Estratégia Concursos

Aula 00

Câmara de Caieiras-SP (Analista de Tecnologia da Informação) Engenharia de Software - 2024 (Pós-Edital)

Autor:
Diego Carvalho

19 de Março de 2024

Índice

APRESENTAÇÃO

O assunto da aula de hoje é: Metodologias Ágeis! Vamos ver agora um novo paradigma de desenvolvimento de software bem interessante, muda bastante coisa em relação às metodologias tradicionais – é bem mais moderno. Esse é um assunto que sempre corre o risco de cair ao menos uma questãozinha na prova porque é o paradigma de desenvolvimento mais utilizado atualmente. Então, venham na fé que vocês vão gostar :)

PROFESSOR DIEGO CARVALHO - WWW.INSTAGRAM.COM/PROFESSORDIEGOCARVALHO

Galera, todos os tópicos da aula possuem Faixas de Incidência, que indicam se o assunto cai muito ou pouco em prova. *Diego, se cai pouco para que colocar em aula?* Cair pouco não significa que não cairá justamente na sua prova! A ideia aqui é: se você está com pouco tempo e precisa ver somente aquilo que cai mais, você pode filtrar pelas incidências média, alta e altíssima; se você tem tempo sobrando e quer ver tudo, vejam também as incidências baixas e baixíssimas. *Fechado?*

INCIDÊNCIA EM PROVA: BAIXÍSSIMA
INCIDÊNCIA EM PROVA: BAIXA
INCIDÊNCIA EM PROVA: MÉDIA
INCIDÊNCIA EM PROVA: ALTA
INCIDÊNCIA EM PROVA: ALTÍSSIMA

Além disso, essas faixas não são por banca – é baseado tanto na quantidade de vezes que caiu em prova independentemente da banca e também em minhas avaliações sobre cada assunto...

#ATENÇÃO

Avisos Importantes

O curso abrange todos os níveis de conhecimento...

Esse curso foi desenvolvido para ser acessível a **alunos com diversos níveis de conhecimento diferentes.** Temos alunos mais avançados que têm conhecimento prévio ou têm facilidade com o assunto. Por outro lado, temos alunos iniciantes, que nunca tiveram contato com a matéria ou até mesmo que têm trauma dessa disciplina. A ideia aqui é tentar atingir ambos os públicos - iniciantes e avançados - da melhor maneira possível..

Por que estou enfatizando isso?

O **material completo** é composto de muitas histórias, exemplos, metáforas, piadas, memes, questões, desafios, esquemas, diagramas, imagens, entre outros. Já o **material simplificado** possui exatamente o mesmo núcleo do material completo, mas ele é menor e bem mais objetivo. *Professor, eu devo estudar por qual material?* Se você quiser se aprofundar nos assuntos ou tem dificuldade com a matéria, necessitando de um material mais passo-a-passo, utilize o material completo. Se você não quer se aprofundar nos assuntos ou tem facilidade com a matéria, necessitando de um material mais direto ao ponto, utilize o material simplificado.

Por fim...

O curso contém diversas questões espalhadas em meio à teoria. Essas questões possuem um comentário mais simplificado porque **têm o único objetivo de apresentar ao aluno como bancas de concurso cobram o assunto previamente administrado**. A imensa maioria das questões para que o aluno avalie seus conhecimentos sobre a matéria estão dispostas ao final da aula na lista de exercícios e **possuem comentários bem mais completos, abrangentes e direcionados.**

METODOLOGIAS ÁGEIS

Conceitos Básicos

INCIDÊNCIA EM PROVA: ALTA

Em meados de 2001, 17 especialistas proeminentes da área de desenvolvimento de software se reuniram em um *resort* em Utah (foto acima) para conversar, esquiar, **discutir e encontrar um terreno comum para suas ideias sobre métodos de desenvolvimento de software**. Essa galera pegou uma mesa, se sentaram, tomaram umas cervejas e começaram a desabafar sobre seus projetos de desenvolvimento de software que estavam falhando por diversos motivos.

Os 17 Engenheiros de Software

Foi quando um deles levantou a mão e **disse que usou o Modelo em Cascata e o projeto estourou o orçamento**; o outro disse que isso também já aconteceu com ele, mas recentemente um projeto falhou porque estourou o prazo; aí outro se compadeceu e disse que os projetos dele viviam falhando porque ele não conseguia construir todo o escopo que foi pedido pelos usuários do sistema. E assim foi...

Eles foram compartilhando suas experiências ruins com o uso das metodologias tradicionais, mas depois cada um desses caras foi dizendo: "*para remediar isso, agora eu uso iterações*"; aí o outro disse que não usa mais tanta documentação como antigamente; aí o outro levantou a mão e falou que não faz mais tanto planejamento; e assim por diante. Então, no decorrer da reunião, foi sendo criado um consenso entre os participantes.

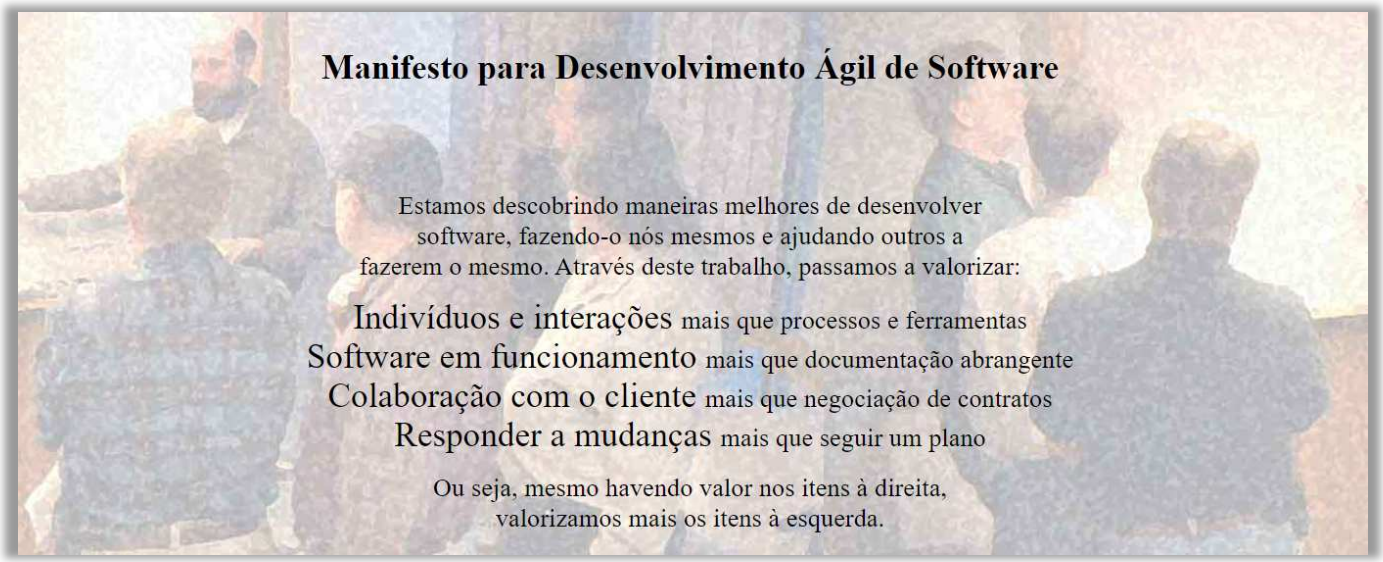
Foi aí que alguns acharam que era o momento de formalizar e elevar aquela reunião em um patamar maior. **Eles decidiram escrever um documento que serviria de grito de guerra contra os processos tradicionais de desenvolvimento de software que vigoravam naquela época**. Para isso, eles pensaram: "*Nós precisamos de um nome que expresse bem o significado dessa reunião e das nossas ideias comuns*".

Na discussão, eles decidiram que a palavra "*leve*" não expressava tão bem o que eles queriam dizer e decidiram trocar pela palavra "*ágil*", que captava melhor a abordagem que eles estavam propondo. Em um segundo momento, eles começaram a escrever um **documento bem pequeno, bem objetivo, bem claro e que conteria as crenças, valores e princípios** daqueles dezessete engenheiros de software.

Foi então que surgiu o documento chamado **Manifesto Ágil Para Desenvolvimento De Software**, que definia bem o que era ágil, o que não era ágil e o que essas pessoas defendiam. Além disso, eles passaram a se autodenominar como Aliança Ágil, que era um grupo de pensadores independentes sobre desenvolvimento de software e muitas vezes concorrentes entre si, mas que concordaram em um documento chamado Manifesto Ágil.

Isso se tornou uma organização sem fins lucrativos que procura promover conhecimento e discussões sobre os vários métodos ágeis que existem hoje em dia. A partir daí, galera... esses caras – que eram os líderes do movimento ágil – começaram a escrever artigos, fazer palestras e disseminar esse novo paradigma. Como vocês sabem, **esse negócio explodiu e hoje a imensa maioria dos projetos de software são feitos utilizando metodologias ágeis**.

Bem, para aqueles que não conhecem, nós trazemos a seguir a imagem original do próprio Manifesto Ágil com seus fundamentos. Vejamos...



Notem que a coluna da esquerda representa os anseios das metodologias ágeis, enquanto a coluna da direita representa o que as metodologias tradicionais costumavam executar. **Agora prestem**

atenção: o manifesto ágil afirma que, mesmo havendo valor nos itens à direita, valorizam-se mais os itens à esquerda. Uma pegadinha comum de prova é dizer que os métodos ágeis não possuem documentação. *Isso está correto?*

Não, isso evidentemente está errado! O manifesto ágil preconiza que se valorize mais software em funcionamento do que documentação abrangente, logo isso não significa que não tenha documentação. **E isso serve para os outros três fundamentos, isto é, os itens da direita tem o seu valor, por outro lado se valoriza mais os itens da esquerda**. *Tudo certo até aqui?* Agora vamos detalhar um pouco mais...

POR QUE VALORIZAR MAIS INDIVÍDUOS E SUAS INTERAÇÕES DO QUE PROCESSOS E FERRAMENTAS?

Porque, em última instância, quem gera produtos e serviços são os indivíduos, que possuem características únicas como talento e habilidade. Pessoal, programar é uma atividade humana e, como tal, depende de questões humanas para que obtenha sucesso. Jim Highsmith, um dos signatários do manifesto ágil, afirma que as habilidades, as personalidades e as peculiaridades de cada indivíduo são críticas para o sucesso dos projetos.

Ele diz também que pessoas muitas vezes são desorganizadas e difíceis de entender, por outro lado elas também são inovadoras, criativas, apaixonadas, entre outros. *E quanto às ferramentas e aos processos, professor?* **Galera, ambos são importantíssimos para guiar e apoiar o desenvolvimento, mas é a capacidade e o conhecimento dos indivíduos que ajudam a tomar decisões críticas no projeto.**

Dessa forma, basta eu ensinar um conjunto de processos para a minha equipe, assim como um conjunto de ferramentas para garantir que a equipe criará bons softwares? Claro que não! **Uma equipe possui características intrínsecas à personalidade, habilidades e capacidades de cada um dos seus integrantes e isso deve ser considerado e valorizado na construção de um software**. *Entendido, pessoal?* Seguindo...

POR QUE VALORIZAR MAIS SOFTWARE EM FUNCIONAMENTO DO QUE DOCUMENTAÇÃO ABRANGENTE?

Porque o que gera valor para o cliente é o resultado que você entrega e, não, a documentação em si. Respondam-me uma pergunta: *quando você compra um carro, você olha o motor, o design, o painel, o interior ou você sai correndo loucamente para ler o manual do carro e outras documentações?* Imagino que vocês tenham respondido a primeira opção! Dito isso, concluímos que o software em funcionamento é o único indicador do que, de fato, a equipe construiu.

Claro, não se exclui a necessidade de documentação, que é bastante útil para o desenvolvimento, mas é recomendável produzir somente a documentação necessária e suficiente para a realização do trabalho em si. **Nada de burocratizar demais e construir trezentas páginas de documentação com quatrocentos diagramas diferentes para representar o software**. *Tudo certo?* Eu vou repetir, porque esse assunto cai bastante em prova!

No ágil, documentação é descartável? Não, ela é útil para ajudar a comunicação e a colaboração dos integrantes da equipe, além de melhorar a transferência de conhecimento, preservar informações históricas, satisfazer necessidades contratuais ou legais, entre outros. **A documentação é importante, sim; mas valoriza-se mais o software em funcionamento, que é o que de fato agrega valor ao cliente**. *Belê?*

POR QUE VALORIZAR MAIS COLABORAÇÃO COM O CLIENTE DO QUE NEGOCIAÇÃO DE CONTRATOS?

Porque é importante o envolvimento contínuo do cliente! Aliás, desenvolvedores e clientes devem estar sempre lado a lado, visto que ambos possuem interesses em comum. *Qual?* Um software que agregue valor! No Modelo em Cascata, vocês devem se lembrar que o cliente até colaborava com a equipe no início do projeto (em geral, na fase de levantamento de requisitos), mas – depois disso – o cliente saía de cena e só aparecia novamente para ver o software já pronto.

E pior: muitas vezes, o cliente saía insatisfeito, porque o resultado não era o que ele esperava. Dessa forma, o manifesto ágil afirma que você tem que valorizar mais a sua relação com o cliente do que ficar discutindo itens de contrato: "*Isso não estava previsto no contrato*"; "*Isso não estava combinado previamente*"; "*Vou cobrar a mais porque você mudou tal coisa*"; entre outros. *Professor, então contratos não são importantes?* Claro que são!

Contratos regulam essa relação entre cliente e fornecedor, mas não se deve ser excessivamente rigoroso, porque isso pode acabar com a relação com seu cliente. Por falar em contrato, existem várias maneiras de fazer contratos de desenvolvimento ágil. Uma maneira comum é fixar o tempo e deixar o escopo variar. É o famoso: "***Tempo Fixo e Escopo Variável***"! Você fala para o seu cliente: "*É o seguinte: eu faço tudo que você pedir desde que seja possível fazer no prazo tal*".

POR QUE VALORIZAR MAIS A RESPOSTA A MUDANÇAS DO QUE SEGUIR UM PLANEJAMENTO ESPECÍFICO?

Porque, em geral, é necessário obter respostas rápidas a mudanças e seguir um desenvolvimento contínuo do software. Todo projeto deve balancear o planejamento com a mudança, dependendo do nível de incerteza do projeto. Manter-se preso a um planejamento ultrapassado pode ser nocivo ao andamento do projeto. Galera, nós estamos no século 21! Uma empresa líder de mercado pode acabar de uma hora para outra – nós vemos isso o tempo todo.

Cadê o Orkut? Cadê o MSN? Cadê a Nokia? Cadê a Kodak? Cadê a BlockBuster? Todas essas empresas foram gigantes pouco tempo atrás e simplesmente morreram! Logo, a única certeza que você tem em um projeto é a instabilidade! **Logo, a equipe deve estar preparada para mudanças no escopo, tempo, custo, tecnologia, arquitetura, no paradigma de programação, regulamentações, leis, regras de conformidade, entre outros**.

Não tem como fazer um planejando e achar que ele vai ficar fixo ali ao longo do tempo – isso é pensamento do século passado (se muito!). Acreditem: mudanças vão ocorrer! Planejar é bom

demais. **É tão bom que é recomendável refazer o planejamento a todo momento, de forma contínua e, não, fazer um planejamento estático e simplesmente segui-lo com todo rigor ignorando mudanças externas que venham a ocorrer**. *Fechou?*

Agilidade x Velocidade

INCIDÊNCIA EM PROVA: BAIXA

Pessoal, agora vamos falar rapidamente sobre uma diferença importante! *Vocês sabem qual a diferença entre agilidade e velocidade?* Antes de explicá-la no contexto de desenvolvimento de software, eu vou explicar como uma metáfora em outros dois contextos para facilitar o entendimento. Vamos pensar no atleta Usain Bolt! *O Usain Bolt é um cara veloz ou um cara ágil?* Bem, em comparação com seres humanos normais, ele é mais ágil e mais veloz que todo mundo! No entanto, vamos pensar só no grupo dos grandes atletas que disputam mundiais e olimpíadas de atletismo. Nesse contexto, ele é absurdamente veloz, mas menos ágil que a maioria dos seus concorrentes. *Como é, professor?*

Vejam as duas imagens a seguir: observem que – à esquerda – temos cerca de vinte metros de corrida e o Bolt é o atleta de azul no meio. Notem também que ele está mais ou menos em quarto lugar na corrida. *Por que?* **Porque agilidade é a capacidade de reagir ou responder adequadamente a mudanças e o Bolt sempre teve problemas de largada, uma vez que ele é mais alto e pesado que os outros**.

Logo, ele acaba reagindo de forma mais lenta que seus adversários quando o tiro de início da corrida é disparado. Vejam: todos estão parados e, ao disparar o tiro, nós temos uma mudança. Essa mudança faz com que o os atletas reajam e saiam da inércia. O Bolt demora mais que seus concorrentes a sair da inércia, uma vez que ele não responde tão bem quanto os outros a mudança. Nesse sentido, ele é ágil, mas não destoa dos outros pela sua agilidade.

Se nós tivéssemos corridas de 50 metros em vez de 100 metros, talvez ele não fosse tricampeão olímpico. Por outro lado, vejam o final da corrida quando ele não tem mais que reagir a mudanças, ele só tem que correr até o fim dos 100 metros. Ele termina muito distante do segundo lugar! *Por*

que? **Porque ele é um cara extremamente veloz, logo ele destoa de todos os outros com muita facilidade**. *Entenderam que agilidade não é velocidade?* É a capacidade de reagir a mudanças!

A velocidade trata de quão rápido é possível entregar um software para o cliente. E, para isso, nós temos outras metodologias de desenvolvimento (Ex: *Rapid Application Development* (RAD) é capaz de desenvolver softwares em poucos meses). Utilizando outra metáfora, isso ocorre também quando você tem uma disputa entre um carro muito potente e pesado, e um carro menos potente e mais leve.

É provável que o carro mais leve, mesmo sendo menos potente, tenha uma arrancada melhor que o carro mais potente, logo ele é mais ágil. *Ele é mais rápido?* Não, o carro mais potente é mais rápido, mas ele é mais potente! Claro, pessoal, que esses são exemplos genéricos – apenas para entender a ideia. *Diego, e como esse conceito de agilidade pode ser utilizado no contexto de um desenvolvimento de software?*

No contexto de projetos de software, podemos imaginar: eu estou gerenciando meu projeto de um novo sistema e, de repente, descubro que vou ter que mudar a arquitetura do software – **não tem problema**; se eu descubro que, por conta de cortes de gastos, eu terei que reduzir o tamanho a minha equipe – **não tem problema**; se eu tiver que trocar a tecnologia utilizada porque ela se tornou defasada – **mais uma vez, não tem problema**.

Pressman afirma que a agilidade pode ser aplicada a qualquer processo de software. No entanto, para obtê-la, é essencial que o processo de software seja projetado para que a equipe possa adaptar e racionalizar suas tarefas; para que a equipe possa conduzir o planejamento compreendendo a fluidez de uma abordagem do desenvolvimento ágil; e para que a equipe possa eliminar tudo, exceto os artefatos essenciais do processo.

Além disso, deve enfatizar a estratégia de entrega incremental, entregando para o cliente o software operacional o mais rapidamente possível para o tipo de produto e ambiente operacional. **Essa são as diretivas para que um processo de software qualquer possa ser, também, ágil.** Métodos ágeis são ágeis porque partem do princípio de que tem que responder adequadamente a mudanças que venham a ocorrer durante o ciclo de vida do projeto.

Eles são mais dinâmicos, adaptativos, interativos e colaborativos – eles se adaptam às necessidades de um projeto e às suas mudanças no decorrer do desenvolvimento; os métodos tradicionais são mais preditivos/prescritivos, processuais, formais, documentais e contratuais – eles valorizam mais o planejamento de todos os aspectos do processo de desenvolvimento de software como um todo.

Princípios Ágeis

INCIDÊNCIA EM PROVA: ALTA

A seguir, nós vamos conhecer quais são os princípios do Manifesto Ágil. Eles vêm expressamente no manifesto e vocês podem encontrá-lo no site oficial:

WWW.AGILEMANIFESTO.ORG

NÓS SEGUIMOS ESSES PRINCÍPIOS...
Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada e software com valor agregado.
Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.
Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo.
Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.
Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho.

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

Professor, metodologias ágeis são recomendadas para projetos de qualquer tamanho e complexidade? **Segundo Sommerville**: *"Todos os métodos têm limites, e os métodos ágeis são somente adequados para alguns tipos de desenvolvimento de sistema. Na minha opinião, eles são mais adequados para o desenvolvimento de sistemas de pequenas e médias empresas e produtos para computadores pessoais".*

Diego, você concorda com essa afirmação? Não, eu discordo! **Acredito que ela já foi válida tempos atrás, mas hoje não é mais! Projetos Ágeis já são suficientemente maduros para serem aplicados a projetos complexos e de grande porte**. Pessoal, essa é só a minha opinião! Não é possível saber ainda a posição das bancas caso isso seja questionado em provas de concurso. *Legal?* Vamos ver agora exemplos de metodologias ágeis de desenvolvimento:

PRINCIPAIS METODOLOGIAS ÁGEIS		
SCRUM	CRYSTAL	XP
TDD	ATDD	BDD
FDD	DDD	MDD
DSDM	ASD	KANBAN
BADM	AUP	AGILE MODELING
OSSD	SCRUMBAN	

Agora vamos ver algumas diferenças básicas entre metodologias de desenvolvimento software tradicionais e metodologias ágeis:

CRITÉRIO	MODELOS TRADICIONAIS	MODELOS ÁGEIS
PLANEJAMENTO	Comumente realizado em detalhe para todo o projeto em sua fase inicial.	Planejamento de alto nível no início do projeto e os detalhes são realizados durante o projeto. Não é necessário possuir um planejamento detalhado de todo o projeto. A restrição se dá apenas em possuir os detalhes do trabalho para a próxima iteração.

RISCOS	Pode exigir um grande esforço e equipe para atuar com os riscos de todo o projeto.	Prioriza os riscos gerais do projeto, mas foca principalmente nos riscos das próximas iterações, atuando assim em um escopo bem reduzido. A própria equipe atua com os riscos e pode obter apoio externo.
EQUIPE	Possui profissionais com papéis bem definidos, quantificada e mobilizada conforme o planejamento do projeto. A equipe executa o projeto guiado pelo Gerente de Projetos conforme o plano estabelecido.	Equipe multidisciplinar, multifuncional e auto-organizada. Ela decide como fazer e atua de forma colaborativa.
TEMPO DE ENTREGA	É realizado conforme o plano estabelecido e pode durar semanas, meses ou até mesmo anos.	Fixo e é conforme a definição de duração das iterações que comumente varia entre 1 e 4 semanas.
ACEITAÇÃO DE MUDANÇAS	Gerenciamento formal de mudanças, pois exige alteração do planejamento já realizado e geralmente precisa passar por aprovações formais de um ou mais níveis hierárquicos.	Mudanças são bem-vindas. Evita-se mudar o escopo da iteração em andamento, mas o escopo das futuras iterações podem ser replanejado conforme a necessidade do cliente.
PREVISIBILIDADE	Depende do intervalo de monitoramento e controle do projeto. Quanto mais curto, maior a chance de prever as ocorrências futuras. Quanto maior o intervalo, menor a chance de prever as ocorrências futuras.	Tende a ter uma grande previsibilidade futura devido à constante análise e feedback através das oportunidades de inspeção e adaptação providas pelo método.
RESULTADOS AO LONGO DO TEMPO	Tende a demorar a dar resultados a curto prazo, pois as entregas são geralmente realizadas ao final do projeto. Melhores resultados são apresentados em projetos de maior duração.	Gera resultados a curto, médio e longo prazo, pois atua com entregas antecipadas e de valor agregado e contínuo ao cliente.
APRESENTAÇÃO DE INFORMAÇÕES DO PROJETO	Geralmente de uma apresentação formal previamente agendada com os stakeholders em intervalos de tempo. As informações podem ser detalhadas ou não conforme a necessidade do público envolvido.	Geralmente informal e utiliza radiadores de informação no ambiente de trabalho durante todo o projeto, de modo que as informações do projeto fiquem visíveis e transparentes a toda equipe e envolvidos.
PRAZO DE ENTREGA	Conforme estabelecido no planejamento do projeto. No caso de mudanças aprovadas, varia conforme os impactos das solicitações e podem ser traumáticas aos envolvidos quanto às suas expectativas.	Conforme o tamanho da iteração e o planejamento das releases para as entregas significativas.

DOCUMENTAÇÃO	Detalhada desde o início do projeto.	Abrangente no início e detalhada somente o necessário durante o projeto conforme os objetivos das iterações e releases.
ATUAÇÃO DO CLIENTE	Nas fases iniciais e nas principais validações do produto.	Durante todo o projeto, o cliente faz parte da equipe.
DISCUSSÕES E MELHORIAS	Geralmente em prazos longos através da realização de reuniões após uma grande etapa ou grande entrega do projeto.	Em prazos curtos, sempre ao final das iterações.
COMANDANTE	Gerente de Projetos.	Equipe do Projeto.
PAPÉIS	Claros e definidos.	Conforme a confiança na equipe e ambiente colaborativo.
PROCESSO	Guiado conforme o planejamento do projeto e nos processos estabelecidos no plano.	Empírico e guiado ao produto e às pessoas. Orientado à geração de valor e conforme priorização dos riscos.
RESULTADO	Melhor resultado em projetos com escopo muito bem definido e orientado a planejamento.	Melhor resultado em projetos cujo escopo é dinâmico e construído durante a execução do projeto.

Profissional Ágil

INCIDÊNCIA EM PROVA: BAIXÍSSIMA

No contexto do paradigma ágil e do manifesto ágil, o perfil de um profissional ágil é caracterizado por um conjunto de qualidades e habilidades que refletem os valores e princípios ágeis. Essas características são essenciais para navegar com sucesso em ambientes que adotam metodologias ágeis (Ex: Scrum, Kanban, XP, entre outras). Vejamos alguns dos aspectos mais importantes que definem um profissional ágil:

CARACTERÍSTICAS	MODELOS
FLEXIBILIDADE E ADAPTABILIDADE	Profissionais ágeis são capazes de se adaptar rapidamente a mudanças no escopo do projeto, requisitos ou prioridades. Eles entendem que mudanças são parte do processo de desenvolvimento e estão preparados para pivotar quando necessário.
COLABORAÇÃO E COMUNICAÇÃO	Eles valorizam a colaboração entre os membros da equipe e com os clientes ou stakeholders. A comunicação aberta é essencial, tanto para o sucesso do projeto quanto para a manutenção de um ambiente de trabalho positivo.
FOCO NO CLIENTE	A satisfação do cliente é uma prioridade. Profissionais ágeis trabalham de forma iterativa e incremental, buscando feedback contínuo para garantir que o produto final atenda ou exceda as expectativas do cliente.
APRENDIZADO CONTÍNUO	O aprendizado contínuo é fundamental em um ambiente ágil, dada a sua natureza evolutiva. Profissionais ágeis são proativos em seu desenvolvimento pessoal e profissional, buscando constantemente melhorar suas habilidades e conhecimentos.
PROATIVIDADE E AUTONOMIA	São profissionais que tomam a iniciativa, não esperando serem ditados para agir. Eles têm um senso de propriedade sobre seu trabalho e são capazes de trabalhar de forma independente, ao mesmo tempo em que entendem a importância da colaboração e do trabalho em equipe.
RESILIÊNCIA E PERSISTÊNCIA	Enfrentar desafios e superar obstáculos é parte do trabalho. Profissionais ágeis são resilientes frente às dificuldades e persistem até alcançar os objetivos desejados, vendo falhas como oportunidades de aprendizado.
HABILIDADE DE PRIORIZAÇÃO	Eles são adeptos de priorizar tarefas e recursos de forma eficaz, focando no que entrega o maior valor. Além disso, baseiam suas decisões em dados e feedbacks reais, em vez de suposições.
EMPATIA E SUPORTE AOS OUTROS	Profissionais ágeis valorizam a contribuição de cada membro da equipe e promovem um ambiente onde todos possam crescer. A empatia permite que entendam as perspectivas dos outros, facilitando a resolução de conflitos e a construção de relações saudáveis.

O perfil de um profissional ágil, portanto, vai além das habilidades técnicas, englobando uma série de competências comportamentais e uma mentalidade alinhada aos princípios do Manifesto Ágil. Estes incluem indivíduos e interações mais do que processos e ferramentas; software funcionando mais do que documentação abrangente; colaboração com o cliente mais do que negociação de contratos; e responder a mudanças mais do que seguir um plano.

Este conjunto de características habilita profissionais a navegar efetivamente em ambientes dinâmicos e orientados à inovação. Ainda nesse contexto, podemos falar sobre como identificar e entregar valor para o cliente digital. **Isso envolve entender profundamente as necessidades,**

preferências e comportamentos dos clientes no ambiente digital, e responder a eles de forma rápida e eficaz.

Identificar e entregar valor para clientes digitais é um processo contínuo de aprendizado e adaptação. Requer uma abordagem ágil que coloque o cliente no centro do desenvolvimento de produtos, garantindo que as soluções não apenas atendam às suas necessidades atuais, mas também antecipem e se adaptem às suas necessidades futuras. Enfatizo que o conceito de valor pode variar significativamente dependendo do cliente e do contexto.

Dessa forma, é crucial adotar uma abordagem centrada no cliente para garantir que o produto ou serviço entregue atenda ou exceda suas expectativas. Vejamos:

ESTRATÉGIAS	MODELOS
PESQUISA E FEEDBACK	Utilize pesquisas, entrevistas, análises de comportamento no site/app e feedback direto para entender o que os clientes valorizam, quais problemas eles precisam resolver, e como preferem interagir digitalmente.
PERSONA E JORNADAS DO CLIENTE	Crie personas detalhadas e mapeie as jornadas do cliente para identificar pontos de dor, necessidades não atendidas e oportunidades para agregar valor.
MVP (MINIMAL VIABLE PRODUCT)	Lance versões iniciais do produto para testar hipóteses sobre o valor para o cliente, permitindo ajustes rápidos com base no feedback real.
ENTREGA CONTÍNUA	Adote práticas de entrega contínua para fornecer melhorias e novas funcionalidades de forma regular, mantendo o produto alinhado às expectativas e necessidades em evolução do cliente.
ANÁLISE DE DADOS E MÉTRICAS DE SUCESSO	Monitore métricas relevantes (como engajamento do usuário, conversão, retenção) para avaliar o sucesso das iniciativas e entender melhor o que os clientes valorizam.
TESTES A/B E EXPERIMENTAÇÃO	Realize testes A/B e experimentos para iterar sobre diferentes abordagens e identificar o que melhor atende às necessidades do cliente.
DESIGN CENTRADO NO USUÁRIO	Garanta que o produto seja intuitivo, fácil de usar e esteticamente agradável, criando uma experiência positiva que aumente a satisfação e a fidelidade do cliente.
USABILIDADE E ACESSIBILIDADE	Certifique-se de que o produto seja acessível a todos os usuários, incluindo aqueles com deficiências, e otimize a usabilidade para diferentes dispositivos e plataformas.
COMUNICAÇÃO PERSONALIZADA	Use a comunicação para educar, informar e engajar os clientes de maneira personalizada, baseando-se em seus interesses e comportamentos.
CONSTRUÇÃO DE COMUNIDADE	Fomente uma comunidade em torno do seu produto ou marca, incentivando o feedback, a participação e o engajamento dos usuários.
NOVAS TENDÊNCIAS E TECNOLOGIAS	Mantenha-se atualizado com as últimas tendências digitais e tecnologias para explorar novas maneiras de criar valor para o cliente.

CULTURA DE INOVAÇÃO	Promova uma cultura que incentive a experimentação e a inovação, permitindo que a equipe explore novas ideias que possam entregar valor adicional aos clientes.

Ferramentas, Artefatos, Métricas e Indicadores

INCIDÊNCIA EM PROVA: BAIXÍSSIMA

Nesse tópico, vamos nos concentrar nas principais **ferramentas**, **artefatos**, **métricas** e **indicadores** ágeis. Vamos lá...

Ferramentas

Ferramentas ágeis são aplicativos de software projetados para apoiar equipes e organizações na implementação de práticas ágeis e na gestão de projetos ágeis. Elas facilitam a colaboração, o planejamento, a execução e o monitoramento de projetos, permitindo que as equipes sejam mais eficientes, transparentes e adaptáveis. Vejamos os principais tipos, características e exemplos de ferramentas ágeis:

TIPOS DE FERRAMENTAS	DESCRIÇÃO
GERENCIAMENTO DE PROJETO E COLABORAÇÃO	Suportam a planejamento de sprints, rastreamento de tarefas, quadros Kanban, e comunicação entre membros da equipe.
INTEGRAÇÃO E ENTREGA CONTÍNUA	Automatizam o processo de desenvolvimento de software, desde a integração do código até o teste e a entrega.
MONITORAMENTO E RELATÓRIOS	Oferecem visibilidade sobre o progresso do projeto, a saúde do código, e métricas de desempenho.
RASTREAMENTO DE BUGS E ISSUES	Facilitam a identificação, atribuição e resolução de bugs e outros problemas.
REPOSITÓRIOS E REVISÃO DE CÓDIGO	Permitem que as equipes gerenciem versões do código e revisem o trabalho uns dos outros para manter a qualidade.

EXEMPLOS	DESCRIÇÃO
JIRA	Oferece suporte para Scrum e Kanban, além de permitir o rastreamento de issues e bugs, planejamento de sprints, e relatórios ágeis.
TRELLO	Baseado em quadros Kanban, o Trello é intuitivo e fácil de usar para o rastreamento de tarefas e a colaboração em projetos de menor escala.
ÁSANA	Um aplicativo de gerenciamento de trabalho que facilita o planejamento de projetos, o rastreamento de tarefas e a colaboração entre equipes.
CONFLUENCE	Uma ferramenta de colaboração que permite que as equipes criem, compartilhem e colaborem em conteúdo e documentação.
GITHUB	Um repositório de código que facilita a colaboração em projetos de software, oferecendo ferramentas de revisão de código, gerenciamento de projetos e integração contínua.
GITLAB	Similar ao GitHub, oferece um repositório de código e ferramentas CI/CD integradas, além de funcionalidades para gerenciamento de issues e revisão de código.
SLACK	Uma plataforma de comunicação que promove a colaboração em tempo real, integrando-se facilmente com muitas outras ferramentas ágeis.

BENEFÍCIOS	DESCRIÇÃO

MELHORIA DA COLABORAÇÃO	Facilitam a comunicação e a colaboração entre membros da equipe e stakeholders, independentemente de sua localização.
VISIBILIDADE E TRANSPARÊNCIA	Proporcionam uma visão clara do progresso do projeto, ajudando na tomada de decisões baseadas em dados.
EFICIÊNCIA OPERACIONAL	Automatizam tarefas repetitivas e promovem a eficiência na gestão de projetos e no desenvolvimento de software.
ADAPTABILIDADE E FLEXIBILIDADE	Permitem que as equipes respondam rapidamente a mudanças no projeto ou no ambiente de mercado.
MELHORIA CONTÍNUA	Facilitam o rastreamento de métricas e feedbacks, apoiando a melhoria contínua dos processos e produtos.

Artefatos

Os artefatos ágeis são elementos tangíveis produzidos durante o ciclo de desenvolvimento ágil. Eles servem para ajudar as equipes a planejar, organizar, executar e revisar seu trabalho, mantendo a transparência e facilitando a comunicação entre todos os envolvidos no projeto. Eles são fundamentais para a implementação bem-sucedida de práticas ágeis e para a entrega de valor contínuo aos clientes. Vejamos alguns exemplos de artefatos:

ARTEFATOS	DESCRIÇÃO
BACKLOG DO PRODUTO	Uma lista ordenada de tudo que é necessário no produto final. É dinâmico e constantemente revisado e priorizado pelo Product Owner. Serve como uma fonte única de requisitos para qualquer mudança a ser feita no produto.
BACKLOG DA SPRINT	Uma lista de itens, escolhidos do Product Backlog, que a equipe se compromete a completar durante um Sprint (um período fixo durante o qual determinadas tarefas devem ser completadas). Guia a equipe sobre o trabalho a ser feito no Sprint atual, promovendo foco e comprometimento.
INCREMENTO	O conjunto de itens do Product Backlog completados durante um Sprint e as versões anteriores do produto. Deve estar em um estado utilizável e pronto para ser entregue ao cliente. Representa o progresso tangível em direção ao objetivo final do projeto.
QUADROS KANBAN	Um quadro visual para gerenciar o trabalho à medida que ele avança através de vários estágios do processo de desenvolvimento (por exemplo, "A Fazer", "Em Progresso", "Concluído"). Maximiza a eficiência do fluxo de trabalho ao limitar o trabalho em progresso e identificar gargalos.
GRÁFICO DE BURNDOWN	Gráficos que mostram o trabalho restante versus tempo. Eles podem ser usados tanto para o Sprint quanto para o Product Backlog. Fornecem uma visualização clara do progresso da equipe em direção à conclusão dos itens do backlog.
HISTÓRIAS DE USUÁRIO	Descrições curtas e simples da perspectiva do usuário final sobre uma funcionalidade específica que ele deseja que o produto tenha. Garantir que o desenvolvimento esteja focado nas necessidades e valores do usuário, facilitando a priorização e o planejamento do trabalho.

DEFINIÇÃO DE PRONTO	Um conjunto claro de critérios que especifica quando um item do backlog, como uma User Story ou uma tarefa, é considerado completo. Garante a qualidade e a consistência ao definir claramente o que é necessário para que o trabalho seja considerado finalizado.
ÉPICOS	Uma grande body of work que pode ser dividida em várias menores User Stories. É uma maneira de agrupar tarefas relacionadas que contribuem para um objetivo comum em larga escala. Facilita o gerenciamento e a priorização de grandes features ou funcionalidades que precisam ser desdobradas em tarefas mais gerenciáveis.

Métricas e Indicadores

As métricas ágeis são indicadores utilizados para medir o progresso, a eficiência e a eficácia das equipes e dos processos em ambientes que adotam metodologias ágeis de desenvolvimento, como Scrum, Kanban, entre outros. Elas fornecem insights valiosos sobre o desempenho da equipe, a qualidade do produto, a satisfação do cliente e outros aspectos críticos do processo de desenvolvimento de software. Vejamos exemplos de métricas ágeis:

MÉTRICAS E INDICADORES	DESCRIÇÃO
VELOCIDADE DA EQUIPE	Mede a quantidade de trabalho que uma equipe consegue completar durante um sprint. Geralmente, é calculada somando os pontos de história (ou qualquer outra unidade de medida) de todas as tarefas concluídas. Ajuda a prever a capacidade de entrega da equipe para futuros sprints, permitindo um planejamento mais preciso.
GRÁFICO DE BURNDOWN	Um gráfico que mostra a quantidade de trabalho restante versus tempo. Pode ser usado para sprints ou para o projeto como um todo. Fornece uma visualização clara de como a equipe está progredindo em direção à conclusão das tarefas dentro do prazo estabelecido.
LEAD/CYCLE TIME	Lead Time é o tempo total desde a solicitação até a entrega de uma tarefa. Cycle Time é o tempo que a tarefa leva para ser concluída, começando quando o trabalho efetivamente inicia. Indica a eficiência do processo de desenvolvimento, ajudando a identificar gargalos e a melhorar o fluxo de trabalho.
TAXA DE FALHAS EM PRODUÇÃO	Mede a frequência de falhas ou bugs que ocorrem no ambiente de produção. Ajuda a avaliar a qualidade do código e a eficácia das práticas de teste.
SATISFAÇÃO DO CLIENTE	Geralmente avaliada através de pesquisas ou Net Promoter Score (NPS), mede o quão satisfeitos os clientes estão com o produto ou serviço entregue. Fornece feedback direto sobre o valor que o produto está entregando aos usuários finais.
VAZÃO (THROUGHPUT)	Número de itens de trabalho (como Histórias de Usuário) completados em um período de tempo específico. Avalia a produtividade da equipe e pode ajudar a prever a entrega de futuras funcionalidades.
WORK IN PROGRESS (WIP)	Quantidade de tarefas em andamento em um determinado momento. Monitorar o WIP ajuda a evitar sobrecarga da equipe e a identificar gargalos no fluxo de trabalho.

