論　文

# ユーザー・コンピュータ相互作用過程における作業の複雑さの定量化

正　員　北島　宗雄†

# Quantification of Task Complexity in Human-Computer Interaction Process

Muneo KITAJIMA †, *Member*

**あらまし**　ユーザーにとって使い勝手のよいシステムを設計するには，ユーザーの特性をよく理解することが必要である．本論文では，その重要な要素の一つとして，あるシステムを利用して作業を遂行するときにユーザーが感じる作業の認知的複雑さを取り上げ，それを定量化するための枠組みを提案する．まず，ユーザーが相互作用をしているシステムを，ユーザーがそのシステムを用いて作業を行うときに知覚する「概念」と，それに付随して参照できる「属性」の組で表される，「認知単位」を記述要素とするプロダクションルールにより表現する．そして，ユーザーが作業遂行過程で抱く目標をシステム表現における作用部に，また，ユーザーが把握するシステムの状態を条件部に対応させることにより作業記憶を構成し，作業の複雑さを，作業記憶内平均要素数，作業記憶への新規付加要素数により定量化する．更に，本枠組みを適用することによる，実験用英文テキストエディタの構成，および，その上で実行される簡単な単語交換作業における3種類の系列に対して作業の複雑さの定量化を行い，各系列の複雑さが本枠組みによりうまく表現されることを示す．更に，本枠組みがシステム設計の指針も提供することを示す．

## 1. まえがき

コンピュータが社会に浸透するに伴い，コンピュータを利用して作業を行う機会が増大してきている．また，その流れに呼応するように，そうした環境におけるユーザーの行動の認知科学的アプローチによるモデル化に関する研究が行われるようになってきた[1],[2]．そこでの目的の一つは，使い勝手のよいシステムを設計するための指針を得ることであるが，その重要な要素の一つとして，あるシステムを用いて作業を遂行するときにユーザーが感じる作業の認知的複雑さが挙げられる．

Kieras と Polson[3]は，ユーザーから見たデバイスあるいはシステムの認知的複雑さを記述するための枠組みを提案している．その枠組みによれば，ユーザーのジョブタスク知識（メソッドと選択規則）は，マニュアルにおける記述を基にして構成されるプロダクションルールにより，また，システムは，GTN（Generalized Transition Network）と呼ばれるネットワークによって表現される．GTNは，生成文法を用いたシステム表現[4]や通常の遷移ネットワーク表現のもつ問題点を克服し，システムの階層構造を自然に表現できる表現方法として考案されたものである．

彼らはユーザーとデバイスの相互作用過程を記述するために，双方の表現とインタプリタから成る実験シミュレーションシステムを構成し，ユーザー表現インタプリタに入力されたタスク表現の認知的複雑さを，両インタプリタを介してユーザーとデバイスが相互作用する過程で得られる情報，例えば，発火したプロダクション数，作業記憶に蓄えられたゴール数，キーストローク数などを基にして，定量化しようとしている．

このアプローチでは，ユーザー表現が重要な役割を果たし，それは，あらかじめ仮定されたプロダクションルールの集合によって表現される．しかしながら，ユーザー表現が，ユーザー間で普遍的でないばかりでなく，同一ユーザーでも経験，学習によって変化する不安定なものであると考えられ，従って，それを直接反映するような複雑さの定量化には問題があるといえる．

† 製品科学研究所基礎人間工学部，茨城県
Human Factors Research Department, Industrial Products Research Institute, Ibaraki-ken, 305 Japan

本論文では，上述のようなユーザー表現をあらかじめ仮定せずに，作業遂行過程を，ユーザーがあるシステムを用いて作業を行うときに知覚する「概念」と，それに付随して参照できる「属性」の組で表される，ユーザーの「認知単位」の系列として記述し，それを基に作業の複雑さを定量的に導出する方法を提案する．

2.では，認知単位によるシステムの一般的な表現方法について述べる．3.では，英文テキストエディタを例として取り上げ，そのシステムにおけるユーザーの認知単位を明らかにしながら，実験用エディタを構成する．4.では，システム表現要素による作業記憶の一般的な表現方法と，作業記憶と複雑さの関係について述べ，具体例として，実験用エディタを用いて遂行される簡単な単語交換作業におけるいくつかの作業系列について複雑さの比較を行う．

## 2. システム表現

### 2.1 認知単位によるシステム状態の表現

ある時点におけるシステムの状態 $S$ は，ユーザーが認知できる情報の集合によって書き表される．すなわち，そのシステムに関してユーザーがもつと想定される概念 $C_i$ ($i=1$，…,$N$ : $N$ は概念数) とそれに付随して参照できる属性 $A_{ij}$ ($j=1$，…,$N_i$ : $N_i$は属性の数) の組で定義される認知単位，$U_{ij}=(C_i, A_{ij})$，および，その値$\langle U_{ij}\rangle$によって，$S$ は，

