Methods for cost estimation in software project management

C V Briciu\(^1\), I Filip\(^1\) and I I Indries\(^2\)

\(^1\)University Politehnica Timisoara, Automation and Applied Informatics Department, Vasile Parvan str., no. 2, 300223 Timisoara, Romania
\(^2\)University Politehnica Timisoara, Computer Science Department, Vasile Parvan str., no. 2, 300223 Timisoara, Romania

E-mail: briciucatalin@yahoo.com

Abstract. The speed in which the processes used in software development field have changed makes it very difficult the task of forecasting the overall costs for a software project. By many researchers, this task has been considered unachievable, but there is a group of scientist for which this task can be solved using the already known mathematical methods (e.g. multiple linear regressions) and the new techniques as genetic programming and neural networks. The paper presents a solution for building a model for the cost estimation models in the software project management using genetic algorithms starting from the PROMISE datasets related COCOMO 81 model. In the first part of the paper, a summary of the major achievements in the research area of finding a model for estimating the overall project costs is presented together with the description of the existing software development process models. In the last part, a basic proposal of a mathematical model of a genetic programming is proposed including here the description of the chosen fitness function and chromosome representation. The perspective of model described it linked with the current reality of the software development considering as basis the software product life cycle and the current challenges and innovations in the software development area. Based on the author's experiences and the analysis of the existing models and product lifecycle it was concluded that estimation models should be adapted with the new technologies and emerging systems and they depend largely by the chosen software development method.

1. Introduction

Nowadays, the main concern of the producers (e.g. carmakers) is reducing the costs of production for different products but in the same time the increasing of the quality of them. At first seen, this sounds not to be very plausible since in the real life it is known the fact that with less money there cannot be bought products with a better quality. What if the producers find the method to do this? In mathematical terms, this can be described as an optimization problem of the quality triangle (Figure 1) from the costs point of view.

As a mathematical description, this can be considered as \( \text{max}(F(u)) \) where \( u \) can be considered the entry considered for all disciplines of the project (software, hardware, mechanical, financial, human resources). In this relation, \( F \) is the objective function and in the same manner as \( u \) is a vertex quantity. A detailed approach above this function can be found later in the article.

Even at the start of the article, all branches of the product development has been taken into consideration, in the next parts of the paper the focus will be set to software development, even maybe there are products in which the software component is the less time/money consuming part, but in
these days, a higher degree of products consists from software parts. (e.g. websites, smart houses, security services). Starting from the beginning of this century, due to higher demands on the software components from requirements point of view, the costs had been increased also and it comes to the situation where most of the projects were been underestimated from time and costs point of view. This fact changed the mindset of the software producers to have a good process to determine the costs and the finishing time even before the quotation time. Also, it is known the fact that defining a development process is not a task to be defined and finished in a half a year or one year. One issue that appears in this situation is related to the startup companies that do not have very experienced people. For example, an estimation of the project from the costs points of view in automotive field can be not a so hard operation if the project team has at least two or three experienced people. They can estimate, of course, with some errors the project costs and the project duration based on their previous experience. But what happens if this experienced people are missing from the quotation phase of the project. For a small company, this can be maybe instead of the "start" phase, the "stop" phase. For this, each company defines its process development rules for achieving the desired level of quality. But this process development rules during the product lifecycle cannot guarantee if they are fulfilled then the projects are successfully. It is estimated that 1/3 of a projects are considered unsuccessfully [1].

