(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)特許出願公表番号
特表2004-507009
(P2004-507009A)
(43)公表日　平成16年3月4日(2004.3.4)

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

審査請求 未請求　予備審査請求 有　（全 97 頁）

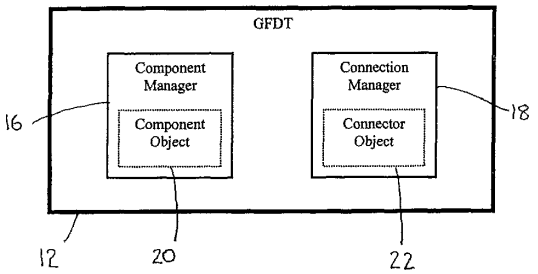
(21)出願番号　特願2002-521699(P2002-521699)
(86)(22)出願日　平成13年8月24日(2001.8.24)
(85)翻訳文提出日　平成14年4月24日(2002.4.24)
(86)国際出願番号　PCT/AU2001/001053
(87)国際公開番号　WO2002/017074
(87)国際公開日　平成14年2月28日(2002.2.28)
(31)優先権主張番号　PQ 9664
(32)優先日　平成12年8月24日(2000.8.24)
(33)優先権主張国　オーストラリア(AU)

(71)出願人　502147834
ゼンプレックス　プロプライエタリー　リミテッド
ＸＥＭＰＬＥＸ　ＰＴＹ　ＬＴＤ
オーストラリア国，　ウェスタンオーストラリア　６１５８，　イースト　フレマントル，　フラサー　ストリート　５７Ｂ
５７Ｂ　Ｆｒａｓｅｒ　Ｓｔｒｅｅｔ　Ｅａｓｔ　Ｆｒｅｍａｎｔｌｅ　ＷＡ　６１５８　Ａｕｓｔｒａｌｉａ
(74)代理人　100097180
弁理士　前田　均
(74)代理人　100099900
弁理士　西出　眞吾

最終頁に続く

(54)【発明の名称】式をグラフィカルに定義する方法

(57)【要約】
少なくとも１つの入力を処理して少なくとも１つの結果を生成する方法を定義するための第１の演算子オブジェクトを提供することを含む、コンピュータにより実行される、グラフィカルに式を定義する方法。第１の演算子オブジェクトのグラフィカルな表現が表示される。データを含むための変数オブジェクトが提供される。変数オブジェクトを第１の演算子オブジェクトの入力の１つ又は結果の１つに関連付けるためのユーザからの入力が受け取られる。第１の変数オブジェクトのグラフィカルな表現及びその演算子オブジェクトに対する関係が表示される。オブジェクト間の関係の論理的な記述が記録され、これによって式を定義する。
【選択図】図２



【特許請求の範囲】
【請求項１】
少なくとも１つの入力を処理し少なくとも１つの結果を生成する方法を定義するための第１の演算子オブジェクトを提供し、
前記第１の演算子オブジェクトをグラフィカルな表現で表示し、
データを保持するための第１の変数オブジェクトを提供し、
前記変数オブジェクトを前記第１の演算子オブジェクトの前記入力の１つまたは前記出力の１つに関係付ける入力をユーザから受け付け、
前記第１の変数オブジェクト及びその前記第１の演算子オブジェクトとの関係をグラフィカルな表現で表示し、
前記オブジェクト間の関係の論理的な記述を記録し、
これにより、論理的記述により式が定義される
コンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項２】
データを保持するための変数オブジェクトを提供し、
前記変数オブジェクトをグラフィカルな表現で表示し、
少なくとも１つの入力を処理し少なくとも１つの結果を生成する方法を定義するための第１の演算子オブジェクトを提供し、
前記第１の演算子オブジェクトの前記入力の１つまたは前記出力の１つを前記変数オブジェクトに関係付ける入力をユーザから受け付け、
前記第１の演算子オブジェクト及びその前記変数オブジェクトとの関係をグラフィカルな表現で表示し、
前記オブジェクト間の関係の論理的な記述を記録し、
これにより論理的記述により式が定義される
コンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項３】
１以上のさらなる変数オブジェクトを提供し、
前記さらなる変数オブジェクトの各々を第１の演算子オブジェクトの前記入力の１つまたは前記出力の１つに関係付けるさらなる入力をユーザから受け付け、
前記さらなる変数オブジェクト及びそれらの前記演算子オブジェクトとの関係をグラフィカルな表現で表示する
工程をさらに有する請求項１又は２に記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項４】
１以上のさらなる演算子オブジェクトを提供し、
各変数オブジェクトを前記さらなる演算子オブジェクトの前記入力の１つまたは前記出力の１つに関係付けるさらなる入力をユーザから受け付け、
前記さらなる演算子オブジェクト及びその前記変数オブジェクトとの関係をグラフィカルな表現で表示する
工程をさらに有する請求項１～３のいずれかに記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項５】
各変数オブジェクトは、データソースからのデータを提供するための入力オブジェクト、データの行き先にデータを供給する出力オブジェクト、又は、１つの演算子オブジェクト又はもう１つの演算子オブジェクトからデータを流すための結合オブジェクトより選択される
請求項１～４のいずれかに記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項６】
結合オブジェクトは演算子オブジェクトの間のリンクとして表現される

請求項１～５のいずれかに記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項７】
各変数オブジェクトは変数ラベルを伴って提供される
請求項１～６のいずれかに記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項８】
各演算子オブジェクトは演算子ラベルを伴って提供される
請求項１～７のいずれかに記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項９】
前記式の前記論理的な記述は前記オブジェクト間の論理的関係により定義される
請求項１～８のいずれかに記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項１０】
オブジェクト間の前記関係の前記グラフィカルな表示を定義する前記式のグラフィカルな定義は記録される
請求項１～９のいずれかに記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項１１】
前記論理的な定義を記述する情報を記憶するステップを含む
請求項１～１０のいずれかに記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項１２】
前記グラフィカルな定義定義を記述する情報を記憶するステップを含む
請求項１～１０のいずれかに記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項１３】
２以上の前記演算子オブジェクトがグループ化されており、グループ化は、グルーピングの演算子オブジェクトを定義し、その中においてはグルーピングの境界を横切りグループ内の演算子オブジェクトの入力に対して結合されている変数オブジェクトがグルーピングオブジェクトコンポーネントの入力となり、グルーピングの境界をクロスしグループ内の演算子オブジェクトの結果に結合された変数オブジェクトは、グルーピングの演算子オブジェクトの結果となる
請求項１～１２のいずれかに記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項１４】
他のオブジェクトにリンクされていないグループ内の演算子オブジェクトの入力及び結果は、各々、グループのオペレータオブジェクトの入力及び結果となる　請求項１３に記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項１５】
グループ化されたオブジェクトのグラフィカルな表現は、グループの演算子オブジェクトのグラフィカルな表現によって配置され、グループのコンテンツへのグラフィカルな表現は、グループのオブジェクトの表現へのリンクのグラフィカルな表現により配置される
請求項１３又は１４に記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項１６】
定義された式の論理的な定義は、グループの演算子オブジェクトの内容を含む　請求項１３又は１５に記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項１７】

表示される全ての式のグラフィックの定義は、グループの演算子オブジェクトの内容を含まない
請求項１３又は１６に記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項１８】
グループの演算子オブジェクトの内容は、式の全てのグラフィカルな表現とは分離されてグラフィカルに表現される
請求項１３又は１７に記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項１９】
変数オブジェクトは保持可能なデータのタイプを定義する属性を付される
請求項１～１８のいずれかに記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項２０】
演算子オブジェクトの入漁ｋ及び結果は、演算子オブジェクトが受け取ること及び生成することが各々予測されるデータのタイプを定義する属性を付される
請求項１９に記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項２１】
変数オブジェクトは、既に定義されており調停する演算子オブジェクトによって関係付けられている他の変数オブジェクトの属性から、属性を受け継ぐ
請求項２０に記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項２２】
変数オブジェクトは、既に定義されており、関係付けられている演算子オブジェクトの入力又は結果の属性から、属性を受け継ぐ
請求項２０に記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項２３】
演算子オブジェクトの入力または結果は、既に定義されており関係付けられている変数オブジェクトの属性から、属性を受け継ぐ
請求項１９～２２のいずれかに記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項２４】
既に付されており対応が関係付けられているオブジェクトの属性をチェックするステップを含む
請求項１９から２３のいずれかに記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項２５】
ラベル付けされた変数オブジェクトのライブラリが予め定義されている
請求項７に記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項２６】
ラベル付けされた変数オブジェクトのライブラリが予め定義されており、各ラベル付けされた変数オブジェクトの処理の方法、その生成のための入力及び出力もまた、予め定義されている
請求項８に記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項２７】
変数オブジェクトの変数ラベルは、予め定義した変数ラベルのリストから選択される
請求項７又は２５に記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項２８】
各変数ラベルには、ラベルによりラベル付けされた変数オブジェクトが含むことができる

データのタイプが定義されている属性が付されている
請求項１９に記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項２９】
変数ラベルの選択は、変数オブジェクトへのラベルに関連付けられた属性が付される
請求項１９に記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項３０】
変数オブジェクトに付されている属性は、選択が有効なラベルの選択を制限する
請求項２９に記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項３１】
演算子オブジェクトは、少なくとも、加算、減算、乗算、除算、ルックアップテーブル及び条件付動作の１つである
請求項１～３０のいずれかに記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項３２】
演算子オブジェクトは、より複雑な演算子の実行にリンクされた多数の単純な演算子を含む多数の段階の演算である
請求項１～３０のいずれかに記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項３３】
一形態においては、演算子オブジェクトはデータベースのキュエリである
請求項１～３０のいずれかに記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項３４】
第１の演算子オブジェクトはデータベースへの書き込みを行う
請求項１～３０のいずれかに記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項３５】
演算子オブジェクトの演算子ラベルは予め定義された演算子ラベルのリストより選択される
請求項８に記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項３６】
各演算子ラベルには、ラベルを付された演算子オブジェクトが受け取り又は提供することのできる入力又は結果の各々のデータのタイプを定義する属性が付される
請求項３５に記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項３７】
演算子ラベルの選択は、演算子オブジェクトへのラベルに関連する属性が付されている
請求項３６に記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項３８】
演算子オブジェクトに付された属性は、選択されることが可能なラベルの選択を制限する
請求項３７に記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項３９】
論理的な記述はランタイムエンジンにより使用され、定義した式の処理が行われ、各変数オブジェクトに提供されたデータは、演算子オブジェクトの入力にリンクされ、これにより、データは、式のオペランドとなり、演算子オブジェクトによって表現される各演算子は、式の演算子となり、演算子オブジェクトの各結果は、次の演算子又は式の最終的な結果となり、これにより、式の処理が実行され式の結果が得られる
請求項１～３８のいずれかに記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項４０】
各変数に対して名前領域が定義され、これにより、変数オブジェクトにより表現される論

理的な変数におけるデータは、名前空間内の変数オブジェクトの各発生と同じである。
請求項１～３９のいずれかに記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項４１】
名前空間は、モデル化された式に対するデフォルトグローバルによる
請求項４０に記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項４２】
論理的な結合は、名前空間内のラベルが付された変数オブジェクトの各発生の間に生成される
請求項４０に記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項４３】
グラフィカルなリンクが表示され、ラベル付けされた変数オブジェクトの発生の間の論理的な結合が示される
請求項４０に記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項４４】
名前空間は各演算子オブジェクトに定義され、これにより、演算子オブジェクトにより表現される論理的な演算子の処理は、名前空間内の演算子オブジェクトの各発生と同じである
請求項１～４３のいずれかに記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項４５】
名前空間は、モデル化された式に対するデフォルトグローバルによる
請求項４４に記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項４６】
グループ化された演算子オブジェクトは、１回以上グループ化された演算子オブジェクトの定義に使用され、式の論理的な定義に適用される
請求項１３に記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項４７】
ラベルの属性は、タイプ、単位及びディメンションを有する
請求項１９に記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項４８】
グラフィカルな定義は、ＸＭＬにより記述される
請求項１０に記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項４９】
論理的な記述は、ＸＭＬにより記述される
請求項１～４８のいずれかに記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項５０】
各演算子オブジェクトは、演算子オブジェクトにより表現される演算子により実行される多数の演算の定義を含み、各定義は、演算子により処理されることができるデータの分離したタイプのためである
請求項１～４９のいずれかに記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項５１】
演算子オブジェクトは、１つ以上の入力及び１つ以上の出力を有するコンポーネントとしてグラフィカルに表現され、そのコンポーネントは、表現される演算子に表示されるインジケータを有する
請求項１～５０のいずれかに記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項５２】

演算子オブジェクトは、結果を生成するための処理の方法の入力が未だ定義されていない演算子の表現である空コンポーネントである
請求項１～５１のいずれかに記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項５３】
前記空コンポーネントは、適切に定義された方法を有する適切な演算子オブジェクトをサーチするための基準を形勢するために使用される
請求項１～５１のいずれかに記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項５４】
オブジェクトのライブラリが提供される
請求項１～５３のいずれかに記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項５５】
オブジェクトは、外部より供給される
請求項１～５３のいずれかに記載のコンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項５６】
ディスプレイ及びユーザ入力手段を有するコンピュータと、
少なくとも１つの入力を処理し少なくとも１つの結果を生成する方法を定義するための第１の演算子オブジェクトを提供する手段と、
前記第１の演算子オブジェクトをグラフィカルな表現で表示する手段と、
データを保持するための第１の変数オブジェクトを提供する手段と、
前記変数オブジェクトを前記第１の演算子オブジェクトの前記入力の１つまたは前記出力の１つに関係付ける入力をユーザから受け付ける手段と、
前記第１の変数オブジェクト及びその前記第１の演算子オブジェクトとの関係をグラフィカルな表現で表示する手段と
を有し、これによりオブジェクト間の関係により式が定義される
グラフィカルに式を定義するシステム。
【請求項５７】
コンピュータをコントロールすることにより式をグラフィカルに定義するコンピュータプログラムであって、
少なくとも１つの入力を処理し少なくとも１つの結果を生成する方法を定義するための第１の演算子オブジェクトを提供し、
前記第１の演算子オブジェクトをグラフィカルな表現で表示し、
データを保持するための第１の変数オブジェクトを提供し、
前記変数オブジェクトを前記第１の演算子オブジェクトの前記入力の１つまたは前記出力の１つに関係付ける入力をユーザから受け付け、
前記第１の変数オブジェクト及びその前記第１の演算子オブジェクトとの関係をグラフィカルな表現で表示し、
前記オブジェクト間の関係の論理的な記述を記録し、
これにより、オブジェクト間の関係により式が定義される
コンピュータプログラム
【請求項５８】
請求項５７に記載のコンピュータプログラムが記録されたコンピュータにより読み取り可能な記録媒体。
【請求項５９】
少なくとも１つのデータを保持するための変数オブジェクトを提供し、
少なくとも１つの前記入力データを処理し結果を生成する方法を定義する演算子を提供し、

ユーザに結果の変数を選択するための変数のリストを表示し、
入力データの処理の結果を記憶するための結果の変数の選択をユーザから受け取り、
前記選択された結果の変数をグラフィカルな表現により表示し、
演算子を選択するための演算子のリストをユーザに表示し、
ユーザより演算子の選択を受け取り、
選択された演算子のグラフィカルな表現を表示し、
ユーザに前記変数か１以上の定数であるところの少なくとも１つの入力を選択させるため、入力データを記憶するための入力のリストを表示し、
ユーザからの少なくとも１つの入力の選択を受け取り、
前記選択された入力のグラフィカルな表現を表示し、
これにより、選択された結果変数を、選択された処理による選択された入力の処理と等しくすることによって、式を定義する
コンピュータにより実行されるグラフィカルに式を定義する方法。
【請求項６０】
予め設定された属性を有する少なくとも１つの変数のタイプを提供し、
少なくとも１つの前記入力データを処理し結果を生成する方法を定義する少なくとも１つの演算を提供し、
ユーザに変数のタイプを選択させるため変数のタイプのリストを表示し、
ユーザからの変数のタイプの選択を受け取り、
選択した変数のタイプに対する名前を受け取り、
名前が付された変数を表示し、
ユーザに演算を選択させるための演算のリストを表示し、
ユーザからの演算の選択を受け取り、
選択された変数を選択された演算と協働させるためのユーザからの入力を受け取り、選択された変数を入力された変数か結果の変数とし、
選択された変数が結果の変数とされた場合には、ユーザから少なくとも１つの入力変数か入力定数及びその入力変数又は入力定数に対する名前を受け取り、その入力された変数又は入力された定数をグラフィカルに表示し、
選択された変数が入力の変数とされた場合には、出力変数の名前を受け取り、これをグラフィカルに表示し、
これにより、入力変数又は入力定数による入力データの選択された演算による処理の結果により式を定義し、結果の変数を得る
コンピュータにより実行されるグラフィカルに式を定義する方法。
【発明の詳細な説明】
【０００１】
【技術分野】
本発明は、入力データを処理して結果を生成するための式をグラフィカルに定義する方法に関する。特に、その方法は、コンピュータプログラムの形態により具体化される。
【０００２】
【背景技術】
データの複雑な処理のためには、通常、多くの複雑な式を必要とする。その式を定義するためには、しばしば、データの処理を表現する多数のレベルのアプローチを提供するモデルが有効である。モデルの各レベルのために必要とされている式を解くことは、しばしば難しいに違いない。もし式がグラフィカルに表現でき定義できるならば、それはこの処理のためにさらに役に立つに違いない。
【０００３】
コドスキー他による米国特許Ｎｏ．４９０１２２１は、プロセスをモデリングするためのグラフィカルなシステム及び方法を開示している。その開示されている方法は、図表を作成するユーザが、創造された図表がある結果を達成するための手続き上の方法をグラフィカルに表示するようなブロック図エディターを使用することを可能にする。ユーザがデー

タ流れ図を作成する時、機械語命令は自動的に生成され、それは表示手続きに対応する実行手続きを特徴つける。ユーザはグラフィカルなプログラミング環境を使用することによって、テキストベースのコンピュータプログラムを単独で創造することができる。この方法の限界は、これが、任意の与えられた時間に入力変数に当てるデータの関数の結果である各出力を生成するための繰り返し制御に依存しているということである。これはまた、データフローの多数の繰り返しを制御するための繰り返し制御手段を参照する繰り返しアイコンを含むデータ流れ図の、スクリーン上での組み立てにも依存している。

【０００４】

特にデータの繰り返しを考慮することなしにモデルを設計する場合には、モデルに適合したオブジェクトを設計する場合が望ましい。

【０００５】

【発明の開示】

本発明の目的は、式をグラフィカルに定義する方法を提供することにある。

【０００６】

本発明の第１の観点によれば、少なくとも１つの入力を処理し少なくとも１つの結果を生成する方法を定義するための第１の演算子オブジェクトを提供し、前記第１の演算子オブジェクトをグラフィカルな表現で表示し、データを保持するための第１の変数オブジェクトを提供し、前記変数オブジェクトを前記第１の演算子オブジェクトの前記入力の１つまたは前記出力の１つに関係付ける入力をユーザから受け付け、前記第１の変数オブジェクト及びその前記第１の演算子オブジェクトとの関係をグラフィカルな表現で表示し、前記オブジェクト間の関係の論理的な記述を記録し、これにより、論理的記述により式が定義されるコンピュータにより実行されるグラフィカルに式を定義する方法が提供される。

【０００７】

本発明の第２の観点によれば、データを保持するための変数オブジェクトを提供し、前記変数オブジェクトをグラフィカルな表現で表示し、少なくとも１つの入力を処理し少なくとも１つの結果を生成する方法を定義するための第１の演算子オブジェクトを提供し、前記第１の演算子オブジェクトの前記入力の１つまたは前記出力の１つを前記変数オブジェクトに関係付ける入力をユーザから受け付け、前記第１の演算子オブジェクト及びその前記変数オブジェクトとの関係をグラフィカルな表現で表示し、前記オブジェクト間の関係の論理的な記述を記録し、これにより論理的記述により式が定義されるコンピュータにより実行されるグラフィカルに式を定義する方法が提供される。

【０００８】

好適には、前記方法は、１以上のさらなる変数オブジェクトを提供し、前記さらなる変数オブジェクトの各々を第１の演算子オブジェクトの前記入力の１つまたは前記出力の１つに関係付けるさらなる入力をユーザから受け付け、前記さらなる変数オブジェクト及びそれらの前記演算子オブジェクトとの関係をグラフィカルな表現で表示する工程をさらに有する。

【０００９】

好適には、前記方法は、１以上のさらなる演算子オブジェクトを提供し、各変数オブジェクトを前記さらなる演算子オブジェクトの前記入力の１つまたは前記出力の１つに関係付けるさらなる入力をユーザから受け付け、前記さらなる演算子オブジェクト及びその前記変数オブジェクトとの関係をグラフィカルな表現で表示する工程をさらに有する。

【００１０】

好適には、各変数オブジェクトは、データソースからのデータを提供するための入力オブジェクト、データの行き先にデータを供給する出力オブジェクト、又は、１つの演算子オブジェクト又はもう１つの演算子オブジェクトからデータを流すための結合オブジェクトより選択される。好適には、結合オブジェクトは演算子オブジェクトの間のリンクとして表現される。好適には、各変数オブジェクトは変数ラベルを伴って提供される。好適には、各演算子オブジェクトは演算子ラベルを伴って提供される。

【００１１】

好適には、前記式の前記論理的な記述は前記オブジェクト間の論理的関係により定義される。好適には、オブジェクト間の前記関係の前記グラフィカルな表示を定義する前記式のグラフィカルな定義は記録される。

【００１２】

好適には、前記方法は、前記論理的な定義を記述する情報を記憶する工程を含む。好適には、前記方法は、前記グラフィカルな定義定義を記述する情報を記憶する工程を含む。

【００１３】

好適には、２以上の前記演算子オブジェクトがグループ化されており、グループ化は、グルーピングの演算子オブジェクトを定義し、その中においてはグルーピングの境界を横切りグループ内の演算子オブジェクトの入力に対して結合されている変数オブジェクトがグルーピングオブジェクトコンポーネントの入力となり、グルーピングの境界をクロスしグループ内の演算子オブジェクトの結果に結合された変数オブジェクトは、グルーピングの演算子オブジェクトの結果となる。

好適には、他のオブジェクトにリンクされていないグループ内の演算子オブジェクトの入力及び結果は、各々、グループのオペレータオブジェクトの入力及び結果となる。好適には、グループ化されたオブジェクトのグラフィカルな表現は、グループの演算子オブジェクトのグラフィカルな表現によって配置され、グループのコンテンツへのグラフィカルな表現は、グループのオブジェクトの表現へのリンクのグラフィカルな表現により配置される。

【００１４】

好適には、定義された式の論理的な定義は、グループの演算子オブジェクトの内容を含む。好適には、表示される全ての式のグラフィックの定義は、グループの演算子オブジェクトの内容を含まない。好適には、グループの演算子オブジェクトの内容は、式の全てのグラフィカルな表現とは分離されてグラフィカルに表現される。

【００１５】

好適には、変数オブジェクトは保持可能なデータのタイプを定義する属性を付される。好適には、演算子オブジェクトの入漁ｋ及び結果は、演算子オブジェクトが受け取ること及び生成することが各々予測されるデータのタイプを定義する属性を付される。

【００１６】

好適には、変数オブジェクトは、既に定義されており調停する演算子オブジェクトによって関係付けられている他の変数オブジェクトの属性から、属性を受け継ぐ。好適には、変数オブジェクトは、既に定義されており、関係付けられている演算子オブジェクトの入力又は結果の属性から、属性を受け継ぐ。好適には、演算子オブジェクトの入力または結果は、既に定義されており関係付けられている変数オブジェクトの属性から、属性を受け継ぐ。

【００１７】

好適には、既に付されており対応が関係付けられているオブジェクトの属性をチェックするステップを含む。

【００１８】

好適には、ラベル付けされた変数オブジェクトのライブラリが予め定義されている。好適には、ラベル付けされた変数オブジェクトのライブラリが予め定義されており、各ラベル付けされた変数オブジェクトの処理の方法、その生成のための入力及び出力もまた、予め定義されている。

【００１９】

好適には、変数オブジェクトの変数ラベルは、予め定義した変数ラベルのリストから選択される。好適には、各変数ラベルには、ラベルによりラベル付けされた変数オブジェクトが含むことができるデータのタイプが定義されている属性が付されている。好適には、変数ラベルの選択は、変数オブジェクトへのラベルに関連付けられた属性が付される。好適には、変数オブジェクトに付されている属性は、選択が有効なラベルの選択を制限する。

【００２０】

好適には、演算子オブジェクトは、少なくとも、加算、減算、乗算、除算、ルックアップテーブル及び条件付動作の１つである。好適には、演算子オブジェクトは、より複雑な演算子の実行にリンクされた多数の単純な演算子を含む多数の段階の演算である。好適には、一形態においては、演算子オブジェクトはデータベースのキュエリである。好適には、第１の演算子オブジェクトはデータベースへの書き込みを行う。

【００２１】

好適には、演算子オブジェクトの演算子ラベルは予め定義された演算子ラベルのリストより選択される。好適には、各演算子ラベルには、ラベルを付された演算子オブジェクトが受け取り又は提供することのできる入力又は結果の各々のデータのタイプを定義する属性が付される。好適には、演算子ラベルの選択は、演算子オブジェクトへのラベルに関連する属性が付されている。好適には、演算子オブジェクトに付された属性は、選択されることが可能なラベルの選択を制限する。

