Journal of Expert Systems (JES)
Vol. 1, No. 4, 2012, ISSN 2169-3064
Copyright © World Science Publisher, United States
www.worldsciencepublisher.org

An Intelligent Software Effort Estimation System

Ziauddin, Khairuz Zaman Khan, Shahid Kamal Tipu, Shahrukh Zia

Gomal University, Pakistan

ziasahib@gmail.com

Abstract: As the computer software has become an integral part of any industry, need for accurate forecasting of software development cost has also been increase. Different software effort estimation models are being used, but unfortunately these models have been developed for specific development environments and they support specific software development methodologies. Modern software development is not bound to some specific technology or methodology. This research focuses on development of an expert system which gathers expert knowledge in software cost estimation and integrates it. The developed expert system provides software engineers an easy mechanism to determine software effort for different types of software. The efficiency of the developed system has been compared with existing cost estimation methods. The results show that it has better accuracy than other models.

Keywords: Software Effort Estimation, Knowledge Base, Expert System, Inference Engine

1. INTRODUCTION

In recent years, software has become the most expensive component of computer system projects. The bulk of the cost of software development is due to the human effort, and most cost estimation methods focus on this aspect and give estimates in terms of person-months.

Accurate software cost estimates are critical to both developers and customers. They can be used for generating request for proposals, contract negotiations, scheduling, monitoring and control. Underestimating the costs may result in management approving proposed systems that then exceed their budgets, with underdeveloped functions and poor quality, and failure to complete on time. Overestimating may result in too many resources committed to the project, or, during contract bidding, result in not winning the contract, which can lead to loss of jobs.

Most cost estimation models attempt to generate an effort estimate, which can then be converted into the project duration and cost. Although effort and cost are closely related, they are not necessarily related by a simple transformation function. Effort is often measured in person/months of the programmers, analysts and project managers. This effort estimate can be converted into a dollar cost figure by calculating an average salary per unit time of the staff involved, and then multiplying this by the estimated effort required.

Practitioners have struggled with three fundamental issues:

· Which software cost estimation model to use?
· Which software size measurement to use – lines of code (LOC), function points (FP), or feature point?
· What is a good estimate?

Most cost models are based on the size measure, such as LOC and FP, obtained from size estimation. These models are suitable for the development technologies which are based on written lines of code, but modern technologies are based on component based software development. These models do not support such methodologies. Similarly modern software development methodologies, like agile, is also not supported by these models. However there are some effort estimation models which have been developed specifically for such type of unconventional software development.

In this paper we are going to develop an intelligent Expert System that supports all type of software development regardless of their type - either using conventional computer languages or component based visual languages. It also supports all types of software development methodologies – from conventional waterfall like sequential methodologies to iterative agile methodologies.

1.1 Cost Estimation Techniques

Cost estimation tools, or model-based estimation techniques use data collected from past projects combined with mathematical formulae to estimate project cost. These models need system size as input. The main model-based techniques include COCOMO, SLIM, RCA PRICE-S, SEER-SEM, and ESTIMACS. The existing effort estimation techniques are broadly classified as regression-based models, learning-oriented models, expert based approaches and composite-Bayesian methods.

Most of the software estimation models are based on regression technique (Matson et al., 1994). Regression models normally use previous data, constructed by collecting data on completed projects and developing regression equations that characterize the relationships among the different variables (Fairley, 1992). Estimates are made by substituting the. New project parameters are substituted into mathematical model. This model is evaluated on regression data to make estimates. In these models software development effort is simply dependent variable of some predicted variables like Size, Effort adjustment factors, Scaling factors etc. for regression equation.

Regression models however need certain conditions in some cases to be fulfilled particularly (Finnie et al., 1997). These conditions are discussed by Boehm and Sullivan (1999), and are based on experience from the use of regression-based models. These typical conditions include availability of a large dataset, no missing data items, no outliers, and the predictor variables are not correlated. The collection of approaches that fall under the heading of regression-models include ordinary least-squares regression (OLS), classification and regression trees (CART), stepwise analysis of variance for unbalanced data sets (stepwise ANOVA), combinations of CART with OLS regression and analogy, multiple linear regression, and stepwise regression (KEAVENEY, 2006).

