Appendix E

Task Descriptions for Proposed Designs

Command Feature

!! – Repeat the Most Recent Command

C_M	Recency("Command Line")
C_M	Get-!! (<!!>)
C_E	Initiate (<!>)
M_K	Key (<!>)
C_E	Initiate (<!>)
M_K	Key (<!>)
C_E	Initiate (<RETURN>)
M_K	Key (<RETURN>)

$$C_M + C_M + C_E + M_K + C_E + M_K + C_E + M_K$$

$$2C_M + 3C_E + 3M_K$$

Assumptions:

(1) Recency("Command Line") performed in a generic memory retrieval operator cycle time.

!name – Repeat Most Recent Command with Given Name

C_M	Get-! (<!>)
C_E	Initiate (<!>)
M_K	Key (<!>)
C_E	Command-Name ← Get-Name("Command Name")
	Count ← 1
	REPEAT BEGIN
C_E	Initiate (Command-Name[Count])
M_K	Key (Command-Name[Count])
	Count ← Count + 1
	UNTIL Is-Equal?(Count, $\bar{n}_0$)
C_E	IF-SUCCEEDED Is-Equal? (Count, $\bar{n}_0$) THEN BEGIN
C_E	Initiate (<RETURN>)
M_K	Key (<RETURN>)
	END

$$C_M + C_E + M_K + C_E + \bar{n}_0(C_E + M_K) + C_E + C_E + M_K$$

$$C_M + (\bar{n}_0 + 4)C_E + (\bar{n}_0 + 2)M_K$$

Assumptions:

(1) No memory retrieval required to Get-Name("Command Name") because the *command* is in a user's working memory, just a mental step to obtain the *command name* .

(2) $\bar{n}_0$ is the average length of the *command name* .

!number – Repeat Command with Given Number

C_M	Number ← Get-Number("Command Line")
C_M	Get-!n (<!n>)
C_E	Initiate (<!>)
M_K	Key (<!>)
C_E	Initiate (Number[First-Digit])
M_K	Key (Number[First-Digit])
C_E	Initiate (Number[Second-Digit])
M_K	Key (Number[Second-Digit])
C_E	Initiate (Number[Third-Digit])
M_K	Key (Number[Third-Digit])
C_E	Initiate (<RETURN>)
M_K	Key (<RETURN>)

$$C_M + C_M + C_E + M_K + 3(C_E + M_K) + C_E + M_K$$

$$2C_M + 5C_E + 5M_K$$

Assumptions:

(1) Users retain an association between the *command* and its number. Therefore, to recall the *number* , users retrieve the association.

(2) The most optimal method was selected, associating *number* with *command* , rather than other methods which require significantly more steps. For example, users can find the number by requesting a listing of their history list, scanning the list to find the *command* , and then noting the associated *number* .

(3) A typical session in which lots of work is being done would involve 100 to 999 commands. Thus, on the average, there are 3 digits to a *command* .

!?argument – Repeat Most Recent Command with Given Argument

C_M	Get-!? (<!?>)
C_E	Initiate (<!>)
M_K	Key (<!>)
C_E	Initiate (<?>)
M_K	Key (<?>)
C_M	Get-One-Argument ("Argument")
	Count $\leftarrow$ 1
	REPEAT BEGIN
C_E	Initiate (Argument[Count])
M_K	Key (Argument[Count])
	Count $\leftarrow$ Count + 1
	UNTIL Is-Equal?(Count, $\overline{n}_A$)
C_E	IF-SUCCEEDED Is-Equal? (Count, $\overline{n}_A$) THEN BEGIN
C_E	Initiate (<RETURN>)
M_K	Key (<RETURN>)
	END

$$C_M + C_E + M_K + C_E + M_K + C_M + \overline{n}_A(C_E + M_K) + C_E + C_E + M_K$$

$$2C_M + (\overline{n}_A + 4)C_E + (\overline{n}_A + 3)M_K$$

Assumptions:

(1) A mental retrieval is required to determine a suitable argument in the *command* which will retrieve the most recent copy of the desired *command* .

(2) The whole *argument* must be provided and not a substring of it.

(3) $\overline{n}_A$ is the average length of a command's *argument* .

History Menu

Popup – Menu Appear on User Request

C_M	Get-PopUpMenu (<MOUSE>)
C_E	Initiate (<MOUSE>)
M_H	Home (<MOUSE>)
C_E	Initiate (<DRAG>)
$M_{D_M}(l)$	Drag ("Item at Position l")
S	Wait-System-Response(< MENU >)
$P_S(l)$	Match ("Item at Position l")
C_E	Initiate (<KEYBOARD>)
M_H	Home (<KEYBOARD>)

$$C_M + C_E + M_H + C_E + \sum_{l=1}^{8} p_l M_{D_M}(l) + S + \sum_{l=1}^{8} p_l P_S(l) + C_E + M_H$$

$$C_M + 2M_H + 3C_E + S + \sum_{l=1}^{8} p_l (P_S(l) + M_{D_M}(l))$$

Assumptions:

(1) In estimates of Drag("Item at Position l"), the time takes into account the button down to cause menu to appear, drag through to the desired item, and then release of button to make selection.

