模擬試験

2022 年 12 月版

目次

はじめに

これは EXIN DevOps Foundation (DEVOPSF.JP)のサンプル試験です。この試験は EXIN 試験の規則および規定を適用します。

本試験は選択式の問題が 40 問で構成されます。特に明記しない限り、各問題には、選択肢が複数ありますが、そのうち正解は 1 つのみです。

この試験で取得できる最大点数は 40 点です。各正解には 1 点の価値があります。試験に合格するには 26 点以上が必要です。

本試験の制限時間は 60 分です。

ご健闘をお祈りいたします。

模擬試験

1 / 40
アジャイルの**誤った**考え方はどれでしょうか？

A) ビジネス側と開発側が、プロジェクト全体を通して一緒に作業を進める。
B) 変化に対応することは、計画に従うことより重要である。
C) 顧客の要件を確実に遂行することで顧客に満足してもらうことが優先事項である。
D) 動くソフトウェアは、進捗の第一の判断基準である。

2 / 40
米国立標準技術研究所（NIST）がクラウドコンピューティングに不可欠な特性として挙げて**いない**のはどれでしょうか？

A) Broad Network Access（ブロードバンドネットワークアクセス）
B) Pay-per-use System（従量制）
C) Rapid Elasticity（スピーディな拡張性）
D) Resource Pooling（リソースの共用）

3 / 40
DevOpsの登場を振り返ってみたとき、ソフトウェア開発の関係者間の新たな取り組み方や会話の仕方によって、新たなITマネジメント方法が必要になってきました。その結果としてDevOpsが誕生しました。

この新しいやり取りの方法を見つけた関係者はだれでしょうか？

A) ビジネスと顧客
B) IT部門と顧客
C) ITの開発と運用

4 / 40
DevOpsは、リーン生産方式の原則とプラクティスを多く取り入れています。ITの中でのムダの1つが、「タスク切り替え」です。

これはリーン生産方式のどのムダを、ITの世界に言い換えたのでしょうか？

A) 動作
B) 作り過ぎ
C) 運搬
D) 待ち

5 / 40
バリューストリーム・マップ（価値の流れ）の最も価値のある情報は、3つの重要な指標（メトリクス）から得られます。

これら3つの重要な指標の1つはどれでしょうか？

A) リードタイムとフローの組み合わせ
B) 完全正確率（percent complete and accurate; % C/A）
C) プロセスタイムをリードタイムで除算した結果
D) 達成価値からムダを減算した結果

6 / 40
DevOpsへ切り替えると、ＩＴに対するより大きな利益が期待されます。

DevOpsは、アジャイル、スクラム、リーンと何が違っているでしょうか？

A) DevOpsによって、DevとOpsとの間で新規および変更された製品のデリバリーが加速される。
B) DevOpsによって、新規および変更された製品を市場や顧客へデリバリーすることが加速される。
C) DevOpsによって、インフラストラクチャの変更に対して予算内で迅速な対応が保証される。
D) DevOpsによって、組織のバリューストリーム（価値の流れ）を妨げるインシデントへの迅速な対応が保証される。

7 / 40
ある会社では、製品を市場へ投入するまでの時間（time-to-market）を数年から数ヶ月へ短縮したいと考えています。ある従業員がDevOpsの手法を検討するべきであると提案しています。

DevOpsへ変化することで、この時間を短縮できる理由は何でしょうか？

A) DevOpsによって開発と運用が1つのチームに統合され、従業員の人数がより少なくて済むようになるから
B) DevOpsチームは、よりコストが高く、時間外を含めてより多くの時間で働き、プロダクトをより早く市場に投入するから
C) DevOpsでは自己完結型の専任チームが編成され、製品への変化する要求に対処できる俊敏性（アジリティ）をより高めることができるから

8 / 40
技術的負債を軽減する2つの一般的な方法とはどれでしょうか？

A) 正式な変更とリリース管理の手法
B) インシデント管理と要求実現の手法
C) 予算とリソースの増加
D) リファクタリングと、問題の直視

9 / 40
DevOpsが組織にもたらす大きな利点はどれでしょうか？

A) ビジネスシステムの脆弱性を排除する
B) 顧客のコストを削減する
C) 文化的な課題を軽減する

10 / 40
DevOpsはアジャイルの一部であるという考え方もあります。

DevOpsであれば答えが得られるものの、アジャイルでは答えが得られない質問はどれでしょうか？

A) 大きな市場シェアを獲得するのに十分な速さで、ソフトウェアを開発しリリースするには、どうするべきか？
B) 顧客の要求をより適切に理解するには、顧客との間でどのような関係が必要か？
C) リリースする製品を、便利で運用しやすいものにするには、どうするべきか？

11 / 40
DevOpsで「バリューストリーム（価値の流れ）」という概念がとても重要である理由は何でしょうか？

A) バリューストリームは、従業員が自分の毎日のタスクを確認し、理解するのに役立つ。
B) バリューストリームは、as-isマップの分析やメトリック（業績指標）を向上するための取り組みに役立つ。
C) バリューストリームは、担当者が自分の担当する仕事をいつ行ったのかを特定するのに役立つ。
D) バリューストリームは、すべてのプロセスステップ（工程）を通してスムーズで均一な流れを実現するのに役立つ。
E) バリューストリームは、現在の作業手順をローカルな範囲内で最適化するのに役立つ。

12 / 40
バリューストリーム・マップ（価値の流れ）の**最初の手順**は何でしょうか？

A) 要求仕様を作成する
B) 現行作業の文書化
C) プロセスの重要なステップを特定する

13 / 40
バリューストリーム・マップ（価値の流れ）は、バリューストリームにある効率の悪い所を明らかにします。

バリューストリーム・マップを作成するべき理由は何でしょうか？

A) ビジネスプロセスの最適化を支援するため
B) 十分に頑張って働いていない人を明らかにするため
C) 仕掛（WIP）中の作業をスピードアップさせるため
D) どのプロダクトラインを終了させるかを可視化するため

14 / 40
タスクには優先順位付けを行う必要があります。 この優先順位付けはバリューストリーム（価値の流れ）の入口にある待ち行列で行われます。

この優先順位付けでしばしば問題を引き起しますが、その理由はなぜですか？

A) ここで、タスクを自動化するにデプロイメント・パイプラインをどのように構築するかを決めるためです。そのための時間がかかり、遅延の原因になるため。
B) バリューストリームにおける重要な指標を計測できるようにすることを、間違った方法や効率の悪い方法で行うことで問題を引き起こすため。
C) タスクの流れのボトルネックを特定するために仕掛（WIP）の制限の状況を示すことができる可視化（見える化）ツールの導入が間違って行われるため。
D) バリューストリームのas-isバージョンとto-beバージョンの開発に加えて、必要とされる変更の一覧を作成する必要があり、かなりの時間がかかるため。
E) 従来型のアプローチでは、仕事を開始する前に多くの判断や決定が必要となり、大きな遅れを引き起こす原因となるため。

15 / 40
「デプロイメント・パイプライン」という概念の由来になっているアイデアはどれでしょうか？

A) 液体が流れるパイプライン
B) 自動車工場などの組み立てライン
C) 並列パイプラインを使用する最新型プロセッサー
D) 複数の組立ラインを使用するというアイデア
E) さまざまな仕事をする人たちを配置するプロセス

16 / 40
デプロイメント・パイプラインを導入する際は、いろいろな問題が発生します。まず最初に、本番稼働環境で安定した運用を保証するために、事前に準備開発された十分なテストを実施することができません。

どの解決策が**もっとも効果的に**この問題に対処できるでしょうか？

A) パイプラインを構築し、できるだけ多くの自動化を行うが、適切なテストがすべて揃うまで使用しない。
B) できるだけ早く解決しなければならない技術的負債として、コードのテストカバレッジを向上させる。
C) 開発したテストをパイプラインで実行し、潜在的な問題が本番環境で顕在化したときに対処する。
D) パイプラインの目的を、記述したコードをテストや品質保証（QA）のチームだけにデリバリーするためのインテグレーションシステムとする。

17 / 40
優れたバージョン管理システムは、DevOpsにおける高いパフォーマンスを予測する最大の要素の1つです。

バージョン管理を適切に適用するには、何が必要でしょうか？

A) 情報とその構成に取り組む作業に対する文化を変えること
B) 変更を導入する速さの大幅な向上
C) カオスや不安定性の本番環境への意図的な注入
D) 正式で自動化された変更管理プロセスの使用