MÉTRICA DE FELICIDADE	Avaliação do nível de satisfação ou felicidade da equipe. Equipes felizes tendem a ser mais produtivas e engajadas. Esta métrica ajuda a identificar problemas que podem estar afetando o moral da equipe.
RETENÇÃO DE CLIENTE	Mede a porcentagem de clientes que continuam utilizando o produto ou serviço ao longo do tempo. Indica o sucesso do produto em manter os usuários engajados e satisfeitos.

Esses indicadores fornecem uma visão abrangente do desempenho do projeto e da equipe, permitindo ajustes contínuos para otimizar os processos e as entregas. No entanto, é muito importante escolher os indicadores que melhor se alinham aos objetivos específicos do projeto e da organização para garantir que as métricas suportem, e não atrapalhem, a entrega de valor. *Entendido?*

Arquitetura Ágil

INCIDÊNCIA EM PROVA: BAIXÍSSIMA

A Arquitetura Ágil é um conjunto de valores, práticas e colaborações que apoiam ativamente o projeto e a arquitetura evolutiva de um sistema. Ela combina os princípios da arquitetura de software com os valores e práticas da metodologia ágil, buscando adaptar os princípios tradicionais de arquitetura de software para se alinhar melhor com as abordagens ágeis de desenvolvimento de software.

A arquitetura ágil busca reunir princípios e práticas que visam criar um software adaptável, flexível e resiliente em um ambiente de desenvolvimento ágil. Em contraste com a arquitetura tradicional, que define a estrutura geral do sistema antes do início do desenvolvimento, a arquitetura ágil se baseia em: planejamento incremental; colaboração e comunicação; qualidade e refatoração; e abordagem empírica.

CARACTERÍSTICA	DESCRIÇÃO
PLANEJAMENTO INCREMENTAL	A arquitetura do sistema é definida e implementada em iterações curtas e incrementais, permitindo adaptações às mudanças e feedback dos stakeholders. O foco está na entrega de valor funcional ao cliente o mais rápido possível.
COLABORAÇÃO E COMUNICAÇÃO	Arquitetos, desenvolvedores, testadores e stakeholders trabalham em conjunto para definir e refinar a arquitetura do sistema ao longo do projeto. A comunicação aberta e transparente é fundamental para garantir o alinhamento entre as diferentes partes interessadas.
QUALIDADE E REFATORAÇÃO	A qualidade da arquitetura é garantida através de práticas como revisão de código, testes automatizados e refatoração contínua. O código é constantemente aprimorado para eliminar duplicação, melhorar a legibilidade e aumentar a flexibilidade.
ABORDAGEM EMPÍRICA	A Arquitetura Ágil se baseia em feedback e aprendizado contínuo. As decisões de arquitetura são tomadas com base em dados e experimentos, ao invés de suposições ou preconceitos.
DESIGN EMERGENTE	A arquitetura evolui gradualmente ao longo do tempo, em vez de ser definida completamente no início do projeto. Ela emerge à medida que os requisitos são entendidos melhor e a equipe ganha experiência.
FEEDBACK RÁPIDO	A equipe busca obter feedback rápido sobre suas decisões arquiteturais por meio de revisões de código, testes e demonstrações frequentes do produto.
PADRÕES E PRÁTICAS	Utilização de padrões de projeto e práticas de engenharia de software que promovam a flexibilidade, escalabilidade, manutenibilidade e testabilidade do sistema.
REFATORAÇÃO CONSTANTE	A equipe está sempre disposta a refatorar o código e a arquitetura para melhorar sua qualidade e adaptá-la às mudanças nos requisitos e no ambiente.

PRINCÍPIOS	DESCRIÇÃO
SIMPLICIDADE	Favorecer soluções simples e diretas, evitando complexidade desnecessária.
FLEXIBILIDADE	Adaptar a arquitetura às mudanças nos requisitos e no ambiente de desenvolvimento.

TESTABILIDADE	Facilitar o teste e a validação da arquitetura.
REUTILIZABI-LIDADE	Criar componentes reutilizáveis para reduzir o tempo e o esforço de desenvolvimento.
EVOLUTIVIDADE	Permitir que a arquitetura evolua facilmente para atender às novas necessidades do negócio.

BENEFÍCIOS
Maior adaptabilidade a mudanças nos requisitos e no ambiente.
Maior qualidade do software através de refatoração contínua.
Maior velocidade de desenvolvimento e entrega de valor ao cliente.
Maior colaboração e comunicação entre as diferentes partes interessadas.
Maior flexibilidade para lidar com incertezas e riscos.

A Arquitetura Ágil não é uma solução mágica para todos os problemas de desenvolvimento de software. É importante ter em mente que a implementação da Arquitetura Ágil exige tempo, esforço e compromisso de toda a equipe. Não existe uma fórmula única para o sucesso. **O importante é adaptar os princípios da Arquitetura Ágil de acordo com as necessidades específicas da organização e buscar a melhoria contínua dos processos.**

Qualidade Ágil

INCIDÊNCIA EM PROVA: BAIXÍSSIMA

A Qualidade Ágil refere-se à abordagem de garantia de qualidade que está alinhada com os princípios e práticas ágeis de desenvolvimento de software. **Em um contexto ágil, a qualidade não é vista como um aspecto separado do processo de desenvolvimento, mas sim como uma responsabilidade compartilhada por toda a equipe.** A importância da Qualidade Ágil no contexto ágil é significativa por várias razões:

CARACTERÍSTICA	DESCRIÇÃO
ENTREGA DE VALOR AO CLIENTE	A qualidade do produto é essencial para atender às expectativas do cliente e garantir sua satisfação. Em uma abordagem ágil, a qualidade é priorizada desde o início do projeto, o que resulta em entregas mais rápidas e consistentes de valor ao cliente.
FEEDBACK CONTÍNUO	A Qualidade Ágil é fundamentada no princípio de feedback contínuo. Isso significa que a equipe está constantemente buscando feedback dos usuários, clientes e stakeholders para identificar áreas de melhoria e garantir que o produto atenda às suas necessidades e expectativas.
ADAPTAÇÃO RÁPIDA	Em ambientes ágeis, as mudanças são inevitáveis e frequentes. Uma abordagem de qualidade ágil permite que a equipe se adapte rapidamente a essas mudanças, ajustando o produto conforme necessário para manter ou melhorar sua qualidade.
COLABORAÇÃO E COMUNICAÇÃO	A Qualidade Ágil promove a colaboração e a comunicação eficaz entre todos os membros da equipe, incluindo desenvolvedores, testadores, analistas de negócios e stakeholders. Isso ajuda a garantir que todos tenham uma compreensão clara dos requisitos e expectativas de qualidade.
REDUÇÃO DE RISCOS	Uma abordagem de Qualidade Ágil ajuda a reduzir os riscos associados ao desenvolvimento de software, identificando e corrigindo problemas mais cedo no ciclo de vida do projeto. Isso minimiza a probabilidade de erros e retrabalho, aumentando assim a eficiência e eficácia do processo de desenvolvimento.
CULTURA DE MELHORIA E CONTÍNUA	A Qualidade Ágil promove uma cultura de melhoria contínua, onde a equipe está constantemente buscando maneiras de aprimorar seus processos, práticas e produtos. Isso leva a uma maior inovação, eficiência e qualidade geral do produto.

Os princípios do **Manifesto Ágil** influenciam diretamente a garantia de qualidade no desenvolvimento de software. Vejamos:

- **Satisfação do Cliente Através da Entrega Contínua de Software Funcionando:** esse princípio destaca a importância de entregar software funcional de forma contínua e incremental. Isso implica em priorizar a entrega de funcionalidades que agreguem valor ao cliente, o que está intrinsecamente ligado à qualidade do produto. A equipe busca garantir que cada incremento entregue atenda aos padrões de qualidade definidos e às expectativas do cliente.

- **Mudanças nos Requisitos São Bem-Vindas, Mesmo Tardiamente no Desenvolvimento:** esse princípio reconhece a inevitabilidade das mudanças nos requisitos do projeto e enfatiza a capacidade de adaptação do time. Isso influencia a garantia de qualidade ao permitir que a equipe ajuste continuamente o produto para atender às novas necessidades e expectativas do cliente, mantendo a qualidade do produto ao longo do tempo.

- **Entregue Software Funcionando Frequentemente, de Preferência em Escalas de Semanas a Meses:** esse princípio promove entregas frequentes e incrementais de software funcional. Essa abordagem permite que a equipe receba feedback regular dos usuários e stakeholders, possibilitando a identificação precoce de problemas e a garantia de qualidade contínua ao longo do ciclo de desenvolvimento.

- **Colaboração Entre Clientes e Desenvolvedores Ao Longo do Projeto:** esse princípio destaca a importância da comunicação e colaboração contínuas entre a equipe de desenvolvimento e os clientes ou stakeholders. Essa colaboração contribui para a garantia de qualidade ao garantir que as necessidades do cliente sejam compreendidas e traduzidas corretamente em funcionalidades de software de alta qualidade.

- **Construa Projetos em Torno de Indivíduos Motivados. Dê a Eles o Ambiente e o Suporte Necessários e Confie:** este princípio enfatiza a importância de uma equipe motivada. Uma equipe engajada e confiável é fundamental para garantir a qualidade do produto, pois são eles que desenvolvem, testam e mantêm o software. Ao fornecer o ambiente e o suporte adequados, e confiar na equipe para realizar o trabalho, a qualidade do produto é promovida.

Nesse contexto, testes ágeis são uma parte fundamental das metodologias ágeis e são projetados para se adaptar aos princípios e práticas dessas metodologias:

CARACTERÍSTICA	DESCRIÇÃO
TEST-DRIVEN DEVELOPMENT (TDD)	TDD é uma prática de desenvolvimento que envolve escrever testes automatizados antes mesmo de escrever o código de produção. Os passos básicos do TDD são: escrever um teste automatizado que falha, escrever o código de produção para fazer o teste passar e, em seguida, refatorar o código para melhorá-lo. O TDD promove o desenvolvimento incremental e a qualidade do código desde o início, garantindo que cada funcionalidade tenha testes automatizados associados a ela.
BEHAVIOR-DRIVER DEVELOPMENT (BDD)	BDD é uma abordagem que se concentra no comportamento esperado do software, expresso em termos de cenários de usuário. Os testes BDD são escritos em uma linguagem natural compreensível por todas as partes interessadas, como clientes, gerentes de projeto e desenvolvedores. Esses testes são então automatizados e executados para verificar se o software está se comportando conforme o esperado. O BDD promove uma compreensão compartilhada dos requisitos e uma abordagem centrada no usuário para o desenvolvimento.
AUTOMAÇÃO DE TESTES	A automação de testes é essencial para a entrega contínua de software de alta qualidade em ambientes ágeis. Os testes manuais consomem muito tempo e recursos, tornando-os impraticáveis para a entrega contínua. A automação de testes permite que a equipe execute testes de regressão de forma rápida e repetitiva, garantindo que as alterações no código não quebrem funcionalidades existentes. Isso aumenta a confiança na estabilidade do software e permite que a equipe mantenha um ritmo de entrega rápido e consistente.

AUTOMAÇÃO DA ENTREGA CONTÍNUA	Na entrega contínua, o software é entregue em ciclos curtos e frequentes. A automação de testes desempenha um papel crucial nesse processo, garantindo que o software entregue atenda aos padrões de qualidade definidos. Sem automação de testes eficaz, seria difícil ou impossível manter o ritmo de entrega contínua, pois os testes manuais seriam muito lentos e propensos a erros.
COBERTURA DE TESTES	A cobertura de testes é um aspecto importante dos Testes Ágeis. É essencial que a equipe tenha uma cobertura abrangente de testes automatizados para garantir que todas as funcionalidades do software sejam testadas de maneira adequada. Isso ajuda a identificar e corrigir problemas precocemente, reduzindo a probabilidade de bugs e garantindo a qualidade do produto final.

Método Ágil x Método Lean

INCIDÊNCIA EM PROVA: BAIXA

Método Lean, ou Método Enxuto, é uma filosofia de gestão focada na redução de desperdícios dentro de sistemas de manufatura enquanto simultaneamente maximiza a produção[1]. Originada no Sistema Toyota de Produção, esta abordagem busca melhorar a eficiência e a eficácia dos processos produtivos através da eliminação contínua de tudo que não agrega valor ao produto final na perspectiva do cliente. O Método Lean para desenvolvimento de software tem 7 princípios:

#	PRINCÍPIO	DESCRIÇÃO
PRINCÍPIO #1	ELIMINAR DESPERDÍCIO	Deve-se eliminar tudo aquilo que não é percebido pelo cliente, por não agregar valor para ele. Ex: passos extras, burocracia, documentação que não será lida, processo pesado, etc. Também existem aqueles desperdícios que são trabalhos parcialmente prontos - tudo que teve um começo, mas não teve fim e, portanto, não será utilizado.
PRINCÍPIO #2	AMPLIFICAR/CRIAR CONHECIMENTO	Deve-se garantir que o conhecimento sobre o software seja criado durante o desenvolvimento, em vez de ter uma lista de requisitos e/ou um layout recomendando como deve ser o resultado da aplicação antes do início de seu desenvolvimento.
PRINCÍPIO #3	FORTALECER O TIME / RESPEITAR AS PESSOAS	O software que está sendo produzido é uma espécie de espelho da equipe que o está desenvolvendo. Para que as pessoas se sintam motivadas e engajadas na atuação em equipe, eles precisam de respeito e confiança. Deve-se criar um ambiente onde a equipe trabalhe de forma auto-organizada e auto-dirigida, evitando micro-gerenciamento.
PRINCÍPIO #4	ENTREGAS RÁPIDAS	Uma dica importante é que, sem entregas rápidas, você não consegue receber um retorno, ou seja, você não consegue saber o que errou para tentar corrigir. Por isso, procurar a velocidade na entrega é uma maneira de garantir que o cliente tenha em mãos aquilo que ele precisava para hoje e não o que precisou.
PRINCÍPIO #5	CONSTRUIR / INTEGRAR QUALIDADE	Segundo os criadores da teoria, a qualidade é inegociável e deve ser entregue em duas dimensões: a integridade percebida e conceitual. A integridade percebida quer dizer que foi entregue ao cliente um produto usual, funcional, confiável. A integridade conceitual quer dizer que o sistema tem pontos centrais altamente coesos e fáceis.
PRINCÍPIO #6	OTIMIZAR O TODO	Deve-se entender que o software concluído é muito mais que a soma das partes entregues e verificar como ele está alinhado com os objetivos da empresa. O ideal não é olhar apenas para o desenvolvimento, mas para como aquele requisito está sendo atendido, como ele está sendo detalhado e repassado para entrar em desenvolvimento, entre outros.
PRINCÍPIO #7	ADIAR DECISÕES/ COMPROMISSOS	Deve-se diminuir as incertezas, retardando decisões até que elas sejam formuladas em cima de acontecimentos mais conhecidos, previsíveis e firmes.

[1] Sendo extremamente rigoroso, há diferenças de nomenclatura: Método Lean é uma filosofia de gestão centrada na criação de valor para o cliente final com o mínimo de desperdício possível, criada pela Toyota; Lean Manufacturing é a aplicação específica da filosofia Lean no contexto da produção industrial; e Lean IT é a aplicação dos princípios Lean no contexto da tecnologia da informação. No entanto, a maioria das provas utilizará o termo Método Lean para o contexto de tecnologia da informação.

Decisões tomadas tardiamente devem ser mais corretas, uma vez que as melhores são baseadas em fatos ocorridos e não em suposições ou especulações.

O Lean Manufacturing tem diversos benefícios:

- **Redução de Custos:** a diminuição de desperdícios e a melhoria da eficiência operacional levam a uma redução significativa nos custos de produção. Isso inclui custos associados a materiais, armazenamento, transporte, e tempo de inatividade, permitindo que a empresa seja mais competitiva com preços e margens de lucro;

- **Melhoria de Qualidade:** ao focar na eliminação de defeitos e na implementação de um processo de melhoria contínua, o Lean Manufacturing ajuda a aumentar a qualidade dos produtos. Isso reduz a quantidade de retrabalho e desperdício associado à correção de erros, além de melhorar a reputação da marca;

- **Aumento da Eficiência:** a implementação de processos mais eficientes e a eliminação de atividades que não agregam valor resultam em um aumento significativo da produtividade. Os trabalhadores podem focar suas energias em tarefas que realmente contribuem para o valor do produto, otimizando o uso do tempo e dos recursos;

- **Maior Flexibilidade:** a simplificação e a otimização dos processos de produção aumentam a flexibilidade da empresa, permitindo uma adaptação mais rápida às mudanças nas preferências dos consumidores e às condições do mercado. Isso é crucial em um ambiente empresarial cada vez mais volátil e competitivo;

- **Redução de Desperdícios:** o principal objetivo do Lean é identificar e eliminar desperdícios (atividades que não agregam valor ao produto final). Isso inclui desperdícios de tempo, material, movimento, espaço, e outros recursos, levando a processos mais eficientes e à redução de custos de produção;

- **Melhoria no Tempo de Entrega:** com processos mais eficientes e um sistema de produção puxada que responde diretamente à demanda do cliente, o Lean Manufacturing pode reduzir significativamente os tempos de entrega. Isso aumenta a satisfação do cliente e permite uma resposta mais rápida às mudanças do mercado;

- **Engajamento dos Funcionários:** o Método Lean promove um ambiente de trabalho inclusivo e colaborativo, onde os funcionários são encorajados a participar ativamente do processo de melhoria contínua. Isso pode aumentar a satisfação e o engajamento dos funcionários, reduzir a rotatividade e melhorar a cultura da empresa.

- **Sustentabilidade:** ao reduzir desperdícios e otimizar o uso de recursos, o Lean contribui para práticas de negócios mais sustentáveis. Isso não só reduz o impacto ambiental, mas também

melhora a imagem da empresa junto aos consumidores, que estão cada vez mais conscientes da sustentabilidade.

Ele serviu de base para o método ágil e tem várias características em comum, mas são diferentes. A tabela abaixo organiza um comparativo para vocês terem noção das diferenças:

CARACTERÍSTICA	MÉTODO LEAN	MÉTODO ÁGIL
ORIGEM E FOCO	Originou-se no Sistema Toyota de Produção na indústria automobilística japonesa, com foco principal na eliminação de desperdícios ("muda") para otimizar a eficiência do processo de produção. Embora tenha começado na manufatura, os princípios Lean foram adaptados para outras áreas, incluindo desenvolvimento de software e serviços.	Desenvolvido inicialmente para o campo do desenvolvimento de software como uma resposta às limitações dos métodos tradicionais de gerenciamento de projetos (como o modelo cascata). O foco é na adaptabilidade, na entrega incremental de produtos e na colaboração constante com o cliente.
PRINCÍPIOS E PRÁTICAS	Baseia-se em princípios como Kaizen (melhoria contínua), eliminação de desperdícios, e Just-In-Time. Práticas incluem mapeamento do fluxo de valor e otimização dos processos.	Baseia-se nos princípios do Manifesto Ágil, como colaboração cliente-desenvolvedor, resposta a mudanças e entrega incremental. Práticas incluem Scrum, Kanban, programação em pares, e integração contínua.
ABORDAGEM DE IMPLEMENTAÇÃO	Pode ser aplicado de maneira mais ampla além do desenvolvimento de produtos, incluindo processos administrativos e operacionais, com um forte foco em eficiência operacional e redução de desperdícios em todos os aspectos da organização.	Focado primariamente no desenvolvimento de software e projetos que beneficiam de uma abordagem iterativa e incremental. A implementação Ágil é caracterizada por sprints ou iterações, planejamento adaptativo e equipe multidisciplinar.
MEDIDA DE SUCESSO	O sucesso é medido pela eficiência do processo, a redução de desperdícios e a capacidade de entregar valor contínuo ao cliente com o mínimo de recursos possíveis.	O sucesso é frequentemente medido pela satisfação do cliente, a capacidade de responder rapidamente a mudanças e a entrega frequente de incrementos de software que funcionam.

Embora o Método Lean e o Método Ágil compartilhem princípios de melhoria contínua e eficiência, eles se diferenciam em suas origens, focos principais, e métodos de implementação. **O Método Lean é mais abrangente em termos de aplicação a processos de negócios e operações, enquanto o Método Ágil é mais específico para o desenvolvimento de software e projetos que se beneficiam de uma abordagem flexível e iterativa.**

Há também o Método Kaizen: conceito japonês que significa "melhoria contínua". **Diferente do Lean, que pode ser aplicado com projetos específicos para melhorias, o Kaizen é um processo contínuo e constante.** Envolve todos os empregados, desde a alta administração até os trabalhadores da linha de frente, na busca por pequenas melhorias diárias que, somadas, resultam em uma significativa melhoria da eficiência e qualidade com o passar do tempo.

O Kaizen pode usar muitas das mesmas ferramentas que o Lean, mas com ênfase na participação dos funcionários e na cultura de melhoria contínua. *Fechado?* Seguindo...

RESUMO

ESTAMOS DESCOBRINDO MANEIRAS MELHORES DE DESENVOLVER SOFTWARE, FAZENDO-O NÓS MESMOS E AJUDANDO OUTROS A FAZEREM O MESMO. ATRAVÉS DESTE TRABALHO. PASSAMOS A VALORIZAR:

INDIVÍDUOS E INTERAÇÕES	MAIS QUE	PROCESSOS E FERRAMENTAS
SOFTWARE EM FUNCIONAMENTO	MAIS QUE	DOCUMENTAÇÃO ABRANGENTE
COLABORAÇÃO COM O CLIENTE	MAIS QUE	NEGOCIAÇÃO DE CONTRATOS
RESPONDER A MUDANÇAS	MAIS QUE	SEGUIR UM PLANO

OU SEJA, MESMO HAVENDO VALOR NOS ITENS À DIREITA, VALORIZAMOS MAIS OS ITENS À ESQUERDA.

INDIVÍDUOS E ITERAÇÕES MAIS QUE PROCESSOS E FERRAMENTAS	Devemos entender que o desenvolvimento de software é uma atividade humana e que a qualidade da interação entre as pessoas pode resolver problemas crônicos de comunicação. Processos e Ferramentas são importantes, mas devem ser simples e uteis.
SOFTWARE EM FUNCIONAMENTO MAIS QUE DOCUMENTAÇÃO ABRANGENTE	O maior indicador de que sua equipe realmente construiu algo é software funcionando. Clientes querem é resultado e isso pode ser com software funcionando. Documentação também é importante, mas que seja somente o necessário e que agregue valor.
COLABORAÇÃO COM O CLIENTE MAIS QUE NEGOCIAÇÃO DE CONTRATOS	Devemos atuar em conjunto com o cliente e não "contra" ele ou ele "contra" a gente. O que deve acontecer é colaboração, tomada de decisões em conjunto e trabalho em equipe, fazendo que todos sejam um só em busca de um objetivo.
RESPONDER A MUDANÇAS MAIS QUE SEGUIR UM PLANO	Desenvolver software e produtos é um ambiente de alta incerteza e por isso não podemos nos debruçar em planos enormes e cheio de premissas. O que deve ser feito é aprender com as informações e feedbacks e adaptar o plano a todo momento.

PRINCIPAIS METODOLOGIAS ÁGEIS		
SCRUM	CRYSTAL	XP
TDD	ATDD	BDD
FDD	DDD	MDD
DSDM	ASD	KANBAN
BADM	AUP	AGILE MODELING
OSSD	SCRUMBAN	

NÓS SEGUIMOS ESSES PRINCÍPIOS...
Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada e software com valor agregado.
Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.
Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo.
Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.
Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho.
O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face.
Software funcionando é a medida primária de progresso.
Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.
Contínua atenção à excelência técnica e bom design aumenta a agilidade.
Simplicidade – a arte de maximizar a quantidade de trabalho não realizado – é essencial.

As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis.

Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

CRITÉRIO	MODELOS TRADICIONAIS	MODELOS ÁGEIS
PLANEJAMENTO	Comumente realizado em detalhe para todo o projeto em sua fase inicial.	Planejamento de alto nível no início do projeto e os detalhes são realizados durante o projeto. Não é necessário possuir um planejamento detalhado de todo o projeto. A restrição se dá apenas em possuir os detalhes do trabalho para a próxima iteração.
RISCOS	Pode exigir um grande esforço e equipe para atuar com os riscos de todo o projeto.	Prioriza os riscos gerais do projeto, mas foca principalmente nos riscos das próximas iterações, atuando assim em um escopo bem reduzido. A própria equipe atua com os riscos e pode obter apoio externo.
EQUIPE	Possui profissionais com papéis bem definidos, quantificada e mobilizada conforme o planejamento do projeto. A equipe executa o projeto guiado pelo Gerente de Projetos conforme o plano estabelecido.	Equipe multidisciplinar, multifuncional e auto-organizada. Ela decide como fazer e atua de forma colaborativa.
TEMPO DE ENTREGA	É realizado conforme o plano estabelecido e pode durar semanas, meses ou até mesmo anos.	Fixo e é conforme a definição de duração das iterações que comumente varia entre 1 e 4 semanas.
ACEITAÇÃO DE MUDANÇAS	Gerenciamento formal de mudanças, pois exige alteração do planejamento já realizado e geralmente precisa passar por aprovações formais de um ou mais níveis hierárquicos.	Mudanças são bem-vindas. Evita-se mudar o escopo da iteração em andamento, mas o escopo das futuras iterações podem ser replanejado conforme a necessidade do cliente.
PREVISIBILIDADE	Depende do intervalo de monitoramento e controle do projeto. Quanto mais curto, maior a chance de prever as ocorrências futuras. Quanto maior o intervalo, menor a chance de prever as ocorrências futuras.	Tende a ter uma grande previsibilidade futura devido à constante análise e feedback através das oportunidades de inspeção e adaptação providas pelo método.
RESULTADOS AO LONGO DO TEMPO	Tende a demorar a dar resultados a curto prazo, pois as entregas são geralmente realizadas ao final do projeto. Melhores resultados são apresentados em projetos de maior duração.	Gera resultados a curto, médio e longo prazo, pois atua com entregas antecipadas e de valor agregado e contínuo ao cliente.
APRESENTAÇÃO DE INFORMAÇÕES DO PROJETO	Geralmente de uma apresentação formal previamente agendada com os stakeholders em intervalos de tempo. As informações podem ser detalhadas ou não conforme a necessidade do público envolvido.	Geralmente informal e utiliza radiadores de informação no ambiente de trabalho durante todo o projeto, de modo que as informações do projeto fiquem visíveis e transparentes a toda equipe e envolvidos.
PRAZO DE ENTREGA	Conforme estabelecido no planejamento do projeto. No caso de mudanças aprovadas, varia conforme os impactos das solicitações	Conforme o tamanho da iteração e o planejamento das releases para as entregas significativas.

	e podem ser traumáticas aos envolvidos quanto às suas expectativas.	
DOCUMENTAÇÃO	Detalhada desde o início do projeto.	Abrangente no início e detalhada somente o necessário durante o projeto conforme os objetivos das iterações e releases.
ATUAÇÃO DO CLIENTE	Nas fases iniciais e nas principais validações do produto.	Durante todo o projeto, o cliente faz parte da equipe.
DISCUSSÕES E MELHORIAS	Geralmente em prazos longos através da realização de reuniões após uma grande etapa ou grande entrega do projeto.	Em prazos curtos, sempre ao final das iterações.
COMANDANTE	Gerente de Projetos.	Equipe do Projeto.
PAPÉIS	Claros e definidos.	Conforme a confiança na equipe e ambiente colaborativo.
PROCESSO	Guiado conforme o planejamento do projeto e nos processos estabelecidos no plano.	Empírico e guiado ao produto e às pessoas. Orientado à geração de valor e conforme priorização dos riscos.
RESULTADO	Melhor resultado em projetos com escopo muito bem definido e orientado a planejamento.	Melhor resultado em projetos cujo escopo é dinâmico e construído durante a execução do projeto.

CARACTERÍSTICA	LEAN	ÁGIL
OBCECADO COM...	DESPERDÍCIO	CLIENTES E MERCADOS
GERENCIA...	PROCESSOS	INCERTEZAS
ENTREGA DE...	VALOR	PRODUTO EM FUNCIONAMENTO
APLICA...	HEURÍSTICAS	PRINCÍPIOS
FOCA NO PROCESSO DE...	PADRONIZAÇÃO E CONFORMIDADE	AUTOGERENCIAMENTO P/ MAXIMIZAR AUTONOMIA

 PARA MAIS DICAS: WWW.INSTAGRAM.COM/PROFESSORDIEGOCARVALHO

QUESTÕES COMENTADAS – CESPE

1. **(CESPE / BANRISUL – 2022)** O modelo ágil não pode ser aplicado a qualquer processo de software, pois, para tanto, é necessário que o processo seja projetado de modo que suas características sejam modeladas como componentes e, em seguida, construídas dentro do contexto da arquitetura do sistema.

Comentários:

O modelo ágil pode ser aplicado a qualquer processo de software, desde que os princípios do pensamento ágil sejam seguidos. Além disso, não é necessário que o processo seja projetado de modo que suas características sejam modeladas como componentes - isso seria verdadeiro para desenvolvimento rápido e, não, desenvolvimento ágil.

Gabarito: Errado

---

2. **(CESPE / Petrobrás - 2022)** Entre as principais características dos métodos ágeis, destacam-se a maximização da documentação formal e o envolvimento dos clientes.

Comentários:

Conforme vimos em aula, o manifesto ágil preconiza que se valorize mais software em funcionamento do que documentação abrangente. Ou seja, não que se falar em documentação abrangente, e sim em uma documentação mais enxuta.

Gabarito: Errado

---

3. **(CESPE / TCE-ES – 2012)** Em virtude de as metodologias ágeis gerarem excessiva documentação, a gestão do conhecimento depende diretamente dos programadores envolvidos no projeto.

Comentários:

No ágil, documentação é descartável? Não, ela é útil para ajudar a comunicação e colaboração na equipe, melhorar a transferência de conhecimento, preservar informações históricas, satisfazer necessidades contratuais ou legais, entre outros. A documentação é importante, sim; mas valoriza-se mais o software em funcionamento. Logo, não é correto dizer que metodologias ágeis geram excessiva documentação.

Gabarito: Errado

---

4. **(CESPE / EBC – 2011)** O que os métodos ágeis buscam é como evitar as mudanças desde o início do projeto e não a melhor maneira de tratar essas mudanças.

Comentários:

Metodologias Ágeis são extremamente afeitas a mudanças de requisitos, adaptando-se a novos contextos e respondendo a cada modificação. Logo, mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento.

Gabarito: Errado

---

5. **(CESPE / BASA – 2010)** Desenvolvimento ágil de software (Agile Software Development) ou método ágil é aplicado, principalmente, a grandes corporações, uma vez que permite produzir grandes sistemas de forma ágil.

Comentários:

Segundo Sommerville: "*Todos os métodos têm limites, e os métodos ágeis são somente adequados para alguns tipos de desenvolvimento de sistema. Na minha opinião, eles são mais adequados para o desenvolvimento de sistemas de pequenas e médias empresas e produtos para computadores pessoais*". A questão afirma que ela é aplicada principalmente a grandes corporações. De fato, isso está errado! Ela ainda é aplicada principalmente a aplicações pequenas e médias, mas permite – sim – produzir grandes sistemas de forma ágil.

Gabarito: Errado

---

6. **(CESPE / TCU – 2010)** A agilidade não pode ser aplicada a todo e qualquer processo de software.

Comentários:

A agilidade pode – sim – ser aplicada a qualquer processo de software. Entretanto, para obtê-la, é essencial que seja projetado para que a equipe possa adaptar e alinhar (racionalizar) tarefas; possa conduzir o planejamento compreendendo a fluidez de uma abordagem do desenvolvimento ágil; e possa eliminar tudo, exceto os artefatos essenciais, conservando-os enxutos.

Gabarito: Errado

---

7. **(CESPE / UNIPAMPA – 2009)** XP, Scrum e Cristal são exemplos de modelos ágeis de desenvolvimento de sistemas.

Comentários:

SCRUM	CRYSTAL	XP
TDD	ATDD	BDD
FDD	DDD	MDD
DSDM	ASD	KANBAN
LEAN	AUP	AGILE MODELING
OSSD	SCRUMBAN	BADM

Todos são exemplos de metodologias ágeis (apesar do nome errado: Crystal e, não, Cristal).

Gabarito: Correto

---

8. **(CESPE / EBC – 2011)** Considerando o conceito de metodologia ágil em apreço, é correto afirmar que as seguintes metodologias são ágeis: XP (Extreme Programming), Scrum, Crystal, FDD (Feature Driven Development), DSDM (Dynamic Systems Development Method) e Open Source Software Development.

Comentários:

METODOLOGIAS ÁGEIS		
SCRUM	CRYSTAL	XP
TDD	ATDD	BDD
FDD	DDD	MDD
DSDM	ASD	KANBAN
LEAN	AUP	AGILE MODELING
OSSD	SCRUMBAN	BADM

De fato, todas essas são exemplos de metodologias ágeis.

Gabarito: Correto

---

9. **(CESPE / CNJ – 2013** O desenvolvimento ágil de sistemas consiste em uma linguagem de modelagem que permite aos desenvolvedores visualizarem os produtos de seu trabalho em gráficos padronizados.

Comentários:

Não, desenvolvimento ágil de sistemas não é uma linguagem de modelagem. *Sabe qual é um exemplo de linguagem de modelagem?* UML (Unified Modeling Language)!

Gabarito: Errado

---

10. **(CESPE / EBC – 2011)** É conveniente que o contrato, entre cliente e fornecedor, para o desenvolvimento de um sistema computacional, contenha a lista de requisitos para o software. Contudo, os métodos ágeis de desenvolvimento preconizam que o referido contrato estabeleça o preço, a ser pago pelo cliente, com base no tempo necessário para o desenvolvimento do sistema e não com base no conjunto de requisitos.

Comentários:

Segundo Martin Fowler, pode-se fixar um orçamento para o software antes de desenvolvê-lo. A abordagem ágil comum é fixar tempo e preço, deixando o escopo variar (os requisitos são variáveis) de forma controlada. É o famoso: "*Tempo fixo, escopo variável*".

Gabarito: Correto

---

11. **(CESPE / MPOG – 2015)** Metodologias de desenvolvimento ágil enfocam atividades de projeto e implementação, desconsiderando as atividades de elicitação de requisitos e a produção de documentação.

Comentários:

É absurdo pensar que se desconsidera atividades de elicitação de requisitos – não há o que se discutir nesse ponto. Além disso, o Manifesto Ágil afirma que, mesmo havendo valor na documentação extensa de software, valoriza-se mais o software em funcionamento. Em outras palavras, é errado afirmar que se desconsidera a produção de documentação, tendo em vista que há uma codificação não formal.

Gabarito: Errado

---

12. **(CESPE / TRE-PI – 2008)** No que se refere a métodos ágeis de desenvolvimento de sistemas, assinale a opção correta.

a) A aplicação de método ágil para desenvolvimento de grandes sistemas pode enfrentar dificuldades que o tornem inviável.

b) O documento de requisitos, apesar de abordar um conjunto pequeno de funcionalidades, deve especificar toda a necessidade do usuário.

c) O sistema é construído em pequenos blocos, que irão compor uma versão a ser entregue aos usuários.

d) A documentação de projeto deve ser feita pelo próprio desenvolvedor, seguindo padrões simplificados.

e) Para atingir os objetivos de agilidade exigidos, os desenvolvedores devem seguir processos simplificados para a construção do software.

Comentários:

a) Correto. Aqui temos a teoria e a prática: no início, tanto a teoria quanto a prática não recomendavam que as metodologias ágeis fossem aplicadas a sistemas grandes. No entanto, atualmente, isso já não é mais uma limitação. Hoje em dia, as metodologias ágeis adquiriram maturidade suficiente para desenvolver sistemas grandes e complexos. Porém, isso ainda está na teoria, por isso as questões ainda cobram.

b) Errado. Em metodologias ágeis, há uma documentação abrangente no início e detalhada somente o necessário durante o projeto conforme os objetivos das iterações e releases. No entanto, não existe um "Documento de Requisitos" - isso é coisa de metodologias tradicionais. Além disso, nenhum documento nunca conseguirá especificar todas as necessidades dos usuários.

c) Correto. Não vislumbro qualquer erro nesse item. Ora, metodologias ágeis são iterativas e incrementais, dividindo o sistema em pequenas partes e sempre entregando versões que agreguem valor aos usuários. Se você errou esse item, fique tranquilo - o vacilo foi da banca!

d) Errado. *Documentação de Projeto?* Isso é coisa de metodologias tradicionais. Além disso, o Product Backlog é feito pelo Product Owner, mas isso não é assunto para essa aula.

e) Errado. Não existe esse negócio de "objetivos de agilidade exigidos". Isso soa como se existissem métricas de agilidade que tivessem que ser atingidas.

Gabarito: Letra A

---

13. (CESPE / TCE-PR – 2016) Os métodos ágeis para o desenvolvimento de software representam uma evolução da engenharia de software tradicional, uma vez que são aplicáveis a todos os tipos de projetos, produtos, pessoas e situações.

Comentários:

É um erro comum achar que metodologias ágeis servem para todas as ocasiões. Não confundam: agilidade pode ser aplicada a todo processo de software; mas métodos ágeis não funcionam bem em qualquer situação.

Gabarito: Errado

---

14. (CESPE / TCE-PR – 2016) Um dos princípios de agilidade da Agile Alliance dispõe que a entrega completa de um software garante a satisfação do cliente.

Comentários:

De acordo com o 1º Princípio Ágil: Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado. Percebam que não existe entrega completa, mas contínua.

Gabarito: Errado

---

15. **(CESPE / Ministério da Economia – 2020)** Os modelos ágeis de desenvolvimento de software dão grande ênfase às definições de atividades e aos processos e pouca ênfase à pragmática e ao fator humano.

Comentários:

Que isso? É exatamente o oposto! O foco é maior no pragmatismo, empirismo e fatores humanos do que nas definições de atividades ou processos.

Gabarito: Errado

---

16. **(CESPE / MEC – 2015)** Acatar as mudanças de requisitos, ainda que o desenvolvimento já esteja avançado, é um dos princípios do Manifesto Ágil.

