(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2010-39949
(P2010-39949A)
(43)公開日　平成22年2月18日(2010.2.18)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 9/44	(2006.01)	G06F	9/06	620J	5B176
G06Q 10/00	(2006.01)	G06F	17/60	164	
		G06F	9/06	620H	
		G06F	9/06	620K	

審査請求 未請求　請求項の数 5　ＯＬ　（全 17 頁）

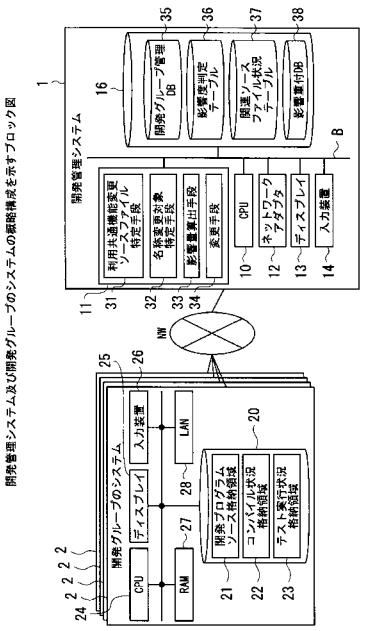
(21)出願番号　特願2008-204662 (P2008-204662)
(22)出願日　平成20年8月7日 (2008.8.7)

(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号
(74)代理人　100090516
弁理士　松倉　秀実
(74)代理人　100113608
弁理士　平川　明
(74)代理人　100105407
弁理士　高田　大輔
(74)代理人　100089244
弁理士　遠山　勉

最終頁に続く

(54)【発明の名称】分散開発管理プログラム

(57)【要約】
【課題】共通機能の改変によって新旧の共通機能が併存するに至った場合に、各開発グループに開発委託された全プログラムモジュールに生じる影響度の総和が最も小さくなるように、新旧何れの共通機能に新名称を付与すべきかを決定する。
【解決手段】何れかの共通機能の変更の提案を受信すると、利用共通機能変更ソースファイル特定手段３１は、当該共通機能を呼び出す各プログラムモジュールについて、当該共通機能の何れの版を利用するかの選択情報を取得する(S103)。影響量算出手段３３は、変更前の版を利用することが選択された全プログラムモジュールに生じる当該共通機能の名称変更に因る影響量，及び、変更後の版を利用することが選択された全プログラムモジュールに生じる当該共通機能の名称変更に因る影響量を夫々算出し(S203-S205)、算出された影響度が小さくなる側のプログラムモジュールについて選択された版を新名称付与対象と決定する(S401)。
【選択図】図１



【特許請求の範囲】
【請求項１】
複数の開発グループが、夫々、共通機能を呼び出して利用できるプログラムモジュールの開発を担当する分散開発環境において、前記共通機能を呼び出すための名称を管理するために、コンピュータを、
何れかの共通機能を利用する各プログラムモジュールについて、当該共通機能の変更前の版を利用するか変更後の版を利用するかの選択情報を取得する選択情報取得手段，
変更前の版を利用することが選択された全プログラムモジュールに生じる当該共通機能の名称変更に因る影響量，及び、変更後の版を利用することが選択された全プログラムモジュールに生じる当該共通機能の名称変更に因る影響量を、夫々算出する影響量算出手段，及び、
前記影響量算出手段によって算出された影響度が小さくなる側のプログラムモジュールについて選択された版を新名称付与対象と決定する変更手段
として機能させる分散開発管理プログラム。
【請求項２】
前記影響量算出手段は、ソースプログラムからオブジェクトプログラムへのコンパイルが完了したプログラムモジュールのみに基づいて、前記影響量を算出する
ことを特徴とする請求項１記載の分散開発管理プログラム。
【請求項３】
前記影響量算出手段は、ソースプログラムからオブジェクトプログラムへのコンパイルが完了した各プログラムモジュールについて、再コンパイルに要する作業量及び既に実行したテストを再実行するのに要する作業量として、前記影響量を算出する
ことを特徴とする請求項２記載の分散開発管理プログラム。
【請求項４】
前記変更手段は、新名称付与対象と決定した版を利用することが選択された各プログラムモジュールのソースプログラム中の呼出文における前記共通機能の名称を、新名称に変更する
ことを特徴とする請求項２記載の分散開発管理プログラム。
【請求項５】
前記変更手段は、新名称付与対象と決定した版の共通機能の名称を新名称に変更するとともに、他方の版が変更後の共通機能であれば、当該変更後の共通機能に元の名称を付与する
ことを特徴とする請求項１記載の分散開発管理プログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、プログラム分散開発環境において、開発対象プログラム製品を構成する複数のプログラムモジュールが共通して呼び出す共通機能の呼出名の管理を、開発管理用のコンピュータに行わしめる分散開発管理プログラムに、関する。
【背景技術】
【０００２】
プログラム製品の大規模化に伴い、その開発は、分散開発環境においてなされることが一般化している。ここに、分散開発環境とは、プログラム製品全体の開発に責任を負う管理者が、所定の仕様を決定し、かかる仕様に従って、当該プログラム製品を構成する各プログラムモジュールの開発を個々の開発グループに委託し、各開発グループが完成したプログラムをまとめることによって開発対象プログラムを完成させる開発形態である。なお、管理者と開発グループとは、同一企業内の製品企画部門と開発グループである場合もあるし、注文者企業と請負人企業である場合もありえる。
【０００３】
かかる分散開発環境において決定される仕様とともに、管理者から各開発グループに対

しては、各プログラムモジュールに共通に用いられるデータ，即ち、共通機能（関数，プログラムモジュール，サブルーチン，副プログラム等）も、提供される。従って、各開発グループは、これら共通機能の内容を認識した上で、自己が開発しているプログラムモジュールにおいて、何れかの共通機能を利用したい場合には、そのソースプログラムに、その共通機能の、名称を含む呼出文（call文）を記述しておけば良い。

【特許文献１】特開2005-158030号公報
【特許文献２】特開2002-82802号公報
【特許文献３】特開平10-254684号公報

【発明の開示】
【発明が解決しようとする課題】

【０００４】
上述したように管理者から各開発グループに対して提供された共通機能は、プログラムモジュールを開発する前段階において管理者が用意したものであるので、各開発グループにおいてプログラムモジュールの開発が具体的に進むにつれて、その内容が開発中のプログラムモジュールの要求を満たし得なくなる場合がある。この場合、当該プログラムモジュールに関しては、当該共通機能を改変することによってその要求を満足させる必要が生じる。

【０００５】
そして、他の開発グループによって開発されている全てのプログラムモジュールにとっても、かかる改変が好都合であるならば、改変後の共通機能によって元の共通機能をオーバーライトすることで、上記仕様については一切変更を行うことなく、開発を継続していくことができる。

【０００６】
これに対して、他の開発グループによって開発されている一部又は全部のプログラムモジュールにとってかかる改変が不都合であるならば、改変後の共通機能によって元の共通機能をオーバーライトさせずに、相互に別個独立のものとして併存させざるを得ない。

