(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2015-84146
(P2015-84146A)
(43)公開日　平成27年4月30日(2015.4.30)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 9/44	(2006.01)	G06F	9/06	620A	5B376

審査請求 未請求　請求項の数 7　ＯＬ　（全 14 頁）

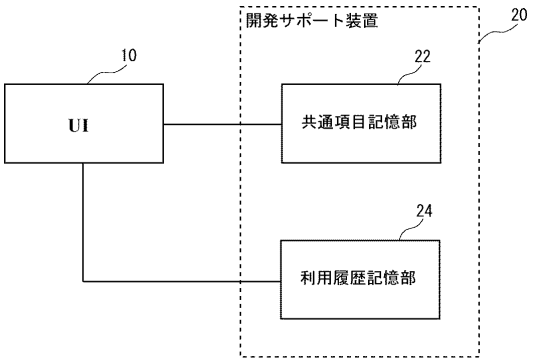
(21)出願番号　特願2013-222094 (P2013-222094)
(22)出願日　平成25年10月25日 (2013.10.25)

(71)出願人　000004237
日本電気株式会社
東京都港区芝五丁目７番１号
(74)代理人　100103894
弁理士　家入　健
(72)発明者　安部　一郎
東京都港区芝五丁目７番１号　日本電気株式会社内

Ｆターム(参考)　5B376 BC74　BC79　DA16　DA18　DA24

(54)【発明の名称】プログラム開発サポート装置および方法

(57)【要約】
【課題】システムの保守性を確実に高めるよう、プログラムの開発をサポートする技術。
【解決手段】プログラム開発サポート装置２０において、共通項目記憶部２２は、予め作成され、複数のプログラムにより参照される共通項目を記憶し、利用履歴記憶部２４は、夫々のプログラムについて、共通項目記憶部２２に記憶された共通項目の参照履歴を記録する。これらの共通項目は、プログラムのコーディング時に他のコードの入力操作とは異なる特定操作で共通項目記憶部２２から入力するように規定されている、利用履歴記憶部２４は、プログラムのコーディング時における上記特定操作の履歴から該プログラムの参照履歴を得て記録する。
【選択図】図１



1

【特許請求の範囲】
【請求項１】
予め作成され、複数のプログラムにより参照される共通項目を記憶する共通項目記憶部と、
夫々の前記プログラムについて、前記共通項目の参照履歴を記録する利用履歴管理部とを備え、
前記共通項目は、前記プログラムのコーディング時に他のコードの入力操作とは異なる特定操作で前記共通科目記憶部から入力するように規定されており、
前記利用履歴管理部は、前記プログラムのコーディング時における前記特定操作の履歴から該プログラムの前記参照履歴を得る、
プログラム開発サポート装置。
【請求項２】
各前記プログラムは、処理単位のノード毎に分割されてコーディングされるものであり、
前記利用履歴管理部は、夫々の前記プログラムについて、該プログラムを構成するノード毎に前記参照履歴を記録する、
請求項１に記載のプログラム開発サポート装置。
【請求項３】
インパクト分析時に、いずれかの前記共通項目が指定されると、プログラム毎に、該共通項目に対応する前記参照履歴ををを表示するインパクト分析サポート部をさらに備える、
請求項１または２に記載のプログラム開発サポート装置。
【請求項４】
記憶装置により、予め作成された、複数のプログラムにより参照される共通項目を記憶し、
夫々の前記プログラムについて、前記共通項目の参照履歴を記録して管理するステップを有し、
前記共通項目は、前記プログラムのコーディング時に他のコードの入力操作とは異なる特定操作で前記記憶装置から入力するように規定されており、
夫々の前記プログラムの前記参照履歴は、該プログラムのコーディング時における前記特定操作の履歴から得られる、
プログラム開発サポート方法。
【請求項５】
各前記プログラムは、処理単位のノード毎に分割されてコーディングされるものであり、
夫々之前記プログラムについて、前記参照履歴は、該プログラムを構成するノード毎に記録される、
請求項４に記載のプログラム開発サポート方法。
【請求項６】
インパクト分析時に、いずれかの前記共通項目が指定されると、プログラム毎に、該共通項目に対応する前記参照履歴ををを表示する、
請求項４または５に記載のプログラム開発サポート方法。
【請求項７】
請求項４から６のいずれか１項に記載のプログラム開発サポート方法をコンピュータに実行せしめるプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、プログラム開発をサポートする技術に関する。
【背景技術】
【０００２】

人間にとって理解し易い仕様記述から、そこに記述されたユーザの要求を満足するプログラムを自動的に生成するプログラム自動生成技術の研究が行われており、プログラム自動生成ツールが開発されている（例えば特許文献１）。

【０００３】

プログラムは、生成された後に、往々にして、変更を要求される場合がある。また、プログラムの変更時には、通常、変更の危険度を測定する所謂インパクト分析が行われている（例えば特許文献２）。

【０００４】

プログラムの自動生成技術があるものの、プログラム変更時のインパクト分析は、ソースコード全件の目視確認、もしくはテキストエディタによるキーワード検索等の手作業で行われている現状である。システム規模が小さい場合には、手作業でも大きな問題が無いが、システムが大規模になるほど、ソースコードが増え、インパクト分析の工数、難易度が増大し、調査漏れ等のリスクも高まる。特に、既に運用フェーズに入ってからプログラムを変更するケースも多いため、少しのミスが大問題になるケースも多く、本番稼働後のプログラム変更、特にインパクト分析は開発者が最も神経を使う部分であり、一番頭を悩ませる問題である。

【０００５】

この問題は、バッチ処理システムの場合においてさらに顕著である。

バッチ処理システムは、複数の処理を夫々担う複数のプログラムが所定の順序で実行されるシステムである。以下、これらのプログラムを「バッチプログラム」という。１つのバッチ処理システムにおいて、複数のバッチプログラム間の共通のレコード（以下「共通項目」という）も、異なるバッチプログラムにおいては、異なる定義文で定義されている場合がある。

