ソフトウェア故障診断ドメインシェル

飯田敏幸[†] 島田茂夫[†] 河岡 司[††]

計算機システムの故障原因をメモリダンプ(MD)を用いて解析するには，オペレーティングシステムの構造や動作等に関する高度な専門知識が必要である．この知識を知識ベースに蓄えたエキスパートシステム(ES)が開発されているが，①解析中に人間の介在が必要なために，大規模なMD解析には利用できない，②専門家が知識を投入・維持管理できない等の問題があった．これらの問題を解決するために，ソフトウェア故障診断ドメインシェルFINDS (Fault Isolation Domain Shell) を実現した．ソフトウェアの故障解析の知識が制御表の構造知識，制御表の見方，制御表間の関係の3種類の知識に分割できることに着目し，知識獲得の容易なソフトウェア故障診断用のドメインモデルを考案した．FINDSは①本モデルに基づいた知識獲得機能(知識エディタ)，②解析実行機能，③説明機能で構成されている．主な技術的特徴は①知識を解析の専門家に適した外部表現（表とフローチャートの形式）と推論エンジンが扱う内部表現（フレームとルール）の2形式で表現し，外部表現形式の知識エディタにより解析の専門家自らが知識を投入し維持管理できること，②既存のMDアクセスプログラムを利用して推論に必要な情報を自動収集することにより故障解析を自動化したこと，③外部表現形式での推論過程の説明を解析知識から自動作成すること，④柔軟な構造により各種アーキテクチャへの適用を可能としたことである．MDアクセスプログラムインタフェースをいくつかのアーキテクチャ向けに作成し，MDによる故障診断ESを構築した．本ESを実際に運用した結果，大規模なMDの故障解析に適用できることが検証できた．具体的には，①解析の専門家が知識処理の専門家の助けを借りることなく知識を容易に作成することが可能であること，②故障解析時間が従来の約2/3に削減されること，③知識投入の生産性は従来のプログラム言語を利用する場合と比べ約10倍になることがわかった．

Domain-specific Shell for Software Fault Isolation

Toshiyuki Iida,[†] Shigeo Shimada[†] and Tsukasa Kawaoka[††]

Analyzing computer system failures using memory dump files need expertise of an operating system. Existing expert systems for computer failure analysis have the following problems: (a) They analyze memory dump files with human help. (b) A human expert cannot input and maintain his knowledge. FINDS, a domain-specific shell for analyzing computer system failures using memory dump files, solves these problems. It consists of (a) a knowledge acquisition facility (knowledge editor), (b) an analyzing facility, and (c) an explanation facility. Its technical characteristics are as follows: (1) Analysis knowledge is represented both as external representations (in tables and flowcharts) for experts and as internal forms for its inference engine. The knowledge editor, which converts external representation knowledge into an internal equivalent, lets a troubleshooting expert input his knowledge and maintain it by himeself. (2) FINDS makes trouble analysis automatic by using an existing memory dump access program. (3) Inference processes are explained with external representations. (4) FINDS's flexible configuration makes it applicable to many computer architectures. FINDS was applied to several computer architectures with the following conclusions. (1) A troubleshooting expert can store his knowledge without the help of a knowledge engineer. (2) Analysis is 33% faster than traditional techniques. (3) Knowledge can be input 10 times faster than is possible with knowledge of ordinary programming languages.

1. はじめに

計算機ソフトウェアは大きく分けると，オペレーティングシステム(OS)，データベース管理システム等のパッケージ，アプリケーションプログラムの3階層から構成され，さらにそれぞれが階層化された複雑な構成となっている．また，計算機システムの利用分野の拡大に伴い，維持管理すべきソフトウェアの規模は年々増加している．このように複雑かつ大規模であるがゆえに，ソフトウェア故障の原因究明には多大な労

† 日本電信電話(株)情報通信網研究所
NTT Network Information Systems Laboratories
†† 日本電信電話(株)コミュニケーション科学研究所
NTT Communication Science Laboratories

力がかかる．特に，大規模なオンラインシステムの運用中での故障は社会的にも，経済的にも非常に大きな影響を及ぼすため，ひとたび故障が発生した場合には早急な原因究明と対処が必要とされる．

運用中のシステムで故障が発生すると，故障発生時のレジスタやメモリのダンプ情報（MD）がファイルにセーブされる．このファイルが故障原因解析のキーとなる．以前は MD を紙に出力し，何千ページものリストの中から解析のキーとなるデータにマークをつけたり，抜き書きしながら解析を進めていた．この場合，解析の専門家がシステムのある場所に駆けつけるか，ファイルを運送しなければならなかったために，迅速な解析は困難であった．さらに，同時走行ジョブ数の増加やアドレス空間の拡大により，紙をベースとした解析は不可能になってきた．そこで，レジスタ名やアドレスを指定することにより対応する情報を遠隔地からアクセスできる NWP[1] や SDA[2] のような MD アクセスのための支援プログラムが開発された．このような MD アクセスプログラムを利用することにより，MD の解析者はリアルタイムにデータを見ることができるようになった．しかし，表示された情報を見ながら次に何をすべきかを自ら判断しなければならないので，依然として解析には OS の構造や動作等に関する高度な専門知識が必要であった．さらに，MD アクセスプログラムの使い方を熟知する必要があった．このような技術のある専門家は非常に少なく，さらに専門家ごとに知識の偏りがあるために，特定の専門家に解析依頼が集中する傾向があった．そこで，専門家の知識を知識ベースに蓄えたエキスパートシステム（ES）の適用が考えられ，例えば，VAX の VMS のシステムダウンの原因を解析する CDx[3] が開発されている．CDx を使った解析は利用者が以下の手順で CDx と対話しながら行われる．

【ステップ1】CDx が SDA コマンドの中から適当なコマンドを選びユーザに示す．

【ステップ2】このコマンドをユーザが DSA に入力する．

【ステップ3】得られた結果をユーザが CDx に入力する．

【ステップ4】さらに情報が必要であればステップ1に戻る．あるいは，CDx が結論を示す．

CDx は MD アクセスプログラムである SDA と結合していないために，解析者は SDA と CDx を交互に使うことになる．また，CDx への知識の投入は，知識エンジニア（KE）が解析の専門家に対するインタビューで得た知識をルールの形で表現することにより行われるため，知識の投入および維持管理には解析の専門家と KE の協力が必要となる（**図 1**）．

