

# Processos de software (OPENUP, XP, SCRUM)

## Apresentação

Desenvolver softwares não é uma tarefa fácil. Além de complexa e de alto risco, envolve problemas, tais como gerenciamento de recursos, problemas com usuários e até mesmo o cancelamento do desenvolvimento devido à inviabilidade de investimentos.

As metodologias ágeis, tais como OPENUP, XP e SCRUM, prometem auxiliar a minimizar riscos e custos no desenvolvimento de software ajudando na atualização e no diálogo com os clientes, entendendo melhor as suas necessidades.

Nesta Unidade de Aprendizagem, você aprenderá o que são métodos ágeis, bem como o seu funcionamento.

Bons estudos.

**Ao final desta Unidade de Aprendizagem, você deve apresentar os seguintes aprendizados:**

- Definir os métodos ágeis de desenvolvimento de software.
- Descrever o funcionamento dos processos de software (OpenUp, Extreme Programming [XP] e Scrum)
- Identificar as principais diferenças entre os processos de software (OpenUp, Xp e Scrum).

# Desafio

Considere o seguinte cenário:

Você é um gestor de projetos e foi convidado para gerenciar um projeto de desenvolvimento de um sistema de e-commerce.

O cliente em questão tem pressa no projeto e pouco recurso para investimento.

Você aceitou o desafio e iniciou o projeto fazendo uma reunião com o cliente para entender as necessidades e começar a definir como trabalhará o projeto, como será a equipe e qual metodologia utilizará.

O cliente já posicionou que não sabe exatamente como o e-commerce deve funcionar.

Baseado nas informações acima, você deverá decidir entre metodologias tradicionais ou ágeis para trabalhar o seu projeto e justificar a sua escolha.

# Infográfico

Processos de softwares é um conjunto de ações que são desmembradas em um conjunto de tarefas ao qual deverão ser trabalhadas, garantindo a qualidade e respeitando os marcos utilizados para indicar seu progresso.

Desenvolver softwares não é uma tarefa fácil. Além de complexa e de alto risco, envolve problemas, tais como **gerenciamento de recursos**, **problemas com usuários** e até mesmo o cancelamento do desenvolvimento devido à **inviabilidade de investimentos**.

Além disso, é necessário estar atento para que ele seja desenvolvido com a rapidez necessária para não perder o *time* em tempos tão competitivos.

Para desenvolver um software, podemos utilizar processos de modelos tradicionais ou processos ágeis. Neste Infográfico, você verá de forma resumida os 12 princípios da metodologia ágil.

# 12 PRINCÍPIOS DO MÉTODO ÁGIL

Para melhorar o gerenciamento e a performance dos projetos, um grupo de líderes da comunidade do Extreme Programming (XP) se reuniu com o intuito de discutir as várias questões que envolviam o processo de desenvolvimento com XP, e dessa reunião surgiu o documento do Manifesto Ágil. O manifesto ágil é simples, direto e claro.

Conheça os 12 princípios dessa metodologia:



Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

**1**

Satisfazer o cliente com a entrega adiantada e contínua. Neste princípio, a prioridade é a felicidade do cliente. As entregas são semanais e antecipadas.



**2**

Aceitar e se adequar às mudanças. O gestor do projeto precisa ser flexível, pois possivelmente ele terá que fazer alterações, inclusões e exclusões com frequência.



**3**

Entregas com frequência e períodos curtos. A cada *sprint* concluída, o time do projeto deve entregar uma funcionalidade ao cliente.



**4**

Times multidisciplinares trabalhando diariamente. O trabalho em equipe é muito importante, pois é o que permite personalizar a solução e validá-la a cada *sprint* concluída.



**5**

Envolver, motivar, apoiar e confiar no time. É fundamental que a equipe do projeto esteja motivada a desempenhar seu papel e tenha um ambiente agradável para o trabalho.



**6**

Priorizar uma conversa direta e pessoal. As informações internas e externas precisam ser repassadas para a equipe de forma direta e clara.



**7**

As entregas representam o progresso do trabalho. O progresso do desenvolvimento é medido pelas entregas e não pela conclusão das atividades.



**8**

Processos ágeis promovem um ambiente sustentável. Deve-se construir o ambiente ideal para o desenvolvimento de projetos, com planejamento por iterações e envolvimento de todos os afetados pelo trabalho.



**9**

Excelência técnica e bom design aumentam a agilidade. A revisão constante dos requisitos técnicos, como também do design, permite a entrega de uma solução realmente alinhada aos objetivos de negócios do cliente.



**10**

Simplicidade. Evitar trabalho que não precisa ser feito. Menos é mais. O trabalho é mais simples de ser executado.



**11**

Times auto-organizáveis geram melhores resultados. Times ágeis são compostos por profissionais com capacidade de se organizarem por si mesmos.



**12**

Reflexões, ajustes e otimização em intervalos regulares. A revisão do trabalho realizada ao final de cada *sprint* permite que a própria equipe avalie sua performance.



# Conteúdo do Livro

---

Processos de desenvolvimento de software baseados em especificações completas de requisitos, projeto, construção e testes de sistema não estão voltados para o desenvolvimento rápido de software.

Em tempos de tanta competitividade, metodologias ágeis são utilizadas para entregas rápidas com foco na satisfação do cliente.

No capítulo Processos de software (OPENUP, XP, SCRUM), do livro *Engenharia de software*, você vai aprender como as principais metodologias ágeis são utilizadas, bem como a diferença entre elas.

Boa leitura.

# ENGENHARIA DE SOFTWARE

A photograph showing four people in a collaborative workspace. A man with a large afro hairstyle is leaning over a table, looking at a tablet device. Next to him, another man in a striped shirt is also focused on the screen. To the right, a woman with long dark hair is looking down at some papers. In the background, another person's head is visible. The table is cluttered with papers, a laptop, and a few cups. The scene is set against a backdrop of abstract geometric shapes in shades of orange, purple, and brown.