$S=\{(U_{ij},\langle U_{ij}\rangle)\}$

と表される．ここで，値は，直接参照される対象の値である場合と，他の認知単位を引き数とする関数により階層的に記述される場合がある．例えば，テキストエディタの概念としては，校正作業中のテキスト，カーソル，システムのモードなどが考えられ，カーソルという概念に関しては，属性として，状態 ( {利用可，不可} )，色 ( {緑，青，赤,…} )，点滅 ( {有，無} )，位置 ( (list〈カーソル　行〉〈カーソル　欄〉) : list はLISPの関数)，行 ( {1，2，…} )，欄 ( {1，2，…} )，などが考えられる．ここで，かっこ内は各属性の取り得る値である．

なお，Anderson[5]は，認知単位という言葉を，命題，文字列，空間イメージを指し示し，要素の集合を特別な関係の中で記号化したものとして定義し，複雑な構造は一つの認知単位を他の認知単位の要素とする階層的な構造によって生成されると考えている．また，この考えは，認知科学の分野ではよく知られたものであり，多くの証拠があると指摘している．本論文での認知単位によるシステム状態の記述もそれを踏襲するものである．

### 2.2 システム表現

ユーザーからの行為を待っている状態にあるシステムに対して，ユーザーが何らかの物理的作用を及ぼすと，システムはそれに応じてさまざまな内的処理を行い，その結果を出力した後に，再びユーザーからの行為を待つ状態に遷移する．システム表現は，システムが受け取れるユーザーからのメッセージの各々に対して定められる規則，

メッセージ
　規則 1 : if (条件部) then (作用部)
　規則 2 : if (条件部) then (作用部)
　　⋮

をすべて網羅したものである．ここで，各規則の条件部には，システムの状態に関する条件が書かれており，そのすべてが満たされれば，その規則が適用され，作用部の記述に従って状態が更新される．

条件部における照合は，そこに書かれた条件式の真偽を判定することによって行われる．条件式は，述語といくつかの引き数によって表され，引き数は，直接参照できる値，もしくは，認知単位の値，〈概念，属性〉であり，後者はその評価値が用いられる．例えば，カーソルがホームポジション“H. P.”にあるかどうかの確認は，条件式，

(＝〈カーソル　位置〉“H. P.”)

の真偽を判定することによって行われる．

作用部では，システム状態の更新を行うが，概念 $C_i$ の属性 $A_{ij}$ の値の新しい値 $V$ (変更値) への更新は，3引き数関数“！”により実行され，

(！　$C_i$　$A_{ij}$　$V$)

と記述される．

例えば，メッセージ“→”(カーソルを右に1欄移動) に対するシステム表現は，

R1 : if (＝〈カーソル　欄〉〈画面　右マージン〉)
　then (！　カーソル　欄　1)
　　(！　カーソル　行 (1＋〈カーソル　行〉))
R2 : if (〈〈カーソル　欄〉〈画面　右マージン〉)
　then (！ カーソル　欄 (1＋〈カーソル　欄〉))

となり，規則 1(R1) は，認知単位 (カーソル　欄) の値が〈画面　右マージン〉に等しければ次の列の先頭を新しいカーソル位置とすることを示し，規則 2 (R2) は，その他の場合には現在の欄の値に1を加えたものを新しい欄の値とすることを示している．

## 3. 実験用エディタ

### 3.1 実験用エディタの認知単位

実験用エディタの概念として，＊TEXT，＊CURSOR-IN-TEXT，＊EOF，＊LOC-IN-TEXT，＊SYSTEMの五つを設定し，各概念に付随して参照できる概念のさまざまな側面を表す属性を，それぞれ，21，4，3，6，20種類定めた．一例として，図1に＊TEXT，＊CURSOR-IN-TEXTの2概念について，属性と値を示した．なお，属性は，ユーザーが直接観測できると仮定されるものと，そうでないものを区別し，後者に属するものを"＝…＝"と記述している．また，下線が施されている値は，作用部によって値が書き換えられる値であり，示されている値はその初期値である．

値には，概念＊CURSOR-IN-TEXTの属性COLのように値が直接書かれているものと，他の認知単位の値を引き数としてもつ関数（小文字で示されている）を評価して値が決定されるものとがある．例えば，概念＊TEXTの属性WORD-AT-CURSORの値は，

(get-word ⟨＊TEXT TEXT-AT-CURSOR⟩
⟨＊CURSOR-IN-TEXT COL⟩)

の評価値であり，関数get-wordの第1引き数，

⟨＊TEXT TEXT-AT-CURSOR⟩の値は，

(get-line ⟨＊TEXT BODY⟩
⟨＊CURSOR-IN-TEXT ROW⟩)

