情 報 シ ス テ ム 39－3
（1992．5．19）

保守ＣＡＳＥの条件：
ＲＥＳＣＵＥをベースに

本村昭二

ケーステクノロジー株式会社

ＲＥＳＣＵＥは既存情報システムのソフトウェア保守作業を支援するＣＡＳＥとして開発した。
既存のソフトウェアを解析、管理対象となる要素を抽出し、台帳、索引としてリポジトリィ上に格納する。データ項目は同一内容をグループ化しておく。
保守作業上で影響分析を中心とした情報は、検索することにより、ソフトウェアと一致した正確な内容が提供される。
これにより個人の知識に依存しない保守作業が可能となる。

Requirements for software maintenance CASE
:Base on RESCUE

Shoji Honmura

Case Technology Co., Ltd.

We have developed CASE named RESCUE which support software maintenance work for existing information systems
This CASE derives components from existing information systems, then stores it on the repository.
On demanded information about neutral relation between software components will be delivered.
The information is exactly same as existing software.
This makes it possible to maintain software not depending on the individual knowledge.

1　はじめに

現在稼働している既存アプリケーションシステムのソフトウェア保守作業を対象にした技術や管理に関する方法論、あるいはＣＡＳＥツールは極めて限られていて、開発に関するそれらとは比較にならない量である。

これから始まる新規開発のための環境をどのように改善、改革しても、開発が終了して現在運用されている大量なアプリケーションソフトウェア保守は、その恩恵をこうむることはない。

保守作業は、新規開発とは異なる環境、条件下にある。従ってその生産性向上策は異なる観点から考えるべきである。この保守作業の生産性向上こそ、情報システム部門が目指すべき緊急な課題である。

本稿では、ソフトウェア保守用ＣＡＳＥ「ＲＥＳＣＵＥ」の開発における考え方と機能を紹介する。

2　保守作業の問題点と解決方法

2．1　影響分析の困難さ

現在運用されている情報システムの保守の対象となるソフトウェアは、保守作業をほとんど考慮しないで開発された。適用業務の機能とソフトウェアの保守部分が対応していることはない。抽象的な表現をすれば適用業務の機能構造とそれをコンピュータシステム化したソフトウェアの構造は対応していない。従って機能の変更や追加が起った場合には、その機能に対応して、ソフトウェアのどの部分を修正、追加すればよいかを特定することがむつかしい。

適用業務の上では変化はないが、情報システムの仕様として単純に発生するデータ項目の増加というだけで、ソフトウェアは大きな影響を受ける。１つの例として消費税がある。

売上額の３％を消費税とするだけの仕様が、ソフトウェアに多大な変更を発生させた。このような単純な修正もソフトウェアの世界では大事件であり、場合によっては全体的な作り替えに匹敵する作業量となる。

この項目の追加によって影響する対象は、図１の例のようにデータベース、ファイル、画面、帳票といったデータの定義やレイアウトなど、その項目が付け加わるものすべてにわたる。その上にプログラムは消費税を計算しているものは当然のことながら、消費税とは直接何の関係もないプログラムまでも、消費税の項目追加に伴ってレコードのレイアウトが変更される結果、定義の修正なり、コンパイル作業なりが発生する。

このような広い範囲の変更がありながら、ソフトウェアや開発用のドキュメントからはその修正箇所を確実に特定することはできない。このことから適用業務の変化にともなう保守の作業に工数や時間を要するのは、外部仕様としての情報システムの要求定義ではない。アプリケーションソフトウェアの調査による修正部分の特定に時間を要していることがわかる。

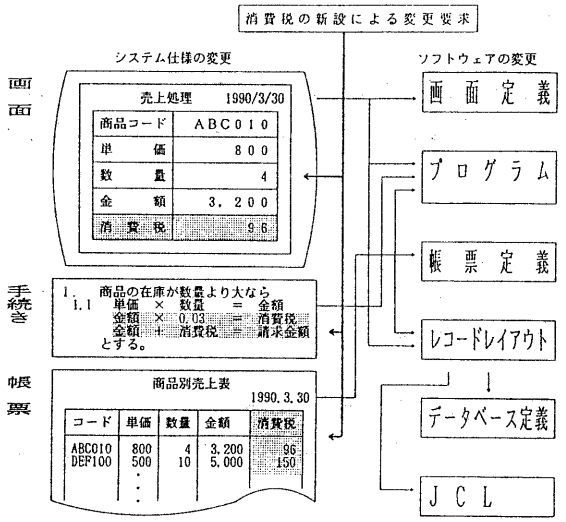

図１：ソフトウェア変更の影響の広さ

２．２　工程標準、作業標準など方法論の欠如

情報システム部門において開発の工程標準を設けているのは常識になっている。そして、各工程は更に詳細な作業手順に細分化されている。この作業手順には個々の作業の方法を述べた作業標準あるいは技術上の手法、たとえば構造化プログラミング手法のような技術標準がある。

