A Very Early Estimation of Software Development Time and Effort using Neural Networks

Pedro F. Salvetto
Universidad ORT Uruguay, Laboratorio de Investigación de Sistemas de Información**,** Cátedra de Teoría
Montevideo, Uruguay, Cuareim 1451, 11100, salvetto@ort.edu.uy
Milton F. Martínez
Universidad ORT Uruguay, Cátedra de Teoría**,**
Montevideo, Uruguay, Cuareim 1451, 11100, miltonmz@montevideo.com.uy
Carlos D. Luna
Universidad ORT Uruguay, Cátedra de Teoría
Montevideo, Uruguay, Cuareim 1451, 11100, luna@ort.edu.uy
Javier Segovia
Universidad Politécnica de Madrid
Departamento de Lenguajes Sistemas Informáticos e Ingeniería del Software
Campus de Montegancedo, 28660 Boadilla del Monte (Madrid), fsegovia@fi.upm.es

Abstract

In spite of years of research and development, formal structured estimation of time and effort required to develop a Management Information System (MIS) is still an open problem. Usual estimation techniques applied by now are supported by the not so realistic premise of **requirements stability**, and often human experts are required to apply them. This paper considers models of estimation based on metrics available on early design phase.

Our research work aims to develop formal estimation models for time and effort needed for MIS development. These models use development team efficiency, requirements volatility, development speed and system complexity as input parameters**.** We also identify which input metrics are adequate for measuring system's cognitive complexity and found that useful metrics can be obtained **automatically from the system users´ data views very early on the life cycle with independence of the technology used and without human intervention.** We tested the metrics estimation capability using Artificial Neural Networks (ANN), and thus confirmed an existing functional relation among input and output metrics (time and effort). Once trained, the ANN predicts *effort* needed with a 15% average error and *time* needed with a 30% average error.

1 INTRODUCTION

1.1 Description

Contrary to industrial process, software development process generates intangible products that require intensive communication and coordination, increasing risk and making difficult estimate. Despite the existing estimation models, COCOMO [4], COCOMO II [3], and even its new version called COSYSMO [25], SLIM [19] and others, it is commonplace that the development's cost and time are often under-estimated [2][3][4][15][16][18]. One problem is that models usually include inputs that need expert judgment to be properly estimated, so in real practice they always seem to work better in their own creators' hands and sometimes users referring to the same scenario obtain extremely different evaluations. Another problem is that they need large databases and also highly trained users to calibrate them.

Despite of years of research and development on formal estimation procedures, the structured estimation of time and effort required on MIS development is still an open problem because most extended estimation techniques are supported upon the not so realistic statement of *requirement stability* [21].

Our research aim is to develop formal models (able to be automated) of early *time and effort estimation* on MIS development based on *Relational Databases*, with an *Agile* evolutionary process and with *automated code generation* from formal specifications[1]. These models include as input parameters the development group efficiency, the requirements volatility, the development speed and the system complexity measured **independently from technology**. As a first phase in model construction, we made a causal analysis of factors related to cognitive complexity of a MIS and its influence upon the time and effort. On a second phase, we identified candidate variables for system complexity estimation, searching for any functional relation among inputs and output variables time and effort.

[1] Our criteria referring to the formal specification not only applies to mathematical formulation of formal specification languages. We consider that if the specification contains enough information to produce an application automatically generated in every language and platform (if it is available the correspondent generator) and this specification is published in order to automate its process, then can be considered a formal specification for the present work.

The variables used were calculated from the information automatically obtained from system specifications with an automated tool already explained on a previous work[2] [11][21]. In our case, *Artificial Neural Networks* (ANN) were useful to explore these data.

2 DATA SET CHARACTERISTICS

As we mentioned before, projects were developed with: agile methodologies, evolutionary life cycle, and a knowledge centered paradigm, by using formal specification tools. We considered reduced teams of five or less members integrating users to the team (a well-known *agile* practice). Redundant work is also avoided, as well as everything useless from the users viewpoint. All projects considered were developed using *Relational Databases* technology. The systems studied are MIS *data-strong* type [24] where high complex algorithms usually don't exist or are exceptional. In every case, input metrics of our models were collected automatically by the tool mentioned above, from the system specification excepting *Requirements Volatility* estimated by Project Managers. In the case of outputs (*Time* and *Effort*), we collected 8 projects, measured by detailed registries of team-work scheduling and in the other cases we used estimations provided by Project Managers.

The formal specification tool used has the ability of automating the specification process[3], and takes into account the users non-normalized data vision of the system, the integration of these user data views, the generation of a relational schema that represents this integration, the change-impact estimation in a relational schema, the code generation necessary to migrate data when a change takes place in the user data views that impacts on the Relational Schema, the code-generation in many languages and platforms from a specification (thus, independent from technology) the integration of many specifications in one detecting the possible inconsistencies (that obviously the developers had to solve), the specification publishing, and the use of an unique name for each attribute.

