A STATISTICAL NEURAL NETWORK FRAMEWORK FOR RISK MANAGEMENT PROCESS

From the Proposal to its Preliminary Validation for Efficiency

Salvatore Alessandro Sarcià, Giovanni Cantone
Dip. di Informatica, Sistemi e Produzione, Università di Roma Tor Vergata, via del Politecnico 1, 00133 Roma, Italy
sarcia@<disp.uniroma2.it | cs.umd.edu>, cantone@uniroma2.it

Victor R. Basili
Dept. of Computer Science, University of Maryland, A.V. Williams Bldg. 115, College Park 20742, MD, USA
basili@cs.umd.edu

Keywords: Risk Management Process, Artificial Neural Networks, Experimental Software Engineering, Prior Probability, Posterior Probability, Bayes' Theorem, Computational Software Engineering.

Abstract: This paper enhances the currently available formal risk management models and related frameworks by providing an independent mechanism for checking out their results. It provides a way to compare the historical data on the risks identified by similar projects to the risk found by each framework Based on direct queries to stakeholders, existing approaches provide a mechanism for estimating the probability of achieving software project objectives before the project starts (Prior probability). However, they do not estimate the probability that objectives have actually been achieved, when risk events have occurred during project development. This involves calculating the posterior probability that a project missed its objectives, or, on the contrary, the probability that the project has succeeded. This paper provides existing frameworks with a way to calculate both prior and posterior probability. The overall risk evaluation, calculated by those two probabilities, could be compared to the evaluations that each framework has found within its own process. Therefore, the comparison is performed between what those frameworks assumed and what the historical data suggested both before and during the project. This is a control mechanism because, if those comparisons do not agree, further investigations could be carried out. A case study is presented that provides an efficient way to deal with those issues by using Artificial Neural Networks (ANN) as a statistical tool (e.g., regression and probability estimator). That is, we show that ANN can automatically derive from historical data both prior and posterior probability estimates. This paper shows the verification by simulation of the proposed approach.

1 INTRODUCTION

Since the late 1980's, when Risk Management (RM) was first applied to the software domain (Boehm, 1989), (Charette, 1989) (Boehm, 1991), there have been limited real advances in formal RM methodologies (Roy, 2004).

One weakness of the existing RM practices (Sarcià, Cantone, 2006) is the inability to check the identified risks against past performance on similar projects. This paper tries to improve on the current RM practices by integrating into the frameworks an independent mechanism for comparing the current risks to historical data. In particular, we propose an Artificial Neural Network-based (ANN) approach for estimating both prior and posterior probability using the historical data of the software organization. Briefly, the main questions that this paper answers are: "*What is the probability that the project succeeds? How can we evaluate whether risks actually have impacted on the achievement of the project objectives?*" The remainder of this paper deals with the following topics: Section 2 provides an overview of the key elements of an RM process focusing on the reliable prediction of project success and failure. Section 3 defines the problem and discusses about RM strategies and methodologies. Section 4 presents and discusses results from a

simulation-based case study for efficiency validation. Section 5 shows some insights for improving the proposed approach. Some final remarks and forewords for future work conclude the paper.

2 RISK MANAGEMENT

Risk is a measure of probability and severity of unwanted effects (Boehm, 1991) on software development projects. Basically, all current risk estimation frameworks in software development process (Boehm, 1991), (Charette, 1989), (Higuera, 1996), (Kontio, 1996), (Karolak, 1997), (Madachy, 1997), (SEI, 2002), (Roy, 2004), and (Albert, 2006) evaluate the risk value (or risk exposure), *V* as:

$$V(UE) = I(UE) * P(UE) \qquad (1)$$

where UE stands for an unwanted event (risk factor); P(UE) is the probability that UE occurs; I(UE) stands for the impact (or cost) due to the occurrence of UE. As an example, we can consider the following situation:
UE = {'Resignation of a senior analyst'}, I(UE) = {6 month's delay} and P(UE) = 0.25 then:

$$V(UE) = 6 * 0.25 = 1.5 \text{ (months)} \qquad (1.a)$$

2.1 Risk Management Process

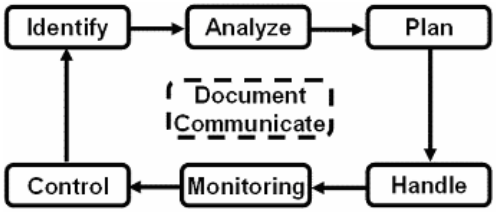
RM is a continuous process, which aims at applying suitable tools, procedures, and methodologies to avoid that risk occurs or keep it within stated limits (SEI, 2002). Some studies describe several basic steps of RM in slightly different ways, but substantially they report a process, which – similarly to the one that Figure 1 depicts – is based on the stages below.

Identify: Includes the identification of the internal and external sources of risk through a suitable taxonomy (Higuera, 1996) (SEI, 2002), as depicted in Table 1. Risk Identification involves stakeholders and depends on the project context.

Analyze: Aims at understanding when, where, and why risk might occur, through direct queries to stakeholders about the probability and impact of risk elements. The prior probability is evaluated. This is the probability that an event (e.g., project delay) could happen before the project starts. This is calculated through prior information.

Plan: In order to establish a strategy to avoid or mitigate the risk, decisions have to be made. In this stage, contingency plans are stated as well as the related triggering thresholds. Risk-controlling actions are defined and selected.

