

# Engenharia de Software

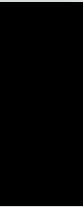
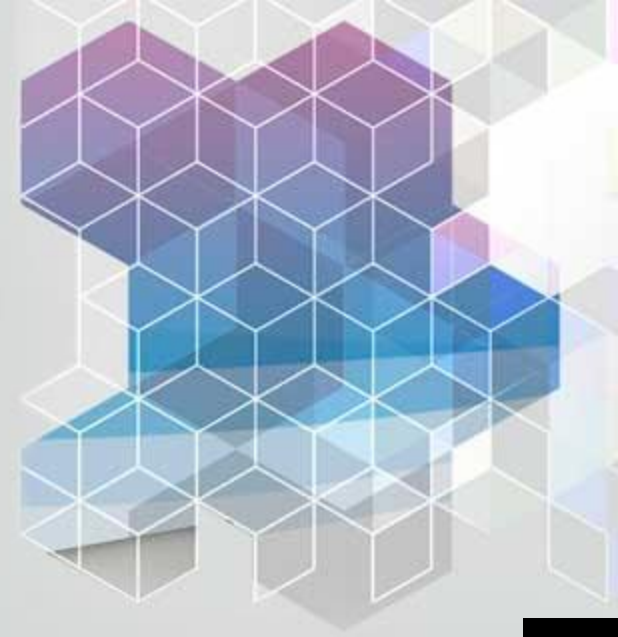
Curso Superior de Tecnologia em Desenvolvimento de  
Software Multiplataforma

Aula 05

Prof. Claudio Benossi

# 2. Unidade

## **Modelos de Processos de Software**





# Modelos de Processos de Software - Trabalho


Como já falamos em aula, existe vários modelos de Processo de Software, e cada **modelo** representa cada abordagem usada para a criação do **software**.

Nosso objetivo é realizar uma pesquisa para conhecer alguns deles:





# Modelos de Processos de Software - Trabalho

1. Modelo V
  2. DSDM (Dynamic Systems Development Method)
  3. TDD (Test Driven Development)
  4. Crystal
  5. FDD (Feature Driven Development)
  6. Open Source Software Development
  7. Rational Unified Process (RUP)
  8. Lean Development
- 




# Modelos de Processos de Software - Trabalho

- ✓ Histórico – origem, disseminação, popularidade, criadores, evolução
- ✓ Modelo em forma de diagrama
- ✓ Processos, etapas, sequenciamento, tarefas
- ✓ Benefícios e desvantagens
- ✓ Exemplos de projetos ou empresas que adotaram o modelo
- ✓ Infográfico em arquivo de imagem
- ✓ Apresentação em 24 de outubro (entre 10 e 25 minutos rigorosamente, nem menos, nem mais)
- ✓ Submissão dos entregáveis no dia 23 de outubro via Teams



# Manifesto Ágil - Princípios

1. Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado.
  2. Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.
  3. Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo.
  4. Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.
  5. Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho.
  6. O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face.
- 

