保守手順に適合する文書情報の編成法†

野村 恵美子†† 落水 浩一郎†††
中村 圭吾†††† 小川 正明††††

本論文では，事務処理システムにおいて，システムの機能に対する変更が要求されることにより発生する保守を対象として，変更対象プログラムの特定の際に必要となる作業を支援するための情報を記述する保守用情報スキーマを提案する．本スキーマは開発時文書を分析することにより設計情報を抽出し，保守者が変更箇所を特定する手順を分析することにより，変更箇所を特定する際に必要な情報を設計情報に付加して作成した．このスキーマに従って実際のシステムからデータを作成し，実験的な保守活動に適用した結果，従来，保守者の経験に依存して進められてきた保守要求に対応するシステム構成要素の特定とその変更に伴う影響範囲の抽出の作業を明示化し，効率化する効果が確認された．

1. はじめに

ソフトウェアのライフサイクルにおいて保守にかかる費用は，全体の 40〜80% をしめるといわれており[1),2)]，保守コストの削減が強く望まれている．

保守コストを削減するために，いくつかの方面から研究が進められている．一つは，開発されるソフトウェアの保守性を高めようとするものである．これには，理解しやすいプログラムを作成する設計方法の開発[3),4)]や，品質を評価しながら開発をすすめるための評価指数の開発[2)]等が含まれる．もう一つは，保守作業の効率化を図るものである．たとえば，プログラム理解を支援するツールの開発[5)]，開発時に作成される文書情報をデータベース化し，保守時に必要な情報を効率的に提供する環境の開発等である[6)〜8)]．

本論文では，保守支援環境のデータベースに保持させる情報内容について述べる．文書情報のデータベース化には，標準開発工程から文書を抽出し，章立て等の文書内の構造，開発活動によりつけられる文書間の関係を定義する方法がある[9)]．このような文書情報の定義により，開発過程にそった文書情報の追跡が容易になる．しかし，保守時における文書の参照状況を調べると，頻繁に参照されている文書は，多種類の開発時文書の中のごく一部であり，開発過程を追跡するような方法での参照は行われていない．このことから，開発時文書の構成は，保守時の情報検索要求には必ずしも適切ではなく，保守の作業過程に適した構造の情報が必要であると考えられる．

実際の保守作業は，開発経験者のようなシステムに対する知識が十分にある人により行われ，しかも，時間上の制約が大きいこと等もあり，作業過程を記録する文書はほとんど作成されない．そのため，ユーザから保守要求を受け付けてから，テストによりユーザが変更結果を確認するまでにどのような作業が行われるのか明らかではない．そこで，事務処理システムを対象に，開発時文書と保守要求の分析により，保守作業の支援に必要な情報を抽出し，構造化した．この情報構造を実験的に保守作業に適用して，保守における効果の確認を行った．

2章では，開発時文書とユーザからの保守要求の分析結果について述べ，3章で，その結果に基づいて設計した，保守用データベースに保持する情報構造を示す．4章では，これを実験的な保守作業に適用した結果と考察について述べる．

2. 保守情報の分析

ユーザから変更要求を受け付けてから，プログラムを修正するまでの過程は，ほぼ，次の二つの作業に分けられる．1）ユーザからの保守要求に基づき，変更するプログラムを特定し，保守要求をプログラムの改訂仕様に変換する．2）ソースリストを読んでプログラムを理解し，改訂仕様に従ってプログラムを修正する．1）の作業は，開発時文書が整備されていても，開発経験者のようなシステムに対する知識が十分にある人以外が行うことは，非常に困難である．本論文で

† A Documentation Schema Fit for a Maintenance Procedure by EMIKO NOMURA (Graduate School of Electronic Science and Technology, Shizuoka University), KOICHIRO OCHIMIZU (Department of Computer and Information Science, Faculty of Engineering, Shizuoka University), KEIGO NAKAMURA and MASAAKI OGAWA (Sumisho Computer Service Corporation, Ltd.).

†† 静岡大学大学院電子科学研究科
††† 静岡大学工学部情報工学科
†††† 住商コンピューターサービス(株)

は，1）の作業を対象に，支援のための情報を設計する．

上記1）の作業において，ユーザは，システムから出力されるデータの計算方法，システムに入力するデータの入力方法としてシステムの機能を理解しており，これらのデータや機能を使って保守者に変更を要求する．通常，このような機能は，いくつかのプログラムにより実現されている．プログラムは，一定のスケジュールで起動され，データの流れを形成する．このデータの流れの入口と出口がユーザからの入力データとユーザへの出力データになっている．したがって，ユーザからの保守要求に基づいて変更プログラムを特定する過程では，データの流れ，プログラム，機能，および入出力データの間の関係の把握が必要であると考えられる．

本章では，まず，開発時文書を使ってこれらの関係を理解する場合の問題点について述べ，次にプログラムの起動によりどのようにデータの流れが形成されるのか，保守要求から変更プログラムを特定するために，これらを機能とどのように対応づけるかについて述べる．

