(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2004-259096
(P2004-259096A)
(43)公開日　平成16年9月16日(2004.9.16)

(51)Int.Cl.[7]	F I			テーマコード（参考）
G06F 9/44	G06F	9/06	620K	5B076
G06F 9/445	G06F	9/06	650C	

審査請求 未請求　請求項の数 21　ＯＬ　（全 17 頁）

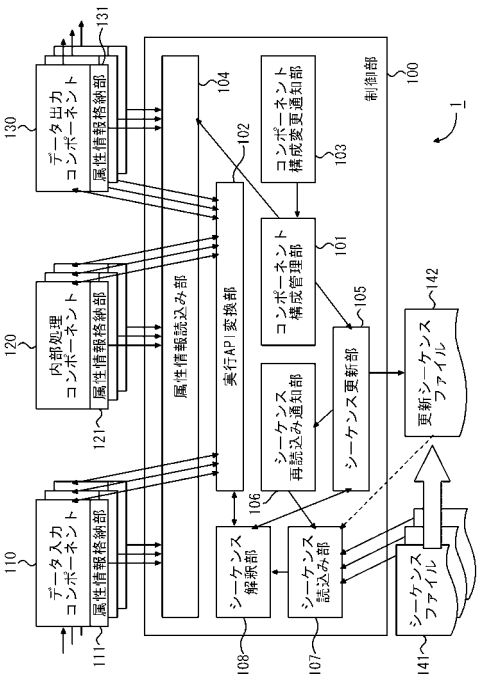
(21)出願番号　特願2003-50495(P2003-50495)
(22)出願日　平成15年2月27日(2003.2.27)

(71)出願人　000005496
富士ゼロックス株式会社
東京都港区赤坂二丁目１７番２２号
(74)代理人　100087480
弁理士　片山　修平
(74)代理人　100098497
弁理士　片寄　恭三
(72)発明者　原田　政彦
神奈川県海老名市本郷２２７４番地　富士ゼロックス株式会社海老名事業所内

Ｆターム(参考)　5B076 AB18　DD06　DD10

(54)【発明の名称】情報処理装置及び画像処理装置、並びに機能追加方法、そのプログラム及びそのプログラムを記録した記録媒体

(57)【要約】
【課題】容易に機能の更新をすることが可能な情報処理装置及び画像処理装置、並びに機能追加方法、そのプログラム及びそのプログラムを記録した記録媒体を提供する。
【解決手段】例えばテキスト形式で作成されたシーケンスファイル１４１に従って各コンポーネント（１１０，１２０，１３０）を制御部１００が起動・実行制御する情報処理装置において、新たな機能実行単位が追加される。この際、機能実行単位には属性情報も対応づけられて追加される。属性情報には機能実行単位のカテゴリ属性やこれが持つ各機能の前後の依存関係が含まれており、制御部１００は、この属性情報に基づいてシーケンスファイル１４１を書き換える。また、書き換えが完了すると、更新シーケンスファイル１４２を再読込みし、追加された機能実行単位が組み込まれた状態へ再度、起動し直す。
【選択図】　図１



【特許請求の範囲】
【請求項１】
特定の機能を実現するための機能実行手段と、該機能実行手段の実行手順が記述されたシーケンスファイルに基づいて該機能実行手段を制御する制御手段とを有する情報処理装置であって、
前記機能実行手段は、実行手順を特定するための属性情報と対応づけられており、
前記制御手段は、新たに追加された機能実行手段の前記属性情報に基づいて前記シーケンスファイルを書き換えることを特徴とする情報処理装置。
【請求項２】
前記制御手段は、前記シーケンスファイルを更新した後、該シーケンスファイルを再読込みすることで前記追加された機能実行手段を動作させることを特徴とする請求項１記載の情報処理装置。
【請求項３】
組み込まれている前記機能実行手段の構成を前記属性情報に基づいて管理するための機能管理テーブルを有し、
前記制御手段は、現在組み込まれている機能実行手段の構成と、前記機能管理テーブルにおける前記機能実行手段の構成とを比較することで、前記追加された機能実行手段を検知することを特徴とする請求項１又は２記載の情報処理装置。
【請求項４】
前記シーケンスファイルはテキスト形式で保存されていることを特徴とする請求項１から３の何れか１項に記載の情報処理装置。
【請求項５】
前記機能実行手段は、所定の表示手段に表示させる画面を描画する画面描画機能実行手段を含み、
前記制御手段は、前記シーケンスファイルに基づいて前記表示手段に表示させる画面を描画して表示することを特徴とする請求項１から４の何れか１項に記載の情報処理装置。
【請求項６】
前記属性情報は、該属性情報と対応づけられた前記機能実行手段が機能するために必要な前及び／又は後の依存関係を含むことを特徴とする請求項１から５の何れか１項に記載の情報処理装置。
【請求項７】
前記属性情報はシーケンス言語で記述されており、
前記制御手段は、前記機能実行手段が追加された場合、前記シーケンスファイルに前記属性情報を付加することで、該シーケンスファイルを更新することを特徴とする請求項１から６の何れか１項に記載の情報処理装置。
【請求項８】
請求項１から７の何れか１項に記載の前記情報処理装置を有し、
前記機能実行手段は、外部から画像データを入力するための画像データ入力手段と、該画像データに所定の処理を施すための画像データ処理手段と、処理した該画像データを出力するための画像データ出力手段とを含むことを特徴とする画像形成装置。
【請求項９】
実行手順が記述されたシーケンスファイルに基づいて動作が制御される該機能実行手段を追加するための機能追加方法において、
実行手順を特定するための属性情報が対応づけられた機能実行手段が追加されたことを検知する追加機能検知ステップと、
前記属性情報に基づいて前記シーケンスファイルを更新するシーケンスファイル更新ステップと
を有することを特徴とする機能追加方法。
【請求項１０】
前記シーケンスファイル更新ステップの後、前記シーケンスファイルを再読込みすること

で前記追加された機能実行手段を動作させる再読込みステップを有することを特徴とする請求項９記載の機能追加方法。
【請求項１１】
前記追加機能検知ステップは、組み込まれている前記機能実行手段の構成を前記属性情報に基づいて管理するための機能管理テーブルと、現在組み込まれている機能実行手段の構成とを比較することで、追加された前記機能実行手段を検知することを特徴とする請求項９又は１０記載の機能追加方法。
【請求項１２】
前記機能実行手段は、所定の表示手段に表示させる画面を描画する画面描画機能実行手段を含み、
該画面描画機能実行手段に基づいて前記表示手段に表示するための画面を作成する表示画面作成ステップを有することを特徴とする請求項９から１１の何れか１項に記載の機能追加方法。
【請求項１３】
前記属性情報は、該属性情報と対応づけられた前記機能実行手段が機能するために必要な前及び／又は後の依存関係を含み、
前記シーケンスファイル更新ステップは、前記前及び／又は後の依存関係に基づいて前記シーケンスファイルを更新することを特徴とする請求項９から１２の何れか１項に記載の機能追加方法。
【請求項１４】
前記属性情報はシーケンス言語で記述されており、
前記シーケンスファイル更新ステップは、前記属性情報を前記シーケンスファイルに付加することで、該シーケンスファイルを更新することを特徴とする請求項９から１３の何れか１項に記載の機能追加方法。
【請求項１５】
特定の機能を実現するための機能実行手段と、該機能実行手段の実行手順が記述されたシーケンスファイルに基づいて該機能実行手段を制御する制御手段とを実現するコンピュータを機能させるためのプログラムであって、
実行手順を特定するための属性情報が対応づけられた機能実行手段が追加されたことを検知する追加機能検知処理と、
前記属性情報に基づいて前記シーケンスファイルを更新するシーケンスファイル更新処理と
を前記コンピュータに実行させるためのプログラム。
【請求項１６】
前記シーケンスファイル更新処理の後、前記シーケンスファイルを再読込みすることで前記追加された機能実行手段を動作させる再読込み処理を前記コンピュータに実行させるための請求項１５記載のプログラム。
【請求項１７】
前記追加機能検知処理は、組み込まれている前記機能実行手段の構成を前記属性情報に基づいて管理するための機能管理テーブルと、現在組み込まれている機能実行手段の構成とを比較することで、追加された前記機能実行手段を検知することを特徴とする請求項１５又は１６記載のプログラム。
【請求項１８】
前記機能実行手段は、所定の表示手段に表示させる画面を描画する画面描画機能実行手段を含み、
該画面描画機能実行手段に基づいて前記表示手段に表示するための画面を作成する表示画面作成処理を前記コンピュータに実行させるための請求項１５から１７の何れか１項に記載のプログラム。
【請求項１９】
前記属性情報は、該属性情報と対応づけられた前記機能実行手段が機能するために必要な

