(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第6928552号
(P6928552)

(45)発行日　令和3年9月1日(2021.9.1)　(24)登録日　令和3年8月11日(2021.8.11)

(51)Int.Cl.　F I
G06F 8/70 (2018.01)　G06F 8/70
G06F 11/36 (2006.01)　G06F 11/36 １８０

請求項の数 16 (全 70 頁)

(21)出願番号　特願2017-520918(P2017-520918)
(86)(22)出願日　平成27年11月5日(2015.11.5)
(65)公表番号　特表2018-501538(P2018-501538A)
(43)公表日　平成30年1月18日(2018.1.18)
(86)国際出願番号　PCT/US2015/059266
(87)国際公開番号　WO2016/073735
(87)国際公開日　平成28年5月12日(2016.5.12)
審査請求日　平成30年8月6日(2018.8.6)
(31)優先権主張番号　62/075,558
(32)優先日　平成26年11月5日(2014.11.5)
(33)優先権主張国・地域又は機関
米国(US)
(31)優先権主張番号　14/738,232
(32)優先日　平成27年6月12日(2015.6.12)
(33)優先権主張国・地域又は機関
米国(US)

(73)特許権者　509123208
アビニシオ　テクノロジー　エルエルシー
アメリカ合衆国　０２４２１　マサチューセッツ州　レキシントン　スプリング　ストリート　２０１
(74)代理人　100079108
弁理士　稲葉　良幸
(74)代理人　100109346
弁理士　大貫　敏史
(74)代理人　100117189
弁理士　江口　昭彦
(74)代理人　100134120
弁理士　内藤　和彦

最終頁に続く

(54)【発明の名称】影響分析

(57)【特許請求の範囲】
【請求項１】
コンピュータ実施方法であって、
少なくとも第１の論理データセットと第２の論理データセットについての論理データセット情報を受信することであって、前記論理データセット情報は、各論理データセットについて、前記論理データセット内の少なくとも１つのフィールドの識別子と、前記フィールドについてのフォーマット情報とを識別し、少なくとも前記第１の論理データセット又は前記第２の論理データセットは、データ要素を記憶する、受信することと、
変換についての情報を受信することであって、前記情報は、前記第１の論理データセットを、前記変換がデータを受信する論理データセットとして、かつ、前記第２の論理データセットを、変換データが提供される論理データセットとして識別し、前記変換は、前記第１の論理データセットからのデータに適用される１つ又は複数のルールを含む、受信することと、
前記第１の論理データセットの少なくとも１つのフィールド、前記第２の論理データセットの少なくとも１つのフィールド、又は、前記変換、への１つ又は複数の提案される変更を受信することと、
前記変換についての前記情報と、前記１つ又は複数のルールと、前記第１の論理データセット及び前記第２の論理データセットについての前記論理データセット情報とに基づいて、前記１つ又は複数の提案される変更を分析することと、
前記分析に基づいて、前記１つ又は複数の提案される変更によって前記データ要素が影

響される量を示す、１つ又は複数のメトリックを計算することであって、少なくとも１つの前記１つ又は複数のメトリックは、前記１つ又は複数の提案される変更によって影響を受ける前記データ要素が前記１つ又は複数のルール内で参照される回数を指定する、計算することと、
前記１つ又は複数のメトリックについての情報を記憶することと
を含む、コンピュータ実施方法。
【請求項２】
前記計算されるメトリックは、直接影響の尺度を提供する、請求項１に記載の方法。
【請求項３】
前記計算されるメトリックは、間接影響の尺度を提供する、請求項１に記載の方法。
【請求項４】
前記提案される変更は、データセット内のフィールドのフォーマットの変更又は変換の変更からなる群の１つである、請求項１に記載の方法。
【請求項５】
前記メトリックに基づいてコストを前記提案される変更に関連付けることを更に含む、請求項１に記載の方法。
【請求項６】
１つ又は複数のコンピュータ及び命令を記憶する１つ又は複数の記憶デバイスを含むシステムであって、前記命令は、前記１つ又は複数のコンピュータにより実行されると、前記１つ又は複数のコンピュータに、
少なくとも第１の論理データセットと第２の論理データセットについての論理データセット情報を受信することであって、前記論理データセット情報は、各論理データセットについて、前記論理データセット内の少なくとも１つのフィールドの識別子と、前記フィールドについてのフォーマット情報とを識別し、少なくとも前記第１の論理データセット又は前記第２の論理データセットは、データ要素を記憶する、受信することと、
変換についての情報を受信することであって、前記情報は、前記第１の論理データセットを、前記変換がデータを受信する論理データセットとして、かつ、前記第２の論理データセットを、変換データが提供される論理データセットとして識別し、前記変換は、前記第１の論理データセットからのデータに適用される１つ又は複数のルールを含む、受信することと、
前記第１の論理データセットの少なくとも１つのフィールド、前記第２の論理データセットの少なくとも１つのフィールド、又は、前記変換、への１つ又は複数の提案される変更を受信することと、
前記変換についての前記情報と、前記１つ又は複数のルールと、前記第１の論理データセット及び前記第２の論理データセットについての前記論理データセット情報とに基づいて、前記１つ又は複数の提案される変更を分析することと、
前記分析に基づいて、前記１つ又は複数の提案される変更によって前記データ要素が影響される量を示す、１つ又は複数のメトリックを計算することであって、少なくとも１つの前記１つ又は複数のメトリックは、前記１つ又は複数の提案される変更によって影響を受ける前記データ要素が前記１つ又は複数のルール内で参照される回数を指定する、計算することと、
前記１つ又は複数のメトリックについての情報を記憶することと
を含む動作を実行させるように動作可能である、システム。
【請求項７】
前記計算されるメトリックは、直接影響の尺度を提供する、請求項６に記載のシステム。
【請求項８】
前記計算されるメトリックは、間接影響の尺度を提供する、請求項６に記載のシステム。
【請求項９】

前記提案される変更は、データセット内のフィールドのフォーマットの変更又は変換の変更からなる群の１つである、請求項６に記載のシステム。
【請求項１０】
前記メトリックに基づいてコストを前記提案される変更に関連付けることを更に含む、請求項６に記載のシステム。
【請求項１１】
少なくとも第１の論理データセットと第２の論理データセットについての論理データセット情報を受信する手段であって、前記論理データセット情報は、各論理データセットについて、前記論理データセット内の少なくとも１つのフィールドの識別子と、前記フィールドについてのフォーマット情報とを識別し、少なくとも前記第１の論理データセット又は前記第２の論理データセットは、データ要素を記憶する、手段と、
変換についての情報を受信する手段であって、前記情報は、前記第１の論理データセットを、前記変換がデータを受信する論理データセットとして、かつ、前記第２の論理データセットを、変換データが提供される論理データセットとして識別し、前記変換は、前記第１の論理データセットからのデータに適用される１つ又は複数のルールを含む、手段と、
前記第１の論理データセットの少なくとも１つのフィールド、前記第２の論理データセットの少なくとも１つのフィールド、又は、前記変換、への１つ又は複数の提案される変更を受信する手段と、
前記変換についての前記情報と、前記１つ又は複数のルールと、前記第１の論理データセット及び前記第２の論理データセットについての前記論理データセット情報とに基づいて、前記１つ又は複数の提案される変更を分析する手段と、
前記分析に基づいて、前記１つ又は複数の提案される変更によって前記データ要素が影響される量を示す、１つ又は複数のメトリックを計算する手段であって、少なくとも１つの前記１つ又は複数のメトリックは、前記１つ又は複数の提案される変更によって影響を受ける前記データ要素が前記１つ又は複数のルール内で参照される回数を指定する、計算する手段と、
前記１つ又は複数のメトリックについての情報を記憶する手段と
を含む、システム。
【請求項１２】
コンピュータプログラム命令が符号化されたコンピュータ記憶媒体であって、前記コンピュータプログラム命令は、１つ又は複数のコンピュータにより実行されると、前記１つ又は複数のコンピュータに、
少なくとも第１の論理データセットと第２の論理データセットについての論理データセット情報を受信することであって、前記論理データセット情報は、各論理データセットについて、前記論理データセット内の少なくとも１つのフィールドの識別子と、前記フィールドについてのフォーマット情報とを識別し、少なくとも前記第１の論理データセット又は前記第２の論理データセットは、データ要素を記憶する、受信することと、
変換についての情報を受信することであって、前記情報は、前記第１の論理データセットを、前記変換がデータを受信する論理データセットとして、かつ、前記第２の論理データセットを、変換データが提供される論理データセットとして識別し、前記変換は、前記第１の論理データセットからのデータに適用される１つ又は複数のルールを含む、受信することと、
前記第１の論理データセットの少なくとも１つのフィールド、前記第２の論理データセットの少なくとも１つのフィールド、又は、前記変換、への１つ又は複数の提案される変更を受信することと、
前記変換についての前記情報と、前記１つ又は複数のルールと、前記第１の論理データセット及び前記第２の論理データセットについての前記論理データセット情報とに基づいて、前記１つ又は複数の提案される変更を分析することと、
前記分析に基づいて、少なくとも前記第１の論理データセット、前記第２の論理データ

セット、又は、前記変換への前記１つ又は複数の提案される変更によって影響される前記データ要素の量を示す、１つ又は複数のメトリックを計算することであって、少なくとも１つの前記１つ又は複数のメトリックは、前記１つ又は複数の提案される変更によって影響を受ける前記データ要素が前記１つ又は複数のルール内で参照される回数を指定する、計算することと、

前記１つ又は複数のメトリックについての情報を記憶することと

を含む動作を実行させる、コンピュータ記憶媒体。

【請求項１３】

前記計算されるメトリックは、直接影響の尺度を提供する、請求項１２に記載の媒体。

【請求項１４】

前記計算されるメトリックは、間接影響の尺度を提供する、請求項１２に記載の媒体。

【請求項１５】

前記提案される変更は、データセット内のフィールドのフォーマットの変更又は変換の変更からなる群の１つである、請求項１２に記載の媒体。

【請求項１６】

前記メトリックに基づいてコストを前記提案される変更に関連付けることを更に含む、請求項１２に記載の媒体。

【発明の詳細な説明】

【技術分野】

【０００１】

背景

この説明はシステム分析に関する。

【背景技術】

【０００２】

大量のデータを処理するためにコンピュータが使用される。一般に、データは、少なくとも部分的にコンピュータプログラマにより記述されるコンピュータプログラムを使用して処理される。これらのデータ処理システムは複雑であり得る。

【発明の概要】

【発明が解決しようとする課題】

【０００３】

ビジネス及び技術での要件により、プログラムの変更が必要になり得る。変更を実施するには、変更を行う人員を割り振る必要がある。

【課題を解決するための手段】

【０００４】

概要

一般的な態様１において、方法は、少なくとも２つの論理データセットについての情報を受信する動作を含み、論理データセット情報は、各論理データセットについて、その論理データセット内の少なくとも１つのフィールドの識別子と、そのフィールドについてのフォーマット情報とを識別する。本方法は、変換についての情報を受信する動作を含み、その情報は、変換がデータを受信する第１の論理データセットと、変換データが提供される第２の論理データセットとを識別する。本方法は、論理データセットのフィールドの少なくとも１つへの１つ又は複数の提案される変更を受信する動作を含む。本方法は、変換についての情報と、第１の論理データセット及び第２の論理データセットについての情報とに基づいて、１つ又は複数の提案される変更を分析する動作を含む。本方法は、分析に基づいて、提案される変更の１つ又は複数のメトリックを計算する動作を含む。本方法は、１つ又は複数のメトリックについての情報を記憶する動作を含む。

【０００５】

この態様の他の実施形態は、本方法の動作を実行するようにそれぞれ構成される対応するコンピュータシステム、装置、及び１つ又は複数のコンピュータ記憶デバイスに記録されるコンピュータプログラムを含む。１つ又は複数のコンピュータのシステムは、動作に

当たり、システムに動作を実行させる、システムにインストールされたソフトウェア、ファームウェア、ハードウェア、又はそれらの組合せにより特定の動作を実行するように構成することができる。１つ又は複数のコンピュータプログラムは、データ処理装置により実行されると、装置に動作を実行させる命令を含むことにより特定の動作を実行するように構成することができる。

【０００６】

本方法は、態様１による態様２を含み、態様２では、計算されるメトリックは、直接影響の尺度を提供する。本方法は、態様１又は２による態様３を含み、態様３では、計算されるメトリックは、間接影響の尺度を提供する。本方法は、態様１、２、又は３による態様４を含み、態様４では、提案される変更は、データセット内のフィールドのフォーマットの変更又は変換の変更からなる群の１つである。本方法は、態様１、２、３、又は４による態様５を含み、態様５では、変換は、第１の論理データセットからのデータに適用される１つ又は複数のルールを含み、１つ又は複数の提案される変更を分析することは、１つ又は複数のルールに更に基づく。本方法は、態様１、２、３、４、又は５による態様６を含み、態様６では、本方法は、この態様の他の実施形態の動作を更に含み、メトリックに基づいてコストを提案される変更に関連付けることを含む。

【０００７】

態様は、以下の利点の１つ又は複数を含むことができる。変更を行うスコア及びコストが推定され得る。変更がプログラムに影響するロケーションが識別され得る。リソースが適宜割り振られ得る。

【０００８】

本発明の他の特徴及び利点は、以下の説明及び特許請求の範囲から明らかになる。

【図面の簡単な説明】

【０００９】

【図１】複数の実行可能プログラムのデータ系譜の例を示す。

【図２】ルールセット例及びルールセットへの入力を示す。

【図３】例としての人間可読ルールセット３００を示す。

【図４】ルールを人間可読形態から機械可読コードを含む変換に変換できるようにするプロセスを示す。

【図５】注釈付きコードに基づいてリポートを生成するリポート生成器を示す。

【図６】影響分析技法を使用することができるデータ処理システムの例を示す。

【図７】例示的な影響分析手順のフローチャートである。

【図８Ａ】主要素の相互関係を示す本発明の一実施形態のブロック図である。

【図８Ｂ】データフローグラフのブロック図である。

【図９】ロールアップコンポーネントと、指定されたランタイムパラメータを有するソートコンポーネント９０４とを有する典型的なグラフのブロック図である。

【図１０】グラフに関連付けられるランタイムパラメータグリッドを表すグラフィカルダイアログの一実施形態の図である。

【図１１】ランタイムパラメータを使用するプロセスを要約したフローチャートである。

【図１２】キープロンプトにより生成されるグラフィカルダイアログの一実施形態の図である。

【図１３】フィルタプロンプトにより生成されるグラフィカルダイアログの一実施形態の図である。

【図１４】ロールアッププロンプトにより生成されるグラフィカルダイアログの一実施形態の図である。

【図１５】リフォーマットプロンプトにより生成されるグラフィカルダイアログの一実施形態の図である。

【図１６Ａ】統合結合（MergeJoin）コンポーネントが、ファイルＡ及びＢからのデータを結合し、結果を出力ファイルに出力する第１のグラフのブロック図である。

【図１６Ｂ】ロールアップコンポーネントが、ファイルＡからのデータを集計し、結果を

出力ファイルに出力する第２のグラフのブロック図である。
【図１６Ｃ】統合結合コンポーネントが、ファイルＡ及びＢからのデータを結合し、ロールアップコンポーネントが、結果データを集計し、最終結果を出力ファイルに出力するグラフのブロック図である。
【図１７】条件－解釈制御を有する条件を提示するグラフィカルダイアログの一実施形態の図である。
【図１８】汚染が生じる状況を示すグラフの図である。
【図１９】完全に削除条件コンポーネントを含むグラフのランタイム準備プロセスを要約したフローチャートである。
【図２０】本発明の特定の実施形態でのフローにおいて置換条件コンポーネントを含むグラフのランタイム準備プロセスを要約したフローチャートである。
【図２１】ランタイムパラメータなしのロールアップアプリケーションを表すグラフの図である。
【図２２】図２１のロールアップアプリケーションのランタイムパラメータ化バージョンを表すグラフの図である。
【図２３】図２２のアプリケーション例のランタイムパラメータグリッドを表すグラフィカルダイアログの一実施形態の図である。
【図２４Ａ】図２３のパラメータグリッド内の情報からウェブインタフェースにより生成される形態を表すグラフィカルダイアログの一実施形態の図である。
【図２４Ｂ】ユーザによりパラメータ値で埋められた図２４Ａの形態の図である。
【図２５】ランタイムパラメータ化ロールアップ及び結合アプリケーションを表すグラフの図である。
【図２６】図２５のアプリケーション例でのランタイムパラメータグリッドを表すグラフィカルダイアログの一実施形態の図である。
【図２７】図２６のパラメータグリッド内の情報からウェブインタフェースにより生成される形態を表すグラフィカルダイアログの一実施形態の図である。
【図２８】ランタイムパラメータ化ロールアップ－結合－ソートアプリケーションを表すグラフの図である。
【図２９】図２８に示されるアプリケーション例でのランタイムパラメータグリッドを表すグラフィカルダイアログの一実施形態の図である。
【図３０Ａ】メタデータが伝搬するグラフの図である。
【図３０Ｂ】図３０Ａのグラフでのコンポーネントのサブグラフの図である。
【図３１】メタデータ伝搬プロセスのフローチャートである。
【図３２Ａ】コンポーネント内依存性及びコンポーネント間依存性を有するパラメータを有するグラフである。
【図３２Ｂ】図３２Ａのグラフのパラメータ間の依存性を表す依存性グラフである。
【図３２Ｃ】図３２Ａのグラフのパラメータ間の依存性を表す依存性グラフである。
【図３３】変更トポロジソートプロセスの図である。
【図３４Ａ】従来技術による給与システムの簡略サンプル例を通したデータフローを示すグラフである。
【図３４Ｂ】本発明による図３４Ａのグラフに対応するグラフである。
【図３５】本発明によるドライバプログラムのブロック図である。
【図３６】本発明によるグラフを実行する方法の流れ図である。
【図３７Ａ】図３６に示される方法のステップの流れ図である。
【図３７Ｂ】図３６に示される方法のステップの流れ図である。
【図３８Ａ】図３６に示される方法のステップの流れ図である。
【図３８Ｂ】図３６に示される方法のステップの流れ図である。
【図３８Ｃ】図３６に示される方法のステップの流れ図である。
【図３８Ｄ】図３６に示される方法のステップの流れ図である。
【図３８Ｅ】図３６に示される方法のステップの流れ図である。

【図３８Ｆ】図３６に示される方法のステップの流れ図である。
【図３８Ｇ】図３６に示される方法のステップの流れ図である。
【図３８Ｈ】図３６に示される方法のステップの流れ図である。
【図３９】図３６に示される方法のステップの流れ図である。
【図４０】図３６に示される方法のステップの流れ図である。
【図４１】本発明によるソースアダプタを挿入するステップを示す流れ図である。
【図４２Ａ】説明のためのグラフのブロック図である。
【図４２Ｂ】本発明によるソースアダプタの挿入を示す。
【図４３】本発明が適用されるグラフ例の第１のフェーズである。
【図４４】本発明が適用されるグラフ例の第２のフェーズである。
【図４５】本発明が適用されるグラフ例の第２のフェーズである。
【図４６】本発明が適用されるグラフ例の第３のフェーズである。
【図４７】本発明が適用されるグラフ例の第４のフェーズである。
【図４８】本発明が適用されるグラフ例の第５のフェーズである。
【図４９】本発明が適用されるグラフ例の第６のフェーズである。
【発明を実施するための形態】
【００１０】
説明
一般に、データ処理システムは、データをソースから読み出し、データに対して演算を実行して、新しいデータを生成し、新しいデータをデータストアに記憶することができる。データ処理システムの複雑性は、わずか複雑なものから極めて複雑なものにまで及ぶ。より複雑なシステムでは、データ処理システムに対して行われる変更は、特定が難しいことがある広範囲に及ぶ影響を有し得る。システムに変更を行うことの影響を特定するために、変更により直接影響を受けるシステムの部分と、変更により間接的に影響を受けるシステムの部分とを特定することが有用である。一般に、変更により直接影響を受けるシステムの部分は、個人がシステムを手動で調整する必要があり得る。例えば、プログラマは、アプリケーションの内容及び挙動を変更する必要があり得る。一般に、変更により間接的に影響を受けるシステムの部分では、プログラマが行った変更がアプリケーションの挙動に悪影響を及ぼさないことを保証するために、それらの部分をテストする必要があり得る。
【００１１】
変更の範囲を特定するために、システムを分析して、データがシステムを通ってどのように流れるかを特定する。システムは、システムが使用するデータへの変更又はそのデータの処理への変更が、システムの他の部分にどのように影響し得るかを特定するためにも分析される。
【００１２】
図１は、複数のコンポーネントのデータ系譜１００の例を示す。コンポーネントは、論理データセット及び変換を含むことができる。変換は、例えば、データフローグラフ、javaプログラム、コンパイルされた実行可能プログラム、又はそれらの任意の組合せとすることができる。一般に、変換は、入力データを受け入れることができ、出力データを生成することができる。例えば、グラフ１　１０４は、論理データセット１　１０２及び論理データセット２　１０３から入力データを受け入れ、論理データセット２に提供される出力データを生成する。
【００１３】
一般に、論理データセットは、データを記憶する１つ又は複数の物理データセットを表す。物理データセットは、日毎に変化し得る一意のデータを含み得る。幾つかの実装形態では、異なるデータを有する物理データセットは、別個のファイルに記憶され得る。例えば、１１月８日の外国為替レートデータのデータセットは、ファイル「ExchangeRate_11_08」に記憶し得、１１月９日の外国為替レートデータのデータセットは、ファイル「ExchangeRate_11_09」に記憶し得るが、１１月８日の為替レートデータのデータセット及び１

１月９日の為替レートデータのデータセットは両方とも、共通する要素を有する。例えば、両データセットとも、共通のデータフォーマットを共有する。論理データセットは、それらのデータセットの具体的な内容から独立して、物理データセットに対して演算を実行するプログラム又は一連のプログラムを構築できるようにする。

【００１４】

一般に、論理データセットについての情報は、データセットに記憶されているデータに伴って変わらないデータセットに起因する情報を含む。例えば、論理データセットについての情報は、フィールド名、データ型、レコードフォーマット、制約、及び他の特徴を含み得る。論理データセットは、データソース及びデータシンクとして分類することができる。単一の論理データセットは、ある変換のデータシンク及び別の変換のデータソースであることができる。論理データセットの対応する物理データセットは、例えば、（場所のなかでも特に）関係データベース内のテーブル又はファイルシステム上のファイルであることができる。データソースは、論理データセットに記憶されているデータレコードを読み出すことができ、インメモリデータレコードを作成することができる。コンポーネントは、データソースにより作成されたインメモリデータレコードを受け入れ、データを変更又は変換する。データ値は変更又は変換することができる。新しいデータレコードを作成することができる。データシンクは、データフローグラフからの出口点を提供することができ、出力レコードを記憶することができる。データソースのように、データシンクは、例えば、関係データベーステーブル又はファイルシステムに記憶されたファイルにアクセスすることができる。コンポーネントは、コンピュータ又は他のタイプのコンピュータデバイスで実行することができる。他の実装形態では、データフローグラフの実行は、複数の計算デバイスに分散することができる。

【００１５】

幾つかの実装形態では、コンポーネントは、例えば、入力ポートで入力データを受け入れ、例えば、出力ポートで出力データを生成することができる。リンクは、第１のコンポーネントの出力ポートを第２のコンポーネントの入力ポートに接続する。幾つかのコンポーネントは、複数の入力ポート及び出力ポートを有することができる。データレコードを入口点から出口点にナビゲートすることができるコンポーネント及びリンクの順番は、パスと呼ばれる。データ系譜を使用して、異なるパスを識別し、１つ又は複数のコンポーネントを通るデータフローをトレースすることができる。

【００１６】

この例では、データ要素「ｘ」１０２は、論理データセット１のメンバであり、データ要素「ｙ」１０３は、論理データセット２のメンバである。一般に、データ要素は、論理データセット内に記憶されている個々のレコードを指す。データ要素は、例えば、論理データセットが関係データベース内のテーブルであることができ、データ要素がそのテーブルからの行であることができる。データ要素「ｘ」及びデータ要素「ｙ」は、グラフ１　１４０に入力される。グラフ１は論理データセット２　１０６を生成する。論理データセット２は、データ要素「Ａ」１０８及びデータ要素「Ｂ」１１０を含む。これらのデータ要素はグラフ２　１１２に入力される。データ要素「Ａ」は、データ要素「Ｃ」１１４に使用される。データ要素「Ｃ」は、入力としてルールセット１　１１６に提供される。一般に、ルールセットは、出力を生成するためにデータに適用されるルールの集合である。ルールセットは、例えば、データ要素内の値に適用される一連のテスト及び結果であることができる。ルールセットは、１つ又は複数の入力を受け入れ、それらの入力の値に基づいて、１つ又は複数の出力を生成することができる。一般に、ルールセットは、コンパイルされるか、又はコンピュータ実行可能変換にすることができる。図１に示されるデータ系譜グラフ１００は、説明を目的として及びスペースを考慮して簡略化されている。一般に、線に沿った省略記号の存在は、１つ又は複数のコンポーネント及びデータソースが省略されたことを示す。示されていないデータ変換を行うこともできる。例えば、データ要素「Ａ」１０８を変換して、データ要素「Ｃ」１１４を生成してもよい。データ要素「Ｅ」１１８を変換して、データ要素「Ｇ」１２２を生成してもよい等である。

【００１７】
ルールセット１は、２つの出力データ要素「Ｅ」１１８及びデータ要素「Ｆ」１２０を生成する。データ要素「Ｅ」１１８を使用して、データ要素「Ｇ」１２２を生成する。データ要素「Ｇ」は、入力としてルールセット２　１３０に提供される。ルールセット２は、データ要素「Ｉ」１３２の出力を生成する。データ要素「Ｉ」を使用して、論理データセット３　１３８のデータ要素「Ｊ」１４０を生成する。データ要素「Ｆ」１２０を使用して、データ要素「Ｈ」１２４及びデータ要素「Ｄ」１２６を生成する。データ要素「Ｂ」１１０を使用して、データ要素「Ｍ」１２８を生成する。データ要素「Ｍ」１２８及びデータ要素「Ｄ」１２６は、入力としてルールセット３　１３４に提供される。ルールセット３は、データ要素「Ｋ」１３６を生成する。データ要素「Ｋ」を使用して、論理データセット３　１３８のデータ要素「Ｌ」１４２を生成する。データ要素「Ｙ」は、入力としてルールセット４　１４４に提供される。ルールセット４　１４４は、論理データセット３　１３８のデータ要素「Ｎ」１４６を生成する。
【００１８】
論理データセット又はデータ要素に対して行われる変更は、異なるルールセット及びデータ要素に影響し得る。これらの変更は、変更のなかでも特に、スキーマ又はレコードフォーマットへの変更及び異なるデータ要素の有効値への変更を含むことができる。例えば、データ要素のレコードスキーマが変更される（例えば、レコードスキーマは、数値フィールドから文字列フィールドに変更し得る）場合、変更は、そのデータ要素を利用する各ルールセットと、変更されたデータ要素に依存するデータ要素を利用する各ルールセットとに影響し得る。例えば、データ要素Ｃのレコードフォーマットに対して行われる変更は、ルールセット１、データ要素Ｅ、データ要素Ｆ、データ要素Ｇ、データ要素Ｈ、データ要素Ｄ、ルールセット２、データ要素Ｉ、データ要素Ｊ、ルールセット３、データ要素Ｋ、及びデータ要素Ｌに影響し得る。データ要素Ｘのレコードフォーマットに対して行われる変更は、データ系譜内の他の全ての要素（データ要素Ｙ、ルールセット４、又はデータ要素Ｎを除く）に影響し得る。
【００１９】
システムは、データ要素又はルールセットへの変更の影響についての情報を提供するリポートを生成することができる。例えば、リポート１５０は、グラフ２上のデータ要素Ａに対する変更の影響についての情報を提供する。
【００２０】
リポート１５０は方向列を含む。方向列は、リポートが生成されたデータ系譜の方向を示す。方向は、データ系譜内でそのデータ要素に先行するルールセット、論理データセット、及びデータ要素を指す上流又はデータ系譜内でそのデータ要素に後続するルールセット、論理データセット、及びデータ要素を指す下流のいずれかであることができる。例えば、データ要素Ｃは、ルールセット１の上流であり、データ要素Ａの下流である。
【００２１】
リポート１５０はグラフ列１５４も含む。グラフ列は、リポートのセクションの対象であるグラフを識別する。この例では、グラフ２　１１２がリポートの対象である。リポートは、グラフフィールド列１５６も含む。グラフフィールド列は、リポートの対象であるフィールドを識別する。一般に、フィールドは、方向が下流の場合、グラフへの入力であり、方向が上流の場合、グラフの出力である。この例では、データ要素Ａ１０８及びＢ１１０がリポートの対象である。
【００２２】
リポート１５０はルールセットフィールド列１５８も含む。ルールセットフィールド列は、入力（下流リポートの場合）又は出力（上流リポートの場合）であるデータ要素を識別する。ルールセット列１６０は、リポートの行の対象であるルールセットを識別する。この例では、リポートは、ルールセット１への入力としてデータ要素Ｃについての情報を提供し（第１の牽引１６６において）、ルールセット２への入力としてデータ要素Ｇについて情報を提供し（２行目１６８において）、ルールセット２への入力としてデータ要素

Hについての情報を提供し（３行目１７０において）、ルールセット３への入力としてデータ要素Ｄについての情報を提供し（４行目１７２において）、ルールセット４への入力としてデータ要素Ｍについての情報を提供する（５行目において）１７４。