18 / 40
DevOpsでは、迅速な移行とアプリケーションの信頼性の維持の適切なバランスを見つけることが重要です。

バージョン管理はこれをどのような方法でサポートしますか？

A) 不要なファイルや文書をチームのメンバー全員が自由に削除できるようにする
B) 少人数の独立した自己完結型の開発チームの編成を認める
C) ムダを排除または削減し、プロセスを最適化する専用ツールを適用する

19 / 40
構成管理にはどのようなメリットがあるでしょうか？

A) チームのすべてのメンバーがリスクなく不要なファイルを削除できる。
B) 主要なチームのメンバーが不在でも問題が発生するのを回避できる。
C) 変更されたコード、変更した人、変更された日付を、チームのメンバーが確認できる。

20 / 40
構成管理によって、作業員の人員を増やすことなく、ITインフラストラクチャやソフトウェアシステムをスケール（規模の拡大）させることが可能になります。

そのようにスケールされた（拡張された）システム環境では、理想的にはどのように変更が行われるべきでしょうか？

A) 継続的インテグレーションを通じた変更
B) 完全にコントロールされたスクリプトによる変更
C) 自動化テストを通じた変更
D) デプロイメント・パイプラインを通じた変更

21 / 40
明確な完了の定義（DoD）はDevOpsにおいて不可欠であり、顧客にとっての価値が考慮されます。

DevOpsにおける完了の正しい説明はどれでしょうか？

A) 要件が構築された段階で、その要件は完了となる。
B) そのコードがテストを終えた時、その要件は完了となる。
C) 製品が受け入れられた段階で、要件は完了となる。
D) 製品が本番環境にデプロイされた時、その要件は完了となる。

22 / 40
従来のプラクティスでは、変更点が文書化されていない、システムが完全にバックアップされていない、またはシステムの以前の状態が保存されていないなど、リリースの段階で多くの問題が発生する可能性があります。

DevOpsは、 これらの問題を発生させずにどのようにして頻繁なリリースを可能としますか？

A) リリースを自動化する
B) リリースを運用に任せる
C) リリースをとても小さくする
D) すべての変更を文書化しない

23 / 40
ある会社が、継続的デプロイメントを実装しています。

新機能のリリース時期を誰が決定するべきでしょうか？

A) ビジネス
B) 顧客
C) IT部門
D) 社内ユーザー

24 / 40
DevOpsプラクティスでは、運用管理のレベルを上げる**最適な**方法とされているのは、どれでしょうか？

A) 手動の運用オペレーションのすべてを自動化
B) 適切な役割と責任の定義
C) 管理手順（control procedure）の設計
D) 運用ガバナンスの改善

25 / 40
DevOpsにおけるインシデントの解決方法はどれでしょうか？

A) 問題管理チームにエスカレーションし、彼らがインシデントを解決するまでソリューションを作成する
B) インシデントを調査し、診断を実行してから、ワークアラウンド(根本原因の回避策)を特定して導入する
C) 関連するインシデントが以前に発生したことがあるかどうかを確認し、問題に対する類似する解決策を実行する
D) インシデントを追跡して最近のデプロイメントまで遡り、システムを以前の安定した状態にロールバックする

26 / 40
DevOpsでプロセスの欠陥が見つかった場合、どうすべきでしょうか？

A) すべての変更はバックログに記録されるべきであり、そうすることで、変更をプロジェクトまたはカイゼンのイベントでリリースできる。
B) 欠陥が見つかったら、できるだけ早く修正方法を見つけて実装する必要がある。
C) 修正方法を見つけて、変更管理者が承認し、優先順位に基づいてリリースする必要がある。
D) 修正方法を見つけて、継続的改善管理者が承認し、ただちにリリースする必要がある。
E) 修正のための変更が適切なスプリントに組み込まれるまで、修正を延期するべきである。

27 / 40
DevOpsチームにとって、動くソフトウェアの開発とデリバリーを成功させるための手段と**ならない**ものは、どれでしょうか？

A) プロジェクトの進行中に短期間でDevOpsチームを作り上げる
B) エラーが見つかったらすぐにそれを特定し、修正し、学習する
C) 組織の使命に沿ったDevOpsチームを編成する
D) 品質を組み込むことを主目的にしてソフトウェアのコードを作成する

28 / 40
DevOpsでは、現行作業の可視化（見える化）が推奨されています。

可視化によって実現される**２つ**の目標とはどれでしょうか？

答えを２つ選んでください。

A) プル型システムの構築
B) 作業の分割
C) コミットメントの育成
D) 非効率性の特定
E) 顧客への通知

29 / 40
仕掛（WIP）を制限する理由に**ならない**のはどれでしょうか？

A) 生産性の低下を軽減するため
B) 制約の解消を支援するため
C) フローのリズムを支援するため
D) リソースを有効活用するため

30 / 40
バックログアイテムを検討するとき、DevOpsチームが考慮すべき要件とは何ですか？

A) 非機能要件と機能要件の両方
B) 非機能要件と機能要件のどちらも考慮しない
C) 機能要件のみ
D) 非機能要件のみ

31 / 40
DevOpsチームが長期にわたって一緒に働くことの利点は何でしょうか？

A) チームがこれ以上プロセスを改善する必要がなくなる。
B) チームが、プロセスの変革や改善を行うために彼らの経験を活かすようになる。
C) チームがより独立して作業を開始するようになる。
D) 想定外の要求をより頻繁に処理する時間ができる。

32 / 40
チームは1週間のイテレーション（反復）で作業を進めており、ボトルネックが頻繁に発生しています。

ボトルネックが明らかになった後の**最も適当**な対応はどれでしょうか？

A) ボトルネックが明らかになった後、できるだけ早く解消する
B) ボトルネックが見つかったイテレーションの期間のみを長くする
C) バッチ内の通常のタスク数を制限してバッチサイズを縮小する
D) 可視化（見える化）ツールを仕掛（WIP）制限と共に使用する

33 / 40
組織的および技術的な変更にDevOpsを使用することで、カオスや制御不能に陥る可能性があるのは、どのような場合でしょうか？

A) 組織の中核ビジネスが情報テクノロジに大きく依存している場合
B) 組織が複雑であり、慢性的な問題を解決したいと考えている場合
C) 組織が新しいビジネスのアイデアや仮説をテストするために急速な変更を必要としている場合
D) 組織で使用されている情報テクノロジの変化の度合いが大きい場合

34 / 40
組織がDevOpsに注目するようになる理由はたくさんあります。

どのような場合に、企業はDevOpsに注目するようになりますか？

A) アジャイルのプラクティスがその組織に適していないと思われる場合
B) DevOps以外の方法では必要な結果を得られないと考えた場合
C) スクラムとリーンのプラクティスを導入することが完了した時

35 / 40
DevOpsを採用する際、困難を引き起こす可能性があるのはどれでしょうか？

A) 機能横断型チーム
B) 仮想化の限定的な使用
C) マイクロサービスアーキテクチャー

36 / 40
あるITシステムが、多くの従業員によって、単一のエンティティとして（単一のシステムを単一の開発体制で）、今も開発／維持されています。

DevOpsプラクティスの採用にあたって、どのような困難が予想されるでしょうか？

A) DevOpsチームを別々の責任分野に割り当てること
B) 組織構造を考慮して、機能横断型チームを編成すること
C) 下位互換性を考慮して、複数のAPIを保守／バージョン管理すること

37 / 40
カスタムソフトウェアの開発には時間がかかるため、必要な結果を迅速に得るために、商用オフザシェルフ(COTS)が使用されます。

COTSの**正しい**説明はどれでしょうか？

A) COTSでは、システムを構成するためのカスタマイズやスクリプトの作成が必要になる。
B) COTSは、戦略的ビジネスラインをサポートする目的で使用するべきである。
C) すぐに使用できるソフトウェアを利用する方が、一般に柔軟性が高い。
D) すぐに使用できるソフトウェアを利用する方が、一般的にはコストがはるかに低い。

38 / 40
密結合またはモノリシックのITアーキテクチャーで困難なのは何でしょうか？

A) ITアーキテクチャーそのものの変更や開発が困難である
B) アーキテクチャー内のサービスの変更を独立して実行すること
C) 現在のバージョンを無効にすることなく、新バージョンへと更新すること
D) すべてのコンポーネントの大規模移行への準備が完了するのを待つこと