Comentários:

De acordo com o 2º princípio ágil: mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.

Gabarito: Correto

---

17. **(CESPE / TRT17 – 2013)** Em um desenvolvimento ágil que segue o manifesto ágil, não se deve aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis não se adequam a mudanças não planejadas.

Comentários:

De acordo com o 2º princípio ágil: mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.

Gabarito: Errado

---

18. **(CESPE / EBSERH – 2018)** Nas metodologias de desenvolvimento ágeis, mudanças em requisitos são bem recebidas, mesmo em fases mais avançadas do desenvolvimento.

Comentários:

De acordo com o 2º princípio ágil: mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.

Gabarito: Correto

---

19. **(CESPE / SEDF – 2017)** Lean manufacturing e kaizen são exemplos de ferramentas de gestão da qualidade aplicadas para o aperfeiçoamento de organizações e preveem a realização de diagnósticos e implementação de melhorias.

Comentários:

Perfeito! Tanto o Lean Manufacturing quanto o Kaizen são metodologias focadas na melhoria contínua e na eficiência dos processos dentro das organizações. O Método Lean visa eliminar desperdícios em todos os aspectos da produção, incluindo tempo, mão-de-obra, e recursos materiais e o Kaizen é aplicado em projetos específicos para melhorias, em um processo contínuo e constante.

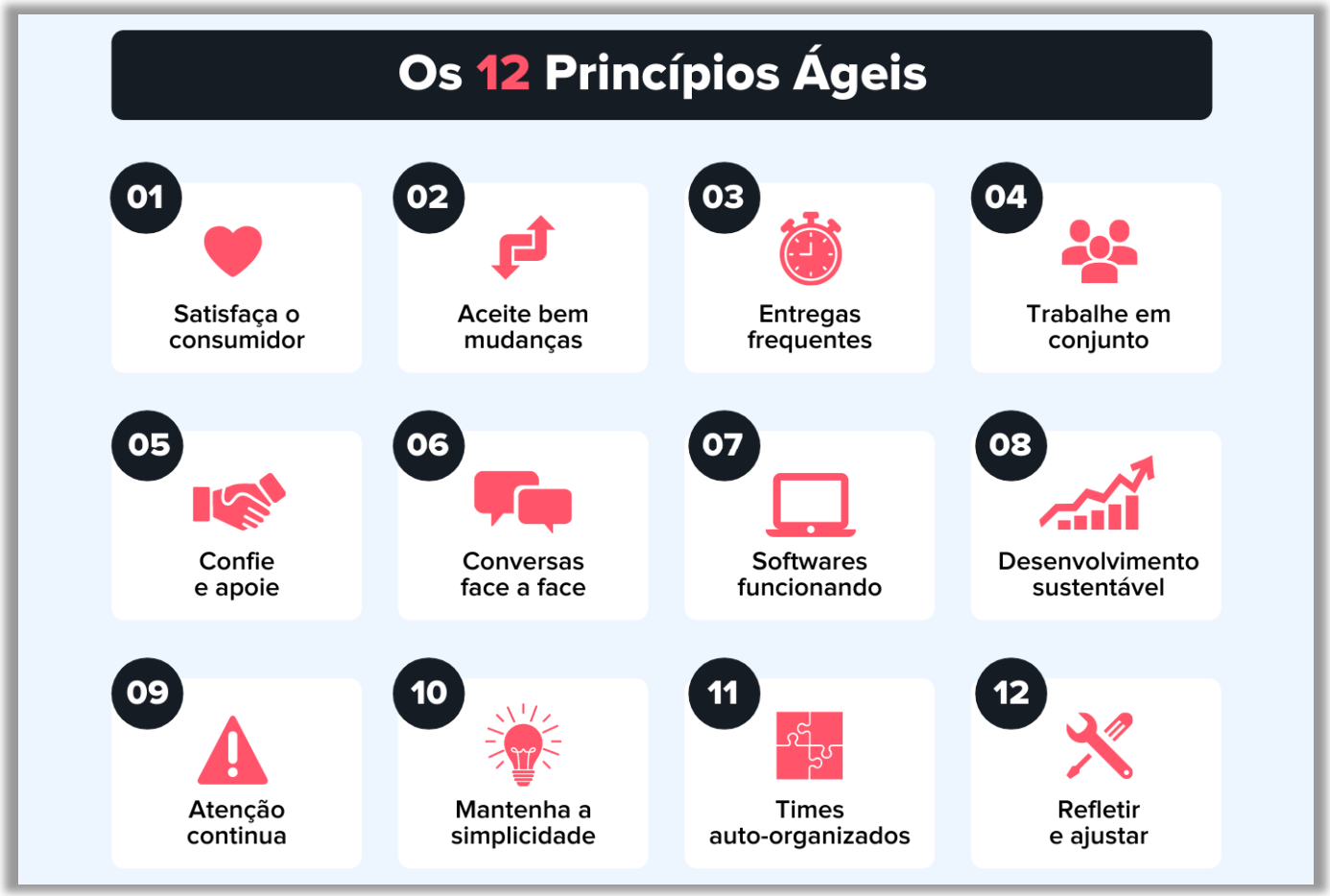
Gabarito: Correto

QUESTÕES COMENTADAS – FCC

1. **(FCC / SEFAZ-AP – 2022)** Dentre os doze Princípios do Manifesto Ágil, incluem-se:

a) funcionalidade, satisfação do cliente e trabalho em conjunto.
b) respeito ao cliente, economia de recursos e paralelismo.
c) resiliência, motivação e trabalho em pares.
d) simplicidade, motivação e paralelismo.
e) especificidade, longevidade do *software* e prazos curtos.

Comentários:



Trata-se de funcionalidade (software funcionando), satisfação do cliente (satisfaça o consumidor) e trabalho em conjunto.

Gabarito: Letra A

---

2. **(FCC / SEFAZ-AP – 2022)** Dentre os doze Princípios do Manifesto Ágil, incluem-se:

a) respeito ao cliente, economia de recursos e paralelismo.
b) resiliência, motivação e trabalho em pares.
c) simplicidade, motivação e paralelismo.
d) especificidade, longevidade do software e prazos curtos.
e) funcionalidade, satisfação do cliente e trabalho em conjunto.

Comentários:

(a) Errado. "*Respeito ao cliente*" e "*economia de recursos*" não são explicitamente citados no Manifesto Ágil, e "*paralelismo*" não é um termo geralmente associado com os princípios ágeis;

b) Errado. "*Resiliência*" não é um dos princípios ágeis. "*Motivação*" é uma interpretação livre do princípio que fala em pessoas motivadas e "*trabalho em pares*" pode ser relacionado com métodos ágeis como a Programação em Pares, mas não é um princípio por si só;

c) Errado. "*Simplicidade*" é explicitamente mencionado como o princípio de maximizar a quantidade de trabalho não realizado. "*Motivação*" pode ser relacionado ao princípio de construir projetos em torno de indivíduos motivados. "*Paralelismo*" novamente não é um termo usado;

d) Errado. "*Especificidade*" e "*longevidade do software*" não são termos usados nos princípios ágeis. "Prazos curtos" se assemelha ao princípio de entregas frequentes, mas não é o termo utilizado;

e) Correto. "*Funcionalidade*" não é um princípio expresso, mas "*satisfação do cliente*" é o primeiro princípio do Manifesto Ágil, e "*trabalho em conjunto*" reflete o princípio de colaboração constante com o cliente.

Gabarito: Letra E

QUESTÕES COMENTADAS – FGV

1. **(FGV / TJ-RN - 2023)** A Equipe de Tecnologia da Informação (ETI) de um tribunal está envolvida no projeto TERC cujo ciclo de vida segue uma abordagem adaptativa (ágil). Considerando o ciclo de vida empregado no TERC, a ETI:

   a) especifica os requisitos do produto final antes do início do desenvolvimento;
   b) solicita o envolvimento das partes interessadas chave em marcos específicos;
   c) controla os riscos à medida em que surgem requisitos e restrições;
   d) ajusta o ciclo de vida do projeto ao ciclo de vida do produto;
   e) estabelece as fases do projeto com base em um ciclo de vida de desenvolvimento preditivo.

Comentários:

(a) Errado, isso seria típico do desenvolvimento em cascata; (b) Errado, ele solicita o envolvimento das partes interessadas chave ao longo de todo o desenvolvimento; (c) Correto, riscos são controlados à medida em que surgem requisitos e restrições; (d) Errado, isso seria típico do desenvolvimento em cascata; (e) Errado, isso também seria típico do desenvolvimento em cascata.

Gabarito: Letra C

---

2. **(FGV / IMBEL – 2021)** Com referência aos valores do The Agile Manifesto, analise as afirmativas a seguir.

   I. Processos e ferramentas mais que indivíduos e interação entre eles.
   II. Software em funcionamento mais que documentação abrangente.
   III. Colaboração do cliente mais que negociação de contratos.
   IV. Seguir um plano mais que responder a mudanças.

   Está correto o que se afirma em:

   a) I e II, somente
   b) II e III, somente.
   c) III e IV, somente.
   d) I e IV, somente.
   e) II e IV, somente.

Comentários:

(I) Errado, trata-se do inverso; (II) Correto; (III) Correto; (IV) Errado, trata-se do inverso.

Gabarito: Letra B

---

3. **(FGV / MPE-MS – 2013)** Considerando a caracterização de agilidade e processo de desenvolvimento ágil, segundo Pressman, analise as afirmativas a seguir.

I. Um processo ágil de software deve ser incrementalmente adaptável.
II. Um processo ágil de software permite que as pessoas e a equipe se moldem a ele com facilidade.
III. Os conceitos ágeis são efetivos, pois diminuem a imprevisibilidade sistêmica ao enfatizar entregas em prazos curtos.

a) se somente a afirmativa I estiver correta.
b) se somente a afirmativa II estiver correta.
c) se somente a afirmativa III estiver correta.
d) se somente as afirmativas I e II estiverem corretas.
e) se todas as afirmativas estiverem corretas.

Comentários:

(I) Correto, idealmente ele deve se adaptar de forma incremental; (II) Errado, a agilidade não tem relação com a facilidade da equipe de se moldar; (III) Errado, mas questão polêmica! Eu acredito que os conceitos ágeis diminuem a imprevisibilidade. Já alguns argumentam que conceitos ágeis não diminuem a imprevisibilidade, na verdade eles aceitam que a imprevisibilidade é inevitável e, dessa forma, provêm métodos de se adaptar às mudanças rapidamente.

Gabarito: Letra A

---

4. **(FGV / PGE-RO – 2015)** Durante 5 anos gerenciando o desenvolvimento de sistemas de informação, Claudia teve que lidar com diversas insatisfações de seus usuários pois os sistemas não atendiam as suas necessidades. Claudia decidiu, então, implantar métodos ágeis de desenvolvimento e definiu os seguintes princípios:

I. Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento.

II. O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através da documentação.

III. Simplicidade é essencial.

Dentre os princípios definidos por Claudia, o que infringe os princípios do manifesto para Desenvolvimento Ágil de Software é o que se afirma em:

a) somente I;
b) somente II;

c) somente III;
d) somente I e III;
e) I, II e III.

Comentários:

NÓS SEGUIMOS ESSES PRINCÍPIOS...
Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.
O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face.
Simplicidade – a arte de maximizar a quantidade de trabalho não realizado – é essencial.

(I) Correto, mudanças são sempre bem-vindas; (II) Errado, o método mais eficiente é frente-a-frente; (III) Correto. Como a questão pede os princípios que infringem o manifesto ágil, trata-se da segunda opção.

Gabarito: Letra B

---

5. **(FGV / TJ-RO – 2015)** O manifesto ágil tem por princípio que:

a) mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento;

b) a contínua atenção à excelência técnica reduz a agilidade;

c) a redução do backlog é a medida primária de progresso;

d) as melhores arquiteturas, requisitos e designs emergem de equipes que possuem um bom líder;

e) pessoas de negócio e desenvolvedores devem trabalhar em ambientes separados para reduzir as interferências no processo de desenvolvimento.

Comentários:

NÓS SEGUIMOS ESSES PRINCÍPIOS...
Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.
Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.
Software funcionando é a medida primária de progresso.
Contínua atenção à excelência técnica e bom design aumenta a agilidade.

As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis.

O único princípio correto é que mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento.

Gabarito: Letra A

---

6. **(FGV / TJ-GO – 2014)** Escreva O Manifesto Ágil lista valores seguidos por desenvolvedores com a finalidade de melhorar a maneira pela qual o software é desenvolvido. A alternativa que se encontra no manifesto é:

a) seguir um plano mais que responder a mudanças;
b) indivíduos e interações mais que processos e ferramentas;
c) documentação abrangente mais que software em funcionamento;
d) negociação de contratos mais que colaboração com o cliente;
e) negociação de contratos mais que indivíduos e interações.

Comentários:

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

Gabarito: Letra B

---

7. **(FGV / Câmara Municipal de Caruaru-PE – 2015)** O desenvolvimento ágil de software é guiado por metodologias que compartilham um conjunto comum de valores e de princípios, conforme definido pelo Manifesto Ágil. Assinale a opção que indica um princípio do desenvolvimento ágil.

a) As mudanças nos requisitos devem ocorrer dentro do quadro de tempo estabelecido para a iteração.

b) O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é por meio de conversa face a face.

c) Os intervalos regulares devem ser evitados para tornar a equipe mais eficaz e maximizar a quantidade de trabalho realizado.

d) As pessoas de negócio e desenvolvedores devem interagir somente no início de cada iteração.

e) A entrega contínua e adiantada de software, mesmo que o conjunto de funcionalidades desenvolvidas não agregue valor, deve ser feita para satisfazer o cliente.

Comentários:

(a) Errado, mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento; (b) Correto; (c) Errado, em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo; (d) Errado, pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto; (e) Errado, nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado.

Gabarito: Letra B

---

8. **(FGV / PROCEMPA – 2014)** O Manifesto Ágil é uma declaração de princípios que fundamentam o desenvolvimento ágil de software. A respeito desses princípios, assinale a afirmativa correta:

a) As melhores arquiteturas, requisitos e designs emergem de equipes lideradas pelo profissional mais sênior.

b) Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

c) Pessoas de negócio e desenvolvedores devem trabalhar separadamente por todo o projeto.

d) Entregar software quando há poucas semanas de desenvolvimento deve ser evitado para não afetar a satisfação do cliente.

e) Mudanças nos requisitos são bem-vindas, desde que não impactem o desenvolvimento.

Comentários:

(a) Errado, as melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis; (b) Correto; (c) Errado, pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto; (d) Errado, entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo; (e) Errado, mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento.

Gabarito: Letra B

---

9. **(FGV / DPE-RO – 2015)** O Manifesto Ágil é uma declaração que reúne os princípios e práticas que fundamentam o desenvolvimento ágil de software. É um dos princípios desse manifesto:

a) defeitos no software são a medida primária de progresso;

b) pessoas de negócio e desenvolvedores devem trabalhar isoladamente e se reunir somente ao final de cada iteração para validação do software;

c) atenção contínua à excelência técnica deve ser evitada para não afetar a agilidade uma vez que simplicidade é essencial;

d) os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente evitando interrupções e intervalos regulares;

e) as melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis.

Comentários:

(a) Errado, software funcionando é a medida primária de progresso; (b) Errado, pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto; (c) Errado, contínua atenção à excelência técnica e bom design aumenta a agilidade; (d) Errado, patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente; (e) Correto.

Gabarito: Letra E

---

10. **(FGV / BANESTES – 2018)** Um dos valores relacionados ao ambiente ágil de desenvolvimento é:

a) documentação abrangente mais que software funcional;
b) negociação de contratos mais que colaboração do cliente;
c) processos e ferramentas mais que indivíduos e iterações;
d) rapidez na construção mais que excelência técnica;
e) responder a mudanças mais que seguir um plano.

Comentários:

(1) Indivíduos e interações acima de processos e ferramentas; (2) Software em funcionamento acima de documentação abrangente; (3) Colaboração com o cliente acima de negociação de contratos; **(4) Responder a mudanças acima de seguir fielmente um plano**.

Gabarito: Letra E

---

11. **(FGV / BANESTES – 2018)** Com relação aos valores relacionados ao desenvolvimento ágil de software, NÃO se pode incluir:

a) colaboração do cliente mais que negociação de contratos;
b) indivíduos e iterações mais que processos e ferramentas;
c) rapidez na construção mais que excelência técnica;
d) responder a mudanças mais que seguir um plano;
e) software funcional mais que documentação abrangente.

Comentários:

(1) Indivíduos e interações acima de processos e ferramentas; (2) Software em funcionamento acima de documentação abrangente; (3) Colaboração com o cliente acima de negociação de contratos; (4) Responder a mudanças acima de seguir fielmente um plano. Notem que *rapidez na construção mais que excelência técnica* não é um dos valores do manifesto ágil.

Gabarito: Letra C

---

12. **(FGV / AL-RO – 2018)** Para o desenvolvimento do Sistema de Informações ao Cidadão (SIC), foi decidida a utilização de uma metodologia ágil. Segundo o Manifesto Ágil, esta decisão indica que foi dado maior valor:

a) aos processos e ferramentas.
b) à resposta a modificações.
c) à documentação abrangente.
d) à negociação do contrato.
e) ao cumprimento do plano.

Comentários:

(1) Indivíduos e interações acima de processos e ferramentas; (2) Software em funcionamento acima de documentação abrangente; (3) Colaboração com o cliente acima de negociação de contratos; **(4) Responder a mudanças acima de seguir fielmente um plano.**

Gabarito: Letra B

---

13. **(FGV / TJDFT – 2022)** Uma equipe de analista de sistemas está desenvolvendo o software ProgramaTJ aplicando a metodologia Lean. A equipe decidiu implementar apenas as funcionalidades formalmente requisitadas pelo cliente, evitando adicionar qualquer funcionalidade extra à ProgramaTJ por conta própria. Essa decisão da equipe remete, de forma direta, ao princípio da metodologia Lean para o desenvolvimento de software de:

a) otimização do todo;
b) adiar comprometimento;
c) eliminação de desperdícios;
d) respeitar as pessoas;

e) criação de conhecimento.

Comentários:

(a) Errado. Embora relevante, esta opção não é a mais diretamente relacionada à decisão da equipe de se concentrar apenas nas funcionalidades requisitadas pelo cliente;

(b) Errado. Este princípio é valioso no desenvolvimento ágil, mas não é o que melhor descreve a decisão de evitar funcionalidades não solicitadas;

(c) Correto. Central para a filosofia Lean, a eliminação de desperdícios envolve identificar e remover qualquer coisa que não agregue valor ao cliente. No contexto do desenvolvimento de software, isso inclui evitar trabalhar em funcionalidades que não foram explicitamente requisitadas pelo cliente, pois representariam um esforço que não contribui diretamente para o valor percebido pelo cliente. Esta opção é a mais diretamente relacionada com a decisão da equipe descrita na questão;

(d) Errado. Embora criar um ambiente de trabalho respeitoso seja crucial para o sucesso de qualquer projeto, este princípio não está diretamente relacionado à decisão de se concentrar apenas nas funcionalidades requisitadas pelo cliente;

(d) Errado. Enquanto esse princípio é importante para a melhoria contínua, ele não se aplica diretamente à decisão da equipe de evitar adicionar funcionalidades não requisitadas.

Gabarito: Letra C

QUESTÕES COMENTADAS – DIVERSAS BANCAS

1. **(ADMTEC / Prefeitura de Palmeira dos Índios-AL – 2024)** Analise as informações a seguir:

I. Entre as metodologias para desenvolvimento de software mais conhecidas e utilizadas atualmente, o Modelo Waterfall (Cascata) ainda se destaca por trabalhar em 5 fases: Requerimento, Projeto, Implementação e Verificação e Manutenção.

II. Entre as metodologias para desenvolvimento de software mais conhecidas e utilizadas atualmente, a metodologia Lean ganha a atenção dos desenvolvedores por se basear em 5 princípios: Reduzir o desperdício; Postergar as decisões; Agilizar as entregas; Empoderar as equipes e Otimizar o todo.

Marque a alternativa CORRETA:

a) As duas afirmativas são verdadeiras.
b) A afirmativa I é verdadeira, e a II é falsa.
c) A afirmativa II é verdadeira, e a I é falsa.
d) As duas afirmativas são falsas.

Comentários:

(I) Errado. A banca considerou esse item como correto, mas o modelo em cascata é utilizado cada vez menos, logo discordo da banca; (II) Errado. Na verdade, são sete princípios: eliminar desperdício; criar conhecimento; fortalecer o time; entregas rápidas; construir qualidade; otimizar o todo; e adiar decisões.

Gabarito: Letra B

---

2. **(OBJETIVO / Prefeitura de Piratininga-SP – 2023)** O método Lean é um dos principais métodos ágeis utilizados na gestão de projetos. Sobre essa metodologia, assinalar a alternativa CORRETA:

a) É utilizada na gestão de projetos que não tenham prazo de entrega, utilizando intervalos de tempo para desenvolver cada etapa.

b) É composta por checklists, oferecendo uma visão global do projeto. É específica para alguns tipos de negócio, pois busca a revolução dos processos.

c) É uma boa alternativa para criar objetivos realistas e possíveis para a situação de cada empresa, baseando-se em cinco primórdios, os quais são indicados pelas letras do seu nome.

d) É utilizada para projetos mais objetivos ou reduzidos, e identifica eficientemente os desperdícios.

Comentários:

(a) Errado. Isso não descreve corretamente o Lean. O método Lean pode ser aplicado a projetos com prazos de entrega definidos e visa a eficiência em todas as etapas, não se limitando a projetos sem prazos;

(b) Errado. Embora o Lean possa envolver a utilização de checklists para garantir que os processos sejam seguidos, dizer que é específico para alguns tipos de negócio é limitante. O Lean é flexível e pode ser adaptado para diversos tipos de projetos e indústrias;

(c) Errado. Esta descrição parece confundir o Lean com outra metodologia. O Lean não se baseia em "cinco primórdios" indicados pelas letras de seu nome;

(d) Correto. Esta é a descrição mais precisa do método Lean entre as alternativas apresentadas. O Lean é conhecido por sua capacidade de identificar e eliminar desperdícios, o que o torna adequado para projetos que buscam eficiência e eficácia, independentemente do seu tamanho.

Gabarito: Letra D

---

3. **(IADES / CRF-TO – 2023)** Assinale a alternativa que apresenta um princípio da metodologia Lean de desenvolvimento de software.

a) Adiar decisões o máximo possível.
b) Abraçar o desperdício.
c) Entregar com atraso.
d) Compartimentalizar o conhecimento.
e) Fortalecer a hierarquia.

Comentários:

(a) Correto. Este princípio está alinhado com a metodologia Lean, pois sugere que as decisões devem ser tomadas no último momento responsável, permitindo que sejam baseadas na maior quantidade de informações disponíveis e evitando suposições prematuras que podem levar a desperdícios.

(b) Errado. Este não é um princípio da metodologia Lean. Pelo contrário, a metodologia Lean foca na eliminação de desperdícios em todas as suas formas.

(c) Errado. Este também não é um princípio da metodologia Lean. A Lean busca maximizar o fluxo de trabalho para garantir entregas rápidas e eficientes.

(d) Errado. Este não é um princípio da metodologia Lean. Na verdade, a Lean enfatiza a importância da transparência e do compartilhamento de conhecimento para melhorar a eficiência e a colaboração.

(e) Errado. Este não é um princípio da metodologia Lean. A metodologia Lean promove a autonomia das equipes e a descentralização das decisões.

Gabarito: Letra A

---

4. **(FEPESE / EPAGRI – 2023)** Qual metodologia ágil tem como foco a excelência na qualidade e aumento da velocidade dos processos e eliminar o desbarato?

a) OpenUP
b) Pragmatic Programming
c) Test Driven Development
d) Lean Software Development
e) Microsoft Solutions Framework

Comentários:

(a) Errado. OpenUP é uma metodologia ágil que enfatiza práticas iterativas e incrementais, mas o seu foco não está especificamente na eliminação de desperdícios, excelência na qualidade ou no aumento da velocidade dos processos de maneira tão direta quanto o Lean Software Development;

(b) Errado. Refere-se a uma abordagem ou filosofia de desenvolvimento de software focada na simplicidade, clareza e pragmatismo. Embora possa encorajar a eficiência, não é uma metodologia com o foco específico mencionado na questão;

(c) Errado. TDD é uma técnica de desenvolvimento de software que envolve escrever testes antes de escrever o código que será testado. O foco não se concentra especificamente na eliminação de desperdícios ou no aumento da velocidade dos processos de forma geral;

(d) Correto. Lean Software Development é baseada nos princípios da manufatura enxuta (Lean Manufacturing) e se concentra na entrega de valor para o cliente, eliminando desperdícios, melhorando a qualidade e aumentando a eficiência dos processos de desenvolvimento de software;

(e) Errado. Trata-se de um conjunto de princípios, modelos e práticas de gestão de projetos destinados a entregar soluções tecnológicas. Não tem como foco principal a eliminação de desperdícios e a excelência na qualidade da mesma forma que o Lean Software Development.

Gabarito: Letra D

---

5. **(QUADRIX / PRODAM-AM – 2022)** A partir dos princípios abordados pela metodologia ágil Lean, assinale a alternativa que apresenta uma forma de desperdício na qual um número excessivo de mudanças de contexto reduz a produtividade.

a) esperas por requisitos
b) troca de tarefas
c) construção da integridade
d) defeitos
e) antecipação das funcionalidades

Comentários:

(a) Errado. Embora as esperas possam ser consideradas um desperdício dentro da metodologia Lean (tempo ocioso esperando por informações ou decisões), essa opção não aborda diretamente o impacto das mudanças de contexto;

(b) Correto. Esse é um exemplo claro de desperdício identificado pela metodologia Lean, conhecido como "task switching" ou "multitasking". A constante troca de tarefas exige que a mente se reajuste a diferentes contextos, o que pode reduzir significativamente a eficiência e a produtividade;

(c) Errado. A construção da integridade é um princípio positivo, focado em manter a qualidade e a consistência do produto, e não é considerado uma forma de desperdício;

(d) Errado. Os defeitos são, sem dúvida, uma forma de desperdício na metodologia Lean, pois exigem retrabalho. No entanto, eles não estão diretamente relacionados à questão das mudanças de contexto;

(e) Errado. Desenvolver funcionalidades antes de serem necessárias pode levar a desperdícios, mas essa opção não capta especificamente o conceito de desperdício por mudanças excessivas de contexto.

Gabarito: Letra B

---

6. **(QUADRIX / PRODAM-AM – 2022)** Com relação à metodologia ágil para o desenvolvimento de software Lean, assinale a alternativa correta.

a) O excesso de processos não é considerado um desperdício, porque eles não demandam recursos.

b) Os processos complexos não aumentam a quantidade de documentos, por isso não caracterizam desperdício.

c) O pensamento Lean foca em oferecer o que o cliente quer, onde e quando ele quiser, sem haver qualquer desperdício.

d) Para a metodologia Lean, o desenvolvimento rápido do software só apresenta desvantagens, pois alguns processos são atropelados.

e) Não há necessidade da realização de testes, uma vez que os softwares desenvolvidos por meio dessa metodologia são eficazes.

Comentários:

(a) Errado. O excesso de processos é, de fato, considerado um desperdício na metodologia Lean, pois demandam recursos desnecessários e podem retardar a entrega do produto;

(b) Errado. Processos complexos aumentam a quantidade de documentos e a burocracia, o que é considerado desperdício na metodologia Lean, pois distrai a equipe do foco em entregar valor ao cliente.

(c) Correto. Esta alternativa capta a essência do pensamento Lean, que é entregar exatamente o que o cliente quer, de forma eficiente e sem desperdício, no momento certo e no local certo.

(d) Errado. A metodologia Lean não vê o desenvolvimento rápido como desvantajoso; pelo contrário, busca a eficiência em todos os processos para entregar valor mais rapidamente ao cliente, sempre com foco na qualidade.

(e) Errado. A realização de testes é uma parte essencial de qualquer metodologia de desenvolvimento de software, incluindo Lean. Os testes garantem que o produto entregue seja de alta qualidade e atenda às necessidades do cliente, evitando desperdícios com retrabalho ou correções pós-entrega.

Gabarito: Letra C

---

7. **(FADESP / UEPA – 2020)** Um dos princípios do Manifesto Ágil é o de que os indivíduos e interações são mais importantes que processos e ferramentas. Um outro princípio é o de que:

a) o usuário é a principal fonte de informação de requisitos de software.
b) os contratos são mais importantes que a colaboração com os clientes.
c) o software funcionando é mais importante do que a documentação completa e detalhada.
d) seguir o plano inicial é mais importante que a adaptação a mudanças.

Comentários:

(a) Errado, isso realmente ocorre, mas não é um dos princípios do manifesto ágil; (b) Errado, colaboração com o cliente é mais valorizado que negociação de contratos; (c) Correto; (d) Errado, responder a mudanças é mais valorizado que seguir um plano.

Gabarito: Letra C

---

8. **(IESES / SCGás – 2019)** A filosofia por trás dos métodos ágeis é refletida no manifesto ágil, que foi acordado por muitos dos principais desenvolvedores desses métodos. Assinale a alternativa correta que contêm os itens deste manifesto.

a) "Estamos descobrindo melhores maneiras de desenvolver softwares, fazendo-o e ajudando outros a fazê-lo. Através desse trabalho, valorizamos mais: Indivíduos e interações do que processos e ferramentas; Software em funcionamento do que documentação abrangente; Colaboração do cliente do que negociação de contrato; Respostas a mudanças do que seguir um plano. Ou seja, embora itens à direita sejam importantes, valorizamos mais os que estão à esquerda".

b) "Estamos descobrindo melhores maneiras de desenvolver softwares, fazendo-o e ajudando outros a fazê-lo. Através desse trabalho, valorizamos mais: A concorrência e o desenvolvimento da competitividade entre as empresas; Software em funcionamento do que documentação abrangente; Colaboração do cliente do que negociação de contrato; Respostas a mudanças do que seguir um plano. Ou seja, embora itens à direita sejam importantes, valorizamos mais os que estão à esquerda".

c) "Estamos descobrindo melhores maneiras de desenvolver softwares, fazendo-o e ajudando outros a fazê-lo. Através desse trabalho, valorizamos mais: Indivíduos e interações do que processos e ferramentas; Software em funcionamento do que documentação abrangente; Colaboração da equipe de desenvolvedores do que negociação de contrato e clientes; Respostas a mudanças do que seguir um plano. Ou seja, embora itens à direita sejam importantes, valorizamos mais os que estão à esquerda".

d) "Estamos descobrindo melhores maneiras de vender softwares, fazendo-o e ajudando outros a fazê-lo. Através desse trabalho, valorizamos mais: Indivíduos e interações do que processos e ferramentas; Software para mobiles e, em funcionamento do que documentação abrangente; Colaboração do cliente do que negociação de contrato; Respostas a mudanças do que seguir um plano. Ou seja, embora itens à direita sejam importantes, valorizamos mais os que estão à esquerda".

Comentários:

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

Gabarito: Letra A

9. **(IESES / SCGás – 2019)** Identifique a opção correta para conceituar desenvolvimentos ágeis ou, que caracterizam métodos ágeis:

a) São métodos de desenvolvimento estáticos em que os incrementos são dinâmicos e, normalmente, as novas versões do sistema são criadas e disponibilizadas aos clientes a cada duas ou três semanas. Elas não envolvem os clientes no processo de desenvolvimento para obter feedback rápido sobre a evolução dos requisitos. Assim, minimiza-se a documentação, pois se utiliza mais a comunicação informal do que reuniões formais com documentos escritos.

b) São métodos de desenvolvimento incremental em que os incrementos são pequenos e, normalmente, as novas versões do sistema são criadas e disponibilizadas aos clientes a cada duas ou três semanas. Neles envolvemos clientes no processo de desenvolvimento para obter feedback rápido sobre a evolução dos requisitos. Assim, minimiza-se a documentação, pois se utiliza mais a comunicação informal do que reuniões formais com documentos escritos.

c) São métodos de desenvolvimento estáticos em que os incrementos são pequenos e, normalmente, as novas versões do sistema são criadas e disponibilizadas aos clientes a cada duas ou três semanas. Elas envolvem os clientes no processo de desenvolvimento para obter feedback rápido sobre a evolução dos requisitos. Assim, minimiza-se a documentação, pois se utiliza mais a comunicação informal do que reuniões formais com documentos escritos.

d) São métodos de desenvolvimento incremental em que os incrementos são intermediários e, normalmente, as novas versões do sistema são descritas e disponibilizadas aos clientes a cada duas ou três semanas. Elas envolvem os desenvolvedores do processo de concepção para obter feedback rápido sobre a evolução dos requisitos. Assim, minimiza-se a documentação, pois se utiliza mais a comunicação informal do que reuniões formais com documentos escritos.

Comentários:

(a) Errado, não são estáticos – são incrementais; (b) Correto; (c) Errado, não são estáticos – são incrementais; (d) Errado, incrementos são pequenos e, não, intermediários; as novas versões são criadas e, não, descritas; os clientes participam do processo de desenvolvimento e, não, os desenvolvedores participam no processo de concepção.

Gabarito: Letra B

10. **(IESES / SCGás – 2019)** Os processos de software podem ser categorizados como dirigidos a planos ou processos ágeis. Considerando esta afirmação, assinale a afirmativa correta:

a) Nos processos ágeis todas as atividades são planejadas antecipadamente, e a avaliação do processo considera a comparação com um planejamento inicial. Já nos processos dirigido a planos, o planejamento é gradativo. Esta característica facilita a alteração do processo de forma a refletir as necessidades de mudança dos clientes.

b) Nos processos dirigidos a planos todas as atividades são planejadas antecipadamente, e a avaliação do processo considera a comparação com um planejamento inicial. Já nos processos ágeis, o planejamento é gradativo. Esta característica facilita a alteração do processo de forma a refletir as necessidades de mudança dos clientes.

c) Nos processos ágeis todas as atividades são planejadas posteriormente, e a avaliação do processo considera a comparação com um planejamento inicial. Já nos processos dirigido a planos, o planejamento é gradativo. Esta característica facilita a alteração do processo de forma a refletir as necessidades de mudança dos clientes.

d) Nos processos dirigidos a planos todas as rotinas são empíricas e, a avaliação do processo considera a comparação com um planejamento final a ser definido. Já nos processos ágeis, o planejamento é gradativo. Esta característica facilita a alteração do processo de forma a refletir as necessidades de mudança dos clientes.

Comentários:

(a) Errado, isso ocorre em modelos dirigidos a planos, como o modelo em cascata; (b) Correto; (c) Errado, a questão inverteu os conceitos; (d) Errado, o empirismo é uma característica dos processos ágeis.

Gabarito: Letra B

---

11. **(INSTITUTO AOCP / EMPREL – 2019)** Em se tratando de desenvolvimento de software, o termo qualidade é bastante subjetivo. Entretanto, no desenvolvimento ágil, é claro o conceito de qualidade. Sabendo disso, assinale a alternativa que apresenta corretamente o conceito de qualidade no desenvolvimento ágil.

a) Envolve a documentação do processo e o estabelecimento de práticas para entregar ao cliente um produto de qualidade.

b) Cumpre os critérios sistêmicos estabelecidos em acordo com o cliente para que os requisitos também sejam cumpridos.

c) Tem como objetivo gerar manuais e código claro por meio de uma equipe especializada no processo.

d) Cumpre os requisitos para o cliente com uma documentação completa do produto desenvolvido.

e) Significa que a qualidade do código e as práticas são utilizadas para garantir um código de alta qualidade.

Comentários:

(a) Errado, software em funcionamento é mais valorizado do que documentação abrangente – e um indicativo melhor de qualidade; (b) Errado, colaboração com o cliente é mais valorizado que negociação e contratos; (c) Errado, metodologias ágeis prezam mais pelo software funcionando do que documentação abrangente; (d) Errado, metodologias ágeis prezam mais pelo software funcionando do que documentação abrangente; (e) Correto, a qualidade de um software é medida baseado na qualidade do código-fonte e das práticas de programação utilizadas.

Gabarito: Letra E

---

12. (IF-PE / IF-PE – 2019) O Manifesto Ágil é um documento que encoraja a utilização de métodos melhores no desenvolvimento de software. Nele foram escritos doze princípios que norteiam o desenvolvimento ágil de sistemas. Um dos princípios mais relevantes é:

a) "A prioridade é satisfazer a equipe de desenvolvimento por meio de uma entrega única de software de valor."

b) "A prioridade é satisfazer ao cliente por meio de uma entrega única de software de valor."

c) "A prioridade é satisfazer ao gerente do projeto por meio de entregas contínuas e frequentes de software de valor."

d) "A prioridade é satisfazer ao gerente de projetos por meio de uma entrega única de software de valor."

e) "A prioridade é satisfazer ao cliente por meio de entregas contínuas e frequentes de software de valor."

Comentários:

NÓS SEGUIMOS ESSES PRINCÍPIOS...
Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada e software com valor agregado.

Gabarito: Letra E

---

13. **(AJURI / Desenvolve - RR – 2018)** Desenvolvimento ágil de software (em inglês: Agile software development) ou Método ágil é uma expressão que define um conjunto de metodologias utilizadas no desenvolvimento de software. As metodologias que fazem parte do conceito de desenvolvimento ágil, tal como qualquer metodologia de software, providenciam uma estrutura conceitual para reger projetos de engenharia de software. Métodos ágeis enfatizam comunicações em tempo real, preferencialmente cara a cara, a documentos escritos. A maioria dos componentes de um grupo ágil deve estar agrupada em uma sala. Isso inclui todas as pessoas necessárias para terminar o software: no mínimo, os programadores e seus clientes(clientes são as pessoas que definem o produto, eles podem ser os gerentes, analistas de negócio, ou realmente os clientes). Considerando o contexto dos Valores da Metodologia Ágil, é correto afirmar que indivíduos e iterações:

a) mais do que processos e ferramentas; software funcional mais do que documentação abrangente; colaboração do cliente menor do que negociação de contratos; responder a mudanças menor do que seguir um plano.

b) mais do que processos e ferramentas; software funcional mais do que documentação abrangente; colaboração do cliente mais do que negociação de contratos; responder a mudanças mais do que seguir um plano.

c) mais do que processos e ferramentas; software funcional menos do que documentação abrangente; colaboração do cliente menor do que negociação de contratos; responder a mudanças na mesma medida que seguir um plano.

d) mais do que processos e ferramentas; software funcional mais do que documentação abrangente; colaboração do cliente na mesma medida que negociação de contratos; responder a mudanças na mesma medida que seguir um plano.

e) na mesma medida que processos e ferramentas; software funcional menos do que documentação abrangente; colaboração do cliente menor do que negociação de contratos; responder a mudanças menor do que seguir um plano.

Comentários:

A questão vacila ao dizer no enunciado "indivíduos e iterações" – o correto seria interações. Ignorando esse deslize, indivíduos e interações são mais valorizados do que processos e ferramentas; software funcional é mais valorizado do que documentação abrangente; colaboração do cliente é mais valorizado do que negociação de contratos; responder a mudanças é mais valorizado do que seguir um plano.

