(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)特許出願公表番号
特表2011-508317
(P2011-508317A)
(43)公表日　平成23年3月10日(2011.3.10)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 9/44	(2006.01)	G06F	9/06	620A	5B376

審査請求 未請求　予備審査請求 未請求　（全 21 頁）

(21)出願番号　特願2010-539586 (P2010-539586)
(86)(22)出願日　平成20年11月26日 (2008.11.26)
(85)翻訳文提出日　平成22年6月21日 (2010.6.21)
(86)国際出願番号　PCT/US2008/084994
(87)国際公開番号　WO2009/085521
(87)国際公開日　平成21年7月9日 (2009.7.9)
(31)優先権主張番号　11/963,346
(32)優先日　平成19年12月21日 (2007.12.21)
(33)優先権主張国　米国 (US)

(71)出願人　500046438
マイクロソフト　コーポレーション
アメリカ合衆国　ワシントン州　９８０５２－６３９９　レッドモンド　ワン　マイクロソフト　ウェイ
(74)代理人　100077481
弁理士　谷　義一
(74)代理人　100088915
弁理士　阿部　和夫
(72)発明者　マイケル　バーネット
アメリカ合衆国　９８０５２　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　インターナショナル　パテンツ内

最終頁に続く

(54)【発明の名称】コードエラーを減らすためのコントラクトプログラミング

(57)【要約】
一実施形態において、コンピュータシステムは、アプリケーションＡＰＩを拡大するためのＡＰＩ（ａｐｐｌｉｃａｔｉｏｎ　ｐｒｏｇｒａｍｍｉｎｇ　ｉｎｔｅｒｆａｃｅ）を提供する。コンピュータシステムは、第１のプログラミング言語で記述されたコントラクトＡＰＩからのコントラクトを用いてアプリケーションＡＰＩを拡大するというユーザの意図を示す、第２のプログラミング言語で記述されたソフトウェアコードを受け取る。ソフトウェアコードは、コントラクトＡＰＩへの参照を含む。コントラクトは、アプリケーションＡＰＩの適切な使用を示すアサーションを含む。コンピュータシステムは、ソフトウェアコード内の参照に従ってコントラクトＡＰＩの一部にアクセスし、受け取ったソフトウェアコードおよびコントラクトＡＰＩの参照された部分を、ソフトウェアコードの中間言語（ＩＬ）バージョンにコンパイルする。ＩＬバージョンは、第１のプログラミング言語および第２のプログラミング言語の両方に共通の中間言語におけるものである。ＩＬバージョンは、アプリケーションＡＰＩの適切な使用を示すアサーションを

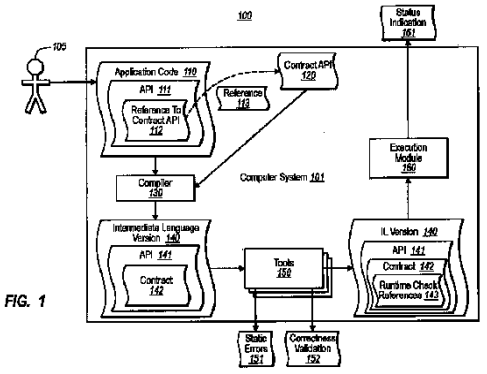


FIG. 1

【特許請求の範囲】
【請求項１】
第２のプログラミング言語で記述されたアプリケーションＡＰＩを拡大するための、第１のプログラミング言語で記述されたコントラクトＡＰＩ（ａｐｐｌｉｃａｔｉｏｎ ｐｒｏｇｒａｍｍｉｎｇ ｉｎｔｅｒｆａｃｅ）をコンピュータ可読媒体上に有する１つまたは複数のコンピュータ可読媒体を備える、コンピュータプログラム製品であって、前記アプリケーションＡＰＩは１つまたは複数のコントラクトを含み、前記コンピュータプログラム製品はコンピュータ実行可能命令を備え、前記コンピュータ実行可能命令は、コンピューティングシステムの１つまたは複数のプロセッサにより実行されるときに前記コンピュータシステムに方法を実施させ、前記方法は、
前記第１のプログラミング言語で記述された前記コントラクトＡＰＩからのコントラクトを用いて前記アプリケーションＡＰＩを拡大するというユーザの意図を示す、前記第２のプログラミング言語で記述されたソフトウェアコードを受け取る動作であって、前記ソフトウェアコードは前記コントラクトＡＰＩへの参照を含み、前記コントラクトは前記アプリケーションＡＰＩの適切な使用を示す１つまたは複数のアサーションを含む動作と、
前記ソフトウェアコード内の前記参照に従って、前記コントラクトＡＰＩの一部にアクセスする動作と、
前記受け取ったソフトウェアコードおよび前記コントラクトＡＰＩの前記参照された部分を前記受け取ったソフトウェアコードの中間言語バージョンにコンパイルする動作であって、前記中間言語バージョンは前記第１のプログラミング言語および前記第２のプログラミング言語の両方に共通の中間言語におけるものであり、前記中間言語バージョンは前記アプリケーションＡＰＩの適切な使用を示す１つまたは複数のアサーションを含む動作と
を含むことを特徴とするコンピュータプログラム製品。
【請求項２】
１つまたは複数のソフトウェアツールを用いて前記コンパイルされた中間言語バージョンにアクセスする動作と、
１つまたは複数のランタイムチェックへの１つまたは複数の参照を前記コントラクトの内の少なくとも１つに挿入して、前記ランタイムチェックが前記対応する参照に従ってアプリケーションコードに注入されることを可能にする動作と
をさらに含むこと特徴とする請求項１に記載の方法を実施するよう構成されたＡＰＩ。
【請求項３】
前記挿入されたランタイムチェックを含む前記中間言語バージョンを実行する動作と、
前記コントラクト内の前記アサーションの内の少なくとも１つが前記中間言語バージョンの実行中に違反されたと判定する動作と、
前記判定に基づき、前記アサーションの内の少なくとも１つが違反されたという表示を与える動作と、
をさらに含むこと特徴とする請求項２に記載の方法を実行するよう構成されたＡＰＩ。
【請求項４】
前記挿入されたランタイムチェックを含む前記中間言語バージョンを実行する動作と、
前記コントラクト内の前記アサーションのいずれも前記中間言語バージョンの実行中に違反されなかったと判定する動作と、
前記判定に基づき、前記アサーションのいずれも違反されなかったという表示を与える動作と
をさらに含むこと特徴とする請求項２に記載の方法を実施するよう構成されたＡＰＩ。
【請求項５】
前記アプリケーションＡＰＩのコントラクトの内の少なくとも１つを静的にチェックして、前記コントラクトのアサーションのいずれかが違反されたかどうかを判定する動作
をさらに含むこと特徴とする請求項１に記載の方法を実施するよう構成されたＡＰＩ。
【請求項６】

