(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2008-20972
(P2008-20972A)
(43)公開日　平成20年1月31日(2008.1.31)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 11/36	(2006.01)	G06F	9/06	620M	5B176
G06F 9/44	(2006.01)	G06F	9/06	620A	

審査請求 未請求　請求項の数 4　OL　（全 10 頁）

(21)出願番号　特願2006-189954 (P2006-189954)
(22)出願日　平成18年7月11日 (2006.7.11)

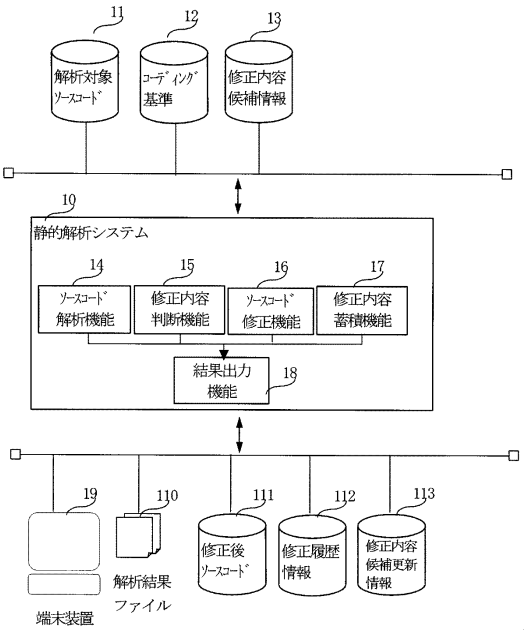
(71)出願人　000005108
株式会社日立製作所
東京都千代田区丸の内一丁目６番６号
(74)代理人　100100310
弁理士　井上　学
(72)発明者　河崎　文雄
神奈川県川崎市幸区鹿島田８９０番地　株式会社日立製作所情報・通信グループ内

Fターム(参考)　5B176 DD04　EC03

(54)【発明の名称】ソフトウエア解析システム

(57)【要約】
【課題】
従来、作成されたソフトウエアについての解析が十分にできていなかった。
【解決手段】
ソフトウェア開発においてソースコードがコーディング基準に準拠しているかを静的に解析し、その解析結果から如何にしてソースコードがコーディング基準に準拠するかを判定し、その判定結果からソースコードを修正し、修正履歴を残し、プログラム開発時の作業効率および品質向上の効果をもたらすシステムを提供する。
【選択図】　図１

図１



【特許請求の範囲】
【請求項１】
作成されたソフトウエアに対する解析処理を実行するソフトウエア解析システムにおいて、
前記ソフトウエアであるソースコード、当該ソースコードのコーディング基準および当該コーディング基準に前記ソフトウエアが違反していることが検知された場合の修正内容候補情報と互いに関連付けて格納する手段と、
前記ソースコードと前記コーディング基準を比較する手段と、
前記比較の結果に基づいて、前記ソースコードが前記コーディング基準に違反していないかどうかを解析する手段と、
前記解析の結果および前記修正内容候補情報から前記ソースコードに対する修正内容を特定する手段と、
前記特定の結果を出力する手段とを備えたことを特徴とするソフトウエア解析システム。
【請求項２】
請求項１に記載のソフトウエア解析システムにおいて、
前記特定された修正内容候補情報に従ってソースコードをコーディング基準に準拠するように修正する手段を設け、
前記出力する手段は、修正したソースコードを出力することを特徴とするソフトウエア解析システム。
【請求項３】
請求項２に記載のソフトウエア解析システムにおいて、
修正したソースコードの修正内容の蓄積であるソースコード修正履歴情報と修正内容候補情報の更新履歴情報とを格納する手段をさらに設けることを特徴とするソフトウエア解析システム。
【請求項４】
請求項３に記載のソフトウエア解析システムにおいて、
前記修正する手段での修正に基づいて、前記格納する手段に格納された前記修正内容候補情報を更新する手段と、
前記修正されたソースコードに対して修正を行った場合に修正履歴と修正内容候補情報を更新する手段とをさらに備えたことを特徴とするソフトウエア解析システム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、作成されたソフトウエアを解析する技術に関する。その中でも特に、ソフトウェア開発において、ソースコードを実行することなく静的に解析し、コーディングルールに違反している箇所、および不良の原因となる箇所を検出し、修正を行う静的解析を行う技術に関するものである。
【背景技術】
【０００２】
ソフトウェア開発において、ソースコードの記述誤りを検出するシステムとしては特許文献１の方法が提案されている。これは、ソースコードの静的解析を行う静的解析装置と、静的解析結果を所定単位からなる解析対象毎に分けるとともに解析対象に解析対象を識別するために識別情報を負荷して格納し、静的解析結果を解析対象毎に識別情報に基づいて分析するシステムである。
【０００３】
【特許文献１】特開２００５－２０２４９４号公報
【発明の開示】
【発明が解決しようとする課題】
【０００４】