Gabarito: Letra B

14. **(INSTITUTO AOCP / PRODEB – 2018)** Assinale a alternativa que apresenta corretamente um dos princípios defendidos pelo Manifesto Ágil.

a) As melhores arquiteturas, requisitos e designs emergem de times com cronogramas bem definidos.

b) O método mais eficiente e eficaz de transmitir informações para um time de desenvolvimento é através de uma update meeting.

c) Deve-se construir projetos ao redor de estruturas hierárquicas verticais. Dando a eles o ambiente e suporte necessário.

d) Pessoas relacionadas a negócios devem trabalhar sem interferência constante ao time de desenvolvimento.

e) Em intervalos regulares, o time reflete como ficar mais efetivo, então se ajustam e otimizam seu comportamento de acordo.

Comentários:

NÓS SEGUIMOS ESSES PRINCÍPIOS...
Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.
Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho.
O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face.
As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis.
Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

Gabarito: Letra E

---

15. **(INSTITUTO AOCP / PRODEB – 2018)** Assinale a alternativa que apresenta uma característica presente em Equipes ágeis:

a) Equipe grande.
b) Equipe modestamente motivada.
c) Equipe que se auto-organiza.
d) Individualismo e talento.
e) Alto formalismo.

Comentários:

(a) Errado, equipes pequenas; (b) Errado, equipe altamente motivada; (c) Correto; (d) Errado, colaboração e talento; (e) Errado, alto empirismo.

Gabarito: Letra C

---

16. (INSTITUTO AOCP / PRODEB – 2018) Assinale a alternativa correta em relação ao manifesto ágil para desenvolvimento de software.

a) Uma documentação detalhada é o método mais eficiente e eficaz de transmitir informações para e por dentro de um time de desenvolvimento.

b) Processos ágeis se adéquam a mudanças para que o cliente possa tirar vantagens competitivas.

c) Não se deve aceitar mudanças de requisitos no fim do desenvolvimento.

d) Pessoas relacionadas a negócios e desenvolvedores devem manter contato em reuniões específicas.

e) Deve-se aceitar mudança de requisitos porém o time deve parar o desenvolvimento e voltar à etapa de validação de requisitos.

Comentários:

(a) Errado, o método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face; (b) Correto; (c) Errado, mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento; (d) Errado, pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto; (e) Errado, não é necessário parar o desenvolvimento.

Gabarito: Letra B

---

17. (INSTITUTO AOCP / PRODEB – 2018) Com a realização do Manifesto Ágil em 2001 por um conjunto de especialistas em processos de desenvolvimento de software, ficaram definidos alguns parâmetros principais que passaram a ser um denominador comum de Metodologias Ágeis. São características atribuídas aos métodos ágeis, EXCETO:

a) processos e ferramentas ao contrário de pessoas e interações.
b) software executável, ao contrário de documentação extensa e confusa.
c) colaboração do cliente, ao contrário de constantes negociações de contratos.
d) indivíduos e interações mais que processos e ferramentas.
e) respostas rápidas para as mudanças, ao contrário de seguir planos previamente definidos.

Comentários:

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

Gabarito: Letra A

---

18. (FCM / IFN-MG – 2018) O Manifesto Ágil para o Desenvolvimento de Software, proposto por Beck, K. et al. (2001), propõe 12 princípios. NÃO correspondem a um desses princípios criados por esses autores:

a) as melhores arquiteturas, requisitos e projetos emergem de equipes auto-organizadas.

b) a simplicidade é a arte de maximizar a quantidade de trabalho que não precisa ser feito.

c) o projeto para ser ágil precisa ter um controle bem definido sobre as pessoas e as tarefas que elas executam.

d) a prioridade é satisfazer o cliente através de entrega antecipada e contínua de um software que tenha valor para o mesmo.

e) a entrega do software deve ser feita com uma frequência predeterminada de tempo, preferencialmente em uma escala de tempo mais curta.

Comentários:

(a) Correto, as melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis; (b) Correto, simplicidade – a arte de maximizar a quantidade de trabalho não realizado – é essencial; (c) Errado, esse não é um dos doze princípios ágeis; (d) Errado, nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado; (e) Errado, entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo.

Gabarito: Letra C

---

19. (CS-UFG / UFG – 2019) O desenvolvimento de software baseado em abordagem ágil estimula:

a) a produção de planos detalhados.
b) a realização de atividades de desenvolvimento em cada iteração.
c) a valorização da equipe de operação em detrimento daquela de desenvolvimento.
d) a aplicação de métodos formais de desenvolvimento de software.

Comentários:

(a) Errado, valoriza-se mais responder a mudanças do que seguir um plano detalhado; (b) Correto; (c) Errado, isso não existe; (d) Errado, métodos formais são – como o próprio nome diz – formais. A abordagem ágil prega o empirismo e a resposta à mudanças.

Gabarito: Letra B

---

20. **(INSTITUTO AOCP / ITEP – RN – 2018)** Qual das alternativas a seguir apresenta somente métodos ágeis de desenvolvimento de software?

a) XP e Scrum.
b) Cascata e XP.
c) Incremental e XP.
d) Evolucionário e Scrum.
e) Incremental e Evolucionário.

Comentários:

(a) Correto; (b) Errado, Cascata é tradicional; (c) Errado, Incremental é tradicional; (d) Errado, Evolucionário é tradicional; (e) Errado, ambos são tradicionais.

Gabarito: Letra A

---

21. **(UECE-CEV / Prefeitura de Sobral - CE – 2018)** Escreva V ou F conforme seja verdadeiro ou falso o que se afirma nos itens abaixo com respeito ao processo de desenvolvimento ágil de software.

( ) Efetuar testes constantemente permite detectar defeitos mais cedo e da forma menos custosa possível.

( ) O uso de uma ferramenta robusta de modelagem e uma completa documentação são imprescindíveis para o desenvolvimento ágil.

( ) É importante produzir em poucas semanas uma versão inicial do software a fim de obter rapidamente uma primeira conquista e um feedback adiantado.

( ) Novas versões do software devem ser lançadas em intervalos cada vez mais frequentes, seja semanalmente, diariamente ou mesmo de hora em hora.

a) V, F, F, V.
b) F, V, F V.

c) V, F, V, F.
d) F, V, V, F.

Comentários:

(V) Efetuar testes constantemente permite detectar defeitos mais cedo e da forma menos custosa possível; (F) Valorizam-se mais indivíduos e interações do que processos e ferramentas; (V) A ideia é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado; (F) A ideia é entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo.

Gabarito: Letra C

---

22. (CETRO / ANVISA – 2013) Com relação aos conceitos do processo ágil, um dos conceitos-chave do Manifesto Ágil é :

I. produzir documentação em vez de software executável.
II. a colaboração do cliente em vez da negociação de contratos.
III. obter respostas rápidas a mudanças em vez de seguir planos.

É correto o que está contido em:

a) I, apenas.
b) II, apenas.
c) III, apenas.
d) II e III, apenas.
e) I, II e III.

Comentários:

(I) Errado, software em funcionamento é mais valorizado do que documentação abrangente; (II) Correto; (III) Correto.

Gabarito: Letra D

---

23. (UNIRIO / UNIRIO – 2014) Dentre os princípios do manifesto ágil para desenvolvimento de software, NÃO se inclui (em):

a) a satisfação do cliente deve ser priorizada através da entrega contínua.
b) conversas face a face são preferíveis para e entre uma equipe de desenvolvimento.
c) simplicidade é essencial.
d) mudança nos requisitos devem ser evitadas.
e) entregas de software funcionando devem ser realizadas frequentemente.

Comentários:

(a) Correto, nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado; (b) Correto, o método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face; (c) Correto, simplicidade – a arte de maximizar a quantidade de trabalho não realizado – é essencial; (d) Errado, mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento; (e) Correto, entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo.

Gabarito: Letra D

---

24. (FCM / IF-RS – 2016) As metodologias ágeis tornaram-se populares em 2001 quando um grupo de especialistas em processos de desenvolvimento de software decidiu se reunir nos Estados Unidos. O objetivo foi discutir maneiras de melhorar o desempenho de seus projetos. Embora tivessem preferências e métodos distintos entre si, concordaram que um pequeno conjunto de princípios sempre parecia ter sido respeitado quando os projetos davam certo. Foi então criada a Aliança Ágil e o estabelecimento do Manifesto Ágil, contendo os conceitos e os princípios comuns compartilhados por todos esses métodos.

NÃO é considerado um princípio por trás do Manifesto Ágil:

a) Responder a mudanças mais que seguir um plano.
b) Colaboração com o cliente mais que negociação de contratos.
c) Processos e ferramentas mais que indivíduos e interação entre eles.
d) Software em funcionamento mais que documentação abrangente.
e) Indivíduos e interação entre eles mais que processos e ferramentas.

Comentários:

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

Gabarito: Letra C

---

25. (FUNCAB / MJ-SP – 2015) O manifesto ágil considera que a medida primária de progresso é:

a) tempo utilizado.
b) quantidadede testes.
c) quantidadede documentação.
d) custo realizado.

e) software funcionando.

Comentários:

De acordo com o 7º princípio ágil: software funcionando é a medida primária de progresso.

Gabarito: Letra E

---

26. (UECE-CEV / FUNCEME – 2018) O Manifesto para o desenvolvimento ágil de software resume os itens mais valorizados pelos praticantes desta abordagem. Considerando os itens listados a seguir, assinale a opção que NÃO representa um valor ágil segundo o Manifesto.

a) indivíduos e interações mais que processos e ferramentas
b) seguir um plano mais que responder a mudanças
c) software em funcionamento mais que documentação abrangente
d) colaboração com o cliente mais que negociação de contratos

Comentários:

(1) Indivíduos e interações acima de processos e ferramentas; (2) Software em funcionamento acima de documentação abrangente; (3) Colaboração com o cliente acima de negociação de contratos; (4) **Responder a mudanças acima de seguir fielmente um plano**.

Gabarito: Letra B

---

27. (ESAF / MF – 2013) O desenvolvimento ágil de software fundamenta-se no Manifesto Ágil. Segundo ele deve-se valorizar:

a) mudança de respostas em vez do seguimento de um plano.
b) indivíduos e interações em vez de processos e ferramentas.
c) documentação extensiva operacional em vez de software funcional.
d) indivíduos e intenções junto a processos e ferramentas.
e) seguimento de um plano em vez de resposta a mudança.

Comentários:

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

Gabarito: Letra B

---

28. (IF-PE / IF-PE – 2016) Sobre o documento conhecido como "manifesto ágil", é CORRETO dizer que:

a) prega uma extensa lista de documentos, processos, atores, métodos e diagramas visando fornecer alta agilidade.

b) lista e cataloga a maioria dos métodos vigentes à época de sua criação, classificando cada um como "ágil" ou "burocrático".

c) foi criado como base para descrever as principais ideias e práticas que eram comuns a muitos dos métodos considerados ágeis e que já existiam na época.

d) foi criado com base na ideia de que se tudo for muito bem controlado e documentado, os processos serão naturalmente ágeis.

e) a partir dele, foram definidos o XP, o scrum, o crystal, o CMM e o RUP, cada um com suas características particulares.

Comentários:

(a) Errado, prega software em funcionamento mais do que documentações abrangentes ; (b) Errado, esse item não faz qualquer sentido; (c) Correto; (d) Errado, prega software em funcionamento mais do que documentações abrangentes; (e) Errado, todos esses métodos já existiam antes do manifesto ágil.

Gabarito: Letra C

---

29. (CS-UFG / UFG – 2018) Ao se empregar métodos ágeis em desenvolvimento de software, as atividades:

a) são planejadas com antecedência, e seu progresso é medido em relação ao plano estabelecido.

b) são realizadas com base na abordagem iterativa/incremental de desenvolvimento.

c) são planejadas com base no modelo cascata, com fases separadas e distintas de especificação e desenvolvimento.

d) são realizadas em fases sequenciais, sendo que cada fase precisa estar completa antes que se passe para a próxima.

Comentários:

(a) Errado, software funcionando é a medida primária de progresso – valoriza-se mais a resposta a mudanças do que seguir um plano; (b) Correto; (c) Errado, modelo em cascata utiliza uma abordagem tradicional e, não, ágil; (d) Errado, são realizadas de forma iterativa e incremental.

Gabarito: Letra B

---

30. (CESGRANRIO / Banco da Amazônia – 2018) O Manifesto Ágil se tornou um marco da Engenharia de Software, chamando a atenção de que vários processos propostos de forma independente tinham valores em comum. Além disso, foram definidos 12 princípios. Entre eles, figura o seguinte princípio:

a) cada pessoa em um projeto deve ter sua função predeterminada para acelerar o desenvolvimento em conjunto.
b) a contínua atenção à simplicidade do trabalho feito aumenta a agilidade.
c) software funcionando é a medida primária de progresso.
d) os indivíduos, clientes e desenvolvedores, são mais importantes que processos e ferramentas.
e) o software funcional emerge de times auto-organizáveis.

Comentários:

Nenhum desses faz parte dos doze princípios, exceto: software funcionando é a medida primária de progresso.

Gabarito: Letra C

---

31. (IADES / ARCON-PA – 2018) Embora esses métodos ágeis sejam todos baseados na noção de desenvolvimento e entrega incremental, eles propõem diferentes processos para alcançar tal objetivo. No entanto, compartilham um conjunto de princípios, com base no manifesto ágil, e por isso têm muito em comum.

SOMMERVILLE, I. Engenharia de software. 9. ed. São Paulo: Person Education, 2011.

Os cinco princípios citados no texto são:

a) envolvimento do cliente; entregas agendadas; pessoas e processos são igualmente importantes; aceitar mudanças; e manter a simplicidade.

b) envolvimento do cliente; entrega incremental; pessoas, não processos; aceitar as mudanças; e manter a simplicidade.

c) envolvimento do cliente apenas no início; entrega incremental; prazos rígidos; evitar mudanças; e manter a equipe.

d) programadores em primeiro lugar; ausência de prazos; cliente como última prioridade; aceitar as mudanças; e investir em controle de versão.

e) programadores em primeiro lugar; entrega por protótipos; processos, não pessoas; aceitar as mudanças; e manter o cronograma.

Comentários:

(a) Errado. Envolvimento do cliente; ~~entregas agendadas~~; ~~pessoas e processos são igualmente importantes~~; aceitar mudanças; e manter a simplicidade;

(b) Correto. Envolvimento do cliente; entrega incremental; pessoas, não processos; aceitar as mudanças; e manter a simplicidade.

(c) Errado. Envolvimento do cliente ~~apenas no início~~; entrega incremental; prazos rígidos; ~~evitar mudanças~~; e manter a equipe.

(d) Errado. ~~Programadores em primeiro lugar~~; ~~ausência de prazos~~; ~~cliente como última prioridade~~; aceitar as mudanças; e ~~investir em controle de versão~~.

(e) Errado. ~~Programadores em primeiro lugar~~; ~~entrega por protótipos~~; ~~processos, não pessoas~~; aceitar as mudanças; e manter o cronograma.

Gabarito: Letra B

---

32. **(FAURGS / TJ-RS – 2018)** Considere as seguintes afirmações sobre princípios dos métodos ágeis.

I - Os clientes devem estar totalmente envolvidos no processo de desenvolvimento. Seu papel é fornecer e priorizar novos requisitos do sistema e avaliar suas iterações.

II - Embora as habilidades da equipe devam ser reconhecidas e exploradas, seus membros não devem desenvolver maneiras próprias de trabalhar, podendo o processo ser prescritivo.

III- Deve-se ter em mente que os requisitos do sistema irão mudar, por isso, o sistema deve ser projetado de maneira a acomodar essas mudanças.

Quais estão corretas?

a) Apenas I.
b) Apenas I e II.
c) Apenas I e III.
d) Apenas II e III.
e) I, II e III.

Comentários:

(I) Correto, pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto; (II) Errado, as melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis da maneira que se sentirem mais adequados; (III) Correto, mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento.

Gabarito: Letra C

LISTA DE QUESTÕES – CESPE

1. **(CESPE / BANRISUL – 2022)** O modelo ágil não pode ser aplicado a qualquer processo de software, pois, para tanto, é necessário que o processo seja projetado de modo que suas características sejam modeladas como componentes e, em seguida, construídas dentro do contexto da arquitetura do sistema.

2. **(CESPE / Petrobrás - 2022)** Entre as principais características dos métodos ágeis, destacam-se a maximização da documentação formal e o envolvimento dos clientes.

3. **(CESPE / TCE-ES – 2012)** Em virtude de as metodologias ágeis gerarem excessiva documentação, a gestão do conhecimento depende diretamente dos programadores envolvidos no projeto.

4. **(CESPE / EBC – 2011)** O que os métodos ágeis buscam é como evitar as mudanças desde o início do projeto e não a melhor maneira de tratar essas mudanças.

5. **(CESPE / BASA – 2010)** Desenvolvimento ágil de software (Agile Software Development) ou método ágil é aplicado, principalmente, a grandes corporações, uma vez que permite produzir grandes sistemas de forma ágil.

6. **(CESPE / TCU – 2010)** A agilidade não pode ser aplicada a todo e qualquer processo de software.

7. **(CESPE / UNIPAMPA – 2009)** XP, Scrum e Cristal são exemplos de modelos ágeis de desenvolvimento de sistemas.

8. **(CESPE / EBC – 2011)** Considerando o conceito de metodologia ágil em apreço, é correto afirmar que as seguintes metodologias são ágeis: XP (Extreme Programming), Scrum, Crystal, FDD (Feature Driven Development), DSDM (Dynamic Systems Development Method) e Open Source Software Development.

9. **(CESPE / CNJ – 2013** O desenvolvimento ágil de sistemas consiste em uma linguagem de modelagem que permite aos desenvolvedores visualizarem os produtos de seu trabalho em gráficos padronizados.

10. **(CESPE / EBC – 2011)** É conveniente que o contrato, entre cliente e fornecedor, para o desenvolvimento de um sistema computacional, contenha a lista de requisitos para o software. Contudo, os métodos ágeis de desenvolvimento preconizam que o referido contrato estabeleça o preço, a ser pago pelo cliente, com base no tempo necessário para o desenvolvimento do sistema e não com base no conjunto de requisitos.

11. **(CESPE / MPOG – 2015)** Metodologias de desenvolvimento ágil enfocam atividades de projeto e implementação, desconsiderando as atividades de elicitação de requisitos e a produção de documentação.

12. **(CESPE / TRE-PI – 2008)** No que se refere a métodos ágeis de desenvolvimento de sistemas, assinale a opção correta.

a) A aplicação de método ágil para desenvolvimento de grandes sistemas pode enfrentar dificuldades que o tornem inviável.

b) O documento de requisitos, apesar de abordar um conjunto pequeno de funcionalidades, deve especificar toda a necessidade do usuário.

c) O sistema é construído em pequenos blocos, que irão compor uma versão a ser entregue aos usuários.

d) A documentação de projeto deve ser feita pelo próprio desenvolvedor, seguindo padrões simplificados.

e) Para atingir os objetivos de agilidade exigidos, os desenvolvedores devem seguir processos simplificados para a construção do software.

13. **(CESPE / TCE-PR – 2016)** Os métodos ágeis para o desenvolvimento de software representam uma evolução da engenharia de software tradicional, uma vez que são aplicáveis a todos os tipos de projetos, produtos, pessoas e situações.

14. **(CESPE / TCE-PR – 2016)** Um dos princípios de agilidade da Agile Alliance dispõe que a entrega completa de um software garante a satisfação do cliente.

15. **(CESPE / Ministério da Economia – 2020)** Os modelos ágeis de desenvolvimento de software dão grande ênfase às definições de atividades e aos processos e pouca ênfase à pragmática e ao fator humano.

16. **(CESPE / MEC – 2015)** Acatar as mudanças de requisitos, ainda que o desenvolvimento já esteja avançado, é um dos princípios do Manifesto Ágil.

17. **(CESPE / TRT17 – 2013)** Em um desenvolvimento ágil que segue o manifesto ágil, não se deve aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis não se adequam a mudanças não planejadas.

18. **(CESPE / EBSERH – 2018)** Nas metodologias de desenvolvimento ágeis, mudanças em requisitos são bem recebidas, mesmo em fases mais avançadas do desenvolvimento.

19. **(CESPE / SEDF – 2017)** Lean manufacturing e kaizen são exemplos de ferramentas de gestão da qualidade aplicadas para o aperfeiçoamento de organizações e preveem a realização de diagnósticos e implementação de melhorias.

GABARITO

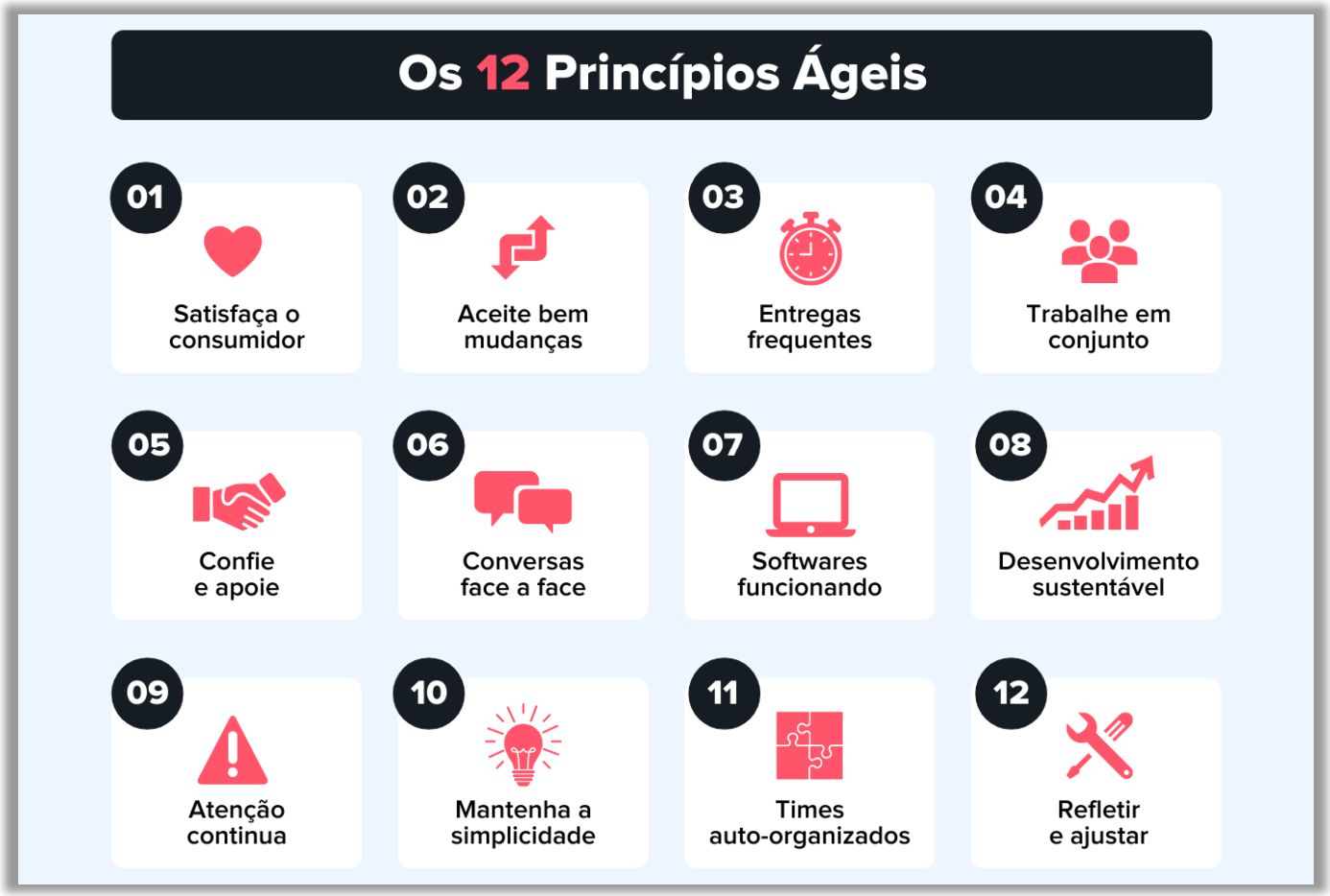
1. ERRADO
2. ERRADO
3. ERRADO
4. ERRADO
5. ERRADO
6. ERRADO
7. CORRETO
8. CORRETO
9. ERRADO
10. CORRETO
11. ERRADO
12. LETRA A
13. ERRADO
14. ERRADO
15. ERRADO
16. CORRETO
17. ERRADO
18. CORRETO
19. CORRETO

LISTA DE QUESTÕES – FCC

1. **(FCC / SEFAZ-AP – 2022)** Dentre os doze Princípios do Manifesto Ágil, incluem-se:

a) funcionalidade, satisfação do cliente e trabalho em conjunto.
b) respeito ao cliente, economia de recursos e paralelismo.
c) resiliência, motivação e trabalho em pares.
d) simplicidade, motivação e paralelismo.
e) especificidade, longevidade do *software* e prazos curtos.

Comentários:



Trata-se de funcionalidade (software funcionando), satisfação do cliente (satisfaça o consumidor) e trabalho em conjunto.

Gabarito: Letra A

2. **(FCC / SEFAZ-AP – 2022)** Dentre os doze Princípios do Manifesto Ágil, incluem-se:

a) respeito ao cliente, economia de recursos e paralelismo.
b) resiliência, motivação e trabalho em pares.
c) simplicidade, motivação e paralelismo.
d) especificidade, longevidade do software e prazos curtos.
e) funcionalidade, satisfação do cliente e trabalho em conjunto.

Comentários:

(a) Errado. "*Respeito ao cliente*" e "*economia de recursos*" não são explicitamente citados no Manifesto Ágil, e "*paralelismo*" não é um termo geralmente associado com os princípios ágeis;

b) Errado. "*Resiliência*" não é um dos princípios ágeis. "*Motivação*" é uma interpretação livre do princípio que fala em pessoas motivadas e "*trabalho em pares*" pode ser relacionado com métodos ágeis como a Programação em Pares, mas não é um princípio por si só;

c) Errado. "*Simplicidade*" é explicitamente mencionado como o princípio de maximizar a quantidade de trabalho não realizado. "*Motivação*" pode ser relacionado ao princípio de construir projetos em torno de indivíduos motivados. "*Paralelismo*" novamente não é um termo usado;

d) Errado. "*Especificidade*" e "*longevidade do software*" não são termos usados nos princípios ágeis. "Prazos curtos" se assemelha ao princípio de entregas frequentes, mas não é o termo utilizado;

e) Correto. "*Funcionalidade*" não é um princípio expresso, mas "*satisfação do cliente*" é o primeiro princípio do Manifesto Ágil, e "*trabalho em conjunto*" reflete o princípio de colaboração constante com o cliente.

Gabarito: Letra E

GABARITO

1. LETRA A
2. LETRA E

LISTA DE QUESTÕES – FGV

1. **(FGV / TJ-RN - 2023)** A Equipe de Tecnologia da Informação (ETI) de um tribunal está envolvida no projeto TERC cujo ciclo de vida segue uma abordagem adaptativa (ágil). Considerando o ciclo de vida empregado no TERC, a ETI:

   a) especifica os requisitos do produto final antes do início do desenvolvimento;
   b) solicita o envolvimento das partes interessadas chave em marcos específicos;
   c) controla os riscos à medida em que surgem requisitos e restrições;
   d) ajusta o ciclo de vida do projeto ao ciclo de vida do produto;
   e) estabelece as fases do projeto com base em um ciclo de vida de desenvolvimento preditivo.

2. **(FGV / IMBEL – 2021)** Com referência aos valores do The Agile Manifesto, analise as afirmativas a seguir.

   I. Processos e ferramentas mais que indivíduos e interação entre eles.
   II. Software em funcionamento mais que documentação abrangente.
   III. Colaboração do cliente mais que negociação de contratos.
   IV. Seguir um plano mais que responder a mudanças.

   Está correto o que se afirma em:

   a) I e II, somente
   b) II e III, somente.
   c) III e IV, somente.
   d) I e IV, somente.
   e) II e IV, somente.

3. **(FGV / MPE-MS – 2013)** Considerando a caracterização de agilidade e processo de desenvolvimento ágil, segundo Pressman, analise as afirmativas a seguir.

   I. Um processo ágil de software deve ser incrementalmente adaptável.
   II. Um processo ágil de software permite que as pessoas e a equipe se moldem a ele com facilidade.
   III. Os conceitos ágeis são efetivos, pois diminuem a imprevisibilidade sistêmica ao enfatizar entregas em prazos curtos.

   a) se somente a afirmativa I estiver correta.
   b) se somente a afirmativa II estiver correta.
   c) se somente a afirmativa III estiver correta.
   d) se somente as afirmativas I e II estiverem corretas.
   e) se todas as afirmativas estiverem corretas.

4. **(FGV / PGE-RO – 2015)** Durante 5 anos gerenciando o desenvolvimento de sistemas de informação, Claudia teve que lidar com diversas insatisfações de seus usuários pois os sistemas não atendiam as suas necessidades. Claudia decidiu, então, implantar métodos ágeis de desenvolvimento e definiu os seguintes princípios:

I. Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento.

II. O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através da documentação.

III. Simplicidade é essencial.

Dentre os princípios definidos por Claudia, o que infringe os princípios do manifesto para Desenvolvimento Ágil de Software é o que se afirma em:

a) somente I;
b) somente II;
c) somente III;
d) somente I e III;
e) I, II e III.

5. **(FGV / TJ-RO – 2015)** O manifesto ágil tem por princípio que:

a) mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento;

b) a contínua atenção à excelência técnica reduz a agilidade;

c) a redução do backlog é a medida primária de progresso;

d) as melhores arquiteturas, requisitos e designs emergem de equipes que possuem um bom líder;

e) pessoas de negócio e desenvolvedores devem trabalhar em ambientes separados para reduzir as interferências no processo de desenvolvimento.

6. **(FGV / TJ-GO – 2014)** Escreva O Manifesto Ágil lista valores seguidos por desenvolvedores com a finalidade de melhorar a maneira pela qual o software é desenvolvido. A alternativa que se encontra no manifesto é:

a) seguir um plano mais que responder a mudanças;
b) indivíduos e interações mais que processos e ferramentas;
c) documentação abrangente mais que software em funcionamento;
d) negociação de contratos mais que colaboração com o cliente;

e) negociação de contratos mais que indivíduos e interações.

7. **(FGV / Câmara Municipal de Caruaru-PE – 2015)** O desenvolvimento ágil de software é guiado por metodologias que compartilham um conjunto comum de valores e de princípios, conforme definido pelo Manifesto Ágil. Assinale a opção que indica um princípio do desenvolvimento ágil.

a) As mudanças nos requisitos devem ocorrer dentro do quadro de tempo estabelecido para a iteração.

b) O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é por meio de conversa face a face.

c) Os intervalos regulares devem ser evitados para tornar a equipe mais eficaz e maximizar a quantidade de trabalho realizado.

d) As pessoas de negócio e desenvolvedores devem interagir somente no início de cada iteração.

e) A entrega contínua e adiantada de software, mesmo que o conjunto de funcionalidades desenvolvidas não agregue valor, deve ser feita para satisfazer o cliente.

8. **(FGV / PROCEMPA – 2014)** O Manifesto Ágil é uma declaração de princípios que fundamentam o desenvolvimento ágil de software. A respeito desses princípios, assinale a afirmativa correta:

a) As melhores arquiteturas, requisitos e designs emergem de equipes lideradas pelo profissional mais sênior.

b) Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

c) Pessoas de negócio e desenvolvedores devem trabalhar separadamente por todo o projeto.

d) Entregar software quando há poucas semanas de desenvolvimento deve ser evitado para não afetar a satisfação do cliente.

e) Mudanças nos requisitos são bem-vindas, desde que não impactem o desenvolvimento.

9. **(FGV / DPE-RO – 2015)** O Manifesto Ágil é uma declaração que reúne os princípios e práticas que fundamentam o desenvolvimento ágil de software. É um dos princípios desse manifesto:

a) defeitos no software são a medida primária de progresso;

b) pessoas de negócio e desenvolvedores devem trabalhar isoladamente e se reunir somente ao final de cada iteração para validação do software;

c) atenção contínua à excelência técnica deve ser evitada para não afetar a agilidade uma vez que simplicidade é essencial;

d) os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente evitando interrupções e intervalos regulares;

e) as melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis.

10. **(FGV / BANESTES – 2018)** Um dos valores relacionados ao ambiente ágil de desenvolvimento é:

a) documentação abrangente mais que software funcional;
b) negociação de contratos mais que colaboração do cliente;
c) processos e ferramentas mais que indivíduos e iterações;
d) rapidez na construção mais que excelência técnica;
e) responder a mudanças mais que seguir um plano.

11. **(FGV / BANESTES – 2018)** Com relação aos valores relacionados ao desenvolvimento ágil de software, NÃO se pode incluir:

a) colaboração do cliente mais que negociação de contratos;
b) indivíduos e iterações mais que processos e ferramentas;
c) rapidez na construção mais que excelência técnica;
d) responder a mudanças mais que seguir um plano;
e) software funcional mais que documentação abrangente.

12. **(FGV / AL-RO – 2018)** Para o desenvolvimento do Sistema de Informações ao Cidadão (SIC), foi decidida a utilização de uma metodologia ágil. Segundo o Manifesto Ágil, esta decisão indica que foi dado maior valor:

a) aos processos e ferramentas.
b) à resposta a modificações.
c) à documentação abrangente.
d) à negociação do contrato.
e) ao cumprimento do plano.

13. **(FGV / TJDFT – 2022)** Uma equipe de analista de sistemas está desenvolvendo o software ProgramaTJ aplicando a metodologia Lean. A equipe decidiu implementar apenas as funcionalidades formalmente requisitadas pelo cliente, evitando adicionar qualquer funcionalidade extra à ProgramaTJ por conta própria. Essa decisão da equipe remete, de forma direta, ao princípio da metodologia Lean para o desenvolvimento de software de:

a) otimização do todo;
b) adiar comprometimento;

c) eliminação de desperdícios;
d) respeitar as pessoas;
e) criação de conhecimento.

GABARITO

1. LETRA C
2. LETRA B
3. LETRA A
4. LETRA B
5. LETRA A
6. LETRA B
7. LETRA B
8. LETRA B
9. LETRA E
10. LETRA E
11. LETRA C
12. LETRA B
13. LETRA C

LISTA DE QUESTÕES – DIVERSAS BANCAS

1. **(ADMTEC / Prefeitura de Palmeira dos Índios-AL – 2024)** Analise as informações a seguir:

   I. Entre as metodologias para desenvolvimento de software mais conhecidas e utilizadas atualmente, o Modelo Waterfall (Cascata) ainda se destaca por trabalhar em 5 fases: Requerimento, Projeto, Implementação e Verificação e Manutenção.

   II. Entre as metodologias para desenvolvimento de software mais conhecidas e utilizadas atualmente, a metodologia Lean ganha a atenção dos desenvolvedores por se basear em 5 princípios: Reduzir o desperdício; Postergar as decisões; Agilizar as entregas; Empoderar as equipes e Otimizar o todo.

   Marque a alternativa CORRETA:

   a) As duas afirmativas são verdadeiras.
   b) A afirmativa I é verdadeira, e a II é falsa.
   c) A afirmativa II é verdadeira, e a I é falsa.
   d) As duas afirmativas são falsas.

2. **(OBJETIVO / Prefeitura de Piratininga-SP – 2023)** O método Lean é um dos principais métodos ágeis utilizados na gestão de projetos. Sobre essa metodologia, assinalar a alternativa CORRETA:

   a) É utilizada na gestão de projetos que não tenham prazo de entrega, utilizando intervalos de tempo para desenvolver cada etapa.

   b) É composta por checklists, oferecendo uma visão global do projeto. É específica para alguns tipos de negócio, pois busca a revolução dos processos.

   c) É uma boa alternativa para criar objetivos realistas e possíveis para a situação de cada empresa, baseando-se em cinco primórdios, os quais são indicados pelas letras do seu nome.

   d) É utilizada para projetos mais objetivos ou reduzidos, e identifica eficientemente os desperdícios.

3. **(IADES / CRF-TO – 2023)** Assinale a alternativa que apresenta um princípio da metodologia Lean de desenvolvimento de software.

   a) Adiar decisões o máximo possível.
   b) Abraçar o desperdício.
   c) Entregar com atraso.

d) Compartimentalizar o conhecimento.
e) Fortalecer a hierarquia.

4. **(FEPESE / EPAGRI – 2023)** Qual metodologia ágil tem como foco a excelência na qualidade e aumento da velocidade dos processos e eliminar o desbarato?

a) OpenUP
b) Pragmatic Programming
c) Test Driven Development
d) Lean Software Development
e) Microsoft Solutions Framework

5. **(QUADRIX / PRODAM-AM – 2022)** A partir dos princípios abordados pela metodologia ágil Lean, assinale a alternativa que apresenta uma forma de desperdício na qual um número excessivo de mudanças de contexto reduz a produtividade.

a) esperas por requisitos
b) troca de tarefas
c) construção da integridade
d) defeitos
e) antecipação das funcionalidades

6. **(QUADRIX / PRODAM-AM – 2022)** Com relação à metodologia ágil para o desenvolvimento de software Lean, assinale a alternativa correta.

a) O excesso de processos não é considerado um desperdício, porque eles não demandam recursos.

b) Os processos complexos não aumentam a quantidade de documentos, por isso não caracterizam desperdício.

c) O pensamento Lean foca em oferecer o que o cliente quer, onde e quando ele quiser, sem haver qualquer desperdício.

d) Para a metodologia Lean, o desenvolvimento rápido do software só apresenta desvantagens, pois alguns processos são atropelados.

e) Não há necessidade da realização de testes, uma vez que os softwares desenvolvidos por meio dessa metodologia são eficazes.

7. **(FADESP / UEPA – 2020)** Um dos princípios do Manifesto Ágil é o de que os indivíduos e interações são mais importantes que processos e ferramentas. Um outro princípio é o de que:

a) o usuário é a principal fonte de informação de requisitos de software.

b) os contratos são mais importantes que a colaboração com os clientes.
c) o software funcionando é mais importante do que a documentação completa e detalhada.
d) seguir o plano inicial é mais importante que a adaptação a mudanças.

8. **(IESES / SCGás – 2019)** A filosofia por trás dos métodos ágeis é refletida no manifesto ágil, que foi acordado por muitos dos principais desenvolvedores desses métodos. Assinale a alternativa correta que contêm os itens deste manifesto.

a) "Estamos descobrindo melhores maneiras de desenvolver softwares, fazendo-o e ajudando outros a fazê-lo. Através desse trabalho, valorizamos mais: Indivíduos e interações do que processos e ferramentas; Software em funcionamento do que documentação abrangente; Colaboração do cliente do que negociação de contrato; Respostas a mudanças do que seguir um plano. Ou seja, embora itens à direita sejam importantes, valorizamos mais os que estão à esquerda".

b) "Estamos descobrindo melhores maneiras de desenvolver softwares, fazendo-o e ajudando outros a fazê-lo. Através desse trabalho, valorizamos mais: A concorrência e o desenvolvimento da competitividade entre as empresas; Software em funcionamento do que documentação abrangente; Colaboração do cliente do que negociação de contrato; Respostas a mudanças do que seguir um plano. Ou seja, embora itens à direita sejam importantes, valorizamos mais os que estão à esquerda".

c) "Estamos descobrindo melhores maneiras de desenvolver softwares, fazendo-o e ajudando outros a fazê-lo. Através desse trabalho, valorizamos mais: Indivíduos e interações do que processos e ferramentas; Software em funcionamento do que documentação abrangente; Colaboração da equipe de desenvolvedores do que negociação de contrato e clientes; Respostas a mudanças do que seguir um plano. Ou seja, embora itens à direita sejam importantes, valorizamos mais os que estão à esquerda".

d) "Estamos descobrindo melhores maneiras de vender softwares, fazendo-o e ajudando outros a fazê-lo. Através desse trabalho, valorizamos mais: Indivíduos e interações do que processos e ferramentas; Software para mobiles e, em funcionamento do que documentação abrangente; Colaboração do cliente do que negociação de contrato; Respostas a mudanças do que seguir um plano. Ou seja, embora itens à direita sejam importantes, valorizamos mais os que estão à esquerda".

