Project Estimation with NDT

J. Armario, J. J. Gutiérrez, M. Alba, J. A. García-García, J. Vitorio and M. J. Escalona
IWT2 Research Group, University of Seville, Seville, Spain

Keywords: Software Project Estimation, Web Engineering.

Abstract: Software Project Estimation is one of the most critical and complex task for a Project manager. Several techniques, tools and mechanisms were proposed in the literature. However, these solutions are sometimes difficult and expensive to be applied and too frequently, the final estimation is made according to the manager experience. In this paper we present a preliminary approach based on the Use Case Points technique, which is adapted for the Model-Driven environment of NDT. This technique is automatically applied, thanks to the metamodels definition, and it is presented in a tool named NDT-Counter. Additionally, the paper presents an initial empirical evaluation of the results.

1 INTRODUCTION

To manage Software projects implies developing an initial phase and a plan as well as tasks or activities. At this stage of the project life cycle, experts must meet and weight up the effort, work, and necessary hardware and software resources as well as set the cost and time required to execute the requested work. In the planning project phase, the different tasks that make up the project, the deadline to be carried out and the people to develop it must be detailed. After analysing the different variables through the estimation process, the cost and time to complete the project will be determined. Estimation is important at this point because by analysing and studying the result obtained, we can assess if the project development is profitable or, by the contrary, the terms set by the cost-benefit ratio are negative.
Considering that this study and its subsequent result are highly demanded and specially relevant, its crucial character to state whether a project should be faced up, together with the high applicability of Navigational Development Techniques (NDT) (Escalona and Koch, 2008) methodology, it is necessary to extend this methodology and include it within the suite of configurable tools that execute the estimation project automatically.

NDT is a Web methodology mainly focussed on the Requirement Engineering phase leaded by objectives related to capture, definition and verification of requirements and their incorporation into the software development life cycle, giving it the importance it deserves. NDT is developed within the Model-Driven paradigm environment. For that purpose, this paper shows the Use Cases Points technique (Karner, 1993) which operates to obtain the cost and time study. In addition, we will demonstrate how, thanks to the NDT formal nature, we can extend the application of this method to obtain and analyse the results in an automated way.

This paper is structured as follows: Section 2 analyses the most widely used estimation techniques. Section 3 presents NDT methodology and Section 4 shows the suggested estimation technique for NDT introduced in the previous section, as well as the extension for this methodology and the later tool development for this phase of the software project. In Section 5, we execute simulations as well as compare and analyse the results, the fixed cost of the tool and the real cost of the project by means of the tool developed with the aim of estimating the project data which have already been completed. To conclude, Section 6 offers final conclusions and ongoing works.

2 RELATED WORK

Many Cost Estimation models have been proposed in the last 40 years. They can be classified in two main groups: algorithm-based models and non algorithm-based models.
Although this paper focuses on algorithm-based models Cost Estimation methods, some non based-

Armario J., Gutierrez J., Alba M., García García J., Vitorio J. and Escalona M..
Project Estimation with NDT.
DOI: 10.5220/0004022101200126
In *Proceedings of the 7th International Conference on Software Paradigm Trends* (ICSOFT-2012), pages 120-126
ISBN: 978-989-8565-19-8

algorithm models must be mentioned, for example:

- Estimation by analogy (Shepperd, Schofield, and Kitchenham, 1996). This method requires one or more completed projects similar to the new project, and derives estimation through reasoning by analogy through the actual costs of previous projects.
- Expert judgement method (Jorgensen, 2005). This method involves working in liaison with a software cost estimation expert or group of experts to use their experience and understanding of the proposed project to reach an estimate cost. This method can be used together with the Delphi technique (Lilja, Laakso and Palomki, 2011) which allows improving and systematising the consulted experts' opinion.
- Bottom-up. In this method, each software system component is separately assessed and results are added in order to produce an estimate **of** the overall system. The requirement for this approach is that an initial design must be in place to indicate how the system is decomposed into different elements.
- Top-down: This method is the opposite of the bottom-up method. An overall cost estimate for the system is derived from global properties, using either algorithmic or non algorithmic methods. The total cost can then be split up into different components. This approach is more suitable for cost estimation at the early stage.