We analyzed projects developed with the same technology to use a stable working methodology and chose teams of similar characteristics to fix team performance that can be considered constant into the data set.

3 METHODOLOGY AND SPECIFICATION TOOL

The specification tool generates: the Relational Database design, the Database creation code and the Application programs required by the MIS in construction, from its formal specification[4]. It is also possible to export the specification by an XML file as to access the specifications details in a tabular format. The tool also saves information in a *Knowledge Database* (KB)[5], which is useful to automatically capture metrics.

From the user data views specifications, the tool automatically generates the Relational Database Schema that represents the information contained in the data views. The specification allows to model the system, generates the Database on third normal form and the Application code in the desired language and platform. When a requirement change takes place, the specification is adjusted and a new Database schema is generated and also the Application code.

3.1 User data views input data

In order to enter the user data views, we included as input data: attributes, groups of repetitive data they could contain and relations among them. The working methodology establishes that each attribute must be named with an unique name in the whole application, with the result that relations between the user data views can be automatically deduced by the tool from the attribute names. Inside a user data view we can identify attribute sets that work together (update, delete and add together). The attribute groups of one level can be repeated once or more times. As an example, in the following chart, we can see the user data views identified during an invoice process. In this example, the * before an attribute indicates that the attribute is a *key* and a group of attributes between parenthesis means that is a *repetitive group*.

---

[2] This tool was developed for student groups as a grade thesis in the Laboratorio de Investigación en Sistemas de Información de la Universidad ORT Uruguay with the support of the Departamento de Informática del Ministerio de Transporte y Obras Públicas

[3] There exists tools with these capacities on the market, as GeneXus www.artech.com, which was the one used in this work because of its availability, market penetration and also because of the support we found in their developers.

[4] More information about GeneXus is available in http://genexus.com. GeneXus a tool that works with formal specifications, generating the application from them in most of the common languages and platforms used worldwide

[5] In this article, Knowledge Base (KB) refers to a set of objects that contains the system specification in a proprietary language.

USER DATA VIEW INVOICE		GENERATED TABLES
InvNum*	Invoice number	**Invoice**
CusCod	Customer code	InvNum*
CusName	Customer name	CusCod
InvDate	Invoice date	CusName
(ProdCod*	Product code	InvDate
ProdName	Product name	**Invoice1**
ProdPri	Product price	InvNum*
InvlinCnt	Line quantity	ProdCod*
InvLinAmo	Line amount	ProdPri
)		InvlinCnt
		InvLinAmo

Adding the user data views CUSTOMERS and PRODUCTS

CUSTOMERS DATA VIEW		GENERATED TABLES	
CusCod*	Customer code	**Customers**	
CusName	Customer name	CusCod*	Customer code
PRODUCTS DATA VIEW		CusName	Customer name
ProdCod*	Product code	**Product**	
ProdName	Product name	ProdCod*	Product code
ProdPri	Product price	ProdName	Product name
		ProdPri	Product price
		Invoice	
		InvNum*	
		CusCod	
		InvDate	
		Invoice1	
		InvNum*	
		ProdCod*	
		InvlinCnt	
		InvLinAmo	

3.2 Specification publishing

The tool publishes the specification details in two ways: text in XML and tabular format by an OLEDB provider. The XML is useful to export the KB and the OLEDB allows developers to access metadata and extend as well as adapt the tool to their own needs. The OLEDB provider shows the specification artifacts and its relations, a property that was a decisive factor on tool's election.

3.3 Accessing system complexity data sources

We have three ways for complexity measuring: a) specification artifacts and its relations b) textual system specifications c) generated source code. These information sources are available along the whole life cycle although obviously the code generation have limitations in early phases. Despite *a* and *b* depend only on the specifications, *c* also depends on the particular generated language. Taking advantage of easy access to specifications artifacts and its relations given by the OLEDB provider, we calculate metrics using the OLEDB's tabular format of them.

4 CANDIDATE METRICS DETERMINATION.

4.1 Research questions

Our main research line was discussed on a previous work [21]. An important part of this research is *Time* and *Effort* Estimation. For that, we should answer the following research questions:

- ¿Which are the early automatically collectable metrics, that predict development time and effort?
- ¿Is it possible to have a system complexity metric entirely independent from the applied technology?
- ¿Can we generalize this metric to other systems developed by different tools, specially that ones which have not the ability to automatically generate the schema from the user data views?

In this paper we aim to answer the first of these questions.

4.2 Metrics characteristics

The use of formal specifications is a must, because it allows us to: *a)* use the same hypothesis as [16]; *b)* collect metrics in an unambiguous and automatic way; *c)* have an early complexity measure, based on requirements; and *d)* collect post mortem metrics in an objective and automatic way, measuring the total complexity of final specifications.