前及び／又は後の依存関係を含み、
前記シーケンスファイル更新処理は、前記前及び／又は後の依存関係に基づいて前記シーケンスファイルを更新することを特徴とする請求項１５から１８の何れか１項に記載のプログラム。
【請求項２０】
前記属性情報はシーケンス言語で記述されており、
前記シーケンスファイル更新処理は、前記属性情報を前記シーケンスファイルに付加することで、該シーケンスファイルを更新することを特徴とする請求項１５から１９の何れか１項に記載のプログラム。
【請求項２１】
請求項１５から２０の何れか１項に記載の前記プログラムを記録した記録媒体。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は、情報処理装置及び画像処理装置、並びに機能追加方法、そのプログラム及びそのプログラムを記録した記録媒体に関し、特定の機能・役割を割り当てられた複数の機能実行単位群が強調して動作する情報処理装置及び画像処理装置、並びに機能追加方法、そのプログラム及びそのプログラムを記録した記録媒体に関する。
【０００２】
【従来の技術】
従来、例えばプリンタに代表される情報処理装置においては、画像データを外部機器から入力し、画像データを出力するという単純な構成であった。このような従来の情報処理装置では、データを入力する入力手段、データを変換する処理手段、出力する出力手段は、それぞれ１つに固定されており、データの流れも単純なため、全体の制御も容易であった。
【０００３】
図１３は、従来の情報処理装置の第１の例を示す構成図である。この例では、上述のように、入力Ａと処理Ｂと出力Ｃとが１つずつに固定されている例を示している。これらがそれぞれ機能実行単位となる。データは入力Ａ，処理Ｂ，出力Ｃの順に渡される。また、各機能実行単位で制御情報等がやりとりされる。このように端ン純な構成では、各機能実行単位において、それぞれの機能実行単位内での動作を制御する制御手段が設けられており（各構成の右下に「制御」として図示）、それぞれの機能実行単位が独立して動作可能に構成されている。また、それぞれの機能実行単位間でのデータや制御情報のやりとりは、それぞれの構成間毎に取り決められていた。
【０００４】
ところが近年、画像処理装置等の情報処理装置における多機能化は目覚ましく、例えば入力手段も外部機器と１対１で一つの物理的なインタフェースでローカルに接続されるだけでなく、複数の物理的なインタフェースを持つようになった。具体的には、ＩＥＥＥ１２８４インタフェースのみの接続から、ＵＳＢ（Ｕｎｉｖｅｒｓａｌ Ｓｅｒｉａｌ Ｂｕｓ）及びＩＥＥＥ１３９４インタフェース等が追加されるケースである。また、ローカルな接続だけでなく、例えばローカル・エリア・ネットワーク（ＬＡＮ）等に接続するインタフェースを有し、ネットワークに接続して共有する形態も追加され、入力手段が複数存在することになった。また、処理手段も様々なデータフォーマットに対応するため、複数の処理手順を持つようになった。更に、スキャナやファクシミリ・モデムを取り込んだ複合機になると、出力先も複数存在することになる。結果的に、全体の制御は通常の処理でさえ複雑なものになり、エラー対応などの例外処理は機能の増加に伴い例外事項が増大し、そのための制御が非常に複雑なものとなっている。
【０００５】
図１４は、従来の情報処理装置の第２の例を示す構成図である。この例では、複雑化した装置の例として、入力手段として入力Ａと入力Ｂ、処理手段として内部処理Ｃと内部処理

Ｄと内部処理Ｅ、出力手段として出力Ｆと出力Ｇ等の機能実行単位が存在する例を示している。データは入力Ａ或いは入力Ｂから入力され、内部処理Ｃ，Ｄ，Ｅの何れか或いは複数により処理された後、出力Ｆ或いは出力Ｇから出力される。また、各機能実行単位で制御情報等がやりとりされている。

【０００６】

このように複雑な構成においても、従来は各機能実行単位において、それぞれの機能実行単位内での動作を制御する制御手段が設けられており（各構成の右下に「制御」して図示）、それぞれの機能実行単位が独立した実行単位として動作可能に構成されている。また、それぞれの機能実行単位は他の機能実行単位と密接に結合しており、それぞれの構成間でのデータや制御情報のやりとりは、それぞれの構成間毎に取り決められていた。

【０００７】

図１３及び図１４に示すような各構成毎に制御手段を設けた制御方式では、動的な制御が各機能実行単位に散在しているため、各機能実行単位毎に同様な内容の制御を重複して実装していることになる。このため、実装量の増加を招き、保守性を下げる原因となっている。又、一部を変更しようとした場合に、その変更作業に多くの工数がかかり、また、他の機能実行単位の動作に矛盾が無いことを検証するにも多くの工数がかかることになる。更に、動作中に不具合が生じた場合も、制御がそれぞれの機能実行単位に散在しているため、原因の特定が困難である。結果的に必要以上に構成が複雑になり、新たな機能の追加や機能の変更などが極めて困難な、柔軟性の低いシステム担ってしまっていた。

【０００８】

このような問題を解決する技術として、例えば構成間の出力状態と遷移とを階層的を定義しておき、この定義に基づいて記述されたスクリプトに応じて、各構成間の出力状態の同期を制御する方法が存在する（例えば以下に示す特許文献１参照）。

【０００９】

【特許文献１】

特開平６－２１４６９０号公報

【００１０】

【発明が解決しようとする課題】

しかしながら、上記した方法では、一般的に制御の記述が他のオブジェクトと同じプログラミング言語（例えばＣ／Ｃ＋＋やＪａｖａ（登録商標）等）によるコードとして記述されていなければならず、また、開発環境においてコンパイル・リング等の手順を踏む必要がある。

【００１１】

更に、実装直後のデバック・フェーズ等では、制御手段そのものの品質が保たれておらず、頻繁な制御手順の入れ換え等により制御の変更が生じる場合がある。このような場合、従来技術では制御手順の変更の度に、プログラム言語で記述された制御文を変更し、これに基づいてコンパイル・リンク等の処理を実行する必要があったため、工数が非常に増加し、開発効率が阻害されてしまう。

【００１２】

また、制御を局所化するということは、その制御部の局所的な複雑化を招く。このため、通常のプログラム言語では制御の理解・保守性に問題が生じる場合がある。特に操作画面等のＵＩ（Ｕｓｅｒ　Ｉｎｔｅｒｆａｃｅ）に関しては、顧客とのインタフェースを実現するために見栄えや操作性等の要求が高い。そのため、開発とテストとの繰り返しになることが多いが、ＵＩ画面の描画をプログラム言語で実現した場合、コンパイル・リンク等のターンアラウンドが問題になる。

【００１３】

また、新たに開発した機能を顧客先に設置した製品に追加する場合、機能そのものはネットワーク等から製品へダウンロードすることが可能であるが、全体の動作を制御するスクリプトは、制御スクリプト全体をその都度作成し直し、これを機能の追加とは別に手動で更新する必要があった。