Adriana de Souza  
Vettorazzo

# Processos de software

## Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Definir os métodos ágeis de desenvolvimento de *software*.
- Descrever o funcionamento dos processos de *software* (OpenUP, *Extreme Programming* [XP] e Scrum).
- Identificar as principais diferenças entre os processos de *software* (OpenUP, XP e Scrum).

## Introdução

Neste capítulo, você vai estudar sobre processos de *software*, que são atividades de produção de um produto de *software*. Porém, cada vez mais novos *softwares* são desenvolvidos com a ampliação ou a modificação de sistemas já existentes.

Os processos de *software* são complexos e dependem da criatividade humana, o que vem causando insucesso nas tentativas de automatizar essas atividades.

A utilização de processos de *software* visa a entregar *softwares* com qualidade, dentro do prazo esperado e garantindo sempre a satisfação do cliente/usuário.

Embora existam muitos processos de *software* diferentes, algumas atividades fundamentais são comuns a todos eles: especificação do *software*, projeto e implementação de *software*; validação do *software*; evolução de *software*.

## Métodos ágeis de desenvolvimento de *software*

As mudanças na economia, nos serviços concorrentes e nos novos produtos tem sido cada vez mais rápida e os negócios precisam responder com a mesma velocidade, para não perder essas novas oportunidades. Como o *software* está presente em todas as operações do negócio, é essencial que novos *softwares*

sejam desenvolvidos rapidamente para não perder essas novas oportunidades e também responder às pressões da competitividade.

Pensando em formas de desenvolver *softwares* com maior velocidade e mantendo a qualidade e a satisfação dos clientes, em 2001, Kent Beck e outros 16 desenvolvedores assinaram o “Manifesto para o desenvolvimento Ágil de Software”, que deu origem aos processos de desenvolvimento para criar *software* útil rapidamente. Geralmente, são processos iterativos nos quais a especificação, o projeto, o desenvolvimento e o teste são intercalados.

Usando esse método, o *software* não é desenvolvido e disponibilizado integralmente, e sim uma série de incrementos e a cada novo incremento uma nova funcionalidade do sistema. Existem muitas abordagens para o desenvolvimento de *software* rápido, e elas compartilham de características fundamentais:

- os processos de especificação, projeto e implementação são concorrentes.
- o sistema é desenvolvido em uma série de incrementos;
- as interfaces com o usuário são desenvolvidas usando um sistema de desenvolvimento interativo.

## Princípios da agilidade

Para garantir a agilidade no desenvolvimento do *software*, veja a seguir os princípios de agilidade. Embora nem todo modelo de processo ágil aplique todos esses métodos, tais princípios definem um espírito ágil.

- A maior prioridade é satisfazer o cliente.
- Acolha bem os pedidos de alterações, mesmo se o projeto estiver atrasado.
- Faça entregas frequentes do *software* em funcionamento.
- A equipe comercial e os desenvolvedores devem trabalhar em conjunto.
- Mantenha a equipe motivada.
- Mantenha conversas abertas e presenciais com a equipe.
- *Software* em funcionamento é a principal medida do progresso.
- Os processos ágeis promovem desenvolvimento sustentável.
- Mantenha a excelência técnica.
- Simplicidade é essencial.
- As melhores arquiteturas, requisitos e projetos emergem de equipes que se auto-organizam.
- A equipe tem sempre foco na eficiência.

Entre os princípios ágeis, destacamos os principais no Quadro 1.

**Quadro 1.** Princípios dos métodos ágeis

Princípio	Descrição
Envolvimento do cliente	Clientes devem ser profundamente envolvidos no processo de desenvolvimento.
Entrega incremental	O <i>software</i> é desenvolvido em incrementos e o cliente especifica os requisitos a serem incluídos em cada um.
Pessoas, não processo	As habilidades da equipe de desenvolvimento devem ser reconhecidas e exploradas.
Aceite as mudanças	Tenha em mente que os requisitos do sistema vão mudar, por isso, projete o sistema para lidar e acomodar essas mudanças.
Mantenha a simplicidade	Concentre-se na simplicidade do <i>software</i> . Sempre que possível, elimine a complexidade.

*Fonte:* Adaptado de Sommerville (2011).

## Processos de software

Das metodologias ágeis de desenvolvimento de *software*, podemos citar, entre as mais usadas, XP, Scrum e OpenUP.

### XP

O método ágil mais conhecido é a XP. Nesse método, todos os requisitos são expressos como cenário que são implementados diretamente como tarefas. Os programadores trabalham em pares e desenvolvem testes para cada tarefa descrita antes do código.

Esse modelo define um conjunto de cinco valores que estabelecem as bases para todo trabalho realizado.

- 1. Comunicação:** a XP procura manter as comunicações certas fluindo por meio do emprego de muitas práticas que não podem ser feitas sem comunicação.
- 2. Simplicidade:** a XP aposta que é melhor fazer uma coisa simples hoje e investir um pouco mais depois para fazer alguma modificação, se

