



APRENDIZAGEM EM FOCO

CONCEITOS GERAIS E PRINCIPAIS ABORDAGENS DE DESENVOLVIMENTO DE SOFTWARE



APRESENTAÇÃO DA DISCIPLINA

Autoria: Thiago Salhab Alves


Leitura crítica: Anderson Paulo Avila Santos

A disciplina **Conceitos gerais e principais abordagens de desenvolvimento de software** apresenta os principais elementos para o desenvolvimento de software, sendo a engenharia de software e os modelos de desenvolvimento de software clássicos e ágeis os principais objetos de estudo da disciplina. Vamos conhecer os conteúdos que serão explorados em nossas aulas:

No Tema 1 serão trabalhados os conceitos sobre engenharia de software, ciclo de desenvolvimento de software, com as principais atividades, assim como o papel do engenheiro de software no processo de desenvolvimento e modelos de certificação de software, tais como o CMMI (*Capability Maturity Model Integration*), que é um modelo internacional de qualidade e certificação de sistemas. Com isso você irá aplicar os conceitos centrais da Engenharia de Software.

No Tema 2 serão trabalhadas as metodologias clássicas, que são as primeiras metodologias propostas para criar um processo de desenvolvimento de sistema, com foco nas metodologias cascata, prototipação, baseado em componentes, incremental e espiral. Dessa forma, você compreenderá a importância de desenvolver softwares que atendam às necessidades dos usuários.

No Tema 3 serão trabalhadas as metodologias ágeis para desenvolvimento de sistemas, metodologias que visam uma construção mais focada em comunicação, feedback constante e simplicidade do projeto, com foco nas metodologias *Extreme Programming* (XP) e *Feature Driven Development* (FDD). Assim, você



compreenderá os princípios e fundamentos das metodologias ágeis.

Para finalizar, no Tema 4 será trabalhado a metodologia Scrum, seus princípios, fundamentos básicos, papéis, artefatos e cerimônias, com foco no desenvolvimento de aplicativos complexos, gerenciamento e controle do desenvolvimento do projeto e autogerenciamento de equipes. Com isso, você conhecerá os principais componentes do processo de desenvolvimento de software ágil com Scrum..

INTRODUÇÃO

Olá, aluno (a)! A *Aprendizagem em Foco* visa destacar, de maneira direta e assertiva, os principais conceitos inerentes à temática abordada na disciplina. Além disso, também pretende provocar reflexões que estimulem a aplicação da teoria na prática profissional. Vem conosco!

TEMA 1

Conceitos gerais da engenharia de software

Autoria: Thiago Salhab Alves

Leitura crítica: Anderson Paulo Avila Santos






DIRETO AO PONTO

Segundo Sommerville (2011), até 1960 as máquinas eram “grandes”, ocupando uma sala de dezenas de metros. Em 1971 surgiu o primeiro microprocessador em silício: o Intel 4004, uma CPU de 4 bits com menos de 3 mil transistores.

Com a mudança tecnológica houve um aumento dramático na produção de software. Em um breve período, os recursos de hardware aumentaram muito e permitiram que produtos mais complexos fossem criados. Seria como se engenheiros civis, depois de anos construindo apenas casas ou pequenos prédios, se vissem na tarefa de construir grandes arranha-céus. Ocorre assim a Crise do Software. Os principais problemas apontados foram:

- Projetos abandonados.
- Programas que não funcionam.
- Programas que não atendem aos requisitos.
- Programas que apresentam defeitos e falham constantemente.
- Módulos que não funcionam corretamente quando integrados.
- Programas que não fazem o esperado.

Os problemas apresentados são os mesmos encontrados atualmente? Sim, isto mostra que muitas organizações são imaturas para o processo de desenvolvimento de softwares. A maturidade avalia o grau de competência, gerência e técnica, que a organização possui para produzir software de qualidade, dentro de prazos e custos razoáveis e previstos. São considerados,



segundo Sommerville (2011), sintomas de imaturidade do processo de desenvolvimento de sistemas:

- Projetos não são definidos com clareza. Atividades de desenvolvimento de software são disfarçadas de manutenção.
- As pessoas não recebem treinamento necessário: ou não existe disponibilidade de tempo para treinamento, ou as pessoas se inscrevem no treinamento que bem entendem.
- Os procedimentos e padrões, quando existem, são definidos e seguidos de forma burocrática.

Como proposta para solucionar os problemas da crise de software, na Conferência de 1968 é proposto o conceito de engenharia de software, com o objetivo de aplicar técnicas e métodos para controlar o desenvolvimento de software. Essas técnicas se tornaram a engenharia de software.

Segundo Pressman (2016), a engenharia de software atua em um processo, com um conjunto de métodos e ferramentas, para permitir que os profissionais possam desenvolver software de altíssima qualidade. A importância da engenharia de software é a capacitação para o desenvolvimento de sistemas complexos dentro do prazo e com qualidade, visando minimizar o ambiente caótico de desenvolvimento.

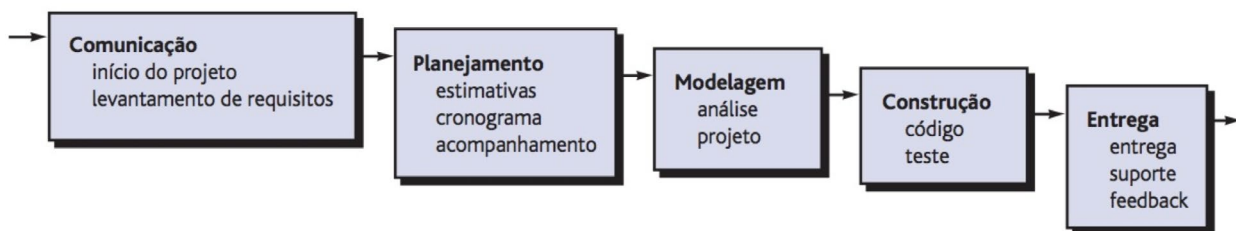
Para se criar um software de computador é necessário aplicar um processo que conduza a um resultado de alta qualidade e atendendo às necessidades daqueles que usarão o produto, aplicando-se assim uma abordagem de engenharia de software.

De acordo com Pressman (2016), para que a engenharia de software tenha um bom resultado, é necessário criar um processo

de desenvolvimento, que são atividades, ações e tarefas que serão realizadas na criação de algum produto. Para se aplicar o processo de desenvolvimento, uma metodologia deve ser utilizada.

Ela estabelece o alicerce para um processo de engenharia de software, conforme apresentado na Figura 1, compreendendo algumas atividades:

Figura 1 – Atividades do processo de Engenharia de Software



Fonte: PRESSMAN (2016, p. 42).

- **Comunicação:** antes de iniciar o trabalho técnico, mantenha comunicação e colaboração com cliente, buscando entender os objetivos dos envolvidos para o projeto e levantar os requisitos que vão ajudar a definir o software.
- **Planejamento:** o planejamento cria um “mapa” que vai ajudar a guiar a equipe no seu trabalho que é denominado plano do projeto, definindo o trabalho de engenharia de software, descrevendo as tarefas técnicas, os riscos, recursos necessários e produtos resultantes.
- **Modelagem:** crie um esboço para se ter ideia do todo, qual será o seu aspecto em termos de arquitetura, criando modelos para se entender as necessidades do software e o projeto que vai atender a essas necessidades.
- **Construção:** o que se projeta deve ser construído, combinando a geração de código e testes para revelar erros na codificação.