以前に，我々は小型機から超大型機までのラインアップを持つ汎用計算機システムである DIPS (Denden Information Processing System) の OS[4] を対象として，MD に基づき故障原因を解析するソフト故障診断 ES のプロトタイプ[5]を作成し，上記 CDx の問題点を克服した．その設計目標は以下の通りである．

①KE の協力なしに，解析の専門家が知識を投入し維持管理できること（KE レス化）

②解析中には人間の介在を極力少なくすること（自動化）

DIPS とは別のアーキテクチャをもつ計算機の MD 解析の手順も基本的には DIPS と同じであり，上記プロトタイプを基に，アーキテクチャに依存する部分を局所化したソフトウェア故障診断ドメインシェル[6] として FINDS (Fault Isolation Domain Shell) を開発した．NWP や SDA のような MD アクセスプログラムとのインタフェースを入れ換えることにより，FINDS を基に，種々のアーキテクチャに対応した ES を構築できることを確認した．

以下，FINDS の実現上の課題，システム構成と機能概要，FINDS を基に作成した ES の評価を DIPS 用の FINDS/DIPS と，CTRON 準拠の OS である POPS[7] 用の FINDS/POPS で行った結果について述べる．

```
IF BugCheck type is INVEXCEPTN
THEN Stack looks like:
SP=>            00000004
                saved FP
                FFFFFFFD
                saved R0
                saved R1
                xxxxxxxx
                Arg count
                Exception type
                Exception parameters
                ...
                Exception PC
                Exception PSL


IF BugCheck type is INVEXCEPTN
  and Exception Type is 444
THEN
Suggest looking for disk errors just before Crash.
```

図 1 CDx のルール例（文献 3）より引用）
Fig. 1 Sample rule of CDx.

2. FINDS 実現上の課題

2.1 要求条件

FINDS を実現する上での要求条件は以下の通りである.

(1) KE レス化

解析の専門家は日々の解析作業と並行して知識を作成しなければならない. 従来のように KE と専門家とが協力して知識を作成するためには, 両者のスケジュール調整が必要である. しかし, 突発的に解析作業が入ることが多いために, 専門家はまとまった時間を割くのがむずかしい. そこで, 空き時間などに解析の専門家自身が知識の作成, 修正, デバグを行える必要がある.

(2) 解析の自動化

計算機システムが大規模なため, 解析対象の MD のサイズば非常に大きく, しかも, 解析に必要な情報が数 MB (連続域とは限らない) に及ぶことがあり, CDx のような人間との対話を必要とするシステムでは実用にならない. そこで, 解析時には極力人間が介在しないようにする必要がある. 解析の自動化の実現のためには, MD から必要な情報を自動的に抽出する必要がある. この手段として, 既存の MD アクセスプログラムがある. FINDS への移行を円滑に行うには, 投入された知識が充実するまでは, FINDS を用いた解析と MD アクセスプログラムを直接利用した従来の解析の併用が条件となる. そこで, MD アクセスプログラムのインタフェースを変更しない必要がある.

(3) 汎用化

解析の対象とするシステムの計算機アーキテクチャが複数種類存在する. そこで, 計算機アーキテクチャに依存する部分を可能な限り局所化する必要がある.

2.2 実現上の課題

上記要求条件を満たすために, MD を用いたソフト故障診断ドメインシェルとして FINDS を実現する上での課題を示す.

(1) ドメインモデル

ES 構築支援ツールが提供するルールやフレーム等の知識表現の枠組みに専門家の知識をどう対応させるかが FINDS 構築上の重要な鍵となる.

(2) 専門家向け知識エディタ

ルールやフレームのような推論エンジンが扱う知識表現 (内部表現と呼ぶ) と利用者 (解析の専門家) が扱う知識表現 (外部表現と呼ぶ) とを区別し, 知識の外部表現を専門家インタフェースとする知識エディタが必要になる.

(3) 知識変換機能

知識表現を区別したために, 外部表現で表された知識を自動的に内部表現に変換する機能が必要になる.

(4) 説明機能

従来の ES 構築支援ツールは, 推論に使われたルールを順に表示する程度の説明機能しか提供していないので, 知識の中に説明のための記述を入れなければならなかった. このため, 専門家が知識投入に専念できないという問題のほかに, 専門家知識に加えて説明部分も維持管理しなければならないという問題があった. この問題を解決するためには, 内部表現での推論過程を外部表現で表すことにより説明を行う必要がある. 特に, 知識のデバグを専門家自身で行えるようにするためには, 外部表現での説明機能は必須となる.

(5) MD アクセスプログラムインタフェース

解析の自動化のために, 推論に必要なデータを MD アクセスプログラムから自動的に取り出すためのインタフェースが必要になる. MD ファイルの形式は対象とする計算機アーキテクチャごとに異なり, さらに, MD アクセスプログラムも個別に開発されているためにインタフェースが異なる. したがって, このインタフェースは MD アクセスプログラムごとに作成する必要があるので, 汎用化のためには極力 MD アクセスプログラムに依存しない構造とする必要がある.

2.3 FINDS のドメインモデル

知識投入時にも説明時にも外部表現が専門家とのインタフェースとなることから, いかに外部表現を専門家に身近なものとするかが重要となる. このような表現を設定するには, ドメインモデルの設定が最重要課題となる.