【０００７】
この場合、改変後の共通の共通機能については、新たな名称を付与し、当該新たな名称によって呼び出すことになる。もっとも、他の開発グループによって開発されているプログラムモジュールの多くにとってかかる改変が好都合である一方、かかる改変が不都合であるプログラムモジュール数が僅かに留まる場合には、改変後の共通機能に元の名称を付し、改変前の共通機能に新たな名称を付した方が、名称の変更に伴う全開発グループの全体としての影響量（追加作業の手間）を、抑えることができる可能性もある。

【０００８】
従来の分散開発環境においては、新旧何れの共通機能に元の名称を引き継がせるかに依って影響量に如何なる違いが生じるかを算出する手法が知られていなかったので、かかる共通機能の改変が一部の開発グループから提案された場合には、その都度、管理者及び各開発グループの代表者が一同に会して、元の名称を新旧何れの共通機能に引き継がせるべきかについて、検討会議を行わざるを得なかった。その結果、作業の進捗が遅れたり、判断ミスに因って不必要に追加作業が増大したりする問題が生じ得た。

【０００９】
そこで、本案は、共通機能の改変によって新旧の共通機能が併存するに至った場合に、各開発グループに開発委託された全プログラムモジュールに生じる影響度の総和が最も小さくなるように、新旧何れの版の共通機能に新名称を付与すべきかを決定することができる分散開発管理プログラムの提供を、課題とする。

【課題を解決するための手段】

【００１０】
本案が適用された分散開発管理プログラムは、コンピュータに対して、何れかの共通機能を利用する各プログラムモジュールについて、当該共通機能の変更前の版を利用するか変更後の版を利用するかの選択情報を取得させ、変更前の版を利用することが選択された

全プログラムモジュールに生じる当該共通機能の名称変更に因る影響量，及び、変更後の版を利用することが選択された全プログラムモジュールに生じる当該共通機能の名称変更に因る影響量を夫々算出させ、算出された影響度が小さくなる側のプログラムモジュールについて選択された版を新名称付与対象と決定させる。

【００１１】

このように、変更前の版の利用が選択されたプログラムモジュール群，及び、変更後の版の利用が選択されたプログラムモジュール群について、夫々、選択された版の共通機能の名称が元の名称から新名称に変更された場合にそのプログラムモジュール群に属する全プログラムモジュールに夫々生じる影響度が算出される。そして、算出された影響度が小さくなる版が新名称付与対象と決定される。このように、影響度が最も小さくなる版が新名称付与対象と自動的に決定されるので、開発中の全プログラムモジュールに生じる影響度の総和を可及的に小さく抑えることができる。また、各開発グループの担当者が一同に会する必要がないので、それによって生じる作業進捗の遅れをも防止することができる。

【００１２】

本案において、プログラムモジュールに生じる影響とは、例えば、ソースプログラムにおける共通機能の呼出文の記述変更，それに伴う再コンパイル，再コンパイル後の再テストに夫々掛かる手間などである。もっとも、ソースプログラムの記述変更は、これを自動的に行うツールが利用可能であれば、影響度は僅かであるとみなすことも可能である。このような前提があれば、影響度の算出対象からは、コンパイル前のプログラムモジュールを除外することも可能である。また、コンパイル後のオブジェクトプログラムに対するテストは、実行回数が多ければ多い程、再テストに要する手間は大きくなるので、テスト実行回数に対して所定の重みを乗ずることによって影響量を算出することが可能である。また、再コンパイルに要する手間よりもテスト実行１回当たりの手間が大きければ、再コンパイル１回による影響量よりもテスト実行回数に乗じられる重みを大きくすれば良い。

【００１３】

なお、本案において、共通機能とは、関数，プログラムモジュール，サブルーチン，副プログラム等をいう。

【発明の効果】

【００１４】

以上のように構成された本案によると、共通機能の改変によって新旧の共通機能が併存するに至った場合に、変更前の共通機能に新名称を付与した場合と変更後の共通機能に新名称を付与した場合とで各開発グループに開発委託された全プログラムモジュールに生じる影響度の総和を夫々自動的に算出し、その比較結果に基づいて、各開発グループに開発委託された全プログラムモジュールに生じる影響度の総和が最も小さくなるように、新旧何れかの版の共通機能に、新名称を付与すべきかを決定することができる。

【発明を実施するための最良の形態】

【００１５】

以下、図面に基づいて、本案の実施の形態を説明する。

【００１６】

＜システム構成＞

図１は、プログラム分散開発環境を構成するネットワークシステムの概略構成を示すブロック図である。図１に示すように、このネットワークシステムは、ネットワークＮＷを介して相互に接続された一台の開発管理システム１及び複数の開発グループのシステム２から、構成されている。このうち、開発管理システム１は、開発対象プログラム製品全体の開発責任を負う管理者が操作権限を有するコンピュータである。また、各開発グループのシステム２は、当該開発対象プログラム製品の仕様に従って開発者から一部のプログラムモジュールの開発の委託を受けた開発グループに夫々設置され、当該開発グループの担当者によって操作されるコンピュータである。

【００１７】

先ず、ネットワークＮＷとしては、管理者及び各開発グループが同一企業に属している

場合には、企業内ＬＡＮ（Local Area Network）を用いることができ、一部又は全部の開発グループが別企業である場合には、インターネットを用いることができる。

【００１８】

次に、開発グループのシステム２は、ネットワーク接続機能を有する通常構成を有するコンピュータであり、バスＢを通じて相互に接続されたＣＰＵ２４，ディスプレイ２５，入力装置２６，ＲＡＭ２７，ネットワークアダプタ２８及びハードディスク２０から、構成されている。

【００１９】

このうち、ハードディスク２０内には、開発プログラムソース格納領域２１，コンパイル状況格納領域２２，及びテスト実行状況格納領域２３が、確保されている。

【００２０】

このうち、開発プログラムソース格納領域２１には、当該開発グループに開発が委託された各プログラムモジュールについて、その開発中及び完成済みのソースプログラムのファイル（ソースファイル）を格納する領域である。

【００２１】

また、コンパイル状況格納領域２２は、完成したソースプログラムをコンパイルした結果であるオブジェクトプログラムのファイル（オブジェクトファイル）を格納する領域である。

【００２２】

また、テスト実行状況格納領域２３は、オブジェクトプログラムに対して、例えばCOBOLのCOUNT機能のようなテスト用のツールを用いてテストを実行した回数を、格納する領域である。

【００２３】

これら開発プログラムソース格納領域２１，コンパイル状況格納領域２２，及びテスト実行状況格納領域２３の夫々の位置情報（パス），並びに、各領域２１～２３に対して夫々最新のファイル乃至情報を格納しなければならない旨は、管理者が当該開発グループに交付する仕様に、定められている。従って、各開発グループの担当者は、各領域２１～２３に新たに格納すべきファイル又はデータが発生する都度、また、これらが更新される都度、そのＣＰＵ２４が実行するプログラムに従って、その入力装置を通じて最新のファイル又はデータを読み込んで、該当する領域２１～２３に格納する。

【００２４】

なお、これらの格納及び更新の処理は、ＣＰＵ２４が実行するプログラムに従って、自動的になされても良い。また、ＣＰＵ２４が実行するプログラムは、開発管理システム１から受信したリクエストに応じて、各領域２１～２３に格納されているファイル又は情報を、開発管理システム１に応答する。さらに、ＣＰＵ２４が実行するプログラムは、開発管理システム１から受信したリクエストに応じて、開発プログラムソース格納領域に格納されている完成済みのソースプログラムをコンパイルして、その結果としてのオブジェクトファイルをコンパイル状況格納領域２２に格納する。

