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

特許第6058498号
(P6058498)

(45)発行日　平成29年1月11日(2017.1.11)　(24)登録日　平成28年12月16日(2016.12.16)

(51)Int.Cl.		F I		
G06F 9/45	(2006.01)	G06F	9/44	322M
G06F 11/36	(2006.01)	G06F	9/44	322L
		G06F	9/44	322J
		G06F	11/36	104

請求項の数 15　(全 23 頁)

(21)出願番号　特願2013-155531(P2013-155531)
(22)出願日　平成25年7月26日(2013.7.26)
(65)公開番号　特開2015-26262(P2015-26262A)
(43)公開日　平成27年2月5日(2015.2.5)
審査請求日　平成28年1月28日(2016.1.28)

(73)特許権者　000005108
株式会社日立製作所
東京都千代田区丸の内一丁目６番６号
(74)代理人　110001678
特許業務法人藤央特許事務所
(72)発明者　弓場元　準
神奈川県横浜市戸塚区戸塚町２１６番地
株式会社日立製作所　通信ネットワーク事業部内
(72)発明者　安部　哲朗
神奈川県横浜市戸塚区戸塚町２１６番地
株式会社日立製作所　通信ネットワーク事業部内

最終頁に続く

(54)【発明の名称】コンパイル方法、プログラム及びコンパイル装置

(57)【特許請求の範囲】
【請求項１】
プロセッサとメモリを備えた計算機で、ソースファイルを読み込んで実行バイナリファイルを出力するコンパイル方法であって、
ビジネスプロセスを構成するプロセスとモジュールを含むインターフェースファイルに前記ビジネスプロセスのデータの入出力情報が定義され、前記インターフェースファイルに前記ビジネスプロセスで使用するデータ集合に対する操作情報が定義され、前記計算機が前記インターフェースファイルを受け付ける第１のステップと、
前記計算機が、前記インターフェースファイルに定義されたデータ集合に対する操作情報を検証する第２のステップと、
前記計算機が、検証結果が不正な場合には前記実行バイナリファイルの生成を禁止する第３のステップと、
前記計算機が、前記検証結果が正当な場合には前記インターフェースファイルを含むソースファイルから前記実行バイナリファイルを生成する第４のステップと、
を含み、
前記第２のステップは、
前記ビジネスプロセスを構成する複数のプロセス間で、前記インターフェースファイルのデータ集合に対する操作情報を検証し、
当該検証は、
前記インターフェースファイルに定義された前記ビジネスプロセスで使用するデータ集

合に対する操作情報を作成、参照、更新、削除で構成される操作種類に分類し、当該操作種類を操作情報の処理順序を保って格納し、さらにデータ集合の作成が含まれる前記操作種類を集計し、当該集計した結果をＣＲＵＤ情報として生成し、前記データ集合への操作種類のパターンが不正であるか否かを予め設定したＣＲＵＤバリデーション情報を用いて、前記データ集合に対する前記ＣＲＵＤ情報の操作種類のパターンについて検証を行うことを特徴とするコンパイル方法。

【請求項２】

請求項１に記載のコンパイル方法であって、

前記第２のステップは、

前記インターフェースファイルに定義された前記入出力情報を検証するステップを含み、

前記第３のステップは、

前記入出力情報の検証結果が不正な場合には前記実行バイナリファイルの生成を禁止するステップを含むことを特徴とするコンパイル方法。

【請求項３】

請求項２に記載のコンパイル方法であって、

前記計算機が、前記入出力情報の前記検証結果が不正な場合、または、前記インターフェースファイルのデータ集合に対する操作情報の前記検証結果が不正な場合には、通知情報を生成する第５のステップをさらに含むことを特徴とするコンパイル方法。

【請求項４】

請求項２に記載のコンパイル方法であって、

前記第２のステップは、

前記ビジネスプロセスを構成する複数のプロセス間で、前記入出力情報を検証することを特徴とするコンパイル方法。

【請求項５】

請求項４に記載のコンパイル方法であって、

前記第２のステップは、

前記インターフェースファイルに定義された前記ビジネスプロセスのデータの入出力情報に基づいて、前記ビジネスプロセスを構成する前記プロセスの入力情報に対してビジネスプロセスの入力情報と定義されているか、または前記ビジネスプロセス内の自らより前のプロセスで出力情報とされていることを検証することを特徴とするコンパイル方法。

【請求項６】

請求項１に記載のコンパイル方法であって、

前記第４のステップは、

前記検証結果が正当な場合には、前記ビジネスプロセスで使用するデータ集合を生成することを特徴とするコンパイル方法。

【請求項７】

プロセッサとメモリを備えた計算機で、ソースファイルを読み込んで実行バイナリファイルを出力するプログラムであって、

ビジネスプロセスを構成するプロセスとモジュールを含むインターフェースファイルに前記ビジネスプロセスのデータの入出力情報が定義され、前記インターフェースファイルに前記ビジネスプロセスで使用するデータ集合に対する操作情報が定義され、前記インターフェースファイルを受け付ける第１の手順と、

前記インターフェースファイルに定義されたデータ集合に対する操作情報を検証する第２の手順と、

検証結果が不正な場合には前記実行バイナリファイルの生成を禁止する第３の手順と、前記検証結果が正当な場合には前記インターフェースファイルを含むソースファイルから

前記実行バイナリファイルを生成する第４の手順と、

を前記計算機に実行させ、

前記第２の手順は、

前記ビジネスプロセスを構成する複数のプロセス間で、前記インターフェースファイルのデータ集合に対する操作情報を検証し、
当該検証は、
前記インターフェースファイルに定義された前記ビジネスプロセスで使用するデータ集合に対する操作情報を作成、参照、更新、削除で構成される操作種類に分類し、当該操作種類を操作情報の処理順序を保って格納し、さらにデータ集合の作成が含まれる前記操作種類を集計し、当該集計した結果をＣＲＵＤ情報として生成し、前記データ集合への操作種類のパターンが不正であるか否かを予め設定したＣＲＵＤバリデーション情報を用いて、前記データ集合に対する前記ＣＲＵＤ情報の操作種類のパターンについて検証を行うことを特徴とするプログラム。

【請求項８】
請求項７に記載のプログラムであって、
前記第２の手順は、
前記インターフェースファイルに定義された前記入出力情報を検証する手順を含み、
前記第３の手順は、
前記入出力情報の検証結果が不正な場合には前記実行バイナリファイルの生成を禁止する手順を含むことを特徴とするプログラム。

【請求項９】
請求項８に記載のプログラムであって、
前記入出力情報の前記検証結果が不正な場合、または、前記インターフェースファイルのデータ集合に対する操作情報の前記検証結果が不正な場合には、通知情報を生成する第５の手順をさらに含むことを特徴とするプログラム。

【請求項１０】
請求項８に記載のプログラムであって、
前記第２の手順は、
前記ビジネスプロセスを構成する複数のプロセス間で、前記入出力情報を検証することを特徴とするプログラム。

【請求項１１】
請求項１０に記載のプログラムであって、
前記第２の手順は、
前記インターフェースファイルに定義された前記ビジネスプロセスのデータの入出力情報に基づいて、前記ビジネスプロセスを構成する前記プロセスの入力情報に対してビジネスプロセスの入力情報と定義されているか、または前記ビジネスプロセス内の自らより前のプロセスで出力情報とされていることを検証することを特徴とするプログラム。

【請求項１２】
請求項７に記載のプログラムであって、
前記第４の手順は、
前記検証結果が正当な場合には、前記ビジネスプロセスで使用するデータ集合を生成することを特徴とするプログラム。

【請求項１３】
プロセッサとメモリを備えた計算機で、ソースファイルを読み込んで実行バイナリファイルを出力するコンパイラを備えたコンパイル装置であって、
前記コンパイラは、
ビジネスプロセスを構成するプロセスとモジュールを含むインターフェースファイルに前記ビジネスプロセスのデータの入出力情報が定義され、前記インターフェースファイルに前記ビジネスプロセスで使用するデータ集合に対する操作情報が定義され、前記計算機が前記インターフェースファイルを受け付けて、前記インターフェースファイルに定義されたデータ集合に対する操作情報を検証し、検証結果が不正な場合には前記実行バイナリファイルの生成を禁止する解析部と、
前記検証結果が正当な場合には前記インターフェースファイルを含むソースファイルか

ら前記実行バイナリファイルを生成する実行バイナリ生成部と、
を備え、
前記解析部は、
前記ビジネスプロセスを構成する複数のプロセス間で、前記インターフェースファイルのデータ集合に対する操作情報を検証し、
当該検証は、
前記インターフェースファイルに定義された前記ビジネスプロセスで使用するデータ集合に対する操作情報を作成、参照、更新、削除で構成される操作種類に分類し、当該操作種類を操作情報の処理順序を保って格納し、さらにデータ集合の作成が含まれる前記操作種類を集計し、当該集計した結果をＣＲＵＤ情報として生成し、前記データ集合への操作種類のパターンが不正であるか否かを予め設定したＣＲＵＤバリデーション情報を用いて、前記データ集合に対する前記ＣＲＵＤ情報の操作種類のパターンについて検証を行うことを特徴とするコンパイル装置。
【請求項１４】
請求項１３に記載のコンパイル装置であって、
前記インターフェースファイルのデータ集合に対する操作情報の前記検証結果が不正な場合には、通知情報を生成するエラー通知部をさらに有することを特徴とするコンパイル装置。
【請求項１５】
請求項１３に記載のコンパイル装置であって、
前記インターフェースファイルのデータ集合に対する操作情報の前記検証結果が正当な場合には、前記ビジネスプロセスで使用するデータ集合を生成する自動生成部をさらに有することを特徴とするコンパイル装置。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、イベント制御基盤におけるアプリケーションの開発効率化に関する。
【背景技術】
【０００２】
基幹系の情報システムには、通信事業者がサービスを提供するための顧客管理や料金請求、物流、代理店等のＯＳＳ／ＢＳＳ（Operations Support System／Business Support System）基盤(以降「イベント制御基盤」と示す)が存在する。
【０００３】
基幹系の情報システムについては、例えば特許文献１には、ビジネスプロセスに、複数の業務処理を実行する順序を定義し、複数の業務処理に対して拡張機能の起動条件を別の定義で管理することで、特定の業務処理が実行されたタイミングを検知し、拡張機能の処理を実行していることが記載されている。
【０００４】
本技術分野の背景技術として、現状のａｐｔ（Ａｎｎｏｔａｔｉｏｎ　Ｐｒｏｃｅｓｓｉｎｇ　Ｔｏｏｌ）について以下に説明する。
【０００５】
ａｐｔは、ｊａｖａｃコマンド（Ｊａｖａ（登録商標）コンパイラ起動コマンド）のオプションによりアクセスされ、新しいソースコードと他のファイルを作成するアノテーションプロセッサを実行し、元ファイルと作成されたファイルのコンパイルを行う（例えば、非特許文献１参照）。
【０００６】
一般にａｐｔより実行されるアノテーションプロセッサは、Ｊａｖａのソースコード及びアノテーションを処理し、チェックとファイル作成の実装を行い、アプリケーション開発補助を目的として使用される。
【先行技術文献】

