1981

Software Metrics Data Collection

T. J. Yu

Brian A. Nejmeh

Report Number:

81-421

Software Metrics Data Collection

T.J. Yu and Brian A. Nejmeh

Department of Computer Sciences
Purdue University
West Lafayette, Indiana 47907

CSD-TR-421
(Revised October 1984)

ABSTRACT

This report describes the organization of the Software Metrics Data Collection (SMDC) system. This is an APL-based system which runs on the Purdue University Department of Computer Science Computing System. The system stores data collected from actual products developed at industrial environments and from experiments conducted at Purdue University. It also provides a number of statistical functions and plotting routines which can be used for detailed analysis of existing data. A description of metrics and requirements for collecting additional data are also given.

Contents

1. UNIX is a Trademark of Bell Laboratories

1. Introduction

The ever-increasing cost of software development has made the measurement of software complexity more important than it has ever been before. The critical role that software metrics can play in analyzing and evaluating software products cannot be over-emphasized. However, the usefulness of software metrics should not only be justified theoretically, but should also be supported by empirical results. This is our motivation – to provide a data collection system to help researchers in the investigation of new and existing metrics.

There are widely acknowledged difficulties in acquiring meaningful data. Economic constraints severely limit the extent to which experiments involving significant programming tasks may be conducted. Most commercial producers of software are very hesitant to release product-related information which could, conceivably, compromise their position in the market place. The result is a situation in which researchers in the academic sector have become overly-dependent on the student programmer as a primary source of data, while researchers in industry are typically restricted to data reflecting only the local environment.

Our Software Metrics Data Collection (SMDC) system is an attempt to gather data related to software development into a central location and to provide useful mathematical and statistical functions for manipulating this data. We have acquired data for the system in two ways. One is via our own experiments with students in the Department of Computer Sciences at Purdue University. A good deal of this data is a result of carefully-controlled experimentation in which measurements have been made algorithmically. The disadvantage is that many of the programs involved are quite small. The other kind of data reflecting non-trivial, commercially-developed software are from a number of different environments; many of the contributors wish to remain anonymous. The typical disadvantage of this external data is that generally it is from less-controlled situations and occasionally some desirable data (e.g., programming time) is not recorded.

SMDC is a set of APL workspaces for storing software metrics data. The data is organized into environments (UNIX directories) in a tree structure (see Appendix 2). Each environment contains "related" but independent workspaces. The SMDC user is expected to be familiar with APL and UNIX, as many APL- and UNIX-related terms are used throughout this report.

There are documents describing each workspace. All documents are online and can be accessed by the "describe" function which will be explained below.

2. Summary of Capabilities

2.1 Logging on and off

When you logon to the UNIX system and want to access the software metrics data, simply enter

cd /usr/smdc

smdc

After changing your current directory to /usr/smdc, the command "smdc" puts you into APL, sets the backspace character to Control-U (or "<-"), and loads the workspace "ws.sofmet". A better way to access SMDC is to include the path name **/usr/smdc/bin** in your **.login** file or **.cshrc** file. In this way, you can access SMDC without changing your directory.

The workspace "ws.sofmet" contains the functions "describe", "help", "listdc", "goto", "table", and "cd".

1) describe – provides documentation for each workspace. There are three levels of documentation:

   a) high-level – **describe 'ws.sofmet'**; this gives a brief description of *all* the environments (directories) and workspaces available, sources, quantities and characteristics of each workspace.

   b) mid-level – **describe 'directoryname'** where **'directoryname'** is the name of any directory in the current SMDC system; this briefly describes each of the workspaces in the specified directory.

   c) low-level – **describe 'workspacename'** where **'workspacename'** is the name of any of the several workspaces available; this describes the variables (metrics) and functions in the workspace. It also gives some other characteristics of the workspace.

2) help – prints this document of the five utility functions provided by SMDC.

3) listdc – lists the current hierarchical structure of the data base.

4) goto – This function attempts to remove the user from the burden of specifying path names; **goto 'directoryname'** simply enables one to move to the directory specified and access workspaces without full qualification of workspace names. For example, suppose you wish to run some statistical functions available in "tool.84" on data in "ws.basic". Then, assuming you have entered **cd /usr/smdc** and **smdc**, you would enter

goto 'boehm'

)copy ws.basic

goto 'tool.84'

)copy ws.stat

Your workspace now contains the "basic" COCOMO data from Boehm's text as well as the statistical functions. Note, "goto" only changes directories but does not copy the workspace itself.

5) table – This function provides the user with a table of current workspaces and their associated metrics.

