(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2017-123174
(P2017-123174A)
(43)公開日　平成29年7月13日(2017.7.13)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 9/445	(2006.01)	G06F	9/06	650C	5B376

審査請求 有　請求項の数 20　ＯＬ　（全 19 頁）

(21)出願番号　特願2017-17884(P2017-17884)
(22)出願日　平成29年2月2日(2017.2.2)
(62)分割の表示　特願2015-236788(P2015-236788)の分割
原出願日　平成23年10月8日(2011.10.8)
(31)優先権主張番号　13/229,697
(32)優先日　平成23年9月10日(2011.9.10)
(33)優先権主張国　米国(US)

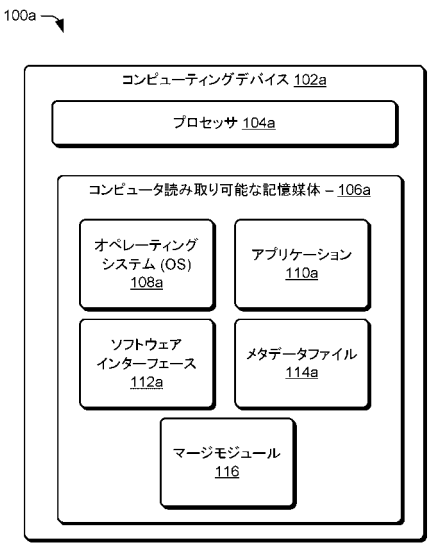
（特許庁注：以下のものは登録商標）
１．ＶＩＳＵＡＬ　ＢＡＳＩＣ

(71)出願人　314015767
マイクロソフト　テクノロジー　ライセンシング，エルエルシー
アメリカ合衆国　ワシントン州　９８０５２　レッドモンド　ワン　マイクロソフト　ウェイ
(74)代理人　100107766
弁理士　伊東　忠重
(74)代理人　100070150
弁理士　伊東　忠彦
(74)代理人　100091214
弁理士　大貫　進介

最終頁に続く

(54)【発明の名称】柔軟性の高いメタデータの合成

(57)【要約】　（修正有）
【課題】複数の異なる型システム同士の間における型解決の仕組みを抽象化するための機能を提供する。
【解決手段】少なくとも１つの型が、プログラムに基づいてアクセス可能な１つ以上のファイルに記述される。複数の異なる型システムを使用するアプリケーションは、型の記述が存在する情報格納場所の知識なしに、プログラムに基づいて少なくとも１つの型システムの型にアクセスし、その型を解決することができる。あるいは、又は加えて、プログラムに基づいてアクセス可能な１つ以上のファイルに含まれる型の記述が分析され、型の記述に少なくとも部分的に基づいて、プログラムに基づいてアクセス可能な１つ以上の新たなファイルへと再構築される。
【選択図】図１Ａ



【特許請求の範囲】
【請求項１】
コンピュータにより実行される方法において、
１つ以上の出力ファイルの生成に関連する１つ以上の入力基準を受け取るステップであって、少なくとも１つの入力基準が名前空間階層に関連付けられる、ステップと、
前記名前空間階層内に含まれる少なくとも１つの型に関連付けられる情報を含む、１つ以上の入力メタデータファイルを受け取るステップであって、前記少なくとも１つの型に関連付けられる前記情報は、抽象型システムで記述される型解決情報を含む、ステップと、
前記１つ以上の入力メタデータファイルを受け取ったことに応答して、少なくとも部分的に前記１つ以上の入力基準に基づいて、前記少なくとも１つの型に関連付けられる前記情報を再編成するステップと、
前記少なくとも１つの型に関連付けられる前記再編成された情報を含むように構成される、少なくとも１つの出力メタデータファイルを生成するステップと、
を具備する、方法。
【請求項２】
前記少なくとも１つの出力メタデータファイルは、少なくとも部分的に前記名前空間階層に基づく命名規則を利用する、
請求項１に記載の方法。
【請求項３】
名前空間階層に関連付けられる前記少なくとも１つの入力基準は、前記情報を前記１つ以上の出力ファイルに分割する名前空間階層のレベルを指定する基準を備える、
請求項１に記載の方法。
【請求項４】
前記少なくとも１つの型は、オブジェクト指向クラスを備える、
請求項１に記載の方法。
【請求項５】
名前空間階層情報について前記少なくとも１つの型に関連付けられる前記情報を分析するステップを更に備える、
請求項１に記載の方法。
【請求項６】
前記少なくとも１つの出力メタデータファイルを検索するための要求を受け取るステップを更に備える、
請求項１に記載の方法。
【請求項７】
前記少なくとも１つの出力メタデータファイルを検索するための前記受け取った要求は、名前空間階層レベルを指定する、
請求項６に記載の方法。
【請求項８】
前記少なくとも１つの型に関連付けられる前記情報は、オブジェクト指向の関連付けを備える、
請求項１に記載の方法。
【請求項９】
前記抽象型システムは、特定のプログラミング言語とは独立の型システムである
請求項１に記載の方法。
【請求項１０】
コンピュータによって実行されると、該コンピュータに、
１つ以上の出力ファイルの生成に関連する１つ以上の入力基準を受け取るステップであって、少なくとも１つの入力基準が名前空間階層を指定する、ステップと、
前記名前空間階層内に含まれる少なくとも１つの型に関連付けられる情報を含む、１つ

以上の入力メタデータファイルを受け取るステップであって、前記少なくとも１つの型は、オペレーティングシステムによって提供される機能を表し、前記少なくとも１つの型に関連付けられる前記情報は、抽象型システムで記述される型解決情報を含む、、ステップと、
少なくとも部分的に前記１つ以上の入力基準に基づいて再編成される前記少なくとも１つの型に関連付けられる前記情報を含むように構成される、少なくとも１つの出力メタデータファイルを生成するステップと、
を備える動作を実行させる、コンピュータプログラム。
【請求項１１】
前記少なくとも１つの出力メタデータファイルは、少なくとも部分的に前記名前空間階層に基づく命名規則を利用する、
請求項１０に記載のコンピュータプログラム。
【請求項１２】
前記少なくとも１つの出力メタデータファイルは、オペレーティングシステムソフトウェアインタフェースに関連付けられる記述を含む、
請求項１０に記載のコンピュータプログラム。
【請求項１３】
名前空間階層を指定する前記少なくとも１つの入力基準は、前記情報を前記１つ以上の出力ファイルに分割する名前空間階層のレベルを指定する基準を備える、
請求項１０に記載のコンピュータプログラム。
【請求項１４】
前記少なくとも１つの型は、オブジェクト指向クラスを備える、
請求項１０に記載のコンピュータプログラム。
【請求項１５】
前記動作は、名前空間階層情報について前記少なくとも１つの型に関連付けられる前記情報を分析するステップを更に備える、
請求項１０に記載のコンピュータプログラム。
【請求項１６】
システムにおいて：
１つ以上のプロセッサと；
前記１つ以上のプロセッサに通信可能に結合されるメモリであって、実行に応答して、前記１つ以上のプロセッサに、
１つ以上の出力ファイルの生成に関連する１つ以上の入力基準を受け取らせ、少なくとも１つの入力基準が名前空間階層の第１のレベルに関連付けられ、
前記名前空間階層内に含まれる少なくとも１つの型に関連付けられる情報を含む、１つ以上の入力メタデータファイルを受け取らせ、前記少なくとも１つの型に関連付けられる前記情報が、抽象型システムで記述される型解決情報を含んでおり、
再グループ化される前記情報が前記名前空間階層の前記第１のレベル及び前記名前空間階層の第２のレベルを含むように前記１つ以上の入力基準に少なくとも部分的に基づいて再グループ化される、前記少なくとも１つの型に関連付けられる前記情報を含むよう構成される、少なくとも１つの出力メタデータファイルを生成させる、
命令を記憶するメモリと；
を具備する、システム。
【請求項１７】
前記１つ以上の出力メタデータファイルは、特定のプログラミング言語とは独立の前記少なくとも１つの型を記述するように構成される、
請求項１６に記載のシステム。
【請求項１８】
前記１つ以上のプロセッサは、前記少なくとも１つの型の名前空間に合致するファイル名を有するファイルについて、前記１つ以上の出力メタデータファイルを検索するように

更に構成される、
請求項１６に記載のシステム。
【請求項１９】
前記１つ以上のプロセッサは、前記少なくとも１つの型が存在する場所についての知識なしに、前記１つ以上の出力メタデータファイルを検索するように更に構成される、
請求項１６に記載のシステム。
【請求項２０】
前記１つ以上のプロセッサは、前記１つ以上の出力メタデータファイルへのアクセスを通して別の型システムを解決するように更に構成される、
請求項１６に記載のシステム。

