# Aula 00 (Prof. Diego Carvalho e Fernando Pedrosa)

*MPO - Analista de Planejamento e Orçamento - APO - Fundamentos de Gestão de Tecnologia da Informação - 2024 (Pós-Edital)*

Autor:
**André Castro, Diego Carvalho, Equipe Informática e TI, Paolla Ramos, Thiago Rodrigues Cavalcanti**

02 de Fevereiro de 2024

# Índice

# APRESENTAÇÃO

PROF. DIEGO CARVALHO

FORMADO EM CIÊNCIA DA COMPUTAÇÃO PELA UNIVERSIDADE DE BRASÍLIA (UNB), PÓS-GRADUADO EM GESTÃO DE TECNOLOGIA DA INFORMAÇÃO NA ADMINISTRAÇÃO PÚBLICA E, ATUALMENTE, AUDITOR FEDERAL DE FINANÇAS E CONTROLE DA SECRETARIA DO TESOURO NACIONAL.

ESTRATÉGIA CONCURSOS

**PROFESSOR DIEGO CARVALHO - WWW.INSTAGRAM.COM/PROFESSORDIEGOCARVALHO**

**Sobre o curso: 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 quando das 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.**

# 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.

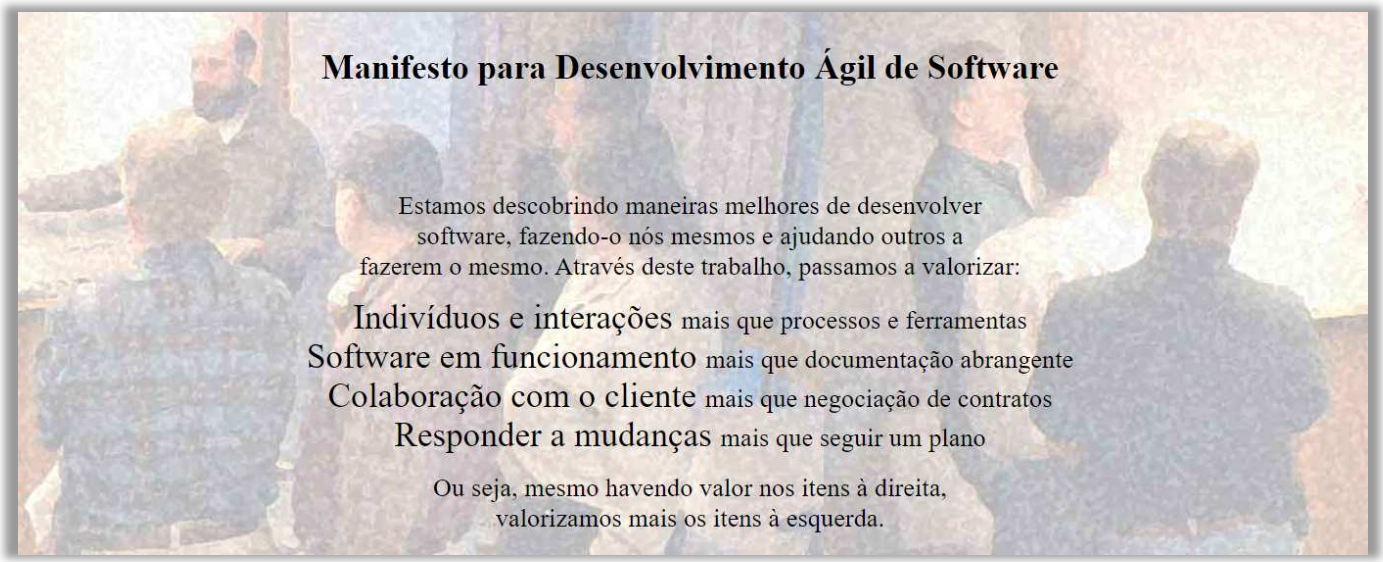
Foi aí que alguns acharam que era o momento de formalizar e elevar aquela reunião em um patamar maior. **Eles decidiram escrever um documento que serviria de grito de guerra contra os processos tradicionais de desenvolvimento de software que vigoravam naquela época**. Para isso, eles pensaram: "*Nós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

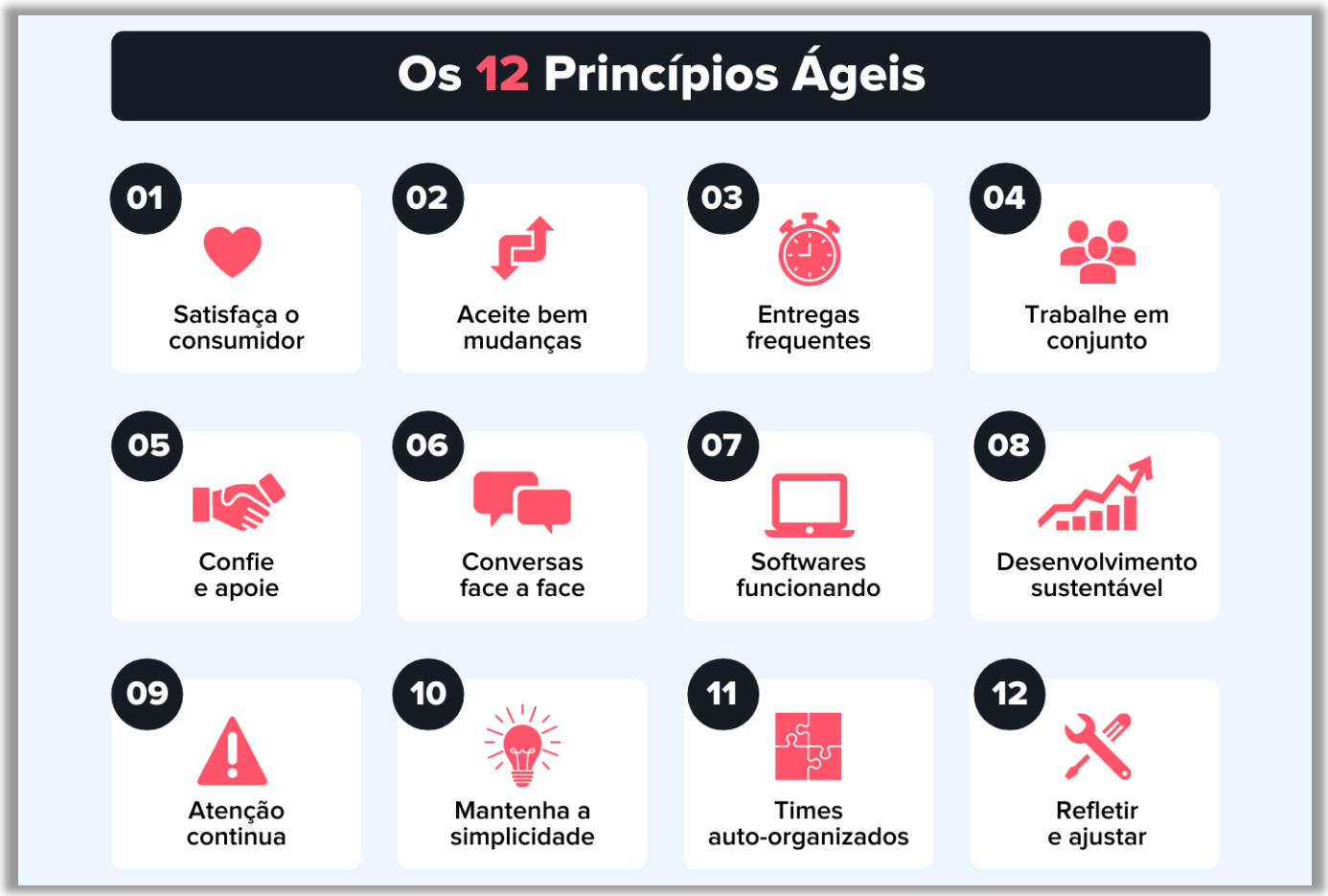
**ESTAMOS DESCOBRINDO MANEIRAS MELHORES DE DESENVOLVER SOFTWARE, FAZENDO-O NÓS MESMOS E AJUDANDO OUTROS A FAZEREM O MESMO. ATRAVÉS DESTE TRABALHO. PASSAMOS A VALORIZAR:**

| | | |
|---|---|---|
| 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

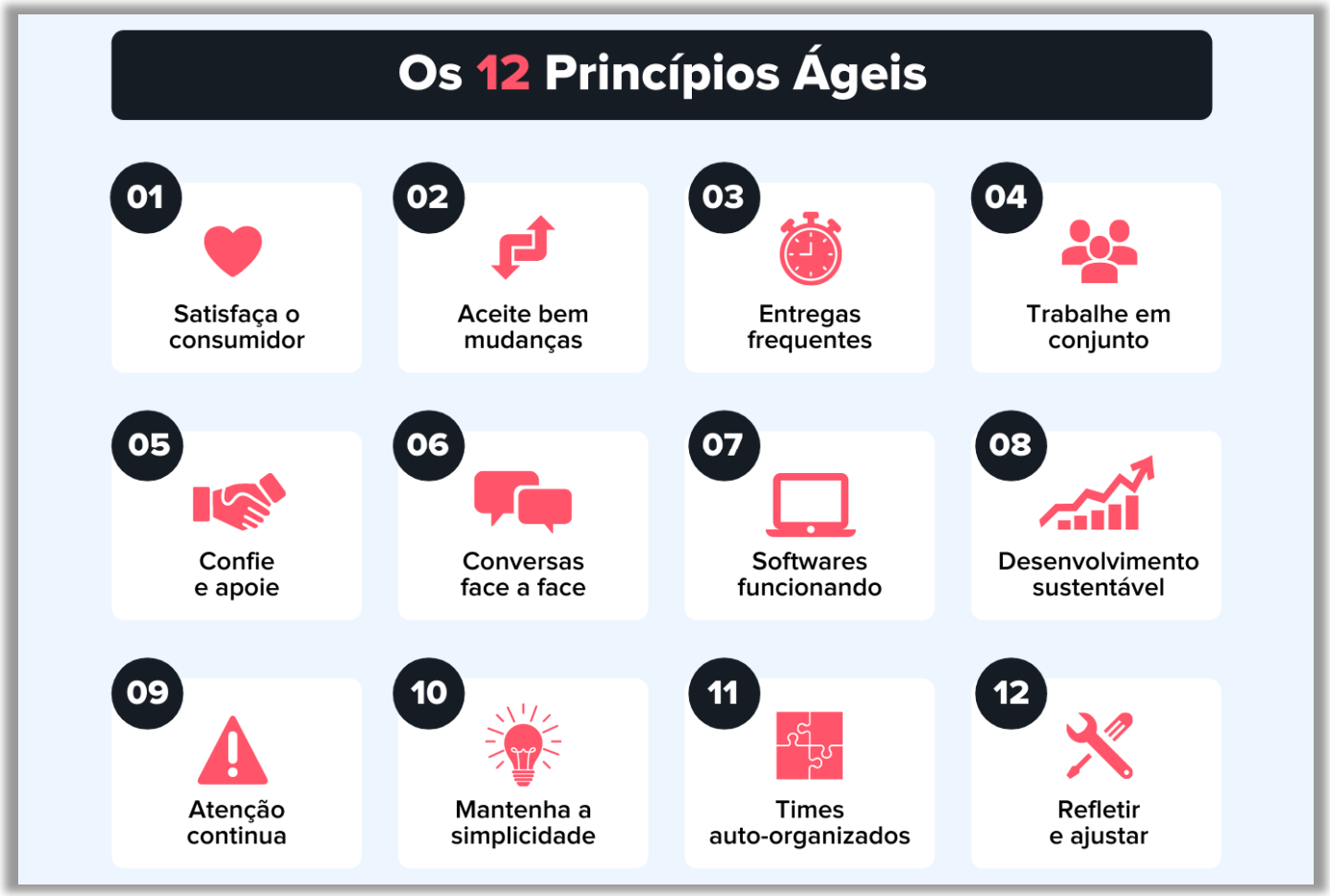
---

# 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

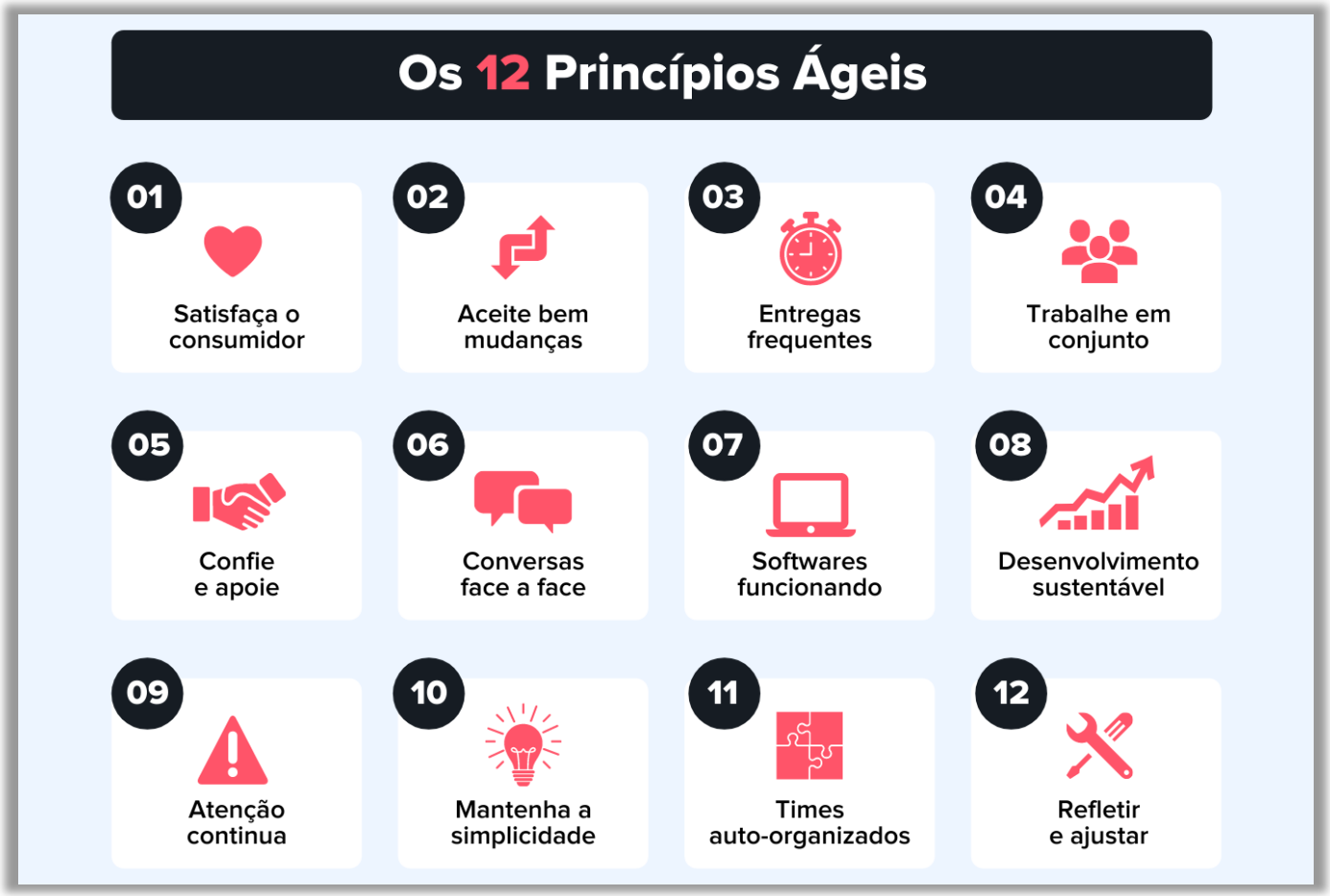
1. ERRADO
2. ERRADO
3. ERRADO
4. ERRADO
5. ERRADO
6. ERRADO
7. CORRETO
8. CORRETO
9. ERRADO
10. CORRETO
11. ERRADO
12. LETRA A
13. ERRADO
14. ERRADO
15. ERRADO
16. CORRETO
17. ERRADO
18. CORRETO
19. CORRETO

# 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.

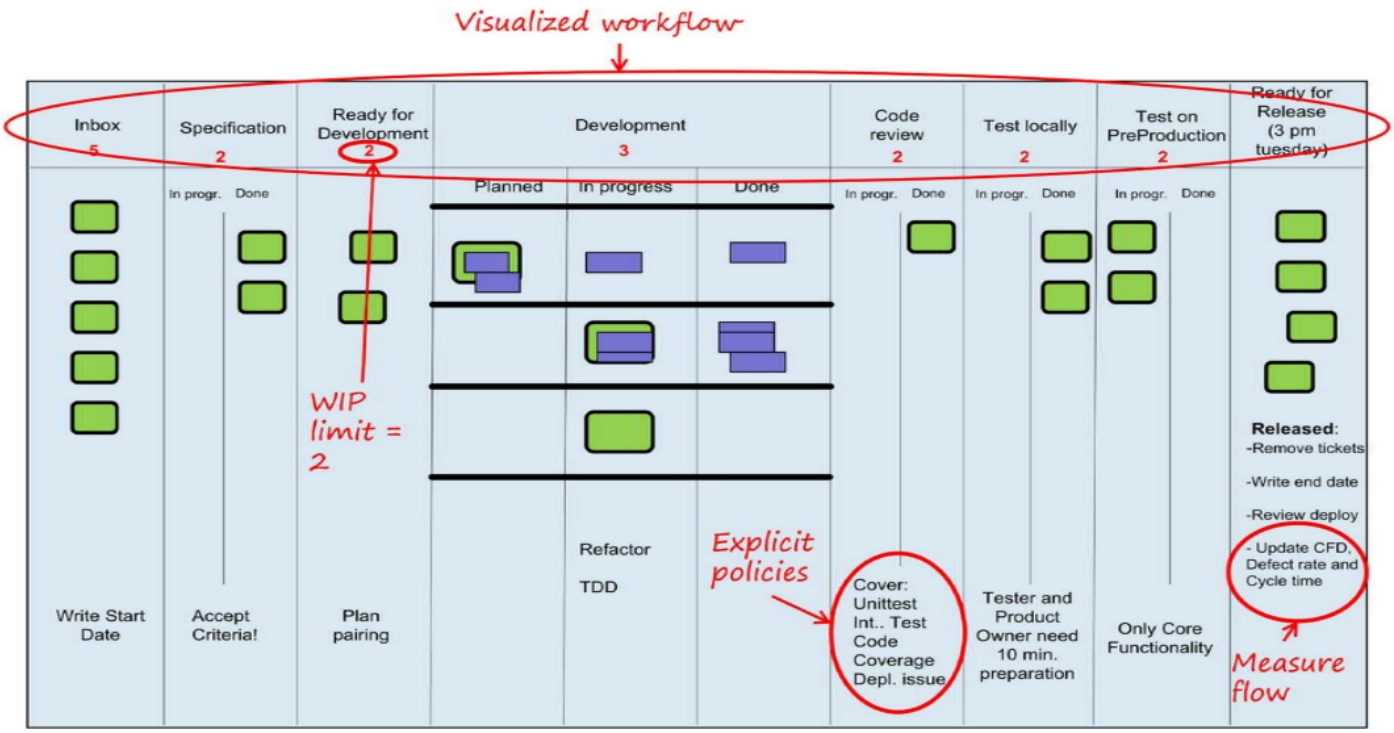
> *"Kanban **is not a software development life cycle or project management methodology**! It is not a way of making software or running projects that make software!" – David J. Anderson*
>
> *"There is **no kanban process for software development**. At least I am not aware of one. I have never published 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.