39 / 40
各組織は、その組織で活用するDevOpsのプラクティスを選択してカスタマイズするよう推奨されています。組織に固有の問いを提起し、組織固有の答えを見つける必要があります。

これが妥当なアイデアである理由は何でしょうか？

A) DevOpsに関する出版物は、必ずしも読者が直面する現実を反映したものではなく、また困難や失敗を過少報告しているため
B) DevOpsを導入するために採用されるDevOpsエンジニアになる方法であるため
C) 自分の組織にDevOpsを導入する方法を最もよく知っているのは、管理チームであるため
D) DevOpsに関する出版物やイベントが少なすぎて、独立した意見を形成するのが困難であるため

40 / 40
ある組織が、従来型のITインフラストラクチャを利用しており、DevOpsを導入したいと考えています。

このような組織において、一般的なアプローチはどれでしょうか？

A) インストールして起動するソフトウェア製品のようにＤｅｖＯｐｓの導入を始める
B) 少なくともアセンブリと初期テストを実行する基本的なパイプライン作りから始めるには
C) 最適化が最もできそうな製品の選択から始める
D) 相互に疎結合されているシステムの特定から開始する
E) 一定の割合の作業時間を改善活動に割り当てることから始める

解答集

1 / 40
アジャイルの**誤った**考え方はどれでしょうか？

A) ビジネス側と開発側が、プロジェクト全体を通して一緒に作業を進める。
B) 変化に対応することは、計画に従うことより重要である。
C) 顧客の要件を確実に遂行することで顧客に満足してもらうことが優先事項である。
D) 動くソフトウェアは、進捗の第一の判断基準である。

A) 不正解。「ビジネス側と開発側の人は、プロジェクトで毎日一緒に作業を進めなければならない」とアジャイル・マニフェストに記載されています。
B) 不正解。変化に対応することは計画に従うことより優先されるとアジャイル・マニフェストに記載されています。
C) 正解。「顧客の要件を確実に遂行することで顧客に満足してもらうこと」は、アジャイル・マニフェストには記載されておらず、アジャイルの目的ではありません。（参考文献：A、第1.1.1章）
D) 不正解。動くソフトウェアを進捗の第一の判断基準とすることは、アジャイル・マニフェストの一部です。

2 / 40
米国立標準技術研究所（NIST）がクラウドコンピューティングに不可欠な特性として挙げて**いない**のはどれでしょうか？

A) Broad Network Access（ブロードバンドネットワークアクセス）
B) Pay-per-use System（従量制）
C) Rapid Elasticity（スピーディな拡張性）
D) Resource Pooling（リソースの共用）

A) 不正解。Broad Network Accessは、クラウドコンピューティングの不可欠な特性の1つとして、米国のNISTが挙げています。
B) 正解。Pay-per-use Systemは、使用されることは多いものの、米国のNISTが挙げている、クラウドコンピューティングに不可欠な特性の1つではありません。これ以外の種類の契約も可能です。（参考文献：A、第1.1.2章）
C) 不正解。Rapid Elasticityは、クラウドコンピューティングの不可欠な特性の1つとして、米国のNISTが挙げています。
D) 不正解。Resource Poolingは、クラウドコンピューティングの不可欠な特性の1つとして、米国のNISTが挙げています。

3 / 40
DevOpsの登場を振り返ってみたとき、ソフトウェア開発の関係者間の新たな取り組み方や会話の仕方によって、新たなITマネジメント方法が必要になってきました。その結果としてDevOpsが誕生しました。

この新しいやり取りの方法を見つけた関係者はだれでしょうか？

A) ビジネスと顧客
B) IT部門と顧客
C) ITの開発と運用

A) 不正解。ビジネス自体が顧客とのやり取りの仕方を変えた可能性はありますが、これによってDevOpsそのものが引き起こされたわけではありません。DevOpsは、IT部門とビジネス（顧客）の間の相互作用によって引き起こされました。
B) 正解。「最初にまず、ビジネス顧客と対話するための新しい方法が出現し、そしてアジャイル開発手法の適切な適用をしたことで、新しいITマネジメントの必要性が高まりました。」市場の変化に応じて、また人々の好みのブランドとの関わり方に応じて、顧客と対話するための新しい方法が必要となっています。また、アジャイル技法の適切な適用により、（IT面だけでなく）ビジネス面からアジャイルによる利益（リターン）をより認識するようになります。（参考文献：A、第1.1.3章）
C) 不正解。IT部門内の対話がきっかけとなってDevOpsが発展したわけではありません。

4 / 40
DevOpsは、リーン生産方式の原則とプラクティスを多く取り入れています。ITの中でのムダの1つが、「タスク切り替え」です。

これはリーン生産方式のどのムダを、ITの世界に言い換えたのでしょうか？

A) 動作
B) 作り過ぎ
C) 運搬
D) 待ち

A) 不正解。ITにおける引継ぎは、リーン生産方式における動作に相当します。
B) 不正解。ITにおける余分な機能は、リーン生産方式における作り過ぎに相当します。
C) 正解。ITにおけるタスクの切り替えは、リーン生産方式における運搬に相当します。（参考文献：A、第2.1.1章）
D) 不正解。ITにおける待ちは、リーン生産方式における待ちに相当します。

5 / 40
バリューストリーム・マップ（価値の流れ）の最も価値のある情報は、3つの重要な指標（メトリクス）から得られます。

これら3つの重要な指標の1つはどれでしょうか？

A) リードタイムとフローの組み合わせ
B) 完全正確率（percent complete and accurate; % C/A）
C) プロセスタイムをリードタイムで除算した結果
D) 達成価値からムダを減算した結果

A) 不正解。リードタイムは主要な指標ですが、フローは指標ではなく、リードタイムとプロセスタイムによって測定できるベロシティの概念です。
B) 正解。完全正確率（%C/A）は、最も価値ある情報の収集に役立つ、バリューストリーム・マップの主要な指標です。（参考文献：A、第3.1章）
C) 不正解。プロセスタイムとリードタイムはどちらも主要な指標すが、両者の比率が有用な尺度というわけではありません。
D) 不正解。達成価値からムダを減算した結果は、主要な指標ではありません。ただし、どちらもDevOpsの概念です。

6 / 40
DevOpsへ切り替えると、ＩＴに対するより大きな利益が期待されます。

DevOpsは、アジャイル、スクラム、リーンと何が違っているでしょうか？

A) DevOpsによって、DevとOpsとの間で新規および変更された製品のデリバリーが加速される。
B) DevOpsによって、新規および変更された製品を市場や顧客へデリバリーすることが加速される。
C) DevOpsによって、インフラストラクチャの変更に対して予算内で迅速な対応が保証される。
D) DevOpsによって、組織のバリューストリーム（価値の流れ）を妨げるインシデントへの迅速な対応が保証される。

A) 不正解。部門間での製品のデリバリーの加速は、DevOpsの結果の1つではありますが、顧客への価値のデリバリーではないため、ITの費用対効果の向上につながる、DevOpsに期待される価値ではありません。
B) 正解。DevOpsは、製品の顧客へのデリバリーを加速し、ビジネスによる高速かつ効率的な価値の実現を支援することで、ITの費用対効果の向上を可能にします。（参考文献：A、第1.2章）
C) 不正解。インフラストラクチャに対する変更は、ITからの内部的なニーズであり、予算に従って実行されるものでもあります。インフラストラクチャの変更だけで、ビジネスにとってのITの費用対効果が向上するわけではありません。
D) 不正解。これは、ITの費用対効果を向上させる方法ではありません。対応が早いほどユーザーエクスペリエンスが向上しますが、それだけで費用対効果が向上するわけではありません。

7 / 40
ある会社では、製品を市場へ投入するまでの時間（time-to-market）を数年から数ヶ月へ短縮したいと考えています。ある従業員がDevOpsの手法を検討するべきであると提案しています。

DevOpsへ変化することで、この時間を短縮できる理由は何でしょうか？

A) DevOpsによって開発と運用が1つのチームに統合され、従業員の人数がより少なくて済むようになるから
B) DevOpsチームは、よりコストが高く、時間外を含めてより多くの時間で働き、プロダクトをより早く市場に投入するから
C) DevOpsでは自己完結型の専任チームが編成され、製品への変化する要求に対処できる俊敏性（アジリティ）をより高めることができるから