9. **(IESES / SCGás – 2019)** Identifique a opção correta para conceituar desenvolvimentos ágeis ou, que caracterizam métodos ágeis:

a) São métodos de desenvolvimento estáticos em que os incrementos são dinâmicos e, normalmente, as novas versões do sistema são criadas e disponibilizadas aos clientes a cada duas ou três semanas. Elas não envolvem os clientes no processo de desenvolvimento para obter feedback rápido sobre a evolução dos requisitos. Assim, minimiza-se a documentação, pois se utiliza mais a comunicação informal do que reuniões formais com documentos escritos.

b) São métodos de desenvolvimento incremental em que os incrementos são pequenos e, normalmente, as novas versões do sistema são criadas e disponibilizadas aos clientes a cada duas ou três semanas. Neles envolvemos clientes no processo de desenvolvimento para obter feedback rápido sobre a evolução dos requisitos. Assim, minimiza-se a documentação, pois se utiliza mais a comunicação informal do que reuniões formais com documentos escritos.

c) São métodos de desenvolvimento estáticos em que os incrementos são pequenos e, normalmente, as novas versões do sistema são criadas e disponibilizadas aos clientes a cada duas ou três semanas. Elas envolvem os clientes no processo de desenvolvimento para obter feedback rápido sobre a evolução dos requisitos. Assim, minimiza-se a documentação, pois se utiliza mais a comunicação informal do que reuniões formais com documentos escritos.

d) São métodos de desenvolvimento incremental em que os incrementos são intermediários e, normalmente, as novas versões do sistema são descritas e disponibilizadas aos clientes a cada duas ou três semanas. Elas envolvem os desenvolvedores do processo de concepção para obter feedback rápido sobre a evolução dos requisitos. Assim, minimiza-se a documentação, pois se utiliza mais a comunicação informal do que reuniões formais com documentos escritos.

10. **(IESES / SCGás – 2019)** Os processos de software podem ser categorizados como dirigidos a planos ou processos ágeis. Considerando esta afirmação, assinale a afirmativa correta:

a) Nos processos ágeis todas as atividades são planejadas antecipadamente, e a avaliação do processo considera a comparação com um planejamento inicial. Já nos processos dirigido a planos, o planejamento é gradativo. Esta característica facilita a alteração do processo de forma a refletir as necessidades de mudança dos clientes.

b) Nos processos dirigidos a planos todas as atividades são planejadas antecipadamente, e a avaliação do processo considera a comparação com um planejamento inicial. Já nos processos ágeis, o planejamento é gradativo. Esta característica facilita a alteração do processo de forma a refletir as necessidades de mudança dos clientes.

c) Nos processos ágeis todas as atividades são planejadas posteriormente, e a avaliação do processo considera a comparação com um planejamento inicial. Já nos processos dirigido a planos, o planejamento é gradativo. Esta característica facilita a alteração do processo de forma a refletir as necessidades de mudança dos clientes.

d) Nos processos dirigidos a planos todas as rotinas são empíricas e, a avaliação do processo considera a comparação com um planejamento final a ser definido. Já nos processos ágeis, o planejamento é gradativo. Esta característica facilita a alteração do processo de forma a refletir as necessidades de mudança dos clientes.

11. **(INSTITUTO AOCP / EMPREL – 2019)** Em se tratando de desenvolvimento de software, o termo qualidade é bastante subjetivo. Entretanto, no desenvolvimento ágil, é claro o conceito

de qualidade. Sabendo disso, assinale a alternativa que apresenta corretamente o conceito de qualidade no desenvolvimento ágil.

a) Envolve a documentação do processo e o estabelecimento de práticas para entregar ao cliente um produto de qualidade.

b) Cumpre os critérios sistêmicos estabelecidos em acordo com o cliente para que os requisitos também sejam cumpridos.

c) Tem como objetivo gerar manuais e código claro por meio de uma equipe especializada no processo.

d) Cumpre os requisitos para o cliente com uma documentação completa do produto desenvolvido.

e) Significa que a qualidade do código e as práticas são utilizadas para garantir um código de alta qualidade.

12. **(IF-PE / IF-PE – 2019)** O Manifesto Ágil é um documento que encoraja a utilização de métodos melhores no desenvolvimento de software. Nele foram escritos doze princípios que norteiam o desenvolvimento ágil de sistemas. Um dos princípios mais relevantes é:

a) "A prioridade é satisfazer a equipe de desenvolvimento por meio de uma entrega única de software de valor."

b) "A prioridade é satisfazer ao cliente por meio de uma entrega única de software de valor."

c) "A prioridade é satisfazer ao gerente do projeto por meio de entregas contínuas e frequentes de software de valor."

d) "A prioridade é satisfazer ao gerente de projetos por meio de uma entrega única de software de valor."

e) "A prioridade é satisfazer ao cliente por meio de entregas contínuas e frequentes de software de valor."

13. **(AJURI / Desenvolve - RR – 2018)** Desenvolvimento ágil de software (em inglês: Agile software development) ou Método ágil é uma expressão que define um conjunto de metodologias utilizadas no desenvolvimento de software. As metodologias que fazem parte do conceito de desenvolvimento ágil, tal como qualquer metodologia de software, providenciam uma estrutura conceitual para reger projetos de engenharia de software. Métodos ágeis enfatizam comunicações em tempo real, preferencialmente cara a cara, a documentos escritos. A maioria dos componentes de um grupo ágil deve estar agrupada em uma sala. Isso inclui todas as pessoas necessárias para terminar o software: no mínimo, os programadores e seus

clientes(clientes são as pessoas que definem o produto, eles podem ser os gerentes, analistas de negócio, ou realmente os clientes). Considerando o contexto dos Valores da Metodologia Ágil, é correto afirmar que indivíduos e iterações:

a) mais do que processos e ferramentas; software funcional mais do que documentação abrangente; colaboração do cliente menor do que negociação de contratos; responder a mudanças menor do que seguir um plano.

b) mais do que processos e ferramentas; software funcional mais do que documentação abrangente; colaboração do cliente mais do que negociação de contratos; responder a mudanças mais do que seguir um plano.

c) mais do que processos e ferramentas; software funcional menos do que documentação abrangente; colaboração do cliente menor do que negociação de contratos; responder a mudanças na mesma medida que seguir um plano.

d) mais do que processos e ferramentas; software funcional mais do que documentação abrangente; colaboração do cliente na mesma medida que negociação de contratos; responder a mudanças na mesma medida que seguir um plano.

e) na mesma medida que processos e ferramentas; software funcional menos do que documentação abrangente; colaboração do cliente menor do que negociação de contratos; responder a mudanças menor do que seguir um plano.

14. **(INSTITUTO AOCP / PRODEB – 2018)** Assinale a alternativa que apresenta corretamente um dos princípios defendidos pelo Manifesto Ágil.

a) As melhores arquiteturas, requisitos e designs emergem de times com cronogramas bem definidos.

b) O método mais eficiente e eficaz de transmitir informações para um time de desenvolvimento é através de uma update meeting.

c) Deve-se construir projetos ao redor de estruturas hierárquicas verticais. Dando a eles o ambiente e suporte necessário.

d) Pessoas relacionadas a negócios devem trabalhar sem interferência constante ao time de desenvolvimento.

e) Em intervalos regulares, o time reflete como ficar mais efetivo, então se ajustam e otimizam seu comportamento de acordo.

15. **(INSTITUTO AOCP / PRODEB – 2018)** Assinale a alternativa que apresenta uma característica presente em Equipes ágeis:

a) Equipe grande.
b) Equipe modestamente motivada.
c) Equipe que se auto-organiza.
d) Individualismo e talento.
e) Alto formalismo.

16. **(INSTITUTO AOCP / PRODEB – 2018)** Assinale a alternativa correta em relação ao manifesto ágil para desenvolvimento de software.

a) Uma documentação detalhada é o método mais eficiente e eficaz de transmitir informações para e por dentro de um time de desenvolvimento.

b) Processos ágeis se adéquam a mudanças para que o cliente possa tirar vantagens competitivas.

c) Não se deve aceitar mudanças de requisitos no fim do desenvolvimento.

d) Pessoas relacionadas a negócios e desenvolvedores devem manter contato em reuniões específicas.

e) Deve-se aceitar mudança de requisitos porém o time deve parar o desenvolvimento e voltar à etapa de validação de requisitos.

17. **(INSTITUTO AOCP / PRODEB – 2018)** Com a realização do Manifesto Ágil em 2001 por um conjunto de especialistas em processos de desenvolvimento de software, ficaram definidos alguns parâmetros principais que passaram a ser um denominador comum de Metodologias Ágeis. São características atribuídas aos métodos ágeis, EXCETO:

a) processos e ferramentas ao contrário de pessoas e interações.
b) software executável, ao contrário de documentação extensa e confusa.
c) colaboração do cliente, ao contrário de constantes negociações de contratos.
d) indivíduos e interações mais que processos e ferramentas.
e) respostas rápidas para as mudanças, ao contrário de seguir planos previamente definidos.

18. **(FCM / IFN-MG – 2018)** O Manifesto Ágil para o Desenvolvimento de Software, proposto por Beck, K. et al. (2001), propõe 12 princípios. NÃO correspondem a um desses princípios criados por esses autores:

a) as melhores arquiteturas, requisitos e projetos emergem de equipes auto-organizadas.

b) a simplicidade é a arte de maximizar a quantidade de trabalho que não precisa ser feito.

c) o projeto para ser ágil precisa ter um controle bem definido sobre as pessoas e as tarefas que elas executam.

d) a prioridade é satisfazer o cliente através de entrega antecipada e contínua de um software que tenha valor para o mesmo.

e) a entrega do software deve ser feita com uma frequência predeterminada de tempo, preferencialmente em uma escala de tempo mais curta.

19. **(CS-UFG / UFG – 2019)** O desenvolvimento de software baseado em abordagem ágil estimula:

a) a produção de planos detalhados.
b) a realização de atividades de desenvolvimento em cada iteração.
c) a valorização da equipe de operação em detrimento daquela de desenvolvimento.
d) a aplicação de métodos formais de desenvolvimento de software.

20. **(INSTITUTO AOCP / ITEP – RN – 2018)** Qual das alternativas a seguir apresenta somente métodos ágeis de desenvolvimento de software?

a) XP e Scrum.
b) Cascata e XP.
c) Incremental e XP.
d) Evolucionário e Scrum.
e) Incremental e Evolucionário.

21. **(UECE-CEV / Prefeitura de Sobral - CE – 2018)** Escreva V ou F conforme seja verdadeiro ou falso o que se afirma nos itens abaixo com respeito ao processo de desenvolvimento ágil de software.

( ) Efetuar testes constantemente permite detectar defeitos mais cedo e da forma menos custosa possível.

( ) O uso de uma ferramenta robusta de modelagem e uma completa documentação são imprescindíveis para o desenvolvimento ágil.

( ) É importante produzir em poucas semanas uma versão inicial do software a fim de obter rapidamente uma primeira conquista e um feedback adiantado.

( ) Novas versões do software devem ser lançadas em intervalos cada vez mais frequentes, seja semanalmente, diariamente ou mesmo de hora em hora.

a) V, F, F, V.
b) F, V, F V.
c) V, F, V, F.

d) F, V, V, F.

22. **(CETRO / ANVISA – 2013)** Com relação aos conceitos do processo ágil, um dos conceitos-chave do Manifesto Ágil é :

I. produzir documentação em vez de software executável.
II. a colaboração do cliente em vez da negociação de contratos.
III. obter respostas rápidas a mudanças em vez de seguir planos.

É correto o que está contido em:

a) I, apenas.
b) II, apenas.
c) III, apenas.
d) II e III, apenas.
e) I, II e III.

23. **(UNIRIO / UNIRIO – 2014)** Dentre os princípios do manifesto ágil para desenvolvimento de software, NÃO se inclui (em):

a) a satisfação do cliente deve ser priorizada através da entrega contínua.
b) conversas face a face são preferíveis para e entre uma equipe de desenvolvimento.
c) simplicidade é essencial.
d) mudança nos requisitos devem ser evitadas.
e) entregas de software funcionando devem ser realizadas frequentemente.

24. **(FCM / IF-RS – 2016)** As metodologias ágeis tornaram-se populares em 2001 quando um grupo de especialistas em processos de desenvolvimento de software decidiu se reunir nos Estados Unidos. O objetivo foi discutir maneiras de melhorar o desempenho de seus projetos. Embora tivessem preferências e métodos distintos entre si, concordaram que um pequeno conjunto de princípios sempre parecia ter sido respeitado quando os projetos davam certo. Foi então criada a Aliança Ágil e o estabelecimento do Manifesto Ágil, contendo os conceitos e os princípios comuns compartilhados por todos esses métodos.

NÃO é considerado um princípio por trás do Manifesto Ágil:

a) Responder a mudanças mais que seguir um plano.
b) Colaboração com o cliente mais que negociação de contratos.
c) Processos e ferramentas mais que indivíduos e interação entre eles.
d) Software em funcionamento mais que documentação abrangente.
e) Indivíduos e interação entre eles mais que processos e ferramentas.

25. **(FUNCAB / MJ-SP – 2015)** O manifesto ágil considera que a medida primária de progresso é:

a) tempo utilizado.
b) quantidadede testes.
c) quantidadede documentação.
d) custo realizado.
e) software funcionando.

26. **(UECE-CEV / FUNCEME – 2018)** O Manifesto para o desenvolvimento ágil de software resume os itens mais valorizados pelos praticantes desta abordagem. Considerando os itens listados a seguir, assinale a opção que NÃO representa um valor ágil segundo o Manifesto.

a) indivíduos e interações mais que processos e ferramentas
b) seguir um plano mais que responder a mudanças
c) software em funcionamento mais que documentação abrangente
d) colaboração com o cliente mais que negociação de contratos

27. **(ESAF / MF – 2013)** O desenvolvimento ágil de software fundamenta-se no Manifesto Ágil. Segundo ele deve-se valorizar:

a) mudança de respostas em vez do seguimento de um plano.
b) indivíduos e interações em vez de processos e ferramentas.
c) documentação extensiva operacional em vez de software funcional.
d) indivíduos e intenções junto a processos e ferramentas.
- e) seguimento de um plano em vez de resposta a mudança.

28. **(IF-PE / IF-PE – 2016)** Sobre o documento conhecido como "manifesto ágil", é CORRETO dizer que:

a) prega uma extensa lista de documentos, processos, atores, métodos e diagramas visando fornecer alta agilidade.

b) lista e cataloga a maioria dos métodos vigentes à época de sua criação, classificando cada um como "ágil" ou "burocrático".

c) foi criado como base para descrever as principais ideias e práticas que eram comuns a muitos dos métodos considerados ágeis e que já existiam na época.

d) foi criado com base na ideia de que se tudo for muito bem controlado e documentado, os processos serão naturalmente ágeis.

e) a partir dele, foram definidos o XP, o scrum, o crystal, o CMM e o RUP, cada um com suas características particulares.

29. **(CS-UFG / UFG – 2018)** Ao se empregar métodos ágeis em desenvolvimento de software, as atividades:

a) são planejadas com antecedência, e seu progresso é medido em relação ao plano estabelecido.

b) são realizadas com base na abordagem iterativa/incremental de desenvolvimento.

c) são planejadas com base no modelo cascata, com fases separadas e distintas de especificação e desenvolvimento.

d) são realizadas em fases sequenciais, sendo que cada fase precisa estar completa antes que se passe para a próxima.

30. **(CESGRANRIO / Banco da Amazônia – 2018)** O Manifesto Ágil se tornou um marco da Engenharia de Software, chamando a atenção de que vários processos propostos de forma independente tinham valores em comum. Além disso, foram definidos 12 princípios. Entre eles, figura o seguinte princípio:

a) cada pessoa em um projeto deve ter sua função predeterminada para acelerar o desenvolvimento em conjunto.
b) a contínua atenção à simplicidade do trabalho feito aumenta a agilidade.
c) software funcionando é a medida primária de progresso.
d) os indivíduos, clientes e desenvolvedores, são mais importantes que processos e ferramentas.
e) o software funcional emerge de times auto-organizáveis.

31. **(IADES / ARCON-PA – 2018)** Embora esses métodos ágeis sejam todos baseados na noção de desenvolvimento e entrega incremental, eles propõem diferentes processos para alcançar tal objetivo. No entanto, compartilham um conjunto de princípios, com base no manifesto ágil, e por isso têm muito em comum.

SOMMERVILLE, I. Engenharia de software. 9. ed. São Paulo: Person Education, 2011.

Os cinco princípios citados no texto são:

a) envolvimento do cliente; entregas agendadas; pessoas e processos são igualmente importantes; aceitar mudanças; e manter a simplicidade.

b) envolvimento do cliente; entrega incremental; pessoas, não processos; aceitar as mudanças; e manter a simplicidade.

c) envolvimento do cliente apenas no início; entrega incremental; prazos rígidos; evitar mudanças; e manter a equipe.

d) programadores em primeiro lugar; ausência de prazos; cliente como última prioridade; aceitar as mudanças; e investir em controle de versão.

e) programadores em primeiro lugar; entrega por protótipos; processos, não pessoas; aceitar as mudanças; e manter o cronograma.

32. **(FAURGS / TJ-RS – 2018)** Considere as seguintes afirmações sobre princípios dos métodos ágeis.

I - Os clientes devem estar totalmente envolvidos no processo de desenvolvimento. Seu papel é fornecer e priorizar novos requisitos do sistema e avaliar suas iterações.

II - Embora as habilidades da equipe devam ser reconhecidas e exploradas, seus membros não devem desenvolver maneiras próprias de trabalhar, podendo o processo ser prescritivo.

III- Deve-se ter em mente que os requisitos do sistema irão mudar, por isso, o sistema deve ser projetado de maneira a acomodar essas mudanças.

Quais estão corretas?

a) Apenas I.
b) Apenas I e II.
c) Apenas I e III.
d) Apenas II e III.
e) I, II e III.

GABARITO

1. LETRA B
2. LETRA D
3. LETRA A
4. LETRA D
5. LETRA B
6. LETRA C
7. LETRA C
8. LETRA A
9. LETRA B
10. LETRA B
11. LETRA E
12. LETRA E
13. LETRA B
14. LETRA E
15. LETRA C
16. LETRA B
17. LETRA A
18. LETRA C
19. LETRA B
20. LETRA A
21. LETRA C
22. LETRA D
23. LETRA D
24. LETRA C
25. LETRA E
26. LETRA B
27. LETRA B
28. LETRA C
29. LETRA B
30. LETRA C
31. LETRA B
32. LETRA C

KANBAN

Conceitos Básicos

INCIDÊNCIA EM PROVA: MÉDIA

Seus lindos, vamos para outro assunto! *O que é esse tal de Kanban?* Kanban significa cartão ou placa visual, em japonês. **Trata-se de um método para gestão de mudanças** com foco na visualização do trabalho em progresso (também chamado de *Work In Progress* – WIP), identificando oportunidades de melhorias, tornando explícitas as políticas seguidas e os problemas encontrados e, por fim, favorecendo uma cultura de melhoria evolutiva.

Antes de continuar, eu tenho que fazer uma pequena pausa! **Há uma certa polêmica sobre se o Kanban é uma metodologia de desenvolvimento de software ou não!** David J. Anderson – pioneiro do Kanban – acha que não é (conforme podemos ver nas declarações apresentadas abaixo)! No entanto, é bastante comum ver algumas bancas o tratando como uma metodologia de desenvolvimento de software.

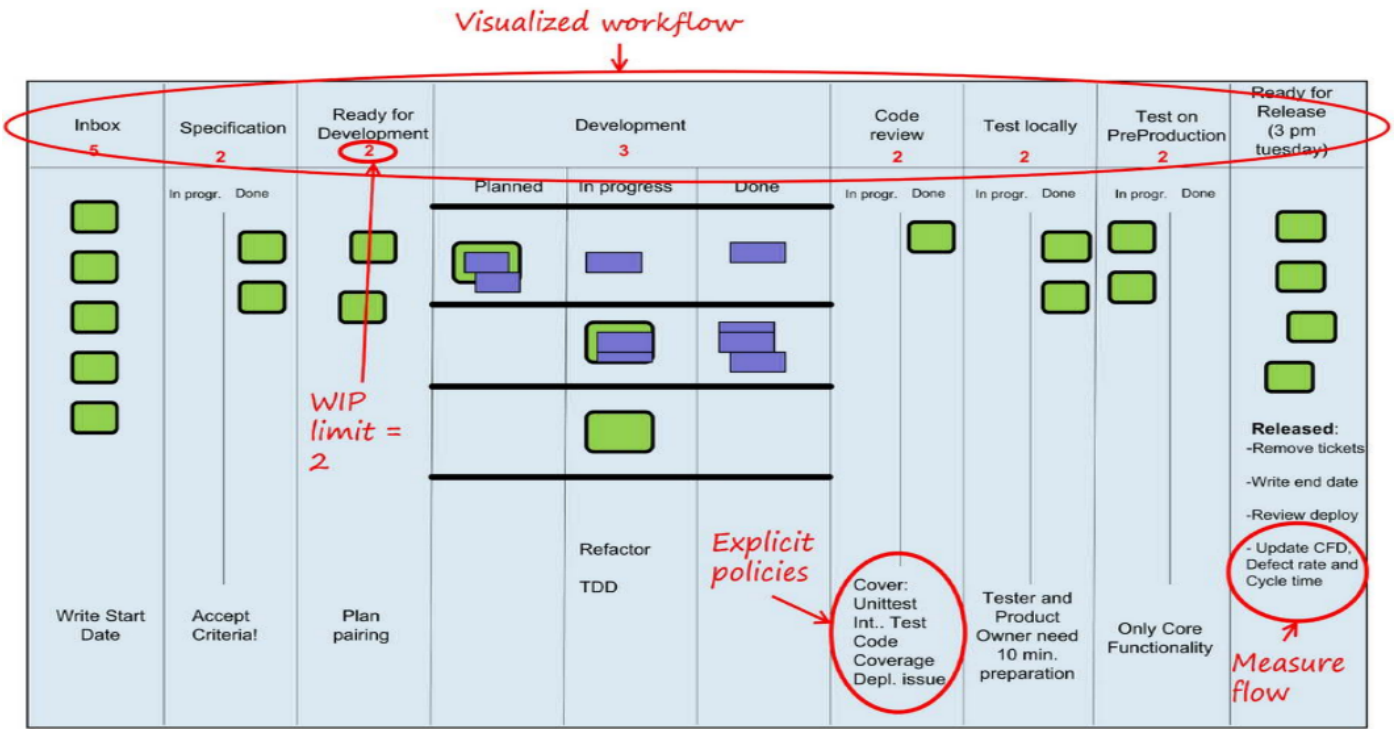
> *"Kanban **is not a software development life cycle or project management methodology**! It is not a way of making software or running projects that make software!" – David J. Anderson*
>
> *"There is **no kanban process for software development**. At least I am not aware of one. I have never published one" – David J. Anderson*
>
> *"It is actually **not possible to develop with only Kanban**. The Kanban Method by itself does not contain practices sufficient to do product development" – David J. Anderson*

O Kanban pode ser visto como um acelerador para a condução de mudanças ou até mesmo um método para implantação de mudanças em uma organização. Ele não prescreve papéis, práticas ou cerimônias específicas (como faz, por exemplo, Scrum). Em vez disso, ele oferece uma série de princípios para otimizar o fluxo e a geração de valor dos sistemas de entrega de software. *Agora o que ele tem a ver com a sua tradução em japonês?*

Galera, ele funciona como uma espécie de cartaz ou placa visual contendo vários post-its – aquele papelzinho colorido para colar lembretes. **Dessa forma, ele permite uma melhor visualização do fluxo de trabalho, favorecendo a transparência para todos os envolvidos.** Além disso, ele permite mudar prioridades facilmente e entregar funcionalidades a qualquer momento. Não há preocupação com estimativas nem em ser iterativo.

Como pode ser visto a partir da imagem apresentada a seguir, todo fluxo de trabalho se torna visível no Kanban. Os limites do Work in Progress são estabelecidos – o fluxo é contínuo sem requerer estimativas. A equipe assume a responsabilidade sobre o processo e se auto-organiza para

otimizá-lo e para ajudar a resolver seus eventuais problemas. O Kanban é construído sobre os conceitos de mudança evolucionária.



Dessa forma, uma possível abordagem é começar a entender como funciona atualmente seu sistema de desenvolvimento de software. Quando conseguir visualizar, medir e gerenciar o fluxo utilizado, melhore-o um passo por vez, aliviando seu maior gargalo, isto é, o processo evoluirá aos poucos. **Isso é muito diferente do que ocorre, por exemplo, no Scrum – em que se inicia definindo papéis, processos e artefatos.**

Isso faz do Kanban um método ideal para utilização em conjunto com outros processos – do Scrum ao Cascata. Ele também é excelente quando estruturas organizacionais inibem mudanças radicais, sendo construído principalmente sob o conceito de melhoria contínua. Ele somente utiliza mudanças radicais em situações especiais, nas quais mudanças estruturais são necessárias ou quando sérias mudanças de desempenho precisam ser feitas.

O modelo é uma boa opção tanto para desenvolvimento de software quanto para operação e manutenção. Kanban e Scrum não são opostos! **Nada impede que se comece a usar o Scrum e se utilize o Kanban para impulsionar mudanças futuras.** Os projetos – apesar de não insistirem no compromisso com iterações planejadas – são muito bem controlados, com cadência fixa, visualização permanente, medição do tempo de ciclo, fluxo de tarefas e ciclos de feedback curtos.

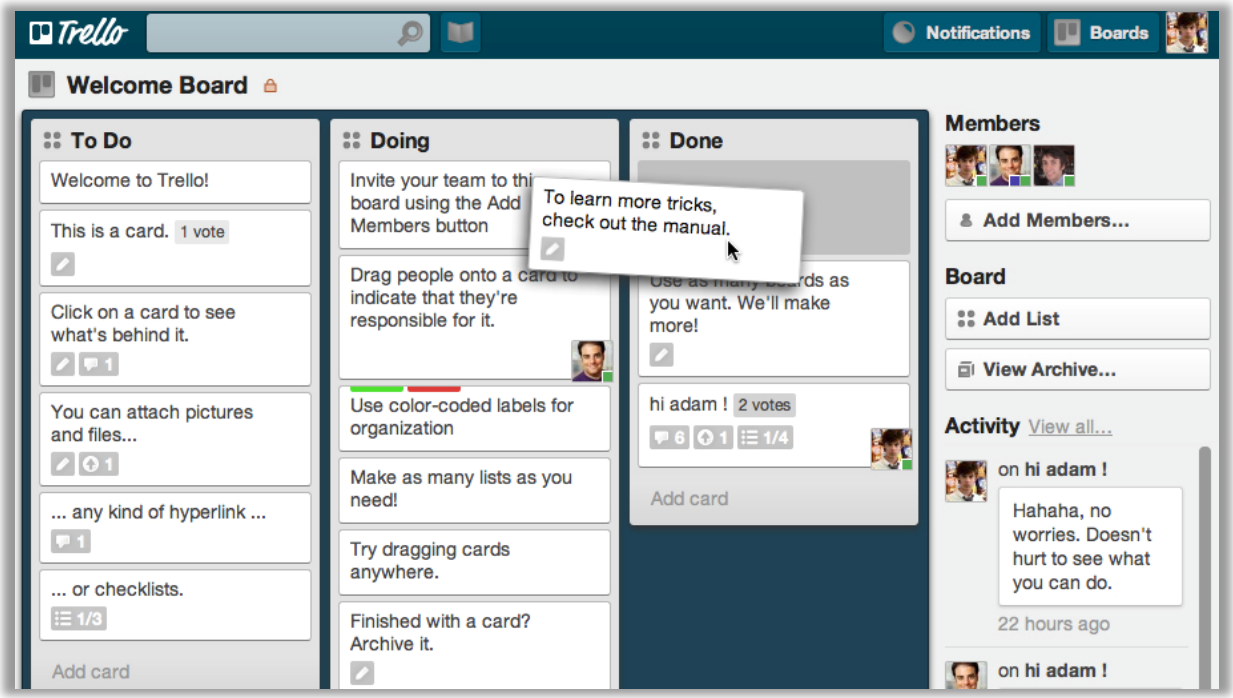
Galera, existem diversas diferenças entre ambos: Scrum requer iterações e o Kanban, não – mas sugere-se que haja uma cadência de entradas e entregas. Quando o Kanban incorpora iterações, ele é tipicamente chamado Scrumban (para diferenciá-lo do Scrum original), uma vez

que Scrum não gerencia explicitamente o trabalho em progresso e o Kanban não utiliza iterações por padrão. *Bacana?* Vamos ver agora os princípios ou restrições:

PRINCÍPIOS OU RESTRIÇÕES DO KANBAN
- Comece com o que você tem hoje;
- Estimule a liderança em todos os níveis da organização;
- Visualize cada passo em sua cadeia de valor, do conceito geral até o software que se possa lançar;
- Limite o Trabalho em Progresso (WIP), restringindo o total de trabalho permitido para cada estágio;
- Torne explícitas as políticas sendo seguidas;
- Meça e gerencie o fluxo, para poder tomar decisões bem embasadas, além de visualizar as consequências;
- Identifique oportunidades de melhorias, na qual a melhoria contínua é responsabilidade de todos.

PRÁTICAS DO KANBAN
- Implemente mecanismos de feedback;
- Gerencie e meça o fluxo de trabalho;
- Visualize o processo;
- Limite o WIP (Work In Progress);
- Torne as políticas dos processos explícitas;
- Melhore colaborativamente e com métodos científicos.

Vamos detalhar o WIP! Trata-se de tarefas que estão em execução em determinado ponto do processo. *Por que devemos limitar o WIP?* **Porque quanto maior o número de tarefas em andamento em determinado ponto do processo, mais tempo a tarefa permanecerá no fluxo.** *Então ele deve ser pequeno?* Não, ele não deve ser muito pequeno nem muito grande. Em geral, se for muito pequeno, qualquer limitação pode parar o processo de desenvolvimento.



E se for muito grande? Nesse caso, muitas tarefas simultâneas levam a grandes perdas e confusões. *Então, qual é o tamanho ideal?* **Bem, isso não existe! Não há um número mágico – é necessário descobrir o tamanho empiricamente de acordo com o contexto da organização.** Ahhh... se você utiliza o Trello, ele é uma forma de apresentar o trabalho sendo realizado, mas também existem outras ferramentas (KanbanFlow, Kanbanery, Leankit, Visual WIP, etc).

QUESTÕES COMENTADAS – DIVERSAS BANCAS

1. **(FGV / SEFAZ-MT - 2023)** Muitas organizações aplicam os princípios e práticas de gestão com Kanban para alcançar os objetivos de seus projetos de desenvolvimento de software. Sobre Kanban, assinale a afirmativa correta:

   a) Incentiva ciclos anuais de feedbacks porque cadências de reuniões e revisões recorrentes criam muitas interrupções e impactam o tempo de entrega.

   b) É um método de gestão baseado no Kaizen e que valoriza entrega de software funcional completo, sem entregas parciais, trabalhos inacabados e débitos técnico.

   c) Fomenta uma cultura organizacional que privilegia o indivíduo em relação à coletividade, orientada ao cliente e sem compartilhar propósitos e metas.

   d) Pressupõe que o escopo do projeto é sempre estável e o desenvolvimento de software precisa seguir de modo linear para garantir a produtividade.

   e) Permite visualizar o processo e o fluxo de trabalho de projetos, programas e portfólios por meio de um conjunto de quadros interconectados.

Comentários:

O Kanban é uma técnica de gestão visual que é amplamente utilizada para gerenciar o fluxo de trabalho em diversos contextos, inclusive no desenvolvimento de software. A principal ideia por trás do Kanban é que você pode visualizar o seu trabalho e o fluxo de trabalho em um quadro Kanban, o que ajuda a identificar gargalos e ineficiências no processo.

(a) Errado. O Kanban não incentiva ciclos anuais de feedbacks, mas sim a utilização de cadências de reuniões e revisões regulares para melhorar continuamente o processo.

(b) Errado. Embora o Kanban valorize a entrega de software funcional completo, ele também permite a entrega de trabalhos parciais e a gestão do débito técnico de forma consciente.

(c) Errado. O Kanban promove uma cultura organizacional colaborativa e orientada ao cliente, valorizando a coletividade e o compartilhamento de propósitos e metas.

(d) Errado. O Kanban não pressupõe que o escopo do projeto seja sempre estável, e sim permite a adaptação e flexibilidade durante o desenvolvimento de software, seguindo abordagens ágeis.

Gabarito: Letra E

2. **(CESPE / BANRISUL – 2022)** O método Kanban pode ser utilizado em substituição à metodologia Scrum, mas também ambos podem ser combinados para o alcance de resultados mais eficazes.

Comentários:

Opa! O Kanban não pode ser utilizado como um substituto para o Scrum, mas os dois podem ser combinados para resultados mais eficazes.

Gabarito: Errado

---

3. **(CESPE / BANRISUL – 2022)** As equipes que utilizam o método Kanban não utilizam timeboxes, embora a maioria das equipes pratique uma cadência fixa de planejamento, revisão e entregas.

Comentários:

O método Kanban não é estruturado em torno de timeboxes, pois é mais flexível. No Kanban, o processo de desenvolvimento de software é dividido em etapas, que podem ser executadas de forma contínua e contínua. Isso significa que, em vez de criar um calendário de entrega e seguir um cronograma, as equipes Kanban se concentram em entregar trabalho de forma constante e incremental. Como resultado, as equipes podem responder mais rapidamente às mudanças no projeto, pois não estão limitadas por um cronograma pré-definido.

Gabarito: Correto

---

4. **(CESPE / BANRISUL – 2022)** O Kanban, devido à adoção dos princípios Lean, é um método ideal para utilização em projetos que adotam o Scrum; por outro lado, não se aplica a projetos tradicionais do tipo cascata.

Comentários:

Ele pode, sim, ser aplicado a projetos tradicionais do tipo cascata. O Kanban não é exclusivo para metodologias ágeis, sendo possível aplicá-lo em outros tipos de projetos. O objetivo do Kanban é ajudar os times a planejar, monitorar e gerenciar o trabalho, o que o torna uma ferramenta útil em qualquer tipo de projeto.

Gabarito: Errado

---

5. **(CESPE / BANRISUL – 2022)** O WIP descreve o total de trabalho que está em progresso no Kanban, podendo incluir todos os itens ou apenas aqueles selecionados para implementação.

Comentários:

O WIP (Work in Progress) descreve o total de trabalho que está em progresso no Kanban. Esta informação é usada para ajudar a equipe a tomar decisões informadas sobre quanto trabalho ela pode realizar ao mesmo tempo. O WIP pode incluir todos os itens que estão no Kanban, ou apenas aqueles itens selecionados para implementação. Isso depende da abordagem de gerenciamento de projeto adotada pela equipe.

Gabarito: Correto

---

6. **(CESPE / BANRISUL – 2022)** O Kanban é um método de gestão de mudanças que dá ênfase à visualização do trabalho em andamento.

Comentários:

Definição simples e precisa de Kanban! Nada a acrescentar...

Gabarito: Correto

---

7. **(CESPE / BNB – 2022)** Diferentemente do Scrum, o Kanban não prescreve interações com metas pré-definidas e de mesmo tamanho para a execução de atividades, como, por exemplo, as de planejamento, de desenvolvimento e de liberação.

Comentários:

Perfeito! O Kanban é uma metodologia de gerenciamento de projetos semelhante ao Scrum, mas com algumas diferenças significativas. Ele é focado na visualização do fluxo de trabalho, na limitação do trabalho em progresso e na otimização contínua. Em contraste com o Scrum, o Kanban não determina a duração de cada ciclo de trabalho. Em vez disso, os projetos são divididos em pequenas unidades de trabalho e as tarefas são desenvolvidas de forma iterativa sem metas pré-definidas. O processo de desenvolvimento é então ajustado de acordo com as necessidades da equipe.

Gabarito: Correto

---

8. **(FGV / BANESTES – 2021)** Observe o quadro comparativo a seguir, publicado em sites ligados ao estudo e à investigação de diferentes estratégias/metodologias para implementar um sistema ágil de desenvolvimento ou gestão de projetos.

Aspectos	X	Y
Ritmo	Sprints	Fluxo contínuo
Funções	Funções bem definidas	Sem funções necessárias
Entregas	Final de cada sprint	Entrega contínua
Mudanças	Evitar durante sprint	A qualquer momento

É correto identificar que X e Y representam, respectivamente:

a) Crystal e Scrum;
b) Extreme Programming e Crystal;
c) Kanban e Lean;
d) Lean e Extreme Programming;
e) Scrum e Kanban.

Comentários:

Usando apenas o primeiro critério, já é possível responder à questão: Sprint é típico do Scrum e Fluxo Contínuo é típico do Kanban.

Gabarito: Letra E

---

9. **(CESPE / TCU – 2015)** O método para a implantação de mudanças denominado Kanban não prevê papéis nem cerimônias específicas.

Comentários:

Realmente não se prevê papéis ou cerimônias no Kanban.

Gabarito: Correto

---

10. **(CESPE / PROCEMPA – 2014)** Kanban considera a utilização de uma sinalização ou registro visual para gerenciar o limite de atividades em andamento, indicando se um novo trabalho pode ou não ser iniciado e se o limite acordado para cada fase está sendo respeitado.

Comentários:

Esse registro visual é o famoso WIP.

Gabarito: Correto

---

11. **(FGV / TJ-GO – 2014)** Scrum e Kanban são metodologias de gerenciamento de projetos de software populares entre praticantes do desenvolvimento ágil. Um aspecto de divergência entre as duas metodologias é:

a) processo incremental;
b) processo iterativo;
c) uso de quadro de tarefas;
d) apresentação do estágio de desenvolvimento de uma tarefa;
e) valorização de feedback.

Comentários:

O Kanban não é necessariamente iterativo como é o Scrum! Na prática, esse assunto é rodeado de polêmicas e divergências. Já os outros itens tratam de convergências!

