(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4496067号
(P4496067)
(45)発行日　平成22年7月7日(2010.7.7)　(24)登録日　平成22年4月16日(2010.4.16)

(51)Int.Cl.　F I
G06F 15/00 (2006.01)　G06F 15/00 310A

請求項の数 23 (全 20 頁)

(21)出願番号　特願2004-354945(P2004-354945)
(22)出願日　平成16年12月8日(2004.12.8)
(65)公開番号　特開2006-12113(P2006-12113A)
(43)公開日　平成18年1月12日(2006.1.12)
審査請求日　平成16年12月8日(2004.12.8)
審判番号　不服2007-23564(P2007-23564/J1)
審判請求日　平成19年8月28日(2007.8.28)
(31)優先権主張番号　03368113.1
(32)優先日　平成15年12月10日(2003.12.10)
(33)優先権主張国　欧州特許庁(EP)

(73)特許権者　390009531
インターナショナル・ビジネス・マシーンズ・コーポレーション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＳＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国１０５０４　ニューヨーク州　アーモンク　ニュー　オーチャード　ロード
(74)代理人　100108501
弁理士　上野　剛史
(72)発明者　マーク・フィアマンテ
フランス０６７００　サン・ローラン・ドュ・ヴァー　アレー・デ・シガール　２９５

最終頁に続く

(54)【発明の名称】サービス指向アーキテクチャ向けのサービス・インターフェースを自動的に生成するための方法およびシステム

(57)【特許請求の範囲】
【請求項１】
サービス指向アーキテクチャにおいて業務プロセスのサービスを呼び出すクライアントと、当該サービスを提供するプロバイダとの間の、マシン実施可能なアプリケーション向けのサービス・インターフェースを自動的に生成する方法であって、
前記アプリケーションを連携した業務プロセスのサービスを、サービス指向アーキテクチャにおける複数のサービス・インターフェースとして配置するステップと、
前記サービス・インターフェースによって定義されたサービスの使用を記録するサービス使用データをログ記録するステップと、
前記サービス使用データから、繰り返されるサービス使用パターンを選別して、対応する細粒度および／または粗粒度サービスの対応するセットを形成するために前記サービス使用パターンを見分けるステップと、
識別された繰り返されるサービス使用パターンに応答して、前記サービス使用パターンから選別し見分けられたいくつかの細粒度および／または粗粒度サービス・インターフェースを組み合わせることによって、新しい粗粒度サービス・インターフェースを生成するステップと、
を含む方法。
【請求項２】
前記新しいサービス・インターフェースを自動的に生成するステップが、１つまたは複数の既存のサービス・インターフェースを修正するステップを含むものである、請求項１

に記載の方法。
【請求項３】
前記新しいサービス・インターフェースが、前記アプリケーションを含む１つまたは複数の機能を具体化するものである、請求項１に記載の方法。
【請求項４】
前記新しいサービス・インターフェースを自動的に生成するステップが、いくつかの既存のサービス・インターフェースの機能を組み合わせるステップを含むものである、請求項３に記載の方法。
【請求項５】
前記サービスの使用を記録するデータをログ記録するステップが、サービス呼び出し署名を含むデータをサービス使用履歴データベースに格納するステップを含むものである、請求項１に記載の方法。
【請求項６】
前記サービス呼び出し署名が、前記サービスが実行される前および後にログ記録されるものである、請求項５に記載の方法。
【請求項７】
前記サービス呼び出し署名のデータが、呼び出し側コンテキスト、タイムスタンプ、および呼び出しパラメータのうちの任意の１つまたはそれらの組み合わせに関するデータと共に、前記サービス使用履歴データベースに格納されるものである、請求項５に記載の方法。
【請求項８】
前記呼び出し側コンテキストデータが、呼び出し者識別子および呼び出し側セッション識別子を含むものである、請求項７に記載の方法。
【請求項９】
名前、電話番号、および電子メール・アドレスのうちのいずれか１つまたはそれらの組み合わせを含む呼び出し者データを取り出すために、呼び出し者識別子データベースに接続するステップを含む、請求項８に記載の方法。
【請求項１０】
前記新しい粗粒度サービス・インターフェースが、配置済みのサービス・データベースに配置されるものである、請求項１に記載の方法。
【請求項１１】
前記新しい粗粒度サービス・インターフェースが、配置される前に、「予定の」粗粒度サービス定義データベースに格納されるものである、請求項１０に記載の方法。
【請求項１２】
前記新しい粗粒度サービス・インターフェースを自動的に生成する前記ステップが、人間のオペレータ、システム管理者、およびスケジュールリング・エンジンのうちのいずれか１つ、またはそれらの組み合わせによってトリガされるものである、請求項１に記載の方法。
【請求項１３】
前記新しい粗粒度サービス・インターフェースを自動的に生成する前記ステップが、繰り返されるサービス使用パターンによって自動的にトリガされるものである、請求項１に記載の方法。
【請求項１４】
前記新しい粗粒度サービス・インターフェースが、サービス指向アーキテクチャ（ＳＯＡ）に配置するために自動的に生成される、請求項１に記載の方法。
【請求項１５】
サービス指向アーキテクチャにおいて業務プロセスのサービスを呼び出すクライアントと、当該サービスを提供するプロバイダとの間の、マシン実施可能なアプリケーション向けのサービス・インターフェースを自動的に生成するためのシステムであって、
前記アプリケーションを連携した業務プロセスのサービスを、サービス指向アーキテク

チャにおける複数のサービス・インターフェースとして配置し、前記サービス・インターフェースによって定義されたサービスの使用を記録するデータをログ記録するサービス実行エンジンと、
ログ記録されたサービス使用データを格納するためのサービス履歴使用データベースと、
前記サービス使用データから、繰り返されるサービス使用パターンを識別するための、サービス要求分析部と、
識別された繰り返されるサービス使用パターンに応答して、使用可能な細粒度サービス・インターフェースのためのデータベースと、既存の粗粒度サービス・インターフェースのためのデータベースとを含むものである自律型自己最適化サービス定義コンポーネントにより新しいサービス・インターフェースを自動的に生成するサービス・インターフェース生成部と、
を含む、システム。
【請求項１６】
前記サービス実行エンジンがＪ２ＥＥに準拠したアプリケーション・サーバを含むものである、請求項１５に記載のシステム。
【請求項１７】
前記自律型自己最適化サービス定義コンポーネントが、推論エンジンと、前記サービス・インターフェース生成部を制御するための規則データベースとを含むものである、請求項１６に記載のシステム。
【請求項１８】
サービス指向アーキテクチャにおいて業務プロセスのサービスを呼び出すクライアントと、当該サービスを提供するプロバイダとの間の、マシン実施可能なアプリケーション向けのサービス・インターフェースを自動的に生成するためのシステムであって、
前記アプリケーションを連携した業務プロセスのサービスを、サービス指向アーキテクチャにおける複数のサービス・インターフェースとして配置するための手段と、
前記サービス・インターフェースによって定義された前記サービスの使用を記録するサービス使用データをログ記録するための手段と、
前記サービス使用データから、繰り返されるサービス使用パターンを選別して、対応する細粒度および／または粗粒度サービスの対応するセットを形成するために前記サービス使用パターンを見分ける手段と、
識別された繰り返されるサービス使用パターンに応答して、前記サービス使用パターンから選別し見分けられたいくつかの細粒度および／または粗粒度サービス・インターフェースを組み合わせることによって、新しい粗粒度サービス・インターフェースを生成するための手段と、
を含む、システム。
【請求項１９】
前記複数のサービス・インターフェースを配置するための前記手段がサービス実行エンジンを含むものである、請求項１８に記載のシステム。
【請求項２０】
サービス指向アーキテクチャにおいて業務プロセスのサービスを呼び出すクライアントと、当該サービスを提供するプロバイダとの間の、マシン実施可能なアプリケーション向けのサービス・インターフェースを自動的に生成するためのサービス・インターフェース生成装置であって、
前記サービス指向アーキテクチャにおけるサービス・インターフェースは、前記アプリケーションを連携して業務プロセスのサービスを提供するものであり、
Ａ）サービス使用データから、繰り返されるサービス使用パターンを選別して、対応する細粒度および／または粗粒度サービスの対応するセットを形成するために前記サービス使用パターンを見分けるため、および
Ｂ）識別された繰り返されるサービス使用パターンに応答して、前記サービス使用パタ

