

(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7316461号
(P7316461)

(45)発行日 令和5年7月27日(2023.7.27)

(24)登録日 令和5年7月19日(2023.7.19)

(51)国際特許分類

F I

G 0 6 F 8/72 (2018.01)

G 0 6 F 8/72

G 0 6 F 8/60 (2018.01)

G 0 6 F 8/60

G 0 6 F 11/36 (2006.01)

G 0 6 F 11/36 1 8 4

G 0 6 F 11/36 1 8 8

請求項の数 20 (全26頁)

(21)出願番号 特願2022-541221(P2022-541221)

(86)(22)出願日 令和3年1月28日(2021.1.28)

(65)公表番号 特表2023-507229(P2023-507229
A)

(43)公表日 令和5年2月21日(2023.2.21)

(86)国際出願番号 PCT/IB2021/050667

(87)国際公開番号 WO2021/171110

(87)国際公開日 令和3年9月2日(2021.9.2)

審査請求日 令和4年8月26日(2022.8.26)

(31)優先権主張番号 202021008655

(32)優先日 令和2年2月28日(2020.2.28)

(33)優先権主張国・地域又は機関
インド(IN)

早期審査対象出願

(73)特許権者 522265729

ヘキサウェア テクノロジーズ リミテッド
HEX AWARE TECHNOLOG
IES LIMITEDインド国, 4 0 0 7 1 0 ナビ ムンバ
イ, マハブ, ティーティーシー インダ
ストリアル エリア, ‘エー’ ブロック,
ミレニアム ビジネス パーク, セクター
-III, 152152, Sector-III, Mi
llenium Business Pa
rk, ‘A’ Block, TTC In
dustrial Area, Maha
pe Navi Mumbai 400 7
10 (IN)

最終頁に続く

(54)【発明の名称】 アプリケーション環境マイグレーションのためのシステムおよび方法

(57)【特許請求の範囲】

【請求項1】

アプリケーション環境マイグレーションのための方法であって、前記方法は、

アプリケーションサーバ(102)のプロセッサ(122)によって、ソースアプリケ
ーション環境のソースアプリケーションコードを評価することと、前記プロセッサ(122)によって、前記ソースアプリケーションコードをターゲット
アプリケーションコードにマイグレーションするための変化の量を確認することと、前記プロセッサ(122)によって、評価統計(302)を予測することであって、前
記評価統計(302)は、前記ソースアプリケーションコードの前記マイグレーションを
完了するために、少なくとも1つの機能的準備(304)およびタイムライン(306)
を提供することと、前記プロセッサ(122)によって、ビジネスロジックを特定するために前記ソースア
プリケーションコードをスキャンすることと、前記プロセッサ(122)によって、前記ソースアプリケーションコードのためのリフ
ァクタリングされたコードを生成することであって、前記リファクタリングされたコード
は、前記ビジネスロジックを保持しながら前記ターゲットアプリケーションコードに従っ
て、前記ソースアプリケーションコードをマクロサービス(426a、…、426n)に
分割し、かつ前記マクロサービス(426a、…、426n)をリパッケージすること
によって生成されることと、

前記プロセッサ(122)によって、予測された前記評価統計(302)および前記リ

ファクタリングされたコードに従って前記ソースアプリケーション環境のコンポーネントを更新することと、

前記プロセッサ（１２２）によって、前記ソースアプリケーション環境をターゲットアプリケーション環境にマイグレーションすることであって、前記マイグレーションは、前記ソースアプリケーション環境の更新された前記コンポーネントおよび前記リファクタリングされたコードを前記ターゲットアプリケーション環境にリプラットフォームすることとを、含む、方法。

【請求項２】

前記ソースアプリケーションコードを前記ターゲットアプリケーションコードにマイグレーションするための前記変化の量が、前記ソースアプリケーション環境のサイズおよび複雑さに基づいて計算される、請求項１に記載の方法。

10

【請求項３】

前記リファクタリングされたコードが、継続的インテグレーションおよび展開フレームワーク（１５０）および自動化されたテストフレームワーク（１５０）を利用して生成される、請求項１に記載の方法。

【請求項４】

前記評価統計（３０２）が、前記ソースアプリケーションコードの前記マイグレーションを完了するための少なくとも１つの阻害要因を提供する、請求項１に記載の方法。

【請求項５】

前記ソースアプリケーション環境のコンポーネント間のリンクを保持しながら、前記ターゲットアプリケーションコードに従って前記マクロサービス（４２６a、…、４２６n）をリパッケージすることが完了される、請求項１に記載の方法。

20

【請求項６】

前記マクロサービス（４２６a、…、４２６n）がAIエンジン（１５２）によって特定される、請求項１に記載の方法。

【請求項７】

前記プロセッサ（１２２）によって、前記ターゲットアプリケーション環境に従って前記ソースアプリケーション環境からライブラリおよび設計パターンをマイグレーションすることをさらに含む、請求項１に記載の方法。

【請求項８】

前記プロセッサ（１２２）によって、前記ターゲットアプリケーション環境に従って前記ソースアプリケーション環境からのセキュリティルールを実装することをさらに含む、請求項１に記載の方法。

30

【請求項９】

ゲートウェイおよび継続的インテグレーションおよび継続的展開フレームワーク（１５０）を利用するアプリケーションプログラムインターフェースプロキシを生成することにより、前記プロセッサ（１２２）によって、各サービスを別個のコンテナにパッケージ化および構築することをさらに含む、請求項１に記載の方法。

【請求項１０】

前記プロセッサ（１２２）によって、ビジネスロジック、リンク、ルールエンジン、利用可能な環境のライブラリ、標準ツール、およびコーディング言語を含む複数のストレージユニット（１４４）にアクセスすることをさらに含む、請求項１に記載の方法。

40

【請求項１１】

アプリケーション環境マイグレーションのためのシステムであって、前記システムは、プロセッサ（１２２）と、

前記プロセッサ（１２２）に結合されたメモリ（１２６）であって、前記プロセッサ（１２２）は、前記メモリ（１２６）に格納された複数のモジュール（１２８）を実行し、前記複数のモジュール（１２８）は、

ソースアプリケーション環境のソースアプリケーションコードを評価し、かつ前記ソースアプリケーションコードをターゲットアプリケーションコードにマイグレーションする

50

ための変化の量を確認するための評価モジュール（１３０）であって、前記評価モジュール（１３０）は、前記ソースアプリケーションコードの前記マイグレーションを完了するための少なくとも１つの機能的準備（３０４）およびタイムライン（３０６）を提供する評価統計（３０２）を予測する評価モジュール（１３０）と、

ビジネスロジックを理解するために前記ソースアプリケーションコードをスキャンし、かつ前記ソースアプリケーションコードのためのリファクタリングされたコードを生成するためのリファクタリングモジュール（１３２）であって、前記リファクタリングされたコードは、前記ビジネスロジックを保持しながら前記ターゲットアプリケーションコードに従って、前記ソースアプリケーションコードをマクロサービス（４２６ a、…、４２６ n）に分割し、かつ前記マクロサービス（４２６ a、…、４２６ n）をリパッケージすることによって生成されるリファクタリングモジュール（１３２）と、

10

予測された前記評価統計（３０２）および前記リファクタリングされたコードに従って前記ソースアプリケーション環境のコンポーネントを更新し、かつ前記ソースアプリケーション環境をターゲットアプリケーション環境にマイグレーションするためのリプラットフォームモジュール（１３４）であって、前記マイグレーションは、前記ソースアプリケーション環境の更新された前記コンポーネントおよび前記リファクタリングされたコードを前記ターゲットアプリケーション環境にリプラットフォームするリプラットフォームモジュール（１３４）と、を含む、システム。

【請求項１２】

前記ソースアプリケーションコードをターゲットアプリケーションコードにマイグレーションするための前記変化の量が、前記ソースアプリケーション環境のサイズおよび複雑さに基づいて計算される、請求項１１に記載のシステム。

20

【請求項１３】

前記評価統計（３０２）が、前記ソースアプリケーションコードの前記マイグレーションを完了するための少なくとも１つの阻害要因を提供する、請求項１１に記載のシステム。

【請求項１４】

前記ソースアプリケーション環境のコンポーネント間のリンクを保持しながら、前記ターゲットアプリケーションコードに従って前記マクロサービス（４２６ a、…、４２６ n）をリパッケージすることが完了される、請求項１１に記載のシステム。

【請求項１５】

前記マクロサービス（４２６ a、…、４２６ n）がＡＩエンジン（１５２）によって特定される、請求項１１に記載のシステム。

30

【請求項１６】

前記リファクタリングモジュール（１３２）は、前記プロセッサ（１２２）によって、前記ターゲットアプリケーション環境に従って前記ソースアプリケーション環境からライブラリおよび設計パターンをマイグレーションすることをさらに含む、請求項１１に記載のシステム。

【請求項１７】

前記リプラットフォームモジュール（１３４）は、前記プロセッサ（１２２）によって、前記ターゲットアプリケーション環境に従って前記ソースアプリケーション環境からのセキュリティルールを実装することをさらに含む、請求項１１に記載のシステム。

40

【請求項１８】

前記リファクタリングモジュール（１３２）は、ゲートウェイおよび継続的インテグレーションおよび継続的展開フレームワーク（１５０）を利用するアプリケーションプログラムインターフェースプロキシを生成することにより、前記プロセッサ（１２２）によって、各サービスを別個のコンテナにパッケージ化および構築することをさらに含む、請求項１１に記載のシステム。

【請求項１９】

前記複数のモジュール（１２８）は、前記プロセッサ（１２２）によって、ビジネスロジック、リンク、ルールエンジン、利用可能な環境のライブラリ、標準ツール、およびコ

50

ーディング言語を含む複数のストレージユニット（１４４）にアクセスすることをさらに実行する、請求項１１に記載のシステム。

【請求項２０】

前記ソースアプリケーション環境は、ソースからターゲット環境への再発展、ソースからターゲットアプリケーションへのリファクタリング、ソースからターゲットアプリケーションへのリホスティング、およびソースからターゲットアプリケーションへのリプラットフォームフォーミングのステップを含む前記ターゲットアプリケーション環境へマイグレーションする、請求項１１に記載のシステム。

