

(19) 日本国特許庁(JP)

(12) 特 許 公 報(B2)

(11) 特許番号

特許第5669433号
(P5669433)

(45) 発行日 平成27年2月12日(2015.2.12)

(24) 登録日 平成26年12月26日(2014.12.26)

(51) Int.Cl.

F I

G 0 6 F 9/44 (2006.01)

G 0 6 F 9/06 6 2 0 K

請求項の数 11 (全 19 頁)

(21) 出願番号	特願2010-106665 (P2010-106665)	(73) 特許権者	000001007
(22) 出願日	平成22年5月6日(2010.5.6)		キヤノン株式会社
(65) 公開番号	特開2011-237883 (P2011-237883A)		東京都大田区下丸子3丁目30番2号
(43) 公開日	平成23年11月24日(2011.11.24)	(74) 代理人	100076428
審査請求日	平成25年3月5日(2013.3.5)		弁理士 大塚 康德
		(74) 代理人	100112508
			弁理士 高柳 司郎
		(74) 代理人	100115071
			弁理士 大塚 康弘
		(74) 代理人	100116894
			弁理士 木村 秀二
		(74) 代理人	100130409
			弁理士 下山 治
		(74) 代理人	100134175
			弁理士 永川 行光

最終頁に続く

(54) 【発明の名称】 部品結合装置及び部品結合方法

(57) 【特許請求の範囲】

【請求項1】

アプリケーションを構成する部品と外部サービスとを結合する方法であって、
前記外部サービスのインタフェースの情報を取得する取得工程と、
 他の部品と結合された処理対象の部品が他の部品のサービスを利用するためのインタフェースに、前記取得工程で取得した情報が示す外部サービスのインタフェースが適合するか判断する判断工程と、
 前記判断工程によって、前記処理対象の部品が前記他の部品を利用するためのインタフェースに、前記取得工程で取得した情報が示す外部サービスのインタフェースが適合すると判断された場合、前記処理対象の部品が前記外部サービスと結合されるように、前記処理対象の部品と前記他の部品との結合を変更する変更工程と、
 前記外部サービスと結合された部品を実行する実行工程と、
 を備えることを特徴とする方法。

【請求項2】

前記判断工程では、前記処理対象の部品が利用可能な部品を示す設定に基づいて選択された外部サービスのインタフェースが前記インタフェースに適合するか判断することを特徴とする請求項1記載の方法。

【請求項3】

前記判断工程では、前記処理対象の部品が利用可能な外部サービスを外部装置に出力し、前記外部装置で指定された外部サービスのインタフェースが前記インタフェースに適合

10

20

するか判断する

ことを特徴とする請求項 1 記載の方法。

【請求項 4】

前記変更工程では、前記処理対象の部品と前記他の部品との結合の変更として、前記処理対象の部品と結合された他の部品と、該処理対象の部品と結合する候補の部品とが記述された定義を、変更する

ことを特徴とする請求項 1 記載の方法。

【請求項 5】

前記変更工程では、前記処理対象の部品が前記外部サービスと結合され、前記他の部品が結合する候補の部品として記述されるように、前記処理対象の部品と前記他の部品との結合の定義を変更する

ことを特徴とする請求項 1 記載の方法。

【請求項 6】

前記変更工程では、前記処理対象の部品と前記他の部品との結合の変更として、サービスコンポーネントアーキテクチャ（SCA）仕様に従って記述された、前記処理対象の部品と前記他の部品との結合の関係を示す結合定義を変更する

ことを特徴とする請求項 1 記載の方法。

【請求項 7】

前記判断工程では、前記処理対象の部品が前記他の部品のサービスを利用するためのインタフェースのクラスと前記外部サービスのクラスが対応するか、判断する

ことを特徴とする請求項 1 記載の方法。

【請求項 8】

前記判断工程では、優先使用部品として指定された前記外部サービスのインタフェースが、前記他の部品のサービスを利用するためのインタフェースに適合するか、判断する

ことを特徴とする請求項 1 記載の方法。

【請求項 9】

前記処理対象の部品が他の部品のサービスを利用するためのインタフェースは、Java インタフェースクラスに従う

ことを特徴とする請求項 1 記載の方法。

【請求項 10】

アプリケーションを構成する部品と外部サービスとを結合する部品結合装置の制御をコンピュータに機能させるためのプログラムであって、

前記コンピュータに、

前記外部サービスのインタフェースの情報を取得する取得手段と、

他の部品と結合された処理対象の部品が他の部品のサービスを利用するためのインタフェースに、前記取得手段で取得した情報が示す外部サービスのインタフェースが適合するか判断する判断手段と、

前記判断手段によって、前記処理対象の部品が前記他の部品を利用するためのインタフェースに、前記取得手段で取得した情報が示す外部サービスのインタフェースが適合すると判断された場合、前記処理対象の部品が前記外部サービスと結合されるように、前記処理対象の部品と前記他の部品との結合を変更する変更手段と、

前記外部サービスと結合された部品を実行する実行手段と、

して機能させることを特徴とするプログラム。

【請求項 11】

アプリケーションを構成する部品と外部サービスとを結合する部品結合装置であって、

前記外部サービスのインタフェースの情報を取得する取得手段と、

他の部品と結合された処理対象の部品が他の部品のサービスを利用するためのインタフェースに、前記取得手段で取得した情報が示す外部サービスのインタフェースが適合するか判断する判断手段と、

前記判断手段によって、前記処理対象の部品が前記他の部品を利用するためのインタフ

10

20

30

40

50

エースに、前記取得手段で取得した情報が示す外部サービスのインタフェースが適合すると判断された場合、前記処理対象の部品が前記外部サービスと結合されるように、前記処理対象の部品と前記他の部品との結合を変更する変更手段と、

前記外部サービスと結合された部品を実行する実行手段と、
を備えることを特徴とする部品結合装置。

【発明の詳細な説明】

【技術分野】

【0001】

本発明は、ソフトウェア部品を結合してアプリケーションを構築する装置に関し、特に結合させるソフトウェア部品を置換又は変更する技術に関する。

10

【背景技術】

【0002】

従来、ソフトウェア部品（以下、単に部品と称す）又はコンポーネントを組み合わせてアプリケーションを構築する多様な技術が存在する。

【0003】

Open SCA Collaborationが仕様策定したサービスコンポーネントアーキテクチャ仕様（Service Component Architecture）はそのような技術の一つである。部品（SCA用語ではコンポーネント）のインタフェースを、サービス、リファレンス、プロパティに単純化し、異言語部品や多様な通信プロトコルを組み合わせし易くしたところに特徴がある。その組み合わせ仕様の詳細は非特許文献1に記載されている。

20

【0004】

SCA仕様では、部品の結合をXML言語で記述された定義ファイルを用いて行うように既定されている。ある部品が別の部品を利用する場合、それらの部品は定義ファイルが定める方法によって結合されるが、利用する側の部品は利用される側の部品の機能を直接呼び出すようなコードを書く必要はない。

【0005】

従って、部品のインタフェースが一致していれば、アプリケーションの中で使用されている部品を別の部品に置き換える時にコードを書き直す必要はない。これはSCA技術を使うメリットである。

【0006】

30

一方、SCA以外の従来の技術として、部品を結合してアプリケーションを構築させる際に部品の置換を行ったり結合させるべき部品間の整合性を検証したりする技術に、以下のようなものが挙げられる。

【0007】

例えば、アプリケーション毎に、結合するモジュールの情報を管理・更新し、問題発生時や自動切替時に、モジュール情報から切替えるべきモジュールのバージョンを決定し、そのアプリケーションが結合するモジュールバージョンの切り替えを行う。例えば、特許文献1参照。

【0008】

また、ソフトウェア内の各既開発プログラム部品間の組み合わせ情報および部品の属性情報に基づいて、開発済みのソフトウェアの構造を保持しつつ、当該ソフトウェアの一部をなす特定の既開発プログラム部品を新たな既開発プログラム部品に置換する。例えば、特許文献2参照。

40

【0009】

また、コンポーネント呼び出しを監視するモニタにより抽出されたコンポーネント呼び出し情報に基づいて、コンポーネント呼び出しとその対象であるコンポーネントとの整合性を検証する。例えば、特許文献3参照。

