```
%%-*- text -*-
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This is a PROMISE Software Engineering Repository data set made publicly
% available in order to encourage repeatable, verifiable, refutable, and/or
% improvable predictive models of software engineering.
%
% If you publish material based on PROMISE data sets then, please
% follow the acknowledgment guidelines posted on the PROMISE repository
% web page http://promise.site.uottawa.ca/SERepository .
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% 1. Title/Topic: cocomo81/software cost estimation

@relation  cocomo81

% 2. Sources:
%       -- Boehm's 1981 text, transcribed by Srinivasan and Fisher
%             @Book{boehm81,
%               Author =        "B. Boehm",
%               Title =         "Software Engineering Economics",
%               Publisher =     "Prentice Hall",
%               Year =  1981}
%
%           then converted by Tim Menzies from
%           http://www.vuse.vanderbilt.edu/~dfisher/tech-reports/raw-TSE-95
%           to arff fort.
%
%        -- Donor: Tim Menzies tim@barmag.net
%
%     -- Date: December 2 2004

% 3. Past Usage
%     1. K. Srinivasan and D. Fisher, Machine Learning Approaches
%        to Estimating Software Development Effort, IEEE Trans. Soft. Eng.,
%        Feb 1995, pages 126--137

%  4. Relevant Information
%
%    The COCOMO software cost model measures effort in calendar months
%    of 152 hours (and includes development and management hours).
%    COCOMO assumes that the effort grows more than linearly on
%    software size; i.e. months=a* KSLOC^b*c. Here, "a" and "b" are
%    domain-specific parameters; "KSLOC" is estimated directly or
%    computed from a function point analysis; and "c" is the product
%    of over a dozen "effort multipliers". I.e.
%
%    months=a*(KSLOC^b)*(EM1* EM2 * EM3 * ...)
%
%    The effort multipliers are as follows:
%
%     increase | acap | analysts capability
%     these to | pcap | programmers capability
%    decrease  | aexp | application experience
%      effort  | modp | modern programing practices
%              | tool | use of  software tools
%              | vexp | virtual machine experience
%              | lexp | language experience
%    ----------+------+---------------------------
%              | sced | schedule constraint
%    ----------+------+---------------------------
%     decrease | stor | main memory constraint
%     these to | data | data base size
%     decrease | time | time constraint for cpu
%       effort | turn | turnaround time
%              | virt | machine volatility
%              | cplx | process complexity
%              | rely | required software reliability
%
%    In COCOMO I, the exponent on KSLOC was a single value ranging from
```

```
%    1.05 to 1.2.  In COCOMO II, the exponent "b" was divided into a
%    constant, plus the sum of five "scale factors" which modeled
%    issues such as ``have we built this kind of system before?''.  The
%    COCOMO~II effort multipliers are similar but COCOMO~II dropped one
%    of the effort multiplier parameters; renamed some others; and
%    added a few more (for "required level of reuse", "multiple-site
%    development", and "schedule pressure").
%
%    The effort multipliers fall into three groups: those that are
%    positively correlated to more effort; those that are
%    negatively correlated to more effort; and a third group
%    containing just schedule information. In COCOMO~I, "sced" has a
%    U-shaped correlation to effort; i.e. giving programmers either
%    too much or too little time to develop a system can be
%    detrimental.
%
%    The standard numeric values of the effort multipliers are:
%
%               very                              very    extra   productivity
%                       low     low     nominal high    high    high    range
%    ------------------------------------------------------------------------
%    acap       1.46    1.19    1.00    0.86    0.71            2.06
%    pcap       1.42.   1.17    1.00    0.86    0.70            1.67
%    aexp       1.29    1.13    1.00    0.91    0.82            1.57
%    modp       1.24.   1.10    1.00    0.91    0.82            1.34
%    tool       1.24    1.10    1.00    0.91    0.83            1.49
%    vexp       1.21    1.10    1.00    0.90                    1.34
%    lexp       1.14    1.07    1.00    0.95                    1.20
%    sced       1.23    1.08    1.00    1.04    1.10            e
%    stor                       1.00    1.06    1.21    1.56    -1.21
%    data                0.94   1.00    1.08    1.16            -1.23
%    time                       1.00    1.11    1.30    1.66    -1.30
%    turn               0.87    1.00    1.07    1.15            -1.32
%    virt               0.87    1.00    1.15    1.30            -1.49
%    cplx       0.70    0.85    1.00    1.15    1.30    1.65    -1.86
%    rely       0.75     0.88    1.00    1.15    1.40            -1.87
%
%    These were learnt by Barry Boehm after a regression analysis of the
%    projects in the COCOMO I data set.
