UNIVERSITÀ DEGLI STUDI DI ROMA "TOR VERGATA"

FACOLTÀ DI INGEGNERIA
DIPARTIMENTO DI INFORMATICA, SISTEMI E PRODUZIONE

DOTTORATO DI RICERCA IN
INFORMATICA ED INGEGNERIA DELL'AUTOMAZIONE

XXI CICLO DEL CORSO DI DOTTORATO

An approach to Improving Parametric Estimation Models in the Case of Violation of Assumptions Based upon Risk Analysis

Salvatore Alessandro Sarcià

A.A. 2008/2009

Docenti Guida/Advisors:
Prof. Giovanni Cantone
Prof. Victor Robert Basili (University of Maryland, MD, USA)

Coordinatore:
Prof. Daniel Pierre Bovet

ABSTRACT

Title of Dissertation:	An Approach to Improving Parametric Estimation Models in the Case of Violation of Assumptions Based upon Risk Analysis
Doctor of Philosophy Candidate:	Salvatore Alessandro Sarcia'
Dissertation directed by[1]:	Professor Victor R. Basili University of Maryland Dept. of Computer Science, MD, USA Fraunhofer Center for Experimental Software Engineering, Maryland, MD, USA Professor Giovanni Cantone University of Rome "Tor Vergata" Dept. of Informatics, Systems, and Production, Italy
Advisory committee[1]:	Professor Lionel C. Briand Carleton University Dept. of Systems and Computer Engineering, Ottawa, Canada University of Oslo, Norway Simula Labs Prof. Gerardo Canfora University of Sannio, Benevento, Italy Research Centre on Software Technology Prof. Giancarlo Succi Free University of Bolzano-Bozen Center for Applied Software Engineering, Bolzano, Italy

In this work, we show the mathematical reasons why parametric models fall short of providing correct estimates and define an approach that overcomes the causes of these shortfalls. The approach aims at improving parametric estimation models when any regression model assumption is violated for the data being analyzed. Violations can be that, the errors are x-correlated, the model is not linear, the sample is heteroscedastic, or the error probability distribution is not Gaussian. If data violates the regression assumptions and

[1] Alphabetical order.

we do not deal with the consequences of these violations, we cannot improve the model and estimates will be incorrect forever. The novelty of this work is that we define and use a feed-forward multi-layer neural network for discrimination problems to calculate prediction intervals (i.e. evaluate uncertainty), make estimates, and detect improvement needs. The primary difference from traditional methodologies is that the proposed approach can deal with scope error, model error, and assumption error at the same time. The approach can be applied for prediction, inference, and model improvement over any situation and context without making specific assumptions. An important benefit of the approach is that, it can be completely automated as a stand-alone estimation methodology or used for supporting experts and organizations together with other estimation techniques (e.g., human judgment, parametric models). Unlike other methodologies, the proposed approach focuses on the model improvement by integrating the estimation activity into a wider process that we call the Estimation Improvement Process as an instantiation of the Quality Improvement Paradigm. This approach aids mature organizations in learning from their experience and improving their processes over time with respect to managing their estimation activities. To provide an exposition of the approach, we use an old NASA COCOMO data set to (1) build an evolvable neural network model and (2) show how a parametric model, e.g., a regression model, can be improved and evolved with the new project data.

AN APPROACH TO IMPROVING PARAMETRIC ESTIMATION MODELS IN THE CASE OF VIOLATION OF ASSUMPTIONS BASED UPON RISK ANALYSIS

By

Salvatore Alessandro Sarcia'
sarcia@disp.uniroma2.it

Dissertation submitted to the Doctoral School of the
University of Rome "Tor Vergata", Department of Informatics, Systems, and Production (Faculty of Engineering), Rome (Italy) in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
December 12, 2008

To Cristina, who gave me the reason of my life

Do not search for Glory,
Do not search for Money,
Do not search for Notoriety,
Succeed by searching for Knowledge.

(Rome, December 12, 2008)

Acknowledgements

I would like to express my sincere gratitude to the following people and organizations for their contributions and support, which made this thesis possible.

I thank my wife Cristina very much. My success is her success. She allowed me to undertake this incredible adventure and unconditionally accepted to stay alone in Italy for more than one year and a half during my job in the USA. I thank my all family for their constant support and encouragement. Thanks Mamma, Papà, Alessia, Ludovica, and Iolanda.

I would like to thank two brilliant people who made this work possible, Vic Basili and Giovanni Cantone. They believed in my capabilities and invested their time and resources on me. I will never forget it. It has been a great honor to work with you. Vic gave me words, concepts, and methods. I remember our discussions on Improvement, Reuse, and Storage of Experience. I appreciate very much his terrific intuition and capability to come up with well-founded concepts and turn them into practical approaches. I also thank Vic for his humanity, goodness, and simplicity. Thank Vic for making me part of your circle. I enjoyed very much our conversations in front of a good bottle of Italian wine on how people feel life, their expectations, and failures. I would like to thank Giovanni Cantone, as well. He is a brilliant professor who gave me the thrust of the work. I learned by him his capability to conjugate new technologies with traditional research methods. Thanks Giovanni for sharing your expertise with me. It is my pleasure to remember our discussions on the state of research in Italy and the perspectives for our younger scientists.

I wish to thank people from the Fraunhofer Center Maryland (USA) and University of Maryland for discussing the topics presented below, and providing suggestions for improvement. In particular, I thank Vic Basili,

Marv Zelkowitz, Forrest Shull, Jeff Carver, Sima Asgari, Daniela Cruzes, Taiga Nakamura, Nico Zazworka, and Rola Alameh. I also would like to thank Nachi Nagappan (Microsoft, US) for spending some time on listening to my research when the work was growing.

A special thank goes to Magne Jørgensen (Simula Research Labs, NO) and Mario Piattini (Universidad De Castilla-La Mancha, SP) who spent their valuable time on reviewing accurately this work. Their contribution was worth for improving both focus and goal of the thesis.

A special thank is for Lionel Briand (Carleton University, CA and Simula Research Labs, NO), Gerardo Canfora (University of Sannio, IT), and Giancarlo Succi (Free University of Bozen-Bolzano, IT), members of the dissertation committee, for accepting this duty and suggesting their valuable improvements.

I am grateful to professors H. Dieter Rombach (Fraunhofer IESE, GE), Filippo Lanubile (University of Bari, IT), Dag Sjoberg (Simula Research Labs, NO), and Tore Dybå (SINTEF ICT, NO) members of the committee of the Intl. Doctoral Symposium on Empirical Software Engineering (IDoESE, Kaiserslautern, 2008) for expressing their comments and notes to this work.

I would like to thank people from the Department of Informatics, Systems, and Production (DISP) of the University of Roma "Tor Vergata" for their support. In particular, I thank Prof. Giuseppe Italiano and Prof. Daniel Pierre Bovet. I also thank Prof. Barbara Torti from the Department of Mathematics for attending and discussing my preliminary defense.

A special thank goes to my colleagues Davide Falessi, Gianluca Grilli, Fabio Dellutri from the University of Rome "Tor Vergata" and Margaret Byrns from the University of Maryland.

Rome, December 12, 2008
Salvatore Alessandro Sarcia'

Table of Contents

List of Tables

List of Figures

Preface

This dissertation deals with software estimation for learning organizations. Our motivation is that, estimation of software engineering variables (e.g., effort, size, number of defects, fault proneness, number of test cases) with high performance is the basis for success of every software organization aiming at improving their processes and products over time. Organizations can increase their competitive advantage by estimating correctly and suitably variables of interest. Nevertheless, estimation of software engineering variables with high performance is still an unsolved problem. There are several causes for this shortfall. We focus on parametric models violating their assumptions because such a situation is one of the most severe and ignored reasons of failure for software organizations.

The starting point of this work is that, unlike the past research, we do not try to find the most accurate model. We focus on improving the model that the organization uses and trusts. The reason for this choice arises from the consideration that, the research effort on software estimation over the last thirty years proved that searching for the best model over all of the contexts and situations leads to misleading outcomes (e.g. there not exists the best model among all of the contexts and situations). The point is that, organizations do not need to find the best model among all. They need to improve their model over time in the context where they operate according to the variables relevant for the organization (i.e., the best model related to their environment). Moreover, there is a practical reason why organizations do not need to search for the best model among all. Organizations do not usually change their estimation model because they cannot measure new variables over historical projects. This means that, the model can only improve in terms of shape, estimation procedure of parameters, and variable redundancy.

With respect to new projects, the model can evolve in terms of new variables, as well.

Novelty of this work is that, we use uncertainty for selecting the right model, not only the accuracy. The problem with the accuracy is that, it is dependent on the statistics used for its evaluation (e.g. once an error measure has been selected, accuracy is expressed by mean and standard deviation over the error sample). If we consider a different error measure the accuracy may change. The motivation of using uncertainty is that, unlike the accuracy, the uncertainty that we define in this work is invariant, traceable, and repeatable.

Another innovative point of this work is that, we define and use a multi-layer neural network for analyzing the estimation error over a number of past observations. Then, we use such a network for quantifying uncertainty arising from the estimation model. The motivation of this choice is that, the neural network we define is able to shrink the magnitude of the uncertainty intervals dramatically with respect to the intervals calculated by traditional approaches. This happens because of the Bayesian paradigm on which the neural network is based. Moreover, the network can deal with every kind of error that affects a parametric estimation model (i.e., model error, scope error, and assumption error). With respect to model error, the network quantifies uncertainty by taking into account violations of the model, as well.

Motivation for using an additional model to quantify uncertainty arising from the estimation model is that, any model will always be affected by error. This occurs because a model is a limited representation of the reality. Therefore, once we have improved the model at best, there is a residual error that we cannot remove. That is why we base on analyzing and predicting residuals in a systematic way (e.g. using a neural network for discrimination), and use uncertainty for improving the estimates of the estimation model correcting their bias.

The approach we propose in this work seems to be slightly in opposition with the concept that "*easy is better*". There is a research thinking line arguing that, it is better using rough non-complex estimation approaches (e.g. human judgment, models based on uncomplicated calibrations) than a well-founded complex model (e.g. non-linear regression functions). We do not have the answer whether the former is actually better than the latter, but this dilemma seems to be very close to another famous dilemma between structured development process such as RUP (IBM-Rational Unified Process) and Agile approaches. Some researchers say that agile is better than traditional ones to exploit unavoidable changes as opportunities to improve and refactor the system. However, when dealing with huge systems involving thousands of people, a well-founded development process should be preferred over the agile methods. Therefore, we prefer talking about "*use the right tool*" for your problem instead of "*easy is better*" because the problem comes first and the solution comes later, not the opposite. Arguing that "*easy is better*" is like putting before the solution to the problem. We also argue that, it is better to use an uncomplicated estimation model whenever we can, but there are many complicated situations where an easy estimation approach does not apply. Moreover, organizations using easy estimation approaches would better trust their estimates whether a well-founded and complex model provided similar results as the ones obtained by the former.

This work is composed of two parts. The first part is based on two further sections 1) the problem definition and 2) the mathematical solution. The second part deals with an application of the mathematical solution in the software engineering field. The mathematical solution is general insofar as it can be used for dealing with problems arising from different contexts (i.e., not limited to software engineering) such as economics, psychology, and science. To explain how the solution works in the software engineering field, we present a case study referring to the most famous problem and model in

the software engineering field, i.e. software cost estimation through the COCOMO-I model. It is worth noting that, we do not want to suggest organizations using COCOMO-I. The case study aim is to show how to apply the approach in a well-know situation where researchers and practitioners are familiar with. The case study does not aim at empirically comparing the involved models, but it shows the way of doing so.

Chapter 1

Introduction

1 Introduction

This research supports learning organizations [BASILI92B] in achieving their business goals and gaining competitive advantage. These organizations need to manage projects effectively and deliver products on time, on budget, and with all the functions and features required. To this end, one of the most important keystones for their success is to be able to estimate correctly the variables of interest of the project, task, and module (e.g., effort, fault proneness, and defect slippage). For instance, a software organization may need to quantify the cost of developing a software system in order to bid on the contract. So, the success (gaining the contract or delivering the sub-system as required) would depend on the capability to get the most accurate software cost estimate. Consequently, getting accurate estimates is a strategic goal for these organizations.

Estimation accuracy is not only about yielding estimates as close to the actual value as possible, but also estimating the variability of the estimates. In this work, we refer to the improvement issue in a twofold way, (1) improving the correctness of the estimates (i.e. shrinking the prediction error) and (2) improving the way the spread of the prediction error is calculated (i.e. improving the inference about the predicted variable). The latter is also referred to as estimating the estimation model (EM) uncertainty (e.g., quantifying the risk) or calculating prediction intervals (PIs) of estimates (e.g., what is the variability of the next predicted value?).

Once an organization has invested in an estimation model and learnt to work with it, it is hard for them to switch models. However, they need to improve their estimation capability over time, and thus their model. We use

the uncertainty arising from the estimation model to evaluate, select, and improve the model that organizations use. A key point is that, the accuracy calculated by some summary statistics over the estimation error (as is usually done) is not sufficient to evaluate and select the estimation model. In this work, we present an approach for dealing with estimation models. We improve the model itself over time and correct the estimates by coping with all of the types of error that affect estimation models.

Kitchenham et al. [KITCHENHAM97] refer to several sources of error, i.e. errors found in the measurements (Measurement error), produced by unsuitable mathematical models (model error), wrongly assumed input values (assumption error), or inadequacy of the projects chosen for building such estimates (scope error). Errors can be represented by (stochastic) variables that we can study and even try to predict, but we cannot avoid. For this reason, in dealing with these issues, we have realized that improving estimation models over time is not enough for supporting those software organizations. They also need to measure the impact of the error on the stated software goals when using the estimation model in their own environment. In other words, software organizations need to both improve their estimation models over time and analyze the risk for planning suitable mitigation strategies.

Researchers and practitioners prefer violating assumptions (e.g. homoscedasticity, model linearity, and normality of the distributions) and ignoring error sources, rather than dealing with the consequences of such violations and errors. To continue violating assumptions and ignoring errors pretending that everything is fine “for the sake of simplicity” is not a good way of managing learning organizations and improving estimation models. Conversely, as we propose in this work, investigating empirically consequences of those violations can improve both estimation and inference of parametric models. For this reasons, even though we refer to parametric

estimation models, the proposed improvement strategy is mainly based on non-linear models, non-parametric and Bayesian statistics.

Organizations should be able to integrate the estimation improvement into their general improvement process (e.g., QIP [BASILI92B]). By contrary, over the last three decades, scientists and practitioners have been trying to support software organizations in finding the best estimation model instead of trying to improve those models that organizations have been using. The result of this huge effort is that currently software engineering practitioners neither have the best estimation model nor appropriate improvement techniques for those models. In other words, as argued in [SHEPPERD07A], [MYRTVEI05], this thirty-year research effort has been practically disappointing. The impossibility to find the best model hints that comparative studies make sense only within a specific context. If we change that context, the results of comparative analyses are no longer valid. This is the reason why, we focus on improving estimation models over time exploiting the past experience of the organization and do not care about finding the best model.

We have organized the work in some parts. First, we define the estimation improvement process that we will refer to in the work. Secondly, we present linear and non-linear estimation models, known improvement strategies, and techniques for evaluating PIs (i.e. uncertainty and risk) for both linear and non-linear models. Thirdly, we define an error taxonomy showing errors that we really need to worry about and their consequences. Subsequently, we present the problem, i.e. we answer the question why currently used parametric estimation models fall short of providing valid and reliable results when regression assumptions are violated. We proceed by defining the mathematical solution and its application to software engineering problems (the proposed approach).

Finally, we show how the approach works in a real case (i.e. COCOMO NASA data set). Of course, we leave out evaluation and comparison between

the proposed approach and the existing ones because the aim of this work is to define the methodology as a whole, not to evaluate it. We finish by discussing benefits and drawbacks of the proposed approach.

Chapter 2

Model definition

2 Parametric Estimation Models

The estimation techniques we refer to are based upon parametric models that receive inputs and provide an output (the estimate), see Fig. 1, where we estimate, through the model EM, a dependent variable (y) based on a number of independent variables (x).

The accuracy is checked by analyzing the estimation error, i.e. the deviation between the estimated output (when we feed the model with actual inputs) and the actual output.

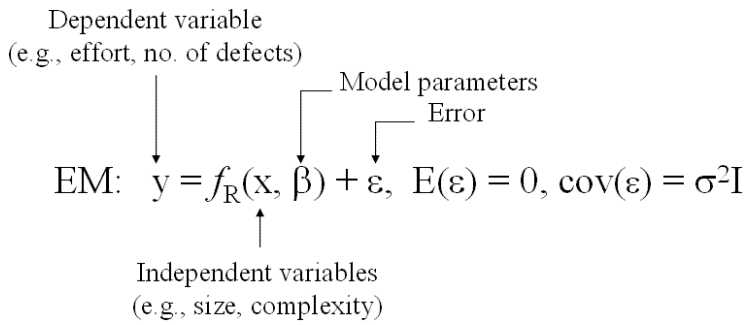

Fig. 1. A Parametric estimation model.

Even though we refer to functional and parametric estimation models, we may think of the EM as a system, which produces estimates based on input values. For instance, human-judgment approaches fall in this category. That is, the experts is the EM, using their expertise and local data as inputs x, to provide an estimates y.

In this work, we use some terms with specific meanings and so we clarify those meanings here to avoid misunderstandings.

An *Estimate/Estimation* is any prediction of the quantity of interest that can be greater or lower than the value it is predicting. An *Estimation model* is any system able to provide estimates on the quantity of interest, e.g. COCOMO, regression models, machine learning, and human-based judgment. An *Evaluation model* is any system able to provide a decision about whether the performance of the *estimation model* is acceptable or not.

We refer to a variable as a (1) *characteristi*c when we want to emphasize its role in the context (2) *feature* when we want emphasize its role in a neural network, (3) *factor* when we want to emphasize its role in regression models.

Chapter 3

Background

3 Estimation Improvement Process

In this section, we provide an overview of the estimation improvement process (EIP) to give a context for the evolution of EM over time. We will define EIP in more details in Section 6.3. The EIP can be considered as a specialization of the Quality Improvement Paradigm (QIP) [BASILI92B], [BASILI92] for estimation processes (Fig. 2). The novelty of the EIP is that, it exploits experience packages based on specific kinds of neural networks, which can provide estimation model uncertainty for the organization without making any specific assumptions. Once such neural networks have been built using the organization's data, the Experience Factory (EF) can provide a project with automated support tools, which simplify the reuse and exploitation of experience. We call such a neural network-based package an Automated Experience Package (AEP), a tool for supporting project organizations in estimations. AEPs together with the strategy defined in Section 6.1 can (1) provide estimates throughout the development process, (2) make estimation uncertainty analyses, (3) drive the risk mitigation strategy, and (4) detect estimation model improvement needs without continuous human interaction. The EIP is a tailored version of the QIP with additional specifications aimed at supporting a project organization in making estimates, an activity usually included in the project management plan at the project organization level.

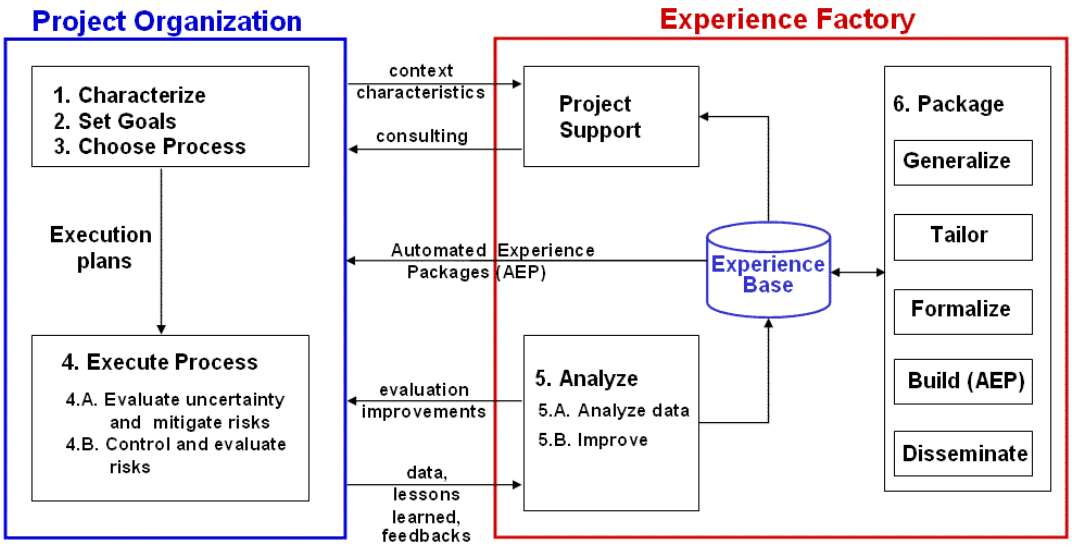

Fig. 2. Experience Factory implementing the Estimation Improvement Process.

The EIP is composed of six steps [BASILI92B]:

1. *Characterize* the current project and its environment with respect to models and variables, e.g., what is the current model for cost estimation?
2. *Set Goals* for successful project performance and improvement, e.g., what the acceptable estimation error?
3. *Choose the Process* for achieving the stated goals of this project, e.g., what added activities are needed to use and evolve the AEP?
4. *Execute the Process* for building the products, collecting data, and perform activities specific to the estimation process: (4.A.) *Estimate uncertainty and mitigate risks*, i.e. based on the products of the AEP, state the project estimate that minimizes the estimation risk (e.g. a failure) and (4.B.) *Manage risks* by controlling and evaluating risks by monitoring the project and checking whether it meets its goals.
5. *Analyze* the data to evaluate the current practices, determine problems, record findings, and make recommendations for future project

improvements. For the estimation process, we need to (5.A.) *Analyze data* about the project performance, and (5.B) *Improve* the estimation model for online support of the project organization.

6. *Package* the experience by building an improved version of the AEP and save it in an experience base to be reused on future estimation of projects.

This way of managing experience allows the EF to check automatically whether the project organization is complying with the organization lessons learned and the project organization to reuse automatically the organization experience to mitigate the risk. The project organization can exploit automated support tools, i.e. the AEP, for managing estimates before, during, and after the project execution (e.g. shrinking the effort spent for the project management).

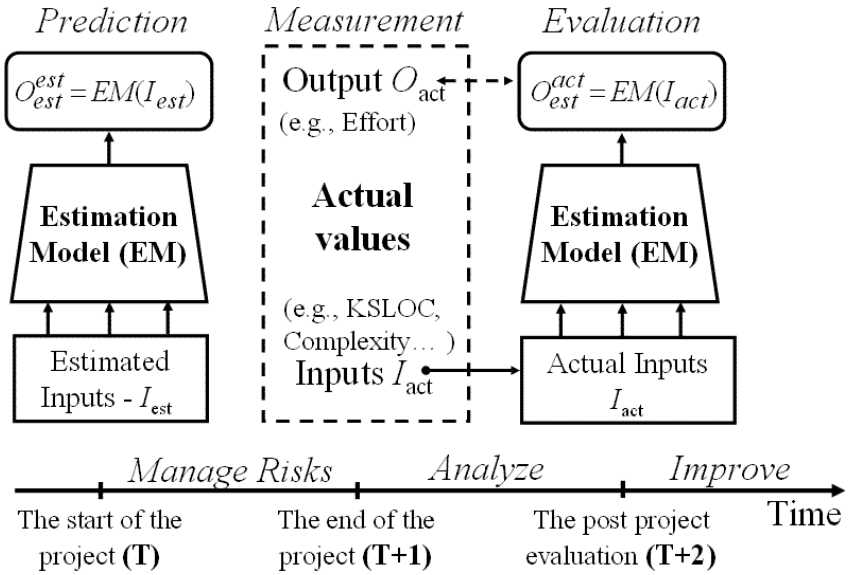

Fig. 3. Different kinds of estimates (Prediction and Evaluation).

Once the EM has been built, we can consider two different kinds of estimates (Fig. 3), the estimate obtained feeding the model with estimated

inputs (time T) and the estimate obtained feeding the model with actual inputs (time T+2).

We will use time T to represent the start of the project, time T+1 to represent the end of the project when actual inputs can be known, and time T+2 to represent the post project accuracy evaluation. Note that, Fig. 3 shows some activities already reported in Fig. 2. Estimate at time T (Prediction) refers to the estimation where the input values (I_{est}) are estimated and the output, o_{est}^{est} = EM(I_{est}) is calculated based upon those estimated inputs. Since we do not know the actual values of the input, I_{act} at time T, we cannot check the accuracy of the EM from these estimates o_{est}^{est} (e.g., in cost estimation models, actual size is different from estimated size),

At time T +1, we know both I_{act} and O_{act}. Thus, at time T+2 we can calculate o_{est}^{act} = EM(I_{act}), the output estimated by actual inputs, and we know I_{act}, the actual output. So to evaluate the accuracy we can compare o_{est}^{act} to O_{act} (horizontal dashed double-arrow line in Fig. 3). If the estimation model was perfect, the estimated values (o_{est}^{act}) should be equal to the actual values (O_{act}). It almost never happens, hence we have to deal with prediction errors and think about how improving the considered estimation models.

3.1 Improving estimation models

Mathematically, the EM in Fig. 1 is a regression function f_R such that y = $f_R(x,\beta) + \varepsilon$, where R stands for regression, x represents a set of independent variables, y is the dependent variable and β is the set of parameters of f_R. The component ε is the aleatory (unknown) part of the model representing our uncertainty about the relationship between independent and dependent variables. Function f_R cannot be formulated because we cannot calculate parameters β, as we do not know every point of the population. But, we can estimate β by finding a set of estimators b such that they minimize an error

function (e.g., least squares). To estimate b, we consider the relationship $O_{act} = f_R(I_{act},b)$, where O_{act} represent the actual values of y, I_{act} are the actual values of x, and b are parameters being estimated. Because of ε, f_R provides $O_{est} = f_R(I_{act},b)$, not O_{act}. Then, the difference $e = O_{act} - O_{est}$ is a vector of errors representing ε (called residuals, with $e \neq \varepsilon$). The most important part in modeling is to find the best estimates for β. This activity is called *parameter estimation*, *fitting,* or *calibration*. For calculating b, there exist some strategies, which are based on selecting b so that the function best fits the observations (O_{act}). One way this can be reached is by minimizing the sum of squared residuals, e.g., the least squares (LS) function. From a practical point of view, the minimum of the sum of squares can be found by setting the gradient to zero, where the derivative is made with respect to each parameter. Therefore, minimizing the cost function means solving an equation system of partial derivatives set to zero. If the system equations are composed of a linear combination of the parameters sought, the EM is linear in the parameters, and we have a closed solution. If the equation system is composed of non-linear equations of the parameters sought, the EM is non-linear in the parameters and the solution can be found iteratively.

3.1.1 Regression assumptions

To have a closed solution, when applying LS an assumption is required. In particular, f_R has to look like the following function, $y = \beta_0 + \beta_1 x_1 + \ldots + \beta_Q x_Q$ (or any polynomial quadratic, cubic etc.), where each monomial is a linear combination of the parameter sought [RAO73], [WEISBERG85]. Note that, since the characteristic expressing the relationship between inputs and output is called the model shape, the equation for y has a linear shape. For example, a second-degree polynomial is linear in the parameters and has a quadratic shape. The LS iterative method does not require that the system of equations of partial derivatives is linear in the parameters. This is the reason why, this

method is applied with estimation models that are non-linear in the parameters. The complexity of a model is another characteristic expressing the number of parameters composing the model. The more parameters, the more complex the model.

If we want to get the best linear unbiased estimators (BLUE) of β (Gauss-Markov theorem [PEDHAZUR97]) and use the model for inference, LS requires some assumptions, reported below. We will call these assumptions "regression assumptions" meaning that if they do not hold the parameter calculation may be affected by error and b would be biased. The regression assumptions are:

(1) Errors ε are not x correlated

(2) The variance of the errors is constant (homoscedasticity), $\text{cov}(\varepsilon) = \sigma^2 I$

(3) Errors ε are not auto-correlated

(4) The probability density of the error is a Gaussian, $\varepsilon \sim \text{NID}(0, \sigma^2 I)$, i.e. there are no outliers, skewed/kurtotic distributions, and measurement error.

It is worth noting that the LS method does not actually require that the probability distribution of the errors is a Gaussian. This assumption is required for making inference (e.g., deriving the confidence and prediction intervals for the model parameters and the variables, see Section 3.2). When dealing with real cases some of those assumptions may be violated and the estimators b may be biased. *The core of this work addresses strategies for dealing with estimation models when regression assumptions are violated.* Conversely, if regression assumptions hold, LS estimators b have minimum variance and represent a *Maximum Likelihood Estimation* (b_{MLE}) to the minimization problem stated above [PEDHAZUR97]. This means that, the parameters of f_R correspond to the observed data (e.g., $y^{(1)\ldots(N)}$) having the highest probability of arising. Eqn. (1) formalizes this situation.

$$b_{MLE} = \arg_b \max p(y^{(1)},...,y^{(N)} \mid x^{(1)},...,x^{(N)}, b) \quad . (1)$$

Where, p(.|..) denotes a conditional probability function, which is approximated by $p_b(y|x,b)$ having the mean $\mu_b(y|x,b)$. Thus if the regression assumptions hold, LS and MLE provide the same parameters b_{MLE} (see Markov-Gauss theorem [MCQUARRIE98]), therefore $\mu_b(y|x,b) = f_R(y|x,b)$.

The main problem with this approach is that if these assumptions do not hold, e.g., the probability distribution $p_b(y|x,b)$ is not Gaussian, the LS method does not yield b_{MLE} any more.

In the literature, however, we can find many examples where these assumptions are violated and the closed LS technique is applied anyway. Another practical problem with using LS concerns the shape of the model. As stated above, very often, to simplify the parameter calculation and have a closed solution for calculating parameters b, researchers and practitioners assume they know the best shape between inputs and output, e.g., "[…] we assume that the multi-regression function has linear or (log-linear) shape […]", when the shape is not known at all. For a practical discussion on this approach see [JØRGENSEN04A], and [JØRGENSEN03]. Then, the resulting regression function will no longer be correct (i.e., it is biased). So, predictions and inferences drawn from this flawed model will be inaccurate (i.e., spread may be overestimated or underestimated and estimates biased).

3.1.2 Dealing with the consequences of violations

The reasons why parameters of a regression function may be biased, its estimates inaccurate, and the inference drawn from it incorrect are the following [MCQUARRIE98]:

- *Variable model error (VrblME)*: The model is missing some relevant variables; hence, it is not able to explain the output.

- *Redundancy model error (RdndME)*: The model includes too many variables that negatively affect the correctness of the model parameters
- *Flexibility model error (FlexME)*: The model is not flexible enough to represent the relationship between inputs and output. For instance, this error can happen when we choose models that are linear in the parameters, e.g. ordinary least squares (OLS)
- *Violation model error (VltnME)*: The model parameters and inferences may be biased because of the violation of one or more regression assumptions (see "regression assumptions" above).

We use the words "model error" meaning that the error is about the incorrect estimation of the model parameters, i.e. the error is about the model.

An estimation model can be used for prediction and inference. When using the model for prediction, we are interested in having estimates as accurate as possible. When using the model for inference, we are interested in evaluating the prediction intervals on the dependent variable as well as evaluating the confidence intervals on the model parameters b. A prediction interval is a range where the next estimate will probably fall within a chosen confidence level (e.g., 95%) and it is directly related to the bias and spread of the prediction error (Section 3.2). In this work, we refer to using estimation models both for prediction and inference on the independent variable. Therefore, based on data previously observed, we aim at improving the prediction capability of models over time and evaluating correctly the related prediction intervals. Improving the prediction capability refers to estimating the best model parameters. Correctness of prediction intervals is about finding a way of reducing the error brought about by violations of assumptions on which the model has been built, e.g. regression assumptions and linear complexity.

To improve parametric estimation models in the sense stated above (i.e. increasing the correctness of the model parameters and estimating the right

prediction intervals on the dependent variable of the model), improvements can be made for each bullet in the list above. In the current section, we mainly refer to improving prediction accuracy, while, in Section 3.2, we show more deeply some techniques for evaluating prediction intervals.

There is not way to deal with *VrblME*. We must find the right variables for the model to be correct. Finding good variable candidates can be done by looking at previous research, asking experts, or inferring missing variables based on a context analysis. For instance, the approach proposed in Section 6.1 shows a way of detecting what the model is missing and the case study discussed in Section 7 illustrates a way of exploiting data coming from previous research [PROMISE].

There are some techniques for dealing with *RdndME*. The most popular is called stepwise regression, forward or backward [MCQUARRIE98]. Of all the possible models, each having different numbers of variables, stepwise regression chooses the one having the most significant correlation, i.e., choosing the model having the highest adjusted R-squared. The problem is that the variables removed might be significant for other situations, e.g., future projects. Thus, since we do not know which variables are least significant, we should take into account all models that can be built using all possible combinations of variables (i.e., use an exhaustive procedure). For instance, if we have Q variables, there will be 2^Q different models, because each variable can be included or not (dichotomous decision). However, this procedure is usually too expensive to be executed in real cases. Instead of considering individual variables, one can take sets of multiple variables, so that the number of different models that need to be considered would be smaller. Although applying stepwise regression can improve the model parsimony and its accuracy, it requires one more assumption. To apply stepwise regression *multicollinearity* should not affect the model. This is a statistical effect where two or more variables are highly correlated, i.e. one or

more variables are redundant, and they can be obtained as a (linear) combination of other variables included in the model. Stepwise regression is requested when we are interested in knowing which variables actually influence the independent variables (e.g., for inference purposes of the independent variables). If we use the model for prediction or inference on the dependent variable, other techniques of feature reduction can be applied.

Feature reduction techniques are another way to deal with *RdndME* as well; they find an equivalent configuration composed of less input variables than the initial set. *Principal Component Analysis* (PCA) is one such technique; it finds independent linear components, which can be used to express the original variables [JOLLIFE86], [FENTON93]. *Curvilinear Component Analysis* (CCA) is another technique that involves finding both linear and non-linear independent components [BISHOP05A, pp. 310-319]. Because CCA is able to also perform PCA, we will mainly focus on CCA. The difference between stepwise regression and CCA is that the former removes irrelevant variables so the resulting variables are a subset of the initial set; it also assumes non-multicollinearity. CCA removes redundancy by turning the initial configuration into a more parsimonious one where the resulting variables may not correspond to the initial set and it does not assume non-multicollinearity. These techniques are explained in more detail in Section 6.

To deal with *FlexME*, i.e., finding the right complexity, the procedure is to (1) consider different families of functions, (2) compare them to each other, and (3) select the best, i.e., the one yielding the least generalization error (i.e., the expected estimation error). For instance, one may consider polynomials with different complexity and for each of them calculate the generalization error, hence, using the generalization error for selecting the best model. The problem is that, we cannot calculate the generalization error because it requires knowing every point of the population. We can only estimate it.

Vapnik proves that leave-one-out cross-validation (LOOCV) provides an unbiased estimate of the generalization error [VAPNIK95]. This procedure is explained in Section 6.1. Let N be the cardinality of the data set. LOOCV starts with considering a linear-in-the-parameter model and it goes on by increasing the complexity, i.e. it considers non-linear-in-the-parameter models having an increasing flexibility, until a satisfactory condition is met. For each considered family of functions having a different complexity (i.e., a different number of parameters), LOOCV builds N models by removing a data point from the data set. Each of the N models is trained with N−1 data points and the model error is calculated on the data point left out. The generalization error estimator is calculated as the average of the N errors coming from each individual model trained with N-1 points. Usually, the error is the square root of the mean squared error (i.e. $\mathrm{SQRT}((1/N)\cdot\sum_h (e_h^{(-h)})^2)$, where the h-th residual $e_h^{(-h)}$ is calculated on the h-th observation left out, with h = 1 to N). Because of the LOOCV cost, sometimes k-fold cross-validation (KFCV) is preferred. However, it does not provide an unbiased estimation of the generalization error. KFCV consists of taking out k data points instead of only one and running the same procedure as LOOCV. For more details on KFCV see [DREYFUS05]. Note that, comparing only polynomial families with different complexity is not enough because polynomials are not indefinitely flexible. If we used only polynomials, we may select the model having a minimum error relative to the model, but we may not be sure that it would be the smallest error overall (i.e. among all the possible models). As explained above, to avoid the complexity error, we have to increase flexibility of the non-linear-in-the-parameter functions until the generalization error decreases. We stop increasing the number of parameters, when the generalization error increases or keeps the same (i.e., the satisfactory condition) [BARRON93]. In other words, the model having the

best complexity is the most parsimonious one being flexible enough to explain any relationship between inputs and output. To have models with arbitrary flexibility, we can consider non-linear-in-the-parameter models that are a generalization of usual regression models.

