



APRENDIZAGEM EM FOCO

EVOLUÇÃO DOS SOFTWARES: APLICAÇÃO DA ENGENHARIA DE SOFTWARE EM SISTEMAS EMERGENTES



APRESENTAÇÃO DA DISCIPLINA

Autoria: Anderson da Silva Marcolino

Leitura crítica: Aline Chagas Rodrigues Marques

Assim como os engenheiros civis precisam de projetos arquiteturais para guiar uma obra, o desenvolvimento de software também necessita de projetos. Além disso, considerando a complexidade da implementação de um software com qualidade, é necessário integrar diferentes elementos que possibilitem alcançar uma melhor solução, além de reduzir a complexidade e os custos relacionados. Nesta perspectiva, a Engenharia de Software, como área e disciplina, busca concentrar padrões, métodos e processos que auxiliem a tomada de decisões no momento de se criar um software. Inclui-se, ainda, o apoio no gerenciamento dos recursos humanos, financeiros e de tempo, que resultam em projetos de sucesso ou insucesso. Deste modo, o papel do engenheiro de software é fundamental, definindo que métodos e padrões serão utilizados em quais processos ou etapas, e como estes se relacionarão. Ademais, especificar-se-á os artefatos, ou seja, os documentos ou partes e funcionalidades de software, casos de testes e seus resultados, entre outros, que serão criados e mantidos no decorrer da evolução de um produto. Destarte, conhecer a engenharia de software e o que a compõe é fundamental para qualquer profissional da área de tecnologia da informação, pois permitirá diferenciar-se dos concorrentes que acabam por não dinamizar a implementação de produtos robustos e de qualidade, quando comparados àqueles que conhecem os métodos, padrões e processos provenientes de tal disciplina e área. Pronto para esta jornada de aperfeiçoamento por meio da engenharia de software? Bons estudos!



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

Evolução e melhorias dos processos da engenharia de software

Autoria: Anderson da Silva Marcolino

Leitura crítica: Aline Chagas Rodrigues Marques





DIRETO AO PONTO

O termo engenharia de software foi cunhado pela primeira vez em 1968, durante a crise do software, resultante da complexidade da criação de soluções mediante o desenvolvimento de hardware mais poderoso que, para ser utilizado com toda sua potencialidade, exigia a criação de soluções compatíveis. Abordagens *ad hoc* para o desenvolvimento de sistemas não eram mais efetivas, visto que resultavam em projetos atrasados, custos elevados, softwares desprovidos de confiança, difíceis de receberem manutenção, além de desempenho ruim.

A engenharia de software surge como uma disciplina da engenharia na área da tecnologia da informação (TI) que provê apoio ao desenvolvimento de software ou sistemas de computação, por meio de métodos de especificação, desenvolvimento e manutenção, com a aplicação de tecnologias e práticas de gerência de projetos, economia, ética e outras áreas, objetivando a organização, produtividade e qualidade do produto final.

Diferentemente da engenharia de sistemas, que objetiva atender a todos os aspectos de desenvolvimento de sistemas computacionais, incluindo a engenharia de hardware, de software e processos, a engenharia de software foca-se especificamente nos processos de concepção de software, sendo estabelecidos os quatro pilares a seguir:

- **Ferramentas:** abrange software e aplicações que apoiam os métodos e processos, facilitando a condução das atividades.
- **Métodos:** referem-se à formalização das atividades e tarefas que englobam cada processo.

- **Processo:** refere-se à sistematização dos métodos para que o software atenda ao fim para o qual foi criado.
- **Qualidade:** obtida por meio da união dos esforços humanos com ferramentas, métodos e processos. É o que define um bom produto.

Como responsável principal por aplicar a engenharia de software e suas técnicas está o engenheiro de software que, conhecedor das diferentes abordagens sistemáticas e organizadas da disciplina, seleciona e aplica-as, com a utilização e o apoio de ferramentas e técnicas ou métodos apropriados para solucionar um ou mais problemas por meio de produtos de software.

Nesta perspectiva, a engenharia de software se preocupa com todos os aspectos da produção de uma solução, com objetivo de se obter um produto de qualidade e bem-sucedido. Portanto, pode-se definir um software quando atende às necessidades das pessoas que o usam e das solicitações das partes envolvidas, também chamados de *stakeholders*.

Entre tais softwares, têm-se os conhecidos como sistemas emergentes, sendo caracterizados por utilizarem tecnologias web, móvel ou a mesclagem de ambas (híbridas). Todos desenvolvidos considerando elementos da computação em nuvem e técnicas de reuso, como arquitetura orientada a componentes, serviços ou microsserviços, podendo, ainda, ser uma mesclagem dessas técnicas arquiteturais. Ainda como sistemas emergentes, podem-se destacar os que utilizam aprendizagem de máquina, do termo em inglês *machine learning*, e inteligência artificial, e sistemas que atendam a uma vasta gama de clientes geograficamente distribuídos, sendo utilizados por meio de um navegador para internet ou tecnologias que encapsulam aplicações totalmente web.

O infográfico da Figura 1 apresenta as etapas ou os processos de concepção de um software e os artefatos que, integrados a este, permitem a aplicação das abordagens e técnicas da engenharia de software na concepção de produtos de software em geral, permitindo, de modo complementar, a visualização de como a disciplina está presente e é indissociável a todas as etapas.

O diagrama ilustra o ciclo de vida de desenvolvimento de software, dividido em cinco etapas principais: Análise, Projeto, Desenvolvimento, Testes e Manutenção. Cada etapa é representada por um retângulo arredondado, e os artefatos produzidos em cada etapa são mostrados como ícones de documentos. As tecnologias utilizadas são indicadas por ícones de documentos com o texto 'CLASS'.

Abordagens de Gerenciamento

O ciclo de vida é dividido em duas partes principais: **Análise** e **Projeto**, e **Desenvolvimento** e **Testes**.

Artefatos e Tecnologias:

- Análise:** Produz o **Documento de Requisitos (Funcionais e Não funcionais)**. Utiliza **Prototipação** e **Diagrama de Casos de Uso**.
- Projeto:** Produz o **Projeto de Software**. Utiliza **Diagramas da UML (Classe, Sequência e)** e **Abordagens, Técnicas e Metodologias**.
- Desenvolvimento:** Produz **Artefatos de Software e Documentação**. Utiliza **Padrões e Técnicas** e **Tecnologias**.
- Testes:** Produz o **Software em Produção**.
- Manutenção:** Não produz artefatos nem utiliza tecnologias.

Legenda:

- Artefato (Abordagem, técnicas, tecnologias)
- Etapa

Referências bibliográficas

7




PARA SABER MAIS

Você sabia que a engenharia de software engloba diferentes métodos que são aplicáveis nas diversas etapas de concepção de software por meio de ferramentas de software? Você conhece alguma delas?

Como qualquer disciplina, a Engenharia de Software visa prover diferentes métodos para apoiar a concepção de software em todos os seus processos ou etapas. Para isso, é sabido que as aplicações de tais métodos não são fáceis, o que torna necessário o uso de diferentes ferramentas de software que apoiam tais usos. Dentre elas, destacam-se as conhecidas como Engenharia de Software Assistida por Computador (CASE), do termo em inglês **Computer-Aided Software Engineering**, sistemas de manipulação de banco de dados e relacionados; ferramentas para gerência de projetos, entre outras. Destacando as ferramentas CASE, temos uma vasta gama de soluções, tanto pagas, como MagicDraw e a Astah, como gratuitas, como o diagrams.net. Esta última é facilmente integrada com o Google Drive, necessitando apenas de uma conta na plataforma da Google.

Tais ferramentas dão apoio principalmente nos processos de análise e projeto, permitindo a modelagem de diagramas que apresentam vários níveis de abstração. Diagramas com baixo nível de abstração são aqueles que apresentam maiores detalhes; os de alto nível, por sua vez, apresentam menos detalhes, ambos são complementares entre si.