【先行技術文献】

【特許文献】

【0010】

50

【特許文献１】特開２００８－１０８１５５号公報

【特許文献２】特開２００２－３６６３５６号公報

【特許文献２】特開２００４－３６２０００号公報

【非特許文献】

【００１１】

【非特許文献１】SCA Service Component Architecture Assembly Model Specification Version 1.00, March 15 2007

【発明の概要】

【発明が解決しようとする課題】

【００１２】

10

部品の置き換えがし易いというSCAの特徴は好ましい。しかしながら、SCA仕様では、アプリケーションを構築する際に、定義ファイルの内容を決定しておかなければならない。定義ファイルはアプリケーションの一部だからである。つまり、部品の置き換えを行う場合は、定義ファイルを変更せねばならない。それはアプリケーションの構築し直しを意味する。アプリケーションの実行時ではなく、開発時に部品の置き換えが可能ということである。

【００１３】

部品の置き換えの柔軟性はできるだけ高めておくことが望ましい。たとえ部品のインタフェースが同一であったとしても、性能や使用リソース、スケーラビリティ、アベイラビリティなど、部品の特性としてはいくつも考えられ、実現したい特性に応じて多数の部品が提供されることはあり得る。このような状況では、部品を置き換えたいという要求が自然に発生すると考えられる。例えば、ユーザが多数の部品の中から所望の部品を選べたり、機器が備えるリソース条件から動作すべき部品が決定できたりする、そのような機能を提供することは望ましいだろう。

20

【００１４】

しかし、部品の置換を実現するのにアプリケーションの開発し直しが必要となるのでは柔軟性が高いとは言えない。多様な機器に搭載するアプリケーションの部品を機器毎に変更したい場合に、その度に開発のやり直しを行うのでは開発コストがかかり過ぎる。一方、部品の置換を行いたい別の状況が存在する。

【００１５】

30

今日、多くのネットワークサービスが提供されている。機器がネットワークサービスと連携することで従来できなかったサービスが実現されていることも多い。機器の付加価値向上のためにはネットワークサービスとの連携が欠かせない。

【００１６】

ここで、新たなネットワークサービスが提供される状況を考えてみる。新ネットワークサービスが作られ、提供される頻度は、機器上のソフトウェアが更新される頻度の比ではない。ネットワーク上のコンピュータの性能、リソース、開発環境と、機器のそれを比べてみれば当然である。従って、機器ソフトウェアが更新されないままであるのに、ネットワークサービスが更新され、向上していく状況が常に起こり得る。

【００１７】

40

こういった場合、機器が出荷された後、後から作られたネットワークサービスと機器を連携させることは従来は困難であった。一部の機器ではファームウェアアップデート機能が備えられる。それを使えば新ネットワークサービスと連携させることは不可能ではない。しかしながら、ファームウェアアップデートはユーザに負担をかけるし、ROM容量等の面からソフトウェアの変更の自由度は低い。さらに、上述したのと同様に、これも開発行為が必要となる。

【００１８】

本発明は、アプリケーションの開発し直しが不要で、アプリケーションの実行時に部品の置換を可能とする装置を提供する。

【課題を解決するための手段】

50

【 0 0 1 9 】

アプリケーションを構成する部品と外部サービスとを結合する方法であって、
前記外部サービスのインタフェースの情報を取得する取得工程と、
他の部品と結合された処理対象の部品が他の部品のサービスを利用するためのインタフェースに、前記取得工程で取得した情報が示す外部サービスのインタフェースが適合するか判断する判断工程と、
前記判断工程によって、前記処理対象の部品が前記他の部品を利用するためのインタフェースに前記外部サービスのインタフェースが適合すると判断された場合、前記処理対象の部品が、前記取得工程で取得した情報が示す外部サービスと結合されるように、前記処理対象の部品と前記他の部品との結合を変更する変更工程と、
前記外部サービスと結合された部品を実行する実行工程と、
を備えることを特徴とする方法。

10

【 発明の効果 】

【 0 0 2 0 】

本発明によれば、アプリケーションの開発し直しが不要で、アプリケーションの実行時に部品の置換が可能になる。

【 図面の簡単な説明 】

【 0 0 2 1 】

【 図 1 】 第 1 の実施形態における部品結合装置 1 0 0 の構成を示すブロック図。

【 図 2 】 部品と部品の結合及び部品と外部サービスとの結合を示す図。

20

【 図 3 】 (A) は部品結合関係を S C A 仕様に則って記述した部品結合定義を示す図、 (B) は alternative タグが追加されている部品結合定義を示す図。

【 図 4 】 複数の部品及び複数の外部サービスの部品結合関係を示す図。

【 図 5 】 図 4 に示す部品結合関係を表現した表形式のデータを示す図。

【 図 6 】 部品置換を行う際の全体の概略処理を示すフローチャート。

【 図 7 】 部品結合処理 (S 6 0 2) の詳細を示すフローチャート。

【 図 8 】 部品 A と部品 B との結合の間に代理オブジェクトが介在する状態を示す図。

【 図 9 】 部品実行処理 (S 6 0 4) の詳細を示すフローチャート。

【 図 1 0 】 部品 A が外部サービス C に結合される場合を説明するための図。

【 図 1 1 】 部品置換処理 (S 6 0 5) の詳細を示すフローチャート。

30

【 図 1 2 】 更新後の部品結合定義の一例を示す図。

【 図 1 3 】 第 2 の実施形態における部品結合処理を示すフローチャート。

【 発明を実施するための形態 】

【 0 0 2 2 】

以下、図面を参照しながら発明を実施するための形態について詳細に説明する。本実施形態では、ソフトウェア部品 (部品) 又はコンポーネントを結合してアプリケーションを構築する部品結合装置について説明する。

【 0 0 2 3 】

〔 第 1 の実施形態 〕

図 1 は、第 1 の実施形態における部品結合装置 1 0 0 の構成を示すブロック図である。
図 1 に示す中央処理装置 (C P U) 1 0 1 は、装置全体を制御すると共に、置換部 1 1 0、結合部 1 1 1、実行部 1 1 2、出力部 1 1 3、入力部 1 1 4、及びインタフェース検証部 1 1 5 として機能する。

40

【 0 0 2 4 】

ハードディスクドライブ (H D D) 1 0 2 は、記憶部 (メモリ) であり、 H D D 1 0 2 にはアプリケーション 1 2 0 が記憶されている。尚、アプリケーション 1 2 0 は部品結合装置 1 0 0 の出荷時から格納されていても良いし、出荷後に H D D 1 0 2 にインストールされても良い。

【 0 0 2 5 】

アプリケーション 1 2 0 は、 1 以上の部品 1 2 1 と、それらの部品 1 2 1 の結合方法を

50

記述した部品結合定義 1 2 2 とを含む。また、部品として提供されるソフトウェア以外のソフトウェア、即ち、アプリケーションロジック（不図示）もアプリケーション 1 2 0 に含まれる。尚、当然のことながら、アプリケーションロジックから部品 1 2 1 を利用することは可能である。

【 0 0 2 6 】

アプリケーションロジックから部品 1 2 1 を利用する方法は、部品 1 2 1 から他の部品 1 2 1 を利用する方法と変わりがない。そのため、以下ではアプリケーションロジックに対して、特に言及しない。

【 0 0 2 7 】

ネットワーク I / F 1 0 3 は、部品結合装置 1 0 0 を装置外部に存在するネットワーク 1 5 0 に接続するためのインタフェースである。ROM 1 0 4 には、各種動作プログラムが読み取り可能な形式で格納されている。RAM 1 0 5 は、CPU 1 0 1 が ROM 1 0 4 に格納されているプログラムに従って処理を実行中に使用される一時的な値等を保存するメモリである。

【 0 0 2 8 】

Flash ROM 1 0 6 は、各種設定データファイル等を保存しておくための不揮発性メモリである。尚、不揮発性という性質が HDD 1 0 2 に近いことから Flash ROM 1 0 6 を本発明における記憶部として用いることも可能である。