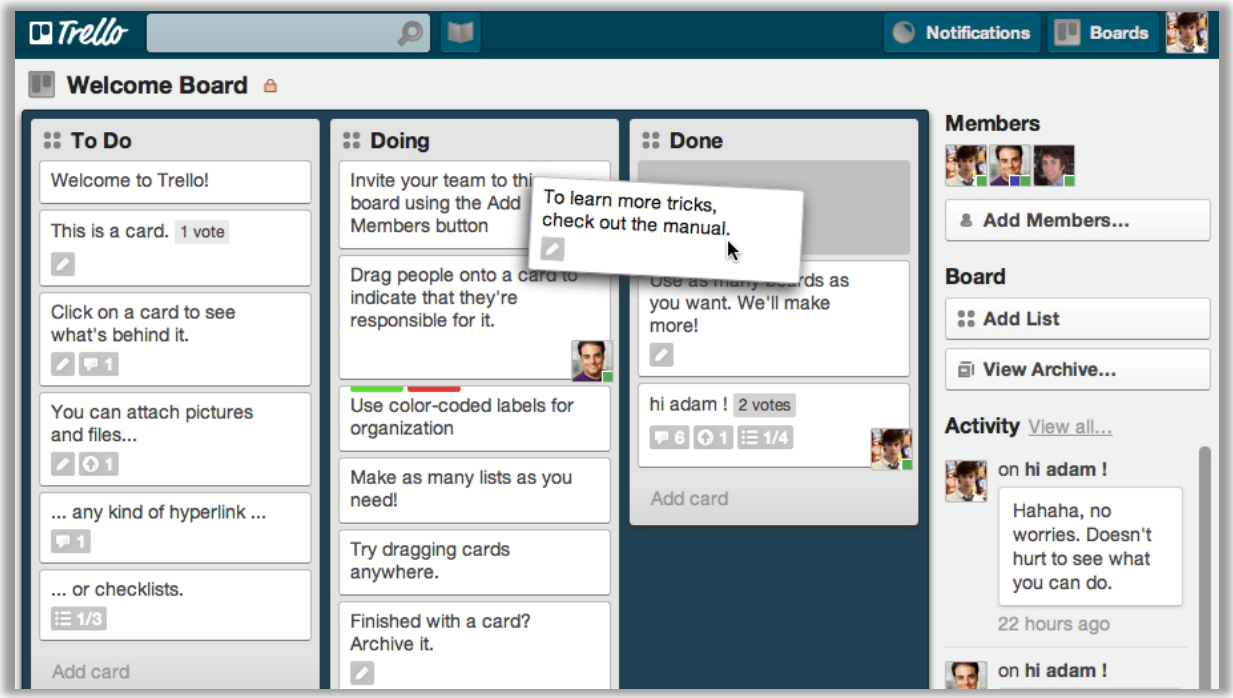
**Galera, existem diversas diferenças entre ambos: Scrum requer iterações e o Kanban, não – mas sugere-se que haja uma cadência de entradas e entregas.** Quando o Kanban incorpora iterações, ele é tipicamente chamado Scrumban (para diferenciá-lo do Scrum original), uma vez

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

# NOÇÕES DE DEVOPS

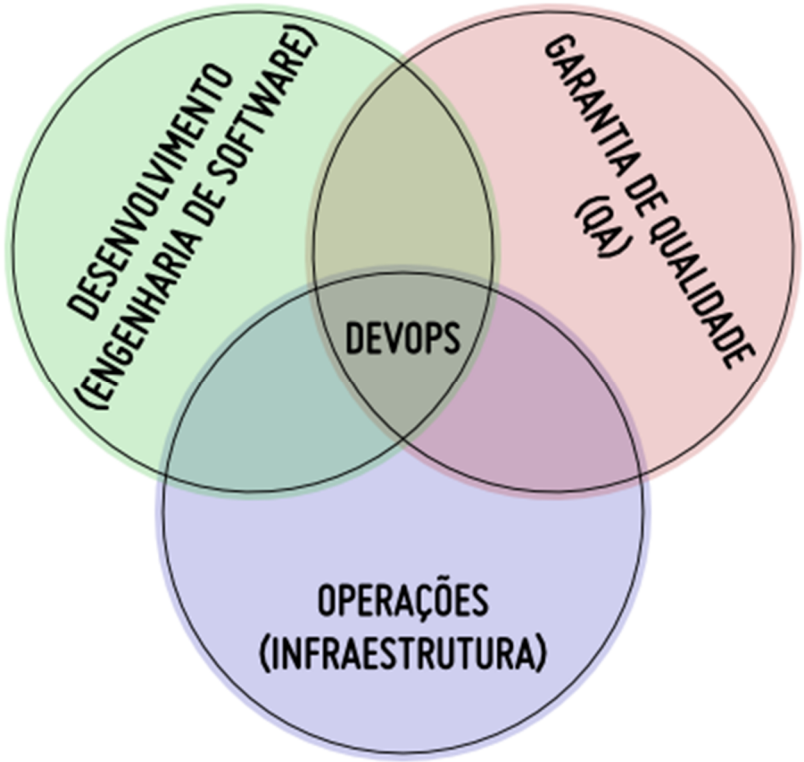
## Conceitos Básicos

INCIDÊNCIA EM PROVA: MÉDIA

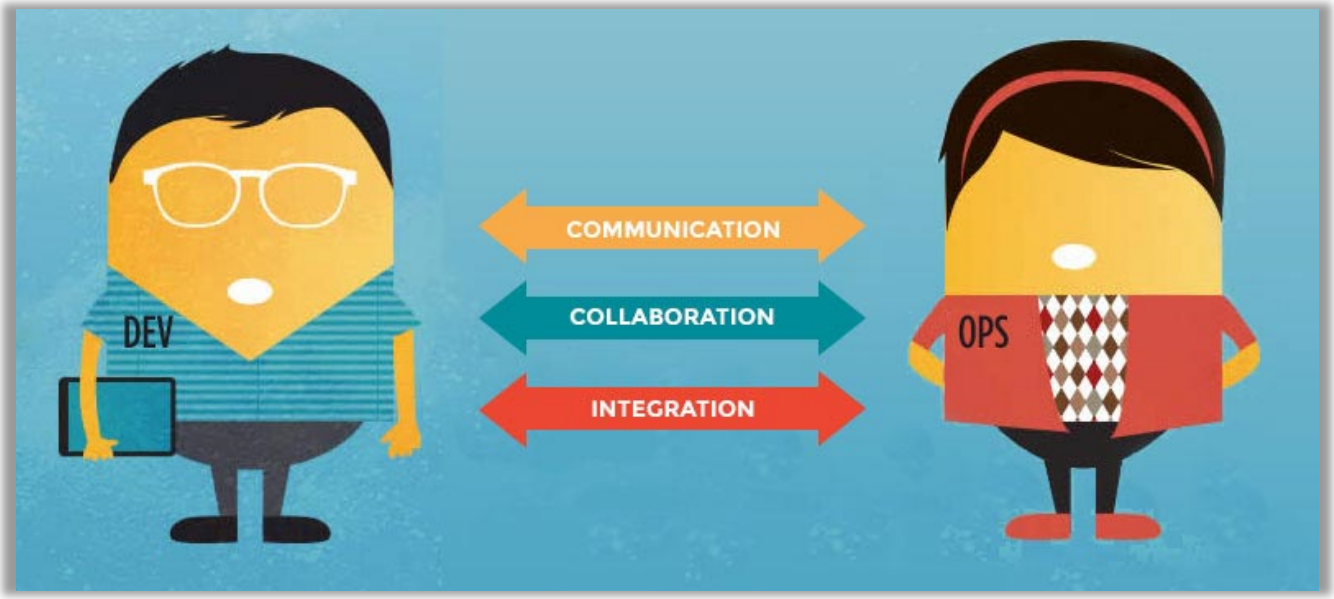
De um tempo para cá, nós temos vivenciado novos paradigmas em tecnologias de software. **As metodologias ágeis, por exemplo, vieram com um conjunto de práticas que desencadearam diversas outras práticas, de forma a obter software de maneira mais ágil e mais adaptável a possíveis mudanças**. No entanto, o grande lance é que as mudanças ocorrem com um intervalo de tempo cada vez menor.

Vocês já devem ter percebido isso! Antigamente, softwares lançavam atualizações em intervalos de 18 a 24 meses (ou até mais). **Atualmente, a dinâmica de consumo de aplicações de tecnologia da informação sofreu uma reviravolta e a demanda dos clientes é insana.** As empresas de software atualmente são duramente pressionadas para lançar e atualizar aplicativos no mercado o mais rápido possível.

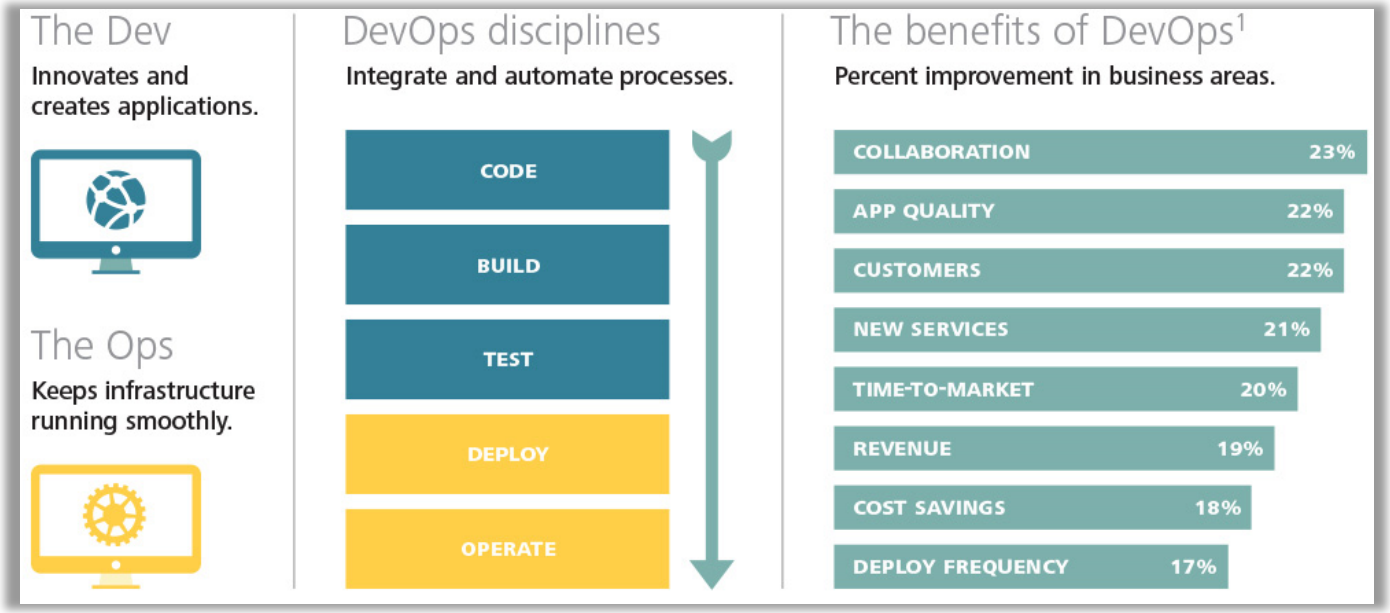
Pois é, o mundo mudou! O ciclo para a criação de novos aplicativos de software dura cerca de três meses para uma versão inicial e mais seis meses para o conjunto completo de recursos**. Não só o ciclo de vida foi encurtado, mas os aplicativos se tornaram muito mais complexos e exigem colaboração e integração cruzada entre os diversos componentes de tecnologia da informação, como Dev, Ops e Q&A.**



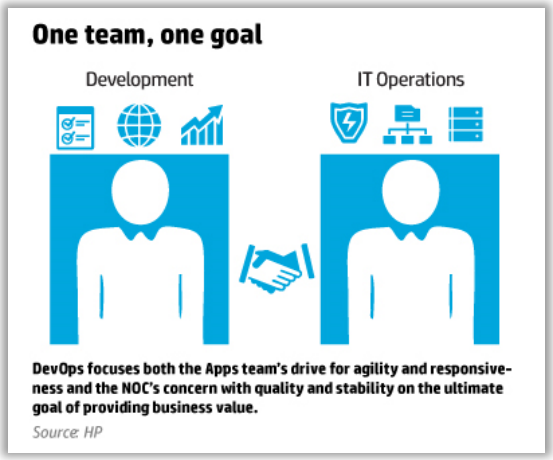
*Professor, espera um pouco aí! O que acontece se eu juntar conceitos de Desenvolvimento de Software, Operação de Sistemas e Garantia de Qualidade?* Surgirá, então, o conceito de DevOps! **Trata-se de um conceito muito simples, como apresenta a imagem acima. Essas ideias são tratadas em conjunto, em vez de separadas.** O lance é ter uma maior comunicação, colaboração e integração entre essas áreas.




*Professor, o que é exatamente DevOps? É uma cultura? É uma técnica? É uma metodologia de desenvolvimento de software?* Ainda não existe uma resposta precisa para essas perguntas! *Por que, professor?* **Porque foi um movimento que começou ao mesmo tempo em diversos lugares diferentes, tratando de infraestrutura e desenvolvimento, mas não houve um manifesto formal, como o manifesto ágil.**



**Parece simples fazer a infraestrutura conversar de forma harmônica com o desenvolvimento, mas não é tão fácil!** *Qual é o papel da infraestrutura?* É sustentar os sistemas em produção; monitorar o funcionamento e a performance; cuidar da estabilidade, segurança, níveis de serviço; planejar mudanças, minimizando riscos; entre outros. *O que acontece se uma aplicação em produção parar de funcionar?*




**Isso pode significar prejuízo financeiro ou institucional. Em suma, podemos dizer que a infraestrutura se preocupa em proteger o valor de negócio.** *E o desenvolvimento?* Esses caras se preocupam com inovação e criatividade, baseado nos requisitos do usuário. Desenvolvedor fica louco quando sai uma biblioteca, componente ou tecnologia nova. A consequência disso é que cada inovação significa um novo Deploy (feito pela rapaziada da infraestrutura).

*E se ocorrer algum problema?* Deve ser realizado um rollback (também pelo pessoal da infraestrutura). **Podemos afirmar, então, que o desenvolvedor se preocupa em aumentar o valor do negócio.** Vocês já devem ter notado que há um conflito interessante nessa conversa. Ora, o desenvolvedor quer colocar suas aplicações no ar o mais rápido possível para que fique disponível para o cliente.

**No entanto, a galera da infraestrutura quer ter certeza de que a aplicação está suficientemente estável para ir para produção sem gerar incidentes que parem o que já está funcionando.** *O que ocorria antigamente?* As empresas permitiam apenas um deploy por semana ou por mês! *Tem coisa mais não-ágil que isso?* Vai totalmente de encontro aos ideais do manifesto ágil. *Lembram-se dos conceitos de entrega, integração e teste contínuos?*

Pois é, a infraestrutura teve que se adaptar a realizar deploys diários. No entanto, os desenvolvedores – muitas vezes – se esqueciam de considerar algumas diferenças importantes entre ambientes de desenvolvimento e produção. **Isso gerava alguns incidentes, o cliente reclamava e começava uma briga muito comum representada pelas duas imagens anteriores.** Sim, galera... rola essa briga!