Na etapa de análise, costuma-se utilizar o diagrama de casos de uso. Ele tem o propósito de auxiliar o analista e sua equipe na descoberta de requisitos e no alinhamento mais preciso das necessidades dos *stakeholders*. Este diagrama possui um nível de abstração alto.



Já na etapa de projeto, têm-se diagramas com níveis de abstração baixos, como os de classe, sequência e componentes. Os diagramas de classe apresentam os elementos chamados classes, originárias do paradigma de orientação a objetos, que permitem representar seus comportamentos, por meio dos métodos, e suas características, por meio de seus atributos. A classe, posteriormente, será programada na fase de desenvolvimento, permitindo a instanciização de diferentes objetos. Considerando tais objetos, têm-se ainda as interações e troca de mensagens destes, representadas por meio dos diagramas de sequência. Finalmente, destaca-se também o diagrama de componentes, que apresenta conjuntos de classes agrupados em elementos denominados componentes, e a interação destes com outros componentes, que podem ser chamados também de artefatos de software. São eles que passam pelos testes unitários e, posteriormente, ao serem integrados, passam por testes de integração. Ao final do desenvolvimento e dos testes, a união de tais componentes formará o produto final.

É importante destacar que todos estes diagramas fazem parte da Linguagem de Modelagem Unificada (UML) e sua importância é fundamental para as etapas iniciais em um projeto de desenvolvimento de software, uma vez que, se algum erro for cometido, será necessário o esforço para correção ou refatoração de componentes e, até mesmo, do software produzido.

Referências bibliográficas

SOMMERVILLE, I. **Engenharia de software**. Tradução Luiz Cláudio Queiroz. Revisão técnica Fábio Levy Siqueira. 10. ed. São Paulo: Pearson Education do Brasil, 2018.



TEORIA EM PRÁTICA

Ao analisarmos um problema que deve ser solucionado por meio de um software, precisamos identificar quais requisitos os clientes desejam em tal produto. Apesar de tais necessidades expressas pelos *stakeholders* serem, em sua maioria, requisitos funcionais, faz-se a identificação de requisitos de qualidade, também chamados de requisitos não funcionais.

Considerando que o levantamento de requisitos é atividade primordial para que o ciclo de concepção de software ocorra corretamente, coloque-se no papel de um engenheiro de software e crie uma lista enumerada de passos que sugeriria para equipes de analistas e projetistas para a criação de um documento de requisitos e projeto arquitetural concisos. Para cada um dos itens enumerados, redija um a dois parágrafos que detalhem e justifiquem sua escolha. Considere que a equipe de analistas e projetistas são especialistas no desenvolvimento de sistemas emergentes para internet, principalmente os de comércio eletrônico (*e-commerce*).

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 apresenta o desenvolvimento de uma ferramenta para apoiar o processo de implementação/concepção de software a

reduzir falhas na construção de sistemas. As etapas apresentadas especificam, com detalhes, como a engenharia de software é importante para a criação de produto que possa atender a diferentes domínios, inclusive a ela mesma.

Para realizar a leitura, acesse a Biblioteca Virtual da Kroton e busque pelo título da obra.

MENEZES, P. M. *et al.* A engenharia de requisitos: um caso de implementação de um sistema para engenharia de requisitos. **Revista Interfaces Científica**, Aracajú. v. 1, n. 3, p. 43-54, 2015.

Indicação 2

Este artigo apresenta o desenvolvimento de um protótipo para um software web. As etapas de desenvolvimento e o apoio que a técnica de prototipação oferece são cruciais para o engenheiro de software. Nesta perspectiva, poderão ser identificadas as principais etapas de criação de um software emergente, para uma solução no domínio de apoio ao planejamento de aposentadoria.

Para realizar a leitura, acesse a Biblioteca Virtual da Kroton e busque pelo título da obra. Após acessar a página do artigo, é possível fazer seu download no formato PDF em português.

PISSINATI, P. S. C. *et al.* Desenvolvimento de um protótipo de web software de apoio ao planejamento da aposentadoria. **Rev. Latino-Am. Enfermagem**, Ribeirão Preto, v. 27, e3169, 2019.


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 surgiu como uma disciplina da área da tecnologia da informação para suprir as necessidades das indústrias de software que não conseguiam desenvolver soluções de qualidade, mediante o aumento de complexidade no desenvolvimento de tais soluções, e acompanhar a evolução do hardware. Sobre a engenharia de software, assinale a alternativa correta.
 - a. A engenharia de software objetiva atender a todos os aspectos de desenvolvimento de sistemas computacionais.
 - b. Os quatro pilares da engenharia de software são: análise, projeto desenvolvimento e testes.
 - c. A engenharia de software se preocupa com todos os aspectos da produção de um software, com objetivo de se obter uma solução de qualidade e bem-sucedido.
 - d. Apesar de importante, as abordagens *ad hoc* são mais efetivas do que as abordagens da engenharia de software nas etapas de concepção de software.
 - e. A engenharia de software não se preocupa com a aplicação de tecnologias e práticas de gerência de projetos, economia, ética e outras áreas.

- 
2. Sobre os principais processos da concepção de um software – análise, projeto, desenvolvimento, testes e manutenção – assinale a alternativa correta.
 - a. No processo de projeto, os requisitos funcionais e não funcionais são transformados em diagramas de mais baixo nível, servindo como modelos para que possam ser desenvolvidos na próxima etapa.
 - b. No processo de testes, o software é testado para se verificar se possui falhas advindas da etapa de manutenção.
 - c. No processo de manutenção, o software sofre manutenção no sentido de ser validado, verificado e testado, logo após a etapa de projeto.
 - d. No processo de desenvolvimento, o software é programado, ou seja, codificado com base no documento de requisitos e resultados dos testes.
 - e. No processo de análise, o software é verificado para que não sejam encontrados erros, podendo, assim, ser disponibilizado em ambiente de produção.



GABARITO

Questão 1 - Resposta C

Resolução: A engenharia de software, ao contrário da engenharia de sistemas, que se preocupa tanto com software e hardware, tem seu foco somente em software, aplicando e desenvolvendo métodos que possam ser utilizados nas cinco principais etapas de concepção de soluções: análise, projeto, desenvolvimento, teste e manutenção. Seu objetivo é obter um software de qualidade e bem-sucedido, ou seja, que atenda a todas as demandas das partes envolvidas.

Questão 2 - Resposta A

Resolução: O primeiro processo é o de análise, que fornece o documento de requisitos funcionais e não funcionais ao processo de projeto, que, após criar diagramas de mais baixo nível, repassa-os para a equipe de desenvolvedores no processo de desenvolvimento. O software, após ser desenvolvido, passa para o processo de testes, em que é verificado, validado e testado, podendo, então, ser disponibilizado em ambiente de produção, e seguindo, ou não, para manutenções que possam corrigir falhas não identificadas no processo de testes ou levar a modificações para atender a novas demandas dos clientes.

TEMA 2

Engenharia de software em aplicativos para dispositivos móveis e aplicações web (WebApps)

Autoria: Anderson da Silva Marcolino

Leitura crítica: Aline Chagas Rodrigues Marques






DIRETO AO PONTO

Segundo Sommerville (2018), um aplicativo é um pacote que executa uma tarefa específica para um usuário final, sendo projetado para atender a requisitos dos usuários finais, ou seja, trata-se de um software que será executado em um dispositivo móvel, cujas tarefas farão uso das especificidades de hardware controladas e gerenciadas pela plataforma ou pelo sistema operacional, geralmente iOS ou Android. Já uma aplicação web é um software desenvolvido para ser executado por meio de um navegador, do termo em inglês *browser* (por exemplo, Google Chrome, Microsoft Edge, Safari, Firefox). Aplicações web são capazes de refletir sistemas complexos de software e, conseqüentemente, suas funcionalidades.