【発明の詳細な説明】
【技術分野】
【０００１】
コンピューティング・デバイスは多くの場合、コンピューティング・デバイスのハードウェア及び／又はソフトウェア・リソースを管理するための手法として、オペレーティングシステムを実行する。いくつかのケースにおいて、オペレーティングシステムは、単純化され、プログラムに基づいた、リソースへのアクセスを提供し得る。例えばオペレーティングシステムは、さまざまなコンポーネントを外部からアクセス可能とするためのアプリケーション・プログラミング・インターフェース（ＡＰＩ）を含み得る。アプリケーションを実装するのに使用されたプログラミング言語及び／又は型システムが、ＡＰＩにより提供されるプログラミング言語及び／又は型システムとは異なっていたとしても、アプリケーションによるＡＰＩを首尾よく呼び出すことが出来る場合がある。がしかし、これは、どの型がＡＰＩに関連付けられているかをアプリケーションが知っている場合に限られる。例えばＡＰＩは、１つ以上の入力及び／又は出力パラメータを含み得る。ＡＰＩを呼び出すために、プログラマは、ＡＰＩのパラメータだけでなく、どのデータ型が各パラメータに関連付けられているかも決定しなくてはならない。
【背景技術】
【０００２】
上述したように、呼び出し側で使用されるプログラミング言語の型システムとは異なる型システムによってＡＰＩが記述されている場合がある。そこで、異なる型システムを橋渡し（ブリッジング）するために、互いに異なる型システム同士の間を翻訳するためのラッパー・コードをプログラマが書くことが一般的に行われている。プログラマがプログラムの中にＡＰＩアクセスを含ませるための１つの手法は、１つ以上のファイル及び／又は名前空間によるＡＰＩ定義をソースコード中に含ませることである。ソースコード中にファイル及び／又は名前空間を首尾よく組み込むために、ソースコードは、ファイル／名前空間の特定の場所の参照（例えばハード・コーディングされたパス、そのパスによりレジストリ・キーにアクセスする、等）を含むように構成され得る。場所、ファイル名、及び／又は、名前空間の名前が変わった場合、コード及び／又はソフトウェア・ツールが適切な変更によって更新されるまで、連係は中断する。
【発明の概要】
【発明が解決しようとする課題】
【０００３】
発明の概要に関する以下の説明は、詳細な説明において以下にさらに説明される概念の選択を、単純化された形態で紹介するために提供される。発明の概要に関する以下の説明は、特許請求の範囲に記載された発明の主題において鍵となる技術的特徴、すなわち、技術的重要な特徴を限定することを意図したものではない。同時に、当該発明の概要に関する以下の説明は、特許請求の範囲に記載された発明の主題に係る技術的範囲を限定するために使用されることを意図したものでもない。
【課題を解決するための手段】

【０００４】
本発明に係るさまざまな実施形態は、複数の異なる型システム同士の間における型解決の仕組みを抽象化するための機能を実現する。少なくとも１つの型が、プログラムに基づいてアクセス可能な１つ以上のファイルに記述され得る。いくつかの実施形態において、異なる型システムを使用するアプリケーションは、型の記述が存在する場所の知識なしに、その型システムの型に、プログラムに基づいてアクセスし、その型を解決することができる。あるいは、又は加えて、プログラムに基づいてアクセス可能な１つ以上のファイルに含まれた型の記述は分析され、型システムの記述に少なくとも部分的に基づいて、プログラムに基づいてアクセス可能な１つ以上の新たなファイルへと再構築されることができる。
【０００５】
同一の番号は、全図面を通して同様の特徴を参照するために使用される。
【図面の簡単な説明】
【０００６】
【図１Ａ】図１Ａは、１つ以上の実施形態に係る、本明細書で説明されるさまざまな原理が用いられ得る動作環境を示す図である。
【０００７】
【図１Ｂ】図１Ｂは、１つ以上の実施形態に係る、本明細書で説明されるさまざまな原理が用いられ得る動作環境を示す図である。
【０００８】
【図２】図２は、１つ以上の実施形態に係るアーキテクチャを示す図である。
【０００９】
【図３】図３は、１つ以上の実施形態に係るフローチャートである。
【００１０】
【図４】図４は、１つ以上の実施形態に係る関係図である。
【００１１】
【図５】図５は、１つ以上の実施形態に係るフローチャートである。
【００１２】
【図６】図６は、１つ以上の実施形態を実現するために利用され得る例示的なシステムを示す図である。
【発明を実施するための形態】
【００１３】
概要
本発明に係るさまざまな実施形態は、複数の異なる型システム同士の間における型解決の仕組みを抽象化するための機能を実現する。１つの型システムを使用するアプリケーションは、そのアプリケーションがどのように異なる型システム同士の間をブリッジするかについての知識を有する場合、もう１つの型システム中において呼び出しを実行し得る。例えば型システムの特性（例えばデータ型、データ型の振る舞い、関数呼び出しパラメータ、イベント、等）が、プログラムに基づいてアクセス可能な１つ以上のファイルに記述され得る。アプリケーションは、そのようなファイルにアクセスすることによって、異なる型システムを使用した型解決を実行し得る。本発明に係るいくつかの実施形態においては、型解決の仕組みが抽象化され得るので、アプリケーションは、どのファイルにアクセスするか、及び／又は、どこにファイルが存在するか、といった予備知識なしに上述した型システムの特性記述にアクセスすることができる。
【００１４】
以下の説明では、「動作環境」と題するセクションが提供され、１つ以上の実施形態が用いられ得る複数の環境を説明する。これに続き、「型解決アーキテクチャ」と題するセクションにおいて、プログラムに基づいて上述した型解決を実行するシステムを実現可能にするアーキテクチャを説明する。次に、「型記述の格納」と題するセクションにおいて、型記述を格納するための柔軟性の高い仕組みを実現可能にするために使用され得るさま

ざまな方法を説明する。最後に、「例示的なシステム」と題するセクションにおいて、１つ以上の実施形態を実現するために利用され得る例示的なシステムを説明する。

【００１５】

本発明に関し、以下に後述するさまざまな実施形態についての概要を提供したので、ここでは、本発明に係る１つ以上の実施形態が実現され得る基盤となる例示的な動作環境を考慮する。

【００１６】

動作環境

図１Ａ及び図１Ｂは、１つ以上の実施形態に係る動作環境を示し、図１Ａ及び図１Ｂにおいては、当該動作環境はそれぞれ、１００ａ及び１００ｂとして一般化した形で示されている。図１Ａは、以下において後述するとおり、１つ以上のメタデータファイルの生成に関連して利用され得る例示的な動作環境を示す。図１Ａの環境は、「ビルド段階」の環境とみなされ得る。図１Ｂは、柔軟性の高い型システムを使用した型解決に関連して利用され得る例示的な動作環境を示す。図１Ｂの環境は、ランタイム時の環境とみなされ得る。本発明に係るいくつかの実施形態において、動作環境１００ａ及び１００ｂは、少なくともいくつかの同様のコンポーネントを有する。したがって、簡潔化のために、図１Ａ及び図１Ｂは一緒に説明される。図１Ａに関連付けられたコンポーネントが、「１ＸＸａ」との命名規則に従って命名されるコンポーネントとして特定される場合、図１Ｂに関連付けられる同様のコンポーネントは、「１ＸＸｂ」との命名規則に従って命名されるコンポーネントとして特定される。同様に、動作環境特有のコンポーネントは、単に「１ＸＸ」として特定される。

【００１７】

動作環境１００ａ及び１００ｂはそれぞれ、１つ以上のプロセッサ１０４ａ、１０４ｂを有するコンピューティング・デバイス１０２ａ、１０２ｂ、及び１つ以上のコンピュータ読み取り可能な記憶媒体１０６ａ、１０６ｂを含む。コンピュータ読み取り可能な記憶媒体は、コンピューティング・デバイスに典型的に関連付けられる全ての形態の揮発性及び不揮発性のメモリ及び／又は記憶媒体を含み得る。そのような媒体の具体例を限定目的ではなく単に例示目的で挙げるならば、当該媒体は、ＲＯＭ、ＲＡＭ、フラッシュ・メモリ、ハードディスク、リムーバブル・メディア、等を含み得る。コンピューティング・デバイスの１つの特定の例が、図６において以下に示され説明される。

【００１８】

加えて、コンピューティング・デバイス１０２ａ、１０２ｂは、オペレーティングシステム（ＯＳ）１０８ａ、１０８ｂ、及びアプリケーション１１０ａ、１１０ｂを含む。オペレーティングシステム１０８ａ、１０８ｂは、コンピューティング・デバイス１０２ａ、１０２ｂのソフトウェア・リソース及び／又はハードウェア・リソースを管理するように構成された機能を表す。これは、メモリ管理、ファイル管理、サービス、関数、リソース管理、周辺機器管理、等を含み得る。アプリケーション１１０ａ、１１０ｂは、コンピューティング・デバイス１０２ａ、１０２ｂ上で、典型的にはオペレーティングシステム１０８ａ、１０８ｂにより支援されて、実行するように構成されたソフトウェアを表す。アプリケーション１１０ａ、１１０ｂは、以下においてさらに説明されるように、オペレーティングシステム１０８ａ、１０８ｂの型システムと同一の及び／又は異なる型システムで実現され得る。

【００１９】

コンピューティング・デバイス１０２ａ、１０２ｂはまた、１つ以上のソフトウェア・インターフェース１１２ａ、１１２ｂを含み、１つ以上のソフトウェア・インターフェース１１２ａ、１１２ｂは、ソフトウェア及び／又はアプリケーションによって提供される関数、サービス、データ、等への、プログラムに基づいたアクセス手段を表す。ソフトウェア・インターフェース１１２ａ、１１２ｂは、オペレーティングシステム１０８ａ、１０８ｂ及び／又はアプリケーション１１０ａ、１１０ｂへの、プログラムに基づいたアクセスを可能にし得る。例えばアプリケーション１１０ａ、１１０ｂは、ソフトウェア・イ