【００２３】

リポート１５０は、直列列１６２及び間接列１６４も含む。直接列及び間接列は、以下に更に説明するように、コードの分析により特定される。しかし、ここでは、完全性のために提示される。直接列は、ルールセットフィールドにより識別されるデータ要素が、ルールセット内で直接参照される回数を報告する。例えば、直接列は、値を出力に直接割り当てる表現のカウントを含むことができる。間接列１６４は、ルールセットフィールドにより識別されるデータ要素が、ルールセットフィールドにより識別されるルールセット内の１つ又は複数の他のデータ要素の値に影響する回数を識別する。例えば、間接列は、データ要素の出力値に寄与するルール事例及び他の表現の総数のカウントを表示することができる。ビジネスルールにより計算される出力の場合、表現カウントは、デフォルト値がある場合、デフォルト値を含むルール事例の数である。この例では、データ要素「Ｃ」は、ルールセット１において直接１３回参照され、１つ又は複数の他のデータ要素の値に７０回影響する。

【００２４】

リポート１５０を生成するために、システムはルールセットを処理して、いずれのデータ要素がルールセットに関連するかを特定する。ルールセットは、例えば、データをあるフォーマットから別のフォーマットに変換するため、データについて判断するため、又は入力データの組に基づいて新しいデータを生成するために使用することができる基準の組として表すことができる。

【００２５】

図２は、ルールセット例及びルールセットへの入力を示す。上述したように、ルールセット３　１３４は２つの入力：データ要素「Ｄ」１２６及びデータ要素「Ｍ」１２８を有する。ルールセット３　１３４は、「年数」パラメータ２０２及び「収入」パラメータ２０４として入力を参照し得る。この例では「年数」パラメータ２０２及び「収入」パラメータ２０４は、以下により詳細に説明される変換２０６により処理される。変換２０６は「リスク」出力２０８を生成し、この出力はデータ要素「Ｄ」１３６として提供される。

【００２６】

図３は、人間可読ルールセット３００の例を示す。人間可読ルールセット３００は、グラフィカルユーザインタフェース（ＧＵＩ）を使用して定義することができるか、又はフラットファイル若しくは他の構造で定義することができる。人間可読ルールセット３００は、後述するように、後に変換、例えば、図２の変換２０６にコンパイルすることができる。例えば、人間可読ルールセット３００は、図１のルールセット３　１３４にコンパイルすることができる。例えば、データ要素Ｄが収入を表し、データ要素Ｍが顧客としての年数を表す場合である。再び図３を参照すると、人間可読ルールセット３００は表形式で示される。図３に提示される人間可読ルールセット３００は、２つの入力３０２：年数３０６及び収入３０８に基づいて、リスクカテゴリ３１０を出力３０４として特定するのに使用することができる。この例では、７つの潜在的な条件がある。第１のルール３１２は、顧客としての年数が１５年を超える場合、収入に関係なく、リスクが低いことを述べている。第２のルール３１４は、顧客が１５０，０００ドルを超える年収を有する場合、顧客である年数に関係なく、最初が低いことを述べている。第３のルール３１６は、顧客である年数が１０年を超え（しかし、１５年未満）、６０，０００ドルを超える収入を有する場合、リスクが低いことを述べている。第４のルール３１８は、顧客である年数が５年を超える場合、収入に関係なく、リスクが中であることを述べている。第５のルール３２０は、５０，０００ドルを超える収入を有する場合、顧客である時間量に関係なく、リスクが中であることを述べている。第６のルール３２２は、顧客である年数が３年を超え、４０，０００ドルを超える収入を有する場合、リスクが低いことを述べている。第７のルール３２４は、その他の場合、リスクが高いことを述べる包括的なルールである。

【００２７】
なお、この例では、ルールは順次評価される。人物がリスクカテゴリで適格であると、ルール処理は完了する。例えば、人物が顧客である年数が１５年を超え、「低」リスクが割り当てられる（行３１２から）場合、残りの行は決して実行されない。
【００２８】
入力フィールド又は出力フィールドのうちの一方への変更の影響を特定するために、システムは、後述するルールセットの分析を実行することができる。
【００２９】
グラフベースの計算環境でルールセットを実施するために、１つ又は複数のデータソースから入力レコード、例えばデータ要素「Ｃ」１０６を受信し、データ要素、例えば、データ要素「Ｅ」１１８及びデータ要素「Ｆ」１２０を出力データセットに挿入する変換が生成される。入力データセット及び出力データセットは、データストリームと呼ぶこともできる。図１に示されるように、変換は次に、データフローを表す要素をリンクすることにより接続されるデータ処理コンポーネントを有するグラフベースの計算で実施することができる。
【００３０】
図４は、機械可読コードを含む変換に人間可読ルールセットを変換できるようにするプロセスを示す。ルールセット４０２、例えば、図３の人間可読ルールセット３００は、ルール生成器４０８に提供される。ルール生成器は、ルールセット４０２を中間形態にコンパイルする。例えば、ルール生成器４０６は注釈付きコード４０８を生成することができる。注釈付きコード４０８は、ルールへの変更の直接影響及び間接影響を定義する、報告されたメトリック４１０を含むことができる。例えば、人間可読ルールセット３００は、結果として、
【数１】

```
/* デフォルトリスク値 */
/*@
報告されるメトリック:
[
出力参照:
[default risk, 7, 0]
入力参照:
[income, 1, 6]
[years, 1, 6]
]
@*/

if (years > 15) {
      default_risk = "Low"
} else if (income > 150000) {
      default_risk = "Low"
} else if (years > 10 && income > 60000) {
      default_risk = "Low"
} else if (years > 5) {
      default_risk = "Medium"
} else if (income > 50000) {
      default_risk = "Medium"
} else if (years > 3 && income > 40000) {
      default_risk = "Medium"
} else {
      default_risk = "High"
}
```

等の注釈付きコードを生成することができる。
【００３１】
上述したように、直接影響は、ルールセットフィールドにより識別されるデータ要素が、ルールセット内で直接参照されるか、又は設定される回数を記述する。間接影響は、ルールセットフィールドにより識別されるデータ要素が、ルールセット内の１つ又は複数の他のデータ要素の値に影響する回数を識別する。
【００３２】
ルール生成器４０６は、直接メトリック及び間接メトリックを生成することができ、様

々な方法で生じうる。例えば、幾つかの実装形態では、ルール生成器４０６は、ルールセットを分析して、データ要素がアクセスされる各回及び別の値がそのデータ要素に依存する各回を識別することができる。より複雑なシナリオを追跡することもできる。ルール生成器４０６は、どの程度間接的であるかに関係なく、入力値又は出力値の値に依存するあらゆる変数を追跡することができる。例えば、変数が中間値に影響し、その中間値が最終値に影響する場合、システムは、中間値及び最終値の両方を間接影響として報告することができる。例えば、人間可読ルールセット３００は、年数３０６入力の値にアクセスする４つのルールと、収入３０８入力の値にアクセスする４つのルールと、リスク３１０出力の値を設定する７つのルールとを有する。幾つかの実装形態では、ルールセットは、各パラメータの値を少なくとも１回設定すると推測し得る。例えば、年数入力は、入力値がルールセットに提供されるときに設定される。

【００３３】

幾つかの実装形態では、ルール生成器４０６は、少なくとも部分的にパラメータに依存する、ルールセット内のルール数をカウントすることができる。例えば、人間可読ルールセット３００は、年数３０６入力に依存する７つのルールと、収入３０８入力に依存する７つのルールと、リスク３１０出力を設定する７つのルールとを含む。上述したように、ルール３２４は包括的ルールである。幾つかの実装形態では、包括的ルールは、ルール生成器４０６により無視し得る。

【００３４】

注釈付きコード４０８は、ルールコンパイラ４１２に提供することができる。ルールコンパイラ４１２は、注釈付きコード４０８を変換２０６にコンパイルすることができる。一般に、変換は、機械（又は仮想機械）実行可能プログラム、例えば、実行可能プログラム４１６である。

【００３５】

図４を参照すると、リポート生成器は、データ系譜４の注釈付きコード４０２に基づいて、リポート４０８を生成することができる。例えば、再び図１を参照すると、システムは、データ系譜を使用して、例えば、データ要素Ｘがデータ要素Ａ、Ｂ、Ｃ、Ｄ、Ｅ、Ｆ、Ｇ、Ｈ、Ｉ、Ｊ、Ｋ、Ｌ、及びＭの値に影響することを特定することができる。したがって、グラフ２の処理時、システムは、データ要素Ｘへの変更がルールセット１、ルールセット２、及びルールセット３に関与することを特定する。しかし、データ要素Ｘはルールセット４に関与しない。したがって、リポート生成器は、データ要素Ｘへの変更の影響分析の一環として、ルールセット４を分析する必要がないと判断することができる。

【００３６】

図５は、リポート生成器が注釈付きコードに基づいてリポートを生成することを示す。リポート生成器５０６は、いずれの入力が変更により影響されるかを識別し、影響の計算されたメトリックからの結果を記録する。例えば、データ要素「Ｘ」は、ルールセット３への両入力に影響する。したがって、リポート生成器は、メトリック（図４の４１０）をリポート（図１の行１７２、１７４等）に記録する。

【００３７】

幾つかの実装形態では、コストを直接計算及び間接計算のそれぞれに関連付けることができる。例えば、直接影響が、プログラマの所定の時間量と、品質保証人員の所定の時間量とを必要とすることを特定することができる。同様に、間接影響が、品質保証人員の所定の時間量を必要とすることを特定することができる。所定の時間、直接影響及び間接影響の計算並びにコンピュータプログラマの時間及び品質保証人員の時間に関連付けられたコストに基づいて、システムは、分析されるシステムに変更を行うことのコスト推定値を提供することができる。

【００３８】

幾つかの実装形態では、ルール生成器を使用して、システム、例えば、図１のデータ系譜１００により表されるシステムの異なる部分を識別するに当たり、開発者を支援することができる。

【００３９】
図６は、影響分析技法を使用することができるデータ処理システム６００を示す。システム６００はデータソース６０２を含み、データソース６０２は、それぞれが任意の様々なフォーマット（例えば、データベーステーブル、スプレッドシートファイル、フラットテキストファイル、又はメインフレームにより使用されるネイティブフォーマット）でデータを記憶又は提供することができる、記憶デバイス又はオンラインデータストリームへの接続等のデータの１つ又は複数のソースを含むことができる。実行環境６０４は、ルール生成器６０６及びリポート生成器６１２を含む。実行環境６０４は、例えば、あるバージョンのＵＮＩＸオペレーティングシステム等の適するオペレーティングシステムの制御下で１つ又は複数の汎用コンピュータでホストすることができる。例えば、実行環境６０４は、ローカルである（例えば、対称マルチ処理（ＳＭＰ）コンピュータ等のマルチプロセッサシステム）か、ローカルに分散する（例えば、クラスタとして結合される複数のプロセッサ若しくは大規模並列処理（ＭＰＰ）システムか、リモートであるか、リモートに分散する（例えば、ローカルエリアネットワーク（ＬＡＮ）及び／又は広域ネットワーク（ＷＡＮ）を介して結合される複数のプロセッサ）か、又はそれらの任意の組合せの複数の中央演算処理装置（ＣＰＵ）又はプロセッサコアを使用する構成のコンピュータシステムを含む複数ノード並列計算環境を含むことができる。
【００４０】
ルール生成器モジュール６０６は、ルール仕様をデータソース６０２から読み出し、ルールの注釈付きコードを記憶する。データソース６０２を提供する記憶デバイスは、実行環境６０４にローカルである、例えば、実行環境６０４をホストするコンピュータに接続される記憶媒体（例えば、ハードドライブ６０８）に記憶することもでき、又は実行環境６０４にリモートである、例えば、リモート接続（例えば、クラウド計算基盤により提供される）を介して、実行環境６０４をホストするコンピュータと通信するリモートシステム（例えば、メインフレーム６１０）でホストすることもできる。
【００４１】
リポート生成器６１２は、データソース６０２に記憶することができる、ルール生成器６０６により生成される注釈付きコード及びデータ系譜を使用して、変更を行うことの影響のリポートを生成する。出力データは、元のデータソース６０２若しくは実行環境６０４がアクセス可能なデータ記憶システム６１６に記憶することができ６１４、又は別の方法で使用することができる。データ記憶システム６１６は、開発環境６１８にもアクセス可能であり、開発環境６１８において、開発者６２０は、データ要素、他のプログラミング構造物のルールに変更を行うことの影響を特定することが可能である。開発環境６１８は、幾つかの実装形態では、頂点間の有向リンク（作業要素、すなわち、データの流れを表す）により接続される頂点（データ処理コンポーネント又はデータセットを表す）を含むデータフローグラフとして、アプリケーションを開発するシステムである。例えば、そのような環境は、Managing Parameters for Graph-Based Applicationsに関してより詳細に記載されている。そのようなグラフベースの計算を実行するシステムについて以下に説明する。このシステムに従って作られるデータフローグラフは、プロセス間で情報を動かし、プロセスの実行順を定義するために、グラフコンポーネントにより表される個々のプロセスに情報を出し入れする方法を提供する。このシステムは、任意の利用可能な方法からプロセス間通信方法（例えば、ＴＣＰ／ＩＰ若しくはＵＮＩＸドメインソケットを使用することができるリンクに従った通信パス又は共有メモリを使用して、プロセス間でデータを渡す）を選ぶアルゴリズムを含む。
【００４２】
図７は、例示的な影響分析手順７００のフローチャートである。プロセスは、図６のデータ処理システム６００等のデータ処理システムにより実行することができる。
【００４３】
２つのデータセットについての情報が受信される（７０２）。論理データセット情報は、各論理データセットで、その論理データセット内の少なくとも１つのフィールドの識別

子と、そのフィールドについてのフォーマット情報とを識別することができる。
【００４４】
変換についての情報が受信される（７０４）。情報は、２つの論理データセットから、変換がデータを受信する第１の論理データセットと、変換されたデータを提供する第２の論理データセットとを識別することができる。変換は、第１の論理データセットからのデータに適用される１つ又は複数のルールについての情報を含み得、１つ又は複数の提案される変更の潜在的な影響を分析することは、１つ又は複数のルールに更に基づく。
【００４５】
１つ又は複数の提案される変更が受信される（７０６）。提案される変更は、データセット内のフィールドのフォーマットへの変更、変換への変更、又はルールセットへの変更であることができる。幾つかの実装形態では、提案される変更は、変更の性質を指定せずに、変更される論理データセット内のフィールド又は変換を識別する。例えば、提案される変更は、変更が十進法レコードフォーマットから文字列レコードフォーマットへのものであることを示すことなく、フィールド「Ｘ」が変更されることを指定することができる。
【００４６】
提案される変更が分析される（７０８）。
【００４７】
提案される変更のメトリックが計算される（７１０）。メトリックは、変更の影響を測定することができる。メトリックは、直接影響の尺度及び／又は間接影響の尺度を含むことができる。直接影響の尺度の例としては、限定ではなく、変更される入力パラメータがアクセスされるルールセット内のロケーションが挙げられる。間接影響の尺度の例としては、限定ではなく、値が、変更された入力パラメータの値に基づいて設定されるルールセット内のロケーションが挙げられる。
【００４８】
メトリックがソートされる（７１２）。メトリックは、フラットファイル、関係データベース、又は任意の他の永続的データストアに記憶することができる。メトリックは、リポートの形態で記憶し得る。影響のメトリックを識別するリポートを生成することができる。リポートは、直接影響の尺度及び間接影響の尺度をデータ系譜の特定の部分に関連付けることができる。例えば、リポートは、特定のデータフローグラフ、データフローグラフフィールド、ルールセットフィールド、又はルールセットに直接間接の尺度及び間接影響の尺度が関連付けられることを示すことができる。
【００４９】
幾つかの実装形態では、リポートは、例えば、ハイパーテキスト転送プロトコル（ＨＴＴＰ）リンクを通してデータ系譜に結びつけることができる。リンクを選択又はクリックすることで、ユーザがデータ系譜の特定の部分を見られるようにするアプリケーション又はウェブサイトにクライアントデバイス上のブラウザをナビゲートすることができる。例えば、図１を参照すると、３行目１７０を選択又はクリックすることで、クライアントデバイス上のブラウザ又は他のアプリケーションにデータフローグラフ「グラフ２」１１２を表示させることができる。幾つかの実装形態では、その特定のグラフ、グラフフィールド、ルールセットフィールド、及びルールセットは、例えば、強調表示により視覚的に区別することができる。
【００５０】
幾つかの実装形態では、リポートは、提案される変更に関連付けることができる平均開発及びテストコストを含むことができる。例えば、リポートは、ドルコストを直接変更に関連付け、ドルコストを間接変更に関連付けることができる。幾つかの実装形態では、ドルコストは、プロセスに提供されるパラメータであることができる。他の実装形態では、デフォルト値を各変更に関連付けることができる。例えば、直接変更は、コストを１００ドルに決定することができ、間接変更はコストを２５ドルに決定することができる。
【００５１】

グラフベースアプリケーションのパラメータ管理

図８Ａは、主要素の相互関係を示す本発明の一実施形態のブロック図である。グラフィック開発環境（ＧＤＥ）８０２は、実行可能グラフを作成し、グラフコンポーネントのパラメータを定義するユーザインタフェースを提供する。ＧＤＥは、例えば、本発明の譲受人から入手可能なCO>OPERATING SYSTEM（登録商標）ＧＤＥであり得る。ＧＤＥ８０２は、リポジトリ８０４及び並列オペレーティングシステム８０６と通信する。リポジトリ８０４及び並列オペレーティングシステム８０６には、ウェブインタフェース８０８及びエグゼクティブ８１０も結合される。

【００５２】

リポジトリ８０４は、好ましくは、グラフベースアプリケーションの開発及び実行並びにグラフベースアプリケーションと他のシステム（例えば、他のオペレーティングシステム）との間でのメタデータ相互交換をサポートするように設計されたスケーラブルなオブジェクト指向データベースシステムである。リポジトリ８０４は、ドキュメンテーション、レコードフォーマット、変換関数、グラフ、ジョブ、及び監視情報を含め（しかし、これに限定されない）、全ての種類のメタデータの記憶システムである。リポジトリは当技術分野で既知であり、例えば、米国特許第５，９３０，７９４号、同第６，０３２，１５８号、同第６，０３８，５５８号、及び同第６，０４４，３７４号を参照のこと。

【００５３】

並列オペレーティングシステム８０６は、ＧＤＥ８０２において生成されるデータフローグラフの表現を受け入れ、グラフにより定義される処理論理及びリソースに対応するコンピュータ命令を生成する。次に、並列オペレーティングシステム８０６は通常、それらの命令を複数のプロセッサ（同質である必要はない）で実行する。適する並列オペレーティングシステムは、本発明の譲受人から入手可能なCO>OPERATING SYSTEM（登録商標）である。

【００５４】

ウェブインタフェース８０８は、リポジトリ１０４の内容のウェブブラウザベースのビューを提供する。ウェブインタフェース８０８を使用して、ユーザは、オブジェクトを閲覧し、新しいオブジェクトを作成し、既存のオブジェクトを変更し、アプリケーションパラメータを指定し、ジョブをスケジュールするなどを行い得る。ウェブインタフェース８０８は、グラフのランタイムパラメータについてリポジトリ８０４に記憶されている情報に基づいて、パラメータ化グラフのフォームベースユーザインタフェースを自動的に作成する。

【００５５】

エグゼクティブ８１０は、ウェブインタフェース８０８を通してアクセスされる任意選択的なリポジトリベースのジョブスケジューリングシステムである。エグゼクティブ８１０は、ジョブ及びジョブキューをリポジトリ８０４内のオブジェクトとして維持し、ウェブインタフェース８０８は、ジョブ及びジョブキューのビューを提供し、それらの操作に役立つ。

【００５６】

図８Ｂは、入力データセット８２２がフロー８２４によりフィルタコンポーネント８２６に接続された簡略データブローグラフ８２０を示す。フィルタコンポーネント８２６は、フロー８２８により出力データセット８３０に接続される。データセットは、例えば、グラフフローグラフにより実行される計算のためにデータ（例えば、入力データセット）を提供するか、又はデータ（例えば、出力データセット）を受信するファイル又はデータベーステーブルを含むことができる。

【００５７】

データフローグラフ中、「フロー」により表されるデータの流れは、離散データ要素に編成することができる。例えば、要素は、レコード（又は行）及びフィールド（又は列）に編成されたデータセットからのレコードを含むことができる。フィールドシーケンス及びレコード内の値に対応するデータ型を記述するメタデータは、「レコードフォーマット

」と呼ばれる。
【００５８】
グラフ中のコンポーネント及びデータセットは、フローに接続する入力ポート及び／又は出力ポートを有する。フロー８２４及び８２８の「ソースエンド」は、入力データセット８２２の出力ポート及びフィルタコンポーネント８２６の出力ポートとそれぞれインタフェースする。フロー８２４及び８２８の「シンクエンド」は、フィルタコンポーネント８２６の入力ポート及び出力データセット８３０の入力ポートとそれぞれインタフェースする。データセット又はコンポーネントの入力ポート又は出力ポートには、ポートに流れ込むか、又はポートから流れ出るデータのレコードフォーマット等のメタデータが関連付けられる。
【００５９】
ポートのレコードフォーマット又はコンポーネントに関連付けられた他のメタデータを含めパラメータは、パラメータスコーピングのルールに従って値にバインドされる。パラメータは、設計時又は実行時（すなわち、後述する「ランタイムパラメータ」）に値にバインドすることができる。パラメータの値は、例えば、ユーザインタフェースを介してユーザにより定義することもでき（例えば、プロンプトに応答して）、ファイルから定義することもでき、又は同じコンテキスト若しくは異なるコンテキスト内の別のパラメータに関して定義することもできる。例えば、パラメータは、別のパラメータに対して「同じ」関係を有するようにパラメータを指定することにより、異なるコンテキストからエクスポートすることができる（例えば、異なるコンポーネントのコンテキストで評価されたパラメータ）。
【００６０】
グラフで使用されるコンポーネントは、「サブグラフ」を形成するフローと相互接続された他のコンポーネントを使用して実施することができる。サブグラフが、別のグラフでコンポーネントとして使用される前に、コンポーネントの入力ポート及び／又は出力ポート等のそのコンポーネントの様々な特徴が定義される。幾つかの場合、サブグラフコンポーネント間の関係と関係があるコンポーネントの特徴は、そのコンポーネントがグラフで使用される前に指定されるべきである。例えば、サブグラフコンポーネントのランタイムパラメータのプロンプト順を選択する必要があり得る。グラフ中のコンポーネントのランタイムパラメータのプロンプト順を選択する手法について、以下により詳細に説明する。
【００６１】
メタデータ伝搬
レコードフォーマットパラメータ等のポートに関連付けられたメタデータの値は、「伝搬」により得ることができる。メタデータ伝搬は、「外部」又は「内部」で行うことができる。外部メタデータ伝搬の場合、第１のコンポーネントのポートのレコードフォーマットパラメータの値は、フローにより第１のコンポーネントに接続された第２のコンポーネントのポートのレコードフォーマット値を伝搬させることにより、値を得ることができる。値は、フローのソースエンドからシンクエンドに下流に、又はフローのシンクエンドからソースエンドに上流に伝搬することが可能である。メタデータは、定義されたメタデータを有するポートから、定義されたメタデータを有さないポートに伝搬する。
【００６２】
内部メタデータ伝搬の場合、コンポーネントのあるポートに定義されたメタデータは、そのコンポーネントを実施するサブグラフに基づいて、そのコンポーネントの別のポートに伝搬する。幾つかの場合、内部メタデータ伝搬は、「非変換」内部データパスを介して行われる。例えば、ユーザは、ソートコンポーネントに流れ込むレコードのデータ型を指定するメタデータをソートコンポーネントの入力ポートに提供し得る。ソートコンポーネントは、レコードを並べ替えるが、変換しないため、データ型はソートコンポーネントにより変更されず、ソートコンポーネントから流れ出るレコードのデータ型を正確に記述するデータ型は、変更されずにソートコンポーネントの出力ポートに伝搬する。
【００６３】

幾つかのコンポーネントは、コンポーネントを通って流れるデータを変換（又は任意選択的に変換）する。例えば、ユーザは、フィルタコンポーネントに流れ込むレコードのフィールドを指定するメタデータをフィルタコンポーネントの入力ポートに提供し得る。フィルタコンポーネントは、各レコードからの所与のフィールドの値を削除し得る。メタデータ定義を使用して、フィルタコンポーネントの出力ポートのメタデータが、コンポーネントのフィルタリング動作に従って入力ポートのメタデータに関連することを指定することができる。例えば、フィルタリングされたフィールドは、レコードフィールドを指定するメタデータから削除され得る。そのようなメタデータ定義は、入力ポートメタデータが分かる前であっても供給することができる。したがって、メタデータは、以下により詳細に説明するように、別のポートのメタデータを含め、ポートに関連付けられたメタデータを１つ又は複数のパラメータの関数として指定できるようにすることにより、変換内部データパスを介してであっても伝搬することができる。

【００６４】

この内部及び外部メタデータ伝搬は、任意選択的に、グラフが構築中である間、設計時に行うように構成することができ、ユーザは、グラフ中の幾つかのコンポーネントの幾つかのポートにメタデータを供給する。代替的には、メタデータ伝搬は、実行時又は実行時直前を含め、グラフが構築された後に行うことができる。

【００６５】

ランタイムパラメータ

ラインタイムパラメータにより、アプリケーション構築者は、パラメータ値の設定（例えば、ソート関数の主要パラメータ、ファイル名、レコードフォーマット、変換関数等）を実行時（例えば、プログラムが実行されるとき又はコンピュータシステムで間もなく実行されるとき）に延ばすことができる。ランタイムパラメータの値は、エンドユーザにより供給してもよく、又は他のランタイムパラメータの組合せ若しくはオブジェクトリポジトリに記憶されているオブジェクトから導出してもよい。

【００６６】

ランタイムパラメータは、特定量の柔軟性をアプリケーションに追加する。追加の柔軟性は、それらのパラメータをオンデマンドでメタデータ（データフォーマット又はデータ型及びプログラム論理又は変換）の計算に使用することにより達成される。型及び変換は、他の型及び変換、ユーザ供給のパラメータ値、及び記憶されているオブジェクト（例えば、リポジトリから）から合成し得る。これにより、任意のタイプの入力データに対して機能するか、又はランタイムパラメータ値を通して構築が直接又は間接的に制御される一連の変換を通してデータを生成する「汎用」アプリケーションを構築することが可能になる。

【００６７】

幾つかの実装形態では、ランタイムパラメータの作成時又は編集時、開発者は、各パラメータのプロンプト及びプロンプトを表示する条件を指定し得る。システムは、プロンプト指示を解釈して、条件が満たされる場合、パラメータ値を受信するためのグラフィカルユーザインタフェース（ＧＵＩ）制御機構を提示する。

【００６８】

ランタイムパラメータの指定

ランタイムパラメータは、開発者が、グラフ実行時（すなわち、実行時）に外部入力に基づいてグラフの挙動を変更するメカニズムを提供する。好ましい実施形態では、これらの外部値は、直接ユーザ入力により提供される。しかし、これらの外部値は、環境変数及びコマンドラインパラメータを含め、幾つかの異なるソースからのものであってもよい。ＧＤＥ８０２は、これらの全ての状況を扱う補正コードを生成すると共に、グラフがＧＤＥから直接実行されるとき、値をテストするように開発者に促す。ランタイムパラメータを使用して、開発者は、例えば、入力ファイルのパスが、特定の名称を有する環境変数により提供されること、次に、環境変数がグラフのインタフェースの既知の部分になることを明示的に宣言することができる。したがって、そのようなパラメータに明確に定義され

たインタフェースがある。例えば、生成されたシェルスクリプトを読み出し、環境変数及びコマンドライン引数への参照について検索して、特定のグラフの実行を制御するパラメータセットを見つける必要がない。

【００６９】

図９は、指定されたランタイムパラメータを有するロールアップコンポーネント９０２及びソートコンポーネント９０４を有する典型的なグラフ９００のブロック図である。ランタイムパラメータ（ソートコンポーネント９０４のキー及びロールアップコンポーネント９０２のルール）は、入力のためにインタフェース９０６においてユーザに提示される。以下のセクションにおいて、ランタイムパラメータをどのように指定し、ユーザ入力を促すランタイムパラメータを提示する統合ユーザインタフェースをどのように作成するかについて説明する。

【００７０】

ランタイムパラメータは、幾つかの方法で指定又は定義し得る。一方法は、ＧＤＥ８０２に表示されるランタイムパラメータグリッドを使用することによるものである。図１０は、グラフに関連付けられるランタイムパラメータグリッド１０００を表すグラフィカルダイアログの一実施形態の図である。新しいランタイムパラメータは、単に適切なフィールドに入力することにより作成される。各ランタイムパラメータに関連付けられたオブジェクトが、リポジトリ８０４において作成され、そのパラメータを利用する全てのグラフコンポーネントにリンクされる。例えば、グラフのソートコンポーネントのソートキーがランタイムパラメータとして定義される場合、ソートキーパラメータを表すオブジェクトが、リポジトリ８０４に記憶され、関連付けられたソートコンポーネントにリンクされる。ランタイムパラメータを定義する代替の方法は、グラフコンポーネントの既存のパラメータを特にフラグ付け、他のコンポーネントから「見える」（他のコンポーネントにエクスポートする）ようにするものである。これらの方法の組合せを使用してもよい。例えば、コンポーネント作成時、開発者は、そのコンポーネントの特定のパラメータをランタイムパラメータとして指定し得る。次に、開発者は、パラメータグリッドを使用して、グラフの全てのランタイムパラメータのデフォルト値及び他の特徴を設定し、新しいランタイムパラメータを定義し得る。

