

(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2005-122407

(P2005-122407A)

(43) 公開日 平成17年5月12日(2005.5.12)

(51) Int.Cl.⁷

G06F 9/44

F I

G06F 9/06 620 L

テーマコード (参考)

5B076

審査請求 未請求 請求項の数 10 O L (全 58 頁)

(21) 出願番号	特願2003-355695 (P2003-355695)	(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
(22) 出願日	平成15年10月15日(2003.10.15)	(74) 代理人	100089118 弁理士 酒井 宏明
		(72) 発明者	伊藤 栄信 神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内
		(72) 発明者	松尾 昭彦 神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内
		(72) 発明者	五味 俊明 神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内

最終頁に続く

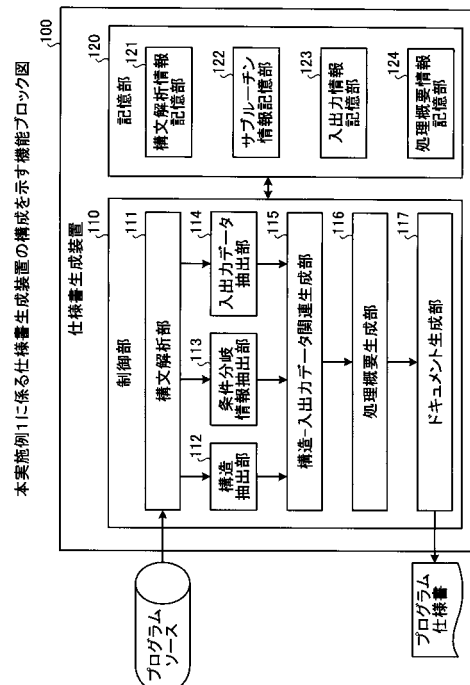
(54) 【発明の名称】 仕様書生成プログラムおよびその記録媒体、仕様書生成装置ならびに仕様書生成方法

(57) 【要約】

【課題】 ソースプログラムを解析して仕様書を生成する仕様書生成装置でソフトウェアの処理の全体像の把握を容易にする仕様書を生成すること。

【解決手段】 構文解析部111による構文解析結果から、条件分岐情報抽出部113が条件分岐の条件情報を抽出し、構造抽出部112がサブルーチンの呼出構造を条件情報とともに抽出し、入出力データ抽出部114が各サブルーチンの入出力情報を抽出する。そして、構文-入出力データ関連生成部116が条件分岐の条件情報、呼出構造情報および入出力情報に基づいてプログラムの呼出構造と入出力情報の関連づけを行い、処理概要生成部116が構文-入出力データ関連生成部116により関連づけられた情報から指定された範囲の情報を抽出して処理概要情報を作成し、ドキュメント生成部117が処理概要生成部116により生成された処理概要情報に基づいてプログラム仕様書を作成する。

【選択図】 図1



【特許請求の範囲】

【請求項 1】

ソースプログラムを解析して仕様書を生成する仕様書生成プログラムであって、
前記ソースプログラムを解析して該ソースプログラムのプログラム呼出構造とデータ入出力とを関連づけた入出力付構造情報を生成する入出力付構造情報生成手順と、
前記入出力付構造情報生成手順により生成された入出力付構造情報から一部の情報を抽出して前記ソースプログラムの処理概要情報を生成する処理概要情報生成手順と、
前記処理概要情報生成手順により生成された処理概要情報を用いて前記ソースプログラムのプログラム仕様書を生成する文書生成手順と
をコンピュータに実行させることを特徴とする仕様書生成プログラム。

10

【請求項 2】

前記処理概要情報生成手順は、特定の呼出階層のサブルーチンの情報と該サブルーチンに関連づけられた入出力情報とを入出力付構造情報から抽出して前記処理概要情報を生成し、

前記文書生成手順は、前記特定の呼出階層のサブルーチンの情報と該サブルーチンに関連づけられた入出力情報とを用いてプログラム仕様書を生成することを特徴とする請求項 1 に記載の仕様書生成プログラム。

【請求項 3】

前記文書生成手順は、サブルーチンの呼出階層を列とし、各サブルーチンの名前をサブルーチンの呼出階層値に対応する列に表示する表形式の呼出構造図をプログラム仕様書に含めることを特徴とする請求項 2 に記載の仕様書生成プログラム。

20

【請求項 4】

前記入出力付構造情報生成手順は、プログラム呼出構造にプログラム呼出条件を関連づけた入出力付構造情報を生成し、

前記処理概要情報生成手順は、プログラム呼出構造にプログラム呼出条件を関連づけた処理概要情報を生成し、

前記文書生成手順は、プログラム呼出構造にプログラム呼出条件を関連づけてプログラム仕様書を生成することを特徴とする請求項 1 に記載の仕様書生成プログラム。

【請求項 5】

前記文書生成手順は、生成したプログラム仕様書の所定の位置に利用者によって追加されたコメントを抽出するコメント抽出手順と、

30

新たに生成するプログラム仕様書の所定の位置に、前記コメント抽出手順により抽出されたコメントを継承するコメント継承手順とを

さらにコンピュータに実行させることを特徴とする請求項 1、2 または 4 に記載の仕様書生成プログラム。

【請求項 6】

バッチジョブ記述言語で記載されたバッチジョブ記述からバッチジョブを構成するジョブステップの入出力情報を抽出するステップ入出力情報抽出手順と、

前記ステップ入出力情報抽出手順により抽出されたジョブステップの入出力情報に基づいて前記バッチジョブ全体の入力情報および出力情報を特定するジョブ入出力情報特定手順と、

40

前記ジョブ入出力情報特定手順により特定された入力情報を入力するかまたは出力情報を出力するジョブステップを特定し、特定したジョブステップで呼び出されるプログラムの情報を抽出するプログラム情報抽出手順と、

前記ジョブ入出力情報特定手順により特定された入力情報および出力情報、ならびに前記プログラム情報抽出手順により抽出されたプログラムの情報を用いて前記バッチジョブのバッチジョブ処理概要書を生成するバッチジョブ処理概要書生成手順と

をさらにコンピュータに実行させることを特徴とする請求項 1、2 または 4 に記載の仕様書生成プログラム。

【請求項 7】

50

画面についての情報を定義した画面定義体を解析して画面遷移情報を作成する画面遷移情報作成手順と、

前記画面遷移情報作成手順により作成された画面遷移情報を用いて画面遷移図を生成する画面遷移図生成手順と

をさらにコンピュータに実行させることを特徴とする請求項 1、2 または 4 に記載の仕様書生成プログラム。

【請求項 8】

ソースプログラムを解析して仕様書を生成する仕様書生成プログラムを記録したコンピュータ読み取り可能な記録媒体であって、

前記ソースプログラムを解析して該ソースプログラムのプログラム呼出構造とデータ入出力とを関連づけた入出力付構造情報を生成する入出力付構造情報生成手順と、 10

前記入出力付構造情報生成手順により生成された入出力付構造情報から一部の情報を抽出して前記ソースプログラムの処理概要情報を生成する処理概要情報生成手順と、

前記処理概要情報生成手順により生成された処理概要情報を用いて前記ソースプログラムのプログラム仕様書を生成する文書生成手順と

をコンピュータに実行させる仕様書生成プログラムを記録したことを特徴とするコンピュータ読み取り可能な記録媒体。

【請求項 9】

ソースプログラムを解析して仕様書を生成する仕様書生成装置であって、

前記ソースプログラムを解析して該ソースプログラムのプログラム呼出構造とデータ入出力とを関連づけた入出力付構造情報を生成する入出力付構造情報生成手段と、 20

前記入出力付構造情報生成手段により生成された入出力付構造情報から一部の情報を抽出して前記ソースプログラムの処理概要情報を生成する処理概要情報生成手段と、

前記処理概要情報生成手段により生成された処理概要情報を用いて前記ソースプログラムのプログラム仕様書を生成する文書生成手段と

を備えたことを特徴とする仕様書生成装置。

【請求項 10】

ソースプログラムを解析して仕様書を生成する仕様書生成方法であって、

前記ソースプログラムを解析して該ソースプログラムのプログラム呼出構造とデータ入出力とを関連づけた入出力付構造情報を生成する入出力付構造情報生成工程と、 30

前記入出力付構造情報生成工程により生成された入出力付構造情報から一部の情報を抽出して前記ソースプログラムの処理概要情報を生成する処理概要情報生成工程と、

前記処理概要情報生成工程により生成された処理概要情報を用いて前記ソースプログラムのプログラム仕様書を生成する文書生成工程と

を含んだことを特徴とする仕様書生成方法。

【発明の詳細な説明】

【技術分野】

【0001】

この発明は、ソースプログラムを解析して仕様書を生成する仕様書生成プログラムおよびその記録媒体、仕様書生成装置ならびに仕様書生成方法に関し、特に、ソフトウェアの処理の全体像の把握を可能とし、もってソフトウェアの保守を容易にすることができる仕様書生成プログラムおよびその記録媒体、仕様書生成装置ならびに仕様書生成方法に関するものである。 40

【背景技術】

【0002】

ソフトウェアの保守や、そのソフトウェアを元にした新たなソフトウェアの開発を行う際には、プログラムの処理内容を理解する必要がある。しかし、計算機の処理に適する形に設計されたプログラム言語は人間が直接読んでも内容を理解するのが難しく、特に多数のプログラムから構成される大規模なソフトウェアにおいては、ソースプログラムだけから全体を把握するのはほとんど不可能である。このため、ソフトウェア保守や開発の際に 50

は、プログラムの処理内容を人間が理解しやすい形で記述した仕様書を用いるのが一般的となっている。

【0003】

特に、運用中のソフトウェアの保守を請け負うAPM(Application Portfolio Management)サービスでは、開発・運用経験の無いソフトウェアを対象として保守を行うため、仕様書の存在が不可欠となる。

【0004】

しかし、ソフトウェアの仕様書は、ほとんどの場合、プログラムの開発の際に作成されるものの、開発後に加えられた修正や仕様変更が反映されない場合も多く、内容が必ずしも信頼できるとは限らない。また、仕様書そのものが残っていない場合もあり、信頼できる仕様書を整備することが、保守作業を行う上で非常に重要となる。 10

【0005】

また、仕様情報をプログラムのコメント文として記入しておくことがよく行われるが、プログラムの修正にあわせてコメントの内容も正しく修正されているという保証がないため、なんら解決とはなっていない。

【0006】

そこで、プログラムの呼出構造などを機械的に解析して仕様書を自動生成することが従来から試みられている（例えば、特許文献1、非特許文献1参照。）。このような仕様書の自動生成では、生成される仕様書の内容がプログラムの内容と一致していることを保証することができ、信頼できる仕様書を得ることができる。 20

【0007】

【特許文献1】特開2002-215391号公報

【非特許文献1】「SIMPLIAシリーズ」、[平成15年9月25日検索]、インターネット<URL: <http://software.fujitsu.com/jp/simplia/>>

【発明の開示】

【発明が解決しようとする課題】

【0008】

しかしながら、従来のように、プログラムの呼出構造などを解析して表示するだけでは、プログラムの詳細な構造はわかっても、プログラムの理解に最も必要な処理概要を把握することができないという問題があった。特に、プログラムの概要理解に最も重要な、プログラムの呼出構造と入出力情報を関連づける情報がないという問題があった。 30

【0009】

また、一度生成した仕様書に対して利用者がプログラムの理解の助けとして書き込みを行った場合、プログラム修正に伴って仕様書を再度自動生成すると、新しく生成された仕様書には書き込んだ情報が欠落してしまうため、人間が内容を書き写す必要があるという問題があった。

【0010】

また、複数のプログラムの実行によって処理が行われるジョブ全体の処理を理解する場合には、個々のプログラムの仕様書ではなく、ジョブ全体の処理概要について記載した仕様書が必要となるが、このような仕様書は、個々のプログラムを解析しただけでは生成することができないという問題があった。 40

【0011】

この発明は、上述した従来技術による問題点を解消するためになされたものであり、ソフトウェアの処理の全体像の把握を可能とし、もってソフトウェアの保守を容易にすることができる仕様書生成プログラムおよびその記録媒体、仕様書生成装置ならびに仕様書生成方法を提供することを目的とする。

【課題を解決するための手段】

【0012】

上述した課題を解決し、目的を達成するため、本発明は、ソースプログラムを解析して仕様書を生成する仕様書生成プログラムであって、前記ソースプログラムを解析して該ソ 50

ースプログラムのプログラム呼出構造とデータ入出力とを関連づけた入出力付構造情報を生成する入出力付構造情報生成手順と、前記入出力付構造情報生成手順により生成された入出力付構造情報から一部の情報を抽出して前記ソースプログラムの処理概要情報を生成する処理概要情報生成手順と、前記処理概要情報生成手順により生成された処理概要情報を用いて前記ソースプログラムのプログラム仕様書を生成する文書生成手順とをコンピュータに実行させることを特徴とする。

【0013】

また、本発明は、上記の発明において、前記入出力付構造情報生成手順は、プログラム呼出構造にプログラム呼出条件を関連づけた入出力付構造情報を生成し、前記処理概要情報生成手順は、プログラム呼出構造にプログラム呼出条件を関連づけた処理概要情報を生成し、前記文書生成手順は、プログラム呼出構造にプログラム呼出条件を関連づけてプログラム仕様書を生成することを特徴とする。

10

【0014】

また、本発明は、上記発明において、前記文書生成手順は、生成したプログラム仕様書の所定の位置に利用者によって追加されたコメントを抽出するコメント抽出手順と、新たに生成するプログラム仕様書の所定の位置に、前記コメント抽出手順により抽出されたコメントを継承するコメント継承手順とをさらにコンピュータに実行させることを特徴とする。

【0015】

また、本発明は、上記発明において、前記ソースプログラムを構成する文をまとめることにより該ソースプログラムのプログラム概要情報を生成するプログラム概要情報生成手順と、前記プログラム概要情報生成手順により生成されたプログラム概要情報から自然言語によるプログラム概要文を生成するプログラム概要文生成手順と、前記プログラム概要文生成手順により生成されたプログラム概要文を用いて前記ソースプログラムのプログラム概要書を生成するプログラム概要書生成手順とをさらにコンピュータに実行させることを特徴とする。

20

【0016】

また、本発明は、上記発明において、バッチジョブ記述言語で記載されたバッチジョブ記述からバッチジョブを構成するジョブステップの入出力情報を抽出するステップ入出力情報抽出手順と、前記ステップ入出力情報抽出手順により抽出されたジョブステップの入出力情報に基づいて前記バッチジョブ全体の入力情報および出力情報を特定するジョブ入出力情報特定手順と、前記ジョブ入出力情報特定手順により特定された入力情報を入力するかまたは出力情報を出力するジョブステップを特定し、特定したジョブステップで呼び出されるプログラムの情報を抽出するプログラム情報抽出手順と、前記ジョブ入出力情報特定手順により特定された入力情報および出力情報、ならびに前記プログラム情報抽出手順により抽出されたプログラムの情報を用いて前記バッチジョブのバッチジョブ処理概要書を生成するバッチジョブ処理概要書生成手順とをさらにコンピュータに実行させることを特徴とする。

30

【0017】

また、本発明は、上記発明において、画面についての情報を定義した画面定義体を解析して画面遷移情報を作成する画面遷移情報作成手順と、前記画面遷移情報作成手順により作成された画面遷移情報を用いて画面遷移図を生成する画面遷移図生成手順とをさらにコンピュータに実行させることを特徴とする。

40

【0018】

また、本発明は、ソースプログラムを解析して仕様書を生成する仕様書生成プログラムを記録したコンピュータ読み取り可能な記録媒体であって、前記ソースプログラムを解析して該ソースプログラムのプログラム呼出構造とデータ入出力とを関連づけた入出力付構造情報を生成する入出力付構造情報生成手順と、前記入出力付構造情報生成手順により生成された入出力付構造情報から一部の情報を抽出して前記ソースプログラムの処理概要情報を生成する処理概要情報生成手順と、前記処理概要情報生成手順により生成された処理

50

概要情報を用いて前記ソースプログラムのプログラム仕様書を生成する文書生成手順とをコンピュータに実行させる仕様書生成プログラムを記録したことを特徴とする。

【0019】

また、本発明は、ソースプログラムを解析して仕様書を生成する仕様書生成装置であって、前記ソースプログラムを解析して該ソースプログラムのプログラム呼出構造とデータ入出力とを関連づけた入出力付構造情報を生成する入出力付構造情報生成手段と、前記入出力付構造情報生成手段により生成された入出力付構造情報から一部の情報を抽出して前記ソースプログラムの処理概要情報を生成する処理概要情報生成手段と、前記処理概要情報生成手段により生成された処理概要情報を用いて前記ソースプログラムのプログラム仕様書を生成する文書生成手段とを備えたことを特徴とする。

10

【0020】

また、本発明は、ソースプログラムを解析して仕様書を生成する仕様書生成方法であって、前記ソースプログラムを解析して該ソースプログラムのプログラム呼出構造とデータ入出力とを関連づけた入出力付構造情報を生成する入出力付構造情報生成工程と、前記入出力付構造情報生成工程により生成された入出力付構造情報から一部の情報を抽出して前記ソースプログラムの処理概要情報を生成する処理概要情報生成工程と、前記処理概要情報生成工程により生成された処理概要情報を用いて前記ソースプログラムのプログラム仕様書を生成する文書生成工程とを含んだことを特徴とする。

【発明の効果】

【0021】

20

本発明によれば、ソースプログラムを解析してソースプログラムのプログラム呼出構造とデータ入出力とを関連づけた入出力付構造情報を生成し、生成した入出力付構造情報から一部の情報を抽出してソースプログラムの処理概要情報を生成し、生成した処理概要情報を用いてソースプログラムのプログラム仕様書を生成するよう構成したので、ソフトウェアの処理の全体像の把握を可能とし、もってソフトウェアの保守を容易にすることができるという効果を奏する。

【0022】

また、本発明によれば、プログラム呼出構造にプログラム呼出条件を関連づけた入出力付構造情報を生成し、プログラム呼出構造にプログラム呼出条件を関連づけた処理概要情報を生成し、プログラム呼出構造にプログラム呼出条件を関連づけてプログラム仕様書を生成するよう構成したので、どのような条件の際に入出力が行われているかの把握を容易にすることができるという効果を奏する。

30

【0023】

また、本発明によれば、生成したプログラム仕様書の所定の位置に利用者によって追加されたコメントを抽出し、新たに生成するプログラム仕様書の所定の位置に、抽出したコメントを継承するよう構成したので、利用者によりプログラム仕様書に追記されたコメントを自動的に継承することができるという効果を奏する。

【0024】

また、本発明によれば、ソースプログラムを構成する文をまとめることによりソースプログラムのプログラム概要情報を生成し、生成したプログラム概要情報から自然言語によるプログラム概要文を生成し、生成したプログラム概要文を用いてソースプログラムのプログラム概要書を生成するよう構成したので、プログラムの処理内容の理解を容易し、理解に要する時間を短縮することができるという効果を奏する。

40

【0025】

また、本発明によれば、バッチジョブ記述言語で記載されたバッチジョブ記述からバッチジョブを構成するジョブステップの入出力情報を抽出し、抽出したジョブステップの入出力情報に基づいてバッチジョブ全体の入力情報および出力情報を特定し、特定した入力情報を入力するかまたは出力情報を出力するジョブステップを特定し、特定したジョブステップで呼び出されるプログラムの情報を抽出し、特定された入力情報および出力情報、ならびに抽出したプログラムの情報を用いてバッチジョブのバッチジョブ処理概要書を生

50

成するよう構成したので、バッチジョブ処理全体の概要の把握を容易にすることができるという効果を奏する。

【0026】

また、本発明によれば、画面についての情報を定義した画面定義体を解析して画面遷移情報を作成し、作成した画面遷移情報を用いて画面遷移図を生成するよう構成したので、オンラインシステムの処理概要の把握を容易にすることができるという効果を奏する。

【発明を実施するための最良の形態】

【0027】

以下に添付図面を参照して、この発明に係る仕様書生成プログラムおよびその記録媒体、仕様書生成装置ならびに仕様書生成方法の好適な実施例を詳細に説明する。なお、本実施例では、本発明をCOBOLで開発されたプログラムに適用した場合を中心に説明する。

【実施例1】

【0028】

まず、本実施例1に係る仕様書生成装置の構成について説明する。図1は、本実施例1に係る仕様書生成装置の構成を示す機能ブロック図である。同図に示すように、この仕様書生成装置100は、プログラムソースからプログラム仕様書を作成する処理を行う制御部110と、制御部110による処理の途中結果などを記憶する記憶部120とを有する。

【0029】

制御部110は、構文解析部111と、構造抽出部112と、条件分岐情報抽出部113と、入出力データ抽出部114と、構造－入出力データ関連生成部115と、処理概要生成部116と、ドキュメント生成部117とを有し、記憶部120は、構文解析情報記憶部121と、サブルーチン情報記憶部122と、入出力情報記憶部123と、処理概要情報記憶部124とを有する。

【0030】

構文解析部111は、プログラムソースを読み込み、読み込んだプログラムソース中の文や変数の情報を抽出して構文解析情報を生成する処理部であり、構文解析情報記憶部121は、構文解析部111が生成した構文解析情報を記憶する記憶部である。

【0031】

構造抽出部112は、構文解析情報記憶部121から構文解析情報を読み出し、プログラム内のサブルーチン呼出における呼出先の解決を行い、実行順序にしたがったプログラムの呼出構造をサブルーチン情報として抽出する処理部である。

【0032】

図2は、構造抽出部112が抽出するサブルーチン呼出の構造を説明するための説明図である。同図は、あるプログラムが「サブルーチン1」を呼び出し、「サブルーチン1」が「サブルーチン2」を呼び出し、「サブルーチン2」が「サブルーチン3」を呼び出す場合を示している。

【0033】

なお、ここでは、サブルーチンはプログラムが他のプログラムを呼び出す場合（COBOLではCALL）と、プログラムの内部に含まれる処理の塊を呼び出す場合（COBOLではPERFORM）の両方を示している。

【0034】

そして、「サブルーチン3」の実行が終了すると、「サブルーチン2」に制御が戻り、「サブルーチン2」の実行が終了すると「サブルーチン1」に制御が戻り、「サブルーチン1」の実行が終了すると、もとのプログラムに制御が戻る。構造抽出部112は、このようなプログラムの構造を抽出し、サブルーチン情報を生成する。

【0035】

サブルーチン情報記憶部122は、構造抽出部112が生成したサブルーチン情報を記憶する記憶部である。図3は、サブルーチン情報記憶部122が記憶するサブルーチン情

10

20

30

40

50

報のデータ構造の一例を示す図である。

【0036】

同図に示すように、このサブルーチン情報記憶部122が記憶するサブルーチン情報には、サブルーチンの名前を示すサブルーチン名と、呼出の深さを示す呼出階層値と、サブルーチンが呼び出されるときに条件のリストである呼出条件リストと、サブルーチンが行う入出力の情報のリストである入出力情報リストと、このサブルーチンが呼び出すサブルーチンのサブルーチン情報のリストである呼出リストが含まれる。ここで、サブルーチン名は、メインルーチンなどサブルーチン名がない場合には、” ”（空文字）がセットされる。

【0037】

また、呼出リストは、同様のデータ構造を有するサブルーチン情報のリストである。図4は、サブルーチン情報のリストのつながり（ツリー構造）を説明するための説明図である。同図は、呼出階層値が「1」であるサブルーチン情報の呼出リストは、呼出階層が「2」であるサブルーチンのサブルーチン情報のリストで構成され、呼出階層値が「2」であるサブルーチン情報の呼出リストは、呼出階層が「3」であるサブルーチンのサブルーチン情報のリストで構成され、全体としてサブルーチン情報がツリー構造を構成することを示している。

【0038】

図5は、図3に示した呼出条件リストのデータ構造の一例を示す図である。同図に示すように、呼出条件リストを構成する各呼出条件には、IF文やSWITCH文における分岐条件、ループ文におけるループ条件などの条件の種別を示す条件種別と、条件を記述した条件記述とが含まれる。例えば、条件記述としては、「 $i < 100$ 」などがある。

【0039】

図6は、図3に示した入出力情報リストのデータ構造の一例を示す図である。同図に示すように、入出力情報リストを構成する各入出力情報には、入力か出力かを示す入出力区別と、入出力の対象がデータベースかファイルかなどを示す対象種類と、入出力の対象の名前を示す対象名と、入出力に使用する入出力命令を示す使用命令文と、対象に固有なその他の情報を示すオプション情報とが含まれる。

【0040】

条件分岐情報抽出部113は、構文解析情報記憶部121に記憶された構文解析情報から図5に示した呼出条件リストを生成する処理部である。この条件分岐情報抽出部113によって生成された呼出条件リストは、サブルーチン情報記憶部122に格納される。