【 0 0 2 9 】

ディスプレイ 1 0 7 は、各種情報をユーザに表示する表示装置である。操作部 1 0 8 は、具体的にはキーボード、マウス、タッチディスプレイ等のユーザが入力操作を行うものである。部品結合装置 1 0 0 はネットワーク I / F 1 0 3、ネットワーク 1 5 0 を介して PC 1 3 0 及びサーバ 1 4 0 に接続されている。

【 0 0 3 0 】

次に、上述の構成を有する部品結合装置 1 0 0 において、アプリケーション 1 2 0 の中の部品 1 2 1 間の結合及び部品 1 2 1 と外部サービスとの部品結合関係を、図 2 を用いて説明する。

【 0 0 3 1 】

図 2 は、部品と部品の結合及び部品と外部サービスとの結合を示す図である。サービスコンポーネントアーキテクチャ（SCA）仕様によれば、部品がその機能を部品外部に公開するインタフェースはサービスと呼ばれ、部品が別の部品のサービスを利用するためのインタフェースは参照と呼ばれる。図 2 に示す例では、部品 A が 2 つの参照 2 1 0 A、2 1 0 B のインタフェースを有し、部品 B がサービス 2 2 0 A のインタフェースを有し、外部サービス C がサービス 2 2 0 B のインタフェースを有する。ここで、ある参照があるサービスを利用するときに、その参照とそのサービスとが結合される。

【 0 0 3 2 】

図 2 では、部品 A が部品 B を利用する際の参照 2 1 0 A とサービス 2 2 0 A との結合を関係線 2 3 0 A で示し、部品 A が外部サービス C を利用する際の参照 2 1 0 B とサービス 2 2 0 B との結合を関係線 2 3 0 B で示している。このとき、結合された参照 2 1 0 A とサービス 2 2 0 A とのインタフェースを整合しなければならない。整合しなければ、参照 2 1 0 A からサービス 2 2 0 A の全ての機能を利用することができないからである。

【 0 0 3 3 】

Java（登録商標）言語で言えば、参照 2 1 0 A が依拠する Java（登録商標）インタフェースクラスとサービス 2 2 0 A が依拠する Java（登録商標）インタフェースクラスとの互換性がとれていなければならない。

【 0 0 3 4 】

具体的には、サービス 2 2 0 A が依拠する Java（登録商標）インタフェースクラスは参照 2 1 0 A が依拠する Java（登録商標）インタフェースクラスと一致しているか、もしくはスーパーセットでなければならない。非特許文献 1 には、その他の条件も記載されている。

10

20

30

40

50

【 0 0 3 5 】

図 2 に示すように、結合は同一のアプリケーション 1 2 0 中の部品同士だけでなく、アプリケーション 1 2 0 内の部品 A とアプリケーション 1 2 0 外の外部サービス C との間でも行うことができる。その意味で、外部サービス C は部品 A、部品 B と同様に扱うことができる。

【 0 0 3 6 】

図 3 の (A) に、図 2 に示す部品結合関係を S C A 仕様に則って記述した部品結合定義 1 2 2 の一部を示す。図 3 に示す (A) では、説明のために行番号が付与されているが、本来は X M L 言語形式で記述されるものである。

【 0 0 3 7 】

2 行目は部品 A についての記述であることを示す。3 行目は部品 A の参照 2 1 0 A を示し、その名前は「 b 」であること、それがターゲット (結合先) 部品 B に結合されることを示す。4 行目は参照 2 1 0 A のインタフェースが J a v a (登録商標) インタフェースクラス B B B (不図示) に従うことを示す。

【 0 0 3 8 】

以下同様に、6 ~ 9 行目は参照 2 1 0 B が外部サービス C に結合されることを示す。尚 8 行目は結合が W e b サービスを用いて行われることを示す。W e b サービスでアクセスするのに必要となる U R I (Uniform Resource Identifier) は u r i 属性で示される。

【 0 0 3 9 】

部品 A の参照 2 1 0 B と外部サービス C のサービス 2 2 0 B との間のインタフェースの整合性は次のように確認する。まず、参照 2 1 0 B のインタフェースについては、7 行目でそれが J a v a (登録商標) インタフェースクラス C C C に従うことが分かる。サービス 2 2 0 B が提供するインタフェースについては、それに対応する W S D L ファイル (不図示) の内容を調べる。一般に、W e b サービスではサービスのインタフェースの内容を示す W S D L ファイルが提供される。従って、W S D L ファイルを調べればサービス 2 2 0 B のインタフェースの内容を知ることができる。それが J a v a (登録商標) インタフェースクラス C C C と互換性があるか否かを調べれば良い。

【 0 0 4 0 】

次に、第 1 の実施形態におけるアプリケーション 1 2 0 内の複数の部品及び複数の外部サービスの部品結合関係を、図 4 を用いて説明する。ここで、アプリケーション 1 2 0 は複数の部品 1 2 1 として部品 A から部品 L を含む。

【 0 0 4 1 】

図 2 では、参照とサービスとを明示していたが、図 4 ではそれらを省略し、任意の部品から別の部品への結合を単純な矢印で表現している。実線の矢印は実際に結合されていることを示し、破線の矢印は結合候補であることを示す。例えば、部品 A は現在、部品 B に結合されている。両者の間の結合インタフェースはインタフェース B B B である。部品 B の他にインタフェース B B B を満たす 2 つの部品 (部品 D、部品 E) がアプリケーション 1 2 0 内に存在し、部品 A はそれらとも結合することができる。

【 0 0 4 2 】

また、現在結合されているものが部品であれば、結合候補も部品でなければならないということはない。即ち、部品 F は部品 G に結合されているが、そのインタフェース G G G を満たす外部サービス C を結合候補とすることもできる。

【 0 0 4 3 】

また、部品 H の場合、その結合が示すように、現在は外部サービス I に結合されているが、インタフェース I I I を満たす外部サービス J を結合候補とすることもできる。更に、部品 K は、現在部品 L に結合されているだけで、結合候補は持っていない例である。

【 0 0 4 4 】

ここで、図 4 に示す部品結合関係を表現した表形式のデータを、図 5 を用いて説明する。この表の内容となるデータは R A M 1 0 5 に格納される。図 5 において、部品列 5 0 1 は参照 2 1 0 を持つ部品 1 2 1 のリストである。参照列 5 0 2 は、対応する部品列 5 0 1

10

20

30

40

50

の部品 1 2 1 が備える参照 2 1 0 である。この例では、一つの部品につき一つの参照しか記述していないが、一つの部品につき複数の参照が存在しても良い。その場合には一つの部品につき複数の行が存在することになる。

【 0 0 4 5 】

使用部品列 5 0 3 は、対応する参照列 5 0 2 の参照が結合している先の部品である。尚、上述したように、外部サービスはアプリケーション 1 2 0 内の部品 1 2 1 と同様に取り扱えるので、使用部品列 5 0 3 に外部サービスが入っていても良い。

【 0 0 4 6 】

部品候補列 5 0 4 は、対応する参照列 5 0 2 の参照が結合する候補となる部品のリストである。また、部品候補列 5 0 4 の一つのセル内に複数の部品及び外部サービスが入っても良い。インタフェース列 5 0 5 は、対応する参照列 5 0 2 の参照が従うインタフェースである。

10

【 0 0 4 7 】

次に、図 5 に示す表データを生成する基となる部品結合定義 1 2 2 を図 3 に示す (B) を用いて説明する。図 3 に示す (A) を用いて説明した、 S C A 仕様で一般的な部品結合定義 1 2 2 とほぼ同じであるが、図 3 の (B) では更に、alternative タグが追加されている。このタグを用いて結合候補部品が記述される。

【 0 0 4 8 】

例えば、5 ~ 6 行目では、参照 a に対する結合候補として部品 D と部品 E が記述されている。1 2 ~ 1 4 行目では、参照 f に対する結合候補として外部サービス C が挙げられている。これらの結合候補が部品候補列 5 0 4 に格納される。

