(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6557603号
(P6557603)
(45)発行日　令和1年8月7日(2019.8.7)　(24)登録日　令和1年7月19日(2019.7.19)

(51)Int.Cl.　F I
G06F 8/34 (2018.01)　G06F 8/34
G06F 3/0481 (2013.01)　G06F 3/0481

請求項の数 31　(全 23 頁)

(21)出願番号　特願2015-534764(P2015-534764)
(86)(22)出願日　平成25年9月27日(2013.9.27)
(65)公表番号　特表2015-535370(P2015-535370A)
(43)公表日　平成27年12月10日(2015.12.10)
(86)国際出願番号　PCT/US2013/062369
(87)国際公開番号　WO2014/052873
(87)国際公開日　平成26年4月3日(2014.4.3)
審査請求日　平成28年9月20日(2016.9.20)
(31)優先権主張番号　61/707,343
(32)優先日　平成24年9月28日(2012.9.28)
(33)優先権主張国・地域又は機関
米国(US)

(73)特許権者　509123208
アビニシオ　テクノロジー　エルエルシー
アメリカ合衆国　０２４２１　マサチューセッツ州　レキシントン　スプリング　ストリート　２０１
(74)代理人　100079108
弁理士　稲葉　良幸
(74)代理人　100109346
弁理士　大貫　敏史
(74)代理人　100117189
弁理士　江口　昭彦
(74)代理人　100134120
弁理士　内藤　和彦
(72)発明者　猪飼　太郎
東京都杉並区荻窪３－４０－６
最終頁に続く

(54)【発明の名称】プログラミング属性のグラフィカル表現

(57)【特許請求の範囲】
【請求項１】
情報を表すための方法であって、
アプリケーションを開発するための少なくとも１つのプログラミング属性の階層を表す第１のデータ構造を定義することであって、前記第１のデータ構造は、前記第１のデータ構造を、他のファイルに記憶された他のデータ構造が使用できるようにするために第１のファイル内に記憶され、前記第１のデータ構造は、第２のファイル内に記憶されている第２のデータ構造の定義に基づいて定義される、定義すること、及び、
前記第１のデータ構造及び前記第２のデータ構造のそれぞれのグラフィカル表現を含む、視覚図を生成することであって、
前記第１のファイルは、前記第１のデータ構造を記憶し、前記第２のファイルは、前記第２のデータ構造を記憶し、それぞれのデータ構造のグラフィカル表現は、前記データ構造の識別子の表現と前記データ構造の一つ以上のフィールドのそれぞれの表現とを含み、それぞれのファイルの前記グラフィカル表現は、前記ファイルの識別子の表現を含み、
前記視覚図は、
前記第１のデータ構造の前記グラフィカル表現を前記第２のデータ構造の前記グラフィカル表現に接続する第１のグラフィカル特徴であって、前記第１のデータ構造についての前記識別子と前記第２のデータ構造についての前記識別子との間の関係を示す第１のグラフィカル特徴、及び、
前記第１のファイルの前記グラフィカル表現を前記第２のファイルの前記グラフィカル

表現に接続する第２のグラフィカル特徴であって、前記第１のデータ構造が前記第１のファイルに記憶され、前記第２のデータ構造が前記第２のファイルに記憶され、前記第１のデータ構造についての前記識別子と前記第２のデータ構造についての前記識別子との間の関係を示す第２のグラフィカル特徴も含む、
生成すること、
を含み、
前記視覚図において、前記第１のデータ構造の前記グラフィカル表現は、前記第１のファイルの前記グラフィカル表現の外側領域内に含まれる前記視覚図の領域内に位置しており、前記第２のデータ構造の前記グラフィカル表現は、前記第２のファイルの前記グラフィカル表現の外側領域内に含まれる前記視覚図の領域内に位置しており、前記視覚図は、ユーザインタフェースに表示される、方法。

【請求項２】
前記第１及び第２のファイルの前記グラフィカル表現は、１つ又は複数のデータ構造グループを定義するため、及び、前記１つ又は複数のデータ構造グループを記憶するための１つ又は複数の新規ファイルを作成するために、前記第１及び第２のデータ構造の前記グラフィカル表現を操作できるように、除去可能である、請求項１に記載の方法。

【請求項３】
前記少なくとも１つのプログラミング属性を変更するために、前記定義された第１のデータ構造の操作を可能にすること、
を更に含む、請求項１に記載の方法。

【請求項４】
前記少なくとも１つのプログラミング属性をファイルに挿入するために、前記定義された第１のデータ構造の操作を可能にすること、
を更に含む、請求項１に記載の方法。

【請求項５】
前記少なくとも１つのプログラミング属性と共に、ステートメントが前記ファイルに挿入される、請求項４に記載の方法。

【請求項６】
前記データ構造を操作することはドラッグ・アンド・ドロップ動作を含む、請求項４に記載の方法。

【請求項７】
前記定義されたデータ構造を操作することは、前記プログラミング属性の前記階層のコンテンツを追加すること、削除すること、及び編集することのうちの少なくとも１つを含む、請求項３に記載の方法。

【請求項８】
前記プログラミング属性は名前付きデータ型である、請求項１に記載の方法。

【請求項９】
前記プログラミング属性は関数である、請求項１に記載の方法。

【請求項１０】
前記データ構造と前記他のデータ構造との間の前記関係は、前記プログラミング属性の系統を表す、請求項１に記載の方法。

【請求項１１】
情報を表現するためのコンピュータ・プログラムを記憶するコンピュータ読み取り可能記憶媒体であって、前記コンピュータ・プログラムは、
アプリケーションを開発するための少なくとも１つのプログラミング属性の階層を表す第１のデータ構造を定義することであって、前記第１のデータ構造は、前記第１のデータ構造を、他のファイルに記憶された他のデータ構造が使用できるようにするために第１のファイル内に記憶され、前記第１のデータ構造は、第２のファイル内に記憶されている第２のデータ構造の定義に基づいて定義される、定義すること、及び、
前記第１のデータ構造及び前記第２のデータ構造のそれぞれのグラフィカル表現を含む

、視覚図を生成することであって、
前記第１のファイルは、前記第１のデータ構造を記憶し、前記第２のファイルは、前記第２のデータ構造を記憶し、それぞれのデータ構造のグラフィカル表現は、前記データ構造の識別子の表現と前記データ構造の一つ以上のフィールドのそれぞれの表現とを含み、それぞれのファイルの前記グラフィカル表現は、前記ファイルの識別子の表現を含み、
前記視覚図は、
前記第１のデータ構造の前記グラフィカル表現を前記第２のデータ構造の前記グラフィカル表現に接続する第１のグラフィカル特徴であって、前記第１のデータ構造についての前記識別子と前記第２のデータ構造についての前記識別子との間の関係を示す第１のグラフィカル特徴、及び、
前記第１のファイルの前記グラフィカル表現を前記第２のファイルの前記グラフィカル表現に接続する第２のグラフィカル特徴であって、前記第１のデータ構造が前記第１のファイルに記憶され、前記第２のデータ構造が前記第２のファイルに記憶され、前記第１のデータ構造についての前記識別子と前記第２のデータ構造についての前記識別子との間の関係を示す第２のグラフィカル特徴も含む、
生成すること、
をコンピューティング・システムに実行させるための命令を含み、
前記視覚図において、前記第１のデータ構造の前記グラフィカル表現は、前記第１のファイルの前記グラフィカル表現の外側領域内に含まれる前記視覚図の領域内に位置しており、前記第２のデータ構造の前記グラフィカル表現は、前記第２のファイルの前記グラフィカル表現の外側領域内に含まれる前記視覚図の領域内に位置しており、前記視覚図は、ユーザインタフェースに表示される、コンピュータ読み取り可能記憶媒体。

【請求項１２】
情報を表すためのコンピューティング・システムであって、
情報を処理するように構成された少なくとも１つのプロセッサであって、
前記処理は、
アプリケーションを開発するための少なくとも１つのプログラミング属性の階層を表す第１のデータ構造を定義することであって、前記第１のデータ構造は、前記第１のデータ構造を、他のファイルに記憶された他のデータ構造が使用できるようにするために第１のファイル内に記憶され、前記第１のデータ構造は、第２のファイル内に記憶されている第２のデータ構造の定義に基づいて定義される、定義すること、及び、
前記第１のデータ構造及び前記第２のデータ構造のそれぞれのグラフィカル表現を含む、視覚図を生成することであって、
前記第１のファイルは、前記第１のデータ構造を記憶し、前記第２のファイルは、前記第２のデータ構造を記憶し、それぞれのデータ構造のグラフィカル表現は、前記データ構造の識別子の表現と前記データ構造の一つ以上のフィールドのそれぞれの表現とを含み、それぞれのファイルの前記グラフィカル表現は、前記ファイルの識別子の表現を含み、
前記視覚図は、
前記第１のデータ構造の前記グラフィカル表現を前記第２のデータ構造の前記グラフィカル表現に接続する第１のグラフィカル特徴であって、前記第１のデータ構造についての前記識別子と前記第２のデータ構造についての前記識別子との間の関係を示す第１のグラフィカル特徴、及び、
前記第１のファイルの前記グラフィカル表現を前記第２のファイルの前記グラフィカル表現に接続する第２のグラフィカル特徴であって、前記第１のデータ構造が前記第１のファイルに記憶され、前記第２のデータ構造が前記第２のファイルに記憶され、前記第１のデータ構造についての前記識別子と前記第２のデータ構造についての前記識別子との間の関係を示す第２のグラフィカル特徴も含む、
生成すること、
を含む、少なくとも１つのプロセッサと、
前記視覚図を提示するための出力デバイスと、