の評価値である．これは，更に，

(get-line "INPUT-TEXT" 1)

となる．また，第2引き数⟨＊CURSOR-IN-TEXTCOL⟩の値は1なので，結局，

⟨＊TEXT WORD-AT-CURSOR⟩
＝(get-word (get-line "INPUT-TEXT" 1) 1)

の形になり，位置(1　1)に存在する1単語が結果として与えられる．

```
Concept : *TEXT
Attribute                  Value
BODY                       "INPUT-TEXT"
LINE-SIZE                  (length <*TEXT BODY>)
TEXT-AT-CURSOR             (get-line <*TEXT BODY>
                                     <*CURSOR-IN-TEXT ROW>)
TEXT-ABOVE-CURSOR          (get-line <*TEXT BODY>
                                     (1- <*CURSOR-IN-TEXT ROW>))
TEXT-BELOW-CURSOR          (get-line <*TEXT BODY>
                                     (1+ <*CURSOR-IN-TEXT ROW>))
TEXT-AFTER-CURSOR          (get-text-after-cursor
                                     <*TEXT TEXT-AT-CURSOR>
                                     <*CURSOR-IN-TEXT COL>)
TEXT-BEFORE-CURSOR         (get-text-before-cursor
                                     <*TEXT TEXT-AT-CURSOR>
                                     <*TEXT TEXT-ABOVE-CURSOR>
                                     <*CURSOR-IN-TEXT COL>)
WORD-AT-CURSOR             (get-word <*TEXT TEXT-AT-CURSOR>
                                     <*CURSOR-IN-TEXT COL>)
WORD-AFTER-CURSOR          (get-word-after-cursor
                                     <*TEXT TEXT-AT-CURSOR>
                                     <*CURSOR-IN-TEXT COL>)
WORD-BEFORE-CURSOR         (get-word-before-cursor
                                     <*TEXT TEXT-AT-CURSOR>
                                     <*CURSOR-IN-TEXT COL>)
CHAR-AT-CURSOR             (get-char <*TEXT TEXT-AT-CURSOR>
                                     <*CURSOR-IN-TEXT COL>)
SELECT-RANGE               (cut-text <*TEXT BODY>
                                     <*CURSOR-IN-TEXT LOC>
                                (search "F" <*SYSTEM =TEXT=>
                                    "BEGINNING" "SELECT-MARK"))
TEXT-LENGTH-AT-CURSOR      ([s] <*TEXT TEXT-AT-CURSOR>)
TEXT-LENGTH-ABOVE-CURSOR   ([s] <*TEXT TEXT-ABOVE-CURSOR>)
TEXT-LENGTH-BELOW-CURSOR   ([s] <*TEXT TEXT-BELOW-CURSOR>)
TEXT-LENGTH-AFTER-CURSOR   ([s] <*TEXT TEXT-AFTER-CURSOR>)
TEXT-LENGTH-BEFORE-CURSOR  ([s] <*TEXT TEXT-BEFORE-CURSOR>)
WORD-LENGTH-AT-CURSOR      ([s] <*TEXT WORD-AT-CURSOR>)
WORD-LENGTH-AFTER-CURSOR   ([s] <*TEXT WORD-AFTER-CURSOR>)
WORD-LENGTH-BEFORE-CURSOR  ([s] <*TEXT WORD-BEFORE-CURSOR>)
SELECT-LENGTH              ([s] <*TEXT SELECT-RANGE>)
Concept : *CURSOR-IN-TEXT
Attribute                  Value
STATUS                     ACTIVE
LOC                        (list <*CURSOR-IN-TEXT ROW>
                                 <*CURSOR-IN-TEXT COL>)
ROW                        1
COL                        1
```

図1　テキストエディタにおける認知単位の例
Fig. 1　An example of cognitive units in a text editor.

### 3.2 実験用エディタのシステム表現

入力キーとして，1文字入力，カーソル移動，文字(単語，行)単位の挿入／削除等の機能を実現できる24種類を設定した．表1は，4.の実験で用いられる11種類についてそれらの機能を説明したものである．

各入力キーに対して**3.1**で定めた認知単位による表現を行うことにより，実験用エディタを記述した．その一例として，1文字挿入に関して定められた四つの規則を図2に示す．規則1は，システムが第2機能モード，選択モードでない場合に適用され，テキストのカーソル位置に1文字を挿入した後，カーソルを現在位置の隣の位置に移動することを示している．規則2，3は，選択モードになっているときの処理を示しており，前者は選択位置がカーソル位置に一致している場合，後者はそうでない場合である．いずれも，規則1の作用部の処理を行うが，その他に，前者では，システム作業用テキスト上で"SELECT-MARK"の付換えを行い，後者は，システム作業用テキストに1文字を挿入する．規則4は，第2機能モードになっている場合に適用されるが，1文字挿入に対してはこのモードは有効でないので，このモードをオフにする．