Desenvolvedores afirmando que a Infraestrutura é engessada, lenta e que não oferece um ambiente adequado para o desenvolvimento de aplicações; já a Infraestrutura afirmava que os desenvolvedores faziam código ruim e instável, e que a culpa não era deles. **Pessoal, voltem agora para a primeira imagem e percebam o que DevOps tenta integrar: Desenvolvimento, Infraestrutura e Qualidade!**

Parece briga de marido e mulher, mas ambos os lados têm que reconhecer seus erros, ceder e se adaptar! **O Desenvolvimento precisa pensar mais na Infraestrutura e controlar as fases de deploy (Ex: Deployment Pipeline). Já a Infraestrutura tem que evoluir para o mundo ágil.** Começar a trabalhar de forma automatizada e dinâmica, ser mais veloz para subir ambientes; reconstruir ou duplicar ambientes de acordo com as necessidades do Desenvolvimento.

**Galera, as principais características do DevOps são:** colaboração entre equipes; fim de divisões; relação saudável entre áreas; teste, integração e entrega contínuos; automação de deploy; controle e monitoração; gerenciamento de configuração; orquestração de serviços; avaliação de métricas e desempenho; logs e integração; velocidade de entrega; feedback intenso; e comunicação constante.

Em suma, podemos dizer que ele é um movimento, um conceito, uma cultura, uma abordagem que trata do feedback, comunicação e colaboração entre áreas de tecnologia da informação com o objetivo de garantir qualidade do software, com menor custo, mais rapidez, menor risco e maior eficiência. **É como se fosse a utilização de metodologias ágeis no desenvolvimento e na infraestrutura.** *Bacana?*

**Para o DevOps, o código gerado pela infraestrutura é apenas mais um artefato qualquer de desenvolvimento e, não, uma parte separada.** Trata-se de uma prática importante, já que não basta somente o desenvolvedor escrever o código do sistema, é necessário que a área de infraestrutura também atue para liberar, controlar e entregar a versão do sistema de forma contínua, periódica e preferencialmente automática.

# TODAY'S DEVOPS

## DEV + OPS

## PART DEV

### Application Performance

Modern developers use APM tools to decrease latency, have complete visibility into code, databases, caches, queues, and third-party services.

### End User Analytics

A great developer understands end users have the best feedback and analytics play an enormous part of understanding users. Developers are constantly monitoring end user latency and checking performance by devices and browsers.

### Quality Code

Developers need to ensure their deployments and new releases don't implode or degrade the overall performance.

### Code-Level Errors

When you have a large distributed application it is vital to lower MTTR by finding the root cause of errors and exceptions.

## PART OPS

### Application Availability

The applications need to be up and running and it's Ops responsibility to ensure uptime and SLAs are in order.

### Application Performance

Classic Ops generally rely on infrastructure metrics - CPU, memory, network and disk I/O, etc. Modern Ops correlate all of those metrics with application metrics to solve problems 10x faster.

### End User Complaints

The goal is to know about and fix problems before end users complain, reduce the number of support tickets, and eliminate false alerts.

### Performance Analytics

Automatically generated baselines of all metrics help Ops understand what has changed and where to focus their troubleshooting efforts. Alerts based upon deviation from observed baselines improve alert quality and reduce alert noise.

Vamos resumir: **a preocupação é com os objetivos quase diametralmente opostos da galera de Desenvolvimento (DEV) e Operações (OPS).** Os desenvolvedores querem programar novos recursos, melhorar o produto; corrigir bugs, etc. As operações querem colocar tudo em funcionamento e nunca mudar, já que as alterações causam novos erros, bugs, problemas de desempenho, entre outros.

O objetivo do DevOps é aliviar a tensão entre esses dois campos! **Ter pessoas de Operações nas trincheiras de Desenvolvimento é a principal maneira de alcançar esse objetivo.** Seu trabalho é facilitar ao máximo que desenvolvedores e operações façam o que precisam fazer. *Como assim?* Por exemplo: fornecendo ambientes idênticos de desenvolvimento, testes, homologação, *staging* e qualidade; configuração de pipelines de teste e implementação automáticos.

Além de estar envolvido no processo de desenvolvimento para que eles estejam mais preparados para lidar com erros de produção. **Tradicionalmente, as operações são quase alheias à base de código, uma vez que se preocupam apenas com sua infraestrutura.** O objetivo é tornar todo o processo transparente de forma que você possa fazer milhares de implantações de produção por dia; ao contrário dos métodos tradicionais de configuração de uma janela de implantação uma vez a cada trimestre ou por mais tempo.

**Claro que para fazer isso, é necessário utilizar um conjunto de práticas e ferramentas.** *Quais, professor?* Ferramentas de Controle de versão (Git, CVS, Tortoise), Servidores de Integração Contínua (Jenkins, Bamboo, Travis), Docker/Vagrant, Gerenciamento de Configuração (SaltStack, Chef, Puppet). Isso reduz a dor de cabeça tanto para os desenvolvedores quanto para os operadores e reduz a quantidade de problemas de desempenho e erros de codificação.

**(STF – 2013)** Integração contínua, entrega contínua, teste contínuo, monitoramento contínuo e feedback são algumas práticas do DevOps.

_____

**Comentários:** as principais características do DevOps são: colaboração entre equipes; fim de divisões; relação saudável entre áreas; teste, integração e entrega contínuos; automação de deploy; controle e monitoração; gerenciamento de configuração; orquestração de serviços; avaliação de métricas e desempenho; logs e integração; velocidade de entrega; feedback intenso; e comunicação constante (Correto).

**(STF – 2013)** Teste contínuo é uma prática do DevOps que, além de permitir a diminuição dos custos finais do teste, ajuda as equipes de desenvolvimento a balancear qualidade e velocidade.

_____

**Comentários:** teste, integração e entrega contínuos são realmente práticas do DevOps que permitem reduzir custos de teste e ajuda a equipe de desenvolvimento a balancear a qualidade e velocidade (Correto).

**(TCU – 2015)** De acordo com a abordagem DevOps (development – operations), os desafios da produção de software de qualidade devem ser vencidos com o envolvimento

dos desenvolvedores na operação dos sistemas com os quais colaboraram no desenvolvimento.

_____

**Comentários:** essa questão permite duas interpretações de 'envolvimento': (1) no sentido de interação e colaboração entre equipes; (2) no sentido de mão na massa mesmo - na operação dos sistemas. Por conta da ambiguidade, a questão foi anulada (Anulada).

**(TRE-PE – 2017)** O DevOps consiste em:

a) um processo similar ao IRUP (IBM Rational Unified Process), que tem como objetivo dividir o processamento em fases e disciplinas de software para paralelizar as ações de desenvolvimento e de manutenção das soluções.

b) uma plataforma aberta cuja função é substituir a virtualização de aplicações e serviços em containers e, com isso, agilizar a implantação de soluções de software.

c) um aplicativo que permite o gerenciamento de versões de códigos-fonte e versões de programas, bem como a implantação da versão mais recente de um software em caso de falha.

d) um processo de promoção de métodos que objetivam aprimorar a comunicação, tornando a colaboração eficaz especialmente entre os departamentos de desenvolvimento e teste e entre os departamentos de operações e serviço para o negócio.

e) uma metodologia ágil que, assim como a XP (extreme programming) e o Scrum, tem foco na gestão de produtos complexos relativos à equipe de desenvolvimento.

_____

**Comentários:** DevOps não é um processo similar ao iRUP, não é uma plataforma aberta, não é um aplicativo e não é uma metodologia ágil. DevOps é um processo de promoção de métodos que objetivam aprimorar a comunicação, tornando a colaboração eficaz especialmente entre os departamentos de desenvolvimento e teste e entre os departamentos de operações e serviço para o negócio (Letra D).

**(TRT-PA e AP – 2016)** Acerca de DevOps, assinale a opção correta.

a) O DevOps concentra-se em reunir diferentes processos e executá-los mais rapidamente e com mais frequência, o que gera baixa colaboração entre equipes.

b) O DevOps tem como princípio produzir, a partir da avaliação dos times de desenvolvimento do serviço, grandes mudanças e farta documentação com valor agregado para os usuários, assemelhando-se, por isso, com objetivos dos métodos iterativos e em cascata.

c) A infraestrutura de nuvem de provedores internos e externos vem restringindo o uso de DevOps pelas organizações.

d) O DevOps parte da premissa de adoção de grandes equipes de especialistas, com a menor interação possível, visando à padronização de processos e à mínima automação de atividades.

e) Atividades típicas em DevOps compreendem teste do código automatizado, automação de fluxos de trabalho e da infraestrutura e requerem ambientes de desenvolvimento e produção idênticos.

_____

**Comentários:** (a) Errado, isso gera alta colaboração entre equipes; (b) Errado, não se assemelha com objetivos de métodos em cascata, visto que essa metodologia possui entregas menos frequentes e é menos dinâmica; (c) Errado, é justamente o inverso – elas usam com cada vez mais frequência; (d) Errado, recomenda-se a maior interação possível entre as equipes; (e) Correto, testes automatizados, automação de fluxos e infraestrutura são atividades frequentes que realmente exigem ambientes de desenvolvimento e produção idênticos (Letra E).

**(STJ – 2015)** DevOps é um conceito pelo qual se busca entregar sistemas melhores, com menor custo, em menor tempo e com menor risco.

_____

**Comentários:** perfeito, perfeito, perfeito – todas são características do DevOps (Correto).

**(STJ – 2015)** O profissional especialista em DevOps deve atuar e conhecer as áreas de desenvolvimento (engenharia de software), operações e controle de qualidade, além de conhecer, também, de forma ampla, os processos de desenvolvimento ágil.

_____

**Comentários:** ele é a combinação entre desenvolvimento, operação e controle de qualidade – portanto questão perfeita (Correto).

**(SLU-DF – 2019)** Em DevOps, o princípio monitorar e validar a qualidade operacional antecipa o monitoramento das características funcionais e não funcionais dos sistemas para o início do seu ciclo de vida, quando as métricas de qualidade devem ser capturadas e analisadas.

_____

**Comentários:** o princípio monitorar e validar a qualidade operacional transfere o monitoramento para o início do ciclo de vida do software, para identificar antecipadamente problemas de qualidade que podem ocorrer no desenvolvimento. Na implantação e teste, as métricas de qualidade são capturadas e analisadas, sendo que essas métricas devem ser entendidas por todos os stakeholders (Correto).

**(MPE-PI – 2018)** A infraestrutura como código é uma prática DevOps caracterizada pela infraestrutura provisionada e gerenciada por meio de técnicas de desenvolvimento de código e de software, como, por exemplo, controle de versão e integração contínua.

_____

**Comentários:** trata-se realmente de uma prática em que a infraestrutura provisionada e gerenciada por meio de técnicas de desenvolvimento de código e de software – ela oferece controle de versão, entrega e integração contínua (Correto).

**(STJ – 2018)** Apesar de ser um processo com a finalidade de desenvolver, entregar e operar um software, o DevOps é incompatível com a aplicação de métodos ágeis como o Scrum ou, ainda, com o uso de ferramentas que permitam visualizar os fluxos do processo.

_____

**Comentários:** pelo contrário, é completamente compatível com as aplicações de métodos ágeis como o Scrum ou, ainda, com o uso de ferramentas que permitam visualizar os fluxos do processo (Errado).

**(STJ – 2018)** O gerenciamento de desenvolvimento de software por meio do Scrum pode ser combinado com o ciclo de vida do DevOps, haja vista que o DevOps combina práticas e ferramentas que aumentam a capacidade de uma organização de distribuir aplicativos e serviços; logo, a integração contínua do software pode ser realizada na sprint do Scrum junto com a operação dos serviços da organização.

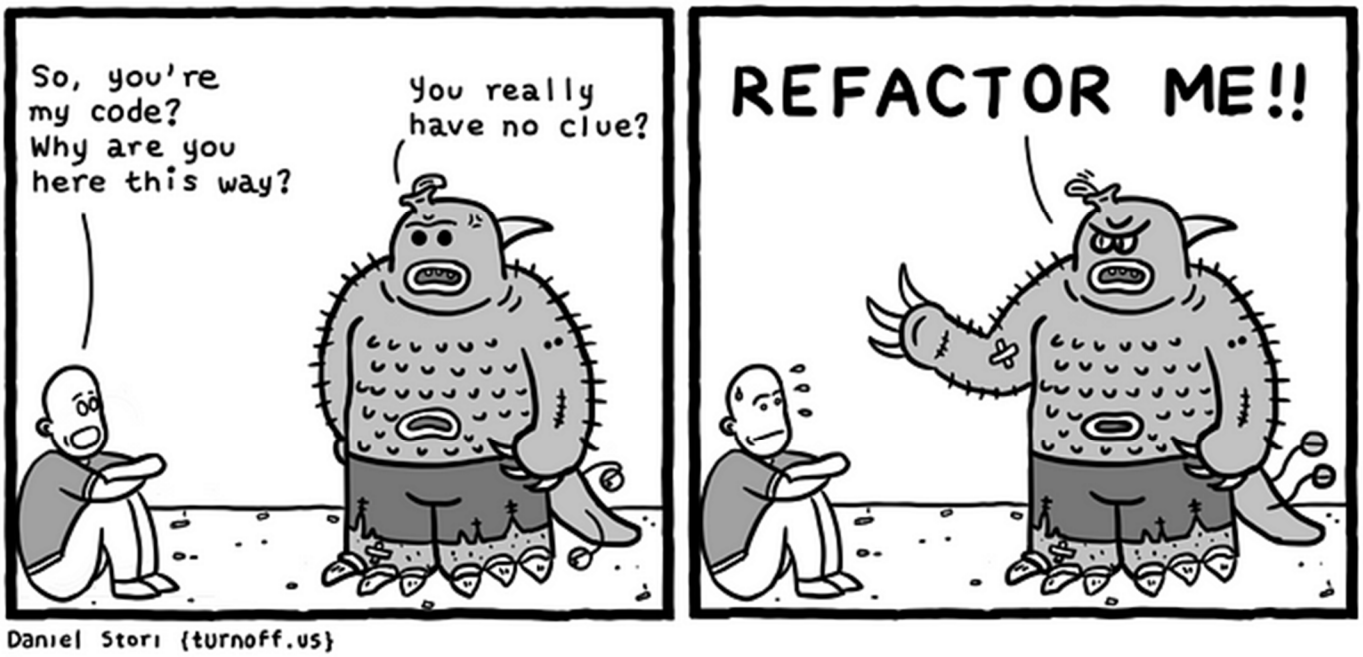
_____

**Comentários:** ele realmente combina práticas e ferramentas que aumentam a capacidade de uma organização de distribuir aplicativos e serviços e pode ser realizado na sprint do Scrum por meio da integração contínua (Correto).

# REFATORAÇÃO

## Conceitos Básicos

INCIDÊNCIA EM PROVA: BAIXA




Uma importante atividade sugerida por diversas metodologias ágeis de desenvolvimento de software, **a refatoração é uma técnica (inclusive preconizada pelo Extreme Programming) de reorganização que simplifica o projeto (ou código) de um componente de software sem modificar sua função ou seu comportamento**. Martin Fowler define refatoração em seu livro da seguinte maneira:

> *"Refatoração é o processo de alterar um sistema de software de modo que o comportamento externo do código não se altere, mas a estrutura interna se aprimore. É uma forma disciplinada de organizar código [e modificar/simplificar o projeto interno] que minimiza as chances de introdução de bugs. Em resumo, ao se refatorar, se está aperfeiçoando o projeto de codificação depois de este ter sido feito.*

Quando um software é refatorado (também chamado de refabricado), o projeto existente é examinado em termos de redundância, elementos de projeto não utilizados, algoritmos ineficientes ou desnecessários, estruturas de dados mal construídas ou inapropriadas, **ou qualquer outra falha de projeto que possa ser corrigida para produzir um projeto melhor**. *Vamos ver um exemplo mais prático?*

Uma primeira iteração de projeto poderia gerar um componente que apresentasse baixa coesão (realizar três funções que possuem apenas relacionamento limitado entre si). Após cuidadosa

consideração, talvez decidamos que o componente devesse ser refabricado em três componentes distintos, cada um apresentando alta coesão. **O resultado será um software mais fácil de se integrar, testar e manter.**

**Dito isso, a refatoração consiste na melhoria da estrutura interna do código-fonte de um programa, enquanto preserva seu comportamento externo**. Prestem bastante atenção: refatoração não é reescrever o código; refatoração não é consertar bugs; refatoração não é melhorar aspectos observáveis do software (Ex: interface). Existem diversos outros benefícios por trás da refatoração...

Ela melhora atributos de código-fonte, tais como tamanho, duplicação, acoplamento, redundância, coesão, complexidade ciclomática, entre outros) que estão relacionados com facilidade de manutenção. **Além disso, ela ajuda a compreensão do código-fonte e encoraja o desenvolvedor a pensar sobre suas decisões de projeto**. *Bacana?* Vamos resumir o que vimos sobre refatoração antes de seguir para outros tópicos...

**Refatoração é uma técnica utilizada na engenharia de software para melhorar a qualidade e a manutenibilidade do código de um programa.** Consiste em modificar o código sem alterar seu comportamento externo, com o objetivo de deixá-lo mais organizado, legível e eficiente. A refatoração é importante porque o código de um programa costuma se tornar cada vez mais complexo e difícil de manter à medida que ele evolui.