【００７１】

グラフ実行時、パラメータは処理されて、ユーザ入力又は外部プログラムソース（例えば、コマンドラインパラメータ若しくは環境変数）から各パラメータの値を得る。図示の実施形態では、ランタイムパラメータグリッド１０００は以下のフィールドを含む。

名称１００２　－　このフィールドはランタイムパラメータの名称を含む。「Score_threshold」は、名称に示される例である。

型１００４　－　このフィールドは、ランタイムパラメータで許される値の型を含む。「整数」は型に示される例である。図示の実施形態においてサポートされる型は以下の通りである。

・ブール　－　値は真又は偽のいいずれかであることができる、
・選択　－　値は値リストのうちの１つである、
・コレータ　－　主要パラメータ値、
・データセット　－　外部データファイル及びロケーション、
・日付　－　日付値、
・式　－　算術式、論理式、及び／又は条件式（例えば、select式）、
・フロート　－　浮動小数点、
・整数　－　整数、
・レイアウト　－　並列又は直列レイアウト定義、
・レコードフォーマット　－　レコード記述又はレコード記述を含むファイル、
・文字列　－　任意の文字列、
・変換　－　変換記述又は変換記述を含むファイル。

【００７２】

ロケーション（ロケ）１００６　－　このフィールドは、レコードフォーマット及び変換型と共に使用される。このフィールドは、型フィールド１００４がファイルロケーションを記述するか否か又は型フィールド１００４が埋め込み記述を含むか否かを指定する。サポートされるロケーションは以下の通りである

・埋め込み　－　パラメータはレコード記述又は変換記述を含む、

・ホスト　－　パラメータは、ホスト機上のファイルへの参照を含む、

・ローカル　－　パラメータはローカル機上のファイルへの参照を含む、

・リポジトリ　－　パラメータは、リポジトリ変換又はレコードフォーマットへの参照を含む。

【００７３】

デフォルト値１００８　－　このフィールドは、（１）他の値が外部プログラムソースから提供されない場合に使用される、ランタイムパラメータのデフォルト値又は（２）ユーザ入力からランタイム値をどのように導出するかを記述するルール若しくは式若しくはグラフを実行中のユーザからインタラクティブにその情報を取得する方法を含む。後者の場合、第２のデフォルト値フィールド（図示せず）を使用して、ユーザが入力値を提供しない場合、ランタイムパラメータの値を提供し得る。「ブール」及び「選択」の型では、このフィールドは、ユーザを有効な選択に制限する。「レイアウト」型では、このフィールドは読み取り専用であり、現在定義されているレイアウト定義を表示する。他の全ての型では、このフィールドは、好ましくは、ユーザが有効文字列をタイプし得る単純なテキストエディタである。

【００７４】

編集１０１０　－　パラメータ行中の編集スペース１０１０（又はアイコン、例えば、鉛筆のアイコン）をクリックすると、より高度な編集ウィンドウが開き、このウィンドウは、デフォルト値フィールド１００８を編集する様々なオプションをユーザに提示する。図示の実施形態では、以下のエディタが関連付けられた型に利用可能である。

・シングルライン編集　－　整数、フロート、日付、及び文字列型に関するものである、

・選択ダイアログ　－　ブール及び選択型に関するものである、

・キーエディタ　－　コレータ型に関するものである、

・ファイルブラウザ　－　データセット型及びロケーションが埋め込まれていないレコードフォーマット及び変換型に関するものである、

・変換エディタ　－　埋め込みロケーションを有する変換型に関するものである、

・レコードフォーマットエディタ　－　埋め込みロケーションを有するレコードフォーマット型に関するものである、

・式エディタ　－　式型に関するものである、

・レイアウトエディタ　－　レイアウト型に関するものである。

【００７５】

上記エディタは、種類フィールド値（以下参照）が「ＰＬ」（パラメータ言語）である場合を除いて起動する。この場合、ユーザには、グラフ実行時にパラメータ値を導出又はプロンプトするルールを定義するエディタが提示される。

【００７６】

記述１０１２　－　これは、開発者がランタイムパラメータの予期値を記述する自由形式フィールドである。このフィールドは、デフォルト値が、入力値をユーザに尋ねるルールを含む場合、実行時にプロンプトとして使用される。

【００７７】

種類１０１４　－　このフィールドは、グラフが、グラフ実行時に、関連付けられたパラメータの値を得る場所を定義する。サポートされる種類フィールド１０１４値は以下の通りである。

・環境　－　ランタイムパラメータの値は、同じ名称の環境変数で見つけられることが予期され、環境変数が定義されない場合、デフォルト値フィールド１００８内の値が使用

される。パラメータが必要とされ（すなわち、エクスポートパラメータ）、デフォルト値フィールド１００８が空である場合、ランタイムエラーが生成され、グラフ実行は停止する、

・位置的　－　ランタイムパラメータの値は、アプリケーションを呼び出しているコマンドライン上の相対位置にあることが予期され、例えば、ランタイムパラメータが定義される３番目の位置のランタイムパラメータである場合、そのパラメータ値は、実行スクリプト中の３番目の位置のコマンドライン引数として予期され、指定されるあらゆる位置的パラメータが提供されなければならず、欠けている場合、ランタイムエラーが生成される、

・キーワード　－　ランタイムパラメータの値はキーワードコマンドラインパラメータとして予期され、図示の実施形態では、キーワードパラメータは、

－　＜パラメータ名＞＜パラメータ値＞

の形態であり、キーワードパラメータは、任意選択的であり、キーワードパラメータが提供されず、デフォルト値フィールド１００８が空白であり、対応するエクスポートパラメータが必要とされる場合のみ、ランタイムエラーが生成される、

・固定　－　パラメータのランタイム値は常にデフォルト値であり、これは、２つ以上のランタイムパラメータ間で一定の値を共有するのに有用である、

・ＰＬ　－　ランタイムパラメータのデフォルト値は、グラフ実行時、他のパラメータからのランタイムパラメータの値を導出するか、又は追加の入力をユーザに促すものとして解釈されるＰＬ式を含み、本発明の任意の特定の実施形態との併用に選択されるコンポーネント記述言語は、公開されているオブジェクト指向スクリプト言語「Python」等の任意の適するスクリプト言語であり得、そのようなスクリプトは、プログラム制御下でメタデータ（型及び変換）を構築し、条件付きテスト、比較、データ変換、算術演算及び論理演算、文字列及びリスト操作、並びに他の機能をユーザ入力、外部からプログラム的に供給される入力、及び他のランタイムパラメータに対して実行して、任意のランタイムパラメータの最終値を生成する。

【００７８】

図示の実施形態では、ランタイムパラメータグリッド１０００で直接作成されたランタイムパラメータを参照する有用な規定は、単にドル符号「$」が先行するパラメータ名を入力することである。例えば、$keyは、keyという名称のランタイム変数を参照する。図示の実施形態では、新しいランタイムパラメータのデフォルトは、「文字列」型及びデフォルトランタイム種類の高度オプションダイアログ中の値に基づくデフォルト種類（デフォルトランタイム種類は「環境」）に設定される。

【００７９】

ランタイムパラメータ値は、実行時に決定することができ、ＰＬスクリプトは条件付きテストを提供することができるため、「条件付き」ラインタイムパラメータを作成することができる。条件付きランタイムパラメータは、パラメータの全ての条件　－　実行時に決定される　－　がイネーブルされている場合のみ、ユーザ入力のプロンプトを生成させる。したがって、例えば、データセットをソートするか否かを求める第１のプロントにユーザが「ノー」で応答する場合、ソートキーを求める第２の条件付きプロンプトは表示する必要がない。

【００８０】

したがって、設計フェーズ（「設計時間」）中、開発者は、グラフコンポーネントの特定のパラメータを「ランタイム」パラメータとして指定する。次に、そのグラフコンポーネントに関連付けられたオブジェクトは、関連するパラメータデータ（例えば、図９のパラメータグリッド１０００からの情報の型）と共に記憶される。

【００８１】

図１１は、ランタイムパラメータを使用するプロセスを要約したフローチャートである。実行時中、実行されるアプリケーションに対応するパラメータオブジェクトが検索される（例えば、リポジトリから）（ステップ１１００）。そのような各オブジェクトで、ユ

ーザ入力が示されるか否かを判断する（ステップ１１０２）。示されている場合、プロンプトを表示する任意の条件が満たされているか否かを判断し（ステップ１１０３）、これは、前のプロンプトへのユーザ入力の評価を含み得る。示されていない場合、デフォルト値が使用される（ステップ１１０８）。代替的には、パラメータ値は必要ないことができ（例えば、ユーザがソート機能のアクティブ化を選ばなかった場合、ソートキーは必要ない）、したがって無視し得る。その他の場合、ユーザ入力用のプロンプトが生成される（ステップ１１０４）。

【００８２】

ユーザが特定のパラメータの値を入力しない場合（ステップ１１０６）、パラメータのデフォルト値を選択し得る（ステップ１１０８）。代替的には、エラー状況が生じて、ユーザ入力がないことを示し得る。いずれのイベント（ユーザ入力がないことによるエラー状況がないと仮定して）でも、入力の変換、依存性、及び他のパラメータに基づく条件を考慮して、パラメータの最終値が決定される（ステップ１１１０）。

【００８３】

特定のパラメータへのユーザ入力が示されていないと判断される場合（ステップ１１０２）、環境変数又はコマンドラインパラメータ等により、パラメータ値を外部からプログラム的に供給するか否かが判断される（ステップ１１１２）。供給しない場合、パラメータのデフォルト値が選択される（ステップ１１１４）。代替的には、エラー状況が生じて、指定された型の利用可能な入力がないことを示し得る。いずれの場合（外部入力がないことによるエラー状況がないと仮定して）でも、入力の変換、依存性、及び他のパラメータに基づく条件を考慮して、パラメータの最終値が決定される（ステップ１１１０）。

【００８４】

最終パラメータ値が決定されると、任意選択的なステップとして、全ての条件付きコンポーネント（後述）は、上述した、指定された条件及びルールに従って、完全に削除するか、又はフロー（すなわち、グラフリンク又はエッジ）で置換することができる（ステップ１１１６）。使用可能なグラフ構造が最終化され、最終パラメータ値が決定されると、グラフは従来通りに実行される（ステップ１１１８）。

【００８５】

テスト値

ランタイムパラメータを有するグラフの作成及びテスト中、開発者をサポートするために、ＧＤＥ８０２の好ましい実施形態は、ランタイムパラメータのテスト値もサポートする。開発者が、ランタイムパラメータを有するグラフを実行するか、又はグラフコンポーネントに影響している、土台をなすコードを見ることを望む場合、ＧＤＥ８０２は、関連付けられたテストパラメータグリッドを表示し、そこで、ユーザは、１つ又は複数のランタイムパラメータの新しいテスト値を入力することができる。好ましくは、最後に使用されたテスト値セットは記憶され、グラフと共に保存される。

【００８６】

ランタイムパラメータ毎に、開発者は、所望のテスト値をテスト値列に入力する。編集フィールドを各テスト値列に関連付け得る。テスト値フィールド及び編集フィールドは、パラメータ種類がＰＬである場合を除き、ランタイムパラメータグリッド９００でのデフォルト値フィールド及び編集フィールドと同じように挙動する。

【００８７】

ＰＬ式により、特定のランタイムパラメータの値をユーザに促すことが示される場合、テスト値フィールド及び編集の挙動は、関連付けられたＰＬ式の解釈に基づく。ＰＬ式が単に、他の入力に基づいて値を導出する場合、通常モードで、ランタイムパラメータはテスト値グリッドで見えない。

【００８８】

ランタイムパラメータが値を得る方法を指定

パラメータがランタイムパラメータとして指定された後、対応するオブジェクトがリポジトリ８０４に作成される。ランタイムパラメータが、値「ＰＬ」の種類フィールド９１

４を有する場合、パラメータのデフォルト値フィールド１００８は、以下の好ましい形態を有するprompt_for疑似関数を含む。

【数２】

```
prompt_for “prompt-kind[modifiers]” options …
```

【００８９】

上述したように、prompt_for疑似関数は、前の入力に基づいてプロンプトを表示するか否かを判断する条件式の一部であり得る。

【００９０】

そのようなオブジェクトでは、ユーザインタフェースを使用して、直接入力ランタイムパラメータをユーザに提示する。好ましい実施形態では、ウェブインタフェース８０８がこの機能を提供する。特に、実行時中、各ランタイムパラメータオブジェクトの各prompt_for疑似関数は、ウェブインタフェース８０８によりパーズされて、対応するユーザプロンプトを有するウェブページ（例えば、ＨＴＭＬでの）を生成する。（代替的には、そのようなウェブページは、実行時前に生成し、実行時に単に提示することができる。しかし、そのようなウェブページの実行時生成は、より大きい柔軟性を提供する。特に、ページの内容が前のユーザ入力に依存することができる）。ウェブインタフェース８０８は、そのようなウェブページを表示し、ユーザ入力を受信することができる従来のウェブブラウザと併せて使用される。

【００９１】

prompt_for疑似関数は、パラメータ値を促す方法をウェブインタフェース８０８に示す。特に、文字列定数であるprompt-kindパラメータは、いずれの種類のユーザインタフェース（ＵＩ）要素を提示するかを示す（テキストボックス、ドロップダウンリスト等）。キーワードのカンマ区切りリストである文字列の変更子部分は、様々な種類のプロンプトに共通する幾つかのオプションを提供する。図示の実施形態では、スペースは、変更子文字列内で有意ではない。変更子キーワードは、以下として解釈される。

・in placeというキーワードは、要素がアプリケーションのサマリレベルユーザインタフェースに直接提示されるべきであることを宣言し、より低いレベルまで「掘る」ことなく値を供給できるようにする。in placeが指定されない場合、単純な「編集」ボタンがサマリレベルインタフェースに提示され、そのボタンが、パラメータ値を供給する別のページをユーザに表示する。

・blank okというキーワードは、ユーザが値を供給する必要がないことを宣言し、アプリケーションは、妥当な方法でデフォルト値に対処する。blank okが指定されない場合、ユーザは、何らかの値を供給せずにはアプリケーションを実行することができない。

【００９２】

以下は、様々な種類の変更子を有するprompt_for呼び出しの幾つかの例である。

【数３】

```
${prompt_for “text,inplace”}
${prompt_for “filter, in place”, $input_type}
${prompt_for “radio, blankok, in place”, ${list 1, 2,
3}}
```

【００９３】

このセクションの残りの部分では、様々なプロンプト種類及びそれに対応するオプションを列挙し、それぞれが、ウェブインタフェース８０８により生成されるウェブページでどのように見えるかを説明する。

text [size] － 従来のシングルラインテキストボックスsize文字幅を提示する（sizeが供給されない場合、テキストボックスのデフォルトをブラウザのデフォルトサイズに設定する）。

radio choice-list[description-list] － ラジオボタンの組の形態で従来の「１つ

を選択」プロンプトを提示し、cohice-listの各要素に１つのボタンがある。description-listが供給される場合、各選択は対応する説明でラベルされ、その他の場合、選択は、choice-listからの対応する項目の文字列形態でラベルされる。

radioplus choice-list[description-list] － ラジオのようであるが、テキストボックスの隣に追加のボタンを提示して、ユーザが、choice-listにない「書き込み」値を選べるようにする。

checkbox choice-list[description-list] － チェックボックスの組の形態で従来の「ゼロ以上を選択」プロンプトを提示し、choice-listの各要素に１つのボタンがある。description-listが供給される場合、各選択は対応する説明でラベルされ、その他の場合、選択は、choice-listからの文字列形態の対応する項目でラベルされる。

dropdown choice-list[descritpion-list, size] － choice-listの要素のドロップダウンリストの形態で従来の「１つを選択」プロンプトを提示する。descritpion-listが供給される場合、各選択は対応する説明でラベルされ、その他の場合、選択は、choice-listからの文字列形態の対応する項目でラベルされる。sizeが供給される場合、その多くの選択は一度に可視であり、その他の場合、１つのみが可視である。

multidropdown choice-list[descritpion-list, size] － choice-listの要素のドロップダウンリストの形態で従来の「ゼロ以上を選択」プロンプトを提示する。descritpion-listが供給される場合、各選択は対応する説明でラベルされ、その他の場合、選択は、choice-listからの文字列形態の対応する項目でラベルされる。sizeが供給される場合、その多くの選択は一度に可視であり、その他の場合、ブラウザのデフォルト数の項目が表示される。

key type-obj[size] － 所与のtype-objからのフィールドで構成されるキー（コレータとしても知られる）のプロンプトを提示する。キーは、size部分と同数を有することができ、このデフォルトはtype-obj内のフィールド数に設定される。図１２は、キープロンプトにより生成されるグラフィカルダイアログ１２００の一実施形態の図である。以下は、３入力キープロンプトのスクリプトテキストの例であり、ここで、ファイル/datasets/fixedは、ドロップダウンボックス１２０２に示される利用可能なキーの内容を定義する。

```
${prompt_for “ key ”, ${dataset_type “ /datasets/fixed ”},3}
```

【００９４】

図示の実施形態では、通常のコレータ順は昇順であるが、ユーザは、関連付けられたチェックボックス１２０４にチェックを入れることにより、キーの降順コレータ順を選択することができる。

filter tpye-obj － 所与のtype-objの各フィールドへの条件で構成されるフィルタ式のプロンプトを提示する。blank ok変更子はフィルタに影響を有さず、空白フィルタは「真」表現をもたらす。図１３は、フィルタプロンプトにより生成されるグラフィカルダイアログ１３００の一実施形態の図である。各式のテキスト編集ボックス１３０４に関連付けられた利用可能なフィールド名１３０２は、type-objにより定義される。比較値がテキスト編集ボックス１３０４に入力され、比較演算子（例えば、等しい、より大きい、以下）が、対応するドロップダウンリストコントロール１３０６から選択される。

flexifilter type-obj － フィルタプロンプトと同様であるが、各ライン上のフィールド名がドロップダウンリストから選択可能な、所与のtype-objの各フィールドへの条件で構成されるフィルタ式のプロンプトを提示する。これにより、複数の条件に対して同じフィールドを使用することが可能である（例えば、フィールドSTATE=MA又はフィールドSTATE=CA）。

rollup type-obj key[size] － 所与のキーによりロールアップ中の所与のtype-objのフィールドに基づいて、ロールアップ計算のプロンプトを提示する。ロールアップは、sizeルールと同数を有することができ、デフォルトはtype-obj内のフィールド数に設定される。変更子はロールアップに影響を有さず、空白ロールアップは、各グループの主要値のみを提供するパッケージをもたらす。図１４は、ロールアッププロンプトにより生成さ

れるグラフィカルダイアログ１４００の一実施形態の図である。図示の実施形態では、ドロップダウンボックス１４０２の列は、利用可能なロールアップ計算関数（例えば、合算、最小、最大）を定義する。各計算に関連付けられる利用可能なフィールド名１４０４は、type-objにより定義される。各ロールアップルールには、ユーザが所望の式を定義するためのテキスト編集ボックス１４０６、ソース値が計算に酸化する基準を定義する（ブール式を通して）ための「但し」テキスト編集ボックス１４０８、及び計算結果を受信するフィールドを指定するための出力フィールドテキスト編集ボックス１４１０が関連付けられる。明確に導出することができる場合、出力フィールドの名称を指定する必要はない。

reformat type-obj[size] － 所与のtype-objのフィールドに基づいてリフォーマット計算のプロンプトを提示する。リフォーマットはsizeルールと同数を有することができ、デフォルトはtype-obj内のフィールド数に設定される。図１５は、リフォーマットプロンプトにより生成されるグラフィカルダイアログ１５００の一実施形態の図である。図示の実施形態では、リフォーマットプロンプトは、単に入力フィールドを同様の名称の出力フィールドにコピーするためのセクション１５０２を含む（チェックボックスコントロールを使用して個々に選択／選択解除されるか、又は全ての選択ボタン又は全て選択せずボタンを使用することによりまとめて）。このプロンプトの第２のセクションは、リフォーマット式を定義する（例えば、total=revenue_1-revenue_2）ことができるテキスト編集ボックス１５０４の列を含む。各ルールには、リフォーマットの結果を受信するフィールドを指定する出力フィールドテキスト編集ボックス１５０６が関連付けられる。

outputspec － 出力データセット仕様のプロンプトを提示する。表示されるコントロールは、利用可能なフォーマットオプションを提示するドロップダウンコントロールと、出力データセットの特定のインスタンスの名称を入力するテキスト編集ボックスとを含む。blank ok変更子は、出力データセット仕様に影響を有さない。

fpath starting-point － ファイルパスについてのプロンプトを提示する。プロンプトは基本的にテキストボックスであるが、ファイルパスを閲覧するためのポップアップウィンドウを表示させる「閲覧」ボタンを隣に有する。テキストボックスが非空白である場合、閲覧動作の開始点として使用され、空白の場合、starting-point引数が使用される。

rpath starting-point － リポジトリパスについてのプロンプトを提示する。プロンプトは基本的にテキストボックスであるが、閲覧用のポップアップウィンドウを表示させる「閲覧」ボタンを隣に有する。テキストボックスが非空白である場合、閲覧動作の開始点として使用され、空白の場合、starting-point引数が使用される。

radiofpath choice-list[description-list] － radioplusのようであるが、fpathスタイルボックスプラス閲覧ボタンを「書き込み」スロットに提示する。

radiorpath choice-list[description-list] － radioplusのようであるが、rpathスタイルボックスプラス閲覧ボタンを「書き込み」スロットに提示する。

【００９５】

条件付きコンポーネント

幾つかの実装形態は、パラメータ値及び計算されるメタデータに基づいて、コンポーネントの構造及びグラフの流れに変更を行える条件付きコンポーネントメカニズムを含む。グラフの各コンポーネントは、そのコンポーネントが実行時にグラフに表示されるか否かをコントロールする条件を有する。条件は、直接又はランタイムパラメータを通して間接的に計算することができる。条件付きコンポーネントは、グラフの最適化又は専門化等の様々な目的で使用することができる。最適化の場合、アプリケーションは、特定のデータセットからの値が使用されない場合、それらの特定のデータセットの処理を省くことができ、それにより、グラフをより効率的に実行することができる。専門化の場合、アプリケーションは、所望の詳細レベルに基づいて幾つかの異なる出力データセットの生成を条件付け得るか、又はグラフの幾つかの任意選択的な部分の１つの実行を許可し得る。

【００９６】

図１６Ａは、統合結合（MergeJoin）コンポーネント１６００がファイルＡ及びＢからのデータを結合し、結果を出力ファイル１６０２に出力する第１のグラフのブロック図で

ある。図１６Ｂは、ロールアップコンポーネント１６０４がファイルＡからのデータを集計し、結果を出力ファイル１６０２に出力する第２のグラフのブロック図である。図１６Ｃは、統合結合コンポーネント１６０６がファイルＡ及びＢからのデータを結合し、ロールアップコンポーネント１６０８が、生成されたデータを集計し、最終結果を出力ファイル１６０２に出力するグラフのブロック図である。条件付きコンポーネントを使用して、これらの３つのグラフを結合して、図１６Ｃのグラフのように最初は見えるが、厳密な構造はランタイムまで決定されない単一のグラフにすることができる。適切な条件を設定することにより、ロールアップコンポーネント１６０８はコネクション（フロー）で置換することができ、図１６Ａのグラフと同様のランタイムグラフを生成する。同様に、適切な条件を設定することにより、統合結合コンポーネント１６０６はファイルＡへのコネクション（フロー）で置換することができ、図１６Ｂのグラフと同様のランタイムグラフを生成する。

【００９７】

示される実施形態では、条件付きコンポーネントは、頂点を定義する任意のグラフコンポーネント（すなわち、入力／出力ファイル等のデータセットコンポーネント、リフォーマット若しくはソートコンポーネント等の処理コンポーネント、又はサブグラフとして知られる他のグラフ）であることができる。好ましい実施形態では、条件付きコンポーネントは、２つの特別なパラメータ：条件及び条件解釈により制御される。条件は、評価がランタイムまで延期されるブール式又は値である。示される実施形態では、値「偽」及び「０」は偽条件を指定し、他の全ての値（空を含む）は真条件を示す。条件解釈パラメータは２つの許される相互に排他的な値を有する：完全に削除及びフローで置換。

【００９８】

図１７は、条件解釈コントロール１７０４を有する条件１７０２を提示するグラフィカルダイアログ１７００の一実施形態の図である。条件解釈コントロール１７０４により、完全に削除解釈１７０６又はフローで置換解釈１７０８のいずれかを選択することができる。

完全に削除：この解釈を用いる場合、条件が満たされる場合、コンポーネント及びそれに接続されたフロー（すなわち、グラフリンク又はエッジ）は全てグラフから削除される。アクティブな完全に削除条件は機能的に、コンポーネント及びそれに直接接続された全てのフローをグラフから削除する。完全に削除条件は任意のコンポーネントで使用することができる。

【００９９】

グラフから削除される条件付きコンポーネントは、その条件付きコンポーネントの存在に依存する他の接続されたコンポーネントを「汚染」し、削除させることがある。図１８は、そのような汚染が生じる状況を示すグラフ１８００の図である。入力ファイルコンポーネント１８０２への条件が削除を示し、それに対応する条件解釈が完全に削除である場合、入力ファイルコンポーネント１８０２及びそれに接続されたフローは両方とも、グラフ１８００から削除される。次にこれは、ソートコンポーネント１８０４を汚染し、ソートコンポーネント１８０４を削除させ、その理由は、ソートコンポーネント１８０４の入力は必須の入力ポートであるが、ソートコンポーネント１８０４の入力に接続されているデータフローはもはやないためである。次にこれは、ロールアップコンポーネント１８０６を汚染し、ロールアップコンポーネント１８０６を削除させ、その理由は、ロールアップコンポーネント１８０６の入力は必須の入力ポートであるが、ロールアップコンポーネント１８０６の入力に接続されているデータフローはもはやないためである。この「消失汚染」を止める唯一のものは、下流コンポーネントの任意選択的又は重要なポートへの接続である。したがって、入力ファイルコンポーネント１８０２への条件が削除を示す場合、ソート－ロールアップグラフ分岐１８０８は全体的に、グラフ１８００から事実上、削除される。図１８における結果は、元のグラフ構造の公称的に３つの入力の結合コンポーネント１８１０が、実行時に２つの入力の結合コンポーネントになることである。

【０１００】

一実装形態では、汚染（「黙示条件」としても知られる）の詳細なセマンティクスは以下である。

・コンポーネントが、必須のポートを有し、それに接続されたライブフローがない場合、そのコンポーネント及びそれに接続された全てのフローがグラフから削除される。

・コンポーネントがグラフから完全に削除される場合、そのポートに接続された全てのフローがグラフから削除される。

・コンポーネントがフローで置換される場合、コンポーネントの指定された入力ポート及び指定された出力ポート以外の全てのポートに接続された全てのフローがグラフから削除される。

・必須のインデックス付きポートが、それに接続されたライブフローを有さない場合、同じインデックスを有する対応する任意選択的なインデックス付きポートのそれぞれにつき、その対応するポートに接続されたあらゆるフローがグラフから削除される。

【０１０１】

これらのルールの驚くべき幾つかの結果がある。例えば、任意選択的なポートのみを有するコンポーネントは、汚染により決して削除することができない。したがって、任意選択的なポートのみを有するコンポーネントは、必要に応じて明示的に削除しなければならない。

【０１０２】

図１９は、完全に削除条件付きコンポーネントを含むグラフのランタイム準備プロセスを要約したフローチャートである。条件解釈が完全に削除であり、条件が満たされない場合（ステップ１９００）、条件付きコンポーネントはグラフから削除されない（ステップ１９０２）。条件が満たされる場合（ステップ１９００）、条件付きコンポーネントは、そのコンポーネントに接続された全てのフローと共にグラフから削除される（ステップ１９０４）。次に、全ての「汚染された」コンポーネント及びフローは、上述したルールに従ってグラフから削除される（ステップ１９０６）。

【０１０３】

フローで置換：この解釈を用いる場合、条件が満たされるとき、コンポーネントはフロー（すなわち、グラフエッジ）で置換されるべきである。フローで置換条件解釈は、追加の情報を必要とする。図１７を参照すると、ユーザは、コンポーネントがグラフから削除される場合、接続する入力ポート１７１０（又は必須ポートのファミリ）及び出力ポート１７１２（又は必須ポートのファミリ）を指定する。デフォルトにより、厳密に１つの所要入力ポート又は必須ポートがあり、厳密に１つの所要出力ポート又は必須ポートがある場合、それらは指定されたフロースルー接続ポート（指定入力ポート及び指定出力ポートとそれぞれ呼ばれる）である。所要ポートは、少なくとも１つのフローを接続する必要があるポートである。

【０１０４】

図２０は、本発明の特定の実施形態でのフローで置換条件付きコンポーネントを含むグラフのランタイム準備プロセスを要約したフローチャートである。示される実施形態での特定の利用可能な入力及び出力への幾つかの依存により（CO>OPERATING SYSTEM（登録商標）で利用可能なコンポーネントに基づく）、幾つかのルールがこの実装形態及びフローで置換条件の使用に適用される。

