(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2011-123842
(P2011-123842A)
(43)公開日　平成23年6月23日(2011.6.23)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 3/12	(2006.01)	G06F	3/12	A	2C061
B41J 29/38	(2006.01)	B41J	29/38	Z	5C062
H04N 1/00	(2006.01)	H04N	1/00	C	

審査請求 未請求　請求項の数 12　ＯＬ　（全 18 頁）

(21)出願番号　特願2009-283310 (P2009-283310)
(22)出願日　平成21年12月14日 (2009.12.14)

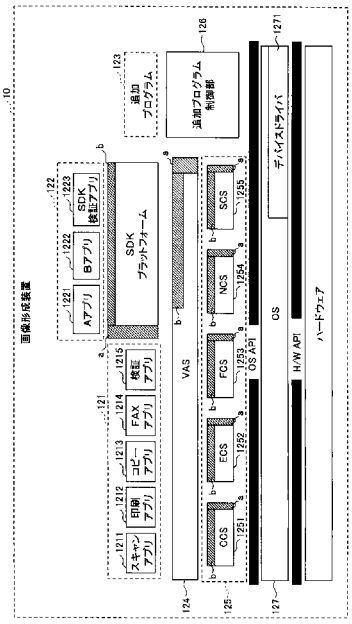
(71)出願人　000006747
株式会社リコー
東京都大田区中馬込１丁目３番６号
(74)代理人　100070150
弁理士　伊東　忠彦
(72)発明者　大橋　英樹
東京都大田区中馬込１丁目３番６号　株式会社リコー内
(72)発明者　大石　勉
東京都大田区中馬込１丁目３番６号　株式会社リコー内

Ｆターム(参考)　2C061 AP01 AP07 HJ08 HK11 HQ01
5C062 AA05 AB22 AB41 AB42 AC22
AC51 AC58 AE15

(54)【発明の名称】画像形成装置、機能追加方法、及びプログラム

(57)【要約】
【課題】新たなＡＰＩの追加を容易化すること。
【解決手段】アプリケーションプラットフォーム有する画像形成装置であって、前記アプリケーションプラットフォームは、実装が空であるインタフェースを有し、他のプログラムに動的に処理を割り込ませることが可能な追加プログラムを前記インタフェースに適用する適用手段を有する。
【選択図】図４

本発明の実施の形態における画像形成装置のソフトウェア構成例を示す図



【特許請求の範囲】
【請求項１】
アプリケーションプラットフォーム有する画像形成装置であって、
前記アプリケーションプラットフォームは、実装が空であるインタフェースを有し、
他のプログラムに動的に処理を割り込ませることが可能な追加プログラムを前記インタフェースに適用する適用手段を有する画像形成装置。
【請求項２】
前記アプリケーションプラットフォームは、複数の階層を有し、前記階層ごとに実装が空であるインタフェースを有し、
前記適用手段は、複数の前記追加プログラムをそれぞれが対応する前記階層における前記インタフェースに適用し、
上位の前記階層の前記追加プログラムは、下位の階層の前記インタフェースを呼び出す請求項１記載の画像形成装置。
【請求項３】
前記適用手段による前記追加プログラムの適用に応じ、当該画像形成装置のリソース情報を取得する取得手段と、
取得されたリソース情報を設定された判定条件と比較することにより、前記追加プログラムの適用の妥当性を判定する判定手段とを有する請求項１又は２記載の画像形成装置。
【請求項４】
前記取得手段は、前記追加プログラムの適用前より前記リソース情報を取得し、
前記判定手段は、前記適用前に取得された前記リソース情報と前記適用に応じて取得された前記リソース情報と前記判定条件に基づいて、前記追加プログラムの適用の妥当性を判定する請求項３記載の画像形成装置。
【請求項５】
アプリケーションプラットフォーム有する画像形成装置が実行する機能追加方法であって、
前記アプリケーションプラットフォームは、実装が空であるインタフェースを有し、
他のプログラムに動的に処理を割り込ませることが可能な追加プログラムを前記インタフェースに適用する適用手順を前記画像形成装置が実行する機能追加方法。
【請求項６】
前記アプリケーションプラットフォームは、複数の階層を有し、前記階層ごとに実装が空であるインタフェースを有し、
前記適用手順は、複数の前記追加プログラムをそれぞれが対応する前記階層における前記インタフェースに適用し、
上位の前記階層の前記追加プログラムは、下位の階層の前記インタフェースを呼び出す請求項５記載の機能追加方法。
【請求項７】
前記適用手順による前記追加プログラムの適用に応じ、当該画像形成装置のリソース情報を取得する取得手順と、
取得されたリソース情報を設定された判定条件と比較することにより、前記追加プログラムの適用の妥当性を判定する判定手順とを有する請求項５又は６記載の機能追加方法。
【請求項８】
前記取得手順は、前記追加プログラムの適用前より前記リソース情報を取得し、
前記判定手順は、前記適用前に取得された前記リソース情報と前記適用に応じて取得された前記リソース情報と前記判定条件に基づいて、前記追加プログラムの適用の妥当性を判定する請求項７記載の機能追加方法。
【請求項９】
画像形成装置が有するアプリケーションプラットフォームは、実装が空であるインタフェースを有し、
他のプログラムに動的に処理を割り込ませることが可能な追加プログラムを前記インタ

フェースに適用する適用手順を前記画像形成装置に実行させるためのプログラム。
【請求項１０】
前記アプリケーションプラットフォームは、複数の階層を有し、前記階層ごとに実装が空であるインタフェースを有し、
前記適用手順は、複数の前記追加プログラムをそれぞれが対応する前記階層における前記インタフェースに適用し、
上位の前記階層の前記追加プログラムは、下位の階層の前記インタフェースを呼び出す請求項９記載のプログラム。
【請求項１１】
前記適用手順による前記追加プログラムの適用に応じ、当該画像形成装置のリソース情報を取得する取得手順と、
取得されたリソース情報を設定された判定条件と比較することにより、前記追加プログラムの適用の妥当性を判定する判定手順とを前記画像形成装置に実行させる請求項９又は１０記載のプログラム。
【請求項１２】
前記取得手順は、前記追加プログラムの適用前より前記リソース情報を取得し、
前記判定手順は、前記適用前に取得された前記リソース情報と前記適用に応じて取得された前記リソース情報と前記判定条件に基づいて、前記追加プログラムの適用の妥当性を判定する請求項１１記載のプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、画像形成装置、機能追加方法、及びプログラムに関し、特にアプリケーションプラットフォームを有する画像形成装置、機能追加方法、及びプログラムに関する。
【背景技術】
【０００２】
近年、特に複合機又は融合機と呼ばれる画像形成装置の中には、ＡＰＩ（Application Program Interface）が外部に公開されたアプリケーションプラットフォーム（以下、単に「プラットフォーム」という。）を備えたものが存在する（例えば、特許文献１）。プラットフォーム上では、サードベンダ等によって多数のアプリケーションが開発されている。ユーザは、自らの業務等に適したアプリケーションを購入し、画像形成装置（以下、「機器」という。）にインストールすることで、業務の効率化等を図ることができる。
【発明の概要】
【発明が解決しようとする課題】
【０００３】
プラットフォームは，可能な限り多くの要望に対応するためにＡＰＩが用意されている。しかし、エンドユーザによる機器の利用形態が多様化し、様々なアプリケーションがプラットフォーム上において実装されている昨今において、機器の量産後におけるプラットフォームに対する変更要求が少なくない。
【０００４】
例えば、新たに開発されるアプリケーションにおいて、機器に関する情報を取得する必要が有る場合、プラットフォームが当該情報を取得するためのＡＰＩを有していないと、プラットフォームの開発者には、当該ＡＰＩの提供が要求される。したがって、このような個別要求が繰り返されると、プラットフォームの開発者に対する負担が非常に大きくなり、新たな開発へ工数を割けなくなってしまうという問題がある。
【０００５】
なお、理論的には、予め、機器に関する全ての情報を提供可能なようにＡＰＩを実装することも考えられるが、実際問題として、機器に関する情報は膨大であり、その全てを取得するためのＡＰＩを予め実装することは困難である。また、実際に使用されるか不明なＡＰＩを実装するのは、開発コストの観点からも現実的ではない。更に、機器は、汎用的