【００２５】

次に、開発管理システム１は、通常構成を有するコンピュータシステムであり、そのハードウェアは、互いにバスＢによって接続されたＣＰＵ１０，ＲＡＭ１１，ネットワークアダプタ１２，ディスプレイ１３，入力装置１４，及びハードディスク１６から、構成されている。

【００２６】

ＣＰＵ１０は、プログラムを読み込んで、プログラムに従った処理を実行する中央処理装置である。

【００２７】

ＲＡＭ１１は、ＣＰＵ１０によってハードディスク１６から読み出されたプログラムが読み込まれて、ＣＰＵ１０が上記処理を実行する際の作業領域が展開される主記憶装置である。ここでは、ハードディスク１６から読み出された分散開発管理プログラムが読み込

まれることにより、ＲＡＭ１１上には、利用共通機能変更ソースファイル特定手段（選択情報取得手段）３１，名称変更対象特定手段３２，影響量算出手段３３及び変更手段３４の各機能が、展開される。これら利用共通機能変更ソースファイル特定手段３１，名称変更対象特定手段３２，影響量算出手段３３及び変更手段３４の各機能については、図６乃至図１０のフローチャートを参照して、後で詳細に説明する。

【００２８】

ネットワークアダプタ１２は、ネットワークＮＷを終端してネットワークＮＷとの間でデータ通信を司る通信装置である。

【００２９】

ディスプレイ１３は、ＣＰＵ１０による処理結果を表示するための表示装置である。

【００３０】

入力装置１４は、ドライブ装置，キーボード，マウス等のポインティングデバイス等の各種入力装置を含む。

【００３１】

ハードディスク１６は、上述した分散開発管理プログラムを含む各種プログラム及び各種データを格納する記憶装置である。このハードディスク１６に格納されている各種データには、分散開発管理プログラムによってアクセスされる開発グループ管理ＤＢ（データベース）３５，影響度判定テーブル３６，関連ソースファイル状況テーブル３７，影響重付ＤＢ３８が、含まれている。

【００３２】

開発グループ管理ＤＢ３５は、各開発グループ毎に、上述した開発プログラムソース格納領域２１，コンパイル状況格納領域２２，及びテスト実行状況格納領域２３の夫々の位置情報（パス）を記録するためのデータベースである。図２は、この開発グループ管理ＤＢ３５の概略データ構造を示す表である。図２に示されるように、この開発グループ管理ＤＢ３５は、各開発グループ毎に一つのレコードを格納している。そして、各レコードは、対応する開発グループのコード番号を記録するための「開発グループ」フィールド，開発開発プログラムソース格納領域２１の位置情報（パス）を記録するための「開発プログラムソース格納領域」フィールド，コンパイル状況格納領域２２の位置情報（パス）を記録するための「コンパイル状況格納領域」フィールド，テスト実行状況格納領域２３の位置情報（パス）を記録するための「テスト実行状況格納領域」フィールド，及び、担当者のメールアドレスを格納するための「メールアドレス」フィールドから、構成されている。

【００３３】

影響度判定テーブル３６は、影響度算出手段３３によって影響度の比較のために用いられるテーブルである。図３は、この影響度判定テーブル３６のデータ構造を示す表である。この図３に示すように、影響度判定テーブル３６は、変更後の版の共通機能を呼び出すプログラムモジュールについて算出された影響度の総和（選択＝新），及び、変更前の版の共通機能を呼び出すプログラムモジュールについて算出された影響度の総和（選択＝旧）を夫々記録するための「影響度合」フィールドを、有している。

【００３４】

関連ソースファイル状況テーブル３７は、各開発グループに開発を委託した各プログラムモジュールについて、その開発の進捗状況を記録するためのテーブルである。図４は、この関連ソースファイル状況テーブル３７の概略データ構造を示す表である。図４に示されるように、この関連ソースファイル状況テーブル３７は、各プログラムモジュール毎に一つのレコードを格納している。そして、各レコードは、対応するプログラムモジュールのソースファイルの名称を記録するための「ソースファイル名」フィールド，開発グループのコード番号を記録するための「開発グループ」フィールド，対応するプログラムモジュールのソースプログラムが変更後の版の共通機能を呼び出すか（新）変更前の版の共通機能を呼び出すか（旧）を記録するための「選択」フィールド，コンパイルが未だなされていないか（未）既に完了しているか（済）を記録するための「コンパイル」フィールド

，テスト実行状況格納領域２３に記録されたテスト実行回数を転載するための「テスト実行回数」フィールド，対応するプログラムモジュールについて算出された「影響度」を記録するための「影響度」フィールドから、構成されている。なお、各プログラムモジュールを各開発グループに開発委託した段階では、「ソースファイル名」フィールド及び「開発グループ」フィールドに値が書き込まれるが、他のフィールドは空欄となっている。なお、「選択」フィールドの値が、共通機能の変更前の版を利用するか変更後の版を利用するかの選択情報に、相当する。

【００３５】

影響重付ＤＢ３８は、影響度算出手段３３による後述する影響度の判定のために用いる重み値を、リコンパイル用及び再テスト用に夫々記憶しているデータベースである。図５は、この影響重付ＤＢ３８のデータ構造を示す表である。この図５に示すように、影響重付ＤＢ３８は、リコンパイル用の重み値を記録するための「リコンパイル」フィールド及び再テスト用の重み値を記録するための「再テスト」フィールドを、有している。

【００３６】

＜データ処理の流れ＞

以下、上述したように構成されたネットワークシステムを構成する開発管理システム１において、ハードディスク１６に格納されている分散開発プログラムを読み込んだＣＰＵ１０が実現する処理（即ち、ＣＰＵ１０が実現する各機能３１～３４の内容）を、図６乃至図１０のフローチャートに基づいて説明する。

【００３７】

図６のフローチャートに示す処理は、管理者が開発対象プログラム製品について仕様を決定し、各開発グループに当該仕様を交付するとともに、開発グループ管理ＤＢ３５に必要なデータを格納することにより、スタートする。

【００３８】

そして、スタート後最初のＳ００１では、ＣＰＵ１０は、各開発グループからの仕様変更提案通知を待つ。この仕様変更提案通知は、管理者から各開発グループに交付された仕様中において定められたフォーマットに則って、各開発グループのシステム２から開発管理システム１に対して通知される、対象共通機能の名称，当該共通機能の変更を求める旨及び変更後の版の共通機能の内容を含むメッセージである。

【００３９】

次のＳ００２では、ＣＰＵ１０は、何れかの開発グループから仕様変更提案通知を受信したかどうかをチェックする。そして、未だ何れの開発グループからも仕様変更提案通知を受信していない場合には、ＣＰＵ１０は、処理をＳ００１へ戻す。これに対して、何れかの開発グループから仕様変更提案通知を受信すると、ＣＰＵ１０は、処理をＳ００３へ進める。

