Delphi Technique for the Software Effort Estimation -- An Outline for an Expert Judgment Method

Dr. A M Nageswara Yogi,
Professor and Head, Department of MCA,
T.John Institute of Technology, Bengaluru – 560083, India

Abstract **– Models for estimating the software effort are essential to arrive at the total cost of software projects. Majority of the available models are empirical and crude but a practical software developer can derive much benefit from these methods. To develop empirical equations for software effort estimation we need data from the large number of completed projects, which is difficult to obtain. In addition data from various projects cannot be compared since every software project will be developed under different environments and technologies. RAND Corporation developed the Delphi method originally to forecast the impact of technology on warfare. Delphi method uses expertise of experienced persons to arrive at useful values for estimates. A group of domain experts anonymously reply to a carefully compiled list of questions. Feedback in the form of statistical representation of their responses will be sent to them in order to improve upon their earlier values. The process repeats till converged values for estimates are obtained. If we replace the estimate by effort in person-months required for development of a software project to be undertaken then the Delphi method can be applied for software estimation. In this paper, we propose Delphi and Wideband Delphi methods for estimation of software projects by outlining various steps involved. In addition we have discussed a few estimation methodologies. We observe that every software effort estimation method uses expert's knowledge in quantifying some of its parameters and hence judgment plays a very important role in software effort estimation.**

Keywords – Effort; Estimation; Expertise; Judgment; Delphi

I. INTRODUCTION

Estimation – "*It is the mark of an instructed mind to rest satisfied with the degree of precision which the nature of a subject requires, and not to seek exactness where an approximation may suffice." – Aristotle, 330BC*

Systematic Software Development Process (SSDP) is the key for successful development of software projects. A software project is said to be successful if it meets all the user requirements including the functional and non-functional requirements and is delivered within the estimated cost and time. To estimate the cost of a software project it is required to accurately estimate effort in Person-Months (PM) which is a dominant component of the budget for a project to be undertaken. But effort required to deliver a software project, is dictated by many intangible parameters like project complexity, volatility and reliability of requirements and platforms, skills and capability of team members, etc. In addition it is important to have the knowledge about the software that must be developed, development process, means, personnel and the user establishment. Therefore, engineering judgment, experience, scientific principles, and techniques are to be synthesized to obtain the effort required to develop a software project successfully. We observe that the formulation of an effort estimation model by taking into account many local factors like available skilled-manpower, work-culture, environment, etc is one of the most important steps in software project management.

Estimating cost of the effort required for a project to be undertaken constitutes the basis for Budgeting, Tradeoff and Risk Analysis, Project Planning and Control, Software Improvement and Investment Analysis, Business Planning and Scheduling. Therefore, estimation plays a very important role in successful development of a software project. As per Wiki, the definition of Estimation is "the process of finding an estimate, or approximation, which is a value that is usable for some purpose even if input data may be incomplete, uncertain, or unstable". Another definition of an estimate is prediction or a rough idea to determine how much effort would take to complete a defined task [28]. An estimate is also a forecast or prediction and approximate of what it would cost. It is a rough idea about, how long a task would take to complete? We can also say that an estimate is an approximate computation of the probable cost of a piece of work. We estimate to avoid overshooting budget and time. Before we start estimating the project activities we must finalize all the requirements, if not analyze the frequency of changes in the requirements. In addition we have to make sure that infrastructure and manpower with well laid down responsibilities are in place and well documented assumptions and forecasted risks with possible remedies [22].

The estimation techniques are based on:

1. **The historical data and past experience:** Historical data can be formulized using Cost Estimation Relationships (CER) for system variables. Limitation is that these relationships do not include the technological changes that may be adopted in future software

developments. Past experience is very useful for validating the estimates. Again here also changes in the technologies and the environments have to be taken into account. To have a positive effect on estimates, we must adopt past experience along with training team members on the new technologies for the software development.

2. **The available data from the documents:** Estimates from the available data from the documents will give some insight to the estimates to be calculated based on the present requirements. Quantitative values from the previous documents amalgamated with qualitative analysis of the future requirements and the technologies will improve the estimates.

3. **The assumption:** It is assumed that the requirements are well understood by both the user and the developer without any ambiguity and doubts.

4. **The calculated risks to be taken:** Here we would like to quote former US Deputy Assistant Secretary of the Air Force Lloyd Mosemann [29] who said: *"Software is so vital to military system that, without it, most could not operate at all. Its importance to overall system performance, and the generally accepted notion that software is always inadequate, makes software the highest risk item and must be steadfastly managed... Failure to address risk has been the downfall of many Department of Defence acquisition programs. The system component with the greatest inherent risk has historically been software."*

II. SOFTWARE PROJECT ESTIMATION

Estimation lays foundation for all project planning activities and the planning provides road map for SSDP which is essential for successful completion of a software project. There is no simple way to make an accurate estimate of the effort required to develop a software project. Actual effort will be two to three times the effort estimated during the initial stage of the project based on inadequate information in user requirements definition [30]. The Standish group [33] Chaos survey of over 350 organization and 8000 projects conducted during 1994 produced the following results:

- 16% of the projects undertaken were only successful,
- 31% cancelled before completion,
- 53% overrun budget and schedule,

Subsequent Standish survey conducted in 2012 indicates that 39% of software projects undertaken were successfully developed, but still leaves much opportunity for further improvement. A field survey by the Eindhoven University of Technology [11] gives an overview of the state of the art of the estimation and control of the software development projects in 598 Dutch organizations. The conclusions are:

- 80% of the projects have overrun budget and schedule,
- 57% of the organizations do not use cost-accounting,
- 50% record no data on an ongoing projects,
- 35% of the organizations do not make an estimate,
- 50% is the mean overruns of the budget and the duration.