前記アサーションは、呼び出し側がメソッドを呼び出すことを許可される前に満たされるべき項目を特定する１つまたは複数の前提条件を含むこと特徴とする請求項１に記載の方法を実施するよう構成されたＡＰＩ。
【請求項７】
前記アサーションは、前記メソッドの通常の終了および前記メソッドの例外的な終了の内の少なくとも一方の際、前記メソッドのリターン時に有効である１つまたは複数の保証を特定する１つまたは複数の事後条件を含むこと特徴とする請求項１に記載の方法を実施するよう構成されたＡＰＩ。
【請求項８】
前記アサーションは、ロック宣言の内の少なくとも１つを含むこと特徴とする請求項１に記載の方法を実施するよう構成されたＡＰＩ。
【請求項９】
前記アサーションは、少なくとも１つのチェックされる例外を含むこと特徴とする請求項１に記載の方法を実施するよう構成されたＡＰＩ。
【請求項１０】
前記アサーションは、オブジェクト不変条件の内の少なくとも１つを含むこと特徴とする請求項１に記載の方法を実施するよう構成されたＡＰＩ。
【請求項１１】
前記コントラクトは、前記コントラクトＡＰＩの結果として拡張可能であり、前記拡張性は、新しいメソッドを前記コントラクトＡＰＩに追加することを可能にして、１つまたは複数のクライアントが前記第２のプログラミング言語で記述された前記コントラクト内の前記新しいメソッドを使用することができるように、新しい種類のコントラクトを記述することを可能にすること特徴とする請求項１に記載の方法を実施するよう構成されたＡＰＩ。
【請求項１２】
前記コントラクト内の前記アサーションの内の少なくとも１つは宣言型であること特徴とする請求項１に記載の方法を実施するよう構成されたＡＰＩ。
【請求項１３】
前記拡大されたコントラクトに基づき、前記ソフトウェアアプリケーションのＡＰＩのドキュメンテーションを自動的に生成する動作さらに含むこと特徴とする請求項１に記載の方法を実施するよう構成されたＡＰＩ。
【請求項１４】
前記コントラクトＡＰＩを使用して記述された前記コントラクトは複数のダウンストリームツールに対して維持されること特徴とする請求項１に記載の方法。
【請求項１５】
コンピュータシステムにおいて、アプリケーションＡＰＩを拡大するための方法であって、前記アプリケーションＡＰＩは１つまたは複数のコントラクトを含み、
コントラクトＡＰＩからのコントラクトを用いて前記アプリケーションＡＰＩを拡大するというユーザの意図を示すソフトウェアコードを、コンピュータユーザから受け取る動作であって、前記ソフトウェアコードは前記コントラクトＡＰＩへの参照を含み、前記コントラクトは前記アプリケーションＡＰＩの適切な使用を示す１つまたは複数のアサーションを含む動作と、
前記ソフトウェアコード内の前記参照に従って前記コントラクトＡＰＩの一部にアクセスする動作と、
前記受け取ったソフトウェアコードおよび前記コントラクトＡＰＩの前記参照された部分を、前記アプリケーションＡＰＩの適切な使用を示す１つまたは複数のアサーションを含む前記受け取ったソフトウェアコードの中間言語バージョンにコンパイルする動作と、
前記ソフトウェアコードの前記コンパイルされた中間言語バージョンを前記コンピュータユーザに提供する動作と
を含むことを特徴とする方法。

【請求項１６】
前記アサーションは、前提条件、事後条件、ロック宣言、チェックされる例外、使用プロトコル、およびオブジェクト不変条件の内の１つまたは複数を含むことを特徴とする請求項９に記載の方法。
【請求項１７】
前記コントラクトは、前記コントラクトを完全汎用のプログラミング言語で記述することができるように拡張可能であることを特徴とする請求項９に記載の方法。
【請求項１８】
ＡＰＩコントラクトを修正するためのコンピュータシステムであって、前記システムは、
コンパイラであって、
ソフトウェアアプリケーションコードとどのように相互作用するかを特定する１つまたは複数のアサーションを識別する少なくとも１つのコントラクトを有するＡＰＩを含む、前記ソフトウェアアプリケーションコードを受け取るステップであって、前記コントラクトは１つの値と交換されるべきアプリケーションコードの一部を示す１つまたは複数の参照を含むよう拡張可能であり、前記コントラクトは汎用のプログラム言語で記述されるステップと、
１つまたは複数のアサーションを前記ソフトウェアアプリケーションコード内の対応するメソッドに統合するステップと
を実施するよう構成されたコンパイラと、
リライタであって、
前記コンパイラからバイナリを受け取るステップであって、前記バイナリは少なくとも１つの前記ソフトウェアアプリケーションコードと、前記少なくとも１つのコントラクトとを含むステップと、
前記コントラクト内の前記参照により示される前記アプリケーションコードの一部がある値と交換されるように、前記受け取ったバイナリを書き換えるステップと、
前記書き換えたバイナリを実行可能バイナリとして出力するステップと
を実施するよう構成されたリライタと
を備えることを特徴とするシステム。
【請求項１９】
前記コンパイラは１つまたは複数の前記バイナリを受け取るよう構成されることを特徴とする請求項１８に記載のシステム。
【請求項２０】
前記アサーションは、前提条件、事後条件、ロック宣言、チェックされる例外、使用プロトコル、およびオブジェクト不変条件の内の少なくとも１つであることを特徴とする請求項１８に記載のシステム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、コードエラーを減らすためのコントラクトプログラミングに関する。
【背景技術】
【０００２】
今日の社会においてコンピュータの使用が急増するにつれ、これらのコンピュータ上で稼働するソフトウェアアプリケーションを開発するための方法及びシステムの数も、増加している。ソフトウェアアプリケーションは、使用されているアプリケーションの設計に基づき、多種多様のタスクを実施するために使用することができる。ソフトウェアアプリケーションは一般的には、共に動作して何らかのタイプの所望の最終結果を生成するよう設計された多数の個々のファイルを含む。そのようなソフトウェアアプリケーションを記述ために、ソフトウェア開発者は一般的には、そのアプリケーションに使用するプログラミング言語を決定する。今日、多くのプログラミング言語が使用され、その結果、多くの

ソフトウェアアプリケーションが異なる言語で記述されている。これらの言語は、お互いに互換性が無いことが多く、異なるシンタックス、異なるコンパイル方法、および他の異なるコード要素を使用するものが少なくない。

【０００３】

複数のプログラミング言語の統合および使用をそれらの種々の実装において簡素化するために、マネージドコードを導入して、これらの種々のアプリケーションが相互作用可能であるプラットフォームを提供している。マネージドコードは、とりわけ、アプリケーション開発者がプログラムの属性を特定することを可能にし、それによりそのアプリケーションのある部分を使用しようとする他の開発者は、どのようにすれば最善であるかが分かる。例えば、開発者は、アプリケーションの変数、メソッド、または全体のプロセスに対応する種々の要素を特定することができる。開発者は、例えば、コントラクトまたは仕様において、変数Ｘが特定の型を有する、または特定の値より大であると記述できる。そのようなアプリケーションコントラクトまたは仕様を使用して、コードがコンパイルされる前に、コード内に生成されるエラーの数を減少させることが多い。

【発明の概要】

【発明が解決しようとする課題】

【０００４】

しかしながら、そのようなアプリケーションコントラクトは、特定できるものが非常に制限されている。例えば、ほとんどのアプリケーションコントラクトが、変数の型を特定することに制限される。これは、型のエラーを取り除くことには有益である。しかしながら、そのようなコントラクトを使用することによって、存在し得る多くの他のエラーを防ぐことはできないであろう。

【課題を解決するための手段】

【０００５】

本明細書に説明する実施形態は、アプリケーションＡＰＩを拡大することを対象とする。一実施形態において、コンピュータシステムは、アプリケーションＡＰＩを拡大するためのＡＰＩ（ａｐｐｌｉｃａｔｉｏｎ　ｐｒｏｇｒａｍｍｉｎｇ　ｉｎｔｅｒｆａｃｅ）を提供する。コンピュータシステムは、第１のプログラミング言語で記述されたコントラクトＡＰＩからのコントラクトを用いてアプリケーションＡＰＩを拡大するというユーザの意図を示す、第２のプログラミング言語で記述されたソフトウェアコードを受け取る。ソフトウェアコードは、コントラクトＡＰＩへの参照を含む。コントラクトは、アプリケーションＡＰＩの適切な使用を示すアサーションを含む。コンピュータシステムは、ソフトウェアコード内の参照に従ってコントラクトＡＰＩの一部にアクセスし、受け取ったソフトウェアコードおよびコントラクトＡＰＩの参照された部分を、受け取ったソフトウェアコードの中間言語（ＩＬ）バージョンにコンパイルする。ＩＬバージョンは、第１のプログラミング言語および第２のプログラミング言語の両方に共通の中間言語におけるものである。ＩＬバージョンは、アプリケーションＡＰＩの適切な使用を示すアサーションを含む。

【０００６】

他の実施形態において、コンピュータシステムは、コントラクトＡＰＩからのコントラクトを用いてアプリケーションＡＰＩを拡大するというユーザの意図を示すソフトウェアコードを、コンピュータユーザから受け取る。ソフトウェアコードは、コントラクトＡＰＩへの参照を含む。コントラクトは、アプリケーションＡＰＩの適切な使用を示すアサーションを含む。コンピュータシステムは、ソフトウェアコード内の参照に従ってコントラクトＡＰＩの一部にアクセスし、受け取ったソフトウェアコードおよびコントラクトＡＰＩの参照された部分を、アプリケーションＡＰＩの適切な使用を示す１つまたは複数のアサーションを含む、受け取ったソフトウェアコードの中間言語バージョンにコンパイルし、ソフトウェアコードのコンパイルされた中間言語バージョンをコンピュータユーザに提供する。