6) cd – This function is same as the UNIX command "cd" which chnages the directory. The only difference is you have to put your path name in quotes as a character string. For example:

cd '/usr/tjy/database'

When you are finished interacting with the SMDC system and desire to log off the system, you may do so by simply entering the APL command

)off

The backspace character is reset to Control-H.

2.2 Metrics Included

A software metric is the count of a certain feature of a software product. The simple features frequently counted by researchers include the number of lines in a program, the number of tokens used by a program, the number of conditional statements, etc. The counts of these basic features are often combined using a formula which is based on a model of the programming process. The resulting value is then hypothesized to reflect certain important properties of the program, such as the effort required to produce the software or the quality of the produced software.

The SMDC system is a collection of data from different sources. In general, each source has its own method for measuring software. Therefore, it is difficult to have a uniform definition for each software metric in the SMDC. Here we discuss each metric using the most common definition.

2.2.1) lines of code (*loc*) – People mean different things when they say "lines of code". Here *loc* means lines of executable statements and declarative statements, excluding comments and blank lines. For many large projects, some code is new and some code is reused. Therefore, we also need to distinguish between the various sources of *loc*:

locnew : refers to all new or modified lines of code.

locbase : refers to lines of code which are unchanged from the previous version.

loccopy : refers to lines of code which are copied or are from another source.

We suggest that *locbase* +*loccopy* be used as a metric for "reused" code.

2.2.2) effort (*pm* or *hour*) – In the SMDC we use two units to measure effort. For large scale projects we measure it in person-months (*pm*), where a person-month is defined to be 160 hours (40 hours / week X 4 weeks). For small and medium size programs, we measure it in hours. Ideally, we would like effort to be measured in a controlled environment in which the subject is working only on the software without distractions and in which time is recorded algorithmically. For each project/program we are also interested in the effort between certain milestones. For small programs these milestones would occur after the end of the specification, design, coding, and testing phases. For large projects the milestones should be the completion of the specification, design, coding, unit testing, integration testing, publication, and performance evaluation phases. We use $r1$, $r2$, ... to represent the fraction of the effort expended between certain milestones.

For security reasons, the effort in some workspaces has been scaled by a constant. Note that this scaling will not affect the correlation coefficients between effort and other metrics. Scaled effort appears as *pms*.

2.2.3) duration(*dur*) – the calendar time during which development effort proceeds without interruption. For a one person project, it is the same as the effort. In general, effort is always greater than or equal to duration.

2.2.4) operators and operands – These are tokens used in composing a program[2]

operators: keywords or symbols which specify an algorithmic action; punctuation marks.

operands: symbols used to represent data including variables, constants, and literals.

In the SMDC system we represent such metrics as

*eta*1 : number of unique operators

*eta*2 : number of unique operands

$n1$: total occurrences of operators

$n2$: total occurrences of operands

2: Halstead M.H. *Elements of Software Science*, Elsevier North-Holland, New York, N.Y. (1977)

2.2.5) Cyclomatic complexity[3] (vg) – This is a count of the conditional statements, loops, procedures (including main program), and binary Boolean operators.

2.2.6) defect (def) – A software defect is hard to define and hard to measure. The defect data collected in SMDC may have different meanings for different environments. Please refer to the documents of the individual workspaces to obtain the meanings of software defect in the various local environments. In general, def is the number of software changes made in response to errors found during formal testing and/or after delivering the product to the customers. Different kinds of software defects were collected; they can be identified by their suffix (such as def 1 or def 2).

Certain workspaces may have only part of these metrics, while others may have more. All the basic metrics are summarized in Appendix 3.

To help users access data efficiently, APL workspaces are kept as small as possible. Thus, only "primitive" software metric data is kept in the SMDC system. Those metrics which can be derived from the primitive software metric data (i.e. Software Science "difficulty") are excluded.

In the SMDC system we define the scale of a program about which data is collected to be *small, medium,* or *large.* The three different program scales are now described.

a. small

1. The size of the program is less than 200 lines of code.
2. The effort required to develop the program is less than one day (8 hours).
3. The development of the program is a one-person project.

b. medium

1. The size of the program is between 200 and 1000 lines of code.
2. The effort required to develop the program is more than one day, but less than a person month (160 hours).
3. Typically, the development of the program is a one-person project.

c. large

1. The size of the program is more than 1000 lines of code.

3: McCabe T.J. *A Complexity Measure*, IEEE Transactions on Software engineering 2,4 (December 1976), 308-320

2. The effort required to develop the program is more than one person-month.

3. Typically, the development of the program is a team project.

2.3 Statistical Functions

In the SMDC system, we provide several statistical functions. They are useful for investigating relationships among data items. Most of the statistical functions are self-explanatory (eg., "mean"). We are almost certain that the functions produce correct output for correct input. However, if the input is incorrect (e.g., arguments in wrong positions or arrays with inconsistent size), strange results (rather than an error message) may be produced. Therefore we suggest the user consider the syntax of each function carefully.