Com a ampliação do acesso à internet, o número de aplicações web que refletem softwares complexos tem aumentado, bem como o número de aplicativos móveis, somado ao barateamento dos dispositivos. A migração para aplicações web não exige a necessidade de uma estrutura do tipo cliente-servidor, mas, com a evolução das tecnologias e o surgimento da virtualização e contêineres, por exemplo, a centralização em um único servidor tem deixado de existir, refletindo, deste modo, no surgimento de padrões arquiteturais de serviços e microsserviços.

A engenharia de software estrutura em processos ou etapas a implementação de um produto de software. Estes processos, comumente, são divididos em análise, projeto, desenvolvimento, testes e manutenção. Em algumas literaturas, são concentrados em especificação, desenvolvimento, validação e evolução de software. Considerando as características dos aplicativos móveis e aplicações web, vejamos como




a engenharia de software pode auxiliar e facilitar o desenvolvimento de cada uma delas.

No processo de análise, o engenheiro de software apoiará o analista a identificar o problema das partes envolvidas (*stakeholders*), identificando tudo que a solução deve e não deve possuir, resultando nos requisitos funcionais – aqueles que são visíveis aos usuários finais; e os requisitos não funcionais, ou também chamados de requisitos de qualidade (por exemplo, tempo de resposta ao clicar em um link, facilidade de utilização, integração com serviços de terceiros).

Na sequência, a etapa de projeto é iniciada considerando como artefato principal o documento de requisitos. Este documento será, então, analisado e, com ajuda do engenheiro de software, o projetista desenvolverá outros diagramas da UML, sendo os principais: de classes, de sequência e componentes.

Na etapa de desenvolvimento, o engenheiro de software apoia os desenvolvedores ou programadores na seleção de padrões de desenvolvimento, dentre estes estão, principalmente, os relacionados a abordagens baseadas em reuso, que fazem uso principalmente do paradigma de desenvolvimento orientado a objetos.

Após a definição dos padrões e desenvolvimento dos artefatos de software, segue-se para a validação, verificação e os testes de software. Esta etapa é abrangente e deriva-se em diversas técnicas. Estas vão fornecer meios de garantir o atendimento à finalidade para a qual o software foi criado (SOMMERVILLE, 2018). Há dois grandes grupos para classificação das técnicas de testes, a saber: (a) técnica estrutural ou de caixa-branca, na qual testa-se o software em termos de código-fonte e as trocas de mensagens entre suas classes e componentes, nos



mais diversos fluxos, desde os básicos até os alternativos; e (b) técnica funcional ou teste de caixa-preta, em que analisa-se o software em termos de funcionalidades para garantir que os requisitos funcionais realizem o que se espera, ou seja, o que havia sido especificado e identificado junto aos *stakeholders*.

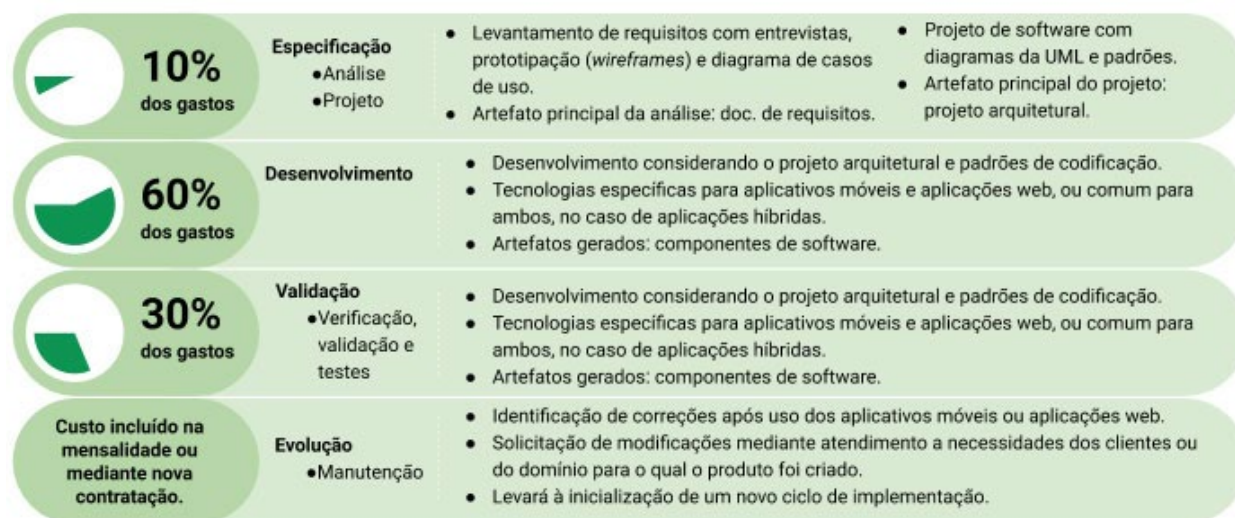
É a partir do momento que os produtos são disponibilizados para uso por todos seus usuários finais que se inicia a etapa de manutenção, na qual os usuários reportarão possíveis mudanças e correções, bem como adequações a legislações vigentes ou ao domínio ao qual a aplicação atende.

Um último ponto a se destacar é que, tal como a adoção de padrões gera transversalidade entre as diferentes etapas, temos o gerenciamento de projetos que é considerada como uma etapa extra, integrada a todas as outras, e garante a realização do cronograma esperado e assinado pelo contratante.

Destarte, com base nesta visão geral da aplicação da engenharia de software no desenvolvimento de aplicativos móveis e aplicações web, notamos sua importância e como ela definirá o sucesso ou insucesso na criação de produtos que atendam, da melhor forma possível, às demandas dos clientes.

O infográfico da Figura 1 apresenta as etapas ou os processos de concepção de um software e os artefatos que, integrados a ele, permitem a aplicação das abordagens e técnicas da engenharia de software na concepção de soluções em geral, permitindo, de modo complementar, a visualização de como a disciplina está presente e é indissociável a todas as etapas.

Figura 1 – Infográfico das atividades da engenharia de software no desenvolvimento de sistemas emergentes



Fonte: Lorem ipsum dolor sit amet.

Referências bibliográficas

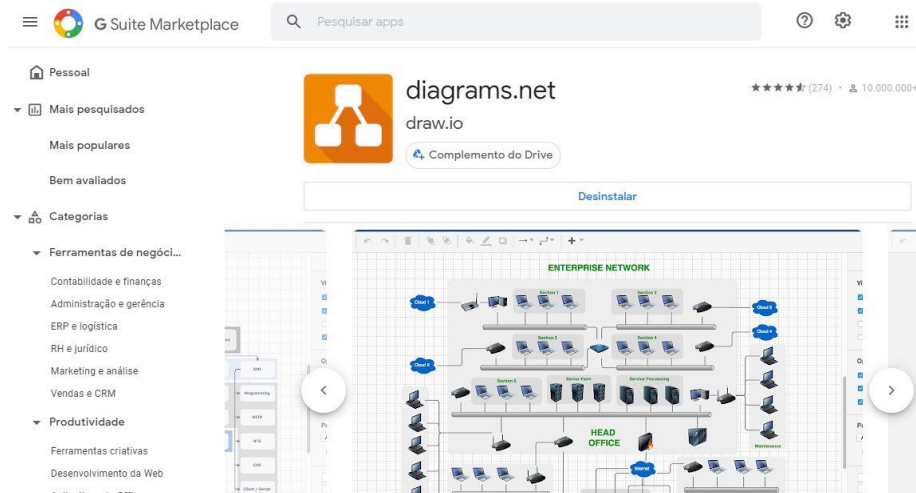
SOMMERVILLE, I. **Engenharia de software**. Tradução Luiz Cláudio Queiroz. Revisão técnica Fábio Levy Siqueira. 10. ed. São Paulo: Pearson Education do Brasil, 2018.

▶ PARA SABER MAIS

Você sabia que é possível utilizar uma ferramenta web gratuita para criar diagramas da Linguagem de Modelagem Unificada?