(1) メモリダンプ (MD)

処理続行できないような致命的なソフトウェアの故障を検出すると, OS は計算機システムをダウンさせる. システムコンソールが効かないような場合に, オペレータがダウンさせることもある. OS はダウンする直前にセーブルーチンを呼び, レジスタとメモリの内容を MD ファイルにセーブする. システムはダウン後再起動されると, リカバリ機能によりデータベース等のデータの復旧が行われ, サービスが再開される. サービス再開と同時に故障原因の解析が始められる. 解析はまず, コンソール情報と MD を用いて原

因となるルーチンの特定（1次解析と呼ぶ）を行い，その後プログラムの設計書やソース等を用いて故障原因の特定（詳細解析と呼ぶ）を行う．障害時以外にも，例えば OS 自身のデバグのためにセーブルーチンが起動され，MD ファイルが取られることもある．

（2） 従来の解析手順

MD を用いた従来の人手による1次解析は，以下の手順により MD アクセスプログラムを用いて，MD の必要部分を順次ディスプレイに表示することにより行われる．

【ステップ1】MD ファイルのあるセンタにログインし，MD アクセスプログラムを起動する．次に，MD ファイルをオープンする．

【ステップ2】レジスタ類を表示する．

【ステップ3】表示された部分から注目すべきフィールドを見る．必要であれば別のフィールドを見る．

【ステップ4】ここまでで判断できるならば，結論を下し，MD アクセスルーチンを終了する．結論を下せなければ，次に注目すべきメモリの範囲（先頭アドレスと長さ）を決める．

【ステップ5】ステップ4の範囲を表示する．

【ステップ6】ステップ3へ戻る．

なお，ディスプレイに表示される情報は 16 進数およびそのキャラクタ表現である（**図 2**）．また，ステップ4の注目すべきメモリの範囲とは，OS 等の制御表（テーブル）に相当し，ステップ3の注目すべきフィールドとは制御表の要素である．注目すべき制御表やフィールドは，解析の専門家の経験やトラブルシューティングマニュアルに基づいて指定される．

（3） 解析手順のモデル化

このような解析の手順は計算機アーキテクチャには依存しない．これは以下の理由による．すなわち，OS は CPU 時間，メモリ空間，周辺装置等の物理資源とジョブ，タスク，ファイル等の論理資源とを管理するリソースマネジャとして位置づけられる[8]．一般には各資源は制御表の形で管理されている．パッケージや一部のアプリケーションプログラムでも同様に資源を制御表の形で管理している．各制御表にはレジスタやアドレスが固定のマスタテーブルから（いくつかの制御表を経由して）ポインタによりたどりつくことができる．そこで，計算機ソフトウェアの故障解析の鍵は，故障の原因となった資源の状態を知ることにあると言える．資源の状態は制御表の中で管理されているので，MD から目的とする制御表の情報を得ることにより知ることができる．

以上より，従来の人手による解析手順の中では，

①表示されている制御表の構造（フィールド構成，属性等）

②制御表の見方（注目すべきフィールドとその値を見てどのように判断するか）

③制御表間の関係（次に注目する制御表へのアプローチ法）

に関する知識を用いていると言える．そこで，この解析手順をドメインモデルとして汎用化することができる．

2.4 知識表現

2.3 節で述べたように，従来の人手によるソフト故障診断作業では制御表が知識の基本単位となっていることがわかる．制御表の構造知識は事実知識に，制御表の見方と制御表間の関係はソフトウェアの仕様に基づくノウハウ知識に対応している．それぞれをテンプレートとビューと呼び，以下のような特徴を持たせる．

（1） テンプレート

制御表は構造体であり，それ自身の名称と長さで特徴づけられる．制御表はいくつかのフィールドに分かれており，フィールドごとにフィールド名，オフセット（先頭からの長さ），長さ，タイプ（文字属性，ポインタ属性，数値属性等）等の属性を持っている．

レジスタはメモリ情報と一緒に MD ファイルにセーブされているので，仮想的に制御表と見なすことができる．例えば，レジスタ

```
#/VI 2C6283C+60

ﾛﾝﾘｱﾄﾞﾚｽ ｿｳﾀｲ                16ｼﾝ ｲﾒｰｼﾞ                    ｷｬﾗｸﾀ ｲﾒｰｼﾞ
         ﾛﾝﾘ:ｼﾞｯｿ = 02C6283C(SYSTEM) : 075EC83C  RL = 3  AR = RW
02C62830 0000                          20000035                ..5
02C62840 0004 13001035 40616A6C 40616004 0F0322E4 ...5@...@.....".
02C62850 0014 00616032 00617F18 00617F18 00000000 .../2.....
02C62860 0024 038CD2A0 0384CA00 02DAAC00 00000000 ..ﾒ ..ﾊ..ﾚｬ.....
02C62870 0034 00616032 00617F18 03987000 03987000 .../2.....
02C62880 0044 039871A0 4061798C 0F02D790 00FF8001 ... @.....ﾗ.....
02C62890 0054 02C627E0 038380D8 42555345          .ﾆ'....ﾘBUSE

*** VIEW     ｺﾏﾝﾄﾞ ｼｮﾘ ｼｭｳﾘｮｳ
```

図 2 NWP による MD アクセス
Fig. 2 Sample run of NWP.

は制御表名が REG，長さが 64 バイト，REG 0～REG 15 の 16 のフィールドを持ち，各フィールドが4バイトの長さを持つ構造体と考えることができる．同様に，バッファ領域は1フィールドから成る制御表と考えることができる．このように，実際には制御表ではない領域についても MD 上では制御表として見ることができる．

このように拡張された意味での制御表の構造をテンプレートとして表現する．すなわち，テンプレートに MD 上のデータを区切る定規（テンプレート）の役割を持たせる．MD 上のデータと対応づけられたインタンスをテンプレートインスタンスと呼び，テンプレート名（制御表名）とデータの先頭アドレスで識別できる．

テンプレートの外部表現は通常の表の形式である(**図 3**)．内部表現はフレームであり，フィールドに対応するスロットにはデーモンが登録される．フィールド名が参照されると，このデーモンが起動される．デーモンは MD から読み出された制御表の実現値から，そのフィールドのオフセット，フィールド長を基に値を切り出し，さらに，タイプによりデータ変換を行う．特に，文字属性を持つフィールドが参照されると，ビュー内ではその値は 16 進数ではなく，文字列として扱うことができる．

（2） ビュー

2.2 節で述べたように，制御表の見方は手続き的な知識であり，これをテンプレート対応に外部表現としてフローチャートの形で表現する．解析の専門家はフローチャートには違和感を感じないので，KE レスを実現するためにこの形を選んでいる．制御表間の関係は，ビューからビューを呼び出すことにより表現する．ビューを呼び出すとは，ビュー名とそのビューに対応する制御表の開始アドレスを指定することである．開始アドレスは，制御表内のポインタのフィールド名や数値を返却値とする式を指定することにより表現する．なお，レジスタを見るビューを呼ぶには，開始アドレスを省略することにより実現する．このようにしてできるビューの連鎖をビューチェーンと呼び，これが1つの解析知識を相当する．