また、ドキュメント標準は規程のフォームとともに記述されていることが多い。しかしながら、保守においてはこれらが欠如している。

保守も開発と同じく

・工程の標準化と工程別の詳細な作業手順

・作業手順別の作業標準、技術標準、および工数の標準値

・ドキュメント標準と規程フォーム

などが同じ水準で用意されていることは必要なことである。

２．３　保守用ＣＡＳＥの考え方

開発の生産性向上問題をＣＡＳＥの活用により解決しようとする方向にあると同様に、保守においてもＣＡＳＥを期待したい。保守作業で特に負荷のかかる工程は、現在のソフトウェアの調査／分析と修正したソフトウェアの品質保証である。この工程をコンピュータに支援させることを可能にしたい。

現在運用されている情報システムは、ソフトウェア、データのいずれもＣＡＳＥから利用可能な状態にある。ソフトウェアの調査／分析には、ライブラリの中のソフトウェアを解析すればよいし、品質保証にはデータの内容を対比すればよい。

このことにより、ソフトウェアの構造、要素、仕様などをデータベース化することが可能になり、保守作業が個人依存型の仕事から組織運用型への転換が可能となる。

3　ＲＥＳＣＵＥの機能

ＲＥＳＣＵＥは、調査／分析工程と検査／品質評価工程を支援するものであるが、大別して静的解析機能、動的解析機能、品質保証機能の３つの機能をもつ。今回は静的解析のみを紹介する。

3．1　情報の抽出

ソフトウェアは、プログラムだけでなく、その他にＪＣＬやデータベース定義、画面定義、ＣＯＰＹ定義、システム定義などの構成要素からなっている。

これらの各構成要素を図２のようにリバースエンジニアリング技法により静的に解析して、その中で定義されている情報を抽出し、管理対象ごとに整理して台帳を作成する。

人手の入力に依らず、ソフトウェアの各構成要素を直接入力して情報を収集するので、管理対象を正確かつ確実に管理できる。

ソフトウェア構成要素の解析では、その中で使用されているデータ項目、ファイル、データベース、画面、帳票、ＣＯＰＹなどを抽出するだけでなく、個々の命令文や定義文の細部まで解析し、それらに対する操作内容まで抽出する。

また、プログラムについては、記述言語をＣＯＢＯＬ、ＰＬ／Ｉに限定せず、アセンブラやユーティリティも解析対象としている。したがって、１つの業務を構成しているすべてのプログラムが解析されることになり、抽出情報に欠落がない。