Figure 1. Quality triangle

Starting from the beginning of the 1980', software managers and quality assurance employers tried to develop methods to forecasting the evolution of the projects though resolutions regarding products life cycle, software development methods and even methods for estimating the projects costs. They were intuitive and very easy to understand, but in our times it seems that they don't have applicability on the current projects. In the second part of the paper, this will be presented in detail. Also, a tendency that can be seen currently is to replace the old development software processes (they are still in use) with new ones that are more closed to the developers' daily work. Also, it is known the fact that a single process cannot be applied in all fields of software activity because in one sector activity the quality is first (maybe in terms of safety and security), but in others functionality is in top (web services, PC desktop application). Also, an old C++ application that have a graphical user interface it can be done now very fast because the IDE (Integrated Development Environment) it has now implemented a drag-and-drop functionality. This was also anticipated by the Philip G. Amour in [2], which considers the software process as "a race for knowledge and new": "Like a climber planning a route over glaciers and up a mountain face, the destination is clear, but getting a safe route to the mountain and back down again takes a mixture of careful planning and an adaptive approach to events. The climbers may find impassable chasms, dangerous overhangs or unpredictable changes in weather. The plan will probably change".

This research paper is the first paper from a research study of three years where the final scope is the respond to the following questions and acts as an introduction for understanding the input parameters of the research:

a) Is there a software development process applicable to all software disciplines in which the obtained results are the same or near the same? If yes, is feasible the development of a new process model that can be generalized to all software process disciplines?

b) Are the software estimation models available in the literature accurate to all projects types? If yes, what is the optimum existing model for cost estimation, otherwise, can be the cost estimation
model generalized for all projects? In this point, it is necessary to be proven that the existing models estimate truly also real projects and not only the projects in which they were developed.

c) Is feasible and realizable an estimation model where the human component to be considered not as an abstract resource, but as a state variable of the mathematical model?

Before continuing, there are necessary some definitions for the main terms and topics described further in the paper.

The second section of the paper presents the existing most known software development processes. For a better understanding, they will be presented in a comparative way, in which the applicability of them in the current times will be pointedly. In the third section of the article, some of the most important and know cost estimations models are briefly described. The fourth section presents a short description of the existing models for estimations. Next section will be reserved for the presentation of the idea in which the study will be continued for achieving the answers to those three questions.

2. Software Development Process Models

Before introducing the definition for software development model, it is necessary to start with the definition of the Software Development Life Cycle (SDLC). SDLC can be considered as a set of techniques and methods that describes the phases needed for the development of the product from order point of view and deliverables. The SDLC suppose that phase X supposed to be performed after Y phase and as inputs for the phase Y (outputs from the phase X) with the resulting deliverables. Some examples can be considered the requirements are translated into design at certain level, design can be considered as input for the code, and the code is an input for the verification and validation phase. The main phases of the SDLC are:

a) Quotation and Concept Refinement: this phase is started with the innovation & roadmap where an opportunity is identified. After the identification of the opportunity there is a needed for the system concept development. As a deliverable, it should be considered that system boundaries are clearly defined, and the documents for cost benefits, risk management and assessment plans are defined.

b) Development: this phase is started with the planning of the activities that acts as a basic for acquiring the resources for achieving the refine concept. As first deliverables documents of this phase can be considered the project management development plan and all other planning documents. Then, this phase is continued with the requirements gathering and analysis whose deliverable is the Functional Requirements Documents (System Requirements Specification - SRS). Based on this document, the requirements are translated into design documents at system level (Software Architecture Document - SWAD) and then at component level (CSDD - Component Software Detailed Design). In this phase, the design is frozen and approved. The approved design is considered as input for the Implementation phase where coding, review protocol documents and all activities regarding the SW integration and SW validation documents are performed. After the implementation phase is end, the integration and validation test cases are performed for assuring the bug-free of the project. As inputs for this phase, it should be taken into consideration the System Requirements Specification when the system testing is performed and component Requirements Specification when the validation tests are done.

c) Industrialization - Production Ramp-up and the series production. This phase starts with the deploying of the application in the client side and describes tasks to operate and maintain information systems in a production environment. It is often use also as Post-Implementation phase means implementation of new features.

d) Disposition - after series, where the delivery obligation is ended and also describes the end-of-system activities.

