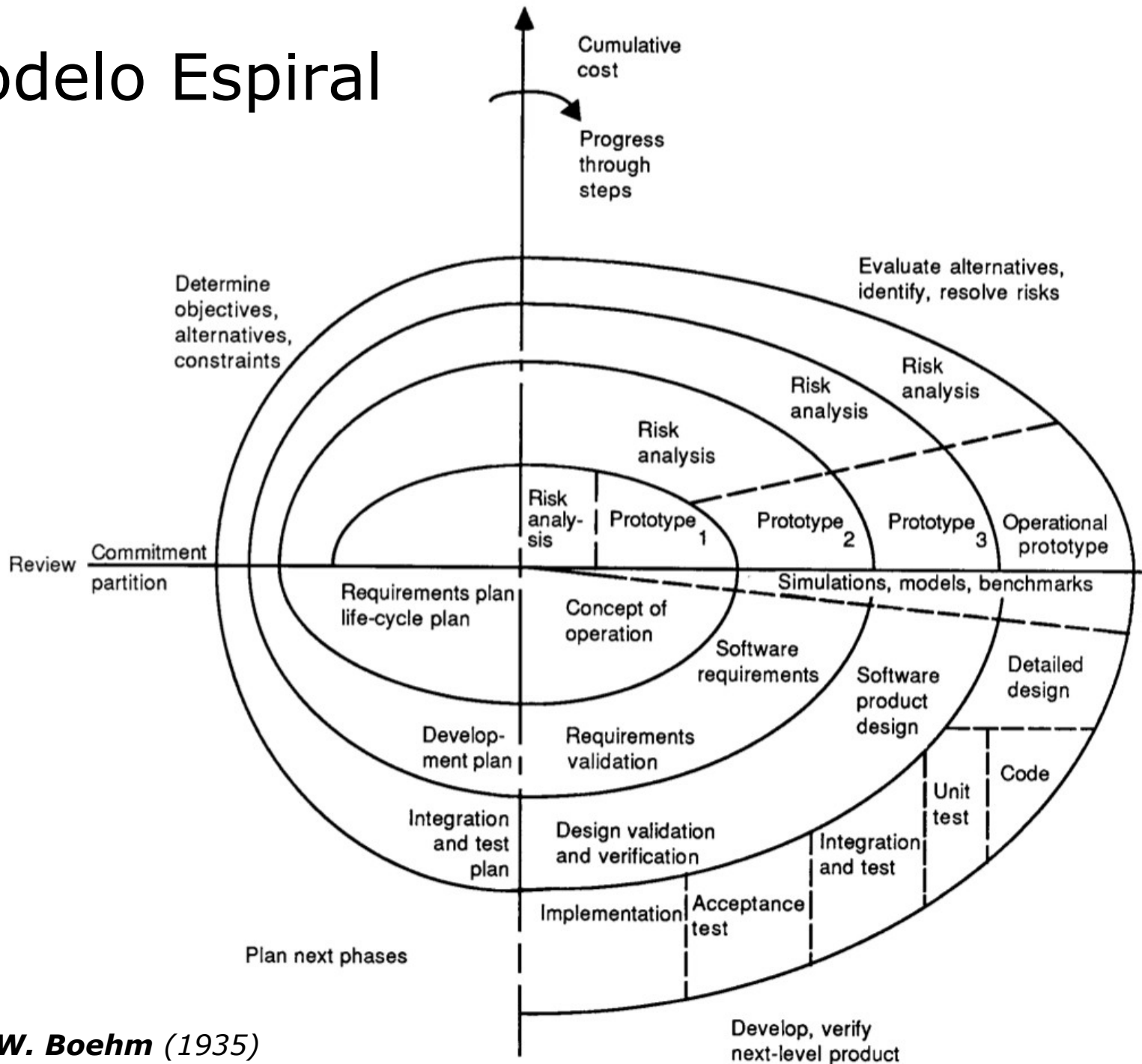
Waterfall Software Life Cycle Model Features and Risks  
<http://www.youtube.com/watch?v=p6vW84Pq-Uc>

# Risco no ciclo de vida Iterativo



Waterfall Software Life Cycle Model Features and Risks  
<http://www.youtube.com/watch?v=p6vW84Pq-Uc>

# Modelo Espiral



# Modelo Espiral

- Barry Boehm, Maio de 1988, "A Spiral Model of Software Development and Enhancement"
  - "Computer", "IEEE", 21(5):61-72
- Orientado à gestão de Risco (Risk-driven)
- Prototipação
- Se no ciclo atual foram tratados todos os riscos o próximo ciclo segue um modelo waterfall
- Assemelha-se a diferentes tipos de modelos em função do perfil de risco

