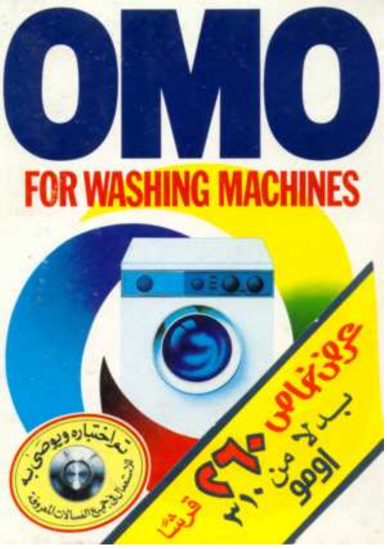

# OMO:
# A Prolog-based Software Cost Estimation Tool

**Tim Menzies**[1], **Santa Clause**[2]

[1] Lane Department of Computer Science, University of West Virginia, PO Box 6109, Morgantown, WV, 26506-6109, USA;
`http://tim.menzies.com`; e-mail: `tim@menzies.com`
[2] Artic Software Systems

Wp ref: tim/src/pl/cocomo/cocomo December 15, 2002

**Abstract** COCOMO is a software effort estimation tool. OMO is COCOMO written in SWI-Prolog [5]. and documented using TEX4LOG.

## Contents

## 1 Introduction

### 1.1 About COCOMO

The COCOMO project aims at developing an open-source, public-domain software effort estimation model. The project has collected information on 161 projects from commercial, aerospace, government, and non-profit organizations [1, 4]. As of 1998, the projects represented in the database were of size 20 to 2000 KSLOC (thousands of lines of code) and took between 100 to 10000 person months to build.

COCOMO measures effort in calendar months where one month is 152 hours (and includes development and management hours). The core intuition behind COCOMO-based estimation is that as systems grow in size, the effort required to create them grows exponentially, i.e. $effort \propto KSLOC^x$. More precisely:

$$months = a * \left(KSLOC^{\left(0.91+\sum_{i=1}^{5} SF_i\right)}\right) * \left(\prod_{j=1}^{17} EM_j\right)$$

where $a$ is a domain-specific parameter, and KSLOC is estimated directly or computed from a function point analysis. $SF_i$ are the scale factors (e.g. factors such as have we built this kind of system before?") and $EM_j$ are the cost drivers (e.g. required level of reliability). Figure 1 lists the scale drivers and effort multipliers.

Software effort-estimation models like COCOMO-II should be tuned to their local domain. Off-the-shelf untuned" models have been up to 600% inaccurate in their estimates, e.g. [3, p165] and [2]. However, tuned models can be far more accurate. For example, [1] reports a study with a bayesian tuning algorithm using the COCOMO project database. After bayesian tuning, a cross-validation study showed that COCOMO-II model produced estimates that are within 30% of the actuals, 69% of the time.

| × | | | √ | | | √√ | | |
|---|---|---|---|---|---|---|---|---|
| 2000 | ga | 1983 | 2000 | ga | 1983 | 2000 | ga | 1983 |
| - | tool | - | tool | - | tool | team | team | team |
| time | time | time | site | site | site | prec | prec | prec |
| stor | stor | stor | sced | sced | sced | pmat | pmat | pmat |
| ruse | ruse | ruse | - | pvol | - | flex | flex | flex |
| rely | rely | rely | pexp | pexp | pexp | arch | arch | arch |
| pvol | - | pvol | pcon | pcon | pcon | | | |
| docu | docu | docu | pcap | pcap | pcap | | | |
| data | data | data | ltex | ltex | ltex | | | |
| cplx | cplx | cplx | aexp | aexp | aexp | | | |
| | | | acap | acap | acap | | | |

### 1.2 Structure of this document

My Prolog code descriptions have the following format:

1. Motivation: why is this system being built?
2. Samples: with this system, what kind of things can a user do? (may include sample inputs/outputs).
3. High-level walk through: before the code is revealed in all its glory, an abstract description of its unique features and architecture is of much benefit.
4. Examples (longer samples): Detailed inputs/outputs; graphs of experimental results; discussion; future work section.

