Forecasting enterprise resource planning software effort using evolutionary support vector machine inference model

Jui-Sheng Chou a,⁎, Min-Yuan Cheng a, Yu-Wei Wu a, Cheng-Chieh Wu b

⁎ Corresponding author at: 43 Sec. 4, Keelung Rd., Taipei 106, Taiwan.
Tel.: +886 2 2737 6321; fax: +886 2 2737 6606.
E-mail address: jschou@mail.ntust.edu.tw (J.-S. Chou).

Abstract

Despite significant advances in procedures that facilitate project management, the continued reliance of software managers on guesswork and subjective judgment causes frequent project time overruns. This study uses an Evolutionary Support Vector Machine Inference Model (ESIM) for efficiently and accurately estimating the person-hour of ERP system development projects. The proposed ESIM is a hybrid intelligence model integrating a support vector machine (SVM) with a fast messy genetic algorithm (fmGA). The SVM mainly provides learning and curve fitting while the fmGA minimizes errors. The analytical results in this study confirm that, compared to artificial neural networks and SVM, the proposed ESIM provides preliminary prediction at early phase of ERP software development effort for the manufacturing firms with superior accuracy, shorter training time and less overfitting. Future research can develop user-friendly expert systems with window or browser interfaces that can be used by planning personnel to flexibly input related variables and to estimate development effort and corresponding project time/cost. © 2012 Elsevier Ltd. APM and IPMA. All rights reserved.

Keywords: Enterprise resource planning; Software effort prediction; Project management; Hybrid intelligence

1. Introduction

Enterprise resource planning (ERP) enables developers to enhance the global competitiveness and sustainability of their client enterprises by ensuring efficient resource allocation. In practice, ERP software functions and specifications are highly unpredictable at early stages of R&D; thus, initial cost estimation relies mostly on the subjective judgments of experienced software engineers. Although knowledgeable sales managers or estimators may generate accurate cost assessments via a cooperative approach, professionals in small and medium-sized software enterprises are often difficult to train and highly mobile. Thus, the difficulty of retaining experienced personnel with project knowledge results in such problems as loss of project know-how.

Despite the significant advances in the procedures that facilitate project management (PMI, 2008), product managers in the software industry still encounter problems requiring guesswork and subjective judgment, which often result in inaccurate estimates. Effort estimation is not functionally related to the basic drivers of ERP system development. Companies can thus lose market share and orders when attempting to attract customers during the early marketing phase. Although human experts can achieve satisfactory outcomes, shortfalls typically result from inefficient information management. Shortcomings in current subjective assessments or analogous methods indicate the urgent need and opportunity for improvement.

Unlike traditional manufacturing, most software product development costs are incurred by investment in human resources. As software is a virtual intelligence and customer service–oriented product, software developers must estimate project completion time at early stages. Many studies (Ahmed and Muzaffar, 2009; de Barcelos Tronto et al., 2008; Elish, 2009; Finnie et al., 1997; Huang and Chiu, 2006; Huang et al., 2008; Jørgensen, 2010; Kazemifard et al., 2011; Lopez-Martín, 2011; Mair et al., 2000; Oliveira et al., 2010; van Koten and Gray, 2006) have proposed cost or effort forecasting methods for software development...
The typical phases of a software development project (Yeo, 2011) include quality management, and human resource management. Functions include investment in human resources. Nevertheless, the effectiveness and efficiency of approximate inference methods for estimating ERP system development efforts via hybrid intelligence (i.e., combination of multiple artificial intelligence techniques) are rarely addressed.

Developing deterministic mathematical models for solving project prediction problems is both difficult and expensive. Approximate inference, a fast and cost effective approach, is a viable alternative to deterministic mathematical modeling. Inference is the process of deriving new knowledge from known information. As the known information changes, the inference process adapts accordingly. Prediction problems are complex and often involve substantial uncertainty, vagueness, and incomplete or inexact data. Therefore, the inference process must fit environmental conditions (Mareels and Polderman, 1996). Humans can process and solve complex problems, even those involving uncertainty, imprecision, and incomplete information. Therefore, imitating the process of human inference is an effective approach to solving project prediction problems.

The primary objective of this research was predicting development time for ERP software project by using an Evolutionary Support Vector Machine Inference Model (ESIM). After reviewing the relevant literature and collecting data and ERP software specifications from past projects in the manufacturing industry, an ESIM was proposed for simultaneously searching for the fittest support vector machine (SVM) parameters within a globally optimized model. An early accurate time estimate during the negotiation stage with potential clients can eliminate unnecessary bargaining and cost changes in subsequent processes.