Before selecting metrics, we established their characteristics including practical porperties such as precision, availability, possibility of automate, human resistance and significance from a users' viewpoint. Here follows a summary of the main characteristics found as a result of our research:

Early Availability, as we need metrics available on early states of the development process.
Repeatable, as we need that different observers arrive to the same results.
Automatically collectable, because metrics collection is a hard activity, takes time, and is often inaccurate.
Easy calculation, because we need metrics that can be calculated with simple algorithms.
Not invasive, not interfering with the development process or at least minimize the interfering level.
Clear, as users must clearly understand the metric significance.
Simplicity, as we prefer the metrics with few parameters (Occam's principle).

4.3 Technology independence

We search for a complexity measure in a *Knowledge Database*, independent from technology. This independence is an immediate consequence of the fact that, from the formal specification it is possible to generate the whole application on different languages and platforms.

4.4 Complexity

We need objective measures, repeatable and automatable avoiding the need of experts, whose viewpoints are often biased [16][19]. From the users' data views, the Relational Database Schema that represents their essential information can be obtained, without considering each particular data view and their possible intersections. This schema is automatically tool-generated.

4.4.1 Our working hypothesis

1) If only valuable and not redundant functionality (from a business viewpoint) is generated**,** then **system essential complexity is determined by the integration of its users data views. This complexity can be measured from the relational schema automatically generated from the integration of these data views.**
2) This essential complexity measure contains enough information to develop effort and time estimation models. From our point of view, these metrics are certainly *very early* ones, since the schema is generated automatically from the users' data views.
3) Although we don't have a deep understanding of complex internal processes and interactions that take place during the development process, we can observe it and construct models that consider their global result. These models could have an acceptable accuracy if we diminish the sources of variability using tools that automate code-generation and module integration tasks and working with reduced teams with users integrated to them and applying a standard methodology.

4.4.2 Complexity metrics cognitive focused

System specification represents its essential complexity, as it is possible to generate the application on different languages and execution platforms. Fig. 1 shows a causal analysis of main factors of the Knowledge Base Cognitive Complexity (KBCC). Systems that we are studying are data strong type, according to De Marco [24] who proposed that these systems have to be studied on a data basis. There are two main cognitive complexity sources of data (Data Structure's Cognitive Complexity, DSCC) and the processes and functions (Process Structure´s Cognitive Complexity, PSCC). Data contributes to the complexity by their own structure. Calero, Piattini, et al [9] define metrics as to evaluate the relational databases complexity classified on (1) **table oriented metrics** Number of unique Attributes (NA) and Referential Degree (RD) y (2) **schema oriented metrics**, Depth Referential Tree (DRT) y Normality Ratio. The RD is the number of foreign keys of the schema. We add the number of tables (NT). The DRT is the major referential path length on the schema. Cycles are only considered once. The normality proportion is the number of tables on third normal form divided by the total number of tables of the schema, a metric useless in this case because the schema is automatically generated on third normal form and then its value is always 1. Calero, Piattini [9] studied formal properties that contains these metrics and they expressed that more research is needed in order to validate them empirically. Shao y Wang [23] proposed a complexity metric based on cognitive weights, useful to measure the contribution of the PSCC to the KBCC. Anyway, this metric is based on control structures, available in the final phases of the development process, so they will be considered in late estimation models (in future works), exploring a weight pattern in a knowledge database, that allows us to manage the contracts in a project on a released functionality basis.

