SOLUTION BRIEF
2017 年 5 月

継続的テストの要素

エンドツーエンドのテスト・エコシステムを使用して、より優れたソフトウェアをより短期間で構築するために必要な知識

ca technologies

概要

課題

ユーザは優れたソフトウェアを必要としているため、バグのあるアプリケーションは許容できません。多くの組織は、ユーザのニーズをより正確に反映したより優れたソフトウェアを提供するために、継続的テストモデルに移行しています。しかし、旧式のソフトウェア・デリバリの手法がボトルネックとなり、プロジェクトの遅延や品質に悪影響を及ぼしています。ソフトウェアのテストもそのようなボトルネックの 1 つです。低品質の要件によってコードが貧弱になり、テストの設計と実行プロセスに引き継がれます。従来のテスト方法は、手作業で行うため時間がかかりすぎて、現在求められる開発スピードに追いつくことができません。

ビジネス・チャンス

欠陥を減らしながらソフトウェア開発を加速するには、新しい継続的テストの手法が必要です。テストは、開発ライフサイクルを通じてすべてのチームが連携して関わる、機能の枠を超えたエンドツーエンドのプロセスになっています。継続的テストはアジャイル開発の手法をテストと QA のプロセスに組み込んだもので、効率が大幅に向上します。その秘訣は自動化にあり、既存の統合と継続的デリバリのプロセスを組み合わせたエンドツーエンドのテスト・ソリューションが必要です。このようなソリューションがなければ、欠陥や手動プロセスを排除して、開発ライフサイクル全体で完璧な継続性を確保することはできません。

品質と正確性

継続的テストでは、要件を定義した直後から品質を確保し、システムに追加される前の開発レベルで各コンポーネントの有効性を確認できます。そのため、最初から反復型 QA が行えます。ユーザのニーズの変化による欠陥や誤解は即座に解決できるため、終盤になってからの再作業や最終的なユーザ・エクスペリエンスの低下を回避できます。継続的テストでは、開発の最初の段階からソフトウェア品質が向上し、デリバリが加速され、コストも削減できます。

セクション 1 :

継続的テストの障害

ソフトウェア開発は、最も遅いプロセスに合わせて進行する傾向があるため、テスト・チームは開発パイプラインの遅延の責任を問われることがよくあります。一般的に、テストは品質を達成するために必要ではあるものの、プロジェクト遅延の原因になると考えられています。実際には、テスト・チームが原因で遅延することはほとんどありません。従来のテスト方法では手作業が多すぎるため時間がかかり、継続性を確保できません。手作業によるテストでは、欠陥が見過ごされるため、コストのかかる再作業が発生します。企業にとって、継続的テストが困難な主な理由は以下のとおりです。

不十分な品質要件と不完全なユーザ・ストーリー

欠陥の多くはコードを作成する前に発生します。欠陥の根本原因は不十分な品質要件にあり、ライフサイクル全体を通じて残ります。こうした欠陥が検出されずに残る期間が長ければ長いほど、システムの品質への悪影響は大きくなり、再作業にかかる時間も長くなります。

要件は通常、自然言語で記述され、異種のファイル形式で保存されますが、新しいアジャイル・プロジェクト管理用ツールと従来のアプリケーション・ライフサイクル管理ツールを使用しても、それらの依存性が関連付けられることはほとんどありません。これらのツールによって要件とユーザ・ストーリーの一元的なリポジトリは提供されますが、それによって要件収集の方法が改善されるわけではありません。ユーザはいまだにテキストを空いているフィールドに入力しています。異なるのは、これらのストーリーと変更要求が、ウォーターフォール・プロジェクトの最初の段階で単一の文書にまとめられないことです。

時間のかかるテストケースの設計

記述された要件とフラットなダイアグラムは静的であるため、そこからテストケースを手作業で作成する必要があります。これは非常に時間がかかり、継続的テストは不可能で、十分な品質のテストにつながることはほとんどありません。現在の一般的な開発システムは、ロジックを通じたパスが数百万から数十億あり、そのそれぞれが潜在的なテストケースです。直線的な手法を使用した手作業によるテスト作成にかかる時間は、予測も測定も困難です。CA Technologies が協力したテスト・チームの中には、1 つのコンポーネントに 11 のテストケースを作成するだけで 6 時間かかった例がありますが、それでも網羅率はわずか 16% でした。[1]