ーンから選別し見分けられたいくつかの細粒度および／または粗粒度サービス・インターフェースを組み合わせることによって、新しい粗粒度サービス・インターフェースを生成して、新しいサービス・インターフェースを自動的に生成するための、
サービス実行エンジンを含むものである、サービス・インターフェース生成装置。
【請求項２１】
サービス指向アーキテクチャにおいて業務プロセスのサービスを呼び出すクライアントと、当該サービスを提供するプロバイダとの間の、マシン実施可能なアプリケーション向けのサービス・インターフェースを自動的に生成するためのサービス・インターフェース生成装置であって、
前記サービス指向アーキテクチャにおけるサービス・インターフェースは、前記アプリケーションを連携して業務プロセスのサービスを提供するものであり、
サービス使用データから、繰り返されるサービス使用パターンを選別して、対応する細粒度および／または粗粒度サービスの対応するセットを形成するために前記サービス使用パターンを見分ける手段と、
識別された繰り返されるサービス使用パターンに応答して、前記サービス使用パターンから選別し見分けられたいくつかの細粒度および／または粗粒度サービス・インターフェースを組み合わせることによって、新しい粗粒度サービス・インターフェースを生成するための手段と、
を含む、サービス・インターフェース生成装置。
【請求項２２】
サービス指向アーキテクチャにおいて業務プロセスのサービスを呼び出すクライアントと、当該サービスを提供するプロバイダとの間の、マシン実施可能なアプリケーション向けのサービス・インターフェースを自動的に生成する方法を実行するように配置構成された、サービス実行装置であって、
前記アプリケーションを連携した業務プロセスのサービスを、サービス指向アーキテクチャにおける複数のサービス・インターフェースとして配置する手段と、
前記サービス・インターフェースによって定義されたサービスの使用を記録するサービス使用データをログ記録する手段と、
前記サービス使用データから、繰り返されるサービス使用パターンを選別して、対応する細粒度および／または粗粒度サービスの対応するセットを形成するために前記サービス使用パターンを見分ける手段と、
識別された繰り返されるサービス使用パターンに応答して、前記サービス使用パターンから選別し見分けられたいくつかの細粒度および／または粗粒度サービス・インターフェースを組み合わせることによって、新しい粗粒度サービス・インターフェースを生成するための手段と、
を含む、サービス実行装置。
【請求項２３】
サービス指向アーキテクチャにおいて業務プロセスのサービスを呼び出すクライアントと、当該サービスを提供するプロバイダとの間の、マシン実施可能なアプリケーション向けのサービス・インターフェースをコンピュータに自動的に生成させるためのコンピュータ・プログラムであって、
前記アプリケーションを連携した業務プロセスのサービスを、サービス指向アーキテクチャにおける複数のサービス・インターフェースとして配置するステップと、
前記サービス・インターフェースによって定義されたサービスの使用を記録するサービス使用データをログ記録するステップと、
前記サービス使用データから、繰り返されるサービス使用パターンを選別して、対応する細粒度および／または粗粒度サービスの対応するセットを形成するために前記サービス使用パターンを見分けるステップと、
識別された繰り返されるサービス使用パターンに応答して、前記サービス使用パターンから選別し見分けられたいくつかの細粒度および／または粗粒度サービス・インターフェ

ースを組み合わせることによって、新しい粗粒度サービス・インターフェースを生成するステップと、
を含む、コンピュータ・プログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、一般に、サービス指向アーキテクチャ（ＳＯＡ）向けのサービス・インターフェースを自動的に生成するための方法およびシステムに関し、具体的には、ＳＯＡでのウェブ・サービス向けのサービス・インターフェースを自動的に生成するための方法およびシステムに関するが、これに限定されるものではない。
【背景技術】
【０００２】
業務プロセスをサポートするために種々異なる情報システムおよびソフトウェア・アプリケーションのオペレーションを統合するプロセスは、従来、複雑で費用や時間のかかる仕事であった。プログラミング言語、オペレーティング・システム、アプリケーション・インターフェース、およびネットワーキング・プロトコルに関する標準がないという状況の中、種々異なる情報システムおよびソフトウェア・アプリケーションのオペレーションを統合する以前の方法では、複雑な技術に支えられた大規模なシステム・インテグレータ・リソースが必要であった。結果として生じる業務プロセスを実行するためのシステムおよびアプリケーションは、緊密に結合されたアプリケーションおよびサブシステムを備えるものである。その欠点の１つは、いずれか１つのサブシステムを変更すると、様々な従属アプリケーションの破損につながりかねないことである。こうしたシステムのこの脆弱な面は、組織の内部および外部の他企業のどちらでも、システムの保守にかかる高額な費用、およびｅビジネスの開発においてこうしたシステムによって課せられる規制に負う部分がある。
【０００３】
サービス指向アーキテクチャ（ＳＯＡ）は、ｅビジネスを実施するための概念的アーキテクチャである。これは、前述の問題に対処する手段として提案された。基本的な考えは、一枚岩的なモノリシックなアプリケーションを「サービス」に変えるために、適切に定義されたサービス・インターフェースでアプリケーションをラップする（覆う）（ｗｒａｐ）ことである。このラッププロセスによって、アプリケーションが使用するプログラミング言語、オペレーティング・システム、ネットワーキング・プロトコル、および／またはデータベースをマスクするアプリケーションの抽象化を作り出す。したがって、サービス・インターフェース、すなわちサービスの記述が、いくつかのサービスを繋ぎ目なくシームレスに統合することができる。セキュリティ、ミドルウェア、および通信技術も、環境的な必須条件としてサービスに加えるためにラップラッパ（ｗｒａｐｐｅｒ）することができる。ラップすることラッパによって、サービスは、たとえばハイパー・テキスト転送プロトコルＨｙｐｅｒＴｅｘｔ　Ｔｒａｎｓｆｅｒ　Ｐｒｏｔｏｃｏｌ（ＨＴＴＰ：ＨyperText Transfer Protocol）を介して仮想企業をその異種システムにリンクさせること、および単一の管理ドメインに参加させることができる。
【０００４】
ここ１０年ほどの間に、ＳＯＡの提供に関して３つの大きな試みが行われてきた。ＣＯＲＢＡ、Ｊ２ＥＥ、およびＤＣＯＭは、それぞれ独自の方法でサービス指向アーキテクチャの概念をサポートしてきた。ＣＯＲＢＡ（Common Object Request Broker Architecture　Ｃｏｍｍｏｎ　Ｏｂｊｅｃｔ　Ｒｅｑｕｅｓｔ　Ｂｒｏｋｅｒ　Ａｒｃｈｉｔｅｃｔｕｒｅ）は、オブジェクト管理グループＯｂｊｅｃｔ　Ｍａｎａｇｅｍｅｎｔ　Ｇｒｏｕｐ（ＯＭＧ：Object Management Group）の後援の下に開発された。ミドルウェアである。ほぼすべてのコンピュータ、オペレーティング・システム、プログラミング言語、およびネットワーク上にあるいずれのベンダーからのＣＯＲＢＡに基づくプログラムも、ほぼすべての他のコンピュータ、オペレーティング・システム、プログラミング言語、およびネッ

トワーク上にある同じかまたは他のベンダーからのＣＯＲＢＡに基づくプログラムと、相互運用することができる。Ｊ２ＥＥ（Java2 PlatformＪａｖａ　２　Ｐｌａｔｆｏｒｍ、エンタープライズ版）は、多層構成のエンタープライズ・アプリケーションを開発するための標準を定義するものである。Ｊ２ＥＥは、エンタープライズ・アプリケーションを標準化されたモジュラー・コンポーネントに基づくものとすること、それらのコンポーネントに完全なサービス・セットを提供すること、および多くのアプリケーション動作の細部を複雑なプログラミングなしで自動的に処理することによって、エンタープライズ・アプリケーションを簡略化するものである。ＤＣＯＭ（Distributed Component Object Model Ｄｉｓｔｒｉｂｕｔｅｄ　Ｃｏｍｐｏｎｅｎｔ　Ｏｂｊｅｃｔ　Ｍｏｄｅｌ）は、コンポーネント・オブジェクト・モデルＣｏｍｐｏｎｅｎｔ　Ｏｂｊｅｃｔ　Ｍｏｄｅｌ（ＣＯＭ： Component Object Model）の拡張である。ＤＣＯＭは１９９６年に導入され、ＨＴＴＰなどのインターネット・プロトコルを含む多重ネットワーク移送を横切って使用するように設計されている。

【０００５】

ＳＯＡが受け入れられるようになるためには、アプリケーションのラップに使用される言語が汎用性のあるものでなければならない。ＤＣＯＭはＭｉｃｒｏｓｏｆｔ－ＩＤＬを使用し、ＣＯＲＢＡはＣＯＲＢＡ－ＩＤＬを使用し、Ｊ２ＥＥはＪａｖａ（登録商標）を使用する。これらのうちのいずれも、汎用言語であるとは考えられない。