In addition, there are many algorithmic models to estimate the project cost. These methods are mathematical-based models that produce cost estimate as a function with a number of variables, which are considered to be the major cost factors. To improve the accuracy of algorithmic models, it is necessary to adjust or calibrate the model to local circumstances. Despite calibration, accuracy can be quite mixed. Some of the most referenced algorithmic models are:

- COCOMO (Constructive Cost Model). These models were proposed by Boehm (Boehm, 1981) in 1981 and later reviewed in the 90's, when the development techniques drastically changed. In 2000, the second version of COCOMOs was published and was called COCOMO II (Boehm, Abts, Winsor Brown, Chulani, Clark, Horowitz, Madachy, Reifer and Steece, 2000). COCOMO uses a basic regression formula with parameters derived from both, historical data of the project and its current characteristics. This model consists in a hierarchy of three increasingly detailed and accurate models.

The first model, Basic COCOMO computes software development effort as a function based on code-size given in thousands of lines of code, (KLOC). This model is suitable to get an early quick rough order of estimates of software costs, but its is accuracy-limited due to its lack of factors to account for differences in project attributes (Cost Drivers, for instance, provide differences in hardware constraints, personnel quality and experience or usage of modern tools and techniques, among others). The second model, Intermediate COCOMO computes software development effort as a function based both on the code-size and a set of Cost Drivers that include subjective assessment of products, hardwares, personnel and project attributes. The third model, Detailed COCOMO incorporates all characteristics of the intermediate version plus an assessment of the influence of Cost Drivers on each individual phase of the project (Analysis, Design, etc.) in the software engineering process.

- The Putnam model (Putnam, 1978). This model represents an empirical software effort estimation model. Putnam focuses his model on Rayleigh's manpower distribution and his finding on analysing many completed projects. Putnam's approach is incorporated into a commercially available cost estimation system called SLIM.
- Bailey-Basili metamodel (Bailey and Basili, 1981). Authors aimed to derive a methodology, thus, they assumed that the coefficients in any effort equation would be highly dependent on the environment and personnel at a particular installation, and that coefficients derived from a local database would lead to a much more accurate model. Their metamodels deal with a rigorous statistical analysis of 18 relevant projects developed at the NASA Goddard Space Flight Center so as to determine the equations that measure the effort and Cost Drivers. This basic methodology is neither important for the specific effort equation nor the particular Cost Drivers. It is important because it provides a methodology used by individual organizations to construct their own models that are tuned to their particular environment.
- Function Points (Albrecht and Gaffney, 1983). This is a functionality-based measure of the program. The Functional User Requirements of the software are identified and the total number of function points depending on each one is categorized into one of these five types: outputs, inquiries, inputs, internal files and external interfaces. Once the function is identified and

categorized into a type, it is then assessed for complexity and assigned a function point number.

- Bayesian Networks (Mendes 2008). It is a structure probability in this case used for the estimation of effort, so the nodes represents the relevant factors that have associated a table of probability, and the arches that connect these nodes, the relationship between the various variables quantified way probabilistic. The resulting effort is obtained through a combination of the probabilistic results of nodes parents of this.
- Use Cases Points technique (Karner, 1993). Use Case modelling is an accepted and widespread technique to capture the business processes and requirements of a software application. Since they provide the functional scope of the application, analysing their contents provide valuable insight on the effort and size needed to design and implement the application. Use Case Points (UCP) is an estimation method that provides the ability to estimate size and effort of an application from its use cases. Section 4 will describe how this tecnique is adapted to the NDT methodology.

Finally, we would like to mention that nowadays, the most traditional models to estimate projects cost are being reviewed with new mathematical models. One example is COCOMO, a model based on artificial neural networks (Attarzadeh and Siew Hock Ow, 2010).