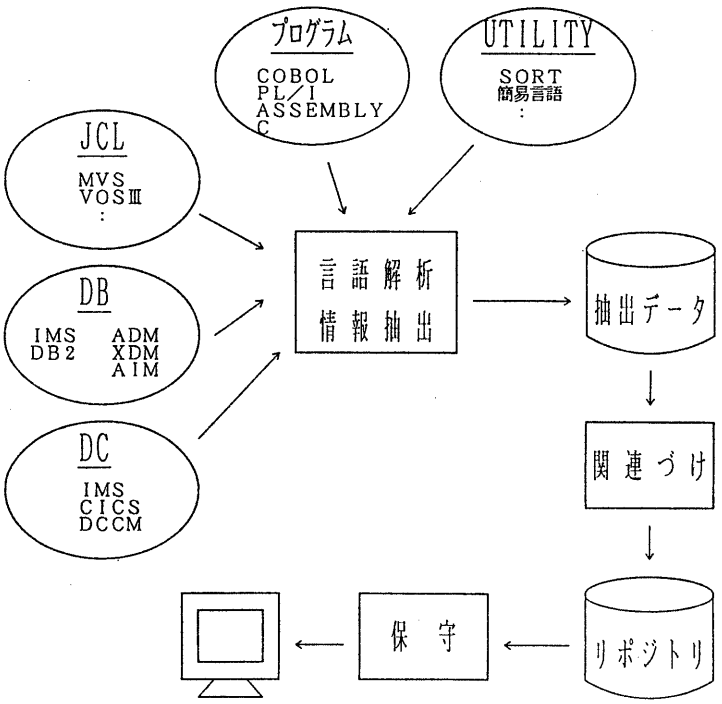

図２：ソフトウェアからの抽出・加工手順

3．2　ソフトウェア台帳の照会

上記で作成した各台帳に関して、管理対象別の一覧および個別管理対象の詳細情報を照会することができる。

なお、個別管理対象ごとに日本語名称を付与することができるので、ソフトウェアで使われている識別名（ＩＤ）がわからなくても、その業務で使用している日本語名称によって照会することができる。台帳照会は図３のようになる。

照会条件の指示により条件に合致した対象が一覧に表示される。その中から該当のものを選択することでその内容が表示される。図４はその一例である。

プログラムの照会内容として、ＪＣＬやデータベース定義の分析結果も付加した情報が表示される。

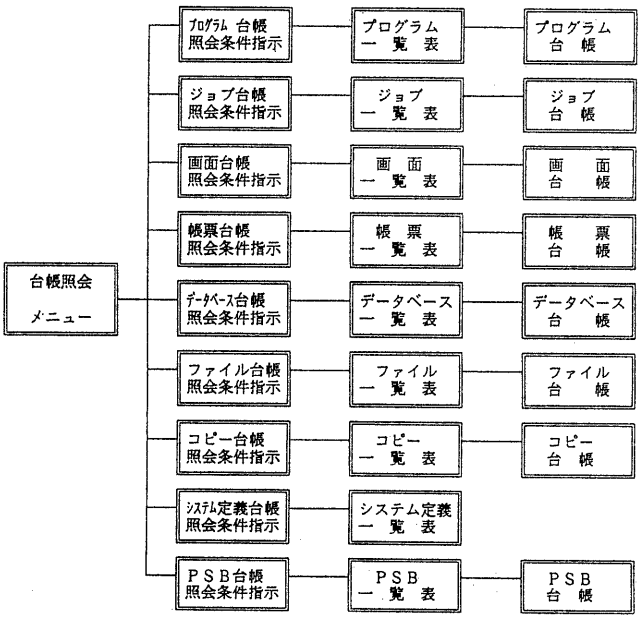

図３：台帳照会における画面構成

```
                    *プログラム台帳*
    PROG_ID  プログラム名                    更新日     言語  ステップ 数
    IMSLST10 事業所別調査統計表              1991/05/01 CCBOL     234

  DD名     JOB名   I/O データセット名          ファイル名称
SYS170   TSJK15B1 I INPUT.FILE
SYSLST   TSJK15B1 O &&TSJK15B1001001SYSLST

 DBD-ID   ＤＢＤ名称                    SEGMENT
LOGICDB1 ＬＯＧＩＣ   ＤＢ１           NAME
LOGICDB1 ＬＯＧＩＣ   ＤＢ１           SKILL

COPY-ID  コピー名称
NAKACPY0 ＮＡＫＡＣＯＰＹ
NAKACPY1 ＮＡＫＡＣＰＹ１

F1 -前画面   F2 -次画面                          F5 -Fld Help F10-キー表示
```

図４：台帳照会例

3．3　相互関連情報

ソフトウェアの保守において最も重要な影響分析を支援するのが、ソフトウェア要素の相互関連情報である。図５に示す例のようにコピー文（通常レコードレイアウトを共通に利用するのに用いる）を指定すると、そのコピーを使用している全プログラムが表示される。

レコードレイアウトの変更があった場合には、コピーを修正するとともに、これだけのプログラムはコンパイルしなおす必要があることがわかる。

ソフトウェアの相互関連は極めて多く、表１に示したようなマトリックスとなる。この黒丸のついている２者の関連が照会可能となっている。

ソフトウェア実体から人手で調べることは極めて困難な作業である。

```
 環境区分 E1              ＊影響範囲選択（コピー専用）＊

 選択項目     範囲内容
-------------------------------------------------------------------------------
 コピー       [NAKACPY1                                                    ]
```

