1984

Software Metrics Data Collection, Appendices 4-6

T.J. Yu

Brian A. Nejmeh

Report Number:
84-421A

http://docs.lib.purdue.edu/cstech/346

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
Appendices 4-6

T.J. Yu and Brian A. Nejmeh

Department of Computer Sciences
Purdue University
West Lafayette, Indiana 47907

CSD-TR-421A
(October 1984)
Appendix 4. General Description of Current Environments
Documentation for the software metrics data collection

The following is a list of the "working environments" (directories) currently available. The suffix number of the directory number is the year when the data was collected.

Enter "goto 'directory-name" to access the workspaces within it.
Enter "describe 'directory-name" to get more information about it.

1) tool.84

In this directory, there are five workspaces available.
Each workspace contains several functions for analyzing formatted data.

2) army

This directory contains four workspaces of data collected from a certain army organization. The workspace "ws.15" contains data for studying effort models; the workspace "ws.335" contains data for studying defect models. The workspaces "ws.70" and "ws.271" can be used to validate Software Science models.

3) boehm

In this directory, there are three workspaces of data collected by Barry Boehm in 1981 and 1979.
The workspace "ws.basic" contains the basic COCOMO data.
The workspace "ws.inter" contains the intermediate COCOMO data.
The workspace "ws.proto" contains the data for comparing specifying technique with prototyping technique.

4) indep

This directory contains three workspaces of data collected from three independent sources. Each workspace contains data from a number of large products for studying effort models.

5) indust.80

This directory contains data of five large products from the same company. Each product corresponds to a workspace which contains module level metric counts and detailed personnel information.

6) indust.81

This directory contains data of five large products from the same company. Each product corresponds to a
workspace which contains Software Science metrics.

7) industr.82
This directory contains four workspaces of data from the same company. Each workspace contains data of a number of large products for studying effort models.

8) industr.83
This directory contains data of six large products from the same company. Each product corresponds to a workspace which contains data for studying defect models.

9) univ.80
This directory contains four workspaces of data of small scale programs. The workspaces, "ws.develop", "ws.confirm", and "ws.acm", contain data for studying Woodfield’s effort estimation models based on Software Science metrics. The workspace "ws.ims" only contains Software Science metrics.

10) univ.82
This directory contains three workspaces of data. The workspaces "ws.paslib" and "ws.cs440" only contains data of software metrics derived from program. The workspaces "ws.cs590e" contains data at each milestone during the software development period which includes effort and software metrics.

11) univ.83
This directory contains detailed and complete information of an experiment conducted in the summer of 1983. The workspace "ws.pret" contains the pretest scores of the subjects which can be used as a measure of their programming ability. The workspaces "ws.calc" and "ws.dbl" contain data of development and testing effort, software metrics, and defects. The workspace "ws.calc.d" and "ws.dbl.d" contain data on each milestone during the development period.
Appendix 5. Detailed Description of Current Environments
Documentation for the directory tool 84

workspaces:

a) ws.io - This workspace contains three functions to
transfer numeric data between Unix files and APL
workspaces.

b) ws.plot - This workspace contains some plotting routines
to generate figures in terminal format, nroff format,
or troff format.

c) ws.reg - This workspace generates all possible regressions
for multiple independent variables.

d) ws.sofsci - This workspace contains several Software Science
functions, such as length equation, volume, difficulty, and
effort.

e) ws.stat - This workspace contains a large number of
statistical functions, such as mean, median, variance,
multiple linear regression etc..

f) ws.test - This workspace contains several functions
which are more sophisticated than those in “ws.stat”,
such as ANOVA, ANCOVA, T-test, and Kruskal-Wallis
rank test.
Documentation for the directory army

This directory contains four workspaces of data collected from a certain army organization.

ws.15 - 15 large products

ws.271 - 271 COBOL programs

ws.70 and ws.335 -

The information received from army concerned 405 COBOL production "programs" of all descriptions: size ranged from small to large and structure varied from monolithic non-modular programs to "programs" which were actually modules of larger programs. Generally, the programs were medium to large in size and were non-modular (here non-modular means that, although a large program may have been divided into logical modules, these were all contained physically within the program). Also, most of the programs were written in what might be termed "unstructured" COBOL as most of the programs used the COBOL GO TO verb liberally. In summary, these are probably a good example of "real-world" COBOL programs.

ws.70 - 70 modules comprising a large COBOL product.

ws.335 - 335 COBOL programs.
Documentation for the directory boehm

workspaces:

a) ws.basic - The data in this workspace corresponds to the published data to validate the basic COCOMO model.

b) ws.inter - The data in this workspace corresponds to the published data to validate the intermediate COCOMO model.


c) ws.proto - This workspace contains the data of an experiment conducted at UCLA in early 1982. Seven software teams developed the same application software (an interactive intermediate COCOMO model). Four teams used the "specifying" approach, and the other three teams used "prototyping" approach.

Documentation for the directory indep

a) ws.belady - this data corresponds to 33 programs developed by a large software house. The data includes lines of code, effort and duration.


b) ws.demarco -
This set of data is based on the following survey: "The Yourdon 78-80 Project Survey" conducted by Tom DeMarco. Companies with a new software project in an early stage of development were invited to provide detailed information on the progress of their efforts on a monthly basis. DeMarco provided forms for this purpose which specified the information sought and the assumptions to be used. Besides the monthly progress reports, initial and final project questionnaires and a follow-up report were requested. Thirty-two companies participated in the study in return for a report on its findings. Data reflecting 18 projects are used in the present. Data for 19 projects were originally selected and made available to Purdue's Software Metrics Research Group on the basis of their relative completeness. However, one of the 19 projects did not run to normal completion and was eliminated from the set. The majority of products represented by this set are associated with business applications and were written in COBOL.

c) ws.sel -
This data comes from the Software Engineering Laboratory of the University of Maryland at College Park and corresponds to 19 products developed at the NASA/Goddard Space Flight Center since 1976. A subset of this data has been used in a previous study at the University of Maryland and published.