There are other types of model, called Learning-oriented models which are based on learning from previous estimation experience. These models attempt to automate the estimation process by training themselves from previous experience to build computerized models (Boehm et al., 2000). These models are capable of learning incrementally and refining themselves as new data are provided over time (Lee et al., 1998). Learning-oriented models cover a wide area and include techniques such as artificial intelligence approaches, artificial neural networks, case-based reasoning (Mukhopadhyay and Kekre, 1992), machine learning models, decision-tree learning, fuzzy logic models, knowledge acquisition and rule induction (Burgess and Lefley, 2001). The main model-based techniques include COCOMO, SLIM, RCA PRICE-S, SEER-SEM, and ESTIMACS. These estimation models produce an estimate of the cost, effort or duration of a project based on factors such as the size and desired functionality of the system.

An important expertise based approach was found by Briand et al. (1998) to be “comparison to similar, past projects based on personal memory”. The expertise based approaches are useful when no quantified, empirical data is available (Boehm et al., 2000). They provide a practical, low-cost and highly useful process (Johnson et al., 2000). Another estimation technique used for software effort estimation is analogy based estimation. The technique examines past projects and uses the information retrieved as a guide estimate for the proposed project (Angelis et al., 2001, Jørgensen et al., 2003). The Checkpoint method is an example of an analogy-based approach to software estimation (Fairley, 1992). In this technique heuristics are derived from actual project data or a formalization of expert opinion. In order to derive these heuristics some form of project data or information are used. These heuristics are, then, used to estimate productivity, quality or size (Hihn and Habib-agahi, 1991, Fairley, 1992). Expert judgment Estimation is also one of the popular estimation technique in software effort estimation which is based on the accumulated experiences of teams of experts in order to come up with project estimates (Peters and Pedrycz, 1999, Stamelos and Angelis, 2001). This technique is used where the estimation process is primarily based on “non-explicit, non-recoverable reasoning processes”, or perception and intuition (Jørgensen, 2004b).

Expert Judgment techniques have been criticized by experts for their reliance on human memory and the lack of repeatability of such memory-based approaches (Mukhopadhyay et al. (1992, (Mendes et al., 2002); however reports have proven it to be the dominant strategy in software development estimation (Jørgensen, 2004a, Höst and Wohlin, 1997, Moløkken and Jørgensen, 2003, Moløkken-Østvold et al., 2004). The Delphi technique and work breakdown structure (WBS), top-down and bottom-up estimation (Tausworthe, 1980), reasoning by analogy, formal reasoning by analogy, informal reasoning by analogy, and rules of thumb (Jones, 1996) fall under expert judgment technique.

The strengths of expertise based methods and regression-based methods were combined to introduce a new estimation approach called the Bayesian approach which is a semi-formal estimation process (Ferens, 1988). Bayesian analysis allows for the fact that the data required for use in most estimation techniques is typically of poor quality or incomplete. Expert judgment is incorporated in this approach to handle the missing data and provide a more robust estimation process (Boehm and Sullivan, 1999). Bayesian analysis has been used in many scientific disciplines and was used in the development of the COCOMO II model (Chulani et al., 1999, Boehm et al., 2000). Cost Estimation, Benchmarking and Risk Analysis (COBRA) is an example of a composite estimation model (Ruhe et al., 2003).

With the introduction of Component based 4GL technologies, existing parametric models failed to determine development effort as these technologies are not based on LOC. Some new effort estimation models have been introduced for such technologies (ZIA et.al 2011).

1.2 Expert Systems

Expert systems are computer programs that are derived from a branch of computer science research called Artificial Intelligence (AI). AI's scientific goal is to understand intelligence by building computer programs that exhibit intelligent behavior. It is concerned with the concepts and methods of symbolic inference, or reasoning, by a computer, and how the knowledge used to make those inferences will be represented inside the machine.