These statistical functions are demonstrated in the next section. There is a short online description which describes the meaning of each function. It can be accessed via **describe 'ws.stat'**. For some complicated functions, SMDC also provides detailed description in the comment section of those functions. It can be accessed via the APL command **)list functionname.**

2.4 Statistical Testing Functions

The SMDC system provides some statistical functions to test statistical hypotheses. These functions include: Kruskal-Wallis one way analysis of variance, ANOVA, ANCOVA, and T-test. They are less frequently used than the functions in "ws.stat", so they are stored separately in the workspace "ws.test" in order to increase the system efficiency. These functions requires some functions that appear in "ws.stat", thus you must copy "ws.stat" before using these functions.

These functions are also demonstrated in the next section. There is a short online description which describes the meaning of each function. It can be accessed via **describe 'ws.test'**. SMDC also provides detailed description in the comment section of these functions. It can be accessed via the APL command **)list functionname.**

2.5 The Linear Regression Function

The SMDC system has the capability of computing all possible linear regressions for a given set of independent variables. You also need to copy "ws.stat" before using this function. It is demonstrated in the next section. There is a short online document describing how to use this function. It can be accessed via **describe 'ws.reg'**. SMDC also provides detailed description in the comment section of this function. It can be accessed via the APL command **)list functionname**.

2.6 Plotting Routines

The SMDC also provides some plotting routines. These routines show graphically the distribution of data on the terminal or in the nroff/troff format. These routines are demonstrated in the next chapter. There is a short online document describing how to use this function. It can be accessed via **describe 'ws.plot'**.

2.7 Software Science Functions

The SMDC system also has the capability of computing several Software Science metrics defined by M.H. Halstead. In particular, their are functions that given eta1, eta2, n1, and n2, can compute length, volume, difficulty, and effort. There is a short online document describing how to use these functions. It can be accessed via **describe 'ws.sofsci'**.

3. Examples of Using the SMDC System

In this section we will show you how to use the SMDC system via a number of examples. To help you differentiate among input, output, and comments, we use three kinds of type fonts:

boldface - commands/data input to the system

normal - output from the system

italics - descriptive comments

3.1 Logging on

After you have entered

cd /usr/smdc

smdc

you will see the following on your terminal:

Welcome to SMDC

erase: < cntrl> u

apl 11

06 may 1982

Software Metrics Research Data

Purdue University VAX 11/780 UNIX

Enter "describe 'ws.sofmet'" for description of data.

Enter "listdc" to list all files in the data base

At this point, you have available to you six functions which were described in section 2.1.

Note that the backspace is changed to Control-U. The normal backspace

(Control-H) is reserved by APL for overstrike characters.

)fns

start describe goto listdc help table

If you enter "describe 'ws.sofmet'", you will get a global description

of the whole system.

If you enter "listdc", you will be told the structure of the system.

Suppose you are interested in the 'ws.sel' data, you can use the following steps:

goto 'indep'

after which you will see a listing of files defined in this directory and

the reminder from SMDC to

enter ")copy xxxxx" to get the workspace

)copy ws.sel

09.24.13 04/26/84 copy ws.sel

in this workspace, if you type in

)vars

you will find the following metrics

dur locnew locall pm

3.2 Examples of Using the Statistical Functions

First, we have to copy the workspace that contains these functions

goto 'tool.84'

after which you will see a listing of files defined in this directory

and the reminder from SMDC to

enter ")copy xxxxx" to get the workspace

)copy ws.stat

23.02.07 07/18/84 copy ws.stat

3.2.1 Basic Statistical Functions

prod ← locnew ÷ pm *to investigate productivity*

mean prod *sample mean of "prod"*

.52959705

(mean prod) num 2 *take only two digits after the decimal point*

.53

med prod *median of "prod"*

.529692633

sd prod *sample standard deviation of "prod"*

.174249508

var prod *variance of "prod"*

.028764844

Note that var is NOT the square of sd. The degree of freedom of sd is (n-1), while the degree of freedom of var is n.