Most of the programs in this set are comprised of ground support routines for various spacecraft projects. The primary language used is FORTRAN with some application of assembler language. Projects were supervised by NASA personnel, but may involve the use of some outside contractors.
Documentation for the directory indust.80

In this directory, there are five workspaces:

ws.27  ws.54  ws.87  ws.259  ws.341

Each workspace corresponds to a large product developed by a large company. The data includes extensive module and organization data. The attributes and functions are essentially the same for each workspace. Each data point corresponds to a module of a product.

Characteristics:

Large projects
No. of projects: 5
Range of program size: 6,785 - 40,187 loc

Note: each workspace contains detailed information of one project.

VARS:

nprog - number of programmers in the development group.

nmod - total number of modules.

pms - number of person months.
Note: this metric has been scaled.

dur - duration of the development of projects.

r1 - fraction of total person months associated with development process.

r2 - fraction of total person months associated with testing.

r3 - fraction of total person months associated with performance analysis.

r4 - fraction of total person months associated with "other" costs.

loc - lines of code for each module.

pgmr - pgmr[i] is the index number of the programmer responsible for module i.

boss - boss[i] is the index number of the boss of programmer i.

callslist - callslist[k] is the module called by k.

callsindex - callsindex[k;1] is the index into callslist for k.
callsindex[k;2] is the number of modules called by k.
cwrklist - cwrklist[k] is the co-worker of programmer k.

cwrkindex - cwrkindex[k;1] is the index into cwrklist for k.
cwrkindex[k;2] is the number of coworkers of programmer k.

smon - if smon[i] = k, this implies that programmer i started
on the project k months after the project began.

emon - if emon[i] = t, implies that programmer i stopped working
on the project t months after it began.

vacmon - vacmon[i] is the number of vacation months taken by
programmer i.

plag - plag[i] is the length (in lines) of the prologue of
module i where a prologue is simply documentation for
some module.

fns:
calls - calls k; returns a list of the modules called by module k.
cwrk - cwrk k; returns a list of programmer indices representing
the co-workers of programmer k.

mp - mp k; returns a list of modules produced by programmer k.

mu - mu k; returns a list of modules used by programmer k.
mimp - mimp k; returns a list of modules imported by programmer k.
sups - i supers j; returns a list of modules supplied by programmer i
and used by programmer j.
id - id; returns a small description of the project which the
data represents.
Documentation for the directory industbl

In this directory, there are five workspaces

- ws.63 - PL/S
- ws.90 - PL/S
- ws.93 - Assembly Language
- ws.211 - Assembly Language
- ws.393 - Not available

Each workspace corresponds to a large product developed at a large company. The data included are the Software Science metrics. Each data point corresponds to a module of a product. The attributes are essentially the same for each workspace.

vars

- \( \text{eta1} \): unique number of operators.
- \( \text{eta2} \): unique number of operands.
- \( \text{n1} \): total number of operators.
- \( \text{n2} \): total number of operands.
- \( \text{loc} \): thousands of lines of code.
- \( \text{pms} \): total person months for the product.

Note: 1) no information for each individual module exists.
2) this metric is scaled.
Documentation for the directory indust.82

In this directory, there are three workspaces.

ws.19  ws.30  ws.41  ws.86

Each workspace corresponds to a collection of (19, 30, 41, 86) large products developed at a large company.

Note: ws.41 is a combination of ws.19 and another 22 points from ws.86.

Please enter "describe 'ws.xx'" to get more information about each workspace.
Documentation for the directory indust.83

This directory contains six workspaces; each corresponding to a large software product.

ws.24.a - 24 modules of a product. This product was written in Pascal, and was used to count the software metrics of PL/S programs.

ws.253.b1 - 253 modules of a product. This product was written in PL/S, and was a compiler.

ws.253.b2 - The successive release of b1.

ws.258.b3 - the successive release of b2 which included five new modules.

ws.258.b4 - the successive release of b3.

ws.639.c - 639 modules of a large product. Most of the modules were written in assembly code. About 1000 base modules were not included in this workspace. It was used in a database management system.
Documentation of the directory univ.80

a) ws.acm - this data corresponds to programs written by members of the 1980 ACM programming team. Metrics were obtained using the logical, physical and integrated module model of Woodfield.

b) ws.develop - this data corresponds to programs written during practice sessions of the Purdue programming team in 1980. Data is organized according to model type: integrated or physical.

c) ws.confirm - this data corresponds to programs written during practice sessions of the Purdue Programming Team in 1980. Data is organized according to model type: integrated or physical.


d) ws.imsl - this data corresponds to subroutines written for the IMSL library. Each of the 497 subroutines is treated as an independent modules.
Documentation for the directory univ.82

ws.paslib - 24 large Pascal programs collected from Purdue University Computing Center library.

ws.cs440 - 33 medium size Pascal programs written by the students of CS 440 in the spring of 1982.

ws.cs590e - 21 medium size Pascal programs written by the students of CS 590E in the summer of 1982. The data collected included the software metrics and the effort of the successive versions of each program.
Documentation for the directory univ.83

This directory contains the data collected in the experiment conducted in CS 590E in the summer of 1983. There were 44 students involved in this study, each of whom had taken at least one computer science course concerning advanced programming technique. Their program design ability and their familiarity with Pascal were tested by a pretest to ensure that all were qualified to participate in this experiment.

Each student wrote two programs. One program was called the calculator program (calc) which simulated a desk calculator. The other was called the database program (dbl) which translated a database query language program into a Pascal program.