One of the most important reasons for such a situation is basically unrealistic software estimation. Estimation depends on many intangible parameters such as the software may run on unfamiliar computers, use of new technology, the people in the project may be inexperienced, underestimation of quality of work and complexity, unrealistic specifications, etc. This means that it is impossible to estimate software development costs accurately during the early stage of a project since most of the requirements are not clear. At this stage an estimate is very much required to bid the contract and we can obtain only rough initial estimates. The project manager can derive much benefit out of these initial estimates by using them for budgeting, tradeoff and risk analysis, project planning and control, software improvement investment analysis etc. Therefore, in literature we find lot of research taking place in formulating cost estimation models suiting the environments under which the software projects are being developed.

Cost estimation models can be classified into algorithmic and non-algorithmic models. In algorthmic models the costs are analyzed using mathematical formulae linking costs and inputs with metrics [31] like Kilo Delivered Source Instructions (KDSI), Function Points, and Object Points etc to produce an estimate. The formulae used in such formal models are developed from the analysis of historical data. A basic algorithmic model can be represented by an equation of the type:

$$\text{EFFORT (PM)} = a\,(\text{KDSI})^b + c,$$

where a, b and c are constants derived using the historical data from the past software projects. Table 1 gives a few models with values of a, b and c.

TABLE 1. ALGORITHMIC MODELS

Method	**a**	**b**	**c**
Bailey – Basil	0.73	1.16	5.5
Boehm Simple method	3.20	1.05	0.0
Doty	5.288	1.047	0.0
Halstead	0.7	1.50	0.0
SEL method	1.4	0.93	0.0
Walston – Felix	5.2	0.91	0.0

The most popular algorithmic cost estimation model for software projects is the COCOMO II developed by Barry Boehm and Ellis Harrowitz [8]. COCOMO is the parametric estimation model, which takes into account historical information as the base, making assumptions regarding changes, and extrapolating the information to the present project. The COCOMO model is an empirical model that was derived by collecting data from a large number of software projects. These data were analysed to discover formulae using statistical approach for the best fit to the observations. These formulae link the size of the software and product, project, team factors to the effort required. COCOMO model has a long history from initial installation in 1981 [4] through a refinement tailored to Ada software development [5] to COCOMO II [8].

Table 2 shows the basic COCOMO formula for three different types of projects, whose complexity is simple, moderate and embedded and M is the multiplier which depends on product, computer, project and personnel attributes. The accuracy of the estimations can be improved by calibrating the model to the local development environment by adjusting the weight-ages assigned to the various parameters of the model.

TABLE 2. COCOMO FORMULAE

Description	Formula	Project complexity
Well-understood applications developed by small teams.	EFFORT (PM) = $2.4(KDSI)^{1.05}M$	Simple
More complex projects where team members may have limited experience of related systems.	FFORT (PM) = $3.0(KDSI)^{1.12}M$	Moderate
Complex projects where the software is part of a strongly coupled complex of hw, sw, regulations and operational procedures.	EFFORT(PM) = $3.6(KDSI)^{1.20}$ M	Embedded

There are two broad approaches for non-algorithmic cost estimation – the top down approach and the bottom-up approach [25]. Top-down approach starts at system level. We start examining the overall functionality of the product and how that functionality is provided by interacting sub-functions. The cost/effort of system-level activities such as integration, configuration management and documentation are taken into account. Bottom-up approach starts at component level. The system is decomposed into components, and we estimate the effort required to develop each of these components. Add all these component efforts to compute the effort required for the whole system development. That is the bottom-up estimation models involve aggregating individual estimates for each task in the Work Breakdown Structure (WBS). The top-down approach allows managers to consider all subsystems whereas the bottom up approach allows for a more detailed assessment.

III. BASIC COST ESTIMATION MODEL

The cost of manufacturing any product or undertaking a project is the sum of various costs factors such as:

- RDDTE (Research, Design, Development, Training and Evaluation) costs,
- Acquisition cost which includes initial purchases, raw material, equipments, spares, training etc,
- Manufacturing costs include fuel/power, machinery and manpower costs,
- Initial maintenance costs like warranty which varies from one to five years from the date of installation,
- Infrastructure and Marketing costs which include buildings, depots, manpower and training of maintenance personnel etc.

A basic cost estimation model identifies the number of units to be produced and the total cost to produce them. The total cost divided by number of units produced gives the cost per unit. The manufacturer may add appropriate profit to the total production cost. In an addition the cost of logistics for transporting units to the desired places will be added. Usually RDDTE costs and the initial maintenance costs up to warranty period are distributed over the number of items produced [19]. More the number of units produced, the cost per unit will be reduced.

Software project cost estimation is no different than that of an engineering product except that the developed software is visible only through its output, which has to be verified, validated and rigorously tested before the delivery of the software product.

IV. SOFTWARE COST COMPONENTS

Software cost components are the costs of Hardware, Software, Design, Development, Testing, Manpower, Travel, Training and Maintenance. It is assumed that required hardware is available in the market. Effort cost is the dominant factor in most software projects which includes the salaries of engineers involved in the project, social and insurance costs etc. Effort cost must also include the overheads such as costs of building, heating, lighting, cost of support staff such as accountants, administrators, system mangers and technicians, costs of networking and communications, costs of shared facilities like library, staff restaurant, cost of social security and employee benefits such as pensions and health insurance.

Travel and training costs can be reduced by using electronic communications such as e-mail, web and videoconferencing. Costs of buildings, heating and lighting circuits are one time investment normally known as Capital costs. Recurring cost is the sum of salaries, power, maintenance; housekeeping etc and is usually known as Revenue costs. Usually revenue costs far exceed the capital costs.