When the size of the sample N goes to infinity, traditional least squares procedures applied to linear-in-the-parameter models such as polynomials provide the "true" parameters β of the regression function (i.e., they are not estimates of β). Conversely, non-linear-in-the-parameter models cannot provide the true parameters of the regression function because there is no closed solution to the least squares problem for such models (i.e., there is an iterative solution). However, as long as N is finite, linear-in-the-parameter models are not able to provide the true value of the regression function as non-linear-in-the-parameter models. Then, since a non-linear-in-the-parameter model can be made indefinitely flexible for a fixed number of input variables and a linear one does not, the former is more flexible and parsimonious than the latter. In fact, operatively we cannot consider models having an infinitive number of variables, but we can increase the number of parameters of non-linear-in-the-parameter models. Fixing the number of variables and increasing the number of parameters of the model is the essence of the parsimony [BARRON93]. Therefore, in real cases where we do not have an infinitive number of variables and observations, non-linear-in-the-parameter models can provide better estimates than linear ones (Section 3.6).

We mainly refer to Multi-Layer Feed-Forward Neural Networks (MLFFNN) trained with Backpropagation [BISHOP95A], [RUMELHART86], which are non-linear-in-the-parameter models and they theoretically have an infinitive flexibility (Section 3.6). MLFFNN are called arbitrary approximators (or universal approximators) not because they provide arbitrary outcomes. The "arbitrary" aspect is only about their unlimited flexibility. Applying LOOCV and feature selection and/or

reduction to MLFFNN allows us to find the best non-linear model without complexity limitations. This is a suitable way of dealing with the flexibility problem. Therefore, if we want to avoid the flexibility model error, the only way is to use arbitrarily flexible models. Note that, Backpropagation does not provide a closed solution to the LS minimization problem for calculating model parameters. Backpropagation is an iterative method that requires some optimization technique to make the training process faster. For instance, Levenberg-Marquardt is an optimization technique, which is 60-80 times faster than Backpropagation without any optimization [HAGAN94]. Using MLFFNN has further benefits than usual regression models. In Section 3.6 and 3.7, we will explain it in more detail. Using the LS error function with MLFFNN requires similar regression assumptions to linear regression, but it removes the limitation about model flexibility and that the model variables may be affected by error (i.e., there will be outliers or skewed probability distributions). Bishop shows that, although the sum-of-squares error function derived from the principle of maximum likelihood, Eqn. (1), does not require that the distribution of the dependent variable is Gaussian, estimates provided by a model trained by the sum-of-squares and a global variance parameter (i.e., the variance is assumed to be constant) are the same as the ones provided by a model having a dependent variable with a Gaussian distribution and the same x-dependent mean and average variance [BISHOP95A pp. 201-206]. Therefore, if we cannot make any assumption on normality of variables and errors, we should choose a different error function with respect to least squares as we explain later in this section.

Note that, as we will explain in Section 3.6, a MLFFNN provides a regression function conditioned to the observations, i.e. it is exclusively built on the observed data. This means that, where this data is not available (i.e., across specific intervals), the prediction capabilities of the model decrease. If the observed data is “regularly” distributed, a MLFFNN is able to provide

correct estimates with a fewer observations than linear models (i.e., usual polynomials). In other words, non-linear-in-the-parameter models can provide higher accuracy than linear-in-the-parameter models with the same number of observations [DREYFUS05]. Moreover, the MLFFNN prediction capability improves as the number of observations grows. Therefore, they are models that are consistent with learning organizations aiming at improving their estimation capability over time [BASILI92B].

When regression assumptions do not hold, we should be aware of a number of consequences. The consequences may affect both the prediction capability of the model and the inference drawn from it, i.e. either bias or spread of the estimation model may be incorrect and/or inefficient, see Section 3.2 for more details. Therefore, if the model is biased, estimates may be incorrect (biased) and prediction intervals underestimated or overestimated (inefficient). To improve the estimation capability of the model, we can use non-linear-in-the-parameter models together with LOOCV and feature reduction (e.g., CCA) as explained above and in Section 3.6. When trying to improve models used for inference (e.g., estimating the expected value and spread of the error), the problem is more complicated. It is important noting that, when estimating the model parameters through an iterative procedure such as Backpropagation, we should consider further uncertainty because of it. Conversely, prediction intervals would be underestimated.

With respect to the regression assumptions reported above, if (1) errors ε are x-correlated, the expected value of the error should be calculated by a regression analysis where the independent variables represent the error and the independent variables are the same x-variables of the estimation model. Obviously, we do not know ε. Therefore, the best we can do is to use the residuals e instead of ε. It is very important noting that, when using a non-linear-in-the-parameter model along with LOOCV and feature reduction such

as CCA, the resulting model is able to provide the expected value of the dependent variable even if there is no x-correlation. This happens because of the intercept, which is called bias in non-linear-in-the-parameter models (Section 3.6). Therefore, as we have already mentioned, if we use such an x-regression analysis we get correct results even if there is no correlation between the x-variables and the error, parsimony a part (Section 3.6). Therefore, we should apply non-linear-in-the-parameter models for estimating the x-dependent expected value of the errors (i.e., the model bias). This second regression analysis should be based on as few assumptions as possible. For instance, we should start with removing the normality assumption, the homoscedasticity assumption, and use non-linear-in-the-parameter models. Note that, the considerations on improving estimation models apply also to the regression analysis between x-variables and errors.

If (2) the model is heteroscedastic, the expected error is not offended, but the spread may be inefficient (i.e., type I or type II errors may occur). We mean that, spread should not be considered constant as it usually is, but it should be a function of the independent variables, i.e., $\sigma^2(x)$. Since we cannot know $\sigma^2(x)$, we may estimate it, i.e. we may calibrate a non-linear-in-the-parameter model r(x,u) where r would be a function of variables x and u the parameter estimators defining the function r calibrated on the observations [NIX94]. Therefore, instead of considering a constant variance parameter, we should use an x-dependent non-linear function yielding a different variance value according to the x-values. In particular, Nix et al. use just one network with two outputs, one for the regression function, and one for the variance [NIX94]. This technique acts as a form of weighted regression that aims at weighting in favor of low-noise regions. Note that, weighted regression is also available for linear-in-the-parameter models [WHITE80]. The problem is that weighted regression can eventually reduce the impact of heteroscedasticity, but it is not able to remove it completely. This means that,

we need further improvement techniques when the sample is heteroscedastic rather than applying only linear or non-linear weighted regression.

In literature, there exist further methods to deal with this problem. The most known method is called bootstrap [EFRON93, pp. 313-315]. It is based on a resampling procedure. The method provides an x-dependent prediction interval, which behaves better than the delta method previously applied. Bootstrap can be applied both with non-linear and linear-in-the-parameter models and it applies to parametric and non-parametric distributions [MOJIRSHEIBANI96]. The problem with using the bootstrap method is the cost of resampling to get suitable prediction intervals. An example of the bootstrap application is provided in [ANGELIS00] where the authors show an application of bootstrap to software cost estimation by analogies.

A completely different approach for inferring the spread of the dependent variable is provided by MacKey. He uses the Bayesian framework for evaluating prediction intervals of non-linear-in-the-parameter models [MACKEY91]. This framework estimates the variance of the dependent variable by exploiting posterior probability distributions instead of a priori distribution as is usually done (see Bayesian theorem in Section 3.7). Instead of considering only a single value for a model parameter, as in the maximum likelihood estimate, Eqn. (1), Bayesian inference expresses the variability of the dependent variable in terms of posterior probability distributions and integrates the interesting subset of the distributions [MACKEY91], [HUSMEIER04, pp. 18-20]. The primary importance of the Bayesian framework is the fact that the uncertainty of the dependent variable depends not only on the most probable model (i.e., the one having parameters b_{MLE}), but also on the probability distributions of models that can be built with the sample. However, even though the MacKey's solution is theoretically correct, it has some practical complications. The problem is evaluating the integral that sums the overall uncertainty of every model. In order to handle

the integral, we have to make further assumptions, such as the normality of the prediction error distribution and the prior weight distribution. Moreover, MacKey considers the sample variance to be constant, even though it may be x-dependent. Bishop et al. [BISHOP95B] extend MacKey's analysis by considering an x-dependent variance of the error. For a complete explanation of these techniques, see [HUSMEIER04] and [BISHOP95A, Chapter 11]. In our analysis, where we would like to make as few assumptions as possible, and exploit methodologies that are easy to apply and figure out, MacKey's solution cannot be used as is. In Section 6.1, we will define an empirical approach based on Bayesian analysis for evaluating the variance of the dependent variable without making any specific assumption about the underlying probability distributions. Moreover, it is important to note that, since errors ε (i.e. the aleatory part of the estimation model) are unknown, we have to use the residuals e = (Actual – Estimated) to investigate whether the homoscedasticity holds or not. The problem is that, in software engineering we know that the residuals grow as project size increases, (e.g., software cost estimation). Then a relative measure of the residuals RE = (Actual – Estimated)/Actual, i.e. a measure weighted by the actual value, should provide better results both for linear and non-linear models. In Section 3.2, we present some hints about the right way of choosing an error measure.

If (3) errors are auto-correlated, the parameters b are unbiased but the spread may be unreliable (type I or type II errors may occur). This kind of error can be removed by considering two different approaches, using Autoregressive Conditional Heteroscedasticity (ARCH) models [ENGLE82] or the Two-Stage Regression Procedure (TSRP) [GREENE97]. We do not deal with this kind of improvement because software engineering data is not usually time-series correlated as occurs with financial data.
If we run an auto-correlation statistical test (e.g. Durbin-Watson) and find time-series auto-correlations, we can infer that the auto-correlation is

determined by chance and so we do nothing. Conversely, if there is an auto-correlation effect, we can remove it by applying ARCH or TSRP. If there is a non-time-series auto-correlation we can remove the auto-correlated observations from the data set used for calibrating the estimation model.

If (4) errors are not normally distributed with $\varepsilon \sim NID(0, \sigma^2 I)$, b may be unbiased, but $E(\varepsilon)$ may be different from zero and the percentiles of neither t-student nor z distributions may be good representatives of the population. Therefore, better spread measures are calculated by non-parametric statistics (Section 3.2).

Actually, when the normality assumptions does not hold, instead of calculating the x-dependent mean we may calculate the x-dependent median, which would better represent the expected value of the dependent variable $y = f_R(x,b)$. The x-dependent median can be obtained by minimizing a different error function from LS. This kind of approach is based on *robust statistics* [HUBER81], [MIYAZAKI94]. These techniques try to reduce the influence of outliers on the model parameter calculation. When using MLFFNNs, it is possible to calculate a (conditional) median instead of a (conditional) mean by minimizing the Minkovski-R error with R = 1, Eqn. (2).

$$E(y) = \sum_{N} \left| y - tr^{(N)} \right| . \qquad (2)$$

Where y is the estimate and tr is the actual value over N observations. Minimizing the error function (2) with respect to y gives Eqn. (3).

$$\sum_{N} \text{sign}\left(y - tr^{(N)}\right) = 0 . \qquad (3)$$

The expected error is satisfied when y is the median of the points $\{tr^n\}$ [BISHOP95A, p. 210].

To deal with the measurement error affecting the variables, there is little that we can do. The only option is to avoid the assumption of the normality of the probability distributions and apply a robust regression as explained above.

TABLE 1

IMPROVING INFERENCE OF LINEAR AND NON-LINEAR MODELS

Violation	Improvement	
	Bias	Spread
x-correlation of ε	Estimate parameters u such that $e=f_e(x,u)$ (1)	No improvement needed
Heteroscedasticity	No improvement needed	Consider the variance as an x-dependent function and/or apply the weighted regression (2)
ε are autocorrelated (3)	Use ARCH models for time series autocorrelation	Apply bootstrap to ARCH models
Non-normality of the distributions	Calculate the x-dependent median minimizing the Minkovski R-distance (R=1, robust regression) of non-linear models instead of the sum-of-squares (mean)	Use empirical distributions (non-parametric spread statistics) and avoid using either t-student or z percentiles

(1) f_e is a non-linear regression function treated with LOOCV and CCA, e = residuals, x = independent variables of the model. (2) Variables x are the same as the EM. (3) Autocorrelation does not usually affect software engineering variables. If does, use ARCH models or TSRP.

Table 1 shows the regression assumption violations (left side) and the corresponding improvement to the bias and spread of the model (right side).

3.2 Risk, uncertainty, and accuracy indicators

So far, we have seen that, to improve estimation models we have to use models that are non-linear in the parameters because they can be made indefinitely flexible. When using non-linear-in-the-parameter models for

inference (e.g., evaluating variance of the dependent variable), the problem is more complicated than using linear models because of their non-linearity, i.e. for each function constituting the model there is a higher number of parameters than for linear models where there is only one parameter.

In this section, we delve into the details of the problem of making an inference when regression assumptions are violated. Practically, we have to evaluate the variance of the dependent variable, when using linear and non-linear-in-the-parameter models. Therefore, we are interested in analyzing the relationships existing between the estimated spread (i.e. a mathematical quantity) and the risk (i.e. a software engineering quantity). We will use software cost estimation models as our primary example for exposition, because of the large amount of associated research. Slightly different measures might be considered for other estimation models, but the main ones are reported in this section. For a complete explanation, see [MYRTVEIT05], [KEMERER87].

3.2.1 Risk and uncertainty

We define risk in applying an estimation model as the uncertainty measure of getting a wrong estimate (i.e. the estimate falls out of the stated prediction interval). For evaluating whether an estimate is wrong, we can look at its accuracy. In fact, an estimate is accurate when it is correct and valid. Estimation correctness refers to the capability of being as close to the actual value as possible and estimation validity refers to the capability of being stable over a number of trials. To quantify the uncertainty, we need to study the deviation between the actual and estimated value in a significant number of trials. This deviation is called error and the observation set over the trials is called the error sample. Mathematically, there exist many equivalent ways of estimating the uncertainty in applying an estimation model. To this end, let us consider the definition of error measure.

3.2.2 Error measures and accuracy

There exist many different measures of error. Some are reported below. We can define i-th measure of an absolute error as follows:

$$AE^i = O^i_{act} - O^i_{est} \quad . \qquad (4)$$

Where o_{est} stands for o^{act}_{est}, i.e. an estimate obtained using actual inputs, and o_{act} stands for its actual value. Then, $D_{AE} = \{AE^1, AE^2, \ldots, AE^N\}$ is a sample of errors and N is the sample size. As stated above, we prefer to deal with a relative error (RE) [BOEHM81], Eqn. (5), because in software estimation, the absolute error grows as the size of the project increases (i.e., the variance is not constant), and a weighted measure should be preferred. The relative error on i-th data point is:

$$RE^i = \frac{O^i_{act} - O^i_{est}}{O^i_{act}} \quad . \qquad (5)$$

Then, $D_{RE} = \{RE_1, RE_2, \ldots, RE_N\}$ is a sample of errors and N is the sample size. Note that, the RE's variability interval is $]-\infty;1]$ because the estimated output (O_{est}) varies in $[0; +\infty[$, where $]a;b]$ is a left-open interval and $[a; b[$ is a right- open interval. The problem is that, a relative measure cannot avoid bias and heteroscedasticity of an estimation model. Moreover, RE may be correlated with other context factors [STENSRUD02], [JØRGENSEN04A] therefore its expected value would be better expressed by a regression function on those factors. Actually, the problem with the heteroscedasticity has never been solved. In fact, one of the most violated assumptions when

applying LS is to consider the error distribution homoscedastic, while it is not [JØRGENSEN03].

Another used measure is the Balanced RE (BRE) defined by Miyazaki et al. [MIYAZAKI94], Eqn. (6):

$$BRE^i = \frac{O^i_{act} - O^i_{est}}{MIN\left\{O^i_{act}, O^i_{est}\right\}} \quad (6)$$

Then, $D_{BRE} = \{BRE^1, BRE^2, \ldots, BRE^N\}$ is a sample of errors and N is the sample size. Miyazaki et al. argue that BRE may have properties that lead to better evaluations for some data sets. Unfortunately, it does not happen for every sample [JØRGENSEN04A]. BRE is slightly different from RE because it is not limited on the right side. Both RE and BRE can distinguish between underestimates and overestimates therefore they can both be used for the purposes of the approach presented here.

Once we define the error measure, we can evaluate the accuracy and consequently risk and uncertainty. This task consists of calculating some statistics over the error sample such as mean and standard deviation [JØRGENSEN03]. For instance, the error mean is a bias measure and the error standard deviation is a spread measure. This happens when the considered error measure is able to distinguish between overestimate and underestimate. This is not always the case. Conte et al. [CONTE86] proposed an error measure based on $MRE^i = abs(RE^i)$. Then, $D_{MRE} = \{MRE^1, MRE^2, \ldots, MRE^N\}$ is a sample of errors and N is the sample size of a test set. Based on MRE, they proposed two statistics, MMRE and PRED(H). MMRE is the mean of D_{MRE}, i.e., $MMRE = Mean(D_{MRE}) = Mean(MRE^i)$ for i = 1 to N and PRED(H) is the percentage of estimates within a given error H. For instance, PRED(25) = 80% means that 80% of the estimates fall into an error of 0.25. Kitchenham et al. [KITCHENHAM01] showed that neither MMRE nor

PRED(H) can be used for comparison because MRE is not able to separate bias and spread. Therefore, the MMRE is not a bias measure and its standard deviation is not a spread measure. In particular, those statistics measure spread and kurtosis of the random variable Z = Estimated/Actual, respectively. However, MMRE may be considered as a goodness-of-fit measure of a model.

3.2.3 Uncertainty measure for a univariate case

To quantify the uncertainty then, we need an error measure that allows separating bias and spread such as AE, RE or BRE. As explained above, we will mainly consider RE and BRE because, in software cost estimation, AE grows as the project size increases. If we know the population probability distribution from which the sample has been sampled (e.g., normal distribution), we may use the mean and standard deviation as a bias and spread measure, respectively [JØRGENSEN03]. To calculate an estimate prediction interval (i.e., quantifying the uncertainty of an estimation model), the strategy consists of considering N measures of error (RE or BRE) and calculating an interval where the error of the next estimate will fall with 90% (or 95%) confidence. We call this interval (two-tail) *error prediction interval*, i.e., $[\mu_{DOWN}; \mu_{UP}]$, see Eqn. (7). Based on the error prediction interval, we can apply Eqns (9), (10) or (11), see below, and calculate the *estimate prediction interval*, for RE, BRE and AE, respectively. We use $[\, o^{N+1}_{est,DOWN} ; o^{N+1}_{est,UP} \,]$ to point out an estimate prediction interval. See Section 3.6 for more details [JØRGENSEN03]. Note that, (7) is a 90% confidence interval, which corresponds to the 95th percentile of the Student's distribution with (N—1) degrees of freedom, i.e., $t_{0.95}(N—1)$.

$$\overline{X} \pm t_{0.95}(N-1) \cdot S\sqrt{\frac{1}{N}+1} = [\mu_{DOWN}, \mu_{UP}] \quad . (7)$$

Where, $\overline{x}$ and S are the sample mean and standard deviation of the error distribution (e.g., RE), respectively. This happens because the t-Student's distribution quantile that we have to consider is obtained by the formula 1 - $\alpha/2$ = 1 - 0.1/2 = 1 - 0.05 = 0.95, where α is the required confidence level.

Note that, the prediction interval is wider than the confidence interval for the mean of the error population, Exn. (8).

$$\overline{X} \pm t_{0.95}(N-1) \cdot S\sqrt{\frac{1}{N}} \quad . (8)$$

In fact, it is narrower than the prediction interval because the error of the next estimate, $X^{(N+1)}$, is variable, while the mean is constant [MCQUARRIE98]. Once we get the error prediction interval, i.e. Eqn. (7), we can calculate the estimate (e.g., the effort) prediction interval as explained above. If we consider RE as an error measure, we have to apply Exn. (9).

$$\frac{O_{est}^{N+1}}{1 - RE} \quad . (9)$$

Because of the sign ± in Eqn. (7), RE in Exn. (9) gets two values, as well, i.e. RE = {μ_{DOWN}, μ_{UP}}. Then, the estimate prediction interval proposed by Jørgensen et al. is $[\, O_{est}^{N+1}/(1-\mu_{DOWN}); O_{est}^{N+1}/(1-\mu_{UP}) \,]$. Considering BRE, we have to apply Exn. (10) [JØRGENSEN03].

$$\begin{cases} \dfrac{O_{est}^{N+1}}{1 - BRE}, \text{ if } BRE \leq 0 \\ O_{est}^{N+1} \cdot (1 + BRE), \text{ otherwise.} \end{cases} \quad (10)$$

Where BRE = $\{\mu_{DOWN}, \mu_{UP}\}$ calculated through Eqn. (7) over D_{BRE}. Exn. (10) provides two values as shown above. For completeness, we reported the formula for AE as well, i.e. Exn. (11).

$$O_{est}^{N+1} + AE \quad . (11)$$

Exn. (11) provides two values, as well.

For simplicity of notation, we use $[\mu_{DOWN}; \mu_{UP}]$ to denote an error prediction interval and $[\, O_{est,DOWN}^{N+1} ; O_{est,UP}^{N+1} \,]$ to denote the corresponding estimate prediction interval. As a summary, the prediction interval technique for quantifying the uncertainty that we have just presented [JØRGENSEN03] is based on (1) selecting a relative error measure that can separate spread and bias, (2) gathering the estimation errors, (3) calculating a prediction interval of this error sample, and (4) obtaining the corresponding estimate prediction interval. This technique can be applied even if the probability distribution of the errors is not known and Eqn. (7) cannot be used for calculating the error prediction interval.

It is important noting that, so far we have considered two kinds of uncertainty measures. One based on an *error prediction interval* $[\mu_{DOWN}; \mu_{UP}]$ and the other based on an *estimate prediction interval* $[\, O_{est,DOWN}^{N+1} ; O_{est,UP}^{N+1} \,]$. They are strictly correlated because each can be obtained from the other. However, the estimate prediction interval, i.e., $[\, O_{est,DOWN}^{N+1} ; O_{est,UP}^{N+1} \,]$, may be

misleading when dealing with relative error measures such as RE or BRE and it does not allow figuring out the real improvement that an estimation model needs. We show this situation through an easy example.

To this end, assume that the next estimate is o_{est}^{N+1} = 45 Person-Months (PM) and the error prediction interval provided by Eqn. (7) is [μ_{DOWN}= - 0.4; μ_{UP} = -0.1]. The magnitude of this interval is |-0.4 – (-0.1)| = 0.3 PM. Applying Exn. (9), we can calculate the effort prediction interval $o_{est,DOWN}^{N+1}$ = 45 / (1-(-0.4)) = 45/1.4 = 32.1 PM and $o_{est,UP}^{N+1}$ = 45 / (1-(-0.1)) = 45/1.1 = 40.9 PM. Therefore, the effort prediction interval is [32.1; 40.9]. The magnitude of this interval is |40.9 – 32.1| = 8.8 PM. Consider now the error prediction interval [μ_{DOWN}= - 0.1; μ_{UP} = 0.2] and the same estimate as before, i.e. o_{est}^{N+1} = 45 PM. The magnitude of this interval is |-0.1 – 0.2| = 0.3 PM, the same as before. Applying Exn. (9), we can calculate the effort prediction interval $o_{est,DOWN}^{N+1}$ = 45 / (1-(-0.1)) = 45/1.1 = 40.9 PM and $o_{est,UP}^{N+1}$ = 45 / (1-(0.2)) = 45/0.8 = 56.3 PM. Therefore, the effort prediction interval is [40.9; 56.3]. The magnitude of this interval is |56.3 – 40.9| = 15.4 PM. Although the two error intervals had the same magnitude (i.e., 0.3) and the estimate was the same, they generated two different magnitude of estimate prediction intervals (i.e., 8.8 and 15.4). For this reason, when focusing on performance of the EM for improvement purposes, we have to consider error prediction intervals [μ_{DOWN}; μ_{UP}]. When focusing on the prediction activity, we use the estimate (effort) prediction interval [$o_{est,DOWN}^{N+1}$; $o_{est,UP}^{N+1}$].

It is important noting that, to apply Eqn. (7), we made some parametric assumptions on the error distribution, i.e., we assumed that, it has been sampled from a normal distribution (with unknown variance), where its bias and spread are well represented by the sample mean ($\overline{x}$) and sample standard deviation (s), respectively. The parametric assumptions support also the idea that mean and standard deviation are constant with respect to other context

variables, i.e. the variation of other context variables (e.g., size, complexity) do not affect bias and spread of the error.

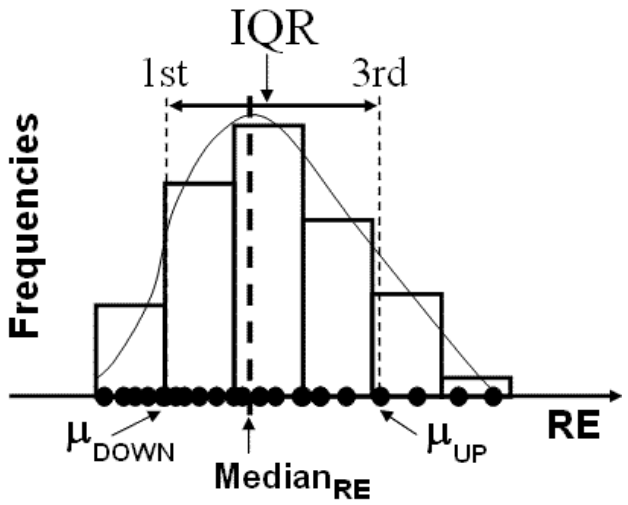

Fig. 4.Median and Interquartile range as non-parametric measures.

Consider now a more realistic situation, where we do not know the population from which the error sample has been sampled, e.g., there are few observations and/or the probability distribution is not Gaussian. This means that, if parametric assumptions do not hold (i.e., there is a significant number of outliers, the distribution is heavily skewed and/or multimodal), better measures of bias and spread are the median and the interquartile range (IQR), respectively (Fig. 4).

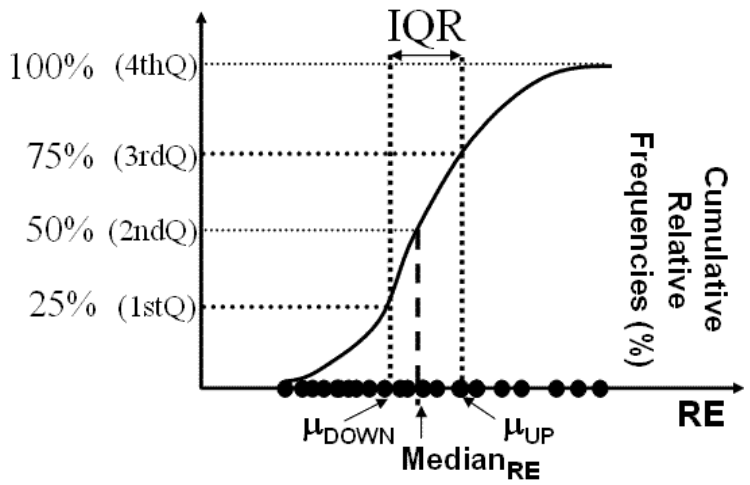

Fig. 5. Empirical Distribution Function (IQR).

The median is able to split up observations into two sets having the same number of elements. The IQR is based on considering the most probable range where the next error estimate will fall (Fig. 5) by taking out the outliers. The IQR can be derived by the empirical distribution (Fig. 5). Both the median and the IQR are less sensitive to outliers. In fact, the IQR provides a range corresponding to 50% of the observations (between the first and third quartile) of the empirical distribution. It would be a 50% prediction interval of the empirical distribution. Note that, IQR is used to build the box-and-whisker plot, which is a non-parametric tool.

IQR takes into account only 50% of the frequency, omitting the frequency in the upper and lower tails (first and forth quartile), to avoid that the resulting error range is affected by outliers, which are usually in the end-tails of the distribution. We take the 50% frequency (IQR) rather than a wider extent (e.g., 90% or 95%) to avoid outliers that can affect the calculation of the error prediction interval.

Some researches use a 90% or 95% error prediction interval instead of using the IQR [JØRGENSEN03] to calculate a non-parametric prediction

interval, when outliers are not assumed to heavily affect the model. For instance, to calculate a 90% prediction interval, one can consider the 5th and 95th percentile of the error empirical distribution, in the y-axis in Fig. 6, and select the corresponding error value in the x-axis.

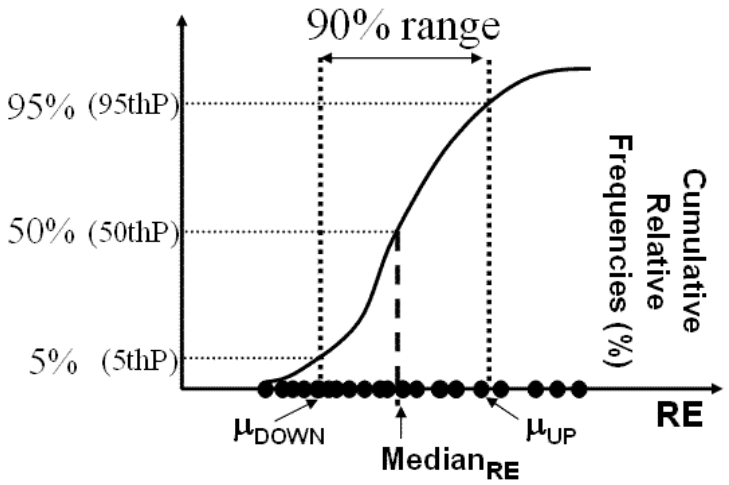

Fig. 6. Empirical Distribution Function (90% confidence).

A 90% prediction interval is wider than an IQR prediction interval; hence, is more likely to include the actual error. But, apart from outliers, the problem with considering a wider confidence is the *utility* of the resulting interval of error. If the error prediction interval is too wide, project managers may not use it in real estimation tasks. Utility is further explained below in this section. Choosing IQR, 90% or 95% prediction intervals, depends on the problem that we are dealing with and the assumptions made. If we know that, there are many outliers, it is better to consider the IQR. If the distribution has fewer outliers, a 90% (or 95%) prediction interval would be better. Since the magnitude of the confidence level (e.g., 50%, 90%, 95%) and risk are in inverse proportion, i.e., the confidence level grows as the risk decreases and vice versa, an organization aiming at earning a contract should decrease the

confidence by increasing the risk. Choosing a lower confidence level, the error prediction interval would get smaller and more useful (i.e., narrower). The decision to fix this threshold should be made at the highest management level of the organization based on the organization strategy [KITCHENHAM97]. The rate to which an organization earns contracts should be tracked for prospective evaluation [BASILI92B] and encapsulated into an experience package in the experience base of the organization [BASILI92B].

3.2.4 Uncertainty measure for a multivariate case

So far, we have dealt with quantifying the uncertainty and risk for a univariate case through a prediction interval-based approach taking into account the relative errors between estimates and actual values. Consider now a multivariate distribution where a variable (y) depends on other variables ($x_1, \ldots, x_Q$). We now deal with risk and uncertainty when using a linear-in-the-parameter estimation model as explained in Section 3.1 (e.g., polynomial models). The problem is to find the x-dependent mean of y and quantify its variance (uncertainty). If the regression model relies upon regression assumptions (Section 3.1), the prediction interval is readily available, Exn. (12) [RAO73], [WEISBERG85].

$$(y') \pm t_{1-\alpha/2}(N-Q-1) \cdot S\sqrt{1+x'^{T}(X^{T}X)^{-1}x'} \quad . (12)$$

Where:

- $y' = f_R(x',b) = b_0 + b_1x'_1 + \ldots + b_Qx'_Q = bx'$ is the expected value (mean) of the dependent variable (y) when the independent variables get a specific value $x' = (x'_1, \ldots, x'_Q)^T$, i.e., $(x_1 = x'_1 \ldots, x_Q = x'_Q)$
- $t_{1-\alpha/2}(N-Q-1)$ is a two tail t-value (Student's percentile) with N-Q-1

degrees of freedom

- X is the observation matrix of independent variables where the first column is composed of only 1s
- N is the number of observations
- Q is the number of the independent variables
- $S = \sqrt{\frac{Y^T (I - H) Y}{N - Q - 1}} = \sqrt{MSE}$ is an unbiased estimator of the standard deviation of the population (σ) [RAO73], [WEISBERG85]
- Y is the observation vector of the dependent variable
- I is the identity matrix
- $H = X(X^T X)^{-1} X^T Y$

MSE stands for Mean Squared Error.

The insight provided by Exn. (12) is that it calculates a two tailed (1-α)% confidence prediction interval of y'= βX' when the model is fed with the vector x'. The Exn. (12) calculation is the same as Eqn. (7) for a multi-linear regression function (i.e., a multivariate sample) under the regression assumptions reported in Section 3.1. This expression is quite similar to the prediction interval approach for a univariate case [JØRGENSEN03]. Both approaches use the variability of the error to calculate the estimate prediction interval. In particular, the variance of y per y = y' is assumed constant and the same as the variance of the residuals.

Once we calculate the error prediction interval, we can calculate the effort prediction interval through Exn. (9), (10), or (11). Those formulas can also be considered as a sort of correction used to make the estimates unbiased [KITCHENHAM97]. For instance, with respect to Exn. (9), if the error bias was zero (i.e., RE = 0), then Exn. (9) would be $O_{est}^{N+1} / (1 - 0) = O_{est}^{N+1}$, i.e. the estimate would not be corrected at all. Similar considerations can be made for Eqns. (10) and (11).

To calculate a two-tail (1-α)% confidence interval for the mean there is Exn. (13) [RAO73], [WEISBERG85].

$$(y') \pm t_{1-\alpha/2}(N-Q-1) \cdot S\sqrt{x'^{T}(X^{T}X)^{-1}x'} \qquad (13)$$

where, the symbols have the same meaning as the ones reported for Exn. (12). Note that, Exn. (13) provides a narrower interval than Exn. (12) because the mean is constant, while the next estimate is variable.

If the error prediction interval is too wide, it may depend not only on the error variance, but also on the effectiveness of the method of calculating the prediction interval. For instance, the Chebyshev's inequality method and the non-parametric ones, usually (Section 3.5) lead to intervals that are too wide, apart from the actual variance in the data. We need an effective method for estimating error prediction intervals as narrow as possible.

In Section 3.1, however, we showed that, to improve the correctness of the model we should use non-linear-in-the-parameter models because of their arbitrary flexibility. Therefore, to exploit the improvement coming from non-linear-in-the-parameter models, we need to calculate prediction intervals for those models, as well. If parametric assumptions hold, formulas for calculating prediction intervals are rightly available. For non-linear models, a slightly different formula can be applied [HUSMEIER04]. The formula for calculating PIs of MLFFNNs is $(y') \pm t_{1-\alpha/2}(N-K-1) \cdot S\sqrt{(1+g(x)^{T}(J^{T}J)^{-1}g(x))}$, where $y' = f_R(x',b)$ is the expected value (mean) of the dependent variable (y) when the independent variables get the next value $x' = (x'_1 \ldots x'_Q)^T$. $t_{1-\alpha/2}(N-K-1)$ is a two tail t-value (Student's percentile) with N-K-1 degrees of freedom. N is the number of observations, K is the number of parameters (note that $K > Q$).

$S = \sqrt{\frac{Y^T (I - H) Y}{N - Q - 1}} = \sqrt{MSE}$ is an unbiased estimator of the standard deviation of the population (σ). Y is the observation vector of the dependent variable. g(x) is a vector whose i-th element is the partial derivative $\partial f_R(x',b)/\partial b_i$ evaluated at its true value. J is a matrix whose ij-th element is the partial derivative $\partial f_R(x_i,b)/\partial b_j$. Matrix J can be calculated iteratively through the training procedure (i.e. Backpropagation) [BISHOP95A]. However, calculating prediction intervals for MLFFNNs when regression assumptions do not hold is still an open problem.

3.2.5 Model evaluation through uncertainty

To illustrate the utility concept of an error prediction interval, we use an easy example picked from a different environment than software engineering. The environment is a *package delivery service*. Assume that, a truck is moving from Seattle (WA, USA) to Boston (MA, USA) to deliver some packages. To know the actual position of the truck, the receiver, in Boston, uses a locator system, which automatically provides two points on the planned route where the truck is moving at a specific time. For instance, the locator system is able to provide [$Point_A$, $Point_B$, $Time_{A-B}$], which means that the truck may be in any point between $Point_A$ and $Point_B$ on the planned route at time $Time_{A-B}$. For instance, assume that the locator system provides [Minneapolis (MN, USA), Albany (NY, USA), 6:30am EST]. Although this piece of information is correct (i.e., the truck is actually between Minneapolis and Albany), it is useless because the receiver cannot know with a suitable approximation how far the truck is from the destination point. Since there are about 1,200 miles between Minneapolis and Albany and every point in the interval is equally probable, the information provided cannot be used for inferring the actual position with a suitable approximation. This situation is very close to the prediction interval situation here.