ンターフェース１１２ａ、１１２ｂを呼び出すことにより、オペレーティングシステム１０８ａ、１０８ｂによって提供される機能にアクセスし得る。いくつかの実施形態において、ソフトウェア・インターフェースを介して提供される型システムとは異なる型システムを使用するアプリケーション１１２ａ、１１２ｂは、以下においてさらに説明するように、型の相違をプログラムに基づいて解決し得る。

【００２０】

加えて、コンピューティング・デバイス１０２ａ、１０２ｂはまた、１つ以上のメタデータファイル１１４ａ、１１４ｂを含み、１つ以上のメタデータファイル１１４ａ、１１４ｂは、ソフトウェア・インターフェース１１２ａ、１１２ｂ、オペレーティングシステム１０８ａ、１０８ｂ、及び／又はアプリケーション１１０ａ、１１０ｂに関連付けられた、入力パラメータの型、パラメータの呼び出し順序、インターフェース間の関係、等といった情報を含む、１つ以上の機械読み取り可能なファイルを表す。あるいは、又は加えて、ソフトウェア・インターフェースは、コンピューティング・デバイス１０２ａ、１０２ｂに接続された周辺機器、例えばプリンタ、スキャナ、スマートフォン、等に関連付けられ得る。いくつかの実施形態において、メタデータファイル１１４ａ、１１４ｂは、以下においてさらに説明されるように、任意の適切な手法でインターフェースを記述するように構成され得る。

【００２１】

コンピューティング・デバイス１０２ａはまた、マージ・モジュール１１６を含む。マージ・モジュール１１６は、１つ以上のメタデータファイルを読み出し、ファイルの内容を分析し、再構成された内容を含む１つ以上の新たなメタデータファイルを出力し得る機能を表す。本発明に係るいくつかの実施形態において、内容は、型記述に少なくとも部分的に基づいて再編成され得る。

【００２２】

コンピューティング・デバイス１０２ｂは、型解決モジュール１１８を含む。型解決モジュール１１８は、関連付けられた型データにアクセスするための要求を受信し、型解決情報の場所を特定するように構成された機能を表す。本発明に係るいくつかの実施形態において、型解決モジュール１１８は、情報が場所を変えても無関係に、ユーザ入力なしで型解決情報の場所を特定し得る。例えば型解決モジュール１１８は、以下においてさらに説明されるように、情報が第１のファイルから第２のファイルに移動した場合、ユーザ入力なしでインターフェース記述情報の場所を特定し得る。

【００２３】

コンピューティング・デバイス１０２ａ、１０２ｂの具体例を限定目的ではなく、例示目的で挙げるならば、これらは、例えばデスクトップ型コンピュータ、ポータブル型コンピュータ、ノートブック型コンピュータ、携帯情報端末（ＰＤＡ）のようなハンドヘルド型コンピュータ、携帯電話、等といった任意の適切なコンピューティング・デバイスとして具現化され得る。

【００２４】

例示的な動作環境を説明したので、ここでは、場所のファイル名及び／又は場所とは無関係に型解決を可能にするように構成された型解決アーキテクチャの説明を考慮する。

【００２５】

型解決アーキテクチャ

プログラミング言語が技術的に進歩するにつれ、それらの能力も進歩する。例えば第１のプログラミング言語で書かれたアプリケーションが、第２のプログラミング言語で書かれたソフトウェアから呼び出され得る。本来、プログラミング言語は、異なる型システムを有する。したがって、異なる型システムの下に存在するソフトウェアを首尾よく呼び出すために、第１のプログラミング言語は、自身の型システムとは異なる型を解決するための技法を使用する。例えばプログラマは、型を変換するためのラッパー・コードを手作業で書くことが可能である。あるいは、型システムの定義がファイルに格納され、プログラムに基づいてアクセスされ得る。

【００２６】
型の定義を含むファイルにプログラムに基づいてアクセスすることは、機能及び記述をアプリケーションがランタイムに決定することを可能にする。しかしながら、ファイルにアクセスするということは、どのファイルにアクセスするべきであるかだけでなく、ファイルの格納場所も知っていなければならないことを暗に意味する。ファイルの内容が変わった場合及び／又はファイルの格納場所が変わった場合、ファイルにアクセスしようと試みるアプリケーションは、適切に更新されない限り、場合によっては失敗し得る。これは時に、アプリケーションとファイルとの間の意図しない結合を生じさせる可能性がある。
【００２７】
本発明に係るさまざまな実施形態は、複数の型システム同士の間における型解決の仕組みを抽象化するための機能を実現する。型解決に関連付けられた情報は、プログラムに基づいてアクセス可能なファイルに格納され得る。いくつかの実施形態において、１つの型システムを使用するアプリケーションは、ファイルへのアクセスを通じて、もう１つの型システムを使用することにより動的に型解決を実行し得る。あるいは、又は加えて、アプリケーションは、どのファイルにアクセスするのか、及び／又は、どこにファイルが存在するのかに関する知識を持つことなしに、ファイルにアクセスし得る。
【００２８】
図２を見ると、１つ以上の実施形態に係る例示的なアーキテクチャ２００が示されている。アーキテクチャ２００は、オペレーティングシステム２０２を含み、オペレーティングシステム２０２は、コンピューティング・デバイス上で実行するように構成され得る。簡潔化のためにオペレーティングシステム２０２の全体が示されていないことが理解されるべきである。オペレーティングシステム２０２は、コンピューティング・デバイスに関連付けられたリソースを管理するように構成された１つ以上のオペレーティングシステム・コンポーネント２０４を含む。いくつかの実施形態において、オペレーティングシステム・コンポーネント２０４は、リソースを管理することに関連付けられた１つ以上のサービス及び／又は特徴だけでなく、リソースへのプログラムに基づいたアクセスをも提供し得る。オペレーティングシステム・コンポーネント２０４はまた、オペレーティングシステム２０２に関連付けられた基本的な要素だけでなく、基本的な要素から構築された複合的な要素をも含み得る。この例はオペレーティングシステム２０２によって提供される機能を外部からアクセス可能とする（エクスポーズする）アーキテクチャを示しているが、このアーキテクチャが、特許請求の主題の精神から逸脱することなく、他の適切なタイプのアプリケーションによって提供される機能を外部からアクセス可能とするために適用され得ることが認められ、理解されるべきである。
【００２９】
本発明に係るいくつかの実施形態において、オペレーティングシステム・コンポーネント２０４は、ここではオペレーティングシステム・インターフェース２０６として示されている１つ以上のインターフェースを介して外部からアクセス可能とされ得る（エクスポーズされ得る）。このようなインターフェースは、任意の適切な形態のインターフェース、例えばアプリケーション・プログラミング・インターフェース（ＡＰＩ）とすることが可能である。アプリケーションは、以下においてさらに説明されるように、呼び出し中及び／又は実行中のオペレーティングシステム・インターフェース２０６により、オペレーティングシステム・コンポーネント２０４によって提供される機能にアクセスし得る。本実施形態に係るいくつかのケースにおいて、アプリケーションは、オペレーティングシステム・インターフェース２０６を記述するために使用される型システムとは異なる型システムを利用する。いくつかのケースにおいて、オペレーティングシステム・インターフェース２０６は、アプリケーション・バイナリ・インターフェース（ＡＢＩ）を含み得る。ＡＢＩは、機械により解釈実行可能な命令語レベルで、関数、方法、ＡＰＩ、等を呼び出すためのバイナリ・コントラクトを記述する。バイナリ・コントラクトは、関数に関連付けられた識別子又は名前、関数を呼び出すために使用され得るシグネチャ、関数に受け渡されるパラメータの順序、及び／又は、パラメータに関連付けられたデータ型、等を含み

得る。あるいは、又は加えて、バイナリ・コントラクトは、型システムの少なくとも１つの型に関連付けられたエクスポーズ動作の挙動に関する定義及び／又は規則を含み得る。

【００３０】

オペレーティングシステム２０２はまた、さまざまなメタデータ２０８を含む。メタデータ２０８は、関連付けられたオペレーティングシステム・インターフェース２０６のさまざまな態様を記述する型解決情報、例えばバージョン情報、どの方法が利用可能か、インターフェースがどのパラメータを採用するか、パラメータのデータ型、パラメータを受け渡す順序、データ型の振る舞い、及び／又は解決情報、等を含み得る。本発明に係るいくつかの実施形態において、メタデータは、インターフェースに関連付けられた階層情報、例えばインターフェース間の関係を記述した、及び／又は、オブジェクト指向でインターフェースを記述した情報、を含み得る。メタデータはまた、クラスの記述、クラスの関連付けられた方法及びパラメータ、等を含み得る。本実施形態に係るいくつかのケースにおいて、メタデータは、抽象型システム（例えば特定のプログラミング言語と無関係の型システム）を使用してインターフェースを記述し得る。あるいは、又は加えて、メタデータは、特定の型システム、例えば抽象型システムの記述を含み得る。すると、特定のプログラミング言語は、特定の型システム（例えば抽象型システム）の記述を特定のプログラミング言語にマッピングし得る。さらに、どのインターフェースが利用可能かを決定することを望むプログラマは、手作業で、及び／又はプログラムに基づいて、各インターフェースの記述にアクセスすることができる。例えばどのインターフェースがオペレーティングシステム・コンポーネント２０４のアクセス手段として存在するか、インターフェースがどの型システムで記述されているか、どのようにそれらを呼び出すか、を決定するために、プログラマは、関連付けられたメタデータ２０８にアクセスし得る。