%      @Book{boehm81,
%        Author    =     "B. Boehm",
%        Title     =     "Software Engineering Economics",
%        Publisher =     "Prentice Hall",
%        Year      =     1981}
%
%     The last column of the above table shows max(E)/min(EM) and shows
%     the overall effect of a single effort multiplier. For example,
%     increasing "acap" (analyst experience) from very low to very
%     high will most decrease effort while increasing "rely"
%     (required reliability) from very low to very high will most
%     increase effort.
%
%     There is much more to COCOMO that the above description. The
%     COCOMO~II text is over 500 pages long and offers
%     all the details needed to implement data capture and analysis of
%     COCOMO in an industrial context.
%       @Book{boehm00b,
%         Author = "Barry Boehm and Ellis Horowitz and Ray Madachy and
%                   Donald Reifer and Bradford K. Clark and Bert Steece
%                   and A. Winsor Brown and Sunita Chulani and Chris Abts",
%          Title = "Software Cost Estimation with Cocomo II",
%      Publisher = "Prentice Hall",
%           Year = 2000,
%           ibsn = "0130266922"}
%
%     Included in that book is not just an effort model but other
%     models for schedule, risk, use of COTS, etc.  However, most
%     (?all) of the validation work on COCOMO has focused on the effort
%     model.
```

```
%       @article{chulani99,
%         author =       "S. Chulani and B. Boehm and B. Steece",
%         title =        "Bayesian Analysis of Empirical Software Engineering
%                          Cost Models",
%         journal =      "IEEE Transaction on Software Engineering",
%         volume =       25,
%         number =       4,
%         month =        "July/August",
%         year =         "1999"}
%
%     The value of an effort predictor can be reported many ways
%     including MMRE and PRED(N).MMRE and PRED are computed from the
%     relative error, or RE, which is the relative size of the
%     difference between the actual and estimated value:
%
%     RE.i = (estimate.i - actual.i) / (actual.i)
%
%     Given a data set of of size "D", a "Train"ing set of size
%     "(X=|Train|) <= D", and a "test" set of size "T=D-|Train|", then
%     the mean magnitude of the relative error, or MMRE, is the
%     percentage of the absolute values of the relative errors,
%     averaged over the "T" items in the "Test" set; i.e.
%
%     MRE.i  = abs(RE.i)
%     MMRE.i = 100/T*( MRE.1 + MRE.2 + ... + MRE.T)
%
%     PRED(N) reports the average percentage of estimates that were
%     within N% of the actual values:
%
%     count = 0
%     for(i=1;i<=T;i++) do if MRE.i <= N/100 then count++ fi done
%     PRED(N) = 100/T * count
%
%     For example, e.g. PRED(30)=50% means that half the estimates are
%     within 30% of the actual.  Shepperd and Schofield comment that
%     "MMRE is fairly conservative with a bias against overestimates
%     while Pred(25) will identify those prediction systems that are
%     generally accurate but occasionally wildly inaccurate".
%       @article{shepperd97,
%         author="M. Shepperd and C. Schofield",
%         title="Estimating Software Project Effort Using Analogies",
%         journal="IEEE Transactions on Software Engineering",
%         volume=23,
%         number=12,
%         month="November",
%         year=1997,
%         note="Available from
%                \url{http://www.utdallas.edu/~rbanker/SE_XII.pdf}"}

%  5. Number of instances: 63

%  6. Number of attributes: 17 (all numeric: 15 for the effort multipliers,
%       one for LOC and one for actual development effort.

%  7. Attribute information:

@attribute rely numeric
@attribute data numeric
@attribute cplx numeric
@attribute time numeric
@attribute stor numeric
@attribute virt numeric
@attribute turn numeric
@attribute acap numeric
@attribute aexp numeric
@attribute pcap numeric
@attribute vexp numeric
@attribute lexp numeric
@attribute modp numeric
```

```
@attribute tool numeric
@attribute sced numeric
@attribute loc numeric
@attribute actual numeric

% 8. Missing attributes: none

% 9: Class distribution: the class value (actual) is continuous.