Figure1: Risk Management Process.



Handle: The planned actions are carried out if the risk occurs.

Monitoring: This is a continuous activity that watches the status of the project, and checks performance indicators (e.g., quality indicators, for instance the number of defects per size unit). In this stage, data concerning the risk trend is gathered.

Table 1: Risk sources taxonomy (based on SEI, 2002)

Project Risk		
Software Development	**Development Environment**	**Program Constraints**
•Requirements •Design •Code and Unit Test •Integration and Test •Engineering Specialties	•Development process •Development System •Management Process •Management Methods •Work Environment	•Resources •Contract •Program Interfaces

Control: It appraises the corrections to be taken on risk mitigation plan in case of deviations. If the indicators show an increase over the fixed threshold, a contingency plan is triggered. In our opinion, this stage should deal with the posterior probability because events have already occurred; that is, one tries to figure out whether an unwanted event actually had an impact on project objectives. For further information about prior and posterior probability (Bayes' Theorem), please see Section 1.8 in (Bishop, 1995).

Document and Communicate: This stage can be truly considered as the core of RM; in fact, all other stages refer to Documentation and Communication for enabling information exchange.

2.2 Project Success and Failure

Project success and failure constitute the environment where risk considerations take place. According to the Standish Group (Standish Group, 1994), budget, schedule, and technical quality are

the primary concepts to evaluate as to whether or not a project meets its objectives.

It was argued (Jones, 2002) that the measure of success varies depending on how we define it. It has been suggested that success should be based on the developer's point of view (Procaccino, Verner, Lorenzet, 2006), and on the stakeholder's perspective (Linberg, 1999). Recently, empirical studies (Berntsson-Svensson, Aurum, 2006) have found that the nature of projects might not be systematically correlated with success and failure factors. (Boehm, 1991) includes in his top ten risk items unrealistic schedules and budgets, but, it was pointed out that having a schedule is not necessarily correlated with project success (Verner, Cerpa, 2005): there are successful projects developed in a shorter time than scheduled. Additionally, it has been suggested that project success factors are intertwined with product success factors (Wallin *et al.*, 2002).

This paper adopts a definition of project success and failure very close to the Standish-Group's definition (Standish Group, 1994), which states that a *project achieves its objectives when it is on time, on budget, with all features and functions as originally specified*. Concerning this point, let us recall that the triad cost-schedule-performance of project management is often termed the "*Iron Triangle*" (Williams, Pandelios, Behrens, 1999). This triadic view will be used in the rest of the paper.

Our understanding of risk uses a threshold concept (Linberg, 1999). In order to check whether or not a project is meeting the stated objectives, we consider thresholds on budget, schedule, and required performances. In this way, project success and failure vary according to how we set these thresholds. This interpretation allows us to define a project to be successful projects just on the basis of a specific aspect (e.g. a project can be considered successful for budget factor but fail for the required defect rate). Actually, we are going to show, below, that this concept can be represented by a function of many variables, which, somehow, impact on the metric that we picked out to represent the success.

3 PROBLEM DEFINITION

To define the problem that this paper deals with, let us consider again equations (1), (1.a) and Figure 1. In particular, through the Analysis phase (Figure 1), we can calculate the risk exposure for each risk element, which we found in the previous phase (Identify). As we have already mentioned (Section 2.1), this kind of evaluation is based on asking direct queries to stakeholders, which should provide a subjective score for each identified risk. This procedure might not be reliable because stakeholders' subjective scores could change over time based upon their feeling (Tversky, Kahneman, 1974). In other words, we need a control mechanism. A possible improvement could be using historical data, if available. The aim would be building a control mechanism to increase our confidence on those subjective scores.

When evaluating the overall risk for achieving a project objective (Iron Triangle), we pick all the risk exposures and estimates for each project objective (that is, each element of the Iron Triangle), calculating the average risk exposure (Roy, 2004). Generally, this average is weighted according to information that RM managers can get by historical data of the organization (Roy, 2004), (Fussel and Field, 2005). Based on these weighted averages, one should state strategies and plans to avoid or shrink the risk occurrence. Actually, over the Control phase (Figure 1), we are faced with the problem to figure out whether and to what extent risks have impacted on project objectives (e.g., scheduling slippage evaluation). For instance, if risks impacted very much upon the project, we should get low probability to be successful. Note that, this evaluation depends on what we pick out as a basis for comparison. For instance, let us assume that the historical data from the considered organization shows that, usually in similar projects, the organization got values for the Schedule Performance Index (SPI) between 0.8 and 0.9– (SPI is an index that provides information about the objective to stay on time). Based on the definition of this index (ratio between what we have done and what we planned to do), only values greater than 1.0 should be considered as success. Let us assume that, current project got $SPI_{Curr} = 0.95$. If we picked out the theoretical value ($SPI_{theor} = 1.0$) as a threshold for comparison, we would get a failure (because $SPI_{Curr} < SPI_{theor}$). On the contrary, if we picked out, for instance, the mean value on past projects (e.g., $SPI_{mean} = 0.85$), we would get a success. Therefore, if we calibrate the evaluation criterion on the observed data then the decisions that we make are based on information that is as large and updated as we can. This happens because the object for comparison is based upon real performances of the organization. Actually, the problem is more complex. As a matter of fact, this calibration should take in account all possible factors (e.g., domain,

complexity, developers' experience etc…) that impacts on the considered performance (e.g., SPI). In other words, we should consider a regression function (e.g., SPI as a dependent variable and impacting factors as independent variables). Often, for the sake of simplicity, those factors are left out; hence, organizations prefer adopting just theoretical values. This research proposes a way to tackle with these issues.