【0041】

入出力データ抽出部114は、構文解析情報記憶部121に記憶された構文解析情報を読み出し、各サブルーチンに対して図6に示した入出力情報リストを生成する処理部であり、入出力情報記憶部123は、入出力データ抽出部114により生成された入出力情報リストをサブルーチン名と対応させて記憶する記憶部である。

【0042】

構造－入出力データ関連生成部115は、入出力情報記憶部123に記憶された入出力情報リストとプログラムの呼出構造とを対応させる処理部である。具体的には、この構造－入出力データ関連生成部115は、入出力情報記憶部123に記憶された入出力情報リストをサブルーチン情報記憶部122の中の対応するサブルーチン情報の入出力情報リストとして格納する。

【0043】

この構造－入出力データ関連生成部115が、入出力情報リストとプログラムの呼出構造とを対応させることによって、仕様書生成装置100は、プログラムの呼出構造と入出力情報を関連付けたプログラム仕様書を生成することができる。

【0044】

処理概要生成部116は、構造－入出力データ関連生成部115により入出力情報リストが格納されたサブルーチン情報からプログラムの処理概要情報を生成する処理部である

10

20

30

40

50

。処理概要情報記憶部 1 2 4 は、処理概要生成部 1 1 6 により生成された処理概要情報を記憶する記憶部である。

【0045】

処理概要生成部 1 1 6 は、

(1) 呼出階層値が特定の範囲 (例えば、2 および 3) にあるサブルーチン情報だけを抽出する

(2) 特定の単語 (例えば、「エラー」、「障害」、「初期化」など) をサブルーチン名に含むサブルーチン情報以下のサブルーチン情報は除く

(3) 入出力を行うサブルーチンだけを残す

(4) ループ処理を含むサブルーチン情報を残す
などの処理によって処理概要情報を生成する。

10

【0046】

図 7 は、処理概要生成部 1 1 6 が生成する処理概要情報を説明するための説明図である。同図は、呼出階層値が「2 および 3」のサブルーチン情報だけを抽出して処理概要情報を作成する場合を示している。

【0047】

このように、この処理概要生成部 1 1 6 が、サブルーチン情報からプログラムの処理概要情報を生成することによって、仕様書生成装置 1 0 0 は、プログラムの呼出構造と入出力情報が関連づけられた処理概要を示すプログラム仕様書を生成することかできる。

【0048】

また、この処理概要生成部 1 1 6 は、プログラムの入出力ファイルを図示するために必要な情報、プログラムが使用する共通領域についての情報、プログラムが入出力するファイルについての情報を生成し、処理概要情報記憶部 1 2 4 に格納する。

20

【0049】

ドキュメント生成部 1 1 7 は、処理概要情報記憶部 1 2 4 に記憶された処理概要情報を用いてプログラム仕様書を生成する処理部である。具体的には、このドキュメント生成部 1 1 7 は、処理概要情報を所定の形式に整形し、プログラム仕様書として出力する。

【0050】

図 8 は、ドキュメント生成部 1 1 7 が生成するプログラム仕様書の一例を示す図である。同図に示すように、このプログラム仕様書は、プログラム名と、プログラムを格納するファイル名と、[コメント欄] と、[入出力関連図] と、[共通領域] と、[ファイル情報] と、[入出力とセクション名] と、[処理構造図] とを有する。

30

【0051】

[コメント欄] は、利用者が業務情報、運用ノウハウなどのコメントを追記する部分であり、[入出力関連図] は、プログラムの入出力ファイルを図示する部分である。[共通領域] は、プログラムが使用する共通領域のレコード名を示す部分であり、[ファイル情報] は、プログラムが入出力するファイルについての情報を示す部分である。

【0052】

[入出力とセクション名] は、処理概要情報として抽出されたサブルーチン名および抽出されたサブルーチン情報にふくまれる入出力情報を示す部分である。ここで、セクションとは、サブルーチンの一種であるが、ここでは、セクションでサブルーチンを代用している。この例では、呼出階層値が「2 および 3」のセクションについての情報が処理概要情報として示されている。

40

【0053】

すなわち、呼出階層値が「2」であるセクションとして「パラメタチェック処理」、「読込処理」および「詳細処理」が示されており、「詳細処理」が呼び出す呼出階層値が「3」であるセクションとして、カッコ内に「データチェック処理」、「調停額確定 F 読込処理」、「名義 F 読込処理」および「編集処理」が示されている。

【0054】

また、これらのセクションのサブルーチン情報に含まれる入出力情報には、入力として

50

「T J S C . 申込ファイル」、「T J S C . 入居ファイル」、「T J S C . 住戸マスタ」、「T J S C . 使用料区分マスタ」、「T J S C . 調停額確定ファイル」、「T J S C . 名義ファイル」、「T J S C . コードマスタ」、「S C _ B 1 0 1 0 0 1 . M B 1 0 1 0 1 2」、「S C _ B 1 0 1 0 0 1 . M B 1 0 1 0 1 4」および「I N 0 1」があり、出力として「T J C C F 0 5 1」がある。

【0055】

〔処理構造図〕は、処理概要情報として抽出されたサブルーチンの呼出構造を示す部分である。この例では、呼出階層値が「2」であるセクションとして「パラメータチェック処理」、「読込処理」および「詳細処理」が示されており、「詳細処理」が呼び出す呼出階層値が「3」の「データチェック処理」が示されている。

10

【0056】

次に、構造抽出部112によるサブルーチン情報生成処理および条件分岐情報抽出部113による条件分岐情報抽出処理の処理手順について説明する。図9は、構造抽出部112によるサブルーチン情報生成処理および条件分岐情報抽出部113による条件分岐情報抽出処理の処理手順を示すフローチャートである。なお、同図において、破線で囲まれた処理が条件分岐情報抽出部113による条件分岐情報抽出処理であり、他の部分は、構造抽出部112によるサブルーチン情報生成処理である。

【0057】

同図に示すように、まず、構造抽出部112が構文解析情報記憶部121に記憶された構文解析情報からサブルーチンごとのプログラム位置を取得し（ステップS111）、また、プログラム全体のスタート位置を検出する（ステップS112）。

20

【0058】

そして、最初のサブルーチン情報を作成し（ステップS113）、スタート位置の次のプログラム文の構造を調べ（ステップS114）、サブルーチン呼出であるか否かを判定する（ステップS115）。ここで、サブルーチン呼出を行うプログラム文としては、PERFORM文やCALL文がある。

【0059】

そして、サブルーチン呼出である場合には、今のプログラム位置を保持し（ステップS11c）、子サブルーチンに対するサブルーチン情報を作成して呼出リストに登録し（ステップS11d）、呼出条件作業リストがあれば子サブルーチン情報に追加する（ステップS11e）。そして、調べるプログラムの位置をサブルーチンのトップに移動し（ステップS11f）、ステップS114に戻って次のプログラム文の構造を調べる。

30

【0060】

一方、サブルーチン呼出でない場合には、条件分岐情報抽出部113が条件文であるか否かを判定し（ステップS116）、条件文である場合には、呼出条件作業リストに条件を追加し（ステップS117）、ステップS114に戻って次のプログラム文の構造を調べる。

【0061】

また、条件文でない場合には、条件文を抜けたか否かを判定し（ステップS118）、条件文を抜けた場合には、呼出条件作業リストから条件を一つ削除し（ステップS119）、ステップS114に戻って次のプログラム文の構造を調べる。

40

【0062】

また、条件文を抜けていない場合には、構造抽出部112が、サブルーチンの終わりか否かを判定し（ステップS11a）、サブルーチンの終わりでない場合には、プログラムの終わりか否かを判定し（ステップS11b）、プログラムの終わりでない場合には、ステップS114に戻って次のプログラム文の構造を調べ、プログラムの終わりの場合には、処理を終了する。

【0063】

一方、サブルーチンの終わりの場合には、操作するサブルーチン情報を親サブルーチンに戻し（ステップS11g）、保持していた親のプログラム位置に、調べるプログラムの

50

位置を戻し（ステップ S 1 1 h）、ステップ S 1 1 4 に戻って次のプログラム文の構造を調べる。

【0064】

このように、構造抽出部 1 1 2 が、プログラムの中のサブルーチン呼出およびサブルーチンの終わりを調べてサブルーチン情報を作成することによって、プログラムからプログラムの呼出構造を抽出することができる。

【0065】

また、条件分岐情報抽出部 1 1 3 が条件文を調べて呼出条件作業リストに保持し、サブルーチン呼出があった場合に、構造抽出部 1 1 2 が、呼出条件作業リストの条件をサブルーチン情報の呼出条件リストに登録することによって、サブルーチン呼出の呼出条件を抽出することができる。 10

【0066】

次に、入出力データ抽出部 1 1 4 による入出力情報抽出処理の処理手順について説明する。図 1 0 は、入出力データ抽出部 1 1 4 による入出力情報抽出処理の処理手順を示すフローチャートである。

【0067】

同図に示すように、この入出力データ抽出部 1 1 4 は、構文解析情報記憶部 1 2 1 に記憶された構文解析情報からプログラム全体のスタート位置を検出し（ステップ S 1 2 1）、また、現在のサブルーチン名を保持する（ステップ S 1 2 2）。 20

【0068】

そして、次のプログラム文を調べ（ステップ S 1 2 3）、入出力文か否かを判定する（ステップ S 1 2 4）。ここで、入出力文としては、R E A D 文、W R I T E 文などがある。

【0069】

そして、入出力文である場合には、今のサブルーチンで入出力情報記憶部 1 2 3 にサブルーチン情報を作成済みか否かを調べ、サブルーチン情報を作成していない場合には、サブルーチン情報を作成して入出力情報記憶部 1 2 3 に登録する（ステップ S 1 2 7）。 30

【0070】

そして、入出力文から入出力情報を作成し（ステップ S 1 2 8）、入出力情報記憶部 1 2 3 に記憶されたサブルーチン情報に入出力情報を追加し（ステップ S 1 2 9）、ステップ S 1 2 3 に戻って次のプログラム文を調べる。 30

【0071】

一方、プログラム文が入出力文でない場合には、サブルーチンが変わったか否かを調べ（ステップ S 1 2 5）、サブルーチンが変わった場合には、ステップ S 1 2 2 に戻って、変わったサブルーチンの名前を現在のサブルーチン名として保持する。

【0072】

また、サブルーチンが変わっていない場合には、プログラムの終わるか否かを調べ（ステップ S 1 2 6）、プログラムの終わりでなければステップ S 1 2 3 に戻って次のプログラム文を調べ、プログラムの終わりであれば処理を終了する。

【0073】

このように、この入出力データ抽出部 1 1 4 が、プログラムから入出力情報を抽出し、抽出した入出力情報を入出力を行うサブルーチンに対応させて入出力情報記憶部 1 2 3 に記憶することによって、サブルーチンと入出力情報を関連づけることができる。 40

【0074】

次に、構造－入出力データ関連生成部 1 1 5 による入出力情報のサブルーチン情報記憶部 1 2 2 への追加処理の処理手順について説明する。図 1 1 は、構造－入出力データ関連生成部 1 1 5 による入出力情報のサブルーチン情報記憶部 1 2 2 への追加処理の処理手順を示すフローチャートである。

【0075】

同図に示すように、この構造－入出力データ関連生成部 1 1 5 は、サブルーチン情報記 50

憶部 1 2 2 に記憶されたサブルーチン情報のツリー構造から一つのサブルーチン情報を取り出す（ステップ S 1 3 1）。そして、サブルーチン名を取得し（ステップ S 1 3 2）、入出力情報記憶部 1 2 3 に記憶されたサブルーチン情報から同じ名前のサブルーチンを探す（ステップ S 1 3 3）。

【0076】

そして、同じ名前のサブルーチンが見つかったか否かを判定し（ステップ S 1 3 4）、同じ名前のサブルーチンが見つかった場合には、入出力情報記憶部 1 2 3 に記憶された入出力情報をサブルーチン情報記憶部 1 2 2 の対応するサブルーチン情報に追加する（ステップ S 1 3 5）。

【0077】

そして、サブルーチン情報記憶部 1 2 2 のサブルーチン情報をすべて取り出したか否かを判定し（ステップ S 1 3 6）、サブルーチン情報をすべて取り出していない場合には、ステップ S 1 3 1 に戻って次のサブルーチン情報を取り出し、サブルーチン情報をすべて取り出した場合には、処理を終了する。

【0078】

このように、この構造－入出力データ関連生成部 1 1 5 が入出力情報記憶部 1 2 3 に記憶された入出力情報をサブルーチン情報記憶部 1 2 2 の対応するサブルーチン情報に追加することによって、プログラム呼出構造と入出力情報を関連づけることができる。

【0079】

なお、サブルーチンとして、プログラムの内部に含まれる処理の塊を呼び出す場合と、別のプログラムを呼び出す場合の両方が混在するとき、階層が非常に深くなる場合がある。このようなとき、別のプログラムを呼び出したその次以降の階層は省略し、入出力情報だけ、その呼び出されるプログラムのサブルーチン情報の入出力情報リストに加えるというような省略方法もある。

【0080】

次に、処理概要生成部 1 1 6 による処理概要生成処理の処理手順について説明する。図 1 2 は、処理概要生成部 1 1 6 による処理概要生成処理の処理手順を示すフローチャートである。なお、ここでは、処理概要生成部 1 1 6 による処理概要生成処理のうち、呼出階層の値について特定の値の範囲のサブルーチン情報を処理概要情報として生成する処理について説明する。

【0081】

同図に示すように、この処理概要生成部 1 1 6 は、サブルーチン情報記憶部 1 2 2 に記憶されたサブルーチン情報のツリー構造を読み出し、読み出したツリー構造から一つのサブルーチン情報を取り出す（ステップ S 1 4 1）。そして、呼出階層値を取得し（ステップ S 1 4 2）、取得した呼出階層値が指定の範囲内か否かを判定する（ステップ S 1 4 3）。

【0082】

その結果、呼出階層値が指定の範囲にない場合には、そのサブルーチン情報をツリー構造から削除する（ステップ S 1 4 4）。そして、全てのサブルーチン情報をツリー構造から取り出して呼出階層値を調べたか否かを判定し（ステップ S 1 4 5）、全てのサブルーチン情報を調べていない場合には、ステップ S 1 4 1 に戻って次のサブルーチン情報を調べ、全てのサブルーチン情報を調べた場合には、残ったツリー構造のサブルーチン情報を処理概要情報記憶部 1 2 4 に格納する（ステップ S 1 4 6）。

【0083】

このように、この処理概要生成部 1 1 6 がサブルーチン情報記憶部 1 2 2 に記憶されたツリー構造のサブルーチン情報から、指定された範囲のサブルーチン情報だけを取り出すことによって、プログラムの呼出構造と入出力情報とが関連づけられた処理概要を生成することができる。

【0084】

上述してきたように、本実施例 1 では、構文解析部 1 1 1 による構文解析結果から構造

10

20

30

40

50

抽出部 1 1 2 がサブルーチンの呼出構造を抽出し、入出力データ抽出部 1 1 4 が各サブルーチンの入出力情報を抽出する。そして、構文－入出力データ関連生成部 1 1 6 が条件分岐情報抽出部 1 1 3、構造抽出部 1 1 2 および入出力データ抽出部 1 1 4 により抽出された情報に基づいてプログラムの呼出構造と入出力情報の関連づけを行い、処理概要生成部 1 1 6 が構文－入出力データ関連生成部 1 1 6 により関連づけられた情報から指定された範囲の情報を抽出して処理概要情報を作成し、ドキュメント生成部 1 1 7 が処理概要生成部 1 1 6 により生成された処理概要情報に基づいてプログラム仕様書を作成することとしたので、プログラムの処理概要の把握を容易にすることができる。

【0085】

また、本実施例 1 では、構文解析部 1 1 1 による構文解析結果から条件分岐情報抽出部 1 1 3 が条件分岐の条件情報を抽出し、構造抽出部 1 1 2 がサブルーチンの条件分岐の条件情報をサブルーチン情報に加えることとしたので、どのような条件の際に入出力が行われているかの把握を容易にすることができる。

【実施例 2】

【0086】

ところで、図 8 に示したプログラム仕様書の「コメント欄」には、利用者は自由にコメントを追記することができる。しかしながら、利用者が追記したコメントは、新たなプログラム仕様書を自動生成した場合に、新たに生成されたプログラム仕様書には反映されないため、利用者はコメントを再度追記する必要がある。そこで、本実施例 2 では、新たにプログラム仕様書を自動生成した場合に、利用者によるコメントの再追記を不要とする仕様書生成装置について説明する。

【0087】

まず、本実施例 2 に係るコメント継承について説明する。図 1 3 は、本実施例 2 に係るコメント継承を説明するための説明図である。同図は、文書作成アプリケーションで作成されたプログラム仕様書に設けられるコメント記述領域を示す。

【0088】

コメント記述領域は、文書作成アプリケーションによって表形式で作成されており、各コメント記述領域には、「コメント（ユーザ記述領域）」や「[変数一覧]」などのキーワードがつけられている。また、コメント記述領域には、一つの記述領域しかない場合（タイプ 1）と、複数の記述領域がある場合（タイプ 2）とがある。

【0089】

本実施例 2 に係る仕様書生成装置は、これらのコメント記述領域に追記された利用者のコメントをプログラム仕様書から読み取り、生成するプログラム仕様書の同じ位置に読み取ったコメントを自動的に継承する。

【0090】

また、プログラム仕様書には補足領域を設け、コメントが追記されたコメント記述領域が新たに生成するプログラム仕様書にない場合には、補足領域にそのコメントを記入する。

【0091】

このように、本実施例 2 に係る仕様書生成装置は、コメントの追記されたプログラム仕様書を読み込み、新たに生成するプログラム仕様書の同じ位置に読み取ったコメントを自動的に継承することによって、利用者がコメントとして追記した業務情報や運用ノウハウなどの重要な情報を新旧のプログラム仕様書間で継承することができる。

【0092】

次に、本実施例 2 に係る仕様書生成装置の構成について説明する。図 1 4 は、本実施例 2 に係る仕様書生成装置の構成を示す機能ブロック図である。なお、ここでは説明の便宜上、図 1 に示した各部と同様の役割を果たす機能部については同一符号を付すこととしてその詳細な説明を省略する。

【0093】

同図に示すように、この仕様書生成装置 2 0 0 は、制御部 2 1 0 と、制御部 2 1 0 によ

10

20

30

40

50

る処理の途中結果などを記憶する記憶部 120 とを有する。制御部 210 は、図 1 に示したドキュメント生成部 117 の代わりにドキュメント生成部 211 を有する。

【0094】

ドキュメント生成部 211 は、ドキュメント生成部 117 の有する機能に加えて、コメント記入内容継承部 211a を有する。このコメント記入内容継承部 211a は、コメント記入済プログラム仕様書を読み込んでコメントを抽出し、新たに生成するプログラム仕様書に、抽出したコメントを反映する。

【0095】

このコメント記入内容継承部 211a がコメント記入済プログラム仕様書を読み込んでコメントを抽出し、新たに生成するプログラム仕様書に、抽出したコメントを反映することによって、仕様書生成装置 200 は、古いプログラム仕様書に記入されたコメントを新たに生成するプログラム仕様書に継承することができる。

【0096】

次に、コメント記入内容継承部 211a がコメント継承処理で使用するデータ構造について説明する。仕様書生成装置 200 は、コメント記述領域の位置情報をプログラム仕様書に付加するプロパティに対応させ、これらのプロパティのリストを値とするプロパティをコメント記述領域リストとして管理する。

【0097】

図 15 は、コメント記述領域リストに対応するプロパティのデータ構造の一例を示す図である。同図に示すように、このコメント記述領域リストは、ReflectionList というプロパティであり、プロパティの値は、各コメント記述領域に対応するプロパティ「Reflection1」～「ReflectionN」から構成されるリストである。

【0098】

図 16-1 は、コメント記述領域に対応するプロパティのデータ構造の一例を示す図である。同図は、コメント記述領域に対応するプロパティの一例である「Reflection1」のデータ構造を示しており、「Reflection1」はコメント記述領域を識別するキーワードをプロパティの値としている。

【0099】

また、図 16-2 は、複数の記述領域がある場合のコメント記述領域に対応するプロパティのデータ構造の一例を示す図である。同図は、複数の記述領域がある場合のコメント記述領域に対応するプロパティの一例である「MultiReflection1」のデータ構造を示しており、「MultiReflection1」は、コメント記述領域を識別するキーワードと、それぞれの記述領域を識別する複数コメントキーワードの列の位置と、コメント記述領域の列の位置をプロパティの値としている。

【0100】

例えば、図 13 に示したコメント記述領域「変数一覧」に対応するプロパティの値は、「[変数一覧]，1，2」となる。すなわち、コメント記述領域「変数一覧」の 1 列目が各記述領域を識別する複数コメント識別キーワードであり、2 列目が記入されたコメントである。

【0101】

次に、図 14 に示したコメント記入内容継承部 211a の処理手順について説明する。図 17 は、図 14 に示したコメント記入内容継承部 211a の処理手順を示すフローチャートである。

【0102】

同図に示すように、このコメント記入内容継承部 211a は、コメントの継承元のプログラム仕様書の ReflectionList を探索し（ステップ S211）、ReflectionList から Reflection を一つ取得する（ステップ S212）。

【0103】

そして、取得した Reflection の値であるキーワードを用いて継承元のコメントと継承先のコメント継承位置とを取得し（ステップ S213 およびステップ S214）、継承先の

プログラム仕様書にコメント継承位置が存在するか否かを判定する（ステップ S 2 1 5）。その結果、コメント継承位置が存在する場合には、継承先のプログラム仕様書に継承元のプログラム仕様書のコメントを反映する（ステップ S 2 1 6）。

【0104】

そして、コメントの継承が成功したか否かを判定し（ステップ S 2 1 7）、継承が成功しなかった場合には、補足領域にコメントを反映する（ステップ S 2 1 8）。そして、ReflectionListの全てのReflectionを処理したか否かを判定し（ステップ S 2 1 9）、全てのReflectionを処理していない場合には、ステップ S 2 1 2 に戻って次のReflectionの処理を行い、全てのReflectionを処理した場合には、処理を終了する。

【0105】

このように、このコメント記入内容継承部 2 1 1 a が ReflectionList を用いて継承元のプログラム仕様書からコメントを読み出して継承先のプログラム仕様書に反映させることによって、利用者がプログラム仕様書に追記したコメントを自動的に継承することができる。

【0106】

次に、他の形式のプログラム仕様書のコメント記述領域について説明する。まず、スプレッドシートを用いたプログラム仕様書のコメント記述領域について説明する。図 1 8 は、スプレッドシートを用いたプログラム仕様書のコメント記述領域の一例を示す図である。

【0107】

同図に示すように、スプレッドシートの場合には、全ての領域が行と列で位置を特定できるため、コメント記述領域に対応するプロパティの値はキーワード、行位置および列位置となる。

【0108】

図 1 9 は、スプレッドシートを用いたプログラム仕様書のコメント記述領域に対応するプロパティのデータ構造の一例を示す図である。同図は、コメント記述領域に対応するプロパティの一例である「Reflection1」のデータ構造を示しており、「Reflection1」はキーワードとコメント記述領域行位置とコメント記述領域列位置とをプロパティの値としている。

【0109】

ここで、コメント記述領域行位置は、キーワードからの相対行位置である。例えば、図 1 8 に示したコメント記述領域「コメント（ユーザ記述領域）」に対応するプロパティの値は、「コメント（ユーザ記述領域）， 1， A）」となる。

【0110】

また、図 2 0 は、スプレッドシートを用いたプログラム仕様書のコメント記述領域（複数の記述領域がある場合）に対応するプロパティのデータ構造の一例を示す図である。同図は、複数の記述領域がある場合のコメント記述領域に対応するプロパティの一例である「MultiReflection1」のデータ構造を示しており、「MultiReflection1」はキーワードとコメント記述領域行位置と一コメントの行数と複数コメント識別キーワード列位置とコメント記述領域列位置とを値とをプロパティの値としている。

【0111】

ここで、コメント記述領域行位置は、キーワードからの相対行位置である。例えば、図 1 8 に示したコメント記述領域「[変数一覧]」に対応するプロパティの値は、「[変数一覧]， 2， 1， A， B）」となる。

【0112】

図 2 1 は、処理構造図におけるコメント記述領域を示す図である。なお、処理構造図の詳細については後述する。同図において、「コメント（n）」の箇所がコメント記述領域である。

【0113】

この処理構造図では、自動生成した内容が入ったセルと利用者がコメントを追記可能な

10

20

30

40

50

セルとを区別するために、前者のセルは書き込み禁止とし、両者のセルの色を変えておく。また、各セクションにコメント記入可能なセルが必ず存在するように、処理構造図生成時に各セクションの有効掛線範囲には、必ず空セルを入れるようにしておく。

