Journal of the Korean Institute
of Industrial Engineers
Vol. 19, No.3, September., 1993.

프로그래밍 언어가 소프트웨어 복잡도에 미치는 영향에 관한 연구 : FORTRAN IV와 FORTRAN 77, PASCAL과 C의 비교

A Study on the Effects of Programming Languages on Software Complexity : Comparison of FORTRAN IV vs. FORTRAN 77 and PASCAL vs. C

윤 정 모*

Jung-mo, Yoon*

Abstract

This paper presents the results of experiments which compare the software complexity between programming languages, i.e, FORTRAN IV and FORTRAN 77, PASCAL and C language, respectively. Each experiment is performed to compare the complexity between programs of same problems using Halstead's method based on operator, and McCabe's based on data flow. As the results of 25 test programs experiments, FORTRAN 77 languages shows superiority to FORTRAN IV languages, and C than that of PASCAL languages, in the aspect of the global software complexity.

1. Introduction

The term, software engineering, is firstly used to discuss the problem of software technique at the workshop in Rome in 1969 according as the field of utilization for the computer is amplified and complicated. Especially a research is actively performed to measure the quality of software by the quantitative method[1]. Namely the state of research for software metrics can be presented as Figure 1. Applying to the software life cycle like this quantitative method may affect greatly to the improvement of software quality and cost reduction[2]. The fact above mentioned, can be proved that the research announcement is active because of the general recognition of the important system management in the computer field.

Up-to-date according to analyzers, as Figure 2, among importance of the software complexity metrics over 70% of system management activity is provided to software maintenance and management[2,3]. Absolutely, if the complexities of program can be

