システムソフトウェアと
オペレーティング・システム 68－5
（１９９５．３．２４）

日本語を基盤とする情報システムの設計開発

岡本 東 *　樋地正浩 *　布川博士 **　宮崎正俊 *

* 東北大学大学院情報科学研究科　** 宮城教育大学理科教育研究施設

本稿では、誰もが簡単に理解でき、利用が容易な情報システムを構築するために、オペレーティングシステムやその上で動く各アプリケーションの記述、及びユーザインタフェースに自然言語(日本語)を用いることを提案する。また、我々は、それらの実現のために、日本語オペレーティングシステム、日本語処理系、日本語データベースなどの設計および開発を進めており、今回、その一部である日本語に近い表現でシステムを記述可能なプログラミング言語を試作したので報告する。この言語を用いてシステムを記述することにより、システム全体を理解することが容易になり、教育への応用にも効果的と考えられる。

Design and Development of Information System Based on Japanese Language

Azuma Okamoto *　Masahiro Hiji *
Hiroshi Nunokawa **　Masatoshi Miyazaki *

***Graduate School of Information Sciences, Tohoku University**
****Institute for Science Education, Miyagi University of Education**

We propose a new programming language and an information system based on Japanese as natural language. We intend to describe this information system using our programming language . This system consists of “user interface” , “Japanese Operating System” , “Japanese Database System” and others based on Japanese . And this system is easy to understand to use and effective in educational applications . In this paper , we explain the concept of our system and specifications of proposed programming language .

1　まえがき

誰もが簡単に利用可能な情報システムを構築しようとする際、そのための工夫は多数考えられる。

たとえば、利用面で特に重要なユーザインタフェースにおいて、物理的な入出力デバイスを例にあげると、コンピュータに慣れ親しんでいない人に使いやすいようにするためには、多くのキーがついたキーボードよりはマウスやタッチパネルのような操作項目の少ないもの、というようにできるだけ単純なものにするという手法が用いられる。

また、ソフトウェア面においても、必要とするもの以外の機能を包み隠してしまうような手法を用いると、必要な機能を見つけやすいなどの利点があり、初心者にも使いやすく、特定用途のシステムではとくに有効である (図 1)。

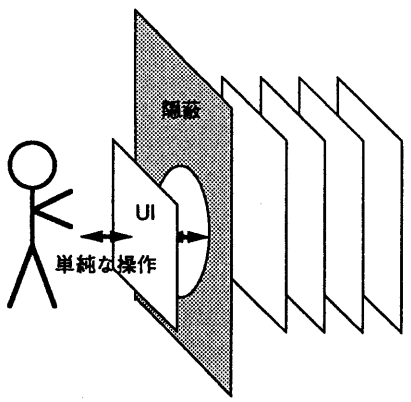

図 1. 初心者の利用を考慮したシステム

しかしながら、ハードウェアにしてもソフトウェアにしても、このように用途を限定し単純化する手法だけでは、用意された機能でカバーしきれない要求が出てきたときに対応できない。

利用する機能やその操作方法を利用者に分かり易く伝え、何か利用上の障害が発生したときにその障害への対応が類推できる方法として、グラフィカルユーザインタフェースにおいて、メタファを利用したユーザインタフェースがある。このユーザインタフェースでは、利用する機能や操作方法、障害時の対応を現実世界との類推に基づき利用者が理解できる [1][2]。しかし、あらかじめ提供された機能や操作以外の要求があった場合には依然としてそのような要求に対応することはできない。

そのため、利用する機能の概要、その機能を利用する操作方法が利用者に理解し易く、用意された機能でカバーできない要求が発生したときには、既存の機能を組み合せたり、変更することで新たな要求をカバーできるシステムが必要となる。このようなシステムの 1 つとして、現在、我々はシステム全体を自然言語 — ここでは日本語 — を用いて記述、操作できるシステムの研究を進めている。