上記従来の技術では、ソースコードの静的解析結果を格納し分析することは可能であるが、ソースコードの修正作業はソフトウェア開発者の手作業で実施されることから、修正の作業工数を削減すること、および修正誤りや修正漏れを防ぐことはできないという問題があった。

【０００５】

本発明は上記の課題を解決し、ソフトウェア開発においてソースコードの記述誤りを検出するために静的解析を行う際に、静的解析結果を出力すると共に、修正案を示すこと、修正案に従って修正を行うこと、および修正結果を蓄積することにより、修正作業の工数を削減して効率を向上し、修正誤りと修正漏れを防いで品質を向上すること目的とする。

【課題を解決するための手段】

【０００６】

上記課題を解決するために、本発明では、ソフトウエアであるソースコード、当該ソースコードのコーディング基準および当該コーディング基準に前記ソフトウエアが違反していることが検知された場合の修正内容候補情報と互いに関連付けて記憶しておき、これに基づいて、作成したソースコードの確認などを実行する。より詳細には、静的解析の基準となるコーディング基準と、ソースコードがコーディング基準に違反していないかどうかを静的に解析する手段と、コーディング基準に違反したソースコードをコーディング基準に準拠するように修正を行うために必要な修正内容候補情報と、コーディング基準に違反したソースコードをコーディング基準に準拠するようにソースコードを修正する手段と、ソースコードを修正した履歴を蓄積する手段と、解析結果と修正内容候補情報とソースコード修正結果と修正履歴を出力する手段と、修正内容にしたがって修正内容候補情報を更新する手段とを設ける。

【０００７】

本発明の静的解析によりコーディング基準に違反しているかどうかの判定は、次のような手順で行う。ソフトウェア開発者は、静的解析対象のソースコードを解析対象プログラムファイルに登録する。解析対象プログラムファイルは、プログラム言語の種類によらない言語非依存のファイルである。コーディング基準ファイルおよび修正内容候補情報ファイルは、プログラム開発開始時に格納しておく。

【０００８】

次に、プログラム解析機能が解析対象プログラムおよびコーディング基準を読み込み、コーディング内容がコーディング基準と合致しているかを解析して、解析結果を解析結果ファイルに格納する。解析結果は画面に表示してもよい。解析結果ファイルの内容は、プログラム名と行番号とコーディングエラー内容といった項目から構成される。

【０００９】

コーディング基準に違反したソースコードの修正は、次のように行う。プログラム修正内容判断機能が解析結果ファイルおよび修正内容候補情報ファイルを読み込み、コーディング基準に違反しているソースコードをどのように修正すればコーディング基準に準拠するかを判定し、修正情報ファイルに出力する。修正内容候補情報ファイルの内容は、使用禁止命令チェックに対してはそれに替わる代替命令、命令の組合せの妥当性チェックに対しては使用命令との組合せが必要な命令といった項目から構成される。

【００１０】

修正情報ファイルのレコード形式は、チェック項目に依存することとなる。例えば禁止命令チェックである場合は、禁止命令、禁止命令に替わる代替命令、変更必須かどうかのフラグといった項目から構成される。命令の組合せの妥当性チェックである場合には命令語、対となる命令語、修正必須かどうかを示すフラグといった項目から構成される。

上記までの処理により、静的解析システムは、ソフトウェア開発時にソースコードを静的に解析して、ソースコードがコーディング基準に違反している箇所に対する修正案を判定して出力する。