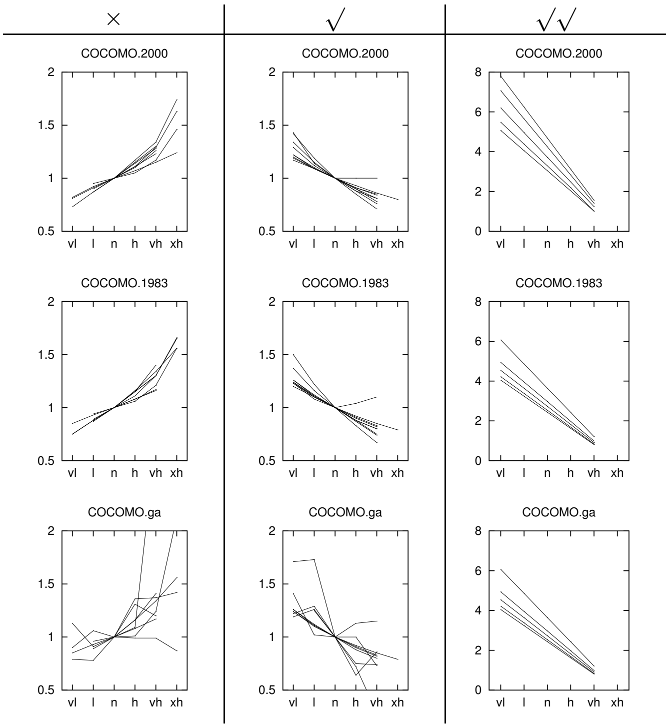

**Fig. 2** Influence of different COCOMO parameters

5. Code: All the Prolog
6. An appendix with acknowledgements, references, and licensing details.

The code section is in three parts:

1. Shell
2. Shell tart-up code (must be loaded into Prolog)
3. Knowledge base

The shell divides up as follows:

1. Initializations:
   - Operator definitions (must be first).
   - Flags (these can usually go just before the start-up actions but, for safety's sake, we place them at the front).
   - hooks (into the Prolog reader)
   - hacks (shameful things we'd rather hide).
2. Library code which, ideally, should be good for more than just this application.
3. The actual system code.

Note that for exposition purposes, it is often to load the library, then load and explain the system code, then expalin the library.

## 2 Initializations

### 2.1 Operators

```
:- op(1,xfx, to).
:- op(700,xfx, :=).
```

| Type | Acronym | Definition | Low-end | Medium | High-end |
|---|---|---|---|---|---|
| EM | acap | analyst capability | worst 15% | 55% | best 10% |
| EM | aexp | applications experience | 2 months | 1 year | 6 years |
| SF | arch | architecture or risk resolution | few interfaces defined or few risk eliminated | most interfaces defined or most risks eliminated | all interfaces defined or all risks eliminated |
| EM | cplx | product complexity | e.g. simple read/write statements | e.g. use of simple interface widgets | e.g. performance-critical embedded systems |
| EM | data | database size<br>(DB bytes/ Program SLOC) | 10 | 100 | 1000 |
| EM | docu | documentation | many life-cycle phases not documented | | extensive reporting for each life-cycle phase |
| SF | flex | development flexibility | development process rigorously defined | some guidelines, which can be relaxed | only general goals defined |
| EM | ltex | language and tool-set experience | 2 months | 1 year | 6 years |
| EM | pcap | programmer capability | worst 15% | 55% | best 10% |
| EM | pcon | personnel continuity<br>(% turnover per year) | 48% | 12% | 3% |
| EM | pexp | platform experience | 2 months | 1 year | 6 years |
| SF | pmat | process maturity | CMM level 1 | CMM level 3 | CMM level 5 |
| SF | prec | precedentedness | we have never built this kind of software before | somewhat new | thoroughly familiar |
| EM | pvol | platform volatility $(\frac{frequency\ of\ major\ changes}{frequency\ of\ minor\ changes})$ | $\frac{12\ months}{1\ month}$ | $\frac{6\ months}{2\ weeks}$ | $\frac{2\ weeks}{2\ days}$ |
| EM | rely | required reliability | errors mean slight inconvenience | errors are easily recoverable | errors can risk human life |
| EM | ruse | required reuse | none | across program | across multiple product lines |
| EM | sced | dictacted development schedule | deadlines moved closer to 75% of the original estimate | no change | deadlines moved back to 160% of the original estimate |
| EM | site | multi-site development | some contact: phone, mail | some email | interactive multi-media |
| EM | stor | main storage constraints<br>(% of available RAM) | N/A | 50% | 95% |
| SF | team | team cohesion | very difficult interactions | basically co-operative | seamless interactions |
| EM | time | execution time constraints<br>(% of available CPU) | N/A | 50% | 95% |
| EM | tool | use of software tools | edit,code,debug | | well intergrated with lifecycle |

**Fig. 1** Parameters of the COCOMO-II effort risk model; adapted from `http://sunset.usc.edu/COCOMOII/expert_cocomo/drivers.html`. `Stor`” and `time`” score N/A”” for low-end values since they have no low-end defined in COCOMO-II. SF” denotes scale factors” and EM” denotes effort multipliers”.