【0114】

図22は、図21に示した処理構造図における各コメント記述領域の位置情報データの一例を示す図である。同図に示すように、この例では、コメント記述領域が対応するセクション構造と、対応セクションからの相対的変位と、付加列名とをコメント記述領域の位置情報としている。

【0115】

例えば、「コメント(1)」は「プログラム名」のセクションに対応し、位置は「プログラム名」のセルから2行0列目であり、「コメント(4)」は「プログラム名ー繰り返し処理[1]ーCALL"COM0002"[1]」のセクションに対応し、位置は「CALL"COM0002"」のセルから1行0列目である。

【0116】

仕様書生成装置200は、このような位置情報データを用いて処理構造図のコメント記述領域を管理することによって、処理構造図に追記された利用者のコメントを自動的に継承することができる。

【0117】

また、仕様書生成装置200は、継承先の処理構造図に対応するセクション構造は存在するが対応するセルや列がない場合には、備考欄にコメントを記入する。一方、継承先の処理構造図に対応するセクション構造が存在しない場合には、プログラム仕様書の末端の補足欄にコメント記述領域の位置情報とコメント内容とを記入する。

【0118】

また、後述する画面遷移図のようなページ制御があるようなワークシートの場合は、出力の度に位置情報が変わるため、上記の方法が使えない。そのため、1枚のシートを丸ごと利用者記入欄として設定し、利用者記入用としたシートに書いたコメントを丸ごと継承するようにする。

【0119】

図65は、利用者記入欄を設けないコメント継承の一例を示す図である。同図は、利用者記入欄を設けずに、コメント記入用のインタフェースを用意し、利用者がコメントを自由に記入する場合を示している。

【0120】

利用者はあらかじめ用意されたインタフェース(例えばコメント記入ボタンなど)を用いてコメントを記入する。そして、記入されたコメントは別文書として管理し、仕様書本体と関連づけておく。関連づけの方法には例えば文書の命名規則を決めておき、「XXX」という名前の仕様書に追加されたコメント文書は「XXX-コメント1」となるようにしておき、仕様書全体としてユニークになるようにしておけばよい。

【0121】

こうしておく、仕様書を再生成しても「XXX」は上書きされるが、「XXX-コメント1」は上書きされない、利用者が記入したコメントは保持されたままになる。また、再生成した仕様書から、前回のコメントを参照する場合は、コメントを表示するインタフェースを使って関連文書からコメント内容を抽出して表示する。

【0122】

例えば、図65は、仕様書「F10802N」へのコメントとして「修正記録030217」および「障害対応メモ」があり、「修正記録030217」の内容が「ジョブステップ3での在庫マスタの更新ロジックは、'03/2/17修正済」であることを示している。

【0123】

上述してきたように、本実施例2では、ドキュメント生成部211のコメント記入内容継承部211aが、コメント記入済のプログラム仕様書からコメントを取り出し、新たに

10

20

30

40

50

生成するプログラム仕様書の対応する場所に反映させることとしたので、利用者によりプログラム仕様書に追記されたコメントを自動的に継承することができる。これによって、保守担当者の交替時などに、前任者の知識をコメントとしてプログラム仕様書に書いておくことで引き継ぎ作業に伴う工数を大幅に減らすことができる。

【実施例 3】

【0124】

実施例 1 で説明したように、サブルーチンの呼出構造と入出力情報とを関連づけた処理概要によってプログラムの基本的な処理構造は理解することができる。しかしながら、プログラムを理解するためには、基本的な処理構造に加えてプログラムの処理内容も理解する必要があることも多い。そこで、本実施例 3 では、プログラムの処理内容の概要をプログラムから抽出してプログラム概要書として生成する仕様書生成装置について説明する。

10

【0125】

まず、本実施例 3 に係る仕様書生成装置の構成について説明する。図 23 は、本実施例 3 に係る仕様書生成装置の構成を示す機能ブロック図である。なお、ここでは説明の便宜上、図 1 に示した各部と同様の役割を果たす機能部については同一符号を付すこととしてその詳細な説明を省略する。

【0126】

同図に示すように、この仕様書生成装置 300 は、制御部 310 と、制御部 310 が使用する情報や制御部 310 による処理の途中結果などを記憶する記憶部 320 とを有する。制御部 310 は、図 1 に示した制御部 110 が有する機能部に加えてプログラム概要生成部 311 を有し、ドキュメント生成部 117 の代わりにドキュメント生成部 312 を有する。

20

【0127】

また、記憶部 320 は、図 1 に示した記憶部 120 が有する記憶部に加えて、まとめ処理記憶部 321 と、日本語テンプレート記憶部 322 と、プログラム概要情報記憶部 323 とを有する。

【0128】

プログラム概要生成部 311 は、プログラムから処理内容の概要をプログラム概要情報として生成する処理部であり、プログラム概要情報記憶部 323 は、プログラム概要生成部 311 が生成したプログラム概要情報を記憶する記憶部である。

30

【0129】

ドキュメント生成部 312 は、処理概要情報を用いてプログラム仕様書を生成するとともに、プログラム概要情報記憶部 323 に記憶されたプログラム概要情報を用いてプログラム概要書を生成する。

【0130】

まとめ処理記憶部 321 は、プログラム概要生成部 311 がプログラム概要情報を生成するために行うまとめ処理の内容を記憶した記憶部であり、日本語テンプレート記憶部 322 は、プログラム概要生成部 311 が日本語によるプログラム概要を生成するための日本語テンプレートを記憶した記憶部である。

【0131】

40

図 24 は、プログラム概要生成部 311 の構成を示す機能ブロック図である。同図に示すように、このプログラム概要生成部 311 は、データ重要度設定部 311a と、ステートメント重要度設定部 311b と、まとめ管理部 311c と、日本語変換部 311d とを有する。

【0132】

データ重要度設定部 311a は、構文解析情報記憶部 121 に記憶された構文解析情報を XML 形式のプログラム中間情報に変換し、プログラム中間情報に含まれるデータに対して重要度を割り当てる処理部である。

【0133】

図 25-1 および図 25-2 は、プログラム中間情報を説明するための図であり、図 2

50

5-1は、プログラムソース例を示す図であり、図25-2は、図25-1に示したプログラムソース例から生成されるプログラム中間情報を示す図である。

【0134】

図25-2に示すように、プログラム中間情報では、基本的にプログラムのステートメント（文）一つにつき一つのタグが対応している。また、パラメータ等はステートメントの内部に入れ子のタグを用いて記述する。

【0135】

例えば、「MOVE 0 TO W-ERR-FLAG」というステートメントは、
<move>

<ref><constant value="0" type="int"/> 10
</ref>

<def><var name="W-ERR-FLAG"/></def>
</move>

というプログラム中間情報に変換される。

【0136】

図26は、データの重要度の分類の一例を示す図である。同図に示すように、データ重要度設定部311aは、「処理パスの分岐条件に関わるデータ」を重要度が最も高い重要度「1」に分類して分類IDを「D-1」とし、「ファイル／データベースなどの出力に使われるデータ」を重要度「2」に分類して分類IDを「D-2」とする。

【0137】

20

また、データ重要度設定部311aは、「ファイル／データベースなどからの入力に使われるデータ」を重要度「3」に分類して分類IDを「D-3」とし、「その他のデータ」を重要度が最も低い重要度「4」に分類して分類IDを「D-4」とする。

【0138】

そして、データ重要度設定部311aは、プログラム中の全ての変数に対して、プログラム中間情報の変数用タグに属性の形でデータの重要度を付加する。図27は、プログラム中間情報の変数用タグに付加される重要度の一例を示す図である。

【0139】

同図に示すように、変数「DATA1」は、プログラム中間情報では<var name="DATA1"/>と表わされ、重要度「1」が付加されると、<var name="DATA1" data_priority="1"/>と表わされる。 30

【0140】

ステートメント重要度設定部311bは、データの重要度が付加されたプログラム中間情報にステートメントの重要度を付加する処理部である。図28は、ステートメントの重要度の分類の一例を示す図である。

【0141】

同図に示すように、ステートメント重要度設定部311bは、「重要度がD-1およびD-2に該当するデータに対し代入等の書き換えを行う文、サブルーチン呼出を行う文またはファイル、データベースに入出力を行う文」を重要度が最も高い重要度「1」に分類して分類IDを「S-1」とし、「条件判定文のうち判定直後に実行される文としてS-1の分類の文を含むもの」を重要度「2」に分類して分類IDを「S-2」とし、「その他の文」を重要度「3」に分類して分類IDを「S-3」とする。 40

【0142】

そして、ステートメント重要度設定部311bは、プログラム中間情報に記述されているステートメントのタグとデータの重要度を分析し、各ステートメントの重要度を決定する。そして、決定したステートメントの重要度をタグの属性として付加する。

【0143】

図29は、プログラム中間情報のステートメントのタグに付加される重要度の一例を示す図である。同図に示すように、「MOVE 0 TO DATA1」というステートメントについては、変数「DATA1」に重要度「1」が付加されているので、<move 50

statement__priority="1">のようにステートメントの重要度「1」が付加される。

【0144】

このようにして、ステートメント重要度設定部311bは、ステートメントの重要度を付加したプログラム中間情報を生成する。なお、データの重要度およびステートメントの重要度の分類については、まとめ管理部311cの処理で必要に応じて変更されたり細分化される場合もある。

【0145】

まとめ管理部311cは、データとステートメントの重要度が付加されたプログラム中間情報について、様々なまとめ処理を実行する処理部である。このまとめ管理部311cが行うまとめ処理には、プログラムの局所部分についてまとめを行うものと、プログラムの全体的な処理の流れについてまとめを行うものがある。

【0146】

局所的なまとめ処理では、プログラムの一部分についてのパターンを抽出し、抽出したパターンに対して所定の処理を行う。例えば、データ項目の代入処理が連続してプログラムに記述されており、そのデータ項目はある同じ集団項目に属している場合、その集団項目を用いた代入という形で代入処理を一つにまとめてしまう。

【0147】

また、局所的なまとめ処理では、重要でない文の削除も行う。あるいは、構造抽出部112によって抽出されたサブルーチン呼出階層値を用いて、呼び出し構造的に上位のサブルーチンを残すといった処理も行う。

【0148】

全体的なまとめ処理では、例えば、ジャンプ文によりプログラム全体が一つの大きなループになっていることを検出し、それをループしているという表現で置き換えるといった処理を行う。

【0149】

具体的には、このまとめ管理部311cは、まとめ処理を実行するまとめ処理プログラムを最初に読み込み、読み込んだまとめ処理プログラムを実行することによって、まとめ処理を実行する。

【0150】

図30は、まとめ管理部311cの初期動作を説明するための説明図である。同図に示すように、まとめ管理部311cは、まとめ処理プログラムのプログラム名を、まとめ登録情報として用意し、このまとめ登録情報に登録されたまとめ処理プログラムを最初に読み込んでおく。

【0151】

同図では、読み込むプログラム名として「ARRANGE__P1」、「ARRANGE__P2」、「ARRANGE__A1」および「ARRANGE__A2」が登録されており、まとめ管理部311cは、これらのまとめ処理プログラムを読み込んで実行する。

【0152】

なお、「ARRANGE__P1」および「ARRANGE__P2」は局所的なまとめ処理を行うプログラムであり、「ARRANGE__A1」および「ARRANGE__A2」は局所的なまとめ処理を行うプログラムである。また、まとめ登録情報に登録するプログラム名を入れ替えることによって、行いたいまとめ処理を容易に変更することができる。

【0153】

図31は、まとめ管理部311cとまとめ処理プログラムとのインタフェース（I/F）を示す図である。同図に示すように、インタフェースとしては、処理名取得I/Fと、種類取得I/Fと、優先度番号取得I/Fと、処理実行I/Fとがある。

【0154】

処理名取得I/Fは、まとめ処理プログラムが行うまとめ処理の処理名（例えば「集団項目のまとめ」）をまとめ管理部311cに通知するためのインタフェースであり、種類

10

20

30

40

50

取得 I / F は、まとめ処理の種類（例えば ” 局所 ”、” 全体 ”）をまとめ管理部 3 1 1 c に通知するためのインタフェースである。

【0 1 5 5】

また、優先度番号取得 I / F は、まとめ処理プログラムの優先度番号（例えば ” 1 0 ”、” 1 0 4 ”）をまとめ管理部 3 1 1 c に通知するためのインタフェースであり、処理実行 I / F は、まとめ管理部 3 1 1 c がまとめ処理プログラムを実行するときに使用するインタフェースである。

【0 1 5 6】

なお、まとめ管理部 3 1 1 c がまとめ処理プログラムを実行するときに渡すパラメータは、主にプログラム中間情報である。また、まとめ処理の中に構造抽出部 1 1 2 によって抽出された構造情報などを必要とするものがあれば、それらの情報もパラメータとして渡す。

【0 1 5 7】

また、優先度番号は、まとめ処理プログラムの処理順を決めるための相対的な番号であり、ここでは、この優先度番号の若い順にまとめ処理プログラムを実行する。まとめ管理部 3 1 1 c は、インタフェースを介して取得した情報を用いて、まとめ処理プログラムの実行順序を決定して保持する。

【0 1 5 8】

図 3 2 は、まとめ処理プログラムの実行順序保持の一例を示す図である。同図は、局所的なまとめ処理を行うまとめ処理プログラムを先に実行し、全体的なまとめ処理を行うまとめ処理プログラムを後に実行し、それぞれの処理では、優先番号の小さい順に実行する順序保持を示している。

【0 1 5 9】

なお、まとめ処理の処理名、種類、優先度番号は、まとめ処理プログラム開発時に開発者によって定義される。また、優先度番号が同じ場合には、まとめ登録情報に登録された順番に実行順序の保持がなされる。

【0 1 6 0】

まとめ管理部 3 1 1 c は、データとステートメントの重要度が付加されたプログラム中間情報を入力し、保持するまとめ処理プログラムの実行順序に従って、まとめ処理プログラムを実行していく。

【0 1 6 1】

また、まとめ管理部 3 1 1 c は、まとめ処理プログラムにプログラム中間情報を入力として与え、各まとめ処理プログラムは、プログラム中間情報を編集していく。そして、各まとめ処理プログラムで編集されたプログラム中間情報はその次に実行されるまとめ処理プログラムの入力となる。

【0 1 6 2】

このようにして、登録されているすべてのまとめ処理プログラムによって、プログラム中間情報の編集が行われていく。なお、まとめ処理プログラムの中には、プログラム中間情報から自分が処理できるプログラム構造のパターンを抽出するものがあるが、パターンが抽出できなかった場合は、そのまとめ処理を行わず、次のまとめ処理プログラムへ実行が引き渡される。そして、まとめ管理部 3 1 1 c は、すべてのまとめ処理プログラムの実行を終了すると、編集したプログラム中間情報を日本語変換部 3 1 1 d に引き渡す。

【0 1 6 3】

図 3 3 - 1 ~ 図 3 3 - 6 は、まとめ処理プログラムが行うまとめの規則の例を示す図である。図 3 3 - 1 ~ 図 3 3 - 4 は、局所的なまとめの規則を示し、図 3 3 - 5 および図 3 3 - 6 は、全体的なまとめの規則を示す。

【0 1 6 4】

図 3 3 - 1 は、「READ 文のファイル終了時の処理が、UNTIL 条件の成立と一致している場合、UNTIL の条件を「ファイル～の終わりまで」に置き換える。また、READ 文のファイル終了時の処理を削除する。」というまとめの規則を示す。

10

20

30

40

50

【0165】

図33-2は、「セクションの内容が短く、3行までの場合でかつ、呼出元（上位セクション）にて無条件で呼ばれている場合、セクションの内容を上位のセクションに含む形に展開する。」というまとめのルールを示す。

【0166】

図33-3は、「MOVE文が連続し、代入元の項目がすべて同じ集団項目に所属し、また、代入先の項目がすべて同じ集団項目に所属するとき、それら集団項目による代入文に置き換える。なお、集団項目内のすべての項目が列挙されていない場合は、その集団項目の部分であることを示す属性を加える。」というまとめのルールを示す。

【0167】

図33-4は、「(1) statement_priorityが"3"になっている文を削除する。また、このとき、削除される文のタグの親タグ<sequence>の属性qtの値を削除した文の数だけ減らす(qtはその内部に含む文の数を示す)。(2)また、このとき、<sequence>の属性qtの値が0の場合、それを含むセクション、パラグラフまたは、条件文(IF、EVALUATE、READ後条件)を削除する。セクションの場合は、その呼出元の呼出文(EVALUATE)も削除する。なお、(1)、(2)は、文の数が変わらなくなるまで対象となるプログラムについて繰り返し行う。」というまとめのルールを示す。

【0168】

図33-5は、「ファイルを読む処理があり、その後にループがあり、ループの内部の処理の最後で、ファイルを読んでいる場合、「ループの条件成立まで、繰り返しファイルを読む」という形に置き換える。ループがセクションの場合は、それを呼び出している部分に展開する。」というまとめのルールを示す。

【0169】

図33-6は、「変数の項目名を集団項目名も付加した形で表現する。名前の間は、"."で連結する。また、もっとも上位の集団項目から表現する。例えば、IN-RECORD、IN-DATA1などの形式。」というまとめのルールを示す。

【0170】

日本語変換部311dは、まとめ処理で編集されたプログラム中間情報を受け取り、各タグを対応する日本語テンプレートに対応させて日本語情報を生成する処理部である。この日本語変換部311dによって生成された日本語情報は、プログラム概要記憶部323に格納され、ドキュメント生成部312によって、プログラム概要書に記載される。

【0171】

図34は、日本語変換部311dの処理を説明するための説明図である。同図は、"MOVE 0 TO DATA1"というMOVE文を「整数(0)を変数[DATA1]に代入する。」という日本語情報に変換する例を示している。

【0172】

ここで、日本語変換部311dは、「<move><ref>~(1)</ref><def>~(2)</def></move>」→「~(1)を~(2)に代入する。」という日本語テンプレートと、「<constant value="~(3)" type="int"/>」→「整数(~(3))」という日本語テンプレートと、「<var name="~(4)"/>」→「変数[~(4)]」という日本語テンプレートを用いて変換を行っている。また、図35-1および図35-2は、その他の日本語テンプレートの例を示す図である。

【0173】

次に、プログラム概要生成部311が生成するプログラム概要の例について説明する。図36は、プログラムサンプルを示す図である。プログラム概要生成部311は、同図に示したプログラムサンプルの構文解析情報をプログラム中間情報に変換し、データおよびステートメントに重要度を付加する。

【0174】

10

20

30

40

50

例えば、図 3 6 の [1] で示される変数 E N D - F L A G は、セクション繰り返し処理の終了条件であり、処理の分岐であるので、重要度は「 1 」となり、図 3 6 の [2] で示される変数 I N - C O U N T は、「 D - 1 」から「 D - 3 」の分類に該当しないので、重要度は「 4 」となる。また、図 3 6 の [3] で示される W R I T E 文は、ファイル入出力に関わるので、重要度は「 1 」となる。

【 0 1 7 5 】

そして、プログラム概要生成部 3 1 1 のまとめ管理部 3 1 1 c が、データおよびステートメントに重要度が付加されたプログラム中間情報に対して図 3 5 - 1 ~ 図 3 5 - 6 に示したまとめのルールを適用してまとめ処理を行う。

【 0 1 7 6 】

図 3 7 は、まとめ管理部 3 1 1 c が利用するまとめ処理プログラムのリストを示す図である。まとめ管理部 3 1 1 c は、図 3 1 に示したインタフェースを用いてまとめ処理プログラムから図 3 7 に示す情報を取得する。

【 0 1 7 7 】

また、図 3 8 は、まとめ処理プログラムの実行順序を示す図である。まとめ管理部 3 1 1 c は、図 3 7 に示したまとめ処理プログラムのリストに基づいて、図 3 8 に示す実行順序でまとめ処理プログラムを実行する。

【 0 1 7 8 】

すなわち、まとめ管理部 3 1 1 c は、局所的な処理として A R _ P 1、A R _ P 2、A R _ P 3 および A R _ P 4 を順に実行し、その後、全体的な処理として A R _ A 1 および A R _ A 2 を実行する。

【 0 1 7 9 】

図 3 9 は、まとめ処理終了時のプログラム中間情報を示す図である。日本語変換部 3 1 1 d は、図 3 9 に示すプログラム中間情報に対して図 3 4、図 3 5 - 1 および図 3 5 - 2 に示した日本語テンプレートを適用し、図 4 0 に示す日本語情報を生成する。

【 0 1 8 0 】

また、ドキュメント生成部 3 1 2 は、図 4 0 に示す日本語情報を用いて、図 4 1 に示すようなプログラム概要書を生成する。

【 0 1 8 1 】

上述してきたように、本実施例 3 では、プログラム概要生成部 3 1 1 が、構文解析部 1 1 1 により生成された構文解析情報をプログラム中間情報に変換し、まとめ処理記憶部 3 2 1 に記憶されたまとめ処理プログラムと日本語テンプレート記憶部 3 2 2 に記憶された日本語テンプレートとを用いてプログラム中間情報からプログラム概要を生成することとしたので、プログラムの処理内容の理解を容易し、理解に要する時間を短縮することができる。

【実施例 4】

【 0 1 8 2 】

ところで、実施例 1 ~ 実施例 3 では、一つのプログラムを解析してその処理概要をプログラム仕様書として生成する仕様書生成装置について説明したが、バッチ処理システムでは、複数のプログラムを順に実行することによって所定の処理を行うことも多い。

【 0 1 8 3 】

このような複数のプログラムを実行するバッチ処理システムでは、各プログラムの処理概要を理解することも重要であるが、バッチ処理全体としての処理概要を理解することが重要となる。

【 0 1 8 4 】

そこで、本実施例 4 では、バッチ処理全体の処理の概要を抽出して仕様書を生成する仕様書生成装置について説明する。具体的には、本実施例 4 に係る仕様書生成装置は、バッチ処理全体における入力データと出力データ、および主要なプログラムを抽出し、バッチジョブ全体の処理概要を示す仕様書を作成する。

【 0 1 8 5 】

10

20

30

40

50

まず、本実施例 4 に係る仕様書生成装置の構成について説明する。図 4 2 は、本実施例 4 に係る仕様書生成装置の構成を示す機能ブロック図である。なお、ここでは説明の便宜上、図 1 に示した各部と同様の役割を果たす機能部については同一符号を付すこととしてその詳細な説明を省略する。

【0186】

同図に示すように、この仕様書生成装置 400 の制御部 410 は、図 1 に示した制御部 110 が有する機能部に加えてバッチジョブ記述言語構文解析部 411 と、ジョブステップ入出力データ抽出部 412 と、全体入出力情報抽出部 413 と、主要プログラム判定部 414 とを有する。また、処理概要生成部 116 の代わりに処理概要生成部 415 を有し、ドキュメント生成部 117 の代わりにドキュメント生成部 416 を有する。

10

【0187】

バッチジョブ記述言語構文解析部 411 は、バッチジョブ記述言語ソースを読み込み、各ジョブステップにおける呼出プログラム名や使用ファイル名の情報を抽出したバッチジョブ解析情報を生成する処理部である。

【0188】

ジョブステップ入出力データ抽出部 412 は、バッチジョブ記述言語構文解析部 411 が生成したバッチジョブ解析情報から、各ジョブステップごとに、ファイルが入力に使用されているか、出力に使用されているか、あるいは削除されているかを判定し、判定した結果をジョブステップ入出力情報として生成する処理部である。

【0189】

全体入出力情報抽出部 413 は、ジョブステップ入出力データ抽出部 412 が生成したジョブステップ入出力情報と、入出力データ抽出部 114 が生成した入出力情報リストとを用いて、ジョブ全体として入力に使用されるファイルと出力に使用されるファイルを決定する処理部である。

20

【0190】

主要プログラム判定部 414 は、全体入出力情報抽出部 413 が決定したジョブ全体としての入出力ファイルを使用するジョブステップを抽出し、抽出したジョブステップで呼び出しているプログラムの一覧を生成する処理部である。

【0191】

また、この主要プログラム判定部 414 は、抽出したジョブステップで呼び出しているプログラムの一覧を生成する場合に、呼び出しているプログラムがユーティリティプログラムである場合には、プログラムの一覧から除外する。

30

【0192】

処理概要生成部 415 は、バッチジョブに含まれる各プログラムの処理概要情報を生成するとともに、全体入出力情報抽出部 413 が決定した入出力ファイルと、主要プログラム判定部 414 が生成したプログラムの一覧とを組み合わせるバッチジョブ処理概要を生成する処理部である。

【0193】

このように、この処理概要生成部 415 が、各処理プログラムの処理概要情報に加えて、全体入出力情報抽出部 413 が決定した入出力ファイルと主要プログラム判定部 414 が生成したプログラムの一覧とを用いてバッチジョブ処理概要を生成することによって、仕様書生成装置 400 は、プログラム仕様書に加えてジョブ処理全体の処理概要書を作成することができる。

