Information Systems Frontiers manuscript No.
(will be inserted by the editor)

I-COMPETERE: Using Applied Intelligence in search of competency gaps in software project managers.

Ricardo Colomo-Palacios · Israel González-Carrasco · Jose Luis López-Cuadrado · Antonio Trigo · Joao Varajao

Received: date / Accepted: date

Abstract People in software development teams are crucial in order to gain and retain strategic advantage inside a highly competitive market. As a result, human factors have gained attention in the software industry. Software Project Managers are decisive to achieve project success. A competent project manager is capable of solving any problem that an organization may encounter, regardless of its complexity. This paper presents I-Competere which is a tool developed to forecast competence gaps in key management personnel by predicting planning and scheduling competence levels. Based on applied intelligence techniques, I-Competere allows the forecast and anticipation of competence needs thus articulating personnel development tools and techniques. The results of the test, using several artificial neural networks, are more than promising and show prediction accuracy.

Keywords Competency Gaps · Software Engineering · Project Manager · Neural Networks · Machine Learning · Genetic Algorithm

Ricardo Colomo-Palacios
Department of Computer Science, Universidad Carlos III, Av. Universidad 30 - 28911. Leganes (Madrid). Spain
Tel.: +34 91 624 5958
Fax: +34 91 624 9129
E-mail: rcolomo@inf.uc3m.es
rcolomo@inf.uc3m.es of Ricardo Colomo-Palacios

Israel Gonzalez-Carrasco E-mail: igcarras@inf.uc3m.es · Jose Luis Lopez-Cuadrado E-mail: jllopez@inf.uc3m.es · Antonio Trigo Ribeiro. Institute of Accounting and Administration of Coimbra, Quinta Agrícola, Bencanta, 3040-316 Coimbra. Portugal E-mail: antonio.trigo@gmail.com · Joao Varajao. University of Tras-os-Montes e Alto Douro, P.O. Box n. 1013 5001-801 Vila Real. Portugal E-mail: joao@varajao.com

1 Introduction

The sustainable success of any organization is strongly associated with the success of the Information System (IS) projects given that IS are considered to be an important component of flexibility for all types of enterprises [28]. In a rapidly changing business and technological environment, the ability to develop and deploy new systems is an important asset that can differentiate one organization from another [62]. Moreover, organizations must continuously innovate in terms of product, process, market and business model in order to remain sustainable [61]. Many organizations rely on their innovation process and, in many cases, are dependent on IT systems. Due to its critical role, there are many studies devoted to demonstrate the implications of IT in corporate innovation (e.g. [13]).

The importance of IT for organizations has shown that software development is an important issue, not only in IT strategy but also in the overall strategy of the organization [60]. Both the industry and academics have acknowledged the importance of software development industrialization [81] as a response to the software crisis, detected in the late sixties and later defined as endemic or chronic (e.g. [31]). More recently, several authors (e.g. [44, 33]) stated that the term "software crisis" is not appropriate or fully justified. However, software project failure rates are quite high [21]and Software Project Management (SPM) is indeed a critical issue for the future of the software industry [74].

SPM is a relatively recent discipline that emerged during the second half of the 20th century [48]. This function has proven to be one of the most difficult tasks in software development [40]. Some authors have indi-

cated that the influence of competencies on the success of projects has not been successfully explored (e.g. [29, 57]). On the other hand, many studies have been carried out to identify and analyze project managers' competencies (e.g. [4,17,56]) in order to find out which fundamental characteristics makes project managers competent and successful.

I-Competere which is based on applied intelligence techniques, predicts competence gaps for SPMs to build better software development teams and to design personnel development solutions so that corporation's competence needs can be ultimately fulfilled. Therefore, this paper focuses on the application of an Artificial Neural Network (ANN) based framework and an optimization methodology to detect competence gaps within software development teams while focusing its predictions on Planning and Scheduling competencies for SPMs. The methodology used is guided with the assurance of the proposed neural model as well as the optimization of its performance both in runtime and accuracy. I-Competere obtains an accuracy rate of 93.23% in its prediction of Planning and Schedulling competencies , which is remarkable for its forecasting accurateness.

It is undeniable that personnel working in software development teams are one of the most decisive resources for the success of projects but they can also be the cause of failure [55]. Recent research (e.g. [23]) suggests that human factors have been largely overlooked or have not been based on empirical studies, in spite of their importance [58,25]. Given this situation, relying on competent SPMs can be advantageous not only for a given project, but also for the organization as a whole. The aim of this paper is to introduce I-Competere, a tool designed to detect competence gaps within software development teams. In the case of I-Competere, these competence gaps are focused on a given competence: scheduling. There are two main reasons for focusing on this specific competence. The first reason is the intrinsic limitation that both technologies and available data present to predict different factors in a given data set. I-Competere should be able to extend its applicability when predicting more than one variable when managing large data sets such as historical data . However, in its current form, it is able to predict a single competence from a defined set of competencies. Taking this into account, authors need to decide which competence is the most accurate in terms of applicability and impact. Thus, the second reason is the applicability and impact of scheduling and planning as a SPM competence. According to [42], despite the proclaimed novelty of the Project Management approach, most recent approaches adopt Fayol's Elements of Management when attempting to define the responsibilities of the project manager: Planning, Organizing, Commanding, Coordinating and Controlling. When focusing on software project management, according to [75], the typical functions of software project management are planning, team building, monitoring, decision making and reporting. Furthermore, some authors stated that planning is a key competence for SPMs [64]. In any case, this competence is one of the most important competences for SPMs and is crucial for project success and common to all project types.

The remainder of this paper proceeds as follows. Section 2 outlines the relevant literature in the area of applied intelligence and its implications on software development. Section 3 discusses the main features of I-Competere, including a usage scenario and the main components of its architecture. Section 4 describes the assessment of this tool. This section also includes a description of the sample, the method used along with test results and a final discussion. Finally, this paper concludes with a discussion on research findings, limitations and concluding remarks.

2 Literature Review

The theory and modeling of Artificial Neural Networks (ANN) have been inspired by the structure and operation of biological neural systems, in which the neuron is the main element. Neurons are the cells which form the cerebral cortex of living beings. A neuron is a microscopic structure composed of three parts, namely, cell body, axon and dendrites. Santiago Ramon y Cajal [14] defined the structure of the Central Nervous System as it is known today. He also showed the biological mechanisms that govern the morphology and the connection processes of the nerve cells. According to these premises, the brain continuously receives input signals from many sources and processes them to provide the appropriate output response. The brain has billions of neurons that interconnect to form neural networks. These neural networks execute the millions of necessary functions needed to sustain normal life.

ANN is an information processing paradigm that is inspired by the biological nervous system [5]. ANN is also considered a mathematical model, composed of a large number of elements or processing units also called neurons. Neurons work together in order to solve specific problems. Similar to its structure, a neural network is a system that connects neurons through a network

and distributes them in different levels to produce an output stimulus.

Finally, there are many ANN types classified according to characteristics such as topology, learning strategy or the type of input received. Thanks to their computational power, generalization capacity and dynamics properties, ANNs have been successfully used in solving complex problems in various fields such as process control (e.g. [51]), vehicle driving (e.g. [63]), weather forecasting (e.g. [34]) or medical diagnosis (e.g. [8]).

2.1 ANNs applications: People in Software Projects

Given the importance of ANN in various fields, human resource management presents some solutions where this technique can be applied.In fact, according to [7], ANNs have been widely applied in social science research. Therefore, [76] used ANN to support performance management, [43] integrated fuzzy data mining and fuzzy artificial neural networks for discovering implicit knowledge, [79] employed a single-layer perceptron neural network model to simulate the knowledge transfer process in organizational environments, [71] included a combination of soft computing techniques to support competency based selection and assignment of human resources and [82] resorted to ANNs to support organizational innovation by exploring competencies.

In the software engineering field and with regard to people's needs, the use of ANNs is not scarce,as it is one of the most widely used in this field [2]. Research conducted by [1] shows an example of Bayesian regularization of an ANN for estimating the size of software developments. [47] estimated the cost of the software by means of a Wavelet ANN. [77] also predicted the reliability of the software by means of hybrid models, citing some of the most representative studies in the field. More recently, [26] focused on an ANN methodology applied to force effort estimation based of a software project.

I-Competere, the application presented in this paper, focuses on SPMs' competencies. Project Managers have been the focus of previous studies in the field of applied intelligence (e.g. [46]); however, no effort has been made to forecast competence gaps in the field of software development projects.

3 I-Competere Architecture

I-Competere focuses on detecting competence gaps within software development teams using applied intelligence techniques. This function leads to build better software development teams and to come up with personnel development solutions to fulfill corporations' competence needs.

Based on this hypothesis, this study describes the designed ANN based framework to predict skill competencies using a number of relevant characteristics of software development teams. For this purpose, several ANNs techniques and heuristics have been applied since they allow induction of knowledge. For example, they are able to generalize behaviours based on unstructured information from previous examples. Furthermore, ANNs can be applied to a wide range of high-complex and non-linear domains, due to their variety of design alternatives. Nevertheless, the variety of design alternatives can sometimes be a disadvantage: the lack of guidelines leads the designer to make arbitrary decisions or to use brute force techniques. Some new theoretical approaches have been proposed in order to facilitate the design process, but they have not been considered as analytical because they cannot be applied in all cases [65].