A) 不正解。製品または製品ラインごとに、開発担当、運用担当、およびその他の専門家で構成された専任の自己完結型チームが編成されるため、通常は人が少なくなるのではなく、おそらく増員されます。さらに追記ですが、DevOpsの統合チームは、開発を素早くできる方法には必ずしもなりえませんが、開発者(Dev)が運用側(Ops)のニーズをより正確に理解できるので、デプロイメント時のソフトウェアをより堅牢にすることができます。
B) 不正解。自己完結型の専任チームはよりコストが高くなる可能性がありますが、DevOpsのプラクティスでは、構造的に仕事をより早く終えるために時間外労働を頼ろうとすることは決してありません。事実は全くの反対で、DevOpsは持続可能なペースとリズムを見つけ出すだそうと試みます。
C) 正解。専任で自己完結型の１つのチームが1つの製品に取り組むので、その製品に対する作業の不注意や中断が少なくなります。さらに、チームは専門家のサポートを待つ必要はなく、待ち時間という大きなムダが削減され、チームの製品化までの時間が短縮されます。（参考文献：A、第1.3.1章）

8 / 40
技術的負債を軽減する2つの一般的な方法とはどれでしょうか？

A) 正式な変更とリリース管理の手法
B) インシデント管理と要求実現の手法
C) 予算とリソースの増加
D) リファクタリングと、問題の直視

A) 不正解。この2つのプロセスは、技術的負債増加する可能性があります。
B) 不正解。この2つのプロセスは技術的負債減らすどころか、技術的負債の原因となる可能性があります。
C) 不正解。技術的負債の軽減に直接役立つわけではありませんが、開発者が多ければ、技術的負債を早く軽減できる可能性があります。
D) 正解。参考文献によると、一般的に使用されている2つの手法は、リファクタリングと、問題を直視し立ち向かう事です。（参考文献：A、第1.3章）

9 / 40
DevOpsが組織にもたらす大きな利点はどれでしょうか？

A) ビジネスシステムの脆弱性を排除する
B) 顧客のコストを削減する
C) 文化的な課題を軽減する

A) 正解。「DevOpsには、市場投入までの時間（time-to-market）を短縮し、技術的負債を軽減し、脆弱性を排除することが期待されます。」（参考文献：A、第1.3.3章）
B) 不正解。DevOpsプラクティスは、コストの増加につながる可能性もあり、顧客にとってのコスト削減に重点的に取り組むものではありません。
C) 不正解。DevOpsでは多様性あるチームが奨励されますが、これによって、文化的課題の軽減が保証されるわけではありません。

10 / 40
DevOpsはアジャイルの一部であるという考え方もあります。

DevOpsであれば答えが得られるものの、アジャイルでは答えが得られない質問はどれでしょうか？

A) 大きな市場シェアを獲得するのに十分な速さで、ソフトウェアを開発しリリースするには、どうするべきか？
B) 顧客の要求をより適切に理解するには、顧客との間でどのような関係が必要か？
C) リリースする製品を、便利で運用しやすいものにするには、どうするべきか？

A) 不正解。製品のリリースは、アジャイルとDevOpsの両方に含まれます。アジャイルは、最終製品の機能の迅速なリリースを支援します。DevOpsは、実際の顧客に対するより高い価値のデリバリー方法を探求します。
B) 不正解。顧客との関係と要件の収集は、アジャイルとDevOpsの両方に含まれます。プロダクトオーナは、顧客との密接な関係を維持することで、製品に価値が付加されるようにします。
C) 正解。開発の段階でリリースや運用の段階を考慮することは、必ずしもアジャイルの一部であるというわけではありませんが、DevOpsにおいて不可欠な部分です。（参考文献：A、第1.5.1章）

11 / 40
DevOpsで「バリューストリーム（価値の流れ）」という概念がとても重要である理由は何でしょうか？

A) バリューストリームは、従業員が自分の毎日のタスクを確認し、理解するのに役立つ。
B) バリューストリームは、as-isマップの分析やメトリック（業績指標）を向上するための取り組みに役立つ。
C) バリューストリームは、担当者が自分の担当する仕事をいつ行ったのかを特定するのに役立つ。
D) バリューストリームは、すべてのプロセスステップ（工程）を通してスムーズで均一な流れを実現するのに役立つ。
E) バリューストリームは、現在の作業手順をローカルな範囲内で最適化するのに役立つ。

A) 不正解。これは「作業の内容」であり、期待される結果の「理由」が欠如しています。
B) 不正解。仕事の最適化のためには、as-isマップを分析するだけでなく、to-beマップを開発する必要があります。それは、現在の仕事の進め方（プラクティス）とまったく違ったものになるかもしれません。
C) 不正解。顧客が期待していた価値をいつ受け取ったか、あるいはいつから受け取り始めたかを知ることは重要です。
D) 正解。「バリューストリーム」の概念によって、一連のプロセスのあるステップから次のステップへスムーズで均一な流れを作り出すことが確実にできます。（参考文献：A、第3.1章）
E) 不正解。バリューストリームは、部分最適化の罠に陥るのを防ぎながら、ボトルネックを特定して解消するのに役立ちます。

12 / 40
バリューストリーム・マップ（価値の流れ）の**最初の手順**は何でしょうか？

A) 要求仕様を作成する
B) 現行作業の文書化
C) プロセスの重要なステップを特定する

A) 不正解。固有のプロセスや組織にとって重要な手順を特定することを最初のステップとするべきです。
B) 不正解。現行作業の文書化は、重要なステップを特定した後にのみ実行するべきです。重要でないステップにある現行の作業は、バリューストリーム・マップの一部分になりません。
C) 正解。これは、バリューストリーム・マップの最初の手順です。組織は、バリューストリーム・マップを作りたいプロセスを眺めて、どの手順で作業が完了して価値を付加するかを特定する必要があります。（参考文献：A、第3.1章）

13 / 40
バリューストリーム・マップ（価値の流れ）は、バリューストリームにある効率の悪い所を明らかにします。

バリューストリーム・マップを作成するべき理由は何でしょうか？

A) ビジネスプロセスの最適化を支援するため
B) 十分に頑張って働いていない人を明らかにするため
C) 仕掛（WIP）中の作業をスピードアップさせるため
D) どのプロダクトラインを終了させるかを可視化するため

A) 正解。これこそが、バリューストリーム・マップを作成するべき理由であり、最も多くの価値がもたらされるところです。（参考文献：A、第3.1章）
B) 不正解。バリューストリーム・マップの目的は、誰が余剰人員か、誰を最初に解雇するか、あるいは誰が十分に仕事をしていないのかを判断することではなく、プロセスを最適化して、社内の人が質量ともに高い方法で生産を開始できるようにすることです。
C) 不正解。プロセスはスピードアップする可能性がありますが、作業自体が必ずしもスピードアップする必要はありません。 そうではなく、作業自体にエラーを無くし、最初から正し結果をだす（first-time-right）ようにすることで、品質管理のステップを省くことができます。 作業をスピードアップすることが目的ではなく、ムダを省き、ビジネスにさらなる価値を付加することが目標です。
D) 不正解。ビジネスがバリューストリーム・マップに基づいてビジネスラインやプロダクトラインの終了を決定する可能性はありますが、これがバリューストリーム・マップの目標ではありません。むしろ、バリューストリーム・マップは成長が期待される製品に対してのみ作成され適用されるべきです。

14 / 40
タスクには優先順位付けを行う必要があります。この優先順位付けはバリューストリーム（価値の流れ）の入口にある待ち行列で行われます。

この優先順位付けでしばしば問題を引き起しますが、その理由はなぜですか？

A) ここで、タスクを自動化するにデプロイメント・パイプラインをどのように構築するかを決めるためです。そのための時間がかかり、遅延の原因になるため。
B) バリューストリームにおける重要な指標を計測できるようにすることを、間違った方法や効率の悪い方法で行うことで問題を引き起こすため。
C) タスクの流れのボトルネックを特定するために仕掛（WIP）の制限の状況を示すことができる可視化（見える化）ツールの導入が間違って行われるため。
D) バリューストリームのas-isバージョンとto-beバージョンの開発に加えて、必要とされる変更の一覧を作成する必要があり、かなりの時間がかかるため。
E) 従来型のアプローチでは、仕事を開始する前に多くの判断や決定が必要となり、大きな遅れを引き起こす原因となるため。