According to Walt Scacchi [3] the software development life cycle is either a descriptive and prescriptive way to describe how the system is or should be developed. The descriptive component presents the way in which a previous software system or component was developed while the prescriptive model suggests the order of operations to be done for achieving the initial scope of the project. In facts, it acts as a guidelines or framework of activities that should be performed.
A software process model can be characterized as "networked sequence of activities, objects, transformations, and events that embody strategies for accomplishing software evolution" [3]. From the beginning of the software development, there was a tendency to learn from previous projects and to organize the lesson learnt into models that can be reused later as improvements. These sequences can be seen as non-linear subsystems which can be done and done until the final product achieves the desired level of quality. It is hard to define for each sequence the exact input parameters without omission of at least one of them due to the fact that maybe these sequences can be executed in parallel. Also, only based on the successful/unsuccessful of a sequence or a part of the sequences it cannot be guaranteed the success of the project. From qualitative point of view, these parameters can be defined with sufficient prevision, but from quantitative point of view, this operation can be often considered subjective depending by the person in charge with these tasks. It is important to know is the fact that planning of the testing sessions will depend by the chosen development process. If a process supposes the developing of the system in linear steps then the testing will be performed through the end of the project. If it is done in increments or repetitive phases then preparation of test cases scripts and specification will be started in the same time with the development of the executables. As a representation of the sequences as a system using system theory notation this can be seen in the picture from below (Figure 2).

Figure 2. Software product life cycle

The main known software process models are the following:

a) **Waterfall model**: this model is known for its linearity because a phase is not started until the previous phase is not ended successfully. The phases of the project are well-defined from input and outputs point of view. A good definition of phases also leads for project managers to an easily way to define the needed activities. What is interesting to follow here is the fact that testing phase is done near the end of the project. This cannot be accepted in today world because customer wants to see at least some prototypes or releases for continuing work. Also, due to complexity of the systems, many times the customers change the requirements during lifecycle and without executables it is hard for them to check if the client understood very well the requirements. Using this model for the software development this cannot be achieved and acts as a structural defect of the model. Also, this structural defect is the main argument of the objectors who said that this is not applicable anymore. This implies also a high risk of uncertainty because any defect from the design phase cannot be corrected. In fact, this method is supposed to be used for low complexity projects where the requirements are defined and clarified.

b) **V-cycle model**: it is one of the most used software process models used in the embedded systems industry and it seems to be feasible also for long and very complex projects. In the same way
as waterfall model, a phase cannot be started if the previous one is not finished. In practice, due to
elasticity of the model, some phases can be started in the same time. As an advantage of the model, it
is very simple to be understood and the verification and validation is done in the same time with the
corresponding phase. In fact, on both wings of the V-shape there is a correspondence between the
phases that can be seen in Figure 3.

![Figure 3. V-cycle model](image)

In the same manner as required for choosing the waterfall model, it can be used for the projects
where requirement are clearly and fixed and the customer has a higher degree of confidence in the
client. The phases meaning are described briefly in the [4].

c) **Incremental model**: using this type of software model, the application is developed in small
increments until the final application is ended from the requirement point of view. To have an example
we can consider a 3x3 panel which represents the final project and each delivery is similar with the
installation of one panel. For each increment, all phases of the SDLC are passed. As a comparison, we
can considered it a multi-waterfall method since for all increments all phases should be passed
/software is done step by step/. What is important to remember from here is the fact that after each
increment working software can be delivered to the customer. Also, this means that requirements
should not be know all from the beginning and the client can "respond" to the working software after
each build, but there is a disadvantage because in this way costs risks management introduces more
uncertainties. Comparative with the waterfall method, the costs can be higher.

d) **RAD model** - (Rapid Application Model) - is a simple model that allows complex systems to be
developed as independent systems. This requires very good knowledge of the project and of the
requirements from the systems engineers. "The generic" solution is the most used software paradigm
where components are developed as independent modules and then built together at the integration.
Nowadays, this model is very used since the AutoSAR suppose that software components for building
a software is not necessary to be developed by the same client. In facts, in [5], authors describe the
methodology for building such types of modules.