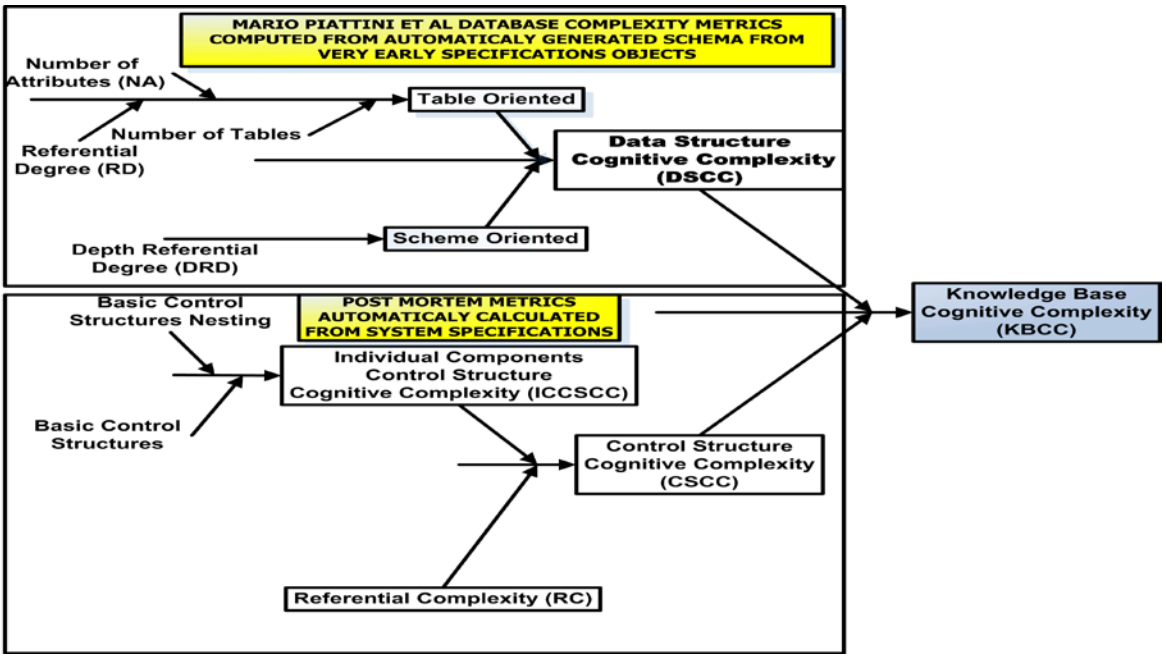

Fig 1 Causal Analisys of the Cognitive Complexity sources

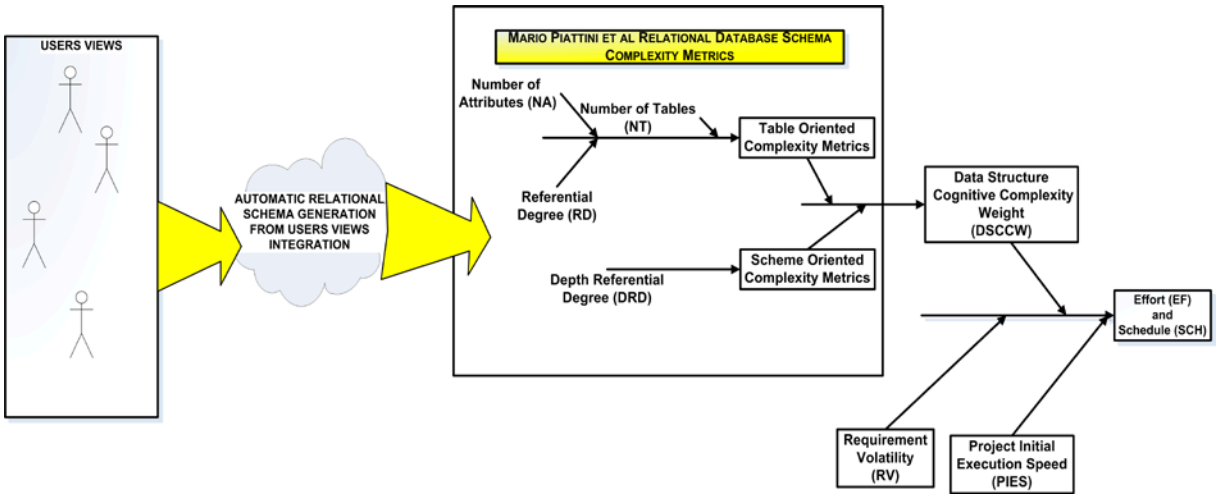

Fig 2 Elements from the models of structured estimation of time and effort

Two main factors that have influence upon *Time* and *Effort* are the project development speed and the requirements volatility. The effort was measured in man-months, the time in months and the speed as effort divided by time. As it is desired to use early metrics, we used *average execution speed* of the initial 10% of the project's execution time. The requirement volatility (RV) was measured as a percentage value according to Nogueira's formula[16]:

$$RV=\frac{DR+BR}{NR}*100$$

Where DR = Number of Requirements Deletes (dead), BR = Number of Born (new) Requirement and NR = Total Number of Requirements. As none of the projects arranged for this registration was requested during the interviews with Projects Managers that reckoned it in <10% low, 10-20% average and greater than 20% high, being assigned the values 10%, 20% and 30% respectively.

5 WHY USING NEURAL NETWORKS?

As Software Engineering is a social activity, it is difficult to predict results in terms of time and effort. Moreover, it is a dynamic domain, and the results depend on the specific project and may vary depending on time. This

is the main reason for using Dynamic Systems, as Neural Networks that have the ability to adjust them for each project [2].

Another important reason is that at the beginning of this research, we didn't have a mathematical expression of the relation among early variables and *Time* or *Effort* needed in every project. Considering previous research on this area, it seems that it's not an easy task to find symbolic formulas that can express these relations, so it is common to use numerical approximations, and Artificial Neural Networks (ANN) are useful to this [10].

Finally, ANNs were used at the very first in our work, as an exploration data tool (i.e.: to show functional dependences among data), and also as an alternative procedure to validate statistic procedures applied in our research.