を含み、
前記視覚図において、前記第１のデータ構造の前記グラフィカル表現は、前記第１のファイルの前記グラフィカル表現の外側領域内に含まれる前記視覚図の領域内に位置しており、前記第２のデータ構造の前記グラフィカル表現は、前記第２のファイルの前記グラフィカル表現の外側領域内に含まれる前記視覚図の領域内に位置している、コンピューティング・システム。
【請求項１３】
情報を表現するためのコンピューティング・システムであって、
アプリケーションを開発するための少なくとも１つのプログラミング属性の階層を表す第１のデータ構造を定義することであって、前記第１のデータ構造は、前記第１のデータ構造を、他のファイルに記憶された他のデータ構造が使用できるようにするために第１のファイル内に記憶され、前記第１のデータ構造は、第２のファイル内に記憶されている第２のデータ構造の定義に基づいて定義される、定義すること、及び、
前記第１のデータ構造及び前記第２のデータ構造のそれぞれのグラフィカル表現を含む、視覚図を生成することであって、
前記第１のファイルは、前記第１のデータ構造を記憶し、前記第２のファイルは、前記第２のデータ構造を記憶し、それぞれのデータ構造のグラフィカル表現は、前記データ構造の識別子の表現と前記データ構造の一つ以上のフィールドのそれぞれの表現とを含み、それぞれのファイルの前記グラフィカル表現は、前記ファイルの識別子の表現を含み、
前記視覚図は、
前記第１のデータ構造の前記グラフィカル表現を前記第２のデータ構造の前記グラフィカル表現に接続する第１のグラフィカル特徴であって、前記第１のデータ構造についての前記識別子と前記第２のデータ構造についての前記識別子との間の関係を示す第１のグラフィカル特徴、及び、
前記第１のファイルの前記グラフィカル表現を前記第２のファイルの前記グラフィカル表現に接続する第２のグラフィカル特徴であって、前記第１のデータ構造が前記第１のファイルに記憶され、前記第２のデータ構造が前記第２のファイルに記憶され、前記第１のデータ構造についての前記識別子と前記第２のデータ構造についての前記識別子との間の関係を示す第２のグラフィカル特徴も含む、
生成すること、
を含む、処理のための手段、及び、
前記視覚図を提示するための手段、
を含み、
前記視覚図において、前記第１のデータ構造の前記グラフィカル表現は、前記第１のファイルの前記グラフィカル表現の外側領域内に含まれる前記視覚図の領域内に位置しており、前記第２のデータ構造の前記グラフィカル表現は、前記第２のファイルの前記グラフィカル表現の外側領域内に含まれる前記視覚図の領域内に位置している、コンピューティング・システム。
【請求項１４】
前記第１及び第２のファイルの前記グラフィカル表現は、１つ又は複数のデータ構造グループを定義するため、及び、前記１つ又は複数のデータ構造グループを記憶するための１つ又は複数の新規ファイルを作成するために、前記第１及び第２のデータ構造の前記グラフィカル表現を操作できるように、除去可能である、請求項１１に記載のコンピュータ読み取り可能記憶媒体。
【請求項１５】
前記少なくとも１つのプログラミング属性を変更するために、前記定義された第１のデータ構造の操作を可能にすること、
を更に含む、請求項１１に記載のコンピュータ読み取り可能記憶媒体。
【請求項１６】
前記少なくとも１つのプログラミング属性をファイルに挿入するために、前記定義され

た第１のデータ構造の操作を可能にすること、
を更に含む、請求項１１に記載のコンピュータ読み取り可能記憶媒体。
【請求項１７】
前記少なくとも１つのプログラミング属性と共に、ステートメントが前記ファイルに挿入される、請求項１６に記載のコンピュータ読み取り可能記憶媒体。
【請求項１８】
前記データ構造を操作することはドラッグ・アンド・ドロップ動作を含む、請求項１６に記載のコンピュータ読み取り可能記憶媒体。
【請求項１９】
前記定義されたデータ構造を操作することは、前記プログラミング属性の前記階層のコンテンツを追加すること、削除すること、及び編集することのうちの少なくとも１つを含む、請求項１５に記載のコンピュータ読み取り可能記憶媒体。
【請求項２０】
前記プログラミング属性は名前付きデータ型である、請求項１１に記載のコンピュータ読み取り可能記憶媒体。
【請求項２１】
前記プログラミング属性は関数である、請求項１１に記載のコンピュータ読み取り可能記憶媒体。
【請求項２２】
前記データ構造と前記他のデータ構造との間の前記関係は、前記プログラミング属性の系統を表す、請求項１１に記載のコンピュータ読み取り可能記憶媒体。
【請求項２３】
前記第１及び第２のファイルの前記グラフィカル表現は、１つ又は複数のデータ構造グループを定義するため、及び、前記１つ又は複数のデータ構造グループを記憶するための１つ又は複数の新規ファイルを作成するために、前記第１及び第２のデータ構造の前記グラフィカル表現を操作できるように、除去可能である、請求項１２に記載のコンピューティング・システム。
【請求項２４】
前記少なくとも１つのプログラミング属性を変更するために、前記定義された第１のデータ構造の操作を可能にすること、
を更に含む、請求項１２に記載のコンピューティング・システム。
【請求項２５】
前記少なくとも１つのプログラミング属性をファイルに挿入するために、前記定義された第１のデータ構造の操作を可能にすること、
を更に含む、請求項１２に記載のコンピューティング・システム。
【請求項２６】
前記少なくとも１つのプログラミング属性と共に、ステートメントが前記ファイルに挿入される、請求項２５に記載のコンピューティング・システム。
【請求項２７】
前記データ構造を操作することはドラッグ・アンド・ドロップ動作を含む、請求項２５に記載のコンピューティング・システム。
【請求項２８】
前記定義されたデータ構造を操作することは、前記プログラミング属性の前記階層のコンテンツを追加すること、削除すること、及び編集することのうちの少なくとも１つを含む、請求項２４に記載のコンピューティング・システム。
【請求項２９】
前記プログラミング属性は名前付きデータ型である、請求項１２に記載のコンピューティング・システム。
【請求項３０】
前記プログラミング属性は関数である、請求項１２に記載のコンピューティング・シス

テム。
【請求項３１】
前記データ構造と前記他のデータ構造との間の前記関係は、前記プログラミング属性の系統を表す、請求項１２に記載のコンピューティング・システム。
【発明の詳細な説明】
【技術分野】
【０００１】
優先権の主張
本願は、米国特許法第１１９条（ｅ）の定めにより、その全文が参照により本明細書に組み込まれた２０１２年９月２８日付け出願の米国特許出願第６１／７０７，３４３号に対する優先権を主張するものである。
【０００２】
本明細書は、プログラミング属性を表すためのグラフ・ベース手法に関する。
【背景技術】
【０００３】
複雑な計算は、しばしば、有向グラフ（「データ・フロー・グラフ」と呼ばれる）を介するデータ・フローとして表すことが可能であり、計算の構成要素はグラフの頂点に関連付けられ、構成要素間のデータ・フローはグラフのリンク（弧、辺）に対応する。構成要素は、１つ又は複数の入力ポートでデータを受信し、データを処理し、１つ又は複数の出力ポートからデータを提供する、データ処理構成要素と、データ・フローのソース又はシンクとして働くデータセット構成要素とを、含むことができる。こうしたグラフ・ベースの計算を実装するシステムは、米国特許第５９６６０７２号「EXECUTING COMPUTATIONS EXPRESSED AS GRAPHS」に記載されている。様々なタイプのデータが、グラフの構成要素によって受信、処理、及び出力され得る。同様の処理機能によって、異なるアプリケーションに対して同等タイプのデータを使用及び再使用することができる。
【発明の概要】
【０００４】
一態様において、情報を表すための方法は、アプリケーションを開発するための少なくとも１つのプログラミング属性の階層を表すデータ構造を定義することを含む。データ構造は、このデータ構造を、他のファイルに記憶された他のデータ構造が使用できるようにするためにファイル内に記憶される。方法は、データ構造のグラフィカル表現及びデータ構造を記憶するファイルのグラフィカル表現を含む、視覚図を生成することも含む。視覚図は、データ構造と別のデータ構造との間の関係のグラフィカル表現、及び、データ構造を記憶するファイルと他のデータ構造を記憶する別のファイルとの間の関係のグラフィカル表現も含む。
【０００５】
別の態様において、コンピュータ読み取り可能記憶媒体は、情報を表現するためのコンピュータ・プログラムを記憶する。コンピュータ・プログラムは、アプリケーションを開発するための少なくとも１つのプログラミング属性の階層を表すデータ構造を、コンピューティング・システムに定義させるための命令を含む。データ構造は、このデータ構造を他のファイルに記憶された他のデータ構造が使用できるようにするために、ファイル内に記憶される。更に命令は、データ構造のグラフィカル表現及びデータ構造を記憶するファイルのグラフィカル表現を含む視覚図を、コンピューティング・システムに生成させる。視覚図は、データ構造と別のデータ構造との間の関係のグラフィカル表現、及び、データ構造を記憶するファイルと他のデータ構造を記憶する別のファイルとの間の関係のグラフィカル表現も含む。
【０００６】
別の態様において、情報を表すためのコンピューティング・システムは、情報を処理するように構成された少なくとも１つのプロセッサを含む。処理は、アプリケーションを開発するための少なくとも１つのプログラミング属性の階層を表すデータ構造を定義するこ

とを含む。データ構造は、このデータ構造を他のファイルに記憶された他のデータ構造が使用できるようにするために、ファイル内に記憶される。処理は、データ構造のグラフィカル表現及びデータ構造を記憶するファイルのグラフィカル表現を含む、視覚図を生成することも含む。視覚図は、データ構造と別のデータ構造との間の関係のグラフィカル表現、及び、データ構造を記憶するファイルと他のデータ構造を記憶する別のファイルとの間の関係のグラフィカル表現も含む。コンピュータ・システムは、データ構造及びファイルのグラフィカル表現、並びにデータ構造の関係及びファイルの関係のグラフィカル表現を含む、視覚図を提示するための出力デバイスも含む。