3 AN OVERVIEW OF NDT

Navigational Development Technique (NDT) (Escalona et-al, 2008) is a Model-Driven Web methodology that was initially defined to deal with Web development requirements. NDT starts with a goal-oriented phase of requirements and establishes a set of transformations to generate analysis models.
NDT has evolved in the last years and offers a complete support for the whole life cycle. Nowadays, it covers viability study, requirements treatment, analysis, design, construction or implementation as well as maintenance and test phases, such as software development phases. Additionally, it supports a set of processes to bear out project management and quality assurance and sustain different life cycles, for instance, sequential, iterative and agile processes.
As an advantage, NDT can be applied in the enterprise environment. Today, many companies in Spain work with NDT and the associated tools for software development. This is possible due to the fact that NDT is completely supported by a set of free tools, grouped in NDT-Suite (NDT-Suite 2012). This suite enables the definition and use of every process and task supported by NDT and offers relevant resources for quality assurance, management and metrics with the aim of developing software projects. NDT is based on the Model-Driven paradigm. It selects a set of metamodels for each development phase (requirements, analysis, design, implementation, construction, test and maintenance) in order to support each artefact defined in the methodology. All concepts in every phase of NDT are metamodeled and formally related to other concepts by means of associations and/or OCL constraints (OMG-OCL 2012). Besides, NDT proposes a set of QVT Transformations (Query/View/Transformation) (OMG-QVT 2012) among each metamodel in every phase, that may enable to get one phase results from the previous one. Nevertheless, transferring this idea to the enterprise environment is not possible. Companies do not actually use metamodels, transformations and other elements, thus technology seems too abstract for them. After assessing different possibilities, some UML-profiles were developed for each NDT metamodel. These UML-profiles were defined in a UML-based tool named Enterprise Architect (Enterprise Architect, 2011). Then, the first tool for NDT-Suite, NDT-Profile, was developed. The remaining NDT-Suite tools are based on this profile and offer a range of different uses when applying NDT, which can be downloaded in www.iwt2.org.

As it can be concluded, in the last years, NDT has become a complete approach offering high support for software project development by exploiting the power of the Model-Driven paradigm.

However, software estimation meant a gap in the approach. For this reason, a solution consisting in providing a new tool named NDT-Counter has been developed. It is presented in detail in the next section.

4 A SOLUTION FOR NDT

Despite NDT supports the project management, its tools, described in the previous section, do not offer special support for the project estimation. As NDT is based on an Object-Oriented environment and Use Cases is the selected technique to describe Functional requirements, Use Cases Points is selected as a first alternative for project estimation

support. This section presents an overview of this technique a well as its application in NDT.

4.1 Use Cases Points

Use Cases Points is a technique that allows us to estimate the effort hour/person that must be carried out to develop a software tool with specified features. The instructions are as follows:

STEP 1: Analyse the requirement to calculate the Unadjusted Use Case Points (UUCP). It covers three steps:

a. Structure every interaction between actor and Use Cases according to their complexity and assign them a weight.
b. Calculate the complexity of every Use Case according to the number of steps or transactions.
c. Calculate Unadjusted Use Case Points according to the previous data.

STEP 2: Study the Technical Complexity Factors (TCF) and the Environment Factors (EF) and find the factors needed to balance Unadjusted Use Case Points (UUCP). This step is divided in **three** phases:

Table 1: Technical Complexity Factors.

Factor	Description
T1	Distributed System
T2	Response adjectives
T3	End-user efficiency
T4	Complex processing
T5	Reusable code
T6	Easy to install
T7	Easy to use
T8	Portable
T9	Easy to change
T10	Concurrent
T11	Security features
T12	Access for third parties
T13	Special training required

a. Calculate the Technical Complexity Factors (TCF). Every defined factor is given a value related to its influence on the Project. Technical Complexity Factors traditionally used in this technique are showed in Table 1. Once all the technical factors have been assigned, it is necessary to calculate the Complexity Coefficient.

b. Calculate Environment Factors (EF). Environment Factors commonly used in this technique are showed in Table 2.

Despite having into account the technical factor for the adjustment of UUCP, the Environment Factors must be analysed. For that purpose, every factor defined is given a value according to its degree of influence on the Project.

Table 2: Environment Factors.

Factor	Description
E1	Familiar with the development process
E2	Application experience
E3	Object Oriented Experience
E4	Lead analyst capability
E5	Motivation
E6	Stable Requirements
E7	Part-time workers
E8	Difficult programming language

Once all the factors are given the influence value, it is necessary to calculate the Complexity Coefficient.

c. Calculate the Use Case Points (UCP) using the previous data.

We should consider that by calculating this expression we obtain a size of the estimation, but not of effort.

STEP 3. Adjust (UCP), and later an Effort Estimation must be obtained (hour/person).

It should be pointed out that the value of effort calculated does not cover all life cycle phases, but only refers to hour/people invested in developing the specified functionality of Use Cases at the codification phase. Generally, this phase represents 40% of the total effort of a Project.

