Desenvolvimento ágil de software

ELTON MORAIS

Objetivos

APRESENTAR OS MÉTODOS DE DESENVOLVIMENTO ÁGIL DE SOFTWARE.

Objetivos

Compreender a lógica dos métodos de desenvolvimento ágil de software, o manifestado ágil e as diferenças entre o desenvolvimento ágil e o dirigido por plano;

Conhecer práticas importantes do desenvolvimento ágil, como as histórias do usuário, a refatoração, a programação em pares e o desenvolvimento com testes a *priori (test-first);*

Apresentar desde a abordagem Scrum até o gerenciamento ágil de projetos;

Compreender as questões de escalabilidade dos métodos de desenvolvimento ágil e a combinação das abordagens ágeis com as dirigidas por plano no desenvolvimento de grandes sistemas;

Introdução

Introdução

As empresas atuam em um ambiente global em **rápida mudança**;

As mudanças econômicas, novas oportunidades, surgimento de novos produtos e serviços;

O **desenvolvimento rápido** são requisitos os requisitos mais importantes da maioria dos sistemas de negócios;

Nesse ambiente é praticamente impossível elencar um **conjunto de requisitos** completamente, como no processo de **desenvolvimento dirigido por plano**;

O desenvolvimento mais rápido de software decolou nos anos 1990, com o surgimento dos **métodos ágeis**;

Introdução

Métodos ágeis, como:

- Programação Extrema (*Extreme Programmiing*, ou XP);
- Scrum; e
- DSDM (*Dynamic Systems Development Method*)

Principais características:

- Os processos de especificação, projeto e implementação são intercalados;
- O sistema é desenvolvido em uma série de incrementos;
- Utilização de ferramentas diversas no processo de desenvolvimento;

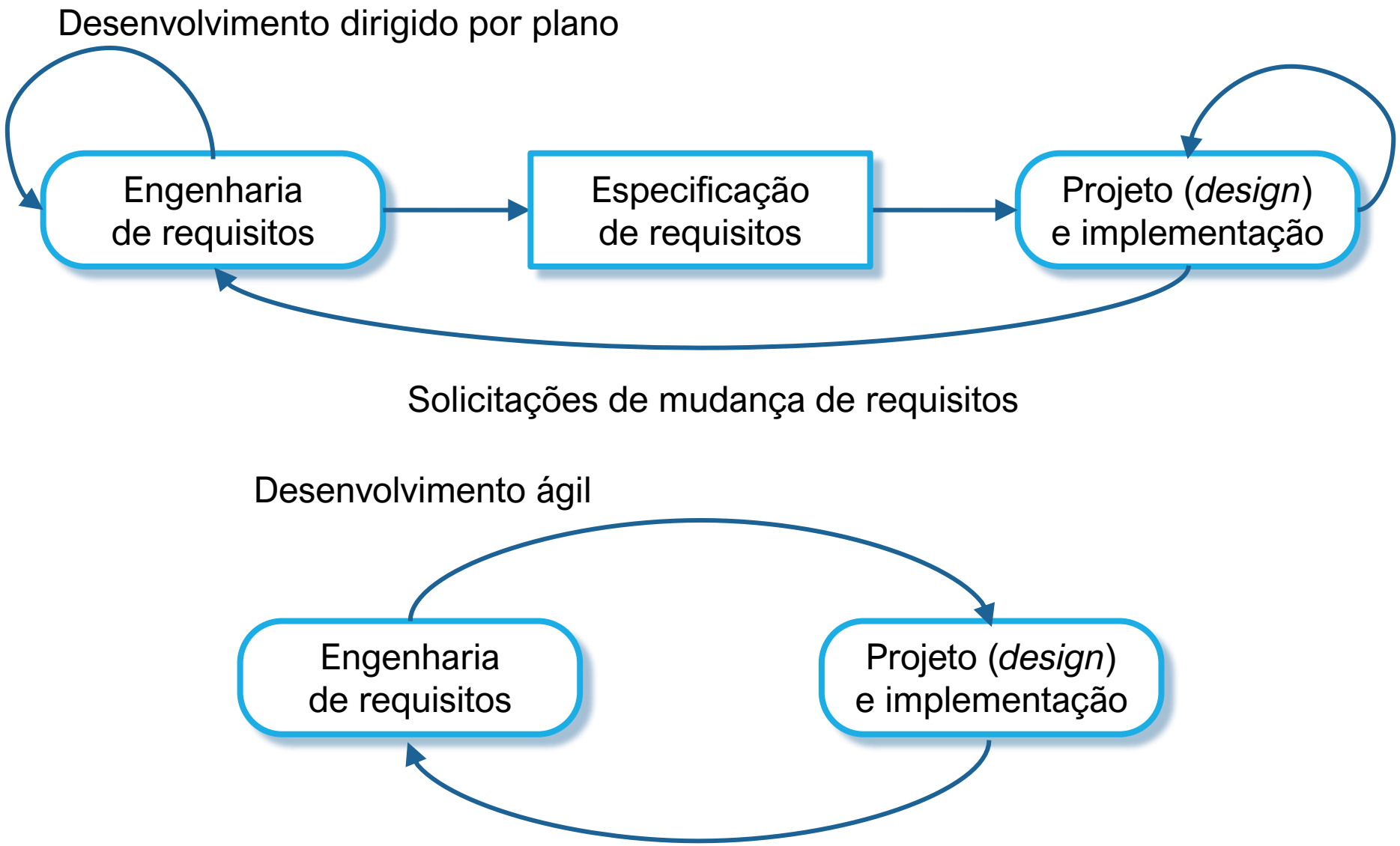
Introdução

Na abordagem ágil, o **projeto** e a **implementação** são consideradas as **atividades centrais** no processo;

- As iterações ocorrem ao longo das atividades, ou seja, requisitos e o projeto são desenvolvidos juntos, e não separadamente;

Na abordagem dirigida por plano, as **etapas são diferentes** no processo, porém com resultados **associados a cada uma delas**;

- As iterações ocorrem dentro das atividades, com documentos formais sendo utilizados como meio de comunicação entre as etapas do processo;



Desenvolvimento dirigido por plano e desenvolvimento ágil.