If the methodology for calculating prediction intervals provides too wide an interval, we may no longer use it as a reliable reference to quantify the actual limits between which the actual error will fall. This means that, quantifying risk and uncertainty through an error prediction interval is always possible, but the interval is useful for prediction only if its magnitude lies within specific thresholds.

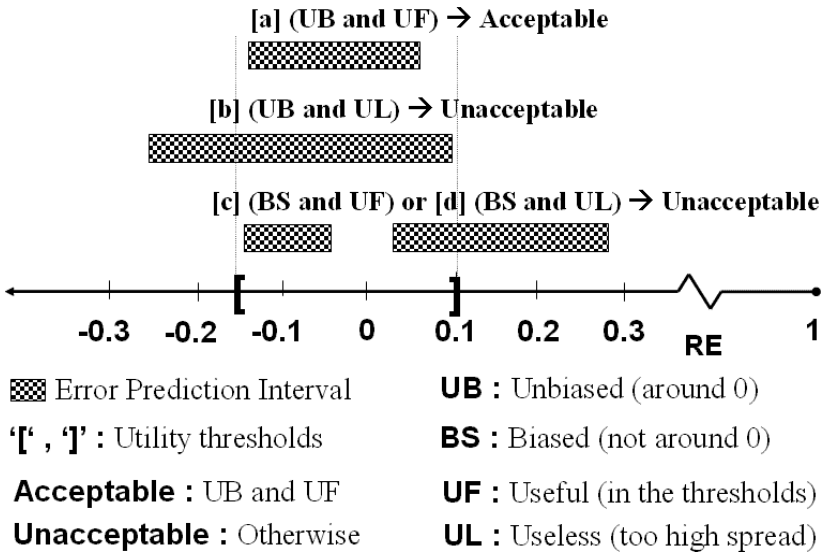

Fig. 7. Acceptability of error prediction intervals.

To explain the acceptability concept, we will use Fig. 7 to identify the spread and bias in relative error measures. From a mathematical point of view, the acceptability of a measure is affected by spread and bias in the error prediction interval. This information is very important to figure out whether and how to improve an estimation model and quantify its uncertainty. An error prediction interval is unbiased, if it includes zero ([a] and [b]). It is biased otherwise ([c] and [d]). A prediction interval is useful if it is within stated thresholds ([a] and [c]). It is useless otherwise ([b] and [d]). A prediction interval is acceptable if it is useful and unbiased at the same time

([a]). It is unacceptable otherwise ([b], [c], and [d]). Based on Fig. 7, we can infer whether the estimation model is prone to overestimate or underestimate. For instance, an estimation model overestimates when its prediction interval broaden towards negative values of the error (Fig. 7, [b]). If the prediction interval broadens towards positive values (Fig. 7, [d]), then the estimation model underestimates.

Therefore, an EM yielding an acceptable error prediction interval does not need any improvement. Conversely, an unacceptable interval requires some improvement. Thus, the meaning of an acceptable error prediction interval is not that, it cannot be improved, but it means that, based on the organization's goals and strategy, the improvement is not requested. Note that, we use the world "useless" not to say that we cannot take advantage of this piece of information for improvement (i.e., the fact that the error prediction interval is too wide). Actually, too wide an interval represents an unacceptable anomaly in the model and it needs to be improved, e.g., we are missing some values, variables or some measurement error occurred. This consideration is based upon the fact that, a model is a limited representation of reality hence can be always improved. For this reason, first we have to figure out whether the model yields acceptable estimates or not. If the model is assumed to provide unacceptable estimates, we may improve it only if we know the causes of its unacceptability. Fig. 7 can show these improvement needs. When dealing with biased intervals, the model is incorrect, e.g., it is missing some important variables or it is not flexible enough (i.e., not able to fit the data), and a different model should be used.

3.2.6 Similarity analysis and scope error

To get the estimate prediction interval from the error prediction interval, we apply one of the Eqns (9), (10), or (11) depending on the error measure that we choose. Applying Eqns (9), (10), or (11) means correcting, on the

average, the estimate from its bias, i.e., making the estimate unbiased. In this case, to improve the estimates, we mainly have to worry about situations [b] and [d] in Fig. 7 (i.e. too wide an error spread). However, if we aim at improving the estimation model itself, e.g., making the error prediction interval acceptable (unbiased and useful), we have to deal with every situation in Fig. 7, i.e., shrinking the error prediction intervals and removing the error bias.

Recognizing bias is quite easy (i.e., checking whether the error interval includes zero or not), while fixing the utility is more complicated. The utility (i.e., the threshold in Fig. 7) is a subjective characteristic that varies from organization to organization based on the required reliability, characteristics of the market (competition), and strategies pursued from the organization's goals. Utility thresholds should not vary with the size of the estimate when dealing with relative error measures, i.e., the severity of an error should be the same for any projects being estimated when considering relative errors. We will see that this is not always true. The analysis in Fig. 7 is the basis of the strategy that we will define in Section 6.1 for evaluating the risk and improving EMs over time. It is worth noting that, Fig. 7 is incomplete because it does not deal with the risky situation described by Kitchenham et al. [KITCHENHAM97] called *scope error*. It may happen when the EM is requested to provide estimates on data never observed before, i.e. the EM has not been calibrated on data close to the project data being estimated. This situation should be considered from project managers as a severe (risky) one because, if we never dealt with the similar projects as the ones being estimated, the predictive capabilities of the EM may be unpredictable. To this end Kitchenham et al. [KITCHENHAM97] propose applying the portfolio concept (Section 5), where the organization deals only with specific segments of the market, e.g., medium size software systems for banking environment.

From a theoretical point of view, applying the portfolio concept deals with scope error because the projects being estimated are similar to the past projects, so the expected accuracy of the estimation model in estimating the new projects would be known. However, the portfolio strategy is based on the assumptions that (1) the organization has sufficient information to execute the grouping task and build the portfolio and (2) that there are a sufficient number of past projects in the portfolio. Modern organizations may have to deal with projects having different characteristics from their experience. Our approach allows an organization to take advantage of every piece of information on past projects for prediction and improvement. Organizations need to group past data into sets of similar projects. We call this grouping as *similarity analysis*, i.e. quantifying the degree of similarity between the project being estimated and the past data used for calibrating the considered estimation model. Note that, Jørgensen et al. [JØRGENSEN03] use the name "similarity analysis" in a slightly different way. They group the past observations (i.e., projects) having the same expected degree of estimation uncertainty as "similar" the project being estimated. Then, they assume to be able to do that. The similarity that we are referring to is more properly about grouping observations that are of similar type (i.e., they have close/same values for the same variables) and consequently getting the expected uncertainty from each group. We explain this concept through an example (Table 2).

TABLE 2

SIMILARITY ANALYSIS – PAST OBSERVATIONS (ACT)

Similar Project Set	Effort	KSLOC	Cplx	Error PI
A	70-80	20-30	High	[-0.30;0.20]
B	50-60	60-70	Low	[-0.10;0.25]

In Table 2, there are two project sets having their own error prediction interval (Error PI). In Table 3, there are three estimated projects, i.e. they are described by estimated values. We are interested in knowing, for each project, the prediction interval where the error will fall. If our estimates are correct (e.g., estimates for KLOC, Cplx, and Effort), P1 can be classified as belonging to set A, therefore the expected error prediction interval would be the same as [-0.30; 0.20]. Project P2 has some characteristic close to set A (i.e., KSLOC = 20) and some characteristic close to set B (i.e., Cplx = 'Low'). How much is "close"? This uncertainty situation on project P2 is workable.

TABLE 3

SIMILARITY ANALYSIS – PROJECT BEING ESTIMATED (EST)

Projects	Est Effort	Est KSLOC	Est Cplx	Similar Project Set	Est Error PI
P1	70	20	High	A	[-0.30;0.20]
P2	15	20	Low	A or B ?	[-0.30;0.25] ?
P3	100	40	Very High	?	?

For instance, we can consider an error prediction interval as the union of the A and B prediction. The union of two distinct ordered sets is wider than each individual set. As a result, the estimation uncertainty of the estimate of project P2 grows.

Consider project P3. This represents a different situation from the previous ones. We do not have any characteristic similar to set A and B even though we may argue that KLOC = 40 is in between KLOCs of set A and B, and Cplx = "Very High" is closer to "High" than "Low". Then, estimating project

P3 with the considered EM is additionally risky. That is, this situation is in addition to the ones in Fig. 7 and refers to a possible scope error.

Therefore, if we used the EM for estimating P3 we may have some unpredictable spread error. From a practical point of view, project managers would have to recognize whether a scope error could happen or not and consequently evaluate risk and uncertainty for situations similar to the ones in Fig. 7. To this end, the estimate of project P3 should be considered as more risky than the estimate of project P2 even though we may not quantify the error prediction interval for P3. Note that, the scope error does not affect the EM parameters. It is about using the EM improperly. The scope error is a risky situation because we do not know the error prediction interval for similar projects even though we may state some rough limits (not very accurate); therefore, the risky situation arises from the fact that, we use an EM without knowing its actual risk. Recognizing the scope error is not so easy when dealing with a number of variables. As an example, consider the situation where instead of having only one variable as in Table 2 (i.e., Cplx) in addition to Effort and KSLOC, we had 15 variables, as the COCOMO model, and for each of them, there were, for instance, five values. Then, there would be 5^{15} = 30,517,578,125 different subsets to be considered in addition to the ratio scale variables. Note that, many of those subsets may have the same uncertainty, so the number of distinct uncertainty intervals may be much less than 5^{15}. However, the huge number of elements shows that recognizing whether or not a scope error occurs, project managers need evaluation systems, which are able to provide a similarity measure automatically; otherwise they will not be able to figure out the real prediction risk.

3.2.7 Assumption error and risk exposure

So far, we have defined mathematically the risk in applying an estimation model $f_R(x,b)$ given x = x' as the expected variability (i.e., prediction interval) of $f_R(x=x',b)$. Note that, x' is an estimated input as shown in Fig. 3 (time T). This kind of uncertainty is all about the correctness of b (i.e., the estimated parameters of β). It is worth noting that we have another uncertainty source: the correctness of the assumed inputs fed into the EM. This kind of uncertainty, called *Assumption error* [KITCHENHAM97], comes from wrongly assumed inputs. That is, the assumption error is one that we get when we assume we know the project characteristics (i.e., the estimation model inputs), when the actual values are different. The estimation model provides some error, not because of the model, but the inputs. For this reason, before observing the actual values of the project being estimated, we can only get a *conditional uncertainty*, i.e., if we observed x' (i.e., the actual inputs were x'), then the error prediction interval would be $[\mu_{DOWN}; \mu_{UP}]$ and, applying Eqns (9), (10), or (11), the estimation prediction interval would be $[\, O^{N+1}_{est,DOWN} ; O^{N+1}_{est,UP} \,]$. This kind of conditional uncertainty is a sort of what-if analysis. What would be the estimate prediction interval if the input values were x'? Therefore, similar to the scope error, the assumption error does not affect the estimation model parameters. It is about the uncertainty in providing the right values to the estimation model.

We can now define *risk exposure* (Re) as a combination of impact (Im) and probability (Pr), i.e. Re = Im x Pr, [BOEHM81], where Im is the expected estimate error and Pr is the probability that Im happens. The impact is the deviation between the chosen estimate and the expected endpoints of the estimate prediction interval (horizontal rectangles no. 1, 2, 3 in Fig. 8). As an example, assume that there are three possible input sets (γ', η', and φ') where each set is a vector of values. Then we can choose each of those sets as an input set of the EM with different probability, e.g. x = γ' with probability

Pr(γ') = 45%, x = η' with probability Pr(η') = 35%, and x = φ' with probability Pr(φ') = 20%, where γ', η', and φ' are vectors of estimated input values (Fig. 8).

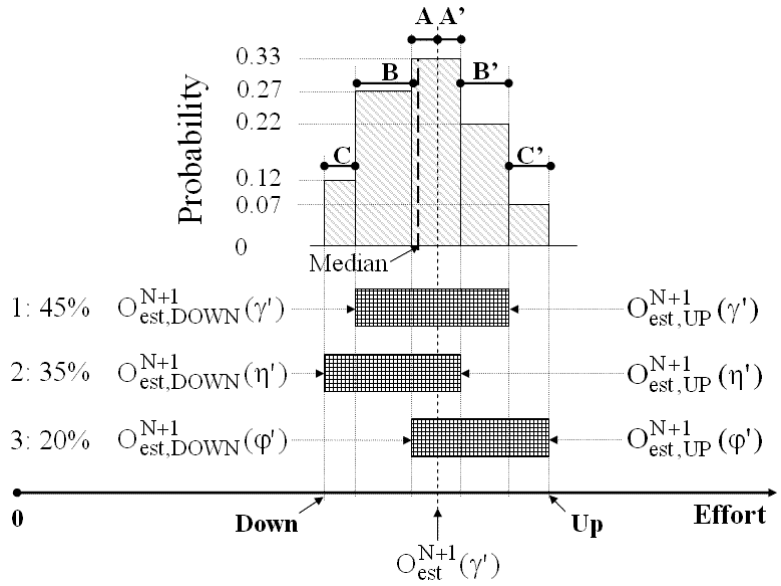

Fig. 8. Risk exposure brought about by the assumption error.

If we are not able to state the probability for each individual set (e.g., we do not have enough a priori information), we have to assign an equal probability to each individual set, e.g., 1/3 = 0.3333 → 33.33%.

If we feed the three input sets into $f_R(x, b)$, we obtain estimates $y_{\gamma'} = O_{est,}^{N+1}(\gamma') = f_R(\gamma', b)$, $y_{\eta'} = O_{est,}^{N+1}(\eta') = f_R(\eta', b)$, and $y_{\varphi'} = O_{est,}^{N+1}(\varphi') = f_R(\varphi', b)$, respectively. For each input set, we can calculate the error prediction interval $[\mu_{DOWN};\mu_{UP}]$ and the effort prediction interval $[\ O_{est,DOWN}^{N+1}\ ;\ O_{est,UP}^{N+1}\ ]$ (i.e., intervals 1, 2, and 3 in Fig. 8). Assume now that we choose $O_{est,}^{N+1}(\gamma')$ as the "ultimate" estimate for the project because it is the most likely estimated value with the probability of 45%. Based on this choice (i.e., x' = γ'), what would be the expected risk (i.e., what would be the expected estimate error

because of the chosen value $O_{est,}^{N+1}(\gamma')$)? To what extent is the error that we get when assuming $O_{est,}^{N+1}(\gamma')$ as the final estimate of the project? At first look, for overestimates, the expected risk (i.e., an impact) would seem $O_{est}^{N+1}(\gamma') - O_{est,DOWN}^{N+1}(\gamma')$ = (A+B) and, for underestimates, $O_{est,\ UP}^{N+1}(\gamma') - O_{est}^{N+1}(\gamma')$ = (A'+B'). Actually, this calculation is incorrect because we did not consider the uncertainty arising from the assumption error.

As Fig. 8 shows, we should take (A+B+C) as the expected impact (Im) for overestimates and (A'+B'+C') as the expected impact for underestimates, i.e. the uncertainty is wider than the one considered initially. As an example, to calculate the risk exposure (Re), if we chose $O_{est}^{N+1}(\gamma')$ as a project estimate (Fig. 8), then the risk exposure for an underestimate would be, A*0.33 + B*0.27 + C*0.12 and, for an overestimate, would be A'*0.33 + B'*0.22 + C'*0.07. Based on the distribution in Fig. 8, we may apply different strategies for choosing the "ultimate" estimate of the project. For instance, if we chose the value corresponding to the median (Fig. 8), we may have a better central tendency than $O_{est}^{N+1}(\gamma')$. Or, since an underestimate is usually considered more risky than an overestimate [MCCONNELL06], – because the former may lead to losing money, while the latter may lead to not gaining money – another strategy for an organization trying to minimize the risk of losing money (i.e., of underestimate) may choose the endpoint "Up" as the "ultimate" estimate of the project (i.e., $O_{est\ UP}^{N+1}(\varphi')$).

3.2.8 Risk mitigation strategy

Generally, the Re is expressed by an ordinal scale [HIGUERA96]. This scale can be odd (e.g., "low", "nominal", and "high") or even (e.g., "very low", "low", "high", and "very high"). We prefer using an even ordinal scale because knowing that an estimation model provides a "nominal" risk exposure is useless and even confusing. In other words, we prefer to avoid

the risk exposure may concentrating on the median value causing further uncertainty.

Apart from the scope error that we cannot quantify in terms of the error prediction interval (even though we may roughly estimate it), the procedure that we propose for estimating risk and uncertainty when assumption error occurs is:

1. Calculate the error prediction interval [μ_{DOWN}; μ_{UP}] for each input set
2. Calculate the estimate prediction interval [$O^{N+1}_{est,DOWN}$; $O^{N+1}_{est,UP}$] for each input set
3. State probabilities for each input set or assign equal probability (if we do not have enough a priori information to assign different probabilities)
4. Build the diagram in Fig. 8
5. State a risk minimization policy
6. Based on the stated policy in step (5), choose the "ultimate" estimate of the project.

This procedure can be applied independently from the methodology that we choose for calculating estimate prediction intervals. Therefore, for instance, it can be used along with the Jørgensen's prediction interval methodology [JØRGENSEN03], Bootstrap methods [ANGELIS00], Regression methods [JØRGENSEN04A], and the one that we propose in this work.

3.3 Measurement and improvement paradigms

In this research, we mainly deal with building evaluation models (e.g., evaluation formulas, algorithms, and criteria) based on the GQM approach [BASILI94A], [LINDVALL05] in the context of learning organizations [BASILI92B]. However, our proposal is general enough for any evaluation and measurement environment such as Practical Software Measurement [MCGARRY02] and Goal-Driven Measurement [PARK96].

GQM is an approach for building, tailoring, and selecting models and metrics for addressing specific goals for any software project in an organization (e.g., stating goals on software processes, products, and quality properties). Goals can be defined for any object, for a variety of reasons, with respect to these quality attributes, from various points of view, relative to any environment. For instance, a typical GQM template is the *study object*, the *purpose*, the *quality focus*, the *point of view*, and the *context*. Based on top-down flows and starting from such a goal, we can generate *Questions*, which in turn generate *Metrics*. It is also possible to have intermediate goals (sub-goals) [LINDVALL05], which help better understand the measurement problem and control its complexity. Based on questions such as "*Is the performance of the object (e.g., process, methodology, estimation model etc.) better?*" we build evaluation models where the current performance can be effectively compared to the past ones. In order to do so, a baseline is built (e.g., based on the history and experimental data).

3.4 Error Taxonomy Summary

As we have already illustrated, parametric estimation models can be affected by specific errors. In this section, we only summarize them for better comprehension (Fig. 9).

When applying an estimation model, we have different kinds of uncertain and risk (see Section 3.4):

(1) *Model error* affects the correctness of the model parameters. It can be inherent to (a) missing variables, (b) redundant variables, (c) unsuitable model complexity, (d) heteroscedasticity, (e) spurious relationships, (f) measurement error, or (g) an iterative calibration procedure. The model parameters thus calculated would be biased. It expresses our uncertainty on the model.

(2) *Scope error* does not affect the correctness of the model parameters. It has to be supported by automatic software tools that perform a similarity analysis between the project being estimated and the past projects used for calibrating the estimation model. It expresses our uncertainty on the unsuitability of using the model for predicting quantities that are out of the model scope.

(3) *Assumption error* does not affect the correctness of the model parameters. It requires knowing (or being able to estimate) the probability that the estimation model inputs are correct. For this kind of error, we can define a risk exposure, i.e. Re = Im x Pr, where Im is an interval as shown in Fig. 8 and refers to underestimates and overestimates, separately. It expresses our uncertainty about the model inputs.

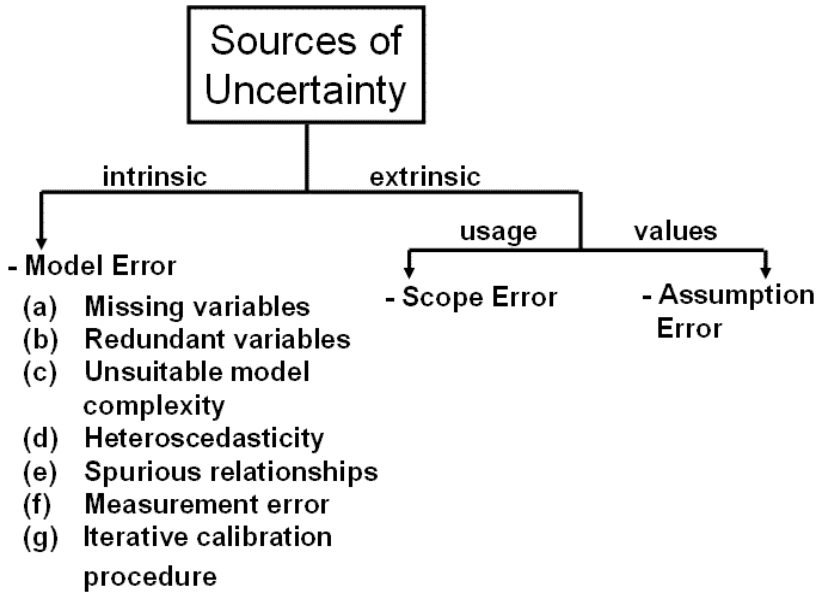

Fig. 9. Error taxonomy.

3.5 Related Work on Prediction Intervals

In the literature, there exist many ways of calculating an effort prediction interval. Angelis and Stamelios [ANGELIS00] calculate prediction intervals of regression model and resampling (bootstrap) analogy-based models. Jørgensen et al. [JØRGENSEN03] calculate the prediction interval through the empirical distribution as explained above. Moreover, Jørgensen [JØRGENSEN04A] uses a regression model to predict the most likely estimation accuracy, directly. There also exist human-based judgment methods [JØRGENSEN03], where the effort prediction interval is derived by error distributions coming from human judgment. Another approach is provided in [CHULANI99], where the uncertainty is evaluated by a Bayesian analysis related to the project schedule. Similarly, to the latter, Fenton [FENTON99] and Pendharkar et al. [PENDHARKAR05] use Bayesian network for evaluating the estimation uncertainty. Further information on the uncertainty evaluation can be found in [MYRTVEIT99].

A prediction interval (PI) consists of a minimum, a maximum value, and a confidence level [BAIN92]. For instance, we may find the upper and lower limits of an estimate such that 90% of observations will fall within the interval (lower and upper bounds). Such an interval can help manage and describe uncertainty [JØRGENSEN03]. In Section 3.2, we have already shown the way to calculate prediction intervals both using t-Student's percentiles and the empirical distribution. There is another parametric approach, which does not assume that the accuracy is normally distributed. This approach uses the Chebyshev's inequality, which works for any distribution. Chebyshev's inequality is based on the following equation:

$$\Pr\left\{\overline{X} - k \cdot S < y < \overline{X} + k \cdot S\right\} \geq 1 - \frac{1}{k^2} \qquad (14)$$

Where $\bar{x}$ is the sample mean value, S is the sample standard deviation, and *k* is a constant ≥ 0. For instance, if k = 2 standard deviations, then the probability that the actual value is within interval [$\bar{x}$ -2S, $\bar{x}$ +2S] is at least 75% ($=1-1/2^2$) [JØRGENSEN03]. This kind of analysis, sometimes, cannot be performed in real cases because it provides too large bounds [BAIN92]. Such a technique may be improved if we could assume that the probability distribution was symmetric [JØRGENSEN03]. Another method for parametric and non-parametric distributions is based on the minimum and maximum limits of the sample. So, if we have N similar projects, then PI = [min(sample), max(sample), (n–1)/(n+1)], where the term "(n –1)/(n+1)" is the confidence level of PI [VARDEMAN92].

3.6 Artificial Neural Networks for regression problems

In this section we refer to feed-forward multi-layer artificial neural networks with supervised training (e.g., Backpropagation [RUMELHART86]), optimization techniques (e.g., Levenberg-Marquardt [HAGAN94], regularization by weight decay [DREYFUS05], early stopping [DREYFUS05]) and feature selection [STOPPIGLIA03], [KOHAVI97], [AHA96], [BRIAND92], [KIRSOPP02A], [JOHN94] (e.g., principal component analysis [JOLLIFE96], [NEUMANN02], curvilinear component analysis [RUMELHART86], [BISHOP95A], leave-one-out [DREYFUS05], and cross-validation [STONE74]). Multi-layer feed-forward artificial neural networks are also called (multi-layer) perceptrons.

In this research, we consider perceptrons with only one hidden layer. This is not a limitation because it is possible to prove that the capability of such a network is the same as those with more than one hidden layer [DREYFUS05, pp. 13].

For instance, let us consider the following 3-D space equation (15):

$$g(X,Y) = c_0 + c_1 \cdot \tanh(c_2 + c_3 X + c_4 Y) + c_5 \cdot \tanh(c_6 + c_7 X + c_8 Y) \quad (15)$$

Where, X and Y are the independent variables, $\{c_0 \ldots c_8\}$ is a set of 9 parameters of g(*X*, *Y*) and *tanh* is the hyperbolic tangent function. Compare Eqn. (15) with Eqn. (16), also a 3-D space equation.

$$g(X,Y) = c_0 + c_1 X + c_2 Y + c_3 X^2 + c_4 XY + c_5 Y^2 \quad (16)$$

Both are mathematical models that can be used to approximate a generic function g(*X*, *Y*) from a number of available observations. We can use either function to calculate the regression function [BISHOP95A, pp.201-203] and based on the available observations, we would estimate parameter values that minimize the cost function.

Equation (15) is a multi-layer feed-forward artificial neural network perceptron (Fig. 10) and Eqn. (16) is a polynomial function. Note that, Eqn. (16) can be transformed into a linear function, Exn. (17).

$$c_0 + c_1 X_1 + c_2 X_2 + c_3 X_3 + c_4 X_4 + c_5 X_5 \quad (17)$$

When $X = X_1$, $Y = X_2$, $X^2 = X_3$, $X*Y = X_4$, and $Y^2 = X_5$. Both Eqn. (15) and Eqn. (16) can be used as model regression functions. Let us consider their similarities and differences more closely.

The first similarity between (15) and (16) is that they are the summation of several functions. In particular, Eqn. (15) sums 3 functions and Eqn. (16) sums 6 functions. But, they use different kinds of functions; the functions in Eqn. (15) are not linear with respect to their parameters, while Eqn. (16) is composed of functions that are linear with respect to their parameters [DREYFUS05, pp. 13].

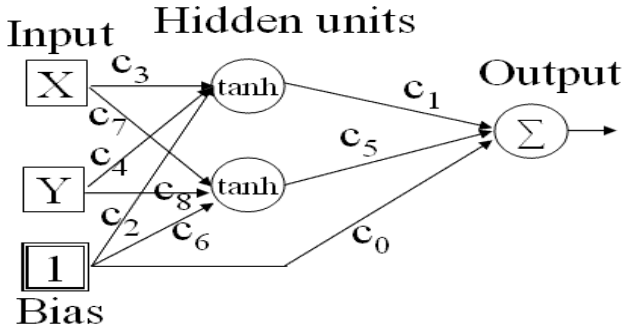

Fig. 10. Graphical representation of Eqn. (15), a multi-layer feed-forward artificial neural network with two input units, one hidden layer of sigmoid functions (tanh), and a bias unit. The output is calculated by multiplying the input units by the corresponding weights (parameters). Note that, the bias term is equivalent to an intercept in a traditional regression model [BISHOP95A].

The Eqn. (16) sums functions that have a fixed shape, while Eqn. (15) combines functions that have shapes adjustable through a number of parameters (e.g., c_2, c_3, c_4). This characteristic, for Eqn. (15), allows us to get more degrees of freedom using a smaller number of functions rather than a smaller number of parameters. It has been proven that, if the model is non-linear with respect to its parameters, it is more *parsimonious* [BARRON93, cap. 1], [DREYFUS95, pp. 13-14]. This means that, in order to approximate a function, if we use a model that is nonlinear with respect its parameters, we need a smaller number of measurements to get the same accuracy or we can get better results using the same number of measurements.

Moreover, it can be shown that the number of required parameters to perform an approximation with a given accuracy varies exponentially with the number of variables for models that are linear with respect to their parameters, while it increases linearly for models, which are nonlinear with respect to their parameters [BARRON93, cap. 1], [DREYFUS05, pp. 13-14]. These two properties of parsimony explain why nonlinear-to-their-parameter models should be preferred to linear-to-their-parameter models, if we need to

get better accuracy. Instances of linear-to-their-parameter models are all kinds of Polynomial Functions, Radial Basis Function with fixed centers and widths, and Wavelet Networks, while nonlinear-to-the-parameter models are multi-layer feed-forward perceptrons and radial basis functions with no fixed centers and widths [DREYFUS05].

Parsimony apart, the main difference is that a regression function calculated by neural networks is actually a regression function conditioned to the observed sample. Operatively, this means that we can perform accurate predictions within intervals where observations are available. If observations are not available, generalization capabilities decrease. Conversely, artificial neural networks do not need to make assumptions on model linearity (or log-linearity) and deterministic input variables as explained in Sections 3.1 and 3.2. As a result, neural networks provide better results than linear methods where sampled data is available (cited parsimony). Therefore, neural networks are particularly useful in dealing with problems when we do not know a priori the relationship between Inputs and the Output.

Another difference is the way they calculate parameters. For models that are linear with respect their parameters (e.g., polynomial functions), the ordinary least squares methods can be used, even though the resulting models are not parsimonious [DREYFUS05, p. 30]. For models that are nonlinear with respect their parameters (e.g., multi-layer artificial neural networks) there is no closed solution and an iterative method has to be applied (Section 3.1). The most used technique is Backpropagation [RUMELHART86]. It is an iterative method based on calculating gradients. The gradient of the cost function is calculated for each step and is used to update the parameters found in the previous step. The algorithm stops when satisfactory conditions have been met [BISHOP95A].

Consider Eqn. (16). If we need to increase the fit to the observations, we normally increase the number of polynomial terms (its degree). The

corresponding increase for Eqn. (15) is to add hidden units (e.g., 3, 4, or more). In both cases, we are increasing the *complexity of the model*, but in non-linear-in-the-parameter models, we keep constant the number of adding functions (parsimony).

For further explanations about Backpropagation and optimization, see the books and papers of Bishop [BISHOP95A], Dreyfus et al. [DREYFUS05], and Guyon et al. [GUYON05]. The latter also describes coding issues with common programming languages. The literature reports on many applications of neural networks used for predicting software cost [KHOSHGOFTAAR97], [SRINIVASAN95], [WITTIG94], [FINNIE97], [SAMSON97], [MAIR00], [MYRTVEIT04], [JØRGENSEN95].

3.7 Bayesian classification through neural networks

Neural networks can use the Bayesian theorem for classification problems.

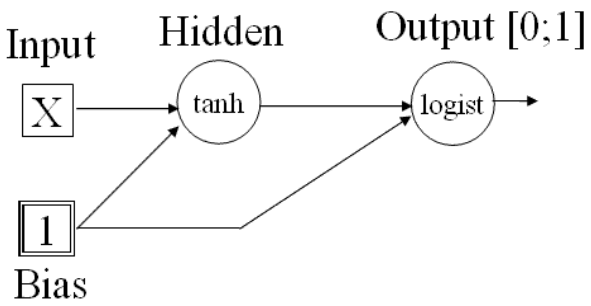

Fig. 11. Neural network for discrimination problems. The network has just one input variable (feature) along with a bias variable.

This problem is slightly different from the regression problem. A Neural network for discrimination problems looks like the one shown in Fig. 11. In particular, the network has just one input variable and one bias unit. The output is a real number in [0;1]. We can use the logistic function to get such a [0;1] interval [RUMELHART86]. The number of hidden units represents the

complexity of the network (Section 3.6). Consider the following analogy. Based upon prior experimental evidence, we hypothesize that healthy people (e.g., those with lower disease incidence) keep the ratio between their weight and height within a specific threshold (unknown). The classification problem is to figure out whether people never treated before belong to the healthy people's class or not (e.g. they will develop some diseases or risk developing some diseases). Let the X-axis in Fig. 12 represent the ratio between weight and height, i.e., $X_i = Weight_i/Height_i$ (where X_i represents i-th person). Class A is represented by 1 and class B by 0, i.e. $Pr(y=1|X) = Pr(A|X) = 1 - Pr(y=0|X)$, where y is the classification function in Fig. 12. Triangles in Fig. 12 are some observations of X for non-healthy people, and circles are some observations of X for non-healthy people.

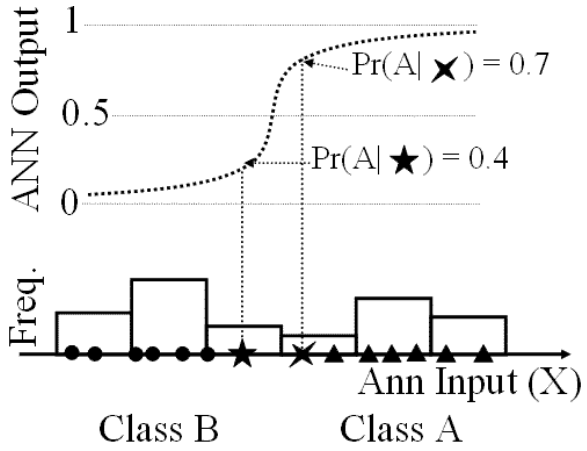

Fig. 12. A Discrimination problem. The X-axis represents the ratio of a person's weight and height. The output refers to the network in Fig. 11.

The problem is then classifying whether new observations belong to class A or class B. Star and cross points represent two new observations in Fig. 12. In order to deal with this classification problem, we use the neural network in Fig. 11, where the input variable takes values from the X-axis and the output represents the decision made (e.g., i-th person belongs to class A). For

instance, if the output is within [0.5;1], we classify the person as belonging to class A. If the output is within [0;0.5[, we classify the person as not belonging to class A (i.e. belongs to class B).

Note that, this problem is a two-class discrimination problem since if i-th person does not belong to class A, we can infer that he/she belongs to class B. We ask our neural network to separate class A from class B and provide a reasonable classification (similarity analysis). Note that, the neural network classifies new items based on previous observations. This problem is simple as it deals with only one input variable and does not require the use of an artificial neural network. The classification problem is about performing a similarity analysis between the people's characteristics never observed before and the ones that we observed for building the network. This analysis has to take into account the sampling data as well.

The classification paradigm that we use is the Bayesian approach since it is able to combine both aspects (a priori belief and sampling data information). The Bayesian approach is quite different from the classical one (frequentist). The former considers both "a priori" belief (what we believe about the object, healthy/non-healthy) and the sampling data information, while the latter is exclusively based on the sampling data information.

Going back to the proposed example, if there are secondary factors that affect the classification decision, such as gender, age, diet, and whether they are a smoker, the problem becomes more complex because five features (input variables) affect the result simultaneously. Mathematically, the classification problem shown above is equivalent to considering a random variable Γ, which is a function of a vector of features X (input variables). Such a random variable is equal to 1 when the input (called pattern) belongs to A and 0 when the input belongs to B [DREYFUS05]. The literature reports many applications of neural networks used for discrimination problems and risk assessment in software engineering such as the one that we have reported

above [KHOSHGOFTAAR95], [LANUBILE97], [BINGUS96], [KARUNANITHI92], [KHOSHGOFTAAR94].

The most meaningful question is then "what does this index (network's output) represent?" We show the proof of the following result [DREYFUS05], [BISHOP95A]: the regression function of the random variable Γ is the posterior probability of class A.

The regression function y (x) of variable Γ is the expected value of Γ given x, therefore y(x) = E(Γ| x). Moreover,

$$E(\Gamma| x) = Pr(\Gamma = 1| x) \times 1 + Pr(\Gamma = 0) \times 0 = Pr(\Gamma = 1| x) \quad (18)$$

which proves the result [DREYFUS05]. In Section 3.6 we showed that, based on Backpropagation algorithm, neural networks could estimate non-linear regression functions from observed samples. Then, if we apply such an algorithm to our network for classification, we can estimate the regression function of Γ and consequently get posterior probabilities. This is a very important result because the interpretation of the mathematical result is that, we can use neural networks to estimate the probability that, given a new input, it belongs to class A (probability of A given x).

When applying linear and non-linear models to classification problems (e.g. logistic regression), we may be interested in evaluating prediction intervals as well. Formulas to evaluate prediction intervals can be derived from the same asymptotic theory shown in Section 3.2 [HUSMEIER04], [HWANG97].

However, it is important noting that, in classification problems where there is a binary decision to make (0 or 1), instead of minimizing a sum-of-squares function, we can consider a cross-entropy error function, which is more appropriate for classification. It happens because the cross-entropy error function is based on a Binomial error distribution, which is more suitable for

classification than a Gaussian distribution on which a sum-of-squares error function is based [BISHOP95A].