【０００６】

もう１つの問題は、サービス・インターフェースに柔軟性が求められることである。サービス・インターフェースとは、経時的に進歩していくことの可能な、様々なメッセージ・タイプに応答するだけの容量を備えたものである必要がある。これが求められるのは、ソフトウェア環境が経時的に変化するものであるためである。Ｊ２ＥＥおよびＣＯＲＢＡでは、サービス・インターフェースは融通のきかないエンティティであり、あまり変化を好まない。ＤＣＯＭはサービス・インターフェースの変化をある程度考慮するが、その方法は複雑で使用するのが非常に困難である。このように柔軟なサービス・インターフェースを展開することができないため、これまでＳＯＡは配置（ｄｅｐｌｏｙ）および管理が困難であった。

【０００７】

ＳＯＡに対するこれまでのすべての手法は、第１にサービス・インターフェースを１つのソフトウェアに構築する概念、第２にそのサービス・インターフェースをインプリメント具現化するために何らかのプログラミング論理を作成する概念という、２つの明確な概念を組み合わせたものである。したがって、Ｊ２ＥＥ、ＤＣＯＭ、またはＣＯＲＢＡのいずれかを利用しようとする場合は、第１にそのプログラミング・モデルおよびそのかなりの量の複雑なＡＰＩセットを学習する必要がある。

【０００８】

上記の内容に鑑み、ＳＯＡを首尾よくインプリメントし具現化するためには、ウェブ・サービスの技術がＳＯＡにとって最良の接続技術であると理解されてきた。これは、多種多様な企業および業界組織からの研究者およびコンサルタントの業務を反映したりする技術セットの中核に依拠するものである。これらの技術には、以下のものが含まれる。

【０００９】

ＸＭＬ： ExtensibleMark-Up LanguageＥｘｔｅｎｓｉｂｌｅ　Ｍａｒｋ－ｕｐ　Ｌａｎｇｕａｇｅ（拡張可能マークアップ言語）。これは、ワールド・ワイド・ウェブ・コンソーシアムＷｏｒｌｄ　Ｗｉｄｅ　Ｗｅｂ　Ｃｏｎｓｏｒｔｉｕｍ（Ｗ３Ｃ： World Wide Web Consortium）からのテキストに基づくマークアップ言語仕様である。プレゼンテーションおよびデータを記述するためにタグを使用するＨＴＭＬとは異なり、ＸＭＬは、厳密には移植可能な構造化データを定義するためのものである。マークアップの文法または語彙、ならびに交換フォーマットおよびメッセージング・プロトコルなどの、データ記述言語を定義するための言語として使用することができる。

【００１０】

ＳＯＡＰ： Simple ObjectAccess ProtocolＳｉｍｐｌｅ　Ｏｂｊｅｃｔ　Ａｃｃｅｓｓ　Ｐｒｏｔｏｃｏｌ（シンプル・簡易オブジェクト・アクセス・プロトコル）。これは、非集中型の分散環境で情報を交換するためのＸＭＬに基づく軽量プロトコルである。ＳＯＡＰは、要求者と提供者のオブジェクト間でのメッセージング・プロトコルを定義し、要求側オブジェクトがオブジェクト指向のプログラミング様式で提供側オブジェクトでのリモート・メソッド呼び出しを実行できるようにする。ＳＯＡＰの仕様は、Ｍｉｃｒｏｓｏｆｔ、ＩＢＭ、Ｌｏｔｕｓ、ＵｓｅｒＬａｎｄ、およびＤｅｖｅｌｏｐＭｅｎｔｏｒの共著である。今やＳＯＡＰは、ほとんどのベンダーのＳＯＡのインプリメンテーション具現化において分散型オブジェクト通信の基礎を形成している。ＳＯＡはメッセージング・プロトコルを定義しないが、近年ＳＯＡＰは、ＳＯＡのインプリメンテーション具現化で一般的に使用されることから、サービス指向アーキテクチャ・プロトコルと呼ばれるようになってきた。ＳＯＡＰの一面はベンダーに中立なことであり、プラットフォーム、オペレーティング・システム、オブジェクト・モデル、およびプログラミング言語に関して独立したインプリメンテーション具現化を可能にする。さらに、移送および言語バインディングならびにデータ・コード化基本設定は、すべてインプリメンテーション具現化に依存する。

【００１１】

ＷＳＤＬ： Web ServicesDescription LanguageＷｅｂ　Ｓｅｒｖｉｃｅｓ　Ｄｅｓｃｒｉｐｔｉｏｎ　Ｌａｎｇｕａｇｅ（ウェブ・サービス記述言語）。これは、サービス・インターフェース定義言語（ＩＤＬ： Service Interface Description Language）を記述する標準的な方法を提供する、ＸＭＬ語彙である。ＷＳＤＬは、ＩＢＭが開発したネットワーク・アクセス可能サービス仕様言語Ｎｅｔｗｏｒｋ　Ａｃｃｅｓｓｉｂｌｅ　Ｓｅｒｖｉｃｅ　Ｓｐｅｃｉｆｉｃａｔｉｏｎ　Ｌａｎｇｕａｇｅ（ＮＡＳＳＬ： Network Accessible Service Specification Language／ネットワーク・アクセス可能サービス仕様言語）と、Ｍｉｃｒｏｓｏｆｔが開発した仕様及び記述言語Ｓｐｅｃｉｆｉｃａｔｉｏｎ　ａｎｄ　Ｄｅｓｃｒｉｐｔｉｏｎ　Ｌａｎｇｕａｇｅ（ＳＤＬ：Specification andDescription Language／仕様記述言語）との間にアクティビティが集中する結果として生じる人工的産物である。これは、リモート・メソッド呼び出し（ＲＭＩ）のために、サービス提供者が要求および応答メッセージの書式を記述する簡単な方法を提供するものである。ＷＳＤＬは、基礎となるプロトコルおよびコード化要件とは無関係に、このサービスＩＤＬの原則に対処する。一般に、ＷＳＤＬは、それぞれのパラメータおよびデータ型を使用してサービスの公表済みオペレーションを定義するための抽象言語を提供する。この言語は、サービスの場所の定義およびバインディングの細部にも対処する。

【００１２】

ＵＤＤＩ： UniversalDescription, Discovery, and Integration（ユニバーサル記述、発見、および統合）Ｕｎｉｖｅｒｓａｌ　Ｄｅｓｃｒｉｐｔｉｏｎ，Ｄｉｓｃｏｖｅｒｙ，ａｎｄ　Ｉｎｔｅｇｒａｔｉｏｎ仕様は、サービス・ブローカのインプリメンテーション具現化を可能にするＳＯＡＰ　ＡＰＩの一般セットを提供する。ＵＤＤＩ仕様は、ウェブに基づくサービスの作成、記述、発見、および統合を実行しやすくするように、ＩＢＭ、Ｍｉｃｒｏｓｏｆｔ、およびＡｒｉｂａによってその概略が定められた。７０を超える業界および企業のリーダ達が提携および協力するUDDI.orgＵＤＤＩ．ｏｒｇの背後にあるモチベーションは、企業間（Ｂ２Ｂ： Business to Business）相互運用性のための標準を定義することであるが、これは企業・消費者間（Ｂ２Ｃ： Business to Customer）アプリケーションに採用することもできる。

【００１３】

これらの技術は、ＳＯＡでウェブ・タイプのサービスを実施するための、包括的かつ独占的な技術セットを有するものでないことを理解されよう。しかしながら、ウェブ・タイプのサービスが、形式および機能を記述するために使用可能なある種のＸＭＬに基づく記述メカニズムを必要とすることが一般に受け入れられているため、これらの技術の中でもＸＭＬは不可欠である。