なデバイスインタフェース（例えば、ＵＳＢ（Universal Serial Bus）等）を備えており、当該デバイスインタフェースを介して接続された装置の制御等に関するＡＰＩを予め予測して実装するのも不可能に近い。

【０００６】

本発明は、上記の点に鑑みてなされたものであって、新たなＡＰＩの追加を容易化することのできる画像形成装置、機能追加方法、及びプログラムの提供を目的とする。

【課題を解決するための手段】

【０００７】

そこで上記課題を解決するため、本発明は、アプリケーションプラットフォーム有する画像形成装置であって、前記アプリケーションプラットフォームは、実装が空であるインタフェースを有し、他のプログラムに動的に処理を割り込ませることが可能な追加プログラムを前記インタフェースに適用する適用手段を有する。

【０００８】

このような画像形成装置では、新たなＡＰＩの追加を容易化することができる。

【発明の効果】

【０００９】

本発明によれば、新たなＡＰＩの追加を容易化することができる。

【図面の簡単な説明】

【００１０】

【図１】本発明の実施の形態における機器管理システムの構成例を示す図である。

【図２】追加プログラムを説明するための図である。

【図３】本発明の実施の形態における画像形成装置のハードウェア構成例を示す図である。

【図４】本発明の実施の形態における画像形成装置のソフトウェア構成例を示す図である。

【図５】新たなＡＰＩの追加例を説明するための図である。

【図６】既存のＡＰＩの変更例を説明するための図である。

【図７】検証アプリの機能構成例を示す図である。

【図８】テスト条件及び判定条件の設定処理の処理手順を説明するためのシーケンス図である。

【図９】検証アプリによる処理手順を説明するためのシーケンス図である。

【図１０】リソース情報の取得処理及び妥当性の判定処理の第一の例を説明するためのシーケンス図である。

【図１１】リソース情報の取得処理及び妥当性の判定処理の第二の例を説明するためのシーケンス図である。

【発明を実施するための形態】

【００１１】

以下、図面に基づいて本発明の実施の形態を説明する。図１は、本発明の実施の形態における機器管理システムの構成例を示す図である。図１の機器管理システムにおいて、画像形成装置１０と管理装置２０とは、オフィスにおけるＬＡＮ（Local Area Network）等のネットワーク４０（有線又は無線の別は問わない。）を介して接続されている。

【００１２】

画像形成装置１０は、コピー、ファクシミリ、プリンタ、及びスキャナ等の複数の機能を一台の筐体において実現する画像形成装置（複合機）である。但し、いずれか一つの機能を実現する機器であってもよい。

【００１３】

管理装置２０は、画像形成装置１０において利用されるプログラムに対して適用される追加プログラムの管理及び当該追加プログラムの画像形成装置１０への転送等を行うコンピュータである。本実施の形態において、追加プログラムとは、適用対象とされるプログラムの任意の箇所において、当該追加プログラムに定義された処理を動的に割り込ませる

ことのできるプログラムをいう。

【００１４】

図２は、追加プログラムを説明するための図である。図２において、５０１は、追加プログラム５０５が適用されるプログラムにおける仮想メモリ上における命令の配列を示す。プログラム５０１は、追加プログラム５０５が適用される前（通常実行時）は、命令１、２、３の順で処理を実行する。５０１ａは、プログラム５０１に追加プログラム５０５が適用された状態を示す。ここでは、命令１と命令２との間に追加プログラム５０５の処理を割り込ませる例が示されている。この場合、命令２がテーブル５０２への分岐命令に置き換えられる。テーブル５０２には、初期化処理、前処理（変数のスタックへの退避等）、追加プログラム５０５の呼び出し処理、後処理（スタックに退避されていた変数等の取り出し等）の後に命令２が実行され、プログラム５０１の命令３に戻るような定義がされている。

【００１５】

すなわち、追加プログラムが適用される場合、適用対象とされたプログラムの実行ステップが予め指定された箇所（追加位置）に到達すると、追加プログラムの処理が実行される。当該追加プログラムの処理が終了すると、適用対象とされたプログラムに処理制御が復帰する。その後、適用対象とされたプログラムは、追加位置より処理を再開する。追加プログラムには、適用対象のプログラムに割り込ませる処理の他、適用対象とするプログラム及び追加位置を識別するための情報が含まれている。

【００１６】

追加プログラム内では、適用対象とされるプログラムの変数等を参照可能である。したがって、追加プログラムによって、適用対象とされるプログラムの任意の箇所における変数の値等を示すログ情報を出力させるための処理や、バグを修正するための処理や、新たな機能を実現するための処理等を適用対象とされるプログラムに割り込ませることができる。

【００１７】

斯かる追加プログラムによれば、適用対象とされるプログラムについて、ソースコードの修正、コンパイル及びリンク、更に、再インストール等を行うことなく（すなわち、動的に）、ログ情報の出力、バグの修正、又は機能強化等を図ることができる。

【００１８】

なお、本実施の形態において、追加プログラムを適用対象のプログラムに適用し、実行可能な状態とすること、すなわち、追加プログラムをメモリ上にロードし、適用対象のプログラムにロードされた追加プログラムの分岐命令を挿入することを追加プログラムの「有効化」という。すなわち、追加プログラムは、画像形成装置１０内に転送されただけでは機能せず、有効化されることにより適用対象のプログラムに処理を割り込ませることができる。一方、追加プログラムの適用を解除することを追加プログラムの「無効化」という。なお、図２に示される技術は、公知技術である。

【００１９】

図３は、本発明の実施の形態における画像形成装置のハードウェア構成例を示す図である。図３において、画像形成装置１０は、ＣＰＵ１０１、ＲＡＭ１０２、ＲＯＭ１０３、ＨＤＤ１０４、スキャナ１０５、プリンタ１０６、オペレーションパネル１０７、ＳＤカードスロット１０８、ＵＳＢポート１０９、及びネットワークインタフェース１１０等のハードウェアを有する。

【００２０】

ＲＯＭ１０３には、各種のプログラムやプログラムによって利用されるデータ等が記録されている。ＲＡＭ１０２は、プログラムをロードするための記憶領域や、ロードされたプログラムのワーク領域等として用いられる。ＣＰＵ１０１は、ＲＡＭ１０２にロードされたプログラムを処理することにより、各種の機能を実現する。ＨＤＤ１０４には、プログラムやプログラムが利用する各種のデータ等が記録される。スキャナ１０５は、原稿より画像データを読み取るためのハードウェアである。プリンタ１０６は、画像データを印

刷用紙に印刷するためのハードウェアである。オペレーションパネル１０７は、ユーザからの入力の受け付けを行うめのボタン等の入力手段や、液晶パネル等の表示手段を備えたハードウェアである。ＳＤカードスロット１０８は、ＳＤカード５０に記録されたプログラムを読み取るために利用される。すなわち、画像形成装置１０では、ＲＯＭ１０３に記録されたプログラムだけでなく、ＳＤカード５０に記録されたプログラムもＲＡＭ１０２にロードされ、実行されうる。ＵＳＢポート１０９は、ＵＳＢ（Universal Serial Bus）インタフェース用の接続口（コネクタ）である。ネットワークインタフェース１１０は、ＬＡＮ（Local Area Network）等のネットワーク（有線又は無線の別は問わない。）を接続するためのハードウェアインタフェースである。

【００２１】

図４は、本発明の実施の形態における画像形成装置のソフトウェア構成例を示す図である。同図において、画像形成装置１０は、標準アプリ１２１、ＳＤＫアプリ１２２、ＳＤＫプラットフォーム１２３、仮想アプリケーションサービス（ＶＡＳ）１２４、コントロールサービス１２５、追加プログラム制御部１２６、及びＯＳ１２７等を有する。