Therefore, an optimization methodology has been applied in order to guide the search for the best neural model in a given problem and hence improve the performance of this task both in time and accuracy. To homogenize the large number of alternatives, they have been grouped into three phases or stages, following the premise "$neuralmodel = pattern + architecture + algorithm$". For each term of the equation, different techniques and methods will be analyzed in order to improve the final performance.

In order to obtain a reliable framework, the description of the data used is gathered from the research conducted by [70]. This study analyzes the role of the project manager from the perspective of the Team Software Process (TSP), and it considers the required skills, attitudes and knowledge for a software development project. Taking into account the team leader's competencies in the skill environment, I-Competere predicts the scheduling skill for a particular team leader so as to build better software development teams and to design personnel development solutions. In order to maintain a firm's position in the competitive market, an important area of research is the team leader's role in job and activity scheduling.

In order to do so, authors have used information from 10 software development teams formed by 7 components, within a time period of 4 years (2006-2010). Consequently, there are 280 available patterns with information on the software team, the components and the 13 skill competencies. Of the 13 skill competencies, 12 have been used as ANN input and 1 has been used as the output. Table 1 shows a description of each pattern for the team role in a skill environment.

The different ANN components of the I-Competere framework will be fully described and the optimization process will be defined so as to obtain the best configuration for each detailed ANN strategy.

3.1 ANN Components

The input vector for each ANN is created based on the features described in Table 1, and will be the following:

$V_{inp} = \{TC, KI, OD, CF, WQ, MF, MD, EV, PI, LT, BR, GO\}$

The output vector predicts what the skill scheduling of a team leader will be $V_{out} = \{SC\}$.

The most important features of the ANN networks included in this study are detailed in Table 2. Two complementary stopping criteria were used during the network training for all ANN alternatives: reaching a specific number of epochs, and early stopping (ES). Therefore, with the set of data generated by numerical simulation, a part has been assigned to the network's training (train and validation subset), whereas others have been assigned to test the results obtained (test subset). To improve the performance of an ANN model, the extended BackPropagation algorithm has been included. This algorithm uses the learning rate (η) and the momentum coefficient (μ) parameters in order to achieve its best. There is no general rule of thumb for determining the best values of μ and η. However, several researchers have developed various heuristics for their approximation. In this case, we have used a genetic algorithm to find the best values for each ANN strategy. Furthermore, the batch training mode was selected because it presents a balanced behavior between accuracy and speed [78], and the heuristic of Haykin has been used to initialize the weights [37].

In regression, the network approximates the output pattern from an input pattern using a nonlinear continuous function (F). In this case, the quality of an ANN model during the validation phase is obtained by calculating the coefficient of linear correlation (r) that exists between the actual values of that variable and those obtained by the ANN, being then calculated and compared to each output variable. This ratio gives an indication of similarity and the network's accuracy of response after the training process:

$$r = \frac{\frac{\sum_i (x_i - \bar{x})(d_i - \bar{d})}{N}}{\sqrt{\frac{\sum_i (d_i - \bar{d})^2}{N}} \sqrt{\frac{\sum_i (x_i - \bar{x})^2}{N}}} \quad (1)$$

where the numerator corresponds to the covariance and the denominator corresponds to the product of the standard deviations of the variables x (value obtained) and d (expected value). This statistical indicator describes the intensity of the relationship between two sets of variables x (output obtained) by the network and d (expected output), for example,the linear relationship between the variables is the intensity measure. As a result, it shows whether the value of a variable x increases or decreases in relation to another variable's value y. In addition, the indicator can take a value in the range $[-1, 1]$. When the value is closer to 1, a strong positive association appears. On the contrary, if the value is closer to -1, a stronger and negative association appears, i.e., an increment in the variable x produces a decrement in d. Finally, if the value is zero or near zero, the variables are uncorrelated. It is said that a correlation is significant when it is between $\|0.7, 1\|$.

3.2 Optimization Process

Following the assumption of "$neuralmodel = patterns + architecture + algorithm$" the developed framework is initially divided into three components with each one linked to a term of the equation. Figure 1 shows each one of the designed framework's components and their interactions. The ANN models have been designed using NeuroSolutions 5.0.

A number of techniques, methods and heuristics have been included for each of the items in the previous equation with the aim of improving the performance and accuracy of the neural models used. The main goal is to determine the best ANN in this scenario and to improve the performance by optimizing the network design and by maximizing the input patterns.

This paper defines the optimization process described in the methodology used in the research conducted by [35] to regression problems. Furthermore, the previous study is extended with the introduction of a new stage that will be tested using I-Competere. Therefore, in

Table 1 The team leader's role in a skill environment

	Skill Indicator	***Acronym***
Inputs (V_{inp})	Maintain team communication	TC
	Identify key issues	KI
	Make objective decisions	OD
	Combine forces	CF
	Work quality as a challenge	WQ
	Meeting facilitator	MF
	Establish and maintain discipline	MD
	Enterprise vision	EV
	Promote initiative and creativeness	PI
	Lead the team effectively	LT
	Being resolute	BR
	Make the team goal oriented	GO
Output (V_{out})	Scheduling	SC

Table 2 Initial features of the ANN networks

Feature	**Description**	**References**
Topology	MLP, RBF, SVM, RNN, ELN and SOFM	See [35] for more details
Inputs	V_{inp}	-
Outputs	V_{out}	-
Hidden Layer and Neuron	SVM and RBF: 0 Layer. Other ANNs: 1 Layer with 4 neurons	Rules of Tarassenko in [80]
Activation Function (input-hidden-output)	f_{iden}-f_{tanh}-f_{tanh}	Based on [10,49,27]
Patterns distribution (train-test-validation)	33%/33%/33% with Hold-Out Cross Validation	Based on [78,59,52,18]
Training Algorithm	Extended BackPropagation	See [83,78,66] for more details
Learning Parameters	Genetic algorithm in each topology: μ and η for input layer and output layer	In accordance with the guidelines of [67]
Cost Function	MSE simple	-
Weight Update	*Batch*	Based on [78]
Weight Initialization	Haykin Heuristic	Based on [37]
Convergence Criteria	*Epochs* [10000] and Early Stopping	-

order to achieve this optimization process, the different techniques and methods are grouped in different stages. There is one stage for each component of the framework, as well as an ANN initialization stage and a quality stage to ensure the validity of the results.

The complete methodology with all five stages is depicted in Figure 2. The first stage includes simulations with different cost functions and compares the behavior of six well-known alternatives of ANN. The second stage focuses on the analysis of different techniques which enable efficient functioning in an environment where information may be limited. The third stage aims towards increasing efficiency, selecting the smaller network architecture by finding the best configuration of hidden items and pruning the input vector V_{inp}. This stage will also analyze the influence of each variable on the performance of the ANN. Stage four is a new stage which includes a comparison of different learning algorithms in order to improve the performance of the ANN. Lastly, the new quality stage which is stage five, will analyze the quality of each neural model using a criteria based information theory to estimate a quantitative measure of fit that is appropriate for the data used.

The procedure followed for each simulation performed within this process is outlined below:

1. Select one ANN alternative (Stage 1).
2. Select one cost function per ANN (Stage 1).
3. Find the best μ and η parameters per ANN using a genetic algorithm (Stage 1).
4. Apply the resampling and distribution method to the trial set (Stage 2).
5. Select the optimal topology using a genetic algorithm to find the best hidden elements (Stage 3).
6. Select the optimal input vector performing the sensitivity analysis and using a genetic algorithm (Stage 3).
7. Train and test the corresponding ANN using the repeated 20x10 cv method (k-fold cross validation repeated 20 times with weights being randomly initialized and 10 folds).
8. Select the optimal learning algorithm (Stage 4).

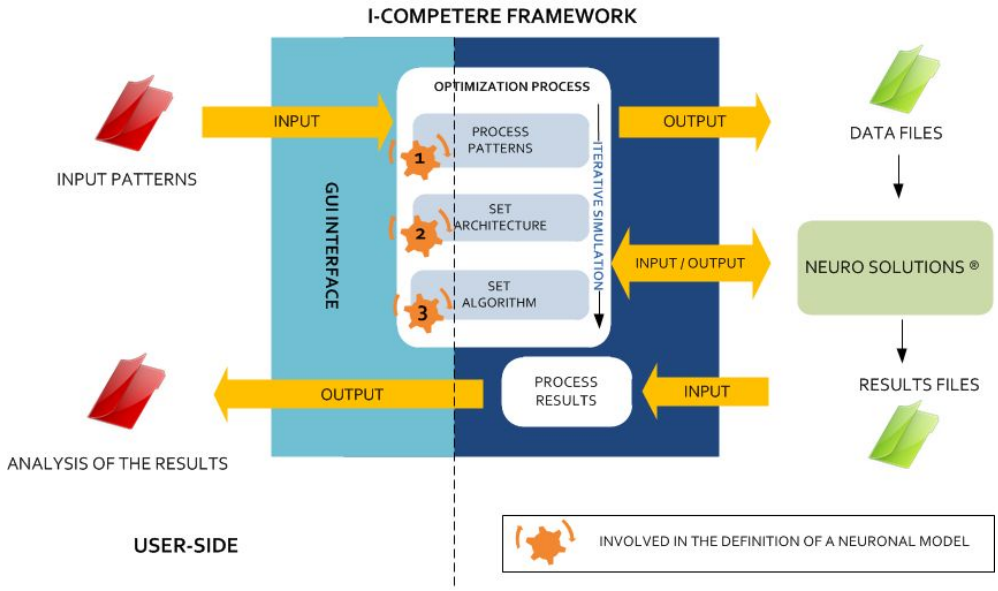

Fig. 1 I-Competere framework

9. Analyze the quality criteria (Stage 5).

Figure 3 shows the component used in for the pattern treatment treatment of the patterns which is related to the Stage 2 of the optimization methodology of I-Competere in stage 2. This component lets the user to select various sampling techniques and to conduct define several experiments using with different configurations for the of distribution patterns.