Métodos Ágeis

Métodos Ágeis

A abordagem dirigida por plano envolvem **sobrecargas** no planejamento, no desenvolvimento e na documentação;

Essa sobrecarga se justifica quando há vários times de desenvolvimento, quando o sistema é crítico e quando muitas pessoas diferentes estarão envolvidas na manutenção do software ao longo de sua vida útil;

Entretanto, essa abordagem pesada, quando aplicada a sistemas de negócio de pequeno ou médio porte, acabada dominando todo o processo de desenvolvimento;

Assim surgiram os **métodos ágeis**;

A filosofia dos métodos ágeis está refletida no **manifesto ágil**;

Métodos Ágeis

“ Estamos descobrindo maneiras melhores de desenvolver software, fazendo-o nós mesmo e ajudando outros a fazerem o mesmo. Através deste trabalho, passamos a valorizar:

Indivíduos e interações mais que processos e ferramentas;

Software em funcionamento mais que uma documentação abrangente;

Colaboração com o cliente mais que negociação de contratos;

Responder a mudanças mais que seguir um plano.

Ou seja, mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda. ”

Princípio	Descrição
Envolvimento do cliente	Os clientes devem ser envolvidos em todo o processo de desenvolvimento. Seu papel é fornecer e priorizar novos requisitos de sistema e avaliar as iterações do sistema.
Acolher as mudanças	Tenha em mente que os requisitos do sistema mudam e, portanto, deve-se projetar o sistema para acomodar essas mudanças.
Entrega incremental	O software é desenvolvido em incrementos, e o cliente especifica os requisitos incluídos em cada um deles.
Manter a simplicidade	Deve-se ter como foco a simplicidade, tanto do software que está sendo desenvolvido quanto do processo de desenvolvimento. Sempre que possível, trabalhe ativamente para eliminar a complexidade do sistema.
Pessoas, não processos	As habilidades do time de desenvolvimento devem ser reconhecidas e aproveitadas da melhor maneira possível. Seus membros devem ter liberdade para desenvolver seu modo próprio de trabalhar sem se prender a processos determinados.

Os princípios dos métodos ágeis.

Métodos Ágeis

Os métodos ágeis têm sido úteis para dois tipos de desenvolvimento de sistemas:

- O desenvolvimento de um produto pequeno ou médio, por uma empresa de software, para venda;
- O desenvolvimento de sistemas personalizados dentro de uma organização, em que há um compromisso claro por parte do cliente de se envolver no processo de desenvolvimento;

Os sistemas pequenos e médios podem ser desenvolvidos por times situados no mesmo local, então a comunicação informal entre seus membros funciona bem;

Técnicas de Desenvolvimento Ágil

Técnicas de Desenvolvimento Ágil

A abordagem mais importante para mudança de cultura no desenvolvimento de software foi o desenvolvimento da **Programação Extrema** (*Extreme Programming*);

Foi concebido por Kent Beck (1990), com o **propósito de entregar software de alta qualidade** de forma rápida e contínua, adaptando-se às mudanças nos requisitos do cliente;

XP enfatiza a comunicação, feedback rápido, simplicidade, coragem e respeito entre os membros da equipe;

Técnicas de Desenvolvimento Ágil

Principais abordagens da **XP**:

- Comunicação constante;
- Feedback rápido;
- Desenvolvimento incremental;
- Testes automatizados;
- Programação em pares;
- Refatoração;
- Propriedade coletiva do código;

Técnicas de Desenvolvimento Ágil

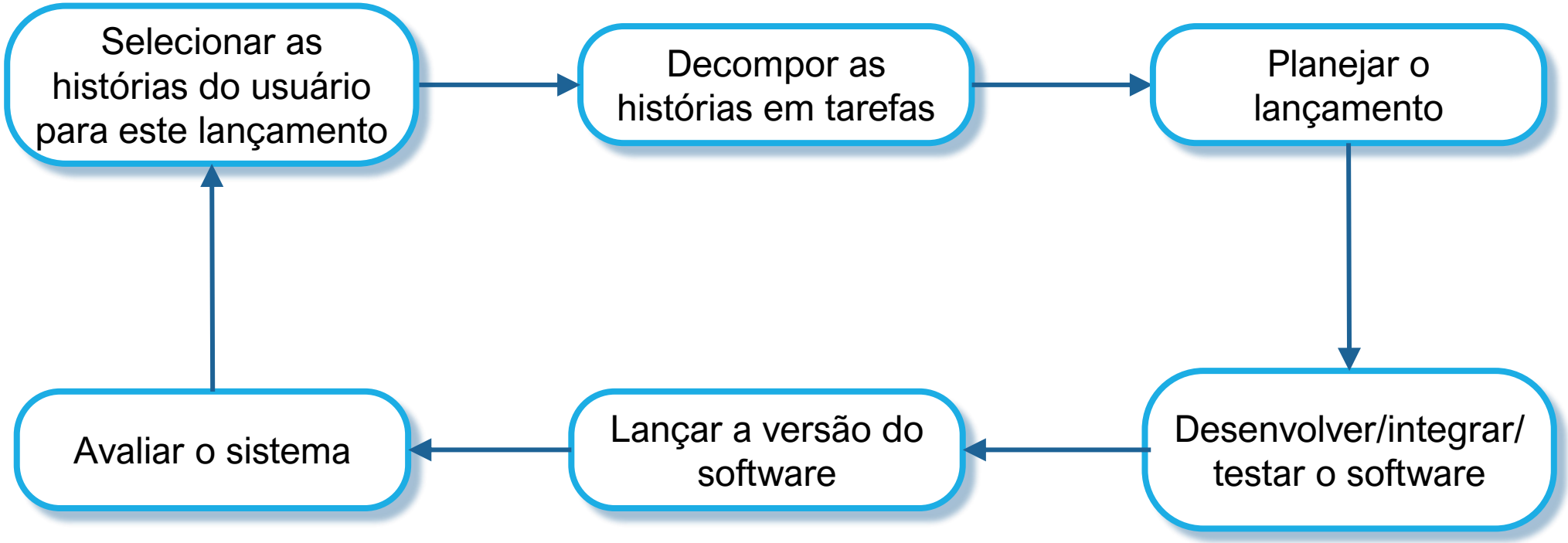
Vantagens da **XP**:

- Entrega rápida;
- Maior qualidade do software;
- Maior satisfação do cliente;
- Maior envolvimento da equipe;

Técnicas de Desenvolvimento Ágil

Desvantagens da **XP**:

- Requer habilidades de comunicação;
- Potencial para aumento de custos;
- Possível resistência cultural;
- Complexidade da implementação;



Ciclo de lançamento (*release*) da Programação Extrema.