【００４０】

Ｓ００３では、ＣＰＵ１０は、Ｓ００２にて受信した仕様変更提案通知から、対象共通機能の名称及び変更後の内容を抽出し、ディスプレイ１３上に表示する。

【００４１】

次のＳ００４では、ＣＰＵ１０は、入力装置１４を通じて入力される操作者からの指示に従って、当該仕様変更提案の内容，即ち、Ｓ００３にて抽出及び表示された変更後の共通仕様の採用可否についてチェックする。そして、操作者が採用しない旨の指示を入力した場合には、ＣＰＵ１０は、そのままこの分散開発プログラムによる処理を終了する。

【００４２】

これに対して、操作者が採用する旨の指示を入力した場合には、ＣＰＵ１０は、Ｓ００５において、変更提案された変更後の版の共通機能を利用するプログラムモジュールのソースファイルを特定する処理を実行する。

【００４３】

図７は、Ｓ００５にて実行される利用共通機能変更ソースファイル特定処理サブルーチンを示すフローチャートである。このサブルーチンに入って最初のＳ１０１では、ＣＰＵ

１０（利用共通機能変更ソースファイル特定手段３１）は、開発グループ管理ＤＢ３５から、各開発グループについての「開発プログラムソース格納領域」フィールドの値，即ち、その位置情報（パス）を読み取る。

【００４４】

次のＳ１０２では、ＣＰＵ１０（利用共通機能変更ソースファイル特定手段３１）は、Ｓ１０１にて取得した位置情報（パス）に基づいて、全開発グループのシステム２における開発プログラムソース格納領域２１を検索し、変更提案されている共通機能を利用しているプログラムモジュールのソースファイルを取得する。

【００４５】

次のＳ１０３では、ＣＰＵ１０（利用共通機能変更ソースファイル特定手段３１）は、Ｓ１０２でのソースファイルの取得元となった開発グループのコード番号，各ソースファイルの名称，及び、必要であればその内容を含む変更ソースファイル選択画面をディスプレイ１３上に表示し、各開発グループにおいて各ソースファイルに関して変更前の版の共通機能（新）を使用するか変更後の版の共通機能（旧）を使用するかの選択を、受け付ける。即ち、図１１に示すように、この変更ソースファイル選択画面には、各開発グループのコード番号４０及び各ソースファイルのソースファイル名４１の組合せ毎に、新又は旧を選択するためのオプションボタン４２及びソースプログラムの詳細内容を表示させるための「参照」ボタン４３が、一覧表示されている。そして、開発管理システム１の操作者は、各開発グループの担当者から事情聴取した新旧の選択結果に従って、各組合せについて新旧何れかのオプションボタン４２をチェックするのである。なお、この選択画面を、各開発グループのシステム２のディスプレイ２５に表示させ、この選択画面上で各開発グループの担当者によって直接入力された新旧選択結果を、ネットワークＮＷを通じて開発管理システム１が受信するように、構成されても良い。何れにしても、全ての開発グループのコード番号４０及びソースファイル名４１の組合せについて新旧選択結果を受け付けると、ＣＰＵ１０（利用共通機能変更ソースファイル特定手段３１）は、処理をＳ１０４へ進める。

【００４６】

次のＳ１０４では、ＣＰＵ１０（利用共通機能変更ソースファイル特定手段３１）は、Ｓ１０３にて選択入力された内容を基に、図４（ａ）に例示するように、関連ソースファイル情況テーブル３７を生成する。この時点では、同テーブル中「コンパイル」フィールド，「テスト実行回数」フィールド及び「影響度」フィールドは未設定となっている。Ｓ１０４を完了すると、ＣＰＵ１０（利用共通機能変更ソースファイル特定手段３１）は、図７のサブルーチンを修了して、図６におけるＳ００５からＳ００６へ進む。

【００４７】

Ｓ００６では、ＣＰＵ１０（名称変更対象特定手段３２）は、各共通機能について、変更後のものに元の名称を引き継がせるか、変更前のものに元の名称を引き継がせるかを特定する処理を実行する。図８は、Ｓ００６にて実行される名称変更対象特定処理サブルーチンを示すフローチャートである。

【００４８】

このサブルーチンに入って最初のＳ２０１では、ＣＰＵ１０（名称変更対象特定手段３２）は、開発グループ管理ＤＢ３５から、各開発グループについての「コンパイル状況格納領域」フィールドの値，即ち、その位置情報（パス）を読み取る。そして、読み取った位置情報（パス）に基づいて、各開発グループのシステム２におけるコンパイル状況格納領域２２を検索し、Ｓ００５にて特定した各プログラムモジュールについてのオブジェクトファイルがあるか否かに基づいて、各プログラムモジュールのソースプログラムがコンパイル済みが否かを判定する。そして、判定結果を、関連ソースファイル状況テーブル３７における各プログラムモジュールについての「コンパイル」フィールドに設定する（図４(ｂ)参照）。

【００４９】

次のＳ２０２では、ＣＰＵ１０（名称変更対象特定手段３２）は、開発グループ管理Ｄ

Ｂ３５から、各開発グループについての「テスト実行状況格納領域」フィールドの値，即ち、その位置情報（パス）を読み取る。そして、位置情報（パス）に基づいて、各開発グループのシステム２におけるテスト実行状況格納領域２３を、関連ソースファイル状況テーブル３７において「コンパイル」フィールドに「済」が設定された各ソースファイル名をキーとして検索し、オブジェクトプログラムに対するテスト実行回数を取得する。そして、取得した各オブジェクトプログラムに対するテスト実行回数を、関連ソースファイル状況テーブルの「テスト実行回数」フィールドに設定する（図４(ｃ)参照）。

【００５０】

次のＳ２０３では、ＣＰＵ１０（名称変更対象特定手段３２）は、各開発グループに開発委託した各プログラムモジュールについて、変更提案された共通機能について、変更前の版の名称を新名称に変更した場合の影響度と変更後の版の名称を新名称に変更した場合の影響度とを夫々算出するための、影響度算出処理を実行する。

【００５１】

図９は、Ｓ２０３にて実行される影響度算出サブルーチンを示すフローチャートである。このサブルーチンに入って最初のＳ３０１では、ＣＰＵ１０（影響度算出手段３３）は、関連ソースファイル状況テーブル３７における処理対象レコードを特定するための変数ｎを初期値“１”にリセットする。

【００５２】

続いて、ＣＰＵ１０（影響度算出手段３３）は、関連ソースファイル状況テーブル３７の各レコード毎に影響度算出を行うために、Ｓ３０２乃至Ｓ３０９のループ処理を実行する。このループ処理に入って最初のＳ３０２では、ＣＰＵ１０（影響度算出手段３３）は、関連ソースファイル状況テーブル３７からｎ番目のレコードを読み取る。

【００５３】

次のＳ３０３では、ＣＰＵ１０（影響度算出手段３３）は、Ｓ３０２にて読み取ったｎ番目のレコードから、「コンパイル」フィールドの値を取得する。

【００５４】

次のＳ３０４では、ＣＰＵ１０（影響度算出手段３３）は、Ｓ３０３にて読み取った値が“済”となっているか否かをチェックする。そして、値が“未”であるか未設定であると、当該レコードについては未コンパイル故に名称変更に因る影響は僅かであると見なせるので、ＣＰＵ１０（影響度算出手段３３）は、処理をそのままＳ３０８へ進める。これに対して、値が“済”であると、当該レコードについてはコンパイル済み故に名称変更に因る影響が大きいと見なせるので、ＣＰＵ１０（影響度算出手段３３）は、処理をＳ３０５へ進める。