A técnica permite que o código seja simplificado e reestruturado, facilitando sua manutenção e evolução. Entre os **benefícios da refatoração**, destacam-se a redução de custos de manutenção, já que um código mais limpo e organizado exige menos tempo e recursos para ser mantido; aumento da produtividade, já que o código é mais fácil de entender e modificar; e redução dos riscos de erros e problemas de segurança, já que o código é mais fácil de auditar e de garantir sua integridade.

Existem alguns princípios fundamentais que orientam a refatoração de software e ajudam a garantir que a técnica seja aplicada de forma segura e eficaz. Alguns dos principais princípios são:

| PRINCÍPIOS | DESCRIÇÃO |
|---|---|
| MANTENHA O COMPORTAMENTO EXTERNO | Isso significa que, ao realizar uma refatoração, é importante garantir que o comportamento externo do programa não seja alterado. Isso é fundamental para que o programa continue funcionando corretamente após a refatoração. |
| SIMPLIFIQUE O CÓDIGO | O objetivo da refatoração é simplificar e melhorar o código do programa. Por isso, é importante eliminar código redundante, simplificar expressões complexas e tornar o código mais legível e fácil de entender. |
| TRABALHE COM TESTES AUTOMATIZADOS | Os testes automatizados são uma parte fundamental da refatoração, pois garantem que o programa continue funcionando corretamente após as mudanças realizadas. É importante criar testes para todas as partes do código que serão modificadas e executá-los antes e depois da refatoração. |

| | |
|---|---|
| REFATORE CONTINUAMENTE | A refatoração é uma atividade contínua, que deve ser realizada ao longo de todo o ciclo de vida do programa. Isso significa que a equipe de desenvolvimento deve estar constantemente procurando maneiras de melhorar o código e simplificá-lo. |
| USE PADRÕES DE PROJETO | Os padrões de projeto são soluções comprovadas para problemas comuns de design de software. Eles podem ser usados para simplificar o código e torná-lo mais fácil de entender e manter. |

Para resumir nosso papo, vamos falar o que é uma refatoração por meio de algumas analogias: digamos que você tenha um guarda-roupa bagunçado, com roupas misturadas e desorganizadas. A refatoração seria como organizar esse guarda-roupa, separando as roupas por tipo, dobrando-as adequadamente e criando categorias para facilitar a busca. Ao refatorar o guarda-roupa, você melhora sua funcionalidade e torna mais fácil encontrar o que precisa.

A minha atividade principal de escrita de aula também passar por refatorações. À medida que eu escrevo, eu vou percebendo que algumas frases podem ser reformuladas para melhorar a clareza ou a coesão do texto de forma que vocês entendam melhor o assunto (essa aula de refatoração mesmo já foi refatorada algumas vezes). Isso é semelhante à refatoração de software, onde você faz alterações no código para torná-lo mais legível, eficiente e fácil de entender.

# Identificação de Oportunidades

INCIDÊNCIA EM PROVA: BAIXA

Imagine que você acabou de se mudar para uma nova casa. No começo, você está feliz com a disposição dos móveis e a forma como a casa foi projetada. **No entanto, à medida que você começa a viver nela, você começa a perceber que alguns móveis estão no lugar errado ou não funcionam como deveriam.** Além disso, há algumas áreas que parecem estar subutilizadas ou mal aproveitadas.

Da mesma forma, quando os desenvolvedores de software criam um sistema, eles o projetam com base em seus conhecimentos e suposições naquele momento. **No entanto, à medida que o sistema é usado, os desenvolvedores podem perceber que algumas partes do código estão causando problemas, tornando-se difíceis de manter ou simplesmente não são mais necessárias.**

**Como resultado, eles identificam oportunidades de refatoração, ou seja, aprimoramentos no design e no código do sistema para melhorar sua qualidade, manutenibilidade e eficiência.** Assim como no exemplo da casa, os desenvolvedores podem utilizar a sua experiência e conhecimentos para identificar áreas problemáticas e oportunidades de melhoria no código-fonte do software.

Eles podem aplicar técnicas de refatoração para reorganizar o código, remover código duplicado, simplificar a lógica e melhorar a estrutura geral do sistema. Isso ajuda a garantir que o software continue atendendo às necessidades dos usuários e da empresa ao longo do tempo. **A identificação de oportunidades de refatoração no código existente é uma atividade importante para garantir a manutenibilidade e a qualidade do software.**

Ao identificar esses sinais de possível refatoração, a equipe de desenvolvimento pode melhorar significativamente a qualidade e a manutenibilidade do código do programa. Vejamos:

| OPORTUNIDADES | DESCRIÇÃO |
|---|---|
| IDENTIFICAÇÃO DE CÓDIGO DUPLICADO | Se houver partes do código que se repetem em vários lugares do programa, pode ser uma oportunidade para refatorar e criar uma função ou classe separada para lidar com essa funcionalidade repetida. |
| MÉTODOS OU FUNÇÕES MUITO LONGOS | Métodos ou funções muito longos podem ser difíceis de entender e modificar. Se um método ou função for muito grande, pode ser uma oportunidade para dividi-lo em partes menores para melhorar a legibilidade e a manutenibilidade. |
| COMPLEXIDADE EXCESSIVA | Se um trecho de código é muito complexo e difícil de entender, pode ser uma oportunidade para simplificá-lo e torná-lo mais fácil de entender. |
| DIFICULDADE P/ ADICIONAR FUNCIONALIDADES | Se for difícil adicionar novas funcionalidades ao código existente, isso pode ser um sinal de que a estrutura do código precisa ser melhorada e simplificada. |

| | |
|---|---|
| CÓDIGO ANTIGO OU LEGADO | Código antigo ou legado pode estar desatualizado e precisar de refatoração para se tornar mais eficiente e manutenível. |
| PROBLEMAS DE PERFORMANCE | Se o código estiver executando lentamente ou apresentando problemas de performance, pode ser uma oportunidade para refatorar e melhorar o desempenho. |

# Técnicas de Refatoração

INCIDÊNCIA EM PROVA: BAIXA

Existem diversas técnicas de refatoração que podem ser aplicadas para melhorar o design e a qualidade do código. Algumas das principais técnicas incluem:

| TÉCNICAS | DESCRIÇÃO |
|---|---|
| EXTRAÇÃO DE MÉTODO | Essa técnica envolve a extração de um trecho de código em um método separado. Isso é útil quando você tem um pedaço de código que é usado várias vezes em diferentes partes do seu programa. Ao extrair o código em um método separado, você pode reutilizá-lo em várias partes do seu programa, tornando o código mais limpo e fácil de manter. |
| MÉTODO INLINE | Essa técnica envolve a substituição de uma chamada de método por seu conteúdo real. Isso pode ser útil em situações em que um método é chamado apenas uma vez e é bastante simples, tornando desnecessária a criação de um método separado para ele. Isso também pode tornar o código mais fácil de entender e depurar. |
| EXTRAÇÃO DE VARIÁVEL | Essa técnica envolve a extração de uma expressão complexa em uma variável separada. Isso pode tornar o código mais legível e mais fácil de entender, especialmente se a expressão for usada várias vezes no programa. A extração de uma variável também pode tornar mais fácil alterar a expressão, pois você só precisa fazer a mudança em um lugar. |
| RENOMEAÇÃO | Essa técnica envolve a alteração do nome de uma variável, método ou classe para um nome mais descritivo e significativo. Isso pode tornar o código mais fácil de entender, especialmente se o nome original não for muito claro ou se for ambíguo. |
| EXTRAÇÃO DE CLASSE | Essa técnica envolve a extração de um conjunto de campos e métodos em uma nova classe separada. Isso pode tornar o código mais organizado e fácil de manter, especialmente se a classe original estiver ficando muito grande. A extração de classe também pode tornar o código mais modular e reutilizável. |
| DIVISÃO DE VARIÁVEL TEMPORÁRIA | Essa técnica envolve a divisão de uma variável temporária em duas ou mais variáveis separadas, cada uma responsável por uma parte específica da variável original. Isso pode tornar o código mais fácil de entender e mais modular, especialmente se a variável original estiver fazendo muitas coisas diferentes. |
| REMOÇÃO DE ATRIBUIÇÕES A PARÂMETROS | Essa técnica envolve a remoção de atribuições a parâmetros de um método. Isso pode tornar o código mais fácil de entender e menos propenso a erros, já que as alterações no parâmetro não afetarão a variável original passada ao método. A remoção de atribuições a parâmetros também pode tornar o código mais fácil de testar, pois não há efeitos colaterais inesperados no parâmetro passado. |
| SUBSTITUIÇÃO DE CÓDIGO DUPLICADO | Essa técnica consiste em identificar trechos de código que se repetem e substituí-los por uma função ou classe separada que possa ser reutilizada. Isso reduz a quantidade de código e torna o programa mais fácil de manter. |

| | |
|---|---|
| ENCAPSULAMENTO DE DADOS | Essa técnica consiste em tornar as variáveis privadas e criar métodos de acesso (*getters* e *setters*) para manipulá-las. Isso ajuda a garantir a integridade dos dados e torna o código mais fácil de entender e modificar. |
| DECOMPOSIÇÃO DE CLASSES | Essa técnica consiste em identificar classes que estão realizando muitas funcionalidades diferentes e dividi-las em classes menores e mais específicas. Isso ajuda a tornar o código mais modular e reutilizável. |
| GENERALIZAÇÃO DE CÓDIGO | Essa técnica consiste em identificar trechos de código que realizam a mesma funcionalidade em diferentes partes do programa e criar uma classe ou função genérica que possa ser reutilizada. Isso reduz a quantidade de código e torna o programa mais fácil de manter. |

# Teste de Refatoração

INCIDÊNCIA EM PROVA: BAIXA

Imagine que você tenha um carro antigo que precisa de alguns consertos e algumas melhorias para continuar funcionando bem. Para fazer isso, você leva seu carro a um mecânico especializado em carros antigos. O mecânico verifica as diferentes partes do carro e identifica as áreas que precisam ser atualizadas ou substituídas. **Antes de realizar essas atualizações, o mecânico faz alguns testes no carro para ter certeza de que o motor está funcionando corretamente.**

Ele testa se as luzes estão funcionando adequadamente e se o carro pode ser conduzido com segurança. Esses testes são necessários para garantir que o carro não apresente nenhum problema que possa colocar o motorista ou outras pessoas em risco. **Da mesma forma, os testes de refatoração são feitos em software para garantir que as atualizações e melhorias não tenham efeitos colaterais indesejados.**

Os testes verificam se o software ainda atende aos requisitos originais e se todas as funcionalidades ainda estão funcionando conforme o esperado. **Eles ajudam a garantir que o software seja seguro e confiável após as atualizações.** Assim como o mecânico verifica o carro antes e depois de fazer as atualizações, os desenvolvedores de software fazem testes antes e depois de realizar as refatorações.

Isso ajuda a garantir que o software continue a ser confiável e eficaz, mesmo após as atualizações e melhorias. Logo, testar a refatoração é uma etapa crucial, pois **garante que as alterações realizadas no código não introduzam novos erros e que o programa continue funcionando corretamente. Independentemente da abordagem escolhida**, é importante garantir que o teste de refatoração seja realizado de forma completa e rigorosa para garantir a integridade do código.

| TESTE DE REFATORAÇÃO | DESCRIÇÃO |
|---|---|
| TESTE MANUAL | essa abordagem envolve a execução manual do programa após a refatoração para verificar se todas as funcionalidades ainda estão funcionando corretamente. Essa abordagem é útil para testar pequenas alterações, mas pode ser demorada e propensa a erros. |
| TESTE DE REGRESSÃO | essa abordagem envolve a criação de uma suíte de testes automatizados antes da refatoração e a execução desses testes após a refatoração para verificar se todas as funcionalidades ainda estão funcionando corretamente. Essa abordagem é mais eficiente e menos propensa a erros do que o teste manual. |
| TESTE DE UNIDADE | essa abordagem envolve a criação de testes de unidade para as partes do código que foram alteradas durante a refatoração. Esses testes são executados automaticamente para garantir que as alterações não introduziram novos erros. Essa abordagem é mais precisa e eficiente do que o teste manual ou de regressão. |

Além disso, é importante documentar todas as alterações realizadas durante a refatoração e manter um histórico de versões do código para facilitar a manutenção e evolução do software.

# Refatoração Contínua

INCIDÊNCIA EM PROVA: BAIXA

Um conceito inovador é a refatoração contínua, que é uma prática que envolve incorporar a refatoração no processo de desenvolvimento contínuo de software, de forma a manter o código limpo, organizado e fácil de manter. **A ideia é que, ao invés de realizar grandes refatorações em momentos pontuais do projeto, a equipe realize pequenas refatorações constantemente, à medida que trabalha no código.**

Para incorporar a refatoração contínua no processo de desenvolvimento, é importante que a equipe de desenvolvimento tenha um bom entendimento dos princípios de refatoração, bem como das técnicas e ferramentas disponíveis. **Além disso, é importante que a equipe esteja comprometida em manter o código limpo e fácil de manter, e que estejam dispostos a dedicar tempo para realizar pequenas refatorações ao longo do desenvolvimento.**

Algumas práticas comuns para incorporar a refatoração contínua no processo de desenvolvimento incluem:

- Realizar pequenas refatorações frequentemente, sempre que for identificado um código duplicado, um método muito grande ou uma classe mal organizada, por exemplo.
- Utilizar ferramentas de refatoração para facilitar o processo de refatoração contínua, automatizando tarefas repetitivas e tornando o processo mais eficiente.
- Integrar a refatoração no processo de revisão de código, de forma que o código refatorado seja revisado e testado antes de ser incorporado ao código-base.
- Utilizar testes automatizados para validar as refatorações, de forma a garantir que as alterações não tenham introduzido novos erros ou comportamentos inesperados no código.

Com a refatoração contínua, a equipe de desenvolvimento pode manter o código sempre limpo e organizado, reduzindo a quantidade de tempo e esforço necessários para realizar grandes refatorações em momentos pontuais do projeto. **Além disso, um código bem organizado e fácil de manter pode ajudar a reduzir os custos de manutenção do software, melhorar a qualidade do produto final e aumentar a satisfação do usuário.**

Existem algumas práticas recomendadas que podem ajudar a maximizar os benefícios da refatoração e minimizar os riscos envolvidos no processo. Algumas dessas práticas incluem:

| PRÁTICAS RECOMENDADAS | DESCRIÇÃO |
|---|---|
| PLANEJE A REFATORAÇÃO | Antes de começar a refatorar, planeje cuidadosamente o que você quer fazer e como irá fazer. Isso pode incluir a identificação das áreas que precisam de refatoração, a escolha |

| | |
|---|---|
| | das técnicas de refatoração apropriadas e a elaboração de um plano para minimizar os riscos. |
| REFATORE EM PEQUENOS PASSOS | Refatorar em pequenos passos ajuda a minimizar os riscos envolvidos no processo, permitindo que você teste as alterações e valide-as antes de avançar para a próxima etapa. Além disso, isso também ajuda a manter o código sempre funcional, evitando que o processo de refatoração cause interrupções no desenvolvimento. |
| USE TÉCNICAS DE VERSIONAMENTO | Antes de iniciar a refatoração, crie uma nova branch do seu código e trabalhe nela. Dessa forma, você pode testar e validar as alterações sem afetar a versão principal do código, garantindo que você possa reverter facilmente para uma versão anterior, caso algo dê errado. |
| TESTE RIGOROSAMENTE | Testar rigorosamente é fundamental para garantir que a refatoração não introduza novos erros ou comportamentos inesperados no código. Certifique-se de criar testes automatizados para validar as alterações e execute testes manuais sempre que possível. |
| ENVOLVER TODA A EQUIPE | Refatorar é um trabalho que deve ser feito em equipe. Envolver todos os membros da equipe pode ajudar a identificar áreas que precisam de refatoração, garantir que as alterações sejam compreendidas por todos e ajudar a minimizar os riscos envolvidos no processo. |
| APRENDER E PRATICAR CONSTANTEMENTE | Refatoração é uma habilidade que requer prática constante. Aprenda novas técnicas de refatoração, pratique sempre que possível e busque feedback da equipe para melhorar continuamente. |

# Estudos de Caso

INCIDÊNCIA EM PROVA: BAIXA

Existem estudos de caso de refatoração que mostram como a aplicação de técnicas de refatoração pode melhorar significativamente a qualidade do código e a eficiência do desenvolvimento:

| ESTUDOS DE CASO | DESCRIÇÃO |
|---|---|
| REFATORAÇÃO DO CÓDIGO-FONTE DO ECLIPSE | O Eclipse é uma plataforma de desenvolvimento popular que é construída em Java. Em 2005, o time de desenvolvimento do Eclipse realizou uma grande refatoração do código-fonte para melhorar sua qualidade. Eles usaram diversas técnicas de refatoração para simplificar e limpar o código, e conseguiram reduzir o tamanho do código em mais de 20%. |
| REFATORAÇÃO DO CÓDIGO DO KENT BECK | Kent Beck, um dos criadores do Extreme Programming (XP), escreveu um livro chamado "Test Driven Development: By Example". Neste livro, ele descreveu como refatorou um sistema bancário legado para melhorar sua qualidade e desempenho. Ele usou diversas técnicas de refatoração para simplificar o código, remover duplicação e melhorar a estrutura do sistema. |
| REFATORAÇÃO DO CÓDIGO DO LINKEDIN | O LinkedIn, uma rede social para profissionais, realizou uma grande refatoração do seu código em 2014. Eles usaram diversas técnicas de refatoração para melhorar a qualidade do código e torná-lo mais fácil de manter. Como resultado, eles conseguiram reduzir o número de erros e melhorar significativamente a performance do site. |
| REFATORAÇÃO DO CÓDIGO DO GOOGLE | O Google é conhecido por sua abordagem rigorosa para a qualidade do código. Eles realizam regularmente grandes refatorações em seus sistemas para melhorar sua qualidade e desempenho. Por exemplo, em 2006, eles realizaram uma grande refatoração no Google Maps, que melhorou significativamente sua performance e tornou-o mais fácil de manter. |