【００２２】

好適には、論理的な記述はランタイムエンジンにより使用され、定義した式の処理が行われ、各変数オブジェクトに提供されたデータは、演算子オブジェクトの入力にリンクされ、これにより、データは、式のオペランドとなり、演算子オブジェクトによって表現される各演算子は、式の演算子となり、演算子オブジェクトの各結果は、次の演算子又は式の最終的な結果となり、これにより、式の処理が実行され式の結果が得られる。

【００２３】

好適には、各変数に対して名前領域が定義され、これにより、変数オブジェクトにより表現される論理的な変数におけるデータは、名前空間内の変数オブジェクトの各発生と同じである。好適には、名前空間は、モデル化された式に対するデフォルトグローバルによる。好適には、論理的な結合は、名前空間内のラベルが付された変数オブジェクトの各発生の間に生成される。好適には、グラフィカルなリンクが表示され、ラベル付けされた変数オブジェクトの発生の間の論理的な結合が示される。

【００２４】

好適には、名前空間は各演算子オブジェクトに定義され、これにより、演算子オブジェクトにより表現される論理的な演算子の処理は、名前空間内の演算子オブジェクトの各発生と同じである。

【００２５】

好適には、グループ化された演算子オブジェクトは、１回以上グループ化された演算子オブジェクトの定義に使用され、式の論理的な定義に適用される。

【００２６】

好適には、ラベルの属性は、タイプ、単位及びディメンションを有する。好適には、グラフィカルな定義は、ＸＭＬにより記述される。好適には、論理的な記述は、ＸＭＬにより記述される。

【００２７】

好適には、各演算子オブジェクトは、演算子オブジェクトにより表現される演算子により実行される多数の演算の定義を含み、各定義は、演算子により処理されることができるデータの分離したタイプのためである。

【００２８】

好適には、演算子オブジェクトは、１つ以上の入力及び１つ以上の出力を有するコンポーネントとしてグラフィカルに表現され、そのコンポーネントは、表現される演算子に表示されるインジケータを有する。好適には、演算子オブジェクトは、結果を生成するための処理の方法の入力が未だ定義されていない演算子の表現である空コンポーネントである。好適には、前記空コンポーネントは、適切に定義された方法を有する適切な演算子オブジェクトをサーチするための基準を形勢するために使用される。

【００２９】

好適には、オブジェクトのライブラリが提供される。好適には、オブジェクトは、外部より供給される。

【００３０】
本発明の第３の観点によれば、本発明に係るグラフィカルに式を定義するシステムは、ディスプレイ及びユーザ入力手段を有するコンピュータと、少なくとも１つの入力を処理し少なくとも１つの結果を生成する方法を定義するための第１の演算子オブジェクトを提供する手段と、前記第１の演算子オブジェクトをグラフィカルな表現で表示する手段と、データを保持するための第１の変数オブジェクトを提供する手段と、前記変数オブジェクトを前記第１の演算子オブジェクトの前記入力の１つまたは前記出力の１つに関係付ける入力をユーザから受け付ける手段と、前記第１の変数オブジェクト及びその前記第１の演算子オブジェクトとの関係をグラフィカルな表現で表示する手段とを有し、これによりオブジェクト間の関係により式が定義される。
【００３１】
本発明の第４の観点によれば、本発明のコンピュータをコントロールすることにより式をグラフィカルに定義するコンピュータプログラムは、少なくとも１つの入力を処理し少なくとも１つの結果を生成する方法を定義するための第１の演算子オブジェクトを提供し、前記第１の演算子オブジェクトをグラフィカルな表現で表示し、データを保持するための第１の変数オブジェクトを提供し、前記変数オブジェクトを前記第１の演算子オブジェクトの前記入力の１つまたは前記出力の１つに関係付ける入力をユーザから受け付け、前記第１の変数オブジェクト及びその前記第１の演算子オブジェクトとの関係をグラフィカルな表現で表示し、前記オブジェクト間の関係の論理的な記述を記録し、これにより、オブジェクト間の関係により式が定義される。
【００３２】
本発明の第５の観点によれば、本発明に関わる記録媒体は、前述したコンピュータプログラムが記録されたコンピュータにより読み取り可能な記録媒体である。
【００３３】
本発明の第６の観点によれば、本発明のコンピュータにより実行されるグラフィカルに式を定義する方法は、少なくとも１つのデータを保持するための変数オブジェクトを提供し、少なくとも１つの前記入力データを処理し結果を生成する方法を定義する演算子を提供し、ユーザに結果の変数を選択するための変数のリストを表示し、入力データの処理の結果を記憶するための結果の変数の選択をユーザから受け取り、前記選択された結果の変数をグラフィカルな表現により表示し、演算子を選択するための演算子のリストをユーザに表示し、ユーザより演算子の選択を受け取り、選択された演算子のグラフィカルな表現を表示し、ユーザに前記変数か１以上の定数であるところの少なくとも１つの入力を選択させるため、入力データを記憶するための入力のリストを表示し、ユーザからの少なくとも１つの入力の選択を受け取り、前記選択された入力のグラフィカルな表現を表示し、これにより、選択された結果変数を、選択された処理による選択された入力の処理と等しくすることによって、式を定義する。
【００３４】
本発明の第７の観点によれば、本発明のコンピュータにより実行されるグラフィカルに式を定義する方法は、予め設定された属性を有する少なくとも１つの変数のタイプを提供し、少なくとも１つの前記入力データを処理し結果を生成する方法を定義する少なくとも１つの演算を提供し、ユーザに変数のタイプを選択させるため変数のタイプのリストを表示し、ユーザからの変数のタイプの選択を受け取り、選択した変数のタイプに対する名前を受け取り、名前が付された変数を表示し、ユーザに演算を選択させるための演算のリストを表示し、ユーザからの演算の選択を受け取り、選択された変数を選択された演算と協働させるためのユーザからの入力を受け取り、選択された変数を入力された変数か結果の変数とし、選択された変数が結果の変数とされた場合には、ユーザから少なくとも１つの入力変数か入力定数及びその入力変数又は入力定数に対する名前を受け取り、その入力された変数又は入力された定数をグラフィカルに表示し、選択された変数が入力の変数とされた場合には、出力変数の名前を受け取り、これをグラフィカルに表示し、これにより、入力変数又は入力定数による入力データの選択された演算による処理の結果により式を定義

し、結果の変数を得る。
【００３５】
【発明を実施するための最良の形態】
式は、１つ以上のオペランドに演算子を適用することによる結果の計算のための方法論の記述である。典型的には、オペランドは代数学上の観念における変数である。本発明は、オペランド（変数）及び演算子に関しての記述としての式を定義する。具体的には、変数及び演算子は関係がある。簡単な式Ｘ＝Ａ＋Ｂは、オペランドＡ及びＢと加算演算子（＋）との間の関係の記述である。本発明は、これがグラフィカルに表現されることを可能にするものであり、これは複雑なモデルや関数を表現するときに望ましい。本発明はまた、定義されたグラフィカルな関係の記述、換言すれば、式を定義する変数と演算子との間の関係を生成する。
【００３６】
本発明の方法は、グラフィカルの式定義ツール（ＧＦＤＴ）１２により実行される。図１に示すように、ＧＦＤＴ１２は、グラフィカルユーザインターフェイス（ＧＵＩ）１４と相互に作用する。ＧＵＩ１４は、コンピュータのオペレーティングシステムの一部を構成する。その一例は、マイクロソフトのウィンドウズ（Ｍｉｃｒｏｓｏｆｔ Ｗｉｎｄｏｗｓ）の種々の版、マッキントゥッシュブランドのコンピュータのためのＭａｃＯＳ、あるいは、ＵＮＩＸオペレーティングシステムの下で動作するＸ－Ｗｉｎｄｏｗｓである。ＧＦＤＴ１２は、グラフィックディスプレイに提供するための命令を提供するＧＵＩ１４と通信する。
【００３７】
図２に示すように、ＦＧＤＴ１２は、２つの主要部、コンポーネントマネージャ（Ｃｏｍｐｏｎｅｎｔ Ｍａｎａｇｅｒ）１６及びコネクションマネージャ（Ｃｏｎｎｅｃｔｉｏｎ Ｍａｎａｇｅｒ）１８を有する。コンポーネントマネージャ１６は、少なくとも１つの入力を処理し結果を生成する方法を定義するために提供される演算子オブジェクト２０を取り扱う。演算子オブジェクトは、式の中の演算子を示す物である。本例においては、演算子はコンポーネントとして参照される。従ってコンポーネントマネージャは好適な具体例の目的のためにコンポーネントオブジェクト（Ｃｏｍｐｏｎｅｎｔ Ｏｂｊｅｃｔ）を管理する。
【００３８】
コネクションマネージャ１８は、変数オブジェクトを取り扱う。変数オブジェクトは、式の結果の計算の各実例のための式の中のオペランドを示す物である。変数オブジェクトには多数の型が生じ、その主な１つは、コンポーネントにまたはコンポーネントからデータを流す、後により詳細に説明するような、結合オブジェクト（Ｃｏｎｎｅｃｔｏｒ Ｏｂｊｅｃｔ）２２である。変数オブジェクトの他の型は、一般的にラベルにより名前が付けられる入力及び出力オブジェクトである。これらは、好適な具体例において、名前付き結合として参照される。それらにはユーザが参照できるラベルが付付されるので、コンポーネントに又はコンポーネントから流されるデータについての情報を知るための、名前付き結合として参照される。
【００３９】
ＧＦＤＴ１２は、ユーザに対して、ＧＵＩ１４を介して、定義される際に式をグラフィカルに表現するインターフェイスを提供する。ＧＦＤＴ１２はまた、それが定義された時に、式の論理的な記述を記録する。グラフィックの記述の記録は論理的な記述の記録とは区別される。論理的な記述は、インターフェイスの中に表示されるオブジェクトの間の関係を論理的な用語により記述される。インターフェイス２４の画面のコピーを図３に示す。
【００４０】
インターフェイス２４は、標準的なＷｉｎｄｏｗｓのウィンドウであり、ツールバーブ２６、基本ウィンドウ２８及び補助ウィンドウ３０を有する。基本ウィンドウ２８は、総括的な式の部分を示すカレントページを表示する。ウィンドウ２６は、ページセレクタ３２を含む。その他のページは、式のほかの部分を示すように提供される。メインウィンドウ

２８は、アキュムレータのための簡単なモデルを表示して示されている。

【００４１】

式またはモデルを作成する方法を、図４Ａ～４Ｄを参照して記述する。この例は、採掘現場に関わる。採掘現場の計画、予算を組むこと又は監視は、予め定義された式に従った計算の結果を用いることにより行われる。式は、入力変数に含まれる情報が特定の演算子に従って処理され結果を生成したときに生成された結果を決定する。その結果から、採掘現場の運営の計画、予算を組むこと又は監視は実行される。具体的には、”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ”の結果が一例として使用される。”Ｌｏａｄ ａｎｄ Ｈａｕｌ”は、このケースでは、デフィアンス（Ｄｅｆｉａｎｃｅ）という名前の穴における、採掘穴からの鉱物に含まれる物質の積載及び運搬のコストである。

【００４２】

図４Ａに示すように、”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ”が計算されることが望まれている。マウスの移動によりポインタの位置を決め、クリックし、ドラッグし、所望の位置に変数オブジェクトを下ろすというよく知られている処理により、変数オブジェクト３４がツールバー２６より選択され、メインウィンドウ中に配置される。その変数オブジェクトには、”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ”という名称が付与される。使用されるシンボルに対する取り決めは、名称の付された変数（名称の付された変数オブジェクト）は円形の中、コンポーネント（演算子オブジェクト）は入力を表す内向きの矢印及び出力を表す外向きの矢印を伴う矩形、及び、データの流れの結合（変数オブジェクト）はラインにより表現する、というものである。

【００４３】

”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ”変数３４には特定の属性が付与されてもよい。これは、補助ウィンドウ３０中の”Ｐｒｏｐｅｒｔｉｅｓ”タブが選択され、これは数値でなければならないというような所望の属性、より具体的にはそれは通貨を示す数値でなければならずその通貨単位は”ＡＵ$”であるというような属性が入力されることにより行われる。一度生成されたその変数は、”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ”とラベルが付された円形３４として、ビデオディスプレイユニット上にグラフィカルに表現される。その変数のアイコンは、それが手入力のような任意の適切なソースからデータを受け取ってもよい、それは他のプログラムから受け取られてもよい、又は データベースから受け取られてもよい、ということを表現することになるだろう。

【００４４】

”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ”は、”Ｌｏａｄ ａｎｄ Ｈａｕｌ（精選）”データが加えられた”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ（バルク）”データから計算される。すなわち、”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ”はバルク物質及び精選物質の積載及び運搬のコストを表現する。バルク状物質は、鉱物が含有する物質及び非鉱物が含有する物質を生成する採掘方法から得られる。精選された鉱物は、上質な、鉱物が含有する物質を生成する採掘方法から得られる。

【００４５】

図４（Ｂ）に示すように、次のステップは、ツールバー２６からコンポーネント３６を選択すること及びメインウインドウ２８中に配置することである。通常、各コンポーネントのボタンの数又はリストはユーザに提供され、そこから、図３に示すような加算、減算、乗算、除算等から選択される。条件付き演算、及び、微積分、三角法等のその他のより複雑な演算等のコンポーネントは、ルックアップテーブルのようにして提供される。これに加えて、データまたはテキストを取り扱う演算も含まれる。実行可能な演算の型は制限されない。実際に多数のレベル化された演算を含む膨大な演算のライブラリが利用可能にされている。これについては、後により詳細に説明する。このケースでは、”加算”演算が選択されている。加算は、ボックス３６としてグラフィカルに表現される。加算は少なくとも２つの（このケースでは実際に２つの）入力及び１つの出力を有する。そして、オブジェクト間の関係が生成される。このケースにおいては、”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ”変数は加算の結果である。従って、結合オブジェクト３８をツールバー２６よ

り選択することによって、これらは結合され、加算コンポーネント３６の出力矢印を”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ”変数３４に接続するライン３８が引かれる。従って、”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ”は連携の特徴の効力によって結果変数となる。入力及び出力は矢印として表現され、連係は、結果の矢印と変数との間の結合３８として表現される。

【００４６】

図４（Ｃ）に示すように、もう１つの変数オブジェクト４０がツールバー２６より選択され、メインウィンドウ２８中に配置され、”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ（バルク）”の名称が付される。変数名は、初めのうちは、ユーザが所望の変数を選択するためのリストを生成するために入力される。その他の点としては、変数の名称は、要求されるように入力されてよい。”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ（バルク）”変数４０は、もう１つの結合オブジェクト４２がツールバー２６より選択されて２つのオブジェクトの間が結合されることによって、加算演算子３６の入力と連携される。このようにして、”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ（バルク）”は入力変数になる。”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ（バルク）”は円形４０として表現され、入力関係は、変数とコンポーネントの間の結合４２として表現される。

【００４７】

図４（Ｄ）に示すように、”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ（精選）”と名称が付されたもう１つの変数４４が選択され、配置され、加算演算子３６の他方の入力と連携される。この式の定義は、これにより完了する。結果の変数、コンポーネント（演算子）及び入力の変数の選択、及び、それらの間の関係は、次のように定義された式という結果をもたらす：「入力変数”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ（バルク）”及び入力変数”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ（精選）”は、加算演算子によって合算され、結果の変数”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ”が生成される。」。式が演算の実行に付された時には、入漁データ変数に入力されたデータが、定義された式に従って、結果変数に入力される結果を生成する。

【００４８】

定義した式を後の回復のために記憶するために、各オブジェクトの型、名称及び属性は、全て、ウィンドウ２８内の各オブジェクトの位置に対応して記録される。この情報により、式は、記憶し後の表示の際に回復することが可能となる。

【００４９】

ブロック図による式の描画と同時に、式の論理的な定義も記録される。再び図４（Ａ）を参照すると、変数オブジェクト”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ”３４ の生成は登録される。図４（Ｂ）においては、加算コンポーネント３６の生成もまた、その配置と同様に登録される。加算コンポーネントの出力と変数”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ”との間の結合３８の配置も登録される。そして、コンポーネントの結果の出力と結果の変数との間の論理的な結合が、記録される。

【００５０】

図４（Ｃ）に示すように、同様に、変数オブジェクト”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ（バルク）”４０及びその加算コンポーネントの第１の入力への結合が登録される。再び図４（Ｄ）においては、”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ（精選）”精選変数の生成及びその加算コンポーネントの第２の入力への結合も登録される。コンポーネント及びそれらの間の結合（及びこれによるそれらの関係）の登録は、これによって登録される。従って、式の論理的な定義が、オブジェクト及びそれらの関係の記述という形態で生成される。画面上の位置及びその他のグラフィカルな情報は、その論理的記述に対しては重要ではなく、単にグラフィックの記述の中に記録される。

【００５１】

そのグラフィックの記述は、ユーザ、ある意味で関わることのできるユーザ、に対する式の表示のために使用され、式の論理的な記述の記録は、式の処理エンジンによって、定義された式を使用に供するために使用される。ユーザは、論理的な記述に関わる必要はなく

、処理エンジンは、グラフィックの記述に関わる必要はない。このような単純な式を対象とするこの段階においては、グラフィックの記述及び論理的な記述は、概念のレベルにおいては実質的に異ならない。しかしながら、より複雑な式がモデル化されるに際しては、後述するように、これらの定義は異なってくる。しかしそれでも、ユーザはまだ、定義される式に対して、式のグラフィカルな表現によって知的なレベルで関わることができ、式の処理エンジンは、単にグラフィカルな表現に関連するのみの情報を除外しなければならないということ無しに、論理的な記述を使用することができる。

【００５２】

変数と演算子との間の連携は、演算子の他の入力の属性が少なくともある程度の範囲で決定されることを可能にする。例えば、結果の変数の演算子との連携は、このケースでは通貨を示す数値であるが、演算子の入力の属性が知られることを可能にする。”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ（バルク）”入力変数が入力の１つと連携されている時には、この入力変数に連携されている属性は、演算子の要求に対してチェックされる。代わりに、もし”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ（バルク）”入力変数がそれに連携された属性を持っていなければ、それはこれらの属性を受け継ぐことができる。従って、”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ”は通貨を示す数値であるという属性を持っているので、”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ（バルク）”及び”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ”の両方の変数もまた、通貨を示す数値であるに違いない。チェックが実行され、もしいずれかの入力が要求されている属性と一致しなかった場合には、警告が発せられるか、又は、この連携は許されない。また別の状態としては、もし入力が連携している属性を持っていなければ、その通貨を示す数値であるとの属性が受け継がれる。

【００５３】

チェックと継承は双方向に働くことができる。すなわち、もし入力が通貨を示す数値であるとの属性を持っているならば、結果もまたチェックされ、それが矛盾のない属性を持っているか否かを確かめられる。通常は、より最近に生成された入力／結果変数が、チェックされる。

【００５４】

コンポーネントの入力及び出力は、コンポーネントに関わる属性情報を保持する。コンポーネントは、入力から出力を生成するという機能とともに、その入力及び出力の属性により定義される。

【００５５】

図５において、”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ（バルク）”４０の値を計算する式は、入力変数”Ｂｕｌｋ Ｒａｔｅ”４８と入力変数”Ｂｕｌｋ ＢＣＭｓ”５０の乗算によって定義される。バルクレート（”Ｂｕｌｋ Ｒａｔｅ”）は、その場所における各１立方メートルのバルクの鉱物が含有する物質の採掘のためのコストを示す。バルクＢＣＭ（”Ｂｕｌｋ ＢＣＭｓ”）は、バルクの物質の（その場所に）堆積された立法メートル（ＢＣＭ）を示す。この式は、同様のステップを使用することにより再び組み立てることができ、先の式が定義される。”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ（バルク）”入力変数４０は、結果変数として使用可能である。それは、従って、選択されディｓプレイ上に表現される。もし、”Ｂｕｌｋ Ｒａｔｅ”及び”Ｂｕｌｋ ＢＣＭｓ”入力変数が、まだ使用可能でないならば、それらは入力される。乗算演算子オブジェクト４６が、次に、選択され、入力変数及び出力変数と連携される。これは、乗算演算子の表現がドラッグされて配置されることにより実行されてよい。入力及び結果変数は演算子に関係付けられる。従って、入力変数（”Ｂｕｌｋ Ｒａｔｅ”及び”Ｂｕｌｋ ＢＣＭｓ”）は演算子の入力に結合され、出力の矢印は結果変数（”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ（バルク）”）に結合される。ここで、変数及び演算子の配置の順番は、先に定義した式とは異なる。この順番は、属性のチェック及び継承の順番の相違のみををもたらす。もし、乗算の時に入力変数の属性が結果変数の属性と矛盾していれば、属性の継承に問題があるとのメッセージがユーザに提供される。すなわち、もし、バルクレート入力変数が容積当たりの価格を示す数値であるとの属性を持っていなければ、及び／又は、バルクＢＣＭ

入力変数が容積を示す数値であるとの属性を持っていなければ、警告メッセージが与えられるか、又は、もし２つのうちの一方が正しい属性を持っていれば他方に正しい属性が継承される。属性の解析と継承は、変数／定数の次元に限定される必要はない。数値の単位もチェック可能であり、例えば、もし一方の単位がＡＵ＄であり他方の単位がＵＳ＄である場合、これは警告という結果になる。これに代わって、後に記述するように転換を行ってもよい。

【００５６】

この式は、図４（Ｄ）の式と同じページに生成されてもよいし、新しいページに生成されてもよい。もし別々のページになるのであれば、適切なページを選択するために、ページセレクタ３２が使用される。これはまた、特に、それらが別々のページ上にあるならば、グラフィックの定義が論理的な定義から分離され始める場所でもある。各ページが各式に対する分離されたグラフィックの定義を含む場合には、論理的な定義は、図５の式の”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ（バルク）”結果変数と図４（Ｄ）の式の”Ｌｏａｄ ａｎｄ Ｈａｕｌ ＩＰＤ（バルク）”入力変数との間の結合を形成する。１つの式は、ユーザに対して、２つのより小さな式として表現される。これは、式の知的な理解を助けるが、論理的には、分離はない。これは、同じ名前が付されている変数が作用する特有の空間を有しているときに分離が生じることができないと言うことではない。変数（及びコンポーネント）に対する名称の空間は、それらのアプリケーションを限定し定義されることが可能である。これについては、後に詳細に説明する。

【００５７】

ここで、コンポーネントマネージャ１６及びコネクションマネージャ１８により行われる処理について、図６，７及び８を参照してより詳細に説明する。図６に示すように、コンポーネントマネージャ１６は、まず、５２において、ユーザにコンポーネントタイプを、ツールバー２６の中に表示されているパレットから選択させる。図３に示す例においては、加算、減算、乗算及び除算のボタンがコンポーネントとして選択されるように提供されている。これらのボタンの１つが押されると、オペレーティングシステムはコンポーネントマネージャ１６にその特定のボタンが選択されたことを通知する。演算子コンポーネントのタイプは知られている。そこで、ユーザは、５４において、描画ウィンドウ２８の中でクリックする。すると、クリックされた場所にコンポーネントが配置されるという結果をもたらす。これは、５６においてコンポーネントが描画ウィンドウ２８の中に描画され、５８においてコンポーネントが式の定義の中に登録されるということを必然的にもたらす。コンポーネントの名称、そのタイプ及びページ上の位置やコンポーネントを表現するためのアイコンの詳細のようなコンポーネントのグラフィカルな表現のための詳細な仕様が記憶される。論理的な定義においては、コンポーネントの名称やタイプのような詳細が登録される。

【００５８】

図７に示すように、名前の付された結合が式の中に含まれることになる時には、ユーザは、６０において、名前の付された結合コンポーネントのタイプを、パレットから選択する。６２において、ユーザが描画ウィンドウ２８をクリックすると、６４において、名称が空の結合変数が描画される。その後ユーザは、変数に名前を付したり、または待機した後で名前を付すことが可能である。もし、それに名前が付されたら、６６においてが、ユーザｓが名前を入力するための設備が提供され、その名前は、その後、６８において名称が付された新しい結合において表示される。その名称が付された結合は、その後、７０において登録される。名前、タイプ、ページ上の位置、名称が付された結合等の名称が付された結合のグラフィカルな表示に関わる詳細は、記録される。名称、タイプ及び名称が付された結合のような名称が付された結合の論理的な記述の詳細は登録される。

【００５９】

図８に示すように、コンポーネント間又はコンポーネントと名前が付された結合の間のような演算子オブジェクト間の結合は、図８に示すような関係で記述される。ユーザは、７２において、ツールバーより結合オプションを選択する。ユーザは、７４において、コン

ポーネント上をクリックし、名前の付された結合または他のコンポーネントに結合されている結合子をドラッグする。コネクションマネージャは、７６において、それから結合を許すか否かを結合されているオブジェクトの属性に基づいて決定する。もし、７８に”ｎｏ”で示すように、結合が許されなければ、８８において、描画されているラインの色を赤に変えることによって警告を発するか、あるいはまた、名前が付された結合の色が赤に変えられる。９０においてユーザがマウスを開放した時には、９２において描画は達成されず、従って結合は登録されない。もし、７８に”ｙｅｓ”で示すように結合が許された場合には、８０において、その名前が付された結合の色は緑に変化される。８２においてユーザがマウスを開放した時には、８４において結合を表現するラインが描画され、８６において結合は登録される。ラインの終了点やラインの頂点（曲がり目）のような結合（ライン）のグラフィカルな表現に関する詳細。結合されているコンポーネントの詳細のような論理的な記述に関する詳細は登録される。

【００６０】

図６，７及び８に記述された処理は、図面の各ページにおいて発生する。これに加えて、グラフィックの記述においては、各図面は記録された名称（ページ名）を有し、図面の各ページに対しては、各登録コンポーネント、名称が付された結合及び結合の詳細が記録される。これに加えて、ネームスペースの詳細が、後により詳細に説明するように、記録される。