There are five workspaces in this directory:

ws.pret - This workspace contains the pretest scores of the students which includes cloze procedure scores, extended cloze procedure scores, and comprehension quiz scores.

ws.calc - This workspace contains the information of the calculator programs, such as software metrics, effort, and defects.

ws.calc.d - This workspace contains the information of the successive versions of each calculator program.

ws.dbl - This workspace contains the information of the database programs, such as software metrics, effort, and defects.

ws.dbl.d - This workspace contains the information of the successive versions of each database program.
Appendix 6. Detailed Description of Current Workspaces
Documentation for ws.stat

fns:

mean - mean x; computes the mean of the components of the vector x

med - med x; compute the median of the components of the vector x

sd - sd x; computes the standard deviation of the components of the vector x

var - var x; compute the variance of vector x

note: var << (sd x sd)

mlin - y mlin x; multiple linear regression for the vectors y and x;


global output variable (yhat) - predicted value of y

mvavg - k mvavg vec; compute the moving average of order k of vector vec

spear - x spear y; computes the spearman rank correlation of the vectors x and y; spear calls 'rank' as a helper function

cor - x cor y; compute the correlation coefficient of x and y

slope - y slope x; compute the slope of regression line

intcp - y intcp x; compute the intercept of the regression line

suppose \( y = Ax + B \) then

slope = A
intcp = B

differ - setA differ setB; the difference of two sets

differ = setA - setB

union - setA union setB; the union of two ordered sets

intersec - setA intersec setB; the intersection of two sets

domain - domain x; return the highest, the lowest, the mean, the median and standard deviation of x.

which - vector which number; return the index(es) of the number(s) in the vector

mform - n mform x; format variable x to be a n by m array,

where m is "(Rx)%n"
It helps to format the input variables of "mlin", "cormat", and "allre(' in ws.reg."
num - x num n; format the variable x to have only "n" decimal points.

lin0 - y lin0 x; Let Y = aX, This function will return "a".
    i.e. force the regression through the origin.

cormat - y cormat x; compute the correlation matrix between y and x.
    The result is saved in the GLOBAL variable "cortbl".
    If Ry=(n,m) and Rx=(n,l) then Rcorrbl=(m,l)

compare - y compare yhat; Compare y and yhat by showing their
    MRE, RE, Pred-25%, correlation coefficient
    (both Pearson and Spearman rank)
    y is actual value, and yhat is predicted value.
Documentation for ws.test

This workspace contains a set of functions for testing statistical hypotheses. Copy "ws.stat" before using these functions.

**gpind** - n gpind x;
(1) Divide x into n groups by the rank of x
(2) The number of points in each group are same
(3) Return the group ID for each point in x
   The smaller the value of x, the smaller the ID number.
   note: This is served as a predecessor of "anova", "kw", and "ancova".

**kw** - dep kw ind; Kruskal-Wallis one way analysis of variance.
for ordinal data. It uses krank as helper function.
Test H0: rank(x1) = rank(x2) ... rank(x3)
   xi is (ind=i)/dep

**anova** - dep anova ind; Single factor one way analysis of variance.
Test H0: mean(x1) = mean(x2) ... = mean(xk)
   xi is (ind=i)/dep
   ind must start from one.

**ancova** - dep ancova ind; analysis of covariance.
   the independent variable is an n by 2 array.
   the first column is the level of the factor which must start from one.
   the second column is the concomitant variable.

**ttest** - y ttest x; T-test.
Test H0: mean(x)=mean(y)

**chisq** - y chisq x; Chi-square test
   y - the outcome of experiment, starting from zero
   x - the group of different treatments, starting from one
Documentation of workspace ws.reg

This workspace contains several functions to compute all possible linear regressions for a given set of independent variables. Note: Copy "ws.stat" before running the regression function.

allreg - y allreg x; where y is the dependent variable of rank n and x is a set of independent variables of rank (n p), p is number of independent variables.

Global variables:

Input:

ng - The first "ng" independent variables need to be grouped together.

Output:

index - index array, each row corresponds to one regression model, those independent variables included in the model are flagged by 1.
statbl - two column matrix, the first column is the R-square and the second column is MSE. Each row corresponds to the same row of the variable "index".

select - select n; select those regression models with "n" independent variables and sort them by their R-square values.

Helper functions:

reg - y reg x; compute R-square and MSE of y and x.

group - n group ind; expand 1st independent variable to a set of (n+1) independent variables.
Documentation for ws.plot

vars:

vlabel - vlabel is a character vector initialized by the user, and used in plot; the character string constitutes the vertical label of the plot.

hlabel - same as above except that this string constitutes the horizontal label of the plot.

title - title is a character vector, initialized by the user, and used in plot; the character string denotes the title of the plot.

xlow - the lower bound on the vector x to be used by plot.

xhigh - the upper bound on the vector x to be used by plot.

ylow - the lower bound on the vector y to be used by plot.

yhigh - the upper bound on the vector y to be used by plot.

l - the length of the plot.

w - the width of the plot.

fd - file descriptor; default = 1 (terminal).

figno - figure number of Versatec plotting.

fns:

plot - y plot x; plots y vs. x in the file specified by fd.
    Global variable: fd - file descriptor
    Default = 1 (terminal output).

vplot - y vplot x; same as plot but in Versatec format.

reset - reset; reset xlow, xhigh, ylow and yhigh to zero.

tdim - tdim; sets l and w to terminal dimensions, namely l = 15 and w = 72, and reset fd to terminal output (fd=1).

fpdim - fpdim; sets l and w to full page dimensions, namely l = 40 and w = 110.

hpdim - hpdim; sets l and w to half page dimensions, namely l = 40 and w = 66.

hist - c hist l; plot the histogram where r is number and l is a list of data.
create a file named 'file' and set the value of the file descriptor to the variable "fd".
Documentation for ws.sofsci

This workspace contains several Software Science functions defined by M.H. Halstead. Make sure that eta1, eta2, n1 and n2 are defined before you use these functions.

fns:

- leq - leq; compute the estimated Software Science length (Nhat) by Software Science length equation.