Gabarito: Letra B

---

12. (FGV / MPE-MS – 2013) Kanban é um dos métodos ágeis mais recentes e sofreu grande influência do movimento "Lean", surgido nos anos 1980. São práticas comuns a esse método:

a) limitar o WIP (Work In Progress) e uma visualização explícita do fluxo de trabalho.
b) integração Contínua e gerenciamento de configuração.
c) limitar o WIP (Work In Progress) e gerenciamento de configuração.
d) gerenciar o fluxo de trabalho e manter estimativas previamente definidas.
e) melhoria contínua e nunca limitar o WIP para evitar folgas no sistema de trabalho.

Comentários:

As práticas são: implemente mecanismos de feedback; gerencie e meça o fluxo de trabalho; visualize o processo; limite o WIP (Work In Progress); torne as políticas dos processos explícitas; melhore colaborativamente e com métodos científicos.

Gabarito: Letra A

---

13. (CESPE / SEDF – 2017) A técnica de Kanban é uma forma simples de visualizar o andamento das tarefas da equipe durante uma Sprint de Scrum. Nessa técnica, as tarefas são representadas por meio de pequenos papéis que indicam o que está pendente, em desenvolvimento e finalizado. Com isso, todos visualizam os gargalos e a equipe se organiza melhor, principalmente quando o projeto envolve ciclos longos de desenvolvimento.

Comentários:

Ela realmente pode ser considerada uma técnica que ajuda a visualizar o andamento do projeto – ele é muito utilizado em conjunto com o Scrum.

Gabarito: Correto

---

14. (FCC / TST – 2017) Um Analista de Sistemas do Tribunal Superior do Trabalho – TST, de modo hipotético, necessitou aplicar princípios ágeis e de controle usando elementos de três modelos, em processos de manutenção de software. Considere:

I. Dividir o cronograma em iterações time-box ou ciclos (sprints).

II. Orientar o trabalho a eventos ao invés de limite de tempo.
III. Aplicar a programação em pares, integração contínua, orientação a testes (TDD), revisão de código e todas as demais prescrições antes da implantação.

As características acima correspondem, respectivamente, a:

a) Kanban, XP e Scrum.
b) Kanban, Scrum e XP.
c) XP, Scrum e Kanban.
d) Scrum, XP e Kanban.
e) Scrum, Kanban e XP.

Comentários:

(I) Quem divide o cronograma em iterações time-box é o Scrum; (II) Quem orienta o trabalho a eventos ao invés de limite de tempo é o Kanban; (III) Quem aplica a programação em pares, TDD e revisão de código é o XP.

Gabarito: Letra E

---

15. (CESPE / STM – 2018) A implementação de um Kanban pressupõe a definição de um fluxo de trabalho pela equipe, o qual poderá ser revisto, mediante a inclusão ou a retirada de estágios, à medida que o trabalho evoluir.

Comentários:

Ele realmente pressupõe um fluxo de trabalho e não há nenhum problema em revisitá-lo com a inclusão ou exclusão de estágios com a evolução do trabalho.

Gabarito: Correto

---

16. (FCC / MPE-PE – 2018) Enquanto o processo de desenvolvimento Scrum usa sprints formais (ciclos de trabalho) com funções específicas atribuídas, o Kanban:

a) não define sprints formais nem papéis específicos para os integrantes da equipe do projeto.

b) não define ciclos formais, porém, prescreve papéis específicos para todos os integrantes da equipe do projeto.

c) define ciclos formais (sprints), porém, não define papéis específicos para os integrantes da equipe do projeto.

d) define ciclos formais de até 4 semanas e papéis específicos para os integrantes da equipe de desenvolvimento.

e) define apenas os papéis de Gerente de Projeto e Líder de Equipe, tendo o desenvolvimento pautado por ciclos de duas semanas chamados slices.

Comentários:

(a) Correto, ele não define iterações (muito menos sprints) – ou papeis específicos; (b) Errado, ele não prescreve papeis para os integrantes da equipe; (c) Errado, ele não define ciclos formais ou sprints; (d) Errado, ele não define ciclos formais; (e) Errado, ele não define nenhum papel.

Gabarito: Letra A

---

17. **(IF-RS / IF-RS – 2018)** Kanban foi criado pela Toyota com o objetivo de controlar melhor os níveis enormes de estoque em relação ao consumo real de materiais. Devido à sua eficiência, muitas empresas adotaram esse sistema para controlar tarefas das equipes do setor de Tecnologia da Informação. A respeito do Kanban, conforme visto em Dooley (2017), classifique cada uma das afirmativas abaixo como verdadeira (V) ou falsa (F) e assinale a alternativa que apresenta a sequência CORRETA, de cima para baixo:

( ) Através do quadro Kanban, compartilhado por todos, torna-se possível visualizar as tarefas com que cada membro da equipe está envolvido.

( ) Diferente do Scrum, Kanban baseia-se em iterações de tempo fixo. Os projetos são divididos em ciclos semanais denominados Sprints.

( ) Usa três ideias para influenciar um processo de desenvolvimento: trabalho em andamento (WIP), fluxo de trabalho e o custo médio financeiro.

( ) Geralmente utilizam-se post-its ou cartões de índice para representar uma tarefa no quadro Kanban.

a) V – V – F – V
b) V – F – F – V
c) F – F – V – F
d) V – V – V – V
e) F – F – V – V

Comentários:

(V) Correto, o WIP é compartilhado com todos para dar transparência; (F) Errado, ele não possui iterações nem ciclos formais ou sprints; (F) Errado, não existe a ideia de custo médio financeiro; (V) Correto, são realmente utilizados os post-its para representar tarefas.

Gabarito: Letra B

18.(CESPE / SERPRO – 2013) Kanban é um método de desenvolvimento de software que tem como uma de suas práticas o gerenciamento do fluxo de trabalho, que deve ser monitorado, medido e reportado a cada estado do fluxo.

Comentários:

Ele pode ser considerado um método de desenvolvimento de software e o fluxo de trabalho é constantemente monitorado a cada mudança, por meio de um quadro de fluxo.

Gabarito: Correto

LISTA DE QUESTÕES – DIVERSAS BANCAS

1. **(FGV / SEFAZ-MT - 2023)** Muitas organizações aplicam os princípios e práticas de gestão com Kanban para alcançar os objetivos de seus projetos de desenvolvimento de software. Sobre Kanban, assinale a afirmativa correta:

   a) Incentiva ciclos anuais de feedbacks porque cadências de reuniões e revisões recorrentes criam muitas interrupções e impactam o tempo de entrega.

   b) É um método de gestão baseado no Kaizen e que valoriza entrega de software funcional completo, sem entregas parciais, trabalhos inacabados e débitos técnico.

   c) Fomenta uma cultura organizacional que privilegia o indivíduo em relação à coletividade, orientada ao cliente e sem compartilhar propósitos e metas.

   d) Pressupõe que o escopo do projeto é sempre estável e o desenvolvimento de software precisa seguir de modo linear para garantir a produtividade.

   e) Permite visualizar o processo e o fluxo de trabalho de projetos, programas e portfólios por meio de um conjunto de quadros interconectados.

2. **(CESPE / BANRISUL – 2022)** O método Kanban pode ser utilizado em substituição à metodologia Scrum, mas também ambos podem ser combinados para o alcance de resultados mais eficazes.

3. **(CESPE / BANRISUL – 2022)** As equipes que utilizam o método Kanban não utilizam timeboxes, embora a maioria das equipes pratique uma cadência fixa de planejamento, revisão e entregas.

4. **(CESPE / BANRISUL – 2022)** O Kanban, devido à adoção dos princípios Lean, é um método ideal para utilização em projetos que adotam o Scrum; por outro lado, não se aplica a projetos tradicionais do tipo cascata.

5. **(CESPE / BANRISUL – 2022)** O WIP descreve o total de trabalho que está em progresso no Kanban, podendo incluir todos os itens ou apenas aqueles selecionados para implementação.

6. **(CESPE / BANRISUL – 2022)** O Kanban é um método de gestão de mudanças que dá ênfase à visualização do trabalho em andamento.

7. **(CESPE / BNB – 2022)** Diferentemente do Scrum, o Kanban não prescreve interações com metas pré-definidas e de mesmo tamanho para a execução de atividades, como, por exemplo, as de planejamento, de desenvolvimento e de liberação.

8. **(FGV / BANESTES – 2021)** Observe o quadro comparativo a seguir, publicado em sites ligados ao estudo e à investigação de diferentes estratégias/metodologias para implementar um sistema ágil de desenvolvimento ou gestão de projetos.

Aspectos	X	Y
Ritmo	Sprints	Fluxo contínuo
Funções	Funções bem definidas	Sem funções necessárias
Entregas	Final de cada sprint	Entrega contínua
Mudanças	Evitar durante sprint	A qualquer momento

É correto identificar que X e Y representam, respectivamente:

a) Crystal e Scrum;
b) Extreme Programming e Crystal;
c) Kanban e Lean;
d) Lean e Extreme Programming;
e) Scrum e Kanban.

9. **(CESPE / TCU – 2015)** O método para a implantação de mudanças denominado Kanban não prevê papéis nem cerimônias específicas.

10. **(CESPE / PROCEMPA – 2014)** Kanban considera a utilização de uma sinalização ou registro visual para gerenciar o limite de atividades em andamento, indicando se um novo trabalho pode ou não ser iniciado e se o limite acordado para cada fase está sendo respeitado.

11. **(FGV / TJ-GO – 2014)** Scrum e Kanban são metodologias de gerenciamento de projetos de software populares entre praticantes do desenvolvimento ágil. Um aspecto de divergência entre as duas metodologias é:

a) processo incremental;
b) processo iterativo;
c) uso de quadro de tarefas;
d) apresentação do estágio de desenvolvimento de uma tarefa;
e) valorização de feedback.

12. **(FGV / MPE-MS – 2013)** Kanban é um dos métodos ágeis mais recentes e sofreu grande influência do movimento "Lean", surgido nos anos 1980. São práticas comuns a esse método:

a) limitar o WIP (Work In Progress) e uma visualização explícita do fluxo de trabalho.
b) integração Contínua e gerenciamento de configuração.
c) limitar o WIP (Work In Progress) e gerenciamento de configuração.
d) gerenciar o fluxo de trabalho e manter estimativas previamente definidas.
e) melhoria contínua e nunca limitar o WIP para evitar folgas no sistema de trabalho.

13. **(CESPE / SEDF – 2017)** A técnica de Kanban é uma forma simples de visualizar o andamento das tarefas da equipe durante uma Sprint de Scrum. Nessa técnica, as tarefas são representadas por meio de pequenos papéis que indicam o que está pendente, em desenvolvimento e finalizado. Com isso, todos visualizam os gargalos e a equipe se organiza melhor, principalmente quando o projeto envolve ciclos longos de desenvolvimento.

14. **(FCC / TST – 2017)** Um Analista de Sistemas do Tribunal Superior do Trabalho – TST, de modo hipotético, necessitou aplicar princípios ágeis e de controle usando elementos de três modelos, em processos de manutenção de software. Considere:

I. Dividir o cronograma em iterações time-box ou ciclos (sprints).
II. Orientar o trabalho a eventos ao invés de limite de tempo.
III. Aplicar a programação em pares, integração contínua, orientação a testes (TDD), revisão de código e todas as demais prescrições antes da implantação.

As características acima correspondem, respectivamente, a:

a) Kanban, XP e Scrum.
b) Kanban, Scrum e XP.
c) XP, Scrum e Kanban.
d) Scrum, XP e Kanban.
e) Scrum, Kanban e XP.

15. **(CESPE / STM – 2018)** A implementação de um Kanban pressupõe a definição de um fluxo de trabalho pela equipe, o qual poderá ser revisto, mediante a inclusão ou a retirada de estágios, à medida que o trabalho evoluir.

16. **(FCC / MPE-PE – 2018)** Enquanto o processo de desenvolvimento Scrum usa sprints formais (ciclos de trabalho) com funções específicas atribuídas, o Kanban:

a) não define sprints formais nem papéis específicos para os integrantes da equipe do projeto.

b) não define ciclos formais, porém, prescreve papéis específicos para todos os integrantes da equipe do projeto.

c) define ciclos formais (sprints), porém, não define papéis específicos para os integrantes da equipe do projeto.

d) define ciclos formais de até 4 semanas e papéis específicos para os integrantes da equipe de desenvolvimento.

e) define apenas os papéis de Gerente de Projeto e Líder de Equipe, tendo o desenvolvimento pautado por ciclos de duas semanas chamados slices.

17. **(IF-RS / IF-RS – 2018)** Kanban foi criado pela Toyota com o objetivo de controlar melhor os níveis enormes de estoque em relação ao consumo real de materiais. Devido à sua eficiência, muitas empresas adotaram esse sistema para controlar tarefas das equipes do setor de Tecnologia da Informação. A respeito do Kanban, conforme visto em Dooley (2017), classifique cada uma das afirmativas abaixo como verdadeira (V) ou falsa (F) e assinale a alternativa que apresenta a sequência CORRETA, de cima para baixo:

( ) Através do quadro Kanban, compartilhado por todos, torna-se possível visualizar as tarefas com que cada membro da equipe está envolvido.

( ) Diferente do Scrum, Kanban baseia-se em iterações de tempo fixo. Os projetos são divididos em ciclos semanais denominados Sprints.

( ) Usa três ideias para influenciar um processo de desenvolvimento: trabalho em andamento (WIP), fluxo de trabalho e o custo médio financeiro.

( ) Geralmente utilizam-se post-its ou cartões de índice para representar uma tarefa no quadro Kanban.

a) V – V – F – V
b) V – F – F – V
c) F – F – V – F
d) V – V – V – V
e) F – F – V – V

18. **(CESPE / SERPRO – 2013)** Kanban é um método de desenvolvimento de software que tem como uma de suas práticas o gerenciamento do fluxo de trabalho, que deve ser monitorado, medido e reportado a cada estado do fluxo.

GABARITO – DIVERSAS BANCAS

1. LETRA E
2. ERRADO
3. CORRETO
4. ERRADO
5. CORRETO
6. CORRETO
7. CORRETO
8. LETRA E
9. CORRETO
10. CORRETO
11. LETRA B
12. LETRA A
13. CORRETO
14. LETRA E
15. CORRETO
16. LETRA A
17. LETRA B
18. CORRETO

TEST DRIVEN DEVELOPMENT (TDD)

Conceitos Básicos

INCIDÊNCIA EM PROVA: MÉDIA

O Test-Driven Development (TDD) é uma abordagem de desenvolvimento de software em que se intercalam testes e desenvolvimento de código. Essencialmente, você desenvolve um código de forma incremental, em conjunto com um teste para esse incremento. Você não caminha para o próximo incremento até que o código desenvolvido passe no teste. O desenvolvimento dirigido a testes foi apresentado como parte dos métodos ágeis, como o XP.

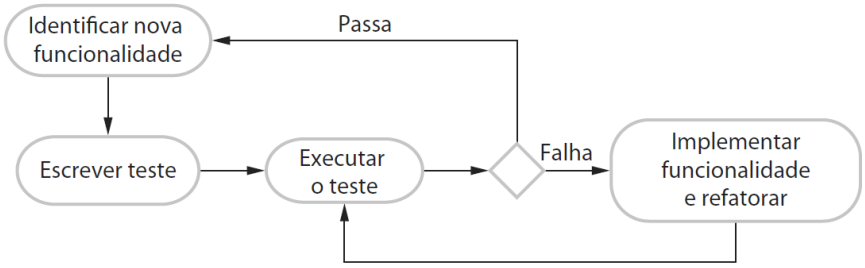
Por outro lado, ele também pode ser utilizado em processos de desenvolvimento dirigido a planos. Trata-se de uma abordagem que **se baseia na repetição de um ciclo de desenvolvimento curto focado em testes unitários**. A ideia fundamental dessa abordagem consiste em escrever o teste, encontrar uma falha nesse mesmo teste e depois refatorá-lo. Vamos agora ver as etapas do processo de desenvolvimento dirigido a testes:

ETAPA	DESCRIÇÃO
1	Você começa identificando o incremento de funcionalidade necessário. Este, normalmente, deve ser pequeno e implementável em poucas linhas de código.
2	Você escreve um teste para essa funcionalidade e o implementa como um teste automatizado. Isso significa que o teste pode ser executado e relatará se passou ou falhou.
3	Você, então, executa o teste, junto com todos os outros testes implementados. Inicialmente, você não terá implementado a funcionalidade, logo o novo teste falhará. Isso é proposital, pois mostra que o teste acrescenta algo ao conjunto de testes.
4	Você, então, implementa a funcionalidade e executa novamente o teste. Isso pode envolver a refatoração do código existente para melhorá-lo e adicionar um novo código sobre o que já está lá.
5	Depois que todos os testes forem executados com sucesso, você caminha para implementar a próxima parte da funcionalidade.

Como o código é desenvolvido em incrementos muito pequenos, você precisa ser capaz de executar todos os testes cada vez que adicionar funcionalidade ou refatorar o programa. Dessa forma, os testes são embutidos em um programa separado que os executa e invoca o sistema que está sendo testado. Usando essa abordagem, é possível rodar centenas e centenas de testes separados em poucos segundos.

Um argumento forte a favor do desenvolvimento dirigido a testes é que ele ajuda os programadores a clarear suas ideias sobre o que um segmento de código supostamente deve fazer**. Para escrever um teste, você precisa entender a que ele se destina, e como esse entendimento faz que seja**

mais fácil escrever o código necessário. Certamente, se você tem conhecimento ou compreensão incompleta, o desenvolvimento dirigido a testes não ajudará.



Se você não sabe o suficiente para escrever os testes, não vai desenvolver o código necessário. Por exemplo: se seu cálculo envolve divisão, você deve verificar se não está dividindo o número por zero. Se você se esquecer de escrever um teste para isso, então o código para essa verificação nunca será incluído no programa. Além de um melhor entendimento do problema, outros benefícios do desenvolvimento dirigido a testes são:

BENEFÍCIOS	DESCRIÇÃO
COBERTURA DE CÓDIGO	Em princípio, todo segmento de código que você escreve deve ter pelo menos um teste associado. Assim, você pode ter certeza de que todo o código no sistema foi realmente executado. Cada código é testado enquanto está sendo escrito; assim, os defeitos são descobertos no início do processo de desenvolvimento.
TESTE DE REGRESSÃO	Um conjunto de testes é desenvolvido de forma incremental enquanto um programa é desenvolvido. Você sempre pode executar testes de regressão para verificar se as mudanças no programa não introduziram novos bugs.
DEPURAÇÃO SIMPLIFICADA	Quando um teste falha, a localização do problema deve ser óbvia. O código recém-escrito precisa ser verificado e modificado. Você não precisa usar as ferramentas de depuração para localizar o problema. Alguns relatos de uso de desenvolvimento dirigido a testes sugerem que, em desenvolvimento dirigido a testes, quase nunca é necessário usar um sistema automatizado de depuração.
DOCUMENTAÇÃO DE SISTEMA	Os testes em si mesmos agem como uma forma de documentação que descreve o que o código deve estar fazendo. Ler os testes pode tornar mais fácil a compreensão do código.

Um dos benefícios mais importantes de desenvolvimento dirigido a testes é que ele reduz os custos dos testes de regressão. O teste de regressão envolve a execução de conjuntos de testes que tenham sido executados com sucesso, após as alterações serem feitas em um sistema. Ele também verifica se essas mudanças não introduziram novos bugs no sistema e se o novo código interage com o código existente conforme o esperado.

O teste de regressão é muito caro e geralmente impraticável quando um sistema é testado manualmente, pois os custos com tempo e esforço são muito altos. Em tais situações, você

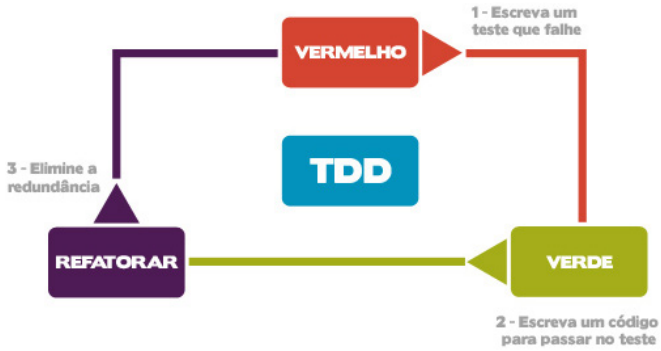
precisa tentar escolher os testes mais relevantes para executar novamente, e é fácil perder testes importantes. No entanto, testes automatizados – fundamentais para o desenvolvimento *test-first* – reduzem drasticamente os custos com testes de regressão.

Os testes existentes podem ser executados novamente de forma rápida e barata. **Após se fazer uma mudança para um sistema em desenvolvimento *test-first*, todos os testes existentes devem ser executados com êxito antes de qualquer funcionalidade ser adicionada.** Como um programador, você precisa ter certeza de que a nova funcionalidade não tenha causado ou revelado problemas com o código existente.

O desenvolvimento dirigido a testes é de maior utilidade no desenvolvimento de softwares novos, em que a funcionalidade seja implementada no novo código ou usando bibliotecas-padrão já testadas. **Se você estiver reusando componentes de código ou sistemas legados grandes, você precisa escrever testes para esses sistemas como um todo.** O desenvolvimento dirigido a testes também pode ser ineficaz em sistemas multi-threaded.

As *threads* diferentes podem ser intercalados em tempos diferentes, em execuções diferentes, e isso pode produzir resultados diferentes. Se você usa o desenvolvimento dirigido a testes, ainda precisa de um processo de teste de sistema para validar o sistema, isto é, verificar se atende aos requisitos dos stakeholders. O teste de sistema também testa o desempenho, a confiabilidade, e verifica se o sistema não faz coisas que não deveria, como produzir resultados indesejados etc.

O desenvolvimento dirigido a testes revelou-se uma abordagem de sucesso para projetos de pequenas e médias empresas. Geralmente, os programadores que adotaram essa abordagem estão satisfeitos com ela e acham que é uma maneira mais produtiva de desenvolver softwares. Em alguns experimentos, foi mostrado que essa abordagem gera melhorias na qualidade do código. Bem, agora vamos falar sobre o ciclo vermelho, verde e refatoração.



ETAPA	DESCRIÇÃO
VERMELHO (RED)	Escreva um teste que falha: casos de teste são escritos sem que exista o recurso a ser testado, levando o teste a falhar;
VERDE (GREEN)	Escreva código para passar no teste: o recurso é implementado com o mínimo de código necessário para que passe no teste.
REFATORAR (REFACTOR)	Elimine redundâncias: o código que implementa o recurso é refinado e melhorado, sem que seja adicionada novas funcionalidades.

Vamos lá! Nós sabemos que – para cada parte da aplicação – adiciona-se um teste escrito antes mesmo do desenvolvimento do código em si. *Por que?* **Porque eles podem ajudar a reduzir riscos de possíveis problemas no código.** Executamos o teste e ele... falha! Ele deve necessariamente falhar! *Por que?* Ora, porque ele é o primeiro teste e você nem criou a funcionalidade ainda, logo ele não irá funcionar!

Então nós adicionamos uma nova funcionalidade ao sistema apenas para que ele passe no teste e execute novamente (agora ele deve passar no teste). **Então, nós adicionamos um novo teste e rodamos o teste anterior e esse novo teste.** Se algum deles falhar, modifica-se o código da funcionalidade e rodam-se todos os testes novamente, e assim por diante – nós já vimos aquela imagem anterior que mostra como tudo isso funciona...

Galera, vocês percebem que o feedback sobre a nova funcionalidade ocorre de maneira bem rápido? **Além disso, cria-se um código mais limpo, visto que o código para passar nos testes deve ser bastante simples.** Há mais segurança na correção de eventuais bugs; aumenta-se a produtividade, visto que se perde menos tempo com depuradores; e o código se torna mais flexível, menos acoplado e mais coeso.

Nós podemos afirmar que, em geral, utilizam-se testes unitários, testes de integração ou testes de aceitação – sendo os dois primeiros os mais comuns. **Algumas ferramentas que podem ser utilizadas para implementar o processo de desenvolvimento orientado a testes:** JUnit, TesteNG, PHPUnit, SimpleTest, NUnit, Jasmine, CUnit, PyUnit, etc. Pessoal, agora um detalhe que nós passamos direto sobre a abordagem de desenvolvimento...

O TEST DRIVEN DEVELOPMENT (TDD) NÃO É UMA ABORDAGEM PARA REALIZAR TESTES – TRATA-SE DE UMA ABORDAGEM PARA DESENVOLVER SOFTWARES. ELA PODE EVENTUALMENTE SER CONSIDERADA TAMBÉM UMA TÉCNICA DE PROGRAMAÇÃO!

Professor, uma curiosidade: isso já caiu em alguma prova discursiva? **Sim, galera... o enunciado dessa prova requisitava ao aluno informar as vantagens do emprego do TDD em relação a outras metodologias ágeis.** Poderíamos responder essa pergunta afirmando que o software desenvolvido, em geral, apresenta maior qualidade, na medida em que é implementado direcionado às expectativas do cliente.

Poderíamos dizer também que há a possibilidade de se testar todo o código desenvolvido, o que oferece maior confiabilidade ao sistema. Por fim, em geral, o código é mais modularizado,

flexível e extensível, visto que a metodologia requer que os desenvolvedores imaginem o software como pequenas unidades[1] que podem ser reescritas, desenvolvidas e testadas de forma independente e integradas em momento posterior.

Essa mesma prova perguntava também quais são os princípios da Metodologia Extreme Programming (XP) apoiados pelo TDD. Uma resposta adequada poderia afirmar que o XP apresenta diversas práticas que podem ser relacionadas com o TDD. *Qual é a mais óbvia?* **A mais óbvia é o *Test-First* (Teste Primeiro), ratificando a característica básica recomendada veementemente pelo desenvolvimento orientado a testes.**

O TDD pode apoiar esse princípio por fornecer detalhes para a realização dos testes de unidade e de funcionalidade, que são importantes e necessários. Ademais, o desenvolvimento orientado a testes apresenta relação intrínseca com a refatoração, tendo em vista que confere ao programador maior segurança para identificar e remover o código duplicado, e permite, assim, a melhoria contínua do programa.

Galera, nós temos também uma variação chamada Acceptance Test-Driven Development (ATDD). **Ele é método ágil de desenvolvimento de software que se baseia na comunicação entre clientes do negócio, desenvolvedores e testadores.** Ele se difere do TDD, na medida em que possui um foco maior na comunicação entre os colaboradores. São utilizados testes de aceitação a partir do ponto de vista dos usuários.

O ATDD é focado na captura de requisitos em testes de aceitação e os utiliza para guiar o desenvolvimento do sistema. Ele ajuda a assegurar que todos os membros do projeto entendam precisamente o que é necessário fazer e implementar, estabelecendo critérios a partir da perspectiva do usuário e criando exemplos concretos.

As equipes que experimentam ATDD normalmente concluem que apenas o ato de se definir testes de aceitação ao discutir requisitos resulta numa melhor compreensão destes requisitos. Os testes em ATDD nos forçam a chegar a um ponto de acordo concreto sobre o exato comportamento que se espera que o software deva ter. *Entenderam?*

A proposta do ATDD é favorecer uma colaboração e comunicação maior entre todos os envolvidos no desenvolvimento de um produto, o que resulta em um entendimento mais claro e refinado dos requisitos, possibilitando um acordo entre ambas as partes do que será desenvolvido durante uma iteração/sprint. **No final, o resultado estará alinhado às expectativas do cliente.**

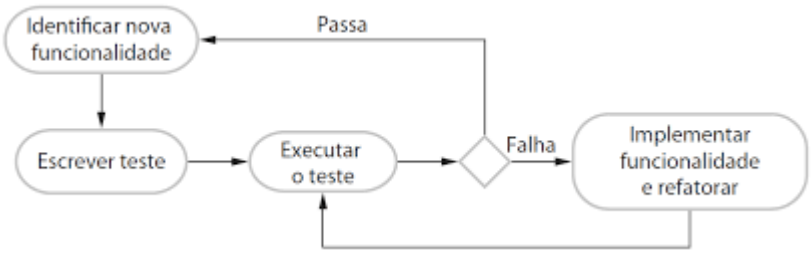
[1] Atenção: apesar de serem tipicamente realizados testes de unidade, não é obrigatória a utilização desse tipo de teste.

QUESTÕES COMENTADAS – DIVERSAS BANCAS

1. **(FGV / Senado Federal – 2022)** Você foi contratado para liderar uma equipe de DevOps. Um dos objetivos da sua liderança é aumentar a velocidade das entregas e a qualidade de novos recursos das aplicações utilizando o desenvolvimento orientado a testes. Assinale a opção que apresenta a ordem que descreve o ciclo de desenvolvimento orientado a testes.

   a) Refatorar - > Escrever um código funcional
   b) Escrever um caso de teste -> Refatorar
   c) Refatorar - > Escrever um código funcional - > Escrever um caso de teste
   d) Escrever um caso de teste -> Escrever um código funcional -> Refatorar
   e) Escrever um código funcional -> Escrever um caso de teste -> Refatorar

Comentários:



O ciclo de desenvolvimento orientado a testes segue o seguinte ciclo: (1) Identificar nova funcionalidade; (2) **Escrever um teste**; (3) Executar o teste; (4) **Implementar funcionalidade e refatorar**.

Gabarito: Letra D

---

2. **(CESPE / BANRISUL – 2022)** No processo de TDD, o código é desenvolvido em grandes blocos de requisitos do usuário. Cada iteração resulta em um novo teste, que faz parte um conjunto de testes de regressão executado no final do processo de integração.

Comentários:

Durante o TDD, o código é desenvolvido em incrementos **pequenos** e nenhum código é escrito enquanto não houver um teste para experimentá-lo. Cada iteração resulta em um ou mais novos testes, os quais são acrescentados a um conjunto de testes de regressão que são executados a cada mudança. Isso é feito para garantir que o novo código não tenha gerado efeitos colaterais que causem erros no código anterior.

Gabarito: Errado

---

3. **(CESPE / BANRISUL – 2022)** O TDD (Test-Driven Development) é uma metodologia que, ao longo do tempo, implica que o aplicativo em desenvolvimento tenha um conjunto abrangente de testes que ofereça confiança no que foi desenvolvido até então.

Comentários:

O TDD é um processo de desenvolvimento de software que incentiva os desenvolvedores a escrever testes antes de escrever código. A ideia por trás deste processo é que os desenvolvedores escrevam testes para cada pequena parte do código que eles escrevem, garantindo que o código funcione corretamente. Estes testes verificam se o código está funcionando como o esperado. O TDD também incentiva o uso de design incremental, onde pequenas partes do código são adicionadas gradualmente, permitindo que os problemas sejam detectados rapidamente e corrigidos. Ao longo do tempo, o TDD cria um conjunto abrangente de testes que fornecem confiança aos desenvolvedores de que o código é robusto e está funcionando como o esperado.

Gabarito: Correto

---

4. **(CESPE / BANRISUL – 2022)** O TDD (Test-Driven Development), como atividade da XP, é uma forma disciplinada de organizar o código, alterando-o de modo a aprimorar sua estrutura interna, sem que se altere o comportamento externo do software.

Comentários:

Essa não é a definição de TDD e, sim, Refactoring. O TDD é uma atividade do XP que envolve a escrita de testes antes mesmo de o código ser escrito, para que o desenvolvedor possa ter certeza de que o código que está escrevendo passará nos testes, e portanto que as funcionalidades desejadas estão sendo implementadas. Refatoração, por outro lado, é o processo de reescrever o código existente, para melhorar a sua qualidade, sem alterar a funcionalidade existente. Refatoração inclui a reestruturação do código para torná-lo mais limpo e legível, a remoção de código desnecessário, a simplificação da lógica usada e a correção de erros.

Gabarito: Errado

---

5. **(FGV / SEFAZ-MG – 2023)** Você entrou para um projeto novo, já em andamento, no qual a metodologia que a equipe do projeto segue é a de definir e escrever testes de software a partir das regras de negócio antes mesmo de implementar as funcionalidades propostas. Assinale a opção que indica o nome desse processo de desenvolvimento de software.

a) DDD
b) TDD
c) BDD
d) XP
e) Scrum

Comentários:

O TDD é um processo de desenvolvimento de software que incentiva a escrita de testes de software antes da implementação de qualquer código. O processo começa com a escrita de um teste para uma regra de negócio específica. Então, o código é escrito e, finalmente, os testes são executados para garantir que a regra de negócio esteja funcionando corretamente. O TDD é amplamente utilizado porque ajuda a garantir que o código seja testado de forma adequada, permitindo que qualquer problema seja detectado e corrigido antes que o código seja liberado para produção.

Gabarito: Letra B

---

6. **(FGV / Senado Federal – Análise de Sistemas – 2022)** Durante o processo de construção de software, a metodologia de Desenvolvimento Orientado a Testes é muito aplicada.

   A ordem utilizada na prática do TDD é:

   a) escrever a funcionalidade após escrever os testes unitários e, por fim, refatorar o código implementado.

   b) escrever os testes unitários após escrever a funcionalidade e, por fim, refatorar o código implementado.

   c) escrever a funcionalidade após refatorar o código implementado e por fim escrever os testes unitários.

   d) escrever os testes unitários após escrever a funcionalidade e, por fim, escrever os testes de integração.

   e) escrever os testes de integração após escrever a funcionalidade e, por fim, refatorar o código.

Comentários:

Essa é uma questão muito clássica! Basta saber a ordem correta: primeiro, escrevemos os testes unitários; depois, escrevemos a funcionalidade; por fim, refatoramos o código. Em outras palavras, a ordem correta é escrever a funcionalidade após escrever os testes unitários e, por fim, refatora o código implementado.

Gabarito: Letra A

---

7. **(CESPE / SERPRO – 2021)** Em TDD, os testes de um sistema devem ocorrer antes da implementação e ser oportunos, isolados e autoverificáveis.

Comentários:

Os testes devem seguir o modelo FIRST: F (Fast): devem ser rápidos, pois testam apenas uma unidade; I (Isolated): são isolados, testando individualmente as unidades e não sua integração; R (Repeateble): repetição nos testes, com resultados de comportamento constante; S (Self-verifying): autoverificação deve verificar se passou ou se deu como falha o teste; T (Timely): deve ser oportuno, sendo um teste por unidade.

Gabarito: Correto

---

8. **(CESPE / INMETRO – 2009)** A rotina diária dos desenvolvedores, ao empregar processos baseados no TDD (Test-Driven Development), é concentrada na elaboração de testes de homologação.

Comentários:

A rotina dos desenvolvedores é concentrada, na verdade, na elaboração de testes unitários e, não, de homologação. Lembrem-se: constrói o teste, constrói a funcionalidade e refatora a funcionalidade.

Gabarito: Errado

---

9. **(CESPE / INPI – 2013)** Usando-se o TDD, as funcionalidades devem estar completas e da forma como serão apresentadas aos seus usuários para que possam ser testadas e consideradas corretas.

Comentários:

ETAPA	DESCRIÇÃO
VERMELHO (RED)	Escreva um teste que falha: casos de teste são escritos sem que exista o recurso a ser testado, levando o teste a falhar;
VERDE (GREEN)	Escreva código para passar no teste: o recurso é implementado com o mínimo de código necessário para que passe no teste.
REFATORAR (REFACTOR)	Elimine redundâncias: o código que implementa o recurso é refinado e melhorado, sem que seja adicionada novas funcionalidades.

Pelo contrário, primeiro são feitos os testes e depois desenvolvem-se as funcionalidades.

Gabarito: Errado

---

10. **(CESPE / ANCINE – 2013)** No desenvolvimento de software conforme as diretivas do TDD (Test-Driven Development), deve-se elaborar primeiramente os testes e, em seguida, escrever o código necessário para passar pelos testes.

Comentários:

ETAPA	DESCRIÇÃO
VERMELHO (RED)	Escreva um teste que falha: casos de teste são escritos sem que exista o recurso a ser testado, levando o teste a falhar;
VERDE (GREEN)	Escreva código para passar no teste: o recurso é implementado com o mínimo de código necessário para que passe no teste.
REFATORAR (REFACTOR)	Elimine redundâncias: o código que implementa o recurso é refinado e melhorado, sem que seja adicionada novas funcionalidades.

Perfeito! Primeiro, criam-se os testes, depois cria-se o código.

Gabarito: Correto

---

11. (CESPE / INMETRO – 2009) Considerando uma organização na qual a abordagem de Test Driven Development (TDD) esteja implementada, assinale a opção correta.

a) Nessa organização, ocorre a execução de iterações com ciclo longo, isto é, com duração de alguns meses.

b) No início de cada iteração, a primeira atividade realizada pela equipe de desenvolvimento é produzir o código que será validado através de testes.

c) O refactoring é uma das primeiras atividades realizada no início de cada iteração.

d) Entre as atividades finais de cada iteração, o desenvolvedor escreve casos de teste automatizados, cuja execução verifica se houve a melhoria desejada ou se uma nova funcionalidade foi implementada.

e) Há coerência e inter-relação com os princípios promovidos pela prática da extreme programming (XP).

Comentários:

(a) Errado, são ciclos curtos e, não, longos; (b) Errado, são produzidos primeiramente os testes e, depois, os códigos; (c) Errado, a refatoração é a última atividade realizada em uma iteração; (d) Errado, escrever casos de testes é uma das atividades iniciais; (e) Correto, trata-se inclusive de uma das práticas recomendadas pelo XP.

Gabarito: Letra E

---

12. (CESPE / MPOG – 2013) Ao realizar o TDD (test-driven development), o programador é conduzido a pensar em decisões de design antes de pensar em código de implementação, o que

cria um maior acoplamento, uma vez que seu objetivo é pensar na lógica e nas responsabilidades de cada classe.

Comentários:

Em Engenharia de Software, há dois conceitos importantíssimos: coesão e acoplamento. Quando eu estudava, eu decorava uma frase pequena para entender: "*Coesão é a divisão de responsabilidades e Acoplamento é a dependência entre componentes*". Há outra que dizia assim: "*Uma boa arquitetura de software deve ter componentes de projeto com baixo acoplamento e alta coesão*".

O Acoplamento trata do nível de dependência entre módulos ou componentes de um software. *Por que é bom ter baixo acoplamento?* Porque se os módulos pouco dependem um do outro, modificações de um não afetam os outros, além de não prejudicar o reúso. Se esse princípio não for observado durante a construção da arquitetura de um sistema de software, pode haver problemas sérios de manutenção futura!

Voltando à questão: se o programador pensa em decisões de design antes de pensar em código de implementação, isso diminui o acoplamento - os componentes ficam menos dependentes, tornando a arquitetura bem mais flexível.

Gabarito: Errado

---

13. (CESPE / MPU – 2013) Na metodologia TDD, ou desenvolvimento orientado a testes, cada nova funcionalidade inicia com a criação de um teste, cujo planejamento permite a identificação dos itens e funcionalidades que deverão ser testados, quem são os responsáveis e quais os riscos envolvidos.

Comentários:

Perfeito! Essa metodologia permite um aprendizado maior sobre o problema a ser resolvido, permitindo (não obrigatoriamente) a identificação de itens, funcionalidades, responsáveis e riscos.

Gabarito: Correto

---

14. (CESPE / STF – 2013) No TDD, o primeiro passo do desenvolvedor é criar o teste, denominado teste falho, que retornará um erro, para, posteriormente, desenvolver o código e aprimorar a codificação do sistema.