e) **Agile model** - The agile model starts from the Manifesto Agile, a reference of software
development methods. It is based on an approach based on development and deliveries in very small
increments. The main idea of the model is based on the continuous integration, continuous
improvement of the code and pair programming. What is different related to the models presented
before is the fact that requirements are classified in Story Cards and based on this the stories and team
velocities are calculated and the releases are planned. From development point of view, it is almost the
same with the incremental model because functionality is added after each release. Refactoring and
simple design are others paradigm that are intensively used during the development using this type of
model. Nowadays, this model seems to be the most used in the web-development and enterprises
areas, except the embedded area, but the tendency is to start the usage of it also in that area.
f) **Iterative model** - the iterative model supposed from beginning that requirements are not clarified and it will be clarified after each release based on the review on the current software. It acts like a genetic algorithm whose behavior improves after each population and at the end of the maximum generation of populations is gives the best solution.

g) **Spiral model** - the spiral model is different from the first four presented models because it is split in other phases where the known SDCL phases are performed. These four phases are: "Planning", "Risk assessment", "Engineering", "Evaluation". The deliverables for each phase are:
- Planning - SRS document
- Risk assessment and evaluation: risk evaluation and risk prevention countermeasures
- Engineering: the software builds executables and the test reports of validation test cases
- Evaluation: feedback from the customer and approval for the next spiral.

Due to evaluation from the client after each spiral round results that the spiral method is applicable for the projects where the risks should be taken into consideration and evaluated because not all requirements are clear from the beginning.

3. Software Development Process Models

A main component of the software models from the quantitative perspective is the cost of the project. Depending by this factor, a software development company can be considered as a good example from financial point of view or the opposite. Theoretically, the company tries to maximize the quality of the delivered software, but practically on most cases, the balance at the end of the project is considered as maximization function. Taking into consideration the quality triangle presented in the first part the costs represents a quality indicator for the final estimation of the project. The software estimation suppose to accurate forecasting the size of the software that should be developed, the time of the software until it can be released and project schedules. From the beginning of the software engineering, there were researches for finding a mathematical model of the cost estimation, but the rhythm of new programming languages and CASE tools did the researcher job very ineffective because the new programming languages were based on high level languages, therefore the number of ELOC (Effective Line Of Codes) decreased. In first part of this chapter a brief description of the empirical parametric estimation cost functions will be presented and in the second part new techniques that are applicable for the current development sandboxes (COCOMO II model, estimation through function points).

There are several methods to forecasting the costs of a project based on some product values measure that can be the number of code lines in the application or other measured values to quantify some characteristics of the software application.

a) **Empirical mathematical estimation models**: first model is a linear model that was developed at the beginning of the software development in a structured way and it has the mathematical formula: \( \text{Effort} = \text{LOCb} \), where effort is expressed in the man-month and the LOC is effective lines of code \([6]\). Regarding the constant, some researches quotes' that it should be above par, others states that should be under par. Continuing with the enumeration of the researched solutions, it was stated that a linear model cannot express entirely an exact model, so a non-linear approach was introduced according to the following: \( \text{Effort} = a + b\text{LOC} + c\text{LOC}^2 \)[7]. The research based on the datasets project found that the value of the \( b \) is almost 1 \([8]\). Nowadays, growing complexity and CASE tools make the cost estimations models not applicable in the "field". Therefore, a new type of estimation that not takes into consideration the number of lines of code was introduced. This is based on the so called function points where the \( \text{Effort} = a + bFP \)[9]. A different approach was done by the Basili and Panilip-Yap in 1985 where they suggest that the cost of the project can be estimated based on the number of pages of documentation instead of the code. Therefore they built a linear mathematical description of this according to \( \text{Effort} = a + b\ast \text{NumberOfPagesOfDocumentation} \), where the number of pages did not include sections of the codes. Regarding the mathematical expression, it was demonstrated later that the model is not usable because the number of pages of documentation vary from the chosen software process, from individual to individual; it depends by the chosen programming language and at least by
the company where it was applied. Apparition of the object oriented objects determines the researchers to take into consideration the number of objects, packets, methods and services based on the mathematical description \( \text{Effort} = a + b \times \text{NumberOfObjects} + c \times \text{NumberOfServices} \) [10]. After 2000, the models are not based exclusively on the LOC because the LOC is not relevant anymore if is not taken into consideration with others characteristics of the software. Some examples are the models of the Jeffery from 2000 according to \( \text{Productivity} = a \times \text{LOC} + b \times \text{MaximalTeamMember} - c \) where the productivity is the productivity is expressed in the man-month effort. Also a good example is the Putnam model [11] where suggests a model related to the lines of code based on the time needed to deliver a system and total effort of lifecycle and size of the system according to the mathematical model: \( \text{LOC} = C_k K^{4/3} t_d^{-4/3} \), where the \( K \) if the total effort related to the chosen lifecycle, \( t_d \) is the time until delivery and the \( C_k \) is a project related coefficient for expressing the project features. As an assumption for this model, the effort invested by the team follows the Rayleigh curve.

