Customization of Software Project Simulator for Improving Estimation Accuracy

Osamu Mizuno, Shinji Kusumoto and Tohru Kikuno
Department of Informatics and Mathematical Science,
Graduate School of Engineering Science, Osaka University, Japan
o-mizuno@ics.es.osaka-u.ac.jp

Abstract

We have developed a Petri-net based project simulator for predicting development effort and residual fault content. The simulator requires customization based on four parameters to reflect project characteristics. In this paper, we propose a new method for determining these parameters and validate the effectiveness of the method in two industrial experiments.

1. Project Simulator

The software project simulator consists of Project model and Process model. Project model specifies three key components: activities, products and developers. Process model defines a set of Activity models that are described by an extended GSPN(Generalized Stochastic Petri-Net).

1.1. Activity Model

Figure 1(a) shows an example of the description of the design activity. In the extended GSPN, a token has three attributes: product size s, the number of faults f and consumed workload[1] w, as shown in Fig. 1(b).

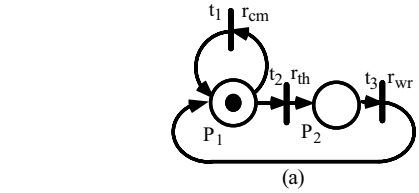
Transitions used here are timed transitions. The firing delay of each transition is exponentially distributed and the average firing delay of a transition is specified by a firing rate assigned to it. In Fig. 1(a), the firing rate r_{cm} of transition t_1 means that the average firing delay of transition t_1 is $1/r_{cm}$.

Each transition has a function (called an execution function) to be evaluated on its firing, as shown in Fig. 1(c). The execution of the function updates the attribute values of the token.

1.2. Customization using parameters

The firing rates r_{cm}, r_{th} and r_{wr} of the transitions are formulated by functions f_{cm}, f_{th} and f_{wr}, respectively[2]. Additionally, fault injections are interpreted as the stochastic events whose occurrences depend on the fault injection rate p_{in}. In general, p_{in} is formulated by function f_{in}.

[1]We define the term "workload" of an activity as the total time needed for a developer who has the standard capability to complete the activity[2].



(a)

attributes of token

s	product size
f	number of faults
w	consumed workload

(b)

transition	t_1	t_2	t_3
execution function	-	$w = w+1$	$s = s+1$ $f = f+1(p_{in})$

(c)

Figure 1. Activity model

In the previous study[2], we define concrete functions for all of these formulas. Since, we consider only design and coding activities in this paper, we present concrete functions for r_{cm}, r_{th}, r_{wr} and p_{in} as follows:

(1) Communicating rate: $r_{cm} = K_{cm} \times \frac{M^2}{L \times R}$

(2) Thinking rate: $r_{th} = K_{th} \times \frac{L}{M} \times M = K_{th} \times L$

(3) Writing rate: $r_{wr} = K_{wr} \times \frac{L}{M} \times M = K_{wr} \times L$

(4) Fault injection rate: $p_{in} = K_{in} \times \frac{M}{L \times R \times D} \times M$

Here, M is the number of the developers engaged in the activity, L is the sum of each developer's experience level, R is the completion rate of the input products, D is the number of the days from the current date to the deadline of the activity.

Before simulating a certain project, we must customize the simulator by tuning up the values of parameters so that each activity in the project can simulate actual situation. But, it is generally very hard to determine parameters, since these are tightly related each other. Therefore, in the previous studies, we used heuristic values shown in Table 1.