現在、我々が開発しているシステムでは、ユーザインタフェースとして誰もが理解でき、使用できる自然言語を用いている。さらに、このユーザインタフェース自身も日本語を用いて記述することによって、提供している機能を理解し、あらかじめ用意された機能を越える要求に対してユーザ自身が日本語を読み書きすることによって対応できることを目指している (図 2)。

また、このようなシステムを利用する人々には、提供された機能やそれをいくつか組み合せた利用をする利用者 (エンドユーザ) から、このような利用者の利用するアプリケーションプログラムを開発する利用者まで、さまざまな技術やシステムへの要求を

持つ利用者が存在する。これらの幅広い利用者に、基本的に同一な利用環境を提供するため、ユーザインタフェースだけではなく、プログラミング言語やオペレーティングシステム (OS) までを含めたシステム全体を、自然言語により記述することを考えている。これにより、エンドユーザから開発者までの多くの人々に使いやすいシステムが構築できる。

ここでいう日本語というのは自然言語である日本語そのものではなく自然言語である日本語にできるだけ近付けた制限言語でかつ生成文法をもつものであるとする。ただし、通常の日本語能力を持っていれば、誰もが特別な学習なしに容易に理解できるものとする。

以下も特に断わりのない場合、「日本語」はこのようなものを指すものとする。

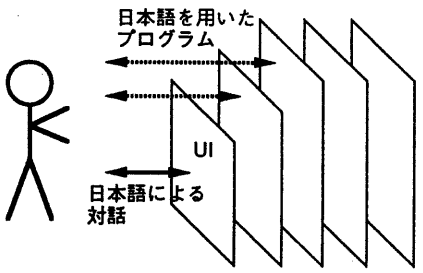

図 2. 日本語を用いたシステム

2 日本語を基盤とする情報システムの全体像

現在我々が開発を進めているシステム全体の構造を以下に示す (図 3)。

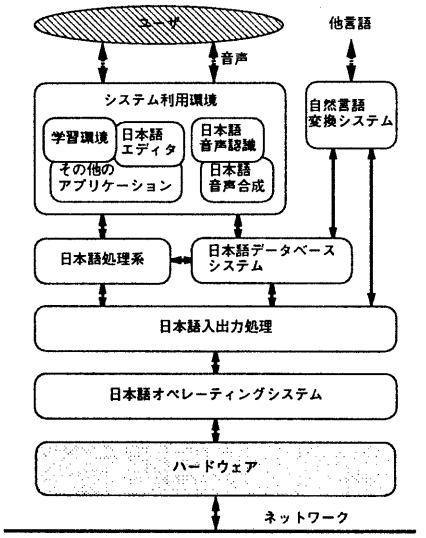

図 3. システムの基本構造

このシステムは、全体の資源を管理する日本語 OS を基盤とし、

- OS と外部との入出力を自然な形で行うために介在する日本語入出力処理。
- 日本語を基盤とするデータベースの構築、およびシステムが日本語を解釈する際に必要なデータを提供するための日本語データベースシステム。
- ユーザの要求に応じて日本語を解釈し実行する日本語処理系。
- システムの利用を容易に、効率よくするための利用環境。

などを中心に構成される。

この他、日本語以外の言語を用いるユーザや日本語以外の言語を基盤とするシステムとのインタフェースとなる自然言語変換システム、も併せて開発することを計画している。

本稿では、主にこのシステム中の日本語処理系部分について述べる。

3 設計

先に述べたように、ユーザインタフェースおよびその記述言語として日本語を用いる。これにより、ユーザインタフェースの利用およびその拡張が容易になる。

記述言語として日本語を用いるということは、1つ下の階層のインタフェースに日本語を用いることである。ここで、そのインタフェースもまた日本語で記述する、というように階層化されたシステムの各階層のインタフェースとして日本語を用いることにより、システム全体を統合的に日本語で記述することができる。これにより、ユーザの要求に応じて無理なくシステム全体を理解し拡張することができるようになると考えられる。