Finally, Figure 4 shows the interface that leads the training and validation of the ANN after the initialization (Stage 1: Initialization: Topology and Cost Function), the setup of the experiments with different techniques of sampling and distribution patterns (Stage 2), and the definition of the architecture and learning algorithm (Stages 3 and 4). The ANN models have been designed using NeuroSolutions 5.0.

One of the problems inherent to the techniques used in I-Competere is the large number of neural model simulations performed when testing different heuristics, techniques and methods. A simulation framework has been developed in order to reduce the number of resources and the complexity of the tests to be performed as much as possible. This framework includes a number of components and tools designed to configure and automate the learning process of different types of ANN. This facilitates the creation, development and evaluation of a series of experiments in a sequential and iterative way, thus not being dependent of the designer during each one of the simulations.

3.2.1 Stage 1. Topology and Cost Function

In this stage and for each scenario, one ANN alternative is selected as well as the different available cost functions in order to increase the performance of the ANN model. This ensures that the cost function is minimized during the training phase. The different topologies and cost functions incorporated in this research are listed in Table 3.

Therefore, the correct choice of the cost function and network topology is important in order to ensure the most efficient solution when predicting the Scheduling and Planning competencies.

3.2.2 Stage 2. Resampling and Distribution

This stage includes several techniques to improve the performance of the ANN in domains with limited data, maximizing the patterns available for training and testing the network. The list of techniques included in this stage are detailed in Table 4. More details of these techniques can be found in [35].

Moreover, to reduce the influence of random data partition, some authors have proposed the use of a repeated cross-validation (repeated k-fold cross validation) [19,36,9,24].

This technique ensures that Type I and Type II errors (probability of accepting the null hypothesis when it is false, known as equal false) will be appropriate

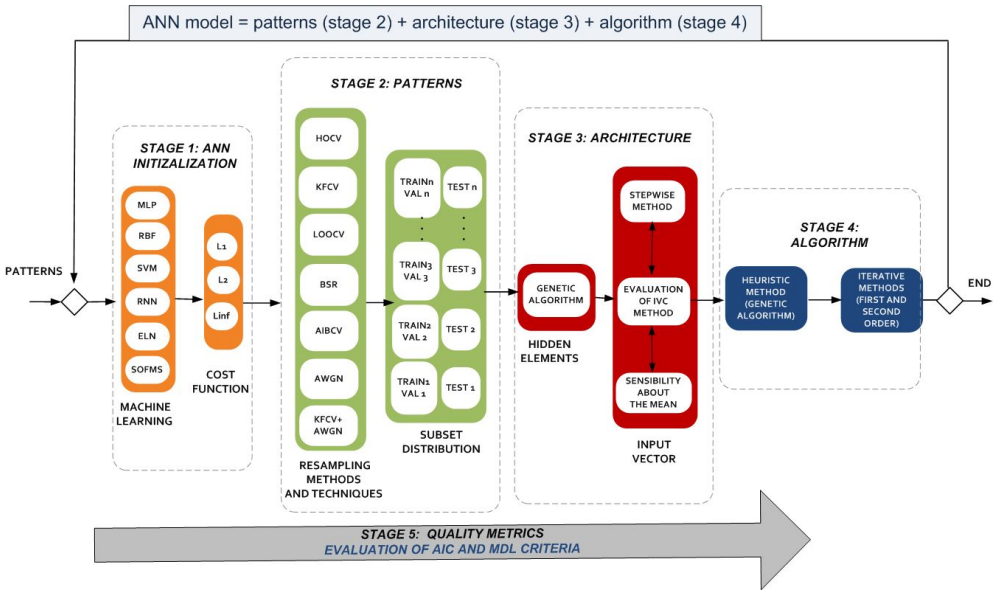

Fig. 2 Stages of the ANN-based optimization process

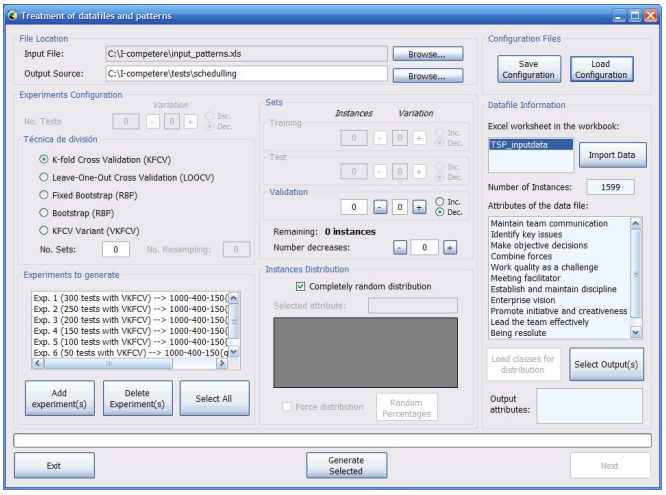

Fig. 3 Example of I-Competere interface analyzing patterns (Stage 2)

Feature	Description	References
Topology	MLP, RBF, SVM, RNN, ELN and SOFM	See [35] for more details
Cost Function	L_p norms: L_1, L_2, L_∞	See [32, 73, 38, 12] for examples

Table 3 Stage 1. Topology and Cost Function

and that reproducibility (the probability that two executions of the same comparison method produce the same results, known as stability) will be greater. Therefore, in this research, the technique of repeated k-fold cross validation using the configuration 20x10cv (i.e. 20 repetitions of KFCV with 10 folds) was used.

The best sampling technique and the ideal distribution of patterns allows I-Competere to achieve the most efficient performance by maximizing the available patterns.

3.2.3 Stage 3. Topology Design and Pruning

To ensure optimal generalization ability, the purpose of this stage is to determine the optimal architecture in each ANN topology. In order to obtain the smallest network possible, a genetic algorithm has been incorporated to determine the best hidden configuration element and to find the best input vector. Furthermore, as shown in Table 5, different heuristics such as Pruning and Sensitivity analysis techniques, have been included to quantify variable importance in the prediction made

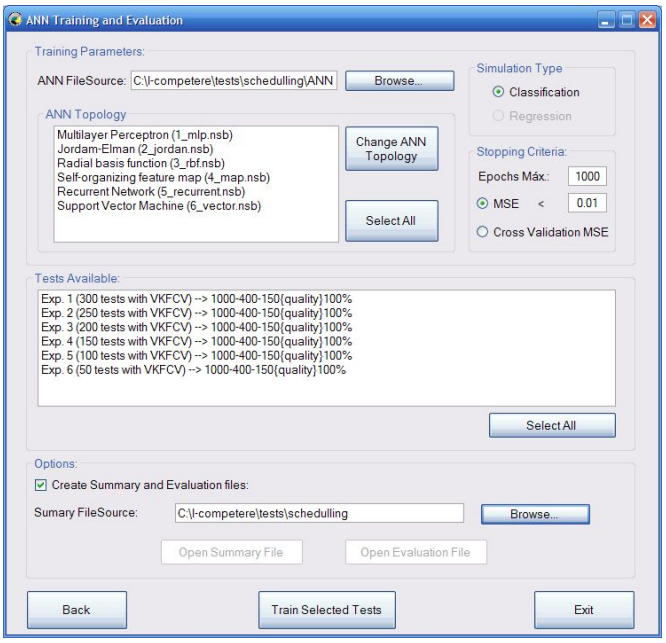

Fig. 4 Example of I-Competere interface during the process of training and validation

Technique	Acronym	Description
Hold-Out Cross Validation	HOCV	Conventional training.
K-Fold Cross Validation	$KFCV_{10}$	KFCV with 10 folds.
Leave-One-Out Cross Validation	LOOCV	KFCV alternative with 220 folds.
Bootstrap	BSR_{100}	Bootstrapping with 100 trials resamplings.
Alternative Iterative and Balanced of Cross Validation	$AIBCV_{100}$	Random generation 100 trials.
Additive White Gaussian Noise (AWGN)	$AWGN_{100}$	Creation of 100 trials with Gaussian distribution noise.
KFCV & AWGN	$K_{10}AWGN_{100}$	$KFCV_{10}$ and $AWGN_{100}$ methods combination.