Ao criar uma conta do Google que lhe forneça acesso ao Google Drive, é possível incluir um complemento disponível no G Suite Marketplace para criar seus diagramas da UML. Para isso, acesse o Google Drive, procure pelo botão “Novo” ou equivalente. Neste botão, clique na opção “Mais” ou equivalente e acesse “Conectar mais apps”. Esta opção abrirá o painel de busca do G Suite Marketplace e, na caixa de busca, digite “diagrams.net”, como pode ser visto na Figura 2.

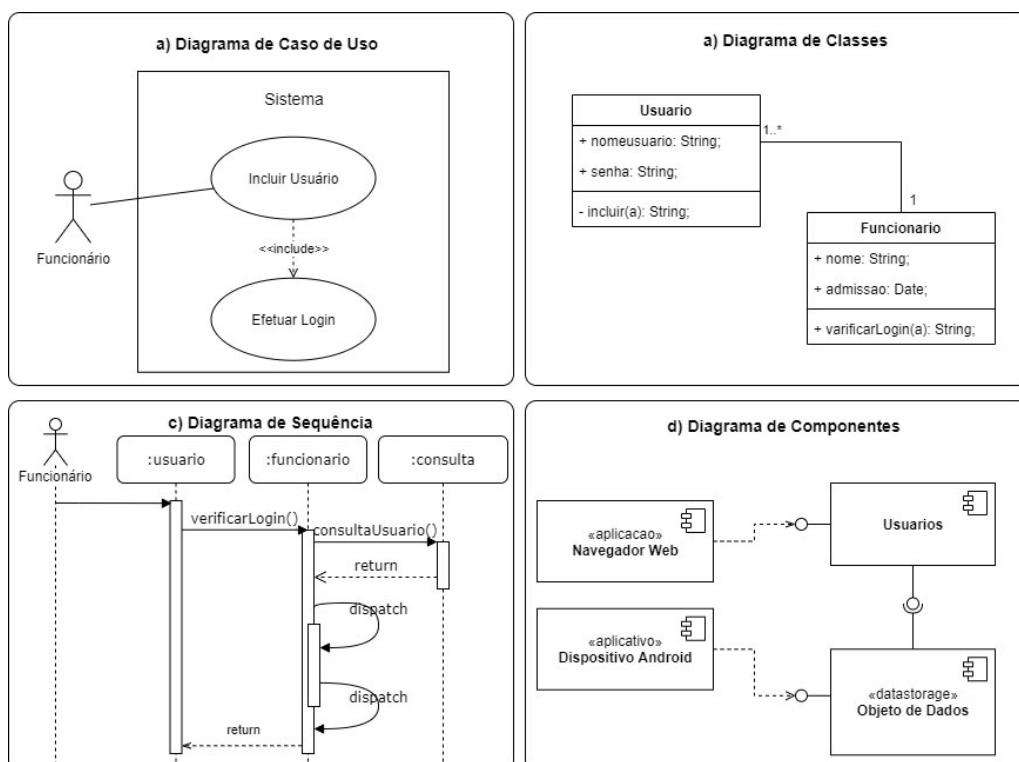
Figura 2 – Diagrams.net



Fonte: captura de tela de Diagrams.net.

Após abrir a aplicação e instalar tal complemento no Drive, você terá acesso à tela inicial, onde poderá selecionar o diagrama que deseja criar, dentre eles, há modelos já predefinidos da UML, mas novos modelos podem ser criados, incluindo diagramas de casos de uso, classe, sequência e componentes, como na Figura 2 a seguir.

Figura 2 – Exemplos de diagramas da UML



Fonte: elaborada pelo autor.

Como tais diagramas apoiam as etapas de especificação de software, é altamente recomendado que todo engenheiro de software saiba modelá-los, neste sentido, tente reproduzir os diagramas por meio desta ferramenta. Caso opte por utilizar outras ferramentas de modelagem, você pode procurar pelos termos “criação de diagramas da UML” nos mecanismos de busca da internet e encontrar diversas opções, algumas delas gratuitas.

Referências bibliográficas

SOMMERVILLE, I. **Engenharia de software**. Tradução Luiz Cláudio Queiroz. Revisão técnica Fábio Levy Siqueira. 10. ed. São Paulo: Pearson Education do Brasil, 2018.

TEORIA EM PRÁTICA

A migração de softwares tradicionais para aplicações web e aplicativos móveis trazem diversas vantagens. Considerando o cenário hipotético no qual um empresário, que recentemente abriu três filiais em diferentes estados do Brasil, deseja migrar seu sistema local para atender a suas novas filiais, coloque-se no papel de um engenheiro de software e elenque as principais vantagens da migração desse sistema para: (a) uma aplicação web e (b) para um aplicativo móvel. Após a listagem das vantagens, considere-as para escolher e indicar qual das soluções pode ser recomendada ao empresário.

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 texto em português de Portugal apresenta um *framework*, ou seja, um arcabouço de desenvolvimento de aplicações web que fornecem uma visão técnica da utilização de tecnologias no contexto da etapa de desenvolvimento para apoiar o processo de implementação/concepção de software. Recomenda-se, principalmente, a leitura do capítulo 3 – “Framework”, que apresenta etapas de desenvolvimento, modelagem e outras características.

Para realizar a leitura, acesse a Biblioteca Virtual da Kroton e busque pelo título da obra.

ALVES, T. F. *Framework* de desenvolvimento de aplicações Web. Orientador: José Carlos Meireles Monteiro Metrôlho. 2014. 92 p. Dissertação (Mestrado em Desenvolvimento de Software e Sistemas Interactivos) – Instituto Politécnico de Castelo Branco, Escola Superior de Tecnologia, Castelo Branco, 2014.

Indicação 2

Esta monografia de graduação apresenta o desenvolvimento de aplicativos para dispositivos móveis utilizando Android, uma das plataformas mais utilizadas no contexto de dispositivos móveis vendidos no mundo. Adicionalmente, o trabalho integra plataforma de código aberto de prototipagem eletrônica Arduino, enriquecendo a temática apresentada por meio da possibilidade de observar como tais aplicações são flexíveis

e, ao mesmo tempo, complexas, o que justifica rigor em seu planejamento e desenvolvimento, providos pela engenharia de software. Recomenda-se, principalmente, a leitura do capítulo 1 – “Introdução”, subseção 1.2 “Sistema operacional Android” e capítulo 2 – “Princípios básicos do sistema operacional Android”.

Para realizar a leitura, acesse a Biblioteca Virtual da Kroton e busque pelo título da obra.

ALVES, R. J.; GOULART, V. S. Desenvolvimento de aplicativos para dispositivos móveis com reconhecimento de voz em português. 2014. 46 f. Monografia (Bacharelado em Engenharia de Redes de Comunicação) —Universidade de Brasília, Brasília, 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. Aplicações Android podem ser escritas na linguagem de programação Java. A ferramenta Android SDK compila, então, o código, além de todos os dados e recursos em um Android package (APK), um arquivo com extensão .apk. Um arquivo APK contém todo o conteúdo da aplicação e é utilizado pelo sistema Android para instalá-la. Considerando

o desenvolvimento e as tecnologias de um aplicativo móvel nativo e híbrido, assinale a alternativa correta:

- a. Um aplicativo nativo é desenvolvido considerando tecnologias compatíveis com a plataforma na qual será executada, o que acaba por resultar em aplicativos mais lentos.
- b. Diz-se um aplicativo nativo aquele que é desenvolvido considerando tecnologias fornecidas pelo mantenedor ou criador da plataforma em que a aplicação será executada.
- c. Um aplicativo híbrido é desenvolvido com a tecnologia fornecida pelo mantenedor ou criador da plataforma em que a aplicação será executada.
- d. Um aplicativo híbrido é desenvolvido considerando requisitos funcionais apenas, o que torna o aplicativo final lento devido ao não atendimento dos requisitos não funcionais.
- e. Um aplicativo será desenvolvido de modo nativo ou híbrido de acordo com as necessidades do cliente, sendo a escolha da tecnologia decorrente dos resultados da etapa de testes.

