Revista Cubana de Ciencias Informáticas
Vol. 8, No. Especial UCIENCIA 2014, Mayo, 2014
ISSN: 2227-1899 | RNPS: 2301
Pág. 1-13
http://rcci.uci.cu

Tipo de artículo: Artículo original
Temática: Ingeniería y Gestión de Software
Recibido: 7/05/2014 | Aceptado: 21/05/2014

A Comparative Study of Three Test Effort Estimation Methods

Un estudio comparativo de tres métodos de estimación de esfuerzo de pruebas

Natália França Felipe [1], Raphael Pena Cavalcanti [1], Eduardo Habib Bechelane Maia [1], Weber Porto Amaral [1], Augusto Campos Farnese [1], Leonardo Daniel Tavares [1], Eustáquio São José de Faria [2], Clarindo Isaias Pereira da Silva e Padua [1], Wilson de Pádua Paula Filho [1*]

[1] Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte, MG – Brazil. E-mail: {natf, raphaelp, habib, weber, farnese, tavares, clarindo}@dcc.ufmg.br; wppf@ieee.org

[2] Pontifícia Universidade Católica de Minas Gerais (PUC-MG) – Betim, MG – Brazil. E-mail: eustaquio@pucminas.br

Abstract

Effort estimation is a big challenge for those trying to manage a project. In a software development project, testing is essential to assure product quality. However, it is a time consuming activity, and its work must be estimated for successful project execution. In our research, we concentrate our efforts on comparing some known methods of test effort estimation. So, this paper aims to analyze three different test effort estimation methods and compare them with the effort spent on real projects. Firstly we compare two widely used effort estimation methods: Test Point Analysis (TPA) and Use Case Points (UCP). Thereafter, we create an artificial neural network (ANN) based on the TPA, trained to estimate the testing work in software development projects, and compare it with pure TPA, to check which of them results in better estimates. Analyzing the experiment results, we concluded that the neural networks gave the best results, followed by TPA and then UCP.

Keywords: Software Engineering, Test Effort Estimation, Test Point Analysis, Testing, TPA, UCP, Use Case Points.

Revista Cubana de Ciencias Informáticas
Vol. 8, No. Especial UCIENCIA 2014, Mayo, 2014
ISSN: 2227-1899 | RNPS: 2301
Pág. 1-13
http://rcci.uci.cu

Resumen

La estimación de esfuerzo es un gran reto para los que gestionan proyectos de desarrollo de software. Estos proyectos deben hacer pruebas para asegurar la calidad del producto. A pesar de la importancia de las pruebas, esa es una actividad costosa en tiempo, así que el esfuerzo debe estar bien planeado para la ejecución exitosa de un proyecto. Este trabajo tiene como objetivo comparar tres métodos de estimación de esfuerzo de pruebas, y compararlos a su vez con el esfuerzo invertido en proyectos reales. En primer lugar comparamos dos métodos de estimación de esfuerzo ampliamente usados: Test Point Analysis (TPA) y Use Case Points (UCP). Después de eso, creamos una red neuronal artificial (RNA) basada en la TPA, entrenada para estimar el esfuerzo en pruebas de proyectos de desarrollo de software. Se comparó con TPA para comprobar cuál de ellos resultaba en estimativa más cercana de la realidad. Con base en los experimentos, se concluyó que las redes neuronales dieron los mejores resultados, seguidas por TPA y luego UCP.

Palabras clave*: Estimación de Esfuerzo de Pruebas, Ingeniería de Software, Pruebas de software, Test Point Analysis, TPA, UCP, Use Case Points.*

Introduction

Every software development company focuses on product quality assurance. Clients demand reliable software that meets their needs and expectations, with a minimum bug count. Software testing has being widely used as an activity to help developers achieve software quality. Moreover, a successful software development project must be delivered on time within budget and high quality software, while meeting customer expectations. Testing is a crucial and time consuming activity in every software development project; therefore, it must be planned, and its execution must be monitored.

A study with 65 companies showed that 44 of them have an independent testing team (Zhu 2008). Those teams need independent resources and must follow a specific test management plan. This plan must consider the work needed to perform the software testing process. That requires techniques for estimating the effort spent on testing.