### 2.2 *Flags*

```
:- dynamic       option/2.
:- discontiguous option/2,lookUp1/4.
:- index(lookUp1(1,1,1,0)).
```

### 2.3 *Hooks*

Fast assertions of named variables.

```
term_expansion((X;Y :- Z),Out) :-
    multis(((X;Y) :- Z),Out).
```

Define tabular material

```
term_expansion(Table = Cols + Rows , Out) :- !,
    lookUpTable(Table=Cols+Rows,Out).
```

Set global options.

```
term_expansion(A=B,[]) :- set(A=B).
```

Instantiate named fields

```
term_expansion(Functor is Fields,Out) :-
    fields(Fields,Functor,Out).
```

### 2.4 *Hacks*

No hacks (yet).

### 2.5 *Library (load)*

Loaded here, explained later in §6.

```
:- [lib].
```

## 3 System code

### 3.1 *Main driver*

```
estimate :-
    cocomo(Coc),
    estimate(Pm,Staff,Months),
    format('COCOMO.~p says ~p months (total);',[Coc,Pm]),
    format('~p staff over ~p months\n', [Staff,Months]).

estimate(Pm,Staff,Months) :-
    tdev(Tdev),
    pm(Pm0),
    Pm is Pm0,
    Staff is ceiling(Pm/Tdev),
    Months is ceiling(Tdev),
    !.
```

### 3.2 *Equations*

#### 3.2.1 *Sizing equations*

```
size((1 + (R/100)) *(N + E)) :-
    revl(R), newKsloc(N), equivalentKsloc(E).

equivalentKsloc(Ak*Aam*(1-(At/100))) :-
    adaptedKsloc(Ak), at(At), aam(Aam).

aam(Am) :- aaf(Af), compare(C,Af,50), aam1(C,Af,Am).

aam1(=,Af, X) :- aam1(<,Af,X).
aam1(>,Af, (Aa+Af+(Su*U))/100) :- aa(Aa),su(Su),unfm(U).
aam1(<,Af,((Aa+Af*(1+(0.02*Su*U)))/100)) :-
    aa(Aa), su(Su), unfm(U).

aaf(0.4*Dm+0.3*Cm+0.3*Im) :- dm(Dm), cm(Cm), im(Im).
```

#### 3.2.2 *Schedule Equations*

```
tdev((C*(P^F))*SP/100) :-
    c(C), pmNs(P), f(F), scedPercent(SP).

f( D + 0.2*(E-B))  :-
    d(D),e(E), b(B).
```

#### 3.2.3 *Effort Equations*

```
%hmmm... sced value never used
pm(Pm0*Em17+Pa) :-
    pmNs(Pm0), w(sced,Em17), pmAuto(Pa).

pmNs(A*(S^E)*Em1 *Em2 *Em3 *Em4 *Em5 *Em6 *Em7*Em8*Em9*
            Em10*Em11*Em12*Em13*Em14*Em15*Em16) :-
    a(A), size(S), e(E), w(rely,Em1), w(data,Em2),
    w(cplx,Em3), w(ruse,Em4), w(docu,Em5), w(time,Em6),
    w(stor,Em7), w(pvol,Em8), w(acap,Em9), w(pcap,Em10),
    w(pcon,Em11),w(aexp,Em12),w(pexp,Em13),
    w(ltex,Em14),w(tool,Em15),w(site,Em16).

e(B + 0.01*(Sf1+Sf2+Sf3+Sf4+Sf5)) :-
    b(B),
    w(prec,Sf1), w(flex,Sf2),w(arch,Sf3),
    w(team,Sf4), w(pmat,Sf5).

pmAuto((Ak*(At/100))/Ap) :-
    adaptedKsloc(Ak), at(At), atKprod(Ap).
```

### 3.3 *Tunings*

#### 3.3.1 *Constants*

```
a(2.5)  :- cocomo(1983).
a(2.94) :- cocomo(2000).
a(2.94) :- cocomo(ga).

b(0.91) :- cocomo(2000).
b(1.01) :- cocomo(1983).
b(1.01) :- cocomo(ga).

c(3.0)  :- cocomo(1983).
c(3.67) :- cocomo(2000).
c(3.67) :- cocomo(ga).

d(0.28) :- cocomo(2000).
d(0.33) :- cocomo(1983).
d(0.33) :- cocomo(ga).
```

#### 3.3.2 *Post-architecture scale factors*

The COCOMO 2000 scale factors learnt via bayesian tuning.

```
postArch(2000,scaleFactors) =
           [ xl, vl,   l,   n,   h,  vh,  xh]+
    [[prec,  _,6.20,4.96,3.72,2.48,1.24,   _]
    ,[flex,  _,5.07,4.05,3.04,2.03,1.01,   _]
    ,[arch,  _,7.07,5.65,4.24,2.83,1.41,   _]
    ,[team,  _,5.48,4.38,3.29,2.19,1.01,   _]
    ,[pmat,  _,7.80,6.24,4.68,3.12,1.56,   _]
    ].
```