【００６１】

図９を参照して、式の定義のもう１つの例を示す。この例においては、”Ｂｕｌｋ　Ｒａｔｅ”４８は、ルックアップテーブル９４によりいくつかの変数及び定数９６～１０４から定義される。ＰＩＴ９６は、採掘穴の名前を表す変数である。Ｓｃｈｅｄｕｌｅ９８は、採掘される鉱物の量に依存する採掘レートを表す変数である。ＲＬ１００は、鉱物が掘り出される採掘穴の深さの相対的なレベルを表す変数である。ＭａｔｅｒｉａｌＴｙｐｅ１０２は、例えばそれが新しい物質または沈殿した土砂であるというような、採掘された物質のタイプを表す変数である。Ｂｕｌｋ”Ｂ”は、文字定数である。ルックアップテーブル９４は、５つの入力の値に基づいて、値を調べる演算である。そしてその結果の値は、結果変数４８に提供される。図は、その入力及びその出力の間の結合を示す。入力、結果及び演算の選択及びこれらの画面上への表現の配置による関係の構築は記録される。これらのオブジェクト及びそれらの間の関係の論理的な記述はこの式を定義する。

【００６２】

図１０に示すように、このように大きく記述された例はトップダウンの方法論を使用し、いくつかの単純な式としてのモデルを定義している。これらの式は合わせられるか、又は、示されたの同様の１つの複雑な式として描画される。理解できるように、入力変数”ＰＩＴ”９６、入力変数”ＳＣＨＥＤＵＬＥ”９８、入力変数”ＲＬ”１００、入力変数”ＭＡＴＥＲＩＡＬ　ＴＹＰＥ”１０２、入力変数”Ｂｕｌｋ”Ｂ”１０４、入力変数”Ｂｕｌｋ　ＢＣＭｓ”５０及び入力変数”Ｌｏａｄ　ａｎｄ　Ｈａｕｌ　ＩＰＤ（精選）”４４が全て使用されて、最終的な出力である結果変数”Ｌｏａｄ　ａｎｄ　Ｈａｕｌ　ＩＰＤ”３４が計算される。輪郭をとって符号を付した領域１０６は、多くのステップが存在する式（モジュール）が連鎖されてより複雑な上のレベルの式を生成されることが可能であることを示す。破線のボックス１０６の内側のステップは、上位のコンポーネントを形成するためにグループ化され得る。上位のコンポーネントの入力は、”Ｘ”により示され、出力は小さいサークルにより示される。

【００６３】

上のレベルの式もまた、変数”Ｂｕｌｋ　Ｒａｔｅ”及び”Ｌｏａｄ　ａｎｄ　Ｈａｕｌ　ＩＰＤ（バルク）”の必要無しに、定義されることが可能である。その代わりに、演算子９４の出力が直接に演算子４６の入力に与えられ、演算子４６の出力は、演算子３６の入力に直接に与えられる。換言すれば、演算子は、直接的に連鎖されることが可能である。しかしながら、もし変数”Ｂｕｌｋ　Ｒａｔｅ”及び”Ｌｏａｄ　ａｎｄ　Ｈａｕｌ　ＩＰＤ（バルク）”がどこかよその場所で使用されている時には、この式の設計は、図１

０に示すように行うことがより適切かもしれない。

【００６４】

演算子の連鎖のコンポーネント化は、コンポーネント化される要素の周りのボックス１０６の描画及びコンポーネント化する機能の選択により行われる。ボックス１０６の中のオブジェクトは、その際、カレントページから消去されて、当たらしページに移される。新しいコンポーネントの内部の働きは、新しいページ上において観察可能である。コンポーネントのグラフィックの記述は新しいページにコピーされる。消去されたオブジェクトの場所には、図１１に示すように、新しいコンポーネント１０８が置かれる。コンポーネントのグラフィックの記述は、新しいコンポーネント１０８への入力を形成する消去されたコンポーネントへの各入力を伴って、カレントページの記述に含まれる。名前が付された結合９６～１０４、５０及び４４からの結合は、新しいコンポーネント１０８の入力に対応して生成される。新しいコンポーネントの出力からの結合は、名前が付されたコンポーネント３４に結合される。カレントページの記述は、新しいコンポーネント及びそれらに対する結合を反映してアップデートされる。新しいコンポーネントの入力の各々と、他のページにおいてグループ化されたコンポーネントの入力の各々との間には、論理的な結合が生成される。同様に、そのコンポーネントの出力と他のページのコンポーネントの出力との間には、論理的な結合が生成される。従って、このモデルの論理的な記述は、モデルのグラフィックの記述が異なるのに対して、実際には変更されずに維持される。

【００６５】

図１１において、図１０の１０６内の連鎖した演算子は１つにグループ化されて、より高いレベルの演算子１０８を提供している。この演算子は、５つの入漁を要求し、”Ｌｏａｄ　ａｎｄ　Ｈａｕｌ　ＩＰＤ”の結果変数を生成する。その後は、上位のレベルの演算子１０８を構成するより低いレベルの式の各々を再定義する必要無しに、新しいコンポーネント１０８は再利用されることが可能である。コンポーネントの働きを見せるための設備は提供されるかもしれない。これは、例えば、コンポーネントの内部の働きが見せられているページまでページがめくられることにより、内部の連鎖が表示されるまで、それがオープンされるように上位のレベルの式の上でダブルクリックをすることによってもよい。このドリルダウンと呼ばれる次に詳細なレベルを見るための処理である。

【００６６】

見てきたように、トップダウンの設計方法は、それらが矛盾なく継承されていることを保証するためのチェックがなされる各レベルの属性を伴う種々の式を定義するために使用され得る。同様にボトムアップの設計方法も採用可能である。これは、多数のレベルのモデルを生成することを許容し、これはグラフィカルに表現され、定義され得る。より上位のレベルの機能のモジュール式の組み立てもまた実行される。

【００６７】

図１２に示すように、継承は、選択が可能な変数／定数の選択肢を制限するために使用されることができる。すなわち、もし、もう１つの変数を選択することによって、選択された変数がある属性を持たなければならない場合、その変数の選択は、要求される属性を保持する変数に制限されるかもしれない。その他の変数は”グレイアウト”され、選択が不可能にされるか、又は、単純に選択肢のリストに表示されなくされる。継承される属性の一例は、ディメンションの継承である。この場合は、各変数は、例えば距離、時間、量等の少なくとも１つのディメンションを有しなければならない。

【００６８】

図１２において提供される例においては、定義される式は、Ａ×Ｂ＝Ｃである。変数には、タイプ、ディメンション及び単位を含む属性が与えられる。この場合、結果変数Ｃは、実数の変数であり、質量・距離のディメンション及びキログラムメータ（ｋｇｍ）の単位を有する。Ａは、実数で、そのディメンションは質量で、単位はキログラム（ｋｇ）であるという属性を有する。Ｂは、実数または整数のいずれかであるという属性により定義される。これは、Ｂは実数または整数のいずれかであるという継承の効果による、Ａ及びＣの属性の定義の結果である。もし、例えば、Ｃが整数として定義されており、Ａが整数と

して定義されていた場合には、これにより、Ｂも必然的に整数となる。これに加えて、Ｃは質量・距離（ｋｇｍ）というディメンションであり、Ａは質量（ｋｇ）というディメンションであるので、必然的に、Ｂは距離（ｍ）のディメンションを有することとなる。同様に、もし、Ａ及びＢが最初にＡがｋｇでＢがｍであると定義されるならば、ディメンション及び各入力の単位の効力及び演算子の効果により、単位Ｃは必然的にｋｇｍとならなければならない。

【００６９】

Ｂは、実数または整数でありディメンションはメータであるので、それらの継承した属性に適合する実際の変数のタイプは、Ｂである入力変数のタイプを制限する。以前に入力されたいくつかの変数のタイプをリストにしたプルダウンメニュー１１０が示される。属性に適合する変数のタイプは、通常のフォントによって示され、属性に適合しない変数は、”グレイアウト”されて示される。もちろん、この代わりに、選択が許されない変数のタイプは、単純に表示されないようにしてもよい。

【００７０】

この場合、変数タイプ”ＳＨＡＦＴ　ＤＩＡＭＥＴＥＲ”又は”ＬＥＶＥＲ　ＡＲＭ　ＬＥＮＧＴＨ”が選択され得る。プルダウンメニュから選択されたこれらの変数のいずれかが、その後変数Ｂの変数タイプとなる。選択され得る変数は、データベース又は変数のライブラリ等の種々のソースから獲得され、手入力されることはない。プルダウンメニュは、図示のごとく、原文通りの変数名のリストを含む。しかしながら、プルダウンメニューの中においては、変数のアイコンの表現も使用可能である。その他の適切な選択手段を使用するようにしてもよい。

【００７１】

同じ処理は、Ａのようなその他の入力変数、同様にＣのような結果の変数にも適用可能である。変数の選択の順番が、その後の変数（又は定数）の属性を必然的に決定する。

【００７２】

式の定義の進行をチェックするために、定義される式の一部分の上のポインタが配置された場所に設備が含まれ、そのような大きく定義された式を表示するウィンドウが出現する（図１３（Ａ）及び図１３（Ｂ）に示すように）。

【００７３】

図１３（Ａ）においては、ポインタはルックアップテーブルの入力のいずれか１つの上に配置されているということがわかる。ポインタの下には、入力の属性は０以上の数字であるグレード（”Ｇｒａｄｅ”）である必要があることを示しているボックスが出現する。グレードはメートルトンあたりのグラムの属性で与えられるということもわかる。

【００７４】

図１３（Ｂ）においては、ポインタはスケジュール結果変数をさしていることが示される。それは、式の定義をそのように大きく示している。このケースにおいては、式の定義は、もし（ルックアップテーブル：変数の中のデータを伴った”ｍｏｎｔｙｌｙ　ｃｏｓｔ　ｐｅｒｉｏｄ”：”ｐｅｒｉｏｄ”が”ｍｏｎｔｈｌｙ　ｃｏｓｔｓ　ｌｏｗｅｒ”値と等しい）であれば、結果は低い方の値（Ｌ）であり、さもなければ結果は高い方の値（Ｕ）である。これはまた、入力された式のシンタックスをチェックすることも可能である。

【００７５】

図１４に示すように、この例においては、はさみの圧力の式が定義されている。はさみの圧力（”ＳＨＡＲＥ　ＳＴＲＥＳＳ”）＝（半径”ＲＡＤＩＵＳ”×曲がりの角度（”ＡＮＧＬＥ　ＯＦ　ＴＷＩＳＴ”）×（はさみの弾性力”ＳＨＡＲＥ　ＭＯＤＵＬＵＳ　ＯＦ　ＥＬＡＳＴＩＣＩＴＹ”））÷長さ（”ＬＥＮＧＴＨ”）。この例においては、表示されるオブジェクトを表現するアイコンが異なる。はさみの弾性力１２０は、入力を受け取るルックアップテーブル演算子である。それは、金属合金入力変数１２２であり、その中においてデータは受け取られる。ルックアップテーブルは、ＧＦＤＴの中に記憶されているデータを参照するコンポーネントである。あるいは、ルックアップテーブルは、外部

データを参照するコンポーネントである。例えば、外部データは、スプレッドシートの形態であってよい。ＧＦＤＴは、データが外部のソフトウエアアプリケーションから転送されることを可能にするプラグインにより提供されることも可能である。標準的なスプレッドシートアプリケーションは、マイクロソフト・エクセルである。ＧＦＤＴは、プラグインを介して、エクセルスプレッドシート内のデータを検索するために、エクセルと通信することができる。ルックアップテーブルは、各金属合金のはさみ弾性力に関しては、物質供給者によって供給されるデータから得られる。データは、いくつかの物質供給者から提供されてよい。データは、物質供給者によって提供されたデータベース１２４から提供されてもよい。データベースは、インターネットのようなコンピュータネットワークを介してアクセスされるようにしてもよい。従って、演算子１２０はデータベース１２４をアクセスするデータベースキュエリを伴ってもよい。データベース１２４は、分散データベースであってもよい。したがって、グラフィカルに式を定義する方法は、データベースキュエリは本方法に従って定義される式の特定のであるとして、データベースキュエリを定義するためにも用いられてもよい。

【００７６】

図１５に示すように、この例においては、結果変数Ｅは、Ｅ＝Ｃ（Ａ×Ｂ）＋Ｄとして定義され、Ｃは他の関数である。もし、Ｃが定義されていなければ、これは空コンポーネントと呼ばれる。Ｃの属性は、図１５に示す式の他の属性により定義されてよい。すなわち、変数Ａ，Ｂ，Ｄ及びＥが、ある程度までは、Ｃが保持しなければならない動作の入力及び出力という属性を定義する。Ｃは徐々に定義されることも可能であり、その結果、もし図１６に示すようにＣに対してもしさらなる入力”ｆ”、”ｇ”及び”ｗ”を加えることが望まれた場合、式が徐々に定義されるように、これらを式に加えられる。新しい入力をコンポーネントＣに加えることが望まれている時には、入力をコンポーネントに加えるための選択が選択され、新しい入力及び出力が要望されるように生成される。このように、更なる名前が付された変数”ｆ”、”ｇ”及び”ｗ”が、ここでコンポーネントＣの各入力に結合されることが可能である。道央に、追加の出力を要求されたように付け加えることが可能である。

【００７７】

コンポーネントＣの機能を明記することが望まれた時には、図１７に示すように次のレベルへのドリルダウンによって、Ｃがオープンされることができる。Ｃの機能のもう１つのレイヤーはボックス１２６の中に示されるように定義される。あるいはまた、Ｃの機能は、コンポーネントのライブラリから描画されてもよい。基本ライブラリがＧＦＤＴには提供されてもよいし、あるいはまた、ライブラリはオンラインにより提供されてもよい。一度、要求を実行するコンポーネントが発見されたら、それは、式の中に挿入される。図１８に示すように、異なるマシン上に存在するコンポーネントが発見されたら、それはコンポーネントの要求される属性を果たし、要求される機能をコンポーネントに付加されている記述のように実行する。そして、これは、インターネットのようなコンピュータネットワークを介してＧＦＤＴのローカルなインスタンスに対して、生成されている式に挿入するために、転送されることができる。

【００７８】

図１６を参照すると、ＣはＡ及びＢの生成物の機能であり、また、入力ｆ、ｇ及びｗを受け取ることがわかる。すなわち、Ｃ＝（Ａ×Ｂ、ｆ、ｇ、ｗ）である。図１７を参照すると、Ｃがドリルダウンした時、Ａ及びＢの生成物は結合子１２８によって借りの変数”ｐ”に結合されていることがわかる。変数”ｐ”は、そこで、統合演算子”＝”によって示されるように、入力変数”ｗ”と比較される。”ＲＥＳＵＬＴ”演算子は、比較の結果が真かどうかをテストし、その場合には、入力”ｆ”及び”ｇ”を受け取るルックアップテーブルの結果が提供される。一方、もし、比較の結果が偽ならば、結果は０である。”ＲＥＳＵＬＴ”演算子の結果は、その後、乗算演算子（”ｘ”）によって示されるように、定数”ｋ”と乗算され、出力１３０が提供される。これは、そして、加算演算子（”＋”）の丹生ロクに提供され、そこで変数Ｄと加算され、結果Ｅが提供される。

【００７９】
空コンポーネントは、いっぱいに定義されているコンポーネントのプレイスホルダとして提供されることができる。空コンポーネントは、まだ、定義されたその入力と出力との間の機能を持っていない。ユーザは、空コンポーネントを設計環境の中に配置し、その入力及び出力を定義する。コンポーネントの機能は、その後に要求されたように定義されることが可能であるか、又は、入力及び出力の定義は、機能を実行可能なコンポーネントをライブラリから探すときのその探索基準を形成することができる。探索基準は、さらに、キーワード、より上位のクラスの構造情報当のように提供されることができる。空コンポーネントは、さらに、トップダウン設計方法の助けとなる。

【００８０】
グラフィカルな表現を支援するために、カラーコーディングを使用することは望ましい。例えば、１つのカラーは、ブルーとし、変数を表現するできる。もう１つのカラーは、グリーンとし、演算子を表現するのに使用する。入力は薄い影付きのライトブルーであり、出力は深い影付きのダークブルーである。これは、式の表現の視覚的な認識の支援となり、特に、複雑で多数レベルの式が表現される時の支援となる。その他の視覚的な表現としては、図中で使用されていたように、アイコンが変数、定数及び演算子を表現するために使用される。

【００８１】
ＧＦＤＴモデルのオブジェクトの多くは、ユーザが観察できて、さらに／又は、変更できるような属性を伴って提供される。図１９に示すように、アキュムレータコンポーネントはエクセルスプレッドシート１３２から受け取った入力を蓄積するように定義される。スプレッドシート１３２から分離した出力は、演算子１３４において、最新のサブトータル（ランニングトータル）に加えられ、メモリ演算子１３６により記憶される。そして、スプレッドシートからの最後の列のデータの時に、ランニングトータル変数１３８からの結果は、ゲート１４０によって、合計変数１４２にゲートされる。そしてこれは別のスプレッドシート１４４に渡る。１４６結合子は、高輝度で示される。補助ウィンドウ３０に、高輝度オブジェクトの属性が示されている。その高輝度結合子１４６の属性は、その名前が”コネクタ１”であり、それが数字タイプであり、それは長さのディメンションのためにメータを単位として提供されるということである。ウィザードが、属性の選択を支援するために、Ｄｅｌｐｈｉ又はＶｉｓｕａｌ　Ｂａｓｉｃのような製品で提供されるものと同様の方法により提供される。いくつかの予め定義されたデータのタイプが、彼らの必要性に応じてユーザによってさらに拡張可能なように提供される。あるいは、ユーザは、拡張したデータタイプのライブラリを購入する又は獲得するようにしてもよい。例えば、複素数のデータタイプは、複素数の実数及び虚数である２つの独立した数字として表現され、また、他の例においては、エンジンの出力の一般的なデータタイプのサブコンポーネントとして、エンジンの出力はパワー、トルク及び角速度によって表現されるかもしれない。

【００８２】
図１９において、ユーザはフィールドの値を編集することによって、または、ドロップダウンリストから選択を作成することにより、属性の値を簡単に変更することができる。

【００８３】
複素数データのタイプは、また、それらに対して実行される動作を定義する必要がある。演算子は、モデルによる表現として、存在するコンポーネントを使用する。例えば、複素数の加算は、実数の加算とは異なるセットの演算子を要求する。それらはともに”プラス（ｐｌｕｓ）”コンポーネントによって表現されるが、コンポーネントがそのデータタイプを取り扱う方法は、そのデータタイプの本質に依存する。コンポーネントへの結合がなされる時には、これまでに結合子のデータタイプ、又は、演算子又は名前が付された結合の入力又は出力の属性が存在している場所においてはネゴシエーション処理が行われ、データタイプの結合は矛盾ないものでなければならない。このように、各オブジェクト間のネゴシエーション処理を通して、正しいデータタイプが選択される。

【００８４】
図２０に示すように、加算コンポーネント１４８は、データタイプに依存して加算のいくつかの方法を実行することができる。これは、データのオーバーローディングとして知られている。データタイプは、テーブルに示すような異なるデータのタイプを処理するための固有の方法により提供されるが、異なるデータタイプを処理するための加算の方法が提供される。追加の定義を加えるためのフォーマットを定義する予め定義された図表が提供されているので、さらなるデータタイプを追加しコンポーネントにより処理されることが可能である。
【００８５】
特有のタイプの単位でデータが提供される場合においては、多くのケースにおいてしばしば、変換のファクタが要求される。例えば、時間という同じディメンションにおいて、１つの単位が秒でありもう１つの単位が分で有る場合等である。そのときに変換が要求される。変換のファクタは、例えば時間当たりのキロメータのスピードを、分当たりのメータに変換する時にも要求される。また、ディメンションの定義は、単位の基本的なディメンションの組み合わせを要求する。基本的な定義は、長さ、時間、質量、金額等である。例えば、加速度は、長さ×時間の二乗である。その他の属性もまた、オブジェクトに対して提供される。その他の属性の例は、セキュリティ情報（その情報を使用することのできるユーザのタイプや暗号化情報）、バージョン情報（バージョン番号及びそのバージョンが有効な期間）、証明書情報（保証されたソースから来たものであることを特定するデータタイプ機能）、課金情報（使用に対する支払いのため及びデータ又はオブジェクトに対する署名のためのアクセス）、位置情報（データやオブジェクトの場所のためのＩＰアドレスやファイル名等）及びブローカ情報（コンポーネントの管理者の情報）を含む。
【００８６】
モデル、名前が付された結合及びコンポーネントの定義の各々は、それ自身の名称のスペースを有する。これは、各モデル又はコンポーネントの定義において使用されている各オブジェクトの名前は、それらのモデルまたはコンポーネントに対してユニークであることを意味する。共通の名前を使用している異なるモデル又は異なるコンポーネントの定義、そして特に名前が付された結合のオブジェクトは、同じオブジェクトではない。しかしながら、名称のスペースは変更可能である。これは、変数の名前は特定の範囲内においてのみ適用されるという、多くのプログラム言語におけるローカル変数と同種である。これは、同じ名前を有するものの関係のない２つのオブジェクトが相互に混同されることを防ぐことになる。
【００８７】
いくつかの動作は非本質的な外部のものである。これは、それらは、式に関わるエンジンの外部で実行されるということを意味する。それらは、通常アプリケーションに固有のプラグインを通して、外部のコンポーネントを、任意の入力の要求とともにコールすることにより実行される。結果は、さらなる式の処理のために、（そのプラグインを介して）そのエンジンにフィードバックされる。非本質的な外部のコンポーネントは、コンポーネントタイプの定義ファイルを読み込むことによって、ＧＦＤＴ内において有効にされる。そしてこれは、ＧＦＤＴによる使用のための定義をコンポーネントに対して提供する。コンポーネントタイプは、標準的には、入力及び出力の結合子のセットを有し、コンポーネントがコンポーネントタイプに応じて生成される。
【００８８】
もし名前が付された結合がコンポーネントの外部の任意の場所で使用されるならば、１つのグループのコンポーネントのコンポーネント化の時には、コンポーネントの外部に導かれるすべての結合は、名前が付されたコンポーネントに対する入力及び出力オブジェクトを生成し入力する。もし、名前が付された結合がコンポーネントの内部のみで使用されるならば、その名前が付された結合の名前のスペースはコンポーネントとなる。これは、もはやコンポーネントの外部においては有効ではない。この方法は、詳細を隠しユーザにモデルを組み立てる時により構造的なアプローチを行うように仕向ける手段を提供する。複

雑なコンポーネントは、十分に定義されたインターフェイスを有する機能ブロックが効果的である。ユーザは、定義されたインターフェイス及び機能ブロックが提供された時に、”スパゲッティ・コード”なモデルを組み立てるのが難しいことに気づく。

【００８９】

いくつかのコンポーネントは、マイクロソフト・エクセル、ワードスプレッドシートのように、非本質的な外部のコンポーネントである。これは、ＧＦＤＴコンポーネントの中に含まれることとなり、モデルの定義の時に使用される。定義が実行され式が計算される時に、実際にエクセルスプレッドシートが使用される。エンジンは、データをコンポーネントに対して及びコンポーネントから流すために、スプレッドシートとエクセルを介して通信を行う。リモートコンポーネントは、メインのモデルとは異なるエンジンで実行する。ローカルな機能エンジンの支店からは、これらは、内部の働きがわからないブラックボックスである。

【００９０】

コンポーネント等の２つの要素の間で結合が要求された場合、名前が付された結合がページ上に配置され、ラベルが割り当てられる。そして、その名前が付された結合に属性が割り当てられる。名前が付された結合は、この段階で他のオブジェクトに結合される必要はないが、ほかの名前が付された結合オブジェクトが設計中のページに配置されて、同じラベルがドロップダウンリストから選択されることによって、その他の場所で使用可能となる。これは、新しいオブジェクトを生成しているのではなく、存在している同じオブジェクトに２番目のインスタンスを許したことになる。提供された２番目のインスタンスは、最初のものと同じ名前スペースの中となる。図２１に示すように、２つの名前が付された結合の論理的な結合が形成されると、その結果、コンポーネントのような他のオブジェクトへの結合子が名前が付された結合に結合された時に、名前が付された結合の全てのインスタンスは、コンポーネントの属性を隠すことが可能となる。同様に、ある場所の名前が付された結合にデータが提供された時に、論理的な結合のために、それは他の場所の名前が付された結合にも提供される。さらに、その他の論理的な結合のインスタンスは、属性を他のコンポーネントまたは名前が付された結合に関係のある他の結合に転送する。標準的には、名前が付された結合は、モデルの中間値と同様に入力されるデータ及び出力されるデータの確認のために使用される。名前が付された結合は、モデル内の多くの場所に現れる伝統的なソフトウエアプログラム言語における変数と同種であり、コードの実行の間、変化する値を伝達する。名前が付された結合、又は、演算子の入力への数値の割り当ては、ただ１箇所において行われる。なぜならば、その値は、名前が付された結合の他のインスタンスに流されるからである。従って、ただ１つの出力の結合が、演算子又は名前が付された結合の入力に結合することが許される。

【００９１】

図２２を参照して、コンポーネントＡがあるページの名前が付された結合に出力を提供し、これにより、ほかのページの名前が付された結合がコンポーネントＢに入漁ｋを提供するときには、効果は、コンポーネントＡの出力とコンポーネントＢの入力の論理的な結合が生成されるということである。論理的な結合は、モデルの種々のコンポーネントを介したデータの通信の本質である。論理的な定義に関しては、名前が付された結合は、これは純粋にコンポーネント及び結合の間の網の目状のネットワークであるため、無意味である。