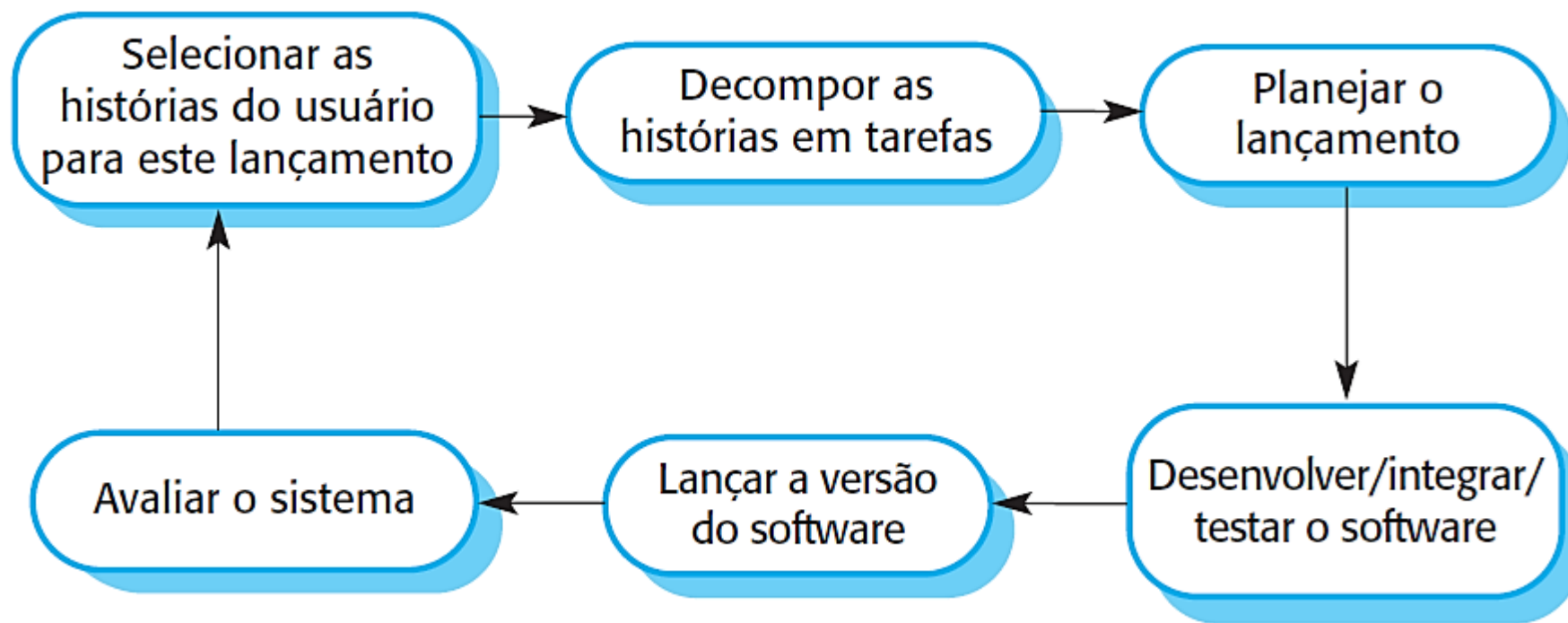


# Manifesto Ágil - Princípios

7. Software funcionando é a medida primária de progresso.
8. Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.
9. Contínua atenção à excelência técnica e bom design aumenta a agilidade.
10. Simplicidade--a arte de maximizar a quantidade de trabalho não realizado--é essencial.
11. As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis.
12. Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

# Técnicas de desenvolvimento ágil

- Ciclo de lançamento (release) da Programação Extrema:








# Técnicas de desenvolvimento ágil

## Práticas de Programação Externa:

Princípio ou Prática	Descrição
<b>Propriedade Coletiva</b>	<p>Os pares de desenvolvedores trabalham em todas áreas do sistema de modo que não se desenvolvem “<b>Ilhas de conhecimento</b>”, e todos os desenvolvedores assumem a responsabilidade por todo o código.</p> <p>Qualquer um pode mudar qualquer coisa.</p>





# Técnicas de desenvolvimento ágil

## Práticas de Programação Externa:


Princípio ou Prática	Descrição
<b>Integração Contínua</b>	<p>Assim que o trabalho em uma tarefa é concluído, ele é integrado ao sistema completo.</p> <p>Após qualquer integração desse tipo, todos os testes de unidade no sistema devem passar.</p>



# Técnicas de desenvolvimento ágil

## Práticas de Programação Externa:

Princípio ou Prática	Descrição
<b>Planejamento Incremental</b>	<p>Os requisitos são registrados em “<b>Cartões de História</b>”, e as histórias a serem incluídas em um lançamento são determinadas de acordo com o tempo disponível e com sua prioridade relativa.</p> <p>Os desenvolvedores decompõem estas histórias em “<b>Tarefas</b>” de desenvolvimentos.</p>





# Técnicas de desenvolvimento ágil

## Práticas de Programação Externa:

Princípio ou Prática	Descrição
<b>Representante do Cliente</b>	<p>Um representante do usuário final do sistema (<b>O Cliente</b>) deve estar disponível em tempo integral para o time de programação.</p> <p>Em um processo como esse, o cliente é um membro do time de desenvolvimento, sendo responsável por levar os requisitos do sistema ao time, visando sua implementação.</p>



# Técnicas de desenvolvimento ágil

## Práticas de Programação Externa:

Princípio ou Prática	Descrição
<b>Programação em Pares</b>	Os desenvolvedores trabalham em pares, conferindo o trabalho um do outro e oferecendo o apoio necessário para que o resultado seja sempre satisfatório.

# Técnicas de desenvolvimento ágil

## Práticas de Programação Externa:

Princípio ou Prática	Descrição
<b>Refatoração</b>  (A Refatoração é o processo de alterar um software de uma maneira que não mude o seu comportamento externo e ainda melhore a sua estrutura interna)	Todos os desenvolvedores devem refatorar o código continuamente logo que sejam encontrados possíveis melhorias para ele.  Isso mantém o código simples e de fácil manutenção.



# Técnicas de desenvolvimento ágil

## Práticas de Programação Externa:


Princípio ou Prática	Descrição
<b>Projeto (Design) Simples</b>	Deve ser feito o suficiente de projeto (design) para satisfazer os requisitos atuais, e nada mais.



# Técnicas de desenvolvimento ágil

## Práticas de Programação Externa:

Princípio ou Prática	Descrição
<b>Lançamento Pequenos</b>	<p>O mínimo conjunto útil de funcionamento que agregue valor ao negócio é desenvolvido em primeiro lugar.</p> <p>Os lançamentos do sistema são frequentes e crescem funcionalidade à primeira versão de maneira incremental.</p>







# Técnicas de desenvolvimento ágil

## Práticas de Programação Externa:

Princípio ou Prática	Descrição
<b>Risco Sustentável</b>	Grandes quantidades de horas extras não são consideradas aceitáveis, já que o efeito líquido muitas vezes é a diminuição da qualidade do código e da produtividade no médio prazo.




# Técnicas de desenvolvimento ágil

## Práticas de Programação Externa:

Princípio ou Prática	Descrição
<b>Desenvolvimento com Testes a Priori (<i>Test First</i>)</b>	Um framework automatizado de teste de unidade é utilizado para escrever os testes de um novo pedaço de funcionalidade antes que ela própria seja implementada.