【００１１】

修正内容情報から、ソースコードの修正を行う場合の処理は以下の手順で実施する。

まず修正情報の内容について、ソースコードの変更を行うかどうかを選択する。この選択は静的解析処理の実行時にソースコード修正の設定にすることも、静的解析結果を画面または解析結果ファイルの内容から確認を行ってから選択を行うことも可能とする。コーディング基準に違反したコーディングがあり、修正を必要とするソースコードに対しては、静的解析処理の実行時にツールで修正を行うように設定した場合には、プログラム修正機能は、修正内容情報に従ってソースコードの修正を行い、修正後のソースコードを修正後プログラムファイルに出力する。修正内容は、その履歴を修正履歴情報ファイルに蓄積する。ソースコードの修正内容は、チェック内容に依存することとなる。例えば禁止命令チェックのエラー修正である場合には、使用禁止命令を代替命令に置き換える。命令の組合せの妥当性チェックのエラー修正である場合には、チェック対象命令語と対となる命令語が妥当となるように追加または修正を行う。ソースコードの修正作業は、静的解析処理の実行時にエラーを修正する機能を使用せずに、修正内容候補情報にない修正を行うことも可能とする。静的解析処理の実行時にエラーを修正する設定により修正したソースコードと、修正内容候補情報にない修正を行ったソースコードは、修正箇所にコメントを付与することにより、両社区別することを可能とする。修正内容候補情報にない修正を行った履歴は、修正内容候補情報にも追加登録し、修正内容候補を増やすことにより、次回の静的解析時の修正内容候補にバリエーションを増やすことも可能とする。

【００１２】

上記までの処理により、静的解析システムは、ソースコードの静的解析結果および修正内容候補から、ソースコードの修正を行って出力する。

【発明の効果】

【００１３】

以上、本発明の静的解析システムによれば、ソフトウェア開発のコーディング工程において、静的解析によりコーディング基準に違反している箇所およびプログラムの不良原因となる箇所を摘出し、解析結果に従ってソースコードの修正候補を出力することが可能であるため、静的解析結果の分析作業の効率を向上させる効果がある。また、プログラム修正機能により、コーディング基準に違反している箇所を修正することが可能であるため、ソースコードの手作業による修正を簡略化して効率を向上させ、修正漏れや修正間違いを無くして品質を向上させる効果がある。さらに、修正履歴情報を蓄積し、修正内容候補情報にフィードバックしていくことが可能であるため、修正内容候補のバリエーションを複数持たせ、修正の精度を向上させる効果がある。

【発明を実施するための最良の形態】

【００１４】

以下、本発明の実施の一形態を図面を使用して説明する。図１は本発明の静的解析システムの実施例の構成を示す機能ブロック図である。

【００１５】

図１において、静的解析システム１０は本システムの全体を示しており、解析対象ソースコードファイル１１、コーディング基準ファイル１２、修正内容候補情報ファイル１３、ソースコード解析機能１４、修正内容判断機能１５、ソースコード修正機能１６、修正内容蓄積機能１７、結果出力機能１８、端末装置１９、解析結果ファイル１１０、修正後ソースコード１１１、修正履歴情報１１２、修正内容候補更新情報１１３から構成される。

【００１６】

解析対象ソースコードファイル１１は、静的解析の対象となるプログラムソースである。コーディング基準ファイル１２は、静的解析のチェックの基準となるルールである。修正内容候補情報ファイル１３は、コーディング基準に準拠していないコードを、コーディング基準に準拠するように修正するためのコーディング例を格納しておく。ソースコード解析機能１４は、解析対象プログラムがコーディング基準に準拠しているかどうかを判定する。修正内容判断機能１５は、解析結果と修正内容候補情報から、ソースコードがコーディング基準に準拠するような修正内容候補を判定する。ソースコード修正機能１６は、

修正内容候補に合わせて、ソースコードの修正を行う。修正内容蓄積機能１７は、ソースコードを修正した履歴を蓄積する。結果出力機能１８は、解析結果、修正内容候補、ソースコード修正の結果を出力する。端末装置１９は、解析結果や修正内容候補を画面で確認するために使用する。解析結果ファイル１１０は、静的解析結果である。