【００３１】

アーキテクチャ２００はまた、１つ以上のアプリケーション２１０を含む。アプリケーション２１０は、１つ以上のプログラミング言語、例えばＨＴＭＬ、ＪａｖａＳｃｒｉｐｔ（登録商標）、Ｖｉｓｕａｌ　Ｂａｓｉｃ、Ｃ＃、Ｃ＋＋、等から生成された１つ以上のアプリケーションを含み得る。いくつかの実施形態において、アプリケーション２１０は、オペレーティングシステム・コンポーネントへの１つ以上の呼び出しを含む。いくつかのケースにおいて、アプリケーション２１０は、まず、どのインターフェースが利用可能かをプログラムに基づいて決定し、次に、決定されたインターフェースのうちの１つ以上への呼び出しを行うように構成され得る。いくつかのケースにおいて、アプリケーション２１０は、以下においてさらに説明するように、１つ以上の生成された言語投影モジュール２１２からの支援を受けて、インターフェースにアクセスする。

【００３２】

１つ以上の実施形態において、生成された言語投影モジュール２１２は、抽象型の定義を特定のプログラミング言語にマッピングする。任意の適切なプログラミング言語がマッピングされることができ、それらの例は上述したとおりである。いくつかの実施形態において、生成された言語投影モジュールは、プログラミング言語ごとに一意であり得る。本発明に係る他の実施形態において、生成された言語投影モジュールは、多目的な機能モジュールであり、複数のプログラミング言語によって利用され得る。上述したマッピングは、抽象型システムを使用して記述される現在及び未来のインターフェースが、追加のプログラミング・ステートメント（例えばラッパー関数）なしに特定のプログラミング言語にアクセスできるようにすることができる。上述したマッピングはさらに、特定のプログラミング言語にとってネイティブの手法で、当該特定のプログラミング言語がインターフェースを呼び出すことを可能にする。任意の適切なタイプの情報、例えばクラス、データ型、関数ポインタ、構造、等が、マッピングされ得る。

【００３３】

インターフェースの記述は、アプリケーションがどのようにインターフェースを呼び出すべきかを記述するだけでなく、インターフェースに関連付けられた型システムの挙動を記述するためにも使用され得る。記述の情報格納場所が特定されたことに応答して、アプ

リケーションは、どのようにインターフェースを呼び出すかを動的に決定し、アプリケーションとインターフェースとの間における型システムの相違を解決し得る。本発明に係るいくつかの実施形態において、異なる型システムを使用するアプリケーションは、型記述が存在する情報格納場所の知識なしに、プログラムに基づいて、インターフェースによって利用される型にアクセスし、その型の型解決を実行し得る。あるいは、又は加えて、以下においてさらに説明するように、どのような方法で記述がグループ化されるかを特定することにより、自動的な型解決を実現可能にすることができる。

【００３４】

以下の実施例においては、ＪａｖａＳｃｒｉｐｔを使用してアプリケーションを書くプログラマがいると仮定する。この例はＪａｖａＳｃｒｉｐｔを使用した実装形態を説明するが、任意のプログラミング言語が特許請求の範囲に記載された技術思想から逸脱することなく使用され得ることが当業者にとって自明なものとして認められ理解されるべきである。外部の機能にアクセスするために、プログラマは、機能に関連付けられた名前空間及び／又は型を識別するように構成されたステートメントをソースコード中に含めることができる。例えば外部の型は、ソースコード内に以下のようにして含められ、及び／又は、以下のソースコードで記述され得る。

ＯｐｅｒａｔｉｎｇＳｙｓｔｅｍ．Ｆｏｏ．Ｂａｒ．Ｔｙｐｅ１

【００３５】

この特定の例において、Ｔｙｐｅ１は、アクセスされる機能を表現する。これは、任意の適切なタイプの機能であることができ、その例は、上述及び後述されるとおりである。上記シンタックスは、Ｔｙｐｅ１の情報格納場所を特定するために横断的に走査することができる多層構造の名前空間階層を表現する。最も高い階層レベルにおいては、Ｔｙｐｅ１は、「ＯｐｅｒａｔｉｎｇＳｙｓｔｅｍ」として識別された名前空間に存在する。「ＯｐｅｒａｔｉｎｇＳｙｓｔｅｍ」内に「Ｆｏｏ」として識別された名前空間が存在する。同様に、「Ｆｏｏ」内には「Ｂａｒ」として識別された名前空間が存在し、「Ｂａｒ」にはＴｙｐｅ１が存在する。したがって、シンタックスは、Ｔｙｐｅ１の論理的な情報格納場所を識別するために使用され得る。しかしながら、Ｔｙｐｅ１の物理的な情報格納場所（例えばどのファイルにＴｙｐｅ１情報が存在するか、及び／又は、どのディレクトリにファイルが存在するか）は時間の経過に従って変わる可能性がある。従来、物理的な情報格納場所は、アプリケーション内においてパス及びファイル名を（直接的又は間接的に）ハード・コーディングすることによって決定されることができた。したがって、Ｔｙｐｅ１情報のファイル名及び／又はパスが変わった場合、アプリケーションによって利用されるファイル名及び／又はパスの任意の直接的又は間接的な参照は、更新されるまで不適切であろう。さらに、直接的又は間接的な参照が更新されるまで、アプリケーションは、場合によっては適切に機能することができなかった。

【００３６】

本発明に係るさまざまな実施形態は、型解決の仕組みを抽象化するための機能を実現する。例えば型は、関連付けられた型解決情報の情報格納場所に関する完全な知識なしに、アプリケーションにより、プログラムに基づいて解決されることができる。本発明に係るいくつかの実施形態において、型解決情報は、情報格納場所を表している新たな情報によって、解決情報を利用するアプリケーションを更新せずに、再配置され得る。アプリケーションは、どのファイル中に型解決情報が存在するかに関わらず、型解決情報の場所を特定するように構成されることが可能となる。

【００３７】

図３を見ると、本発明の１つ以上の実施形態に係る方法を実施するための一連の処理ステップを説明したフローチャートが示されている。当該方法は、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、又はそれらの組み合わせによって実行され得る。本発明に係る少なくともいくつかの実施形態において、当該方法の実施態様は、１つ以上のコンピューティング・デバイス上で実行中の１つ以上の適切に構成されたソフトウェア・モジュール（例えば図１Ｂの型解決モジュール１１８）によって実現され得る。本発明に

係るいくつかの実施形態において、当該方法は、ユーザによる介入なしに実行され得る。

【００３８】

ステップ３０２は、解決される型と一致するファイル名を有するファイルを検索する。これは、上述した複数のメタデータファイルの検索を含み得る。上記シンタックスの例を参照すると、アプリケーションがＯｐｅｒａｔｉｎｇＳｙｓｔｅｍ．Ｆｏｏ．Ｂａｒ．Ｔｙｐｅ１にアクセスすることに応答して、ステップ３０２は、「ＯｐｅｒａｔｉｎｇＳｙｓｔｅｍ．Ｆｏｏ．Ｂａｒ．Ｔｙｐｅ１」という名前を有するファイルを検索する。任意の適切なタイプの検索が実行され得る。例えば検索は、型とファイル名との間の完全一致、型とファイル名との間の部分一致、等を探すように構成され得る。あるいは、又は加えて、検索は、１つのディレクトリ、複数のディレクトリ、及び／又はサブディレクトリ、又はそれらの任意の組み合わせの中を検索するように構成され得る。

【００３９】

ステップ３０４は、ファイルが存在するかどうかを決定する。ファイルが存在するとの決定に応答して、ステップ３０６は、型が名前空間であることを示す情報を送信する。ファイルが存在しないとの決定に応答して、ステップ３０８は、解決される型に関連付けられた名前空間と一致するファイル名を検索する。いくつかのケースにおいて、これは、どの名前空間上を検索するかを決定するために、名前空間の階層を表す情報を含むストリングを操作することを含み得る。上述した例において、Ｔｙｐｅ１は、名前空間の階層において３階層下に配置されるものとして示されている。ストリングに対する操作は、型（例えばＴｙｐｅ１）を表す部分を削除し、名前空間の階層を表す部分を残すように実行され得る。ここで、ステップ３０８は、名前空間の階層「ＯｐｅｒａｔｉｎｇＳｙｓｔｅｍ．Ｆｏｏ．Ｂａｒ」に関連付けられた名前を検索するだろう。上記と同様に、この検索は、完全一致又は部分一致の検索方法に従って検索を実行し、１つのディレクトリの中を検索し、さらにそのサブディレクトリの中を検索する、といった具合に構成され得る。

【００４０】

ステップ３１０は、ファイル名に関連付けられたファイルが存在するかどうかを決定する。ファイルが存在するとの決定に応答して、ステップ３１２は、型に関連付けられた情報のためにファイルを処理する。

【００４１】

ステップ３１４は、型に関連付けられた情報がファイル内に配置されているかどうかを決定する。型に関連付けられた情報がファイル内にあるとの決定に応答して、ステップ３１６は、情報を戻す。例えば情報は、呼び出しているプログラムに戻され得る。しかしながら、型に関連付けられた情報がファイル内にないとの決定に応答して、処理はステップ３１８に進む。