The rest of this paper is organized as follows. Section 2 reviews pertinent literature in software project estimation. Section 3 then discusses the research methodology and model adaptation process. Next, Section 4 illustrates the applicability of the ESIM technique in a case study of a leading ERP software system provider. Conclusions are finally drawn in Section 5 along with managerial implications and suggestions for future research.

2. Literature review

The ERP software systems typically manage internal enterprise resources and departmental business operations. Functions include business management, production management, financial planning, quality management, and human resource management. The typical phases of a software development project (Yeo, 2001) are, in chronological order, system analysis, system design, programming, testing, deployment, and online operations.

Estimation accuracy heavily depends on the amount and quality of information available at estimation time. Thus, providing accurate preliminary project estimates is extremely challenging, particular during the software development phases (Lopez-Martin, 2011). A major limitation of current practices is that, when contacting clients for make-to-order quotes, project managers must estimate time-to-complete based solely on known product attributes whereas virtually all software costs at early stages of software development are implicitly determined by investment in human resources.

High-quality ERP software development projects require efficient project control mechanisms to schedule and manage software development progress. Project estimation is thus a prerequisite for all subsequent planning activities. Although none of the studies have quantified the effort or duration requirements for ERP software development, project estimation schemes proposed for other software project and industries include case-based reasoning (Belecheanu et al., 2003; Chou, 2009b; Finnie et al., 1997; Mendes et al., 2002, 2003; Yang and Wang, 2008), multiple regression analysis (Adriano L.I, 2006; Bashir and Thomson, 2001, 2004; M. Camargo et al., 2003; Caputo and Pelagagge, 2008; Chou, 2009a; Chou et al., 2006; Ferens, 1998; Finnie et al., 1997; Huang et al., 2008; Marban et al., 2008; Mittas and Angelis, 2010; Sentas and Angelis, 2006), activity-based costing (Baykasoglu and Kaplanoglu, 2008; Ben-Arie and Qian, 2003; Tornberg et al., 2002), artificial neural networks (Attarzadeh and Ow, 2010; Berlin et al., 2009; Caputo and Pelagagge, 2008; de Barcelos Tronto et al., 2008; Delen et al., 2005; Finnie et al., 1997; Huawang and Wanying, 2008), support vector machine (Adriano L.I, 2006; An et al., 2007; Lu and Tsai, 2008; Pal and Deswal, 2008), genetic algorithms (Huang and Chiu, 2006; Oliva et al., 2010) and function point analysis (Albrecht and Gaffney, 1983; Finnie et al., 1997; IFPUG, 2009; Longstreet, 2002; Mendes et al., 2003; Myrveit et al., 2005; Yin-huan et al., 2009). However, the above works considered single techniques rather than hybrid intelligence schemes.

Artificial intelligence (AI)-based approaches are related to computer system designs that attempt to resolve problems intelligently by emulating human brain processes. As AI technology enhances the ability of computer programs to handle tasks for which humans are still superior (Haykin, 1999), AI models are typically used to solve project estimation problems. Various scientific and engineering fields have recently combined different AI paradigms to enhance efficacy. Numerous studies confirm that hybrid AI schemes outperform single techniques in project estimation (Chen, 2007; Kim and Shin, 2007; Lee, 2009; Li et al., 2005; Min et al., 2006; Nandi et al., 2004; Wu, 2010; Wu et al., 2009). Fast messy genetic algorithm (fmGA) (Goldberg et al., 1993) and the SVM (Vapnik, 1995) are two such tools that have proven effective for solving various project management problems.

Specifically, the SVM in the ESIM is used mainly for leaning and curve fitting while the fmGA optimizes prediction error. Given the characteristics and merits of fmGA and SVM, the ESIM proposed in this study combines them for predicting development effort in ERP software projects. Fig. 1 shows the software project quotation and engineer-to-order processes integrated with the proposed ESIM. This ESIM was designed to achieve the fittest C and gamma parameters with minimal prediction error. The proposed approach considers implicit knowledge from historical cases in the software industry so that project managers or decision-makers can overcome challenges during early cost quotation.