【特許文献】
【０００７】
【特許文献１】特開２０１２－１４５０６号公報
【非特許文献】
【０００８】
【非特許文献１】apt（Annotation Processing Tool）、”annotation processing tool”、［online］、［平成25年03月15日検索］、インターネット＜URL： http://docs.oracle.com/javase/6/docs/technotes/guides/apt/index.html＞
【発明の概要】
【発明が解決しようとする課題】
【０００９】
イベント制御基盤は、ビジネスプロセスをプロセスという処理単位に分割して、完全に独立させることにより疎結合での開発を可能にして開発効率を高めた。さらに、このプロセスは共通的な処理を切り出して、複数のビジネスプロセスで使用することで開発工数の削減も可能としている。
【００１０】
しかし、この時プロセスは自らより前に実行されたプロセスの結果を使用して処理するようになるため、起動するためには「Ａというデータは必ずある」や「テーブルに対象レコードは必ずある」というような前提条件ができてしまった。
【００１１】
そのため、一部のビジネスプロセスの変更によって共通のプロセスの前提条件に変化があった場合、開発者が当該共通のプロセスを使用している全てのビジネスプロセスの見当を付け、設計書やソースを調べて特定してから、共通のプロセスの修正を行っている。共通のプロセスの修正漏れは、結合試験や総合試験で発見されることになり、リリースまでの時間の余裕もなく手戻りが発生してしまうという問題があった。
【００１２】
そこで本発明は、上記問題点に鑑みてなされたもので、ソースファイルの不具合を早期に検出してアプリケーションの開発効率を向上させることを目的とする。
【課題を解決するための手段】
【００１３】
本発明は、プロセッサとメモリを備えた計算機で、ソースファイルを読み込んで実行バイナリファイルを出力するコンパイル方法であって、ビジネスプロセスを構成するプロセスとモジュールを含むインターフェースファイルに前記ビジネスプロセスのデータの入出力情報が定義され、前記インターフェースファイルに前記ビジネスプロセスで使用するデータ集合に対する操作情報が定義され、前記計算機が前記インターフェースファイルを受け付ける第１のステップと、前記計算機が、前記インターフェースファイルに定義されたデータ集合に対する操作情報を検証する第２のステップと、前記計算機が、検証結果が不正な場合には前記実行バイナリファイルの生成を禁止する第３のステップと、前記計算機が、前記検証結果が正当な場合には前記インターフェースファイルを含むソースファイルから前記実行バイナリファイルを生成する第４のステップと、を含み、前記第２のステップは、前記ビジネスプロセスを構成する複数のプロセス間で、前記インターフェースファイルのデータ集合に対する操作情報を検証し、当該検証は、前記インターフェースファイルに定義された前記ビジネスプロセスで使用するデータ集合に対する操作情報を作成、参照、更新、削除で構成される操作種類に分類し、当該操作種類を操作情報の処理順序を保って格納し、さらにデータ集合の作成が含まれる前記操作種類を集計し、当該集計した結果をＣＲＵＤ情報として生成し、前記データ集合への操作種類のパターンが不正であるか否かを予め設定したＣＲＵＤバリデーション情報を用いて、前記データ集合に対する前記ＣＲＵＤ情報の操作種類のパターンについて検証を行う。
【発明の効果】
【００１４】

したがって、本発明は、ソースファイルのコンパイル時に入出力情報と操作情報の検証を行うことで、ソースファイルの修正漏れや相違を早期に検出し、アプリケーションの開発効率を向上させることが可能となる。

【図面の簡単な説明】

【００１５】

【図１】本発明の実施形態を示し、アプリケーションを開発する計算機システムの一例を示すブロック図ある。

【図２】本発明の実施例を示し、アプリケーション開発の手順を示すシーケンス図の一例である。

【図３Ａ】本発明の実施例を示し、イベント制御基盤を用いたビジネスプロセスの処理単位の一例を示す分解図である。

【図３Ｂ】本発明の実施例を示し、イベント制御基盤を用いたビジネスプロセスの処理単位の拡大図である。

【図３Ｃ】本発明の実施例を示し、イベント制御基盤を用いた他のビジネスプロセスの処理単位の拡大図である。

【図４Ａ】本発明の実施例を示し、図３Ａ、図３Ｃに示したビジネスプロセス２０１のｊａｖａインターフェースファイルの一例を示す。

【図４Ｂ】本発明の実施例を示し、図３Ａ、図３Ｃに示したプロセス２１１のｊａｖａインターフェースファイルの一例を示す。

【図４Ｃ】本発明の実施例を示し、図３Ａ、図３Ｃに示したモジュール２２１のｊａｖａインターフェースファイルの一例を示す。

【図４Ｄ】本発明の実施例を示し、図３Ａ、図３Ｃに示したモジュール２２５のｊａｖａインターフェースファイルの一例を示す。

【図５】本発明の実施例を示し、アプリケーション開発端末及びビルドサーバで実行されるコンパイラの機能ブロック図の例である。

【図６】本発明の実施例を示し、アプリケーション開発端末及びビルドサーバの構成の一例を示すブロック図である。

【図７】本発明の実施例を示し、コンパイル処理のコンパイル対象が正常またはワーニング時のシーケンス図である。

【図８】本発明の実施例を示し、コンパイル処理のコンパイル対象がエラー時のシーケンス図である。

【図９】本発明の実施例を示し、入出力情報バリデーションの一例を示すフローチャートである。

【図１０】本発明の実施例を示し、ＣＲＵＤ情報バリデーションのフローチャートの例である。

【図１１】本発明の実施例を示し、前提ビジネスプロセスのＣＲＵＤ情報集計時を示した図の例である。

【図１２】本発明の実施例を示し、ビジネスプロセスのＣＲＵＤ情報バリデーション結果の図の例である。

【図１３】本発明の実施例を示し、自動生成処理の一例を示すフローチャートである。

【発明を実施するための形態】

【００１６】

以下、本発明の一実施形態を添付図面に基づいて説明する。

【００１７】

図１は、本発明の実施例を示し、アプリケーションを開発する計算機システムの一例を示すブロック図ある。

【００１８】

図１において、リポジトリサーバ１０１とビルドサーバ１０２とアプリケーションサーバ１０３とアプリケーション開発端末１０５及び１０６は、それぞれネットワーク１０４に接続され、各々の計算機間の通信はネットワーク１０４を経由して行われる。なお、ア

プリケーション開発端末１０５、１０６は２台に限定されるものではなく、所望の台数を配置すれば良い。

【００１９】

リポジトリサーバ１０１は、アプリケーションの開発に関連するデータを格納し、例えば、アプリケーションの仕様や、ソースコードや、実行バイナリファイルなどが格納される。また、アプリケーションのバージョンの管理などを行うことができる。

【００２０】

ビルドサーバ１０２は、ソースコードのコンパイルを行って実行バイナリファイル（またはオブジェクトコード）を出力する。アプリケーション開発端末１０５及び１０６は、ソースコードの編集やコンパイルを実行する。アプリケーションサーバ１０３は、開発されたアプリケーション（実行バイナリファイル）を実行する。なお、本実施例では、ビルドサーバ１０２とアプリケーション開発端末１０５及び１０６は同一の構成として、ソースコードの編集（コーディング）、コンパイルを実行することが可能となっている。

【００２１】

図２は、アプリケーション開発端末１０５、アプリケーション開発端末１０６及びビルドサーバ１０２でアプリケーションを開発し、アプリケーションサーバ１０３でアプリケーション実行を行うまでの処理の一例を示すシーケンス図である。

【００２２】

まず、アプリケーション開発端末１０５及びアプリケーション開発端末１０６で、図４Ａから図４Ｄのｊａｖａインターフェースファイルを含むソースファイルをステップ１２１及び１２２で開発者が編集（コーディング）する。

【００２３】

次にステップ１２３及び１２４でアプリケーション開発端末１０５、１０６がソースファイルを受け付けてコンパイルを実行し、実行バイナリファイルを生成する。このとき、アプリケーション開発端末１０５、１０６は、図４Ａ～図４Ｄに示す本発明に固有のアノテーション解析部４０７と、エラー通知部４０８及び自動生成部４０９を実行する。

【００２４】

アプリケーション開発端末１０５、１０６を利用する開発者は、コンパイルの結果を確認し、問題がある場合は、再度ステップ１２１及び１２２のコーディングを行う。一方、コンパイルの結果に問題ない場合、ステップ１２５及び１２６の単体試験を実施し、コーディングしたｊａｖａインターフェースファイルが正しいことを確認する。

【００２５】

その後、アプリケーション開発端末１０５、１０６では、ステップ１２７及び１２８でリポジトリサーバ１０１に、コーディングしたｊａｖａインターフェースファイルを含むソースファイルのコミット（格納）要求を行う。コミット要求を受け付けたリポジトリサーバ１０１は、ステップ１２９及び１３０で最新のソースファイルとしてコミットを実行し、コミット結果をステップ１３１及び１３２で要求元に返却する。開発者はコミット結果を受けたアプリケーション開発端末１０５及び１０６でコミットの結果に問題ないことを確認する。