# Refatoração

- Exemplos de refatoração incluem:
    1. a reorganização de uma hierarquia de classe para remover código duplicado,
    2. a organização e renomeação de atributos e métodos e
    3. a substituição de seções de código similares, com chamadas para métodos definidos em uma biblioteca.
  - A princípio, quando a refatoração faz parte do processo de desenvolvimento, o software sempre deve ser fácil de compreender e mudar quando são propostos novos requisitos. Na prática, nem sempre é o que acontece.
- 



# Desenvolvimento com testes *a priori* (*test-first*)

- As características-chave do teste em Programação Extrema são:
    1. desenvolvimento com testes *a priori* (*test-first*);
    2. desenvolvimento de teste incremental a partir de cenários;
    3. envolvimento do usuário no desenvolvimento e validação dos testes e;
    4. o uso de frameworks de teste automatizados.
  - Escrever os testes define implicitamente uma interface e uma especificação do comportamento da funcionalidade que está sendo desenvolvida.
- 



# Programação em Pares

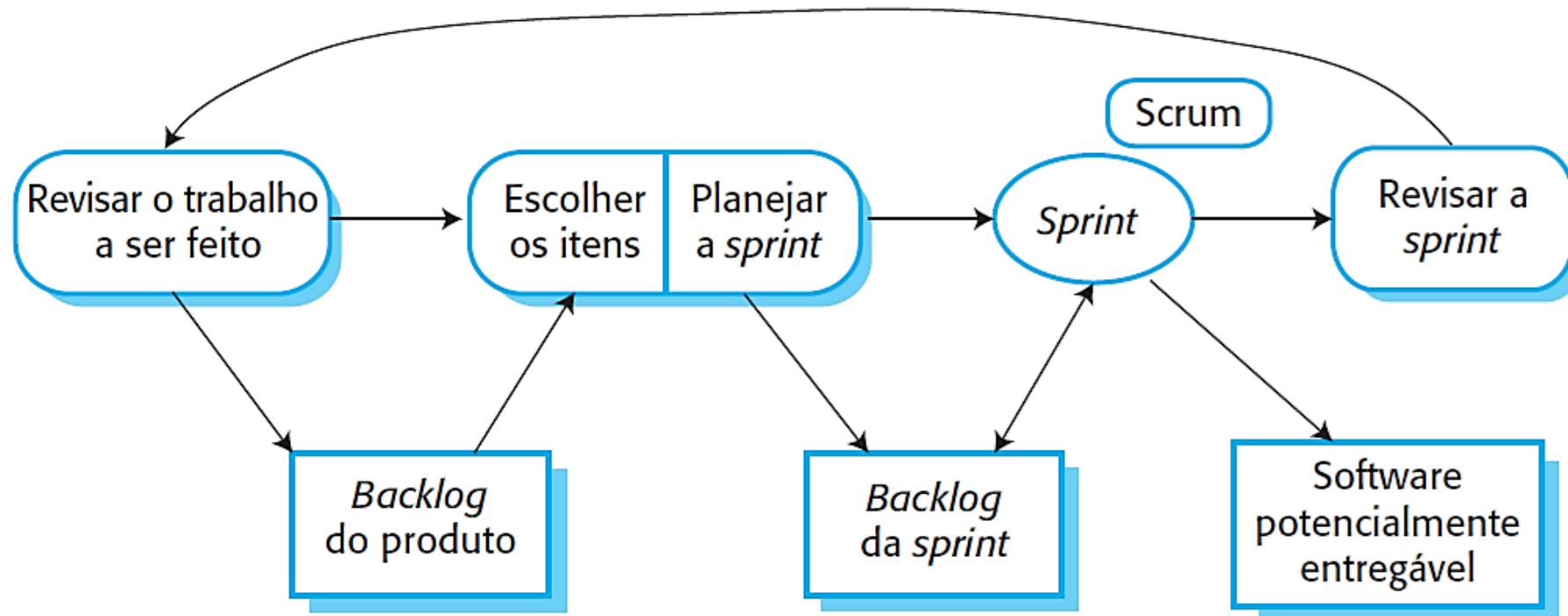
- Os pares são criados dinamicamente para que todos os membros do time trabalhem uns com os outros durante o processo de desenvolvimento.
- A programação em pares tem uma série de vantagens:
  1. Apoia a ideia de propriedade e responsabilidade coletivas pelo sistema.
  2. Ela age como um processo de revisão informal, já que cada linha de código é examinada por ao menos duas pessoas.
  3. Incentiva a refatoração para melhorar a estrutura do software.

# Gerenciamento de Projetos

## ➤ Terminologia do Scrum:

Termo do Scrum	Definição
Time de desenvolvimento	Um grupo auto-organizado de desenvolvedores de software que não deve ter mais de sete pessoas. Elas são responsáveis por desenvolver o software e outros documentos essenciais do projeto.
Incremento de produto potencialmente entregável	O incremento de software entregue a partir de uma <i>sprint</i> . A ideia é que ele seja 'potencialmente entregável', significando que está em estado acabado e não é necessário um trabalho adicional, como testes, para incorporá-lo ao produto final. Na prática, contudo, nem sempre isso é realizável.
<i>Backlog</i> do produto	É uma lista de itens 'a fazer' que o time Scrum deve cumprir. Podem ser definições de características e requisitos do software, histórias do usuário ou descrições de tarefas suplementares que são necessárias, como a definição da arquitetura ou a documentação do usuário.
<i>Product Owner</i>	Um indivíduo (ou possivelmente um pequeno grupo) cujo dever é identificar características ou requisitos do produto, priorizá-los para desenvolvimento e revisar continuamente o <i>backlog</i> do produto para garantir que o projeto continue a satisfazer as necessidades críticas do negócio. O <i>Product Owner</i> , também chamado de dono do produto, pode ser um cliente, mas também poderia ser um gerente de produto em uma empresa de software ou um representante de um <i>stakeholder</i> .
Scrum	Uma reunião diária do time Scrum que examina o progresso e prioriza o trabalho a ser feito naquele dia. Em condições ideais, deve ser uma reunião presencial que inclua todo o time.
<i>Scrum Master</i>	O <i>Scrum Master</i> é responsável por assegurar que o processo Scrum seja seguido e guiar o time no uso eficaz do Scrum. Essa pessoa é responsável pela interação com o resto da empresa e por garantir que o time Scrum não seja desviado por interferências externas. Os desenvolvedores Scrum são inflexíveis quanto ao <i>Scrum Master</i> não ser considerado um gerente de projeto. No entanto, outros nem sempre podem ver a diferença facilmente.
<i>Sprint</i>	Uma iteração de desenvolvimento. As <i>sprints</i> normalmente duram de 2 a 4 semanas.
Velocidade	Uma estimativa de quanto esforço de <i>backlog</i> do produto um time pode cobrir em uma única <i>sprint</i> . Compreender a velocidade de um time ajuda a estimar o que pode ser coberto por uma <i>sprint</i> e constitui a base para medir a melhoria do desempenho.