【０００７】

別の態様において、情報を表現するためのコンピューティング・システムは、アプリケーションを開発するための少なくとも１つのプログラミング属性の階層を表すデータ構造を定義することを含む、処理のための手段を含む。データ構造は、このデータ構造を他のファイルに記憶された他のデータ構造が使用できるようにするために、ファイル内に記憶される。処理は、データ構造のグラフィカル表現及びデータ構造を記憶するファイルのグラフィカル表現を含む、視覚図を生成することも含む。視覚図は、データ構造と別のデータ構造との間の関係のグラフィカル表現、及び、データ構造を記憶するファイルと他のデータ構造を記憶する別のファイルとの間の関係のグラフィカル表現も含む。コンピュータ・システムは、データ構造及びファイルのグラフィカル表現、並びにデータ構造の関係及びファイルの関係のグラフィカル表現を含む、視覚図を提示するための手段も含む。

【０００８】

実装は、以下の特徴のうちのいずれか又はすべてを含むことができる。ファイルのグラフィカル表現は、１つ又は複数のデータ構造グループを定義するため、及び、１つ又は複数のデータ構造グループを記憶するための１つ又は複数の新規ファイルを作成するために、データ構造のグラフィカル表現を操作できるように、除去可能な場合がある。定義されたデータ構造は、少なくとも１つのプログラミング属性を調整するように操作可能である。ステートメントを、少なくとも１つのプログラミング属性と共にファイルに挿入することができる。データ構造を操作することは、ドラッグ・アンド・ドロップ動作を含むことができる。定義されたデータ構造を操作することは、プログラミング属性の階層のコンテンツの追加、削除、及び編集のうちの、少なくとも１つを含むことができる。プログラミング属性は、名前付きデータ型、機能などとすることができる。データ構造と他のデータ構造との間の関係は、プログラミング属性の系統を表すことができる。

【０００９】

態様は、以下の利点のうちの１つ又は複数を含むことができる。

【００１０】

プログラミング属性（例えばフィールド、名前付きデータ型構造、関数など）をグラフィカルに表すことで、開発者は、属性の詳細（例えば使用されているデータ型）、及び属性（例えば、以前に定義されたフィールドを使用する名前付きデータ型）と属性を記憶するために使用されるファイルとの間の関係を、比較的迅速に確認することができる。グラフィカル形式で提示された属性、属性を記憶するファイルなどを効率的に操作して、例えば名前付きデータ型構造を編集し、それにより必要に応じて変更を伝搬することができる。

【００１１】

本発明の他の特徴及び利点は、以下の説明から、及び特許請求の範囲から明らかとなろう。

【図面の簡単な説明】

【００１２】

【図１】グラフ・ベースの計算を実行するためのシステムを示すブロック図である。

【図２】データ型情報を提示するユーザ・インターフェースである。

【図３】データ型情報を提示するユーザ・インターフェースである。

【図４】データ型情報を提示するユーザ・インターフェースである。

【図５】データ型情報を提示するユーザ・インターフェースである。
【図６】データ型情報を提示するユーザ・インターフェースである。
【図７】データ型情報のグラフィカル表現を提示するユーザ・インターフェースである。
【図８】データ型情報のグラフィカル表現を提示するユーザ・インターフェースである。
【図９】データ型情報のグラフィカル表現を提示するユーザ・インターフェースである。
【図１０】データ型情報のグラフィカル表現を提示するユーザ・インターフェースである。
【図１１】データ型情報のグラフィカル表現を提示するユーザ・インターフェースである。
【図１２】データ型情報のグラフィカル表現を提示するユーザ・インターフェースである。
【図１３】例示的なプログラミング属性提示手順のフローチャートである。
【発明を実施するための形態】
【００１３】
図１は、例えば、偶然閲覧した人がコンテンツ、階層、及び属性の系統を効率的に決定できるようにするために、データ型、実行可能関数などのプログラミング属性をグラフィカルに提示することが可能な、例示のデータ処理システム１００を示す。一般に、こうした機能を提供するために、システム１００は、それぞれが様々な記憶フォーマット（例えば、データベース・テーブル、スプレッドシート・ファイル、フラット・テキスト・ファイル、又はメインフレームによって使用されるネイティブ・フォーマット）のいずれかでデータを記憶することが可能な、記憶デバイス又はオンライン・データ・ストリームへの接続などの、１つ又は複数のデータ・ソースを含み得る、データ・ソース１０２を含む。この例では、実行環境１０４が、前処理モジュール１０６及び実行モジュール１１２を含む。実行環境１０４は、ＵＮＩＸオペレーティング・システムなどの好適なオペレーティング・システムの制御の下で、１つ又は複数の汎用コンピュータ上でホストすることができる。例えば実行環境１０４は、ローカル（例えば、ＳＭＰコンピュータなどのマルチプロセッサ・システム）又はローカル分散型（例えば、クラスタ又はＭＰＰとして結合された複数のプロセッサ）、或いは、リモート又はリモート分散型（例えば、ローカル・エリア・ネットワーク（ＬＡＮ）及び／又はワイド・エリア・ネットワーク（ＷＡＮ）を介して結合された複数のプロセッサ）、或いはそれらの任意の組み合わせである、複数の中央処理ユニット（ＣＰＵ）を使用するコンピュータ・システムの構成を含む、多重ノード並列コンピューティング環境を含むことができる。
【００１４】
前処理モジュール１０６は、データ・ソース１０２からデータを読み取り、他のモジュールによる更なる進行を予想して、対応する処理動作を実行する。データ・ソース１０２を提供する記憶デバイスは、実行環境１０４に対してローカルであり、例えば実行環境１０４を実行するコンピュータに接続された記憶媒体（例えばハード・ドライブ１０８）上に記憶されることが可能であるか、又は、実行環境１０４に対してリモートであり、例えば、実行環境１０４を実行するコンピュータとリモート接続を介して通信するリモート・システム（例えばメインフレーム１１０）上でホストされることが可能である。
【００１５】
実行モジュール１１２は、例えば、実行環境１０４にアクセス可能なデータ記憶システム１１６内に記憶されたデータ１１４（例えば企業データ、社内記録など）を処理するために、前処理モジュール１０６によって生成された処理済みデータを使用する。データ記憶システム１１６は、開発者１２０がデータ記憶システム１１６内に記憶された情報の検討、編集などを実行できる、開発環境１１８にもアクセス可能である。いくつかの配置構成において、開発環境１１８は、望ましい動作を実行するために実行環境１０４を作成及び調整する際に使用可能である。例えば開発環境１１８は、頂点間の（作業要素の流れを表す）有向辺によって接続された（構成要素又はデータセットを表す）頂点を含むデータフロー・グラフとしてアプリケーションを開発するための、システムとすることができる

。例えば、こうした環境は、参照により本明細書に組み込まれた、「Managing Parameters for Graph-Based Applications」という名称の米国公開第２００７／００１１６６８号に、より詳細に記載されている。こうしたグラフ・ベースの計算を実行するためのシステムは、参照により本明細書に組み込まれた米国特許第５，５６６，０７２号、EXECUTING COMPUTATIONS EXPRESSED AS GRAPHSに記載されている。このシステムに従って作られたデータフロー・グラフは、グラフ構成要素によって表される個々のプロセス内外の情報を取得するため、プロセス間で情報を移動させるため、プロセスに関する実行順序を定義するための、方法を提供する。このシステムは、プロセス間通信方法（例えば、グラフの辺に従った通信経路は、ＴＣＰ／ＩＰ又はＵＮＩＸドメイン・ソケットを使用すること、又はプロセス間でデータを渡すために共有メモリを使用することが可能である）を選択するアルゴリズムを含む。

【００１６】

前処理モジュール１０６は、異なる形のデータベース・システムを含む様々なタイプのシステムからデータを受信することができる。データは、場合によってはヌル値を含む、それぞれのフィールド（「属性」又は「列」とも呼ばれる）について値を有するレコードとして編成可能である。データ・ソースから最初にデータを読み取る場合、前処理モジュール１０６は、典型的には、そのデータ・ソース内のレコードに関する何らかの初期フォーマット情報から始める。いくつかの環境において、データ・ソースのレコード構造は初期には知られていない場合があり、代わりにデータ・ソースの分析後に決定することができる。レコードに関する初期情報は、別個の値、レコード内のフィールドの順序、及び、ビットによって表されるデータ型（例えば、文字列、符号付／符号なし整数）を表す、ビット数を含むことができる。

【００１７】

レコード内で使用されるのと同様に、異なるデータ型（例えば、文字列、整数、フローイング・ポイント値）を、実行環境１０４、開発環境１１８などによって実行されるデータ（例えばレコード）を処理するために使用可能である。例えば、レコード（例えば従業員レコード、学生レコードなど）を処理するために、様々なタイプのデータ型を定義及び使用することができる。こうしたレコードを処理するために、個々の基本データ型（例えば、文字列、日付、日時、整数、１０進数など）を同時に使用して、データ型構造と呼ばれるデータ型の編成を定義することができる。例えば以下のレコードは、ある人物の属性を表すために４つの基本タイプをグループ化している。

```
record
 string(",")surname;
 string(",")given#name;
 date("YYYY-MM-DD")date#of#birth;
 integer(1)gender; // 0 for male,1 for female
 decimal(9)SSN;
end
```