The original scale factors.

```
postArch(1983,scaleFactors) =
           [ xl,  vl,   l,   n,   h,  vh, xh]+
    [[prec,   _,4.05,3.24,2.43,1.62,0.81,  _]
    ,[flex,   _,6.07,4.86,3.64,2.43,1.21,  _]
    ,[arch,   _,4.22,3.38,2.53,1.69,0.84,  _]
    ,[team,   _,4.94,3.95,2.97,1.98,0.99,  _]
    ,[pmat,   _,4.54,3.64,2.73,1.82,0.91,  _]
    ].
```

Some scale factors learnt via some genetic algorithms.

```
postArch(ga,scaleFactors) =
            [ xl,  vl,   l,   n,   h,  vh,  xh]+
    [[prec,   _,4.05,3.24,2.43,1.62,0.81,   _]
    ,[flex,    _,6.07,4.86,3.64,2.43,1.21,   _]
    ,[arch,    _,4.22,3.38,2.53,1.69,0.84,   _]
    ,[team,    _,4.94,3.95,2.97,1.98,0.99,   _]
    ,[pmat,    _,4.54,3.64,2.73,1.82,0.91,   _]
    ].
```

#### 3.3.3 *Post-architecture effort multipliers:*

The COCOMO 2000 effort multipliers learnt via bayesian tuning.

```
postArch(2000,effortMultiplers) =
           [xl,   vl,   l,   n,  h,  vh,  xh]+
    [[rely,  _,0.82,0.92,1.00,1.10,1.26,   _]
    ,[data,  _,_   ,0.90,1.00,1.14,1.28,   _]
    ,[cplx,  _,0.73,0.87,1.00,1.17,1.34,1.74]
    ,[ruse,  _,_   ,0.95,1.00,1.07,1.15,1.24]
    ,[docu,  _,0.81,0.91,1.00,1.11,1.23,   _]
    ,[time,  _,   _,   _,1.00,1.11,1.29,1.63]
    ,[stor,  _,   _,   _,1.00,1.05,1.17,1.46]
    ,[pvol,  _,   _,0.87,1.00,1.15,1.30,   _]
    ,[acap,  _,1.42,1.19,1.00,0.85,0.71 ,  _]
    ,[pcap,  _,1.34,1.15,1.00,0.88,0.76,   _]
    ,[pcon,  _,1.29,1.12,1.00,0.90,0.81,   _]
    ,[aexp,  _,1.22,1.10,1.00,0.88,0.81,   _]
    ,[pexp,  _,1.19,1.09,1.00,0.91,0.85,   _]
    ,[ltex,  _,1.20,1.09,1.00,0.91,0.84,   _]
    ,[tool,  _,1.17,1.09,1.00,0.90,0.78,   _]
    ,[site,  _,1.22,1.09,1.00,0.93,0.86,0.80]
    ,[sced,  _,1.43,1.14,1.00,1.00,1.00,   _]
    ].
```

The original effort multipliers.

```
postArch(1983,effortMultiplers) =
              [ xl,  vl,   l,   n,   h,  vh,  xh]+
     [[rely,   _,0.75,0.88,1.00,1.15,1.40,   _]
     ,[data,   _,   _,0.94,1.00,1.08,1.16,   _]
     ,[cplx,   _,0.75,0.88,1.00,1.15,1.30,1.65]
     ,[ruse,   _,   _,0.89,1.00,1.16,1.34,1.56]
     ,[docu,   _,0.85,0.93,1.00,1.08,1.17,   _]
     ,[time,   _,   _,   _,1.00,1.11,1.30,1.66]
     ,[stor,   _,   _,   _,1.00,1.06,1.21,1.56]
     ,[pvol,   _,   _,0.87,1.00,1.15,1.30,   _]
     ,[acap,   _,1.50,1.22,1.00,0.83,0.67,   _]
     ,[pcap,   _,1.37,1.16,1.00,0.87,0.74,   _]
     ,[pcon,   _,1.26,1.11,1.00,0.91,0.83,   _]
     ,[aexp,   _,1.23,1.10,1.00,0.88,0.80,   _]
     ,[pexp,   _,1.26,1.12,1.00,0.88,0.80,   _]
     ,[ltex,   _,1.24,1.11,1.00,0.90,0.82,   _]
     ,[tool,   _,1.20,1.10,1.00,0.88,0.75,   _]
     ,[site,   _,1.24,1.10,1.00,0.92,0.85,0.79]
     ,[sced,   _,1.23,1.08,1.00,1.04,1.10,   _]
     ].
```