ワイルドカードが使用できます．

F1 -前画面　F2 -次画面　　F5 -Fld Help F10-キー表示

```
 環境区分 E1                  ＊プログラム一覧表＊                    1/4
 コピー    NAKACPY1
 指示  PROG-ID    プログラム名称                    形態  言語  STEP数   更新日
-------------------------------------------------------------------------------
 [ ]   IMSLST10   事業所別調査統計表                B    COB      234  91/05/01
 [ ]   IMSSUB21   部課別学歴別人員分布              B    COB       64  91/04/15
 [ ]   IMSTST00   履用調査統計管理                  B    COB      121  91/04/12
 [ ]   IMSTST10   人事考課管理                      B    COB      123  91/04/12
 [ ]
 [ ]
 [ ]
 [ ]
 [ ]
 [ ]
 [ ]
 [ ]
```

'S' RETURNで、プログラム台帳が表示されます．

F1 -前画面　F2 -次画面　　F5 -Fld Help F10-キー表示

図５：相互関連情報の照会例

表１：相互関連情報のマトリックス

キー ＼ 検索対象	ジョブ	ジョブステップ	プロシージャ	プロシージャステップ	システム定義	プログラム	データセット	ファイル	データベース	セグメント	PSB	帳票	フォーム	オーバレイ	画面	MID	MOD	トランザクション	コピー	レコード	データ項目	リテラル	DD名
ジョブ		●	●	●		●	●	●	●	●	●	●	●	●									●
システム定義						●					●							●					
プログラム	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●		●
データセット	●	●	●	●		●		●															●
ファイル	●	●	●	●		●	●																●
データベース	●	●	●	●		●				●	●												
セグメント	●	●	●	●		●			●		●												
PSB	●	●	●	●	●	●			●	●								●					
帳票	●	●				●							●	●									●
フォーム	●	●				●						●		●									●
オーバレイ	●	●				●						●	●										●
画面						●										●	●						
MID						●									●		●						
MOD						●									●	●							
トランザクション					●	●					●												
コピー						●													●				
レコード						●													●	●	●	●	
データ項目						●													●	●	●	●	
リテラル						●														●	●		

3．4　データ項目のグルーピング

プログラムＩＤ、ジョブＩＤ、データベース名とか多くのソフトウェアの単位にはユニークな名称あるいはコードがキーとしてつけられている。これはＯＳ（オペレーティングシステム）やＤＢＭＳ（データベースマネジメントシステム）などのシステムソフトウェアからの制限なり、規則なりでユニークであることを強制されている。また記述頻度も少なくない。従って相互の関連づけを行うときには、このＩＤなり、名称なりキーを参照することで可能となる。

一方、情報システムの最小の管理単位であるデータ項目は１つのプログラム内でユニークであれば良く、従って同じ内容の項目がプログラムによって異る名称を持ったり、異なる内容の項目に同じ名称が付与されることがある。このためデータ項目の相互関連づけは、特別な方法でこの問題を解決しなければならない。

アプリケーションソフトウェアを分析して、データ項目相互間の関係を明らかにし、直接影響を与え合うデータ項目を集める。これによりプログラマが任意につけた項目名でも影響分析が可能となる。

ＲＥＳＣＵＥにおいては、基本的にデータ項目間の結びつけを２つの方法で行っている。

第１は、実質的には同じ領域に異なった項目名が命名されている場合

第２は、データ項目が実行時に移動される場合

の２つのケースである。

第１の例としては、図６に示したように同一データセットの同じ位置、桁数のものは同じとみなす場合である。これはＪＣＬを仲介としてプログラム上のデータ定義とデータセット名を結びつけることでできる。