2.1 開発時文書の記述体系

プログラム本数100本程度の実際に開発されたシステムの文書を素材として，開発時文書の作成過程を分析した結果，図1に示すように文書体系が得られた．図1において，□は文書を示し，□内に各文書の記述内容を説明する．

文書の作成過程は，3段階にわけられる．レベル1は，システム化する業務を分析することにより，ファイル，データベースのようなシステム内のデータを導入し（文書⑭），これらのデータを介した新しい業務フローを作成する（文書⑧）．業務の分析では，ユーザとシステムとの間のインタフェースとなるデータ（入力票，帳表）の洗い出し，業務内の主要な機能の抽出，業務間を流れるデータの調査が行われる（文書①～⑤）．

レベル2では，再構成された業務フローに基づいて，詳細な業務分析を行い，システムの構造が設計される．業務の分析結果は，独立した文書としては残されない．フォーマット等のデータの詳細な構造が定義され（文書⑦，⑯），業務フローは，ファイル等のデータとプログラムからなるデータフローに詳細化される（文書⑩）．それぞれの業務で実施される機能は，プログラムに割り当てられ，プログラムを構成するコンパイル単位の基本設計書として文書化される（文書⑫）．

レベル3では，文書⑩に基づいて，オペレータによるプログラムの起動単位（ジョブ）が設計され（文書⑪），データの定義（文書⑦，⑯）と基本設計書（文書⑫）に基づいて，コンパイル単位の詳細設計が行われる（文書⑬）．

開発時文書の編成上の特徴は，システムの構造が詳細化されるに従って，システムの機能に関する記述も詳細化されることにある．開発初期に作成される文書は，業務名，帳表名，ファイル名等を使って記述され，計算式のような，業務で行われる機能の具体的な内容は含まれない．機能の具体的な内容は，開発の後半に作成される基本設計書，仕様書（文書⑫，⑬）に記述されるが，これらの文書は，プログラム，コンパイル単位といったシステムの構成要素を単位に編集されている．

そのため，ユーザから保守要求を受け付け，開発時文書に基づいて変更内容の確認を行う際に，次のような問題がある．すなわち，開発初期に作成される文書では，記述内容が不十分であり，しかも十分な記述内容をもつ文書を参照するためには，変更が要求されている機能がどのプログラムのどのコンパイル単位で実現されているかについて，保守者が知らなければならない．保守するシステムに関する保守者の知識を仮定せずに保守作業を支援するためには，文書情報が以下のように編成されていることが必要である．

- ユーザからの保守要求に基づいて，要求に対応する機能の記述が抽出でき，それを用いてユーザと保守者の間で保守要求の確認が行える．
- その結果に基づいて，変更するプログラム等が特定できる．

2.2 データ処理の構造

大規模な事務処理システムでは，多数のプログラムを一定のスケジュールに従って起動することにより業務を遂行する．各プログラムの処理は，ファイル，データベース，およびオンラインプログラム間の通信に使われるメッセージエリアのデータを通じて他のプログラムの処理と関連する．ある処理の結果がメッセージエリアのデータのみに反映される場合，その処理が関係する範囲は，そのメッセージエリアを共有している数個のオンラインプログラムにとどまる．しかし，ファイル，データベースのデータに反映される場合，そのファイル，データベースを参照しているプログラムの処理と関連し，そこで作成されるデータを介