ビューの内部表現はルールである．

3. FINDS のシステム構成と機能概要

3.1 システム構成と動作環境

FINDS は**図 4** に示すように，KBMS/PC[9),10)]上に構築されている．MD アクセスプログラムを使った従来の解析時の PC と専門家を FINDS で置き換えた形態になっている．

解析の自動化のための要求条件に従い，MD ファイルの存在する計算機との間のデータの授受は MD アクセスプログラムのインタフェースだけを使うことにした．FINDS を導入しても MD ファイルのある計算機上のソフトウェアに影響は全くないため，従来の情

フィールド一覧　テーブル名：PTE　全長：184

フィールド名	ロケーション	長さ	属性
PTTNXPATPT	0	4	PTR
PTTFPID	4	3	HEX
PTTYOBI1	7	1	CHAR
C_PTTSYONFG	8	1	HEX
PTTNPID	9	3	HEX
PTTMATPT	12	4	PTR
PTTPROPT	16	4	PTR
PTTWANWEPT1	20	4	PTR
PTTWANABPT	24	4	PTR
PTTWANWEPT2	24	4	PTR
PTTPONDBPT	28	4	PTR
PTTNPNWECN	32	2	HEX
PTTSQTNO	34	1	HEX
PTTSQRNO	35	1	HEX
PTTSPTPT	36	4	PTR
PTTMARU	40	1	HEX
PTTROOT	41	1	HEX
PTTTMNO1	42	1	HEX
PTTTMNO2	43	1	HEX
PTTC1ST	44	1	HEX
PTTF1ST	45	1	HEX

図 3 テンプレートの外部表現
Fig. 3 An external representation of a "template."

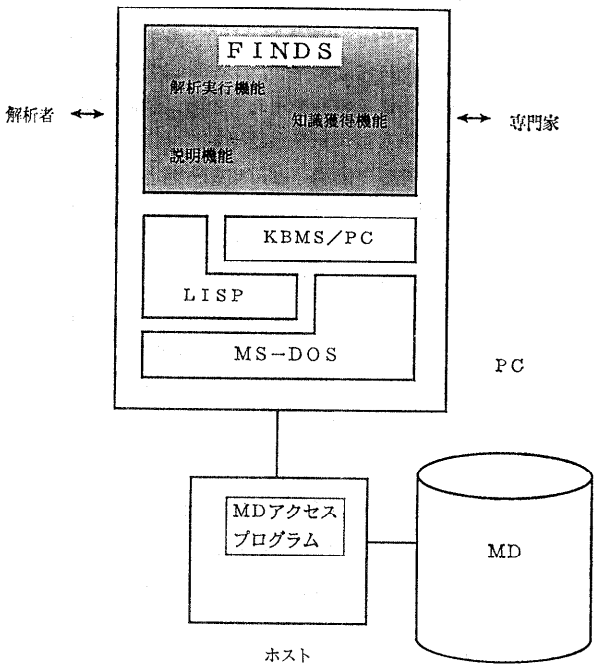


図 4 システム構成
Fig. 4 System configuration.

表 1 FINDS の動作環境
Table 1 FINDS's environment.

項 目	条 件
機 種	NEC PC-9801 (i80286, i80386, または i80286/i80386 相当の CPU を搭載したモデル)
メモリ	プロテクトメモリ 12 MB
ハードディスク	40 MB 以上
通信インタフェース	モデム, 電話回線 (半二重ベーシック, JUST-PC) LAN ボード, LAN (TCP/IP)

報処理システムに対する付加型の ES となっている.

FINDS の動作環境をまとめて**表 1** に示す.

3.2 機能概要

FINDS は次の 3 機能から構成されている.

①知識獲得機能
②診断実行機能
③説明機能

以下, それぞれの機能について説明する.

(1) 知識獲得機能

(a) テンプレート編集機能

制御表の定義情報のうち, 制御表の名称, 全長, 各フィールドについて, 名称, オフセット, 長さ, および属性を仕様書等から抽出し, あらかじめテンプレートとして知識ベースに登録する. テンプレートの登録・修正には, 一括して登録・修正する機能と会話的に登録・修正するエディタ機能とがある. 故障解析時に注目しないフィールドは省略する等, 実際の制御表とは異なる構成でテンプレートを設定することもできる. テンプレートは外部表現と内部表現の両表現で同時に登録される.

事前にテンプレートを投入しておくことにより, 専門家はノウハウ知識の投入に専念できる利点がある.

(b) ビュー編集機能

各ビューは**表 2** に示されるボックスと呼ばれる要素と, ボックス間を結ぶ線とから成るフローチャートで表現される (**図 5**). 画面上ではフローチャートの部分と, その内容を示す式の部分とを分けて別のウィンドウに表示する.

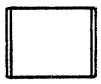
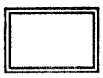
両者はボックスにつけられた番号により対応づけられる. この番号は FINDS が自動的に付与する. ビュー編集機能はフローチャートのエディタであり, 解析の専門家は FINDS が表示するメニューに従って容易に知識を投入・編集することができる. 外部表現で表されたビューは, コンパイル機能により内部表現に変換される.

ビュー名は "制御名-id" の形式をしており, ビューチェーン内で一意になるように id (自然数) が付与される. ビューチェーンはビューの call-called 関係に基づいて自動的に作成される.

(c) 解析知識編集機能

ビューチェーンで表される解析知識は, 自動知識, パス知識, 知識部品の 3 種類に分けられる. 自動知識は特定のダウンコード用の解析知識であり, 解析実行時に MD のダウンコードに対応した自動知識を FINDS が選択する. パス知識は解析実行時に利用者がパス知識名を指定することにより選択される. パス知識が利用されるのは, ①ダウンが原因でない場合, ②ダウンコードの知識が登録されていない場合, ③解析の専門家が自らの経験等でその知識を利用すべきと判断した場合等である. 知識部品は共通的なビューチェーンを部品化したもので, 他の知識から呼ばれる.

