WHITE PAPER | 2014 年 11 月

サービス仮想化によるソフトウェア開発でのテストの制約を排除

ca technologies

目次

概要

課題

アプリケーションが複雑になり組織が世界中に分散化した結果、チームはデリバリに向けた取り組みの中で次々に発生するボトルネックに直面しています。ボトルネック（つまり制約）の例としては、メインフレームまたは ERP 環境へのアクセス、テスト・データの欠如、サードパーティ・システムへのアクセス、予算の制限などがあります。多くの場合、制約は、並行して作業している複数の開発チームが同じ環境にアクセスしようとすることが原因となって発生します。チームはこれに対処するために、たいていテスト・ラボに環境全体をコピーするか、テスト用の独自バージョンをコーディングして対応システムを「モックアップ」しようと試みます。これには多大なコストと時間が掛かる可能性があります。

ビジネス・チャンス

基本的に、サービス仮想化（Service Virtualization）とは、開発およびテスト環境を「モックアップし、スタブ化」する手順を製品化したもので、開発を加速する十分な現実性とコンテキストを持っているだけでなく、開発工程の中で**テスト工程を短縮（左シフト）**し、統合およびリリース・プロセスを前倒しして、高品質と低リスクを実現します。

メリット

これにより、サービス仮想化ソリューションは組織に多数の機能をもたらし、拡大したチームが、高品質のアプリケーションを短期間に低コスト / リスクで市場に送り出すことができます。これには、以下が含まれます。

開発で「実際に近い」環境を利用できる	高品質の開発とより効果的なリグレッション / システム・テスト
並行開発およびテストの実現	開発工程時間の短縮、早期不具合検出、リソースの効率的な使用
範囲外システムに関するテスト・データの仮想化	素早いセットアップ / テスト自動化の安定性
高パフォーマンス環境の実現	パフォーマンス・テストをより実際に近づけ大幅なコスト削減の上での品質向上

どのようなアプリケーション開発やテスト環境もそれだけ単独で成り立つわけではないため、サービス仮想化ソリューションは、チームに備わっているそれぞれのツールに対応できる、「ベンダに依存しない」基盤を提供することが重要です。サービス仮想化は、テスト管理（TM）、不具合管理 / 問題追跡と主要ハードウェア、および環境内に存在するテスト・ラボ仮想化製品など、既存のアプリケーション・ライフサイクル・ソリューションと合わせて機能するターゲット環境を提供する必要があります。

セクション 1:「実際に近い」環境の生成

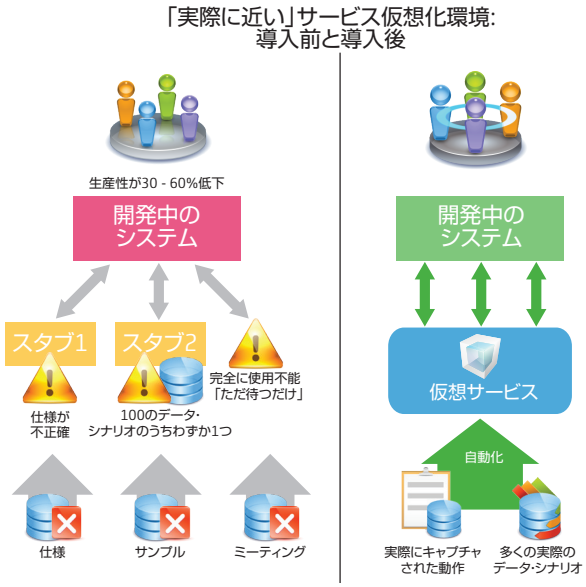
開発およびテスト環境で依存システムをシミュレート

アプリケーション開発は、より複雑でサービス指向のアーキテクチャを重視する傾向にあり、開発環境やテスト環境において、さらに幅広いアップストリームおよびダウンストリーム・システムをシミュレートする必要がでてきました。サービス仮想化を依存性が強いレイヤ間に適用すると、本番もしくは「実際に近い」環境を提供できます。

従来の手法では、ダウンストリーム・システムのみを「スタブ化」して、それぞれのコンポーネント開発を進めようとしていました。

たとえば、Web UI を開発している場合は、直下のレイヤ（すなわち、Web サービス）から想定される応答にスタブを構築します。次に、Web サービスの開発者が、基本 ESB レイヤをスタブ化、もしくは Web UI からのユーザ要求の一部をモックアップしようとします。残念ながら、このような手動プロセスは、複雑なソフトウェアのアーキテクチャ内に存在するさまざまなタイプの接続やデータを十分にカプセル化することができず、以下に示したように、UI がコーディングされていなければ利用不能となる場合もあります。