3.2.2 Moving Average

syntax: n mvavg x

semantics: compute the moving average of every consecutive n elements of vector x

prod num 2 *this lists the vector prod*

.68 .42 .53 .50 .55 .60 .85 .61 .33 .43 .93 .38 .54 .72 .57 .46 .32 .34 .32

(2 mvavg prod) num 2

.55 .48 .52 .52 .57 .73 .73 .47 .38 .68 .65 .46 .63 .64 .51 .39 .33 .33

(1 mvavg prod) num 2 *this is the same as prod itself*

.68 .42 .53 .50 .55 .60 .85 .61 .33 .43 .93 .38 .54 .72 .57 .46 .32 .34 .32

((ρ prod) mvavg prod) num 2 *this is the same as the mean*

.53

3.2.3 Correlation Coefficients

a) Pearson correlation coefficient

locnew cor pm

.899744232

pm cor locnew *it should be the same*

.899744232

b) Spearman rank correlation coefficient

locnew spear pm

.926315789

locall spear pm

.914035088

c) multiple variables formatter

syntax: n mform var_1, var_2, ... var_n

semantics: if the rank of each variable is r, mform will return a $(r \times n)$ *matrix. The ith column corresponds to the variable* var_i.

application: This function is to serve those function with multiple input variables.

y ← 3 mform prod, pm, dur *format 3 variables*

x ← 5 mform prod, pm, dur, locnew, locall *format 5 var...s*

d) Pearson correlation matrix

y cormat x *compute the correlation matrix*

cortbl num 3 *the result is stored in cortbl*

prod	*pm*	*dur*	*locnew*	*locall*	
1.000	-.194	-.096	.095	.046	*prod*
-.194	1.000	.592	.900	.871	*pm*
-.096	.592	1.000	.469	.381	*dur*

3.2.4 Linear Regression

a) regression through origin

pm lin0 locnew

1.88347325 *i.e. pm = 1.88 locnew*

b) slope and intercept

pm slope locnew

1.7483875

pm intcp locnew

6.69904306 *i.e., pm = 6.70 + 1.75 locnew*

locnew slope pm

.463020744

locnew intcp pm

1.94727702 *i.e., locnew = 1.95 + 0.46 pm*

c) Simple Linear Regression

pm mlin locnew *note the order of the parameters*

coefficients:

Y = b0 + b1 X1 + ... + bn Xn

6.69904306 *i.e. pm = 6.70 + 1.75 locnew*

1.74838750

Anova: Source of Variation, df, ss, ms

regression	1	35538.1249	35538.1249
error	17	8361.05094	491.826526

R-square: .809539683 *the coefficient of multiple determination*

d) Multiple Linear Regression

You have to set the input parameters carefully to avoid a syntax error.

You can also use "mform" to format your multiple independent variables.

y ← pm

x ← 2 mform locnew, locall *x is a 19 by 2 matrix*

y mlin x

coefficients:

Y = b0 + b1 X1 + ... + bn Xn

4.89749730 *i.e. pm = 4.90 + 1.29 locnew + 0.37 locall*

1.28696800

.37086491

Anova: Source of Variation, df, ss, ms

regression	2	35928.4308	17964.2154
error	16	7970.74508	498.171567

R-square: .818430644

"mlin" will generate a global variable called "yhat" which is the predicated value of the dependent variable of the regression model. You can use the function "compare" to compare the actual value with the predicated value of the regression model.

pm compare yhat *note the order of the parameters namely, the actual value appears first and the predicted value appears second*

number of data points	19
mean relative error	-.239
mean magnitude of relative error	.37
% of prediction within 25%	.421
mean square error	442.819171
correlation coefficient	.905
spearman rank cor. coef.	.944

3.2.5 Set Functions

a ← ι 10

b ← 5 + a

a

1 2 3 4 5 6 7 8 9 10

b

6 7 8 9 10 11 12 13 14 15

a union b

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

a intersec b

6 7 8 9 10

a differ b

1 2 3 4 5

b differ a

11 12 13 14 15

3.2.6 Others

a) domain- compute the domain of a given vector

domain locnew

Length:	19
Low:	2.052
Low+1:	2.451
Mean:	26.5098421
Median:	12.227
Std. Dev:	25.4140165
High-1:	76.883
High:	84.729

b) which - get the index of a number in a vector

locnew which 2.052

12 *i.e. locnew[12] = 2.052*

3.3 Statistical Testing Functions

We are also using the "ws.sel" data to demonstrate these functions.
The workspace containing these functions is "ws.test" which calls some functions in "ws.stat", so you have to copy both workspaces.

goto 'indep'

)copy ws.sel

goto 'tool.84'

)copy ws.stat

)copy ws.test

3.3.1) factor level formatter

syntax: n gpind metric

semantics: (1) Divide x into n groups by the rank of "metric"

(2) The number of points in each group are same

(3) Return the group ID for each point in "metric"

The smaller the value of x, the smaller the ID number.

ind ← 3 gpind locall *three levels of program size*

(ind=1)/locall *small size*

14.282 5.497 4.525 9.727 2.052 5.204 10.172

(ind=2)/locall *medium size*

55.237 50.911 14.863 32.822 14.765 17.271

(ind=3)/locall *large size*

111.868 75.393 75.420 85.369 67.325 66.266

3.3.2) Kruskal-Wallis one-way analysis of variance

pm kw ind

This is to test the effect of the factor "ind"

on the rank of the programming effort.