40

【0194】

ドキュメント生成部 416 は、処理概要生成部 415 が生成した処理概要情報を用いてプログラム仕様書を生成するとともに、バッチジョブ処理概要を所定の形式に整形し、ジョブ全体の処理概要書を生成する処理部である。

【0195】

次に、ジョブステップ入出力データ抽出部 412 による入出力判定処理の処理手順について説明する。図 4 3 は、ジョブステップ入出力データ抽出部 412 による入出力判定処

50

理の処理手順を示すフローチャートである。

【0196】

同図に示すように、このジョブステップ入出力データ抽出部412は、バッチジョブ記述言語構文解析部411が生成したバッチジョブ解析情報からジョブステップのジョブ記述解析情報を作成する（ステップS411）。

【0197】

そして、ジョブステップのジョブ記述解析情報を解析してファイルが新規作成または追記で割り当てられているか否かを調べ（ステップS412）、ファイルが新規作成または追記で割り当てられている場合には、そのファイルを出力ファイルと判定する（ステップS41a）。

10

【0198】

一方、ファイルが新規作成または追記で割り当てられていない場合には、ジョブステップの終了時にファイルが削除されているか否かを調べ（ステップS413）、ファイルが削除されている場合には、そのファイルを削除されるファイルと判定する（ステップS414）。

【0199】

そして、ジョブステップで呼び出されるプログラムの構文解析情報を取得し（ステップS415）、ファイルに対して書き込みを行う文があるか否かを調べ（ステップS416）、書き込みを行う文がある場合には、そのファイルを出力ファイルと判定する（ステップS41a）。

20

【0200】

一方、ファイルに対して書き込みを行う文がない場合には、ファイルに対して読みこみを行う文があるか否かを調べ（ステップS417）、読み込みを行う文がある場合には、そのファイルを入力ファイルと判定し（ステップS418）、読み込みを行う文がない場合には、そのファイルを入出力なしと判定する（ステップS419）。

【0201】

このように、このジョブステップ入出力データ抽出部412は、ジョブ記述解析情報およびプログラム構文解析情報を用いてファイルが入力に使用されているのか出力に使用されているのかなどを判定することによって、各ジョブステップでのジョブステップ入出力情報を生成することができる。

30

【0202】

次に、全体入出力情報抽出部413によるジョブ全体としてのファイル入出力判定処理の処理手順について説明する。図44は、全体入出力情報抽出部413によるジョブ全体としてのファイル入出力判定処理の処理手順を示すフローチャートである。

【0203】

同図に示すように、この全体入出力情報抽出部413は、まず対象ジョブステップを先頭のジョブステップとし（ステップS421）、判定の対象ファイルをそのジョブステップで読んでいるか否かを調べる（ステップS422）。

【0204】

その結果、そのジョブステップで対象ファイルを読んでいる場合には、そのファイルが参照済か否かを調べ（ステップS423）、参照済でない場合には、そのファイルを入力ファイルとし、かつそのファイルを参照済とする（ステップS424）。

40

【0205】

そして、対象ファイルをそのジョブステップで書き込んでいるか否かを調べ（ステップS425）、そのジョブステップで対象ファイルを書き込んでいる場合には、そのファイルを出力ファイルとし、かつそのファイルを参照済とする（ステップS426）。

【0206】

そして、対象ファイルをそのジョブステップで削除しているか否かを調べ（ステップS427）、そのジョブステップで対象ファイルを削除している場合には、そのファイルが出力ファイルになっていれば消し、かつそのファイルを参照済とする（ステップS428

50

）。

【0207】

そして、ジョブステップが最終ジョブステップであるか否かを調べ（ステップS429）、最終ジョブステップでない場合には、判定の対象ジョブステップを次のジョブステップとし（ステップS42a）、ステップS422に戻り、最終ジョブステップである場合には、処理を終了する。

【0208】

このように、この全体入出力情報抽出部413がバッチジョブ全体としてのファイル入出力を判定することによって、仕様書生成装置400は、バッチジョブ全体の入力ファイルと出力ファイルを記載した仕様書を作成することができる。

10

【0209】

次に、主要プログラム判定部414によるユーティリティ除外処理の処理手順について説明する。ここで、ユーティリティ除外処理とは、主要プログラム判定部414が主要プログラムを抽出する場合に、ユーティリティと呼ばれる一般的なプログラムを除外してプログラムを抽出するための処理である。

【0210】

図45は、ユーティリティ除外処理の処理手順を示すフローチャートである。同図に示すように、このユーティリティ除外処理は、まず、全体としての入力ファイルを読み込んでいるジョブステップを対象ジョブステップとする（ステップS441）。

【0211】

そして、入力ファイルを読んでいるジョブステップはユーティリティであるか否かを調べ（ステップS442）、入力ファイルを読んでいるジョブステップがユーティリティでない場合には、そのジョブステップが呼び出しているプログラムを主要プログラムとする（ステップS44a）。

20

【0212】

また、入力ファイルを読んでいるジョブステップがユーティリティである場合には、そのジョブステップの出力ファイルは一つだけであるか否かを調べ（ステップS443）、一つだけでない場合には、ユーティリティ除外処理の結果を空とする（ステップS449）。

【0213】

一方、そのジョブステップの出力ファイルが一つだけである場合には、そのジョブステップからそのファイルを削除するジョブステップもしくは終了ジョブステップまでの間で、そのファイルを入力しているジョブステップを対象のジョブステップとし（ステップS444）、対象となるジョブステップがあるか否かを調べ（ステップS445）、対象となるジョブステップがない場合には、ユーティリティ除外処理の結果を空とする（ステップS449）。

30

【0214】

これに対して、対象となるジョブステップがある場合には、それぞれのジョブステップを対象としたユーティリティ除外処理を行い（ステップS446）、全ての対象ジョブステップで主要プログラムが得られなかったかを調べ（ステップS447）、全ての対象ジョブステップで主要プログラムが得られなかった場合には、ユーティリティ除外処理の結果を空とし（ステップS449）、主要プログラムが得られたジョブステップがある場合には、全ての対象ジョブステップでの処理結果をまとめて主要プログラムとする（ステップS448）。

40

【0215】

このように、主要プログラム判定部414がユーティリティ除外処理によってユーティリティを除外することによって、仕様書生成装置400は、バッチジョブの主要プログラムからユーティリティプログラムを取り除くことができる。

【0216】

次に、仕様書生成装置400が生成するバッチジョブ全体の処理概要の例について説明

50

する。図 4 6 は、バッチジョブ記述の一例を示す図である。同図は四つのジョブステップ「STEP 1」～「STEP 4」から構成されるバッチジョブを示している。

【0217】

この例では、まず最初のジョブステップ「STEP 1」でプログラム「PROGA」を呼び出す。このとき、「PROGA」が使用するファイルとして、外部名「IN01」にファイル「DENPYO1」を、外部名「OT01」にファイル「WORK1」を割り当てている。

【0218】

これは、プログラム「PROGA」内で、「IN01」、「OT01」という名前で参照している入出力先に、それぞれ「DENPYO1」と「WORK1」を割り当てるということを意味している。ファイルのアクセス方法はDISPと呼ばれるパラメータで記述しており、SHRなら既存のファイルの読み書き、(NEW, PASS)なら新規作成を意味している。

【0219】

また、ジョブステップSTEP 2～STEP 4でも同様のファイルの割当を行っており、STEP 2およびSTEP 3のDISPパラメータの(OLD, DELETE)は、既存ファイルの読み込みと終了時の削除を意味している。

【0220】

図 4 7 は、図 4 6 に示したバッチジョブ記述からバッチジョブ記述言語構文解析部 4 1 1 が生成するバッチジョブ解析情報を示す図である。ジョブステップ入出力データ抽出部 4 1 2 は、このバッチジョブ解析情報を用い各ジョブステップで使用する入出力ファイルを判定する。

【0221】

図 4 8 は、図 4 6 に示したバッチジョブ記述から生成されるバッチジョブ全体の処理概要を示す図である。同図に示すように、仕様書生成装置 4 0 0 は、バッチ処理全体の入力ファイルとして「DENPYO1」を抽出し、出力ファイルとして「SYUKEI1」を抽出している。

【0222】

また、それらのファイルを扱う処理を主要なプログラムとしてPROGAおよびPROGBを抽出している。これらの情報を出力することで、このバッチ処理が全体としてどんなデータを扱い、何を行っているのかを明確にすることができる。また、主要なプログラムについては、処理概要情報記憶部 1 2 2 に記憶された処理概要情報を用いて、その処理概要を付加することもできる。

【0223】

上述してきたように、本実施例 4 では、バッチジョブ記述言語構文解析部 4 1 1 がバッチジョブ記述言語ソースからバッチジョブ解析情報を生成し、ジョブステップ入出力データ抽出部 4 1 2 がバッチジョブ記述言語構文解析部 4 1 1 により生成されたバッチジョブ解析情報からジョブステップ入出力情報を生成し、全体入出力情報抽出部 4 1 3 がジョブステップ入出力データ抽出部 4 1 2 により生成されたジョブステップ入出力情報と、入出力データ抽出部 1 1 4 が生成した入出力情報リストとを用いて、ジョブ全体として入出力ファイルを決定し、主要プログラム判定部 4 1 4 が全体入出力情報抽出部 4 1 3 により決定されたジョブ全体としての入出力ファイルを使用するジョブステップで呼び出しているプログラムの一覧を生成することとしたので、バッチジョブ処理全体の概要の把握を容易にすることができる。

【実施例 5】

【0224】

オンラインシステムを対象として概要を把握するためのドキュメントとしては、画面遷移図が有効である。そこで、本実施例 5 では、オンラインシステムを対象として画面遷移図を自動生成する仕様書生成装置について説明する。

【0225】

まず、本実施例 5 に係る仕様書生成装置の構成について説明する。図 49 は、本実施例 5 に係る仕様書生成装置の構成を示す機能ブロック図である。なお、ここでは説明の便宜上、図 1 に示した各部と同様の役割を果たす機能部については同一符号を付すこととしてその詳細な説明を省略する。

【0226】

同図に示すように、この仕様書生成装置 500 は、制御部 510 と、制御部 510 が使用する情報や制御部 510 による処理の途中結果などを記憶する記憶部 520 とを有する。制御部 510 は、図 1 に示した制御部 110 が有する機能部に加えて画面定義体解析部 511 と、画面遷移データ生成部 512 とを有し、ドキュメント生成部 117 の代わりにドキュメント生成部 513 を有する。

10

【0227】

また、記憶部 520 は、図 1 に示した記憶部 120 が有する記憶部に加えて、画面遷移情報記憶部 521 と、レイアウト情報記憶部 522 と、画面遷移データ記憶部 523 とを有する。

【0228】

画面定義体解析部 511 は、画面定義体を読み込み、各画面のレイアウト情報と、画面間の遷移条件を抽出した振り分け定義情報とを生成し、また、振り分け定義情報から、どの画面からどの画面またはプログラム呼出に遷移し、そのときの遷移条件は何かといった画面遷移情報を生成する処理部である。

【0229】

画面遷移情報記憶部 521 は、画面定義体解析部 511 により生成された画面遷移情報を記憶する記憶部であり、レイアウト情報記憶部 522 は、画面定義体解析部 511 により生成されたレイアウト情報を記憶する記憶部である。

20

【0230】

図 50 は、画面定義体の一例を示す図である。同図に示すように、レイアウト情報は、「LAYOUT」と「LAYEND」の間に定義された情報から生成され、画面遷移情報は、DIST 文によって定義された情報から生成される。

【0231】

図 51 は、画面遷移情報の一例を示す図である。同図に示すように、この画面遷移情報は、遷移元画面と、遷移条件と、遷移先画面と、遷移先種別とを対応させた情報である。この例では、遷移元画面「M05000」は、「選択処理＝1」の遷移条件のもとに「M05010」の画面に遷移し、「選択処理＝2」の遷移条件のもとに「PG0001」のプログラムに遷移する。

30

【0232】

画面遷移データ生成部 512 は、画面定義体解析部 511 により生成された画面遷移情報と、処理概要生成部 116 により生成された処理概要情報とを用いて、特定の画面を起点として画面遷移図全体を構造化した画面遷移データを生成する処理部である。

【0233】

すなわち、この画面遷移データ生成部 512 は、画面遷移情報をもとにして、特定の画面を起点として遷移元から遷移先へと順に繋いで、全体の遷移情報を階層構造の形で表した画面遷移データを作成する。

40

【0234】

例えば、「画面 A から画面 B」への画面遷移情報と、「画面 B からプログラム C」への画面遷移情報があったとすると、画面 B が共通部分になり、「画面 A → 画面 B → プログラム C」という一連の流れができる。これをすべての画面遷移情報に対して行うことによって、ひとつの巨大な画面遷移データを生成することができる。

【0235】

また、この画面遷移データ生成部 512 は、プログラムから生成した処理概要情報からも個々のプログラム概要文と各プログラムからの入出力ファイルデータを抽出したデータを作成し、先の画面遷移データとマージし、図 52 に示すような画面遷移データを生成す

50

る。例えば、プログラム C からファイル X への書き込みがある場合、先の画面遷移データと合わせて「画面 A → 画面 B → プログラム C → ファイル X」という流れが出来上がる。

【0236】

画面遷移データ記憶部 523 は、画面遷移データ生成部 512 により生成された画面遷移データを記憶する記憶部である。

【0237】

ドキュメント生成部 513 は、処理概要情報からプログラム仕様書を生成するとともに、画面遷移図データを画面遷移図として所定の形式に変換して画面遷移図書を生成する。

【0238】

図 53-1 は、画面遷移図の一例を示す図である。同図では、例えば「画面 M0500 → 画面 M05010 → プログラム M05040 → MOD パラメータ DB」といった流れを示してしる。また、図 53-2 は、画面遷移図の他の例を示す図であり、この例では、画面遷移図にプログラム概要が埋め込まれている。 10

【0239】

また、このドキュメント生成部 513 は、出力設定情報として指定された画面数／頁などの情報を用いてページ制御を行う。ここで、1 ページに収まらなかった遷移先は、遷移先が何ページのどこにあるかという位置情報を遷移図の終端に記述しておくことで、遷移先が人目でわかるようにする。または、ページ制御をまったく行わずに 1 ページあたりに収容可能な画面数を指定せずに、ひとつの大きな画面遷移図を生成することも可能である。 20

【0240】

さらに、画面遷移図生成時に、画面遷移図中の各画面から別に生成していた画面レイアウト図へのリンクをはることによって、全体の流れと各画面の詳細を関連づけておくことも可能である。

【0241】

また、このドキュメント生成部 513 は、画面定義体解析部 511 により生成されたレイアウト情報から作成される画面レイアウト図も仕様書として出力することもできる。

【0242】

上述してきたように、本実施例 5 では、画面定義体解析部 511 が画面定義体から画面遷移情報を生成し、画面遷移データ生成部 512 が画面定義体解析部 511 により生成された画面遷移情報と処理概要生成部 116 により生成された処理概要情報とを用いて、特定の画面を起点として画面遷移図全体を構造化した画面遷移データを生成し、ドキュメント生成部 513 が画面遷移データ生成部により生成された画面遷移図データを所定の形式に変換して画面遷移図書を生成することとしたので、オンラインシステムの処理概要の把握を容易にすることができる。 30

【実施例 6】

【0243】

プログラムの処理概要を把握するためには、実施例 1 で説明した仕様書生成装置 100 が生成するプログラム仕様書が有効であるが、プログラムの処理概要を把握した後に、プログラムの処理構造の詳細を把握する必要がある場合も多い。そこで、本実施例 6 では、プログラムの呼出構造と入出力情報の関係の詳細を処理構造図として生成する仕様書生成装置について説明する。 40

【0244】

まず、本実施例 6 に係る仕様書生成装置の構成について説明する。図 54 は、本実施例 6 に係る仕様書生成装置の構成を示す機能ブロック図である。なお、ここでは説明の便宜上、図 1 に示した各部と同様の役割を果たす機能部については同一符号を付すこととしてその詳細な説明を省略する。

【0245】

同図に示すように、この仕様書生成装置 600 は、制御部 610 と、制御部 610 による処理の途中結果などを記憶する記憶部 620 とを有する。制御部 610 は、図 1 に示し 50

た制御部 110 が有する機能部に加えて辞書情報合成部 611 と、処理構造生成部 612 とを有し、ドキュメント生成部 117 の代わりにドキュメント生成部 613 を有する。

【0246】

記憶部 620 は、図 1 に示した記憶部 120 が有する記憶部に加えて処理構造情報記憶部 622 を有し、サブルーチン情報記憶部 122 の代わりにサブルーチン情報記憶部 621 を有する。

【0247】

辞書情報合成部 611 は、辞書情報を参照してプログラム名や変数名などに対する補足情報を併記したサブルーチン情報を生成し、サブルーチン情報記憶部 621 に格納する処理部である。すなわち、サブルーチン情報記憶部 621 は、サブルーチン情報記憶部 122 が記憶する情報に加えて補足情報を記憶する。

10

【0248】

処理構造生成部 612 は、サブルーチン情報記憶部 621 に記憶されたサブルーチン情報を表形式に展開して処理構造情報記憶部 622 に格納し、利用者のオプション指定によって指定された情報を処理構造情報記憶部 622 に格納された表形式のサブルーチン情報に付加する処理部である。

【0249】

すなわち、処理構造情報記憶部 622 は、表形式に展開されたサブルーチン情報を記憶する記憶部である。また、オプション指定としては、サブルーチン呼出条件、引数、ジャンプ先、文の分類などの情報の処理構造図への付加を指定することができる。

20

【0250】

ドキュメント生成部 613 は、処理概要情報からプログラム仕様書を生成するとともに、処理構造情報記憶部 622 に記憶された表形式のサブルーチン情報から処理構造図を出力する。

【0251】

次に、辞書情報合成部 611 の処理手順について説明する。図 55 は、辞書情報合成部 611 の処理手順を示すフローチャートである。同図に示すように、この辞書情報合成部 611 は、プログラム内のプログラム名、変数名、サブルーチン名などを取得し（ステップ S611）、取得した情報を元に、辞書情報を参照して補足情報を検索する（ステップ S612）。

30

【0252】

そして、補足情報が得られたか否かを判定し（ステップ S613）、補足情報が得られた場合には、補足情報をサブルーチン情報記憶部 621 のサブルーチン情報に付加し、補足情報が得られなかった場合には、そのまま処理を終了する（ステップ S614）。

【0253】

このように、この辞書情報合成部 611 が辞書情報を参照し、辞書情報の参照によって得られた補足情報をサブルーチン情報記憶部 621 のサブルーチン情報に付加することによって、処理構造図に補足情報を加えることができる。

【0254】

次に、処理構造生成部 612 の処理手順について説明する。図 56 は、処理構造生成部 612 の処理手順を示すフローチャートである。同図に示すように、この処理構造生成部 612 は、サブルーチン情報記憶部 621 に記憶されたサブルーチン情報を表形式に展開し、処理構造情報記憶部 622 に格納する（ステップ S621）。

40

【0255】

そして、オプション指定を読み込み、利用者が条件文表示のオプションを指定しているか否かを調べ（ステップ S622）、利用者が条件文表示のオプションを指定している場合には、サブルーチン呼出条件を検索し（ステップ S623）、処理構造情報記憶部 622 のサブルーチン情報に付加する（ステップ S624）。

【0256】

また、利用者が呼出プログラムの引数情報表示のオプションを指定しているか否かを調

50

べ（ステップ S 6 2 5）、利用者が呼出プログラムの引数情報表示のオプションを指定している場合には、値を設定しているプログラム引数を取得し（ステップ S 6 2 6）、処理構造情報記憶部 6 2 2 のサブルーチン情報に引数情報を付加する（ステップ S 6 2 4）。

【0 2 5 7】

また、利用者が呼出プログラムの制御文表示のオプションを指定しているか否かを調べ（ステップ S 6 2 8）、利用者が制御文表示のオプションを指定している場合には、ジャンプ文のジャンプ先の情報を取得し（ステップ S 6 2 9）、処理構造情報記憶部 6 2 2 のサブルーチン情報にジャンプ先の情報を付加する（ステップ S 6 2 a）。

【0 2 5 8】

また、利用者がサブルーチン内の文の分類別表示のオプションを指定しているか否かを調べ（ステップ S 6 2 b）、利用者がサブルーチン内の文の分類別表示のオプションを指定している場合には、サブルーチン内で文の数を「入力、出力、計算、代入、条件、呼出、その他」の 7 種類に分類して数を数え（ステップ S 6 2 c）、処理構造情報記憶部 6 2 2 のサブルーチン情報に種類別の文の数を付加する（ステップ S 6 2 d）。

【0 2 5 9】

このように、この処理構造生成部 6 1 2 が、サブルーチン情報記憶部 6 2 1 に記憶されたサブルーチン情報を表形式に展開し、展開したサブルーチン情報にオプション指定に基づくオプション情報を付加することによって、オプション情報付の処理構造図を生成することができる。

【0 2 6 0】

次に、本実施例 6 に係る仕様書生成装置 6 0 0 が生成する処理構造図について説明する。図 5 7 - 1 および図 5 7 - 2 は、処理構造図を生成する対象プログラムを示す図である。図 5 7 - 1 は呼出元のプログラムを示し、図 5 7 - 2 は呼出先のプログラムを示す。

【0 2 6 1】

図 5 7 - 1 に示すように、呼出元のプログラムは、プログラム名が P R O G A（P R O G R A M - I D P R O G A）であり、R E A D - S E C T と M A I N - S E C T の二つのセクションを有する。また、M A I N - S E C T は、P R O G W を呼び出し、図 5 7 - 2 は、P R O G W を示している。

【0 2 6 2】

図 5 8 は、図 5 7 - 1 および図 5 7 - 2 に示したプログラムから生成される処理構造図の一例を示す図である。同図において、S E C T I O N（1 層目）～S E C T I O N（3 層目）の列は、プログラムの呼出構造を示し、読み取りファイルおよび書き込みファイルの列は、呼出構造に対応した入出力ファイルを示している。

【0 2 6 3】

具体的には、この処理構造図は、P R O G A は 2 層目のセクションとして R E A D - S E C T と M A I N - S E C T を有し、R E A D - S E C T は I N F I L E を入力ファイルとして使用することを示している。さらに、M A I N - S E C T は C A L L 文で P R O G W を呼び出しており、P R O G W 内で O U T F I L E を出力ファイルとして使用することを示している。

【0 2 6 4】

上述してきたように、本実施例 6 では、辞書情報合成部 6 1 1 が辞書情報から補足情報を検索してサブルーチン情報に付加し、処理構造生成部 6 1 2 がサブルーチン情報を表形式に展開するとともに、展開したサブルーチン情報にオプション指定で指定された情報を付加し、ドキュメント生成部 6 1 3 が表形式のサブルーチン情報から補足情報やオプション情報付の処理構造図を生成することとしたので、プログラムの呼出構造と入出力情報との詳細な関係の把握を容易にすることができる。

【0 2 6 5】

すなわち、処理構造図には、呼出先のサブルーチンを全て辿った入出力情報が表形式で示されているため、あるサブルーチンが下位のサブルーチンも含めてアクセスする可能性のある入出力情報の把握を容易にすることができる。

10

20

30

40

50

【実施例 7】

【0266】

実施例 6 では、プログラムの処理構造の把握を容易にする処理構造図を生成する仕様書生成装置 600 について説明したが、プログラムの実行時の動きを把握するためには、プログラムの静的な処理構造だけでなく、プログラムを実行して得られる実行ログ情報が有効となる。そこで、本実施例 7 では、プログラムの処理構造図に実行ログ情報を付加する仕様書生成装置について説明する。

【0267】

まず、本実施例 7 に係る仕様書生成装置の構成について説明する。図 59 は、本実施例 7 に係る仕様書生成装置の構成を示す機能ブロック図である。なお、ここでは説明の便宜上、図 54 に示した各部と同様の役割を果たす機能部については同一符号を付すこととしてその詳細な説明を省略する。 10

【0268】

同図に示すように、この仕様書生成装置 700 は、制御部 710 と記憶部 720 とを有する。制御部 710 は、図 54 に示した制御部 610 が有する機能部に加えて実行ログ情報解析部 711 と、実行情報合成部 712 とを有し、ドキュメント生成部 613 の代わりにドキュメント生成部 713 を有する。また、記憶部 720 は、図 54 に示した記憶部 620 の処理構造情報記憶部 622 の代わりに処理構造情報記憶部 721 を有する。

【0269】

実行ログ情報解析部 711 は、プログラムを実行して得られる実行ログからサブルーチン単位での実行回数情報を抽出する処理部であり、抽出した実行回数を実行情報合成部 712 に渡す。 20