V. WORK BREAKDOWN STRUCTURE

Work Breakdonw Structure (WBS) based techniques are good for planning and control. WBS was a standard of engineering practice in the development of both hardware and software. In this technique the complex project is divided into smallest pieces known as sub-functionalities (smallest components) which are considered as tasks. A set of related sub-functionalities leading to a common feature forms a functionality. A set of related functionalities makes a sub-module. A group of related sub-modules makes a module.

A software project consists of number of modules. Breakdown of the work into smallest components must be reviewed to confirm that all functionalities are included in WBS. Using such a breakdown of a project into micro level sub-functionalities we can easily figure out what are all the tasks need to be completed. Estimating each of the tasks would be easier than estimating overall complex project. Hence WBS is a way of organizing project components into a hierarchy that simplifies the tasks for effort (cost) estimation and control the project scheduling.

Moreover, if probabilities are assigned to the costs associated with each individual component of the hierarchy, an overall expected value can be determined from the bottom-up approach for total project development cost [2]. Expertise comes into play with this method in the determination of the most useful specification of the components within the structure and of those probabilities associated with each component. By using the components from WBS, the project manager and team will have an idea whether or not captured all the necessary tasks based on the project requirements. The WBS helps the project manager and the team to create the task scheduling and detailed cost estimation of the project.

A software WBS actually consists of two hierarchies, one representing the software product itself, and the other representing the activities needed to build that product [4]. The activity hierarchy Figure 1 [6] indicates the activities those may be associated with the given software component. The product hierarchy Figure 2 describes the fundamental structure of the software, showing how the various software components fit into the overall system.

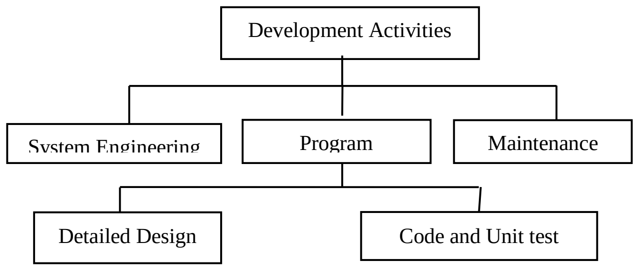

Figure 1. Activity Hierarchy WBS

In addition to effort estimation the other major use of the WBS is cost accounting and reporting. Each element of the WBS can be assigned its own budget and cost control number, allowing staff to report the amount of time they have spent working on any given project task or component. This information can then be summarized for budget control purposes.

Finally if an organization consistently uses a standard WBS for all its projects, over the time it accrues a very valuable database reflecting its software cost distributions. This data can be used to develop a software cost estimation models tailored to the organization's own experience and practices. While estimating the cost of each smallest component, it is required to determine the level of complexity of each task such as very simple, simple, average, moderate or complex. When more than one person with required skill level is available, then Project Manager can assign a task by reasonably judging the requirements of the task and the skill level of the person. Later on project manager should not repent that the task would have been assigned to the other person with similar skills. The parameter "Assign task to a team member" requires right judgment. Similarly, we can identify a number of parameters which needs judgment in software project management.

The advantages of WBS are

- In the WBS we generate all micro level tasks (smallest component) known as sub-functionalities to macro level tasks (modules) of a software project to be designed and developed. Such a division of the whole project into detailed tasks helps the Project manager, Team and Customer to raise the critical issues during the initial development phase of the project itself. This helps to focus on the scope of the project. Discussions at this stage help to understand the assumptions and clear ambiguities

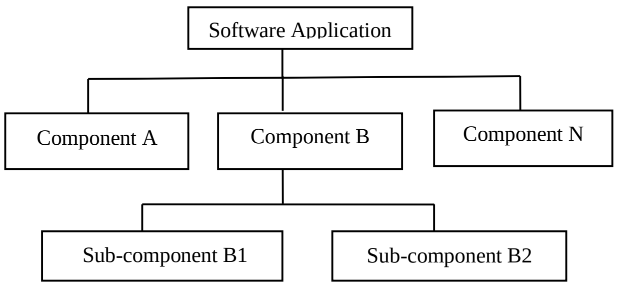

Figure 2. Software Product WBS

- An effective schedule and good budget plans can be made since all micro level tasks related to software project are available. It helps in generating a meaningful schedule and reliable budget.
- The micro level details of tasks help to assign particular module task to an individual. This makes the responsibility of team member very clear and so the individual is accountable to complete the task. Team members cannot escape from their duties as the detailed task in WBS is available.
- Since every member of the project team can understand the micro level tasks, it creates interest in them. This leads to increased participation and commitment of all the team members.

VI. PRICE-TO-WIN MODEL

In many cases the costs of many software projects must be estimated using incomplete user requirements for the system. Usually available information will be very little. Initial cost estimation is required to propose budgetary requirements. Under these circumstances pricing to win is commonly used strategy. The notion of pricing to win may be unethical and un-business like [25]. Here the project cost is agreed on the basis of outline proposal and based on the customer's budget rather than resources and capabilities. The specifications are constrained by the agreed cost. The customer and the developer must agree on acceptable system functionality. The fixed factor is cost and not functional requirements. The user requirements may change but not the cost.

The advantage is that the developer gets the project contract and there is no need of experts' opinions on any of the parameters. The disadvantages are that this approach does not consider the possibility of projects incurring loss owing to scope creep or any other reason and the probability that the customer getting the required system is small. Costs do not accurately reflect the work required [32]. There is no requirement of any judgment playing any role in this model.