3. Evolutionary support vector machine inference model

3.1. Support vector machine integrated with fast messy genetic algorithm

Support vector machine was first introduced by Vapnik (1995) and colleagues at AT&T Bell Laboratories (Vapnik,
The numerous applications of SVMs in various other disciplines include customer churn prediction (Coussement and Van den Poel, 2008), reliability forecasting in engine systems (Chen, 2007), document categorization (Hao et al., 2007), performance rating (Lee, 2007; Ravi et al., 2008), content-based image retrieval (Seo, 2007), Fault detection and diagnosis (Konar and Chattopadhyay, 2011; Widodo and Yang, 2007), prediction of air entrainment rate and aeration efficiency of weirs (Baylar et al., 2009), and bankruptcy prediction (Chaudhuri and De, 2011; Min et al., 2006; Shin et al., 2005). The SVM classifies data with different class labels by determining a set of support vectors that are members of the set of training inputs that outline a hyper plane in a feature space. A kernel function of a generic mechanism is used to fit the hyper plane surface to training data. The input-output model in this study was constructed by Epsilon Support Vector Regression (ε-SVR) (Smola and Schölkopf, 2004), a variation of generic SVM for function estimation. In SVM regression, the input \( x \) is first mapped onto an \( m \)-dimensional feature space by using fixed (nonlinear) mapping. A linear model is then constructed in this feature space (Fig. 1). The linear model (in the feature space) \( f(x, \omega) \) can be expressed mathematically as

\[
f(x, \omega) = \sum_{j=1}^{m} \omega_{j} g_{j}(x) + b
\]

where \( g_{j}(x), j=1, \ldots, m \) denotes a set of nonlinear transformations, and \( b \) is the “bias” term. Because data are typically assumed to have a mean of zero (and are therefore obtainable at the pre-processing stage), the bias term is dropped. Estimation quality is measured by the loss function \( L(y, f(x, \omega)) \). The SVM regression uses the following \( \varepsilon \)-insensitive loss function, which was proposed by Vapnik (1998):

\[
L_{\varepsilon}(y, f(x, \omega)) = \begin{cases} 0 & \text{if} |y - f(x, \omega)| \leq \varepsilon \\ |y - f(x, \omega)| - \varepsilon & \text{otherwise} \end{cases}
\]

Linear regression performed in SVM regression in the high-dimension feature space uses \( \varepsilon \)-insensitive loss and reduces model complexity by minimizing \( ||\omega||^{2} \). This procedure is implemented when (non-negative) slack variables, \( \xi_{i}, \xi_{i}^{*}, j=1, \ldots, n \), are introduced to identify training samples that deviate from the
ε-insensitive zone. Thus, SVM regression is formulated as a minimization of the following function:

$$\min \frac{1}{2} \|\omega\|^2 + C \sum_{i=1}^{n} (\xi_i + \xi_i^*)$$

subject to

$$y_i - f(x_i, \omega) \leq \varepsilon + \xi_i$$
$$f(x_i, \omega) - y_i \leq \varepsilon + \xi_i^*$$
$$\xi_i, \xi_i^* \geq 0, i = 1,...,n$$

This optimization problem can be transformed into the dual problem, which is solved by

$$f(x) = \sum_{i=1}^{n_{SV}} (\alpha_i - \alpha_i^*) K(x_i, x)$$

subject to

$$0 \leq \alpha_i^* \leq C, 0 \leq \alpha_i \leq C$$

where $n_{SV}$ is the number of support vectors, and the kernel function is

$$K(x, x_i) = \sum_{j=1}^{m} g_j(x)g_j(x_i)$$

SVM generalization performance (estimation accuracy) requires correct settings for meta-parameters $C$ and $\gamma$, and for $\varepsilon$ kernel parameters (that are related to the kernel type, e.g., $\gamma$ parameter in RBF kernel $K(x, y) = \exp(-\gamma \|x-y\|^2)$; default values of kernel parameters clearly depend on the kernel parameter type and the implementing software. Existing software implementations of SVM regression usually treat meta-parameters as user-defined inputs (Smola and Schölkopf, 2004; Vapnik, 1995). Selecting specific kernel types and kernel function parameters usually requires application-domain knowledge and should reflect the distribution of training data inputs.

A user may select the SVM kernel function (e.g., linear, radial basis, polynomial, or sigmoid function) during the training process to identify support vectors along the function surface. Since the RBF kernel maps samples nonlinearly into a higher dimensional space and presents fewer numerical difficulties, RBF is a reasonable first choice. However, when using an SVM, users must determine the optimal kernel parameters. Therefore, SVM parameters must be obtained when optimizing parameter settings. For enhanced SVM prediction accuracy, parameter optimization should include penalty parameter $C$, and the gamma of the radial basis function (RBF) kernel. Fig. 2 shows the generic SVM procedure.

Fast messy genetic algorithm (fmGA), which was first developed by Goldberg et al. (1993), is an alternative means of identifying the best $C$ and gamma parameters when using the RBF kernel function. In contrast with the conventionally adopted simple GA (sGA), which uses fixed length strings to represent possible solutions, an fmGA applies messy chromosomes to form strings of various lengths. Efficient solution optimization for large-scale permutation problems enables fmGA to generate SVM parameters $C$ and gamma simultaneously.