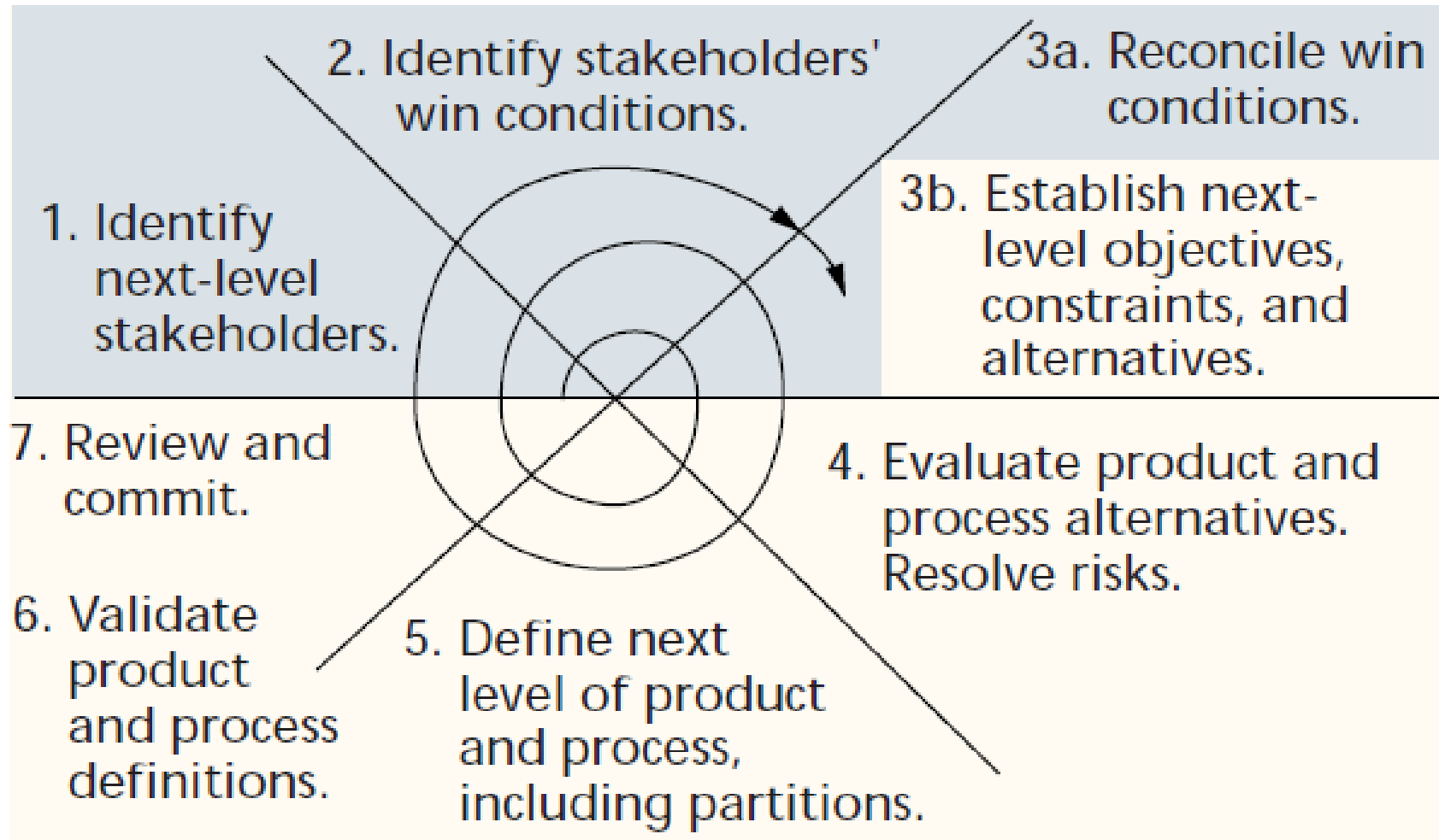
# Modelo Espiral

## •Usando o modelo Espiral:

- Objetivos
- Restrições
- Alternativas
- Riscos
- Resolução dos Riscos
- Resultados da Resolução dos Riscos
- Plano para a próxima fase
- Compromisso



# Modelo Win-Win Espiral



# Modelo Win-Win Espiral

## •Evolução do Modelo Espiral

- Barry Boehm, Alexander Egyed, Julie Kwan, Dan Port, Archita Shah, Ray Madachy
- **Teoria W**, uma abordagem e teoria de gestão, que prega que todas as principais partes interessadas devem ganhar. Isso é uma condição necessária e suficiente para o sucesso do projeto.
- O modelo **WinWin espiral**, que estende o modelo Espiral de desenvolvimento de software adicionando à Teoria W à frente das atividades de cada ciclo.
- **WinWin**, uma ferramenta “groupware” que torna mais fácil para “stakeholders” distribuídos negociar mutuamente de forma satisfatória (win-win) as especificações do sistema.

