情報処理学会第29回(昭和59年後期)全国大会

サービスベースシステムにおける論理型言語向きサービス記述

6H-7

荻野　正　　深沢　友雄　　田中　英彦　　元岡　達

(東京大学　工学部)

1.　はじめに

計算機網中に分散して存在するサービス(各種のプログラム、データ等)をユーザに提供する際に、

・ユーザに対しては、サービスの分散性を意識させない。

・計算機網中の各計算機上では独立にサービスの拡張が行なえる。

等を目標として開発しているシステムがサービスベースシステム(SBS)である([1])。現在、サービスの要求言語として論理型言語を用いたSBSについて検討を行なっている([2])。

本稿では、Prolog を用いてサービスを記述する方法について発表する。

2.　SBSのサービス記述

SBSでは各計算機に於いて、サービスを次の3つのレベルに分けて記述する(図1)。

①内部ビュー

②概念ビュー

③外部ビュー

上記の三層のビューのうち、内部ビューは、各計算機が独立にサポートするサービスのビューであり、外部ビューは、ユーザや、他計算機に対して分散性を意識させないビューである。このギャップを埋めるのが概念ビューであり、このレベルの記述に基いて各ノードの処理系は、分散しているサービスを処理する。即ち、分散性を統合する時に中心となるビューが概念ビューである。

次にビューの記述について考えてみる。まず、SBSにおいてサービスは次のように分類できる。

(1)自計算機の組み込み述語、及びProlog のプログラム

(2)自計算機の実行可能モジュール(OSが実行する)

(3)他計算機の組み込み述語、Prolog のプログラム、及び実行可能モジュール

現実装では、それぞれのサービスに対して、概念ビューとして、次のような情報を記述してある。

(1)の場合　処理系が知っているので記述していない。

(2)の場合　自計算機の内部ビューへの写像情報

(3)の場合　サービスを要求するSBS(バックエンドSBS、BES)名と外部ビューへの写像情報

自計算機の内部ビューや、BESの外部ビューへの写像情報とは、

1．それぞれのビュー上での名前

2．入出力データの型

等である。

これらの情報を、SBSではデータベースの関係モデルを用いて記述、管理するものとする。

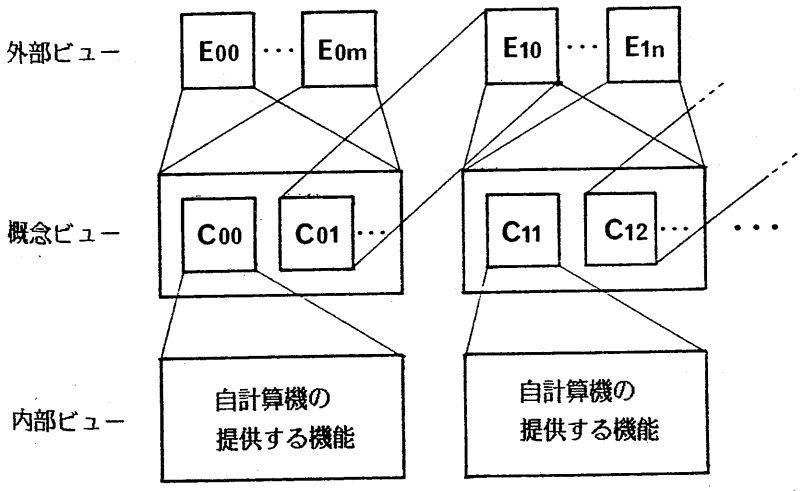

図1　SBSの三層ビュー

3.　Prolog 向きSBSのサービス記述

現在、VAX-11/ 730, VAX-11/ 780,　M-280Hの三台の計算機を用いて実験システムを構築中である(図2)。Prolog の処理系としては、エジンバラ大学で開発されたC-Prolog を使用している。

3.1　内部ビューの記述

内部ビューは、各ノードに存在する、データ、ロードモジュール、コマンド等の記述であり、各ノードのOS(UNIX、VOS3)が独立に定義、管理する。

3.2 概念ビューの記述

概念ビューの記述はサービス名（述語名+引数の数）をキーとするリレーションとして表すことにする。但し、今のところリレーションのデータを直接扱う事はできないので、

service （サービス名、属性名、属性値）.