Table 4 Stage 2. Resampling and Distribution techniques

Technique	Acronym	Description	References
Sensitivity About the Mean	SAM	Shows the determination of the influence of each of the inputs in the output which the network obtains.	See [66,72,53] for more details.
Indicator of Variable Criticality	IVC	Represents in the range $[0,1]$ the number of times which a variable k was outside the range when the prediction has failed, and the total number of times that a variable k has been outside the range.	See [35] for more details.
Backward Stepwise Method	SWM	Eliminates a variable from the input vector sequentially and analyzes the effect on the output of the network.	See [30] for more details.

Table 5 Stage 3. Topology Design and Pruning

by the ANN model.

This stage ensures an optimal generalization ability, since it is responsible for obtaining the smallest ANN architecture possible.

3.2.4 Stage 4. Learning Algorithm

ANN training is based on minimizing the error function E with the variation of the weights set which define the ANN (see Equation 2). Therefore, it is a multi-variable optimization without restrictions since there is no other additional requirement needed with respect to the function or the input variables.

Figure 2 shows the optimization methods applied in this research study applicable to the weight's modification in the ANN's training process. These methods, classified as heuristic and iterative methods, are applied

to multi-variable problems without restrictions where the analytical solution is not easy to find.

$$\frac{\partial E}{\partial w} \equiv 0 \Leftrightarrow \nabla E(w*) = 0 \quad (2)$$

Iterative methods present a loop structure where the problem is initialized and the obtained response is evaluated. Depending on whether certain criteria is met or not, the process either continues or comes to an end. The problem with this kind of method lies in the fact that when the algorithm converges to a solution, it is difficult to recognize if this is a global minimum or simply a local solution.

On the other hand, heuristic methods, unlike iterative ones, represent an alternative for cases in which it is necessary to assure the location of global minimums or cases where the requirements of the optimization methods are not achievable (like derivability of functions restrictions, etc.). A typical example of this kind of heuristic method is genetic algorithms, which try to simulate laws of natural selection when obtaining sets of solutions that better fit their environment.

In Table 6, the different learning algorithms included in this research study are shown. As a result of this stage, the most adequate learning algorithm has been selected.

The learning or adaptation process is a crucial step in the ANN theory, and is responsible for the procedure to extract the required information from the input data. If the learning process is incomplete, the values of the weights will not be close to their optimum values and as a result, the performance of the ANN will suffer.

3.2.5 Genetic Algorithm

Genetic algorithms (GA) are adaptive methods which may be used to solve search and optimization problems. A genetic algorithm is a randomized global search technique that solves problems by imitating processes observed from natural evolution. Based on the survival and reproduction of the fittest, GA continually exploits new and better solutions without any pre-assumptions, such as continuity and unimodality. This evolutionary solution has been successfully adopted in many complex optimization problems and shows its merits over traditional optimization methods, especially when the system under study has multiple local optimum solutions [45].

It is assumed that a potential solution to a problem may be represented as a set of parameters. These parameters (known as genes) are joined together to form a string of values (known as a chromosome). The fitness of each chromosome is then evaluated using a performance function after the chromosome has been decoded. Upon completion of the evaluation, a biased roulette wheel is used to randomly select pairs of better chromosomes to undergo such genetic operations as crossover and mutation that mimic nature. The newly produced chromosomes turn out to be stronger than the ones from the previous generation; they will replace these weaker chromosomes. This evolution process continues until the stopping criteria are reached.

Table 7 Parameters and criterion used in the Genetic Algorithm

Population	50
Maximum generations	100
Maximum time of evolution (minutes)	60
Progression	Generational
Termination Type	Fitness Threshold
Selection	Roulette
Crossover	One point
Crossover Probability	0.9
Mutation Probability	0.01

The advantages of using evolutionary computation like genetic algorithms in conjunction with the ANN have been clearly demonstrated by the scientific community [16,20,39,69,84]. To use the characteristics of genetic algorithm within the different functions of the optimization methodology, a component that has been based on a genetic algorithm has been included in order to determine the number of hidden neurons (Stage 3), to optimize the input vector (Stage 3) and to initialize the parameters of the learning algorithm (Stage 4).

In this study, the criterion used to evaluate the ability of each potential solution is the lowest cost obtained during training. The criteria and values used were chosen or fixed in accordance with the guidelines issued by Principe et al. [67] and their application in practical problems. The values used are shown in Table 7.

3.2.6 Stage 5. Quality Metrics

The last phase of the methodology is aimed at measuring the results of the quality and validity of the various alternatives analyzed. The performance parameters calculated in each ANN for the regression scenario and the correlation factor set out in Section 3.1 are able to set the function and performance of an ANN. However, they do perceive the best choice when various alterna-

Table 6 Stage 4. List of Learning Algorithms

Learning Algorithm	Acronym	Parameters	References
Gradient	GR	Parameter η in hidden and output layer (with Genetic Algorithm)	Introduced in 1847 by Cauchy [15].
Extended BackPropagation	EBP	Parameter η and μ in hidden and output layer (with Genetic Algorithm)	See [83,78,66] for more details.
Conjugate Gradient	CG	Parameter η and μ in hidden and output layer (with Genetic Algorithm)	Invented by Hestenes and Stiefel in 1952 [41].
Quick-Propagation	QP	Second order method	Developed by Fahlman in 1989 [22].
Levenberg Marquardt	LM	Second order method	Created by Levenberg and Marquardt [50,54].

tive models with similar results appear. To facilitate this task, two subtasks have been included to evaluate and compare, from a statistical point of view, the ability of the designed neural models to be generalized.

The first of these tasks includes two statistical estimates that indicate the measure of goodness of fit of the statistical model estimated. These indicators are quality measures based on Akaike Information Criterion (AIC) developed by [3] and the principle of Minimum Description Length (MDL) proposed by [68].

4 Experiments and Results

The basic architecture of the ANNs used in this study are shown in Table 2. The learning rule for the ANNs is the Extended BackPropagation, with or without the parameters, except for the Support Vector Machine (SVM) topology. The values of the learning parameters were approximated using the genetic algorithm component and the rules established in [66]:

- RBF: Extended BackPropagation Learning. η=1.0 y μ=0.7 for the output layer. .
- RNN: Extended BackPropagation Learning. η=0.01 y μ=0.7 for the hidden layer. η=0.1 y μ=0.7 for the output layer.
- SVM: Learning with η=0.01 for the output layer.
- MLP: Extended BackPropagation Learning. η=0.9 y μ=0.8 for the hidden layer. η=0.1 y μ=0.7 for the output layer.
- ELN: Extended BackPropagation Learning.η=0.1 y μ=0.8 for the hidden layer. η=0.1 y μ=0.8 for the output layer.
- SOFM: Extended BackPropagation Learning. η=0.9 y μ=0.7 for the hidden layer. η=0.1 y μ=0.7 for the output layer.

In the following sections, the results obtained for each framework stage are explained in detail. The first stage selects the best ANN strategy for the estimation of competency gaps and the best cost function configuration for each ANN is evaluated. The second stage focuses on the resampling methods and distribution techniques included when working with limited data. The third stage determines the best topology possible and measures the influence of the input variables based on the results of the gap forecasting. The final stage presents the quality and viability of the results obtained during all the stages of the framework.

4.1 Stage 1. ANN and Cost Function selection

Table 9 shows the results obtained for each ANN and the L_p, being the L_2 norm the one with the best behavior found in the majority of the experiments. Only Recurrent Neural Network (RNN) and SVM with L_1 differ. The first conclusion obtained is that the Radial Basis Function (RBF), Multilayer Perception (MLP) and Elman Network (ELN) networks with L_2 norm are the ones with the best performance when analyzing competence gaps. The Self- Organizing Map (SOFM) and SVM are the less classified ones, with an average difference of 0.1705 and 0.1546 after 20 runs in comparison to the RBF with L_2 norm.

Once the best cost function configuration has been determined for each topology, ANN performance will be improved in the following stage by maximizing the input patterns.

Table 9 Stage 1. ANN performance based on different cost functions (correlation factor. 20 runs)

ANN	L_1	L_2	L_{inf}
ELN	0.8071	0.8564	0.7612
MLP	0.8438	0.8659	0.7755
RBF	0.8395	0.8699	0.7818
RNN	0.7512	0.7059	0.6891
SOFM	0.5092	0.6994	0.4591
SVM	0.7153	0.5252	0.6757

Table 8 Overview of ANN and cost function strategies included in the stage 1.

Method	Acronym	Description
Radial Basis Function	RBF	Conscience full competitive rule. Euclidean metric. Cluster Centers=70. 14-0-1 topology.
Recurrent Neural Network	RNN	Partially recurrent. 14-1-1 topology.
Support Vector Machine	SVM	Kernel Adatron algorithm. Step size 0.01. 14-0-1 topology.
MultiLayer Perceptron	MLP	14-4-1 topology
Elman Network	ELN	Inputs feeds the context units. 0.8 time constant. Momentum learning rule 14-1-1 topology.
Self-Organizing Map	SOFM	Square Kohonen Full method. Rows=5, Columns=5, Starting Radius=2, Final Radius=0. 13-1-1 topology.