【００９２】

２つのオブジェクトの結合の間においては、チェック、照合及び採択が行われる。例えば、もしディメンションが両方の結合に対して定義されていた場合、結合が許可されるためには、それらは同じでなければならない。単位は、変換ファクタが決定されることができる限りにおいては、位置する必要はない。もし一方の端部で属性のいずれかが定義されていない場合は、属性が矛盾ないものであることを維持するために、それらが採択されるか、又は、異なる属性が破棄される。一度、結合の間で矛盾のないデータタイプが調整されたら、コンポーネントは、データタイプに対して有効に動作する機能を保持していること

を確認するためのチェックが実行される。もし、保持してない場合には、次に、任意の変換手段が、矛盾なく互換性のあるものに変換するか否かを確認するチェックを行う。例えば、数値は、テキスト文字列に変換する必要がある等である。

【００９３】

出力は、多くの入力に結合されてよく、したがって２つの結合の間のネゴシエーションは単純ではないということに注意すべきである。さらなる入力が結合のセットに付加される各瞬間に、データタイプに矛盾がないことの確認を行うための、さらなる調停が実施され、もし必要であれば、データタイプが新しい結合に適合するものに変更される。コンポーネントマネージャは、有効で過負荷な機能で、適切なデータタイプとともに動作するセットをサーチする。これに加えて、有効なデータ変換手段がチェックされ、データが受け入れ可能なデータタイプに変換可能であるか否かの確認がなされる。もし、これが不可能であれば、コネクションマネージャは、結合が結合するコンポーネントと再度調停をするよう試みる。これは、全ての結合の属性を、モデル全体において再度調停する状態に導くかもしれない。しかしながら、ユーザは、調停の続行が許される新結合からの距離を定義することにより、この処理の拡大を制限することができる。

【図面の簡単な説明】

【図１】

図１は、本発明の方法を実行する、グラフィカルな式定義ツールを含むシステムの概略図である。

【図２】

図２は、図１のグラフィカルな式定義ツールの概略図である。

【図３】

図３は、図１及び２におけるグラフィカルな式定義ツールにより生成されたウィンドウの画面のコピーである。

【図４】

図４（Ａ）は、好適な実施例におけるグラフィカルな式の表現の第１のステップの概略図であり、図４（Ｂ）は、グラフィカルな式の表現の第２のステップの概略図であり、図４（Ｃ）は、グラフィカルな式の表現の第３のステップの概略図であり、図４（Ｄ）は、グラフィカルな式の表現の第４のステップの概略図である。

【図５】

図５は、グラフィカルに表現された式の概略図である。

【図６】

図６は、コンポーネント（演算子）オブジェクトの生成のステップを示すフローチャートである。

【図７】

図７は、名前が付された結合（変数）オブジェクトの生成のステップを示すフローチャートである。

【図８】

図８は、結合オブジェクトの生成のステップを示すフローチャートである。

【図９】

図９は、グラフィカルに表現された式の概略図である。

【図１０】

図１０は、図４（Ｄ）、５及び９において定義され、統合されて上位のレベルの式が形成されたしいのグラフィカルな表示を示す図である。

【図１１】

図１１は、個々の段階は示さないように短縮した図１０に示した上位のレベルの式のグラフィカルな表示を示す図である。

【図１２】

図１２は、変数属性の継承による変数タイプの制限のさらなる例を示す図である。

【図１３】

図１３（Ａ）は、ルックアップテーブルコンポーネントへの入力の属性を示すポインタを伴う式の概略図であり、図１３（Ｂ）は、名前が付された結合への入力の属性を示すポインタを伴う式の概略図である。
【図１４】
図１４は、式への入力変数を伴うグラフィカルに定義された式の概略図である。
【図１５】
図１５は、演算子を含むほかのグラフィカルに定義された式の概略図である。
【図１６】
図１６は、図１５の機能Ｃがどのように変更されたかを示す図である。
【図１７】
図１７は、図１２の機能Ｃが追加の詳細を追加されるかを示す図である。
【図１８】
図１８は、第１のコンピュータからコンピュータネットワークを介して第２のコンピュータに、コンピュータを通ってコンポーネントが通過する状態を示す概略図である。
【図１９】
図１９は、図２に示したグラフィカルな式定義ツールのほかの画面を示す図である。
【図２０】
図２０は、演算子により実行されるオーバーフローの方法を示す概略図である。
【図２１】
図２１は、異なるページ上の名前が付された結合オブジェクトの間の論理的な結合を示す概略図である。
【図２２】
図２２は、異なるページ上の名前が付された結合オブジェクトの間の論理的な結合及び異なったページ上の機能Ｂの入漁への機能Ａの出力の概略図である。
【図２３】
図２３は、本発明を説明するための図である。
【図２４】
図２４は、本発明を説明するための図である。
【図２５】
図２５は、第１の付録の前半部分である。
【図２６】
図２６は、第１の付録の後半部分である。
【図２７】
図２７は、第２の付録である。
【符号の説明】
１２…ＧＦＤＴ
１４…マイクロソフトウィンドウズＧＵＩ
１６…コンポーネントマネージャ
１８…コネクションマネージャ
２０…コンポーネントオブジェクト
２２…コネクションオブジェクト

【図１】

図 1

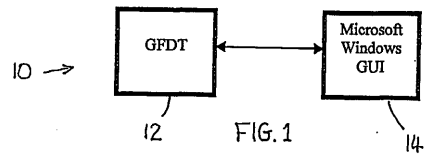


FIG. 1

【図２】

図 2

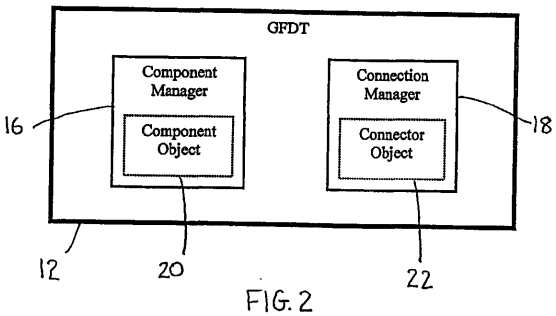


FIG. 2

【図３】

FIG. 3

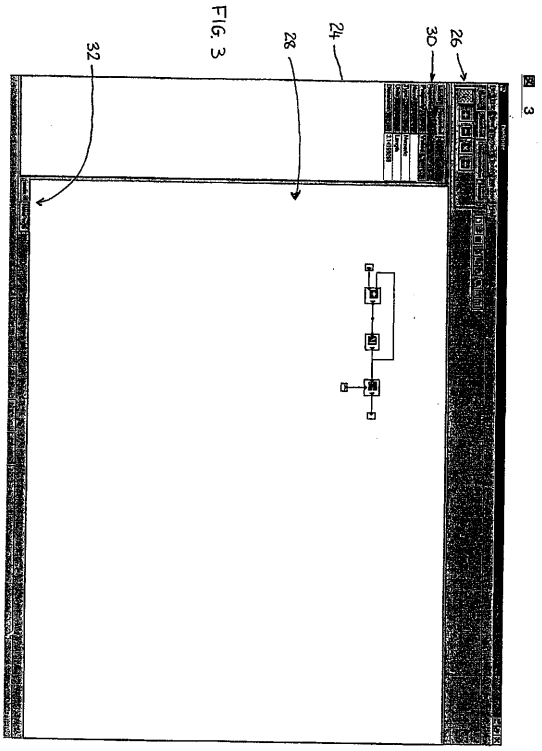
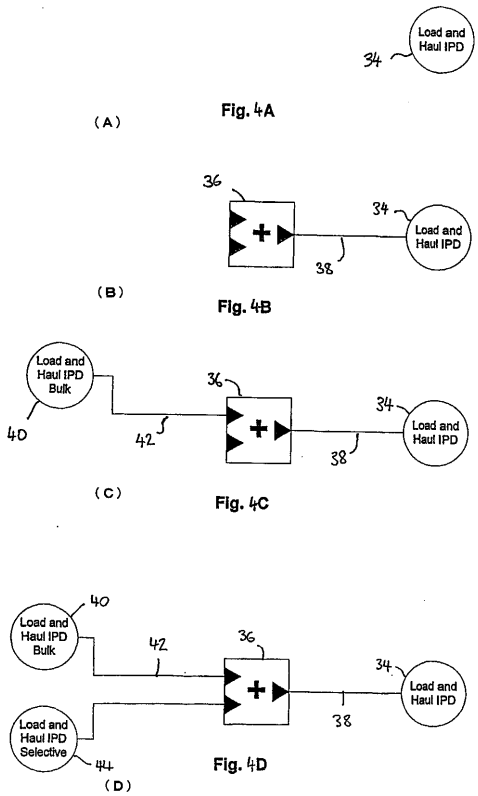


図 3

【図４】

図 4



【図５】

図 5

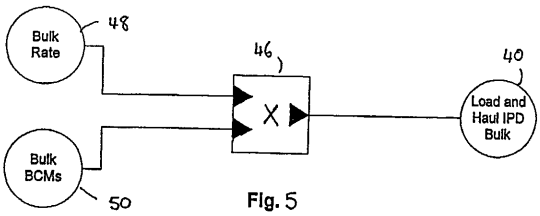


Fig. 5

【図６】

図 6

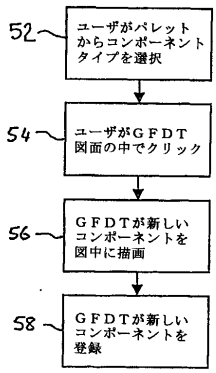


FIG. 6

【図７】

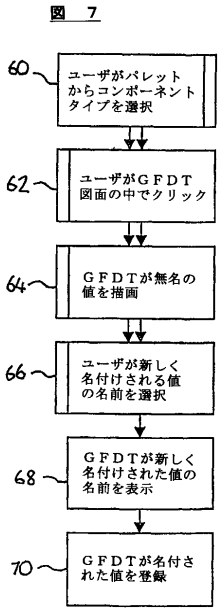


FIG. 7

【図８】

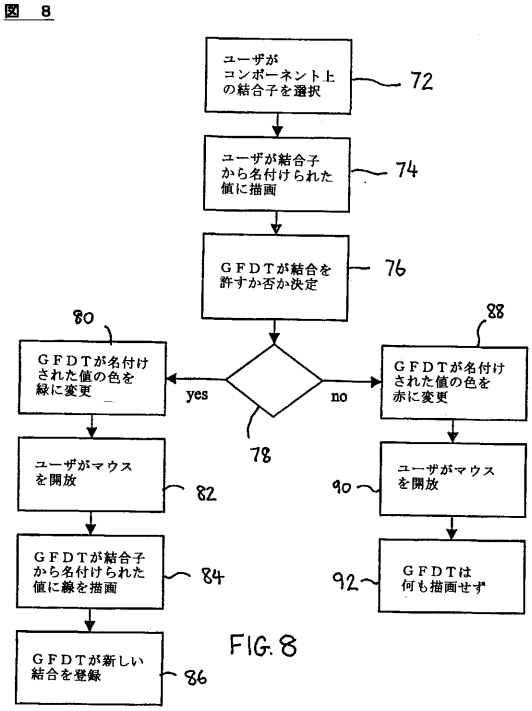


FIG. 8

【図９】

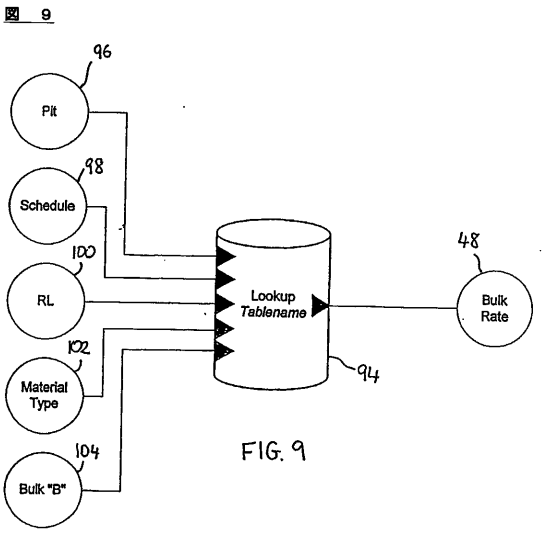


FIG. 9

【図１０】

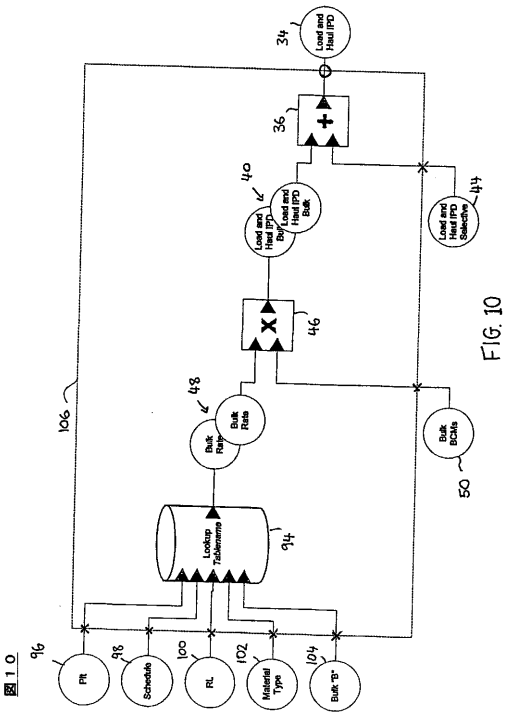


FIG. 10

【図１１】

図１１

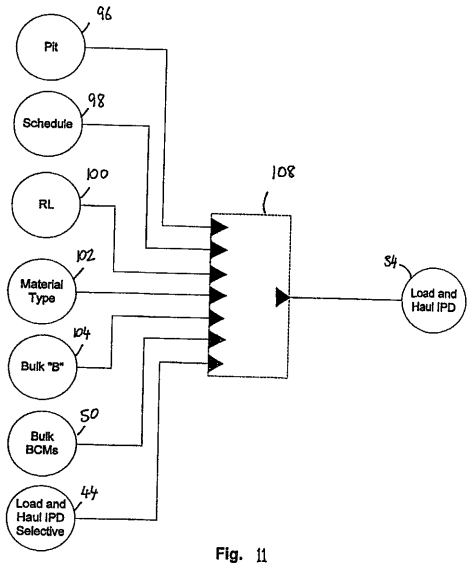


Fig. 11

【図１２】

図１２

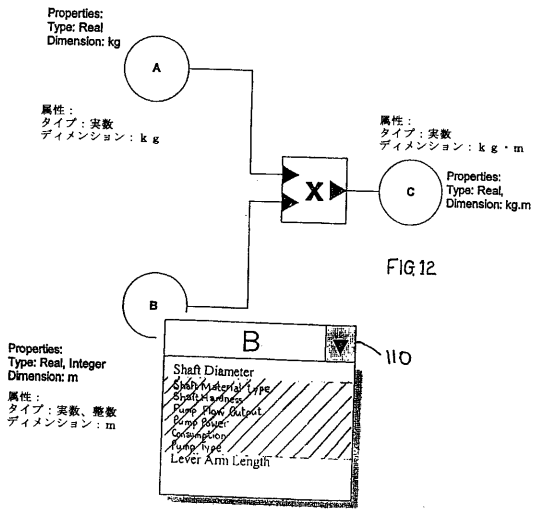


FIG. 12

【図１３】

図１３

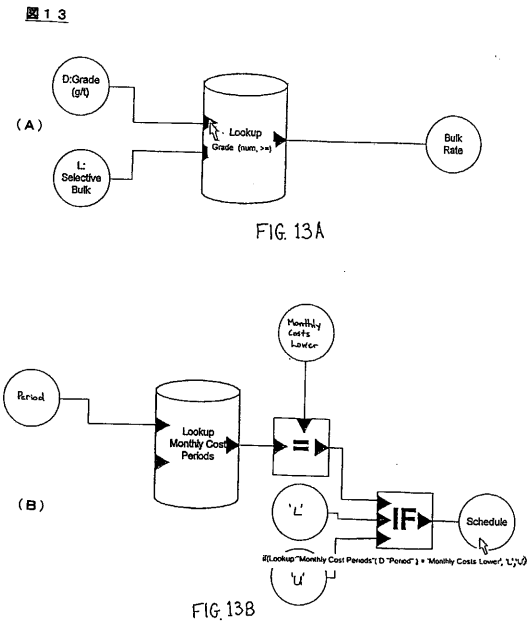


FIG. 13A

FIG. 13B

【図１４】

図１４

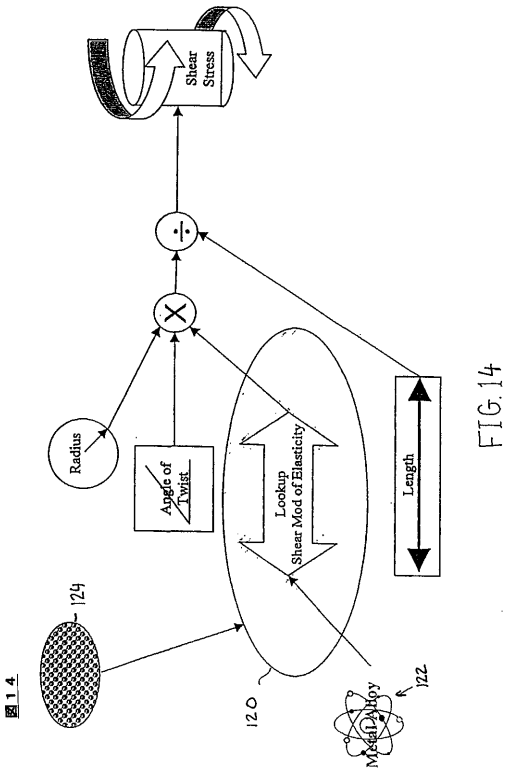


FIG. 14

【図１５】

図１５

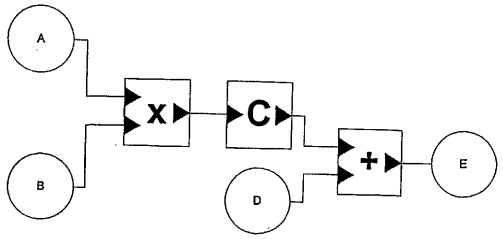


Fig. 15

【図１６】

図１６

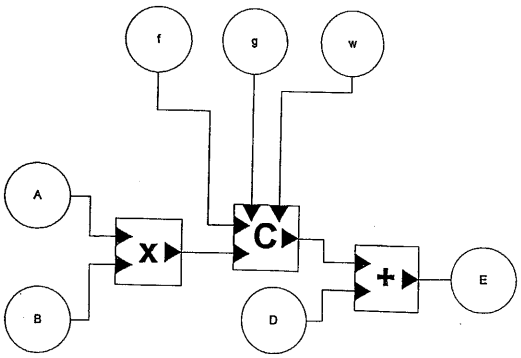


Fig. 16

【図１７】

図１７

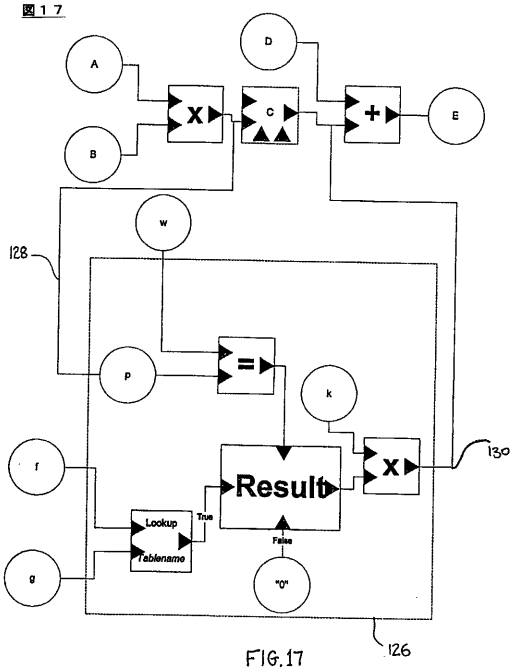


FIG.17

【図１８】

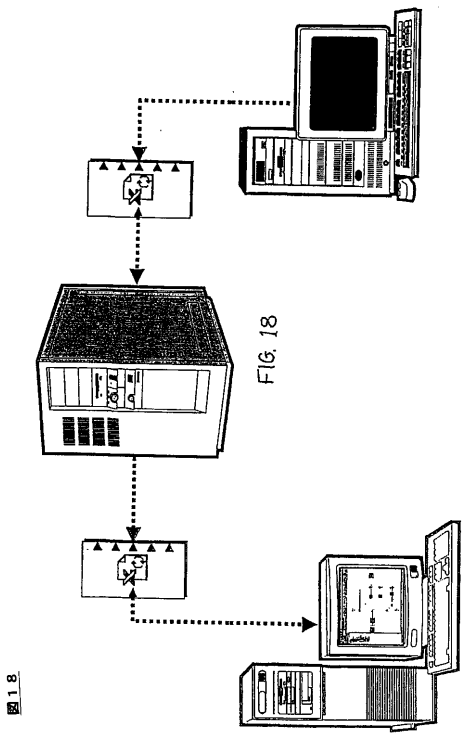


FIG. 18

図１８

【図１９】

15/18

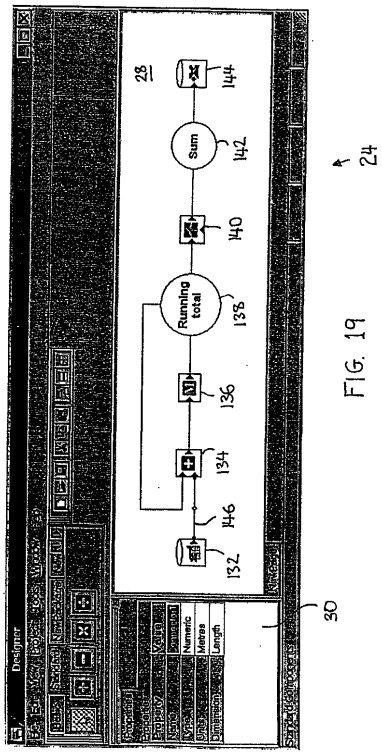


FIG. 19

図１９

【図２０】

図２０

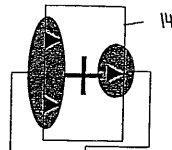


FIG. 20

Methods For "+" Component			
Inputs	Output	Address for Method	
All Numeric	Numeric	Address 1	Arithmetic add
All Text	Text	Address 2	Concatenation of Strings
Numeric/Text	Numeric	Address 3	Converts String to Numeric (if possible) and then arithmetic add
Numeric/Text	Text	Address 4	Converts Numeric to String (if possible) and then concatenates the strings