6 DECISION OF USING OUR OWN IMPLEMENTATION

From the two alternatives of using a standard parametrical implementation of ANN or implementing one, we chose the second option due to the following reasons:

- Because we needed a good level of convergence on effort estimation with a reasonable relative error, which is not an easy goal, it is preferable to have the possibility of defining the ANN's topology, parameters and training algorithms in order to optimize the convergence and the result's precision.
- As in the data exploration phase we used both crude and normalized data (using logarithms), the transformation function needed to be frequently swapped from a logarithmic function (which varies from 0 to 1) to a hyperbolic tangent function (which varies from –1 to 1) depending on the data set, and therefore, that showed the convenience of implementing an automatic swap procedure on every Process Element of the ANN, depending on the range of data inputs.
- As some of the inputs were normalized sometimes linearly and sometimes not, the training stop-methods had to take into account this, considering some relative error measures instead of the Least Square Error (LSE) of common use in most of the standard ANN implementations available [17].
- Last but not least, the large amount of data contained in each set made it necessary to have an automated mechanism of loading data sets for ANN's training, so we implemented our ANN to receive the inputs in the format we have them, and by this way it was easy for us to test several sets of data and check the results once we found the best configuration.

In the Fig 3 we show an image of the graphical interface, while testing data:

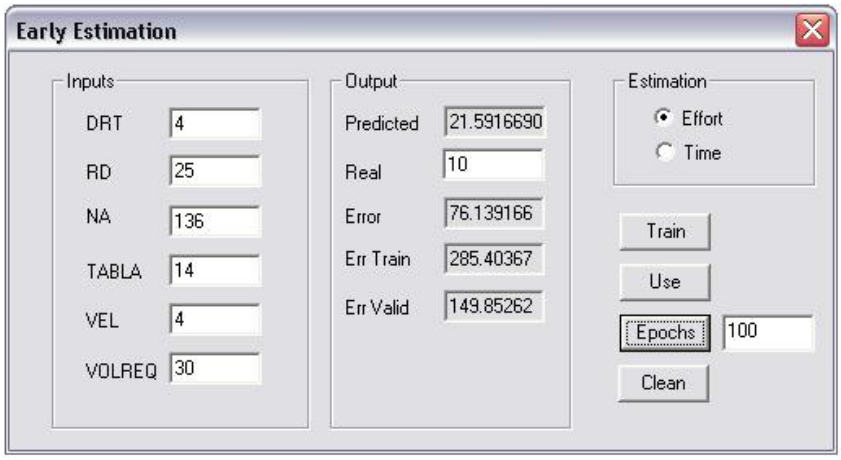

Fig 3 Graphical Interface while Testing Data

7 TOPOLOGY AND TRAINING ALGORITHMS

We implemented a Multi-Layer Perceptron (MLP) with three layers with a Backpropagation training algorithm, an algorithm that uses the Descendent Gradient of the error surface in order to minimize the global error of the ANN. We had tried several levels of Learning Rate until we found the best for our domain of data and also added Momentum to each Process Element (saving the previous state of each Neuron, to monitor the change ratio and thus adjusting the convergence speed of the net, in order to avoid oscillations) [13]. We also implemented error control both for the training set and for the testing set, for the cross-validation procedure. Finally we randomized initial data and epochs loading (order of recurrent load of data sets) to avoid stopping the training phase in local minima, which we experienced as useful methods to increase convergence in other domains of data [14].

8 FUNCTIONAL DEPENDENCY

First we selected a set of eight projects in which data exploration statistical procedures showed connection between the input variables and the effort needed to build the software. We trained an ANN with the whole set (because such a low number of projects wont allow us to apply cross-validation) and obtained a mean relative error lower than 0.5 %, which showed us a functional dependency between the inputs and the output, although we were not be able to generalize this to any project, since we couldn't validate the dependency with a cross-validation method (using around the 70% of the data set for training and the rest for testing), due to the low number of projects, each of them contributes considerably to the functional formulation, with the result that eliminating any of them throws an unacceptable level of error in the Estimation of the ANN. Anyway, this first phase of researching threw enough basis to go back to the theoretical frame, selected the correct variables, and formulated a mathematic exponential expression that fits with the results of this first 8-projects data set.

9 MODEL VALIDATION

As we considered, the Neural Networks were trained from the 70% of the data set randomly selected and they were validated evaluating its predicting ability for the total data set. The statistic performance of the Neural Network was evaluated using the determination coefficient (R^2) and the precision and consistence was measured by using the mean relative error value (MRE) and the prediction ability on a specific level k (PRED(k)) defined by Conte et al [8].

The relative error (RE) , MRE y PRED(K) are calculated with the following formulas.

$$RE = \frac{\text{Actual Value- Estimated Value}}{\text{Actual Value}} \qquad MRE = \frac{\sum |RE|}{N} \qquad \text{PRED(K)} = \frac{\text{Number of Observations with } |RE| < K}{\text{Total Number of Observations}}$$

The metric PRED (K), is the quantity of predictions where the absolute value of the RE is less or equal to K divided into the total quantity of the observed values. In most cases, the dependent and independent variables where normalized using logarithms, but it is important to establish that the errors where studied taking back the variable-change, that means that the real predictive power of the models was evaluated instead of their logarithms, which probably were much higher. Later, the results were evaluated according to the following table from [1].