# Modelo Win-Win Espiral

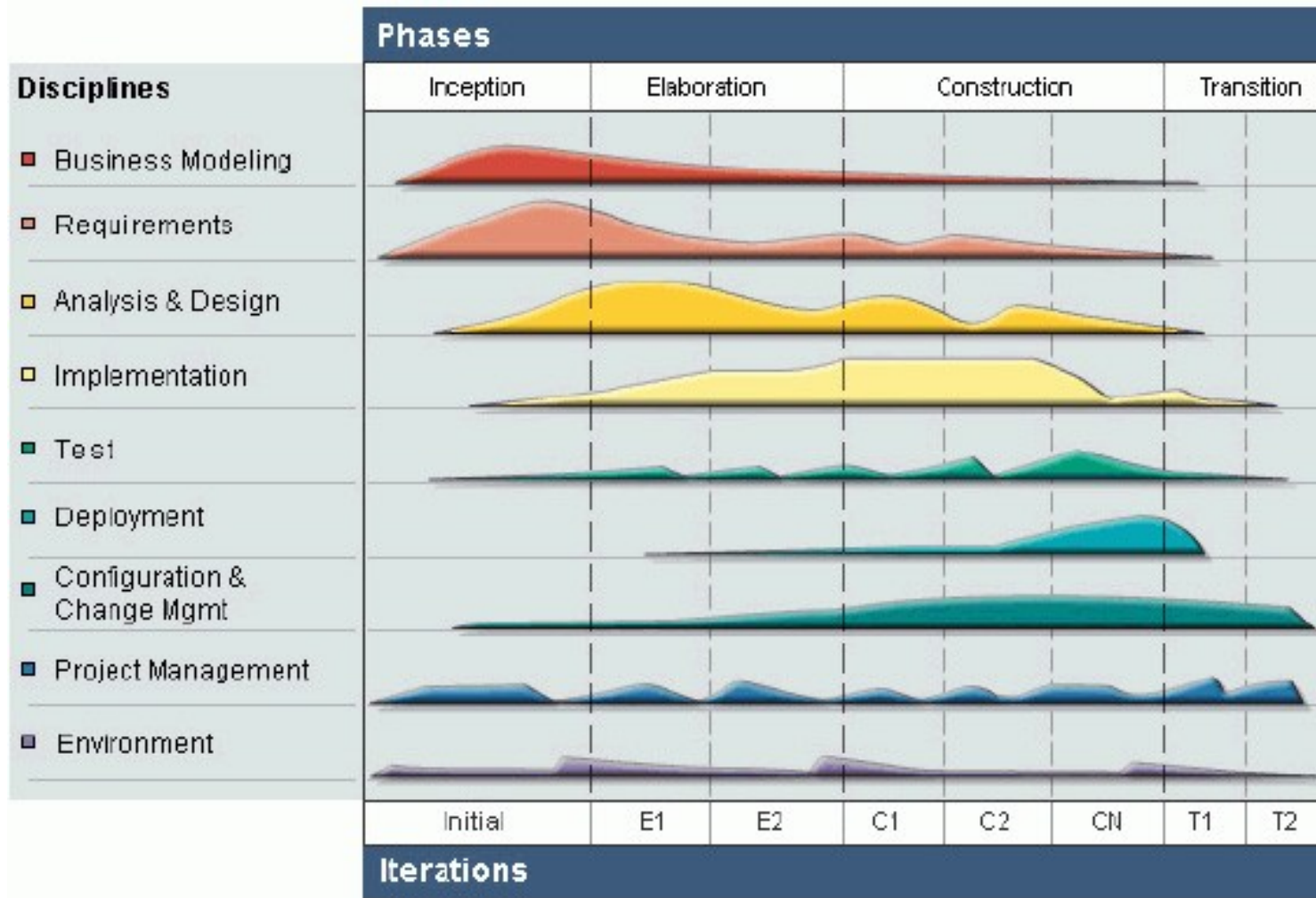
## •Diferenças:

- Identificação de Stakeholders
- Identificação das condições de “ganha-ganha”
- Negociação e reconciliação dessas condições

## •Problemas:

- Requer mais tempo para gestão de requisitos
- Dificuldade em adequar interesses e agendas dos envolvidos

# IBM Rational Unified Process



# O que é o RUP?



- É um modelo de processo
  - Incremental
  - Iterativo
  - Dirigido por casos de uso
  - Centrado em arquitetura.
- Bidimensional.
- É um framework de boas práticas.

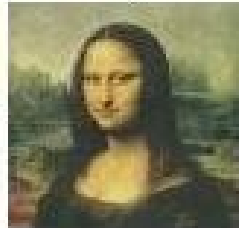


# Incremental vs Iterativo

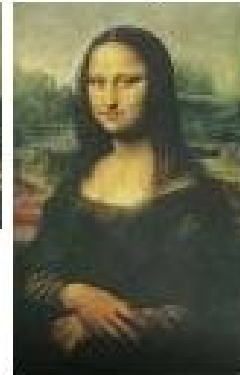
Delivery 1



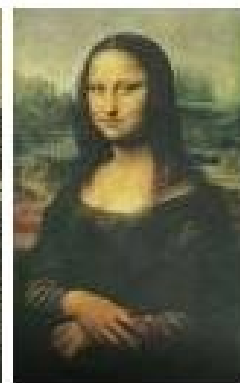
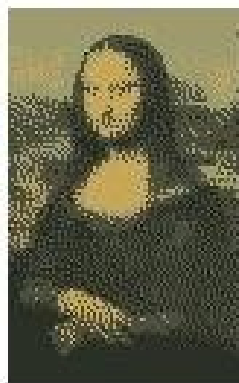
Delivery 2



Delivery 3



Incremental plan



Iterative plan

Fonte:

<http://www.agileway.com.br/2009/08/18/incremental->

# O que é o RUP?

## •Fases

### •Iniciação

- Planejamento do projeto

### •Elaboração

- Elaboração da arquitetura
- Mitigação dos riscos

### •Construção

- Implementação
- Codificação

### •Transição

- Implantação do sistema em ambiente de homologação/produção
- Projeto piloto
- Homologação

# Disciplinas do RUP

É um conjunto de atividades relacionadas a uma 'área de interesse' importante em todo o projeto.