Software testing is defined as the activity conducted to verify the software implementation results through test planning, design and execution (Pádua 2009). Without consistent effort estimation the test manager will not be able to plan the resources to be used on that activity (Sharma 2012). According to Black (2012) there is a set of characteristics of good test effort estimations, based on the most probable cost and effort of each task, as listed below:

- It is based on the knowledge and wisdom of experienced practitioners.

- It is supported by those who will do the work.
- It is specific and detailed about costs, resources, tasks, and people.
- It is based on the most likely cost, effort, and duration for each task.

Test Effort Estimation: An Overview

In a software development project, the project work must be estimated in order to plan time and cost (Sommerville 2003). Test estimation tries to determine how much effort will be needed execute the planned tests for a given project, and is essential to forecast the cost of a project (Lopes 2008). Estimation enables the project manager to predict how long will take to test a system, how many workers will need to be allocated, and how much will it cost to perform the tests. To achieve high accuracy on estimation, it is not enough to have historical data: we need to use metrics and techniques.

The test effort is usually estimated within the software development effort estimation, not as an individual task. According to (Nageswaran 2001) the most common methods for estimating test effort are:

- **Ad-hoc methods**: tests are performed until the project manager decides otherwise, or the test budget is over;
- **Percentage of software development effort**: it is assumed that the effort needed for testing is some fixed fraction of the development work;
- **Function Points Estimation**: test effort is based on the functional size of the functions being tested (in function points).

Besides the methods cited above, there are other three important methods – not so widely used – that are the main subjects of this paper. **1)** Use Case Points (Nageswaran 2001); **2)** Test Point Analysis (Van Veenendaal 1999); **3)** Neural Networks Based Estimations.

Use Case Points Based Test Effort Estimation

The Use Case Point (UCP) was created in 1993 by Gustav Karner (Belgamo e Fabbri 2004).

"A use case captures a contract between the stakeholders of a system about its behavior. The use case describes the system's behavior under various conditions as it responds to a request from one of the stakeholders, called the

primary actor. The primary actor initiates an interaction with the system to accomplish some goal. The system responds, protecting the interests of all the stakeholders. Different sequences of behavior, or scenarios, can unfold, depending on the particular requests made and conditions surrounding the requests. The use case collects together those different scenarios." (Cockburn 2001) The UCP technique allows estimation on the project initial phase, if it based on use cases. The steps to estimate the project's effort based on the UCP are: (1) Actors evaluation; (2) Use cases evaluation; (3) Adjust factors calculation; (4) Final UCP calculation; (5) Final effort calculation.

In order to estimate test effort, some changes on those steps may be needed. Some changes, proposed by (Nageswaran 2001), adapt the steps to estimate test tasks effort. The author detailed those steps as: (1) Set the weight for each actor; (2) Set the weight of each use case; (3) Set the UCP; (4) Set the technical and environmental factors; (5) Set the UCP adjusted; (6) Calculate the final effort; (7) Add some percentage for complexity and management;

The UCP technique has 2 main problems:

1. There is a lack of standardized definition of the parameter used by technique. Therefore, estimates made by different people can get very different results.
2. The complexity of a use case can vary greatly, depending on which design is used.

Test Point Analysis Method

The Test Point Analysis (TPA) is based on Function Point Analysis and considers the system's size – calculated in Function Points – as the basis for estimating. The steps to be taken to execute the TPA are shown in Figure 1.

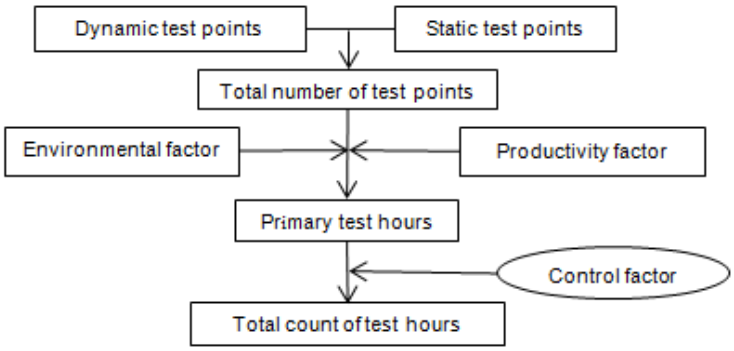

Figure 1: An overview of Test Point Analysis procedure **(Van Veenendaal 1999)**

In the first step for the TPA the total count of test points is calculated. It is given by the sum of the dynamic test points – which is the sum of the test points assigned to the individual functions – and static test points – which is the count of test points necessary for testing the static measurable quality characteristics.