4.2 Use Cases Points in NDT

If we intended to offer an automatic support, the integration of this estimation technique would require an extension of both, the methodology and structure. Thus, the initial requirements metamodel of NDT has to be analysed and extended to support and manage aspects required by the Use Cases Points. In fact, aspects supported in STEP 1, Actors, Use Cases and Complexity, were included in the original metamodel, so no changes were required.

However, we need special support for the aspect included in STEP 2. Thus, the metamodel was enriched with Technical Complexity Factors and the Environment Factors. This extension was implemented and included in NDT-Profile and a suitable interface was developed in order to make easier the application of techniques. The following technique begins with obtaining Actors and Use Cases that take part in the software project being currently studied. They are assigned a complexity from 1 to n. In the case of Actors, NDT considers that the complexity of an Actor depends on the number of use cases in which it is involved. The Use Cases Complexity is determined by the number of sub-tasks in which it is involved and it is assigned a number from 1 to n. The more sub-tasks a Use Case has, the more complex it is. The adjusted Use Cases

are calculated by means of this data. Once Actors and Use Cases Complexities are defined, non adjusted Use Cases Points are obtained evaluating, technical and environment factors of the project. For that purpose, as determined elements have influence on each project, NDT assigns them a default complexity, so if these values are aimed to be changed, the user will adapt them to his/her needs. This complexity represents the relevance of a factor within the project. The higher the number associated to the complexity is, the higher influence this factor will have on the project being estimated. After setting these elements, the process continues by calculating the effort. The estimation obtained will be given in hour/ person. These steps are automatically performed. Considering the definition of Actors and Use Cases in NDT, and taking advantage of the methodology integration with the Enterprise Architect tool, we conclude that a software tool can perform this automatic and feasible process. For this reason, it is decided to implement a software tool which can cope with this estimation technique.

4.3 NDT-Counter

NDT-Counter is a desktop application integrated into the suite of the methodology with the same name, developed by the research group IWT2 at the University of Seville. This application helps us apply the Use Case Points previously explained.

This application provides a detailed hour/person-cost report from/in the system we are developing. These estimates are related to the Implementation phase and translated into the economic cost of the project.

NDT-Counter has a number of outstanding features, for example, multiple language support or the possibility of exporting the results obtained by offering reports. At the same time, due to the needs showed in the previous section, this tool offers the possibility of analysing software projects developed with agile and not agile methodologies and configuring any parameter involved in our project.

The NDT-Counter interface is simple and intuitive. The effort estimation process for this tool is showed in the following figures and explained below.

In the main screen, we start by writing our project estimation in P*roject name* field. This will be the name of the report obtained when the estimation may be carried out. Then, the *File* selection button allows selecting the desired Enterprise Architect file to execute the estimation. This Case tool is used in the NDT Suite to build software systems. A baseline for Enterprise Architect is adopted to make the tools of the suite automate the phases of software development proposed by NDT. In this file, we get Actors and Use Cases involved in the project which are necessary to estimate effort in NDT-Counter.

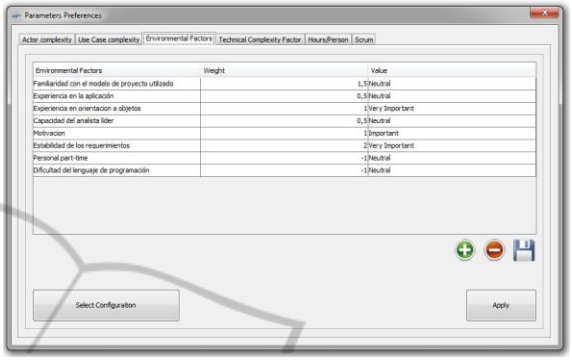

Figure 1: NDT-Counter main interface.

Finally, the estimation process starts by clicking on *Start estimation* button. The effort estimation of the selected project will be achieved when the chosen parameters preferences are set up. Initially, this button will be unavailable until the parameters preferences have been correctly set up. When clicking on *Parameters preferences* button, the Preferences window will open and the values of the parameters involved in the estimation process can be chosen. Next, the parameters preferences screen is showed and the important details are explained. The previous screen shows how the parameters involved in our software project can be configured. After loading the selected file, Actors *(Actor complexity tab)*, and Use Cases *(Use case complexity tab)*, it is possible to configure several parameters. Different options must be selected: using default values, load a configuration previously saved or configure the factors and their complexities manually. In the latter case, and in order to adjust these factors as much as possible to our project specifications, the user can add and/or remove Technical Complexity Factors *(Technical Complexity*

Figure 2: Parameters preferences interface.