Table 10 Stage 2. Resampling methods performance (% correlation factor. 20 runs)

ANN	HOCV	KFCV	LOCV	BSR	AIBCV	AWGN	KFCV+AWGN
ELN	0.8564	0.8201	0.8755	0.8256	0.8854	0.8614	0.8893
MLP	0.8659	0.8746	0.8913	0.8425	0.8588	0.9049	0.9150
RBF	0.8699	0.8527	0.8633	0.8575	0.8697	0.8875	0.8924
RNN	0.7512	0.7854	0.8302	0.7971	0.8175	0.8093	0.8148
SOFM	0.6994	0.7543	0.7046	0.7960	0.7665	0.7852	0.7532
SVM	0.7153	0.7893	0.8058	0.7681	0.7183	0.7234	0.8063

4.2 Stage 2. Resampling and Distribution

Early stopping and cross-validation has been introduced in this experimentation to reduce computation time and to prevent the occurrence of overlearning during training. The results obtained after the second stage as shown in Table 10, ratify the conclusions made by the authors in previous research studies. This shows that the techniques included in this framework can be used in the domain of competence gap forecasting and thus, provide ANN performance improvement. The column "HOCV" in Table 10 contains the same results as Stage 1 since this technique has been used in the experiments involving this stage.

The most efficient technique in this stage has been the noise injection with the Additive White Gaussian Noise method (100 new patterns with noise) and ten k-fold cross validation ($AWGN_{100} + KFCV_{10}$), because the accuracy of all tested ANN has increased. Another interesting technique is the Leave One Out Validation (LOOCV). Despite its positive results, LOOCV has certain disadvantages such as its complexity in application as a model selector and its high variability. In this case, there is a lack of continuity, given that a small change in the data can cause a large alteration in the model selected [11]. Although the Bootstrapping technique (BSR with 100 pattern resamplings) is the best resampling alternative for the SOFM network, the results obtained are far from those obtained with ANN. The other techniques, Alternative Iterative and Balanced of Cross Validation (AIBCV with random generation of 100 patterns), Kfold Cross Validation (KFCV with 10 folds) and Additive White Gaussian Noise (AWGN with 100 patterns and with Gaussian distribution noise) present in some cases, an irregular behavior during the experiment and show a lower chance of improvement.

Among the three best networks (RFB, MLP and ELN) the ANN topology with a higher rate of improvement has been the MLP with the $AWGN_{100} + KFCV_{10}$ technique. The RBF and ELN networks also showed a remarkable improvement but do not match up with the results obtained in MLP. Therefore, taking into account the experiments of this stage, the $AWGN_{100} + KFCV_{10}$ with early stopping resampling technique will be used in the following stages.

4.3 Stage 3. Topology Design and Pruning

One of the most important aspects of achieving a proper generalization ability is to adjust the size of the network according to the requirements of the problem to be solved. The two aspects that can be chosen by the network designer are the hidden elements, layers and neurons as well as the variables of the input vector.

For the first case, based on the architecture shown in Table 2, the genetic algorithm has reached the best configuration of hidden elements, layers and neurons for each ANN. As seen in Table 11, one single hidden layer is enough in all the ANN, except for the SVM because it does not need any hidden elements. This hidden elements configuration simplifies the final architecture of

Table 11 Stage 3. Best network design and input vector with Genetic Algorithm (correlation factor. 20x10cv)

ANN	Hidden Layer-Neuron	Genetic Algorithm (Input Vector)	Stage 3
ELN	L_1N_3	Full V_{inp} - {PI,CF,GO}	0.8980
MLP	L_1N_6	Full V_{inp} - {CF,BR,GO}	0.9224
RBF	L_1N_4	Full V_{inp} - {PI,BR,GO}	0.9107
RNN	L_1N_5	Full V_{inp} - {PI,BR,GO}	0.8442
SOFM	L_1N_4	Full V_{inp} - {CF,MF,GO}	0.8153
SVM	None	Full V_{inp} - {CF,BR,GO}	0.8255

the ANNs and the complexity associated with them. Furthermore, Table 11 indicates the best input vector found using the Genetic Algorithm.

Once the input vector has been pruned with the genetic algorithm component, the SWM and SAM and IVC methods are used to corroborate the conclusions. According to the study shown in Table 12, the most important variables of V_{inp} are the following: KI, WQ, MD, LT. These variables should always appear as a component of the input vector to ensure higher accuracy. The least important variables of V_{inp} are the following: CF, PI, BR, GO. Therefore, their non-inclusion will once again simplify the final architecture as well as reduce the network training time. The pruning process provides valuable information not only for the ANN area, but also for project managers For example, it provides relevant variables related to software project managers that should be taken into account when detecting future competency gaps in software projects.

Finally, Table 11 also includes the correlation factor obtained after the location of the best architecture for each ANN has been found (best input vector and best hidden elements configuration). Once again, the results obtained classify the MLP as having the highest accuracy.

4.4 Stage 4. Learning Algorithm

Table 13 Stage 4. Best Learning Algorithm (correlation factor. 20x10cv)

ANN	GR	EBP	CG	QP	LM	Epochs
ELN	0.8188	0.8980	0.9215	0.9304	0.9132	+327
MLP	0.8273	0.9224	0.9199	0.9297	0.9323	-202
RBF	0.8243	0.9107	0.8678	0.8726	0.9103	0
RNN	0.7517	0.8442	0.8642	0.8819	0.8601	-143
SOFM	0.7731	0.8153	0.7901	0.8209	0.8352	+210
SVM	0.8122	0.8255	0.8082	0.8551	0.8990	+197

The learning algorithm variants are intended to help locate the global minimum of error surface in weight space. Each alternative studied, first or second order, is intended to optimize and accelerate this search process using different approaches, such as updating weights, adapting the parameters or using second derivatives of the error.

The results obtained in Stage 4 are presented in Table 13. In the EBP column, the results are the same as those obtained in Stage 3 for all topologies because this algorithm has been used in the initial ANN designs (see Table 2). The epochs column indicates the variation between the best algorithm as well as the epochs in Stage 3 for each ANN topology (with the EBP algorithm).

The results show that in the scenario studied, the algorithms with best behavior in the prediction of competence gaps are the LM and the QP variants. Furthermore, the QP algorithm obtained a lower the number of epochs needed to train the network which reduces the computation time. In regards to the RBF network, the best algorithm for this scenario is the EBP. Finally, the best result was obtained with the MLP topology and the algorithm LM.

In summary, Table 14 provides an accurate rate of each ANN (using the correlation factor) and shows the improvement rate obtained by applying each phase of the framework. The final column, "% Improve", gives the percentage of improvement obtained over the original ANN and after the application of the complete framework. The MLP is the alternative with the highest final accuracy rate of 0.9323. The highest rate of improvement is achieved by SVM (18.37%), and the average rate of improvement for all ANNs is 10.52%. Thus, it can be concluded that the I-COMPETERE framework is applicable when forecasting competency gaps. In addition, the optimization methodology included improves the performance and accuracy of the ANNs used in the framework.
In the next section, quality metrics will be used to rat-

Table 12 Stage 3. Influence of input parameters and the best input vector (SAM, IVC and SWM methods)

ANN	SAM (more and less influential)	IVC (more and less influential)	SWM (Input Vector)
ELN	↑: MD,LT ↓: CF,PI	↑: KI,MD,LT ↓: BR,GO	Full V_{inp} - {PI,CF,GO}
MLP	↑: KI,WQ,MD,LT ↓: CF,PI,BR,GO	↑: KI,WQ,MD,LT ↓: CF,PI,BR,GO	Full V_{inp} - {CF,PI,BR,GO}
RBF	↑: WQ,MD,LT ↓: BR,GO	↑: WQ,MD,LT ↓: CF,BR,GO	Full V_{inp} - {PI,BR,GO}
RNN	↑: KI,WQ,MD, LT ↓: CF,GO	↑: KI,WQ,MD,LT ↓: CF,PI,BR,GO	Full V_{inp} - {PI,BR,GO}
SOFM	↑: KI,LT ↓: CF,MF,GO	↑: KI,MD,LT ↓: CF,MF,BR,GO	Full V_{inp} - {CF,GO}
SVM	↑: KI,WQ,MD ↓: GO	↑: KI,WQ,MD,LT ↓: CF,BR,GO	Full V_{inp} - {CF,BR}

ify the validity of the results obtained.

Table 14 Final summary and comparison between the different ANN (correlation factor. 20x10cv)

ANN	Stage 1	Stage 2	Stage 3	Stage 4	% Improve
ELN	0.8564	0.8893	0.8980	0.9304	7.39
MLP	0.8659	0.9150	0.9224	0.9323	6.64
RBF	0.8699	0.8924	0.9107	0.9107	4.07
RNN	0.7512	0.8302	0.8442	0.8819	13.07
SOFM	0.6994	0.7960	0.8153	0.8352	13.57
SVM	0.7153	0.8063	0.8255	0.8990	18.36

4.5 Stage 5. Quality Metrics

To corroborate the above conclusions, an analysis focused on measuring the quality of each ANN after applying the framework, has been carried out. Two statistical criteria based on information theory, AIC and MDL, are included to compare the degrees of goodness of fit for each proposal. Table 15 shows the results for these criteria sorted by quality, i.e. lower AIC and MDL. Once again, the MLP model showed the best performance and adjusted the data of these statistical indicators. This allows the researcher to choose ANN and be sure of its ability to detect future competency gaps. Furthermore, the other two topologies with better accuracy, RBF and ELN networks, have obtained good performance with the AIC and MLP indicators.