手作業で作成したテストケースには、通常、テスト担当者が思いついたすべてのテストが網羅されています。不完全な文書と複雑なアプリケーションに直面すれば、どんなに優秀なテスト担当者でも、数百万という潜在的なパスのほんの一部を網羅するにも苦労するでしょう。そのためテストは通常、「成功するパス」が中心となり、特定の機能が繰り返しテストされる一方で、アプリケーションに障害が発生する可能性のあるシナリオは放置されます（あるいは単に見逃されます）。CA が行った監査の経験では、10 ～ 20% の機能しか網羅されず、特定の機能を 4 倍も過剰にテストするような状況が一般的に起きています。特にあるチームでは、3 つのテストケースを作成しても、テスト対象システムの 5% が網羅されただけでした。これでは、テストの本来の目的である保証を提供するには十分ではありません。

手作業によるテスト・データの作成

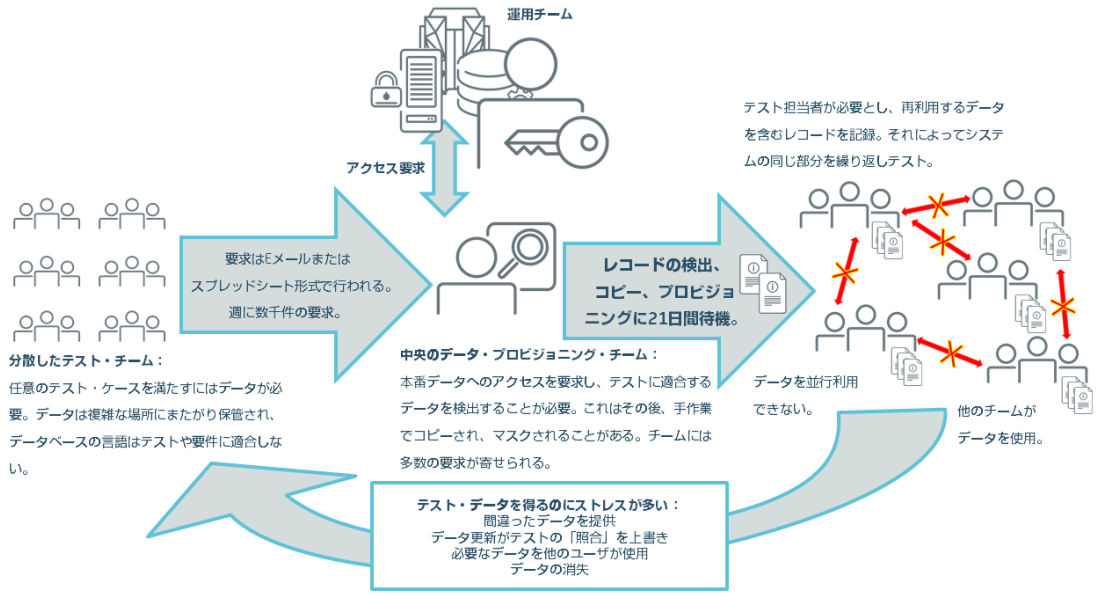
テストケースが定義されると、テスト担当者はそれを実行するためのデータが必要です。ここでも、第一の課題は品質です。多くの組織にとって、テスト・データとは任意に抽出され、マスキングされた本番デ - タのコピーを意味しますが、これには十分なテストを必要とする可能なシナリオのほんの一部しか含まれません。このデータは通常、本番で発生した過去のシナリオからサンプリングされているため、常に古く、サニタイズされて不適

切なデータや期待されない結果は省かれています。テストが必要な機能をすべて網羅するテストがあっても、それを実行するために必要なデータがない可能性が高いのです。

テストデータの準備にかかる時間も大きな問題です。図 1 に示すように、多くのデータ･プロビジョニングは、チーム間の依存関係と時間のかかる手作業につながる非体系的な方法で実行されています。手作業によるデータ・プロビジョニングでは、可視化、レポーティング、追跡もほとんど不可能です。