図 A



また、チームが仮想サービスとしてキャプチャされる実際のデータ・シナリオや動作で作業を進めている場合は、手動でコーディングし、維持しなければならないスタブ・セットよりも、得られる環境がずっと現実的で最新のものとなります。

したがって、「実際に近い」環境を実現する重要な技法とは、仮想サービス作成およびデータ維持の自動化となります。ユーザ・インタフェースがまだ完成していなくても、開発チームは、実際に近い仮想環境により、生産性が大きく向上するだけでなく、スタブをメンテナンスする作業時間が減ります。

期待されるメリット

- インタフェース・システムが利用できなくても、開発を開始できる
- テスト工程の時間を短縮する
- 他のアプリケーションへの依存度の低減および利用可能なテスト時間が増えることにより、テストのカバレッジを改善できる
- 少ない作業でユニット・テストを向上する
- テストのカバレッジおよびリグレッション・テストの拡大により、品質を向上する
- 少ないメンテナンスで素早くシミュレータを構築

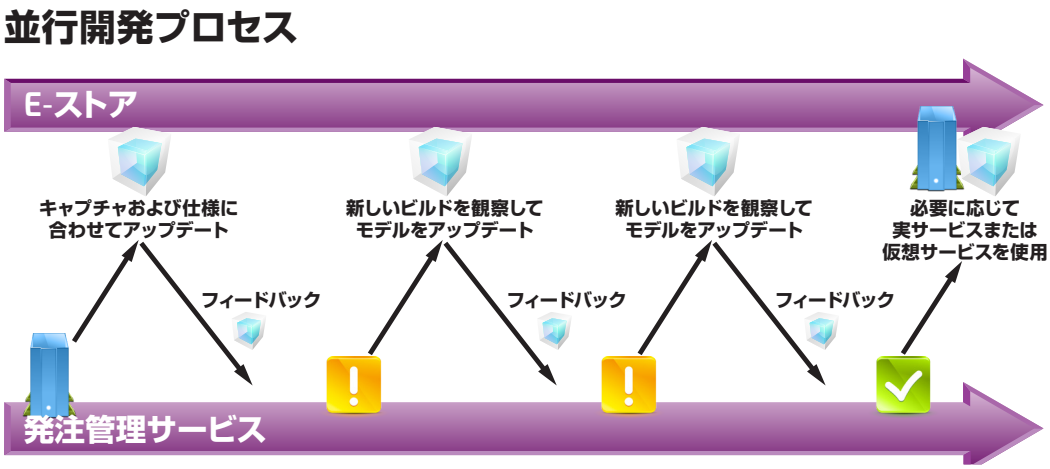
セクション 2: 並行開発およびテストの実現

開発およびテストを並行に実施

サービス仮想化ソリューションの 2 番目の重要な機能に、並行開発および並行テストを実現できることがあります。開発およびテスト・チームが同時に作業を行う場合、ソフトウェア・ライフサイクル全体が効果的で効率的な新しい域に達します。新しいソリューションを、優れた価値で組織に提供できます。

並行して開発とテストを行う場合、仮想サービスは、開発対象システムとテスト対象システムの間に存在することができます。そのため、以下の例では、ひとつのチームが発注管理サービス（OMS）を開発しており、上側のチームは「e- ストア」アプリケーションを開発およびテストしています。

図 B



仮想サービスは、e - ストアのテスト作業の初期バックエンドとして、既存の OMS システムからキャプチャされます。テストを継続することで、e- ストア・チームは、予期しない、または新しい応答要件を（基本的に開発の次の要件セットとなる）「フィードバック」仮想サービス要求として送り返すことができます。新しいビルドで仮想サービス・モデルのアップデートが繰り返されるたびに、各並行開発およびテスト・サイクルは加速され、フィードバックもさらに速くなります。

完璧な並行開発ソリューションでは、チームが、実際に利用可能で、堅牢な機能を持ち、データが同期化された本物のサービスに対して実行できるようになります。コンポーネントを正しくサポートするサービスがまだ完成していない場合、チームは、仮想サービスにすぐに切り替えることができます。仮想サービスによる仮想ダウンストリーム・システムや実システムの切り替えまたは仲介が可能であることは、新しいビルドが発生したとき、または新しいデータ・シナリオが必要なときに、いつでも元へ戻せることを意味し、堅牢な並行開発機能を作成するうえで非常にパワフルな資産となります。

ソフトウェアのテスト環境を作成および更新する方法は現実的だと思いますか。基本的に、サービス仮想化の並行性は、開発およびテスト・サイクルが短く、多くのリリースが必要となる現在のビジネス目標に近づくことで、アジャイルを実現します。