Factors tab) and Environment Factors, *(Environment Factors tab)*, and modify the weights associated with them, as needed.

A very important aspect to note is the *Scrum tab*. This tab supports agile methodologies, so that we can select to "sprint", to run the estimation process and get an effort result for each sprint. Once the parameters to be estimated related to our project are defined and configured, the estimation process begins. After applying the Use Cases Points technique, with their steps and calculations, the report screen with the estimate results appears.

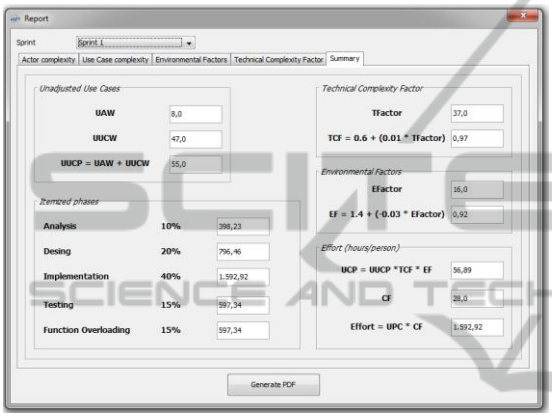

Figure 3: Estimation, result reading and report.

The screen above shows the estimate for our software project. The tabs offers the factors defined and involved in the project to obtain the final result together with their configuration. The *Summary* tab shows all the hour/person information included in the project. We obtain partial results of the effort required to complete it at each stage of the life cycle of a software project. It should be remarked that the result obtained by the Use Case Points technique corresponds to the Implementation phase, and the hour/person references of the other phases have been calculated by adjustment. Finally, to facilitate the presentation and portability of results, a PDF file will be generated in order to record the final estimation result, just by clicking on the *Generate PDF* button.

5 EMPIRICAL RESULTS

After studying in depth some projects involving NDT, it must be checked that the result of the estimation given by NTD-Counter tool is similar to the final project length. Even though we must assume a percentage of risk when we undertake an enterprise like the Software Project Development, these risk elements may affect the necessary effort by delaying the project and increasing its cost.

We should remember that estimation is applied to obtain the necessary effort at the Implementation phase. The total effort to develop a software project is obtained when applying a generic effort distribution; the Analysis phase takes up a 10%, the Design phase a 20%, the Implementation phase a 40%, the Testing phase a 15%, and overload and other activities a 15%. Figure 4 shows some projects where we face up the estimate data with NDT-Counter and their total duration, once finished. This confrontation of results has been done when the projects finished, to see the power of the tool and the similarity of the result of effort between our application and the reality. The graphic above represents how the three analysed projects had a total duration similar to that estimated with NDT-Counter. The projects are diverse. The first project (project 35) is a web application where we can highlight the inclusion of a search engine for documents with numerous search formats, plain-text, date, advanced search, parameterized... The second project (project 56) concerns SMITA system that allows users to locate nearby activities and sights to visit. The power of this system lies in its access via mobile phone, allowing users to have the information in real time. The third project (project 72) discussed is reference to @rchivA system, which has been developed for the Junta de Andalucía. This sets up a single information system for files attached to the Administration of the Government of Andalusia in the File System as well as court records. This shows the basic tool for the management of archived not only paper but also in digital format as a fundamental part of eGovernment model of the Junta de Andalucía. In this regard, we have selected a configuration as close as possible to the features studied at that moment by the responsible for these projects. Once these factors are configured, NDT-Counter returns an estimation result given in hour/person. To present data, we have transformed these hours in days and have extrapolated them according to the generic distribution, previously showed, so as to obtain the total length of a project. We note that in the first two projects, NDT-Counter has given an estimate, measured in days, lower than the total project duration. This is due to many random factors that may influence the duration of any types of projects and delay the deadline for a project completion.

Nevertheless, in the last projects studied, we observed that the tool estimates a longer period of less significance than the final length the project

finally had. According to the studies analyzed, we can conclude that, when identifying the factors that are involved in the project and assigning complexities as closest to reality as possible, the developed tool to implement NDT methodology obtained similar estimation results to those occurred in real life. Our idea consists in applying this estimation to the projects at the Requirements phase, so the viability study will be easy and directly applied by a software tool in an automatic way.

