1981

Software Metrics Data Collection

T. J. Yu

Brian A. Nejmeh

Report Number:
81-421


This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.
Software Metrics Data Collection

T.J. Yu and Brian A. Nejneh
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 Introduction
2 Summary of Capabilities
   2.1 Logging On and Off
   2.2 Metrics Included (Data Collected)
   2.3 Statistical Functions
   2.4 Statistical Testing Functions
   2.5 The Linear Regression Function
   2.6 Plotting Routines
   2.7 Software Science Functions
3 Examples of Using the SMDC System
   3.1 Logging On
3.2 Statistical Functions
3.3 The Linear Regression Function
3.4 Plotting Routines
3.5 Software Science Functions

4 Communication between SMDC and the UNIX shell
4.1 Transferring Data from SMDC to UNIX Files
4.2 Transferring Data from UNIX Files to SMDC

Appendices
A.1 APL Character Set
A.2 The Current Structure of SMDC
A.3 Table of Current Workspaces and Their Associated Metrics
A.4 General Description of Current Environments (Directories)
A.5 Detailed Description of Current Environments (Directories)
A.6 Detailed Description of Current Workspaces

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/smde
```

After changing your current directory to /usr/smde, the command "smde" 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/smde/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 'workspace name' where 'workspace name' 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 "tool84" on data in "ws basic". Then, assuming you have entered cd /usr/smde and smde, you would enter
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:
\textit{locnew} : refers to all new or modified lines of code.
\textit{locbase} : refers to lines of code which are unchanged from the previous version.
\textit{loccopy} : refers to lines of code which are copied or are from another source.

We suggest that \textit{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 \times 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 \( r_1, r_2, \ldots \) 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 \( pmu \).

2.2.3) duration(\textit{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

\[ eta_1 : \text{number of unique operators} \]
\[ eta_2 : \text{number of unique operands} \]
\[ n_1 : \text{total occurrences of operators} \]
\[ n_2 : \text{total occurrences of operands} \]

2.2.5) Cyclomatic complexity\(^2\) \((v_g)\) – 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 \(\text{small, medium, or large}\). 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.

---

\(^2\) McCabe TJ. 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 (e.g., "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
\texttt{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 \texttt{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 \texttt{describe 'ws sofact'}.

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:

\textbf{boldface} - commands/data input to the system

\textit{italics} - descriptive comments

3.1 Logging on

\textit{After you have entered}
\begin{verbatim}
ed /usr/smde
smde
\end{verbatim}

you will see the following on your terminal:

Welcome to SMDC
erase: < control > 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.

```apl
)ins
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 'lDdep'
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 local 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
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

prodm = prod

sd prod = sample standard deviation of "prod"

var prod = variance of "prod"

.028764844

3.2.3 Correlation Coefficients

a) Pearson correlation coefficient

locnew cor pm

.899744232

pm cor locnew

.899744232

b) Spearman rank correlation coefficient

locnew spear pm

.926315789

tocall spear pm

.91405088

c) multiple variables formatter

Syntax: n mform var1, var2, ... varn

Semantics: If the rank of each variable is r, mform will return a (r x 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, dim format 3 variables
x - 5 mform prod, pm, dur, locnew, local

format 5 var

d) Pearson correlation matrix

\[ \begin{align*}
  & \text{prod} & \text{pm} & \text{dur} & \text{locnew} & \text{local} \\
  \text{prod} & 1.000 & -0.194 & 0.906 & 0.046 \\
  \text{pm} & -0.194 & 1.000 & 0.592 & 0.900 & 0.871 \\
  \text{dur} & 0.906 & 0.592 & 1.000 & 0.469 & 0.381 \\
\end{align*} \]