3.8 Bayes' Theorem

The posterior probability is the one that Bayes' theorem defines. In particular, Eqn. (19) is the formula of such a theorem.

$$\Pr(A \mid x) = \frac{p_X(x \mid A)\Pr(A)}{p_X(x \mid A)\Pr(A) + p_X(x \mid B)\Pr(B)} \qquad (19)$$

where Pr(A|x) is the posterior probability that an observed x belongs to A, i.e., given a particular project, the probability that it belongs to A; Pr(A) represents the prior probability of class A, i.e. the percent of prior observations that belong to A; $p_X(x|A)$ represents the prior probability that any x belongs to A knowing A, i.e., the likelihood that when we observe any project (x), it belongs to A. Similar definitions hold for B. Note that Pr(B|x) = 1 – Pr(A|x) hence we can directly get the complementary probability. Calculating Pr(A|x) or Pr(B|x) depends on the problem that we are dealing with. Our focus is on "unwanted events" (risk and uncertainty) since we calculate Pr(A|x), where, concerning the example considered above, A represents the non-healthy people's class.

Based on the Bayesian theorem and Eqn. (18), the dotted sigmoidal line in Fig. 12 is actually a posterior probability density that any point expressed by X belongs to class A (y=1).

Theoretically, Bayes' theorem is a very important result, but it is difficult to apply in real cases [AITKEN95, pp 36-41]. The problem is that, we could, somehow, estimate P(A) and P(B) (e.g., based on prior information), but we could not estimate $p_X(x|A)$ and $p_X(x|B)$, we need to know such conditional probabilities for applying the Bayesian theorem.

Based on Eqn. (18) and (19), these considerations explain the reason why neural networks are so important for Bayesian classification. Neural networks (or any binomial model such as a logistic regression model) are able to estimate empirically the posterior probability even if we do not know conditional probabilities $p_X(x|A)$ and $p_X(x|B)$. So, mathematically, we can use a neural network to estimate the conditional probability density; that is, given the features of a new item, we can know it belongs to class A without knowing the inverse conditional probabilities $p_X(x|A)$ and $p_X(x|B)$. Therefore, we can use multi-layer feed-forward neural networks for estimating posterior probability density functions and using them for discrimination.

Let us go back to the classification example above and consider the following questions: "what is the probability that i-th person is non-healthy, given features X_i? In other words, what is the risk that i-th person is non-healthy knowing his/her features? If we consider "non-healthy" as an event, what is the probability of the (unwanted) event "non-healthy" given i-th person's features?

Based on the posterior probability density function in Fig. 12, we can deal with the inverse problem. The question to answer would be what is the X range that corresponds to a 95% of posterior probability? In other words, we aim at using neural networks for empirically (i.e. based on observations) building posterior probability density functions and exploiting them to infer information about the population. Mathematically we have

$$Pr_{\Theta|X=x}(L<\Theta<U) = 1 - \alpha \quad . \qquad (20)$$

Where $Pr_{\Theta|X=x}$ expresses the probability of Θ conditioned to the observed values of $X = x$, L is the lower bound, and U is the upper bound of the credible interval, where a credible interval is the homologous of the confidence interval for the Bayesian approach. Θ is the unknown parameter

of the population, $1 - \alpha$ is the confidence that the real value of the parameter Θ will fall between L and U. From a practical point of view, to get a 95% (credible) interval empirically (Fig. 12), we can fix a posterior probability of 0.05 and 0.95 on the vertical axis (ANN output) and select the corresponding values (i.e. L and U) on the X-axis.

As we have explained in Section 3.1, the Bayesian approach considers the uncertainty caused by the model parameter calculations. MacKey [MACKEY91] estimated the uncertainty by integrating it over many models built by simulation techniques. The problem with the MacKey's approach is the difficulty of calculating the integral of each piece of uncertainty. Because of the approximation in evaluating it and the normality assumptions on the models, we believe that the MacKey's approach is not appropriate for software engineering practitioners. Later in the paper, we will present an empirical procedure without making any specific assumptions. We will simplify the calculation of the endpoints L and U by applying an empirical approach like Jørgensen's empirical approach for calculating prediction intervals [JØRGENSEN03] and avoiding as many assumptions as possible.

Chapter 4

Problem definition

4 The problem

The aim of this paper is to define a way to improve parametric estimation models with respect to their prediction and uncertainty (inference). The main problem of concern in this work is that, if we continue to violate assumptions on which the model is based, pretending that everything is fine, we will have neither accurate estimates nor suitable model improvements. Therefore, we argue that to improve our modeling capability we have to recognize violations and deal with their consequences (e.g. making corrections to the bias and spread of the model). First, we have to evaluate the uncertainty. Once we know the expected error in terms of spread, we can decide what we should do to improve our modeling capability.

We have seen that for improving the prediction capability of linear models, we can use non-linear models because of their parsimony and indefinite flexibility over linear models [DREYFUS05, Chapter 1]. Further, we have seen that, for improving inference (e.g., estimating PIs) of both linear and non-linear models, when regression assumptions are violated, there are a number of actions we can take (Section 3.1, Table 1). Since we aim at improving estimation and inference at the same time, we focus on non-linear models (i.e., multi-layer feed-forward neural networks).

Therefore, the mathematical problem we deal with is to estimate PIs of linear (polynomials) and non-linear (MLFFNN) models when regression assumptions are violated (Section 3.1). Although we illustrate the problem in two-dimensional space, the considerations made below can be applied, without loss of generality, to an N-dimensional space, where the error

variance depends on the independent variables of the model, i.e. $x = x_1, \ldots, x_Q$.

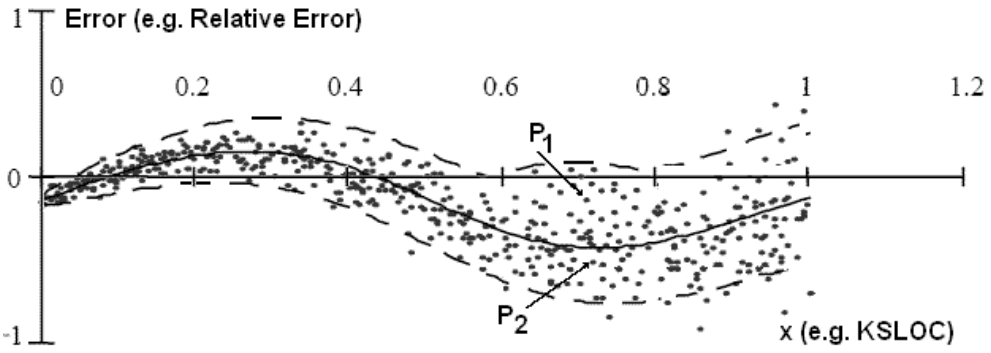

Fig. 13. Errors are x correlated, with increasing variance, biased, and with outliers. The solid line is the expected relative error (x-dependent median) and the region bounded by dashed lines is the associated 95% error prediction interval.

Even though we can apply improvements to a linear and non-linear estimation model when regression assumptions are violated, as shown in Table 1, we may have a situation like the one in Fig. 13. We use a relative (i.e. weighted) measure of the error (Section 3.2), i.e. RE = e/Actual = (Actual − Estimated)/ Actual, instead of the residuals (i.e., e) to avoid the error increasing as x values grow. RE is well known and applied in software cost estimation though other error measures can be used (e.g. BRE) as explained in Section 3.2. The error in calculating PIs by formulas based on t-Student's or z percentiles is that, in software engineering we can almost never assume that the error distribution is a Gaussian with fixed parameters $(0, \sigma^2 I)$. Consequently, we may have type I or II errors.

As an example, assuming the situation in Fig. 13, where for x = 0.8 KSLOC the variance of the RE is expected to be greater than the average variance of the sample (σ^2). Therefore, if we estimated the spread by the average variance we would underestimate the real uncertainty. In Fig. 13,

when x = 0.8, the expected bias is not zero. It should be calculated by the non-linear regression function in Fig. 13 (solid line).

Therefore, the improvement problem that we deal with is the evaluation of the uncertainty of the estimation model assuming violations have occurred, and identifying suitable improvements to the modeling capability.

Chapter 5

State of the Art in the Field

5 Literature review

There has been work in estimating uncertainty, comparing estimation models, using measurement to improve and artificial neural networks related to our work and we discuss them here.

5.1. Uncertainty

Estimation uncertainty has received little attention in the literature [SHEPPERD07A]. It has been dealt with in two different ways: focusing on risk as an unwanted event and on variability when calculating the estimate. Pendharkar criticizes traditional effort estimation techniques "for not providing appropriate causal relationships for predicting software development effort" [PENDHARKAR05], Fenton argues that traditional models do not "provide support for risk assessment and reduction, inability to combine empirical evidence and expert judgment, inability to handle incomplete information" [FENTON00]. Fenton proposes a Bayesian network approach in order to encapsulate casual influences on the dependent variable (e.g., what we are estimating) [FENTON99], [FENTON00]. There exist some studies applying this type of approach, e.g. Moses et al. [MOSES00] propose a Bayesian network for improving estimations in small software companies. Chulani and Boehm [CHULANI99] use the Bayesian theorem to combine prior expert judgment and historical data. Pendharkar et al. [PENDHARKAR05] perform a comparison among non-parametric methodologies such as Bayesian networks, neural networks, and regression tree models.

On the other hand, there exist approaches dealing with the problem in a slightly different way. For instance, Jørgensen [JØRGENSEN03], [JØRGENSEN04A], [JØRGENSEN04B] manages uncertainty considering Prediction Intervals (see Section 3.5). He describes models whose purpose is to explain the accuracy and bias variation of estimates (human-centric) through a regression analysis. Kitchenham and Linkman [KITCHEHNAM97] adopt an organizational perspective based on the *portfolio* concept.

Känsälä [KÄNSÄLÄ97] defines a tool for risk management and applies it to software cost estimation. Recently, Ohsugi et al. [OHSUGI07] have tested empirically the hypothesis that "using more analogues produces a more reliable cost estimate". This analogy-based approach provides meaning to support decision making in project management because knowing the estimation reliability (uncertainty or risk) increase the confidence in make decision about project scheduling and budgeting.

To deal with uncertainty, our approach tries to take advantage of both of these methodologies (Bayesian analysis and Prediction Intervals).

5.2. Comparison of estimation models

The state of the art, in this specific field, is mainly occupied by software cost evaluation models. The research activity is huge. Shepperd [SHEPPERD07A] and Jørgensen and Shepperd [JØRGENSEN07] analyzed such a production by systematic review [KITCHENHAM04] from 320 journal and 333 conference articles since 1990. Other researches are by Stensrud et al. [STENSRUD02], Foss et al. [FOSS03], and Myrtveit et al. [Myrtveit05]. The focus, here, is on model comparison in order to find which is best. They substantially agree on the statement that *MMRE* is not always a good indicator for model comparison. Realistic improvements, in this field, should be based on such statement.

Actually, we could choose universally recognized measures for model comparison (e.g., Mean Square Error, Root Mean Square Error, and χ^2 test [BUSEMEYER00]). We also could adopt the Kitchenham et al.'s proposal [KITCHENHAM01] that considers the distribution z = estimate/actual. In such a research, authors argue that *MMRE* and *PRED*(N) are, respectively, measures of the spread and the kurtosis of z, therefore they cannot be used for model comparison. They suggest using boxplots of the z values or the residuals, which give better assessment of prediction quality than summary statistics. Although slightly related to accuracy indicators, further work can be found in Briand et al.'s work [BRIAND99], [BRIAND00]. They apply *MMRE*, *MdMRE* (the Median Magnitude of Relative Error) and *PRED*(25) with cross-validation [STONE74] to many different estimation techniques (e.g., ordinary least squares regression, stepwise ANOVA, CART, and analogies [MYRTVEIT05]). They find that, the performances of the modeling techniques are not significantly different (with the exception of the analogy-based models, which seem to be less accurate) [BRIAND99].

5.3 Measurement approaches

Basically, there exist few approaches for measurement that software organizations really use [Wohlin00]. They are Goal-Question-Metric (GQM) [BASILI84], [BASILI94A], [BASILI95], [CANTONE00], [GENERO05], [LINDVALL05], [SOLINGEN99] Goal-Driven Measurement [PARK96], and Practical Software Measurement [MCGARRY02]. Indeed, the latter partially deals with improvement issues, while the second one builds on the same rationale of the former, but adopts a different process to figure out what measuring. For these reasons, we are mainly focusing on GQM approach, Quality Improvement Paradigm (QIP), and Experience Factory (EF) [BASILI92B]. Another more general approach that is worth to quote is the Balanced Scorecard Method [KAPLAN92], [KAPLAN96A], and

[KAPLAN96B]. This technique relies on some key concepts of previous management ideas such as Total Quality Management [FEIGENBAUM56], including customer-defined quality, continuous improvement, employee empowerment, and measurement-based management and feedback. Based on metrics and indicators, balanced scorecard analyzes performances of the organization for improvement over time. This is not specific for software organizations, but it may be also used in that context [BROCK03].

5.4 Artificial neural networks

In addressing the third topic, it is important to note that Artificial Neural Network (ANN) research is mainly based on findings concerning mathematical modeling techniques. The research activity is huge in this field, as well. Interested readers can find main valuable contributions in [BISHOP95A] (e.g., pattern recognition), in [DREYFUS05] (e.g., non-linear multi-regression analysis), and in [GUYON05] (e.g., implementation techniques).

Machine Learning (ML) techniques such as cited artificial neural networks, rule induction, genetic algorithms and case-based reasoning have been applied in a wide variety of research fields such as image processing, pattern recognition and classification, econometrics, and biology. In this paper, we deal with ANN applied to Software Engineering problems. ANN has substantially been used in two different ways in this field. For pattern classification (e.g., testing, fault proneness, software risk assessment) [KHOSHGOFTAAR95], [LANUBILE97], [BINGUS96], [KARUNANITHI92], and [SARCIA07], where the output is a real number falling within interval [0; 1]. For non-linear regression (e.g., software cost estimation) [KHOSHGOFTAAR97], [SRINIVASAN95], [SHP11], [WITTIG94], [SAMSON97], and [SRINIVASAN95], fault prediction [KHOSHGOFTAAR94] where the output can be any real number. Some

comparative researches [JØRGENSEN95], [BRIAND99], [BRIAND99], and [SHEPPERD97B] show opposite results with respect to performances. Sometimes, researchers, mainly coming from statistical environment, found out worse results [MYRTVEIT04], [KITCHENHAM01] than ordinary techniques (e.g., log-linear regression [GULEZIAN91], [JEFFERY90], [MEZIES05]).

Current ANN research focuses on improving model assessment and feature selection techniques, [AHA96], [BRIAND92], [KIRSOPP02A], [JOHN94], which are supposed to provide improvements to ML performances (e.g., v-fold cross-validation [STONE74], feature subset selection [KOHAVI97], [CHEN05], principal [JOLLIFE86], [NEUMANN02] and curvilinear component analysis [RUMELHART86], [BISHOP95A], leave-one-out [DREYFUS05], virtual leave-one-out [STOPPOGLIA03]).

Chapter 6

The Solution

6 The solution

Before defining the complete framework for improving models in the software engineering field, we present the mathematical solution to the problem presented in Section 4.

6.1 The mathematical solution

The approach that we propose is an alternative to the non-parametric bootstrap method [EFRON93] and the Bayesian approach [MACKEY91]. It is an extension of the Jørgensen's approach for calculating empirical PIs for both regression-based models [JØRGENSEN04A] and human judgment [JØRGENSEN03]. Although the proposed methodology may be bootstrapped or included in a Markov Chain Monte Carlo simulation framework, we consider neither resampling procedures nor simulation approaches because their computational cost does not agree with our goal of proposing a practical approach that can be used in real situations with an affordable cost.

The proposed strategy of estimating PIs is based on removing as many regression assumptions as possible and considering the error sample in Fig. 13 (i.e. relative error, RE). Since we do not assume that the distribution is a Gaussian and the sample is homoscedastic, we consider the distribution asymmetric, affected by outliers, and with variable spread. Moreover, we assume RE to be x-correlated.

The solution is composed of six steps. (1) We calculate the non-linear robust regression function of y (= RE) with respect to x (= KSLOC), i.e. solid line in Fig. 13. Such a kind of regression function provides an x-dependent

median, minimizing the Minkowski R-distance (R = 1). It is called robust regression because it is less sensitive to outliers and asymmetric distributions.

(2) To deal with the heteroscedasticity issue, we estimate the x-dependent variance empirically. The strategy is based on turning the problem into a two-class discrimination problem. In particular, we use the x-dependent median calculated above for splitting the sample into two classes (Fig. 13); class A (upper) and class B (lower). We use observed elements of classes A and B as representatives of the unobserved data points of each class.

(3) Then, we train a Multi-layer Feed-Forward Neural Network for Discrimination (MLFFNND) so that its output provides a classification decision, i.e. a new data point is classified as belonging to A or B according to its similarity to the data used for training. Therefore, our "a priori" information (i.e. "our belief" as explained in Section 3.7) tells us how far a point is above or below the x-dependent median (binomial choice). Note that, in the example of Section 3.7, we assumed that our "a priori" information concerned people's healthiness, while here our belief is whether an unobserved point is above or below the x-dependent median.

We call this MLFFNND a Bayesian Discrimination Function (BDF) because its output can be interpreted as the posterior probability that any input belongs to class A [BISHOP95A], [DREYFUS05] (Section 3.7). As shown earlier, any input is classified as belonging to class A, if the BDF output is between [0.5, 1], e.g. 0.85. It is classified as belonging to class B otherwise, i.e. the BDF output is in [0,0.5[, e.g. 0.25, where [.,.] is a closed interval and [.,.[is a right-open interval.

Actually, we are not interested in classifying our new observations as shown in Section 3.7. Our aim is to use the BDF for inference (the inverse problem). Making inference by the BDF in Fig. 14 means selecting the interval (Me_{DOWN}, Me_{UP}) on the x-axis corresponding to the two fixed confidence limits (e.g. [0.025, 0.975]). Note that, the BDF performs a

similarity analysis between the characteristics of the project being estimated and the observed projects upon which we built the BDF. Therefore, we use the BDF capabilities to deal with the scope error presented in Section 3.2.

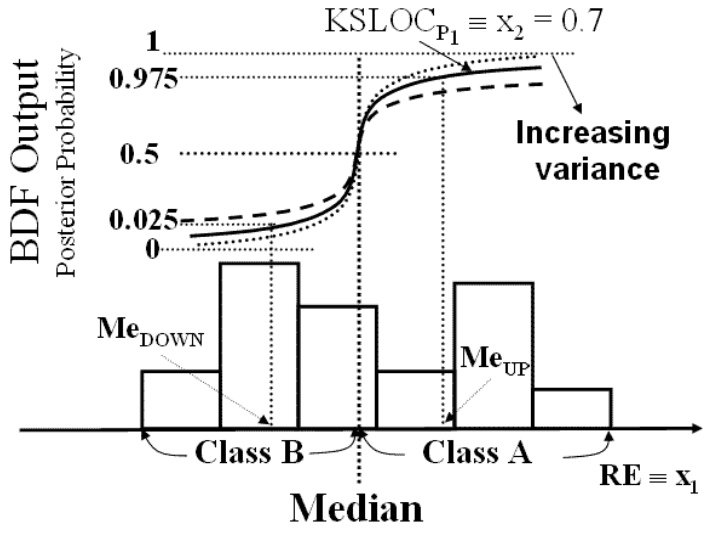

Fig. 14. Posterior probability density obtained by fixing KSLOC and letting RE vary. Dotted and dashed lines represent posterior probability functions with an increasing variance.

The defined BDF in Fig. 14 is expressed by the following relationship, $f_{BDF}(x_1 = RE, x_2=KSLOC) = [0,1]$, i.e. $y = f_{BDF}(x_1, x_2) = Pr(y=1|x_1, x_2)$, where [0, 1] points out any real number in [0,1] (e.g. the posterior probability), x_1 and x_2 represent the sample information, and y=1 represents class A (y=0 represents class B). Assume that, the BDF yields $f_{BDF}(P_1) = 0.85$ and $f_{BDF}(P_2) = 0.25$. Then, project P_1 is classified as belonging to class A, because $f_{BDF}(P_1) \geq 0.5$ and project P_2 is classified as belonging to class B because $f_{BDF}(P_2) < 0.5$.

Assume now that instead of fixing both values x_1 = RE and x_2=KSLOC, we fix only x_2 (= constant c), and let x_1 vary. Then, BDF turns into $Pr_{x_1}(y=1|x_2) = f_{|x_2=c}(x_1) = y(x_1)$. Note that, in case of an N-dimensional space (with

N > 1), we fix all variables except x_1 (the relative error RE), i.e. $Pr_{x_1}(y=1|x_2..x_N)=f_{|x_2=c_1,...,x_N=c_{N-1}}(x_1)=y(x_1)$ where the variables (x_2 … x_N) are the same as the independent variables of the estimation model.

(4) Once we build the posterior probability density (solid line in Fig. 14), we can obtain a Bayesian PI by fixing a 95% confidence level, i.e. (0.025, 0.975), and picking the corresponding values of RE on the x-axis, i.e. (Me_{DOWN}, Me_{UP}). This interval represents the expected range where the next RE will fall. The posterior probability density in Fig. 14 has an important characteristic. Its slope gets steeper as the variance decreases; it gets flatter as the variance increases [HUSMEIER04]. Therefore, the increasing variance in Fig. 13 has a geometric representation in the model of Fig. 14. The increasing variance in Fig. 13 corresponds to a flatter slope of the sigmoid curves in Fig. 14, e.g. variance corresponding to the dotted line is lower than the variance corresponding to the dashed line. Therefore, the proposed strategy based on the BDF is able to evaluate the variance (calculating the RE range) from the slope of the posterior probability density function empirically. This approach is quite different from estimating the variance by resampling procedures such as the bootstrap. See [HUSMEIER04, pp 20-24] for additional explanations.

(5) To calculate the estimation PI (e.g. the effort) for the RE range, i.e. (Me_{DOWN}, Me_{UP}), we first consider the formula RE = (Actual − Estimated)/Actual and then, we deduce Actual = Estimated/(1 − RE). In Section 3.2 we used different symbols to indicate prediction intervals (i.e. [μ_{DOWN}, μ_{UP}]). That is because, previously we used the mean (μ), while we use the median (Me) now.

As shown by Jørgensen et al. [JØRGENSEN03], the PI is $[O_{est}^{N+1}/(1-Me_{DOWN}), O_{est}^{N+1}/(1-Me_{UP})]$, where $O_{est}^{N+1}=f_R(x',b)$, e.g. $f_R(x' = 0.7,b)$, see Section 3.2. For instance, assuming that the RE interval obtained from

Fig. 14 is [-0.9, 0.1] and Estimated effort = 3 person months, the PI is [3/(1-(-0.9)), 3/(1-0.1)] =[1.6, 3.4] person months.

Although the proposed prediction interval shown in Fig. 14, i.e. (Me_{DOWN}, Me_{UP}), has been derived empirically without making any specific assumptions, it actually represents an underestimate of the actual uncertainty. That is because the BDF was derived from the principle of maximum likelihood estimate (MLE) that considers only the most probable parameter set. We should consider the uncertainty over the unknown parameters of the BDF, as well. To correct this underestimation, MacKey proposes to apply the Bayesian framework to classification problems. As we have already explained above, in our view his approach is too computationally cumbersome and based on too many approximations and assumptions (e.g. normality). Another approach to correct the underestimation can be to apply Markov Chain Monte Carlo (MCMC) simulation, which avoids the Gaussian approximations [GILKS96], but is also computationally expensive and its reliability depends on the simulation assumptions.

(6) To avoid the problem of a too high computational cost without any promise of getting better results, we prefer estimating this additional uncertainty through the generalization error provided by cross-validation (leave-one-out or K-fold). This procedure is more convenient because the cross-validation procedure has to be performed anyway when selecting the classification model and it has a comparable computational cost with respect to the bootstrap or MCMC procedures. Therefore, the proposed procedure would be more convenient from a practical point of view avoiding any specific assumptions.

Since the (standardized) leave-one-out cross-validation score (CVS) is calculated by the SQRT(MSE), which is an unbiased estimator of the generalization error of the BDF [VAPNIK95], we use this quantity as a correction factor.

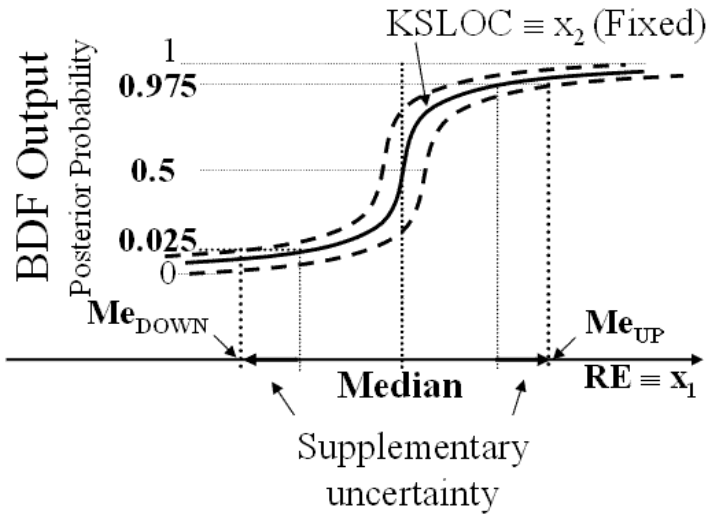

Fig. 15. Posterior probability density (solid line) obtained by fixing KSLOC and letting RE vary with supplementary uncertainty due to the error in calculating the model parameters.

Instead of using LOOCV, we may apply K-fold CV, as well. However, its score would not be an unbiased estimator of the generalization error even though it would be more realistic than the score obtained by leaving out only one data point [DREYFUS05]. We sum and subtract CVS to the posterior probability density function (solid line in Fig. 14), obtaining the further two dashed lines in Fig. 15. In particular, the upper dashed line is $f_{|x_2=c}(x_1)+\text{SQRT(MSE)}$ and the lower dashed line is $f_{|x_2=c}(x_1)-\text{SQRT(MSE)}$. The band included within the two dashed lines represents the overall (standardized) uncertainty due to the variation in the BDF parameters. The upper and lower shifts determine an increase to the magnitude of the prediction interval (Me_{DOWN}, Me_{UP}) due to the supplementary uncertainty.

The final prediction interval can be derived by fixing the 95% confidence as above, i.e. (0.025, 0.975), and selecting the corresponding values of RE on the x-axis. In particular, Me_{DOWN} is calculated by the crossing point between the 0.025-horizontal line and the upper dashed line. The Me_{UP} is calculated

by the crossing point between the 0.975-horizontal line and the lower dashed line.

The procedure presented here can be applied at times T and T+2 (See Fig. 3). If we apply the PI calculation at time T, when only estimated values are available, we can address model error, scope error, and assumption error. If we apply the proposed PI calculation approach at time T+2 when actual values are also available, we can work on improving the estimation model.

6.2 Benefits of applying the mathematical solution

The approach proposed in this work can be used to evaluate and improve the performance of any estimation methodology/model, e.g., regression functions, machines learning, human-based judgment, COCOMO, SLIM, Bayesian networks, and function point analysis. Here we focus on methodologies based on parametric estimation models (e.g. regression functions and machine learning) because they not only provide a rational and repeatable improvement process but also provide the opportunity of showing how we can improve the models in terms of missing variables, model complexity, and scope extension.

There are some important implications coming from the solution presented in Section 6.1. Generally, when using an estimation model for prediction and inference, we can apply improvement techniques as reported in Table 1 and evaluate the model in terms of its relative error. Because of the unmanaged violations of parametric models, model evaluation is not reliable, limiting our ability to improve the estimation model and making parametric models unattractive for prediction in reality.

The approach presented in Section 6.1 overcomes these issues. It not only extends our ability to make improvements, as described in Table 1, by dealing with the consequences of violations, it also manages the improvement process by building experience packages, i.e. the Bayesian

Discrimination Function (BDF). This package makes it faster and easier to control, reuse, retrieve, and disseminate organizational experience than using simple diagrams and statistics as is usually done. This way of packaging, deploying and exploiting experiences within a learning organization enhances the decision making process in terms of competitiveness, business goal achievement, and organization-wide common procedures (e.g., project control, easy-to-use tools, standardized procedures over a number of projects, tracing improvements, and making predictions on the estimation model uncertainty).

Besides the benefits discussed in Section 6.1, the novelty of this approach is that we can automate the evaluation and prediction tool (i.e. the BDF) based on neural networks to improve estimation models. In particular, the BDF can execute the similarity analysis to solve the problem posed in Table 3 (Section 3.2). We encapsulate the estimation model capability in terms of relative error in the BDF so that the BDF becomes a sort of an intelligent agent supporting the estimation model improvement process, what we called Automated Experience Package (AEP), in Fig. 2. It provides the expected relative error of the estimation model according to the characteristics of the project being estimated. For instance, sending queries to the BDF, we can answer questions such as what would be the uncertainty (and so the risk) in estimating project P when using the estimation model? Which is the estimate that minimizes the risk of getting an estimation failure? How can we improve the estimation model (what is missing)? What is required for improving the estimation model? What is the organization's experience with respect to project P? Can the organization's experience help us deal with projects that differ partially from the projects estimated so far? In this section, we show how to build and use the BDF for prediction and improvement.

We summarize benefits of the proposed methodology over previous approaches. (1) We embody the estimation model error into a discrimination

neural network, which can provide error prediction intervals according to a similarity analysis without making any specific assumptions. (2) The BDF can be used as an automatic evaluation and prediction tool. Therefore, the BDF is faster and easier to apply than traditional methods that provide the prediction interval but do not provide the way of improving the model. (3) The proposed strategy allows us to offload the complexity of developing the prediction models to experts (e.g. the experience factory). Note that, the BDF can be also used as a support tool for experts in making project predictions and inference (e.g. in case of expert-based predictions [JØRGENSEN03]). (4) The BDF capability of predicting the estimation model error is fully known at time T (Fig. 3). Therefore, the BDF can be used for investigating whether the organization's experience (i.e., its prediction capability) in estimating specific kinds of projects (or variables) is sufficient, e.g. before bidding on a contract, we may analyze the strength and weakness of the organization in predicting the proposed software system cost and then decide whether to bid or not. (5) The BDF allows simplification of the implementation of Improvement-Oriented Software Environments such as TAME systems [BASILI88] by adopting standardized packages of experience. (6) The proposed approach uses parametric models to overcome the limitations posed by regression violations. As we explained in Section 3.2 through Fig. 7 and Fig. 8, the proposed approach turns the usual parametric estimation process into an analysis aiming at choosing an estimate that minimizes the risk of an estimation failure (e.g. underestimate).

In Section 6.1, we built the Bayesian Discrimination Function (BDF) by considering a hypothetical data set of projects, where each project was described by two sets of variables (and related data). The first set of variables was composed of the same set as the estimation model (i.e. variables X). The second set was composed of the relative error. RE values were obtained by feeding the X-values into the estimation model and calculating the RE =

(Actual – Estimated)/Actual for each data point. We built a data set where each data point was described by the project characteristics and the relative error of the estimation model (X, RE). We considered the variables X as independent variables and the relative error RE as a dependent variable. For more details on using an x-dependent variance, see [BISHOP95A, pp. 211-212]. Then, based on the X-dependent median of the relative error (solid line in Fig. 13), we split the data set into two subsets (binary choice). Consequently, we associated with class A each data point having a relative error equal to or greater than the X-dependent median (on that point). We associated with class B each data point having a relative error less than the X-dependent median (on that point) (Fig. 14). Then, we trained the BDF in Fig. 14 and Fig. 15 to map classes A and B to one and zero, respectively. Based on the BDF thus calibrated, we inferred the expected relative error range by feeding the values of the project being estimated into the BDF. We answered the following question, what is the expected error range with a 95% (or 90%) of confidence (i.e. credibility) for the considered project (i.e. given the X values) avoiding any specific assumption? In other words, we performed a similarity analysis in the sense of Section 3.2 (Table 2 and Table 3) to infer the expected error range of the project being estimated, given the characteristics and related errors of the projects used for building the BDF (the history).

The similarity analysis performed by the BDF has an important consequence. In Section 3.2, we argued that Fig. 7 was incomplete because of the inability of the usual approaches to deal with scope error. Now, we argue that the solution proposed in Section 6.1 (Fig. 14 and Fig. 15) can be applied to deal with the scope error as well. To do so, we have to evaluate the prediction interval in Fig. 7 by the BDF (as explained in Section 6.1). Of course, the similarity analysis that we propose has some limitations. Nevertheless, we take advantage of such limitations for assessing whether or

not the BDF is reliable. For instance, when we try to estimate prediction intervals on projects characterized by values never observed before, we may get a scope error in addition to model and assumption errors. Therefore, the prediction interval obtained from the BDF may be inaccurate (where the accuracy of the BDF prediction interval is defined as its capability to include the relative error of the estimation model [JØRGENSEN03]).

We can execute a scope error analysis (i.e. the similarity analysis illustrated so far) using both estimated data (prediction) and actual data (control) as inputs to the BDF. Executing the BDF reliability analysis with actual data (step 5.B in Fig. 2), we can improve the estimation model in terms of model scope. The improvement comes about by building a new version of the estimation model and adding the data points on which the scope error occurred to the training set of the estimation model. Therefore, this kind of improvement involves extending the estimation model scope. If we added data points on which no scope error occurred to the training set of the estimation model, the improvement might be only about shortening the magnitude of the error prediction interval, but no scope extension would be made.

Executing the BDF reliability analysis with estimated data (step 4.A in Fig. 2), we cannot improve the model because the relative error is just estimated. The only thing that we can do is to mitigate the risk by choosing a suitable estimate according the organization risk policy.

6.3 An Estimation Improvement Process overview

Our approach is modular insofar as each suggested technique can be eventually improved, integrated, or even replaced by better tools based on new findings. What we want to keep fixed is the improvement strategy part of this approach.

We use software cost estimation (i.e. effort prediction) to explain and demonstrate the procedure. In particular, we use the COCOMO-NASA data set [PROMISE] because of its huge popularity in the research community. As a public data set, it provides a common ground for experimental replication and discussion. We are not suggesting the use of the COCOMO-I model for improving estimation performance, but use it to show how the limitations of parametric models can be overcome.

6.3.1 The Estimation Improvement Process

The Estimation Improvement Process (EIP) that we refer to (Fig. 16) is a specialization of the six steps of the QIP (Fig. 2).

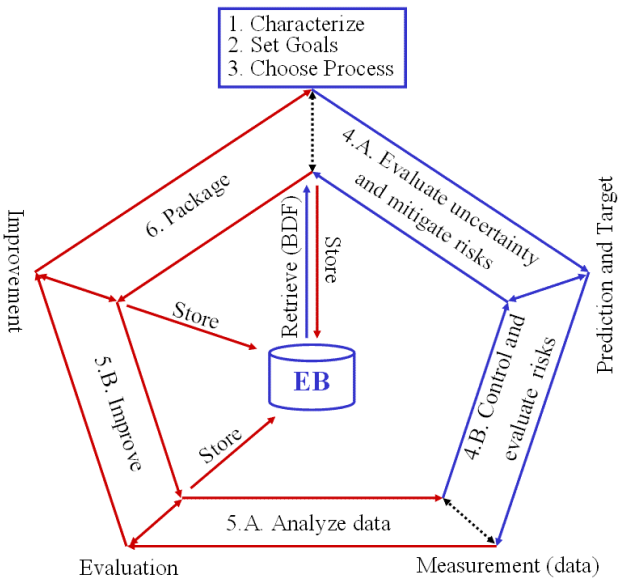

Fig. 16. Estimation Improvement Process as a specialization of the Quality Improvement Paradigm (QIP).

We need to initialize the loop by building an initial BDF_0, which is stored in the Experience Base (EB). Then, for each new project, we perform an iteration of the EIP to evolve the BDF (Fig. 17).

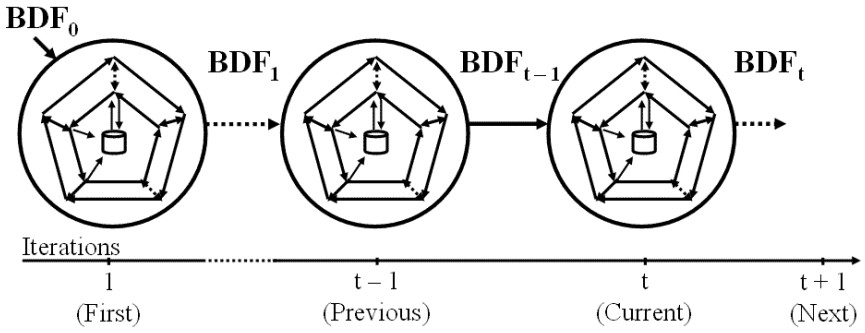

Fig. 17. Estimation Improvement Process Iterations.