The proposed ESIM uses SVM for learning and curve fitting while the fmGA optimizes prediction error. The purpose of the
model was to obtain the fittest C and gamma parameters with minimized prediction error. The ESIM adaptation process can be demonstrated with a pseudo code algorithm (Section 3.2) developed by the authors for efficient project prediction. Fig. 3 shows the ESIM flowchart. The steps are as follows. (1) Training the SVM. In this step, the SVM uses default parameters and a training dataset to train a prediction model. (2) Fitness evaluation. For each chromosome representing C and γ, a training dataset is used to train the SVM and to calculate prediction accuracy. When accuracy is sufficient, each chromosome is evaluated using a fitness function. (3) Termination criteria. The process stops once termination criteria are satisfied. If any criteria are unsatisfied, the model proceeds to the next generation. (4) The search for fmGA parameters. In this stage, the model searches for superior solutions via genetic operations.

3.2. ESIM adaptation process

The ESIM adaptation process can be programmed by the following pseudo code algorithm.

Begin
Epoch = 1;
Generate the competitive template = random string;
While (not termination condition) //Outer Loop
    { //Inner Loop
        //Initialization Phase
        Era = 0;
        Probabilistic_Initialize(Pop(Era), Epoch);
        Evaluate(Pop(Era), template); //Evaluate fitness value
        //Primordial Phase
        While (not primordial termination condition)
            { Episode = 0;
              While (Episode < Episode_max(Era))
                { Thresholding_Selection(Pop(Era));
                  Episode = Episode + 1;
                }
            Building-Blocks_Filtering(Pop(Era));
            }
    }
End

3.2.1. Probabilistic.Initialize
This step simulates a natural chromosome by randomly assigning 0s and 1s to gene values to produce random variables, including C and γ.

3.2.2. Evaluate
The objective function of the model (f^ob) combines model accuracy and model complexity. The fitness function is the reciprocal of the objective function.

3.2.3. Thresholding_Selection
To restrict competition between highly dissimilar building blocks, a generic thresholding mechanism was used to limit tournament selection to pairs of strings that shared a greater than expected number of common genes. In random strings of two different lengths, \( \lambda_1, \lambda_2 \), the expected number of common genes is \( \theta = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2} + C'(\alpha') \sigma \), where \( \sigma \) is the standard deviation in the number of genes that two randomly chosen strings of possibly differing lengths have in common, and parameter \( C'(\alpha') \) is simply the ordinate of a one-sided normal distribution with tail probability \( \alpha \).
3.2.4. Building-Blocks_Filtering

A building block repetition factor $\gamma$ acquires the desired building block, where $\gamma = \left[ \frac{\lambda_{i-1}}{\lambda_{j}} \right]_{\lambda_{i-1} - k}^{\lambda_{j} - k}$.

3.2.5. Cut_and_Splice

In this step, a cut probability recombinates different strings into new strings.

3.2.6. Mutation

The mutation produces spontaneous random changes in various chromosomes to protect against premature loss of important notations. For the ESIM, the purpose of mutation is to adjust the value of $C$ and $\gamma$ for better performance. It alters one or more genes with a probability ($p^{\text{mu}}$), which is smaller than or equal to the mutation rate ($p^{\text{mut}}$). Mutation operation incrementally compares the $p^{\text{mu}}$ of the gene with $p^{\text{mut}}$. If $p^{\text{mu}} \leq p^{\text{mut}}$, then the gene value is altered.

4. Project description and analytical results

The ERP software development projects are funded mainly by direct investment in human resources. Therefore, if the number of person-hours can be estimated accurately, then the software development costs can be determined rationally by multiplying the known corresponding hourly rate of engineers required during the preliminary phase. This study collected and analyzed empirical data for 182 ERP software projects developed by a leading Taiwan software provider over the last five years for the manufacturing units.

4.1. Variable selection and definition

4.1.1. Output variables

In practice, ERP software development projects normally estimate costs based on the devoted person-hour of effort. Project development costs are obtained by multiplying the development time by the hourly rate. The output variable in this investigation referred to software development effort (SDE) in order to accurately represent in-house project costs and to prevent distortions due to inconsistent project quotes or discounts from external sources.