レコードは、いくつかのフィールド（例えば、個人の氏名はその生年月日、性別、及び社会保障番号と共に表される）を含み、それぞれのフィールドは名前及びデータ型からなる。図２を参照すると、こうしたレコードはエディタによって定義可能である。この例では、ユーザ・インターフェース２００は、入力ファイルから読み取られるデータのフォーマットを定義するためのエディタを提示する。読み取られるデータ（例えば、姓、生年月日、性別）を定義すると共に、個々のデータ型（例えば文字列、日付、整数）及び各フィールドに関する制限が定義される。この例は、レコード・タイプを情報の５つの個別エントリを含むものとして提示しているが、レコード・タイプは拡大又は縮小可能である。例えばより多くのデータに、現行データの追加、削除、組み合わせが可能である。上記の命令に続いて、例えば企業団体及び対応する雇用者のリストを表すために、レコード・タイプを入れ子にすることができる。

```
/* Business entity */
```

```
record
 string(",")business#name;
 decimal(9)tax#payer#id;
 decimal(11)main#TEL#no;
 record
    record
           string(",")last;//last name
           string(",")first;//first name
    end name;//subrecord
    record
           date("YYYY-MM-DD")date#of#birth;
           integer(1)gender;//0 for male,1 for female
    end bio;//subrecord
    decimal(9)SSN;
 end[integer(4)]employees;//vector of employees
end
```

【００１８】

データ型構造の作成及び名前付けを介して、（例えば、学生情報、いくつかの会社の従業員に関する順序情報などを表すために）同等のフィールドなどを複数のアプリケーションに使用することができる。図３を参照すると、入れ子にされたデータ型情報を含むデータ型構造を提示するために、ユーザ・インターフェース３００を実装することができる。この配置構成では、会社に関連付けられたフィールドが上部（括弧３０２で示される）に定義され、従業員情報に関するフィールドが下部（括弧３０４で示される）に定義されている。想像し得るように、データ型構造は多種多様のアプリケーションについて定義可能であり、多くの同様のフィールドを（例えば学生情報に関するレコード、従業員情報に関するレコード、などを定義するために）使用及び再使用することができる。

【００１９】

図４を参照すると、例えばアプリケーション開発の際に、フィールドを適切に定義すること、及び、使用のためにデータ型構造を形成するようにフィールドをグループ化することによって、開発者を支援するために、１つ又は複数の技法を実装することができる。例えば、ユーザ・インターフェース４００と対話すること（例えばポインティング・デバイスを用いてフィールドを選択すること）によって、ドロップダウン・メニュー４０２が表示され、アプリケーションで使用するために（開発者によって）潜在的に選択可能な名前付きデータ型構造のリストを提示することができる。ドロップダウン・メニュー４０２に示されるように、更により多くの名前付きデータ型構造が定義されるため、提示されるリストはユーザにとって煩わしいほど多く、興味ある名前付きデータ型構造を識別するためにナビゲートするのが困難になる可能性がある。例えば、頻繁に選択される名前付きデータ型構造と共に、リストは、特定アプリケーションに固有の多数のほとんど使用されない名前付きデータ型構造を含めるために経時的に多くなっていく可能性がある。図に示されるように、メニュー４０２内の非常に多くのエントリは、開発者の効率に大きな影響を与える可能性があり、ユーザ・インターフェース４００の使用を妨げる場合さえある。更に、（メニュー４０２のエントリに見られるように）名前のみが割り当てられ、他のいずれの情報もないため、多くの名前付きデータ型構造は冗長であり、他の名前付きデータ型構造間の関係（あれば）に関する情報は一切提供していない可能性がある。

【００２０】

データ型構造のより管理可能な表現を提供するために、１つ又は複数の技法及び方法を実装することができる。一例では、複数の異なるアプリケーションで使用されるデータ型構造は共通に定義可能である。上記の例から、各学生に関するデータ型構造は各会社従業員のデータ型構造と等価であり、以下のように共通に定義可能である。

```
record
```

```
  record
    string(",")last;//last name
    string(",")first;//first name
  end name; // subrecord
  record
    date("YYYY-MM-DD")date#of#birth;
    integer(1)gender;//0 for male,1 for female
  end bio;//subrecord
  decimal(9)SSN;
end
```

この共通データ型構造に名前（例えば「person#t」）を付けることによって、名前付きデータ型構造を呼び出し、学生及び従業員アプリケーションのための情報を収集するなどの同様の目的で、アプリケーション内で使用することができる。例えば、名前付きデータ型構造（「person#t」）は以下のように定義することができる。

```
  type person#t=
  record
   record
    string(",")last;//last name
    string(",")first;//first name
   end name;//subrecord
   record
    date("YYYY-MM-DD")date#of#birth;
    integer(1)gender;//0 for male,1 for female
   end bio;//subrecord
   decimal(9)SSN;
end;
```

定義されると、名前付きデータ型構造（「person#t」）を使用して、それぞれ会社従業員及びクラス学生のアプリケーションに対してデータ型構造を定義することができる。

```
/* Business entity */
record
 string(",")business#name;
 decimal(9)tax#payer#id;
 decimal(11)main#TEL;
 person#t[integer(4)] employees; // here
end

/* Classroom*/
record
 string(",")home#room#instructor;
 decimal(2)grade;
 person#t[integer(4)] students; // and here
end;
```

同様に、新規のデータ型構造は複数のアプリケーションで使用するために名前付けすることができる。例えば、名前付きデータ型構造「person#t」を使用する（「business#t」とタイトル付けされた）名前付きデータ型構造を、ビジネス情報を記憶するために定義することができる。

```
type business#t=
record
 string(",")business#name;
 decimal(9)tax#payer#id;
```

```
 decimal(11)main#TEL;
 person#t[integer(4)] employees; / /vector of employees
end;
```

同様に、名前付きデータ型構造「person#t」を使用する（「class#t」と名前付けされた）名前付きデータ型構造を、クラス情報を記憶するために定義することができる。

```
type class#t=
record
 string(",")home#room#instructor;
 decimal(2)grade;
 person#t[integer(4)] students; // vector of students
end;
```

【００２１】

複数の他のデータ型構造を定義するために共通のデータ型構造を使用することから、様々な利点が与えられる。例えば、（含まれるデータ型構造を調整するために）２つのデータ型構造を別個に調整することが必要な代わりに、共通に使用される名前付きデータ型構造を１回再定義することができる。それに相応して、名前付きデータ型構造を調整することにより、両方のデータ型構造（及び名前付きデータ型構造を使用する任意の他のデータ型構造）におけるいずれの変化も反映されることになる。それにより、開発者が複数のデータ構造内で使用されている名前付きデータ型構造の単一インスタンスを調整できるようにすることによって、効率を向上させることができる。しかしながら開発者は、（例えば、開発者が、含まれるデータ型構造を使用するデータ型構造のうちのすべてではない、含まれるデータ型構造の調整にのみ関心がある場合）、こうした伝搬調整に引き続き注意すべきである。

【００２２】

図５を参照すると、定義された場合、名前付きデータ型構造を記憶するため、及び使用する構造を取り出すために、１つ又は複数の技法又は方法を実装することができる。例えば、名前付きデータ型構造は、データ操作言語（ＤＭＬ）などの言語で表現すること、及び、テキスト・ファイル（ＤＭＬファイルと呼ばれる）に記憶することが可能である。上記で定義された（例えば「person#t」）名前付きデータ型構造を使用して、「project.dml」とタイトル付けされたＤＭＬファイルを記憶デバイス内に記憶し、定義されたデータ型構造を使用するために取り出すことができる。ＤＭＬファイル内に記憶された名前付きデータ型構造にアクセスして使用するために、様々な技法を実装することができる。例えば、ファイル（例えば別のＤＭＬファイル）は、１つ又は複数の他のＤＭＬファイル（及びファイル内に定義された名前付きデータ型構造）にアクセスして使用できるようにする、１つ又は複数の命令（例えば「include（含める）」ステートメント、「package（パッケージする）」ステートメントなど）を含むことができる。例えば、ＤＭＬファイルのコンテンツは、以下の言語を含むことができる。

```
include"project.dml";

type sales#prospects#t=
record
 class#t[integer(4)]educational;
 business#t[integer(4)]commercial;
end;
```

「include」ステートメントの使用を介してＤＭＬファイル（「project.dml」）にアクセスすることによって、ＤＭＬファイル内に定義された名前付きデータ型構造（例えば「business#t」）を取り出し、ファイル内に定義されたデータ型構造（例えば「sales#prospects#t」）を定義するために使用することができる。こうした機能は、矛盾するデータ型構造（例えば、同じ名前であるが異なる型定義を備えるデータ型構造）の可能性と共に、データ型定義の冗長性を低減させる。

【００２３】
図に示されるように、定義及び記憶されると、名前付きデータ型構造は開発者による選択及び使用のために提示することができる。例えば、ユーザ・インターフェース５００は、アプリケーション（例えば、グローバル変数（Globals）５０２、関数（Functions）５０４、ユーザ定義タイプ（User-Defined Types）５０６など）を構築するためのプログラミング属性含むことができる。ユーザ・インターフェース５００に提示されるように、３つの名前付きデータ型構造（例えば「person#t」、「business#t」、及び「class#t」）がユーザ定義タイプ５０６に含まれ、開発者が使用のために選択することができる。しかしながら、図３に提示されたドロップダウン・メニュー４０２と同様に、様々なアプリケーションに関するデータ型構造のリストが増えるにつれて、冗長なエントリが急増すると共にユーザ定義タイプ５０６に含まれるエントリが扱いにくくなる可能性がある。頻繁に使用される名前付きデータ型構造及びそれほど使用されない名前付きデータ型構造の増加により、いずれの特定データ型構造が使用可能であるかを識別することが困難になる可能性がある。開発者にとって、以前に定義されたデータ型構造を識別することが困難になり、過度に時間のかかるものになるにつれて、冗長性も問題になる可能性がある。ユーザ・インターフェース５００の提示によって、各名前付きデータ型構造のリストが提供されるが、追加の情報はほとんど提供されない。例えば、データ型構造間の依存性の系統、及び関連するデータ型を如何にしてグループ化できるかなどは、一般に、ユーザ・インターフェース５００にはない。更に、データ型構造の名前は提示されるが、データ型構造のコンテンツに関する情報は、各データ型構造のタイトルからひらめく以外にはインターフェースから一切提供されない。