【００１７】

修正後ソースコード１１１は、コーディング基準に準拠するように修正されたソースコードである。修正履歴情報１１２は、修正履歴である。修正内容候補更新情報１１３は、修正内容候補にない修正をソースコードに対して行った場合に、修正内容候補として新たに登録し、次回の修正内容候補とする。なお、コーディング基準ファイル１２は、開発するプログラムの特性に合わせてカスタマイズを行うことも可能とする。上記までの過程で、静的解析システムは、ソースコードがコーディング基準に違反している箇所の修正案を示し、ソースコードの修正を行い、修正結果を蓄積する。

【実施例１】

【００１８】

以下、本実施例の動作について説明する。図２はソースコードの一部である。ここではCOBOL言語を例としているが、静的解析システムは言語に依存しないため、多言語に対応可能である。２１は行番号を示し、２２は命令語を示す。

【００１９】

図３は、コーディング基準ファイル１２のデータ構成例である。コーディング基準の各レコードは、チェック項目ID、チェック内容といった項目から構成される。本発明では図１が示すように解析対象ソースコード１１およびコーディング基準１２を入力として、ソースコード解析機能１４がソースコードがコーディング基準に違反していないかどうかを解析してコーディング解析結果ファイル１１０に結果を出力する。図４および図５は修正内容候補情報１３のデータ構成例である。図４は使用禁止命令の一覧であり、コーディング基準とともに予め使用を禁止する命令をおよびそれに替わる代替命令を登録しておく。図５は命令語の組合せの一覧であり、命令語の組合せの妥当性をチェックするために登録しておく。図６にプログラム解析機能および修正候補出力のフローチャートを示す。処理ブロック６１では、解析対象ソースプログラムがコーディング基準に違反していないかを解析する。チェックＩＤの００１の禁止文格納ファイルに指定した命令語を使用しないことのチェックにおいては、まず図４の所定以上の文字数で構成されるかを判定し、所定数以上の場合には禁止命令の一覧に存在する禁止命令かどうかを判定する。

【００２０】

この図６の処理は、図１の静的解析システム１０の各機能（１４～１８の少なくとも１つの機能）により実現される。また、これらの機能は、メモリ（図示せず）に展開されたプログラムに従ったＣＰＵ（図示せず）により実現するよう構成してもよい（図７の内容も同様である）。以下、処理の詳細を説明する。

【００２１】

まず、ステップ６１において、静的解析の実行を開始する。そして、ステップ６２において、この結果を出力する。

【００２２】

次に、ステップ６３において、ステップ６２でのソースコードにコーディング違反がないか、コーディング基準１２と解析対象ソースコード１１を比較して判定する。この結果，基準違反があれば、ステップ６４に進む。

【００２３】

ステップ６４において、コーディング違反の修正候補の存在の有無を判定する。つまり、図４に示す使用禁止命令一覧の内容や図５に示命令語の組合せの一覧を用いて、代替命令や対となる命令があるかを検索する。この場合、ステップ６３での判定でチェックＩＤ（もしくは使用禁止命令や使用命令など）を特定し、これを用いて図４や５のテーブルを検索し、代替命令等の有無を判定してもよい。

【００２４】

ステップ６４で「有」と判定された場合、ステップ６３で検索された代替命令等を修正内容情報として、出力する。なお、図４，５に示す内容は、コーディング基準１２と修正内容候補情報１３，］修正内容候補情報のいずれをも兼ねるものである。

【００２５】

図２のソースコードは、9100行目と9200行目および9300行目にて禁止命令である”PICTURE”命令を使用しており、図３のコーディング基準によると、チェックＩＤの００１に違反しているため、9100行目と9200行目および9300行目をコーディング基準違反として検出する。チェックＩＤの００２の命令語の組合せが妥当であることのチェックにおいては、まず図５の使用命令が存在するかどうかを判定し、それに続くソースコードに”END-”を含む命令が存在するかを判定し、次に他にも対となる命令があるかどうかを判定する。図２のソースコードは、20000行目から20500行目にて、”EVALUATE”命令の後に続く”WHEN”命令の後に、”WHEN OTHER”命令が無く、”END-EVALUATE”命令となっており、チェックＩＤの００２に違反しているため、20300行目をコーディング基準違反として検出する。処理ブロック６２では、コーディング基準違反を検出したソースコードの行番号、チェックID、エラー内容といった項目を出力する。出力は解析結果ファイル又は端末装置のどちらでも可能とする。