20

【 0 0 4 9 】

ここで、第 1 の実施形態における部品結合装置 1 0 0 が部品置換を行う際の全体の概略処理を、図 6 を用いて説明する。最初に、部品結合装置 1 0 0 が外部装置 (不図示) からアプリケーション 1 2 0 をインストールする (S 6 0 1)。このインストールの操作は、ユーザが操作部 1 0 8 とディスプレイ 1 0 7 とを用いて行う。そして、操作部 1 0 8 からインストールが指示されると、 C P U 1 0 1 がネットワーク 1 5 0 に接続されている外部装置からネットワーク I / F 1 0 3 を介してアプリケーション 1 2 0 をダウンロードして H D D 1 0 2 に格納する。

【 0 0 5 0 】

30

次に、ユーザが操作部 1 0 8 からアプリケーション 1 2 0 の開始を指示すると、 C P U 1 0 1 がアプリケーション 1 2 0 内の部品結合定義 1 2 2 を読み出し、解析することで、アプリケーション 1 2 0 内の部品 1 2 1 の結合処理を行う (S 6 0 2)。この結合処理を行った後、 C P U 1 0 1 はアプリケーション 1 2 0 を実際に開始する。それに伴い、部品 1 2 1 のコードを実行する (S 6 0 4)。

【 0 0 5 1 】

これ、ユーザが現在使用している部品 1 2 1 を置き換えなくなった時、部品 1 2 1 の置換処理を指示する。この指示に応じて C P U 1 0 1 は部品 1 2 1 の置換処理を実行する (S 6 0 5)。部品 1 2 1 の置換処理を行った後には (S 6 0 6)、ループの先頭に戻り (S 6 0 3)、置き換えた部品を使用する。つまり、 C P U 1 0 1 は置換した部品 1 2 1 のコードを実行する (S 6 0 4)。

40

【 0 0 5 2 】

尚、この例では、一旦、部品 1 2 1 を実行した後、部品 1 2 1 の置換を行っているが、アプリケーション 1 2 0 をインストールした直後、即ち、実行する前に部品 1 2 1 の置換を行っても良い。また、図 8 では省略しているが、アプリケーション 1 2 0 を停止したり、アンインストールしたりすることも当然のことながら可能である。

【 0 0 5 3 】

次に、第 1 の実施形態における部品結合処理 (S 6 0 2) の詳細を、図 7 を用いて説明する。以下のシーケンスでは結合部 1 1 1 として機能する C P U 1 0 1 が部品結合処理の動作主体となる。

50

【 0 0 5 4 】

最初に、CPU 101がHDD 102に格納されたアプリケーション 120内の部品結合定義 122を読み出す(S 701)。そして、CPU 101は読み出した部品結合定義 122を解釈する(S 702)。即ち、CPU 101は図3に示す(B)のようなXML言語で記述された部品結合定義 122を解釈し、図5に示すような表形式のデータを生成し、RAM 105に格納する。

【 0 0 5 5 】

部品結合定義 122には、1以上の部品 121についてのデータが存在する。そこで、ループ1の先頭(S 703)から末尾(S 709)までの処理を部品 121毎に繰り返す。また、一つの部品 121内には1以上の参照が存在し得る。ここで、SCAの仕様上は参照が存在しない部品 121もあり得るが、本発明は部品結合を関心とするので、結合が不可能な参照がない部品 121のことはここでは考えない。つまり、ループ2の先頭(S 704)から末尾(S 708)までの処理を参照毎に繰り返す。

10

【 0 0 5 6 】

次に、CPU 101は処理の対象である参照の結合先部品が外部サービスか否かを判定する(S 705)。尚、使用部品列 503について、外部サービスは部品 121と同様な取り扱いが可能であると説明したが、それは両者の区別がつかないという意味ではない。S 702で部品結合定義 122を解釈した時に、結合先部品がアプリケーション 120内の部品 121(内部部品)か、外部サービスかを示す情報がRAM 105に格納される。

【 0 0 5 7 】

20

S 705で、CPU 101が外部サービスではなく、内部部品 121と判定した場合、内部部品 121用の代理オブジェクトのインスタンスを生成する(S 706)。この代理オブジェクトのインスタンスとは、あるクラスのインタフェースを満たすように、動的に生成される代理クラスのインスタンスである。

【 0 0 5 8 】

Java(登録商標)言語では、あるインタフェースクラスに対して、代理クラスをプログラム中で生成することができる。その代表的な用途は、あるインタフェースクラスを実装した実装クラスのオブジェクトのメソッドをプログラムが呼び出すときに、代理クラスがその呼び出しをキャッチし、そのタイミングで何か別の処理を行うことである。代理クラスは間接的に呼び出しを行う機能を提供する。部品結合においては、その機能が有用であるため、第1の実施形態においては代理クラスを使用している。

30

【 0 0 5 9 】

代理クラスは、部品 121の参照が従うインタフェース 401に基づいて生成される。従って、S 706で代理オブジェクトインスタンスを生成する直前に、代理クラスが生成されれば良い。但し、当然のことながら、図7のループ処理を実行中、同一の代理クラスから複数個の代理オブジェクトインスタンスを生成する場合は、その都度、代理クラスを生成する必要はなく、一度代理クラスを生成すれば良い。

【 0 0 6 0 】

また、上述のS 705で、結合先部品が外部サービスと判定した場合、CPU 101は外部サービス用の代理オブジェクトのインスタンスを生成する(S 710)。ここで代理オブジェクトのインスタンスは結合先部品に関連する情報を含むことに注意すべきである。例えば、代理オブジェクトのインスタンス生成時には、結合先の内部部品 121のクラス情報或いは外部サービスのアクセス用部品(詳細は後述する)のクラス情報がコンストラクタパラメータとして渡され、インスタンス内部に保存される。これらのクラス情報は結合先内部部品 121或いは外部サービスアクセス用部品のインスタンスを生成するのに用いられる。

40

【 0 0 6 1 】

上述したS 706又はS 710の処理の後に、部品 121の参照に対応するフィールドに代理オブジェクトインスタンスを設定する(S 907)。ここでフィールドとは、部品 121オブジェクトの中にある情報格納場所であり、Java(登録商標)言語で言うと

50

ころのフィールドに相当する。フィールドに設定されたオブジェクトを通じて、他のクラスのメソッドを呼び出すことができる。他の言語でもこのフィールドに相当するものは存在する。

【 0 0 6 2 】

図 8 に、部品 A と部品 B との結合の間に代理オブジェクトが介在する状態を示す。アプリケーション 1 2 0 側から見ると、アプリケーション 1 2 0 は代理オブジェクト B の存在を知らないので、部品 A が部品 B に直接結合されているように見える。

【 0 0 6 3 】

しかし、CPU 1 0 1 は結合の間に代理オブジェクト B を介在させる。即ち、S 7 0 6 又は S 7 1 0 で代理オブジェクト B のインスタンスを生成し、S 7 0 7 でそれを部品 A の参照 2 1 0 A に対応するフィールド 1 2 0 3 A に設定する。設定動作を矢印 1 2 0 2 A で示している。この設定により部品 A と代理オブジェクト B が結合される。

【 0 0 6 4 】

代理オブジェクト B は、部品 A の参照 2 1 0 A が従うインタフェースと整合性を持つ代理クラスから生成されたものである。従って、代理オブジェクト B のサービス 2 2 0 C のインタフェースは、必然的に参照 2 1 0 A のインタフェースと互換性を持ち、参照 2 1 0 A とサービス 2 2 0 C が結合される。つまり、部品 A は、参照 2 1 0 A を通じて代理オブジェクト B のメソッドの呼び出しを確実に行うことができる。代理オブジェクト B と部品 B との結合については後述する。

【 0 0 6 5 】

図 7 の説明に戻ると、S 7 0 8 でループ 2 の末尾に至り、S 7 0 4 のループ先頭に戻る。S 7 0 9 でループ 1 の末尾に至り、S 7 0 3 のループ先頭に戻る。以上で部品結合手順を終了する。

【 0 0 6 6 】

次に、第 1 の実施形態における部品実行処理 (S 6 0 4) の詳細を、図 9 を用いて説明する。以下のシーケンスでは実行部 1 1 2 として機能する CPU 1 0 1 が部品実行処理の動作主体となる。