という clause を assert する事によりリレーションを表すものとする。

現在、属性としては、

place …ＢＥＳ名（自計算機の実行可能モジュールの場合はint ）。

map …内部ビューまたは外部ビュー上の名前。

out …画面に出力される時この値はstdoutになる。

arg-type…入出力データの型。

等を考えている。

概念ビュー上でサービスを定義する為に以下の様な述語を作成した。

（１）自計算機の実行可能モジュールの定義

assert_int （サービス名、
内部ビュー上での名前、
属性名（属性値）のリスト）.

（２）他計算機のサービスの定義

assert_ext （サービス名、
ＢＥＳの外部ビュー上での名前、
ＢＥＳ名、
属性名（属性値）のリスト）.

サービスの定義例

flist （引数１つ、画面に出力がでる）というＯＳのコマンドを、概念ビュー上でflという名前で定義する場合

assert_int （fl（X）,
flist （X）,
[out （stdout）]）.

を実行すると、

service （fl（X）, place , int ）.
service （fl（X）, map , flist （X））.
service （fl（X）, out , stdout）.

という３つの clause が assert される。

3.3 外部ビューの記述

外部ビューは、概念ビューの部分集合をユーザあるいは、ノード対応に提供する。Prolog では、あるゴールをある制限された範囲の定義を用いて解くモデル（多世界モデル）を導入することにより、外部ビューを形成できる。このモデルの詳細については、現在検討中である。

4. 検討

今回、考察したサービスの記述方法は、すでにＳＢＳが稼動している場合である。ＳＢＳを立ち上げる時の概念ビューの形成については、特に考慮していない。

ＳＢＳのビューの二次記憶上での管理は、各計算機のローカルなデータベース機能によって実現する。従って、本稿で述べたビューの、ローカルなデータベースでの表現を検討する必要がある。

5. 今後の課題

今後は、データも含めたサービスの記述の詳細の検討とシステムの構築、及びアプリケーションの実装等を行なう予定である。

残された課題としては、

（１）サービスの並列化

（２）サービスの管理機構

等に関する検討が考えられる。

6. 参考文献

［１］深沢、田中、元岡、「サービスベースシステムの概念と基本構成」、信学会、電子計算機研究会、昭和５７年１０月、pp. 53-62

［２］深沢、荻野、田中、元岡、「論理型言語向きサービスベースシステムの構成」、本大会、 6H-6

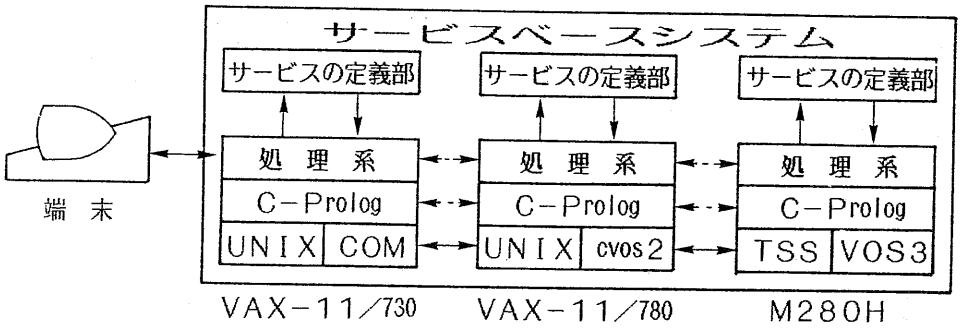

図２
実験システムの構成

4L-9

Intelligent Manual System に関する一考察

青柳 龍也　田中 英彦　元岡 達

(東京大学 工学部)