3.2.4 Linear Regression

a) regression through origin

\[ \text{pm} = 0 \text{ locnew} \]

\[ 1.88347325 \] i.e. \( \text{pm} = 1.88 \text{ locnew} \)

b) slope and intercept

\[ \text{pm} = \text{slope} \text{ locnew} \]

\[ 1.7483875 \]

\[ \text{pm} = 6.70 + 1.75 \text{ locnew} \]

\[ 1.94727702 \]

i.e. \( \text{locnew} = 1.95 + 0.46 \text{ pm} \)

c) Simple Linear Regression

\[ \text{pm} = \text{mlln} \text{ locnew} \]

note the order of the parameters

coefficients:

\[ Y = b_0 + b_1 X_1 + \ldots + b_n X_n \]

\[ 6.69904306 \]

i.e. \( \text{pm} = 6.70 + 1.75 \text{ locnew} \)

\[ 1.74838750 \]

R-square: \( 0.89539683 \) 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.

\[ y = \text{pm} \]

\[ x = 2 \text{ mform locnew, local} \] x is a 19 by 2 matrix
\[ y \text{ mlm } x \]

coefficients:

\[ Y = b_0 + b_1 X_1 + \ldots + b_n X_n \]

\[ \text{i.e. pm } = 4.90 + 1.29 \text{ locnew } + 0.37 \text{ local} \]

4.89749730

1.28696800

37086491

Anova: Source of Variation, df, ss, ms

\begin{tabular}{ccc}
regression & 2 & 35928.4308 \hspace{1cm} 17964.2154 \\
error & 16 & 7970.74508 \hspace{1cm} 498.171567 \\
\end{tabular}

R-square: .818430644

"mlm" 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.

\[ \text{pm compare yhat} \]

\[ \text{note the order of the parameters} \]

\[ \text{namely, the actual value appears first and the predicted value appears second} \]

number of data points \hspace{1cm} 19

mean relative error \hspace{1cm} -2.39

mean magnitude of relative error \hspace{1cm} .37

\% of prediction within 25\% \hspace{1cm} .421

mean square error \hspace{1cm} 442.819171

correlation coefficient \hspace{1cm} .905

spearman rank cor. coef. \hspace{1cm} .944

3.2.5 Set Functions

\begin{align*}
a & = 1 \ldots 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 & \text{ union } b \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\
a & \text{ intersect } b \\
6 & 7 & 8 & 9 & 10 \\
a & \text{ differ } b \\
\end{align*}
3.2.6 Others

a) domain - compute the domain of a given vector

domain locnew

<table>
<thead>
<tr>
<th>Length</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>2.052</td>
</tr>
<tr>
<td>Low+1</td>
<td>2.451</td>
</tr>
<tr>
<td>Mean</td>
<td>26.5098421</td>
</tr>
<tr>
<td>Median</td>
<td>12.227</td>
</tr>
<tr>
<td>Std. Dev</td>
<td>25.4140165</td>
</tr>
<tr>
<td>High-1</td>
<td>76.883</td>
</tr>
<tr>
<td>High</td>
<td>84.729</td>
</tr>
</tbody>
</table>

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 local three levels of program size

(Ind=1)/local small size

(ind=2)/local: medium size
55.237 50.911 14.863 32.822 14.765 17.27

(ind=3)/local: 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)

<table>
<thead>
<tr>
<th>source</th>
<th>ss</th>
<th>df</th>
<th>ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Between</td>
<td>30452.9078</td>
<td>2</td>
<td>15226.4539</td>
</tr>
<tr>
<td>Error</td>
<td>13446.268</td>
<td>16</td>
<td>840.391753</td>
</tr>
<tr>
<td>F</td>
<td>18.118281</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.3.3) Single-factor analysis of covariance (ANCOVA)

\[ x = 2 \text{mform ind, dur} \]
The independent variables of "ancova" must be a two-column array.
The first column is the factor to be analysed.
The second column is the concomitant variable.
"ancova" can only accept one concomitant variable.

\[ pm \anova x \]

This is to test the effect of the factor "ind"
after eliminating the effect of duration.