・条件解釈がフローで置換であり、条件が満たされない場合（ステップ２０００）、条件付きコンポーネントはグラフから削除されない（ステップ２００２）。

・指定入力ポート及び指定出力ポートを有するコンポーネントは、その指定入力ポートに接続された厳密に１つのライブストレートフローがあり、その指定出力ポートに接続された厳密に１つのライブストレートフローがある場合のみ、フローで置換することができる（「ライブ」フローは、実行時に削除されなかったフローである）（ステップ２００４）。そのような場合、コンポーネント自体はグラフから削除され、その指定入力ポートに接続されたストレートライブフロー及びその指定出力ポートに接続されたストレートライブフローは、一緒にリンクされる（ステップ２００６）。削除されたコンポーネントの他

のポート（すなわち、特定に指定された入力ポート及び出力ポート以外のあらゆるポート）に直接リンクされたあらゆる他のフローは、グラフから削除される。「汚染」された及び削除されたコンポーネントに接続されていたフローはいずれも、上述したように、削除される（ステップ２００８）。

・フローで置換条件を有するコンポーネントが、必須入力のファミリ内の２つ以上の指定入力ポートに取り付けられる場合（ステップ２０１０）、そのコンポーネントは、グラフを有効にするために必要であるため、グラフから削除されない（ステップ２０１２）。

・所要入力へのライブファンインフローを有するコンポーネントは、特別な取り扱いを必要とする。「ライブファンインフロー」とは、コンポーネントが所要入力ポートに接続されたライブファンイン又はオールツーオールフローを有するか、又はコンポーネントが１つの所要入力ポートに接続された２つ以上のライブストレートフローを有することを意味する。そのようなコンポーネントの場合、フローで置換条件の解釈は、全てのライブ入力フローを収集する収集コンポーネントで条件付きコンポーネントを置換すべきである（ステップ２０１４）。次に、置換されたコンポーネントに接続されていたあらゆる「汚染」フロー及びコンポーネントは、上述したように削除される（ステップ２０１６）。

【０１０５】

メタデータ伝搬の態様

グラフのメタデータは、例えば、グラフ開発者、グラフユーザ、又はグラフの別の部分からの伝搬により供給することができる。ポートのレコードフォーマット（例えば、ポートに流入又は流出するレコードのフィールド及びデータ型のシーケンス）、ソート性（sortedness）、圧縮方法、文字セット、バイナリ表現（ビッグエンディアン、スモールエンディアン）、パーテーション、コンポーネントが使用し得る計算リソース（例えば、プロセッサ、一時的なディスク空間）、データ変換、及びコンポーネントが使用し得るメモリ量等のデータ又はデータでの計算に関連付けられたメタデータを含め、様々な種類のメタデータが伝搬可能である。グラフ構築の様々な態様がメタデータの伝搬に影響を及ぼし得る。これらの態様のうちの２つについて以下説明する。

【０１０６】

コンポーネント削除後の伝搬

幾つかの実装形態では、グラフコンポーネントの削除後にフローが生成される場合、そのようなフロー内のデータを定義するメタデータが、改訂されたグラフ内でどのように伝搬すべきかについて選択しなければならない。メタデータは、フローの一端部から利用可能であり得る。幾つかの実装形態では、フローの上流端部からのメタデータが好ましい。

【０１０７】

フローの上流端部が削除されたコンポーネント（又は収集コンポーネントで置換されたコンポーネント）である場合、ＧＤＥ８０２は、削除されていないコンポーネントを見つけるまで、グラフを上流に「歩く」ことによりフローのメタデータを見つける。その上流コンポーネントにより明るみに出たメタデータを使用して、生成されたフローのデータの特徴を定義する。

【０１０８】

変換されたメタデータの伝搬

上述したように、メタデータは、ポートに関連付けられたメタデータを、別のポートのメタデータを含め、１つ又は複数のパラメータの関数として指定できるようにすることにより、変換中の内部データパスを介してであっても伝搬することができる。例えば、図３０Ａは、データセット３００２及びデータセット３００４からのデータに対して結合演算を計算するグラフ３０００を示す。この例では、グラフ開発者は、メタデータをデータセットの出力ポートに供給する。次に、メタデータは、入力データセットのレコードに対して結合演算を計算する「スマート結合」コンポーネント３００６に伝搬する。例えば、メタデータは、出力ポート３００８から入力ポート３０１０に伝搬する。次に、メタデータは、「スマート結合」コンポーネント３００６により変換され、「スマート結合」コンポーネント３００６の出力ポート３０１６からフィルタコンポーネント３０１８の入力ポー

ト３０１７に伝搬する。
【０１０９】
図３０Ｂは、「スマート結合」コンポーネント３００６を実施するサブグラフを示す。コンポーネント３００６は、結合コンポーネント３０５０により実行される結合演算のキーフィールドを表す値を有するkey_fieldパラメータを使用する。コンポーネント３００６はまた、条件付きソートコンポーネント３０５４及び３０５６を含むための条件として、key_fieldパラメータを使用する。入力ポート３０１０に流入するレコードが、key_fieldで既にソートされている場合、ソートコンポーネント３０５４は条件により除外される。同様に、入力ポート３０１４に流入するレコードが、key_fieldで既にソートされている場合、ソートコンポーネント３０５６は条件により除外される。入力レコードのいずれかの流れがkey_fieldで依然としてソートされていない場合、ソートコンポーネント３０５４及び３０５６は、結合コンポーネント３０５０に流入する前にレコードをソートする。
【０１１０】
この「スマート結合」コンポーネントを通した、変換されたメタデータの伝搬を可能にするために、グラフ開発者は、第１の入力ポート３０１０のメタデータinput0.metadata、第２の入力ポート３０１４のメタデータinput1.metadata、及びキーフィールドパラメータkey_fieldの関数として、「スマート結合」コンポーネント３００６の出力ポート３０１６のメタデータ（例えば、フィールドを記述するメタデータ）を定義する。
output.metadata=metadata_join(key_field,input0.metadata,input1.metadata)
【０１１１】
出力ポートメタデータは、関数引数を値にバインド（適切なコンテキストに関して）し、結果に対して関数metadata_joinを実行することにより特定される。この例では、ポート３０１０及び３０１４のメタデータは未定義であるため、伝搬したメタデータをメタデータパラメータinput0.metadata及びinput1.metadataにバインドされる。ユーザは、ポート３００８から流れるレコードのフィールド「Ａ」及び「Ｂ」を指定する出力ポート３００８のメタデータを「スマート結合」コンポーネント３００６の入力ポート３０１０に供給する。ユーザはまた、ポート３０１２から流れるレコードのフィールド「Ａ」及び「Ｃ」を指定する出力ポート３０１２のメタデータを「スマート結合」コンポーネント３３０６の入力ポート３０１４に供給する。このユーザ供給のメタデータはポート３０１０及び３０１４に伝搬する。結合演算のキーフィールドはフィールドＡであり、したがって、「正式パラメータ」key_fieldは値「Ａ」にバインドされる。
【０１１２】
関数metadata_joinは、まず、key_fieldパラメータの値が、input0.metadata及びinput1.metadataにより指定されるフィールドの両集合のメンバであるか否かを判断することにより、出力メタデータを特定する。メンバである場合、出力メタデータは２つのフィールドの集合の和集合である。メンバではない場合、出力メタデータは空のフィールド集合を示す。
【０１１３】
メタデータが「スマート結合」コンポーネント３００６の入力ポートに伝搬した（又は別の方法、例えば、ユーザにより供給された）後、「スマート結合」コンポーネント３００６の出力ポートの変換されたメタデータは、フィールドＡ、Ｂ、及びＣを含む。次に、この変換メタデータは、他のコンポーネントに伝搬することができる。この例では、変換されたメタデータはフィルタコンポーネント３０１８に伝搬する。
【０１１４】
メタデータは、ユーザにより供給されるか、それともポート間を伝搬するかに関係なく、ユーザに表示することができる。例えば、ユーザは、入力デバイス（例えば、マウス）を使用して、メタデータ値を見るコンポーネントの部分を選択することができる。メタデータ伝搬は、そのようなユーザ選択に応答してトリガーすることもできる。
【０１１５】

例示的なメタデータ伝搬プロセス

図３１は、例示的なメタデータ伝搬プロセス３１００のフローチャートを示す。プロセス３１００は、例えば、グラフが変更される都度、ユーザ活動に応答して、及び／又はグラフが実行される直前に実行することができる。プロセス３１００は、フローにより決定される部分順序に従って並べられるグラフ内の各コンポーネントを有する作業リスト（例えば、コンポーネントＡからコンポーネントＢへのフローがある場合、コンポーネントＡはコンポーネントＢの前に来る）を生成する（３１０２）。フローが２つのコンポーネント間に一意の順序を決定しない場合、コンポーネントラベルのアルファベット順を同点決着として使用し得る。これは、作業リスト内のコンポーネントに安定した順序を提供する（コンポーネントラベルが一意であると仮定して）。伝搬プロセス３１００が、グラフに対して繰り返される（例えば、新しいコンポーネントの追加後）場合、新しい作業リストは、作業リストでの前のコンポーネント間の順序と同じ順序を保持する。

【０１１６】

プロセス３１００は、作業リストの冒頭から開始され、作業リスト内の各リストについて、プロセス３１００は、コンポーネントを実施するサブグラフの仕様（例えば、サブグラフでのデータフロー）に基づいて、コンポーネント内のメタデータを内部伝搬する（例えば、入力ポートから出力ポートに又は出力ポートから入力ポートに）（３１０４）。この内部メタデータ伝搬は、非変換中のデータパスのいずれかの端部でのポート間で変換されないメタデータの変換を含む。内部メタデータ伝搬は、上述したように、グラフのパラメータ及び／又は他のポートのメタデータを参照するメタデータ定義を有するポートのメタデータを導出することも含む。プロセス３１００は、そのようなメタデータ定義に直面する場合、メタデータの導出に値が必要とされるあらゆるパラメータを評価する。

【０１１７】

作業リスト上のコンポーネントに対して内部メタデータ伝搬を実行した後、プロセス３１００は、メタデータを有するコンポーネントの各ポートからメタデータを有さない関連コンポーネントのポートにメタデータを外部伝搬する（３１０６）。この外部伝搬によりメタデータを取得するあらゆるコンポーネントは、作業リストの末尾に移動する（３１０８）。プロセス３１００は、作業リストの最後のコンポーネントが処理された後、終了する（３１１０）。

【０１１８】

このタイプの外部メタデータ伝搬をサポートするコンポーネント間の一種の関係は、２つのコンポーネントのポート間のデータフローリンク（例えば、入力ポートから出力ポートに又は出力ポートから入力ポートに）である。

【０１１９】

このタイプの外部メタデータ伝搬をサポートするコンポーネント間の別の種類の関係は、あるポートのメタデータが別のポートにも使用し得ることを示すリンクである。このタイプの「メタデータリンク」は必ずしもデータフローリンクに対応するわけではない。例えば、ポートは、特にいかなるポートにも関連付けられていないグラフ中のメタデータへのメタデータリンクを有することができる。

【０１２０】

コンポーネント化サブグラフでのランタイムパラメータ

サブグラフが、別のグラフでのコンポーネントとして使用されるように「コンポーネント化」される前に、コンポーネントの入力ポート及び／又は出力ポート等のコンポーネントの様々な特徴が定義される。ランタイムパラメータを有するコンポーネントを含むサブブラフでは、ランタイムパラメータのプロンプト順を選択すべきである。グラフ中のコンポーネントは必ずしも、順次である必要はないため、ユーザを促すランタイムパラメータの複数の可能な大域的順序が存在することができる。大域的順序の幾つかは、各コンポーネントに関連付けられた元の順序ほど一貫しない。依存性を考慮に入れることが適切な場合には並べ替えながら、各コンポーネント内のパラメータの順序を可能な限り保存した、プロンプトの全域的順序を生成することが有用である。例えば、コンポーネントは、「処

理されたデータをどこに記憶したいか？」と尋ねるプロンプトの前に、「いずれのデータを処理したいか？」と尋ねるプロンプトを並べ得る。いずれの順序でもプロンプトを提供することが可能であり得る場合であっても、プロンプトをこの順序で提供することが望ましいことがある。

【０１２１】

プロンプトされるランタイムパラメータを評価するプロセスにおいて、プロンプトされないランタイムパラメータを評価することが必要になることがあるため、プロンプト順は、全てのランタイムパラメータの評価順から得られる。グラフのランタイムパラメータ（いかなるコンポーネントにも関連付けられていないグラフのパラメータを含む）の評価順を決定する一手法は、パラメータ間の依存性を表す１つ又は複数の有向非巡回グラフに基づいて、トポロジソートを実行することを含む。しかし、幾つかのトポロジソートアルゴリズムは、不必要にパラメータを並べ替え得、ランタイムパラメータに望ましくないプロンプト順を生成し得る。

【０１２２】

ソート例１

第１の例では、パラメータソートプロセスは、２つのグラフコンポーネント：コンポーネントＩ及びコンポーネントＩに接続されたコンポーネントＩＩのパラメータの初期パラメータリストを提供する。この例では、パラメータは、「コンポーネント内」依存性のみを有する。すなわち、コンポーネントのパラメータは、同じコンポーネント内の他のパラメータのみに依存する。パラメータは、以下のように定義される。

【０１２３】

コンポーネントＩは以下のパラメータを含む。

【数４】

```
x = ${prompt_for “text”}
y = x + ${prompt_for “text”}
     z = x + y + ${prompt_for “text”}
q = ${prompt_for “text”}
```

【０１２４】

コンポーネントＩＩは以下のパラメータを含む。

【数５】

```
a = ${prompt_for “text”}
b = a + ${prompt_for “text”}
     c = ${prompt_for “text”}
```

【０１２５】

パラメータが列挙される順序は、値についてユーザを促す所望の順序を定義する。初期パラメータリストは、各コンポーネントにこの「初期順序」を維持する。「序数」が各パラメータに割り当てられて、初期順序でのそのパラメータの場所を示す。以下の表は、この初期順序においてパラメータを列挙する。

【０１２６】

【表１】

パラメータ	序数	依存性
X	0	
Y	1	x
Z	2	x, y
Q	3	
A	4	
B	5	a
C	6	

【０１２７】
「依存性」列は、列挙されたパラメータが依存する他のパラメータを示す。依存性は、パラメータの評価に対して順序制約を課す。パラメータは、別のパラメータにより使用される（例えば、参照される）前に定義される必要がある。
【０１２８】
「共通トポロジソート」アルゴリズムが、リストを通り、各パスで、ゼロ依存性を有するパラメータを順序付き出力リストに転送する。各パス後、転送されたあらゆるパラメータは、依存性列から削除される。このプロセスは、全てのパラメータが転送されるまで繰り返される。出力リスト内のパラメータの順序は、他のパラメータに依存するパラメータが、それらの他のパラメータが評価された後に評価されるような「最終順序」を表す。
【０１２９】
この例では、最初のパスにおいて、パラメータｘ、ｑ、ａ、及びｃが出力リストに転送される。２番目のパスにおいて、パラメータｙ及びｂが出力リストに転送される。３番目の最終パスにおいて、パラメータｚが出力リストに転送される。したがって、パラメータの最終順序はｘ、ｑ、ａ、ｃ、ｙ、ｂ、ｚである。この順序はパラメータ依存性により課される順序制約を満たすが、パラメータを不必要に並べ替えない。この例では、初期順序も、パラメータ依存性により課される順序制約を満たす。
【０１３０】
順序制約を満たすグラフのパラメータの評価順を決定する他の手法は、初期順序を尊重する。例えば、幾つかの手法は、順序制約を満たすようにパラメータを順序づけ、初期順序に基づく基準に従って順序を選ぶ。基準は、順序を初期順序に近いまま保つ（例えば、初期順序への変更に基づくメトリックを最小化する）ことを優先する様々な基準のいずれかを含むことができる。幾つかの場合、複数の順序が基準に従って所与の基準を等しく良好に満たすため、一意の「最良」順序がないことがある。
【０１３１】
初期順序を尊重する手法の一例は、「トポロジ変更ソート」手法である。この手法では、初期順序に基づく基準は、いかなる未転送パラメータにも依存しない先行パラメータが転送される前に、初期リストから転送されたパラメータ数を最小化するというものである。換言すれば、「トポロジ変更ソート」は、転送されたパラメータを、ゼロ依存性を有する次のパラメータを転送する前に、依存性列から削除する。上記例では、「トポロジ変更ソート」手法は、初期順序と同じ最終順序ｘ、ｙ、ｚ、ｑ、ａ、ｂ、ｃを生成する。
【０１３２】
初期順序を尊重したトポロジ変更ソートプロセス
両方とも各パラメータに割り当てられた序数により決定される初期順序を尊重する、２つの例示的な「トポロジ変更ソート」プロセスの擬似コードを以下に与える。２番目のプロセスは、幾つかの場合、時間効率を改善するような最適化を含む。プロセスは、パラメータについて入力データから生成されるデータ構造を操作する。
【０１３３】

順序付けるＮ個のパラメータがあると仮定すると、入力データは、一意のパラメータ名、名称付けられたパラメータが依存するパラメータセット（「依存性セット」と呼ばれる）、及び名称付けられたパラメータに関連する情報を記憶する任意選択的な属性データオブジェクトからなるＮ個の三つ組のリストを含む。

【０１３４】

この入力データには、「依存性グラフ」と呼ばれるパラメータ間の依存性を表す１つ又は複数の有向非巡回グラフが関連付けられる。一意の各パラメータ名は、依存性グラフ内のノードに対応し、関連付けられた依存性セットは、他のノードからそのノードへのリンクのセットに対応する。したがって、リンクは、第１のパラメータの第１のノードから、第１のパラメータに依存する第２のパラメータの第２のノードを指す。代替的には、リンク方向とパラメータ依存性との対応を逆にすることができる。

【０１３５】

出力データ構造result_listは、パラメータが、初期順序に近い状態を保つことを優先しながら、別のパラメータの評価に使用される前に評価されるように（必要に応じて）並べ替えられた入力データからのＮ個のパラメータのリストを含む。出力データ構造result_listを生成するために、プロセスは、パラメータを１度に１つずつ作業データ構造param_listから出力データ構造result_listに転送することにより、パラメータを「なくす」。出力データ構造は、全てのパラメータがなくなった後に完成する。

【０１３６】

第１の「トポロジ変更ソート」プロセスは２つのフェーズを含む。第１のフェーズでは、プロセスは、ソートされた出力データ構造に使用するために、入力データに基づいて作業データ構造を構築する。第２のフェーズでは、プロセスは、これらの作業データ構造により表される依存性制約に従って、パラメータを繰り返しソートしてなくす。

【０１３７】

プロセスが第１のフェーズにおいて構築する作業データ構造の幾つかは、ディクショナリであり、これは、ハッシュ法に基づくデータ構造である。ディクショナリ内の項目には、Ｏ（ｌｏｇＮ）時間で効率的にアクセスすることができる。以下の例示的なデータ構造が、第１のフェーズにおいて構築される。

parm_list[index]：番号インデックスが付与された非消失パラメータ名の順序付きリスト（ここで、インデックス＝０はリスト中の最初の項目に対応する）。このデータ構造は「動的」である（すなわち、プロセスの実行中に変化する）。リストは、項目がリストの中央から削除される場合、削除された項目後の項目のインデックスはそれに従ってシフトするように、位置によりインデックス付与される。

n_dependencies_dict[name]：パラメータ名（name）をキーとしたディクショナリであり、エントリは、キーとなるパラメータが依存するパラメータ数を含む。このディクショナリは動的である。

dependers_dict[name]：パラメータ名（name）をキーとしたディクショナリであり、エントリは、キーとなるパラメータに依存するパラメータセットを表すディクショナリ（これもパラメータ名をキーとする）である。このディクショナリは「静的」である（すなわち、プロセスの実行中に変化しない）。

order_doct[name]：パラメータ名（name）をキーとしたディクショナリであり、初期順序でのパラメータの順序位置である０～Ｎ－１の範囲の整数を記憶する。このディクショナリは静的である。

attribute_dict[name]：パラメータ名（name）をキーとしたディクショナリであり、キーとなるパラメータの任意選択的な属性データオブジェクトを記憶する。このディクショナリは静的である。

result_list[index]：番号インデックスが付与されたプロセスの出力を表すパラメータ名及び属性の順序付きリストである（ここで、インデックス＝０はリスト中の最初の項目に対応する）。このデータ構造は最初空である。このデータ構造は動的である。

【０１３８】

プロセスの時間効率を分析するために、依存性グラフの平均「度」（又はノードからのリンク数）がｚであると仮定する。これらのデータ構造の構築には、Ｏ（Ｎ＊ｚ）時間がかかるn_dependencies_dict及びdependers_dictを除き、Ｏ（Ｎ）時間がかかる。

【０１３９】

第２のフェーズにおいて、プロセスは、まず、依存する非消失パラメータの数により（すなわち、パラメータのn_dependencies_dictの値により）最低から最高にパラメータを順序付け、次に、パラメータの順序により（すなわち、order_dictの値により）最低から最高に順序付けるソート基準by_n_deps_and_orderに従って、param_listデータ構造内のパラメータをソートする。次に、プロセスは、ソートされたparam_listにおいて最初のパラメータを消す。このパラメータのn_dependencies_dictの値はゼロであるべきである。（ソートされたparam_list中の最初のパラメータのn_dependencies_dictの値がゼロではない場合、エラーがフラグ付けられる）。

【０１４０】

パラメータを消すために、プロセスは、パラメータをresult_list（対応するあらゆる属性と共に）に添付し、そのパラメータに依存する全てのパラメータ（すなわち、dependers_dict中のパラメータ）の依存性カウント（すなわち、n_dependencies_dictの値）を１だけデクリメントする。最後に、パラメータはparm_listから削除される。結果として最初になるパラメータのこのソート及び消失は、全てのパラメータがなくなるまで繰り返される。

【０１４１】

以下は、消失手順の疑似コード定義である。

【数６】

```
def eliminate(list, index):
    result_list.append( (list[index], attribute_dict[list[index]]))
    for depender in dependers_dict[list[index]]:
        n_dependencies_dict[depender] = n_dependencies_dict[depender] - 1
    delete list[index]
```

【０１４２】

消失手順の引数はリスト（値は、例えば、param_listである）及びインデックスである。関数result_list.appendは、位置インデックスにおいて示されたリスト項目をそれに関連付けられた属性と共にresult_listに添付する。次に、手順は、消失したパラメータをキーとしたdependers_dictデータ構造のメンバである、消失したパラメータに依存する各パラメータのn_dependencies_dictの値をデクリメントする。次に、手順は、listからそのパラメータを削除する。消失手順の実行時間はＯ（ｚｌｏｇＮ）である。

【０１４３】

以下は、第１の「トポロジ変更ソート」プロセスのソート／消失ループの疑似コードである。

【数７】

```
while parm_list is not empty:
    parm_list.sort(by_n_deps_and_order)
    while parm_list is not empty and n_dependencies_dict[parm_list[0]] == 0:
        eliminate(parm_list, 0)
        parm_list.sort(by_n_deps_and_order)
    if parm_list is not empty and n_dependencies_dict[parm_list[0]] > 0:
        delete parm_list[0]
        < 循環エラーを記録し、継続 >
```

【０１４４】

プロセスはまず、上述したソート基準by_n_deps_and_orderに従ってparam_listのパラ

メータを順序付ける関数parm_list.sort(by_n_deps_and_order)を使用して、param_listの初期ソートを実行する。次に、プロセスは消失手順を実行し、その後、param_listが空になるまで、param_listの別のソートを実行する。プロセスは、param_list内の最初のパラメータ（インデックス＝０を有する）への依存数がゼロになることを確実にするためにチェックする。ゼロではない場合、プロセスはそのパラメータを削除し、循環エラーを記録し、継続する。ソートはＯ（Ｎ＼ｌｏｇＮ）かかり、ループ範囲はＮであり、したがって、ループの全体実行時間の推定はＯ（Ｎ２ｌｏｇＮ）である。

【０１４５】

第２の「トポロジ変更ソート」プロセスは、ｚ＜＜Ｎであるように依存性グラフが疎である場合を利用する。１回の初期ソート後、プロセスは、いかなる他のパラメータにも依存しないパラメータのリスト候補のソート性を維持することができる。これは、この予期される実行時間を後述するように短縮する。

【０１４６】

以下は、第２の「トポロジ変更ソート」プロセスの疑似コードである。

【数８】

```
parm_list.sort(by_n_deps_and_order)
while parm_list is not empty:
    # section 1
    candidates = []
    for p in parm_list:
        if n_dependencies_dict[p] == 0:
            candidates.append(p)
    # section 2
    while candidates is not empty and n_dependencies_dict[candidates[0]] == 0:
        this_parm = candidates[0]
        eliminate(candidates, 0)
        idx = parm_list.index(this_parm)
        delete parm_list[idx]
        tmp = get_new(this_parm)
        candidates = merge(candidates, tmp)
    # section 3
    if parm_list is not empty:
        parm_list.sort(by_n_deps_and_order)
        if n_dependencies_dict[parm_list[0]] > 0:
            delete parm_list[0]
            < 循環エラーを記録し、継続 >
```

【０１４７】

プロセスはまず、上述したソート基準by_n_deps_and_orderに従って、param_listのパラメータを順序付ける関数parm_list.sort(by_n_deps_and_order)を使用して、param_listの初期ソートを実行する。次に、プロセスは、３つのセクション（「#section 1」、「#section 2」、及び「#section 3」と記される）を有するループを実行する。

【０１４８】

セクション１において、プロセスは、ゼロ依存性を有するパラメータのみを含む候補リストを構築する。プロセスは、parm_list中の全てのパラメータをスキャンし、それらを候補に添付し、相対順序を保存する。

【０１４９】

セクション２において、プロセスは、候補からのパラメータがなくなり、新しいパラメ

ータが候補に統合されるループを実行する。this_parmとして保存される候補中の第１のパラメータは、候補からなくなり、param_listから削除される。関数get_new(this_parm)は、新たに消えたthis_parmでのdependers_dictのメンバであり、ゼロ依存性のままであるパラメータの名称のリストを返す。次に、最後の依存性がなくなったパラメータを表すこれらのパラメータは、by_n_deps_and_orderに従ってソートされ（それぞれの序数に従って並べられることを保証するために）、候補に統合される。したがって、候補リストは、序数によりソートされたゼロ依存性パラメータのリストのままである。

【０１５０】

例えば、２つのパラメータが互いに関して定義される場合に生じる「循環エラー」がある場合のみ、セクション３に入る。この場合、プロセスは、parm_listを再びソートし、parm_list中の最初のパラメータは、非ゼロ依存性を有する場合、削除され、ループはセクション１から繰り返される。

【０１５１】

循環エラーがないと仮定すると、Ｎパラメータリストであるparm_listは、最初のみソートされ、その結果、ソート時間はＯ（ＮｌｏｇＮ）である。その後、ソートは、候補リストの最上部にあるパラメータをなくした結果生成される、新たに生成されるゼロ依存性パラメータのはるかに小さいリストに対してのみ行われる。このリストのサイズはｚ未満であり（平均で）、その結果、ソート時間はＯ（ｚｌｏｇｚ）であり、統合時間はＯ（ｚ）である。したがって、ループの一反復はＯ（ｚｌｏｇｚ）であり、総時間はＯ（Ｎｚｌｏｇｚ＋ＮｌｏｇＮ）である。ｚがＮの増大に伴って増大しない場合、この時間は事実上、Ｏ（ＮｌｏｇＮ）である。

【０１５２】

ソート例２

別の例では、パラメータソートプロセス（例えば、第１又は第２の「トポロジ変更ソート」プロセス）は、図３２Ａに示されるように、グラフコンポーネント３２０２、３２０４、及び３２０６を有するグラフ３２００のランタイムパラメータの初期リストを決定する。グラフ３２００は、入力データセット３２１０の出力ポート３２０８及び出力データセット３２１４の入力ポート３２１２に関連付けられたランタイムパラメータも有する。この例では、パラメータは、「コンポーネント内」依存性及び「コンポーネント間」依存性の両方を有する。すなわち、コンポーネントのパラメータは、同じコンポーネント内のパラメータに依存すると共に、他のコンポーネント内のパラメータに依存する。この例では、コンポーネント間依存性は、幾つかのパラメータが依存するメタデータの伝搬を可能にするコンポーネント間のフローに起因して生じる。

【０１５３】

依存性は、図３２Ａにおいて、第１のパラメータ又はポートから第２のパラメータ又はポートへの点線矢印で示される。ポートへの矢印は、リンクされたパラメータの値がそのポートから下流ポートに伝搬することを示す。ポートからの矢印は、値が上流ポートからリンクされたパラメータに伝搬することを示す。第１のパラメータから第２のパラメータへの矢印は、第２のパラメータの値が第１のパラメータの値に依存（例えば、参照）することを示す。

【０１５４】

図３２Ｂは、グラフ３２００に基づくパラメータｐ０、ｐ１、ｐ２、ｐ４、ｐ５、及びｐ６間の順序制約を表す依存性グラフ３２５０を示す。図３２Ｃは、グラフ３２００に基づくパラメータｐ３、ｐ７、ｐ８、及びｐ９間の順序制約を表す依存性グラフ３２５２を示す。

【０１５５】

パラメータソートプロセスは、グラフ３２００での要素の配置の順序に従って、様々なグラフ要素の１０個のパラメータｐ０、ｐ２、．．．、ｐ９のそれぞれに序数を割り当てる。図３２Ａでは、グラフ３２００に追加される（例えば、ＧＤＥ８０２を使用してユーザにより）最初のグラフ要素は、パラメータｐ０、ｐ１、及びｐ２を有するコンポーネン