【００１４】
本発明は、このような問題に鑑みてなされたもので、容易に機能の更新をすることが可能な情報処理装置及び画像処理装置、並びに機能追加方法、そのプログラム及びそのプログラムを記録した記録媒体を提供することを目的とする。
【００１５】
【課題を解決するための手段】
かかる目的を達成するために、本発明は、請求項１記載のように、特定の機能を実現するための機能実行手段と、該機能実行手段の実行手順が記述されたシーケンスファイルに基づいて該機能実行手段を制御する制御手段とを有する情報処理装置であって、前記機能実行手段が、実行手順を特定するための属性情報と対応づけられており、前記制御手段が、新たに追加された機能実行手段の前記属性情報に基づいて前記シーケンスファイルを書き換えるように構成される。機能実行単位である機能実行手段の実行手順をシーケンスファイルで一括管理できるように構成しておくことで、機能追加時に、このシーケンスファイルを更新するだけで追加された機能実行手段が管理できるように構成できる。更に、この際、実行手順が記述された属性情報を機能実行手段に対応づけておくことで、容易にシーケンスファイルを更新することが可能となる。これにより、例えばネットワーク等を介した遠隔操作でも、容易に機能の更新をすることが可能となる。
【００１６】
また、請求項１記載の前記制御手段は、請求項２記載のように、前記シーケンスファイルを更新した後、該シーケンスファイルを再読込みすることが好ましい。これにより、容易且つ短時間で前記追加された機能実行手段を動作させることが可能となる。
【００１７】
また、請求項１又は２記載の前記情報処理装置は、例えば請求項３記載のように、組み込まれている前記機能実行手段の構成を前記属性情報に基づいて管理するための機能管理テーブルを有し、前記制御手段が、現在組み込まれている機能実行手段の構成と、前記機能管理テーブルにおける前記機能実行手段の構成とを比較することで、前記追加された機能実行手段を検知するように構成されても良い。これにより、随時、容易に機能の追加を検知することが可能となる。
【００１８】
また、請求項１から３の何れか１項に記載の前記シーケンスファイルは、請求項４記載のように、テキスト形式で保存されていることが好ましい。テキスト形式で保存しておくことで、プラットホームに依存せずにシーケンスファイルを読み書きすることが可能となる。
【００１９】
また、請求項１から４の何れか１項に記載の前記機能実行手段は、請求項５記載のように、所定の表示手段に表示させる画面を描画する画面描画機能実行手段を含み、前記制御手段が前記シーケンスファイルに基づいて前記表示手段に表示させる画面を描画して表示するように構成されることが好ましい。表示画面も更新できるように構成することで、機能の追加に応じて的確に操作画面を変更することが可能となる。
【００２０】
また、請求項１から５の何れか１項に記載の前記属性情報は、例えば請求項６記載のように、該属性情報と対応づけられた前記機能実行手段が機能するために必要な前及び／又は後の依存関係を含むように構成することもできる。これにより、容易にシーケンスファイルを更新することが可能となる。
【００２１】
また、請求項１から６の何れか１項に記載の前記情報処理装置は、例えば請求項７記載のように、前記属性情報がシーケンス言語で記述されており、前記制御手段が、前記機能実行手段が追加された場合、前記シーケンスファイルに前記属性情報を付加することで、該シーケンスファイルを更新するように構成することもできる。これにより、容易にシーケンスファイルを更新することができる。

【００２２】
また、本発明による画像処理装置は、請求項８記載のように、請求項１から７の何れか１項に記載の前記情報処理装置を有し、前記機能実行手段が、外部から画像データを入力するための画像データ入力手段と、該画像データに所定の処理を施すための画像データ処理手段と、処理した該画像データを出力するための画像データ出力手段とを含むように構成される。このように、上記の情報処理装置を用いて画像処理装置を構成することができる。
【００２３】
また、本発明は、請求項９記載のように、実行手順が記述されたシーケンスファイルに基づいて動作が制御される該機能実行手段を追加するための機能追加方法において、実行手順を特定するための属性情報が対応づけられた機能実行手段が追加されたことを検知する追加機能検知ステップと、前記属性情報に基づいて前記シーケンスファイルを更新するシーケンスファイル更新ステップとを有するように構成される。機能実行単位である機能実行手段の実行手順をシーケンスファイルで一括管理できるように構成しておくことで、機能追加時に、このシーケンスファイルを更新するだけで追加された機能実行手段が管理できるように構成できる。更に、この際、実行手順が記述された属性情報を機能実行手段に対応づけておくことで、容易にシーケンスファイルを更新することが可能となる。これにより、例えばネットワーク等を介した遠隔操作でも、容易に機能の更新をすることが可能となる。
【００２４】
また、請求項９記載の前記機能追加方法は、請求項１０記載のように、前記シーケンスファイル更新ステップの後、前記シーケンスファイルを再読込みすることで前記追加された機能実行手段を動作させる再読込みステップを有することが好ましい。これにより、容易且つ短時間で前記追加された機能実行手段を動作させることが可能となる。
【００２５】
また、請求項９又は１０記載の前記追加機能検知ステップは、例えば請求項１１記載のように、組み込まれている前記機能実行手段の構成を前記属性情報に基づいて管理するための機能管理テーブルと、現在組み込まれている機能実行手段の構成とを比較することで、追加された前記機能実行手段を検知するように構成することもできる。これにより、随時、容易に機能の追加を検知することが可能となる。
【００２６】
また、請求項９から１１の何れか１項に記載の前記機能追加方法は、請求項１２記載のように、前記機能実行手段が所定の表示手段に表示させる画面を描画する画面描画機能実行手段を含み、該画面描画機能実行手段に基づいて前記表示手段に表示するための画面を作成する表示画面作成ステップを有するように構成されることが好ましい。表示画面も更新できるように構成することで、機能の追加に応じて的確に操作画面を変更することが可能となる。
【００２７】
また、請求項９から１２の何れか１項に記載の前記属性情報は、例えば請求項１３記載のように、該属性情報と対応づけられた前記機能実行手段が機能するために必要な前及び／又は後の依存関係を含み、前記シーケンスファイル更新ステップが前記前及び／又は後の依存関係に基づいて前記シーケンスファイルを更新するように構成することもできる。これにより、容易にシーケンスファイルを更新することが可能となる。
【００２８】
また、請求項９から１３の何れか１項に記載の前記機能追加方法は、例えば請求項１４記載のように、前記属性情報がシーケンス言語で記述されており、前記シーケンスファイル更新ステップが前記属性情報を前記シーケンスファイルに付加することで、該シーケンスファイルを更新するように構成することもできる。これにより、容易にシーケンスファイルを更新することが可能となる。
【００２９】

また、本発明は、請求項１５記載のように、特定の機能を実現するための機能実行手段と、該機能実行手段の実行手順が記述されたシーケンスファイルに基づいて該機能実行手段を制御する制御手段とを実現するコンピュータを機能させるためのプログラムであって、実行手順を特定するための属性情報が対応づけられた機能実行手段が追加されたことを検知する追加機能検知処理と、前記属性情報に基づいて前記シーケンスファイルを更新するシーケンスファイル更新処理とを前記コンピュータに実行させる。機能実行単位である機能実行手段の実行手順をシーケンスファイルで一括管理できるように構成しておくことで、機能追加時に、このシーケンスファイルを更新するだけで追加された機能実行手段が管理できるように構成できる。更に、この際、実行手順が記述された属性情報を機能実行手段に対応づけておくことで、容易にシーケンスファイルを更新することが可能となる。これにより、例えばネットワーク等を介した遠隔操作でも、容易に機能の更新をすることが可能となる。

【００３０】

また、請求項１５記載の前記プログラムは、請求項１６記載のように、前記シーケンスファイル更新処理の後、前記シーケンスファイルを再読込みすることで前記追加された機能実行手段を動作させる再読込み処理を前記コンピュータに実行させることが好ましい。これにより、容易且つ短時間で前記追加された機能実行手段を動作させることが可能となる。

