

CONCEITOS DE SISTEMAS DE SOFTWARE

Johnatan Oliveira &
Eveline Alonso Veloso*

* Adaptado do material da Prof.^a Maria Augusta
Vieira Nelson

O que é um Programa de Computador?



- Conjunto de **instruções**
 - para serem **executadas por um computador.**
- O programa é o **código** que vai ser **executado pelo computador.**

O que é *Software*?



1. **instruções** (programas de computador) que, quando executadas, **produzem a função e o desempenho desejados**;
2. **estruturas de dados** que permitem a **manipulação das informações**;
3. **documentos** que descrevem a **operação e uso dos programas**.

O que é *Software*?



- É o **produto** que os engenheiros de *software* projetam e constroem.
- É a **parte lógica** de um sistema cuja **função** é **comandar o hardware**.

O que é Sistema?

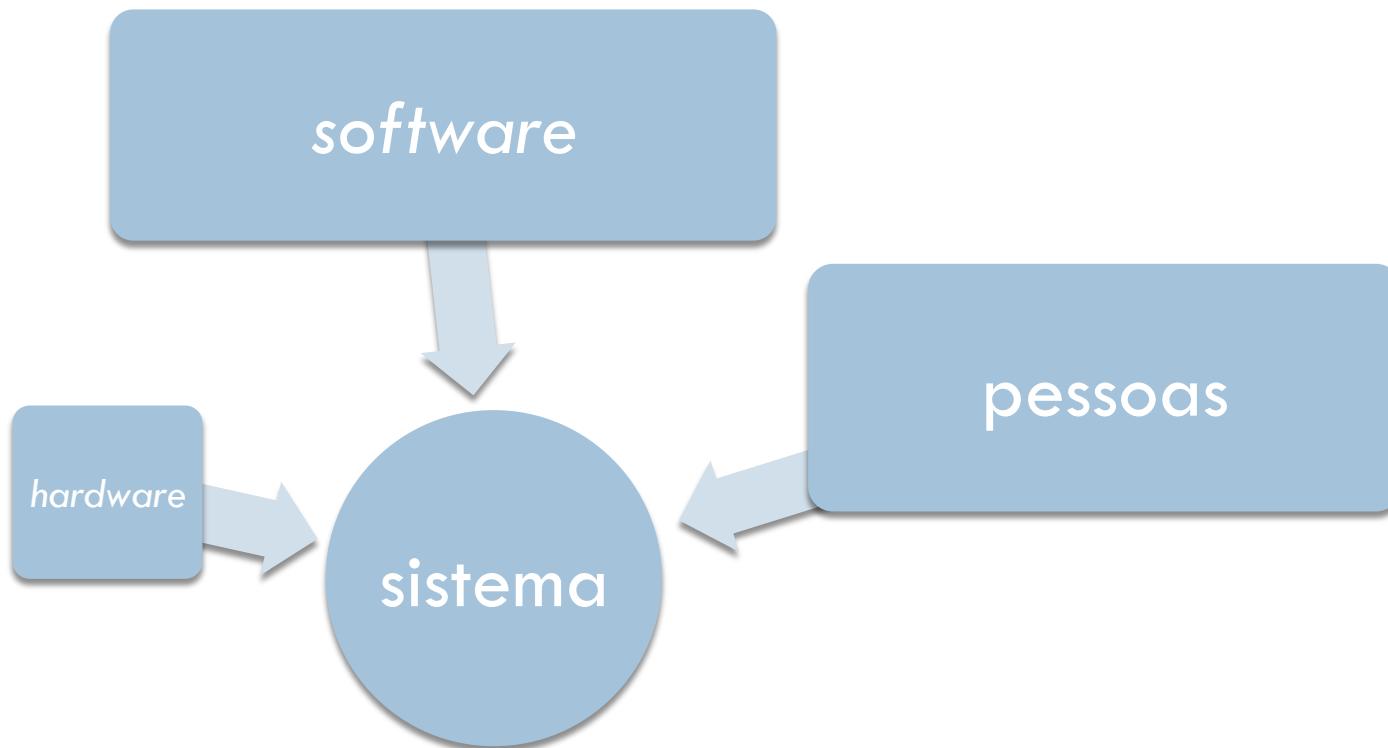


- É um **conjunto de elementos interdependentes;**
- de modo a **formar um todo organizado.**

Sistemas de *Software*

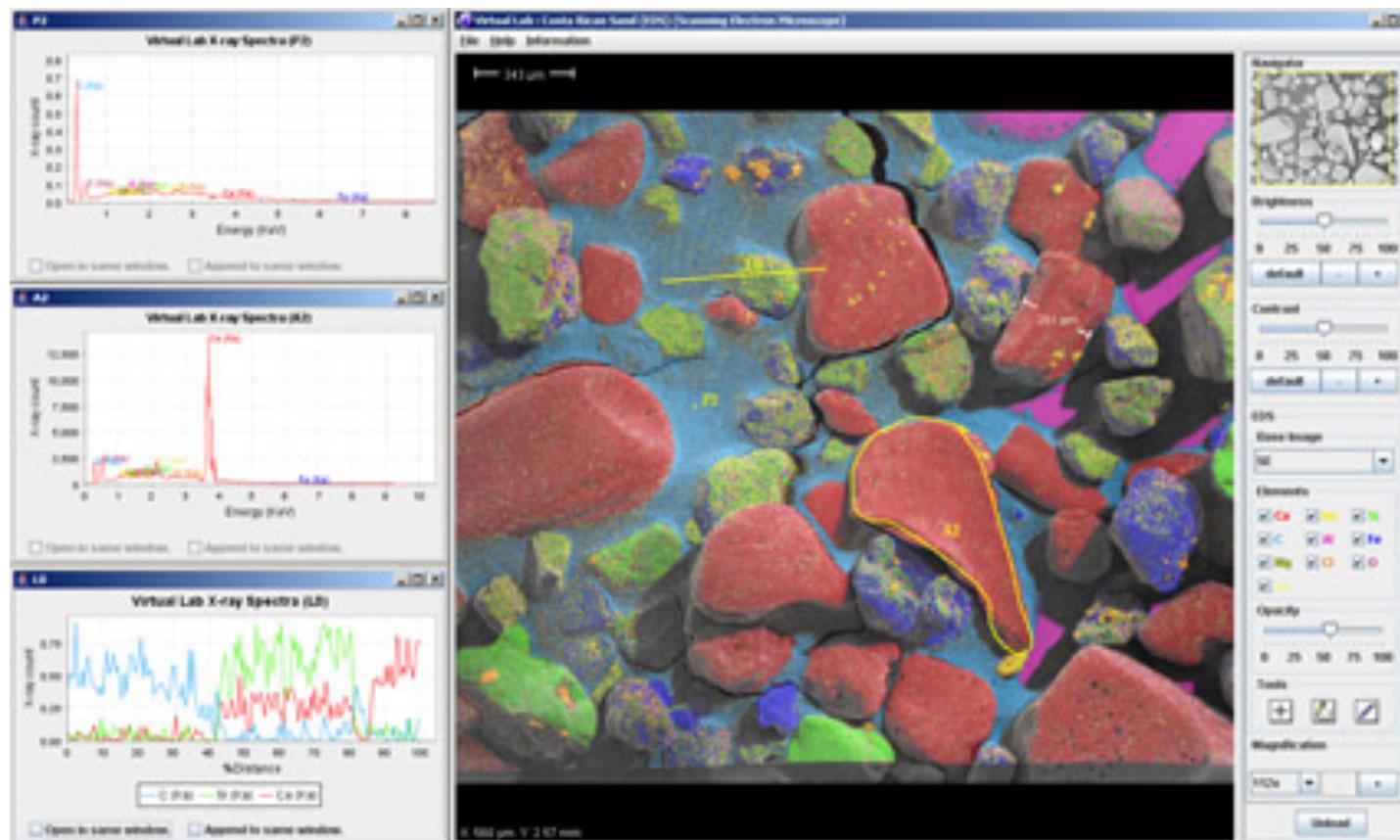


Sistemas Intensivos em *Software*



Exemplo

□ Microscópio virtual



É Difícil Desenvolver *Software*?

- É fácil desenvolver *software* que **aparentemente funciona**;
- mas que na prática **não funciona corretamente para todos os casos**.
- Engenheiros de *software* não agem de má fé;
 - mas agem de forma inocente quando acreditam que o *software* está funcionando.

Por quê Isso Ocorre?

Falsas Premissas

- Você se sente confiante de que **sabe o que o software deve fazer**;
- Seu **conhecimento do problema ou domínio é correto e completo**;
- Você consegue **traduzir precisamente o seu conhecimento em código**;
- O **código produzido faz exatamente o que você pensa que ele faz**;

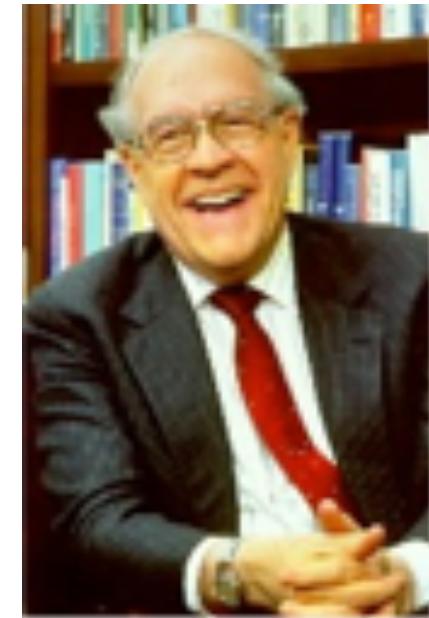
Por quê Isso Ocorre?

Falsas Premissas

- O **computador funciona da forma como esperamos** que ele deveria funcionar;
- Além do **software fazer o que você quer que ele faça**;
 - ele **não faz nada que você não quer** que ele faça;
- O **software desempenha todo seu papel conforme o esperado** para seu uso pretendido;
- O **software se comporta razoavelmente bem** quando encontra **entradas maliciosas ou absurdas**.

No Silver Bullet

"A parte mais difícil da construção de um sistema de *software* é decidir precisamente o que construir. Nenhuma outra parte do trabalho conceitual é tão difícil quanto estabelecer os requisitos técnicos detalhados, incluindo todas as interfaces com pessoas, máquinas e outros sistemas de *software*. Nenhuma outra parte do trabalho prejudica tanto o sistema resultante se for feita incorretamente. Nenhuma outra parte é tão difícil de se corrigir posteriormente."



"No Silver Bullet: Essence and Accidents of Software Engineering"
Fred Brooks

Por quê *Software* é Importante?



- **Erros introduzidos nos softwares podem causar:**
 - **perda de vidas;**
 - **prejuízos financeiros;**
 - **atrasos nas entregas;**
 - **aumento de riscos;**
 - **baixa qualidade.**

Prejuízos Financeiros Atribuídos a *Software*

□ Ariane 5:

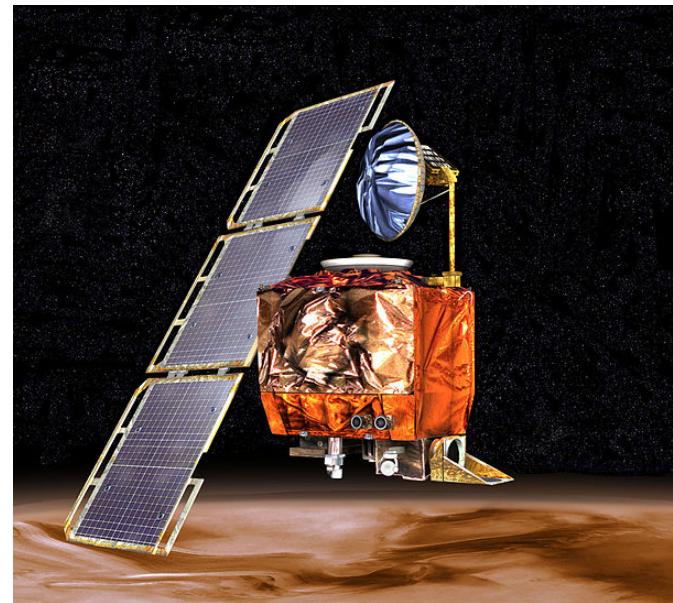
- custo:
 - \$7 bilhões de dólares;
- erro:
 - **overflow** durante conversão de ponto flutuante para inteiro;
- curiosidade:
 - código que causou o acidente era desnecessário após a decolagem!



Foguete Ariane V
4 de junho de 1996

Prejuízos Financeiros Atribuídos a *Software*

- Mars Climate Orbiter:
 - custo:
 - \$125 milhões de dólares;
 - erro:
 - **utilização de diferentes unidades de medida** por parte de componentes do *software*:
 - um deles usou o **sistema métrico** e o **outro**, o **sistema inglês**;
 - curiosidade:
 - *software* estava “correto”;
 - especificação (descrição do que deveria ser feito) é que estava incorreta.



Orbitador Climático de
Marte
23 de setembro de 1999

Perda de Vidas Humanas Atribuídas a *Software*

□ Therac-25:

- 6 acidentes envolvendo doses altíssimas de radiação;
 - pelo menos 4 mortes;
 - mais 2 pessoas gravemente contaminadas;
- diversos problemas:
 - **sistema de controle não operava sincronizado com a interface** do equipamento;
 - projeto **não continha travas de segurança** de *hardware*;
 - **documentação** fornecida aos usuários era **incompleta**;
- curiosidade:
 - falta de sincronização entre o sistema de controle e a interface já ocorria em modelos anteriores.



Therac-25
1985 - 1987

Perda de Vidas Humanas Atribuídas a *Software*

- **London Ambulance System:**
 - ambulâncias levaram horas para chegar aos locais das chamadas de emergência;
 - estima-se cerca de **11** mortes;
 - diversos problemas, inclusive relacionados à especificação do que deveria ser construído:
 - **premissas inadequadas;**
 - **falta de envolvimento de futuros usuários do sistema** em seu processo de especificação;
 - **falta de observação da estrutura organizacional do serviço de ambulâncias** de Londres;
 - curiosidade:
 - em 2006, o lançamento de uma nova versão causou nova falha do sistema.



Sistema de Ambulâncias
de Londres
1992

Classificação dos *Softwares*



- ***Software de segurança crítica:***
 - **falhas** podem ter **sérias consequências:**
 - perda de vidas humanas;
 - danos ao meio ambiente;
 - danos materiais expressivos;
 - danos à reputação da empresa que o utiliza.

Classificação dos *Softwares*



- ***Software de missão crítica:***
 - **indispensável** para que os **serviços, aplicações e processos** de uma organização **continuem operando**.
 - **Falhas:**
 - **paralisação do negócio;**
 - **perda de dados importantes.**

Classificação dos *Softwares*

- ***Software de custo crítico:***
- **falhas:**
 - **custos elevados** para a empresa que o utiliza.



Software na Sociedade

- 
- *Software* foi responsável por grandes **mudanças na sociedade**:
 - **sociedade industrial** para;
 - **sociedade da informação**.
 - Material tem menos valor;
 - informação tem mais valor.

Sociedade da Informação



- **Informação:**
 - **meio para criação de conhecimento.**
 - Desempenha papel fundamental na:
 - **produção de riqueza;**
 - **melhoria do bem-estar e qualidade de vida dos cidadãos.**

Sociedade da Informação



- **Condição para a Sociedade da Informação avançar:**
 - possibilidade de **todos acessarem e utilizarem as Tecnologias de Informação e Comunicação (TICs);**
 - instrumentos indispensáveis às comunicações pessoais, de trabalho, de construção do conhecimento e de lazer.

Referências

Alguns *sites*

- **Ariane 5**

- <https://youtu.be/5tJPXYA0Nec>
- <https://www-users.cse.umn.edu/~arnold/disasters/a-bug-and-a-crash.pdf>

- **No Silver Bullet**

- <http://worrydream.com/refs/Brooks-NoSilverBullet.pdf>

ATIVIDADES DA ENGENHARIA DE SOFTWARE

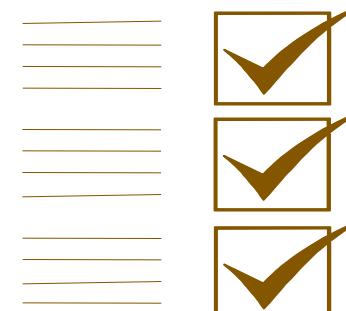
Johnatan Oliveira &
Eveline Alonso Veloso*

* Adaptado do material da Prof.^a Maria Augusta
Vieira Nelson

Produção de *Software*



- **Segue um conjunto ordenado de atividades.**
- **Quais** são essas atividades?



Atividades da Engenharia de *Software*



- **Técnicas:**

- Modelagem de Processos de Negócios, Engenharia de Requisitos, Projeto de *Software*, Implementação, Testes, Manutenção e Evolução, Medição, Estimativas.

- **Gerenciais:**

- Aceite pelo Cliente, Gerência de Configuração, Gerência de Projetos, Gerência de Requisitos, Gestão da Qualidade, Gestão de Processos.

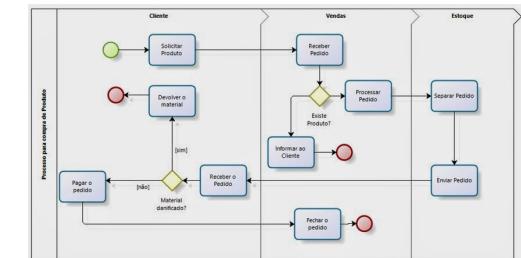
- **Apoio:**

- Comunicação, Gestão de Pessoas, Gestão de Equipes.

Modelagem de Processos de Negócios

□ Objetivos:

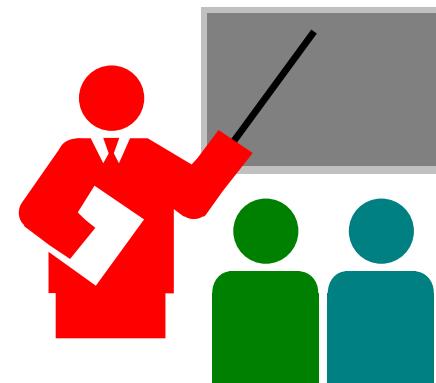
- entender **como funciona** a organização do cliente;
- modelar o domínio do negócio;
- encontrar **oportunidades** de informatização.



Levantamento de Requisitos



- Os **problemas** têm que ser **enunciados...**
- ... antes de serem **resolvidos.**



Levantamento de Requisitos



- **Objetivos:**

- **definir o problema e as necessidades do cliente;**
- **identificar as características do domínio do cliente;**
- **obter um conjunto de requisitos do produto;**
 - acordado entre cliente e fornecedor.

Levantamento de Requisitos

□ Técnicas empregadas:

- entrevistas;
- questionários;
- leitura de documentos;
- observação;
- pesquisas de levantamento de dados;
- prototipação.

PAFEI

Análise de Requisitos



- **Objetivo:**
 - ▣ **detalhar os requisitos** levantados;
 - procurando **identificar inconsistências, conflitos, lacunas em suas descrições.**
 - **Modelagem** dos requisitos levantados ajuda a **descobrir falhas** no entendimento, descrições incompletas;
 - ▣ por outro lado, pode levar a um **engessamento precoce da percepção do domínio.**

Especificação de Requisitos



- **Objetivo:**
 - ▣ **descrever** os **requisitos** identificados **de maneira clara e precisa, sem ambiguidades.**
- **Técnicas empregadas:**
 - ▣ linguagem natural;
 - ▣ linguagem controlada em um *template*:
 - casos de uso;
 - ▣ notações específicas para diagramas de modelagem de casos de uso, classes, tabelas de condições.

Validação de Requisitos



- **Objetivo:**
 - **revisão**, por parte do **cliente e usuários**, para **determinar** se a **descrição dos requisitos** está **correta, clara, precisa e completa**.
- **Protótipos** podem ser empregados nesta atividade.
- Notações mais sofisticadas dificultam a realização desta atividade por parte do cliente.

Engenharia de Requisitos



Fundamentos de Engenharia de Software

Projeto (*Design*) de *Software*



□ **Objetivos:**

- **analisar** diversas **alternativas de solução**, ponderando sobre uma ou outra alternativa;
 - todas as alternativas **precisam satisfazer os requisitos levantados**;
- **escolher uma solução**;
- **detalhar a solução escolhida** por meio de notações específicas para o projeto de *software*.

Projeto (*Design*) de *Software*



- Existem **duas etapas** no projeto de *software*:
 - ▣ **Projeto da Arquitetura do *Software*:**
 - **visão mais abstrata;**
 - quais são as **grandes partes do *software*, como elas se comunicam, onde elas executam?**
 - necessário para **satisfazer requisitos de qualidade.**

Projeto (*Design*) de *Software*



- Existem **duas etapas** no projeto de *software*:
 - ▣ **Projeto Detalhado dos Objetos** ou elementos de pequena granularidade:
 - quais são os **objetos, como** eles **interagem entre si, quem é responsável por cada função?**
 - necessário para **satisfazer os requisitos funcionais**;
 - preocupa-se com a **escolha** adequada de **algoritmos e estruturas de dados** para **resolver o problema de forma eficiente**.

Implementação ou Codificação

- **Objetivo:**

- **transformar modelos** de projeto do *software*;
 - **em código** escrito em uma linguagem de programação:
 - construir o sistema executável.
- Preocupa-se com a **implementação** adequada **dos algoritmos e estruturas de dados escolhidos** durante o projeto do *software*.

Testes de *Software*



- **Objetivo:**

- **verificar** que o **software funciona como o esperado** e **satisfaz os requisitos** dos clientes;
 - encontrando defeitos e não-conformidades no sistema.

- **Testes:**

- manuais;
 - automatizados;
 - que concentram-se mais no código;
 - ou no usuário;
 - ou na especificação dos requisitos.