Ancova Table

<table>
<thead>
<tr>
<th>source</th>
<th>y</th>
<th>x</th>
<th>xy</th>
<th>df</th>
</tr>
</thead>
<tbody>
<tr>
<td>Treatment</td>
<td>30452.9078</td>
<td>44.8818437</td>
<td>1124.61165</td>
<td>2</td>
</tr>
</tbody>
</table>
Error 13446.268 290.638475 1148.38339 16
Total 43899.1759 355.520319 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: H0: 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, local
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

<table>
<thead>
<tr>
<th>R-square</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>.0378 .0309</td>
</tr>
<tr>
<td>00100</td>
<td>.0226 .0314</td>
</tr>
<tr>
<td>01000</td>
<td>.0093 .0319</td>
</tr>
<tr>
<td>00010</td>
<td>.0090 .0319</td>
</tr>
<tr>
<td>00001</td>
<td>.0021 .0321</td>
</tr>
</tbody>
</table>

**select 2** select two variables models

<table>
<thead>
<tr>
<th>R-square</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>10010</td>
<td>.4205 .0198</td>
</tr>
<tr>
<td>00110</td>
<td>.3007 .0239</td>
</tr>
<tr>
<td>10001</td>
<td>.2301 .0263</td>
</tr>
<tr>
<td>00101</td>
<td>.2245 .0265</td>
</tr>
<tr>
<td>10100</td>
<td>.0471 .0325</td>
</tr>
<tr>
<td>11000</td>
<td>.0383 .0328</td>
</tr>
<tr>
<td>01010</td>
<td>.0345 .0330</td>
</tr>
<tr>
<td>01100</td>
<td>.0251 .0333</td>
</tr>
<tr>
<td>00011</td>
<td>.0222 .0334</td>
</tr>
<tr>
<td>01001</td>
<td>.0173 .0336</td>
</tr>
</tbody>
</table>

**select 5**

<table>
<thead>
<tr>
<th>R-square</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>11111</td>
<td>.4485 .0232</td>
</tr>
</tbody>
</table>

**all independent variables are included**

### 3.4 plotting routines

First you have to get the workspace which contains those routines

```latex
\texttt{goto 'tool84'}
\texttt{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
hlable - horizontal label
vlable - 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   w = 72
   (ii) reset the file descriptor (fd) to the terminal
hpdim - set half page paper dimensions(8.5" × 11")
l = 40   w = 66
fpdim - set full page paper dimensions(15" × 11")
l = 40   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 directory to
              your own environment before creating a file.

c) The Example
tdim plotting on the terminal
 title - 'lines of new code vs. person months'
 vlable - 'effort'
 hlable - '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 
"Z".
If more than 9 points, it uses "A", "B", ... to represent them.
lines of new code vs. person months

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

lines of new code

d) Selection

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

```plaintext
slow 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 "jscript" to write the output to a UNIX file.

If you want to do that, type

```plaintext
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 initialising 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:

/user/smdc/bin/plvn page# file.name Versatec output
or /user/smdc/bin/plvn -s page# file.name Laser output

page# is the page number on your output paper.

3.5.3 Histogram

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

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 sof ace
```

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

You now have to copy the workspace that contains this data:

```
  goto 'unix.80'
  copy ws acm
```