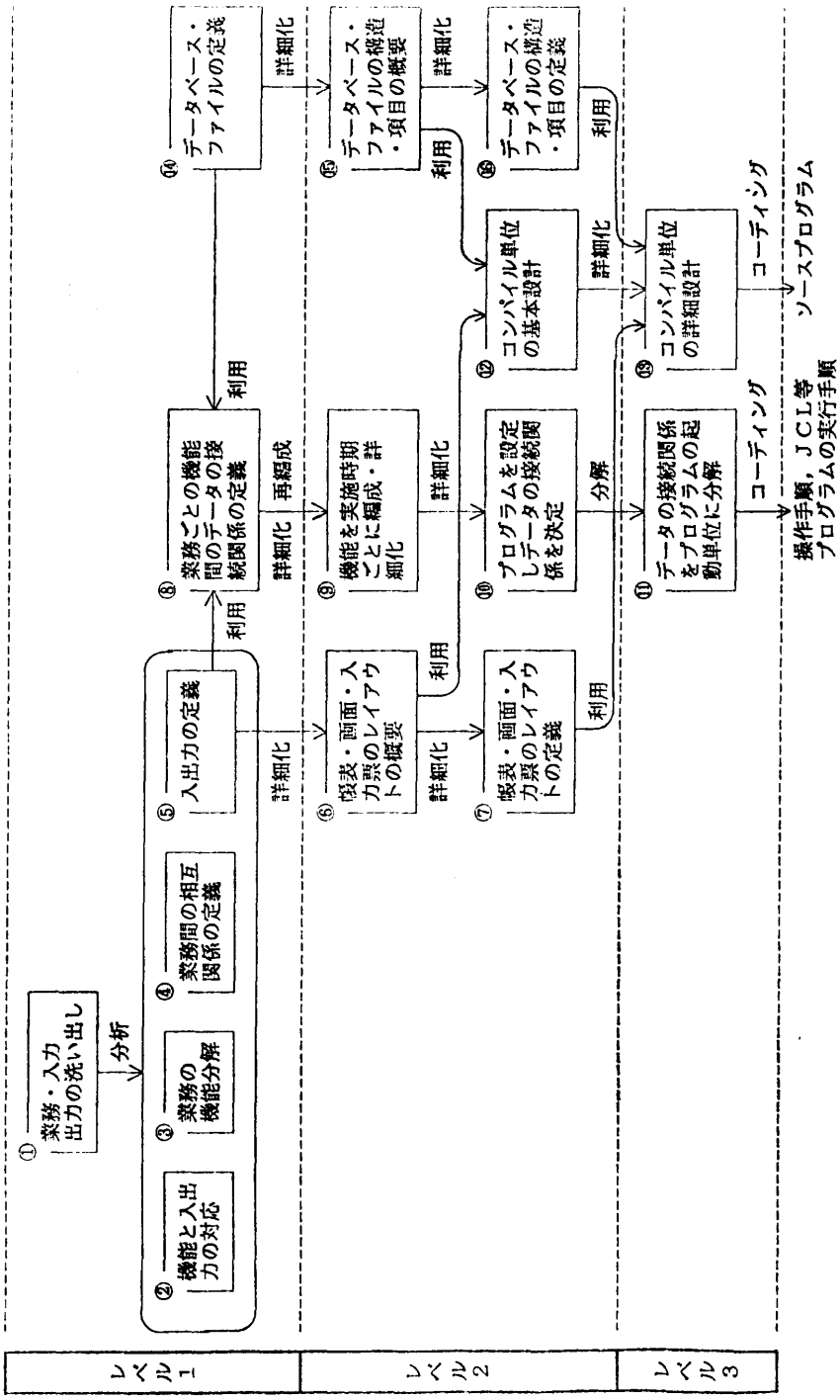

図 1　事務処理システムの文書体系
Fig. 1　Documentation for a data processing system.

して，さらに多くのプログラムの処理と関連する．したがって，システム内におけるファイル，データベース中のデータの流れがデータ処理の重要な構造となる．

システム内のデータの流れは，ファイルからファイルへデータを接続する処理を連続させることによって形成される．データの流れの端点は，画面等によるユーザ入力のファイルへの接続，ファイルから帳表への接続である．このようなデータの接続処理は，プログラムを起動することにより引き起こされる．バッチ