Princípio ou prática	Descrição
Propriedade coletiva	Os pares de desenvolvedores trabalham em todas as áreas do sistema de modo que não se desenvolvem ‘ilhas de conhecimento’, e todos os desenvolvedores assumem a responsabilidade por todo o código. Qualquer um pode mudar qualquer coisa.
Integração contínua	Assim que o trabalho em um tarefa é concluída, ele é integrado ao sistema completo. Após qualquer integração desse tipo, todos os testes de unidade no sistema devem passar.
Planejamento incremental	Os requisitos são registrados em ‘cartões de histórias’, e as histórias a serem incluídas em um lançamento são determinadas de acordo com tempo disponível e com sua prioridade relativa. Os desenvolvedores decompõem essas histórias em ‘tarefas’ de desenvolvimento.
Representante do cliente	Um representante do usuário final do sistema (o cliente) deve estar disponível em tempo integral para o time de programação. Em um processo com esse, o cliente é um membro do time de desenvolvimento, sendo responsável por levar os requisitos do sistema ao time, visando sua implementação.

Princípio ou prática	Descrição
Programação em pares	Os desenvolvedores trabalham em pares, conferindo o trabalho um do outro e oferecendo o apoio necessário para que o resultado final seja sempre satisfeito.
Refatoração	Todos os desenvolvedores devem refatorar o código continuamente logo que sejam encontradas possíveis melhorias para ele. Isso mantém o código simples e de fácil manutenção.
Projeto (*design*) simples	Deve ser feito o suficiente de projeto (*design*) para satisfazer os requisitos atuais, e nada mais.
Lançamentos pequenos	O mínimo conjunto útil de funcionalidade que agregue valor ao negócio é desenvolvido em primeiro lugar. Os lançamentos do sistema são frequentes e acrescentam funcionalidade à primeira versão de um maneira incremental.
Ritmo sustentável	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.
Desenvolvimento com testes a *priori (test-first)*	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.

Práticas de programação Extrema.

Técnicas de Desenvolvimento Ágil

Os requisitos de software sempre mudam;

Nos métodos ágeis, a elicitação de requisitos é integrada ao desenvolvimento;

Os cenários são baseados nas experiências de um usuário – **Histórias do usuário**;

Cada história do usuário é registrada resumidamente em um **cartão de história**;

Prescrição de medicação

Kate é uma médica que deseja prescrever medicação para um paciente que frequenta sua clínica. O registro do paciente já é exibido em seu computador, então ela deve clicar no campo de medicação e selecionar ‘medicação atual’, ‘nova medicação’ ou ‘substâncias’.

Se escolher ‘medicação atual’, o sistema pedirá para que ela verifique a dose; se quiser muda-la, Kate fornecerá a nova dose e depois deverá confirmar a prescrição.

Se escolher ‘nova medicação’, o sistema presumirá que ela sabe qual medicação prescrever. Ao digitar as primeiras letras da medicação, o sistema exibirá uma lista de possíveis medicamentos que começam com essas letras. Ao escolher a medicação necessária, o sistema responderá pedindo que ela confirme se a medicação escolhida está correta. Ela deverá fornecer a dose e depois confirmar a prescrição.

Se escolher ‘substâncias’, o sistema exibirá uma caixa de busca para a substância aprovada, e então ela poderá pesquisar o medicamento que deseja. Feita a busca, lhe será solicitado que confirme se o medicamento selecionado está correto. Ela deverá fornecer a dose e depois confirmar a prescrição.

O sistema sempre verificará se a dose está dentro da faixa aprovada. Se não estiver, será pedido a Kate que faça uma alteração.

Depois de confirmar a prescrição, ela será exibida para conferência. Kate deverá clicar em ‘OK’ ou em ‘Alterar’. Se ‘OK’ for selecionado, a prescrição será registrada no banco de dados de auditoria. Se clicar em ‘Alterar’, o processo de ‘Prescrição de medicação’ é executado novamente.

Tarefa 1: Mudar a dose do medicamento prescrito

Tarefa 2: Seleção de substância

Tarefa 3: Verificação da dose

A verificação da dose é uma precaução de segurança para conferir se o médico não prescreveu uma dose perigosamente pequena ou grande.

Usando a identificação da substância para o nome genérico do medicamento, procure a substância e recupere a dose máxima e a mínima recomendadas.

Confira a dose prescrita em relação ao máximo e ao mínimo. Se estiver fora da faixa, emita uma mensagem de erro dizendo que a dose é alta ou baixa demais. Se estiver dentro da faixa, habilite o botão “Confirmar”.

Exemplos de cartões de tarefa para prescrição de medicação.

Técnicas de Desenvolvimento Ágil

A premissa fundamental da engenharia de software tradicional é a de que se deve projetar com vistas as mudanças;

A XP descartou esse princípio, considerando que é um desperdício de trabalho;

A XP sugere a **refatoração** constante do código, o que significa que o time de programação deve buscar possíveis melhorias no software e implementá-las imediatamente;

A **refatoração** melhora a estrutura do software a sua clareza, evitando a deterioração estrutural que ocorre naturalmente;

Técnicas de Desenvolvimento Ágil

A **refatoração** incluem:

- Reorganização de uma hierarquia de classe para remover código duplicado;
- Organização e renomeação de atributos e métodos;
- Substituição de seções de código similares, como chamadas para métodos definidos em uma biblioteca;

Na maioria dos ambientes de desenvolvimento incluem ferramentas para refatoração;

12 ferramentas para revisão de código

Review Board
Crucible
GitHub
Axolo
Collaborator
CodeScene
Visual Expert
Gerrit
Rhodecode
Veracode
Reviewable
Peer Review for Trac

Fonte: https://kinsta.com/pt/blog/ferramentas-de-revisao-de-codigo/

Técnicas de Desenvolvimento Ágil

Dentre as **diferenças importantes** entre o desenvolvimento incremental e o desenvolvimento dirigido por plano é a maneira **como o sistema é testado**;

No desenvolvimento incremental **não há uma especificação do sistema** que possa ser utilizada por uma **equipe de testes externa**;

Logo, os processos de testes são muito informais. Para contornar isso, a **Programação Extrema** desenvolveu uma nova abordagem ao teste de programas, automatizando-os, fazendo com que o processo de desenvolvimento só avance após todos os **testes terem sido executados com sucesso**;

Técnicas de Desenvolvimento Ágil

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;

Nos testes *a priori,* primeiro são escritos os testes e depois o código.