なお，表1の他のキーに関連して定められた規則数は，2ND(1)，SELECT(3)，→(2)，WORD(3)，EOL(7)，DELCHAR(9)，DELCHAR<(5)，DELWORD(10)，DELWORD<(5)，CUT(7)となっている．

表1　実験用エディタの入力キー

| 入力キー | 第 1 機 能 | 第 2 機 能 |
|---|---|---|
| 2ND | 第2機能を指定 | なし |
| SELECT | 選択範囲の一つの端の位置を指定 | 第2機能の中止，選択範囲の解除 |
| “A” etc. | 入力された文字をカーソル位置に挿入 | なし |
| → | カーソルを次の文字位置に移動 | なし |
| WORD | カーソルを単語の先頭位置に移動 | なし |
| EOL | カーソルを改行文字位置に移動 | カーソル位置から次の改行文字までの範囲を消去し，行バッファに格納 |
| DELCHAR | カーソル位置の1文字を消去し，文字バッファに格納 | 文字バッファ内の文字を挿入 |
| DELCHAR< | カーソル直前の1文字を消去し，文字バッファに格納 | なし |
| DELWORD | カーソル位置から次単語の先頭文字までの範囲を消去し，単語バッファに格納 | 単語バッファ内の文字列を挿入 |
| DELWORD< | カーソル位置直前からそれ以前の単語の先頭文字位置の範囲を消去し，単語バッファに格納 | なし |
| CUT | 選択範囲を消去し，ペーストバッファに格納 | ペーストバッファ内の文字列を挿入 |

## 4. 複雑さの定量化

### 4.1 作業記憶と複雑さ

ユーザーは，与えられたタスクを遂行するために一連のキー操作を行うが，各キー操作を行う際には，目的の意識化，タスクの進行状況を含むシステム状態の把握を行っていると考えられる．ここでは，作業記憶がこれらの要素により表現されると仮定し，システム表現の条件部を状態把握に，作用部を目的に対応させることによりその構成を行う．

いま，ユーザーが $n$ 番目のキー操作 $K_n$ を行ったときに適用された規則を $R_n$ とする．作業記憶 $W_n$ は，規則 $R_n$ の条件 / 作用部の個々の式，$C_{ni}$ ($1 \leq i \leq R_n$ の条件数)，$A_{nj}$ ($1 \leq j \leq R_n$ の作用数) について，以下の基準を当てはめることによって構成される．なお，作業記憶の要素は，条件 / 作用式に作業記憶内での状態に関する情報 $S_{wn}$ を付加したものであり，
$[C_{ni},\ S_{wm}]$，$[A_{nj},\ S_{wm}]$ と表される．

```
Rule 1
if   (== <*CURSOR-IN-TEXT STATUS> ACTIVE)
     (== <*SYSTEM =2ND=> OFF)
     (== <*SYSTEM =SELECT=> OFF)
then (! *TEXT BODY
       (insert <*TEXT BODY> <*CURSOR-IN-TEXT LOC> <*SYSTEM KEY>))
     (! *CURSOR-IN-TEXT (ROW COL)
       (next <*TEXT BODY> <*CURSOR-IN-TEXT LOC> 1))
Rule 2
if   (== <*CURSOR-IN-TEXT STATUS> ACTIVE)
     (== <*SYSTEM =2ND=> OFF)
     (== <*SYSTEM =SELECT=> ON)
     (== <*SYSTEM =CHAR-AT-CURSOR=> "SELECT-MARK")
then (! *TEXT BODY
       (insert <*TEXT BODY> <*CURSOR-IN-TEXT LOC> <*SYSTEM KEY>))
     (! *SYSTEM =TEXT=
       (put-mark <*TEXT BODY> <*CURSOR-IN-TEXT LOC> "SELECT-MARK"))
     (! *CURSOR-IN-TEXT (ROW COL)
       (next <*TEXT BODY> <*CURSOR-IN-TEXT LOC> 1))
Rule 3
if   (== <*CURSOR-IN-TEXT STATUS> ACTIVE)
     (== <*SYSTEM =2ND=> OFF)
     (== <*SYSTEM =SELECT=> ON)
     (!= <*SYSTEM =CHAR-AT-CURSOR=> "SELECT-MARK")
then (! *TEXT BODY
       (insert <*TEXT BODY> <*CURSOR-IN-TEXT LOC> <*SYSTEM KEY>))
     (! *SYSTEM =TEXT=
       (insert <*SYSTEM =TEXT=> <*CURSOR-IN-TEXT LOC>
                                <*SYSTEM KEY>))
     (! *CURSOR-IN-TEXT (ROW COL)
       (next <*TEXT BODY> <*CURSOR-IN-TEXT LOC> 1))
Rule 4
if   (== <*CURSOR-IN-TEXT STATUS> ACTIVE)
     (== <*SYSTEM =2ND=> ON)
then (! *SYSTEM =2ND= OFF)
```

図2　認知単位によるシステム表現の例
Fig. 2　An example of system representation with cognitive units.