Aceite pelo Cliente



- **Objetivo:**
 - **validar se o *software* entregue satisfaz as necessidades do cliente;**
 - tanto em termos de **funcionalidade**;
 - quanto de **qualidade**.
 - **Atividade realizada pelo cliente.**

Desenvolvimento de *Software*

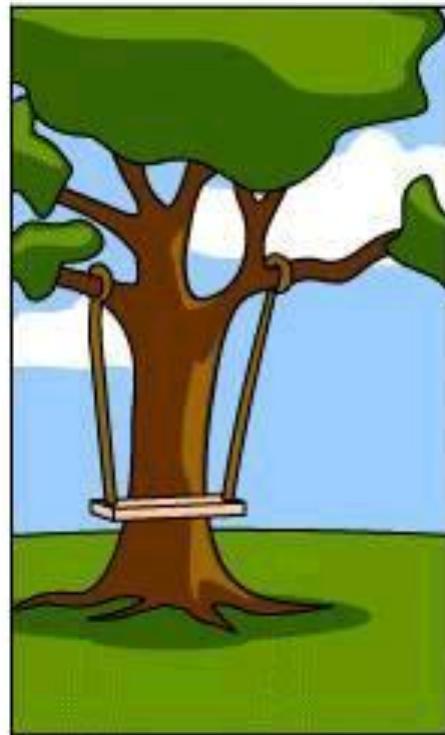
- Como o cliente explicou...



Desenvolvimento de *Software*



- Como os analistas de requisitos entenderam



Desenvolvimento de *Software*



- O que os arquitetos de *software* projetaram



Desenvolvimento de *Software*



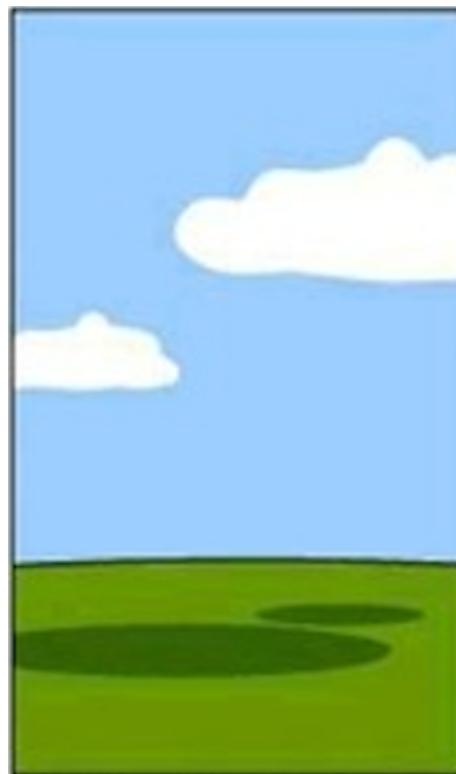
- O que foi construído



Desenvolvimento de *Software*



- Como o projeto foi documentado



Desenvolvimento de *Software*

- Quais funcionalidades foram entregues



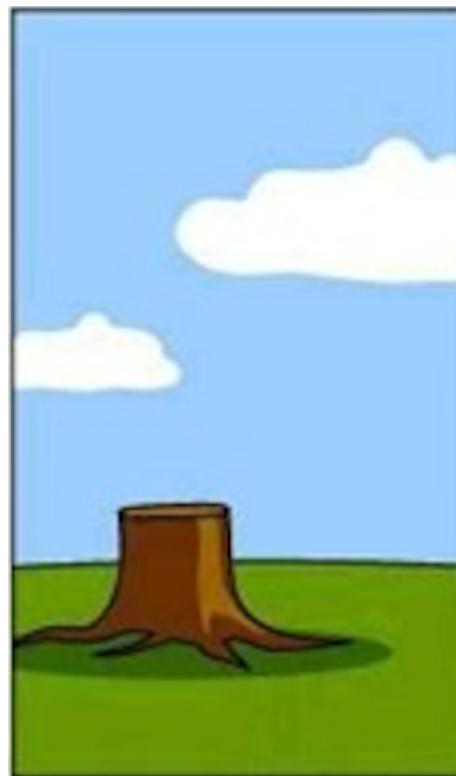
Desenvolvimento de *Software*

- Como o cliente foi cobrado



Desenvolvimento de *Software*

- Como o produto foi mantido



Desenvolvimento de *Software*

- O que o usuário realmente queria...



Manutenção e Evolução de *Software*

- Atividade **contínua** após a entrega do **software** ao cliente.



Manutenção e Evolução de *Software*



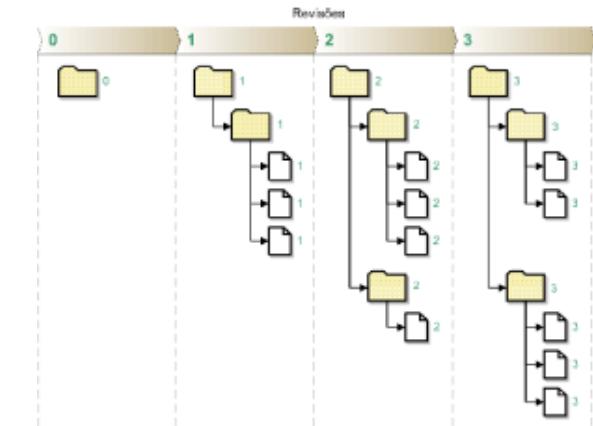
□ **Objetivos:**

- **corrigir defeitos** que não foram descobertos antes da implantação do produto;
- implementar **melhorias** no produto;
- desenvolver **novos requisitos**;
- atender **novas necessidades** dos clientes;
- **mudar** as **tecnologias** empregadas.

Gerência de Configuração

- **Objetivo:**

- **gerenciar**, todos os **artefatos** produzidos nos projetos, **por meio de ferramentas**;
 - que **não violem** as **regras** estabelecidas na organização **para o controle de mudanças**.
- Realizada durante todo o processo de **desenvolvimento, manutenção e evolução** do *software*.



Gerência de Projetos

- **Objetivos:**

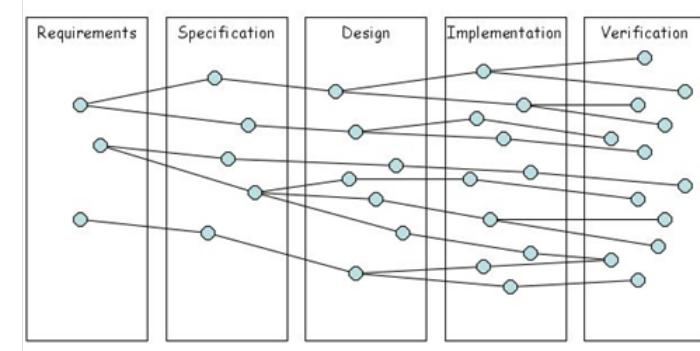
- **planejar, acompanhar e controlar a execução de um projeto;**
 - com duração no **tempo**;
 - envolvendo **pessoas alocadas** em sua execução;
 - e um **custo** de produção;
- **gerenciar riscos.**



Gerência de Requisitos

□ Objetivos:

- **priorizar** os **requisitos** levantados;
- **aprovar mudanças** de requisitos;
- **controlar** o **escopo** do projeto de *software*;
- **manter** a **rastreabilidade** dos requisitos e artefatos produzidos.



Gestão da Qualidade

- **Objetivo:**

- **avaliar o processo, as práticas, a comunicação, as políticas;**
 - enfim, **tudo o que pode impactar na qualidade de um *software*;**
 - e não apenas a sua correta execução.



Gestão de Processos

- **Objetivo:**
 - **definir processos e propor melhorias;**
 - com base nas **boas práticas da Engenharia de Software;**
 - e nos **modelos de capacitação e maturidade** como CMMi, MPS.Br.



Medição de *Software*



- **Objetivos:**

- **definir metas;**
- **definir métricas;**
 - como unidade de medida das metas;
- **coletar dados ou realizar medições;**
 - a respeito do **processo**, do **projeto** ou do **produto** de *software*.



Estimativas de *Software*

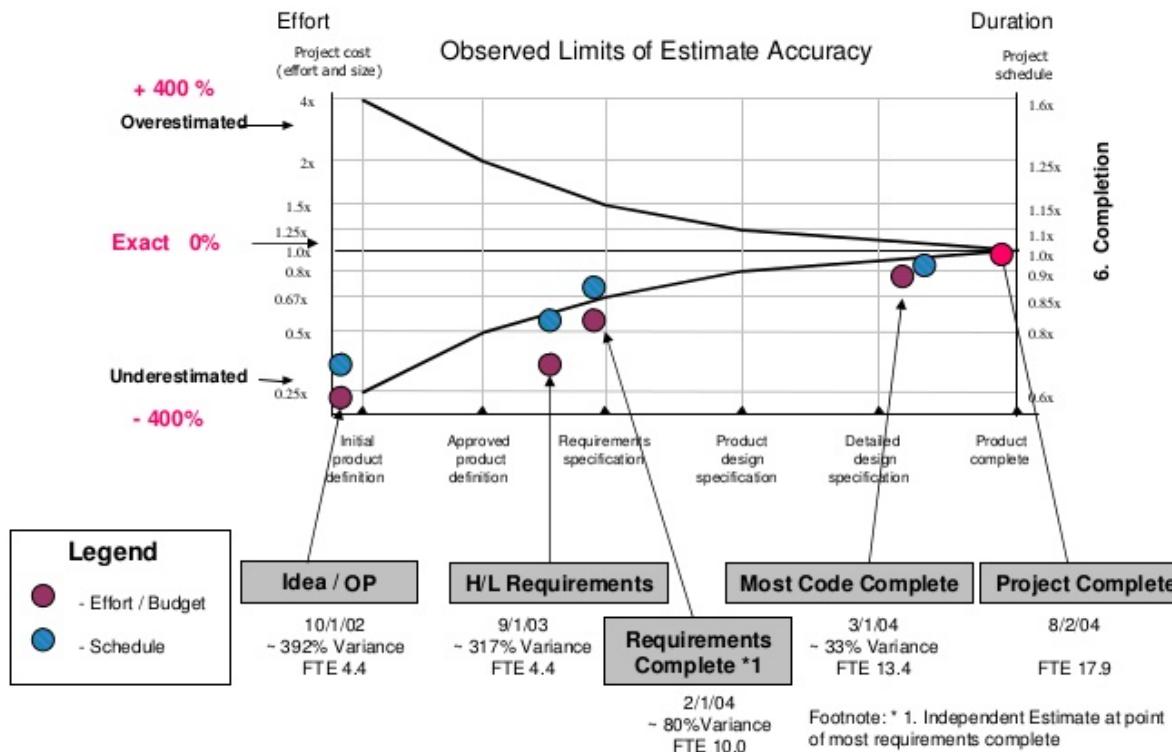


- **Objetivo:**
 - **prever o desenvolvimento ou manutenção do *software*;**
 - com base em **dados históricos** do processo de produção.
 - Pode-se **estimar** o **tempo**, o **custo**, o **esforço**, o **tamanho do *software***, o **tamanho da equipe** necessária e outros.

Estimativas de Software



Historical Data: Example of Estimating Maturity



Referências



- PRESSMAN, Roger. **Engenharia de *software*: uma abordagem profissional.** Capítulo 2.
- SOMMERVILLE, Ian. **Engenharia de *software*.** Capítulo 2.
- CMMi
 - ▣ <http://cmmiinstitute.com>
- MPS.Br
 - ▣ <http://www.softex.br/mpsbr>

TIPOS DE REQUISITOS DE SOFTWARE

Eveline Alonso Veloso*

* Adaptado do material da Prof.^a Maria Augusta
Vieira Nelson

O que é um Requisito?



- **Característica observável** do sistema;
- **Efeito** que um **sistema** deve **exercer**;
 - **sobre o mundo real**;
- Uma **condição** ou **capacidade necessitada** por um **usuário**;
 - para **resolver** um **problema** ou **atingir** um **objetivo**.

O que é um Requisito?



- Uma **condição** ou **capacidade** que deve ser cumprida ou possuída por um sistema ou componente do sistema;
- para **satisfazer** um **contrato, padrão, especificação** ou outro **documento formal** imposto.

IEEE Standard Glossary of Software Engineering Terminology

O que é um Requisito?



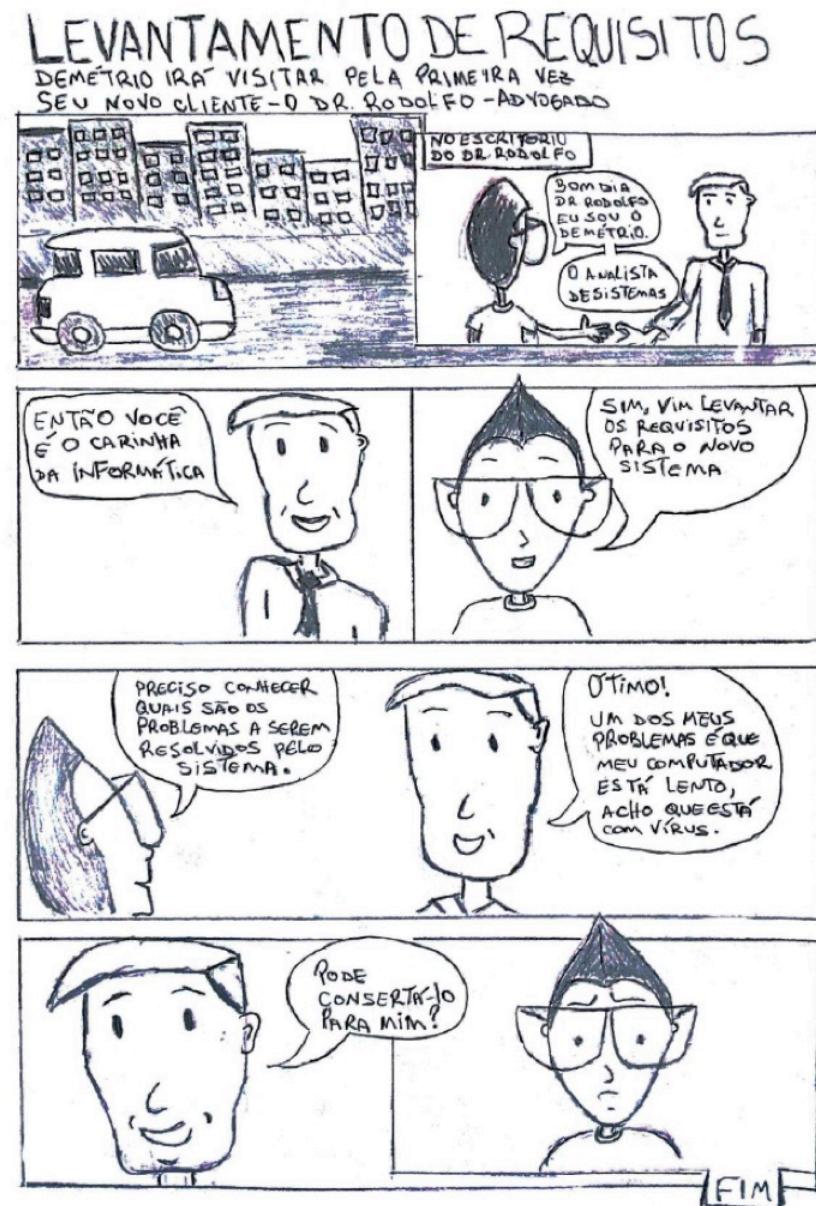
- O requisito se refere a um **problema** que existe no **mundo real**;
 - não na máquina.
 - O **software** é uma **solução para algum problema**:
 - se você não sabe o quê fazer;
 - não adianta definir como fazer;
 - ainda será muito prematuro para tomar decisões.
- Envolve **aspectos do domínio**;
 - na **fronteira do software**.
- Características que definem os **critérios de aceitação** de um **produto**.

Levantamento de Requisitos



- Deve ser **feito junto aos stakeholders;**
 - **interessados** no projeto.
- Possíveis *stakeholders*:
 - **usuários;**
 - **clientes;**
 - **especialistas no domínio;**
 - investidores ou **patrocinadores.**

Figura 1 - Exemplo de um estudo de caso explicando os conceitos de Levantamento de Requisitos.



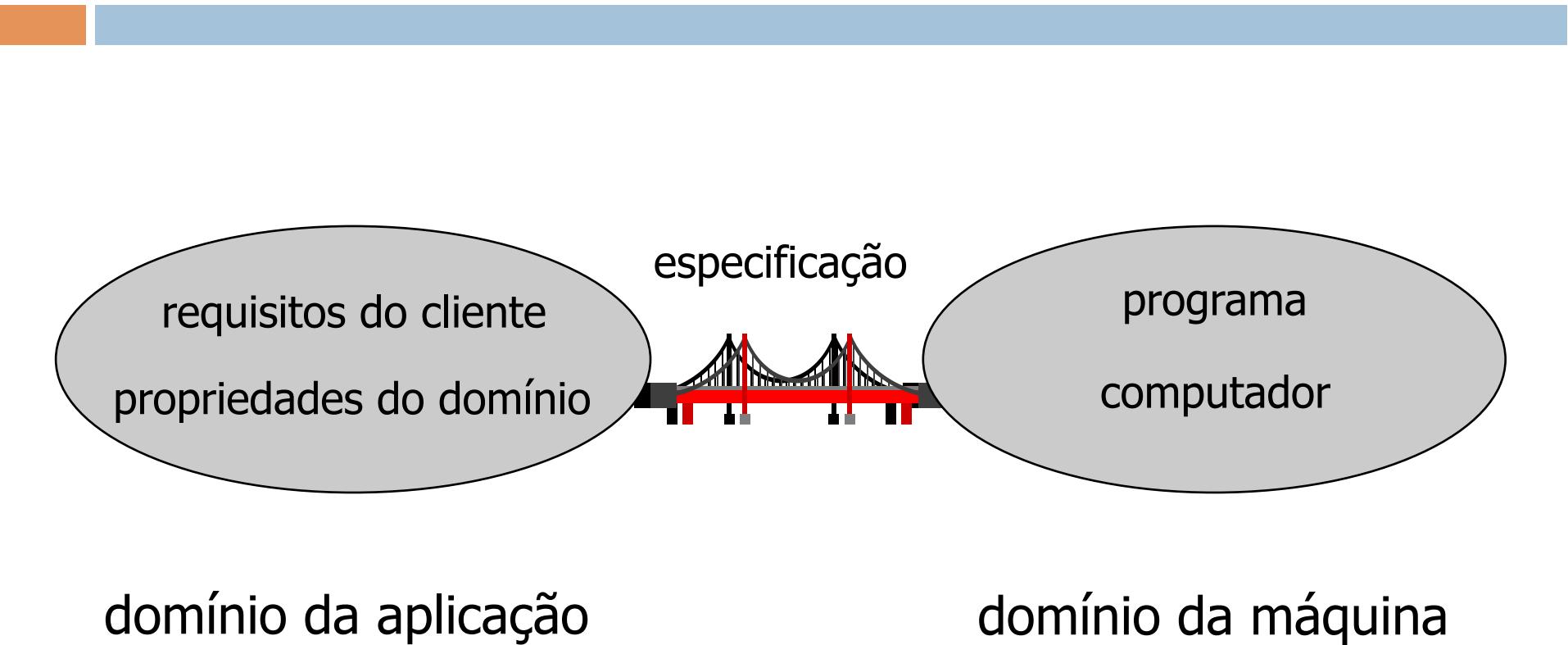
Fonte: Elaborado por Jair Jersey Marinho e Jefferson Marinho.

www.facebook.com/analistademetrio.com.br

Levantamento de Requisitos

Fundamentos de Engenharia de Software

Levantamento de Requisitos



O que são Requisitos do Cliente?



- **Descrevem o problema** enfrentado pelo **cliente ou usuário**;
 - materializado **a partir de suas necessidades**.
- **Expressam os resultados desejados** para **superar os problemas reais**.
- Requisitos **não falam sobre o sistema**;
 - mas sobre *os efeitos do sistema* sobre o **mundo real**.

Exemplos de Requisitos do Cliente



- Eu, como secretaria escolar, preciso que o histórico escolar contenha todas as notas de apenas um aluno.
- Eu, como operador do reator nuclear, gostaria de ser notificado sobre as mudanças de temperatura do reator para tomar decisões.
- Eu, como monitor de satélites, preciso calcular a órbita e trajetória dos satélites para estimar a posição dos satélites em um determinado momento no tempo.

O que são Requisitos de *Software*?



- Descrevem o **comportamento** de um **sistema** de computação;
 - apresentado como **solução** para o **problema do cliente**.
- **Delimitam** as **interfaces** de um **sistema**;
 - que **soluciona o problema**.
- Também chamados de **requisitos do produto**.

Exemplos de Requisitos de *Software*

- O sistema deve notificar os operadores, com avisos luminosos e sonoros, sobre mudanças de temperatura nas partes centrais do reator e permitir ao operador registrar a sua tomada de decisão diante da temperatura notificada.
- O sistema deve calcular e mostrar a órbita e trajetória de todos os satélites.
- O sistema deve apresentar a posição dos satélites em um determinado momento no tempo fornecido como entrada pelo usuário.

O que são Requisitos Funcionais?

- Descrevem **o que o software deve fazer**.
- Definem detalhadamente:
 - as **funcionalidades, serviços e comportamento esperados do sistema**;
 - mediante cada entrada;
 - ou seja, **como o sistema deve reagir** a entradas específicas.
- São aquilo que descreve **o que o sistema tem que fazer**;
 - a cada ação de um usuário ou outro sistema.

Exemplos de Requisitos Funcionais

- Quando o cliente pesquisar por um restaurante, devem ser exibidos no mapa os restaurantes mais próximos.
- O sistema deve permitir ao usuário incluir um perfil de cliente.
- O sistema deve gerar a nota fiscal, dar baixa no estoque e enviar o boleto automaticamente.
- O sistema deve emitir um relatório de vendas por cliente indicando: a data da compra, o que foi comprado e a rentabilidade do cliente para empresa.

Exemplos de Requisitos Funcionais

- Em um sistema de transporte individual, o *software* deve permitir que o usuário veja o motorista mais próximo e possa selecioná-lo para a corrida.
- O sistema deve monitorar e capturar entradas do sensor meteorológico.
- O *software* precisa emitir um gráfico com as oscilações de temperatura do dia.

Exemplos de Requisitos Funcionais

- Em um sistema de agenda eletrônica acadêmica, devem ser armazenadas as datas de entrega de tarefas dos alunos.
- O sistema deve permitir aos usuários comprar e vender títulos.
- O sistema deve permitir ao gerente do banco visualizar todos os empréstimos efetuados no mês, indicando o funcionário que disponibilizou o empréstimo, o cliente que obteve o empréstimo e o valor emprestado.

O que são Requisitos Não-Funcionais?



- **Quantificam** determinados **aspectos do comportamento do sistema**.
- Definem **o que se espera do sistema**;
 - em termos de **qualidade**;
 - **normas e padrões** que devem conduzir o seu funcionamento e construção.
- Descrevem **restrições desejadas ou necessárias, atributos do sistema** ou de seu **ambiente**.