【0270】

実行情報合成部 712 は、実行ログ情報解析部 711 から受け取ったサブルーチン単位での実行回数に基づいて、処理構造図の各サブルーチンに対応する領域の色分けを行う処理部であり、色分けした情報を処理構造情報記憶部 721 に格納する。すなわち、処理構造情報記憶部 721 は、処理構造情報記憶部 622 が記憶する情報に加えて各サブルーチンの色情報を有する。

【0271】

ドキュメント生成部 713 は、処理概要情報からプログラム仕様書を生成するとともに、処理構造情報記憶部 721 に記憶された色分け情報付のサブルーチン情報から、色分け付の処理構造図を出力する処理部である。 30

【0272】

次に、実行ログ情報解析部 711 の処理手順について説明する。図 60 は、実行ログ情報解析部 711 の処理手順を示すフローチャートである。同図に示すように、この実行ログ情報解析部 711 は、プログラムの実行ログがあるか否かを判定する（ステップ S711）。 40

【0273】

そして、実行ログがある場合には、実行ログからサブルーチン単位の実行回数を取得し（ステップ S712）、取得した実行回数をサブルーチン名とともに実行情報合成部 712 に渡す（ステップ S713）。一方、実行ログがない場合には、そのまま処理を終了する。 40

【0274】

次に、実行情報合成部 712 の処理手順について説明する。図 61 は、実行情報合成部 712 の処理手順を示すフローチャートである。同図に示すように、この実行情報合成部 712 は、実行ログ情報解析部 711 からサブルーチン名と実行回数を受け取り（ステップ S721）、受け取ったサブルーチン名を用いて処理構造情報記憶部 721 を検索する（ステップ S722）。 50

【0275】

そして、処理構造情報記憶部 721 からサブルーチンが検索できたか否かを調べ（ステ 50

ップ S 7 2 3)、サブルーチンが検索できた場合には、そのサブルーチンの処理構造図における領域を実行回数に応じて色づけする情報を処理構造情報記憶部 7 2 1 に加える(ステップ S 7 2 4)。一方、サブルーチンが検索できなかった場合には、そのまま処理を終了する。

【0 2 7 6】

このように、実行ログ情報解析部 7 1 1 が実行ログを解析してサブルーチンの実行回数を抽出し、実行情報合成部 7 1 2 がサブルーチンの実行回数に基づいて処理構造図におけるサブルーチンの領域を色分けすることによって、色分け付の処理構造図を生成することができる。

【0 2 7 7】

次に、本実施例 7 に係る仕様書生成装置 7 0 0 が生成する処理構造図について説明する。図 6 2 は、図 5 7-1 および図 5 7-2 に示したプログラムから生成される処理構造図の一例を示す図である。

【0 2 7 8】

同図に示すように、この処理構造図では、READ-SECT と MAIN-SECT に対応する領域が実行回数に応じて異なる方向に網掛けされている。実際には、これらの領域に対して、網掛けでなく異なる色が割り付けられる。

【0 2 7 9】

また、この処理構造図は、サブルーチン呼出条件、引数、ジャンプ先、文の分類などの情報を処理構造図に付加するオプション指定を利用者が行った場合を示しており、例えば、READ-SECT には、「入力：1、計算：1、代入：1」といった分類別ステートメント数が出力されている。

【0 2 8 0】

上述してきたように、本実施例 7 では、実行ログ情報解析部 7 1 1 が実行ログを解析してサブルーチンの実行回数を抽出し、実行情報合成部 7 1 2 が実行ログ情報解析部 7 1 1 により抽出されたサブルーチン実行回数に基づいて色分けした情報を処理構造情報記憶部 7 2 1 に付加し、ドキュメント生成部 7 1 3 が実行情報合成部 7 1 2 により色分けられたサブルーチン情報を用いて色分け付の処理構造図を生成することとしたので、プログラムの実行時の動作の理解を容易にすることができる。

【0 2 8 1】

なお、本実施例 1~7 の仕様書生成装置で生成されるプログラム仕様書などの仕様書では、仕様書内のデータまたはオブジェクトにハイパーリンクを設けることによって、仕様書間の相互参照、または他のドキュメントなどへの参照を可能にすることもできる。

【0 2 8 2】

例えば、画面遷移図の場合、図中のプログラム名をクリックするとプログラム概要書へジャンプしたり、画面図形をクリックすると他のツールで生成した画面レイアウト図へジャンプしたりすることを可能とすることができる。

【0 2 8 3】

また、本実施例 1~7 では、仕様書生成装置について説明したが、この仕様書生成装置が有する構成をソフトウェアによって実現することで、同様の機能を有する仕様書生成プログラムを得ることができる。そこで、この仕様書生成プログラムを実行するコンピュータシステムについて説明する。

【0 2 8 4】

図 6 3 は、本実施例 1~7 に係る仕様書生成プログラムを実行するコンピュータシステムを示す図である。同図に示すように、このコンピュータシステム 8 0 0 は、本体部 8 0 1 と、本体部 8 0 1 からの指示により表示画面 8 0 2 a に情報を表示するディスプレイ 8 0 2 と、このコンピュータシステム 8 0 0 に種々の情報を入力するためのキーボード 8 0 3 と、ディスプレイ 8 0 2 の表示画面 8 0 2 a 上の任意の位置を指定するマウス 8 0 4 と、LAN 8 0 6 または広域エリアネットワーク(WAN)に接続する LAN インタフェースと、公衆回線 8 0 7 に接続するモデムとを有する。ここで、LAN 8 0 6 は、他のコン

10

20

30

40

50

ピュータシステム（PC）８１１、サーバ８１２、プリンタ８１３などとコンピュータシステム８００とを接続している。

【０２８５】

また、図６４は、図６３に示した本体部８０１の構成を示す機能ブロック図である。同図に示すように、この本体部８０１は、CPU８２１と、RAM８２２と、ROM８２３と、ハードディスクドライブ（HDD）８２４と、CD-ROMドライブ８２５と、FDドライブ８２６と、I/Oインタフェース８２７と、LANインタフェース８２８と、モデム８２９とを有する。

【０２８６】

そして、このコンピュータシステム８００において実行される仕様書生成プログラムは、フロッピーディスク（FD）８０８、CD-ROM８０９、DVDディスク、光磁気ディスク、ICカードなどの可搬型記憶媒体に記憶され、これらの記憶媒体から読み出されてコンピュータシステム８００にインストールされる。 10

【０２８７】

あるいは、この仕様書生成プログラムは、LANインタフェース８２８を介して接続されたサーバ８１２のデータベース、他のコンピュータシステム（PC）８１１のデータベースなどに記憶され、これらのデータベースから読み出されてコンピュータシステム８００にインストールされる。

【０２８８】

そして、インストールされた仕様書生成プログラムは、HDD８２４に記憶され、RAM８２２、ROM８２３などを利用してCPU８２１により実行される。 20

【０２８９】

また、本実施例では、COBOLで開発されたプログラムから仕様書を生成する場合について説明したが、本発明はこれに限定されるものではなく、FORTRANやCなどの他のプログラミング言語で開発されたプログラムから仕様書を生成する場合にも同様に適用することができる。

【０２９０】

（付記１）ソースプログラムを解析して仕様書を生成する仕様書生成プログラムであって、

前記ソースプログラムを解析して該ソースプログラムのプログラム呼出構造とデータ入出力とを関連づけた入出力付構造情報を生成する入出力付構造情報生成手順と、 30

前記入出力付構造情報生成手順により生成された入出力付構造情報から一部の情報を抽出して前記ソースプログラムの処理概要情報を生成する処理概要情報生成手順と、

前記処理概要情報生成手順により生成された処理概要情報を用いて前記ソースプログラムのプログラム仕様書を生成する文書生成手順と

をコンピュータに実行させることを特徴とする仕様書生成プログラム。

【０２９１】

（付記２）前記処理概要情報生成手順は、特定の呼出階層のサブルーチンの情報と該サブルーチンに関連づけられた入出力情報とを入出力付構造情報から抽出して前記処理概要情報を生成し、 40

前記文書生成手順は、前記特定の呼出階層のサブルーチンの情報と該サブルーチンに関連づけられた入出力情報とを用いてプログラム仕様書を生成することを特徴とする付記１に記載の仕様書生成プログラム。

【０２９２】

（付記３）前記文書生成手順は、サブルーチンの呼出階層を列とし、各サブルーチンの名前をサブルーチンの呼出階層値に対応する列に表示する表形式の呼出構造図をプログラム仕様書に含めることを特徴とする付記２に記載の仕様書生成プログラム。

【０２９３】

（付記４）前記入出力付構造情報生成手順は、プログラム呼出構造にプログラム呼出条件を関連づけた入出力付構造情報を生成し、 50

前記処理概要情報生成手順は、プログラム呼出構造にプログラム呼出条件を関連づけた処理概要情報を生成し、

前記文書生成手順は、プログラム呼出構造にプログラム呼出条件を関連づけてプログラム仕様書を生成することを特徴とする付記 1 に記載の仕様書生成プログラム。

【0294】

(付記 5) 前記文書生成手順は、生成したプログラム仕様書の所定の位置に利用者によって追加されたコメントを抽出するコメント抽出手順と、

新たに生成するプログラム仕様書の所定の位置に、前記コメント抽出手順により抽出されたコメントを継承するコメント継承手順とを

さらにコンピュータに実行させることを特徴とする付記 1、2 または 4 に記載の仕様書生成プログラム。 10

【0295】

(付記 6) 前記ソースプログラムを構成する文をまとめることにより該ソースプログラムのプログラム概要情報を生成するプログラム概要情報生成手順と、

前記プログラム概要情報生成手順により生成されたプログラム概要情報から自然言語によるプログラム概要文を生成するプログラム概要文生成手順と、

前記プログラム概要文生成手順により生成されたプログラム概要文を用いて前記ソースプログラムのプログラム概要書を生成するプログラム概要書生成手順と

をさらにコンピュータに実行させることを特徴とする付記 1、2 または 4 に記載の仕様書生成プログラム。 20

【0296】

(付記 7) 前記プログラム概要情報生成手順は、前記ソースプログラムに含まれるデータの重要度を決定するデータ重要度決定手順と、

前記ソースプログラムを構成する文の重要度を前記データ重要度決定手順により決定されたデータの重要度を用いて決定する文重要度決定手順と、

前記ソースプログラムを構成する文を前記文重要度決定手順により決定された文の重要度を用いてまとめることによって前記プログラム概要情報を生成するまとめ処理手順と

をコンピュータに実行させることを特徴とする付記 6 に記載の仕様書生成プログラム。

【0297】

(付記 8) バッチジョブ記述言語で記載されたバッチジョブ記述からバッチジョブを構成するジョブステップの入出力情報を抽出するステップ入出力情報抽出手順と、 30

前記ステップ入出力情報抽出手順により抽出されたジョブステップの入出力情報に基づいて前記バッチジョブ全体の入力情報および出力情報を特定するジョブ入出力情報特定手順と、

前記ジョブ入出力情報特定手順により特定された入力情報を入力するかまたは出力情報を出力するジョブステップを特定し、特定したジョブステップで呼び出されるプログラムの情報を抽出するプログラム情報抽出手順と、

前記ジョブ入出力情報特定手順により特定された入力情報および出力情報、ならびに前記プログラム情報抽出手順により抽出されたプログラムの情報を用いて前記バッチジョブのバッチジョブ処理概要書を生成するバッチジョブ処理概要書生成手順と 40

をさらにコンピュータに実行させることを特徴とする付記 1、2 または 4 に記載の仕様書生成プログラム。

【0298】

(付記 9) 画面についての情報を定義した画面定義体を解析して画面遷移情報を作成する画面遷移情報作成手順と、

前記画面遷移情報作成手順により作成された画面遷移情報を用いて画面遷移図を生成する画面遷移図生成手順と

をさらにコンピュータに実行させることを特徴とする付記 1、2 または 4 に記載の仕様書生成プログラム。

【0299】

(付記 10) 前記画面定義体は、画面とプログラムとの間の遷移に関する情報を含み、
前記画面遷移情報作成手順により作成される画面遷移情報は、画面とプログラムとの間の遷移を含み、

画面とプログラムとの間の遷移を含む画面遷移情報と前記入出力付構造情報生成手順により生成された入出力付構造情報とをマージし、画面とプログラムとの遷移および該プログラムの入出力をマージしたマージ図面情報を作成するマージ手順をさらにコンピュータに実行させ、

前記画面遷移図生成手順は、前記マージ手順により作成されたマージ図面情報を用いて画面とプログラムとの遷移および該プログラムの入出力をマージした図面を生成すること
を特徴とする付記 9 に記載の仕様書生成プログラム。

10

【0300】

(付記 11) ソースプログラムを解析して仕様書を生成する仕様書生成プログラムを記録したコンピュータ読み取り可能な記録媒体であって、

前記ソースプログラムを解析して該ソースプログラムのプログラム呼出構造とデータ入出力とを関連づけた入出力付構造情報を生成する入出力付構造情報生成手順と、

前記入出力付構造情報生成手順により生成された入出力付構造情報から一部の情報を抽出して前記ソースプログラムの処理概要情報を生成する処理概要情報生成手順と、

前記処理概要情報生成手順により生成された処理概要情報を用いて前記ソースプログラムのプログラム仕様書を生成する文書生成手順と

をコンピュータに実行させる仕様書生成プログラムを記録したことを特徴とするコンピュータ読み取り可能な記録媒体。

20

【0301】

(付記 12) ソースプログラムを解析して仕様書を生成する仕様書生成装置であって、

前記ソースプログラムを解析して該ソースプログラムのプログラム呼出構造とデータ入出力とを関連づけた入出力付構造情報を生成する入出力付構造情報生成手段と、

前記入出力付構造情報生成手段により生成された入出力付構造情報から一部の情報を抽出して前記ソースプログラムの処理概要情報を生成する処理概要情報生成手段と、

前記処理概要情報生成手段により生成された処理概要情報を用いて前記ソースプログラムのプログラム仕様書を生成する文書生成手段と

を備えたことを特徴とする仕様書生成装置。

30

【0302】

(付記 13) ソースプログラムを解析して仕様書を生成する仕様書生成方法であって、

前記ソースプログラムを解析して該ソースプログラムのプログラム呼出構造とデータ入出力とを関連づけた入出力付構造情報を生成する入出力付構造情報生成工程と、

前記入出力付構造情報生成工程により生成された入出力付構造情報から一部の情報を抽出して前記ソースプログラムの処理概要情報を生成する処理概要情報生成工程と、

前記処理概要情報生成工程により生成された処理概要情報を用いて前記ソースプログラムのプログラム仕様書を生成する文書生成工程と

を含んだことを特徴とする仕様書生成方法。

【産業上の利用可能性】

40

【0303】

以上のように、本発明にかかる仕様書生成プログラムおよびその記録媒体、仕様書生成装置ならびに仕様書生成方法は、ソフトウェアの保守に有用であり、特に、大規模なプログラムの処理の概要を把握したい場合に適している。

【図面の簡単な説明】

【0304】

【図 1】 本実施例 1 に係る仕様書生成装置の構成を示す機能ブロック図である。

【図 2】 構造抽出部が抽出するサブルーチン呼出の構造を説明するための説明図である。

【図 3】 サブルーチン情報記憶部が記憶するサブルーチン情報のデータ構造の一例を示す図である。

50

【図 4】 サブルーチン情報のリストのつながり（ツリー構造）を説明するための説明図である。

【図 5】 図 3 に示した呼出条件リストのデータ構造の一例を示す図である。

【図 6】 図 3 に示した入出力情報リストのデータ構造の一例を示す図である。

【図 7】 処理概要生成部が生成する処理概要情報を説明するための説明図である。

【図 8】 ドキュメント生成部が生成するプログラム仕様書の一例を示す図である。

【図 9】 構造抽出部によるサブルーチン情報生成処理および条件分岐情報抽出部による条件分岐情報抽出処理の処理手順を示すフローチャートである。

【図 10】 入出力データ抽出部による入出力情報抽出処理の処理手順を示すフローチャートである。

【図 11】 構造－入出力データ関連生成部による入出力情報のサブルーチン情報記憶部への追加処理の処理手順を示すフローチャートである。

【図 12】 処理概要生成部による処理概要生成処理の処理手順を示すフローチャートである。

【図 13】 本実施例 2 に係るコメント継承を説明するための説明図である。

【図 14】 本実施例 2 に係る仕様書生成装置の構成を示す機能ブロック図である。

【図 15】 コメント記述領域リストに対応するプロパティのデータ構造の一例を示す図である。

【図 16－1】 コメント記述領域に対応するプロパティのデータ構造の一例を示す図である。

【図 16－2】 複数の記述領域がある場合のコメント記述領域に対応するプロパティのデータ構造の一例を示す図である。

【図 17】 図 14 に示したコメント記入内容継承部の処理手順を示すフローチャートである。

【図 18】 スプレッドシートを用いたプログラム仕様書のコメント記述領域の一例を示す図である。

【図 19】 スプレッドシートを用いたプログラム仕様書のコメント記述領域に対応するプロパティのデータ構造の一例を示す図である。

【図 20】 スプレッドシートを用いたプログラム仕様書のコメント記述領域（複数の記述領域がある場合）に対応するプロパティのデータ構造の一例を示す図である。

【図 21】 処理構造図におけるコメント記述領域を示す図である。

【図 22】 図 21 に示した処理構造図における各コメント記述領域の位置情報データの一例を示す図である。

【図 23】 本実施例 3 に係る仕様書生成装置の構成を示す機能ブロック図である。

【図 24】 プログラム概要生成部の構成を示す機能ブロック図である。

【図 25－1】 プログラムソース例を示す図である。

【図 25－2】 図 25－1 に示したプログラムソース例から生成されるプログラム中間情報を示す図である。

【図 26】 データの重要度の分類の一例を示す図である。

【図 27】 プログラム中間情報の変数用タグに付加される重要度の一例を示す図である。

【図 28】 ステートメントの重要度の分類の一例を示す図である。

【図 29】 プログラム中間情報のステートメントのタグに付加される重要度の一例を示す図である。

【図 30】 まとめ管理部の初期動作を説明するための説明図である。

【図 31】 まとめ管理部とまとめ処理プログラムとのインタフェース（I / F）を示す図である。

【図 32】 まとめ処理プログラムの実行順序保持の一例を示す図である。

【図 33－1】 まとめ処理プログラムが行うまとめのルール（局所的）の例を示す図（その 1）である。

【図 33－2】 まとめ処理プログラムが行うまとめのルール（局所的）の例を示す図（そ

10

20

30

40

50

の 2) である。

【図 3 3 - 3】まとめ処理プログラムが行うまとめのルール (局所的) の例を示す図 (その 3) である。

【図 3 3 - 4】まとめ処理プログラムが行うまとめのルール (局所的) の例を示す図 (その 4) である。

【図 3 3 - 5】まとめ処理プログラムが行うまとめのルール (全体的) の例を示す図 (その 1) である。

【図 3 3 - 6】まとめ処理プログラムが行うまとめのルール (全体的) の例を示す図 (その 2) である。

【図 3 4】日本語変換部の処理を説明するための説明図である。

10

【図 3 5 - 1】その他の日本語テンプレートの例を示す図 (その 1) である。

【図 3 5 - 2】その他の日本語テンプレートの例を示す図 (その 2) である。

【図 3 6】プログラムサンプルを示す図である。

【図 3 7】まとめ管理部が利用するまとめ処理プログラムのリストを示す図である。

【図 3 8】まとめ処理プログラムの実行順序を示す図である。

【図 3 9】まとめ処理終了時のプログラム中間情報を示す図である。

【図 4 0】日本語変換部が生成した日本語情報を示す図である。

【図 4 1】ドキュメント生成部が生成するプログラム概要書の一例を示す図である。

【図 4 2】本実施例 4 に係る仕様書生成装置の構成を示す機能ブロック図である。

【図 4 3】ジョブステップ入出力データ抽出部による入出力判定処理の処理手順を示すフローチャートである。

20

【図 4 4】全体入出力情報抽出部によるジョブ全体としてのファイル入出力判定処理の処理手順を示すフローチャートである。

【図 4 5】ユーティリティ除外処理の処理手順を示すフローチャートである。

【図 4 6】バッチジョブ記述の一例を示す図である。

【図 4 7】図 4 6 に示したバッチジョブ記述からバッチジョブ記述言語構文解析部が生成するバッチジョブ解析情報を示す図である。

【図 4 8】図 4 6 に示したバッチジョブ記述から生成されるバッチジョブ全体の処理概要を示す図である。

【図 4 9】本実施例 5 に係る仕様書生成装置の構成を示す機能ブロック図である。

30

【図 5 0】画面定義体の一例を示す図である。

【図 5 1】画面遷移情報の一例を示す図である。

【図 5 2】画面遷移データの一例を示す図である。

【図 5 3 - 1】画面遷移図の一例を示す図である。

【図 5 3 - 2】画面遷移図の他の例を示す図である。

【図 5 4】本実施例 6 に係る仕様書生成装置の構成を示す機能ブロック図である。

【図 5 5】辞書情報合成部の処理手順を示すフローチャートである。

【図 5 6】処理構造生成部の処理手順を示すフローチャートである。

【図 5 7 - 1】処理構造図を生成する対象プログラム (呼出元) を示す図である。

【図 5 7 - 2】処理構造図を生成する対象プログラム (呼出先) を示す図である。

40

【図 5 8】図 5 7 - 1 および図 5 7 - 2 に示したプログラムから生成される処理構造図の一例を示す図である。

【図 5 9】本実施例 7 に係る仕様書生成装置の構成を示す機能ブロック図である。

【図 6 0】実行ログ情報解析部の処理手順を示すフローチャートである。

【図 6 1】実行情報合成部の処理手順を示すフローチャートである。

【図 6 2】図 5 7 - 1 および図 5 7 - 2 に示したプログラムから生成される処理構造図の一例を示す図である。

【図 6 3】本実施例 1 ~ 7 に係る仕様書生成プログラムを実行するコンピュータシステムを示す図である。

【図 6 4】図 6 3 に示した本体部の構成を示す機能ブロック図である。

50

【図 6 5】利用者記入欄を設けないコメント継承の一例を示す図である。

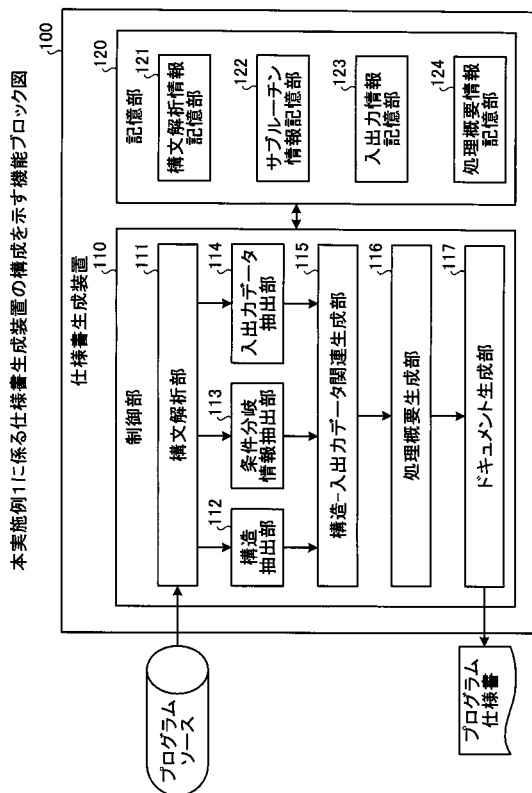
【符号の説明】

【0305】

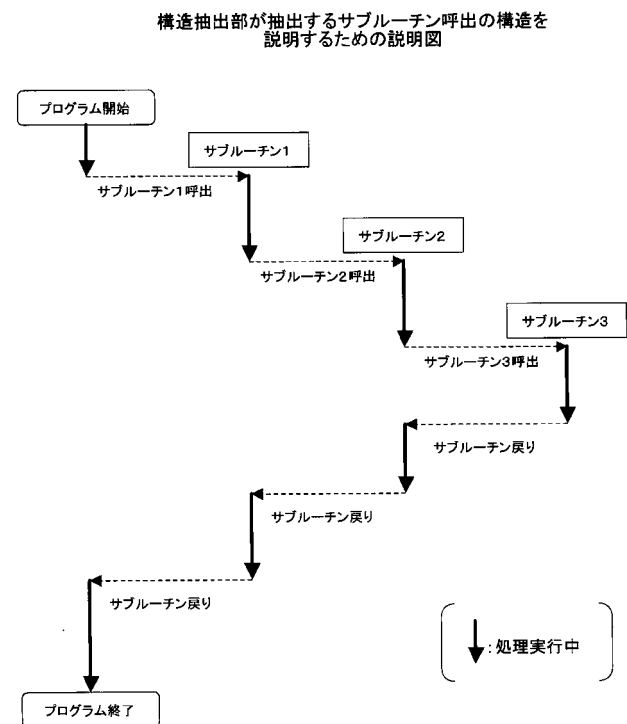
100, 200, 300, 400, 500, 600, 700	仕様書生成装置	
110, 210, 310, 410, 510, 610, 710	制御部	
111	構文解析部	
112	構造抽出部	
113	条件分岐情報抽出部	
114	入出力データ抽出部	
115	構造－入出力データ関連性生成部	10
116, 415	処理概要生成部	
117, 211, 312, 416, 513, 613, 713	ドキュメント生成部	
120, 320, 520, 620, 720	記憶部	
121	構文解析情報記憶部	
122, 621	サブルーチン情報記憶部	
123	入出力情報記憶部	
124	処理概要情報記憶部	
211a	コメント記入内容継承部	
311	プログラム概要生成部	
311a	データ重要度設定部	20
311b	ステートメント重要度設定部	
311c	まとめ管理部	
311d	日本語変換部	
321	まとめ処理記憶部	
322	日本語テンプレート記憶部	
323	プログラム概要情報記憶部	
411	バッチジョブ記述言語構文解析部	
412	ジョブステップ入出力データ抽出部	
413	全体入出力情報抽出部	
414	主要プログラム判定部	30
511	画面定義体解析部	
512	画面遷移データ生成部	
521	画面遷移情報記憶部	
522	レイアウト情報記憶部	
523	画面遷移データ記憶部	
611	辞書情報合成部	
612	処理構造生成部	
622, 721	処理構造情報記憶部	
711	実行ログ情報解析部	
712	実行情報合成部	40
800, 811	コンピュータシステム	
801	本体部	
802	ディスプレイ	
802a	表示画面	
803	キーボード	
804	マウス	
806	L A N	
807	公衆回線	
808	フロッピーディスク	
809	C D - R O M	50