図 1

通常のデータ・プロビジョニング・プロセス



このアプローチでは、テスト担当者はデータ要求を行う際、E メールを送信するか、基本的なスプレッドシートの要求フォームに入力します。データの検出、コピー、テスト環境へのプロビジョニングを担当する中央のチームが、こうした要求を処理します。このチームは潜在的に機密性が高い本番データへのアクセスを制御する別のチームに依存することが多いため、さらなる依存性が作られ、遅延も起きやすくなります。

利用不可能なシステム・リソース

テスト担当者にテストケースとそれを実行するデータが揃っていると想定した場合、必要なのはそれを実行するための本番同様のシステムです。しかし、限られた数の共有環境を使用している場合が多く、これは設定に数週間もかかり、コストの高い本番前インフラストラクチャを必要とします。

さらに、テスト担当者は複合アプリケーション内の未完成のバックエンドまたはサードパーティのコンポーネントへのアクセスを必要とすることがよくあります。これらのコンポーネントは別のチームが使用中の可能性があり、テスト担当者はそれが利用可能になるまで待機しなければなりません。多くの組織は当然のようにサービス仮想化を使用して、利用できないコンポーネントや制約のあるコンポーネントをシミュレートしますが、これには実用的で一貫して定義された仮想データが必要です。複雑な要求 / 応答のペア・データを手作業で定義すると、大幅に遅延する可能性があります。

テスト自動化の不足

すべてのテストを手作業で実行するのは時間がかかりすぎ、たとえば、再帰テストを短い周期で反復する場合などの選択肢になりえません。テストを実際に実行するには自動化が必要です。それによって、開発プロセスでテストの手順をシームレスに実行し、頻繁に繰り返すことができます。現在、開発者が一度に開発するのはコードの一部のであるため、その後のコード変更を検証するために、テスト・プロセスは常に利用できなければなりません。

さらに、テストは API（アプリケーション・プログラム・インタフェース）を含む UI 層を超えて移動する必要があり、よりデータ中心になっています。そのため、自動テストはアプリケーション・ライフサイクルのより早期に開始することが必要で、開始前に UI が用意されるまで待つことはできません。

「GUI テストは時間がかかり、変化するユーザ・インタフェースが原因で保守が困難です。代わりに階層的なアプローチを重視することで、開発者とテスト担当者は GUI テストの使用を減らしています。」

—ガートナー社、「階層的なテストを使用してデジタル・ビジネス機能の継続的デリバリを実現」[2]

一時的な性能テスト

現在、速度の遅いアプリケーションは受け入れられません。ページのロードに 3 秒以上かかる場合、顧客の 57% がカートを放棄すると回答しています[3]。そのため、開発初日からピーク時の性能低下を最小限またはゼロに抑えることを視野に入れてアプリケーションを設計し、構築する必要があります。

レガシーの性能ツールやテスト技術を使用していると、早期に厳格な性能テストを実行できません。時代遅れのテストツールでは、ピーク時の性能を現実的にテストするのに必要な数百万のユーザへの規模拡大は不可能です。また、実際のユーザの動作を反映できるほど洗練されてもいません。結果として、エクスペリエンス重視のユーザに対して、アプリケーションとそのサードパーティの統合によって十分な速度が実現すると保証できるほど、性能テストは厳密でも現実的でもありません。

変更への対応が不可能

継続的テストでは、コードが変更されたときにすばやく対応できることが重要です。ただし、現在では変更要求が正式に取得されることはほとんどありません。その代わりに、断片的なユーザ・ストーリーが付属した E メールの要求で構成されています。変更要求が届くとまず開発者とテスト担当者は、相互に依存した大量のシステム・コンポーネントにどのような影響があるかを特定する必要があります。これは簡単な作業ではありません。

これまでは、1 つ変更があると、2 人のテスト担当者が 2 日間かけてテストケースの確認と更新を行っていました。

—大手金融機関[1]

また、変更を実装するために必要な新規のテストを作成する前に、既存のテストと開発成果物を更新するための時間も必要です。テスト・アセットは要件から手作業で抽出され、要件とは異なる形式で保存されるため、要件までさかのぼってアセットを追跡することはできません。そのため、手作業で更新する必要があります。場合によっては、すべてのテストを手作業で確認し更新することもあります。あるいは、すべての既存のテストを廃棄して新しく作り直すこともあります。