- Entrega: o software é entregue ao cliente, que irá avaliar o produto entregue e fornece feedback.

Por meio do processo de engenharia de software, utilizando as atividades de comunicação, planejamento, modelagem, construção e entrega, é possível manter a qualidade do software e minimizar os problemas relacionados à imaturidade do processo de desenvolvimento de sistemas.

Referências bibliográficas

PRESSMAN, R. **Engenharia de software**: uma abordagem profissional. 8. ed. Porto Alegre: AMGH, 2016.

SOMMERVILLE, I. **Engenharia de software**. 9. ed. Pearson Education do Brasil, 2011.

PARA SABER MAIS

Segundo Pressman (2016), as atividades de metodologia do processo de engenharia de software são complementadas por atividades de apoio. Essas atividades são aplicadas em todo projeto de software e têm por objetivo auxiliar a equipe a gerenciar e controlar o andamento, a qualidade, as alterações e riscos do projeto. São atividades de apoio:

- Controle e acompanhamento do projeto: permitem que a equipe acompanhe o progresso do projeto frente ao plano do projeto e adote medidas necessárias para cumprir o cronograma.

- Administração de riscos: avalia os possíveis riscos que possam impactar o resultado ou a qualidade do produto ou projeto.
- Garantia da qualidade de software: são definidas e utilizadas atividades que vão garantir a qualidade do software.
- Revisões técnicas: são avaliados os artefatos da engenharia de software, buscando identificar e eliminar erros, antes que se propaguem para a atividade seguinte.
- Medição: define e coleta medidas do processo, projeto e produto, auxiliando na entrega do software com base nos requisitos.
- Gerenciamento da configuração do software: gerencia efeitos de mudanças do processo.
- Gerenciamento da capacidade de reutilização: define critérios para reutilizar artefatos e define mecanismos para obtenção de componentes reutilizáveis.
- Produção de artefatos de software: são atividades necessárias para criar artefatos, tais como, modelos, documentos, logs, formulários e listas.

Cada uma dessas atividades de apoio visa o gerenciamento e controle do projeto, proporcionando maior qualidade e controle de possíveis riscos relacionados ao projeto.

Referências bibliográficas

PRESSMAN, R. **Engenharia de software**: uma abordagem profissional. 8. ed. Porto Alegre: AMGH, 2016.



TEORIA EM PRÁTICA

Refleta sobre a seguinte situação: você foi contratado para auxiliar uma empresa de desenvolvimento de sistemas que vem apresentando muitas dificuldades relacionadas à qualidade de seus produtos e controle do andamento de seus projetos. Os programas não têm funcionado corretamente e não atendem aos requisitos, o que gera grande reclamação por parte dos clientes. Os clientes reportaram que os programas falham constantemente e que os módulos não têm funcionado corretamente, o que indica a presença de defeitos e falhas no projeto. Nesse sentido, a empresa acabou perdendo muitos contratos e os analistas e desenvolvedores estão bem desmotivados e o dia a dia de trabalho tem se tornado muito estressante. Dessa forma, você como engenheiro de software, quais propostas apresentaria para recuperação dessa empresa? Faça as propostas e apresente-as, em forma de relatório, à direção da empresa.

Para conhecer a resolução comentada proposta pelo professor, acesse a videoaula deste *Teoria em Prática* no ambiente de aprendizagem.

LEITURA FUNDAMENTAL

Indicações de leitura

Indicação 1

Este artigo teve como objetivo compreender o processo de implementação de um sistema de informação de uma instituição pública de saúde, por meio de um estudo de caso com base no conceito de ciclo de vida dos sistemas de informação. Para realizar

a leitura, acesse a plataforma Biblioteca Virtual da Kroton/ EBSCO HOST, busque por artigos e depois pelo título da obra.

LEHNHART, E. dos R. et al. Ciclo de vida dos sistemas: uma análise dos desafios da implementação de um sistema de informação em uma instituição de saúde pública. **Rev. Adm. UFSM**, Santa Maria, v. 10, n. 4, p. 474-591, out.-dez. 2017.

Indicação 2

Este artigo teve como objetivo apresentar as principais abordagens do *Capability Maturity Model Integration* (CMMI) e as melhores práticas para gestão de projeto propostas pelo *Project Management Institute* (PMI) por intermédio do *Project Management Body of Knowledge* (PMBOK), com intuito de compará-los e levantar requisitos para a eficácia no gerenciamento de projetos de software. Para realizar a leitura, acesse a plataforma Biblioteca Virtual da Kroton/ EBSCO HOST, busque por artigos e depois pelo título da obra.

PINTO, E. B.; VASCONCELOS, A. M.; LEZANA, A. G. R. Abordagens do PMBOK e CMMI sobre o sucesso dos projetos de softwares. **Revista de Gestão e Projetos** – GeP, v. 5, n. 1, jan./abr. 2014.

QUIZ

Prezado aluno, as questões do Quiz têm como propósito a verificação de leitura dos itens *Direto ao Ponto, Para Saber Mais, Teoria em Prática e Leitura Fundamental*, presentes neste *Aprendizagem em Foco*.

Para as avaliações virtuais e presenciais, as questões serão elaboradas a partir de todos os itens do *Aprendizagem em Foco* e dos slides usados para a gravação das videoaulas, além de questões de interpretação com embasamento no cabeçalho da questão.

1. A engenharia de software visa alinhar e coordenar o processo de desenvolvimento de software. Neste contexto, complete as lacunas da sentença a seguir:

Engenharia de software atua em um _____ de desenvolvimento de software, com um conjunto de _____ e _____, para permitir que os profissionais possam desenvolver softwares de altíssima qualidade.

Assinale a alternativa que completa adequadamente as lacunas:.

- a. Processo; Métodos; Ferramentas.
- b. Projeto; Métodos; Ferramentas.
- c. Processo; Pessoas; Ferramentas.
- d. Processo; Pessoas; Estratégias.
- e. Projeto; Métodos; Estratégias.

2. Para que a engenharia de software tenha um bom resultado, é necessário criar um processo de desenvolvimento, que são atividades, ações e tarefas que serão realizadas na criação de algum produto. Nesse contexto, assinale a alternativa que apresenta a sequência correta de etapas relacionados ao processo de engenharia de software:

- a. Planejamento – comunicação – modelagem – construção – entrega.
- b. Comunicação – planejamento – modelagem – construção – entrega.
- c. Modelagem - comunicação – planejamento – construção – entrega.
- d. Construção - comunicação – planejamento – modelagem – entrega.
- e. Comunicação – construção - planejamento – modelagem – entrega.



GABARITO

Questão 1 - Resposta A

Resolução: A engenharia de software visa alinhar e coordenar o processo de desenvolvimento de software. Sendo assim, a engenharia de software atua em um processo de desenvolvimento de software, com um conjunto de métodos e ferramentas, para permitir que os profissionais possam desenvolver softwares de altíssima qualidade.