【発明の詳細な説明】

【技術分野】

10

【０００１】

本開示は概して、コンピュータの分野に関し、より具体的には、本開示は、構造的健全性と品質基準を維持しつつ、ユーザーが面倒な手間および時間をかけることなく、アプリケーションをより古い環境からより新しい高速な環境に自動的にマイグレーションさせることができるアプリケーション環境マイグレーションのためのシステムおよび方法に関する。

【背景技術】

【０００２】

時代の変化とともに技術も変化し、車輪の発明から、我々は今では、モバイルデバイスを介して通勤がオーダーされ得る時代に立っている。このように、テクノロジーは、コンピュータが舞踏室のように大きい時代から、親指のつめのように小さい今の時代へと、大きな飛躍を遂げた。我々は、少しずつ、複数のアプリケーションを通じて人類を助ける技術を刻みました。この絶え間なく変化し、絶えず変化するテクノロジーの分野では、アプリケーション環境は改善を続けており、それらの高度な要件により合わせてアプリケーションを更新し続ける必要が生じる。

20

【０００３】

従来、アプリケーションは、新しい環境に合わせてアプリケーションを完全に再現することでより新しい環境にマイグレーションされてきたが、時間がかかり、既存のアプリケーションの利益をあまり提供していなかった。環境の更新がより速くなると、アプリケーションがより新しい環境にマイグレーションされるまでにより新しいそれがリリースされるので、アプリケーションを手動で更新する時間が不十分である。さらに、限定されないが、アプリケーションを異なる環境にマイグレーションする際には、変換中にロジックが失われないようにすること、および古い環境およびより新しい環境に対応したマイグレーションを保証することを含むその独自の課題もあった。

30

【０００４】

企業がそのＩＴ戦略の重要な部分としてクラウドを活用することで、クラウドの採用が主流になった。これは、ＡＷＳ、Ａｚｕｒｅ、およびＧＣＰなどの主要なクラウドプロバイダーが過去３年間で年々成長していることから明らかである。最初のフェーズとして、これらの企業は、クラウドネイティブ設計の原則を使用してクラウド上に新しいアプリケーションを開発することにより、クラウドの採用に成功した。それで成功を実現したので、企業はクラウドの利益を最大限に活用するために、それらの重要なアプリケーションの残りをクラウドにマイグレーションする方法を模索している。しかし、課題は、既存のアプリケーションをクラウドにマイグレーションすることで、別の一連の問題が発生することであり、例えば、適切なクラウドマイグレーションアプローチ、リスクのバランス、クラウドマイグレーションを完了するためのコストとタイムライン、既存のアプリケーションの複雑さ、特にＳＭＥのない古いアプリケーション、総所有コスト（ＴＣＯ）の節約を最大化し、将来のリリースの生産性、実行速度、パフォーマンスの向上などを増加させることである。

40

【０００５】

アプリケーション環境のマイグレーションには複雑なパラメータおよび労力が必要なた

50

め、定期的な更新に間に合うようにクランチと一緒に、アプリケーションおよびそのコンポーネントに関連するさまざまな問題に対処しながら、手動による支援を必要とせずに、または最小限の方法でアプリケーションをある環境から別の環境にマイグレーションする、アプリケーション環境マイグレーションのためのシステムおよび方法を開発する必要がある。

【発明の概要】

【0006】

この要約は、アプリケーション環境マイグレーションのためのシステムおよび方法に関連する概念を紹介するために提供されており、その概念は、以下の詳細な説明でさらに説明される。この要約は、クレームされた主題の本質的な特徴を特定することを意図したものである。この要約は、クレームされた主題の範囲を決定または制限するために使用することを意図したものである。

【0007】

一実施態様では、アプリケーション環境マイグレーションのための方法が開示される。方法は、アプリケーションサーバのプロセッサによって、ソースアプリケーション環境のソースアプリケーションコードを評価することと、プロセッサによって、ソースアプリケーションコードをターゲットアプリケーションコードにマイグレーションするための量子変化を予測することと、プロセッサによって、評価統計を予測することを含み、評価統計は、ソースアプリケーションコードのマイグレーションを完了するための少なくとも1つの機能的準備およびタイムラインとを提供する。方法は、プロセッサによって、ビジネスロジックを特定するためにソースアプリケーションコードをスキャンすることと、プロセッサによって、ソースアプリケーションコードのためのリファクタリングされたコードを生成することとをさらに含み、リファクタリングされたコードは、ビジネスロジックを保持しつつターゲットアプリケーションコードに従ってソースアプリケーションコードをマクロサービスに分割し、マクロサービスをリパッケージすることによって生成される。方法は、プロセッサによって、予測された評価統計およびリファクタリングされたコードに従ってソースアプリケーション環境のコンポーネントを更新し、それによってプロセッサによって、ソースアプリケーション環境をターゲットアプリケーション環境にマイグレーションすることをさらに含み、マイグレーションは、ソースアプリケーション環境の更新されたコンポーネントおよびリファクタリングされたコードをターゲットアプリケーション環境へリプラットフォームする。

【0008】

さらに別の実施態様では、方法は、アプリケーションのサイズおよび複雑さに基づいてソースアプリケーションコードを計算されたターゲットアプリケーションコードにマイグレーションするための量子変化を含む。

【0009】

さらに別の実施態様では、方法は、継続的インテグレーションおよび展開フレームワークおよび自動化されたテストフレームワークを利用して生成されたリファクタリングされたコードを有する。

【0010】

さらに別の実施態様では、方法は、ソースアプリケーションコードのマイグレーションを完了するための少なくとも1つの阻害要因を提供する評価統計を有する。

【0011】

さらに別の実施態様では、方法は、ソースアプリケーションコードのコンポーネント間のリンクをスキャンすることを含むソースアプリケーションコードをスキャンする。

【0012】

さらに別の実施態様では、方法は、ソースアプリケーションコードのコンポーネント間のリンクを保持しながら、ターゲットアプリケーションコードの完了に従ってマクロサービスをリパッケージすることを含む。

【0013】

さらに別の実施態様では、方法は、A Iエンジンによって特定されるマクロサービスを有する。

【0014】

さらに別の実施態様では、方法は、プロセッサによって、ターゲットアプリケーション環境に従ってソースアプリケーション環境からライブラリおよび設計パターンをマイグレーションすることを含む。

【0015】

別の実施態様では、方法は、プロセッサによって、ターゲットアプリケーション環境に従ってソースアプリケーション環境からのセキュリティルを実装することを含む。

【0016】

別の実施態様では、方法は、ゲートウェイおよび継続的インテグレーションおよび継続的展開フレームワークを利用してアプリケーションプログラムインターフェースプロキシを生成することにより、プロセッサによって各サービスを個別のコンテナにパッケージ化および構築することを含む。

【0017】

別の実施態様では、方法は、プロセッサによって、ビジネスロジック、リンク、ルールエンジン、利用可能な環境のライブラリ、標準ツール、およびコーディング言語を含む複数のストレージユニットにアクセスすることを含む。

【0018】

別の実施態様では、方法は、ソースアプリケーション環境を、ソースからターゲット環境への再発展、ソースからターゲットアプリケーションへのリファクタリング、ソースからターゲットアプリケーションへのリホスティング、およびソースからターゲットアプリケーションへのリプラットフォームのステップを含むターゲットアプリケーション環境へマイグレーションさせる。

【0019】

一実施態様では、アプリケーション環境マイグレーションのためのシステムが開示される。システムは、プロセッサと、プロセッサに結合されたメモリとを備え、プロセッサは、メモリに格納された複数のモジュールを実行する。複数のモジュールはさらに、ソースアプリケーション環境のソースアプリケーションコードを評価し、かつソースアプリケーションコードをターゲットアプリケーションコードにマイグレーションするための量子変化を確認するための評価モジュールを有し、評価モジュールは、ソースアプリケーションコードのマイグレーションを完了するための少なくとも1つの機能的準備およびタイムラインを提供する評価統計を予測する。複数のモジュールは、ビジネスロジックを理解するためにソースアプリケーションコードをスキャンし、かつソースアプリケーションコードのためのリファクタリングされたコードを生成するためのリファクタリングモジュールをさらに含み、リファクタリングされたコードは、ビジネスロジックを保持しつつターゲットアプリケーションコードに従ってソースアプリケーションコードをマクロサービスに分割し、マクロサービスをリパッケージすることによって生成される。複数のモジュールは、予測された評価統計およびリファクタリングされたコードに従ってソースアプリケーション環境のコンポーネントを更新し、かつソースアプリケーション環境をターゲットアプリケーション環境にマイグレーションするためのリプラットフォームモジュールをさらに含み、マイグレーションは、ソースアプリケーション環境の更新されたコンポーネントおよびリファクタリングされたコードをターゲットアプリケーション環境にリプラットフォームする。

【0020】

さらに別の実施態様では、システムは、アプリケーションのサイズおよび複雑さに基づいてソースアプリケーションコードを計算されたターゲットアプリケーションコードにマイグレーションするための量子変化を含む。

【0021】

さらに別の実施態様では、システムは、継続的インテグレーションおよび展開フレーム

10

20

30

40

50

ワークおよび自動化されたテストフレームワークを利用して生成されたリファクタリングされたコードを有する。

【0022】

さらに別の実施態様では、システムは、ソースアプリケーションコードのマイグレーションを完了するための少なくとも1つの阻害要因を提供する評価統計を有する。

【0023】

さらに別の実施態様では、システムは、ソースアプリケーションコードのコンポーネント間のリンクをスキャンすることを含むソースアプリケーションコードをスキャンする。

【0024】

さらに別の実施態様では、システムは、ソースアプリケーションコードのコンポーネント間のリンクを保持しながら、ターゲットアプリケーションコードの完了に従ってマクロサービスのリパッケージを含む。

【0025】

さらに別の実施態様では、システムは、AIエンジンによって特定されるマクロサービスを有する。

【0026】

さらに別の実施態様では、システムは、ターゲットアプリケーション環境に従ってソースアプリケーション環境からライブラリおよび設計パターンをマイグレーションするリファクタリングモジュールを含む。

【0027】

別の実施態様では、システムは、ターゲットアプリケーション環境に従ってソースアプリケーション環境からセキュリティルールを実装するリプラットフォームモジュールを含む。

【0028】

別の実施態様では、システムは、ゲートウェイおよび継続的インテグレーションおよび継続的展開フレームワークを利用してアプリケーションプログラムインターフェースプロキシを生成することにより、各サービスを個別のコンテナにパッケージ化して構築するリファクタリングモジュールを含む。