Some effort multipliers learnt via some genetic algorithms.

```
postArch(ga,effortMultiplers) =
               [ xl,  vl,   l,   n,   h,  vh,  xh]+
     [[rely,    _,0.79,0.78,1.00,1.16,1.41,   _]
     ,[data,    _,   _,0.96,1.00,1.31,1.20,   _]
     ,[cplx,    _,0.90,1.06,1.00,0.99,0.99,0.87]
     ,[ruse,    _,   _,0.89,1.00,1.16,1.34,1.56]
     ,[docu,    _,0.85,0.93,1.00,1.08,1.17,   _]
     ,[time,    _,   _,   _,1.00,1.01,1.24,2.13]
     ,[stor,    _,   _,   _,1.00,1.36,1.37,1.42]
     ,[pvol,    _,   _,1.25,1.00,1.13,1.15,   _]
     ,[acap,    _,1.19,1.26,1.00,1.00,0.73,   _]
     ,[pcap,    _,1.71,1.73,1.00,0.75,0.74,   _]
     ,[pcon,    _,1.26,1.11,1.00,0.91,0.83,   _]
     ,[aexp,    _,1.41,1.02,1.00,0.64,0.86,   _]
     ,[pexp,    _,1.26,1.12,1.00,0.88,0.80,   _]
     ,[ltex,    _,1.24,1.11,1.00,0.90,0.82,   _]
     ,[tool,    _,1.13,0.91,1.00,1.09,2.86,   _]
     ,[site,    _,1.24,1.10,1.00,0.92,0.85,0.79]
     ,[sced,    _,1.22,1.29,1.00,0.72,0.29,   _]
     ].
```

### *3.4 Data dictionary*

#### *3.4.1 General*

```
languageP(X) :- upf2sloc(X,_).

sym(X) :- rsym(X).

onezeroP(X) :- rin(0,1,0.2,X), number(X).

percentP(X) :- rin(0,100,1,X),integer(X).

posint(X)    :- rin(0,65536,X),integer(X).
posnum(X)    :- rin(0,inf,X),number(X).

num10(X) :- rin(0,10,X), number(X).

cocomoP(2000).
cocomoP(1983).
cocomoP(ga).

vlvh(n). vlvh(l). vlvh(h). vlvh(vl). vlvh(vh).

lvh(n). lvh(l). lvh(h). lvh(vh).

vlxh(n).  vlxh(l).  vlxh(h).
vlxh(vl). vlxh(vh). vlxh(xh).

lxh(n). lxh(l). lxh(h). lxh(vh). lxh(xh).

nxh(n). nxh(h). nxh(vh). nxh(xh).
```

#### *3.4.2 ”project”*

```
(cocomo(Coc); label(L); language(Lan)
;revl(R); newKsloc(K)
;adaptedKsloc(A) ;cm(C); dm(D); im(I) ;aa(Aa) ;unfm(U)
;su(Su) ;at(At) ;atKprod(Atp) ;scedPercent(Sc)
) :-
    project(Coc,L,Lan,R,K,A,C,D,I,Aa,U,Su,At,Atp,Sc),

    cocomoP(Coc),
    sym(L), languageP(Lan), percentP(R), percentP(K),
    posint(A), percentP(C), percentP(I), percentP(Aa),
    onezeroP(U), percentP(Su) ,percentP(At),
    posnum(Atp) ,posint(Sc),!.
```

#### *3.4.3 ”scores”*

```
(s(prec,Prec) ;s(flex,Flex) ;s(arch,Arch)
;s(team,Team) ;s(pmat,Pmat) ;s(rely,Rely)
;s(data,Data) ;s(cplx,Cplx) ;s(ruse,Ruse)
;s(docu,Docu) ;s(time,Time) ;s(stor,Stor)
;s(pvol,Pvol) ;s(acap,Acap) ;s(pcap,Pcap)
;s(pcon,Pcon) ;s(aexp,Aexp) ;s(pexp,Pexp)
;s(ltex,Ltex) ;s(tool,Tool) ;s(site,Site) ;s(sced,Sced)
):-
    scores(Prec,Flex,Arch,Team,Pmat,Rely,Data,Cplx,
            Ruse,Docu,Time,Stor,Pvol,Acap,Pcap,Pcon,
        Aexp,Pexp,Ltex,Tool,Site,Sced),

    vlvh(Prec), vlvh(Flex), vlvh(Arch), vlvh(Team),
    vlvh(Pmat), vlvh(Rely),  lvh(Data), vlxh(Cplx),
     lxh(Ruse), vlvh(Docu),  nxh(Time),  nxh(Stor),
     lvh(Pvol), vlvh(Acap), vlvh(Pcap), vlvh(Pcon),
    vlvh(Aexp), vlvh(Pexp), vlvh(Ltex), vlvh(Tool),
    vlxh(Site),!.
```

#### *3.4.4 scores2Weight*

```
w(A,W) :-
    demand(s(A,S)),
    postArch(A,S,W),
    demand(num10(W)).

postArch(A,S,W) :-
    cocomo(When),
    lookUp(postArch(When,_),A,S,W).
```