【０００７】

本「発明の概要」は、以下「発明を実施するための形態」でさらに述べる概念を選択して簡略化した形式で紹介するために提供するものである。本「発明の概要」は、請求の主題の重要な特徴または主要な特徴を特定することを意図しておらず、請求の主題の範囲を決定する際の援助として使用されることも意図していない。

【図面の簡単な説明】

【０００８】

本発明の実施形態の上記および他の利点および特徴をさらに明確にするために、本発明の実施形態のさらに特定の説明を、添付の図面を参照して与える。当然のことながら、これらの図面は本発明の典型的な実施形態のみを示し、従って、その範囲を制限するものとみなされるべきではない。本発明を、添付の図面を使用して、追加的な特異性および詳細と共に記述し説明する。

【０００９】

【図１】本発明の実施形態が、ＡＰＩコントラクトの修正および／または生成を含めて動作することができるコンピュータアーキテクチャを示す図である。

【図２】アプリケーションＡＰＩを拡大するための方法の一例のフローチャートである。

【図３】アプリケーションＡＰＩを拡大するための方法の代替例のフローチャートである。

【図４】修正されたコントラクトの例示的実施形態を示す図である。

【図５】生成されたコントラクトの例示的実施形態を示す図である。

【図６】コントラクトのＡＰＩを使用して実行可能バイナリを生成するための例示的システムを示す図である。

【発明を実施するための形態】

【００１０】

本明細書に説明する実施形態は、アプリケーションＡＰＩを拡大することを対象とする。一実施形態において、コンピュータシステムは、アプリケーションＡＰＩを拡大するためのＡＰＩ（ａｐｐｌｉｃａｔｉｏｎ　ｐｒｏｇｒａｍｍｉｎｇ　ｉｎｔｅｒｆａｃｅ）を提供する。コンピュータシステムは、第１のプログラミング言語で記述されたコントラクトＡＰＩからのコントラクトを用いてアプリケーションＡＰＩを拡大するというユーザの意図を示す、第２のプログラミング言語で記述されたソフトウェアコードを受け取る。ソフトウェアコードは、コントラクトＡＰＩへの参照を含む。コントラクトは、アプリケーションＡＰＩの適切な使用を示すアサーションを含む。コンピュータシステムは、ソフトウェアコード内の参照に従ってコントラクトＡＰＩの一部にアクセスし、受け取ったソフトウェアコードおよびコントラクトＡＰＩの参照された部分を、受け取ったソフトウェアコードの中間言語（ＩＬ）バージョンにコンパイルする。ＩＬバージョンは、第１のプログラミング言語および第２のプログラミング言語の両方に共通の中間言語におけるものである。ＩＬバージョンは、アプリケーションＡＰＩの適切な使用を示すアサーションを含む。

【００１１】

他の実施形態において、コンピュータシステムは、コントラクトＡＰＩからのコントラクトを用いてアプリケーションＡＰＩを拡大するというユーザの意図を示すソフトウェアコードを、コンピュータユーザから受け取る。ソフトウェアコードは、コントラクトＡＰＩへの参照を含む。コントラクトは、アプリケーションＡＰＩの適切な使用を示すアサーションを含む。コンピュータシステムは、ソフトウェアコード内の参照に従ってコントラクトＡＰＩの一部にアクセスし、受け取ったソフトウェアコードおよびコントラクトＡＰＩの参照された部分を、アプリケーションＡＰＩの適切な使用を示す１つまたは複数のアサーションを含む、受け取ったソフトウェアコードの中間言語バージョンにコンパイルし、ソフトウェアコードのコンパイルされた中間言語バージョンをコンピュータユーザに提供する。

【００１２】

他の実施形態において、コンピュータシステムは、ソフトウェアアプリケーションが第

１のプログラミング言語で記述された少なくとも１つのコントラクトを含むよう構成されたＡＰＩを有すると判定する。コンピュータシステムは、第２のプログラミング言語で記述されたソフトウェアコードを使用してソフトウェアアプリケーションの実行可能コントラクトを生成するというユーザの意図を示すコンピュータユーザから、第２のプログラミング言語で記述されたソフトウェアコードを受け取る。

【００１３】

コンピュータシステムは、ソフトウェアコードを、第２のプログラミング言語で記述されたソフトウェアコードを受け取るよう構成されたマネージドフレームワークコンポーネントに送り、第１のプログラミング言語および第２のプログラミング言語の両方に共通のランタイム言語でソフトウェアコードのバージョンを出力する。コンピュータシステムは、第２のプログラミング言語で記述されたソフトウェアコードの中間言語バージョンを受け取り、ソフトウェアコードの受け取った中間言語バージョンに基づき、ソフトウェアアプリケーションの実行可能コントラクトを生成する。

【００１４】

本発明の実施形態は、以下にさらに詳細に検討するように、コンピュータハードウェアを含む専用コンピュータまたは汎用コンピュータを含むか、または利用できる。本発明の範囲内の実施形態にはまた、コンピュータ実行可能命令および／またはデータ構造を搬送または記憶するための、物理的コンピュータ可読媒体および他のコンピュータ可読媒体を含む。そのようなコンピュータ可読媒体は、汎用コンピュータシステムまたは専用コンピュータシステムによりアクセス可能な任意の利用可能な媒体とすることが可能である。コンピュータ実行可能命令を記憶するコンピュータ可読媒体は、物理的記憶媒体である。コンピュータ実行可能命令を搬送するコンピュータ可読媒体は、伝送媒体である。従って、限定ではなく例として、本発明の実施形態は、少なくとも２つの明らかに異なるタイプのコンピュータ可読媒体、すなはち、物理的記憶媒体および伝送媒体を含むことが可能である。

【００１５】

物理的記憶媒体には、ＲＡＭ、ＲＯＭ、ＥＥＰＲＯＭ、ＣＤ－ＲＯＭもしくは他の光ディスク記憶装置、磁気ディクス記憶装置もしくは他の磁気記憶装置、または、コンピュータ実行可能命令またはデータ構造の形式の所望のプログラムコード手段を記憶するために使用することが可能であり、かつ、汎用コンピュータまたは専用コンピュータによりアクセス可能である任意の他の媒体を含む。

【００１６】

「ネットワーク」は、コンピュータシステムおよび／またはモジュールおよび／または他の電子デバイスの間で電子データの移送を可能にする１つまたは複数のデータリンクとして定義される。情報が、ネットワークまたは別の通信接続（有線、無線、または有線もしくは無線の組み合わせのいずれか）を介してコンピュータに転送または提供されるとき、コンピュータは、適切にその接続を伝送媒体と見なす。伝送媒体は、コンピュータ実行可能命令またはデータ構造の形式の所望のプログラムコード手段を搬送または移送するために使用することが可能であり、かつ、汎用コンピュータまたは専用コンピュータによりアクセス可能である、ネットワークおよび／またはデータリンクを含むことが可能である。上記のものの組み合わせもまた、コンピュータ可読媒体の範囲に含むべきである。

【００１７】

しかしながら、当然ことながら、種々のコンピュータシステムコンポーネントに届くとすぐに、コンピュータ実行可能命令またはデータ構造の形式のプログラムコード手段を、伝送媒体から物理的記憶媒体に自動的に転送することができる。例えば、ネットワークまたはデータリンクを介して受け取ったコンピュータ実行可能命令またはデータ構造を、ネットワークインターフェースカード内のＲＡＭにバッファリングすることが可能であり、そして最終的には、コンピュータシステムＲＡＭへ、および／またはコンピュータシステムにおいて低不揮発性物理的記憶媒体へ転送することが可能である。従って、当然のことながら、物理的記憶媒体は、伝送媒体をも（または、さらには主として）利用するコンピ

ュータシステムコンポーネントに含むことが可能である。

【００１８】

コンピュータ実行可能命令は、例えば、汎用コンピュータ、専用コンピュータ、または専用処理装置に、ある特定の機能または機能の群を実施させる命令およびデータを含む。コンピュータ実行可能命令は、例えば、バイナリ、アセンブリ言語といった変換形式の命令、またはさらにはソースコードであることができる。本発明を、構造的特徴および／または方法論的な行為に特有の言語で説明しているたが、当然のことながら、添付の請求項に定義される主題は必ずしも上述の特徴または行為に限定されない。むしろ、説明する特徴および行為は、請求項を実装する例示の形式として開示するものである。