A) 不正解。この手順はプロセスの後半で実行されるものであり、バリューストリームの入口にある待ち行列で問題が発生する原因にはなりません。
B) 不正解。これはバリューストリームの最初の手順の目的（測定指標の数をできるだけ多くすること）ではなく、この段階で実行するべきものではないため、このような問題が発生する原因にはなりません。
C) 不正解。これは、フローを遅延のない均等な状態にし、ボトルネックを明らかにする方法ではありますが、この段階で発生する問題ではありません。
D) 不正解。これは、バリューストリームのプロセスの後半で達成されるものであるため、上記の段階で問題が発生する原因にはなりません。
E) 正解。多くの従来型の手法を組織が使用し続け、作業に関するすべての判断が作業の開始前に行われる状況は、上記の問題を発生させる原因になります。（参考文献：A、第4.10章）

15 / 40
「デプロイメント・パイプライン」という概念の由来になっているアイデアはどれでしょうか？

A) 液体が流れるパイプライン
B) 自動車工場などの組み立てライン
C) 並列パイプラインを使用する最新型プロセッサー
D) 複数の組立ラインを使用するというアイデア
E) さまざまな仕事をする人たちを配置するプロセス

A) 不正解。これは、よくある誤った意見であり、この概念の正しい意味ではありません。
B) 不正解。これは、よくある誤った意見であり、この概念の正しい意味ではありません。
C) 正解。HumbleとFarleyは造語にあたって、はるかに高速で結果を生成できる最新型プロセッサーのアーキテクチャーから、パイプラインのアイデアを採用したと説明しています。（参考文献：A、第3.2章）
D) 不正解。これは、よくある誤った意見であり、この概念の正しい意味ではありません。
E) 不正解。これは、よくある誤った意見であり、この概念の正しい意味ではありません。

16 / 40
デプロイメント・パイプラインを導入する際は、いろいろな問題が発生します。まず最初に、本番稼働環境で安定した運用を保証するために、事前に準備開発された十分なテストを実施することができません。

どの解決策が**もっとも効果的に**この問題に対処できるでしょうか？

A) パイプラインを構築し、できるだけ多くの自動化を行うが、適切なテストがすべて揃うまで使用しない。
B) できるだけ早く解決しなければならない技術的負債として、コードのテストカバレッジを向上させる。
C) 開発したテストをパイプラインで実行し、潜在的な問題が本番環境で顕在化したときに対処する。
D) パイプラインの目的を、記述したコードをテストや品質保証（QA）のチームだけにデリバリーするためのインテグレーションシステムとする。

A) 不正解。パイプラインを構築し、すべてのテストが準備できるまで実行しないというのは、時間の浪費によるビジネス上の損失が発生します。最も重要なテストを最初に行い、新しいテストを継続的に生成しながらカバレッジを向上させるという反復的なアプローチを試みるべきです。
B) 正解。コードのテストカバレッジを向上させながらテストすることが、この問題に対する唯一の解決策です。（参考文献：A、第3.2章）
C) 不正解。少ないテスト数でパイプラインが実行されると、テストのカバレッジが低くなり、本番環境で多くの問題が発生する可能性があります。これは、パイプラインを採用することの妨げとなりまる。
D) デプロイメント・パイプラインは、動くコードをテストやQAチームだけでなく本番環境へデリバリーすることを目的としているために、この目的に合っていません。

17 / 40
優れたバージョン管理システムは、DevOpsにおける高いパフォーマンスを予測する最大の要素の1つです。

バージョン管理を適切に適用するには、何が必要でしょうか？

A) 情報とその構成に取り組む作業に対する文化を変えること
B) 変更を導入する速さの大幅な向上
C) カオスや不安定性の本番環境への意図的な注入
D) 正式で自動化された変更管理プロセスの使用

A) 正解。バージョン管理によって、運用のシステムのすべてに関係する構成要素をコントロールできるようになります。これは他のツールでは不可能です。バージョン管理を成功させるには、情報と構成に関する作業文化を変える必要があります。（参考文献：A、第3.3章）
B) 不正解。仮想クラウドテクノロジの利用に伴い、自動化の程度がこの数年で大幅に上昇し、変更が導入される速度も速くなりましたが、これは、バージョン管理に必須の原則ではありません。
C) 不正解。反脆弱性（アンチフラジャイル）に関するDevOpsの優れたプラクティスの1つは、運用環境へのカオスと不安定性の意図的な注入です。この技法は、game day、chaos monkey、simian armyなどのいくつもの名前で知られています。これらのどれも、バージョン管理に必須の原則ではありません。
D) 不正解。ITインフラストラクチャの脆弱性に対処するために、組織によっては、変更の流れを構造化し、導入に伴うリスクを最小限に抑えるように設計された、正式かつ自動化された変更管理プロセスを使用します。しかし、バージョン管理の原則がそのような場合に必須というわけではありません。

18 / 40
DevOpsでは、迅速な移行とアプリケーションの信頼性の維持の適切なバランスを見つけることが重要です。

バージョン管理はこれをどのような方法でサポートしますか？

A) 不要なファイルや文書をチームのメンバー全員が自由に削除できるようにする
B) 少人数の独立した自己完結型の開発チームの編成を認める
C) ムダを排除または削減し、プロセスを最適化する専用ツールを適用する

A) 正解。バージョン管理によって、重要な情報や製品を誤って失うリスクがなくなり、チームのメンバー全員が、不要なファイルや文書を自由に削除できるようになります。（参考文献：A、第3.3章）
B) 不正解。小規模で自己完結型の多様性があるチームの編成は、DevOpsにおいて重要なアイデアではありますが、バージョン管理がシステムのアジリティや信頼性をサポートする方法ではありません。
C) 不正解。これは、リーン生産方式のITへの実践的な応用です。専用ツールを使用してムダを特定してから、他の何らかの専用ツールを適用してムダを排除あるいは削減します。ただし、これはバージョン管理でシステムのアジリティや信頼性をサポートする方法ではありません。

19 / 40
構成管理にはどのようなメリットがあるでしょうか？

A) チームのすべてのメンバーがリスクなく不要なファイルを削除できる。
B) 主要なチームのメンバーが不在でも問題が発生するのを回避できる。
C) 変更されたコード、変更した人、変更された日付を、チームのメンバーが確認できる。

A) 不正解。バージョン管理によって、チームのメンバーが自由に情報を削除できるようになります。また、問題が発生した場合は以前のバージョンに復元することができます。
B) 正解。すべての変更が構成管理によってコントロールされている場合、必要に応じて、システムは以前の安定した状態へ自動的に復元されます。さらに、チームの主要なメンバーがいなくなっても、その人のナレッジが失われることなく、構成情報の中に維持されます。（参考文献：A、第3.4章）
C) 不正解。いつ誰が何を変更したかという記録は、バージョン管理の一部であり、構成管理の一部ではありません。

20 / 40
構成管理によって、作業員の人員を増やすことなく、ITインフラストラクチャやソフトウェアシステムをスケール（規模の拡大）させることが可能になります。

そのようにスケールされた（拡張された）システム環境では、理想的にはどのように変更が行われるべきでしょうか？

A) 継続的インテグレーションを通じた変更
B) 完全にコントロールされたスクリプトによる変更
C) 自動化テストを通じた変更
D) デプロイメント・パイプラインを通じた変更

A) 不正解。継続的インテグレーションは、デプロイメント・パイプライン導入の第2段階です。しかし、このためにシステム管理者が本番環境で何らかの変更が許される唯一の方法にはなりえません。従って、理想的にはこれらすべて自動化スクリプトによって行われるべきです。
B) 正解。実際のところ、システム管理者であっても本番環境へのアクセス権限を持つべきではないと言えます。完全にコントロールされた（そして自動化された）スクリプトを通じた変更以外に、どのような変更も許可されるべきではありません。（参考文献：A、第3.3、3.4章）
C) 不正解。テストの自動化は、デプロイメント・パイプライン導入の第3段階です。しかし、システム管理者が本番環境で何らかの変更が許される唯一の方法にはなりえません。 従って、理想的にはこれらすべて自動化スクリプトによって行われるべきです。
D) 不正解。抽象的な言い方をすれば、正しく機能し完全に自動化されたデプロイメント・パイプラインとは、ソフトウェアをバージョン管理からユーザーの手に渡せるようにするための自動化プロセスとして形となって出現したもの（manifestation）です。ただし、手動のデプロイメント・パイプラインも可能ですが、システム管理者が本番環境で何かの変更が許される方法については言及されていません。従って、理想的にはすべての変更は自動化スクリプトによって行われるべきです。