Iteration 1 of the EIP uses BDF_0 and provides the BDF_1 for the next iteration. Analogously, t-th iteration receives the BDF_{t-1} and provides the BDF_t. The process in Fig. 17 can be considered as a state machine describing the evolutionary improvement of the estimation model, where the number of the states evolves over time and the next state depends only on the last state (Markov chain process).

Since the first iteration in Fig. 17 can take place if and only if the BDF_0 is available, we start first with explaining step 6 (Package) in Fig. 2 and Fig. 16. In particular, we focus on the stage "Build (AEP)" because the remaining stages have a few differences with respect to the QIP [BASILI92B].

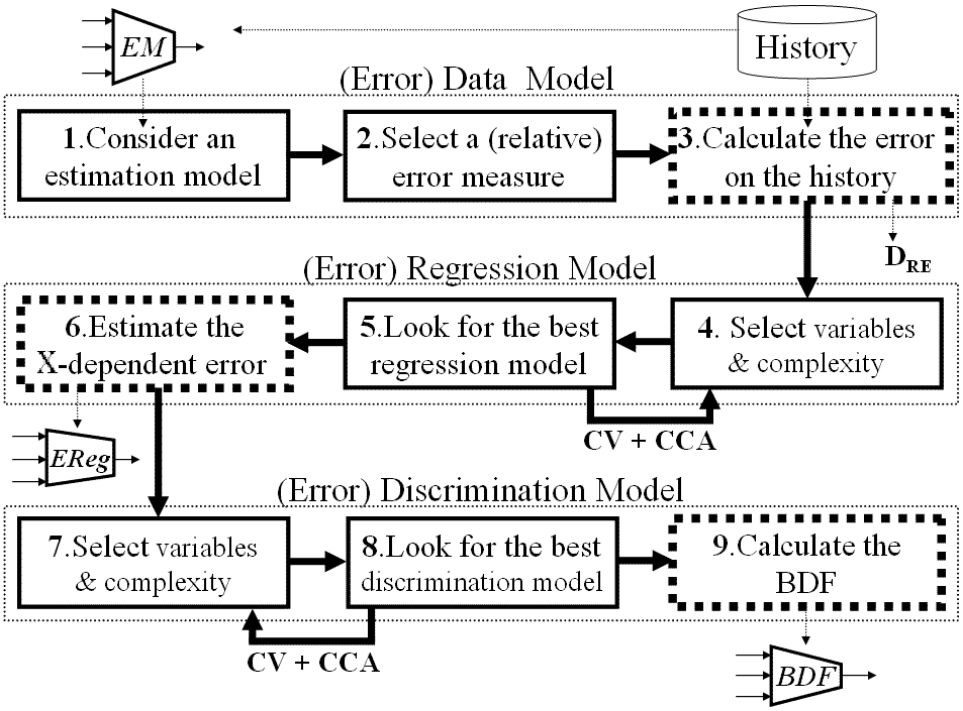

Fig. 18. The 9-step procedure for building the Bayesian Discrimination Function (BDF) summarizes the "Build (AEP)" of step 6. (Package) in Fig. 2 and Fig. 16.

The "Build (AEP)" stage is presented in Fig. 18, showing the procedure for building an Automated Experience Package (AEP) for estimation model improvement that we called Bayesian Discrimination Function (BDF). The BDFs can be built and packaged by the experience factory, where such expertise should reside. This eliminates any overhead to the project manager and provides specific focused knowledge to the project.

In particular, a project organization:

- evaluates uncertainty,
- mitigates risks,
- makes predictions on its specific project environment,
- controls the project, and
- evaluates risks.

The EF:

- analyzes data, lessons learned, and feedbacks coming from the project organizations,
- improves the estimation model, and
- packages the experience by building a new version of the estimation model and the BDF,
- stores experience into the EB.

As we have illustrated in Section 6.1, the proposed approach is able to deal with the scope error, as well. Based on the considerations in Section 3.2 (Fig. 7), however, evaluating the error prediction interval in the case of scope and model error is not enough. We have to deal with the assumption error, as well. This means that, we have to apply the risk exposure analysis (Fig. 8) reported in the end of Section 3.2.

Step 4.A in Fig. 16 is summarized in Fig. 19. In particular, because of the assumption error (Fig. 8) we have to consider a variety of possible estimation model inputs, represented by overlapping rectangles to show multiple sets of inputs, error PIs, and estimate PIs.

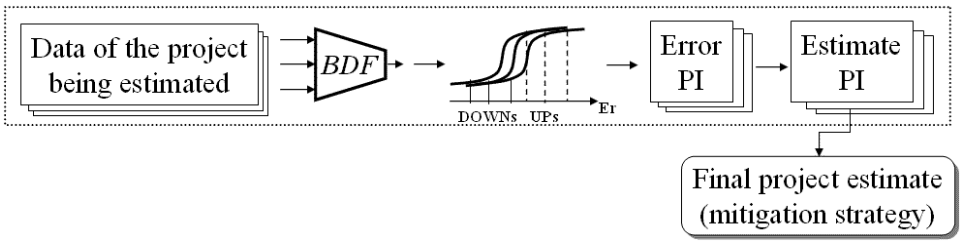

Fig. 19. Evaluate uncertainty and mitigate risks (estimated data).

Based on the mathematical solution explained in Section 6.1, once we get the estimate prediction interval from the BDF as shown in Fig. 19, we can apply the mitigation strategy explained in Fig. 8. The mitigation strategy

consists of fixing the final estimate of the project to minimize the estimation risk. For instance, the organization may have different risk policies, (1) get the contract accepting the risk of earning less money, (2) avoid loosing money once the contract has been obtained, or (3) something in between. The organization may increase the chance of getting the contract by increasing the underestimate risk (i.e. decreasing the bid) or decrease the risk of loosing money once the contract has been obtained by decreasing the underestimate risk (i.e. increasing the bid), see Fig. 8. It is important noting that step 4.A in Fig. 19 is based on feeding the estimated data into the BDF. In fact, we do not know the actual data at that time.

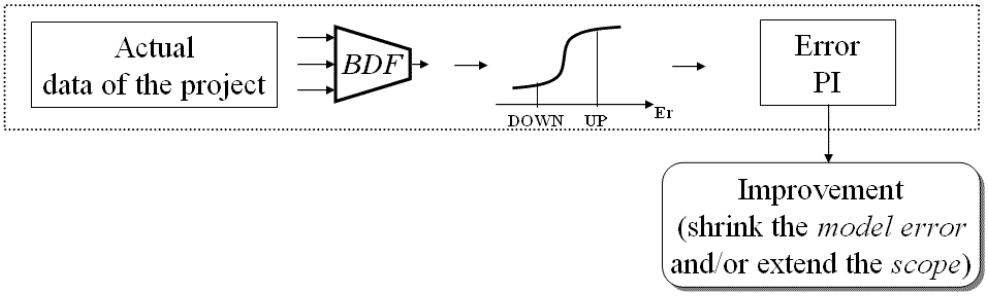

Fig. 20. Improve the estimation model EM (actual data).

The improvement takes place in step 5.B (Fig. 16) and is summarized in Fig. 20. To improve the estimation model we need to know the actual data of the project. Although we use a relative error measure instead of an absolute measure, the magnitude of the estimate prediction interval $(O^{N+1}_{est\ DOWN}, O^{N+1}_{est\ UP})$ cannot be used for investigating the improvement needs of the estimation model. We have to look at the error prediction interval (Me_{DOWN}, Me_{UP}), i.e. performing the analysis shown in Fig. 7 on the relative error.

When dealing with actual data, the only two kinds of error that we have to worry about are model error (intrinsic to the model) and scope error (coming

from an out-of-scope use of the model). For model error, improving the model means unbiasing the error prediction interval (e.g. [c] and [d] in Fig. 7) and/or reducing the magnitude of the prediction interval (e.g. [b] and [d] in Fig. 7). Improving the model in the case of scope error means extending the project prediction capability of the model. Based on the analysis in Fig. 20, the EIP finishes up by building an improved version of the estimation model and the related BDF that embodies the current organization experience (Fig. 17).

6.3.2 Strategic organization control

The Estimation Improvement Process in Fig. 16 is a process executed for each project organization and supported by the EF. For each EIP iteration, the EF produces a new BDF. At the beginning of iteration t, we would have a number of BDFs, i.e. (BDF_0, BDF_1, …, BDF_{t-1}) Fig. 17. The number of variables included in each BDF may be different from each other.

The set of all of the BDFs (Fig. 17) can be used to trace the estimation model improvement over time. For a fixed project P, the set of BDFs can be used to figure out whether the estimation model has actually been improved, kept the same, or worsened over time (Fig. 21).

For instance, in Fig. 21, assume that we are at the beginning of iteration t and we would like to investigate the evolution of the error prediction interval for the next project P over the previous iterations. We feed values describing P into each element of (BDF_0, BDF_1, …, BDF_{t-1}). Based on the procedure in Section 6.1, we can get a 90% (or 95%) prediction interval from each BDF, i.e. we get $ST^P = (PI^P_1, PI^P_2, \ldots, PI^P_{t-1}, PI^P_t)$, where PI^P_1 is provided by BDF_0, PI^P_2 is provided by BDF_1, and PI^P_t is provided by BDF_{t-1}. Possible results of such an analysis are shown in Fig. 21, where an improvement takes place when the PIs shortened over the iterations, the same if their magnitude keeps constant, and worsening if their magnitude increases.

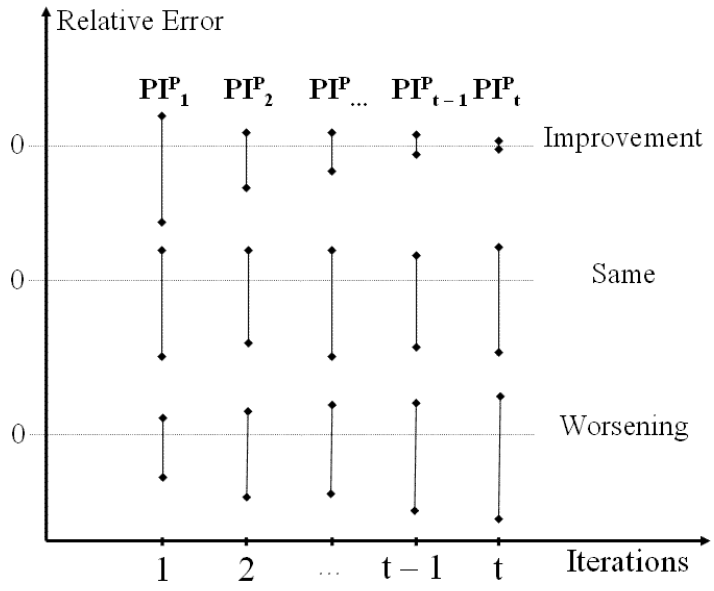

Fig. 21. Error Prediction Intervals provided by each BDF fed with the data of the next project P.

Based on this analysis, the latest version of the estimation model may no longer be useful for predicting the variable of interest (e.g. effort) because of the increased risk. This situation might mean that something has changed in the organization (e.g. productivity, the development environment, people) and new variables or estimation models should be considered. This means that, the process in Fig. 17 may be used by the EF to provide proactive support for higher levels in the organization in understanding changes that cause risks before they occur.

6.4 Building BDFs (the framework)

The term “framework” is used to denote a set of rules, techniques, strategies, methods, and procedures that lead to tools (the BDF) for evaluating the risk and uncertainty (i.e., prediction intervals) of an estimation model and improving its prediction and inference capabilities.

In the current section, we delve into the details of the approach. We describe activities for (1) building the BDF (Fig. 18), (2) using the BDF in evaluating uncertainty and mitigating the risk (Fig. 19), and (3) exploiting the BDF for improvement (Fig. 20). We will not deal in depth with steps 4.B and 5.A in Fig. 16, because they are typically addressed in risk management approaches [CMMI].

The process for building a BDF flows through three different layers, each layer corresponding to a different error model: (Error) Data model, (Error) Regression model, and (Error) Discrimination model. Each layer produces an output that is used by the next layer. Outputs are, respectively, the relative error sample (D_{RE}), the error regression model (EReg), which provides the X-dependent median of the relative error, and the BDF. Each model is composed of three different activities, which can be iterated.

Each output can be viewed as an experience package in its own right, representing a different abstraction of the error. For instance, the error data layer is the lowest abstraction where we consider estimation model errors, e.g. this package is a sample of relative errors (D_{RE}). In the second layer (regression), we map D_{RE} to the independent variables X to predict the expected error, e.g., this package is an error regression function (EReg). The discrimination layer is a higher abstraction where we predict the error variability according the project characteristics, e.g. this package is the Bayesian Discrimination Function (BDF).

6.4.1 Preconditions

Historical data is available (e.g., the history stored in the experience base, Fig. 18). The data set is a QxN matrix (DS_{QxN}) where Q represents the number of X variables that describe each observation and N represents the number of observations (called data points or examples). Note that, DS_{QxN} is also the data set of independent variables used for building the estimation

model (EM). Y_{act} is the dependent variable (e.g. actual effort) of the estimation model and $Y_{act,\,(1xN)}$ is a 1xN vector of Y_{act} values not included in DS_{QxN}. Therefore,

$$DS_{QxN} = \begin{bmatrix} x_{11} & \cdots & \cdots & x_{1N} \\ x_{21} & \cdot & \cdot & x_{2N} \\ & \cdot & \cdot & \\ x_{Q1} & \cdots & \cdots & x_{QN} \end{bmatrix} . \quad (21)$$

In Eqn. (21), there are N data points (one for each column) each described by Q variables (X_1, …, X_Q). The estimation model (EM) provides Y_{est} = EM (DS_{QxN}, b), where b represents the parameter values of the EM. In software cost estimation, Y_{est} would be the estimated effort on the history, Y_{act} the actual effort, and DS_{QxN} the project input values.

6.4.2 (Error) Data Layer

The aim in this layer is to characterize the performance of the chosen estimation model according to the relative error measure.

Step 1 – Consider an estimation model

We mainly refer to improvable models where a linear, log-linear, or non-linear parametric function (e.g. machine learning) is sought. Model variables may come from a known model (e.g., COCOMO-I, COCOMO-II [COCOMO2]) or from a specific environment (variables that are known for a specific organization). Except for the first time, when performing this step, we assume that we have an estimation model (EM), see Fig. 17. Therefore, the model being selected in this step has already evolved over the previous improvement iterations of the EIP. As an example, the EM may be a log-linear function where variables refer to the COCOMO-I model [CHEN05] or

a multi-layer neural network for regression based on the same variables [FINNIE97]. Of course, we may consider other models based on different variables, as well.

Step 2 – Select a (relative) error measure

We have discussed several issues about selecting the relative error measure. In this step, we select an error measure that can separate spread and bias such as RE or BRE. Here we focus on the RE because of its wider popularity.

Step 3 – Calculate the (relative) error on the history

Based on the two previous steps and the available historical data, we can calculate the relative error measures over the history (D_{RE}). For instance, choosing RE, we would have $RE_i = (Y^i_{act} - Y^i_{est})/Y^i_{act}$ for i = 1 to N. Therefore, the output of this step is a relative error sample $D_{RE}=\{RE_1, RE_2 \ldots RE_N\}$. These N error measures represent the performance of the estimation model based on the chosen relative error (weighted residuals). We add D_{RE} to the data set, DS_{QxN}, , i.e. $DS_{(Q+1)xN} \equiv \{DS_{QxN} \cup D_{RE}\}$, where D_{RE} is a 1xN vector as it will be used to build the BDF in step 9. This union operation is shown in Eqn. (22). Note that D_{RE} may also represent the performance of human-based techniques as well since the X variables are the variables affecting the relative prediction error [JØRGENSEN03].

$$DS_{(Q+1)xN} = \begin{bmatrix} x_{11} & \cdots & \cdots & x_{1N} \\ x_{21} & \cdot & \cdot & x_{2N} \\ & \cdot & \cdot & \\ x_{Q1} & \cdot & \cdot & x_{QN} \\ x_{(Q+1)1} & \cdots & \cdots & x_{(Q+1)N} \end{bmatrix} . \quad (22)$$

Where, (Q+1)-th row of the matrix in (22) is

$$D_{RE} = [x_{(Q+1)1} \quad \dots \quad \dots \quad x_{(Q+1)N}] = [RE_1 \quad \dots \quad \dots \quad RE_N] \ . \qquad (23)$$

6.4.3 (Error) Regression Layer

The aim of this layer is to calculate a regression function predicting the expected error according to the variables (X_1, …, X_Q). As we have shown, when regression assumptions are violated, the estimation model can be biased depending on the X variables. Because of the bias, it is wrong to use univariate statistics such as mean or median of the sample D_{RE} to calculate the expected error. We should use a regression function EReg (Fig. 18) taking (X_1, …, X_Q) as independent variables and the RE as a dependent variable, i.e. RE = EReg (X_1, …, X_Q) + ε would be the regression model, see Fig. 13. If there is no correlation between the X variables and RE, the EReg can be replaced with summary statistics on RE such as the median.

Since we cannot assume that the variables have a Gaussian distribution and may be skewed and have outliers, the error regression function EReg should be calculated by a (non-linear) regression function minimizing the Minkowski-R distance with R = 1 (non-linear robust regression) instead of using the least squares linear regression (Sections 3.1 and 3.2). Another approach to calculating the regression function EReg may be to remove outliers making the distributions as close to a Gaussian distribution as possible and applying the least squares strategy. However, removing outliers may be dangerous and unreliable. Therefore, we should remove only the few data points that are in strong disagreement with the rest of the error sample using a box-plot diagram (or other non-parametric techniques [KNOR98], [PAPADIMITRIOU03]) and apply the non-linear robust regression to minimize the Minkowski-R distance with R = 1. Other options may be applied as well. We will focus on the one based on not removing outliers and estimating a non-linear regression function to minimize the Minkowski-R

distance with R = 1. This choice is based on the goal of avoiding any specific assumption.

Step 4 – Select variables and complexity

In this step, we select the independent variables (X_1, …, X_Q) of the error regression function (EReg) that affect the relative error RE and the least complex family of functions in terms of flexibility that describe the dependent variable according to the selected independent variables (Section 3.6). If the RE is not correlated with any X variable, to estimate the expected RE, we can calculate the median of RE, i.e. Median(D_{RE}). It is important noting that, since the selected estimation model (EM) is a parametric function, the X variables of EReg are the same as the X variables of the EM [BISHOP95A]. If we considered human judgment-based approaches, the error may also be affected by variables other than X, see [JØRGENSEN04A] for some additional explanations.

Note that, for multi-layer feed-forward neural networks complexity refers to the hidden unit number (Fig. 10) while for polynomials, complexity refers to the degree of the polynomial in the sense of Eqns (16) and (17) (Section 3.6). The current step is strictly related to Step 5, where we look for the best model in terms of generalization error ("best" refers to the model that shows the smallest generalization error) [DREYFUS05, pp. 134-137].

To select the most suitable model and variables, we apply leave-one-out cross validation (LOOCV) together with the exhaustive procedure as explained in Section 3.2. We start by considering the suitability of linear models with respect to their generalization error, and whether we have to increase their complexity (i.e., changing their shape) until an acceptable model is found. LOOCV consists of calculating a score for each possible model and choosing the best one [DREYFUS05, pp. 134-137]. For instance, the score for each model is calculated by the SQRT(MSE) where MSE is the mean of the squared error calculated on the data points left out and averaged

over the scores of the models calibrated by leaving out the data points used for calculating the error. To perform LOOCV together with the exhaustive variable selection procedure, the score has to be calculated for each buildable model by changing the number of the input variables (e.g. starting with the complete set, we remove a variable stepwise backwards until the set has only one variable). For instance, if we had Q input variables then we would consider 2^Q possible models, each composed of different numbers of input variables. For each model, we calculate the LOOCV score and choose the best model among them. For each of the 2^Q models, we calculate N different LOOCV scores, one for each project. The overall cost of executing the exhaustive procedure is $N * 2^Q$ for one family of functions having the same complexity. So, if we had Q = 15 variables and N = 77 projects, we should calculate $77 * 2^{15} = 77 * 32{,}768 = 2{,}523{,}136$ different models for only one family. Assuming that it takes 0.5 seconds on average to calculate the parameters of one model, calculating all the LOOCV scores would take about 14 days; and this for only one family of functions having the same complexity. If on the average for each family of functions, we increased the set of complexities to K, then the calculation would take (on the average) K * 14 days. The exhaustive procedure is too expensive to be applied in reality.

Based on the motivations described in Section 3.1, we use LOOCV (or K-fold CV) together with Curvilinear Component Analysis (CCA), because they avoid performing the exhaustive procedure and make the model more parsimonious [SARCIA08].

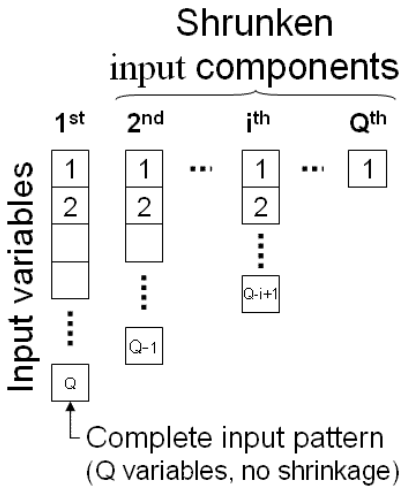

Fig. 22. This diagram illustrates the strategy to performing the LOO CV with CCA. For instance, in the ith vertical pattern we use Q - (i -1) = Q - i +1 components to represent the complete pattern. For each of Q vertical patterns we calculate the LOOCV score and keep the best model (the one having the smallest score).

When applying CCA together with LOOCV (Fig. 22), instead of considering 2^Q possible models (each composed of a different set of input variables), we consider Q models, e.g. Q-th model has Q input variables, (Q-1)-th model has Q-1 input variables, and the last one has only one input variable. Using the example above, the cost of this procedure for only one family of functions having the same complexity is N * Q. If for each family of functions, we increased the complexity to K on the average, then the number of models would be on the average K * N * Q. Moreover, when considering non-linear models using an iterative procedure to calibrate the model parameters, we must try several initial variables as the starting point of the iterative procedure [DREYFUS05]. If the model is linear in the parameters, we have a closed solution to the calibration parameter problem and K would be 1. We then pick the best model, i.e. the one with the least generalization error. Let R be the average number of trail runs, then the cost of executing LOOCV with CCA for a non-linear model for K different

families of function on the average would be R * K * N * Q. For instance, with R = 6 repetitions, Q = 15 variables, N = 77 data points, if each model took 0.5 seconds to be calculated, the overall cost would be about 1 hour and 10 minutes for each iteration. Therefore, if we performed on the average K iterations, the overall cost would be K * (1h 10').

As we explained in Section 3.1, stepwise regression assumes the sample is not affected by multicollinearity. Since we want to avoid any specific assumptions about the selection procedure, we apply CCA instead of using other techniques such as stepwise regression.

LOOCV could also be combined with Principal Component Analysis (PCA) [JOLLYFE86], [NEUMANN02]. Both CCA and PCA are able to find a shrunken configuration of the complete input pattern that does not suffer from multicollinearity (Section 3.4). If we compress the information expressed through Q variables by applying CCA, e.g. substituting the variables with a shrunken representation, we lose information. However, if some components among the Qs are redundant, CCA (or PCA) turns the complete input pattern into an equivalent representation without loosing information. Note that, PCA captures the linear relationships and CCA captures both the linear and curvilinear ones [DREYFUS05, p. 96]. For this improved capability, we suggest using CCA. The CCA output is shown in Fig. 22, where there are Q configurations, obtained one by one by fixing the number of components for shrinkage. For instance (Fig. 22), the 1-st item coincides with the complete input pattern (no shrinkage), the 2-nd one is turned into a configuration with Q-1 components, i-th one is turned into a configuration with Q-i+1 components, the last one (Q-th) is turned into a configuration with only 1 component. In Section 7, we will show a practical application with CCA.

Note that, Steps 4 and 5 are repeated cyclically. We start b y considering the first configuration of Q components (Fig. 22), and then we perform Step

5. In the subsequent executions of Step 4, we consider the remaining configurations in Fig. 22. This means that we have the same number of executions of Step 5 as of Step 4, Q, as we feed each column in Fig. 22 (from 1-st through Q-th), into Step 5.

Step 5 – Look for the best regression model

For each execution of this step, we receive i-th input component calculated by CCA in Step 4.

For each column in Fig. 22, we consider a set of non-linear regression models (EReg), where each element of this set has increasing complexity starting from 1 forward. For each element of the model set, we calculate the LOOCV score. If we increase the model complexity, the LOOCV score can no longer decrease [DREYFUS05, pp. 134-137]; it can stay the same or increase. We continue increasing the complexity as the score decreases. If we have executed K steps before stopping the procedure, then (K-1)-th model is the best (Fig. 22). This (K-1)-th model is stored. The procedure goes on cyclically executing Steps 4 and 5 until all the columns in Fig. 22 have been processed. Overall, the best model is the one with the smallest LOOCV score among the stored Qs. Note that, in performing Steps 4 and 5, we turn the data set DS_{QxN} into a shrunken data set X_{RxN}

$$(DS_{QxN}) \Rightarrow X_{RxN} \qquad (24)$$

where $R \leq Q$.

Step 6 – Estimate the X-dependent error

Based on outputs of previous steps, we now have a less redundant data set (i.e., a shrunken input representation of the initial data set). Of course, it may be possible that the best data set is the one with the the initial size (no shrinking). Based on X_{RxN}, Eqn. (24), we can build a model as parsimonious as possible (Section 3.6).

Then, we calculate the unknown parameters W of the error regression function EReg, Eqn (25).

$$D_{RE} = EReg\ (X_{RxN}, W) \qquad (25)$$

where, D_{RE} is the known vector of observed relative errors and X_{RxN} is the matrix of input components affecting D_{RE} (R is the number of components representing Q variables with $R \leq Q$, N is the cardinality of the data set). To fit parameters W, we can apply iterative methods such as the Backpropagation algorithm along with the Levenberg-Marquardt optimization technique [RUMELHART86], [HAGAN94]. Based on the considerations in Section 6.1, parameters W have to be fitted by minimizing the Minkowski-R distance with R = 1 (robust regression). Note that, once parameters W have been fitted, EReg can provide the X-dependent median of the relative error, Eqn. (26),

$$EReg\ (X_{RxN}, w) = Me_{RE} \qquad (26)$$

where w represents the fitted values of parameters W and Me_{RE} is the X-dependent median of the relative error on the history (X_{RxN}). A two-dimensional example is shown in Fig. 13 (solid-line).

Based on properties and characteristics of the robust regression (i.e. the X-dependent median), EReg is able to split the data set (history) into two subsets whose elements represent sets A and B, respectively. Set A is composed of any data point whose relative error is greater than the X-dependent median in that point and set B is composed of any data point whose relative error is lower than the X-dependent median in that point.

6.4.4 (Error) Discrimination Layer

The aim of this layer is to calculate the Bayesian Discrimination Function (BDF).

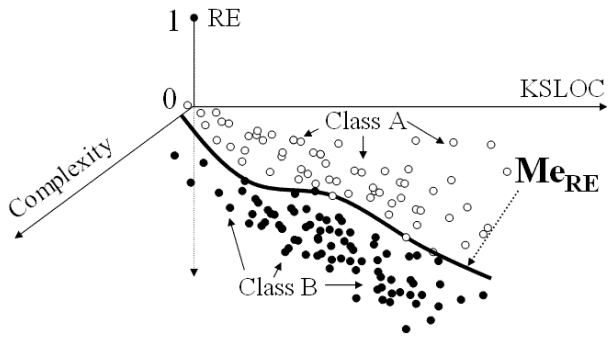

Fig. 23. A 3-D view of the EReg output before applying the CCA. RE is the dependent variable and KSLOC and Complexity are the independent variables (i.e. X1 = KSLOC and X2 = Complexity).

As illustrated above, based on the Me_{RE}, we can (1) split the sample into two subsets A and B, (2) associate target 1 with Class A and target 0 with Class B, and (3) calculate the best discrimination function BDF by applying LOOCV+CCA and minimizing the cross-entropy error function (Section 3.7).

For instance, in Fig. 23, the clear points represent Class A (where the actual RE value is greater than the Me_{RE} in that point) and the black points represent Class B (where the actual RE value is lower than the Me_{RE} in that point).

Step 7 – Select variables and complexity

The independent variables (also called features) selected as inputs to the BDF are the Q + 1 variables of Eqn. (22), $DS_{(Q+1)xN}$, i.e. Q variables (X) and the RE. The dependent variable of the BDF takes on real values in [0; 1]. For instance, with respect to Fig. 23, the BDF would be built upon 3 features (or input variables), KSLOC, Complexity, and RE.

In this step, we select the 'right' complexity for the multi-layer feed-forward neural network for discrimination problems. This selection is the same as in Step 4, except here we have a different number of input variables and the dependent variable is bounded in [0;1]. We apply the LOOCV with Curvilinear Component Analysis. Since the sought model deals with discrimination problems, the LOOCV score has to be calculated by selecting the model that can classify all of the data points in $DS_{(Q+1)xN}$ (history) with the most correct classification rate, i.e. hopefully a 100% correct classification. For instance, a 100% correct classification means that the neural network can correctly discriminate all the data points in the data set $DS_{(Q+1)xN}$ with no misclassification error. The shrunken input configurations are fed into Step 8 where we continue to calculate the LOOCV score as long as it is decreasing.

Step 8 – Look for the best discrimination model

This step is the same as Step 5, except that again we have one more input variable (i.e. Q+1) and the output variable is bounded in [0; 1]. Fig. 22 can still be applied by turning the number Q into Q+1. We receive Q+1 shrunken input configurations from Step 7 one by one and calculate, for each of them, the LOOCV score for models with increasing complexity (where the complexity is still expressed by the number of the hidden units). Again, we stop increasing model complexity when the score starts to increase (see Step 5). The procedure executes Step 7 and Step 8 cyclically until all Q+1 shrunken input configurations have been processed. In performing Steps 7 and 8, we turn the data set $DS_{(Q+1)xN}$ into a shrunken data set $X^{\dagger}_{SxN}$. That is,

$$DS_{(Q+1)xN} \Rightarrow X^{\dagger}_{SxN} \qquad (27)$$

where S (≤ Q+1) is the number of input components and N is the cardinality of $X^{\dagger}_{SxN}$.

Step 9 – Calculate the Bayesian Discrimination Function

In this step, we show the process for building the Bayesian Discrimination Function (BDF) using the shrunken input matrix $X^{\dagger}_{SxN}$.

Let Γ be a random variable (Section 3.7), which is a function of a vector of features $X^{\dagger}_{S,}$ and is equal to 1 when the input belongs to A, 0 otherwise. We build Γ by observations in $X^{\dagger}_{SxN}$ such that Γ = 1 if RE ≥ Me_{RE}, i.e. if the relative error RE on an observation in $X^{\dagger}_{SxN}$ is greater than its X-dependent median calculated by Eqn. (26). Γ = 0 otherwise.

Based on Eqn. (18), Γ is defined as follows:

$$\Gamma: (X^{\dagger}_{SxN}) \rightarrow \{0, 1\} \ . \quad (28)$$

Based on the theorem in Eqn. (18), the regression function of Γ is the expected value of Γ given x. This means that, if we calculate the regression function of Γ by minimizing the cross-entropy error function, we get the posterior probability of class A, given x. In particular, the function that we have to fit is

$$PP = BDF(X^{\dagger}_{SxN}, U) \quad (29)$$

where, PP is the known N-vector of {0, 1} given by Eqn. (28), i.e. 1 if RE ≥ Me_{RE} and 0 otherwise. The BDF is the function that calculates the regression function of Γ (representing the Posterior Probability of Class A, given x), $X^{\dagger}_{SxN}$ is the shrunken known observation matrix, and U is the set of unknown parameters, which define the BDF.

Based on the Backpropagation algorithm together with some optimization techniques such as Levenberg-Marquardt [RUMELHART86], [HAGAN94],

we can estimate U. Once U has been calculated, if we feed the BDF with X_{SxN}, we obtain

$$Pr(A|x) = E(\Gamma|x) = BDF(X^{\dagger}_{SxN}; u) \quad . \quad (30)$$

Where Pr(A|x) is the posterior probability of class A, given x with $x \in X^{\dagger}_{SxN}$, and u represents the estimated values of U.

6.5 Prediction by the BDF

We can use the BDF for both prediction and model improvement. The former takes place in the end of step 4.A, while the latter takes place in the end of step 5.B (Fig. 16). In the current section, we deal with prediction. In the next section, we will deal with improvement. As a further reference, step 4.A is represented in Fig. 19 (Section 6.3). To carry out step 4.A (i.e. making the prediction), we apply the risk exposure procedure reported in the end of Section 3.2 and shown in Fig. 8, with some additional improvements.

In Section 3.2, we argued that the analysis in Fig. 8 was incomplete. The problem was that the traditional methodologies that we used for calculating the estimate prediction intervals dealt with neither the scope nor the model error in case of regression violations. To overcome such limitations, we suggested applying the mathematical solution presented in Section 6.1, a methodology that requires an instance of the BDF, which has to be available into the EB, before the Estimation Improvement Process (EIP) can take place (Fig. 16).

6.5.1 Estimating (Bayesian) error prediction intervals

EIP starts with a project organization retrieving the BDF from the EB. To cope with assumption error, a project organization has to consider a variety of possible estimation model inputs. Assume that there are C most likely

inputs describing the project P being estimated, i.e. $I = (I_1, I_2, ..., I_C)$, where i-th element is represented by variables of (X_1, X_2, … , X_Q). The i-th instance gets the values $I_i = (x'_{1i}, x'_{2i}, \dots, x'_{Qi})$ with i = 1 to C. For instance, each element of set I may differ from the other by project size, complexity, or something else. Moreover, for each I_i, the project organization associates a subjective probability that I_i occurs for P, i.e. $Pr(I_1), Pr(I_2), ..., Pr(I_C)$. Note that, if project organization managers have no information for assigning those probabilities, then the uncertainty has to be considered as its maximum, i.e. $Pr(I_1) = Pr(I_2) = ... = Pr(I_C) = 1/C$.

We feed each element of set I into the BDF and, based on the procedure shown in Fig. 14 and Fig. 15, obtain as many (Bayesian) error prediction intervals, i.e. [$Me_{DOWN}(I_1)$, $Me_{UP}(I_1)$], [$Me_{DOWN}(I_2)$, Me_{UP} (I_2)], … , [$Me_{DOWN}(I_C)$, Me_{UP} (I_C)] (Fig. 24).

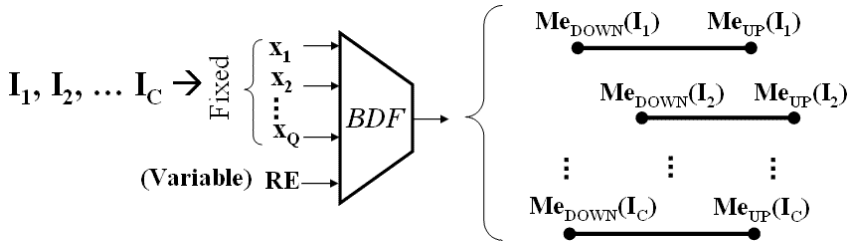

Fig. 24. Inverting the BDF. Estimating error prediction intervals from the assumed inputs by applying the mathematical solution in Section 6.1.

Each (90/95% credibility) error prediction interval can be obtained by fixing values of the variables X and letting the RE vary as explained in Section 6.1 and shown in Fig. 24.

6.5.2 Scope error evaluation algorithm (similarity analysis)

Since we now use the BDF, the magnitude of the error prediction intervals takes into account any kind of model error and we can figure out whether or

not inputs in the set I will bring about a scope error. In Section 3.2 (Table 3), we described two different problems with scope error. The first involved a project P2 whose characteristics had been partially observed previously in projects A and B (Table 2). The second involved a project P3 never observed before. Situations 2 and 3 in Fig. 25 show how to deal with both kinds of scope error.

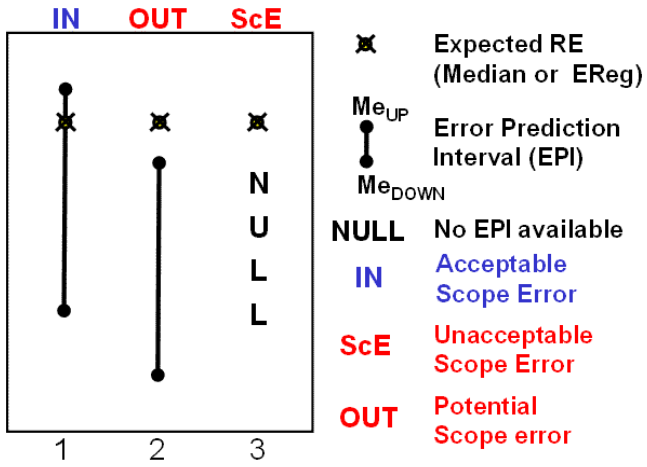

Fig. 25. BDF reliability and scope error analysis.