Questão 2 - Resposta B

Resolução: São consideradas as sequências de etapas do processo de engenharia de software a comunicação, planejamento, modelagem, construção e entrega.

TEMA 2

Metodologias clássicas

Autoria: Thiago Salhab Alves

Leitura crítica: Anderson Paulo Avila Santos





DIRETO AO PONTO


Na engenharia de software processos são atividades para a construção de um produto de software, tais como, desenvolvimento, manutenção, aquisição e contratação de software. Dentro de um processo de desenvolvimento de software, o primeiro passo é a escolha de um modelo de ciclo de vida (PAULA FILHO, 2019).

No modelo de ciclo de vida cascata, as atividades são executadas em sequência, podendo ser demarcados como pontos de controle, que facilitam a gestão dos projetos, fazendo com que o processo seja confiável e utilizável em projetos de qualquer escala.

O modelo cascata, segundo Paula Filho (2019), é considerado muito burocrático e rígido, pois não prevê a correção posterior de problemas nas fases anteriores e também apresenta baixa visibilidade do projeto ao cliente, que só recebe o produto no final do processo.

Um modelo considerado radicalmente diferente do modelo cascata é o espiral. Nesse modelo, o produto é desenvolvido em uma série de iterações, e cada iteração representa uma volta na espiral. Quando se tiver uma versão operacional do produto, este é submetido à avaliação dos usuários, permitindo assim construir produtos em prazos curtos, com novas características, que são adicionadas à medida que o usuário descobre suas necessidades. O ciclo de vida espiral permite que os requisitos sejam definidos de forma progressiva, apresentando alta flexibilidade e visibilidade (PAULA FILHO, 2019).

Uma variante do modelo espiral, por Paula Filho (2019), é a prototipagem evolucionária. Nesse modelo são construídas



versões provisórias chamadas de protótipos, que atendem cada vez mais requisitos até se ter um produto desejado. É possível que se tenha modelos mistos, como a entrega evolutiva, em que as atividades de especificação do problema são executadas em cascata e as atividades restantes são executadas em espiral.

O desenvolvimento baseado em componentes trata o sistema final como vários pequenos sistemas, permitindo avaliar os componentes disponíveis com foco em apenas uma funcionalidade ou um conjunto de funcionalidades semelhantes, denominadas componentes. O desenvolvimento baseado em componentes traz uma série de benefícios:

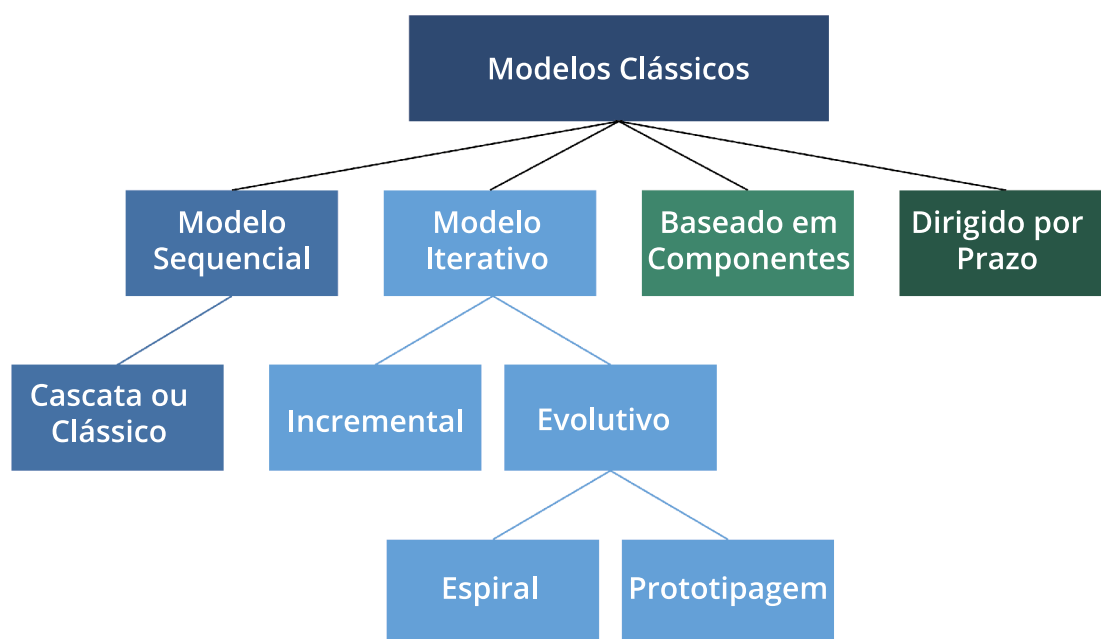
- Produtividade: economia de tempo dependendo do portfólio de componentes já prontos.
- Robustez: maior qualidade no produto final, pois já foram largamente testados em um projeto dedicado à construção deles.
- Padrão de desenvolvimento: equipe segue um padrão de desenvolvimento.

O modelo de processo incremental aplica sequências lineares (como no modelo cascata) de forma escalonada, à medida que o tempo for avançando. Cada uma das sequências lineares gera um incremento do software. Esses incrementos são entregáveis e prontos para o cliente.

Outro modelo de ciclo de vida é o dirigido por prazo (*time-boxed*), em que o produto é construído dentro de um determinado prazo. O modelo pode ser aceitável desde que se consiga definir um conjunto de requisitos indispensável dentro de um prazo suficiente para sua implementação, tendo na pior hipótese um produto com resultado parcial (PAULA FILHO, 2019).

A Figura 1 ilustra a organização dos modelos clássicos. Como modelo sequencial, há o modelo cascata ou clássico. Como modelos iterativos, os principais modelos são o incremental e evolutivo. Como modelos evolutivos, os principais são o espiral e a prototipagem. Os outros modelos clássicos estudados são o baseado em componentes e o dirigido por prazo.

Figura 1 – Organização dos modelos clássicos




Fonte: elaborado pelo autor.

Referências bibliográficas

PAULA FILHO, W. P. **Engenharia de software: produtos**. 4. ed. Rio de Janeiro: Editora LTC, 2019.

PARA SABER MAIS

A especificação de software ou engenharia de requisitos é o processo para compreender e definir quais serviços são




necessários e identificar as restrições de operação e de desenvolvimento do sistema (SOMMERVILLE, 2011).

A engenharia de requisitos é um estágio particularmente crítico do processo de software, pois os erros conduzem inevitavelmente a problemas posteriores no projeto e na implementação do sistema.

Existem quatro fases principais no processo de engenharia de requisitos (PRESSMAN, 2016):

- Estudo de viabilidade: é feita uma avaliação para verificar se as necessidades dos usuários identificadas podem ser satisfeitas por meio das tecnologias atuais de software e hardware. O estudo considera se o sistema proposto terá custo adequado do ponto de vista comercial e se pode ser desenvolvido.
- Elicitação e análise de requisitos: é o processo de derivação de requisitos de sistema por meio da observação de sistemas existentes, discussões com usuários potenciais e compradores. Pode envolver o desenvolvimento de um ou mais modelos de sistema e protótipos.
- Especificação de requisitos: atividade de traduzir as informações coletadas durante a atividade de análise em um documento que define um conjunto de requisitos.
- Validação de requisitos: essa atividade verifica os requisitos em relação ao realismo, consistência e abrangência. Durante esse processo, erros no documento de requisitos são inevitavelmente descobertos. Devem, então, ser feitas modificações para corrigir esses problemas.