* 국립서울산업대학교 전자계산학과

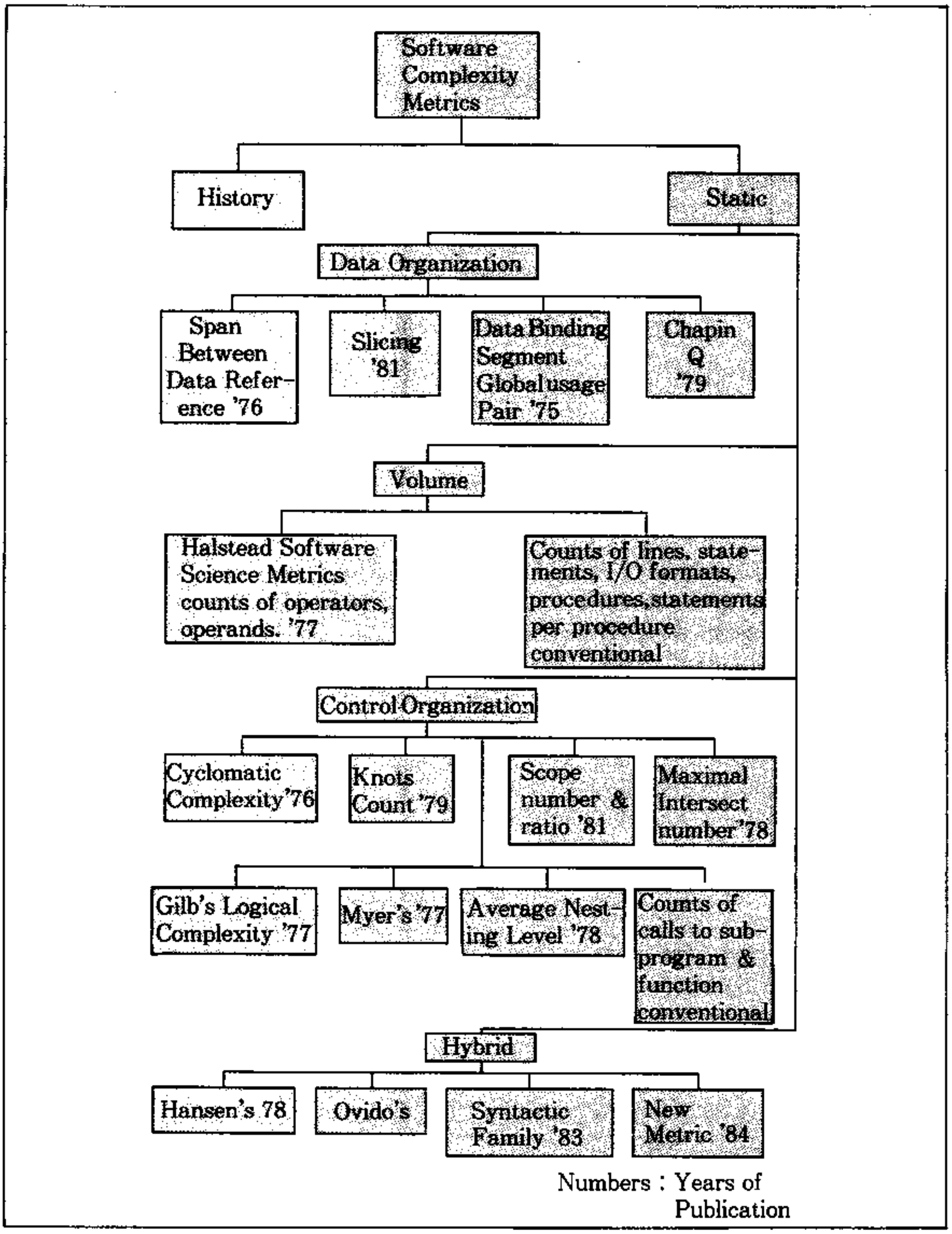

Software Complexity Metrics
History
Static
Data Organization
Span Between Data Reference '76
Slicing '81
Data Binding Segment Globalusage Pair '75
Chapin Q '79
Volume
Halstead Software Science Metrics counts of operators, operands. '77
Counts of lines, statements, I/O formats, procedures, statements per procedure conventional
Control Organization
Cyclomatic Complexity'76
Knots Count '79
Scope number & ratio '81
Maximal Intersect number'78
Gilb's Logical Complexity '77
Myer's '77
Average Nesting Level '78
Counts of calls to sub-program & function conventional
Hybrid
Hansen's 78
Ovido's
Syntactic Family '83
New Metric '84
Numbers : Years of Publication

Figure 1. Classification of Complexity Metrics.