One of the most known estimation model is the COCOMO (Constructive Cost Model) model. It was developed in two phases and the models are known under the name of COCOMO I and COCOMO II. According to Dr. Barry Boehm, the researcher that creates the mathematical description for the estimation model, the model is built "to help people reason about the cost and schedule implications of their software decision". The first version of COCOMO was stable for the waterfall model, but evolution of software to object oriented paradigm and RAD approach make the model ineffective. Therefore, COCOMO II was introduced for supporting also the new models which capabilities are KLOC, Function Points and Object Points [12]. Also, it has three sub-models whom preconditions are the type of the software process used [13]:

- **Post Architecture model**: this model estimates the actual development process and the maintenance of the actual project. It is known the fact that not all calibration parameters for all COCOMO II are fixed, but they can be used in an experimental way.

- **Early Design model**: according to [14] involves exploration of alternatives software/system architecture and concepts of operation.

- **Application Compositions model**: this model estimates also the prototyping phase and the effort indicated by the risk assessment procedure and the technology maturity.

As a conclusion about the COCOMO models, they can be used for setting the project budgets according to the estimated cost, deciding which parts of the projects can be developed from scratch, reused from an old part or just bought as a COTS (components off-the-shelf) component and also allows companies to set up their development process and product life cycle for balancing the "triangle of the quality".

b) **Estimation models based on the expert's experience**: this fundamental of the approach is based on the fact that a number of experts in domain are grouped for estimating the costs. For this, some steps should be followed:
- Every expert will received a set of the requirements of the estimated system and will have an amount of time in which all requirements will have to be understood at all.
- Project manager will give the experts some forms to be filled with the estimation for each component. These forms will be filled individually.
- In a meeting, the results are presented and the differences will be discussed and reasoned.
- The steps 2), 3) will repeat until there will not be differences higher than a predefined threshold.

c) **Estimation models based on comparison with other existing models**: in the automotive industry it is preferred a "bottom-up" approach because each component is estimated then the costs for each component is summed and this is the total cost. Now, also these estimations are not very cleared because the estimated cost frequently is not taken into account the interruptions or the changes in the requirements in a release starting from the middle of the project. Now, since the AutoSAR is "entering" faster in the automotive industry, the forecasting of the costs will be easier because the basic software modules can be build from others suppliers(tiers) and based on the requested prices a estimation can be done. The costs for them can be requested because the components are already built.
4. A proposal for a cost estimation model using genetic algorithms

Evolution of the software procedures and technologies makes very difficult to estimate the work packages for a developer and directly this in involving also in the estimation of the entire project. As we can see in the picture from below that show the dependency between the productivity and the effort estimation based on the current tasks of the developer. In a developer life, in the beginning, the productivity (or development velocity) is at a lower value, but due increasing in the experience growth this value. Also, after reaches to know the current tools, the functionality and language, the developer will be able to develop generic component that is something new for him and resulting another decrease in the velocity. New programming languages and the development of the COTS acts as obstruction in increasing of velocity.