マンーマシンインタフェースを向上させるために音声認識、自然言語理解等、様々な方策が考えられるが、従来からあるオンラインマニュアルやヘルプの機能を改善することによってもコンピュータを使いやすくすることができる。現在我々は、そのような目的の知的なマニュアルシステムを開発している。本稿では、そのシステムの構想について述べる。

1. 背景

人がマニュアルを見るのは次のようなときである。

1) コンピュータシステムを使って行ないたい仕事があるが、コマンドがわからない。

2) コンピュータシステムにどんな仕事ができるのか知らない。

1)のような状況は、たとえコンピュータのエキスパートであっても、多種多様なソフトウエアを使う際には生じる状況である。また、2)のような状況は、そのコンピュータシステムの初心者には当然起こりうる。コンピュータシステムの機能が増えればマニュアルも増え、捜すことや読むことが困難になる。そこで、1)ではユーザの質問に的確に答え、2)では積極的にユーザに、そのコンピュータシステムで実行可能なことを教えていくマニュアルシステムが必要である。

2. 目標

当マニュアルシステムの最終的な目標は、そのコンピータシステムについてよく知っている人が常に隣りにいるような環境を作ることである。よく知っている人が隣りにいれば、わからないことには適切な解答が得られ、もっといいやり方があるようなときは適宜それを教えてくれるだろう。そのコンピュータシステムの使用経験がなかったり、使用頻度の低いような人でも、準備期間なしに(単に練習のためだけに実際の使用目的と異なることをしたりせずに)、教えられながら使っていくことができるようなシステムを理想とする。

大きくわけて二つの機能を実現する。

1) help－聞かれたことに的確に答える。

そのためには、ユーザが何を聞きたいのかはっきりわからなければならない。しかし、ユーザの質問には情報が不足していたり、ユーザ自身も聞きたいことがはっきりしていないことも多い。前者についてはユーザのモデルを持つことが必要であり、後者についてはユーザとの対話によって聞きたいことをはっきりさせる必要がある。

2) instruction －教えるべきときに教える。

人は自分の知っている他のコンピュータシステムの機能と類似する機能については、ヘルプを使って聞いてくるが、まるで別種の機能については教えられない限りそのような機能が存在するだろうと推論したりしない。たとえば、マイクロコンピュータのBASICのエディタのように、文字列のサーチ機能のないエディタを使っている人は、他のエディタを使うときも、サーチ機能があると教えられるまでは、そのような機能をヘルプで聞いたりしない。そこで、準備期間なしにコンピュータシステムを使えるようにするためには、その機能を使うべき状態になったときに適宜教えていくことが必要である。そのためには、コンピュータシステムの状態のモニタリングとユーザのモデルから、教えるべきタイミングと教えるべきことを決めなければならない。

3. 構成

図のような構成を考える。

・推論モジュール

ユーザの質問に対し答えるべきことを、ユーザのモデルとマニュアルの記述などから決めたり、コンピュータシステムの状態、ユーザのモデル、教育戦略等から、教えるべきタイミングと内容を決めたりするモジュール。

・ユーザのモデル

ユーザはどんなことを知っているのか、どんなことをこれまでに教えたか等に関する情報。

・マニュアルの記述

そのコンピュータシステムの持つ機能に関する記述。

・コンピュータシステムの状態

ユーザが使用しているコンピュータシステムの状態をモニタリングして、それを推論モジュールに理解できるように、記号化する必要がある。

・教育戦略
どのような状態になったら教えるべきか、どのような順で教えるべきか等に関する記述。
・ヒューリスティックス
推論する際のヒューリスティックスがあれば、それを記述しておく。
・インターフェイス
ユーザとのインターフェイスの部分。できれば自然言語によるインターフェイスが望ましい。