Of course, the term intelligence covers many cognitive skills, including the ability to solve problems, learn, and understand language; AI addresses all of those. But most progress to date in AI has been made in the area of problem solving -- concepts and methods for building programs that reason about problems rather than calculate a solution. AI programs that achieve expert-level competence in solving problems in task areas by bringing to bear a body of knowledge about specific tasks are called knowledge-based or expert systems. Often, the term expert systems is reserved for programs whose knowledge base contains the knowledge used by human experts, in contrast to knowledge gathered from textbooks or non-experts. More often than not, the two terms, expert systems (ES) and knowledge-based systems (KBS), are used synonymously. Taken together, they represent the most widespread type of AI application. The area of human intellectual endeavor to be captured in an expert system is called the task domain. Task refers to some goal-oriented, problem-solving activity. Domain refers to the area within which the task is being performed. Typical tasks are diagnosis, planning, scheduling, configuration and design.

Building an expert system is known as knowledge engineering and its practitioners are called knowledge engineers. The knowledge engineer must make sure that the computer has all the knowledge needed to solve a problem. The knowledge engineer must choose one or more forms in which to represent the required knowledge as symbol patterns in the memory of the computer -- that is, he (or she) must choose a knowledge representation. He must also ensure that the computer can use the knowledge efficiently by selecting from a handful of reasoning methods.

1.2.1. Components of Expert System

Every expert system consists of two principal parts: the knowledge base; and the reasoning, or inference, engine.

The knowledge base of expert systems contains both factual and heuristic knowledge. Factual knowledge is that knowledge of the task domain that is widely shared, typically found in textbooks or journals, and commonly agreed upon by those knowledgeable in the particular field.

Heuristic knowledge is the less rigorous, more experiential, more judgmental knowledge of performance. In contrast to factual knowledge, heuristic knowledge is rarely discussed, and is largely individualistic. It is the knowledge of good practice, good judgment, and plausible reasoning in the field. It is the knowledge that underlies the "art of good guessing."

Knowledge representation formalizes and organizes the knowledge. One widely used representation is the production rule, or simply rule. A rule consists of an IF part and a THEN part (also called a condition and an action). The IF part lists a set of conditions in some logical combination. The piece of knowledge represented by the production rule is relevant to the line of reasoning being developed if the IF part of the rule is satisfied; consequently, the THEN part can be concluded, or its problem-solving action taken. Expert systems whose knowledge is represented in rule form are called rule-based systems.

Another widely used representation, called the unit (also known as frame, schema, or list structure) is based upon a more passive view of knowledge. The unit is an assemblage of associated symbolic knowledge about an entity to be represented. Typically, a unit consists of a list of properties of the entity and associated values for those properties.

Since every task domain consists of many entities that stand in various relations, the properties can also be used to specify relations, and the values of these properties are the names of other units that are linked according to the relations. One unit can also represent knowledge that is a "special case" of another unit, or some units can be "parts of" another unit.

The problem-solving model, or paradigm, organizes and controls the steps taken to solve the problem. One common but powerful paradigm involves chaining of IF-THEN rules to form a line of reasoning. If the chaining starts from a set of conditions and moves toward some conclusion, the method is called forward chaining. If the conclusion is known (for example, a goal to be achieved) but the path to that conclusion is not known, then reasoning backwards is called for, and the method is backward chaining. These problem-solving methods are built into program modules called inference engines or inference procedures that manipulate and use knowledge in the knowledge base to form a line of reasoning.

The knowledge base an expert uses is what he learned at school, from colleagues, and from years of experience. Presumably the more experience he has, the larger his store of knowledge. Knowledge allows him to interpret the information in his databases to advantage in diagnosis, design, and analysis.

Though an expert system consists primarily of a knowledge base and an inference engine, a couple of other features are worth mentioning: reasoning with uncertainty, and explanation of the line of reasoning.
Knowledge is almost always incomplete and uncertain. To deal with uncertain knowledge, a rule may have associated with it a confidence factor or a weight. The set of methods for using uncertain knowledge in combination with uncertain data in the reasoning process is called reasoning with uncertainty. An important subclass of methods for reasoning with uncertainty is called "fuzzy logic," and the systems that use them are known as "fuzzy systems."

Because an expert system uses uncertain or heuristic knowledge (as we humans do) its credibility is often in question (as is the case with humans). When an answer to a problem is questionable, we tend to want to know the rationale. If the rationale seems plausible, we tend to believe the answer. So it is with expert systems. Most expert systems have the ability to answer questions of the form: "Why is the answer X?" Explanations can be generated by tracing the line of reasoning used by the inference engine (Feigenbaum, McCorduck et al. 1988).