【００４２】

ステップ３１８においては、より高い階層レベルの名前空間と一致するファイル名が検索される。ステップ３１８におけるこの検索動作は、例えばステップ３１０の実行時におけるファイルが存在しないとの決定に応答する形で、又は、ステップ３１４の実行時に、型に関連付けられた情報の格納場所をファイル内において特定することができないことに応答する形で実行することが可能である。より高い階層レベルの名前空間は、任意の適切な手法で決定され得る。例えば上記の例では、ストリングが、名前空間階層における１つ上のレベル（例えば「ＯｐｅｒａｔｉｎｇＳｙｓｔｅｍ．Ｆｏｏ」）を反映させるようにさらに操作され得る。ステップ３１８は、より高い階層レベルの名前空間を決定し、関連付けられた名前を有するファイルを検索する。

【００４３】

ステップ３２０は、ファイル名に関連付けられたファイルが存在するかどうかを決定する。ステップ３１０のケースのように、ファイルが存在すると決定された場合、処理は、ステップ３１２、３１４、及び／又は３１６に進み、ファイルが、関連付けられた型情報のために処理される。

【００４４】

ファイルが存在しないとの決定に応答して、ステップ３２２は、さらに別の名前空間階層レベルが存在するかどうかを決定する。これは、任意の適切な手法で決定され得る。例えばレベル分離文字についてストリングを検索することができる。上記の例では、シンタックス「．」が、名前空間階層のレベルを区別する。

【００４５】

さらに別の名前空間階層レベルが存在するとの決定に応答して、上述した処理が繰り返され、処理はステップ３１８に戻り、新たに決定された名前空間を有するファイルを検索する。ステップ３１８、３２０、及び３２２は、適切なファイルが見つかるまで、又は、名前空間階層の最上部に到達し、最上部が検索されるまで、繰り返し実行される。さらに別の名前空間階層レベルが存在しないとの決定に応答して、ステップ３２４はエラーを戻す。これは、例外処理を実行すること、ポップアップ・ダイアログ・ボックスを表示すること、等を含み得る。

【００４６】

上述した方法の使用により、複数のファイル及び／又は情報格納場所が、型解決情報について検索され得る。説明したように、型の検索は、階層名前空間情報に少なくとも部分的に基づき得る。図３は、より低い名前空間階層レベル（例えば「ＯｐｅｒａｔｉｎｇＳｙｓｔｅｍ．Ｆｏｏ．Ｂａｒ．Ｔｙｐｅ１」）から始まり、型の格納場所が特定されるか又は最上部の名前空間階層レベル（例えば「ＯｐｅｒａｔｉｎｇＳｙｓｔｅｍ」）に到達するまで、名前空間の各階層レベルを上に向かって横断的に走査するように実行される型の検索動作を説明する。しかしながら、この検索動作は、任意の適切な順序で実行され得る。本発明に係るいくつかの実施形態では、型の検索動作は、より高い名前空間階層レベル（例えば「ＯｐｅｒａｔｉｎｇＳｙｓｔｅｍ」）から始まり、型の格納場所が特定されるか最下部の名前空間階層レベル（例えば「ＯｐｅｒａｔｉｎｇＳｙｓｔｅｍ．Ｆｏｏ．Ｂａｒ．Ｔｙｐｅ１」）に到達するまで、名前空間の各階層レベルを下に向かって横断的に走査し得る。名前空間階層上の検索は、型解決情報がどこに存在し得るかを抽象化することができ、さらに、型解決情報にアクセスするアプリケーションに影響を及ぼさずに型解決情報の情報格納場所を変更することを可能にする。

【００４７】

場所のファイル名及び／又は場所とは無関係に型解決を可能にするように構成された型解決アーキテクチャが考慮されたので、ここでは、１つ以上の実施形態に係るフレキシブルな型記述の格納の説明を考慮する。

【００４８】

型記述の格納

メタデータファイルが、ソフトウェア・インターフェースのさまざまな態様を記述するために使用され得る。上述したように、メタデータファイルは、クラス階層、抽象型システム、関連付けられた方法、プロパティ、及びイベント、等によってインターフェースを記述する情報を含み得る。本実施形態に係るいくつかのケースにおいて、関連付けられた情報は、複数のファイル内に存在し得る。例えば異なるメタデータファイルが、同一の名前空間に関連する情報を含み得る。複数のメタデータファイルは、メタデータファイルの開発者に高い柔軟性を与え得る一方で、それは時に、メタデータファイルをユーザが検索する際の妨げとなり得る。以下の例では、各名前空間が、それ独自の関連付けられたメタデータファイルを有する例を考慮する。分割の観点からは、各名前空間のための別個のファイルは、名前空間に対する変更を、関連付けられたファイルに分離することができる。しかしながら、検索の観点からは、情報について複数のファイルを検索しなくてはならないことは、ランタイムでの検索時の性能を低下させ得る。さらに、知識の観点からは、何の情報が何のファイルの中にあるかの追跡を維持することは、ファイル数が増加すると複雑化し得る。

【００４９】

本発明に係るさまざまな実施形態は、型解決情報が存在するファイル名及びファイル格納場所の予備知識なしの型解決を実現可能にする。複数の入力ファイルから成るファイル

群の内容が分析され、少なくとも部分的には合成規則に基づいて分割され得る。複数の出力ファイルから成るファイル群を生成することができ、分割された内容を含むように構成される。出力ファイルは、内容がどこに存在するかの予備知識なしに内容の格納場所を特定することを可能にし得る。

【００５０】

例として、図４を見ると、１つ以上の実施形態に係る、記述言語ファイル、メタデータファイル、及びマージ・モジュール間の関係が示されている。本発明に係る少なくともいくつかの実施形態において、この関係図に示されたモジュールは、ソフトウェア、ハードウェア、又はそれらの任意の組み合わせ、例えば図１Ａのマージ・モジュール１１６として実現され得る。

【００５１】

例示され説明された実施形態において、Ｆｏｏ．ｉｄｌ　４０２は、１つ以上のインターフェースを記述するように構成された記述言語ファイルを表す。本発明に係るいくつかの実施形態において、インターフェースは、図１のオペレーティングシステム・インターフェース１０８及び／又はアプリケーション１１０といった、オペレーティングシステム及び／又はアプリケーションに関連付けられ得る。記述言語ファイルは、任意の適切な記述、マーク付け言語、及び／又はシンタックスを使用して、例えばインターフェース定義言語（ＩＤＬ）、拡張可能なマーク付け言語（ＸＭＬ）、等を用いて、インターフェースを記述し得る。この特定の例において、Ｆｏｏ．ｉｄｌは、ＩＤＬファイルを表す。

【００５２】

Ｆｏｏ．ｉｄｌ　４０２は、３つの型の記述、Ｔｙｐｅ　Ｆｏｏ．Ｂａｒ．Ｔｙｐｅ１　４０４、Ｔｙｐｅ　Ｆｏｏ．Ｂａｒ．Ｔｙｐｅ２　４０６、及びＴｙｐｅ　Ｆｏｏ．Ｂａｒ．Ｔｙｐｅ３　４０８を含む。３つのエントリを用いて説明するが、任意の適切な数の記述、ならびに型の記述が、特許請求の範囲に記載された発明の技術思想から逸脱することなく、Ｆｏｏ．ｉｄｌ　４０２に含まれ得ることが認められ理解されるべきである。例えば型は、クラス、インターフェース、方法、プロパティ、イベント、等の記述を含み得る。この例では、型４０４、４０６、及び４０８は、「Ｆｏｏ．Ｂａｒ」の階層名前空間に含まれるものとして記述される。

【００５３】

図４はまた、第２の記述言語ファイル、Ｂａｒ．ｉｄｌ　４１０を示す。Ｆｏｏ．ｉｄｌ　４０２と同様に、Ｂａｒ．ｉｄｌ　４１０は、３つの型、Ｔｙｐｅ　Ｆｏｏ．Ｂａｒ．Ｔｙｐｅ４　４１２、Ｔｙｐｅ　Ｆｏｏ．Ｑｕｕｘ．Ｔｙｐｅ１　４１４、及びＴｙｐｅ　Ｆｏｏ．Ｑｕｕｘ．Ｔｙｐｅ２　４１６を含む記述言語ファイルを表す。型４１２、４１４、及び４１６のシンタックスを参照すると、Ｂａｒ．ｉｄｌ　４１０は、名前空間「Ｆｏｏ．Ｂａｒ」における少なくとも１つの型、及び、名前空間「Ｆｏｏ．Ｑｕｕｘ」における少なくとも２つの型を含む。