期待されるメリット

- ますます加速するテストおよび開発のライフサイクル
- アジャイル反復の正しい応答と、テスト結果およびビジネス要件に基づいた継続的インテグレーションとビルド
- バージョン管理の負担を低減し、既存の ALM および開発管理ツールと連動してその有効性を向上
- 本番前に、障害の受け入れ解決率を向上
- 最大で 60% の機能ポイント高速化と、高品質および正確な仕様

セクション 3: スコープ外の依存に関するテスト・データの取り扱い

ダウンストリーム・データの損失はもう心配いらない

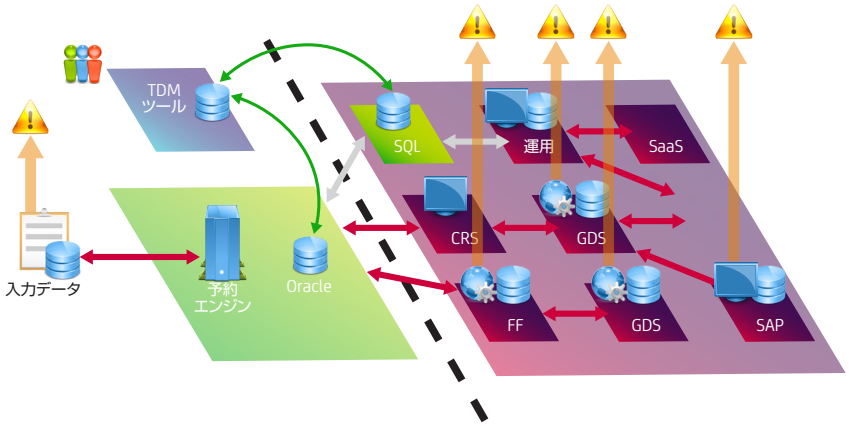
テストを実施する際には管轄内のシステムとそれ以外（管轄外）のシステムがあり、それに、関連するデータがあります。管轄内システムとは、システム更改またはテストを直接実行しているシステムです。範囲外システムとは、範囲内システムのサポートとして必要であるが、開発またはテスト作業の対象ではないシステムです。これを依存性といいます。依存性は必要ですが、開発またはテスト作業の対象ではありません。

このシナリオでは、チームが予約エンジンを開発およびテストしており、必要なダウンストリームおよびユーザ入力データ・シナリオを収集しようとしているが、そのカギを見出していません。

最も明確なソリューションは、テスト・データ管理の通常手順により、コードを作成し、データ・ラインをインポートして、管轄外依存性の予期される応答を示すことで、管轄内システムから直接データをインポートし、それを管轄外システムに「スタブ化」またはモッキングすることです。このようなスタブは本質的に不安定であり、開発者は、基本機能または予期される応答シナリオをシミュレートするという、最も都合のよい手順を取ります。しかし、分散ソフトウェアの複雑度が拡大する今、手動でコーディングし、スタブをメンテナンスする作業はコストが掛かりすぎます。

図 C

スコープ外のデータはテスト・データ管理でカバーされない



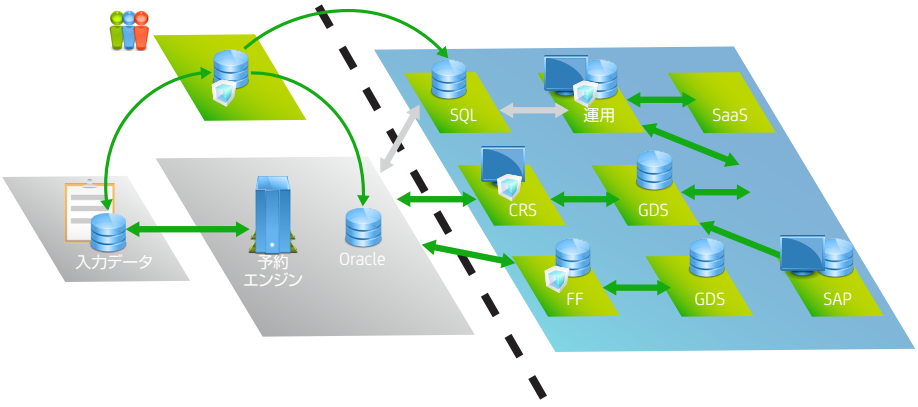
本物か、仮想データか

必要なのは、管轄外システムの動作をシミュレートし、実際には本番のシステムではないが本番のシステムと連携していると、管轄内システムが認識できるようにする方法です。

サービス仮想化は、該当する管轄外のダウンストリーム・シナリオのキャプチャを自動化することで、管轄内シ ステムの背後で欠落しているデータが問題にならない手法を取っています。

