I. Fedorov, I. Safonov
(International Unity Science Institute, Arlington, VA, USA)

PROCESS RELIABILITY OPTIMIZATION

A lot of articles and books were written about business process optimization, in particular, process reliability optimization. There are less adequate models and effective tools. Balance between sufficient trust and forced risk can be reached only when trust engineering and risk management will be armed by vendor-independent models and model-independent tools. Motor-car industry, aircraft and spacecraft industries, shipbuilding, electrical and electronic (!!!) engineering had never even dreamt of the striking situation established in the information technology – the monopoly of object-oriented methodology and preeminence of dropout creativity over regular knowledge and experience.

Is it possible that automation of information processing reached maturity before the automation of named above traditional industries? Tell me please, are you know analogs of the five capability maturity levels for software engineering or management in these industries? If it were not for unloved by all competitors and many customers Microsoft, operating systems were standardized following the same way as some programming languages and design methodologies. Let us go from one funny extreme to another: How long the statements like this “We have to constrain the unpredictable nature of technology. That’s why we’ve eliminated gotos, explicit pointer manipulation and potentially chaotic constructs” (Bran Selic cited by Alexandra Weber Morales in the article “Not Just Model – Model-Driven”, Software Development, September 2005) will surprise professionals? Why people like Bran Selic in minority? Wake up! XXI century is out of doors.

The analysis of world experience and our own expertise demonstrated principal possibility of practical decision of trust engineering and risk management problems for information and business technologies to keep to certain conditions. The first of them is necessity of process procedural description and metric specification allowing to account of and manipulate by all parameters, which are vitally important for customers. In our case, it was found that four canonical forms of algorithms and programs create all necessary and sufficient conditions for the model-driven engineering and management of any discrete or discrete-continuous processes. They are: linear, parallel, disjunctive and cyclic forms. The second condition is the ontological framework reflecting a goal, structure, behavior and resources of optimizing process.

For the first time, the methodology prototype was published by V. Karas

and I. Safonov (“Reliability Enhancement of MIS Functioning and Development: Optimization of Algorithms”, RDNTP, Kiev, 1974). The most complete description of this methodology was published in the article “Methodology of the Structured Algorithmic Systems’ Formal Design for Reliability” (Mechanization and Automation of Control, # 4, 1975) and book “Reliability Design for Management Algorithms”(IAPU, Vladivostok, 1982) one of the authors.