【0029】

別の実施態様では、システムは、ビジネスロジック、リンク、ルールエンジン、利用可能な環境のライブラリ、標準ツール、およびコーディング言語を含む複数のストレージユニットへのアクセスを実行する複数のモジュールを含む。

【0030】

さらに別の実施態様では、システムは、ソースアプリケーション環境を、ソースからターゲット環境への再発展、ソースからターゲットアプリケーションへのリファクタリング、ソースからターゲットアプリケーションへのリホスティング、およびソースからターゲットアプリケーションへのリプラットフォームのステップを含むターゲットアプリケーション環境へマイグレーションさせる。

【0031】

あるプラットフォームから別のプラットフォームにアプリケーションをマイグレーションするために使用され得るアプリケーション環境マイグレーションのためのシステムおよび方法を提供することが、主題の主な目的であり、より具体的には、それは、アプリケーションをより古い環境からより新しい環境にマイグレーションするために使用され得る。アプリケーション環境マイグレーションのシステムおよび方法は、マイグレーションされるアプリケーション、ならびにアプリケーションが開発されたより古い環境およびアプリケーションがマイグレーションされるべきより新しい環境を含む環境に基づいてカスタマイズされ得る。

【0032】

多くのアプリケーションビジネスロジックを維持しながら、それらに対応するアプリケーション環境マイグレーションを提供することは、主題の別の目的である。さらに、アプ

10

20

30

40

50

リケーション環境マイグレーションのためのシステムおよび方法は、選択されたマイグレーション対象のアプリケーションを、広範囲の環境から別の広範囲の環境へマイグレーションすることができるようにしてもよい。

【0033】

手動入力を必要とせずに自動化された方法でアプリケーションの環境をマイグレーションする、アプリケーション環境マイグレーションのためのシステムおよび方法を提供することは、主題の別の目的である。

【0034】

開発チームによって行われる動作の繰り返し可能なパターンを排除し、それによってコストも削減することによって、アプリケーションをより古い環境からより新しい環境にマイグレーションするために開発チームが費やす時間と労力を削減するアプリケーション環境マイグレーションを提供することは、主題の別の目的である。

10

【0035】

開発チームが既存のアプリケーションをより古い環境からより新しい環境にマイグレーションするために利用され得るアプリケーション環境マイグレーションを提供することは、主題の別の目的である。

【0036】

既存のコードのチャンクを調査し、それを分析し、それを処理し、そしてその後それをリパッケージするパーサーのように動作するアプリケーション環境マイグレーションを提供することは、主題の別の目的である。

20

【0037】

JAVA（登録商標）または.NETでコーディングされたウェブアプリケーションのようなより古い環境からクラウドインターフェースのようなより新しい環境にアプリケーションをマイグレーションするために利用され得るアプリケーション環境マイグレーションを提供することは、主題の別の目的である。

【0038】

実装コストを削減し、総所有コストを削減して収益性を高め、実行速度を向上させ、アプリケーションのパフォーマンスを向上させ、そしてアプリケーションの生産性を向上させることによって、アプリケーションをより古い環境からより新しい環境にマイグレーションするアプリケーション環境マイグレーションを提供することは、主題の別の目的である。

30

【0039】

特定の態様、実施形態、実施態様および／または構成に応じて多くの利点を提供することは、主題の別の目的である。

【0040】

動作の労力およびコストを制御しながら、信頼性の高い実行、スケーラビリティ、および付加価値サービスを提供できるプラットフォームを提供することは、主題の別の目的である。

【0041】

多数のインスタンスを同時に効率的に管理し、さまざまな規制要件で作業し、リソースがユーザーフレンドリーなインターフェースと緊密に、効率的に、そして集合的に協働および連携できるようにすることは、主題の別の目的である。

40

【0042】

主題のこれらおよび他の実施態様、実施形態、プロセス、および特徴は、以下の詳細な説明が付随する実験の詳細とともに読まれるときに、より完全に明らかになるであろう。しかしながら、主題の前述の要約およびそれに関する以下の詳細な説明の両方は、1つの潜在的な実施態様または実施形態を表し、本開示または主題の他の代替の実施態様もしくは実施形態を限定するものではない。

【図面の簡単な説明】

【0043】

50

上に要約された主題の主要な特徴の明確な理解は、主題の方法およびシステムを説明する添付の図面を参照することによって得ることができるが、そのような図面は、主題の好ましい実施形態を描写し、したがって、主題が考慮することができる他の実施形態に関して、その範囲を制限すると見なされるべきではない。したがって：

【図 1】本主題の実施形態による、アプリケーション環境マイグレーションの方法を示す概略モジュール図を示す。

【図 2】本主題の実施形態による、例示的なアプリケーション環境マイグレーションシステムの動作を説明するシステム図を示す。

【図 3】本主題の実施形態による、例示的なアプリケーション環境マイグレーションシステムによって自動的に生成される例示的な評価レポートを示す。

10

【図 4】本主題の実施形態による、例示的なアプリケーション環境マイグレーション方法の例示的な図を示す。

【図 5】本主題の実施形態による、アプリケーション環境マイグレーションの方法の例示的なフローチャートを示す。

【図 6】本主題の実施形態による、アプリケーション環境マイグレーションの方法の別の例示的なフローチャートを示す。

【発明を実施するための形態】

【0044】

(発明の詳細な説明)

以下は、添付の図面に示されている本開示の実施態様の詳細な説明である。実施態様は、開示を明確に伝えるように詳細である。しかしながら、提供される詳細の量は、実態態様を制限することを意図するものではなく、本開示の精神および範囲内にあるすべての改変物、同等物、および代替物を網羅することを意図している。アプリケーション環境マイグレーションのための記載されたシステムおよび方法の態様は、任意の数の異なるコンピューティングシステム、環境、および／または構成で実装され得るが、実施形態は、以下の例示的なシステム（複数可）の文脈で説明される。

20

【0045】

本開示は、ソースアプリケーション環境からターゲットアプリケーション環境へのアプリケーションクラウドリプラットフォームを提供する。アプリケーションのクラウドリプラットフォームは、既存のオンプレミスアプリケーションをクラウドにマイグレーションして、クラウドのスケラビリティおよび柔軟性のある機能を利用することにより、ソースアプリケーション環境を効率的に機能させることを含む。ソースアプリケーション環境の構造も改変されて、これらのアプリケーションの総所有コスト（TCO）が削減され、これは、仮想マシン、アプリケーションサーバ、およびデータベースのライセンスコストを削減するために、よりオープンソースでクラウドフレンドリーなソフトウェアを採用することを含む。実際のTCO削減量はアプリケーションごとに異なり、アプリケーションの評価後に決定される。本開示は、アプリケーションで実行されるいくつかの重要なステップをリストし、アプリケーションごとに異なり、Cloud PlatformまたはPaaSで効果的に機能するように、既存のアプリケーションを変更し得るが、ソースおよびオンプレミスの設計パターンに関連するすべてのブロックが変更され、基盤となるすべてのライブラリが更新されて、アプリケーションがコンテナまたはPaaS、アプリケーションサーバ（WebLogic、WebSphere、JBossなど）と互換性を持つようになり、そしてデータベースはオープンソースのクラウドフレンドリーなソフトウェアに置き換えられ得、モノリスアプリケーションは効率および速度のためにより小さなマイクロサービスに分離され得る。

30

40

【0046】

本開示は、単純なコンテナ化によるクラウドリプラットフォームのためのメカニズムを提供し、コンテナ化を妨げるすべてのコードブロックを削除し、Dockerなどのコンテナにソースアプリケーションを展開してクラウド上でそれを動作させることを含む。さらに、テストケースおよびCI/CDの自動化を含む、ソースアプリケーションリリ

50

ースプロセスも自動化されよう。またさらに、ソースアプリケーションおよび／またはソースデータベースをTomcat for JAVA（登録商標）アプリケーションなどの軽量のオープンソースサーバにリプラットフォームして、ライセンスコストを削減し、クラウドでのコンピューティングコストを削減することで、これらのアプリケーションのTCOを削減することもできる。しかし、安定性およびパフォーマンスを保証するために、モノリスアプリケーションをより小さなサービスに分割する必要がある、これらのアプリケーションをすべてステートレスにし、それぞれを個別のコンテナに展開する個別のTomcatサーバにそれらを展開する。したがって、ソースアプリケーションは、モノリスアプリケーションを分離し、マクロサービス概念を使用してAPIを介してそれを公開し、それをクラウド準備アプリケーションにすることで、ターゲットアプリケーションプラットフォームにマイグレーションされよう。モノリシックアプリケーションをマクロサービスに分割することに加えて、CI／CDも実装され、高速で自動化された展開を実現する。コンポーネントごとの構造とステップごとの方法を以下に示す。

【0047】

図1は、本主題の実施形態による、アプリケーション環境マイグレーションの方法を示す概略モジュール図100を示す。

【0048】

一実施態様では、アプリケーション環境マイグレーションシステム120は、サーバ102上でアプリケーション環境マイグレーションのための方法を実装し、システム120は、プロセッサ（複数可）122、インターフェース（複数可）124、フレームワーク（複数可）150、AIエンジン152、およびプロセッサ（複数可）122に結合されたまたはプロセッサ（複数可）122と通信しているメモリ126を含む。プロセッサ（複数可）122は、1つ以上のマイクロプロセッサ、マイクロコンピュータ、マイクロコントローラ、デジタル信号プロセッサ、中央処理装置、状態機械、論理回路、および／またはアプリケーション環境マイグレーション命令に基づいて信号を操作する任意のデバイスとして実装され得る。他の機能の中でも、プロセッサ（複数可）122は、メモリ126に格納されたコンピュータ可読命令をフェッチして実行するように構成される。フレームワーク150は、継続的インテグレーションおよび継続的展開フレームワーク、自動化されたテストフレームワークなどを含むがこれらに限定されない。AIエンジン152は、データベース（DB）リプラットフォームエンジン、アプリリプラットフォームエンジン、バッチおよびスケジューラリプラットフォームエンジン、メッセージブローカーリプラットフォームエンジン、オープンソースAIエンジンなどを含むが、これらに限定されない。