In situation no. 1 (Fig. 25), the error prediction interval provided by the BDF includes the expected relative error RE, where the expected relative error is provided by the RE median or the EReg (Section 6.4, Step 6). Therefore, the BDF can be considered reliable and the error prediction interval accurate (the scope error is acceptable). In situation no. 2, the error prediction interval provided by the BDF does not include the estimated relative error. Then, it can be a warning that the BDF is unreliable and the error prediction interval is inaccurate, i.e. the scope error may be unacceptable. Actually, we are not sure whether a scope error may happen. We have to check the actual values once they are available (see Section 6.6).

Situation 2 may show that, the data set has an insufficient number of observations to allow the building of the BDF. In that case, we would have a scope error. Conversely, the actual observations may exclude this conjecture when the actual relative error would fall within the interval. This may happen if the x-dependent median (EReg) or the median that we use to split up the RE sample would not be accurate. In that case, there would not be any scope error. For prediction purposes, however, before knowing the actual values, we should consider situation 2 as a potential scope error.

Note that, situation no. 2 (Fig. 25) can be improved. We can make the error prediction interval accurate even though the BDF stays biased. It can be done by increasing the (upper or lower) interval endpoint beyond the expected RE so that the interval can contain the expected RE itself. This kind of improvement is useful only if the magnitude of the final error prediction interval is acceptable in the sense of Fig. 7. If the magnitude exceeds the acceptable threshold, this correction is not recommended. Therefore, situation no. 2 may not be a problem because it can be turned into situation no. 1 (increasing uncertainty).

With respect to situation no. 3, there is no error prediction interval to use, i.e. the BDF is not able to provide any interval. This happens when the historical data used for building the BDF (Section 6.4) did not include data points similar to the project being estimated. The Backpropagation algorithm cannot generalize information if no information is available [DREYFUS05]. Therefore, we are sure that a scope error will occur in situation 3. Mathematically, this case is not about “overfitting” [DREYFUS05]. Technically, the point here is that the classification capabilities of a neural network decrease when it is no longer able to provide significant results. Usually, researchers and practitioners use this characteristic to design new experiments and gather new information. We exploit this characteristic to detect and evaluate the scope error impact on the relative error.

Based on Fig. 25 we can perform a scope error analysis for each error prediction interval in Fig. 24. Based on the Jørgensen's strategy [JØRGENSEN03], i.e. getting an estimate prediction interval applying Exn. (9), we can turn the error prediction intervals of Fig. 24 into estimate prediction intervals (Fig. 26), i.e. we calculate $[O^{N+1}_{est\ DOWN}(I_1), O^{N+1}_{est\ UP}(I_1)]$, $[O^{N+1}_{est\ DOWN}(I_2), O^{N+1}_{est\ UP}(I_2)]$, …, $[O^{N+1}_{est\ DOWN}(I_C), O^{N+1}_{est\ UP}(I_C)]$.

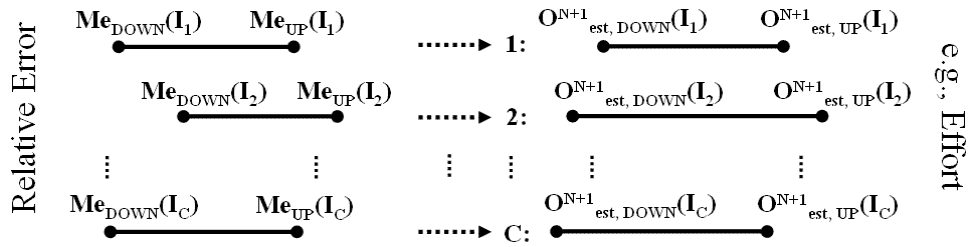

Fig. 26. Turning error prediction intervals into estimate prediction intervals.

This operation takes the bias out of the model by correcting the estimates, as explained in Section 3.2. Removing the model bias from the estimates by applying Exn. (9) improves the estimates (not the model). Of course, some of the intervals in Fig. 26 may not be available because of scope error.

We should apply now the procedure shown in Fig. 8 to get the ultimate estimate, i.e. the one mitigating the estimation risk.

6.5.3 Risk exposure analysis

After performing the procedure in Fig. 26, assume we have the error prediction intervals in Fig. 27, with C = 4. That is, assume we have four possible inputs describing project P with four different probabilities, respectively, as shown in Fig. 27. Note that, input no. 3 has no estimate prediction interval because of the scope error. Nevertheless, it has a 15% probability of occurring.

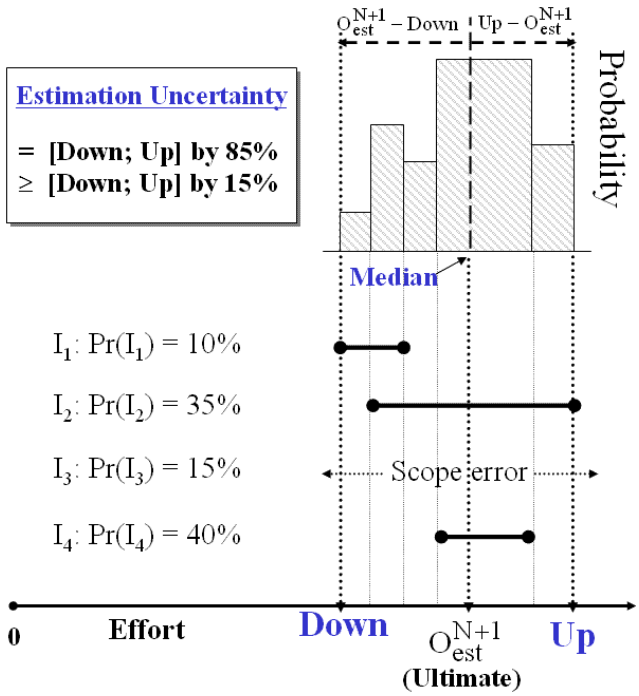

Fig. 27. Risk exposure analysis.

We can now apply risk exposure analysis. We build the histogram (and so eliciting the probability) of each segment obtained by projecting each interval endpoint onto the x-axis. Based on the risk mitigation strategy of the organization, we choose the ultimate estimate for project P, i.e. o_{est}^{N+1}. In Fig. 27, we chose the estimate corresponding to the median. This means the risk exposure analysis of the organization is aimed at having the same probability for both underestimates and overestimates.

If the organization wants to reduce the risk of underestimating (usually the most severe [MCCONNELL06]), we would choose a value greater than the median (i.e. on the right side with respect to the median). To make the risk of underestimation minimum, we would choose the value corresponding to the point "Up". As we have already explained, we may apply different risk

mitigation strategies. Nevertheless, since an underestimate is usually considered more severe than an overestimate [MCCONNELL06, pp. 21-26], we should pick an estimate in [Median, Up] (Fig. 27).

In Fig. 27, we considered a 15% chance of a scope error happening as acceptable. Suppose now the organization decides that anything greater than a 10% scope error is unacceptable, then this estimation model would be too risky. Then they should change the approach, e.g., use a human based approach, until additional historical data is available. Then they can revert to this approach, calibrate another model, and check out the acceptability level.

It is worth noting that unlike traditional methodologies, the presented approach is able to improve the performances of parametric estimation models by taking into account model error regression violations, such as scope error, and assumption error at the same time. Moreover, the approach is able to signal in terms of risk whether the available historical data is sufficient for building the parametric estimation model or not.

6.6 Model improvement using the BDF

Once actual data is available, we can check whether the prediction was accurate enough. In particular, whether the relative error, i.e. RE = (Actual Effort – o_{est}^{N+1})/ Actual Effort (where o_{est}^{N+1} is the ultimate estimate, Section 6.5), fell into the interval [Down; Up], Fig. 27. Such an analysis, however, cannot be used for model improvement. To improve the estimation model, we need to evaluate the model behavior using actual values.

To perform such an improvement analysis, we exploit the BDF, the expected RE (Median or EReg), and the actual RE. Note that, the actual RE is different from the actual relative error calculated by the ultimate estimate (see above). As we explained in Section 6.3 through Fig. 20, we first feed the actual input I_{act} into the BDF and than execute the analysis.

6.6.1 Scope extension algorithm (posterior similarity analysis)

Since there is no uncertainty with respect to the input values (even though we might get some measurement error, see Fig. 9), we do not consider the assumption error anymore, i.e. once actual data is available, we deal with model error and scope error. The improvement analysis starts with evaluating the scope error (Fig. 28).

Situations 1.a and 1.b in Fig. 28 refer to situation 1 in Fig. 25, where the expected RE falls within the interval. In situation 1.a there is no scope error (i.e. no scope extension). Both the expected RE and the actual RE fall within the error prediction interval. Rebuilding a new instance of the estimation model including the actual project data in 1.a (i.e. the procedure explained in Section 6.3 Fig. 17) does not extend the scope of the EM.

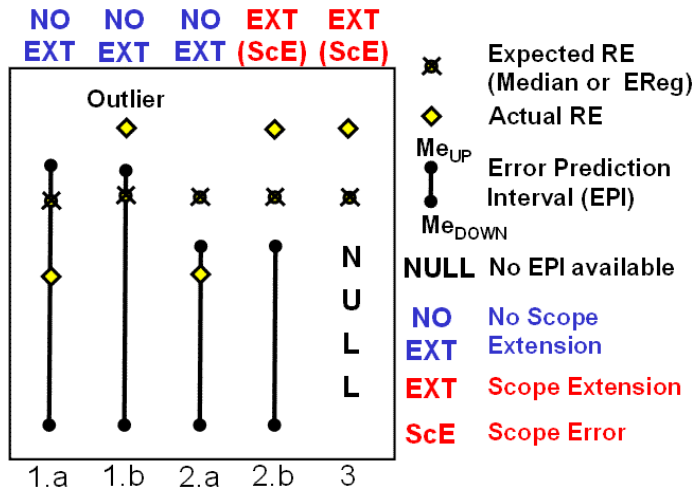

Fig. 28. Scope extension analysis on Fig. 25.

In situation 1.b, there is also no scope error. This is because the error prediction interval (EPI) is reliable (the BDF was correctly built), but the actual RE falls outside the interval. This situation is similar to the traditional evaluation of outliers in statistics. If we included the project in 1.b in the

training set when building a new version of the EM, we might observe an inclusion of the expected RE within the interval, but the model would not extend its scope. Therefore, situation 1.b would lead to increasing the uncertainty of the EM, event though it may make the error prediction interval closer to reality.

Therefore, we should decide whether it is worth including that project data in the training set for building the next version of the EM. In fact, situation 1.b may imply that some kind of model error would affect the EM unacceptably. In that case, we may decide not to include the project data in the next building procedure of the EM since we would not remove the model error (e.g., finding the missing variables, the right model complexity). However, when we have too "few" data points to build the EM, as is usually the case in software engineering, the best we can do is to use projects like those of 1.b to build a new version of the EM.

Situations 2.a and 2.b in Fig. 28 refer to situation 2 in Fig. 25, where the expected RE falls outside the interval. In situation 2.a, we can make sure that no scope error occurs because the actual RE falls within the interval even though the interval does not include the expected RE. This means that the BDF was biased because of the median rather than for a lack of observations. Therefore, situation 2.a is similar to situation 1.a apart from the fact that the EReg (or median) used for splitting up the RE sample did not represent correctly the expected value of the RE.

Situation 2.b refers to situation 2 in Fig. 25. Unlike situation 2.a, in situation 2.b, the actual RE falls outside the interval. This situation occurs not because the EReg (or the median) did not correctly represent the expected value, but the problem is that there are not sufficient observations to build the BDF. Therefore, situation 2.b refers to a scope error because of this lack of observations. Rebuilding a new instance of the estimation model with that data point extends the scope of the EM and we would observe a shorter

distance between the actual RE and the interval, or possibly an interval that includes the RE. Note that, this kind of improvement would increase the EM uncertainty making the prediction interval more credible.

Situation 3 in Fig. 28 refers to situation 3 in Fig. 25. We duplicated it for the sake of completeness. As we have already explained in Fig. 25, situation 3 leads to increasing the scope of the model. This happens when there is no interval because the data set did not have data similar to the considered project. Therefore, if we rebuilt the EM by including the project in the training set, we would observe a scope extension. Of course, including only one data point may not be sufficient. Nevertheless, situation 3 leads to extending the EM scope.

6.6.2 Model-error improvement algorithm

To check whether the model error affects the EM, we exploit the error prediction intervals with respect to the zero point. In particular, Fig. 29 shows four cases (4, 5, 6, and 7) arising from both situation 1 and 2 in Fig. 25.

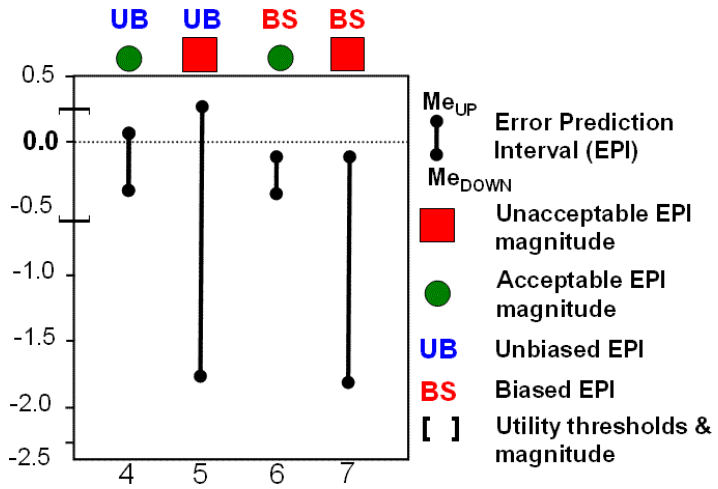

Fig. 29. Model error analysis on situations no. 1 and 2 in Fig. 25.

Situation 4 (Fig. 29) does not need improvement because the model error is within the utility thresholds and the model is unbiased (it includes zero), see also Fig. 7. Note that, situation no. 4 does not mean that the estimation model cannot be further improved. We can improve the model as far as the prediction error is different from zero. In situation 4, we just mean that the uncertainty is acceptable for the organization goals.

In situation 5, the estimation model remains unbiased even though the magnitude of the error prediction interval is unacceptable. Improvement involves shrinking its magnitude. To do that, we can try to include some dummy variables into the estimation model [BISHOP95A, p. 300], [KEUNG08]. Dummy variables are (dichotomous) categorical variables that group the available data points into distinct subsets. To check whether the improvement with dummy variables has been effective (the interval is shorter), we can apply the procedure in Fig. 17 and the analysis in Fig. 21.

In situation 6, the estimation model is biased. That is because of one or more missing variables (i.e. we did not consider some relevant independent variables affecting the dependent variable of the estimation model) or the estimation model is not flexible enough. To improve the model we can try to increase the model flexibility by increasing the degree of the polynomial, the number of the hidden units for neural network, or we may apply a logarithmic transformation (log-linear regression). Note that, if the estimation model is linear in the parameters we can turn it into a non-linear-in-the-parameter model as explained in Section 3.1 and subsequently increase its flexibility. If such an improvement falls short of improving the model, the only thing that we can do to improve it is to find the right independent variables affecting the dependent variable. If we do not find the right variables, the model cannot be improved. Note that, due to the analysis in Fig. 29, we are able to identify the EM improvement needs. To find the right variables we can use previous research, experts, or context analysis (Section

3.1). Situation 1.d can be improved by applying the improvements of both situation 1.b and 1.c.

It is worth noting that, since situations 1 and 2 in Fig. 25 and situations 4, 5, 6, 7 in Fig. 29 are independent of each other, the model and scope errors may affect the estimation model at the same time. As an example, if the project were classified as situation 2.b in Fig. 28 and situation 7 in Fig. 29, we would infer that the EM would be affected by scope error and model error (biased and unacceptable spread) at the same time.

6.7 Discussion

When applying Exn. (9) to turn the error prediction intervals into estimate prediction intervals (see Fig. 26), we actually make estimates unbiased since we deal with a relative measure of error. However, it only happens when the error prediction interval is reliable and unbiased (situations 4 and 5 in Fig. 29). In situations 6 and 7, applying Exn. (9) may not make the estimates unbiased. The real problem for prediction that we have to worry about is to shrink error prediction intervals. This is one of the real benefits of using relative errors for evaluating the estimation model uncertainty proposed by Jørgensen et al. [JØRGENSEN03].

Note that, considerations in Fig. 29 are similar to the ones made in Fig. 7. There are some differences, however. The main difference is that the traditional methodologies used in Fig. 7 cannot deal with the scope error quantitatively, i.e. they are not able to discriminate among all of the situations in Fig. 28. Moreover, error prediction intervals in Fig. 7 have been calculated without considering any regression violation. So, if we used traditional methodologies as in Fig. 7 instead of applying the proposed approach, we would confuse the scope error with the model error and we would not be able to improve the model.

To avoid such a misleading situation, many researchers suggest not undertaking the development of projects never dealt with before, i.e. projects where a scope error may happen. As we have already explained in Section 3.2, Kitchenham et al. [KITCHENHAM97] suggest applying the portfolio concept to overcome this problem. While, Jørgensen et al. deal with this issue by assuming that they are able to select historical projects on which the estimation model has the same accuracy [JØRGENSEN03]. Angelis et al. [ANGELIS00] deal with the problem by applying the bootstrap method. Once they select similar projects to make a baseline for prediction, the selected data points may be not enough to make any prediction, i.e. calculating statistics [KIRSOPP02B]. Then, a resampling procedure such as bootstrap may somehow enlarge the data set for calculating and making significant the required statistics.

We argue that, (1) the portfolio concept cannot be applied to every situation hence it is not a solution to the scope error. (2) The Jørgensen et al.'s assumption stated above might lead to taking very different projects into account so the error prediction interval may be incorrectly calculated. (3) The bootstrap method cannot make sure that the error prediction interval is correct hence we may base our inference on wrongly assumed uncertainty.

Conversely, the proposed approach overcomes such limitations and assumptions by using a particular kind of multi-layer feed-forward neural network (i.e. the BDF), which is able to detect and discriminate between scope error and model error. Moreover, the proposed approach exploits benefits of parametric estimation models (e.g., estimates come from mathematical applications, the process is traceable and repeatable) and avoids their drawbacks (i.e. unreal parametric assumptions).

Chapter 7

A practical application

7 The Case Study

In this section, we apply the approach to the NASA COCOMO data set [PROMISE] by considering different models (e.g., linear, log-linear, and non-linear). The analysis aims at demonstrating the application of the proposed approach using real data. Conclusions enacted from this case study offer some new insights for the projects developed at NASA.

Regarding the improvement procedure, we focus on both improving accuracy and decreasing the uncertainty of the estimation model. The best model will be the one having the best accuracy and the least uncertainty among those considered. We first build a linear model. Because this shows to perform poorly on both aspects (i.e. accuracy and uncertainty), we improve the model by considering a log-linear transformation. The resulting model performs very well. To improve the log-linear model from an uncertainty point of view, we include some categorical variable as independent variables. Since the nature of the approach is to evolve the model over time by adding the results of new projects, we show that further rebuilding the log-linear model with new non-outlier projects (see Fig. 28) does not make the uncertainty worse. But rebuilding the log-linear model with new outlier projects makes the model more risky; a price we may have to pay if we are moving into new territory with the projects we are developing. We stop improving the model because we have use up all of the variables in the considered data set. In real cases, however, we would continue improving the model from an accuracy and uncertainty point of view trying out new variables and models.

An important consideration arising from this study is that, the uncertainty analysis proposed above is useful for model selection as well. In particular, along with the traditional model selection techniques based on accuracy, the one that we propose in this work is based on selecting the model that shows the least uncertainty among the available models. Note that, the uncertainty comparison has to be performed over a fixed test set of projects (16 projects in this study). Instances of models are linear, log-linear, non-linear, and generalized (e.g., non-linear in the parameters).

7.1 The Context

Consider the situation where NASA is the learning organization [BASILI92B] using measurement [BASILI94A] and, from 1971 to 1987, they developed 8 projects, e.g., Hubble Space Telescope, involving 93 software systems. We start with our analysis at the beginning of 1985, when NASA has already developed 77 software systems so their experience is based upon those 77 software systems, which we will use as the basis for prospective evaluations, risk analysis, and model improvement. In fact, the NASA's goal is to exploit such experience to evaluate uncertainty in estimating the effort of the 16 next software systems (from 1985 to 1987). Since we actually know the data from these 16 software systems, we can use them to carry out an overall iteration of the proposed framework as explained in Section 6.2 (Fig. 17).

As a learning organization, NASA also has the goal of shortening the risk and improving their models to better manage resources such as time, personnel, and budget.

7.2 Applying the framework

Based on Fig. 18, we now apply the procedure to build the first instance of the BDF as explained in Section 6.3.

GQM template: *Analyze* the uncertainty (risk) of a linear regression model *for the purpose of* evaluation *with respect to* the Relative Error (RE) from the *point of view* of the project managers in the *context of* NASA's projects. This goal focuses our study.

7.2.1 Preconditions

Historical data is available according to the COCOMO-I variables. We call instances of the data set "data points" or "projects". Note that, what we call a project or data point is actually the project undertaken to develop an individual software system. This name must not be confused with the NASA's projects, which are only 8 (de, erb, gal, X, hst, slp, spl, Y), and include several software systems.

7.2.2 (Error) Data layer

The data set [PROMISE] is composed of 93 project instances. Each instance is described by 24 attributes (
Table 4). In particular, "Size", 15 COCOMO-I multipliers, "Effort", and 7 attributes describing further characteristics of the NASA software system (project ID, project name, category of application, flight/ground system, NASA center, "YEAR" finished, and development mode). Note that, "Effort" is measured by calendar months of 152 hours, including development and management hours [BOEHM81].

For demonstration purposes, we start with a simple linear model and will demonstrate how to identify risks and make improvements. To calibrate the linear model we start by considering only numerical variables. It is also possible to include categorical variables into the model as we explained in Section 6.6. The problem is that, regression models cannot deal with categorical variables. They have to be coded first. A common mistake is to use an ordinal value for nominal scale. For example, consider the attribute "NASA center" in

Table 4.

TABLE 4

DATA SET DESCRIPTION

Attribute	Description	Example
Size	Continuous	112.3 KSLOC
15 COCOMO-I Multipliers	Discrete (Ratio)	"very low" = 1.46
Effort	Continuous	117.2 Man-Months
ID	Categorical	1, 2, … 101
Project name	Categorical	"gal" = Galileo
Category of application	Categorical	"avionics", "science"
Flight/Ground system	Categorical	"F" or "G"
NASA center	Categorical	"center 3"
Development mode	Categorical	"embedded"
YEAR	Discrete (Interval)	1986

If we included an ordinal variable (e.g., 1, 2, 3 …) in the regression model to describe the NASA centers, the parameters of the resulting regression model would be biased because the training procedure would create parameters for an ordered variable, while that variable has no order at all. The right way of coding categorical values is to use dummy (dichotomous) variables. If we have C categorical values, we create (C – 1) dichotomous variables, i.e. the variable can have values 0 or 1. For instance, if we have 5 NASA centers, then "NASA center 1" becomes "0001", "NASA center 2" becomes "0010" and so on up to "NASA center 5", which becomes "0000" [BISHOP95A], [KEUNG08]. If we considered irrelevant categorical variables, however, the regression model would increase in complexity (i.e. a greater number of parameters) and it would be less parsimonious. Thus, we should include dummy variables with care.

Note that, we only use the attribute "YEAR" to split up projects, i.e. the first set (before 1985) was composed of 77 software systems, and the second set (after 1984) was composed of 16 software systems. In particular, the first set was considered as a training set (history) and the second set was considered as a test set (what being estimated), i.e. the object of our analysis.

Step 1 – Consider an estimation model

The estimation model selected is a linear regression model trained by Ordinary Least Squares (OLS).

TABLE 5

OLS ESTIMATES

Parameter	Estimate	SE	Statistic	p-value
CONSTANT	-1159.49	4517.32	-0.256677	0.7983
KSLOC	5.47988	0.882886	6.20678	0
rely	-202.032	1140.47	-0.177149	0.86
data	1291.4	1950.58	0.662061	0.5105
cplx	-396.674	779.727	-0.508734	0.6128
time	3193.19	954.538	3.34527	0.0014
stor	385.939	753.358	0.512292	0.6103
virt	-1416.19	1443.69	-0.98095	0.3306
turn	-1516.94	1629.15	-0.931126	0.3555
acap	3300.09	1707.83	1.93233	0.058
aexp	-662.109	2153.1	-0.307515	0.7595
pcap	-762.806	1402.92	-0.543729	0.5886
vexp	-5444.19	1667.39	-3.2651	0.0018
lexp	5219.86	3064.15	1.70352	0.0936
modp	-2339.76	1434.65	-1.63089	0.1082
tool	-598.167	1720.57	-0.347656	0.7293
sced	962.486	3615.38	0.26622	0.791

It is based on input variables from the COCOMO-I model where the dependent variable (DV) is the effort and the independent variables (IVs) are the size and 15 COCOMO-I multipliers. Based on the training set, we applied OLS and obtained the parameter estimates in Table 5.

The analysis of variance (ANOVA) shows that there is a statistically significant relationship between the IVs and the DV at 99% confidence level (p-value = 0.0000). The R-Squared statistic indicates that the model explains 66.4% of the variability in the DV. The adjusted R-Squared statistic is 57.5%. The standard error (SE) of the estimate shows the standard deviation of the residuals to be 796.56. The model may be simplified by removing the parameters having a p-value greater than 0.10 in Table 5 (e.g. rely p-value = 0.86). Although applying stepwise regression may simplify the model, it would require the assumption that multicollinearity does not affect the model as explained in Section 3.1. For this reason, we do not apply the stepwise regression.

TABLE 6

PREDICTION ON THE TEST SET

EST	125.3	-199.6	-161.1	75.8	152.2	183.4	-133.7	171.9	424.1	234.0	44.0	203.2	256.9	139.4	460.2	398.5
ACT	18	42	42	48	50	36	60	60	90	114	155	60	120	210	300	444
RE	-6.0	5.8	4.8	-0.6	-2.0	-4.1	3.2	-1.9	-3.7	-1.1	0.7	-2.4	-1.1	0.3	-0.5	0.1
ID	33	36	39	26	34	13	37	35	24	40	52	21	23	25	22	38

We can now use the model for prediction by feeding the test set into the model. Table 6 shows the results, where EST stands for “Estimated Effort”, “ACT” stands for “Actual Effort”, RE stands for “Relative Error”, and “ID” stands for software system identifier. Notice that, Table 6 contains results that are ordered based on their size. From a software engineering point of view, the estimation model (EM) provides three unreliable results, IDs 36, 39, and 37 because they are negative value while the effort may only be a positive value. From a mathematical point of view, these values are correct and they must not be considered as outliers in statistical terms. The implication is that, the model cannot be used for predicting the effort of those projects. Consequently, from a software engineering point of view, the RE

value for those projects makes no sense even though it is correctly calculated from a mathematical point of view.

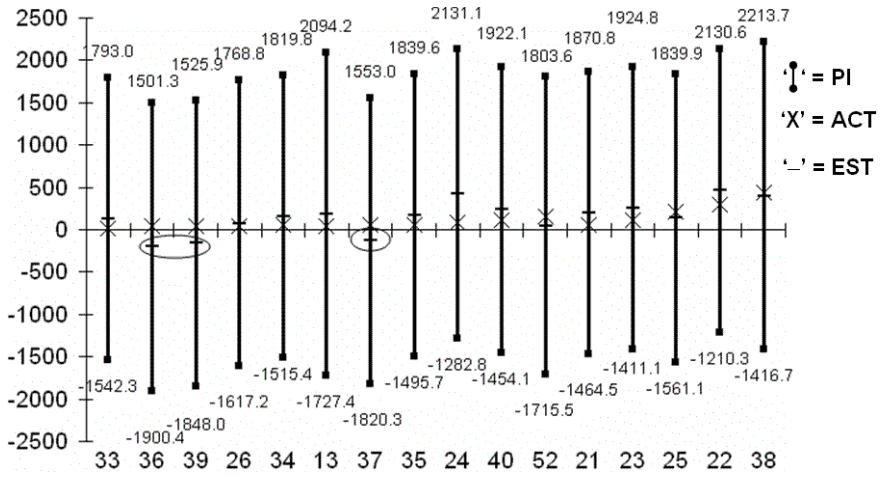

Fig. 30. Prediction intervals for the test set in Table 6 obtained by applying the canonical formula. The circled values of EST represent negative estimates that cannot be used for prediction.

Based on formula (12), we calculate the prediction intervals for the estimates in Table 6. As Fig. 30 shows, the prediction intervals are definitively huge and they do not have any utility for evaluating uncertainty and improvement needs as explained in Section 3.1. Since effort cannot be negative, the prediction intervals in Fig. 30 can be shrunk by ignoring the lower negative limit. Even though we considered prediction intervals having their lower limit on zero, the situation would not change, because the interval would be too wide, as well. Further, having an estimated effort equal to zero would make no sense either.

From a prediction point of view, the situation does not change. The prediction intervals do not provide any utility because the magnitude is too high. Note that, traditional approaches stop the analysis at this point. Fig. 30 aims at showing that, in evaluating risk and uncertainty arising from a

parametric estimation model, we cannot rely upon traditional statistics. We need better techniques that are able to show the real improvements that the estimation model requires as we explained in Section 6.1.

Step 2 – Select a (relative) error measure

We selected RE as a relative measure of error.

Step 3 – Calculate the error on the history

To calculate the RE on the historical data points (i.e. the training set composed of 77 data points), we feed the training set into the EM.

TABLE 7

RELATIVE ERROR SAMPLE (RE) ON THE TRAINING SET

ID	RE	ACT	EST
58	-10.9	8.4	100.1
6	-11.8	8.4	107.3
101	-6.94	38	301.6
7	-9.59	10.8	114.4
11	16.22	24	-365
100	-25.6	12	319.1
97	1.456	648	-295
9	-24.5	72	1838
3	-3.4	31.2	137.4
4	-2.89	36	140.1
5	-4.89	25.2	148.4
51	0.992	98.8	0.744
30	-0.38	62	85.79
19	7.581	48	-316
27	-0.43	70	100
29	-0.28	82	105
99	0.22	480	374.5
10	3.312	72	-166
15	7.011	48	-289
74	1.321	240	-77.1

ID	RE	ACT	EST
49	-1.49	107	266.2
89	0.162	430	360.4
2	-0.96	117.6	230
1	-1.02	117.6	237.1
18	5.751	60	-285
81	-0.45	1350	1952
20	3.838	60	-170
31	-0.14	170	194.3
71	1.832	72	-59.9
32	-0.09	192	210.2
64	2.076	150	-161
84	0.243	599	453.6
48	2.332	252	-336
28	-0.18	239	281.4
55	-0.13	370	418.4
94	-0.28	1925	2454
82	-1.53	480	1212
79	-0.31	409	537.2
92	-0.51	1773	2675
8	-0.3	352.8	460.2

ID	RE	ACT	EST
57	1.3	278	-83.4
61	-0.73	458	791.6
93	-0.76	1646	2899
50	0.36	571.4	365.7
45	-0.62	400	647.2
73	0.01	300	297
63	0.305	162	112.7
72	-0.23	300	368.3
12	0.245	360	271.9
14	0.344	215	141.1
16	-1.54	360	915.7
80	-0.08	703	756.4
53	0.415	750	438.7
75	0.297	600	421.5
65	-0.09	636	691.5
69	-0.02	576	590.1
17	-1.35	324	762.1
66	0.748	882	222.6
70	-0.48	432	640.7
76	-0.18	756	895.3

ID	RE	ACT	EST
78	-5.7	97	645.6
91	0.2	4178	3223
41	0.2	1248	1059
47	0.1	420	371.1
54	0.4	2120	1345
56	-0.3	1181	1572
98	0.6	8211	3595
68	-2.8	192	725.2
62	0.2	2460	1893
43	0.1	1368	1213
44	-0.3	973	1249
42	0.2	2400	1897
67	-2.9	444	1748
60	-1.1	720	1512
77	-0.5	1200	1774
46	0.2	2400	1888
59	-0.1	4560	5072

Once we obtain the estimates, we apply the Eqn. (5). The sample error that we obtained is shown in Table 7. In particular, the RE sample is ordered according to the project size. “ID” refers to the original data set identifier, “RE” is the relative error, “ACT” is the actual effort, and “EST” is the estimated effort. Notice that, we highlighted with dark rectangles (Table 7) the data points where the EM provided negative values of effort. As

explained above, since our analysis is based on as few assumptions as possible, we decide not to remove these points from the data set. The reason for this choice is that, these data points are correctly calculated even though they make no sense from a software engineering point of view. Therefore, the actual behavior of the estimation model in terms of estimation error over the history has to be evaluated on these data points, as well.

7.2.3 (Error) Regression layer

In this layer, the aim is to calculate the expected (relative) error arising from the estimation model over the historical data. As we have explained in Section 4, assumptions on which the EM is based may not hold (e.g., the error is x-correlated). Then, a better measure of the expected RE would be the output of a robust regression function having as a DV the RE and as IVs the x variables. Therefore, we apply the procedure explained in Section 6.4 (Step 4) where we called such a robust non-linear regression EReg, Eqn. (26).

Step 4 – Select variables and complexity

In order to select the model having the lowest generalization error, we calculate the Leave-One-Out Cross-Validation (LOOCV) score, as explained in Section 6.4 (Step 4).

Step 5 – Look for the best regression model

Here we find the best regression model for estimating RE, i.e., EReg. Fig. 31 shows the results. In particular, the non-linear model having the lowest generalization error (best) is the one composed of 4 hidden units and 16 input components. The result in Fig. 31 is confirmed by the fact that, we have also calculated the LOOCV score for a linear-in-the-parameter model (linear polynomial), which provided the value 0.86, which is greater than the best value (0.56).

The number of iterations to generate Fig. 31 was 5856 (with R = 6, the procedure is explained in Section 6.4, Step 4) and it took about 2 hours with an ordinary laptop.

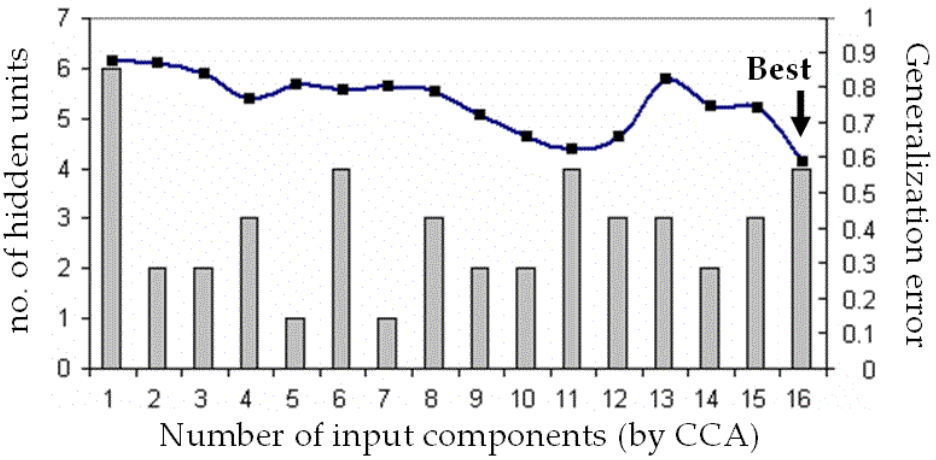

Fig. 31. Leave-one-out cross validation and curvilinear component analysis applied to a non-linear model based on Multi-Layer Feed-Forward Neural Networks for regression (EReg).

Step 6 – Estimate the x-dependent error

Based on the result in Fig. 31, we selected the non-linear model based on neural networks having 16 input components and 4 hidden units.

Then, we calculated the expected RE by applying a robust regression, which provided an x-dependent median. Fig. 32 shows the results. The straight line in Fig. 32 is a line that points out whether the model estimates are valid (RE = 1). In particular, if an RE data point (Actual RE) falls above this line, it means that that point comes from a negative effort estimates and it is invalid. If the RE data point falls below the line, the effort estimate from which it comes is valid (i.e., the estimate is positive). Note that, there are 12 data points falling above the straight line as shown in Table 7 by the dark rectangles.

In Fig. 32, we drew the median of the RE, as well (dashed line). The median seems to be a better representative of the expected RE than the x-dependent median calculated by the robust regression because the REs are not biased with respect to the x-axis. For this reason, we use the median for splitting up the relative error sample, not the x-dependent median. It is worth noting that, even though the error sample seems not to be biased with respect to the KSLOC (it is very close to zero), the assumption on the homoscedasticity is definitively violated. In fact, the error variance decreases as the KSLOC increases.