Qualification	**Consistency**	**Accuracy**
Excellent	**MRE ≤ 0.20**	**PRED(20) ≥ 0.80**
Good	**MRE ≤0.25**	**PRED(25) ≥□0.75**
Acceptable	**MRE ≤□0.30**	**PRED(30) ≥0.70**
Poor	**MRE > 0.30**	**PRED(30) < 0.70**

Table 1 ANN Estimation Ability Evaluation

Inputs: DRT, RD, NA, NT, IPES y RV. Also were taking into account the user data views, but they were not successful in terms of convergence, which is not surprising, considering that the same system can be viewed through different sets of user data views, therefore it is more reasonable to measure the essential complexity by the relational schema automatically derived from de user data views.

Outputs: EFFORT and TIME.

10 MODEL ESTIMATION ABILITY

In a second phase, we collected a larger number of projects (25), and then trained another ANN, adjusting parameters and topology, and then we found that our model fits with a reasonable error excepting some outliers in the case of time estimation (in a difficult context such as software estimation) , as we can see in Table 2 and Fig 4.

ANN TIME ESTIMATION ABILITY										
PRED(K)										
RE	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Frequency	6	5	4	3	2	2	1	1	0	1
% Cumulative	0,24	0,44	0,60	0,72	0,80	0,88	0,92	0,96	0,96	1,00

RELATIVE ERRORS	MAX	58,25%	ABSOLUTE RELATIVE ERRORS	MAX	100,00%
	MIN	-100,00%		MIN	1,33%
	AVE	-6,57%		AVE	29,57%
	STD	38,55%		STD	24,90%

Table 2 ANN Time Estimation Ability

ANN EFFORT ESTIMATION ABILITY					
PRED(K)					
RE	10%	20%	30%	40%	50%
Frequency	14	4	1	3	3
% Cumulative	0,56	0,72	0,76	0,88	1,00

RELATIVE ERRORS	MAX	39,38%	ABSOLUTE RELATIVE ERRORS	MAX	44,00%
	MIN	-44,00%		MIN	0,00%
	AVE	-3,04%		AVE	15,14%
	STD	22,13%		STD	16,15%

Table 3 Ann Effort Estimation Ability

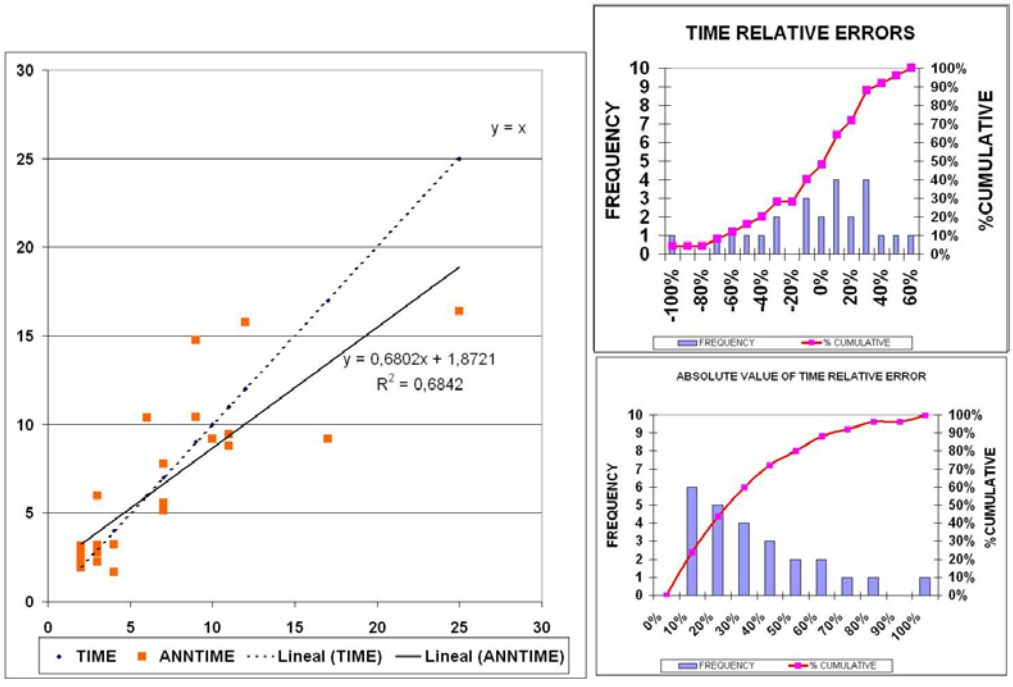

Fig 4 ANN Time Estimation Ability Graphics

And for the Effort estimation, we found that the model fits with a lower relative error than time estimation, and a considerable functional correlation between real and predicted data [12] as Table 3 and Fig 5 shows.