【００５４】

Ｆｏｏ．ｍｅｔａｄａｔａ　４１８は、Ｆｏｏ．ｉｄｌ　４０２に少なくとも部分的に基づいた機械読み取り可能なメタデータファイルを表す。説明は省略するが、本発明に係るいくつかの実施形態において、Ｆｏｏ．ｍｅｔａｄａｔａ　４１８は、Ｆｏｏ．ｉｄｌ　４０２からコンパイラによって生成され得る。Ｆｏｏ．ｉｄｌ　４０２と同様に、Ｆｏｏ．ｍｅｔａｄａｔａ　４１８は、メタデータファイルにネイティブなフォーマットの型の記述、Ｆｏｏ．Ｂａｒ．Ｔｙｐｅ１　４２０、Ｆｏｏ．Ｂａｒ．Ｔｙｐｅ２　４２２、及びＦｏｏ．Ｂａｒ．Ｔｙｐｅ３　４２４を含む。Ｂａｒ．ｍｅｔａｄａｔａ　４２６もまた、Ｂａｒ．ｉｄｌ　４１０に少なくとも部分的に基づいた機械読み取り可能なメタデータファイルを表す。Ｆｏｏ．ｍｅｔａｄａｔａ　４１８のケースのように、Ｂａｒ．ｍｅｔａｄａｔａ　４２６は、型の記述、Ｆｏｏ．Ｂａｒ．Ｔｙｐｅ４　４２８、Ｆｏｏ．Ｑｕｕｘ．Ｔｙｐｅ１　４３０、及びＦｏｏ．Ｑｕｕｘ．Ｔｙｐｅ３　４３２を含む。

【００５５】

図４に含まれるのは、マージ・モジュール４３４である。マージ・モジュール４３４は

、１つ以上の入力ファイルを受け取ると、判定基準及び／又は規則のセットに基づいて１つ以上の出力ファイルを生成し得る。例えば入力ファイルを検索する入力ディレクトリ、出力ファイルを置く出力ディレクトリ、適用する検査のレベル、どの名前空間階層レベルで統合及び／又は分割するか、どのようにして出力ファイルに命名するか、出力ファイルをいくつ生成するか、どの深度でメタデータファイルが生成されるべきか、等を指定する入力コマンド及び／又は規則が、マージ・モジュール４３４に適用され得る。判定基準及び／又は規則は、任意の適切な手法、例えば１つ以上の規則を含むコマンド・ライン及び／又は「ｍａｋｅｆｉｌｅ」によって、マージ・モジュール４３４に与えられ得る。

【００５６】

この例では、Ｆｏｏ．ｍｅｔａｄａｔａ　４１８及びＢａｒ．ｍｅｔａｄａｔａ　４２６が、マージ・モジュール４３４への入力である。マージ・モジュール４３４は、入力ファイルをオペランド解析し、指示どおりに出力ファイルを生成する。ここで、マージ・モジュール４３４は、２つの出力ファイル、Ｆｏｏ．Ｂａｒ．ｍｅｔａｄａｔａ　４３６及びＦｏｏ．Ｑｕｕｘ．ｍｅｔａｄａｔａ　４３８を生成する。Ｆｏｏ．Ｂａｒ．ｍｅｔａｄａｔａ　４３６は、「Ｆｏｏ．Ｂａｒ」の名前空間に関連する型（例えばＦｏｏ．Ｂａｒ．Ｔｙｐｅ１　４４０、Ｆｏｏ．Ｂａｒ．Ｔｙｐｅ２　４４２、Ｆｏｏ．Ｂａｒ．Ｔｙｐｅ３　４４４、及びＦｏｏ．Ｂａｒ．Ｔｙｐｅ４　４４６）を含む。同様に、Ｆｏｏ．Ｑｕｕｘ．ｍｅｔａｄａｔａ　４３８は、「Ｆｏｏ．Ｑｕｕｘ」の名前空間に関連する型（例えばＦｏｏ．Ｑｕｕｘ．Ｔｙｐｅ１　４４８、Ｆｏｏ．Ｑｕｕｘ．Ｔｙｐｅ２　４５０）を含む。上記の例におけるように、Ｆｏｏ．Ｂａｒ．ｍｅｔａｄａｔａ　４３６及びＦｏｏ．Ｑｕｕｘ．ｍｅｔａｄａｔａ　４３８は、場合によっては関連付けられた入力ファイルから再編成及び／又は再グループ化されたデータを含む、機械読み取り可能なメタデータファイルを表す。したがって、マージ・モジュール４３４は、複数のファイルからの内容を分析し、判定基準のセットに従って内容を再構築し得る。

【００５７】

本発明に係るいくつかの実施形態において、ファイルの命名規則が、抽象型解決を可能にするために利用され得る。例えば名前空間階層が命名規則として使用され得る。図４において、ファイルＦｏｏ．Ｂａｒ．ｍｅｔａｄａｔａ　４３６及びＦｏｏ．Ｑｕｕｘ．ｍｅｔａｄａｔａ　４３８は、それらのファイル名に、関連付けられた名前空間階層（例えばそれぞれ、「Ｆｏｏ．Ｂａｒ」及び「Ｆｏｏ．Ｑｕｕｘ」）を含む。この命名規則を利用することにより、型情報は、特定のファイル名ではなく、それに関連付けられた名前空間に基づいて、検索されることが可能である。あるいは、又は加えて、出力ファイルは、複数の名前空間階層レベルのデータを含み得る。図４において、Ｆｏｏ．Ｂａｒ．ｍｅｔａｄａｔａ　４３６は、「Ｆｏｏ．Ｂａｒ」の名前空間にデータを含むものとして示されている。しかしながら、Ｆｏｏ．Ｂａｒ．ｍｅｔａｄａｔａ　４３６はまた、「Ｆｏｏ．Ｂａｒ．Ｌｅｖｅｌ１．Ｌｅｖｅｌ２．Ｌｅｖｅｌ３」という名前空間階層レベルに存在する型情報を含むように構成され得る。したがって、命名規則は、ファイル内に含まれ得る最も高いレベルの名前空間を示す。

【００５８】

加えて、適切な型を格納している格納場所が特定されるまで、図３において説明されたような検索アルゴリズムが、異なる階層レベルの名前空間（及び関連付けられたファイル）を横断的に走査するために用いられ得る。同一の名前空間階層レベルにおける全ての型情報を同一のファイル内に存在するように構成することにより、特定のレベルに関連付けられた情報は、情報を利用するアプリケーションに影響を及ぼさずにファイルからファイルへ移動させられることが可能となる。名前空間階層に関連付けられた情報は、任意の適切な手法で分割され得る。例えば複数の階層レベルが同一のファイル内に存在する（例えばＦｏｏ．Ｂａｒ、Ｆｏｏ．Ｂａｒ．Ｌｅｖｅｌ１、及びＦｏｏ．Ｂａｒ．Ｌｅｖｅｌ２がすべて、「Ｆｏｏ．Ｂａｒ．ｍｅｔａｄａｔａ」と命名されたファイル内に存在する）ようにすることも可能であるし、又は、各ファイルがそれ独自の関連付けられたファイルを有する（例えばＦｏｏ．Ｂａｒのレベルにおける情報が、「Ｆｏｏ．Ｂａｒ．ｍｅｔａ

ｄａｔａ」内に存在し、Ｆｏｏ．Ｂａｒ．Ｌｅｖｅｌ１における情報が、「Ｆｏｏ．Ｂａｒ．Ｌｅｖｅｌ１．ｍｅｔａｄａｔａ」ファイル内に存在する）ようにすることも可能である、といった具合である。情報がそれに関連付けられた名前空間階層に従って置かれる場合、型解決は、特定のファイル名への依存を除くことによって、抽象化され得る。

【００５９】

ここで図５を見ると、本発明の１つ以上の実施形態に係る方法を実施するための一連の処理ステップを説明したフローチャートが示されている。この方法は、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、又はそれらの組み合わせによって実行され得る。本発明に係る少なくともいくつかの実施形態において、この方法の実施態様は、図１Ａのマージ・モジュール１１６のような、１つ以上のコンピューティング・デバイス上で実行中の、適切に構成された１つ以上のソフトウェア・モジュールによって実現され得る。

【００６０】

ステップ５０２は、１つ以上の出力ファイルを生成することに関連付けられた１つ以上の入力判定基準を受信する。本発明に係るいくつかの実施形態において、当該入力判定基準は、情報をどのようにして１つ以上の出力ファイルへと分割するかを指定する規則を含み得る。例えば当該入力判定基準は、出力ファイルに一緒にグループ化する１つ以上の名前空間階層レベルを指定し得る。あるいは、又は加えて、当該入力判定基準は、例えばどのファイル及び／又はどのディレクトリから情報を引き出すかを指定することによって、どこで情報を見つけるかを記述し得る。

【００６１】

１つ以上の入力判定基準を受信することに応答して、ステップ５０４は、１つ以上のソフトウェア・インターフェースと関連付けられた情報を含む１つ以上の入力ファイルを受信する。当該情報は、抽象型システムを使用するソフトウェア・インターフェースの記述、オブジェクト指向の関連付け、方法、プロパティ、関数ポインタ、入力及び／又は出力パラメータ、型システム情報、等といった、任意の適切なタイプの情報とすることが可能である。本発明に係る実施形態において、型情報は、名前空間の階層情報を含み得る。

【００６２】

１つ以上の入力ファイルを受信することに応答して、ステップ５０６は、１つ以上の入力判定基準に少なくとも部分的に基づいて情報を分析する。これは、どの名前空間階層の型情報が各ファイル内に存在するかを決定するために入力ファイルを分析することを含み得る。