VII. ANALOGY COSTING METHODOLOGY

An analogy costing model is a non-algorithmic costing model that estimates the cost of a project by relating it to another similar completed project. In this case the benchmark will be the completed project similar in scope, size, structure, environment, constraints, and functions to the current project [32]. Estimations of the completed projects will be used for reasoning and analogy to relate the costs to be incurred for the reasonably similar components in the current project. Such estimation may be conducted either top-down or bottom-up. This method has the advantage of assessing costs based on actual project experience rather than theoretical assumptions. However, effective estimation of the current project depends on the extent to which the completed project bears resemblance to the current project. The relationship between the similar components will always be subjective and based on one's own judgment. Expert opinions can be obtained while comparing the components. However in software development, changes in technologies and environment are very dynamic. Analogy model will not take these changes into consideration. A corrective factor for changes in technologies and environment has to be derived using judgment again to make the estimate more accurate.

If the WBS is available for the previouly completed project to which we are comparing with project to be taken up then we can generate WBS for this project for comparison. This will also helps to validate the effort values obtained by analogy costing methodology.

VIII. EXPERT JUDGMENT METHOD

An expert looks at long range future - fifteen or more years distant (that is three five-year plans and after) in terms of probabilities of occurrences. To an expert the future does not appear as unique, unforeseeable or inevitable, but visualizes a multitude of possible futures each associated with certain objectively qualified subjective probabilities. With an advance vision of the possible futures and identification of factors influencing their occurrences and an estimated extent of their influence it should become possible to manipulate or even design the future of one's choice. To be able to predict the future, an expert must have relevant basic information. The quality of prediction would not only depend upon the quality and quantity of information but also on the expertise with which a futurist can handle the raw information. The information may be three kinds – knowledge, speculation and opinion.

Knowledge is the kind of information which is highly substantiated by solid evidence of one kind or another. Speculation is the kind of the information that has little or no foundation. Speculation lies on one end of the information spectrum which has knowledge on the other. Opinion refers to the grey region that lies between knowledge on one end and speculation on the other. Opinion is the type of the information for which there is some evidence, but not enough to say it is solid. In the literature on the long-range forecasting this kind of material is often called “wisdom” or “insight” or “informed judgment” or “experience”. From the point of view of the amount of supporting evidence these terms can be considered euph emisms for opinion. The splitting /information spectrum into three distinct bands provides a crude quantifiable scale for its processing and evaluation. The probabilities may be considered to be high for knowledge and low for speculation and for opinion it may be considered to lie in between these two extremes. The probabilities that could be assigned to opinions would not only depend upon the professional stature of the person expressing them, but also upon his qualifications in the subject under consideration. In the area of opinions, therefore, there is always a significant probability that what is being expressed may be incorrect.

A futuristic would very much wish to base his/her predictions on knowledge and would positively wish speculation away. But the very nature of long range forecasting where a number of parameters are foggy at best and where a large number of visualized situations in the not-so-clearly-known-domain for example requirements specification for software project to be developed are clouded leaves very little option for avoiding speculation entirely.

Most of us are fond of algorithmic models, because majority of us try to formulate mathematical models and hence we feel that the majority of research work carried out in the software cost estimation has been devoted to algorithmic models. However by an overwhelming majority, expert judgment is the most commonly used estimation method. Studies by Fred Heemstral [11] and Vigder and Kark [27] reveal that 62% of estimators/ organizations use this intuition technique for software estimation.

Expertise-Based methods are useful in the absence of quantified and empirical data. The expert judgment costing model makes the assessment of costs by leveraging the experience of one or more Subject Matter Experts (SMEs) [32]. The method captures the knowledge and experience of practitioners seasoned within a domain of interest. Experts provide estimates based upon synthesis of the known outcomes of all the past projects to which they are privy or participated. In addition experts will take into account the current technologies and future trends while providing the estimates. However Vidger and Kark indicated that in general estimators did not refer to previous documents as it was too difficult to access or the expert could not see how the information would help in the accuracy of estimate. The study reveals that majority of the experts tended to use their memories of previous projects. Some may feel that expert judgment is simply a matter of guessing but Hughes [16] research indicates that experts tend to use a combination of informal analogy approach when similar projects from the past are identified. Therefore, the obvious drawback to this method is that an estimate is only as good as the expert's opinion, and there is no way to test that opinion until it is too late to correct the damage if that opinion proves wrong. Years of experience do not necessarily translate in to high levels of competency. Moreover, even the most highly competent individuals will sometimes simply guess wrong. These are some of the risks in this method. These risks can be reduced by listing a group of experts to utilize their expertise and by taking weighted average of their opinions. As we know that expert judgment is asking for an estimate of the task effort from some knowledgeable person about either the application or the development environment. It is often required to refine estimates as and when additional data becomes available. Furthermore, we have to use refined information in the next phases of an evolving project which is also a challenge for decision makers. Therefore judgment plays a very important role in software estimation [24]. Judgment includes assumptions and logical thinking. Assumptions are made based on the experience and expertise.

A common expert judgment model is the Delphi technique. The objective of this technique is the exploration of most creative and reliable ideas for decision making. The method involves constituting an experts panel, conducting a survey where each expert states their opinion independently, followed by a controlled feedback to the experts, and repeating the exercise multiple rounds until all the experts develop a consensus to identify a common cost estimate.

IX. THE DELPHI METHOD

The word Delphi refers to a place in Greece which was supposed to confer predictive powers to the person visiting the temple built there. The Rand Corporation adopted this name for a procedure to obtain most reliable consensus of opinion of group of experts. The Delphi technique [14] was developed at The Rand Corporation in 1948 originally to forecast the impact of technology on warfare (that is a way of making predictions about future events) - thus its name, recalling the divinations of the Greek revelation of archeological find located on the southern edge of Mt. Parnassus at Delphi.