[基準1]（観測可能な条件）　$C_{ni}$ の中のすべての引き数が観測可能な属性に関するものである場合で，その条件式が直前の作業記憶 $W_{n-1}$ 内にあれば[$C_{ni}$“継続”]，なければ[$C_{ni}$“付加”]を $W_n$ に入れる．

[基準2]（観測可能な作用）　$A_{nj}$ の中で値が変更される属性が観測可能な属性であり，かつ，変更値の記述に現れるすべての属性が観測可能である場合で，その作用が $W_{n-1}$ 内にあれば [$A_{nj}$“継続”]，なければ [$A_{nj}$“付加”] を $W_n$ に入れる．

[基準3]（観測不可能な条件（作用））　$C_{ni}$ の中の引き数（$A_{nj}$ の中の変更値）の中に観測不可能な属性を含む認知単位の値<＊C＝ATR=>があるが，その認知単位，あるいは，それに間接的に関連する認知単位が以前のキー操作 $K_{n'}$ ($n' \leq n$) における作用 $A_{n'j'}$ によって変更されていれば $A_{n'j'}$ が実行されたときの作業記憶 $W_{n'}$ に [$A_{n'j'}$“付加”or“継続”] を入れ，途中に入力されたキーがあれば，$W_{n''}$ ($n' < n'' < n$) に [$A_{n'j'}$“心的継続”] を入れる．また，当該条件（作用）を $W_n$ に [$C_{ni}$（$A_{nj}$）“解決”（and“継続”）] として入れる．

以上の手続きにより導出される作業記憶内の要素数は作業の複雑さを反映しており，平均要素数が多いほど，また，作業記憶への新規付加の割合が大きいほど作業の複雑性が高いといえる．

### 4.2 単語交換作業

課題として，初期テキスト，
“COMPUTER AND INFORMATION PROCESSING>”を，ゴールテキスト，
“INFORMATION PROCESSING AND COMPUTERS>”
に変換する，2単語の移動と1単語の複雑化からなる単語交換作業を設定した．ここで，“>”は改行コードである．

この作業に関して想定されたキー系列は，“DELCHAR”を13回，“→”を22回，1文字挿入を14回行う，非常に単純な系列1，系列1における文字

Sequence 1

| No. | KEY | Working Memory | Mi | Mo | Mm | Mt | COMPUTER AND INFORMATION PROCESSING> |
|---|---|---|---|---|---|---|---|
| 1 | DELCHAR | C-01 C+03 A+04 | 3 | 0 | 0 | 3 | OMPUTER AND INFORMATION PROCESSING> |
| 2 | DELCHAR | \| \| \| | 0 | 0 | 3 | 3 | MPUTER AND INFORMATION PROCESSING> |
| : | : | \| \| \| | : | : | : | : | : |
| 13 | DELCHAR | \| \| \| | 0 | 0 | 3 | 3 | INFORMATION PROCESSING> |
| 14 | -> | \| \| A+19 | 1 | 1 | 2 | 3 | INFORMATION PROCESSING> |
| 15 | -> | \| \| \| | 0 | 0 | 3 | 3 | INFORMATION PROCESSING> |
| : | : | \| \| \| | : | : | : | : | : |
| 35 | -> | \| \| \| | 0 | 0 | 3 | 3 | INFORMATION PROCESSING> |
| 36 | ' ' | \| A+01 \| | 1 | 1 | 2 | 3 | INFORMATION PROCESSING > |
| 37 | 'A' | \| \| \| | 0 | 0 | 3 | 3 | INFORMATION PROCESSING A> |
| 38 | 'N' | \| \| \| | 0 | 0 | 3 | 3 | INFORMATION PROCESSING AN> |
| 39 | 'D' | \| \| \| | 0 | 0 | 3 | 3 | INFORMATION PROCESSING AND> |
| 40 | ' ' | \| \| \| | 0 | 0 | 3 | 3 | INFORMATION PROCESSING AND > |
| 41 | 'C' | \| \| \| | 0 | 0 | 3 | 3 | INFORMATION PROCESSING AND C> |
| 42 | 'O' | \| \| \| | 0 | 0 | 3 | 3 | INFORMATION PROCESSING AND CO> |
| 43 | 'M' | \| \| \| | 0 | 0 | 3 | 3 | INFORMATION PROCESSING AND COM> |
| 44 | 'P' | \| \| \| | 0 | 0 | 3 | 3 | INFORMATION PROCESSING AND COMP> |
| 45 | 'U' | \| \| \| | 0 | 0 | 3 | 3 | INFORMATION PROCESSING AND COMPU> |
| 46 | 'T' | \| \| \| | 0 | 0 | 3 | 3 | INFORMATION PROCESSING AND COMPUT> |
| 47 | 'E' | \| \| \| | 0 | 0 | 3 | 3 | INFORMATION PROCESSING AND COMPUTE> |
| 48 | 'R' | \| \| \| | 0 | 0 | 3 | 3 | INFORMATION PROCESSING AND COMPUTER> |
| 49 | 'S' | \| \| \| | 0 | 0 | 3 | 3 | INFORMATION PROCESSING AND COMPUTERS> |
| Total | | | 5 | 2 | 142 | 147 | |
| Average | | | 0.1 | 0.0 | 2.9 | 3.0 | |