4.1.2. Input variables

Based on interviews with experienced in-house project managers, this work identified factors that affect ERP SDE. The attributes were then designated as input variables and defined as follows:

(a) Program Type_Entry (PTE) allows business users to compile basic information for future business transactions and applications.
(b) Program Type_Report (PTR) extracts data from the ERP system database and generates charts with specific formats to verify transactions or to analyze management practices.
(c) Program Type_Batch (PTB) processes the information stored in the ERP system database.
(d) Program Type_Query (PTQ) extracts information stored in the ERP system database and compiles them appropriately for presentation on the system interface.
(e) Program Type_Transaction (PTT) manages the creation of various internal transaction forms in an enterprise.
(f) Number of Programs (NP) indicates the number of scheduled programs in the development project.
(g) Number of Zoom (NZ) is the source data in the operating interface column.
(h) Number of Columns in Form (NCF) refers to the number of columns in the operating interface.
(i) Number of Actions (NA) refers to specific keys typically arranged to correspond to functions specified by the program.
(j) Number of Multi-angle Trade Tasks (NMTT) refers to the trade model that a business uses to accept orders from overseas clients, to purchase supplies from a third country, and to either deliver the goods to the overseas clients directly from third country suppliers or to transport the goods to the client through Taiwan.
(k) Number of Multi-unit Tasks (NMT) requires the use of multiple measurement units, including inventory, purchasing, sales, costs, and products.
(l) Number of Reference Calls (NRC) calls for an external system for independent tasks.
(m) Number of Confirmed Tasks (NCT) refers to data generated through file and invoice creation programs which are officially adopted through confirmation procedures.
(n) Number of Post Tasks (NPT) calculates the inventory quantity and control functions for incoming and outgoing goods and materials for daily enterprise operations.
(o) Number of Batch Serial Numbers (NBSN) is used for customer complaints about product quality, and batch serial numbers are used to monitor these problems so effective solutions can be found.
(p) Number of Signature Tasks (NST) refers to the use of a signature function in workflow software programs so that important tasks can be verified by administrators.
(q) Number of Industry Type Tasks (NITT) indicates the number of different companies from the same enterprise, which may use various business models and may require specialized system designs.

4.2. Validation and descriptive statistics

Model validation usually involves checking the model against observed or independent data (Neter et al., 1996). The predictive ability of the ESIM model was checked through the hold-out data extracted from the original ERP projects. The purpose of test data is to examine whether the model developed from the earlier data is still applicable for other projects. Since it’s a common approach to randomly split data into 90–10 or 80–20% (Witten and Frank,
2005), historical cases were randomly separated into training sets (163 projects, 90%) and test sets (19 projects, 10%).

Table 1 summarizes the descriptive statistical data obtained by randomly filtering 90% of collected data for 163 historical projects. The minimum SDE was 2,694 (hours), and the standard deviation was 394.690 (hours). The statistical results indicated that the size and complexity of each development project depend on the features and functions of programs they incorporate. The minimum value for the number of programs is 1, the maximum value is 88, and the standard deviation is 19.122. The empirical data indicate that the ERP SDE is proportional to the NP in a project, and NZ, NCF, and NA vary according to the functions of each program.

Meanwhile, NST, NBSN, NMTT, NMT, NCT, NPT, and NITT are related to specific function areas. Restated, not all programs in each project are equipped with this functional requirement. Therefore, the average values in Table 1 are relatively low, ranging from 0.23 to 1.5. However, as the number of specified features increases, ERP SDE significantly increases.

### 4.3. Model performance and prediction accuracy

The following performance measures were used to evaluate the proposed estimating techniques.

#### 4.3.1. Root mean squared error

Root mean squared error (RMSE) is the square root of the mean square error. The RMSE is thus the average distance of a data point from the fitted line measured along a vertical line. The RMSE is given by the following equation:

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

where $y_i$ is actual value, $\hat{y}_i$ is the predicted value, and $n$ is the number of data samples.

#### 4.3.2. Mean magnitude of relative error

Mean magnitude of relative error (MMRE) is a statistical measure of predictive accuracy. It usually expresses accuracy as a percentage. The MMRE is commonly used in quantitative forecasting methods because it indicates relative overall fit (goodness-of-fit). The MMRE is given by the following equation:

$$\text{MMRE} = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$

#### 4.4. Cross-fold validation

When comparing the predictive accuracy of two or more methods, researchers often use k-fold cross-validation to minimize bias associated with the random sampling of the training and holdout data samples. Specifically, the cross-fold technique can be employed to examine the internal model validity while the comparison with other techniques can assess the external validity. Thus, this work further includes conventional artificial neural networks (ANNs) and support vector regression (SVR) as baseline models. Since cross-validation requires random assignment of individual cases into distinct folds, a common practice is stratifying the folds themselves. In stratified k-fold cross-validation, the proportions of predictor labels (responses) in the folds are intended to approximate those in the original dataset. Empirical studies show that, compared to regular k-fold cross-validation, stratified cross-validation tends to reduce bias in the comparison results.