## • Disciplinas Operacionais:

- Modelagem do Negócio
- Requisitos
- Análise-Design
- Implementação
- Teste
- Implantação

## • Disciplinas Gerenciais

- Gestão de Configuração e Mudança
- Gestão do Projeto
- Ambiente

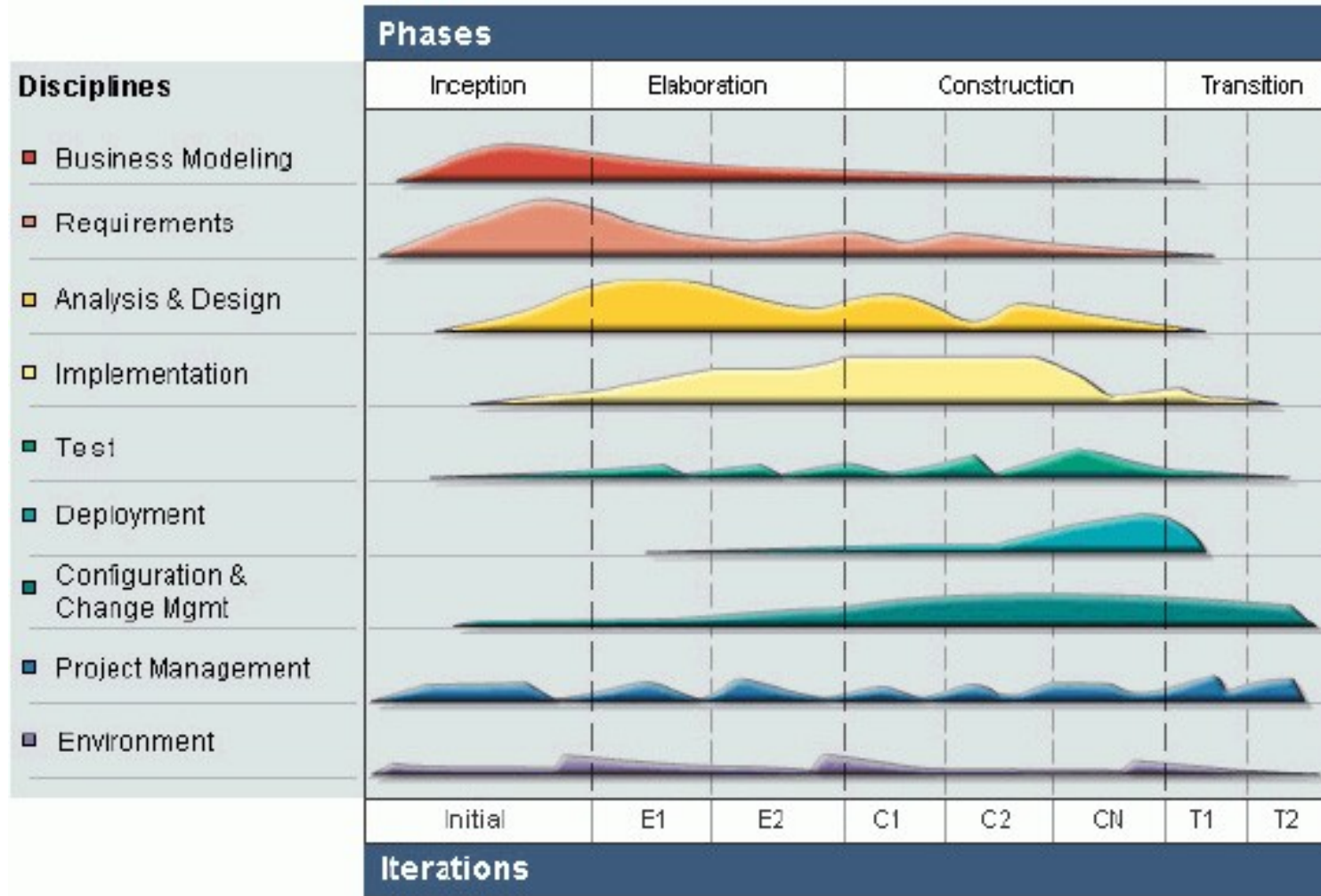
# Disciplinas do RUP

- O conjunto das disciplinas oferece uma visão semelhante ao processo sequencial clássico.
- As disciplinas podem ser executadas em paralelo.
  - Exemplo:  
A disciplina de Requisitos pode ser executada em paralelo com a disciplina de Análise-Design.

# Bidimensionalidade do RUP

TEMPO

WORKFLOW



**Gráfico de Baleias → Expressa a intensidade de esforço em cada disciplina ao longo do tempo**

# Ágil



- Inicialmente, métodos ágeis eram conhecidos como métodos leves.
- Em 2001, membros proeminentes da comunidade se reuniram em Snowbird e adotaram o nome métodos ágeis = Manifesto ágil - [www.manifestoagil.com.br](http://www.manifestoagil.com.br)
- 12 princípios e práticas desta metodologia
- Os métodos ágeis iniciais:
  - Scrum (1986)
  - XP (1996)
  - Crystal Clear
  - Feature Driven Development
  - Entre outras...