【０００６】

共通項目なのに、バッチプログラム間で定義文が異なるのでは、プログラムの変更や変更時のインパクト分析の効率が悪く、ミスも生じやすい。

【０００７】

特許文献３に開示されたデータ項目定義標準化装置は、各バッチプログラムのソースコードを解析し、夫々のバッチプログラムから、上述した共通項目の定義文を抽出する。次いで、共通項目毎に、該共通項目に対して抽出した複数の定義文から１つの定義文を選択して標準コピーとして保持しておく。そして、各バッチプログラムに対して、該共通項目については、標準コピーを参照するようにソースコードを書き換える。

【０００８】

特許文献３の手法によれば、プログラムの変更時やインパクト分析時において、共通項目については、標準コピーを変更／調査すればよいため、効率が良いと共に、ミスも軽減することができる。

【先行技術文献】

【特許文献】

【０００９】

【特許文献１】特開平９－６４８８７号公報
【特許文献２】特開２００３－０４４２７５号公報
【特許文献３】特開平７－１０４９８９号公報

【発明の概要】

【発明が解決しようとする課題】

【００１０】

しかしながら、複数のプログラムのソースコードを解析して共通項目を抽出するのは容易なことではない。共通項目の抽出が失敗して、互いに異なるものが共通項目として抽出されたなどの場合には、システムに大きなダメージを与え、システムの保守性を反って悪化させてしまう恐れがある。

【００１１】

本発明は、上記事情に鑑みてなされたものであり、プログラムの開発に対して、システムの保守性を確実に高めることができるサポート技術を提供する。

【課題を解決するための手段】

【００１２】

本発明の１つの態様は、プログラム開発サポート装置である。該装置は、予め作成され、複数のプログラムにより参照される共通項目を記憶する共通項目記憶部と、夫々の前記プログラムについて、前記共通項目の参照履歴を記録する利用履歴管理部とを備える。

【００１３】

前記共通項目は、前記プログラムのコーディング時に他のコードの入力操作とは異なる特定操作で前記共通科目記憶部から入力するように規定されている。

【００１４】

前記利用履歴管理部は、前記プログラムのコーディング時における前記特定操作の履歴から該プログラムの前記参照履歴を得る。

【００１５】

なお、上記態様のサポート装置をシステムや方法に置き換えて表現したもの、コンピュータを該サポート装置として動作させるプログラム、該プログラムを記録した記録媒体、該サポート装置を含むプログラム開発システム等も、本発明の態様としては有効である。

【発明の効果】

【００１６】

本発明にかかる技術によれば、プログラムの開発をサポートし、システムの保守性を確実に高めることができる。

【図面の簡単な説明】

【００１７】

【図１】第１の実施の形態にかかるプログラム開発システムを示す図である。

【図２】第２の実施の形態にかかるプログラム設計管理システムを示す図である。

【図３】図２に示すプログラム設計管理システムにおけるプログラム開発画面の例を示す図である。

【図４】ノードの種類毎のコーディング画面の例を示す図である。

【図５】共通項目を入力する特定の操作のイメージを示す図である（その１）。

【図６】共通項目を入力する特定の操作のイメージを示す図である（その２）。

【図７】共通項目を入力する特定の操作のイメージを示す図である（その３）。

【図８】参照履歴の記録例を示す図である（その１）。

【図９】参照履歴の記録例を示す図である（その２）。

【図１０】参照履歴の記録例を示す図である（その３）。

【図１１】図２に示すプログラム設計管理システムによるインパクト分析を説明するための図である（その１）。

【図１２】図２に示すプログラム設計管理システムによるインパクト分析を説明するための図である（その２）。

【図１３】図２に示すプログラム設計管理システムによるインパクト分析を説明するための図である（その３）。

【図１４】図２に示すプログラム設計管理システムによるインパクト分析を説明するための図である（その４）。

【図１５】図２に示すプログラム設計管理システムによるインパクト分析を説明するための図である（その５）。

【発明を実施するための形態】

【００１８】

以下、図面を参照して本発明の実施の形態について説明する。説明の明確化のため、以下の記載及び図面は、適宜、省略、及び簡略化がなされている。また、様々な処理を行う機能ブロックとして図面に記載される各要素は、ハードウェアとソフトウェア（プログラム）の組合せによっていろいろな形で実現できることは当業者には理解されるところであ

り、ハードウェアとソフトウェアのいずれかに限定されるものではない。なお、各図面において、同一の要素には同一の符号が付されており、必要に応じて重複説明は省略されている。

【００１９】

また、上述したプログラムは、様々なタイプの非一時的なコンピュータ可読媒体（ｎｏｎ－ｔｒａｎｓｉｔｏｒｙ　ｃｏｍｐｕｔｅｒ　ｒｅａｄａｂｌｅ　ｍｅｄｉｕｍ）を用いて格納され、コンピュータに供給することができる。非一時的なコンピュータ可読媒体は、様々なタイプの実体のある記録媒体（ｔａｎｇｉｂｌｅ　ｓｔｏｒａｇｅ　ｍｅｄｉｕｍ）を含む。非一時的なコンピュータ可読媒体の例は、磁気記録媒体（例えばフレキシブルディスク、磁気テープ、ハードディスクドライブ）、光磁気記録媒体（例えば光磁気ディスク）、ＣＤ－ＲＯＭ（Ｒｅａｄ　Ｏｎｌｙ　Ｍｅｍｏｒｙ）ＣＤ－Ｒ、ＣＤ－Ｒ／Ｗ、半導体メモリ（例えば、マスクＲＯＭ、ＰＲＯＭ（Ｐｒｏｇｒａｍｍａｂｌｅ　ＲＯＭ）、ＥＰＲＯＭ（Ｅｒａｓａｂｌｅ　ＰＲＯＭ）、フラッシュＲＯＭ、ＲＡＭ（Ｒａｎｄｏｍ　Ａｃｃｅｓｓ　Ｍｅｍｏｒｙ））を含む。また、プログラムは、様々なタイプの一時的なコンピュータ可読媒体（ｔｒａｎｓｉｔｏｒｙ　ｃｏｍｐｕｔｅｒ　ｒｅａｄａｂｌｅ　ｍｅｄｉｕｍ）によってコンピュータに供給されてもよい。一時的なコンピュータ可読媒体の例は、電気信号、光信号、及び電磁波を含む。一時的なコンピュータ可読媒体は、電線及び光ファイバ等の有線通信路、又は無線通信路を介して、プログラムをコンピュータに供給できる。