【 0 0 6 7 】

最初に、アプリケーション 1 2 0 が、部品 A が提供するサービスを呼び出す (S 9 0 1)。即ち、CPU 1 0 1 が、部品 A が提供するサービスの実装であるメソッドを呼び出す。図 8 に示すように、部品 A が部品 B を使用しているものとする。部品 A は部品 B を使用するために、参照 2 1 0 A が従うインタフェースのメソッドを呼び出す (S 9 0 2)。

【 0 0 6 8 】

参照 2 1 0 A のインタフェースのメソッドは、フィールド 1 2 0 3 A を通じて代理オブジェクト B のメソッドを呼び出す (S 9 0 3)。図 7 を用いて説明したように、部品結合処理でフィールド 1 2 0 3 A への設定が済んでいるので、この処理が可能になる。

【 0 0 6 9 】

代理オブジェクト B のメソッドは、自オブジェクトが外部サービス用か否かを判定する (S 9 0 4)。S 7 0 6 又は S 7 1 0 で代理オブジェクトインスタンスを作成した時に、内部部品 1 2 1 用或いは外部サービス用の何れの用途で作成したかの情報を記憶しておき、その情報に基づいて判定を行う。判定の結果、内部部品 1 2 1 用と判定した場合、代理オブジェクト B のメソッドは、代理オブジェクトインスタンス内に格納された結合先内部部品のクラス情報を用いて結合先内部部品のインスタンスを生成する (S 9 0 5)。この処理では、結合先内部部品は部品 B であるので、部品 B のインスタンスが生成される。

【 0 0 7 0 】

また、S 9 0 4 で外部サービス用と判定した場合、代理オブジェクト B のメソッドは、結合先外部サービスアクセス用部品のインスタンスを生成する (S 9 0 8)。このインスタンスの生成は、代理オブジェクトインスタンス内に格納された結合先外部サービスアクセス用部品のクラス情報を用いて行われる。例えば、図 1 0 に示すように、部品 A が外部サービス C に結合される場合、外部サービス C にアクセスするための外部サービス C アク

10

20

30

40

50

セス用部品のインスタンスが生成される。

【 0 0 7 1 】

代理オブジェクト B のメソッドは、S 9 0 5 又は S 9 0 8 で生成されたインスタンスを、代理オブジェクト B 内のフィールド 1 2 0 3 B に設定する (S 9 0 6)。結合先部品が部品 B の場合の設定動作を図 8 の矢印 1 2 0 2 B に示す。結合先部品が外部サービス C の場合の設定動作を図 1 3 の矢印 1 2 0 2 C に示す。

【 0 0 7 2 】

最後に、代理オブジェクト B のメソッドは、フィールド 1 2 0 3 B に設定されたインスタンスのメソッドを呼び出す (S 1 0 0 7)。インスタンスが部品 B のものである場合は、このメソッド呼び出しは部品 B のメソッド呼び出しとなる。また、インスタンスが外部サービス C アクセス用部品のものである場合は、このメソッド呼び出しは外部サービス C アクセス用部品のメソッド呼び出しとなる。外部サービス C アクセス用部品のメソッド呼び出し (不図示) は、外部サービス C にアクセスするために必要な初期化を行い、その後実際にサービスのアクセス処理、例えば W e b サービス通信処理を行う。

【 0 0 7 3 】

以上の処理をもって、部品 A から部品 B の呼び出し、又は部品 A から外部サービス C へのアクセスが完結する。以上が部品実行処理である。

【 0 0 7 4 】

次に、第 1 の実施形態における部品置換処理 (S 6 0 5) の詳細を、図 1 1 を用いて説明する。以下のシーケンスでは置換部 1 1 0、出力部 1 1 3、入力部 1 1 4、インタフェース検証部 1 1 5 として機能する C P U 1 0 1 が部品置換処理の動作主体となる。

【 0 0 7 5 】

最初に、ユーザが P C 1 3 0 にインストールされている W e b ブラウザを用いて部品結合装置 1 0 0 が提供する W e b ページにアクセスする (S 1 1 0 1)。ここでアクセスは、H T T P (Hyper Text Transfer Protocol) プロトコルを用いて行われる。このとき、ユーザが要求したページは部品の置換機能を提供するページであり、J a v a (登録商標) サンプルット (出力サンプルット 1 1 3) によって実現されている。ここで J a v a (登録商標) サンプルットは、W e b ページを出力する手法の一例であり、他の言語や方法を用いても差し支えない。また、J a v a (登録商標) サンプルットによる W e b ページ出力も H T T P プロトコルを用いて行われる。

【 0 0 7 6 】

P C 1 3 0 の W e b ブラウザから部品置換の要求を受けて、部品結合装置 1 0 0 の出力サンプルット 1 1 3 が部品置換のための W e b ページを P C 1 3 0 に対して送信する (S 1 1 0 2)。ここで部品置換のための W e b ページ (不図示) のページ内容には、現在の部品結合関係と、部品結合候補が記述される。ここでは、図 4 及び図 5 において、部品 A の参照 a に結合される部品を置換しようとして W e b ページが要求された場合を考える。この場合、部品 A の参照 a には現在部品 B が結合されており、参照 a に結合され得る候補部品として部品 D と部品 E が存在することを示す W e b ページが送信される。典型的には、結合候補部品は選択リストとして表現されているものとする。

【 0 0 7 7 】

P C 1 3 0 のユーザは、送信された W e b ページを見て、部品 A の参照 a に結合すべき部品を選択する。例えば、W e b ページの選択リストから部品 D を選択し、フォーム送信ボタンを押下する。これによって、選択部品を記述したデータが P C 1 3 0 から部品結合装置 1 0 0 に送られ、入力サンプルット 1 1 4 が部品の置換内容であるデータを受信する (S 1 1 0 3)。

【 0 0 7 8 】

尚、図 1 に示す例では、出力サンプルット 1 1 3 と入力サンプルット 1 1 4 とを異なるものとしているが、同一のサンプルットであっても良い。その場合、部品置換内容データの存在の有無により、出力として機能すべきか、入力として機能すべきかを判断し、判断結果に応じて機能すれば良い。

10

20

30

40

50

【 0 0 7 9 】

入力サーブレット 1 1 4 は、部品置換内容データを置換部 1 1 0 に渡す。置換部 1 1 0 は置換後の結合先が外部サービスか否かを判定する (S 1 1 0 4)。ここでは、部品置換内容データを見て、部品 A の参照 a に新たに結合すべき部品が部品 D であることを知る。即ち、外部サービスではなく部品 D なので、置換部 1 1 0 は部品 D 用の代理オブジェクトインスタンスを生成する (S 1 1 1 2)。そして、置換部 1 1 0 は部品 A の参照 a に対応するフィールドにその代理オブジェクトインスタンスを設定する (S 1 1 0 8)。

【 0 0 8 0 】

次に、部品置換内容を反映するように、部品結合関係を示すデータ (図 5 の表データ) を更新する (S 1 1 0 9)。具体的には、部品 A の参照 a に対応する使用部品列 5 0 3 のセルの内容を、部品 B から部品 D に置き換える。また、部品候補列 5 0 4 のセルの内容のうち、部品 D を部品 B に置き換える。部品 E についてはそのままとする。これによって、部品 D が使用部品となると共に、今まで使用していた部品 B は結合候補となる。

【 0 0 8 1 】

続いて、部品置換内容を反映するように、部品結合定義 1 2 2 を更新する (S 1 1 1 0)。更新後の部品結合定義 1 2 2 の一例を図 1 2 に示す。図 1 2 と図 3 に示す (B) とを比較すると、3 行目において、部品 A の参照 a に対応する結合先部品が部品 B から部品 D に置き換えられている。また、5 行目において、結合候補が部品 D から部品 B に置き換えられている。

【 0 0 8 2 】

また、部品結合定義 1 2 2 の更新内容は、H D D 1 0 2 に書き込まれる。これにより、部品結合装置 1 0 0 が再起動された後、部品結合処理が実行されたとき、置換後の内容で部品 1 2 1 が結合されるという効果がある。