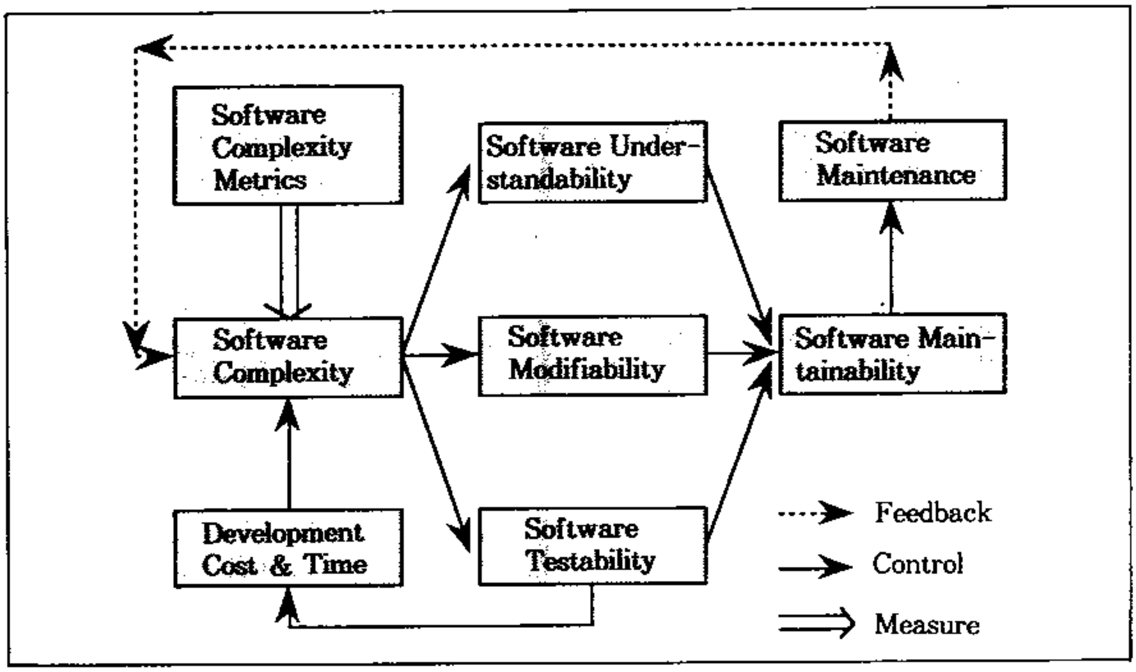


Figure 2. Importance of Software Complexity.

firmly recognized, cost and time to maintain, to repair, and to manage may be reduced.

The purpose of this study, in the problem of software complexity, is to offer problems from comparison and analysis of the complexity between different languages. In this study, comparisons between FORTRAN Ⅳ and FORTRAN 77, and between PASCAL and C language, using the Halstead's method are performed, which is proved the superioity in the existing measure of the software complexity at the recent conferences[4,5]. In order to achieve the purpose of this study, the form of the analysis by the research experimental design is mainly used.

2. Analysis and problems of existing software complexity

2.1 Complexity measurement in accordance with program size

(1) Halstead's Software Science Metrics [6]

As Halstead's method is based on the size, its superiority to various possible estimated-characteristics for the program has been proved; e.g., line of code, program length, development effort, language level, number of paragraph, and problem difficulties.

In order to develop a primitive code oriented to the complexity measurement, we can use the value of a basic measurement after a software design is finished and produced. This value is used to formulate an equation to calculating the total program length. In this case, the equation depends on the number of operators n1, the number of operands n2, and the total number of operators N1, the total number of operands N2. This is a method of using operator and operand as the components of execution code.

It has tried to measure the scale of software with the number of token, not line, and discovered the difference of size in each program and program language. In · the Halstead's method, the complexity of a program is not related to the data flow and control flow in the program, but to the number of operators and operands.

2.2 Complexity measurement in accordance with the control flow of the program

(1) McCabe's Cyclomatic Complexity[5]

McCabe defined the cyclomatic number, V(G), as the complexity based on the control flow of a program, mathematically.

$$V(G) = e - n + 2p \qquad (1)$$

where, e : number of edges,
n : number of nodes,
p : number of connection components.

As the value of V(G) is increasing, the more program control flow is complicated. V(G) is independent with the external size of program or the paragraph layout order, but it is only decided by the number of control flows.

As Figure 3 shows a graph of the control flow with 9-edges and 6-nodes, the value of V(G) is 5. Supposing that a value is returned from outlet-node, f, to in-node, a, in connection with edge to strongly connected graph, the graph G is the same as lineally independent circuit number.