Kohavi (1995) showed that ten folds were optimal (i.e., performed validation testing in the shortest time and with

---

**Table 1** Variables and descriptive statistics for ERP software project effort prediction.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Min.</th>
<th>Max.</th>
<th>Mean</th>
<th>Standard deviation</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDE (person-hour)</td>
<td>4</td>
<td>2694</td>
<td>258.55</td>
<td>394.690</td>
<td>Numerical</td>
</tr>
<tr>
<td>PTE</td>
<td>0</td>
<td>1</td>
<td>Dummy variable</td>
<td>Boolean</td>
<td></td>
</tr>
<tr>
<td>PTR</td>
<td>0</td>
<td>1</td>
<td>Dummy variable</td>
<td>Boolean</td>
<td></td>
</tr>
<tr>
<td>PTB</td>
<td>0</td>
<td>1</td>
<td>Dummy variable</td>
<td>Boolean</td>
<td></td>
</tr>
<tr>
<td>PTQ</td>
<td>0</td>
<td>1</td>
<td>Dummy variable</td>
<td>Boolean</td>
<td></td>
</tr>
<tr>
<td>PTT</td>
<td>0</td>
<td>0</td>
<td>Referential category</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>NP</td>
<td>1</td>
<td>88</td>
<td>16.73</td>
<td>19.122</td>
<td>Numerical</td>
</tr>
<tr>
<td>NZ</td>
<td>0</td>
<td>2028</td>
<td>100.22</td>
<td>255.395</td>
<td>Numerical</td>
</tr>
<tr>
<td>NCF</td>
<td>3</td>
<td>3216</td>
<td>397.75</td>
<td>548.057</td>
<td>Numerical</td>
</tr>
<tr>
<td>NA</td>
<td>0</td>
<td>1645</td>
<td>288.44</td>
<td>339.614</td>
<td>Numerical</td>
</tr>
<tr>
<td>NST</td>
<td>0</td>
<td>15</td>
<td>.39</td>
<td>1.772</td>
<td>Numerical</td>
</tr>
<tr>
<td>NBSN</td>
<td>0</td>
<td>11</td>
<td>.31</td>
<td>1.497</td>
<td>Numerical</td>
</tr>
<tr>
<td>NMTT</td>
<td>0</td>
<td>22</td>
<td>.55</td>
<td>2.662</td>
<td>Numerical</td>
</tr>
<tr>
<td>NMT</td>
<td>0</td>
<td>21</td>
<td>1.10</td>
<td>3.409</td>
<td>Numerical</td>
</tr>
<tr>
<td>NRC</td>
<td>0</td>
<td>528</td>
<td>13.96</td>
<td>49.923</td>
<td>Numerical</td>
</tr>
<tr>
<td>NCT</td>
<td>0</td>
<td>21</td>
<td>1.50</td>
<td>3.990</td>
<td>Numerical</td>
</tr>
<tr>
<td>NPT</td>
<td>0</td>
<td>12</td>
<td>.23</td>
<td>1.330</td>
<td>Numerical</td>
</tr>
<tr>
<td>NITT</td>
<td>0</td>
<td>21</td>
<td>.80</td>
<td>2.967</td>
<td>Numerical</td>
</tr>
</tbody>
</table>

---

**Table 2** Cross-fold validation.

<table>
<thead>
<tr>
<th>Measure</th>
<th>Fold Model</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>Average</th>
<th>New case prediction</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMRE</td>
<td>ESIM</td>
<td>242.86%</td>
<td>42.92%</td>
<td>61.18%</td>
<td>46.09%</td>
<td>43.53%</td>
<td>44.33%</td>
<td>27.05%</td>
<td>34.80%</td>
<td>31.66%</td>
<td>27.72%</td>
<td>60.21%</td>
<td>26.16%</td>
</tr>
<tr>
<td>SRV</td>
<td>1295.88%</td>
<td>380.98%</td>
<td>711.85%</td>
<td>425.50%</td>
<td>283.47%</td>
<td>360.01%</td>
<td>73.70%</td>
<td>478.62%</td>
<td>141.06%</td>
<td>205.85%</td>
<td>435.69%</td>
<td>177.63%</td>
<td></td>
</tr>
<tr>
<td>PTR</td>
<td>248.06%</td>
<td>47.35%</td>
<td>113.40%</td>
<td>56.73%</td>
<td>45.47%</td>
<td>52.79%</td>
<td>20.59%</td>
<td>68.29%</td>
<td>26.14%</td>
<td>36.94%</td>
<td>71.58%</td>
<td>29.09%</td>
<td></td>
</tr>
<tr>
<td>SVR</td>
<td>49.07410</td>
<td>82.70228</td>
<td>56.85330</td>
<td>39.67703</td>
<td>43.81111</td>
<td>43.63777</td>
<td>44.51864</td>
<td>43.28778</td>
<td>60.43048</td>
<td>36.01950</td>
<td>50.00080</td>
<td>135.39524</td>
<td></td>
</tr>
</tbody>
</table>
acceptable bias and variance) (Kohavi, 1995). Thus, to assess estimation performance across artificial intelligence techniques, a stratified 10-fold cross-validation approach was used. The entire dataset was divided into ten mutually exclusive subsets (or folds) with class distributions approximating those of the original dataset (stratified). The subsets were extracted in five steps:

1. Randomize the dataset
2. Extract one tenth of the original dataset size from the randomized dataset (single fold).
3. Remove the extracted data from the original dataset.
4. Repeat steps 1-3 eight times.
5. Assign the remaining portion of the dataset to the last fold (10th fold).