【００１４】
ＳＯＡは、企業内および企業間のコンピューティングのための優れた基盤を提供する。これによって、業務プロセスを具体化するアプリケーションをウェブ・タイプのサービスとして迅速かつ効率良く実施すること、ならびにこうしたサービスの実施を管理する際に人間が関与する割合を減らすことが可能となる。多くの実施には、いかなる人間の関与も必要としないマシン間でのサービス呼び出しが含まれることになる。
【００１５】
企業がＳＯＡを導入しようとする意欲は、結果として生じる恩恵に関係する。ＳＯＡを介して業務プロセスを実施することで、ビジネス・パートナーとの連携を改善し、競争相手のそれよりも優れた連携の拠点（ｈｕｂ）を企業間に生み出し、市場で勝ち残る能力を向上させる可能性がある。こうした連携の拠点は、たとえば供給チェーンを圧縮することによって市場に出すまでの時間を改善することもできる。結果として、今日の多くの企業は、外部サービスの方が自社の業務の産出および競争力により多くの影響を与えられるため、内部サービスよりも外部サービスの設計に焦点を当てている。
【００１６】
ウェブに基づくサービスの品質は、セキュリティ、サービスの公表、説明責任、追跡可能性、性能、およびサービス・インターフェース定義を含む、多くの要素に依存している。
【００１７】
本発明は、サービス・インターフェースの設計を対象とするものである。良いサービス・インターフェースを設計するために、サービス設計者は定義されたサービスの粒度（細かさ）を注意深く考慮しなければならない。サービスは、一方で性能およびカプセル化の理由から粒度を粗くしなければならないが、他方では、最高の柔軟性にするために粒度を細かくしなければならない。サービスの粒度が粗すぎる場合、「クライアント」が要求するよりも多くのデータがネットワークを横切って送られるが、サービスの粒度が細かすぎると、「クライアント」は必要なすべてのデータを取得するのに複数のネットワークを経由トリップしなければならない。
【００１８】
現在、サービス・インターフェースは、ウェブ・サービスに基づく業務プロセスの動的なオンデマンド合成を可能にするＩＢＭのＡｌｐｈａｗｏｒｋｓウェブ・サービシス・アウトソーシング・マネージャ（Alphaworks Web Services Outsourcing Manager）Ａｌｐｈａｗｏｒｋｓ Ｗｅｂ Ｓｅｒｖｉｃｅｓ Ｏｕｔｓｏｕｒｃｉｎｇ Ｍａｎａｇｅｒ、およびオープンＪ２ＥＥアーキテクチャをサポートする製品が提供する任意のサービスと業務プロセスとを組み合わせることのできるＩＢＭのＷｅｂｓｐｈｅｒｅアプリケーション・サーバ・エンタープライズ・プロセス・コレオグラファ（Websphere Application Server Enterprise ProcessChoreographer）Ｗｅｂｓｐｈｅｒｅ Ａｐｐｌｉｃａｔｉｏｎ Ｓｅｒｖｅｒ Ｅｎｔｅｒｐｒｉｓｅ Ｐｒｏｃｅｓｓ Ｃｈｏｒｅｏｇｒａｐｈｅｒなどの、リアルタイム・ツールを使用して、人間のオペレータ（設計者）によってリアルタイムで設計される。しかし、現在のところ、「クライアント」がこのようにサービスを呼び出した結果によるサービスの使用を考慮に入れて、サービス・インターフェースを自動的に生成または修正するための手段はない。
【特許文献１】米国特許第５６６１６６８号
【特許文献２】米国特許第６２４９７５５号
【特許文献３】米国特許第６５１６２８８号
【発明の開示】
【発明が解決しようとする課題】
【００１９】
本発明の目的は、既存の配置構成の欠点を除去または緩和する方法およびシステムを提供することである。
【００２０】

本発明の具体的な目的は、前記サービスの使用に応答して、ＳＯＡでインプリメント具現化されるウェブ・サービス向けのサービス・インターフェースを自動的に生成および／または修正するための方法およびシステムを提供することである。

【課題を解決するための手段】

【００２１】

第１の態様では、本発明は、マシン実施可能アプリケーション向けのサービス・インターフェースを自動的に生成する方法を提供するものであって、当該方法は、前記アプリケーションに関連付けられたそれぞれのサービスを定義する複数のサービス・インターフェースを配置するステップと、前記サービス・インターフェースによって定義されたサービスの使用を記録するデータをログ記録するステップと、前記使用データから、繰り返されるサービス使用パターンを識別するステップと、前記識別された繰り返されるサービス使用パターンに応答して、新しいサービス・インターフェースを自動的に生成するステップと、を含むことを特徴とする。

【００２２】

マシン実施可能アプリケーションは、インターネットなどのネットワーク環境では、前記アプリケーションを含む機能を定義する複数のサービス・インターフェースによって表される。これにより、ネットワークに接続されたユーザ（クライアント）は、前記サービス・インターフェースによって定義されたサービスを呼び出すことができる。本発明のこの態様は、人間のオペレータ（設計者）がリアルタイム・ツールを使用してサービス・インターフェースを手動で設計変更する必要なしに、サービスの使用に基づいてサービスの粒度を改善するという問題に対処するものである。実際、本発明は、クライアントの使用パターンに基づいてサービス・インターフェースを自動的に最適化するための実行時のランタイム方法を提供する。

【００２３】

新しいサービス・インターフェースを生成するステップは、１つまたは複数の既存のサービス・インターフェースを修正するステップを含むことができる。

【００２４】

新しいサービス・インターフェースの生成は、既存のサービス・インターフェースの更新または修正を含むことができることを理解されよう。したがって、この状況での「新しい」という用語は、この用語がすでに存在するインターフェースから新しいサービス・インターフェースを形成することを除外しているという趣旨で、本発明の範囲を限定するものとみなされるべきではない。

【００２５】

当該またはそれぞれのサービス・インターフェースは、前記マシン実施可能アプリケーションを含む１つまたは複数の機能を具体化することができる。

【００２６】

サービスは、業務プロセスを実施するためのデータの交換またはアプリケーション・コンポーネント（機能）の論理グループ化などの単純なアプリケーション機能を含むことができる。

【００２７】

好ましくは、新しいサービス・インターフェースを生成するステップは、いくつかの既存のサービス・インターフェースの機能を組み合わせることを含む。

【００２８】

これは、初期にサービスは、最大の柔軟性を提供するために粒度が細かくなるように設計されるという前提に基づくものである。その後、サービス使用パターンによって、クライアントが通常パターンのネットワーク訪問でサービス・グループを体系的に呼び出すことが明らかになる場合がある。こうした繰り返される使用パターンに応答して、本発明の方法は、サービス・グループを含む新しいサービス・インターフェースが自動的に定義および配置され、その結果、新しいサービス・インターフェースの粒度がそのように形成され、これを呼び出すために１回のネットワーク訪問でよくなることを提案する。

【００２９】
サービスの使用を記録するデータをログ記録するステップには、サービス呼び出し署名を含むデータをサービス使用履歴データベースに格納することが含まれる場合がある。
【００３０】
好ましくは、当該サービス呼び出し署名は、サービスが実行される前および／または後にログ記録される。
【００３１】
さらに好ましくは、当該サービス呼び出し署名は、呼び出し側コンテキスト、タイムスタンプ、および呼び出しパラメータのうちの任意の１つまたはそれらの組み合わせに関するデータと共に、サービス使用履歴データベースに格納される。
【００３２】
呼び出し側コンテキストデータは、呼び出し者識別子および呼び出し側セッション識別子を含むことができる。
【００３３】
発呼呼び出し者ＩＤ、呼び出し側セッションＩＤ、およびタイムスタンプを表すデータの格納および分析によって、サービス・インターフェースを最適化するための触媒の働きをするもの（ｃａｔａｌｙｓｔ）である繰り返されるサービス使用パターンの識別が、比較的簡単な処理タスクとなる。
【００３４】
好ましくは、繰り返されるサービス使用パターンを前記使用データから識別するステップは、サービス使用パターンを選別して、対応する細粒度サービスの対応するセットを形成するためにサービス使用パターンを見分けることを含む。
【００３５】
細粒度サービスは、最適化済み最適化粗粒度サービス・インターフェースを生成するための構築ブロックを形成する。
【００３６】
好ましくは、新しいサービス・インターフェースを生成するステップは、サービス使用パターンから選別された見分けられたいくつかの細粒度サービス・インターフェースを組み合わせることによって、新しい粗粒度サービス・インターフェースを生成することを含む。
【００３７】
新しい粗粒度サービス・インターフェースは、自動的に配置することができる。
【００３８】
別の方法として、新しい粗粒度サービス・インターフェースは、配置される前に、「予定の（Ｔｏ　ｂｅ）」粗粒度サービス定義データベースに格納される。
【００３９】
本発明は、最適化済み最適化サービス・インターフェースを自動的に生成することを対象としているが、本発明の方法には、適切であれば新しく生成されたサービス・インターフェースの配置に人間が関与する機会が含まれることになると予想される。
【００４０】
新しいサービス・インターフェースを自動的に生成するステップは、人間のオペレータ、システム管理者、およびスケジュールリング・エンジンのうちのいずれか１つによってトリガすることができる。
【００４１】
別の方法として、新しいサービス・インターフェースを自動的に生成するステップは、繰り返されるサービス使用パターンによって自動的にトリガされる。
【００４２】
好ましくは、新しいサービス・インターフェースは、サービス指向アーキテクチャ（ＳＯＡ）に配置するために自動的に生成される。
【００４３】