【００２６】

次に、ビルドサーバ１０２はステップ１３３でビルド実行を行い、ステップ１３４でソースファイル取得要求をリポジトリサーバ１０１に送信する。なお、本実施例では、ビルドサーバ１０２は、所定の周期でビルドを実行するが、リポジトリサーバ１０１やアプリケーション開発端末１０５、１０６からの指令を受けてビルドを行うようにしても良い。

【００２７】

ソースファイル取得要求を受け付けたリポジトリサーバ１０１は、ステップ１３５でソースファイルをビルドサーバ１０２に送信する。ソースファイルを受信したビルドサーバ１０２はステップ１３３のビルド実行を再開し、受信したソースファイルを対象にコンパイルを実行し、図４Ａから図４Ｄに示す本発明に固有のアノテーション解析部４０７、エラー通知部４０８及び自動生成部４０９を実行させる。なお、図示はしないが、ビルドサ

ーバ１０２は、エラー通知部４０８等によりビルドの結果をアプリケーション開発端末１０５等に通知したり、あるいは、リポジトリサーバ１０１にビルドの結果を格納することができる。開発者はコンパイルの結果をアプリケーション開発端末１０５、１０６で確認し、問題がある場合は、原因となるソースファイルに対して、再度ステップ１２１及び１２２のコーディングを行うことができる。また、ビルドサーバ１０２は、コンパイルの結果に問題ない場合、ソースファイルのコンパイル結果である実行バイナリファイルを生成してから圧縮する。

【００２８】

次に、アプリケーションサーバ１０３は、ステップ１３６で、ビルドサーバ１０２が生成した実行バイナリファイルの取得を行い、ステップ１３７の実行バイナリファイル取得要求をビルドサーバ１０２に送信する。

【００２９】

実行バイナリファイル取得要求を受けたビルドサーバ１０２は、ステップ１３８で実行バイナリファイルをアプリケーションサーバ１０３に送信する。アプリケーションサーバ１０３は、受信した圧縮済みの実行バイナリファイルをステップ１３９の実行バイナリファイル展開でメモリ（図示省略）に展開する。そして、アプリケーションサーバ１０３は、アプリケーション開発端末１０５及びアプリケーション開発端末１０６で開発したアプリケーションをステップ１４０で実行する。

【００３０】

なお、上記ステップ１３６の実行バイナリファイルの取得処理は、アプリケーションサーバ１０３が所定の周期で実行することができる。あるいは、アプリケーションサーバ１０３が、アプリケーション開発端末１０５等から要求を受け付けたときに、実行バイナリファイルの取得処理を実行するようにしてもよい。

【００３１】

図３Ａ～図３Ｃは、イベント制御基盤を用いて開発されたアプリケーションの一例としてビジネスプロセスを示し、図３Ａはビジネスプロセスの処理単位の一例を示す分解図である。図３Ｂ、図３Ｃは、ビジネスプロセスの処理単位の拡大図である。ビジネスプロセス２０１、２０２はそれぞれアプリケーションサーバ１０３で実行される。

【００３２】

図３Ａ、図３Ｃに示すビジネスプロセス２０１（新規契約などの処理単位）には、複数のプロセス（契約者登録などの処理単位）が直列に結合されており、プロセスは１以上のモジュール（契約テーブル登録などの処理単位）の集合から構成される。プロセスのモジュールは必ずしも直列ではなく分岐が存在し、条件によっては実行されないモジュールが存在する。

【００３３】

まず、ビジネスプロセス２０１では、データ２０３（データＡ）を入力情報とし、データ２０４（データＦ）を出力情報として、プロセス２１１、プロセス２１７、プロセス２１８の順番でプロセスが順次実行される。

【００３４】

また、ビジネスプロセス２０１の実行時に、必ず実行されているビジネスプロセス（以下、前提ビジネスプロセス）として図３Ａ、図３Ｂのビジネスプロセス２０２があり、アプリケーションサーバ１０３はプロセス２１７を、ビジネスプロセス２０１と２０２で共有して使用する。

【００３５】

次に、ビジネスプロセス２０１で実行されるプロセス２１１は、データ２１２（データＡ＝２０３）を入力情報とし、データ２１３（データＣ）を出力情報として、内部のモジュール２２１、モジュール２２５の順番でアプリケーションサーバ１０３が実行する。

【００３６】

次に、プロセス２１１中で実行されるモジュール２２１はデータ２２２（データＡ＝２１２）を入力情報とし、データ２２３（データＢ）を出力情報として実行し、出口２２４

から出力する。
【００３７】
次に、プロセス２１１中で実行されるモジュール２２５はデータ２２７（データＢ＝２２３）を入力情報とし、処理２２６としてテーブルＡの作成を行い、データ２２８（データＣ＝２１３）を出力情報として実行し、出口２２９から出力する。
【００３８】
図４Ａから図４Ｄは、図３Ａ～図３Ｃのビジネスプロセス２０１、２０２をｊａｖａインターフェースファイルで表現した例であり、アプリケーションの開発者がコーディングするファイルである。また、図中「@（アットマーク）」から書かれるものがアノテーションである。
【００３９】
図４Ａは、図３Ａ、図３Ｃに示したビジネスプロセス２０１を示すｊａｖａインターフェースファイルの例である。以下、ビジネスプロセス２０１のｊａｖａインターフェースファイルの記述内容について説明する。
【００４０】
記載３０１は、ビジネスプロセス２０１には、当該ビジネスプロセス２０１が利用するプロセスまたはモジュールを含む前提ビジネスプロセスとしてビジネスプロセス２０２が存在することを示す。
【００４１】
記載２０３は、ビジネスプロセス２０１の入力情報であるデータＡを示す。
【００４２】
記載２０４は、ビジネスプロセス２０１の出力情報であるデータＢを示す。
【００４３】
記載３０２は、ビジネスプロセス２０１が実行するプロセス２１１、２１７、２１８と実行順序を示す。
【００４４】
図４Ｂは、プロセス２１１を示すｊａｖａインターフェースファイルの例である。以下、プロセス２１１のｊａｖａインターフェースファイルの記述内容について説明する。
【００４５】
記載２１２は、プロセス２１１への入力情報であるデータ２１２（データＡ＝２０３）を示す。
【００４６】
記載２１３は、プロセス２１１の出力情報であるデータ２１３（データＣ＝２２８）を示す。
【００４７】
記載３２１は、プロセス２１１で最初に実行するモジュールがモジュール２２１であることを示す。
【００４８】
記載３２２は、プロセス２１１でモジュール２２１が「ｐａｓｓ（出口２２４）」の場合にモジュール２２５へ遷移することを示す。
【００４９】
記載３２３は、プロセス２１１でモジュール２２５が「ｐａｓｓ（出口２２９）」の場合にプロセス２１１の終了へ遷移することを示す。
【００５０】
図４Ｃは、モジュール２２１を示すｊａｖａインターフェースファイルの例である。以下、モジュール２２１のｊａｖａインターフェースファイルの記述内容について説明する。
【００５１】
記載２２２は、モジュール２２１の入力情報であるデータ２２２（データＡ＝２０３）を示す。

【００５２】
記載２２３は、モジュール２２１の出力情報であるデータ２２３（データＢ＝２２７）を示す。
【００５３】
記載２２４は、モジュール２２１の出口２２４を示す。
【００５４】
図４Ｄは、モジュール２２５を示すｊａｖａインターフェースファイルの例である。以下、モジュール２２５のｊａｖａインターフェースファイルの記述内容について説明する。記載２２６は、モジュール２２５のＣＲＵＤ（Create、Read、Update、Delete）情報でＴａｂｌｅＡを作成することを示す。
【００５５】
記載２２７は、モジュール２２５の入力情報であるデータ２２７（データＢ＝２２３）を示す。
【００５６】
記載２２８は、モジュール２２５の出力情報であるデータ２２８（データＣ＝２１３）を示す。
【００５７】
記載２２９は、モジュール２２５の出口２２９を示す。
【００５８】
図５は、本発明のコンパイラのブロック図であり、アプリケーション開発端末１０５、１０６及び、ビルドサーバ１０２で実行される。
【００５９】
コンソール部４０１は、図示しない入力装置と出力装置を含む。コンソール部４０１が受け付けたコンパイル要求により、コンパイル実行部４０２及びコンパイル部４０３が機能し、アプリケーション開発端末１０５，１０６であれば自ストレージ内のソースファイルを対象にし、ビルドサーバ１０２であればリポジトリサーバ１０１から取得して自ストレージ内に格納したソースファイルを対象としコンパイルを実行する。なお、コンパイル実行部４０２及びコンパイル部４０３からコンパイラが構成される。
【００６０】
コンパイル実行部４０２は、コンパイル部４０３を構成するｊａｖａコンパイル時の標準であるソースコード解析部４０４と、ａｐｔ（Ａｎｎｏｔａｔｉｏｎ　Ｐｒｏｃｅｓｓｉｎｇ　Ｔｏｏｌ）部４０６と、実行バイナリファイル作成部４０５を呼び出して、取得したソースファイルから実行バイナリファイルを作成する。
【００６１】
ＡＰＴ部４０６は、図４Ａから図４Ｄに示したｊａｖａインターフェースファイルを対象として機能する本発明に固有のアノテーション解析部４０７を起動する。アノテーション解析部４０７では、ｊａｖａインターフェースファイルのアノテーションを取得し、データの入出力情報バリデーションと、テーブルなどに対する操作情報であるＣＲＵＤ情報バリデーションを実行し、通知情報（エラーやワーニング）がある場合は、エラー通知部４０８に通知要求を発行する。
【００６２】
また、アノテーション解析部４０７は、通知情報の内容を判定し、自動生成可能と判定した場合に自動生成部４０９に、ソースファイルの自動生成要求６０８を出力する。本コンパイラを実行させる構成図を図６に、コンパイラの実行時のシーケンス図を図７及び図８に示す図を用いて説明する。
【００６３】
図６は、図５のコンパイラとしてのコンパイル実行部４０２及びコンパイル部４０３を機能させるアプリケーション開発端末１０５、１０６及びビルドサーバ１０２の機能の一例を示すブロック図である。
【００６４】