【図２１】

図２１

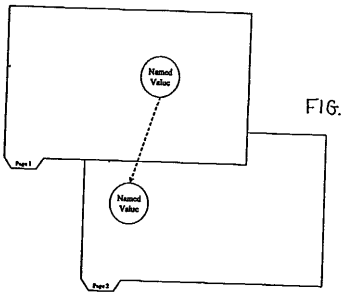


FIG. 21

【図２２】

図２２

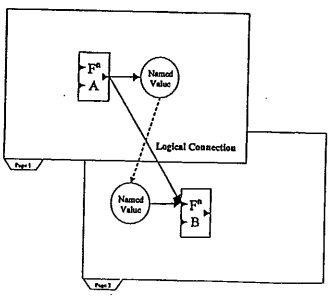


FIG. 22

【図２３】

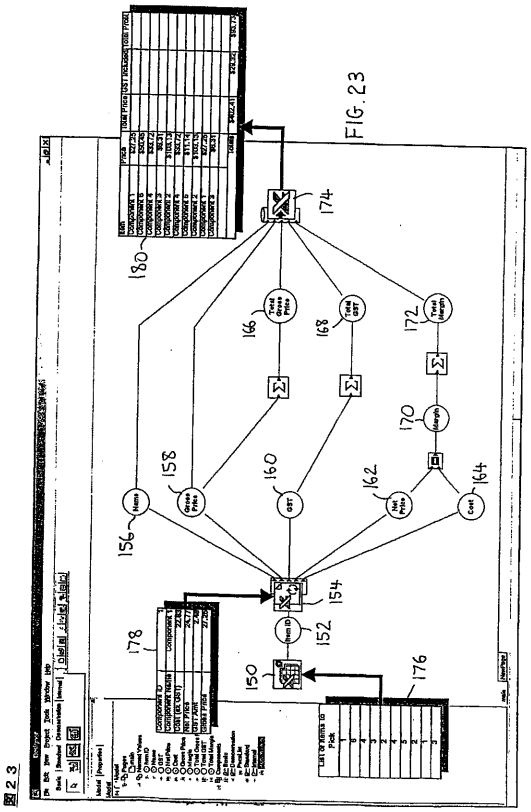


FIG. 23

【図２４】

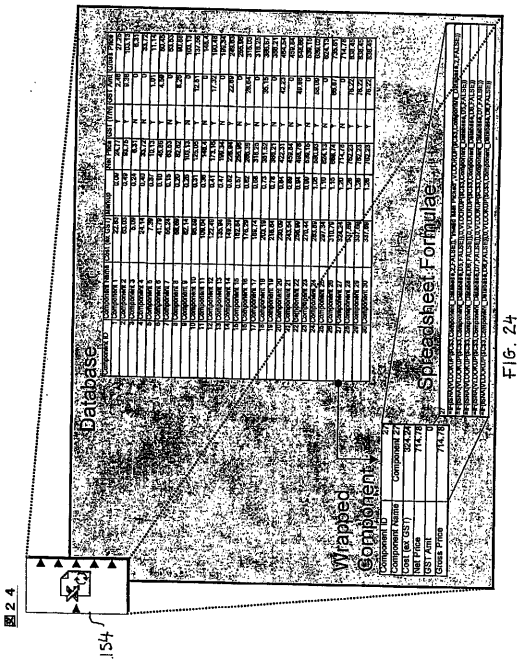


FIG. 24

【図２５】

図２５

APPENDIX 1

```
<?xml version = "1.0"?>

<Xemplex>
   <Package Name = "Internal" Description = "">
      <Type Name = "Item List" Description = "Z:\Demo\Price List NQ.xls" Type =
"InputData">
         <Application Name = "Excel97" Workbook = "Z:\Demo\Price List NQ.xls"
Worksheet = "Items to Pick" Start = "2"/>
         <Icon>

<![CDATA[NDI0REU2MDQwMDAwMDAwMDAzNjAwMDAwMDI4MDAwMDAwMTQwMDAwMDAxNDAwMDAwMDA

0ODQ4NDg0ODQ4NDg0ODQ4NDg0ODQ4NDg0ODQ4NDg0ODQ4NDg0ODQ4NDg0ODQ4NDg0]]>
         </Icon>
         <Outputs>
            <Output Name = "ItemID" Type = "Number" DataType = "Number" Line =
"A"/>
         </Outputs>
      </Type>
      <Type Name = "Sum" Description = "Z:\Demo\Price List NQ.xls" Type =
"OutputData">
         <Application Name = "Excel97" Workbook = "Z:\Demo\Price List NQ.xls"
Worksheet = "Items to Pick" Start = "2"/>
         <Icon>

<![CDATA[NDI0REU2MDQwMDAwMDAwMDAzNjAwMDAwMDI4MDAwMDAwMTQwMDAwMDAxNDAwMDAwMDA

0ODQ4NDg0ODQ4NDg0ODQ4NDg0ODQ4NDg0ODQ4NDg0ODQ4NDg0ODQ4NDg0ODQ4NDg0]]>
         </Icon>
         <Inputs>
            <Input Name = "Total" Type = "Number" Line = "F"/>
         </Inputs>
      </Type>
   </Package>
   <ComponentPackages>
      <Package Filename = "Z:\Delphi\ComponentManager\Basic.xmp"/>
      <Package Filename = "Z:\Delphi\ComponentManager\Standard.xmp"/>
   </ComponentPackages>
   <Model>
      <Page Name = "NewPage">
         <Component Name = "Add0" ComponentPackage = "Standard" ComponentType =
"Add" Left = "161" Top = "83">
            <Connector Type = "Input" Name = "Connector #0" UID = "12" Edge =
"Left"/>
            <Connector Type = "Input" Name = "Connector #1" UID = "13" Edge =
"Left"/>
            <Connector Type = "Output" Name = "Connector #2" UID = "14" Edge =
"Right"/>
         </Component>
         <Component Name = "Memory3" ComponentPackage = "Basic" ComponentType =
"Memory" Left = "266" Top = "85">
            <Connector Type = "Input" Name = "MI" UID = "15" Edge = "Left"/>
            <Connector Type = "Output" Name = "MO" UID = "16" Edge = "Right"
Initial = "0"/>
         </Component>
         <Component Name = "Block4" ComponentPackage = "Basic" ComponentType =
"Block" Left = "485" Top = "84">
```

【図２６】

図２６

```
            <Connector Type = "Input" Name = "BI" UID = "17" Edge = "Left"/>
            <Connector Type = "Output" Name = "BO" UID = "18" Edge = "Right"/>
            <Connector Type = "Gate" Name = "BG" UID = "19" Edge = "Bottom"/>
         </Component>
         <Component Name = "Item List5" ComponentPackage = "Internal"
ComponentType = "Item List" Left = "27" Top = "86">
            <Connector Type = "Output" Name = "ItemID" UID = "20" Edge =
"Right">
               <Attributes Line = "A"/>
            </Connector>
         </Component>
         <Component Name = "Sum5" ComponentPackage = "Internal" ComponentType =
"Sum" Left = "716" Top = "77">
            <Connector Type = "Input" Name = "Total" UID = "21" Edge = "Left">
               <Attributes Line = "F"/>
            </Connector>
         </Component>
         <NamedConnection Name = "Running total" UID = "22" Left = "400" Top =
"101"/>
         <NamedConnection Name = "Sum" UID = "23" Left = "620" Top = "102"/>
         <VisibleConnection EndpointA = "15" EndpointB = "14"/>
         <VisibleConnection EndpointA = "16" EndpointB = "22"/>
         <VisibleConnection EndpointA = "12" EndpointB = "22">
            <Vertex Top = "92" Left = "160"/>
            <Vertex Top = "92" Left = "115"/>
            <Vertex Top = "33" Left = "115"/>
            <Vertex Top = "33" Left = "400"/>
         </VisibleConnection>
         <VisibleConnection EndpointA = "17" EndpointB = "22"/>
         <VisibleConnection EndpointA = "18" EndpointB = "23"/>
         <VisibleConnection EndpointA = "13" EndpointB = "20"/>
         <VisibleConnection EndpointA = "21" EndpointB = "23"/>
      </Page>
   </Model>
</Xemplex>
```

【図２７】

図２７

APPENDIX 2

```
<?xml version = "1.0"?>

<Xemplex>
   <Executable>
      <Component Name = "Add0" Type = "Generic" Application = "Xemplex"
Operation = "Add">
         <Input Name = "Connector #0" Type = "Input"/>
         <Input Name = "Connector #1" Type = "Input"/>
         <Output Name = "Connector #2" Type = "Output"/>
      </Component>
      <Component Name = "Memory3" Type = "Generic" Application = "Xemplex"
Operation = "Memory">
         <Input Name = "MI" Type = "Input"/>
         <Output Name = "MO" Type = "Output" Initial = "0"/>
      </Component>
      <Component Name = "Block4" Type = "Generic" Application = "Xemplex"
Operation = "Block">
         <Input Name = "BI" Type = "Input"/>
         <Output Name = "BO" Type = "Output"/>
         <Gate Name = "BG" Type = "Gate"/>
      </Component>
      <Component Name = "Item List5" Type = "InputData" Application = "Excel97"
Workbook = "Z:\Demo\Price List NQ.xls" Worksheet = "Items to Pick" Start = "2">
         <Output Name = "ItemID" Type = "Number" Line = "A"/>
      </Component>
      <Component Name = "Sum5" Type = "OutputData" Application = "Excel97"
Workbook = "Z:\Demo\Price List NQ.xls" Worksheet = "Items to Pick" Start = "2">
         <Input Name = "Total" Type = "Number" Line = "F"/>
      </Component>
      <Connection From = "Add0" Output = "Connector #2" To = "Memory3" Input =
"MI"/>
      <Connection From = "Memory3" Output = "MO" To = "Add0" Input = "Connector
#0"/>
      <Connection From = "Memory3" Output = "MO" To = "Block4" Input = "BI"/>
      <Connection From = "Item List5" Output = "ItemID" To = "Add0" Input =
"Connector #1"/>
      <Connection From = "Block4" Output = "BO" To = "Sum5" Input = "Total"/>
   </Executable>
</Xemplex>
```

【国際公開パンフレット】

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
28 February 2002 (28.02.2002)

PCT

(10) International Publication Number
WO 02/17074 A1

(51) International Patent Classification[7]: G06F 9/455, 17/00, G05B 17/00

(21) International Application Number: PCT/AU01/01053

(22) International Filing Date: 24 August 2001 (24.08.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
PQ 9664 24 August 2000 (24.08.2000) AU

(71) Applicant *(for all designated States except US)*: **XEMPLEX PTY LTD** [AU/AU]; 57B Fraser Street, EAST FREMANTLE, Western Australia 6158 (AU).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **BARGH, Christopher, Ian** [AU/AU]; 35 Ventnor Street, SCARBOROUGH, Western Australia 6109 (AU). **JOHNSTON, Gregory, Owen** [AU/AU]; 57B Fraser Street, EAST FREMANTLE, Western Australia 6158 (AU). **JONES, Russell, Benedict** [AU/AU]; 78 Refum Street, SUBIACO, Western Australia 6008 (AU).

(74) Agent: **GRIFFITH HACK**; Level 6, 256 Adelaide Terrace, PERTH, Western Australia 6000 (AU).

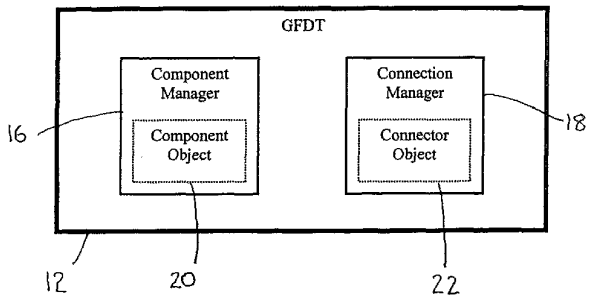
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) **Title:** METHOD OF GRAPHICALLY DEFINING A FORMULA



(57) **Abstract:** A computer-implemented method of graphically defining a formula, includes providing a first operator object for defining a method of manipulating at least one input to produce at least one result. A graphical representation of the first operator object is displayed. A variable object for containing data is provided. An input from a user to relate the variable object to one of inputs or one of the results of the first operator object is received. A graphical representation of the first variable object and its relation to the operator object is displayed. A logical description of the relationship between objects is recorded thereby defining the formula.

WO 02/17074 A1

METHOD OF GRAPHICALLY DEFINING A FORMULA

Field of the Invention

The present invention relates to a method of graphically defining a formula for manipulating input data to produce a result. Preferably the method is embodied in the form of a computer program.

Background of the Invention

It is common for complex manipulation of data to involve many complex formulae. Often a model that provides a multilevel approach to representing the manipulation of the data is useful to define the formulae. It can often be difficult working out the formulae needed for each level of the model. It can also be helpful to this process if the formulae can be represented and defined graphically.

US Patent No. 4,901,221 to *Kodosky et al* discloses a graphical system and method for modelling a process. The method disclosed allows a user to construct a diagram using a block diagram editor such that the diagram created graphically displays a procedural method for accomplishing a certain result. As the user constructs the data flow diagram, machine language instructions are automatically constructed with which characterise an execution procedure which corresponds to the displayed procedure. A user can create a text based computer program solely by using a graphically based programming environment. A limitation of this method is that it relies upon iteration control for producing each output that is the result of a function of data applied to an input variable at any given time. It also relies on assembling on a screen a data flow diagram including an iteration icon that references an iteration control means for controlling multiple iterations of data flow.

It is desirable when designing the model not to be concerned with iterations of data particularly when designing an object orientated model.

Summary of the Invention

An object of the present invention is to provide a method of graphically defining a formula.

According to a first aspect of the present invention there is provided a computer-implemented

method of graphically defining a formula, said method including:

providing a first operator object for defining a method of manipulating at least one input to produce at least one result;

displaying a graphical representation of the first operator object;

providing a first variable object for containing data;

receiving an input from a user to relate the variable object to one of the inputs or one of the results of the first operator object;

displaying a graphical representation of the first variable object and its relation to the first operator object; and

recording a logical description of the relationship between the objects;

whereby the formula is defined by the logical description.

According to a second aspect of the present invention there is provided a computer-implemented method of graphically defining a formula, said method including:

providing a first variable object for containing data;

displaying a graphical representation of the variable object;

providing a first operator object for defining a method of manipulating at least one input to produce at least one result;

receiving an input from a user to relate one of the inputs or one of the results of the first operator object to the variable object;

displaying a graphical representation of the first operator object and its relation to the first variable object; and

recording a logical description of the relationship between the objects;

whereby the formula is defined by the logical description.

Preferably the method further includes the steps of:

providing one or more further variable objects;

receiving further inputs from the user to relate each further variable object to one of the inputs or one of the results of the first operator object;

displaying a graphical representation of the further variable objects and their relation to the operator object.

Preferably the method further includes the steps of:

providing one or more further operator objects;

receiving further inputs from the user to relate each variable objects to one of inputs or one of the results of the further operator objects;

displaying a graphical representation of the further operator objects and their relation to the variable objects.

Preferably each variable object is selected from: an input object for providing data from a data source; an output object to provide data to a data destination; or a connection object for passing data from one operator object or another. Preferably a connection object represented as a link between the operator objects. Preferably each variable object may be provided with a variable label. Preferably each operator object may be provided with a operator label.

Preferably, the logical description of the formula is defined by the logical relationship between the objects. Preferably a graphical definition of the formula is recorded that defines the graphical display of the relationship between objects.

Preferably the method includes the step of storing information describing the logical definition. Preferably the method includes the step of storing information describing the graphical definition.

Preferably, two or more related operator objects may be grouped such that the grouping defines a grouping operator object, wherein variable objects crossing the border of the grouping and connecting to inputs of operator objects in the group become the inputs of the grouping object component and variable objects crossing the border of the grouping and connecting to results of the operator objects in the group become results of the grouping operator object. Preferably inputs and results of operator objects in the group not linked to another object become inputs and results, respectively, of the grouping operator object. Preferably the graphical representation of the grouped objects is replaced by a graphical representation of the grouping operator object and the graphical representation of links to the contents of group are replaced with graphical representations of links to the representation of the grouping object.

Preferably the logical definition of the formula defined includes the contents of the grouping

operator object. Preferably the graphical definition of the overall formula displayed excludes the contents of the grouping operator object. Preferably the contents of the grouping operator object may be graphically represented separately from the overall graphical representation of the formula.

Preferably, variable objects may be attributed with properties that define the type of data they can hold. Preferably each input and result of an operator object may be attributed with properties that define the type of data that the operator object expects to receive and be able to produce, respectively.

Preferably, a variable object may inherit the properties from the properties of another variable object that has already been defined and is related by an intervening operator object. Preferably, a variable object may inherit the properties from the properties of an operator object input or result that has already been defined and to which it is related. Preferably, an input or result of an operation object may inherit the properties from the properties of a variable object that has already been defined and to which it is related.

Preferably, the method includes a step of checking that the properties of objects with already attributed which are being related match.

Preferably a library of labelled variable objects is predefined. Preferably a library of labelled operator objects is predefined, each labelled operator object's method of manipulating its input/s to produce its result/s also being predefined.

Preferably the variable label of a variable object may be selected from a list of predefined variable labels. Preferably each variable label may be attributed with properties that define the type of data a variable object labelled with the label can contain. Preferably the selection of a variable label attributes the properties associated with the label to the variable object. Preferably the properties attributed to a variable object limit the selection of labels available to be selected.

Preferably, the first operator object is at least one of addition, subtraction, multiplication, division, a look-up table and conditional operation. Alternatively, the first operator object

may be a multiple stage operation containing a plurality of simple operators linked to perform a more complex operator. In one form, the first operator object is a query of the database. In another form, the first operator object performs a write to a database.

Preferably the operator label of an operator object may be selected from a list of predefined operator labels. Preferably each operator label may be attributed with properties that define the type of data that inputs and results of a labelled operator object can receive or provide, respectively. Preferably the selection of an operator label attributes the properties associated with the label to the operator object. Preferably the properties attributed to an operator object limit the selection of labels available to be selected.

Preferably the logical definition may be used by a runtime engine to put into operation the defined formula, wherein data provided to each variable objects linked to an input of an operator object, whereby the data becomes operands of the formula, each operator represented by the operator object becomes the operator of the formula and each result of the operator object becomes the next operand of the next operator or the final result/s of the formula, whereby computation of the formula can be conducted to produce a formula result.

Preferably a namespace may be defined for each variable, whereby the data in a logical variable represented by the variable object is the same for each occurrence of the variable object within the namespace. Preferably the name space is by default global to the formula being modelled. Preferably a logical connection is created between each occurrence of a labelled variable object within a namespace. In one embodiment a graphical link may be displayed showing the logical connection between occurrences of labelled variable objects.

Preferably a namespace may be defined for each operator object, whereby the operation of a logical operator represented by the operator object is the same for each occurrence of the operator object within the namespace.

Preferably a grouped operator object may be used more than once with the definition of the grouped operator object being applied to the logical definition of the formula.

Preferably the properties of a label include type, units and dimension.

Preferably the graphical definition is described in XML. Preferably the logical definition is described in XML.

Preferably each operator object includes a plurality of definitions of the operation performed by the operator represented by the operator object, each definition being for a separate type of data able to be manipulated by the operator.

Preferably the operator object is graphically represented as a component having one or more inputs and one or more outputs, the component having an indicator representative of the operator represented. Preferably the operator object may be an empty component that is representative of a operator with methodology of manipulation inputs to produce results yet to be defined. Preferably the empty component is used to form criteria for searching for a suitable operator object that has a suitable defined methodology.

Preferably a library of objects is provided. Preferably objects may be externally sourced.

According to a third aspect of the present invention there is provided a system for graphically defining a formula, comprising:

a computer including a display screen and a user input means;

means for providing a first operator object for defining a method of manipulating at least one input to produce at least one result;

means for displaying a graphical representation of the first operator object on the screen;

means for providing a variable object for containing data;

means for receiving an input from the user input means to relate the variable object to one of inputs or one of the results of the first operator object;

means for displaying a graphical representation of the first variable object and its relation to the operator object on the screen;

whereby the formula is defined by the relationship between the objects.

According to a fourth aspect of the present invention there is provided a computer program for controlling a computer for graphically defining a formula, said computer program causing

the computer to undertake steps including:

providing a first operator object for defining a method of manipulating at least one input to produce at least one result;

displaying a graphical representation of the first operator object on a computer screen;

providing a variable object for containing data;

receiving an input from a user input means to relate the variable object to one of inputs or one of the results of the first operator object;

displaying a graphical representation of the first variable object and its relation to the operator object on the screen;

whereby the formula is defined by the relationship between the objects.

According to a fifth aspect of the present invention there is provided a computer readable medium for storing a computer program as defined above.

According to a sixth aspect of the present invention there is provided a method of graphically defining a formula for manipulating input data to produce a result, said method including:

providing at least one variable for containing data;

providing at least one operator defining the method of manipulating the input data to produce the result;

displaying a list of the variables for a user to select a result variable therefrom;

receiving a selection of the result variable from the user for containing the result of the manipulation of the input data;

displaying a graphical representation of the selected result variable;

displaying a list of the operators for a user to select an operator therefrom;

receiving a selection of an operation from the user;

displaying a graphical representation of the selected operation;

displaying a list of inputs for containing the input data for a user to select at least one input therefrom, the inputs being either said variables or one or more constants;

receiving a selection of at least one input from the user;

displaying a graphical representation of the selected input,

whereby the formula is defined by the selected result variable being equal to the manipulation of selected input(s) by the selected operation.

According to a seventh aspect of the present invention there is provided a method of graphically defining a formula for manipulating input data to produce a result, said method including:

providing at least one variable type, said variable type having pre-determined properties;

providing at least one operation defining the method of manipulating the input data to produce the result;

displaying the variable types for a user to select a variable type therefrom;

receiving a selection of the variable type from the user;

receiving a name for the selected variable type;

displaying a representation of the named variable;

displaying a list of operations for a user to select an operation therefrom;

displaying a graphical representation of the selected operation;

receiving a selection of an operation from the user;

receiving input from the user so as to associate the selected variable with the selected operation so that the selected variable is either an input variable or a result variable;

where the selected variable is associated to be a result variable, receiving from the user a selection of at least one of either an input variable or an input constant and a name for the input variable or the input constant, displaying a graphical representation of the input variable(s) and/or input constant(s);

where the selected variable is an input variable, receiving a name for an output variable, displaying a graphical representation of the output variable;

whereby the formula is defined by the result of the manipulation by the selected operation of the input data in the input data variable or input constant provided to the result variable.

Detailed Description of Preferred Embodiments

In order to provide a better understanding, a preferred embodiment of the present invention will be described in more detail, by way of example only, with reference to the accompanying drawings, in which:

Figure 1 is a schematic representation of a system for performing the method of the present invention, including a graphical formula definition tool;

Figure 2 is a schematic representation of the graphical formula definition tool of Figure 1;

Figure 3 is a screen shot of a window produced by the graphical formula definition tool in Figures 1 and 2;

Figure 4A is a schematic representation of a first step in a preferred embodiment of graphically representing a formula;

Figure 4B is a schematic representation of a second step of graphically representing a formula;

Figure 4C is a schematic representation of a third step of graphically representing a formula;

Figure 4D is a schematic representation of a fourth step of graphically representing a formula;

Figure 5 is a schematic representation of a graphically represented formula;

Figure 6 is a flow chart showing steps in the creation of a component (operator) object;

Figure 7 is a flow chart showing steps in the creation of a named connection (variable) object;

Figure 8 is a flow chart showing steps in the creation of a connection object;

Figure 9 is a schematic representation of a graphically represented formula;

Figure 10 is a graphical representation of the formulas defined in Figure 4D, Figure 5 and Figure 9 linked together to form a high level formula;

Figure 11 is a graphical representation of the high level formula of Figure 10 contracted so as to not show the individual stages in the high level formula;

Figure 12 shows a further example of restriction of variable types due to inheritance of variable properties;

Figure 13A shows a schematic representation of a formula with a pointer showing properties of an input to a lookup table component;

Figure 13B shows a schematic representation of a formula with a pointer showing properties of a named variable object;

Figure 14 shows a schematic representation of a graphically defined formula with an input variable to the formula being a database;

Figure 15 is a schematic representation of another graphically defined formula including an operator;

Figure 16 shows how the function C of Figure 15 may be changed;

Figure 17 shows how the function C of Figure 12 may have additional detail added;

Figure 18 shows schematic representation of a component being passed through a computer from a first computer through a computer network to a second computer;

Figure 19 shows another screen shot provided by the graphical formula definition tool of Figure 2;

Figure 20 provides a schematic representation of an overflow methodology performed by an operator;

Figure 21 is a schematic representation of logical connection between named variable objects on separate pages; and

Figure 22 is a schematic representation of a logical connection between named variable objects on separate pages and the output of function A to the input of a function B on separate pages.

A formula is a description of a methodology for calculation of a result by applying an operator to one or more operands. Typically the operand is a variable, in the algebraic sense. The present invention defines a formula as a description in terms of operands (variables) and operators. Specifically the variables and operators have a relationship. The simple formula X = A + B, is a description of the relationship between the operands A and B and an addition operator (+). The present invention enables this to be represented graphically, which is desirable when representing complex models and functions. The present invention also produces a description of the graphical relationship defined. In other words, the relationship between the variables and operators that defines the formula.

The method of the present invention is performed by a graphical formula definition tool (GFDT) 12. Referring to Figure 1, the GFDT 12 interacts with a graphical user interface (GUI) 14. The GUI 14 forms part of a computer's operating system, examples of which are Microsoft Windows, in its various editions, Mac OS for the Macintosh brand of computers, or X-Windows which runs under the UNIX operating system. The GFDT 12 communicates with the GUI 14 to provide instructions for providing graphical display. The actual handling of the graphical display is conducted by the GUI 14.

Referring to Figure 2, the GFDT 12 comprises two main parts, a component manager 16 and a

connection manager 18. The component manager 16 handles operator objects 20 that are provided for defining a method of manipulating at least one input to produce a result. An operator object is representative of an operator in the formula. In the present example, operators are referred to as components, thus the component manager manages component objects for purposes of the preferred embodiment.

The connection manager 18 handles variable objects. Variable objects are representative of operands in the formula for each instance of calculation of the result of the formula. Variable objects come in a number of types the main one of which is a connection object 22 that passes data into or out of a component, as will be described in more details below. Other types of variable objects are input and output objects which are generally named with a label. These are referred to in the preferred embodiment as named connections, as they are given a label that the user can refer to, to know information about the data being passed to or from a component. This will be described in more detail below.

The GFDT 12 provides to the user, via the GUI 14, with an interface that graphically represents the formula as being defined. The GFDT 12 also records a logical description of the formula as it is defined. Recording of the graphical description is separate from the recording of the logical description. The graphical description describes what is displayed on the interface. The logical description describes in logical terms the relationship between objects displayed in the interface. A screen shot of an interface 24 is shown in Figure 3. The interface 24 is a standard Windows window, it has a tool bar portion 26, a primary window 28, and a secondary window 30. The primary window 28 displays a current page showing a portion of the overall formula. The window 26 includes a page selector 32. Other pages may be provided that show other parts of the formula. The main window 28 is shown displaying simple model for an accumulator.

The method of constructing a formula or model will be described with reference to Figures 4A to 4D. This example relates to a mine site. Planning, budgeting or monitoring of the mine site is conducted by using the results of calculations in accordance with predefined formulae. The formulae determine the results produced when information contained in input variables is manipulated according to a specific operator to produce the result. From the results planning, budgeting or monitoring of the operations of the mine site can be conducted. Specifically, the

result of Load and Haul IPD is used as an example. Load and Haul is the cost of loading and hauling mineral bearing material out of a mining pit, in this case, In a Pit named Defiance (IPD).

Referring to Figure 4A, the Load and Haul IPD is desired to be calculated. A variable object 34 is selected from the tool bar 26 and placed in the main window by the well known process of positioning the pointer by moving the mouse, clicking, dragging and dropping the variable object in the desired position. The variable is given the name "Load and Haul IPD". A convention for symbols used is that named variables (labelled variable objects) are in circles, components (operator objects) are rectangular, with inwardly pointing arrows representing inputs and outwardly pointing arrows representing outputs, and data flow connections (variable objects) are represented as lines.