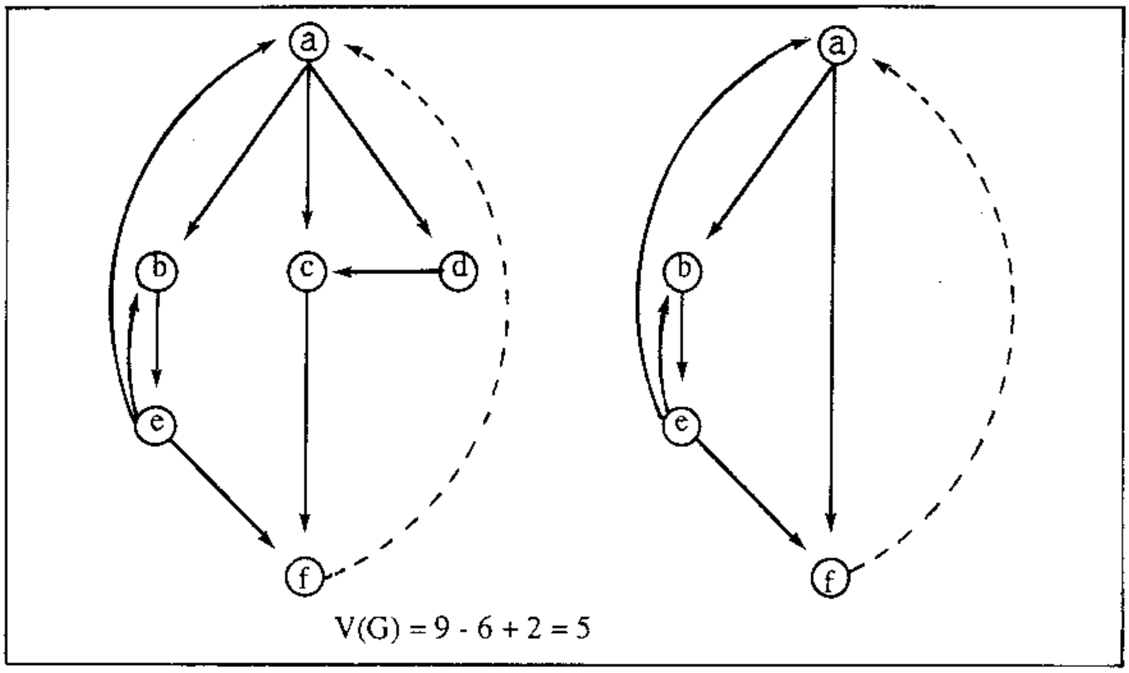


Figure 3. Control Flow Graph and Complexity.

(2) Knots Count

Knots Count is a measure based on the number of structures in the physical position of the control flow, then the number of measurement of the control structure is the same as Figure 4[7].

(3) Moreword's W measurement

```
   IF (CON.NE.0) GO TO 20       (a)
   IF (CON.LT.COT) GO TO 10     (b)
   TOT=1
   GO TO 40                     (c)
10 TTOT=0
   GO TO 40                     (d)
20 IF (CON.ST.TER) GO TO 30     (e)
   TTOT=1
   GO TO 40                     (f)
30 TTOT=0                       (g)
40 IF(TTOT.LE.1) GO TO 50       (h)
   JTOT=JTOT+1
   JVOT=JVOT+LA+1               (i)
50 STOP                         (j)
   END
```

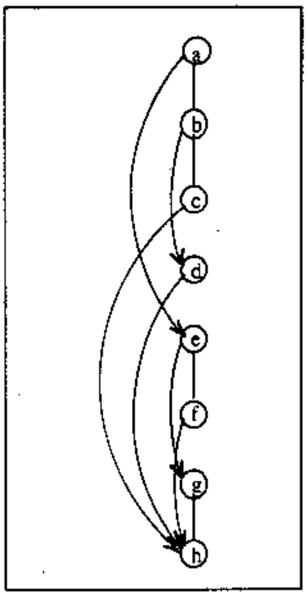


Figure 4. Measure of Knots Count.

Moreword's W measure is an improved one of McCabe's measure, and this is based on the Loop in the program which affects the complexity[7].

Moreword's W, is defined as the sum of the number of Loops and cyclomatic number in the control flow graph.

Therefore, it can be expressed as follows :

$$W = V(G) + \text{Number of Loops} \qquad (2)$$

2.3 Data Structure of program and Complexity measurement in accordance with Data Flow

(1) Span in Data Reference

This method is based on the measurement of data reference in the program. The span means the number of paragraphs in the identifier in case of two same references without any intervention of different references.

(2) Chapin's Q measurement[8]

This measurement is based on how to use the data items in a program. The data can be classified into 4 types as follows :

① P data classification : Input data necessary to produce segment,

② M data classification : Producing or changing data in the segment,

③ C data classification : Using data to "Controlling" in the segment,

④ T data classification : Data unchanged and using in the segment.

This method is based on the data classification which affects the complexity of the program with different densities.

(3) **Henry's measurement of the Information flow[9]**

This is the complexity measurement based on information flow in system components.

$$INFO = Length * (Fan-in * Fan\text{-}out) ** 2 \quad (3)$$

where, INFO : Information flow complexity,

Length : Number of source codes,

Fan-in : Number of local flows into Module A+number of data structures from which Module A retrives information,

Fan-out : Number of local flows from Module A+the number of data structures which Module A updates.