4.6 Numerical Example

A numerical example has been provided to give information on the correlation between the inputs and the prediction of I-Competere. However, the prediction process of I-Competere is iterative and several experimental runs need to be performed.

The numerical example shown in Table 16 reflects one of these runs performed for the MLP topology. The acronyms of the columns are described in Table 1.

The column "SC Output" reflects the prediction obtained by I-Competere for each of the input patterns. The correlation factor obtained in this run, using Equation 1, is 0.9052.

4.7 Discussion

Previous sections have described the results obtained during each stage of the methodology applied. These stages showed how the framework was capable of detecting competency gaps within software development teams Firstly, a selection of six topologies with their respective better cost functions were analyzed. When comparing the results, one would think that the best options for predicting competencies would be RBF, MLP and ELN with L_2 norm. The lack of similar studies in this domain made this stage mandatory in order to optimize the framework. The second phase determined the best resampling method using the best configurations possible. The results of the second phase show that for the majority of the configurations, the $AWGN_{100} + KFCV_{10}$ with early stopping was a technique for the given problem, even though LOCV and BSR achieved better results for RNN and SOFM respectively. At this stage, the network's configuration and the resampling method has been selected. However, the choice of the best configuration implies the optimization of each topology and the selection of the best option which is MLP due to its architecture and performance. The results obtained in phase 4 shows that all topologies require only one hidden layer (except SVM which does not have any hidden layers). The best input network determined by the elements (KI, WQ, MD, LT) revealed the most important skills to consider in estimation. The selection of these skills using a genetic algorithm not only helps improve the network's performance, but also provides valuable information to project managers. As a result, it attracts attention to all the possible skills when predicting future necessities. Once again, with the improved topologies and the optimal set of inputs, the best performance was achieved by

Table 15 Stage 5. Analysis using Task 4.1: Indicators AIC and MDL

ANN	AIC	MDL	ANN	AIC	MDL
ELN	65.991	92.765	MLP	-27.836	-19.760
RBF	2069.34	2208.627	RNN	94.021	88.231
SOFM	879.945	934.081	SVM	5294.477	5628.455

MLP, with little difference concerning the RBF performance. Finally, the last stage selected the best learning algorithm for each configuration, with the best options being LM and QP.

When comparing all the possible alternatives, the MLP was selected for the I-Competere architecture. The systematic approach shown in this experimentation confirms that I-Competere is a valid alternative for the forecasting of the Scheduling and Planning skills, thanks to the careful selection of its architecture. Final results show that the framework obtained an accuracy rate of 0.9323 with little difference in respect to the ELN topology. However, quality metrics have indicated that MLP was the best option.

Finally, it is important to note that the model depends on historical data. All the models were trained using a set of historical data which is coherent in terms of project size, project type and team competence. These three examples are some of the most important aspects to consider. Base on previous studies, authors agree (e.g. [6]) that there may be a considerable variation in the performance of prediction systems when the nature of the historical data changes. The accuracy of I-Competere could change when applied to unknown scenarios.

5 Conclusions and Future Work

This paper presents I-Competere, a framework for competency gaps forecasting in the area of software project management. The framework is able to predict the competence level of a team leader based on their features. In this sense, this study provides a powerful framework used in forecasting competency gaps in software development teams.

I-Competere provides a new point of view regarding the application of intelligent systems in software engineering. As mentioned in the literature review, despite the application of ANN and intelligent systems for size estimation, cost or software projects effort, there is no similar approach related to the forecasting of required Project Manager competencies . However, there is a set of problems in software development projects that are caused by the lack of training concerning the manager or the members of the project team. Based on a set of characteristics related to a software project and its manager, the proposed framework predicts the competency gaps that could be affected during the project. Thanks to the methodology applied, the most relevant variables for this prediction have been identified. The most important variables of V_{inp} are the following: KI, WQ, MD, LT. These variables should always appear as a component of the input vector to ensure higher accuracy. It provides decision makers with valuable information when anticipating future problems and provides training programs for SPMs and project team members before starting a project thus avoiding future problems.

The adoption of the ANN based framework and optimization methodology has lead to compare different ANN configurations, optimizing them and selecting the most accurate one: in this case the MLP. The obtained accuracy rate is 93.23% which is a high indicator of success concerning estimations. This measure leads one to believe that the proposed framework can be a valuable tool for both SPMs and organizations.

Further research will be centered on the inclusion of other types of projects in order to prove that the proposed framework can be adapted when predicting competency gaps in different projects (buildings, manufacturing, etc.). The inclusion of this estimation framework along with the previous studies by other authors on the estimation of software development [26] and its full integration in a collaborative environment will serve as a powerful tool for project management.The adoption of this estimation method combined with the collaborative work between the managers could benefit the project management area. It can be especially advantageous in the very early phases in which project information is scarce but the decisions made can determine project success or failure.

Table 16 Numerical Example of I-Competere operation (1 run)

TC	KI	OD	CF	WQ	MF	MD	EV	PI	LT	BR	GO	SC	SC Output
6	3	2	2	2	4	3	1	7	3	7	0	2	2.1124
7	3	2	2	2	2	3	0	7	3	6	0	2	2.3250
6	3	1	2	2	5	2	2	8	3	7	0	2	1.9848
6	3	4	2	2	5	3	2	5	2	9	0	2	2.2902
6	3	2	3	2	4	3	1	7	3	7	1	2	2.7110
6	3	4	4	2	4	4	2	9	2	5	0	1	2.9347
8	3	3	2	3	4	3	1	7	3	8	1	2	2.8205
0	6	7	1	6	5	7	0	5	6	6	5	6	8.1502
0	6	7	1	6	5	7	0	5	5	6	5	6	8.1503
0	6	8	1	6	5	7	0	5	7	6	5	6	8.2504
2	7	8	2	7	5	7	0	4	7	6	6	6	8.3679
0	7	9	2	6	5	7	1	5	6	6	5	6	8.3561
0	6	7	2	6	3	7	1	5	6	6	6	5	8.3827
0	6	6	0	6	5	7	1	5	6	5	5	6	8.3077
2	0	2	5	0	6	0	8	10	0	10	2	0	0.8938
3	0	2	5	0	6	0	9	10	0	10	2	0	0.8980
2	0	1	4	0	6	0	8	10	0	10	0	0	0.9556
2	0	2	4	0	4	0	7	10	0	9	4	1	1.0118
1	0	2	5	0	7	0	10	10	0	10	2	0	1.1223
1	0	0	5	0	6	0	10	10	0	10	2	0	1.1377
2	0	0	6	0	5	0	9	10	0	10	2	0	1.2125
5	6	4	5	5	6	7	0	10	6	4	8	5	7.6243
5	6	2	5	5	6	7	0	10	6	2	9	5	7.6739
5	6	3	5	5	4	7	0	10	6	4	10	5	7.8913
5	5	4	5	5	4	7	0	10	6	4	9	6	7.7584
6	6	4	4	5	7	7	0	10	7	3	7	5	7.6834
6	6	5	5	5	6	7	1	10	6	4	8	5	7.8131
5	7	4	6	5	6	7	0	8	6	2	6	4	7.9918
6	5	7	4	6	1	5	5	4	5	1	6	6	7.4704
6	5	6	5	6	1	4	5	4	5	0	6	6	7.1866
4	5	7	4	6	0	6	5	4	5	1	6	6	8.0069
6	5	7	4	6	1	5	5	5	5	1	5	6	7.5294
6	5	7	4	7	1	5	5	4	5	0	7	5	7.8857
5	5	7	6	6	1	6	5	4	6	1	6	6	7.9012
7	5	9	4	5	1	5	5	4	4	3	6	6	7.2414
1	0	5	5	1	2	0	3	3	0	1	0	0	2.6879
1	0	5	6	1	4	0	3	3	0	1	1	0	2.7829
0	0	5	6	1	1	1	3	4	0	2	2	0	2.9730
1	0	6	3	1	2	0	1	3	0	1	0	0	2.8546
1	0	6	3	1	0	0	3	3	0	3	0	0	2.8657
0	0	5	5	1	2	0	4	3	1	1	1	0	3.3614
1	0	5	5	1	2	0	5	4	0	1	2	0	3.1548
6	6	2	5	6	10	7	8	4	7	2	9	7	7.1151
8	6	2	3	6	10	7	8	4	7	3	8	7	6.8940
6	5	3	5	6	10	7	9	3	7	2	10	7	6.7805
6	6	2	6	6	10	8	8	4	7	2	9	7	7.4502
7	6	3	5	6	9	7	10	4	7	2	8	8	7.1647
6	6	0	5	6	10	7	8	2	7	2	9	7	7.1340
8	5	4	5	5	10	7	8	4	6	2	8	7	5.9271
8	5	8	5	6	2	5	9	3	6	9	10	6	7.5956
7	5	8	5	6	2	5	10	3	6	10	10	6	7.7063
7	5	8	5	6	2	5	9	4	6	10	9	7	7.7709
9	5	6	5	6	2	5	9	5	6	9	10	6	7.6571
8	5	8	5	6	1	4	9	3	5	10	10	7	7.6796
8	5	7	5	5	2	5	9	3	6	9	10	6	7.5747
8	5	7	5	6	0	5	9	4	6	10	10	5	8.0554
9	9	3	9	9	2	8	0	10	9	0	9	9	8.4684
9	9	1	9	10	2	8	0	10	9	2	9	9	8.4894
9	9	3	10	9	2	8	0	10	9	0	7	9	8.4761
9	9	3	10	8	0	8	1	10	9	0	9	10	8.5009
9	9	4	9	8	2	8	0	9	9	0	9	9	8.4879
9	8	3	10	9	0	8	0	10	9	0	9	9	8.5144
9	9	3	9	9	4	8	0	10	9	2	7	9	8.4992
9	6	7	4	6	2	7	10	9	6	4	4	7	7.8008
8	6	6	4	6	2	7	8	9	6	5	4	8	7.9494
7	6	7	5	6	2	7	10	10	6	2	4	7	7.9935
9	7	8	4	6	2	7	10	9	6	5	4	7	8.1262
10	5	7	4	6	2	7	10	9	5	4	4	7	7.4457
10	5	6	4	6	2	7	10	9	6	3	3	7	7.4554
10	6	7	5	6	1	7	10	9	6	4	4	7	7.9968