【００３１】

また、請求項１５又は１６記載の前記追加機能検知処理は、例えば請求項１７記載のように、組み込まれている前記機能実行手段の構成を前記属性情報に基づいて管理するための機能管理テーブルと、現在組み込まれている機能実行手段の構成とを比較することで、追加された前記機能実行手段を検知するように構成されてもよい。これにより、随時、容易に機能の追加を検知することが可能となる。

【００３２】

また、請求項１５から１７の何れか１項に記載の前記プログラムは、請求項１８記載のように、前記機能実行手段が所定の表示手段に表示させる画面を描画する画面描画機能実行手段を含み、該画面描画機能実行手段に基づいて前記表示手段に表示するための画面を作成する表示画面作成処理を前記コンピュータに実行させるように構成されることが好ましい。表示画面も更新できるように構成することで、機能の追加に応じて的確に操作画面を変更することが可能となる。

【００３３】

また、請求項１５から１８の何れか１項に記載の前記プログラムは、例えば請求項１９記載のように、前記属性情報が該属性情報と対応づけられた前記機能実行手段が機能するために必要な前及び／又は後の依存関係を含み、前記シーケンスファイル更新処理が前記前及び／又は後の依存関係に基づいて前記シーケンスファイルを更新するように構成することもできる。これにより、容易にシーケンスファイルを更新することが可能となる。

【００３４】

また、請求項１５から１９の何れか１項に記載の前記プログラムは、例えば請求項２０記載のように、前記属性情報がシーケンス言語で記述されており、前記シーケンスファイル更新処理が前記属性情報を前記シーケンスファイルに付加することで、該シーケンスファイルを更新するように構成することもできう。これにより、容易にシーケンスファイルを更新することができる。

【００３５】

また、本発明は、請求項２１記載のように、請求項１５から２０の何れか１項に記載の前記プログラムを記録媒体に記録した。これにより、上記の効果を奏するプログラムを広く頒布することができる。

【００３６】

【発明の実施の形態】

以下、本発明の好適な一実施形態について図面を用いて詳細に説明する。図１に本実施形

態による情報処理装置１の概略構成を示す。

【００３７】

情報処理装置１は、Ｅｔｈｅｒｎｅｔ（登録商標）やＩＥＥＥ１２８４やＵＳＢ（Ｕｎｉｖｅｒｓａｌ　Ｓｅｒｉａｌ　Ｂｕｓ）やファクシミリ（以下、ＦＡＸともいう）モデム受信部やスキャナ読込み部等の物理的なインタフェースを利用して外部からデータを入力するデータ入力コンポーネント１１０と、入力されたデータを処理する内部処理コンポーネント１２０と、処理したデータを上記の物理的な外部インタフェースや内部出力機構（例えば画像処理装置とした場合、プリンタエンジン等の出力エンジンやＦＡＸモデム送信部等）へ出力するデータ出力コンポーネント１３０と、これらを統括的に制御する制御部１００とを有して構成される。

【００３８】

データ入力コンポーネント１１０とデータ出力コンポーネント１３０とは、それぞれ与えられた処理を独立して動作する機能実行単位（コンポーネントともいう）を１つ以上有して構成される。以下の説明において、１つ以上の機能実行単位の集まりを機能実行単位群という。

【００３９】

内部処理コンポーネント１２０内部には、入力したデータを処理するための様々な機能（例えば画像処理装置とした場合、パース処理，ＰＤＬ処理，色変換処理，圧縮・伸張処理等）が、分担毎に機能実行単位として独立に設けられている。尚、図１では、ＰｏｓｔＳｃｒｉｐｔ（登録商標）等に代表されるＰＤＬ（Ｐａｇｅ　Ｄｅｓｃｒｉｐｔｉｏｎ　Ｌａｎｇｕａｇｅ）に対応した、ネットワークプリンタとして情報処理装置１を構成した場合を例に挙げている。

【００４０】

このような構成において、制御部１００は、所定の記憶手段（メモリ等）に格納してあるシーケンスファイル１４１に従って各コンポーネント（１１０，１２０，１３０）を制御する。即ち、シーケンスファイル１４１（但し、更新シーケンスファイル１４２も同様）には、情報処理装置１を構成する各機能実行単位を呼び出し、適切な処理及び例外的な処理を遂行するための状態遷移手順や各種パラメータが記述されている。これは例えばテキスト形式で作成されている。従って、制御部１００は、シーケンスファイル１４１（更新シーケンスファイル１４２）を参照することで、それぞれが独立して動作しているコンポーネント（１１０，１２０，１３０）とを統括的に制御し、データに対する所定の処理（例えば印刷ジョブ等）を実行・遂行する。尚、画像処理装置とした場合に入力される画像データは、例えばＰＤＬの規約に従って記述されているものとする。

【００４１】

ここで図２に、シーケンスファイル１４１（更新前）の記述内容の一例を示す。シーケーンスファイル１４１には、各種パラメータを定義する領域（パラメータ定義領域）と、情報処理装置１に搭載された各機能実行単位における各機能を呼び出して適切な処理および例外処理を遂行するための状態遷移手順が記述された状態遷移手順記述領域とが含まれる。尚、状態遷移手順としては、例えば情報処理装置１の電源投入時における初期化処理手順や、所定の処理の実行が要求された際の処理手順（但し、本説明では例として印刷処理手順を挙げる）等が存在する。

【００４２】

また、図１において、各コンポーネント（１１０，１２０，１３０）を構成する機能実行単位は属性情報を格納するための属性情報格納部（１１１，１２１，１３１）を有している。属性情報には、機能実行単位の名称（機能実行単位（コンポーネント）名），機能実行単位が所属するカテゴリ属性，優先度等を含み、更にその機能実行単位が持つ各機能の属性についての情報も含まれている。図３及び図４に、属性情報格納部１１１，１２１，１３１の構成を示す。尚、図３に示すテーブルには、ある機能実行単位に関する属性情報が格納されており、図４に示すテーブルには、この機能実行単位が持つ１つ以上の機能に関する詳細な情報が含まれている。即ち、本実施形態では、属性情報が機能実行単位に関

する情報（これを上位属性情報という）と、機能実行単位に含まれる機能毎の詳細な情報（これを下位属性情報という）とを含んで構成されている。これらの属性情報（上位属性情報及び下位属性情報）は、図示しない所定のメモリ上に格納されており、コンポーネント構成管理部１０１の要求により属性情報読込み部１０２が読み取れるように構成されている。図３及び図４を用いた説明では、“ｘｘＣｏｍｐ”というコンポーネント名の機能実行単位を例に挙げる。

【００４３】

図３に示すように、上位属性情報には、コンポーネント名と，カテゴリ属性と，優先度情報（Ｐｒｉｏｒｉｔｙ）と，下位属性情報を参照する際に使用するポインタ（下位属性情報ポインタ）との項目が格納されている。カテゴリ属性とは、機能実行単位が担当する機能の分類である。図１に示す例では、データ入力（データ入力コンポーネント１１０）と内部処理（内部処理コンポーネント１２０）とデータ出力（データ出力コンポーネント１３０）との３つの機能が存在する。以下の説明では、データ入力を“ＩＮ”で表し、内部処理を“Ｐｒｏｃ”で表し、データ出力を“ＯＵＴ”で表す。Ｐｒｉｏｒｉｔｙとは、各機能実行単位の優先順位を示す。これには、例えば高い（“Ｈｉｇｈ”）と低い（“Ｌｏｗ”）との２段階を設けることができるが、例えば３段階など種々変形しても良い。以下の説明では、上述のようにコンポーネント名が“ｘｘＣｏｍｐ”であり、カテゴリ属性が“Ｐｒｏｃ”であり、Ｐｒｉｏｒｉｔｙが“Ｌｏｗ”である機能実行単位について例を挙げる。尚、下位属性情報ポインタとは、所謂、アドレス情報である。即ち、制御部１００は、この下位属性情報ポインタを参照して対応する下位属性情報が格納されているメモリ領域を特定する。