# Reengenharia x Refatoração

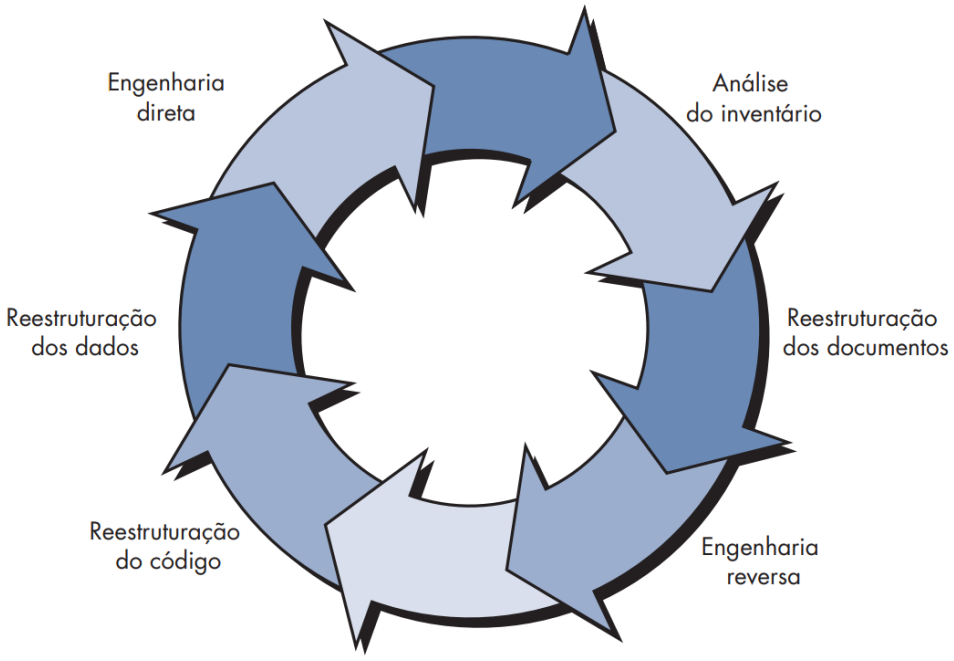
INCIDÊNCIA EM PROVA: BAIXA

Por fim, é importante diferenciar a refatoração de reengenharia. Pressman explica assim: imaginem que um aplicativo atendeu às necessidades de negócio de uma empresa por dez ou quinze anos. **Durante esse tempo, ele foi corrigido, adaptado e aperfeiçoado em diversas oportunidades.** Profissionais realizaram este trabalho com as melhores intenções, mas as boas práticas de engenharia de software foram sempre deixadas de lado (devido à pressão por outros aspectos).

Agora o aplicativo está instável. **Ainda funciona, mas sempre que se tenta fazer uma alteração, ocorrem efeitos colaterais sérios e inesperados. No entanto, o aplicativo deve continuar evoluindo.** *O que fazer?* Software que não pode ser mantido não é novidade. Na verdade, a ênfase cada vez maior sobre a engenharia de software foi motivada pelos problemas de manutenção criados por mais de quatro décadas.

**Reengenharia toma tempo, tem um custo significativo em dinheiro e absorve recursos que poderiam de outra forma ser usados em necessidades mais imediatas.** Por todas essas razões, a reengenharia não é realizada em alguns meses ou mesmo anos. A reengenharia dos sistemas de informação é uma atividade que absorverá recursos da tecnologia da informação por muito tempo, logo todas as organizações precisam de uma estratégia pragmática para reengenharia de software

Segue abaixo um modelo de processo de reengenharia de software. Em alguns casos, essas atividades ocorrem em sequência linear, mas nem sempre é o caso. Por exemplo: pode acontecer de a engenharia reversa ter de ocorrer antes do início da reestruturação dos documentos. Deve-se realizar uma análise de inventário, isto é, toda organização de software deve ter um inventário de todos os aplicativos com informações detalhadas (tamanho, idade, criticidade no negócio, etc).



Assim que surgem candidatos à reengenharia. **Deve ocorrer uma reestruturação dos documentos – isso não significa que se deve documentar absolutamente tudo.** No entanto, há um mínimo necessário e, de preferência, deve estar atualizado. A engenharia reversa daria um livro inteiro, mas trata do processo de analisar um programa na tentativa de criar uma representação do programa em um nível mais alto de abstração do que o código-fonte.

É como se eu pegasse um aplicativo pronto e chegasse em sua documentação de análise, projeto, requisitos, entre outros. Em seguida, temos a reestruturação do código, que reestrutura ou reescreve o código em uma tecnologia mais moderna ou de forma mais eficiente, preservando a funcionalidade e a arquitetura global. Depois, temos a reestruturação dos dados, em que se identificam objetos de dados, e as estruturas de dados existentes são revisadas quanto à qualidade.

**Por fim, a engenharia direta recupera as informações do projeto de software existente e usa as informações para alterar ou reconstituir o sistema existente em um esforço para melhorar sua qualidade geral.** Em muitos casos, o software que passou pela reengenharia reimplementa a função do sistema existente e também acrescenta novas funções e/ou melhora o desempenho geral da aplicação ou componente.

**Reengenharia (também chamada Manutenção Preventiva) é, portanto, o exame, análise e reestruturação de um sistema de software existente para reconstituí-lo em uma nova forma.** O objetivo da reengenharia é: compreender os atuais artefatos de software, isto é, especificação, projeto, implementação e documentação; e melhorar a funcionalidade e qualidade de atributos do sistema.

**Alguns exemplos de atributos de qualidade são:** capacidade de evolução, desempenho e capacidade de reutilização. Fundamentalmente, um novo sistema é gerado a partir de um sistema operacional, de tal modo que o sistema de destino possua melhores fatores de qualidade, como confiabilidade, exatidão, integridade, eficiência, facilidade de manutenção, facilidade de utilização, flexibilidade, capacidade de teste, a interoperabilidade, reusabilidade e portabilidade.

**Em outras palavras, reengenharia é feita para converter um sistema de "ruim" em um sistema de "bom".** Claro que existem riscos envolvidos nesta transformação (Ex: o sistema novo não pode ter qualidade inferior ao anterior). Sistemas de software são redesenhados, mantendo um ou mais dos quatro objetivos gerais seguintes: melhorar manutenibilidade; migração para uma nova tecnologia; melhorar a qualidade do software; e preparar para melhorias funcionais.

**Uma boa compreensão dos processos de desenvolvimento de software é útil em fazer um plano de reengenharia. Vários conceitos aplicados durante o desenvolvimento de software são fundamentais para a reengenharia.** Por exemplo: abstração e requinte são os principais conceitos utilizados no desenvolvimento de software, e ambos os conceitos são igualmente úteis na reengenharia.

O princípio da abstração afirma que o nível de abstração da representação de um sistema pode ser aumentado gradualmente substituindo sucessivamente os detalhes com informações abstratas, isto é, retirando detalhes menos importantes. **Por meio da abstração, pode-se produzir uma visão que incide sobre as características selecionadas do sistema ao esconder informações sobre outras características.**

Já o princípio do refinamento afirma que o nível de abstração da representação do sistema é gradualmente reduzido por sucessivas substituições de alguns aspectos do sistema com mais detalhes. Enfim, galera... a definição que eu mais gosto é do Sommerville e diz assim: **reengenharia é a reorganização e modificação de sistemas de software existentes, parcial ou totalmente, para torná-los mais manuteníveis.**

# QUESTÕES COMENTADAS – DIVERSAS BANCAS

1. **(FUNDATEC / BRDE - 2023)** Avalie as assertivas abaixo:

I. Melhoria do design interno (arquitetura) do software.
II. Código mais legível.
III. Localização de bugs.
IV. Mudança do comportamento externo do software.

Quantas podem vir a ser benefícios do processo de refatoração de código?

a) 0
b) 1
c) 2
d) 3
e) 4

**Comentários:**

Podem ser benefícios do processo de refatoração de código:

I. Melhoria do design interno (arquitetura) do software.
II. Código mais legível.
III. Localização de bugs.

A assertiva IV não é um benefício da refatoração, dado que ela busca alterar um software de uma maneira que não mude o seu comportamento externo e ainda melhore a sua estrutura interna.

**Gabarito:** Letra D

---

2. **(QUADRIX / PRODAM-AM - 2022)** Assinale a alternativa que apresenta a prática de XP (Extreme Programming) que é definida como uma técnica disciplinada para reestruturar um corpo de código existente, alterando a sua estrutura interna sem alterar seu comportamento externo. Essa prática mantém a semântica do código, ou seja, após as mudanças, o código ainda funciona da mesma forma.

a) refatoração
b) metáfora
c) *test-driven development*
d) *coding standards*
e) *on-site customer*.

**Comentários:**

A prática descrita na questão é a refatoração. A refatoração é uma prática da metodologia XP (Extreme Programming) que consiste em melhorar a estrutura interna do código sem alterar o seu comportamento externo, mantendo a sua semântica. A refatoração é uma técnica disciplinada que ajuda a manter o código limpo, legível, fácil de manter e evoluir.

**Gabarito:** Letra A

---

3. **(FUNDEP / UFJF - 2022)** Considere o trecho de código a seguir, que acabou de ser refatorado.

```
delta = b*b-4*a*c; // nova variável
x1 = (-b + sqrt(delta)) / (2*a);
x2 = (b + sqrt(delta)) / (2*a);
```

Assinale a refatoração aplicada para essa situação.

a) Inline de método.
b) Extração de método.
c) Extração de variável.
d) Renomeação.
e) Extração de classe.

**Comentários:**

A refatoração aplicada para esse trecho de código foi a extração de variável. Foi criada uma nova variável chamada **delta** para armazenar o resultado do cálculo **b*b-4*a*c**. Essa refatoração tem como objetivo tornar o código mais legível e fácil de entender, além de evitar a repetição do cálculo em ambos os cálculos de raízes.

**Gabarito:** Letra C

---

4. **(CESPE / BANRISUL - 2022)** A filosofia da modelagem ágil não admite decisões que levem um projeto a sua rejeição e a sua refatoração, pois os profissionais de tecnologia não possuem todas as respostas e outros stakeholders envolvidos no negócio devem ser respeitados e integrados ao processo.

**Comentários:**

Opa! A filosofia da modelagem ágil, ao contrário do que foi afirmado, considera que é normal e esperado que as decisões tomadas possam levar a ajustes, refatorações ou até mesmo rejeição de um projeto, pois o objetivo é desenvolver soluções iterativamente e de forma incremental, com a participação dos stakeholders e validações constantes. A metodologia ágil valoriza a

adaptabilidade e a flexibilidade para lidar com mudanças, por isso a refatoração é vista como uma prática importante para melhorar a qualidade do software.

**Gabarito:** Errado

---

5. **(CESPE / MC - 2022)** Refatorar um *software* consiste em modificar o seu comportamento interno e externo, mantendo-se inalterada sua estrutura interna.

**Comentários:**

Opa! Refatorar um software consiste em melhorar sua estrutura interna sem alterar o comportamento externo, mantendo a mesma semântica e funcionalidade do software. Ou seja, a refatoração não deve modificar o comportamento externo do software, pois isso poderia afetar a sua integridade e desestabilizar a aplicação.

**Gabarito:** Errado

---

6. **(CONSULPAM / Prefeitura de Irauçuba - CE - 2022)** Dentro das metodologias ágeis, o processo de desenvolvimento de software especificado pela Programação Extrema (eXtreme Programming, XP) possui algumas características específicas. Uma das características do XP versa sobre as necessidades de melhoria no projeto, que devem ser realizadas através de um tipo de processo específico para este fim. Assinale a alternativa com o nome deste tipo de processo.

a) Testes.
b) Refatoração.
c) Histórias do Usuário.
d) Programação em Pares.

**Comentários:**

A característica descrita na questão refere-se à prática da refatoração na metodologia XP (Extreme Programming). A refatoração é um processo específico para a melhoria contínua do projeto de software, que consiste em reestruturar o código sem alterar seu comportamento externo, mantendo a semântica e funcionalidade do software.

**Gabarito:** Letra B

---

7. **(IDECAN / IF-CE - 2021)** No que diz respeito à manutenção e reengenharia de software, um termo define o processo de alterar o código-fonte, de modo que não altere o comportamento externo e ainda melhore a sua estrutura interna. É uma técnica disciplinada de limpar e organizar o código, e por consequência, minimizar a chance de introduzir novos bugs. Esse termo é conhecido como

a) elicitação.
b) refatoração.
c) recodificação.
d) replicação.

**Comentários:**

O termo que define o processo de alterar o código-fonte, de modo que não altere o comportamento externo e ainda melhore a sua estrutura interna é a refatoração. A refatoração é uma técnica disciplinada de limpar e organizar o código, a fim de torná-lo mais legível, fácil de entender e de manter, além de minimizar a chance de introduzir novos bugs.

**Gabarito:** Letra B

---

8. **(IBADE / Prefeitura de Vilhena - 2019)** Em Orientação a Objetos define-se o processo de Refatoração como:

a) processo de criação de entidades com herança e polimorfismo.
b) classe abstrata que permite a criação de métodos fatorados.
c) classe que permite herdar da classe pai todos os seus objetos.
d) processo para melhorar a legibilidade ou a eficiência do código, preservando seu comportamento.
e) objeto que permite acessar classes fatoradas.

**Comentários:**

Em Orientação a Objetos, o processo de Refatoração é definido como o processo para melhorar a legibilidade ou a eficiência do código, preservando seu comportamento. A Refatoração é uma técnica disciplinada de limpar e organizar o código, reestruturando sua estrutura interna, sem alterar seu comportamento externo, mantendo a semântica e funcionalidade do software.

**Gabarito:** Letra D

---

9. **(CESPE / MPC-PA - 2019)** No contexto da manutenção preventiva de sistemas no paradigma orientado a objetos, a refatoração é uma técnica empregada com o objetivo de:

a) adicionar uma nova funcionalidade no sistema.
b) incluir uma generalidade que possa ser necessária no futuro.
c) realizar a reengenharia de um sistema.
d) reduzir o risco da introdução de novos erros no programa.
e) substituir métodos similares em subclasses por um único método em uma superclasse.

**Comentários:**

(a) Errado. A refatoração não tem como objetivo adicionar novas funcionalidades ao sistema. Ela se concentra em melhorar a qualidade do código existente, sem adicionar novas funcionalidades;

(b) Errado. Desconheço o sentido de generalidade apresentada no texto desse item. De toda forma, a refatoração busca melhorar a qualidade do código;

(c) Errado. Refatoração é diferente de reengenharia – a primeira não busca mudar o comportamento externo do software; já a segunda, sim;

(d) Correto. Ao melhorar a qualidade do código existente, tornando-o mais legível, eficiente e fácil de entender, a refatoração ajuda a prevenir a introdução de novos erros no programa.

(e) Correto. A substituição de métodos similares em subclasses por um único método em uma superclasse é uma técnica que pode ser aplicada para fazer a refatoração.

A banca considerou o item (d) como errado, mas eu discordo. Após a refatoração, é interessante fazer testes de regressão para verificar se novos erros não foram inseridos. Logo, é claro que a refatoração ajuda a reduzir o risco da introdução de novos erros por conta da melhoria da qualidade do software. Enfim, divirjo respeitosamente da banca e acredito que a questão deveria ser anulada.

**Gabarito:** Letra E

---

**10. (IF-PA / IF-PA - 2019)** Ao analisarmos uma classe Java, nos deparamos com um método que implementa diversas funcionalidades, tornando-se um método com muitas linhas de código, de difícil compreensão e manutenção. Para melhorar essa situação, decidimos dividi-lo em métodos menores, mais fáceis de entender e de efetuar manutenções. A esse processo de organizar e melhorar a estrutura interna de uma aplicação, denominamos de:

a) indentação.
b) depuração.
c) inspeção.
d) integração.
e) refatoração.

**Comentários:**

A esse processo de organizar e melhorar a estrutura interna de uma aplicação, dividindo um método em métodos menores, mais fáceis de entender e de efetuar manutenções, denominamos de refatoração. A indentação se refere à organização do código para melhorar sua legibilidade e compreensão. Depuração é o processo de encontrar e corrigir erros em um programa. A inspeção é uma atividade de revisão do código realizada por um ou mais desenvolvedores. A integração se

refere ao processo de combinar diferentes partes de um sistema de software para formar um todo coeso.

**Gabarito:** Letra E

---

**11. (CESPE / SLU – 2019)** Refactoring (refatoração) é o processo utilizado para reescrever aplicações desatualizadas, com a finalidade de incrementar e melhorar suas funcionalidades; o uso dessa técnica normalmente aprimora aplicações para disponibilizá-las na Internet.