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

```
ivars
loc eta1 eta2 n1 n2 vg houri hourp houml 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 4555 2255 1890 905 2580 5230
2242 1993 3202 4707 2908 2336 3907 5167 2054
```

```
diff 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
```

```
eff 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 jo" 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 - plocnew 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]
locali = data[;4]
)erase sl sc name data getnum putnum create fd erase the junks
)save ws.sel

Acknowledgements

The authors wish to thank Steve Thebault and Andrew Wang for collecting the initial
portions of the data and for writing lots of the statistical functions and plotting routines. The
ey 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.
<table>
<thead>
<tr>
<th>APL</th>
<th>UNIX</th>
<th>Dynamic</th>
<th>Monadic</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>+</td>
<td>Addition</td>
<td>Identity</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>Subtraction</td>
<td>Negation</td>
</tr>
<tr>
<td>*</td>
<td>O&lt;bs&gt;*</td>
<td>Power</td>
<td>Natural exp.</td>
</tr>
<tr>
<td>/</td>
<td>X&lt;bs&gt;%</td>
<td>Division</td>
<td>Reciprocal</td>
</tr>
<tr>
<td>%</td>
<td>Z&lt;bs&gt;^</td>
<td>Residue</td>
<td>Absolute</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D&lt;bs&gt;M</td>
<td>Max.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>S&lt;bs&gt;*</td>
<td>Min.</td>
</tr>
<tr>
<td>≤</td>
<td>V&lt;bs&gt;_≥</td>
<td>And</td>
<td>And</td>
</tr>
<tr>
<td>≥</td>
<td>V&lt;bs&gt;_≤</td>
<td>Or</td>
<td>Or</td>
</tr>
<tr>
<td>&lt;</td>
<td>V&lt;bs&gt;_&gt;</td>
<td>Nand</td>
<td>Nand</td>
</tr>
<tr>
<td>&gt;</td>
<td>V&lt;bs&gt;_&lt;</td>
<td>Nor</td>
<td>Nor</td>
</tr>
<tr>
<td>=</td>
<td>S&lt;bs&gt;_≠</td>
<td>EQ</td>
<td>EQ</td>
</tr>
<tr>
<td>≠</td>
<td>S&lt;bs&gt;_=</td>
<td>NE</td>
<td>NE</td>
</tr>
<tr>
<td>?</td>
<td>K&lt;bs&gt;_?</td>
<td>Deal</td>
<td>Deal</td>
</tr>
<tr>
<td>?</td>
<td>K&lt;bs&gt;_?</td>
<td>Random</td>
<td>Random</td>
</tr>
<tr>
<td>R</td>
<td>R&lt;bs&gt;R</td>
<td>Reshape</td>
<td>Reshape</td>
</tr>
<tr>
<td>I</td>
<td>L&lt;bs&gt;I</td>
<td>Index of</td>
<td>Index of</td>
</tr>
<tr>
<td>E</td>
<td>E&lt;bs&gt;E</td>
<td>Membership</td>
<td>Membership</td>
</tr>
<tr>
<td>N</td>
<td>N&lt;bs&gt;N</td>
<td>Representation</td>
<td>Representation</td>
</tr>
<tr>
<td>B</td>
<td>B&lt;bs&gt;B</td>
<td>Base value</td>
<td>Base value</td>
</tr>
<tr>
<td></td>
<td>&lt;bs&gt;0</td>
<td>Transpose</td>
<td>Transpose</td>
</tr>
<tr>
<td></td>
<td>Y&lt;bs&gt;Y</td>
<td>Concatenate</td>
<td>Concatenate</td>
</tr>
<tr>
<td></td>
<td>U&lt;bs&gt;U</td>
<td>Take</td>
<td>Take</td>
</tr>
<tr>
<td></td>
<td>I&lt;bs&gt;I</td>
<td>Drop</td>
<td>Drop</td>
</tr>
<tr>
<td></td>
<td>I&lt;bs&gt;I</td>
<td>Assignment</td>
<td>Assignment</td>
</tr>
<tr>
<td></td>
<td>L&lt;bs&gt;%</td>
<td>Matrix div.</td>
<td>Matrix div.</td>
</tr>
<tr>
<td></td>
<td>G&lt;bs&gt;≤</td>
<td>Grade up</td>
<td>Grade up</td>
</tr>
<tr>
<td></td>
<td>H&lt;bs&gt;≥</td>
<td>Grade down</td>
<td>Grade down</td>
</tr>
<tr>
<td></td>
<td>G&lt;bs&gt;≤</td>
<td>Format</td>
<td>Format</td>
</tr>
<tr>
<td></td>
<td>I&lt;bs&gt;I</td>
<td>Compression</td>
<td>Compression</td>
</tr>
<tr>
<td></td>
<td>I&lt;bs&gt;I</td>
<td>Expand</td>
<td>Expand</td>
</tr>
<tr>
<td></td>
<td>L&lt;bs&gt;l</td>
<td>Rotate</td>
<td>Rotate</td>
</tr>
<tr>
<td></td>
<td>O&lt;bs&gt;l</td>
<td>Reverse</td>
<td>Reverse</td>
</tr>
<tr>
<td></td>
<td>R&lt;bs&gt;R</td>
<td>Reduction</td>