【００４４】

また、図４に示すように、機能実行単位“ｘｘＣｏｍｐ”に含まれる機能についての詳細な属性情報を格納する下位属性情報には、個々の機能の名称（機能名）と，カテゴリ属性と，前後の依存関係（ＰｒｅＦｕｎｃ，ＰｏｓｔＦｕｎｃ）との項目が格納されている。図４に示す例では、機能実行単位“ｘｘＣｏｍｐ”の中に、機能として初期化機能（“Ｉｎｉｔｉａｌｉｚｅ（）”）と，前処理機能（“ＰｒｅＣｏｎｖ（）”）と，後処理機能（“ＰｏｓｔＣｏｎｖ（）”）とが含まれている。このような各機能実行単位に含まれる機能は、データ入力“ＩＮ”と内部処理“Ｐｒｏｃ”とデータ出力“ＯＵＴ”とでそれぞれ固定されているとよい。即ち、例えば内部処理“Ｐｒｏｃ”のカテゴリ属性に属するコンポーネントは、基本的に上述の３つの機能（“Ｉｎｉｔｉａｌｉｚｅ（）”，“ＰｒｅＣｏｎｖ（）”，“ＰｏｓｔＣｏｎｖ（）”）を含むように構成するとよい。また、カテゴリ属性は、図３に示すものと異なり、機能が例えば初期化処理手順や印刷処理手順等において何れの手順に属するかを示すものである。また、依存関係において、ＰｒｅＦｕｎｃは前の依存関係を示している。即ち、例えば図４で説明すると、機能“ＰｒｅＣｏｎｖ（）”を実行する前に機能“ＰＤＬ．ｘｘ（）”（尚、これは他の機能実行単位である“ＰＤＬ”に含まれる機能である）が実行されている必要があることを示している。ＰｏｓｔＦｕｎｃは後の依存関係を示している。即ち、図４で説明すると、機能“ＰｒｅＣｏｎｖ（）”を実行した後には機能“ＰｏｓｔＣｏｎｖ（）”を実行する必要があることを示している。尚、例えば図４において、機能“Ｉｎｉｔｉａｌｉｚｅ（）”には前後の依存関係が必要ないことが空欄で示されている。また、同様に図４では、機能“ＰｒｅＣｏｎｖ（）”及び機能“ＰｏｓｔＣｏｎｖ（）”をカテゴリ属性で分類する必要が無いことが空欄で示されている。

【００４５】

図１に戻って制御部１００の詳細な構成を説明する。制御部１００は、コンポーネント構成管理部１０１と，実行ＡＰＩ変換部１０２と，コンポーネント構成変更通知部１０３と，属性情報読込み部１０４と，シーケンス更新部１０５と，シーケンス再読込み通知部１０６と，シーケンス読込み部１０７と，シーケンス解釈部１０８とを有して構成される。

【００４６】

コンポーネント構成管理部１０１は、各コンポーネント（１１０，１２０，１３０）にお

ける機能実行単位の構成を管理している。これは、例えば図５（更新前）及び図６（更新後）に示すようなテーブル（機能管理テーブル１０１ａ）を用いて行われる。機能管理テーブル１０１ａは、例えば機能名とカテゴリ属性と属性情報ポインタとの項目が格納されている。機能名とカテゴリ属性とは上述の内容と同じである。属性情報ポインタとは、各コンポーネント（１１０，１２０，１３０）における属性情報格納部（１１１，１２１，１３１）の格納位置を示す、所謂、アドレス情報である。即ち、コンポーネント構成管理部１０１は、この属性情報ポインタを参照して対応する属性情報格納部１１１を特定する。

【００４７】

コンポーネント構成変更通知部１０３は、各コンポーネント（１１０，１２０，１３０）に新たな機能実行単位が追加されたか否かを管理し、追加された場合、追加されたことをコンポーネント構成管理部１０１に通知する。尚、機能実行単位の追加方法としては、例えばローカルエリアネットワーク（ＬＡＮ）やインターネットやその他の通信回線を介してダウンロード、若しくは持ち運び可能な記録媒体を介して入力し、これを組み込む方法を用いることができる。

【００４８】

コンポーネント構成変更通知部１０３から新たなコンポーネントの追加が通知された場合、コンポーネント構成管理部１０１は現在の各コンポーネント（１１０，１２０，１３０）の属性情報を属性情報読込み部１０４を用いて読み取ることで、現在の各コンポーネントの機能構成を取得する。次に、取得した機能構成と機能管理テーブル１０１ａに格納されている機能一覧とを比較することで、新たに追加された機能実行単位を特定する。本説明では、機能実行単位“ｘｘＣｏｍｐ”が追加された場合を例に挙げている。

【００４９】

コンポーネント構成管理部１０１は、上記で特定した機能実行単位の名称及びカテゴリ属性，並びにこれらを含む属性情報の参照先（属性情報ポインタ）とを、属性情報読込み部１０４から入力された属性情報に基づいて特定し、これらを機能管理テーブル１０１ａに追加登録する（図６参照）。尚、“ｘｘＣｏｍｐ”は、内部処理コンポーネント１２０のカテゴリ属性に分類される機能実行単位である。従って、コンポーネント構成管理部１０１は、属性情報（図３及び図４参照）からコンポーネント名“ｘｘＣｏｍｐ”とカテゴリ属性“Ｐｒｏｃ”とを特定し、更に同時に属性情報読込み部１０４から入力された各属性情報の参照先に基づいて属性情報ポインタ“０ｘｘｆｆ００ｘｘｘｘｘｘ”を特定し、これらを機能管理テーブル１０１ａに追加登録する。

【００５０】

また、コンポーネント構成管理部１０１は、上記で特定した属性情報をシーケンス更新部１０５に入力する。シーケンス更新部１０５は、属性情報が入力されると、シーケンスファイル１４１（図２参照）を読み出し、これをシーケンス言語の規約に基づいて、入力された属性情報に含まれる各機能毎の前後の依存関係（機能実行単位毎の各機能の読み出して順ともいう）に従うように更新する。これを具体的に説明する。但し、この説明において、更新前のシーケンスファイル１４１の記述内容を図２に示すものとし、新たにダウンロードした機能実行単位“ｘｘＣｏｍｐ”の属性情報を図３及び図４に示すものとする。

【００５１】

機能実行単位“ｘｘＣｏｍｐ”の属性情報がコンポーネント構成管理部１０１から入力されると、シーケンス更新部１０５は、シーケンスファイル１４１を読み出す。このシーケンスファイル１４１は図示しない所定のメモリに格納されており、シーケンス更新部１０５及びシーケンス読込み部１０７が読み込めるように構成されている。

【００５２】

次に、シーケンス更新部１０５は、入力された属性情報におけるカテゴリ属性“Ｐｒｏｃ”から機能実行単位“ｘｘＣｏｍｐ”に含まれる機能には“Ｉｎｉｔｉａｌｉｚｅ（）”と“ＰｒｅＣｏｎｖ（）”と“ＰｏｓｔＣｏｎｖ（）”とがあることを特定する。尚、各機能を読み出すための記述には、機能名“Ｉｎｉｔｉａｌｉｚｅ（）”と“ＰｒｅＣｏｎ