The Delphi Method is an information gathering technique in which expert opinions are systematically solicited and collected. This method is applicable when estimates have to be made based on informed judgment. Its salient features are

- Anonymity,
- Structured information flow,
- Iterations with regular and controlled feedback and
- Statistical group response.

In its simplest form Delphi method eliminates the committee activity altogether thus reducing the influence of certain psychological factors such as hollow opinions, the unwillingness to abandon publicly expressed opinions and bandwagon effect of majority opinions [13].

The method entails a group of experts who anonymously reply to questionnaires and subsequently receive feedback in the form of a statistical representation of the "group response," after which the process repeats itself. The goal is to reduce the differences in responses and arrive at something closer to expert consensus. The expert surveys are conducted over multiple rounds until a consensus is reached. After each round, controlled feedback is provided to the experts, which encourages convergence of thought.

A key point to note about the Delphi Technique is that while gathering thoughts, the participants do not know who the other participants are? Hence, participants are not influenced by others taking part in the process. By using this method, we aim at obtaining both qualitative and quantitative results. Delphi represents a useful communication device among a group of experts thus facilitates the formation of group judgment [15]. The Delphi method is a top-down approach in aiming at the convergence of judgmental values estimated by a number of experts to obtain performance, cost of engineering products.

This method is very useful during acquisition of systems/ products, when more than one model of the same product/system are available in the open market. The only available information is the advertisement or brochures from the manufacturers. The Delphi method comes handy for us in deciding to select a model from available list. Usually many of us apply such a technique knowingly or unknowingly without formally recording the procedure. The Delphi method allows us to systematically record the experts' opinions for decision making.

Expert judgment relies on experience, background and common sense of key people. An expert might arrive at an estimate by considering the following:

- Similarity of the proposed project with completed ones.
- Earned experience from the previous projects.
- Available equipment including computers.
- Training given to the personnel who are going to be involved in the future projects.

The expert may be confident that the project is similar to the previous project but might have overlooked some factors that make the new project significantly different. It is also possible that expert making the estimate may not have an experience with a project similar to the present one. In order to compensate these factors, group of experts are requested to give the estimates. This minimizes individual oversights, personnel biases etc. Disadvantage of group estimation is the effect of interpersonal group dynamics, non-availability of authoritative figures in the group or the dominance of an overly assertive group member. The Delphi technique can be used to overcome these disadvantages. This method replaces direct debate with carefully designed program of sequential individual interrogations best conducted by a questionnaire interspersed with information and opinion feedback derived by computed consensus from the earlier part of the program. Some of the questions directed to the respondents (experts) may for instance inquire into reasons for previously expressed opinions, and collection of such reasons may then be presented to respondent in the group, together with an invitation to reconsider and possibly revive their earlier estimates. Both the inquiry into reasons and subsequent feedback of reasons adduced by others may serve to stimulate the experts to take into account considerations they might through inadvertence have neglected and to give due weight to factors they were inclined to dismiss as unimportant at first thought.

The mechanics of this technique maybe briefly explained as follows. A control group formulates a precise and unambiguous statement of the problem and prepares a questionnaire and invites opinions from group of carefully chosen experts for the first round of estimates. On receiving the replies from these experts, which generally have a very wide spread, the control group determines the inter quartile range that is the interval containing 50% of the responses. The processed first median response is fed back to experts who are asked to reconsider their previous answers and revise them if they so desired for second round. If the fresh response of an expert is still outside the inter quartile range, he is requested to state his reasons for thinking that answer should be that much lower or that much higher than the majority of the group.

Placing the onus of justifying relative extreme responses on the experts has the effect of causing those without strong conviction to move their estimates closer to the median, while those who feel they have good arguments for a “deviationist” opinion, to retain their original estimate and defend it.

In the third round the estimates now a narrower spread are again surmised and the experts given a concise summary of reasons advanced in support of their extreme positions. They are once again requested to revise their second-round estimates, taking the proffered reasons into consideration and giving them whatever weight-age they thought was justified. A respondent whose answer still remains outside the inter quartile range would be requested to state why he is un-persuaded by the opposing arguments.

In the fourth and generally the final round these criticisms of the reasons previously offered are resubmitted to the respondents who are given the final chance to revise their estimates. The median of these final responses may be taken as responses representing the nearest thing to group consensus.

There are many applications of Delphi method for engineering projects. The very first application of the Delphi method carried out at RAND Corporation as illustrated in publication by Gordon and Helmer [12] was to assess the direction of long range trends, with special emphasis on science and technology and probable effects on society. The study covered six topics [12] Automation; Population control; Scientific breakthroughs; Space program; War prevention and Weapon system.

In literature, I was not able to find either a case study or the details regarding utilizing the Delphi method completely for software effort estimation for a proposed software project. I have utilized Delphi method for finding the performance of certain parameters of fighter aircraft to develop a model for comparative performance of certain fighter aircraft. In this paper, I am proposing and outlining the following steps to employ Delhi method for software effort estimation as follows:

1. Project Director will form a Control Team. Control team in consultation with the Project Director and Users prepares a System Definition document containing specifications.

2. Control team analyses the System Definition document which includes both user requirements and system requirements. After the analyses, a meeting with the user will be arranged to sort out any ambiguities, misunderstandings, lack of clarity, confusions, and amalgamated requirements etc to finalize the Software Requirements Specifications (SRS) document.

