Predicting Runaway Projects for Reliable Software Developments Using Bayesian Classifier

Seiya ABE, Osamu MIZUNO, and Tohru KIKUNO
Graduate School of Information Science and Technology, Osaka University
{s-abe, o-mizuno, kikuno}@ist.osaka-u.ac.jp

1. Introduction

In order to realize reliable software development, problems in software development must be detected and avoided as soon as possible. Thus, detecting signs of problems at an early stage of the software project is important. Much research has been carried out on the detection of problem signs of software development projects [2, 3]. This study shows an easy-to-use approach to predict runaway projects in an organization to achieve reliable development process. Experimental results of predicting runaway projects are also shown.

2. Predicting Runaway Project by Bayesian Classifier

Figure 1 shows the outline of our approach to make a Bayesian classifier to predict runaway projects. First, in Step 1, we designed a questionnaire to be distributed to project members to collect the assessment data. The questionnaire is constructed based on our previous works and various software risk research [4]. Next, in Step 2, the SEPG distributed the questionnaire to project members, and asked them to fill out the questionnaire. After they finished filling out the questionnaire, the SEPG collected responses to the questionnaire. At the same time, in Step 3, the SEPG determined the success variable and the value of it for each development project from available quantitative project data. Finally, in Step 4, we apply the naive Bayesian learning to the responses to the questionnaire, which are prepared for the model learning. In case of prediction, we apply the learned model to responses of new projects and the value of success variable is determined by obtained probability.

The naive Bayesian classification is an optimal method of supervised learning if the values of the attributes of an example are independent given the class of the example. Although this assumption is almost always violated in practice, recent research has shown that naive Bayesian learning is also effective in practice [1].

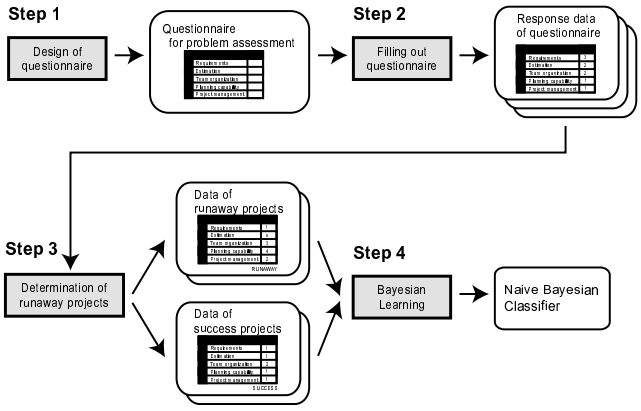

Figure 1. Bayesian classification

3. Design of the Questionnaire

The questionnaire consists of 30 questions to ask about possible signs of problems in software development. Each question must be filled in the specified ranks. For constructing this questionnaire, we first quoted the questionnaire used in our previous study [4]. We then discussed with the SEPG members in the company to be applied this questionnaire and determine company-specific questions. (See Table 1.)

Table 1. Overview of questionnaire

No.	Viewpoint	(# of questions)
1	Requirements	(4)
2	Estimations	(3)
3	Project organizations	(4)
4	Project planning	(5)
5	Progress management	(2)
6	Development technology	(3)
7	Development members	(4)
8	Environment	(2)
9	Other external factors	(3)

4. Evaluation with Empirical Data

4.1. Target Projects and Success Variable

The questionnaire was delivered to total 116 project members in 19 development projects in a certain company on July, 2003. The responses were collected one month

Table 2. Responses of questionnaire (a part)

Response ID	Developer ID	Project ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	...	Q30
R1	D1	P1	2	2	3	2	1			2	1	1	...	1
R2	D2	P2	1	1	1	1	1	1	2	1	1	1	...	2
R3	D3	P2	1	1	2	2	1	2	2	1	2	1	...	1
R4	D4	P2	3	2	3	4	2		2	1	2	1	...	
R5	D5	P2	2	1	1					1	2		...	
R6	D6	P2	2	1	3	2	2	3	2	1	2		...	3
R7	D7	P2	3	2	3	4	3	3	3	1	1	5	...	1
R8	D3	P3	1	1	2	2	2	2	2	2	2	1	...	2
R9	D5	P3	1	1	1					1			...	
R10	D6	P3	3	2	1	2	2	3	2	2	1	3	...	2
R11	D7	P3	3	1	1	1	3	3	4	2	1	5	...	1
R12	D8	P4	1	1	1	1	1	1	2	2	2	1	...	1
R13	D3	P4	1	1	2	2	2	2	2	2	2	1	...	2
R14	D4	P4	2	2	2	2	2	2	2	1	2		...	
R15	D5	P4	1	1	1					1	1		...	
R16	D6	P4	1	1	2	1	1	1	1	2	2	1	...	1
R17	D7	P4	2	1	1	2	3	3	2	1	1	5	...	1
R18	D8	P5	4	4	4	4	4	4	3	4	4	5	...	4
:	:	:	:	:	:	:	:	:	:	:	:	:		:
R80	D7	P15	3	2	2	2	3	3	3	2	2	5	....	1