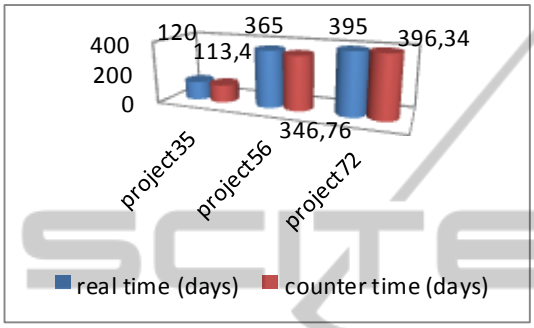

Figure 4: NDT-Counter estimation vs real time.

6 CONCLUSIONS AND FUTURE WORK

Software Cost Estimation in software development is a very relevant phase to manage and decide whether a project should be undertook in terms of profitability. Several techniques have been presented in the article, but due to NDT methodology nature, its Requirements phase analysis and definition of Actors and Use Cases, we decided to extend the methodology to carry out estimates by means of the Use Cases Points technique. For this reason, and the fact that NDT is well integrated within the Enterprise Architect tool and this technique could be automatically applied, we developed the software tool NDT-Counter. NDT-Counter allows you to extract the necessary information from the files containing the requirements in a software project as well as develop the estimation technique automatically. The set of factors involved in a project can be modified by the user, so it can be adapted to one's needs. This article shows how NDT-Counter has been applied for the estimation of some completed projects and how this estimation result is very close to the final efforts of the project. Up to date, estimates were calculated when projects had been completed. From now on, as this tool allows automation, we will calculate effort before executing the project.

ACKNOWLEDGEMENTS

This research has been supported by the Tempros project (TIN2010-20057-C03-02) and Red CaSA (TIN 2010-12312-E) of the Ministerio de Ciencia e Innovación, Spain, and NDTQ-Framework project of the Junta de Andalucía, Spain (TIC-5789).

REFERENCES

Karner, Gustav. "Resource Estimation for Objectory Projects" *Objective Systems SF AB*, 1993.

Escalona, M.J., Aragón, G. "NDT. A Model-Driven approach for Web requirements". *IEEE Transaction on Software Engineering*, 34(3), pp. 370-390, 2008.

Shepperd, M.; Schofield, C.; Kitchenham, B. *"Effort estimation using analogy"*. Dept. of Comput., Bournemouth Univ. Software Engineering, 1996.

Jorgensen, M."Practical guidelines for expert-judgment-based software effort estimation". Oslo Univ., Norway. *Software*, pp. 57 - 63, 2005.

Lilja, K. K.; Laakso, K.; Palomki, J. *"Using the Delphi method"*. Technology Management in the Energy Smart World (PICMET), 2011 Proceedings of PICMET '11. Tampere Univ. of Technol., Pori, Finland, 2011

Boehm, B. W. "*Software engineering economics",* Englewood Cliffs, NJ: Prentice-Hall, 1981.

Boehm, B. W., Abts C., Winsor Brown A., Chulani S., Clark B. K., Horowitz E., Madachy R., Reifer D. J., and Steece B.. "*Software Cost Estimation with COCOMO II"*. Englewood Cliffs, NJ: Prentice-Hall, 2000.

Object Constraint Language. OCL. Specification Beta 2.3. http://www.omg.org/spec/OCL/2.3 /Beta2. 03/2011. Last visited 04/2012.

Object Management Group. Query View Transformation Specification 1.0. 2010. http://www.omg.org. Last visited 10/2011.

Enterprise Architect. www.sparxsystems.com. Last visited 04/2012.

Putnam, L. H., "A general empirical solution to the macrosoftware sizing and estimating problem", *IEEE Transaction on Software Engineering*, 4(4), pp 345-361, 1978.

Albrecht, A. J.; Gaffney, J. E. "Software function, source lines of codes, and development effort prediction: a software science validation", *IEEE TSE*. SE-9, pp.639-648, 1983.

Emilia Mendes: "The Use of Bayesian Networks for Web Effort Estimation: Further Investigation". *ICWE* 2008.

Bailey J. W., Basili V. R., "A meta-model for software development resource expenditures", *Proceedings of the 5th ICSE*, pp 107-116, March 09-12, 1981, San Diego, California, United States.

Attarzadeh, I.; Siew Hock Ow. "Proposing a new software cost estimation model based on artificial neural networks". (*ICCET*), Vol. 3, pp. 487-4.