In this paper we intend to demonstrate how to design programs with optimal reliability for database processing in the Oracle environment. For simplicity and conciseness, we illustrate the methodology by optimization of a linear canonical form. Other canonical forms and their arbitrary canonical compositions are optimized in the similar way. The contemporary approach to Process Design for Reliability was published in the article “Aspect-Oriented Software Reliability Engineering” (Modeling and Analysis of Safety and Risk in Complex Systems. International Scientific School. – RAS, Saint-Petersburg, 2003).

```
-- Table A_LPA

CREATE TABLE a_lpa
(
 id      NUMBER(5) NOT NULL,
 p       NUMBER(3, 2) NOT NULL,
 t       NUMBER(10, 2) NOT NULL,
 x       NUMBER(5, 2),
 gb      NUMBER(5, 4),
 p0_p0 NUMBER(3,2) NOT NULL,
 a                  NUMBER(5,3)
)
/
 -- Constraints for A_LPA
 ALTER TABLE a_lpa
 ADD CONSTRAINT ala_pk PRIMARY KEY (id)
/
 ALTER TABLE a_lpa
 ADD CONSTRAINT ala_p0_fk FOREIGN KEY (p0_p0)
      REFERENCES P0(p0)
/
ALTER TABLE a_lpa
 ADD CHECK ("ID" IS NOT NULL)
/
 ALTER TABLE a_lpa
 ADD CHECK ("P" IS NOT NULL)
/
 ALTER TABLE a_lpa
 ADD CHECK ("T" IS NOT NULL)
/
 ALTER TABLE a_lpa
 ADD CHECK ("P0_P0" IS NOT NULL)
 /
 CREATE TABLE p0
 (
 p0      NUMBER(3,2) NOT NULL,
 m       NUMBER(5) DEFAULT 0 NOT NULL,
 px      NUMBER(5,4),
 t       NUMBER(6,3)
 )
 /
 -- Constraints for P0
 ALTER TABLE p0
 ADD CONSTRAINT p0_pk PRIMARY KEY (p0)
 /
 ALTER TABLE p0
 ADD CHECK ("P0" IS NOT NULL)
 /
 ALTER TABLE p0
 ADD CHECK ("M" IS NOT NULL)
 /
 PACKAGE BODY ALG_OPT IS
 PROCEDURE A_LPA_XM
 (P_M IN P0.M%TYPE
 )
 IS
 A_LPA_J_ROW A_LPA%ROWTYPE;
 V_PX0 NUMBER(5, 4);
  V_X A_LPA.X%TYPE;V_P0 P0.P0%TYPE;
  A_LPA_ROW A_LPA%ROWTYPE;
  CURSOR A_LPA_CUR
  IS
 select * from a_lpa;
 CURSOR A_LPA_J_CUR
  (P_ID IN NUMBER
  )
  IS
 select * from a_lpa where id<>p_id;
 begin
    select p0 into v_p0 from p0 where rownum<2;
    open a_lpa_cur;
    loop
       fetch a_lpa_cur into a_lpa_row;
       exit when a_lpa_cur%notfound;
       v_x:=floor(ln(1-v_p0)/ln(1-a_lpa_row.p));
       update a_lpa set x=v_x where id=a_lpa_row.id;
       commit;
    end loop;
    close a_lpa_cur;
 end;
 PROCEDURE A_LPA_MAIN
  IS
 V_PX P0.PX%TYPE;
 V_P0 P0.P0%TYPE;
 V_M P0.M%TYPE;
 begin
    update p0 set m=0;
    commit;
    select m,p0 into v_m,v_p0 from p0 where rownum<2;
    a_lpa_xm(v_m);
    alg_opt.a_lpa_pxm;
    select px into v_px from p0 where rownum<2;
    while v_p0 > v_px loop
       a_lpa_gb(v_m);
       alg_opt.a_lpa_pxm;
       select px into v_px from p0 where rownum<2;
    end loop;
    a_lpa_a;
    a_lpa_t;
 end;
 PROCEDURE A_LPA_PXM
  IS
 V_P0 P0.P0%TYPE;
 V_PX P0.PX%TYPE;
 V_M P0%ROWTYPE;
 A_LPA_ROW A_LPA%ROWTYPE;

 CURSOR A_LPA_CUR
  IS
 select * from a_lpa;
```

```
    begin
       v_px:=1;
       open a_lpa_cur;
       loop
          fetch a_lpa_cur into a_lpa_row;
          exit when a_lpa_cur%notfound;
          v_px:=v_px*(1-power((1-
a_lpa_row.p),(a_lpa_row.x+1)));
   end loop;
       close a_lpa_cur;
       update p0 set px=v_px where rownum<2;
       commit;
    end;
    PROCEDURE A_LPA_GB
     (P_M IN NUMBER
     )
     IS
    V_MAX_GB_ID A_LPA.ID%TYPE;
    A_LPA_J_ROW A_LPA%ROWTYPE;
    A_LPA_ROW A_LPA%ROWTYPE;
    V_GB NUMBER(5, 4);
    CURSOR A_LPA_CUR
     IS
    select * from a_lpa;
    CURSOR A_LPA_J_CUR
     (P_ID IN NUMBER
     )
     IS
    select * from a_lpa where id<>p_id;
    begin
       open a_lpa_cur;
          loop
             fetch a_lpa_cur into a_lpa_row;
             exit when a_lpa_cur%notfound;
             v_gb:=0;
             open a_lpa_j_cur(a_lpa_row.id);
             loop
                fetch a_lpa_j_cur into a_lpa_j_row;
                exit when a_lpa_j_cur%notfound;
                v_gb:=v_gb+(1-power((1-
a_lpa_j_row.p),(a_lpa_j_row.x+1)));
             end loop;
    close a_lpa_j_cur;
    v_gb:=v_gb/(a_lpa_row.t*(a_lpa_row.x+1));
             update a_lpa set gb=v_gb where
id=a_lpa_row.id;
             commit;
          end loop;
    close a_lpa_cur;
    select a1.id into v_max_gb_id from a_lpa a1
    where a1.gb = (select max(a2.gb) from a_lpa a2);
       update a_lpa set x=x+1 where id=v_max_gb_id;
       update p0 set m=m+1;
       commit;
    end;
    PROCEDURE A_LPA_A
    A_LPA_ROW A_LPA%ROWTYPE;
CURSOR A_LPA_CUR
IS
    select * from a_lpa;
    IS
    V_A A_LPA.A%TYPE;
    begin
       open a_lpa_cur;
       loop
          fetch a_lpa_cur into a_lpa_row;
          exit when a_lpa_cur%notfound;
          v_a:=(1-power((1-
a_lpa_row.p),(a_lpa_row.x+1)))/(1-power((1-
a_lpa_row.p),(a_lpa_row.x)));
          update a_lpa set a=v_a where
id=a_lpa_row.id;
          commit;
       end loop;
       close a_lpa_cur;
    end;
    PROCEDURE A_LPA_T
     IS
    K NUMBER(5, 0);
    V_TX A_LPA.T%TYPE;
    V_X A_LPA.X%TYPE;
    A_LPA_ROW A_LPA%ROWTYPE;
    V_T A_LPA.T%TYPE;
    CURSOR A_LPA_CUR
     IS
    select * from a_lpa;
    begin
       v_tx:=0;
       open a_lpa_cur;
          loop
             fetch a_lpa_cur into a_lpa_row;
             exit when a_lpa_cur%notfound;
             --select x into v_x from a_lpa where
id=a_lpa_row.id;
             k:=1;
             v_t:=0;
             for i in 1..a_lpa_row.x
             loop
                v_t:=v_t+(power((1-
a_lpa_row.p),(k)))*k;
                k:=k+1;
             end loop;
    v_tx:=v_tx+(a_lpa_row.t*(1+a_lpa_row.p*v_t));
          end loop;
       close a_lpa_cur;
       update p0 set t=v_tx;
       commit;
    end;
    PROCEDURE A_LPA_INIT
     IS
    begin
       update a_lpa set x=null,gb=null,a=null;
       commit;
       update p0 set px=null,t=null,m=0;
       commit;
    end;
END ALG_OPT;
```