Sequence 2

| No. | KEY | Working Memory | Mi | Mo | Mm | Mt | COMPUTER AND INFORMATION PROCESSING> |
|---|---|---|---|---|---|---|---|
| 1 | DELWORD | C+01 C+03 A+08 Av05 | 4 | 0 | 0 | 4 | AND INFORMATION PROCESSING> |
| 2 | DELWORD | \| \| A+17 \| Y | 1 | 0 | 4 | 5 | INFORMATION PROCESSING> |
| 3 | EOL | \| \| (I) (I) A+23 | 1 | 1 | 4 | 5 | INFORMATION PROCESSING> |
| 4 | ' ' | \| (I) (I) A+01 A+19 | 2 | 2 | 3 | 5 | INFORMATION PROCESSING > |
| 5 | 2ND | \| A+15 (I) (I) | 1 | 2 | 3 | 4 | INFORMATION PROCESSING > |
| 6 | DELWORD | \| Cv06 Cv08 Av02 A+16 | 4 | 3 | 1 | 5 | INFORMATION PROCESSING AND > |
| 7 | WORD | \| Cv07 C+03 A+20 | 3 | 4 | 1 | 4 | INFORMATION PROCESSING AND > |
| 8 | 'C' | \| A+01 A+19 | 2 | 3 | 1 | 3 | INFORMATION PROCESSING AND C> |
| 9 | 'O' | \| \| \| | 0 | 0 | 3 | 3 | INFORMATION PROCESSING AND CO> |
| 10 | 'M' | \| \| \| | 0 | 0 | 3 | 3 | INFORMATION PROCESSING AND COM> |
| 11 | 'P' | \| \| \| | 0 | 0 | 3 | 3 | INFORMATION PROCESSING AND COMP> |
| 12 | 'U' | \| \| \| | 0 | 0 | 3 | 3 | INFORMATION PROCESSING AND COMPU> |
| 13 | 'T' | \| \| \| | 0 | 0 | 3 | 3 | INFORMATION PROCESSING AND COMPUT> |
| 14 | 'E' | \| \| \| | 0 | 0 | 3 | 3 | INFORMATION PROCESSING AND COMPUTE> |
| 15 | 'R' | \| \| \| | 0 | 0 | 3 | 3 | INFORMATION PROCESSING AND COMPUTER> |
| 16 | 'S' | \| \| \| | 0 | 0 | 3 | 3 | INFORMATION PROCESSING AND COMPUTERS> |
| Total | | | 18 | 15 | 41 | 59 | |
| Average | | | 1.1 | 0.9 | 2.6 | 3.7 | |

Sequence 3

| No. | KEY | Working Memory | Mi | Mo | Mm | Mt | COMPUTER AND INFORMATION PROCESSING> |
|---|---|---|---|---|---|---|---|
| 1 | SELECT | C+01 A+13 A+10 | 3 | 0 | 0 | 3 | COMPUTER AND INFORMATION PROCESSING> |
| 2 | WORD | \| (I) (I) C+03 A+20 | 2 | 0 | 3 | 5 | COMPUTER AND INFORMATION PROCESSING> |
| 3 | DELCHAR< | \| Cv04 C+13 C+12 C+02 A+24 Av12 A+04 | 7 | 4 | 1 | 8 | COMPUTERAND INFORMATION PROCESSING> |
| 4 | 'S' | \| Y \| Av11 A+01 A+19 | 3 | 5 | 3 | 6 | COMPUTERSAND INFORMATION PROCESSING> |
| 5 | CUT | \| Y C+11 A+09 Av22 Av06 A+14 | 5 | 4 | 2 | 7 | AND INFORMATION PROCESSING> |
| 6 | WORD | \| (I) (I) C+03 A+20 | 2 | 4 | 3 | 5 | AND INFORMATION PROCESSING> |
| 7 | DELWORD< | \| C+02 (I) Cv05 A+18 A+07 Av25 Av05 | 6 | 3 | 2 | 8 | INFORMATION PROCESSING> |
| 8 | EOL | \| C+03 (I) A+23 (I) (I) | 2 | 4 | 4 | 6 | INFORMATION PROCESSING> |
| 9 | ' ' | \| A+01 A+19 (I) (I) (I) | 2 | 2 | 4 | 6 | INFORMATION PROCESSING > |
| 10 | 2ND | \| A+15 (I) (I) (I) | 1 | 2 | 4 | 5 | INFORMATION PROCESSING > |
| 11 | DELWORD | \| Cv06 (I) Cv09 Av02 Av21 | 4 | 3 | 2 | 6 | INFORMATION PROCESSING AND > |
| 12 | 2ND | \| A+15 (I) | 1 | 4 | 2 | 3 | INFORMATION PROCESSING AND > |
| 13 | CUT | \| Cv06 Cv10 Av03 Av22 | 4 | 2 | 1 | 5 | INFORMATION PROCESSING AND COMPUTERS> |
| Total | | | 42 | 37 | 31 | 73 | |
| Average | | | 3.2 | 2.8 | 2.4 | 5.6 | |