3.1 The Strategy

Figure 2 depicts our point of view. Each risk element might impact (or might not) on one or more project objectives. We can get the impact of the assumed risk elements by checking suitable metrics (Basili, Caldiera, Cantone, 1992), (Basili, Caldiera, Rombach, 1994), (Basili, Caldiera, Rombach, 1994b), (Cantone, Donzelli, 2000), and (Piattini, Genero, Jiménz, 2001) on the *Iron Triangle* during the project, as depicted on the right side of Figure 2 (*lollipop* notation). Concerning the posterior probability analysis, based on the proposed approach (ANN), we propose to calculate the probability that the current project has succeeded; if this probability is less than 0.5 then risks occurred and they cannot be left out. Moreover, the more this probability approaches 1.0, the more the impact of risk factors decreases. Therefore, if we get no effects on the *Iron Triangle*, we can decide that the leverage of those considered risks could be left out. Finally, if the historical data evaluated by ANN and the weighted scores obtained by existing frameworks agree, then we can get more confident on our results. If, on the contrary, they do not (e.g., if the theoretical performance indicators and the posterior probability do not point out success or failure at the same time), then we have to carry out further investigations on risk elements.

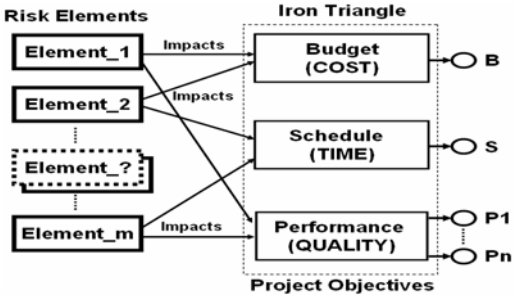


Figure 2: Risk factors that impact on the Iron triangle

Our strategy proposes to use historical data to build a baseline to evaluate risk value and to automatically check whether the project is meeting the stated objectives or not. This baseline is actually generated by an ANN, whose output represents the posterior probability that can be used in the risk evaluation, as mentioned above.

3.2 The Methodology

In this section we describe the procedure that one should follow in order to get risk probabilities from historical data of the organization as claimed above.

(1) Identifying Success Metrics: In Figure 3 we can see, by a lollipop notation, that for each project objective (O) a threshold function (T_O) can be stated, which provides success and failure measures. Note that, T_O, in this framework, is a function of many factors as mentioned in Section 3 and as explained below. We could also define metrics on software quality aspects such as Defect Rate, Effort per Defect, Number of Changes per Release (T_{P1}, T_{P2} … T_{Pn}). Therefore, with regard to Figure 3, T_B could be a threshold function on Cost Performance Index (CPI) and T_S a threshold function on Schedule Performance Index (SPI) or Productivity (ratio between SLOC and Effort).

(2) Selecting Impacting Factors on the Success Metric: By impacting factor, we mean a variable that can affect the measurement of success metric (e.g., Domain, KSLOC, Complexity, and Developers' Expertise). It is possible that we should take in account a big number of factors to get good results. But, the more this number grows, the more regression function is difficult to be estimated, if we use canonical methods. However, if we use ANN and Backpropagation for calculating regression functions, the number of factors does not influence the procedure.

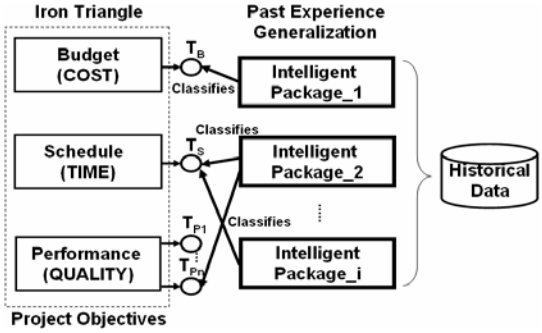


Figure 3: Using Computational Intelligence technique to evaluate project successes and failures

Of course, the effects of this grow impact on the computation time, which could be very big. Hence, we are considering ANN as a mean to estimate non-linear regression functions because we can obtain

reliable estimates as automatically as possible, even if we have to consider a high number of impacting factors. Refer to (Bishop, 1995) and (Dreyfus, 2005) for more information about the use of ANN and Backpropagation as a mean to estimate non-linear regression functions.

In order to understand which factors impact the considered project objective, we can use GQM approach (Basili, Caldiera, Rombach, 1994). Briefly, some possible questions are the following:

- *What can impact on the achievement of the project objective?* (E.g., effort ÷ team size, team experience, used development tools).
- *What is the software context?* (E.g., Domain, Category)
- *What is the development context?* (E.g. knowledge of the software system, organization maturity, development flexibility, etc.)

(3) Evaluating non-linear regression function: Now, we can build an ANN, for regression, as depicted in Figure 4, where input variables (independent variables) are the impacting factors said above, and the output (dependent variable) is the regression function estimate (on the success metric).