2. O *design pattern* intitulado Modelo Visão e Controle (MVC), do inglês *model, view, controller*, é um padrão que auxilia na divisão do código dos componentes em unidades menores, chamadas entidades. Esta estrutura propicia a criação de código estruturado e sua simplificação, já que, por meio de conceitos da orientação a objetos, utilizam-se as classes, métodos e atributos para centralizar e manter a responsabilidade sobre cada elemento. O modelo corresponde à estrutura de dados; a visão, por sua vez, apresenta as interfaces e o controle, a lógica de negócio. Sobre o uso de padrões no contexto da engenharia de software aplicada no desenvolvimento de aplicações web, assinale a alternativa correta:


- a. Os padrões de projeto, ou *design pattern*, permitem a criação de código mais limpo e organizado, favorecendo a comunicação entre os membros da equipe e, ao final, facilitam a manutenção.
- b. Os padrões de projeto, ou *design pattern*, permitem padronizar o processo de desenvolvimento, contudo, acabam ampliando a complexidade de desenvolvimento.
- c. Os padrões de projeto, ou *design pattern*, só podem ser aplicados se os stakeholders possuírem domínio sobre o padrão a ser adotado.
- d. Os padrões de projeto, ou *design pattern*, apesar de favorecerem a criação de produtos de software com maior qualidade, acabam por dificultar o desenvolvimento, já que cada membro da equipe pode adotar um padrão diferente.
- e. É responsabilidade do analista somente a identificação e definição dos padrões arquiteturais e de desenvolvimento que serão adotados.



GABARITO

Questão 1 - Resposta B

Resolução: Um aplicativo será desenvolvido de modo nativo ou híbrido de acordo com as necessidades do cliente. A escolha pela tecnologia será baseada na arquitetura adotada, bem como especificidades e indicações do próprio cliente durante a fase de análise. Um aplicativo nativo é desenvolvido considerando tecnologias fornecidas pelo mantenedor ou criador da plataforma em que a aplicação será executada, sendo, em alguns casos, mais rápidos que os híbridos, que são desenvolvidos com tecnologias de terceiros.



Contudo, nota-se que a velocidade tem sido reduzida na medida em que o hardware dos dispositivos móveis evolui.

Questão 2 - Resposta A

Resolução: Os padrões de projeto, ou *design pattern*, são adotados para facilitar a comunicação dos membros das equipes, facilitando a padronização do código e sua organização. Apesar de mais de um padrão poder ser adotado, já que existem arquiteturas, de desenvolvimento, entre outros, a escolha ocorre de acordo com todos os membros da equipe, visto que, ao selecionar muitos padrões, esta adoção resultaria em aumento da complexidade e dificuldades de comunicação. Ao final, a aplicação dos padrões também facilita a manutenção, já que um código organizado permitirá a identificação e modificação de modo mais fácil.

TEMA 3

Engenharia de software no desenvolvimento de softwares de games

Autoria: Anderson da Silva Marcolino

Leitura crítica: Aline Chagas Rodrigues Marques






DIRETO AO PONTO

Jogos digitais são apenas um dos tipos de jogos existentes. Segundo Miranda e Stadzisz (2017, p. 299), um jogo digital é uma atividade voluntária, com ou sem interesse material, com propósitos sérios ou não, sendo composta por regras e objetivos bem definidos, que são capazes de engajar os jogadores na resolução de algum conflito, variando e quantificando resultados, sendo gerenciada por um software e executada em hardware.

Logo, considerando que os jogos digitais são, ao final, um software que integra especificidades únicas, tal como soluções convencionais, requer análise, planejamento, desenvolvimento, testes e manutenção. O que é obtido por meio da engenharia de software.

Com base em problemas recorrentes, identificados por estudos ao longo da experiência no desenvolvimento de jogos digitais, padrões foram identificados e documentados na engenharia de software, para permitir o design ou “desenho” de jogos – em algumas bibliografias, design é interpretado também como projeto, mas para não gerar ambiguidades, optou-se por adotar o termo desenho. Tais padrões definidos no contexto da engenharia de software permitem o estabelecimento de soluções para problemas, fornecendo diretrizes para sua adoção.


Há 11 categorias de padrões para os jogos digitais. O nome deles fornece uma visão holística da complexidade e quantidade de elementos envolvidos, sendo estes (BJÖRK; HOLOPAINEN, 2004): padrões de elementos de jogo; de recursos e sua gestão; da informação, comunicação e apresentação; de ações e eventos; para estruturas narrativas, previsão e imersão; para interação social; para objetivos; para estruturas e objetivo; para sessões



de jogo; para mestria do jogo e equilíbrio; e padrões para repetibilidade de meta de jogos e curvas de aprendizagem.

Esses padrões e as etapas para a implementação de um jogo digital são documentadas e mantidas no documento de desenho do jogo, do termo em inglês *game design document*. Esse documento recebe todas as especificações para a criação de um jogo, bem como a definição das tecnologias adotadas, padrões de desenvolvimento, arquiteturais e, principalmente, os elementos específicos integrados a um produto de software de jogo digital, em especial:

- 1. Visão geral do jogo:** especifica, juntamente com os diagramas de casos de uso da Linguagem de Modelagem Unificada (UML), quais são os elementos que serão traduzidos para requisitos funcionais e não funcionais e resultarão, ao final, no jogo digital pronto. São essas funcionalidades que deverão ser testadas para garantir que o produto atingiu o que era esperado.
- 2. Gameplay e mecânicas de jogo:** requer gerenciamento e equipe de desenvolvimento específica, assim como o desenvolvimento dos itens relacionados aos elementos de história, personagens e suas animações, interfaces gráficas de usuário, inteligência artificial, *game art* (artes do jogo) e integração com software de terceiros (por exemplo, gerenciamento de conexões em rede, software de controles e de sensores). Nesta perspectiva, as etapas de desenvolvimento e testes são executadas em paralelo, integrando, para cada uma das suas peculiaridades, padrões e tecnologias diferentes se necessário.
- 3. Gerenciamento:** neste item, são centralizadas as metodologias e estratégias adotadas para o gerenciamento das diferentes equipes de desenvolvimento, de testes, de análise e projeto. Define-se o cronograma, as atividades



e sua granularidade. É essencial, pois, considerando a complexidade de um jogo digital e o tempo de desenvolvimento (geralmente anos para jogos complexos), a ausência de gerenciamento pode resultar no insucesso do projeto. Como metodologia a ser adotada e considerando a necessidade de finalização dos projetos de modo rápido, geralmente é adotada a metodologia ágil Scrum.

Os itens do documento de desenho do jogo são apenas os principais, e todos eles estão integrados às etapas de engenharia de software, o que permite concluir que todos os diagramas da UML adotados na etapa de projeto também devem ser modelados para apoiar amplamente o desenvolvimento. Ademais, considerando a diversidade de equipes trabalhando em paralelo nas diferentes etapas, o projeto arquitetural prima pela implementação de componentes com fraco acoplamento e baixa coesão, ou seja, atômicos e que são responsáveis por funcionalidades específicas, possibilitando o reúso para projetos futuros.

Nesta perspectiva, como resultado, temos a complexidade de unir tais componentes e testá-los. Mais uma vez, o gerenciamento torna-se fator determinante para o sucesso do jogo digital a ser produzido. Portanto, gerenciamento é ainda mais crucial no desenvolvimento de tais softwares.

O infográfico da Figura 1 apresenta as ferramentas essenciais para a criação de um jogo digital. Os itens 1 e 2 representam a ferramenta para a modelagem de modelos 3-D e o motor de jogo para a programação e criação do jogo digital, que importará os modelos 3-D, respectivamente. Os itens no quadro inferior indicam ferramentas e sites que apoiam as atividades dos itens 1 e 2.

Figura 1 – Infográfico de ferramentas e sites para desenvolver um projeto de um jogo digital



Fonte: elaborada pelo autor.