When these conditions are occurred, it defines that there is a sectional flow from Module A to Module B.

① When A calls B.

② When B calls A, so B uses the value of A at the later stage.

③ When the optional C calls A and B so it can transmit output-value of A to B.

It is possible for Henry's et al. complexity metric to measure interface complexity of module, but impossible to measure the logic of module itself.

2.4 Measurement by hybrid method

Hybrid complexity measurement method is used to prevent the weak points when using only one componet in the complexity measurement by providing more than two types of the components; e.g, the program size, dta structure of the program and data flow, and control flow of the program[2].

It is Ramamurity's et al. weighted metric to analyze both the size of the program and control flow simultaneously. Combining McCabe's V(G)[5] and Halstead's[6] software science techniques can actually supplement their weak points. Based on this new nesting level, this can impose an extra weighted value to the operator and the operand of control text.

$$NW1 = [1 + d(x) * L(x)]x \text{ ; Operator} \quad (4)$$

$$NW2 = [1 + d(x) * L(x)]x \text{ ; Operator} \quad (5)$$

In case of using x as a part of the control text, $d(x) = 1$ or 0. In accordance with nesting level of the control text $L(x)$ is increasing an extra weighted value one by one from 1.

$$VW = (NW1 + NW2) * Log2(n1 + n2) \quad (6)$$

$$EW = VW * [(n1 * NW2)/(2 * n2)] \quad (7)$$

It is supposed that weighted metric provides reliable EW value but its usefulness may be used only at the stage of maintenance and repair.

There are Hansen's measurement considering control flow and the size of the program or Ramamurity's et al. weighted metric, Ovido's measurement of hybrid form with data flow and control flow.

3. An examples

3.1 Halstead's Software Science Metrics[6]

It is one of the most widely accepted measures with several empirical studies. It involves metrics defined by some key constituents of a program implementing an algorithm.

(1) Operators & Operands : An algorithm consists of operators and operands, and of nothing else. Operators fail into three classes :

(a) Basic : + − * ** / // = ( ). GT. .GE..LT..LE..NE..EQ..NOT..AND..OR..EQV. .XOR..NEQV.

(b) Keyword : IF THEN ELSE ELSEIF ENDIF DO DOWHILE GOTO ASSIGN CONTINUE ENDDO READ WRITE TYPE PRINT ACCEPT EOS

(c) Special : Names of subroutines, functions.

Operands consist of all variable names and constants such as, .TRUE. .FALSE. and Esnn(real). Then Halstead's metrics can be defined based on the number of distinct operations n1, the number of distinct operands n2, the total number of operators N1, and the total number of operands N2.

(2) Derived Metrics : Halstead defines the vocabulary of the program as $n=n1+n2$ (the total number of distinct operators and operands) and the implementation length as $N=N1+N2$. These are meaningful volume measures of a program. He hypothesizes an estimator $N\hat{}=n1\mathrm{Log}_2\, n1+n2\mathrm{Log}_2\, n2$ for N. A program volume metric V defined as $N\mathrm{Log}_2\, n$ characterizes the size of an implementation, which can be regarded as the number of bits necessary to encode the whole module.

To evaluate the programming effort, propensity of error, and ease of understanding, the program level L of an implementation is defined as $(2/n1) * (n2/N2)$. It follows that only the most succinct expression can have a level of unity. Program difficulty D is the difficulty of coding an algorithm. and defined as $D=1/L$, and can be estimated by $D2=1/L\hat{}$. The languages level is $L**2/V$. The effort required to generate an algorithm is $E=V/L$.

For the smaple program in Figure 5, Halstead's software science metrics are exemplified in Table 1.