Kruskal-Wallis H: 14.9684211

df = 2

Average Rank of Each Group : 16 9 4

3.3.2) Single-factor one-way analysis of variance (ANOVA)

pm anova ind

This is to test the effect of the factor "ind"

on the programming effort.

Group Mean : 9.69 50.03 106.65

Group Std. Dev. : 3.686 35.437 37.646

Group Population: 7 6 6

Anova Table (Single factor)

source	ss	df	ms
Between	30452.9078	2	15226.4539
Error	13446.268	16	840.391753

F = 18.118281

3.3.3) Single-factor analysis of covariance (ANCOVA)

x ← 2 mform ind, dur

The independent variables of "ancova" must be a two-column array.

The first column is the factor to be analyzed.

The second column is the concomitant variable.

"ancova" can only accept one concomitant variable.

pm ancova x

This is to test the effect of the factor "ind"

after eliminating the effect of duration.

Ancova Table

source	y	x	xy	df
Treatment	30452.9078	44.8818437	1124.61165	2

Error	13446.268	290.638475	1148.38339	16
Total	43899.1759	335.520319	2272.99504	18

Adjusted

source	ss	df	ms
Treatment	19591.9597	2	9795.97987
Error	8908.72561	15	593.915041

F = 16.4939077 *This should be less than the F-value of ANOVA*

3.3.4) T-test

syntax: x ttest y

semantics: H_0: mean(x) = mean(y)

med locnew *median of locnew*

12.227

((locnew< 12)/pm) ttest (locnew> 12)/pm

This is to test: pm(small size) = pm(large size)

df= 17

t-value= -4.95010515

3.4 The Linear Regression Function

We are also using the "ws.sel" data to demonstrate this function.

The workspace containing this function is "ws.reg" which calls some functions in "ws.stat", so you need to copy both workspaces.

```
goto 'indep'
)copy ws.sel
goto 'tool.84'
)copy ws.stat
)copy ws.reg
```

Before compute the all possible regressions, you need to format the independent variables.

```
x ← 5 mform pm, dur, (pm÷dur), locnew, locall
y ← pm X dur
prod allreg x
```

done *all regressions have been computed*

SMDC provides a "select" function to help you find the best regression model.

The input to the function "select" is a scalar number which is the number of the independent variables of the regression models.

The output is an index table (left) and a statistical table (right).

The index table indicates the independent variables included in the regression.

The first column of the statistical table is R-square, and the second column is Mean Square Error (MSE).

The output is sorted by the R-square value.

Note: "select" cannot select the regressions with more than three independent variables except the regression with all independent variables.

select 1 *select the regression models with only "one" independent variables*

	R-square	*MSE*	
1 0 0 0 0	.0378	.0309	*ind. var. = pm*
0 0 1 0 0	.0226	.0314	*ind. var. = pm÷dur*
0 1 0 0 0	.0093	.0319	*ind. var. = dur*
0 0 0 1 0	.0090	.0319	*ind. var. = locnew*
0 0 0 0 1	.0021	.0321	*ind. var. = locall*

select 2 *select two variables models*

1 0 0 1 0	.4205	.0198	*ind. var. = pm and locnew*
0 0 1 1 0	.3007	.0239	*ind. var. = dur and locnew*
1 0 0 0 1	.2301	.0263	
0 0 1 0 1	.2245	.0265	
1 0 1 0 0	.0471	.0325	
1 1 0 0 0	.0383	.0328	
0 1 0 1 0	.0345	.0330	
0 1 1 0 0	.0251	.0333	
0 0 0 1 1	.0222	.0334	
0 1 0 0 1	.0173	.0336	

select 5

1 1 1 1 1	.4485	.0232	*all independent variables are included*

3.4 plotting routines

First you have to get the workspace which contains those routines

goto 'tool.84'

)copy ws.plot

3.4.1) general plotting function

y plot x - plot the distribution of y vs. x

y is Y-axis, and x is X-axis.

a) Global Variables

title - title of the plot

hlabel - horizontal label

vlabel - vertical label

fd - file descriptor. If the value of fd is one, two, or three, the file been described is terminal. That is the output of the plotting routines will appear on the terminal. (Default=1)

xlow, xhigh, ylow, yhigh - the range of x and y.