【００２４】
図６を参照すると、名前付きデータ型構造及び構造を定義するフィールドの集合を表す、グラフィカル表現（graphical representation）６００が示されている。加えてグラフィカル表現６００は、データ型構造が定義及び記憶されるＤＭＬファイル６０２（例えば「project.dml」）を識別する。グラフィカル表現６００の一部は、ファイルの外部で呼ばれた場合にファイル内に定義された属性の名前の接頭辞を提供する、「package」と呼ばれるファイルに関するオプション名６０４を提供する。この特定の例において、パッケージ名は空白のままである。

【００２５】
この例では、グラフィカル表現６００は、以下に定義されるように名前付きデータ型構造「person#t」を提示する。

```
type person#t=
record
    string(",")surname;
    string(",")given#name;
    date("YYYY-MM-DD")date#of#birth;
    integer(1)gender; // 0 for male,1 for female
    decimal(9)SSN;
end
```

この例では、グラフィカル表現６００は、閲覧者の左から右へと読み取られるように配向され、最初に名前付きデータ型構造６０６（例えば「person#t」）を閲覧者の左側に提示する。上記で定義された構造から、一連の矩形は、データ型構造「person#t」に入れ子にされた個々のフィールド（例えば、surname（姓）６０８、given#name（名）６１０、date of birth（生年月日）６１２、gender（性別）６１４、及びsocial security number（社会保障番号）６１６）を提示している。グラフィカル表現６００は、名前付きデータ型構造及びそのコンテンツの階層レイアウトを伝える。グラフィカル表現６００の一番左側に配置された名前６０６は構造全体を識別し、個々のフィールド６０８、６１０、６１２、６１４、６１６は更に右側に配置され、データ型構造の階層内の下位レベルに常駐することを示す。この提示から、閲覧者には、名前付きデータ型構造に関連付けられた情報

の読みやすいグラフィカル・レイアウトが提供される。この場合、情報を提示するために矩形の形状が使用されているが、他の形状及び／又は異なる形状の集合も提示に使用可能である。閲覧者が効率的に名前付きデータ型構造の情報を確認するのを支援するために、他のグラフィカル特徴を組み込むことも可能である。例えば、異なる色を実装し、例えば閲覧者に潜在的な問題（例えば、名前付きデータ型構造内に反復又は矛盾するフィールドが定義されていること）を迅速に警告することが可能である。このインスタンスでは、情報を提示するために静的グラフィックが使用されているが、例えば閲覧者の注意を即時に引き付けるために、経時的に変化するグラフィクス（例えばアニメーション、ビデオなど）も使用可能である。例えば左から右への読み取り方向を使用する代わりに、他のグラフィカル表現も使用可能であり、１つ又は複数の名前付きデータ構造を提示するために他のレイアウト及び方向が実装可能である。

【００２６】

図７を参照すると、グラフィカル表現７００は、３つの名前付きデータ型構造が定義された（例えば「project.dml」と名付けられた）ＤＭＬファイルを示す。特に、以前に定義された名前付きデータ型構造「person#t」（図６に図示）のグラフィカル表現６００と共に、２つの追加の（「business#t」及び「class#t」とタイトル付けされた）名前付きデータ型構造がＤＭＬファイル内にグラフィカルに表現されている。

```
type business#t=
record
 string(",")business#name;
 decimal(9)tax#payer#id;
 decimal(11)main#TEL;
 person#t[integer(4)] employees; // here
end
```

及び

```
type class#t=
record
 string(",")home#room#instructor;
 decimal(2)grade;
 person#t[integer(4)] students; / /and here
end;
```

対応するグラフィカル表現７０２、７０４によって示された、「business#t」名前付きデータ型構造及び「class#t」名前付きデータ型構造の両方が、名前付きデータ型構造「person#t」によって定義されたフィールド（例えば、「business#t」名前付きデータ型構造に含まれる「employee」フィールド、及び「class#t」データ型構造に含まれる「student」フィールド）を含む。「business#t」及び「person#t」名前付きデータ型構造による「person#t」名前付きデータ型構造の使用をグラフィカルに示すために、対応する矢印７０６、７０８は、名前付きデータ型構造のペア間の接続を示している。これらのグラフィカルに表された関係から、閲覧者（例えば開発者）は、データ型構造間で共有される情報、（「person#t」などの）以前に定義された名前付きデータ型構造の使用の系統などの、名前付きデータ型構造間の関係を、比較的迅速に識別することができる。名前付きデータ型構造の関係のレイアウトを提供すると共に、グラフィカル表現７００は、潜在的な調整の問題を閲覧者に視覚的に警告する。例えば、「person#t」名前付きデータ型構造が変更される場合、矢印７０６、７０８によって示されるように、「business#t」及び「class#t」の名前付きデータ型構造の両方が、この変更によって影響を受けることになる（例えば、「business#t」名前付きデータ型構造の「employees」フィールド及び「class#t」名前付きデータ型構造の「students」フィールドは、「person#t」名前付きデータ型構造へのいずれの変更も体験することになる）。データ型構造間の関係を提示することと同様に、ファイル間の関係などの他のタイプの関係をグラフィカルに提示することができる。

【００２７】

図８を参照すると、２つのＤＭＬファイルがそれらの関係と共にグラフィカルに表されている。「project.dml」ファイル（図７に表示）のグラフィカル表現７００は、ＤＭＬファイルによって定義された３つの名前付きデータ構造（例えば、「class#t」名前付きデータ構造７０２、「business#t」名前付きデータ構造７０４、及び「person#t」名前付きデータ構造６００）と共に提示されている。加えて、「CRM.dml」とタイトル付けされた別のＤＭＬファイルが、（「sales#prospects#t」とタイトル付けされた）名前付きデータ型構造が示されたグラフィカル表現８００と共に示されている。この例では、２つの名前付きデータ型構造が「CRM.dml」ファイルによって定義され（例えば「educational」及び「commercial」と名付けられ）、２つの名前付きデータ型構造のそれぞれが、「project.dml」によって提供された名前付きデータ型構造を介して定義される。特に「educational」名前付きデータ型構造は「class#t」名前付きデータ型構造から定義され、「commercial」名前付きデータ型構造は「business#t」名前付きデータ型構造によって定義される。「CRM.dml」ファイルと「project.dml」ファイルとの間のこれら２つの関係を表すために、２つの矢印８０２、８０４がファイルの２つのグラフィカル表現をリンクしているものとして示されている。ファイル内でリンクされている名前付きデータ型構造の表現と同様に、２つ又はそれ以上のファイル間のフィールド及び名前付き型定義などのリンクを介して同様の関係を形成することが可能である。例えば、「project.dml」ファイル内の「class#t」名前付きデータ型構造７０２又は「business#t」名前付きデータ型構造７０４の定義に対して行われる（例えば「person#t」名前付きデータ構造６００の変更による）調整は、リンクされた「CRM.dml」ファイル内のフィールド（例えば「educational」及び「commercial」）及び名前付きデータ型構造（例えば「sales#prospects#t」名前付きデータ型構造）に影響を与えることができる。

【００２８】

複数ファイルのフィールド及び名前付きデータ型構造間の関係についてグラフィカル表現を提示すると共に、ファイル間の関係を閲覧者にグラフィカルに表現することができる。例えばファイル・レベル動作を表現することができる。この例では、CRM.dmlファイルのフィールド（例えば「educational」及び「commercial」）が「project.dml」ファイル内の名前付きデータ型構造へのアクセスを達成するために、「project.dml」ファイルが「CRM.dml」ファイルによって識別される必要がある。「project.dml」ファイルへのアクセスを達成するために、例えば「include」ステートメントを「CRM.dml」ファイル内に入力することができる。識別をグラフィカルに表現するために、グラフィカル表現８００内に独自のファイル名８０６をリスト表現すると共に、必要なパッケージ８０８として１つ又は複数の必要なファイル（例えば「project.dml」）も表現される。更にこの例では、破線矢印８１０が、「project.dml」ファイルのコンテンツへのアクセスを達成するために、「include」ステートメントを使用する「CRM.dml」ファイルのファイル・レベル動作をグラフィカルに表現している。

【００２９】

図９ａを参照すると、様々なタイプのグラフィカル表現を使用して、ファイル、フィールド、及び名前付きデータ型構造の間の関係を示すことができる。例えばグラフィカル表現の視覚的な複雑さを低減させるために、様々な量の細部を削減又は除去することができる。示された例では、２つのＤＭＬファイルのグラフィカル表現９００、９０２から個々のフィールド情報が除去される。情報を除去することにより、データ型構造のグラフィカル表現９０４、９０６、９０８、９１０は、各名前付きデータ型構造の名前（例えば、「person#t」、「business#t」、「class#t」、及び「sales#prospects#t」）を簡単に表現することによって簡略化される。グラフィカル表現の視覚的ビジネスの量を削減すると共に、ファイル内の名前付きデータ型構造間の関係を表す矢印９１２、９１４、及び異なるファイル内に常駐する名前付きデータ型構造間の関係を表す矢印９１６、９１８などの、他の情報のために、面積が節約される。同様に、面積が節約されることによって、例えば、ファイル・アクセスを提供するための「include」ステートメントの使用を示すために、破線矢印９２０を用いてファイルが他のファイルを識別する、他の関係を、閲覧者が迅

速に認識するのを助けることができる。名前付きデータ型構造及びそれらの関係の簡潔なビューを閲覧者（例えば開発者）に提供すると共に、グラフィカル表現によって他の機能を提供することができる。例えば、フィールド、名前付きデータ型構造、及び関連情報を操作することは、グラフィカル表現を使用することによってより効率的に実行することができる。