ト３２０２である。追加される２番目の要素は、パラメータｐ３、ｐ４、及びｐ５を有するコンポーネント３２０６である。追加される３番目の要素は、パラメータｐ６を有するデータセット３２１０である。追加される４番目の要素は、パラメータｐ７を有するデータセット３２１４である。追加される最後の要素は、ランタイムパラメータを有さないデータセット３２１６である。以下の表は、割り当てられた序数により定義される初期順序でパラメータを列挙する。

【０１５６】

【表２】

パラメータ	序数	依存性
p0	0	
p1	1	p0, p6
p2	2	p6
p3	3	p8
p4	4	p1
p5	5	p1
p6	6	
p7	7	p3
p8	8	
p9	9	p8

【０１５７】

様々な処理段階でのparam_list及びresult_list中のパラメータの以下のリストは、上述した第１の「トポロジ変更ソート」プロセスに対応する。param_listは、各段階においてソート基準by_n_deps_and_orderに従ってソートされて示される。

【数９】

param_list	result_list
p0 p6 p8 p2 p3 p4 p5 p7 p9 p1	空
p6 p8 p1 p2 p3 p4 p5 p7 p9	p0
p1 p2 p8 p3 p4 p5 p7 p9	p0 p6
p2 p4 p5 p8 p3 p7 p9	p0 p6 p1
p4 p5 p8 p3 p7 p9	p0 p6 p1 p2
p5 p8 p3 p7 p9	p0 p6 p1 p2 p4
p8 p3 p7 p9	p0 p6 p1 p2 p4 p5
p3 p9 p7	p0 p6 p1 p2 p4 p5 p8
p7 p9	p0 p6 p1 p2 p4 p5 p8 p3
p9	p0 p6 p1 p2 p4 p5 p8 p3 p7
空	p0 p6 p1 p2 p4 p5 p8 p3 p7

p9

【０１５８】

様々な処理段階での候補及びresult_list中のパラメータの以下のリストは、上述した第２の「トポロジ変更ソート」プロセスに対応する。パラメータは各段階で同じ順序のま

まであるため、段階間で候補をソートする必要はない。

【数１０】

候補	*result_list*
p0 p6 p8	空
p6 p8	p0
p1 p2 p8	p0 p6
p2 p4 p5 p8	p0 p6 p1
p4 p5 p8	p0 p6 p1 p2
p5 p8	p0 p6 p1 p2 p4
p8	p0 p6 p1 p2 p4 p5
p3 p9	p0 p6 p1 p2 p4 p5 p8
p7 p9	p0 p6 p1 p2 p4 p5 p8 p3
p9	p0 p6 p1 p2 p4 p5 p8 p3 p7
空	p0 p6 p1 p2 p4 p5 p8 p3 p7

p9

【０１５９】

したがって、図３３を参照すると、「トポロジ変更ソート」プロセス３３００は、入力として、ランタイムパラメータの値をユーザに促すための所望の第１の順序３３０２及びパラメータの順序制約３３０４（例えば、依存性グラフ３３５０及び３３５２）をとる。プロセス３３００は、所望の第１の順序３３０２に従って順序制約を満たすパラメータセットの新しい順序３３０６を提供する。

【０１６０】

典型的な使用

通常、ユーザは、ウェブインタフェース８０８の前に座り、リポジトリ１０４において、実行しようとするアプリケーションのグラフを見つける。アプリケーショングラフに関連付けられた全てのオブジェクトをスキャンすることにより、ウェブインタフェース８０８は、ユーザがアプリケーションのランタイムパラメータの値を指定することができるウェブページフォームを生成する。全てのランタイムパラメータが指定されると、アプリケーションとパラメータ設定との組合せはジョブとして一緒にされ、エグゼクティブ８１０による実行にスケジュールされる。ジョブを実行するとき、エグゼクティブ８１０は、既知の方式での並列オペレーティングシステム８０６の下における実行のためにアプリケーションをキューに入れる。並列オペレーティングシステム８０６は、追跡情報及びジョブステータスを収集し、この情報をリポジトリ８０４に記憶し、それにより、ユーザ及び管理者は、ジョブの進行及び実行を追跡することができる。

【実施例】

【０１６１】

例

図２１は、ランタイムパラメータがないロールアップアプリケーションを表すグラフ２１００の図である。このグラフは、各種のアカウント数を計算し、結果を出力ファイルに書き込む。このアプリケーションのあらゆる態様は、グラフを作成した開発者によって決定されている：入力ファイルコンポーネント２１０２の名称、入力データのフォーマット、ハッシュロールアップコンポーネント２１０４でのデータのロールアップに使用されるキー及び変換ルール、出力フォーマット、並びに出力ファイルコンポーネント２１０６の名称。ユーザは、厳密に定義されたようにこのグラフを実行することのみ可能である。

【０１６２】

図２２は、図２１のロールアップアプリケーションのランタイムパラメータ化バージョンを表すグラフ２２００の図である。このアプリケーションのデータフローグラフ構造は、非ランタイムパラメータ化バージョンにかなり類似するが、アプリケーションははるか

に柔軟である。ランタイムパラメータを通して、エンドユーザは、抽象入力データセット２２０２（入力ファイル名及びフォーマットが導出される保存オブジェクト）の名称、ハッシュロールアップコンポーネント２２０４のロールアップキー及びロールアップルール、並びに出力ファイルコンポーネント２２０６の名称を指定し得る。

【０１６３】

図２３は、図２２のアプリケーション例のランタイムパラメータグリッド２３００を表すグラフィカルダイアログの一実施形態の図である。これは、図９に示されるパラメータグリッドの入力済みバージョンである。なお、デフォルトパラメータの数は、上述したように、prompt_for疑似関数を使用して定義され、したがって、ウェブインタフェース８０８を通したユーザ入力を必要とする。このグラフの外観は、非ランタイムパラメータ化アプリケーショングラフと殆ど変わらないが、１つ又は複数のパラメータグリッド（又は他の適するコントロール）により、開発者は、グラフの実行を制御する全てのパラメータを完全に追跡することができる。

【０１６４】

図２４Ａは、図２３のパラメータグリッド２３００内の情報からウェブインタフェース８０８により生成されるフォーム２４００を表すグラフィカルダイアログの一実施形態の図である。この例では、フォーム２４００は、ユーザ入力の４つのランタイムパラメータを提示する：入力データセットリポジトリパス２４０２、ロールアップキー２４０４、ロールアップルール２４０６、及び出力パス２４０８。図１７Ｂは、ユーザがパラメータ値を入力した図２４Ａのフォーム２４００の図である。直接入力及び／又はランタイムパラメータ２４０２～２４０８に関連付けられた編集又はブラウザ制御ボタンを使用して、ユーザは、関連付けられたグラフを実行するための対応するパラメータ値２４１０～２４１６を提供する。

【０１６５】

図２５は、ランタイムパラメータ化ロールアップ及び結合アプリケーションを表すグラフ２５００の図である。図２６は、図２５のアプリケーション例のランタイムパラメータグリッド２６００を表すグラフィカルダイアログの一実施形態の図である。ここで、アプリケーションの幾つかの態様はパラメータ化されているが、結合キー及び入力データセットを含め、大半は固定されたままである。図２７は、図２６のパラメータグリッド２６００内の情報からウェブインタフェース８０８により生成されるフォーム２７００を表すグラフィカルダイアログの一実施形態の図である。なお、ロールアップへの入力タイプは、トップレベルフォームが表示されるときには分かっているため、ロールアップルール２７０２は現場でプロンプトすることができる。

【０１６６】

図２８は、ランタイムパラメータ化ロールアップ－結合－ソートアプリケーションを表すグラフ２８００の図である。図２５の例と同様であるが、条件付きソートコンポーネント２８０２がグラフ２８００に追加されている。図２９は、図２８に示されるアプリケーション例のランタイムパラメータ化グリッド２９００を表すグラフィカルダイアログの一実施形態の図である。sort_keyランタイムパラメータ２９０２は、ソートが望ましいことをユーザが示す場合のみ、プロンプトされる。この効果を得るために、開発は、sort_keyランタイムパラメータ２９０２のデフォルト値２９０４のif条件付きテスト内にprompt_for疑似関数を配置する。if条件付きテストは、第２のランタイムパラメータdo_sort２９０６を参照する。do_sortパラメータ２９０６のデフォルト値フィールド２９０８及び説明フィールド２９１０は、テキストプロンプト「データをソートすべきか？」に対する真／偽又はイエス／ノーの答えについてユーザに尋ねるラジオプロンプトを生成するために定義される。do_sortパラメータ２９０６に提供された値が「真」である場合、ソートコンポーネント２８０２は、実行時にグラフの一部として含まれることになる。その他の場合、ソートコンポーネント２８０２は、指定された条件解釈に応じて、グラフから完全に削除されるか、又はフローで置換されることになる。

【０１６７】

スクリプト実施

ＧＤＥ８０２は、パラメータ化グラフの構築に役立つが、フォームベースのインタフェースを提供しようとする非グラフプログラムがあることがある。アプリケーションレベルＰＬ及びリポジトリ８０４を使用して、任意のシェルスクリプトをパラメータ化することができる。例えば、アプリケーションの説明は、以下と同様の構造を有するファイルに書き込むことができる。

【数１１】

```
application AppName(
 description("One-line Description"),
 comment("Longer description"),

 parameter ParmName1(
 string, kind(keyword), required,
 description("Short prompt for top-level form"),
 comment("Longer prompt for out-of-line form"),
 default(${prompt_for …})
 ),

 parameter ParmName2(
 type, kind(derived),
 default(PL-expression)
 ),

 . . . より多くのパラメータ . . .

 script(="scriptname.ksh")
)
```

【０１６８】

一般コンピュータ実施

本発明は、ハードウェア、ソフトウェア、又はそれら両方の組合せ（例えば、プログラマブル論理アレイ）で実施し得る。別段のことが指定される場合を除き、本発明の一環として含まれるアルゴリズムは本質的に、いかなる特定のコンピュータ又は他の装置にも関連しない。特に、様々な汎用マシンが、本明細書の教示に従って記述されたプログラムと併用してもよく、又は必要とされる方法ステップを実行するようにより専用化された装置を構築することがより好都合であることもある。しかし、好ましくは、本発明は、少なくとも１つのプロセッサ、少なくとも１つのデータ記憶システム（揮発性及び不揮発性メモリ及び／又は記憶要素を含む）、少なくとも１つの入力デバイス又はポート、及び少なくとも１つの出力デバイス又はポートをそれぞれ含む１つ又は複数のプログラマブルコンピュータシステムで実行される１つ又は複数のコンピュータプログラムで実施される。プログラムコードは、プロセッサで実行されて、本明細書に記載の機能を実行する。

【０１６９】

そのような各プログラムは、任意の所望のコンピュータ言語（機械、アセンブリ、又は高水準手続き型、論理型、又はオブジェクト指向型プログラミング言語を含む）で実施されて、コンピュータシステムと通信し得る。いずれの場合でも、言語はコンパイル型又はインタープリタ型言語であり得る。

【０１７０】

そのような各コンピュータプログラムは、好ましくは、汎用又は専用プログラマブルコ

ンピュータにより可読な記憶媒体又はデバイス（例えば、固体状態、磁気、又は光学媒体）に記憶され、記憶媒体又はデバイスがコンピュータシステムにより読み取られた場合、本明細書に記載の手順を実行するように、コンピュータを構成し動作させる。本発明のシステムは、コンピュータプログラムが構成されたコンピュータ可読記憶媒体として実施されるものとして見なすこともでき、記憶媒体は、コンピュータシステムを特定の予め定義された方式で動作させて、本明細書に記載の機能を実行させるように構成される。

【０１７１】

本発明の幾つかの実施形態について説明した。それにもかかわらず、本発明の趣旨及び範囲から逸脱せずに、様々な変更形態をなし得ることが理解される。例えば、上述した幾つかの機能ステップは、全体プロセスに実質的に影響せずに、異なる順序で実行し得る。例えば、図１１のステップ１１０２及び１１１２は、逆の順序で実行し得る。したがって、他の実施形態も以下の特許請求の範囲内にある。

【０１７２】

グラフとして表現される計算の実行

計算基板の概説

本発明により、グラフとして表現される計算を実行するグラフ実行システム及びグラフ実行方法は、多くの場合、以下の一般に利用可能な設備を有する計算環境又は計算基板で使用される：通信チャネル、データストア、プロセス制御及びデータアクセス方法。そのために、グラフ実行システム及びグラフ実行方法について、そのような基準基板を参照して説明するが、システム及び方法はそのような基板に限定されない。

【０１７３】

通信チャネルについて、基準基板は、通信チャネルを作成及び破壊し、システム又は方法が分散環境で使用される場合、リモートプロセッサでそれを行う設備を提供する。そのような通信チャネルは、同じプロセッサ上の２つのプロセス間又は異なるプロセッサ上の２つのプロセス間でデータを伝送するように機能し得る。幾つかの種類の通信チャネルを提供し得る。データストアに関して、基準基板は、データストアを作成し、システム又は方法が分散環境で使用される場合、リモートプロセッサでそれを行う設備を提供する。データストアは、ジョブ間又は処理段階間でデータを保存するメモリである。基板は、プロセスがデータストアの内容を読み／書きできるようにするメカニズムも提供する。幾つかの場合、基板は、プロセスがリモートノードに配置されたデータストアの内容を読み／書きできるようにするメカニズムを提供し得るが、そのような設備は本発明により必要とされない。プロセス制御に関して、基板は、プロセスを開始し、実行を終了したときを特定し、プロセスが通常通り終了したか否かを判断する設備を提供する。基板は、システム又は方法が分散環境で使用される場合、リモートプロセッサでプロセスを開始する設備を提供する。プロセスを開始する動作は、実行するプログラムの名称及びアクセスされるあらゆるファイル又は通信チャネルの識別子を含め、パラメータを渡すことを可能にしなければならない。データアクセス方法に関して、基板は、基板で実行中のプロセスが使用し得るデータアクセス方法の設備を提供する。

【０１７４】

有用な基板は、様々なバージョンのいずれかのＵＮＩＸオペレーティングシステムにより提供される基板の拡張として開発し得る。ＵＮＩＸオペレーティングシステムは、２つのタイプの通信チャネルを提供する：名前付きパイプ及びＴＣＰ／ＩＰストリーム。名前付きパイプとは、同じノード上のプロセス間でデータを伝送するのに使用し得るポイントツーポイント通信チャネルである。（「ノード」という用語は、場合により複数のプロセッサ及び記憶デバイス又はデータストアを有するが、１つのプールの共有メモリを有するコンピュータを指す）。名前付きパイプは、ファイルシステム内のパスにより識別される。ＴＣＰ／ＩＰストリームとは、ＴＣＰ／ＩＰネットワーク（例えば、インターネット）の任意の場所にある２つのプロセス間でデータを伝送するのに使用し得るポイントツーポイント通信チャネルである。ＴＣＰ／ＩＰストリームを確立するには、２つのプロセスが接続を確立するプロトコルを使用する必要がある。多くのそのようなプロトコルは、使用

されているが、大半は、システムにより必要とされるピアツーピア接続ではなく、クライアントーサーバ接続の確立に向けられている。

【０１７５】

拡張基板は、共有メモリと呼ばれる１つの追加のタイプのチャネルを提供する。共有メモリは、既知のように、複数のプロセスによりアクセス可能であり、データの伝送に使用し得るメモリのプールである。共有メモリチャネルは、アクセスを同期するメカニズムを含む。共有メモリは、同じノード上のプロセス間のデータ伝送に適用可能であり、名前付きパイプに効率的な代替物である。

【０１７６】

本発明と併用するために、拡張基板のＴＣＰ／ＩＰ接続プロトコルは、ピアツーピアリンクに適するべきである。そのようなプロトコルでは、エンティティは「ストリーム識別子」を作成し、ストリーム識別子を受信するように所望のストリームの両エンドポイントを準備する。２つのエンドポイントは、ストリーム識別子を基板に与え、それに応答して、ＴＣＰ／ＩＰストリームへの接続を受信する。同様のメカニズムを使用して、共有メモリ通信チャネルを確立する。そのようなプロトコルの作成は、当技術分野で周知である。

【０１７７】

ＵＮＩＸオペレーティングシステムは、３つのデータアクセス方法を提供する：ファイルインタフェース、ストリームインタフェース、及びファイル記述子インタフェース。ファイルインタフェースを用いて、プログラムにファイルの名称を提供し得、次に、プログラムは、ファイルを開き、それから読み出し、それに書き込み、その中で検索し得る。ストリームインタフェースを用いて、プログラムには、ファイルの名称又は名前付きパイプを提供し得、次に、プログラムは、それを開き、それから読み出し、又はそれに書き込み得る。ストリームインタフェースは、シーク動作又はファイルのみに適用可能な他の動作の使用を許さない。ファイル記述子インタフェースを用いる場合、ファイル記述子（例えば、番号により識別される）は、プログラムが呼び出される前に、ファイル、名前付きパイプ、又はＴＣＰ／ＩＰストリームにバインドし得る。次に、プログラムは、ファイル記述子から読み出すか、又はファイル記述子に書き込み得る。これはストリームインタフェースと同様であるが、ストリームインタフェースを用いる場合、ストリームはプログラム外で開かれる。すなわち、プログラムが呼び出される前に開かれる。

【０１７８】

本発明と併用するために、拡張基板は、１つの追加のアクセス方法：ストリームオブジェクト接続（ＳＯＣ）を提供する。「ストリームオブジェクト」が、ファイルの名称若しくは名前付きパイプ又はＴＣＰ／ＩＰストリーム若しくは共有メモリチャネルの作成に使用される識別子であり得る「ストリームオブジェクト識別子」（一意の文字列）、「通信方法名」、及び「チャネル／ファイル識別子」を拡張基板に提示することにより作成される。呼び出し側は、ソース及び宛先が配置されるノードの識別子も提供しなければならない。

【０１７９】

ＵＮＩＸオペレーティングシステムは、１種のデータストア：ファイルを提供する。ファイルとは、通常、ディスクに記憶されるバイトシーケンスである。ファイルは、ファイルシステム中のパス及びホストを識別する識別子、すなわち、ファイルが存在するノードにより識別される。

【０１８０】

ＵＮＩＸオペレーティングシステムは、プロセス制御を提供する。プロセスを開始するために、以下の情報が提供される：実行するプログラムの名称、任意のコマンドライン引数、及びファイル記述子の任意のバインド。コマンドライン引数は、プログラムによりアクセスされるファイルの名称及び名前付きパイプを含み得る。ファイル記述子の各バインドは、ファイル記述子番号と、ファイル若しくは名前付きパイプを識別するパス又はＴＣＰ／ＩＰストリームを識別する補助チャネルのいずれかとからなる。更に、コマンドライン引数は、環境変数の値を含み得、環境変数は一般に、特定のリソースを配置し得る場所

を記述するため、又は実行オプションを構成するためにＵＮＩＸプログラムにより要求される。

【０１８１】

グラフの概説

本発明のシステムは、これよりそれぞれ説明するプロセス頂点の組、ファイル頂点の組、及びデータリンクの組の状態変数における計算の状態を追跡する。これらの構造を使用したシステムの動作について後に説明する。

【０１８２】

本発明の概念の幾つかを示すために、従来技術による例を使用する。図３４Ａは、従来技術の給与システムの簡略例を通るデータフローを示す。この例への入力は、全従業員の永久レコードを含む旧マスタファイル３４１０及び１週間分のタイムシートを含む更新ファイル３４１２である。旧マスタファイル３４１０は従業員ＩＤによりソートされる。グラフは以下のデータフローを示す。

（１）更新ファイル３４１２がソートされる（ステップ３４１４）。

（２）更新がチェックされ（ステップ３４１６）、不良レコードは不良レコードファイル３４１８に入れられる。

（３）ステップ３４１６からの更新及び旧マスタファイル３４１０が処理され（ステップ３４２０）、新しいマスタファイル３４２２及び給与３４２４の組を生成する。

【０１８３】

プロセス頂点

プロセス頂点は以下の情報を含む：

・プロセス頂点識別子、

・プログラムテンプレート、

・作業ディレクトリ（すなわち、スクラッチファイルを作成し得るディレクトリ）を識別する作業ディレクトリ識別子、及び

・作業ノード（すなわち、処理が行われるノード）を識別する作業ノード識別子、

・以下の値の１つを有する状態変数：ディセーブル、イネーブル、実行可能、非実行可能、又は完了（最初はディセーブル）。

・頂点がアダプタ頂点（後述）である場合を示すフラグ。

【０１８４】

以下の表は、図３４Ａに示されるソートプロセス、チェックプロセス、及び処理プロセス（プログラム）の３つのプロセス頂点の内容を示す。

【０１８５】

【表３】

識別子	プログラムテンプレート	作業ディレクトリ	作業ノード	状態	アダプタ頂点?
ソート	ソートテンプレート	/work/tmp	ノード1	ディセーブル	偽
チェック	チェックテンプレート	/work/tmp	ノード1	ディセーブル	偽
処理	処理テンプレート	/work/tmp	ノード1	ディセーブル	偽

【０１８６】

ファイル頂点

ファイル頂点は、ファイルに関連し、以下の情報を含む：

・ファイル頂点識別子、

・データノード識別子、

・データファイル識別子、
・ファイルにアクセスしようとするプログラムが使用し得る作業ノード（これは通常、データノードと同一である）を識別する作業ノード識別子、
・ファイルにアクセスしようとするプログラムがどこでスクラッチファイル（これは通常、データファイルと同じファイルシステム内にある）を作成し得るかを識別する作業ディレクトリ識別子、及び
・以下の値の１つを有する状態変数：完成、未完成（最初は未完成）。

【０１８７】
計算基板がファイル以外のデータストアを提供する場合、ファイル頂点内の情報は、データストアのタイプを示すインジケータ並びにその識別及び使用に必要な情報を含むように拡張し得る。

【０１８８】
以下の表は、図３４Ａのグラフに示されるファイルの５つのファイル頂点の内容を示す。

【０１８９】
【表４】

ファイル頂点 識別子	データ ノード	データファイル 識別子	作業 ノード	作業 ディレクトリ	状態
旧マスタ	ノード0	/data/master.old	ノード0	/work/tmp	不完全
更新	ノード2	/input/update	ノード2	/work/tmp	不完全
不良レコード	ノード1	/err/bad-recs	ノード1	/work/tmp	不完全
給与	ノード1	/output/paychk	ノード1	/work/tmp	不完全
新マスタ	ノード0	/data/master.new	ノード0	/work/tmp	不完全

【０１９０】
データリンク
データリンク（又は単に略して「リンク」）は以下の情報を含む：
・ソース頂点　－　ファイル頂点又はプロセス頂点のいずれか、
・ソースポート識別子　－　ソースがファイル頂点の場合、出力のみが許される、
・宛先頂点　－　ファイル頂点又はプロセス頂点のいずれか、
・宛先ポート識別子　－　宛先がファイル頂点である場合、入力のみが許される、
・非バインドであり得るか、又はファイル、名前付きパイプ、ＴＣＰ／ＩＰ、又は共有メモリ等の通信チャネルタイプの名前であり得る通信方法識別子（最初は非バインド）、及び
・非起動又は起動のいずれかであり得る状態変数（最初は非起動）。

【０１９１】
２つ以上のデータリンクを所与のファイル頂点又はプロセス頂点の所与のポートに取り付け得る。

【０１９２】
図３４Ａでは、接続矢印は、プロセスとファイル頂点との間のリンクを図で示す。リンクの概念には、本発明ではより具体的な形態が与えられる。図３４Ｂを参照すると、グラフ３４３０は、図３４Ａに示されるグラフ３４００から本発明により導出されるプロセス頂点、ファイル頂点、及びデータリンクを示す。

【０１９３】
以下の表は、グラフ３４３０に示される７つのデータリンク３４３２、３４３４、３４３６、３４３８、３４４０、３４４２、及び３４４４を表にしたものである。

【０１９４】

【表５】

参照番号	ソース頂点	ソースポートID	宛先頂点	宛先ポートID	通信方法	状態
132	更新	出力	ソート	入力	非バインド	非起動
134	ソート	出力	チェック	入力	非バインド	非起動
136	チェック	不良	不良レコード	入力	非バインド	非起動
138	チェック	良好	処理	更新	非バインド	非起動
140	旧マスタ	出力	処理	マスタ入力	非バインド	非起動
142	処理	マスタ出力	新マスタ	入力	非バインド	非起動
144	処理	給与	給与	入力	非バインド	非起動

【０１９５】

プログラムテンプレート

プロセス頂点はプログラムテンプレートを含む。プログラムテンプレートは、プログラムについての２つの基本的な種類の情報を含む：（１）プログラムの名称、コマンドライン引数、及び環境変数等のプログラムを呼び出すために必要な情報、並びに（２）プログラムがデータにアクセスする手段を記述するポート記述子のアレイ。呼び出し情報の厳密な形態は計算基板に依存する。説明したＵＮＩＸベースの基板の場合、プログラム名は、実行可能ファイルの名称に、スペースで区切られた一連の文字列からなるコマンドライン引数を加えたものである。これらの文字列の幾つかは、「$portname」の形態であり得、ここで、「portname」は、テンプレート内のポートの１つの名称であり、そのポートは、ファイルインタフェース又はストリームインタフェースデータアクセス方法を使用しなければならない。この場合、文字列「$portname」は、呼び出し時、そのポートに接続されたファイルの名称又は名前付きパイプで置換される。

【０１９６】

以下の表は、図３４Ｂのグラフ３４３０に示される３つのプログラム（プロセス）の例示的なプログラムテンプレートの内容を示す。

【０１９７】

【表６】

テンプレート名	プログラム	引数	ポートID	方向	方法
ソートテンプレート	/bin/sort	$Input EmpNo	入力 出力	入力 出力	ReqsFile ReqsFD
チェックテンプレート	/pay/check	$Input $Good $Bad	入力 良好 不良	入力 出力 出力	ReqsNamedPipe ReqsNamedPipe ReqsNamedPipe
プロシージャテンプレート	/pay/process	ヌル	マスタ入力 更新 マスタ出力 給与	入力 入力 出力 出力	ReqsSOC ReqsSOC ReqsSOC ReqsSOC

【０１９８】
プログラムを呼び出すために必要な情報は、２つの形態で提供し得る。第１に、この情報は上記表に示されるように、プログラムテンプレート内に明示的に記憶し得る。第２に、この情報は動的に生成し得、その場合、情報を生成するルーチンのアドレスは、プログラムテンプレートに記憶される。
【０１９９】
プログラムテンプレート内のポート記述子は、上記表中の一番右側の３つの要素として示される以下の情報を含む：
・ポートのポート識別子、
・ポートが入力に使用されるか、それとも出力に使用されるかについての指示、及び
・いずれの通信方法がポートで許容可能であるかを示す、ReqsFile、ReqsNmaedPipe、ReqsFD、又はReqsSOC（「Reqs」は「要求（Requires）」を表す）等の許容可能な方法のコード。
【０２００】
これらの許容可能な方法コードは、計算基板によりサポートされるデータアクセス方法を指す。
【０２０１】
ドライバプログラム
図３５を参照すると、ドライバプログラム３５００（又は単に略して「ドライバ」）は、ユーザインタフェース３５０４を通してユーザ３５０２から受信される入力に基づいて、グラフを示す手段を提供する。特定のグラフを表す１つ又は複数のグラフデータ構造３５０６（例えば、図３４Ｂに示される等）は、ドライバ３５００により生成される。次に、ドライバ３５００のグラフ実行制御機能３５０８は、ドライバ３５００に以下の外部制御３５１０機能を任意の順序で、外部プロセス３５１２と対話するために必要な頻度実行できるようにする：
・プロセス頂点の作成、
・ファイル頂点の作成、
・任意の対の頂点（又はいずれかの種類）間のデータリンクの作成、
・ディセーブルからイネーブルへのプロセスの現在状態の変更、及び
・グラフを実行させること。
【０２０２】
ドライバ３５００は最終的に、ユーザ３５０２により行われる要求に応答して、ユーザインタフェース３５０４を通してユーザ３５０２により提供される情報を使用して、これらの動作を実行する。ユーザ３５０２は、グラフィカルユーザインタフェースを通して直接、ドライバ３５００に入力を提供している人物であり得る。代替的には、ユーザ３５０２は、例えば、オブジェクト指向又は手続き型インタフェースを通してドライバ３５００を制御する別個のプログラムであり得る。このようにして、ユーザ３５０２は、ドライバ３５００を使用して、グラフを構築しグラフの実行を制御し得る。
【０２０３】
ドライバ３５００は、ユーザ３５０２が識別子、プログラムテンプレート、作業ノード、及び作業ディレクトリを提供した場合、グラフのプロセス頂点を作成し得る。作業ディレクトリには、作業ノードに基づいてデフォルト値を与え得る。
【０２０４】
ドライバ３５００は、ユーザ３５０２が識別子、データノード、データファイル名、作業ノード、及び作業ファイル名を提供した場合、グラフのファイル頂点を作成し得る。作業ノードのデフォルトは、好ましい実施形態では、データノードに設定される。作業ファイル名には、作業ノード及び／又はデータファイル名に基づいてデフォルト値を与え得る。
【０２０５】
ドライバ３５００は、ユーザ３５０２がソース頂点、宛先頂点、ソースポート識別子、