If these are not defined, the default is the max or min of the original vector.

l,w - the length and width of the plotting box

b) Initializing procedures

tdim - (i) set the dimensions appropriate for a typical video terminal

$l = 15 \quad w = 72$

(ii) reset the file descriptor (fd) to the terminal

hpdim - set half page paper dimensions(8.5" × 11")

$l = 40 \quad w = 66$

fpdim - set full page paper dimensions(15" × 11")

$l = 40 \quad w = 110$

reset - turn xhigh, xlow, yhigh and ylow back to the default values

create - syntax: create 'file.name'

semantics: create a file named "file.name" for the output of the plotting routines.

note: "create" is used for create a new file. If the file already exists, "create" will clear it. You are not allowed to create any file inside SMDC, so change your directoy to your own environment before crating a file.

c) The Example

tdim *plotting on the terminal*

title ← ' lines of new code vs. person months'

vlabel ← 'effort'

hlabel ← 'lines of new code'

pm plot locnew *note the order of the parameters*

The single point is represented by the symbol "".*

The double points are represented by the symbol "2".

If more than 9 points, it uses "A", "B", ... to represent them.

lines of new code vs. person m...

```
RANGE OF X AXIS:  2.052 84.729
RANGE OF Y AXIS:  5.368750 139.506875

 ------------------------------------------------------------
 |                                    *                                   |
 |                                    *                                * |
 |                                                                        |
 |                                                             *          |
 |                                   *    *                               |
e|                                                                        |
f|                                    *                                   |
f|                                                                        |
o|                                                                        |
r|                                                                        |
t|                                                                        |
 |         *     **                                                       |
 |         **                                                             |
 |       **                                                               |
 | 2 2  *                                                                 |
 ------------------------------------------------------------

                                  lines of new code
```

d) Selection

If you want to examine only a portion of this graph, type

xlow ← 0

ylow ← 0

xhigh ← 15

yhigh ← 20

Type "pm plot locnew" will only plot the points with locnew in [0, 15] and pm in [0, 20].

e) printing the output to a UNIX file

**** See Note before continuing ****

The plotting routines use a APL Quad function to do the output;

therefore you cannot use ")script" to write the output to a UNIX file.

If you want to do that, type

create 'file.name' *create an output file*

y plot x *plot the output on the file 'file.name'*

Note: You cannot create files in the directories of SMDC, you have to do it in your own directory.

3.5.2) vplot - plot in the troff format for the Versatec printing

If you want to use "troff" to format your output, you can use this *function. The global variables and initializing routines are the same as the function "plot". You only need to set one more global variables "figno" which is the figure number.*

create 'file.name'

figno ← figure# *default is 0*

pm vplot locnew

To get the output on Versatec, you need to leave SMDC and type in the following command.

/usr/smdc/bin/plvn **page# file.name** *Versatec output*

or /usr/smdc/bin/plvn **-s page# file.name** *Lasar output*

page# is the page number on your output paper.

3.5.3 Histogram

list ← 1,2,2,3,3,3,4,4,4,4,5,5,5,6,6,7,7,7,7,7,7,7

7 hist list

```
      *          vertical is the frequency
      *          horizontal is the data value
      *
   *  *
  *** *
 ******
*******
```

20 hist list *make the graph sparse*

```
                    *
                    *
                    *
          *         *
      *   *  *      *
   *  *   *  *  *   *
*  *  *   *  *  *   *
```

4 hist list *merge some data points*

```
  *
  *.
 **
 **
 **
 **
****
****
****
```

3.6 Examples Using Software Science Functions

First you have to get the workspace which contains these functions

goto'tool.84'

)copy ws.sofsci

Suppose the data you are interested in obtaining some Software Science estimates are in ws.acm which is located in directory univ.80.

You now have to copy the workspace that contains this data

goto'univ.80'

)copy ws.acm

eta1, eta2, n1, and n2 must be available before computing any functions in this workspace.