Esse processo leva à produção de um documento de requisitos, que é a especificação do sistema. O processo de engenharia de requisitos se inicia com o estudo de viabilidade, que irá verificar a



real necessidade do sistema e se há tecnologia disponível para o seu desenvolvimento. Considere, por exemplo, o desenvolvimento de um sistema para controle de estoque para uma loja de roupas. Uma vez que o estudo de viabilidade aponta que o desenvolvimento é viável, passa-se para a etapa de elicitação dos requisitos, que é um processo de descoberta dos requisitos, realizado junto ao proprietário ou responsável pela loja de roupas, para entendimento das necessidades.

Na fase de análise, esses requisitos vão ser analisados pela equipe de engenharia de software, buscando abstrair e determinar os requisitos funcionais (requisitos que estão relacionados à funcionalidade do sistema, por exemplo, cadastro de produtos, cadastro de fornecedores, venda de produtos etc.) e os requisitos não funcionais (requisitos relacionados à interface com usuário, backup de dados, segurança, portabilidade etc.).

De posse dos requisitos analisados, é possível criar um documento de especificação dos requisitos, que deve conter a lista detalhada dos requisitos (funcionais e não funcionais), requisitos de usuário e sistema. Na fase de especificação, são gerados também diagramas que fornecem informações detalhadas do funcionamento do sistema, tais como, diagrama de casos de uso (que detalha a funcionalidade do sistema), diagrama de classe (que apresenta as classes e seus relacionamentos), diagrama de sequência (que apresenta a sequência de troca de mensagens entre classes), dentre outros. Esses documentos devem passar pela validação, tanto da equipe de projeto quando do proprietário da loja de roupas, e, então, gerar o documento de requisitos final do sistema.

O projeto e implementação de software é o processo de conversão de uma especificação de um sistema em um sistema executável. Um projeto de software é a descrição da estrutura de software

a ser implementada, dos dados que são partes do sistema, das interfaces entre os componentes do sistema (SOMMERVILLE, 2011).

Referências bibliográficas

PRESSMAN, R. **Engenharia de software**: uma abordagem profissional. 8. ed. Porto Alegre: AMGH, 2016.

SOMMERVILLE, I. **Engenharia de software**. 9. ed. Pearson Education do Brasil, 2011.

TEORIA EM PRÁTICA

Refleta sobre a seguinte situação: você foi promovido a engenheiro de software em sua organização e tem a difícil tarefa de orientar a mudança do atual processo de desenvolvimento de sistemas, com o objetivo de melhorar os produtos e a aceitação por parte dos clientes. A empresa trabalha com o desenvolvimento de sistemas para diversos setores comerciais, o que faz com que os requisitos mudem constantemente. Muitos dos sistemas desenvolvidos atualmente apresentam atrasos e muitos erros de funcionamento. Dessa forma, uma vez que você conheceu vários modelos clássicos de desenvolvimento de sistemas, quais metodologias de desenvolvimento você indicaria para serem aplicadas no dia a dia de trabalho da organização? Faça as propostas e apresente-as, em forma de relatório, à direção da empresa.

Para conhecer a resolução comentada proposta pelo professor, acesse a videoaula deste *Teoria em Prática* no ambiente de aprendizagem.



LEITURA FUNDAMENTAL

Indicações de leitura

Indicação 1


Este artigo teve como objetivo apresentar a atividade de elicitação de requisitos, considerada uma das atividades mais difíceis do processo de engenharia de software, realizando um mapeamento sobre projetos de desenvolvimento de software no setor público com objetivo de identificar as lições aprendidas que impactam na atividade de elicitação. Para realizar a leitura, acesse a plataforma Biblioteca Virtual da Kroton/ EBSCO HOST, busque por artigos e depois pelo título da obra.

COSTA, L. A.; ZOUCCAS, A. C.; ALVES, J. B. M. Elicitação de requisitos de software no setor público: lições aprendidas e recomendações para mitigação de riscos. **Revista Brasileira de Administração Científica**, Aquidabã, v. 3, n. 2, p. 214-226, 2012.

Indicação 2

Este artigo teve como objetivo apresentar a proposta de um modelo baseado no modelo de software espiral, denominado iSprial, para auxiliar alunos para transformar suas ideias em projetos. Para realizar a leitura, acesse a plataforma Biblioteca Virtual da Kroton/ EBSCO HOST, busque por artigos e depois pelo título da obra.

ATOUM, I. A spiral software engineering model to inspire innovation and creativity of university students. **International Journal of Engineering Pedagogy**. 2019, Vol. 9, Issue 5, p. 7-23. 17p.



QUIZ

Prezado aluno, as questões do Quiz têm como propósito a verificação de leitura dos itens *Direto ao Ponto, Para Saber Mais, Teoria em Prática e Leitura Fundamental*, presentes neste *Aprendizagem em Foco*.

Para as avaliações virtuais e presenciais, as questões serão elaboradas a partir de todos os itens do *Aprendizagem em Foco* e dos slides usados para a gravação das videoaulas, além de questões de interpretação com embasamento no cabeçalho da questão.

1. A engenharia de software é a área da engenharia que visa o desenvolvimento de sistemas. Neste contexto, complete as lacunas da sentença a seguir:
A especificação de software ou _____ é o processo para _____ e definição de quais serviços são _____ e identificar as restrições de operação e de desenvolvimento do sistema.
Assinale a alternativa que completa adequadamente as lacunas:.
 - a. Engenharia de requisitos; Compreensão; Necessários.
 - b. Engenharia de software; Programação; Avaliados.
 - c. Engenharia de sistemas; Desenvolvimento; Testados.
 - d. Engenharia de computação; Teste; Programados.
 - e. Engenharia de sistemas; Avaliação; Compreendidos.
2. A engenharia de requisitos é um estágio particularmente crítico do processo de software, pois os erros conduzem inevitavelmente a problemas posteriores no projeto e na

implementação do sistema. Nesse contexto, assinale a alternativa que apresenta a sequência correta de etapas relacionadas ao processo de engenharia de requisitos:

- a. Elicitação e análise de requisitos - estudo de viabilidade – especificação de requisitos – validação de requisitos.
- b. Estudo de viabilidade – especificação de requisitos - elicitação e análise de requisitos – validação de requisitos.
- c. Estudo de viabilidade – elicitação e análise de requisitos – especificação de requisitos – validação de requisitos.
- d. Especificação de requisitos - estudo de viabilidade – elicitação e análise de requisitos – validação de requisitos.
- e. Estudo de viabilidade – elicitação e análise de requisitos – validação de requisitos - especificação de requisitos.



GABARITO

Questão 1 - Resposta A

Resolução: A especificação do software ou a engenharia de requisitos é uma área da engenharia de software que compreende e define quais serviços são necessários e identifica as restrições de operação e desenvolvimento do sistema.