The most important ingredient in any expert system is knowledge. The power of expert systems resides in the specific, high-quality knowledge they contain about task domains. AI researchers will continue to explore and add to the current repertoire of knowledge representation and reasoning methods. But in knowledge resides the power. Because of the importance of knowledge in expert systems and because the current knowledge acquisition method is slow and tedious, much of the future of expert systems depends on breaking the knowledge acquisition bottleneck and in codifying and representing a large knowledge infrastructure.

2. PROPOSED MODEL

The architecture of the proposed model has been shown in fig. 1. This model has four main components.

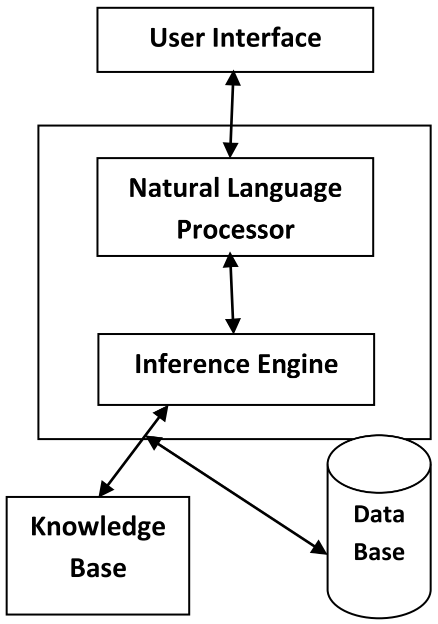

Fig 1. Architecture of the Model

i. **User Interface:** A graphical user interface has been developed which provide user with some predefined options as well as some options are provided where user can input in plain English. Predefined options are provided in cases where a numeric value is needed, otherwise natural language has been used for both questions as well as answers. The next question displays on the basis of previous response from the user. Thus an intelligent interaction occurs between user and computer.
ii. **Natural Language Processor:** NLP has been used to translate user response and query to specific rules and vice versa. It simply acts as an interface between User Interface and Inference Engine.
iii. **Inference Engine:** The basic objective of Inference Engine is to access knowledge Base on the basis of input parameters, supplied by the user. The developed Inference Engine is level 2-Type engine which not only provides basic reasoning but explanation facility has also been added that reproduces the logic to reach its conclusion. In order to reach a conclusion and offer an expert advice to the user, reasoning of the engine has been further strengthened by adding a database of static information. This database contains static information needed for calculation like effort adjustment factors in COCOMO.
iv. **Knowledge Base:** As the objective of the system is to effort estimation for different types of software development including variation of technology used as well as methodology followed, therefore four sets of rules have been incorporated in the knowledge base to support software effort determination for:

   a. Line of Code base software,
   b. Component base software
   c. Sequential methodology
   d. Iterative methodology

The developed rule base contains 394 rules and 101 actions. The knowledge base is capable of learning and addition knowledge can be asserted to knowledge base easily. The newly added knowledge can also be easily synchronized with the existing database. Similarly addition in database can also be easily synchronized with knowledge base.

3. RESULTS

Ten software has been used for experimentation. The development effort has been calculated using the developed tool, COCOMO II and Function Point Analysis. In some cases COCOMO II was not applicable as the technology used for the development of software is not Line of Code based.

The performance measure considered here is Mean Magnitude of Relative Error (MMRE), which is calculated as

$$\text{MRE} = \frac{|\text{E}-\text{E}'|}{|\text{E}|}$$

$$\text{MMRE} = \frac{\sum_{i=1}^{n} \text{MRE}_i}{n}$$

Where E is the actual and E' is the Calculated Effort. Table 1 shows the results of this experiment.

Table 1. Effort Estimation Comparison

No	Size	Actual	COCOMO	FPA	Proposed
1	19	301	214	243	293
2	49	951	841	931	984
3	41	521	601	453	537
4	25	208	282	181	201
5	15	151	128	95	127
6	11	115	131	91	121
7	4	15	15	13	16
8	12	81	78	65	82
9	36	678	751	634	722
10	14	256	241	201	249

Table 1 shows estimated efforts of three models. Table 2 shows comparison of the proposed model with other models.