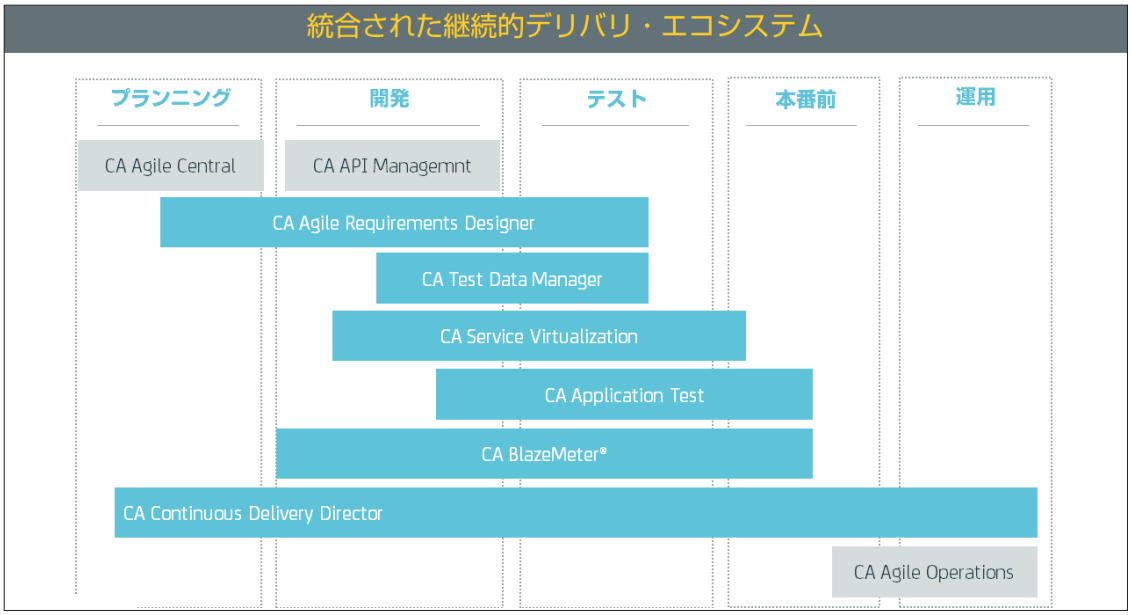
セクション 2 :

エンドツーエンドの継続的テストのエコシステム

ソフトウェア・デリバリに完全な継続性を確保するには、全体的なテスト手法が必要です。設計、計画からリリース、デプロイまで、異種混合のツールやプロセスを連携させる必要があります。これらがあれば、異なるフォーマットやツール間を移動する無駄な労力を排除できます。また、ライフサイクル全体での機能を超えたコラボレーションと可視化の不足による欠陥も防止できます。CA Technologies は、統合された継続的デリバリ・エコシステムを実現するために必要なツールを開発しています。図 2 は、テスト・エコシステムに対応する製品を示しています。次に、継続的テストに必要な手順と製品を簡単に説明します。

図 2

CA Technologies によるエンドツーエンドのデリバリ・パイプラインにおける並行した段階とテクノロジ



継続的なテスト・エコシステムには、以下が含まれている必要があります。

- 積極的な要件収集。必要なユーザ機能を正確かつ反復的に取得します。
- 最適化および自動化されたテストケースの設計。必要なユーザ機能の有効性を確認するために最小限のテストケースを作成します。
- セルフサービスのデータ・プロビジョニング。すべてのテストケースを実行するのに必要なデータを検出、保護または作成します。
- サービス仮想化。包括的で正確な仮想データを使用して、すべてのテストに対応した本番同様の低コストのシステムを構築します。
- 自動化された保守と追跡可能性。設計の変更時にユーザ・ストーリー、テストケース、データ、仮想エンドポイントをすべて同時に更新できます。
- テストの正確な自動実行機能。すべてのシステム、API、サービスの管理が簡略化されます。
- 完全なオーケストレーションとリリース自動化。すべての可動部分を整合させて、シームレスにコードをデプロイ、テスト、リリースできます。
- 最新のアセットとタスクの一元的なリポジトリ。部門の枠を超えたコラボレーションとプロジェクト全体の可視性が実現します。
- 実際のユーザの動作を厳密に再現した性能テスト。運用で起こりうるストレス・レベルを超えて規模拡大でき、誰でも開発のどの段階でも実行できることが重要です。