3.1 記述言語の設計

ここで、日本語を基盤とするプログラミング言語の具体的な設計について述べる。日本語をユーザインタフェースとして用いることについては3.2節で述べる。

既存のプログラミング言語はコンピュータで処理するために作られたものであるから、当然コンピュータで処理しやすく、その手法も確立されている。それとは逆に、自然言語そのものをコンピュータで処理させるのは不可能であり、自然言語に近い処理系を作るのも難しいとされている。

そこで、まず、コンピュータで扱いやすいプログラミング言語に自然言語である日本語の考え方を導入することを考える。

導入の仕方には

- 既存の言語の変数名などに日本語を使えるようにする。
- 既存の言語の構文を日本語に置き換える。
- 文の要素の順序が入れ替わるなどしても受け付けるようにする。
- 文の要素の省略を補う。

⋮

といったように、いくつものレベルが考えられる。

変数名などに日本語を使えるようにするというのは、既存の言語での

```
max = 100;
```

というものを、

```
最大値 = 100;
```

といった表記も認めるようにするというものである。

このレベルで解決しなければならない問題は、ほとんどが文字コードの問題である。これを解決し(日本人にとっての)プログラムの可読性を向上させることができたという報告がある[3]。

更にもう一歩すすめて変数名に日本語の文字を導入するだけでなく、制御構文などに日本語を取り入れることが考えられる。

これは、先の例を

```
最大値に 100 を代入する。
```

という形で評価することや、

```
while (max < 100) {
    ⋮
}
```

という意味のものを、

```
最大値 < 100 を満たしている間、
    ⋮
を繰り返す。
```

あるいは、

最大値が100より小さい間、
⋮
を繰り返す。

のように表記するというものである。

このレベルでは見た目は日本語に近くなるが、少し違ういいまわしで表現しようとすると、この程度の方法では文法エラーとなってしまう。

既存のプログラミング言語の構文と日本語を1対1に対応させている以上、これは避けられない。そこで、もう少し柔軟な構造をとることが考えられる。

具体的には、C言語における「a+ =2;」を「aに2を足す。」と表記できるようにする際、「X+=Y」を単に「XにYを足す」と置き換えるのではなく、「足す」という機能には「何に」「何を」という引数が必要であるといった文節単位での要求を文法として持ち、その順序に自由度を与えることによって「XにYを足す。」でも「YをXに足す。」でも同じものと解釈できるようになる。

更に、「足す」という機能には「何に」「何を」という引数が必要であるが、「Xを足す。」のように「何に」が省略された場合文法エラーとしてしまうのではなく、文脈に沿って適切なものにXを足すようにすることが考えられる。具体的には、「『直前に出てきたXを足すことができるもの』に」を補うなどが考えられる。

また、「足す」のかわりに「加える」「加算する」などの表現でも同じ内容を書き表すことができる。これについては、システム全体として、あるいはユーザの好みに応じて、同様の意味を持つ語を登録していくことができるようにすることによって、違う表現でも同じように処理することができるようにしている。これらの同様の意味を持つ語の登録・管理には、日本語データベースシステムを用いる。

その他、より自然言語である日本語に近付けるためには、文同士の連結やそれに伴う動詞の活用、などについても考慮しなければならない。

3.2 ユーザインタフェースの設計

日本語を用いたユーザインタフェースの設計も、プログラミング言語と同様に考えることができる。

ただし、プログラムよりも一過性の動作の指示が多く、文脈の解釈の仕方に違いがあると考えられる。プログラムの場合、作成したものに誤りがあった場合、それを修正した後に再実行することになるが、ユーザインタフェースとして用いる場合のようにインタラクティブな利用形態である場合には、誤りが発生する前の状態に戻したうえで、再び指示を与えるようにしたほうがよいであろう。このために、誤りが発生する前の状態に戻すことができるような仕組みが必要であるが、コンピュータが検出できる誤りと検出できない誤りがあるので、そういったことも考慮にいれなければならない。