【００５５】

Ｓ３０５では、ＣＰＵ１０（影響度算出手段３３）は、影響度重み付けＤＢ３８から、「コンパイル」に対応した値を読み取り、読み取った値を、関連ソースファイル状況テーブル３７のｎ番目のレコードの「影響度」フィールドに設定する。

【００５６】

次のＳ３０６では、ＣＰＵ１０（影響度算出手段３３）は、関連ソースファイル状況テーブル３７のｎ番目のレコードから「テスト実行回数」フィールドの値を取得する。

【００５７】

次のＳ３０７では、ＣＰＵ１０（影響度算出手段３３）は、影響重付ＤＢ３８から、「再テスト」に対応した値を読み取り、読み取った値をＳ３０６にて取得した「テスト実行回数」フィールドの値に乗じ、算出された積を、関連ソースファイル状況テーブル３７のｎ番目のレコードの「影響度」フィールドの値に、加算する。Ｓ３０７を完了すると、ＣＰＵ１０（影響度算出手段３３）は、処理をＳ３０８へ進める。

【００５８】

Ｓ３０８では、ＣＰＵ１０（影響度算出手段３３）は、変数ｎを一つインクリメントする。

【００５９】

次のＳ３０９では、ＣＰＵ１０（影響度算出手段３３）は、関連ソースファイル状況テーブル３７の全レコードについてＳ３０２以下の処理を完了したかどうかチェックする。そして、未だ全レコードについて処理を完了していない場合には、ＣＰＵ１０（影響度算出手段３３）は、処理をＳ３０２へ戻す。

【００６０】

これに対して、全レコードについて処理を完了した場合には、ＣＰＵ１０（影響度算出手段３３）は、この影響度算出処理サブルーチンを完了して、処理を図８のルーチンに戻す。

【００６１】

処理が戻された図８のルーチンでは、ＣＰＵ１０（名称変更対象特定手段３２）は、Ｓ２０３の次のＳ２０４において、関連ソースファイル状況テーブル３７の全レコードのうち「選択」フィールドの値が“旧”であるものの「影響度」フィールドの値の総和を算出し、算出した総和を、影響度判定テーブル３６における「選択」フィールドの値が“旧”であるレコードの「影響度合」フィールドに、設定する。

【００６２】

次のＳ２０５では、ＣＰＵ１０（名称変更対象特定手段３２）は、関連ソースファイル状況テーブル３７の全レコードのうち「選択」フィールドの値が“新”であるものの「影響度」フィールドの値の総和を算出し、算出した総和を、影響度判定テーブル３６における「選択」フィールドの値が“新”であるレコードの「影響度合」フィールドに、設定する。Ｓ２０５を完了すると、ＣＰＵ１０（名称変更対象特定手段３２）は、この名称変更対象特定処理サブルーチンを終了して、処理を図６のメインルーチンに戻す。

【００６３】

処理が戻された図６メインルーチンでは、ＣＰＵ１０は、Ｓ００６の次のＳ００７において、変更処理を実行する。図１０は、Ｓ００７にて実行される変更処理サブルーチンを示すフローチャートである。このサブルーチンに入って最初のＳ４０１では、ＣＰＵ１０（変更手段３４）は、影響度判定テーブル３６の二つのレコードの「影響度合」フィールドの値同士を比較し、小さい方の「選択」フィールドの値を、変数“新名称対象（新名称付与対象）”に設定する。

【００６４】

次のＳ４０２では、ＣＰＵ１０（変更手段３４）は、新名称の入力を促すメッセージをディスプレイ１３上に表示し、操作者が入力装置１４を通じて何らかの新名称を入力するを待つ。そして、操作者が新名称を入力すると、これを受け付ける。そして、当該開発管理システム１に保存している更新前の版の変更対象共通機能及びＳ００３にて仕様変更提案通知から抽出した更新後の版の変更対象共通機能のうち、変数“新名称対象”に対応するものの名称を、受け付けた新名称に変更し、他方（更新後の変更対象共通機能に限る）の名称を、当該変更対象共通機能の元の名称に変更する。

【００６５】

次のＳ４０３では、ＣＰＵ１０（変更手段３４）は、関連ソースファイル状況テーブル３７中の「選択」フィールドの値が変数“新対象名称”に一致している全レコードを特定する。

【００６６】

次のＳ４０４では、ＣＰＵ１０（変更手段３４）は、Ｓ４０３にて特定した個々のレコード毎に、そのレコードの「開発グループ」の値に対応する「開発プログラムソース格納領域」フィールドの値，即ち、位置情報（パス）を開発グループ管理ＤＢ３５から読み出し、当該読み出した値，即ち、位置情報（パス）が示す開発プログラムソース格納領域２１にアクセスして、そのレコードの「ソースファイル名」フィールドの値が示すソースファイルを読み出し、読み出したソースファイル中のソースプログラムに記述されている変更対象共通機能の名称を、Ｓ４０２にて受け付けた新名称に変更する。かかる処理をＳ４０３にて特定した全レコードに対して実行すると、ＣＰＵ１０（変更手段３４）は、処理をＳ４０５へ進める。