手順 1 : ユーザ・ストーリーの収集

CA Agile Central では機能を実現するために必要なテストおよび開発アセットの一元的なリポジトリが提供されるため、ビジネス・アナリストは新しいユーザ・ストーリーと変更を迅速に記録できます。一元的なダッシュボード、ロードマップ、バックログによって、相互に依存するストーリーとそれに関連するタスクのステータスを容易に視覚的に追跡できます。また、プロジェクト全体の可視性が得られるため、部門の枠を超えたチームがより緊密にコミュニケーションとコラボレーションを行えます。

手順 2 : テスト・モデルの構築

CA Agile Requirements Designer で作成されるフローチャート・モデルは、CA Agile Central のユーザ・ストーリーにエクスポートでき、ユーザ機能の明確で完全な仕様が開発者に提供されます。たとえば、製品オーナが CA Agile Central でユーザ・ストーリーを作成します（手順 1）。そのユーザ・ストーリーのテストを行う担当者（モデル化担当者など）は、それを CA Agile Requirements Designer に移動し、モデル化して詳細（複数のフロー、他の否定的なシナリオ、受け入れの基準、実行済みの定義など）を追加できます。最後に、完成したユーザ・ストーリーのテキスト・バージョンを CA Agile Central にエクスポートして、製品バックログにベースラインにします。

「CA Agile Requirements Designer では、要件がフローチャートでグラフィカルに表示されます。フローチャート全体の明確なパスによって、システムのロジックを簡単に読み取り、理解できます」。

—テスト・データ・エンジニア（a.s.r.）[4]

最適化されたテストケース設計

フローチャートはシステム・ロジックの正式な描写であるため、テストケースをフローチャートから生成できます。CA Agile Requirements Designer では、テストケースに必要なフローチャートのすべてのパスを特定するための自動アルゴリズムが提供されます。利用可能なすべての情報に基づき完全なシステム・モデルを構築すると、これらのテストによって起こり得るすべての否定的なパス、異常値、予測できない結果も網羅できます。また、手作業のテスト設計に時間を費やさなくても、テストの正確性を確保できます。

「1 週間程度ですべてのプロセスを文書化して、関連するテストケースを生成し、すべてを 100% テストすることができました。すべてのテストケースをわずか 3 日の追加作業だけで実行できました。」

—テスト・データ・エンジニア（a.s.r.）[4]

手順 3 : テスト / データの検出と構築

CA Agile Requirements Designer と CA Agile Central は CA Test Data Manager を統合すると、テストの作成と管理に包括的なエンドツーエンドのテスト・データ管理機能が提供されます。CA Agile Requirements Designer ではテスト・マッチを設定し、最適化されたテストケースを実行するのに必要なデータを検出できます。データがない場合は、合成データ生成機能を使用して、不足するデータを特定して作成できます。これは CA Test Data Manager で作成された正確なデータ・プロファイルに基づいて実行するか、CA Agile Requirements Designer でフローチャート・レベルで定義されたデータ変数を使用して実行することが可能です。本番データを使用する必要がある場合は、CA Test Data Manager の複数のネイティブ・マスキング・エンジンを使用して、機密性の高い情報を迅速に削除できます。

データはテスト・データ・ウェアハウスに再利用可能なアセットとして、自動検出に必要なクエリとともに保管されます。そこからデータをプッシュアウトして CA Agile Central のテストケースに添付でき、テスト担当者はセルフサービス Web ポータルを使用してオンデマンドでデータを要求できます。

「CA Test Data Manager ソリューションによってテスト・データの取得にかかる時間が 21 日間から 2 日間へと大幅に削減され、ソフトウェア・デリバリ・サイクル全体の効率性が向上しました。」

— 金融サービス企業、品質保証および戦略ディレクター [5]

手順 4 : テストの制約の排除

アプリケーション・アーキテクチャが複雑になり、組織が世界中に分散化した結果、開発チームとテスト・チームはデリバリに向けて取り組む中で制約に直面しています。すべての相互依存性を考慮すると、これらのチームは多くの人手を要する作業に取り組まなければなりません。チームは開発またはテスト・ラボの完全な環境のコピーを生成するか、依存先のシステムまたは環境の動作をレプリケートするために、独自のコードを構築する必要があります。適切な環境へのアクセスに伴う多額のコストと時間の制約が、サービス仮想化の増加につながりました。