【００２６】

図８に、解析結果ファイルの出力例を示す。８１は、コーディング基準違反のあるソースコードの行番号を示す。８２は、解析したチェックIDを示す。８３は、エラー内容を示す。この例では、使用禁止命令チェックおよび命令語の妥当性チェックにてコーディング基準違反が検出されている。判断ブロック６３では、解析対象ソースコードにコーディング基準違反が検出されたかどうかを判断する。判断ブロック６４では、静的解析結果にエラーがあった場合に、コーディング基準違反の箇所がコーディング基準に合致するように修正を行うべき修正内容候補が修正内容候補情報ファイルに存在するかどうかを判断する。処理ブロック６５では、判断ブロック６４で存在を確認した修正内容を出力する。出力は修正内容ファイル又は端末装置のどちらでも可能とする。図９に、修正内容ファイルの出力例を示す。９１は、コーディング基準に違反したソースコードの行番号を示す。９２は、修正対象のコーディング内容を示す。９３は、コーディング基準に合致するように修正する内容を示す。この例では、9100行目と9200行目および9300行目にて禁止命令”PICTURE”命令を”PIC”命令に修正している。

【００２７】

さらに、”EVALUATE”命令の後に続く”WHEN”命令の後に、”WHEN OTHER”命令が無く、”END-EVALUATE”命令となっているため、”WHEN OTHER”命令を挿入し、”CONTINUE”命令を仮定している。

【００２８】

図７にプログラム自動修正機能のフローチャートを示す。判断ブロック７１では、静的解析機能にてコーディング基準違反が検出された場合に、自動修正を行う設定かどうかの判定を行う。図１０は自動修正実行選択画面の例である。この画面を端末装置に表示して、ＹＥＳボタンを押下しても、コーディング基準ファイルの33の修正方法の項目を参照して、自動修正になっているチェックIDを取得して自動修正の判定を行ってもよい。処理ブロック７２では、静的解析結果にエラーがあるソースコードを、コーディング基準に準拠するように、修正内容情報に従って修正する。処理ブロック７３では、修正したソースコードを修正後プログラムファイルに出力する。処理ブロック７４では、修正した結果を修正履歴情報に出力する。

【００２９】

図１１に修正後ソースコードの例を示す。１１１は行番号を示し、１１２は命令語を示す。修正後ソースコードは、図２のソースコードに対して、図９の修正内容が反映されている。

【００３０】

図１２に修正内容候補情報蓄積機能のフローチャートを示す。判断ブロック１２１では

、静的解析結果にエラーがあった場合に、自動修正を行う設定にしているかどうかの判定を行う。図１０の画面を端末装置に表示して、自動修正しない場合にはＮＯボタンを押下する。処理ブロック１２２では、静的解析結果にエラーがあるソースコードを、手作業によって修正する。処理ブロック１２３では、修正したソースコードを修正後プログラムファイルに出力する。処理ブロック１２４では、修正した結果を修正履歴情報に出力する。処理ブロック１２５では、手作業によって修正を行ったソースコードの内容を、修正内容候補情報ファイルに追加する。追加された修正内容候補情報は、次に静的解析ツールを実行する際に、新たな修正内容候補情報として参照される。