【００６３】

情報を分析することに応答して、ステップ５０８は、当該情報を含む少なくとも１つの出力ファイルを生成する。本発明に係るいくつかの実施形態において、当該出力ファイルは、上述した入力判定基準に少なくとも部分的に基づいて異なるファイルに再分割された情報を含み得る。当該出力ファイルは、任意の適切な手法で構成されることができ、そのような手法の例は、上述されたとおりである。例えば当該出力ファイルは、メタデータファイルに含まれる名前空間階層に基づいた命名規則を用いて命名されたメタデータファイルとして構成され得る。あるいは、又は加えて、当該出力ファイルは、複数のレベルの名前空間階層を含むように構成され得る。名前空間階層情報を出力ファイル中に一緒にグループ化することによって、アプリケーションは、特定のファイル名ではなく、名前空間階層レベルに基づいて情報を検索することができる。これは、アプリケーションに影響を及ぼさない形で情報の分割を柔軟に実行することを可能にする。アプリケーションがどの名前空間階層レベルに特定の型が存在するのかを知っている限り、どのように名前空間階層レベルがグループ化されるかに関わらず、関連付けられた型情報の情報格納場所が出力ファイル中で特定され得る。

【００６４】

柔軟性の高い型記述情報の格納方法を説明したので、以下の説明においては、本発明の１つ以上の実施形態に係る例示的なシステムの説明を考慮する。

【００６５】
例示的なシステム
図６は、上述したさまざまな実施形態を実現するために使用され得る例示的なコンピューティング・デバイス６００を示す。コンピューティング・デバイス６００は、例えば図１Ａ及び図１Ｂのコンピューティング・デバイス１０２ａ及び／又は１０２ｂ、又は任意の他の適切なコンピューティング・デバイスとすることが可能である。
【００６６】
コンピューティング・デバイス６００は、１つ以上のプロセッサ又は処理ユニット６０２、１つ以上のメモリ及び／又はストレージ・コンポーネント６０４、１つ以上の入力／出力（Ｉ／Ｏ）デバイス６０６、及び、さまざまなコンポーネント及びデバイスに互いに通信することを可能にさせるバス６０８を含む。バス６０８は、メモリ・バス又はメモリ・コントローラ、周辺機器用バス、アクセラレイティッド・グラフィックス・ポート、及び、さまざまなバス・アーキテクチャのうちのいずれかを使用するプロセッサ又はローカル・バスを含む、いくつかのタイプのバス構造のうちのいずれかの１つ以上を表す。バス６０８は、有線及び／又は無線バスを含み得る。
【００６７】
メモリ／ストレージ・コンポーネント６０４は、１つ以上のコンピュータ・ハードウェア記憶媒体を表す。コンポーネント６０４は、揮発性媒体（例えばランダム・アクセス・メモリ（ＲＡＭ））及び／又は不揮発性媒体（例えば読み出し専用メモリ（ＲＯＭ）、フラッシュ・メモリ、光学ディスク、磁気ディスク、等）を含み得る。コンポーネント６０４は、固定媒体（例えばＲＡＭ、ＲＯＭ、固定ハード・ドライブ、等）だけでなく、取り外し可能な媒体（例えばフラッシュ・メモリ・ドライブ、リムーバブル・ハードドライブ、光学ディスク、等）を含み得る。
【００６８】
１つ以上の入力／出力デバイス６０６は、ユーザによるコンピューティング・デバイス６００へのコマンド及び情報の入力を可能にし、また、ユーザ及び／又は他のコンポーネント又はデバイスへの情報の提示を可能にする。入力デバイスの例は、キーボード、カーソル操作用デバイス（例えばマウス）、マイクロフォン、スキャナ、等を含む。出力デバイスの例は、ディスプレイ装置（例えばモニタ又はプロジェクタ）、スピーカー、プリンタ、ネットワーク・カード、等を含む。
【００６９】
さまざまな技法が、ソフトウェア又はプログラム・モジュールの一般的なコンテキストで、本明細書において説明され得る。一般的に、ソフトウェアは、特定のタスクを実行するか、又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、等を含む。これらのモジュール及び技法の実現は、何らかの形態のコンピュータ読み取り可能な媒体上に記憶されるか、又はそのような媒体によって伝送され得る。コンピュータ読み取り可能な媒体は、コンピューティング・デバイスによってアクセス可能な、任意の利用可能な単数の媒体又は複数の媒体とすることが可能である。限定ではなく例として、コンピュータ読み取り可能な媒体は、「コンピュータ読み取り可能な記憶媒体」を備え得る。
【００７０】
「コンピュータ読み取り可能な記憶媒体」は、コンピュータ読み取り可能な命令、データ構造、プログラムモジュール、又は他のデータ、といった情報の記憶のための、任意の方法又は技術において実現される、揮発性及び不揮発性の、取り外し可能及び取り外し不可能な、ハードウェア媒体を含む。コンピュータ読み取り可能なハードウェア記憶媒体は、ＲＡＭ、ＲＯＭ、ＥＥＰＲＯＭ、フラッシュ・メモリ、又は他のメモリ技術、ＣＤ－ＲＯＭ、デジタル多用途ディスク（ＤＶＤ）、又は他の光学ストレージ、磁気カセット、磁気テープ、磁気ディスクストレージ、又は他の磁気ストレージデバイス、又は、所望の情報を記憶するために使用可能かつコンピュータによってアクセス可能な任意の他の媒体を含むが、これらに限定されない。

【００７１】

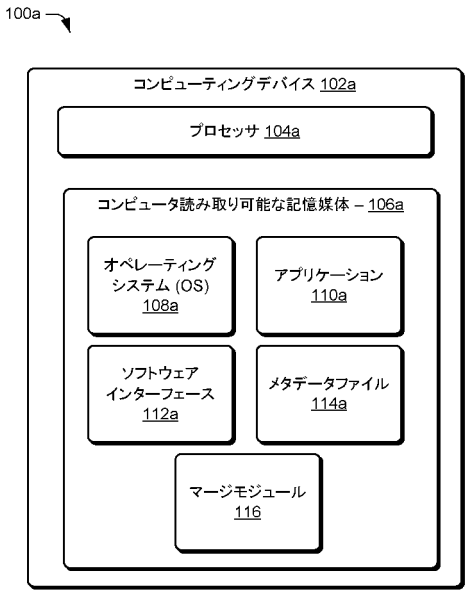
結論

さまざまな実施形態は、複数の型システム間の型解決を抽象化するための能力を提供する。少なくとも１つの型が、プログラムに基づいてアクセス可能な１つ以上のファイルに記述され得る。いくつかの実施形態において、異なる型システムを使用するアプリケーションは、型の記述が存在する場所の知識なしに、少なくとも１つの型システムの型に、プログラムに基づいてアクセスし、その型を解決することができる。あるいは、又は加えて、プログラムに基づいてアクセス可能な１つ以上のファイルに含まれる型の記述は分析され、型の記述に少なくとも部分的に基づいて、プログラムに基づいてアクセス可能な１つ以上の新たなファイルへと再構築されることができる。

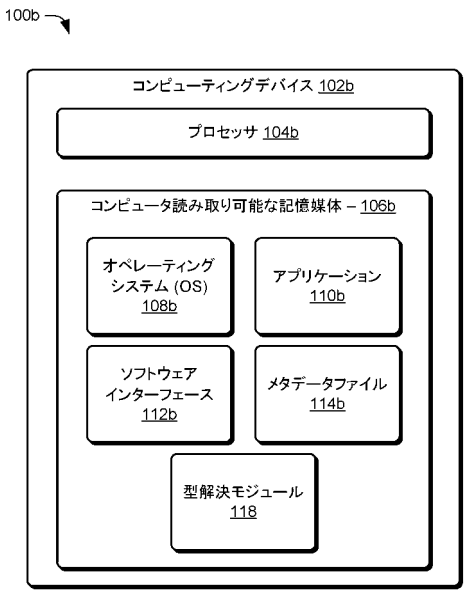
【００７２】

主題は、構造上の特徴及び／又は方法の動作特有の表現で説明されたが、添付の請求項において定義される主題は、必ずしも上述した特定の特徴又は動作に限定されるものではないことが理解されるべきである。そうではなく、上述した特定の特徴及び動作は、請求項を実現する例示的な形態として開示されている。