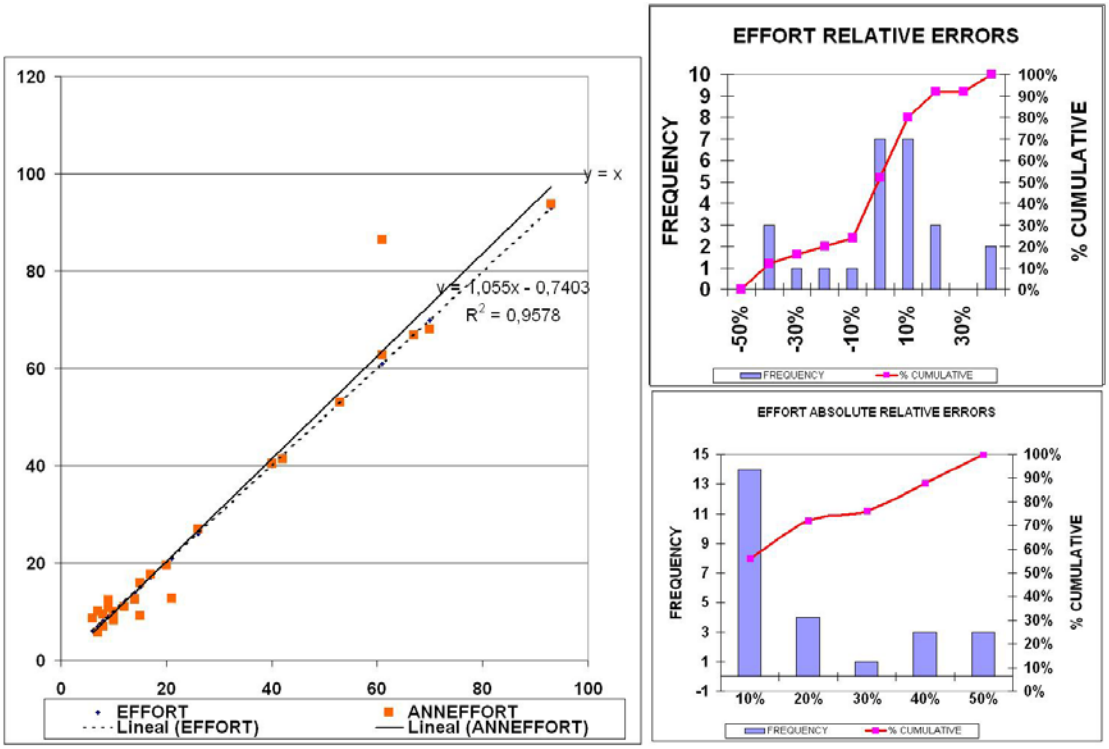

Fig 5 ANN Effort Estimation Ability

We estimate that the effect of outliers can be minimized by training the ANN with a large number of projects. Despite of that, we think we can conclude that the theoretical schema considered on this paper has enough experimental support to be a useful tool for time and specially for effort estimation.

11 CONCLUSIONS

It was possible to design, construct and train Neural Networks with DRT, NA, NT, IPES y RV inputs that adequately converged on time and effort Estimation. These ANN can predict the effort with a relative error of 50% on the worst case and qualify as acceptable according to table 1 criteria. These ANN are an useful tool for early estimation and their design and training were of valuable help to us to properly identify the most worthy input parameters for our research estimation models. This confirms our hypothesis in terms that it exists a functional relation among the independent variables DRT, NA, NT, IPES y RV and the dependent variables TIME and EFFORT.

This result also confirms our working hypothesis and what earlier De Marco envisaged [24] in the sense that data-strong systems complexity could be estimated from data. The metrics proposed by Calero, Piattini et al [9] have proved to be very useful. Neural Networks are also useful for our research taking into account that they are (1) dynamical systems, with the ability to adapt to new information and to (2) the characteristics of a particular team. Nevertheless, they don't provide an explicit mathematic expression of the functional relation. The techniques used in this work showed a functional correlation in the case of time estimation and specially in the effort software estimation.

12 FUTURE RESEARCH LINES

To determine the functional dependence in an explicit way. We are already working on this area[22]. To define a complexity metric early estimable from the complexity measures proceeding from the relational schema. To define a weight pattern of the knowledge database, that allows to manage the project contracts on a released functionality basis. To increase the size of the data set for Neural Networks training.