表 2 ボックスの種類
Table 2 Sorts of "Boxes."

種別	形状	条件	機能
分岐		真偽値を返す式 真：ＮＩＬ以外 偽：ＮＩＬ	ｉｆ相当 真の場合真下に 偽の場合右下に分岐
多重分岐		値を返す式	ｃａｓｅ相当 ルート別の分岐条件により分岐
処理		単なる処理式	処理（計算，表示等）の実行
ビュー呼出し		値（アドレスに相当）を返す式と呼出し先ビュー名の対	呼出し先ビューに制御移行（ビューの処理終了後，自ビュー内の次のボックスに制御移行）
部品呼出し		値（アドレスに相当）を返す式と呼出し先部品名の対	ｃａｓｅ相当 ルート別の分岐条件により分岐
ジャンプ		移行先ボックス番号	自ビュー内の他のボックスに制御移行
停止		（なし）	診断の強制的な終了

上述のビュー単位の編集機能と同様に，解析知識単位でのコピー，削除，知識内容印刷等ができる．

（d） 関数登録機能

ルーチン間のリンク情報のようなデータベースへのアクセス，収集データの表形式での編集等，通常使われると想定される関数はあらかじめシステム関数として登録されている．これ以外の機能を関数として登録することもできる．この関数は LISP の関数である．関数を利用することにより，テーブルビュー内のボックスの内容を簡単に記述することができる．

（e） ヘルプ機能

ビュー編集中にテンプレート一覧，テンプレートの詳細情報，関数一覧等を表示して，利用することができる．

（2） 診断実行機能

診断実行機能は，MD アクセスプログラムを使って読み出されたデータを知識に従って解析する機能である．MD アクセスプログラムの基本機能には，

①MD アクセスプログラムの起動
②MD ファイルのオープン
③データの送信要求
④MD ファイルのクローズ
⑤MD アクセスプログラムの終了

がある．これらを使って，受信データのフォーマット解析を行うことによりデータがテンプレートインスタンスにバインドされる．①と②は解析を開始する時点で，④と⑤は解析が終了した時点で必要である．③とフォーマット解析はテンプレートインスタンスの作成ごとに必要である．

解析者はまず MD ファイルを特定する情報（ファイル名と格納媒体名の対，パス名等）を指定する．この情報を基に，MD アクセスプログラムを起動し，MD ファイルをオープンする．次に，自動知識とパス知識のどちらを使うかを選択する．自動知識が選択された場合，MD 中のダウンコードに対応した知識を知識ベースから選び，この知識に従って FINDS が自動的に解析を進める．一方，パス知識が選択された場合は，先頭のビューに対応するテーブルのアドレスを解析者が指定すると（先頭がレジスタであればアドレスの指定は不要である），自動知識の場合と同様に知識に従って解析が行われる．

ビューが呼ばれると，アドレスとビューに対応するテンプレートの対が初めてのものであれば，そのアドレスとテンプレートに登録された制御表の長さをパラメータとしてデータの送信を要求する．得られた文字列をフォーマット解析することにより，制御表の実現値を得て，テンプレートインスタンスを作成する．初めての対でなければ，既にあるテンプレートインスタンスを利用する．次に，このテンプレートインスタンスに対して，ビューを構成するフローチャートの先頭のボックスから順に処理を実行する．フローチャートの葉に相当するボックスの処理が終了すると，そのビューの呼び元に制御を戻す．このようにして，解析

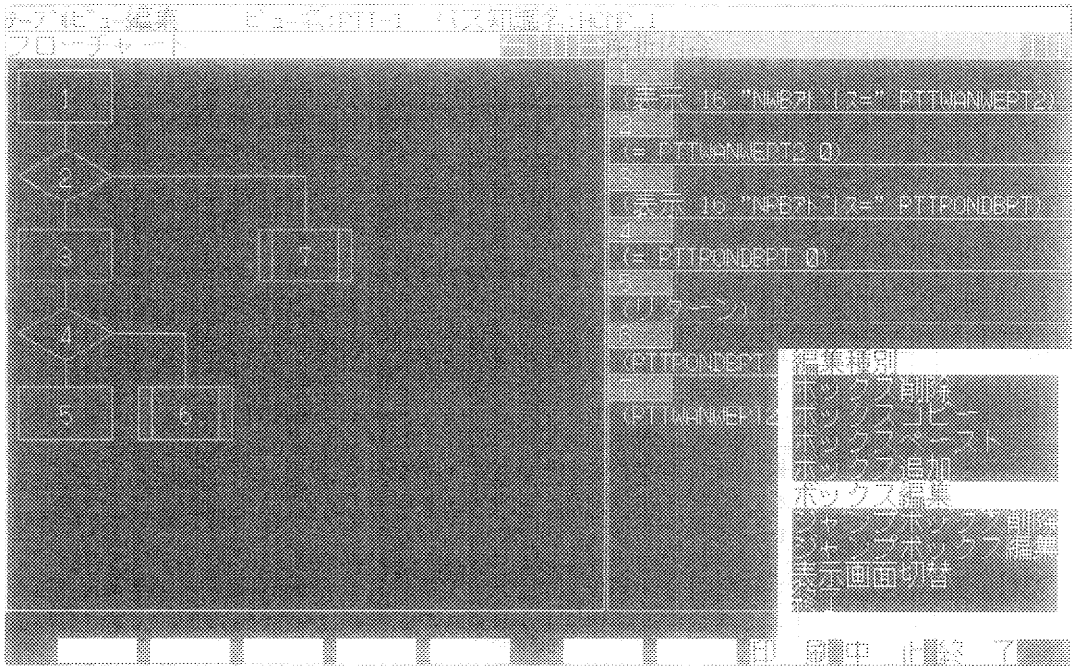


図 5 ビュー編集機能
Fig. 5 "View" editing facility.