- 8 1 2 サーバ
- 8 1 3 プリンタ
- 8 2 1 C P U
- 8 2 2 R A M
- 8 2 3 R O M
- 8 2 4 ハードディスクドライブ
- 8 2 5 C D－R O Mドライブ
- 8 2 6 フロッピーディスクドライブ
- 8 2 7 I／Oインタフェース
- 8 2 8 L A Nインタフェース
- 8 2 9 モデム

【図 1】

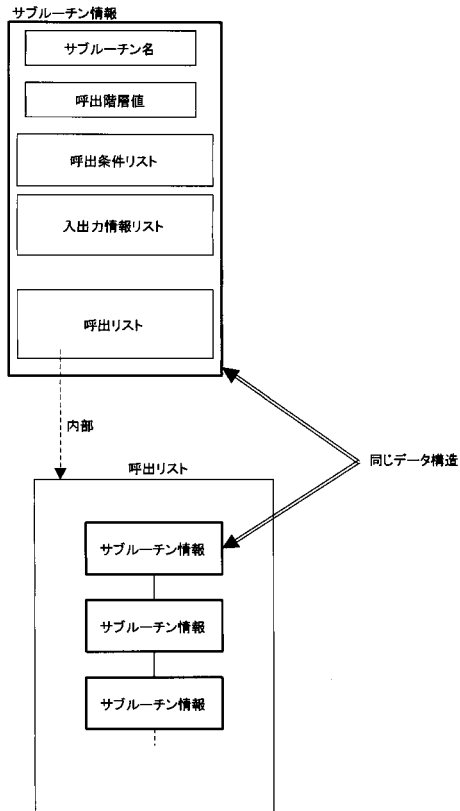


【図 2】



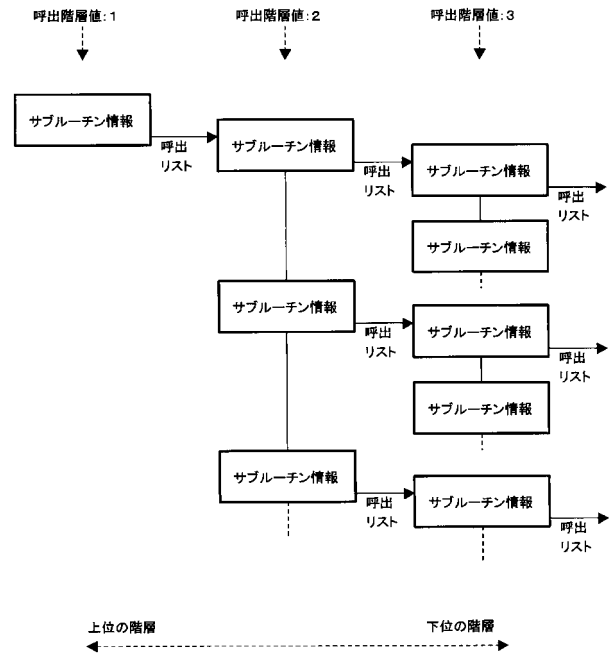
【図 3】

サブルーチン情報記憶部が記憶するサブルーチン情報のデータ構造の一例を示す図



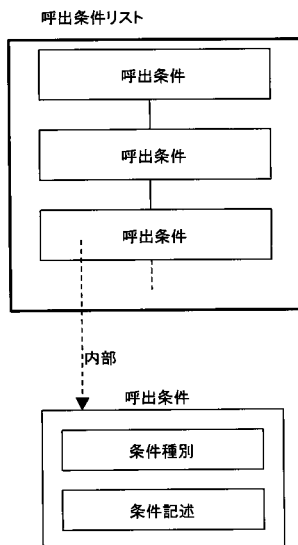
【図 4】

サブルーチン情報のリストのつながり(ツリー構造)を説明するための説明図



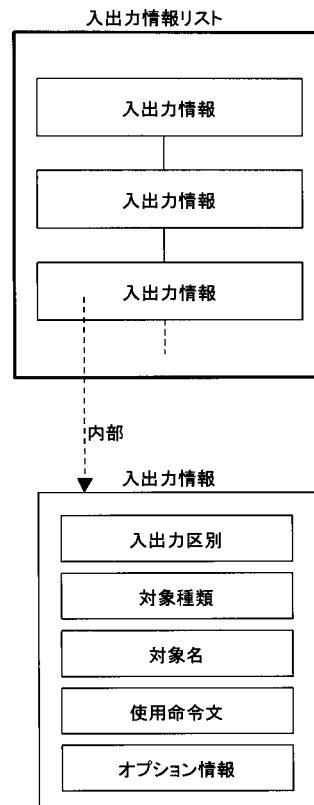
【図 5】

図3に示した呼出条件リストのデータ構造の一例を示す図



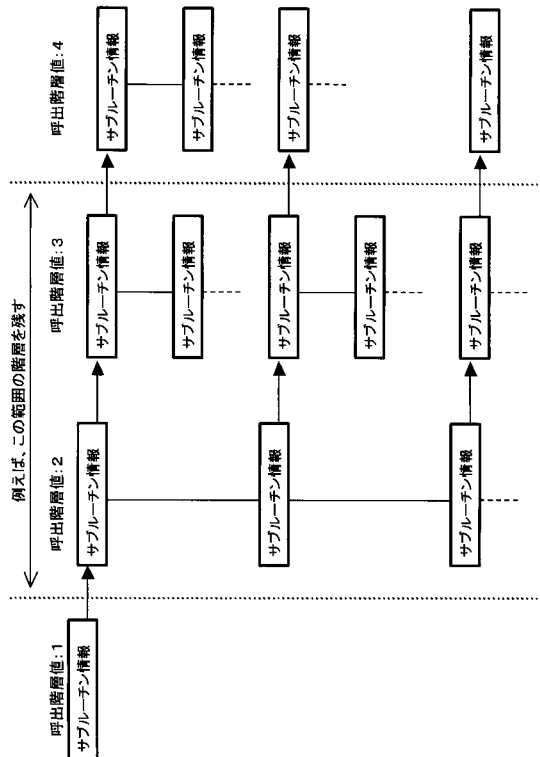
【図 6】

図3に示した入出力情報リストのデータ構造の一例を示す図



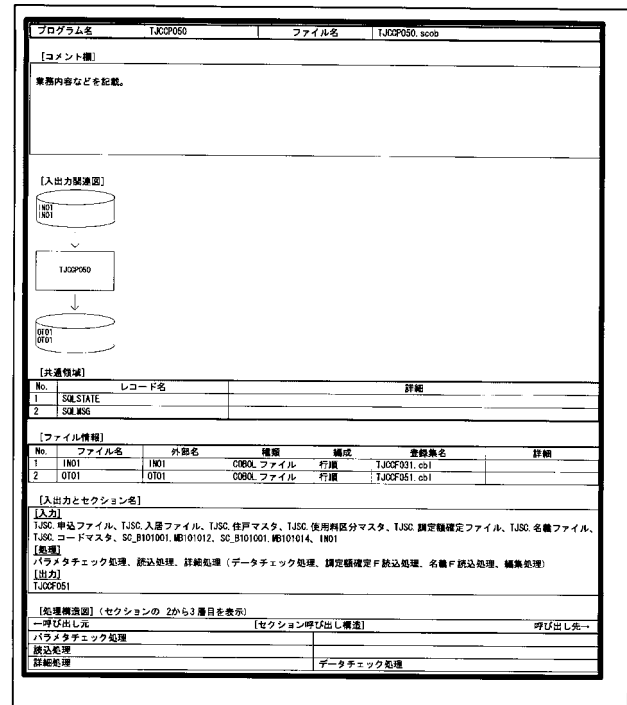
【図 7】

処理概要生成部が生成する処理概要情報を説明するための説明図



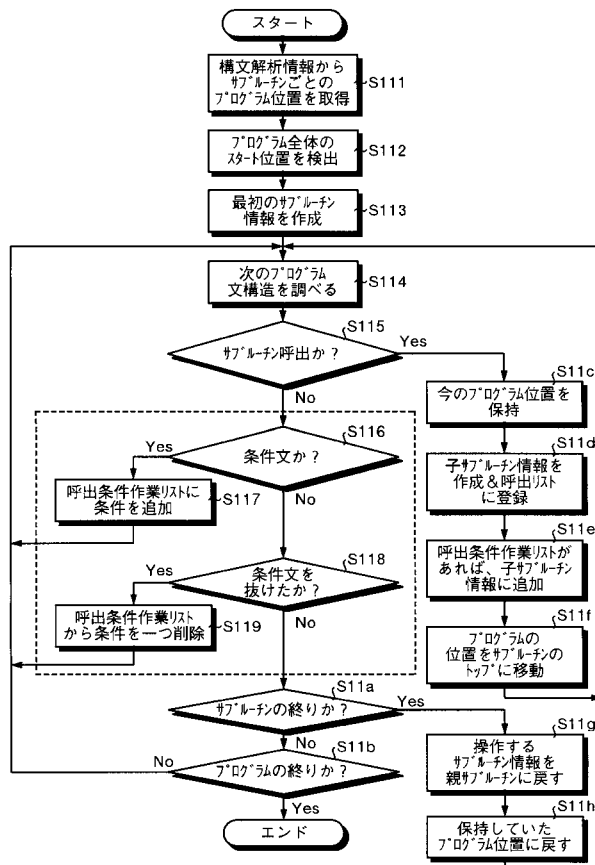
【図 8】

ドキュメント生成部が生成するプログラム仕様書の一例を示す図



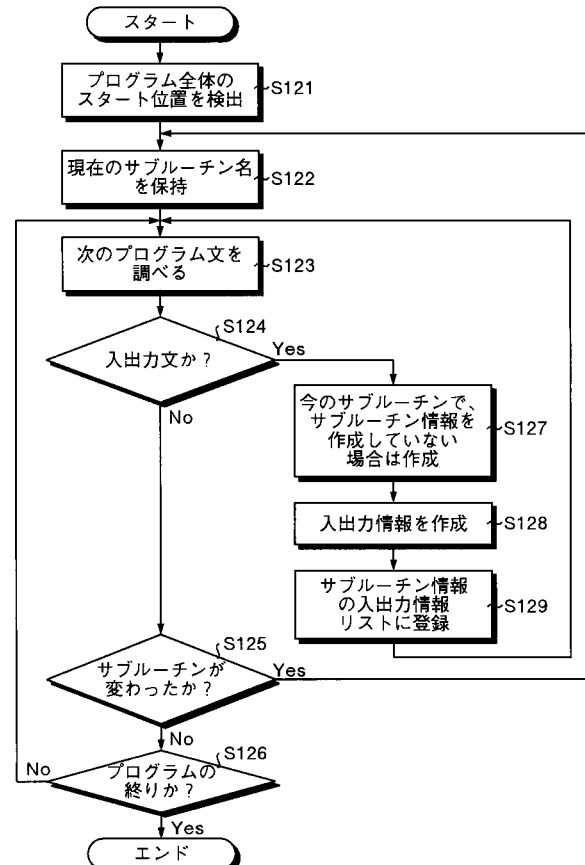
【図 9】

構造抽出部によるサブルーチン情報生成処理および条件分岐情報抽出部による条件分岐情報抽出処理の処理手順を示すフローチャート



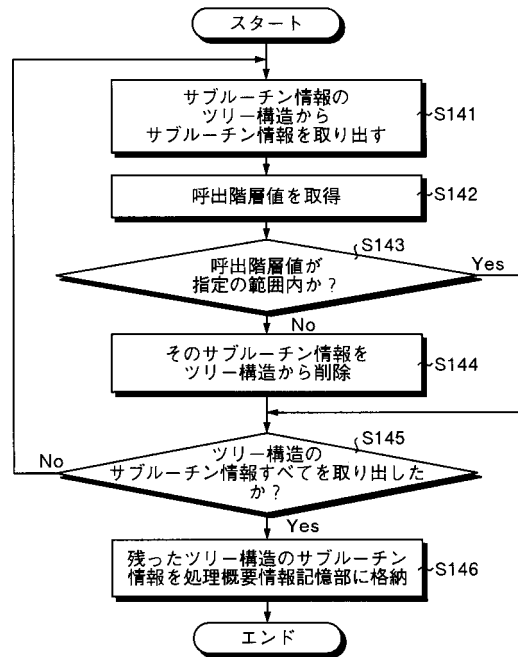
【図 10】

入出力データ抽出部による入出力情報抽出処理の処理手順を示すフローチャート

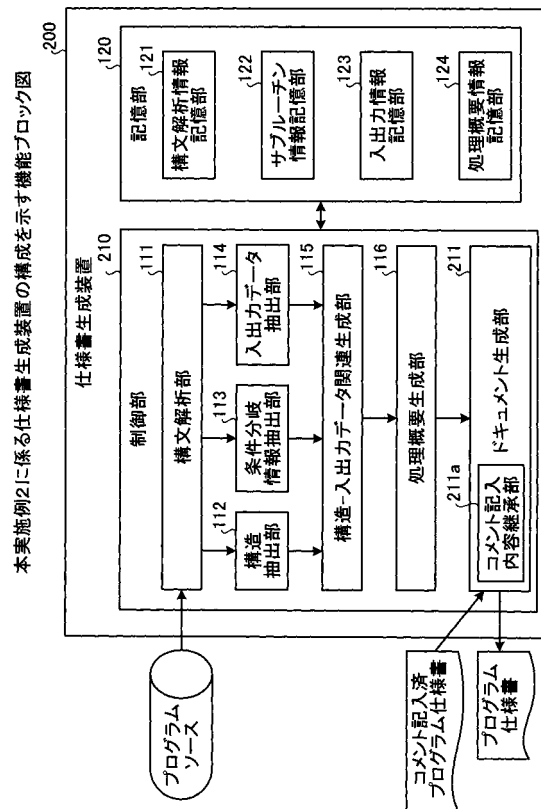


【図 1 2】

処理概要生成部による処理概要生成処理の処理手順を示すフローチャート



【 図 1 4 】



【図 15】

コメント記述領域リストに対応するプロパティのデータ構造の一例を示す図

ReflectionList (固定のプロパティ名)
Reflection1, Reflection2, ..., ReflectionN

【図 16-1】

コメント記述領域に対応するプロパティのデータ構造の一例を示す図

Reflection1
キーワード

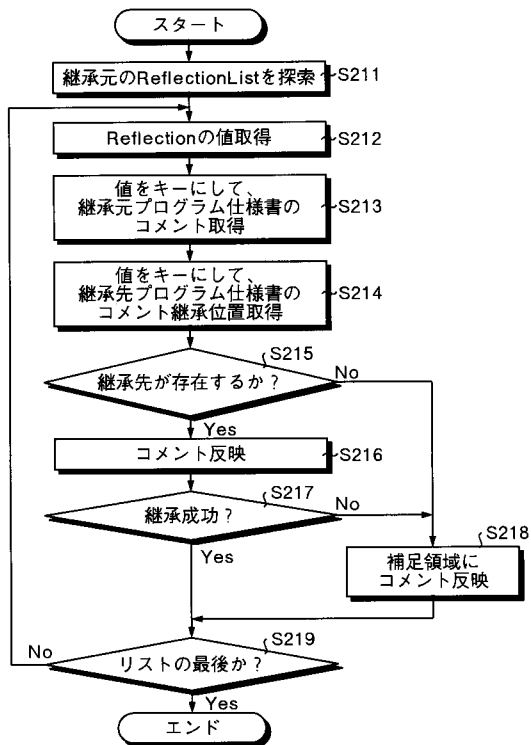
【図 16-2】

複数の記述領域がある場合のコメント記述領域に対応するプロパティのデータ構造の一例を示す図

MultiReflection1
キーワード, 複数コメント識別キーワード列位置, コメント記述領域列位置

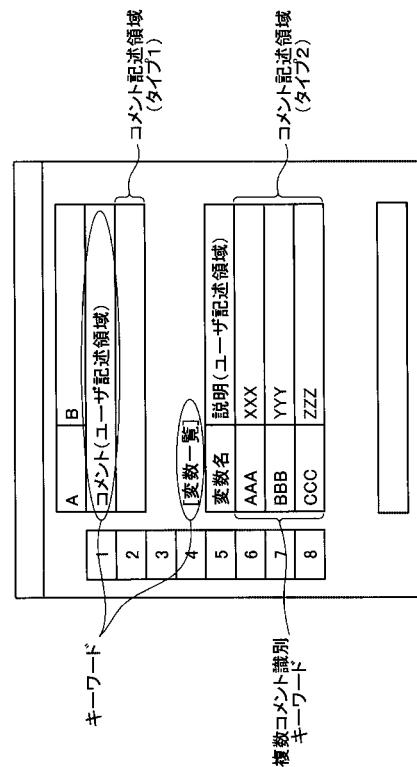
【図 17】

図 14 に示したコメント記入内容継承部の処理手順を示すフローチャート



【図 18】

スプレッドシートを用いたプログラム仕様書のコメント記述領域の一例を示す図



【図 19】

スプレッドシートを用いたプログラム仕様書のコメント記述領域に対応するプロパティのデータ構造の一例を示す図

Reflection1
キーワード、 コメント記述領域行位置、 コメント記述領域列位置

【図 20】

スプレッドシートを用いたプログラム仕様書のコメント記述領域（複数の記述領域がある場合）に対応するプロパティのデータ構造の一例を示す図

MultiReflection1
キーワード、 コメント記述領域行位置、 コメントの行数、 複数コメント識別キーワード列位置、 コメント記述領域列位置

【図 22】

図21に示した処理構造図における各コメント記述領域の位置情報データの一例を示す図

情報名→	セクション構造 (出現個数を意識した構造)	変位 (対応セクションから)	付加列名 (構造以外の欄の 場合)
コメント(1)	プログラム名	行2、列0	
コメント(2)	プログラム名	行0、列2	
コメント(3)	プログラム名 └─ 初期処理 [1]	行4、列0	
コメント(4)	プログラム名 └─ 繰り返し処理 [1] └─ CALL "COM002" [1]	行1、列0	
コメント(5)	プログラム名 └─ 繰り返し処理 [1]		読み取りファイル
コメント(6)	プログラム名 └─ 繰り返し処理 [2]		書き込みファイル
コメント(7)	プログラム名 └─ 繰り返し処理 [2] └─ <finish> [1]	行2、列0	
コメント(8)	プログラム名 └─ 終了処理 [1]		備考

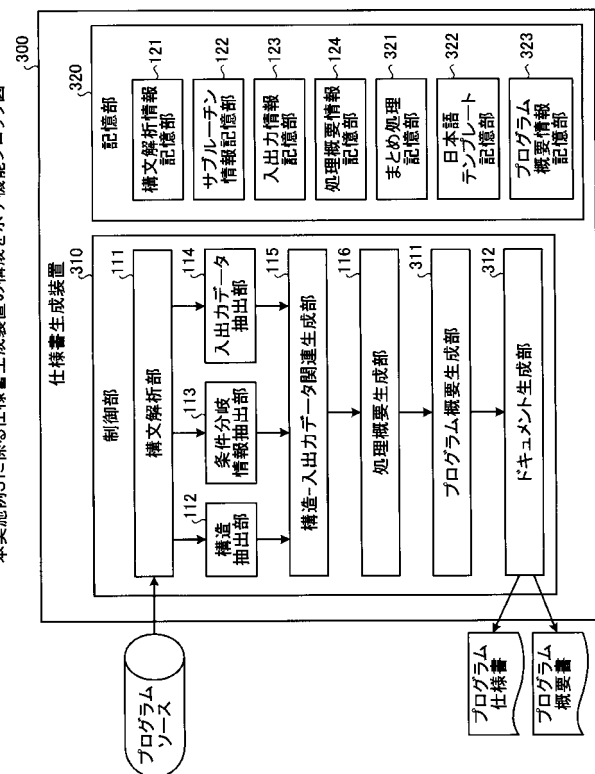
【図 21】

処理構造図におけるコメント記述領域を示す図

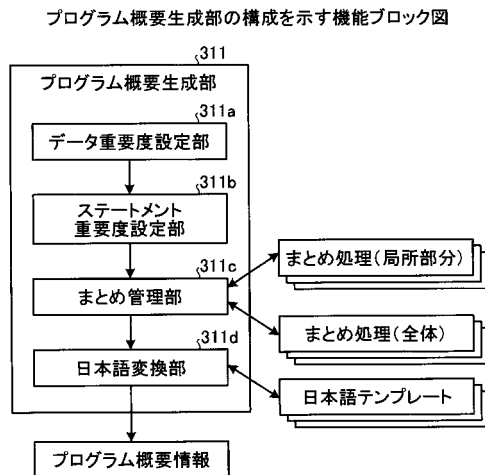
プログラム名	SECTION(1 層目)	SECTION(2 層目)	読み取り ファイル	書き込み ファイル	備考
プログラム					
コメント(1)	初期処理	コメント(2)			
	実行条件 UNTIL(終了フラグ=定数-終了)				
	コメント(3)				
	繰り返し処理		in1(rec1)	out1(rec1) out2(rec2) out3(rec3) out4(rec4) out5(rec5)	
		CALL "COM002"			
		コメント(4)			
	繰り返し処理			コメント(6)	
		GOTO <loop>			
		<finish>			
		コメント(7)			
	終了処理				コメント(8)

【図 23】

本実施例3に係る仕様書生成装置の構成を示す機能ブロック図



【図 2 4】



【図 2 5 - 1】

プログラムソース例(COBOL)を示す図

```

MAIN          SECTION.
MAIN-START.

  MOVE 0 TO W-ERR-FLAG.
  IF IN-CODE NOT = 103
    GO TO NEXT-DATA-READ
  END-IF.

  PERFORM MASTER-MODIFY-SECT.
  IF W-ERR-FLAG NOT = 0
    GO TO MAIN-END
  END-IF.

NEXT-DATA-READ.
  PERFORM FILE-READ-SECT.

MAIN-END.
  EXIT.
  