及び宛先ポート識別子を提供した場合、グラフの任意の頂点対間にデータリンクを作成し得る。好ましい実施形態では、ソースポート識別子のデフォルトは出力に設定され、宛先ポート識別子のデフォルトは入力に設定される。

【０２０６】

ユーザ３５０２がプロセス頂点のイネーブル／ディセーブル状態を制御できるようにすることにより、ドライバ３５００は、ユーザ３５０２が、プロセス頂点のサブセットを選択的にイネーブルすることにより、後述する処理の実行順序を制御できるようにする。

【０２０７】

グラフの実行

図３６を参照すると、初期グラフが生成された後、ドライバ３５００は、グラフの実行と、したがってグラフにより示されるプロセスの実行とを制御する。ドライバ３５００は、グラフを実行する場合、以下の一般フェーズＡ～Ｉを実行することによりそれを行う。

Ａ．プロセス頂点のいずれか１つがイネーブル状態である限り、ドライバ３５００は以下のステップＢ～Ｉを繰り返す。ドライバ３５００はフェーズＣ、Ｄ、及びＩを省略することがあり、ステップＢ、Ｃ、Ｅ、及びＨで実行される動作を混ぜ得る。

Ｂ．ドライバ３５００は、実行に向けてグラフを準備する。このフェーズでは、ドライバ３５００は、説明するように、実行可能なプロセス頂点を識別し、リンクの通信方法を選び、アダプタノードを生成し得る。

Ｃ．ドライバ３５００は、より詳細に後述するように、データリンクを起動する。このフェーズでは、ドライバ３５００は、説明するように、通信方法の実施に必要な特定の計算構造を作成する。

Ｄ．ドライバ３５００は、計算基板により必要とされる任意の他のデータ構造又はファイルを作成する。上述した拡張基板の場合、ドライバ３５００は、説明するように、リンクファイルを作成する。これにより、プログラムは、実行時にグラフ接続情報にアクセスすることができる。

Ｅ．ドライバ３５００は、説明するように、プロセスを起動する。

Ｆ．ドライバ３５００は、プロセスの終了を待つ。このフェーズは、全てのプロセスが正常終了するときに、又は任意のプロセスが異常終了するときに完了する。

Ｇ．任意のプロセスが異常終了する場合、グラフの実行は中止される。

Ｈ．その他の場合、実行可能状態にある全てのプロセス頂点は完了状態に遷移する。いずれのプロセス頂点も実行可能状態になかった場合、クリーンアップフェーズＩが実行され、制御は、実行が停止された指示と共に呼び出し側（例えば、ドライバ３５００のユーザ３５０２）に戻る。

Ｉ．ドライバ３５００は、説明するように、データリンク及びリンクファイルをクリーンアップする。このクリーンアップは、フェーズＣ及びＤにおいて作成されたデータ構造の幾つかをクリーンアップする。

【０２０８】

特定のフェーズの更なる詳細について以下に説明する。

【０２０９】

フェーズＢ：実行に向けてグラフを準備する

図３７Ａを参照すると、ドライバ３５００は、ユーザ２０２により最初に示されたグラフにアクセスし、グラフ変換を適用することにより実行に向けてグラフを準備する（ステップ３７００）。これらの変換を実行するに当たり、初期グラフを定義するグラフデータ構造は、既知のようにトラバースされて、各頂点及び関連付けられたあらゆるリンクをフェッチする。好ましい実施形態では、実行に向けてグラフを準備するために、５つのグラフ変換が、フェッチされたデータ構造に対して使用される。

【０２１０】

グラフは、依然として実行形態にない間（ステップ３７０２）、後述する５つのグラフ変換を選択し、任意の順序で、実行可能グラフが得られるまで（ステップ３７１６）、必要とされる頻度で（一切なしを含め）適用し得る（ステップ３７０４）。５つのグラフ変

換は、（１）ファイルアダプタの挿入（ステップ３７０６）、（２）通信アダプタの挿入（ステップ３７０８）、（３）ファイル頂点の状態の完了への設定（ステップ３７１０）、（４）プロセス頂点の状態の実行可能又は非実行可能への設定（ステップ３７１２）、及び（５）データリンクの通信方法の設定（ステップ３７１４）である。これらの各変換及び各変換を実行し得る条件について、これより説明する。

【０２１１】

ファイルアダプタの挿入：この変換では、ドライバ３５００は、リンクをファイルアダプタ（すなわち、リンク、ファイル頂点、及び別のリンク）で置換する。すなわち、リンクを表す各グラフデータ構造が、グラフデータ構造３５０６（図３５）のトラバース中、フェッチ又はアクセスされる際、元のデータ構造を変更、拡張、又は置換する新しいデータ構造を作成し得る。

【０２１２】

ソース（宛先）ファイルアダプタの場合、ファイル頂点のホストはソース（宛先）頂点のホストと同じであり、ファイル頂点のファイルは、ソース（宛先）頂点の作業ディレクトリに配置される新しいファイルである。この変換は、

（１）ソースが、ファイル頂点又は完了状態ではないプロセス頂点のいずれかである場合、及び

（２）宛先が、不完全状態のファイル頂点又は完了状態ではないプロセス頂点のいずれかである場合

のみ実行し得る。

【０２１３】

通信アダプタの挿入：この変換では、ドライバ３５００は、リンクを通信アダプタ（すなわち、リンク、プロセス頂点、及び別のリンク）で置換する。プロセス頂点はコピープログラムを実行し、コピープログラムは、データを入力から出力にコピーし、土台となる基板によりサポートされる通信チャネル又はデータストアのいずれかの読み出し／書き込みを行うことができる。ソース（宛先）通信アダプタの場合、プロセス頂点のホストは、ソース（宛先）頂点のホストと同じであり、作業ディレクトリはソース（宛先）頂点の作業ディレクトリと同じである。プロセス頂点は、イネーブル状態で作成される。この変換は、

（１）ソースが、完了以外の状態のプロセス頂点又はファイル頂点のいずれかである場合、及び

（２）宛先が、完了以外の状態のプロセス頂点又は不完全状態のファイル頂点のいずれかである場合

のみ実行し得る。

【０２１４】

ファイル頂点の状態の完全への設定：この変換では、ファイル頂点の状態は完全に設定される。この変換は、ファイル頂点の状態が不完全であり、ファイル頂点への全ての入力が、完了状態のプロセス頂点である場合のみ、実行し得る。

【０２１５】

プロセス頂点の状態の実行可能又は非実行可能への設定：この変換では、プロセス頂点の状態は実行可能又は非実行可能のいずれかに設定される。この変換は、プロセス頂点の状態がイネーブルである場合のみ、実行し得る。

【０２１６】

データリンクの通信方法の設定：この変換では、データリンクの通信方法が設定される。この変換は、データリンクの通信方法が非バインドである場合のみ、実行し得る。

【０２１７】

以下の３つの属性を有するグラフは実行可能である。

（１）全てのプロセス頂点は、以下の状態の１つである：完了、実行可能、非実行可能、又はディセーブル。

（２）全てのデータリンクは以下の基準の全てを満たす。

１）データリンクのソース又は宛先のいずれかが実行可能プロセス頂点である場合、データリンクの通信方法は、特定の通信方法にバインドされなければならない。
２）データリンクの通信方法が、ファイル以外の何かである場合、そのソース及び宛先は両方とも、プロセス頂点でなければならず、１つのプロセス頂点が実行可能である場合、両プロセス頂点は実行可能でなければならない。
３）データリンクの通信方法がファイルである場合、そのソース及び宛先はファイル頂点でなければならない。宛先が実行可能プロセス頂点である場合、ソースは完全ファイル頂点でなければならない。ソースが実行可能ファイル頂点である場合、宛先は不完全ファイル頂点でなければならない。
（３）通信方法にバインドされた全てのリンクは、通信方法に固有の制約を満たす。
１）通信方法は、そのソースポート及び宛先ポートのアクセス方法と互換性を有さなければならない（これは、プログラムテンプレートを調べることにより判断し得る）。説明した拡張基板の場合、全ての通信方法はＳＯＣアクセスと互換性を有し、共有メモリ以外の全てはファイル記述子アクセスと互換性を有し、名前付きパイプ及びファイルは、名前付きパイプアクセスと互換性を有し、ファイルのみがファイルアクセスと互換性を有する。
２）幾つかの通信方法では、ソース頂点及び宛先頂点のノードが同一である必要がある。説明した拡張基板の場合、これは、ＴＣＰ／ＩＰ以外の全ての通信方法で該当する。
【０２１８】
グラフ変換は、実行可能なグラフが得られるまで、任意の順序で適用し得る（例えば、グラフデータ構造は、全ての変換が完成するまで、繰り返しトラバースし得る）。図３７Ｂを参照すると、グラフ変換は、一実施形態では、以下の順序でとられる以下のステップで適用される：（１）ファイルアダプタを挿入し（ステップ３７５０）、（２）ファイルーファイルリンクを置換し（ステップ３７５２）、（３）完全なファイル頂点を識別し（ステップ３７５４）、（４）非実行可能プロセス頂点を識別し（ステップ３７５６）、（５）実行可能プロセス頂点を識別し（ステップ３７５８）、（６）残りのイネーブル頂点を非実行可能に設定し（ステップ３７６０）、（７）条件が満たされるより多くのファイルアダプタを挿入し（ステップ３７６２）、（８）通信方法を選び（ステップ３７６４）、（９）通信アダプタを挿入する（ステップ３７６６）。この実施形態のステップについて、より詳細にこれより説明する。
【０２１９】
（１）ファイルアダプタを挿入する。図３８Ａを参照すると、ファイルアダプタを挿入するために、以下のステップがグラフ内の全てのリンクに対して実行される（ステップ３８００）。リンクのソースポートが、ファイルの使用を必要とするデータアクセス方法を有し（ステップ３８０２）、宛先が同じノード上のファイルではない（ステップ３８０４）場合、ソースファイルアダプタを挿入する（ステップ３８０６）。リンクの宛先ポートが、ファイルの使用を必要とするデータアクセス方法を有し（ステップ３８０８）、ソースが同じノード上のファイルではない（ステップ３８１０）場合、宛先ファイルアダプタを挿入する（ステップ３８１２）。リンクの宛先が、ディセーブル状態のプロセス頂点であり（ステップ３８１４）、ソースがイネーブル状態のプロセス頂点である（ステップ３８１６）場合、宛先ファイルアダプタを挿入する（ステップ３８１２）。
【０２２０】
（２）ファイルーファイルリンクを置換する。図３８Ｂを参照すると、ファイルーファイルリンクを置換するために、以下のステップが、グラフ中の全てのリンクに対して実行される（ステップ３８２０）。ソース及び宛先が両方ともファイル頂点である場合（ステップ３８２２）、ソース通信アダプタを挿入する（ステップ３８２４）。（更に、ソース及び宛先が異なるノードにある場合にも、宛先通信アダプタを挿入する：図示せず）。
【０２２１】
（３）完全なファイル頂点を識別する。図３８Ｃを参照すると、完全なファイル頂点を識別するために、以下のステップが、グラフ中の全てのファイル頂点に対して実行される

（ステップ３８３０）。全ての上流頂点が完了状態のプロセス頂点である場合（ステップ３８３２）、その状態を完全に設定する（ステップ３８３４）。

【０２２２】

（４）非実行可能なプロセス頂点を識別する。図５Ｄを参照すると、非実行可能なプロセス頂点を識別するために、以下のステップが、グラフ中の全てのリンクに対して実行される（ステップ３８４０）。「非実行可能性」テストが以下のように実行される（ステップ３８４２）。リンクのソースが不完全ファイル頂点であり、宛先がイネーブル状態のプロセス頂点である場合、プロセス頂点の状態を非実行可能に設定し（ステップ３８４４）、ソースが、イネーブル以外の任意の状態であり、宛先がイネーブル状態のプロセス頂点である場合、宛先プロセス頂点は非実行可能と記される（ステップ３８４４）。非実行可能と記され得る頂点がなくなるまで、このテストを繰り返す。

【０２２３】

（５）実行可能なプロセス頂点を識別する。図５Ｅを参照すると、実行可能なプロセス頂点を識別するために、以下のステップが、グラフ中の全てのプロセス頂点に対して実行される（ステップ３８５０）。「実行可能性」テストが以下のように実行される（ステップ３８５２）。頂点がイネーブル状態であり、全ての上流頂点が完全ファイル頂点又は実行可能プロセス頂点である場合、頂点の状態を実行可能に設定する（ステップ３８５４）。実行可能と記され得る頂点がなくなるまで、このテストを繰り返す。

【０２２４】

（６）残りのイネーブル頂点を非実行可能に設定する。図５Ｆを参照すると、残りのイネーブル頂点を非実行可能に設定するために、以下のステップがグラフ中の全てのプロセス頂点に対して実行される（ステップ３８６０）。頂点がイネーブル状態である場合（ステップ３８６２）、その状態を非実行可能に設定する（ステップ３８６４）。

【０２２５】

（７）より多くのファイルアダプタを挿入する。図５Ｇを参照すると、より多くのファイルアダプタを挿入するために、以下のステップがグラフ中の全てのリンクに対して実行される（ステップ３８７０）。リンクのソースが実行可能なプロセス頂点であり（ステップ３８７２）、宛先が非実行可能なプロセス頂点である（ステップ３８７４）場合、ソースファイルアダプタを挿入する（ステップ３８７６）。

【０２２６】

（８）通信方法を選ぶ。図５Ｈを参照すると、通信方法を選ぶために、以下のステップがグラフ中の全てのリンクに対して実行される（ステップ３８８０）。このステップは、実行可能プロセスにいずれかの端部において取り付けられ、通信方法にバインドされていないリンクに対してのみ適用される。リンクのソース（宛先）がファイル頂点であり（ステップ３８８１）、その宛先（ソース）が同じノード上のプロセス頂点である場合、リンクの通信方法をファイルに設定する（ステップ３８８２）。その他の場合、通信方法の全ての制約が満たされるように、利用可能な通信方法の１つを選ぶ（ステップ３８８３～３８８５）。スピードのために、通信方法は共有メモリ、名前付きパイプ、及びＴＣＰ／ＩＰの順序で検討し得る。上述した制約を満たす最初の方法が選択される（ステップ３８８６）。基準基板では、以下のルールを使用し得る。第１に、リンクが、ＳＯＣ接続を許容するポートに取り付けられる場合、ソース及び宛先が同じノードにあるとき、共有メモリを使用し、又は異なるノードにあるとき、ＴＣＰ／ＩＰを使用する。その他の場合、ソース及び宛先が同じノードにあるとき、名前付きパイプ方法が使用される。他の全ての場合、単一の通信方法では不十分であり、システムは通信アダプタ（以下）に戻る。

【０２２７】

（９）通信アダプタを挿入する。単一の通信方法が、通信方法を選ぶ先行ステップにおいて選択されず、全てが試みられた場合（ステップ３８８３）、ソース通信アダプタを挿入し、アダプタの２つのリンクに通信方法を選ぶことを試みる（ステップ３８８７）ことに続く。これが失敗する場合（ステップ３８８８）、新しく挿入されたソース通信アダプタを宛先通信アダプタで置換することを試みる（ステップ３８８９）。これが失敗する場

合（ステップ３８９０）、ソース及び宛先通信アダプタの両方を挿入し、その結果生成される２つのアダプタ内の３つのリンクに通信方法を選ぶ（ステップ３８９１）。基準基板では、通信アダプタは、ソース及び宛先が異なるノードにあり、リンクが、ファイル頂点又はＳＯＣ接続方法を許容しないプロセス頂点のいずれかに接続される場合のみ、必要とされる。この場合、アダプタは以下のように選び得る。

・ソースがファイル頂点である場合、ソース通信アダプタを挿入する。次に、ソース通信アダプタ内の２つのリンクは、ファイル及びＴＣＰ／ＩＰ通信方法を使用する。

・ソースが、ＳＯＣ通信方法を許容していないポートである場合、ソース通信アダプタを挿入する。次に、ソース通信アダプタ内の２つのリンクはＴＣＰ／ＩＰ及びファイル通信方法を使用する。

・宛先がファイル頂点である場合、宛先通信アダプタを挿入する。次に、アダプタ内の２つのリンクはＴＣＰ／ＩＰ及びファイル通信方法を使用する。

・宛先が、ＳＯＣ通信方法を許容しないポートである場合、宛先通信アダプタを挿入する。次に、アダプタ内の２つのリンクは、ＴＣＰ／ＩＰ及び名前付きパイプ通信方法を使用する。

【０２２８】

フェーズＣ：データリンクの起動

図３９を参照すると、データリンクは、非起動状態で作成され、起動されなければならない。リンクを起動するために、リンクをスキャンして（ステップ３９００）、非起動であり（ステップ３９０２）、通信方法にバインドされ（ステップ３９０４）、実行可能なソース又は宛先を有する（ステップ３９０６）リンクを見つける。そのような全てのリンクで、様々な通信方法により使用し得る識別子が生成される（ステップ３９０８）。上述した拡張基板では、識別子は以下のように作成される。全てのリンクは２つの識別子を有する：ストリームオブジェクト識別子及び通信チャネル／ファイル識別子。ストリームオブジェクト識別子は、ＳＯＣメカニズムにより使用され、リンクの名称と同一である。チャネル／ファイル識別子は、リンクにより利用されるファイル、名前付きパイプ、共有メモリ領域、又はＴＣＰ／ＩＰ接続の識別に使用される。更に、プロセス頂点が名前付きパイプ又はファイル通信方法を必要とする場合、チャネル／ファイル識別子は利用可能になり、それにより、プロセス頂点は、起動されると（以下参照）、ＵＮＩＸファイルシステムを使用してチャネル／ファイルに取り付けることが可能になる。

【０２２９】

識別子が生成された後、基板が呼び出されて、チャネル又はストリームオブジェクトを作成する（ステップ３９１０）。通信方法が名前付きパイプである場合も、基板が呼び出されて、名前付きパイプを作成する。

【０２３０】

フェーズＤ：リンクファイルの作成

拡張基板は、各ノードで、そのノードのソース又は宛先のいずれかを有するリンクを列挙する「リンクファイル」を保持する。プログラムは、実行時にこのリンクファイルを調べて、いずれのリンクにアクセスしなければならないかを特定し得る。これは一般に、ＳＯＣインタフェースを使用するプログラムに対して行われる。したがって、拡張基板の場合、システムはリンクファイルを作成しなければならない。これは以下のように行われる：計算に関わるあらゆるノードについて、ドライバ３５００は、そのノードに割り当てられた実行可能なプロセス頂点を識別し、そのような頂点に取り付けられたあらゆるリンクについて、リンクファイルに以下の情報を記憶する：

・頂点の識別子、

・リンクが取り付けられるポートの名称、

・通信チャネルの識別子、及び

・該当する場合、データの転送に使用されるファイル又は名前付きパイプの識別子。

【０２３１】

フェーズＥ：プロセスの起動

図４０を参照すると、プロセスは、以下のステップを実行可能状態の全てのプロセス頂点に対して実行することにより起動される（ステップ４０００）。まず、頂点のプログラムテンプレートを使用して、呼び出し情報を生成する（ステップ４００２）。この情報は、

・呼び出すプログラムの名称、
・コマンドライン引数（コマンドライン引数は、通信チャネルの識別子及び頂点に取り付けられたリンクに関連付けられたファイルを含み得る）、
・任意選択的に、様々な環境変数の値、及び
・任意選択的に、プログラムの予期される「終了コード」

を含む。

【０２３２】

呼び出し情報は、少なくとも２つの方法の１つで生成し得る：ドライバ３５００がそのような情報をプログラムテンプレートに予め記憶し得るか、又はドライバ３５００が、そのような情報を動的に計算するルーチンを含み得、それらのルーチンのアドレスをプログラムテンプレートに記憶し得る。

【０２３３】

次に、頂点の識別子及び頂点のノードのリンクファイルの識別子は、環境変数セットに追加される（ステップ４００４）。次に、示されたノードで実行中のエージェントが、「プログラム呼び出しプロセス」を作成する（ステップ４００６）。プログラムのテンプレートが、入力ポート又は出力ポートをＵＮＩＸファイル記述子にバインドされることを必要とする場合、プログラム呼び出しプロセスに、入力ポート又は出力ポートに関連付けられたファイルの名称、名前付きパイプ、又はＴＣＰ／ＩＰストリーム識別子が提供され、プログラム呼び出しプロセスは、示されたファイル記述子を使用してファイル又は名前付きパイプを開く。プログラム呼び出しプロセスは、必要とされる環境変数をセットアップし、示されたコマンドライン引数を使用して、示されたプログラムを実行する（ステップ４００８）。

【０２３４】

フェーズＦ：待機

全てのプロセスが起動すると、システムは、好ましくはプログラムの起動に使用されたエージェントと同じエージェントを使用して、プロセスの実行を監視する。周期的に、システム（エージェント）は、プロセスが終了したことに気付く。これが生じると、システム（エージェント）は、プロセスが「正常」に終了したか、それとも「異常」終了したかを判断する。ＵＮＩＸの場合、これは終了コードを介して行われる。終了コードは、プログラムが、プログラムエラー、算術例外、無効メモリアクセス等に起因して中止したことを示し得る。そのような事例は常に「異常終了」として解釈される。代替的には、プログラムは、被制御方式で終了し、「イグジットコード」を返し得る（イグジットコードは、終了コードのサブセットを含む）。規約により、イグジットコード０はプログラムが正常終了したことを示し、他の全てのコードは異常終了を示す。上述したように、プログラムテンプレートは、この解釈を変更し得、例えば、全てのイグジットコードが「正常」終了として解釈されるべきであることを宣言し得る。

【０２３５】

システムは、プロセスが正常終了したと判断すると直ちに、任意選択的に、「デバッグ」ルーチンに入り、ユーザが異常終了の理由を診断できるようにし得る。デバッグが完了する（又はスキップされる）と、システムは、例えば、なお実行中の全てのプロセスのキル、部分的に書き込まれたファイルの削除等の中止手順を開始する。次に、ドライバプログラムは終了する。

【０２３６】

システムは、プロセスが正常終了したと判断する場合、このことに留意し、より多くのプロセスが終了するのを待つ。全てのプロセスが正常終了した場合、システムはクリーンアップフェーズＩに進む。

【０２３７】
フェーズＩ：クリーンアップ
全ての実行可能プロセスの実行が終わった後、ドライバ３５００は以下のステップを実行する。第１に、各ノードのリンクファイルは削除される。第２に、起動状態の全てのリンクはスキャンされる。リンクに取り付けられた全てのプロセス頂点が完了状態である場合、基板を使用して、リンクに関連付けられたあらゆる通信チャネルを破壊する。拡張基板の場合、これは、リンクのストリームオブジェクト識別子を取得し、示されたストリームオブジェクトを破壊するように基板に命令することにより行われる。通信方法が名前付きパイプである場合、これはまた、名前付きパイプも削除させる。更に、ファイルアダプタ頂点が完全であり、下流の全てのプロセスが完了である場合、そのファイルは削除される。
【０２３８】
アダプタの挿入
アダプタがリンクの場所に挿入される幾つかの状況について言及した。アダプタは、データリンク、又はファイル、又はプロセス頂点（アダプタ頂点）、及び一連の通信方法を使用して通信リンクを統合するために挿入される別のデータリンクである。アダプタ頂点がプロセスであるアダプタは、「通信アダプタ」と呼ばれる。アダプタ頂点がファイルであるアダプタは、「ファイルアダプタ」と呼ばれる。一般に、通信アダプタは、通信方法の組合せ（例えば、名前付きパイプ及びＴＣＰ／ＩＰ接続）が、リンクのソース及び宛先により課される制約を満たすために必要な場合、追加される。ファイルアダプタは、リンクのソース及び宛先が同時に実行されない（例えば、ソースは実行可能であるが、宛先は非実行可能又はディセーブルである）場合、又はリンクのソース又は宛先がファイルにのみ取り付け可能である場合、追加される。通信アダプタの場合、アダプタ頂点は、その入力をその出力にコピーし、任意の種類の通信チャネルに取り付け得るプログラムを指定する。アダプタリンクは続けて、ステップ３８８３～３８８５「通信方法を選ぶ」において記載される制約を受けて、任意の通信方法にバインドし得る。アダプタ頂点は単に、データをその入力からその出力にコピーする。アダプタリンクは、ソース及び宛先制約を受けて、任意の好都合な通信方法を有し得る。
【０２３９】
図４１を参照すると、ソースアダプタを挿入するために（ステップ４１００）、新しいアダプタ頂点（ステップ４１０２）及び新しいアダプタリンクが作成される（ステップ４１０４）。ソース通信アダプタの場合、アダプタ頂点は、以下の特徴を有するプロセス頂点である。
・プログラムテンプレートは、コピープログラム、すなわち、全ての入力データをその出力にコピーするプログラムを指定する。
・プログラムテンプレートは、任意の通信方法に取り付け可能な入力ポート記述子及び出力ポート記述子を指定する。
・新しい頂点がアダプタ頂点としてフラグ付けられる。
・元のソース頂点の作業ディレクトリ及び作業ノードは、アダプタ頂点の作業ディレクトリ及びノードとして使用される。
【０２４０】
ソースファイルアダプタの場合、アダプタ頂点は、以下の特徴を有するファイル頂点である。
・ファイルは、ソースプログラムにより使用されるノードに配置される。
・ファイルは、ソースプログラムの作業ディレクトリに配置される。ファイルは「アダプタ」としてマークされ、それにより、宛先プロセスにより消費された場合、削除し得る。
【０２４１】
新しいアダプタリンク（ステップ４１０４）は以下の特徴を有する。
・アダプタリンクのソース頂点は、元のリンクのソース頂点と同じであり、アダプタリ

ンクのソースポート名は、元のリンクのソースポート名と同じである。
・アダプタリンクの宛先頂点はアダプタ頂点であり、アダプタリンクの宛先ポート名は入力である。
・アダプタリンクの通信方法は、ソースアダプタを挿入する手順により指定される値に設定される（これは多くの場合、非バインドである）。
【０２４２】
最後に、元のデータリンクのソースは、新しいアダプタ頂点に設定され（ステップ４１０６）、ソースポート名は出力に設定される。
【０２４３】
宛先アダプタ頂点を挿入する手順は対称であり、「ソース」及び「宛先」並びに「入力」及び「出力」はそれぞれ相互交換される。
【０２４４】
図４９Ａ及び図４９Ｂは、頂点Ｖ１のポートＰ１と頂点Ｖ２のポートＰ２との間のリンクＬの場所への通信アダプタ４２００の挿入を示す。図４２Ｂでは、リンクＬは、第１のリンク４２０２、頂点９０４、及び第２のリンク４２０６を含むソースアダプタ４２００で置換されている。アダプタ４２００がファイルアダプタである場合、頂点４２０４は、ホストＨ１上のディレクトリＤ１内のファイルのファイル頂点であり、これらは頂点Ｖ１のディレクトリ及びホストである。アダプタ４２００が通信アダプタである場合、頂点４２０４はコピープログラムを実行中のプロセス頂点である。第２のリンク４２０２は、ソース頂点がアダプタ頂点４２０４に設定され、ソースポート名が出力に設定された元のリンクＬである。第１のリンク４２０２のソースポート名は、元のリンクＬの元のソースポート名Ｐ１に設定され、その宛先頂点は、新たに追加された頂点４２０４の入力ポートである。
【０２４５】
新しいアダプタ４２００が宛先アダプタであった場合、頂点４２０４は、ホストＨ１及びソース頂点Ｖ１のディレクトリＤ１ではなく、ホストＨ２及び宛先頂点Ｖ２のディレクトリＤ２を使用する。
【０２４６】
グラフの入力及び実行の例
これより、図３４に記載される給与プログラムに適用される本発明を考慮する。まず、アプリケーションの作成前に、ユーザは、必要とされる全てのプログラムのテンプレート、特にソートプログラム、データチェッカープログラム、及び給与プログラムのテンプレートをシステムに提供している。次に、ユーザは、上述したように、アプリケーションをグラフとして表現する。その結果生成されるグラフを図４３に示し、示されるように、ノード０、１、及び２で実行中の頂点４３００～４３７０を有する（様々なリンクが取り付けられるポートの名称は省略されるが、図３４Ｂとの比較から明らかであるはずである）。
【０２４７】
このグラフを処理するに当たり最初のステップは、実行に向けてグラフを準備することである。まず、ステップ３７５０を実行中、ソートプログラム４３１０のテンプレートを調べ、ソートプログラムのテンプレートが入力としてファイルを必要とすることを観察する。したがって、ステップ３７５０は、宛先ファイルアダプタ４３０２（図４４）を挿入する。
【０２４８】
次に、ステップ３７５２を実行し、ノード２上の更新ファイル４３００からノード１上の一時ファイル４３０２へのリンクを有することに留意する。したがって、ソース通信アダプタ及び宛先通信アダプタ（コピー４３０４、コピー４３０６）の両方が、このリンク（図４５）に挿入される。
【０２４９】
次に、ステップ３７５４において、完全なファイル頂点（更新４３００及び旧マスタ４

３５０）を識別する。これが行われると、ステップ３７５６において、不完全なファイル上流を有するため、非実行可能なプロセス頂点を探す。ソートプログラム４３１０が、この基準を満たすことが見出される。ステップ３７５８において、実行可能なプロセス頂点のみ及び／又は完全なファイル上流のみを有するため、実行可能なプロセス頂点を識別する。２つのコピープログラム４３０４、４３０６（すなわち、ステップ３７５２において挿入される通信アダプタ）はこの基準を満たし、実行可能と記される。残りの全てのノードは非実行可能と記される。最後に、実行可能なプロセス頂点に取り付けられるリンクの通信方法が選ばれる。ファイルに取り付けられたリンクにはファイル方法が選ばれ、２つのコピープログラム４３０４、４３０６（異なるノード上にある）間のリンクにはＴＣＰ／ＩＰが選ばれる。これは、図４６に示される状況を残す（実行可能／完全として記されない頂点は非実行可能／不完全であり、通信方法が記されていないリンクは非バインドである）。