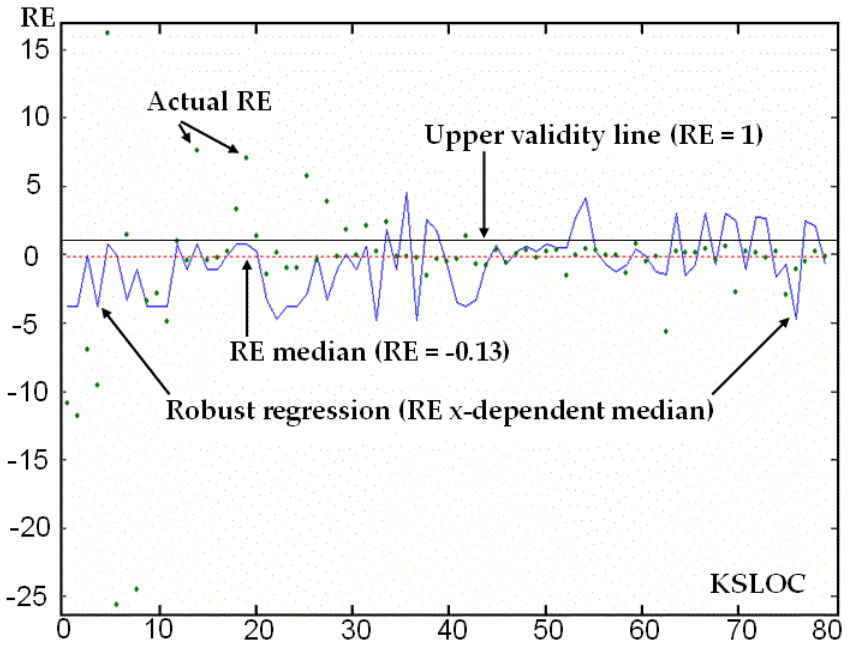

Fig. 32. Calculating the expected relative error (RE).

Based on the median (dashed line in Fig. 32), we calculate the target values, as follows. If the actual RE is not less than the median then the target value is (Γ =) 1, it is zero otherwise (Γ = 0). Table 8 shows the target values obtained, see Eqn. (18) in Section 3.7. Note that, the "ID" row in Table 8 is the project identifier of the data set (

Table 4) ordered by the KSLOC.

TABLE 8

VALUES OF THE RANDOM VARIABLE Γ (TARGET)

Γ	0	0	0	0	1	0	1	0	0	0	0	1	0	1	0	0	1	1	1	1
ID	58	6	101	7	11	100	97	9	3	4	5	51	30	19	27	29	99	10	15	74

Γ	0	1	0	0	1	0	1	0	1	1	1	1	1	0	0	0	0	0	0	0
ID	49	89	2	1	18	81	20	31	71	32	64	84	48	28	55	94	82	79	92	8

Γ	1	0	0	1	0	1	1	0	1	1	0	1	1	1	1	1	0	1	0	0
ID	57	61	93	50	45	73	63	72	12	14	16	80	53	75	65	69	17	66	70	76

Γ	0	1	1	1	1	0	1	0	1	1	0	1	0	0	0	1	1
ID	78	91	41	47	54	56	98	68	62	43	44	42	67	60	77	46	59

7.2.4 (Error) Discrimination layer

The aim in this layer is to build the discrimination function that we called BDF. As explained in Section 6.4, the BDF is a non-linear function based on multi-layer feed-forward neural networks having a real variable ranging in [0;1] as a dependent variable (DV) and the variables (KSLOC, 15 COCOMO multipliers, RE) as independent variables (IVs). To calibrate such a BDF, we consider the target values in Table 8 as observations of the DV and consider the COCOMO NASA data set together with the RE values in Table 7 as observations of the IVs. Then, we apply the Backpropagation together with the Levenberg-Marquardt algorithm to obtain the parameters of the BDF. Before applying the Backpropagation, we select the best non-linear model by executing LOOCV and CCA, as explained below.

Step 7 – Select variables and complexity

To select the non-linear discrimination model having the lowest generalization error, we calculate the LOOCV score, as explained in Section 6.4 (Step 7).

Step 8 – Look for the best discrimination model

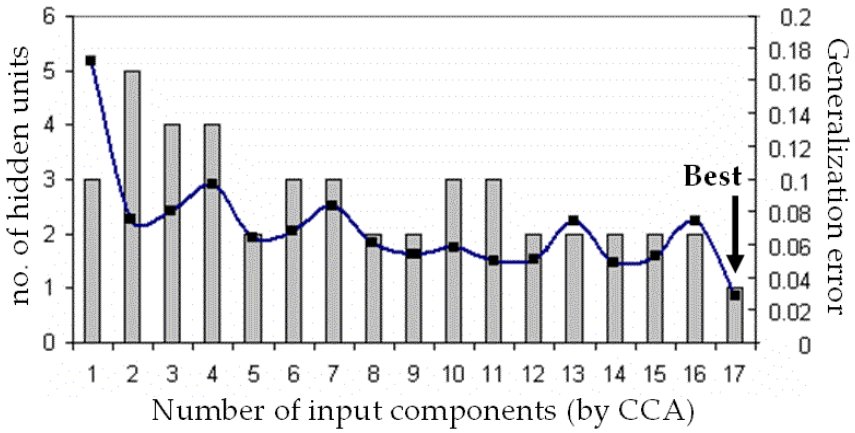

Fig. 33. Leave-one-out cross validation and curvilinear component analysis applied to a non-linear model based on Multi-Layer Feed-Forward Neural Networks for discrimination (BDF).

Fig. 33 shows the results. In particular, the non-linear model having the lowest generalization error (best) is the one composed of 1 hidden unit and 17 input components. The best generalization error obtained by executing the procedure was 0.04.

Step 9 – Calculate the Bayesian function

Based on the result in Fig. 33, we select the non-linear model based on neural networks having 17 input components and 1 hidden unit.

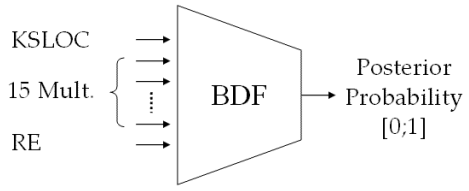

Fig. 34. Representation of the BDF as a non-linear function of 17 independent variables. The BDF provides a measure in between [0;1] of how far the input is from the median. Mathematically, the BDF provides the posterior probability that the input is not less than the median (i.e. the posterior probability of class A, Fig. 14).

Then, we calculate the BDF by minimizing the cross-entropy error function by the Backpropagation together with the Levenberg-Marquardt algorithm (Fig. 34).

We stopped the training procedure when all of the data points in the training set were correctly classified. Notice that, a data point is correctly classified when the BDF provides a value greater than or equal to 0.5 for the target "1" and less than 0.5 for the target "0". Other stopping techniques may be applied (e.g., validation-set stopping technique).

7.2.5 Prediction by the BDF

The BDF that we calculated by applying the 9-step framework can be used both for prediction and model improvement. Nevertheless, we show the use of the BDF for prediction partially because the COCOMO NASA data set does not include estimated inputs. Since we know the actual values of the projects being estimated, there is no assumption error on the inputs. Therefore, we execute analysis in Fig. 25, but we do not execute analysis in Fig. 27 (Risk exposure analysis). As we explained in Section 6.5, before using the BDF, we have to invert it, so that the BDF yields the RE range where the error of the next estimate will probably fall (i.e., what we called Bayesian error prediction interval in Section 6.1).

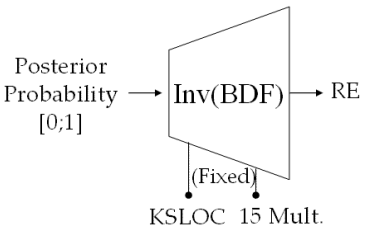

Fig. 35. Representation of the inverted BDF of Fig. 34.

Based on the Inv(BDF) in Fig. 35, we can calculate a 95% (Bayesian) error prediction interval for each project belonging to the test set.

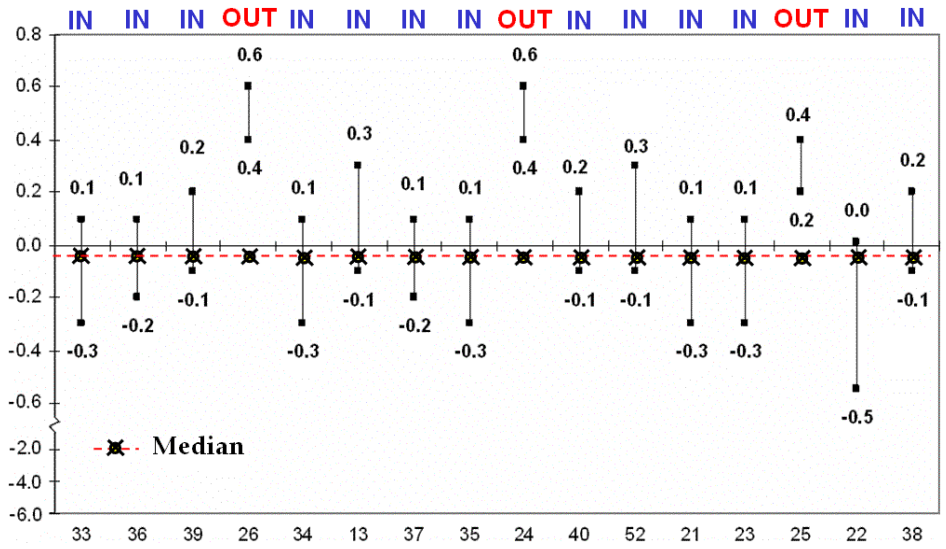

Fig. 36 . BDF reliability and scope error analysis.

To check whether the BDF correctly trained to learn the required discrimination function, we run the analysis in Fig. 25. The results of such an analysis are shown in Fig. 36. In particular, the vertical segments represent 95% Bayesian prediction intervals and the crossed points are the expected RE (median). In three cases (IDs 26, 24, and 25), the BDF could not learn correctly the required discrimination function ("OUT"). In the remainder cases, the BDF performed correctly ("IN"). In real cases, when using the BDF for prediction, we can use analysis in Fig. 36 to evaluate the reliability of the BDF before using it. As explained in Section 6.5, using the BDF for predicting projects 26, 24, and 25 may be very risky, implying that other approaches should be used. Once we obtain the actual values for each project being estimated, we find out whether a scope error actually occurred.

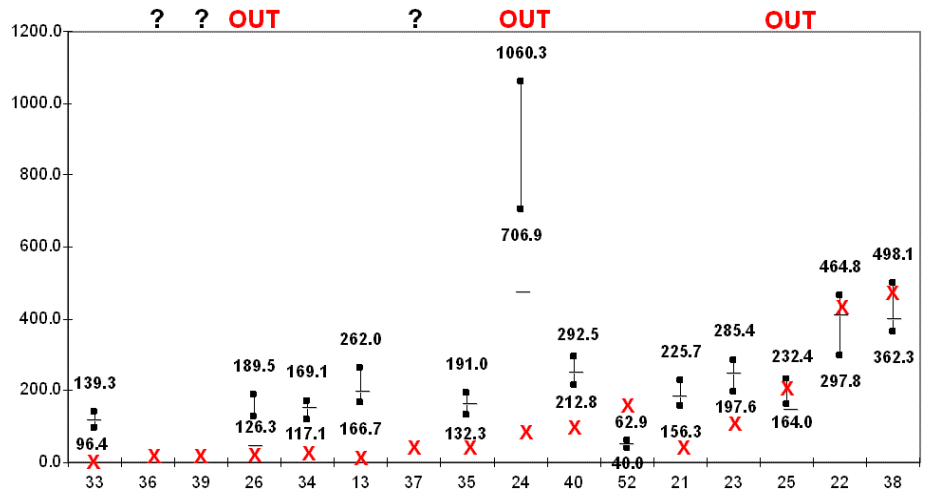

Fig. 37.Calculating estimate prediction intervals.

Fig. 37 shows the estimate prediction intervals that we calculated by applying Exn. (9). The y-axis represents the effort and the vertical segments represent the (Bayesian) estimate prediction intervals for the projects being estimated. "X" is the actual effort and "–" is the estimated effort. "?" refers to the fact that the EM provided negative effort estimates (invalid).

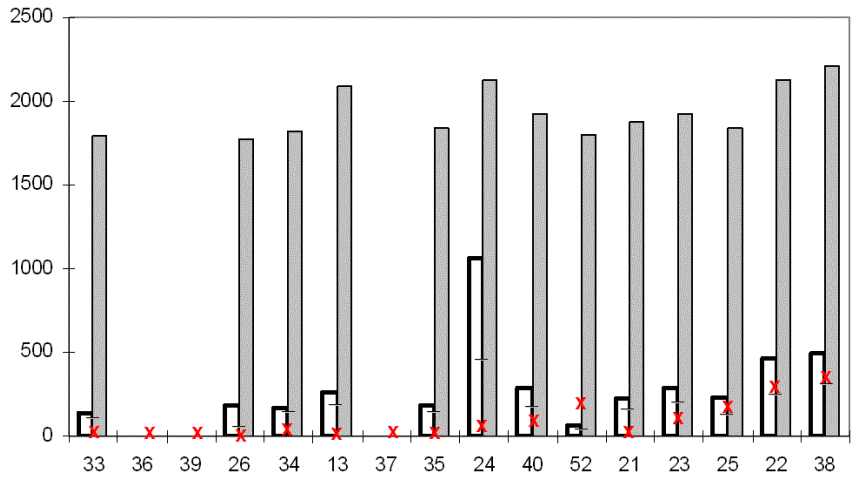

Fig. 38 . Comparison of prediction intervals calculated by the traditional methodology (Fig. 30) and the proposed one (Fig. 37) in the worst case.

Note that, the intervals in Fig. 37 are much narrower than the ones calculated with the traditional approach (Fig. 30). As hinted above, if we considered only the upper limit of each estimate prediction interval as the worst case of our prediction, the suggested approach may definitively shrink the prediction intervals making them more useful for prediction (Fig. 38). In Fig. 38, the solid rectangles represent the upper limit of the prediction intervals in Fig. 30 (traditional), while the other rectangles represent the upper limit of the prediction intervals in Fig. 37 (proposed), "X" stands for actual effort and "–" stands for estimated effort. Apart from the projects on which the EM provides negative effort values (i.e., 36, 39, and 37), where the comparison is not possible, the proposed approach provides better prediction intervals in terms of magnitude than the traditional one for all of the remaining cases except on project 52 where the interval does not include the actual RE. A valuable result is that the proposed approach is a valid alternative to the traditional methodology of shrinking the estimate prediction intervals for the worst case (where the worst case corresponds to the upper limit of the estimate prediction interval). Note that, it is trivial to see that, the proposed approach provides better intervals than traditional methodologies for the best case as well (the lower limit of the estimate prediction interval).

7.2.6 Model improvement by the BDF

Once we know the actual effort values of the projects in the test set, we can calculate the actual RE and perform the model improvement explained in Fig. 28 and Fig. 29.

Fig. 39 shows the model improvement analysis. The actual RE of each individual project is represented by the diamonds and the expected RE is represented by the crossed circles. As already discussed in Section 6.6, once we know the actual values of the projects, we can check whether the potential scope errors (situation 2 in Fig. 25) turns into actual scope errors.

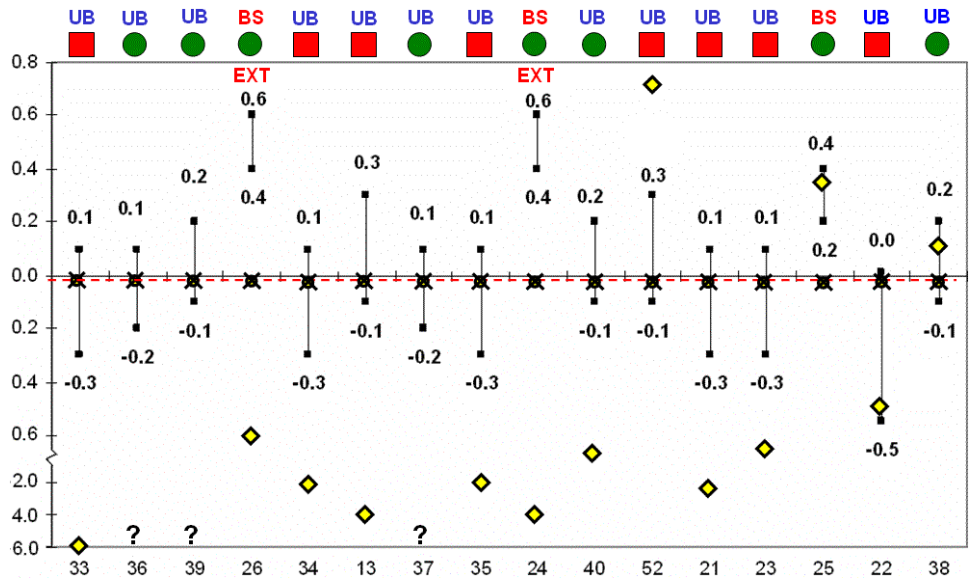

Fig. 39. Model improvement by the BDF in terms of scope error and model error.

In particular, to check it out, we run the analysis in Fig. 28 situations 2.a and 2.b. Since the actual RE falls outside the interval for projects 26 and 24, we conclude that those two cases represented a scope error. That is, the data set had an insufficient number of observations similar to projects 26 and 24 to build the BDF. This means that, we can extend the scope of our estimation model (i.e. the linear regression function in Table 5) by rebuilding it with the projects labeled "EXT" in Fig. 39 (26 and 24). Since the actual RE falls within the interval for project 25, we conclude that no scope error occurred and the problem was with the median used to split up the RE sample.

With respect to the outlier analysis (situation 1.b in Fig. 28), we can see that (Fig. 39), only the RE on projects 25, 22, and 38 fall within the interval while the RE falls out of the interval for the remaining projects, where we consider the BDF as reliable on project 25 for the reasons explained above. This means that if we used the remaining projects to build a new version of

the EM, the uncertainty of the EM would increase. So, to avoid degrading the EM we should not include those data points in the rebuilding of the model. An alternate choice would be to include those projects anyway because of the scarcity of observations and the potential of improving estimates if there are more data points of that kind.

Fig. 39 allows us to perform a model error analysis. Since we fixed a relative error magnitude of 0.3 as acceptable, we need to find categorical variables to shrink the magnitude of the error prediction intervals on projects 33, 34, 13, 35, 52, 21, 23, and 22 (unbiased solid squares in Fig. 39). The magnitude of the RE was acceptable on projects 36, 39, 37, 40, 38 (unbiased circle in Fig. 39).

The EM is biased on projects 26, 24, and 25. In that case, more suitable variables should be found and a more flexible function should be tried (e.g. log-linear models, non-linear models).

Overall, the EM can be extended in its scope by including projects 26 and 24 in the training to build a new version of the EM. The EM uncertainty can be improved by (1) including in the model the right categorical variables, (2) using a more flexible function, and (3) taking out outliers. Projects 25, 22, and 38 can be included in the training, but they will neither improve nor worsen the model.

7.2.7 Including (irrelevant) categorical variables

As shown in
Table 4, the COCOMO NASA data set provides some categorical variables that can be included in the estimation model (EM) to check whether the EM improves accuracy and lessens the risk. Including categorical variables in the EM involves the use of dummy variables. If we use dummy variables that are irrelevant, however, we can make the EM worse because dummy variables split up observations into distinct sub sets. This data splitting increases the

number of similarity groups. If the categorical (dummy) variables are irrelevant, a higher number of projects may be affected by scope error. This is because, for each subset of similar projects, there would be a less data points available to build the BDF. By contrast, if we have relevant categorical variables and a sufficient number of observations, we would shrink the (Bayesian) error prediction intervals.

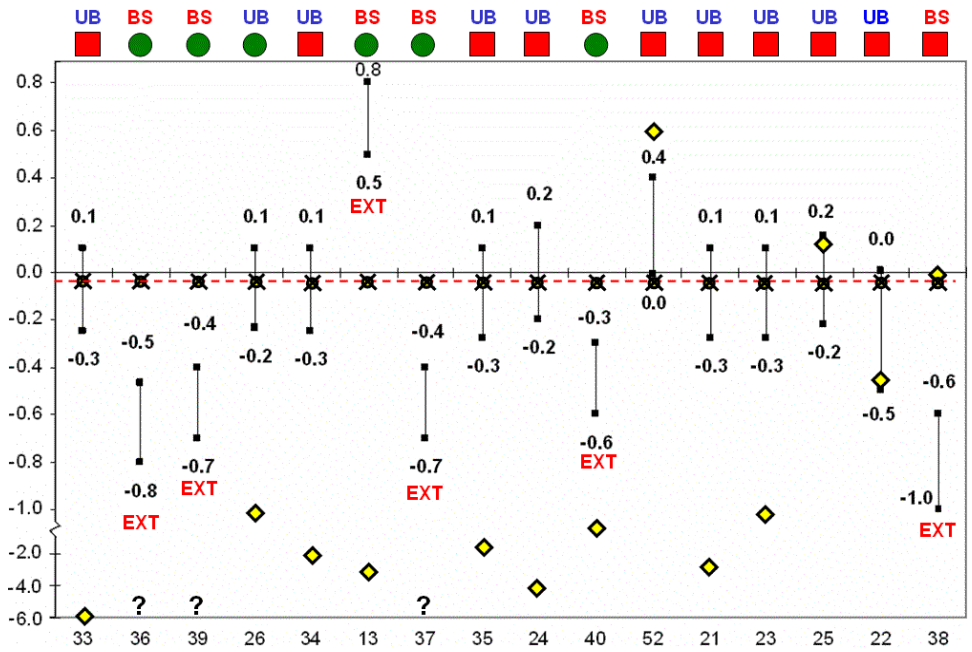

Fig. 40. The effect of including irrelevant dummy variables.

Fig. 40 shows the effect of including the categorical variable “Mode” in the EM. Since “Mode” has three categories (“Semidetached”, “Embedded”, “Organic”), we included two dummy variables (D1, D2) in the EM and coded the categories as follows, “Semidetached” became {D1 = 0, D2 = 1}, “Embedded” became {D1 = 1, D2 = 0}, “Organic” became {D1 = 0, D2 = 0}.

As Fig. 40 shows, the number of projects where a scope error would occur increased from two (26 and 24, in Fig. 39) to six (36, 39, 13, 37, 40, and 38, in Fig. 40). Notice that, the accuracy in terms of RE was similar to the previous one. The comparison can be made by calculating the mean and the standard deviation of the relative error on the same test set for both models [BOEHM81], [CONTE86], [MYRTVEIT05]. The linear model has Mean(RE_{TsS}) = -0.525 and STD(RE_{TsS}) = 3.116. The linear model with dummy variables has Mean(RE_{TsS}) = -0.740 and STD(RE_{TsS}) = 2.751. Therefore, the former is less biased than the latter, i.e. its Mean(RE_{TsS}) is closer to zero than the latter. However, the former has a higher spread than the latter, i.e. the STD(RE_{TsS}) of the former is greater than the STD(RE_{TsS}) of the latter. Since many organizations are still using $MMRE_{TsS}$, we also show this statistic (see Section 3.2 for its definition and the discussion about its inappropriateness for comparison [KITCHENHAM01]). The linear model has $MMRE_{TsS}$ = 2.396 and the linear model with dummy variables has $MMRE_{TsS}$ = 2.214. Based on the discordant statistics, we concluded that the former and the latter have similar accuracy with respect to the considered data set.

It is worth noting that, the error prediction interval on project 25 became unbiased while the interval on project 38 became biased. This is because of the splitting effect discussed above, which increased the occurrence of scope errors. As a practical consideration, even though using categorical variables may increase the EM accuracy in terms of RE, the risk of getting a scope error may increase. This is why using categorical variables should be done with care.

We also considered the other categorical variables in
Table 4. However, the results in terms of magnitude of error prediction interval were similar to the one shown in Fig. 40 and in some cases were even worse (not shown).

7.2.8 Improving the model shape (logarithmic transformation)

We tried to improve the model by considering a logarithmic transformation. The logarithmic transformation provides a non-linear model that is linear in the-parameters for which the least squares estimates can be calculated without using iterative procedures. Unlike the linear model, the logarithmic model provides much more accurate and less risky estimates than the linear model.

To build the log-linear model, we considered only numeric variables in Table 4, i.e. the dependent variable (DV) was the "Effort" and the independent variables (IVs) were the "Size" and the "15 COCOMO-I multipliers". Then, we calculated the logarithm of each value and applied the least squares procedure for estimating the model parameters. To build the BDF, we followed a procedure with the same settings as the one shown above. Then, we used the BDF to calculate the error prediction intervals (Fig. 41).

The EM provided valid estimates (non-negative) for all of the projects considered. In Fig. 41, the vertical intervals represent the 95% (Bayesian) error prediction intervals for the considered projects. The diamonds are the actual RE values and the crossed circles are the expected RE values (the median). As we can see in Fig. 41, all of the intervals were acceptable and valid, i.e. they included the expected RE (the median), except for project 38 in which the interval does not include the expected error ("EXT"). Based on the decision algorithm in Fig. 28 (2.b), we concluded that the EM would cause a scope error on project 38. Conversely, the EM would not provide any scope error on the remaining projects.

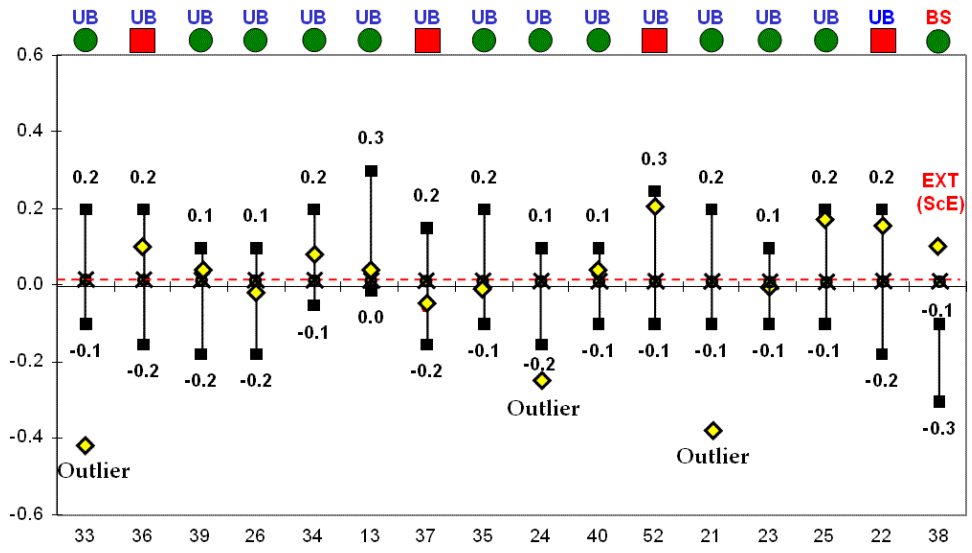

Fig. 41. Model improvement by applying a log-linear transformation.

The outlier analysis shows that there are only three data points that fall outside the interval (33, 24, and 21). It is worth noting that, the EM is much less risky than the linear one. In fact, only four intervals exceed the acceptable RE limit of 0.3 (36, 37, 52, and 22), while the linear model had 8 intervals exceeding the limit (rectangles in Fig. 39) and the linear model with dummy variables had 10 unacceptable intervals (rectangles in Fig. 40). Except for project 38, all of the intervals were unbiased.

To calculate the estimate prediction interval related to the RE intervals in Fig. 41, we applied Exn. (9). The results are shown in Fig. 42, where the vertical intervals are the (Bayesian) estimate prediction intervals for the project considered, "X" stands for the actual effort, and "–" stands for the estimated effort by the log-linear EM.

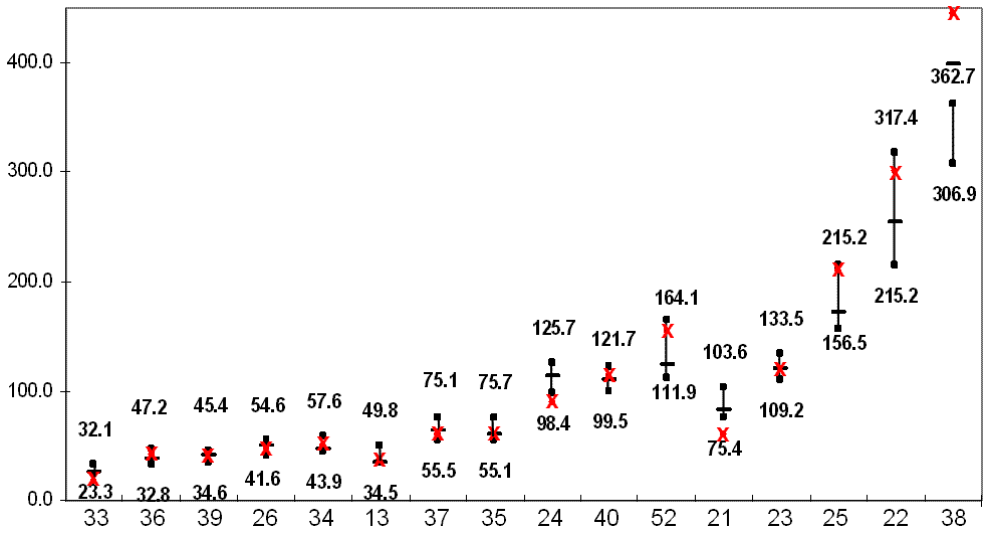

Fig. 42. Estimate (effort) prediction intervals related to Fig. 41.

As Fig. 42 shows, the estimation model was definitely improved with respect to the linear one (Fig. 37).

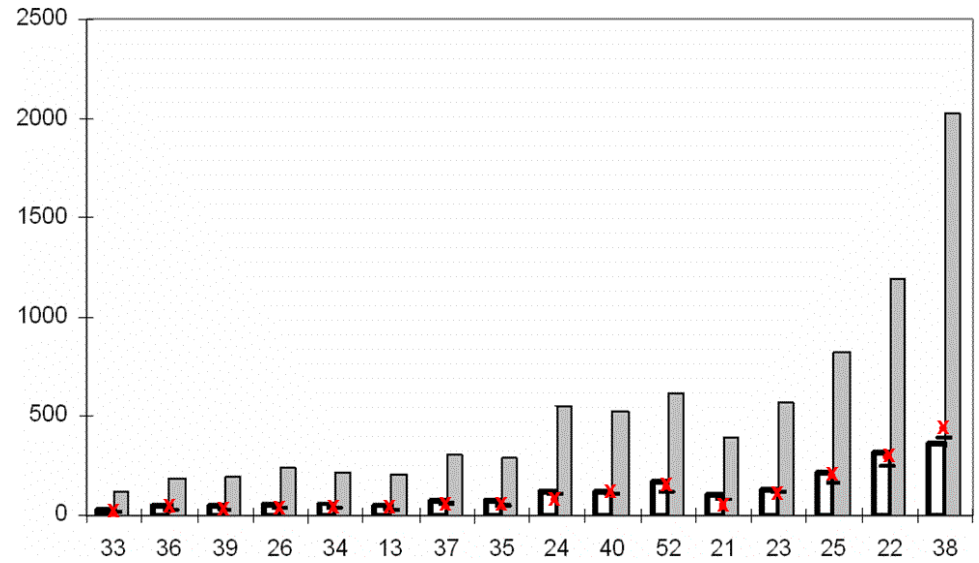

Fig. 43. Comparison of prediction intervals calculated by the traditional methodologies and the proposed one (worst case).

In fact, all of the estimates fell within the expected range except for projects 33, 24, and 21 (outliers) and project 38 (unacceptable scope error).

In Fig. 43, we again look at a worst case scenario (examining the upper prediction interval endpoints to estimate the risk). We compared the (effort) prediction intervals calculated by the traditional methodologies (solid rectangles in Fig. 43) and the ones provided by the proposed approach (the unfilled rectangles in Fig. 43). As we can see in Fig. 43, the proposed approach was definitely able to shrink the intervals. The only actual effort value that fell outside the interval was on project 38.

As an application of the proposed approach, before undertaking a project, a software organization may use the upper endpoints of the rectangles in Fig. 43 to figure out whether the risk of a failure (e.g. exceeding the schedule, budget, and quality) would be acceptable. In particular, the approach is able to provide a useful risk evaluation in quantitative terms that traditional methodologies cannot yield because the intervals produced are too large and the regression assumptions are violated.

7.2.9 Including (relevant) categorical variables

Based on Fig. 41, we can continue to improve the model in terms of accuracy and risk by applying the approach again, e.g. including categorical variables in the model.

We tried to consider the same categorical variable as the one taken for the linear model, i.e. “Mode” (
Table 4). To this end, we applied the same dichotomous (dummy) coding as above. The result is shown in Fig. 44.

Unlike Fig. 40, the categorical variable “Mode” is relevant for the log-linear model. In fact, all of the error prediction intervals have been shrunk and made unbiased. In particular, the intervals on projects 36, 37, 52, and 22 (squares in Fig. 41) turned into acceptable intervals (circles in Fig. 44). However, the relative error on project 37 turned in an outlier. Instead, the

interval on project 38 turned into a valid one, even though the relative error became an outlier.

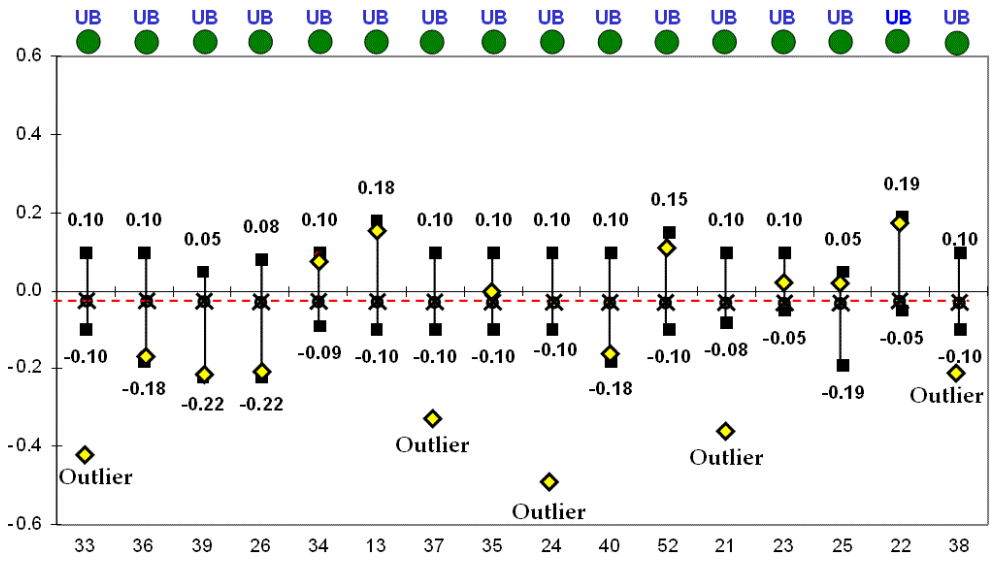

Fig. 44. Error prediction intervals for the log-linear model with the categorical variable "Mode".

From an accuracy point of view, the variable "Mode" did not improve the model. In fact, the model without categories (Fig. 41) has Mean(RE_{TsS}) = -0.005, STD(RE_{TsS}) = 0.045, and $MMRE_{TsS}$ = 0.030. The model with categories (Fig. 44) has Mean(RE_{TsS}) = -0.026, STD(RE_{TsS}) = 0.049, and $MMRE_{TsS}$ = 0.044. From an uncertainty (risk) point of view, however, the latter is less risky than the former because the magnitude of the intervals in Fig. 44 is less than the magnitude of the ones in Fig. 41. Another interesting effect is that, the categorical variable "Mode" was unable to improve the linear model in terms of uncertainty (Fig. 40) but, it was able to improve the log-linear one (Fig. 44). From a computational point of view, if we select some dummy variables as relevant for a model (e.g. linear) and then turn the model into another model (e.g. log-linear or non-linear) having the same

variables as the former, the relevance of the dummy variables has to be rechecked.

7.2.10 Using uncertainty for model comparison

The uncertainty analysis presented in this work has an important implication for software organizations and practitioners in terms of model comparison. Analyses in Fig. 41 and Fig. 44 show that, one estimation model can be more accurate than another, even though the former may be more risky than the latter. Therefore, unlike traditional approaches, model selection should be based upon both accuracy and uncertainty.

Before claiming that one model is "better" than another, however, an organization should specify the nature of the comparison and the context where the comparison is made. For instance, one should say that a model is better than another in terms of accuracy (or in terms of uncertainty/risk) and select the one that is more appropriate with respect to organization's goals. In the example shown above, an organization aiming at accuracy should select the log-linear model without categories (Fig. 41). An organization aiming at shrinking the uncertainty should select the the log-linear model with categories (Fig. 44). It is worth noting that, some authors [MCCONNELL06], [JØRGENSEN03], however, argue that providing one-point estimates (i.e. aiming at accuracy) is ineffective and even misleading. Organizations should aim at uncertainty, i.e., estimates should always be provided in terms of prediction intervals (two-point estimates) because two-point estimates are more realistic and useful for software organizations.