【００２２】

標準アプリ１２１は、画像形成装置１０に標準的に（出荷時に予め）実装されているアプリケーションプログラムの集合である。同図では、スキャンアプリ１２１１、印刷アプリ１２１２、コピーアプリ１２１３、ＦＡＸアプリ１２１４、及び検証アプリ１２１５が例示されている。スキャンアプリ１２１１は、スキャンジョブを実行する。印刷アプリ１２１２は印刷ジョブを実行する。コピーアプリ１２１３は、コピージョブを実行する。ＦＡＸアプリ１２１４は、ＦＡＸの送信ジョブ又は受信ジョブを実行する。検証アプリ１２１５は、画像形成装置１０のアプリケーションプラットフォームに対して機能強化等が行われた場合に、当該機能強化等の妥当性を検証するための処理を実行する。なお、アプリケーションプラットフォームの範囲については後述する。

【００２３】

コントロールサービス１２５は、各種のハードウェアリソース等を制御するための機能（ＡＰＩ）を上位アプリケーション等（標準アプリ１２１、ＳＤＫプラットフォーム１２３）に対して提供するソフトウェアモジュール群である。同図では、オペレーションパネルコントロールサービス（ＯＣＳ）１２５１、エンジンコントロールサービス（ＥＣＳ）１２５２、ファクシミリコントロールサービス（ＦＣＳ）１２５３、ネットワークコントロールサービス（ＮＣＳ）１２５４、及びシステムコントロールサービス（ＳＣＳ）１２５５等が例示されている。ＯＣＳ１２５１は、オペレーションパネル１０７に関する制御を行うためのＡＰＩを提供する。ＥＣＳ１２５２は、プリンタ１０５やプリンタ１０６等の画像形成処理のエンジン部分関する制御を行うためのＡＰＩを提供する。ＦＣＳ１２５３は、ファクシミリに関する制御を行うためのＡＰＩを提供する。ＮＣＳ１２５４は、ネットワーク通信に関する制御を行うためのＡＰＩを提供する。ＳＣＳ１２５５は、画像形成装置１０内のシステム管理に関するＡＰＩや、画像形成装置１０に関する各種情報を取得するためのＡＰＩ等を提供する。

【００２４】

ＶＡＳ１２４は、コントロールサービス１２５と、その上位アプリケーション（標準アプリ１２１、ＳＤＫプラットフォーム１２３）との仲介を行う。具体的には、ＶＡＳ１２４は、上位アプリケーションからコントロールサービス１２５のＡＰＩをラッピングしたＡＰＩを上位アプリケーションに対して提供する。ＶＡＳ１２４によってコントロールサービス１２５がラッピングされることにより、コントロールサービス１２５のＡＰＩが上位アプリケーションから隠蔽される。その結果、コントロールサービス１２５のバージョンアップ等に伴うＡＰＩの変更がＶＡＳ１２４によって吸収され、当該バージョンアップ等に対する上位アプリケーションの互換性が確保される。

【００２５】

ＳＤＫアプリ１２２は、画像形成装置１０の出荷後において、画像形成装置１０の機能拡張を図るためのプラグインとして追加的に開発及びインストールされるアプリケーショ

ンプログラムである。同図では、ＳＤＫアプリ１２２として、Ａアプリ１２２１、Ｂアプリ１２２２、及びＳＤＫ検証アプリ１２２３が例示されている。Ａアプリ１２２１及びＢアプリ１２２２は、所定のサービスを提供するＳＤＫアプリ１２２である。例えば、Ａアプリ１２２１、ベンダＡによって開発されたＳＤＫアプリ１２２であり、Ｂアプリ１２２２は、ベンダＢによって開発されたＳＤＫアプリ１２２である。ＳＤＫ検証アプリ１２２３は、画像形成装置１０のアプリケーションプラットフォームに対して機能強化等が行われた場合に、ＳＤＫプラットフォーム１２３への影響という観点において、当該機能強化等の妥当性を検証するための処理を実行する。

【００２６】

ＳＤＫプラットフォーム１２３は、ＳＤＫアプリ１２２の実行環境を提供する。各ＳＤＫアプリ１２２は、ＳＤＫプラットフォーム１２３が提供するＡＰＩ（クラスライブラリ）利用して開発される。ＳＤＫプラットフォーム１２３は、コントロールサービス１２５によって提供されるＡＰＩに対し、機種に対する依存度がより低く、より開発効率の高いＡＰＩをＪａｖａ（登録商標）言語によって提供する。したがって、ＳＤＫアプリ１２２は、Ｊａｖａ（登録商標）言語によって実装される。なお、同図において、ＳＤＫプラットフォーム１２３は、Ｊａｖａ（登録商標）仮想マシンを含む。したがって、ＳＤＫアプリ１２２は、Ｊａｖａ（登録商標）標準のクラスライブラリを利用することもできる。

【００２７】

なお、ＳＤＫアプリ１２２及びＳＤＫプラットフォーム１２３は、ＳＤカード５０に記録されている。但し、ＳＤＫアプリ１２２及びＳＤＫプラットフォーム１２３は、携帯可能な他の記録媒体（例えば、ＵＳＢメモリやＣＤ－ＲＯＭ等）に記録されていてもよいし、ネットワークを介して配布されてもよい。

図４は、ＳＤＫアプリ１２２及びＳＤＫプラットフォーム１２３がＳＤカード５０から仮想メモリにロードされた状態が示されている。

【００２８】

追加プログラム制御部１２６は、管理装置２０より転送される追加プログラムを受信し、当該追加プログラムをＲＡＭ１０２に展開させる。また、追加プログラム制御部１２６は、管理装置２０より送信される指示に応じ、当該追加プログラムの有効化又は無効化等を実行する。

【００２９】

ＯＳ１２７は、いわゆるＯＳ（Operating System）であり、デバイスドライバ１２７１等を含む。画像形成装置１０上の各ソフトウェアは、ＯＳ１２７上においてプロセス又はスレッドとして動作する。デバイスドライバ１２７１は、ＵＳＢポート１０９等の汎用的なデバイスインタフェースを介して接続される装置の制御を行う。

【００３０】

なお、図４において、ＳＤＫアプリ１２２及びＳＤＫプラットフォーム１２３（Ｊａｖａ（登録商標）仮想マシンは除く）以外は、ネイティブコードである。すなわち、ＣＰＵ１０１が解釈可能なマシン語に変換されたプログラムである。

【００３１】

図４において、ＳＤＫプラットフォーム１２３が狭義のアプリケーションプラットフォームであるとすると、ＳＤＫプラットフォーム１２３、ＶＡＳ１２４、及びシステムコントロール１２５は、広義のアプリケーションプラットフォーム（アプリケーションプラットフォームを実現する手段）に相当する。ＳＤＫアプリ１２２の開発ベンダから直接見えるのは、ＳＤＫプラットフォーム１２３であるが、ＳＤＫプラットフォーム１２３が機能するためには、ＶＡＳ１２４及びシステムコントロール１２５が必要だからである。したがって、以下において、単にアプリケーションプラットフォームというとき、ＳＤＫプラットフォーム１２３、ＶＡＳ１２４、及びシステムコントロール１２５より構成される部分をいう。すなわち、本実施の形態において、アプリケーションプラットフォームは、上位層から順に、ＳＤＫプラットフォーム１２３、ＶＡＳ１２４、システムコントロール１２５といったように、複数の階層を有する。

