ソフトウェアデバッグエキスパートシステムのための知識エディタ

5D－4

鈴木宏文　今井恵一*　土田賢省*

日本電気技術情報システム開発（株）　*日本電気（株）

1　はじめに

我々は交換系ソフトウェアを対象としたデバッグエキスパートシステム(DBES)を開発した。([1][2]) DBESは交換系ソフトウェアを試験する工程（試験工程）で見つかった障害やエラーに対する診断、デバッグをユーザと対話的に進めていくシステムである。図1はDBESのシステム構成である。DBESは図に示すように5つのコンポーネント（ユーザインタフェース、推論機能、知識編集機能、知識ベース、補助機能）から構成される。

システムの評価結果から、専門家自身が容易に知識の保守（入力、追加、編集）を行えることが必要であることが明かになった。この要求に応えるために、グラフベースの知識エディタを開発した。以下、知識エディタについて説明する。

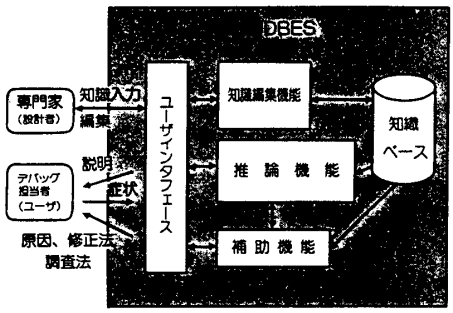

図1 システム構成

2　デバッグの知識

2.1　問題点

デバッグ知識の性質から予め完全な知識を蓄積しておくことは不可能である。交換ソフトのデバッグでは、開発対象システム、リアルタイムシステム、OS、ハードウェアなど非常に広範な領域の知識が要求される。さらに交換ソフト自身が大規模で複雑であるため、デバッグの知識も膨大になる。このような知識を全て最初に網羅することは実際的ではない。また、専門家の持つデバッグの知識は顕在化されてない部分が多い。それらはデバッグ時に実際のエラー症状に遭遇した時に初めて想起されることがある。さらにこの時に専門家自身の頭の中で知識の再整理（修正・追加）なども行なわれる。 以上のことから、最初に核となる知識を入れ、デバッグにおいて随時、知識の追加・修正を行ない、知識を蓄積するという形態が要求される。この形態ではKEが介在し知識を保守することは現実的でない。このような理由でKEを介することなく専門家自身が容易に知識の編集を行ないたいという要求が出てきた。この要求を満たすためには、以下の問題点を解決する必要がある。

(1) 知識表現

専門家にとってルール形式よりもDAG(Directed Acyclic Graph)形式の方が自然である。([3]) 知識獲得作業において、我々はまずルール形式のフォームによる専門家へのインタビューを試みたが、その方法では全く知識を獲得することはできなかった。これは専門家自身の知識（頭の中で整理されている体系）がルール形式でなかったためと思われる。最終的にグループディスカッション方式によりDAG形式の知識を専門家から得ることができた。それらは信頼性の高い（複数の専門家による合意の下の）知識であり、かつ効率的にある程度満足のいく量が得られた。

(2) 整理方法

知識の整理方法には個人差があり、また同じ個人でも整理の段階や状況により異なってくる。例えば、個々の症状に対して、他の症状と合わせてグループ化したり、細分化していく作業をある時はボトムアップに、またある時はトップダウンに整理する。

(3) 知識の複雑さ

知識編集では深くて膨大な知識を取り扱う必要がある。実際にデバッグで有用な知識は対象システムの細部に関する情報を持つので、DAG形式で表現するとレベルが深いものになる。また、実用のためには、対象ソフト全てをカバーする必要があり、知識全体として大規模なものになる。

2.2　対応するアプローチ

我々は、知識の保守に関しては専門家自身が容易にデバッグ知識を編集することができる知識エディタの開発を行なうことにした。専門家自身が使いやすい知識エディタの実現のために複数の専門家に机上で知識の整理のシミュレーションを行なってもらい、知識エディタの備えるべく要件をインタビューして切りだした。その際、KE側からは考えられるだけのアイデアを出し、専門家側か

[0]Knowledge Editor for Software Debug Expert System, Hirofumi Suzuki, Keiichi Imai*, Kensei Tsuchida*, NEC Scientific Information System Development Corporation, *NEC Corporation.