**Comentários:**

A refatoração é uma técnica de reorganização que simplifica um projeto, sem modificar suas funções ou seu comportamento. Logo, a questão erra ao afirmar que a finalidade é incrementar e melhorar as funcionalidades – isso não é possível na refatoração, de modo que o comportamento deve ser preservado.

**Gabarito:** Errado

---

**12. (AOCP / PRODEB – 2018)** "Processo de alteração de um sistema de software de tal forma que não se altere o comportamento externo do código, mas se aprimore a estrutura interna". O enunciado se refere a:

a) Reúso.
b) Teste Unitário.
c) Teste de Integração.
d) Refatoração.
e) Prototipação.

**Comentários:**

(a) Errado, o reúso é uma estratégia para reutilização de um software existente; (b) Errado, o teste unitário é um teste que foca na lógica interna de processamento; (c) Errado, o teste de integração verifica se os componentes de um sistema trabalham corretamente juntos; (d) Correto, trata-se da refatoração; (e) Errado, a prototipação se baseia na ideia de construção de um protótipo (esboços, desenhos ou imagens) que auxiliem a construção do produto.

**Gabarito:** Letra D

---

**13. (CS-UFG / Câmara de Goiânia – 2018)** Sejam as classes A e B tais que o relacionamento entre elas é dado pelo fato de A usar (referenciar) a classe B. Dessa forma, qual das refatorações a seguir implementa o princípio da inversão de dependência?

a) Cria interface para serviços oferecidos por B; a classe A passa a usar a interface criada; a classe B passa a implementar a interface criada; a classe A não usa mais a classe B.

b) Cria interface para serviços oferecidos por A; a classe A passa a implementar a interface criada; a classe B passa a usar a interface criada; a classe A não usa mais a classe B.

c) Cria um relacionamento de herança entre as classes A e B (A torna-se uma especialização de B); métodos da classe B empregados pela classe A são migrados para a classe A; a classe A não usa mais a classe B.

d) Cria uma referência para a classe B na classe A; cria um método para receber uma instância de B (injeção de dependência) e guarda-a na referência criada; a classe A não usa mais a classe B.

**Comentários:**

Basicamente o princípio da inversão de dependência consiste em: Uma interface deve ser projetada pela classe que a usará, e não pela classe que a implementará. Além disso, ele afirma que: (1) Módulos de alto nível não devem depender de módulos de baixo nível. Ambos devem depender de abstrações; (2) Abstrações não devem depender de detalhes. Os detalhes devem depender das abstrações. Dito isso, vamos analisar as alternativas.

(a) Correto, pois a classe A não irá mais depender da classe B, ela irá usar a interface criada; (b) Errado, pois B não pode depender da classe A; (c) Errado, pois deve-se utilizar uma interface; (d) Errado, pois a classe B não pode depender da classe A.

**Gabarito:** Letra A

---

**14. (FAURGS / TJ-RS – 2018)** Em relação à refatoração, assinale com V (verdadeiro) ou F (falso) as afirmações abaixo.

( ) O melhor momento para se refatorar um código é durante os testes de aceitação, pois o cliente tem interesse em um código de qualidade.

( ) Um dos passos da refatoração é a aplicação dos testes que verificarão sua implementação.

( ) Rotinas muito longas e código duplicado são exemplos de bad smells.

( ) Refatorações são modificações no código que são simples a ponto de não gerarem nenhum efeito prático.

( ) Um código que já foi refatorado uma vez não precisará ser refatorado no futuro, pois já atende aos critérios de qualidade exigidos.

( ) A refatoração de um código implica apenas a melhoria de sua qualidade interna e não deve afetar sua funcionalidade original.

A sequência correta de preenchimento dos parênteses, de cima para baixo, é

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

**Comentários:**

(F) Errado, não é feita durante os testes de aceitação, mas sim durante os testes unitários; (V) Correto, a refatoração por realizar mudanças na estrutura interna do código necessita de uma bons testes de implementação, principalmente os testes de unidade; (V) Correto, bas smell é uma situação no qual a estrutura do programa pode ser melhorada com refatoração. Ademais, rotinas longas e códigos duplicados devem ser melhorados com a refatoração; (F) Errado, nem sempre as refatorações são simples; (F) Errado, a refatoração deve ser feita sempre que necessário; (V) Correto, a refatoração não deve alterar o comportamento do software.

**Gabarito:** Letra C

---

**15. (CEPS-UFPA / UFPA – 2018)** Acerca do tema refatoração de software, considere as afirmativas.

I A refatoração busca evoluir o projeto e código-fonte de um sistema de software para se alcançar alta coesão, isto é, suas classes devem possuir conjuntos extensos de responsabilidades.

II A refatoração busca evoluir o projeto e código-fonte de um sistema de software para alcançar baixo acoplamento, isto é, a colaboração entre as classes deve ser mantida em um nível mínimo aceitável.

III A refatoração é o processo de mudar um sistema de software de tal forma que não altere o comportamento externo do código-fonte, embora melhore sua estrutura interna.

Está(ão) correta(s)

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

**Comentários:**

(I) Errado, de fato, a refatoração busca que se tenha mais coesão no software, mas ela não busca a necessidade de se possuir conjuntos extensos de responsabilidade; (II) Correto, a refatoração busca baixo acoplamento e alta coesão; (III) Correto, é a definição perfeita de refatoração.

**Gabarito:** Letra E

---

**16. (CESPE / STJ – 2018)** A refatoração de um código escrito em Delphi pode levar um método a ser separado e transformado em alguns outros métodos.

**Comentários:**

Perfeito! A refatoração consiste na melhoria da estrutura interna do código-fonte de um programa, enquanto preserva seu comportamento externo. Essa melhoria na estrutura interna pode gerar a separação e a transformação de métodos.

**Gabarito:** Correto

---

**17. (COPERVE-UFSC / UFSC – 2018)** Considere os seguintes exemplos de procedimentos de manutenção, no contexto da necessidade de alteração de um programa hipotético de controle acadêmico de cursos de graduação da UFSC:

I. fazer com que o resultado da matrícula passe a ter a opção de gerar o resultado em formato PDF, além da atual possibilidade de informar na tela;

II. incluir funcionalidade para permitir que o trancamento de matrícula possa ser feito on-line;

III. reorganização da hierarquia de herança das classes do programa;

IV. criar classes no programa;

V. remover classes do programa;

Assinale a alternativa que relaciona apenas procedimentos de manutenção que podem ser classificados como ações de refatoração (refactoring).

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

**Comentários:**

Vamos relembrar o conceito de refatoração: a refatoração consiste na melhoria da estrutura interna do código-fonte de um programa, enquanto preserva seu comportamento externo.

(I) Errado, a refatoração não permite a incorporação de novas funcionalidades; (II) Errado, na refatoração não se pode incluir novas funcionalidades; (III) Correto, essa reorganização preserva o comportamento externo do software; (IV) Correto, trata-se de uma reorganização interna, sem afetar o comportamento externo; (V) Correto, também se trata de uma reorganização interna.

**Gabarito:** Letra A

---

**18. (CESPE / CGM-JP – 2018)** A refatoração recomendada pela metodologia XP consiste na reorganização interna do código-fonte sem alteração no seu comportamento, o que permite melhorias no projeto, mesmo após o início da implementação.

**Comentários:**

Perfeito! A refatoração consiste na melhoria da estrutura interna do código-fonte de um programa, enquanto preserva seu comportamento externo.

**Gabarito:** Correto

---

**19. (CESPE / DPE-RO – 2021)** No contexto das metodologias ágeis, o conceito de refatoração compreende:

a) o desenvolvimento de testes incrementais a partir de novos cenários.
b) a renomeação de atributos e métodos para implementar melhorias no software.
c) a decomposição de histórias de usuário em uma série de tarefas de desenvolvimento.
d) a substituição do resultado de uma sprint inteira para atender a requisitos diferentes dos originais.
e) a junção de alterações de código às funcionalidades do software já entregue.

**Comentários:**

A refatoração é o processo de alteração de um sistema de software de tal forma que não se altere o comportamento externo do código, mas se aprimore a estrutura interna. A alternativa (e) pode gerar dúvida, mas na refatoração não há – de nenhuma forma – alterações nas funcionalidades do software. As demais alternativas não fazem sentido...

**Gabarito:** Letra B

---

**20. (FCC / DPE-SP – 2010)** A refatoração é o processo de modificar um sistema de software para melhorar a estrutura interna do código sem alterar seu comportamento externo.

**Comentários:**

Perfeito! Refatoração consiste na melhoria da estrutura interna do código-fonte de um programa, enquanto preserva seu comportamento externo.

**Gabarito:** Correto

---

**21. (CESPE / CENSIPAM – 2006)** A refatoração modifica a estrutura interna de um software visando facilitar o entendimento e as futuras modificações sem alterar o comportamento apresentado pelo software. Não é uma prática que possa ser aplicada em processos de desenvolvimento ágeis, pois requer a construção de modelos tanto para o projeto de alto nível quanto para o projeto detalhado.

**Comentários:**

A segunda sentença está incorreta – ele tanto é aplicado que é explicitamente referenciado pelo XP.

**Gabarito:** Errado

---

**22. (CESPE / CENSIPAM – 2006)** A refatoração é aplicável quando são identificados fragmentos de código que podem ser agrupados, expressões complicadas, atributos acessados mais por outras classes que pelas classes das quais são membros, enunciados condicionais complexos, códigos duplicados, longos métodos, longas classes, muitos parâmetros, métodos ou classes pouco usadas.

**Comentários:**

Perfeito! Entre os benefícios esperados, podemos afirmar que a refatoração melhora atributos objetos de código (tamanho, duplicação, acoplamento, coesão, complexidade ciclomática, entre outros) que estão relacionados com facilidade de manutenção. Além disso, ele ajuda a compreensão do código-fonte e encoraja o desenvolvedor a pensar sobre suas decisões de projeto

**Gabarito:** Correto

---

**23. (CESPE / INMETRO – 2009)** As técnicas de refatoração de código compreendem, entre outras, a remoção de números mágicos e a introdução de padrões de desenho.

**Comentários:**

Perfeito! A refatoração é a reorganização interna do código, logo bastante ligada aos padrões de projeto. Já os números mágicos são aqueles não tem um significado documentado e esclarecido no

código e que, em geral, não estão atribuídos diretamente a uma variável. Eles foram ficando por conta de modificações no sistema e atualmente ninguém sabe o que eles significam.

Exemplo: Suponha que um programa de cálculo trigonométrico faça uso do número π em diversos lugares. A princípio o programador usou a aproximação 3.14 e a colocou numericamente em todos os lugares que ela era necessária. O número 3.14 a princípio é facilmente reconhecível como π por qualquer pessoa com algum conhecimento de matemática. Porém nos testes o programador descobriu que precisaria de uma aproximação melhor, como 3.1415926. Agora ele tem que procurar todas as ocorrências de 3.14 no programa e substituí-la pela nova aproximação. Este procedimento é trabalhoso e sujeito a erros. Se o programador tivesse usado uma constante com o nome PI, em vez do número mágico 3.14, bastaria mudar a aproximação na definição da constante.

**Gabarito:** Correto

---

**24. (CESPE / INMETRO – 2010)** A técnica de refatoração, utilizada no paradigma de orientação a objetos, é mais bem enquadrada como uma técnica de reengenharia de software, isto é, que altera um software para reconstituí-lo em uma nova forma, função e implementação, que como uma técnica de engenharia reversa de software, isto é, uma técnica que analisa um software e cria novas representações abstratas do mesmo.

**Comentários:**

Trata-se do exato oposto: lembrem-se que a refatoração muda dentro sem mudar fora, logo não se encaixa como uma técnica de reengenharia, porque essa altera o software para reconstruí-lo de uma nova forma.

**Gabarito:** Errado

---

**25. (CESPE / BASA – 2012)** Denomina-se refatoração a atividade de reestruturação de programas, classes e métodos existentes para adaptá-los a alterações de funcionalidades e requisitos.

**Comentários:**

A questão está errada (mas a banca não entendeu dessa forma). Se houve alterações de funcionalidade, não é uma refatoração! Isso talvez seria uma reengenharia de software. Enfim, discordo...

**Gabarito:** Correto

---

**26. (CESPE / BASA – 2012)** A refatoração objetiva tornar o código mais claro e limpo.

**Comentários:**

Perfeito! Ela ajuda a compreensão do código-fonte e encoraja o desenvolvedor a pensar sobre suas decisões de projeto.

**Gabarito:** Correto

---

**27. (CESPE / BASA – 2012)** Ao refatorar um código, altera-se a funcionalidade do sistema.

**Comentários:**

Na verdade, a funcionalidade permanece a mesma.

**Gabarito:** Errado

---

**28. (CESPE / UNIPAMPA – 2013)** No que concerne às mudanças futuras, a refatoração de um programa orientado a objetos e a manutenção preventiva têm propostas opostas.

**Comentários:**

Não, têm propostas similares – ambas buscam melhorar a estrutura do código sem alterar seu comportamento externo.

**Gabarito:** Errado

---

**29. (CESPE / ANAC – 2009)** A técnica conhecida como refactoring é constantemente aplicada no desenvolvimento baseado no método ágil extreme programming.

**Comentários:**

Perfeito! Ela é realmente preconizada pelo XP para reorganizar e simplificar o projeto (ou código) de um componente de software sem modificar sua função ou seu comportamento.

**Gabarito:** Correto

---

**30. (CESPE / INMETRO – 2009)** A refabricação (ou refactoring) significa que primeiro deve ser desenvolvido um conjunto mínimo de funcionalidades que agreguem valor ao negócio e, depois, novas funcionalidades devem ser incrementadas ao produto já entregue.

**Comentários:**

Não faz o menor sentido. Nada de novas funcionalidades, é apenas otimização do comportamento interno.

**Gabarito:** Errado

---

31. **(CESPE / SAD-PE – 2010)** Em ferramentas CASE, como refactoring, é melhor adotar-se uma abordagem formal que uma abordagem heurística.

**Comentários:**

Bem... a despeito de a questão tratar *refactoring* como uma ferramenta e, não, como uma técnica, é melhor adotar uma abordagem heurística, devido a imensa variedade de algoritmos, entradas, saídas, etc. Além disso, é uma abordagem puramente empírica, baseada em fatos reais, na procura de possíveis melhorias.

**Gabarito:** Errado

---

32. **(CESPE / STF – 2013)** O refactoring aprimora o design de um software, reduz a complexidade da aplicação, remove redundâncias desnecessárias, reutiliza código, otimiza o desempenho e evita a deterioração durante o ciclo de vida de um código.

**Comentários:**

Perfeito! Ele melhora o design, reduz a complexidade, remove redundâncias, aumento reuso e otimiza o desempenho do software – isso auxilia a evitar a deterioração durante o ciclo de vida de um código.

**Gabarito:** Correto

---

33. **(CESPE / TCU – 2015)** A cada nova funcionalidade de software adicionada na prática de refactoring (refatoração) em XP, a chance, o desafio e a coragem de alterar o código-fonte de um software são aproveitados como oportunidade para que o design do software adote uma forma mais simples ou em harmonia com o ciclo de vida desse software, ainda que isso implique a alteração de um código com funcionamento correto.

**Comentários:**

A questão está completamente errada! Vimos insistentemente que a refatoração não adiciona novas funcionalidades. Quando vi que o gabarito preliminar veio como correto, achei um absurdo. Eu tinha certeza de que a banca mudaria o gabarito, mas ela não deu o braço a torcer e apenas anulou a questão. Menos mal...

**Gabarito:** Anulada

---

34. **(CESPE / TRE-PE – 2017)** Refactoring é o processo que:

a) implementa todas as funcionalidades da camada de model para depois implementar as camadas de controller e de viewer, nos casos em que a arquitetura MVC é utilizada.

b) efetua mudanças em um código existente e funcional sem alterar seu comportamento externo, com o objetivo de aprimorar a estrutura interna do código.

c) inclui funcionalidades extras no código, com o intuito de aprimorá-lo (rich source-code).

d) aprimora a extração e o refinamento iterativo dos requisitos do produto ainda na fase de planejamento do software, sendo considerado um valor na XP (extreme programming).

e) estabelece os métodos, um após o outro, para depois definir as classes e suas abstrações e implementar as interfaces.

**Comentários:**

Refatoração é o processo de mudar um sistema de software de tal forma que não altere o comportamento externo do código, embora melhore sua estrutura interna – nenhum dos outros itens faz qualquer sentido.

**Gabarito:** Letra B

---

**35. (FCC / MPE-SE – 2009)** Quanto à caracterização, a reengenharia de software é classificada como manutenção:

a) preventiva.
b) criptográfica.
c) de melhoria.
d) adaptativa.
e) corretiva.

**Comentários:**

A reengenharia de software é também conhecida como manutenção preventiva.

**Gabarito:** Letra A

---

**36. (CESPE / PEFOCE – 2012)** Como regra geral, não se deve tentar reestruturar um sistema com o uso da reengenharia se a abordagem inicial do sistema legado for funcional e a versão melhorada desejada for orientada a objetos.