3. Control team prepares a list of experts or estimators who will have no contacts and appoints a coordinator who is called as the facilitator. While selecting the experts the control team will consider their experience in software development, knowledge in application domain. It is better to select an odd number of experts with a minimum of three for estimation of statistical parameters.

4. Usually the SRS document will be quiet big and it may not be possible for experts to completely go through the document. Therefore the control team carefully prepares a questionnaire for recording responses of the experts to estimate effort, time, resources required for the software project to be undertaken. Along with the questionnaire the control team will send a brief about the project, listing the objectives, scope, timeline before which the experts are required to send their answers. Experts may call for clarifications from the control team for which the control team must be prepared to provide the explanations.

5. Experts study the specifications and complete the questionnaire anonymously. They may send queries to the facilitator for clarifications. After that they will send estimates/ responses which we designate as the first round estimates.

6. The control team based on the responses determines the inter quartile range, prepares and distributes the summary of the responses including any unusual rationales noted by some experts and again sends this information to all the experts.

7. Estimators complete another round of estimates, again anonymously, using the analyses of the results from the previous estimates and feedback from the facilitator and these results we call as second round estimates.

8. On receiving the responses from the experts the control team again determines the inter-quartile range. Experts whose estimates differ sharply from the group may be requested to provide justification for their estimates. Control team has to analyze the justification. If

acceptable the same can be sent to other estimators for their reactions.

9. If only one or two experts differ sharply and reasons are not justifiable in general or too adamant to change, then they can be left out of the group.
10. We repeat steps 8 and 9 until a consensus is reached by the panel of experts.

The process will be iterated for a minimum of four rounds; hopefully by this round the convergence might have occurred and the median of the final responses can be taken as answer representing the consciousness of the group of experts.

11. The control team prepares a report of analysis of all the rounds.
12. The project Director will arrange a presentation by the control team to the Management to conclude the procedure and for a possible decision making.
13. A letter expressing gratitude and thanks to all the experts participated in the process must be sent by the Head of the organization.

The above process can be adopted if the experts /estimators are outside the organization.

If there are a number of experts within the organization itself, then the control team can have restricted group discussions with experts to focus on issues where estimates vary widely at the end of each cycle and try to reduce the differences. However, control team has to see that there will be no dominance of one expert over the other. Even rationale for the widely varying estimates can also be discussed and the reasons cited can be analyzed giving equal weight for each of the experts. It is to be noted that the original Delphi technique avoided group discussions. Therefore we have the Wideband Delphi method which allows limited group discussions.

A. *Wideband Delphi Method*

The Wideband Delphi technique [4] accommodated group discussions in between the rounds. It attempts to gather the opinions of a group of experts with the aim of producing an accurate unbiased estimate. It is a structured technique of expert judgment involving the following procedure:

1. Control team issues specifications and an estimation questionnaire to the panel of experts.
2. A group meeting of project stakeholders including the experts will be conducted to discuss the software specifications and estimation issues.
3. Experts produce an independent first round estimates and confidentially submit to the facilitator of the control team.
4. The control team returns the estimates to the panel of experts indicating the inter quartile range of estimates along with the expert's personnel estimate.
5. Another group meeting of project stakeholders including the experts is conducted to discuss results and the issues.
6. Experts prepare revised independent estimates for the second round.
7. We repeat the steps 3-6 until a consensus is reached by the panel of experts.

The advantages of this method are that it removes the politics from an estimate as the experts do not communicate about their particular estimate and filters out extreme opinions making the estimate unbiased. The group discussion of the stakeholders is particularly advantageous as it ensures that any estimation issues are not overlooked. Boehm [4] indicates the effectiveness of the Wideband Delphi technique and stresses the importance of the group meeting. The disadvantages of this technique are that the method consumes lot of time and the panel of experts needs to have very good experience, patience to listen and not air their opinions during the discussions. Also availability of experts can not be guaranteed all the time

The Delphi technique has been successfully applied to a number of forecasting problems. In majority of cases using Delphi technique approach, a convergence of opinions has in effect been observed. In a few cases no convergence towards relatively narrow interval of values took place; the opinions have polarized around two distinct values, so that two schools of thought regarding a particular issue seemed to emerge. This may have been an indication that opinions were based either on different sets of data or on different interpretations of the same problem and the data.

The quality of prediction in the Delphi technique depends directly on the expertise of the participant respondents. If the problem is of an inter-disciplinary character, it is possible to refine the method to introduce assignment of weight factors to expressed opinions. An expert may be requested to make a self-appraisal of his relative competence in the matter under consideration. The Control group on the basis of this self-appraisal process of expert opinions determines the weighted median values.

B. *Convergence of estimates*

Depending on the requirements and the situation, an average of the estimates can be derived using arithmetical average or statistical mode from the opinions of the experts. The highest estimate can be sent to the expert who has given the lowest estimate for his opinions and comments and vice versa. While sending the estimates, the control team will request to revise their earlier estimates. This process may bring convergence of extreme answers received from the experts.

C. *Delphi Method for Software Risk and Schedule Management*

We can use the Delphi Method for any activity that requires the convergence of expert thought, such as in Risk Management and Scheduling. For identifying risks drivers and classifying and quantifying them, we can develop a suitable questionnaire so that the answers from the experts will be helpful to assign numerical values for the probability of occurrence and the consequence of the occurrence of each of the identified risk drivers. In addition to experts, we can consult experienced project leaders and members of teams for identifying the risk factors [22]. For schedule management, we can develop story cards to sum up the customer needs and each of these stories can be evaluated and estimated. If needed, these stories can be broken into tasks to estimate the required effort and resources [25].