*Introduction to Agile for Traditional Project Managers*

<http://www.infoq.com/presentations/Introduction-Agile-Stacia-Broderick>

# Manifesto Ágil

Valorizar...

**Indivíduos e  
interação entre eles**

mais que

**Processos e ferramentas**

**Software em  
funcionamento**

mais que

**Documentação abrangente**

**Colaboração com  
o cliente**

mais que

**Negociação de contratos**

**Responder a  
mudanças**

mais que

**Seguir um plano**

# Em poucas palavras...

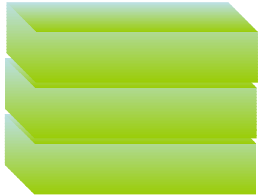
- Usar **feedback contínuo** dos stakeholders para desenvolver **código funcional** com alta-qualidade através de “Histórias de Usuários” (**user stories**) em uma série de **iterações curtas** e bem delimitadas.



# Fluxo do Processo Ágil

## Pré-Concepção

Product Backlog



Modelo de Negócio



## Concepção

Questionário de decisão



Arquitetura Alto Nível



User Stories



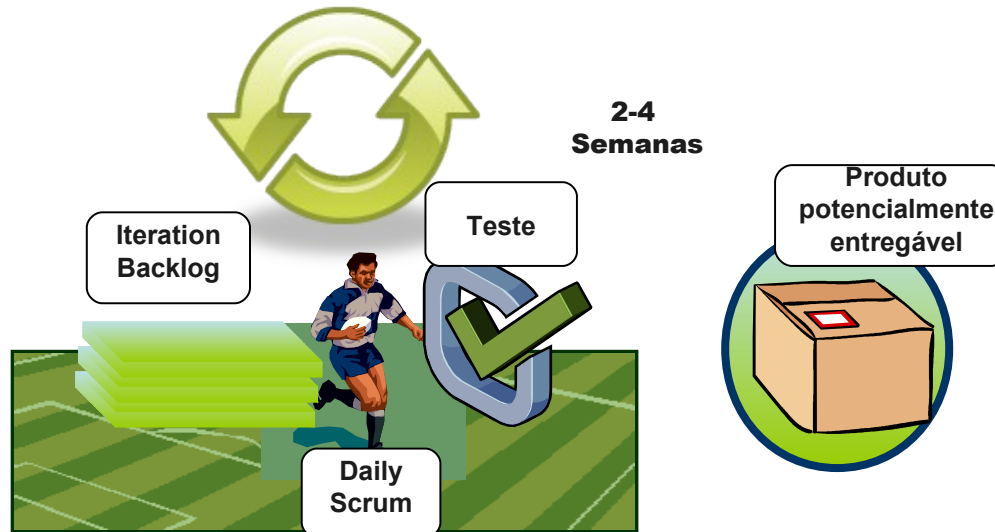
Confirmar Requisitos



Release Estimate



## Planejar, Desenvolver, Qualidade



# Agile Framework

## Extreme Programming (XP):

- Baseado nos valores de simplicidade, comunicação, feedback, coragem e respeito
- Comece com uma solução simples e adicione complexidade através de “refatoração” (*refactoring*)

## SCRUM:

- Equipes pequenas entre 6 e 8 pessoas
- “Backlog” define os requisitos que serão enderessados em cada Sprint
- Reunião diária de 15min. (Scrum) para discutir o trabalho do dia

## Unified Process:

- Versão simplificada do Rational Unified Process – redução no número de disciplinas

## Crystal:

- Entregas frequentes
- Melhorias reflectivas (Reflective improvement)

## Adaptive:

- Repetição dos ciclos de Especular, Colaborar e Aprender
- Aprendizagem contínua e adaptações para alterar o estado do projeto