Static – Menu Remains on Screen

C_M	Get-StaticMenu (<MOUSE>)
P_E	Eye-Move (<MENU>)
$P_S(l)$	Match ("Item at position l")
C_E	Initiate (<MOUSE>)
M_H	Home (<MOUSE>)
C_E	Initiate (<POINT>)
$M_{P_M}(l)$	Point ("Item at Position l")
C_E	Initiate (<KEYBOARD>)
M_H	Home (<KEYBOARD>)

$$C_M + P_E + \sum_{l=1}^{8} p_l P_S(l) + C_E + M_H + C_E + \sum_{l=1}^{8} p_l M_{P_M}(l) + C_E + M_H$$

$$C_M + 2M_H + 3C_E + P_E + \sum_{l=1}^{8} p_l (P_S(l) + M_{P_M}(l))$$

Assumptions:

(1) In estimates of Point("Item at Position l"), the movement time takes into account the time to get to the desired item, and a button down and up for the selection.

(2) To find the static menu, users must first move their eyes to region where the static menu is. P_E is 230 msecs which represents a typical eye movement value.

(3) No system time because menu remains on the screen.

Step Buffer

Single Step – Visually Examine Each Item Individually

C_M	Get-StepMenu (<STEP MENU>)
P_{S_0}	Prep-Search (<STEP MENU>)
	REPEAT BEGIN
C_E	Initiate (<STEP MENU>)
M_K	Key (<STEP MENU>)
S	Wait-System-Response(< STEP MENU>)
P_I	Command-Line ← Perceive-Text ("Command Line")
P_{S_1}	Match-Result ← Match (Command-Line)
	UNTIL Found? (Match-Result)
C_E	IF-SUCCEEDED Found? (Match-Result) THEN BEGIN
C_E	Initiate (<RETURN>)
M_K	Key (<RETURN>)
	END

$$C_M + P_{S_0} + \sum_{l=1}^{8} lp_l(C_E + M_K + S + P_I + P_{S_1}) + C_E + C_E + M_K$$

$$C_M + P_{S_0} + \sum_{l=1}^{8} lp_l(S + P_I + P_{S_1}) + (\sum_{l=1}^{8} lp_l + 2)C_E + (\sum_{l=1}^{8} lp_l + 1)M_K$$

Assumptions:

(1) <STEP MENU> key is a cursor key available on the keyboard. The time to press this key is equivalent to the time to press any other key on the keyboard, M_K.

Recall & Step – Recall How Recent and Step Right to It

C_M	Recency("Command Line")
C_M	Get-StepMenu (<STEP MENU>)
P_{S_0}	Prep-Search (<STEP MENU>)
	Count $\leftarrow$ 1
	REPEAT BEGIN
C_E	Initiate (<STEP MENU>)
M_K	Key (<STEP MENU>)
	Count $\leftarrow$ Count + 1
	UNTIL Is-Equal? (Count, l)
C_E	IF-SUCCEEDED Is-Equal? (Count, l) THEN BEGIN
S	Wait-System-Response(< STEP MENU>)
P_I	Command-Line $\leftarrow$ Perceive-Text ("Command Line")
P_{S_1}	Match-Result $\leftarrow$ Match (Command-Line)
C_E	IF-SUCCEEDED Found? (Match-Result) THEN BEGIN
C_E	Initiate (<RETURN>)
M_K	Key (<RETURN>)
	END
	END

$$C_M + C_M + P_{S_0} + \sum_{l=1}^{8} lp_l(C_E + M_K) + C_E + S + P_I + P_{S_1} + C_E + C_E + M_K$$

$$2C_M + P_{S_0} + P_{S_1} + S + P_I + (\sum_{l=1}^{8} lp_l + 3)C_E + (\sum_{l=1}^{8} lp_l + 1)M_K$$

Assumptions:

(1) Recency("Command Line") performed in a generic memory retrieval operator cycle time.

(2) <STEP MENU> key is a cursor key available on the keyboard. The time to press this key is equivalent to the time to press any other key on the keyboard, M_K.