Using this regression function, we can divide our past projects between the successful and failed ones. In particular, if a project got a threshold value better than the regression function it is considered as a success, if this value was worse than regression function, it is considered as a failure. Note that, all the projects can belong to just one of those classes.

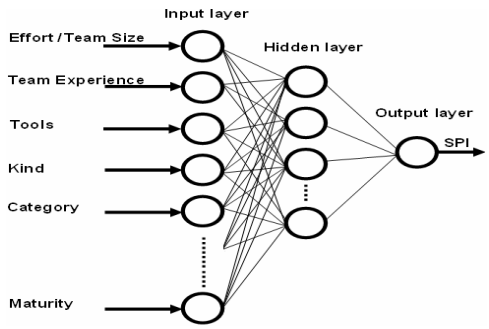


Figure 4: A Multi-layered Fed-forward Neural Network (Perceptron) for estimating regression functions.

Figure 5 depicts a trivial case in two dimensions. We show such an example because it can be represented on a plane. In a real case (e.g., Figure 4), we could have many factors and just one success metric. It is important to note that, what we are actually calculating is an iperplane that splits the considered iperspace into two disjoined subsets. Class A is the success class and class B is the failed one.

(4) Rejecting non-relevant factors: It is possible prove (Dreyfus, 2005) that an ANN that has a minimal configuration in terms of input variables and internal parameters (hidden units) provides the best regression function estimate. This characteristic is called parsimony. Therefore, over this stage, we should reject factors that do not impact on the success metric as well as shrink the ANN internal configuration (hidden units) as much as possible. In order to carry out this task, we can apply many different techniques, such as Principal Component Analysis (Jollife, 1986), Wrapper for Feature Subset Selection (John, Kohavi, Pfleger, 1994), Cross-validation or Leave-one-out (Dreyfus, 2005).

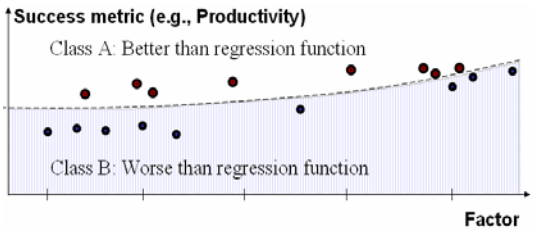


Figure 5: Two-dimension regression function that splits the plane into two disjoined subsets. The dashed line is a threshold function.

(5) Building a Two-Class Classification Network*:* So far we have turned our evaluation problem into a two-class discrimination problem. Now, we build a new network, for discrimination, as depicted in Figure 6. Such a system is able to classify all possible inputs as belonging to A or B. Actually, we can obtain much more from this kind of network. In fact, this is able to provide the posterior probability said above (Bishop, 1995). This is a very important result because, generally, Bayes' theorem has just theoretical importance, but it cannot be practically applied. The problem is that we cannot know all terms of Equation (2) right side.

In order to explain this problem let us consider Equation (2). Let Pr(A | x) be the posterior probability that Bayes' theorem provides, that is, the probability that the considered project belongs to A after its measurements have been executed.

$$\Pr(A \mid x) = \frac{p_X(x \mid A)\Pr(A)}{p_X(x \mid A)\Pr(A) + p_X(x \mid B)\Pr(B)} \quad (2)$$

Pr(A) and Pr(B) represent prior probability of class A and B respectively, $p_X(x \mid A)$ and $p_X(x \mid B)$ represent the probability density that x occurs conditioned to class A or B, that is, the probability that x occurs with respect to A or B.

The real problem is that we could, somehow, get estimates for P(A) and P(B), but we are not able to estimate $p_X(x \mid A)$ and $p_X(x \mid B)$, that is, Bayes' Theorem is not useful for real classifications. Based on the mathematical proof reported in (Bishop, 1995), if we use an ANN like the one that Figure 6 depicts, we can directly obtain Pr (A | x) even if we cannot know the conditioned probabilities said above.

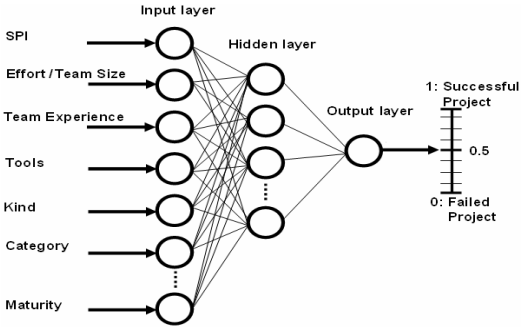


Figure 6: An Artificial Neural Network for two-class discrimination problems.

3.3 Prior Probability Estimate

So far we have shown that if we use a particular ANN for two-class discrimination, we can directly calculate the posterior probability. Now we show how to estimate the prior probability, that is, the probability that a project can achieve a stated objective before the project starts.

Let us observe that an individual input for the network in Figure 6 is a vector $\mathbf{V}_i$, like Expression (3) shows, where, $v_{i,1}$, $v_{i,2}$, …. , $v_{i,n}$ are the $\mathbf{V}_i$'s components and stand for the actual values that the network receives as an input.

$$\mathbf{V}_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,n}) \qquad (3)$$

$\mathbf{V}_i$ is a generic vector that represents the i^{th} project, and the second number (1 through n) represents the index of the input variables. With regards to Figure 6, for instance, the values in Expression (4) might be gathered.