After that, the primary test work (in person-hours) is calculated by multiplying the total count of test points by the calculated environmental factor and the applicable productivity factor. The result represents the volume of work involved in the primary testing activities.

Neural Networks Based Estimation

The third estimation technique used in this paper is based on artificial neural networks (ANN). ANN is a mathematical model inspired on the capability of the human brain to perform very high complexity tasks. It can handle with three main purposes (but not limited to): (i) nonlinear function regression, (ii) data classification and (iii) time series forecast (Haykin 1994). In the context of regression and time series forecast, it can be used as an aid in decision making in project management (Peña, et al. 2013).

As a human brain, an ANN has simple calculation units called "neurons" or "perceptron". The name "perceptron" comes from the pioneer McCulloch and Pitts work, where one simple neuron could solve classification problems in a linearly separable space (McCulloch e Pitts 1943).

Mathematically speaking, the perceptron is modeled as (Freeman e Skapura 1991):

$$\hat{o} = \varphi\left(\sum_{i=1}^{n} x^i . w^i + b\right)$$

Where n is the input dimension, x^i is the *i*-th input dimension, w^i is the *i*-th neuron weight parameter, b is the neuron bias, $\hat{o}$ is the neuron output and $\varphi(.)$ is the neuron activation function. A graphical representation of neuron and logistic sigmoid shape are shown in the figures 2 and 3 (B. e van der Smagt 1996). Several activation functions can be used. The most common, mainly in the hidden layers, is the logistic sigmoid, on the form:

$$\varphi(u) = \frac{1}{1 + e^{-u}}$$

Revista Cubana de Ciencias Informáticas
Vol. 8, No. Especial UCIENCIA 2014, Mayo, 2014
ISSN: 2227-1899 | RNPS: 2301
Pág. 1-13
http://rcci.uci.cu

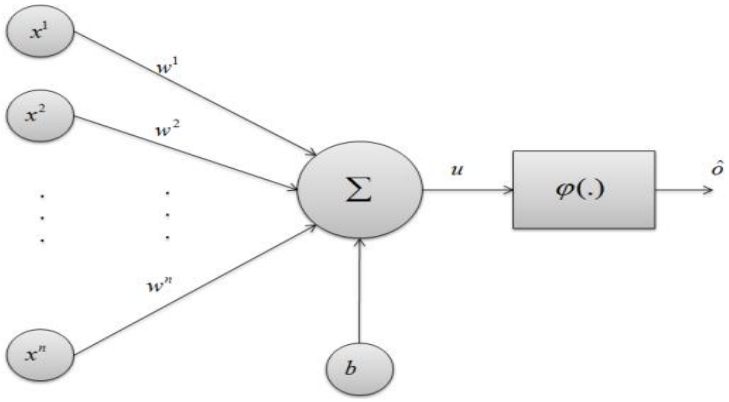

Figure 2 - Graphical representation of a perceptron.

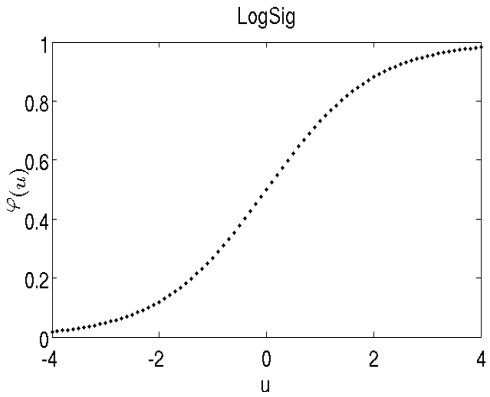

Figure 3 - Logistic sigmoid shape.

Most of real world mathematical problems are not easy to resolve. They can contain non linear characteristics, discontinuity, noises, among other difficulties. In order to transpose this situation, it is possible to organize the neurons in layers, increasing significantly the processing capacity, so that the problem becomes easily resolvable (Haykin 1994).