Therefore, the results of Halstead's software science metrics can be expressed as follows :

```
C   SUM OF 1 THRU 10000
    N = 0
    SUM = 0.0
 10 N = N + 1
    SUM = SUM + N
    IF (N .LT. 10000) GO TO 10
    WRITE (6,20) N, SUM
 20 FORMAT (//,' SUM OF 1 THRU ',
  *            I5,'IS ',F10.0)
    STOP
    END
```

Figure 5. A sample program

Table 1. The Results of Halstead's Software Science metrics

No.of Operator	Number	No.of Operands	Number
=	4	N	6
+	2	0	2
IF	1	SUM	4
.LT.	1	1	1
GOTO	1	10000	1
WRITE	1	10	2
STOP	1		
END	1		
,	1		
n1=9	N1=13	n2=6	N2=16

where, $n=n1+n2=9+6=15$ (8)

$N=N1+N2=13+16=29$ (9)

$N\hat{}=n1Log_2n1+n2Log_2n2$ (10)
$=9Log_29+6Log_26=28.53+15.51=44.04$

$V=NLog_2n=29Log_215=113.30$ (11)

$L=(2/n1)*(n2/N2)=2/9*6/16=0.083$ (12)

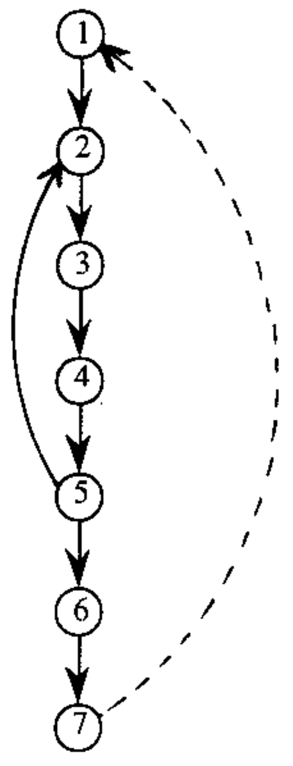
$D=1/L=1/0.083=12.05$ (13)

$\lambda=L**2V=(0.083)**2*113.30=0.781$ (14)

$E=V/L=113.30/0.083=1365.06$ (15)

3.2 McCabe's Cyclomatic Complexity[5]

McCabe's cyclomatic complexity is well accepted, intuitively reasonable, and easily calculated by $V(G)=e-n+2p$. In a strongly connected graph, this cyclomatic number is the number of linear independent circuits. For programs with single entry and single exit, V(G) is one plus the number of decisions. The graph theoretic metric is independent of the program size but depends only on the decision structure. Decision making of a program affects its error probability and development time and cost. For the sample program, the results of McCabe's cyclomatic complexity metrics are shown in Figure 6.



$V(G)=e-n+2p$

$=7-7+2=2$

Figure 6. The Results of McCabe's Cyclomatic Complexity Metrics

4. Experiments and results

The languages level of Assembler, Algol58, Algol68, FORTRAN, PL/I, PASCAL, APL and Basic etc., are shown in Table 2 [6].

As offered in the preface, after making a program individually, for the same problem (program volume etc.) with FORTRAN IV, FORTRAN 77, PASCAL and C language, and making 25 programs(numerical progression, array, sort etc.) by investigated meth-

Table 2. The Languages Level(λ) of each Language

Languages	Average of Lambda
PASCAL	2.54
APL	2.42
ALGOL 68	2.12
PL/I	1.53
ALGOL 58	1.12
FORTRAN	1.14
ASSEMBLY	0.88
BASIC	0.81

od, one of software complexity measurement methods, with Halstead's method and then compared, measured measurement of each factor and sought its connected correlation.

4.1 Case Ⅰ : FORTRAN Ⅳ vs. FORTRAN 77 languages[1]

As you see at Table 3, it is decreased as much as 17.25% for the program line number of FORTRAN 77 than that of FORTRAN Ⅳ, and decreased as much as 11.2% for the program volume(V) of FORTRAN 77. The program level of FORTRAN 77 is decreased as much as 1.53%, and language level(λ) decreased as much as 8%.

And, as you see at Table 4, it showed high correlation in the part of the program line number(r=0.843), number of program volume(r=0.946), estimated length(r=0.331) etc., between FORTRAN Ⅳ and FORTRAN 77 showed low correlation in the part of the program languages, language level etc.