References

1. Aggarwal, K., Singh, Y., Chandra, P., Puri, M.: Bayesian regularization in a neural network model to estimate lines of code using function points. Journal of Computer Science **1**(4), 505–509 (2005)
2. Ahmeda, M., Saliub, M., AlGhamdia, J.: Adaptive fuzzy logic-based framework for software development effort prediction. Information and Software Technology **47**(1), 31–48 (2005)
3. Akaike, H.: A new look at the statistical model identification. IEEE Transactions on Automatic Control **19**(6), 716–723 (1974)
4. Aramo-Immonen, H., Bikfalvi, A., Mancebo, N., Vanharanta, H.: Project managers competence identification. International Journal of Human Capital and Information Technology Professionals (IJHCITP) **2**(1), 33–47 (2011)
5. Arbib, M.: The Handbook of Brain Theory and Neural Networks. MIT Press (1995)
6. de Barcelos Tronto, I., da Silva, J., Sant'Anna, N.: An investigation of artificial neural networks based prediction systems in software project management. Journal of Systems and Software **81**(3), 356–367 (2008)
7. Barr, J., Saraceno, F.: A computational theory of the firm. Journal of Economic Behavior & Organization **49**(3), 345 – 361 (2002)
8. Bascil, M., Temurtas, F.: A study on hepatitis disease diagnosis using multilayer neural network with levenberg marquardt training algorithm. Journal of Medical Systems **35**, 433–436 (2011). 10.1007/s10916-009-9378-2
9. Bermejo, P., Joho, H., Joemon, M., Villa, R.: Comparison of feature construction methods for video relevance prediction. In: Advances in Multimedia Modeling. 15th International Multimedia Modeling Conference, MMM 2009, Sophia-Antipolis, France, January 7-9, 2009. Proceedings., pp. 185–196 (2009)
10. Bisagni, C., Lanzi, L., Ricci, S.: Optimization of helicopter subfloor components under crashworthiness requirements using neural networks. Journal of Aircraft **39**(2), 296–304 (2002)
11. Breiman, L., Spector, P.: Submodel selection and evaluation in regression. the x-random case. International Statistical Review **60**(3), 291–319 (1992)
12. Burger, M., Neubauer, A.: Analysis of tikhonov regularization for function approximation by neural networks. Neural Networks **16**(1), 79– 90 (2003)
13. Bygstad, B., Aanby, H.: Ict infrastructure for innovation: A case study of the enterprise service bus approach. Information Systems Frontiers **12**(3), 257–265 (2010)
14. Ramon-y Cajal, S.: The croonian lecture: La fine structure des centres nerveux. Proceedings of the Royal Society of London **55**, 444–468 (1894)
15. Cauchy, A.: Methodes generales pour la resolution des systemes d'equations simultanees. Compte Rendu des Séances de L'Académie des Sciences **25**(3), 536–538 (1847)
16. Cho, S.B., Shimohara, K.: Evolutionary learning of modular neural networks with genetic programming. Applied Intelligence **9**(3), 191–200 (1998)
17. Crawford, L.: Senior management perceptions of project management competence. International Journal of Project Management **23**(1), 7–16 (2005)
18. Crowther, P., Cox, R.: A method for optimal division of data sets for use in neural networks. In: Knowledge-Based Intelligent Information and Engineering Systems, 9th International Conference, KES 2005, Melbourne, Australia, September 14-16, 2005, Proceedings, Part IV, pp. 906–912 (2005)
19. Dietterich, T.: Approximate statistical tests for comparing supervised classification learning algorithms. Neural Computation **10**(7), 1895–1923 (1998)
20. Durand, N., Alliot, J.M., Médioni, F.: Neural nets trained by genetic algorithms for collision avoidance. Applied Intelligence **13**(3), 205–213 (2000)
21. El Emam, K., Koru, A.: A replicated survey of it software project failures. Software, IEEE **25**(5), 84–90 (2008)
22. Fahlman, S.: Faster-learning variations on back-propagation: an empirical study. In: Proceedins of Connectionist Models Summer School, pp. 38–51 (1988)
23. Feldt, R., Angelis, L., Torkar, R., Samuelsson, M.: Links between the personalities, views and attitudes of software engineers. Information and Software Technology **52**(6), 611–624 (2010)
24. Ferri, C., Hernández-Orallo, J., Modroiu, R.: An experimental comparison of performance measures for classification. Pattern Recogn. Lett. **30**(1), 27–38 (2009)
25. Galinec, D.: Human capital management process based on information technology models and governance. International Journal of Human Capital and Information Technology Professionals **1**(1), 44–60 (2010)
26. García-Crespo, A., González-Carrasco, I., Colomo-Palacios, R., López-Cuadrado, J., Ruiz-Mezcua, B.: Methodology for software development estimation optimization based on neural networks. IEEE Latin America Transactions **9**(3), 391–405 (2011)
27. Garcia-Crespo, A., Ruiz-Mezcua, B., Fernandez, D., Zaera, R.: Prediction of the response under impact of steel armours using a multilayer perceptron. Neural Computing & Applications **16**(2), 147–154 (2006)
28. Garg, A., Goyal, D., Lather, A.: The influence of the best practices of information system development on software smes: a research scope. International Journal of Business Information Systems **5**(3), 268–290 (2010)
29. Geraldi, J., Lee-Kelley, L., Kutsch, E.: The titanic sunk, so what? project manager response to unexpected events. International Journal of Project Management **28**(6), 547–558 (2010)
30. Gevrey, M., I., D., Lek, S.: Review and comparison of methods to study the contribution of variables in artificial neural network models. Ecological Modelling **160**(3), 249–264 (2003)
31. Gibbs, W.: Software's chronic crisis. Scientific American **271**(3), 72–81 (1994)
32. Girosi, F., Jones, M., Poggio, T.: Regularization theory and neural networks architectures. Neural Computation **7**(2), 219–269 (1995)
33. Glass, R.: The standish report: does it really describe a software crisis? Communications of the ACM **49**(8), 15–16 (2006)
34. Gomez, I., Martin, M.P.: Prototyping an artificial neural network for burned area mapping on a regional scale in mediterranean areas using modis images. International Journal of Applied Earth Observation and Geoinformation **13**(5), 741 – 752 (2011)
35. Gonzalez-Carrasco, I., Garcia-Crespo, A., Ruiz-Mezcua, B., Lopez-Cuadrado, J.: An optimization methodology for machine learning strategies and regression problems in ballistic impact scenarios. Applied Intelligence (in press) (2011)
36. Grochowski, M., Dutch, W.: Learning highly non-separable boolean functions using constructive feedforward neural network. In: ICANN'07 Proceedings of the 17th international conference on Artificial neural networks, pp. 180–189 (2007)
37. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice Hall PTR (1994)