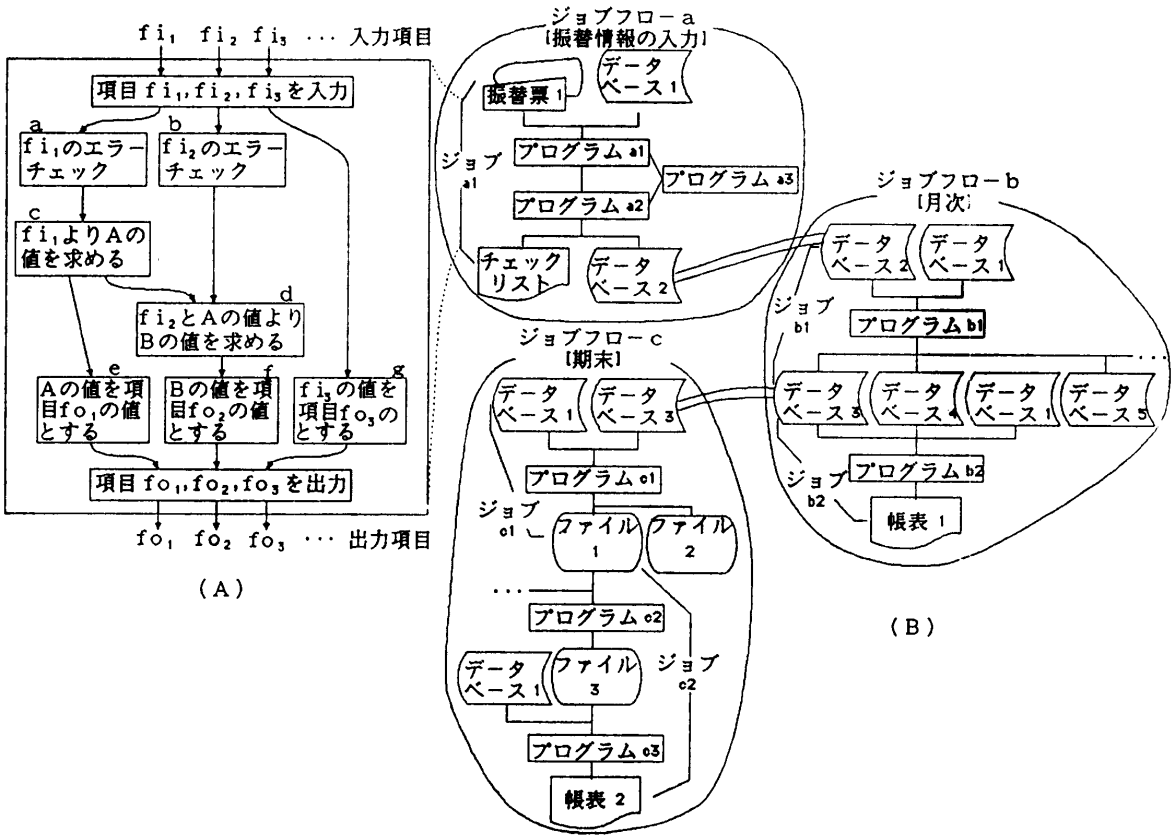

図 2 事務処理システムの構造
Fig. 2 The structure of a business data processing system.

で起動されるプログラムの場合，それぞれのプログラムが単独で接続を行う．オンラインで起動されるプログラムの場合は，一つのプログラムが単独で接続を行う場合と，複数のプログラムが連続して接続を行う場合とがある．このようなデータの接続単位となっているオンラインプログラム群をオンラインジョブとする．

データの接続は，バッチプログラム，オンラインジョブで以下のように行われる．まず，ファイル等からレコード単位でデータを入力し，レコードを構成する項目の値を使って一定の手順で計算を行う．計算結果は，再びレコードに編集され，ファイル等に出力される．レコードの入力から出力までのデータ処理の過程は，ソースプログラムの制御フロー，およびデータフローを解析することにより，図 2(A) に示すような入力項目を出力項目に接続するパスとして抽出できる．これを**項目の接続パス**とよぶ．図2(A)において，a～dは入力項目を出力項目に接続するために行われる計算処理であり，計算処理の規模に応じて，コンパイル単位やステートメント群に対応する．

データの流れは，バッチプログラムとオンラインジョブに対して以下のような起動制御を行うことにより，項目の接続パスによるデータの接続が連続的に起こり，形成される．連続して起動されるいくつかのバッチプログラムは，バッチジョブに編成され，バッチジョブが起動されたときに一定の順序で起動される．バッチ，オンラインとも，それぞれのジョブには，起動のタイミングが与えられている．タイミングには，“月次”，“期末”のような定期的な繰返しと，“振替情報の入力”のようなシステム外部からの非定期的なデータの到着とが含まれる．同一のタイミングで複数のジョブが実行される場合，これらのジョブ間に一定の起動順序が与えられる．同一のタイミングで起動されるジョブ群をジョブフローとよぶ．すなわち，各々のプログラムは，タイミングの発生，ジョブ間の起動順序，ジョブ内の起動順序に従って起動される．

データの流れを把握する際には，ある接続処理で出力されたデータが次にどの接続処理で入力されるかと

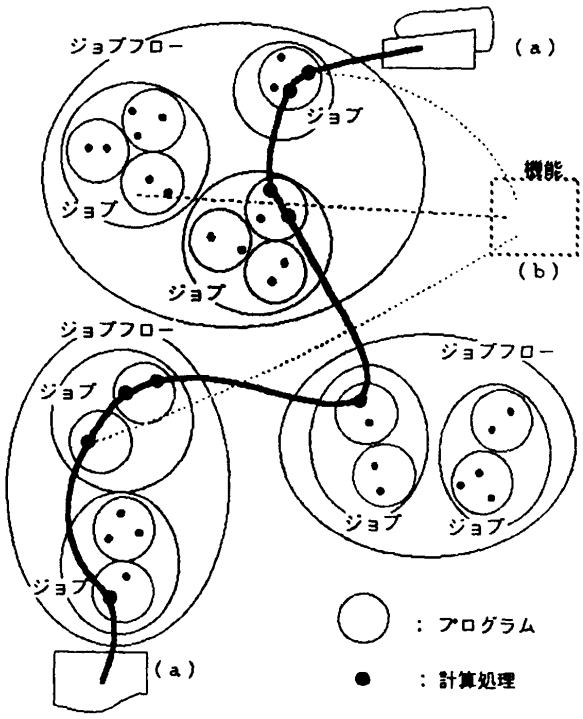

図 3 データの流れと機能
Fig. 3 A data flow and function.