- volume - volume; Software Science volume.

- dif - dif; Software Science difficulty.

- ssoft - sso; Software Science effort.
Documentation of ws.io

This workspace contains several functions to transfer data between APL workspaces and ASCII files.

create - create 'file'; create an ASCII file named 'file' and set the variable "fd" to be the file descriptor.

putnum - putnum r; put data into a specified ASCII file.
  r is the rank of each data array.
  Each data array is entered interactively.
  input GLOBAL variable:
  fd - file descriptor

getnum - getnum 'file'; get data from the ASCII file named 'file'
  input GLOBAL variables:
  sl - skip the first 'sl' lines of the file
  sc - set the first 'sc' columns to be the ID of each data point.

The result is put into two GLOBAL variables
  data - is a two-dimensional numeric array.
  name - is the ID of each data point.
Documentation for the workspace ws.15

vars

  pm - effort in person months.

  dur - duration.

  loc - lines of code excluding comments.
Documentation for the workspace ws.70

vars

loc - lines of code excluding comments.

vg - McCabe's cyclomatic complexity number.

armyvg - vg revised by Army.

eta1 - number of unique operators.

eta2 - number of unique operands.

n1 - total number of operators.

n2 - total number of operands.
Documentation for ws.271

vars

loc - lines of code excluding comments.

etal - number of unique operators.

eta2 - number of unique operands.

n1 - total number of operators.

n2 - total number of operands.
Documentation for the workspace ws.33S

This workspace contains two sets of data. The first set of data has 335 data points, each corresponds to a COBOL product. The second set has 35 data points which is a subset of the first one but contains detailed information about maintenance activity.

The data in the first set which are not in the second set are the "programs" which were actually modules or for which the data was not available.

The definition of a DEFECT is any report from the user concerning a malfunction or a failure to meet product specifications. The data were collected from June of 1981 to April of 1982. Most of the programs involved in this study had been "running" for at least one year prior to the data collection. The metrics were collected from a program listing in May of 1982.

The following variables each contain 335 data points.

- loc - lines of code excluding comments.
- vg - McCabe's cyclomatic complexity number.
- armyvg - vg revised by Army.
- eta1 - number of unique operators.
- eta2 - number of unique operands.
- n1 - total number of operators.
- n2 - total number of operands.
- def1 - number of defect reports from user.

The following variables each contain 335 data points.

- index - the index of the program in the 335 points.
  Note: There are two points which have the same index (115). This program had a "defect" which required 134 hours of analysis time. (The average analysis time is less than 20 hours.) After discussing this matter with the programmer from the Army, we cannot decide if this reported "defect" should indeed be considered a defect. Therefore, we singled it out to be used at the researchers discretion.

- def2 - the number of defect correction reports filed by the maintenance personnel.

- ahrs - the average analysis time (in hours) required to locate an error in the program.
chrs - the average coding time (in hours) required to fix an error in the program.

totmos - the average total experience of all programmers who corrected errors in this program.

cobmos - the average COBOL experience of all programmers who corrected errors in this program.

cscmos - the average CSC experience of all programmers who corrected errors in this program.

sysmos - the average system experience of all programmers who corrected errors in this program.

lang - the average number of languages known by programmers who corrected errors in this program.

progmos - the average number of months of experience with the program being corrected.

freq - the frequency with which the program is run. Denoted as follows:
   1 = daily  2 = weekly  3 = monthly
Documentation for ws.basic

characteristics:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>large projects</td>
<td>no. of projects: 63</td>
</tr>
<tr>
<td>Range of program size:</td>
<td>1.98 - 1,150 Kloc</td>
</tr>
<tr>
<td>Range of program effort:</td>
<td>5.9 - 11,400 person months</td>
</tr>
</tbody>
</table>

modes:

<table>
<thead>
<tr>
<th>Mode</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Organic</td>
<td>23</td>
</tr>
<tr>
<td>Semidetached</td>
<td>12</td>
</tr>
<tr>
<td>Embedded</td>
<td>28</td>
</tr>
</tbody>
</table>

Types:

<table>
<thead>
<tr>
<th>Type</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Business</td>
<td>7</td>
</tr>
<tr>
<td>Process Control</td>
<td>10</td>
</tr>
<tr>
<td>Human Machine Interactive</td>
<td>13</td>
</tr>
<tr>
<td>Scientific</td>
<td>17</td>
</tr>
<tr>
<td>Support (tools, utilities)</td>
<td>8</td>
</tr>
<tr>
<td>Systems (OS, compiler)</td>
<td>8</td>
</tr>
</tbody>
</table>

Programming Languages:

<table>
<thead>
<tr>
<th>Language</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Fortran</td>
<td>24</td>
</tr>
<tr>
<td>COBOL</td>
<td>5</td>
</tr>
<tr>
<td>Jovial</td>
<td>5</td>
</tr>
<tr>
<td>PL/1</td>
<td>4</td>
</tr>
<tr>
<td>Pascal</td>
<td>2</td>
</tr>
<tr>
<td>Other High Level</td>
<td>3</td>
</tr>
<tr>
<td>Assembly</td>
<td>20</td>
</tr>
</tbody>
</table>