【００６７】
Ｓ４０５では、ＣＰＵ１０（変更手段３４）は、Ｓ４０３にて特定した全レコードのうち、「コンパイル」フィールドの値が“済”であり且つ「テスト実行回数」フィールドの値が零以上であるもの全てを選別する。そして、選別した個々のレコード毎に、そのレコードの「開発グループ」の値　に対応する「メールアドレス」フィールドの値（メールアドレス）を開発グループ管理ＤＢ３５から読み出し、読み出したメールアドレス宛に、そのレコードの「ソースファイル名」フィールドの値が示すソースファイルに対するリコンパイルの指示及び再テストの指示を含む電子メールを、通知する。この場合、ＣＰＵ１０（変更手段３４）は、当該開発グループのシステム２に対して、直接、当該ソースファイルに対するリコンパイル及び再テストを命じるコマンドを送信しても良い。かかる処理を選別した全レコードに対して実行すると、ＣＰＵ１０（変更手段３４）は、処理をＳ４０６へ進める。
【００６８】
Ｓ４０６では、ＣＰＵ１０（変更手段３４）は、Ｓ４０３にて特定した全レコードのうち、「コンパイル」フィールドの値が“済”であり且つ「テスト実行回数」フィールドの値が未設定又は零であるもの全てを選別する。そして、選別した個々のレコード毎に、そのレコードの「開発グループ」の値　に対する「メールアドレス」フィールドの値（メールアドレス）を読み出し、読み出したメールアドレス宛に、そのレコードの「ソースファイル名」フィールドの値が示すソースファイルに対するリコンパイルの指示を含む電子メールを、送信する。この場合、ＣＰＵ１０（変更手段３４）は、当該開発グループのシステム２に対して、直接、当該ソースファイルに対するリコンパイルを命じるコマンドを送信しても良い。かかる処理を選別した全レコードに対して実行すると、ＣＰＵ１０（変更手段３４）は、処理をＳ４０６へ進める。なお、Ｓ４０６では、Ｓ４０５と異なり開発グループに対して再テストの指示がなされないが、これは、Ｓ４０６の連絡対象となる開発グループでは、元々テストの実行が予定されているからである。Ｓ４０６を完了すると、ＣＰＵ１０（変更手段３４）は、この変更処理サブルーチンを終了して、処理を図６のメインルーチンに戻し、このメインルーチンを終了する。
【００６９】
＜実施形態の作用＞
以上の処理が全て完了すると、一部の開発グループから何れかの共通機能についての仕様変更提案通知があった場合、その提案が採用される限り（Ｓ００４：ＹＥＳ）、変更前の版の当該共通機能及び変更後の版の当該共通機能の何れか一方には元の名称が承継されるとともに他方には新名称が与えられるので（Ｓ４０４）、当該共通機能の新旧二つの版が併存できるようになり、各開発グループでは、自己が開発委託されたプログラムモジュールのソースプログラムを作成するに当たり、これら新旧両版の当該共通機能を、任意に選択して呼び出すように記述することができる。
【００７０】
但し、各プログラムモジュールについて開発がある時点まで進んだ段階で、呼び出し対象を変更前の版の共通機能から変更後の版の共通機能に変更することになった場合、変更後の版の共通機能に新名称が付与されたのならば、ソースプログラムにおける当該共通機能の呼出文の記述が、書き換えられなければならない。同様に、呼び出し対象を変更前の版の共通機能ままとした場合でも、変更前の版の共通機能に新名称が付与されたのならば、ソースプログラムにおける当該共通機能の呼出文の記述を、書き換えられなければならない。特に、ソースプログラムを一旦完成させてコンパイルまで実行した段階で、かかる共通機能の仕様変更があった場合には、ソースプログラムの記述変更、ソースプログラムの再コンパイル，再コンパイルされたオブジェクトプログラムに対するテスト実行を行わなければならないので、全体の作業量に対する影響が大きい。
【００７１】
そこで、本実施形態では、かかる影響を最小限に抑えるべく、各開発グループにおける各プログラムモジュールに関する新旧何れの版の共通機能の選択状況に応じて（Ｓ００５

，Ｓ１０３）、変更前の版の共通機能に新名称が付与したならば各開発グループによって開発されている全プログラムモジュールに生ずべき影響度の総和を算出するとともに（Ｓ００６，Ｓ２０３，Ｓ２０４）、変更後の版の共通機能に新名称が付与したならば各開発グループによって開発されている全プログラムモジュールに生ずべき影響度の総和を算出し（Ｓ００６，Ｓ２０３，Ｓ２０５）、算出された影響度の総和が小さくなる版の共通機能に、新名称を付与する（Ｓ００７，Ｓ４０１，Ｓ４０２）。よって、共通機能の仕様変更があり、且つ、変更後の版　の当該共通機能が提案元以外の開発グループによって採用されたとしても、全体に対する影響度が最も小さくなるように、当該共通機能の新名称付与対象の版が決定されるので、変更に伴う影響を最小限に止めることができる。

【００７２】

なお、本実施形態では、変更対象となった共通機能の新旧何れの版に新名称を付与するか（元の名称を承継させるか）を決定すると（Ｓ４０１，Ｓ４０２）、新名称が付与されることとなった版の当該共通機能を採用することが申請されていた（Ｓ１０３）プログラムモジュールのソースプログラムにおける当該共通機能の呼出文の記述が、新名称を対象とするように自動的に書き換えられる（Ｓ４０４）。従って、未だコンパイル前のソースプログラムに対する呼出対象共通機能の名称変更の影響は小さいと考えられるので、上述した影響度算出の対象から、未コンパイルのプログラムモジュールは除外されている。

【００７３】

さらに、新名称が付与されることとなった版の当該共通機能を採用することを申請していた（Ｓ１０３）開発グループに対しては、リコンパイルの指示がなされる（Ｓ４０５）。

【００７４】

さらに、かかるプログラムモジュールのオブジェクトプログラムに対してテストが実行されている場合には、今まで実施したのと同じ回数のテストの実行の指示がなされる（Ｓ４０５）。このように、呼出文の記述を変更したプログラムモジュールについては、実施済みのテストと同じ数及び内容のテストを、再コンパイル後に再実行しなければならない。従って、テスト実行回数が多いほど、呼出対象共通機能の名称変更の影響が大きくなる。よって、上述した影響度算出においては、テスト実行回数に対して重みの値を乗じることによって、影響度を算出しているのである（Ｓ３０７）。しかも、この再テストは、再コンパイルよりも作業量が大きくなるので、リコンパイル１回に対する重みよりも、再テストに対する重みをお大きくすることにより、現実的な影響度を算出できるのである。

（付記１）

複数の開発グループが、夫々、共通機能を呼び出して利用できるプログラムモジュールの開発を担当する分散開発環境において、前記共通機能を呼び出すための名称を管理するために、コンピュータを、

何れかの共通機能を利用する各プログラムモジュールについて、当該共通機能の変更前の版を利用するか変更後の版を利用するかの選択情報を取得する選択情報取得手段，

変更前の版を利用することが選択された全プログラムモジュールに生じる当該共通機能の名称変更に因る影響量，及び、変更後の版を利用することが選択された全プログラムモジュールに生じる当該共通機能の名称変更に因る影響量を、夫々算出する影響量算出手段，及び、