later. These 19 projects were development of software products related to the software process improvement activities in this company. Among them, 15 projects had already finished their development in 2001 and 2002, and the rest 4 projects were on-going projects in 2003.

In this experiment, since SEPG has to determine the value of success variable, the projects must be finished. For this reason, we used 80 responses in 15 projects in 2001 and 2002. A part of collected 80 responses is shown in Table 2. We can see that some of responses have blank answers. Since Bayesian classification allows incomplete data such that some of questions are left no answer, we can use the collected data as they are.

In order to evaluate the final status of software development projects, we define a subjective metric *STATUS* which represents whether a project is runaway or success. To determine the value of *STATUS*, several aspects of projects such as cost, quality, and duration are considered. It is evaluated by SEPG members without referring the responses of the questionnaire but considering quantitative data obtained from these projects. The result of evaluation for 15 projects are shown in "Actual Result" column of Table 3.

4.2. Evaluation

In this experiment, we tried to show prediction of *STATUS* for a project from the project member's response by applying naive Bayesian classifier. In order to show the effectiveness of the proposed approach, we performed the jackknife method on the collected data. Simply speaking, we first excluded responses related to a certain project for testing. Next, we consider the other responses as learning data. Finally, we applied the excluded responses to the Bayesian classifier and obtained predicted *STATUS* for applied responses. This procedure was iterated until all projects were predicted.

Project members in a project have their own opinion of the problem signs in their project. There could exist different responses to the questionnaire for a project. Thus, predicted results from responses for a project could vary. We therefore determine whether a project is runaway or not with a majority voting of predicted results.

Table 3 shows predicted *STATUS* for 80 responses, and Table 3 also shows classified results of prediction for each project. Each row shows a project ID, its actual *STATUS* of development (Success or Runaway), and result of prediction by the Bayesian classifier using responses from project members. Grayed rows indicate that the prediction of a project is the same as the actual result by majority voting. For example, project P15 has 6 members who responded the questionnaire. The result of prediction based on the responses of 4 members out of 6 shows that P15 is to be success. By majority voting, P15 can be considered to be success, and the predicted result is correct since actual result of P15 was success. We can see that 12 out of 15 projects were correctly predicted.

At this point, we cannot state whether or not the accuracy of prediction is good enough. To do so, we need to perform more experiments. This is our important future research.

Table 3. Comparison of actual *STATUS* and predicted result

Project	Actual Result	Result of Prediction	
		Success	Runaway
P2	Success	3	3
P3		**4**	0
P4		**6**	0
P7		0	**1**
P14		**5**	1
P15		**4**	2
P1	Runaway	0	**1**
P5		2	**4**
P6		**6**	0
P8		3	**4**
P9		0	**9**
P10		0	**3**
P11		0	**6**
P12		0	**5**
P13		2	**6**

Acknowledgment Authors would like to thank Mr. Kenji Matsuse and Mr. Hidenobu Sakata of RICOH Company, Ltd. who gave us empirical data of their software development and invaluable advise to this research.

References

[1] P. Domingos and M. J. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3):103–130, 1997.

[2] J. Jiang and G. Klein. Software development risks to project effectiveness. *Journal of Systems and Software*, 52:3–10, 2000.

[3] J. D. Procaccino, J. M. Verner, S. P. Overmyer, and M. E. Darter. Case study: factors for early prediction of software development success. *Information and Software Technology*, 44:53–62, 2002.

[4] Y. Takagi, O. Mizuno, and T. Kikuno. An empirical approach to characterizing risky software projects based on logistic regression analysis. *Empirical Software Engineering*, (to appear).