As ferramentas e os sites apresentados na Figura 1 exemplificam algumas das soluções adotadas pelas equipes de desenvolvimento em diferentes frentes no processo de implementação de um jogo digital. Dentre elas, a mais importante é a indicada no item 2, que se refere ao motor de jogo Unity, sendo a única ferramenta que requer a aquisição de licença ao criar-se um jogo digital para vendas. Contudo, esse motor de criação permite o desenvolvimento de jogos 3-D e 2-D com facilidade e possibilidade de instalação de *plugins*.

Referências bibliográficas

- MIRANDA, F. S.; STADZISZ, P. C. Jogo Digital: definição do termo. Simpósio de Brasileiro de Jogos e Entretenimento Digital, XVI., Curitiba. Anais eletrônico. Curitiba: PR, Pontifícia Universidade Católica do Paraná, p. 296-299, 2017.
- BJORK, S.; HOLOPAINEN, J. **Patterns in game design** (game development series). USA: Charles River Media, Inc., 2004.
- SALEN, K. *et al.* **Rules of play**: game design fundamentals. USA: MIT Press, 2004.



PARA SABER MAIS

Dentre as metodologias para gerenciamento de equipes de desenvolvimento, destaca-se a metodologia ágil Scrum. Mas você sabe o que é uma metodologia ágil?

As metodologias ágeis tiveram seu início com uma reunião realizada em 2001 em uma estação de esqui em Utah, nos Estados Unidos. Este encontro reuniu 17 profissionais que divulgavam metodologias “leves” para declarar pontos em comum e chegar a resultados expressivos.

Neste contexto, surgiu o Manifesto Ágil. Um conjunto de princípios fundamentados em quatro premissas que mudaram, a partir daquele encontro, os modelos de gerenciamento tradicionais. Tais premissas são:

1. Indivíduos e interações mais que processos e ferramentas.
2. Software em funcionamento mais que documentação abrangente.
3. Colaboração com o cliente mais que negociação de contratos.
4. Responder a mudanças mais que seguir um plano.

Considerando estas quatro premissas, diferentes metodologias ágeis foram criadas, com destaque para o Scrum, nome que não possui uma tradução nem um significado definido.

O Scrum provê uma alteração de foco no contexto do gerenciamento de equipes. A relação entre as partes, tanto entre membros internos do projeto quanto com os *stakeholders* (partes envolvidas) é primada. As entregas no contexto dos processos de desenvolvimento e testes ocorrem no prazo de uma a quatro semanas, no máximo, estas interações para desenvolvimento

recebem o nome de Sprint. Todos os dias ocorrem reuniões rápidas (com os participantes em pé) para identificar como se encontram as atividades que cada membro da equipe assumiu e se há dificuldades que precisam ser resolvidas, estas reuniões são chamadas *daily meeting*. Deste modo, é possível regularizar divergências e dificuldades – até mesmo as originárias da falta de experiência do colaborador – mais rapidamente.

Destaca-se, ainda, que no início da Sprint ocorre reunião para seleção do que será desenvolvido. As histórias selecionadas de um quadro chamado Product Backlog, que refletem um ou mais requisitos, são classificadas quanto à sua dificuldade, inseridas na Sprint Backlog e cada membro da equipe assume uma das tarefas. Ao final da Sprint, é realizada uma reunião que valida o artefato desenvolvido. Se aprovado, segue-se para a seleção de novas histórias do Product Backlog ou, se reprovado, retorna para o Sprint Backlog.

Agora que você sabe o que é uma metodologia ágil, acha que algum destes conceitos pode ajudá-lo também nas tarefas do dia a dia?

Referências bibliográficas

SOMMERVILLE, I. **Engenharia de software**. Tradução Luiz Cláudio Queiroz. Revisão técnica Fábio Levy Siqueira. 10. ed. São Paulo: Pearson Education do Brasil, 2018.

TEORIA EM PRÁTICA

Ao definir elementos de um jogo digital, como sua história e seus personagens, bem como qual será o objetivo principal a ser atingido pelos jogadores, tem-se o essencial que servirá como

ponto de partida para a especificação das diferentes atividades que serão integradas nas etapas de criação de tal jogo. Na análise, esta definição será então transcrita para os requisitos funcionais e não funcionais. Na etapa de projeto, o documento de requisitos será modelado com apoio da UML, resultando em diferentes diagramas, em especial o modelo arquitetural, para então seguir para o desenvolvimento e os testes. Por último, após o jogo ser comercializado, ele pode ser modificado ou melhorado. Isso ocorre na etapa de manutenção. Considerando um cenário hipotético em que testes indicam que o jogo digital desenvolvido não atingiu a qualidade esperada, aponte quais as possíveis falhas que podem ter ocorrido em três perspectivas: a) do ponto de vista dos artefatos criados na análise e no projeto; b) da seleção de um ou mais padrões para o desenvolvimento; c) da condução das atividades realizadas pelas equipes.

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

O impressionante crescimento do mercado de jogos digitais nos últimos anos aumentou a demanda por materiais didáticos que forneçam ao futuro técnico da área o suporte necessário para sua formação. Neste livro, em especial no capítulo 1 – “O que são jogos digitais?”, é possível ter uma visão geral dos jogos digitais e entender conceitos básicos para o desenvolvimento de

tais softwares. Recomenda-se, ainda, para aqueles que querem ir além, a leitura do capítulo 5 – “Elementos necessários para a criação de jogos digitais”, como complemento ao aprendizado no tema.

Para realizar a leitura, acesse a Biblioteca Virtual da Kroton e busque pelo título da obra.


ARRUDA, Eucidio Pimenta. **Fundamentos para o Desenvolvimento de Jogos Digitais: Série Tekne**. Porto Alegre: Bookman Editora, 2014.

Indicação 2

As identidades no contexto de jogos digitais são essenciais, sendo definidas de três maneiras: virtual, real e projetiva. Este artigo é discutira da maneira como jogadores se identificam com as personagens durante o jogo. Este entendimento leva ao desenvolver a observar de outro modo como suas narrativas e avatares podem ser criados, direcionando ao sucesso mais assertivo em seus jogos digitais.

Para realizar a leitura deste artigo, acesse a plataforma Biblioteca Virtual da Kroton e busque pelo título da obra na no campo de busca, na tela principal.

ZACCHI, V. J. **Identidade em Jogos Digitais: Entre a Identificação e a Mecânica do Jogo**. Cascavel: Línguas & Letras, [S.l.], v. 19, n. 44, p. <http://dx.doi.org/10.5935/1981-4755.20180028>, dez. 2018. ISSN 1981-4755. Disponível em: <http://e-revista.unioeste.br/index.php/linguaseletras/article/view/20411>. Acesso em: 5 nov. 2020.



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. O documento de desenho do jogo, do termo em inglês *game design document*, é um documento e artefato essencial para a condução das diferentes etapas para a implementação de um jogo digital. Sobre os principais itens documentados neste artefato, assinale a alternativa correta:
 - a. Esse documento recebe as especificações principais para a criação de um jogo, bem como a definição das tecnologias adotadas, padrões de desenvolvimento, arquiteturais e, principalmente, os elementos específicos integrados a um produto de software de jogo digital.
 - b. O documento de desenho de jogos pode ser substituído pelo diagrama de atividades da UML.
 - c. O item de visão geral do jogo especifica, em conjunto com os diagramas de casos de uso da UML, quais são os elementos que serão traduzidos para requisitos funcionais e não funcionais.
 - d. *Gameplay* e mecânicas de jogo é o único item que requer gerenciamento e equipe de desenvolvimento específica.
 - e. O cronograma das atividades da criação do jogo digital é definido e mantido no item que discorre sobre história e personagens do jogo digital.