Questão 2 - Resposta C

Resolução: A sequência correta de etapas da engenharia de requisitos são o estudo de viabilidade (verificando a necessidade e possibilidade tecnológica para o desenvolvimento do sistema), elicitação e análise de requisitos (processo de descoberta e análise dos requisitos), especificação de requisitos (lista descritiva de requisitos funcionais e não funcionais) e validação de requisitos (tudo o que foi analisado deve estar especificado para avaliação do cliente).

TEMA 3

Metodologias ágeis

Autoria: Thiago Salhab Alves

Leitura crítica: Anderson Paulo Avila Santos






DIRETO AO PONTO

O desenvolvimento de software sempre foi considerado uma atividade complexa e caótica, principalmente se o software é desenvolvido sem um plano definido, alinhado a decisões de curto prazo que são tomadas ao longo do seu desenvolvimento. A ausência de um planejamento formal pode até dar certo quando o software a ser desenvolvido é de baixa complexidade (SBROCCO; MACEDO, 2012).

Considerando que novas funcionalidades podem ser agregadas, acaba se tornando mais difícil gerenciar a inclusão desses novos requisitos sem um planejamento adequado. Neste cenário, normalmente percebe-se que comportamentos indesejados se tornam frequentes e são cada vez mais difíceis de serem eliminados.

Portanto, podemos afirmar que o uso de metodologias, independentemente de quais sejam, apresenta grandes benefícios para as organizações. A principal motivação no uso de processos disciplinados é que eles têm o objetivo de tornar o desenvolvimento mais previsível e eficiente. As metodologias ágeis surgiram em resposta a metodologias tradicionais mais “engessadas”, proporcionando mudanças de ênfase significativas e estimulando pequenos grupos de desenvolvedores a adotar um processo disciplinado para projetos de qualquer natureza.

Uma das grandes motivações, que tem atraído cada vez mais profissionais da área de tecnologia da informação e também de outras áreas, deve-se ao fato das metodologias ágeis serem menos centradas em documentações, sendo mais voltadas ao código-fonte do sistema de software (BECK; FOWLER, 2000).



Contudo, não é conveniente associar apenas essa característica quando se pensa em metodologias ágeis, pois existem outras diferenças tão impactantes quanto esta. Uma delas é o fato de que as metodologias ágeis são adaptativas, em vez de predeterminantes.

Isso se deve ao fato de que as metodologias propostas pela engenharia de software tradicional tendem a planejar grande parte do processo de desenvolvimento por um longo período de tempo, prática que funciona bem até que haja a necessidade de mudar o projeto. Neste caso, a tendência é resistir a essas mudanças. Já as metodologias ágeis aceitam mudanças ao longo do desenvolvimento de maneira natural, adaptando seus processos às mudanças exigidas.

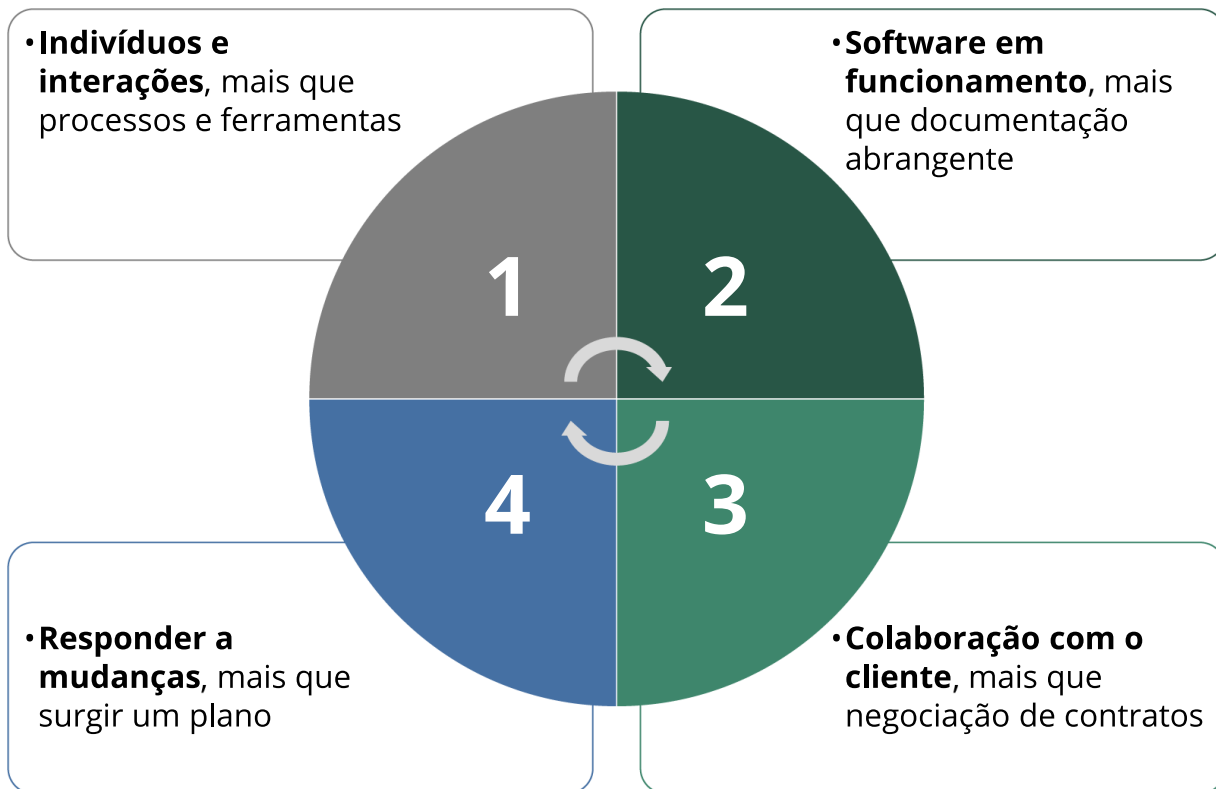
Outro aspecto muito importante que diferencia as metodologias ágeis das tradicionais é que os métodos ágeis utilizados são orientados a pessoas, e não a processos. Na engenharia de software tradicional, o objetivo dos métodos adotados é definir processos que possam funcionar bem independentemente de quem os estiver usando (SBROCCO; MACEDO, 2012).

Os métodos ágeis pregam que nenhum processo pode ser equivalente à habilidade dos integrantes da equipe de desenvolvimento. Assim, os processos existentes nos métodos ágeis têm o papel de dar suporte à equipe de desenvolvimento.

O manifesto ágil foi criado em 2001 em uma reunião com 17 profissionais que já utilizavam as metodologias ágeis como XP, DSDM, Scrum e FDD. Nesta reunião foram observados fatores de sucesso das metodologias já utilizadas e criado um manifesto com base nesses pontos. Os valores do manifesto ágil são os indivíduos e interações, mais que processos e ferramentas, software em funcionamento mais que documentação, colaboração do cliente

mais que negociação de contratos e resposta a mudanças mais que seguir um plano, ilustrados pela Figura 1.

Figura 1 – Princípios do manifesto ágil



Fonte: elaborado pelo autor

O *Extreme Programming* (XP) é uma metodologia de desenvolvimento ágil com foco na agilidade de equipes, utilizando valores como simplicidade, comunicação e coragem. São boas práticas de XP: cliente sempre presente, metáfora no projeto, planejando o jogo, pequenas versões, testes de aceitação, integração contínua, simplicidade de projeto, refatoração, programação em pares, propriedade coletiva e padronização do código.