アプリケーション開発端末１０５、１０６及びビルドサーバ１０２は、同一の構成であるので、以下ではアプリケーション開発端末１０５の構成について説明する。

【００６５】

アプリケーション開発端末１０５は、ＣＰＵ５０１とメモリ５０２とストレージ５０６とネットワークインターフェース５１２は、データバス５１１により接続され、各々の通信をデータバス５１１を経由して行う。ネットワークインターフェース５１２はネットワーク１０４と接続される。また、アプリケーション開発端末１０５には入力装置と出力装置を含むコンソール部４０１も接続される。

【００６６】

図５に示す各機能ブロックは、通常時には、ストレージ５０６にファイルとして格納される。すなわち、コンパイル実行部４０２、ソースコード解析部４０４及び、実行バイナリファイル作成部４０５はフレームワークプログラムファイル５０７に格納され、アノテーション解析部４０７、エラー通知部４０８及び、自動生成部４０９はアノテーションプロセッサプログラムファイル５０８に格納される。

【００６７】

実際にコンパイルが実行される際には、ＣＰＵ５０１の命令によって、ストレージ５０６のそれぞれ格納された場所からデータバス５１１を経由してメモリ５０２に展開され、ＣＰＵ５０１によって実行される。

【００６８】

図６において、フレームワークプログラムファイル５０７は、メモリ５０２において、フレームワークプログラム５０３として展開され、アノテーションプロセッサプログラムファイル５０８はアノテーションプロセッサプログラム５０４として展開され、コンパイル実行部４０２を実行し、実行結果として、ストレージ５０６の実行バイナリファイル５１０及び、ソースファイル５０９を作成する。

【００６９】

ソースファイル５０９が作成された場合は、当該ソースファイル５０９を対象に再度コンパイル実行部４０２を実行する。一時データ５０５は、一時的な記憶であり、コンパイル実行部４０２の終了後に消去される。

【００７０】

ネットワークインターフェース５１２は、ネットワーク１０４を経由し、図１に示したそれぞれの端末との通信を接続する。

【００７１】

個々のモジュールが実行される際に必要なデータは、一時データ５０５及びストレージ５０６に格納されている。一時データ５０５は、必要に応じて参照及び更新される。

【００７２】

図７は、図５のコンパイル部４０３を実行したときにコンパイル対象が正常またはワーニング時のシーケンス図である。

【００７３】

まず、コンソール部４０１はステップ６０１で、コンパイル実行部４０２へコンパイル実行要求を行う。コンパイル実行部４０２は、コンパイルの経過をコンソール部４０１で表示するため、ステップ６１３のソースコード確認結果出力と、ステップ６０６のコンソール出力及び、ステップ６０７のコンパイル完了を出力し、コンソール部４０１がこれらの出力を受け付けて表示を行う。

【００７４】

コンパイル実行要求を受けたコンパイル実行部４０２は、ステップ６１０でソースコード解析部４０４にソースコード解析要求を行う。ソースコード解析要求を受けたソースコード解析部４０４は、ステップ６１１でソースファイル５０９のソースコード解析を実行し、ステップ６１２でソースコード解析結果をコンパイル実行部４０２に応答する。ソースコード解析結果を受けたコンパイル実行部４０２は、ステップ６１３でコンソール部４０１にソースコード解析結果出力を行う。

【００７５】
次に、コンパイル実行部４０２は、ステップ６１４でＡＰＴ部４０６にＡＰＴ実行要求を行う。ＡＰＴ実行要求を受けたＡＰＴ部４０６は、ステップ６０２でアノテーション解析部４０７にソースファイル５０９のアノテーション解析要求を行う。
【００７６】
アノテーション解析要求を受けたアノテーション解析部４０７は、ステップ６０３で入出力情報バリデーション処理と、ステップ６０４でＣＲＵＤ情報バリデーション処理を実行し、ステップ６０３及び６０４のバリデーション処理の結果である通知情報がワーニングの場合、ステップ６０５でエラー通知部４０８に通知要求を行う。
【００７７】
入出力情報バリデーション処理またはＣＲＵＤ情報バリデーション処理でワーニングの通知要求を受けたエラー通知部４０８は、受け付けた通知要求を、ステップ６１５でＡＰＴ部４０６にコンソール出力要求を行い、さらにＡＰＴ部４０６は、ステップ６１６でコンパイル実行部４０２にコンソール出力要求を行い、コンパイル実行部４０２はステップ６０６でコンソール部４０１へコンソール出力を実行する。
【００７８】
次にアノテーション解析部４０７は、ステップ６０８で自動生成部４０９に自動生成要求を行う。自動生成要求を受けた自動生成部４０９は、ステップ６０９の自動生成処理で自動生成ソースを作成する。
【００７９】
自動生成部４０９は処理を終えたら、ステップ６１７でアノテーション解析部４０７に自動生成完了を応答する。自動生成完了を受けたアノテーション解析部４０７は、ステップ６１８でＡＰＴ部４０６にアノテーション解析完了を通知する。アノテーション解析完了を受けたＡＰＴ部４０６は、ステップ６１９でコンパイル実行部４０２にＡＰＴ完了を通知する。
【００８０】
次に、コンパイル実行部４０２はステップ６２０で実行バイナリファイル作成部４０５に実行バイナリファイル作成要求を行う。実行バイナリファイル作成要求を受けた実行バイナリファイル作成部４０５は、ステップ６２１で上記ステップ６１１で解析したソースファイル５０９と、ステップ６０９で作成した自動生成ソースを対象に実行バイナリファイルと、ビジネスプロセスで使用するテーブル等のデータ集合を生成する。実行バイナリファイル作成部４０５は、ステップ６２２でコンパイル実行部４０２に対して実行バイナリファイル作成完了を応答し、ステップ６０７でコンソール部４０１にコンパイル完了を通知する。
【００８１】
上記図７のステップ６０３で実行する入出力情報バリデーション処理を図９に示し、ステップ６０４で実行するＣＲＵＤ情報バリデーション処理を図１０から図１２に示し、ステップ６０９で実行する自動生成処理を図１３に示す図を用いて説明する。
【００８２】
図８は、図５のコンパイル実行部４０２でソースファイル５０９を処理したときにコンパイル対象がエラーとなったときのシーケンス図である。ステップ６０１のコンパイル実行要求～ステップ６０６コンソール出力までは、上記図７と同一の処理である。
【００８３】
ステップ６０３の入出力バリデーション処理またはステップ６０４のＣＲＵＤ情報バリデーション処理でエラーとなった場合、図７で示したステップ６０８の自動生成要求以降の処理を実施せずに終了するため、自動生成ソース及びコンパイル対象に関連する実行バイナリファイルは作成されない。
【００８４】
図９は、図７及び図８に示したアノテーション解析部４０７で行われる処理であるデータの入出力情報バリデーションのフローチャートであり、図３Ａ～図３Ｃに示したビジネ

スプロセス２０２、２０１のｊａｖａインターフェースファイル（図４Ａ）ごとに起動される。

【００８５】

まず、アノテーション解析部４０７は、プロセスの入力情報として定義されているデータ（変数）が、ビジネスプロセスの入力情報として定義されているか否かをステップ７０１で判定する。この判定は、定義されている変数の型と変数名が一致するか否かによって判定する。

【００８６】

アノテーション解析部４０７は、プロセスの入力情報として変数の型と変数名が一致する項目がある場合は、ステップ７０４に遷移する。一方、一致する項目がない場合は、ステップ７０２へ遷移する。

【００８７】

次に、アノテーション解析部４０７は、ステップ７０２では、ビジネスプロセス内のチェック対象のプロセスについて、当該プロセスより前のプロセスの出力情報として定義されているか否かを、変数の型と変数名が一致するか否かによって判定する。

【００８８】

アノテーション解析部４０７は、変数の型と変数名が一致する項目がある場合は、ステップ７０４に遷移し、一致する項目がない場合（図３Ａのプロセス２１７の入力情報であるデータ２１９が該当する）はステップ７０３に遷移する。

【００８９】

次に、ステップ７０３では、アノテーション解析部４０７が、エラーなどの通知情報を後述するように作成し、ステップ７０４へ遷移する。

【００９０】

次にステップ７０４では、アノテーション解析部４０７が、ビジネスプロセス内のプロセスの全てについて入出力情報のチェックが完了したか否かを判定する。処理が未了のプロセスがあればステップ７０１に戻って上記処理を繰り返し、全てのプロセスについて処理を終えたら本処理を終了する。なお、図示はしないが、同様の処理をプロセスとプロセス内のモジュールについても行う。

【００９１】

以上の処理により、プロセスで使用するデータの入力情報がビジネスプロセスの入力情報としての定義がなく、かつ、ビジネスプロセス内の他のプロセスで入力情報に対応する出力情報が定義されていない場合には、ビジネスプロセスで入力できない情報となるので、アノテーション解析部４０７は、入出力力情報の検証結果が不正であると判定し、エラー等の通知情報を発行することができる。

【００９２】

図１０は、図７及び図８のアノテーション解析部４０７で行われる処理であるＣＲＵＤ情報バリデーションのフローチャートであり、ビジネスプロセス２０２、２０１のｊａｖａインターフェースファイル（図４Ａ）ごとに起動される。ＣＲＵＤ情報は、プロセスまたはモジュールが使用するテーブル等のデータ集合に対する操作の情報である。

【００９３】

まず、ステップ８１１では、アノテーション解析部４０７が、バリデーションの対象のビジネスプロセスの前提ビジネスプロセスが定義されているか否かを判定する。前提ビジネスプロセス（図３Ａのビジネスプロセス２０２）が定義されている場合はステップ８１２へ遷移し、定義されていない場合はステップ８１３へ遷移する。