The Load and Haul IPD variable 34 may be given certain properties. This is conducted by selecting the "Properties" tab in the secondary window 30 and entering the desired properties, such as it must be a numeric value, and more specifically it must be a currency numeric value and that the units of currency are AU$. The variable, once created, is graphically represented on a video display unit as a circle 34 labelled "Load and Haul IPD". The icon of the variable will represent that it may receive data from any suitable source, such as manual entry, it may be received from another program, or it may be received from a database.

The Load and Haul IPD is calculated from "Load and Haul IPD (Bulk)" data added to the "Load and Haul (Selective)" data. That is, the Load and Haul IPD represents the cost of loading and hauling bulk material and selective material. Bulk material is from a mining method that produces mineral bearing material and non-mineral bearing material. Selective mineral is from a mining method that produces high-grade mineral bearing material.

Referring to Figure 4B, the next step is to select a component 36 from the tool bar 26 and placing in the main window 28. Normally, a list or a number of buttons for each component are provided to the user to select from, such as addition, subtraction, multiplication, division, as shown in Figure 3. More components can also be provided, such as look-up tables, conditional operations and other more complicated operations such as calculus, trigonometric and other operations. Additionally, data or text manipulation operations can be included. The

type of operations able to be conducted should not be limited. Indeed, a vast library of operations could be made available including multi-levelled operations, which are discussed in more detail below. In this case, the "addition" operation is selected. The addition operation is graphically represented as a box 36. The addition operation has at least two (and in this case actually two) inputs and one output. A relationship between the objects is then created. In this case the "Load and Haul IPD" variable is the result of the addition, so it can be connected by selecting a connection object 38 from the tool bar 26 to draw a line 38 connecting the output arrow of the addition component 36 to the Load and Haul IPD variable 34. Thus the Load and Haul IPD becomes a result variable, by virtue of the nature of the association. The inputs and outputs are represented as arrows and the association is represented as the connection 38 between the result arrow and the variable.

Referring to Figure 4C, another variable object 40 is selected from the tool bar 26, placed in the main window 28 and is given the name Load and Haul IPD (Bulk). Variables names may be initially entered to produce a list that the user selects the desired variable from. Otherwise, the name of the variable may be entered as required. The Load and Haul IPD (Bulk) variable 40 is associated with an input of the addition operator 36 by selecting another connection object 42 from the tool bar 26 and making a connection between the two objects. Load and Haul IPD (Bulk) thus becomes an input variable. The Load and Haul IPD (Bulk) input variable is represented as a circle 40 and the input relationship is represented as the connection 42 between the variable and the component.

Referring to Figure 4D, another variable 44 is named Load and Haul IPD (Selective) is selected, placed and associated with the other input of the addition operator 36. The definition of the formula is now complete. The selection of the result variable, the component (operator) and the input variables and the relationship therebetween has resulted in a formula being defined as follows: input variable "Load and Haul IPD (Bulk)" and input variable "Load and Haul IPD (Selective)" are summed together by the addition operator to produce the result variable "Load and Haul IPD". When the formula is put into operation, input data entered into the input data variables will produce a result in the result variable according to the defined formula.

In order to save the defined formula for later retrieval the type, name and properties of each

object are all recorded along with the position of each object within the window 28. With this information the formula can be stored and retrieved for later display.

Simultaneously with the drawing of the block diagram formula, a logical definition of the formula is recorded. Referring back to Figure 4A the creation of the variable object Load Haul IPD 34 is registered. In Figure 4B the creation of the addition component 36 is also registered as it is placed. The placement of the connector 38 between the output of the addition component and the variable Load and Haul IPD is also registered. A logical connection between the result output of the component and the result variable is then recorded.

Referring to Figure 4C likewise the creation of the variable object "Load and Haul IPD (Bulk)" 40 and its connection to a first input of the addition component is registered. Again in Figure 4D the creation of the "Load and Haul (Selective)" selective variable and its connection to the second addition component input is also registered. The registration of the components and their connections therebetween (and thus their relationship) is therefore registered and thus a logical definition of the formula is created in the form of a description of the objects and their relationships. The positioning on the screen and other graphical information is not important to the logical description and is only recorded in the graphical description.

The graphical description is used for display of the formula to the user in a manner that the user can relate to and the recording of a logical description of the formula is used by a formula processing engine to put the defined formula to use. The user need not be concerned with the logical definition and the processing engine need not be concerned with the graphical definition. At this stage for such a simple formula the graphical definition and the logical definition are not substantially different at a conceptual level. However with more complex formula being modelled, as will be described below, these definitions will diverge. Yet the user will still be able to relate to the formula being defined at an intellectual level by the graphical representation of the formula and the formula processing engine will be able to use the logical definition without having to exclude information only relevant to the graphical representation.

An association between a variable and an operator enables the properties of other inputs of the operator to be determined, at least to some extent. For example, the association of the result variable with the operator enables the properties of the inputs of the operator to be known, in this case currency numeric values. When the Load and Haul IPD (Bulk) input variable is associated with one of the inputs the properties associated with this input variable can be checked against that required by the operator. Alternatively, if the Load and Haul IPD (Bulk) input variable does not already have properties associated with it, it can inherit these properties. So, because Load and Haul IPD has the property of being a currency numeric value, both Load and Haul IPD (Bulk) and Load and Haul IPD (Selective) variables must also be currency numeric values. A check can be performed and if either input does not match the required properties a warning can issue or the association will not be allowed. Otherwise if an input does not have any properties associated, then it will inherit the currency numeric value property.

Checking and inheritance can work both ways. That is, if an input has a currency numeric value property, the result is also checked to see whether it has consistent properties. Normally, the more recently created input/result variable is the one checked.

Inputs and outputs of components hold property information relating to the component. A component is defined by the properties of its inputs and outputs along with the functionality that produces the output from the inputs.

In Figure 5, a formula for calculating the value of "Load and Haul IPD (Bulk)" 40 is defined by multiplying the input variable "Bulk Rate" 48 by input variable "Bulk BCMs" 50. Bulk rate represents the cost for mining each in-situ cubic meter of "Bulk" mineral bearing material. Bulk BCM's represents the Bank (in-situ) Cubic Meters (BCM) of Bulk material. This formula can again be built using similar steps to define the previous formula. The "Load and Haul IPD (Bulk)" input variable 40 is able to be used as a result variable. It may therefore be selected and represented on the display. If "Bulk Rate" and "Bulk BCMs" input variables are not already available, they may be entered. The multiply operator object 48 is then selected and associated with the input variables and output variables. This may be done by dragging the representation of the multiply operator and placing it. The input and result variables are related to the operator so the input variables ("Bulk Rate" and "Bulk BCMs")

are connected to the operator inputs and the output arrow is connected to the result variable ("Load and Haul IPD Bulk"). Here the order of placement of the variables and operation is different to the previously defined formula. The order only makes a difference to order of checking properties and inheritance. If the properties of the input variables when multiplied together are not consistent with the properties of the result variable, a message may be provided to the user that there is a problem with property inheritance. That is, if the Bulk rate input variable does not have the property of currency per volume numeric value and/or Bulk BCM's input variable does not have the property of volume numeric value a warning message will be given or if one of the two have the correct property, the other will inherit the correct property. Property analysis and inheritance need not be limited to the dimensions of the variable/constant. The units of the dimension can be checked, for example, if one unit is AU$ and the other is US$, this will result in a warning. Alternatively, a conversion may be conducted as described further below.

This formula may be created on the same page as the formula of Figure 4D or it may be created on a new page. The page selector 32 may be used to select the appropriate page if they are on separate pages. This is also where the graphical definition may begin to diverge from the logical definition, particularly if they are on separate pages. While each page will contain a separate graphical definition for each formula, the logical definition will form a connection between the Local and Haul IPD (Bulk) result variable of the formula of Figure 5 and the Load and Haul IPD (Bulk) input variable of the formula of Figure 4D. Thus, to the user, one formula is represented as two smaller formulae. This will aid in intellectual understanding of the formula, but logically there is no separation. This is not to say a separation cannot occur where the same named variable has a particular space within which to operate. A namespace for a variable (and a component) can be defined limiting their application. This will be described in more detail below.

The processes conducted by the component manager 16 and the connection manager 18 are now described in more detail with reference to Figures 6, 7 and 8. Referring to Figure 6, a component manager 16 first allows the user to select a component type from a palate displayed in the tool bar 26 at 52. In the example shown in Figure 3, an addition, subtraction, multiplication and division buttons are provided for the selection as components. When one of these buttons is depressed the operating system informs the component manager 16 that

that particular button has been selected. The type of operator component is known. The user then clicks in the drawing window 28 at 54 which results in the placement of a component in the location clicked. This entails the component being drawn in the drawing window 28 at 56 and the component being registered in the formula definition at 58. Details specific to the graphical representation of the component are stored such as the component name, its type and the position on the page and details of the icon representing the component. In the logical definition the details such as the name and type of the component are registered.

Referring to Figure 7 when a named connection is to be included in the formula, the user selects a named connection component type from the palate at 50. The user clicks in the drawing window 28 at 62 and an empty named connection variable is drawn at 64. The user can then name the variable or wait and name it later. If it is named, a facility is provided at 66 for the user to enter the name which is then displayed in the new named connection at 68. The named connection is then registered at 70. Details related to the graphical display of the named connection are recorded such as the name, the type, the position on the page, the named connection. Logical description details of the named connection are registered such as the name, type of named connection.

Referring to Figure 8, a connection between operator objects such as components or components and named connections is described in relation to Figure 8. The user selects a connection option in the tool bar at 72. The user clicks on a component and drags a connector link to a named connection or another component at 74. The connection manager then determines whether the connection is allowed based on the properties of the objects being connected at 76. If the connection is not allowed, as indicated by "no" on 78, a warning is provided by changing the colour of the line being drawn to red, alternatively the colour of the named connection may be changed to red at 88. When the user releases the mouse at 90 the drawing is not completed at 92 and thus the connection is not registered. If the connection is allowed, as indicated by "yes" on 78, the colour of the named connection is turned to green at 80. As the user releases the mouse at 82 the line representing the connection is drawn at 84 and the connection registered at 86. Details relating to the graphical representation of the connection (line) such as the end points of the line and any vertexes (bends) on the line. Details relating to the logical description are registered such as the details of the components being connected.

The processes described in relation to Figures 6, 7 and 8 occur for each page of the drawing. In addition, in the graphical description each drawing has a name recorded (the page name) and for each page of the drawing each registered component named connection and connection details are recorded. In addition, namespace details are recorded as will be described in more detail below.

Referring to Figure 9, another example of formula definition is shown. In this instance the "Bulk Rate" 48 is defined by a look-up table 94 from a number of variables and constants 96 to 104. Pit 96 is a variable that represents the name of the mining Pit. Schedule 98 is a variable that represents a mining rate that depends on the amount of mineral mined. RL 100 is a variable that represents the relative level of depth into the Pit that the mineral is taken from. Material type 102 is a variable that represents the type of material being mined, for example, it may be fresh or sediment material. Bulk "B" 104 is a text constant. The look up table 94 is an operation that looks up a value based on the values of the five inputs. The resulting value is then provided to the result variable 48. The figure shows the connections between the inputs and the outputs. The building of the relationship by the selection of the inputs, results and operators and the placement of the representations of these on the screen is recorded. The logical description of these objects and relationships therebetween defines this formula.

Referring to Figure 10, the example described thus far has been using a top down methodology to define a model as a number of simple formulae. These formulae can be collated or the model drawn as one complex formula as shown. It can be seen that the input variable PIT 96, input variable SCHEDULE 98, input variable RL 100, input variable MATERIAL TYPE 102, input constant Bulk "B" 104, input variable Bulk BCM's 50 and input variable Load and Haul IPD (Selective) 44 are all used to calculate the final output result variable Load and Haul IPD 34. The outlined label area 106 shows that the existing multistep formula (modules) can be chained together to produce a more complicated higher level formula. The steps inside the dashed box 106 can be grouped to form a high order component. The inputs on the high order component are shown as an "X" and the output being shown as a small circle.

This high level formula could also be defined without the need for the variables "Bulk Rate" and "Load and Haul IPD (Bulk)". Instead the output of operators 94 can feed directly in the input of operator 46, and the output of operator 46 can feed directly into the input of operator 36. In other words the operators can be directly chained together. However, it may be more suitable to design this formula as shown in Figure 10 if the variables "Bulk Rate" and "Load and Haul IPD (Bulk)" are used elsewhere.

The componetising of a chain of operators is conducted by drawing a box 106 around the components to be componetised and selection of a componentise function. The objects within the box 106 are then deleted from the current page and shifted to a new page. The internal workings of the new component can be viewed on the new page. The graphical description of the components are copied to the new page. In place of the deleted objects is a new component 108 as shown in Figure 11. A graphical description of the component is included in the current page description with each input into the deleted components forming an input into the new component 108. A connection from the named connections 96 to 104, 50 and 44 is created to the corresponding input of the new component 108. The connection from the output of the new component is connected to the named component 34. The description on the current page is updated to reflect the new component and the connections thereto. A logical connection is created between each of the inputs of the new component and each of the inputs of the grouped component on the other page. Likewise a logical connection is created between the output of the component and the output of the component on the other page. Thus the logical description of the model in effect remains unchanged whereas the graphical description of the model is different.

In Figure 11, the chained operators inside 106 of Figure 10 have been grouped together to provide a higher level operator 108. This operator 108 requires the five inputs to produce the Load and Haul IPD result variable. The new component 108 can now be reused without the need to redefine the individual lower level formula that make up the high level operator 108. A facility may be provided to show the workings of a component. This may be for example, by "double clicking" on the high level formula to open it up to display the chain inside by "turning" to the page in which the inner workings of the component are shown. This process is called "drilling down" to see the next level of the detail.

As can be seen that a top down design methodology can be used to define various formulae with the properties of each level being checked to ensure they have consistent inheritance. Equally, a bottom up design methodology could be adopted. This allows for a multi-level model to be created, which can be graphically represented and defined. Modular building of higher level functions can also be conducted.

Referring to Figure 12, inheritance can be used to restrict the options of variables/constants available for selection. That is, if due to the selection of another variable, the variable being selected must have certain properties the selection of the variable may be restricted to those variables that have the required properties. Other variables can be "greyed out" and made unavailable for selection or simply not displayed in the list of options. An example of property inheritance is dimensional inheritance. In this case each variable must have at least one dimension, for example distance, time, mass, etc.

In the example provided in Figure 12, the formula being defined is: A x B = C. The variables are given properties including type, dimension and units. In this case, the result variable C is a real variable and has a dimension of mass.distance and units of kilogram meters (kgm). A has the properties of a real number with its dimension being mass and units kilograms (kg). B has been defined with the properties of being either a real or an integer number. This may have been the result of defining the properties of A and C by virtue of inheritance B must be either a real or an integer number. If, for example, C had been defined as an integer and A had been defined as an integer, then B would, out of necessity, have been an integer. In addition, because C has the dimension mass.distance (kgm) and A has the dimension mass (kg), by necessity, B must have the dimension distance (m). Likewise, if A and B had been defined first, A being kg and B being m, C out of necessity would have had to have been kgm by virtue of the dimensions and units of each of the inputs and the effect of the operator.

Since B has the properties of being a real or an integer number and the dimension is in metres, the actual variable type meeting those inherited properties will restrict the type of input variables that B may be. A pull down menu 110 is shown that lists a number of variables types that have been previously entered. Variables types that meet the properties are shown in a normal font and variables not meeting the properties are shown "greyed out". Of course, an alternative may be to simply not display the variable types that are not available for selection.

In this case, the variable type "SHAFT DIAMETER" or the variable type "LEVER ARM LENGTH" may be selected. Whichever of these variables types is selected from the pull down menu will then become the variable type of the variable B. The variables able to be selected from may be obtained from a variety of sources, such as databases or a library of variables, and not just manually entered. The pull down menu may contain a list of textual variable names, as shown, however icons representative of the variables may also be used in the pull down menu. Other suitable selection means may also be employed.

The same process can apply to other input variables such as A, as well as, result variables such as C. The order of selection of the variables will necessarily determine the properties of subsequent variables (or constants).

To check the progress of the definition of the formula, a facility may be included where a pointer is placed over a part of the formula being defined and a window will appear that displays the formula defined thus far (as shown in Figures 13A and 13B).

In Figure 13A, it can be seen that the pointer is placed above one of the inputs of a look up table. Beneath the pointer, a box appears that shows that the properties of the input needs to be a "Grade", which is a number greater than or equal to zero. It can also be seen that grade is given the property of grams per tonne.

In Figure 13B, the pointer is shown pointing to Schedule result variable, which shows the definition of the formula thus far. In this case, the formula so far defined is: if (look up table: "monthly cost periods" with the data in the variable: "period" equals "monthly costs lower" value), the result is the lower value (L), otherwise the result is upper value (U). It is also possible to check the syntax of the formula entered.

Referring to Figure 14, in this example the formula for shear stress is defined. "SHEAR STRESS" = ("RADIUS" x "ANGLE OF TWIST" x "SHEAR MODULUS OF ELASTICITY") ÷ "LENGTH". In this example the icons that represent the objects displayed are different. Shear modules of elasticity 120 is a look up table operator that receives an input, which is a metal alloy input variable 122, into which data is received. The look up table may be a component that references data stored within the GFDT. Alternatively the

look up table may be a component that references external data. For example, the external data may be in the form of a spreadsheet. The GFDT can be provided with a plug-in that enables data to be transferred from external software applications. A typical spreadsheet application would be Microsoft Excel. The GFDT, via the plug-in, can communicate with Excel to retrieve data in an Excel spreadsheet. The look up table may be derived from data supplied by a material supplier regarding the sheer modulus of elasticity for each metal alloy. The data may be sourced from a number of material suppliers. The data may be sourced from a database 124 provided by the material supplier. The database may be accessed through a computer network, such as the Internet. Therefore, the operator 120 may involve a database query that accesses the database 124. The database 124 may be a distributed database. Thus, the method of graphically defining a formula may also be used to define a database query, with a database query being a particular type of formula defined in accordance with the present method.

Referring to Figure 15, in this example the result variable E is defined as: E = C (A x B) + D, where C is another function. If C has not been defined it is referred to as an empty component. The properties of C may be defined by other properties of the formula shown in Figure 15. That is, the variables A, B, D and E will, to some extent, define the properties that the input and the output of the operation C must have. C can also be progressively defined so that if it is desired to add further inputs "f", "g" and "w" into C as indicated in Figure 16, these can be added as the formula is progressively defined. When it is desired to add a new input to the component C a selection to add an input to the component is selected and a new input or output created as desired. Thus further named variables "f", "g" and "w" can then be connected to the respective inputs of component C. Likewise, additional output can be added as required.

When it is desired to specify the functionality of the component C, C can be opened by drilling down to the next level as indicated in Figure 17. Another layer of functionality of C is defined as indicated in the box 126. Alternatively functionality of C may be drawn from a library of components. A basic library may be provided with the GFDT. Alternatively a library may be provided on-line. A component that fulfils the requirements can be searched for through the Internet. Once a component that fulfils the requirements is found it may be inserted into the formula. As shown in Figure 18 a component residing on a different

machine is found that fulfils the properties required of the component and performs the required functionality as described in a description attached to the component and this can be forwarded through a computer network such as the Internet to the local instances of the GFDT for insertion into the formula being created.

Referring to Figure 16, it can be seen that C is a function of the product of A and B and also receives the inputs f, g and w. That is C = (A x B, f, g, w). Referring to Figure 17, when C is "drilled down" it can be seen that the product of A x B is connected by connector 128 to a temporary variable "p" which is then compared to the input variable "w" as indicated by the equals sign ("=") operator. The RESULT operator tests to see if the comparison is true, in which case the result of the LOOKUP table, which receives inputs "f" and "g", is provided. Otherwise, if the comparison results in false the result is "0". The result of the RESULT operator is then multiplied as indicated by the multiplication operator ("x"), with a constant "k" and then provided to the output 130. This is then provided to input the addition operator ("+") where it is added to the variable D to provide the result E.

Empty components can be provided as a place holder for a fully defined component. An empty component is yet to have its functionality between its inputs and outputs defined. The user can place an empty component in a design environment and define its inputs and outputs. The functionality of the component can then be defined later as required or the definitions of the inputs and outputs can form a search criteria for searching for components that can perform the function from a library. Further search criteria can be provided such as key words, higher class level structure information and so on. Empty components further assist in top down design methodologies.

It is desirable to use colour coding to assist in the graphical representation. For example, one colour, say blue, can represent variables. Another colour, say green, can represent operator and yet another colour could represent constants. Input can be shaded lighter, say light blue, and outputs shaded darker, say dark blue. This assists in visualising the representation of the formula, particularly when complex multi-level formulas are represented. Other visual representations, such as icons can be used to represent variables, constants and operators, such as is used in the figures.

Many of the objects in a GFDT model will be provided with properties that the user can view and/or modify. As shown in Figure 19, an accumulator component is defined that accumulates an input received 132 from an Excel spreadsheet. The discrete output from the spreadsheet 132 is added to the last sub-total (Running Total) by operator 134 and stored by memory operator 136. Then at the end of a row of data from the spreadsheet, the result from the "Running Total" variable 138 is gated by gate 140 to a "Sum" variable 142 which then passes it back into another spreadsheet 144. The 146 connector is shown highlighted. In the secondary window 30, properties of the highlighted object are shown. The properties of a highlighted connector 146 are that its name is "Connector 1", it is of a "numeric" type and it provides units in "metres" for the dimension of "length". A wizard can be provided to assist in the selection of properties for objects in a similar manner to that provided in products such as Delphi or Visual Basic. A number of predefined data types may be provided which can then be further extended by the user depending on their needs. Or the user may buy or obtain a library of extended data types. For example a data type that relates to complex numbers may be represented as two independent numbers being the real and the imaginary components of the complex number while in another example the output of an engine might be represented by power, torque and angular velocity as sub-components of the general data type "output of an engine".

In Figure 19 the user can change values of the properties simply by editing the values of the fields or by making a selection from a drop down list.

Complex data types will also need to define the operations that can be performed on them. Operators may use existing components as the representation in the model. For example, the addition of complex numbers requires a different set of operators from the addition of real numbers. They both can be represented by a "plus" component however the way in which the component deals with the data type depends on the nature of the data type. When a connection is made to a component a negotiation process takes place whereby where ever there are existing properties of a connector data type or input or output of an operator or named connection, the data type connection must be consistent. Thus through a process of negotiation between each of the objects a correct data type can be selected.

Referring to Figure 20, an addition component 148 may be able to perform several methods of

addition depending on the data type. This is known as data overloading. The data type may be provided with an intrinsic methodology for dealing with different types of data as indicated in the table, however additional methodologies of dealing with different data types can be provided. Predefined schema defining the format for adding additional definitions can be provided so that further data types can be added and dealt with by components.

Where data is provided in a particular type of unit often in many cases a conversion factor may be required, for example one unit may be in seconds and another unit may be in minutes for the same dimension of time. Conversion may then be required. A conversion factor may be required again to convert for example a speed in kilometres per hour into metres per second. Further, dimensional definitions require the combination of fundamental dimensions of a unit. The fundamental definitions being length, time, mass, charge etc. For example, acceleration is length x time^{-2}. Other properties may also be provided to objects. Examples of other properties include security information – such as the type of user that can use the information and encryption information; version information – the version number and how long the version is valid for; certification information – the data type function identified as coming from a certified source; charging information – for use in pay as you use and subscription access to data or objects; location information – such as an IP address and file name for the location of data or objects; and broker information – information on the manager of a component.

Each model, named connection and component definition will have its own namespace. This means that the names of particular objects used in a particular model or component definition are unique to that model or component. Objects in different models or different component definitions, especially named connections, sharing a common name are not the same object. However the namespace can be modified. This is akin to local variables in many programming languages where the name of the variable only applies within a particular space. This is to prevent two objects that have the same name but are unrelated being confused for one another.

Some operations are extrinsic, meaning that they are performed outside the formula engine. They are performed by making a call usually through an application specific plug-in to the external component along with any input required, whereupon the result is fed back to the

engine (via the plug-in) for further computation of the formula. Extrinsic components are made available in the GFDT by importing a component type definition file. This then provides a definition for the component for use by the GFDT. A component type will typically have a set of input and output connectors which a component may then be created according to the component type.

When componetising a group of components all the connections that lead outside the component will create input Input and Output objects for named component, if a named connection is used anywhere outside the component. If a named connection is only used inside the component then the name space of the named connection will become the component. It will no longer be available outside the component. This method provides a manner for hiding detail and forces a user to follow a more structured approach when building a model. Complex components are effectively functional blocks with well defined interfaces. The user then finds it difficult to build "spaghetti code" models as defined interfaces and functional blocks are provided.

Some components are extrinsic components, such as Microsoft Excel word spreadsheet. This can be wrapped into a GFDT component and used when defining a model. When the definition is executed to calculate the formula it actually uses the Excel spreadsheet. The engine will communicate with the spreadsheet via Excel in order to pass data into and out of the component. Remote components execute in a different engine from the main model. From the point of view of the local function engine they are "black boxes" with the internal workings unknown.

Where a connection is desired between two elements, such as components, a named connection can be placed on the page and assigned a label. Properties can then be assigned to the named connection. The named connection does not have to be connected another object at this stage but can be used elsewhere by placing another named connection object on the design page and selecting the same label, such as from a drop down list. This will not create a new object but rather allows a second instance of the same object to exist, provided the second instance is in the same namespace as the first. As indicated by Figure 21, a logical connection between the two named connections is formed so that when a connector of another object such as a component is connected to the named connection, every instance of the

named connection can inherit the properties of the component. Likewise, when data is provided to the named connection in one location it will also be provided at the other location because of the logical connection. Furthermore other instances of the logical connection will transfer the properties to other components or connections that are related to the named connection. Typically named connections are used to identify incoming data and outgoing data as well as intermediate values in the model. A named connection is akin to a variable in a convention software programming language in that it may appear in many places in the model and carries a value which it may vary during the execution of the code. An assignment of a value to a named connection or an input of an operator can only be done in one place because the value is passed on to the other instances of the named connection. Therefore only one output connection is allowed to be connected to an input of an operator or a named connection.