図3 単語交換作業遂行過程と作業記憶
Fig. 3 Key sequences for a word switching task and working memory.

削除を“DELWORD”，カーソル移動を“EOL”で行い，文字列“AND”の挿入を単語バッファを用いて行う，中程度のプランニングが要求される系列2，系列2における文字挿入をすべてバッファを用いて行う，高度なプランニングが要求される系列3の3種類である．

### 4.3 作業の複雑さ

図3は，系列1～3の各々について，入力されたキー，各キーが入力されたときの作業記憶の内容，入力後の画面表示内容（下線はカーソル位置を示す），作業記憶内要素数を示している．Mi，Mo，Mm，Mtはそれぞれ，付加された要素数，削除された要素数，保持されている要素数，全要素数である．また，記号C，Aはそれぞれ，条件，作用を，＋，|，(|)，vは，それぞれ，作業記憶内状態，“付加”“継続”“心的継続”“解決”を，それに続く数字は条件／作用識別番号を表す．なお，識別番号で表される条件，作用の詳細な記述は図4に与えられているが，作用の記述において関数“！”は省かれ，また，同じ認知単位に関する作用の変更値は“－”に続いて示されている．

```
Conditions
01 (== <*CURSOR-IN-TEXT STATUS>      ACTIVE)
02 (!= <*CURSOR-IN-TEXT LOC>         'BEGINNING')
03 (<< <*CURSOR-IN-TEXT LOC>         <*EOF LOC>)
04 (== <*SYSTEM =SELECT=>            ON)
05 (== <*SYSTEM =SELECT=>            OFF)
06 (== <*SYSTEM =2ND=>               ON)
07 (== <*SYSTEM =2ND=>               OFF)
08 (== <*SYSTEM =CURSOR-FIX-WORD=>   ON)
09 (== <*SYSTEM =CURSOR-FIX-WORD=>   OFF)
10 (!= <*SYSTEM =PASTE-BUFFER=>      '')
11 (<= <*LOC-IN-TEXT SELECT-LOC>     <*CURSOR-IN-TEXT LOC>)
12 (!= <*LOC-IN-TEXT SELECT-LOC>     <*CURSOR-IN-TEXT LOC>)
13 (!= <*LOC-IN-TEXT SELECT-LOC>
        (back <*TEXT BODY> <*CURSOR-IN-TEXT LOC> 1))

Actions
01 *TEXT BODY
     (insert <*TEXT BODY> <*CURSOR-IN-TEXT LOC> <*SYSTEM KEY>)
02 - (insert <*TEXT BODY> <*CURSOR-IN-TEXT LOC>
             <*SYSTEM =WORD-BUFFER=>)
03 - (insert <*TEXT BODY> <*CURSOR-IN-TEXT LOC>
             <*SYSTEM =PASTE-BUFFER=>)
04 - (delchar <*TEXT BODY> <*CURSOR-IN-TEXT LOC> 1)
05 - (delchar <*TEXT BODY> <*CURSOR-IN-TEXT LOC>
                           <*SYSTEM =WORD-BUFFER-LENGTH=>)
06 - (delchar <*TEXT BODY> <*CURSOR-IN-TEXT LOC>
                           <*SYSTEM =PASTE-BUFFER-LENGTH=>)
07 *SYSTEM =WORD-BUFFER= <*TEXT WORD-BEFORE-CURSOR>
08 *SYSTEM =WORD-BUFFER= <*TEXT WORD-AFTER-CURSOR>
09 *SYSTEM =PASTE-BUFFER= <*TEXT SELECT-RANGE>
10 *SYSTEM =TEXT=
     (put-mark <*TEXT BODY> <*CURSOR-IN-TEXT LOC> 'SELECT-MARK')
11 - (insert <*SYSTEM =TEXT=> <*CURSOR-IN-TEXT LOC> <*SYSTEM KEY>)
12 - (delchar <*SYSTEM =TEXT=> <*CURSOR-IN-TEXT LOC> 1>
13 *SYSTEM =SELECT= ON
14 *SYSTEM =SELECT= OFF
15 *SYSTEM =2ND= ON
16 *SYSTEM =2ND= OFF
17 *SYSTEM =CURSOR-FIX-WORD= ON
18 *SYSTEM =CURSOR-FIX-WORD= OFF
19 *CURSOR-IN-TEXT (ROW COL)
     (next <*CURSOR-IN-TEXT LOC> 1)
20 - (next <*CURSOR-IN-TEXT LOC> <*TEXT WORD-LENGTH-AFTER-CURSOR>)
21 - (next <*CURSOR-IN-TEXT LOC> <*SYSTEM =WORD-BUFFER-LENGTH=>)
22 - (next <*CURSOR-IN-TEXT LOC> <*SYSTEM =PASTE-BUFFER-LENGTH=>)
23 - <*LOC-IN-TEXT NEXT-LINE-TARMINATOR>
24 - (back <*CURSOR-IN-TEXT LOC> 1)
25 - (back <*CURSOR-IN-TEXT LOC> <*SYSTEM =WORD-BUFFER-LENGTH=>)
26 - (back <*CURSOR-IN-TEXT LOC> <*SYSTEM =PASTE-BUFFER-LENGTH=>)
```