One of these approaches is the multilayer perceptron (MLP) neural network. The figure 4 shows a graphical representation of an MLP-NN.

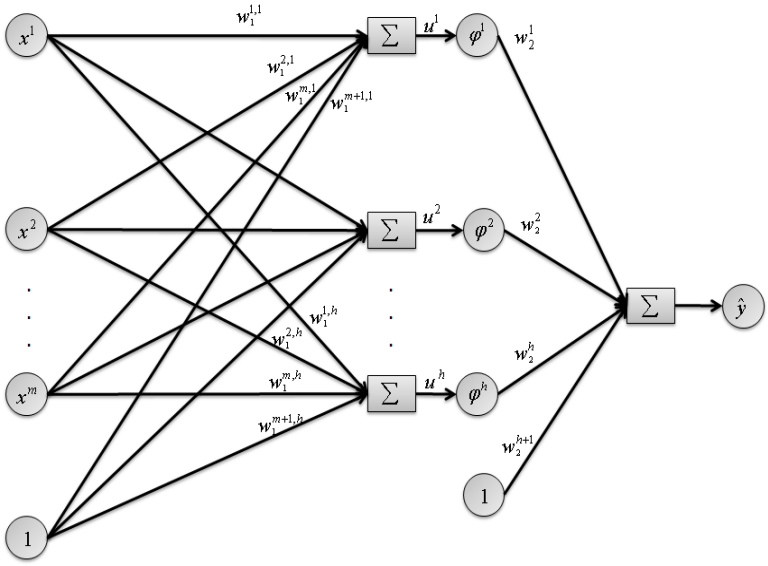

Figure 4 - Graphical representation of an artificial neural network

In this context, the neurons belonging to the same layer use the same activation function. Generally, the hidden layer has a non-linear function, such as the logistic sigmoid, and an output layer uses an activation function consistent with the purpose of the network. In this work, the output layer uses linear function (Ludemir, Braga e Carvalho 2000).

Once chosen topology, it is necessary to conduct the training phase. The most commonly used method for this is the back-propagation. The back-propagation is a supervised training method based on local search optimization, more specifically, the gradient decent method.

The back-propagation works as follows: all the training data are presented to the MLP-NN so that they pass through all the layers (from input to output layer). This is the feed forward step. Then the difference between estimated output and the desired output is calculated. This difference is called "empirical risk". The aim of the method is then to minimize the empirical risk, so that, the estimated output is as close as possible to the desired output. In this study the empirical risk chosen is the Root-mean square error (RMSE).

The error is then back-propagated (from the output layer to the input layer), and thus adjusting the associated parameters. Each iteration of the described process is called 'epoch'. The process is repeated until one of the conditions is satisfied: (i) low error value or (ii) number of time limit reached (Tang, Kay Chen e Yi 2007).

The topology used in this work is the three-layers neural network, being the first the input layer, composed by 6 neurons, the second as the hidden layer, with 4 neurons and the last as output layer with 1 neuron. This approach is interesting once it meets all the requirements of the universal approximation theorem (T.J. 1981) (Hornik, Stinchcombe e White 1989).

To use an ANN for estimation, a user has to setup the same attributes used in the TPA. Those attributes will be preprocessed – based on the TPA – so the network will only receive as input data generated as intermediate results from the TPA, listed below:

- **PF** – Amount of Function Points of the Use Cases being tested
- $\mathbf{D_f}$ – Specific factors for each function
- $\mathbf{Q_d}$ – Dynamic Quality Characteristics
- **E** – Environment Factor
- **P** – Productivity Factor
- **%** – Plan and control estimation margin

Revista Cubana de Ciencias Informáticas
Vol. 8, No. Especial UCIENCIA 2014, Mayo, 2014
ISSN: 2227-1899 | RNPS: 2301
Pág. 1-13
http://rcci.uci.cu

Evaluation Methodology

In order to evaluate the three methodologies for test effort estimation, we used data from real projects and help from Neural Network Specialist. To estimate the work using the TPA and the UCP methods, we developed two different applications to automate the effort estimation, reducing time consumption and probability of error when estimating. Those applications accepted project data as input (e.g. actors' weights and use cases weights) to output the work estimation. First, we compared the TPA and the UCP, to find the best between them. Then we compared Neural Network and the TPA (which went better in the first experiment).