Figura 3 - Exemplo de um estudo de caso explicando os conceitos de Requisitos Não Funcionais.



Fonte: Elaborado por Jair Jersey Marinho e Jefferson Marinho
www.facebook.com/analistademetrio.com.br

O que são Requisitos Não-Funcionais?

Exemplos de Requisitos Não-Funcionais



- Toda consulta ao acervo da biblioteca deve ser respondida em até 5 segundos.
- O *software* deve apresentar as opções de restaurantes mais próximos em, no máximo, 2 segundos.
- O sistema deve ser dimensionado para suportar, pelo menos, 1.000 usuários simultaneamente.

Exemplos de Requisitos Não-Funcionais



- O sistema deve apresentar uma disponibilidade de, pelo menos, 99%.
- O sistema deve ser acessível aos usuários com necessidades especiais de visão, permitindo a utilização de um leitor de tela.
- O sistema deve ser seguro identificando seus usuários por meio de autenticação com identificador único.

Exemplos de Requisitos Não-Funcionais

- O sistema precisa de uma infraestrutura composta por máquinas com a configuração xyz, ou superior, e provedor de Internet com taxa de transmissão yyy, ou superior.
- O sistema deve ser desenvolvido para funcionar a partir de determinada versão do sistema operacional wyz.
- O sistema deve ser suportado a partir de determinada versão de um sistema operacional *mobile* específico.

Critérios de Aceitação de Requisitos Não-Funcionais

- Requisitos não-funcionais devem descrever um critério de aceitação **mensurável** ou **quantificável**.
 - Deve-se **associar** uma **métrica** a cada **requisito não-funcional** elicitado;
 - que pode ser **objetivamente medida**.
- Esse critério de aceitação é uma forma de **verificar ou testar** se o **requisito foi satisfeito ou não**, fazendo alguma **medição**.
- Exemplo: o sistema deve executar 90% de suas transações em até 1 segundo.

Tipos de Requisitos Não-Funcionais



- **Desempenho:**
 - O sistema deve processar os dados recebidos dos sensores em, no máximo, 1 segundo.
- **Disponibilidade:**
 - O sistema deve recuperar-se de uma falha em seu banco de dados em, no máximo, 30 segundos.
- **Portabilidade:**
 - O sistema deve permitir que a interface com o usuário possa ser migrada para uma plataforma nova em 6 meses.

Tipos de Requisitos Não-Funcionais



□ **Usabilidade:**

- ▣ O sistema deve permitir ao usuário completar uma tarefa de cadastro de cliente em até 1 minuto.

□ **Degradação:**

- ▣ O sistema deve ser capaz de atender 100 requisições por segundo e ignorar requisições acima da 100^a para que não aconteça degradação em seu tempo de resposta.

Tipos de Requisitos Não-Funcionais



- **Capacidade:**

- O sistema deve utilizar apenas 50% da capacidade de seu processador.

- **Manutenibilidade ou Adaptabilidade:**

- O sistema deve ser passível de manutenção para migração para outra plataforma em até 3 semanas.

Restrições de Processo



- Restrições relativas à **forma como o desenvolvimento do sistema será conduzido**, como:
 - **recursos disponíveis:**
 - A equipe desenvolvedora deve ser composta por programadores sênior com experiência comprovada;
 - **documentação exigida:**
 - Deve existir uma documentação do sistema no formato de *help online* para o usuário;

Restrições de Processo



- **normas** a serem seguidas:
 - A norma ISO 9126 deve ser usada para avaliar a qualidade do sistema;
- **processo de desenvolvimento de *software*** a ser seguido:
 - O sistema deve ser desenvolvido seguindo o processo RUP e todos os artefatos previstos no processo devem ser produzidos.

Restrições de Projeto



- Restrições relativas a **fatores externos ao sistema**, como:
 - **usuários do sistema:**
 - Devem ser capazes de utilizar o sistema, usuários sem conhecimento prévio em informática, após treinamento;
 - **interfaces com outros sistemas:**
 - O sistema deve integrar-se com sistemas legados da organização;

Restrições de Projeto



- **ambiente de desenvolvimento ou de operação:**
 - O banco de dados MySQL deve ser usado para persistência de dados;
 - O sistema deve ser implementado em C#;
 - O sistema deve executar nas plataformas *Windows* e *Unix*;
 - O sistema deve estar acessível na *Web*.

Requisitos no SCRUM



- Descritos no ***backlog*** do produto.
- **Requisitos funcionais:**
 - escritos em formato de **histórias de usuário**.
- **Requisitos não-funcionais:**
 - escritos na “**Definição de Pronto**”.
 - A “Definição de Pronto” é um **acordo formal** entre o *Product Owner* e o Time de Desenvolvimento sobre **o que é necessário** para se considerar que o **trabalho realizado na *sprint* está “pronto”**.

USER STORY



Requisitos no SCRUM

Requisitos no RUP



- Descritos em um documento chamado:
 - **Especificação de Requisitos de Software (ERS).**
- **Requisitos funcionais:**
 - descritos por meio de **casos de uso** na ERS.
- **Requisitos não-funcionais:**
 - descritos em uma **seção de requisitos não-funcionais** na ERS.

Requisitos no RUP

Realizar Inscrição (CSU01)

Sumário: Aluno usa o sistema para realizar inscrição em disciplinas.

Ator Primário: Aluno

Atores Secundários: Sistema de Faturamento

Precondições: O Aluno está identificado pelo sistema.

Fluxo Principal

1. O Aluno solicita a realização de inscrição.
2. O sistema apresenta as disciplinas disponíveis para o semestre corrente e para as quais o aluno tem pré-requisitos.
3. O Aluno seleciona as disciplinas desejadas e as submete para inscrição.
4. Para cada disciplina selecionada, o sistema aloca o aluno em uma turma que apresente uma oferta para tal disciplina.
5. O sistema informa as turmas nas quais o Aluno foi alocado. Para cada alocação, o sistema informa o professor, os horários e os respectivos locais das aulas de cada disciplina.
6. O Aluno confere as informações fornecidas.
7. O sistema envia os dados sobre a inscrição do aluno para o Sistema de Faturamento e o caso de uso termina.

Fluxo Alternativo (4): Inclusão em lista de espera

- a. Se não há oferta disponível para alguma disciplina selecionada pelo aluno, o sistema reporta o fato e fornece a possibilidade de inserir o Aluno em uma lista de espera.
- b. Se o Aluno aceitar, o sistema o insere na lista de espera e apresenta a posição na qual o aluno foi inserido na lista. O caso de uso retorna ao passo 4.
- c. Se o Aluno não aceitar, o caso de uso prossegue a partir do passo 4.

Fluxo de Exceção (4): Violação de RN01

- a. Se o Aluno atingiu a quantidade máxima de inscrições (RN01), o sistema informa ao aluno a quantidade de disciplinas que ele pode selecionar, e o caso de uso retorna ao passo 2.

Pós-condições: O aluno foi inscrito em uma das turmas de cada uma das disciplinas desejadas, ou foi adicionado a uma ou mais listas de espera.

Regras de Negócio: RN01, RN02, RN03

MODELOS DE PROCESSOS DE SOFTWARE E DE CICLO DE VIDA

Johnatan Oliveira &
Eveline Alonso Veloso*

* Adaptado do material da Prof.^a Maria Augusta
Vieira Nelson

O que é um Processo de *Software*?



- **Conjunto estruturado de passos;**
 - necessários para se **desenvolver** um **sistema de *software***;
 - o produto da execução do processo.
- As atividades, as técnicas, a duração, a ordem, a distribuição no tempo;
 - podem variar de processo para processo.

O que é um Processo de *Software*?



- É a “**receita**” seguida por um projeto:
 - não é o produto final;
 - não é o projeto.

Camadas da Engenharia de *Software*

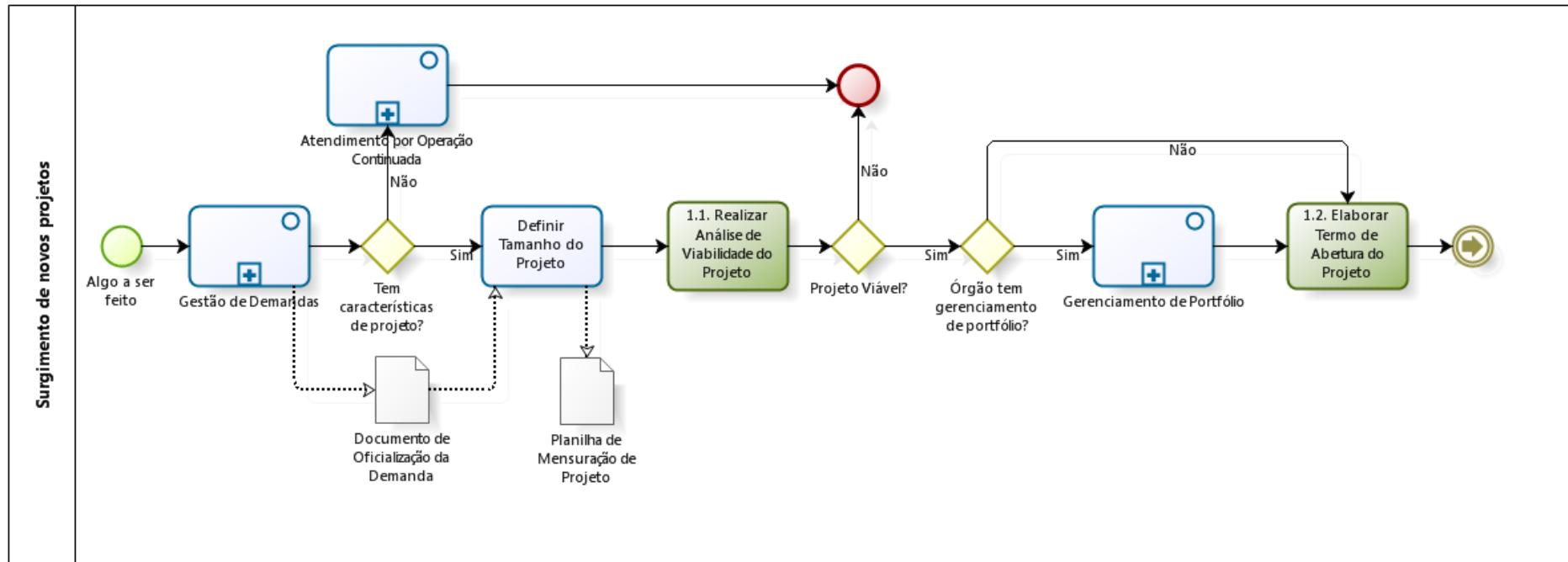


O que é um Modelo de Processo de *Software*?



- **Representação abstrata** de um processo de *software*.
- Em geral, **baseia-se** no **modelo de ciclo de vida** do *software*.

Modelo de Processo de Software – Exemplo



Powered by
bizagi
Modeler

O que é Ciclo de Vida?



- Compreende a **vida útil** do *software*;
- desde a concepção da **ideia inicial** do *software*;
- até o momento em que é aposentado e **retirado do ambiente de produção** na empresa.
- Determina seu **caráter temporal**.

O que é Ciclo de Vida?



- "Estrutura contendo **processos, atividades e tarefas** envolvidas no **desenvolvimento, operação e manutenção** de um produto de *software*, abrangendo a **vida do sistema**, desde a **definição** de seus **requisitos** até o **término** de seu **uso.**"

NBR ISO/IEC 12207:1998

Quais são os Elementos de um Modelo de Processo de *Software*?



- **Atividade:**

- termo genérico utilizado para designar **unidades de trabalho**;
- indica **o que precisa ser feito**;
 - e a **técnica** que deve ser **empregada**.

- **Técnica:**

- **método ou prática aplicável** à execução de um **conjunto de atividades**.

Quais são os Elementos de um Modelo de Processo de *Software*?

- **Fase:**
 - **divisão** maior de um **processo**;
 - para **fins gerenciais**;
 - corresponde aos **pontos** principais de **aceitação** por parte do cliente.
- **Iteração:**
 - **parte** constituinte de uma **fase**;
 - **atinge** um **conjunto** bem definido de **metas parciais** de um **projeto**.

Quais são os Elementos de um Modelo de Processo de *Software*?



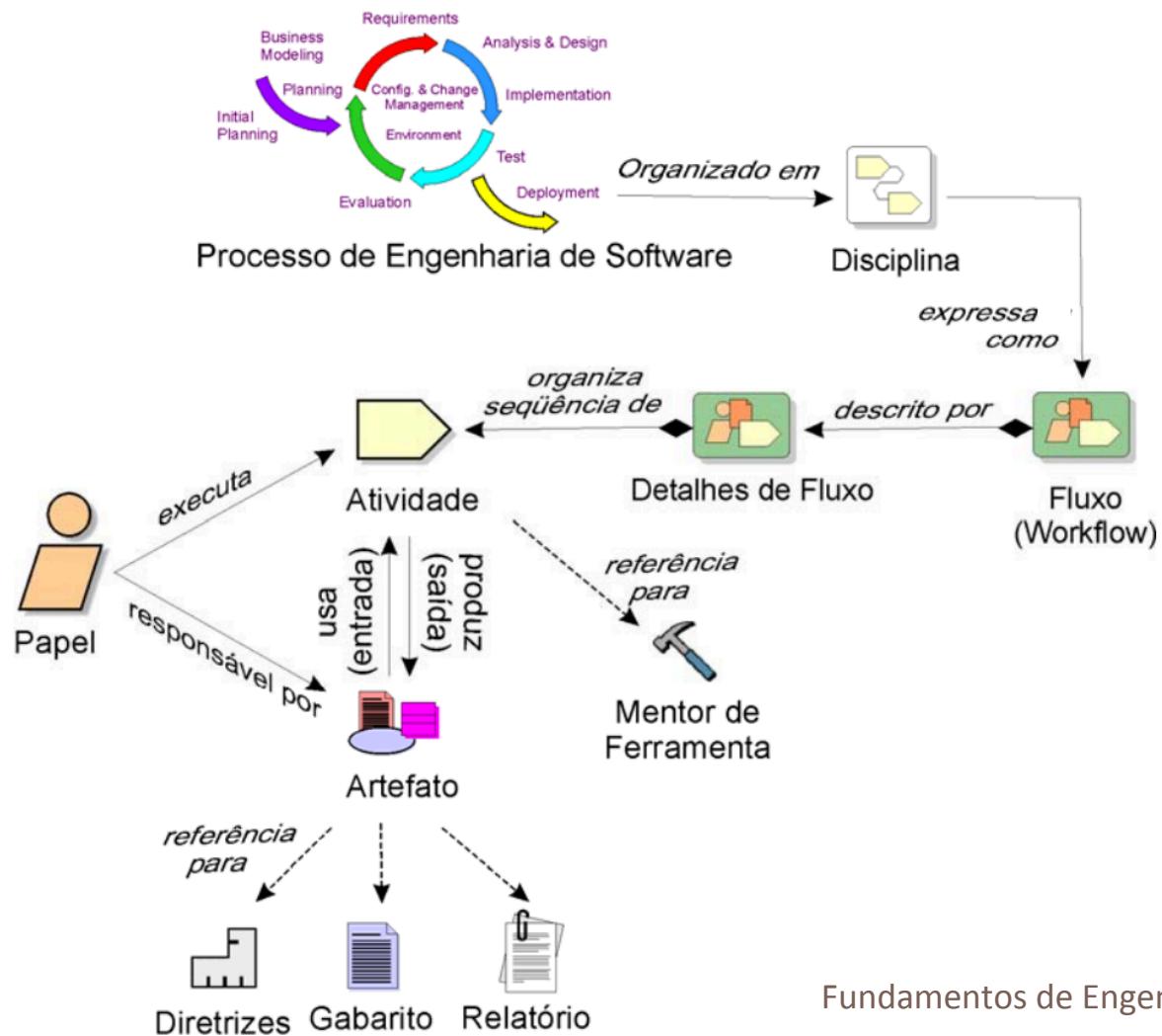
- **Produto ou artefato:**
 - **resultado** de uma **atividade** do processo;
 - representa um dos **resultados da execução** do **processo**.
 - O **término** de **fases** e **iterações** geralmente está associado com a **entrega** de um conjunto de **artefatos**.
- **Insumo:**
 - **artefato** que serve de **entrada** para a execução de uma **atividade** do processo.

Quais são os Elementos de um Modelo de Processo de *Software*?

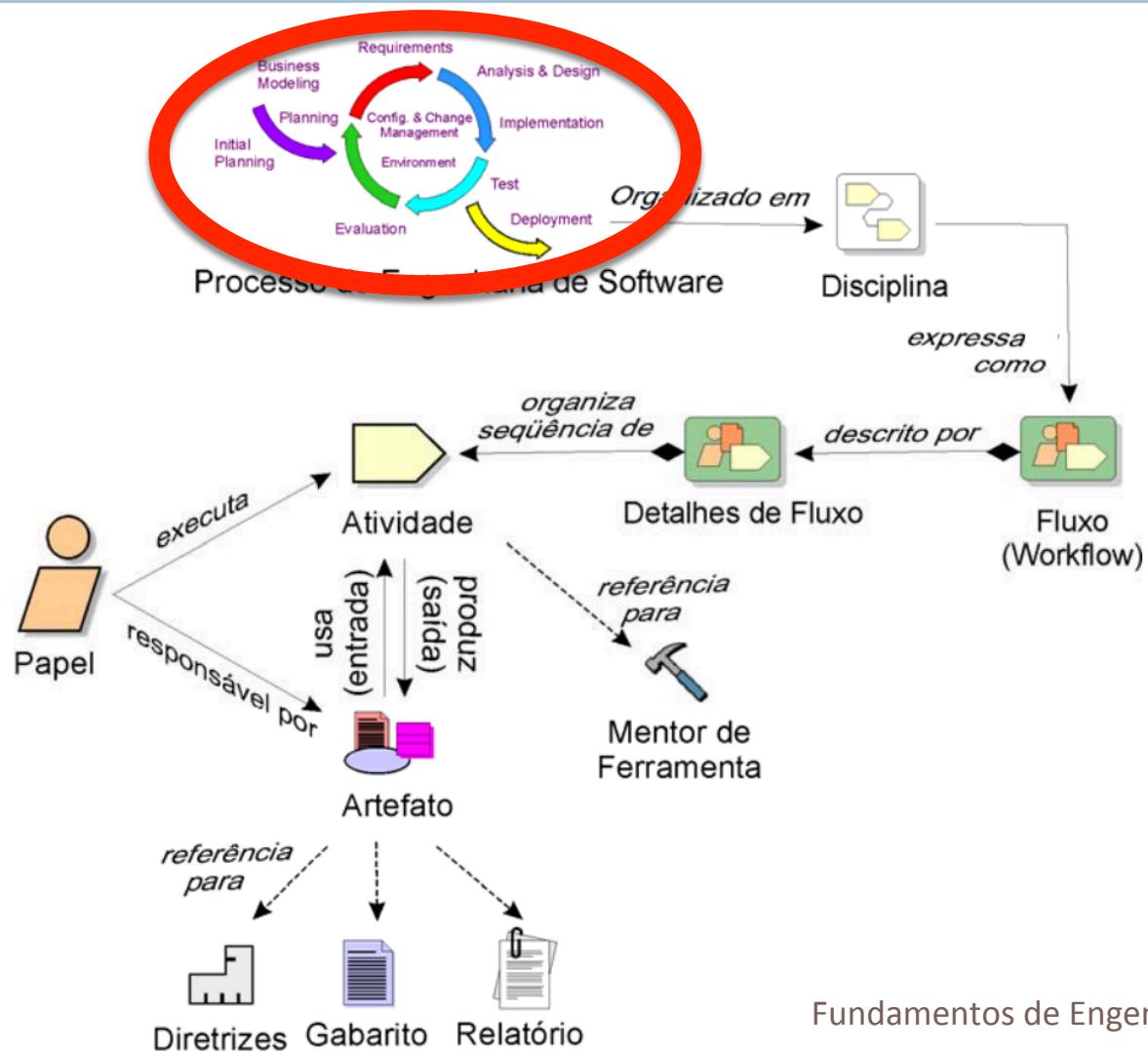


- **Papel:**
 - reflete as **responsabilidades** das **pessoas envolvidas** no processo.
- **Pré e pós-condições:**
 - **declarações** que são **verdadeiras**;
 - **antes e depois** de uma **atividade** do processo ser **executada** ou um **produto** ser **produzido**.