vars:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>loc</td>
<td>Total delivered source instructions (in thousands)</td>
</tr>
<tr>
<td></td>
<td>includes: executable and declarative statements, job control, and format.</td>
</tr>
<tr>
<td></td>
<td>excludes: comments and unmodified utility software.</td>
</tr>
<tr>
<td>pm</td>
<td>Actual person months required for the project to be completed.</td>
</tr>
<tr>
<td></td>
<td>includes: management, programming, test and documentation.</td>
</tr>
<tr>
<td></td>
<td>excludes: user training, installation or conversion planning.</td>
</tr>
<tr>
<td>dur</td>
<td>Actual development time for the project in months</td>
</tr>
<tr>
<td></td>
<td>begin: design</td>
</tr>
<tr>
<td></td>
<td>end: integration and test</td>
</tr>
<tr>
<td>type</td>
<td>Type of project:</td>
</tr>
<tr>
<td></td>
<td>1: business application</td>
</tr>
<tr>
<td></td>
<td>2: process control</td>
</tr>
<tr>
<td></td>
<td>3: human/machine interaction</td>
</tr>
<tr>
<td></td>
<td>4: scientific application</td>
</tr>
<tr>
<td></td>
<td>5: software support</td>
</tr>
<tr>
<td></td>
<td>6: systems software</td>
</tr>
</tbody>
</table>
locadj - Adjusted thousands of delivered source instructions. Defined by \((\text{loc} \times \text{AAF})\) where AAF is the adaptation adjustment factor defined on page 134 of:


new - new \([i] = 1\) implies project \(i\) is new which implies \(\text{loc} = \text{locadj}\).
new \([i] = 0\) implies project \(i\) has reused code which implies \(\text{loc} > \text{adjloc}\).

nincdev - \(\text{nincdev} [i] = 1\) implies project \(i\) underwent nonincremental development.
\(\text{nincdev} [i] = 0\) implies project \(i\) underwent incremental development.

mode - Software development mode:
mode\([i] = 1\) implies project \(i\) is embedded mode
2 implies project \(i\) is semidetached mode
3 implies project \(i\) is organic mode
Documentation for ws.inter

This workspace contains all 15 multiplicative attributes of intermediate COCOMO model

(A) Product Attributes

rely - required software reliability
data - database size
cplx - product complexity

(B) Computer Attributes

time - execution time constraint
stor - main storage constraint
virt - virtual machine volatility
turn - computer turn around time

(C) Personnel Attributes

acap - analyst capability
aexp - applications experience
pcap - programmer capability
vexp - virtual machine experience
lexp - programming language experience

(D) Process Attributes

modp - modern programming practices
tool - use of software tools
seed - required development schedule

(E) Others

aaf - adjusted factor for modified code
year - year that the product is developed
type -
lang - language used of the product
cont -
rvol - requirements volatility
Documentation for ws.proto

This workspace contains the software metrics of seven Pascal programs written by UCLA students in a software engineering course. The first 4 programs are designed by a specifying technique. The final 3 programs are designed by a prototyping technique.

vars:

proto - index of the programming technique
    0: specifying; 1: prototyping

hour - development effort

loc - text line without comments and blank lines.

inst - delivered source instructions

tline - text line with comments and blanks

semicolon - number of semicolons in the program

comment - comment and blank lines

doc - number of pages in the documentation

np - number of persons involved

eta1 - number of unique operators.

eta2 - number of unique operands.

n1 - total number of operators.

n2 - total number of operands.

vg - McCabe's cyclomatic complexity number.

vars - number of unique variables.

mscore - maintenance score
Documentation for ws.balady

characteristics:

Large projects
No. of projects: 33
Range of program size: 4.747 - 712.362 Kloc
Range of program effort: 10 - 11758 person months

vars:

loc - thousands of lines of code.

pm - person months.

dur - duration in months.
Documentation for w.sel

characteristics:

Large projects
No. of projects: 19
Range of program size (total): 2.02 - 111.868 Kloc
Range of programming effort: 5.37 - 139.51 person months

vars:

pm - number of development person months charged to a given product
associated with management, programming and support.

locnew - thousands of "lines of codes"; refers to all new
or modified source instructions, executable or declarative,
including comments.

locall - thousands of executable or declarative source instructions for
the entire product; includes unchanged base code, new or
modified code and any copied or acquired code

dur - duration in months
begin: design
end : acceptance testing
Documentation for ws.demarco

Characteristics:

----------------------------------------
large projects
no. of projects: 18
Range of program size: 7.1 - 132.2 Kloc
Range of program effort: 3.4 - 142 person months
----------------------------------------

Entire Set:

Types: Business 11
       Human-Machine Interactive 1
       Data Base System 4
       Systems(Operating Systems, 2
       Compilers, etc)

Programming
Languages: PL/1 5
          COBOL 11
          Assembly 2

Variables:

loc - Total number of delivered lines of code,
      including job control language, data declarations
      and comments.

pm - Person Month, reflects the total effort of all project-
     related time charges, except for those associated with
     user participation, clerical support or machine operations.
     Only direct project management activity is included.

dur - Duration corresponds to the time period which begins with
      the start of system analysis and ends when all coding
      and testing has been completed.
Documentation for ws.19

characteristics:

Large Projects

No. of projects: 19

Range of program size (total): 36.60 - 666.88 Kloc

vars:

pms - number of development man months charged to a given product

(loc: this metric is scaled).

locnew - thousands of "lines of code"; refers to all new

or modified source instructions, executable or declarative,

excluding comments.

locbase - thousands of source instructions used that were unchanged

from previous versions of this product.

loccopy - thousands of source instructions copied or acquired from other

sources.

locall - total lines of codes computed as follows:

locall = locnew + loccopy + locbase.

r1-r4 - ratio of charged person months associated with

r1 - design, coding and unit testing

r2 - integration testing

r3 - publications

r4 - performance evaluation

to total development person months (pm).

dur - duration of the development of projects.

begin: design

end: first customer shipment

f test - functional test.
Documentation for ws.30

characteristics:

Large projects
No. of projects: 30
Range of program size (total): 13 - 877.20 Kloc

vars:

pms - number of development person months charged to a given product.
Note: this metric has been scaled.

locnew - thousands of "lines of codes"; refers to all new
or modified source instructions, executable or declarative,
excluding comments.

locall - thousands of executable or declarative source instructions for
the entire product; includes unchanged base code, new or
modified code and any copied or acquired code
Documentation for ws.41

This workspace is a superset of ws.19. The rest of the 22 points are a subset of ws.86.

vars:

- pms: number of development man months charged to a given product. Note: this metric is scaled.