【０２５０】

この時点で、グラフは実行可能である。リンクファイルが作成され（図示せず）、示されたリンクは起動し、実行可能なプロセス頂点は起動する。全てのプロセスが終了すると、システムは、示されたリンクを非起動し、プロセス頂点の状態を「実行可能」から「完了」に変更することにより「クリーンアップ」する。この結果、図４７に示される状況が生成される。

【０２５１】

この時点で、システムは、全てのプロセスが完了したわけではないと判断し、したがって、新たな実行ラウンドを開始する。まず、システムは、前と同じようにグラフを準備する。システムは、一時ファイル４３０２が完全であることに留意することにより開始する。次に、システムは、非実行可能なプロセス頂点がないと判断する。最後に、システムは、実際、全てのプロセス頂点が実行可能であると判断する。この結果、図４８に示される状況が生成される。

【０２５２】

この時点で、通信方法を選択する準備ができる（ステップ３７６４）。第１に、一時ファイル４３０２、不良４３３０、及び給与４３６０に取り付けられたリンクは、ファイル及び実行可能プロセス頂点に接続し、ノード境界を越えないため、ファイルに設定される。第２に、チェック４３２０に取り付けられた残りのリンクは、チェック４３２０が名前付きパイプを必要とし、これらのリンクがノード境界を越えないため、名前付きパイプに設定される。

【０２５３】

これは、異なるノードで実行中のプロセスにファイルを接続する、旧マスタ４３５０、新マスタ４３７０、及びプロセス４３４０間の接続を残す。両事例で通信アダプタが必要であり、通信アダプタは、ステップ３７６６においてコピー４３４２及びコピー４３４４として挿入される。両アダプタはノード１で実行される。通信方法が選ばれると（ファイルに接続するリンクにはファイル、ノード境界を越えるリンクにはＴＣＰ／ＩＰ）、図４９に示されるグラフになる。

【０２５４】

再び、グラフは実行可能な状態になる。システムは、リンクファイルを作成し、リンク及びプロセスを起動し、待機し、クリーンアップする。この時点で、全てのプロセス頂点は「完了」され、したがって、グラフの実行は終了する。

【０２５５】

プログラム実施

本発明は、ハードウェア、ソフトウェア、又は両方の組合せで実施し得る。しかし、好ましくは、本発明は、プロセッサ、データ記憶システム（揮発性及び不揮発性メモリ及び／又は記憶要素を含む）、少なくとも１つの入力デバイス、及び少なくとも１つの出力デバイスをそれぞれ含むプログラマブルコンピュータで実行されるコンピュータプログラムで実施される。プログラムコードは、入力データに適用されて、本明細書に記載の機能を

実行し、出力情報を生成する。出力情報は、既知の方式で１つ又は複数の出力デバイスに適用される。

【０２５６】

各プログラムは、好ましくは、高水準手続き型又はオブジェクト指向型プログラミング言語で実施されて、コンピュータシステムと通信する。しかし、プログラムは、必要に応じて、アセンブリ又は機械語で実施することができる。いずれの場合でも、言語はコンパイル型又はインタープリタ型言語であり得る。

【０２５７】

そのような各コンピュータプログラムは、好ましくは、汎用又は専用プログラマブルコンピュータにより読み取り可能な記憶媒体又はデバイス（例えば、ＲＯＭ又は磁気ディスク）に記憶されて、記憶媒体又はデバイスがコンピュータにより読み取られると、本明細書に記載の手順を実行するようにコンピュータを構成し動作させる。本発明のシステムは、コンピュータプログラムが構成されたコンピュータ可読記憶媒体として実施されるものとして見なすこともでき、ここで、記憶媒体は、本明細書に記載の機能を実行するように特定の予め定義される方式でコンピュータを動作させるように構成される。

【０２５８】

本発明の幾つかの実施形態について説明した。それにもかかわらず、本発明の趣旨及び範囲から逸脱せずに、様々な変更形態をなし得ることが理解される。したがって、本発明が特定の示される実施形態により限定されず、添付の特許請求の範囲によってのみ限定されることを理解されたい。

【０２５９】

上述した影響分析手法は、適するソフトウェアを実行する計算システムを使用して実施することができる。例えば、ソフトウェアは、少なくとも１つのプロセッサ、少なくとも１つのデータ記憶システム（揮発性及び／又は不揮発性メモリ及び／又は記憶要素を含む）、少なくとも１つのユーザインタフェース（少なくとも１つの入力デバイス又はポートを使用して入力を受信し、少なくとも１つの出力デバイス又はポートを使用して出力を提供するため）をそれぞれ含む１つ又は複数のプログラムされているか、又はプログラム可能な計算システム（分散、クライアント／サーバ、又はグリッド等の様々なアーキテクチャのものであることができる）で実行される１つ又は複数のコンピュータプログラム内の手順を含むことができる。ソフトウェアは、例えば、データフローグラフの設計、構成、及び実行に関連するサービスを提供するより大きいプログラムの１つ又は複数のモジュールを含むことができる。プログラムのモジュール（例えば、データフローグラフの要素）は、データリポジトリに記憶されるデータモデルに準拠するデータ構造又は他の編成データとして実施することができる。

【０２６０】

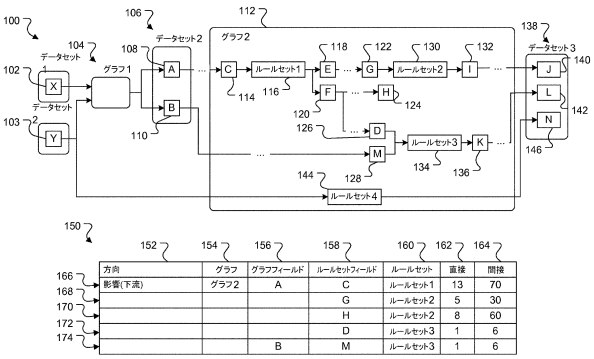
ソフトウェアは、ＣＤ－ＲＯＭ又は他のコンピュータ可読媒体（例えば、汎用又は専用計算システム又はデバイスにより可読）等の有形の非一時的媒体で提供することができ、又はネットワークの通信媒体を介して、実行される計算システムの有形の非一時的媒体に送る（例えば、伝搬信号に符号化する）ことができる。処理の幾つか又は全ては、専用コンピュータで、又はコプロセッサ若しくはフィールドプログラマブルゲートアレイ（ＦＰＧＡ）若しくは専用の特定用途向け集積回路（ＡＳＩＣ）等の専用ハードウェアを使用して実行することができる。処理は、ソフトウェアにより指定される計算の異なる部分が異なる計算要素により実行される分散方式で実施することができる。そのような各コンピュータプログラムは、好ましくは、汎用又は専用プログラマブルコンピュータによりアクセス可能な記憶デバイスのコンピュータ可読記憶媒体（例えば、固体状態メモリ若しくは媒体又は磁気若しくは光学媒体）に記憶されるか、又はダウンロードされて、記憶デバイス媒体がコンピュータにより読み取られると、本明細書に記載の処理を実行するようにコンピュータを構成し動作させる。本発明のシステムは、コンピュータプログラムが構成される有形の非一時的媒体として実施されるものとして見なされることもでき、ここで、媒体は、特定の予め定義される方式でコンピュータを動作させて、本明細書に記載の処理ステ

ップの１つ又は複数を実行させるように構成される。

【０２６１】

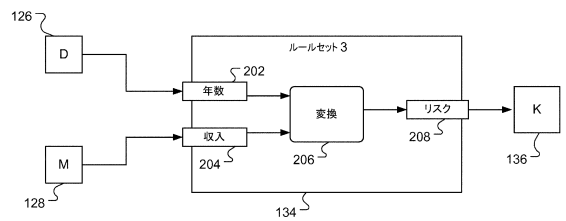
本発明の幾つかの実施形態について説明した。それにもかかわらず、上記説明が、例示を意図し、本発明の範囲の限定を意図せず、本発明の範囲が以下の特許請求の範囲により規定されることを理解されたい。したがって、他の実施形態も以下の特許請求の範囲内にある。例えば、本発明の範囲から逸脱せずに、様々な変更形態がなされ得る。更に、上述したステップの幾つかは、独立して順序付けることができ、したがって、記載される順序と異なる順序で実行され得る。

【図１】



	方向	グラフ	グラフフィールド	ルールセットフィールド	ルールセット	直接	間接
166	影響(下流)	グラフ2	A	C	ルールセット1	13	70
168				G	ルールセット2	5	30
170				H	ルールセット2	8	60
172				D	ルールセット3	1	6
174			B	M	ルールセット3	1	6

【図２】

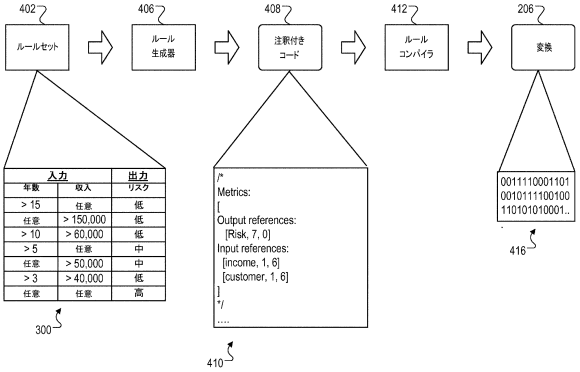


【図３】

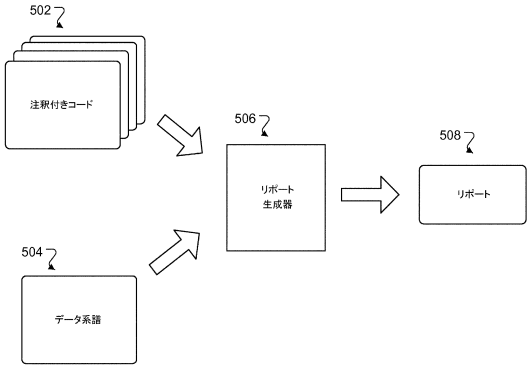
300

	302 入力		304 出力
	306 年数	308 収入	310 リスク
312	>15	任意	低
314	任意	>150,000	低
316	>10	>60,000	低
318	>5	任意	中
320	任意	>50,000	中
322	>3	>40,000	低
324	任意	任意	高

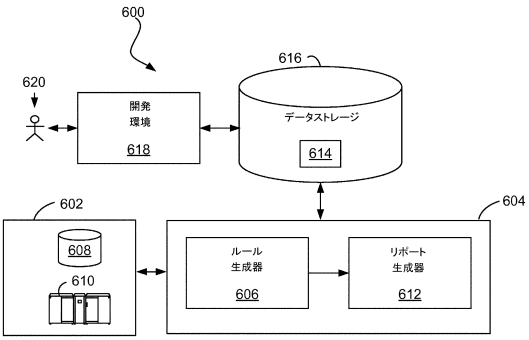
【図４】



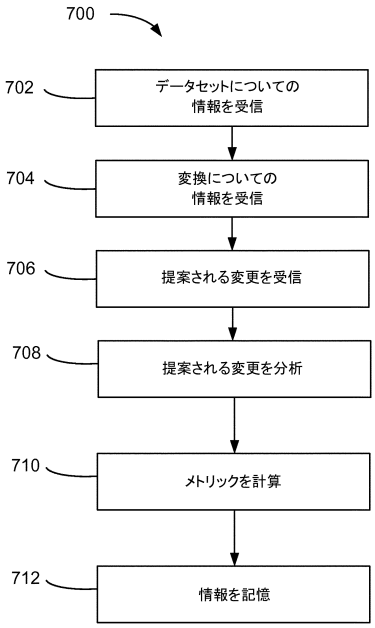
【図５】



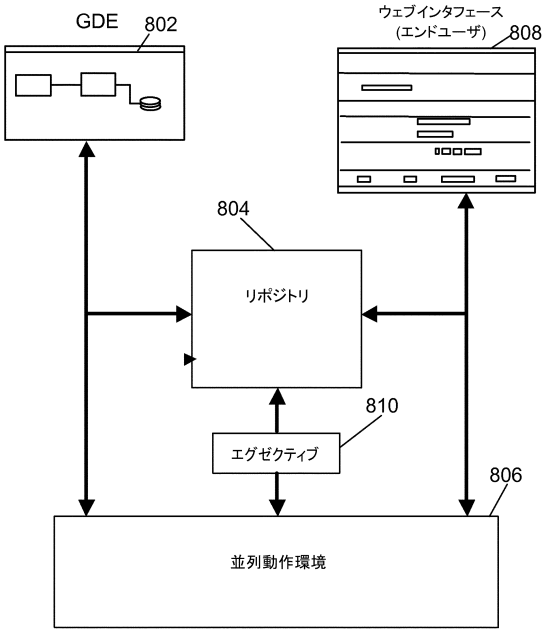
【図６】



【図７】



【図８Ａ】



【図８Ｂ】

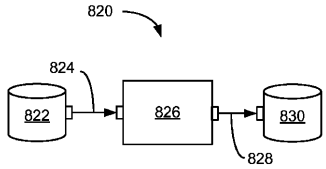
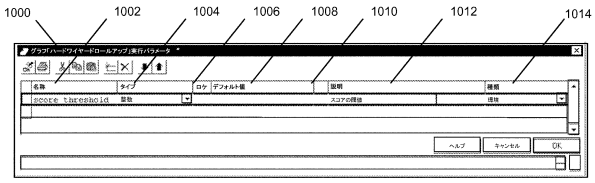
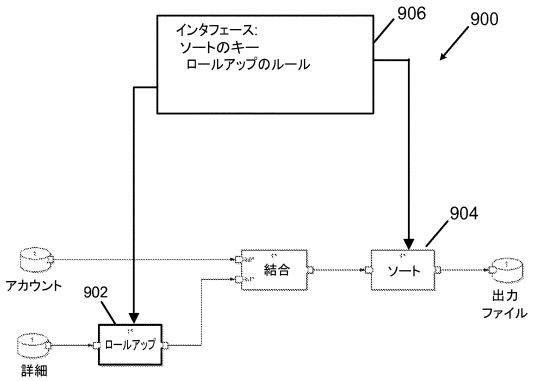


FIG. 8B

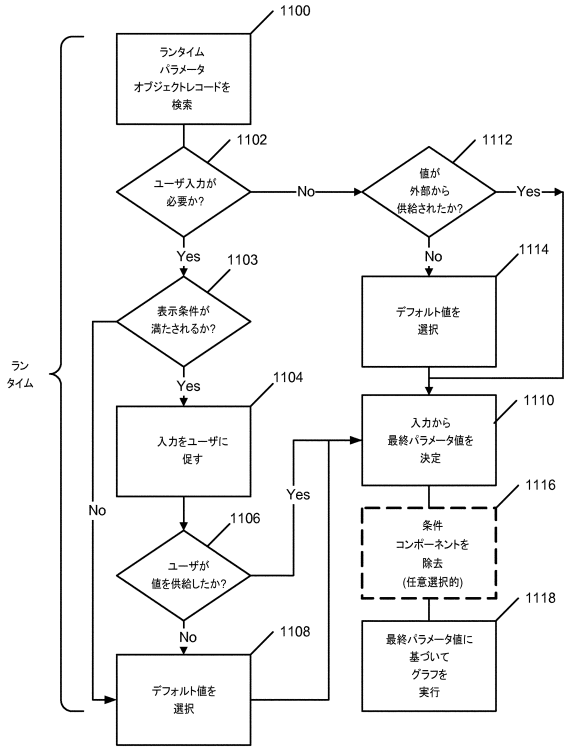
【図１０】



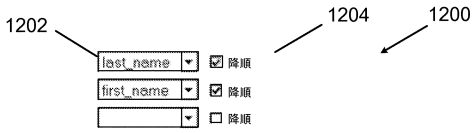
【図９】



【図１１】



【図１２】



【図１３】

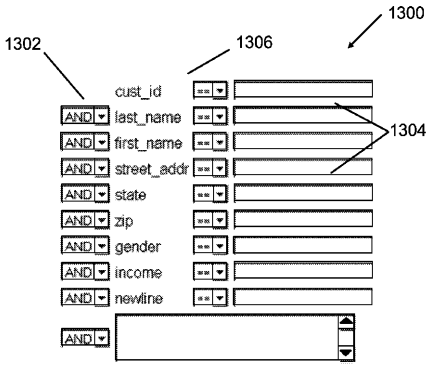
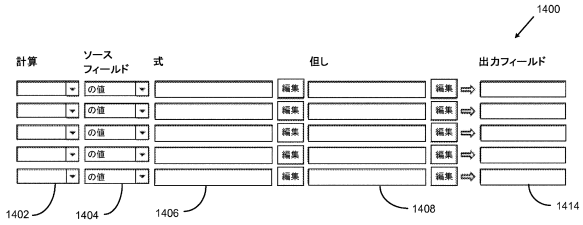
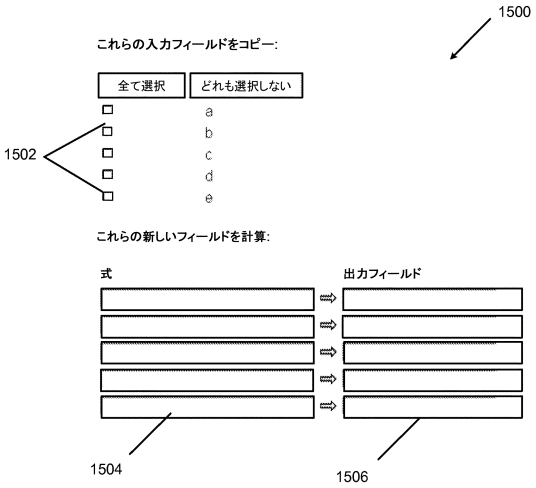