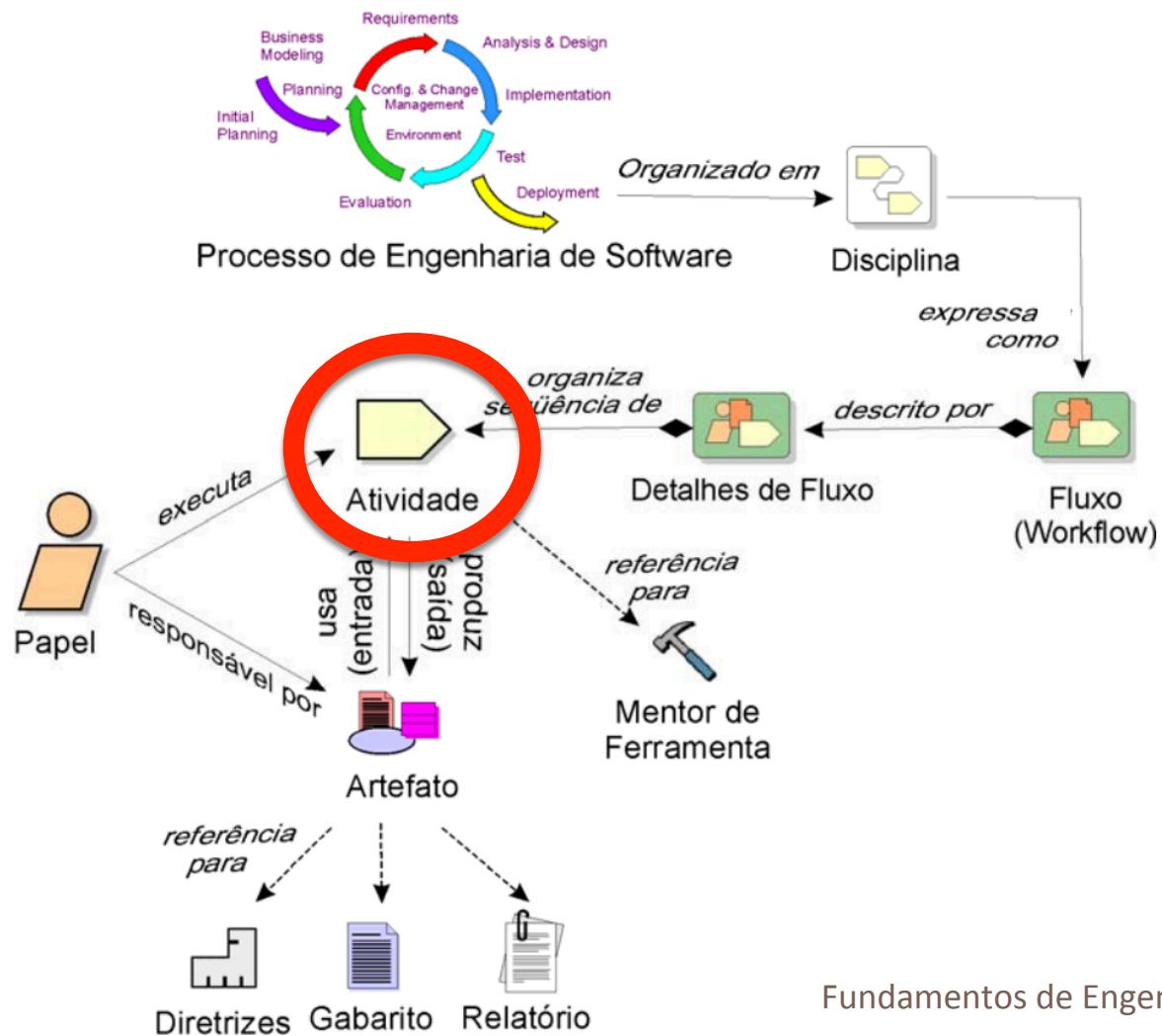
Elementos de um Modelo de Processo de *Software* – Exemplos



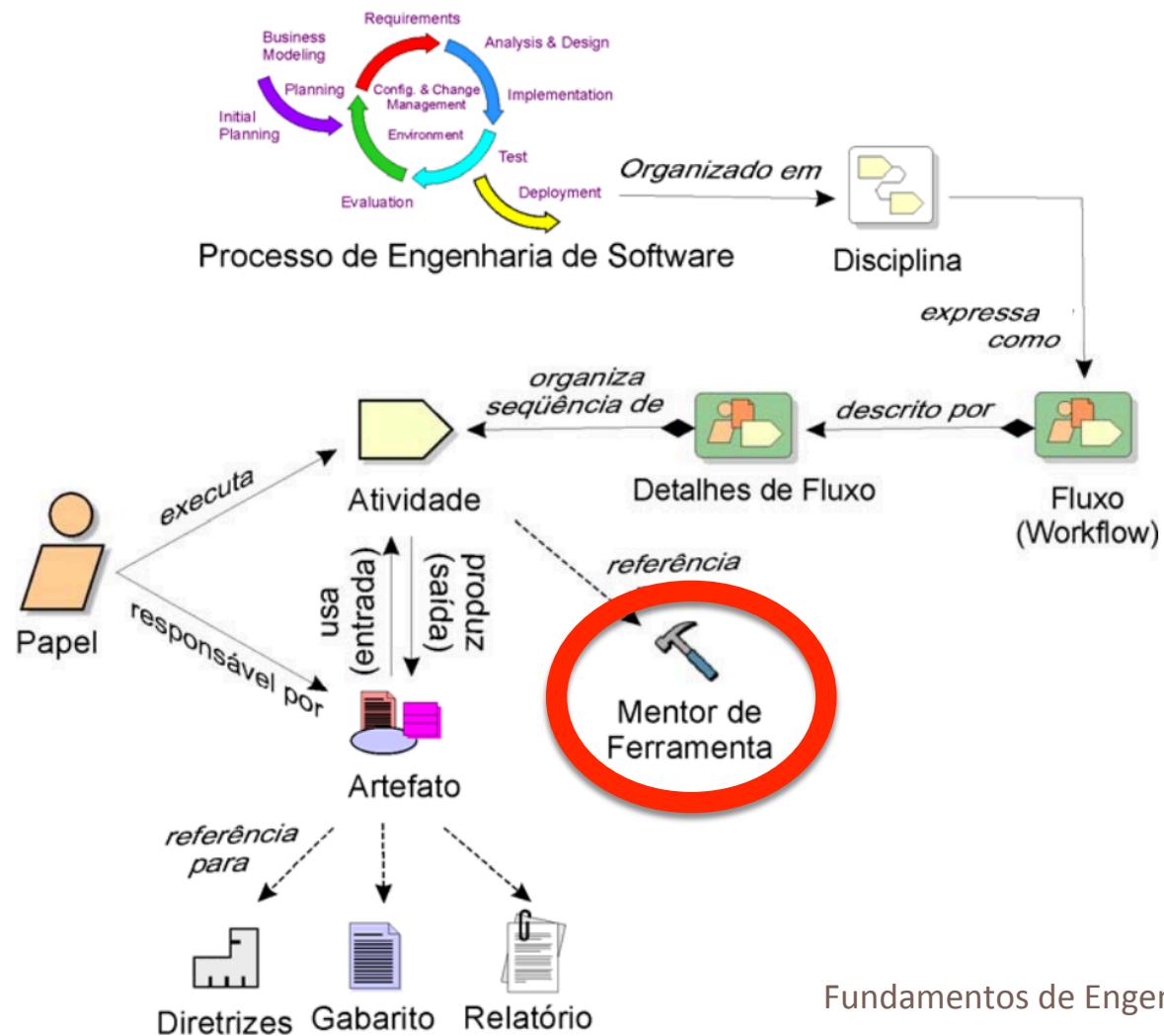
Elementos de um Modelo de Processo de *Software* – Exemplos



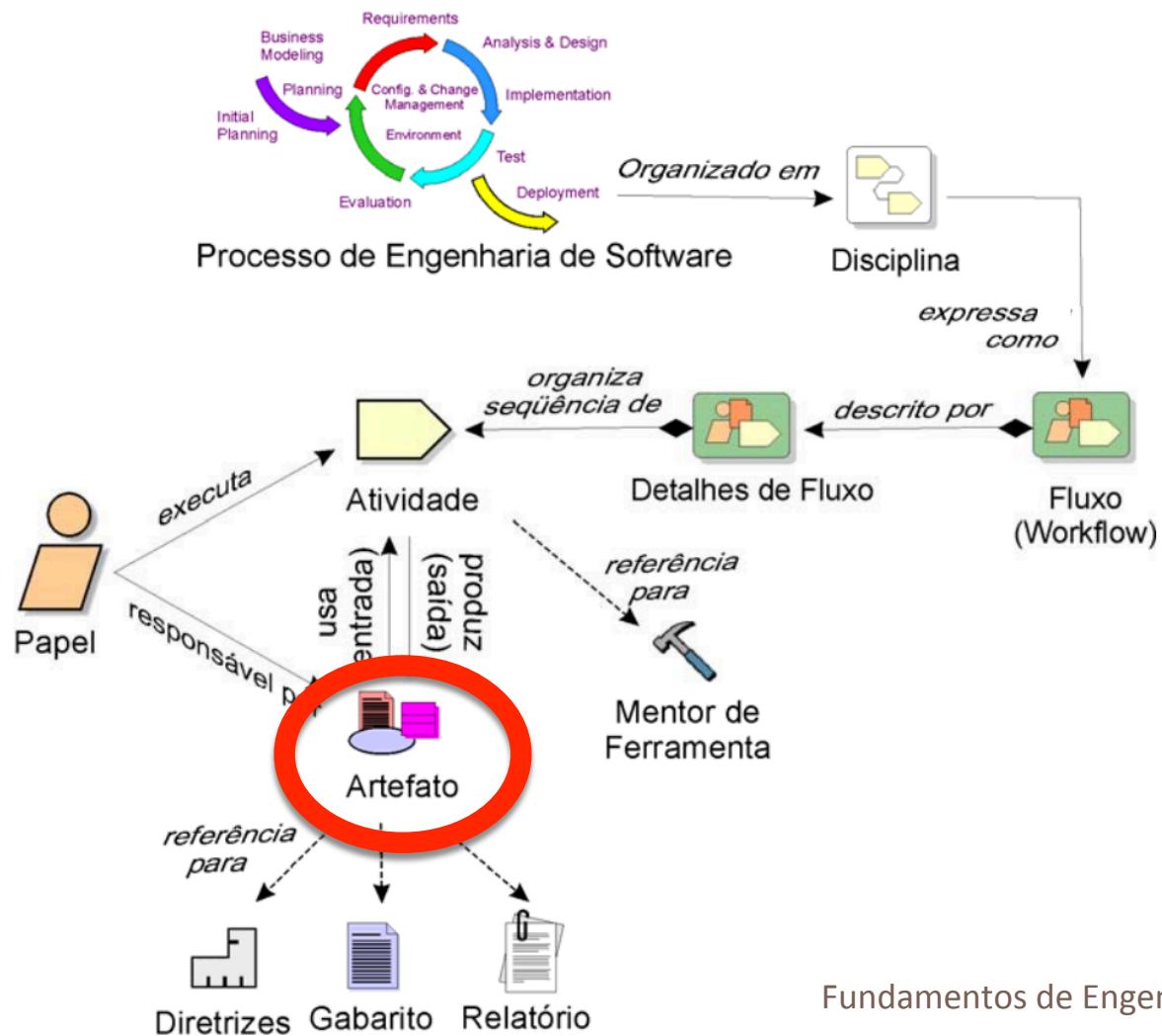
Elementos de um Modelo de Processo de *Software* – Exemplos



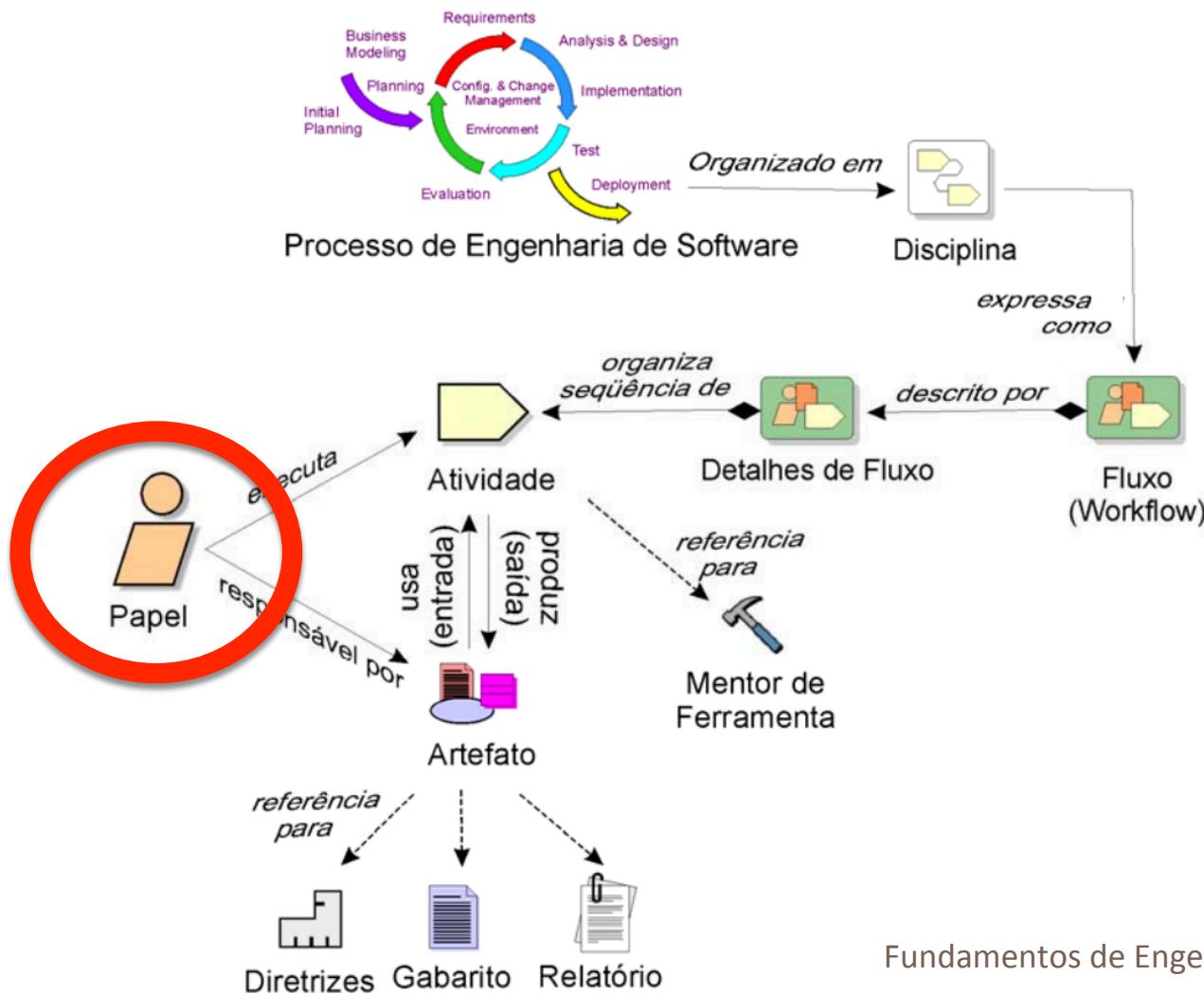
Elementos de um Modelo de Processo de *Software* – Exemplos



Elementos de um Modelo de Processo de *Software* – Exemplos



Elementos de um Modelo de Processo de *Software* – Exemplos

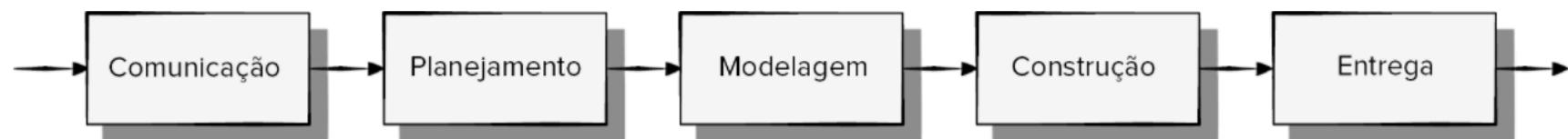


Modelos de Ciclo de Vida

- Modelo Sequencial Linear
- Modelo em Cascata
- Modelo V
- Modelo Incremental
- Modelo Iterativo
- Modelo Evolucionário
- Modelo Espiral
- Modelo Incremental e Iterativo

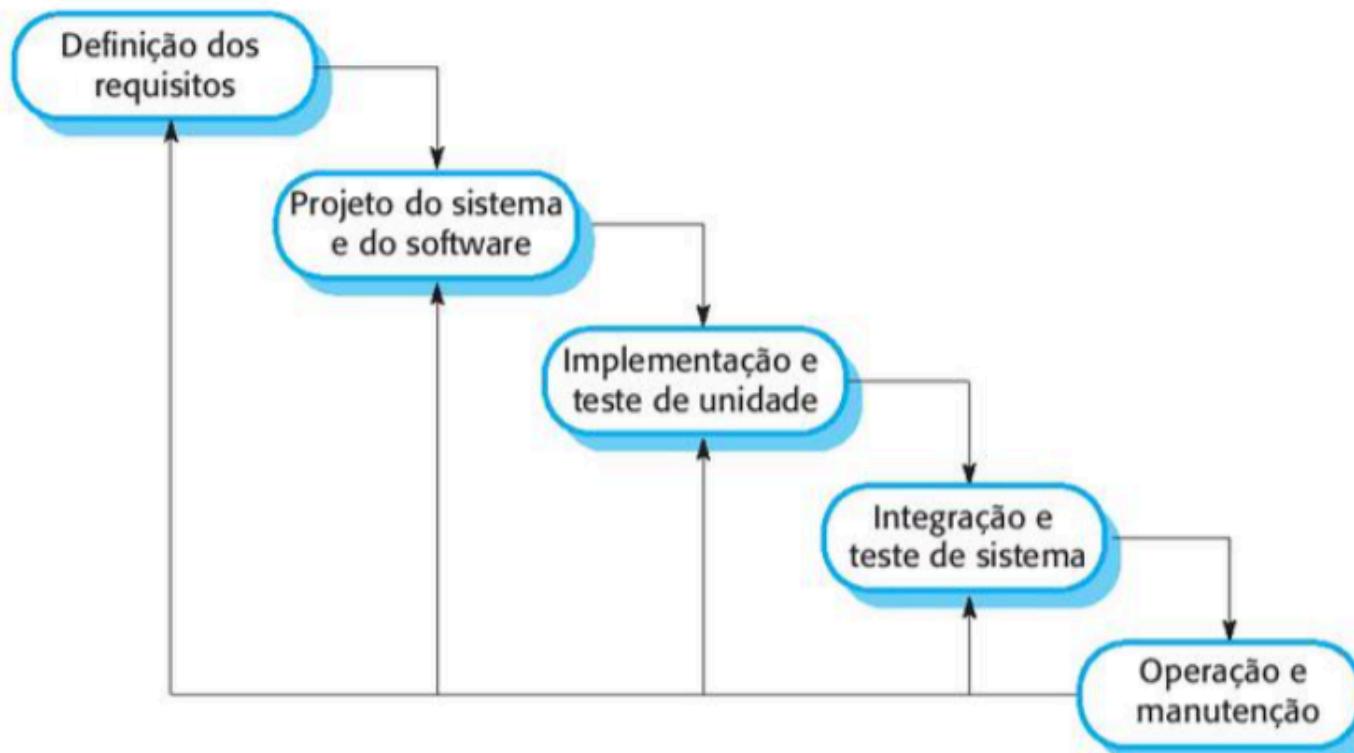
Modelo Sequencial Linear

□ Executa cada uma de suas **atividades metodológicas em sequência.**



Modelo Em Cascata

□ Modelo Sequencial Linear.



Modelo Em Cascata

- Trabalho flui de modo relativamente **linear**:
 - **pontos de controle bem definidos**;
 - **facilitam a gestão** do projeto.
- Aplicável em projetos em que os **requisitos** estão **bem definidos** desde o **íncio**;
 - e são razoavelmente **estáveis**.
- Prevê ***feedback loops***;
 - **correção** de possíveis **erros** em **etapas anteriores**.

Modelo Em Cascata

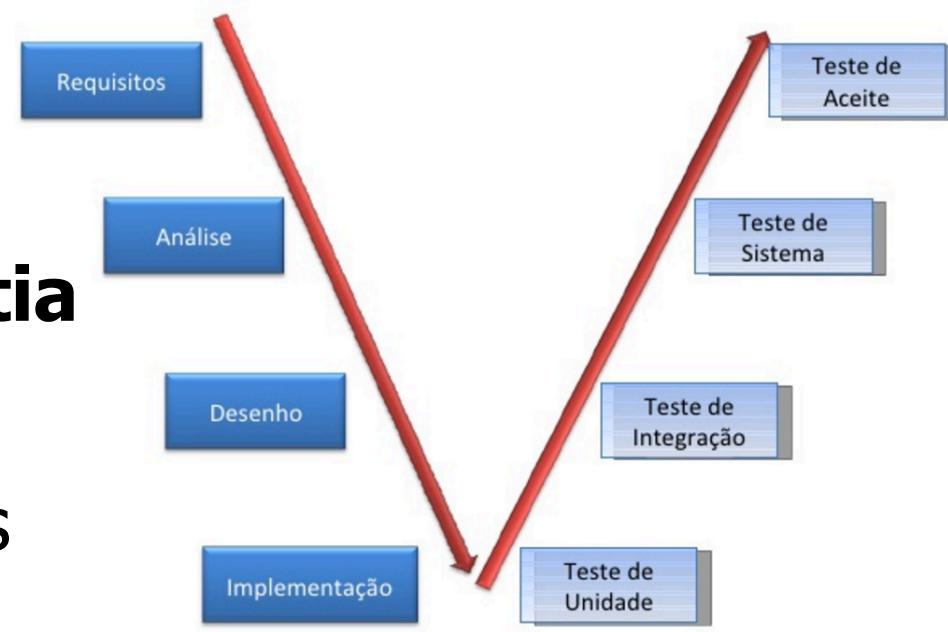
□ Problemas:

- é **difícil** para o cliente **estabelecer todas** as suas **necessidades**;
 - no **início** do projeto;
- **erros graves** podem **não ser detectados**;
 - até o *software* ser testado;
- **baixa visibilidade** para o cliente:
 - uma **versão operacional** do *software* não estará **disponível** antes de estarmos próximos ao **final do projeto**.