【図面の簡単な説明】

【００３１】

【図１】本発明に係る一実施形態の静的解析システムの構成を示す図である。

【図２】図１の解析対象ソースコードの例を示す。

【図３】図１のコーディング基準の例を示す。

【図４】図１の修正内容候補情報の使用禁止命令チェックの例を示す。

【図５】図１の修正内容候補情報の命令の妥当性チェックの例を示す。

【図６】静的解析処理のフローチャートを示す。

【図７】ソースコード修正処理のフローチャートを示す。

【図８】図１の解析結果ファイルの出力例を示す。

【図９】図１の修正内容ファイルの出力例を示す。

【図１０】自動修正実行選択画面の例を示す。

【図１１】図１の修正後ソースコードの例を示す。

【図１２】修正内容候補情報の更新処理のフローチャートを示す。

【符号の説明】

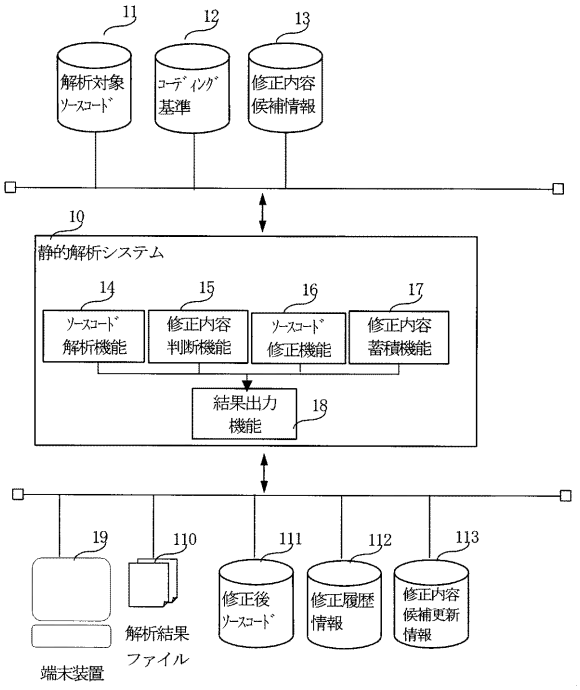
【００３２】

１０　静的解析システム
１１　解析対象ソースコード
１２　コーディング基準
１３　修正内容候補情報
１４　ソースコード解析機能プロジェクトカスタマイズファイル
１５　修正内容判断機能
１６　ソースコード修正機能
１７　修正内容蓄積機能
１８　結果出力機能
１９　端末装置
１１０　解析結果ファイル
１１１　修正後ソースコード
１１２　修正履歴情報
１１３　修正内容候補更新情報
２１　行番号
２２　命令語
３１　チェックＩＤ
３２　チェック内容
３３　修正方法
４１　チェックＩＤ
４２　使用禁止命令
４３　代替命令
５１　チェックＩＤ
５２　使用命令
５３　対となる命令
６１　静的解析実行ブロック

６２　解析結果出力ブロック
６３　コーディング基準違反判断ブロック
６４　修正内容候補存在判断ブロック
６５　修正内容出力ブロック
７１　自動修正設定判断ブロック
７２　ソースコード修正実行ブロック
７３　実行結果出力ブロック
７４　修正履歴蓄積ブロック
８１　行番号
８２　チェックＩＤ
８３　エラー内容
９１　行番号
９２　コーディング内容
９３　修正内容
１０１　自動修正実行選択画面
１１１　行番号
１１２　命令語
１２１　自動修正設定判断ブロック
１２２　ソースコード修正実行ブロック
１２３　実行結果出力ブロック
１２４　修正履歴蓄積ブロック
１２５　修正内容候補蓄積ブロック

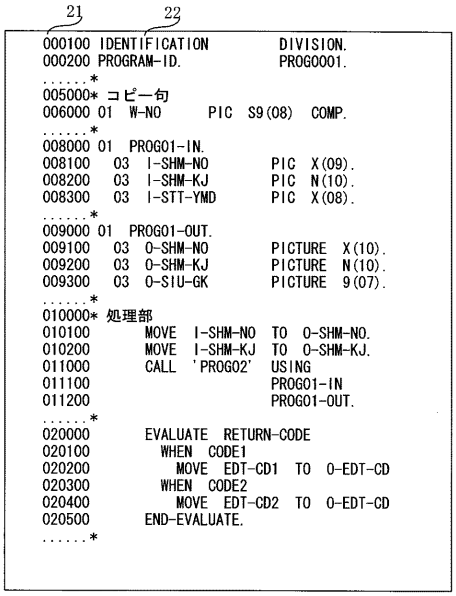
【図１】

図1



【図２】

図2



【図３】

図3

31 チェックID	32 チェック内容	33 修正方法
001	禁止文格納ﾌｧｲﾙに指定した命令語を使用しないこと	自動修正
002	命令語の組合せが妥当であること	自動修正
003	命令の内容が妥当であること(属性)	自動修正
004	命令の内容が妥当であること(桁数)	自動修正
005	命令の内容が妥当であること(論理演算)	自動修正