O FDD (*Feature Driven Development*) busca realizar o desenvolvimento por funcionalidade, por meio de um requisito funcional do sistema, apresentando cinco processos básicos:

desenvolvimento de modelo abrangente (análise orientada por objetos), construção de lista de funcionalidades (decomposição funcional), planejar por funcionalidade (planejamento incremental), detalhe por funcionalidade (projeto orientado a objetos) e construção por funcionalidade (programação e teste orientado a objetos).

Referências bibliográficas

BECK, K.; FOWLER, M. **Planning Extreme Programming**. 1. ed. Boston: Addison Wesley, 2000.


SBROCCO, J. H. T.; MACEDO, P. C. **Metodologias ágeis: engenharia de software sob medida**. 1. ed. São Paulo: Érica, 2012.



PARA SABER MAIS

Uma outra metodologia que faz parte dos métodos de desenvolvimento ágil é o *Dynamic Systems Development Methodology* (DSDM). Segundo Sbrocco e Macedo (2012), DSDM é uma metodologia de desenvolvimento de projetos de software que visa estabelecer recursos e tempo fixo para o desenvolvimento de um projeto, ajustando funcionalidades de maneira a atender prazos estipulados. Criada em 1994, no Reino Unido, por uma organização sem fins lucrativos e não proprietária, que hoje é mantida por uma associação de empresas e órgãos públicos internacionais. A estrutura do DSDM baseia-se em nove princípios que são considerados boas práticas dessa metodologia:

1. Participação ativa dos usuários e *stakeholders*: todos os envolvidos no projeto e que conhecem o processo devem

- 
- acompanhar o desenvolvimento para garantir que tudo seja entregue a tempo e de acordo com o solicitado.
2. Abordagem cooperativa e compartilhada: todas as partes interessadas devem cooperar e assumir o compromisso das entregas de partes do software e escolher a ordem de sua implementação.
 3. Equipes com poder de decisão: as pessoas envolvidas no desenvolvimento devem ter conhecimento e autonomia para decidir o destino do sistema, não sendo tolerável aguardar decisões por longo período.
 4. Entregas contínuas: busca trazer retorno ao cliente do projeto desde o início, com o intuito de promover a validação do que está sendo feito.
 5. Desenvolvimento iterativo e incremental: melhorar o sistema continuamente usando um processo iterativo, visando corrigir problemas o mais cedo possível.
 6. Feedback: foco nas entregas de produtos de software, permitindo que o usuário manifeste suas opiniões e solicite modificações.
 7. Todas as alterações durante o desenvolvimento devem ser reversíveis: as modificações podem ser testadas e, caso não apresentem o resultado esperado, elas podem ser revertidas.
 8. Fixar os requisitos essenciais: os requisitos principais devem ser obtidos logo no início do projeto para se garantir os objetivos gerais.
 9. Teste em todo ciclo de vida: devido ao prazo apertado, não realizar os testes apenas no final da implementação, estes devem ser realizados em todas as fases e componentes do projeto.

A DSDM, detalhada por Sbrocco e Macelo (2012), apresenta cinco fases distintas para aplicação de sua metodologia:

1. Estudo de viabilidade: realizado apenas uma vez durante o projeto, buscando analisar sua viabilidade. Nessa fase, observa-se a organização com um todo, listando pessoas envolvidas, técnicas a serem utilizadas e ferramentas de trabalho.
2. Estudo de negócio: as regras de negócio são analisadas, bem como todos os processos envolvidos, visando capturar as características do negócio. Reuniões com *stakeholders* para se chegar a um acordo sobre as prioridades de desenvolvimento e definir um plano geral de desenvolvimento devem ser realizados.
3. Modelo de iteração funcional: planejamento de conteúdo e abordagem aplicado após a execução de cada iteração para analisar se os resultados estão corretos. As funcionalidades são analisadas e implementadas, resultando em um protótipo que serve como modelo de experiência para melhoria de todo o processo de iteração.
4. Projeto e construção de iteração: fase em que o sistema é implementado, sendo a saída dessa fase um sistema testado e validado com os requisitos iniciais, de acordo com as necessidades dos usuários. Os envolvidos devem revisar o sistema e comentar seus resultados, dando feedback aos desenvolvedores.
5. Implementação: fase de colocar o sistema em funcionamento no ambiente real, treinando usuários para usar o sistema e analisar os resultados do uso do sistema.

Referências bibliográficas

SBROCCO, J. H. T.; MACEDO, P. C. **Metodologias ágeis**: engenharia de software sob medida. 1. ed. São Paulo: Érica, 2012.



TEORIA EM PRÁTICA

Refleta sobre a seguinte situação: você foi promovido a engenheiro de software em sua organização e tem a difícil tarefa de orientar a mudança do atual processo de desenvolvimento clássico de sistemas, com o objetivo de agilizar o desenvolvimento dos produtos e a aceitação por parte dos clientes. A empresa trabalha com o desenvolvimento de sistemas para diversos setores comerciais, o que faz com que os requisitos mudem constantemente. Dessa forma, uma vez que você conheceu vários modelos ágeis de desenvolvimento de sistemas, qual(is) metodologia(s) de desenvolvimento de desenvolvimento ágil você indicaria para ser aplicado no dia a dia de trabalho da organização? Faça as propostas e apresente-as, em forma de relatório, à direção da empresa.

Para conhecer a resolução comentada proposta pelo professor, acesse a videoaula deste *Teoria em Prática* no ambiente de aprendizagem.

LEITURA FUNDAMENTAL

Indicações de leitura

Indicação 1

Este artigo tem como objetivo apresentar como aumentar o conhecimento da utilização, de forma efetiva, das técnicas XP, para que o sistema que está sendo desenvolvido atenda ao ambiente tecnológico acelerado, permitindo que os desenvolvedores respondam rapidamente a requisitos inovadores e variáveis. Para

realizar a leitura, acesse a plataforma Biblioteca Virtual da Kroton/ EBSCO HOST, busque por artigos e depois pelo título da obra.

FRUHLING, A.; DE VREEDE, G. J. Field experiences with eXtreme programming: developing an emergency response system. **Journal of Management Information Systems**, [s. l.], v. 22, n. 4, p. 39–68, 2006.

Indicação 2

Este artigo tem como objetivo investigar os fatores que afetam a conscientização dos profissionais ágeis na implementação de práticas ágeis. Para realizar a leitura, acesse a plataforma Biblioteca Virtual da Kroton/ EBSCO HOST, busque por artigos e depois pelo título da obra.

SULAIMAN, N. L.; MAHRIN, M. N.; YUSOFF, R. C. M. Influential factors on the awareness of agile software development methodology: a systematic literature review. **Journal of Korean Society for Internet Information**, [s. l.], v. 17, n. 5, p. 161–172, 2016.



QUIZ

Prezado aluno, as questões do Quiz têm como propósito a verificação de leitura dos itens *Direto ao Ponto, Para Saber Mais, Teoria em Prática e Leitura Fundamental*, presentes neste *Aprendizagem em Foco*.