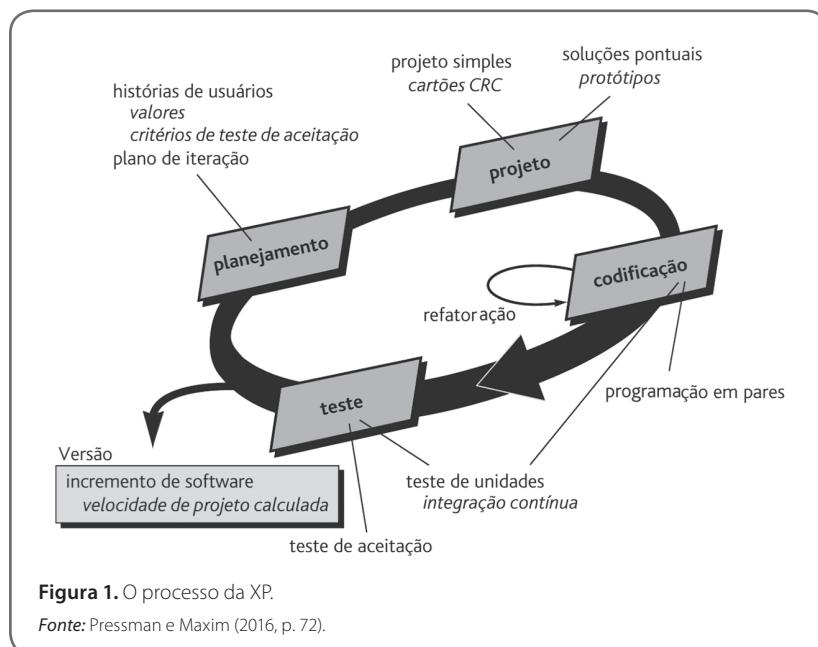
necessária, do que fazer algo mais complexo hoje e que talvez nunca será utilizado.

3. **Feedback:** o *feedback* concreto melhora a comunicação e a simplicidade.  
Quanto mais *feedbacks*, mais fácil será a comunicação.
4. **Coragem:** um projeto começa simples e depois é transformado para algo mais complexo e ousado.
5. **Respeito:** respeito é um valor que dá sustentação a todos os demais.  
Sem o respeito com a equipe, os outros valores ficam comprometidos.

## Processo XP

A XP emprega uma abordagem orientada a objetos e envolve um conjunto de regras e práticas (Figura 1):

- planejamento;
- projeto;
- codificação;
- testes.



**Figura 1.** O processo da XP.

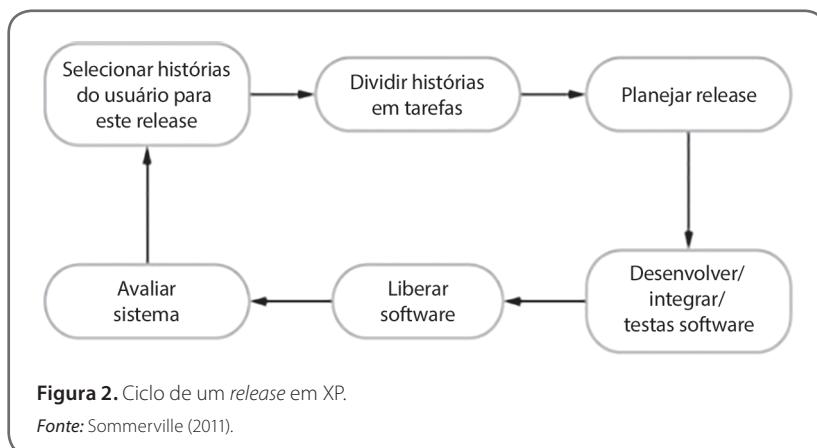
*Fonte:* Pressman e Maxim (2016, p. 72).

Essas práticas se enquadram nos seguintes princípios dos métodos ágeis.

- O desenvolvimento incremental é apoiado por pequenos e frequentes *releases* do sistema baseado nos cenários dos clientes.
- O envolvimento do cliente é apoiado pelo engajamento em tempo integral.
- As pessoas, não o processo, são apoiadas ao dividirem a programação com seus pares.
- As mudanças são apoiadas por meio de releases regulares do sistema.
- A manutenção da simplicidade é apoiada pelo *refactoring* constante para aprimorar a qualidade do código.

Em um processo XP, os clientes são envolvidos na especificação e priorização dos requisitos do sistema. Ao invés de especificar os requisitos como lista de funções, o cliente é parte da equipe de desenvolvimento e participa da discussão dos cenários com os outros membros da equipe e, dessa forma, desenvolvem um “cartão de histórias” que engloba as suas necessidades.

Depois, a equipe de desenvolvimento trabalha na implementação do cenário descrito no “cartão de histórias” em um *release do software* (Figura 2).



O cliente prioriza as histórias para implementação, escolhendo as que podem ser utilizadas imediatamente para apoio útil aos negócios. Essas novas versões são entregues aos clientes, mas só será aceita somente se todos os testes forem executados com sucesso.

Enquanto a engenharia de *software* tradicional espera que o desenvolvedor faça previsões futuras e antecipe e incorpore essas mudanças ao novo *software*, a

XP descarta esse princípio alegando que projetar para a mudança é, geralmente, um esforço inútil, pois muitas vezes essas mudanças não ocorrem e as solicitações de mudanças concretas podem ser completamente diferentes.

## Os papéis no XP

Os papéis no XP têm por objetivo auxiliar para que cada membro da equipe contribua com o melhor de suas habilidades, para o sucesso da equipe. Com relações de respeito entre a equipe, cada um dá o melhor de si, mesmo que o seu papel dentro da equipe seja outro. Dessa forma, o foco é a equipe.

Dentro de uma equipe XP, podemos ter os seguintes papéis.