FIG. 13

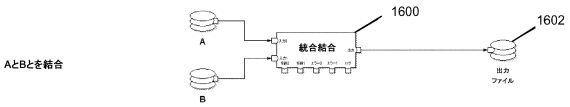
【図１４】



【図１５】

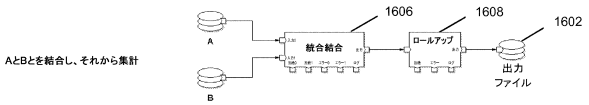


【図１６Ａ】

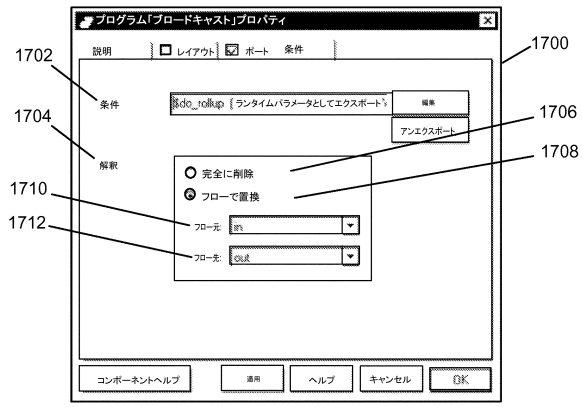


【図１６Ｂ】

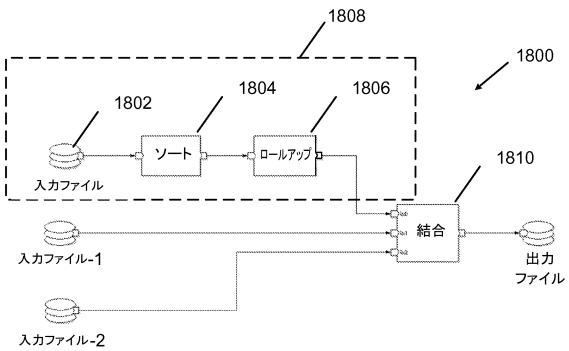
【図１６Ｃ】



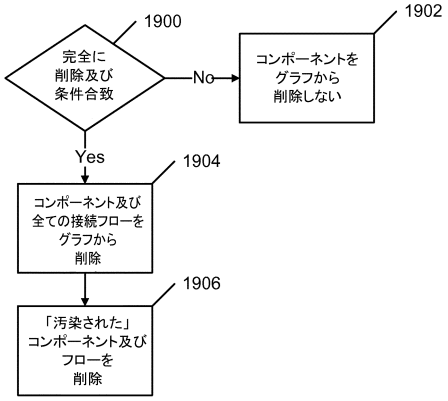
【図１７】



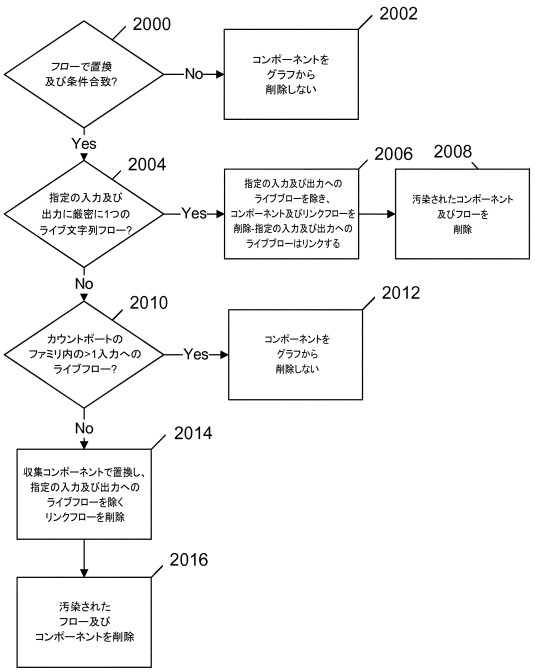
【図１８】



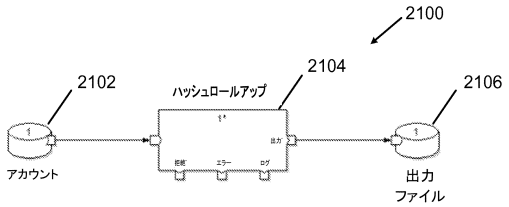
【図１９】



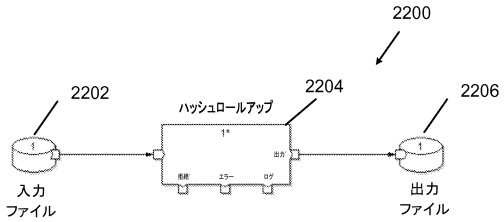
【図２０】



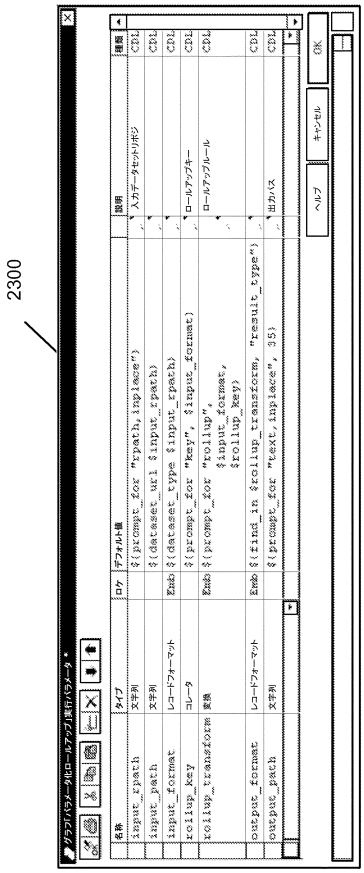
【図２１】



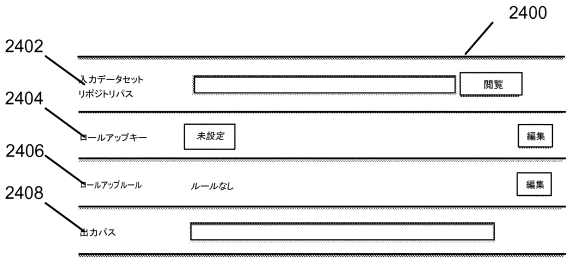
【図２２】



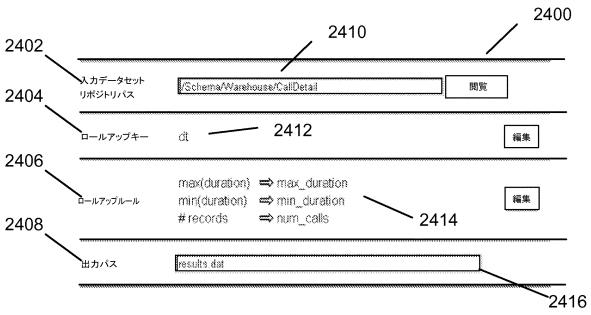
【図２３】



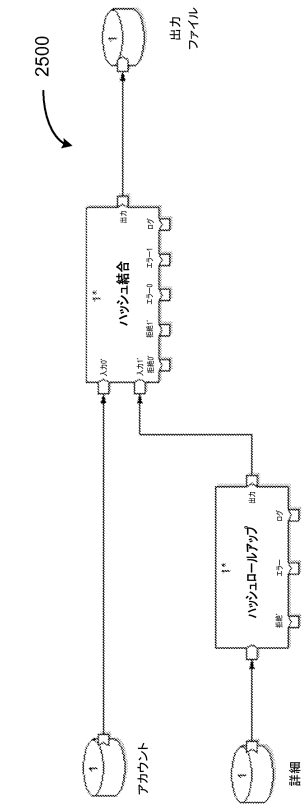
【図２４Ａ】



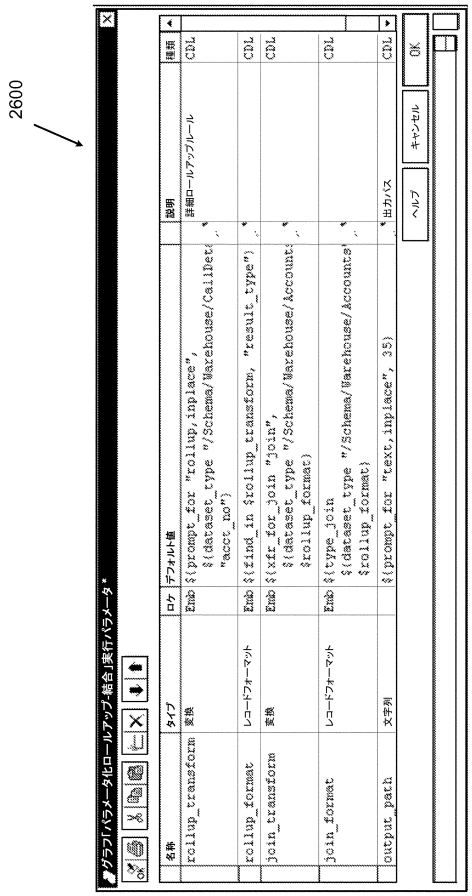
【図２４Ｂ】



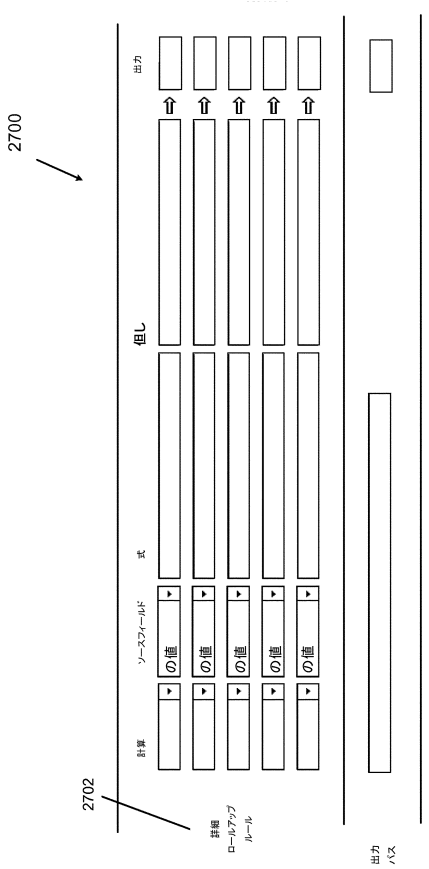
【図２５】



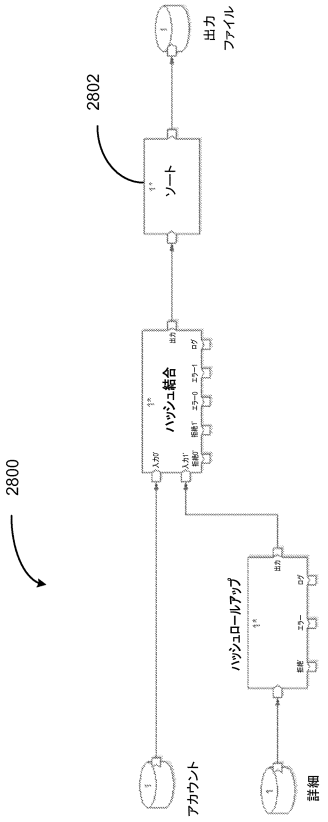
【図２６】



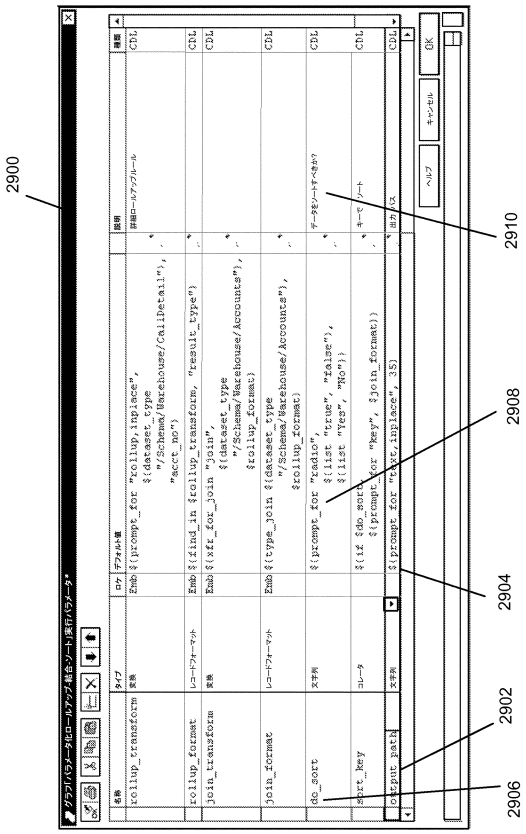
【図２７】



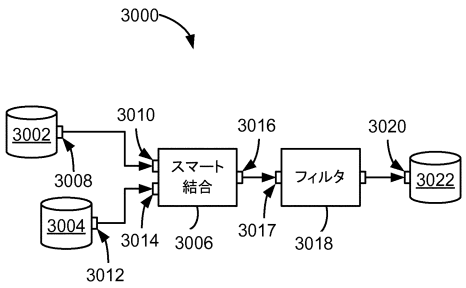
【図２８】



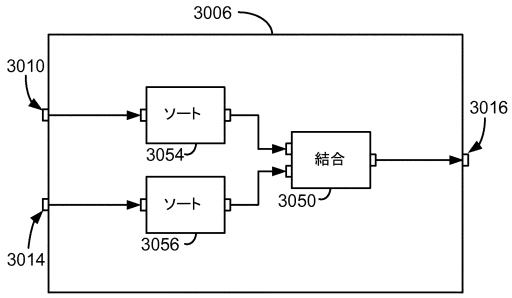
【図２９】



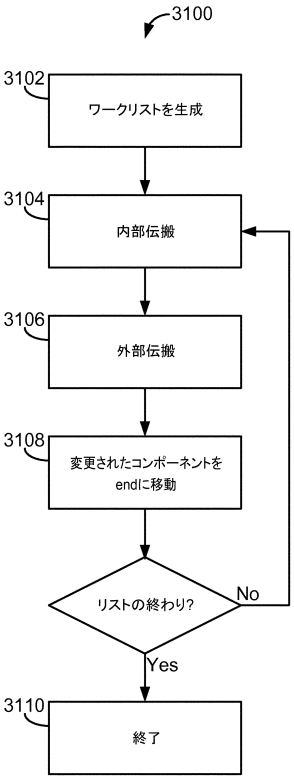
【図３０Ａ】



【図３０Ｂ】



【図３１】



【図３２Ａ】

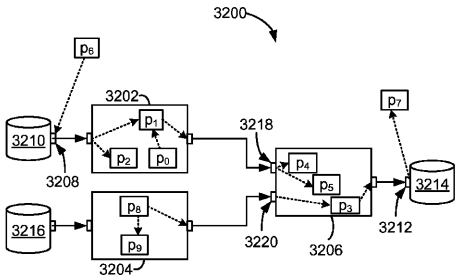


FIG. 32A

【図３２Ｂ】

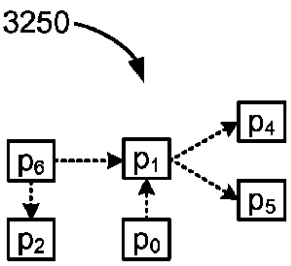


FIG. 32B

【図３２Ｃ】

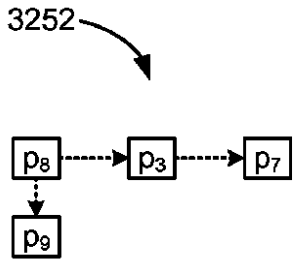


FIG. 32C

【図３３】

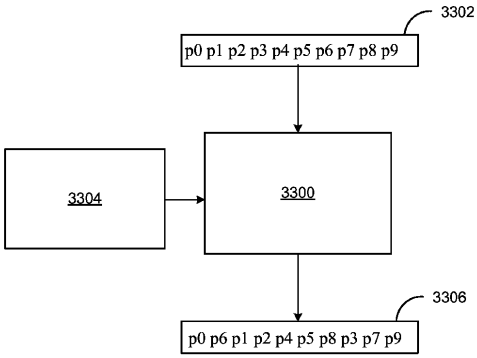
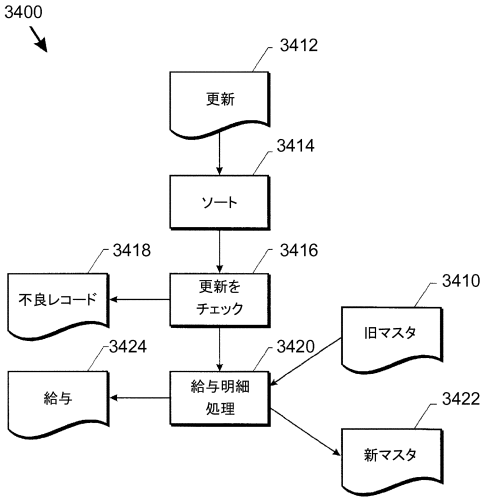
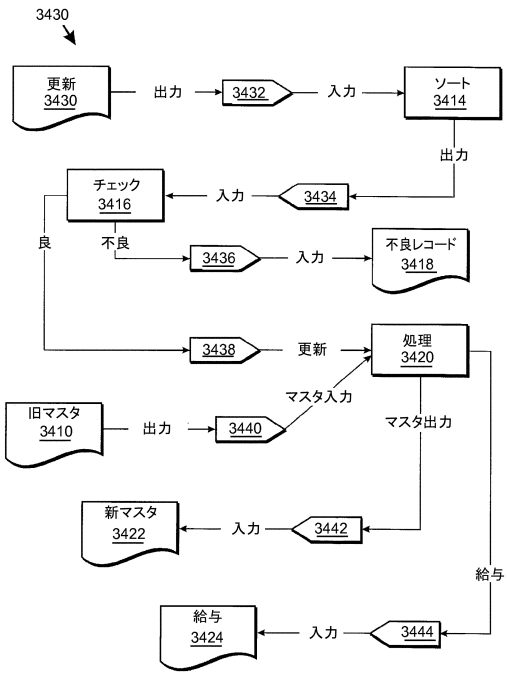


FIG. 33

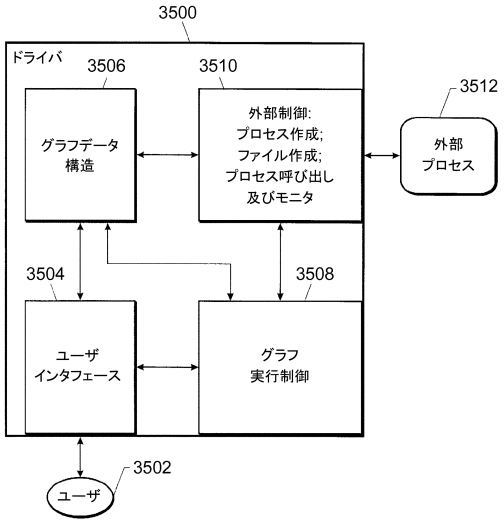
【図３４Ａ】



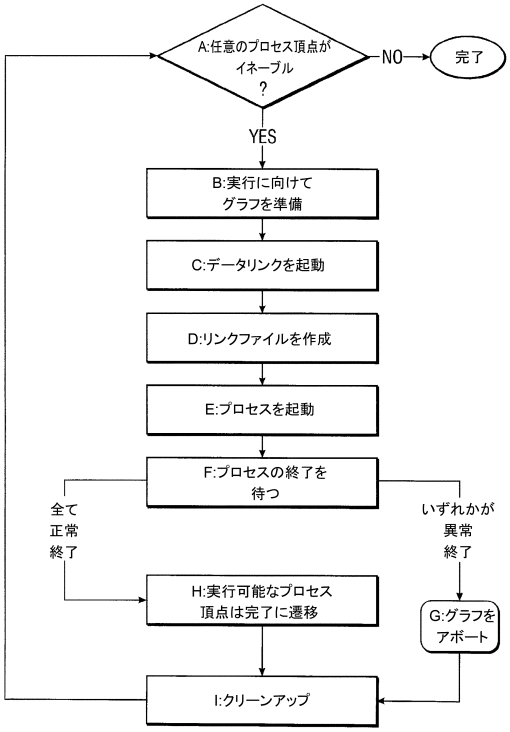
【図３４Ｂ】



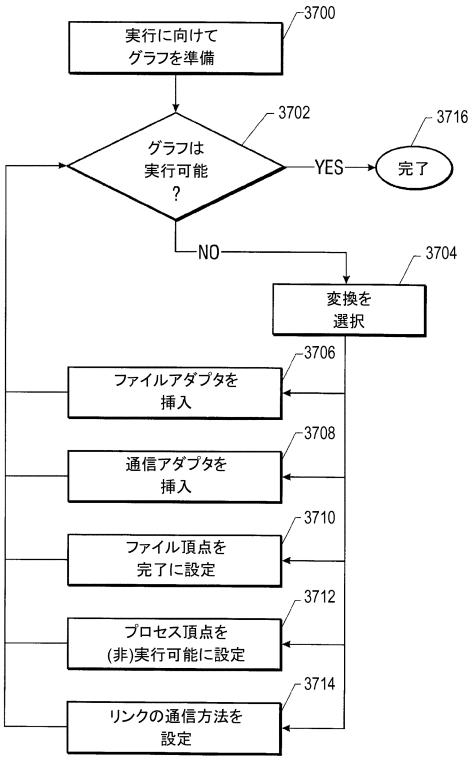
【図３５】



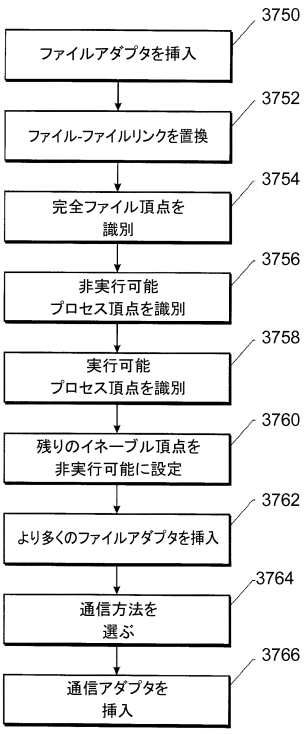
【図３６】



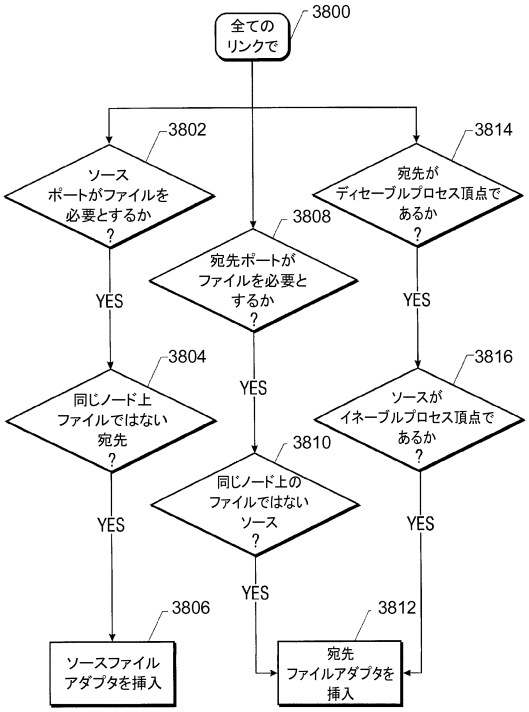
【図３７Ａ】



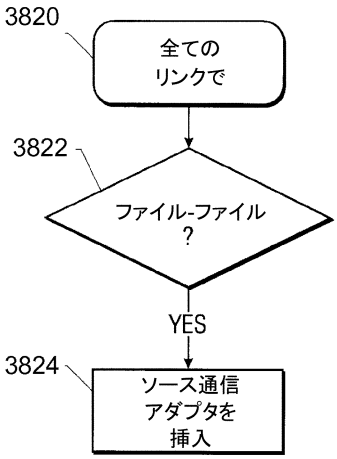
【図３７Ｂ】



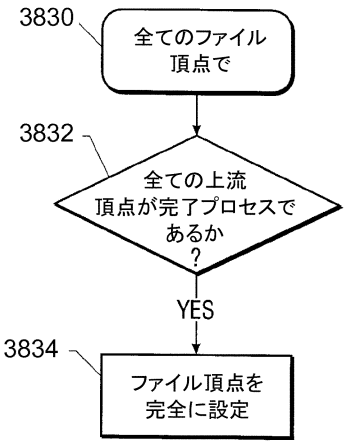
【図３８Ａ】



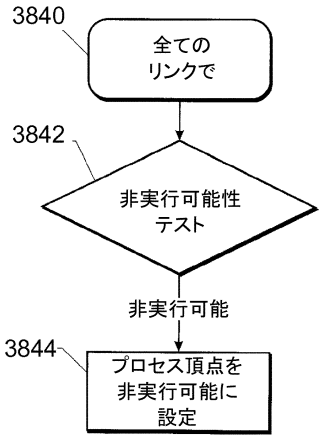
【図３８Ｂ】



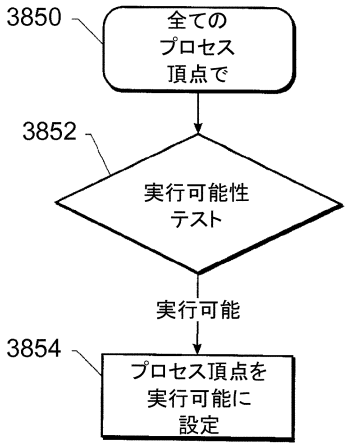
【図３８Ｃ】



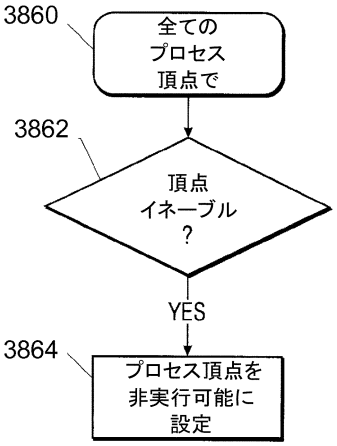
【図３８Ｄ】



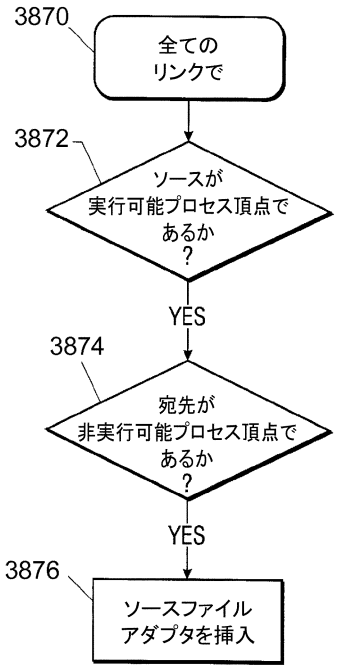
【図３８Ｅ】



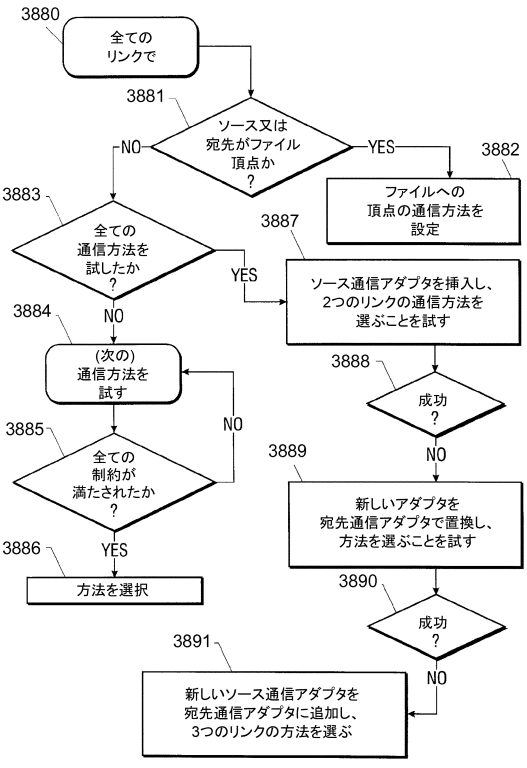
【図３８Ｆ】



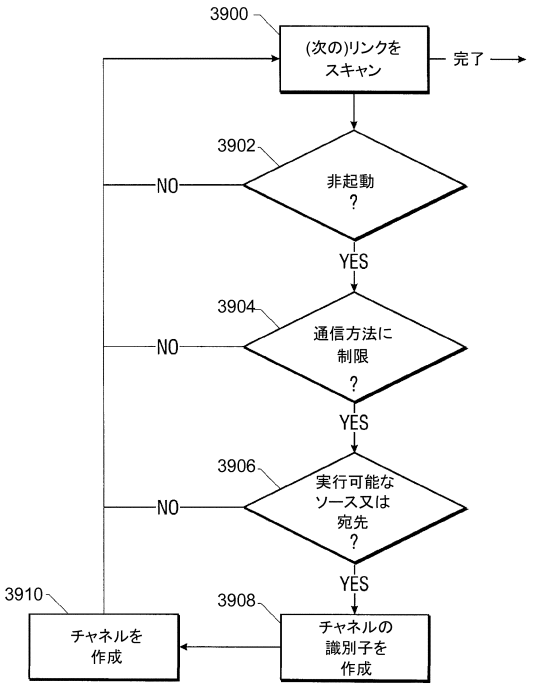
【図３８Ｇ】



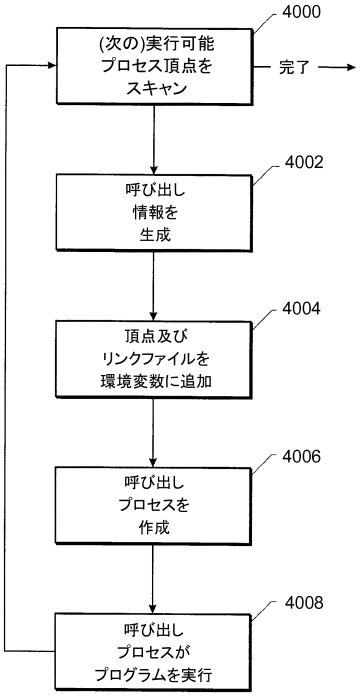
【図３８Ｈ】



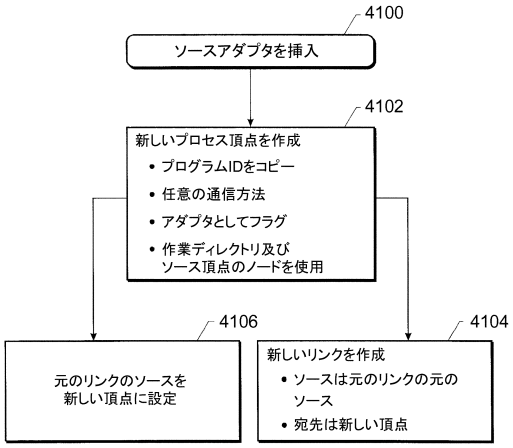
【図３９】



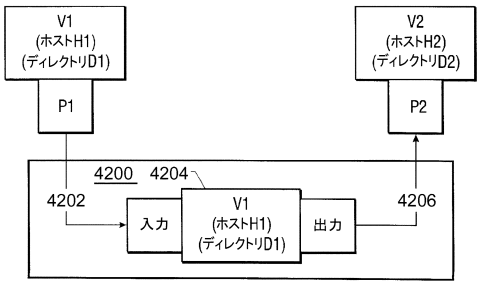
【図４０】



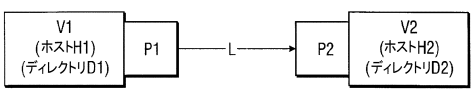
【図４１】



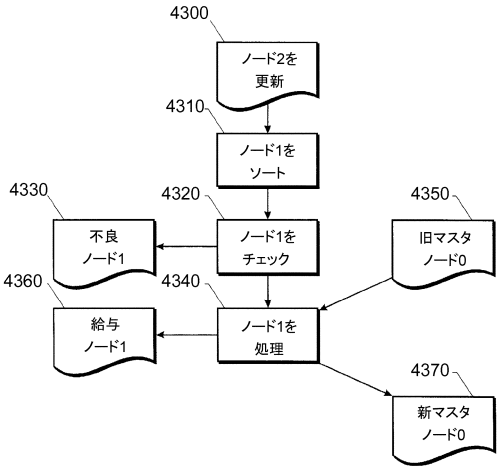
【図４２Ｂ】



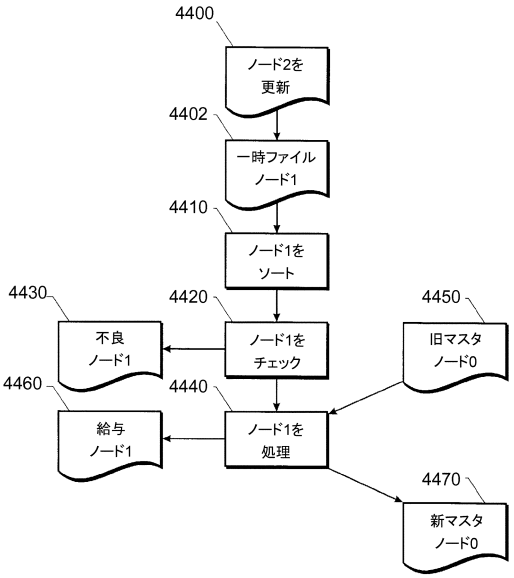
【図４２Ａ】



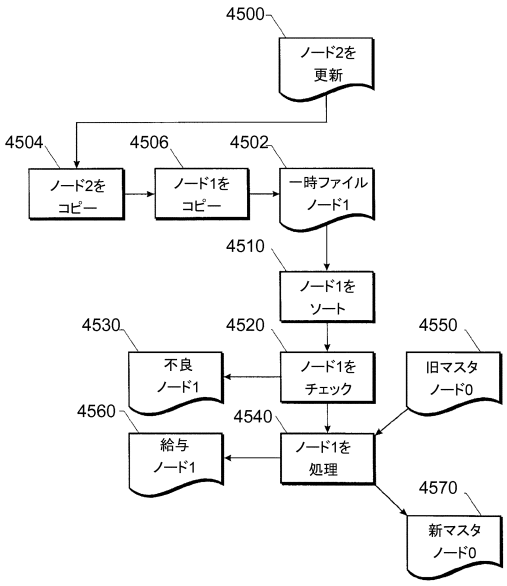
【図４３】



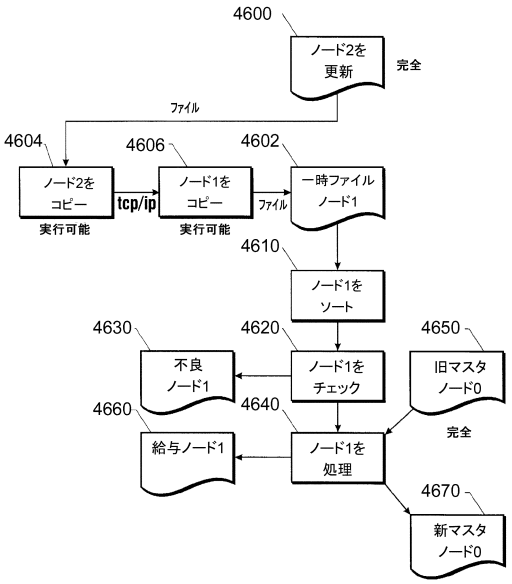
【図４４】



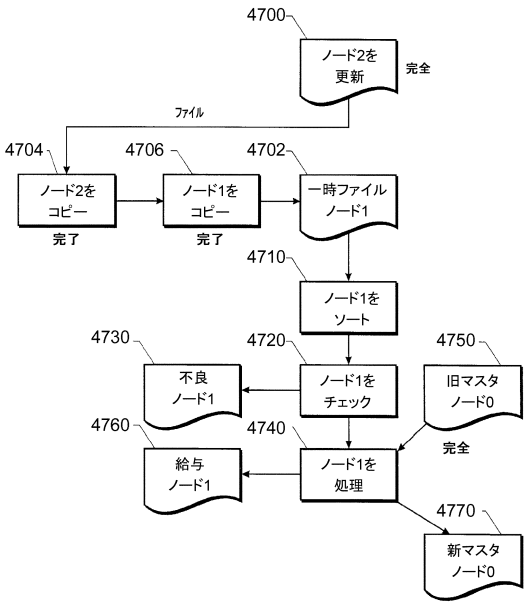
【図４５】



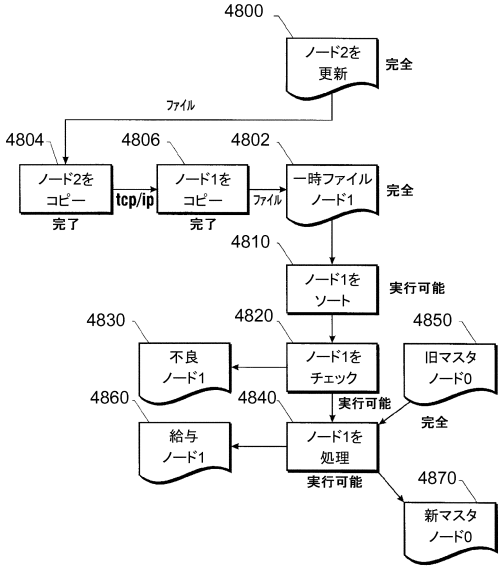
【図４６】



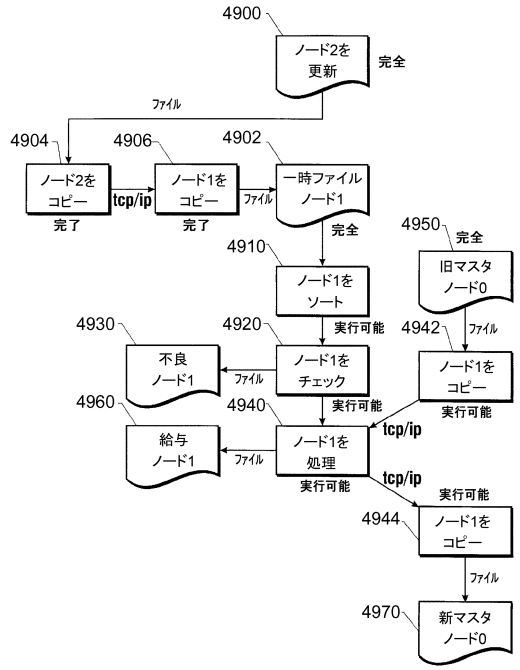
【図４７】


4700
ノード2を
更新
完全
ファイル
4704
4706
4702
ノード2を
コピー
ノード1を
コピー
一時ファイル
ノード1
完了
完了
4710
ノード1を
ソート
4730
4720
4750
不良
ノード1
ノード1を
チェック
旧マスタ
ノード0
4760
4740
完全
給与
ノード1
ノード1を
処理
4770
新マスタ
ノード0

【図４８】


4800
ノード2を
更新
完全
ファイル
4804
4806
4802
ノード2を
コピー
tcp/ip
ノード1を
コピー
ファイル
一時ファイル
ノード1
完了
完了
完全
4810
ノード1を
ソート
実行可能
4830
4820
4850
不良
ノード1
ノード1を
チェック
旧マスタ
ノード0
4860
4840
実行可能
完全
給与
ノード1
ノード1を
処理
実行可能
4870
新マスタ
ノード0

【図４９】


4900
ノード2を
更新
完全
ファイル
4904
4906
4902
ノード2を
コピー
tcp/ip
ノード1を
コピー
ファイル
一時ファイル
ノード1
完了
完了
完全
4950
完全
4910
旧マスタ
ノード0
ノード1を
ソート
実行可能
4930
4920
4942
ファイル
不良
ノード1
ファイル
ノード1を
チェック
ノード1を
コピー
実行可能
実行可能
4960
4940
tcp/ip
給与
ノード1
ファイル
ノード1を
処理
実行可能
tcp/ip
実行可能
4944
ノード1を
コピー
ファイル
4970
新マスタ
ノード0

フロントページの続き

前置審査

(72)発明者 ゴウルド，ジョエル
アメリカ合衆国，マサチューセッツ州 ０２４７４，アーリントン，リー テラス ２７
(72)発明者 ストゥーダー，スコット
アメリカ合衆国，メリーランド州 ２１４０３－２８０２，アナポリス，ジャニス ドライブ ８０４

審査官 松崎 孝大

(56)参考文献 特開２０００－３３９１４５（ＪＰ，Ａ）
特開２００７－１２８１２３（ＪＰ，Ａ）
特開２００９－１５７５０５（ＪＰ，Ａ）
特開２０１０－０１５４５８（ＪＰ，Ａ）
国際公開第２００９／０１１０５７（WO，Ａ１）
米国特許出願公開第２０１０／０１２２２４０（ＵＳ，Ａ１）
米国特許出願公開第２０１０／０１１４９６２（ＵＳ，Ａ１）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ ８／７０
Ｇ０６Ｆ １１／３６