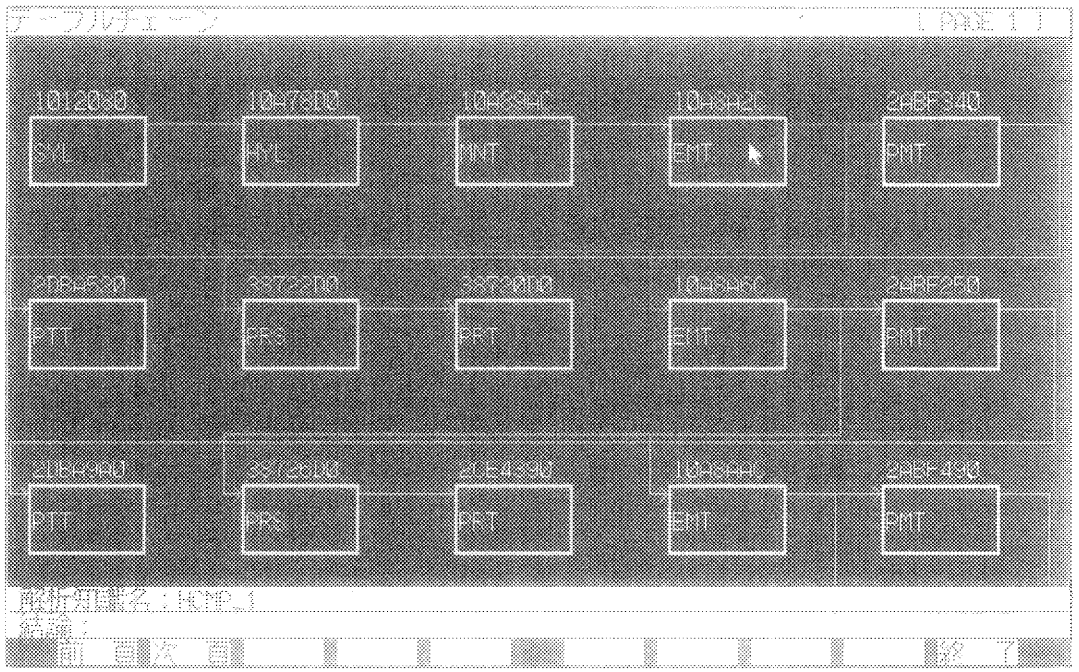


図 6 説明機能（制御表のつながり）
Fig. 6 Explanation facility (tables chained).

が進められる．なお，知識部品は単独では解析に使うことはできない．

解析の途中経過は PC のディスプレイ上に表示される．この表示内容はファイルに保存することができる．解析終了後に，市販のエディタ等で内容を参照することができるので，解析中は利用者は付き切りになる必要がない．この機能は，特に大量のデータを整理して出力するような知識では，非常に大きな省力化の効果が得られる．

（3） 説明機能

診断実行直後に診断の過程である制御表のつながり（**図 6**）をビューとテンプレートインスタンスを用いて表示する．この図は，従来故障解析の経過として障害処理票と呼ばれる帳票に記入されていたものと同様である．この画面上にある制御表をマウスで指定すると，その制御表のフィールドのうち，解析中に参照された値が表示される．また，診断で使用されたビューのフローチャート上で診断時に使われたルートを強調表示することができる．

診断結果をファイルに保存しておくと，上記と同様の表示が可能である．この機能は，過去の診断結果を参考にする場合に利用される．

4. 具 体 例

特に大規模システムのソフト故障解析では，連続的にポインタでつながっている同種の制御表を順次解析することが多い．このような知識は，先頭アドレスを変え，再帰的に同一ビューを呼び出すことにより実現できる．以下，具体例で説明する．

4.1 知 識

以下に示すような知識があり，制御表 STACK は**図 7** に示すようにたどることができるものとする．

【知識】STKCHK 1 が文字 “B” か “C” であって，STKCHK 2 が文字列 “ST” 以外である制御表 STACK を探す．このような STACK が存在すれば STKPRG というルーチンの使用法誤りである．この知識は**図 8** に示すビュー REG-1, TASK-1, STACK-1 からなるビューチェーンで実現されるパス知識である．

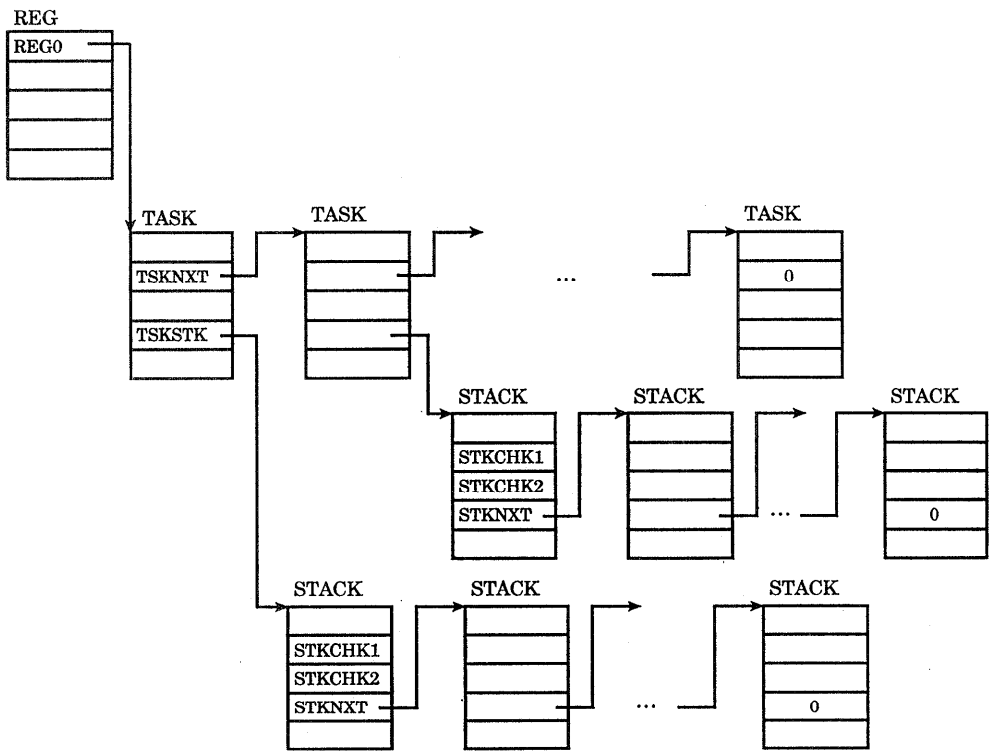


図 7 制御表のつながり
Fig. 7 Table chain.