図 D

スコープ外のテスト・データの仮想化



ここでは仮想化されたテスト・データ管理の最終例をかなり単純化して示していますが、テスト・プロセスと外部または管轄外依存性との対話は実際にこのように進められています。仮想モデルにより、チームは常に、テスト対象システムの該当するデータセットにオンデマンド・アクセスが可能で、このデータは、大量のパフォーマンスおよびリグレッション・テストのニーズをサポートするほぼ無限のデータ・シナリオに対応します。

サービスの仮想化は、必要なシステムを開発およびテスト環境に盛り込むメカニズムが必要になります。これには、管轄外システムのデータのプロビジョニングや、経時的にワークフローのシステム間を通過する「ステートフル」ランザクション・データのメンテナンスなどがあります。たとえば、長期的に実行されているトランザクションで日付や合計などの値の変化をシミュレートし、このような複雑なワークフローを必要な詳細レベルで検証できるようにします。

期待されるメリット

- 管轄外システムからのアクセスまたは最新データがないことが原因による遅延を排除
- 複数のテストおよび開発チームが 365 日 24 時間有効なテスト・シナリオを利用可能
- システムでのデータの上書きまたは変更によってテスト・データの矛盾が生じたり、他のチームの作業が無効になることがない
- クリティカルな実働システムに影響がまったくないか、ほとんどない
- 複数の技術レイヤを通り、日付や顧客 ID などの特定の値を維持する必要がある複雑なプロセスをサポートする、「ステートフル」トランザクションのサポート
- データのセットアップまたはリセットに要する時間が最大で 90% 短縮、テスト・ライフサイクル時間全体を 40 ～ 60% 削減

セクション 4: 多様な技術およびプラットフォームをサポート

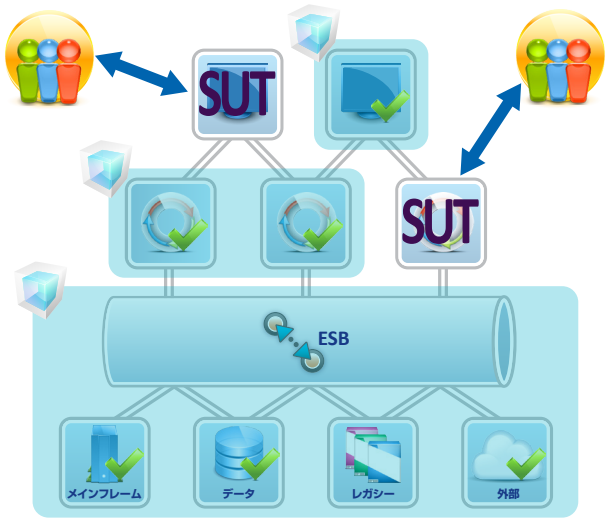
完全な環境の生成

最近、世界でも最大規模の銀行のアーキテクチャ・チームを訪問する機会がありましたが、このとき、驚くような数字を耳にしました。ある担当者によると、同行のハードウェア資産管理システムの分析では、従業員よりも多い数のサーバが導入されているということでした。どのプロジェクト・チームにきいても、それぞれの開発、テスト、本番前および本番ハードウェアの支出が正しいと主張するでしょう。また現在使用されているほぼすべてのアプリケーションには、少なくとも 4 種類の環境があり、多くの場合これらのアプリケーションの次のリリースが出るまで、何か月あるいは何年もかかることがあります。

皮肉なことに、大部分の既存の本番前環境における最大の課題は、システムが完全ではないということです。プロジェクト・チームは、いくつものサーバをキャプチャし、アクセス可能な VM 上のハードウェアまたはコンポーネントをレプリケートできます。しかし、それぞれハードウェア予算が割り当てられていても、アクセスが可能になるまで待ったり、共有システムへのアクセスが非効率なため、いまだに何か月も開発サイクルに費やしています。

異種混合システムは、エンタープライズ IT 環境では標準になっています。このため、サービス仮想化を適用して、テスト対象システム（SUT）に影響を及ぼすすべてまたは一部の依存性を仮想化する必要があります。これには、Web トラフィック（HTTP）、Web サービス（SOAP/XML）、統合レイヤと ESB（JMS など）に加えて、基本メインフレーム（CICS、CORBA など）、データベース（JDBC など）およびサードパーティ・サービスとのトランザクションおよび接続のシミュレーションが含まれます。