また、ユーザが誤って重要な語の定義を書き換えてしまわない様にする保護機能なども、プログラムにおける同様の機能とは違った形態をとる必要があると考えられる。

4 日本語を基盤とした言語の実装

3.1節で述べた設計に基づき今回試作した日本語を基盤とするプログラミング言語は、

- 名詞の定義。
- 動詞の定義。
- プリミティブ、もしくは定義された動詞の記述 (実行)。

からなる。

名詞は従来のプログラミング言語の定数や変数、またはその型にあたり、動詞は関数にあたる。ただし、名詞の中には関数にあたるものもある。

現在、この言語は UNIX の上に構築されており、UNIX システムコールがプリミティブの一部となっている。将来的にはこの言語を用いて OS を作成し、既存の OS に依存せず完全に日本語で記述するものとする。

4.1 プリミティブ

暫定的に UNIX を基盤としているため、UNIX における概念、「ファイル」「ファイルディスクリプタ」などといった名詞 (型)、「開く」「読み出す」「書き込む」などといった動詞はすべてプリミティブである。

また、算術演算に必要な語や、動詞に引数を与えたり文のつながりを指示するための助詞や助動詞、その他、名詞や動詞などを定義するために必要な語はプリミティブとして用意する。

その他、句読点は文節や文の区切りとし、更に要素を細かく区切らないと解釈を誤る可能性がある場合には空白で区切る。文より大きな単位として、段落や節や章を設ける。

4.2 文脈

今回試作した処理系では、文脈に沿った文の解釈を行うために、いくつかの工夫をしている。

1つはそれぞれの型毎に規定値となる変数を1つ用意し、その型の値で最後に使われたものを保存しておく。動詞 (関数) の引数に不足がある場合は、足りない引数の型の規定値を利用する。また、指示代名詞がある場合にも、その規定値を参照するものとする。これにより、同じ型の変数を1つしか使わない場合には変数名を省略でき、より自然な日本語に近い表現が可能となる。

また、正式な型名が「セマフォのイベント変数」という型があって、それを利用する場合、直前で「セマフォ」という型が用いられていれば、単に「イベント変数」として参照することも可能にしている。

また、章・節・段落・文のように階層を設け、利用しようとする名詞や動詞の定義が複数ある場合には、前でかつもっとも近い階層のものを有効とする。

ただし、プログラムではなくユーザインタフェースとしての用途には、こういった階層は作らず、常に最新のものを有効にするなどとしたほうがよいと思われる。

4.3 形態素解析

この処理系では、プログラミング言語のパーサにあたる部分には形態素解析プログラムを用いている。この形態素解析プログラムでは、あらかじめ用意した単語やユーザが使用中に登録した単語の品詞情報と、あらかじめ用意された接続情報や活用を元に、入力された文を解析する。

現在用いているものでは前から順に考え

られる組み合わせすべてについて調べ、一意に定まった場合にのみそれを解釈するというものである (一意に定まらない場合にはエラー)。

語同士の接続に重み付けをするなどして一意にする方法もあるが、問題点も多いため、今回は行っていない。

動詞や名詞にひらがなが多く含まれていると、助詞や助動詞と混同して文が一意に定まらないことが多いため、現在はうまく解析できない部分は人間が手で空白を入れることによって回避している。

4.4 実装

現在、ここまで述べた処理系のインタプリタを、C言語で実装している。これは、形態素解析を行い中間形式として格納する部分と、その中間形式の実行部分に大きく別れる。中間形式を実行する部分は lisp インタプリタに近い構造になっておりこれはほぼ完成したと考えられるが、形態素解析部分は改良しなければならない点が数多くある。