【００３０】

図９ｂを参照すると、ファイル・データ型構造、ファイルなどを効率的に構築、再構築などするために、グラフィカル表現を調整及び操作することができる。例えば、新規ファイルを定義するために、データ構造の表現を別々にグループ化することができる。図に示されるように、ファイルのグラフィカル表現（例えばＤＭＬファイル表現９００及び９０２）が除去され、開発者はデータ型構造のグラフィカル表現９０４、９０６、９０８、９１０のグループ化を調整することができる。同じか又は異なるファイル内に記憶するためにデータ型構造を認識可能にすると共に、こうした操作は、ファイル間の関係を向上させ、不必要な「include」ステートメントの発生を削減することができる。単に示された例で説明するために、（データ型構造「business#t」に関する）グラフィカル表現９０６を、より効率的な動作のために、（データ型構造「sales#prospects#t」に関する）グラフィカル表現９１０とグループ化することができる。グループ化されると、グラフィカル表現は、新規にグループ化されたデータ型構造を記憶するためのファイル作成を開始することができる。データ型構造のグループ化及び操作のためにこうした動作を使用すると共に、ファイル・レベル動作を実行することも可能である。例えば、ファイルのコンテンツ（例えばデータ型構造）の組み合わせ、除去、付加などのために、ファイルのグラフィカル表現（例えば表現９００、９０２）を操作することが可能である。

【００３１】

図１０を参照すると、フィールド、名前付きデータ型構造、及び他のタイプのプログラミング属性のグラフィカル表現を操作するためのエディタ１００２を提供する、ユーザ・インターフェース１０００が示されている。エディタ１００２は、フィールド、名前付きデータ型構造、及び関連情報（例えばデータ型の関係を表現するための矢印）のグラフィカル表現を提示する、ウィンドウ１００４を含む。エディタ１００２は、ユーザ（例えば開発者）が、開発中のアプリケーション内に含めるために、様々なフィールド、名前付きデータ型構造などから選択できるようにするパレット１００６も含む。例えば、図中に太字の矢印１００８で示されるように、ポインティング・デバイスを使用して、名前付きデータ型構造を選択し、プロジェクト開発用のウィンドウ１００４内に挿入すること（例えばドラッグ・アンド・ドロップ）ができる。同様に、選択及び挿入動作を逆に実行して、（開発された後の）名前付きデータ型構造をウィンドウ１００４から選択し、パレット１００６内に挿入することができる。選択及び挿入動作は、ウィンドウ１００４又はパレット１００６内で単独に実行することも可能である。例えば動作（例えばドラッグ・アンド・ドロップ動作）は、１つ又は複数のフィールド、名前付きデータ型構造などの作成、編集などのために、パレット１００６内で実行することができる。同様に、動作（例えば選択、挿入、削除、付加など）は、フィールド、名前付きデータ型構造などを調整するために、ウィンドウ１００４内でユーザによって開始することができる。他のタイプの操作動作も実行することが可能であり、例えば、フィールド、名前付きデータ型構造、ファイル（例えばＤＭＬファイル）などの間の関係をグラフィカルに操作することができる。

【００３２】

図１１を参照すると、フィールド、名前付きデータ型構造、ファイルなど、及び関係を操作するための動作は、ユーザ（例えば開発者）によってグラフィカルに開始することができる。例えば、フィールド及び名前付きデータ型構造間の関係を調整するために、矢印を操作（例えば削除、追加、移動など）することができる。この図示された例では、２本の線１１００及び１１０２が削除される（ユーザのポインティング・デバイスを介してグラフィカル記号「ｘ」を各線上にそれぞれ配置することによって示されている）。線の削除によって「project.dml」ファイルと「CRM.dml」ファイルとの間で処理される関係に基

づき、「CRM.dml」はもはや「project.dml」ファイルのコンテンツにアクセスする必要がない。したがって、「project.dml」ファイルにアクセスするための「CRM.dml」ファイル内の命令（例えば「include「project.dml」」）は、（破線ボックス１１０４によって示されるように）「CRM.dml」ファイルから除去される。したがって、「CRM.dml」ファイルと「project.dml」ファイルとの間のこの関係を表す破線１１０６も、同様に、ファイルのグラフィカル表現から除去することができる。別の方法として、ファイル間のこうした関係が（例えば２つのファイルを線１１００及び１１０２によって接続することによって）確立又は再確立される場合、命令（例えば「include「project.dml」」）を適切なファイル（例えば「CRM.dml」）内に挿入又は再挿入することが可能であり、この関係を示すために、グラフィカル表現（例えば破線１１０６）を再度提示することができる。

【００３３】

図１２を参照すると、名前付きフィールド、名前付きデータ型構造、ファイルなどの量が増えるにつれて、ユーザ（例えば開発者）が、アプリケーション開発中に使用するために選択可能な潜在的フィールド、名前付きデータ型構造、ファイルなどの間をナビゲートするのを支援するために、１つ又は複数の技法を実装することができる。例えば、１つ又は複数のグラフィカル表現は、使用可能なフィールド及び名前付きデータ型構造を含むファイルの選択可能リストを提示することができる。いくつかの配置構成において、階層リストを使用して、ユーザがファイル、名前付きデータ型構造などをナビゲートする際に支援することができる。図に示されるように、ユーザが異なるパッケージ（例えば、ＸＭＬ処理データ型、ルックアップ・データ型、メタ・プログラミング・データ型、日付／時刻データ型、メタデータ型など）のリスト間をナビゲートできるようにするペイン１２００が、ユーザ・インターフェース１２０２内に含められる。一配置構成において、選択が実行されると、名前付きデータ型構造及び関数などの選択されたパッケージ内に定義されたアーチファクトを、ユーザ・インターフェース１２０２の右側に配置されたグラフィカル表現内に表示し、必要に応じて操作（例えば選択、ナビゲート、ペイン１２００上へのドラッグ・アンド・ドロップ）することができる。

【００３４】

アプリケーション開発のためのフィールド、名前付きデータ型構造、ファイルの編成、操作などと共に、開発者を支援するために他のタイプのプログラミング属性を同様にグラフィカル表現することができる。例えば、アプリケーションによって使用される関数、変数などは、更に多くのこうしたプログラミング属性が作成され、後に取り出して再使用するためにライブラリに記憶されるにつれて、同様に扱いにくくなっていく可能性がある。開発者が適切なプログラミング属性を識別及び選択する際に支援するための、他の技法も実装可能である。例えば、ファイル間でプログラミング属性を論理的に分散させるためのアルゴリズム（クラスタリング・アルゴリズムと呼ばれる）を、例えばフィールド、名前付きデータ型構造、関数などを編成するために使用することができる。こうした編成技法を介して、名前付きデータ型構造の冗長使用と共に、命令（例えば「include」ステートメント）の量を削減することができる。

【００３５】

図１３を参照すると、フローチャート１３００が、アプリケーション開発で使用される名前付きデータ型構造などのプログラミング属性をグラフィカルに表現するための手順の動作を示している。動作は、典型的には単一のコンピューティング・デバイス（例えば開発環境を提供する）によって実行されるが、動作は、複数のコンピューティング・デバイスによる実行も可能である。単一サイトでの実行と共に、動作の実行は２つ又はそれ以上の場所に分散させることが可能である。

【００３６】

動作は、アプリケーションを開発するための１つ又は複数のプログラミング属性の階層を表すデータ構造を定義すること１３０２を、含むことができる。１つ又は複数のデータ構造グループを定義するため、及び１つ又は複数のデータ構造グループを記憶するための１つ又は複数の新規ファイルを作成するために、データ構造のグラフィカル表現を操作で

きるようにするために、ファイルのグラフィカル表現を除去することができる。例えば、サブレコード、レコード、フィールド、名前付きデータ型構造などの階層を使用して、データ構造を定義することができる。データ構造は（記憶のために）単一ファイル内に含めるように定義されるが、いくつかの配置構成では、データ構造は（記憶及びその後の取り出しのために）複数ファイル内に含めることが可能である。動作は、データ構造のグラフィカル表現及びデータ構造を記憶するファイルのグラフィカル表現を含む視覚図を生成すること１３０４も含む。視覚図は、データ構造と別のデータ構造との間の関係のグラフィカル表現、及び、データ構造を記憶するファイルと他のデータ構造を記憶するファイルとの間の関係のグラフィカル表現も含む。例えば図７に示されるように、２つの名前付きデータ型構造（例えば「business#t」及び「class#t」）が、ファイル「project.dml」内に含まれる名前付きデータ型構造を表す視覚図内にグラフィカルに表現される。同じく視覚図内に示されたグラフ線７０６、７０８によって示されるように、他の名前付きデータ型構造（例えば「business#t」及び「class#t」）の両方によって使用されるフィールドの階層を含む、第３のデータ型構造（例えば「person#t」）も提示される。各データ構造のコンテンツを示すと共に、例えば、フィールド、名前付きデータ型構造などの系統及びそれらの関係を、閲覧者（例えば開発者）が比較的迅速に確認できるようにするために、データ構造間の関係がグラフィカルに表現されている。

【００３７】

前述の計算アーチファクトをグラフィカルに表現するための手法は、コンピュータ上で実行するためのソフトウェアを使用して実装可能である。例えば、ソフトウェアは、それぞれが少なくとも１つのプロセッサ、少なくとも１つのデータ記憶システム（揮発性及び不揮発性のメモリ及び／又は記憶素子を含む）、少なくとも１つの入力デバイス又はポート、及び少なくとも１つの出力デバイス又はポートを含む、１つ又は複数のプログラム済み又はプログラム可能なコンピュータ・システム（分散型、クライアント／サーバ、又はグリッドなどの様々なアーキテクチャとすることが可能である）上で実行する、１つ又は複数のコンピュータ・プログラム内に手順を形成する。ソフトウェアは、例えば、データフロー・グラフの設計及び構成に関する他のサービスを提供する、より大きなプログラムの１つ又は複数のモジュールを形成することが可能である。グラフのノード及び要素は、コンピュータ読み取り可能媒体内に記憶されたデータ構造、又は、データ・リポジトリ内に記憶されたデータ・モデルに準拠する他の編成済みデータとして実装可能である。