## 4 Shell start up actions

```
:- current_prolog_flag(max_integer,X),
   assert(inf(X)).

:- arithmetic_function(inf/0).

:- sneak(
     ['defaults.omo' % see §??
         ,'config.omo'   % see §??
         , ufp2sloc
     ]).

:- commandLine.
:- ?verbose -> hello ; true.
```

## 5 Knowledge base

### 5.1 Sample project

```
scores is [s(pmat,vl)
      ,s(pvol,l)
          ,s(ltex,l)
          ].

project is [cocomo(ga)
            ,label('eg#1')
            ,language(prolog)
            ,revl(10)
            ,newKsloc(100)
            ,adaptedKsloc(0)
            ,cm(0)      % new code
            ,dm(0)      % new code
            ,im(0)      % new code
            ,aa(2)      % basic module search + docu [4, p24]
            ,unfm(0.4)  % somewhat familiar
            ,su(30)     % nominal value [4, p23]
            ,at(0)
            ,atKprod(2.4)
            ,scedPercent(100)
            ].
```

## 6 Library (source code)

### 6.1 Meta

```
demand(X) :- X.
demand(X) :- \+ X, barph(failed(X)).

mybagof(X,Y,Z) :- bagof(X,Y,Z),!.
mybagof(_,_,[]).
```

### 6.2 Transforms

```
c2l((X,Y),[X|Z]) :- !,c2l(Y,Z).
c2l(X,[X]).

d2l((X;Y),[X|Z]) :- !,d2l(Y,Z).
d2l(X,[X]).
```

### 6.3 I/O

```
chars(F) :- see(F), get_byte(X), ignore(chars1(X)), seen.

chars1(-1) :- !.
chars1(X)  :- put(X), get_byte(Y), chars1(Y).

sneak(X) :- load_files(X,[silent(true),if(changed)]).

spit(N1,N2,X) :- (0 is N1 mod N2 -> spit(X) ; true).
spit(X) :- ?verbose,!,write(user,X),flush_output(user).
spit(_).

barph(X) :- format('%W> ~p\n',X),fail.
```

### 6.4 Maths

```
sum([H|T],X) :- sum(T,H,X).
sum([],X,X).
sum([H|T],Temp,X) :- Y is H + Temp, sum(T,Y,X).

average(N,G,Sum/L) :-
    bagof(N,G,All), sum(All,Sum), length(All,L).
```

### 6.5 Pretty print a list of terms.

```
portrays(L) :- portrays(L,_,_).

portrays([],_,_).
portrays([H|T],F0,A0) :-
    functor(H,F,A),
    (F0=F,A0=A
        -> portray_clause(H),
           portrays(T,F0,A0)
         ; nl,portray_clause(H),
           portrays(T,F,A)).
```

### 6.6 Random types

#### 6.6.1 Random strings

```
rsym(X)   :- nonvar(X),!.
rsym(X)   :- gensym(g,X).

rsym(_,X)   :- nonvar(X),!.
rsym(A,X)   :- gensym(A,X).
```

#### 6.6.2 Random number within a range

```
rin(M,N,_,X) :- nonvar(X),!, number(X),M =< X, X =< N.
rin(M,N,O,X) :- Steps is integer((N-M)/O),
            between(1,Steps,_),
        Y is random(Steps+1),
        X is min(M + Y*O,N).
```

#### 6.6.3 Random value of a list

```
rin(M,N,X) :- nonvar(X),!, number(X),M =< X, X =< N.
rin(M,N,X) :- Steps is integer(N-M),
          between(1,Steps,_),
          Y is random(Steps+1),
          X is min(M + Y,N).

rin(X,L) :- number(X),!, member(Y,L), X =:= Y.
rin(X,L) :- nonvar(X),!, member(X,L).
rin(X,L) :- length(L,N), rmember1(L,N,X).

rmember1([H],_,H)    :- !.
rmember1([H|T],N,X) :- Pos is random(N) + 1,
                       less1(Pos,[H|T],Y,L),
                       (X=Y
                       ; N1 is N - 1,
                         rmember1(L,N1,X)).

less1(1,[H|T],H,T) :- !.
less1(N0,[H|T0],X,[H|T]) :- N is N0 - 1, less1(N,T0,X,T).
```

### 6.7 Fast, named, assertions

Define some named fields.

```
multis(Stuff,All) :-
    bagof(One,Stuff^multi(Stuff,One),All).

multi((Heads :- Tail),(Head :- Tail)) :-
    d2l(Heads,List),
    member(Head,List).
```

Poke some values into the named fields.

```
fields(Fields,Functor,Term) :- fields1(Fields,Functor,Term),!.
fields(_,_,[]).

fields1([],_,_).
fields1([Field|Fields],Functor,Term) :-
    fields2(Field,Functor,Term),
    fields1(Fields,Functor,Term).

fields2(Field,Functor,Term) :-
    clause(Field,(Term,_)),
    functor(Term,Functor,_),!.
fields2(Field,Functor,_) :-
    barph(badField(Functor is [Field])).
```