いうことが重要である．そこで，起動順序により決まるプログラム間の順序関係から，次のようなデータの入出力に基づく順序関係を抽出する．すなわち，あるファイルがプログラムAで出力され，プログラムBで入力されており，しかもこのファイルからプログラムBに入力されるデータに，プログラムAからの出力データが含まれているとき，AとBの間の順序関係を抽出する．

このようなデータに対する入出力の関係から決められるプログラム間の実行順序の関係を図2(B)に示す．ジョブフローa，bにおけるデータベース2，ジョブフローb，cにおけるデータベース3のようなジョブフロー間を結合しているデータをジョブフローの入力（出力）データ，ジョブb1，b2におけるデータベース3，4，ジョブc1，c2におけるファイル1のようなジョブ間を結合しているデータをジョブの入力（出力）データとする．

2.3 保守要求と変更箇所の特定

2.2節で述べたシステムの構造を図3に示す．プログラム，ジョブ，ジョブフローといったシステムの構成要素の間には，○の包含関係で示すような階層構造があり，太線で示すデータの流れにより，構成要素が互いに関係づけられる．以下では，ユーザの保守要求に基づいて変更プログラムを特定するために，このような構造に付加する機能情報について述べる．

システムの運用開始後1～2年の間に実際に発生した保守要求を調査したところ，多くの保守要求は，次に示す二つの型のいずれかで記述されていた．

1) 出力帳表，帳表に出力されたデータと入力したデータの関係，現行のシステムに対してデータ入力を行った際のシステムの応答等が示され，それらに対して，変更が要求される．データ入力手順の変更，入力時エラー処理方式の追加・変更，フォーマットの変更，項目値の計算方法の変更等が含まれる．
2) システムにより実現されている機能が指定され，計算方式の変更，処理対象とするデータの追加・変更等が要求される．

1)では，図3(a)に示すように，システムとユーザのインタフェース部分にあたるデータを使って変更が要求される．このような要求から変更箇所を特定する場合，フォーマットの変更のように，ファイル等のシステム内データに対する変更を伴わない場合，入力票，帳表に対する入出力を行っているプログラムを調べ，そのプログラム内で変更する箇所を調査することになる．しかし，システム内のデータに対する変更になると，項目の接続パスを接続することによりできる項目レベルのデータの流れを追跡し（図3太線部分），修正すべき接続処理を行っているプログラムをつきとめなければならない．

2)では，図3(b)に示すように，システムの構成要素とは独立にユーザが認識している機能があり，それを使って変更が要求される．この機能は，通常，いくつかの計算（部分機能）から構成されており，それぞれの計算が，各々の実施時期に応じて異なったプログラムで実現される．したがって，変更箇所の特定には，部分機能ごとに実現しているプログラムとの対応情報が必要である（図3点線部分）．

3. 保守用情報スキーマの設計

本章では，前章での分析の結果，保守時に必要であると考えられる次の二つの情報をスキーマとして表現する．

①図3の太線で示すようなシステム内におけるデータの流れ．
②図3の点線で示す，ユーザが認識する機能と，そ

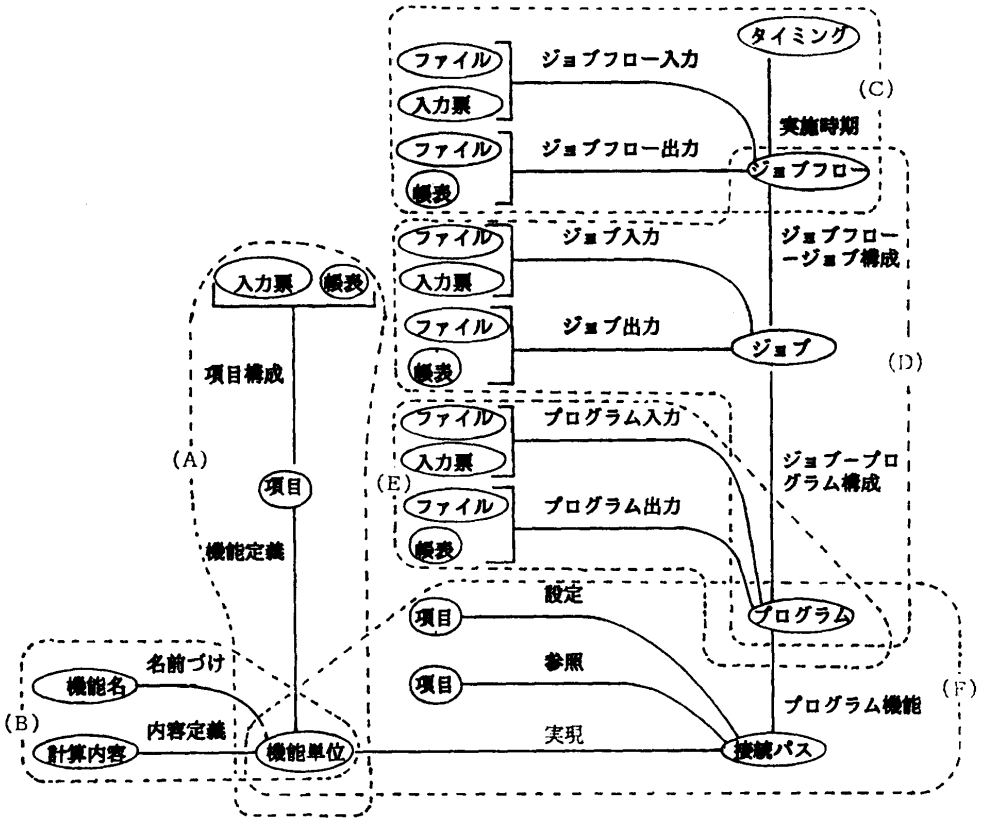

図 4 保守用情報スキーマ
Fig. 4 An information schema for software maintenance.