【００１９】

当業者は理解するであろうが、本発明は、パーソナルコンピュータ、デスクトップコンピュータ、ラップトップコンピュータ、メッセージプロセッサ、ハンドヘルドデバイス、マルチプロセッサシステム、マイクロプロセッサベースの家庭用電化製品またはプログラム可能な家庭用電化製品、ネットワークＰＣ、ミニコンピュータ、メインフレームコンピュータ、携帯電話、ＰＤＡ、ページャ、ルータ、スイッチ等を含む、多数のタイプのコンピュータシステム構成を有するネットワークコンピュータ環境において実践することができる。本発明はまた、ネットワークを介してリンクされる（有線データリンク、無線データリンク、または有線データリンクおよび無線データリンクの組み合わせのいずれかにより）、ローカルコンピュータシステムおよびリモートコンピュータシステムの両方がタスクを実施する分散システム環境において実践することができる。分散システム環境においては、プログラムモジュールを、ローカルのメモリ記憶装置およびリモートのメモリ記憶装置の両方に配置できる。

【００２０】

図１は、本発明の原理を採用できるコンピュータアーキテクチャ１００を示す。コンピュータアーキテクチャ１００は、コンピュータシステム１０１を含む。いくつかの実施形態において、コンピュータシステム１０１はアプリケーションコード１１０を含む。アプリケーションコード１１０は、任意のサイズまたは複雑性を持つ任意のタイプのソフトウェアアプリケーションコードであることができ、任意のタイプの使用に対して設計される。アプリケーションコード１１０は、ソフトウェアアプリケーションの一部であることができ、またはそれ自身の内部にソフトウェアアプリケーションを含むことができる。アプリケーションコード１１０は、任意のプログラム言語で記述された任意のタイプのコードを含むことができる。コード１１０には、メソッド、プロセス、リンクされたライブラリ、コードの個々の行、またはコンピュータシステムと相互作用するために使用できる任意の他のタイプの情報を含むことができる。

【００２１】

いくつかの実施形態において、アプリケーションコード１１０はまた、ＡＰＩ（ａｐｐｌｉｃａｔｉｏｎ ｐｒｏｇｒａｍｍｉｎｇ ｉｎｔｅｒｆａｃｅ）１１１を含む。ＡＰＩは、ユーザまたは開発者がソフトウェアアプリケーションの機能性にアクセスすることを可能にする、任意のタイプのインターフェースであることができる。例えば、アプリケーション開発者は一般的には、開発者のアプリケーションの特徴を使用するソフトウェアアプリケーションをサードパーティの開発者が記述できるよう、ＡＰＩを記述する。いくつかの場合において、以下でさらに詳細に説明するように、アプリケーションのＡＰＩをどのように使用するかを示すＡＰＩのコントラクト（例えば、コントラクト１４２）内に、アサーションを与えると有利であろう。コントラクト内の方法を使用して、開発者は、ソフトウェアアプリケーションのＡＰＩとどのように相互作用するかをより良く理解することができ、その結果、高品質でバグの無いコードを作成することができる。

【００２２】

コンピュータシステム１０１におけるコントラクトＡＰＩ１２０を、参照１１３を受け取るよう構成することができ、参照は、コントラクトＡＰＩがＡＰＩコントラクトを修正または生成するために使用されることを示す。例えば、ＡＰＩ１１１は、コントラクトＡ

ＰＩ１２０がＡＰＩ１１１のコントラクトの１つを修正するために使用されることを示す、コントラクトＡＰＩへの参照１１２を含むことができる。コントラクトは、アプリケーションＡＰＩ１１１がどのように使用されるかを示すアサーションを含むことができる。アサーションは、内部でアプリケーションＡＰＩ１１１が動作するのに最も適している境界を定義することができる。アサーションの例には、前提条件、事後条件、特定のアサーション、不変条件、ロック宣言およびチェックされる例外が含まれる。アサーションについては、以下でさらに詳細に説明する。

【００２３】

ユーザ１０５は、ソフトウェア開発者、クライアント／エンドユーザ、システム管理者または任意の他のタイプのコンピュータユーザを含む、任意のタイプのコンピュータユーザであることができる。いくつかの場合において、ユーザ１０５は、アプリケーションコード１１０をコンパイラ１３０に送ることができる。コンパイラ１３０は、ソフトウェアコードの何らかの部分をコンパイルするよう構成された任意のタイプのソフトウェアアプリケーションであることができる。コンパイラ１３０は、中間形式で、何らかのタイプの中間言語（ＩＬ）で、コードを出力することができ、または実行可能な機械可読コードを出力することができる。いくつかの実施形態において、コンパイラ１３０は、マネージドフレームワークの一部であることができ、かつ、マネージドフレームワークコンポーネントと称することができる。

【００２４】

本明細書において使用するとき、マネージドフレームワークは、複数の異なるプログラミング言語で記述されたコードファイルを、統一された方法でマネージドフレームワーク（例えば、Ｍｉｃｒｏｓｏｆｔ社の共通言語ランタイム、Ｊａｖａ仮想マシン、またはＬＩＳＰ（ｌｉｓｔ　ｐｒｏｃｅｓｓｉｎｇ　ｌａｎｇｕａｇｅ）インタプリタ）により動作することが可能な中間言語命令に縮小する（またはコンパイルする）ことを可能にする手段であることができる。これにより、複数のプログラミング言語間の相互運用が可能になる。従って、各マネージドフレームワークは、そのフレームワークと互換性がある言語および互換性が無い言語を有する。

【００２５】

いくつかの実施形態において、コンピュータシステム１０１を、アプリケーションコード１１０が、所与のマネージドフレームワークまたはコンパイラ（例えば、コンパイラ１３０）と互換性のある言語で記述されているかどうかを判定するよう構成することができる。さらに、コンピュータシステム１０１に記憶され、コンパイラ１３０によりアクセスされるソフトウェアコードを、その言語の互換性を判定するよう処理することができる。従って、少なくともいくつかの実施形態において、コードがユーザ１０５により受け取られたかどうか、またはコンピュータシステム１０１上のデータストアにおいてアクセスされたかどうかにかかわらず、アプリケーションコード１１０がコンパイラ１３０と互換性があると判定することができる。

【００２６】

コンパイラ１３０は、アプリケーションコード１１０を処理するよう構成され、かつ、ＡＰＩ１４１およびＡＰＩコントラクト１４２を含む中間言語（ＩＬ）バージョンのソフトウェアコード１４０をツール１５０に送ることができる。いくつかの実施形態において、ツール１５０には、様々な機能を実施するよう構成された様々なソフトウェアツールを含むことができる。ツール１５０には、静的エラーがＩＬバージョン１４０に存在するかどうかを判定するツール、ＩＬバージョンの正確性の検証を実施するツール、または、ランタイムチェックへの参照をコントラクトに挿入して、対応する参照に従ってランタイムチェックをアプリケーションコードに注入することを可能にするツールを含むことができる。多くの他のツールをツール１５０に含むことができる。ツール１５０を、ＩＬバージョン１４０をＡＰＩ１４１およびランタイムチェック参照１４３を有するコントラクト１４２と共に実行モジュール１６０に送るよう構成することができる。実行モジュール１６０は、ＩＬバージョン１４０を実行するよう構成された任意のタイプのソフトウェアメソ

ッド、プロセス、またはアプリケーションであることができる。実行モジュール１６０は、実行により失敗またはコントラクト違反が生じたこと、または、あるいは何の失敗もコントラクト違反も起きなかったことを示す状態表示１６１を出力できる。この処理について、図２および３に関連して以下でさらに詳細に説明する。

【００２７】

図２は、アプリケーションＡＰＩを拡大するための方法２００のフローチャートを示す。方法２００を、ここでは図１、４および６の構成要素およびデータを頻繁に参照して説明する。