21 / 40
明確な完了の定義（DoD）はDevOpsにおいて不可欠であり、顧客にとっての価値が考慮されます。

DevOpsにおける完了の正しい説明はどれでしょうか？

A) 要件が構築された段階で、その要件は完了となる。
B) そのコードがテストを終えた時、その要件は完了となる。
C) 製品が受け入れられた段階で、要件は完了となる。
D) 製品が本番環境にデプロイされた時、その要件は完了となる。

A) 不正解。顧客に価値を提供した時を物事の完了とみなします。ビルドとは、DevOpsパイプラインの一つの段階にすぎず、まだ価値が提供されていません。
B) 不正解。顧客に価値を提供した時を物事の完了とみなします。テストとは、DevOpsパイプラインの一つの段階にすぎず、まだ価値が提供されていません。
C) 不正解。顧客に価値を提供した時を物事の完了とみなします。受け入れテストとは、DevOpsパイプラインの一つの段階にすぎず、まだ価値が提供されていません。
D) 正解。顧客に価値を提供した時を物事の完了とみなします。これは、製品が本番環境に入った時にあてはまります。（参考文献：A、第3.5章）

22 / 40
従来のプラクティスでは、変更点が文書化されていない、システムが完全にバックアップされていない、またはシステムの以前の状態が保存されていないなど、リリースの段階で多くの問題が発生する可能性があります。

DevOpsは、 これらの問題を発生させずにどのようにして頻繁なリリースを可能としますか？

A) リリースを自動化する
B) リリースを運用に任せる
C) リリースをとても小さくする
D) すべての変更を文書化しない

A) 正解。自動化は、リリースを頻繁に行い、リリースプロセスを日常業務の一つにするための重要な要素です。バックアップ、文書化、およびロールバックのすべての人的要因が自動化されると、リリースに関する問題の発生する可能性は劇的に減少します。リリースが正しく実行できない場合、システムはチームに警告をだします。（文献：A、第4.1章）
B) 不正解。運用に管理を任せても、ここで言及された問題を防ぐことはできません。運用に着目し、運用プラクティスに開発を統合することで、リリースプロセスのどこに自動化や標準化が必要とされるかが明確になることがあります。リリースを運用に引き継ぐだけでは、言及されている問題は解決できません。
C) 不正解。リリースのサイズは、リリースを日常業務の一部にするための自動化ほど重要ではありません。リリースのサイズを小さくしても、言及されている問題を必ずしも予防できるわけではありません。頻繁にリリースするとリリースが小さくなります。
D) 不正解。これは、問題を悪化させることになります。（可能であれば自動化された）バージョン管理システムを導入すれば、不適切な文書化によって引き起こされる問題の防止に役立ちます。

23 / 40
ある会社が、継続的デプロイメントを実装しています。

新機能のリリース時期を誰が決定するべきでしょうか？

A) ビジネス
B) 顧客
C) IT部門
D) 社内ユーザー

A) 正解。継続的デプロイメントを実践する場合、使用する新機能のリリースは、ビジネス上の判断になります。新しい機能のデプロイを前もって行っておいた後に、ビジネス側が機能スイッチをオンにする時期を判断をすることもあります。IT部門が自分たちの独自のテンポでデプロイし、ビジネス部門がいつ新機能をリリースすべきかを判断します。（参考文献：A、第 4.1章）
B) 不正解。このステークホルダーは判断において重要ではありますが、リリースは主としてビジネス上の判断です。
C) 不正解。このステークホルダーは判断において重要ではありますが、リリースは主としてビジネス上の判断です。
D) 不正解。このステークホルダーは判断において重要ではありますが、リリースは主としてビジネス上の判断です。

24 / 40
DevOpsプラクティスでは、運用管理のレベルを上げる**最適な**方法とされているのは、どれでしょうか？

A) 手動の運用オペレーションのすべてを自動化
B) 適切な役割と責任の定義
C) 管理手順（control procedure）の設計
D) 運用ガバナンスの改善

A) 正解。できるだけ多くの運用を自動化することで、すべての運用が瞬時に実行され、均一になるため、完全なコントロールが可能になります。運用オペレーションが最適化されていないと、自動化の変更によって、将来の運用オペレーションがすべて変更されることになります。（参考文献：A、第 4.1.3章）
B) 不正解。役割と責任の定義は重要ですが、それがすべての運用オペレーションに対するコントロールに、直接的に影響するわけではありません。
C) 不正解。手順を設計するだけではあまり意味がありません。手順を設計したら自動化するべきであり、そうすることで、エラーの発生を防止して、すべての運用を均一にすることができます。
D) 不正解。望ましいことではありますが、ガバナンスだけでそれ以上のコントロールが可能になるわけではありません。自動化によって、完全なコントロールが保証されます。

25 / 40
DevOpsにおけるインシデントの解決方法はどれでしょうか？

A) 問題管理チームにエスカレーションし、彼らがインシデントを解決するまでソリューションを作成する
B) インシデントを調査し、診断を実行してから、ワークアラウンド(根本原因の回避策)を特定して導入する
C) 関連するインシデントが以前に発生したことがあるかどうかを確認し、問題に対する類似する解決策を実行する
D) インシデントを追跡して最近のデプロイメントまで遡り、システムを以前の安定した状態にロールバックする

A) 不正解。この解決策が有効なワークアラウンドにつながるかもしれませんが、その場合は、本当の問題が解決されない可能性があります。
B) 不正解。ワークアラウンドを導入しても、インシデントは解決されません。これは、正しい方法ではありません。
C) 不正解。この解決策はおそらくはワークアラウンドであり、望ましい方法ではありません。誰かが問題を解決してくれるのを待つべきではありません。
D) 正解。参考文献には、「インシデントを追跡して最近のデプロイメントまで遡る場合、パイプラインコントロールシステムによって、以前の既知の安定した状態に自動的にロールバックされる」と説明されています。自動化されているか手動で実行されるかにかかわらず、このプロセスで問題を解決するべきです。（参考文献：A、第4.1章）

26 / 40
DevOpsでプロセスの欠陥が見つかった場合、どうすべきでしょうか？

A) すべての変更はバックログに記録されるべきであり、そうすることで、変更をプロジェクトまたはカイゼンのイベントでリリースできる。
B) 欠陥が見つかったら、できるだけ早く修正方法を見つけて実装する必要がある。
C) 修正方法を見つけて、変更管理者が承認し、優先順位に基づいてリリースする必要がある。
D) 修正方法を見つけて、継続的改善管理者が承認し、ただちにリリースする必要がある。
E) 修正のための変更が適切なスプリントに組み込まれるまで、修正を延期するべきである。

A) 不正解。大規模な変更イベントや改善イベントは便利かもしれませんが、継続的改善の一環としてのプロセスに対する定期的な変更は、できるだけ迅速に実行するべきです。
B) 正解。検出されたときにプロセスの欠陥を修正しておかないと、必要以上に多くの問題が発生することになります。「したがって、DevOpsでは、プロセスで特定されたすべての欠陥はただちに解消するべきであるという、異なるアプローチを使用しています。」（参考文献：A、第4.1章）
C) 不正解。変更管理者は存在しません。修正は直ちに実装されるべきです。
D) 不正解。修正を承認する継続的改善管理者は存在しません。
E) 不正解。延期は妥当なアイデアではありません。システムが非効率的になり、作業に悪影響を及ぼします。

27 / 40

DevOpsチームにとって、動くソフトウェアの開発とデリバリーを成功させるための手段と**ならない**ものは、どれでしょうか？

A) プロジェクトの進行中に短期間でDevOpsチームを作り上げる
B) エラーが見つかったらすぐにそれを特定し、修正し、学習する
C) 組織の使命に沿ったDevOpsチームを編成する
D) 品質を組み込むことを主目的にしてソフトウェアのコードを作成する