【００３８】

ソフトウェアは、汎用又は特定用途向けのプログラム可能コンピュータによって読み取り可能であるか、或いは、ネットワークの通信媒体を介して、実行されるコンピュータの記憶媒体へと送達される（伝搬信号内に符号化される）、ＣＤ－ＲＯＭなどの、記憶媒体上に提供することが可能である。すべての機能は、特定用途向けコンピュータ上で、又は、コプロセッサなどの特定用途向けハードウェアを使用して、実行可能である。ソフトウェアは、ソフトウェアによって指定された計算の異なる部分が異なるコンピュータによって実行される、分散様式で実装可能である。こうした各コンピュータ・プログラムは、好ましくは、本明細書で説明される手順を実行するために、記憶媒体又はデバイスがコンピュータ・システムによって読み取られた時にコンピュータを構成及び動作させるために、記憶媒体又はデバイス（例えば、ソリッド・ステート・メモリ又は媒体、或いは磁気又は光媒体）上に記憶されるか又はダウンロードされる。本発明のシステムは、コンピュータ・プログラムで構成された、コンピュータ読み取り可能記憶媒体として実装されるものとみなすことも可能であり、そのように構成された記憶媒体は、本明細書で説明された機能を実行するために、特定の事前に定義された様式でコンピュータ・システムを動作させる。

【００３９】

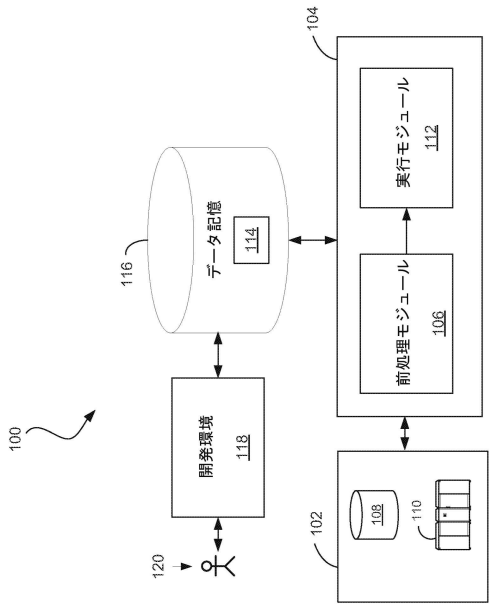
以上、本発明のいくつかの実施形態を説明してきた。しかしながら、本発明の趣旨及び範囲を逸脱することなく、様々な修正が実行可能であることを理解されよう。例えば、上記で説明したステップのいくつかは順序に依存していない場合があるため、説明した順序

とは異なる順序で実行することができる。

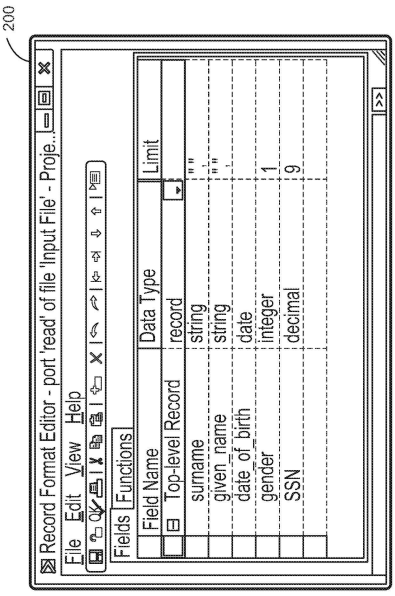
【００４０】

前述の説明は、添付の特許請求の範囲によって定義される本発明の範囲を例示するものであり、それらを制限することは意図されていない旨を理解されよう。例えば、前述のいくつかの機能ステップは、処理全体に大きく影響を与えることなく異なる順序で実行可能である。他の実施形態は、以下の特許請求の範囲の範囲内にある。

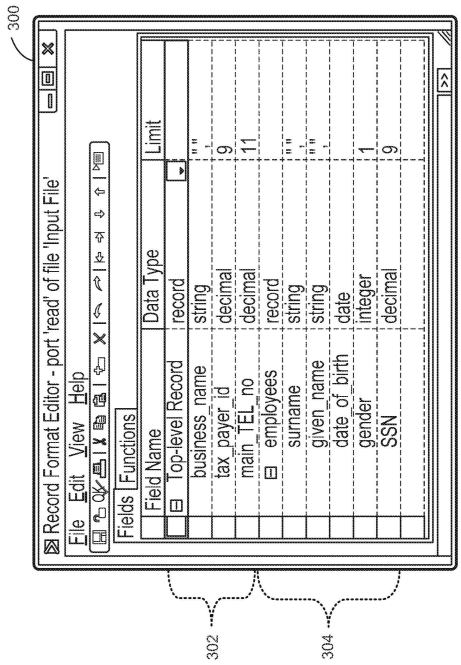
【図１】



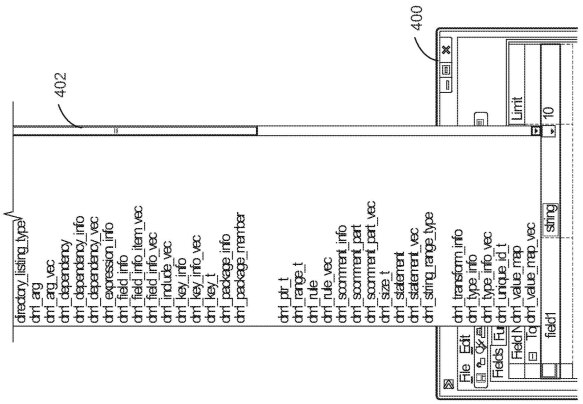
【図２】



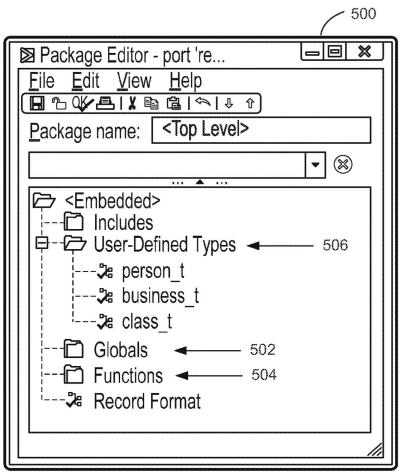
【図３】


300
Record Format Editor - port 'read' of file 'Input File'
File Edit View Help
Fields Functions
Field Name
Data Type
Limit
Top-level Record
record
business_name
string
tax_payer_id
decimal
9
main_TEL_no
decimal
11
employees
record
surname
string
given_name
string
date_of_birth
date
gender
integer
1
SSN
decimal
9
302
304

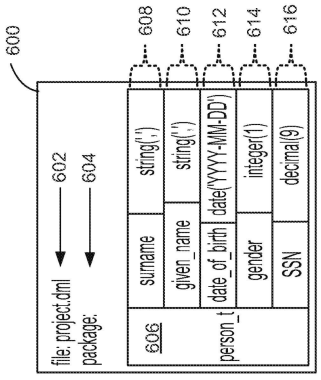
【図４】


400
402
directory_listing_type
dml_arg
dml_arg_vec
dml_dependency
dml_dependency_info
dml_dependency_vec
dml_expression_info
dml_field_info
dml_field_info_item_vec
dml_field_info_vec
dml_include_vec
dml_key_info
dml_key_info_vec
dml_key_t
dml_package_info
dml_package_member
dml_ptr_t
dml_range_t
dml_rule
dml_rule_vec
dml_scomment_info
dml_scomment_part
dml_scomment_part_vec
dml_size_t
dml_statement
dml_statement_vec
dml_string_range_type
dml_transform_info
dml_type_info
dml_type_info_vec
dml_unique_id_t
dml_value_map
dml_value_map_vec
field1
string
Limit
10

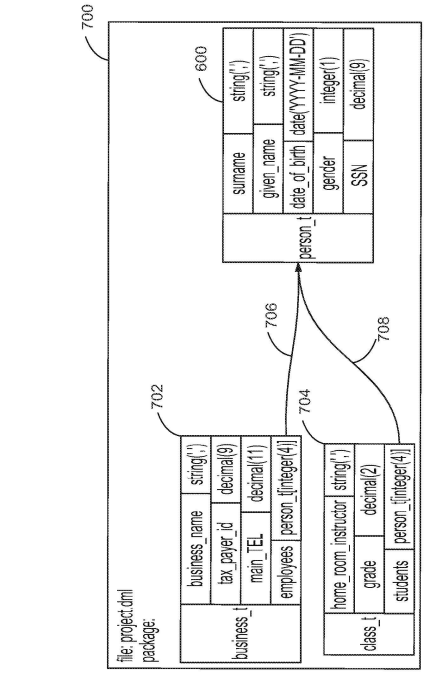
【図５】


500
Package Editor - port 're...
File Edit View Help
Package name: <Top Level>
<Embedded>
Includes
User-Defined Types
506
person_t
business_t
class_t
Globals
502
Functions
504
Record Format

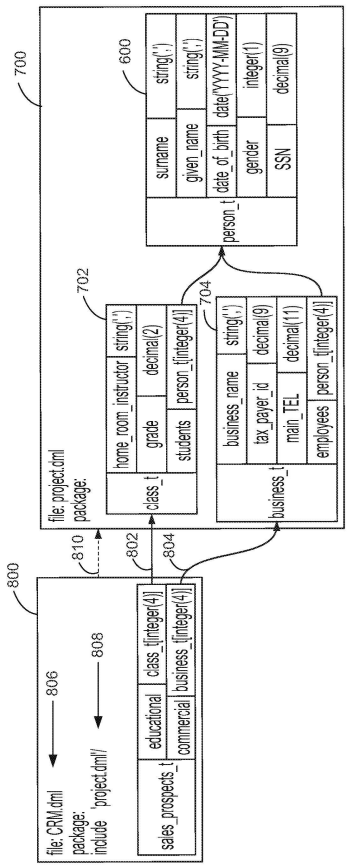
【図６】


600
608
610
612
614
616
602
604
file: project.dml
package:
606
person_t
surname
given_name
date_of_birth
gender
SSN
string(',')
string(',')
date('YYYY-MM-DD')
integer(1)
decimal(9)