In order to obtain all the possible $\mathbf{V}_i$ instances that the network can receive in input (henceforth we are denoting those vectors by the term *Input Set*, writing **V**), we consider the Cartesian in Expression (5), where $\{v_i\}$, is the set of all possible values that the i^{th} component of **V** can get. As experimentalists, we often call $\{v_i\}$ with Factor and the values that it can get ($v_{i,1}$, $v_{i,2}$, …. , $v_{i,n}$) with Levels or Treatments.

$$\begin{aligned} &v_{i,1} = \{\text{Val(SPI)}\} = \{0.9\}, \\ &v_{i,2} = \{\text{Val(Effort/TeamSize)}\} = \{1000/20\} \\ &\qquad = \{50\} \qquad (4) \\ &\ldots \end{aligned}$$

$$V = \{v_1\} \times \{v_2\} \times \ldots \times \{v_i\} \times \ldots \times \{v_n\} \qquad (5)$$

Note that, in a software engineering environment the cardinality of this Input Set is a finite number, i.e. the size of $\{v_i\}$ is a finite number. For instance, even if SPI might get real (infinitive) values in [0, $+\infty$[, actually, significant values for an organization could lie, for instance, inside [0.4, 1.5]. Moreover, significant approximations for those elements of the set [0.4, 1.5] are bounded to the first or second fractional, that is, finer grain values are not significant for the organization. For instance, we could make decision to have, in [0.4, 1.5[, just 11 possible values (0.4, 0.5 … 1.4), i.e. 1 factor and 11 treatments. A question arises naturally:

Expression (6) shows how to calculate the cardinality of **V**.

$$N = \text{Card}\{v_1\} * \text{Card}\{v_2\} * \ldots * \text{Card}\{v_n\} \qquad (6)$$

Note that, based on the definition of *Input Set*, **V** includes all the significant values for the organization. As experimentalists we generally call this set with Population. Now, in order to calculate the prior probability, we can feed the ANN with the whole *Input Set* **V** and obtain the cardinalities of both classes, A and B, Figure 7.

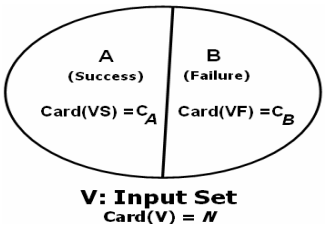


Figure 7: Prior probability estimation by a two-class classification network.

This happens because such a network is able to classify all possible significant inputs as belonging to A or B. In particular, the input vector belongs to A if and only if the ANN output lies inside [0.5, 1.0]. If this output lies inside [0, 0.5[, then the considered input belong to B (Figure 7).

Let us denote by **A** the subset of **V** composed of all the success vectors and by **B** the subset composed of all failure vectors, with $\{A\} \cap \{B\} = \Phi$ and $\{A\} \cup \{B\} = \{V\}$ (Figure 7).

Said C_A and C_B the cardinalities of **A** and B, respectively, we can calculate the prior probabilities to be successful (to achieve a project objective) by Expressions (7). Overall, P(A) represents the probability that the successful class can occur. Hence, P(B) = 1 - P(A) is the probability to fail. Based on historical data, therefore, P(A) value represents the success probability that the organization succeeds in developing a project before the project starts.

$$P(\mathbf{A}) = Card(\mathbf{A})/Card(\mathbf{V}) = C_A/N$$
$$P(\mathbf{B}) = CARD(\mathbf{B})/CARD(\mathbf{V}) = C_B/N = 1- P(\mathbf{A}) \quad (7)$$

Note that, what we have been presented in this Section could be viewed as a methodology for estimating prior probability like Bootstrap and Jacknife (Efron, Tibshirani, 1993) estimate confidence intervals for population parameters.

4 CASE STUDY

Let us give, now, some demonstrations concerning the efficiency of the proposed approach by showing a typical application. As usual in Artificial Neural Network computation (Dreyfus, 2005), we consider an artificial set of observations, which we obtained by a complete randomization. This kind of data is supposed to be definitely not correlated; therefore, if we are able to obtain the expected results from this data set, we can show that this procedure is able to reach the claimed goal (efficiency). Basically, artificial data is more useful when one wants to measure the efficiency of procedures, whereas real data, which is supposed to be strongly correlated, is more helpful when one wishes to evaluate the effectiveness. At this stage, we verify efficiency.

We have taken in account the simulation design explained below. Based on the proposed strategy (see Section 3.2), our project objective was to "Carry out the project on time". We used the SPI as a metric for success. Higher SPI values (more or equal to 1) point out higher performances, and lower values (less than 1) point out lower performances. We picked 8 attributes to characterize a project: Schedule Performance Index (SPI), as already mentioned, Effort/TeamSize rate (ETS), Developer Experience (DVE), Analyst Experience (ANE), Advanced Development Tool Use (TOL), Software Kind (SWK), Software Category (SWC), Required Software Reliability (RSR). Table 2 shows all the values that each attribute could get. In particular, the cardinality of the Input Set was 600,000 elements. Firstly, we considered an ANN for regression.

Table 2: Project attributed taken in consideration.