の実現部分との対応.

これらから**対象** (object, 図4の○) と**関連** (relationship, 図4の—) を抽出し, 実データベーススキーマには依存しない形で, 保守用情報スキーマを設計した (図4実線部分).

3.1 データの流れ

図2(a), (b)に示す項目レベルのデータの流れとファイルレベルのデータの流れの両方が追跡できるようにスキーマを設計した. ジョブフロー, ジョブ, プログラムを, それぞれの入力データを出力データに接続する接続点とし, 接続点と接続点に対する入力, 出力データ (ファイル・データベース, 帳表, 入力票) との間の関連を保持する. ただし, オンラインプログラムに対する入出力の関連は, 各プログラムで直接入力, または出力するデータに対して関連をつけるものとする. ファイルレベルのデータの流れは, 接続点間で入出力の関連をたどることで作成するものとする. また, 接続点を表す対象間にシステム構成部分間の階層を表す関連をつけ, さらに, ジョブフローに対してタイミングを対応させる.

項目レベルの場合も同様に, 項目の接続パスと, 項目の接続パスで設定, 参照される項目との関連を「設定」,「参照」で保持し, データの流れは, この関連をたどることで作成するものとする. 対象「接続パス」は, それを含んでいるプログラムとの間に関連「プログラム機能」で対応をつける.

3.2 機能とシステム部分との対応

機能の具体的な内容は, コンパイル単位ごとに編集される基本設計書 (図1文書⑫) に記述される. この記述をユーザが述べる保守要求から直接検索でき, その結果に基づいて変更対象プログラムを特定できるようにするため, 機能の記述を以下のように再構成した.

バッチプログラム, およびオンラインジョブを構成するソースプログラムを解析することにより, 図2(A)のような項目の接続パスを抽出する*. 抽出した項目の接続パスは, 入力項目値に対して一定の計算処理が行われ出力されるもの (図2 fi_1 から fo_1 へのパ

* 今回の実験では, 手作業で作成した. 現在抽出プログラムを開発中である.

ス，fi_1, fi_2 から fo_2 へのパス）と，入力項目値を出力項目へ転送するだけのもの（図2 fi_3 から fo_3 へのパス）とに分けられる．前者には，次のような計算処理が含まれる．

①ユーザからの入力データに対するエラー処理，デフォルト処理等の入力時処理（図2 a部，b部）．

②出力項目値を算出するための計算式，計算式を適用するデータを選択するための条件判断等の計算処理（図2 c部，d部）

このような計算処理を単位として機能を記述するものとし，これを**機能単位**とよぶ．

図1文書⑫から機能単位に対応する記述部分を抽出して，**図5**に示すような文章形式の説明を作成し，「計算内容」とする．さらに，機能名は，文書⑫より次のようにデータ名を抽出し，「減価償却額の計算」のようにデータ名を含む名前を与える．

- ①のような機能単位の場合，機能単位を含む接続パスで参照される項目名．
- ②のような機能単位の場合，計算の結果得られるデータ値に対する名前．

また，いくつかのバッチプログラム，またはオンラインジョブから得られた機能単位が，全く同じ内容の計算内容に対応する場合，これらを一つの機能単位とする．

機能単位と，これを抽出する基になった項目の接続パスとの対応は，関連「実現」で保持する．

このように情報構造を定義することにより，ユーザから指示される保守要求に含まれる機能名，データ名等を使って，機能名，計算内容を直接検索し，保守対象となる機能単位を抽出して，計算内容の記述に基づいて保守要求の確認を行う．保守対象として確定した機能単位より，関連「実現」，「プログラム機能」を使って，変更対象プログラムが特定できる．

さらに，保守要求が帳表，入力票の項目に基づいて述べられる場合を考慮して，次のような関連を定義した．

- 関連「項目構成」

  対象「帳表」，「入力票」とそれらを構成する「項目」との間に対応をつける．

- 関連「機能定義」

  図3太線部分に示すように，項目レベルのデータの流れにより，入力票の項目が帳表の項目に接続されるとき，項目の接続パス上の計算処理に対応する機能単位を次のように入力票の項目または帳表の項目に対応づける．上記①の計算処理の場合，機能単位を入力票の項目に対応させ，②の計算処理の場合，機能単位を帳表の項目に対応させる．

```
a．勘定科目が土地，電話権のいずれかの場合
   減価償却額＝0
b．勘定科目が建物，機械のいずれかの場合
   ①年間償却額を求める．
     年間償却額＝期首価額×年率
   ②減価償却額を求める．
     年間償却額≦期首価額－残存価額ならば，
         減価償却額＝年間償却額/12
     年間償却額＞期首価額－残存価額ならば，
         減価償却額＝(期首価額－残存価額)/12
c．その他の勘定科目の場合
   ①年間償却額を求める．
     年間償却額＝(取得価額－残存価額)×年率
   ②減価償却額を求める．
     年間償却額≦期首価額－残存価額ならば，
         減価償却額＝年間償却額/12
     年間償却額＞期首価額－残存価額ならば，
         減価償却額＝(期首価額－残存価額)/12
```

図5 計算内容の記述例
Fig. 5 An example of function description.