2. A metodologia ágil vem ao encontro da necessidade de dinamizar o processo de desenvolvimento, das etapas de implementação de um software no contexto da engenharia de software. Considerando os quatro pilares que norteiam os métodos ágeis, assinale a alternativa correta:
- a. É considerado um dos pilares a colaboração com o cliente mais que responder a mudanças.
 - b. O pilar mais importante é o que indica que é fundamental ter um software em testes mais que documentação abrangente.
 - c. É considerado um dos pilares a negociação de contratos mais que a colaboração com o cliente.
 - d. É considerado um dos pilares da metodologia ágil seguir um plano mais do que responder a mudanças.
 - e. Dá-se maior atenção aos indivíduos e interações mais que a processos e ferramentas.



GABARITO

Questão 1 - Resposta C

Resolução: O documento de desenho do jogo recebe não apenas as especificações principais de um jogo, mas todas elas. Como é essencial para o desenvolvimento de produtos de software de qualidade, não pode ser substituído por um diagrama de atividades da UML, contudo, recomenda-se que tal diagrama seja parte integrante deste documento. Adicionalmente, este servirá para guiar o gerenciamento das equipes dos diferentes artefatos de software a serem desenvolvidos. Eles não se restringem apenas ao *gameplay* e mecânicas do jogo. Finalmente, o cronograma das



atividades é mantido no gerenciamento e tal como todas as demais partes do documento de desenho do jogo, leva em consideração a visão geral do jogo que, juntamente com modelos UML, designam quais são os elementos que serão traduzidos para requisitos funcionais e não funcionais.

Questão 2 - Resposta E

Resolução: Os quatro pilares que embasam a metodologia ágil são indivíduos e interações mais que processos e ferramentas, software em funcionamento mais que documentação abrangente, colaboração com o cliente mais que negociação de contratos e responder a mudanças mais que seguir um plano. Logo, não há um pilar que preconiza colaboração com o cliente mais que responder a mudanças, que indica que é fundamental ter um software em testes mais que documentação abrangente, que valoriza mais a negociação de contratos que a colaboração com o cliente e que indica seguir um plano mais do que responder a mudanças.

TEMA 4

Desenvolvimento e gestão de projetos com DevOps

Autoria: Anderson da Silva Marcolino

Leitura crítica: Aline Chagas Rodrigues Marques





DIRETO AO PONTO

Automatizar atividades de nosso cotidiano nos permite ter mais tempo para focarmos em outros afazeres. Consequentemente, se o processo de automação aplicado for executado com presteza, gera-se um resultado com melhor qualidade. Nesta perspectiva, a união do desenvolvimento com operações dá origem ao que se intitula DevOps (união de “Dev” de *developer* – desenvolvedor – e “Ops” de *operations* – operações).

O DevOps é uma mudança de paradigmas culturais que integra e automatiza práticas e ferramentas que tornam as empresas de desenvolvimento de software capazes de entregar produtos (por exemplo, aplicativos, serviços de software) mais rapidamente. Gera-se um alinhamento entre times de desenvolvimento com operação que, com a automação de diferentes etapas envolvidas no processo de implementação de software, entregam produtos de qualidade em menor tempo.

DevOps é uma cultura da engenharia de software, dentre os principais benefícios de sua adoção, destacam-se: (1) melhorias na qualidade dos produtos de software; (2) maior número de entregas; (3) melhor comunicação entre as áreas de desenvolvimento e operações; (4) melhor estabilidade nas entregas das modificações nos softwares; (5) valor do negócio aumentado.

Para atingir tais benefícios, DevOps é embasado na união da engenharia de software, garantia de qualidade e operações. Cada uma destas áreas cria, por sua vez, um conjunto de pilares que apoia as diferentes etapas de implementação de software. Esses pilares são:

- 1. Cultura:** desenvolve-se a integração entre os times de desenvolvimento e de operações por meio da alteração do

modo como eles se comunicam e se integram. Tal união permite a aceitação facilitada das mudanças e a adoção de novas práticas, emergentes ou não.

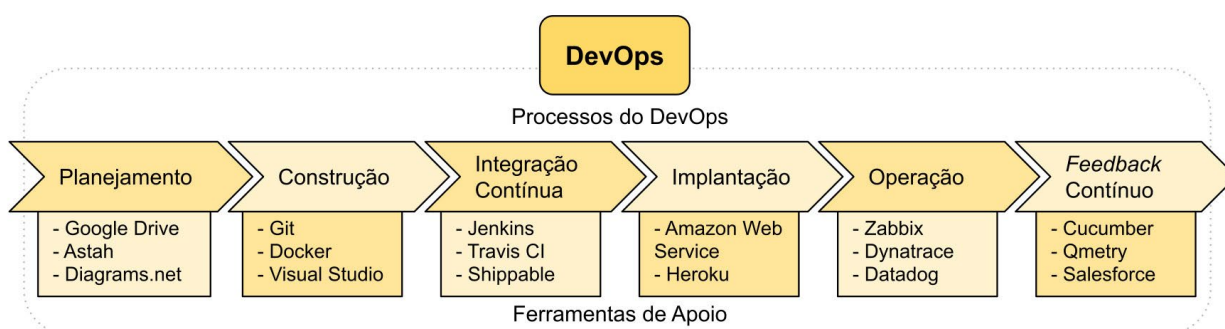
2. **Automação:** preza-se pela redução de trabalhos manuais, de modo que tudo que pode ser automatizado assim o seja. Esta automatização ocorre por meio de algoritmos e ferramentas de software que permitam desde testes automatizados à geração do produto de software final e sua implantação em ambiente de produção.
3. **Medição/mensuração:** as métricas são fundamentais para que o projeto seja analisado tanto no que tange à sua condução (cronograma) quanto em relação à qualidade e ao atendimento dos requisitos esperados pelos *stakeholders*. É por meio das métricas que a mensuração de como está o andamento do projeto pode ser feito, principalmente em relação às entregas, que, no contexto do DevOps, devem ser pequenas, mas sempre necessitam ocorrer.
4. **Fluxo:** não é apresentado em todas as literaturas, mas indica a necessidade de identificar o fluxo que gera valor ao cliente, fazendo-o seguir e, assim, garantir tal geração de valor, ou seja, é analisar como o problema do cliente é solucionado pelo produto que está sendo desenvolvido e focar nas entregas dos serviços, microsserviços ou artefatos de software que atendam a tal demanda. Este pilar também está relacionado ao termo *lean*, que significa enxuto. Este termo remete ao conceito com que empresas emergentes, também conhecidas como *startups*, definem seus processos e atividades. Devido ao número reduzido de recursos humanos e financeiros, estas empresas optam por enxugar, ou seja, reduzir os processos para que possam atendê-los com sucesso.
5. **Compartilhamento:** na implementação de software, todos possuem responsabilidades e elas devem ser assumidas, fornecendo meios de dinamizar os projetos a

serem desenvolvidos e as atividades de desenvolvimento. Contudo, mais que assumidas, as experiências devem ser compartilhadas. Este compartilhamento, por sua vez, ocorre, muitas vezes, por meio do feedback, ou seja, a retroalimentação a um resultado ou uma situação compartilhada entre os participantes dos times em projetos.

Destarte, considerando tais pilares, as etapas de implementação de software, comumente definidas pela engenharia de software (a saber: análise, projeto, desenvolvimento, testes e manutenção) são modificadas, integrando os pilares e as equipes de desenvolvimento e operação. Apesar do nome DevOps estar relacionado apenas a desenvolvimento e operações, as mudanças devem ocorrer em todas as etapas, incluindo análise e projeto. Com este alinhamento, o sucesso de sua aplicação será mais bem trabalhado, conseguindo usufruir dos benefícios dessa mudança cultural que tem sido fortemente adotada pela indústria de desenvolvimento de software.

O infográfico da Figura 1 apresenta as ferramentas de software que podem ser adotadas nos processos do DevOps, garantindo a execução e presença dos pilares e, como consequência, a obtenção dos benefícios possíveis por meio do estabelecimento de tal cultura.

Figura 1 – Infográfico de ferramentas e processos do DevOps



Fonte: elaborada pelo autor.