A) 正解。DevOpsチームは時間をかけて形成されます。そうすることで、経験を活用した、新しいソフトウェアの迅速なデリバリーと継続的なイノベーションが可能になります。（参考文献：A、第4.2章）
B) 不正解。バッチが小さいほど、エラーを容易に発見し、すぐに修正できます。DevOpsは、エラーを迅速に発見してただちに修正し、プロセスを最適化する、リーン生産方式のプロセスを取り入れています。
C) 不正解。DevOpsの主な利点の1つは、チームが自らの作業を特定の組織の目標に合わせることで、ビジネスに価値がもたらされることです。
D) 不正解。品質の作りこみは、DevOpsが採用している、リーン生産方式の1つの特徴です。品質を考慮してコーディングすると時間がかかる可能性はありますが、発見されるバグが少なくなり、システムがより堅牢に構築されるため、最終的には価値が付加されます。

28 / 40
DevOpsでは、現行作業の可視化（見える化）が推奨されています。

可視化によって実現される**２つ**の目標とはどれでしょうか？

答えを２つ選んでください。

A) プル型システムの構築
B) 作業の分割
C) コミットメントの育成
D) 非効率性の特定
E) 顧客への通知

A) 正解。これは、可視化をするメリットです。可視化によってプル型システムが構築でき、結果としてワークフローが改善され、ダウンタイムが短縮し、調整の必要性が削減できます。可視化は非効率性の特定にも役立ちます。（参考文献：A、第4.3章）
B) 不正解。可視化しなくても、チームが効果的に作業を分割することはできます。コミットメントはDevOpsを成功させるにあたって重要ではありますが、可視化によって醸成されるものではありません。「顧客の声」あるいは「ビジネスの声」としての役割を果たす人は、顧客の状況を最新のものに保つすべきです。
C) 不正解。可視化しなくても、チームが効果的に作業を分割することはできます。コミットメントはDevOpsを成功させるにあたって重要ではありますが、可視化によって醸成されるものではありません。「顧客の声」あるいは「ビジネスの声」としての役割を果たす人は、顧客の状況を最新のものに保つすべきです。
D) 正解。これは、可視化をするメリットです。可視化によってプル型システムが構築でき、結果としてワークフローが改善され、ダウンタイムが短縮し、調整の必要性が削減できます。可視化は非効率性の特定にも役立ちます。（参考文献：A、第4.3章）
E) 不正解。可視化しなくても、チームが効果的に作業を分割することはできます。コミットメントはDevOpsを成功させるにあたって重要ではありますが、可視化によって醸成されるものではありません。「顧客の声」あるいは「ビジネスの声」としての役割を果たす人は、顧客の状況を最新のものに保つすべきです。

29 / 40
仕掛（WIP）を制限する理由に**ならない**のはどれでしょうか？

A) 生産性の低下を軽減するため
B) 制約の解消を支援するため
C) フローのリズムを支援するため
D) リソースを有効活用するため

A) 不正解。WIPを制限することで、チームのメンバーが1つの項目に集中的に取り組むようになり、結果として、タスクの切り替えによる不要な中断がなくなって、最終的には生産性が向上します。
B) 不正解。WIPを制限すると、エラーの迅速な修正と、タスク間のシステムを最適化することが容易になります。
C) 正解。これは、WIPの制限のメリットの1つではありません。（参考文献：A、第4.4、4.5章）
D) 不正解。WIPを制限することで、チームのメンバーが1つの項目に集中的に取り組むようになり、結果として、タスクの切り替えによる不要な中断がなくなって、最終的にはリソースの有効活用が実現します。

30 / 40
バックログアイテムを検討するとき、DevOpsチームが考慮すべき要件とは何ですか？

A) 非機能要件と機能要件の両方
B) 非機能要件と機能要件のどちらも考慮しない
C) 機能要件のみ
D) 非機能要件のみ

A) 正解。機能的要件と非機能的要件の両方を考慮する必要があります。（参考文献：A、第4.6章）
B) 不正解。機能的要件と非機能的要件の両方を考慮する必要があります。
C) 不正解。機能的要件と非機能的要件の両方を考慮する必要があります。
D) 不正解。機能的要件と非機能的要件の両方を考慮する必要があります。

31 / 40
DevOpsチームが長期にわたって一緒に働くことの利点は何でしょうか？

A) チームがこれ以上プロセスを改善する必要がなくなる。
B) チームが、プロセスの変革や改善を行うために彼らの経験を活かすようになる。
C) チームがより独立して作業を開始するようになる。
D) 想定外の要求をより頻繁に処理する時間ができる。

A) 不正解。DevOpsチームは常に改善に取り組みます。それが継続的改善というものです。より長く協力して働くチームは、より自信を持って日常業務のやり方を変えプロセスの変革を進めるようになるでしょう。
B) 正解。一緒に働く期間が長いDevOpsチームは、彼らの経験をこれからの開発に活かすようになり、またより迅速なデリバリーおよびプロセスの変革を行うような機会を持ちます。（参考文献：A、第4.9章）
C) 不正解。一緒に働くことが、チームがより独立して働くことを意味するわけではありません。チームには、方向性を堅持するという、組織としての使命があります。それ以外の点においては、最初の段階から自己完結型のチームとして働くべきです。
D) 不正解。バックログ項目とその優先順位を考慮して、バッチが計画されます。多数の予期しない要求を処理することがDevOpsの目標になることはありません。要求をバックログに入れて優先順位を設定し、次のイテレーションで処理されるようにします。

32 / 40
チームは1週間のイテレーション（反復）で作業を進めており、ボトルネックが頻繁に発生しています。

ボトルネックが明らかになった後の**最も適当**な対応はどれでしょうか？

A) ボトルネックが明らかになった後、できるだけ早く解消する
B) ボトルネックが見つかったイテレーションの期間のみを長くする
C) バッチ内の通常のタスク数を制限してバッチサイズを縮小する
D) 可視化（見える化）ツールを仕掛（WIP）制限と共に使用する

A) 正解。ボトルネックの原因を解消する方法をできるだけ早く見つける必要があります。ボトルネックが解消されれば、取り決められていたイテレーションで作業が完了する場合もありますが、これは一般的ではありません。（参考文献：A、第4.11章）
B) 不正解。このケースは、問題に対処する最善の方法ではありません。スクラムでは、時にイテレーションの期間を延長することもあります。ただし、DevOpsでは、スクラムよりもリズムの確立に重点を置いています。したがって、最後の手段としてイテレーションの期間を延長すべきです。
C) 不正解。バッチサイズを制限すると、ボトルネックの原因となる問題の特定が容易になります。ただし、これはDevOpsプラクティスの一つとして導入すべきです。チームがボトルネックを特定した後にだけ実行されるべきものではありません。
D) 不正解。これはボトルネックの特定に役立ちますが、チームがボトルネックを特定した後で導入するべきではありません。

33 / 40
組織的および技術的な変更にDevOpsを使用することで、カオスや制御不能に陥る可能性があるのは、どのような場合でしょうか？

A) 組織の中核ビジネスが情報テクノロジに大きく依存している場合
B) 組織が複雑であり、慢性的な問題を解決したいと考えている場合
C) 組織が新しいビジネスのアイデアや仮説をテストするために急速な変更を必要としている場合
D) 組織で使用されている情報テクノロジの変化の度合いが大きい場合

A) 不正解。中核ビジネスが情報テクノロジに大きく依存している組織は、DevOpsに興味を持つようになります（なるべきです）。
B) 正解。複雑な状況では、DevOpsは大きな利益をもたらさない可能性があり、短期間で成果を上げられる保証はありません。慢性的な問題は、十分に注意しながら慎重に解決するべきです。DevOpsをすべての問題を解決してくれる魔法の薬と考えるべきではありません。（参考文献：A、第5.1章）
C) 不正解。主要ビジネスが新しいビジネスのアイデアや仮説をテストするために急速な変更を必要としていると、組織は、DevOpsに興味を持つようになります（なるべきです）。
D) 不正解。使用している情報テクノロジで発生する変化の度合いが多いと、組織は、DevOpsに興味を持つようになります（なるべきです）。

34 / 40
組織がDevOpsに注目するようになる理由はたくさんあります。

どのような場合に、企業はDevOpsに注目するようになりますか？

A) アジャイルのプラクティスがその組織に適していないと思われる場合
B) DevOps以外の方法では必要な結果を得られないと考えた場合
C) スクラムとリーンのプラクティスを導入することが完了した時