【００３２】
アプリケーションプラットフォームに含まれる各階層は、符号ａ又は符号ｂが付された部分を有する。部分ａは、コーディング及びコンパイル等を行わずに実質的に新たなＡＰＩの追加を可能とするための実装部分である。部分ｂは、コーディング及びコンパイル等を行わずに、処理内容（実装内容）を変更可能な既存のＡＰＩの実装部分である。但し、部分ｂに関しては特別な実装は必要ではない。既存のＡＰＩの実施内容は、追加プログラムを適用することにより動的に変更可能だからである。強いて言えば、部分ｂは、追加プログラムの追加位置となるステップに相当する。
【００３３】
一方、コーディング及びコンパイル等を実行することなく、新たなＡＰＩを追加するのは、特にネイティブコードに関しては厳密には不可能である。そこで、本実施の形態では、新たなＡＰＩの追加を擬似的に実現する。具体的には、部分ａは、中身が空の関数又はメソッド（以下、「ダミー関数」という。）として実装しておく。空とは、実質的に空であればよく、完全に空である（すなわち、関数内に何も記載されていない）ことに限定されない。実質的に空であるとは、関数に特有の機能を実現するためのステップが記述されていないことをいう。例えば、関数が呼ばれたことを記録するログの記録ステップについては、関数に特有の機能を実現するためのステップではない。したがって、ログの記録ステップが含まれていたとしても、当該関数は実質的に空である。また、追加プログラムの追加位置を設けるためのダミーのステップについても、関数が空であることの阻害要件とはならない。
【００３４】
アプリケーションプラットフォームの各階層におけるダミー関数の数は、１つでもよいし複数でもよい。また、関数名は、ｆｕｎｃ１、ｆｕｎｃ２等、適当なものとすればよい。引数及び戻り値の型は、あらゆるデータ型のデータを格納又は参照可能なものとしておくとよい。
【００３５】
ダミー関数の実装は、追加プログラムによって行う。追加プログラムの追加位置を、ダミー関数としておき、追加プログラムをダミー関数に適用しておくことで、ダミー関数が呼び出された際に追加プログラムに実装された処理を実行させることができる。その結果、当初のアプリケーションプラットフォームには含まれていなかった機能を実現する新たなＡＰＩを、ＳＤＫプラットフォーム１２３、ＶＡＳ１２４、及びコントロールサービス１２５等に実質的に追加することができる。
【００３６】
なお、ダミー関数を実装する部分や追加プログラムの適用の対象とされる部分は、ネイティブコード（ＣＰＵ１０１が解釈可能なコード）に変換（コンパイル）されている部分に限定されてもよい。換言すれば、Ｊａｖａ（登録商標）によって実装されている部分は、新たなＡＰＩの追加や既存のＡＰＩの変更に際し、ダミー関数の実装や追加プログラムの適用は必ずしも必要とはされない。Ｊａｖａ（登録商標）によって実装されている部分は、拡張性が高く、新たなＡＰＩの追加も容易だからである。また、Ｊａｖａ（登録商標）言語が備える継承等によって既存のＡＰＩの処理内容は容易に変更可能だからである。但し、このことは、Ｊａｖａ（登録商標）によって実装されている部分に対してダミー関数が実装されたり、追加プログラムが適用されたりする実施形態を除外する趣旨ではない。
【００３７】
なお、アプリケーションプラットフォームのうち、ＳＤＫプラットフォーム１２３のＡＰＩ部分は、Ｊａｖａ（登録商標）によって実装された部分である。一方、ＶＡＳ１２４及びコントロールサービス１２５は、ネイティブコードに変換された部分である。
【００３８】
まず、新たなＡＰＩの追加例についてより具体的に説明する。図５は、新たなＡＰＩの追加例を説明するための図である。同図では、例えば、ＵＳＢポート１０９を介して、新

デバイス３０（例えば、Ｚ折りを実現する装置）が画像形成装置１０に接続された例が示されている。ここで、ＯＳ１２７１レベルでは、ｉｏｃｔｌ等の汎用的なＡＰＩによってデバイスドライバ１２７１を介して新デバイス３０を制御可能である。一方、アプリケーションプラットフォームには、新デバイス３０を制御するためのＡＰＩが用意されていないこととする。

【００３９】

この場合、ＳＣＳ１２５５の部分ａ（以下、「ＳＣＳ１２５５ａ」という。他の階層についても同様の命名規則に従う。）の一つのダミー関数（以下、「ＳＣＳダミー関数」という。）を追加位置とする追加プログラム１３１と、ＶＡＳ１２４ａの一つのダミー関数（以下、「ＶＡＳダミー関数」という。）を追加位置とする追加プログラム１３２とが実装される。実装された追加プログラム１３１及び１３２は、管理装置２０に保存され、管理される。また、ＳＤＫプラットフォーム１２３ａには、新デバイス３０の制御要求を受け付けるためのメソッドを有する新たなクラスのラスファイルが追加される。

【００４０】

ＳＣＳダミー関数に適用される追加プログラム１３１は、例えば、ＳＣＳダミー関数の引数値に応じて、新デバイス３０を制御するための引数値を指定してＯＳ１２７のｉｏｃｔｌ、ｒｅａｄ、又はｗｒｉｔｅ等の汎用的のＡＰＩを呼び出すような実装を有する。ＳＣＳダミー関数の引数には、例えば、新デバイス３０に対してどのような制御を行うかを示す値が指定される。

【００４１】

ＶＡＳダミー関数に適用される追加プログラム１３２は、ＶＡＳダミー関数の引数値に応じた引数値を指定してＳＣＳダミー関数を呼び出すような実装を有する。すなわち、上位の階層のダミー関数に適用される追加プログラムは、下位の階層のダミー関数を呼び出すように実装される。なお、ＶＡＳダミー関数に指定された引数値がそのままＳＣＳダミー関数の引数値とされてもよいし、ＶＡＳダミー関数に指定された引数値が変換されてＳＣＳダミー関数の引数値とされてもよい。後者は、例えば、ＶＡＳダミー関数の引数に、より抽象化又は汎用化された値を指定させたい場合等に有効である。

【００４２】

ＳＤＫプラットフォーム１２３ａとして追加されるクラスのメソッドは、指定された引数の内容に応じた引数値を指定してＶＡＳダミー関数を呼び出すような実装を有する。なお、当該クラスのメソッドに指定された引数値がそのままＶＡＳダミー関数の引数値とされてもよいし、当該メソッドに指定された引数値が変換されてＶＡＳダミー関数の引数値とされてもよい。

【００４３】

管理装置２０において、操作者によって追加プログラム１３１及び１３２について画像形成装置１０への転送が指示されると、管理装置２０は、追加プログラム１３１及び１３２を画像形成装置１０に転送する。画像形成装置１０の追加プログラム制御部１２６は、追加プログラム１３１及び１３２を受信すると、それぞれを有効化する。すなわち、追加プログラム１３１はＳＣＳダミー関数に適用され、追加プログラム１３２はＶＡＳダミー関数に適用された状態となる。

【００４４】

この状態で、ＳＤＫアプリ１２２は、新デバイス３０を制御又は利用可能となる。図５では、Ａアプリ１２２１が新デバイス３０を制御している例が示されている。すなわち、Ａアプリ１２２１は、ＳＤＫプラットフォーム１２３ａとして追加されたクラスのメソッドを呼び出す（Ｓ１１）。当該メソッドの引数には、Ａアプリ１２２１が要求する制御内容に応じた値が指定される。当該メソッドは、実装内容に従いＶＡＳダミー関数を呼び出す（Ｓ１２）。ＶＡＳダミー関数の呼び出しに応じ、追加プログラム１３２が実行される。その結果、実質的にＶＡＳダミー関数からＳＣＳダミー関数が呼び出される（Ｓ１３）。ＳＣＳダミー関数の呼び出しに応じ、追加プログラム１３１が実行される。その結果、実質的にＳＣＳダミー関数からＯＳ１２７のＡＰＩ（ｉｏｃｔｌ、ｒｅａｄ、又はｗｒｉ

ｔｅ等）が呼び出される（Ｓ１４）。いずれのＡＰＩが呼び出されるかは、ＳＣＳダミー関数に指定された引数値に依る。ＯＳ１２７は、呼び出されたＡＰＩに応じた制御をデバイスドライバ１２７１を介して新デバイス３０に対して行う（Ｓ１５）。