Teste 4: Conferir a dose

Entrada:

1. Um número em mg representando uma única dose de medicamento.
2. Um número representando a quantidade de doses únicas por dia.

Testes:

1. Teste das entradas em que a dose única está correta, mas a frequência é alta demais.
2. Teste das entradas em que a dose única é alta demais e baixa demais.
3. Teste das entradas em que a dose única * frequência é alta demais e baixa demais.
4. Teste das entradas em que a dose única * frequência está no intervalo permitido.

Saída:

“OK” ou mensagem de erro indicando que a dose está fora do intervalo seguro.

Descrição de caso de teste para conferir a dose.

Técnicas de Desenvolvimento Ágil

Outra prática que foi introduzida na Programação Extrema é que os **programadores trabalham em pares** no desenvolvimento do software;

Vantagens da programação em pares:

1. Apoia a ideia de propriedade e responsabilidade coletivas pelo sistema;
2. Age como um processo de revisão informal, já que cada linha de código é examinada por aos menos duas pessoas;
3. Incentiva a refatoração para melhorar a estrutura do software;

Gerenciamento ágil de Projetos

Gerenciamento Ágil de Projetos

Os **gerentes** precisam saber o que está acontecendo e se um projeto tente ou não a cumprir seus **objetivos**, entregar o software no **prazo** e dentro do **orçamento** proposto;

Na **abordagem dirigida por plano**, o **gerente** tem uma **visão estável** sobre tudo que deve ser **desenvolvido** e sobre os **processos** de desenvolvimento;

O ***planejamento informal*** proposta pelos métodos ágeis conflitaram com as necessidades de visibilidade impostas pelas empresas;

Para tratar dessa questão, foi desenvolvido o método ágil chamado **Scrum**;

Gerenciamento Ágil de Projetos

O **Scrum** tem o objetivo de proporcionar um **arcabouço para organizar os projetos ágeis** e, dar visibilidade externa, na medida do possível, do que está acontecendo;

O **Scrum** é uma metodologia ágil de desenvolvimento de software que se concentra na entrega contínua e iterativa de produtos de alta qualidade;

Surgiu inicialmente no campo da engenharia de software (1980), mas sua aplicabilidade se estende a uma variedade de setores além da tecnologia;

O termo “Scrum” surgiu em um artigo de HirotakaTakeuchi e Ikujiro Nonaka, publicado na revista Harvard Business Review em 1986;

Gerenciamento Ágil de Projetos

1990 que o **Scrum** começou a ser formalizado como uma metodologia específica para o desenvolvimento de software;

As principais características do **Scrum**:

1. Iterativo e incremental;
2. Transparência;
3. Adaptabilidade;
4. Papéis definidos;
5. Reuniões estruturadas;

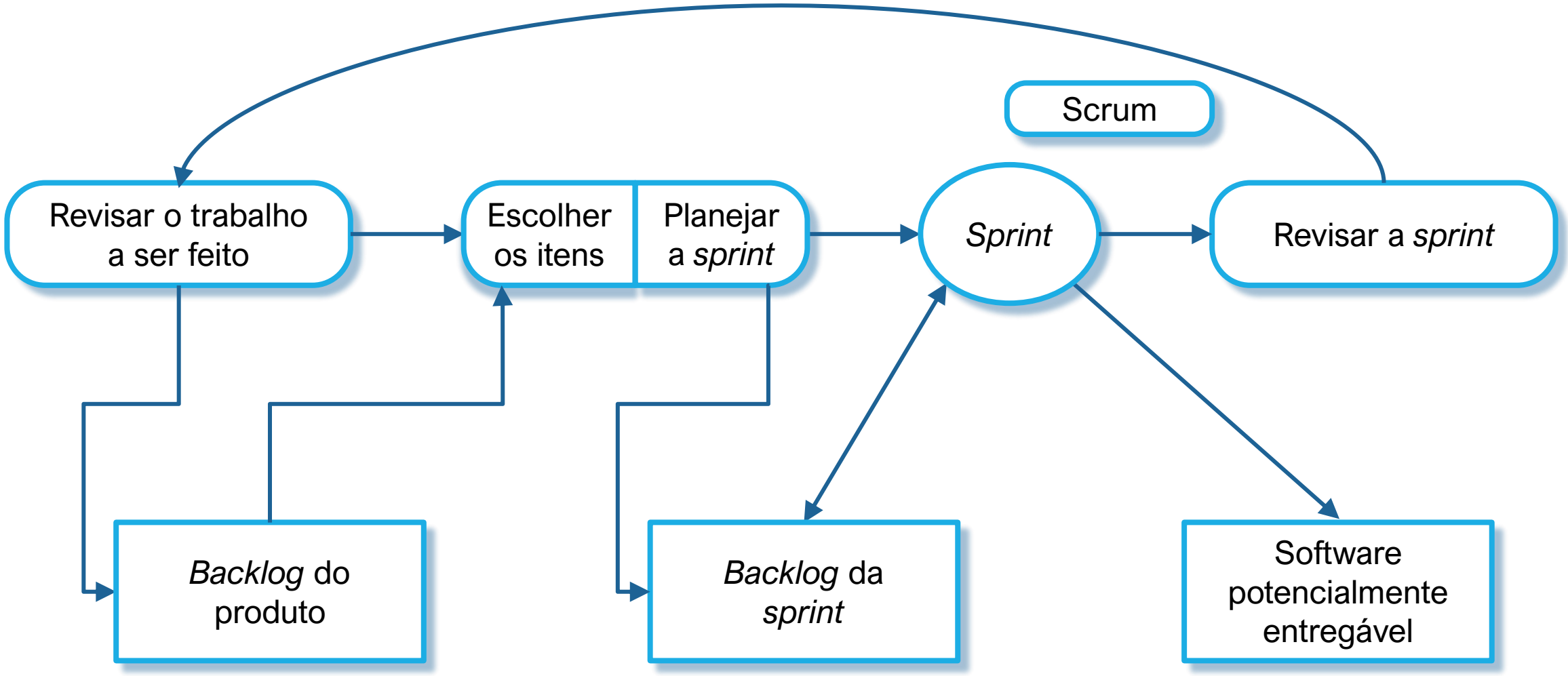
Gerenciamento Ágil de Projetos

Objetivos do **Scrum**:

1. Acelerar o tempo de entrega do produto;
2. Adaptar-se rapidamente às mudanças nas necessidades e requisitos dos clientes;
3. Melhorar continuamente a qualidade do produto e o processo de desenvolvimento;
4. Maximizar o valor entregue ao cliente;

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 reesposáveis por desenvolver o software e outros documentos essenciais do projeto.
Incremento de projeto potencialmente entregável	O incremento de software entregue a partir de uma *sprint*. 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.
Backlog 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.
Product Owner	Um indivíduo (ou possivelmente um pequeno grupo) cujo dever é identificar características ou requisitos do produto, priorizá-los para desenvolvimento e revisar continuamente o *backlog* do produto para garantir que o projeto continue a satisfazer as necessidades críticas do negócio. O *Product Owner*, também chamado de dono do produto, pode ser um cliente, mas também poderia ser um gerente de produto de produto em uma empresa de software ou um representante de um *stakeholder*.