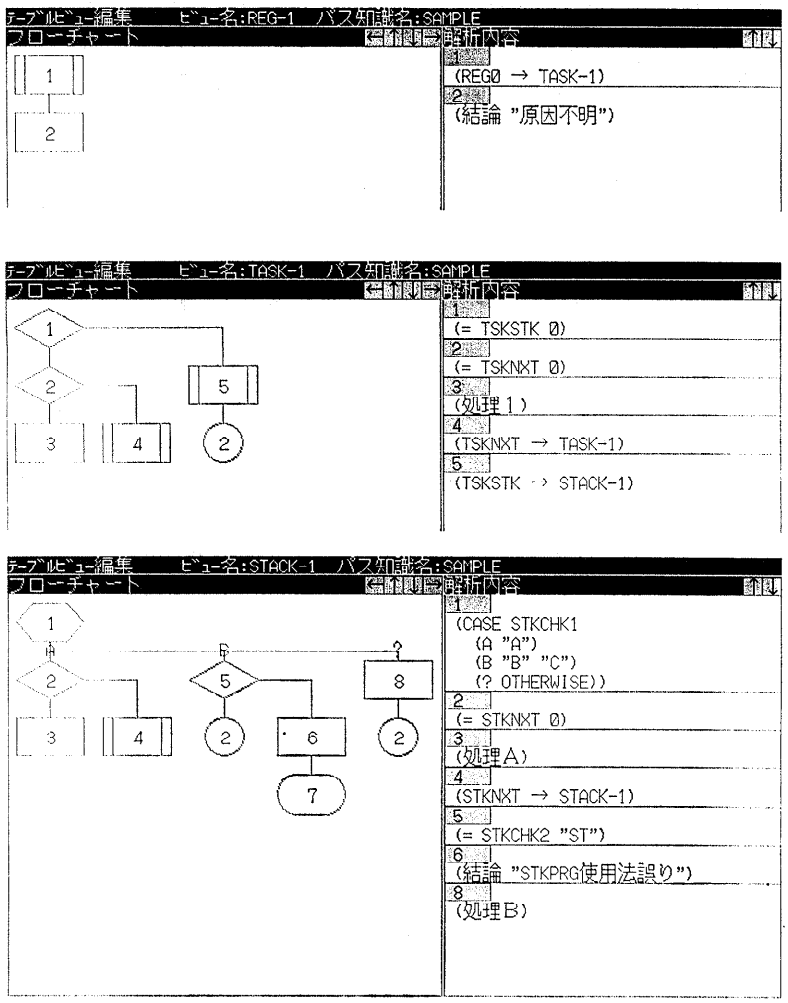


図 8 知識の具体例
Fig. 8 Sample knowledge.

4.2 動作説明

この知識を指定して MD の解析を行うと，次のように動作する．

①アドレス指定なしで REG-1 が呼ばれると，レジスタ情報を MD から得て，REG のテンプレートインスタンスを作成し，このテンプレートインスタンスに対して REG-1 を適用する (REG-1 の先頭である1のボックスに制御を移行する)．

②REG-1 の1のボックスでは REG 0 の内容を TASK の先頭アドレスとして TASK の長さ分のデータを MD から得て，テンプレートインスタンスを作成し，これに対して TASK-1 を適用する．

③TASK-1 では，TASK に STACK がつながっていなければ (TSKSTK が0のとき)，次の TASK を調べる (次の TASK のアドレス TSKNXT を TASK の先頭アドレスとして別のテンプレートインスタンスを作成し，これに対し TASK-1 を適用する)．次の TASK がなければ (TSKNXT が0のとき)，呼び元の TASK-1 に制御が戻る．呼び元では4のボックスが終了したことになるので，さらに1つ前の呼び元に戻る．このようにして，最終的には最初の TASK-1 の呼び元である REG-1 に制御が戻り，REG-1 の2のボックスが

実行される．一方，TASK に STACK がつながっていれば，最初の STACK を調べる（TSKSTK を STACK の先頭アドレスとして STACK のテンプレートインスタンスを作成し，これに対し STACK-1 を適用する）．

④STACK-1 では，その STACK が STKPRG 使用法誤りの条件を満たせば，結論を出力してすべての処理を終了する．条件を満たさない場合には，次の STACK を調べる（STKNXT を STACK の先頭アドレスとして STACK のテンプレートインスタンスを作成し，これに対して STACK-1 を適用する）．最後の STACK であれば（STKNXT が 0 のとき），呼び元の STACK-1 に制御が戻る．呼び元の STACK-1 では 4 のボックスが終了したことになり，さらにその呼び元の STACK-1 に制御が戻る．以下，順次制御が戻り，最終的には TASK-1 の 5 のボックスが終了したことになる．次に，TASK-1 では 2 の判断ボックスに制御が移り，同様にして次の TASK を調べる．

5. 評　　価

5.1 汎 用 性

2.2 節で述べたように，MD アクセスプログラムとのインタフェースは MD アクセスプログラムごとに作成する必要があるが，その規模は全体（LISP で約 40 K行）の約 1% である．現在までに FINDS を用いて，DIPS，POPS，DIPS 用の通信制御装置である ICA[11]，および，電子交換機 D 70[12] 用の MD ファイルを用いたソフト故障診断用の ES が構築できることを確認している．なお，DIPS，POPS，ICA，D 70 はそれぞれ別のアーキテクチャであり，MD ファイルの形式，および，MD アクセスプログラムも全く異なっている．

以上からわかるように，FINDS は計算機の機種に対する汎用性の高い構成となっている．

5.2 FINDS/DIPS

DIPS の基本ソフトウェアの故障解析を行っている部門において，DIPS 用の故障診断 ES である FINDS/DIPS を用いて評価試験を行い，良好な結果が得られたことから，現在，本格的に故障解析に利用されている．以下，評価結果について具体的に述べる．