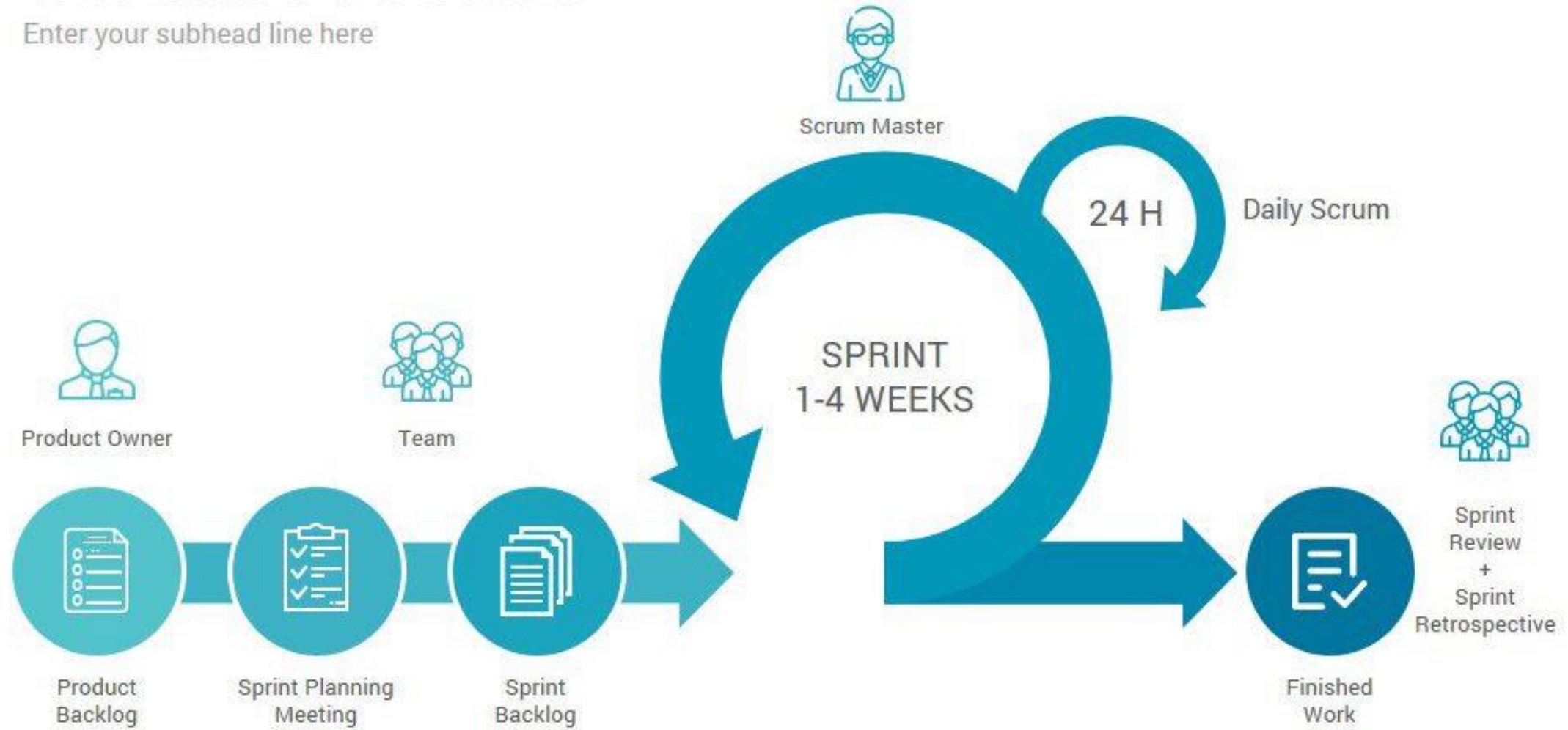
# Gerenciamento de Projetos





# Scrum Process

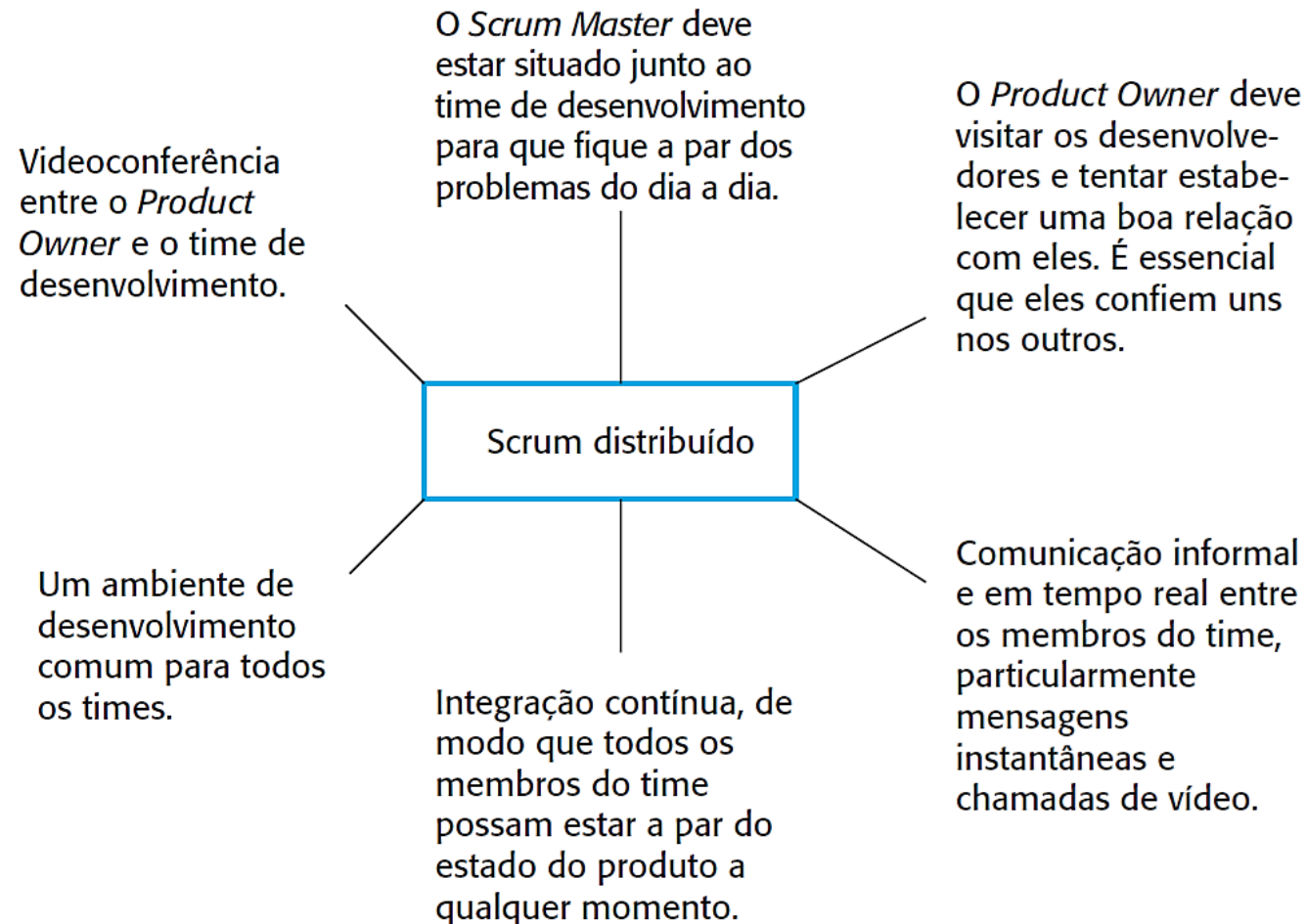
Enter your subhead line here





# Gerenciamento ágil de projetos

## ➤ Scrum distribuído:



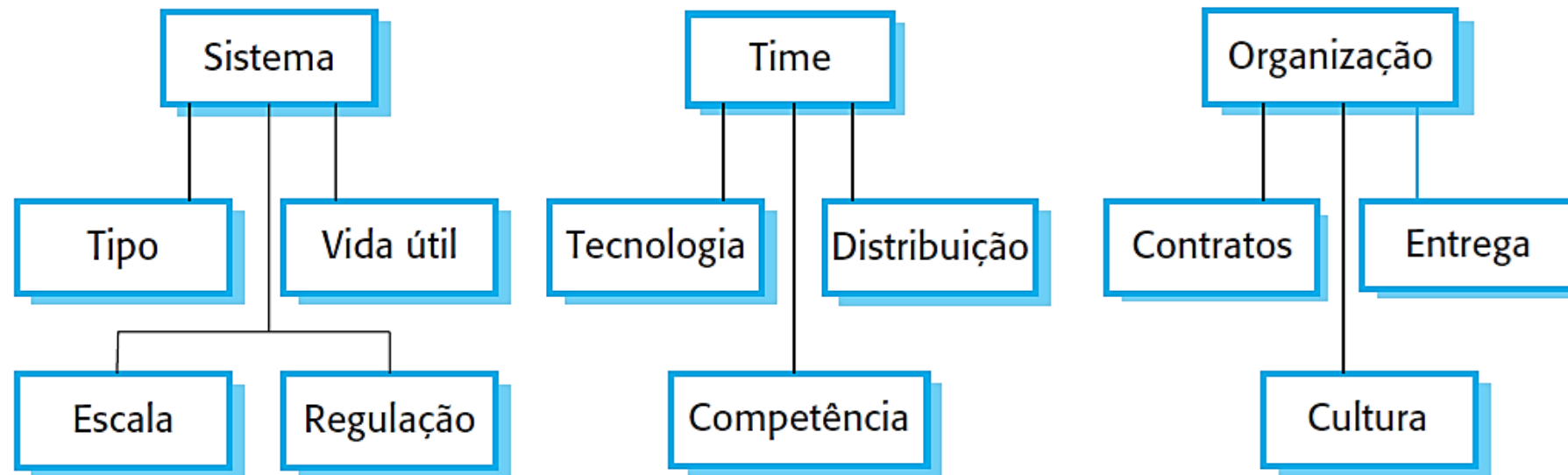


# Problemas práticos dos métodos ágeis

- Nos sistemas grandes e de longa vida útil desenvolvidos por uma empresa de software para um cliente externo, o uso de uma abordagem ágil apresenta uma série de problemas:
  1. A informalidade do desenvolvimento ágil é incompatível com a abordagem legal normalmente utilizada para a definição dos contratos em grandes empresas.
  2. Os métodos ágeis são mais adequados para o desenvolvimento de software novo em vez de manutenção de software.
  3. Esses métodos são concebidos para pequenas empresas que compartilham uma localização geográfica.