Attribute	Values	Cardinality
SPI	**0.12:0.05:1.10**	**20**
ETS	**10:10:100**	**10**
DVE	**1:1:5**	**5**
ANE	**1:1:5**	**5**
TOL	**1,0**	**2**
SWK	**1:1:3**	**3**
SWC	**1:1:4**	**4**
RSR	**1:1:5**	**5**

We used a feedforward neural network (Rumelhart, Hilton, Williams, 1986), (Dreyfus, 2005), made by three levels (one hidden layer), with 7 input units (one for each attribute in Table 2, except SPI), 5 hidden units, and 1 output unit (SPI). The hidden unit activation functions were sigmoid (hyperbolic tangent) and the output had a linear function.

Table 3: Project data (columns) and instructions (bottom) as provided to MatLab™ for simulation.

```
p = [...
1.1  1.0  1.3  1.0  1.2  0.9  1.3  1.2  1.4  1.1...
60   90   80   50   60   40   30   100  20   50 ...
4    1    2    5    1    4    1    3    4    5  ...
4    1    1    3    2    4    5    1    2    5  ...
0    0    1    0    1    0    1    0    1    0  ...
1    1    2    3    1    2    3    1    2    3  ...
1    4    4    1    4    1    3    2    3    4  ...
4    3    2    4    1    5    4    3    2    1  ...

  1.4  1.0  0.9  0.8  0.5  0.6  0.7  0.5  0.4  0.8;
  20   60   90   80   100  10   10   40   20   100;
  3    4    1    5    2    3    4    5    2    1;
  4    5    3    2    1    1    4    5    2    4;
  1    0    1    1    1    1    0    0    1    0;
  1    2    3    1    2    3    1    2    3    1;
  4    3    1    2    2    2    4    1    3    2;
  5    4    3    2    1    5    3    1    4    5; ];


tv = [...
0.43  0.91  0.45  0.52  0.37 0.29 0.61  1.04  0.88  0.26
30    20    30    90    30   20   50    90    80    10;
3     3     2     3     4    5    4     1     1     2;
2     2     5     2     2    2    5     3     2     1;
1     0     1     0     1    1    0     1     0     1;
3     1     3     3     1    2    2     1     1     3;
1     4     4     2     1    4    3     1     2     2;
2     1     5     3     2    5    1     3     2     1;
];
```

As a training set of this regression network, we considered matrix *p* (Table 3) except the first row, which was used as a target set. In particular, each column of *p* represents a past project and each row

represents all of the observations on the considered input variable. A completely randomized validation set (matrix *tv*, Table 3), was also considered. We trained the network by the optimization algorithm called Levenberg-Marquardt with early stopping on matrix *tv* (validation set) and mean square error (network performance) of 0.05. In order to increase the performance of this regression network, firstly, we carried out data normalization, that is, we transformed the training set in a set with mean 0 and deviation standard 1. Subsequently we performed Principal Component Analysis (impact rate of an input variable less than 0.05) (Jollife, 1986), as said above (Dreyfus, 2005). Note that, this kind of analysis reduced our input set from 7 to 6 input variables. Then, we exploited this regression network to split the training set into two subsets: the first one composed of all values of *p* that showed SPI values more than regression function (success) and the second one composed of all other values (failure). In particular, we obtained the following classification result, represented by vector *t*:

$$t = [0\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 1];$$

Where 0 means that the corresponding column in p belongs to the failed subset, and 1 means that the corresponding column in p belongs to the success subset. Therefore, for instance, the first three columns of p belong to the failed subset and the fourth one belongs to the success subset. Then, we built a classification network where the training set was the overall matrix p and the target set was t (Lanubile, Visaggio, 1997). The topology of this classification network was also of 5 hidden units of hyperbolic-tangent activation functions and an output unit of sigmoid logarithmic function. We also trained this second network, by the Levenberg-Marquardt algorithm with early stopping on the overall matrix *tv* and mean square error of 0.05. The target vector for the validation set was *tt* = [0 1 0 0 0 0 0 1 1 0]. Also for this second network, we carried out data normalization and Principal Component Analysis (rate = 0.05), which reduced our training set from 8 to 7 input variables. We made use of MatLab™ Neural Network Toolbox for training all of the considered networks and performing principal component analysis. In order to evaluate the performance of this second network, we used VMRE (the mean square error calculated on the validation set) and TMRE (the one calculated on the training set). We obtained the following performances:

$$\begin{aligned} &\text{VMRE} = 0.2204 \\ &\text{TMRE} = 0.0304 \end{aligned} \qquad (8)$$

Notice that, although the validation set was obtained by a complete randomization (there should be no correlation between training set and validation set), we obtained acceptable results using, as well.

Finally, we calculated the prior probability as said above. In particular, the cardinality of Input Set (V) was N = 600,000, hence, the overall Input Set was a matrix 8×600,000. We fed the network with those 600,000 vectors and obtained the classification map depicted in Figure 8, where x-axis represents the network output, and y-axis shows the number of occurrences for each considered output slot.