- **Executivos:** o papel dos executivos está focado no auxílio da definição do escopo do projeto, bem como em manter alinhado com a organização. Além disso, atuam no auxílio à equipe para desenvolver o seu trabalho, administrando as pressões e os obstáculos.
- **Gerentes de projeto:** gerentes de projeto servem de ponte entre a equipe, os clientes e os fornecedores. Os gerentes precisam assegurar que as pessoas certas dialoguem dentro da equipe e fora dela.
- **Gerentes de produto:** o gerente de produto é o responsável por ajudar o produto a tomar corpo de acordo com o que foi projetado, evitando que as iterações modifiquem o projeto.
- **Arquitetos:** ajudam os desenvolvedores no dia a dia por meio da programação em par e podem ajudar a equipe a criar testes que exercitem a arquitetura como um todo.
- **Analistas de teste:** são os responsáveis por ajudar clientes e desenvolvedores, no início das iterações, a escrever os testes.
- **Designers de interação:** ajudam a criar a interface. Os *designers* trabalham bem próximo dos clientes do projeto.
- **Programadores:** trabalham em pares na implementação do projeto. Eles automatizam as tarefas repetitivas e criam as novas funcionalidades.
- **Recursos humanos:** o setor de recursos humanos é a equipe responsável pelo recrutamento e pela seleção da equipe para o projeto. Habilidades técnicas e de trabalho em equipe são itens avaliados no momento do recrutamento.
- **Redatores técnicos:** são os responsáveis em criar e manter a documentação do projeto.
- **Usuários:** têm participação valiosa no projeto, ainda mais quando têm domínio sobre o negócio. São envolvidos desde o início.

## Scrum

Scrum é um *framework* dentro do qual as pessoas podem tratar e resolver problemas complexos e adaptativos, enquanto produtiva e criativamente entregam produtos com o mais alto valor possível. É leve, simples de entender e extremamente difícil de dominar (Figura 3).

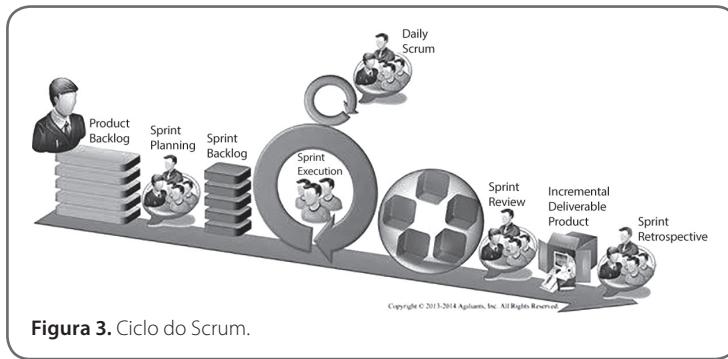


Figura 3. Ciclo do Scrum.

No Scrum, os projetos são divididos em ciclos (tipicamente mensais) chamados de *sprints*. O *sprint* representa um *time box* dentro do qual um conjunto de atividades deve ser executado, sendo elas:

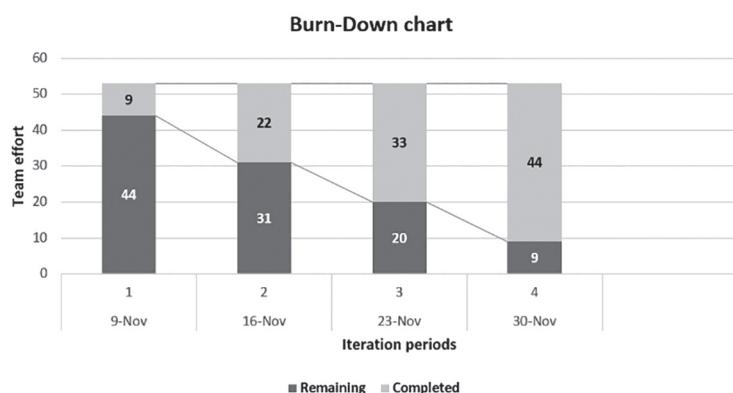
- requisitos;
- análise;
- projeto;
- evolução;
- entrega.

Cada uma dessas atividades é realizada dentro de um padrão de processos que definem um conjunto de ações de desenvolvimento, descritas a seguir.

- **Registro pendente de trabalho (*backlog*):** trata-se de uma lista de prioridades dos requisitos ou funcionalidades que fornecem valor ao cliente. Os itens podem ser adicionados a essa lista a qualquer tempo e o gerente de produto avalia e atualiza a lista de prioridades para a entrega.
- **Urgências/*sprints*:** são unidades de trabalho solicitadas para atingir um requisito estabelecido no *backlog* e que precisa ser ajustado pra atender o prazo já acordado para entrega.

- **Reuniões Scrum:** são reuniões rápidas realizadas com frequência diária com a equipe. Nessas reuniões, trabalham-se três perguntas:
  - O que você realizou desde a última reunião?
  - Quais foram as dificuldades encontradas?
  - O que planeja realizar até a próxima reunião?
- **Demos:** uma demonstração é entregue ao cliente para avaliação. Deve contemplar funções que devem ser entregues ao cliente dentro do prazo estipulado.

Em um projeto Scrum, a equipe monitora seu progresso em relação a um plano atualizando um *release burndown chart* ao final de cada *sprint* (iteração). *Release burndown chart* é um gráfico atualizado diariamente para demonstrar o progresso do trabalho feito pela equipe (Figura 4).