# Método ágil e método dirigido por plano

- Fatores que influenciam a escolha do desenvolvimento dirigido por plano ou ágil:



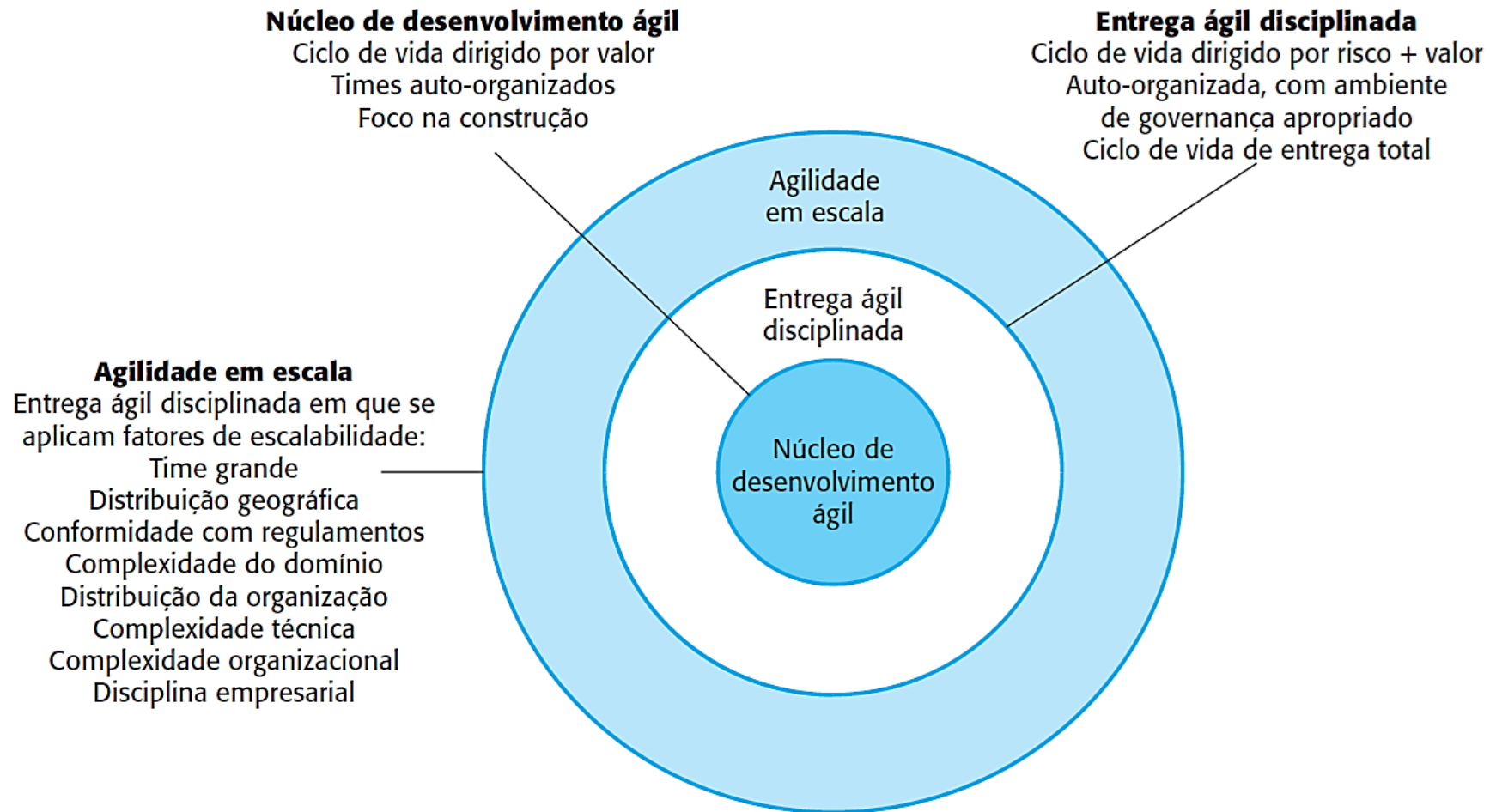


# Método ágil e método dirigido por plano

➤ Alguns dos pontos-chave são:

1. Qual é o tamanho do sistema que está sendo desenvolvido?
2. Que tipo de sistema está sendo desenvolvido?
3. Qual é a vida útil prevista para o sistema?
4. O sistema está sujeito a controle externo?
5. Qual é o nível de competência dos projetistas e programadores do time de desenvolvimento?
6. Como está organizado o time de desenvolvimento?

# Métodos ágeis para sistemas grandes





# Métodos ágeis nas organizações

- Naturalmente, empresas maiores experimentaram métodos ágeis em projetos específicos, mas é mais difícil para elas introduzi-los na organização.
- Essa dificuldade acontece por uma série de razões:
  1. Os gerentes de projeto podem relutar em aceitar o risco de uma nova abordagem, já que não sabem como isso vai afetar seus projetos particulares.
  2. As grandes organizações frequentemente têm procedimentos e padrões de qualidade que todos os projetos devem seguir e, devido à sua natureza burocrática, esses procedimentos e padrões tendem a ser incompatíveis com os métodos ágeis.