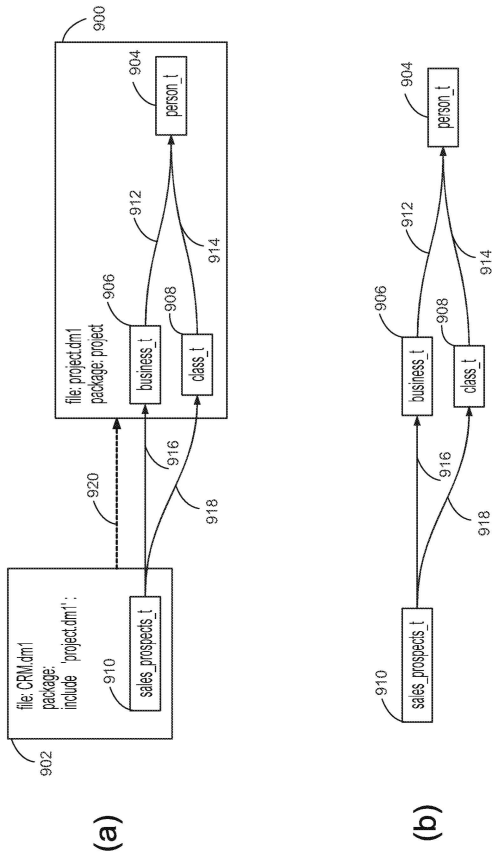
【図７】


700
file: project.dml
package:
business_t
business_name
tax_payer_id
main_TEL
employees
string(',')
decimal(9)
decimal(11)
person_t[integer(4)]
class_t
home_room_instructor
grade
students
string(',')
decimal(2)
person_t[integer(4)]
person_t
surname
given_name
date_of_birth
gender
SSN
string(',')
string(',')
date('YYYY-MM-DD')
integer(1)
decimal(9)
600
702
704
706
708

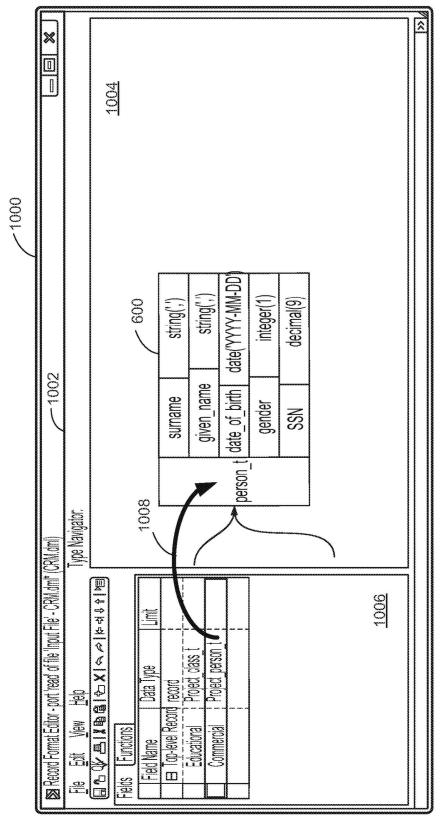
【図８】


700
file: project.dml
package:
600
702
704
810
802
804
800
file: CRM.dml
package:
include 'project.dml//'
806
808
sales_prospects_t
educational
commercial
class_t[integer(4)]
business_t[integer(4)]

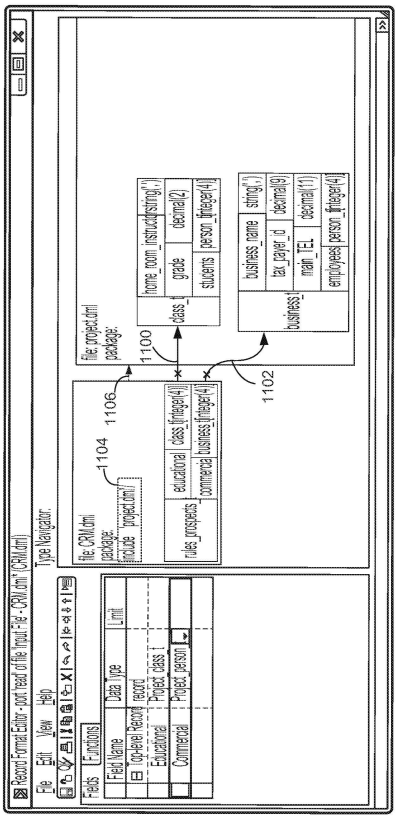
【図９】


900
file: project.dm1
package: project
file: CRM.dm1
package:
include 'project.dm1';
sales_prospects_t
business_t
class_t
person_t
902
904
906
908
910
912
914
916
918
920
(a)
(b)

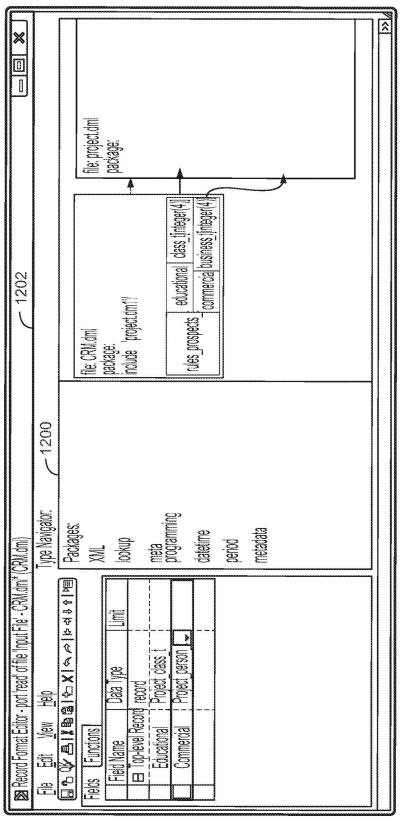
【図１０】


1000
1002
1004
1006
1008
600
Record Format Editor - work head of file 'Input File' -'CRM.dm1' (CRM.dm1)
File Edit View Help
Type Navigator:
Fields Functions
Field Name
Data Type
Limit
top-level record
educational
commercial
Project_class_t
Project_person_t
person_t
surname
given_name
date_of_birth
gender
SSN
string(',')
string(',')
date('YYYY-MM-DD')
integer(1)
decimal(9)

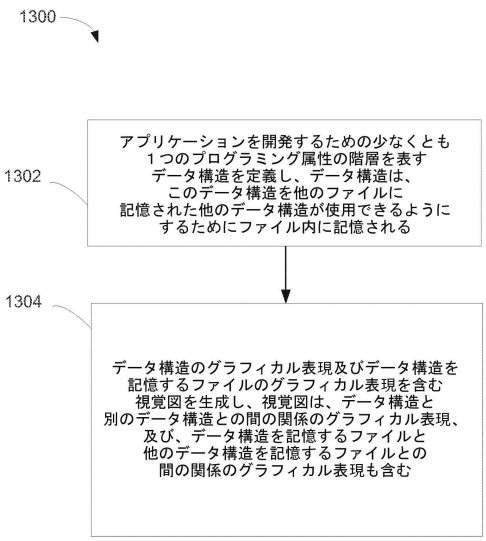
【図１１】


Record Format Editor - root head of file 'input' File - CRM.dmi (CRM.dmi)
File Edit View Help
Type Navigator:
Fields Functions
Field Name Data Type Limit
top-level Record record
Educational Project_class_1
Commercial Project_person
file: project.dmi
package:
class_1
home_room_instructor&string(*,)
grade decimal(2)
students person_t(integer(4))
business
business_name string(*,1)
tax_payer_id decimal(9)
main_TEL decimal(11)
employees person_t(integer(4))
file: CRM.dmi
package:
include 'project.dmi'!
rules_prospects
educational class_t(integer(4))
commercial business_t(integer(4))
1100
1102
1104
1106

【図１２】


Record Format Editor - root head of file 'input' File - CRM.dmi (CRM.dmi)
File Edit View Help
Type Navigator:
Packages:
XML
lookup
meta programming
datetime
period
metadata
Fields Functions
Field Name Data Type Limit
top-level Record record
Educational Project_class_1
Commercial Project_person
file: project.dmi
package:
file: CRM.dmi
package:
include 'project.dmi'!
rules_prospects
educational class_t(integer(4))
commercial business_t(integer(4))
1200
1202

【図１３】


1300
1302
アプリケーションを開発するための少なくとも
１つのプログラミング属性の階層を表す
データ構造を定義し、データ構造は、
このデータ構造を他のファイルに
記憶された他のデータ構造が使用できるように
するためにファイル内に記憶される
1304
データ構造のグラフィカル表現及びデータ構造を
記憶するファイルのグラフィカル表現を含む
視覚図を生成し、視覚図は、データ構造と
別のデータ構造との間の関係のグラフィカル表現、
及び、データ構造を記憶するファイルと
他のデータ構造を記憶するファイルとの
間の関係のグラフィカル表現も含む

フロントページの続き

(72)発明者　アンダーソン，アーレン
イギリス国，キッドリントン　オーエックス５　２エスジー，アイスリップ，ローワー　ストリート　３

審査官　三坂　敏夫

(56)参考文献　国際公開第０１／０８２０６８（WO，Ａ１）
国際公開第０１／０８２０７２（WO，Ａ１）
特開平０４－２６７４３４（ＪＰ，Ａ）
特開平０７－３１９６７６（ＪＰ，Ａ）
特表２０１２－５１０６８８（ＪＰ，Ａ）
米国特許出願公開第２００２／００９７２５３（ＵＳ，Ａ１）
米国特許出願公開第２００２／００３２９００（ＵＳ，Ａ１）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　　８／００－　８／７７
９／４４－　９／４５１
３／０４８１