【 0 0 8 3 】

最後に、置換部 1 1 0 が置換成功を出力サーブレット 1 1 3 に通知し、出力サーブレット 1 1 3 が置換成功通知のための W e b ページを P C 1 3 0 に送信する (S 1 1 1 1)。これにより、P C 1 3 0 の W e b ブラウザ上には置換成功を示す W e b ページが表示され、ユーザは置換成功を知る。

【 0 0 8 4 】

また同様に、ユーザが部品 F の参照 f に結合される部品を部品 G から外部サービス C に置き換える場合を、図 1 1 を用いて説明する。S 1 1 0 1 ~ S 1 1 0 3 については上述の場合と同様である。ここで、部品 F の参照 f に結合される部品として、外部サービス C を P C 1 3 0 のユーザは選択したとする。

【 0 0 8 5 】

置換部 1 1 0 は、ユーザが選択した部品 1 2 1 を見て置換後の結合先が外部サービスか否かを判定する (S 1 1 0 4)。この場合、外部サービスなので S 1 1 0 5 へ進み、インタフェース検証部 1 1 5 が、指定された部品である外部サービス C のインタフェース情報を取得する。具体的には、外部サービス C に対応する W S D L ファイルの内容を調べる。W S D L ファイルの位置は、部品結合定義 1 2 2 の中に記述されている外部サービス C に対するアクセス U R I 情報又はその W S D L ファイル位置情報 (不図示) から知ることができる。

【 0 0 8 6 】

次に、インタフェース検証部 1 1 5 が、取得したインタフェース情報から外部サービス C のインタフェースが参照 f のインタフェースに適合しているか否かを判断する (S 1 1 0 6)。ここで適合していると判断すれば、置換部 1 1 0 は外部サービス C 用の代理オブジェクトインスタンスを生成する (S 1 1 0 7)。そして、置換部 1 1 0 は部品 F の参照 f に対応するフィールドに、その代理オブジェクトインスタンスを設定する (S 1 1 0 8)。S 1 1 0 9 以降は上述の通りである。

【 0 0 8 7 】

これに対して、S 1 1 0 6 で適合していないと判断すれば、置換部 1 1 0 は置換エラー

10

20

30

40

50

を出力サプレット 113 に通知し、出力サプレット 113 が置換エラー通知のための Web ページを PC 130 に送信する (S 1113)。これにより、PC 130 の Web ブラウザ上には置換エラーを示す Web ページが表示され、ユーザは置換エラーを知ることができる。

【0088】

図 12 に示すように、10 ~ 14 行目は、部品 F の参照 f に結合される部品が部品 G から外部サービス C に置き換えられている。更に、処理は説明していないが、図 12 に示す 17 ~ 23 行目は、部品 H の参照 h に結合される部品が外部サービス I から外部サービス J に置き換えられている。

【0089】

上述した処理では、PC 130 の Web ブラウザに対して結合部品候補を出力し、PC 130 のユーザがその結合部品候補から結合部品を選択していたが、本発明はこれに限定されるものではない。

【0090】

S 1102 で、部品置換用の Web ページを送信するとき、自由入力用のテキストフィールドを含んだ Web ページを送信する。ここで、候補にはない外部サービスの URL を PC 130 のユーザがテキストフィールドに入力し、その入力内容を部品結合装置 100 に送ったとする。この場合には、S 1103 でその URL が置換内容として入力される。尚、S 1104 以降の処理は上述の処理と同様であり、新たに指定された外部サービスが結合される。

【0091】

このとき、S 1109 と S 1110 は上述の処理と少し異なる。新たに指定された外部サービスは使用部品として認識され、今まで使用していた部品 121 は結合部品の候補として認識される。従って、結合候補の部品が一つ増加する。それを示すように図 5 に示す表データと部品結合定義 122 が更新される。

【0092】

第 1 の実施形態は、結合候補部品の出力と選択した部品の入力を、HTTP プロトコルを用いて行っており、そのため、出力部 113 は出力サプレットであり、入力部 114 は入力サプレットであった。

【0093】

しかし、本発明はこれだけに限定されるものではない。結合候補部品の出力と選択した部品の入力は他の一般的な入出力デバイスを用いて行っても良い。例えば、ディスプレイ 107 と操作部 108 を用いて同様に結合候補部品の出力と選択した部品の入力が可能である。

【0094】

[第 2 の実施形態]

次に、図面を参照しながら本発明に係る第 2 の実施形態を詳細に説明する。第 1 の実施形態では、出力部 113 と入力部 114 を用いて部品 121 を置換した。第 2 の実施形態では、出力部 113 と入力部 114 を用いずに部品 121 を置換する方法を説明する。

【0095】

第 2 の実施形態における部品結合装置の構成は、第 1 の実施形態での図 1 に示す構成と同様であり、その説明は省略する。

【0096】

ここで、第 2 の実施形態における部品結合処理を、図 13 を用いて説明する。図 13 において、S 1301 ~ S 1304 は S 701 ~ S 704 と同様である。

【0097】

S 1305 で、結合部 111 として機能する CPU 101 が現在処理中の参照について使用部品と候補部品の情報を取得する。S 1302 において、図 5 に示す表データを取得済みなので、そこから取得することができる。

【0098】

次に、結合部 1 1 1 はアプリケーション 1 2 0 のメタ情報を取得する (S 1 3 0 6) 。ここでメタ情報とは、アプリケーション 1 2 0 に付随する情報であり、第 2 の実施形態では不図示であるが、アプリケーション 1 2 0 に内包されるものとする。例えば、J a v a (登録商標) アプリケーションの場合、j a r ファイルの中のマニフェストファイルがメタ情報に相当する。

【 0 0 9 9 】

メタ情報には様々な情報が格納されるが、ここでは優先使用部品情報が格納されているものとする。この優先使用部品情報とは、それが指す部品 1 2 1 が候補部品の中にあれば、使用部品として指定された部品 1 2 1 より優先して使用するものである。

【 0 1 0 0 】

次に、結合部 1 1 1 は、取得したアプリケーション 1 2 0 のメタ情報中の優先使用部品情報を見て現在処理中の参照に対応する使用部品をそのまま使用すべきか否かを判定する (S 1 3 0 7) 。

【 0 1 0 1 】

現在処理中の参照に対応する使用部品が優先使用部品情報の中にあれば、そのまま使用すべきだと判断される。一方、現在処理中の参照に対応する候補部品が優先使用部品情報の中にあれば、現在の使用部品がその候補部品に置き換えられるべきだと判断される。

【 0 1 0 2 】

ここで使用部品をそのまま使用すべきだと判断された場合は、S 1 3 0 9 以降の処理を実行する。S 1 3 0 9 ~ S 1 3 1 4 は S 7 0 5 ~ S 7 1 0 と同様である。

【 0 1 0 3 】

また、使用部品が置き換えられるべきだと判断された場合は、現在の使用部品を優先使用部品情報の中に存在する候補部品に置換する (S 1 3 0 8) 。具体的には、S 1 1 0 9 と S 1 1 1 0 で説明したように、表データと部品結合定義 1 2 2 が更新される。そして、S 1 3 0 9 以降の処理を実行する。

【 0 1 0 4 】

尚、第 2 の実施形態では、優先使用部品情報として候補部品情報が格納されている例を示したが、代わりに新たな外部サービスに関する情報が格納されていても良い。その場合、候補にない、新たな外部サービスを使用部品として入力した上述の場合と同様の処理を行う。

【 0 1 0 5 】

また、第 2 の実施形態では、アプリケーション 1 2 0 のメタ情報の中の優先使用部品情報を用いて部品の置換が行われたが、他の方法によるものでも良い。例えば、F l a s h R O M 1 0 6 に優先使用部品情報が格納され、結合部 1 1 1 がそれを読み出し、使用部品の置換の可否を判断しても良い。

【 0 1 0 6 】

優先使用部品情報そのものを使用しない実施形態も考えられる。例えば、部品結合装置 1 0 0 のスペック情報を結合部 1 1 1 が取得し、規定以上のスペックであれば、部品 D を、或いは規定以下のスペックであれば部品 B を使用するなどと判定しても良い。