D. Planning Poker

Assume that we need to estimate for project activities for an Extreme programming software development method. Extreme programming is perhaps the best known and most widely used agile method. The name was coined by Beck [3] because the approach was developed by using the best practices such as iterative development and customer involvement to extreme level. Here each project activity/ task that gives the client a value is called a story. Story writing is an imaginatively created account of possible events those can be visualized to occur with finite probabilities. The experts with their academic background, professional experience and specialized knowledge of past, present and trends for the future, synthesize situations that appear within the bounds of possibility and attempt to assign relative probabilities to the corresponding possibilities. Such technique is extensively used by military thinkers, social planners, long-term policy-makers and the like.

Accurate estimates for stories or activities are crucial for planning the schedule. Based on Delphi method Planning Poker was developed by Mike Cohen, the agile guru [18]. In this technique, each story is estimated individually by a number of experts and if there is consensus then the estimates are taken. If there is a disagreement, people share their thoughts and the estimation enters a second round. This is repeated until the estimation for the story has reached a consensus. Typically, only experts participate in the Delphi Method. However, in software project estimation, we can encourage the team members to participate. This creates enthusiasm, ownership within the team and is a typical characteristic of Scrum approach. The Scrum approach is a general agile method but it focus is on managing iterative development rather than specific technical approaches to agile software engineering.

X. PERFORMANCE OF ESTIMATION MODELS

There are various measures for assessing the accuracy of the models: Root Mean Square Error (RMSE), Mean Relative Error (MRE) and so on. RMSE is calculated based on a number of actual data observed and estimated values by a model; it is derived from basic magnitude of relative error which is defined as

$$RMSE = \sqrt{\sum (E_a - E_i)^2 / N},$$

where, E_a is the actual effort and E_i is the effort estimated by a selected method for ith case study and N is the number of case studies or number of observations. Mean Relative Error (MRE) is the main performance measure. It is estimated from relative error, which is the relative size of the difference between the actual and the estimated value. MRE is the percentage of the absolute values of the relative errors averaged over the number of observations. Here number of observations is again the number of case studies worked out. Hence,

$$MRE = (100/N)\sum(|E_a - E_i|)/E_i.$$

A large positive MRE would suggest that the model generally overshoots the effort, while a large negative value would indicate undershoots. AMN Yogi and Mala V Patil [20, 21] have used RMSE and MRE for comparative assessment of Yogi and Patil, Bailey Bassili, COCOMO, Doty and Walston-Felix estimation models. Boehm [4] specifed ten factots for evaluating cost estimation models. They are Constructiveness, Definition, Detail, Ease of Use, Fidelity, Objectivity, Parsimony, Prospectiveness, Stability, Scope. These factors give us a basic idea about the performance evaluation of estimation models.

XI. APPLICATIONS OF DELPHI TO SOFTWARE RELATED STUDIES

There are many applications of Delphi method for engineering projects. The very first application of the Delphi method carried out at RAND Corporation as illustrated in publication by Gordon and Helmer [12] was to assess the direction of long range trends, with special emphasis on science and technology and probable effects on society. Following are the few case studies in which the Delphi technique was applied to software related studies:

1. Abts and Boehm used Delphi technique to estimate initial parameter values for Effort Adjustment Factor (EAF) appearing in glue code effort estimation component of the COCOTS (COnstructive COTS) integration Cost model [1]. EAFs are similar to the estimation dimensions and corresponding project factors [7] which form the basis of the five sub-models that comprise ESTIMACS a software estimation model originally developed by Rubin [23]. Soliciting the opinions of a group of experienced software development professionals, Abts and Boehm arrived at converged values.

2. Chulani and Boehm used the Delphi technique to estimate software defect introduction and removal rates during various phases of software development life-cycle. These factors appear in COQUALMO (COnstructive QUALity Model) which predicts the residual defect density in terms of number of defects/unit size of code [9].

3. Chulani and Boehm also used the Delphi technique to specify the prior information required for the Bayesian calibration of COCOMO II [10].

4. Performance evaluation of estimation models is very useful for comparative assessment of these models. It gives statistical insight to the accuracy of models. To

my understanding it is very rare that expert opinion is sought for performance evaluation of estimation models. However, Vicinanza, Mukhyopadhya and Prietula [26] used experts opinion to estimate effort using Kemerer's data set without formal algorithmic techniques and found that the results outperformed the models in original study. MRE varied from 32 to 1107%. Kemerer himself performed empirical validation of four algorithmic models COCOMO, Estimacs, Function Point Analysis (FPA) and SLIM [17].

XI. CONCLUSIONS

There are a number of factors that influence the software development process. The technologies and environments are changing continuously. Most of the factors representing technology and environment are qualitative in nature. Quantification of these factors is essentially made on the basis of experience and expertise. Here judgment plays a very crucial role. An estimation method which is either non-algorithmic or algorithmic has to relay on quantities that are quantified by judgment. Keeping the importance of judgment to quantify various factors for software effort estimation in mind, in this paper I have discussed about a few estimation techniques and most important the Work Breakdown Structure, the Expert Judgment and the Delphi Technique. I have brought clearly how Work Breakdown Structure helps in software estimation to find the cost of micro level components of a software project. WBS also becomes a tool for validating the values obtained by Analogy and other methods.

There are lot of differences in the environments and the technologies under which the past software projects were developed and the projects which are under development or going to be undertaken. Only experienced persons will be able to forecast the future scenarios. In the absence of quantified, empirical data, expert judgment methods are very useful but are highly subjective. Success depends wholly on the skills and competence of the experts chosen. Problems with expert judgment methods are the expertise-calibration and scalability problems for extensive analyses. Therefore, I have carefully outlined various steps to be followed for successful utilization of Delphi method for software effort estimation.