ｖ（）”と“ＰｏｓｔＣｏｎｖ（）”）を用いることができる。更に、シーケンス更新部１０５は、各機能を呼び出すための前後の依存関係（条件）を特定する。

【００５３】

このように各機能を呼び出すための記述及び各機能の前後の依存関係を特定すると、シーケンス更新部１０５は、まず、機能“Ｉｎｉｔｉａｌｉｚｅ（）”に関する記述を更新する。この更新は、“Ｉｎｉｔｉａｌｉｚｅ（）”がカテゴリ属性“ＩＮＩＴ”に属するため、図２の２０行目から２６行めまでのコンポーネントの初期化処理手順が記述された領域を図７に示すように更新することで行われる。この際、“Ｉｎｉｔｉａｌｉｚｅ（）”は前後の依存関係が存在しないため、この機能を呼び出すための記述“ｘｘＣｏｍｐ．Ｉｎｉｔｉａｌｉｚｅ（）”は、この領域のどこに追加してもよい。図７に示す更新後のシーケンスファイル（更新シーケンスファイル１４２）では、２６行目の後、即ち、２７行目に記述を追加している。尚、シーケンス言語に規約に従い、更新前に初期化処理手順の終了位置であった２６行目における出力条件を記述する領域には、“ｘｘＣｏｍｐ　Ｉｎｉｔｉａｌｉｚｅ（）”の記述が追加され、同じく２６行目の次のステート名を記述する領域には“Ｉ７”の記述が追加されている。これにより、２６行目の終了後に２７行目の初期化処理手順を行うように記述される。

【００５４】

次に、シーケンス更新部１０５は、機能“ＰｒｅＣｏｎｖ（）”及び“ＰｏｓｔＣｏｎｖ（）”に関する記述を更新する。この更新は、“ＰｒｅＣｏｎｖ（）”及び“ＰｏｓｔＣｏｎｖ（）”のカテゴリ属性が定義されていないため、前後の依存関係に基づいて行われる。“ＰｒｅＣｏｎｖ（）”は図４に示すように、ＰｒｅＦｕｎｃとして“ＰＤＬ．ｘｘ（）”が先に呼び出されている必要があり、且つ、ＰｏｓｔＦｕｎｃとして“ＰｏｓｔＣｏｎｖ（）”が呼び出される必要がある。但し、“ＰｏｓｔＣｏｎｖ（）”は未だ追加されていない機能である。従って、この機能を呼び出すための記述“ｘｘＣｏｍｐ．ＰｒｅＦｕｎｃ（）”を記載する位置は“ＰＤＬ．ｘｘ（）”の記述位置に基づいて、３５行目の後（但し、図７では３６行目の後、即ち３７行目）に追加される。また、“ＰｏｓｔＦｕｎｃ（）”は同じく図２に示すように、ＰｒｅＦｕｎｃとして“ＰｒｅＣｏｎｖ（）”が読み出されている必要があり、且つ、ＰｏｓｔＦｕｎｃとして“Ｃｏｍｐｒｅｓｓ．ｘｘ（）”が呼び出される必要があるため、上記で追加した“ＰｒｅＣｏｎｖ（）”の後（図７では３８行目）に追加される。尚、“ｘｘＣｏｍｐ．Ｉｎｉｔｉａｌｉｚｅ（）”の追加時と同様に、シーケンス言語に規約に従い、追加行及びこれらの前後の行における出力条件と入力条件との記述が追加／変更される。

【００５５】

以上のような手順により、シーケンス更新部１０５は、新たに追加された機能実行単位（コンポーネント）の属性情報から新たなコンポーネントが備えている機能とその機能の呼び出し条件とを判断してシーケンスファイル１４１を更新する。これにより作成された更新シーケンスファイル１４２は、元のシーケンスファイル１４１に対して上書きして保存される。この際、元のシーケンスファイル１４１の内容は、別のファイル名等で保存しておくとよい。

【００５６】

このようにシーケンスファイル１４１を更新すると、シーケンス更新部１０５は、画像処理装置１に再起動をかけるために、シーケンス再読込み通知部１０６を呼び出す。シーケンス再読込み通知部１０６は、シーケンス読込み部１０７に対して、更新シーケンスファイル１４２を再び最初から読み込むように指示する。この際、上記のように更新シーケンスファイル１４２を元のシーケンスファイル１４１と同一ファイル名で保存した場合、シーケンス読込み部１０７は、元のファイル名を読み込むことにより、制御の再起動をかけることができる。

【００５７】

以上のような手順を踏まえることで、ネットワーク等の外部からダウンロードした新たなコンポーネントの属性情報からコンポーネントの内容を解析し、新たなコンポーネントの

機能を呼ぶためのシーケンスを更新し、シーケンスファイルを再読込みすることにより情報処理装置に新たな機能を追加して再構成することが可能となる。

【００５８】

尚、これらの構成及び手順は、例えばプログラムを実行することでも実現し得る。即ち、上記のような制御部１００の各構成を実現するプログラムをＲＯＭ（Ｒｅａｄ Ｏｎｌｙ Ｍｅｍｏｒｙ）又はハードディスク等の記憶装置に記憶しておき、これを読み出して実行することで、本実施形態を実現するように構成することが可能である。また、このプログラムを例えば記録媒体にき録して頒布することで、広く提供することが可能となる。

【００５９】

次に、本実施形態による情報処理装置１を用いて画像処理装置１０を構成した場合の例を以下に図面を用いて詳細に説明する。図８は、本実施形態による画像処理装置１０の構成を示すブロック図である。尚、図８では、制御部１００の構成を簡略化して示す。

【００６０】

画像処理装置１０は、データ入力コンポーネント１１０として、例えばＥｔｈｅｒｎｅｔ（登録商標）／ＩＥＥＥ１２８４／ＵＳＢ／ＦＡＸモデム受信部／スキャナ読込み部等の物理的なインタフェースを利用して外部から画像データを入力するネットワーク入力コンポーネント１１２を有し、データ出力コンポーネント１３０としては、プリンタ出力コンポーネント１３２を有する、所謂、ネットワークプリンタとして構成されている。また、内部処理を実現する内部処理コンポーネント１２０としては、ＵＩ（Ｕｓｅｒ Ｉｎｔｅｒｆａｃｅ）処理コンポーネント１２２と，パース処理コンポーネント１２３と，ＰＤＬ処理コンポーネント１２４と，色変換コンポーネント１２５と，圧縮・伸張コンポーネント１２６とが設けられており、それぞれが独立して動作する。

【００６１】

これらの機能実行単位は、共通化されたＡＰＩ（Ａｐｐｌｉｃａｔｉｏｎ Ｐｒｏｇｒａｍ Ｉｎｔｅｒｆａｃｅ）を用いて起動／停止を含む動作が制御されると共に、共通化された情報フォーマットを用いて機能実行単位間で情報の授受が行われる。

【００６２】

図８に示す構成を模式的に表した図を図９に示す。図９に示すように、各機能実行単位（１１２，１２２，１２３，１２４，１２５，１２６，１３２）は、制御部１００にて個々に独立に制御されている。このような構成において、図１０に示すように、例えば内部処理コンポーネント１２０である機能実行単位（倍率変換処理コンポーネント１２７）が新たに追加された場合、図１１に示すように、制御部１００におけるコンポーネント構成変更通知部１０３がこのことを検知し、これをコンポーネント構成管理部１０１（図１参照）へ通知する。コンポーネント構成管理部１０１は、上述のような処理をして新たに追加されたコンポーネントの属性情報を取得した後に、シーケンス変更部１０５を呼び出して、シーケンスファイル１４１を書き換え、更新シーケンスファイル１４２を作成する。これにより、図１２に示すように、新たな機能実行単位倍率変換処理コンポーネント１２７が追加された画像処理装置１０が構成される。

【００６３】

また、上記の構成において、ＵＩ処理コンポーネント１２２は、タッチパネルやディスプレイ等で構成された操作画面に表示する図形やボタンや文字等の表示内容を描画するための機能であり、これもシーケンスファイル１４１（更新シーケンスファイル１４２も含む）に従って制御部が動作制御するように構成することよい。これにより、新たな機能を追加した際に、これを使用するための表示画面も、ネットワークを介して遠隔に更新することが可能となる。