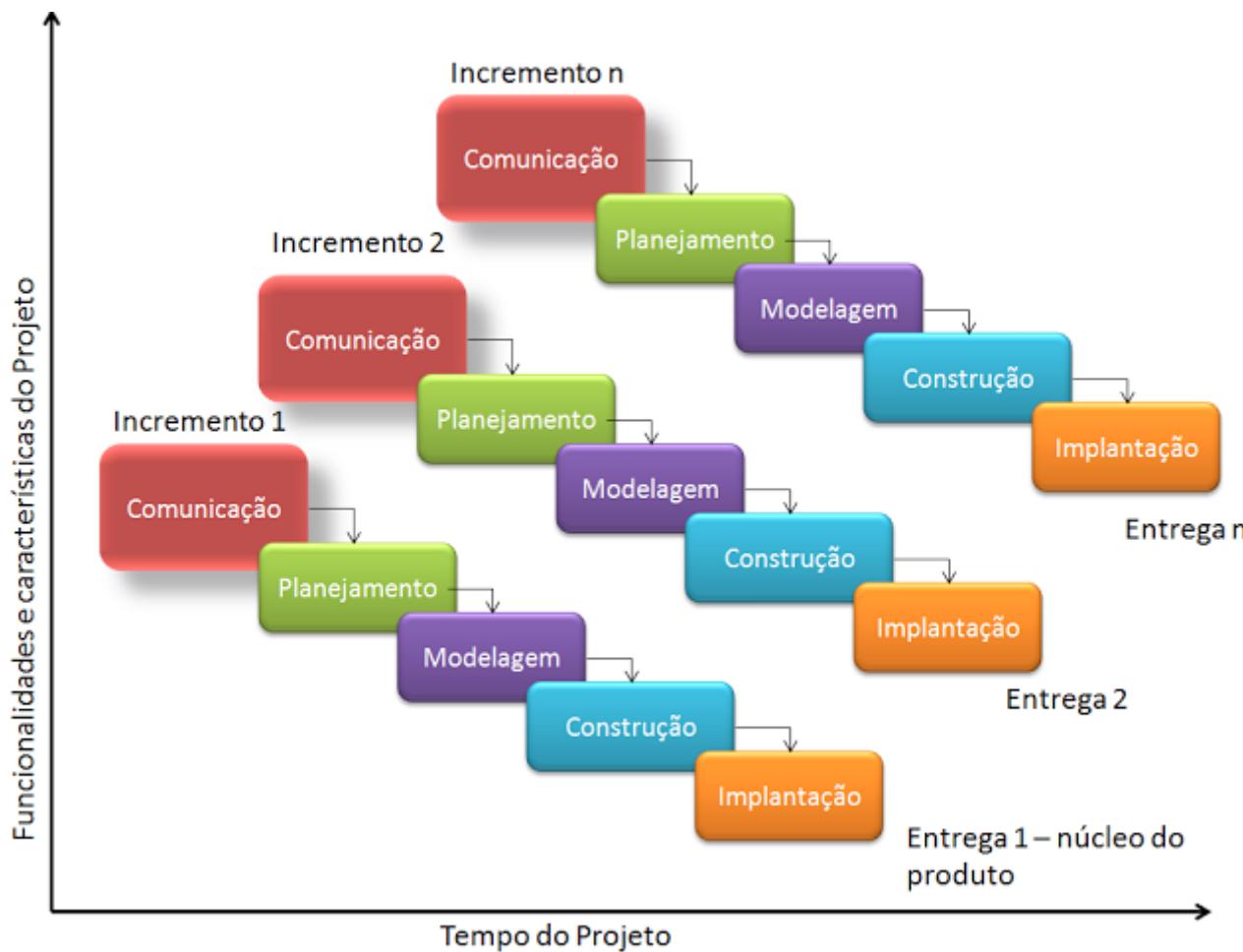
Modelo V

- Variação do modelo **em Cascata**.

- Descreve a **relação** entre:
 - ações de **garantia da qualidade**;
 - ações associadas às **atividades anteriores**.



Modelo Incremental

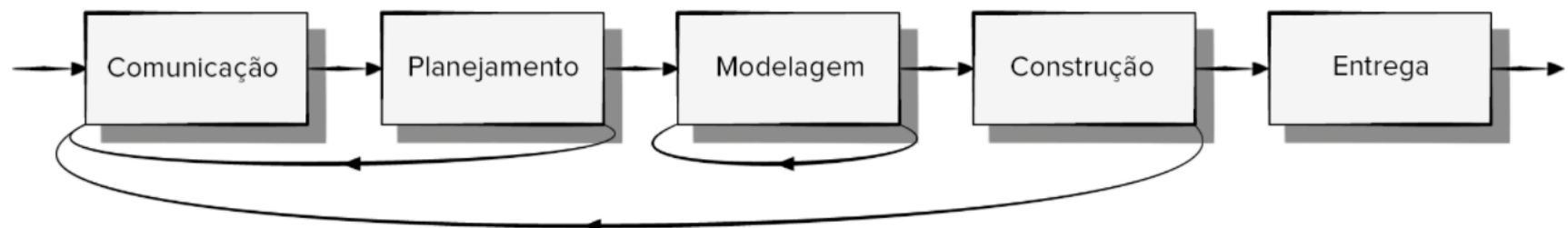


Modelo Incremental

- Aplica **sequências lineares escalonadas**;
 - à medida que o **tempo** avança.
- **Cada sequência linear produz um incremento entregável do software.**
- Modelo incremental puro é um **estilo linear**:
 - a **equipe conhece todo o escopo no início**;
 - só possível com grandes análise e *design* iniciais;
 - e então começa a **construir partes**.

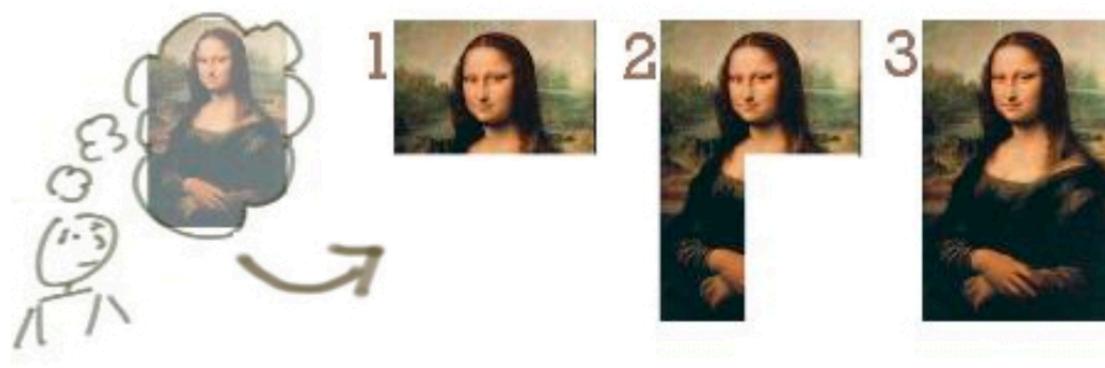
Modelo Iterativo

- Repete uma ou mais de suas atividades;
- antes de prosseguir para a seguinte.

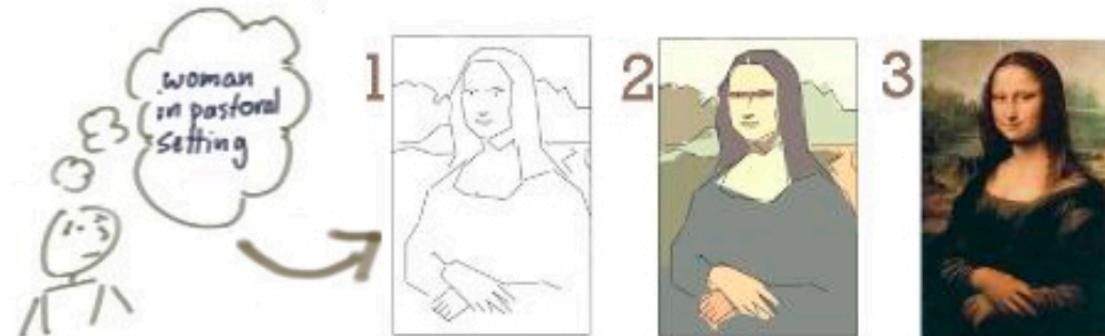


Incremental x Iterativo

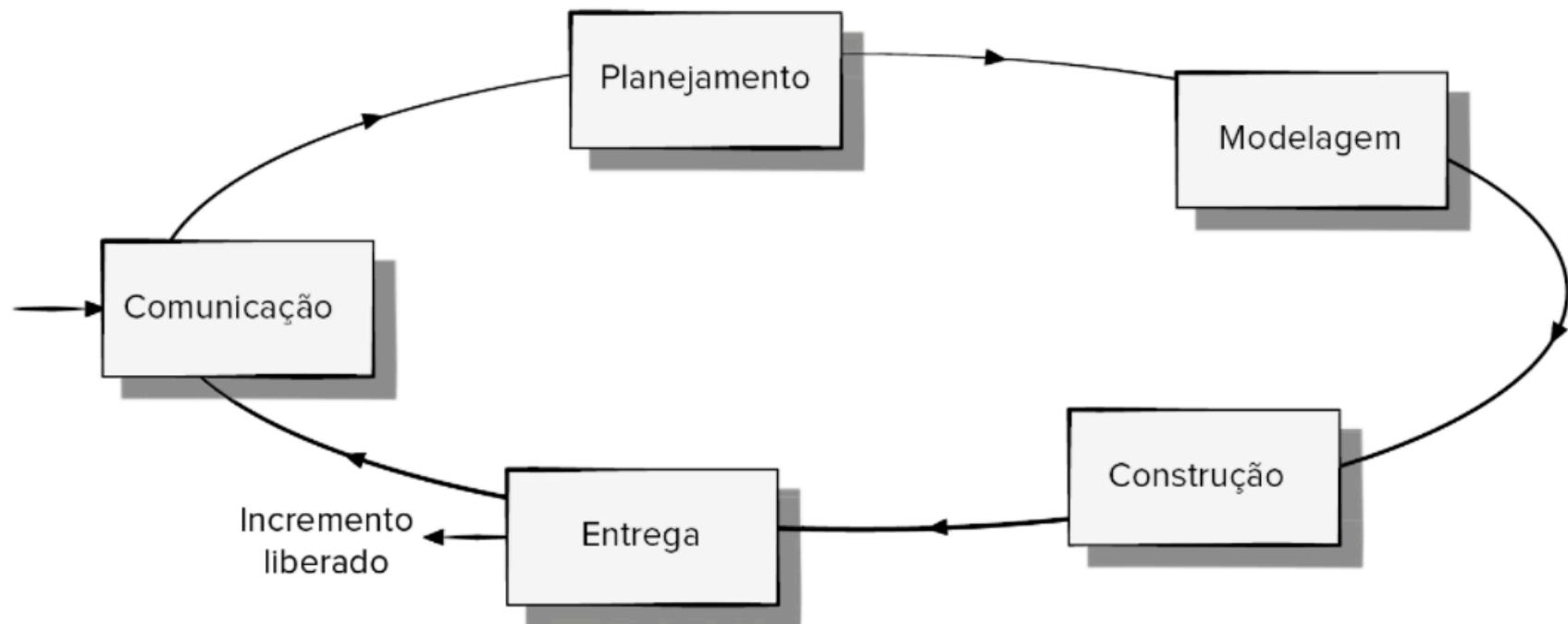
Incremental



Iterative



Modelo Evolucionário

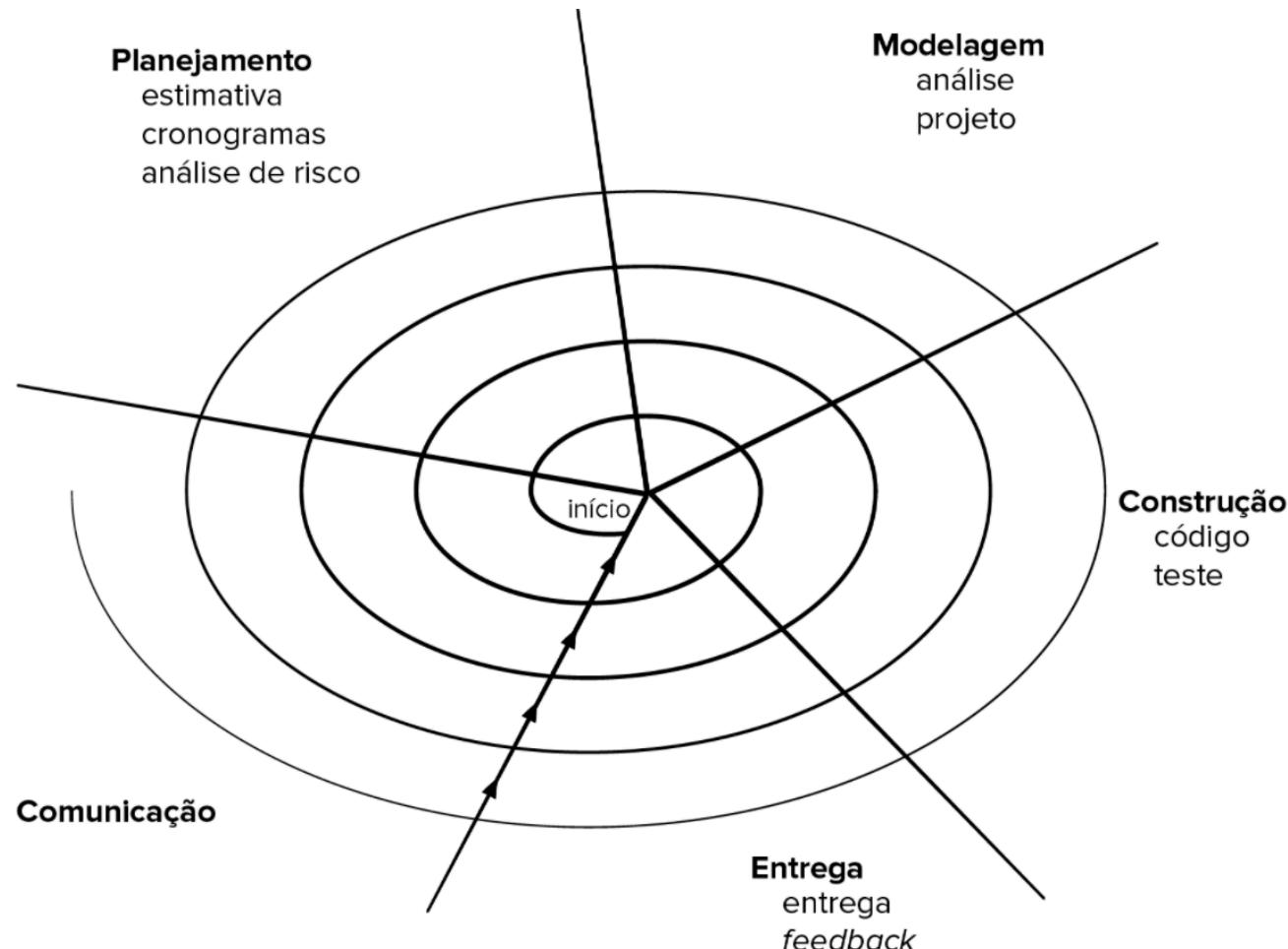


Modelo Evolucionário



- Executa suas **atividades de forma circular**.
- Desenvolve um **produto que cresce e muda**.
- É **iterativo**.
- Rápido desenvolvimento de **versões**;
- **cada vez mais completas do software.**

Modelo Espiral



Modelo Espiral



- O **produto é desenvolvido em iterações.**
- Cada nova **iteração**;
 - corresponde a uma **volta na espiral.**
- Os **problemas identificados** em uma iteração;
 - são utilizados para **alimentar** a próxima.

Modelo Espiral

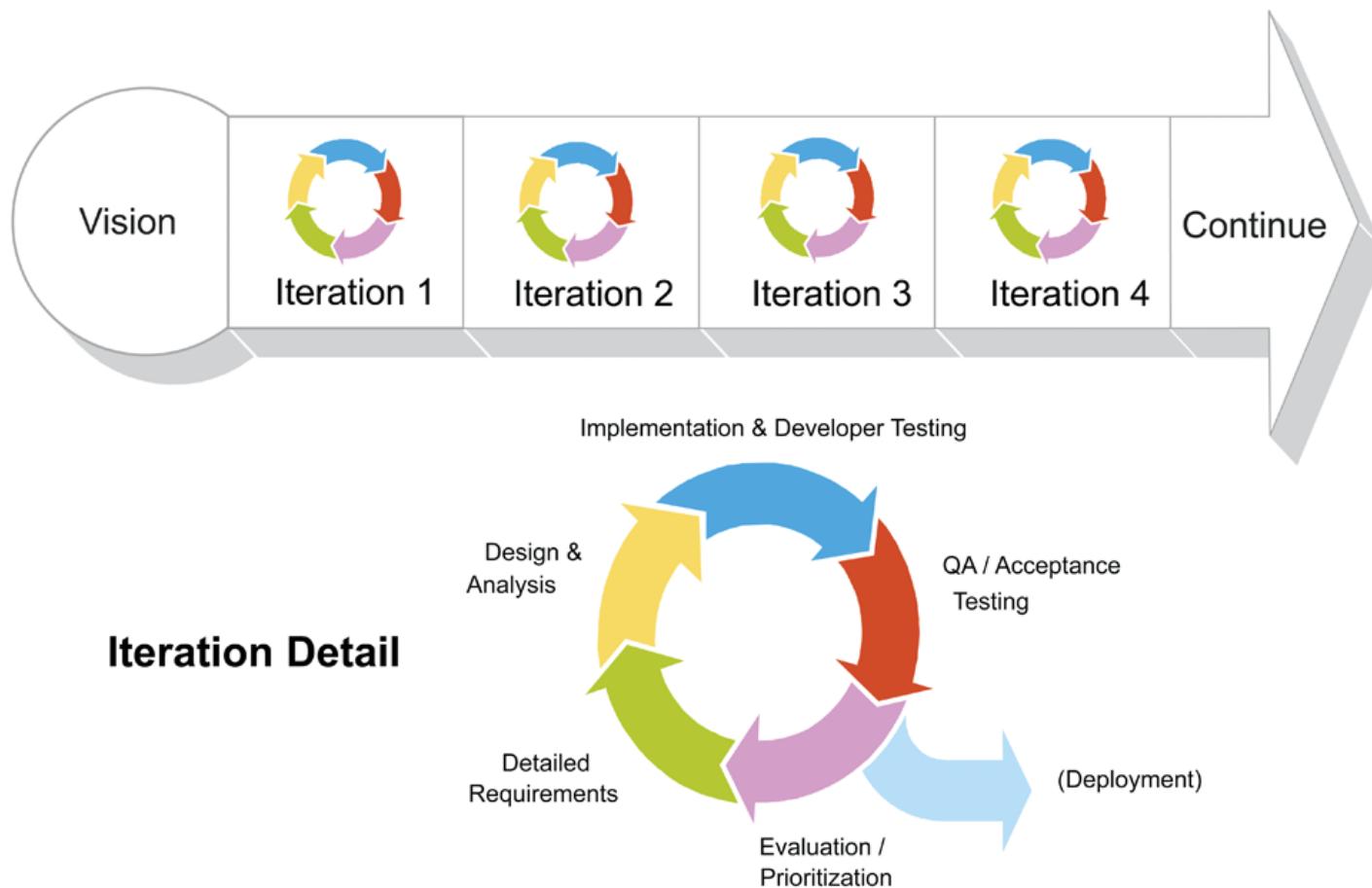


- Modelo **evolucionário dirigido por riscos:**
 - *software* é desenvolvido em uma **série de versões evolucionárias;**
 - **consideração** direta dos **riscos técnicos** em todos os estágios do projeto;
 - **reduz** os **riscos** antes destes se tornarem problemáticos.

Modelo Espiral

- Permite construir produtos em **prazos curtos**.
- Novas **características** são **adicionadas à medida que o problema é entendido**;
 - favorecendo para que as **verdadeiras necessidades** apareçam.
- É um modelo **complexo**;
 - requer **gestão sofisticada** para ser previsível e confiável.
- Requer **desenho robusto**;
 - para evitar que a estrutura do produto se degride.

Modelo Incremental e Iterativo

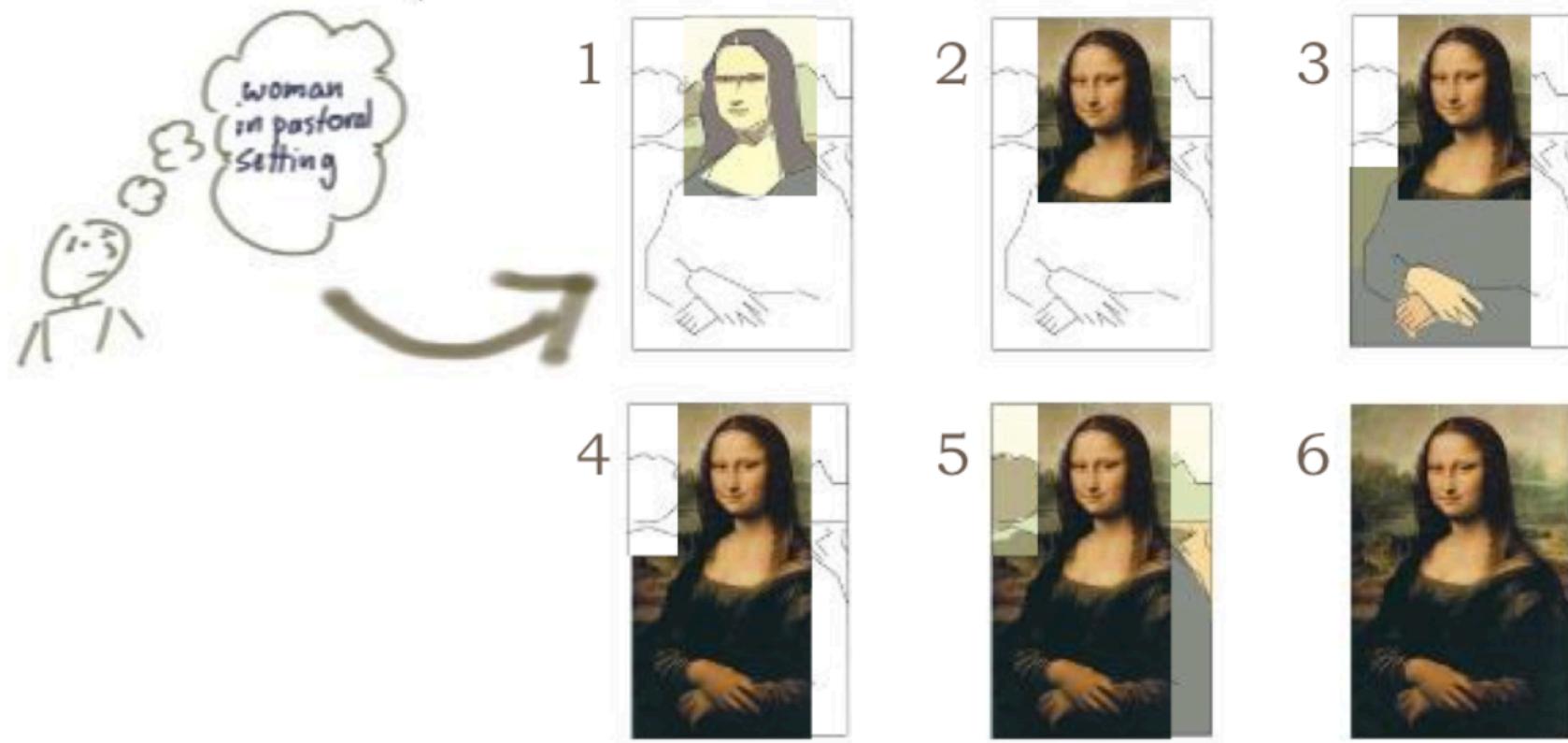


Modelo Incremental e Iterativo



- **Incremental:**
 - **adiciona recursos** completamente **novos**;
 - ampliando o escopo da funcionalidade oferecida.
- **Iterativo:**
 - cada **incremento** provavelmente também **refinará as funcionalidades existentes**.