**Figura 4.** Exemplo de *burndown chart*.

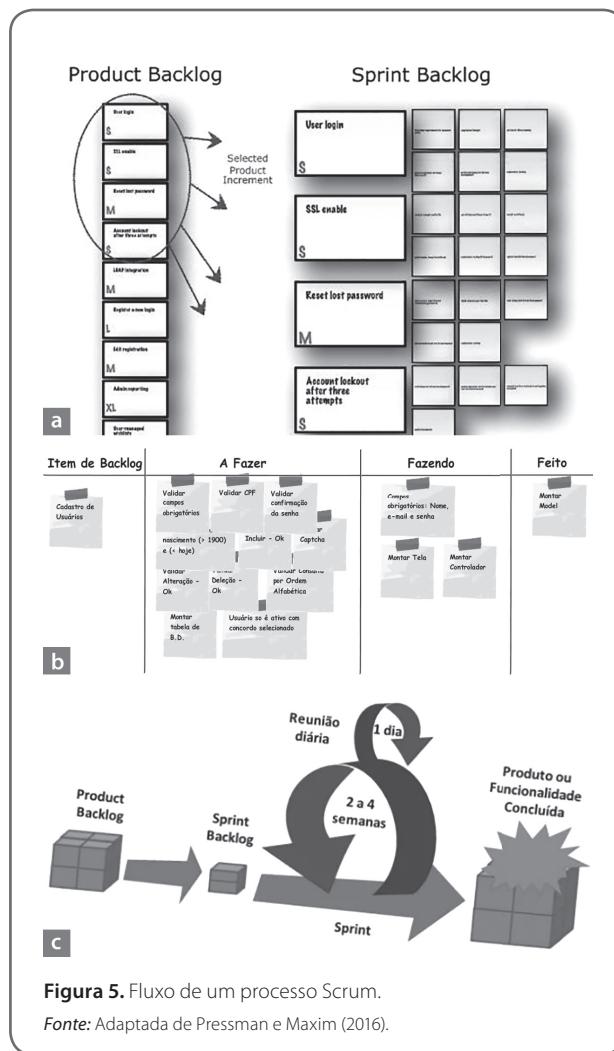
*Fonte:* Muhsinzoda (2015, documento on-line).

O eixo horizontal de um *release burndown chart* mostra os *sprints*. O eixo vertical mostra a quantidade de trabalho que ainda precisa ser feita no início de cada *sprint*. O trabalho que ainda resta pode ser mostrado na unidade preferencial da equipe: *story points*, dias ideais, *team days* e assim por diante.

Um princípio importante do Scrum é a ideia de transparência. Todos na equipe precisam estar sabendo o que os outros estão fazendo, os progressos que estão acontecendo e onde a equipe quer chegar. É por isso que dar visibilidade a tudo é tão importante.

Uma boa parte disso está no Quadro Scrum. É o lugar onde você pode organizar seu *backlog*, assim como tarefas que estão sendo trabalhadas no *sprint* atual e seu andamento (Figura 5).

Um quadro Scrum pode ser tão simples quanto um quadro branco com notinhas de papel para cada tarefa, ou tão complicado quanto um *software* especializado, com gráficos e ferramentas de acompanhamento de tarefas.



## OpenUP

OpenUP é uma metodologia que segue o processo unificado de engenharia de *software*, porém com uma cultura aberta e enxuta. Ele utiliza a filosofia ágil, que foca na natureza colaborativa de desenvolvimento de *software*.

Essa metodologia é construída por meio de processos simples, os quais podem ser expandidos para quaisquer tipos de *softwares*. O OpenUP foi criado como um projeto de exemplo aplicando as melhores práticas de engenharia de *software* do *Eclipse Process Framework* (EPF). Assim sendo, o OpenUP é um conjunto das metodologias ágeis, como Scrum e XP, somado a um pouco da arquitetura do processo unificado.

OpenUP tem em seu núcleo um conjunto de premissas que faz dele uma metodologia funcional e em que, ao seguir essas premissas, é possível obter boas referências ao projeto e tornar este um produto com qualidade e funcional. São princípios da OpenUP:

- colaborar para alinhar os interesses e compartilhar os conhecimentos adquiridos.
- balancear as prioridades concorrentes para maximizar os valores dos *stakeholders*.
- focar primariamente na arquitetura, visando a minimizar os riscos e planejar o processo.
- envolver os *stakeholders* para obter contínuo *feedback* do desenvolvimento.

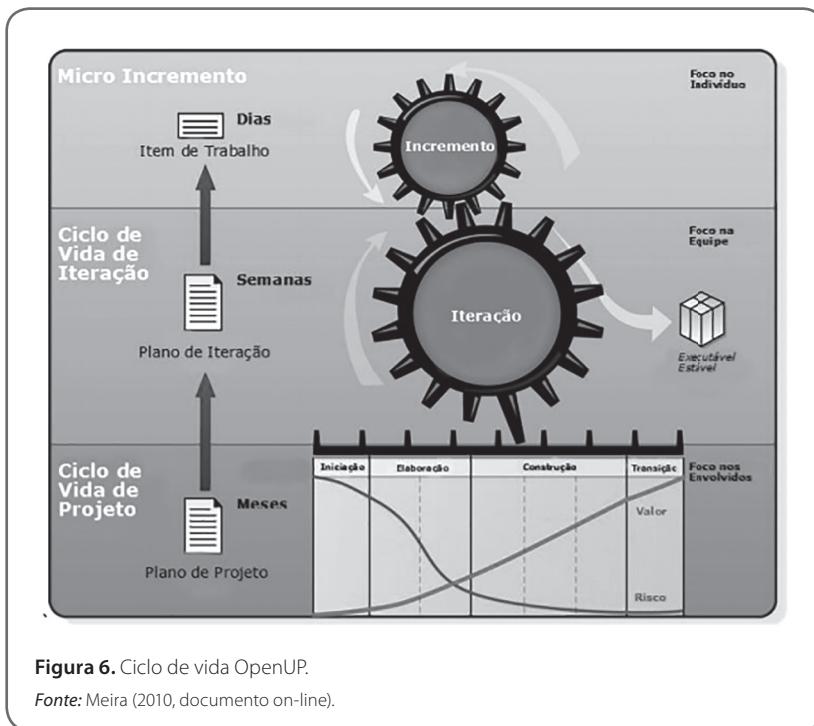


### Saiba mais

OpenUP se baseia em um ciclo iterativo e incremental, porém, esse processo foca em criar um modelo ágil e enxuto. Outro diferencial da metodologia é que ela tem, além da iteração normal, um microincremento, que é uma pequena unidade de trabalho de uma pessoa da equipe em que são executadas iterações.