)vars

loc eta1 eta2 n1 n2 vg houri hourp hourl name

leq num 1 *estimated length using Software Science length equation (1 significant digit)*

238.6 156.7 382.5 233.6 221.6 859.9 430.2 190.5 821.1 333.0
250.6 173.2 399.2 1030.7 333.3 226.3 372.6 768.4 501.6
257.6 524.1 653.0 202.1

volume num 0 *Software Science volume (1 significant digit)*

1117 594 2179 965 1033 4869 1808 846 4565 2255 1890 905 2580 5230
2242 1993 3202 4707 2908 2336 3907 5167 2054

dif num 1 *Software Science difficulty*

52.3 32.3 49.6 25.7 58.0 37.7 38.1 40.3 38.8 83.9 68.4 42.8
76.0 43.9 70.8 64.1 93.5 34.8 55.5 110.5 96.8 66.2 77.5

sseft num 0 *Software Science effort*

58497 19162 108065 24771 59927 183498 68884 34072 177148 189242
129235 38702 196047 229629 158842 127835 299405 164016 161413
258232 378117 342017 159137

4. Communication between SMDC and the UNIX shell

SMDC has the capability of getting and putting data from/to UNIX files. The functions which serve this purpose are stored in the workspace "ws.io" which is in the directory "tool.84".

4.1 Transferring Data from SMDC to UNIX Files

We also use an example to show this capability.

First we need to copy the workspace "ws.io"

goto 'tool.84'

)copy ws.io

Suppose we want to put the data of "ws.sel" into

a UNIX file called "data.sel".

**** See Note on Page 19 ****

goto 'indep'

)copy ws.sel

create 'data.sel'

no ← ρlocnew *number of points*

putnum no

The function "putnum" will ask you to enter variables one by one.

It works with you interactively.

Please enter the character ID of each data point:
If no, type <return>
L:
type <return> to continue the process
Please enter the variables you want to output:
1th variable
L:
pm
2th variable
L:
dur
3th variable
L:
locnew
4th variable
L:
locall
5th variable
L:
type <return> to stop transferring data

print out the checksum of each variable :
1007.92187 264.19800 503.68700 718.96900

Now, you have created a data file named "data.sel"

which contains all data of "ws.sel" in ASCII format.

4.2 Transferring Data from UNIX Files to SMDC

We also use an example to show this capability.

First you need to copy the workspace "ws.io"

```
goto 'tool.84'
)copy ws.io
```

Suppose you want to read data from a UNIX file called "data.sel"

which has the same format as the file which you created in the previous section,

**** See Note on Page 19 ****

```
sl ← 0   skip first "sl" lines
sc ← 0   the first "sc" columns are character ID
getnum 'data.sel'
```

print out the checksum of each column :

1007.92187 264.19800 503.68700 718.96900

After running the function "getnum", you will have two global variables "name" and "data";

"name" stores the character ID of each point, and "data" stores the numeric data.

You can rename these data and save them for further use.

```
pm ← data[;1]
dur ← data[;2]
locnew ← data[;3]
locall ← data[;4]
)erase sl sc name data getnum putnum create fd  erase the junks
)save ws.sel
```

Acknowledgements

The authors wish to thank Steve Thebaut and Andrew Wang for collecting the initial portions of the data and for writing lots of the statistical functions and plotting routines. The early work on the SMDC was done by Dennis Volpano, who also wrote the original version of this report. The report also includes many suggestions from other members of the Software Metrics Research Group at Purdue University, especially those of Dr. H.E. Dunsmore and Dr. V.Y. Shen. This work has been sponsored by the IBM Corporation through the Santa Teresa Laboratory, San Jose, California and by the U. S. Army Institute for Research in Management Information and Computer Systems, Atlanta, Georgia.

Appendix 1 APL character mapping

APL	UNIX	Dyadic	Monadic
+	+	Addition	Identity
	`		Negation
−	-	Subtraction	Negation
×	X	Multiplication	Sign
÷	%	Division	Reciprocal
|	|	Residue	Absolute
⌊	D	Min.	Floor
⌈	S	Max.	Ceiling
⋆	*	Power	Natural exp
⍟	O<bs>*	Logarithm	Natural log
○	O	Circular func.	pi times
!	!	Combinatorial	Factorial
∧	^	And	
∨	V	Or	
⍲	^<bs>~	Nand	
⍱	V<bs>~	Nor	
<	<	LT	
>	>	GT	
≤	$	LE	
≥	&	GE	
=	=	EQ	
≠	#	NE	
~	~		Not
?	?	Deal	Random
⍴	R	Reshape	Shape
⍳	I	Index of	Index generator
∊	E	Membership	
⊤	N	Representation	
⊥	B	Base value	
⍉	\<bs>O		Transpose
,	,	Concatenate	Ravel
↑	Y	Take	
↓	U	Drop	
←	{	Assignment	
→	}		Go To
⌹	L<bs>%	Matrix div.	Matrix inverse
⍋	|<bs>H	Grade up	
⍒	G<bs>|	Grade down	
⍎	B<bs>J		char. ==> numeric
⍕	N<bs>J	Format	numeric ==> char.
/	/	Compression	
\	\	Expand	
⌽	O<bs>|	Rotate	Reverse
<op> /	<op> /		Reduction
∘.<op>	J.<op>	Outer product	
<op>.<op>	<op>.<op>	Inner product	
⎕	L		Quad
⍞	L<bs>'		Quote-Quad
⌶	B<bs>N	I-beam	I-beam