【0049】

本開示は、システムがサーバ上のアプリケーションとして実装されるシナリオを考慮して説明されているが、システムおよび方法は、様々なコンピューティングシステムに実装され得る。記載された方法（複数可）を実装し得るコンピューティングシステムは、メインフレームコンピュータ、ワークステーション、パーソナルコンピュータ、デスクトップコンピュータ、ミニコンピュータ、サーバ、マルチプロセッサシステム、ラップトップ、タブレット、SCADAシステム、スマートフォン、モバイルコンピューティングデバイスなどを含むが、これらに限定されない。

【0050】

インターフェース（複数可）124は、システム120がユーザーと対話することを可能にする、例えば、ウェブインターフェース、グラフィカルユーザーインターフェースなどの様々なソフトウェアおよびハードウェアインターフェースを含み得る。さらに、インターフェース（複数可）124は、システム120が、ウェブサーバおよび外部データサーバ（図示せず）などの他のコンピューティングデバイスと通信することを可能にし得る。インターフェース（複数可）124は、例えばLAN、ケーブルなどの有線ネットワーク、およびWLAN、セルラー、または衛星などの無線ネットワークを含む、多種多様なネットワークおよびプロトコルタイプ内の複数の通信を促進し得る。インターフェース（

10

20

30

40

50

複数可) 124は、多数のデバイスを互いにまたは別のサーバに接続するための1つ以上のポートを含んでもよい。

【0051】

アプリケーションサーバおよびクラウド環境におけるすべての要素間で通信するために使用されるネットワークは、無線ネットワーク、有線ネットワーク、またはそれらの組み合わせであり得る。ネットワークは、イントラネット、ローカルエリアネットワークLAN、ワイドエリアネットワークWAN、インターネットなどの異なるタイプのネットワークのうちの1つとして実装され得る。また、ネットワークは、専用ネットワークであっても、共有ネットワークであってもよい。共有ネットワークは、例えば、ハイパーテキスト転送プロトコルHTTP、伝送制御プロトコル／インターネットプロトコルTCP/IP、無線アプリケーションプロトコルWAPなどの様々なプロトコルを使用して互いに通信する異なるタイプのネットワークのつながりを表す。さらにネットワークは、ルータ、ブリッジ、サーバ、コンピューティングデバイスを含む様々なネットワークデバイスを含んでもよい。ネットワークはさらに、クライアントサイトコンピュータ、ホストサイトサーバもしくはコンピュータ、クラウド上、またはそれらの組み合わせなどにあるストレージデバイスへのアクセスを有する。ストレージは、1つまたは多数のメモリ記憶デバイス、データベースなどを含む、1つまたは多数のローカルおよびリモートのコンピュータ記憶媒体を有する。

【0052】

メモリ126は、例えば、揮発性メモリ（例えば、RAM）、および／または不揮発性メモリ（例えば、EPROM、フラッシュメモリ等）を含む本分野で既知の任意のコンピュータ可読媒体を含み得る。一実施形態では、メモリ126は、モジュール（複数可）128およびシステムデータ140を含む。

【0053】

モジュール128は、評価モジュール130、リファクタリングモジュール132、リプラットフォームモジュール134、およびブロック評価モジュール、スクリプトジェネレータモジュール等を含むがこれらに限定されない他のモジュール136をさらに含む。スクリプトジェネレータモジュールはさらに、テストケースジェネレーター、ビルドスクリプトジェネレーター、ドッカースクリプトジェネレーター、パイプラインスクリプトジェネレーター、展開スクリプトジェネレーターなどを含むが、これらに限定されない。このようなモジュールは、単一のモジュールとして表されてもよいし、異なるモジュールの組合せとして表されてもよいことが理解されよう。さらに、メモリ126は、とりわけ、モジュール128のうちの1つ以上によってフェッチ、処理、受信、および生成されたデータを格納するためのリポジトリとして機能するシステムデータ140をさらに含む。システムデータ140は、例えば、ストレージ144における運用データ、ワークフローデータ、および他のデータを含む。システムデータ140は、場合により、144a、144b、...、144nで表されるストレージ144を有する。一実施形態では、システムデータ140は、ウェブまたはクラウドネットワークを介して他のデータベースへのアクセスを有する。ストレージ144は、評価モジュールデータ、リファクタリングモジュールデータ、リプラットフォームモジュールデータ、ライブラリ、リンク識別子、データベースアダプタ、データベースディクショナリ、データベースパーサー、技術ディクショナリ、ファイルアダプタ、アプリケーションパーサー、パターン認識子、オープンソースライブラリ、技術スタックなどを含むが、これらに限定されない複数のデータベースを含む。このようなデータベースは、単一のデータベースとして表されてもよいし、異なるデータベースの組合せとして表されてもよいことが理解されよう。一実施形態では、データは、データ構造の形態でメモリ126に格納され得る。さらに、前述のデータは、関係データモデルまたは階層データモデルなどのデータモデルを使用して編成され得る。

【0054】

サーバ102は、ターゲットアプリケーション環境サーバ104、および別のターゲットアプリケーション環境サーバ106、ならびにデータベース108を含むがこれに限定

されないデータベースを含む複数のコンピューティングシステムを有するクラウド環境 110 にさらに接続されるまたは接続可能であるまたはその一部である。コンピューティングシステムおよびデータベースは、クラウドコンピューティングルールの下で互いに通信し、また、ウェブまたは他の利用可能な通信媒体／プロトコルの下でサーバ 102 と通信することが必須である。ターゲットアプリケーションサーバ 104、106 のコンピューティングシステムは、一般に、ネットワークによって接続され、かつネットワークを介して通信する複数のコンピューティングデバイスを含む分散処理システムである。ソフトウェアアプリケーションは、特定のクラウドコンピューティング環境／システムにおける 1 つ以上のコンピューティングデバイスにわたって実行するようにそれらを構成することによって、「クラウドで」実行され得る。クラウドコンピューティングシステムのコンピューティングデバイスは、それぞれソフトウェアアプリケーションの個別のコピーを実行してもよいし、場合によっては、ソフトウェアアプリケーションの動作が異なるコンピューティングデバイス間で分割されて並列実行されてもよい。クラウドコンピューティングシステムは、アプリケーションを実行するために利用可能なリソースを表す複数のクラウドコンピューティングインスタンスを含んでもよい。各インスタンスは、特定の能力（ストレージサイズ、処理速度、ネットワーク帯域幅など）を有する物理コンピューティングデバイスであってもよいし、または特定の能力を有する仮想コンピューティングデバイスであってもよい。特定のクラウドコンピューティングシステムは、ソフトウェアアプリケーションを実行するための異なる能力セットを有する異なるインスタンスタイプを提供し得る。

【0055】

一実施態様では、最初に、マイグレーション専門家、データベース管理者、アプリケーション開発者、品質アナリストまたはテスト専門家を含むがこれらに限定されないユーザーが、ユーザーアクセスデバイスを使用して、サーバ 102 を使用してインターフェース 124 を介してシステム 120 にアクセスし得る。システム 120 ならびに関連する方法および関連するモジュール、サブモジュール、方法の動作は、以下に説明する図 2、図 3、図 4、図 5、および図 6 を用いて詳細に説明され得る。

【0056】

一実施形態では、システム 120 は、サーバ 102 上のインターフェース 124 を介してユーザー命令データを受信する。システム 120 のモジュール 128 は、システムデータ 140、他のモジュール 136、フレームワーク 150、およびサポートコンポーネントを使用しながら、プロセッサ 122 を使用して命令を処理する。アプリケーション環境マイグレーションのための評価モジュール 130 は、ある環境から別の環境へのアプリケーションのマイグレーションを開始する前に利用される。リスク、コスト、タイムライン、および変更がターゲットアプリケーション環境にマイグレーションする他の依存アプリケーションに与える影響を評価しながら、評価モジュール 130 は、ソースアプリケーション環境の評価を実行して、ソースアプリケーション環境が受けなければならない量子変化の深い理解を確認する。評価モジュール 130 は、様々なコーディングスタイルを有する様々な開発者によって開発された基礎となるアプリケーションソースコードを理解する。さらに、AI エンジン 152 を用いて評価モジュール 130 によって生成された予測評価レポート 302 は、必要なタイムライン、変更する必要があるコードの割合、複数の準備パラメータ、複数の阻害要因またはコードブロックおよび理由を伴う警告、コードを変更する必要がある強調表示された部分、モノリスを分割できるサービスの数、モノリスを分割できるサービスの種類、ならびに継続的インテグレーションおよび継続的展開のためのアプリケーションの準備に関するがこれらに限定されない深い詳細を提供する。評価モジュール 130 によって生成された評価レポート 302 は、ソースアプリケーション環境のマイグレーションを完了するための正確な見積もりおよびタイムラインを提供する。一実施形態では、RPA ボットは、プロアクティブアプリケーション環境マイグレーション予測（PAEMF）アルゴリズムを使用して、評価レポート 302 を生成するように構成される。一実施形態では、RPA ボットは、ソフトウェアボットまたはソフトウェアボッ

トおよびハードウェアボットの組み合わせである。一実施形態では、ソフトウェアボットは、プロセッサがAIを利用することによってロボティックプロセスオートメーションを実行することを可能にするコンピュータプログラムである。別の実施形態では、ボットは、ハードウェアおよびソフトウェアの組み合わせを有し、ハードウェアは、特に、アプリケーション環境マイグレーションのためのロボティックプロセス自動化を可能にする機能を実行するために特に専用のメモリ、プロセッサ、コントローラおよび他の関連するチップセットを含む。

【0057】

アプリケーション環境マイグレーションのためのリファクタリングモジュール132は、モノリスウェブアプリケーションの既存のコードを変換し、ビジネスロジック（複数可）を変更せずにターゲットアプリケーション環境に対応するリファクタリングされたコードを生成することにより、ソースアプリケーション環境マイグレーションからターゲットアプリケーション環境マイグレーションへの自動化されたリプラットフォームニングを支持する。リファクタリングされたコードは、ソースアプリケーションコードをマクロサービスに分解し、ビジネスロジックを保持したままターゲットアプリケーションコードに従ってマクロサービスをリパッケージすることによって生成される。リファクタリングモジュール132は、継続的インテグレーションおよび展開フレームワーク150、自動化されたテストフレームワーク150、ならびにミドルウェアリプラットフォームおよび環境に優しい設計パターンを利用して、ソースアプリケーション環境のためのリファクタリングされたコードを生成する。