Another application was built to estimate work using neural networks. The application was also able to be trained using data from past projects of a company. After training the neural network, the results were closer to the real effort spent by the company. In other words, after training, the neural network provides better estimation.

The data used to evaluate and compare the estimation methods were gathered from two real software development projects, hereinafter called "Project A" and "Project B". The experiments were conducted using data from both projects to compare the TPA, UPC and Neural Network estimations.

There were ten use cases from Project A and other ten from Project B, all of them with distinct characteristics. With the help of a test analyst, the use cases from Project A were analyzed in order to generate input for the TPA, the UCP and neural networks, so that the test effort (in person-hours) could be calculated. Data from Project B was used to re-training the neural network to achieve better results.

The complexity for each use case and the actors count grouped by complexity can be seen in Table 1 for Project A and in Table 2 for Project B. To use the TPA there is a set of aspects that must be considered, shown in Table 3.

Table 1 Classification for each Use Case from Project A and actors count, grouped by complexity

Use Case #	1	2	3	4	5	6	7	8	9	10
Complexity *	S	S	S	S	M	S	S	S	M	S
∑ **Actors (S)** *	1	1	1	1	1	1	1	1	1	1
∑ **Actors (M)** *	0	0	1	0	0	0	0	1	1	1
∑ **Actors (C)** *	0	0	0	0	0	0	0	0	0	0

* S – Simple; M – Midterm; C – Complex

Table 2 Classification for each Use Case from Project B and actors count, grouped by complexity

Use Case #	1	2	3	4	5	6	7	8	9	10
Complexity *	S	C	S	S	S	M	S	M	C	M
∑ **Actors (S)** *	1	1	1	1	1	1	1	1	1	1
∑ **Actors (M)** *	0	3	1	1	1	2	1	0	2	1
∑ **Actors (C)** *	0	0	0	0	0	0	0	0	0	0

* S – Simple; M – Midterm; C – Complex

Table 3 Weight setup for technical factors and environment

Factor	Description	Value	Weight
T1	Testing tools	5	5
T2	Documented Inputs (Use Cases)	5	5
T3	Development Environment	2	4
T4	Testing Environment	3	5
T5	Testware Reuse	3	1
T6	Distributed Systems	4	4
T7	Performance	2	4
T8	Security Characteristics	4	3
T9	Interface Complexity	5	3

Results and Discussions

When estimating using the UCP we verified a high estimation error in the results. The final results for each project exceed the actual work in about 140%. As a good estimate should be close to reality, we can conclude that the UCP results were not good. The comparison chart can be seen below in Figure 2.

The UCP does not distinguish between normal and alternate flows, but, in fact, alternate flows usually require less time to be tested then a normal flow (de Almeida, de Abreu and Moraes 2009). Moreover, in order to consider technical and environmental factors, the UCP requires data from many projects. Since we did not have such data, we had to use educated guesses, in order to choose values to these proprieties to estimate the test effort.

Besides the problems cited above, another aspect that may have influenced the error on the UCP estimation is the conversion factor used to find the total work value. On the UCP description (Nageswaran 2001), it is not shown how to find that factor; it is only stated that it can be determined by the organization, according to some other factors. To determine this factor using historical data would require a long period applying this technique in other projects. So we used the calculation procedure suggested in (Banerjee 2001). Other studies on the same technique (de Almeida, de Abreu e Moraes 2008) did not have satisfactory results either. When comparing the results with the real work, they had an error of about 1300%. In our experiments, we had an error of about 140% (Figure 5).