Para as avaliações virtuais e presenciais, as questões serão elaboradas a partir de todos os itens do *Aprendizagem em Foco* e dos slides usados para a gravação das videoaulas,

além de questões de interpretação com embasamento no cabeçalho da questão.

1. Sobre as metodologias ágeis, complete as lacunas da sentença a seguir:

O que tem atraído cada vez mais profissionais da área de tecnologia da informação se deve ao fato que as metodologias ágeis são _____ concentradas em _____, sendo mais voltadas ao _____ do sistema.

Assinale a alternativa que completa adequadamente as lacunas:

- a. Menos; Documentações; Código-fonte.
 - b. Mais; Análise; Documento.
 - c. Menos; Código-fonte; Documento.
 - d. Mais; Testes; Uso.
 - e. Menos; Projeto; Código-fonte.
-
2. O DSDM foi criado em 1994, no Reino Unido, por uma organização sem fins lucrativos e não proprietária que reúne membros como Hewlett Packard, British Airways, Vodafone e Ministério da Defesa. A estrutura do DSDM é composta de nove princípios, considerados boas práticas. Nesse contexto, assinale a alternativa que apresenta algumas das práticas que pertencem ao DSDM:
- a. Participação ativa dos usuários e *stakeholders* - análise de riscos - abordagem cooperativa e compartilhada - entregas contínuas.
 - b. Equipes com poder de decisão – desenvolvimento iterativo e incremental – feedback – entrega final.

- c. Equipes com poder de decisão – desenvolvimento iterativo e incremental – fixar os requisitos essenciais – análise de riscos.
- d. Fixar os requisitos essenciais - teste em todo ciclo de vida - equipes com poder de decisão – programação em pares.
- e. Participação ativa dos usuários e *stakeholders* - análise de riscos - abordagem cooperativa e compartilhada - todas as alterações durante o desenvolvimento devem ser reversíveis.



GABARITO

Questão 1 - Resposta A

Resolução: Uma das grandes motivações, que tem atraído cada vez mais profissionais da área de tecnologia da informação e também de outras áreas, deve-se ao fato de as metodologias ágeis serem menos centradas em documentações, sendo mais voltadas ao código-fonte do sistema de software.

Questão 2 - Resposta E

Resolução: Algumas das práticas que pertencem ao DSDM são: participação ativa dos usuários e *stakeholders* - análise de riscos - abordagem cooperativa e compartilhada - todas as alterações durante o desenvolvimento devem ser reversíveis. As análises de riscos, entrega final e programação em pares não fazer parte do DSDM.

TEMA 4

Metodologia Scrum

Autoria: Thiago Salhab Alves

Leitura crítica: Anderson Paulo Avila Santos





DIRETO AO PONTO

Scrum, por Pressman (2016), é um método de desenvolvimento ágil de software, criado por Jeff Sutherland no início dos anos 1990. Apresenta total aderência aos princípios do manifesto ágil e é usado para orientar as atividades de desenvolvimento dentro de um processo que possui as seguintes atividades metodológicas: requisitos, análise, projeto, evolução e entrega.

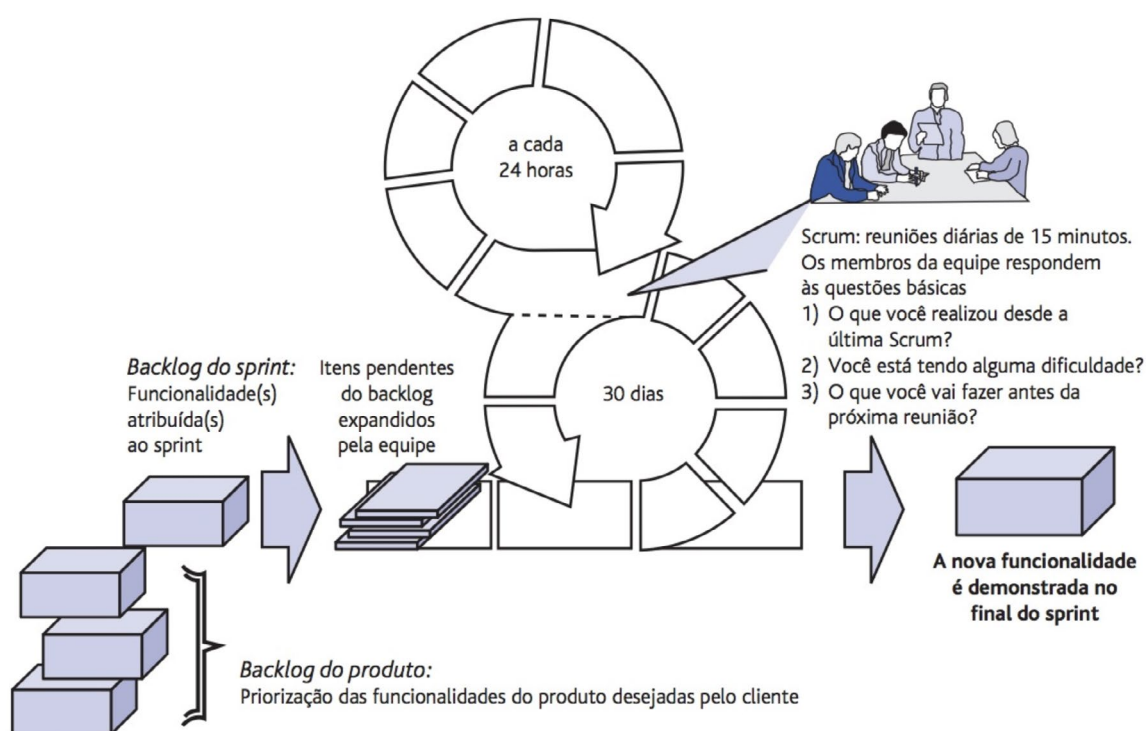
Cada atividade metodológica é constituída de atividades que são realizadas dentro de um padrão de processo chamado Sprint. O trabalho realizado dentro da Sprint é adaptado aos problemas e definido pela equipe Scrum. A Figura 1 ilustra o fluxo geral do processo Scrum.

Scrum utiliza um conjunto de padrões de processo de software que se mostra ser muito eficaz em projetos com prazos de entrega curtos e com requisitos que mudam constantemente. Cada um desses padrões de processos apresentam um conjunto de atividades de desenvolvimento:

- **Product Backlog:** lista com prioridades dos requisitos ou funcionalidades do projeto que agregam valor ao cliente. Os itens podem ser adicionados a esse registro a qualquer momento. O gerente do produto avalia o registro e atualiza as prioridades.
- **Sprints:** são unidades de trabalho solicitadas para atingir um requisito estabelecido no backlog e que precisa ser atendido dentro de um prazo (tipicamente de 30 dias). Não se é permitido alterar, por exemplo, itens do registro de trabalho, que não foram introduzidos durante a execução.

- Reuniões Scrum: reunião curtas, de aproximadamente quinze minutos, realizadas diariamente pela equipe Scrum, em que três perguntas-chave são respondidas:
 - O que você realizou desde a última reunião de equipe?
 - Quais obstáculos está encontrando?
 - O que planeja realizar até a próxima reunião da equipe?
- Scrum master: líder da equipe responsável por conduzir a reunião e avaliar as respostas de cada integrante. A reunião Scrum é realizada diariamente e auxilia a equipe a revelar problemas em potencial.