History Completion on Command Name

C_M	Get-StepMenu (<ESC>)
C_E	Command-Name ← Get-Name("Command Name")
	Count ← 1
	REPEAT BEGIN
C_E	Initiate (Command-Name[Count])
M_K	Key (Command-Name[Count])
	Count ← Count + 1
	UNTIL Is-Equal?(Count, $\bar{n}_0$)
C_E	IF-SUCCEEDED Is-Equal? (Count, $\bar{n}_0$) THEN BEGIN
	REPEAT BEGIN
C_E	Initiate (<ESC>)
M_K	Key (<ESC>)
S	Wait-System-Response(< ESC>)
P_I	Command-Line ← Perceive-Text ("Command Line")
P_{S_1}	Match-Result ← Match (Command-Line)
	UNTIL Found? (Match-Result)
C_E	IF-SUCCEEDED Found? (Match-Result) THEN BEGIN
C_E	Initiate (<RETURN>)
M_K	Key (<RETURN>)
	END
	END

$$C_E + \bar{n}_0(C_E + M_K) + C_E + C_M + e(C_E + M_K + S + P_I + P_{S_1}) + C_E + C_E + M_K$$

$$(\bar{n}_0 + 4)C_E + (\bar{n}_0 + 1)M_K + C_M + e(C_E + M_K + S + P_I + P_{S_1})$$

$$C_M + e(S + P_I + P_{S_1}) + (\bar{n}_0 + e + 4)C_E + (\bar{n}_0 + e + 1)M_K$$

Assumptions:

(1) <ESC> key is a cursor key available on the keyboard. The time to press this key is equivalent to the time to press any other key on the keyboard, M_K.

(2) e items are assumed to be examined.

Typing

```
C_E   Command-Name ← Get-Name("Command Name")
      Count ← 1
      REPEAT BEGIN
C_E       Initiate (Command-Name[Count])
M_K       Key (Command-Name[Count])
          Count ← Count + 1
      UNTIL Is-Equal?(Count, n̄_0)
C_E   IF-SUCCEEDED Is-Equal? (Count, n̄_0) THEN BEGIN
C_E       Argument ← Get-Next-Word ("First Argument")
C_E       IF-SUCCEEDED An-Argument? (Argument) THEN BEGIN
              REPEAT BEGIN
C_E               Initiate (<SPACE>)
M_K               Key (<SPACE>)
                  Count ← 1
                  REPEAT BEGIN
C_E                   Initiate (Argument[Count])
M_K                   Key (Argument[Count])
                      Count ← Count + 1
                  UNTIL Is-Equal?(Count, n̄_A)
C_E               IF-SUCCEEDED Is-Equal? (Count, n̄_A) THEN BEGIN
                      A ← A + 1
                  END
C_E               Argument ← Get-Next-Word ("Next Argument")
              UNTIL No-Argument? (Argument)
C_E           IF-SUCCEEDED No-Argument? (Argument) THEN BEGIN
C_E               Initiate (<RETURN>)
M_K               Key (<RETURN>)
              END
          END
C_E       ELSE IF-SUCCEEDED No-Argument? (Argument) THEN BEGIN
C_E           Initiate (<RETURN>)
M_K           Key (<RETURN>)
          END
      END
```

$$C_E + \bar{n}_0(C_E + M_K) + C_E + C_E + C_E + p_a(C_E + C_E + M_K) + (1 - p_a)(a(C_E + M_K + \bar{n}_A(C_E + M_K) + C_E + C_E) + C_E + C_E + M_K)$$

$$4C_E + \bar{n}_0(C_E + M_K) + 2C_E + M_K + a(1 - p_a)(3C_E + M_K + \bar{n}_A(C_E + M_K))$$

$$(6 + \bar{n}_0)C_E + (1 + \bar{n}_0)M_K + a(1 - p_a)((3 + \bar{n}_A)C_E + (1 + \bar{n}_A)M_K)$$

Assumptions:

(1) No mental retrieval operator is required in this task as the *command* is in a user's working memory.

(2) $\overline{n}_0$ is the average command *name* length.

(3) $\overline{n}_A$ is the average command *argument* length.

Appendix F

Task Descriptions for Experimental Tasks

Recall By Argument

P_I	Task ← Perceive-Task ("Task Request")
C_{CRT}	ChoiceReactionTime(Task)
C_M	Get-!? (<!?>)
C_E	Initiate (<!>)
M_K	Key (<!>)
C_E	Initiate (<?>)
M_K	Key (<?>)
C_M	Argument ← Get-One-Argument("Argument")
	Count ← 1
	REPEAT BEGIN
C_E	Initiate (Argument[Count])
M_K	Key (Argument[Count])
	Count ← Count + 1
	UNTIL Is-Equal?(Count, n_A)
C_E	IF-SUCCEEDED Is-Equal? (Count, n_A) THEN BEGIN
C_E	Initiate (<RETURN>)
M_K	Key (<RETURN>)
	END

$$P_I + C_{CRT} + C_M + C_E + M_K + C_E + M_K + C_M + n_A(C_E + M_K) + 2C_E + M_K$$

$$P_I + C_{CRT} + 2C_M + (n_A + 4)C_E + (n_A + 3)M_K$$

Assumptions:

(1) A mental retrieval is required to determine a suitable argument in the *command* which will retrieve the most recent copy of the desired *command* .