もう１つの例としては、メインプログラムとサブプログラムの結合において、連絡領域につけられる項目名を結びつける方法である。

第２の例は、１つのプログラムの命令文を解析し、図７のように移動命令の移動元と移動先を結びつける方法である。

これらの方法で、従来機械的に結びつかなかった異なる項目名のデータ項目間同士を論理的に結びつけられるようになった。

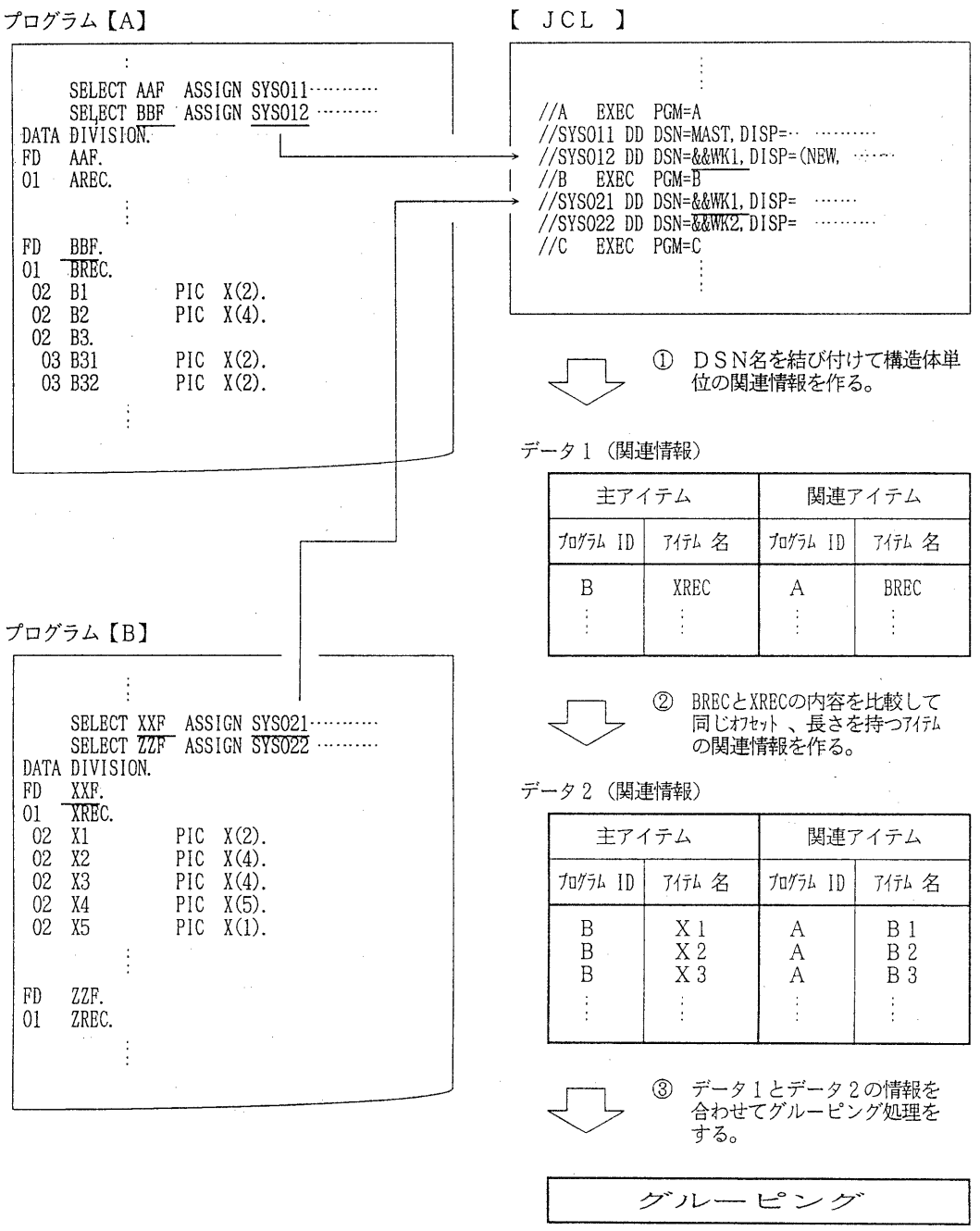

図６：グルーピング（データセットでの結びつき）

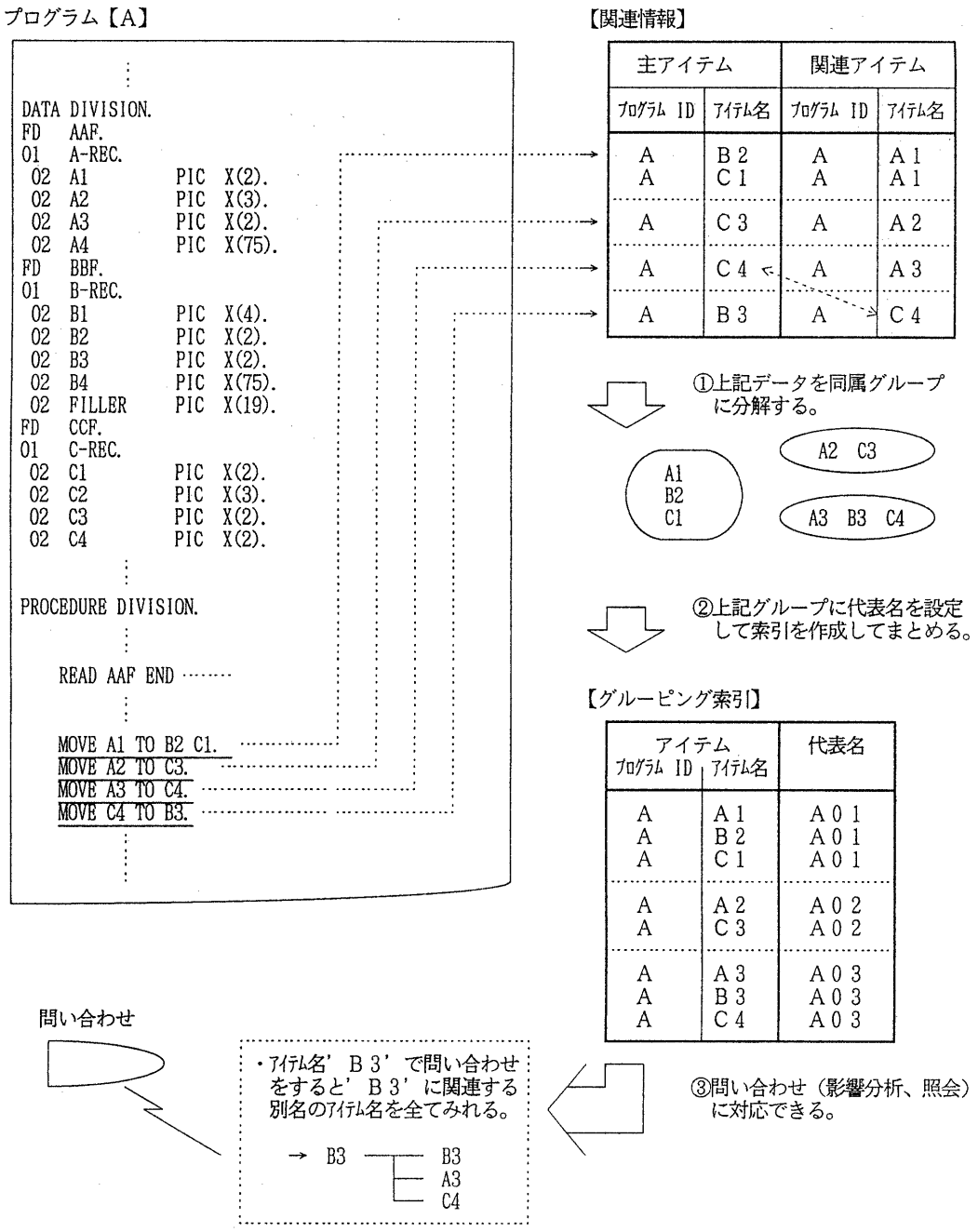

図7：グルーピング（項目移動での結びつき）

4　おわりに

保守支援システムによって保守作業は変革し、その生産性は従来の方法より向上する。しかしながらこの状態は現状からの改善であって、最も望ましい理想状態とはいえない。既に開発された情報システムのソフトウェアを前提とした保守である。保守においては、ソフトウェアの構造や表現方法を根本的に変えることはできない。

データベース、プログラム言語、開発方法も現状維持である。新しい方法論、開発用ＣＡＳＥの採用も保守の領域を越えている。

従って保守支援システムにより、保守の本質が解決したわけではない。しかしながら、直ちに着手できて、短期間で効果を発揮する方法として採用する価値はある。ソフトウェアをコンピュータを使って管理するステップは、現存のソフトウェアを利用する限り、１度は通過しなければ次の理想の状態に移行することは極めてむづかしい。

この保守支援システム「ＲＥＳＣＵＥ」の体験から次のステップとしてソフトウェアの全面的なつくりかえをする再構築支援システムへ進みたい。

参考文献

(1)本村昭二：ソフトウェア保守用ＣＡＳＥ　ＲＥＳＣＵＥの開発
日経コンピュータ　１９９１．１１．１８号