<td>Reduction</td>
</tr>
<tr>
<td>&lt;op&gt;/</td>
<td>&lt;op&gt;/</td>
<td>Outer product</td>
<td>Outer product</td>
</tr>
<tr>
<td>&lt;op&gt;.&lt;op&gt;</td>
<td>&lt;op&gt;.&lt;op&gt;</td>
<td>Inner product</td>
<td>Inner product</td>
</tr>
<tr>
<td>0</td>
<td>0&lt;bs&gt;0</td>
<td>I-beam</td>
<td>I-beam</td>
</tr>
<tr>
<td>0</td>
<td>L&lt;bs&gt;*</td>
<td>I-beam</td>
<td>I-beam</td>
</tr>
<tr>
<td>1</td>
<td>B&lt;bs&gt;N</td>
<td>I-beam</td>
<td>I-beam</td>
</tr>
</tbody>
</table>

Appendix 1 - APL character mapping

- Addition: Identity
- Subtraction: Negation
- Multiplication: Sign
- Division: Reciprocal
- Residue: Absolute
- Max: Ceiling
- Min: Floor
- Power: Natural exp
- Logarithm: Natural log
- Circular func.: pi times
- Combinatorial: Factorial
- And: Not
- Or: Random
- LT: Deal
- GE: Random
- LE: Shape
- EQ: Index generator
- NE: Index generator
- Concatenate: Transpose
- Take: Ravel
- Drop: Ravel
- Assignment: Ravel
- Matrix div.: Go To
- Grade up: Matrix inverse
- Grade down: char. => numeric
- Format: numeric => char.
- Expand: Reverse
- Rotate: Reduction
- Outer product: Quad
- Inner product: Quote-Quad
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.

sofnet

-----------------------------
| 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.aam

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.db1 (* database language programs *)
ws.db1.d (* detailed information *)
Appendix 3.

<table>
<thead>
<tr>
<th>directory</th>
<th>workspace</th>
<th>scale</th>
<th>loc</th>
<th>$E_p$</th>
<th>dur</th>
<th>SS</th>
<th>$v(G)$</th>
<th>def</th>
</tr>
</thead>
<tbody>
<tr>
<td>army</td>
<td>ws.15</td>
<td>L</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ws.70</td>
<td>L</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ws.271</td>
<td>L</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ws.335</td>
<td>L</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td></td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>bochm</td>
<td>ws.basic</td>
<td>L</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ws.inter</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ws.proto</td>
<td>L</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>indep</td>
<td>ws.bclady</td>
<td>L</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ws.demarco</td>
<td>L</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ws.sel</td>
<td>L</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>industri.80*</td>
<td></td>
<td>L</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>industri.81*</td>
<td></td>
<td>L</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>industri.82*</td>
<td></td>
<td>L</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>industri.83*</td>
<td></td>
<td>L</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>univ.80</td>
<td>ws.acm</td>
<td>S</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ws.confirm</td>
<td>S</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ws.develop</td>
<td>S</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ws.imsl</td>
<td>L</td>
<td>V</td>
<td></td>
<td>V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>univ.82</td>
<td>ws.paslib</td>
<td>L</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ws.cs440</td>
<td>M</td>
<td>V</td>
<td></td>
<td>V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ws.cs590c</td>
<td>M</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>univ.83</td>
<td></td>
<td>M</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td></td>
<td>V</td>
</tr>
</tbody>
</table>

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).