【 0 1 0 7 】

実施形態によれば、アプリケーションの開発し直しが不要で、アプリケーションの実行時に部品の置換が可能になる。また、結合候補部品を選ぶだけで部品の置換が可能であり、アプリケーションの実行時に部品の置換が簡単に行える。

【 0 1 0 8 】

また、外部サービスを指定でき、インタフェースを検証できることでアプリケーションが意図していなかったサービスとも確実に連携できる。例えば、アプリケーションが完成した後に作成されたネットワークサービスとも連携できる。

【 0 1 0 9 】

[他の実施形態]

また、本発明は、以下の処理を実行することによっても実現される。即ち、上述した実

10

20

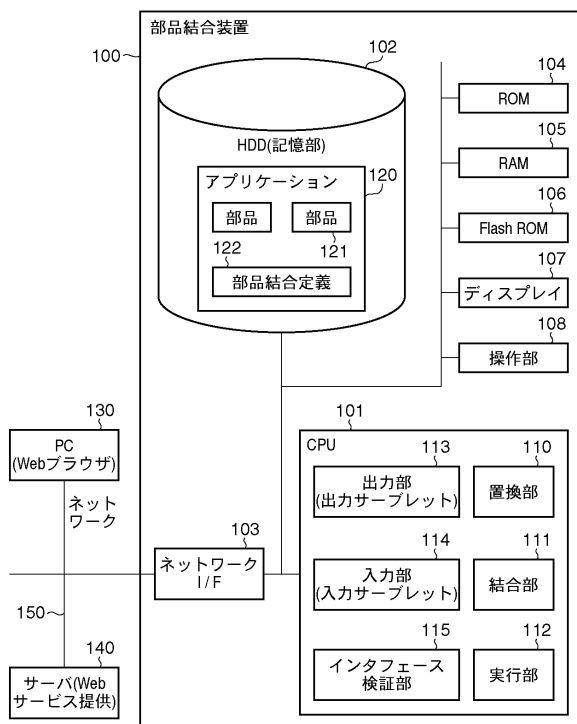
30

40

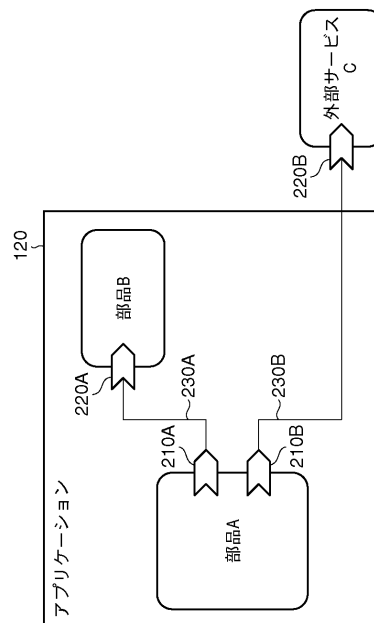
50

施形態の機能を実現するソフトウェア（プログラム）を、ネットワーク又は各種記憶媒体を介してシステム或いは装置に供給し、そのシステム或いは装置のコンピュータ（またはCPUやMPU等）がプログラムを読み出して実行する処理である。

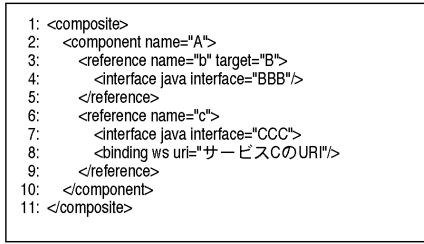
【図 1】



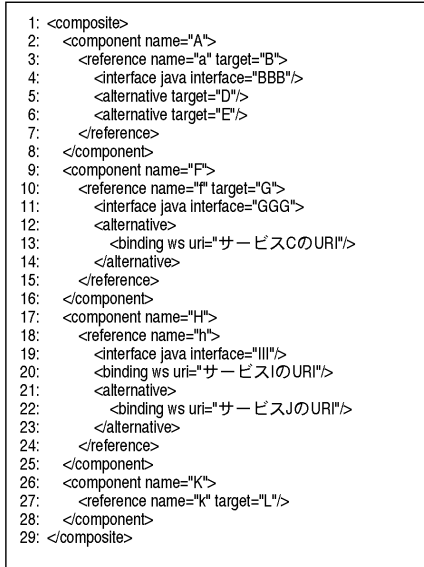
【図 2】



【図 3】

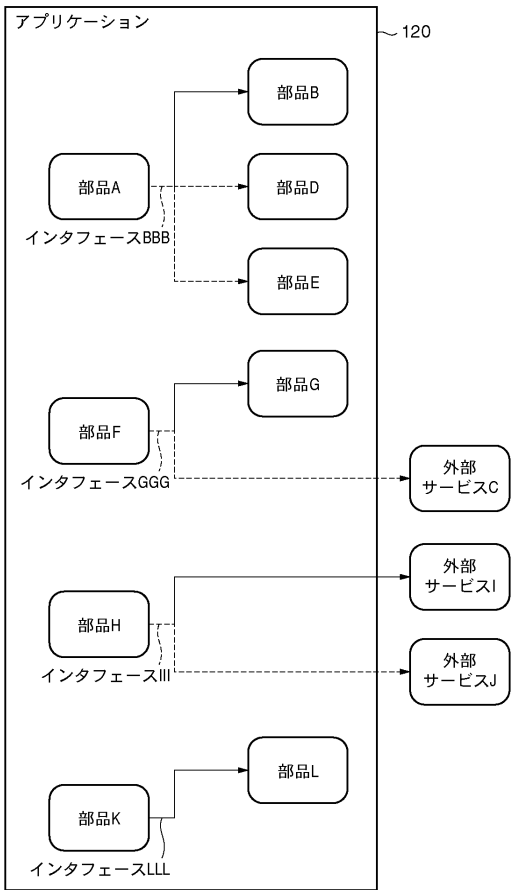


(A)



(B)

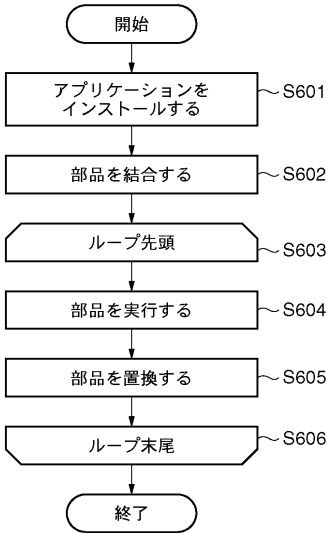
【図 4】



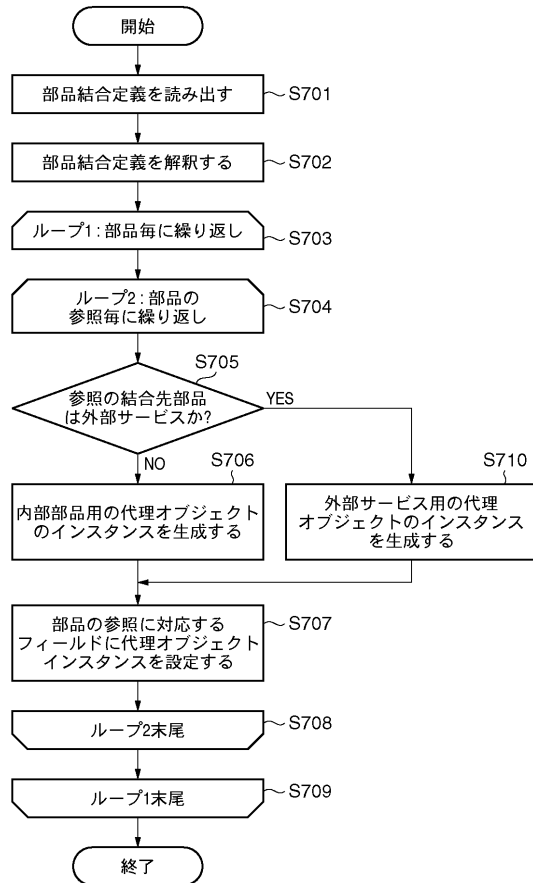
【図 5】

501	部品	参照	使用部品	部品候補	インタフェース
	部品A	参照a	部品B	部品D, 部品E	インタフェースBBB
	部品F	参照f	部品G	外部サービスC	インタフェースGGG
	部品H	参照h	外部サービスI	外部サービスJ	インタフェースIII
	部品K	参照k	部品L	N/A	インタフェースLLL
511					
512					
513					
514					