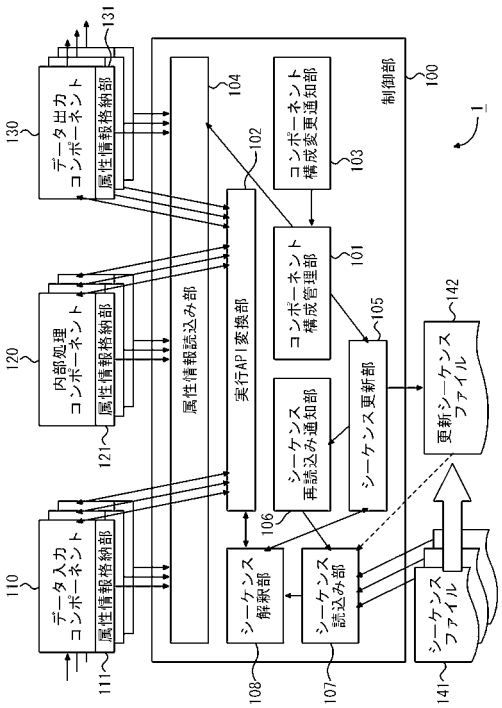
【００６４】

また、上記した説明では、各機能実行単位に対応づけられた属性情報がテーブル形式で記述されていたが、これを例えばシーケンス言語で記述された形式とすることも可能である。即ち、属性情報が例えばサブルーチンとしてシーケンス言語で記述されており、機能実行単位の追加時にこれをシーケンスファイル１４１に追加（付加）だけで、シーケンスフ

ァイルを書き換えることが可能なようにも構成することが可能である。
【００６５】
以上、説明した実施形態は本発明の好適な一実施形態にすぎず、本発明はその趣旨を逸脱しない限り種々変形して実施可能である。
【００６６】
【発明の効果】
以上説明したように、本発明によれば、ネットワークを介して遠隔且つ容易に機能の更新をすることが可能な情報処理装置及び画像処理装置、並びに機能追加方法、そのプログラム及びそのプログラムを記録した記録媒体が実現できる。
【００６７】
具体的には、全体の制御をテキストファイルで記述することにより、他のプラットフォームでも容易に編集することが出来るため、動作手順の変更が効率良く行える。また、機能実行単位の動的な構成変更にもシーケンスファイルを更新することにより柔軟なシステムを組むことが出来る。
【００６８】
また、シーケンスファイルの再読込みによりシステムを再起動することで、装置全体の電源を再投入するような再起動方法に比べて短時間で再起動が可能になる。更に、ＵＩなどの描画をテキスト形式の制御ファイルに記述することにより、ＵＩ表示の変更・修正等が容易な画像処理装置を実現することが可能になる。
【図面の簡単な説明】
【図１】本発明の一実施形態による情報処理装置１の概略構成を示すブロック図である。
【図２】図１におけるシーケンスファイル１４１の記述内容を示す図である。
【図３】本発明の一実施形態の属性情報における上位属性情報の内容を示すテーブルである。
【図４】本発明の一実施形態の属性情報における下位属性情報の内容を示すテーブルである。
【図５】図１に示すコンポーネント構成管理部１０１が管理する機能管理テーブル１０１ａの内容を示すテーブルである（更新前）。
【図６】図１に示すコンポーネント構成管理部１０１が管理する機能管理テーブル１０１ａの内容を示すテーブルである（更新後）。
【図７】図１における更新シーケンスファイル１４２の記述内容を示す図である。
【図８】本発明の一実施形態による情報処理装置１を用いて構成した画像処理装置１０の概略構成を示すブロック図である。
【図９】図８に示す画像処理装置１０の構成を模式的に示す図である（機能実行単位追加前）。
【図１０】図８に示す画像処理装置１０の構成を模式的に示す図である（機能実行単位追加時）。
【図１１】図８に示す画像処理装置１０の構成を模式的に示す図である（機能実行単位追加後）。
【図１２】倍率変換処理コンポーネント１２７を追加した後の画像形成装置１０の概略構成を示すブロック図である。
【図１３】従来の情報処理装置の第１の例を示す構成図である。
【図１４】従来の情報処理装置の第２の例を示す構成図である。
【符号の説明】
１　情報処理装置　　１０　画像処理装置
１００　制御部　　１０１　コンポーネント構成管理部
１０１ａ　機能管理テーブル　　１０２　実行ＡＰＩ変換部
１０３　コンポーネント構成変更通知部
１０４　属性情報読込み部　　１０５　シーケンス更新部
１０７　シーケンス読込み部　　１０８　シーケンス解釈部

１１０　データ入力コンポーネント
１１１、１２１、１３１　属性情報格納部
１１２　ネット入力コンポーネント　　１２０　内部処理コンポーネント
１２２　ＵＩ処理コンポーネント　　１２３　パース処理コンポーネント
１２４　ＰＤＬ処理コンポーネント　　１２５　色変換処理コンポーネント
１２６　圧縮／伸張処理コンポーネント
１２７　倍率変換処理コンポーネント
１３０　データ出力コンポーネント　　１３２　プリンタ出力コンポーネント
１４１　シーケンスファイル　　１４２　更新シーケンスファイル

【図１】



【図２】

シーケンスファイル141

```
 1 /*
 2  * root.seq
 3  */
 4
 5 // for System control.
 6 _SYS_SEQUENCER_LOGLEVEL = 4:
 7 _SYS_COMPONENT_LOGLEVEL = 4:
 8
 9 // for Lpd component.
10 STREAM_BLOCK_SIZE       = 512:
11 STREAM_BLOCK_DEPTH      = 3:
12 SERV_TCP%PORT           = 515:
13
14 // for pinter.
15 RESOLUTION              = 600:
16 PaperSize               = "A4":
17
18 // State  : Input                    : Output                 : Next
19 // Initialize sequence.
20 INIT     :                          : Lpd.Initialize()       : I1:
21 I1       : Lpd.Initialize[==0]      : Spool.Initialize()     : I2:
22 I2       : Spool.Initialize[==0]    : Parce.Initialize()     : I3:
23 I3       : Parce.Initialize[==0]    : PDL.Initialize()       : I4:
24 I4       : PDL.Initialize[==0]      : Compress.Initialize()  : I5:
25 I5       : Compress.Initialize[==0] : Print.Initialize()     : I6:
26 I6       : Print.Initialize[==0]    :                        : C1:
27
28 // Lpd Connection enable.
29 C1       :                          : Lpd.connect()          : C2:
30
31 // Watting for job.
32 C2       : Lpd.connect[==0]         : A.read()               : C3:
33 C3       : A.read[--0]              : Spool.XX()             : C4:
34 C4       : Spool.xx[==0]            : Parce.xx()             : C5:
35 C5       : Parce.xx[==0]            : PDL.xx()               : C6:
36 C6       : PDL.xx[--0]              : Compress.xx()          : C7:
37 C7       : Compress.xx[==0]         : Print.xx()             : C8:
38 C8       : Print.xx[==0]            :                        : C1:
```

パラメータ定義領域
初期化処理手順
印刷処理手順
ステート名　入力条件　出力条件　次のステート名

【図３】

上位属性情報

コンポーネント名	カテゴリ属性	Priority	下位属性情報ポインタ
xxComp	Proc	Low	0xff00xxxx

【図４】

下位属性情報

機能名	カテゴリ属性	PreFunc	PostFunc
Initialize()	Init		
PreConv()		PDL.xx()	PostConv()
postConv()		PreConv()	Compress.xx()

【図５】

機能管理テーブル 101a

機能名	カテゴリ属性	属性情報ポインタ
Lpd	IN	0xffxxxxxx
Spool	Proc	0xffxxxxxx
Parce	Proc	0xffxxxxxx
PDL	Proc	0xffxxxxxx
Compress	Proc	0xffxxxxxx
Print	OUT	0xffxxxxxx