### 6.8 Lookup Tables

Generate them

```
    lookUpTable(X,Out) :-
        bagof(Y,X^list2Relation1(X,Y),Out).
351
    list2Relation1(Table=Cols+Rows, lookUp1(Table,R,C,X)):-
        nth1(Pos,Cols,C),
354     member([R|Cells],Rows),
        nth1(Pos,Cells,X),
        nonvar(X).
```

Use them:

```
    lookUp(T,X,Y,Out) :-
        lookUp1(T,R,C,Out), gt(X,R), gt(Y,C), !.
359
    gt(Value,X to Y) :- !,X =< Value, Value =< Y.
    gt(Value,Value).
```

### 6.9 Configuration Control

```
    set(X=Y) :-
        retractall(option(X,_)),
364     assert(option(X,Y)).

    [] := []     :- !.
367 [H0|T0] := [H|T] :- !, H0 := H, T0 := T.
    X := Y       :-  option(X,Z),!, Y=Z.
    X := _       :- !, barph(missingOption(X)).
370 ?X           :- atomic(X), X := 1.

    commandLine :-
373     current_prolog_flag(argv, Argv),
        append(_, [--|Args], Argv), !,
        concat_atom(Args, ’ ’, SingleArg),
376     term_to_atom(Term, SingleArg),
        c2l(Term,List),
        forall(member(One,List), set(One)).
379 commandLine.
```

### 6.10 Demo support code

Catches the output from some predicate X and saves it a file X.spy. The command:

```
\SRC{X.spy}{Caption}
```

includes the generated file into the LaTeX document.

The code `demos/1` deletes any old output and runs some goal twice: once to trap it to a file and once to show the results on the screen.

```
    demos(G) :-
        sformat(Out,’~w.spy’,G),
382     (exists_file(Out) -> delete_file(Out) ; true),
        tell(Out),
        format(’% output from ’’:- demos(~w).’’\n\n’,G),
385         T1 is cputime,
        ignore(forall(G,true)),
        T2 is (cputime - T1),
388     format(’\n% runtime = ~w sec(s)\n’,[T2]),
            told,
        format(’% output from ’’:- demos(~w).’’\n’,G),
391         ignore(forall(G,true)),
        format(’\n% runtime = ~w sec(s)’,[T2]).
```

### 6.11 License

```
    hello :-  % ◀................................................. 393
        [program,version,copyright,motto,copywho]:=[N,V,Y,M,C],
395     format(’~s version ~s\n Copyright (C) ~s by ~s\n’,
               [N,V,Y,C]),
        format(’ "~s"\n\n~s ~s ’,[M,N,V]),
398     chars(’nowarranty.txt’). % see §A.1

    warranty :-
401     [program,copyright,copywho]:=[P, Y,C],
        format(’~s by ~s\n Copyright (C) ~s\n\n’,[P,C,Y]),
        chars(’warranty.txt’),nl. % see §A.2
404
    conditions :-
        chars(’conditions.txt’),nl. % see §A.3
```

*Acknowledgements* This research was conducted at West Virginia University; the University of British Columbia, Vancouver, Canada; and at the Jet Propulsion Laboratory, California Institute of Technology under NASA contract NCC2-0979. The work was sponsored by the NASA Office of Safety and Mission Assurance under the Software Assurance Research Program led by the NASA IV&V Facility. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.

## References

1. S. Chulani, B. Boehm, and B. Steece. Bayesian analysis of empirical software engineering cost models. *IEEE Transaction on Software Engineerining*, 25(4), July/August 1999.
2. C. Kemerer. An empirical validation of software cost estimation models. *Communications of the ACM*, 30(5):416–429, May 1987.
3. T. Mukhopadhyay, S. Vicinanza, and M. Prietula. Examining the feasibility of a case-based reasoning tool for software effort estimation. *MIS Quarterly*, pages 155–171, June 1992.
4. B. W.Boehm, E. Horowitz, R. Madachy, D. Reifer, B. K. Clark, B. Steece, A. W. Brown, S. Chulani, and C. Abts. *Software Cost Estimation with Cocomo II.* Prentice Hall, 2000.
5. J. Wielemaker. *SWI-Prolog*. Available from `http://swi.psy.uva.nl/projects/xpce/SWI-Prolog.html`.

## A License

This software is distributed under the GNU General Public License. Routines to display that license are shown at line 393.

### A.1 nowarranty.txt

OMO comes with ABSOLUTELY NO WARRANTY: for more details type ’warranty’.

This is free software, and you are welcome to redistribute it under certain conditions: for more details, type ’conditions’.