13 ACKNOWLEDGMENTS

This work would not have been possible without the collaboration of Karina Santo, José Luis Chalar, Gustavo Carriquiry and Claudia Araujo from Artech Consulting, Enrique Latorres and Jose Luis Subelzú from Transport and

Public Works Ministry of Uruguay Informatic Department, Juan Andrés Leiras from the Police Sanity Infomratic Department, Óscar Camargo from the Work University of Uruguay and ORT Uruguay University and Gonzalo Pérez y Joaquín González from CONEX Consulting. We are also grateful to Nicolás Jodal, Karina Santo and José Luis Chalar of Artech Consulting for their valuable help. We would like especially to mention the support and intelligent suggestions of Julio Fernández, ORT University´s Academic Development Dean and Ernestina Menasalvas of UPM.

Grade and postgraduate students of ORT Uruguay University at the Information Systems Research Lab and Theory Chair supported this investigation. The visits to UPM of the first author are financed by the BID's Technological Development Program.

14 REFERENCES

[1] Bennett , Warren. Predicting software system development effort Very early in the life-cycle using Idef0 and idef1x models. Phd Dissertation Submitted to the Faculty of Mississippi State University. 1996.

[2] Boehm, B. A Spiral Model of Software Development and Enhancement. Computer. May, 1988.

[3] Boehm, B. et al. Software Cost Estimation with COCOMO II. Prentice Hall, 2000.

[4] Boehm, B. Software Engineering Economics. Prentice Hall, 1981.

[5] Boehm, B. Software Risk Management. IEEE Computer Society Press. 1989.

[6] Briand Lionel, El Emam Khaled , Morasca Sandro. On the Application of Measurement Theory in Software International Software Engineering Research Network technical report #ISERN-95-04.

[7] Common Software Internacional Consortium Full Function Point Measurement Manual. (THE COSMIC IMPLEMENTATION GUIDE FOR ISO/IEC 19761: 2003) VERSION 2.2 January 2003

[8] Conte, Samuel D., H. E. Dunsmore, and Vincent Y. Shen. 1982. Software engineering metrics and models. Menlo Park, CA: Benjamin/Cummings.

[9] Coral Calero, Mario Piattini, Macario Polo, Francisco Ruiz. Grupo ALARCOS, Departamento de Informática,Universidad de Castilla La Mancha. Métricas para la evaluación de Complejidad de Bases de Datos Relacionales. Computación y Sistemas Vol. 3, Nº 4, pp 264-273, 2000, CIC – IPN. ISSN 1405-5546.

[10] Freeman, James A. and Skapura, David *Neural Networks. Algorithms, Applications and Programming Techniques*, Addison-Wesley U.S.A. 1991

[11] Latorres, Salvetto, Nogueira, Larreborges. Una herramienta de apoyo a la gestión del proceso de desarrollo de software. Proceedings CACIC 2003. La Plata, 2003.

[12] Lothar Sachs "Estadística Aplicada" 263-264, 341-342 Ed. Springer 1967.

[13] Martínez, Milton *Diseño, Entrenamiento y Validacion de Perceptrones Multi-Capa*, COMPUMAT 2003

[14] Martínez, Milton F. , Martínez V., Teachers assessment: a comparative study with Neural Networks, Delta 03, New Zealand, 2003.

[15] Nogueira, J.C. A Formal Estimation Model for Software Projects. International Conference on Computer Science, Software Engineering, Information Technology, e-Business, and Applications (CSITeA 02). Foz do Iguacu, Brasil, 2002.

[16] Nogueira, J.C. A Formal Risk Assessment Model for Software Projects. Ph.D. Dissertation. Naval Postgraduate School, 2000.

[17] Picton, Phil , *Neural Networks*, pp. 37-47.

[18] Putnam, L. and Myers, W. Five Core Metrics: The intelligence behind successful software mangement. Dorset House.2003

[19] Putnam, L. and Myers, W. Industrial Strength Software. Effective Management Using Measurement. IEEE Computer Society Press, 1997.

[20] Randy, Julian *Multi-Layered Neural Networks*, Lilly Research Laboratories

[21] Salvetto, P., Nogueira J.C. Size estimation for Management Information Systems Based on Early Metrics. Proceedings CSITeA03. Río de Janeiro, 2003.

[22] Salvetto, P., Nogueira, J.C., Segovia, Javier. Modelos Automatizables de Estimación muy Temprana del Tiempo y Esfuerzo de Desarrollo de Software de Gestión. Accepted XXX Conferencia Latinoamericana de Informatica (CLEI2004). SUBMISSION CODE (also known as user id) 129

[23] Shao, J. and Wang Y. A new measure of software complexity based on cognitive weights. Can. J. Elec. Comput. Eng, Vol 28,Nº 2, April 2003.

[24] T. DeMarco, Structured Analysis and System Specification New York:Yourdon, 1978.

[25] Valverdi, Ricardo, Bohem, Barry and Reifer Donald. COSYSMO: A Constructive Systems Engineering Cost Model Coming of Age. *Submitted for the INCOSE 2003 Symposium, June 29 – July 3 2003, Washington, DC*