Table 1. Old values of parameters[2]

	K_{cm}	K_{th}	K_{wr}	K_{in}
$Design$	0.10	0.20	0.20	15.5
$Coding$	0.10	0.20	0.20	17.0

```
/* Phase 1 */
for i := 0.02 to 0.30 step 0.02 do begin
  for j := 0.05 to 0.30 step 0.05 do begin
    for k := 0.05 to 0.30 step 0.05 do begin
      #Effort := (the amount of effort obtained
            by simulation under (K_cm, K_th, K_wr) = (i, j, k));
      if (#Effort is within ±5% of Effort_actual) then
        Store (i, j, k) as a candidate;
    end;
  end;
end
Choose the best candidate (K_cm, K_th, K_wr)_best.
/* Phase 2 */
lo := 1.0 ; hi := 100
while lo ≤ hi do begin
  K_in := (lo + hi)/2 ;
  #Fault := (the number of faults obtained
        by simulation under K_in and (K_cm, K_th, K_wr)_best);
  if #Fault is within ±5% of Fault_actual) then break;
  if Fault_actual ≤#Fault then hi := K_in else lo := Kin ;
end
```

Figure 2. Algorithm for determining parameters

Table 2. New values of parameters

	K_{cm}	K_{th}	K_{wr}	K_{in}
$Design$	0.05	0.26	0.22	18.6
$Coding$	0.04	0.20	0.15	43.8

2. Proposed Algorithm

For customization of the simulator, we assume that an actual project data, for which the data on the amount of efforts and the number of faults are collected, is given.

Based on the lessons learned from the previous studies, we consider the following algorithm with two phases: Phase 1 is to determine K_{cm}, K_{th} and K_{wr}, and Phase 2 is to determine K_{in} with the values of K_{cm}, K_{th} and K_{wr} obtained at Phase 1.

The outline of the algorithm is shown in Fig. 2 In Fig. 2, variables *Effort*$_{actual}$ and *Fault*$_{actual}$ are the target values for the simulation and are obtained from the actual project data.

3. Experimental Evaluation

In order to confirm the effectiveness of proposed method, we have performed two experimental evaluations. In Experiments 1 and 2, we use two actual project data PR_1 and PR_2 offered from a certain company. Both projects are the development of the embedded software, and for almost the same application. The sizes of PR_1 and PR_2 are 12.3 Kstep and 9.6 Kstep, respectively.

In Experiment 1, we determine the value of firing parameters from the data of PR_1. Successively, in Experiment 2, we execute the simulation of PR_2 and evaluate the efforts and residual faults in PR_2.

3.1. Experiment 1

By applying the data of the project PR_1 to the algorithm shown in Fig. 2, we determine the values of parameters for the design and coding activities. The values of parameters obtained are shown in Table 2.

Comparing the values in Table 1 and Table 2, the values of K_{cm}, K_{th}, K_{wr} are almost the same. But the values of K_{in} are quite different each other.

3.2. Experiment 2

In order to evaluate the validity of the proposed algorithm, we estimate the number of faults in another project PR_2 by the project simulator using the values of parameters in Table 2.

Table 3. Simulated result of PR_2

	Effort (person-days)	Residual faults (# of faults)
Actual project data	41.7	17.0
New estimation	38.6	17.9
Estimation in [2]	37.1	13.2

Table 3 shows the simulated result of the project PR_2(For comparison, we also show the simulation result in [2]). From Table 3, the simulated effort is 38.6 (person-days) and the actual effort is 41.7 (person-days). The simulated value of residual faults is 17.9 and the actual value is 17. Thus, the new estimations of both effort and the residual faults for PR_2 are very close to the actual data. Furthermore, with respect to estimation accuracy, the new estimation by the proposed method is superior to the one in [2].

As a result, we can say that the values of parameters obtained from the project PR_1 are applicable to the estimation of the software quality in another project PR_2. Unfortunately, we don't have any more generalized data yet.

4. Future Work

In this study, we have discussed only the design and coding activities. Thus, determining the values of parameters in the test and debug activities is an important future research work. Additionally, we should contrast our approach for simulation and prediction with other approach[1].

References

[1] L. C. Briand, K. E. Emam, B. Freimut and O. Laitenberger: "Quantitative evaluation of capture-recapture models to control software inspections," Proc. 8th International Symposium on Software Reliability Engineering, pp.234–244, 1997.

[2] S. Kusumoto, O. Mizuno, Y. Hirayama, T. Kikuno, Y. Takagi and K. Sakamoto: "A new software project simulator based on generalized stochastic petri-net," Proc. 19th International Conference on Software Engineering, pp.293–302, 1997.