サービス仮想化は連携先システムの動作、データ、性能の特徴のシミュレーション・プラクティスで、開発とテストのすべてのステージで使用できます。スタブやモックとは異なり、仮想サービスは契約による実際のサービスの記録、または、サンプル要求 / 応答によって構築されます。仮想サービスはより動的で、開発者はこれを記述する必要がありません。サービス仮想化を使用すれば、必要なときにいつでも、実質的に制約のないテストケースやシナリオのセットをテストできます。また、独立したすべてのサービスとエンドポイントを、他と無関係な状況ではなく、連動させて検証するようテストを設定できます。サービス仮想化によってコールの限界から解放されるため、アプリケーションが完全にデバッグされるまで何度でも必要なだけテストを続けられます。

SunTrust 銀行は管理サービスにかかるコストを 22%削減しました。これには自動化による 197 万ドルの削減と、90 万ドルの直接的な削減が含まれています。[6]

手順 5 : テストの自動化

この時点までに、求められるユーザ機能が使用され、テストおよび開発アセットが定義されます。それらを実行するために必要なテスト・データや仮想サービスと同時に、最適化されたテストケースが作成され、本格的な継続的テストを妨げるボトルネックが解消されます。さらに、アセットは一か所、つまりフローチャートに保管され、テストを関連データや仮想サービスに適合させます。

これは、ユーザから始まり要件収集担当者、QA、開発、テスト・チーム、そして運用チームまでも関与するコラボレーションのプロセスです。これらのチームが同じ画面から効果的に作業するために必要なツールを装備することは重要です。その後、このコラボレーティションを整合させ、実行させる必要があります。ここで、標準化されたオーケストレーションと、すべての層にまたがるテストの自動実行が登場します。

テスト実行の自動化

CA Agile Central に保管され最適化されたテストは、変換プロセスを加速する視覚的なワークフローと使いやすい Web エディタを使用して、CA Application Test の自動テスト製品群に変換されます。テストは可能な限り最小数に最適化されているため、自動化エンジニアに求められる手作業も大幅に削減されます。自動化エンジニアは最新のユーザ・ニーズをテストするためにはどのテストを自動化する必要があるか正確に把握しており、反復的で直線的なスクリプト定義やキーワードの選択に費やされる時間を回避できます。

「すでに 10,000 以上のテストケースを作成し、手作業から自動実行へ切り替えました。当社の QA チームはこれで複雑な手作業のテストケースにより多くの労力をかけられるようになりました。

— Real User Review（ IT Central Station）[7]

手順 6 : アプリケーションのロード・テスト

デプロイされた QA 環境での継続的テストは、機能検証だけで終わりません。アプリケーションの性能がエクスペリエンス重視のユーザを満足させるのに十分高いレベルであることも、検証する必要があります。そこで CA BlazeMeter® が役立ちます。これを使用すると、性能テストを作成し、CA Agile Central のユーザ・ストーリーとテストケースに添付できます。アプリケーションの動作を説明する CA Agile Requirements Designer で作成されたモデルは、性能テストの適用範囲に従って、性能テストのシナリオ / ビジネス機能や自動スクリプト（JMeter、Gatling、Taurus など）を抽出するために使用できます。CA BlazeMeter のテストは CA Release Automation を使用して、デプロイの一部としてトリガできます。

CA BlazeMeter ではセルフサービスの SaaS ツールが提供され、ユーザは開発ライフサイクルのどの段階でも性能テストを実行できます。モバイルおよび Web アプリケーションのテストを実行するにはインストールは必要なく、設定も最小限です。また、クラウドおよびグローバル・データセンタ上の仮想ユーザを使用してユーザ性能テストとロード・テストを実行できます。これによってコストが高すぎるインフラストラクチャの要件を回避でき、アプリケーションを運用で可能な範囲以上に重視できます。

正確なストレス・テストを容易に数百万に規模拡大でき、実際のモバイルおよび Web トラフィックをロードしてユーザ動作を再現できます。ユーザ・インタラクションが Selenium テストにすでに定義されている場合、これらを Apache JMeter などのオープンソース・テストと組み合わせて活用でき、性能テストがユーザ動作を正確に再現するために役立ちます。