**Comentários:**

O problema com a reengenharia de software é que existem limites práticos para o quanto você pode melhorar um sistema por meio da reengenharia. Não é possível, por exemplo, converter um sistema escrito por meio de uma abordagem funcional para um sistema orientado a objetos. As principais mudanças de arquitetura ou a reorganiza-ção radical do sistema de gerenciamento de dados não podem ser feitas automaticamente, pois são muito caras. Embora a reengenharia possa melhorara manutenibilidade, o sistema reconstruído provavelmente não será tão manutenível como um novo sistema, desenvolvido por meio de métodos modernos de engenharia de software.

**Gabarito:** Correto

---

**37. (CESPE / PEFOCE – 2012)** Reestruturação de software é uma atividade do processo de reengenharia de software voltada para a modificação da arquitetura global do programa, cujo objetivo consiste em tornar mais fácil o entendimento, os testes e a manutenção dos softwares.

**Comentários:**

Na verdade, não se modifica a arquitetura global. A reestruturação de software modifica o código-fonte e/ou dados para torná-lo mais amigável para futuras alterações. Lembrando que a reestruturação de software é composta pela reestruturação de código e a reestruturação de dados.

**Gabarito:** Errado

---

**38. (CESPE / PC-ES – 2011)** A reengenharia procura introduzir melhorias em processos já existentes, reformulando o que já existe ou fazendo pequenas mudanças que deixem as estruturas básicas intactas.

**Comentários:**

A reengenharia reconstitui o sistema em uma nova forma, logo não se trata de pequenas mudanças que deixem as estruturas básicas intactas.

**Gabarito:** Errado

---

**39. (CESPE / TRE-BA – 2010)** Das várias estratégias de mudança de software, realizar alterações significativas na arquitetura do sistema de software diz respeito a reengenharia de software.

**Comentários:**

Discordo do gabarito dessa questão! Para mim, ela está correta. Não vislumbro qualquer erro – caso alguém saiba, favor informar :)

**Gabarito:** Errado

---

# LISTA DE QUESTÕES – DIVERSAS BANCAS

1. **(FUNDATEC / BRDE - 2023)** Avalie as assertivas abaixo:

   I. Melhoria do design interno (arquitetura) do software.
   II. Código mais legível.
   III. Localização de bugs.
   IV. Mudança do comportamento externo do software.

   Quantas podem vir a ser benefícios do processo de refatoração de código?

   a) 0
   b) 1
   c) 2
   d) 3
   e) 4

2. **(QUADRIX / PRODAM-AM - 2022)** Assinale a alternativa que apresenta a prática de XP (Extreme Programming) que é definida como uma técnica disciplinada para reestruturar um corpo de código existente, alterando a sua estrutura interna sem alterar seu comportamento externo. Essa prática mantém a semântica do código, ou seja, após as mudanças, o código ainda funciona da mesma forma.

   a) refatoração
   b) metáfora
   c) *test-driven development*
   d) *coding standards*
   e) *on-site customer*.

3. **(FUNDEP / UFJF - 2022)** Considere o trecho de código a seguir, que acabou de ser refatorado.

```
delta = b*b-4*a*c; // nova variável
x1 = (-b + sqrt(delta)) / (2*a);
x2 = (b + sqrt(delta)) / (2*a);
```

   Assinale a refatoração aplicada para essa situação.

   a) Inline de método.
   b) Extração de método.
   c) Extração de variável.
   d) Renomeação.
   e) Extração de classe.

4. **(CESPE / BANRISUL - 2022)** A filosofia da modelagem ágil não admite decisões que levem um projeto a sua rejeição e a sua refatoração, pois os profissionais de tecnologia não possuem todas as respostas e outros stakeholders envolvidos no negócio devem ser respeitados e integrados ao processo.

5. **(CESPE / MC - 2022)** Refatorar um *software* consiste em modificar o seu comportamento interno e externo, mantendo-se inalterada sua estrutura interna.

6. **(CONSULPAM / Prefeitura de Irauçuba - CE - 2022)** Dentro das metodologias ágeis, o processo de desenvolvimento de software especificado pela Programação Extrema (eXtreme Programming, XP) possui algumas características específicas. Uma das características do XP versa sobre as necessidades de melhoria no projeto, que devem ser realizadas através de um tipo de processo específico para este fim. Assinale a alternativa com o nome deste tipo de processo.

   a) Testes.
   b) Refatoração.
   c) Histórias do Usuário.
   d) Programação em Pares.

7. **(IDECAN / IF-CE - 2021)** No que diz respeito à manutenção e reengenharia de software, um termo define o processo de alterar o código-fonte, de modo que não altere o comportamento externo e ainda melhore a sua estrutura interna. É uma técnica disciplinada de limpar e organizar o código, e por consequência, minimizar a chance de introduzir novos bugs. Esse termo é conhecido como

   a) elicitação.
   b) refatoração.
   c) recodificação.
   d) replicação.

8. **(IBADE / Prefeitura de Vilhena - 2019)** Em Orientação a Objetos define-se o processo de Refatoração como:

   a) processo de criação de entidades com herança e polimorfismo.
   b) classe abstrata que permite a criação de métodos fatorados.
   c) classe que permite herdar da classe pai todos os seus objetos.
   d) processo para melhorar a legibilidade ou a eficiência do código, preservando seu comportamento.
   e) objeto que permite acessar classes fatoradas.

9. **(CESPE / MPC-PA - 2019)** No contexto da manutenção preventiva de sistemas no paradigma orientado a objetos, a refatoração é uma técnica empregada com o objetivo de:

a) adicionar uma nova funcionalidade no sistema.
b) incluir uma generalidade que possa ser necessária no futuro.
c) realizar a reengenharia de um sistema.
d) reduzir o risco da introdução de novos erros no programa.
e) substituir métodos similares em subclasses por um único método em uma superclasse.

10. **(IF-PA / IF-PA - 2019)** Ao analisarmos uma classe Java, nos deparamos com um método que implementa diversas funcionalidades, tornando-se um método com muitas linhas de código, de difícil compreensão e manutenção. Para melhorar essa situação, decidimos dividi-lo em métodos menores, mais fáceis de entender e de efetuar manutenções. A esse processo de organizar e melhorar a estrutura interna de uma aplicação, denominamos de:

a) indentação.
b) depuração.
c) inspeção.
d) integração.
e) refatoração.

11. **(CESPE / SLU – 2019)** Refactoring (refatoração) é o processo utilizado para reescrever aplicações desatualizadas, com a finalidade de incrementar e melhorar suas funcionalidades; o uso dessa técnica normalmente aprimora aplicações para disponibilizá-las na Internet.

12. **(AOCP / PRODEB – 2018)** "Processo de alteração de um sistema de software de tal forma que não se altere o comportamento externo do código, mas se aprimore a estrutura interna". O enunciado se refere a:

a) Reúso.
b) Teste Unitário.
c) Teste de Integração.
d) Refatoração.
e) Prototipação.

13. **(CS-UFG / Câmara de Goiânia – 2018)** Sejam as classes A e B tais que o relacionamento entre elas é dado pelo fato de A usar (referenciar) a classe B. Dessa forma, qual das refatorações a seguir implementa o princípio da inversão de dependência?

a) Cria interface para serviços oferecidos por B; a classe A passa a usar a interface criada; a classe B passa a implementar a interface criada; a classe A não usa mais a classe B.

b) Cria interface para serviços oferecidos por A; a classe A passa a implementar a interface criada; a classe B passa a usar a interface criada; a classe A não usa mais a classe B.

c) Cria um relacionamento de herança entre as classes A e B (A torna-se uma especialização de B); métodos da classe B empregados pela classe A são migrados para a classe A; a classe A não usa mais a classe B.

d) Cria uma referência para a classe B na classe A; cria um método para receber uma instância de B (injeção de dependência) e guarda-a na referência criada; a classe A não usa mais a classe B.

14. **(FAURGS / TJ-RS – 2018)** Em relação à refatoração, assinale com V (verdadeiro) ou F (falso) as afirmações abaixo.

() O melhor momento para se refatorar um código é durante os testes de aceitação, pois o cliente tem interesse em um código de qualidade.

() Um dos passos da refatoração é a aplicação dos testes que verificarão sua implementação.

() Rotinas muito longas e código duplicado são exemplos de bad smells.

() Refatorações são modificações no código que são simples a ponto de não gerarem nenhum efeito prático.

() Um código que já foi refatorado uma vez não precisará ser refatorado no futuro, pois já atende aos critérios de qualidade exigidos.

() A refatoração de um código implica apenas a melhoria de sua qualidade interna e não deve afetar sua funcionalidade original.

A sequência correta de preenchimento dos parênteses, de cima para baixo, é

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

15. **(CEPS-UFPA / UFPA – 2018)** Acerca do tema refatoração de software, considere as afirmativas.

I A refatoração busca evoluir o projeto e código-fonte de um sistema de software para se alcançar alta coesão, isto é, suas classes devem possuir conjuntos extensos de responsabilidades.

II A refatoração busca evoluir o projeto e código-fonte de um sistema de software para alcançar baixo acoplamento, isto é, a colaboração entre as classes deve ser mantida em um nível mínimo aceitável.

III A refatoração é o processo de mudar um sistema de software de tal forma que não altere o comportamento externo do código-fonte, embora melhore sua estrutura interna.

Está(ão) correta(s)

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

16. **(CESPE / STJ – 2018)** A refatoração de um código escrito em Delphi pode levar um método a ser separado e transformado em alguns outros métodos.

17. **(COPERVE-UFSC / UFSC – 2018)** Considere os seguintes exemplos de procedimentos de manutenção, no contexto da necessidade de alteração de um programa hipotético de controle acadêmico de cursos de graduação da UFSC:

I. fazer com que o resultado da matrícula passe a ter a opção de gerar o resultado em formato PDF, além da atual possibilidade de informar na tela;

II. incluir funcionalidade para permitir que o trancamento de matrícula possa ser feito on-line;

III. reorganização da hierarquia de herança das classes do programa;

IV. criar classes no programa;

V. remover classes do programa;

Assinale a alternativa que relaciona apenas procedimentos de manutenção que podem ser classificados como ações de refatoração (refactoring).

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

18. **(CESPE / CGM-JP – 2018)** A refatoração recomendada pela metodologia XP consiste na reorganização interna do código-fonte sem alteração no seu comportamento, o que permite melhorias no projeto, mesmo após o início da implementação.

19. **(CESPE / DPE-RO – 2021)** No contexto das metodologias ágeis, o conceito de refatoração compreende:

a) o desenvolvimento de testes incrementais a partir de novos cenários.
b) a renomeação de atributos e métodos para implementar melhorias no software.
c) a decomposição de histórias de usuário em uma série de tarefas de desenvolvimento.
d) a substituição do resultado de uma sprint inteira para atender a requisitos diferentes dos originais.
e) a junção de alterações de código às funcionalidades do software já entregue.

20. **(FCC / DPE-SP – 2010)** A refatoração é o processo de modificar um sistema de software para melhorar a estrutura interna do código sem alterar seu comportamento externo.

21. **(CESPE / CENSIPAM – 2006)** A refatoração modifica a estrutura interna de um software visando facilitar o entendimento e as futuras modificações sem alterar o comportamento apresentado pelo software. Não é uma prática que possa ser aplicada em processos de desenvolvimento ágeis, pois requer a construção de modelos tanto para o projeto de alto nível quanto para o projeto detalhado.

22. **(CESPE / CENSIPAM – 2006)** A refatoração é aplicável quando são identificados fragmentos de código que podem ser agrupados, expressões complicadas, atributos acessados mais por outras classes que pelas classes das quais são membros, enunciados condicionais complexos, códigos duplicados, longos métodos, longas classes, muitos parâmetros, métodos ou classes pouco usadas.

23. **(CESPE / INMETRO – 2009)** As técnicas de refatoração de código compreendem, entre outras, a remoção de números mágicos e a introdução de padrões de desenho.

24. **(CESPE / INMETRO – 2010)** A técnica de refatoração, utilizada no paradigma de orientação a objetos, é mais bem enquadrada como uma técnica de reengenharia de software, isto é, que altera um software para reconstituí-lo em uma nova forma, função e implementação, que como uma técnica de engenharia reversa de software, isto é, uma técnica que analisa um software e cria novas representações abstratas do mesmo.

25. **(CESPE / BASA – 2012)** Denomina-se refatoração a atividade de reestruturação de programas, classes e métodos existentes para adaptá-los a alterações de funcionalidades e requisitos.

26. **(CESPE / BASA – 2012)** A refatoração objetiva tornar o código mais claro e limpo.

27. **(CESPE / BASA – 2012)** Ao refatorar um código, altera-se a funcionalidade do sistema.

28. **(CESPE / UNIPAMPA – 2013)** No que concerne às mudanças futuras, a refatoração de um programa orientado a objetos e a manutenção preventiva têm propostas opostas.

29. **(CESPE / ANAC – 2009)** A técnica conhecida como refactoring é constantemente aplicada no desenvolvimento baseado no método ágil extreme programming.

30. **(CESPE / INMETRO – 2009)** A refabricação (ou refactoring) significa que primeiro deve ser desenvolvido um conjunto mínimo de funcionalidades que agreguem valor ao negócio e, depois, novas funcionalidades devem ser incrementadas ao produto já entregue.

31. **(CESPE / SAD-PE – 2010)** Em ferramentas CASE, como refactoring, é melhor adotar-se uma abordagem formal que uma abordagem heurística.

32. **(CESPE / STF – 2013)** O refactoring aprimora o design de um software, reduz a complexidade da aplicação, remove redundâncias desnecessárias, reutiliza código, otimiza o desempenho e evita a deterioração durante o ciclo de vida de um código.

33. **(CESPE / TCU – 2015)** A cada nova funcionalidade de software adicionada na prática de refactoring (refatoração) em XP, a chance, o desafio e a coragem de alterar o código-fonte de um software são aproveitados como oportunidade para que o design do software adote uma forma mais simples ou em harmonia com o ciclo de vida desse software, ainda que isso implique a alteração de um código com funcionamento correto.

34. **(CESPE / TRE-PE – 2017)** Refactoring é o processo que:

a) implementa todas as funcionalidades da camada de model para depois implementar as camadas de controller e de viewer, nos casos em que a arquitetura MVC é utilizada.

b) efetua mudanças em um código existente e funcional sem alterar seu comportamento externo, com o objetivo de aprimorar a estrutura interna do código.

c) inclui funcionalidades extras no código, com o intuito de aprimorá-lo (rich source-code).

d) aprimora a extração e o refinamento iterativo dos requisitos do produto ainda na fase de planejamento do software, sendo considerado um valor na XP (extreme programming).

e) estabelece os métodos, um após o outro, para depois definir as classes e suas abstrações e implementar as interfaces.

35. **(FCC / MPE-SE – 2009)** Quanto à caracterização, a reengenharia de software é classificada como manutenção:

a) preventiva.
b) criptográfica.
c) de melhoria.
d) adaptativa.
e) corretiva.

36. **(CESPE / PEFOCE – 2012)** Como regra geral, não se deve tentar reestruturar um sistema com o uso da reengenharia se a abordagem inicial do sistema legado for funcional e a versão melhorada desejada for orientada a objetos.

37. **(CESPE / PEFOCE – 2012)** Reestruturação de software é uma atividade do processo de reengenharia de software voltada para a modificação da arquitetura global do programa, cujo objetivo consiste em tornar mais fácil o entendimento, os testes e a manutenção dos softwares.

38. **(CESPE / PC-ES – 2011)** A reengenharia procura introduzir melhorias em processos já existentes, reformulando o que já existe ou fazendo pequenas mudanças que deixem as estruturas básicas intactas.

39. **(CESPE / TRE-BA – 2010)** Das várias estratégias de mudança de software, realizar alterações significativas na arquitetura do sistema de software diz respeito a reengenharia de software.

# GABARITO – DIVERSAS BANCAS

1. LETRA D
2. LETRA A
3. LETRA C
4. ERRADO
5. ERRADO
6. LETRA B
7. LETRA B
8. LETRA D
9. LETRA E
10. LETRA E
11. ERRADO
12. LETRA D
13. LETRA A
14. LETRA C
15. LETRA E
16. CORRETO
17. LETRA A
18. CORRETO
19. LETRA B
20. CORRETO
21. ERRADO
22. CORRETO
23. CORRETO
24. ERRADO
25. CORRETO
26. CORRETO
27. ERRADO
28. ERRAD
29. CORRETO
30. ERRADO
31. ERRADO
32. CORRETO
33. ANULADA
34. LETRA B
35. LETRA A
36. CORRETO
37. ERRADO
38. ERRADO
39. ERRADO

# INTEGRAÇÃO, ENTREGA E IMPLANTAÇÃO CONTÍNUA

## Conceitos Básicos

INCIDÊNCIA EM PROVA: BAIXA

**A integração contínua é um termo originado na metodologia ágil XP (Extreme Programming), apesar de ser utilizada em diversos outros contextos. Ela consiste em um processo bastante simples:** o desenvolvedor integra o código alterado e/ou desenvolvido ao projeto principal na mesma frequência com que as funcionalidades são desenvolvidas, sendo feito idealmente muitas vezes ao dia ao invés de apenas uma vez.