【００９４】

次にステップ８１２では、アノテーション解析部４０７が、バリデーションの対象のビジネスプロセスの前提ビジネスプロセスに含まれるＣ（作成）とＤ（削除）のみを対象として集計する。そして、前提ビジネスプロセスのプロセス毎に集計した後に全体で集計し、バリデーションの対象とはせず、バリデーションの対象のビジネスプロセスの前提のＣＲＵＤ情報とする。

【００９５】
また、バリデーションの対象のビジネスプロセスの前提ビジネスプロセスに、さらに前提ビジネスプロセスがある場合も、同様にＣＲＵＤ情報を集計する。なお、Ｄ（削除）は物理削除と論理削除を意味する。この集計に関し、１つのテーブルに対する操作のパターンの例を図１１に示す。なお、プロセス毎のＣＲＵＤ情報の集計方法は、プロセスのフローには分岐が存在するため、全分岐を探索してＣＲＵＤ情報が最大のものを集計結果とする。
【００９６】
次にステップ８１３では、バリデーション対象のビジネスプロセス内のプロセス毎にＣＲＵＤ情報を集計する。このプロセス毎のＣＲＵＤ情報の集計方法は、ステップ８１２と同様であるが、プロセスを跨ぐものを集計するため、集計対象のＣＲＵＤ情報がステップ８１２とは異なり、テーブルのＣ（作成）とＤ（削除）は必ず集計し、Ｒ（参照）とＵ（更新）は自プロセス内で解決できないもののみ集計し、実行するプロセスの順番に並べる。
【００９７】
次に、ステップ８１４では、アノテーション解析部４０７がステップ８１３で集計したＣＲＵＤ情報に対応するテーブル操作が、Ｒ（参照）、Ｕ（更新）、Ｄ（削除）のいずれの定義であるかを順次判定をする。
【００９８】
上記テーブル操作がＲ（参照）、Ｕ（更新）、Ｄ（削除）のいずかひとつに一致すれば、アノテーション解析部４０７はステップ８１５に遷移し、一致しなければステップ８１７に遷移する。次に、アノテーション解析部４０７は、ステップ８１５では、上記ステップ８１４の対象テーブルに対して自らより前のプロセスでＣ（作成）があるか否かを判定する。この判定の結果、自らより前のプロセスで操作対象テーブルのＣ（作成）が行われていれば、アノテーション解析部４０７は、ステップ８１６へ遷移する。
【００９９】
一方、アノテーション解析部４０７は、自らより前のプロセスで操作対象テーブルＡのＣ（作成）が行われていなければ（図３Ａのプロセス２１８でＴａｂｌｅＢを参照している処理２２０が該当する）ステップ７０３へ遷移する。ステップ７０３では、アノテーション解析部４０７が、作成されていないテーブルに対する操作（Ｒ）についてエラーまたはワーニングを通知情報として生成する。
【０１００】
次に、ステップ８１６では、ステップ８１４の対象テーブルに対する操作がＤ（削除）であり、後のプロセスにＣ（作成）がなく、かつＲ（参照）Ｕ（更新）Ｄ（削除）を行っているか否かを判定する。アノテーション解析部４０７は、削除されたままのテーブルに対する操作がなければ、ステップ８１７へ遷移し、削除されたままのテーブルに対する操作があればステップ７０３に遷移する。
【０１０１】
ステップ７０３は、上記図９のステップ７０３と同様の処理であり、削除されたままのテーブルに対する操作について、エラーやワーニング等の通知情報をアノテーション解析部４０７が生成する。
【０１０２】
次に、ステップ８１７では、ステップ８１３で集計したバリデーション対象のビジネスプロセスの全てのＣＲＵＤ情報について処理を終了したか否かを判定する。処理が未了のＣＲＵＤ情報があれば、ステップ８１４に戻って上記処理を繰り返し、全てのＣＲＵＤ情報について処理を終えたら終了する。このＣＲＵＤ情報バリデーションに関し、１つのテーブルに対する通知情報のパターンの一例を図１２に示す。
【０１０３】
図１１は、１つのテーブルに対して図９のステップ８１２のバリデーション対象の前提ビジネスプロセスのＣＲＵＤ情報集計のパターンを示す。なお、前提ビジネスプロセスに

さらに前提ビジネスプロセスがあるパターンも含まれる。以下、項番ごとにパターンの例を説明する。なお、ＣＲＵＤ情報の集計結果は、例えば、メモリ５０２に格納される。

【０１０４】

図１１において、ＣＲＵＤ情報の集計結果は、パターンを識別する項番１１０１と、１番目のビジネスプロセスでの操作１１０２と、２番目のビジネスプロセスでの操作１１０３と、３番目のビジネスプロセスでの操作１１０３と、全体の集計結果１１０５からひとつのエントリが構成される。

【０１０５】

項番１は、新規契約などテーブルの追加系の操作のビジネスプロセスを想定し、集計結果は「Ｃ（作成）」とする。

【０１０６】

項番２は、解約などのテーブルの削除系の操作のビジネスプロセスを想定し、集計結果は「－（なし）」とする。

【０１０７】

項番３は、テーブルの削除の後にテーブルを作成する変更などの更新系のビジネスプロセスを想定し、集計結果は「Ｃ（作成）」とする。

【０１０８】

項番４は、テーブルの削除が連続するパターンで最終的に作成されているため、項番３と同じく更新系のビジネスプロセスを想定し、集計結果は「Ｃ（作成）」とする。

【０１０９】

項番５は、１番目はテーブルの追加系のビジネスプロセスで、２番目は削除系のビジネスプロセスと想定し、集計結果は「－（なし）」とする。

【０１１０】

項番６は、１番目がテーブルの追加系のビジネスプロセスで、２番目と３番目で更新系のビジネスプロセスを想定し、集計結果は「Ｃ（作成）」とする。

【０１１１】

項番７は、１番目のテーブルのビジネスプロセスは追加系、２番目と３番目で更新系のビジネスプロセスを想定し、集計結果は「Ｃ（作成）」とする。

【０１１２】

なお、以上は前提ビジネスプロセスが、３つのビジネスプロセス（またはモジュール）で構成される例を示すが、これに限定されるものではなく、任意の数のビジネスプロセスまたはモジュールで構成することができる。

【０１１３】

なお、上記では操作情報を集計するデータ集合の一例としてテーブルを扱う例を示したが、データ集合としては配列や変数を扱う場合も同様である。

【０１１４】

図１２は、図１１の集計結果とバリデーション対象のビジネスプロセスの集計結果のパターンと通知情報の関係を示す。以下、項番ごとに想定パターンの例を示す。なお、バリデーションのパターンに対応する通知情報は、例えば、メモリ５０２に格納される。

【０１１５】

図１１において、バリデーションのパターンに対応する通知情報は、パターンを識別する項番１２０１と、図１１の集計結果１１０５を格納する集計結果１２０２と、１番目のビジネスプロセスでの集計結果１２０３と、２番目以降のビジネスプロセスでの集計結果１２０４と、集計結果のパターンに対応する通知情報１２０５からひとつのエントリが構成される。

【０１１６】

項番１は、通知情報の設定対象となるビジネスプロセスが新規契約などテーブルの追加系のビジネスプロセスを想定できるため通知情報は「－（なし）」とする。ＣＲＵＤ集計２番目以降（１２０４）は、Ｒ（参照）またはＵ（更新）の連続は同様のパターンとする。

【０１１７】
項番２は、前提ビジネスプロセスが新規契約などテーブルの追加系のパターンで、通知情報の設定対象となるビジネスプロセスが、追加契約などテーブルの追加系のビジネスプロセスを想定できるため通知情報は「－（なし）」とする。ＣＲＵＤ集計２番目以降（１２０４）は、Ｒ（参照）またはＵ（更新）の連続は同様のパターンとする。
【０１１８】
項番３は、前提ビジネスプロセスが新規契約などテーブルの追加系のパターンで、通知情報の設定対象となるビジネスプロセスが解約などテーブルの削除系のビジネスプロセスを想定できるため通知情報は「－（なし）」とする。
【０１１９】
項番４は、前提ビジネスプロセスが新規契約などテーブルの追加系のパターンで、通知情報の設定対象となるビジネスプロセスがテーブルの変更などの更新系のビジネスプロセスを想定できるため通知情報は「－（なし）」とする。
【０１２０】
項番５は、項番３と同様のビジネスプロセスのパターンだが、テーブルの削除後にＲ（参照）・Ｕ（更新）・Ｄ（削除）を行うが、論理削除を対象とすることが想定されるため通知情報は「ワーニング」とする。
【０１２１】
項番６と項番７は、通知情報の設定対象となるビジネスプロセスで、テーブルのＣ（作成）の後にＤ（削除）を行うパターンであり、一つのビジネスプロセスとして想定できないパターンのため通知情報は「エラー」とする。
【０１２２】
以上のように、ビジネスプロセスの入出力情報のバリデーションとＣＲＵＤ情報のバリデーションにより、ビジネスプロセスを構成するプロセスの入出力情報の検証と、使用するデータ集合に対する操作を検証し、検証結果が不正な場合、換言すれば想定外の操作についてはエラーの通知情報を生成し、条件によっては実行可能な操作についてはワーニングの通知情報を生成する。
【０１２３】
これにより、一部のビジネスプロセスの変更によって、共通のプロセス（２１７）の前提条件に変化が生じた場合に、アノテーション解析部４０７の入出力情報のバリデーションとＣＲＵＤ情報のバリデーションにより、エラーやワーニングの通知情報があった場合に、修正作業を行えば良く、疎結合のビジネスプロセスの開発（あるいは修正）作業を効率良く実施することが可能となる。
【０１２４】
そして、通知情報がエラーの場合には、実行バイナリファイルの生成を禁止することで、無駄な単体試験の実行などを回避でき、疎結合のビジネスプロセスの開発期間を短縮し、開発に要する労力を低減することが可能となる。
【０１２５】
図１３は、図７の自動生成部４０９で行われるステップ６０９の自動生成処理の一例を示すフローチャートである。
【０１２６】
まずステップ９０３では、自動生成対象がビジネスプロセスの場合、対象のビジネスプロセス内のプロセスの実行順序を表したファイル（フロー定義ファイル）を生成し、自動生成対象がプロセスの場合、当該プロセス内のモジュールの実行順序を表したファイル（フロー定義ファイル）を生成する処理を行い、ステップ９０４へ遷移する。
【０１２７】
次に、ステップ９０４では、自動生成部４０９が、対象となるビジネスプロセスまたはプロセスの入出力管理用の個別プログラムのソースの生成処理を行い、自動生成処理を終了する。なお、本処理はビジネスプロセス・プロセス・モジュールのｊａｖａインターフェースファイル毎に起動される。また、生成された自動生成ソースは、メモリ５０２また