The model proposed to be analyzed considered a person as the following triplet \( P = \{ S, A, W \} \), where \( S = \{ s_i \mid s_i \in \text{SKILLS}, 1 < x < \text{MAX\_SKILLS} \} \), \( A \) is the allocation for the person for the current task, and \( W \) represent a coefficient for a people that is willing to learn; \( 0 < W < 1 \). A project is consists from several task that can be defined as a cost means the task duration and the necessary skills to be executed. As a mathematical description \( T = \{ C, S \} \) where \( C \) means the task duration and it will be expressed in man days \( (MD) \) and the required skills can be one or several skills from the set of skills. Also as an assumption for calculation of the costs for the overall project is the fact that a \( MD \) represents a fixed rate because even if a person has a higher wage than other the differences can be calculated as a medium value. A project also comprises a plurality of tasks until if finished, but also this task has dependency between them because some of them cannot be performed until others are finished. There are constraints of types "no earlier than", "as soon as possible". It is clear that the set of tasks can be grouped into a matrix as the adjacency. Now because all of these have been considered, my proposal is to define a model for cost estimation using as inputs the resources available at the beginning of the project and the multitude of the tasks that can be performed in terms of man days. For calculation of these there are several methods, but the preferred one is using genetic algorithms in the first phase. Thus the mathematical model is formulated as follows:

a) **Input parameters**: \( T_i \) - means the totality of the tasks of the project. The relation between two tasks is as follows: it \( T_i = T_j \) means that the tasks can be started in the same time, \( T_i - T_j = X > 0 \) means that the \( T_i \) can be started after \( T_j \) after \( X \) days. Also as inputs parameters will be taken into consideration the person available at the beginning of the proposed model and for each task a cost expressed in the man days is taken into consideration. Taking into consideration all described above, a project will be represented as an oriented graph with weights on arcs.

b) **Genetic representation**: a chromosome will be represented by a binary array where in each position is coded the ID of the person which is needed to perform the task. Number of position in the array it will be the same with number of tasks.

c) **Fitness function**: the reason for the fitness function is to select the "fittest" chromosome (gene) from the current genome for reproduction in later stages. The risk for the selection of a "not-fit" fitness function is to have a local maximum instead of a global one as it can be seen in the picture from below. As fitness function the following algorithm is proposed: calculate for each task the effectiveness factor considering the allocation factor and the current skills. If the person assigned has not the skills for performing the task, then the cost should be increased with the \( \text{cost} \times \text{willing to learn coefficient} \). It is understandable that at the total cost of the task will be added also the \( \text{allocation factor} \times \text{total cost} \). For all tasks, the effectiveness factor shall be calculated in the same way. The function to minimize is the total cost of the project optimizing in the same time also the resource allocation.

d) **Selection of the fittest chromosomes for reproduction** will be done using the roulette wheel method where the probability to extract an individual will be proportional with the sum of the fitness functions for all chromosomes from the current genome. The extraction will be repeated as many times as the number of individuals, but in the end the same number of chromosomes into the population we should have, so it might need for regenerating the of the population. Next step is to recombine the existent chromosomes through crossover operation with a chosen probability of 0.6. No
special operators will be used for the first variant of the genetic algorithm, but it will be taken into consideration for the next variant the definition of some operators for not allowing the generation of not allowed chromosome. What is important to know is the fact that the aging is not taken into consideration in the current model.

As additional improvements of the algorithms can be used the model using the adaptive evolutionary algorithms for travelling salesman problem with precedence constraints as authors described in [15]. The results will be compared and based on this the tuning of the parameter will be done. Also an alternative solution for estimation of the costs for the proposed model is using neural networks. The results obtained using the neural network can be compared with the results obtained using genetic algorithms. For all steps above, the datasets will be used from the COCOMO 81 datasets provided by the PROMISE organization. The reason for comparing the results is to have a unified estimation model for different software process model.