【0058】

アプリケーション環境マイグレーションのためのリプラットフォームモジュール134は、アプリケーションのサイズならびに基礎となるライブラリの更新、アプリケーションモノリスからのフロントエンドおよびバックエンドの結合解除、バックエンドを新しい環境に従って複数のサービスに分割、各サービスを個別のコンテナにパッケージングおよび構築、ゲートウェイを利用したアプリケーションプログラムインターフェースプロキシ生成、継続的インテグレーションおよび展開フレームワーク150を生成することなどを含むがこれに限定されないアクティビティを実行することによるその複雑性に基づいて、ある環境から別の環境へアプリケーションを変換するために、ターゲットアプリケーション環境へのソースアプリケーション環境マイグレーションのための自動化されたリプラットフォームニングを支持する。リプラットフォームモジュール134は、評価レポート302の予測評価統計およびリファクタリングされたコードに従って、ソースアプリケーション環境のサポートコンポーネントを更新し、これにより、ソースアプリケーション環境の更新されたサポートコンポーネントおよびリファクタリングされたコードをターゲットアプリケーション環境にリプラットフォームニングすることにより、ソースアプリケーション環境をターゲットアプリケーション環境にマイグレーションする。リプラットフォームモジュール134はまた、ソースアプリケーション環境のパターンをターゲットアプリケーション環境の設計パターンに統合すること、コードを微調整すること、展開の問題を解決すること、マイグレーションしたアプリケーションが完全に機能することを確認するためにセキュリティを実装することなどを含むがこれに限定されないアクティビティを実行する。リプラットフォームモジュール134は、リホスティング環境マイグレーション、アプリケーションリプラットフォームニング、アプリケーションリファクタリングおよびネイティブ環境開発などを含むがこれらに限定されない、複数の形態の環境マイグレーションを支援する。

【0059】

他のモジュール136のブロック評価モジュール、オンプレミス設計パターンを有するモノリシックウェブアプリケーションのような任意のアプリケーションは、アプリケーション環境マイグレーションのためのシステムおよび方法に入力され得、アプリケーション環境マイグレーションのためのシステムおよび方法のブロック評価モジュールは、モノリシックウェブアプリケーションのオンプレミス設計パターンのパターンを発見し得る。さ

らに、この情報は、図1に示すように、アプリケーション環境マイグレーションのためのシステムおよび方法の自動化されたリプラットフォームモジュールに転送され得る。

【0060】

システム120の参照アーキテクチャはさらに、テストジェネレーター—テストジェネレーターは、生成されたAPIのためのユニット、機能および性能テストケースのためのテストケースを生成するなどのサポートコンポーネントを有する。ビルドスクリプトジェネレーター—ビルドスクリプトジェネレーターは、アプリケーションを正常に構築して実行するためのスクリプトを生成する。Dockerスクリプトジェネレーター—Dockerスクリプトジェネレーターは、アプリケーションとその依存関係をコンテナ化するためのスクリプトを生成する。パイプラインスクリプトジェネレーター—パイプラインスクリプトジェネレーターは、開発、テスト、および本番環境用のCI/CDパイプラインを作成するためのスクリプトを生成する。展開スクリプトジェネレーター—展開スクリプトジェネレーターは、アプリケーションをターゲット環境に展開するためのスクリプトを生成する。データベースアダプター—データベースアダプターユーティリティは、IBM DB2、Oracle、Microsoft SQL Server、Sybase ASE、およびPostgreSQLなどの複数のデータベースサーバへの接続を確立する。データベースディクショナリー—データベースディクショナリーは、上記の複数のデータベースサーバにより使用される膨大なデータベース関連のキーワードと組み込み関数を含む。データベースパーサー—データベースパーサーは、データベースディクショナリーを使用してデータベースオブジェクトを解析し、ターゲットデータベースと互換性のないパターンを見つける。AIエンジン—AIエンジンは、レガシーアプリケーションの既存および潜在的なマクロ/マイクロサービスを特定する。DBリプラットフォームエンジン—DBリプラットフォームエンジンは、表、制約、シーケンス、トリガー、関数プロシージャなどのソースデータベースオブジェクトを、ターゲットデータベース互換オブジェクトにリプラットフォームする。ファイルアダプター—ファイルアダプターは、アプリケーションのソースファイルへのアクセスを提供する。ファイルはスキャンされてレポートが作成され、最終的にターゲットアプリケーションとしてリプラットフォームされる。テクノロジーディクショナリー—テクノロジーディクショナリーは、モノリスアプリケーション内で使用されるテクノロジー関連の膨大なパターンを含む。アプリケーションパーサー—アプリケーションパーサーは、パターン認識子を使用してアプリケーションソースコードを解析し、アプリケーションが使用しているテクノロジーのインベントリ全体を見つける。このインベントリは、評価とリプラットフォームに使用される。パターン認識子—パターン認識子は、テクノロジーディクショナリーに基づいてパターンを認識し、アプリケーションで使用されているテクノロジーを見つける。アプリリプラットフォームエンジン—アプリリプラットフォームエンジンは、アプリケーションパーサーによって解析されたソースアプリケーションをリプラットフォームする。このエンジンは、クラウドに展開される準備ができていないアプリケーションを生成する。JAVA（登録商標）1.8；Maven；JUnit、JUnit、XUnit JMeter；Docker Jenkins；AWS CLI、Azure CLIなどを含むがこれに限定されないテクノロジースタック。

【0061】

図2は、本主題の実施形態による、例示的なアプリケーション環境マイグレーションシステムの動作を説明するシステム図200を示す。

【0062】

本開示は、ソースアプリケーション環境評価および所望のターゲットプラットフォームへのソースアプリケーションコード変換である2つの重要なアクティビティを示す。この2つのアクティビティは、以下にフローチャート200で説明される。

【0063】

一実施形態では、JAVA（登録商標）アプリケーションまたは.NETアプリケーションなどのモノリスウェブアプリケーション202は、ソースアプリケーション環境のソースアプリケーションである。モノリスウェブアプリケーション202は、データベース

、バッチ、インテグレーション、アプリ、スケジューラ、メッセージブローカなどを含むがこれらに限定されない様々なサポートコンポーネント204にリンクされる。

【0064】

第1のステップでは、クラウドブロッカー評価206は、ソースアプリケーションを評価し、コード障害、ソースコード洞察、およびオンプレミス設計パターン214を含む評価レポート212を生成する。オンプレミスの設計パターン214は、EJB、RMI、STRUTS/SPRING、SOAP/XML、JSP/ASP、COM/COM++、ESBなどを含むが、これらに限定されない。ソースアプリケーションの評価は、現在のアプリケーションのテクノロジスタックとコード分析を含む。この評価は、ソースアプリケーションをリプラットフォームングに伴う時間、労力、およびリスクへの自動化された洞察を提供する。評価はコード分析から始まり、それはソースアプリケーションの静的コードの自動化された評価を実行し、アプリケーションと準備パラメータのコンテナ化されたクラウドマイグレーションに失敗する阻害要因またはブロッカ（パッケージ/フレームワーク/API/ライブラリなど）を特定し、それによって、評価レポート212としてさらに詳細な詳細評価レポート302を提供する。それは、アプリケーション、データベース、バッチプログラムおよびスケジューラ、インテグレーションメッセージブローカー、構築および展開モジュールを含むソースアプリケーションエコシステム全体についてこのコード評価を実行する。クラウドブロッカー評価206は、リスク、コスト、タイムライン、および他の依存コンポーネントおよびアプリケーションへの影響を評価するために、ソース環境が通過しなければならない変化の量（quantum of changes）を深く理解するようになる。レポートは主に、変更が必要なコードの割合、コード準備、ブロッカおよび理由を伴った警告、コードを変更する必要がある正確な場所、モノリスが分解され得るサービスの数、DevOpsおよびCI/CD準備、ならびにそのような実行のタイムラインを提供する。

【0065】

次のステップでは、ソース環境からターゲット環境へのアプリケーションおよびサポートコンポーネントのリプラットフォームングのための自動化された変換/リプラットフォームング208を実行して、ターゲットアプリケーションでソースアプリケーションのビジネスロジックを保持しながら、JAVA（登録商標）アプリケーションまたは.NETアプリケーション、継続的インテグレーションおよび展開フレームワーク216、自動化されたテストフレームワーク218、ミドルウェアリプラットフォーム220、およびソースアプリケーション環境のソースコードをリファクタリングした後のクラウドフレンドリーな設計パターン222で構成されるOracleデータベースまたはDB2データベースのような、クラウド準備アプリケーションおよびサポートコンポーネント210を生成する。自動化されたリプラットフォームング208のサブステップは、基礎となるライブラリおよびその他のサポートコンポーネントの更新、モノリスからのフロントエンドおよびバックエンドの分離、バックエンドを複数のマクロサービスに分解、各マクロサービスを個別のコンテナにパッケージ化して構築することであり、ウェブフロントとマクロサービスとの統合、コードの微調整、展開問題の解決および、セキュリティの実装などの残りのアクティビティは、アプリケーションが完全に機能するようにコンサルタントによって完了されるだろう。一例では、アプリケーション環境変換は、Zuulを用いたAPIプロキシの生成、API用テストフレームワークの生成、CI/CDフレームワークの生成をさらに含む。一例では、表Aおよび表Bを例として、例示的なコード変換を説明するために用いられ得る。

10

20

30

40

50

【表 A】

表A: 例示的なコード変換機能

S. No.	ソーステクノロジー	ターゲットテクノロジー
1	セッション EJB	スプリングブート
2	エンティティEJB	JPA
3	メッセージ駆動型EJB	Spring JMS
4	バッチ	Spring batch
5	SOAPサービス	SOAPサービス + RESTエンドポイント
6	JAVA POJOサービス	RESTエンドポイントを用いたSpringブート
7	. NET	. NETコア
8	Telerik ORM	Entity framework
9	Weblogic/WebSphere	Tomcat
10	Oracle/SQLサーバ/DB 2	Postgres