【００２０】

＜第１の実施の形態＞

図１は、第１の実施の形態にかかるプログラム開発システム１を示す。プログラム開発システム１は、例えばバッチ処理システムを構成する複数のバッチプログラムを開発するためのコンピュータであり、ユーザ・インタフェース（Ｕｓｅｒ　Ｉｎｔｅｒｆａｃｅ：ＵＩ）１０、プログラム開発サポート装置２０を備える。なお、図１において、本発明の技術を説明する上で必要な機能ブロックのみを示し、この種のシステムに通常備えられる他の構成要素については、省略する。

【００２１】

ＵＩ１０は、プログラムの開発者がソースコードの入力用のユーザ・インタフェースであり、キーボード、マウスなどの入力手段や、モニタなどの表示手段を含む。

【００２２】

プログラム開発サポート装置２０は、バッチプログラムのコーディング時に、複数のバッチプログラム間の共通項目の入力の効率化と、開発したバッチ処理システムの保守性を高めるためのものであり、共通項目記憶部２２と利用履歴記憶部２４を備える。

【００２３】

本実施の形態において、バッチプログラムのコーディングについて、該バッチプログラムを含む複数のバッチプログラム間の共通項目は、他のソースコードの入力操作とは異なる特定操作で共通項目記憶部２２から入力するように規定されている。

【００２４】

例えば、他のソースコードについては、通常のコーディングと同様に、ユーザがＵＩ１０に含まれるキーボードで直接入力するようになっているが、共通項目記憶部２２に格納された共通項目については、ドラッグ＆ドロップ操作で共通項目記憶部２２から読み出して入力するようになっている。

【００２５】

なお、ドラッグ＆ドロップ操作で共通項目記憶部２２から当該共通項目を読み出して入力するためには、例えば、共通項目記憶部２２に記憶された共通項目の一覧をＵＩ１０の表示画面に展開し、ドラッグ＆ドロップ操作でコーディング中のバッチプログラムのカーソル位置に入力できるようにすればよい。

【００２６】

共通項目記憶部２２は、上述したような、複数のバッチプログラムにより参照される共

通項目を記憶している。これらの共通項目は、予め作成され、共通項目記憶部２２に格納される。

【００２７】

バッチ処理システムにおいて、システム全体に大きな影響を与える共通項目としては、テーブルの入出力項目、ファイルの入出力項目、共通関数、クラス継承時の親クラス、共通定数、環境変数などがある。

【００２８】

共通項目の作成については、例えば、バッチ処理システムの設計者が予めプログラムコードの形で作成して共通項目記憶部２２に格納するようにしてもよい。または、特許文献１に開示されたようなプログラム自動生成ツールを利用する場合には、バッチ処理システムの設計者が作成した仕様記述に基づいて、プログラム自動生成ツールが、個々のバッチプログラムのコーディングの前に作成して共通項目記憶部２２に格納するようにしてもよい。

【００２９】

利用履歴記憶部２４は、バッチプログラム毎に、それのコーディング時における共通項目の参照履歴を記録する。例えば、バッチプログラムＡのコーディング時に、特定の操作（本実施の形態ではドラッグ＆ドロップ操作）により共通項目記憶部２２に記憶された共通項目Ａ１の入力が３回行われた場合に、利用履歴記憶部２４は、バッチプログラムＡについて、利用履歴として、例えば「共通項目Ａ１、３回」旨の内容を記録する。

【００３０】

このように、本実施の形態にかかるプログラム開発システム１では、プログラム開発サポート装置２０は、予め作成された共通項目を記憶しておき、バッチプログラム毎に、該バッチプログラムのコーディング時における特定の操作（ドラッグ＆ドロップ操作）の履歴に応じて、夫々の共通項目の参照履歴を得て記録する。

【００３１】

まず、複数のプログラムで参照される共通項目をソースコードと独立させて一元管理し、画面上に表示させて、手入力なしに特定の操作で入力させることで、ユーザの入力ミスを排除することができる。また、プログラム毎に共通項目の扱いにバラつきが生じないようになるため、全ソースコードが統一感の取れたものになり、ソースコードの保守性が増す。

【００３２】

また、バッチプログラム毎に、共通項目記憶部２２に記憶された共通項目の参照履歴が記録されているため、プログラムの変更時やインパクト分析時には、利用履歴記憶部２４に記憶された参照履歴を参照することにより、複数のバッチプログラム間での共通項目の共通項目の利用状況をプログラム毎に細かく把握できるようになるため、インパクト分析工数と修正漏れ・考慮漏れ等のユーザミスを軽減できる。システムが大規模になればなるほど有効性が増す。

【００３３】

さらに、予め作成された共通項目を記憶しておき、共通項目の入力に用いると規定された特定の操作の履歴から、共通項目の各バッチプログラムにおける分布状況に該当する参照履歴を得ているため、バッチプログラムのソースコードを解析することにより共通項目を取得して置き換える特許文献２の手法より簡単かつ確実である。