**O objetivo principal de utilizar a integração contínua é verificar se as alterações ou novas funcionalidades implementadas não criaram novos defeitos no projeto já existente.** A prática da integração contínua pode ser feita através de processos manuais ou automatizados, utilizando ferramentas como o Jenkins, Hudson, entre outros. Um dos nossos maiores guias, Martin Fowler, já dizia sobre esse tema:

> *"A Integração Contínua se trata de uma prática de desenvolvimento de software em que os membros de um time integram seu trabalho frequentemente, geralmente cada pessoa integra pelo menos diariamente – podendo haver múltiplas integrações por dia. Cada integração é verificada por um build automatizado (incluindo testes) para detectar erros de integração o mais rápido possível. Muitos times acham que essa abordagem leva a uma significante redução nos problemas de integração e permite que um time desenvolva software coeso mais rapidamente".*

**Basicamente, a grande vantagem da integração contínua está no feedback instantâneo.** A prática da integração contínua se mostra muito eficaz nas equipes de desenvolvimento e está sendo amplamente utilizada. Inúmeros projetos *open-source* utilizam várias máquinas dedicadas a serem servidores de integração, rodando muitas vezes em softwares/plataformas diferentes. Ela é recomendada tanto com equipes distantes geograficamente quanto com equipes no mesmo local.

No primeiro caso, recomenda-se o uso de integração contínua assíncrona; e no segundo caso, recomenda-se o uso de integração contínua síncrona. **Em ambos os casos, é importante ter pelo menos uma máquina dedicada à integração que seja uma cópia idêntica do ambiente de produção, pois quanto mais rápido for o feedback melhor**. Dessa forma, a integração contínua busca alcançar alguns objetivos...

Primeiro, minimizar a duração e esforço requeridos por cada episódio de integração; e segundo, ser capaz de entregar uma versão do produto que possa ser lançada a qualquer momento. **É requerido um procedimento de integração que seja reproduzível no mínimo e, em grande parte, automatizável**. Essa automatização não visa apenas evitar trabalho repetitivo. Ela na verdade permite aos desenvolvedores alcançarem um nível muito maior de produtividade e qualidade.

Um problema recorrente no desenvolvimento de software é que os erros ocultos nos programas que desenvolvemos são cada vez mais caros e difíceis de corrigir à medida que o tempo avança. Qualquer pessoa que trabalhe com programação já deve ter passado pela situação de achar que um programa estava quase pronto e "só faltava testar". **Depois descobre-se que, na verdade, muita coisa ainda estava errada ou até mesmo faltando no código**.

Outra situação recorrente é a "preguiça" de testar o software, ou pelo menos a prática de postergar os testes. Sem testes e *deploys* automatizados, o desenvolvedor tende a escrever uma grande quantidade de código antes de realmente colocar o programa para executar, afinal se o processo é trabalhoso e toma muito tempo, ele vai evitar de fazer isso ao máximo. **Só que no final, gasta-se muito tempo para corrigir todos os "detalhes" que não estavam corretos**.


Para mitigar todos esses problemas, surgiu a Integração Contínua – justamente para automatizar o processo para que, com bastante frequência, uma ou mais vezes ao dia, seja possível integrar todas as alterações de todos os desenvolvedores envolvidos no projeto e realizar um teste geral. **Tem uma metáfora que eu gosto de usar bastante para explicar esse tema: um filme de ação**. Ora, todo filme é composto de várias cenas. Em nosso caso, uma aplicação é composta de vários builds.

Toda vez que eu adiciono uma nova cena em um filme, o editor faz um teste rodando todo o filme novamente para ver se o filme faz sentido com a nova cena. Caso o teste falhe, i.e., a nova cena não se encaixe na história, ela deve ser refeita! **Em nosso caso, dizemos que ele a build foi quebrada**. **Dessa maneira, em vez de construir vários componentes separadamente e depois juntá-los em um software final, nós vamos integrar continuamente**.

Novos builds só são integrados quando a build anterior for corrigida. Esses procedimentos auxiliam a reduzir riscos e a entregar soluções livres de defeitos.

| AS PRÁTICAS... |
|---|
| - Mantenha um único repositório de código-fonte. |
| - Automatize um build. |
| - Faça um build auto-testável. |
| - Todo commit devem ser um build na máquina de integração. |
| - Mantenha os builds rápidos. |
| - Teste em uma cópia do ambiente de produção. |
| - Mantenha fácil que todos consigam o último executável. |
| - Todos consegue visualizar o processo. |
| - Automatização do deployment. |

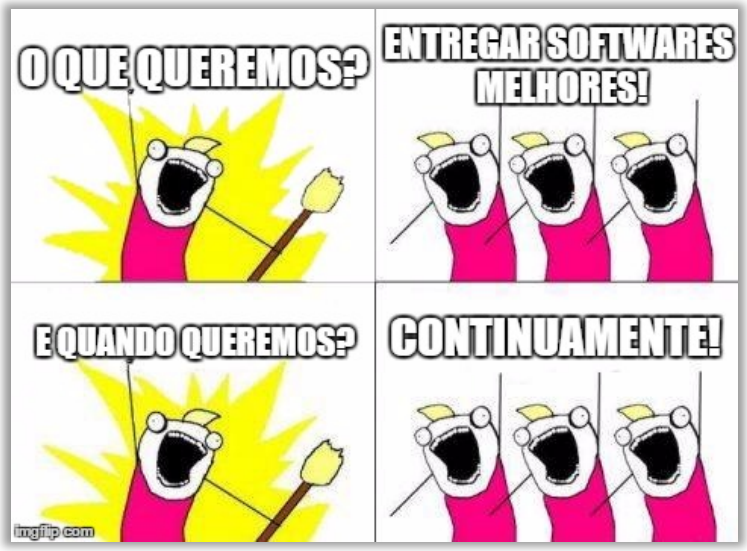
| COMO FAZER... |
|---|
| - Desenvolvedores devem fazer checkout do código em seus workspaces privados. |
| - Quando finalizado, o commit modifica o repositório. |
| - O Servidor de Integração Contínua monitora o repositório e faz checkout das mudanças quando elas ocorrem. |
| - O Servidor de Integração Contínua constrói o sistema e roda testes de integração e testes de unidade. |
| - O Servidor de Integração Contínua lança artefatos implantáveis para testes. |
| - O Servidor de Integração Contínua atribui um rótulo de build para a versão do código que ele construiu. |
| - O Servidor de Integração Contínua informa ao time sobre o sucesso do build. |
| - Se a build ou teste falhar, o Servidor de Integração Contínua alerta a equipe. |
| - A equipe corrige o problema na melhor oportunidade. |
| - Continue a integrar continuamente e a testar durante o projeto. |

**Vamos falar um pouco sobre a entrega contínua. Uma equipe ágil geralmente entrega seu produto nas mãos dos usuários finais, ouvindo seus feedbacks (críticas e elogios).** A frequência de entregas varia de acordo com aspectos técnicos e negociais do contexto que se encontra, mas se nós tivéssemos que chutar um número, diríamos que há uma entrega a cada quatro a seis iterações, no máximo.

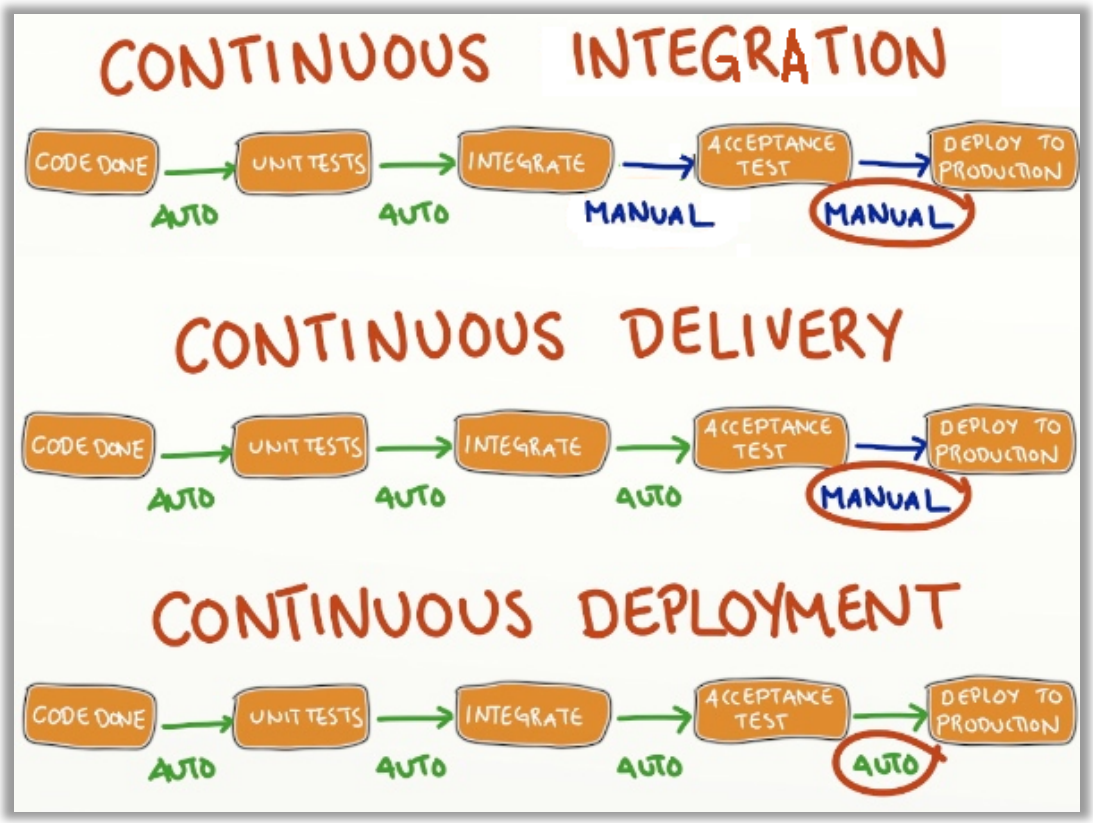
Em contextos técnicos favoráveis, como desenvolvimento web, um ritmo mais frequente de entregas pode ser alcançado (Ex: uma entrega a cada iteração). Já algumas equipes vão ao limite dessa prática por meio de *continous deployments*. **Apresentar a última versão do produto para o gerente de projetos/produtos para teste não é suficiente, nem entregar a uma equipe de garantia de qualidade**.

Uma entrega, em seu sentido primário, deve ser pelo menos uma versão beta avaliada por usuários representativos. Em alguns casos (como em softwares embarcados), não é possível organizar entregas contínuas para todos os usuários, mas isso não deve se tornar um pretexto para desistir das entregas contínuas mesmo que somente para alguns usuários. Entre os benefícios esperados, busca-se mitigar a falha de planejamento bastante conhecida de descobrir atrasos muito tarde.

Ela ajuda a validar o ajuste do produto ao mercado mais cedo, fornece feedbacks adiantados sobre a qualidade e estabilidade, e permite um retorno mais rápido do investimento sobre o produto.




*Professor, qual a diferença entre Integração, Entrega e Implantação Contínua?* **A primeira é o processo de testes e integração de um novo código com a base de código já existente – geralmente são feitos testes automatizados**. A segunda é o processo de garantir que o novo código pode ser implantado no ambiente de produção a qualquer momento – geralmente são feitos testes de comportamento.



Por fim, o terceiro trata do processo de publicação (*deploy*) no ambiente de produção, tão logo você esteja certo de que o código passou por todos os testes e está pronto para ser publicado. **Se o código já passou pelos testes automatizados e pelos testes manuais, visuais e de comportamento, então ele pode ser publicado de forma automatizada**. Observem a imagem anterior...

Quem ainda não entendeu, vai entender agora! **Imaginem que temos uma equipe de desenvolvimento formada por quatro pessoas trabalhando em um sistema**. Cada desenvolvedor está focado em implementar uma funcionalidade diferente. Caso o primeiro desenvolvedor termine sua parte na codificação, ele pode fazer um *commit* do código em uma ferramenta de controle de versão (Ex: Git).

A Integração Contínua é uma prática que pode ser aplicada por meio de uma ferramenta (Ex: Jenkins). Essa ferramenta pode ser configurada para, de tempos em tempos, buscar tudo que foi *comitado* pelos desenvolvedores e realizar testes de unidade e de integração automaticamente. **Caso algum problema seja encontrado, temos um feedback imediato**. Os desenvolvedores são notificados e ninguém faz check-in do repositório até que o problema seja resolvido.

**Se os testes passarem, o primeiro desenvolvedor saberá que o código que ele implementou não quebrou o sistema**. Então, percebam que a integração contínua consiste em, periodicamente, realizar testes automatizados e integrar novos códigos ao sistema (via de regra, no ambiente de desenvolvimento e, não, de produção). Já a diferença entre entrega contínua e implantação contínua é um pouco mais sutil...

A primeira é uma prática ágil na qual o software é construído de tal forma que ele pode ser colocado em produção a qualquer momento. A segunda é uma prática ágil na qual o software é construído de tal forma que ele é colocado em produção em determinado momento (isso é configurável). A grande diferença é por conta do último ponto no nosso desenho. A entrega contínua apresenta uma versão candidata para publicação após cada funcionalidade passar por todos os estágios.

Após isso, o negócio poderá decidir quando colocá-la efetivamente em produção. Já na implantação contínua, a entrada em produção ocorre automaticamente. *Do início, vamos andar pelo desenho?* Idealmente, o terceiro desenvolvedor pode terminar uma nova funcionalidade e realizar o *commit* do seu código para sua ferramenta de controle de versão. **Serão realizados testes de unidade e testes de integração automatizados**.

Passando por esses estágios, o código é integrado ao ambiente de desenvolvimento. Agora, todos os desenvolvedores poderão usufruir dessa nova funcionalidade implementada pelo terceiro desenvolvedor. Caso tenhamos também Entrega Contínua, o fluxo não para por aí! **De tempos em tempos, o sistema integrado passará por testes de aceitação automatizados**. Caso não haja erros, ele estará pronto para ser implantado em um ambiente de produção.

**É comum que esse código fique em um ambiente de homologação, aguardando o negócio decidir quando ele deverá ser implantado, finalmente, no ambiente de produção**. Caso tenhamos também a implantação contínua, o fluxo continua e, não mais será uma decisão do negócio a implantação ou não do software em produção – a implantação ocorrerá automaticamente. Fim ;)

# QUESTÕES COMENTADAS – DIVERSAS BANCAS

1. **(CESGRANRIO / Petrobrás– 2010)** É comum, na Engenharia de Software, o uso de ferramentas de software que auxiliam na realização de diversas atividades do desenvolvimento. Nesse contexto, ferramentas de integração contínua são destinadas a automatizar a implantação do produto de software no ambiente de produção.

**Comentários:**

Não se trata de automatizar a implantação, mas de oferecer feedbacks tempestivos por meio da integração a todo momento. Automatizar a implantação é apenas um detalhe!

**Gabarito:** Errado

---

2. **(CESPE / TCE-PE – 2004)** A prática de integração contínua depende fortemente do uso de ferramentas de build e controle de versão.

**Comentários:**

Perfeito! Necessita de ferramentas de build e controle de versão. Cada integração é verificada por um build automatizado (incluindo testes) para detectar erros de integração o mais rápido possível.

**Gabarito:** Correto

---

3. **(FCC / BACEN – 2006)** A técnica de *Continuous Integration* diz que o código desenvolvido por cada par de desenvolvedores deve ser integrado ao código base constantemente. Quanto menor o intervalo entre cada integração, menor a diferença entre os códigos desenvolvidos e maior a probabilidade de identificação de erros, pois cada vez que o código é integrado, todos os *unit tests* devem ser executados, e, se algum deles falhar, é porque o código recém integrado foi o responsável por inserir erro no sistema.

**Comentários:**

Perfeito! O desenvolvedor integra o código alterado e/ou desenvolvido ao projeto principal na mesma frequência com que as funcionalidades são desenvolvidas, sendo feito idealmente muitas vezes ao dia ao invés de apenas uma vez.

**Gabarito:** Correto

---

4. **(CESPE / TCU – 2015)** Para que a prática de integração contínua seja eficiente, é necessário parametrizar e automatizar várias atividades relativas à gerência da configuração, não somente do código-fonte produzido, mas também de bibliotecas e componentes externos.

**Comentários:**

Perfeito! Parametrizam-se e automatizam-se componentes como código-fonte, bibliotecas, scripts de build, etc – ajustando a dependência entre eles.

**Gabarito:** Correto

---

# LISTA DE QUESTÕES – DIVERSAS BANCAS

1. **(CESGRANRIO / Petrobrás– 2010)** É comum, na Engenharia de Software, o uso de ferramentas de software que auxiliam na realização de diversas atividades do desenvolvimento. Nesse contexto, ferramentas de integração contínua são destinadas a automatizar a implantação do produto de software no ambiente de produção.

2. **(CESPE / TCE-PE – 2004)** A prática de integração contínua depende fortemente do uso de ferramentas de build e controle de versão.

3. **(FCC / BACEN – 2006)** A técnica de *Continuous Integration* diz que o código desenvolvido por cada par de desenvolvedores deve ser integrado ao código base constantemente. Quanto menor o intervalo entre cada integração, menor a diferença entre os códigos desenvolvidos e maior a probabilidade de identificação de erros, pois cada vez que o código é integrado, todos os *unit tests* devem ser executados, e, se algum deles falhar, é porque o código recém integrado foi o responsável por inserir erro no sistema.

4. **(CESPE / TCU – 2015)** Para que a prática de integração contínua seja eficiente, é necessário parametrizar e automatizar várias atividades relativas à gerência da configuração, não somente do código-fonte produzido, mas também de bibliotecas e componentes externos.

# GABARITO – DIVERSAS BANCAS

1. ERRADO
2. CORRETO
3. CORRETO
4. CORRETO

# 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.**