MODEL	MMRE
COCOMO II	14.08
FPA	16.64
Proposed	5.08

Comparison of the model results in Table 2 shows that the proposed model has better estimation accuracy as compared to other models having 5.08% as compared to COCOMO II and FPA with 14.08% and 16.64% respectively, which shows at least 9% more accuracy than the existing models.

4. CONCLUSION

There exist many software effort estimation techniques, which need extensive training, even for experienced software engineers, to use them properly. Furthermore, there are situations where one technique can be implemented effectively but the same technique can not be implemented in all cases. The Intelligent Effort Estimation tool can e easily used even by novice users. The strong knowledge base enables it to e used in different situations. Using the same tool, Effort can be calculated using different estimation methodologies.

REFERENCES

AGARWAL, R., KUMAR, M., YOGESH, MALLICK, S., BHARADWAJ, R. M. & ANANTWAR, D. (2001) Estimating Software Projects. *ACM SIGSOFT Software Engineering Notes,* 26**,** 60-67.

ANGELIS, L., STAMELOS, I. & MORISIO, M. (2001) Building a Software Cost Estimation Model Based on Categorical Data. *Proceedings of the 7th International Software Metrics Symposium.*

BECK, K., BEEDLE, M., VAN BENNEKUM, A., COCKBURN, A., CUNNINGHAM, W., FOWLER, M., HIGHSMITH, J., HUNT, A., GRENNING, J., MELLOR, S., JEFFRIES, R., KERN, J., MARICK, B., MARTIN, R. C., SCHWABER, K., SUTHERLAND, J. & THOMAS, D. (2001) The Agile Manifesto.

BOEHM, B. W., ABTS, C. & CHULANI, S. (2000) Software Development Cost Estimation Approaches: A Survey. USC-CSE.

BOEHM, B. W. & SULLIVAN, K. J. (1999) Software Economics: Status and Prospects. Information and Software Technology, 41, 937-946.

BOSSAVIT, L. (2003) Project Management, The Movie. Cutter IT Journal, 16, 18-23.

BRIAND, L. C., EL EMAM, K. & BOMARIUS, F. (1998) COBRA: A Hybrid Method for Software Cost Estimation, Benchmarking, and Risk Assessment. Proceedings of the 20th International Conference on Software Engineering. Kyoto, Japan.

BRIAND, L. C., LANGLEY, T. & WIECZOREK, I. (2000) A Replicated Assessment and Comparison of Common Software Cost Modeling Techniques. Proceedings of the 22nd International Conference on Software Engineering. Limerick, Ireland.

BURGESS, C. J. & LEFLEY, M. (2001) Can Genetic Programming Improve Software Effort Estimation? A Comparative Evaluation. Information and Software Technology, 43, 863-873.

CHULANI, S., BOEHM, B. W. & STEECE, B. M. (1999) Bayesian Analysis of Empirical Software Engineering Cost Models. IEEE Transactions on Software Engineering, 25, 573-583.

FAIRLEY, R. E. (1992) Recent Advances in Software Estimation Techniques. Proceedings of the 14th International Conference on Software Engineering. Melbourne, Australia

FERENS, D. V. (1988) Software Size Estimation Techniques. Proceedings of the IEEE 1988 National Aerospace and Electronics Conference.

FINNIE, G. R., WITTIG, G. E. & DESHARNAIS, J.-M. (1997) A Comparison of Software Effort Estimation Techniques: Using Function Points with Neural Networks, Case-Based Reasoning and Regression Models. Journal of Systems and Software, 39, 281-289.

FOWLER, M. & HIGHSMITH, J. (2001) The Agile Manifesto. *Software Development,* August.
GOLDEN, J. R., MUELLER, J. R. & ANSELM, B. (1981) Software Cost Estimating: Craft or Witchcraft. *ACM SIGMIS Database,* 12**,** 12-14.

HIHN, J. & HABIB-AGAHI, H. (1991) Cost Estimation of Software Intensive Projects: A Survey of Current Practices. *Proceedings of the 13th International Conference on Software Engineering.* Austin, Texas.