手順 7 : アプリケーションのデプロイ

CA Continuous Delivery Director を使用すると、異種混合のアプリケーションのテストは、最新のアセットを並べて再生ボタンを押すのと同じくらい簡単に行えます。CA Application Test、CA Test Data Manager、CA BlazeMeter、CA Service Virtualization で行ったコラボレーションを完全に活用できます。自動の機能テストと性能テストを実行する前に、仮想サービスを活用し、豊富なテスト・データをプラグインできます。

中核となる自動エンジンによって再利用される前に、必要とされる正確なリリース・プロセスを定義できます。CA Continuous Delivery Director はマルチチーム、マルチアプリケーションのリリース向けに調整され、依存関係を特定し、解決するのに必要なツールが提供されます。つまりサーバやデータセンタ、環境がどれだけ多様であっても、最新のビルドを迅速に一貫してデプロイできます。また、テスト・チームと開発チームは、継続的デリバリ・パイプライン内で継続的テストを統合でき、必要なユーザ機能を反映した革新的なソフトウェアのデリバリに集中できます。

セクション 3 :

まとめ

ユーザのアプリケーションに対するニーズは急速に変化しており、ユーザの期待はかつてないほど高くなっています。企業は常に変化するユーザ･ニーズをソフトウェアに正確に反映し、継続的にデリバリする必要があります。ビジネス・チームと開発チームは必要なユーザ機能をソフトウェアとテスト・アセットに反映させる前に、反復的に正確に取得できなければなりません。

さらに、これらのソフトウェア・コンポーネントはすべて、ユーザ・ストーリーからテストケース、テスト・データ、仮想サービス、およびデプロイされたシステムまで、整合性を維持する必要があります。継続的デリバリの目標達成するための最善の方法は、要件そのものから直接アセットを動かすことです。これは、アイデアを設計から運用まで品質を低下させずに迅速に進められる、エンドツーエンドの継続的デリバリ・エコシステムに統合された継続的テストによって可能になります。

セクション 4 :

参考資料

1 Huw Price、「Test Case Calamity」、2016 年 2 月 24 日

2 ガートナー社、「階層的なテストを使用してデジタル・ビジネス機能の継続的デリバリを実現」、2016 年 11 月

3 Voucher Cloud、「Consumer Psychology and the E-Commerce Checkout」、2015 年 7 月

4 CA Technologies、「CA Agile Requirements Designer at a.s.r.」、2016 年

5 Forrester Research、「The Total Economic Impact ™ of the CA Technologies Test Data Manager Solution」、2015 年 12 月

6 CA Technologies および UBM、「Enterprise IT leaders share their personal continuous delivery experiences」、2016 年 10 月

7 IT Central Station、「CA Service Virtualization Review: I Like the Support to Build Modular Test Cases Using Reusable Blocks」、2016 年 2 月

商品・サービスに関するお問い合わせ・ご相談受付

【電話による受付】

0120-421-126 [通話料無料]

受付時間 10:00～17:30　月曜日～金曜日（祝日、当社休業日を除く）

【メールによる受付】

webmaster@hitachi-solutions.com

※ご相談、ご依頼いただいた内容は、回答などのため、当社のグループ会社に情報を提供し対応させていただくことがあります。取り扱いには十分注意し、お客さまの許可なく他の目的に使用することはありません。

株式会社 日立ソリューションズ

https://www.hitachi-solutions.co.jp/

Copyright © 2017 CA. All rights reserved. 本書に記載されているすべての商標、商号、サービス・マーク、ロゴは、該当する各社に帰属しています。本書は情報提供のみを目的としています。本書に含まれる情報の正確性または完全性について CA は一切の責任を負いません。準拠法で認められる限り、本書は CA が「現状有姿のまま」提供するものであり、いかなる種類の保証（市場性または特定の目的に対する適合性、他者の権利に対する不侵害についての黙示の保証が含まれますが、これに限定されません）も伴いません。CA は、この文書の使用によって直接的または間接的に生じた損害について、たとえ CA がかかる損害の可能性について明確な通知を受けた場合でも、一切責任を負いません。これには、利益の損失、事業の中断、営業権、データの損失が含まれますが、これに限定されるものではありません。

CS200-284691_0417