（1） 知識作成

約 100 種類の知識を整理・投入したボックスの総数は約 13,000 である．知識投入は，上記部門の解析の専門家が行い，操作方法の簡単な説明以外には KE の関与はほとんど必要としなかった．

（2） ES としての効果

FINDS/DIPS を用いて，DIPS の故障解析に利用した結果，解析時間が約 2/3 に短縮された．知識の拡充により，さらに時間短縮が期待される．

メモリ使用状況を調べる知識のように，MD から収集したデータを表形式に整理するような人手で行うと手間のかかる繰り返しを必要とする解析には特に有効であった．

5.3 FINDS/POPS

POPS のメモリの使用状況を調べる C のプログラム（約 4 K ステップ）と同等の知識を FINDS/POPS に投入した．その結果，知識投入とデバグに要する時間は C のプログラム作成とデバグに要する時間の約 1/10 であった．また，1 ボックスは C のプログラムの約 10 ステップに相当した．

6. おわりに

ソフトウェアの故障解析の知識が制御表の構造知識，制御表の見方，制御表間の関係の 3 種類の知識に分割できることに着目し，知識獲得の容易なソフト故障診断用のドメインモデルを考案した．具体的には，制御表の構造知識をテンプレートとしてあらかじめ知識ベースに格納しておき，専門家のノウハウ知識である制御表間の関係と制御表の見方をビューとし，その外部表現であるフローチャートの形で投入できる知識エディタを実現した．これにより，専門家は KE の協力も KBMS の知識もなしに自ら知識を投入し，維持管理することが可能となった．さらに，外部表現である表とフローチャートの形式で行う説明のためのデータは，投入された知識から自動的に作成することと合わせ，専門家はノウハウ知識の投入だけに専念できる．FINDS への知識投入は，同等の機能を C 言語で作成する場合と比較すると約 10 倍の生産性が得られる．

複数のアーキテクチャの大規模な MD 解析に FINDS を適用でき，さらに，既存の情報処理システムに全く影響を及ぼさずに情報システムと結合して付加型の ES を構築することができる．

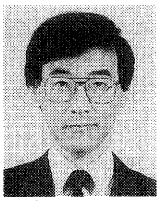
現在，FINDS は社内において DIPS，ICA，D 70 の大型 OS の故障診断に利用されており，さらに知識を拡充中である．

謝辞 FINDS プロジェクトの推進に当たり，助言いただいた神奈川大学村上教授，NTT ソフトウェア(株)吉田グループ長に深く感謝いたします．また，知識提供，評価試験に協力いただいた基本アーキテクチャ研究部，網オペレーション研究部，データベース研究部，ネットワークシステム開発センタ，情報システム本部，通信ソフトウェア本部，NTT ソフトウェア(株)の関係各位に深く感謝いたします．

参考文献

1) 藤原，中安：DISP 情報処理ネットワークの遠隔運転保守技術，*NTT R&D*, Vol. 39, No. 7, pp. 1083-1094 (1990).
2) Digital Equipment Corporation: VAX/VMS System Dump Analyzer Reference Manual (1985).
3) Latham, B. and Swartwount, M. W.: CDx-Crash Diagnostician for VMS, *Expert Systems and Knowledge Engineering*, pp. 199-206 (1985).
4) 竹内，大高：高度分散処理系実現に向けた DIPS 106-200 S の実用化，*NTT R&D*, Vol. 39, No. 5, pp. 737-746 (1990).
5) 飯田，立花：ソフトウェア故障診断エキスパートシステム，*NTT R&D*, Vol. 39, No. 3, pp. 403-410 (1990).
6) 飯田：ソフトウェア故障診断ドメインシェル，1989 年電子情報通信学会秋季全国大会，D-148 (1989).
7) 大南，大町，畠中，平山：POPS-100 (R 1) の概要，*NTT R&D*, Vol. 38, No. 9, pp. 957-964 (1989).
8) Brinch Hansen, P.: *Operating System Principles*, p. 366, Prentice-Hall, Englewood Cliffs (1973).
9) 飯田：KBMS における ES 開発支援機能—フローチャート形式での知識入力機能，第 39 回情報処理学会全国大会論文集，3 B-9 (1989).
10) 服部，森原，古屋，村山：KBMS におけるエキスパートシステム開発支援方式，*NTT R&D*, Vol. 39, No. 3, pp. 393-402 (1990).
11) 菊池，東海林，山下，宮保：DIPS-CCP における INS ネット接続方式，*NTT R&D*, Vol. 38, No. 8, pp. 895-905 (1989).
12) 秋野：サービスの総合化を目指して—最近のディジタル交換技術の動向—，NTT 施設，Vol. 38, No. 1, pp. 13-20 (1986).

(平成 5 年 2 月 16 日受付)
(平成 5 年 6 月 17 日採録)

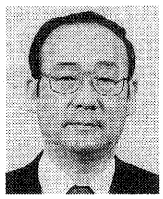


飯田 敏幸（正会員）

昭和 26 年生．昭和 49 年東京大学工学部計数工学科卒業．昭和 51 年同大学院修士課程修了．同年日本電信電話公社（現 NTT）入社．現在，情報通信網研究所知識処理研究部主幹研究員．分散データベースシステム，人工知能などの研究・開発に従事．電子情報通信学会，人工知能学会各会員．

島田 茂夫（正会員）

昭和 29 年生．昭和 53 年大阪大学工学部通信工学科卒業．同年日本電信電話公社（現 NTT）入社．現在，情報通信網研究所知識処理研究部主任研究員．VLSI プロセッサ，高並列処理，人工知能の研究・開発に従事．電子情報通信学会，人工知能学会各会員．



河岡 司（正会員）

昭和 43 年大阪大学工学部通信工学科卒業．昭和 45 年同大学院修士課程修了．同年日本電信電話公社（現 NTT）入社．データ通信研究部データ通信網研究室長，研究開発技術本部運営施策部長，情報通信網研究所知識処理研究部長を経て，現在，コミュニーケーション科学研究所長．データ通信システムＯＳ，コンピュータネットワーク，分散処理システム，人工知能の研究・開発に従事．工学博士．電子情報通信学会，人工知能学会，IEEE 各会員．