図 E



ソフトウェア・アーキテクチャのすべての接続ポイントは潜在的な変更ポイントであり、障害リスクとなり得るため、サービス仮想化は、すべてを仮想化」し、多様なコンポーネントが混合している依存性から解放される方法をチームに提供することが必須となります。

サービス仮想化を使用すると、多数の本番前のテストを管理が容易な 1 つのインフラストラクチャにまとめることができ、必要な環境でオンデマンドによるソフトウェア・ベースのプロビジョニングが可能になります。現在変更を行っていないプロジェクトは、電力を消費せず、放熱せず、フロア・スペースを使用しません。依存性がキャプチャされ、排除されます。メインフレーム・パーティションで実システムにアクセスしなくても、チームは、顧客情報管理システムの仮想サービスをオンデマンドでプロビジョニングできます。

期待されるメリット

- すべての開発およびテスト・チームに特化されたラボ環境に、毎日 24 時間アクセスできるため、デリバリが迅速化
- 従来の本番前インフラストラクチャのコスト削減 - 大規模エンタープライズでは 1 年で 2000 万ドル以上の削減
- リモート・システムへのアクセス・コストおよび料金の排除
- 従来のユーザ受け入れテスト作業よりもずっと前に、コンポーネント開発およびシステム統合レベルで、チームが「テスト工程を短縮（左シフト）」し、品質を盛り込めるようにします。データのセットアップまたはリセットに要する時間が最大で 90% 短縮、テスト・ライフサイクル時間全体が 40 ～ 60% 削減できます。

また、仮想化対象の資産だけではなく、チームがコラボレーションに使用するプロセス・レベルのツールにも適用されます。サービス仮想化ソリューションは、チームが使用するツールの選択肢に「ベンダに依存しない」基板を提供することが重要です。サービス仮想化は、テスト管理（TM）、不具合管理 / 問題追跡、テスト・データ管理、主要ハードウェアおよび環境内に存在するテスト・ラボ仮想化製品など、既存のアプリケーション・ライフサイクル・ソリューションと合わせて機能するターゲット環境を提供する必要があります。

セクション 5: まとめ

正しいファースト・ステップの実行

開発またはテスト担当者に、次のような質問をしたことはありますか。「予定通りにリリースされなかったのはなぜですか」遅れた理由として、別のチームが約束を守らなかったとか、その他の言い訳が次々と出てきます。関係があるもので、実際に体験した例を、いくつかご紹介しましょう。

- 「はい、しかし、新しい発注管理システムのビルドを先月には入手できるはずだったのですが、実際に手に入ったのは先週でした」
- 「はい、しかし、発注管理システムの新しいビルドはどれも、必要としていた新しい何らかの機能をもたらしたのですが、それが正しく動作していた従来の機能を壊したのです」
- 「はい、しかし、発注管理に他のチームから届いたスタブには 1 つの顧客プロファイルしか入っておらず、作業を完了するのに必要な他のシナリオを構築したり、テストすることができないのです」

最終的に、依存している作業で他のスタッフの「はい、しかし…」が終わらないと、自分たちの作業も終われないのです。複雑なアプリケーションに関与している各チームは、インフラストラクチャからオンデマンドで独自のラボを自由に構築できることが必要です。しかしそこには、避けられないチーム間の依存性があります。このため、サービス仮想化機能が非常に重要なのです。各チームは、ダウンストリーム依存性の最初のビルドを必要とする前にでも、ダウンストリーム依存性から自由に仕様を取り出し、想定されるダウンストリーム依存性の将来状態を構築できます。

サービス仮想化は、エンタープライズ・ソフトウェア開発およびテストに不可欠なプラットフォームを提供します。特に組織側では、SV 実現戦略を計画するときに、その他多数の要素を検討する必要があります。誰がソリューションに貢献するのか、誰が使用するのか。チームおよびパートナーの拡張組織内で、誰が仮想サービスを所有し、管理するのか。サービス仮想化への幅広い手法では、技術的詳細だけではなく、多様な採用および成功を保証する運用的要素も検討します。

ca.com/jp/でCA Technologiesにアクセスしてください

CA Technologies（NASDAQ:CA）は、企業の変革を推進するソフトウェアを作成し、アプリケーション・エコノミーにおいて企業がビジネス・チャンスを獲得できるよう支援します。ソフトウェアはあらゆる業界であらゆるビジネスの中核を担っています。プランニングから開発、管理、セキュリティまで、CA は世界中の企業と協力し、モバイル、プライベート・クラウドやパブリック・クラウド、分散環境、メインフレーム環境にわたって、人々の生活やビジネス、コミュニケーションの方法に変化をもたらしています。詳細については **ca.com/jp** をご覧ください。

CS200-94676_1114