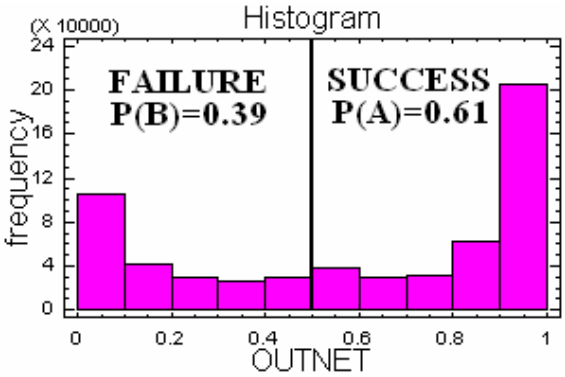


Figure 8: Histogram obtained by an Artificial Neural for Classification.

In Table 4, we reported a descriptive analysis of the obtained results. The calculation of the prior probability, that is, the success/failure probability before the project starts, is executed by Expression (9).

$$\begin{aligned} &P(\mathbf{A}) = 366{,}078/600{,}000 = 0.61 \\ &P(\mathbf{B}) = 233{,}922/600{,}000 = 0.39 \end{aligned} \qquad (9)$$

As said in Section 3.3, we split the histogram in Figure 8 by considering OUTNET = 0.5, as a bound, where OUTNET stands for artificial neural network output. Note that, for each possible value that we provide to the classification network (typically the Input Set *IS*), we get a value that represents the posterior probability that the considered input belongs to class A. Prior and posterior probabilities thus calculated can be used, respectively, as a probability estimate in achieving project objective before the project starts (Analysis phase) and whenever a further project iteration has been carried out (Control phase).

Class	Lower Limit	Upper Limit	Frequency	Cumulative Frequency	Cum. Rel. Frequency
1	0.0	0.1	105952	105952	0.1766
2	0.1	0.2	41689	147641	0.2461
3	0.2	0.3	29651	177292	0.2955
4	0.3	0.4	26361	203653	0.3394
5	0.4	0.5	30269	233922	0.3899
6	0.5	0.6	37560	271482	0.4525
7	0.6	0.7	29595	301077	0.5018
8	0.7	0.8	32219	333296	0.5555
9	0.8	0.9	62158	395454	0.6591
10	0.9	1.0	204546	600000	1.0000

Mean = 0.590044 Standard deviation = 0.366506

Table 4: Descriptive Analysis of results.

In figure 9, we reported further representation of the classification.

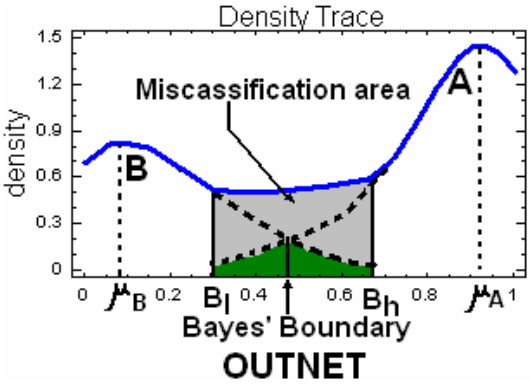


Figure 9: A geometrical interpretation of Bayes' decision rule.

The blue continuous line is the obtained probability density function. Concerning Figure 9, in a theoretical case, those two dashed tails (Figure 9) together with their continuances (blue) would represent the probability density functions that a generic element in the population belongs to A or B, respectively. Green area, below their intersection, is the theoretical misclassification area, that is, the one that Bayes' theorem would address. Actually, we got the blue continuous line. Therefore, the actual misclassification area is the one that is bounded by B_l and B_h (grey and green area). Figure 9 also shows that, if we pick up OUTNET = 0.5 as a class boundary, we can correctly manage this misclassification area and obtain the expected results.

5 METHODOLOGY IMPROVEMENT

It is important noting that, performances of such a classification network could be improved by carrying out cross-validation (Dreyfus, 2005). Sometimes, real applications do not allow us to split training set into two subsets, the one for training and the other for validating. For this reason, in order to evaluate the generalization capability of such a network, we could apply leave-one-out technique. For more details about those optimization procedures, it is worth consulting (Dreyfus, 2005) and (Bishop, 2006), which provide sound mathematical motivations about the optimization of ANN(s) together with a number of practical implementation techniques.

6 CONCLUSION AND FUTURE WORK

This paper proposed a new model for risk evaluation, and an approach to build a statistical neural network-based framework for supporting risk management process. We showed that such a tool is able to provide suitable probability estimates exploiting both prior information (the one that is available before the project starts) and posterior information (the one that is available after events have occurred). Future work will be concerned with using data gathered by real applications for effectiveness validation of the proposed approach. Our prospective work should also investigate its effectiveness when it is used together with other frameworks in terms of reliability and effort improvement in carrying out risk management process.

ACKNOWLEDGEMENTS

We would like to thank people at the University of Maryland Empirical Software Engineering Group for discussing topics presented above, and providing suggestions for improvement.

REFERENCES

Alberts, C.J., 2006. Common Element of Risk. *TN-014*, pp. 1-26, CMU/SEI.

Basili, V., Caldiera, G., and Cantone, G., 1992. A Reference Architecture for the Component Factory. *Transactions on Software Engineering and Methodology*, vol. 1(1): 53-80, ACM.

Basili, V. R., Caldiera, G., Rombach, H. D., 1994. *Goal Question Metric Paradigm*. In Encyclopedia of Software Engineering, Ed. J.J. Marciniak, John Wiley & Sons.

Basili, V. R., Caldiera, G., Rombach, H. D., 1994. *The Experience Factory*. In Encyclopedia of Software Engineering, Ed. J.J. Marciniak, John Wiley & Sons.