As I have mentioned that every software effort estimation method uses expert's knowledge in quantifying at least a few of its parameters. Therefore judgment plays a very important role in effort estimation. Never the less it should be noted that all software estimation techniques are challenged by rapid pace of change in software technology. As per Boehm no one method should be preferred over all others. The key in arriving at sound estimates is to use variety of methods and then investigate the reasons why the estimates provided by one might differ significantly from those provided by others. If the software engineer can explain such differences to a reasonable level of satisfaction then it is likely that he or she has good grasp of the factors which are driving the costs of the project at hand; and will be better equipped to support the necessary project planning and control functions performed by management.

Work Breakdown Structures, identifying risks and opportunities, compiling the lessons learnt, brainstorming sessions will help in convergence of answers. Predicting the future is not an exact science, but the Delphi method can help us to understand the likelihood of future events and what impact they may have on our project.

ACKNOWLEDGMENTS

The author wishes to express his gratitude and heartfelt thanks to Dr. Thomus P John, Chairman, T.John Group of Institutions and Dr. Thippeswamy HN, Principal, T.John Institute of Technology, Bengaluru for their support and encouragement during preparation of this paper.

REFERENCES

[1] Abts,C, Baily, B and Boehm B. "COCOTOS Software Integration Cost Model: An Overview". Proceedings of the California Software Symposium, 1998.
[2] Baird B. Managerial Decisions Under Uncertainty, John Wiley & Sons, 1989.
[3] Beck, K. Extreme Programming Explained. Addition- Wesley, 2000.
[4] Boehm, B. Software Engineering Economics. Eaglewood Cliffs, NJ: Prentice Hall. 1981.
[5] Boehm, B. W. and Royce, W. Ada COCOMO and the Ada process model. Proc. 5th COCOMO users' Group Meeting, Pittsburgh: Software Engineering Institute. 1989.
[6] Boehm B. W, Chris Abts and Sunita Chulani, "Software development cost estimation Approaches – A survey", Annals of Software Engineering, Vol. 10, No.1-4, pp. 177-205, 2000.
[7] Boehm, B. W. Abts, C. et al. Software Cost Estimation with COCOMO II. Upper Saddle River. NJ: Prentice Hall. 2000.
[8] Boehm B. W and Ellis Harrowitz, "Software Cost Estimation with COCOMO II", Prentice Hall, 2000.
[9] Chulani S. "Modeling Defect Introduction", California Software Symposium – Nov. 1997.
[10] Chulani S, Boehm B and Steece B. (1998). "Calibrating Software Cost Models using Bayesian Analysis. IEEE Transaction on Software Engineering. Special Issue on Empirical methods in Software Engineering. 1998.
[11] Fred Heemstra, F. J. 'Software Cost Estimation', Information & Software Technol 34 (10) pp 627-639. 1992.
[12] Gordon, T. J., and Olaf Helmer. Report on a Long-Range Forecasting Study. P-2982 . Santa Monica (CA): The RAND Corporation. 1964
[13] Helmer Olaf and Rescher Nicholas. On Epistemology of inexact Sciences, Mgt Science, Vol. 6, pp. 47. 1959
[14] Helmer, O. Social Technology, NY. 1966.
[15] Helmer, O. Reassessment of cross-impact analysis. Futures, pp. 389-400. Oct. 1981
[16] Hughes, R.T., 'Expert Judgment as the estimation method'. Inf. and Software Technol. 38(2) pp. 67-75, 1996.
[17] Kermerer, C. F. "An empirical validation of software cost estimation models:. Communications of the ACM Vol. 30, No. 5 1987, pp-416-429.
[18] Mike Cohen "Agile Estimating and Planning". Prentice Hall Nov. 2005.
[19] Nageswara Yogi AM. "A model for LCC Estimation for Defence Equipment", Proc. of Int. Conf. on Trends in PLMSS-2006, pp. 415-423. 2006.
[20] Nageswara Yogi AM and Mala. V. Patil. Software effort estimation models and performance analysis with case studies." Int. J. of Comp. App. in Engineering, Technology and Sciences. Vol. 1 No. 2, 2009. Pp 558-565.
[21] Mala. V. Patil and Nageswara Yogi AM. "Importance of data collection and validation of for Systematic Software Development Process". Int. J. of Comp. Sc. & Inf. Tech. Vol. 3 No. 2. 2011.

[22] Nageswara Yogi AM and Mala. V. Patil. "Identifying Risks and Possible Remedies to Mitigate Them during Systematic Software Development Process". Int. J. of Engg. Res. and Tech. NCSE'14 Conference Proceedings Feb. 2014.
[23] Rubin H, "ESTIMATICS", IEEE, 1983.
[24] Steven Fraser, Boehm B. W. Hakan Erdogmus. Jørgensen M. Stan Rifkin. Mike Ross, "The Role of Judgment in Software Estimation," ICSE' 09, pp. 13-17, 2009.
[25] Somerville, Software Engineering 2007
[26] Vicnanza S. S., Mukhyopadhya, T and Prietula, M. J. "Software-effort estimation: an exploratory study of expert performance", Information Systems Research, Vol. 2 no.4 1991, pp-243-262.
[27] Vigder, M. R., and A. W Kark. "Software cost estimation and control. Feb. 1999, Report available in the net.
[28] http://www.softwaretestingclass.com/software-estimation-techniques/
[29] http://www.unf.edu/~ncoulter/cen6070/ handouts / ManagingRisk.pdf
[30]http://www.developer.com/java/other/article.php/1463281[31] http://yunus.hacettepe.edu.tr/~sencer/cocomo.html/
[32] http://www.brighthubpm.com/project planning/121839
[33] http://www.standishgroup.com