【００３４】

＜第２の実施の形態＞

図２は、第２の実施の形態にかかるプログラム設計管理システム５０を示す。プログラム設計管理システム５０は、第１の実施の形態にかかるプログラム開発システム１をより具現化したものであり、バッチプログラムの開発及び保守をより強力にサポートするものである。以下において、バッチプログラムを単に「プログラム」ともいう。

【００３５】

プログラム設計管理システム５０は、ユーザが作成した仕様記述（後述の設計情報）に

基づいてほぼ自動的にプログラムのソースコードを生成できるものであり、この機能については、例えば特許文献１に記載されたものを用いる。また、プログラム設計管理システム５０は、共通項目についても、ユーザが作成した仕様書や定義書などに基づいて自動的に生成可能である。

【００３６】

図２に示すように、プログラム設計管理システム５０は、インパクト分析画面１０３、設計リポジトリ入出力部１０５、共通項目アクセス表示出力部１０６、プログラム開発画面１０４、設計リポジトリ入出力部１０５、ソース・設計書出力部１１９、共通項目定義生成１１２、整合コントローラ部１１７を備える。これらの機能ブロックのうちに、共通項目定義１１６、設計リポジトリ入出力部１０５、設計リポジトリ１１８は、プログラム開発サポート装置として機能する。

【００３７】

インパクト分析画面１０３は、インパクト分析時においてプログラム変更箇所１００を入力するための画面である。

【００３８】

設計リポジトリ入出力部１０５は、インパクト分析画面１０３から入力されたプログラム変更箇所１００に関連するプログラムとプログラム中の該当処理を設計リポジトリ１１８から抽出する。

【００３９】

共通項目アクセス表示出力部１０６は、設計リポジトリ入出力部１０５で抽出した結果を基に共通リソースアクセス表１０７もしくはファイルアクセス表１０８もしくはテーブルアクセス表１０９を出力する。

【００４０】

プログラム開発画面１０４は、プログラムコーディング時において、設計情報１０１とコーディング情報１０２と共通項目定義１１６が入力される。

【００４１】

設計リポジトリ入出力部１０５は、プログラム開発画面１０４から入力された設計情報１０１とコーディング情報１０２と共通項目定義を各処理単位に関連づけて設計リポジトリ１１８へ記録する。

【００４２】

ソース・設計書出力部１１９は、設計リポジトリ１１８から設計リポジトリ入出力部１０５によって抽出された設計情報とコーディング情報を基にソースコード１１０と各種設計書１１１を生成する。

【００４３】

共通項目定義生成１１２は、データベーススキーマ情報１１３とファイル仕様書１１４とその他共通リソース定義書１１５を基に共通項目定義１１６を生成する。

【００４４】

整合コントローラ部１１７は、共通項目定義１１６に変更が生じた場合に設計リポジトリ１１８へ変更データを反映する。

【００４５】

本実施の形態にかかるプログラム設計管理システム５０は、特許文献１に記載されたプログラム自動生成システムに対して、主に下記の４つの機能を追加している。

【００４６】

＜第１の機能：共通項目の入力＞

プログラム開発画面１０４に共通項目定義１１６から共通項目を入力させる機能。

【００４７】

＜第２の機能：共通項目の紐付け＞

プログラム開発画面１０４上で入力した設計情報１０１とコーディング情報１０２を設計リポジトリ入出力部１０５によって、関連付けを行って設計リポジトリ１１８に記録する機能。

【００４８】
＜第３の機能：インパクト分析サポート＞
インパクト分析画面１０３上で入力したプログラム変更箇所１００を基に設計リポジトリ入出力部１０５によって設計リポジトリ１１８から関連する部分を抽出して、共通項目アクセス表出力部１０６にて、共通リソースアクセス表１０７、ファイルアクセス表１０８、テーブルアクセス表１０９を作成する機能。
【００４９】
＜第４の機能：データ整合機能＞
共通項目に変更が生じた場合にデータ整合コントローラ部１１７によって、変更箇所を一括して設計リポジトリ１１８に反映する機能。
【００５０】
以下において、この４つの機能に重点をおいて説明する。
＜共通項目の入力＞
「共通項目の入力」に関しては、まず、複数のプログラムで参照される共通項目（テーブルの入出力項目、ファイルの入出力項目、共通関数、クラス継承時の親クラス、共通定数、環境変数）を、他のロジックと独立して一元管理できるよう定義ファイル化し、特定の操作（ドラッグ＆ドロップ）でプログラム開発画面１０４から入力することで、共通リソースを他のロジックとは区別して管理する。
【００５１】
定義ファイルに関しては図２に示す共通項目定義生成１１２にてデータベーススキーマ情報１１３とファイル仕様書１１４とその他共通リソース定義書１１５からある一定の書式で共通項目定義１１６として記録する。
【００５２】
プログラム開発画面１０４では、共通項目定義１１６から共通リソース情報を抽出し、プログラム開発画面上のメモリに展開する。図３は、画面イメージを示す。
【００５３】
図３に示すように、プログラム開発画面１０４は、画面左側のプログラム構造を定義するプログラムフロー図エリア３００と、画面右側の共通項目を表示する共通項目一覧エリア３０１を備える。共通項目一覧エリア３０１上部のプルダウンリスト３０２で対象の共通リソースを選択すると、メモリ上に退避された共通項目定義情報の中から対象項目を抽出してリスト上に展開する。
【００５４】
プログラムのコーディングを行う際には、まず、図３のプログラムフロー図エリア３００でプログラムを処理単位（ノード）に分割し、ノード３０４を用いてツリー形式でプログラム構造を組み立てる。１つのノードが１つの処理を担う。
【００５５】
本実施の形態において、ノードは、入力処理ノード、出力処理ノード、フリー記述ノードの３タイプに分かれ、プログラム構造設計をする際には、これらのうちのどれかを利用して設計する。フリー記述ノードは、入力処理ノードと出力処理ノード以外のすべてのノードを吸収する。
【００５６】
各ノードの中身のコーディングについては、各ノードタイプ別に入力画面を用意する。図４は、それらのイメージを示す画面４００～４０２を示す。これらの画面は図３のプログラムフロー図エリア３００上の各ノードをクリックすると新規ウィンドウとして起動する。
【００５７】
プログラムのコーディング情報は必ずこれら３つの画面からしか入力できず、更に、共通項目に関しては、図３の共通項目一覧エリア３０１上のリストアップされている項目の中から対象項目をドラッグ＆ドロップ操作で図４の４００～４０２のウィンドウ上のコーディングエリアに展開し、手入力しないルールとする。