(2) The whole *argument* must be provided and not a substring of it.

(3) n_A is the length of a command's *argument* .

Popup Menu

P_I	Task ← Perceive-Task ("Task Request")
C_{CRT}	ChoiceReactionTime(Task)
C_M	Get-PopUpMenu (<MOUSE>)
C_E	Initiate (<MOUSE>)
M_H	Home (<MOUSE>)
C_E	Initiate (<DRAG>)
$M_{D_M}(l)$	Drag ("Matching Item")
S	Wait-System-Response(< MENU >)
$P_S(l)$	Match ("Item at Position l")

$$P_I + C_{CRT} + C_M + C_E + M_H + C_E + M_{D_M}(l) + S + P_S(l)$$

$$P_I + C_{CRT} + C_M + 2C_E + M_H + M_{D_M}(l) + S + P_S(l)$$

Assumptions:

(1) In estimates of Drag("Item at Position l"), the time takes into account the button down to cause menu to appear, drag through to the desired item, and then release of button to make selection.

Single Step Buffer

P_I	Task ← Perceive-Task ("Task Request")
C_{CRT}	ChoiceReactionTime(Task)
C_M	Get-StepMenu (<STEP MENU>)
P_{S_0}	Prep-Search (<STEP MENU>)
	REPEAT BEGIN
C_E	Initiate (<STEP MENU>)
M_B	Key (<STEP MENU>)
P_I	Command-Line ← Perceive-Text ("Command Line")
P_{S_1}	Match-Result ← Match (Command-Line)
	UNTIL Found? (Match-Result)
C_E	IF-SUCCEEDED Found? (Match-Result) THEN BEGIN
C_E	Initiate (<RETURN>)
M_K	Key (<RETURN>)
	END

$$P_I + C_{CRT} + C_M + P_{S_0} + l(C_E + M_B + P_I + P_{S_1}) + C_E + C_E + M_K$$

$$P_I + C_{CRT} + C_M + P_{S_0} + lP_{S_1} + lP_I + (l + 2)C_E + lM_B + M_K$$

$$P_I + C_{CRT} + C_M + P_S(l) + lP_I + (l + 2)C_E + lM_B + M_K$$

Assumptions:

(1) In the experiment, the subject's hands were already positioned over the keys. The keyboard interaction is limited primarily to pressing the <STEP MENU> key. Thus, the time to press the <STEP MENU> key is the time to press a button, M_B.

(2) However, the time to press the <RETURN> key is equivalent to the time to perform a keystroke operator, M_K

Typing

```
P_I     Task ← Perceive-Task ("Task Request")
C_CRT   ChoiceReactionTime(Task)
C_E     Command-Name ← Get-Name("Command Name")
        Count ← 1
        REPEAT BEGIN
C_E             Initiate (Command-Name[Count])
M_K             Key (Command-Name[Count])
                Count ← Count + 1
        UNTIL Is-Equal?(Count, n_0)
C_E     IF-SUCCEEDED Is-Equal? (Count, m) THEN BEGIN
                A ← 1
                REPEAT BEGIN
C_E                     Initiate (<SPACE>)
M_K                     Key (<SPACE>)
C_E                     Argument ← Get-Argument ("Next Argument")
                        Count ← 1
                        REPEAT BEGIN
C_E                             Initiate (Argument[Count])
M_K                             Key (Argument[Count])
                                Count ← Count + 1
                        UNTIL Is-Equal?(Count, n_A)
C_E                     IF-SUCCEEDED Is-Equal? (Count, n_A) THEN BEGIN
                                A ← A + 1
                        END
                UNTIL Is-Greater? (A, a)
C_E             IF-SUCCEEDED Is-Greater? (A, a) THEN BEGIN
C_E                     Initiate (<RETURN>)
M_K                     Key (<RETURN>)
                END
        END
```

$$P_I + C_{CRT} + C_E + n_0(C_E + M_K) + C_E + \left(\sum_{A=1}^{a}(C_E + M_K + C_E + n_A(C_E + M_K) + C_E) + C_E + C_E + M_K\right)$$

$$P_I + C_{CRT} + (1 + n_0 + 1 + a + a + \sum_{A=1}^{a} n_A + \mathrm{a} + 2)C_E + (n_0 + a + \sum_{A=1}^{a} n_A + 1)M_K$$

$$P_I + C_{CRT} + (4 + 3a + \sum_{A=0}^{a} n_A)C_E + (1 + a + \sum_{A=0}^{a} n_A)M_K$$

Assumptions:

(1) n_0 is the length of a command *name* .

(2) n_A is the length of a command *argument* .