Correlation between FORTRAN Ⅳ vs. FORTRAN 77 in the same problem is :

- Program Length : 0.843
  Correlation : very high
  Length : decreased 17.25%
- Program Volume(V): 0.946
  Correlation : very high
  Number : FORTRAN 77 is 11.2% higher than that of FORTRAN Ⅳ
- Language Level(λ) : 0.331
  Correlation : low

I showed that correlation to the program volume(r=0.946) has almost nothing to do with. With each factor value by using the Regression of SPSS the first regression equation as follows[20] :

- Program line Number(LC)
  $Y=0.6218X+3.2970$ (16)
- Program Volume(V)
  $Y=0.8390X+19.090$ (17)
- Program Level(L)
  $Y=0.9686X+0.0008$ (18)
- Language Level(λ)
  $Y=0.5036X+0.3007$ (19)

Table 3. Average and standard deviation for each measure (FORTRAN Ⅳ vs. FORTRAN 77 language)

Language	FORTRAN Ⅳ		FORTRAN 77		Remarks
Metrics	Average	Standard Deviation	Average	Standard Deviation	F77/FⅣ
LC	16.033	5.4920	13.266	4.049	17.25% ↓
n	22.433	5.1610	23.333	5.491	4.01% ↑
N	62.733	30.206	56.033	25.998	10.68% ↓
N^	78.039	24.988	84.988	27.090	8.90% ↑
V	287.09	154.77	259.94	137.43	11.20% ↓
L	0.0552	0.2207	0.0543	0.2750	1.53% ↓
D	20.869	7.3280	21.584	7.3649	3.43% ↑
λ	0.7147	0.2940	0.6607	0.4466	8.00% ↓
E	6831.5	5417.5	6211.5	4479.9	9.08% ↓
T	379.52	300.96	345.08	248.88	9.07% ↓
V(G)	3.2000	1.2220	2.8666	1.0973	11.00% ↓

Table 4. Correlation table between each measure(FORTRAN Ⅳ vs. FORTRAN 77 Language)

FIV / F77	LC	n	N	N^	V	L	D	λ	E	T	V(G)
LC	0.843										
n		0.886									
N			0.946								
N^				0.945							
V					0.946						
L						0.777					
D							0.754				
λ								0.331			
E									0.869		
T										0.869	
V(G)											0.873

where X : each factor of FORTRAN Ⅳ,
Y : each factor of FORTRAN 77.

4.2 Case Ⅱ : PASCAL vs. C language

As you see at Table 5, it is decreased as much as 12.5% for the program line number of C than that of PASCAL, and decreased as much as 5.8% for the program volume (V) of C. The program level of C is decreased as much as 47.7%, and language level(λ) decreased as much 70.2%.

And, as you see at Table 6, it showed high correlation in the part of the program line number(r=0.809), number of vocabulary(r =0.826), estimated length(r=0.810) etc., between C and PASCAL etc., showed low correlation in the part of the program language, language level etc.

Correlation between C and PASCAL in the same problem is :

Table 5. Average and standard deviation for each Measure(PASCAL vs. C)

Language	PASCAL		C		Reamarks
Metrics	Average	Standard Deviation	Average	Standard Deviation	Incr./Dec.
LC	24.00	9.691	21.00	11.314	12.5% ↓
n	16.20	5.418	18.12	5.033	10.8% ↑
N	75.08	41.275	70.68	63.883	5.8% ↓
N^	17.41	8.637	20.69	7.700	15.8% ↑
V	115.05	69.923	108.34	89.184	5.8% ↓
L	0.088	0.023	0.046	0.059	47.7% ↓
D	16.737	18.007	28.57	11.077	41.4% ↑
λ	0.582	0.095	0.173	0.512	70.2% ↓
E	2816.50	4397.70	3804.64	3894.90	25.9% ↑
T	156.47	242.98	211.37	216.38	25.9% ↓

Table 6. Correlation table between each measure(PASCAL vs. C Language)