【００５８】
ノードタイプ毎のドラッグ＆ドロップ操作イメージは図５～図７通りである。このような操作をすることで、画面から入力された情報の中でどの部分が共通項目領域なのかを区別させ、内部で独立させて管理できるようにしている。
【００５９】
＜共通項目の紐付け＞
図４の４００、４０１、４０２の画面から入力されたコーディング情報はノード単位に設計リポジトリに記録する。
【００６０】
本実施の形態では、共通項目がどのよに利用されたのかを記録する専用のエリア（図８の８００、図９の９００、図１０の１０００）が設けられており、共通項目を入力した箇所が特定できる情報を付加した形で設計リポジトリへ記録するようになっている。
【００６１】
例えば、フリー記述ノードであれば、図８のようにまず、共通項目であることを識別するために共通項目の先頭に＄マークを付与し、他の変数と区別させ、共通項目利用履歴エリアに記録する。その際には共通項目ＩＤの後ろにノード名を付与する。こうすることで、この共通項目がプログラムの中のどの処理で呼び出されているかピンポイントで特定することが可能となる。
【００６２】
次に入力処理ノードでは、フリー記述ノードと同様、共通項目の後ろにノード名を付与するが、更に入力処理ノードでは図９の通り、入力時の条件となる属性（結合条件、抽出条件、ソートキー）もあわせて付与する。
【００６３】
次に出力処理ノードも入力処理ノードと同様、出力の条件となる情報や、転記の際の入力元と出力先の識別もできるような情報も図１０のように付加して記録する。従来からＣＲＵＤ表を出力するシステムはあったが、ＣＲＵＤ表はプログラムでアクセスしているテーブルに対するＣＲＥＡＴＥ、ＲＥＡＤ、ＵＰＤＡＴＥ、ＤＥＬＥＴＥのいずれかの操作があるかないかの識別しかできなかった。本発明では、ＣＲＵＤの識別は当然ながら、更に踏み込んで、テーブルへのＳＱＬ操作の中でどの条件（結合条件、抽出条件、ソートキー条件等）で利用されているのかより詳細な個所の特定が可能となるため、より精密なインパクト分析が可能となる。
【００６４】
＜インパクト分析サポート＞
インパクト分析サポート機能は、図２のインパクト分析画面１０３から影響範囲を特定したい共通項目を選択させ、画面上の検索ボタンを押下するとプログラム毎の設計リポジトリの共通項目利用履歴エリア（図１１の１１００が示す範囲）をサーチし、選択した共通項目ＩＤと一致したものを抽出して結果を一覧形式で画面に表示する。
【００６５】
テーブルの入出力項目が選択された場合には、図１２に示す情報（図２のテーブルアクセス表１０９）を出力し、ファイルの入出力項目を選択した場合は、図１３に示す情報（図２のファイルアクセス表１０８）を出力する。また、共通関数と共通定数、環境変数が選択された場合は、図１４のような情報（図２の共通リソースアクセス表１０７）を出力し、親クラスが選択された場合は、図１５のような情報（図２の共通リソースアクセス表１０７）を出力する。
このようにして、ピンポイントで共通項目の影響範囲を正確に瞬時に特定することができる。
【００６６】
＜データ整合＞
共通項目に対して、変更が生じた場合は、図２のデータベーススキーマ情報１１３もしくは、ファイル仕様書１１４もしくは、その他共通リソース定義書１１５の変更箇所を修

正し、共通項目定義生成１１２によって、共通項目定義１１６を再生成後にデータ整合コントローラ部１１７によって、設計リポジトリの該当箇所の変更を一括して行う。

【００６７】

ただし、この機能は共通項目の変更時に限り、追加・削除時はロジックの見直しが必要となるため、インパクト分析画面にて影響範囲を特定してから手作業による修正が必要となる。

【００６８】

本実施の形態にかかるプログラム設計管理システム５０は、プログラム開発システム１をより具現化／高機能化したものであり、プログラム開発システム１により得られる全ての効果を発揮することができる。

【００６９】

さらに、本実施の形態にかかるプログラム設計管理システム５０は、夫々のプログラムについて、ノード毎に共通項目への参照履歴を記録して管理するため、インパクト分析時において、共通項目の分布状況や影響範囲をより精確に把握することができる。

【００７０】

また、複数のプログラムで参照される共通項目をソースコードと独立させて一元管理しているため、共通項目の変更時には共通項目のデータを整合させる機能によって、設計リポジトリへ一括反映させることで、プログラム修正工数と修正漏れを軽減できる。

【００７１】

以上、実施の形態をもとに本発明を説明した。実施の形態は例示であり、本発明の主旨から逸脱しない限り、上述した各実施の形態に対してさまざまな変更、増減、組合せを行ってもよい。これらの変更、増減、組合せが行われた変形例も本発明の範囲にあることは、当業者に理解されるところである。