- locnew: thousands of "lines of code"; refers to all new or modified source instructions, executable or declarative, excluding comments.

- locbase: thousands of source instructions used unchanged from previous versions of this product.

- loccopy: thousands of source instructions copied or acquired from other sources.

- loccall: total lines of codes computed as follows:
  \[ \text{loccall} = \text{locnew} + \text{locopy} + \text{locbase} \]

- r1-r4: ratio of charged person months associated with
  \[ \frac{\text{r1}}{\text{r2}} = \frac{\text{r1}}{\text{r2}} - \text{design, coding and unit testing} \]
  \[ \frac{\text{r2}}{\text{r3}} = \text{integration testing} \]
  \[ \frac{\text{r3}}{\text{r4}} = \text{publications} \]
  \[ \frac{\text{r4}}{\text{r4}} = \text{performance evaluation} \]

- dur: duration of the development of projects.
  begin: design
  end: first customer shipment

- ftest: functional test.
Documentation for ws.86

characteristics:

Large Projects
No. of projects: 86
Range of program size (total): .04 - 632,998 Kloc

vars:

pms - number of development man months charged to a given product.
Note: this metric is scaled.

dur - duration of the project developments.
begin: design
end: first customer shipment

locnew - thousands of "lines of code"; refers to all new
or modified source instructions, executable or declarative,
excluding comments.

locbase - thousands of source instructions used which were unchanged
from previous versions of this product.

loccopy - thousands of source instructions copied or acquired from other
sources.

Note: new, base, and copy broken down by languages
available; see reference below.

local1 - total lines of codes computed as such:
local1 = locnew + loccopy + locbase

r1-r4 - ratio of charged person months associated with

r1 - design, coding and unit testing
r2 - integration testing
r3 - publications
r4 - performance evaluation

to total development person months (pm).
Documentation for the workspace ws.24.a

vars

loc - declarative and executable lines of code.

vars - number of variables.

eta1 - unique number of operators.

eta2 - unique number of operands.

n1  - total number of operators.

n2  - total number of operands.

de  - number of decision points, similar to \( v(G) \).

def1 - number of defects found during formal testing.

def2 - number of defects found after project has been delivered to the customer.

et  - testing effort.

pgmr - index of programmer.
Documentation for the workspace ws.253.b*

vars

eta1 - unique number of operators.
eta2 - unique number of operands.
n1 - total number of operators.
n2 - total number of operands.
de - number of decision points, similar to \( v(G) \). It differs from \( v(G) \) in that the number of modules in a program is not included in this count.

Those variables with prefix "c" are changed metrics, which are different from the previous release.

def1 - defects found during formal testing.
def2 - defects found after delivering to the customers.

cet - testing effort.
sev - severity of def1; the larger the number, the more serious the defects are.

pgmr - index of programmer (not available for b2, b3, b4).

ind = 1  Hand Written Modules
  2  Machine Generated Modules
  3  Hand Translated Modules

There are five NEW modules in b3 which have ind = 0.

table: modules that initialize tables, which have large "eta2", but small "de".

changed: modules which have metrics that have been changed from the previous release.
Documentation for the workspace ws.639.c

vars:

eta1 - number of unique operators.
eta2 - number of unique operands.
n1 - total number of operators.
n2 - total number of operands.
de - number of decision points, similar to \( v(G) \).

Those variables with prefix "c" are changed metrics, which are different from the previous version.

def1 - defects found during formal testing.
def2 - defects found after the product has been delivered to the customer.

pgmr - index of programmer.

type = 1  Assembly Code
  2  Copy (assembly code )
  3  Macro (assembly code )
  4  PLS Code

new = 0  changed modules
  1  newly developed modules

There are approximately 1070 base modules which are NOT included in this workspace.
Documentation for workspace ws.confirm:

1) General description: The data contained in this workspace is based on data collected by observing members of the Purdue programming team while participating in the 1980 ACM Programming Contest. There were 12 participants, all of whom were experienced in FORTRAN programming.

A more detailed breakdown of the data follows:

No. of completed programs: 30  
No. of modules: 87  
No. of distinct programs: 12  
Range of programming times: 0.33 - 2.97 hrs.  
Range of program sizes: 18 - 196 lines of code  
Source language: FORTRAN

2) Integrated Model versus Physical Module Model

The metrics in this workspace include data collected using the integrated model and the physical module model. In the integrated model, measurements were made over the entire program taken as a whole. In the physical module model, each program was partitioned into modules and separate measurements were made for each module.

Those metrics with names ending with "i" refer to measurements from the integrated model, while metric names ending with "p" refer to physical module model measurements.

3) Metrics included:

a) Integrated model: All metrics are vectors of length 30, with one data point per program.

- \( \eta_{1i} \) - number of unique operators.
- \( \eta_{2i} \) - number of unique operands.
- \( n_{1i} \) - total number of operators.
- \( n_{2i} \) - total number of operands.
- \( loc_i \) - lines of code.
- \( vgi \) - McCabe's Cyclomatic Complexity number.

b) Physical module model: All metrics are vectors of length 87, with one data point per physical module, ordered by program number.