図4　単語交換作業遂行過程で現れる作業記憶内要素
Fig. 4　Precise descriptions for the working memory elements in Fig. 3

さて，各系列における平均作業記憶内要素数は，系列1，2，3でそれぞれ，3，3.7，5.6，また，作業記憶への新規付加の割合（Miの総数／Mtの総数）はそれぞれ，3％，30％，58％である．この結果は，高度なプランニングが必要とされるほど，各時点で保持していなければならない要素，新しい情報の把握，また，ゴールの生成が必要とされることをよく表しており，本枠組みにより，種々の作業系列の複雑さの定量的比較が可能であることを示している．

更に，本枠組みは，作業の複雑さという観点から見た種々のシステムの比較を可能とする．例えば，作業の複雑さを増大させる要因の一つとして，観測不可能性によって作業記憶に付加された“心的継続”要素があるが，その認知単位を観測可能とするようにシステムを改良することによって，系列2では，平均作業記憶内要素数を3.3に，また，系列3では，4.4に減少させることができる．

## 5. むすび

本論文では，ユーザーがあるシステムを利用して作業を遂行する過程で感じる作業の複雑さを定量化するための枠組みとして，(1)認知単位によるシステム表現，(2)システム表現要素による作業記憶の構成，を提案した．そして，エディタに関して定めた認知単位を用いて実験用エディタを構成し，その上で実行される単語交換作業の3種類の系列に対して得られる作業記憶から，作業の複雑さを導出し，それが各系列の複雑さをよく反映していることを示した．また，本枠組みがシステム設計の指針も提供できることを示した．

より良いシステムを設計するには，ユーザーの特性をよく理解することが必要である．本論文では，その一つとしてユーザーの感じる複雑さを取り上げ，その定量化を試みた．そこでの重要な考え方は，認知単位を記述要素とする作業記憶の表現であり，それは，複雑さという側面だけでなく，学習や，プランニングなどを含むより広いユーザーの認知活動を表現する上で有用な表現方法であると考えられる．その可能性を検討していくことは，今後の課題である．

### 文　献

(1) S. K. CARD, T. P. MORAN and A. NEWELL : “The Psychology of Human-Computer Interaction”, Lawrence Erlbaum Associates, Hillsdale, New Jersey (1983).
(2) eds. D. A. Norman and S. W. Draper : “User Centered

System Design", Lawrence Erlbaum Associates, Hillsdale, New Jersey (1986).
(3) D. E. Kieras and P. G. Polson : "An approach to the formal analysis of user complexity", Int. J. Man-Machine Studies, **22,** pp. 365-394 (1985).
(4) P. Reisner : "Formal grammar and human factors design of an interactive graphics system", IEEE Trans. Software Eng., **SE-7,**2, pp. 229-240 (1981).
(5) J. R. Anderson : "The Architecture of Cognition", Harvard University Press, Cambridge, Massachusetts, London (1983).

(昭和62年3月5日受付)

**北島 宗雄**

昭53東工大・理・物理卒．昭55同大大学院修士課程了．同年工業技術院製品科学研究所入所．以来，盲人用読書支援システムに関する研究，知的支援システムのための思考過程の研究に従事．工博．現在，同所基礎人間工学部心理情報工学課勤務．