【００７２】

例えば、ソースコードに限らず、設計情報についても、作成時において、複数のプログラムに夫々対応する設計情報を入力する際にも、これらの設計情報の共通項目を予め作成して記憶しておき、他の項目の入力操作とは異なる特定の操作（例えばドラッグ＆ドロップ操作）で入力させるようにすれば、同様の方式で設計リポジトリに設計情報における共通項目の利用履歴を記録することができると共に、設計情報に対するインパクト分析にも有利な効果を得ることができる。

【００７３】

本発明にかかる技術は、業種を問わず、データベースやファイルシステムを利用したいかなるバッチ処理システムの開発時にも用いることができ、バッチプログラムの本数が多く大規模なシステムになればなるほど効果が顕著である。

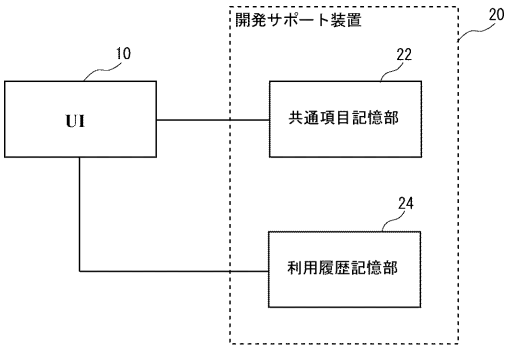
【符号の説明】

【００７４】

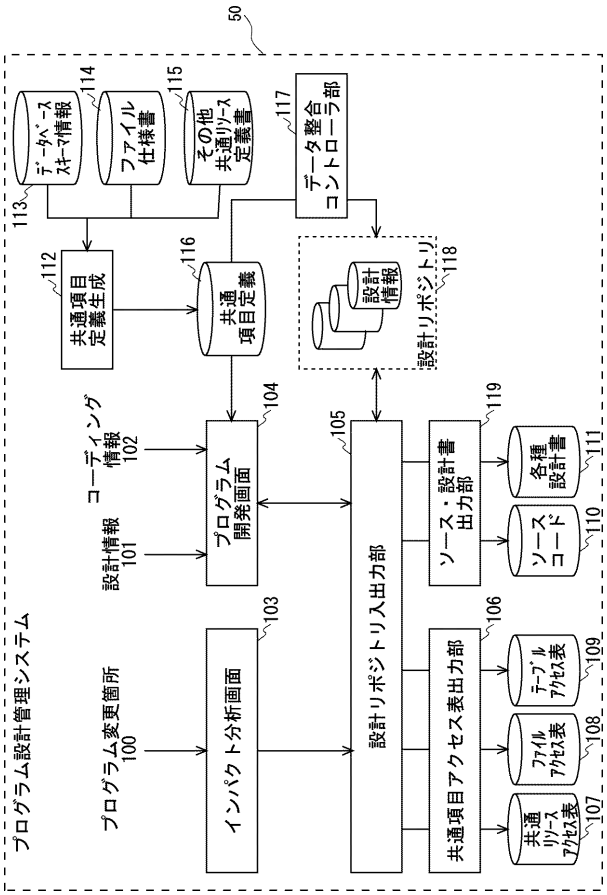
１　プログラム開発システム
１０　ＵＩ（ユーザ・インタフェース）
２０　プログラム開発サポート装置
２２　共通項目記憶部
２４　利用履歴記憶部
５０　プログラム設計管理システム
１００　プログラム変更箇所
１０１　設計情報
１０２　コーディング情報
１０３　インパクト分析画面
１０４　プログラム開発画面
１０５　設計リポジトリ入出力部
１０６　共通項目アクセス表示出力部
１０７　共通リソースアクセス表

１０８　ファイルアクセス表
１０９　テーブルアクセス表
１１０　ソースコード
１１１　設計書
１１２　共通項目定義生成
１１３　データベーススキーマ情報
１１４　ファイル仕様書
１１５　その他共通リソース定義書
１１６　共通項目定義
１１７　整合コントローラ部
１１８　設計リポジトリ
１１９　ソース・設計書出力部
３００　プログラムフロー図エリア
３０１　共通項目一覧エリア
３０２　プルダウンリスト
４００　フリー記述ノードの入力画面
４０１　入力処理ノードの入力画面
４０２　出力処理ノードの入力画面
８００　参照履歴用エリア
９００　参照履歴用用エリア
１０００　参照履歴用エリア
１１００　検索対象