- \( \eta_{1p} \) - number of unique operators.
- \( \eta_{2p} \) - number of unique operands.
- \( n_{1p} \) - total number of unique operators.
- \( n_{2p} \) - total number of unique operands.
- \( loc_p \) - lines of code.
vgp  - McCabe's Cyclomatic Complexity number.

c) Other metrics:

  e  - effort measures
  30 X 5 array with the following data:

  col 1: Software Science effort for integrated model.
  col 2: Software Science effort for physical model.
  col 3: effort for Woodfield logical module model.
  col 4: effort for Woodfield syntactic model.
  col 5: effort for Woodfield semantic model.

Notes:
1) Software Science effort is calculated
   using the following equation:

   \[ \text{effort} = \frac{(c_{t1} X n_2 X n X \log cta)}{(2 X cta^2)} \]
   where \( n = n_1 + n_2 \), and \( cta = c_{t1} + cta_2 \).

2) Software Science effort for physical module model
   is calculated by simply summing up the efforts
   required for each physical module within each
   program.

3) See references for information regarding
   Woodfield's effort measures.

hour - total time.
rl - ratio of design effort to total effort.
r2 - ratio of debug effort to total effort.

4) Accessing Metrics

   a) For the integrated model metrics, access data
      directly by indexing. For example, \( cta2i[3] \) returns the
      unique operand count of program 3.

   b) For the physical module model metrics, values
      for each physical module within a specified
      program can be obtained by using function "listp".
      The calling sequence is: \( n \ listp \ metric \)
      where \( n \) specifies the program number, and
      "metric" is the desired physical module metric
      (must end in "p"). For example, \( 5 \ listp \ locp \) returns
      the lines of code for each module in program 5.

5) Utility variable:

   pindex - 30 X 2 array.

   Used by listp to index physical module data.
Column 1 contains the number of physical modules in each program. Column 2 contains the location of the first physical module for each program.

6) References: for further information, see

Documentation for workspace ws.develop:

1) General description: The data contained in this workspace is based on data collected by observing the practice sessions conducted for potential members of the Purdue programming team which participated in the 1980 ACM Programming Contest. The participants were all students experienced in FORTRAN programming.

A more detailed breakdown of the data follows:

No. of completed programs: 33
No. of modules: 143
No. of distinct programs: 10
Range of programming times: 0.9 - 4.83 hrs.
Range of program sizes: 57 - 265 lines of code
Source language: FORTRAN

2) Integrated Model versus Physical Module Model

The metrics in this workspace include data collected using the integrated model and the physical module model. In the integrated model, measurements were made over the entire program taken as a whole. In the physical module model, each program was partitioned into modules and separate measurements were made for each module.

Those metrics with names ending with "i" refer to measurements from the integrated model while metric names ending with "p" refer to physical module model measurements.

3) Metrics included:

a) Integrated model: All metrics are vectors of length 33, with one data point per program.

\( \eta_1 \) - number of unique operators.
\( \eta_2 \) - number of unique operands.
\( n_1 \) - total number of operators.
\( n_2 \) - total number of operands.
\( loc \) - lines of code.
\( vgi \) - McCabe's Cyclomatic Complexity number.

b) Physical module model: All metrics are vectors of length 143, with one data point per physical module and ordered by program number.

\( \eta_1 p \) - number of unique operators.
\( \eta_2 p \) - number of unique operands.
\( n_1 p \) - total number of operators.
\( n_2 p \) - total number of operands.
locp - lines of code.
vgp - McCabe's Cyclomatic Complexity number.

c) Other metrics:

- effort measures
  33 x 5 array with the following data:
  
col 1: Software Science effort for integrated model.
col 2: Software Science effort for physical model.
col 3: effort for Woodfield logical module model.
col 4: effort for Woodfield syntactic model.
col 5: effort for Woodfield semantic model.

Notes:
1) Software Science effort is calculated using the following equation:
   effort = (eta1 x n2 x n x log eta)/(2 x eta2)
   where n = n1 + n2, and eta = eta1 + eta2.
2) Software Science effort for physical module model is calculated by summing the efforts required for each physical module within each program.
3) See references for information regarding Woodfield's effort measures.

hour - total observed development time (in hours), there are 33 data points.

4) Accessing Metrics

a) For the integrated model metrics, access data directly by indexing.

   e.g. eta2i[3] returns the unique operand count for program 3.

b) For the physical module model metrics, the value for all the physical modules within a specified program can be obtained by using function "listp"

   The calling sequence is:
   n listp metric

   where n specifies the program number
   metric is the desired physical module metric
   (must end in "p")

   e.g. 5 listp locp lists the lines of code for each module in program 5
5) Utility variable:

\[ \text{pindex} - 33 \times 2 \text{ array.} \]

Used by listp to index physical module data.
Column 1 contains the number of physical modules in each program.
Column 2 contains the location of the first physical module for each program.

6) References: for further information, see

Documentation for ws.imsl

characteristics:

Large program
No. of program modules: 497
Range of module size: 6 - 413 loc
Range of programming time: N/A

vars:

eta1 - number of unique operators.
eta2 - number of unique operands.
01 - total number of operators.
02 - total number of operands.
vg - McCabe Cyclomatic Complexity number.
loc - lines of code.

Note: each of the 497 subroutines was counted as an independent module.

imslname - module names
Documentation for ws.acm

characteristics:

small programs
No. of programs: 23
Range of program size: 29 - 102 loc
Range of programming time: 2.96 - 5.835 hrs

vars:

eta1 - number of unique operators.
eta2 - number of unique operands.
n1 - total number of operators.
n2 - total number of operands.
v_g - McCabe's cyclomatic complexity number v(G).
hour - effort in hours.
loc - lines of code.