Termo do Scrum	Definição
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.
Scrum Master	O *Scrum Master* 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 *Scrum Master* não ser considerado um gerente do projeto. No entanto, outros nem sempre podem ver a diferença facilmente.
Sprint	Uma iteração de desenvolvimento. As *sprints* normalmente duram de 2 a 4 semanas.
Velocidade	Uma estimativa de quanto esforço de *backlog* do produto um time pode cobrir em uma única *sprint*. Compreender a velocidade de um time ajuda a estimar o que pode ser coberto por uma *sprint* e constitui a base para medir a melhoria do desempenho.



O ciclo de *sprint* do Scrum.

Gerenciamento Ágil de Projetos

O ponto de partida para o ciclo da *sprint* do **Scrum** é o *backlog* do produto;

Backlog é uma lista de itens como características do produto, requisitos e melhorias de engenharia que precisam ser trabalhos pelo time **Scrum**;

O *backlog* é de responsabilidade do *Product Owner* (Dono do Produto), e pode ser especificado em vários níveis de detalhe;

Cada ciclo da *sprint* dura um intervalo de tempo fixo, que normalmente é de 2 a 4 semanas;

Procut Owner prioriza os itens no *backlog* do produto de cada ciclo;

Gerenciamento Ágil de Projetos

Durante a duração da *sprint*, o time faz reuniões curtas para revisar o progresso e, onde for necessário, alterar as prioridades de trabalho;

Os membros do time **compartilham** informações, descrevem seu progresso desde a última reunião, trazendo os problemas que surgiram e declaram o que foi planejado para o dia seguinte;

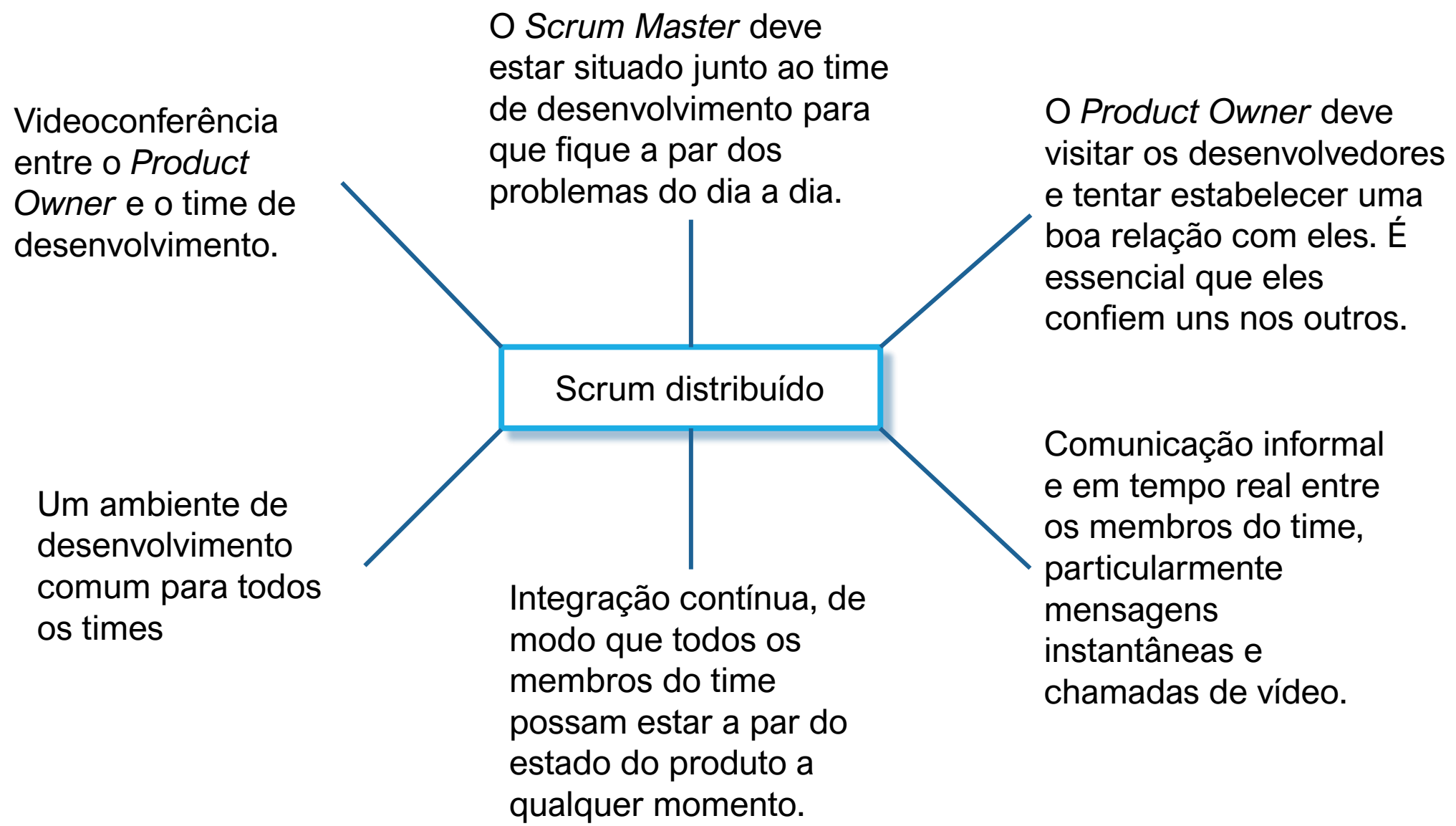
As interações diárias entre os times podem ser coordenadas usando um quadro do **Scrum**;

Próxima aula aqui

Gerenciamento Ágil de Projetos

O método Scrum proporciona:

1. O produto seja decomposto em um conjunto de “pedaços” gerenciáveis e compreensíveis aos quais os *stakeholders* podem se referir;
2. Requisitos instáveis não impedem o progresso;
3. O time inteiro tem visibilidade de tudo e, consequentemente, a comunicação e disposição de seus membros são melhores;
4. Os clientes veem a entrega dos incrementos na hora e obtêm *feedback* de como o produto funciona;
5. A confiança entre clientes e desenvolvedores é estabelecida, criando uma cultura positiva na qual todos esperam que o projeto tenha sucesso;



Scrum distribuído.