【００４５】

以上のように、新デバイス３０に対するＡＰＩがアプリケーションプラットフォームに用意されていなくても、ダミー関数と追加プログラムとを用いることによって、実質的に新デバイス３０を制御するためのＡＰＩをアプリケーションプラットフォームに追加することができる。

【００４６】

続いて、既存のＡＰＩの変更例について具体的に説明する。図６は、既存のＡＰＩの変更例を説明するための図である。同図の前提として、所定のハードウェア（例えば、スキャナ１０５、プリンタ１０６、又はオペレーションパネル１０７等）について、従前よりＳＣＳ１２５５には通知され、ＳＣＳ１２５５によって管理されていたパラメータ（以下、「パラメータＡ」という。）が有ったとする。しかし、パラメータＡは、ＳＣＳ１２５５の処理判断に用いられ、ＳＤＫアプリ１２２では必要とされていなかった。したがって、ＳＣＳ１２５５が有する、当該所定のハードウェアに関する情報を提供するためのＡＰＩ（以下、「情報取得関数Ｓ」という。）において、パラメータＡは、提供対象（戻り値の対象）に含まれていなかった。図６は、このような状況において、パラメータＡをＳＤＫアプリ１２２に参照可能とした例を示す。

【００４７】

この場合、情報取得関数Ｓを追加位置とする追加プログラム１３３が実装される。また、情報取得関数Ｓを呼び出すことにより取得される情報をＳＤＫプラットフォーム１２３等に提供するためのＶＡＳ１２４における既存の関数（以下、「情報取得関数Ｖ」という。）を追加位置とする追加プログラム１３４が実装される。実装された追加プログラム１３３及び１３４は、管理装置２０に保存され、管理される。

【００４８】

情報取得関数Ｓに適用される追加プログラム１３３は、例えば、既存の情報取得用関数Ｓの戻り値に対してパラメータＡを追加する実装を有する。例えば、戻り値のバイト数を増加させ、増加分の領域にパラメータＡを格納するようにする。また、情報取得関数Ｖに適用される追加プログラム１３４は、情報取得関数Ｓからの戻り値に含まれるようになるパラメータＡを、情報取得関数Ｖの戻り値に対して追加するための実装を有する。

【００４９】

また、ＳＤＫプラットフォーム１２３において、情報取得関数Ｖを呼び出すことにより取得される情報をＳＤＫアプリ１２２に提供するためのメソッド（以下、「情報取得メソッド」という。）を有する既存のクラスのサブクラスが実装される。当該サブクラスでは、情報取得メソッドがオーバーライドされる。オーバーライドの内容は、情報取得関数Ｖからの戻り値に含まれるようになるパラメータＡを、情報取得メソッドの戻り値に対して追加することである。

【００５０】

管理装置２０において、操作者によって追加プログラム１３３及び１３４について画像形成装置１０への転送が指示されると、管理装置２０は、追加プログラム１３３及び１３４を画像形成装置１０に転送する。画像形成装置１０の追加プログラム制御部１２６は、追加プログラム１３３及び１３４を受信すると、それぞれを有効化する。すなわち、追加プログラム１３３は情報取得関数Ｓの所定位置に適用され、追加プログラム１３４は情報取得関数Ｖの所定位置に適用された状態となる。

【００５１】

この状態で、ＳＤＫアプリ１２２は、パラメータＡを取得可能となる。図６では、Ａアプリ１２２１がパラメータＡを含む所定のデバイスの情報を取得する例が示されている。

【００５２】

パラメータＡを含む所定のデバイスの情報は、予めＳＣＳ１２５５に通知され、ＳＣＳ

１２５５によって管理されている（Ｓ２１）。Ａアプリ１２２１は、ＳＤＫプラットフォーム１２３の情報取得メソッドを呼び出す（Ｓ２２）。続いて、情報取得メソッドのオーバーライド部分によって、情報取得関数Ｖが呼び出される（Ｓ２３）。情報取得関数Ｖは、従前通り情報取得関数Ｓを呼び出す（Ｓ２４）。情報取得関数Ｓの処理の過程において、追加位置に到達すると追加プログラム１３３が実行される。追加プログラム１３３は、ＳＣＳ１２５５において管理されているパラメータＡを、戻り値に含める処理を実行する。例えば、追加プログラム１３３は、ＳＣＳ１２５５がパラメータＡを記録しているメモリ１０２よりパラメータＡの値を取得し、当該値を情報取得関数Ｓの戻り値に追加する。情報取得関数Ｓに処理の制御が戻ると、パラメータＡが追加された戻り値が情報取得関数Ｖに返却される（Ｓ２５）。続いて、情報取得関数Ｖの追加位置に到達すると、追加プログラム１３４が実行される。追加プログラム１３４は、情報取得関数Ｓの戻り値に含まれているパラメータＡを、情報取得関数Ｖの戻り値に追加する処理を実行する。情報取得関数Ｖに処理の制御が戻ると、パラメータＡが追加された戻り値が情報取得メソッドに返却される（Ｓ２６）。情報取得メソッドは、オーバーライド部分によって、パラメータＡを戻り値に含めてＡアプリ１２２１に返却する（Ｓ２７）。これにより、Ａアプリ１２２１は、パラメータＡを用いて処理を行うことが可能となる。

【００５３】

なお、パラメータＡの取得は、新たなＡＰＩの追加によって実現してもよい。その場合の実現方法は、図５の説明より自明であるため省略する。

【００５４】

続いて、新規ＡＰＩの追加又は既存のＡＰＩの変更等のための追加プログラムの適用（すなわち、機能強化等）の妥当性の検証について説明する。当該妥当性の検証は、検証アプリ１２１５又はＳＤＫ検証アプリ１２２３によって行われる。

【００５５】

図７は、検証アプリの機能構成例を示す図である。同図において、検証アプリ１２１５は、設定部１２１５Ａ、取得部１２１５Ｂ、及び判定部１２１５Ｃ等を有する。

【００５６】

設定部１２１５Ａは、テスト条件及び判定条件等の設定を受け付け、設定されたテスト条件及び判定条件をＨＤＤ１０４に記録する。テスト条件とは、いずれのリソースの使用状況（例えば、消費量を示す情報。以下、「リソース情報」という。）をどのようなタイミングで取得するかを示す情報である。すなわち、テスト条件は、リソース情報の取得方法を示す情報である。テスト条件の一例として、ＣＰＵの使用率を１秒間隔で取得する、仮想メモリの使用率（又は物理メモリの使用率）を１秒間隔で取得する等が挙げられる。仮想メモリの使用率（又は物理メモリの使用率）は、スワッピングの発生頻度を把握するための情報として用いることができる。スワッピングの頻度が高くなると、画像形成装置１０の性能は著しく劣化する傾向が有る。したがって、スワッピングの発生頻度を把握することは、妥当性の検証において重要な要素となりうる。

【００５７】

また、リソース情報を取得する期間（開始時刻及び終了時刻、又は開始時からの時間）を示す情報がテスト条件に含まれていてもよい。なお、リソース情報の取得対象とされるリソースは、一つに限定されない。また、リソースとは、ＲＡＭ１０２、ＣＰＵ１０１、ＨＤＤ１０４、電源等のハードウェアに限られず、ファイルディスクリプタや仮想メモリ等、ソフト的なものも含まれる。

【００５８】

判定条件とは、機能強化等の妥当性を判定するための基準を示す情報である。判定条件は、テスト条件に基づいて取得されるリソース情報に対する条件として設定される。判定条件の一例として、ＣＰＵの使用率の平均が追加プログラムの適用前と比較して＋１５％以下であること、仮想メモリの使用率が追加プログラムの適用前と比較して＋１０％以下であること、等が挙げられる。なお、判定条件は、必ずしも追加プログラムの適用前との比較でなくてもよい。例えば、ＣＰＵの使用率の平均がＸＸ以下であること。といったよ

うに、絶対値との比較を示すものであってもよい。

【００５９】

取得部１２１５Ｂは、テスト条件に基づいてリソース情報を取得する。判定部１２１５Ｃは、取得部１２１５Ｂによって取得されたリソース情報と判定条件とに基づいて妥当性を判定する。