はストレージ５０６に格納される。あるいは、生成された自動生成ソースをソースファイル５０９に付加するようにしても良い。

【０１２８】

上記処理によって、ビジネスプロセス内のプロセスの実行順序またはプロセス内のモジュールの実行順序を表したフロー定義ファイルに基づいて、ビジネスプロセスまたはプロセスの入出力管理用の個別プログラムとしての自動生成ソースを生成することができる。自動生成ソースは、ソースファイル５０９に記述された操作対象のテーブルなどのデータ集合が記述される。

【０１２９】

図７で示したように、実行バイナリファイル作成部４０５では、ソースファイル５０９の実行バイナリファイル５１０に加えて、自動生成ソースに記述された操作対象のテーブル等のデータ集合を加えて生成する。

【０１３０】

したがって、アプリケーション開発端末１０５では、図２のステップ１２５の単体試験では、開発者が予めテーブルを用意することなく、コンパイルによって処理に必要なテーブルなどが自動的に生成され、実行バイナリファイル５１０を迅速に実行することが可能となるのである。

【０１３１】

例えば、図３Ｃに示したビジネスプロセス２０１を構成するプロセス２１１では、モジュール２２５が処理２２６でテーブルＡを作成する。そして、異なるプロセス２１８でテーブルＡが参照される。ここで、自動生成部４０９は、図１２に示したビジネスプロセスのＣＲＵＤ情報集計のパターンのうち項番１と同様になるため、自動生成ソースに、プロセス２１８が参照するテーブルＡを生成するソースコードを付加する。

【０１３２】

そして、図２で示したように、アプリケーション開発端末１０５でコンパイル１２３を行うと、ビジネスプロセスに応じた実行バイナリファイルと、ビジネスプロセスで入出力するデータとしてのテーブルＡが生成される。これにより、アプリケーション開発端末１０５を操作する開発者は、テーブルＡを手動で用意することなく、単体試験１２５を迅速に行うことが可能となる。

【０１３３】

＜まとめ＞

以上のように、ビジネスプロセスと、ビジネスプロセスを構成するプロセスやモジュールを図４Ａ～図４Ｄで示したようなｊａｖａインターフェースファイルとし、これらのｊａｖａインターフェースファイルに入出力情報とＣＲＵＤ情報をアノテーションを用いて定義する。そして、アプリケーション開発端末１０５またはリポジトリサーバ１０１のコンパイラ（コンパイル実行部４０２、コンパイル部４０３）は、入出力情報とＣＲＵＤ情報を付加されたインターフェースファイルを受け付ける。そして、コンパイル部４０３は、アノテーション解析部４０７としてｊａｖａコンパイラの拡張機能であるアノテーションプロセッサを有し、当該アノテーション解析部４０７でインターフェースファイルを解析して入出力情報とＣＲＵＤ情報のバリデーションを実施する。そして、入出力情報のバリデーションとＣＲＵＤ情報のバリデーションによる検証結果が正当であれば、ソースの自動生成と、実行バイナリファイルの生成を実行する。一方、アノテーション解析部４０７は、入出力情報のバリデーションまたはＣＲＵＤ情報のバリデーションによる検証結果が不正であれば、ソースの自動生成と、実行バイナリファイルの生成を禁止し、コンソール部４０１へ通知情報を出力する。

【０１３４】

以上により、ビジネスプロセスを構成するプロセスやモジュールの入出力情報の検証と、データ集合に対する操作を検証し、検証結果が不正な操作についてはエラーの通知情報を生成し、条件によっては実行可能な操作についてはワーニングの通知情報を生成する。

【０１３５】

これにより、一部のビジネスプロセスの変更によって、共通のプロセスの前提条件に変化が生じた場合、アノテーション解析部４０７の入出力情報のバリデーションとＣＲＵＤ情報のバリデーションにより、ソースファイル５０９の解析でエラーやワーニングの通知情報があった場合に、修正作業を行えば良く、疎結合のビジネスプロセスの開発（あるいは修正）作業を効率良く実施することが可能となる。特に、従来では、結合試験や総合試験で発覚していたソースファイルの修正漏れや相違を、アプリケーション開発端末１０５またはビルドサーバ１０２でのコンパイル時のソースファイル解析段階で検出できるため、従来例に比してより早い段階でソースファイル５０９のバグ等を摘出できる。

【０１３６】

そして、通知情報がエラーの場合には、実行バイナリファイルの生成を禁止（あるいはコンパイルの中止）することで、無駄な単体試験の実行などを回避でき、疎結合のビジネスプロセスの開発期間を短縮し、開発に要する労力を低減することが可能となる。

【０１３７】

さらに、ビジネスプロセス内のプロセスの実行順序またはプロセス内のモジュールの実行順序を表したフロー定義ファイルに基づいて、ビジネスプロセスまたはプロセスの入出力管理用の個別プログラムとしての自動生成ソースを生成することができる。これにより、実行バイナリファイル作成部４０５では、ソースファイル５０９の実行バイナリに加えて、自動生成ソースに記述された操作対象のテーブル等を加えて実行バイナリファイル５１０を生成する。そして、アプリケーション開発端末１０５では、図２のステップ１２５の単体試験等では、開発者が予めテーブルを用意することなく、実行バイナリファイル５１０を実行することで、処理に必要なテーブルなどが自動的に生成されるのである。これにより、疎結合のビジネスプロセスの開発効率を向上させることが可能となる。なお、通知情報がエラーの場合には、自動生成を禁止することで、無駄な出力を生成するのを防止できる。

【０１３８】

なお、本発明において説明した計算機等の構成、処理部及び処理手段等は、それらの一部又は全部を、専用のハードウェアによって実現してもよい。

【０１３９】

また、本実施例で例示した種々のソフトウェアは、電磁的、電子的及び光学式等の種々の記録媒体（例えば、非一時的な記憶媒体）に格納可能であり、インターネット等の通信網を通じて、コンピュータにダウンロード可能である。

【０１４０】

また、コンパイラの各機能を実現するプログラム、テーブル等の情報は、ストレージ５０６や不揮発性半導体メモリ、ハードディスクドライブ、ＳＳＤ（Ｓｏｌｉｄ　Ｓｔａｔｅ　Ｄｒｉｖｅ）等の記憶デバイス、または、ＩＣカード、ＳＤカード、ＤＶＤ等の計算機読み取り可能な非一時的データ記憶媒体に格納することができる。

【０１４１】

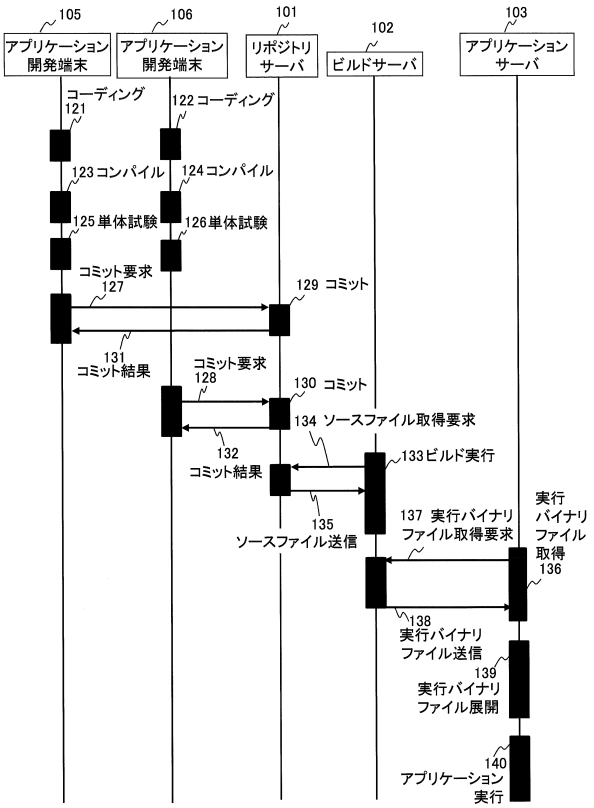
また、本発明は上記した実施例に限定されるものではなく、様々な変形例が含まれる。例えば、上記した実施例は本発明をわかりやすく説明するために詳細に説明したものであり、必ずしも説明した全ての構成を備えるものに限定されるものではない。

【符号の説明】

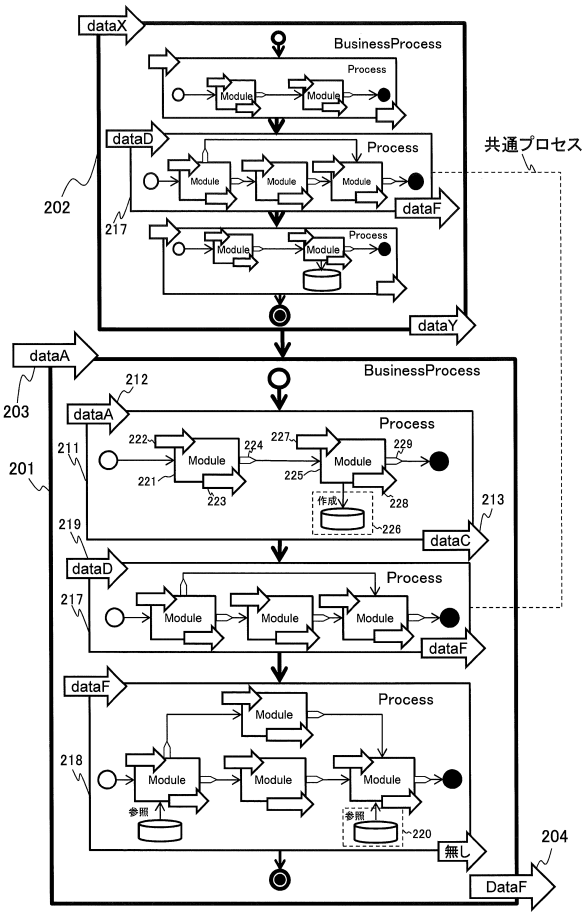
【０１４２】

１０２　ビルドサーバ
１０５、１０６　アプリケーション開発端末
４０７　アノテーション解析部
４０８　エラー通知部
４０９　自動生成部
５０１　ＣＰＵ
５０２　メモリ
５０６　ストレージ