今後、この処理系を完成させ、必要に応じてコンパイルして利用できるようなコンパイラを作成する。さらにこの処理系を用いて、OSを記述する。

4.5 記述例

例1. $\sum_{n=1}^{10} n$ を求める。

Cでの記述例:

```
int i, x = 0;
for (i = 1; i <= 10; i++)
    x += i;
printf("%d\n", x);
```

日本語での記述例:

1から10までの整数を足したものを
表示する。

(「1から10までの整数を加算し、
それを表示する。」などでも可。)

ただし、Cでprintf()が標準ライブラリにあるように、「表示する」等は定義済みであるものとする。

例2. セマフォの記述例。

日本語での記述例:

セマフォはイベント変数と待ち行列
からなる型である。

セマフォを獲得するときには、
イベント変数が0ならば、
実行中のプロセスを待ち行列に入れ、
そのプロセスを待ち状態に移す。
そうでなければ、イベント変数から
1を引く。

セマフォを解放するときには、
待ち行列が空でなければ
そこからプロセスを取り出し、
そのプロセスを実行状態に移す。
そうでなければ、イベント変数に1を足す。

4.6 実行

ここまでで述べたようなプログラムを、ユーザが実際に作成して実行する手順を以下に図示する (図4)。

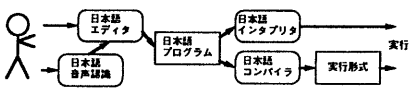

図4. プログラム作成の流れ

4.7 問題点

このような処理系を用いる上で問題となる点として、いままであげた以外に以下のようなものがある。

- 日常使っている自然言語をコンピュータで完全に扱うことは不可能であるため、似て非なるものとなってしまい、先入観がある分使いにくくなってう。
- 曖昧な文やどう補うかわからないような省略のある文をどう扱うか。文脈をどう解釈するか。
- 日本語は入力が面倒である。

これらの解決するために、できるだけ普段使っている自然言語を正しく解釈できるように試行を繰り返す、解釈困難な部分の訂正をを前もってユーザに促す機能を付加する、よく使われる語の略記法を用意する、などの解決法をとることを検討している。

5 今後の課題

日本語によってユーザインタフェースを始めとする種々のプログラムを記述することによって、誰もが使えるユーザインタフェースを構築し、また、誰もが必要に応じてユーザインタフェースを拡張することができる環境を構築することができると考えているが、そのためにはまだ考えなければならない事項が数多くある。

語の定義を書き換える際、その影響が及ぶ範囲を現在のところ単一の階層構造によって制限しているが、OSを構築する際にはユーザ権限など多岐にわたる制限の階層が必要となると考えられる。こういった構造をどのように表現・保持し、実行するかは今後の課題である。

また、日本語によって記述されたプログラムから、そのプログラムがどのような手続きによって構成されているかを知ることができるが、これだけではユーザの「どんな機能か」「どんな機能があるか」という要求には、必ずしも答えられない。これは、言語のみの問題ではなくこの言語を用いて機能がわかりやすいように手続きを記述する、といったプログラミングスタイルが重要になってくる。また、そういったスタイルが反映され、書きやすい言語にしていくのも課題のひとつである。

今回、言語として日本語に的をしぼって考えてきたが、このような環境で作成したプログラムを国際的に流通させる場合にどうするか、などといった問題も残されている。

参考文献

[1] 佐藤究, 布川博士, 野口正一: “分散環境のためのユーザインタフェースとその実現”, 電気情報通信学会技術研究報告 通信方式研究会, CS-93-13, pp.103-112 (1993).

[2] 佐藤究, 布川博士, 野口正一: “ユーザインタフェース・メタファーの定性的評価とその考察”, 情報処理学会 グループウェア研究会報告, Vol.93, No.75, 93-GW-3, pp.41-48 (1993).

[3] 中川正樹, 早川栄一, 並木美太郎, 高橋延匡: “母語プログラミングの理念, 実現, 実践とその効果”, 電子情報通信学会論文誌, J77-D-I, 5, pp.364-374 (1994).