【００６０】

なお、ＳＤＫ検証アプリ１２２３も、検証アプリ１２１５と同様の機能構成を有する。但し、ＳＤＫ検証アプリ１２２３は、Ｊａｖａ（登録商標）仮想マシン上で動作するため、Ｊａｖａ（登録商標）仮想マシンプロセスにおける仮想的なリソースのリソース情報に基づいて、妥当性を判定する。

【００６１】

以下、妥当性の検証に関する処理手順について説明する。図８は、テスト条件及び判定条件の設定処理の処理手順を説明するためのシーケンス図である。同図の処理は、追加プログラムを管理装置２０から画像形成装置１０に転送する前に実行しておく。

【００６２】

ステップＳ１０１において、例えば、管理者によってテスト条件及び判定条件が設定される。テスト条件及び判定条件の設定は、設定部１２１５Ａがオペレーションパネル１０７又は画像形成装置１０とネットワークを介して接続されるＰＣ（Personal Computer）等に表示させる設定画面を介して行われる。続いて、設定部１２１５Ａは、設定されたテスト条件及び判定条件をＨＤＤ１０４に記録（保存）する（Ｓ１０２）。

【００６３】

なお、図８の処理は、検証アプリ１２１５及びＳＤＫ検証アプリ１２２３のそれぞれについて実行される。

【００６４】

続いて、図９は、検証アプリによる処理手順を説明するためのシーケンス図である。

【００６５】

検証アプリ１２１５の取得部１２１５Ｂは、例えば、テスト条件等の設定に応じて、リソース情報の取得を開始し、取得されたリソース情報をＨＤＤ１０４に記録する（Ｓ２０１）。リソース情報の取得は、テスト条件に応じて定期的に行われうる。

【００６６】

その後、追加プログラム制御部１２６は、管理装置２０より追加プログラム及び当該追加プログラムの有効化の指示を受信する（Ｓ２０２）。続いて、追加プログラム制御部１２６は、受信された追加プログラムを有効化する（Ｓ２０３）。すなわち、当該追加プログラムが、適用対象のプログラムの追加位置に適用される。続いて、追加プログラム制御部１２６は、追加プログラムの有効化を検証アプリ１２１５に通知する（Ｓ２０４）。当該通知において、追加プログラムの識別子（プログラムＩＤ）が指定されてもよい。

【００６７】

追加プログラムの有効化が通知された後においても、検証アプリ１２１５は、テスト条件に従ったリソース情報の取得を継続する（Ｓ２０５）。但し、検証アプリ１２１５は、追加プログラムの有効化の通知前に取得されたリソース情報（Ｓ２０１において取得されたリソース情報）と、通知後に取得されたリソース情報とを区別可能なように記録する。例えば、両者のリソース情報が記録されるファイルを異なるものとしてもよいし、時系列に記録されるリソース情報の履歴中に、追加プログラムの有効化が行われたことを示す行を追加してもよい。

【００６８】

続いて、検証アプリ１２１５の判定部１２１５Ｃは、リソース情報と判定条件とに基づいて妥当性を判定する（Ｓ２０６）。例えば、判定条件が、追加プログラムの有効化前後のリソース情報の比較を示すものである場合、ステップＳ２０１において取得されたリソース情報とステップＳ２０５において取得されたリソース情報との差分に基づく値と、判定条件との比較に基づいて妥当性が判定される。この場合、追加プログラムの有効化前に

取得されたリソース情報と、有効化後に取得されたリソース情報との差分は、追加プログラムの適用による影響によるものとして扱われる。一方、判定条件が絶対値との比較を示すものである場合、ステップＳ２０５において取得されたリソース情報と当該絶対値との比較に基づいて妥当性が判定される。

【００６９】

なお、ステップＳ２０６は、テスト条件に基づいてリソース情報の取得期限を特定可能な場合は、当該期限の到来に応じて自動的に実行されればよい。テスト条件に基づいてリソース情報の期限を特定可能な場合とは、例えば、テスト条件に、取得回数（例えば、１０回）又は取得期間（例えば、３０分）等が指定されている場合である。一方、テスト条件に基づいてリソース情報の取得期限を特定できない場合は、追加プログラムの有効化から所定時間後に判定部１２１５Ｃが自動的に実行してもよいし、オペレーションパネル１０７等を介して入力される操作者の指示入力に応じて実行されてもよい。

【００７０】

続いて、判定部１２１５Ｃは、妥当性の判定結果（妥当であるか否かを示す情報）を追加プログラム制御部１２６に通知する（Ｓ２０７）。追加プログラム制御部１２６は、当該判定結果に応じた処理を実行する。例えば、当該判定結果が妥当でないことを示す場合、追加プログラム制御部１２６は、追加プログラムを無効化する（Ｓ２０８）。一方、当該判定結果が妥当であることを示す場合、追加プログラム制御部１２６は、追加プログラムを有効化したままとする。

【００７１】

続いて、リソース情報の取得処理（Ｓ２０５）及び妥当性の判定処理（Ｓ２０６）の詳細について説明する。

【００７２】

図１０は、リソース情報の取得処理及び妥当性の判定処理の第一の例を説明するためのシーケンス図である。同図の処理は、図９のステップＳ２０４における追加プログラムの有効化の通知に応じて開始される。

【００７３】

検証アプリ１２１５の取得部１２１５Ｂは、設定部１２１５Ａに対してテスト条件の取得を要求する（Ｓ３０１）。設定部１２１５Ａは、ＨＤＤ１０４に保存されているテスト条件を取得し、取得部１２１５Ｂに出力する（Ｓ３０２）。続いて、取得部１２１５Ｂは、テスト条件に応じてリソース情報を取得する。同図では、定期的にリソース情報が取得される（ポーリングされる）例が示されている。リソース情報がポーリングされる場合、ポーリング期間、ポーリング間隔、及びポーリング対象のリソースの識別情報等がテスト条件に含まれているのが好適である。

【００７４】

Ｓ３０３によって示される破線の矩形で囲まれた部分は、コントロールサービス１２５が有するリソース情報のポーリング用のＡＰＩを介してリソース情報を取得しているステップを示す。この場合、取得部１２１５Ｂは、ＶＡＳ１２４を介してコントロールサービス１２５に対してリソース情報のポーリングを要求する。コントロールサービスは、当該要求に応じてリソース情報をポーリングし、ポーリング結果を取得部１２１５Ｂに出力する。

【００７５】

また、Ｓ３０４によって示される破線の矩形で囲まれた部分は、取得部１２１５Ｂが直接的にＯＳ１２７対してリソース情報のポーリングを行っているステップを示す。ＯＳ１２７を介して取得されたリソース情報は、ポーリングのたびに取得部１２１５Ｂに返却される。

【００７６】

但し、ステップＳ３０３及びＳ３０４の処理内容は、テスト条件においていずれのリソースのリソース情報が取得対象とされているかに応じて異なりうる。例えば、コントロールサービス１２５を介して取得可能なリソース情報が取得対象でない場合、ステップＳ３

０３に示されるような処理手順は実行されない。

【００７７】

なお、図９のステップＳ２０１においても、ステップＳ３０１～Ｓ３０４と同様の処理手順が実行される。

【００７８】

リソース情報の取得が終了すると（例えば、テスト条件に指定されたリソースの取得期間が終了すると）、判定部１２１５Ｃは、判定条件の取得を設定部１２１５Ａに要求する（Ｓ３０５）。設定部１２１５Ａは、ＨＤＤ１０４に保存されている判定条件を取得し、判定部１２１５Ｃに出力する（Ｓ３０６）。続いて、判定部１２１５Ｃは、判定条件と、取得部１２１５Ｂによって取得されたリソース情報とに基づいて妥当性を判定する（Ｓ３０７）。