Referring to Figure 22, where a component A provides an output to a named connection on one page and then on another page the named connection provides an input to component B the effect is the creation of a logical connection between the output of component A and the input of component B. A logical connection is in essence a communication of data through various components of the model. In terms of the logical definition, named connections are irrelevant as it is purely a meshed network of components and connections therebetween.

During the connection of two objects checking, matching and adoption takes place. For example, if dimensions have been defined for both connections then they must be the same in order for the connection to be allowed. Units need not be identical as long as a conversion factor can be determined. If any of the properties are undefined at either end then they can be adopted or a different property can be discarded in order that the properties remain consistent. Once a consistent data type has been negotiated between the connections there is then a check performed to see that the component has functions available to work with the data type. If there is not then there is a check to see whether any converters may be made to make then compatible. For example, a number may need to be converted into a text string.

It is to be noted that an output may be connected to many inputs and thus negotiation may not simply be between two connections. Each time another input is added to the set of connections further negotiations will take place to ensure that the data types are consistent and

if necessary a data type may be changed to accommodate the new connection. The component manager will search a set of available overload functions, those that work with possible data types. In addition, available data converters may be checked to see whether the data can be converted to an acceptable data type. If this is not possible then the connection manager may try to renegotiate with the components that a connection connects to. This can lead to all the properties of connections being renegotiated in the entire model. However the user may be able to limit the extent of this process by defining a distance from a new connection that negotiations are allowed to proceed over.

Referring back to Figure 19, the accumulator model receives data from an input spread sheet. This component represents an Excel spread sheet containing a column of numbers. Each number is represented on the output connector sequentially, with the next one becoming available only once all the connections have used the previous value. This component is connected to an input of a addition component. The other input is connected to a "Running Total" named connection. The output of the addition component is connected to a memory component that maintains at its output the value presented to its input. All other components reset the outputs (to undefined) for each iteration of data from the input spread sheet. The output of the memory component is provided to the named connection "Running Total" that represents an intermediate "Sum" during the calculation. The "Running Total" named connection is provided to an input of a switch component. This component blocks the transfer of the input value to the output connector until the value at a gate input connector (on the bottom) is true or there is no more data. In this way the output only shows the sum of the calculation not any intermediate values. The output of the switch component is provided to a "Sum" named connection. This value represents the sum of the accumulation which is written to the output of an output spread sheet. The output spread sheet represents another Excel spread sheet to which the result of the calculation will be written. In Appendix 1 an example of an extensible mark up language (XML) description of the formula defined is shown. It is noted that a section of data relating to the definition of icons is not shown. This defines the recorded description of the formula as shown in the window. In Appendix 2 a logical description in XML is shown that defines the components and the connections thereto.

Referring to Figure 23, a more complicated formula is shown which receives data from a

spread sheet 150 of "List of Items to Pick" which is provided to a named connection "Item ID" 152. This is then provided in turn to a look up component 154 that looks up the Name, Gross Price, GST, Net Price and the Cost of each item within a spreadsheet. The spreadsheet is shown in Figure 24 which forms the basis of a wrapped component with the values of each of the outputs being calculated by a spreadsheet formula being based on a database table of 30 components. Further calculations are performed on the "Name" 156, "Gross Price" 158, "GST" 160, "Net Price" 162 and "Cost" 164 by summing the "Gross Price" to produce a "Total Gross Price" 166, summing the "GST" to produce a "Total GST" 168, determining the "Margin" 170 by subtracting the "Cost" from the "Net Price", summing the individual "Margin" to produce a "Total Margin" 172. Then the "Name", the "Gross Price", the "Total Gross Price", the "Total GST" and the "Total Margin" are provided into another Excel spreadsheet 174. The processing of the formula can be seen by entering values 176 into the first spreadsheet 150. In processing this formula each item ID is used to retrieve data on the item using the database retrieval component which provides outputs as each item is processed and another entry is created in the final database 154 as can be seen by 178. The result of the calculation 180 is returned to the final spreadsheet by component 174.

A function engine can perform the calculation by receiving the logical definition of the formula which describes the function in terms of its objects and relationship between them and can then use this definition of the formula to process data received and thus calculate the result of the formula when provided with the input data.

The method of graphically defining a formula as defined by the present invention has a number of advantages. It provides a simple method of building a model based on a number of formulae that can be built using either top down or bottom up design methodology. The method can check to see that variable properties are inherited correctly and provide a warning if an error would result. The method provides an extremely versatile and simple method of creating multilevel models used to manipulate query data. The representations are easily understood and so the checking/auditing of the accuracy of a formula is more easily achieved.

As will be appreciated by the skilled addressee, modifications and variations can be made to the present invention without departing from the basic inventive concept, such as: various graphical user interface technologies can be used with this methodology, such as pull down

menus, manipulation of graphics and information bubbles.

Such modifications and variations are intended to be within the scope of the present invention, the nature of which is to be determined from the foregoing description.

APPENDIX 1

```
<?xml version = "1.0"?>

<Xemplex>
   <Package Name = "Internal" Description = "">
      <Type Name = "Item List" Description = "Z:\Demo\Price List NQ.xls" Type =
"InputData">
         <Application Name = "Excel97" Workbook = "Z:\Demo\Price List NQ.xls"
Worksheet = "Items to Pick" Start = "2"/>
         <Icon>

<![CDATA[NDI0REU2MDQwMDAwMDAwMDAwMDAzNjAwMDAwMDI4MDAwMDAwMTQwMDAwMDAxNDAwMDAwMDA

0ODQ4NDg0ODQ4NDg0ODQ4NDg0ODQ4NDg0ODQ4NDg0ODQ4NDg0ODQ4NDg0ODQ4NDg0]]>
         </Icon>
         <Outputs>
            <Output Name = "ItemID" Type = "Number" DataType = "Number" Line =
"A"/>
         </Outputs>
      </Type>
      <Type Name = "Sum" Description = "Z:\Demo\Price List NQ.xls" Type =
"OutputData">
         <Application Name = "Excel97" Workbook = "Z:\Demo\Price List NQ.xls"
Worksheet = "Items to Pick" Start = "2"/>
         <Icon>

<![CDATA[NDI0REU2MDQwMDAwMDAwMDAwMDAzNjAwMDAwMDI4MDAwMDAwMTQwMDAwMDAxNDAwMDAwMDA

0ODQ4NDg0ODQ4NDg0ODQ4NDg0ODQ4NDg0ODQ4NDg0ODQ4NDg0ODQ4NDg0ODQ4NDg0]]>
         </Icon>
         <Inputs>
            <Input Name = "Total" Type = "Number" Line = "F"/>
         </Inputs>
      </Type>
   </Package>
   <ComponentPackages>
      <Package Filename = "Z:\Delphi\ComponentManager\Basic.xmp"/>
      <Package Filename = "Z:\Delphi\ComponentManager\Standard.xmp"/>
   </ComponentPackages>
   <Model>
      <Page Name = "NewPage">
         <Component Name = "Add0" ComponentPackage = "Standard" ComponentType =
"Add" Left = "161" Top = "83">
            <Connector Type = "Input" Name = "Connector #0" UID = "12" Edge =
"Left"/>
            <Connector Type = "Input" Name = "Connector #1" UID = "13" Edge =
"Left"/>
            <Connector Type = "Output" Name = "Connector #2" UID = "14" Edge =
"Right"/>
         </Component>
         <Component Name = "Memory3" ComponentPackage = "Basic" ComponentType =
"Memory" Left = "266" Top = "85">
            <Connector Type = "Input" Name = "MI" UID = "15" Edge = "Left"/>
            <Connector Type = "Output" Name = "MO" UID = "16" Edge = "Right"
Initial = "0"/>
         </Component>
         <Component Name = "Block4" ComponentPackage = "Basic" ComponentType =
"Block" Left = "485" Top = "84">
```

```
            <Connector Type = "Input" Name = "BI" UID = "17" Edge = "Left"/>
            <Connector Type = "Output" Name = "BO" UID = "18" Edge = "Right"/>
            <Connector Type = "Gate" Name = "BG" UID = "19" Edge = "Bottom"/>
         </Component>
         <Component Name = "Item List5" ComponentPackage = "Internal"
ComponentType = "Item List" Left = "27" Top = "86">
            <Connector Type = "Output" Name = "ItemID" UID = "20" Edge =
"Right">
               <Attributes Line = "A"/>
            </Connector>
         </Component>
         <Component Name = "Sum5" ComponentPackage = "Internal" ComponentType =
"Sum" Left = "716" Top = "77">
            <Connector Type = "Input" Name = "Total" UID = "21" Edge = "Left">
               <Attributes Line = "F"/>
            </Connector>
         </Component>
         <NamedConnection Name = "Running total" UID = "22" Left = "400" Top =
"101"/>
         <NamedConnection Name = "Sum" UID = "23" Left = "620" Top = "102"/>
         <VisibleConnection EndpointA = "15" EndpointB = "14"/>
         <VisibleConnection EndpointA = "16" EndpointB = "22"/>
         <VisibleConnection EndpointA = "12" EndpointB = "22">
            <Vertex Top = "92" Left = "160"/>
            <Vertex Top = "92" Left = "115"/>
            <Vertex Top = "33" Left = "115"/>
            <Vertex Top = "33" Left = "400"/>
         </VisibleConnection>
         <VisibleConnection EndpointA = "17" EndpointB = "22"/>
         <VisibleConnection EndpointA = "18" EndpointB = "23"/>
         <VisibleConnection EndpointA = "13" EndpointB = "20"/>
         <VisibleConnection EndpointA = "21" EndpointB = "23"/>
      </Page>
   </Model>
</Xemplex>
```

APPENDIX 2

```
<?xml version = "1.0"?>

<Xemplex>
   <Executable>
      <Component Name = "Add0" Type = "Generic" Application = "Xemplex"
Operation = "Add">
         <Input Name = "Connector #0" Type = "Input"/>
         <Input Name = "Connector #1" Type = "Input"/>
         <Output Name = "Connector #2" Type = "Output"/>
      </Component>
      <Component Name = "Memory3" Type = "Generic" Application = "Xemplex"
Operation = "Memory">
         <Input Name = "MI" Type = "Input"/>
         <Output Name = "MO" Type = "Output" Initial = "0"/>
      </Component>
      <Component Name = "Block4" Type = "Generic" Application = "Xemplex"
Operation = "Block">
         <Input Name = "BI" Type = "Input"/>
         <Output Name = "BO" Type = "Output"/>
         <Gate Name = "BG" Type = "Gate"/>
      </Component>
      <Component Name = "Item List5" Type = "InputData" Application = "Excel97"
Workbook = "Z:\Demo\Price List NQ.xls" Worksheet = "Items to Pick" Start = "2">
         <Output Name = "ItemID" Type = "Number" Line = "A"/>
      </Component>
      <Component Name = "Sum5" Type = "OutputData" Application = "Excel97"
Workbook = "Z:\Demo\Price List NQ.xls" Worksheet = "Items to Pick" Start = "2">
         <Input Name = "Total" Type = "Number" Line = "F"/>
      </Component>
      <Connection From = "Add0" Output = "Connector #2" To = "Memory3" Input =
"MI"/>
      <Connection From = "Memory3" Output = "MO" To = "Add0" Input = "Connector
#0"/>
      <Connection From = "Memory3" Output = "MO" To = "Block4" Input = "BI"/>
      <Connection From = "Item List5" Output = "ItemID" To = "Add0" Input =
"Connector #1"/>
      <Connection From = "Block4" Output = "BO" To = "Sum5" Input = "Total"/>
   </Executable>
</Xemplex>
```

The claims defining the invention are as follows:

1. A computer-implemented method of graphically defining a formula, said method including:

providing a first operator object for defining a method of manipulating at least one input to produce at least one result;

displaying a graphical representation of the first operator object;

providing a first variable object for containing data;

receiving an input from a user to relate the variable object to one of the inputs or one of the results of the first operator object;

displaying a graphical representation of the first variable object and its relation to the first operator object; and

recording a logical description of the relationship between objects;

whereby the formula is defined by the logical description.

2. A computer-implemented method of graphically defining a formula, said method including:

providing a variable object for containing data;

displaying a graphical representation of the variable object;

providing a first operator object for defining a method of manipulating at least one input to produce at least one result;

receiving an input from a user to relate one of the inputs or one of the results of the first operator object to the variable object;

displaying a graphical representation of the operator object and its relation to the variable object; and

recording a logical description of the relationship between objects;

whereby the formula is defined by the logical description.

3. A computer-implemented method of graphically defining a formula, the method according to either claim 1 or claim 2, further includes the steps of:

providing one or more further variable objects;

receiving further inputs from the user to relate each further variable object to one of the inputs or one of the results of the first operator object;

displaying a graphical representation of the further variable objects and their relation to the operator object.

4. A method according to any one of claims 1, 2 or 3, wherein the method further includes the steps of:

providing one or more further operator objects;

receiving further inputs from the user to relate each variable objects to one of inputs or one of the results of the further operator objects;

displaying a graphical representation of the further operator objects and their relation to the variable objects.

5. A method according to any one of claims 1 to 4, wherein each variable object is selected from: an input object for providing data from a data source; an output object to provide data to a data destination; or a connection object for passing data from one operator object or another.

6. A method according to any one of claims 1 to 5, wherein a connection object represented as a link between the operator objects.

7. A method according to any one of claims 1 to 6, wherein each variable object may be provided with a variable label.

8. A method according to any one of claims 1 to 7, wherein each operator object may be provided with a operator label.

9. A method according to any one of claims 1 to 8, wherein the logical description of the formula is defined by the logical relationship between the objects.

10. A method according to any one of claims 1 to 9, wherein a graphical definition of the formula is recorded that defines the graphical display of the relationship between objects.

11. A method according to any one of claims 1 to 10, wherein the method

includes the step of storing information describing the logical definition.

12. A method according to any one of claims 1 to 10, wherein the method includes the step of storing information describing the graphical definition.

13. A method according to any one of claims 1 to 12, wherein two or more related operator objects may be grouped such that the grouping defines a grouping operator object, wherein variable objects crossing the border of the grouping and connecting to inputs of operator objects in the group become the inputs of the grouping object component and variable objects crossing the border of the grouping and connecting to results of the operator objects in the group become results of the grouping operator object.

14. A method according to claim 13, wherein inputs and results of operator objects in the group not linked to another object become inputs and results, respectively, of the grouping operator object.

15. A method according to claim 13 or 14, wherein the graphical representation of the grouped objects is replaced by a graphical representation of the grouping operator object and the graphical representation of links to the contents of group are replaced with graphical representations of links to the representation of the grouping object.

16. A method according to claim 13 or 15, wherein the logical definition of the formula defined includes the contents of the grouping operator object.

17. A method according to claim 13 or 16, wherein the graphical definition of the overall formula displayed excludes the contents of the grouping operator object.

18. A method according to claim 13 or 17, wherein the contents of the grouping operator object may be graphically represented separately from the overall graphical representation of the formula.

19. A method according to any one of claims 1 to 18, wherein variable objects may be attributed with properties that define the type of data they can hold.

20. A method according to claim 19, wherein each input and result of an operator object may be attributed with properties that define the type of data that the operator object expects to receive and be able to produce, respectively.

21. A method according to claim 20, wherein a variable object may inherit the properties from the properties of another variable object that has already been defined and is related by an intervening operator object.

22. A method according to claim 20, wherein a variable object may inherit the properties from the properties of an operator object input or result that has already been defined and to which it is related.

23. A method according to any one of claims 19 to 22, wherein an input or result of an operation object may inherit the properties from the properties of a variable object that has already been defined and to which it is related.

24. A method according to any one of claims 19 to 23, wherein the method includes a step of checking that the properties of objects with already attributed which are being related match.

25. A method according to claim 7, wherein a library of labelled variable objects is predefined.

26. A method according to claim 8, wherein a library of labelled operator objects is predefined, each labelled operator object's method of manipulating its input/s to produce its result/s also being predefined.

27. A method according to claim 7 or 25, wherein the variable label of a variable object may be selected from a list of predefined variable labels.

28. A method according to claim 19, wherein each variable label may be attributed with properties that define the type of data a variable object labelled with the label

can contain.

29. A method according to claim 19, wherein the selection of a variable label attributes the properties associated with the label to the variable object.

30. A method according to claim 29, wherein the properties attributed to a variable object limit the selection of labels available to be selected.

31. A method according to any one of claims 1 to 30, wherein the operation object is at least one of addition, subtraction, multiplication, division, a look-up table and conditional operation.

32. A method according to any one of claims 1 to 30, wherein the operator object may be a multiple stage operation containing a plurality of simple operators linked to perform a more complex operator.

33. A method according to any one of claims 1 to 30, wherein in one form, the operator object is a query of the database.

34. A method wherein the first operator object performs a write to a database.

35. A method according to claim 8, wherein the operator label of an operator object may be selected from a list of predefined operator labels.

36. A method according to claim 35, wherein each operator label may be attributed with properties that define the type of data that inputs and results of a labelled operator object can receive or provide, respectively.

37. A method according to claim 36, wherein the selection of an operator label attributes the properties associated with the label to the operator object.

38. A method according to claim 37, wherein the properties attributed to an operator object limit the selection of labels available to be selected.

39. A method according to any one of claims 1 to 38, wherein the logical definition may be used by a run time engine to put into operation the defined formula, wherein data provided to each variable object is linked to an input of an operator object, whereby the data becomes operands of the formula, each operator represented by the operator object becomes the operator of the formula and each result of the operator object becomes the next operand of the next operator or the final result/s of the formula, whereby computation of the formula can be conducted to produce a formula result.

40. A method according to any one of claims 1 to 39, wherein a namespace may be defined for each variable, whereby the data in a logical variable represented by the variable object is the same for each occurrence of the variable object within the namespace.

41. A method according to claim 40, wherein the name space is by default global to the formula being modelled.

42. A method according to claim 40, wherein a logical connection is created between each occurrence of a labelled variable object within a namespace.

43. A method according to claim 40, wherein a graphical link may be displayed showing the logical connection between occurrences of labelled variable objects.

44. A method according to any one of claims 1 to 43, wherein a namespace may be defined for each operator object, whereby the operation of a logical operator represented by the operator object is the same for each occurrence of the operator object within the namespace.

45. A method according to claim 44, wherein the name space is by default global to the formula being modelled.

46. A method according to claim 13, wherein a grouped operator object may be used more than once with the definition of the grouped operator object being applied to the logical definition of the formula.

47. A method according to claim 19, wherein the properties of a label include type, units and dimension.

48. A method according to claim 10, wherein the graphical definition is described in XML.

49. A method according to any one of claims 1 to 48, wherein the logical definition is described in XML.

50. A method according to any one of claims 1 to 49, wherein each operator object includes a plurality of definitions of the operation performed by the operator represented by the operator object, each definition being for a separate type of data able to be manipulated by the operator.

51. A method according to any one of claims 1 to 50, wherein the operator object is graphically represented as a component having one or more inputs and one or more outputs, the component having an indicator representative of the operator represented.

52. A method according to any one of claims 1 to 51, wherein the operator object may be an empty component that is representative of a operator with methodology of manipulation inputs to produce results yet to be defined.

53. A method according to any one of claims 1 to 52, wherein the empty component is used to form criteria for searching for a suitable operator object that has a suitable defined methodology.

54. A method according to any one of claims 1 to 53, wherein a library of objects is provided.

55. A method according to any one of claims 1 to 53, wherein objects may be externally sourced.

56. A system for graphically defining a formula, comprising:

a computer including a display screen and a user input means;

means for providing a first operator object for defining a method of manipulating at least one input to produce at least one result;

means for displaying a graphical representation of the first operator object on the screen;

means for providing a variable object for containing data;

means for receiving an input from the user input means to relate the variable object to one of inputs or one of the results of the first operator object;

means for displaying a graphical representation of the first variable object and its relation to the operator object on the screen;

whereby the formula is defined by the relationship between the objects.

57. A computer program for controlling a computer for graphically defining a formula, said computer program causing the computer to undertake step including:

providing a first operator object for defining a method of manipulating at least one input to produce at least one result;

displaying a graphical representation of the first operator object on a computer screen;

providing a variable object for containing data;

receiving an input from a user input means to relate the variable object to one of inputs or one of the results of the first operator object;

displaying a graphical representation of the first variable object and its relation to the operator object on the screen;

whereby the formula is defined by the relationship between the objects.

58. A computer readable medium for storing a computer program as defined in claim 57.

59. A computer-implemented method of graphically defining a formula for manipulating input data to produce a result, said method including:

providing at least one variable for containing data;

providing at least one operator defining the method of manipulating the input

data to produce the result;

displaying a list of the variables for a user to select a result variable therefrom;

receiving a selection of the result variable from the user for containing the result of the manipulation of the input data;

displaying a graphical representation of the selected result variable;

displaying a list of the operators for a user to select an operator therefrom;

receiving a selection of an operation from the user;

displaying a graphical representation of the selected operation;

displaying a list of inputs for containing the input data for a user to select at least one input therefrom, the inputs being either said variables or one or more constants;

receiving a selection of at least one input from the user;

displaying a graphical representation of the selected input,

whereby the formula is defined by the selected result variable being equal to the manipulation of selected input(s) by the selected operation.

60. A computer-implemented method of graphically defining a formula for manipulating input data to produce a result, said method including:

providing at least one variable type, said variable type having pre-determined properties;

providing at least one operation defining the method of manipulating the input data to produce the result;

displaying the variable types for a user to select a variable type therefrom;

receiving a selection of the variable type from the user;

receiving a name for the selected variable type;

displaying a representation of the named variable;

displaying a list of operations for a user to select an operation therefrom;

displaying a graphical representation of the selected operation;

receiving a selection of an operation from the user;

receiving input from the user so as to associate the selected variable with the selected operation so that the selected variable is either an input variable or a result variable;

where the selected variable is associated to be a result variable, receiving from the user a selection of at least one of either an input variable or an input constant and a name for the input variable or the input constant, displaying a graphical representation of the input

variable(s) and/or input constant(s);

where the selected variable is an input variable, receiving a name for an output variable, displaying a graphical representation of the output variable;

whereby the formula is defined by the result of the manipulation by the selected operation of the input data in the input data variable or input constant provided to the result variable.