【図４】

図４

41 チェック ID	42 使用禁止命令	43 代替命令
001	PICTURE	PIC
001	COMPUTATIONAL	COMP
001	COMPUTATIONAL-3	COMP-3

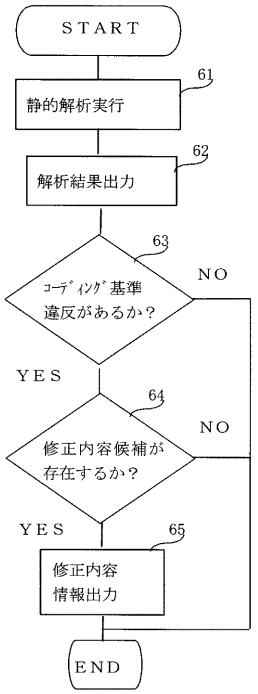
【図５】

図５

51 チェック ID	52 使用命令	53 対となる命令１	対となる命令２	対となる命令３
002	IF	THEN	ELSE	END-IF
002	EVALUATE	WHEN	WHEN OTHER	END-EVALUATE
002	PERFORM	END-PERFORM		

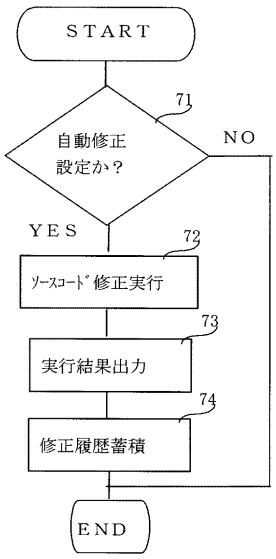
【図６】

図６



【図７】

図７



【図８】

図８

81 行番号	82 チェック ID	83 エラー内容
9100	001	使用禁止命令を使用しています。
9200	001	使用禁止命令を使用しています。
9300	001	使用禁止命令を使用しています。
20300	002	EVALUATE, WHEN, WHEN OTHER, END-EVALUATE が対になっていません。

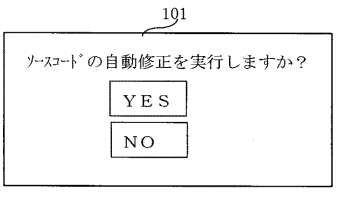
【図９】

図９

91 行番号	92 コーディング内容	93 修正内容
9100	03 O-SHM-NO PICTURE X(10).	03 O-SHM-NO PIC X(10).
9200	03 O-SHM-KJ PICTURE N(10).	03 O-SHM-KJ PIC N(10).
9300	03 O-SIU-GK PICTURE 9(07).	03 O-SIU-GK PIC 9(07).
20300 20400 20500	WHEN CODE2 MOVE EDT-CD2 TO O-EDT-CD END-EVALUATE.	WHEN CODE2 MOVE EDT-CD2 TO O-EDT-CD WHEN OTHER CONTINUE END-EVALUATE.

【図１０】

図１０



【図１１】

図１１

111 112

```
000100 IDENTIFICATION          DIVISION.
000200 PROGRAM-ID.             PROG0001.
......*
005000* コピー句
006000 01  W-NO       PIC  S9(08)  COMP.
......*
008000 01  PROG01-IN.
008100     03  I-SHM-NO          PIC  X(09).
008200     03  I-SHM-KJ          PIC  N(10).
008300     03  I-STT-YMD         PIC  X(08).
......*
009000 01  PROG01-OUT.
009100     03  O-SHM-NO          PIC  X(10).
009200     03  O-SHM-KJ          PIC  N(10).
009300     03  O-SIU-GK          PIC  9(07).
......*
010000* 処理部
010100       MOVE  I-SHM-NO  TO  O-SHM-NO.
010200       MOVE  I-SHM-KJ  TO  O-SHM-KJ.
011000       CALL  'PROG02'  USING
011100                       PROG01-IN
011200                       PROG01-OUT.
......*
020000       EVALUATE  RETURN-CODE
020100         WHEN  CODE1
020200           MOVE  EDT-CD1  TO  O-EDT-CD
020300         WHEN  CODE2
020400           MOVE  EDT-CD2  TO  O-EDT-CD
020410         WHEN  OTHER
020420           CONTINUE
020500       END-EVALUATE.
......*
```

【図１２】

図１２