Escalabilidade dos Métodos Ágeis

Escalabilidade dos Métodos Ágeis

Os **métodos ágeis** foram desenvolvidos para serem **utilizados por pequenos times** de programação que poderiam trabalhar juntos na mesma sala e se comunicar informalmente;

Naturalmente, a necessidade de uma entrega mais rápida assolam o desenvolvimento de sistemas grandes;

A **escalabilidade** dos métodos ágeis tem duas facetas:

1. Escalar verticalmente esses métodos para lidar com o desenvolvimento de sistemas grandes demais para serem assumidos por um único time;
2. Escalar horizontalmente esses métodos dos times de desenvolvedores especializados para o uso disseminado em uma grande empresa com muitos anos de experiência em desenvolvimento de software;

Escalabilidade dos Métodos Ágeis

De modo geral, o desenvolvimento ágil tem sido **bem-sucedido**;

Entretanto, esses métodos podem não ser adequados para sistemas grandes e de longa vida útil desenvolvidos por uma empresa de software para um cliente externo;

Nestes casos, 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. Contudo, em empresas grandes, a maioria dos custos é na manutenção de sistemas existentes;

Escalabilidade dos Métodos Ágeis

3. Esses métodos são concebidos para pequenas empresas que compartilham uma localização geográfica, ainda que muito do desenvolvimento de software envolva, hoje em dia, times distribuídos mundialmente;

Questões contratuais podem ser um grande problema quando são utilizados os métodos ágeis;

Situações de manutenção que envolvam novos requisitos do negócio implicam em três problemas:

1. Falta de documentação do produto;
2. Dificuldade em manter os clientes envolvidos;
3. Continuidade do time de desenvolvimento;

Escalabilidade dos Métodos Ágeis

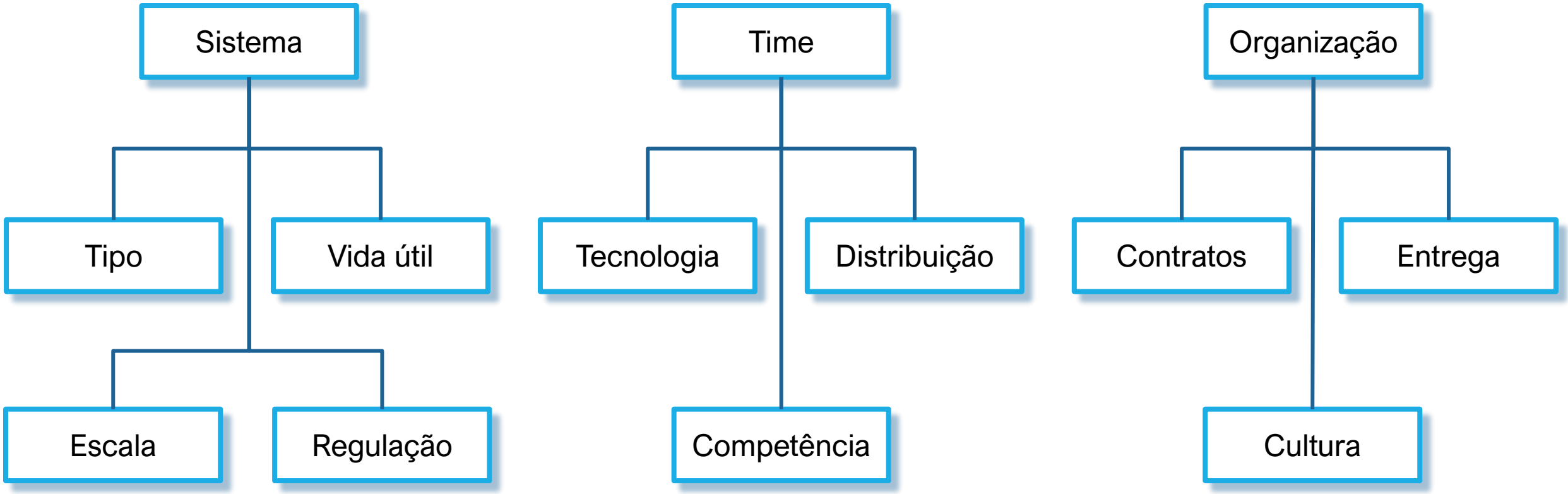
Um **requisito fundamental da escalabilidade** dos métodos ágeis é integrá-los às abordagens dirigidas por plano;

Pequenas empresas podem trabalhar com planejamento informal e de curto prazo;

Empresas grandes precisam ter planos e orçamentos de mais longo prazo para investimento, pessoal e evolução comercial;

Princípio	Prática
Envolvimento do cliente	Isso depende de ter um cliente disposto e capaz de investir tempo com o time de desenvolvimento e que possa representar todos os *stakeholders* do sistema. Muitas vezes, os representantes dos clientes têm outras demandas e não podem fazer parte do time de desenvolvimento em tempo integral. Nas situações em que existem *stakeholders* externos, como autoridades reguladoras, é difícil representar suas opiniões para o time ágil.
Acolher as mudanças	Pode ser extremamente difícil priorizar as mudanças, especialmente nos sistemas em que há muitos *stakeholders*. Geralmente, cada um deles atribui prioridades diferentes a mudanças diferentes.
Entrega incremental	As iterações rápidas e o planejamento de curto prazo do desenvolvimento nem sempre se encaixam nos ciclos de longo prazo do planejamento e marketing empresarial. Os gestores de marketing podem ter de conhecer as características do produto com vários meses de antecedência para preparar uma campanha eficaz.