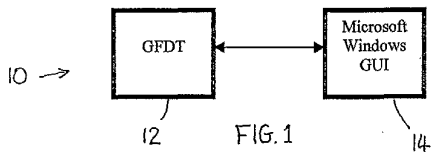

GFDT
Microsoft Windows GUI
10
12
14

FIG. 1

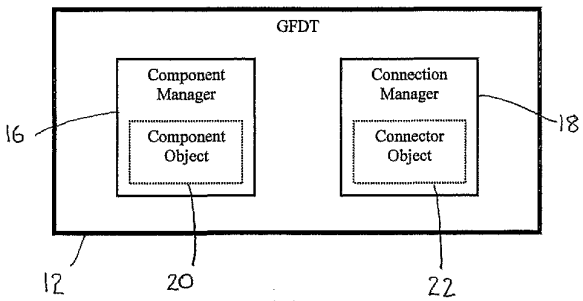

GFDT
Component Manager
Component Object
Connection Manager
Connector Object
16
18
12
20
22

FIG. 2

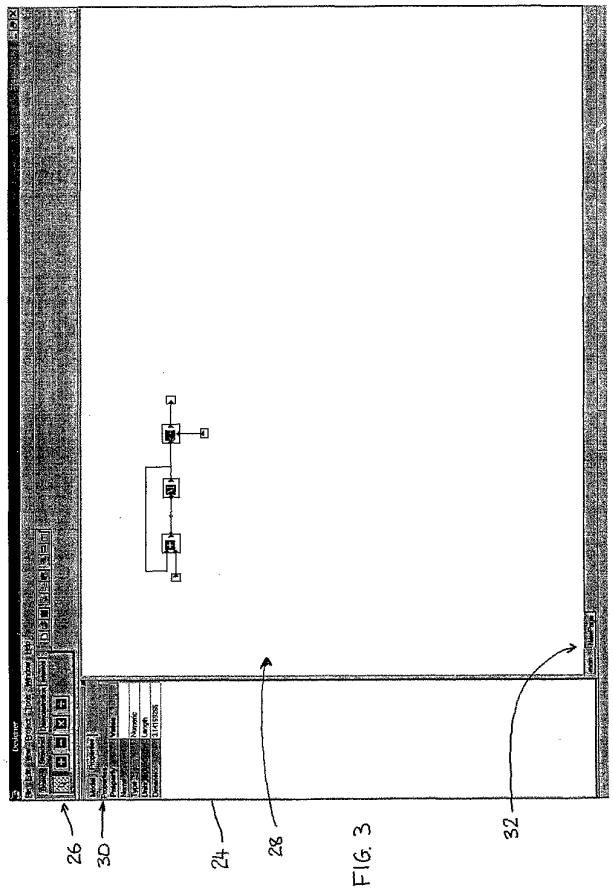


FIG. 3

Fig. 4A

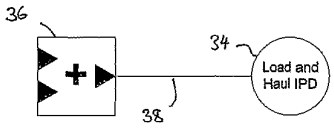


Fig. 4B

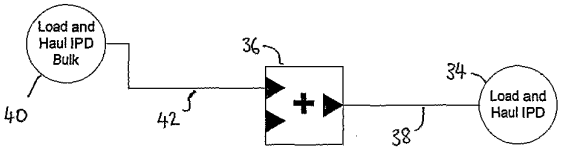


Fig. 4C

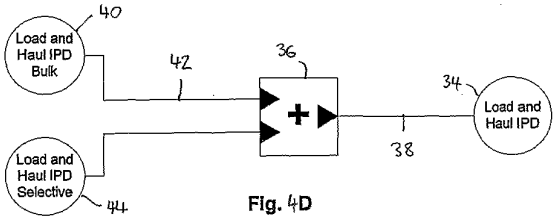


Fig. 4D

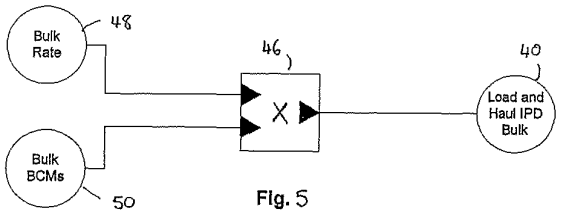


Fig. 5

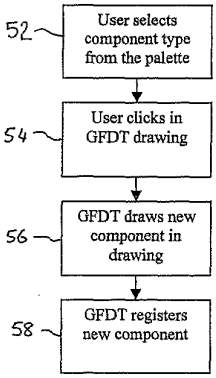


FIG. 6

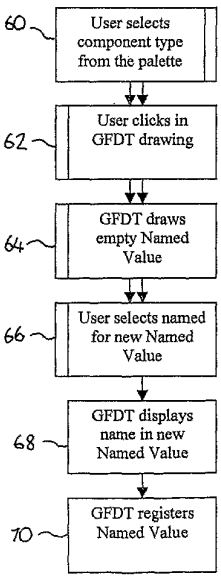


FIG. 7

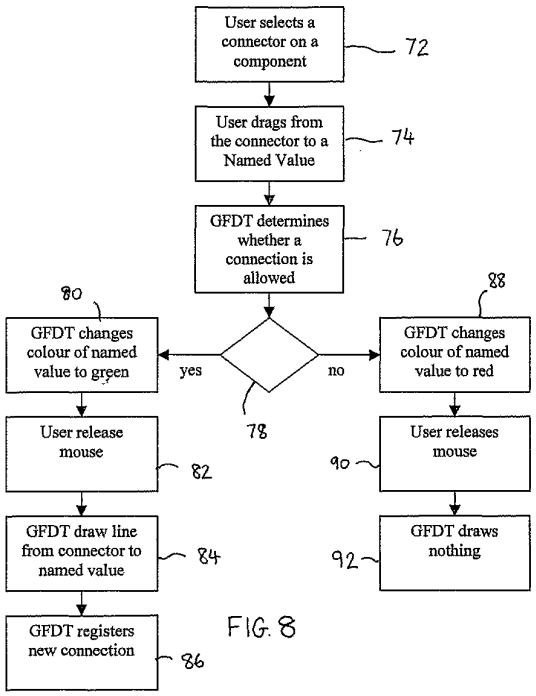

User selects a connector on a component
72
User drags from the connector to a Named Value
74
GFDT determines whether a connection is allowed
76
80
88
GFDT changes colour of named value to green
yes
no
GFDT changes colour of named value to red
78
User release mouse
82
User releases mouse
90
GFDT draw line from connector to named value
84
GFDT draws nothing
92
GFDT registers new connection
86

FIG. 8

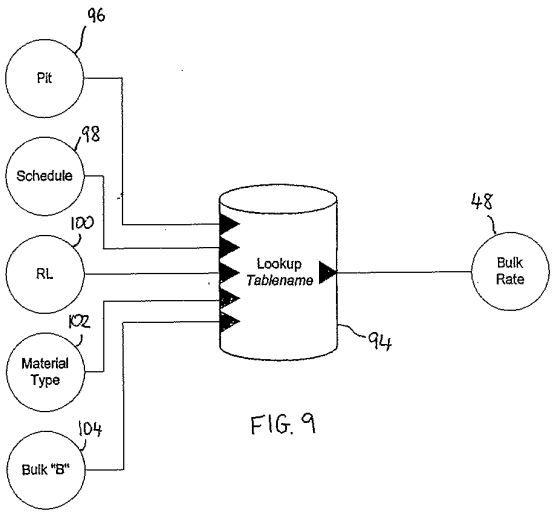

96
Pit
98
Schedule
100
RL
102
Material Type
104
Bulk "B"
Lookup
Tablename
94
48
Bulk Rate

FIG. 9

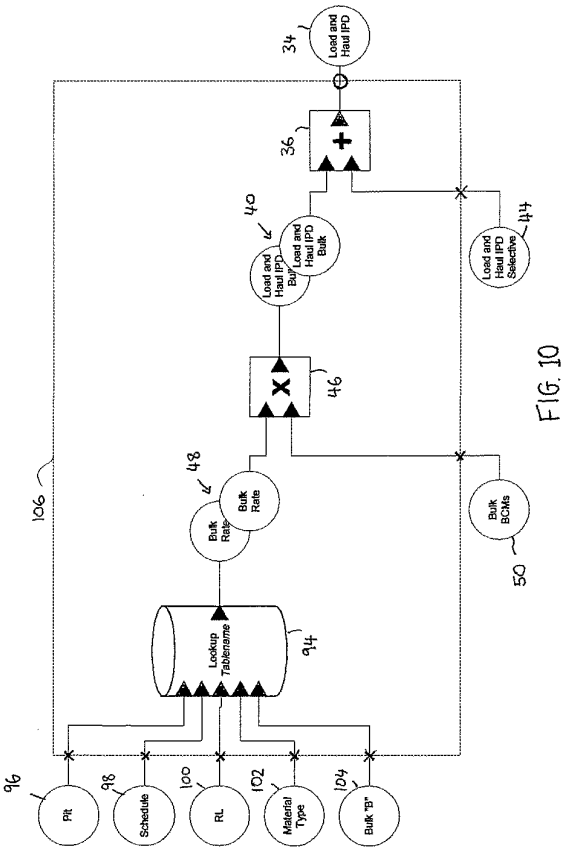

34
Load and Haul IPD
36
40
Load and Haul IPD Bulk
Load and Haul IPD Bulk
44
Load and Haul IPD Selective
46
48
Bulk Rate
Bulk Rate
106
Bulk BCMs
50
Lookup Tablename
94
96
Pit
98
Schedule
100
RL
102
Material Type
104
Bulk "B"

FIG. 10

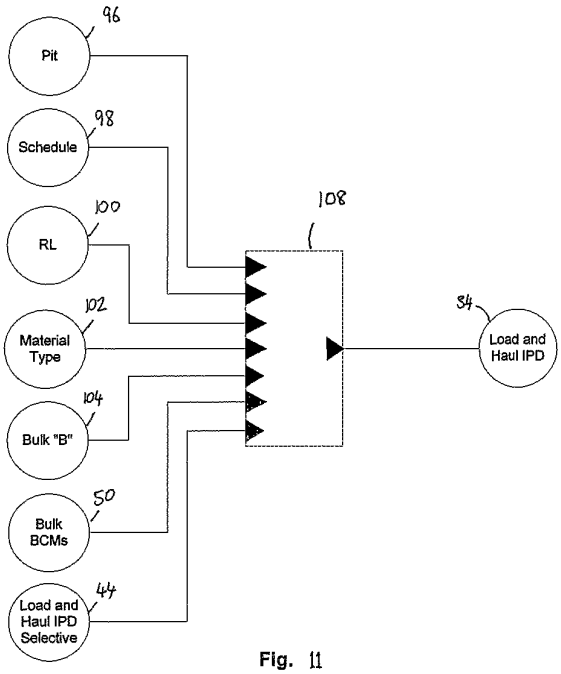

96
Pit
98
Schedule
100
RL
102
Material Type
104
Bulk "B"
50
Bulk BCMs
44
Load and Haul IPD Selective
108
84
Load and Haul IPD

Fig. 11

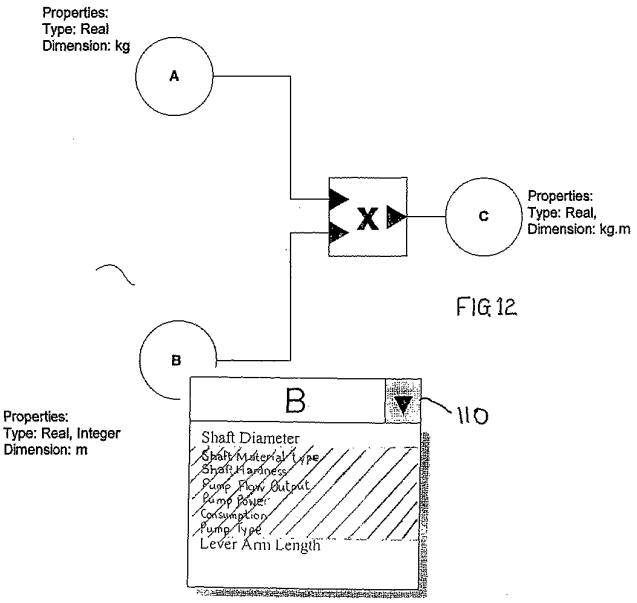

Properties:
Type: Real
Dimension: kg
A
X
C
Properties:
Type: Real,
Dimension: kg.m
B
Properties:
Type: Real, Integer
Dimension: m
B
110
Shaft Diameter
Shaft Material Type
Shaft Hardness
Pump Flow Output
Pump Power Consumption
Pump Type
Lever Arm Length

FIG 12

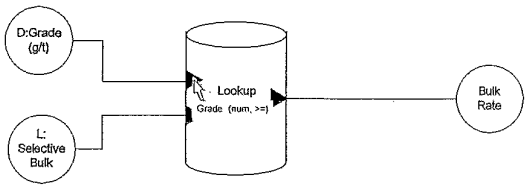


FIG. 13A

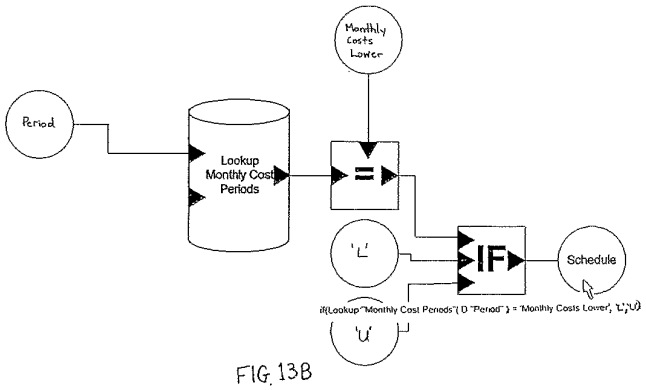


FIG. 13B

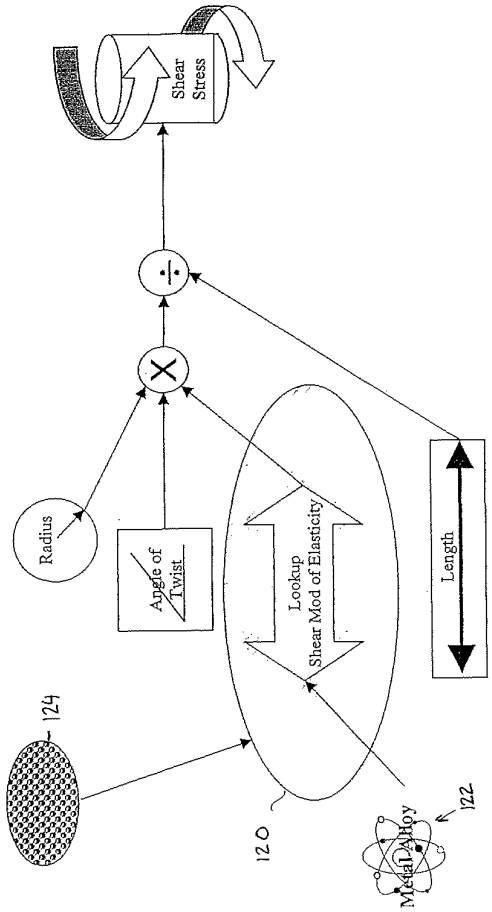

Shear Stress
Radius
Angle of Twist
Lookup Shear Mod of Elasticity
Length
124
120
122
Metal Alloy

FIG. 14

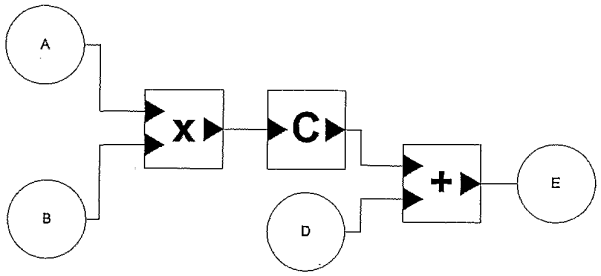


Fig. 15

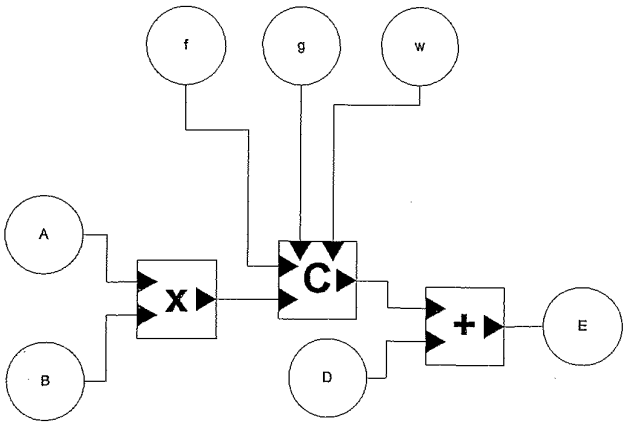


Fig. 16

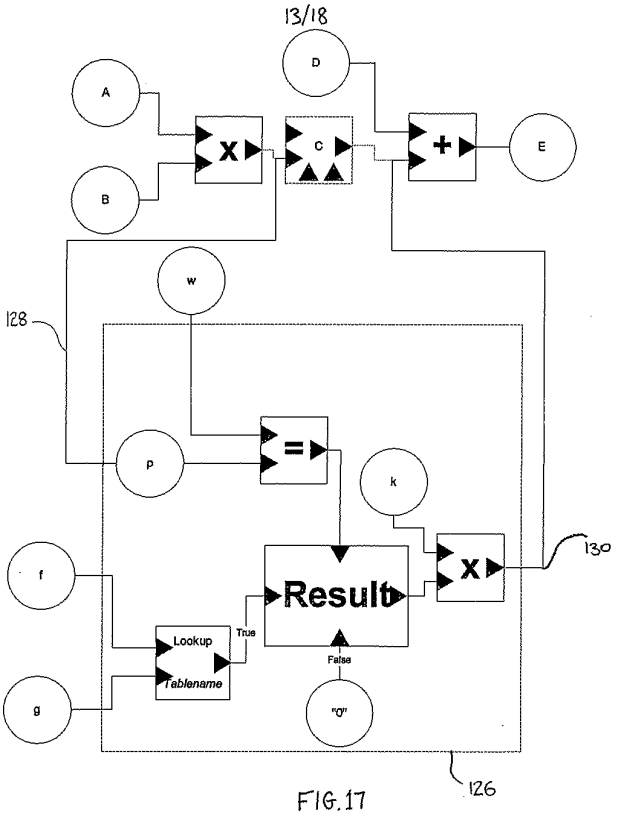

13/18
D
A
X
C
+
E
B
w
128
=
p
k
130
f
Result
X
Lookup
True
Tablename
False
g
"0"
126

FIG. 17

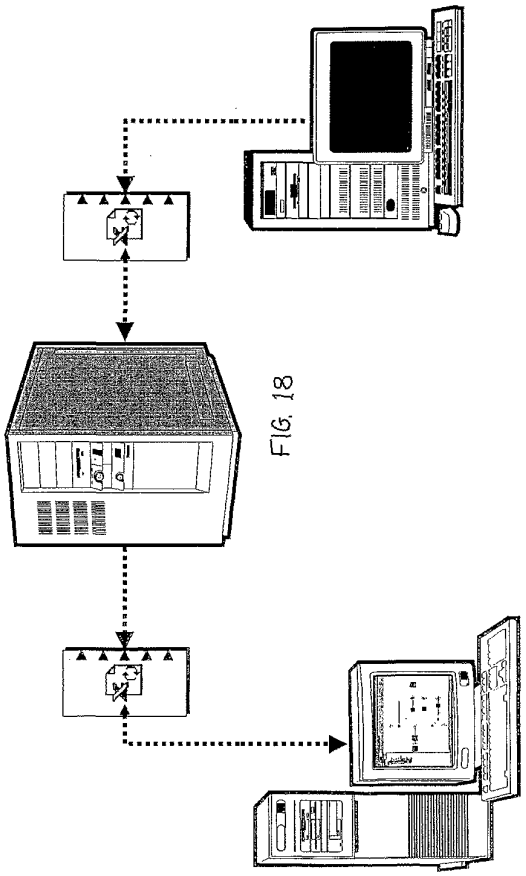


FIG. 18

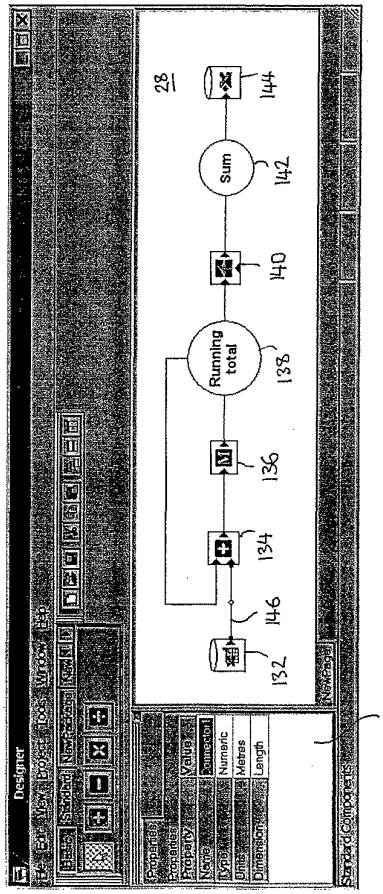

Designer
Standard Components
Property
Value
Suggested
Numeric
Metres
Length
Sum
Running total
28
30
132
134
136
138
140
142
144
146

FIG. 19

24

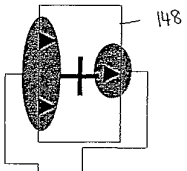


FIG. 20

Methods For "+" Component			
Inputs	Output	Address for Method	
All Numeric	Numeric	Address 1	Arithmetic add
All Text	Text	Address 2	Concatenation of Strings
Numeric/Text	Numeric	Address 3	Converts String to Numeric (if possible) and then arithmetic add
Numeric/Text	Text	Address 4	Converts Numeric to String (if possible) and then concatenates the strings

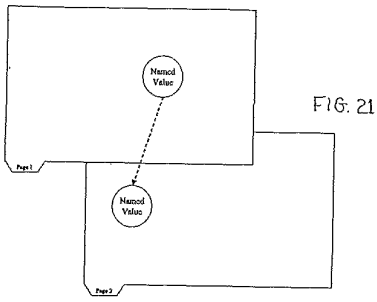


FIG. 21

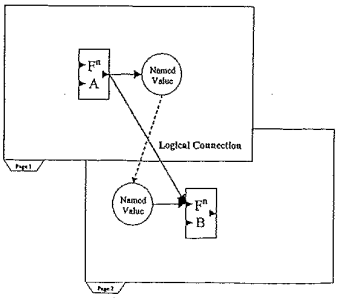
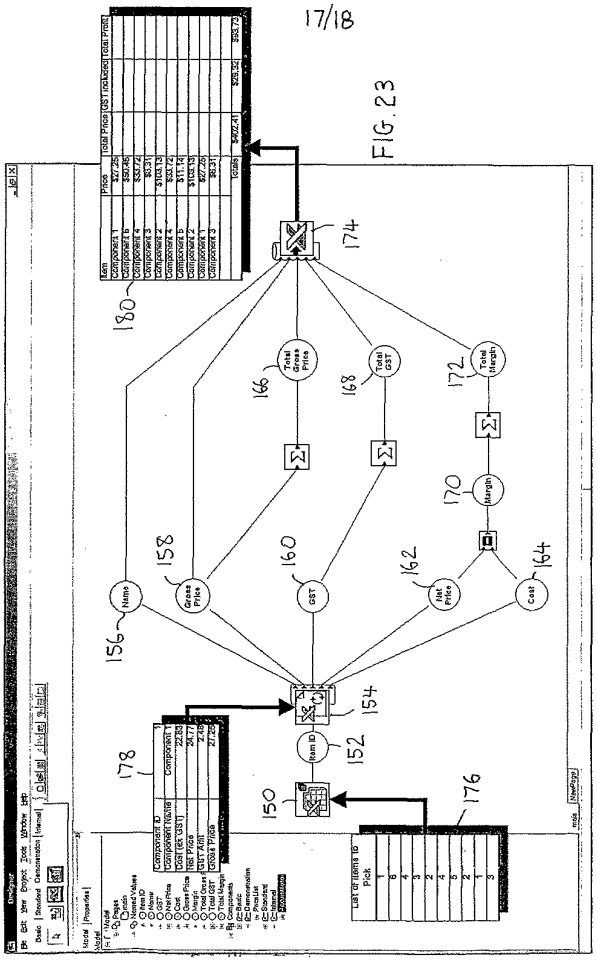


FIG. 22

FIG. 23



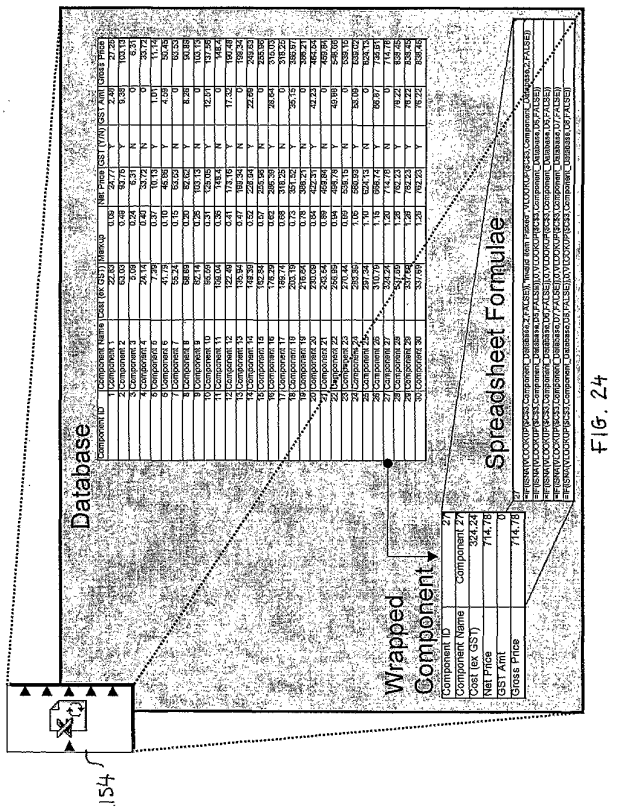


FIG. 24

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/AU01/01053

A. CLASSIFICATION OF SUBJECT MATTER

Int. Cl. [7]: G06F 9/455, 17/00, G05B 17/00

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
WPAT, USPTO.
Example keywords: graphic, formula.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 6051029 (Paterson et al.) 18 April 2000.	1-10, 39, 51, 56-58
X	WO 99/27444 (Entelos, Inc.) 3 June 1999.	1-10, 39, 51, 56-58
X	WO 99/27443 (Entelos, Inc.) 3 June 1999.	1-10, 39, 51, 56-58
X	WO 99/18527 (Invention Machine Corporation) 15 April 1999.	1-10, 39, 51, 56-58
A	US 5021976 (Wexelblat et al.) 4 June 1991.	1-60

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

* Special categories of cited documents:

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

Date of the actual completion of the international search	Date of mailing of the international search report
16 October 2001	18 OCT 2001
Name and mailing address of the ISA/AU AUSTRALIAN PATENT OFFICE PO BOX 200, WODEN ACT 2606, AUSTRALIA E-mail address: pct@ipaustralia.gov.au Facsimile No. (02) 6285 3929	Authorized officer SEAN APPLEGATE Telephone No : (02) 6283 2207

Form PCT/ISA/210 (second sheet) (July 1998)

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/AU01/01053

This Annex lists the known "A" publication level patent family members relating to the patent documents cited in the above-mentioned international search report. The Australian Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

Patent Document Cited in Search Report		Patent Family Member					
US	6051029	AU	11990/99	EP	1027647	WO	9923554
WO	9927444	AU	11208/99	EP	1032874	US	6078739
WO	9927443	AU	13640/99				
WO	9918527	AU	97902/98	EP	1049999	NO	20001790
		US	5901068				
US	5021976	NONE					

END OF ANNEX

フロントページの続き

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

（特許庁注：以下のものは登録商標）
ＵＮＩＸ

(74)代理人 100111419
弁理士 大倉 宏一郎
(72)発明者 バーグ クリストファー イアン
オーストラリア国 ウェスタンオーストラリア ６１０９, スカーボラー, ベントナー ストリート ３５
(72)発明者 ジョンストン グレゴリー オーウェン
オーストラリア国 ウェスタンオーストラリア ６１５８, イースト フレマントル, フラサー ストリート ５７Ｂ
(72)発明者 ジョーンズ ラッセル ベネディクト
オーストラリア国 ウェスタンオーストラリア ６００８, サヴァンコ, レッドファーン ストリート ７８
Ｆターム(参考) 5B076 DC09 DE06 DF06