38. He, Y., Sun, Y.: Neural network-based l1-norm optimisation approach for fault diagnosis of nonlinear circuits with tolerance. Circuits, Devices and Systems, IEE Proceedings - **148**(4), 223–228 (2001)
39. Henderson, C., Potter, W., McClendon, R., Hoogenboom, G.: Predicting aflatoxin contamination in peanuts: A genetic algorithm/neural network approach. Applied Intelligence **12**(13), 183–192 (2000)
40. Henry, J.: Software project management: a real-world guide to success. Addison Wesley Longman (2003)
41. Hestenes, R., Stiefel, E.: Methods of conjugate gradients for solving linear systems. Journal of Research of the National Bureau of Standards **49**(6), 409–436 (1952)
42. Hodgson, D.: Disciplining the professional: The case of project management. Journal of Management Studies **39**(6), 803–821 (2002)
43. Huang, M., Tsou, Y., Lee, S.: Integrating fuzzy data mining and fuzzy artificial neural networks for discovering implicit knowledge. Knowledge-Based Systems **19**(6), 396âĂŞ–403 (2006)
44. Jorgensen, M., Molokken-Ostvold, K.: How large are software cost overruns? a review of the 1994 chaos report. Information and Software Technology **48**(4), 297–301 (2006)
45. Kao, Y.T., Zahara, E.: A hybrid genetic algorithm and particle swarm optimization for multimodal functions. Applied Soft Computing **8**(2), 849– 857 (2008)
46. Kelemenis, A., Ergazakis, K., Askounis, D.: Support managers' selection using an extension of fuzzy topsis. Expert Systems with Applications **38**(3), 2774 – 2782 (2011)
47. Kumar, K.V., Ravi, V., Carr, M., Raj Kiran, N.: Software development cost estimation using wavelet neural networks. Journal of Systems and Software **81**(11), 1853–1867 (2008)
48. Kwak, Y.: A brief history of project management. In: The story of managing projects. Greenwood Publishing Group (2005)
49. Lanzi, L., Bisagni, C., Ricci, S.: Neural network systems to reproduce crash behavior of structural components. Computers structures **82**(1), 93–108 (2004)
50. Levenberg, K.: A method for the solution of certain nonlinear problems in least squares. Quarterly Journal of Applied Mathmatics **II**(2), 164–168 (1944)
51. Li, J., Wu, C., Wu, H.: Wavelet neural network process control technology in the application of aluminum electolysis. In: X. Wan (ed.) Electrical Power Systems and Computers, *Lecture Notes in Electrical Engineering*, vol. 99, pp. 937–941. Springer Berlin Heidelberg (2011)
52. Looney, C.: Advances in feedforward neural networks: demystifying knowledge acquiring black boxes. Transactions on Knowledge and Data Engineering **8**(2), 211–226 (1993)
53. Majumder, M., Roy, P., Mazumdar, A.: Optimization of the water use in the river damodar in west bengal in india: An integrated multi-reservoir system with the help of artificial neural network. Engineering Computing and Architecture **1**(2), 1–12 (2007). Article 1192
54. Marquardt, D.: An algorithm for least-squares estimation of nonlinear parameters. SIAM Journal on Applied Mathematics **11**(2), 431–441 (1963)
55. McConnell, S.: Professional software development. Addison-Wesley (2004)
56. Morris, P., Crawford, L., Hodgson, D., Shepherd, M., Thomas, J.: Exploring the role of formal bodies of knowledge in defining a profession-the case of project management. International Journal of Project Management **24**(8), 710–721 (2006)
57. Muller, R., Turner, J.: Matching the project manager's leadership style to project type. International Journal of Project Management **25**(1), 21–32 (2007)
58. Natarajan, T., Rajah, S., Manikavasagam, S.: Snapshot of personnel productivity assessment in indian it industry. International Journal of Information Technology Project Management **2**(1), 48–61 (2011)
59. Nelson, M., Illingworth, W.: A practical guide to neural nets. Reading, MA (USA); Addison-Wesley Publishing Co., Inc. (1991)
60. Onita, C., Dhaliwal, J.: Alignment within the corporate it unit: an analysis of software testing and development. European Journal of Information Systems **20**(1), 48–68 (2010)
61. O'Sullivan, D., Dooley, L.: Collaborative innovation for the management of information technology resources. International Journal of Human Capital and Information Technology Professionals **1**(1), 16–30 (2010)
62. Patnayakuni, R., Ruppel, C.: A socio-technical approach to improving the systems development process. Information Systems Frontiers **12**(2), 219–234 (2010)
63. Peng, X., zhe, H., Guifang, G., Gang, X., Binggang, C., Zengliang, L.: Driving and control of torque for direct-wheel-driven electric vehicle with motors in serial. Expert Systems with Applications **38**(1), 80 – 86 (2011)
64. Pereira, J., Cerpa, N., Verner, J., Rivas, M., Procaccino, J.: What do software practitioners really think about project success: A cross-cultural comparison. Journal of Systems and Software **81**(6), 897–907 (2008)
65. Priddy, K., Keller, P.: Artificial Neural Networks: An Introduction. SPIE Press (2005)
66. Principe, J., Euliano, N., Lefebvre, W.: Neural and Adaptive Systems: Fundamentals through Simulations with CD-ROM. John Wiley & Sons, Inc. (1999)
67. Principe, J., Lefebvre, C., Lynn, G., Fancourt, C., Wooten, D.: Neuro Solutions Documentation. NeuroDimension, Inc. (2007)
68. Rissanen, J.: Modeling by shortest data description. Automatica **14**(5), 445–471 (1978)
69. Roy, N., Potter, W., Landau, D.: Designing polymer blends using neural networks, genetic algorithms, and markov chains. Applied Intelligence **20**(3), 215–229 (2004)
70. Ruano-Mayoral, M., Colomo-Palacios, R., Garcia-Crespo, A., Gomez-Berbis, J.: Software project managers under the team software process: A study of competences. International Journal of Information Technology Project Management **1**(1), 42–53 (2010)
71. Shahhosseini, V., Sebt, M.: Competency-based selection and assignment of human resources to construction projects. Scientia Iranica **18**(2), 163 – 180 (2011)
72. Sokolova, M., Rasras, R., Skopin, D.: The artificial neural network based approach for mortality structure analysis. American Journal of Applied Science **3**(2), 1698–1702 (2006)
73. Songwu, L., Member, S., Basar, T.: Robust nonlinear system identification using neural network models. IEEE Transactions on Neural Networks **9**(3), 407–429 (1998)
74. Stamelos, I.: Software project management anti-patterns. Journal of Systems and Software **83**(1), 52–59 (2010)
75. Stamelos, I.: Software project management anti-patterns. Journal of Systems and Software **83**(1), 52–59 (2010)
76. Stavrou, E.T., Charalambous, C., Spiliotis, S.: Human resource management and performance: A neural network analysis. European Journal of Operational Research **181**(1), 453 – 467 (2007)
77. Su, Y., Huang, C.: Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models. Journal of Systems and Software **80**(4), 606–615 (2007)
78. Swingler, K.: Applying Neural Networks. A Practical Guide. Academic Press (1996)

79. Tang, F., Mu, J., MacLachlan, D.L.: Disseminative capacity, organizational structure and knowledge transfer. Expert Systems with Applications **37**(2), 1586 – 1593 (2010)
80. Tarassenko, L.: A guide to neural computing applications. Arnol / NCAF (1998)
81. Watanabe, J., Maruyama, T.: Software development industrialization by process-centered style. Fujitsu Scientific and Technical Journal **46**(2), 168–176 (2010)
82. Wong, T., Wong, S., Chin, K.: A neural network-based approach of quantifying relative importance among various determinants toward organizational innovation. Expert Systems with Applications **38**(10), 13,064 – 13,072 (2011)
83. Wythoff, B.: Backpropagation neural networks: A tutorial. Chemometrics and Intelligent Laboratory Systems **18**, 115–155 (1993)
84. Xie, C., Ling, Chang, J., Yuan, Shi, X., Cheng, Dai, J., Min: Fault diagnosis of nuclear power plant based on genetic-rbf neural network. Int. J. Comput. Appl. Technol. **39**(1/2/3), 159–165 (2010)

Ricardo Colomo-Palacios is an Associate Professor at the Computer Science Department of the Universidad Carlos III de Madrid. His research interests include applied research in Information Systems, Software Project Management, People in Software Projects and Social and Semantic Web. He received his PhD in Computer Science from the Universidad Politécnica of Madrid (2005). He also holds a MBA from the Instituto de Empresa (2002). He has been working as software engineer, project manager and software engineering consultant in several companies including Spanish IT leader INDRA. He is also an Editorial Board Member and Associate Editor for several international journals and conferences and Editor in Chief of International Journal of Human Capital and Information Technology Professionals.

Israel Gonzalez-Carrasco is an assistant professor in the Computer Science Department of Universidad Carlos III of Madrid. He holds his PhD degree in Computer Science by this University. He is co-author of several papers in lectures and congress and his main lines of research are Neural Networks, Expert Systems and Software Engineering. He is involved in several international projects and is a member of reviewer board and editorial advisory board of different journals.

Jose Luis Lopez-Cuadrado is assistant professor in Computer Science Department at the Universidad Carlos III of Madrid. He holds his PhD degree in Computer Science by this University. His research is focused on web based expert systems applications, neural networks, software engineering and processes improvement. Also he is co-author of several contributions published in international congresses.

Antonio Trigo is an Assistant Professor of Computer Science at Institute of Accounting and Administration of Coimbra, which is part of the Polytechnic Institute of Coimbra, Portugal, where he teaches business intelligence, management information systems, software engineering and computer programming, supervising several MSc students. He received his PhD in informatics from the University of Trás-os-Montes e Alto Douro. His research interests include information systems management and enterprise information systems. He worked as software engineer, project manager and software engineering consultant in several companies including Portugal Telecom, Cabo Verde Telecom, Meditel, Telesp Celular and Portuguese National Institute of Statistics. He has publications in international journals, book chapters and international conferences. He serves as editorial board member for international journals and has served in several organization and scientific committees of international conferences

Joao Eduardo Varajao is professor of information systems management and project management at the University of Tras-os-Montes e Alto Douro. He graduated in 1995, received his master degree in Computer Science in 1997 and, in 2003, received his PhD in Technologies and Information Systems, from University of Minho (Portugal). He supervises several Msc and PhD thesis in the information systems field. His current research includes information systems management and project management. He has over 200 publications, including books, book chapters, refereed publications, and communications at international conferences. He serves as associate editor and member of editorial board for international journals and has served in several committees of international conferences. He is founder of CENTERIS - Conference on ENTERprise Information Systems.