10

【表 B】

表B: 例示的なコード生成機能

S. No.	機能
1	APIテスト自動化
2	JunitおよびNunitのテストケース(評価は手動)
3	APIプロキシ(Zuul)
4	OAuth2認証
5	CI/CDフレームワーク

20

【0066】

クラウドリプラットフォームニングアクティビティは、オンプレミス設計パターンからクラウドフレンドリーな設計パターン（Springフレームワーク上のEJBからPOJOへ）のソースコードの変換、それをクラウドコンテナまたはPaaS、JDK、Spring、Mybatisなどと互換性を持たせるために、アプリケーションで使用される基礎的なライブラリの更新、バックエンドビジネスロジックを複数のサービスに分離し、重厚なWeblogicまたはWebSphereアプリケーションサーバをTomcatサーバへのリプラットフォーム、ユニットテストケースの開発、APIテストの自動化、CI/CDパイプラインなどを含む。さらに、アプリケーションクラウドリプラットフォームニングは、コード解析、コードリファクタリング、ライブラリ更新、ビルドスクリプトのコンパイル、展開スクリプト、コンテナ化スクリプトなど、いくつかの反復的なステップを含む。この結果は、市場投入までの時間が大幅に改善され、次いで全体的なコストを削減し、高品質のリプラットフォームニングを実現し得る。

30

【0067】

図3は、本主題の実施形態による、例示的なアプリケーション環境マイグレーションシステムによって自動的に生成される例示的な評価レポート302を示す。

【0068】

一実施形態では、評価モジュール130は、コード障害、ソースコード洞察、およびオンプレミス設計パターン214を含むソースアプリケーション環境を評価した後、評価レポート302を生成する。評価レポート302は、ソースアプリケーション環境をリプラ

40

50

ットフォーミングに伴う時間、労力、およびリスクへの自動化された洞察を提供する。

【0069】

評価コード分析は、ソースアプリケーションの静的コードの自動化された評価を実行し、アプリケーションと準備パラメータのコンテナ化されたクラウドマイグレーションに失敗する阻害要因またはブロック（パッケージ／フレームワーク／API／ライブラリなど）を特定し、それによって、詳細な評価レポート302を提供する。レポートの生成は、ソースアプリケーションとターゲットアプリケーション環境に基づいたいくつかのユーザー構成を伴う自動化されたプロセスである。

【0070】

評価レポート302は、リスク、コスト、タイムライン、および他の依存コンポーネントおよびアプリケーションへの影響を評価するために、ソースアプリケーションが通過しなければならない変化の量の深い理解を準備パラメータ304および視覚的構成306の形態で得る。変化の量は、アプリケーションマイグレーションに必要なすべての変更の合計である。評価レポートは様々なセクションを有し、アプリケーションスナップショットのためのいくつかの例示的な図は、とりわけ、コンテナ化準備304a、リプラットフォーム準備304b、DevOps準備304c、マクロ／マイクロAPI準備304n、スケジューラリプラットフォーム準備304n、バッチ準備、メッセージブローカ準備、DBリプラットフォーム準備304n等のパラメータを含む。各パラメータは、準備状態をパーセンテージ(%)などで示し、それぞれのパラメータにどれだけの準備ができているか、および／またはどれだけの準備ができていない／ブロック／阻害要因状態が存在するかを示す。例えば、コンテナ化準備304aは、準備状態と必要とされるリファクタリング／阻害要因率を備えたコンテナ準備率を示し、リプラットフォーム準備304bは、準備状態と必要とされるリファクタリング／阻害要因率を備えたTomcat準備率を示し、DevOps準備304cは、準備状態とCI準備／阻害要因率を備えたCD準備率を示し、マクロ／マイクロAPI準備304nは、準備状態とマイクロAPI既存／阻害要因状態を備えたマクロAPI既存ポインターを示し、スケジューラのリプラットフォーム準備304nは、AWSバッチ準備状態とAzure Logic App準備／阻害要因率を備えたリプラットフォーム準備率を示し、そしてDBリプラットフォーム準備304nは、PostgreSQL準備状態と必要とされるリファクタリングブロッカー／阻害要因率などを備えたリプラットフォーム準備率を示す。レポートの他のセクションの1つは、データベースオブジェクトスプレッド [#Objects] 306a、データベースオブジェクトのスプレッド [#Loc] 306b、Numberによるプラットフォーム構成306n、Locによるプラットフォーム構成306n等の視覚的構成306がある。一例では、データベースオブジェクトのスプレッド [#Objects] 306aは、トリガー、シーケンス、パッケージ、プロシージャ、ビュー、マテリアライズドビュー、関数および表のスプレッドまたはカウント数または割合を示し、データベースオブジェクトのスプレッド [#Loc] 306bは、トリガー、シーケンス、パッケージ、プロシージャ、ビュー、マテリアライズドビュー、関数および表のスプレッドまたはカウント数または割合を示し、数306nによるプラットフォーム構成は、JMS、XML構成、ウェブサービス、Mavenビルド、構成、HTML、Servlet、スタイルシート、reactJS、JSP、JUnit、POJOのスプレッドまたはカウント数または割合を示し、Loc306nによるプラットフォーム構成は、JMS、XML構成、ウェブサービス、Mavenビルド、構成、HTML、Servlet、スタイルシート、reactJS、JSP、JUnit、POJOなどのスプレッドまたはカウント数または割合を示す。

【0071】

さらに一例では、評価レポート302の様々なセクションは、以下の情報セクション、使用されるフレームワーク／ライブラリ／パッケージからなるアプリケーションスナップショット、アプリケーションコンテナ化準備、TomcatまたはKestrelアプリケーションサーバへのアプリケーションサーバのリプラットフォーム準備、Post

10

20

30

40

50

greSQLなどへのデータベースサーバのリプラットフォーム準備、(Docker/Kubernetesへの)コンテナ化準備、CI/CD準備(ユニット/機能/性能テストケース、構築および継続的インテグレーション、利用可能な自動化)、マクロおよびマイクロAPI候補、API準備(アプリケーション内の既存および潜在的API)を提供する。

【0072】

本方法は、ソースアプリケーションの環境を評価し、その後、クラウドにフレンドリーな設計パターンを使用してクラウドにそれらを変換することにより、エンドツーエンドの変換能力を提供し、最終的にあなたの利害関係者および意思決定者に迅速な意思決定のための透明な可視性を提供することにより、この行程全体を管理する。データベースからバッチプログラム、スケジューラからメッセージブローカに至るまで、アプリケーションの依存をすべて全体的にクラウドに変換しよう。評価レポート302のアプリケーション分析セクションでは、方法は、ソースアプリケーション全体を分析し、クラウドマイグレーションを阻害する設計パターンを含むソースコード内のブロックを特定する。また、すべての依存も分析され、設計および計画目的のために全体的なビューを提供する。それは、アプリケーションコンポーネントのインベントリ、テクノロジスタックおよびその依存関係を発見および分析し、アプリケーションのビジネス、データサービスおよびそのAPI準備を特定し、コンテナ化およびクラウドリプラットフォームのブロック&DevOps準備を特定し、そしてJAVA(登録商標) vX.X からJAVA(登録商標) v1.8;EJB(Session Bean/Entity Bean/MDB)からSpring Framework(Pojo)を伴うSpring Boot Service;.NET vX.X から .NET Core v2.2 などの変更のための取り組みを特定する。

【0073】

評価レポート302のデータベース分析セクションでは、方法は、データベースを分析して、データベースインベントリ、データオブジェクトの詳細およびデータベースサイズを分析および発見し、データベースリプラットフォーム準備およびブロックを特定し、データベースリプラットフォームのためのアプリケーションブロッカーおよび依存を特定し、そしてOracle/SQL Server/DB2/SybaseからPostgres/MySQLなどの変更のための取り組みを特定する。評価レポート302のバッチ&スケジューラ分析セクションでは、方法は、バッチ&スケジューラを分析して、バッチプログラムインベントリとバッチスケジューラの分析および発見、バッチスケジューラリプラットフォーム準備、ブロックと依存関係の特定、バッチリプラットフォームのためのアプリケーションバッチサービスの特定、Autosys/ControlMからAWS Batch(AWS Cloud)/Azure Logic App(Azure Cloud)/Quartz(Private Cloud)などの変更のための取り組みを特定する。評価レポート302のメッセージブローカ分析セクションでは、方法は、メッセージブローカを分析し、メッセージブローカーインベントリとメッセージングオブジェクトを分析および発見し、メッセージブローカーリプラットフォーム準備、ブロックおよび依存を特定し、メッセージブローカーリプラットフォームのためのアプリケーションブロッカーおよび依存、TIBCO EMS/RendezvousまたはWebLogic/WebSphereまたはメッセージキューまたはSonicメッセージキューからアクティブMQなどの変更のための取り組みを特定する。

【0074】

図4は、本主題の実施形態による、例示的なアプリケーション環境マイグレーション方法の例示的な図400を示す。

【0075】

一実施形態では、本主題の実施形態による、ターゲットアプリケーション環境へのソースアプリケーション環境マイグレーションの方法およびシステムを用いて、事前および事後リプラットフォームを有するアプリケーション展開ビュー400が描かれる。モノリス

アプリケーションは一般に、サーバ102によって常駐またはアクセス可能なソースアプリケーション環境に属しており、クラウド環境110によって常駐またはアクセス可能なターゲットアプリケーション環境に属するモダナイズされたリプラットフォームされたアプリケーションに変換またはマイグレーションする。まず、ソースアプリケーション環境のターゲットアプリケーション環境へのマイグレーションのために、コード評価206が起こる。アプリケーションの評価は、コンテナ化のためのコード阻害要因を除去し、変更するために、コード変更に必要な変更の量を決定する。マイグレーションは、以下のステップで実行され、サーバ102によってアクセス可能なようにアプリケーションを有するアプリケーションウェブサーバは、場合により、ウェブフロントインスタンス422a、422b、...、422nに変換される。アプリケーションサーバ（IISクラスタ）202は、様々なAppマクロサービスインスタンス426a、426b、426c、...、426nを有するAPI管理層426にさらにリファクタリングされる。マクロサービスインスタンスは、ソースアプリケーションのサイズおよび複雑性に基づく。一実施態様では、リファクタリングモジュール132は、マクロサービスのファクタリングおよびリパッケージング208をターゲットアプリケーション環境に実行する。

【0076】

ソースアプリケーションデータベース204はまた、ターゲットデータベース424のリファクタリングされたPaaS/CaaSクラスタ424aである。ウェブフロントインスタンス422a、422b、...、422nおよびマクロサービスインスタンス426a、426b、426c、...、426nはさらに、クラウドコンピューティング環境110のサービスメッシュ、AKSクラスタ、Appサービス、ACIなどを含むがこれに限定されないコンポーネントにリンクされる。一実施態様では、ソースOracleデータベースは、クラウド環境110のコンテナ/オーケストレーションプラットフォームとともに、PostgreSQLデータベースにマイグレーションされる。