## Dynamic Systems Development Method (DSDM):

- 3 fases primárias: Pré-Projeto, Ciclo de vida do Projeto, Pós-Projeto

## Feature Driven Dev.:

- Listar funcionalidades
- Planejamento, Design, Construção por Funcionalidade

***Técnicas ágeis: os métodos acima envolvem diferentes técnicas, entre elas:***

Test-driven development  
Planning game  
Pair Programming  
Refactoring

Continuous integration  
Design improvement  
Small releases  
Simple design

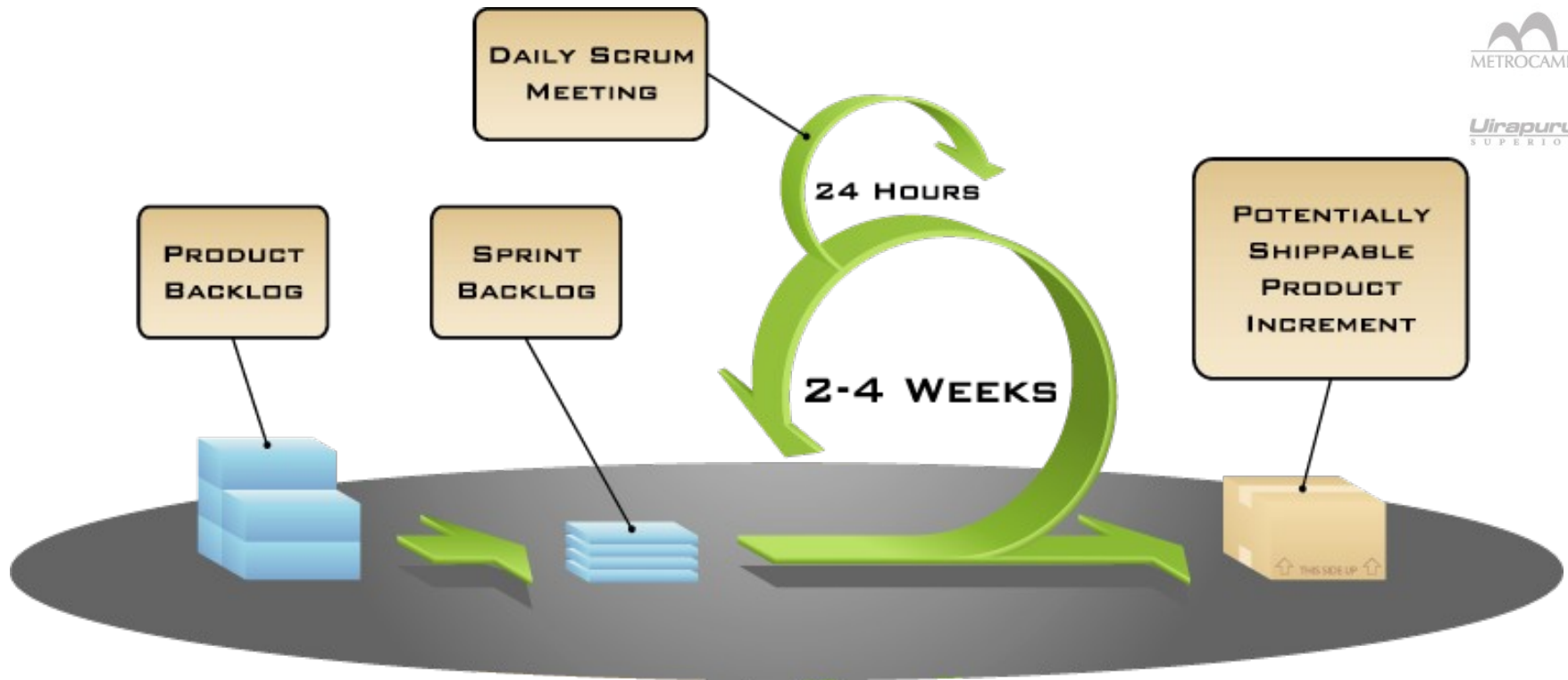
Static Analysis  
Coding standard  
Sustainable pace  
Whole team

# Scrum

- Foi criado também nos anos 80 e 90, primeiramente em círculos de desenvolvimento OO como uma metodologia altamente iterativa de desenvolvimento.  
Os colaboradores mais conhecidos são Ken Schwaber, Jeff Sutherland, e Mike Beedle
- Concentra mais os aspectos da gerência de desenvolvimento de software, dividindo o desenvolvimento em iterações de 2 a 4 semanas (chamadas "sprints") e aplicando uma monitoração mais próxima e um controle baseado em reuniões diárias
- Coloca muito **menos ênfase em práticas de engenharia** e muitos combinam suas **práticas de gerência de projeto** com as práticas de XP

# Scrum framework

Nota: se tornou a  
metodologia Ágil mais  
utilizada pelo mercado



COPYRIGHT © 2005, MOUNTAIN GOAT SOFTWARE

Sprint = Iteração

## Scrum Development Process

## Roles



Product Owner:  
Set priorities



ScrumMaster:  
Manage process,  
remove blocks



Team: Develop product



Stakeholders:  
observe & advise

## Key Artifacts

## Product Backlog

- List of requirements & issues
- Owned by Product Owner
- Anybody can add to it
- Only Product Owner prioritizes

## Sprint Goal

- One-sentence summary
- Declared by Product Owner
- Accepted by team

## Sprint Backlog

- List of tasks
- Owned by team
- Only team modifies it

## Blocks List

- List of blocks & unmade decisions
- Owned by ScrumMaster
- Updated daily

## Increment

- Version of the product
- Shippable functionality (tested, documented, etc.)