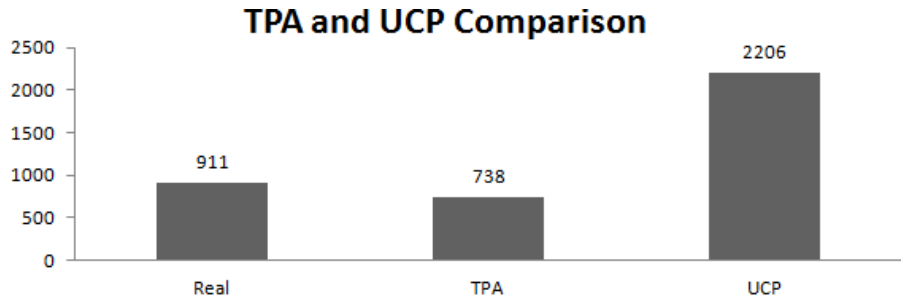

Figure 5 Comparison between real work, TPA and UCP estimation.

Compared to the UCP, the TPA had better results (Figure 5). This might be explained by the fact that the TPA considers more factors than the UCP. Furthermore, the TPA steps to estimate are better explained than the UCP and much less subjective (Van Veenendaal 1999). The TPA was also used as the basis of the built neural network.

At last, we compared how close to reality are the TPA and Neural Networks. For Neural Network estimation, we calculated estimates before and after re-training the network with real data. The re-training is supposed to output estimates closer to reality, as it is uses actual data from a specific enterprise, in order to train the network.

Comparing total work time for each method, we can see that the neural network, after re-training, outputs estimates closer to reality. The general error dropped from 19% on the TPA to 4.85% on neural network after re-training. The error in individual use cases stills bigger than expected.

When analyzing individual use cases we can see that eight of them had results closer to the real work spent after re-training with historical data. This demonstrates that re-training the network leads to significantly better results.

We can also observe that even before re-training the neural network, its results are very close to TPA's results. It means that training the network with an efficient method, than re-training it with historical data from real project is a procedure that works and gives better results.

Table 4 Estimation results for TPA and Neural Networks (NN) before and after re-training with historical data. On each follow the difference to the real work spent on each use case.

UC	Real	TPA	% Error	NN	% Error	NN	% Error
UC 1	81,82	64,26	-21%	88,88	9%	97,59	19%
UC 2	144,48	199,82	38%	200,2	39%	166,99	16%
UC 3	46,29	22,25	-52%	23,28	-50%	41,18	-11%
UC 4	74,04	51,57	-30%	57,09	-23%	73,15	-1%
UC 5	67,94	15,13	-78%	35,26	-48%	57,39	-16%
UC 6	128,75	114,65	-11%	123,79	-4%	121	-6%

UC 7	110,93	36,27	-67%	57,21	-48%	73,74	-34%
UC 8	87,17	112,46	29%	120,3	38%	117,97	35%
UC 9	105,75	99,81	-6%	77,02	-27%	86,41	-18%
UC 10	64,7	18,25	-72%	17,04	-74%	32,26	-50%
TOTAL	**911,87**	**734,47**	**-19%**	**800,07**	**-12%**	**867,68**	**-5%**

Conclusions

To make this paper possible we developed two different web applications to calculate the test effort estimation using the TPA and the UCP. Comparing those methods we concluded that the TPA resulted in an error of 19%, while the UCP had 142% for the use case set used. A third command line application was developed to estimate using neural networks trained with the TPA method.

Applying those methods on a real testing environment showed that they can be very effective and result in accurate effort estimation. After using neural networks for estimation, they showed to be a promising approach for work estimation, and become even better when re-trained with historical data from past projects.

Although we had good results using neural networks, the error for each use case is still large, showing that they might not be adequate, when estimating for a single use case. This was somewhat expected, because none of the methods consider the individual productivity of the professional responsible for the use case testing. We consider such extension as a good opportunity for future work.

References

- B., Kröse, and Patrick van der Smagt. *An Introduction to Neural Networks.* University of Amsterdam, 1996.
- Banerjee, Gautam. *Use Case Points: an estimation approach.* Unpublished, 2001.
- Belgamo, Anderson, and Sandra Fabbri. "Um estudo sobre a influência da sistematização da construção de modelos de casos de uso na contagem dos pontos de caso de uso." *III Simpósio Brasileiro de Qualidade de Software.* Brasília, 2004.
- Black, Rex. *Advanced Software Testing-Vol. 2: Guide to the Istqb Advanced Certification as an Advanced Test Manager.* O'Reilly, 2012.
- Cockburn, Alistair. *Writing effective use cases (Vol. 1).* Addison-Wesley, 2001.