```

【図 2 5 - 2】

図25-1に示したプログラムソース例から生成されるプログラム中間情報の例を示す図

```

< section name="MAIN">
  < paragraph name="MAIN - START">
    < sequences num="4">
      < move>
        < ref>< constant value="0" type="int"/></ref>
        < def>< var name="W -ERR- FLAG"/></def>
      </move>
      < if>
        < condition>< expression>
          < var name="IN - CODE"/>
          < comparison_operator name="NOT ="/>
          < constant value="103" type="int"/>
        </expression></condition>
        < then>
          < sequences num="1">
            < goto>
              < target type="paragraph"
                name="NEXT - DATA - READ"
                section_name="MAIN"/>
            </goto>
          </sequences>
        </then>
      </if>
      < perform_external>
        < target type="section" name="MASTER - MODIFY - SECT"/>
      </perform_external>
      < if>
        < condition>< expression>
          < var name="W -ERR- FLAG"/>
          < comparison_operator name="NOT ="/>
          < figurative_constant name="ZERO"/>
        </expression></condition>
        < then>
          < sequences num="1">
            < goto>
              < target type="paragraph"
                name="MAIN - END"
                section_name="MAIN"/>
            </goto>
          </sequences>
        </then>
      </if>
    </sequences>
  </paragraph>
  < paragraph name="NEXT - DATA - READ">
    < sequences num="1">
      < perform_external>
        < target type="section" name="FILE - READ - SECT"/>
      </perform_external>
    </sequences>
  </paragraph>
  < paragraph name="MAIN - END">
    < sequences num="1">
      < exit_sentence/>
    </sequences>
  </paragraph>
</section>
  
```

【図 2 6】

データの重要度の分類の一例を示す図

重要度	分類ID	説明
1	D-1	処理バスの分岐条件に関わるデータ
2	D-2	ファイル/データベースなどの出力に使われるデータ
3	D-3	ファイル/データベースなどからの入力に使われるデータ
4	D-4	その他のデータ

重要度高 ← → 重要度低

【图 28】

変数DATA1の表現 ----> <var name="DATA1"/>

↓

変数DATA1に
重要度1を付加 ----> <var name="DATA1" data_priority="1"/>

ステートメントの重要度の分類の一例を示す図

重要度	分類ID	説明
1	S-1	重要度D-1D-2に該当するデータに列し、代人等の書き換えを行う文、サブルーチン呼出を行う文、ファイル、データベースに入出力を行う文
2	S-2	条件判定文、ただし、判定直後に実行される文としてS-1の分類の文を含むもの
3	S-3	その他の文

【 ㄨ 3 0 】

MOVE 0 TO DATA1.---->

```

<move>
  <ref>
    <constant value="0" type="int"/>
  </ref>
  <def>
    <var name="DATA1" data_priority="1"/>
  </def>
</move>

```

↓

代入先の変数 DATA1
に重要度1が付加され
ているので、文の重要
度1を付加

```

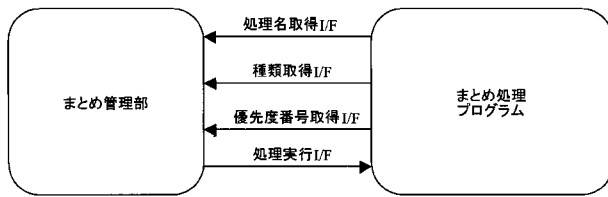
<move statement_priority="1">
  <ref>
    <constant value="0" type="int"/>
  </ref>
  <def>
    <var name="DATA1" data_priority="1"/>
  </def>
</move>

```

[illegible]

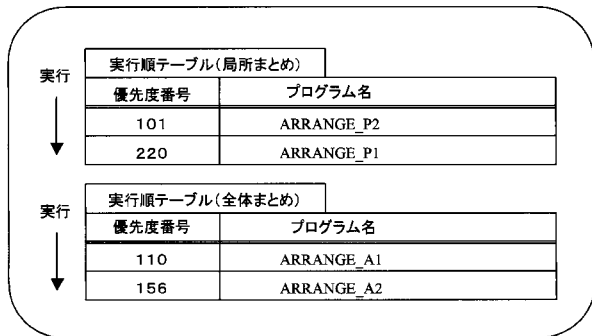
【図 3 1】

まとめ管理部とまとめ処理プログラムのインタフェース(I/F)を示す図



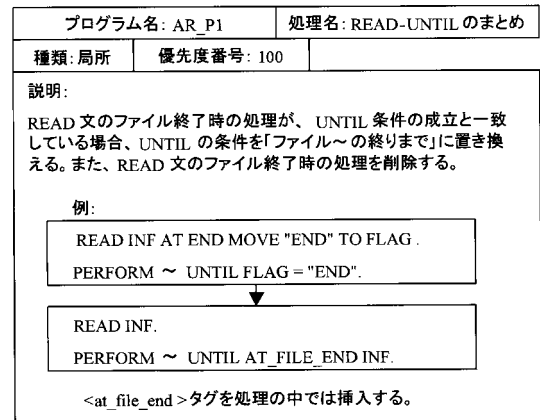
【図 3 2】

まとめ処理プログラムの実行順序保持の一例を示す図



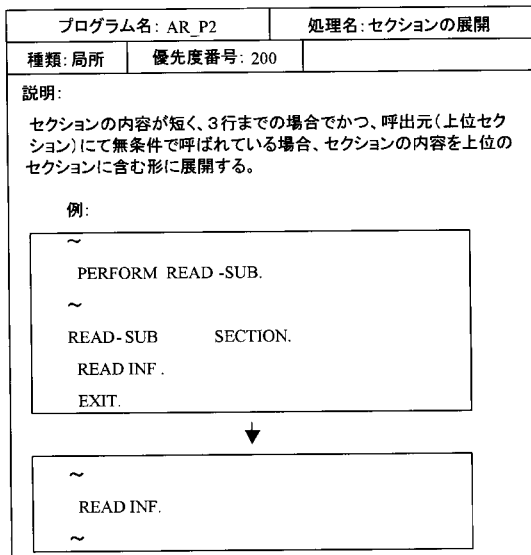
【図 3 3 - 1】

まとめ処理プログラムが行うまとめのルール(局所的)の例を示す図(その1)



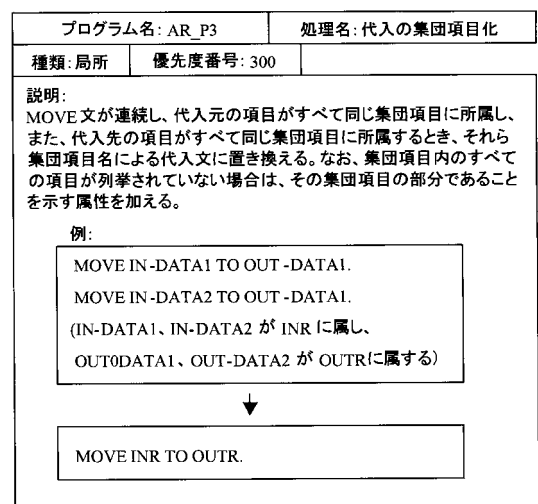
【図 3 3 - 2】

まとめ処理プログラムが行うまとめのルール(局所的)の例を示す図(その2)



【図 3 3 - 3】

まとめ処理プログラムが行うまとめのルール(局所的)の例を示す図(その3)



【図 3 3 - 4】

まとめ処理プログラムが行うまとめのルール(局所的)の例を示す図(その4)

プログラム名: AR_P4		処理名: 重要度の低い文の削除
種類: 局所	優先度番号: 400	
説明:		
(1) statement_priority が"3"になっている文を削除する。また、このとき、削除される文のタグの親タグ <sequence> の属性 qt の値を削除した文の数だけ減らす。(<qt>はその内部に含む文の数を示す) (2) また、このとき、<sequence> の属性qtの値が0の場合、それを含むセクション、パラグラフまたは、条件文(IF,EVALUATE, READ 後条件)を削除する。セクションの場合はその呼出元の呼出文(EVALUATE)も削除する。 なお、(1)、(2)は文の数が変わらなくなるまで対象となるプログラムについて繰り返し行う。		

【図 3 3 - 5】

まとめ処理プログラムが行うまとめのルール(全体的)の例を示す図(その1)

プログラム名: AR_A1		処理名: READのループ検出
種類: 全体	優先度番号: 100	
説明:		
ファイルを読む処理があり、その後にループがあり、ループの内部の処理の最後で、ファイルを読んでいる場合、「ループの条件成立まで、繰り返しファイルを読む」という形に置き換える。ループがセクションの場合はそれを呼び出している部分に展開する。		
例:		
処理1 READ INF. PERFORM SUB1 UNTIL 条件A. 処理2 SUB1 SECTION. 処理3 READ INF. EXIT.		
↓		
処理1 LOOP-READ INF UNTIL 条件A 処理3 END-LOOP-READ 処理2		
		新たな文を設定する。 処理の中では、 <loop_read_file>タグを新たに用意する。

【図 3 3 - 6】

まとめ処理プログラムが行うまとめのルール(全体的)の例を示す図(その2)

プログラム名: AR_A2		処理名: 変数の全体名記述
種類: 全体	優先度番号: 200	
説明:		
変数の項目名を集団項目名も付加した形で表現する。名前の間は"."で連結する。また、もっとも上位の集団項目名から表現する。 IN-RECORD.IN-DATA1 などの形式。		

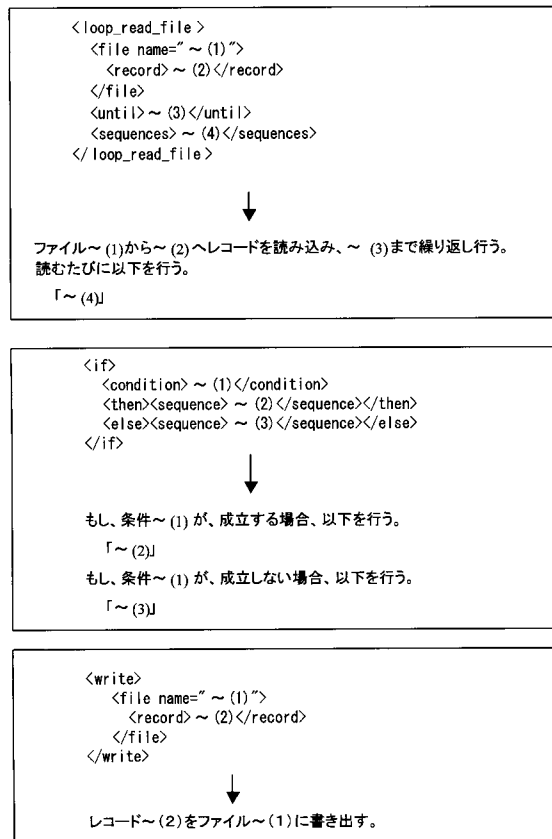
【図 3 4】

日本語変換部の処理を説明するための説明図

"MOVE 0 TO DATA1."	
<pre><move statement_priority="1"> <ref> <constant value="0" type="int"/> </ref> <def> <var name="DATA1" data_priority="1"/> </def> </move></pre>	
テンプレート	
<pre><move> <ref>~(1)</ref> <def>~(2)</def> </move></pre>	
↓	
「~(1)を~(2)に代入する。」	
<pre><constant value="~(3)" type="int"/></pre>	
↓	
「整数(~(3))」	
<pre><var name="~(4)"/></pre>	
↓	
「変数[~(4)]」	
生成する日本語文	
<pre>整数(0) を 変数[DATA1] に代入する。</pre>	

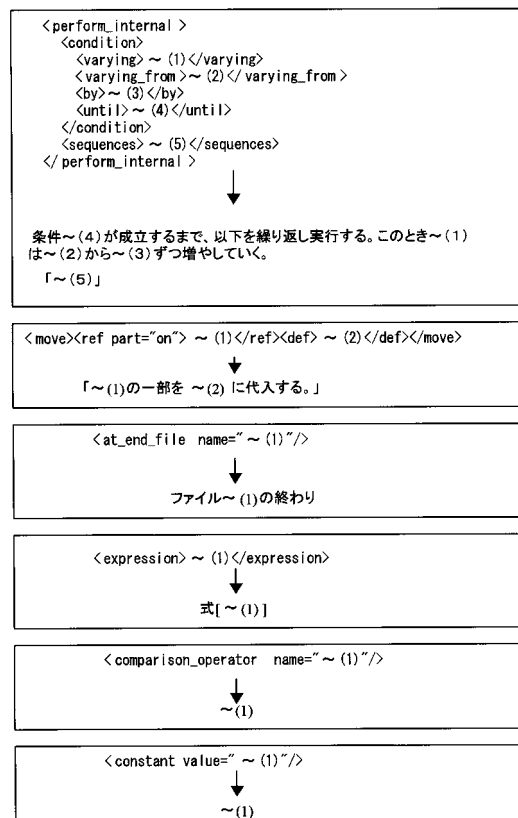
【図 35-1】

その他の日本語テンプレートの例を示す図(その1)



【図 35-2】

その他の日本語テンプレートの例を示す図(その2)



【図 36】

プログラムサンプルを示す図

IDENTIFICATION DIVISION PROGRAM-ID. TESTSAMPLE. AUTHOR. TARO.YAMADA. ENVIRONMENT DIVISION CONFIGURATION SECTION SOURCE-COMPUTER. VIRTUAL/HOST1. OBJECT-COMPUTER. VIRTUAL/HOST1.	PROCEDURE DIVISION OPEN INPUT INFILE OUTPUT OUTFILE. DISPLAY "## プログラムスタート" UPON CONSOLE. PERFORM READ-SECT. IF END-FLAG = "END" DISPLAY "## 一つもレコードを読み取れません。" UPON CONSOLE DISPLAY "## エラー終了しました。" UPON CONSOLE STOP RUN END-IF. PERFORM MAIN-SECT UNTIL END-FLAG = "END" CLOSE INFILE OUTFILE. DISPLAY "〇出力件数 = " OUT-COUNT "〇" UPON CONSOLE. DISPLAY "正常終了しました。" UPON CONSOLE. STOP RUN. ***** READ ROUTINE ***** READ-SECT SECTION. READ INFILE AT END MOVE "END" TO END-FLAG NOT AT END ADD 1 TO IN-COUNT END-READ. READ-SECT-END. EXIT. ***** MAIN ROUTINE ***** MAIN-SECT SECTION. IF FAMILYDATA = ZERO ADD 1 TO SKIP-COUNT ELSE ADD 1 TO PROC-COUNT PERFORM VARYING I FROM 1 BY 1 UNTIL I > 20 MOVE MEMBRCODE TO OUT-MEMBRCODE MOVE F-TYPE(1) TO OUT-F-TYPE MOVE F-MEMBER-F(1) TO OUT-F-MEM-FLAG MOVE F-CODE(1) TO OUT-F-CODE WRITE OUT-RECORD ADD 1 TO OUT-COUNT END-PERFORM END-IF. PERFORM READ-SECT. MAIN-SECT-END. EXIT.
***** IN OUT ***** INPUT-OUTPUT SECTION. FILE-CONTROL. SELECT INFILE ASSIGN TO INFILE. SELECT OUTFILE ASSIGN TO OUTFILE.	
***** DATA DIVISION ***** DATA DIVISION. FILE SECTION. ***** INPUT FILE ***** FD INFILE BLOCK 0 RECORDS 01 IN-RECORD 03 MEMBRCODE PIC 9(5). 03 JOB-CODE PIC 9(3). 03 NAME. 05 NAMEFAMILY PIC X(14). 05 NAMEFIRST PIC X(18). 03 SEIBETSU PIC 9. 03 YUBINBANGU PIC 9(3). 05 YUBINCODE PIC 9(4). 05 YUSMALLCODE PIC X(14). 03 TEL PIC X(14). 03 BIRTHDAY. 05 BIRTH-YEAR PIC 9(4). 05 BIRTH-MONTH PIC 9(2). 05 BIRTH-DAY PIC 9(2). 03 FAMILYDATA. 05 F-MEMBER OCCURS 20. 07 F-TYPE PIC 9(2). 07 F-MEMBER-F PIC 9. 07 F-CODE PIC 9(3). 03 FILLER PIC X(25).	
***** OUTPUT FILE ***** FD OUTFILE BLOCK 0 RECORDS 01 OUT-RECORD 03 OUT-MEMBRCODE PIC 9(5). 03 OUT-F-TYPE PIC 9(2). 03 OUT-F-MEM-FLAG PIC 9. 03 OUT-F-CODE PIC 9(3).	
***** WORKING STORAGE SECTION ***** WORKING-STORAGE SECTION. 01 COUNTER-TABLE. 03 IN-COUNT PIC 9(9) VALUE ZERO. 03 SKIP-COUNT PIC 9(9) VALUE ZERO. 03 PROC-COUNT PIC 9(9) VALUE ZERO. 03 OUT-COUNT PIC 9(9) VALUE ZERO. 01 I PIC 9(4). 01 END-FLAG PIC X(40) VALUE SPACE.	

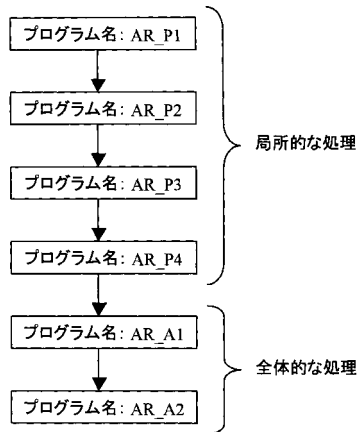
【図 37】

まとめ管理部が利用するまとめ処理プログラムのリストを示す図

プログラム名: AR_P1	処理名: READ-UNTIL のまとめ	種類: 局所	優先度番号: 100
プログラム名: AR_P2	処理名: セクションの展開	種類: 局所	優先度番号: 200
プログラム名: AR_P3	処理名: 代入の集約項目化	種類: 局所	優先度番号: 300
プログラム名: AR_P4	処理名: 重要度の低い文の削除	種類: 局所	優先度番号: 400
プログラム名: AR_A1	処理名: READ のループ検出	種類: 全体	優先度番号: 100
プログラム名: AR_A2	処理名: 変数の全体名記述	種類: 全体	優先度番号: 200

【図 38】

まとめ処理プログラムの実行順序を示す図



【図 39】

まとめ処理終了時のプログラム中間情報を示す図

```

< loop_read_file statement_priority="1">
  < file name="INFILE">
    < record>< var name="IN-RECORD" data_priority="3"/></record>
  </file>
  < until>< at_end_file name="INFILE"/></until>
  < sequences qt="2">
    < if statement_priority="2">
      < condition>
        < expression>
          < var name="IN-RECORD.FAMILYDATA" data_priority="1"/>
          < comparison_operator name="=" />
          < constant value="ZERO"/>
        </expression>
      </condition>
    </if>
    < else>
      < sequences qt="2">
        < perform_internal statement_priority="1">
          < condition>
            < varying>< var name="I" data_priority="1"/></varying>
            < varying_from>< constant value="1" type="int"/></varying_from>
            < by>< constant value="1" type="int"/></by>
          </condition>
          < expression>
            < var name="I" data_priority="1"/>
            < comparison_operator name=">" />
            < constant value="20" type="int"/>
          </expression>
        </until>
      </condition>
    </sequences>
    < move statement_priority="1">
      < ref part="on">< var name="IN-RECORD" data_priority="3"/></ref>
      < def>< var name="OUT-RECORD" data_priority="2"/></def>
    </move>
    < write statement_priority="1">
      < file name="OUTFILE">
        < record>< var name="OUT-RECORD" data_priority="2"/></record>
      </file>
    </write>
  </sequences>
</perform_internal>
</sequences>
</if>
</sequences>
</loop_read_file>
  
```

【図 40】

日本語変換部が生成した日本語情報を示す図

ファイルINFILEから変数IN-RECORDへレコードを読み込み、ファイル INFILEの終わりまで繰り返して行う。ファイルを読むたびに以下を行う。

「もし、条件式[変数IN-RECORD.FAMILYDATA=ZERO]」が、成立しない場合、以下を行う。

「条件式[変数I>20]」が成立するまで、以下を繰り返し実行する。このとき変数 Iは1から1ずつ増やしていく。

「変数 IN-RECORDの一部を変数 OUT-RECORDに代入する。

レコード変数 OUT-RECORDをファイルOUTFILEに書き出す。

」

」

」

【図 41】

ドキュメント生成部が生成するプログラム
概要書の一例を示す図

プログラム概要書		作成日時	
		2003/07/11 13:45	

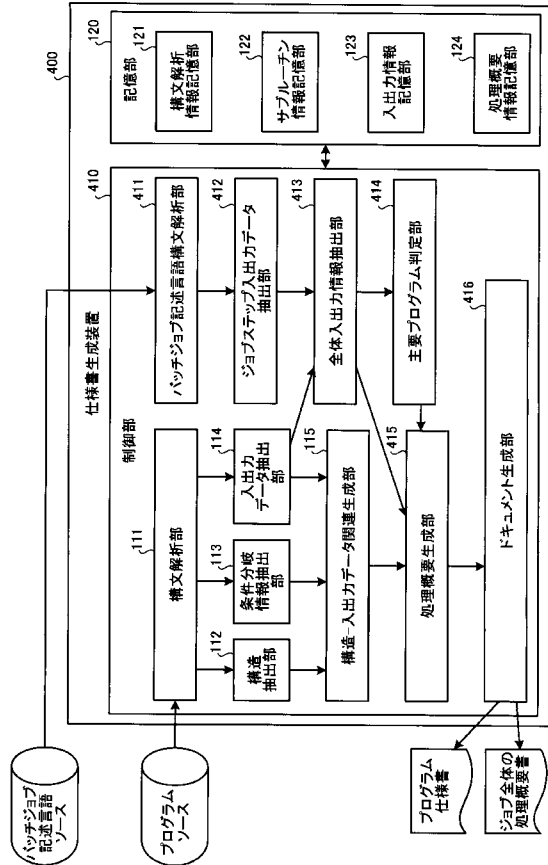
プログラム名	TESTSAMPLE	ファイル名	TESTSAMPLE.COB
コメント欄			

ファイル情報			
No.	ファイル名	外部名	種類
1.	INFILE	INFILE	COBOL ファイル
2.	OUTFILE	OUTFILE	COBOL ファイル

プログラム概要	
ファイルINFILEから変数IN-RECORDへレコードを読み込み、ファイル INFILEの終わりまで繰り返して行う。ファイルを読むたびに以下を行う。 「もし、条件式[変数IN-RECORD.FAMILYDATA=ZERO]」が、成立しない場合、以下を行う。 「条件式[変数I>20]」が成立するまで、以下を繰り返し実行する。このとき変数 Iは1から1ずつ増やしていく。 「変数 IN-RECORDの一部を変数 OUT-RECORDに代入する。 レコード変数 OUT-RECORDをファイルOUTFILEに書き出す。 」 」 」	

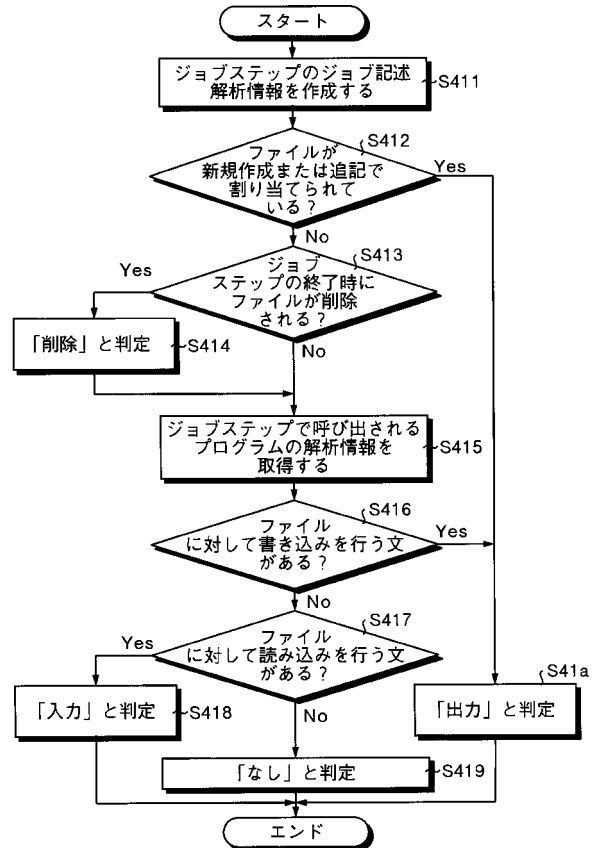
【 図 4 3 】

本実施例4に係る仕様書生成装置の構成を示す機能ブロック図



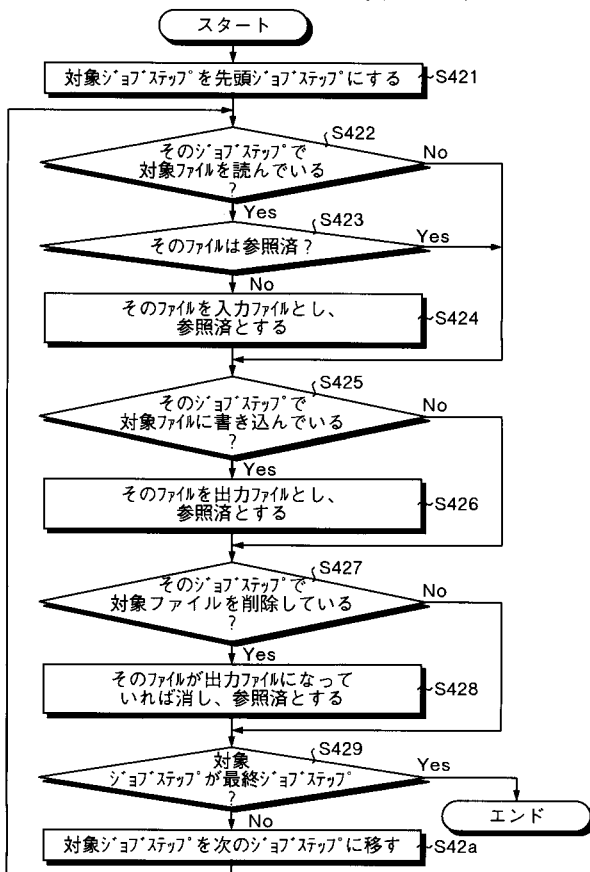
【 図 4 3 】

ジョブステップ入出力データ抽出部による入出力判定処理の
処理手順を示すフローチャート



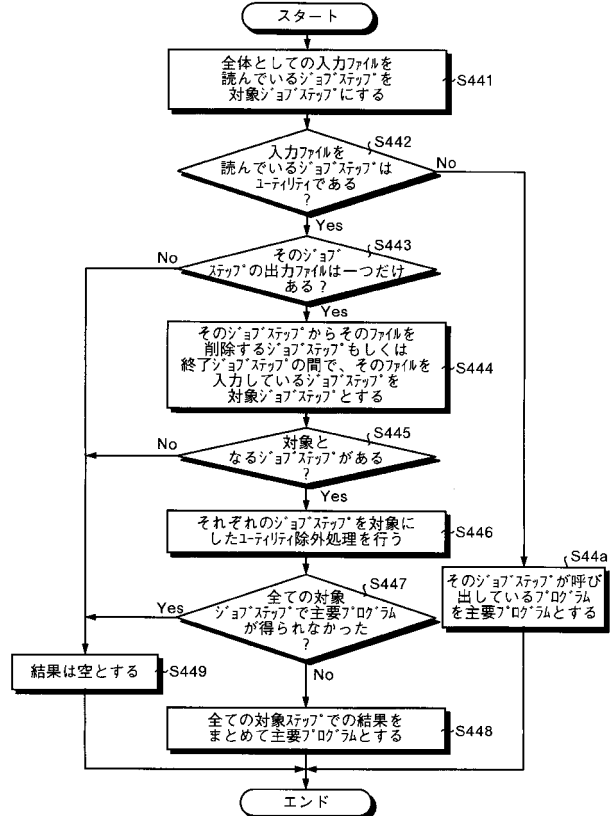
【図 4 4】

全体入出力情報抽出部によるジョブ全体としての
ファイル入出力判定処理の処理手順を示すフローチャート



【图 4 5】

ユーティリティ除外処理の処理手順を示すフローチャート



【図 4 6】

パッチジョブ記述の一例を示す図

```

///JOB01 JOB
//*****
//STEP1 EXEC PGM=PROGA
//IN01 DD DSN=DENPYO01,DISP=SHR
//OT01 DD DSN=WORK1,DISP=(NEW,PASS)
//*****
//STEP2 EXEC PGM=SORT
//SORTIN DD DSN=WORK1,DISP=(OLD,DELETE)
//SORTOT DD DSN=WORK2,DISP=(NEW,PASS)
//*****
//STEP3 EXEC PGM=PROGB
//IN01 DD DSN=WORK2,DISP=(OLD,DELETE)
//OT01 DD DSN=SYUKEI1,DISP(NEW,PASS)
//*****
//STEP4 EXEC PGM=ENDMSG
  