Comentários:

ETAPA	DESCRIÇÃO

VERMELHO (RED)	Escreva um teste que falha: casos de teste são escritos sem que exista o recurso a ser testado, levando o teste a falhar;
VERDE (GREEN)	Escreva código para passar no teste: o recurso é implementado com o mínimo de código necessário para que passe no teste.
REFATORAR (REFACTOR)	Elimine redundâncias: o código que implementa o recurso é refinado e melhorado, sem que seja adicionada novas funcionalidades.

Perfeito! Cria-se o teste falho, desenvolve-se um código e só então ocorre a refatoração.

Gabarito: Correto

---

15. (CESPE / TRT17 – 2013) TDD consiste em uma técnica de desenvolvimento de software com abordagem embasada em perspectiva evolutiva de seu desenvolvimento. Essa abordagem envolve a produção de versões iniciais de um sistema a partir das quais é possível realizar verificações de suas qualidades antes que ele seja construído.

Comentários:

A questão diz que versões iniciais são produzidas e, a partir dessas versões, é possível realizar verificações de suas qualidades antes que ele seja construído. Na verdade, testes são criados inicialmente para verificar sua qualidade e, a partir daí, versões são produzidas. Logo, a questão inverteu os conceitos!

Gabarito: Correto

---

16. (CESPE / AL-RN – 2013) Um típico ciclo de vida de um projeto em TDD consiste em:

I. Executar os testes novamente e garantir que estes continuem tendo sucesso.
II. Executar os testes para ver se todos estes testes obtiveram êxito.
III. Escrever a aplicação a ser testada.
IV. Refatorar (refactoring).
V. Executar todos os possíveis testes e ver a aplicação falhar.
VI. Criar o teste.

A ordem correta e cronológica que deve ser seguida para o ciclo de vida do TDD está expressa em:

a) IV – III – II – V – I – VI.
b) V – VI – II – I – III – IV.
c) VI – V – III – II – IV – I.
d) III – IV – V – VI – I – II.
e) III – IV – VI – V – I – II.

Comentários:

A ordem correta é: (VI) Criar o teste; (V) Executar todos os possíveis testes e ver a aplicação falhar; (III) Escrever a aplicação a ser testada; (II) Executar os testes para ver se todos estes testes obtiveram êxito; (IV) Refatorar (refactoring); (I) Executar os testes novamente e garantir que estes continuem tendo sucesso.

Gabarito: Letra C

---

17. (FGV / ALMT – 2013) Com relação ao desenvolvimento orientado (dirigido) a testes (do Inglês Test Driven Development – TDD), analise as afirmativas a seguir.

I. TDD é uma técnica de desenvolvimento de software iterativa e incremental.

II. TDD implica escrever o código de teste antes do código de produção, um teste de cada vez, tendo certeza de que o teste falha antes de escrever o código que irá fazê-lo passar.

III. TDD é uma técnica específica do processo XP (Extreme Programming), portanto, só pode ser utilizada em modelos de processos ágeis de desenvolvimento de software.

Assinale:

a) Se somente as afirmativas I e II estiverem corretas.
b) Se somente as afirmativas I e III estiverem corretas.
c) Se somente as afirmativas II e III estiverem corretas.
d) Se somente a afirmativa III estiver correta.
e) Se somente a afirmativa I estiver correta.

Comentários:

(I) Correto, é iterativo e incremental (lembrando que, em sua imensa maioria, metodologias ágeis são iterativas/incrementais); (II) Correto, escrevem-se os testes antes, fá-lo falhar e só depois escreve-se o código da aplicação; (III) Errado, ele é uma abordagem independente que pode ser utilizado com metodologias ágeis ou tradicionais.

Gabarito: Letra A

---

18. (FCC / TRT-MG – 2015) Um analista de TI está participando do desenvolvimento de um software orientado a objetos utilizando a plataforma Java. Na abordagem de desenvolvimento adotada, o código é desenvolvido de forma incremental, em conjunto com o teste para esse incremento, de forma que só se passa para o próximo incremento quando o atual passar no teste. Como o código é desenvolvido em incrementos muito pequenos e são executados testes a cada vez que uma funcionalidade é adicionada ou que o programa é refatorado, foi necessário definir um

ambiente de testes automatizados utilizando um framework popular que suporta o teste de programas Java.

A abordagem de desenvolvimento adotada e o framework de suporte à criação de testes automatizados são, respectivamente,

a) Behavior-Driven Development e JTest.
b) Extreme Programming e Selenium.
c) Test-Driven Development e Jenkins.
d) Data-Driven Development and Test e JUnit.
e) Test-Driven Development e JUnit.

Comentários:

A abordagem claramente é o TDD e o framework evidentemente é o JUnit (ferramenta de suporte à criação de testes unitários automatizados). Lembrando que: DDD é um modelo de programação em que os próprios dados controlam o fluxo do programa e não a lógica do programa; XP é uma metodologia ágil de gerenciamento de projetos que suporta lançamentos frequentes em curtos ciclos de desenvolvimento para melhorar a qualidade do software e permitir que os desenvolvedores respondam às mudanças nos requisitos dos clientes; BDD é um método de desenvolvimento ágil que encoraja a colaboração entre desenvolvedores; Selenium é uma ferramenta usada para testes funcionais em aplicações web; e JTest é um ramework de análise estática que usa a linguagem Java.

Gabarito: Letra E

---

19. (CESPE / TRE-PE – 2017) O desenvolvimento orientado a testes (TDD):

a) é um conjunto de técnicas que se associam ao XP (extreme programming) para o desenvolvimento incremental do código que se inicia com os testes.

b) agrega um conjunto de testes de integração para avaliar a interconexão dos componentes do software com as aplicações a ele relacionadas.

c) avalia o desempenho do desenvolvimento de sistemas verificando se o volume de acessos/transações está acima da média esperada.

d) averigua se o sistema atende aos requisitos de desempenho verificando se o volume de acessos/transações mantém-se dentro do esperado.

e) testa o sistema para verificar se ele foi desenvolvido conforme os padrões e a metodologia estabelecidos nos requisitos do projeto.

Comentários:

O único item que faz algum sentido em relação ao TDD é o primeiro, isto é, conjunto de técnicas intimamente ligadas ao XP para o desenvolvimento de software que se inicia com os testes.

Gabarito: Correto

---

20. (CESPE / STM – 2018) O TDD (test driven development) parte de um caso de teste que caracteriza uma melhoria desejada ou nova funcionalidade a ser desenvolvida, de modo a confirmar o comportamento correto e possibilitar a evolução ou refatoração do código.

Comentários:

TDD é um método para construir software que enfatiza a criação de testes antes da criação do código-fonte. Logo, faz-se o teste - se não passou, refatora!

Gabarito: Correto

---

21. (CESPE / TRE/PI – 2016) O TDD (test driven development):

a) apresenta como vantagem a leitura das regras de negócio a partir dos testes, e, como desvantagem, a necessidade de mais linhas de códigos que a abordagem tradicional, o que gera um código adicional.

b) impede que seja aplicada a prática de programação em pares, que é substituída pela interação entre analista de teste, testador e programador.

c) é um conjunto de técnicas associadas ao eXtremme Programing e a métodos ágeis, sendo, contudo, incompatível com o Refactoring, haja vista o teste ser escrito antes da codificação.

d) refere-se a uma técnica de programação cujo principal objetivo é escrever um código funcional limpo, a partir de um teste que tenha falhado.

e) refere-se a uma metodologia de testes em que se devem testar condições, loops e operações; no entanto, por questão de simplicidade, não devem ser testados polimorfismos.

Comentários:

(a) Errado, esse item não faz nenhum sentido em relação ao TDD; (b) Errado, não impede a programação em pares; (c) Errado, ela é totalmente compatível e dependente da refatoração; (d) Correto, ela pode ser vista como uma técnica de programação cujo objetivo é escrever um código funcional limpo a partir de um teste falho; (e) Errado, não se trata de uma metodologia de testes.

Gabarito: Letra D

---

22. (UFRRJ / UFRRJ – 2015) Os testes de unidade têm papel central na metodologia de implementação dirigida por testes, popularizada pelo processo XP e adotada em outros métodos. Esses testes são criados primeiro, exercitando o contrato de cada operação implementada pelos métodos. Em seguida, o código dos métodos é escrito para cumprir os contratos e, portanto, passar nos testes de unidade. Esse cenário corresponde à abordagem

a) TDD.
b) MDD.
c) DDC.
d) MDE.
e) FDD.

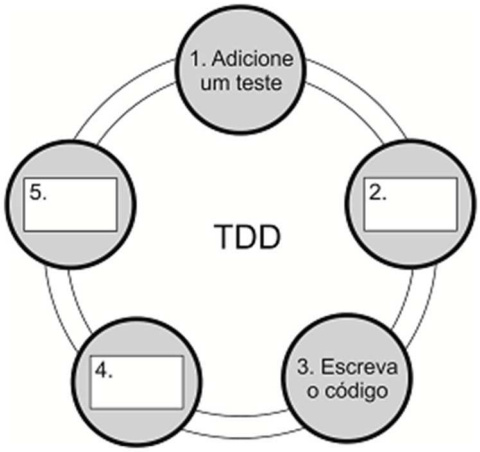
Comentários:

Testes de unidade têm papel central? Metodologia dirigida a testes? Popularizada pelo XP? Testes são criados primeiro? Tudo isso nos remente a... TDD!

Gabarito: Letra A

---

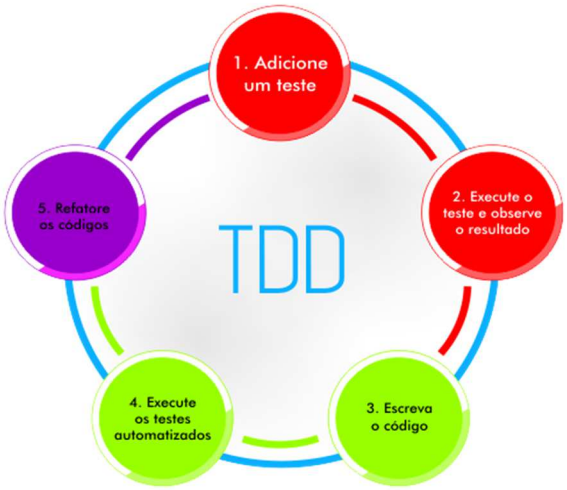
23. (FCC / TRE-PR – 2017) Considere o ciclo do Test-Driven Development – TDD.



A Caixa:

a) 2. corresponde a "Execute os testes automatizados".
b) 4. corresponde a "Refatore o código".
c) 5. corresponde a "Execute os testes novamente e observe os resultados".
d) 4. corresponde a "Execute os testes automatizados".
e) 5. corresponde a "Faça todos os testes passarem".

Comentários:



(a) Errado, corresponde a "Execute o teste e observe o resultado"; (b) Errado, corresponde a "Execute os testes automatizados"; (c) Errado, corresponde a "Refatore os códigos"; (d) Correto, corresponde realmente a "Execute os testes automatizados"; (e) Errado, corresponde a "Refatore os códigos".

Gabarito: Letra D

---

24. (IESES / TRE-MA – 2015) A respeito da técnica de testes TDD é correto afirmar que:

a) Testa o software com base no comportamento esperado.
b) É uma prática para desenvolvimento de testes unitário que pode utilizar o processo Red-Green-Refactor.
c) Utiliza-se da estrutura Dado, Quando e Então para montar os testes.
d) Prega que os testes devem ser realizados sempre após a implementação ser concluída.

Comentários:

(a) Errado, não vejo nada de errado nesse item – ele pode testar o software com base no comportamento esperado! No entanto, a banca considerou o item errado; (b) Correto, ele realmente utiliza o processo RED/GREEN/REFACTOR; (c) Errado, esse item não faz qualquer sentido; (d) Errado, testes são realizados antes de a implementação ser concluída.

Gabarito: Letra B

---

25. (CESPE / TER-RS – 2015) Projeto para o desenvolvimento de software que utilize TDD deve:

a) realizar sprints a cada quinzena.
b) desenvolver pequenos releases.
c) apresentar grande quantidade de testes unitários de código-fonte previamente desenvolvidos.

d) apresentar linguagem de programação estruturada.
e) recomendar a preparação dos testes para que, posteriormente, seja desenvolvido o código.

Comentários:

(a) Errado, não há nenhuma relação entre TDD e Sprints; (b) Errado, são desenvolvidas pequenas unidades e, não, releases; (c) Errado, os testes unitários são escritos iterativamente a medida que o desenvolvimento ocorre; (d) Errado, não há nenhuma relação ou dependência com linguagens de programação; (e) Correto, recomenda-se preparar testes antes do desenvolvimento do código.

Gabarito: Letra E

---

26. (FCC / TRE-AP – 2015) O TDD – Test Driven Development (Desenvolvimento orientado a teste):

a) é parte das metodologias ágeis UP – Unified Process e XP – Extreme Programming, tendo sido criado para ser usado em metodologias que respeitam os 4 princípios do Manifesto Ágil.

b) transforma o desenvolvimento, pois deve-se primeiro implementar o sistema antes de escrever os testes. Os testes são utilizados para facilitar no entendimento do projeto e para clarear o que se deseja em relação ao código.

c) baseia-se em um ciclo simples: escreve-se um código -> cria-se um teste para passar no código -> refatora-se.

d) propõe a criação de testes que validem o código como um todo para reduzir o tempo de desenvolvimento.

e) beneficia-se de testes que seguem o modelo FIRST: F (Fast) I (Isolated) R (Repeatable) S (Self-verifying) T (Timely).

Comentários:

ETAPA	DESCRIÇÃO
VERMELHO (RED)	Escreva um teste que falha: casos de teste são escritos sem que exista o recurso a ser testado, levando o teste a falhar;
VERDE (GREEN)	Escreva código para passar no teste: o recurso é implementado com o mínimo de código necessário para que passe no teste.
REFATORAR (REFACTOR)	Elimine redundâncias: o código que implementa o recurso é refinado e melhorado, sem que seja adicionada novas funcionalidades.

(a) Errado, Unified Process (UP) não é uma metodologia ágil; (b) Errado, devem ser implementados os testes antes do sistema; (c) Errado, cria-se um teste falho > escreve-se o código que passa > refatora-se; (d) Errado, criam-se testes que validam unidades e, não, o código como um todo; (e) Correto, vamos falar um pouquinho sobre isso agora:

Existe uma representação chamada Modelo FIRST: **F (Fast):** devem ser rápidos, pois testam apenas uma unidade; **I (Isolated):** testes unitários são isolados, testando individualmente as unidades e não sua integração; **R (Repeateble):** repetição nos testes, com resultados de comportamento constante; **S (Self-verifying):** a auto verificação deve verificar se passou ou se deu como falha o teste; **T (Timely):** o teste deve ser oportuno, sendo um teste por unidade.

Gabarito: Letra E

---

27. (CESPE / TRE-TO – 2017) O TDD (Test-Driven Development), que vem sendo adotado para testar os projetos de software,

a) utiliza os testes de caixa preta antes da entrega do software.
b) agiliza os testes por amostragem sem compatibilidade retroativa.
c) cobre amplamente os testes unitários.
d) escreve o teste antes da codificação do software.
e) realiza refactoring antes de escrever a aplicação a ser testada.

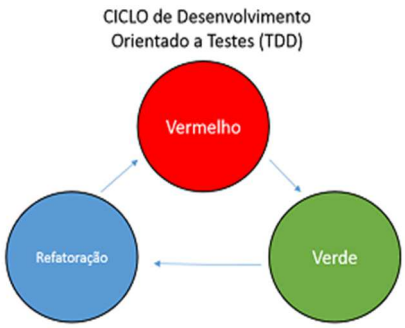
Comentários:

Eu já começo discordando do enunciado – ele vem sendo utilizado para desenvolver software utilizando testes, mas não para testar projetos de software. Ignorando a redação da questão, vamos aos comentários: (a) Errado, ele tipicamente utiliza testes de unidade antes da entrega do software; (b) Errado, esse item não faz qualquer sentido; (c) Errado, ele não tem o intuito principal de cobrir amplamente testes unitários; (d) Correto, o teste é escrito antes da codificação do software; (e) Errado, não faz sentido fazer *refactoring* antes de escrever a aplicação.

Gabarito: Letra D

---

28. (FGV / IBGE – 2016) O Desenvolvimento Orientado a Testes (TDD) é um método de desenvolvimento criado e disseminado por Kent Beck em seu livro "Test-driven development". O método define regras, boas práticas e um ciclo de tarefas com 3 etapas: a etapa vermelha, a etapa verde e a etapa de refatoração, ilustrado na imagem abaixo:



Com relação às regras e boas práticas de TDD e ao seu ciclo, é correto afirmar que:

a) pode-se escrever testes que não compilam na etapa vermelha;
b) na etapa verde deve-se escrever código que testa uma funcionalidade a fundo de forma criteriosa e detalhada;
c) código novo só é escrito se um teste automatizado passar;
d) a duplicação é tolerada na etapa de refatoração;
e) é uma boa prática de TDD iniciar o desenvolvimento do código de uma funcionalidade e, logo em seguida, testá-la.

Comentários:

ETAPA	DESCRIÇÃO
VERMELHO (RED)	Escreva um teste que falha: casos de teste são escritos sem que exista o recurso a ser testado, levando o teste a falhar;
VERDE (GREEN)	Escreva código para passar no teste: o recurso é implementado com o mínimo de código necessário para que passe no teste.
REFATORAR (REFACTOR)	Elimine redundâncias: o código que implementa o recurso é refinado e melhorado, sem que seja adicionada novas funcionalidades.

(a) Correto. Nessa etapa, o objetivo do desenvolvedor é escrever um pequeno teste que não funcione e que talvez nem mesmo compile inicialmente; (b) Errado. Nessa etapa, escreve-se o código que apenas passa no teste, sem necessidade de aprofundamento em detalhes de implementação; (c) Errado. O código novo é escrito para o teste automatizado passar; (d) Errado. Não é tolerada a duplicação, uma vez que o objetivo é eliminar redundâncias e melhorar o código; (e) Errado. Vimos centenas de vezes que o primeiro passo é criar o teste e depois escrever.

Gabarito: Letra A

---

29. (IADES / EBSERH – 2013) Assinale a alternativa que não corresponde a uma das fases do processo de desenvolvimento, dirigido a testes (TDD).

a) Executar o teste, com os outros testes implementados, que rodarão e fornecerão o resultado de que o software está sem problemas.
b) Escrever o teste para a funcionalidade e implementação.
c) Realizar a identificação do incremento de funcionalidade.
d) Implementar a funcionalidade e executar novamente o teste.
e) Implementar a próxima parte da funcionalidade, após todos os testes terem sido executados, com sucesso

Comentários:

(a) Errado. Na terceira fase, executa-se realmente o teste junto com todos os outros testes implementados. No entanto, inicialmente você não terá implementado a funcionalidade, logo o

novo teste falhará; (b) Correto, essa é a segunda fase; (c) Correto, essa é a primeira fase; (d) Correto, essa é a quarta fase; (e) Correto, essa é a quinta fase.

Gabarito: Letra A

---

30. (PR-4 / UFRJ – 2018) O ciclo do TDD - Test Driven Development, ou, em português, Desenvolvimento Guiado por Testes consiste em:

a) implementar teste unitário falho, tornar o teste bem-sucedido e refatorar.
b) implementar a funcionalidade, executar teste unitário e refatorar.
c) implementar teste unitário falho, refatorar e tornar o teste bem-sucedido.
d) implementar a funcionalidade, refatorar e tornar o teste bem-sucedido.
e) refatorar, executar teste unitário e implementar a funcionalidade.

Comentários:

(a) Correto, as atividades são exatamente essas e nessa ordem; (b) Errado, o teste vem antes da implementação da funcionalidade; (c) Errado, primeiro você torna o teste bem sucedido e depois refatora; (d) Errado, não faz sentido refatorar antes do teste; (e) Errado, não faz sentido refatorar antes do teste.

Gabarito: Letra A

---

31. (CESPE / STJ – 2015) Um dos passos executados no ciclo de atividades do processo TDD é a criação de novos testes para as falhas encontradas no código original, sem alteração deste.

Comentários:

Nooooope... testes devem realmente encontrar falhas – o que se altera é o código para passar no teste e, não, o teste em si.

Gabarito: Errado

---

32. (CS-UFG / AL-GO – 2015) O desenvolvimento dirigido a testes (TDD, do Inglês Test-Driven Development) é uma abordagem de desenvolvimento de software na qual se intercalam testes e desenvolvimento de código. Uma das características da abordagem TDD é:

a) a sua utilidade no desenvolvimento de softwares novos.
b) o maior custo associado aos testes de regressão.
c) a redução da importância da automatização dos testes.
d) a sua adequação a processos de software sequenciais.

Comentários:

(a) Correto. Ela é mais ideal para o desenvolvimento de softwares novos; (b) Errado, ela reduz custos de testes de regressão; (c) Errado, ela aumenta a importância da automatização de testes; (d) Errado, ela é adequada a processos iterativos – lembrem-se do ciclo de testes, falhas e refatorações.

Gabarito: Letra A

---

33. (UECE-CEV / FUNCEME – 2018) Test-driven Development (TDD) é uma abordagem para o desenvolvimento de programas em que se intercalam testes e desenvolvimento de código
(Sommerville, I. Engenharia de Software, 9a edição, 2011).

A respeito do TDD, é correto afirmar que:

a) consiste em um processo iterativo que se inicia escrevendo um código de uma funcionalidade do sistema e, logo em seguida, testa-o para saber se a implementação foi correta.
b) apesar de útil, não diminui o custo de testes de regressão do sistema.
c) sua utilização elimina a necessidade de testes de validação do sistema, uma vez que ele já foi testado incrementalmente.
d) apesar de ter sido apresentado como parte dos métodos ágeis, também pode ser usado em outros processos de desenvolvimento de software.

Comentários:

(a) Errado, inicia-se escrevendo o teste; (b) Errado, diminui – sim – os custos de testes de regressão; (c) Errado, não elimina em nenhuma hipótese testes de validação/aceitação; (d) Correto, ele pode ser utilizado com métodos ágeis ou tradicionais.

Gabarito: Letra D

---

34. (FAUGRS / UFRGS – 2018) _____ é uma abordagem para o desenvolvimento de programas em que se intercalam testes e desenvolvimento de código. Essencialmente, desenvolve-se um código de forma incremental em conjunto com um teste para este incremento. Não se avança para o próximo incremento até que o código desenvolvido passe no teste. Essa abordagem foi introduzida como parte de métodos ágeis, mas pode ser também usada em processos de desenvolvimento dirigido a planos. Assinale a alternativa que preenche corretamente a lacuna do texto acima.

a) Desenvolvimento Guiado por Testes (TDD)
b) Desenvolvimento em Espiral
c) Engenharia Dirigida a Modelos (MDD)
d) Rational Unified Process (RUP)
e) Teste de Sistema

Comentários:

TDD é uma **abordagem para o desenvolvimento** de programas em que se **intercalam testes e desenvolvimento de código**. Essencialmente, desenvolve-se um código de forma incremental em conjunto com um teste para este incremento. Não se avança para o próximo incremento até que o código desenvolvido passe no teste. Essa abordagem foi introduzida como parte de métodos ágeis, mas pode ser também usada em processos de desenvolvimento dirigido a planos.

Gabarito: Letra A

---

35. **(VUNESP / TCE-SP – 2015)** No Desenvolvimento Orientado a Testes (TDD), os casos de teste que definem o recurso a ser implementado devem ser elaborados:

a) assim que o código do teste estiver pronto.
b) antes de o código do recurso ser desenvolvido.
c) após o código do recurso ter sido completamente documentado.
d) simultaneamente com o desenvolvimento do código do recurso.
e) somente se o código do recurso apresentar erros.

Comentários:

(a) Errado, os casos de testes já devem estar prontos antes do código do teste ser feito; (b) Correto, devem ser elaborados antes de o código do recurso ser desenvolvido; (c) Errado, não existe o conceito de documentação formal, os próprios testes agem como uma forma de documentação que descreve o que o código deve fazer; (d) Errado, as etapas de desenvolvimento do código do recurso e a implementação dos casos de testes ocorrem em etapas distintas e não concomitantes; (e) Errado, casos de testes sempre são implementados porque eles representam a funcionalidade a ser desenvolvida.

Gabarito: Letra B

---

36. **(IBFC / EMBASA – 2017)** No Ciclo de Desenvolvimento do TDD (Test-Driven Development), utiliza-se a estratégia que aplica três palavras-chaves (em inglês), que é denominada:

a) Red, Green, Refactor
b) White, Gray, Black
c) White, Black, Refactor
d) Green, Yellow, Red

Comentários:

ETAPA	DESCRIÇÃO
VERMELHO (RED)	Escreva um teste que falha: casos de teste são escritos sem que exista o recurso a ser testado, levando o teste a falhar;

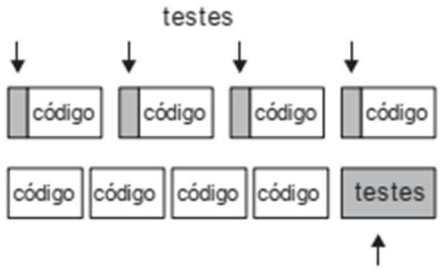
VERDE (GREEN)	Escreva código para passar no teste: o recurso é implementado com o mínimo de código necessário para que passe no teste.
REFATORAR (REFACTOR)	Elimine redundâncias: o código que implementa o recurso é refinado e melhorado, sem que seja adicionada novas funcionalidades.

Trata-se do RED > GREEN > REFACTOR.

Gabarito: Letra A

---

37. (FCC / CREMESP – 2016) Considere a figura abaixo que apresenta duas abordagens de teste.



A figura:

a) ilustra as duas fases do TDD, que correspondem a escrever pequenos testes e testá-los no final.
b) mostra o ciclo conhecido como Vermelho-Verde-Refatora.
c) apresenta a diferença entre testes automatizados e testes manuais no XP.
d) mostra que um desenvolvedor que pratica TDD tem mais feedbacks do que um que escreve testes ao final.
e) evidencia que TDD é impraticável, pois o desenvolvedor gasta muito tempo escrevendo código de testes.

Comentários:

Qual é a diferença entre a abordagem superior e inferior? Na parte superior, testes são aplicados individualmente em cada unidade de código. Na parte inferior, todos os códigos foram escritos e, somente depois, foram testados. Logo, existem mais feedbacks na abordagem superior (TDD) do que na abordagem inferior.

Gabarito: Letra D

---

38. (CESPE / ANATEL – 2014) Em se tratando de desenvolvimento de softwares dirigidos a testes (TDD), a execução dos testes é realizada antes da implementação da funcionalidade.

Comentários:

Perfeito! A execução dos testes de realmente ocorrem antes da implementação da funcionalidade.

Gabarito: Correto

---

39. (CESPE / TC-DF – 2014) No TDD, o refatoramento do código deve ser realizado antes de se escrever a aplicação que deve ser testada.

Comentários:

Opaaaaa... como você vai refatorar um código que ainda não foi escrito? Não faz sentido!

Gabarito: Errado

---

40. (CESPE / STJ – 2015) No método de desenvolvimento TDD (Test Driven Development), o desenvolvedor escreve primeiro um caso de teste e, posteriormente, o código.

Comentários:

Perfeito! Esse é o conceito do *test-first*, isto é, o desenvolvedor escreve primeiro um caso de teste e, posteriormente, o código.

Gabarito: Correto

---

41. (CESPE / MPE-PI – 2018) O TDD possibilita o desenvolvimento de softwares fundamentado em testes. O ciclo de desenvolvimento do TDD segue os seguintes passos:

- escrever um teste que inicialmente não passa;
- adicionar uma nova funcionalidade do sistema;
- fazer o teste passar;
- realizar a integração contínua do código;
- escrever o próximo teste.

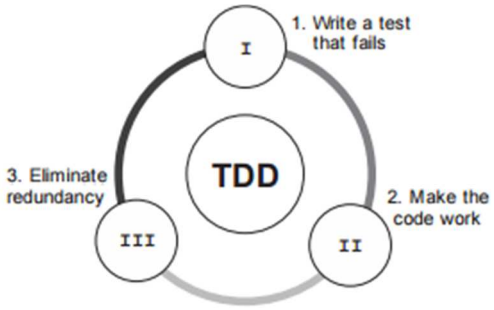
Comentários:

A ordem realmente está correta, no entanto há uma pegadinha: na quarta etapa, realiza-se a refatoração do código e, não, a integração contínua.

Gabarito: Errado

---

42. (FCC / Prefeitura de Teresina-PI – 2016) O Test Driven Development – TDD é uma das práticas sugeridas na eXtreme Programming – XP, onde o programador escreve o teste antes de escrever o código. O ciclo de desenvolvimento utilizando TDD é mostrado abaixo.



Considere:

I. Etapa inicial, onde se escreve um teste que falha, para alguma funcionalidade que ainda será Escrita.
II. Já com o teste criado, é o momento de executar o teste.
III. Eliminar códigos redundantes, remover acoplamentos, enfim, identificar pontos de melhoria no código.

As etapas I, II e III são, respectivamente,

a) Iniciação, Execução e Controle.
b) Red, Green e Refactor.
c) Iniciação, Atuação e Otimização.
d) Plan, Do e Check.
e) Planejamento, Execução e Melhoria.

Comentários:

ETAPA	DESCRIÇÃO
VERMELHO (RED)	Escreva um teste que falha: casos de teste são escritos sem que exista o recurso a ser testado, levando o teste a falhar;
VERDE (GREEN)	Escreva código para passar no teste: o recurso é implementado com o mínimo de código necessário para que passe no teste.
REFATORAR (REFACTOR)	Elimine redundâncias: o código que implementa o recurso é refinado e melhorado, sem que seja adicionada novas funcionalidades.

(RED) Etapa inicial, onde se escreve um teste que falha, para alguma funcionalidade que ainda será Escrita; (GREEN) Já com o teste criado, é o momento de executar o teste; (REFACTOR) Eliminar códigos redundantes, remover acoplamentos, enfim, identificar pontos de melhoria no código.

Gabarito: Letra B

---

43. **(FAUGRS / BANRISUL – 2018)** Considere as ações abaixo, executadas em desenvolvimento orientado a testes, Test-Driven Design (TDD).

I - Escrever código de teste.

II - Verificar se o teste falha.
III - Escrever código de produção.
IV - Executar teste até passar (reescrevendo o código de produção, se for necessário, até que o teste passe).
V - Refatorar código de produção e/ou de teste para melhorá-lo.

Considerando que se deseja incluir um novo caso de teste, assinale a alternativa que apresenta a sequência de ações que devem obrigatoriamente ocorrer para essa inclusão, segundo o TDD.

a) I, III e IV.
b) III, I e IV.
c) I, II, III e IV.
d) I, III, IV e V.
e) I, II, III, IV e V.

Comentários:

A ordem correta é: (I) Escrever código de teste; (II) Verificar se o teste falha; (III) Escrever código de produção; (IV) Executar teste até passar (reescrevendo o código de produção, se for necessário, até que o teste passe); (V) Errado, essa não é uma ação obrigatória – como apresenta o enunciado.

Gabarito: Letra C

---

44. (FGV / IBGE – 2017) Test Driven Development (TDD) é uma prática muito utilizada no processo de desenvolvimento de sistemas computacionais. Analise as afirmativas a seguir sobre o uso da prática de TDD:

I. Tornam os testes de regressão mais demorados porque o desenvolvedor precisará fazer testes manuais várias vezes por dia.

II. Garante que os requisitos do sistema sejam atendidos porque o desenvolvedor escreverá o código de testes sempre que acabar a implementação do código do sistema.

III. Ajuda o desenvolvedor a escrever código de qualidade porque ele gastará parte do seu tempo escrevendo código de testes.

Está correto o que se afirma em:

a) somente I;
b) somente II;
c) somente III;
d) somente II e III;
e) I, II e III.

Comentários:

(I) Errado. De acordo com Ian Sommerville, um dos benefícios mais importantes de desenvolvimento dirigido a testes é que ele reduz os custos dos testes de regressão, uma vez que testes automatizados – fundamentais para o desenvolvimento *test-first* – reduzem drasticamente os custos com testes de regressão; (II) Errado. O desenvolver escreverá o código de testes antes de acabar a implementação do código do sistema; (III) Correto. O TDD realmente ajuda o desenvolvedor a escrever código de qualidade porque ele gastará parte do seu tempo escrevendo código de testes.

Gabarito: Letra C

---

45. (CESPE / ANATEL – 2014) Na atividade de TDD (test-driven development), a escrita de teste primeiro define implicitamente tanto uma interface quanto uma especificação do comportamento para a funcionalidade que está sendo desenvolvida, estando, entretanto, a viabilidade do uso dessa abordagem limitada aos processos de desenvolvimento de software que seguem as práticas ágeis.

Comentários:

Opa... a interface deve ser definida explicitamente! Além disso, conforme afirma Ian Sommerville, ele também pode ser utilizado em processos de desenvolvimento dirigido a planos (tradicionais) e, não só, aos ágeis.

Gabarito: Errado

---

46. (CESPE / TRE-MT – 2015) Considere as seguintes etapas de um processo do tipo desenvolvimento orientado a testes (TDD).

I Implementar funcionalidade e refatorar.
II Identificar nova funcionalidade.
III Executar o teste.
IV Escrever o teste.
V Implementar a próxima parte da funcionalidade.

Assinale a opção que apresenta a sequência correta em que essas etapas devem ser realizadas.

a) I; IV; III; II; V
b) IV; III; II; I; V
c) I; IV; II; III; V
d) II; IV; III; I; V
e) IV; II; III; I; V

Comentários:

A ordem correta é: (II) Identificar nova funcionalidade; (IV) Escrever o teste; (III) Executar o teste; (I) Implementar funcionalidade e refatorar; (V) Implementar a próxima parte da funcionalidade.

Gabarito: Letra D

---

47. **(FCC / SEFAZ-SC – 2018)** O Test-Driven Development (TDD) é uma abordagem para o desenvolvimento de programas em que se intercalam testes e desenvolvimento de código. As etapas do processo fundamental de TDD são mostradas abaixo em ordem alfabética:

I. Escrever um teste para a funcionalidade identificada e implementá-lo como um teste automatizado.

II. Executar o teste, junto com os demais testes já implementados, sem implementar a nova funcionalidade no código.

III. Identificar e implementar uma outra funcionalidade, após todos os testes serem executados com sucesso.

IV. Identificar uma nova funcionalidade pequena para ser incrementada com poucas linhas em um código.

V. Implementar a nova funcionalidade no código e reexecutar o teste.

VI. Refatorar o código com melhorias incrementais até que o teste execute sem erros.

VII. Revisar a funcionalidade e o teste, caso o código execute sem falhar.

Considerando o item IV a primeira etapa e o item III a última etapa, a sequência intermediária correta das etapas do processo é:

a) I – II – VII – V e VI.
b) I – V – II – VII e VI.
c) I – VI – V – VII e II.
d) V – I – II – VII e VI.
e) V – I – VI – VII e II.

Comentários:

São somente cinco etapas: I. Escrever um teste para a funcionalidade identificada e implementá-lo como um teste automatizado; II. Executar o teste, junto com os demais testes já implementados,

sem implementar a nova funcionalidade no código; VII. Revisar a funcionalidade e o teste, caso o código execute sem falhar; V. Implementar a nova funcionalidade no código e reexecutar o teste; VI. Refatorar o código com melhorias incrementais até que o teste execute sem erros.

Gabarito: Letra A

LISTA DE QUESTÕES – DIVERSAS BANCAS

1. **(FGV / Senado Federal – 2022)** Você foi contratado para liderar uma equipe de DevOps. Um dos objetivos da sua liderança é aumentar a velocidade das entregas e a qualidade de novos recursos das aplicações utilizando o desenvolvimento orientado a testes. Assinale a opção que apresenta a ordem que descreve o ciclo de desenvolvimento orientado a testes.

   a) Refatorar - > Escrever um código funcional
   b) Escrever um caso de teste -> Refatorar
   c) Refatorar - > Escrever um código funcional - > Escrever um caso de teste
   d) Escrever um caso de teste -> Escrever um código funcional -> Refatorar
   e) Escrever um código funcional -> Escrever um caso de teste -> Refatorar

2. **(CESPE / BANRISUL – 2022)** No processo de TDD, o código é desenvolvido em grandes blocos de requisitos do usuário. Cada iteração resulta em um novo teste, que faz parte um conjunto de testes de regressão executado no final do processo de integração.

3. **(CESPE / BANRISUL – 2022)** O TDD (Test-Driven Development) é uma metodologia que, ao longo do tempo, implica que o aplicativo em desenvolvimento tenha um conjunto abrangente de testes que ofereça confiança no que foi desenvolvido até então.

4. **(CESPE / BANRISUL – 2022)** O TDD (Test-Driven Development), como atividade da XP, é uma forma disciplinada de organizar o código, alterando-o de modo a aprimorar sua estrutura interna, sem que se altere o comportamento externo do software.

5. **(FGV / SEFAZ-MG – 2023)** Você entrou para um projeto novo, já em andamento, no qual a metodologia que a equipe do projeto segue é a de definir e escrever testes de software a partir das regras de negócio antes mesmo de implementar as funcionalidades propostas. Assinale a opção que indica o nome desse processo de desenvolvimento de software.

   a) DDD
   b) TDD
   c) BDD
   d) XP
   e) Scrum

6. **(FGV / Senado Federal – Análise de Sistemas – 2022)** Durante o processo de construção de software, a metodologia de Desenvolvimento Orientado a Testes é muito aplicada.

   A ordem utilizada na prática do TDD é:

a) escrever a funcionalidade após escrever os testes unitários e, por fim, refatorar o código implementado.

b) escrever os testes unitários após escrever a funcionalidade e, por fim, refatorar o código implementado.

c) escrever a funcionalidade após refatorar o código implementado e por fim escrever os testes unitários.

d) escrever os testes unitários após escrever a funcionalidade e, por fim, escrever os testes de integração.

e) escrever os testes de integração após escrever a funcionalidade e, por fim, refatorar o código.

7. **(CESPE / SERPRO – 2021)** Em TDD, os testes de um sistema devem ocorrer antes da implementação e ser oportunos, isolados e autoverificáveis.

8. **(CESPE / INMETRO / 2009)** A rotina diária dos desenvolvedores, ao empregar processos baseados no TDD (Test-Driven Development), é concentrada na elaboração de testes de homologação.

9. **(CESPE / INPI / 2013)** Usando-se o TDD, as funcionalidades devem estar completas e da forma como serão apresentadas aos seus usuários para que possam ser testadas e consideradas corretas.

10. **(CESPE / ANCINE / 2013)** No desenvolvimento de software conforme as diretivas do TDD (Test-Driven Development), deve-se elaborar primeiramente os testes e, em seguida, escrever o código necessário para passar pelos testes.

11. **(CESPE / INMETRO / 2009)** Considerando uma organização na qual a abordagem de Test Driven Development (TDD) esteja implementada, assinale a opção correta.

a) Nessa organização, ocorre a execução de iterações com ciclo longo, isto é, com duração de alguns meses.

b) No início de cada iteração, a primeira atividade realizada pela equipe de desenvolvimento é produzir o código que será validado através de testes.

c) O refactoring é uma das primeiras atividades realizada no início de cada iteração.

d) Entre as atividades finais de cada iteração, o desenvolvedor escreve casos de teste automatizados, cuja execução verifica se houve a melhoria desejada ou se uma nova funcionalidade foi implementada.

e) Há coerência e inter-relação com os princípios promovidos pela prática da extreme programming (XP).