Figura 1 – Fundamentos básicos do Scrum



Fonte: Presman (2016, p. 78).

Referências bibliográficas

PRESSMAN, R. **Engenharia de software**: uma abordagem profissional. 8. ed. Porto Alegre: AMGH, 2016.


PARA SABER MAIS

O Scrum master desempenha um papel essencial dentro da metodologia Scrum, sendo o responsável pelas práticas e valores. Dessa forma, certificações Scrum, principalmente para se atuar como Scrum master, são muito procuradas e desejadas.

Uma certificação Scrum master mostra que o profissional tem domínio das práticas e dos processos do Scrum, estando apto para atuar com um time e vem a ser um profissional valorizado por organizações da área de tecnologia da informação.

Diversas são as empresas que proporcionam certificação Scrum master e cada uma possui características próprias, mas que visam preparar o profissional para atuar nessa função. No processo, deve-se estudar um programa de treinamento e ser aprovado em um exame, já que algumas empresas acabam até por exigir que seja realizado um curso oficial. Algumas das empresas certificadoras são:

- Scrum Alliance: empresa fundada pelos criadores do Scrum, oferecendo treinamento e certificações. A certificação mais conhecida é a CSM (*Certified Scrum Master*). Para se obter essa certificação, o profissional deve realizar um curso oficial da Scrum Alliance, com duração de dois dias. Após a realização do curso, o profissional realiza um exame, ao



custo de US\$ 250,00 e que deve ser renovado a cada dois anos.

- Scrum.org: criado por um dos criadores do Scrum Alliance, apresenta ao público algumas opções de certificações e que estão agrupadas em categorias. As principais certificações são:
 - Professional Scrum Master I (intermediário): disponível para profissionais que desejam validar profundamente o conhecimento no framework Scrum. Investimento de US\$ 150,00, sem pré-requisito, com avaliação com 80 questões, no formato de múltipla escolha e utilizando o material Guia de Scrum.
 - Professional Scrum Master II (avançado): disponível a qualquer pessoa que passou no exame Professional Scrum Master I e pretende demonstrar sua capacidade para aplicar o Scrum em problemas avançados. Investimento de US\$ 250,00, com avaliação de 30 questões, no formato de múltipla escolha.
 - Professional Scrum Master III: necessário alto nível de conhecimento Scrum e experiência em profundidade antes de realizar a avaliação, sendo considerado difícil e com perguntas de estudo de caso e ensaios. Investimento de US\$ 500,00.
- Exin: reconhecida pela certificação ITIL, a empresa lançou certificações para a metodologia ágil e Scrum, sendo a Agile Scrum Foundations (ASF) e a Agile Scrum Master

(ASM). O custo do exame é de US\$ 200,00 para ASF e US\$ 300,00 para ASM.

Referências bibliográficas

METODOLOGIA AGIL. **Certificação Scrum Master**. Disponível em: <http://metodologiaagil.com/certificacao-scrum-master/>. Acesso em: 20 maio 2020.

TEORIA EM PRÁTICA

Refleta sobre a seguinte situação: você recebeu a proposta de auxiliar uma empresa de desenvolvimento de sistemas, que utiliza metodologias clássicas, a implantar a metodologia ágil Scrum, pois o principal objetivo da direção é mudar o atual processo de desenvolvimento clássico de sistemas, com o objetivo de agilizar o desenvolvimento dos produtos e a aceitação por parte dos clientes. A empresa trabalha com o desenvolvimento de sistemas para diversos setores comerciais, o que faz com que os requisitos mudem constantemente. O gerente da sua empresa disse estar interessado na utilização do Scrum. Dessa forma, uma vez que você já conheceu a metodologia Scrum, faça uma apresentação à direção da empresa sobre o funcionamento do Scrum, seus componentes e suas principais vantagens sobre as metodologias clássicas.

Para conhecer a resolução comentada proposta pelo professor, acesse a videoaula deste *Teoria em Prática* no ambiente de aprendizagem.



LEITURA FUNDAMENTAL

Indicações de leitura

Indicação 1

Este artigo tem como objetivo analisar a influência do papel do Scrum master no desenvolvimento de projetos Scrum. Para realizar a leitura, acesse a plataforma Biblioteca Virtual da Kroton/ EBSCO HOST, busque por artigos e depois pelo título da obra.

RAMOS, A. B.; VILELA JUNIOR, D. C. A influência do papel do Scrum master no desenvolvimento de projetos Scrum. **Revista de Gestão e Projetos**, [s. l.], v. 8, n. 3, p. 80–99, 2017.

Indicação 2

Este artigo tem como objetivo apresentar os principais pontos de dificuldade na adoção do método ágil Scrum em empresas que usam processos *plan-driven*. Para realizar a leitura, acesse a plataforma Biblioteca Virtual da Kroton/ EBSCO HOST, busque por artigos e depois pelo título da obra.

DE ASSIS, D. M.; LARIEIRA, C. L. C.; COSTA, I. As dificuldades na adoção e uso de método Scrum em empresas brasileiras utilizando processos plan-driven: estudo de caso múltiplo. **Revista de Gestão e Projetos**, [s. l.], v. 8, n. 3, p. 66–79, 2017.



QUIZ

Prezado aluno, as questões do Quiz têm como propósito a verificação de leitura dos itens *Direto ao Ponto, Para Saber Mais, Teoria em Prática e Leitura Fundamental*, presentes neste *Aprendizagem em Foco*.

Para as avaliações virtuais e presenciais, as questões serão elaboradas a partir de todos os itens do *Aprendizagem em Foco* e dos slides usados para a gravação das videoaulas, além de questões de interpretação com embasamento no cabeçalho da questão.

1. Sobre o Scrum, complete as lacunas da sentença a seguir:
Scrum utiliza um conjunto de _____ de software que se mostrou ser muito eficaz em projetos com _____ e com requisitos que _____ constantemente.
Assinale a alternativa que completa adequadamente as lacunas:.
 - a. Padrões de processo; Prazos de entrega curtos; Mudam.
 - b. Processos; Prazos de entrega longos; Variam.
 - c. Ferramentas; Prazos de entrega variáveis; Alteram.
 - d. Padrões; Prazos de entrega curtos; Não mudam.
 - e. Elementos; Prazos de entrega longos; Não variam.

2. Scrum é um método de desenvolvimento de software, criado por Jeff Sutherland no início dos anos 1990. Nesse contexto, assinale a alternativa que apresenta com quem o Scrum tem total aderência de princípios:

- a. Engenharia de software.
- b. Engenharia de requisitos.
- c. Manifesto ágil.
- d. Metodologia ágil.
- e. Metodologia clássica.



GABARITO

Questão 1 - Resposta A

Resolução: O Scrum é uma metodologia de desenvolvimento que utiliza um conjunto de padrões de processo de software que se mostrou ser muito eficaz em projetos com prazos de entrega curtos e com requisitos que mudam constantemente.

Questão 2 - Resposta C

Resolução: Scrum é uma metodologia ágil de desenvolvimento de software, criado por Jeff Sutherland no início dos anos 1990, que possui total aderência aos princípios do manifesto ágil.



BONS ESTUDOS!