%    After sorting all the instances on actual, the following
%    distribution was found:
%    Instances     Range
%    ---------  ----------------
%      1 .. 10    5.9 ..    15
%     11 .. 20    18  ..    47
%     21 .. 30    50  ..    87
%     31 .. 40    88  ..   218
%     41 .. 50   230  ..   539
%     51 .. 60   605  ..  2455
%     61 .. 63  6400  .. 11400

@data

0.88,1.16,0.7,1,1.06,1.15,1.07,1.19,1.13,1.17,1.1,1,1.24,1.1,1.04,113,2040
0.88,1.16,0.85,1,1.06,1,1.07,1,0.91,1,0.9,0.95,1.1,1,1,293,1600
1,1.16,0.85,1,1,0.87,0.94,0.86,0.82,0.86,0.9,0.95,0.91,0.91,1,132,243
0.75,1.16,0.7,1,1,0.87,1,1.19,0.91,1.42,1,0.95,1.24,1,1.04,60,240
0.88,0.94,1,1,1,0.87,1,1,1,0.86,0.9,0.95,1.24,1,1,16,33
0.75,1,0.85,1,1.21,1,1,1.46,1,1.42,0.9,0.95,1.24,1.1,1,4,43
0.75,1,1,1,1,0.87,0.87,1,1,1,0.9,0.95,0.91,0.91,1,6.9,8
1.15,0.94,1.3,1.66,1.56,1.3,1,0.71,0.91,1,1.21,1.14,1.1,1.1,1.08,22,1075
1.15,0.94,1.3,1.3,1.21,1.15,1,0.86,1,0.86,1.1,1.07,0.91,1,1,30,423
1.4,0.94,1.3,1.11,1.56,1,1.07,0.86,0.82,0.86,0.9,1,1,1,1,29,321
1.4,0.94,1.3,1.11,1.56,1,1.07,0.86,0.82,0.86,0.9,1,1,1,1,32,218
1.15,0.94,1.3,1.11,1.06,1,1,0.86,0.82,0.86,1,0.95,0.91,1,1.08,37,201
1.15,0.94,1.3,1.11,1.06,1.15,1,0.71,1,0.7,1.1,1,0.82,1,1,25,79
1.15,0.94,1.65,1.3,1.56,1.15,1,0.86,1,0.7,1.1,1.07,1.1,1.24,1.23,3,60
1.4,0.94,1.3,1.3,1.06,1.15,0.87,0.86,1.13,0.86,1.21,1.14,0.91,1,1.23,3.9,61
1.4,1,1.3,1.3,1.56,1,0.87,0.86,1,0.86,1,1,1,1,1,6.1,40
1.4,1,1.3,1.3,1.56,1,0.87,0.86,0.82,0.86,1,1,1,1,1,3.6,9
1.15,1.16,1.15,1.3,1.21,1,1.07,0.86,1,1,1,1,1.24,1.1,1.08,320,11400
1.15,1.08,1,1.11,1.21,0.87,0.94,0.71,0.91,1,1,1,0.91,0.91,1,1150,6600
1.4,1.08,1.3,1.11,1.21,1.15,1.07,0.71,0.82,1.08,1.1,1.07,1.24,1,1.08,299,6400
1,1.16,1.15,1.06,1.14,0.87,0.87,0.86,1,1,1,1,0.91,0.91,1,252,2455
1.15,1,1,1.27,1.06,1,1,0.86,0.82,0.86,0.9,1,0.91,1,1.23,118,724
1.15,1,1,1.08,1.06,1,1,0.86,0.82,0.86,0.9,1,1,1,1.23,77,539
0.88,1,0.85,1.06,1.06,1,0.87,1,1.29,1,1.1,0.95,0.82,0.83,1,90,453
1.15,1.16,1.3,1.15,1.06,1,0.87,0.86,1,0.86,1.1,1,0.82,0.91,1.08,38,523
0.94,1,0.85,1.07,1.06,1.15,1.07,0.86,1,0.86,1.1,1,0.91,1.1,1.08,48,387
1.15,0.94,1.15,1.35,1.21,1,0.87,1,1,1,1,1,0.82,1.1,1.08,9.4,88
1.15,1.08,1.3,1.11,1.21,1.15,1.07,0.86,1,0.86,1.1,1.07,1.1,1.1,1,13,98