12. **(CESPE / MPOG / 2013)** Ao realizar o TDD (test-driven development), o programador é conduzido a pensar em decisões de design antes de pensar em código de implementação, o que cria um maior acoplamento, uma vez que seu objetivo é pensar na lógica e nas responsabilidades de cada classe.

13. **(CESPE / MPU – 2013)** Na metodologia TDD, ou desenvolvimento orientado a testes, cada nova funcionalidade inicia com a criação de um teste, cujo planejamento permite a identificação dos itens e funcionalidades que deverão ser testados, quem são os responsáveis e quais os riscos envolvidos.

14. **(CESPE / STF – 2013)** No TDD, o primeiro passo do desenvolvedor é criar o teste, denominado teste falho, que retornará um erro, para, posteriormente, desenvolver o código e aprimorar a codificação do sistema.

15. **(CESPE / TRT-17 – 2013)** TDD consiste em uma técnica de desenvolvimento de software com abordagem embasada em perspectiva evolutiva de seu desenvolvimento. Essa abordagem envolve a produção de versões iniciais de um sistema a partir das quais é possível realizar verificações de suas qualidades antes que ele seja construído.

16. **(CESPE / ALRN – 2013)** Um típico ciclo de vida de um projeto em TDD consiste em:

I. Executar os testes novamente e garantir que estes continuem tendo sucesso.
II. Executar os testes para ver se todos estes testes obtiveram êxito.
III. Escrever a aplicação a ser testada.
IV. Refatorar (refactoring).
V. Executar todos os possíveis testes e ver a aplicação falhar.
VI. Criar o teste.

A ordem correta e cronológica que deve ser seguida para o ciclo de vida do TDD está expressa em:

a) IV – III – II – V – I – VI.
b) V – VI – II – I – III – IV.
c) VI – V – III – II – IV – I.
d) III – IV – V – VI – I – II.
e) III – IV – VI – V – I – II.

17. **(FGV / ALMT – 2013)** Com relação ao desenvolvimento orientado (dirigido) a testes (do Inglês Test Driven Development – TDD), analise as afirmativas a seguir.

I. TDD é uma técnica de desenvolvimento de software iterativa e incremental.

II. TDD implica escrever o código de teste antes do código de produção, um teste de cada vez, tendo certeza de que o teste falha antes de escrever o código que irá fazê-lo passar.

III. TDD é uma técnica específica do processo XP (Extreme Programming), portanto, só pode ser utilizada em modelos de processos ágeis de desenvolvimento de software.

Assinale:

a) Se somente as afirmativas I e II estiverem corretas.
b) Se somente as afirmativas I e III estiverem corretas.
c) Se somente as afirmativas II e III estiverem corretas.
d) Se somente a afirmativa III estiver correta.
e) Se somente a afirmativa I estiver correta.

18. **(FCC / TRT-MG – 2015)** Um analista de TI está participando do desenvolvimento de um software orientado a objetos utilizando a plataforma Java. Na abordagem de desenvolvimento adotada, o código é desenvolvido de forma incremental, em conjunto com o teste para esse incremento, de forma que só se passa para o próximo incremento quando o atual passar no teste. Como o código é desenvolvido em incrementos muito pequenos e são executados testes a cada vez que uma funcionalidade é adicionada ou que o programa é refatorado, foi necessário definir um ambiente de testes automatizados utilizando um framework popular que suporta o teste de programas Java.

A abordagem de desenvolvimento adotada e o framework de suporte à criação de testes automatizados são, respectivamente,

a) Behavior-Driven Development e JTest.
b) Extreme Programming e Selenium.
c) Test-Driven Development e Jenkins.
d) Data-Driven Development and Test e JUnit.
e) Test-Driven Development e JUnit.

19. **(CESPE / TRE-PE – 2017)** O desenvolvimento orientado a testes (TDD):

a) é um conjunto de técnicas que se associam ao XP (extreme programming) para o desenvolvimento incremental do código que se inicia com os testes.

b) agrega um conjunto de testes de integração para avaliar a interconexão dos componentes do software com as aplicações a ele relacionadas.

c) avalia o desempenho do desenvolvimento de sistemas verificando se o volume de acessos/transações está acima da média esperada.

d) averigua se o sistema atende aos requisitos de desempenho verificando se o volume de acessos/transações mantém-se dentro do esperado.

e) testa o sistema para verificar se ele foi desenvolvido conforme os padrões e a metodologia estabelecidos nos requisitos do projeto.

20. **(CESPE / STM – 2018)** O TDD (test driven development) parte de um caso de teste que caracteriza uma melhoria desejada ou nova funcionalidade a ser desenvolvida, de modo a confirmar o comportamento correto e possibilitar a evolução ou refatoração do código.

21. **(CESPE / TRE/PI – 2016)** O TDD (test driven development):

a) apresenta como vantagem a leitura das regras de negócio a partir dos testes, e, como desvantagem, a necessidade de mais linhas de códigos que a abordagem tradicional, o que gera um código adicional.

b) impede que seja aplicada a prática de programação em pares, que é substituída pela interação entre analista de teste, testador e programador.

c) é um conjunto de técnicas associadas ao eXtremme Programing e a métodos ágeis, sendo, contudo, incompatível com o Refactoring, haja vista o teste ser escrito antes da codificação.

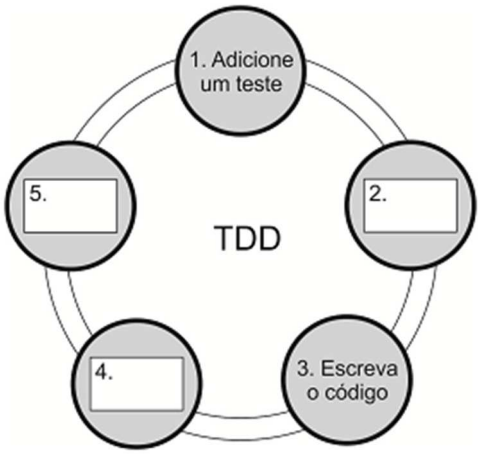
d) refere-se a uma técnica de programação cujo principal objetivo é escrever um código funcional limpo, a partir de um teste que tenha falhado.

e) refere-se a uma metodologia de testes em que se devem testar condições, loops e operações; no entanto, por questão de simplicidade, não devem ser testados polimorfismos.

22. **(UFRRJ / UFRRJ – 2015)** Os testes de unidade têm papel central na metodologia de implementação dirigida por testes, popularizada pelo processo XP e adotada em outros métodos. Esses testes são criados primeiro, exercitando o contrato de cada operação implementada pelos métodos. Em seguida, o código dos métodos é escrito para cumprir os contratos e, portanto, passar nos testes de unidade. Esse cenário corresponde à abordagem

a) TDD.
b) MDD.
c) DDC.
d) MDE.
e) FDD.

23. **(FCC / TRE-PR – 2017)** Considere o ciclo do Test-Driven Development – TDD.



A Caixa:

a) 2. corresponde a "Execute os testes automatizados".
b) 4. corresponde a "Refatore o código".
c) 5. corresponde a "Execute os testes novamente e observe os resultados".
d) 4. corresponde a "Execute os testes automatizados".
e) 5. corresponde a "Faça todos os testes passarem".

24. (IESES / TRE/MA – 2015) A respeito da técnica de testes TDD é correto afirmar que:

a) Testa o software com base no comportamento esperado.
b) É uma prática para desenvolvimento de testes unitário que pode utilizar o processo Red-Green-Refactor.
c) Utiliza-se da estrutura Dado, Quando e Então para montar os testes.
d) Prega que os testes devem ser realizados sempre após a implementação ser concluída.

25. (CESPE / TRE/RS – 2015) Projeto para o desenvolvimento de software que utilize TDD deve:

a) realizar sprints a cada quinzena.
b) desenvolver pequenos releases.
c) apresentar grande quantidade de testes unitários de código-fonte previamente desenvolvidos.
d) apresentar linguagem de programação estruturada.
e) recomendar a preparação dos testes para que, posteriormente, seja desenvolvido o código.

26. (FCC / TRE/AP – 2015) O TDD – Test Driven Development (Desenvolvimento orientado a teste):

a) é parte das metodologias ágeis UP – Unified Process e XP – Extreme Programming, tendo sido criado para ser usado em metodologias que respeitam os 4 princípios do Manifesto Ágil.

b) transforma o desenvolvimento, pois deve-se primeiro implementar o sistema antes de escrever os testes. Os testes são utilizados para facilitar no entendimento do projeto e para clarear o que se deseja em relação ao código.

c) baseia-se em um ciclo simples: escreve-se um código -> cria-se um teste para passar no código -> refatora-se.

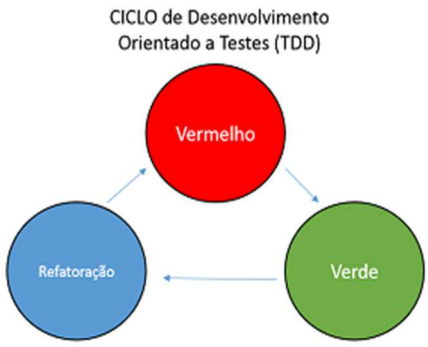
d) propõe a criação de testes que validem o código como um todo para reduzir o tempo de desenvolvimento.

e) beneficia-se de testes que seguem o modelo FIRST: F (Fast) I (Isolated) R (Repeatable) S (Self-verifying) T (Timely).

27. **(CESPE / TRE/TO – 2017)** O TDD (Test-Driven Development), que vem sendo adotado para testar os projetos de software,

a) utiliza os testes de caixa preta antes da entrega do software.
b) agiliza os testes por amostragem sem compatibilidade retroativa.
c) cobre amplamente os testes unitários.
d) escreve o teste antes da codificação do software.
e) realiza refactoring antes de escrever a aplicação a ser testada.

28. **(FGV / IBGE – 2016)** O Desenvolvimento Orientado a Testes (TDD) é um método de desenvolvimento criado e disseminado por Kent Beck em seu livro "Test-driven development". O método define regras, boas práticas e um ciclo de tarefas com 3 etapas: a etapa vermelha, a etapa verde e a etapa de refatoração, ilustrado na imagem abaixo:



Com relação às regras e boas práticas de TDD e ao seu ciclo, é correto afirmar que:

a) pode-se escrever testes que não compilam na etapa vermelha;
b) na etapa verde deve-se escrever código que testa uma funcionalidade a fundo de forma criteriosa e detalhada;
c) código novo só é escrito se um teste automatizado passar;
d) a duplicação é tolerada na etapa de refatoração;

e) é uma boa prática de TDD iniciar o desenvolvimento do código de uma funcionalidade e, logo em seguida, testá-la.

29. **(IADES / EBSERH – 2013)** Assinale a alternativa que não corresponde a uma das fases do processo de desenvolvimento, dirigido a testes (TDD).

a) Executar o teste, com os outros testes implementados, que rodarão e fornecerão o resultado de que o software está sem problemas.
b) Escrever o teste para a funcionalidade e implementação.
c) Realizar a identificação do incremento de funcionalidade.
d) Implementar a funcionalidade e executar novamente o teste.
e) Implementar a próxima parte da funcionalidade, após todos os testes terem sido executados, com sucesso

30. **(PR-4 UFRJ / UFRJ – 2018)** O ciclo do TDD - Test Driven Development, ou, em português, Desenvolvimento Guiado por Testes consiste em:

a) implementar teste unitário falho, tornar o teste bem-sucedido e refatorar.
b) implementar a funcionalidade, executar teste unitário e refatorar.
c) implementar teste unitário falho, refatorar e tornar o teste bem-sucedido.
d) implementar a funcionalidade, refatorar e tornar o teste bem-sucedido.
e) refatorar, executar teste unitário e implementar a funcionalidade.

31. **(CESPE / STJ – 2015)** Um dos passos executados no ciclo de atividades do processo TDD é a criação de novos testes para as falhas encontradas no código original, sem alteração deste.

32. **(CS-UFG / AL-GO – 2015)** O desenvolvimento dirigido a testes (TDD, do Inglês Test-Driven Development) é uma abordagem de desenvolvimento de software na qual se intercalam testes e desenvolvimento de código. Uma das características da abordagem TDD é:

a) a sua utilidade no desenvolvimento de softwares novos.
b) o maior custo associado aos testes de regressão.
c) a redução da importância da automatização dos testes.
d) a sua adequação a processos de software sequenciais.

33. **(UECE-CEV / FUNCEME – 2018)** Test-driven Development (TDD) é uma abordagem para o desenvolvimento de programas em que se intercalam testes e desenvolvimento de código (Sommerville, I. Engenharia de Software, 9a edição, 2011).

A respeito do TDD, é correto afirmar que:

a) consiste em um processo iterativo que se inicia escrevendo um código de uma funcionalidade do sistema e, logo em seguida, testa-o para saber se a implementação foi correta.
b) apesar de útil, não diminui o custo de testes de regressão do sistema.

c) sua utilização elimina a necessidade de testes de validação do sistema, uma vez que ele já foi testado incrementalmente.
d) apesar de ter sido apresentado como parte dos métodos ágeis, também pode ser usado em outros processos de desenvolvimento de software.

34. **(FAUGRS / UFRGS – 2018)** _____ é uma abordagem para o desenvolvimento de programas em que se intercalam testes e desenvolvimento de código. Essencialmente, desenvolve-se um código de forma incremental em conjunto com um teste para este incremento. Não se avança para o próximo incremento até que o código desenvolvido passe no teste. Essa abordagem foi introduzida como parte de métodos ágeis, mas pode ser também usada em processos de desenvolvimento dirigido a planos.

Assinale a alternativa que preenche corretamente a lacuna do texto acima.

a) Desenvolvimento Guiado por Testes (TDD)
b) Desenvolvimento em Espiral
c) Engenharia Dirigida a Modelos (MDD)
d) Rational Unified Process (RUP)
e) Teste de Sistema

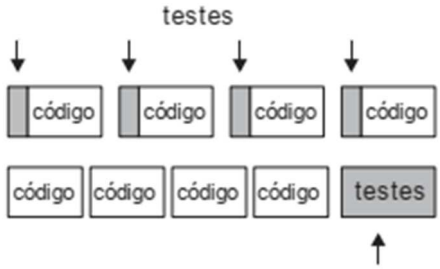
35. **(VUNESP / TCE-SP – 2015)** No Desenvolvimento Orientado a Testes (TDD), os casos de teste que definem o recurso a ser implementado devem ser elaborados:

a) assim que o código do teste estiver pronto.
b) antes de o código do recurso ser desenvolvido.
c) após o código do recurso ter sido completamente documentado.
d) simultaneamente com o desenvolvimento do código do recurso.
e) somente se o código do recurso apresentar erros.

36. **(IBFC / EMBASA – 2017)** No Ciclo de Desenvolvimento do TDD (Test-Driven Development), utiliza-se a estratégia que aplica três palavras-chaves (em inglês), que é denominada:

a) Red, Green, Refactor
b) White, Gray, Black
c) White, Black, Refactor
d) Green, Yellow, Red

37. **(FCC / CREMESP – 2016)** Considere a figura abaixo que apresenta duas abordagens de teste.



A figura:

a) ilustra as duas fases do TDD, que correspondem a escrever pequenos testes e testá-los no final.
b) mostra o ciclo conhecido como Vermelho-Verde-Refatora.
c) apresenta a diferença entre testes automatizados e testes manuais no XP.
d) mostra que um desenvolvedor que pratica TDD tem mais feedbacks do que um que escreve testes ao final.
e) evidencia que TDD é impraticável, pois o desenvolvedor gasta muito tempo escrevendo código de testes.

38. **(CESPE / ANATEL – 2014)** Em se tratando de desenvolvimento de softwares dirigidos a testes (TDD), a execução dos testes é realizada antes da implementação da funcionalidade.

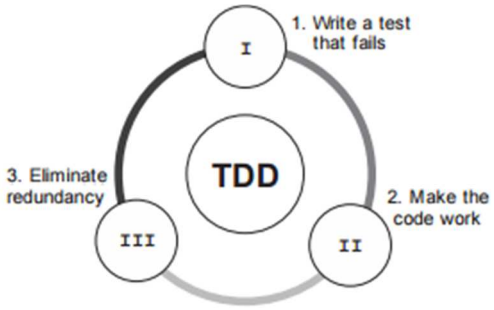
39. **(CESPE / TC/DF – 2014)** No TDD, o refatoramento do código deve ser realizado antes de se escrever a aplicação que deve ser testada.

40. **(CESPE / STJ – 2015)** No método de desenvolvimento TDD (Test Driven Development), o desenvolvedor escreve primeiro um caso de teste e, posteriormente, o código.

41. **(CESPE / MPE-PI – 2018)** O TDD possibilita o desenvolvimento de softwares fundamentado em testes. O ciclo de desenvolvimento do TDD segue os seguintes passos:

- escrever um teste que inicialmente não passa;
- adicionar uma nova funcionalidade do sistema;
- fazer o teste passar;
- realizar a integração contínua do código;
- escrever o próximo teste.

42. **(FCC / Pref. de Teresina/PI – 2016)** O Test Driven Development – TDD é uma das práticas sugeridas na eXtreme Programming – XP, onde o programador escreve o teste antes de escrever o código. O ciclo de desenvolvimento utilizando TDD é mostrado abaixo.



Considere:

I. Etapa inicial, onde se escreve um teste que falha, para alguma funcionalidade que ainda será Escrita.
II. Já com o teste criado, é o momento de executar o teste.
III. Eliminar códigos redundantes, remover acoplamentos, enfim, identificar pontos de melhoria no código.

As etapas I, II e III são, respectivamente,

a) Iniciação, Execução e Controle.
b) Red, Green e Refactor.
c) Iniciação, Atuação e Otimização.
d) Plan, Do e Check.
e) Planejamento, Execução e Melhoria.

43. **(FAUGRS / BANRISUL – 2018)** Considere as ações abaixo, executadas em desenvolvimento orientado a testes, Test-Driven Design (TDD).

I - Escrever código de teste.
II - Verificar se o teste falha.
III - Escrever código de produção.
IV - Executar teste até passar (reescrevendo o código de produção, se for necessário, até que o teste passe).
V - Refatorar código de produção e/ou de teste para melhorá-lo.

Considerando que se deseja incluir um novo caso de teste, assinale a alternativa que apresenta a sequência de ações que devem obrigatoriamente ocorrer para essa inclusão, segundo o TDD.

a) I, III e IV.
b) III, I e IV.
c) I, II, III e IV.
d) I, III, IV e V.
e) I, II, III, IV e V.

44. **(FGV / IBGE – 2017)** Test Driven Development (TDD) é uma prática muito utilizada no processo de desenvolvimento de sistemas computacionais. Analise as afirmativas a seguir sobre o uso da prática de TDD:

I. Tornam os testes de regressão mais demorados porque o desenvolvedor precisará fazer testes manuais várias vezes por dia.

II. Garante que os requisitos do sistema sejam atendidos porque o desenvolvedor escreverá o código de testes sempre que acabar a implementação do código do sistema.

III. Ajuda o desenvolvedor a escrever código de qualidade porque ele gastará parte do seu tempo escrevendo código de testes.

Está correto o que se afirma em:

a) somente I;
b) somente II;
c) somente III;
d) somente II e III;
e) I, II e III.

45. **(CESPE / ANATEL – 2014)** Na atividade de TDD (test-driven development), a escrita de teste primeiro define implicitamente tanto uma interface quanto uma especificação do comportamento para a funcionalidade que está sendo desenvolvida, estando, entretanto, a viabilidade do uso dessa abordagem limitada aos processos de desenvolvimento de software que seguem as práticas ágeis.

46. **(CESPE / TRE/MT – 2015)** Considere as seguintes etapas de um processo do tipo desenvolvimento orientado a testes (TDD).

I Implementar funcionalidade e refatorar.
II Identificar nova funcionalidade.
III Executar o teste.
IV Escrever o teste.
V Implementar a próxima parte da funcionalidade.

Assinale a opção que apresenta a sequência correta em que essas etapas devem ser realizadas.

a) I; IV; III; II; V
b) IV; III; II; I; V
c) I; IV; II; III; V
d) II; IV; III; I; V
e) IV; II; III; I; V

47. **(FCC / SEFAZ-SC – 2018)** O Test-Driven Development (TDD) é uma abordagem para o desenvolvimento de programas em que se intercalam testes e desenvolvimento de código. As etapas do processo fundamental de TDD são mostradas abaixo em ordem alfabética:

I. Escrever um teste para a funcionalidade identificada e implementá-lo como um teste automatizado.

II. Executar o teste, junto com os demais testes já implementados, sem implementar a nova funcionalidade no código.

III. Identificar e implementar uma outra funcionalidade, após todos os testes serem executados com sucesso.

IV. Identificar uma nova funcionalidade pequena para ser incrementada com poucas linhas em um código.

V. Implementar a nova funcionalidade no código e reexecutar o teste.

VI. Refatorar o código com melhorias incrementais até que o teste execute sem erros.

VII. Revisar a funcionalidade e o teste, caso o código execute sem falhar.

Considerando o item IV a primeira etapa e o item III a última etapa, a sequência intermediária correta das etapas do processo é:

a) I – II – VII – V e VI.
b) I – V – II – VII e VI.
c) I – VI – V – VII e II.
d) V – I – II – VII e VI.
e) V – I – VI – VII e II.

GABARITO – DIVERSAS BANCAS

1. LETRA D
2. ERRADO
3. CORRETO
4. ERRADO
5. LETRA B
6. LETRA A
7. CORRETO
8. ERRADO
9. ERRADO
10. CORRETO
11. LETRA E
12. ERRADO
13. CORRETO
14. CORRETO
15. CORRETO
16. LETRA C
17. LETRA A
18. LETRA E
19. CORRETO
20. CORRETO
21. LETRA D
22. LETRA A
23. LETRA D
24. LETRA B
25. LETRA E
26. LETRA E
27. LETRA D
28. LETRA A
29. LETRA A
30. LETRA A
31. ERRADO
32. LETRA A
33. LETRA D
34. LETRA A
35. LETRA B
36. LETRA A
37. LETRA D
38. CORRETO
39. ERRADO
40. CORRETO
41. ERRADO
42. LETRA B
43. LETRA C
44. LETRA C
45. ERRADO
46. LETRA D
47. LETRA A

BEHAVIOUR DRIVEN DEVELOPMENT (BDD)

Conceitos Básicos

INCIDÊNCIA EM PROVA: BAIXA

No ano de 2003, Dan North desenvolveu o *Behaviour Driven Development* (BDD) como uma resposta aos famosos Test Driven Development (TDD) e Acceptance Test Driven Development (ATDD). Na verdade, eu diria que foi mais uma evolução do que uma resposta! **Essa metodologia de desenvolvimento ágil busca tornar as práticas destas outras metodologias mais acessíveis e intuitivas para os novatos e especialistas.**

O BDD colabora para que o desenvolvimento foque na entrega de valor, através da formação de um vocabulário comum, reduzindo a distância entre o Negócio e a TI. Em geral, o cliente deve prover à equipe informações sobre o problema que ele quer resolver, **e juntos podem pensar nos exemplos concretos que vão nortear o processo de desenvolvimento.** Galera, exemplos são essenciais nessa metodologia! *Por que, professor?*

Cara, vocês são estudantes, vocês sabem muito bem. *Qual a melhor maneira de entender algo?* Por meio de exemplos! **Dessa forma, é muito mais fácil entender um domínio de negócios complexo.** *Vocês já tiveram que modelar um domínio em que vocês não sacam nada? Que é tudo vago e obscuro?* Pois é, a luz no fim do túnel é o exemplo. O ser humano precisa de exemplos para compreender um tópico.

Exemplos reais são uma excelente forma de se comunicar, e no nosso dia-a-dia nós os usamos sem nem mesmo perceber. Ao trabalhar com exemplos reais, nós nos comunicamos melhor, pois as pessoas serão capazes de se relacionar com eles mais facilmente. **Isso tudo é muito mais perceptível quando o domínio do negócio é complexo.** O BDD busca melhorar a comunicação e a interação entre os stakeholders (clientes, programadores, analistas de qualidade, etc).

Fica claro como o software deve se comportar por meio de cenários escritos em linguagem natural. Na verdade, é a combinação da linguagem natural com uma linguagem ubíqua – que é uma linguagem comum entre programadores e especialistas no domínio (jargões técnicos, terminologias do dia-a-dia, entre outros). **Isso minimiza ruídos de comunicação, previne falhas e reduz riscos – este é o grande foco dessa metodologia.** Vamos ver as principais práticas:

PRÁTICAS DO BDD
Envolver as partes interessadas no processo através de Outside-in Development (Desenvolvimento de fora para dentro);
Usar exemplos para descrever o comportamento de uma aplicação ou unidades de código;
Automatizar os exemplos para prover um feedback rápido e possivelmente testes de regressão;

Usar "*deve*" na hora de descrever o comportamento ajuda a esclarecer responsabilidades e permitir que funcionalidades sejam questionadas;
Usar simuladores de teste (*Mocks*, *Stubs*, *Fakes*, *Dummies*, *Spies*) para auxiliar na colaboração entre módulos e códigos que ainda não foram escritos.

Professor, como eu posso escrever um cenário? Cara, é muito fácil! Primeiro, eu crio uma história – todos elas seguem mais ou menos o mesmo padrão:

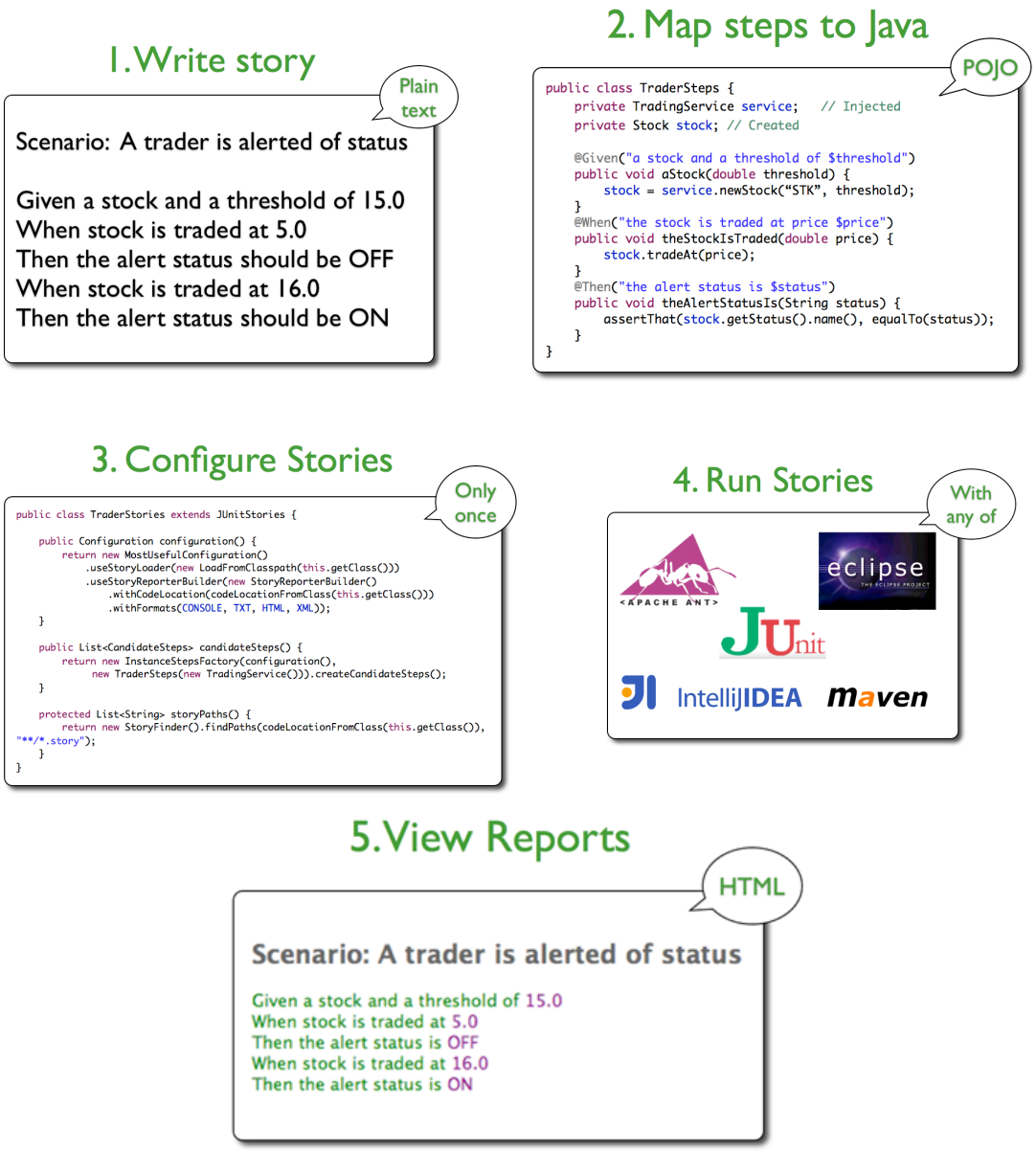
EU, COMO <PAPEL>, DESEJO <NECESSIDADE> PARA <MOTIVO>

Em outras palavras, pode ser entendido como: "*Eu, como Analista de Relacionamento, desejo poder manipular todas as informações de um chamado para que eu possa resolvê-lo*". **Pronto, essa é uma história que possui algum valor para o negócio!** No entanto, nosso papo não acabou! Baseado nessa história, eu devo criar um cenário – todos eles também seguem mais ou menos um mesmo padrão:

DADO QUE <CONTEXTO INICIAL>, QUANDO <EVENTO>, ENTÃO <RESULTADO>

Em outras palavras, pode ser entendido como: "*Dado que não existam chamados abertos, quando abrirem um chamado, então eu devo poder manipulá-lo*". Percebam que a história é apenas uma narrativa de base para a especificação de um ou mais cenários. **Vejam também que a linguagem é voltada para o domínio do negócio, sem detalhes técnicos.** *Entendido?* Agora vem a parte mais interessante: a comparação entre BDD x TDD!

Galera, vocês se lembram que eu falei para vocês que essa metodologia era uma evolução de TDD/ATDD? **Pois é, é possível automatizar testes escritos dessa maneira por meio de ferramentas específicas para realizar a validação do software, tais como: SpecFlow, JBehave, JSpec, RSpec, etc.** O JBehave, por exemplo, segue cinco passos bem definidos apresentados na imagem a seguir:



O BDD associa os benefícios de uma documentação formal, escrita e mantida pelo negócio, com testes de unidade que "demonstram" que essa documentação é efetivamente válida. Na prática, isso garante que a documentação deixa de ser um registro estático, que se converte em algo gradualmente ultrapassado, em um artefato dinâmico que reflete constantemente o estado atual de um projeto.

Legal, mas chegou o momento de resumir esse nosso papo! Se você estiver desesperado porque não conseguiu fechar o edital e precisa saber rapidamente apenas os fundamentos básicos dessa metodologia, eu vou facilitar sua vida e te mostrar o que você precisa saber em poucas palavras. **Vou fazer isso em um conjunto de pontos para que fique mais claro para vocês – espero que vocês compreendam. Vamos lá...**

- Behaviour-Driven Development (BDD) é uma técnica ágil de desenvolvimento de software focada em atender **funcionalidades ou comportamentos** desejados pelos usuários;

- Tentativa de **abarcar diversos conceitos**, tais como: Especificação por Exemplo; Definição de "Pronto"; Estórias de Usuário; Desenvolvimento Outside-In; TDD; ATDD; DDD; DSL; etc;

- Assim como o Test-Driven Development (TDD), essa técnica busca **escrever testes antes de codificar qualquer regra de negócio do sistema**;

- Em contraste com o Test-Driven Development (TDD), **testes são escritos sob o ponto de vista do usuário** e, não, sob o ponto de vista do desenvolvedor;

- Assim como o Acceptance Test-Driven Development (ATDD), essa técnica busca desenvolver o software **baseado em critérios automáticos de aceitação**;

- Assim como o Domain-Driven Development (DDD), **utiliza uma linguagem ubíqua**, i.e., um vocabulário comum entre usuários e desenvolvedores baseado no domínio do negócio;

- Isso permite uma **melhor comunicação/colaboração entre equipes**; compartilhamento de conhecimento; documentação dinâmica; visão holística do sistema; foco na entrega de valor;

- Apresenta um conjunto de cenários, protótipos e **especificações orientadas a exemplos**, que servem de entrada para o processo de desenvolvimento;

- TDD é similar a um Teste Caixa-Branca (especificado pelo desenvolvedor) e o **BDD é similar a um Teste Caixa-Preta (especificado pelo usuário)**;

(TRE/MA – 2015) Qual a alternativa correta a respeito do BDD?

a) Utiliza o conceito de Ubiquitous Language de modo a facilitar o entendimento de todos os fazendo com que os integrantes falem a mesma língua.

b) É uma técnica de desenvolvimento de testes que funciona exclusivamente se utilizada alguma metodologia ágil de desenvolvimento.

c) Técnica de testes onde somente os programadores participam, deixando analistas e testadores cuidando das demais atividades.

d) Maneira de testar os sistemas de acordo com o comportamento esperado, portando só pode ser feito quando o software estiver concluído.

Comentários: (a) Correto, nós vimos que ele realmente utiliza uma linguagem ubíqua de modo a facilitar a colaboração entre a equipe; (b) Errado, não é uma técnica de desenvolvimento de testes, é uma técnica de desenvolvimento de software orientada a testes comportamentais; (c) Errado, não é uma técnica de teste e não participam apenas programadores – participam todos

os membros da equipe, inclusive usuários; (d) Errado, é iterativo e incremental, logo pode ser feito enquanto o software está sendo construído (Letra A).

(TST – 2012) Leia o texto:

"O TDD – Test-Driven Development é focado em testes unitários, em que você isola um modelo (por exemplo) e monta-o de acordo com os testes que você escrever. Quando você tiver um determinado número de modelos, aí você testa a integração entre eles, e assim por diante. Fazendo uma analogia, isso é mais ou menos como construir o software "de dentro para fora", em que partimos escrevendo testes específicos (unitários) e depois vamos abrangendo outras regiões do sistema (integração, funcional, aceitação etc). Já em (.....) podemos dizer que o software é desenvolvido "de fora para dentro", já que os testes são construídos baseados nos requisitos do cliente ou em um roteiro de aceitação (também conhecidos por estórias). Esta prática é semelhante ao TDD: testes são escritos primeiro, funções depois. O diferencial é que (.....) aborda o comportamento e o resultado como um todo, não se preocupando, necessariamente, com as classes, métodos e atributos de forma isolada. Neste texto, foi omitida a referência à técnica conhecida como:

a) TFD – Test First Development
b) FTR – Formal Test Review
c) BDD – Behaviour-Driven Development
d) RTT – Real Time Test
e) FDT – Feature-Driven Test

Comentários: essa é uma excepcional definição de Behaviour-Driven Development (Letra C).

(TRE/BA – 2017) Entre os métodos e técnicas ágeis, a técnica que utiliza a linguagem ubíqua, visando, entre outras coisas, a integração de regras de negócios com linguagem de programação, com foco no comportamento do software, e na qual os testes orientam o desenvolvimento, ou seja, primeiro se escreve o teste e depois o código, é a:

a) BDD (Behavior Driven Development).
b) Kanban.
c) automação de builds.
d) automação de testes.
e) TDD (Test Driven Development).

Comentários: *Linguagem Ubíqua?* BDD! *Integração de regras de negócio com linguagem de programação?* BDD! *Testes orientam o desenvolvimento?* BDD ou TDD. Logo, só pode tratar do BDD (Letra A).

(BANRISUL – 2022) Desenvolvedores que se beneficiam das vantagens do BDD escrevem os testes em sua língua nativa, em combinação com a linguagem ubíqua.

_____

Comentários: BDD (Behavior Driven Development) é uma técnica de desenvolvimento usada para ajudar a criar software de forma colaborativa. Ele usa um formato de comunicação comum para descrever e validar a funcionalidade do software, permitindo que desenvolvedores, testadores, gerentes de projeto e outros participantes falem a mesma linguagem. Os testes são escritos em sua língua nativa, que é geralmente uma linguagem de programação, como Java ou C#. Esses testes então são traduzidos para uma linguagem ubíqua, como Gherkin, que é a linguagem usada para escrever o BDD. Essa linguagem é universalmente compreendida pelos participantes, independentemente de sua experiência com programação, permitindo que eles tomem parte ativa na definição de requisitos e na verificação (Correto).

(BANRISUL – 2022) Entre os métodos de testes ágeis, o TDD (Test-Driven Development) é uma extensão do BDD (Behavior Driven Development).

_____

Comentários: a questão inverteu os conceitos: BDD é uma extensão do TDD (Errado).

(BANRISUL – 2022) Entre os métodos de testes ágeis, o BDD (Behavior Driven Development) é aquele que, por natureza, é o mais orientado para o cliente.

_____

Comentários: BDD é uma abordagem de desenvolvimento orientada a comportamento. É uma estratégia de teste na qual os desenvolvedores de software, os clientes e os testadores colaboram para criar testes de software que descrevem o comportamento do sistema. O objetivo é criar testes baseados no comportamento esperado do software, que é entendido por todos os envolvidos (Correto).

(BANRISUL – 2022) No desenvolvimento orientado a comportamento (BDD), os ciclos iniciam-se com a criação de testes de unidade e integração.

_____

Comentários: BDD está relacionado a comportamentos de uma regra - quem se inicia com um teste de unidade é o TDD (Errado).

ESSA LEI TODO MUNDO CONHECE:
PIRATARIA É CRIME.

Mas é sempre bom revisar o porquê e como você pode ser prejudicado com essa prática.

1. 

Professor investe seu tempo para elaborar os cursos e o site os coloca à venda.

2. 

Pirata divulga ilicitamente (grupos de rateio), utilizando-se do anonimato, nomes falsos ou laranjas (geralmente o pirata se anuncia como formador de "grupos solidários" de rateio que não visam lucro).

3. 

Pirata cria alunos fake praticando falsidade ideológica, comprando cursos do site em nome de pessoas aleatórias (usando nome, CPF, endereço e telefone de terceiros sem autorização).

4. 

Pirata compra, muitas vezes, clonando cartões de crédito (por vezes o sistema anti-fraude não consegue identificar o golpe a tempo).

5. 

Pirata fere os Termos de Uso, adultera as aulas e retira a identificação dos arquivos PDF (justamente porque a atividade é ilegal e ele não quer que seus fakes sejam identificados).

6. 

Pirata revende as aulas protegidas por direitos autorais, praticando concorrência desleal e em flagrante desrespeito à Lei de Direitos Autorais (Lei 9.610/98).

7. 

Concurseiro(a) desinformado participa de rateio, achando que nada disso está acontecendo e esperando se tornar servidor público para exigir o cumprimento das leis.

8. 

O professor que elaborou o curso não ganha nada, o site não recebe nada, e a pessoa que praticou todos os ilícitos anteriores (pirata) fica com o lucro.

Deixando de lado esse mar de sujeira, aproveitamos para agradecer a todos que adquirem os cursos honestamente e permitem que o site continue existindo.