Na Figura 6, é possível visualizar as etapas do ciclo de vida de um projeto OpenUP baseado em quatro fases.

1. **Iniciação:** é a fase em que se enfatiza o processo de análise de negócios e análise de requisitos do negócio analisado, dando uma ênfase menor à arquitetura e à implementação.
2. **Elaboração:** é a fase em que se enfatiza o processo de desenvolvimento da análise arquitetural da solução proposta.
3. **Construção:** é a fase em que se enfatiza o processo de implementação da solução proposta, bem como testes e integração.
4. **Transição:** é a fase em que se enfatiza o processo de implantação do *release*, com importante foco na realização do teste beta e reconfiguração necessária do sistema, além de foco no processo de treinamento do usuário e conversão dos dados legados.



**Figura 6.** Ciclo de vida OpenUP.

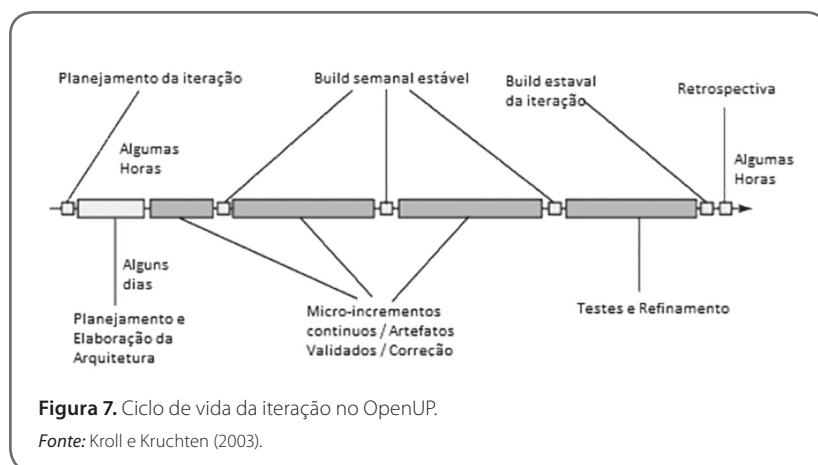
*Fonte:* Meira (2010, documento on-line).

Durante as fases citadas, alguns itens precisam ser respondidos. Caso esteja faltando algum item ou esteja incompleto, o projeto pode sofrer com alguns problemas. Veja a seguir.

- Na iniciação, todos devem concordar com o escopo e o objetivo do projeto. Caso haja discordância, problemas futuros colocarão em risco todo o projeto.
- Na elaboração, a arquitetura precisa ser concreta e funcional. Se houver riscos, estes deverão ser mitigados.
- Na construção, é feito o desenvolvimento do sistema que deve fazer entregas incrementais, demonstrando as funcionalidades aos *stakeholders*.
- Na fase de transição, se a aplicação sair da fase de construção, é necessário que seja totalmente funcional e sem problemas para ser colocada em produção.

Nesse ciclo o foco é a entrega semanal de um protótipo ao cliente. Dessa forma, é possível detectar erros e até mesmo requisitos distorcidos. Se encontrados, o foco é somente corrigir os erros e continuar o cronograma do projeto.

O plano da iteração está concentrado nos itens entregáveis de acordo com sua prioridade. Quanto mais alta a prioridade, mais cedo deve entrar no plano da iteração. Veja a Figura 7.



**Figura 7.** Ciclo de vida da iteração no OpenUP.

Fonte: Kroll e Kruchten (2003).

O método OpenUP separa os papéis nos seguintes grupos de pessoas.

1. **Analista:** é o responsável por entender as necessidades do usuário final e priorizar os requisitos. Ele é responsável também por fazer a interlocução entre a equipe interna e os clientes.

2. **Arquiteto:** é o responsável por definir a arquitetura de *software*. Também é sua função tomar decisões técnicas sobre o design e a implementação do sistema.
3. **Desenvolvedores:** são os responsáveis em transformar os dados levantados em pontos funcionais do sistema.
4. **Gerente do projeto:** é o responsável por coordenar as iterações com os clientes e manter a equipe focada nos objetivos.
5. **Stakeholders:** são todos os interessados no projeto.
6. **Testador:** é o responsável por identificar, conduzir, implementar, registrar e analisar os testes visando à qualidade do *software*.

## Prováveis problemas das metodologias ágeis

As principais causas de falha na adoção de uma metodologia ágil são:

- filosofia ou cultura da empresa que conflitam com os valores e princípios do *agile*;
- falta de suporte gerencial para apoiar as mudanças, pois é preciso desejar as mudanças;
- falta de experiência ou treinamento insuficiente no novo processo;
- é preciso tornar-se capaz de trabalhar de maneira ágil;
- boicote e falta de comprometimento da própria equipe;
- é preciso reconhecer que há espaço para melhorias e desejar-las.

No Quadro 2, estão listadas as semelhanças entre as metodologias Scrum e XP.

**Quadro 2.** Comparativo entre as metodologias Scrum e XP

Scrum	XP
Gestão	Engenharia
<i>Sprint</i>	<i>Iteration</i>
<i>Sprint planning</i>	<i>Iteration planning</i>
<i>Daily scrum</i>	<i>Stand-up meeting</i>
<i>Retrospective</i>	<i>Reflection</i>



## Referências

KROLL, P.; KRUCHTEN, P. *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP*. London: Addison-Wesley Professional, 2003.

MEIRA, F. L. *Introdução ao processo unificado aberto*. 01 abr. 2010. Disponível em: <<http://open2up.blogspot.com/>>. Acesso em: 26 out. 2018.