【００２８】

方法２００は、第１のプログラミング言語で記述されたコントラクトＡＰＩからのコントラクトを用いてアプリケーションＡＰＩを拡大するというユーザの意図を示す、第２のプログラミング言語で記述されたソフトウェアコードを受け取る動作を含み、ソフトウェアコードはコントラクトＡＰＩへの参照を含み、コントラクトはアプリケーションＡＰＩの適切な使用を示す１つまたは複数のアサーションを含む（動作２１０）。例えば、コンパイラ１３０は、第１のプログラミング言語（例えば、ＶＢ（Ｖｉｓｕａｌ　Ｂａｓｉｃ）、Ｃ＃等）で記述されたコントラクトＡＰＩ１２０からのコントラクト（例えば、コントラクト１４２）を用いてアプリケーションＡＰＩ１１１を拡大するというユーザ１０５の意図を示す、第２のプログラミング言語（例えば、ＶＢ（Ｖｉｓｕａｌ　Ｂａｓｉｃ）、Ｃ＃等）で記述されたアプリケーションコード１１０（用語ソフトウェアコードおよびアプリケーションコードは、本明細書において同義的に使用される）を受け取ることができる。アプリケーションコード１１０は、コントラクトＡＰＩへの参照１１２を含み、いくつかの場合において、参照１１３を用いてコントラクトＡＰＩ１２０と通信、またはコントラクトＡＰＩ１２０を参照することができる。コントラクト（例えば、コントラクト１４２）は、アプリケーションＡＰＩ１１１の適切な使用を示すアサーションを含む。コントラクトＡＰＩ１２０の一般的性質により、コントラクトは、任意のタイプのコード、メソッド、プロセス等を含むよう十分に拡張可能であることができる。さらに、第１のプログラミング言語は、任意の一般使用のプログラミング言語であることができる。コントラクトの拡張性は、新しいメソッドがコントラクトＡＰＩに追加されることを可能にすることができ、従って、新しい種類のコントラクトが記述されることを可能にする。クライアントは、第２のプログラミング言語で記述されたコントラクト内のそのような新しいメソッドを使用することが可能である。

【００２９】

いくつかの場合において、アサーションは、そのＡＰＩを使用するアプリケーションを記述する開発者が使用できる。例えば、サードパーティの開発者がアプリケーションＡを記述し、アプリケーションＢに提供された機能を使用したい場合、該開発者は、アプリケーションＢのＡＰＩを使用してアプリケーションＢの機能にアクセスするであろう。いくつかの実施形態において、アプリケーションＢのＡＰＩ内のコントラクトには、アプリケーションＢのＡＰＩを使用する際に満たされるべき前提条件、事後条件といったアサーションを含むことができる。アサーションは、内部でアプリケーションＢが動作するのに最も適している境界を定義することができる。

【００３０】

アサーションには、前提条件、事後条件、特定のアサーション、不変条件、ロック宣言およびチェックされる例外を含むがこれに限らない。前提条件は、本明細書で使用する場合、機能の呼び出し側が満たすべき要件を示す。前提条件は、機能へのエントリ時において真であることが要求される。事後条件は、本明細書で使用する場合、機能の実装が、終了（潜在的には、成功的終了および例外的な終了の両方）時に実現させる保証を示す。事後条件は、機能から出た時に真でなければならない。前提条件および事後条件の両方とも、メソッドの宣言の一部として含むことができる。

【００３１】

特定のアサーションは、本明細書で使用する場合、プログラムのコードの一部として含

まれ、サニティチェックまたはプログラム論理の明確化作用として働く。特定のアサーションはまた、静的ベリファイアへの補助として働くこともできる。特定のアサーションは、これらが含まれるプログラムの制御フロー内の適切なポイントで真でなければならない。不変条件は、本明細書で使用する場合、ある型に対する全ての動作に渡って真である条件を示す。いくつかの実施形態において、不変条件は、型の宣言の一部として含まれ、データ上のアサーションとして動作できる。外部のクライアント（人またはプログラム）が無効な期間を何ら発見することができない場合には、いくつかのウィンドウのプログラムの実行中に、不変条件を一時的に無効にすることができる。ロック宣言は、本明細書で使用する場合、並行処理下のプログラムの挙動に関する分析の支援に使用されるステートメントである。ロック宣言は、どのロックがどのデータを保護するかを特定するよう設計される。前提条件および事後条件を使用して、どのロックを保持すべきまたは保持すべきでないかを表すことができる。チェックされる例外は、本明細書で使用する場合、メソッドから投入することが可能な例外の組を制約する。チェックされる例外は、事後条件に似ているが、例外的な退出を対象とする。チェックされる例外を、メソッドの宣言の一部として含むことができる。

【００３２】

一般に、アサーションは、同様の機能を実施するよう設計される。すなはち、記述された条件が実行時の特定のポイントにおいて真でない場合、プログラムが正しくないことを示すよう設計される。従って、アサーションを、特定のポイントにおけるプログラムの記述または挙動の文書化に使用すること、プログラム内のエラーの分析または検出の補助に使用される計算可能な式として使用すること、プログラムの異なる部分間のコントラクト（アサーションポイントに通じるコードが満たさなければならない要件、またはアサーションポイントに続くコードが依存することができる保証）として使用すること、または任意の他の多数の使用（例えば、アサーションが信頼できるまたは正しいことが証明されていると仮定した場合、コンパイラを介したコードの生成）が可能である。いくつかの場合において、コントラクト内のアサーションを、コードがコントラクトに対して合致するかどうかを静的プログラムベリファイアがチェックするために、またはバグ発見ツールがコード内の起こり得るエラーを見つけるために、使用できる。

【００３３】

従って、アプリケーションコード１１０は、既存のコントラクトを有する、またはコントラクトを何ら有しないＡＰＩ１１１を含むことができる。いくつかの場合において、ＡＰＩ１４１は、複数のコントラクトを含むことができ、それぞれのコントラクトが、異なるメソッドに対応する複数のアサーションを有することができる。いくつかの場合において、ＡＰＩコントラクト１４２には、呼び出し側がメソッドを呼び出すことを許可される前に満たされるべき項目を特定する、１つまたは複数の前提条件を含むことができる。従って、そのような前提条件を有するメソッドを使用するために、前提条件は、メソッドが呼び出される前に満たされなければならない。言いかえれば、前提条件が満たされない場合、そのメソッドを呼び出すことはできない。（または、メソッドが呼び出される場合、そのメソッドは失敗する）。例えば、前提条件が違反されるときに、構築プロセスで実施されるチェックを介してメソッドの呼び出しを阻止することは有利であろう。さらに、ユーザが完全な構築プロセスを使用できないときのための潜在的なバックアップとして、ランライム時に前提条件を評価することができる。前提条件が満たされない場合、例外を投入すること、または何らかの他のタイプの挙動（例えば、エスカレーションポリシーをトリガすること）を実施することにより、メソッドの実行は失敗する可能性がある。

【００３４】

同様に、ＡＰＩコントラクト１４２は、メソッドのリターン時に有効である１つまたは複数の保証を特定する、１つまたは複数の事後条件を含むことができる。従って、少なくともいくつかの場合において、事後条件がメソッドのリターン時に満たされない場合、コントラクトは、それ以上ＡＰＩの使用を許可しない。ＡＰＩコントラクト１４２はまた、ある型に対する全ての動作に渡って真である条件を示す、１つまたは複数のオブジェクト

不変条件を含むことができる。コントラクト１４２内のアサーションはいずれも宣言型であり得ることに留意すべきである。従って、宣言型アサーションは、実装の詳細のそれぞれを記載せずに、アサーションの機能を記載することができる。

【００３５】

方法２００は、ソフトウェアコード内の参照に従って、コントラクトＡＰＩの一部にアクセスする動作を含む（動作２２０）。例えば、コンパイラ１３０は、アプリケーションコード１１０内の参照１１２に従って、コントラクトＡＰＩ１２０の一部にアクセスすることができる。従って、いくつかの場合において、コンパイラ１３０は、参照１１２を使用して、中間言語バージョン１４０をコンパイルする際に使用するためにコントラクトＡＰＩ１２０の一部にアクセスすることができる。

【００３６】

方法２００はまた、受け取ったソフトウェアコードおよびコントラクトＡＰＩの参照された部分を、受け取ったソフトウェアコードの中間言語バージョンにコンパイルする動作を含み、中間言語バージョンは第１のプログラミング言語および第２のプログラミング言語の両方に共通の中間言語におけるものであり、中間言語バージョンはアプリケーションＡＰＩの適切な使用を示す１つまたは複数のアサーションを含む（動作２３０）。例えば、コンパイラ１３０は、アプリケーションコード１１０およびコントラクトＡＰＩ１２０の参照された部分をアプリケーションコード１１０の中間言語バージョン１４０にコンパイルすることができる。中間言語バージョン１４０は、第１のプログラミング言語（例えば、ＶＢ、Ｃ＃等）および第２のプログラミング言語（例えば、ＶＢ、Ｃ＃等）の両方に共通の中間言語（ＩＬ）におけるものであることができる。ＩＬバージョン１４０は、アプリケーションＡＰＩ１４１の適切な使用を示すアサーションを含むことができる。