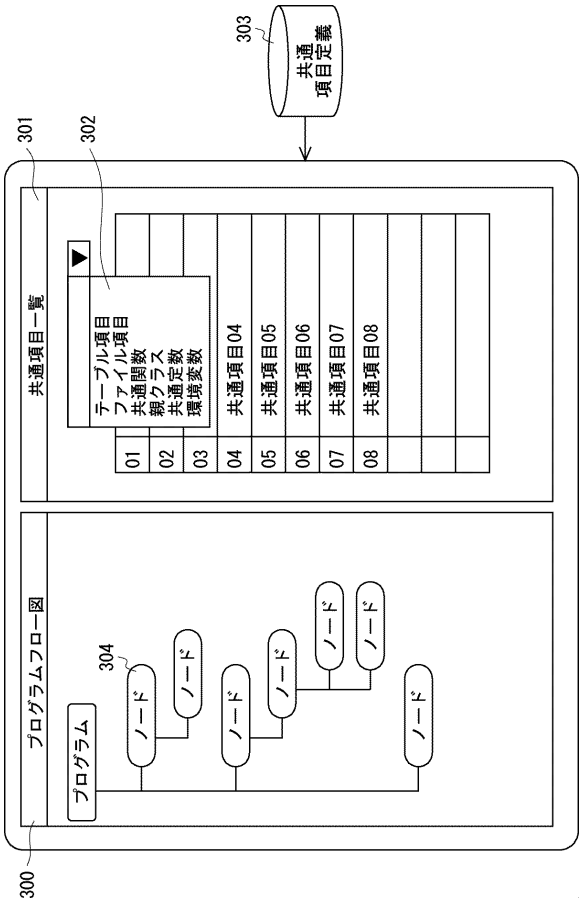
【図１】



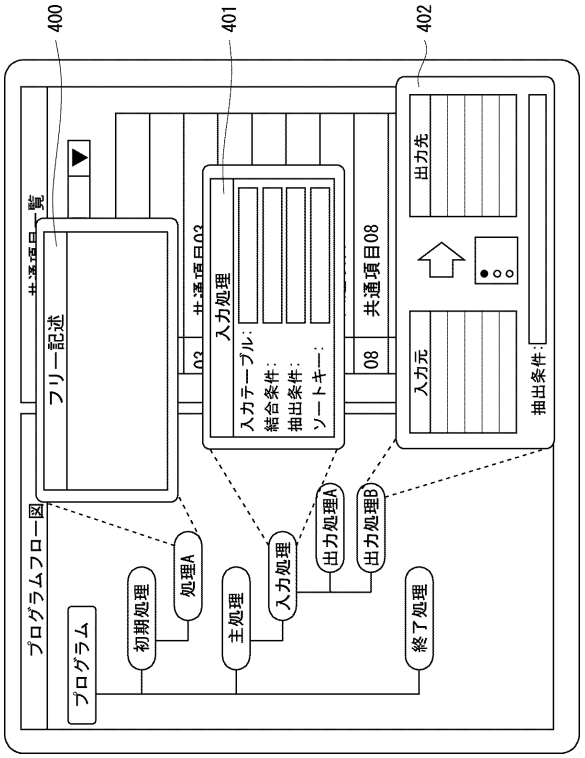
【図２】



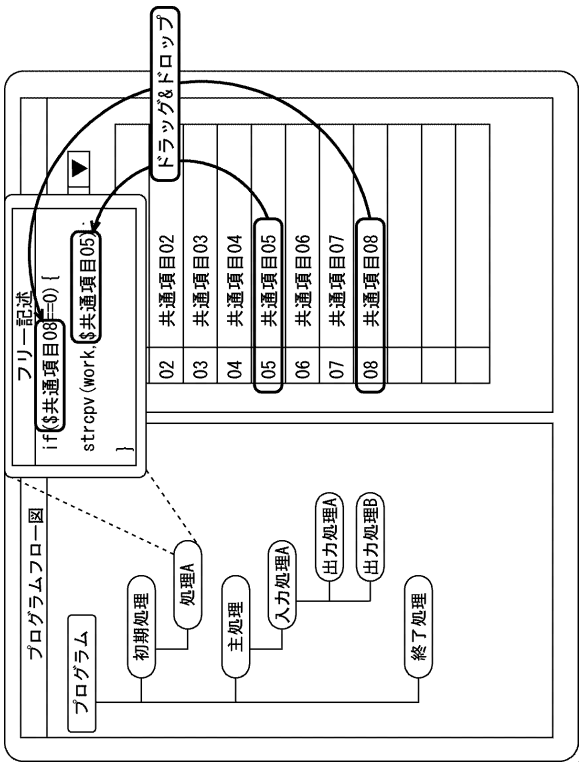
【図３】


300
301
302
303
304
共通項目一覧
テーブル項目
ファイル項目
共通関数
親クラス
共通定数
環境変数
01
02
03
04
05
06
07
08
共通項目04
共通項目05
共通項目06
共通項目07
共通項目08
共通項目定義
プログラムフロー図
プログラム
ノード

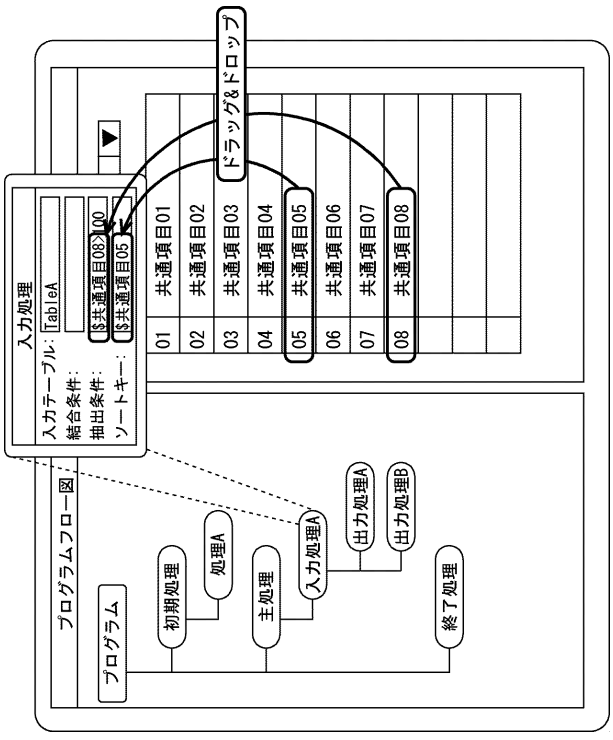
【図４】


400
401
402
プログラムフロー図
フリー記述
入力処理
入力テーブル:
結合条件:
抽出条件:
ソートキー:
出力先
入力元
抽出条件:
共通項目08
プログラム
初期処理
処理A
主処理
入力処理
出力処理A
出力処理B
終了処理