This procedure was first used to obtain ten distinct folds. Each fold was used once for performance tests of the prediction models, and the remaining nine folds were used for training, which obtained ten independent performance estimates. The cross-validation estimate of overall accuracy was calculated by simply averaging the $k$ individual accuracy measures for cross-validation accuracy $CVA$, where $k$ is the number of folds used and $A_i$ is the accuracy measure of each fold.

$$CVA = \frac{1}{k} \sum_{i=1}^{k} A_i$$  \hspace{1cm} (8)

Based on the cross-fold technique, the experiment compared prediction performance in the three regression models, i.e., ESIM, SVR, and ANNs. Table 2 shows $MMRE$ and $RMSE$ for the three models. The comparison results show that ESIM generally outperformed SVR and ANNs in terms of average $MMRE$. Interestingly, ESIM and ANNs have similar RMSE rates which are both better than SVR. Notably, to test the prediction power, this study collects five more newly completed project cases. The prediction performance of ESIM is also better than ANNs and SVR in terms of $MMRE$ and $RMSE$ (Table 2).

4.5. Model performance

The previous internal and external validity assessments have proven the superiority of ESIM and ANNs in ERP project effort prediction. This study further randomly split the whole case dataset into training (90%) and testing (10%) groups for evaluating these two models’ performance. Fig. 4 compares the ERP project training results for the person-hour predicted by the ESIM with those for the actual effort. Although the range of these individual prediction errors (PEs) was 0.00–177.00%, roughly 60% of the training cases had PEs lower than 20%, and about 40% had PEs lower than 10%. The overall $MMRE$ of the ESIM training dataset was 26.8%, and the $RMSE$ was 234.0157 (hours).

Model validity can be verified by the data in Fig. 5, which shows the difference between the predicted and actual development effort for the test dataset; 58% of the test cases had PEs lower than 25.4%. The overall $MMRE$ of the ESIM testing set was 27.3% with a smaller dispersion of $RMSE$ (97.2667 hours). Based on the randomly split training and test datasets, the analytical results confirm the efficacy of ESIM for preliminary prediction. Table 3 lists the specified $C$ constant and gamma value required as input patterns for the ESIM case study.

This investigation adopted the same set of historical project data used in ESIM to construct a general back-propagation ANNs model for comparison purposes. This work used the PASW modeler 13.0 data mining tool (formerly Clementine) for the analyses. The network-related parameters included

Table 3

<table>
<thead>
<tr>
<th>Table 3</th>
<th>ESIM Results.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C$, $\gamma$</td>
<td>110, 0.9995</td>
</tr>
<tr>
<td>RMSE (Training)</td>
<td>234.0157 hr</td>
</tr>
<tr>
<td>MMRRE (Training)</td>
<td>26.8%</td>
</tr>
<tr>
<td>RMSE (Testing)</td>
<td>97.2667 hr</td>
</tr>
<tr>
<td>MMRRE (Testing)</td>
<td>27.3%</td>
</tr>
<tr>
<td>No. of training cases</td>
<td>163</td>
</tr>
<tr>
<td>No. of testing cases</td>
<td>19</td>
</tr>
</tbody>
</table>
input ranges, number of neurons in the input layer, number of neurons in the hidden layers, number of input neural networks, learning rate-related initial value, decay rate, maximum value, minimum value, and inertia factor (alpha).

Training was curtailed when either of the following conditions was met: (a) accuracy higher than the designated value; (b) training cycle; (c) training time. The ANNs simulation was set to 2–9 to avoid model overfitting. Since empirical experience indicated that low learning rates obtain highly precise solutions, the initial value was set to 0.05. The error value may be nonlinear, possibly resulting in the classification of a solution obtained by steepest descent method as the local optimal solution. To avoid local optimal solutions during training, a higher inertia factor of 0.95 is suggested (SPSS, 2009).

Table 4 shows the project person-hour predicted by the model for individual test cases along with their actual output. Although the ANNs clearly generated less prediction dispersion, the ESIM was more accurate and had a narrower range of prediction error for the ERP software project development.