他の態様では、本発明は、マシン実施可能アプリケーション向けのサービス・インターフェースを自動的に生成するためのシステムを提供するものであって、当該システムは、前記アプリケーションに関連付けられたそれぞれのサービスを定義する複数のサービス・インターフェースを配置するための手段と、前記サービス・インターフェースによって定義されたサービスの使用を記録するデータをログ記録するための手段と、前記使用データから、繰り返されるサービス使用パターンを識別するための手段と、前記識別された繰り返されるサービス使用パターンに応答して、新しいサービス・インターフェースを自動的に生成するための手段と、を含むことを特徴とする。

【００４４】

好ましくは、複数のサービス・インターフェースを配置するための手段は、ＩＢＭ　Ｗｅｂｓｐｈｅｒｅアプリケーション・サーバ　Ａｐｐｌｉｃａｔｉｏｎ　Ｓｅｒｖｅｒなどのサービス実行エンジンを含む。

【００４５】

さらに好ましくは、サービス実行エンジンはＪ２ＥＥに準拠したアプリケーション・サーバを含む。

【００４６】

サービス実行エンジンは、サービスの使用を記録するデータをログ記録するための手段も含み、ログ記録されたデータを格納するためのデータベースを含むことができる。

【００４７】

前記使用データから、繰り返されるサービス使用パターンを識別するための手段、および／または、前記識別された繰り返されるサービス使用パターンに応答して、新しいサービス・インターフェースを自動的に生成するための手段は、自律型自己最適化サービス定義コンポーネントを含むことができる。

【００４８】

自律型自己最適化サービス定義コンポーネントは、前記使用データから、繰り返されるサービス使用パターンを識別するための、サービス要求類似性分析部を含むことができる。

【００４９】

自律型自己最適化サービス定義コンポーネントは、前記識別された繰り返されるサービス使用パターンに応答して新しいサービス・インターフェースを自動的に生成するための、サービス・インターフェース生成部を含むことができる。

【００５０】

好ましくは、自律型自己最適化サービス定義コンポーネントは、使用可能な細粒度サービス・インターフェースのためのデータベースと、既存の粗粒度サービス・インターフェースのためのデータベースのうちの、１つまたは両方を含む。

【００５１】

さらに好ましくは、自律型自己最適化サービス定義コンポーネントは、推論エンジンと、前記サービス・インターフェース生成部を制御するための規則データベースとを含む。

【００５２】

本発明の他の態様では、サービス指向アーキテクチャにはマシン実施可能アプリケーション向けのサービス・インターフェースを自動的に生成するためのサービス・インターフェース生成コンポーネントがあり、これは、サービス使用データから、繰り返されるサービス使用パターンを識別するための手段と、前記識別された繰り返されるサービス使用パターンに応答して新しいサービス・インターフェースを自動的に生成するための手段と、を含むことを特徴とするものである。

【００５３】

好ましくは、サービス使用データから、繰り返されるサービス使用パターンを識別するための前記手段は、サービス使用パターンを選別して、対応する細粒度サービスおよび／または粗粒度サービスのセットを形成するために、サービス使用パターンを見分けるための手段を含む。

【００５４】
好ましくは、前記識別された繰り返されるサービス使用パターンに応答して新しいサービス・インターフェースを自動的に生成するための手段は、サービス使用パターンから選別された見分けられたいくつかの細粒度サービス・インターフェースおよび／または粗粒度サービス・インターフェースを組み合わせることによって、新しい粗粒度サービス・インターフェースを生成する。
【００５５】
本発明の他の態様では、前述の方法で使用するためのサービス実行エンジンが提供される。
【００５６】
本発明の他の態様では、前述の方法を実施するためのコンピュータ・プログラムが提供される。
【００５７】
本発明の他の態様では、前述の方法のコンピュータによるインプリメンテーション具現化が提供される。
【００５８】
本発明の前述および他の特徴は、図面を参照しながら、例として示したものに過ぎず、本発明の範囲を限定することのない以下の好ましい実施形態の説明を読めば、より容易に理解されよう。
【発明を実施するための最良の形態】
【００５９】
図１を参照すると、全体として１０で示された本発明のシステムは、破線１２によって示されたネットワーク・インターフェースのサービス・プロバイダ（発行者）側に、クライアントがインターネットなどのウェブ実行可能ネットワーク（図示せず）を介して配置されたサービスを呼び出すことができるようにする、サービス実行エンジン１４を含む。配置されたサービスは、配置されたサービス・データベース１６に格納される。サービスを定義するサービス・インターフェースは、それらの間に１つまたは複数のサービス・プロバイダ業務プロセスまたはアプリケーションを含むことができる。実行エンジン１４は、たとえばＩＢＭ Ｗｅｂｓｐｈｅｒｅアプリケーション・サーバ Ａｐｐｌｉｃａｔｉｏｎ Ｓｅｒｖｅｒまたは任意のＪ２ＥＥ準拠サーバを含むことができる。さらにネットワーク・インターフェース１２のサービス・プロバイダ側には、配置されたサービスを管理するためのサービス管理者１８も置くことができる。サービス管理者１８は人間のオペレータも可能であるが、インテリジェント・ソフトウェア・エージェントでもこの役割を果たすことができる。配置されることになる新しいサービスを設計するための設計者２０には、ウェブ・サービスに基づく業務プロセスの動的なオンデマンド合成を可能にするＩＢＭのＡｌｐｈａｗｏｒｋｓウェブ・サービシス・アウトソーシング・マネージャ Ｗｅｂ Ｓｅｒｖｉｃｅｓ Ｏｕｔｓｏｕｒｃｉｎｇ Ｍａｎａｇｅｒ、およびオープンＪ２ＥＥアーキテクチャをサポートする製品によって提供される任意のサービスと業務プロセスとを組み合わせることのできるＩＢＭのＷｅｂｓｐｈｅｒｅアプリケーション・サーバ・エンタープライズ・プロセス・コレオグラファ Ａｐｐｌｉｃａｔｉｏｎ Ｓｅｒｖｅｒ Ｅｎｔｅｒｐｒｉｓｅ Ｐｒｏｃｅｓｓ Ｃｈｏｒｅｏｇｒａｐｈｅｒなどの、リアルタイム設計ツールを採用する、知られたリアルタイム技法に従って新しいサービスを設計する人間のオペレータが含まれる。ネットワーク・インターフェース１２のクライアント側には、既存のサービス定義を使用して新しいアプリケーションを開発するための、クライアント・アプリケーション・プログラマ２２を置くことができる。サービス・プロバイダの設計者２０の場合と同様に、これにはリアルタイム・モードで働く人間のオペレータが含まれる。したがって、これまでに述べた配置構成に新たなものはなく、こうした配置構成は当業者には周知であることを理解されよう。
【００６０】
本発明に従ったシステムには、自律型自己最適化サービス定義コンポーネントとも呼ば

れるサービス・インターフェース生成コンポーネント２４が含まれる。このコンポーネント２４の目的は、識別された繰り返されるクライアント使用パターンに基づいて新しいサービス・インターフェースを自動的に生成することである。したがって以下の説明からより明らかになるように、このコンポーネントはシステムの他のコンポーネントと協力して、新しいサービス・インターフェースをランタイム・モードで自動的に生成および配置することができる。

【００６１】

図２では、サービス・インターフェース生成コンポーネント２４およびシステムの他のコンポーネントとの対話の構造も見ることができる。サービス・インターフェース生成コンポーネント２４は、サービス要求類似性分析部２６、推論エンジン２８、規則データベース３０、および最適化サービス生成部３２を含む。レポート生成部３４を含むこともできる。サービス・インターフェース生成コンポーネント２４のコンポーネントは、規則データベース３４を除いて、すべてがソフトウェア・コンポーネントとしてインプリメントし具現化される。サービス要求類似性分析部２６はウェブ・サイト分析部と等価であるが、ウェブ・サイトの使用統計量（ウェブ・サイト・ログ）を処理しないという点が異なる。サービス要求類似性分析部２６は、サービス実行エンジン１４によって呼び出されたウェブ・サービスの使用統計量を分析する。推論エンジン２８は、Ｖｅｒｓａｔａロジック・スイート（Versata Logic Suite）Ｌｏｇｉｃ Ｓｕｉｔｅの一部を形成するＶｅｒｓａｔａ論理エンジン、またはＩＢＭのＷｅｂｓｐｈｅｒｅアプリケーション・サーバ Ａｐｐｌｉｃａｔｉｏｎ Ｓｅｒｖｅｒの一部を形成するＩＢＭのＷｅｂｓｐｈｅｒｅビジネス・ルールズ・ビーン（Websphere Business Rules Bean）ｂｕｓｉｎｅｓｓ ｒｕｌｅｓ ｂｅａｎなどの、市販の推論エンジン（規則に基づくエンジン）を使用してインプリメントし具現化することができる。最適化サービス生成部３２には、典型的にはＷＤＳＬのようなサービス定義を生成するための推論エンジン２８処理の結果を取るコード生成部・ジェネレータが含まれる。レポート生成部３４は、推論エンジン２８処理の結果を人間が理解可能な形で生成する。これらのレポートは、「予定の」最適化粗粒度サービス定義データベース４４のコンテンツへのリンクを有する。「予定の」粗粒度サービス定義データベース４４は、分析されたサービス使用パターンに応答して生成される新しいサービス・インターフェースを含む。たとえばサービス使用レポートは、特定のサービスによって公開される演算ＯｘおよびＯｙが、これらの演算の９０％のケースがＯｘ＋ｙのシーケンスで呼び出されることから、１つの演算Ｏｘ＋ｙにまとめられることを示す場合がある。