Princípio	Prática
Manter a simplicidade	Sob a pressão dos cronogramas de entrega, os membros do time podem não ter tempo para realizar simplificações desejáveis no sistema.
Pessoas, não processos	Membros do time podem não ter a personalidade adequada para o envolvimento intenso, o que é característico dos métodos ágeis, e, portanto, podem não interagir bem com os demais membros do time.



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

Escalabilidade dos Métodos Ágeis

Os métodos ágeis forma desenvolvidos e refinados em projetos para desenvolver sistemas de negócio e produtos de **pequeno** e **médio porte**, em que o desenvolvedor controla a especificação do sistema;

Outros tipos de sistema têm atributos como **tamanho**, **complexidade**, resposta em **tempo real** e **regulação externa**;

Nesses casos é preciso algum planejamento, projeto e documentação antecipados no processo de engenharia de sistemas;

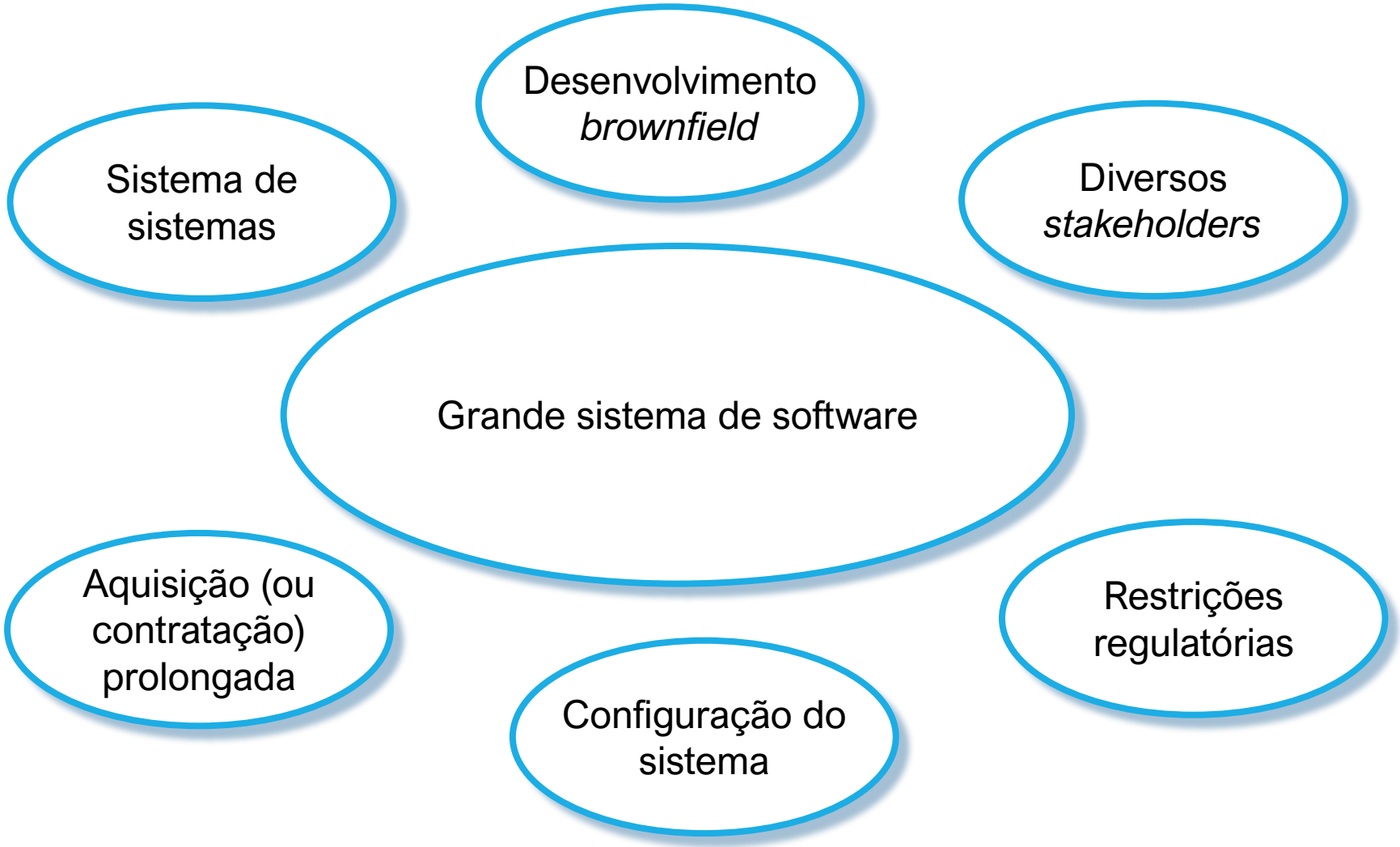
Escalabilidade dos Métodos Ágeis

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?

Os métodos ágeis têm que **evoluir** para serem utilizados no desenvolvimento de software de **larga escala**;

Esses tipos de software são muito mais **complexos** e **difíceis de compreender** e **gerenciar** do que os sistemas ou produtos de software pequenos;



Características dos grandes projetos.

Escalabilidade dos Métodos Ágeis

IBM desenvolveu um arcabouço para uso em larga escala dos métodos ágeis, chamado *Agile Scaling Model* (ASM);