前記影響量算出手段によって算出された影響度が小さくなる側のプログラムモジュールについて選択された版を新名称付与対象と決定する変更手段

として機能させる分散開発管理プログラム。

（付記２）

前記影響量算出手段は、ソースプログラムからオブジェクトプログラムへのコンパイルが完了したプログラムモジュールのみに基づいて、前記影響量を算出する

ことを特徴とする付記１記載の分散開発管理プログラム。

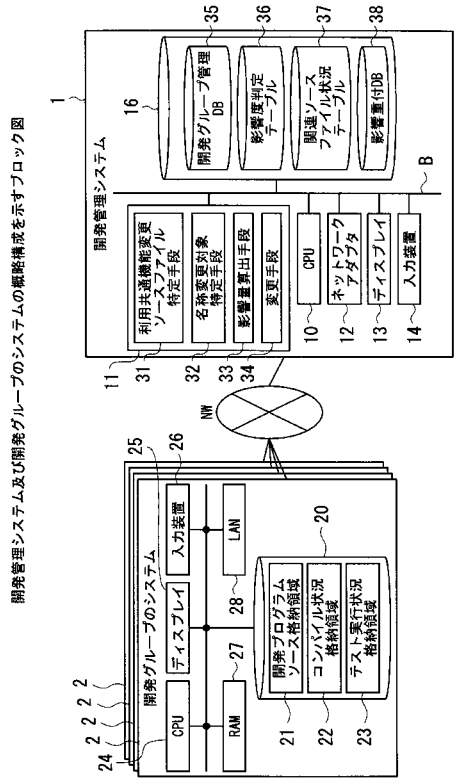
（付記３）

前記影響量算出手段は、ソースプログラムからオブジェクトプログラムへのコンパイル

が完了した各プログラムモジュールについて、再コンパイルに要する作業量及び既に実行したテストを再実行するのに要する作業量として、前記影響量を算出する
ことを特徴とする付記２記載の分散開発管理プログラム。
（付記４）
前記影響量算出手段は、前記再コンパイルに要する作業量よりも、テスト１回当たりに要する作業量が大きくなるように重み付けをして、前記影響量を算出する
ことを特徴とする付記３記載の分散開発管理プログラム。
（付記５）
前記影響量算出手段は、各プログラムモジュールについて、夫々、コンパイル実行の有無及びテスト実行回数を記録しているデータベースにアクセスして、各プログラムモジュールについてのコンパイル実行の有無及びテスト実行回数を読み出して、前記影響量を算出する
ことを特徴とする付記３記載の分散開発管理プログラム。
（付記６）
前記変更手段は、新名称付与対象と決定した版を利用することが選択された各プログラムモジュールのソースプログラム中の呼出文における前記共通機能の名称を、新名称に変更する
ことを特徴とする付記２記載の分散開発管理プログラム。
（付記７）
前記変更手段は、新名称付与対象と決定した版の共通機能の名称を新名称に変更するとともに、他方の版が変更後の共通機能であれば、当該変更後の共通機能に元の名称を付与する
ことを特徴とする付記１記載の分散開発管理プログラム。
【図面の簡単な説明】
【００７５】
【図１】開発管理システム及び開発グループのシステムの概略構成を示すブロック図
【図２】開発グループ管理ＤＢのデータ構造を示す表
【図３】影響度判定テーブルのデータ構造を示す表
【図４】関連ソースファイル状況テーブルのデータ構造を示す表
【図５】影響重付ＤＢのデータ構造を示す表
【図６】開発管理プログラムによる処理を示すフローチャート
【図７】図６のＳ００５にて実行される利用共通機能変更ソースファイル特定処理サブルーチンの内容を示すフローチャート
【図８】図６のＳ００６にて実行される名称変更対象特定処理サブルーチンの内容を示すフローチャート
【図９】図８のＳ２０３にて実行される影響度算出処理サブルーチンの内容を示すフローチャート
【図１０】図６のＳ００７にて実行される変更処理サブルーチンの内容を示すフローチャート
【図１１】変更ソースファイル選択画面を示す図
【符号の説明】
【００７６】
１　開発管理システム
２　開発グループのシステム
１０　ＣＰＵ
１１　ＲＡＭ
１４　入力装置
１６　ハードディスク
３１　利用共通機能変更ソースファイル特定手段
３２　名称変更対象特定手段

３３　影響量算出手段
３４　変更手段
３５　開発グループ管理ＤＢ
３６　影響度判定テーブル
３７　関連ソースファイル状況テーブル
３８　影響重付ＤＢ

【図１】



【図２】

開発グループ管理DBのデータ構造を示す表

開発グループ	開発プログラムソース格納領域	コンパイル状況格納領域	テスト実行状況格納領域	メールアドレス
DEV001	ftp://dev001.com/src/	ftp://dev001.com/cmp/	ftp://dev001.com/test/	dev001@○○.com
DEV002	ftp://dev002.com/src/	ftp://dev002.com/cmp/	ftp://dev002.com/test/	dev002@○○.com
DEV003	ftp://dev003.com/src/	ftp://dev003.com/cmp/	ftp://dev003.com/test/	dev003@○○.com
DEV004	ftp://dev004.com/src/	ftp://dev004.com/cmp/	ftp://dev004.com/test/	dev004@○○.com
DEV005	ftp://dev005.com/src/	ftp://dev005.com/cmp/	ftp://dev005.com/test/	dev005@○○.com
・	・	・	・	・

【図３】

影響度判定テーブルのデータ構造を示す表

選択	影響度合
新	
旧	

【図４】

関連ソースファイル状況テーブルのデータ構造を示す表

（ａ）

ソースファイル名	開発グループ	選択	コンパイル	テスト実行回数	影響度
jikoufurikomi01	DEV01	新	-	-	
jikoufurikomi02	DEV01	旧	-	-	
takoufurikomi01	DEV03	新	-	-	
takoufurikomi02	DEV03	旧	-	-	
zandakasyoukai01	DEV04	旧	-	-	
hikiotosi01	DEV05	新	-	-	

（ｂ）

ソースファイル名	開発グループ	選択	コンパイル	テスト実行回数	影響度
jikoufurikomi01	DEV01	新	済	-	
jikoufurikomi02	DEV01	旧	未	-	
takoufurikomi01	DEV03	新	未	-	
takoufurikomi02	DEV03	旧	済	-	
zandakasyoukai01	DEV04	旧	済	-	
hikiotosi01	DEV05	新	済	-	

（ｃ）

ソースファイル名	開発グループ	選択	コンパイル	テスト実行回数	影響度
jikoufurikomi01	DEV01	新	済	24	
jikoufurikomi02	DEV01	旧	未	-	
takoufurikomi01	DEV03	新	未	-	
takoufurikomi02	DEV03	旧	済	12	
zandakasyoukai01	DEV04	旧	済	15	
hikiotosi01	DEV05	新	済	0	

（ｄ）

ソースファイル名	開発グループ	選択	コンパイル	テスト実行回数	影響度
jikoufurikomi01	DEV01	新	済	24	7201
jikoufurikomi02	DEV01	旧	未	-	0
takoufurikomi01	DEV03	新	未	-	0
takoufurikomi02	DEV03	旧	済	12	3601
zandakasyoukai01	DEV04	旧	済	15	4501
hikiotosi01	DEV05	新	済	0	1