4. 研究の方針

現在エディタに関するマニュアルシステムの設計を進めている。エディタを対象として選んだ理由は、1の背景で述べた二つの状況が顕著にあらわれると考えられるからである。

1) どんなエディタでも基本的動作は同じなので、やりたいことははっきりしているがコマンドがわからないという状況が生じやすい。

2) 多くのエディタはそれぞれ固有の便利な機能を持っているので、その便利な機能を知らないという状況が生じやすい。

5. 議論

▷自然言語でコンピュータシステムと対話できれば、マニュアルやマニュアルシステムは不要になるのではないかとも考えられるが、たとえば、エディタに対して「カーソルを1つ右に動かす。」と毎回声に出して言うのと、キーを一つ押すのとでは、比較にならないほど労力の差がある。結局、自然言語のインタフェースが可能になってもコマンドというものはなくならないだろう。コマンドがなくならないなら、コマンドとその機能の間の対応（つまり、マニュアル）は必要である。

▷エキスパートにとっては、このようなマニュアルシステムは不要なのだろうか。確かにエキスパートにとっては不要である。しかし、今後コンピュータシステムの機能が増えれば増えるほど、そのコンピュータシステムのすべてのことを知っているような人はいなくなるだろう。たとえば、用途別にエディタが10個あったとしよう。あるエディタを使っていれば、他のエディタのコマンドは忘れていくものなので、それらすべてのエディタについてエキスパートであり続けることは不可能である。従って、マニュアルシステムは不可欠である。

▷それならば10個あるエディタのコマンドを統一できないのか。同じような機能を持つものなら、たとえば、エディタ同志のコマンドならば統一できる。しかし、機能の異なるソフトウェアのコマンドを統一することは不可能である。OSとエディタとデバッガのコマンドを統一することはできない。結局、多種多様なコマンドについてマニュアルシステムが必要となる。

▷現在あるようなマニュアルはソフトウェア等の実体とは独立に作成されるが、マニュアルシステムではそれが難しい。コンピュータシステムの状態とマニュアルシステムの動作が密接に結びついているので、独立に作るわけにはいかない。この点は今後の課題である。

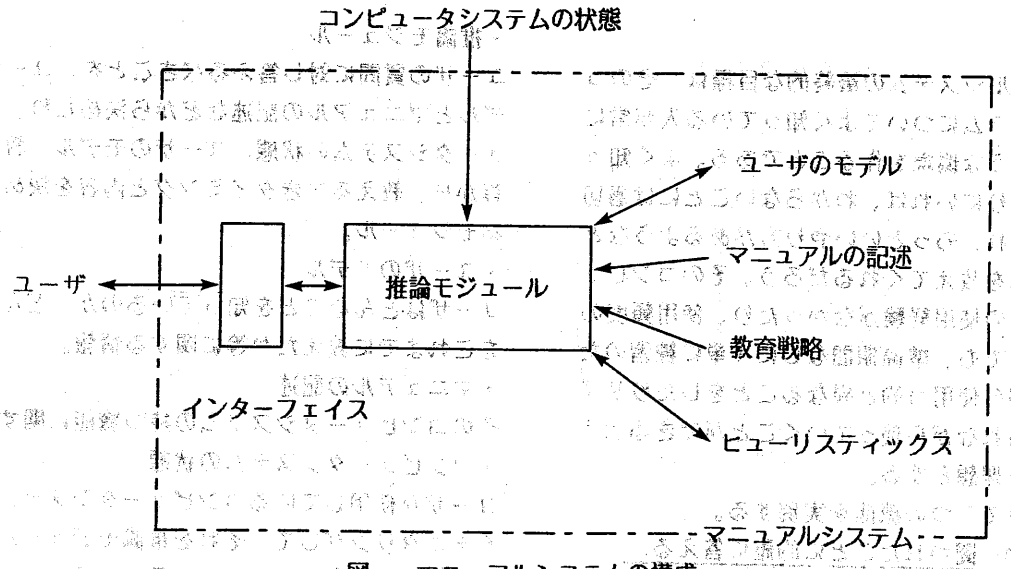

図. マニュアルシステムの構成