【００３７】

いくつかの実施形態において、例えば、アプリケーションコード１１０をＶｉｓｕａｌ Ｂａｓｉｃで記述することができ、コントラクトＡＰＩをＣ＃で記述することができ、コンパイラ１３０を、Ｃ＃で記述されたコントラクトＡＰＩ１２０と互換性のあるＶｉｓｕａｌ Ｂａｓｉｃのコードの ＩＬバージョンを出力するよう構成することができる。いくつかの場合において、両言語が、マネージドフレームワークの一部であることができる。そのような場合、両言語が、共通のマネージドフレームワークの一部であるため、１つの言語で記述されたコードを使用して、別の言語で記述されたコードを修正することができる。これは、コントラクトＡＰＩ１２０がユーザ１０５には未知のプログラミング言語で記述されており、それにもかかわらずユーザ１０５が、場合によっては種々のアサーションを追加することにより、アプリケーションＡＰＩのコントラクトを修正することを所望する状況においては有利であろう。

【００３８】

いくつかの実施形態において、コンピュータシステム１０１は、様々な異なるソフトウェアツールを含むことができるツール１５０を含む。例えば、ツール１５０には、コンパイルされた中間言語バージョン１４０にアクセスし、かつ、１つまたは複数のランタイムチェック参照１４３をコントラクト１４２に挿入するよう構成されるツールを含むことができる。ランタイムチェック参照１４３を、ランタイムチェックを挿入できるアプリケーションコード内の場所を識別するプレースホルダとして使用することができる。そのようなツールを、さらに、ランタイムチェック参照を有するコントラクト１４２を含むＩＬバージョン１４０を実行するよう、かつ、コントラクト１４２内の少なくとも１つのアサーションが、ＩＬバージョンの実行中に違反されたと判定するよう構成することができる。さらに、そのような判定に基づき、ツールを、アサーションの内の少なくとも１つが違反されたという表示を与えるよう構成することができる（例えば、状態表示１６１において）。他の場合において、ツール１５０の１つを、挿入されたランタイムチェック参照を有するコントラクト１４２を含むＩＬバージョン１４０を実行するよう、かつ、コントラクト１４２内のアサーションのいずれもＩＬバージョンの実行中に違反されなかったと判定するよう構成することができる。そして、そのような判定に基づき、ツールを、アサーシ

ョンのいずれも違反されなかったという表示を与えるよう構成することができる（例えば、状態表示１６１において）。

【００３９】

いくつかの実施形態において、ツール１５０には、ＩＬバージョン１４０に対して静的分析を実施して、コントラクト１４２のアサーションが満たされたかどうかを判定するよう構成されたツールを含むことができる。いくつかの場合において、静的分析が、コントラクトのアサーションが満たされたことを判定する場合、ユーザにそれを通知することができる（例えば、正確性の検証１５１において）。同様に、１つまたは複数のアサーションが満たされなかった場合、通知（例えば、静的エラー１５１）を、そのような情報を含むように構成することができる。いくつかの場合において、ユーザには、単にコントラクトが修正されたことを通知することができる。

【００４０】

図４に説明する例示的実施形態において、コントラクトを、上述した方法でアサーションを用いて拡大することができる。アプリケーションコード４０６は、手つかずの元のままの形式（４０５）であり、ＡＰＩ４０８および、前提条件、事後条件、不変条件、または他の要素といった種々のアサーションに対応できる複数のアサーション１、２および３を有するコントラクト４０９を含む。コントラクトＡＰＩ１２０の一部と共にコンパイルされた後、コントラクト４０９を拡大するために使用されるアプリケーションコード４０６を、ユーザ１０５が入力できる。アプリケーションコード４２６の結果として得られた修正バージョン４２５は、ＡＰＩ４２８および、拡大された（または新しい）アサーション１Ａ、３および４を有するコントラクト４２９を含む。アサーション１Ａは、元のアサーション１の修正バージョンであり、アサーション３は同一であり、アサーション４は追加されたものである。従って、コントラクト拡大には、既存のアサーションを変更すること、アサーション（例えば、要素２）を削除すること、および／または、アサーションをコントラクトに追加することを含むことができる。

【００４１】

図３は、アプリケーションＡＰＩを拡大するための代替の方法３００のフローチャートを示す。方法３００を、ここでは図１、５および６の構成要素およびデータを頻繁に参照して説明する。

【００４２】

方法３００は、コントラクトＡＰＩからのコントラクトを用いてアプリケーションＡＰＩを拡大するというユーザの意図を示すソフトウェアコードを、コンピュータユーザから受け取る動作を含み、ソフトウェアコードはコントラクトＡＰＩへの参照を含み、コントラクトはアプリケーションＡＰＩの適切な使用を示す１つまたは複数のアサーションを含む（動作３１０）。例えば、コントラクトＡＰＩ１２０からのコントラクト１４２を用いてアプリケーションＡＰＩ１１１を拡大するというユーザ１０５の意図を示すアプリケーションコード１１０を、コンピュータユーザ１０５から受け取ることができる。アプリケーションコード１１０は、コントラクトＡＰＩ１１２への参照を含む。コントラクト１４２は、アプリケーションＡＰＩ１４１の適切な使用を示す１つまたは複数のアサーションを含む。上記で示したように、コントラクト１４２は、呼び出し側がメソッドを呼び出すことを許可される前に満たされるべき項目を特定する、１つまたは複数の前提条件を含むことができる。加えてまたはあるいは、コントラクト１４２は、メソッドのリターン時に有効である１つまたは複数の保証を特定する１つまたは複数の事後条件を含むことができる。不変条件、特定のアサーション、ロック宣言、チェックされる例外および他のアサーションを、ＡＰＩコントラクト１４２内に個別にまたは集合的に含むこともできる。

【００４３】

方法３００はまた、ソフトウェアコード内の参照に従ってコントラクトＡＰＩの一部にアクセスする動作を含む（動作３２０）。例えば、コンパイラ１３０は、アプリケーションコード１１０内の参照１１２に従って、コントラクトＡＰＩ１２０の一部にアクセスすることができる。

【００４４】
方法３００は、受け取ったソフトウェアコードおよびコントラクトＡＰＩの参照された部分を、アプリケーションＡＰＩの適切な使用を示す１つまたは複数のアサーションを含む受け取ったソフトウェアコードの中間言語バージョンにコンパイルする動作（動作３３０）を含む。例えば、コンパイラ１３０は、アプリケーションコード１１０およびコントラクトＡＰＩ１２０の参照された部分を、ＡＰＩ１４１の適切な使用を示す種々のアサーションを用いてＡＰＩ１４１およびコントラクト１４２を含むＩＬバージョン１４０にコンパイルすることができる。いくつかの場合において、コンパイラ１３０は、ソフトウェアコードを受け取り、かつ、マネージドフレームワークに含まれる複数の異なるプログラミング言語に共通するアプリケーションコード１４０の中間言語（ＩＬ）バージョンを出力するよう構成される、マネージドフレームワークの一部であることができる。例えば、ソフトウェアコード１１０がＣ＃で記述され、かつ、コントラクト１４２がＶｉｓｕａｌ Ｂａｓｉｃで記述されている場合、コンパイラ１３０を、Ｖｉｓｕａｌ Ｂａｓｉｃで記述されたコントラクト１４２と互換性のあるＣ＃コードのＩＬバージョンを出力するよう構成することができる。

【００４５】
方法３００は、ソフトウェアコードのコンパイルされた中間言語バージョンをコンピュータユーザに提供する動作を含む（動作３４０）。例えば、コンピュータシステム１０１は、コンパイルされたＩＬバージョン１４０をコンピュータユーザ１０５に提供することができる。いくつかの場合において、ＩＬバージョンを、ツール１５０の後のアプリケーションのユーザ１０５に提供することができる。いくつかの場合において、コンピュータシステム１０１は、ＩＬバージョン１４０をツール１５０に直接提供することができる。上記で示したように、多くの異なるツールを使用して、エラーチェック（ランタイムまたは静的）、ドキュメンテーション生成、ロギング、または他の特徴からの任意のものを実施することができる。例えば、ドキュメンテーションを使用して、アプリケーションＡＰＩを用いてコントラクトをどのように使用するかをそのようなユーザに通知することができる。そのようなドキュメンテーションは、ツール１５０により自動的に生成することができる。