HÖST, M. & WOHLIN, C. (1997) A Subjective Effort Estimation Experiment. Information and Software Technology, 39, 755-762.

JONES, C. (1996) By Popular Demand: Software Estimating Rules of Thumb. *Computer,* 29**,** 116-118.

JONES, C. (2003) Why Flawed Software Projects are Not Cancelled in Time. *Cutter IT Journal,* 16**,** 12-17.

JØRGENSEN, M. (2003) How Much Does a Vacation Cost? or What is a Software Cost Estimate? *ACM SIGSOFT Software Engineering Notes,* 28**,** 1-4.

JØRGENSEN, M. (2004a) A Review of Studies on Expert Estimation of Software Development Effort. *Journal of Systems and Software,* 70**,** 37-60.

JØRGENSEN, M. (2004b) Top-Down and Bottom-Up Expert Estimation of Software Development Effort. *Information and Software Technology,* 46**,** 3-16.

JØRGENSEN, M., INDAHL, U. & SJØBERG, D. (2003) Software Effort Estimation by Analogy and "Regression Toward the Mean". *Journal of Systems and Software,* 68**,** 253-262.

JØRGENSEN, M. & MOLØKKEN, K. (2003) A Preliminary Checklist for Software Cost Management. *Proceedings of the 3rd International Conference on Quality Software*

KEAVENEY S. and CONBOY K. (2006) Cost Estimation in Agile Development Projects. *Proceedings of the 14th European Conf. Information Systems (ECIS)*

LEE, A., HUNG CHENG, C. & BALAKRISHNAN, J. (1998) Software Development Cost Estimation: Integrating Neural Network with Cluster Analysis. *Information & Management,* 34**,** 1-9.

MATSON, J. E., BARRETT, B. E. & MELLICHAMP, J. M. (1994) Software Development Cost Estimation Using Function Points. *IEEE Transactions on Software Engineering,* 20**,** 275-287.

MENDES, E., WATSON, I., TRIGGS, C., MOSLEY, N. & COUNSELL, S. (2002) A Comparison of Development Effort Estimation Techniques for Web Hypermedia Applications. *Proceedings of the 8th IEEE Symposium on Software Metrics*

MILLER G.G. (2001) The Characteristics of Agile Software Processes. *Proceedings of the 39th Int'l Conf. and Exhibition on Technology of Object-Oriented Languages and Systems (TOOLS'01)*

MOLØKKEN-ØSTVOLD, K., JØRGENSEN, M., TANILKAN, S. S., GALLIS, H., LIEN, A. C. & HOVE, S. E. (2004) A Survey on Software Estimation in the Norwegian Industry. *Proceedings of the 10th International Symposium on Software Metrics.*

MOLØKKEN, K. & JØRGENSEN, M. (2003) A Review of Software Surveys on Software Effort Estimation. *Proceedings of the 2003 International Symposium on Empirical Software Engineering.*

MUKHOPADHYAY, T. & KEKRE, S. (1992) Software Effort Models for Early Estimation of Process Control Applications. *IEEE Transactions on Software Engineering,* 18**,** 915-924.

MUKHOPADHYAY, T., VICINANZA, S. S. & PRIETULA, M. J. (1992) Examining the Feasibility of a Case-Based Reasoning Model for Software Effort Estimation. *MIS Quarterly,* 16**,** 155-171.

PAULK, M. C. (2002) Agile Methodologies and Process Discipline. *CrossTalk, The Journal of Defense Software Engineering***,** 15-18.

PETERS, J. F. & PEDRYCZ, W. (1999) *Software Engineering: An Engineering Approach*, John Wiley & Sons, Inc.

RUHE, M., JEFFERY, R. & WIECZOREK, I. (2003) Cost Estimating for Web Applications, *Proceedings of the 25th International Conference on Software Engineering.* Portland, Oregon.

SCHMIETENDORF A., KUNZ M., DUMKE R. (2008) Effort estimation for Agile Software Development Projects, *Proceedings 5th Software Measurement European Forum, Milan*

STAMELOS, I. & ANGELIS, L. (2001) Managing Uncertainty in Project Portfolio Cost Estimation, *Information and Software Technology,* 43**,** 759-768.