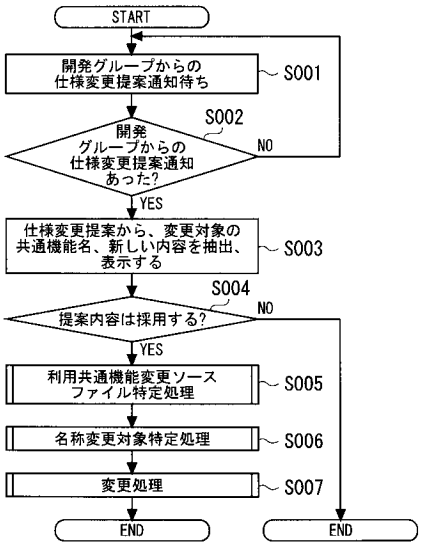
【図５】

影響重付DBのデータ構造を示す表

リコンパイル	再テスト
1	300

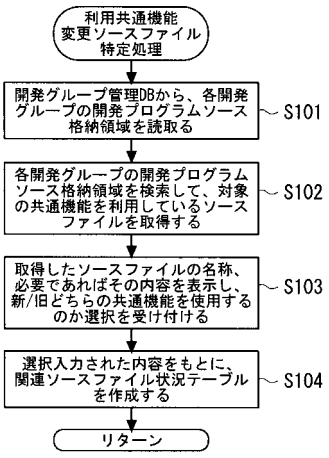
【図６】

開発管理プログラムによる処理を示すフローチャート



【図７】

図6のS005にて実行される利用共通機能変更ソースファイル特定処理サブルーチンの内容を示すフローチャート



【図８】

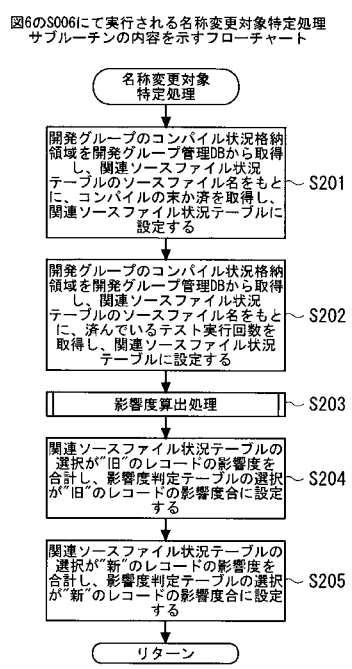


図6のS006にて実行される名称変更対象特定処理
サブルーチンの内容を示すフローチャート
名称変更対象
特定処理
開発グループのコンパイル状況格納領域を開発グループ管理DBから取得し、関連ソースファイル状況テーブルのソースファイル名をもとに、コンパイルの末か済を取得し、関連ソースファイル状況テーブルに設定する
S201
開発グループのコンパイル状況格納領域を開発グループ管理DBから取得し、関連ソースファイル状況テーブルのソースファイル名をもとに、済んでいるテスト実行回数を取得し、関連ソースファイル状況テーブルに設定する
S202
影響度算出処理
S203
関連ソースファイル状況テーブルの選択が"旧"のレコードの影響度を合計し、影響度判定テーブルの選択が"旧"のレコードの影響度合に設定する
S204
関連ソースファイル状況テーブルの選択が"新"のレコードの影響度を合計し、影響度判定テーブルの選択が"新"のレコードの影響度合に設定する
S205
リターン

【図９】

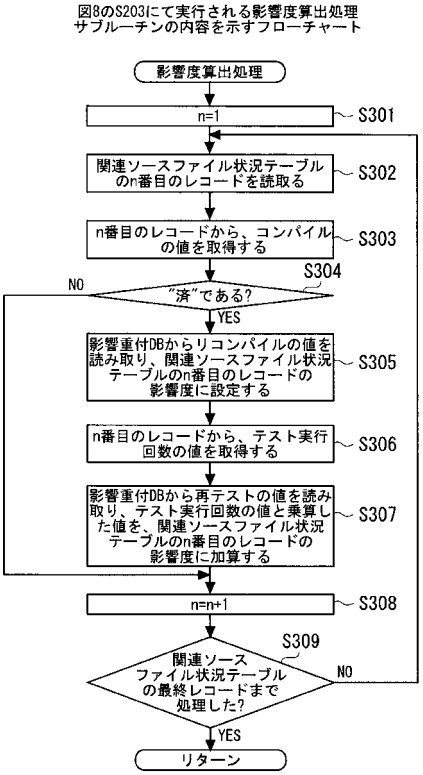


図8のS203にて実行される影響度算出処理
サブルーチンの内容を示すフローチャート
影響度算出処理
n=1
S301
関連ソースファイル状況テーブルのn番目のレコードを読取る
S302
n番目のレコードから、コンパイルの値を取得する
S303
S304
NO
"済"である?
YES
影響重付DBからリコンパイルの値を読み取り、関連ソースファイル状況テーブルのn番目のレコードの影響度に設定する
S305
n番目のレコードから、テスト実行回数の値を取得する
S306
影響重付DBから再テストの値を読み取り、テスト実行回数の値と乗算した値を、関連ソースファイル状況テーブルのn番目のレコードの影響度に加算する
S307
n=n+1
S308
S309
関連ソースファイル状況テーブルの最終レコードまで処理した?
NO
YES
リターン

【図１０】

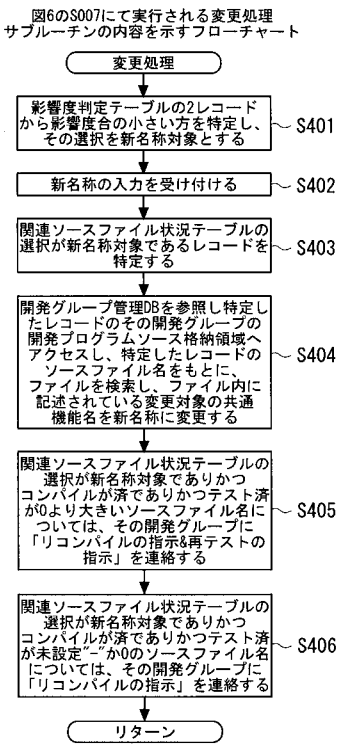
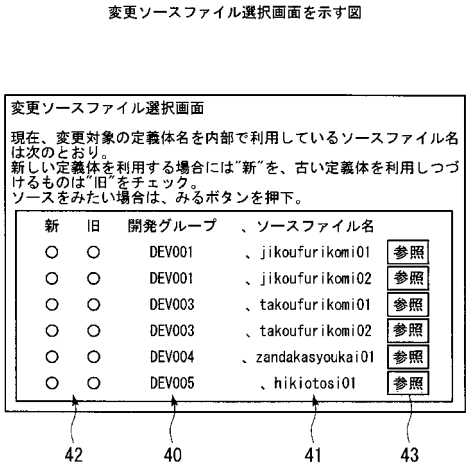


図6のS007にて実行される変更処理
サブルーチンの内容を示すフローチャート
変更処理
影響度判定テーブルの2レコードから影響度合の小さい方を特定し、その選択を新名称対象とする
S401
新名称の入力を受け付ける
S402
関連ソースファイル状況テーブルの選択が新名称対象であるレコードを特定する
S403
開発グループ管理DBを参照し特定したレコードのその開発グループの開発プログラムソース格納領域へアクセスし、特定したレコードのソースファイル名をもとに、ファイルを検索し、ファイル内に記述されている変更対象の共通機能名を新名称に変更する
S404
関連ソースファイル状況テーブルの選択が新名称対象でありかつコンパイルが済でありかつテスト済が0より大きいソースファイル名については、その開発グループに「リコンパイルの指示&再テストの指示」を連絡する
S405
関連ソースファイル状況テーブルの選択が新名称対象でありかつコンパイルが済でありかつテスト済が未設定"-"か0のソースファイル名については、その開発グループに「リコンパイルの指示」を連絡する
S406
リターン

【図１１】


変更ソースファイル選択画面を示す図
変更ソースファイル選択画面
現在、変更対象の定義体名を内部で利用しているソースファイル名は次のとおり。
新しい定義体を利用する場合には"新"を、古い定義体を利用しつづけるものは"旧"をチェック。
ソースをみたい場合は、みるボタンを押下。
新 旧 開発グループ 、ソースファイル名
○ ○ DEV001 、jikoufurikomi01 参照
○ ○ DEV001 、jikoufurikomi02 参照
○ ○ DEV003 、takoufurikomi01 参照
○ ○ DEV003 、takoufurikomi02 参照
○ ○ DEV004 、zandakasyoukai01 参照
○ ○ DEV005 、hikiotosi01 参照
42
40
41
43

フロントページの続き

(72)発明者　御園生　章彦
神奈川県横浜市神奈川区新子安一丁目２番４号株式会社富士通アドバンストソリューションズ内

Ｆターム(参考) 5B176 AC01　EC07