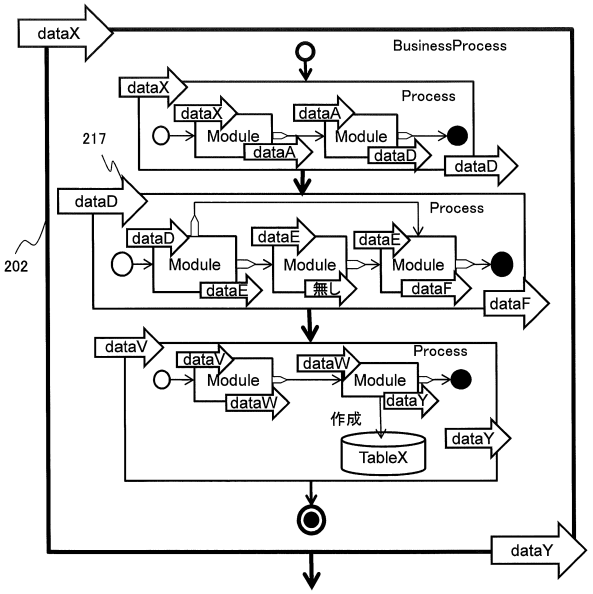
【図２】



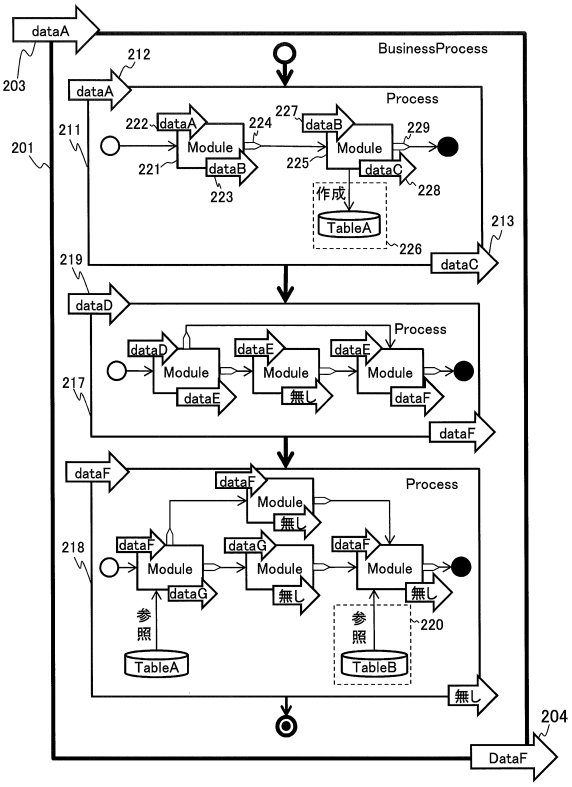
【図３Ａ】



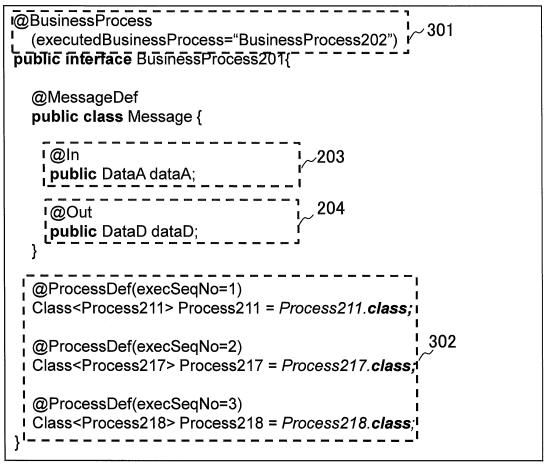
【図３Ｂ】



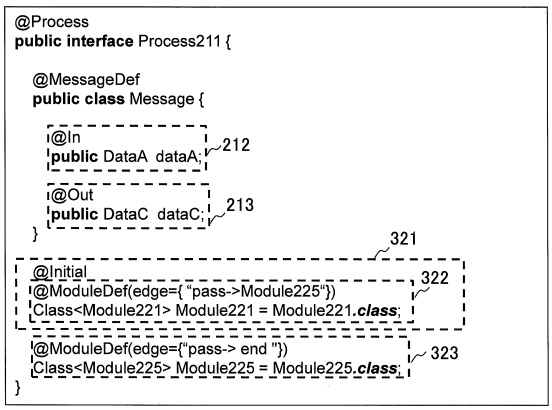
【図３Ｃ】



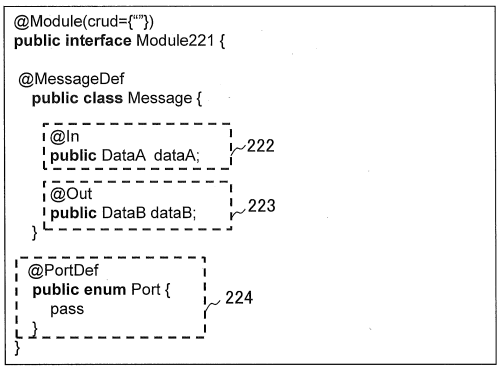
【図４Ａ】



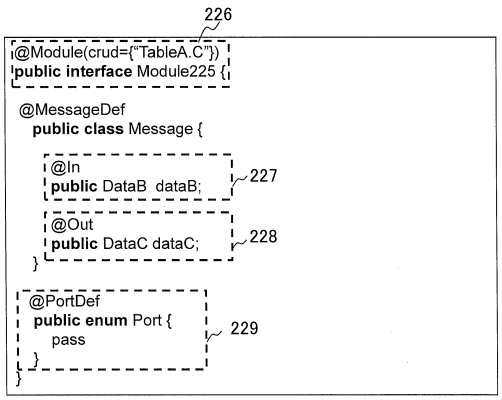
【図４Ｂ】



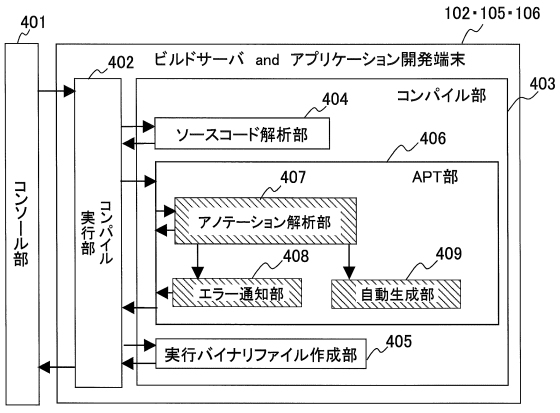
【図４Ｃ】



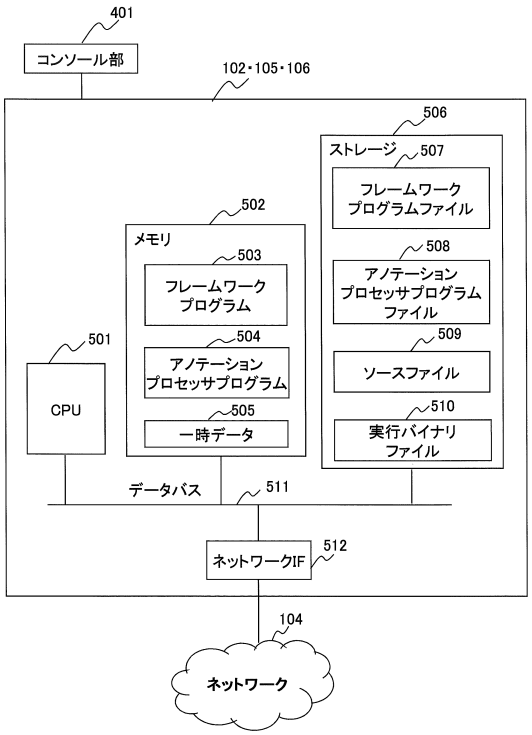
【図４Ｄ】



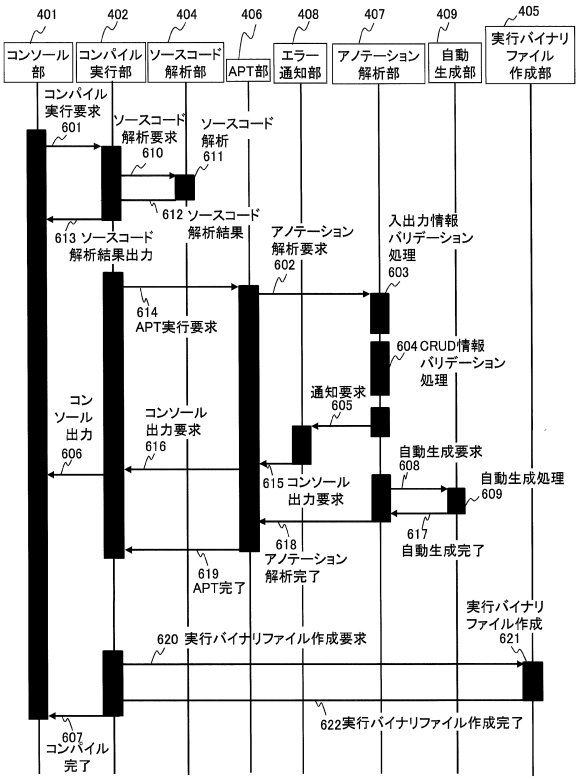
【図５】



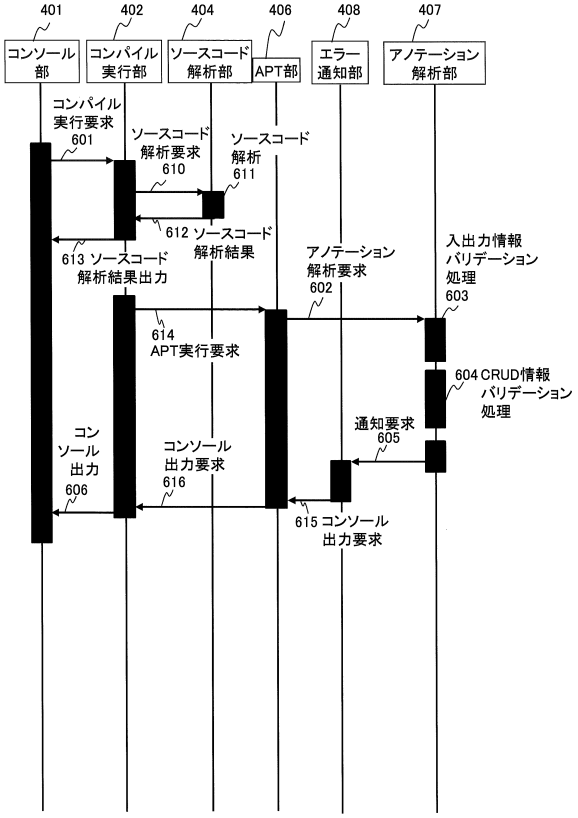
【図６】



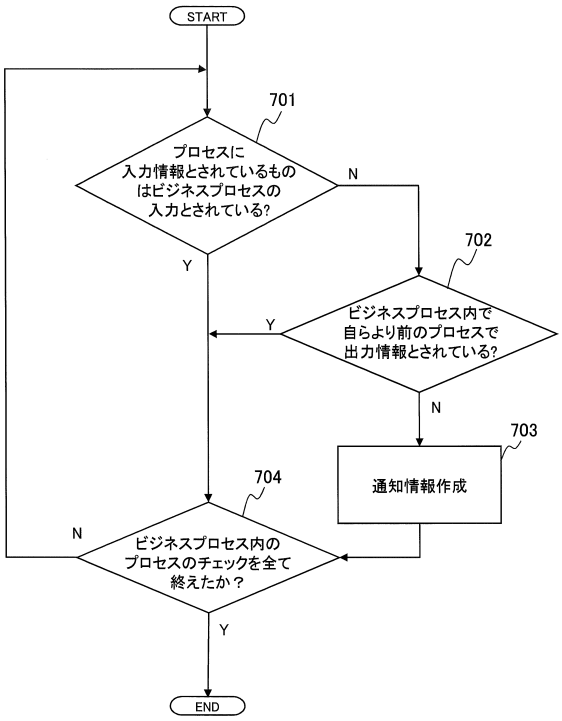
【図７】


401
コンソール部
402
コンパイル実行部
404
ソースコード解析部
406
APT部
408
エラー通知部
407
アノテーション解析部
409
自動生成部
405
実行バイナリファイル作成部
コンパイル実行要求 601
ソースコード解析要求 610
ソースコード解析 611
612 ソースコード解析結果
613 ソースコード解析結果出力
アノテーション解析要求 602
入出力情報バリデーション処理 603
614 APT実行要求
604 CRUD情報バリデーション処理
通知要求 605
コンソール出力 606
コンソール出力要求 616
615 コンソール出力要求
自動生成要求 608
自動生成処理 609
617 自動生成完了
618 アノテーション解析完了
619 APT完了
実行バイナリファイル作成 621
620 実行バイナリファイル作成要求
622 実行バイナリファイル作成完了
607 コンパイル完了

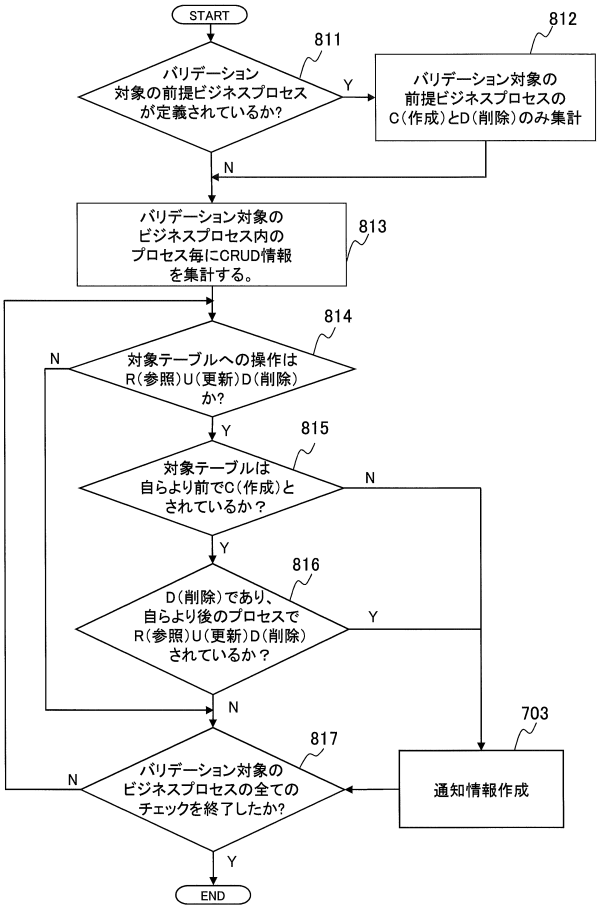
【図８】


401
コンソール部
402
コンパイル実行部
404
ソースコード解析部
406
APT部
408
エラー通知部
407
アノテーション解析部
コンパイル実行要求 601
ソースコード解析要求 610
ソースコード解析 611
612 ソースコード解析結果
613 ソースコード解析結果出力
アノテーション解析要求 602
入出力情報バリデーション処理 603
614 APT実行要求
604 CRUD情報バリデーション処理
通知要求 605
コンソール出力 606
コンソール出力要求 616