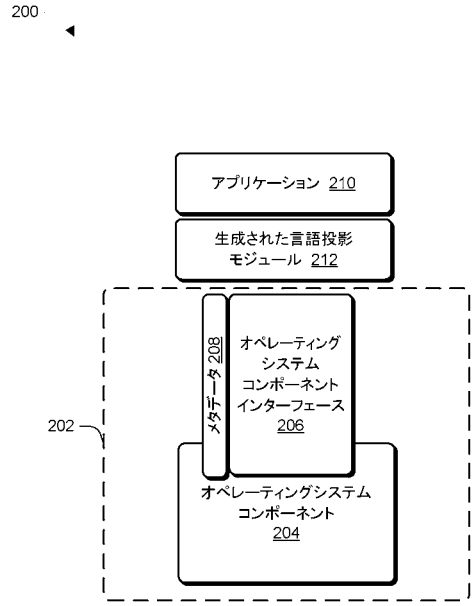
【図１Ａ】



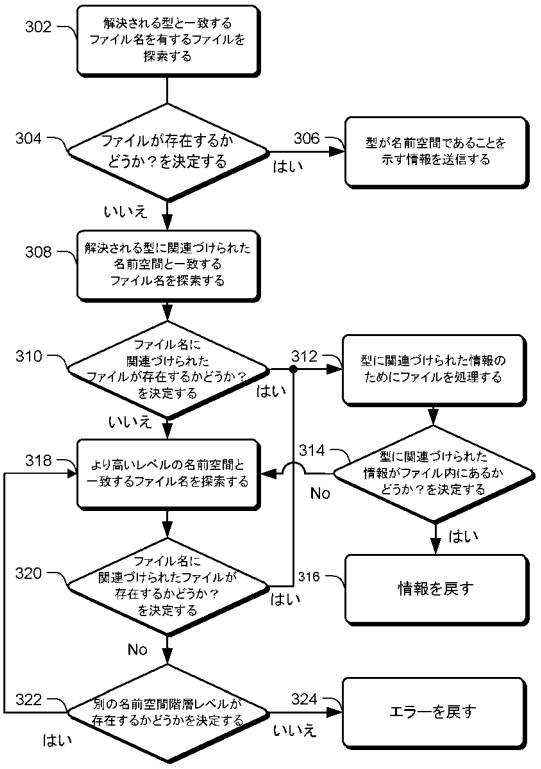
【図１Ｂ】



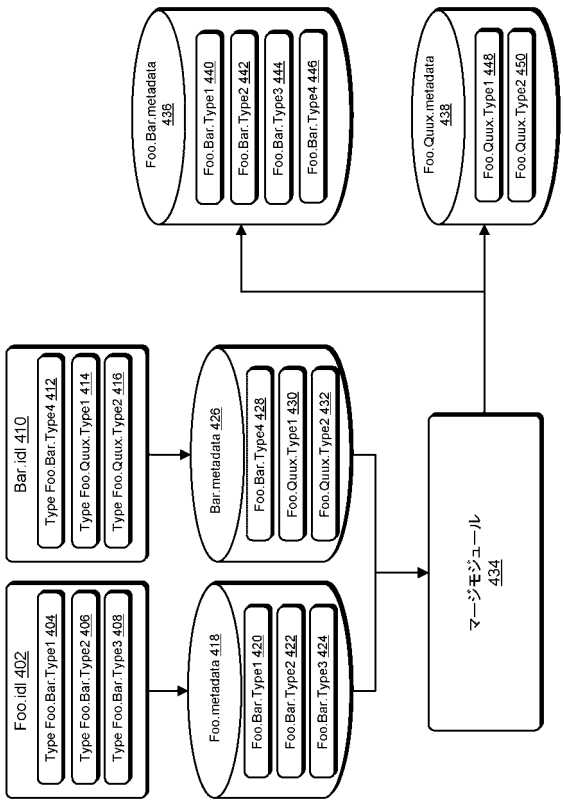
【図２】


200
アプリケーション 210
生成された言語投影
モジュール 212
202
メタデータ 208
オペレーティング
システム
コンポーネント
インターフェース
206
オペレーティングシステム
コンポーネント
204

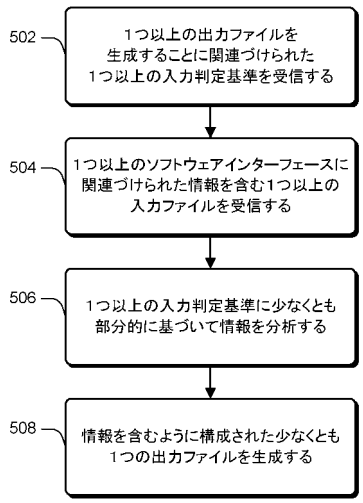
【図３】


302
解決される型と一致する
ファイル名を有するファイルを
探索する
304
ファイルが存在するか
どうか？を決定する
306
はい
型が名前空間であることを
示す情報を送信する
いいえ
308
解決される型に関連づけられた
名前空間と一致する
ファイル名を探索する
310
ファイル名に
関連づけられた
ファイルが存在するかどうか？
を決定する
312
はい
型に関連づけられた情報の
ためにファイルを処理する
いいえ
318
より高いレベルの名前空間と
一致するファイル名を探索する
314
型に関連づけられた
情報がファイル内にあるか
どうか？を決定する
No
はい
320
ファイル名に
関連づけられたファイルが
存在するかどうか？
を決定する
316
はい
情報を戻す
No
322
別の名前空間階層レベルが
存在するかどうかを決定する
324
はい
いいえ
エラーを戻す

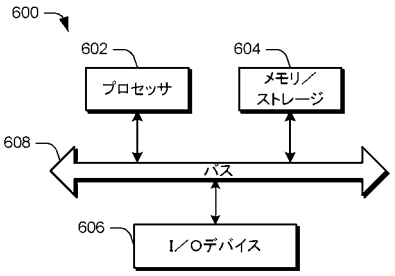
【図４】


Foo.Bar.metadata
436
Foo.Bar.Type1 440
Foo.Bar.Type2 442
Foo.Bar.Type3 444
Foo.Bar.Type4 446
Foo.Quux.metadata
438
Foo.Quux.Type1 448
Foo.Quux.Type2 450
Bar.idl 410
Type Foo.Bar.Type4 412
Type Foo.Quux.Type1 414
Type Foo.Quux.Type2 416
Bar.metadata 426
Foo.Bar.Type4 428
Foo.Quux.Type1 430
Foo.Quux.Type2 432
Foo.idl 402
Type Foo.Bar.Type1 404
Type Foo.Bar.Type2 406
Type Foo.Bar.Type3 408
Foo.metadata 418
Foo.Bar.Type1 420
Foo.Bar.Type2 422
Foo.Bar.Type3 424
マージモジュール
434

【図５】


502
1つ以上の出力ファイルを
生成することに関連づけられた
1つ以上の入力判定基準を受信する
504
1つ以上のソフトウェアインターフェースに
関連づけられた情報を含む1つ以上の
入力ファイルを受信する
506
1つ以上の入力判定基準に少なくとも
部分的に基づいて情報を分析する
508
情報を含むように構成された少なくとも
1つの出力ファイルを生成する

【図６】


600
602
プロセッサ
604
メモリ／
ストレージ
608
バス
606
I／Oデバイス

フロントページの続き

(72)発明者 オスターマン，ローレンス，ダブリュー．
アメリカ合衆国 ９８０５２－６３９９ ワシントン州 レッドモンド ワン マイクロソフト ウェイ マイクロソフト コーポレーション エルシーエー－インターナショナル パテンツ 内
(72)発明者 ピアソン，ハロルド，エル．
アメリカ合衆国 ９８０５２－６３９９ ワシントン州 レッドモンド ワン マイクロソフト ウェイ マイクロソフト コーポレーション エルシーエー－インターナショナル パテンツ 内
(72)発明者 オミヤ，エリオット，エイチ．
アメリカ合衆国 ９８０５２－６３９９ ワシントン州 レッドモンド ワン マイクロソフト ウェイ マイクロソフト コーポレーション エルシーエー－インターナショナル パテンツ 内
(72)発明者 ロヴェル，マーティン，エス．
アメリカ合衆国 ９８０５２－６３９９ ワシントン州 レッドモンド ワン マイクロソフト ウェイ マイクロソフト コーポレーション エルシーエー－インターナショナル パテンツ 内
(72)発明者 プラクリヤ，マヘシュ
アメリカ合衆国 ９８０５２－６３９９ ワシントン州 レッドモンド ワン マイクロソフト ウェイ マイクロソフト コーポレーション エルシーエー－インターナショナル パテンツ 内
(72)発明者 ロウ，スティーヴン，シー．
アメリカ合衆国 ９８０５２－６３９９ ワシントン州 レッドモンド ワン マイクロソフト ウェイ マイクロソフト コーポレーション エルシーエー－インターナショナル パテンツ 内
(72)発明者 バスー，タッサドゥク，エイチ．
アメリカ合衆国 ９８０５２－６３９９ ワシントン州 レッドモンド ワン マイクロソフト ウェイ マイクロソフト コーポレーション エルシーエー－インターナショナル パテンツ 内
(72)発明者 ウロダルチュク，ロバート，エー．
アメリカ合衆国 ９８０５２－６３９９ ワシントン州 レッドモンド ワン マイクロソフト ウェイ マイクロソフト コーポレーション エルシーエー－インターナショナル パテンツ 内
(72)発明者 ズオン，ウエイ
アメリカ合衆国 ９８０５２－６３９９ ワシントン州 レッドモンド ワン マイクロソフト ウェイ マイクロソフト コーポレーション エルシーエー－インターナショナル パテンツ 内
(72)発明者 ワドゥワ，ネーラージ，エヌ．
アメリカ合衆国 ９８０５２－６３９９ ワシントン州 レッドモンド ワン マイクロソフト ウェイ マイクロソフト コーポレーション エルシーエー－インターナショナル パテンツ 内
(72)発明者 ソルカル，シャケール，アイ．
アメリカ合衆国 ９８０５２－６３９９ ワシントン州 レッドモンド ワン マイクロソフト ウェイ マイクロソフト コーポレーション エルシーエー－インターナショナル パテンツ 内
(72)発明者 アクシオンキン，マイケル
アメリカ合衆国 ９８０５２－６３９９ ワシントン州 レッドモンド ワン マイクロソフト ウェイ マイクロソフト コーポレーション エルシーエー－インターナショナル パテンツ 内

Ｆターム(参考) 5B376 BC13 BC14 BC26 BC57 BC79 DA28 EA01 FA01 FA25