Berntsson-Svensson, R., Aurum, A., 2006. Successful Software Project and Product: An Empirical Investigation. In *ISESE06, Intl. Synposium on Empirical Software Engineering*, IEEE CS Press.

Bishop C., 1995. *Neural Network for Pattern Recognition*, Oxford University Press.

Boehm, B.W., 1989. *Tutorial on Software Risk Management*, IEEE CS Press.

Boehm, B.W., 1991. Software Risk Management: Principles and Practices, *IEEE Software*, No. 1, pp. 32-41, IEEE CS Press.

Briand, L.C., Basili, V.R., Thomas, W.M., 1992. A Pattern Recognition Approach for Software Engineering Data Analysis. *Transactions on Software Engineering*, Vol. 8, No. 1, pp. 931-942, IEEE CS Press.

Cantone, G., Donzelli, P., 2000. *Production and Maintenance of Goal-oriented Measurement Models.* Intl. Journal of Software Engineering & Knowledge Engineering, Vol. 10, No. 5, pp. 605-626. World Scientific Publishing Company.

Charette, R.N., 1989. *Software Engineering Risk Analysis and Management*, McGraw-Hill.

Chen, Z., Mezies, T., Port, D., Boehm, B. W., 2005. Feature Subset Selection Can Improve Software Cost Estimation Accuracy. PROMISE06, Conference on Predictor Models in Software Engineering. ACM.

CMMI Product Team, 2002. *Capability Maturity Model Integration (CMMI), Version 1.1. TR-012*, pp. 397-416, CMU/SEI.

Dreyfus G., *Neural Networks Methodology and Applications*, Springer, 2005.

Efron B. and Tibshirani R. J.. *An Introduction to the Bootstrap*, volume 57 of Monographs on Statistics and Applied Probability. Chapman & Hall, 1993.

Fussel, L., Field, S., 2005. The Role of the Risk Management Database in the Risk Management Process. *ISCEng05, International Conference on Systems Engineering*, IEEE CS Press.

Higuera, R.P. , 1996. Software Risk Management. *TR-012*, pp. 1-48, CMU/SEI.

John G., Kohavi R., Pfleger K., 1994. *Irrelevant features and the subset selection problem. 11th Intl. Conference on Machine Learning*, pp. 121-129. Morgan Kaufmann.

Jollife I.T., 1986. *Principal Component Analysis*, Springer.

Jones, C., 2002. Patterns of large software systems: failure and success. *Computer*, N.o 23, pp. 86-87, IEEE CS Press.

Madachy, R.J., 1997. Heuristic Risk Assessment Using Cost Factors. *Software*, pp. 51-59, IEEE CS Press.

Piattini, M., Genero, M., Jiménez, L., 2001. *A Metric-Based Approach for Predicting Conceptual Data Models Maintainability*. Intl. Journal of Software Engineering and Knowledge Engineering 11(6): 703-729, World Scientific.

Rumelhart, D.E., Hilton, G.E., Williams, R.J., 1986. Learning internal representations by error propagation, *Nature*, pp. 318-364, Nature Publishing Group.

Karolak, D.W., 1997. *Software Engineering Risk Management*, IEEE CS Press.

Kontio, J., 1996. The Riskit Method for Software Risk management, Version 1.00. *TR-3782*, UMDCS.

Lanubile, F., Visaggio, G., 1997. Evaluating Predictive Quality Models Derived from Software Measures: Lessons Learned. JSS p. 38:225-234, *Journal of Systems and Software*, Elsevier.

Linberg, K.R., 1999. Software developer perceptions about software project failure: a case study. JSS, pp. 177-192, *Journal of Systems and Software*, Elsevier.

Procaccino, J. D., Verner, J. M., Lorenzet, S. J., 2006. Defining and contributing to software development success. *Commununications of the ACM*, No. 49, ACM.

Roy, G. G., 2004. A Risk management Framework for Software Engineering Practice. *ASWEC04, Australian Software Engineering Conference*. IEEE CS Press.

Sarcià A. S., Cantone, G., 2006. Using Artificial Neural Networks to Improve Risk Management Process. *TR06*, ESEG-DISP, Univ. of Roma Tor Vergata.

Srinivasan, K., Fisher, D., 1996. Machine Learning Approaches to Estimating Development Effort. *Transactions on Software Engineering*, IEEE CS Press.

Standish Group, 1994. CHAOS Report 1994-99. http://www.standishgroup.com, last access Feb. 2006.

Tversky A., and D. Kahneman, 1974. Judgment under Uncertainty: Heuristic and Biases, pp. 1124-1131, *Science*, AAAS Press.

Verner J.M., Cerpa N., 2005. Australian Software Development: What Software Project Management Practices Lead to Success? *ASWEC05, Australian Software Engineering Conference*. IEEE CS Press.

Wallin, C., Larsson, S., Ekdahl, F., Crnkovic, I., 2002. Combining Models for Business Decisions and Software Development. Euromicro02, *Euromicro Intl. Conf.*, pp. 266-271, IEEE CS Press.

Williams, R.C., Pandelios, G.J., Behrens, S.G., 1999. Software Risk Evaluation (SRE) Method Description (Version 2.0). *TR-029*, pp. 1-99, CMU/SEI.