【0077】

ソースアプリケーションの仮想マシンインスタンス408a、408b、...、408nは、クラウド環境110のコンテナクラスタにマイグレーションされる。このように、オンプレミスエコシステム412を有するソースアプリケーション環境410のオンプレミスデータセンター410は、クラウド環境110のプライベート/パブリッククラウド428にリプラットフォーム210/マイグレーションされる。一実施態様では、EBS、キャッシング、ローカルファイル、ルールエンジン、バッチおよびスケジューラ、メッセージブローカ、認証プロトコル、セキュリティプラットフォームなどを含むがこれらに限定されないオンプレミスエコシステム412のコンポーネント（412a、412b、...、412n）は、Azureキュー、イベントハブ、サービスバス、AppInsights、Redisキャッシュ、ストレージ、Vnet、Azure ADなどを含むがこれらに限定されない、独立または標準サービス/コンポーネント（430a、430b、430c、...、430n）プライベート/パブリッククラウド428にリプラットフォーム210/マイグレーションされる。

【0078】

一実施態様では、方法は、コンテナまたはPaaSを使用してクラウド上で効率的に動作するように既存のオンプレミスアプリケーションを修正し、ソースコードと設計パターンに関連するすべてのブロック、および基礎となるすべてのライブラリを更新または変更して、それらをコンテナまたはPaaSと互換性を持たせ、アプリケーションサーバとデータベースを再配置してTCOを削減し、そしてモノリスアプリケーションは効率と速度を向上させるためにサービスに切り離され得る。

【0079】

一例では、表Cは、リプラットフォームマイグレーションを説明するための例として使用され得る。

10

20

30

40

50

【表 C】

表C: 例示的な現状の状態から将来の状態へ

現状の状態(As-Is State)	将来の状態(To-be State)
JSF、Tiles、Servlets、Struts	JSF、Tiles、Servlets、Struts
(EJB) Session Bean	REST Client、Redis HttpSession Cache
(EJB) Entity Bean、Hibernate	Tomcatサーバ
	APIゲートウェイ
	REST API
	Spring Bootマクロサービス
	(ORM) JPA、Hibernate
WebLogic App Server 12c	Tomcatサーバ9
Oracle Database 12c	PostgreSQL

10

【0080】

一実施態様では、現状の状態は、JSPベースのUIからいくつかのデータサービスを有するセッションEJB呼び出しおよび商用アプリケーションサーバを有するモノリシックウェブアプリケーションまでであり、セッションの永続性およびステートフルサービスは現状の状態であり、マイグレーションされた将来の状態は、APIゲートウェイを介したコンテナ上のJSF Web FrontおよびRESTful Spring Bootマクロサービス（独立したスケーリング）を有する分離されたウェブアプリケーションである。

20

【0081】

図5は、本主題の実施形態による、アプリケーション環境マイグレーションの方法の例示的なフローチャート500を示す。

【0082】

一実施態様では、アプリケーション環境マイグレーションのための方法500が示される。方法は、コンピュータ実行可能命令の一般的な文脈で説明され得る。一般に、コンピュータ実行可能命令は、特定の機能を実行するまたは特定の抽象データ型を実装するルーチン、プログラム、オブジェクト、コンポーネント、データ構造、手順、モジュール、機能などを含み得る。方法は、通信ネットワークを介してリンクされるリモート処理デバイスによって機能が実行される分散コンピューティング環境でも実施され得る。分散コンピューティング環境において、コンピュータ実行可能命令は、メモリ記憶デバイスを含むローカル及びリモートのコンピュータ記憶媒体の両方に配置され得る。方法が記載され、限定として解釈されることを意図しない順序は、記載された方法ブロックの任意の数を任意の順序で組み合わせて、方法または代替方法を実施し得る。さらに、個々のブロックは、本明細書に記載される開示の精神及び範囲から逸脱することなく、方法から削除され得る。さらに、方法は、任意の適切なハードウェア、ソフトウェア、ファームウェア、またはそれらの組み合わせで実施され得る。しかし、説明を容易にするために、以下に説明する実施形態では、本方法は、上述したシステムに実装され得る。

30

40

【0083】

ステップ／ブロック502において、アプリケーションサーバ102のシステム120のプロセッサ122によって、ソースアプリケーション環境のソースアプリケーションコードを評価する。一実施態様では、ソースアプリケーションコードは、システム120の評価モジュール130によって評価され得る。

【0084】

ステップ／ブロック504において、ソースアプリケーションコードをターゲットアプ

50

リケーションコードにマイグレーションさせるための量子変化を確認する。一実施態様では、量子変化は、評価モジュール130によって確認され得る。

【0085】

ステップ／ブロック506において、ソースアプリケーションコードのマイグレーションを完了するための少なくとも1つの機能的準備とタイムラインとを提供する評価統計を予測する。一実施態様では、評価統計は、評価モジュール130によって予測され得る。

【0086】

したがって、方法500は、予測された評価統計およびソースアプリケーション環境からターゲットアプリケーション環境へのマイグレーションのためのタイムラインを提供することによって、アプリケーション環境マイグレーションを支援する。

【0087】

図6は、本主題の実施形態による、アプリケーション環境マイグレーションの方法の別の例示的なフローチャートを示す。

【0088】

一実施態様では、アプリケーション環境マイグレーションのための方法500が示される。方法は、コンピュータ実行可能命令の一般的な文脈で説明され得る。一般に、コンピュータ実行可能命令は、特定の機能を実行するまたは特定の抽象データ型を実装するルーチン、プログラム、オブジェクト、コンポーネント、データ構造、手順、モジュール、機能などを含み得る。方法は、通信ネットワークを介してリンクされるリモート処理デバイスによって機能が実行される分散コンピューティング環境でも実施され得る。分散コンピューティング環境において、コンピュータ実行可能命令は、メモリ記憶デバイスを含むローカル及びリモートのコンピュータ記憶媒体の両方に配置され得る。方法が記載され、限定として解釈されることを意図しない順序は、記載された方法ブロックの任意の数を任意の順序で組み合わせて、方法または代替方法を実施し得る。さらに、個々のブロックは、本明細書に記載される開示の精神及び範囲から逸脱することなく、方法から削除され得る。さらに、方法は、任意の適切なハードウェア、ソフトウェア、ファームウェア、またはそれらの組み合わせで実施され得る。しかし、説明を容易にするために、以下に説明する実施形態では、本方法は、上述したシステムに実装され得る。

【0089】

ステップ／ブロック602において、アプリケーションサーバ102のシステム120のプロセッサ122によって、ソースアプリケーション環境のソースアプリケーションコードを評価する。一実施態様では、ソースアプリケーションコードは、システム120の評価モジュール130によって評価され得る。

【0090】

ステップ／ブロック604において、ソースアプリケーションコードをターゲットアプリケーションコードにマイグレーションさせるための量子変化を確認する。一実施態様では、量子変化は、評価モジュール130によって確認され得る。

【0091】

ステップ／ブロック606において、ソースアプリケーションコードのマイグレーションを完了するための少なくとも1つの機能的準備とタイムラインとを提供する評価統計を予測する。一実施態様では、評価統計は、評価モジュール130によって予測され得る。

【0092】

ステップ／ブロック608において、ビジネスロジックを特定するために、ソースアプリケーションコードをスキャンする。一実施態様では、ソースアプリケーションコードは、リファクタリングモジュール132によってスキャンされ得る。

【0093】

ステップ／ブロック610において、ソースアプリケーションコードをマクロサービスに分解し、かつビジネスロジックを保持しながらターゲットアプリケーションコードに従ってマクロサービスをリパッケージすることによって、ソースアプリケーションコードのためのリファクタリングされたコードを生成する。一実施態様では、リファクタリングさ

10

20

30

40

50

れたコードは、リファクタリングモジュール 1 3 2 によって生成され得る。

【0094】

ステップ／ブロック 6 1 2 において、予測された評価統計およびリファクタリングされたコードに従って、ソースアプリケーション環境のコンポーネントを更新する。一実施態様では、ソースアプリケーション環境のコンポーネントは、リプラットフォームモジュール 1 3 4 によって更新され得る。

【0095】

ステップ／ブロック 6 1 4 において、ソースアプリケーション環境の更新されたコンポーネントおよびリファクタリングされたコードをターゲットアプリケーション環境にリプラットフォームすることにより、ソースアプリケーション環境をターゲットアプリケーション環境にマイグレーションする。一実施態様では、ソースアプリケーション環境は、リプラットフォームモジュール 1 3 4 によってターゲットアプリケーション環境にマイグレーションされ得る。

【0096】

このように、方法 6 0 0 は、予測評価を提供し、コードをリファクタリングし、ソースアプリケーション環境からターゲットアプリケーション環境へのマイグレーションのためにアプリケーションをリプラットフォームすることによって、アプリケーション環境マイグレーションを支援する。

【0097】

本ソリューションは、複雑なアプリケーションをクラウドにリプラットフォームし、生産環境にそれらを展開するために使用される。本主題の実施形態による、ターゲットアプリケーション環境へのソースアプリケーション環境マイグレーションの方法及びシステムを用いたいくつかの例が示される。以下では、準備、計画、および実行を、検討の結果とともに示す。一例として、先導する二次抵当会社の J A V A（登録商標）バッチアプリのクラウドリプラットフォームでは、ソースアプリケーション環境は、米国で事業を行う数十億ドルの企業であり、上場企業でもある不動産担保証券を提供する先導する二次抵当会社に所属する。主なビジネス課題は、O r a c l e データベースの使用と並んでメッセージ駆動型ビーン E J B（MDB） P o j o S e r v i c e s、J D B C などを使用した、10 年前に A u t o s y s & W e b L o g i c M Q 上の E J B で開発された複雑な J A V A（登録商標）バッチアプリケーションである。このアプリケーションは、ビジネスクリティカルであり、重要な長時間稼働のプロセスを処理する。このアプリケーションを開発した主要な S M E が利用できないため、クラウドリプラットフォームに課題を抱えていた。本ソリューションアプローチを用いて、O r a c l e データベースは、方法ステップを使用して P o s t g r e s にリプラットフォームされ、A u t o s y s スケジューラは Q u a r t z にリプラットフォームされ、バックエンド E J B（MDB）は R E S T エンドポイントを備えた S p r i n g B o o t / B a t c h S e r v i c e A p p s を備えた独立した S p r i n g J M S にリプラットフォームされた。次いで、各サービスアプリは、A W S E C S V P C 上の個別のコンテナ上に展開され、無限のスケールビリティ、高可用性、高いフォールトトレランス、およびパフォーマンスの改善を提供する。A u t o s y s の集中型 J O B 依存は、アプリ固有のクォーツスケジューラを使用して自己管理依存にリファクタリングされ、運用の複雑さが軽減された。主な利益の中で、主に W e b L o g i c と A u t o s y s の排除とコンテナ化にもより、T C O が 5 0 % 以上削減され、マクロサービスと C I / C D 自動化により開発の生産性が向上し、自動化を使用すると、実施態様の総コストが 5 0 % 超削減され、実装は 6 週間で完了する（手動で実行した場合も同様に 9 か月かかる）。