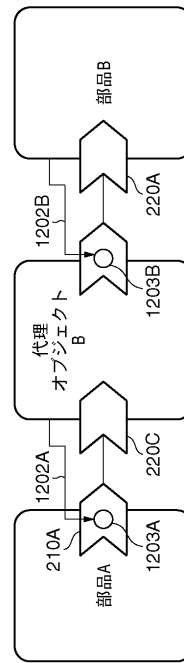
【図 6】



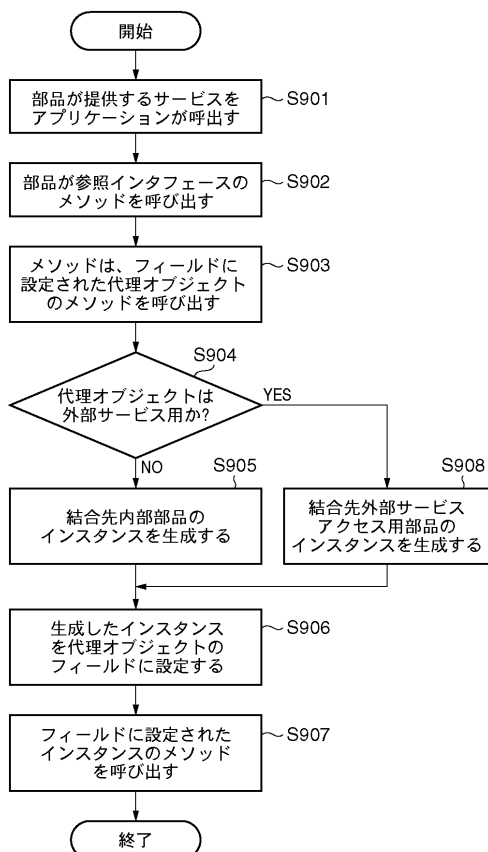
【図 7】



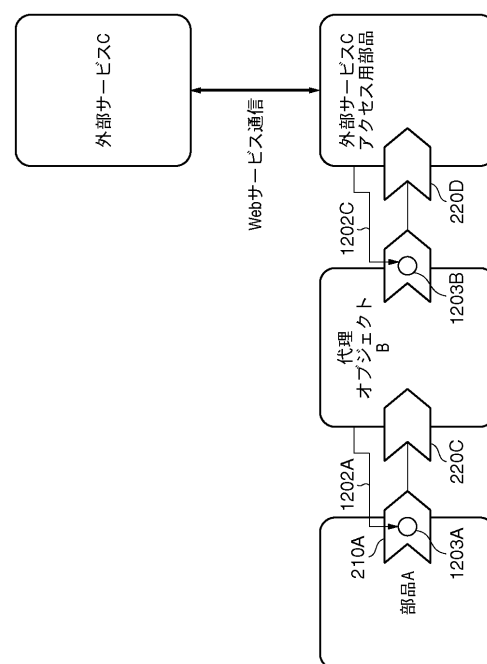
【図 8】



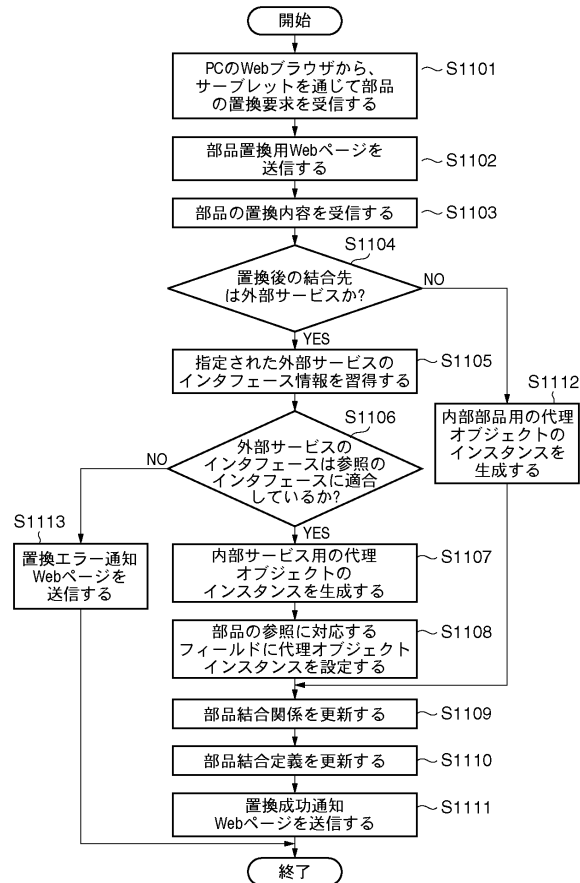
【図 9】



【図 10】



【図 1 1】

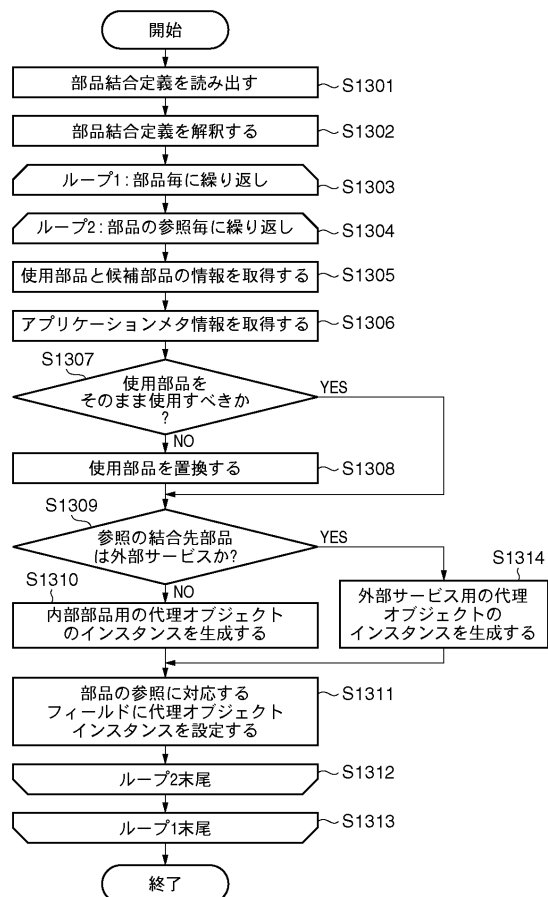


【図 1 2】

```

1: <composite>
2:   <component name="A">
3:     <reference name="a" target="D">
4:       <interface java interface="BBB">
5:         <alternative target="B">
6:           <alternative target="E">
7:             </reference>
8:           </alternative>
9:         </interface>
10:      </reference>
11:    </component>
12:    <component name="F">
13:      <reference name="f">
14:        <interface java interface="GGG">
15:          <binding ws uri="サービスCのURI">
16:            </binding>
17:          </interface>
18:        </reference>
19:      </component>
20:      <component name="H">
21:        <reference name="h">
22:          <interface java interface="III">
23:            <binding ws uri="サービスJのURI">
24:              </binding>
25:            </interface>
26:          </reference>
27:        </component>
28:      </component>
29:    </composite>
  
```

【図 1 3】



フロントページの続き

(72)発明者 大井 浩一
東京都大田区下丸子3丁目30番2号 キヤノン株式会社内

審査官 塚田 肇

(56)参考文献 特開2006-285313(JP,A)
特開平10-254701(JP,A)
特開2008-262351(JP,A)
Cosminexusアプリケーションサーバ V8 SOAPアプリケーション開発の手引
手引・文法書,株式会社日立製作所,2010年 2月,第2版,pp.126-129
浅井 直樹,情報サービス産業白書2008,2009年 5月13日,第1版,第95頁

(58)調査した分野(Int.Cl., DB名)
G06F 9/44