PAS / C	LC	n	N	N^	V	L	D	λ	E	T
LC	0.809									
n		0.826								
N			−0.259							
N^				0.810						
V					−0.117					
L						−0.303				
D							−0.382			
λ								−0.303		
E									−0.258	
T										−0.258

- Program Length(LC) : 0.843
  - Correlation : very high
  - Length : decreased 12.5%
- Vocabulary Number(n) : 0.826
  - Correlation : very high
  - Number : C is 10.8% higher than that of PASCAL
- Language Level(λ) : −0.303
  - Correlation : very low

I showed that correlation to the program volume($r=-0.1170$) has almost nothing to do with. With each factor value by using the Regression of SPSS the first regression equation as follows[11] :

- Program Line Number(LC)

  $Y=0.9440X+4.6939$ (20)
- Program Volume(V)

  $Y=-0.1492X+131.2252$ (21)
- Program Level(L)

  $Y=-0.7783X+0.12569$ (22)
- Language Level(λ)

  $Y=-1.7364X+0.881$ (23)

where X : each factor of PASCAL,
Y : each factor of C.

5. Conclusion

In this paper, the software complexity of program languages is analyzed and compared using Halstead's measures.

(1) As a result of the experiment, FORTRAN 77 showed the decrease more in program line by 17.25%, in Halstead's volume by 11.2% and in McCabe's V(G) by 11%, than those of FORTRAN Ⅳ anguage. The program level of FORTRAN 77 is lowered as much as 1.53%[1].

Also, it is decreased as much as 12.5% in program line, 5.8% in Halstead's volume and 5.8% in program length of C than that of PASCAL language. The program level of C is lowered as much as 47.7%

(2) This study, couldn't present a perfect estimated value due to the level and the short steps of the program, but has reflected the difference in the subjectivity of selecting criteria of operand and operator, and in the program logic flow. Also, the component of C program can be forecasted with the

knowlege of the components of PASCAL program, which can be acquired by the linear regression equation for each measures.

(3) In conclusion, this paper, has calculated and compared the complexity between different languages by using the Halstead's measure for the software complexity. However, an accurate comparision of complexity between languages can be achieved by using the Hybrid measure and/or providing adequate weights to each languages. Also, in accordance with the processing method, type, characteristics of the flow of the program and the proper selection of software, it may be able to improve the quality of software and the program complexity drastically.

References

1. Sung, K.K. and Lee, G.S.(1990), A Study on Effect the Extensional version of Programming Language on Software Complexity, Hannam University, Master Thesis.
2. Li, H.F. and Cheung, W.K(1987), "An Empirical Study of Software Metrics", IEEE Trans. on Software., Vol, SE.13, No.6, pp.697–708.
3. Yang, H.S.Noh, H.Y. and Baek, C.H. (1990), "Study for Software Complexity and Estimation", Korea Information Science Society Review, Vol.8, No.4, pp.89–96.
4. Yang, H.S.Noh, H.Y. and Baek, C.H. (1990), "For Complexity of the Program Considering the Formal Grammar", Korea Information Science Society, Vol. 17, No.2, pp.665–668
5. McCabe, T.J.(1976), "A Complexity Measure", IEEE Trans. on Software Engineering, Vol.SE–2, No.4, pp.308–320.
6. Halstead, M.(1977), Element of Software Science, Elsevier, North-Holland.
7. Woodward, M.Hennel, M. and Heddey, D. (1979), "A Measure of Control Flow Complexity in Program Text", IEEE Trans. on Software Engineering, Vol.SE –5, pp.45–50.
8. Chapin, N.(1979), "A Measure of Software Complexity", Proceedings of NCC, pp.995–1002.
9. Henry, S.. and Kafura, D.(1981), "Software Structure Metrics Based on Information Flow", IEEE Trans. on Software, Vol.SE.7, No.5, pp.510–519
10. Ovido, E.(1980), "Control Flow, Data Flow and Program Complexity", Proceedings COMPSAC-80, pp.146–152.
11. Jung, Y.J.(1990), SPSS/PC+, Crown Publishing Co.