【００７９】

図１０に示されるようなポーリングによるリソース情報の取得（測定）は、瞬間的に発生する最大消費量を検出できない可能性があるが、単位時間当たりの平均値が重要となるようなリソース情報に好適である。例えば，ＣＰＵの使用率又は仮想メモリの使用率については、平均的にどの程度使用されたかが重要であるため、ポーリングによる取得が好適である。

【００８０】

なお、リソース情報をポーリングする場合、画像形成装置１０が実行するジョブ等に負荷をかけない程度にポーリング間隔を大きくしたり、又はポーリングによる負荷を考慮してリソース情報を補正したりしてもよい。

【００８１】

続いて、図１１は、リソース情報の取得処理及び妥当性の判定処理の第二の例を説明するためのシーケンス図である。テスト条件及び判定条件に応じて、図１０の変わりに図１１に示される処理が実行されてもよい。

【００８２】

ステップＳ４０１及びＳ４０２は、図１０のステップＳ３０１及びＳ３０２と同様である。続いて、検証アプリ１２１５の取得部１２１５Ｂは、例えば、ＯＳ１２７に対してコールバック関数と、当該コールバック関数を呼び出す条件（以下、「呼び出し条件」という。）とを登録する（Ｓ４０３）。呼び出し条件は、テスト条件に基づく。すなわち、図１１の場合のテスト条件は、例えば、「ＣＰＵの使用率がＮ％を超えた」等、いずれかのリソースのリソース情報が所定の状態となったことを示すものであることが想定される。この場合、呼び出し条件は、「ＣＰＵの使用率がＮ％を超えた」等、である。なお、それぞれ呼び出し条件が異なる複数のコールバック関数が登録されてもよい。

【００８３】

その後、ＯＳ１２７は、ハードウェア（Ｈ／Ｗ）からのリソースの状態の変化に応じて通知されるイベントに基づいて、登録されている呼び出し条件のうちの少なくとも一つが満たされたことを検知すると、当該呼び出し条件に対応するコールバック関数を呼び出す（Ｓ４０５）。

【００８４】

検証アプリ１２１５の判定部１２１５Ｃは、当該コールバック関数の呼び出しに応じ、判定条件の取得を設定部１２１５Ａに要求する（Ｓ４０６）。設定部１２１５Ａは、ＨＤＤ１０４に保存されている判定条件を取得し、判定部１２１５Ｃに出力する（Ｓ４０７）。続いて、判定部１２１５Ｃは、判定条件と、呼び出されたコールバック関数に係る呼び出し条件（テスト条件）とに基づいて妥当性を判定する（Ｓ４０８）。例えば、判定条件が、「ＣＰＵ使用率が７０％以下であること」であり、呼び出し条件が「ＣＰＵ使用率が７０％以下を超えた」である場合、コールバックの呼び出しにより判定条件は満たされていないことが検知されたことになる。したがって、この場合、妥当でないと判定される。

【００８５】

図１１に示されるようなイベントドリブン型のリソース情報の取得（測定）は、瞬間的

に発生する最大値が重要となるリソース情報に好適である。例えば，ファイルディスクリプタ数等の制限のあるリソース情報については，瞬間的にでも発生する最大値が重要となるため、イベントドリブン型による取得が好適である。

【００８６】

なお、図９～図１１は、検証アプリ１２１５について説明したが、ＳＤＫ検証アプリ１２２３についても同様でよい。

【００８７】

ところで、上記では、追加又は変更されたＡＰＩを呼び出すＳＤＫアプリ１２２（例えば、図５又は図６におけるＡアプリ１２２１）による当該ＡＰＩの呼び出しとは非同期にリソース情報の取得が行われる。そこで、当該ＡＰＩを呼び出すＳＤＫアプリ１２２自身が、当該ＡＰＩの呼び出し時にリソース情報の取得を実行し、取得されたリソース情報に基づいて妥当性を判定するようにしてもよい。又は、ＳＤＫアプリ１２２が、当該ＡＰＩの呼び出し時に、検証アプリ１２１５又はＳＤＫ検証アプリ１２２３に対して、ＡＰＩを呼び出すことを通知してもよい。検証アプリ１２１５又はＳＤＫ検証アプリ１２２３は、当該通知に応じてリソース情報を取得するようにしてもよい。

【００８８】

以上のようにすることにより、当該ＡＰＩの呼び出し時のリソース情報を適時的に取得し、適時的に取得されたリソース情報に基づいて妥当性を判定することができる。

【００８９】

上述したように本実施の形態における画像形成装置１０によれば、新たなＡＰＩの追加や既存のＰＩの変更等を容易化することができる。また、当該追加や変更による影響（妥当性）を適切に検証することができる。したがって、新たなＡＰＩの追加や既存のＰＩの変更等による画像形成装置１０の性能の顕著な劣化等を適切に防止することができる。

【００９０】

以上、本発明の実施例について詳述したが、本発明は斯かる特定の実施形態に限定されるものではなく、特許請求の範囲に記載された本発明の要旨の範囲内において、種々の変形・変更が可能である。

【符号の説明】

【００９１】

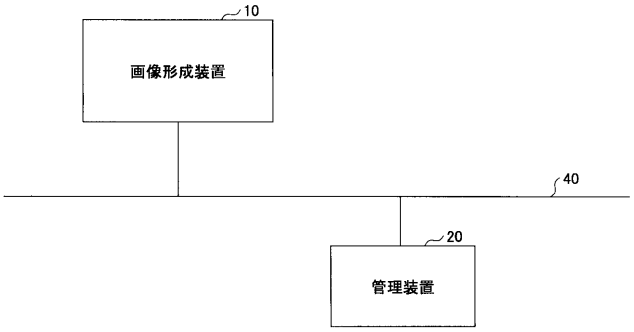
１０	画像形成装置
２０	管理装置
４０	ネットワーク
１０１	ＣＰＵ
１０２	ＲＡＭ
１０３	ＲＯＭ
１０４	ＨＤＤ
１０５	スキャナ
１０６	プリンタ
１０７	オペレーションパネル
１０８	ＳＤカードスロット
１０９	ＵＳＢポート
１１０	ネットワークインタフェース
１２１	標準アプリ
１２２	ＳＤＫアプリ
１２３	ＳＤＫプラットフォーム
１２４	ＶＡＳ
１２５	コントロールサービス
１２６	追加プログラム制御部
１２７	ＯＳ
１２１１	スキャンアプリ

１２１２　　印刷アプリ
１２１３　　コピーアプリ
１２１４　　ＦＡＸアプリ
１２１５　　検証アプリ
１２１５Ａ　　設定部
１２１５Ｂ　　取得部
１２１５Ｃ　　判定部
１２２１　　Ａアプリ
１２２２　　Ｂアプリ
１２２３　　ＳＤＫ検証アプリ
１２５１　　ＯＣＳ
１２５２　　ＥＣＳ
１２５３　　ＦＣＳ
１２５４　　ＮＣＳ
１２５５　　ＳＣＳ
１２７１　　デバイスドライバ