Shoukat A., A Reducibility of the Kampéde Fériet Function, Advances in Computational Mathematics and its Applications (ACMA), Vol. 1, No. 1, March 2012; pp 76-79

Shubatah M.Q.H., Domination in product fuzzy graphs,Advances in Computational Mathematics and its Applications (ACMA), Vol.1,No.3,2012; pp 119-125

Syafadhli A.A.B., Mohamad D. and Sulaiman N.H., Distance-Based Ranking Fuzzy Numbers, Advances in Computational Mathematics and its Applications (ACMA), Vol. 1, No. 3, 2012; pp 146-150

TAUSWORTHE, R. C. (1980) The Work Breakdown Structure in Software Project Management. The Journal of Systems and Software, 1, 181-186.

Vajargah B. and Jahanbin A., Approximation theory of matrices based on its low ranking and stochastic computation, Advances in Computer Science and its Applications (ACSA), Vol. 2, No. 1, 2012; pp 270-280

Vajargah1 B.F., Moradi M. and Kanafchian M., Monte Carlo optimization for reducing the condition number of ill conditioned matrices, Advances in Computational Mathematics and its Applications (ACMA), Vol. 1, No. 1, March 2012; pp 169-173

Viswanadham K.N.S. and Raju Y.S., Quintic B-spline Collocation Method for Eighth Order Boundary Value Problems, Advances in Computational Mathematics and its Applications (ACMA),Vol. 1, No. 1, March 2012; pp 47-52

Yang X., Zhang Y., A New Successive Approximation to Non-homogeneous Local Fractional Volterra Equation, Advances in Information Technology and Management (AITM) Vol. 1, No. 3, 2012; pp 138-141

Ziauddin, Shahid Kamal Tipu, Khairuz Zaman, Shahrukh Zia, An Effort Estimation Model for Agile Software Development, Advances in Computer Science and its Applications (ACSA) Vol. 2, No. 1, 2012; pp 314-324

Ziauddin, Shahid Kamal Tipu, Khairuz Zaman, Shahrukh Zia, HOW TO USE REGRESSION OUTPUT FOR BETTER ESTIMATION, Journal of Science (JOS) Vol. 1, No. 3, 2012; pp 40-45
Ziauddin, Shahid Kamal Tipu, Khairuz Zaman, Shahrukh Zia, Software Cost Estimation Using Soft Computing Techniques, Advances in Information Technology and Management (AITM) Vol. 2, No. 1, 2012; pp 233-238

ZIA, Z.; RASHID, A.; UZ ZAMAN, K. (2011) Software cost estimation for component based fourth-generation-language software applications, *IET Software* , 5, Page(s): 103-110

VITAE

Dr. Ziauddin

Dr. Ziauddin is working as Assistant Professor in Institute of Computing & Information Technology, Gomal University, D.I.Khan since 1990. He has published more than 15 research papers in reputed International journals. His area of expertise is Software Engineering, Software Process Improvement, Software Reliability Engineering, Software Development Improvement, Software Quality Assurance and Requirement Engineering. He is Gold Medalist from Peshawar University in M.Sc Computer Science. He has Diploma in Computer Forensics from School of Education, Indiana University USA.

Shahid Kamal Tipu

Mr. Shahid Kamal Tipu is working as Assistant Professor in Institute of Computing & Information Technology, Gomal University, D.I.Khan since 2000. Currently he is doing his Ph.D. in Malaysia. His area of expertise is Information System Security and Software Quality. He has published his research in reputed International Journals. He is a renowned software engineer in local industry, having more than 50 software projects to his credit.

Dr. Khair uz Zaman Khan

Dr. Khair uz Zaman Khan is currently working as Director COMSATS University, Vehari Campus at deputation from Gomal University where he was acting as Dean of Science. He performed as Chairman of Economics department for 5 years. From Basic Education to Post Doctorate from UK, he passed all his examinations with distinction. He is one the most respected educationist of the country. He has more than 50 research papers to his credits.

As a well known economist, he also contributed to various national projects.

Shahrukh Zia

Mr. Shahrukh Zia is studying in Bachelor of Business Administration at CBA Gomal University, D.I.Khan. He is a young researcher, having keen interest in exploring relationship of business management with other disciplines. His research has been published in reputed journals.