As ferramentas indicadas para cada um dos processos do DevOps auxiliam na aplicação de seus pilares no contexto de implementação de software. Destaca-se que as ferramentas indicadas são apenas algumas da infinidade de ferramentas existentes. Caberá à equipe, com base na sua experiência, selecionar as que melhor lhe cabem para serem adotadas, bem como analisar se os recursos financeiros permitem sua adoção.

Referências bibliográficas

SATO, D. **DevOps na prática**: entrega de software confiável e automatizada. São Paulo: Editora Casa do Código, 2014.

SOMMERVILLE, I. **Engenharia de software**. Tradução Luiz Cláudio Queiroz. Revisão técnica Fábio Levy Siqueira. 10. ed. São Paulo: Pearson Education do Brasil, 2018.

PARA SABER MAIS

Você sabe o que é um processo de software? As cinco etapas ou os cinco processos da engenharia de software são parte integrante deles, e são cruciais para o sucesso na implementação de software.

Um processo de software pode ser também chamado de abordagem sistemática para desenvolvimento, é a sequência das atividades que resultarão na produção de um produto de software com qualidade. Este processo é composto por outros cinco, já conhecidos: análise, projeto, desenvolvimento, testes e manutenção. Em algumas literaturas, são conhecidos como etapas ou atividades fundamentais, sendo resumidas em quatro: especificação (análise e projeto), desenvolvimento, validação e evolução.

Nesta perspectiva, ao longo da evolução de hardware e software, diferentes modelos de processos foram criados. E muitos ainda o serão. Os principais modelos de processo de software são:

- **Cascata:** neste modelo, as etapas são executadas seguindo uma ordem, levando de uma etapa à outra (modelo sequencial e linear), por isso o nome cascata. É também chamado de ciclo de vida clássica. As etapas são separadas e dependentes, deste modo, a etapa seguinte não pode ser iniciada até que a anterior esteja finalizada. Seu uso é indicado quando os requisitos estão bem definidos e compreendidos e não serão modificados radicalmente no decorrer da execução das etapas.
- **Incremental:** também chamado evolucionário, neste modelo são desenvolvidas versões até que o produto de software esteja concluído. Logo, as etapas de implementação ocorrem em várias etapas, resultando em entregas. Somente quando todos os requisitos tiverem sido concluídos, devidamente implementados no produto de software, este estará pronto para a implantação em ambiente de produção. O modelo pode integrar etapas iterativas. Um processo iterativo pode ocorrer por meio de tentativas sucessivas de refinamento, já o processo incremental é aquele em que o software é construído e entregue em partes menores.
- **Orientado a reuso:** também chamado de baseado em componentes, utiliza elementos denominados componentes já desenvolvidos, reduzindo o tempo de implementação e finalização do produto. O processo de desenvolvimento se concentra principalmente na integração desses componentes, no produto final. Uma das grandes limitações é que, como as funcionalidades foram desenvolvidas previamente, há a necessidade de adaptar os requisitos, visto que a modificação dos componentes levaria a esforços maiores do que se os mesmos fossem desenvolvidos do zero.

Finalmente, também é possível integrar diferentes modelos. Qual deles você gostou mais?

Referências bibliográficas

SOMMERVILLE, I. **Engenharia de software**. Tradução Luiz Cláudio Queiroz. Revisão técnica Fábio Levy Siqueira. 10. ed. São Paulo: Pearson Education do Brasil, 2018.

TEORIA EM PRÁTICA

Refleta sobre a seguinte situação: um gerente de equipes de desenvolvimento tem tido dificuldades com duas equipes em dois projetos de software distintos. A maior limitação tem sido reunir as equipes nas reuniões que ocorrem a cada 15 dias. Considerando os conteúdos estudados neste tema, você indicaria a adoção de DevOps como meio de solucionar tal problema ou apenas partes integrantes deste modelo cultural para implementação de software? Discorra justificando a sua resposta com base nas áreas, nos pilares e processos envolvidos no DevOps.

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

Esta dissertação de mestrado apresenta um panorama sobre o uso de práticas DevOps, identificado por meio de um mapeamento sistemático da literatura. Nesta perspectiva, permite

uma visão aprofundada sobre o tema, especialmente no capítulo 2, que apresenta a revisão da literatura.

Para realizar a leitura, acesse a Biblioteca Virtual da Kroton e busque pelo título da obra.


BRAGA, F. A. M. **Um panorama sobre o uso de práticas DevOps nas indústrias de software**. 2015. Dissertação de Mestrado. Universidade Federal de Pernambuco. Disponível em: https://repositorio.ufpe.br/bitstream/123456789/15989/1/Filipe_Versao_Final_Pos_Defesa_Deposito.pdf. Acesso em: 4 nov. 2020.

Indicação 2

O DevOps é frequentemente referenciado como um movimento emergente que permite a entrega de software por meio da colaboração entre as equipes de desenvolvimento e da equipe de operações. Nesta tese de doutorado escrita em português de Portugal, é relatado um estudo de caso da implementação do DevOps, o que permite a visualização prática deste movimento, com destaque para suas vantagens e desvantagens nos itens 3.6 e 3.8, respectivamente, do capítulo 3 – “Análise e discussão dos resultados”.

Para realizar a leitura, acesse a Biblioteca Virtual da Kroton e busque pelo título da obra.

SOUSA, L. F. R. **DevOps: estudo de caso**. 2019. Tese de Doutorado. Disponível em: http://comum.rcaap.pt/bitstream/10400.26/31932/1/Leandro_Sousa.pdf. Acesso em: 4 nov. 2020.



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. O DevOps é uma mudança cultural para integração de áreas de desenvolvimento e operações. Nesta perspectiva, assinale a alternativa correta que apresenta os pilares desta prática:
 - a. Compartilhamento, medição, cultura, automação e fluxo.
 - b. Compartilhamento, medição, cultura, autoconhecimento e fluxo.
 - c. Compartilhamento, medição, cultura, automação e robustez.
 - d. Feedback, colaboração, ferramentas, processos enxutos e métricas.
 - e. Compartilhamento, medição, cultura, pipelines e feedback.

2. Os modelos de processo de software definem a sequência e como as diferentes etapas de implementação podem ser organizadas. Estas etapas, por sua vez, podem ser utilizadas no contexto de metodologias ágeis, DevOps e outras. Considerando os modelos de processo de software, assinale a alternativa correta:

- a. No modelo em cascata, as etapas são executadas fora de ordem.
- b. No modelo incremental, também chamado revolucionário, várias versões são desenvolvidas até que o produto de software esteja concluído.
- c. No modelo baseado em componentes, a prática de reuso é realizada por meio da utilização de componentes já desenvolvidos, reduzindo o tempo de implementação e finalização do produto.
- d. No modelo em cascata, o processo de desenvolvimento se concentra principalmente na integração de componentes, gerando um produto final.
- e. No modelo incremental, somente quando todos os requisitos não funcionais tiverem sido implementados no produto de software, o produto final será implantado em ambiente de produção.



GABARITO

Questão 1 - Resposta A

Resolução: O DevOps se baseia em cinco pilares, sendo estes: cultura, que preza pela colaboração dos times; automação, que se baseia em ferramentas e pipelines; medição, que integra métricas e monitoramento das entregas, qualidade e cronograma; fluxo, que refere-se à identificação da demanda de valor dos clientes; e compartilhamento, que refere-se a difusão das experiências, dificuldades, abordagens e responsabilidades entre os integrantes do projeto.

Questão 2 - Resposta C

Resolução: No modelo em cascata, as etapas são executadas ordenadamente, não concentrando o processo de desenvolvimento em componentes. Já o modelo incremental é também conhecido como evolucionário, gerando várias versões do produto, até que este esteja concluído. É neste modelo que, somente quando todos os requisitos (funcionais e não funcionais) tiverem sido implementados, que o software poderá ser implantado em ambiente de produção. Finalmente, o modelo baseado em componentes objetiva o reúso de componentes já desenvolvidos, o que reduz o tempo de finalização.



BONS ESTUDOS!