【００６２】

「予定の」粗粒度サービス定義データベース４４に加えて、サービス・インターフェース生成コンポーネント２４は、累積サービス使用履歴データベース３６、「現状のままの」粗粒度サービス定義データベース４０、使用可能な細粒度エンタープライズ・サービスデータベース４２、およびレポート４６にリンクされる。

【００６３】

サービス定義データベース３６、４０、４２、および４４はそれぞれ、リレーショナル・データベース、ＸＭＬフラット・ファイル、およびレポート用のＰｌｕｓ Ｗｏｒｄ処理を使用してインプリメント具現化可能な、構造化データ・ストアを含む。

【００６４】

累積サービス使用履歴データベース３６は、ウェブ・サイト・ログと等価であるが、本発明では、これらには代わりにウェブ・サービス・ログが含まれる。これらは、プロセスに何の知的性インテリジェンスも追加することなく、ウェブ・サービス使用に関して生じたことを記録するだけであるという意味において、低レベルのログである。

【００６５】

サービス・インターフェース生成コンポーネント２４のサービス要求類似性アナライザ分析部２６は、１つの入力で、サービス実行エンジン１４のサービス使用ログ記録機能３８によってログ記録されたサービス使用データを格納する、累積サービス使用履歴データ

ベース３６に接続される。累積サービス使用履歴データベース３６に格納された使用データは、本発明に従って新しいサービス・インターフェースを自動的に生成する方法の１ステップとして、繰り返されるクライアント使用パターンを識別するために分析される。他の入力として、サービス要求類似性分析部２６が「現状のままの」粗粒度サービス定義データベース４０に接続される。「現状のままの」粗粒度サービス定義データベース４０は、間接的に推論エンジンの入力である。「現状のままの」粗粒度サービス定義データベース４０は、図１では別々のデータベースとして示されているが、配置されたサービス・データベース１６（図）と同じ１つのデータベースである。

【００６６】

サービス要求類似性分析部２６は、サービス・インターフェースでどの演算が使用されたか、および使用されなかったか、どの演算が（推論エンジンがより大きな粗粒度サービスを提案するのを助けるために）相互に相関付けられているか、所与の演算に使用されるパラメータの数（推論エンジンが粗粒度サービスのサイズを減じることができるようにするため）、などに関する統計量を、推論エンジン２８に提供する。したがって、サービス要求類似性分析部２６の出力は、「現状のままの」粗粒度サービス定義データベース４０のスーパーセットとみなすことができる。

【００６７】

推論エンジン２８は、他の入力として、使用可能な細粒度エンタープライズ・サービス・データベース４２への接続を有する。細粒度エンタープライズ・サービスは、配置されたサービス・データベース１６、または／「現状のままの」サービス・データベース１６／４０でウェブ・タイプ・サービスとして使用できるようになっている業務アプリケーション／プロセスを含む方法／サービス要素を定義する、基本的なリアルタイム設計のサービス・インターフェースを含む。

【００６８】

推論エンジン２８への最終入力は、推論エンジン２８のオペレーションを規定する規則を定義する、規則データベース３０への接続である。規則データベース３０は、リレーショナル・データベース、ＸＭＬフラット・ファイル、およびレポート用のＰｌｕｓ　Ｗｏｒｄ処理を使用してインプリメント具現化可能な、構造化データ・ストアを含む。たとえこの発明をインプリメント具現化するシステムに事前にパッケージされた規則セットが付属している場合であっても、この規則はある業務に極めて特有のものとすることもできる。事前にパッケージされた規則は、以下のような標準サービスの再度の再因子因数分解（ｒｅｆａｃｔｏｒｉｚａｔｉｏｎ）規則である場合がある。

・サービス演算ＯｘおよびＯｙが、９０％のケースでユーザの９０％によって２秒以内に順序どおり使用される場合、新しいサービス演算Ｏｘ＋ｙを提案する。

・サービス演算Ｏｘでパラメータ Ｐｘが９０％のユーザによって使用されない場合、このパラメータなしで新しいサービス演算を提案する。

業務特有の規則は、たとえば以下のような場合がある。

・マーケティング・キャンペーン開始後３ヶ月の間に、サービス「Ｃａｒ　Ｇｏｌｄ　Ｗａｒｒａｎｔｙ（車両ゴールド保証）」と「Ｃａｒ　Ｓｐｅｃｉａｌ　Ｉｎｓｕｒａｎｃｅ（車両特別保険）」との間に、相関関係を見つける。

【００６９】

推論エンジン出力は、提案された（または「予定の」）粗粒度サービス定義を「予定の」粗粒度サービス定義データベース４４に出力する、最適化サービス生成部３２の入力と接続される。推論エンジンは、レポートを生成するためにレポート生成部３４に接続することもできる。

【００７０】

図３に示されるように、本発明の方法は、ＩＢＭのＡｌｐｈａｗｏｒｋｓウェブ・サービシス・アウトソーシング・マネージャ　Ｗｅｂ　Ｓｅｒｖｉｃｅｓ　Ｏｕｔｓｏｕｒｃｉｎｇ　Ｍａｎａｇｅｒなどのリアルタイム・ツールをリアルタイムで使用してサービス・インターフェースを手動で設計する、知られた方法の上に構築され、クライアントのサ

ービス使用のログ記録の第１の方法部分５０と、使用データの分析、および前記使用データから識別された繰り返される使用パターンに応答した新しいサービス・インターフェース（定義）の自動生成の第２の方法部分５２とを含む。サービス・インターフェース生成コンポーネント２４によって自動的に生成された新しいサービス定義を配置するか、または別の方法として、新しいサービス・インターフェースが追加の手動設計を必要とすると決定（図３の決定ポイント５４）し、配置されたサービス・データベース１６／「現状のままの」データベース１６／４０に転送される前に手動で処理するために「予定の」粗粒度サービス定義データベース４４に向けて送ることができる。この決定ポイントは常に手動である。

【００７１】

図４を参照すると、第１の方法部分５０には、サービス実行エンジン１４によって実行されるサービス呼び出し検出ステップ５６が含まれる。サービス呼び出し検出は、ＩＢＭのＷｅｂｓｐｈｅｒｅアプリケーション・サーバ Ａｐｐｌｉｃａｔｉｏｎ Ｓｅｒｖｅｒなどのサービス実行エンジンの既存の機能である。第１の方法部分５０の第２のステップ５８として、サービス呼び出しデータはログ記録され、サービス使用履歴データベース３６に格納される。呼び出しデータには、発呼呼び出し者識別子（たとえば<bank account number（銀行口座番号）>、<warrantynumber（保証番号）>、<user id（ユーザＩＤ）+ purchase order id（購入注文ＩＤ）>）、呼び出し側セッション識別子（たとえば、所与のユーザ（ユーザＩＤによって識別される）によって購入される本などについてのすべてのオペレーションに関する<book serial number（本の通し番号）>、または何人かのユーザ（ユーザＩＤによって識別される）間での所与の主題に関する考察にまつわるすべてのオペレーションなどに関する<discussion thread id（協議考察スレッドＩＤ）>）、呼び出し前タイムスタンプ、および呼び出し後タイムスタンプ、のうちの、１つまたは任意の組み合わせを含む、いくつかのデータ要素（パラメータ）が含まれる場合がある。呼び出し入力パラメータはサービス呼び出し者によって設定されるものであるのに対して、出力パラメータはサービス・サーバによって返されるものである。たとえばウェブ・サービス定義言語Ｗｅｂ Ｓｅｒｖｉｃｅ Ｄｅｆｉｎｉｔｉｏｎ Ｌａｎｇｕａｇｅ（ＷＳＤＬ：Web Service Definition Language）は、特定のオペレーションに関する入力および出力パラメータを明示的に定義する。この点に関して、「requestServiceDownload」と呼ばれるオペレーションに関するＪａｖａソース・コードの一例を以下に示す。