ら必要なもの不必要なものを指摘してもらい次第に洗練していく方法を採った。このようにして開発された知識エディタは簡単な知識表現形式をグラフィカルな操作で編集でき、専門家自身がDBESで推論（デバッグ、診断）中でも入力・修正が可能である。そして本エディタは、上記の問題点'1)'に対してはデバッグの知識をDAG形式そのままのイメージで入力編集できるDAG編集機能、'2)'に対しては（DAG形式の）知識の入力順などに制限を設けず様々な方法で整理できる機能とシンタックスチェック機能、そして'3)'に対しては知識全体を分かりやすく表示する清書機能でそれぞれ対処している。

3 知識エディタ

3.1 操作対象の知識

知識エディタの操作対象となる知識について述べる。形式的には知識は質問、選択肢、結論の3つのノードのタイプを持つDAGとする。各ノードには実体としてテキストを設定できる。これらは症状（symptom）と原因（cause）を結びつけたルールの集合に変換される。DBESではこのルールベースに基づいて診断（推論）を行なう。DAG形式の知識における質問ノード（Qノード）と選択肢ノード（Sノード）組が1つの症状に対応する。また結論はDAG形式ではリーフにあたる。意味的には、Qノードと複数のSノードからなる部分木がテスト方法を表す。デバッグの戦略は、階層構造のコンテキストとして埋め込まれている。つまりその構造に沿ってテストしていくことが効率的なエラー原因追求につながると考えられる。

3.2 主な機能

以下では本エディタの特徴的機能について述べる。

(1)DAG編集機能

DAG形式で表される知識構造を自由に編集する機能として、ノードの編集（生成、削除、移動、変更）機能、アーク（ノード間の関係）の編集（生成、削除、張変え）機能、ノードのテキスト編集機能、部分木（グループ）の編集（削除、移動）機能などを備えている。

(2) 知識構造のシンタックスチェック機能

専門家の知識整理に柔軟に対応するために、知識構造（DAG）編集には制約がない（いかなるDAG構造も作成可能）のでシンタックスに沿わない知識が入力される可能性がある。これに対しては知識の整合性チェックを行なう機能を備えている。

(3) 清書機能

知識構造全体を様々な見やすい方法で自動的に表示する機能として、部分木を6種類の方法で縦型／横型に表示する機能、部分木を1ノードに圧縮したり、その逆を行なう（部分木圧縮／解放）機能、指定したパス以外の部分木を圧縮する（パスフォーカス／解放）機能、ノードのテキスト表示長を変更する機能を備えた。これらの機能は大量の知識を整理する時に役立つと思われる。

以上のような機能は全て（テキスト編集以外）マウスのみで操作可能とし、これにより専門家でも容易に知識編集が行なえる。また、入力した知識を効率的に確認するためのブラウジング機能（表示の拡大／縮小、全構造表示機能）や、専門家が知識整理の過程でメモ用紙がわりに使用する補足情報編集機能も備えている。

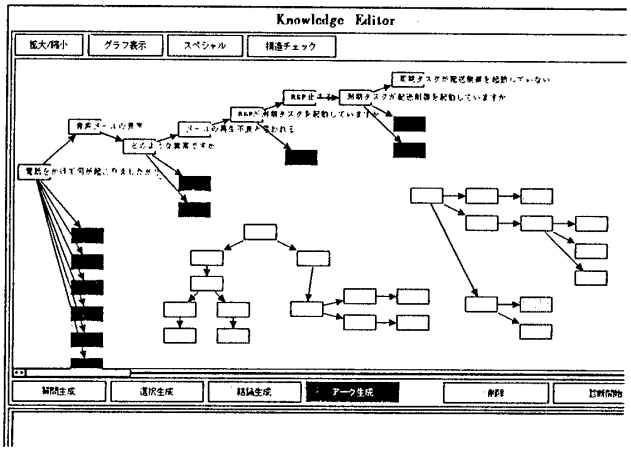

図2 知識エディタ

4 おわりに

本エディタは「ユーザインタフェース構築環境: 鼎」([4])を利用して構築した。本エディタを組み込んだDBESは現在EWS4800(UNIX SYSTEM V)のXウィンドウ(X11R2)上で実行可能である。

謝辞

研究開発にあたり、御指導下さった日本電気(株)ソフトウェア生産技術開発本部川越課長に感謝いたします。

[参考文献]

[1] 赤尾, 今井, 土田：ソフトウェアデバッグエキスパートシステム, 情処学会第38回全国大会, pp.553-554, 1989.

[2] C.Y.Akao, K.Imai and K.Tsuchida:Debug Expert System for Switching Systems Software, NEC R&D, No.95, pp.107-115, Oct. 1989.

[3] 土田, 赤尾, 今井：交換ソフトの領域モデルに基づくデバッグエキスパートシステムの開発, 人工知能学会誌, Vol.5, No.2, 3月号(1990)掲載予定.

[4] 暦本、他: Xウィンドウ上のマルチメディアユーザインタフェース構築環境 - 鼎、情処学会、第30回プログラムシンポジウム、pp.105-115(1989.1).