Modelo Incremental e Iterativo



Processos Dirigidos por Planos

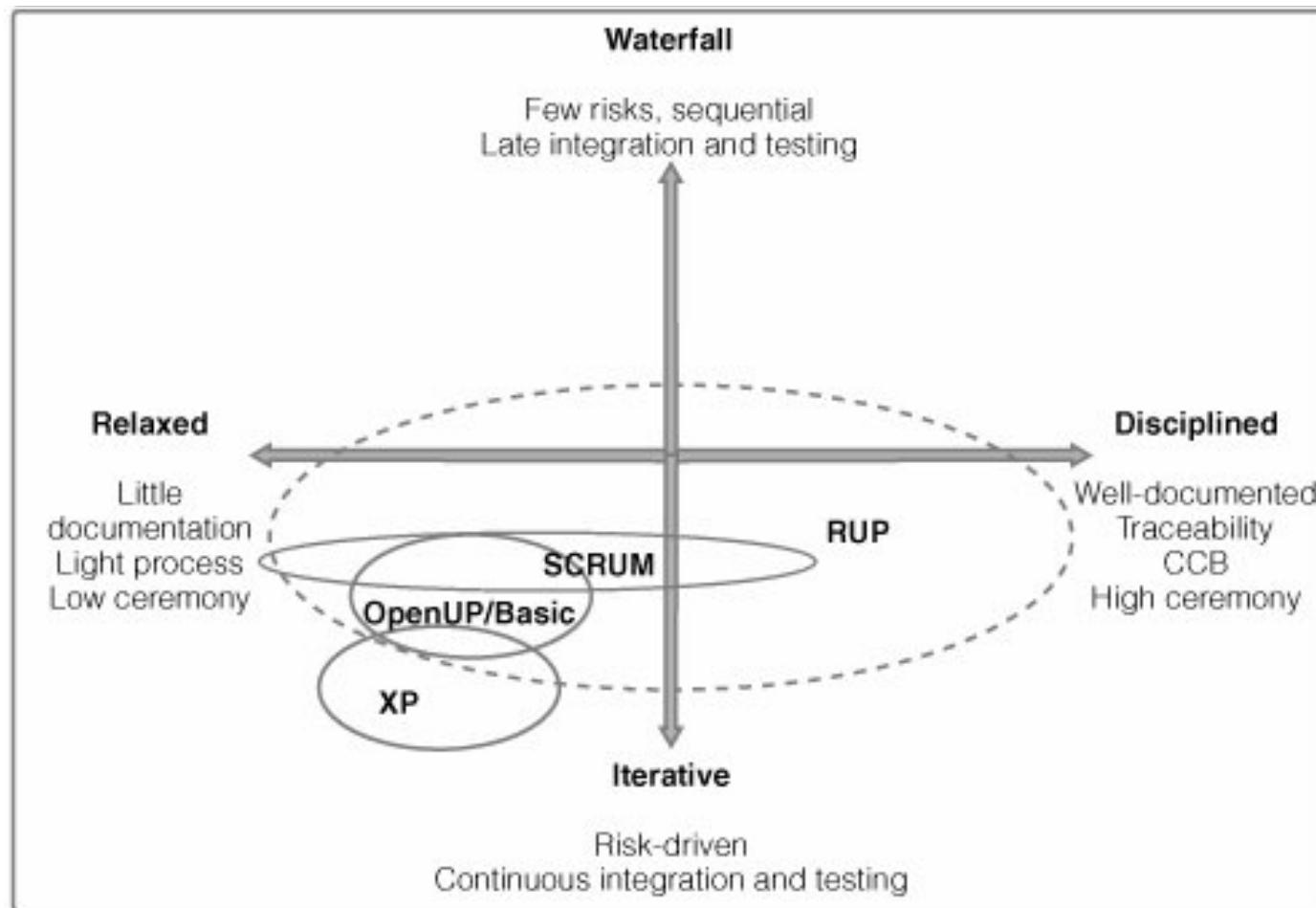
- Todas as **atividades** do processo são **planejadas com antecedência**;
 - ▣ **progresso** é medido em relação a esse **plano**.
- **Prescrevem** um conjunto de **elementos de processo**:
 - ▣ atividades, tarefas, artefatos, mecanismos de garantia da qualidade e de controle de mudanças;
 - ▣ e a **forma** pela qual esses elementos **se relacionam**.
- Também conhecidos como **processos prescritivos**.

Processos Ágeis



- **Planejamento incremental e contínuo.**
- **Acomodam** mais facilmente **mudanças**;
 - para refletir alterações nos requisitos do cliente.
- Consistem em um conjunto de **valores**, **princípios** e **práticas** voltados para o desenvolvimento do *software*;
 - que podem ser aplicados de **forma leve e efetiva**.

Processos Dirigidos por Planos x Processos Ágeis



Referências



- PRESSMAN, Roger. **Engenharia de software: uma abordagem profissional.** Capítulo 2.
- SOMMERVILLE, Ian. **Engenharia de software.** Capítulo 2.
- NBR ISO/IEC 12207:1998
 - https://pt.wikipedia.org/wiki/ISO/IEC_12207

MÉTODOS ÁGEIS

Eveline Alonso Veloso*

* Adaptado do material da Prof.^a Maria Augusta
Vieira Nelson

O que é um Método Ágil de Desenvolvimento de *Software*?

- Método para o desenvolvimento **iterativo e incremental** de *software*;
- **incrementos** muito **pequenos**;
- novas **versões** do *software*, criadas e **disponibilizadas ao cliente**;
 - em um intervalo de **duas a quatro semanas**.

O que é um Método Ágil de Desenvolvimento de *Software*?



- **Cientes envolvidos** no processo;
 - fornecem ***feedback*** rápido
 - sobre os **requisitos**;
 - e **o que foi produzido**.
- **Minimiza-se a documentação**;
 - utiliza-se mais a **comunicação informal**;
 - do que reuniões formais com documentos escritos.

Manifesto para o Desenvolvimento Ágil de *Software*



- Grupo de 17 **desenvolvedores e consultores de software**;
 - dispostos a **atacar problemas típicos** relacionados ao **desenvolvimento de software**.
- Criação da **Aliança Ágil (Agile Alliance)**, em **2001**:
 - organização não-lucrativa;
 - auxilia indivíduos e organizações que utilizam abordagens ágeis para desenvolver *software*.

Manifesto para o Desenvolvimento Ágil de *Software*



- Definição de **4 valores**;
 - ▣ preferências, não alternativas.
- **Indivíduos e interações**;
 - ▣ têm mais valor que processos e ferramentas.
- **Software funcionando**;
 - ▣ tem mais valor que uma documentação abrangente.
- **Colaboração com o cliente**;
 - ▣ tem mais valor que negociação contratual.
- **Respostas a mudanças**;
 - ▣ têm mais valor que aderência a um plano.

Indivíduos e Interações em vez de Processos e Ferramentas



- Os fatores mais importantes a serem considerados são as **pessoas**;
 - e a forma **como** elas **trabalham juntas**.
- Se as pessoas não trabalham juntas adequadamente;
 - nem as melhores ferramentas nem os melhores processos ajudarão.

Software Funcionando em vez de Documentação Abrangente

- A maioria dos desenvolvedores prefere **escrever *software***;
 - a documentos.
- ***Software* funcionando é mais importante do que artefatos intermediários;**
 - nem sempre uma boa prática.

Colaboração com o Cliente em vez de Negociação Contratual



- Somente **clientes e usuários** podem **dizer** o que eles realmente **querem e precisam**;
 - eles provavelmente **não pedem** as **coisas certas** da **primeira vez**.
- Contratos são importantes;
 - mas o **entendimento** do **papel** de **cada um**;
 - com **adequada comunicação**;
 - é mais importante ainda.

Respostas a Mudanças em vez de Aderência a um Plano



- **Pessoas mudam suas prioridades;**
 - **mudança é uma realidade do desenvolvimento de *software*.**
- Não existe problema nenhum em se ter um **plano**;
 - ele só precisa ser **maleável para acomodar as mudanças.**



Princípios da Aliança Ágil

- **Satisfação do cliente** desde o início;
 - por meio de **entrega antecipada e contínua** de **software de valor**.
 - **Maior prioridade** do desenvolvimento ágil.
- **Modificações de requisitos** são **bem-vindas**;
 - mesmo que tardivamente no desenvolvimento.
 - Processos ágeis **submetem-se a mudanças**;
 - em favor da vantagem competitiva do cliente.

Princípios da Aliança Ágil



- **Entrega freqüente de *software* funcionando:**
 - de **duas semanas a dois meses**;
 - favorecendo a **menor escala**.
- *Software* funcionando:
 - **principal medida de progresso** do projeto.
- Construção de projetos por **indivíduos motivados**:
 - dê a eles o ambiente e o suporte que precisam;
 - e **confie** neles para obter o trabalho feito.

Princípios da Aliança Ágil

- **Equipe de negócios e desenvolvedores devem trabalhar em conjunto;**
 - **diariamente durante todo o projeto.**
- **Conversa face-a-face:**
 - método mais eficiente e efetivo de transmitir informação para e dentro de uma equipe de desenvolvimento de *software*.
- Processos ágeis promovem **desenvolvimento sustentável:**
 - clientes, desenvolvedores e usuários devem estar aptos a **manter um ritmo constante indefinidamente.**

Princípios da Aliança Ágil

- Atenção contínua à **excelência técnica e bom desenho**;
 - facilitam a agilidade.
- **Simplicidade é essencial**:
 - arte de maximizar a quantidade de trabalho não realizado.
- As **melhores arquiteturas, requisitos e desenhos**;
 - emergem de **equipes auto-organizadas**.
- Em intervalos regulares, a **equipe** deve refletir sobre **como pode se tornar mais efetiva**;
 - **ajustar** seu comportamento adequadamente.

Exemplos de Métodos Ágeis



- *eXtreme Programming (XP);*
 - Programação Extrema
- SCRUM
- *Dynamic System Development Method (DSDM)*
- *Feature Driven Development (FDD)*
- *Crystal Families*

Referências



- PRESSMAN, Roger. **Engenharia de software: uma abordagem profissional.** Capítulo 3.
- SOMMERVILLE, Ian. **Engenharia de software.** Capítulo 3.
- AMBLER, Scott. *Agile Modeling, Wiley Computer Publishing, 2002.*

SCRUM

Eveline Alonso Veloso*

* Adaptado do material da Prof.^a Maria Augusta
Vieira Nelson

SCRUM



- Abordagem **empírica**;
 - baseada em **experiências prévias**.
- Aceita que o **problema** pode **não ser** totalmente **entendido** ou definido no **início**;
 - e que provavelmente os **requisitos mudarão com** o passar do **tempo**.

SCRUM



- Concentra-se em maximizar a habilidade em **responder de forma ágil** aos **desafios emergentes**.
- Foco:
 - **entrega do produto no menor tempo** possível.

SCRUM



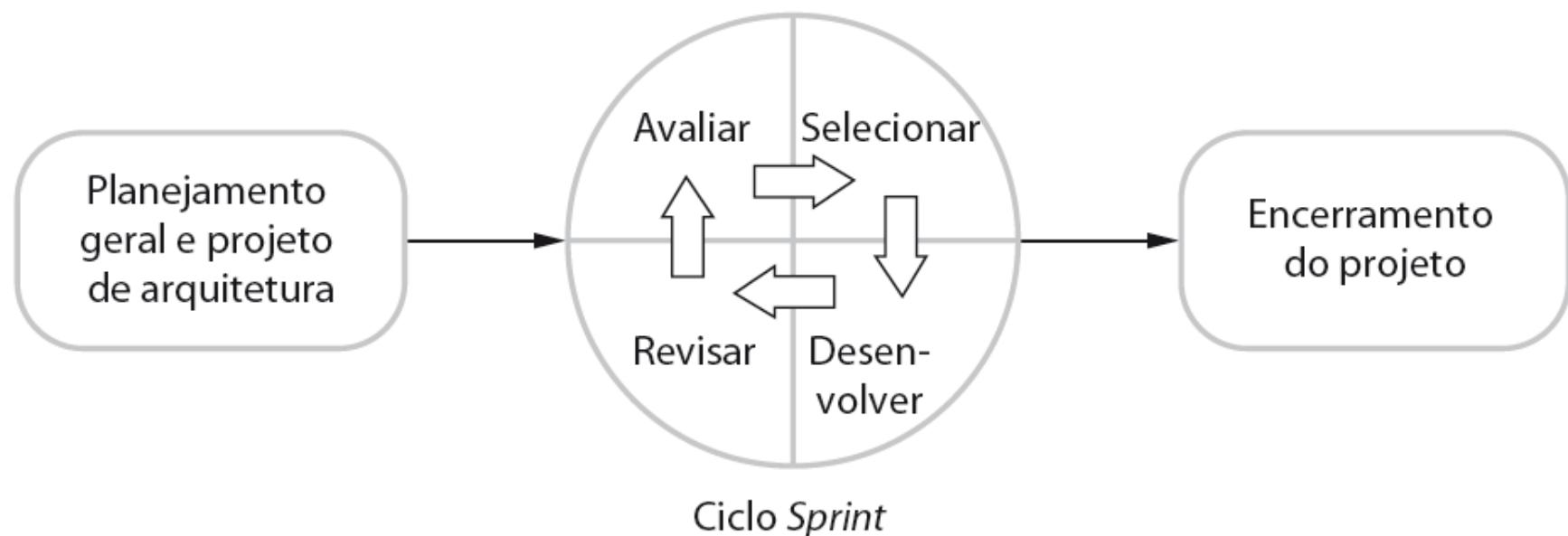
- Proporciona um **arcabouço** para a **organização ágil do projeto**;
- **não impõe** o uso de **práticas** de desenvolvimento **específicas**.
- Método **democrático**;
- **todos** os envolvidos **podem opinar**.

Fluxo do SCRUM

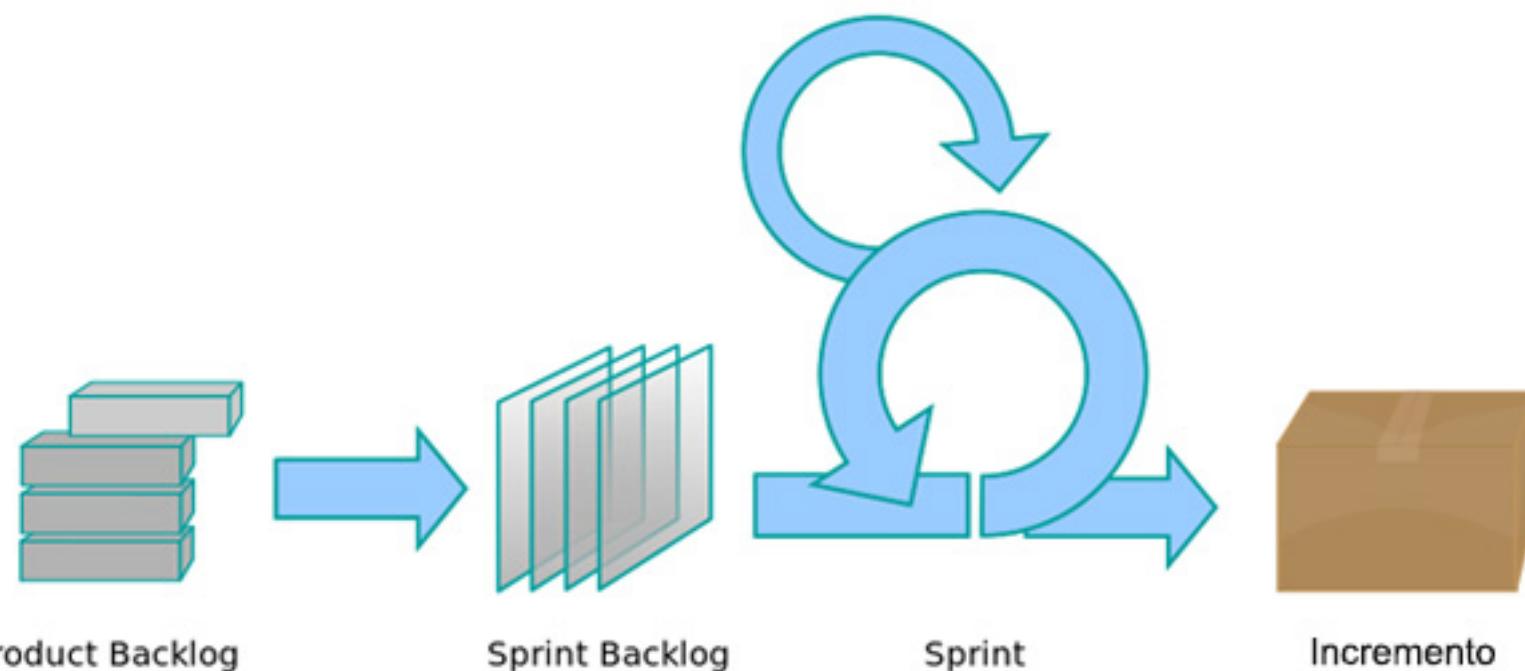


- Trabalho organizado a partir do ***backlog do produto:***
 - **lista** atualizada e **priorizada do que ainda precisa ser feito.**
- **Versões incrementais** do produto;
 - com **iterações curtas.**
- **Tarefas** são agrupadas em ***Sprints:***
 - blocos com **duração de 1 a 4 semanas.**

Fluxo do SCRUM



Fluxo do SCRUM



Equipes de Trabalho do SCRUM



- **Pequenas;**
- **Multidisciplinares;**
- **Auto-organizadas com liderança diluída;**
 - ▣ **facilitador: *Scrum Master*,**
- **Comunicação e cooperação entre os membros da equipe se intensificam;**
 - ▣ para produzir os melhores resultados.

Três Papéis



Product Owner



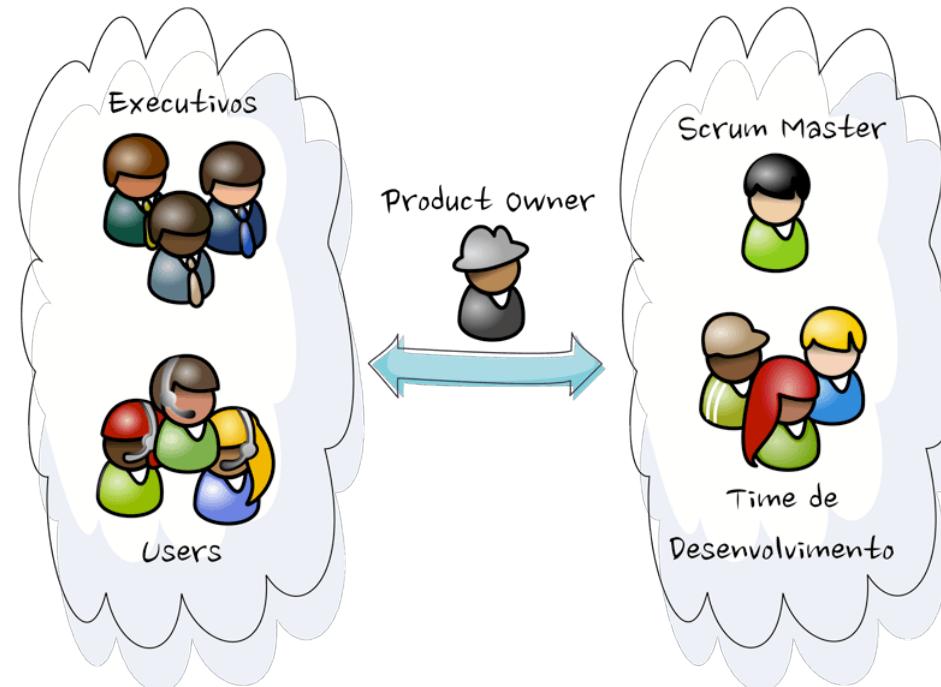
Scrum Master



Equipe de
Desenvolvimento

Product Owner – Dono do Produto

- **Representante do cliente** junto da equipe de trabalho do SCRUM.



Product Owner – Dono do Produto

- Responsável pelo **valor de negócio do produto**.
- **Define e prioriza as funcionalidades** do produto;
 - criando seu ***backlog***;
 - de acordo com seu valor de mercado.
- Especifica os **requisitos** incluídos em **cada incremento** do produto.
- **Avalia as iterações**:
 - **aceita ou rejeita o resultado** dos trabalhos.

Scrum Master – Facilitador

- **Não** se envolve com o **trabalho técnico**.
- **Responsável** pela **aplicação do SCRUM**.
 - Toma iniciativas para melhorar o trabalho e a produtividade da equipe;
 - **resolve os impedimentos**.
 - Garante a **colaboração** entre os **diversos papéis**.
 - **Escudo** contra **interferências externas**.



Team – Time de Desenvolvimento

- Possui **habilidades distintas**.
- **Auto-organizada**.
- **Reporta problemas** graves ao *Scrum Master*.
- Responsável por **entregar valor**;
 - por meio de **software com qualidade**.
- **Estima** as **durações** das **tarefas**.
- Cada membro do time;
 - **escolhe** suas **tarefas** na *Sprint*.
- **Executa** as **tarefas**.



Três Artefatos



- *Backlog* do produto.
- *Backlog* da *Sprint*.
- Incremento do produto.

Backlog do Produto



- **Lista atualizada e priorizada do que ainda precisa ser feito:**
 - **requisitos** ou características do produto;
 - que agregam valor de negócio para o cliente.
 - **mini-tarefas;**
 - com prazos de conclusão estimados.
- Descreve todo o **trabalho planejado**.
- O *backlog* do produto **nunca está completo**;
 - **enquanto o produto evolui** para atender às necessidades dos envolvidos.