5. Conclusions and future directions

This study demonstrated an ESIM approach for predicting development time for ERP software projects. The ESIM is a hybrid intelligence technique integrating an SVM with an fmGA. The SVM provides learning and curve fitting by mapping non-linear input and output, and the fmGA primarily deals with global optimization concurrently while minimizing prediction error. According to the empirical case study, the proposed systematic approach efficiently predicts the development effort of ERP software project at early stages as prediction error is within a satisfactory limit.

Through the cross-validation and prediction power testing, ESIM and ANNs show better performance than conventional SVR in ERP project effort prediction. Further, analytical results via random sampling process indicate that the overall MMRE of ESIM is 26.8% for training datasets. For the randomly separated data for testing, the MMRE achieved by ESIM was 27.3%, which is lower than the 36.5% by ANNs (Table 3) with a 25.2% improvement in prediction accuracy. Particularly, the current in-house analogous practice is 35.0% in average indicating an improvement of 22.0% via ESIM over the MMRE of the software provider. Notably, the effort required to complete software development projects also depends on collaboration among multiple engineers, their experience level, and location. The historical project data considered by the ESIM did not include such information and could not be covered in this research. Therefore, to further improve the ESIM performance, future studies are needed to gather comprehensive project data.

Software companies typically have high employee turnover. Therefore, they may often have difficulty retaining experience-based and implicit corporate knowledge. Conversely, only incomplete design specifications are available during the early quotation and negotiation stages. Rapidly responding to customer enquiries using little available information is critical to securing orders. Fairly accurate tools that can be used to predict the manpower requirements for developing an ERP system are needed so that engineered-to-order suppliers can maintain competitive advantage in global markets.

With experience, good judgment, and timely adjustments, systematic knowledge discovery from sales records offers project managers in the software industry an effective and efficient way of estimating project effort and quoting prices in the early stages of software development for customers. Future research can develop user-friendly expert systems with window or browser interfaces that can be used by planning personnel to flexibly input related variables and to estimate development effort. Such a system can be designed to forecast completion date based on project size and participating project resources.

References


Table 4
Model performance.

<table>
<thead>
<tr>
<th>Case No.</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual Output (hr)</td>
<td>688</td>
<td>58</td>
<td>351</td>
<td>58</td>
<td>364</td>
<td>658</td>
<td>53</td>
<td>143</td>
<td>12</td>
<td>862</td>
<td>317</td>
<td>410</td>
<td>68</td>
<td>383</td>
<td>168</td>
<td>65</td>
<td>44</td>
<td>12</td>
<td>156</td>
</tr>
<tr>
<td>ANNs (hr)</td>
<td>571</td>
<td>90</td>
<td>196</td>
<td>53</td>
<td>164</td>
<td>419</td>
<td>20</td>
<td>241</td>
<td>17</td>
<td>746</td>
<td>186</td>
<td>491</td>
<td>58</td>
<td>582</td>
<td>198</td>
<td>54</td>
<td>89</td>
<td>7</td>
<td>259</td>
</tr>
<tr>
<td>PE (%)</td>
<td>17</td>
<td>56</td>
<td>44</td>
<td>9</td>
<td>55</td>
<td>36</td>
<td>61</td>
<td>69</td>
<td>45</td>
<td>13</td>
<td>41</td>
<td>20</td>
<td>15</td>
<td>52</td>
<td>18</td>
<td>17</td>
<td>102</td>
<td>42</td>
<td>66</td>
</tr>
<tr>
<td>ESIM (hr)</td>
<td>600</td>
<td>51</td>
<td>119</td>
<td>43</td>
<td>214</td>
<td>399</td>
<td>66</td>
<td>82</td>
<td>13</td>
<td>861</td>
<td>257</td>
<td>332</td>
<td>57</td>
<td>482</td>
<td>178</td>
<td>68</td>
<td>71</td>
<td>18</td>
<td>222</td>
</tr>
<tr>
<td>PE (%)</td>
<td>12.7</td>
<td>11.6</td>
<td>66.0</td>
<td>25.4</td>
<td>41.1</td>
<td>39.4</td>
<td>24.4</td>
<td>42.4</td>
<td>9.4</td>
<td>0.1</td>
<td>18.8</td>
<td>19.0</td>
<td>16.8</td>
<td>25.8</td>
<td>6.0</td>
<td>5.2</td>
<td>61.0</td>
<td>51.2</td>
<td>42.5</td>
</tr>
</tbody>
</table>

1. Prediction error.


Jørgensen, M., 2010. Identification of more risks can lead to increased over-optimism of and over-confidence in software development effort estimates. Information and Software Technology 52, 506–516.