615 コンソール出力要求

【図９】


START
701
プロセスに入力情報とされているものはビジネスプロセスの入力とされている?
N
Y
702
ビジネスプロセス内で自らより前のプロセスで出力情報とされている?
Y
N
703
通知情報作成
704
ビジネスプロセス内のプロセスのチェックを全て終えたか？
N
Y
END

【図１０】


START
811
バリデーション対象の前提ビジネスプロセスが定義されているか?
Y
812
バリデーション対象の前提ビジネスプロセスのC(作成)とD(削除)のみ集計
N
バリデーション対象のビジネスプロセス内のプロセス毎にCRUD情報を集計する。
813
814
対象テーブルへの操作はR(参照)U(更新)D(削除)か?
N
Y
815
対象テーブルは自らより前でC(作成)とされているか？
N
Y
816
D(削除)であり、自らより後のプロセスでR(参照)U(更新)D(削除)されているか？
Y
N
703
通知情報作成
817
バリデーション対象のビジネスプロセスの全てのチェックを終了したか?
N
Y
END

【図１１】

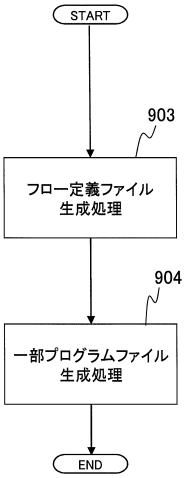
1101 項番	1102 1番目	1103 2番目	1104 3番目	1105 集計結果
1	C(作成)	-	-	C(作成)
2	D(削除)	-	-	-(なし)
3	D(削除)	C(作成)	-	C(作成)
4	D(削除)	D(削除)	C(作成)	C(作成)
5	C(作成)	D(削除)	-	-(なし)
6	C(作成)	C(作成)	D(削除)	C(作成)
7	C(作成)	D(削除)	C(作成)	C(作成)

【図１２】

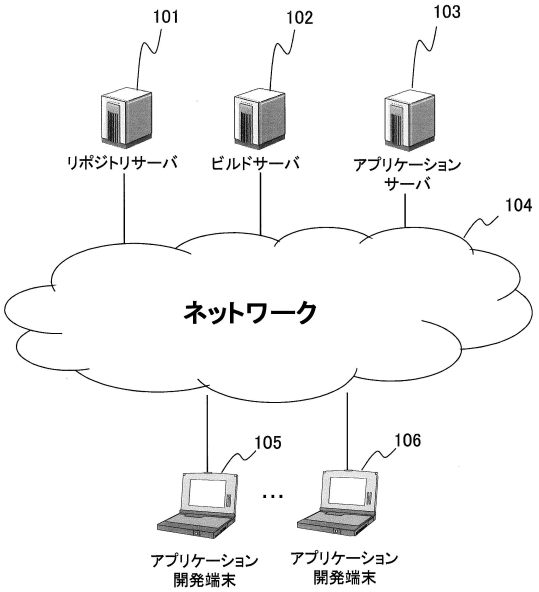
項番	図11の集計結果	対象ビジネスプロセスのプロセス毎CRUD集計		通知情報
		1番目	2番目	
1	-(なし)	C(作成)	-	-(なし)
2	C(作成)	C(作成)	-	-(なし)
3	C(作成)	D(削除)	-	-(なし)
4	C(作成)	D(削除)	C(作成)	-(なし)
5	C(作成)	D(削除)	R(参照) or U(更新) or D(削除)	ワーニング
6	-(なし)	C(作成)	D(削除)	エラー
7	C(作成)	C(作成)	D(削除)	エラー

1201 1202 1203 1204 1205

【図１３】



【図１】



フロントページの続き

(72)発明者　木村　広
神奈川県横浜市戸塚区戸塚町２１６番地　株式会社日立製作所　通信ネットワーク事業部内

審査官　石川　亮

(56)参考文献　特開２００７－２６５０８９（ＪＰ，Ａ）
特開２００１－２３６２１５（ＪＰ，Ａ）
特開２００１－３４４１１３（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　　９／４５
Ｇ０６Ｆ　１１／３６