- Cybenko, G. "Approximation by superpositions of a sigmoidal function." *Mathematics of Control, Signals, and Systems (MCSS)* 2, no. 4 (1989): 303-314.
- de Almeida, Érica Regina Campos, Bruno Teixeira de Abreu, and Regina Moraes. "Avaliação de um Método para Estimativa de Esforço para Testes baseado em Casos de Uso." *SBQS Simpósio Brasileira de Qualidade de Software da SBC.* Florianópolis, 2008. 331-338.
- de Almeida, Érika Regina Campos, Bruno Teixeira de Abreu, and Regina Moraes. "An alternative approach to test effort estimation based on use cases." *Software Testing Verification and Validation, 2009. ICST'09. International Conference on.* IEEE, 2009. 279-288.
- de Barcelos Tronto, Iris Fabiana, José Demísio Simões da Silva, and Nilson Sant'Anna. "An investigation of artificial neural networks based prediction systems in software project management." *Journal of Systems and Software*, 2008: 356--367.
- Freeman, James A., and David M. Skapura. *Neural networks: algorithms, applications, and programming techniques.* Addison-Wesley, 1991.
- Haykin, Simon. *Neural networks: a comprehensive foundation.* Prentice Hall PTR, 1994.
- Hornik, Kurt, Maxwell Stinchcombe, and Hallbert White. "Multilayer feedforward networks are universal approximators." *Neural Networks* 2, no. 5 (1989): 359-366.
- Lopes, Fernanda Amorim and Nelson, Maria Augusta Vieira. "Análise das Técnicas de Estimativas de Esforço para o Processo de Teste de Software." *III Encontro Brasileiro de Testes de Software.* Recife, 2008.
- Ludemir, TB, AP Braga, and A Carvalho. *Redes Neurais Artificiais: Teoria e Aplicações.* Livros Técnicos e Científicos Editora, 2000.
- McCulloch, WarrenS, and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity." *The bulletin of mathematical biophysics*, 1943: 115-113.
- Nageswaran, Suresh. "Test effort estimation using use case points." *Quality Week.* 2001. 1-6.
- Pádua, Wilson. *Engenharia de Software.* Rio de Janeiro: LCT, 2009.

- Peña, Anié Bermudez, José Alejandro Lugo García, Pedro Yobanis Piñero Pérez, and Iliana Pérez Pupo. "Optimización de reglas borrosas parael apoyo a la toma de decisiones en la Dirección Integrada de Proyectos." 2013.
- Rios, Emerson, and Trayahú Moreira. *Teste de Software.* Rio de Janeiro: Alta Books, 2006.
- Russell, Stuart J., and Peter Norvig. *Artificial Intelligence: A Modern Approach.* 2003.
- Sharma, Ashish and Kushwaha, Dharmender Singh. "Applying requirement based complexity for the estimation of software development and testing effort." *ACM SIGSOFT Software Engineering Notes*, 2012: 1-11.
- Smith, John. "The estimation of effort based on use cases." *Rational Software, white paper.* 1999.
- Sommerville, Ian and Melnikoff, Selma Shin Shimizu and Arakaki, Reginaldo and de Andrade Barbosa, Edilson. *Engenharia de software.* Addison Wesley, 2003.
- T.J., Rivlin. *An Introduction to the Approximation of Functions.* Dover Publications, 1981.
- Tang, Huajin, Tan Kay Chen, and Zhang Yi. *Neural Networks: Computational Models and Applications.* Springer, 2007.
- Van Veenendaal, EPWM and Dekkers, Ton. "Test Point Analysis: A Method for Test Estimation." *Project Control For Software Quality: Proceedings Of The Combined 10th European Software Control And Metrics Conference And The 2nd SCOPE Conference On Software Product Evaluation.* Herstmonceux Castle, United Kingdom: Shaker Publishing B.V., 1999. 47-59.
- Werbos, Paul. "Beyond regression: New tools for prediction and analysis in the behavioral sciences." 1974.
- Zhu, Xiaochun and Zhou, Bo and Hou, Li and Chen, Junbo and Chen, Lu. "An experience-based approach for test execution effort estimation." *Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for.* IEEE, 2008. 1193--1198.