## Métodos ágeis nas organizações

3. Os métodos ágeis parecem funcionar melhor quando os membros do time têm um nível de habilidade relativamente elevado.
  4. Pode haver uma resistência cultural aos métodos ágeis, especialmente nas organizações que tenham um longo histórico de uso de processos convencionais de engenharia de sistemas.
- Introduzir e manter o uso dos métodos ágeis em uma grande organização é um processo de mudança de cultura.
  - Leva muito tempo para que esta seja implementada e isso requer uma troca na gerência antes de ser feito.



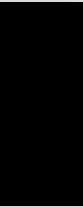
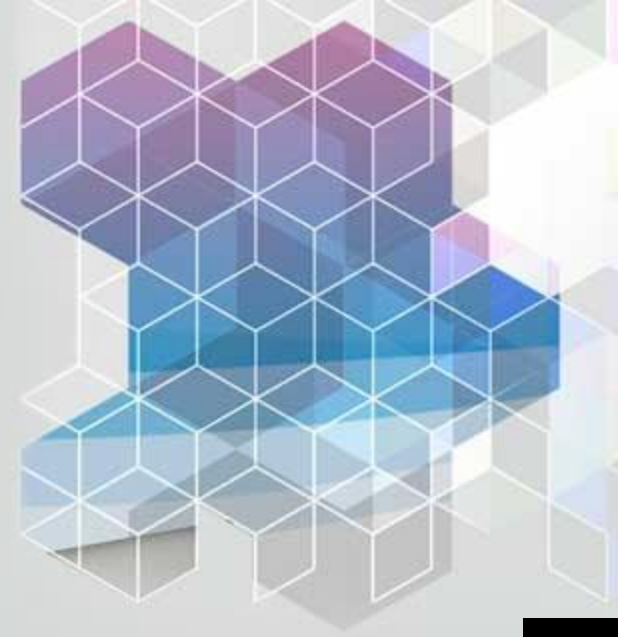
# Leitura recomendada

- Modelos de Processos de Software:
  - SOMMERVILLE, Ian. **Engenharia de Software**. 10. ed. São Paulo: Pearson Education do Brasil, 2019. **Capítulo 3: Desenvolvimento ágil de software**.



# 2. Unidade

## **Engenharia de Requisitos**



# Panorama

Dificuldades para  
extrair os  
requisitos dos  
clientes

Registros de  
requisitos de  
forma  
desorganizada

Quando as  
mudanças tomam  
o controle do  
projeto

Falta de base  
sólida para a  
construção de  
software



# Requisitos

- Antes de iniciar qualquer trabalho técnico, é uma boa ideia criar um conjunto de requisitos para todas as tarefas de engenharia
- Quem realiza?
  - Engenheiros de software
  - Analistas de sistemas
  - Scrum Master
  - Product Owner
  - E outras pessoas envolvidas no projeto de software




# Engenharia de Requisitos

ER é o conjunto de tarefas e técnicas que conduzem a um **entendimento dos requisitos**

Do ponto de vista do processo de software, a engenharia de requisitos é uma ação que se inicia durante a atividade de comunicação e continua na etapa de modelagem.

**ER deve ser adaptada às necessidades do processo, do projeto, do produto e das pessoas** que estão realizando o trabalho.



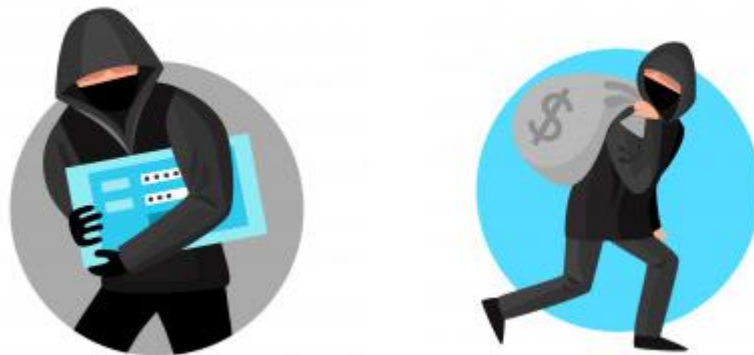
# Função da ER

- A engenharia de requisitos constrói uma ponte entre o projeto e a construção
- Definição da necessidade do negócio,
- Descrição de cenários de usuários,
- Determinação das funções e recursos
- Identificação das restrições de projeto.



# ER na visão contemporânea

A computação onipresente permite que a tecnologia computacional seja integrada a muitos objetos do cotidiano.



Quando esses objetos são interligados em rede, permitem a criação de perfis de usuário mais completos, com as respectivas preocupações com a privacidade e a segurança.

# Tarefas da Engenharia de Requisitos

- ✓ Concepção
- ✓ Levantamento
- ✓ Elaboração
- ✓ Negociação
- ✓ Especificação
- ✓ Validação
- ✓ Gestão de requisitos

“As sementes das principais catástrofes de software normalmente são semeadas nos três primeiros meses do projeto.”

Caper Jones



# Tarefas da Engenharia de Requisitos

- ✓ Concepção
- ✓ Levantamento
- ✓ Elaboração
- ✓ Negociação
- ✓ Especificação
- ✓ Validação
- ✓ Gestão de requisitos

Ponto de partida do projeto de software;

Pode ser um Termo de Abertura de Projeto (TAP), uma conversa informal, a identificação de um problema a ser resolvido;

Deve ser estabelecido um entendimento básico do problema, o que as pessoas querem que seja resolvido, e quem são essas pessoas (stakeholders).