7.2.11 Proving that "outliers" behave as outliers

With respect to Fig. 44, we rebuilt the log-linear estimation model with the same categories as above by including only outliers (33, 37, 24, 21, and 38). The result is shown in Fig. 45.

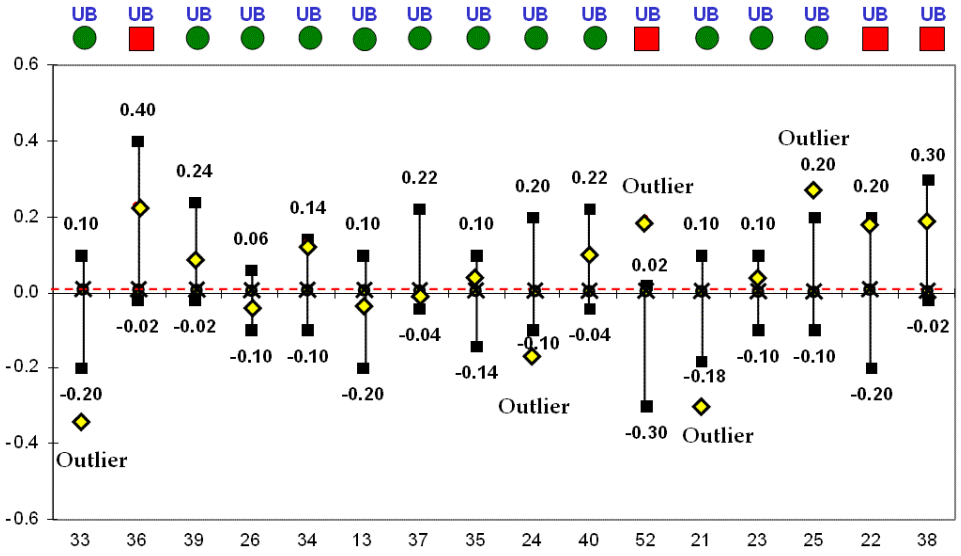

Fig. 45. Error prediction intervals for the log-linear model with the categorical variable "Mode" trained with five more outliers in Fig. 44.

From an accuracy point of view, the model in Fig. 44 is less accurate than the one in Fig. 45 because the former has Mean(RE_{TsS}) = -0.026, STD(RE_{TsS}) = 0.049, and $MMRE_{TsS}$ = 0.044, while the latter has Mean(RE_{TsS}) = -0.007, STD(RE_{TsS}) = 0.045, and $MMRE_{TsS}$ = 0.035. This result was predictable because five projects in Fig. 45 (i.e. the outliers of Fig. 44) were included in the estimation model training. Therefore, only 11 projects out of 16 were independent from the remaining projects. Table 9 shows the relative errors of both models (i.e. models in Fig. 44 and Fig. 45) on those projects.

From an uncertainty point of view, however, the effect of including outliers in the training set of the model in Fig. 45 made the uncertainty worse as we expected (compare the magnitude of the intervals of both figures). In particular, the intervals on projects 36, 52, 22, and 38 increased insofar as their magnitude exceeded the acceptability threshold (RE = 0.3).

TABLE 9

RELATIVE ERROR COMPARISON

EM in Fig. 44	-0.42	-0.32	-0.49	-0.36	-0.21
EM in Fig. 45	-0.34	0.00	-0.18	-0.31	0.19
Project	33	37	24	21	38

Note that, both models have 5 outliers, three in common (33, 24, and 21) and two different (52 and 25 in Fig. 44, and 37 and 38 in Fig. 45). The log-linear model with dummy variables is unable to explain these points because of the model error, which can be removed if and only if further explanatory variables are included in the model and /or a more suitable (flexible) model is considered. So, we tried to use a non-linear model (MLFFNN), but we did not get any significant improvement (not shown). We concluded that the improvement depended on the lack of some relevant variables, rather than the model shape/complexity. Then, we stopped trying to improve the model because we used up all the variables available in the COCOMO NASA data set.

7.2.12 Proving that "non-outliers" behave as non-outliers

With respect to Fig. 44, we also rebuilt the log-linear estimation model with categories by not including the outliers and using the remaining projects, i.e. 36, 39, 26, 34, 13, 35, 40, 52, 23, 25, and 22. The result is shown in Fig. 46.

Fig. 46 shows the expected result, i.e. the magnitude of the error prediction intervals did not increase with respect to Fig. 44. Nevertheless, from an accuracy point of view, the model in Fig. 46 did not improve. In fact, it has $\text{Mean}(RE_{TsS}) = -0.024$, $\text{STD}(RE_{TsS}) = 0.198$, and $MMRE_{TsS} = 0.146$. Note that, the model in Fig. 46 has three outliers as the model in Fig. 44 (33, 24, and 21). It has three more outliers (36, 52, and 25), and two fewer outliers (37 and 38) than the model in Fig. 44. The outliers in Fig. 46 are the same as

the ones in Fig. 45, apart from project 36, where the interval did not include the actual RE. This means that, actually the log-linear model with dummy variables is not able to explain these points confirming the conclusions made above.

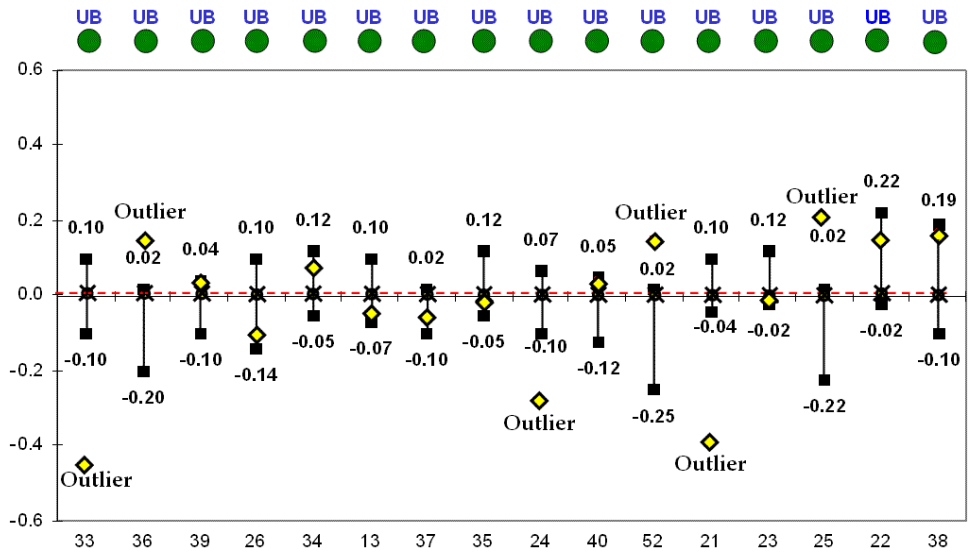

Fig. 46. Error prediction intervals for the log-linear model with the categorical variable "Mode" trained with eleven non-outliers in Fig. 44.

7.2.13 Discussion

Analyses in Fig. 44, Fig. 45, and Fig. 46 show that the uncertainty (risk) calculated by the (Bayesian) prediction interval on the relative error is more stable and reliable for evaluating and selecting estimation models than using some summary statistics on the relative error expressing the estimation model accuracy.

This case study shows that, to improve the accuracy of the linear estimation model, we have to change the model shape (i.e. applying a logarithmic transformation). To improve the model from an uncertainty point of view, we have to include the categorical variable "Mode" in the log-linear

model. Moreover, the case study shows that, further rebuilding the EM with either outliers or non-outliers does not improve the accuracy. Nevertheless, rebuilding the model with non-outliers does not increase the uncertainty. Therefore, it is acceptable to include non-outliers as expected.

Based on the uncertainty analysis, before predicting a new project, we should consider different models as stated above and select the one that provides the least risk. Once the most accurate and least risky model has been selected among those considered and we can use the model for predicting new projects and continue improving the model repeatedly.

Chapter 8

Conclusion

8 Conclusion

The problem addressed in this work was that the traditional methodologies for estimating prediction intervals (i.e. the most probable interval where the next estimate will fall) of parametric models provide intervals that are too wide to be useful in terms of prediction accuracy and model improvement. We showed that this happens because the traditional methodologies do not take into account the fact that assumptions on which the estimation models are based may not hold. We showed that the estimation risk analysis performed by traditional methodologies might be incorrect because they do not deal with the consequences of violations.

We provide an alternative point of view to the traditional way of improving the accuracy of estimation models. We not only try to improve the model as traditional methodologies do, but also correct the estimates by analyzing the estimation error trend of the model over its history of application. Moreover, we define a strategy for improving the model over time by assessing the actual errors versus the expected error ranges. We use the uncertainty as a discrimination parameter, not the accuracy as usually done. To analyze the estimation error trend of the model, we define an additional model based on a specific multi-layer neural network for discrimination, which helps us calculate the estimation error of the model, without making specific assumptions. Once we know the estimation error trend of the model, we can correct the estimates, improving the accuracy of the model.

This new point of view is based on the fact that, any model is a limited representation of the reality. Therefore, errors cannot be removed completely. That is why we use an additional model (e.g. the neural network for

discrimination) for overcoming the limitations of the estimation model. These limitations concern the fact that, estimation models violate assumptions on which they are built without dealing with the consequences of those violations. Moreover, unlike traditional techniques, the defined neural network is able to cope with scope and assumption errors, as well. To highlight the impact of violations against the estimation model, we use the neural network because it is based on the Bayesian paradigm, which can consider all of the available information (prior and posterior information). In fact, traditional estimation improvement methodologies do not provide good results just because they do not exploit all of the information that can be gathered from the model application history (i.e., past observations).

Another important point discussed in this work is that, traditional estimation approaches use accuracy to select the best model. Then, based on the organization's history, they find an estimation model that is able to improve the accuracy calculated by some summary statistics over the estimation error. This is not a suitable way of improving the accuracy because it depends on the statistics used for evaluating the accuracy. The approach proposed in this work is based upon evaluating and selecting candidate estimation models by uncertainty, which is invariant with respect to the accuracy statistics. In other words, the approach selects the least risky model by uncertainty. This means that, instead of finding the most accurate model, we find the least risky model (i.e. having the least uncertainty). However, to apply such an approach, we need a methodology that is able to provide a narrower uncertainty range and is based on avoiding any assumption over the model. That is why we defined the proposed approach extending the traditional methodologies.

8.1 Benefits and drawbacks

Benefits of applying the proposed approach are that the approach

- supports learning organizations because it focuses on evolving the estimation model over time,
- is based on an Estimation Improvement Process (EIP) that makes the approach formal, traceable, repeatable, and improvable over time,
- introduces the concept of packaging the experience of using the estimation model into a feed-forward multi-layer neural network, therefore it facilitates packaging, storing, delivering and exploiting the experience of an organization over time,
- can deal with the scope error, assumption error, and model error at the same time,
- defines an "a priori" strategy to mitigate the estimation risk by considering all kinds of errors,
- shrinks the magnitude of the prediction intervals to make them useful for prediction and model improvement,
- allows comparing models in terms of uncertainty and accuracy at the same time,
- is not based on any specific assumptions,
- can be applied to estimating any software engineering variable such as effort, size, defects, number of test cases, fault proneness over any stage of the project (e.g. inception, construction); the approach does not care about the development process hence it can be used for iterative, agile, and traditional development processes,
- improves the competitive advantage of learning organizations by improving the estimation model over time; the approach does not focus on finding a unreachable best estimation model, but it deals with improving the parametric model that each organization uses and trusts,
- is able to perform a similarity analysis in terms of risk between the historical projects and the one being estimated, i.e. the approach can

answer questions like the following, what is the estimation uncertainty on this project considering its similarities with my past projects?

- can be implemented as a stand-alone estimation methodology or used for supporting experts and organizations together with other estimation techniques. Software applications based on such an approach offload the complexity of dealing with neural networks to experts and allow project managers to rely on effective and powerful automatic tools for analyzing estimation risk and improving the estimation model
- helps the highest organizational management make proactive strategic decisions by checking automatically over time whether the software development environment is changing with respect to its history,
- can be implemented as a support software tool and used by organizations that apply human-based estimation methodologies,
- is modular insofar as it can be eventually improved by new findings. Modularity allows us to apply the approach partially, e.g. we may only run the similarity analysis or check whether our estimation process is improving over time,

The proposed approach has a few drawbacks. The approach requires

- being able to apply neural networks and their optimization techniques,
- a support software tool implementing the overall procedure,
- the expense of rerunning the training procedure for enhancing the estimation model capabilities as the history grows,
- spending non-negligible time for training the network.

What we proposed in this paper is of course a first step towards using computational intelligence techniques for dealing with statistical problems that traditional methodologies have not solved. We hope that researches and practitioners will contribute to enhance and consolidate research in this rising field of software engineering that promises fascinating solutions.

8.2 Future work

In this work, we have argued the theoretical benefits of the proposed approach and demonstrated its use on existing model and data (i.e., regression functions calibrated on the COCOMO-NASA data set). However, we did not empirically evaluate our results against the results that traditional improvement methodologies can yield. To this end, we are planning a comparative analysis between the proposed approach and the traditional ones.

Further future work is aimed at experiments to increase confidence in the use of uncertainty as a selection criterion for estimation models. We believe that uncertainty can be a keystone for eventually evaluating models in a consistent way.

Ideally, we would like to run a pilot study, where we apply the approach to a real software development environment allowing an organization to evaluate strategically their estimation capability (Fig. 21). The organization could use the methodology to make proactive decisions about the organization's future, e.g. evaluating whether variables are changing in the organization, knowing whether new variables are required for figuring out the environment, assessing changes, evaluating development tools, and evaluating people in terms of capability and productivity.

Appendix

Acronyms

Acronym	Description	Acronym	Description
ACT	Actual Effort	**KSLOC**	Thousand Source Lines of Code
AE	Absolute Error	**LOOCV**	Leave One Out Cross Validation
AEP	Automated Experience Package	**LS**	Least Squares
ANN	Artificial Neural Networks	**MCMC**	Markov Chain Monte Carlo
ANOVA	Analysis of Variance	**MdMRE**	Median Magnitude of Relative Error
ARCH	Autoregressive Conditional Heteroscedasticity	**Me**	Median
BDF	Bayesian Discrimination Function	**ML**	Machine Learning
BRE	Balanced Relative Error	**MLE**	Maximum Likelihood Estimation
BS	Biased EPI	**MLFFNN**	Multi-Layer Feed-Forward Neural Network
CCA	Curvilinear Component Analysis	**MLFFNND**	Multi-Layer Feed-Forward Neural Network for Discrimination
COCOMO	Constructive Cost Model	**MMRE**	Mean Magnitude of Relative Error
Cplx	Complexity	**MRE**	Magnitude of Relative Error
DS	Data Set	**MSE**	Mean Squared Error
DV	Dependent Variable	**NID**	Normal and Identically Distributed
Dxx	xx Sample	**NULL**	No EPI available
EB	Experience Base	**OUT**	Potential Scope Error
EF	Experience Factory	**PCA**	Principal Component Analysis
EIP	Estimation Improvement Process	**PI**	Prediction Interval
EM	Estimation Model	**PM**	Person Month
EPI	Error Prediction Interval	**Pr**	Probability
Eqn.	Equation	**PRED(x)**	Average fraction of the MRE's off by no more than x
EReg	Error Regression	**QIP**	Quality Improvement Paradigm
EST	Estimated Effort	***RdndME***	Redundancy Model Error
Exn.	Expression	**RE**	Relative Error
EXT	Scope Extension	**Re**	Risk Exposure
Fig.	Figure	**RUP**	IBM-Rational Unified Process
FlexME	Flexibility Model Error	**ScE**	Scope Error
GQM	Goal Question Metrics	**SQRT()**	Square Root
ID	Software System Identifier	**STD**	Standard Deviation
Im	Impact	**Tab.**	Table
IN	Acceptable Scope Error	**TAME**	Tailoring a Measurement Environment
IQR	Interquartile Range	**TSRP**	Two-Stage Regression Procedure
IVs	Independent Variables	**UB**	Unbiased EPI
KFCV	K-Fold Cross Validation	***VltnME***	Violation Model Error
KLOC	Thousand Lines of Code	***VrblME***	Variable Model Error

Bibliography

[AHA96] D.W. Aha, and R.L. Bankert, "A comparative evaluation of sequential feature selection algorithms", Artificial Intelligence and Statistics, Springer-Verlag, NY, 1996.

[AITKEN95] C.G.G. Aitken, "Statistics and the Evaluation of Evidence for Forensic Scientists," New York, J. Wiley & Son. 1995.

[ANGELIS00] L. Angelis, I. Stamelos, "A Simulation Tool for efficient analogy based cost estimation," Empirical Software Engineering, Vol. 5, no. 1, pp. 35-68, March 2000.

[BAILEY81] J W. Bailey, V.R. Basili, "A meta-model for software development resource expenditures," in Proceeding of the 5th International Conference on Software Engineering, pp. 107-116, 1981.

[BAIN92] L.J. Bain, M. EngelHardt, "Introduction to Probability and Mathematical Statistics," PWS-Kent, Boston, 1992.

[BARRON93] A. Barron, "Universal approximation bounds for superposition of a sigmoidal function", IEEE Transaction on Information Theory, 39, pp. 930-945, 1993.

[BASILI92] V. R. Basili, G. Caldiera, and G. Cantone, "A Reference Architecture for the Component Factory," ACM Transactions on Software Engineering and Methodology (TOSEM), Vol. 1(1): 53-80, January 1992.

[BASILI94a] V. R. Basili, G. Caldiera, H. D. Rombach, "Goal Question Metric Paradigm," In Encyclopedia of Software Engineering, Ed. J.J. Marciniak, John Wiley & Sons, 1994.

[BASILI92B] V. R. Basili, G. Caldiera, H. D. Rombach, "The Experience Factory," In Encyclopedia of Software Engineering, Ed. J.J. Marciniak, John Wiley & Sons, 1994.

[BASILI95] V.R. Basili, "Software Quality Assurance and Measurement: A Worldwide perspective," Applying the Goal/Question/Metric Paradigm in the Experience factory, Chapter 2, pp. 21-44, International Thomson Computer Press, 1995.

[BASILI88] V.R. Basili and H.D. Rombach, "The TAME Project: Towards Improvement-Oriented Software Environments," IEEE Transactions on Software Eng., vol. 14, #6, June 1988.

[BASILI84] V.R. Basili and D. Weiss, "A Methodology for Collecting Valid Software Engineering Data," IEEE Transactions on Software Engineering, pp. 728-738, November 1984.

[BINGUS96] J. P. Bingus, "Data Mining With Neural Networks: Solving Business

Problems From Application Development to Decision Support," McGraw-Hill, NY, 1996.

[BISHOP95A] Bishop C., "Neural Network for Pattern Recognition," Oxford University Press, 1995.

[BISHOP95B] C. Bishop and C.S. Qazaz, "Bayesian inference of noise levels in regression," ICANN95, EC2 & Cie, Paris, pp. 59-64, 1995.

[BOEHM81] B.W. Boehm, "Software Engineering Economics," Prentice Hall, 1981.

[BRIAND92] L.C. Briand, V.R. Basili , and W. Thomas, "A pattern recognition approach to software engineering data analysis", IEEE Transactions on Software Engineering, 18 (11), pp. 931-942, 1992.

[BRIAND99] L.C. Briand, K. El-Emam, K. Maxwell, D. Surmann, and I. Wieczorek, "An Assessment and Comparison of Common Cost Software Project Estimation Methods," Proc. 21st Int'l Conf. Software Eng. (ICSE 21), pp. 313-322, 1999.

[BRIAND99] L.C. Briand, T. Langley, and I. Wieczorek, "A Replicated Assessment and Comparison of Common Software Cost Modeling Techniques," Proc. Int'l Conf. Software Eng. (ICSE 22), pp. 377-386, 2000.

[BROCK03] S. Brock, D. Hendricks, S. Linnell, D. Smith, "A Balanced Approach to IT Project Management", ACM, SAICSIT'03, Sep. 2003.

[BUSEMEYER00] J.R. Busemeyer, and Y.M. Wang, "Model comparisons and model selections based on generalization criterion methodology," Journal of Mathematical Psychology, 44(1), 171-189, 2000.

[CANTONE00] G. Cantone, P. Donzelli, "Production and Maintenance of Goal-oriented Measurement Models," World Scientific, Vol. 10, no. 5, pp. 605-626, 2000.

[CHEN05] Z. Chen, T. Mezies, D. Port, B.W Boehm, "Feature Subset Selection Can Improve Software Cost Estimation Accuracy", PROMISE '05, ACM, Missouri USA, pp. 1-6, 2005.

[CHULANI99] S. Chulani, B.W. Boehm and B. Steece, "Bayesian analysis of empirical software engineering cost models," IEEE Transactions on Software Engineering 25(4): 573-583, 1999.

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

[COCOMO2] The COCOMO II Suite, http://sunset.usc. edu/research/cocomosuite/index.html, 2004.

[CONTE86] S.D. Conte, H.E. Dunsmore, and V.Y. Shen, "Software Engineering Metrics and Models," Benjamin-Cummings, Menlo Park CA, 1986.

[DREYFUS05] G. Dreyfus, "Neural Networks Methodology and Applications," Springer, 2005.

[EFRON93] B. Efron and R.J. Tibshirani, "An Introduction to the Bootstrap," Chapman & Hall, NY, 1993.

[ENGLE82] R.F. Engle, "Autoregressive Conditional Heteroscedasticity with estimates of Variance of United Kindom Inflaction," Econometrica, Vol. 50, pp. 987-1007, 1982.

[FEIGENBAUM56] A.V. Feigenbaum, "Total Quality Control," pp. 93-101, November-December, 1956.

[FENTON93] N. E. Fenton, "Software Metrics - A Rigorous Approach", Chapman & Hall, 1993.

[FENTON99] N.E. Fenton and M. Neil, "Software Metrics: Successes, Failures and New Directions," J. Systems and Software, vol. 47, nos. 2-3, pp. 149-157, 1999.

[FENTON00] N.E. Fenton and M. Neil, "Software Metrics: Roadmap," The Future of Software Eng., A. Finkelstein, ed., pp. 357-370, 2000.

[FINNIE97] G. Finnie, G. Wittig, and J.-M. Desharnais, "A comparison of software effort estimation techniques using function points with neural networks, case based reasoning and regression models", J. of Systems & Software, 39: pp281-289, 1997.

[FOSS03] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit, "A Simulation Study of the Model Evaluation Criterion MMRE," IEEE Trans. Software Eng., vol. 29, no. 11, pp. 985-995, Nov. 2003.

[GENERO05] M. Genero, M. Piattini, and C. Calero, "Metrics for Software Conceptual Models, Imperial College Press, 2005.

[GILKS96] SW R. Gilks, R. Richardson, and D.J. Spiegelhalter, "Markov Chain Monte Carlo in Practice," Chapman & Hall, London, 1996.

[GREENE97] W.H. Greene, Econometric Analysis, 3rd edn. (Prentice-Hall, Upper Saddle River, NJ, 1997).

[GULEZIAN91] R. Gulezian, "Reformulating and calibrating COCOMO", J. of Systems & Software, 16: pp235-242, 1991.

[GUYON05] Guyon, S. Gunn, M. Nikravesh, L. Zadeh, "Feature Extraction foundations and applications," Springer, 2005.

[HAGAN94] M.T. Hagan, M.B. Menhaj, "Training Feedforward Networks with the Marquardt Algorithm," IEEE Transactions on Neural Networks, vol. 5, no. 6, Nov. 1994.

[HIGUERA96] R.P. Higuera, "Software Risk Management." TR-012 CMU/SEI, pp. 1-48, 1996.

[HUBER81] P. J. Huber, "Robust Statistics," Wiley, 1981.

[HWANG97] J.T. G. Hwang and A.A. Ding, "Prediction Intervals for Artificial Neural Networks," Journal of American Statistical Association, Vol. 92, no. 438, pp. 748-757, 1997.

[HUSMEIER04] D. Husmeier, R. Dybowski, and S. Roberts, "Probabilistic Modeling in Bioinformatics and Medical Informatics," Springer, 2004.

[JEFFERY90] R. Jeffery, and G. Low, "Calibrating estimation tools for software development," Software Engineering Journal, 5(4), pp215-221, 1990

[JOHN94] G. John, R. Kohavi, K. Pfleger, "Irrelevant features and the subset selection problem," 11th Intl. Conference on Machine Learning, Morgan Kaufmann, pp. 121-129, 1994.

[JOLLIFE96] I.T. Jollife, "Principal Component Analysis," Springer, 1986.

[JØRGENSEN95] M. Jørgensen, "Experience With the Accuracy of Software Maintenance Task Effort Prediction Models," IEEE TSE, 21(8), pp. 674-681, August 1995.

[JØRGENSEN04A] M. Jørgensen, "Regression Models of Software Development Effort Estimation Accuracy and Bias," Empirical Software Engineering, Vol. 9, 297-314, 2004.

[JØRGENSEN07] M. Jørgensen, and M. Shepperd, "A Systematic Review of Software Development Cost Estimation Studies", IEEE Transactions on Software Engineering, 33(1): pp. 33-53, 2007.

[JØRGENSEN03] M. Jørgensen and D.I.K. Sjøberg, "An Effort Prediction Interval Approach Based on the Empirical Distribution of Previous Estimation Accuracy" Journal of Information Software and Technologies 45: 123-136, 2003.

[JØRGENSEN04B] M. Jørgensen, K. Teigen, and K. Moløkken, "Better sure than safe? Overconfidence in judgment based software development effort prediction intervals," J. of Systems & Software, 70, pp79-93, 2004.

[KÄNSÄLÄ97] K. Känsälä, "Integrating Risk Assessment with Cost Estimation," IEEE software, pp. 61-66, May-July 1997.

[KAPLAN92] R. S. Kaplan, D. P. Norton, "The Balanced Scorecard – Measures that drive Performance," Harvard Business Review 70, Nr. 1, 1992.

[KAPLAN96A] R. S. Kaplan, D. P. Norton, "Using the BSC as a Strategic Management," Harvard Business Review 74, Nr. 1, 1996.

[KAPLAN96B] R.S. Kaplan, D.P. Norton, "The Balanced Scorecard: Translating Strategy into Action," Harvard Business Press, Boston, 1996.

[KARUNANITHI92] N. Karunanithi, D. Whitley, and Y.K. Malaiya, "Using

Neural Networks in Reliability Prediction," IEEE Software, vol. 9, no. 4, pp. 53-59, 1992.

[KEMERER87] C.F. Kemerer, "An Empirical Validation of Software Cost Estimation Models," ACM Comm., vol. 30, no. 5, pp. 416-429, May 1987.

[KEUNG08] J.W. Keung, B. A. Kitchenham, and D. R. Jeffery, "Analogy-X: Providing Statistical Inference to Analogy-Based Software Cost Estimation," IEEE Trans. On Software Engineering, (34) 4, pp. 1-14, 2008.

[KHOSHGOFTAAR97] T. M. Khoshgoftaar, E. B. Allen, J. P. Hudepohl, and S. J. Aud, "Neural Networks for Software Quality Modeling of a Very Large Telecommunications System," IEEE Trans. on Neural Networks, (8)4, pp. 902-909, 1997.

[KHOSHGOFTAAR95] T. M. Khoshgoftaar and D. L. Lanning, "A Neural Network Approach for Early Detection of Program Modules Having High Risk in the Maintenance Phase," J. Systems Software, 29(1), pp. 85-91, 1995.

[KHOSHGOFTAAR94] T. M. Khoshgoftaar, D. L. Lanning, and A. S. Pandya, "A comparative-study of pattern-recognition techniques for quality evaluation of telecommunications software," IEEE Journal on Selected Areas In Communications, 12(2):279-291, 1994.

[KIRSOPP02A] C. Kirsopp, and M. Shepperd, "Case and Feature Subset Selection in Case-Based Software Project Effort Prediction", Research and Development in Intelligent Systems XIX, Springer-Verlag, 2002.

[KIRSOPP02B] C. Kirsopp, and M. Shepperd, "Making inferences with small numbers of training sets," IEE Proceedings- Software, 149(5), 2002.

[KITCHENHAM04] B. Kitchenham, T. Dyb°a, and M. Jørgensen, "Evidence-based Software Engineering", Proc. of 27th IEEE Intl. Softw. Eng. Conf. (ICSE 2004), Edinburgh: IEEE Computer Society, 2004.

[KITCHENHAM97] B. Kitchenham and S. Linkman, "Estimates, Uncertainty, and Risk." IEEE Software, 14(3), pp. 69-74, May-June 1997.

[KITCHENHAM01] B. Kitchenham, S. MacDonell, L. Pickard, and M. Shepperd, "What accuracy statistics really measure," IEE Proceedings - Software Engineering, 48, pp81-85, 2001.

[KNOR98] E.M. Knor and R.T. Ng, "Algorithm for Mining Distance-Based Outliers in Large Databases," In VLDB, NY, 1998.

[KOHAVI97] R. Kohavi, G.H. John, "Wrappers for feature subset selection," Artificial Intelligence, Vol. 97, no. 1-2, pp.273-324, 1997.

[LANUBILE97] F. Lanubile and G. Visaggio, "Evaluating predictive quality models derived from software measures lessons learned," J. Systems and Software, 38:225-234, 1997.

[LINDVALL05] M. Lindvall, P. Donzelli, S. Asgari. V.R. Basili, "Towards Reusable Measurement Patterns," Proc. Of 11th IEEE Symposium on Software Metrics, METRICS'05, 2005.

[MACKEY91] D.J.C. MacKey, "Bayesian Models for Adaptive Models, " Ph.D. Thesis, California Institute of Thecnology, Pasadena, CA, USA, 1991.

[MAIR00] C. Mair, et al., "An investigation of machine learning based prediction systems", J. of Systems & Software, 53(1) pp23-29, 2000.

[MCCONNELL06] S. McConnell, "Software Estimation, Demystifying the Black Art" Microsoft press, 2006.

[MCGARRY02] J. McGarry, D. Card, J. Cheryl, B. Layman, E. Clark, J. Dean, and F. Hall, "Practical Software Measurement – Objective Information for Decision Makers", Addison-Wesley, 2002.

[MCQUARRIE98] A.D.R. McQuarrie, C. Tsai, "Regression and Time Series Model Selection," World Scientific, 1998.

[MEZIES05] T. Menzies, D. Port, Z. Chen, J. Hihn, "Validation Methods for Calibrating Software Effort Models", Proc. 27st Int. Conf. Software Eng. (ICSE), 2005.

[MIYAZAKI94] Y. Miyazaki, et al., "Robust Regression for Developing Software Estimation Models", J. of Systems & Software, 27(1) pp3-16, 1994.

[MOJIRSHEIBANI96] M. Mojirsheibani and R. Tibshirani, "Some Results on Bootstrap Prediction Intervals," Statistical Society of Canada, 1996.

[MOSES00] J. Moses and J. Clifford, "Learning How to Improve Effort Estimation in Small Software Development Companies," Proc. 24th Ann. Int'l Computer Software and Applications Conf. (COMPSAC), pp. 522-527, 2000.

[MYRTVEIT99] I. Myrtveit, E. Stensrud, "A controlled experiment to assess the benefits of estimating with analogy and regression models," IEEE Transaction on Software Engineering 25(4), pp. 510-525, 1999.

[MYRTVEIT04] I. Myrtveit, E. Stensrud, "Do Arbitrary Function Approximators make sense as Software Prediction Models?," 12 International Workshop on Software Technology and Engineering Practice, pp. 3-9, 2004.

[MYRTVEIT05] I. Myrtveit, E. Stensrud, and M. Shepperd, "Reliability and Validity in Comparative Studies of Software Prediction Models," IEEE Trans. Software Eng., vol. 31, no. 5, pp. 380-391, May 2005.

[NEUMANN02] D.E. Neumann, "An Enhanced Neural Network Technique for Software Risk Analysis," Trans. on Soft. Eng., Vol. 28, no. 9, pp. 904-912, Sept. 2002.

[NIX94] A.D. Nix and A.S. Weigend, "Estimating the mean and variance of the target probability distribution," IEEE International Conference of Neural Networks,

Vol. 1, pp. 55-60, New York, 1994.

[OHSUGI07] N. Ohsugi, A. Monden, N. Kikuchi, M.D. Barker, M. Tsunoda, T. Kakimoto, K. Matsumoto, "Is This Cost Estimate Reliable? – The Relationship between Homogeneity of Analogues and Estimation Reliability." Empirical Software Engineering and Measurement, pp. 384-392, Sept. 2007.

[PAPADIMITRIOU03] S. Papadimitriou, H. Kitagawa, P. Gibbous, and C. Faloutsos, "Loci: Fast outlier detection using the local correlation integral," 2003.

[PARK96] R. E. Park, W. B. Goethert, W. A. Florac, "Goal-Driven Software Measurement," Handbook CMU/SEI-96-HB-002, SEI, August 1996.

[PENDHARKAR05] P.C. Pendharkar, G.H. Subramanian, J.A. Rodger, "A Probabilistic Model for Predicting Software Dvelopment Effort", Transaction on Soft. Eng., Vol. 31, no. 7, pp. 615-624, July 2005.

[PEDHAZUR97] E.J. Pedhazur, "Multiple Regression in Behavioral Research," Orlando, FL, Harcourt Brace, 1997.

[RAO73] C.R. Rao, "Linear Statistical Inference and Its Applications," NY, J. Wiley & Sons, 2nd Ed., 1973.

[RUMELHART86] D.E. Rumelhart, G.E. Hilton, R.J. Williams, "Learning Internal Representations by Error Propagation," Parallel Distributing Computing: Explorations in the Microstructure of Cognition, pp. 318-362, MIT press, 1986.

[SAMSON97] B. Samson, D. Ellison, and P. Dugard, "Software Cost Estimation Using an Albus Perceptron (CMAC)," Inform. and Software Technology, vol. 39, nos. 1/2, 1997.

[SARCIA07] S.A. Sarcia, G. Cantone, V.R. Basili, "A Statistical Neural Network Framework for Risk Management," ICSOFT'07, Barcelona, SP, 2007.

[SARCIA08] S.A. Sarcia, G. Cantone, V.R. Basili, "Adopting Curvilinear Component Analysis to Improve Software Cost Estimation Accuracy," EASE08, Bari, Italy, 2008.

[SHEPPERD07A] M. Shepperd, "Software project economics: a roadmap," FOSE'07, IEEE, 2007.

[SHEPPERD97B] M.J. Shepperd, and C. Schofield, "Estimate software project using analogies", IEEE Transaction on Software Engineering. 23 (12), pp. 736-743, 1997.

[PROMISE] J.S. Shirabad, and T. Menzies, "The PROMISE Repository of Software Engineering Databases," School of Information Technology and Engineering, University of Ottawa, Canada. http://promise.site.uottawa.ca /SERepository, 2005.

[SOLINGEN99] R. van Solingen and E. Berghout, "The Goal/Question/Metric Method," McGraw Hill, 1999.

[SRINIVASAN95] K. Srinivasan and D. Fisher, "Machine Learning Approaches to Estimating Software Development Effort," IEEE Trans. on Soft. Eng., 21(2), pp. 126-137, February 1995.

[STENSRUD02] E. Stensrud, T. Foss, B. Kitchenham, I. Myrtveit, "An empirical Validation of the Relationship between the Magnitude of Relative Error and Project Size," Proc. of 8th IEEE Symposium on Software Metrics, METRICS'02, 2002.

[STONE74] M. Stone, "Cross-validation choice and assessment of statistical predictions," Journal of Royal Statistical Society, B 36, pp. 111-147, 1974.

[STOPPIGLIA03] H. Stoppiglia, G. Dreyfus, R. Dubois, Y. Oussar, "Ranking a Random Feature for Variable and Feature Selection," Journal of machine Learning Research, pp. 1399-1414, 2003.

[VARDEMAN92] S. B. Vardeman, "What About the Other Intervals?," The American Statistician, 46, 193-197, 1992.

[VAPNIK95] V.N. Vapnik, "The Nature of Statistical Learning Theory," Springer, 1995.

[WEISBERG85] S. Weisberg, "Applied Linear Regression", 2nd Ed., John Wiley and Sons, NY, 1985.

[WHITE80] H. White, "A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroscedasticity," Econometrica Vol. 48, pp. 817-838, 1980.

[WITTIG94] G.E. Wittig and G.R. Finnie, "Using Artificial Neural Networks and Function Points to Estimate 4GL Software Development effort," Australian J. Information Systems, vol. 1, no. 2, pp. 87-94, 1994.

[WOHLIN00] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, A. Wesslén, "Experimentation in Software Engineering – An Introduction", Springer, 2000.