## Key Meetings

## Sprint Planning Meeting

Hosted by ScrumMaster

Attended by all

Input: Product backlog, existing product, business and technology conditions

1. Select highest priority items in Backlog; declare the sprint goals
2. Team turns selected items into Sprint Backlog

Output: Sprint Goal, Sprint Backlog

## Daily Scrum

Hosted by ScrumMaster

Attended by development team

Same time every day

**Answer**

1. What did you do yesterday?
2. What will you do today?
3. What's in your way?

### Team updates Sprint Backlog

Scrum Master updates Blocks List

## Sprint Review Meeting

Hosted by ScrumMaster

Attended by all

Informal, 4 hour, informational

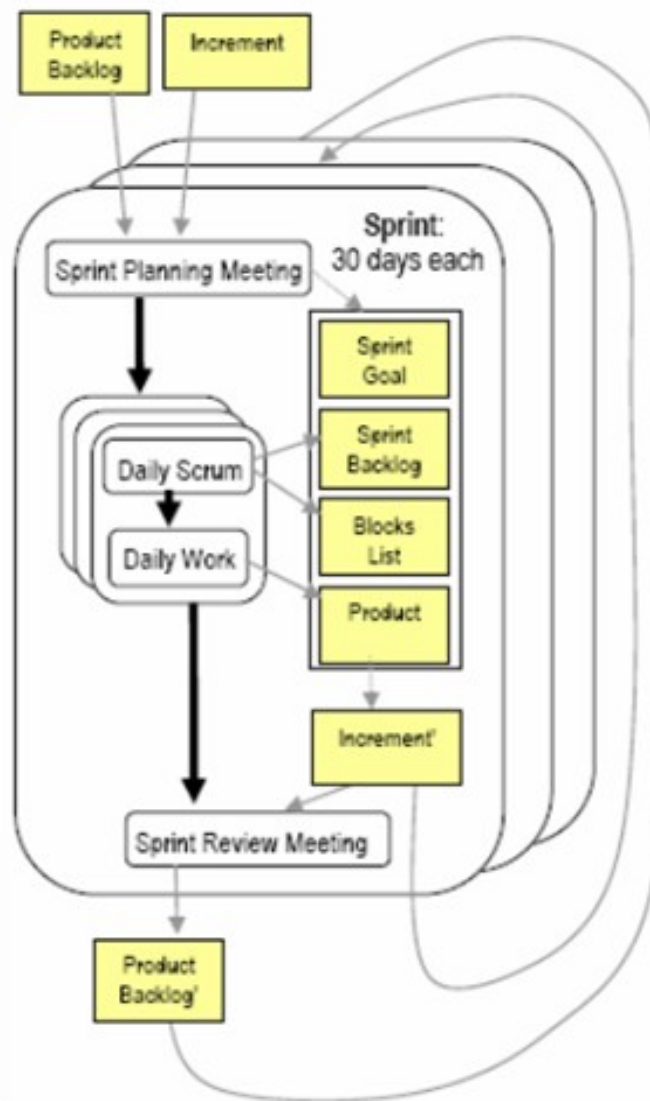
Team demos increment

All Discuss

Hold reflection

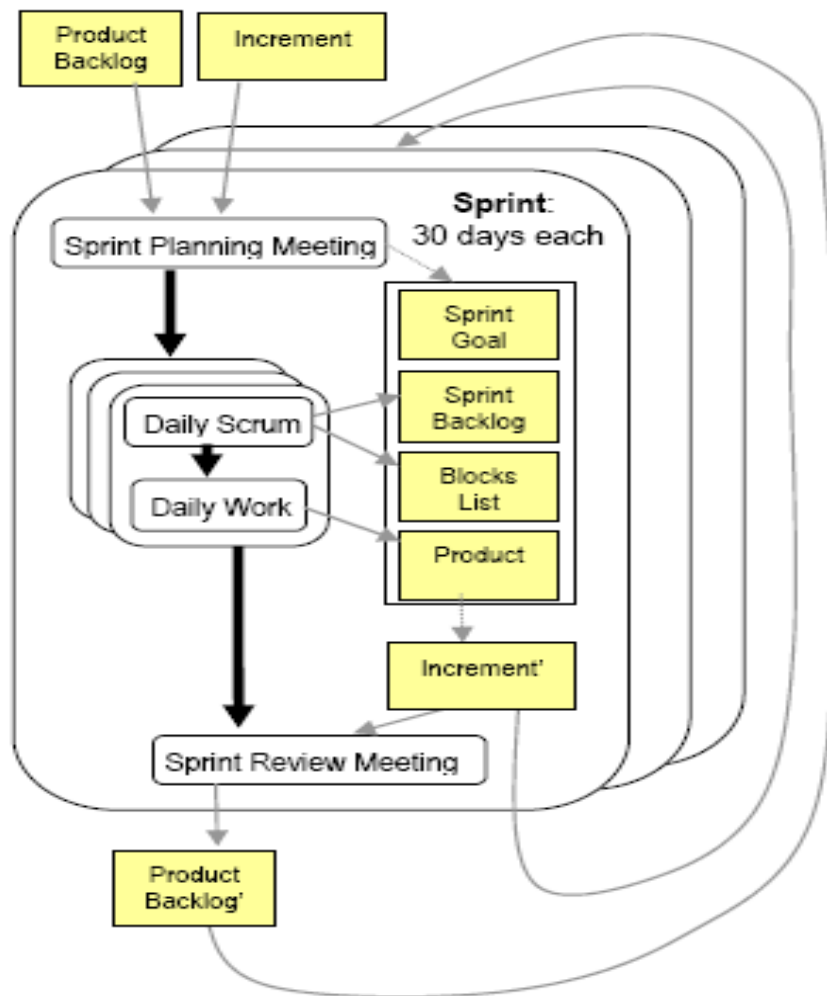
### Announce next Sprint Planning Meeting

## Development Process



# Scrum Development Process

## Development Process



## •Começa quando:

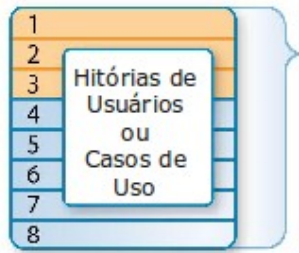
- A fase de Concepção está concluída
- As histórias de alta prioridade estão quebradas ao ponto de serem possíveis de implementar em uma iteração