# Tarefas da Engenharia de Requisitos

- ✓ Concepção
- ✓ Levantamento
- ✓ Elaboração
- ✓ Negociação
- ✓ Especificação
- ✓ Validação
- ✓ Gestão de requisitos

Geralmente se pergunta ao cliente, aos usuários e aos demais envolvidos:

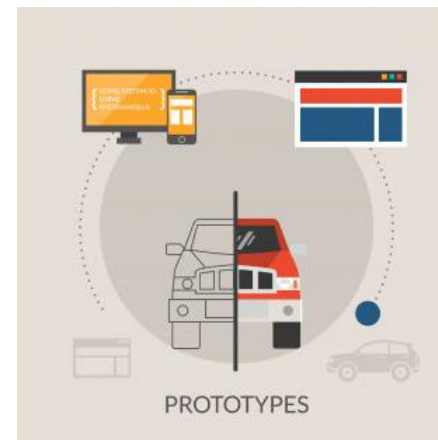
- Quais são os objetivos para o sistema ou produto?
- O que deve ser obtido?
- Como o sistema ou produto atende às necessidades da empresa?
- Como o sistema ou produto deve ser utilizado no dia a dia?

É preciso identificar qual a meta da organização em relação ao futuro uso do sistema a ser desenvolvido.

# Tarefas da Engenharia de Requisitos

- ✓ Concepção
- ✓ Levantamento
- ✓ Elaboração
- ✓ Negociação
- ✓ Especificação
- ✓ Validação
- ✓ Gestão de requisitos

Chega a hora de refinar todas aquelas informações levantadas junto ao cliente. Descrição de como os usuários vão interagir com o software, identificação das entidades e atributos que serão manipulados pelos usuários. Produção de diagramas suplementares.



# Tarefas da Engenharia de Requisitos

- ✓ Concepção
- ✓ Levantamento
- ✓ Elaboração
- ✓ Negociação
- ✓ Especificação
- ✓ Validação
- ✓ Gestão de requisitos

Não é raro clientes e usuários pedirem mais do que é possível, considerando os recursos limitados do projeto.

Também é relativamente comum diferentes clientes ou usuários proporem necessidades conflitantes, argumentando que o que se pede é essencial.

É preciso conciliar esses conflitos por meio de um processo de negociação.

Usuários e clientes devem participar da priorização, avaliação de custos e riscos, os requisitos são alterados ou eliminados visando a satisfação de todos.



# Tarefas da Engenharia de Requisitos

- ✓ Concepção
- ✓ Levantamento
- ✓ Elaboração
- ✓ Negociação
- ✓ Especificação
- ✓ Validação
- ✓ Gestão de requisitos

A formalidade e o formato dependem do cliente e do projeto de software.

O mais comum é Documento de Especificação de Requisitos onde constam:

- Finalidade do produto de software
  - Ambiente operacional
  - Restrições do projeto
  - Características do sistema
  - Requisitos funcionais e não funcionais
- Entre outros

# Tarefas da Engenharia de Requisitos

- ✓ Concepção
- ✓ Levantamento
- ✓ Elaboração
- ✓ Negociação
- ✓ Especificação
- ✓ Validação
- ✓ Gestão de requisitos

Na etapa de validação o documento de especificação de requisitos é avaliado para certificar que tenham sido declarados de forma não ambígua; que as inconsistências, omissões e erros tenham sido detectados e corrigidos; e que os artefatos estejam de acordo com os padrões estabelecidos para o processo, projeto e produto.

# Tarefas da ER - Validação



## *Lista de controle para validação de requisitos*

Muitas vezes é útil examinar cada requisito em relação a um conjunto de perguntas contidas em uma lista de controle. A seguir, um pequeno subconjunto do que poderia ser aplicado:

- Os requisitos estão expressos de forma clara? Eles podem ser mal interpretados?
- A fonte (por exemplo, uma pessoa, uma regulamentação, um documento) do requisito foi identificada? A declaração final do requisito foi examinada pela fonte original ou com ela?
- O requisito está limitado em termos quantitativos?
- Quais outros requisitos se relacionam a este requisito? Eles estão claramente indicados por meio de uma matriz de referência cruzada ou algum outro mecanismo?





# Tarefas da ER - Validação

- O requisito viola quaisquer restrições do domínio do sistema?
- O requisito pode ser testado? Em caso positivo, podemos especificar testes (algumas vezes denominados critérios de validação) para verificar o requisito?
- O requisito pode ser rastreado por algum modelo de sistema que tenha sido criado?
- O requisito pode ser rastreado pelos objetivos globais do sistema/produto?
- A especificação está estruturada de forma que leve ao fácil entendimento, referência e tradução em artefatos mais técnicos?
- Criou-se um índice para a especificação?
- Os requisitos associados ao desempenho, ao comportamento e às características operacionais foram declarados de maneira clara? Quais requisitos parecem estar implícitos?

# Tarefas da Engenharia de Requisitos

- ✓ Concepção
- ✓ Levantamento
- ✓ Elaboração
- ✓ Negociação
- ✓ Especificação
- ✓ Validação
- ✓ Gestão de requisitos

O desejo de mudar os requisitos persiste ao longo da vida de um sistema.

A gestão de requisitos é um conjunto de atividades que ajuda a equipe de projeto a identificar, controlar e acompanhar as necessidades e suas mudanças à medida que o projeto prossegue.





# Leitura recomendada

- PRESSMAN, Roger; MAXIM, Bruce. **Engenharia de Software**. 8. ed. Porto Alegre: AMGH, 2019. **Capítulo 8: Entendendo os requisitos**.

“Nada na vida deve ser  
temido, somente  
compreendido.  
Agora é hora de  
compreender mais  
para temer menos”



**Marié Curié**

# Obrigado!

**Se precisar ...**

Prof. Claudio Benossi

**[Claudio.benossi@fatec.sp.gov.br](mailto:Claudio.benossi@fatec.sp.gov.br)**