### A.2 warranty.txt

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; version 2 (see http://www.gnu.org/copyleft/gpl.html or type ’conditions’).

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, US.

## *A.3 conditions.txt*

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The 'Program', below, refers to any such program or work, and a 'work based on the Program' means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term 'modification'.) Each licensee is addressed as 'you'.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and 'any later version', you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM 'AS IS' WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF

# B Work-in-progress

I.e. not working yet

## *B.1 Early design effort multiplers*

```
effortMultipliers(early) =
              [  xl,  vl,   l,n,   h,  vh,  xh]+
    [[rcpx,    0.49,0.60,0.83,1,1.33,1.92,2.72]
    ,[ruse,       _,   _,0.95,1,1.07,1.15,1.24]
    ,[pdif,       _,   _,   1,1,   1,   _,   _]
    ,[pers,    2.12,1.62,1.26,1,0.83,0.63,0.50]
    ,[prex,    1.59,1.33,1.12,1,0.87,0.74,0.62]
    ,[fcil,    1.43,1.30,1.10,1,0.87,0.73,0.62]
    ,[sced,       _,1.43,1.14,1,   1,   1,   _]
    ].
```

## *B.2 Function point calculations*

### *B.2.1 Unadjusted function points to SLOC conversion ratios*

As loaded from `ufp2sloc.pl` (source: `http://www.theadvisors.com/langcomparison.htm`):

```
upf2sloc("aas macro", 91).
upf2sloc("abap/4",    16).
upf2sloc("accel",     19).
upf2sloc("access",    38).
upf2sloc("actor",     21).
upf2sloc("acumen",    28).
upf2sloc("ada 83",    71).
upf2sloc("ada 95",    49).
...
```

### *B.2.2 Function point complexity tables*

For internal logical files and external interface files:

```
ilfEif2Complexity =
    % record elements  data elements
    % ---------------  ----------------------------
                      [1 to 19, 20 to 50, 51 to inf]+
    [[0 to 1,              low,      low,       avg]
    ,[2 to 5,              low,      avg,      high]
    ,[6 to inf,            avg,      high,     high]
    ].
```

For external output and external inquiry:

```
eoEq2Complexity =
    % record elements  data elements
    % ---------------  --------------------------
                      [1 to 5, 6 to 19, 20 to inf]+
    [[0 to 1,             low,     low,       avg]
    ,[2 to 3,             low,     avg,      high]
    ,[4 to inf,           avg,    high,      high]
    ].
```

For external input:

```
ei2Complexity =
    % record elements  data elements
    % ---------------  --------------------------
                      [1 to 4, 5 to 15, 16 to inf]+
    [[0 to 1,              low,    low,       avg]
    ,[2 to 3,              low,    avg,      high]
    ,[3 to inf,            avg,   high,      high]
    ].
```

## *B.3 Key process areas*

COCOMO.2000 lets `pmat` be calcuated from answers to a questionnaire on pages 37-40 of [4].

### *B.3.1 Key process area answers*

From [4, p34-36]. First, we need some English words:

```
pmatc(P) :- empl(E), empl2pmat(E,P).

empl(E) :- kpas(Ks), E is round(5*Ks/100).

kpas(Av) :- average(K,kpa(K),Av).

kpa(K)  :- kpa(_,K).

empl2pmat(0,vl).
empl2pmat(1,l).
empl2pmat(2,n).
empl2pmat(3,h).
empl2pmat(4,vh).
empl2pmat(5,xh).
```

The answers to the questionnaire can be represented as follows:

```
almostAlways(100).
frequently(75).
aboutHalf(50).
occasionally(25).
rarelyIfEver(1).

kpa(requirementsManagement,X)        :- aboutHalf(X).
kpa(softwareProjectPlanning,X)       :- almostAlways(X).
kpa(softwareProjectTrackingAndOversight,X) :- occasionally(X).
kpa(softwareSubcontractManagement,X) :- aboutHalf(X).
kpa(sofwareQualityAssurance,X)       :- aboutHalf(X).
kpa(sofwareConfigurationManagement,X):- aboutHalf(X).
kpa(organizationProcessFocus,X)      :- occasionally(X).
kpa(organizationProcessDefinition,X) :- occasionally(X).
kpa(trainingPrograms,X)              :- aboutHalf(X).
kpa(integratedSoftwareManagement,X)  :- occasionally(X).
kpa(softwareProductEngineering,X)    :- occasionally(X).
kpa(intergroupCoordination,X)        :- occasionally(X).
kpa(peerReviews,X)                   :- rarelyIfEver(X).
kpa(quantitativeProcessManagement,X) :- rarelyIfEver(X).
kpa(defectPrevention,X)              :- rarelyIfEver(X).
kpa(technologyChangeManagement,X)    :- rarelyIfEver(X).
kpa(processChangeManagement,X)       :- rarelyIfEver(X).
```