0.88,1,1,1,1,1,1,1.1,1.29,0.86,1,1,0.91,0.91,1.23,2.14,7.3
0.88,1,1,1,1,1,1,1,1.29,0.86,1,1,0.91,0.91,1.23,1.98,5.9
1.4,1.08,1,1.48,1.56,1.15,1.07,0.86,0.82,0.86,1.1,1.07,1,1,1,62,1063
0.88,1.08,0.85,1,1,1,1,0.71,0.82,1,1,1,1.1,1.1,1,390,702
1.4,1.08,1.3,1.48,1.56,1.15,0.94,0.86,0.82,0.86,0.9,1,0.91,0.91,1,42,605
1.15,1.08,1,1.06,1,1,0.87,1,1,1,1,1,0.91,1.1,1.23,23,230
0.75,0.94,1.3,1.06,1.21,1.15,1,1,0.91,1,1.1,1,1.24,1.24,1,13,82
0.88,1.05,0.81,1,1,0.87,0.87,1.19,1,1.17,0.9,0.95,1,0.91,1.04,15,55
0.88,0.94,0.7,1,1.06,1,1,0.86,0.82,0.86,1,1,1,1,1,60,47
1,1,1.15,1,1,0.87,0.87,0.71,0.91,1,0.9,0.95,0.82,0.91,1,15,12
1,1,1.15,1,1,0.87,1,0.71,0.82,0.7,1,0.95,0.91,1.1,1,6.2,8
1,0.94,1.3,1,1,1,0.87,0.86,0.82,1.17,1,1,1.1,1,1,3,8
0.88,0.94,1,1,1,0.87,0.87,1,0.82,0.7,0.9,0.95,0.91,0.91,1,5.3,6
0.88,1.04,1.07,1,1.06,0.87,1.07,0.86,1,0.93,0.9,0.95,0.95,0.95,1.04,45.5,45
1,1.04,1.07,1,1.21,0.87,1.07,0.86,1,1,0.9,0.95,1,1,1.04,28.6,83
0.88,1.04,1.07,1.06,1.21,0.87,1.07,1,1,1,0.9,0.95,1.1,1,1.04,30.6,87
0.88,1.04,1.07,1,1.06,0.87,1.07,1,1,1,0.9,0.95,1,0.95,1.04,35,106
0.88,1.04,1.07,1,1.06,0.87,1.07,1,1,0.86,0.9,0.95,1,1,1.04,73,126
0.75,0.94,1.3,1,1,0.87,0.87,0.71,0.82,0.7,1.1,1.07,1.1,1,1.04,23,36
```

```
0.88,0.94,0.85,1,1,0.87,1,1.19,0.91,1.17,0.9,0.95,1.1,1,1.04,464,1272
1,1,0.85,1,1,1,0.87,0.71,1,0.7,1.1,1,0.82,0.91,1,91,156
1.15,1,1,1.3,1.21,1,0.87,0.86,1,0.86,1.1,1,1,1,1,24,176
0.88,1,1,1,1,1,1.15,1.19,1,1.42,1,0.95,1.24,1.1,1.04,10,122
0.88,0.94,0.85,1,1.06,1.15,1,1,1,1,1.1,1.07,1.24,1.1,1,8.2,41
0.88,0.94,1.15,1.11,1.21,1.3,1,0.71,1,0.7,1.1,1.07,1,1.1,1.08,5.3,14
1,0.94,1,1,1.06,1.15,0.87,1,0.82,1,1,0.95,0.91,1.1,1,4.4,20
0.88,0.94,0.7,1,1,0.87,0.87,0.86,0.82,1.17,0.9,0.95,1.1,1,1,6.3,18
1.15,0.94,1.3,1.3,1.21,1,1,0.86,0.91,1,1.1,1.07,1.1,1.1,1.08,27,958
1,0.94,1.15,1.11,1.21,1.3,1,1,1,1,1.1,1.07,1.1,1.1,1.23,17,237
1.4,0.94,1.3,1.66,1.21,1,1,0.71,0.82,0.7,0.9,0.95,0.91,1,1,25,130
1,0.94,1.15,1.06,1.06,1,0.87,1,1,1,1,1,0.91,1,1,23,70
1.15,0.94,1.3,1.11,1.06,1,1,0.86,1.13,0.86,1.1,1.07,1.1,1.1,1.08,6.7,57
1,0.94,1.15,1,1,0.87,0.87,0.86,1,0.86,0.9,1,0.82,1,1,28,50
0.88,0.94,1.3,1.11,1.21,1.15,1,0.78,0.82,0.7,1.21,1.14,0.91,1.24,1,9.1,38
1,0.94,1.15,1,1,1,0.87,0.71,0.82,0.86,1,1,0.82,1,1,10,15
```