MUHSINZODA, M. *Kanban agile planning with Burn-Down chart*. 01 dez. 2015. Disponível em: <<https://blogs.deusto.es/master-informatica/kanban-agile-planning-with-burn-down-chart/>>. Acesso em: 26 out. 2018.

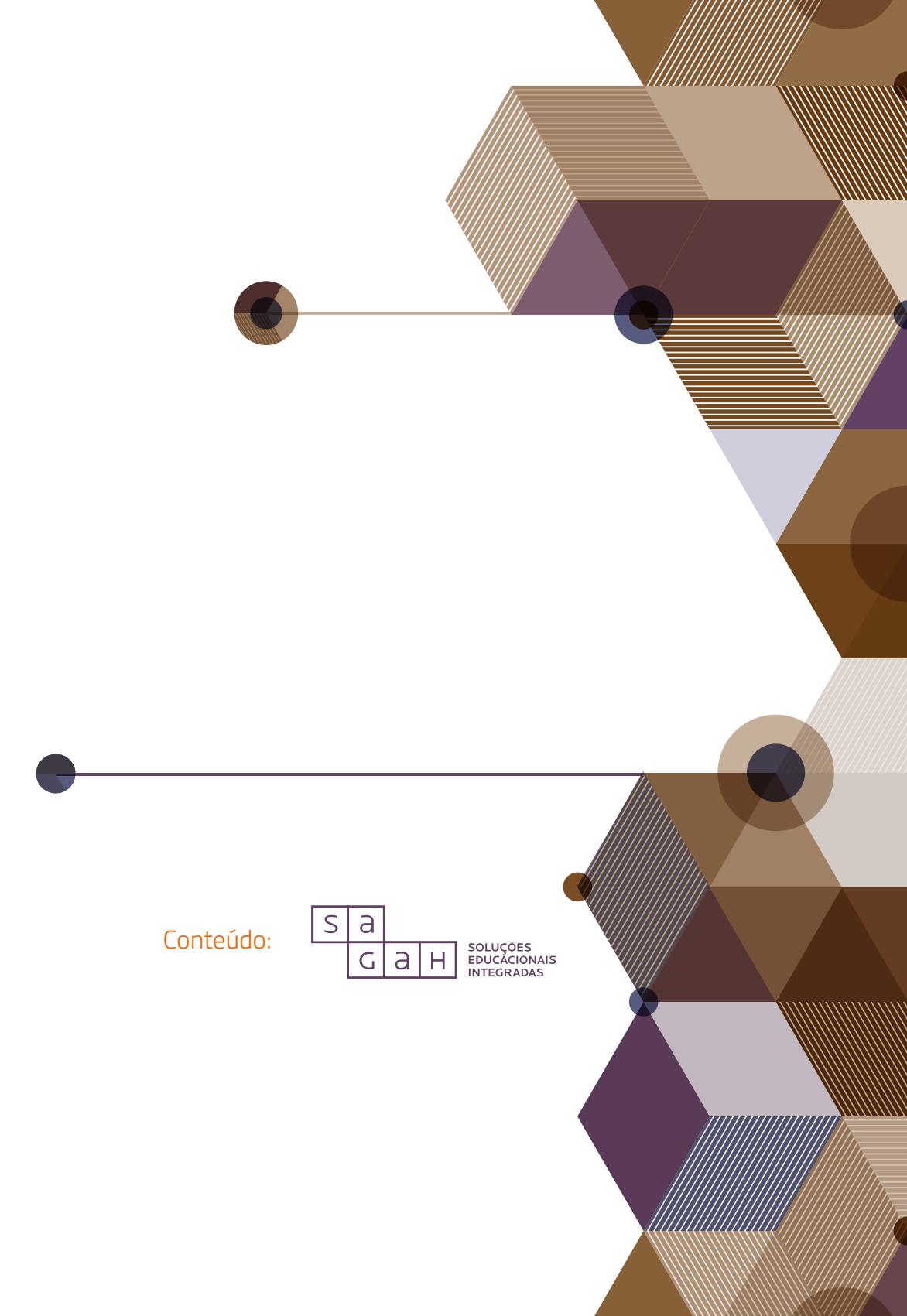
PRESSMAN, R. S.; MAXIM, B. R. *Engenharia de software: uma abordagem profissional*. 8. ed. Porto Alegre: Penso, 2016.

SOMMERVILLE, I. *Engenharia de Software*. 9. ed. São Paulo: Pearson, 2011.

## Leitura recomendada

STEFFEN, J. B. *O que são essas tais de metodologias Ágeis?* 23 jan. 2012. Disponível em: <[https://www.ibm.com/developerworks/community/blogs/rationalbrasil/entry/mas\\_o\\_que\\_s\\_c3\\_a3o\\_essas\\_ta%C3%ADs\\_de\\_metodologias\\_\\_c3\\_a1geis?lang=en](https://www.ibm.com/developerworks/community/blogs/rationalbrasil/entry/mas_o_que_s_c3_a3o_essas_ta%C3%ADs_de_metodologias__c3_a1geis?lang=en)>. Acesso em: 19 out. 2018.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.



Conteúdo:



SOLUÇÕES  
EDUCACIONAIS  
INTEGRADAS



# Dica do Professor

---

Metodologias são recursos disponíveis para que o gestor escolha de acordo com as necessidades do seu projeto. Geralmente, essa escolha se dá após análises e verificação da metodologia que mais se enquadra e não de preferência pessoal.

A Dica do Professor a seguir mostra a diferença entre metodologias tradicionais e ágeis para o desenvolvimento de software e faz um breve comparativo entre a metodologia SCRUM e a XP.



Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

# Exercícios

1) São princípios dos métodos ágeis:

- A) Aceitação de mudanças e maior ênfase nos processos em detrimento das pessoas.
- B) Rejeição de mudanças e envolvimento dos clientes.
- C) Foco na simplicidade e maximização da documentação formal.
- D) Entrega contínua ao usuário e maior ênfase nas pessoas em detrimento dos processos.
- E) Maximização da documentação formal e envolvimento dos clientes.

2) Quais são os princípios da Extreming Programming (XP), método ágil relacionado ao desenvolvimento de código:

- A) Comunicação, Respeito, Velocidade, Complexidade e Feedback
- B) Compartilhamento, Rapidez, Rigidez, Feedback e Simplicidade
- C) Comunicação, Respeito, Coragem, Feedback e Simplicidade
- D) Compartilhamento, Respeito, Coragem, Feedback e complexidade
- E) Respeito, Coragem, Feedback, ousadia e Simplicidade

3) Desenvolver um software de forma rápida é uma necessidade nos tempos atuais. Os métodos ágeis possibilitam que um software seja desenvolvido rapidamente. Analise as opções abaixo e assinale a que representa um dos princípios desse método.

- A) Entregar o projeto apenas quando estiver concluído.
- B) Envolver o cliente.
- C) Prescrever os processos.
- D) Entender o software por completo.

- E) Evitar mudanças.
- 4) **Analise as alternativas abaixo e assinale a alternativa correta sobre Processos de software.**
- A) Processos de software são passos imprevisíveis para o desenvolvimento do software.
- B) Um processo de software são um conjunto de atividades previsíveis que levam à produção de software de acordo com a necessidade do cliente.
- C) Trata-se de um único processo, que será levado até o final do projeto.
- D) O processo fornece um conjunto de informações técnicas que definem as tarefas do desenvolvimento de software.
- E) Mesmo que o processo esteja incorreto, será possível chegar ao resultado esperado.
- 5) **Um processo de software é um conjunto de atividades e resultados associados que levam à produção de um software.**
- Dentre essas atividades, existem as fundamentais comuns a todos. Leia as opções abaixo e assinale as que representam essas atividades comuns.**
- A) Projeto, implementação, validação, evolução e integração de software.
- B) Especificação, validação, evolução e integração de software.
- C) Projeto, implementação, integração e validação de software.
- D) Especificação, estimativa de custo, projeto, validação e evolução de software.
- E) Especificação, projeto, implementação, validação e evolução de software.

## Na prática

Metodologias ágeis, como o próprio nome diz, são utilizadas para dar agilidade nos processos de Desenvolvimento de software.

Com a utilização do SCRUM é possível evoluir no projeto utilizando incrementos no sistema. Veja neste **Na Prática** um exemplo de como funciona esse processo e suas etapas.

# METODOLOGIAS ÁGEIS - SCRUM



Na utilização dos métodos ágeis, o projeto passa basicamente pelas mesmas etapas dos métodos tradicionais, porém de forma cílica. As etapas acontecem em iterações e as entregas são feitas de acordo com a necessidade do momento.

**Utilizando o Scrum.** Isso é feito de forma **Iterativa e Incremental.**



As mesmas etapas são repetidas até se chegar ao resultado desejado. O software vai sendo construído, avaliado e ajustado com o intuito de melhorá-lo.

**Desenvolvimento (Incremento)** - significa melhorar gradativamente, ou seja, adicionar partes novas. É adicionar novas partes ao longo do tempo.

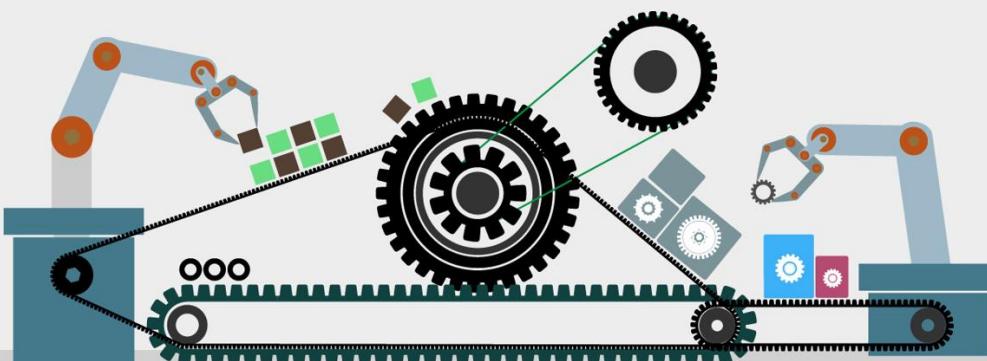


O produto é feito aos poucos e entregue constantemente. Dessa forma, toda mudança é bem-vinda, pois o projeto está em desenvolvimento.

A cada iteração, um novo incremento é feito, e isso se repete até que o software fique pronto por **completo**.

## EXEMPLO:

Em um **sistema de vendas**, a parte do cadastro de clientes seria desenvolvida - incremento - primeiro com as funcionalidades básicas e evoluiria - iterações - com as outras partes do software, priorizando sempre o que é mais importante para o cliente.



Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

# Saiba mais

Para ampliar o seu conhecimento a respeito desse assunto, veja abaixo as sugestões do professor:

## **Guia da metodologia ágil e SCRUM para iniciantes**

Leia o guia da metodologia ágil e SCRUM para iniciantes e aprenda mais sobre metodologias ágeis e o modelo SCRUM.



Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

## **Agile Development: XP e SCRUM em uma abordagem comparativa**

Leia o artigo Desenvolvimento ágil e SCRUM em uma abordagem comparativa e aprenda mais sobre desenvolvimento ágil e suas particularidades.



Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

## **OpenUp - processo unificado aberto**

Leia o artigo OpenUp - processo unificado aberto e aprenda como utilizar o método ágil OpenUp.



Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.