Entradas dos Executivos,  
Time, Clientes, Usuários e  
outros Envolvidos



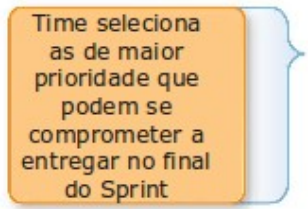
**Dono do Produto\*  
(Product Owner)**



**Backlog do Produto\*\***  
Lista priorizada dos requisistos



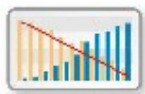
**Time\***



**Reunião de Planejamento do Sprint\*\*\***



**Scrum Master\***



**Gráfico Burndown\*\***



**Scrum\*\*\***  
Reunião diária

Cada 24 horas



**Sprint  
1-4  
Semanas**



**Backlog do Sprint\*\***

Data de Entrega e Backlog do Sprint não sofrem alterações após o início do Sprint



**Revisão do Sprint\*\*\***



**Trabalho Pronto\*\***



**Retrospectiva do Sprint\*\*\***

\* Papel, \*\* Artefato, \*\*\* Cerimônia

- Escolher algum dos modelos envolve analisar e avaliar:

- Estratégia de desenvolvimento
- Necessidades de entrega
- Recursos disponíveis
- Riscos do projeto/tecnologia
- Interação dos usuários/clientes e desenvolvedores
- Conhecimento dos desenvolvedores sobre negócio e tecnologia



# Referências



- Walker W. Royce. Managing the development of large software systems: concepts and techniques. Proc. IEEE WESTCON, Los Angeles, pages 1–9, August 1970.
  - <http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf>
- Barry W. Boehm. A spiral model of software development and enhancement. Computer, 21(5):61-72, May 1988.
  - <http://weblog.erenkrantz.com/~jerenk/phase-ii/Boe88.pdf>
- Philippe Kruchten. The Rational Unified Process: An Introduction. Addison-Wesley, Boston, MA, 3rd edition, 2003.
- Walker Royce. Software Project Management: A Unified Framework. Addison-Wesley, Boston, MA, 1st edition, 1998.
- Hirotaka Takeuchi and Ikujiro Nonaka. The New New Product Development Game. Harvard Business Review, 1986.
- Ken Schwaber. SCRUM Development Process. OOPSLA'95 Workshop on Business Object Design and Implementation, 1995.
- Ken Schwaber and Mike Beedle. Agile Software Development with Scrum. PrenticeHall, Upper Saddle River, New Jersey, 2002.

# Workshop sobre Ciclos de Vida de Projeto

Utilização prática dos modelos de processo de software apresentados

# Workshop sobre Ciclos de Vida de Projeto

## •Objetivos

- Consolidar os conhecimentos sobre modelos de processos de software alinhados às estratégias de projeto
- Propiciar discussão sobre melhoras práticas

## •Dinâmica

- Apresentação de cenário de projeto
- Discussão em grupos sobre alternativas
- Proposição de um ciclo de vida como estratégia
- Fechamento

## •Produtos Entregues

- Pontos Fracos e Pontos Fortes - SWOT
- Alternativa escolhida
- Justificativas sobre escolha

# SWOT ANALYSIS



# Cenário de Projeto



## •Fornecedor

- Fábrica de Software
- Fundada em 2000
- Localizada em Campinas
- Possui certificações ISO9001:2000 e CMMI Nível 2
- Aproximadamente 70 colaboradores
- Especializada em Java/JavaEE/OO
- Gestores realizam relacionamento comercial
- Baixa utilização de ferramentas

## •Cliente

- Multinacional americana
- Localizada em São Paulo
- Provedor de soluções em Telecomunicações
- Possui profissionais PMP

# Cenário de Projeto

## • Software

- Objetivo de negócio: Call Center de Serviços de Telefonia
- Análise de requisitos deve ser no ambiente do cliente, implementação deve ser na Fábrica
- Devem-se utilizar os frameworks disponibilizados pelo cliente
- Integração do software deve ser realizado no ambiente do cliente
- Tecnologias utilizadas: Java/JSP (Frontend) integrando com Mainframe (Server)
- Dados do sistema atual devem ser convertidos e migrados
- Negócio requer alto desempenho e confiabilidade
- Sistema deverá ser multi-línguas (internacionalização)
- Processo de desenvolvimento deverá seguir templates do cliente, quando existirem
- Urgência do negócio exige implantar em 8 meses
- São disponibilizadas 2 posições (baías) no ambiente do cliente
- Alocação prevista de 10 profissionais, Full-time
- Recebimentos de parcelas de faturamento do projeto ao fim de cada fase

# Encerrando nossa aula



- Nesta aula, tratamos sobre:
- Modelos de Processo de Software
  - Conhecendo os ciclos de vida aplicáveis a projetos de software
  - Identificando as estratégias inerentes à cada um destes ciclos e suas vantagens/desvantagens de uso
  - Entendendo como a escolha e implementação de um ciclo de vida de projeto ou a combinação destes pode se tornar uma importante estratégia de gestão de projetos de software
- Não existe processo que se encaixa em todos os contextos!
- O processo escolhido SEMPRE deverá ser adaptado!