```

【図 4 7】

図46に示したパッチジョブ記述からパッチジョブ記述言語構造解析部が生成するパッチジョブ解析情報

ステップ名	呼び出しプログラム名	ファイル			
		DEPYO1	WORK1	WORK2	DATA1
STEP1	PROGA	外部名: IN01 モード: S	外部名: OT01 モード: NP	外部名: NP モード: SOTRIN	外部名: OT01 モード: NP
STEP2	SORT				
STEP3	PROGB				
STEP4	ENDMSG				

【図 4 8】

図46に示したパッチジョブ記述から生成されるパッチジョブ全体の処理概要を示す図

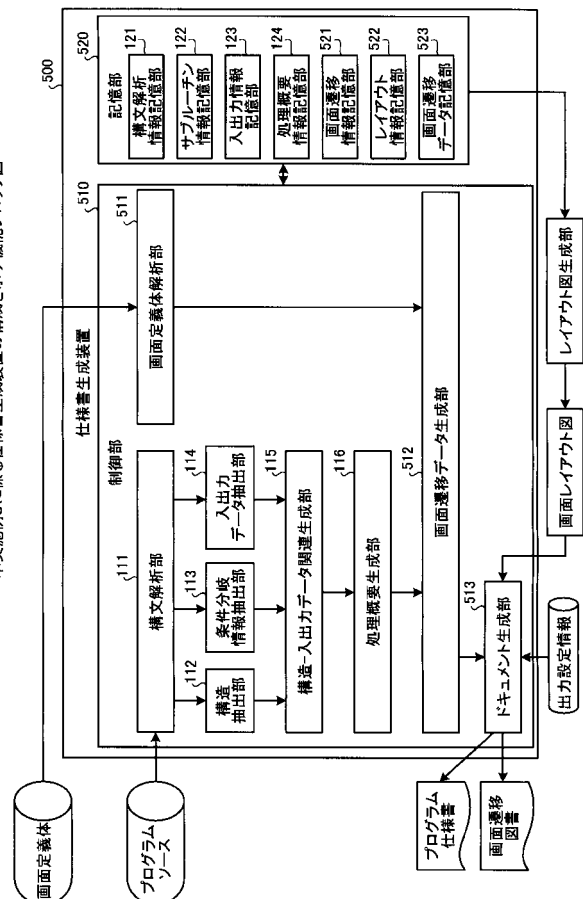
[入力]
DENPYO1

[出力]
SYUKEI1

[主要プログラム]
PROGA(新規伝票抽出処理),PROGB(当日集計処理)

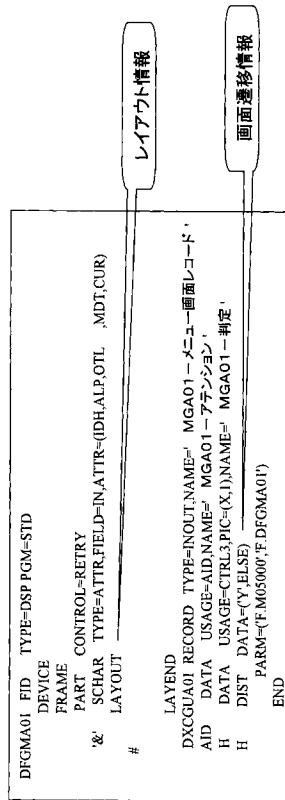
【図 4 9】

本実施例に係る仕様書生成装置の構成を示す機能ブロック図



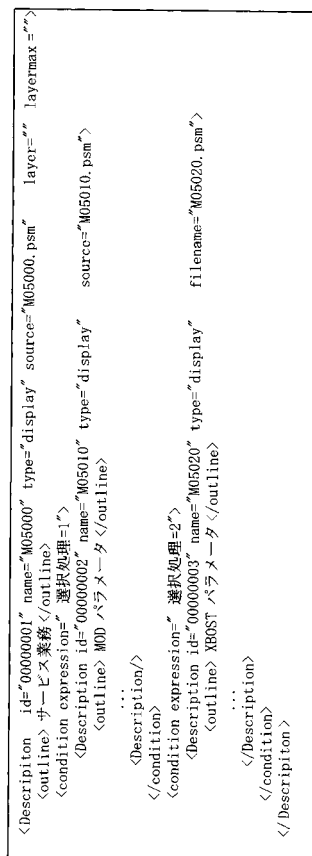
【図 50】

画面定義体の一例を示す図



【図 52】

画面遷移データの一例を示す図



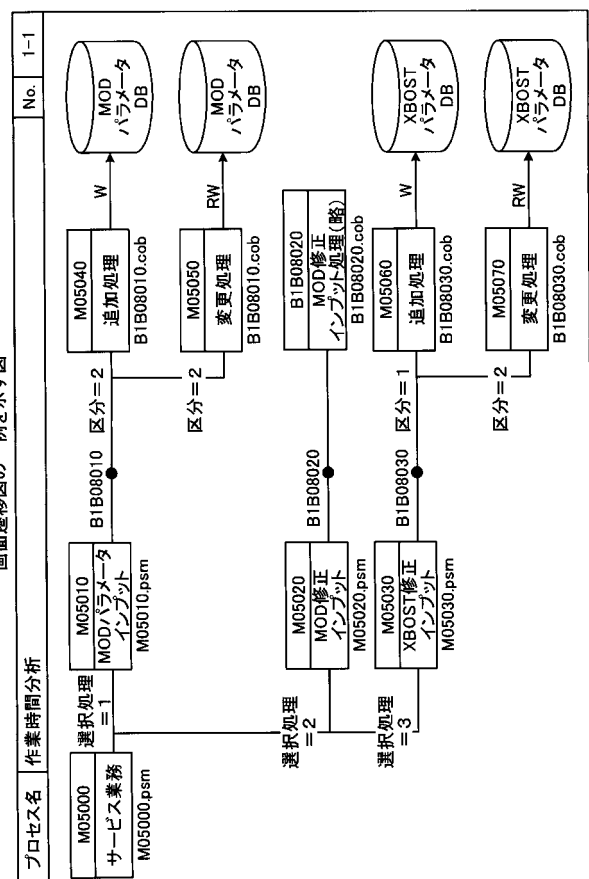
【図 51】

画面遷移情報の一例を示す図

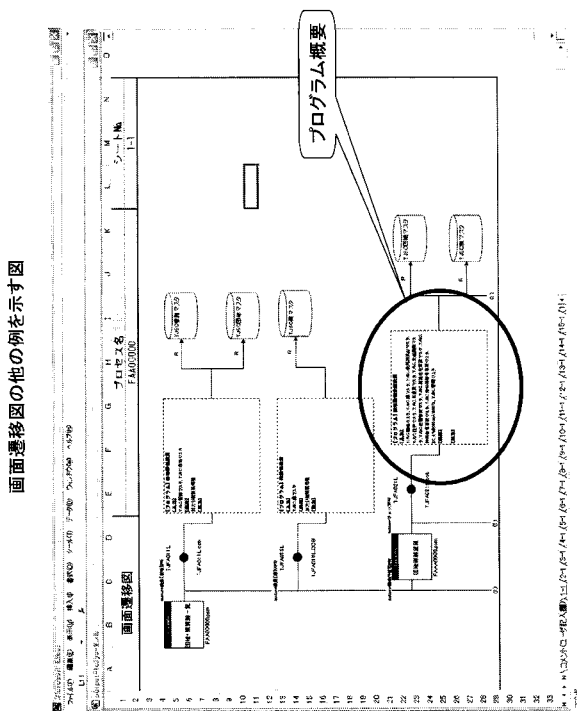
遷移元画面	遷移条件	遷移先画面	遷移先種別
M05000	選択処理 =1	M05010	画面
M05000	選択処理 =2	PG0001	プログラム

【図 53-1】

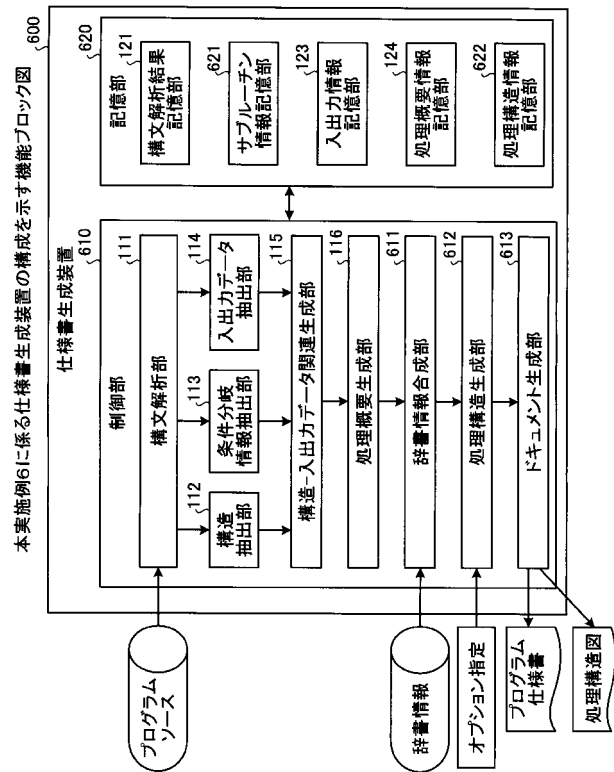
画面遷移図の一例を示す図



【図 5 3 - 2】

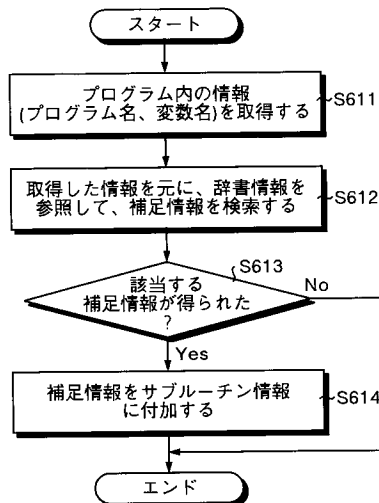


【図 5 4】



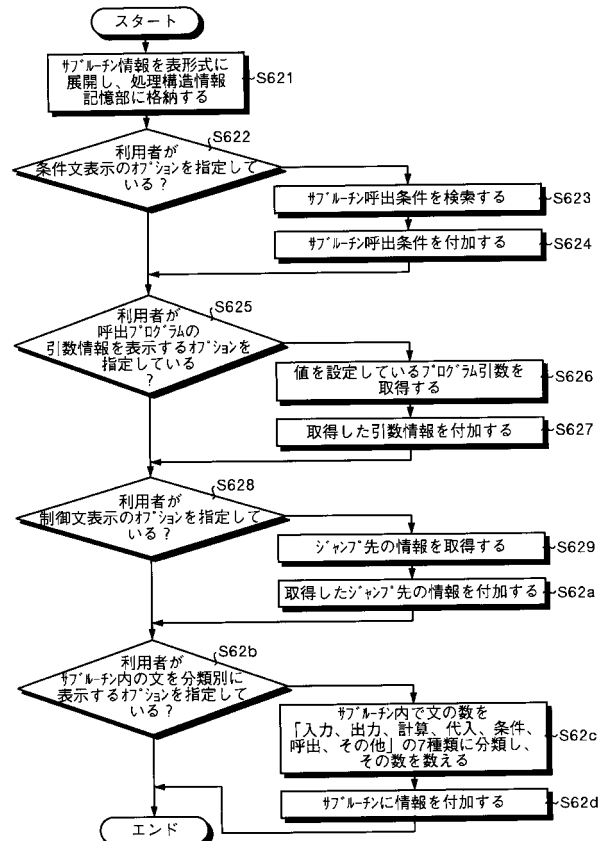
【図 5 5】

辞書情報合成部の処理手順を示すフローチャート



【図 5 6】

処理構造生成部の処理手順を示すフローチャート



【図 57-1】

処理構造図を生成する対象プログラム(呼出元)を示す図

```

PROGRAM-ID PROGA
***** PROCEDURE DIVISION *****
PROCEDURE DIVISION.

OPEN INPUT INFILE.
DISPLAY "## プログラムスタート " UPON CONSOLE.

PERFORM READ-SECT.
IF END-FLAG = "END"
    DISPLAY "## 一つもレコードを読めません。 " UPON CONSOLE.
    DISPLAY "## エラー終了しました。 " UPON CONSOLE.
    STOP RUN.
END-IF.

PERFORM MAIN-SECT UNTIL END-FLAG = "END"

CLOSE INFILE.
DISPLAY "○出力件数      =[ " OUT-COUNT "]" UPON CONSOLE.
DISPLAY "正常終了しました。 " UPON CONSOLE.

STOP RUN.

***** READ ROUTINE *****
READ-SECT SECTION.
READ INFILE
    AT END MOVE "END" TO END-FLAG
    NOT AT END ADD 1 TO IN-COUNT
END-READ.
READ-SECT-END.
EXIT.

***** MAIN ROUTINE *****
MAIN-SECT SECTION.
IF FAMILYDATA = ZERO
    ADD 1 TO SKIP-COUNT
ELSE
    ADD 1 TO PROC-COUNT
    PERFORM VARYING I FROM 1 BY 1
        UNTIL I > 20 OR F-MEMBER(I) = ZERO
        MOVE MEMBERCODE TO PARA1-MEMBERCODE
        CALL PROGRAM USING PARA1
        ADD 1 TO OUT-COUNT
    END-PERFORM
END-IF.

PERFORM READ-SECT.
MAIN-SECT-END.
EXIT.

```

【図 57-2】

処理構造図を生成する対象プログラム(呼出先)を示す図

```

PROGRAM-ID PROGW
***** PROCEDURE DIVISION *****
PROCEDURE DIVISION USING PARA1.

OPEN OUTPUT OUTFILE.
MOVE PARA1 TO OUT-RECORD.
WRITE OUT-RECORD
CLOSE OUTFILE.

STOP RUN.

***** READ ROUTINE *****

```

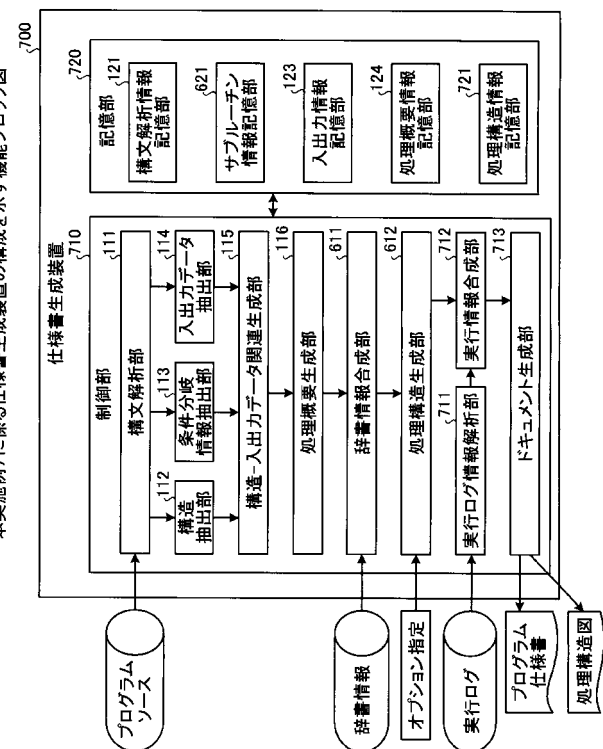
【図 58】

図57-1および図57-2に示したプログラムから生成される処理構造図の一例を示す図

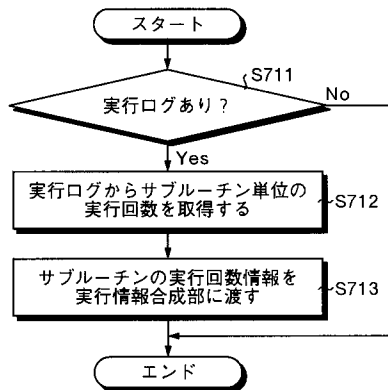
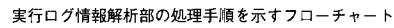
文書名	処理構造図
プログラム名 PROGA	SECTION(1(層目)) first section (no name)
	SECTION(2(層目)) READ-SECT
	SECTION(3(層目)) CALL "PROGW"
	読み取りファイル IN(INFILE)
	書き込みファイル OUT(OUTFILE)

【図 59】

本実施例7に係る仕様書生成装置の構成を示す機能ブロック図

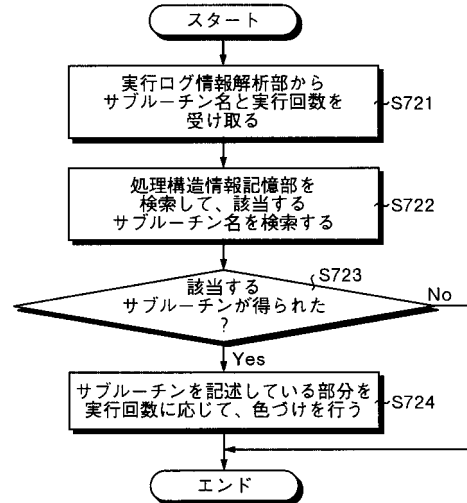


【図 6 0】



【図 6 1】

実行情報合成部の処理手順を示すフローチャート



【图 6 2】

図57-1および図57-2に示したプログラムから生成される処理構造図の一例を示す図

文書名	処理構造図	SECTION (1 階目)	SECTION (2 階目)	SECTION (3 階目)	読み取りファイル	書き込みファイル
		first section (no header)				
		分節別スタートメント数	READ-SECT		INF(INFILE)	
		入力1出力5時出た2条件1 その他の計9回	分節別スタートメント数 [入力1計算1代入1]	<READ-SECT-END>		
			MAIN-SECT 実行条件	"PROG"		OTF(OUTFILE)
			UNTIL(END-FLAG = "END")			
		分節別スタートメント数		USING [PARAL-MEMBERCODE-MEMBERCODE]		
		計算時出た3代入1条件1 計算時出た3代入1条件1	分節別スタートメント数 [計算時出た3代入1条件1]	<MAIN-SECT-END>		

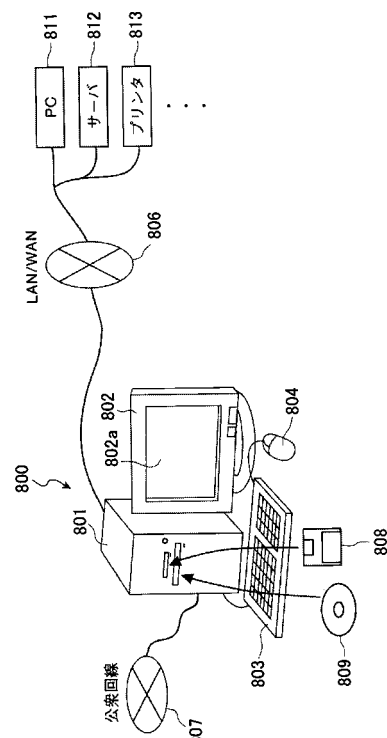
実行情報

1 回

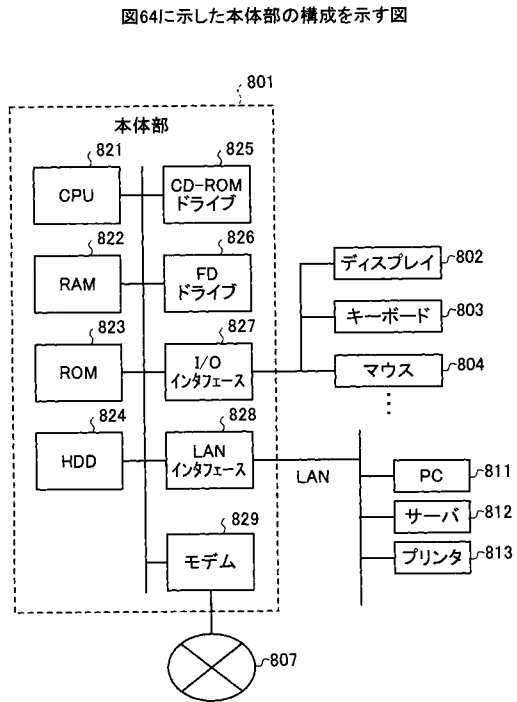
2 回

【図 6 3】

本実施例1～7に係る仕様書生成プログラムを実行する
コンピュータシステムを示す図

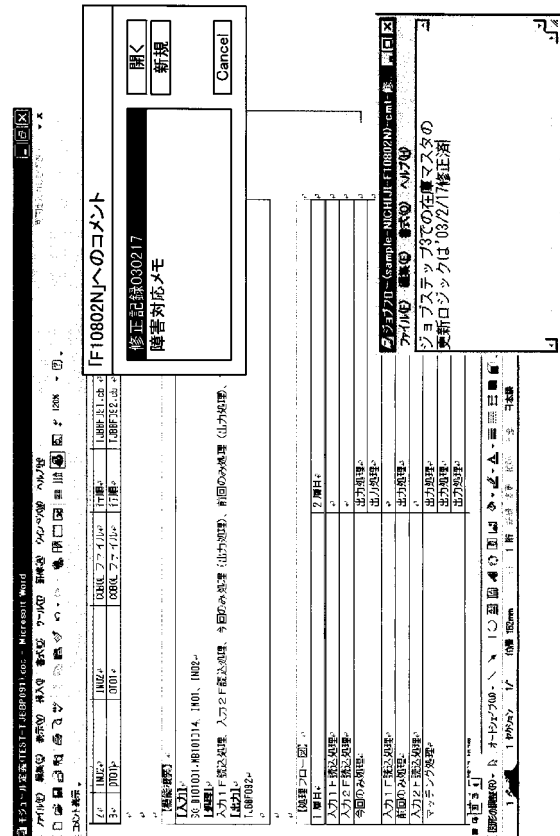


【图 6-4】



【図 6 5】

利用者記入欄を設けないコメント継承の一例を示す図



フロントページの続き

(72)発明者 上村 学

神奈川県川崎市中原区上小田中4丁目1番1号 富士通株式会社内

Fターム(参考) 5B076 DD02 DF01