As additional improvements of the algorithms can be used the model using the adaptive evolutionary algorithms for traveling salesman problem with precedence constraints as authors described in [15]. The results will be compared and based on this the tuning of the parameter will be done. Also an alternative solution for estimation of the costs for the proposed model is using neural networks. The results obtained using the neural network can be compared with the results obtained using genetic algorithms. For all steps above, the datasets will be used from the COCOMO 81 datasets provided by the PROMISE organization. The reason for comparing the results is to have a unified estimation model for different software process model.

5. Conclusion
As a conclusion, it can be seen that the estimation models are adapting in the same time with the new technologies and emerging systems and they depend largely by the chosen software development process. An Agile based project for sure cannot be compared with a V-cycle based software. Finding a mathematical model for forecasting the costs of a project it is very likely to not to be developed, but a mathematical description that can be applied on one or two development process with a high precision can be found. Now, taking into consideration the things stated at the related work section, these models even they can predict now precisely the cost estimation, it can happen that starting with the third generation of the product in the same conditions as was considered for the previous generations the used model to not allow forecasting with the precision of the costs. This can be solved by doing also continuous improvements on the models in the same way that for software development the continuous integration and improvements methods are used at a large scale. Mathematics is also extending from algorithms point of view (e.g. genetic algorithms, neural networks) so these new methods together with the extending of databases for projects can be the solution of counteract time.

References
[1] Ambler S V 2014 The Non-Existing Software Crisis: Debunking the chaos report, Dr. Dobb's Journal (Online: http://www.drdobbs.com/architecture-and-design/the-non-existent-software-crisis-debunki/240165910)
[2] Amour P G 2003 The laws of software process: A New Model for the Production and Management of Software, First Edition, Auerbach Publication, Boca Raton, FL USA
[3] Scacchi W 2002 Process Models in Software Engineering, Encyclopedia of Software Engineering, John Wiley and Sons, Hoboken, NJ USA
[4] Balaji S and Sundararajan M M 2012 Waterfall Vs V-Model Vs Agile: A comparative study on SDLC, International Journal of Information Technology and Business Management 2(1) 26-29
[5] Briciu C V, Filip I and Heininger F 2013 A new trend in automotive software: AUTOSAR concept, 8th IEEE International Symposium on Applied Computational Intelligence and Informatics SACI 2013, Timisoara, Romania, May 23-25, pp 251 – 256
[6] Walston C E and Felix C P 1977 A method of programming measurement and estimation, IBM
Systems Journal 16(1) 54-73

[7] Banker R D, Chang H and Kemmerer C F 1994 Evidence on Economies of Scale in Software-Development, Information and Software Technology 36(5) 275-282

[8] Zivadinovic J, Medic Z, Maksinovic D, Damnjanovic A and Vujici S 2011 Methods of effort estimation in software estimation, International Symposium Engineering Management and Competiveness EMC 2011, Zrenjanin, Serbia, June 24-25, pp 417-422

[9] Kemerer C F 1993 Reliability of function points measurement: a field experiment, Communications of the ACM 36(2) 85-97

[10] Brownlow L S 1994 Early Estimating in the Object-Oriented Analysis and Design Environment, Proceedings of the European Software Cost Modelling Conference, Ivrea, Italy

[11] Putnam L H and Myers W 2003 Five core metrics: the intelligence behind successful software management, Dorset House Publishing, New York, NY USA

[12] Arnuphaptrairong T 2013 Early Stage Software Effort Estimation using Function Point Analysis: An Empirical Validation, International Journal of Design, Analysis and Tools for Integrated Circuits and Systems 4(1) 15-21

[13] Boehm B W 2006 The COCOMO II Suite of Software Estimation Models, COCOMO/SSCM Forum 21 Tutorial, University of Southern California, Center for Software Engineering, Las Vegas, USA

[14] Selby R W 2007 Software Engineering: Barry W. Boehm's Lifetime Contributions to Software Development, Management and Research, Wiley-IEEE Computer Society Press, Hoboken, NJ USA

[15] Sung J and Jeong B 2014 An adaptive evolutionary algorithm for traveling salesman problem with precedence constraint, The Scientific World Journal 2014(4) 1-11