Backlog do Produto

Backlog 10		Create Sprint
	↑ BOOT-13 Eu como administrativo preciso criar a empresa na Junta Comercial	Administrativo 13
	↑ BOOT-6 Eu como CTO preciso registrar domínio rivendel.com.br	Infra 2
	↑ BOOT-7 Eu como CTO preciso criar conta Google Apps da Rivendel	Infra 2
	↑ BOOT-5 Eu como CTO preciso ter uma conta AWS para infra Rivendel	Infra 2
	↑ BOOT-8 Eu como CEO preciso ter um site e blog para comunicação institucional	Comercial 20
	↑ BOOT-10 Eu como CEO preciso fechar parcerias técnicas e comerciais	Comercial 13
	↑ BOOT-9 Eu como CEO preciso criar perfis da Rivendel nas redes sociais	Comercial 5
	↑ BOOT-11 Eu como comercial preciso ter um template de Proposta Comercial	Comercial 5
	↑ BOOT-12 Eu como comercial preciso ter um template de Contrato	Comercial 5
	↑ BOOT-14 Eu como administrativo preciso escolher um escritório	Administrativo 20

Backlog do Produto

- 
- **Novos requisitos** podem ser **adicionados** ao *backlog* do produto a **qualquer momento**;
 - com a aprovação do *Product Owner*;
 - e o consentimento da equipe de desenvolvimento.
 - Requisitos novos podem ser **implementados** apenas nas **iterações seguintes**.

Histórias de Usuário



- Cenários de uso baseados nas **experiências de usuários** do sistema;
 - implementados diretamente como uma **série de tarefas**.
- Maneira como os **requisitos** do produto são descritos.
- Cada história de usuário é **escrita** pelo **Product Owner** em uma ficha.
- O *Product Owner* **atribui** um **valor** à história de usuário;
 - baseando-se no valor de negócio global do requisito para cumprir as metas mais importantes de todos os envolvidos;
 - **prioridade**.

Histórias de Usuário

<<Título da História>>

Como um <<ator>>

Eu <<quero/preciso/desejo>> <<objetivo da história>>

Para <<razão da história ser necessária>>

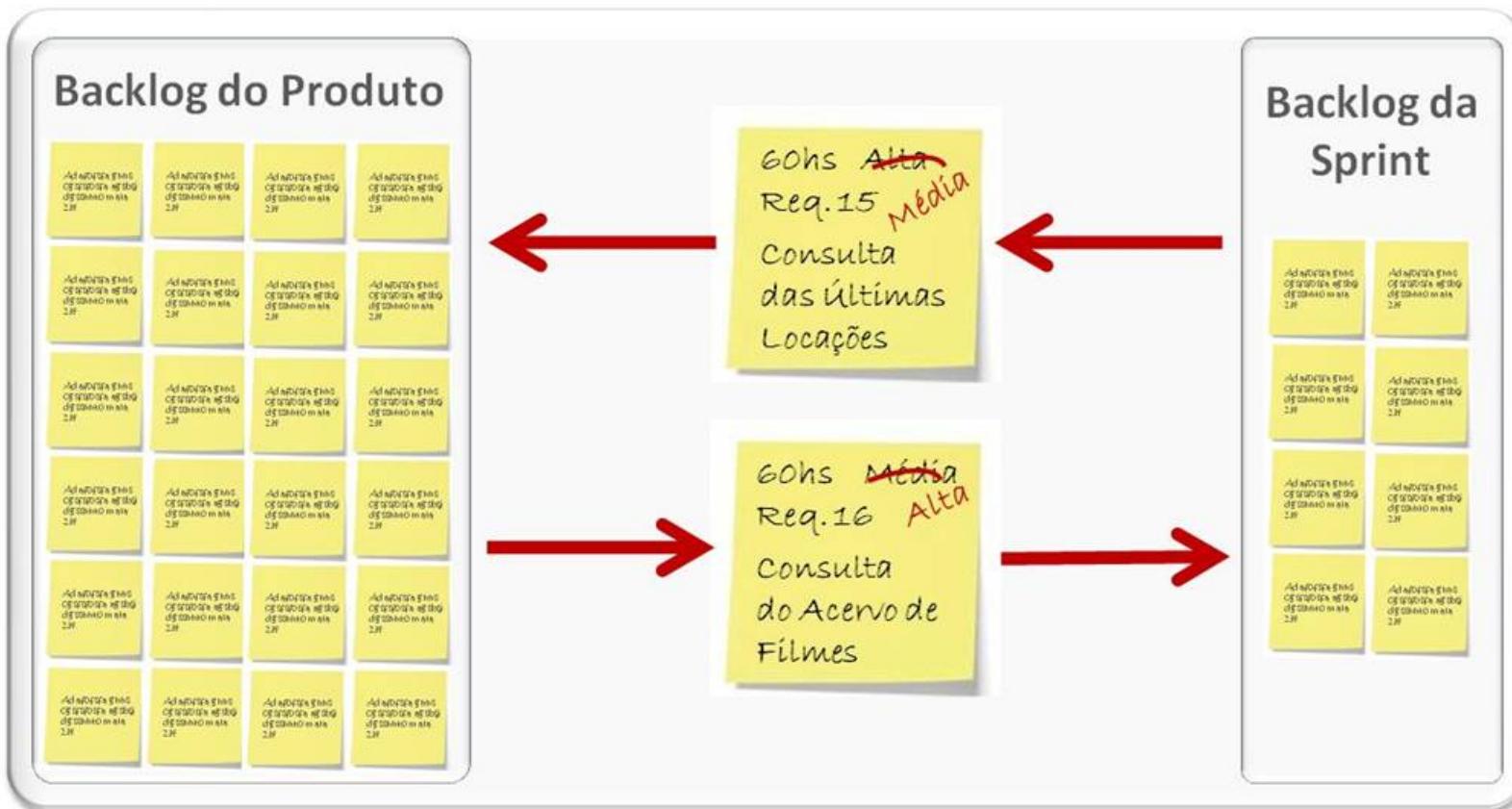
Formas de Pagamento

Como um Cliente,
Eu quero que sejam disponibilizadas
diversas formas de pagamento
Para pagar meu pedido.

Backlog da Sprint

- 
- ***Product Owner*** enuncia a **meta** de desenvolvimento para o incremento a ser completado na **próxima Sprint**.
 - *Scrum Master* e time de desenvolvimento selecionam os **itens** a serem **transferidos** para o ***backlog da Sprint***;
 - o time de desenvolvimento determina **o que pode ser entregue no incremento**;
 - **dentro** dos limites da janela de **tempo** alocada à ***Sprint***.

Backlog da Sprint



Incremento do Produto

- Ao **final da Sprint**, o time de desenvolvimento terá produzido um **incremento PRONTO no produto**;
 - valor para o cliente.



Sprints

- **Iterações semanais;**
 - ou mensais;
 - que contêm um grupo de **tarefas a serem concluídas** nesse **período**.
- As tarefas da *Sprint* estão indicadas no **backlog da Sprint**.
 - Cada **característica** é **planejada, implementada e testada** na mesma *Sprint*.
- Após o início da *Sprint*, nenhuma nova característica pode ser adicionada a seu *backlog*;
 - a menos que a *Sprint* seja cancelada ou reiniciada.

Tarefa



- **Time de desenvolvimento define o esforço** necessário para a **execução** de cada **tarefa**;
 - baseando-se em **experiências anteriores**.
 - Estimativa de esforço de cada tarefa é **mais realista**;
 - maior a chance de ser cumprida!
- **Cada membro do time**;
 - **escolhe** e fica responsável pela conclusão de suas **tarefas**.

Tarefa



- Baseado em **tarefas curtas**.
- **Prazo máximo** para conclusão de cada tarefa:
 - **uma semana**.
- Se a **estimativa de duração** de uma tarefa **extrapolar** o prazo de **uma semana**;
 - deve ser **subdividida** em **tarefas menores**.
- Seu ***status*** é atualizado;
 - no **quadro Kanban**.

Quadro Kanban

- Possui:
 - uma coluna para cada estágio dos requisitos:
 - **backlog do produto**;
 - **backlog da Sprint (planejado e em execução)**;
 - e **realizado**.
 - Pode apresentar também:
 - o gráfico **Burndown**;
 - uma região para **novos requisitos ou tarefas** identificados;
 - e **impedimentos**.



Quadro Kanban

- A evolução do projeto é facilmente monitorada.

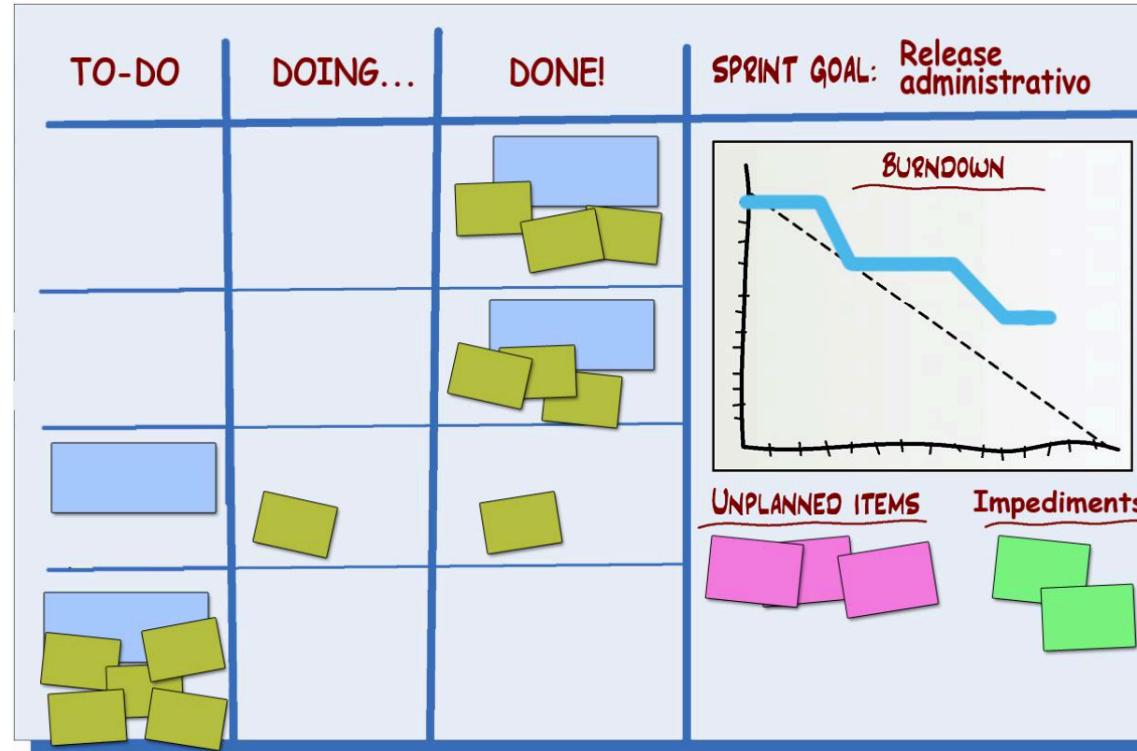
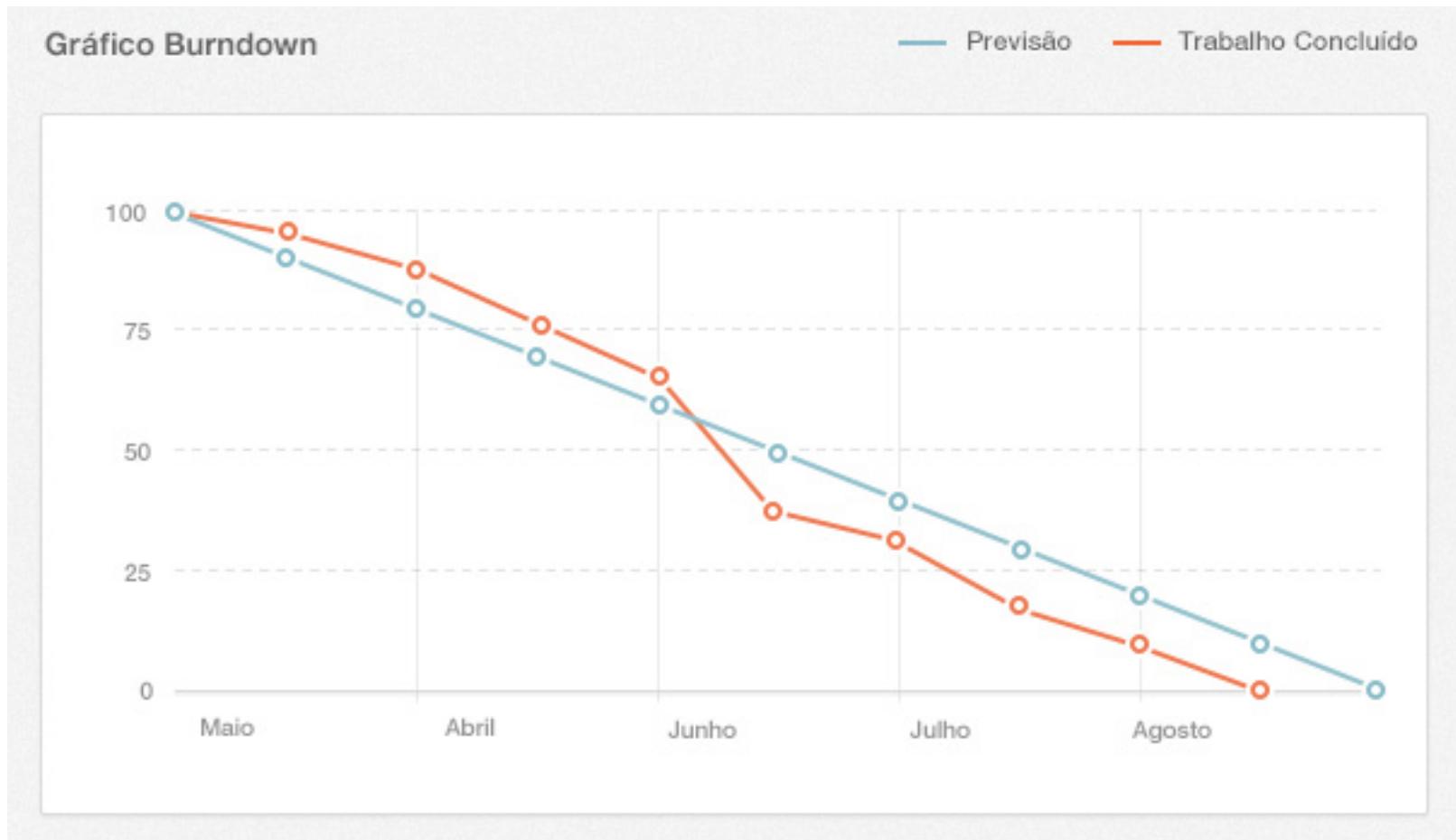


Gráfico *Burndown*



- Exibe o número de **tarefas ou horas restantes**;
 - ao longo do **tempo**:
 - **Sprint ou todo o projeto.**
- Acompanhamento do fluxo de desenvolvimento:
 - **previsto X concluído.**

Gráfico *Burndown*

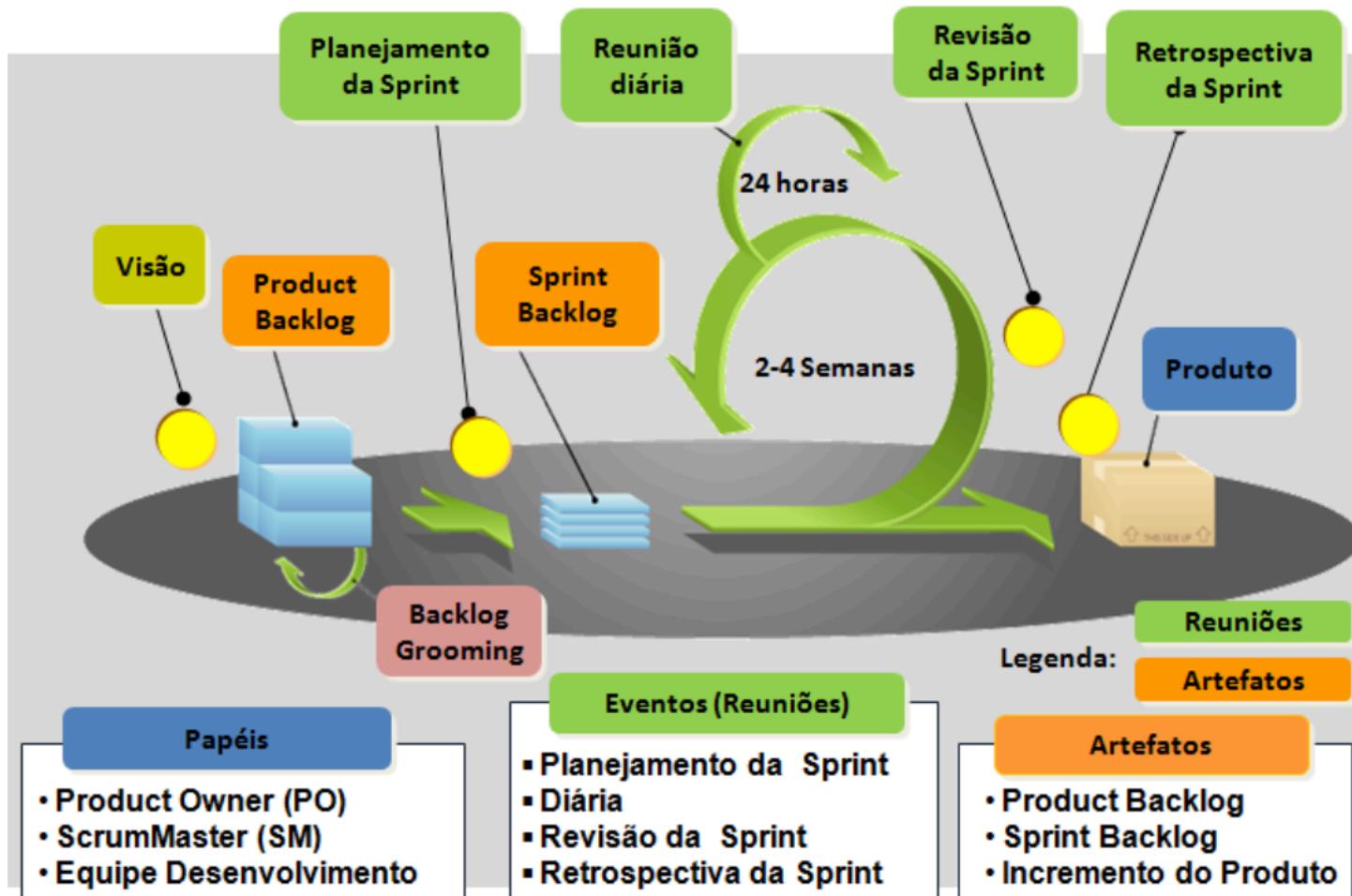


Quatro Cerimônias



- Reunião de planejamento da *Sprint*.
- Reuniões diárias.
- Revisão da *Sprint*.
- Retrospectiva da *Sprint*.

Cerimônias



Reunião de Planejamento da *Sprint*



- Duas perguntas:
 - ▣ **O que será entregue?**
 - ▣ **Como faremos para entregar?**
- **Define-se:**
 - ▣ **meta** da *Sprint*;
 - ▣ **histórias de usuário** selecionadas;
 - ▣ **tarefas técnicas** necessárias para transformar as histórias de usuário selecionadas em um incremento de produto.

Reunião Diária



- Envolve o Time de Desenvolvimento e o *Scrum Master*.
- **Conversas informais.**
- Duração máxima:
 - **15 minutos.**

Reunião Diária

- Três perguntas:
 - ▣ **O que fiz desde a última reunião diária?**
 - ▣ **O que pretendo fazer até a próxima?**
 - ▣ **Existe algo me impedindo de concluir alguma tarefa?**
- **Resulta em:**
 - ▣ **pequenos ajustes necessários;**
 - ▣ **atualização do quadro Kanban.**
- É responsabilidade do *Scrum Master* eliminar os obstáculos apresentados;
 - ▣ antes da próxima reunião diária, se possível.

Revisão da *Sprint*



- Ocorre no **final da *Sprint***,
 - quando a equipe de desenvolvimento decide que o **incremento** está **PRONTO**.
- **Tudo** o que foi **produzido na *Sprint*** é **apresentado**:
 - **demonstração do incremento de software** completado durante a *Sprint*.
- **Product Owner** valida o **incremento** produzido de acordo com a **meta da *Sprint***,
 - pode **aceitar** o incremento como completo **ou não**.

Revisão da *Sprint*



- **Foco:**
 - **aprimoramento do produto.**
- **Objetivos:**
 - **inspecionar** o que o **time produziu**;
 - coletar **impressões** para **ajustar** o **planejamento** da *Sprint* **seguinte**;
 - envolvidos fornecem **feedback** para possibilitar o planejamento da próxima *Sprint*.

Retrospectiva da *Sprint*



- Realizada ao **final da *Sprint*.**
- ***Scrum Master* lidera a reunião.**
- **Problemas encontrados na execução da *Sprint* são apresentados e discutidos.**
- *Scrum Master* encoraja a **equipe** a **melhorar suas práticas de desenvolvimento;**
 - para se tornar mais eficaz para a próxima *Sprint*.
- **Time** planeja formas de **melhorar a qualidade do produto;**
 - com a adaptação de sua definição de “PRONTO”.

Retrospectiva da *Sprint*



- Duas perguntas:
 - **O que deu certo na *Sprint*?**
 - **O que poderia melhorar?**
- **Objetivos:**
 - **aprimorar o processo:**
 - interação entre os membros, práticas e ferramentas utilizadas;
 - **identificar problemas e medidas para correção;**
 - com o que a equipe se compromete em melhorar na próxima *Sprint*.

Referências

- PRESSMAN, Roger. **Engenharia de *software*: uma abordagem profissional.** Capítulo 3.
- SOMMERVILLE, Ian. **Engenharia de *software*.** Capítulo 3.
- PRIKLADNICKI, Rafael, WILLI, Renato, e MILANI, Fabiano. **Métodos ágeis para desenvolvimento de *software*.** Capítulos 1 e 3.