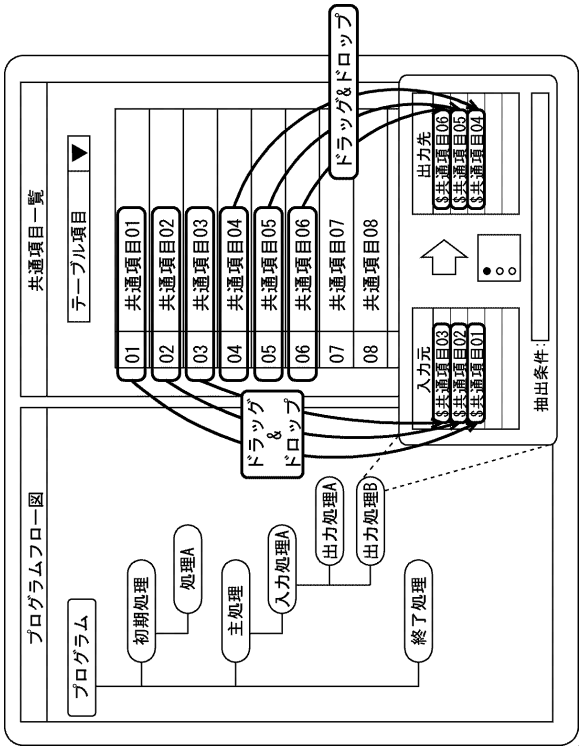
【図５】


ドラッグ&ドロップ
フリー記述
if($共通項目08==0){
strcpy(work,$共通項目05);
}
共通項目02
共通項目03
共通項目04
共通項目05
共通項目06
共通項目07
共通項目08
プログラムフロー図
プログラム
初期処理
処理A
主処理
入力処理A
出力処理A
出力処理B
終了処理

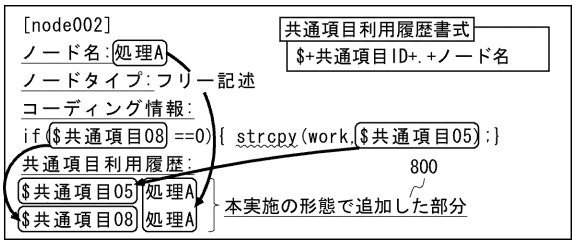
【図６】


ドラッグ&ドロップ
入力処理
入力テーブル:
結合条件:
抽出条件:
ソートキー:
tableA
$共通項目08
00
$共通項目05
共通項目01
共通項目02
共通項目03
共通項目04
共通項目05
共通項目06
共通項目07
共通項目08
プログラムフロー図
プログラム
初期処理
処理A
主処理
入力処理A
出力処理A
出力処理B
終了処理

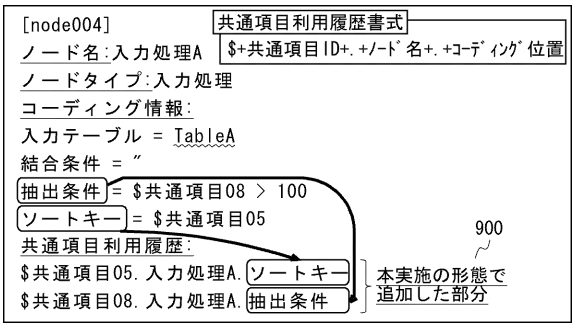
【図７】



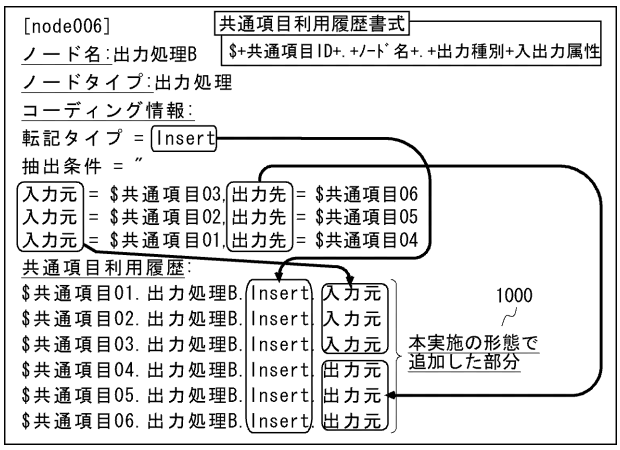
【図８】



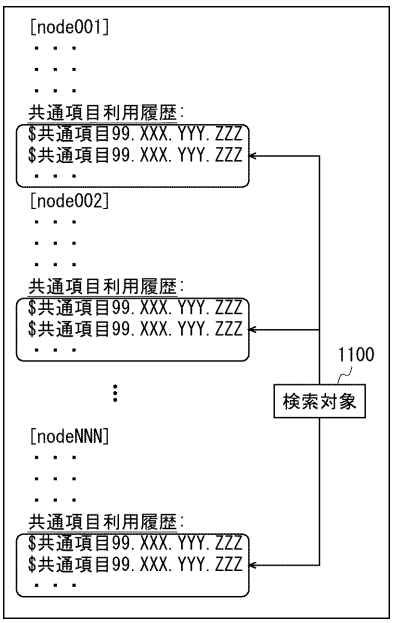
【図９】



【図１０】



【図１１】



【図１２】

	検索結果
01	Program001.共通項目03.入力処理A.抽出キー
02	Program003.共通項目03.入力処理A.結合キー
03	Program003.共通項目03.出力処理A.Insert.入力元
04	Program007.共通項目03.出力処理A.Update.出力先
05	Program009.共通項目03.出力処理C.Delete.出力先
06	Program011.共通項目03.入力処理A.ソートキー

【図１３】

	検索結果
01	Program021.共通項目03.出力処理A.Output.入力元
02	Program023.共通項目03.出力処理A.Append.出力先
03	Program033.共通項目03.出力処理A.Append.入力元
04	Program034.共通項目03.出力処理A.Output.出力先

【図１４】

	検索結果
01	Program002.共通項目03.処理A
02	Program003.共通項目03.処理A
03	Program005.共通項目03.処理C
04	Program040.共通項目03.処理D

【図１５】

	検索結果
01	Program001.共通項目03.子クラスA
02	Program003.共通項目03.子クラスB
03	Program004.共通項目03.子クラスC