A) 不正解。これは、組織がDevOpsプラクティスを真剣に再検討するきっかけになるはずです。アジャイルのプラクティスがビジネスにとって適切ではないと思われる場合、DevOpsプラクティスを検討しても害はありません。ただし、DevOpsプラクティスにはアジャイルの多くの概念が含まれているため、DevOpsがその組織に適していない可能性もあります。
B) 正解。企業がDevOpsに関心を示すようになるのは、効果を高めるための他のすべての実証済み手法が、もはや有意な結果をもたらさなくなったときです。これがDevOpsプラクティスの使用を始める最良の理由とはなりませんが、他に有効な方法がないのであれば、DevOpsを必ず検討するべきです。（参考文献：A、第5.1章）
C) 不正解。DevOpsプラクティスを開始するにあたって、スクラムやリーンのプラクティスの導入を待つ必要はありません。事実、多くのDevOpsプラクティスがスクラムやリーンの概念を拠り所にしているために、それぞれがシームレスに機能します。DevOpsでは組織の中で有効に機能するものであれば何でも選択することができます。

35 / 40
DevOpsを採用する際、困難を引き起こす可能性があるのはどれでしょうか？

A) 機能横断型チーム
B) 仮想化の限定的な使用
C) マイクロサービスアーキテクチャー

A) 不正解。DevOpsチームは機能横断型です。機能横断型のチームにすることで、組織がDevOpsを円滑に始めることができます。
B) 正解。仮想化をほとんど使用していない組織は、DevOpsのプラクティスを導入する上で困難に直面することになります。（参考文献：A、第5.1章）
C) 不正解。DevOpsのいくつかの一般的な思想（イデオロギー）から誕生したマイクロサービスアーキテクチャーによって、組織は、DevOpsを円滑に始めることができます。

36 / 40
あるITシステムが、多くの従業員によって、単一のエンティティとして（単一のシステムを単一の開発体制で）、今も開発／維持されています。

DevOpsプラクティスの採用にあたって、どのような困難が予想されるでしょうか？

A) DevOpsチームを別々の責任分野に割り当てること
B) 組織構造を考慮して、機能横断型チームを編成すること
C) 下位互換性を考慮して、複数のAPIを保守／バージョン管理すること

A) 正解。DevOpsのプラクティスを導入する上での大きな障害は、モノリシックで密結合のITアーキテクチャです。小さなチームにするには、それぞれのチームに個別の責任範囲を割り当てることができることが必要となります。数十から数百人の従業員が単一エンティティとして対象となるITシステムを開発／保守している状況では、個々の独立したチームのために複数の部品に分割し、非同期で作業を進めるのは困難でしょう。（参考文献：A、第5.1章）
B) 不正解。機能横断的なチームを編成する障害となるものはありません。
C) 不正解。モノリシックアプリケーションでは、開発者はクラス名とAPIだけを変更することになります。マイクロサービスでは、開発者は下位互換性を維持するために、APIのバージョン番号を変更し、複数のAPIをメインテナンスする必要があります。このケースでは、まだマイクロサービスが実装されていないので、これは予期される問題ではありません。さらに、バージョン管理用のシステムが必要になる可能性がありますが、これは問題ではなく進歩であり、予想される結果であり、最終的にはDevOpsのプラクティスを導入することのメリットです。

37 / 40
カスタムソフトウェアの開発には時間がかかるため、必要な結果を迅速に得るために、商用オフザシェルフ(COTS)が使用されます。

COTSの**正しい**説明はどれでしょうか？

A) COTSでは、システムを構成するためのカスタマイズやスクリプトの作成が必要になる。
B) COTSは、戦略的ビジネスラインをサポートする目的で使用するべきである。
C) すぐに使用できるソフトウェアを利用する方が、一般に柔軟性が高い。
D) すぐに使用できるソフトウェアを利用する方が、一般的にはコストがはるかに低い。

A) 正解。スクリプトを作成することで、この種のソフトウェアの構成が可能になります。ただし、制限がある場合もあり、スクリプトを作成したとしても、システムの構成には時間がかかります。（参考文献：A、第5.2章）
B) 不正解。COTSソフトウェアが戦略的ビジネスラインを自動的にサポートするわけではありません。サポートのためのスクリプトの作成が必要になります（サポートしている場合）。戦略的ビジネスラインへのCOTSの利用は推奨されません。
C) 不正解。COTSソフトウェアでは、常に柔軟性が制限されます。必ずしも、チームの希望や要件に沿った方法で構成できるわけではありません。
D) 不正解。このようなソフトウェアの構成には時間がかかるため、労力もコストも余分にかかります。さらには、ビジネスで実際に必要とされる価値が提供されない可能性があります。

38 / 40
密結合またはモノリシックのITアーキテクチャーで困難なのは何でしょうか？

A) ITアーキテクチャーそのものの変更や開発が困難である
B) アーキテクチャー内のサービスの変更を独立して実行すること
C) 現在のバージョンを無効にすることなく、新バージョンへと更新すること
D) すべてのコンポーネントの大規模移行への準備が完了するのを待つこと

A) 正解。これは、密結合ITアーキテクチャーの問題です。アーキテクチャーが大規模で密結合であるほど、何かを変更し、それと同時に、その変更のアーキテクチャーの他の部分への影響を把握するのが困難になります。（参考文献：A、第5.3章）
B) 不正解。これは、問題の解決策です。すべてのサービスを独立して変更できるようになると、そのアーキテクチャーは密結合とは言えなくなります。
C) 不正解。アーキテクチャーが密結合ではない場合でも、これが課題になる可能性があります。
D) 不正解。大規模な移行が完了し、すべてのコンポーネントを移行できる状態にする必要があるからと言って、必ずしも密結合であるわけではありません。これは、どのような種類のITアーキテクチャーでも発生する可能性があります。

39 / 40
各組織は、その組織で活用するDevOpsのプラクティスを選択してカスタマイズするよう推奨されています。組織に固有の問いを提起し、組織固有の答えを見つける必要があります。

これが妥当なアイデアである理由は何でしょうか？

A) DevOpsに関する出版物は、必ずしも読者が直面する現実を反映したものではなく、また困難や失敗を過少報告しているため
B) DevOpsを導入するために採用されるDevOpsエンジニアになる方法であるため
C) 自分の組織にDevOpsを導入する方法を最もよく知っているのは、管理チームであるため
D) DevOpsに関する出版物やイベントが少なすぎて、独立した意見を形成するのが困難であるため

A) 正解。参考文献の数が膨大であるからといって、必ずしもチームがあらゆる困難に備えられるようになるわけではなく、失敗という現実に直面することになる可能性があります。情報を絞り込み、組織の状況に最も該当するものを参考にすることが重要です。（参考文献：A、第5.6章）
B) 不正解。DevOpsは「導入」できるものではなく、エンジニアを雇用したからといって、この新しい秩序がITに持ち込まれるわけではありません。
C) 不正解。DevOpsは導入するようなものではありません。
D) 不正解。組織が判断する際の参考になる書籍やイベントは数多く存在します。

40 / 40
ある組織が、従来型のITインフラストラクチャを利用しており、DevOpsを導入したいと考えています。

このような組織において、一般的なアプローチはどれでしょうか？

A) インストールして起動するソフトウェア製品のようにＤｅｖＯｐｓの導入を始める
B) 少なくともアセンブリと初期テストを実行する基本的なパイプライン作りから始めるには
C) 最適化が最もできそうな製品の選択から始める
D) 相互に疎結合されているシステムの特定から開始する
E) 一定の割合の作業時間を改善活動に割り当てることから始める

A) 不正解。DevOpsは、インストールして起動できるソフトウェア製品ではありません。
B) 不正解。これはDevOpsを開始するためのアプローチではありません。 これは、ストリームの自動化ができそうな部分からデプロイメント・パイプラインの構築を推進するために必要です。
C) 不正解。これは、バリューストリーム・マップ（価値の流れ）を作り始めるためのアプローチです。
D) 正解。DevOpsは、どこからでも、ビジネスの現状に関係なく開始できます。疎結合のシステムを特定することが最初の手順です。（参考文献：A、第5.6章）
E) 不正解。これは、技術的負債に関するものです。

評価

次の表に、本模擬試験問題の正解を示します。

番号	正解	番号	正解
1	C	21	D
2	B	22	A
3	B	23	A
4	C	24	A
5	B	25	D
6	B	26	B
7	C	27	A
8	D	28	A & D
9	A	29	C
10	C	30	A
11	D	31	B
12	C	32	A
13	A	33	B
14	E	34	B
15	C	35	B
16	B	36	A
17	A	37	A
18	A	38	A
19	B	39	A
20	B	40	D

EXIN の連絡先

www.exin.com