4. 評価実験

設計した保守用情報が，保守の作業過程においてどのような効果があるかについて確認するため，図4のスキーマを関係データベースとして実現し，開発時文書とソースコードからデータを作成し，これを用いて実験的に保守を行った．本実験では，保守の作業過程を明らかにし，各作業で行われたデータベースの検索過程を調査した．

4.1 関係スキーマの設計

図4のスキーマを関係データベースのスキーマに編成する際に，以下のような方針をとった．

1）関係が実験者にとってなじみやすい概念に対応すること．

2）関係の結合操作を行わずに，できるだけ多くの情報をみることができるように，関係の数はなるべく少なくすること．

3）関係の実現値に重複するデータが多く存在することになっても，現時点ではよいとする．

その結果，図4の点線に示すようにスキーマを分解し，それぞれの部分に対応して関係を設計した．

4.2 実験の概要

データ作成の対象としたシステムは，2章の調査で利用したシステムで，期末，上期期首，下期期首，月

次，いくつかの特定月と，随時（入出力発生時）に実施されるジョブフローから構成される．この中から，ユーザ入力から帳表出力までのデータの流れが存在する比較的小さな部分として，随時実施される四つのジョブと，月次に実施される三つのジョブを選び，データを作成した．実験設備には，IBM 5540 上の informix を用いた．

2.3 節で述べた二つの型の保守要求をいくつか想定し，変更対象となるプログラムの特定に関して，どのような検索が行われるかを追跡した．実験者は，システムの開発，保守には十分な経験があるが，保守対象としたシステムに対しては，概要についての知識があるのみで，システム内のデータの流れなど，詳細についての知識はなかった．

4.3 実験結果の考察

保守作業は，**図 6** に示すような過程で行われた．保守要求の受付では，ユーザの要求から入力票名，帳表名，項目名，機能名等を抽出する．ユーザの要求が計算式等の機能に対する記述である場合，要求中に含まれるデータ名等を使って，図 4 (B)部の計算内容を検索し，機能名を得る．

影響調査では，変更要求に応じて，入力票，帳表，機能間の関係が調査される．ここでは，図 4 (A)部，(B)部が利用された．変更が入力票名，帳表名，およびそれらに含まれる項目名で要求された場合（例えば，「帳表××の項目○○を変更したい」），まず，指示された項目名を使って，同名の項目を含む他の入力票，帳表を検索し，変更の対象となる入力票・帳表の項目をユーザに確認する．次に，入力票名，帳表名，項目名から変更対象となった項目を計算する機能（機能名，計算内容，機能単位）を検索し，計算式等に対する変更内容をユーザに確認する．さらに，他の機能に対する変更の影響を調査する．例えば，機能名が「減価償却計算」である機能の中の計算式「償却額＝期首価額×償却率」を変更するとき，「償却額」を計算している他の機能，計算式で「償却額」を使っている機能等を検索し，これらの機能に対する変更の有無をユーザに確認する．変更が機能名で要求された場合は，まず機能に対する調査を行い，次に変更が反映される入力票，帳表を調査する．

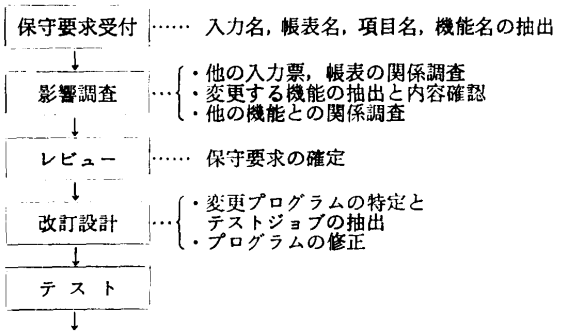

図 6 保守手順
Fig. 6 The maintenance process.