【図６】

機能管理テーブル 101a

機能名	カテゴリ属性	属性情報ポインタ
Lpd	IN	0xffxxxxxx
Spool	Proc	0xffxxxxxx
Parce	Proc	0xffxxxxxx
PDL	Proc	0xffxxxxxx
Compress	Proc	0xffxxxxxx
Print	OUT	0xffxxxxxx
xxComp	Proc	0xffxxxxxx

【図７】

更新シーケンスファイル142

```
 1 /*
 2  * root.seq
 3  */
 4
 5 // for System control.
 6 _SYS_SEQUENCER_LOGLEVEL = 4;
 7  SYS_COMPONENT_LOGLEVEL = 4;
 8
 9 // for Lpd component.
10 STREAM_BLOCK_SIZE       = 512;
11 STREAM_BLOCK_DEPTH      = 3;
12 SERV_TCPPORT            = 515;
13
14 // for pinter.
15 RESOLUTION              = 600;
16 PaperSize               = "A4";
17
18 // State  : Input                    : Output                    : Next
19 // Initialize sequence.
20 INIT     :                          : Lpd.Initialize()          : I1;
21 I1       : Lpd.Initialize[==0]      : Spool.Initialize()        : I2;
22 I2       : Spool.Initialize[==0]    : Parce.Initialize()        : I3;
23 I3       : Parce.Initialize[==0]    : PDL.Initialize()          : I4;
24 I4       : PDL.Initialize[==0]      : Compress.Initialize()     : I5;
25 I5       : Compress.Initialize[==0] : Print.Initialize()        : I6;
26 I6       : Print.Initialize[==0]    : xxComp.Initialize()       : I7;
27 I7       : xxComp.Initialize[==0]   :                           : C1;
28
29 // Lpd Connection enable.
30 C1       :                          : Lpd.connect()             : C2;
31
32 // Watting for job.
33 C2       : Lpd.connect[==0]         : A.read()                  : C3;
34 C3       : A.read[==0]              : Spool.XX()                : C4;
35 C4       : Spool.xx[--0]            : Parce.xx()                : C5;
36 C5       : Parce.xx[==0]            : PDL.xx()                  : C6;
37 C6       : PDL.xx[==0]              : xxComp.PostConv()         : C7;
38 C7       : xxComp.PreConv[--0]      : xxCcomp.PreConv()         : C8;
39 C8       : xxComp.postConv[==0]     : Compress.xx()             : C9;
40 C9       : Compress.xx[--0]         : Print.xx()                : C10;
41 C10      : Print.xx[==0]            :                           : C1;
```

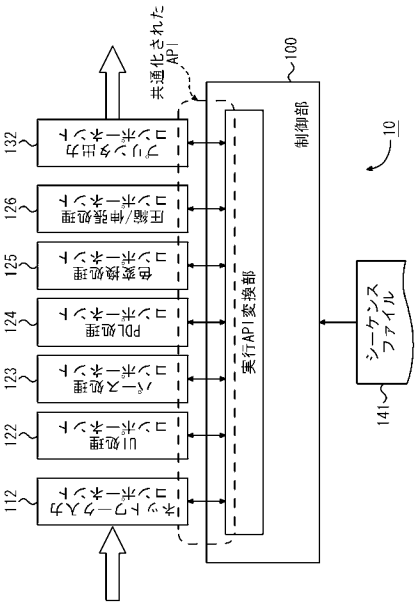
パラメータ定義領域

初期化処理手順

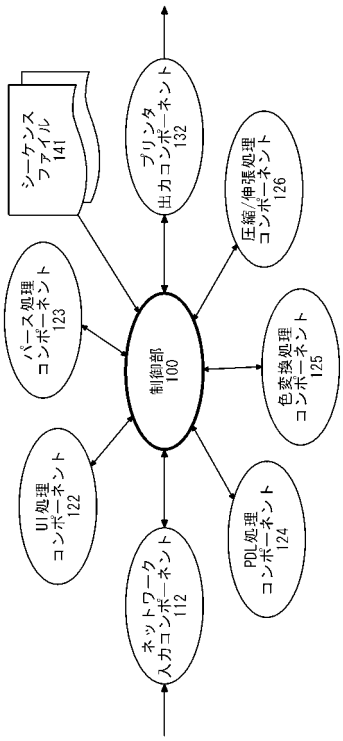
印刷処理手順

ステート名　入力条件　出力条件　次のステート名

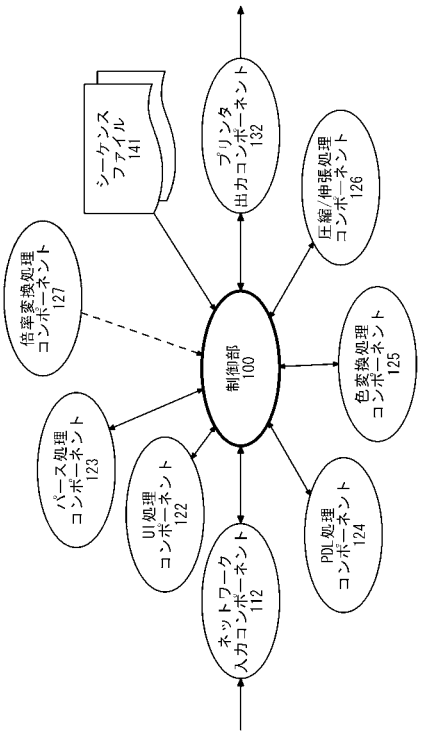
【図８】



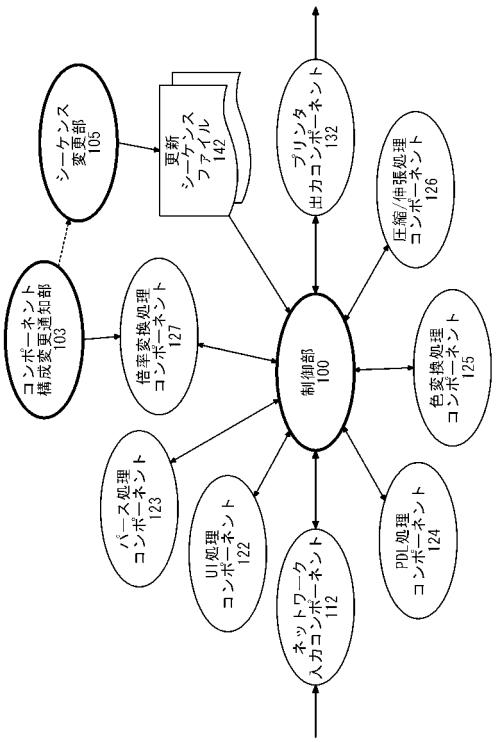
【図９】



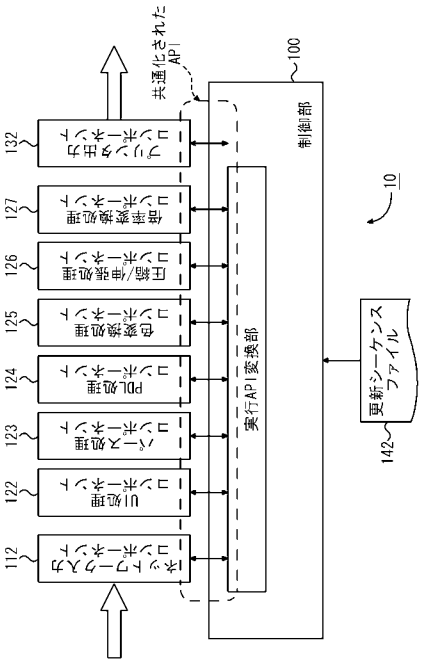
【図１０】



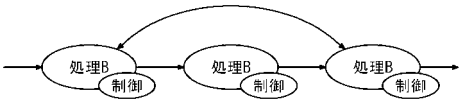
【図１１】



【図１２】



【図１３】



【図１４】