【数１】

```
<wsdl:operation name="requestServiceDownload">
                <wsdl:input                 message="intf:requestServiceDownloadRequest"
name="requestServiceDownloadRequest" />
              <wsdl:output                 message="intf:requestServiceDownloadResponse"
name="requestServiceDownloadResponse" />
</wsdl:operation>
<wsdl:message name="requestServiceDownloadRequest">
  <wsdl:part element="intf:requestServiceDownload" name="parameters" />
  </wsdl:message>
<wsdl:message name="requestServiceDownloadResponse">
  <wsdl:part element="intf:requestServiceDownloadResponse" name="parameters" />
  </wsdl:message>
```

【００７２】

サービス呼び出しデータを取り込む（ログ記録および格納する）ための具体的な知られた配置構成の１つが、図５に示される。これは、Ａｐａｃｈｅ Ａｘｉｓエンジンを利用するものである。サービス呼び出しデータ取り込みメカニズムは、１つのＡｘｉｓ要求ハンドラおよび１つのＡｘｉｓ応答ハンドラからなる。どちらも知られた方法で、呼び出し側コンテキスト全体を取り込んで累積使用履歴データベース３６にログ記録する。図６は、Ａｐａｃｈｅ Ａｘｉｓエンジンを使用してこの呼び出しデータ取り込み方法を実施するための、Ｊａｖａソース・コードの一例を提供するものである。

【００７３】

本発明の第２の方法部分５２は、図７に概略的に示されている。これには、たとえばスケジュールリング・エンジン（図示せず）またはサービス管理者１８によるサービス・インターフェース生成コンポーネント２４のトリガの、ステップ７０で始まる一連のステップが含まれる。別の方法としては、矛盾が識別された場合にサービス・インターフェース生成コンポーネント２４を自動的にトリガすることができる。矛盾は、呼び出されているサービスが、サービス設計者の予想した方法で使用されているかどうかを判別するプロセスを介して識別することができる。これは、サービスが実際に使用されている方法と予想された使用方法とを比較することによって判別することができる。この比較は、「現状のままの」粗粒度サービス定義データベース４０および「予定の」粗粒度サービス定義データベース４４の、監視プロセスの結果として生じる。矛盾は、前記データベースそれぞれに含まれるサービス・インターフェース（定義）の数、およびそれらサービス・インターフェースのコンテンツを比較することによって発見される。この監視プロセスによって、２つのデータベースのコンテンツがもはや同一ではない、またはほとんど同一ではないことが明らかになった場合、これは矛盾であると認識され、サービス・インターフェース生成コンポーネント２４をトリガするために使用される。２つのデータベース４０、４４のコンテンツの監視プロセスは、サービス実行エンジン１４によって実行することができる。

【００７４】

次のステップ７２では、格納されたサービス使用データがサービス要求類似性分析部２６によって読み取られ、これがステップ７４で前記データを分析し、繰り返されるサービス使用パターンを識別および判別する。繰り返される使用パターンを識別するステップは、一般に、米国特許第５６６１６６８号、米国特許第６２４９７５５号、または米国特許第６５１６２８８号で教示された方式のいずれか１つに従って実施することができる。

【００７５】

識別された繰り返される使用パターンそれぞれについて（ステップ７６）、推論エンジン２８は、サービス使用パターンを選別して、前記使用パターンを含む対応する細粒度サービスを識別するために、サービス使用パターンを見分ける（ステップ７６ａ）。これらは、細粒度サービス定義の振り付け（ｃｈｏｒｅｏｇｒａｐｈｉｎｇ）によって、最適化を介して粗粒度サービス定義を生成するために、推論エンジン２８によりる呼び出し側コンテキストにおける非同一なものの間の（ｄｉｓｊｏｉｎｔ）を相関をとって、細粒度サービス定義の振り付けによって、最適化を介して粗粒度サービス定義を生成することによって、細粒度サービス呼び出しのグラフとして再編成される（ステップ７６ｂ）。使用パターンをグラフとして有する細粒度サービスの編成は、ウェブ・サービス用のビジネス・プロセス実行言語Ｂｕｓｉｎｅｓｓ　Ｐｒｏｃｅｓｓ　Ｅｘｅｃｕｔｉｏｎ　Ｌａｎｇｕａｇｅで記述することができる。以下のような２種類のサービス使用パターンがある。

・ 同じ呼び出し側コンテキストを共有するいくつかのサービス呼び出しを識別するパターンであって、結果としてさらに大きな粗粒度サービスを作成する可能性を生み出す。

・ 部分的なサービス・インターフェース使用を識別するパターンであって、結果として単一の細粒度サービスに戻る際の制限によって、粗粒度サービスの機能を低下させる可能性を生み出す。

【００７６】

新しい粗粒度サービス定義（インターフェース）は、「予定の」粗粒度サービス定義データベース４４に格納され（ステップ７６ｃ）、ここで、その後、新しいサービス定義を配置するために配置済みのサービス・データベース１６に転送するか、または、配置できるようにするために何らかの追加の手動設計が必要であるかどうかが決定される（第１の方法部分５０、決定ポイント５４）。最終ステップ７８は、サービス管理者１８および／またはサービス設計者２０が再検討できるレポートを生成するためのものである場合がある。

【００７７】

本発明は、新しいサービス定義を自動的に生成するためにログ記録されたサービス使用

データを使用するものであって、前記新しいサービス定義は１つまたは複数の使用可能な細粒度サービス定義を含む。

【図面の簡単な説明】

【００７８】

【図１】本発明に従ってサービス・インターフェースを自動的に生成するためのシステムを示す概略ブロック図である。

【図２】図１のシステムのサービス・インターフェース生成コンポーネントを示す概略ブロック図である。

【図３】本発明に従った方法を示す流れ図である。

【図４】本発明に従った方法の第１の部分を含むステップを示す概略ブロック図である。

【図５】本発明に従った方法の第１の部分のサービス呼び出しデータ取り込みステップの一実施を、より詳細に示す概略ブロック図である。

【図６】図５のサービス呼び出しデータ取り込みステップを実施するためのＪａｖａソース・コードの例を含む図である。

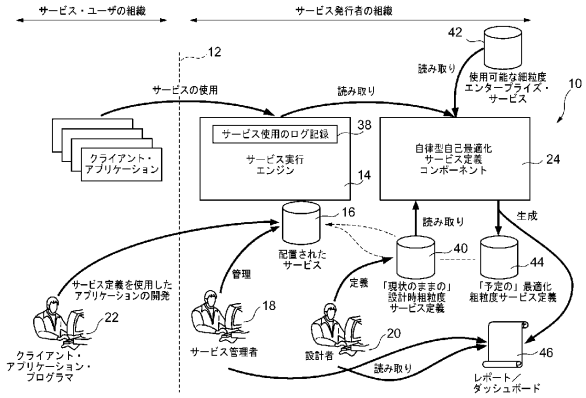
【図７】本発明に従った方法の第２の部分を含むステップを示す概略ブロック図である。

【符号の説明】

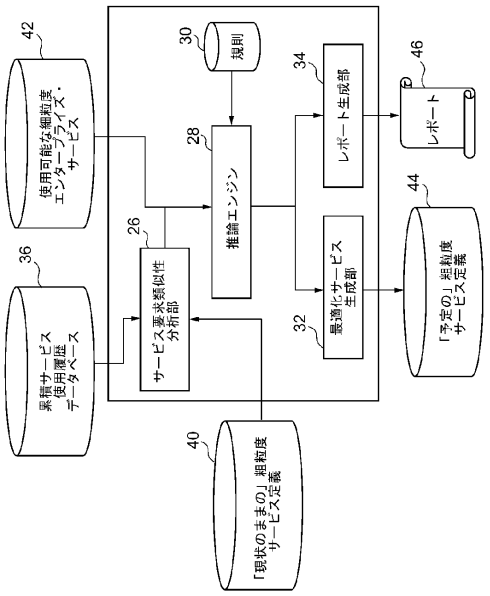
【００７９】

２６　サービス要求類似性分析部
２８　推論エンジン
３０　規則
３２　最適化サービス生成部
３４　レポート生成部
３６　累積サービス使用履歴データベース
４０　「現状のままの」粗粒度サービス定義データベース
４２　使用可能な細粒度エンタープライズ・サービス定義データベース
４４　「予定の」粗粒度サービス定義データベース
４６　レポート