【先行技術文献】
【特許文献】
【００９２】
【特許文献１】特開２００８－１６０１３号公報

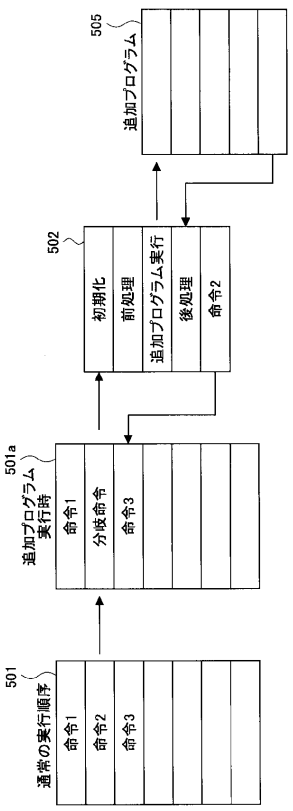
【図１】

本発明の実施の形態における機器管理システムの構成例を示す図



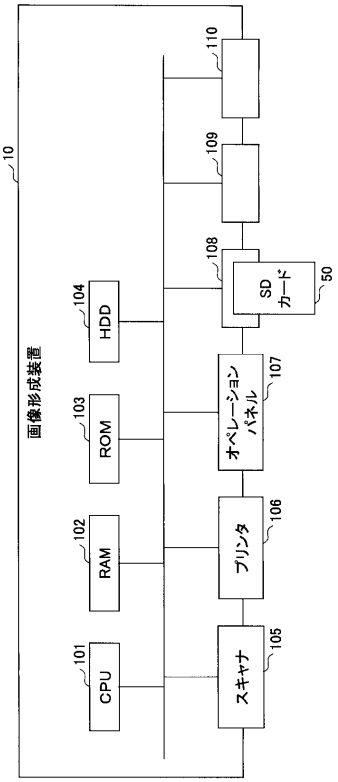
【図２】

追加プログラムを説明するための図



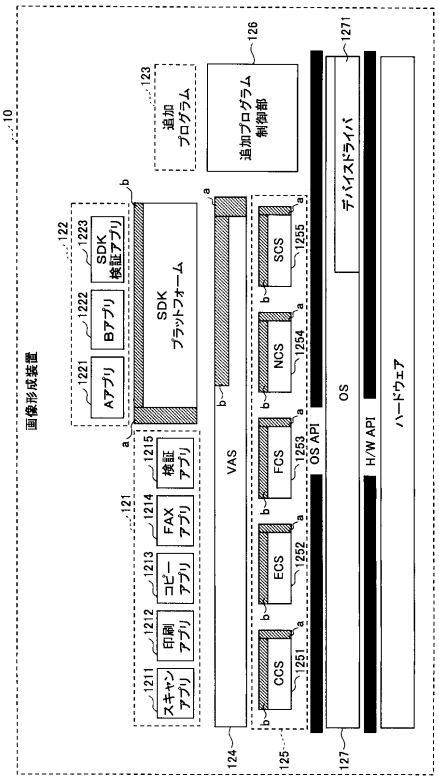
【図３】

本発明の実施の形態における画像形成装置のハードウェア構成例を示す図



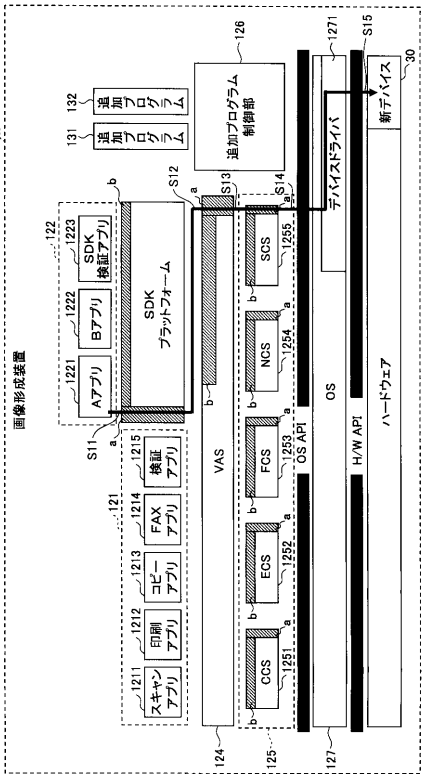
【図４】

本発明の実施の形態における画像形成装置のソフトウェア構成例を示す図



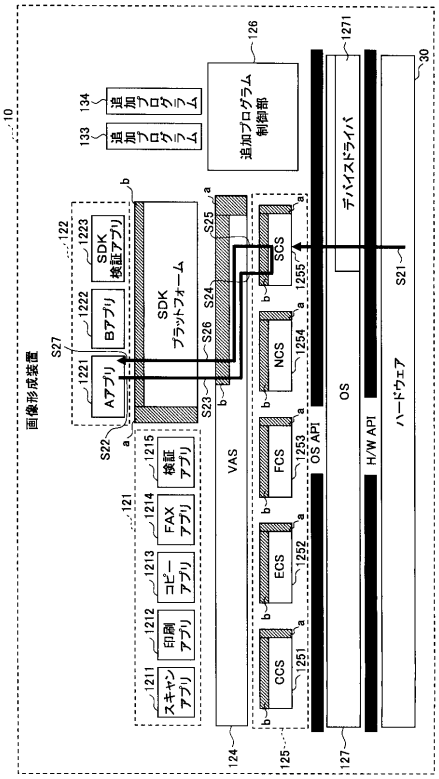
【図５】

新たなAPIの追加例を説明するための図



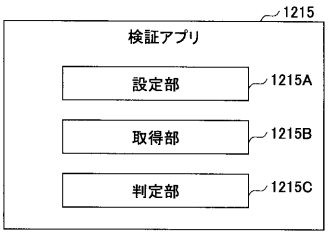
【図６】

既存のAPIの変更例を説明するための図



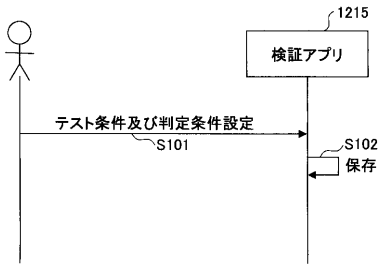
【図７】

検証アプリの機能構成例を示す図



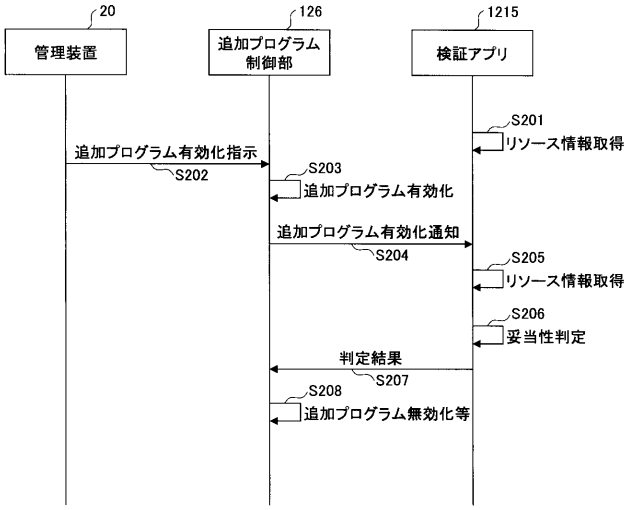
【図８】

テスト条件及び判定条件の設定処理の
処理手順を説明するためのシーケンス図



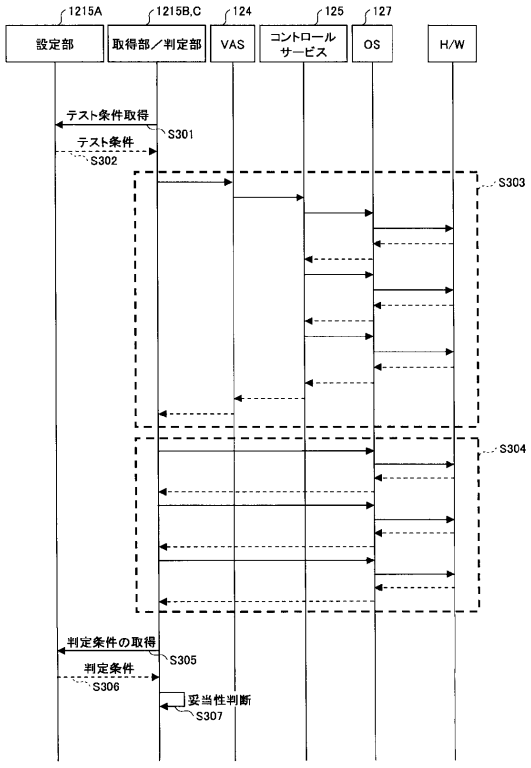
【図９】

検証アプリによる処理手順を説明するためのシーケンス図



【図１０】

リソース情報の取得処理及び妥当性の判定処理の
第一の例を説明するためのシーケンス図



【図１１】

リソース情報の取得処理及び妥当性の判定処理の
第二の例を説明するためのシーケンス図