PROCESSOS PRESCRITIVOS – UP E RUP

Eveline Alonso Veloso*

* Adaptado do material da Prof.^a Maria Augusta
Vieira Nelson

Processos Prescritivos

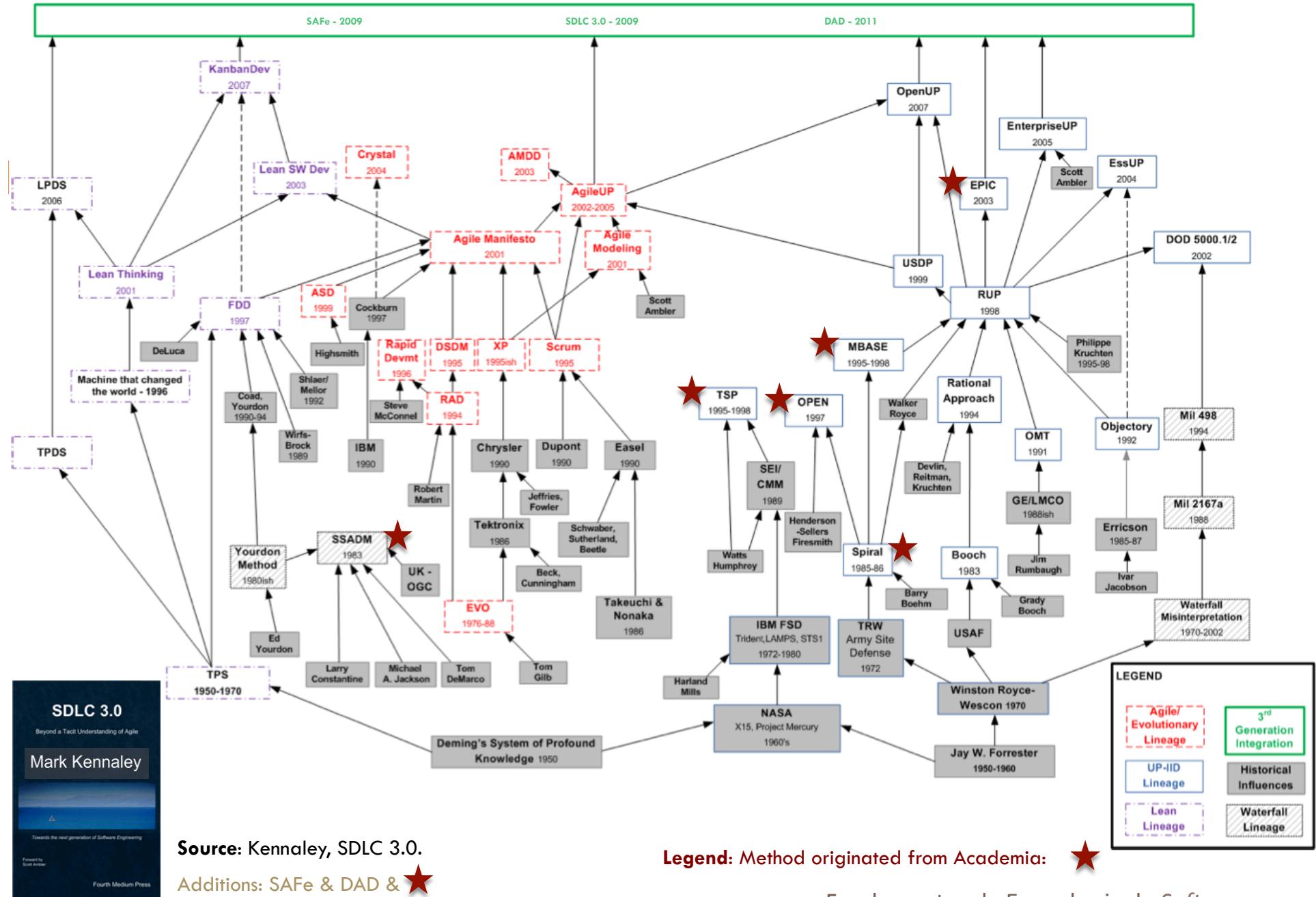


- Também chamados de **Processos Dirigidos a Planos.**
- **Todas as atividades** do processo **são planejadas com antecedência;**
- **progresso é medido em relação a esse plano.**

Processos Reais



- Em empresas de desenvolvimento de *software*, os **processos mais práticos e adequados** geralmente **incluem**
 - ▣ **elementos dos processos ágeis e prescritivos.**
- A **prática, o domínio da aplicação, a experiência da equipe**
 - ▣ **influenciam no processo** empregado e em **seu aprimoramento.**



Processo Unificado

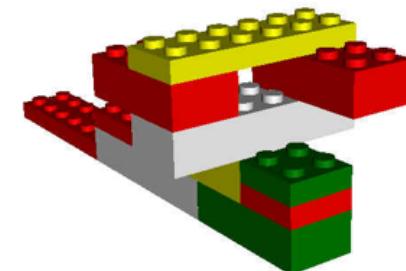
- 
- **Proposto por Booch, Jacobson e Rumbaugh;**
 - **proponentes da UML;**
 - notação de modelagem orientada a objetos.
 - **Descende de métodos anteriores;**
 - propostos pelos mesmos autores.
 - **Utiliza a UML como notação de uma série de modelos;**
 - que compõem os principais **resultados das atividades** do processo.

Características do Processo Unificado

- **Dirigido por casos de uso:**
 - **caso de uso:**
 - um pedaço de **funcionalidade** que fornece a um usuário um resultado de valor.
 - Todos os casos de uso juntos respondem à pergunta:
 - “o que é suposto que o sistema faça?”;
 - “para cada usuário?”
 - Casos de uso não são somente uma ferramenta para especificar as funcionalidades do sistema:
 - o **desenho ou projeto do sistema é dirigido por eles:**
 - **baseados** no modelo de **casos de uso**, os engenheiros de *software* criam uma **série de refinamentos derivados dele**.
 - Dirigido por casos de uso significa que o **desenvolvimento do software segue** um **fluxo** que **deriva dos casos de uso**.

Características do Processo Unificado

- **Centrado na arquitetura:**
 - **arquitetura do sistema:**
 - visão com as **características mais importantes** colocadas **em evidência**;
 - e com os **detalhes deixados de lado**.
 - descrição dos **componentes principais** do sistema **e de suas interfaces**.
 - Todo produto tem função e forma:
 - a **arquitetura** representa a **forma**;
 - enquanto os **casos de uso** representam as **funções**.



Características do Processo Unificado

- **Centrado na arquitetura:**
 - Deve-se trabalhar com uma **arquitetura inicial**;
 - e **evidenciar** essa arquitetura em **casos de uso arquiteturalmente relevantes**:
 - 5 a 10% do total;
 - **projetar, implementar e testar a arquitetura**:
 - protótipos arquiteturais.
 - À medida que os **casos de uso** **vão sendo implementados**;
 - a **arquitetura** pode ser **maturada**.
- **Vantagens:**
 - permite **refinar** as **estimativas de esforço**;
 - permite que os **membros** da **equipe** se **familiarizem com as tecnologias utilizadas**;
 - **ataca** alguns dos **principais riscos**.

Características do Processo Unificado



- **Iterativo e incremental:**
 - em **cada iteração**, os engenheiros de *software* apresentam os **casos de uso mais relevantes**;
 - **guiados pela arquitetura**;
 - de forma a **cada vez** estarem **mais próximos** do **sistema final**.

Características do Processo Unificado



- Modelo de **ciclo de vida**:
 - **em espiral**.
- **Cada ciclo**:
 - entrega uma **liberação do produto**.
- É **dividido em fases**.
- **Atividades técnicas**;
 - são **organizadas em fluxos**.

Fases x Fluxos

	Concepção	Elaboração	Construção	Transição
Requisitos				
Análise				
Projeto				
Implementação				
Testes				

RUP – *Rational Unified Process*



- **Produto comercial**
 - ▣ **baseado no Processo Unificado.**
- **Utiliza a UML;**
 - ▣ *Unified Modeling Language.*
- **Características-chave:**
 - ▣ **dirigido por casos de uso;**
 - ▣ **centrado na arquitetura;**
 - ▣ **iterativo e incremental.**

RUP – *Rational Unified Process*



- **Detalha e amplia o Processo Unificado;**
 - **complementando-o** com guias, atividades gerenciais, *templates*, ferramentas, etc.
- Não pode mais ser considerado um processo único;
 - mas uma **plataforma**;
 - que **permite construir** uma grande família de **processos personalizados**.

Princípios do RUP



- **Atacar os riscos principais;**
 - de forma **rápida e contínua.**
- Certificar-se de que o **produto** ou funcionalidade;
 - irá **agregar valor** ao negócio do cliente.
- **Acomodar mudanças** no projeto.

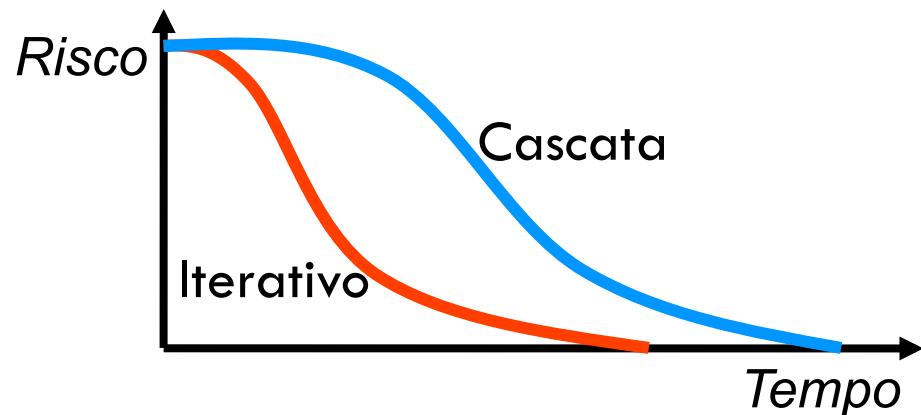
Princípios do RUP



- Definir uma **arquitetura executável**;
 - o mais cedo possível.
- Concentrar-se em **produzir**;
 - **software executável**.
- **Construir o sistema com componentes**.
- Trabalhar em **equipe**.
- **Verificação contínua da qualidade**.

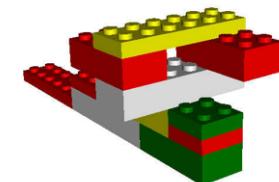
Atacar os Riscos

- Deve-se **atacar primeiro**;
 - ▣ os **problemas** que apresentam **maior risco para o sucesso do projeto**.
- Em **cada iteração**, deve-se listar os **principais riscos**;
 - ▣ e **agir** de forma a **mitigá-los**.



Uso de Componentes Arquiteturais

- **Software é construído a partir de componentes;**
 - **interconectados** por meio de **interfaces bem definidas**.
- **Componentes separam a interface da implementação.**
- **Vantagens:**
 - **encapsulamento**, ocultação de informações;
 - possibilita a **reutilização** de partes;
 - **reduz o impacto de mudanças**.



Dimensões do RUP

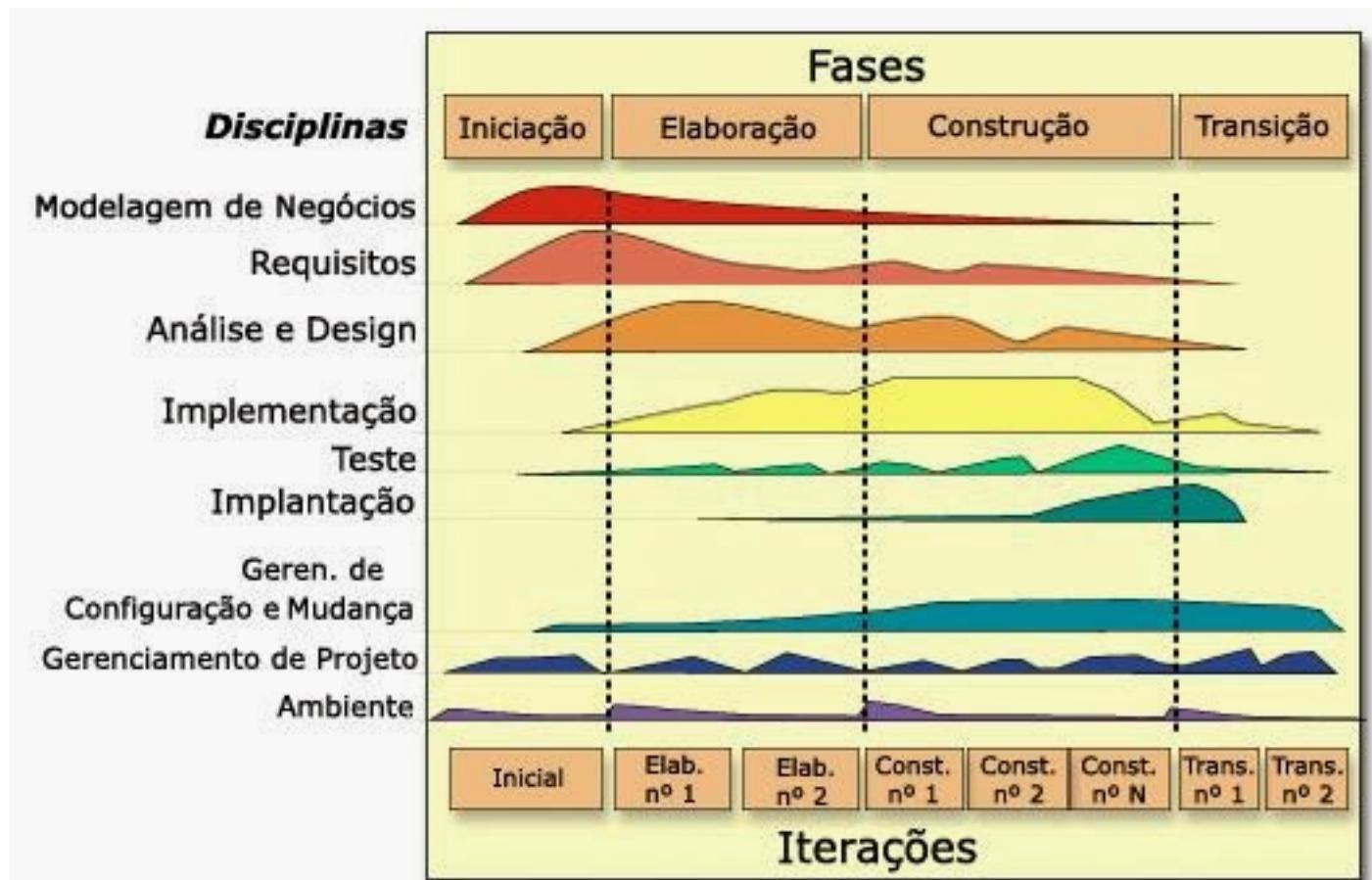


- RUP é organizado em **fases e disciplinas**.
- **Fases:**
 - distribuição das **atividades no tempo**.
 - **Cada fase tem uma ou mais iterações.**
- **Disciplinas:**
 - distribuição das **atividades por categoria**.
 - **Apresentam fluxos de trabalho.**

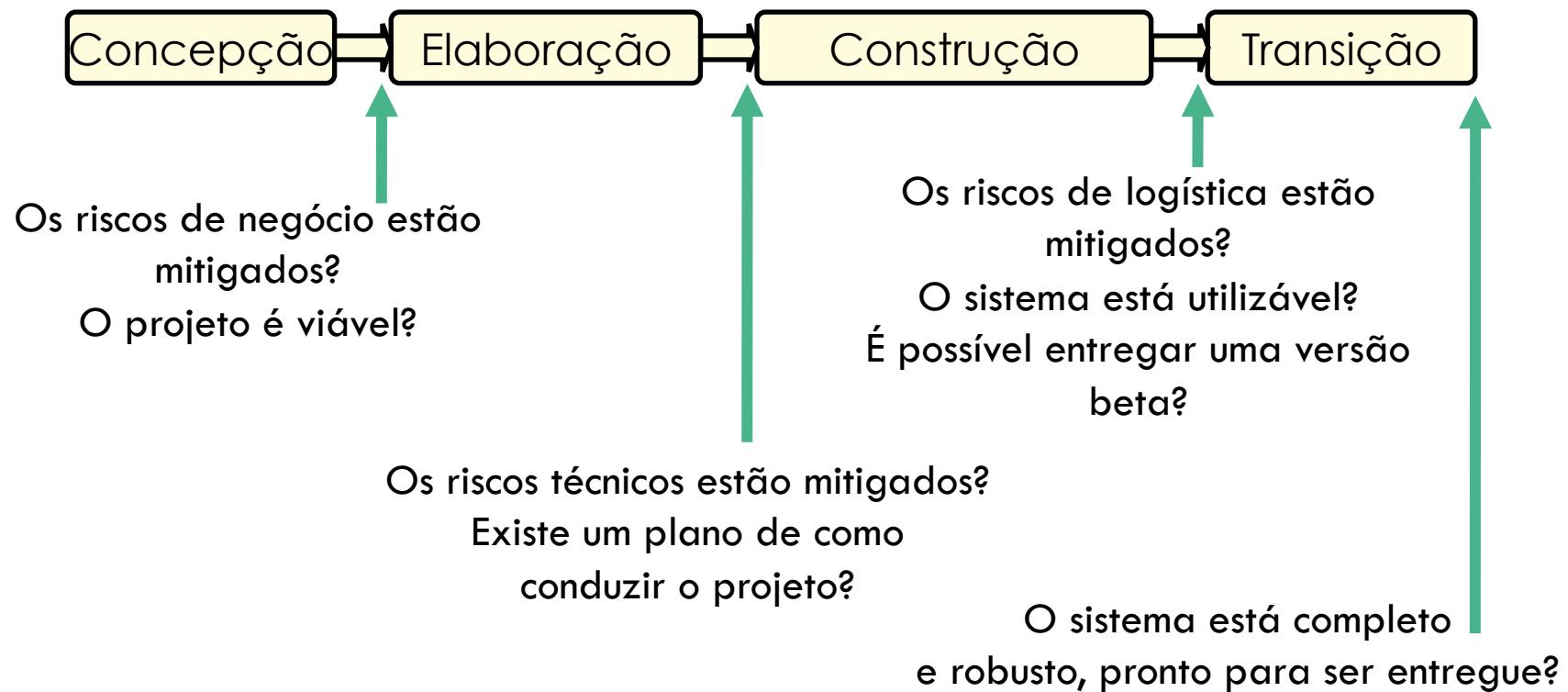
Iterativo e Incremental



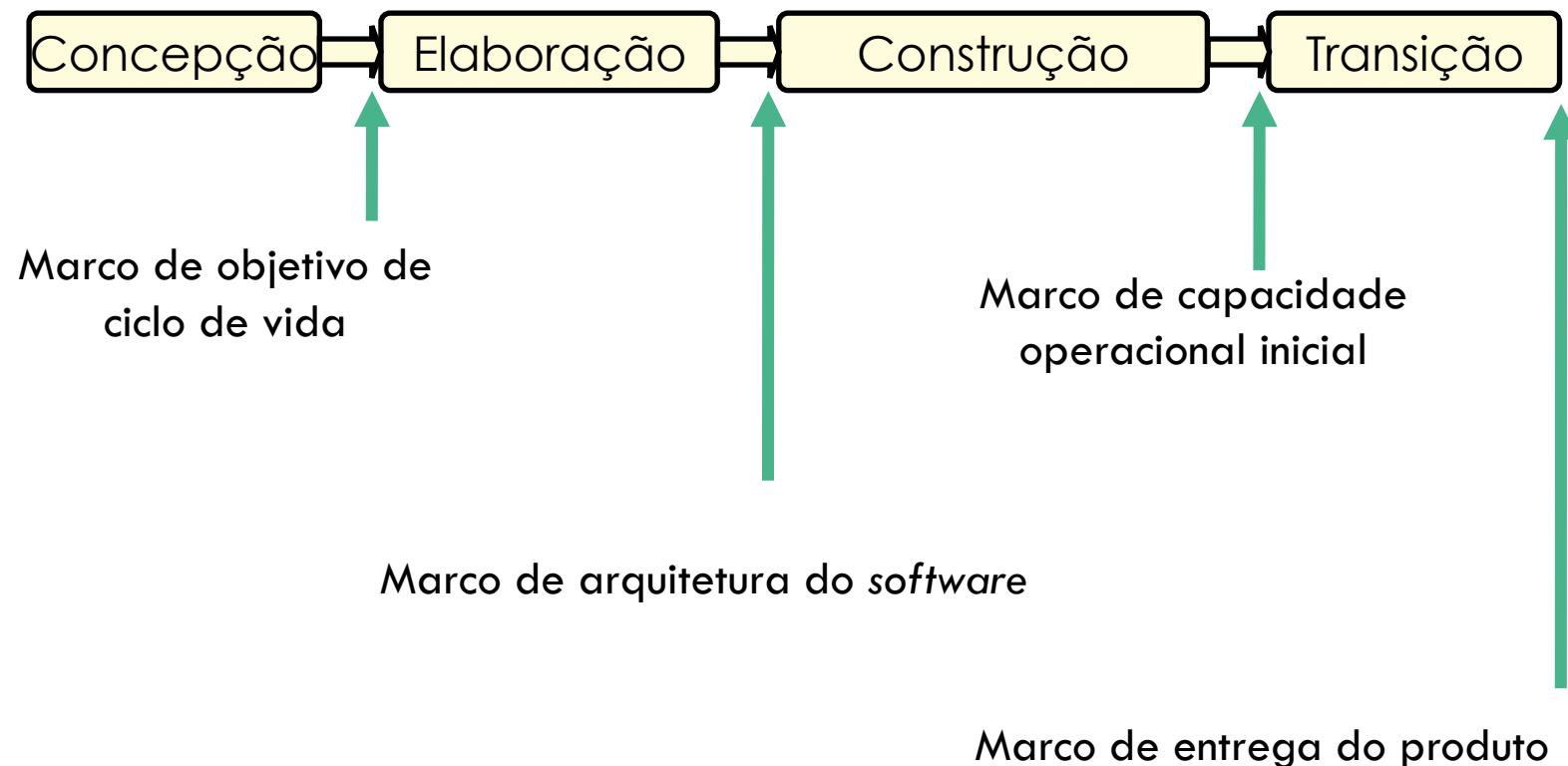
Gráfico das “Baleias”



Marcos Principais



Marcos Principais



Problemas com Processos Prescritivos



- O desenvolvimento de *software* em processos prescritivos:
 - é baseado no **paradigma comando e controle**;
 - são **mais atrativos para gerentes** de projetos;
 - do que para a maioria dos desenvolvedores;
 - **segmenta o desenvolvimento de *software* em especialistas**;
 - que **não possuem visão do todo**;
 - muitas vezes são **abandonados**;
 - em **situações críticas**.

Referências

- KROLL, Per; KRUCHTEN, Philippe. **The Rational Unified Process made easy: a practitioner's guide to the RUP.** 2003.
- PRESSMAN, Roger. **Engenharia de software: uma abordagem profissional.** Capítulo 2.
- SOMMERVILLE, Ian. **Engenharia de software.** Capítulo 2.
- Vários artigos sobre o RUP:
 - <http://philippe.kruchten.com/process/>