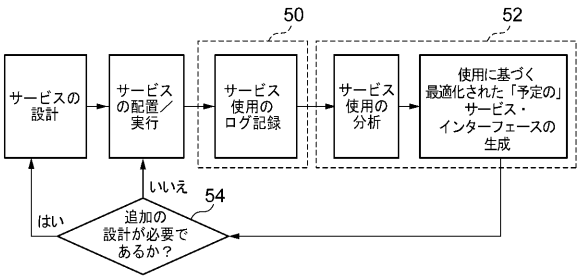
【図１】


サービス・ユーザの組織
サービス発行者の組織
12
42
読み取り
使用可能な細粒度
エンタープライズ・
サービス
10
サービスの使用
読み取り
クライアント・
アプリケーション
サービス使用のログ記録
38
サービス実行
エンジン
14
自律型自己最適化
サービス定義
コンポーネント
24
16
読み取り
生成
配置された
サービス
40
44
管理
定義
「現状のままの」
設計時粗粒度
サービス定義
「予定の」最適化
粗粒度サービス定義
サービス定義を使用した
アプリケーションの開発
22
18
20
46
クライアント・
アプリケーション・
プログラマ
サービス管理者
設計者
読み取り
レポート／
ダッシュボード

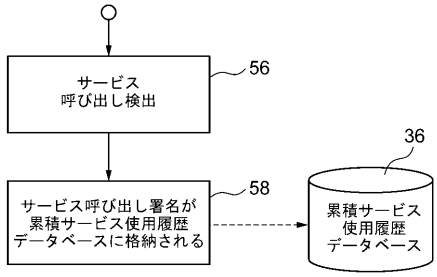
【図２】


42
使用可能な細粒度
エンタープライズ・
サービス
30
規則
34
レポート生成部
46
レポート
28
推論エンジン
44
「予定の」粗粒度
サービス定義
26
サービス要求類似性
分析部
36
累積サービス
使用履歴
データベース
32
最適化サービス
生成部
40
「現状のままの」粗粒度
サービス定義

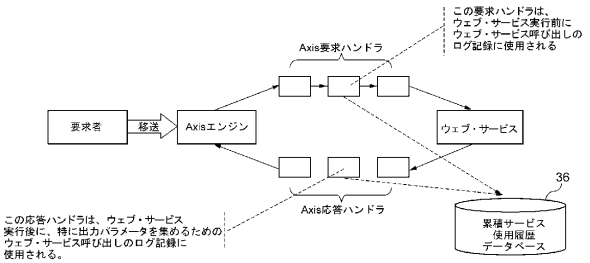
【図３】


50
52
サービスの
設計
サービス
の配置／
実行
サービス
使用の
ログ記録
サービス
使用の
分析
使用に基づく
最適化された「予定の」
サービス・
インターフェースの
生成
いいえ
54
はい
追加の
設計が必要で
あるか？

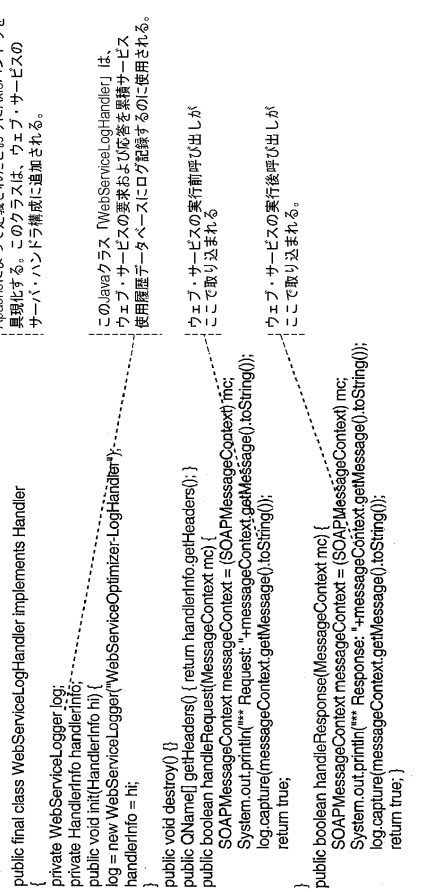
【図４】


サービス
呼び出し検出
56
36
サービス呼び出し署名が
累積サービス使用履歴
データベースに格納される
58
累積サービス
使用履歴
データベース

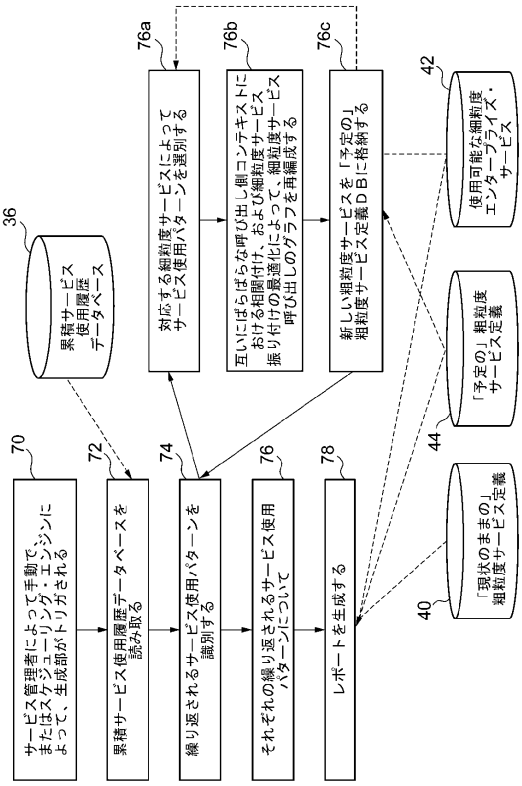
【図５】


この要求ハンドラは、
ウェブ・サービス実行前に
ウェブ・サービス呼び出しの
ログ記録に使用される
Axis要求ハンドラ
要求者
移送
Axisエンジン
ウェブ・サービス
36
Axis応答ハンドラ
累積サービス
使用履歴
データベース
この応答ハンドラは、ウェブ・サービス
実行後に、特に出力パラメータを集めるための
ウェブ・サービス呼び出しのログ記録に
使用される。

【図６】


このJavaクラス「WebServiceLogHandler」は、
Apacheによって定義されたとおりにAxisハンドラを
具現化する。このクラスは、ウェブ・サービスの
サーバ・ハンドラ構成に追加される。
このJavaクラス「WebServiceLogHandler」は、
ウェブ・サービスの要求および応答を累積サービス
使用履歴データベースにログ記録するのに使用される。
ウェブ・サービスの実行前呼び出しが
ここで取り込まれる
ウェブ・サービスの実行後呼び出しが
ここで取り込まれる。
public final class WebServiceLogHandler implements Handler
{
private WebServiceLogger log;
private HandlerInfo handlerInfo;
public void init(HandlerInfo hi) {
log = new WebServiceLogger("WebServiceOptimizer-LogHandler");
handlerInfo = hi;
}
public void destroy() {}
public QName[] getHeaders() { return handlerInfo.getHeaders(); }
public boolean handleRequest(MessageContext mc) {
SOAPMessageContext messageContext = (SOAPMessageContext) mc;
System.out.println("*** Request: "+messageContext.getMessage().toString());
log.capture(messageContext.getMessage().toString());
return true;
}
public boolean handleResponse(MessageContext mc) {
SOAPMessageContext messageContext = (SOAPMessageContext) mc;
System.out.println("*** Response: "+messageContext.getMessage().toString());
log.capture(messageContext.getMessage().toString());
return true; }

【図７】


36
累積サービス
使用履歴
データベース
70
サービス管理者によって手動で、
またはスケジューリング・エンジンに
よって、生成部がトリガされる
72
累積サービス使用履歴データベースを
読み取る
74
繰り返されるサービス使用パターンを
識別する
76
それぞれの繰り返されるサービス使用
パターンについて
78
レポートを生成する
76a
対応する細粒度サービスによって
サービス使用パターンを選別する
76b
互いにばらばらな呼び出し側コンテキストに
おける相関付け、および細粒度サービス
振り付けの最適化によって、細粒度サービス
呼び出しのグラフを再編成する
76c
新しい粗粒度サービスを「予定の」
粗粒度サービス定義ＤＢに格納する
42
使用可能な細粒度
エンタープライズ・
サービス
44
「予定の」粗粒度
サービス定義
40
「現状のままの」
粗粒度サービス定義

フロントページの続き

(72)発明者　ライオネル・モンメイヤ
フランス０６０００　ニース　ルー・ヴァルペルガ　１

合議体
審判長　吉岡　浩
審判官　宮司　卓佳
審判官　石井　茂和

(56)参考文献　特開２００２－１８３０９２（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F15/00