【００４６】
コントラクト１４２は、既にＡＰＩ１１１の一部である既存のコントラクトの拡大されたバージョンを表しことができる。加えてまたはあるいは、コントラクト１４２は、コントラクトＡＰＩ１２０を使用して生成された新しいコントラクトを表すことができる。図５にて説明する例示的実施形態において、ＡＰＩ５２８のコントラクト５２９は、上述のコントラクト拡大と同様の方法で生成することができる。言い換えると、コントラクト生成は、情報を含有しない（すなわち、アサーションを含有しない）コントラクトを拡大することと等価であることができる。本例において、アプリケーションコード５０６は、手つかずの元のままの形式（５０５）であり、ＡＰＩ５０８を含む。アプリケーションコード１１０は、ＩＬバージョンに変換された後、生成されたコントラクトを有するアプリケーションＡＰＩを作成するために使用することができる（５２５）。その結果として生成されるアプリケーションコード５２６は、ＡＰＩ５２８および、前提条件、事後条件、不変条件、または他の要素といった種々のアサーションに対応できる、アサーション１、２および３を有する生成されたコントラクト５２９を含む。従って、ＡＰＩ５２８を、コントラクト内に含まれる種々のアサーションの任意または全てのものを含む、生成されたコントラクト５２９を含みかつ使用するよう構成することができる。

【００４７】
いくつかの実施形態には、アプリケーションコード、１つまたは複数のコントラクト、ライブラリ参照、および他の要素を含む実行可能バイナリを生成するためのシステムを含むことができる。例えば、図６にて説明するように、コントラクト６０６を有するＡＰＩ６０５を含むソフトウェアアプリケーションコードを受け取るようコンパイラ６１５を構成することができ、コントラクト６０６は、ソフトウェアアプリケーションコードとどの

ように相互作用するかを特定する１つまたは複数のアサーション６０７（そのうちのあるものは条件付きである）を識別する。コントラクト６０６は、拡張可能であり、ある値と交換されるアプリケーションコードの一部を示す１つまたは複数の参照を含むことができる。さらに、コントラクト６０６を、任意の汎用プログラミング言語で記述することができる。

【００４８】

コンパイラ６１５は、１つまたは複数のアサーションを、任意の受け取ったバイナリ６１０と共に、ソフトウェアアプリケーションコードの内の対応するメソッドに統合するよう構成することができる。リライタ６２５は、ソフトウェアアプリケーションコード６２１、コントラクト６２２およびライブラリ参照６２３を含む、コンパイルされたバイナリ６２０をコンパイラ６１５から受け取るよう、かつ、コントラクト内の参照により示されるアプリケーションコードの一部を中間（ランタイム）言語内の値と交換するように受け取ったバイナリをリライトするよう構成することができる。

【００４９】

いくつかの場合において、バイナリを、ランタイムチェックおよびそれに続く読み込みの両方のために書き換えることができる。さらに、この新しい（書き換えられた）バイナリを、書き換えられたソフトウェアを使用するソフトウェアの新しい部分をコンパイルするために入力として使用することができる。書き換えられたバイナリを、実行可能バイナリ６３０として出力することができる。リライタ６２５は、書き換え時に使用するために、１つまたは複数のライブラリ６２６へ選択的にアクセスすることができる。いくつかの場合において、ソフトウェアコードおよびコントラクトを、別個にコンパイルし、次いで、後にリライタ６１５により結合することができる。上記で示したように、アサーションには、前提条件、事後条件、オブジェクト不変条件、ロック宣言、チェックされる例外、使用プロトコル、または任意の他のタイプのアサーションを含むことができる。

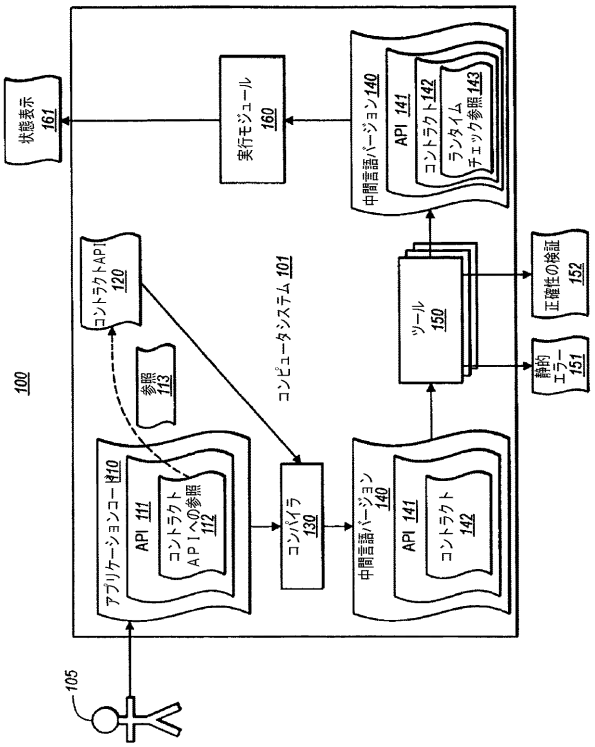
【００５０】

従って、いくつかの実施形態によると、ユーザは、１つの言語でソフトウェアコードを記述し、そのソフトウェアコードを使用して、別の言語で記述されたアプリケーションＡＰＩのコントラクトを拡大することができる。コントラクトは、コントラクトに関連するアプリケーションＡＰＩを適切に使用するために満たされるべき前提条件および事後条件を含む複数のアサーションを含み、これによりプログラミングエラーをより減少させることができる。

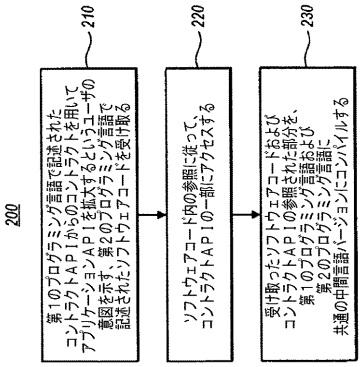
【００５１】

本発明を、その精神または主要な特徴から逸脱することなく他の特定の形式で具現化することができる。説明した実施形態は、あらゆる点において単に例示であり制限するものではないと見なされるべきである。本発明の範囲は、従って、前述の説明よりもむしろ添付の請求項により示される。請求項と等価の意味および範囲内にある全ての変更が、請求項の範囲に包含される。

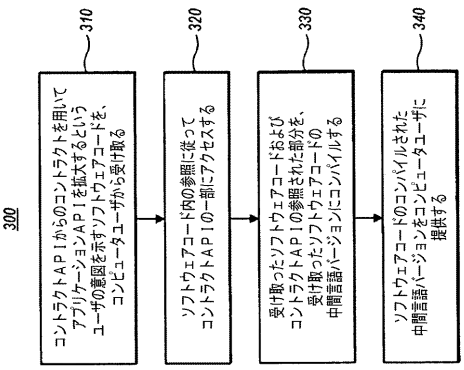
【図１】


100
105
状態表示 161
実行モジュール 160
コントラクトAPI 120
参照 113
コンピュータシステム 101
アプリケーションコード 110
API 111
コントラクトAPIへの参照 112
コンパイラ 130
中間言語バージョン 140
API 141
コントラクト 142
ツール 150
中間言語バージョン 140
API 141
コントラクト 142
ランタイムチェック参照 143
正確性の検証 152
静的エラー 151

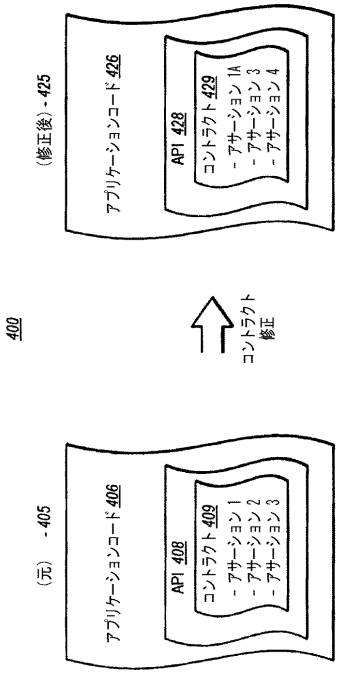
【図２】


200
210
第１のプログラミング言語で記述された
コントラクトＡＰＩからのコントラクトを用いて
アプリケーションＡＰＩを拡大するというユーザの
意図を示す、第２のプログラミング言語で
記述されたソフトウェアコードを受け取る
220
ソフトウェアコード内の参照に従って、
コントラクトＡＰＩの一部にアクセスする
230
受け取ったソフトウェアコードおよび
コントラクトＡＰＩの参照された部分を、
第１のプログラミング言語および
第２のプログラミング言語に
共通の中間言語バージョンにコンパイルする