Appendix 2. The structure of the current SMDC system:

The suffix number of the directory name is the year when the data were collected.
The number in the workspace name is the number of data points in that workspace.
Enter **describe 'workspace name'** to get more information about the workspace you are interested in.

```
sofmet

     --------------------
     |   Functions      |
     --------------------

     tool.84
      ws.io         (* input/output functions      *)
      ws.plot
      ws.reg        (* the regression models       *)
      ws.sofsci     (* Software Science functions *)
      ws.stat       (* statistical functions       *)
      ws.test       (* statistical test functions *)

     --------------------
     |   Data           |
     --------------------

     army
      ws.15         (* 15 products                    *)
      ws.70         (* 70 modules of a product        *)
      ws.271        (* 271 products                   *)
      ws.335        (* 335 products                   *)

     boehm
      ws.basic      (* basic COCOMO                   *)
      ws.inter      (* intermediate COCOMO            *)
      ws.proto      (* prototype experiment           *)

     indep              (* independent sources            *)
      ws.belady
      ws.sel        (* Software Engineering Lab.  *)
      ws.demarco

     indust.80          (* module level                   *)
      ws.27         (* 27 modules of a product        *)
      ws.54         (* 54 modules of a product        *)
      ws.87         (* 87 modules of a product        *)
      ws.259        (* 259 modules of a product       *)
      ws.341        (* 341 modules of a product       *)

     indust.81          (* module level                   *)
      ws.63         (* 63 modules of a product        *)
      ws.90         (* 90 modules of a product        *)
      ws.93         (* 93 modules of a product        *)
```

```
 ws.211        (* 211 modules of a product   *)
 ws.393        (* 393 modules of a product   *)

indust.82         (* product level              *)
 ws.19         (* 19 products                *)
 ws.30         (* 30 products                *)
 ws.41         (* 41 products                *)
 ws.86         (* 86 products                *)

indust.83         (* module level               *)
 ws.25.a       (* 25 modules of a product    *)
 ws.253.b1     (* b1, b2, b3, and b4 are     *)
 ws.253.b2     (* successive versions of     *)
 ws.258.b3     (* the same product           *)
 ws.258.b4
 ws.639.c      (* 639 modules of a product   *)

univ.80
 ws.confirm
 ws.develop
 ws.imsl
 ws.acm

univ.82
 ws.paslib     (* Pascal library programs    *)
 ws.cs440      (* CS440 programs             *)
 ws.cs590e     (* CS590E programs            *)

univ.83
 ws.pret       (* pretest scores             *)
 ws.calc       (* calculator programs        *)
 ws.calc.d     (* detailed information       *)
 ws.dbl        (* database language programs *)
 ws.dbl.d      (* detailed information       *)
```

Appendix 3.

Software Metrics								
directory	workspace	scale	*loc*	E_p	*dur*	SS	$v(G)$	*def*
army	ws.15	L	V	V	V			
	ws.70	L	V			V	V	
	ws.271	L	V			V	V	
	ws.335	L	V			V	V	V
boehm	ws.basic	L	V	V	V			
	ws.inter							
	ws.proto	L	V	V		V	V	
indep	ws.belady	L	V	V	V			
	ws.demarco	L	V	V	V			
	ws.sel	L	V	V	V			
indust.80	*	L	V	V	V			
indust.81	*	L	V	V		V		
indust.82	*	L	V	V	V			
indust.83	*	L	V			V	V	V
univ.80	ws.acm	S	V	V	V	V	V	
	ws.confirm	S	V	V	V	V	V	
	ws.develop	S	V	V	V	V	V	
	ws.imsl	L	V			V	V	
univ.82	ws.paslib	L	V			V	V	
	ws.cs440	M	V			V	V	
	ws.cs590c	M	V	V	V	V	V	
univ.83	*	M	V	V	V	V	V	V

Program scale

S - small scale programs with *loc* less than 200
M - medium scale programs with *loc* between 200 and 1000
L - large scale programs with *loc* more than 1000

E_p : development effort
dur : development duration
SS : Software Science metrics.
def : defect
* means all workspaces under the directory. V means data is available.

Further Information

For further information or questions, see Vickie Owens or Mark Pasch. Appendices 4-6 appear in the Technical Report CSD-TR-421A and can be found online while operating the SMDC system (see p.4).