Note: The above counts were obtained using Woodfield's integrated-model counting rule.

hourp - effort using Woodfield's physical-module-model counting rule.

hourl - effort using Woodfield's logical-module-model counting rule.

name - program names.
Documentation for ws.paslib

vars:

loc   - lines of executable and declarative code.
vars  - number of unique variables.
eta1  - number of unique operators.
eta2  - number of unique operands.
n1    - total number of total operators.
n2    - total number of total operands.
Documentation for ws.cs440

vars:

loc - lines of executable and declarative code

vars - number of unique variables

eta1 - number of unique operators

eta2 - number of unique operands

n1 - number of total operators

n2 - number of total operands
Documentation for ws.cs590c

In the summer of 1982, 21 programmers were involved in a software experiment, in which each programmer wrote a program. There were 121 versions of these 21 programs.

The following variables are related to program versions, each containing 121 data points.

- **loc** - lines of executable and declarative code.
- **vg** - McCabe's cyclomatic complexity number.
- **vars** - number of unique variables.
- **eta1** - number of unique operators.
- **eta2** - number of unique operands.
- **n1** - total number of operators.
- **n2** - total number of operands.
- **xhours** - programming hours, excluding documentation efforts.
- **e** - estimated measures by programmers.

The following variables are related to individual programs, each containing 21 data points.

- **index** - program type index
  - 1: huffman code implementation
  - 2: Desk calculator simulation
  - 3: Lisp interpreter
  - 4: Database language translator
- **first** - The first version of the successive program versions.
- **number** - number of program versions for each program.
- **design** - the version of the program at end of design.
- **fcc** - the version of the program after the first clean compile.

Example:

- **loc[design]**: lines of code after design phase.
- **vars[first+number-1]**: final count of the number of variables.
- **exhours[first]**: estimated development hours by the programmers at the early stage of the design.
Documentation for ws.pret

vars:

cloze - cloze procedure scores (50 cloze blanks).
extend - extended cloze procedure scores (in minutes).
quiz  - compression quiz (10 questions).
Documentation for ws.calc

vars:

case  - number of unique test cases used.
ep    - programming effort (in hours).

xep   - programming effort excluding documentation effort.
xepdb - programming hours to complete database program.

et    - testing effort in minutes (effort of running test cases).

ef    - fixing effort in minutes
        (effort to detect and correcting defects).

loc   - lines of executable and declarative code.

vg    - McCabe's cyclomatic complexity number.

vars  - number of unique variables.

eta1  - number of unique operators.
eta2  - number of unique operands.

n1    - total number of unique operators.
n2    - total number of unique operands.

def1o - number of defects before formal testing (original).
def1y - number of defects before formal testing (revised by Yu).
def1p - number of defects before formal testing (revised by Pasch).

def2o - number of defects after formal testing (original).
def2y - number of defects after formal testing (revised by Yu).
def2p - number of defects after formal testing (revised by Pasch).

dfndy - number of defects found during formal testing (revised by Yu).
        Note: dfndy is NOT (def1y - def2y).
        This is because new defects may be introduced during
        the formal testing.
Documentation for ws.calc.d

In the summer of 1983, 44 programmers were involved in a software experiment, in which each programmer wrote two programs. The data collected in this workspace were from the first program, called the CALCulator program. There are 328 program versions for these 44 programs.

The following variables are related to program versions, each contains 328 data points:

- loc - lines of executable and declarative code.
- vg - McCabe’s cyclomatic complexity number.
- vars - number of unique variables.
- eta1 - number of unique operators.
- eta2 - number of unique operands.
- n1 - total number of operators.
- n2 - total number of operands.
- xhours - programming hours, excluding documentation effort.

- c* - estimated measures by programmers.

The following variables are related to individual programs, each contains 44 data points:

- first - The first version of the successive program versions.
- number - number of program versions of each program.
- design - the version of the program at end of design.

Example:

- loc[design] : lines of code after design phase.
- vars[first+number-1] : final count of the number of variables.
- exhours[first] : estimated development hours by the programmers at the early stage of the design.
Documentation for ws.dbl

vars:

- **case** - number of unique test cases used.
- **ep** - programming effort (in hours).
- **xep** - programming effort excluding documentation efforts.
- **et** - testing effort in minutes (effort to run test cases).
- **ef** - fixing effort in minutes (effort of correcting and detection errors).

- **loc** - lines of executable and declarative code.
- **vg** - McCabe’s cyclomatic complexity number.
- **vars** - number of unique variables.

- **ota1** - number of unique operators.
- **ota2** - number of unique operands.
- **n1** - total number of operators.
- **n2** - total number of operands.

- **def1** - number of defects before formal testing.
- **defi1** - number of incorrect input echos before formal testing.
- **def1o** - number of incorrect output codes before formal testing.
- **def1c** - number of defects of CPU abort before formal testing.

- **def2** - number of defects after formal testing.
- **def2i** - number of incorrect input echos after formal testing.
- **def2o** - number of incorrect output codes after formal testing.
- **def2c** - number of CPU abort defects after formal testing.

Note: def1 = defi1 + def1o + def1c
      def2 = def2i + def2o + def2c
Documentation for ws.dbl.d

In the summer of 1983, 44 programmers were involved in a software experiment in which each programmer wrote two programs. The data collected in this workspace were from the second program, called Database Language program (DBL). There were 259 versions of these 44 programs.

The following variables are related to program versions, i.e. their ranks are 259.

- loc - lines of executable and declarative code.
- vg - McCabe's cyclomatic complexity number.
- vars - number of unique variables.
- eta1 - number of unique operators.
- eta2 - number of unique operands.
- n1 - total number of operators.
- n2 - total number of operands.
- xhours - programming hours, excluding documentation efforts.
- e* - estimated measures by programmers.

The following variables are related to individual programs, each containing 44 data points.

- first - first version of the successive program versions.
- number - number of program versions of each program.
- design - version of the program at end of design.

Example:

loc[ design ] : lines of code after design phase.
vars[ first+number-1 ] : final count of the number of variables.
exhours[ first ] : estimated development hours by the programmers at the early stage of the design.