【0098】

別の例では、本ソリューションは、先導する金融サービス会社の J A V A（登録商標）W e b アプリのクラウドリプラットフォームに使用され、ソースアプリケーション環境は、企業、機関投資家、および個人に対応する先導する金融サービス会社で、数十億ドル規模の会社であり、米国、ヨーロッパ、オーストラリア、アジア太平洋などで事業を行

10

20

30

40

50

う、米国最大の銀行の1つに属する。主なビジネス上の課題は、12年前にWebLogic (App ServerおよびMQ) でSession EJB、Entity Bean、Struts、JSPなどを使用して開発された複雑なJAVA (登録商標) ウェブアプリケーションである。このアプリケーションはビジネスクリティカルであり、主要な金融ビジネスプロセス、このアプリケーションを開発した主要なSMEが利用できないことを処理し、クラウドのリプラットフォームで課題を提起する。現在のソリューションアプローチを使用して、ソースアプリケーションはウェブフロントエンドとサービスバックエンドに分割されることになる。バックエンドEJB (Session Beans) は、RESTエンドポイントを備えた独立したSpring Boot Service アプリにリプラットフォームされる。その後、ウェブフロントと各サービスアプリはTomcatで、プライベートクラウド上のKubernetes Cluster上の個別のコンテナに展開され、無限のスケーラビリティ、高可用性、高いフォールトトレランス、およびパフォーマンスの改善を提供する。ウェブフロントおよびマクロサービスは、Redisキャッシュによりステートレス化される。サービスのREST APIエンドポイントはOAuth2によって保護される。ウェブフロントとすべてのマクロサービスに対して、自動化テスト+CI/CDが有効になるだろう。主な利益の中で、主にWebLogic & MQの廃止により、TCOが50%超削減され、実装コストを50%超削減し、手作業の労力を50~75%削減し、マクロサービスとCI/CD自動化により、開発者の生産性が最大25%向上し、クラウドのリプラットフォームは、手動で行った場合の8か月と比較して、わずか6週間で完了した。

【0099】

別の例では、我々を先導するコンサルティング会社の、NETウェブアプリケーションのクラウドリプラットフォームに本ソリューションが使用され、ソースアプリケーション環境は、世界150カ国の700超のオフィスに従業員がいる、コンテンツ管理会社で最大の専門サービスおよび会計監査機関の1つに所属する。主なビジネス課題は、ASP.NET、C#、Telerik ORM、およびSQL Serverを使用して何年もかけて開発されたComplex.NETウェブアプリケーションである。このアプリケーションは、先導するプロフェッショナルサービスプロバイダーの主要なB2C機能を扱うビジネスクリティカルなものである。このアプリケーションを開発した主要なSMEが利用できないため、この作業はさらに複雑になった。現在のソリューションアプローチを使用して、ソースアプリケーションはウェブフロントエンドとサービスバックエンドに分割されることになる。Telerik ORMを使用したバックエンドのレガシー.NETサービスは、RESTエンドポイントを持つエンティティフレームワークで独立した.NET Coreサービスアプリにリプラットフォームされる。ウェブフロントと各サービスアプリは、Kestrel & NGINXでAzureクラウド上の個別のコンテナに配置され、無限のスケーラビリティ、高可用性、高いフォールトトレランスおよびパフォーマンスの改善を提供する。ウェブフロントおよびマクロサービスは、Redisキャッシュによりステートレス化された。サービスのREST APIエンドポイントはOAuth2によって保護される。ウェブフロントとすべてのマクロサービスに対して、自動化テスト+CI/CDが有効になった。上位の利益の間では、リプラットフォームは、主にIIS & Telerik ORMの廃止によりTCOが45%超削減されることをもたらし、実装の総コストが45%超削減され、マクロサービスとCI/CD自動化により開発の生産性が上がり、手作業の場合7ヶ月かかるのと比較して6週間で完了した。

【0100】

アプリケーション環境マイグレーションのためのシステムおよび方法の実施態様は、構造的な特徴および／または方法に特有の言語で記述されるが、添付の請求項は、必ずしも記述された特定の特徴または方法に限定されないことが理解されるものとする。むしろ、特定の機能および方法は、アプリケーション環境マイグレーションのための実施態様の例として開示される。

10

20

30

40

50

【図面】
【図 1】

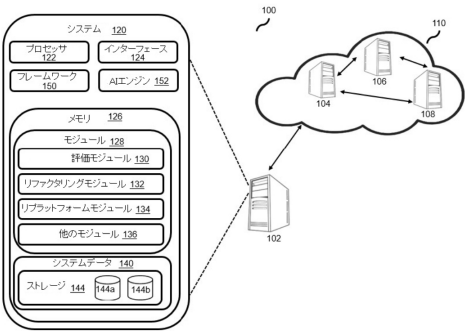


FIGURE 1

【図 2】

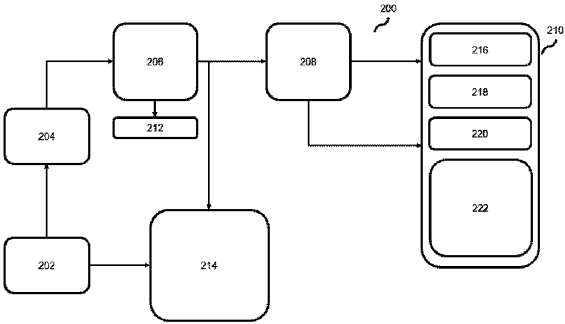


FIGURE 2

【図 3】

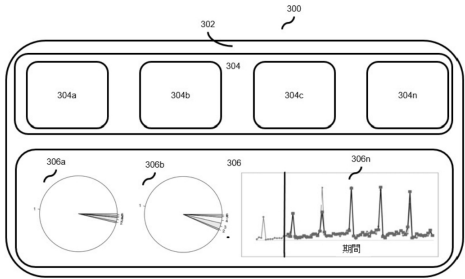


FIGURE 3

【図 4】

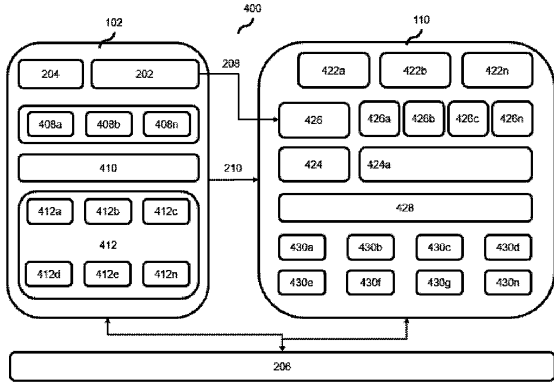


FIGURE 4

10

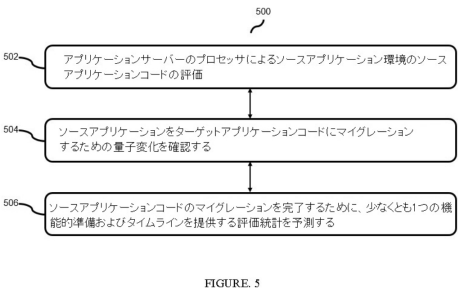
20

30

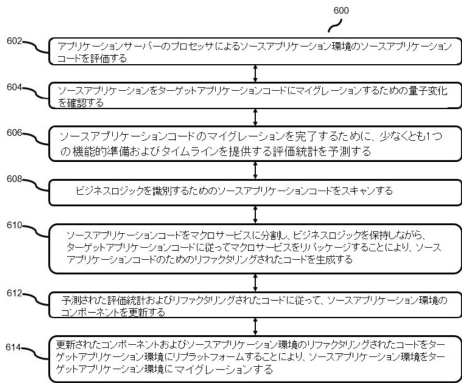
40

50

【図 5】



【図 6】



10

20

30

40

50

フロントページの続き

- (74)代理人 110003487
弁理士法人東海特許事務所
- (72)発明者 パル, カイロディップ
インド国, 6 0 0 0 4 2 チェンナイ, ベラッカリ, サブラマニウム コロニー, シャンムガ ア
パートメンツ, エフ 2
- (72)発明者 ガナパティ, ナタラジャン
インド国, 6 0 0 0 8 2 チェンナイ, ベトリ ナガー, ジェガナサン ストリート, ナンバー 1 ビー
- (72)発明者 ピー, ミーナクシスンドラム
インド国, 6 0 3 1 0 3 チェンナイ, パドゥアー, マントリ シナジー アパートメンツ, ビー
3 0 3
- 審査官 福西 章人
- (56)参考文献 欧州特許出願公開第 0 2 8 0 8 7 9 0 (EP, A2)
欧州特許出願公開第 0 3 5 2 5 0 9 7 (EP, A1)
欧州特許出願公開第 0 3 1 1 8 7 8 1 (EP, A1)
米国特許出願公開第 2 0 1 4 / 0 3 5 9 0 5 3 (US, A1)
米国特許出願公開第 2 0 1 6 / 0 0 9 4 4 8 3 (US, A1)
米国特許出願公開第 2 0 1 9 / 0 1 7 9 6 3 8 (US, A1)
特開 2 0 0 7 - 3 1 0 7 2 7 (JP, A)
- (58)調査した分野 (Int.Cl., DB名)
G 0 6 F 8 / 0 0 - 8 / 7 7
9 / 4 4 - 9 / 4 5 5
1 1 / 0 7
1 1 / 2 8 - 1 1 / 3 6