レビューでは，影響調査の結果に基づいて，変更内容（変更する入力票名，帳表名，項目名，機能単位，機能名と，計算内容の変更部分）を決定し，保守要求を確定させる．

改訂設計では，確定した保守要求に基づいて，変更するプログラムとテストするジョブが特定され，ソースコードが修正される．ここでは，図 4 (F)部のデータを使って，機能単位からそれを実現しているプログラム名を検索する（変更プログラムの特定）．次に，(D)部のデータを使って，変更プログラムを含むジョブを特定し，ジョブの入出力を追跡することにより，入力票の入力から帳表の出力までのデータの流れをジョブレベルで抽出し，テストするジョブ名を特定する．その後，ソースコードを読み，確定した保守要求に従って，プログラムを修正することになる．最後に，変更結果をユーザが確認するためのテストが行われる．帳表，プログラム，ジョブは，十分に，または正しく抽出されていた．

上記の保守過程で，データベースを用いた検索により効果的に情報が得られた作業は，影響調査と改訂設計の中の変更プログラムの特定とテストジョブの抽出であった．2.1 節で述べたように，従来，文書化は保守時に検索しやすい形ではなされていないため，影響調査は，開発者の知識に大きく依存する作業であった．図 4 のスキーマをもとに設計したデータベースを用いた保守実験では，実験者は，保守するシステムについて十分な知識はなかったが，データベースを検索することで変更要求に関連する入力票，帳表，機能をもれなく抽出することができ，保守要求を確定させるために十分な調査が行えた．また，確定した保守要求から，容易に変更するプログラムを特定できたことから，図 4 のスキーマに従ってデータを編成することで，保守するシステムについて十分な知識がなくても，保守要求の受付から変更プログラムの確定までの作業が可能になると判断される．

本データベースを使って特定されるプログラムにおいて，さらにそのソースコードのどの部分を変更すべ

きかを特定するには，制御フローの解析，変数レベルでのデータフローの解析等のより詳細な情報が必要であり，今後の課題である．

5. おわりに

保守においては，最終的には，ソースリストに対して，詳細な調査を行い，変更を行わなければならない．しかし，大規模な事務処理システムでは，まず，このような調査の対象となるプログラムを特定することが必要である．本スキーマは，従来，保守者の経験に依存して進められてきたこのレベルの作業の内容と手順を明示化し，作業を効率化する効果があった．

今回の実験では，すでに開発されたシステムを対象に，開発時に作成された文書，またはソースプログラム群から，手作業でデータを作成したが，この作業には，かなりの期間を要した．本来このようなデータは開発時に作成すべきであり，本論文の成果をふまえて，ソースコードを解析して項目の接続パスを抽出するプログラム，また，開発過程と連動して，その他のデータベース内容を収集する手段を検討中である．

参考文献

1) Lientz, B. P., Swanson, E. B. and Tompkins, G. E.: Characteristics of Application Software Maintenance, *Comm. ACM*, Vol. 21, No. 6, pp. 466-471 (1978).
2) Yau, S. S. and Collofello, J. S.: Design Stability Measures for Software Maintenance, *IEEE Trans. Softw. Eng.*, Vol. SE-11, No. 9, pp. 849-856 (1985).
3) Yourdon, E. and Constantine, L. L.: *Structured Design*, Prentice-Hall, Englewood Cliffs, NJ (1979).
4) Jackson, M. A.: *Principles of Program Design*, Academic Press, London (1975).
5) Warren, S.: MAP: a Tool for Understanding Software, *Proceedings 6th ICSE*, pp. 28-37 (1982).
6) Horowitz, E. and Williamson, R.: SODOS—A Software Documentation Support Environment: Its Use, *Proceedings 8th ICSE*, pp. 8-14 (1985).
7) Penedo, M. H. and Stuckle, E. Don: PMDB—A Project Master DATABASE for Software Engineering Environments, *Proceedings 8th ICSE*, pp. 150-157 (1985).
8) Bersoff, E. H., Henderson, V. D. and Siegel, G.: Software Configuration Management: A Tutorial, *Computer*, Vol. 12, No. 1, pp. 6-14 (1979).
9) Horowitz, E. and Williamson, R.: SODOS—A Software Documentation Support Environment: Its Definition, *IEEE Trans. Softw. Eng.*, Vol. SE-12, No. 8, pp. 849-859 (1986).

(昭和61年10月20日受付)
(昭和62年7月9日採録)

野村恵美子（正会員）

昭和34年生．昭和57年静岡大学工学部情報工学科卒業．昭和59年同大学院修士課程修了．現在同大学院博士課程在学中．ソフトウェア開発・保守支援環境に興味をもつ．

落水浩一郎（正会員）

昭和21年生．昭和44年大阪大学基礎工学部電気工学科卒業．昭和49年同大学院博士課程（物理系専攻）修了．工学博士．同年静岡大学工学部情報工学科講師．昭和50年同助教授．ソフトウェア工学の研究に従事．ソフトウェア開発・保守支援環境の構築に興味をもつ．電子情報通信学会，日本ソフトウェア科学会各会員．

中村 圭吾

昭和22年生．昭和41年大阪市立都島工業高等学校電気科卒業．昭和45年住商コンピューターサービス(株)入社．現在，システム営業第8部副部長．

小川 正明

昭和15年生．昭和39年大阪工業大学電子工学科卒業．昭和45年住商コンピューターサービス(株)入社．現在，同社開発本部新規事業開発部長．AAAI，人工知能学会各会員．