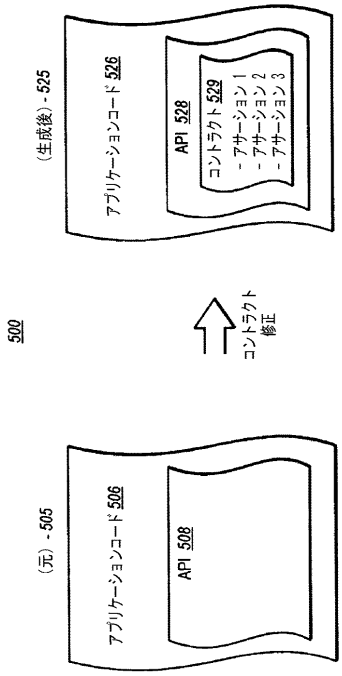
【図３】


300
310
コントラクトＡＰＩからのコントラクトを用いて
アプリケーションＡＰＩを拡大するという
ユーザの意図を示すソフトウェアコードを、
コンピュータユーザから受け取る
320
ソフトウェアコード内の参照に従って
コントラクトＡＰＩの一部にアクセスする
330
受け取ったソフトウェアコードおよび
コントラクトＡＰＩの参照された部分を、
受け取ったソフトウェアコードの
中間言語バージョンにコンパイルする
340
ソフトウェアコードのコンパイルされた
中間言語バージョンをコンピュータユーザに
提供する

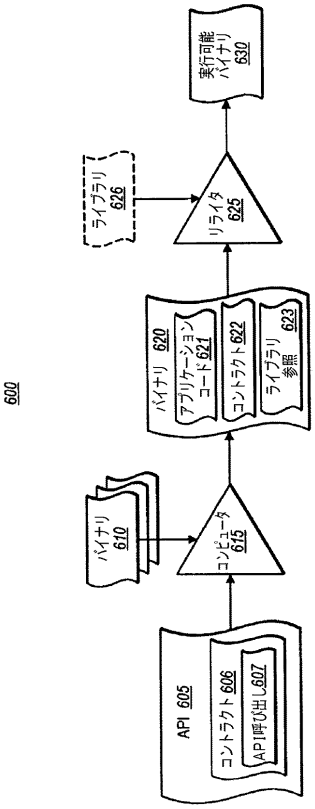
【図４】


400
（元） - 405
アプリケーションコード 406
API 408
コントラクト 409
- アサーション１
- アサーション２
- アサーション３
コントラクト修正
（修正後） - 425
アプリケーションコード 426
API 428
コントラクト 429
- アサーション１A
- アサーション３
- アサーション４

【図５】


500
(元) - 505
アプリケーションコード 506
API 508
コントラクト
修正
(生成後) - 525
アプリケーションコード 526
API 528
コントラクト 529
- アサーション１
- アサーション２
- アサーション３

【図６】


600
実行可能
バイナリ
630
ライブラリ
626
リライタ
625
バイナリ 620
アプリケーション
コード 624
コントラクト 622
ライブラリ
参照 623
バイナリ
610
コンピュータ
615
API 605
コントラクト 606
API呼び出し 607

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2008/084994

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/30(2006.01)i, G06F 9/06(2006.01)i, G06F 11/08(2006.01)i, G06F 9/44(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC : G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models since 1975
Japanese utility models and applications for utility models since 1975

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKIPASS(Kipo Internal), NDSL, Google
keywords: verif*, debug*, error, code, application, contract, API,library, compil*, interpret*, IL, intermediate, language

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US2007/0039010 A1 (GADRE, M.A.) 15 FEBRUARY 2007 See figures 4,5; paragraphs [3][5][14][27][31]~[33]	1-20
X	TRAN, N. et al., 'Design and Implementation of Assertions for the Common Language Infrastructure' In: Proceedings of the IEE Software Engineering, pp.329-336, 27 October 2007. Retrived from the Internet: <http://messagelab.monash.edu.au/Publications/Publication?action=download&upname=managed.pdf> See abstract, pages 1,2.	1-20
A	TRAN, N. et al., 'Managed Assertions for Components Contracts' In: Proceedings of the Integrated Design and Process Technology(IDPT-2003), June 2003. Retrived from the Internet:<http://messagelab.monash.edu.au/Publications/Publication?action=download&upname=ipdt2003.pdf> See abstract, figures 1 & 2; section IV(page 5).	1-20
A	US2005/0102656 A1 (VIEHLAND, R.E. et al.) 12 MAY 2005 See figures 2,5 and their descriptions.	1-20

☒ Further documents are listed in the continuation of Box C. ☒ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents,such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
15 JUNE 2009 (15.06.2009)	**16 JUNE 2009 (16.06.2009)**

Name and mailing address of the ISA/KR	Authorized officer
Korean Intellectual Property Office Government Complex-Daejeon, 139 Seonsa-ro, Seo-gu, Daejeon 302-701, Republic of Korea Facsimile No. 82-42-472-7140	YOON, Hye Sook Telephone No. 82-42-481-8370

Form PCT/ISA/210 (second sheet) (July 2008)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2008/084994

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2004/0261065 A1 (ABRAMS, B. M. et al.) 23 DECEMBER 2004 See figures 2,3; paragraphs [8][30][34].	1-20
A	US 2004/0255268 A1 (MEIJER, E. et al.) 16 DECEMBER 2004 See figures 2,4,8; paragraphs [11][85][86]	1-20
PA	US 2008/0209395 A1 (ERNST, B.) 28 AUGUST 2008 See figure 5; paragraph [45]; claim 1.	1-20

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/US2008/084994

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2007-0039010 A1	15.02.2007	NONE	
US 2005-0102656 A1	12.05.2005	NONE	
US 2004-0261065 A1	23.12.2004	NONE	
US 2004/0255268 A1	16.12.2004	NONE	
US 2008/0209395 A1	28.08.2008	NONE	

フロントページの続き

(81)指定国 AP(BW,GH,GM,KE,LS,MW,MZ,NA,SD,SL,SZ,TZ,UG,ZM,ZW),EA(AM,AZ,BY,KG,KZ,MD,RU,TJ,TM),EP(AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR,GB,GR,HR,HU,IE,IS,IT,LT,LU,LV,MC,MT,NL,NO,PL,PT,RO,SE,SI,SK,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,ML,MR,NE,SN,TD,TG),AE,AG,AL,AM,AO,AT,AU,AZ,BA,BB,BG,BH,BR,BW,BY,BZ,CA,CH,CN,CO,CR,CU,CZ,DE,DK,DM,DO,DZ,EC,EE,EG,ES,FI,GB,GD,GE,GH,GM,GT,HN,HR,HU,ID,IL,IN,IS,JP,KE,KG,KM,KN,KP,KR,KZ,LA,LC,LK,LR,LS,LT,LU,LY,MA,MD,ME,MG,MK,MN,MW,MX,MY,MZ,NA,NG,NI,NO,NZ,OM,PG,PH,PL,PT,RO,RS,RU,SC,SD,SE,SG,SK,SL,SM,ST,SV,SY,TJ,TM,TN,TR,TT,TZ,UA,UG,US,UZ,VC,VN,ZA,ZM,ZW

（特許庁注：以下のものは登録商標）
１．ＪＡＶＡ

(72)発明者 マニュエル エー．ファンドリッチ
アメリカ合衆国 ９８０５２ 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 インターナショナル パテンツ内
(72)発明者 ブライアン エム．グランケマイヤー
アメリカ合衆国 ９８０５２ 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 インターナショナル パテンツ内
(72)発明者 ウルフラム シュルト
アメリカ合衆国 ９８０５２ 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 インターナショナル パテンツ内

Ｆターム(参考) 5B376 BC17 BC23 BC57 BC64 BC69

【要約の続き】
含む。