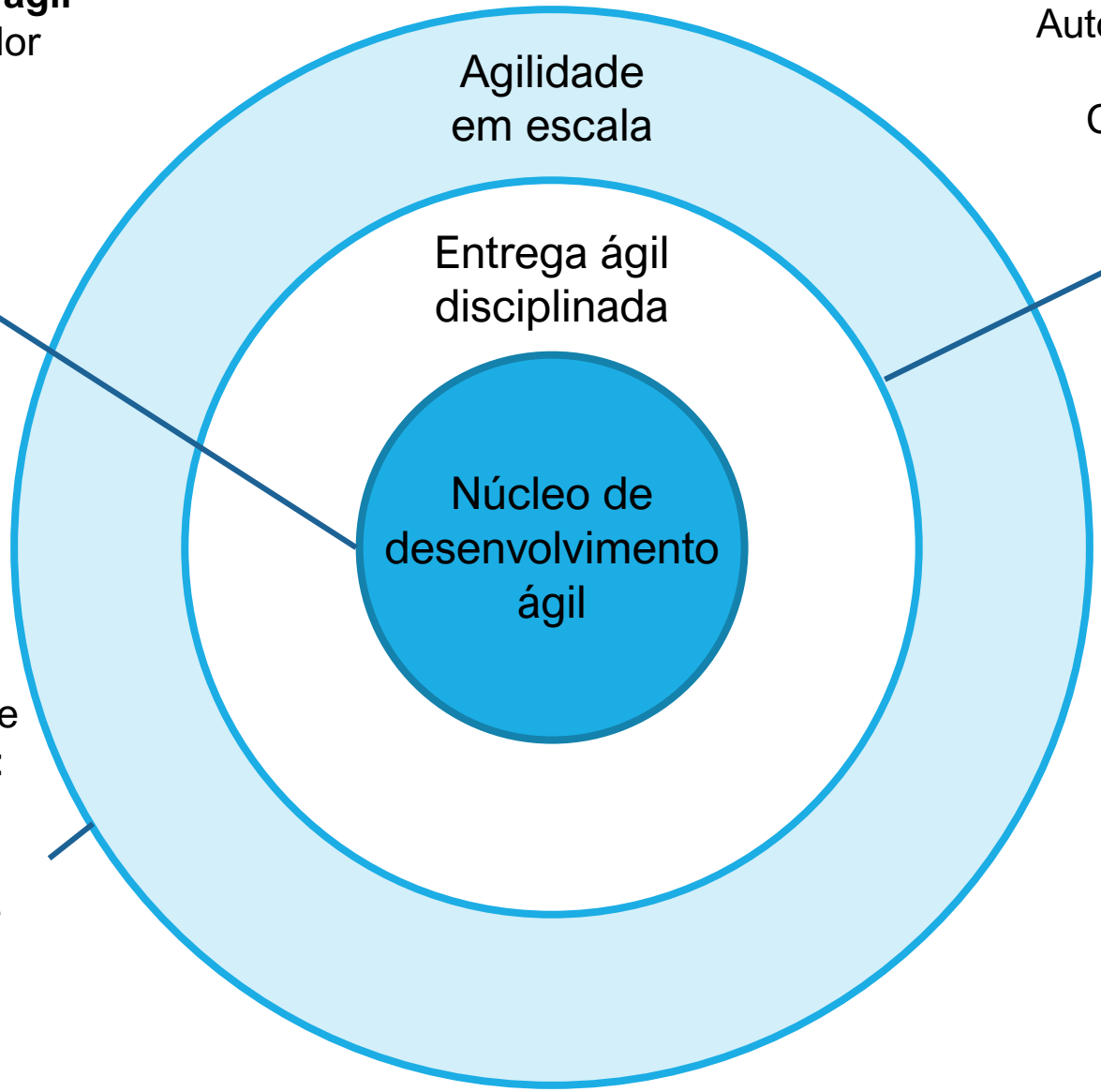
ASM reconhece que a escalabilidade é um processo que ocorre em etapas;

Os times de desenvolvimento passam do chamado "núcleo de desenvolvimento ágil" para "entrega ágil disciplinada";

A etapa final da escalabilidade no ASM é a passagem para a "agilidade em escala", quando é reconhecida a complexidade inerente aos projetos grandes;

Núcleo de desenvolvimento ágil
Ciclo de vida dirigido por valor
Times auto-organizados
Foco na construção

Entrega ágil disciplinada
Ciclo de vida dirigido por risco + valor
Auto-organizada, com ambiente de governança apropriado
Ciclo de vida de entrega total



Agilidade em escala
Entrega ágil disciplinada em que se aplicam fatores de escalabilidade:
Time grande
Distribuição geográfica
Conformidade com regulamentos
Complexidade técnica
Complexidade organizacional
Disciplina empresarial

Modelo de agilidade em escala da IBM.

Escalabilidade dos Métodos Ágeis

O **Scrum** tem sido adaptado ao desenvolvimento em larga escala;

As características fundamentais do Scrum multitimes são:

1. *Replicação de papéis*. Cada time tem um *Product Owner* para o seu componente de trabalho e um *Scrum Master*;
2. *Arquitetos de produto*. Cada time opta por um arquiteto de produto e eles colaboram para projetar e desenvolver a arquitetura global do sistema;
3. *Alinhamento de entrega*. As datas das entregas do produto de cada time são alinhadas;
4. *Scrum de Scrums*. Há um Scrum dos Scrums diário, no qual representantes de cada time se reúnem para discutir o progresso, identificar problemas e planejar o trabalho a ser feito naquele dia;

Escalabilidade dos Métodos Ágeis

Empresas pequenas são adeptos dos **métodos ágeis**, pois não são limitadas pelas burocracias organizacionais ou pelos padrões de processo;

Naturalmente, **empresas maiores** tem mais **dificuldade** em utilizar os métodos ágeis;

Essa dificuldade acontece por uma série de razões:

1. Gerentes de projeto sem experiência com métodos ágeis;
2. Padrões e procedimentos de qualidade que são incompatíveis com os métodos ágeis, devido a sua burocracia;
3. Habilidades e capacidades dos membros do time;
4. Resistência cultural aos métodos ágeis;

Resumo

Resumo

Os métodos ágeis são de desenvolvimento iterativo e se concentram na redução de sobrecarga no processo e na documentação, bem como na entrega incremental de software. Eles envolvem representantes do cliente diretamente no processo de desenvolvimento.

A decisão de usar uma abordagem ágil ou uma dirigida por plano para o desenvolvimento deve depender do tipo de software que está sendo desenvolvido, da capacidade do time de desenvolvimento e da cultura da empresa que está desenvolvendo o sistema. Na prática, pode-se utilizar uma mistura de técnicas ágeis e dirigidas por plano.

Resumo

As práticas de desenvolvimento ágil incluem requisitos expressados na forma de histórias do usuário, programação empares, refatoração, integração contínua e desenvolvimento com testes a *priori (test-first).*

O Scrum é um método ágil que define um arcabouço para organizar projetos ágeis. Ele gira em trono de *sprints*, que são períodos de tempo fixos nos quais um incremento do sistema é desenvolvido. O planejamento se baseia em priorizar um *backlog* de trabalho e selecionar as tarefas de maior prioridade para uma *sprint.*

Resumo

Para promover a escalabilidade de métodos ágeis, algumas práticas dirigidas por plano precisam ser integradas à prática ágil. Elas incluem requisitos antecipados, vários representantes do cliente, mais documentação, ferramental comum aos times de projeto e o alinhamento das entregas entre os times.

Referências Bibliográficas

SOMMERVILLE, Ian. **Engenharia de software**. 10. ed. São Paulo, SP: Pearson, 2018.