Organizational Strategies and Social Interaction Influence in Software Development Effort Estimation

B V A N S S Prabhakar Rao¹ & P Seetharamaiah²
¹Research Scholar, Department of Computer Science Engineering, JNTUK, Kakinada, India
²Professor Emeritus, Department of CS & SE, College of Engineering (A), Andhra University, India

Abstract: In software development cost estimation, effort allocation is an important and usually challenging task for project management. This paper observes the use of concepts in software effort estimation by analyzing group work with organizational strategies as outgoing practice. The purpose is to improve our understanding of how software professionals raise different types of information when talking, way of thinking and reaching a decision on a software effort estimate. Software effort estimation is a core task regarding planning, budgeting and controlling software development projects. However, providing exact effort estimates is not easy. Estimation work is increasingly group based, and to support it, there is a need to reveal how work practices are carried out as mutual efforts. This paper contributes to an understanding of the role of concepts in group work and of software effort estimation as a specific work practice. In this paper we investigate the estimation practice through a detailed data analytical technique with machine learning approach.

Keywords: Effort Estimation, Group work, Interaction, Qualitative Analysis, and Organizational Strategies.

I. Introduction

Software development effort estimation is carried out more than once in the life of software project. The project acquisition stage is the one where the detail available to estimator is minimal. The confusion of estimation arises in this stage. All other stages, progressively, increase the detail and hence reduce uncertainty in estimation. Accurate estimation of software development effort is critical because the estimated figure drives the budget to be allocated, and if this is beyond allowable limits, the project itself might not be approved. Underestimates lead to time pressures to the developers that may negotiation full functional development and painstaking testing. In contrast, overestimates can result in noncompetitive contract bids and/or over allocation of development resources and personnel [1-5].

Without the proper data and experience, software development teams usually generate inaccurate estimates of the effort required for the product to be developed. As a result, the teams are required to renegotiate with the clients to ensure that the product to be developed is within the scope achievable by the development team. Without the necessary data, it is nearly impossible for teams to make proper predictions with respect to project scope, complexity, and resources required. Typically, projects progress through their life cycles based on these inaccurate estimates. This means that regardless of how well or poorly the projects progress, the estimates remain constant. When projects begin with the initial overestimation of resources or effort required, the teams must negotiate with the clients to reduce the size of the projects. This often results in clients needing to throw away some of the critical core capabilities of the product, thus losing some of the expected benefits they had hoped for from the completed project. The reality is that the team may actually have enough resources to deliver all of the initial requirements prior to the re-scoping of the project [6, 7].

On the other hand, when projects underestimate the resources, the teams tend to over promise the goals that the project can achieve. As the project progresses towards the end of its life cycle, the team may start to realize that the remainder of the project is more than they can manage to complete. When this happens, one scenario is that they try to satisfy the client by attempting to complete the project as quickly as possible, while the quality of the project may suffer greatly from this attempt and result in higher long-term maintenance costs. Another scenario is that they end up delivering a project that is not complete, thus leaving the clients with unusable or unsustainable products [8-11].

Most systems can be seen from two viewpoints, the user’s view and the developer’s view. The perspective of the user relates to what the system can do for the user and function size measure support the user’s perspective. The developer sees in terms of the internals of what needs to be built and is able to relate better to technical size measures [10].

When software development teams lack the proper data and experience, they can’t accurately assess project size and team capabilities. These unknowns and uncertainties can typically be reduced with proper assessments as the project progresses. Unfortunately, team assessments are often overlooked, even though personnel uncertainties often have significant influence on the cone of uncertainty. For most projects, estimates
during this phase are expected to be very rough estimates. Budget estimated figures could vary between +30 %
to -30% of actual numbers.

The main motivation behind this research is derived from the well-known software cone of uncertainty
and calibrated to completed projects [Barry Boehm]. Good team performance and accurate software cost and
schedule estimations are essential in determining the quality and timely delivery of the final product. According
to Standish report few projects were delivered with full capabilities within budget and schedule. Many projects
were cancelled, and were either over budget, over schedule, or under-delivered. These numbers are also
consistent. This figure shows that nearly half the projects were unsuccessful due to issues related to cost and
schedule estimations, and that software development projects have been consistent in replicating these shortfalls.

In software cost estimation, effort allocation is an important and usually challenging task for project
management. Due to the Cone of Uncertainty effect on overall effort estimation and lack of representative effort
distribution data, project managers often find it difficult to plan for staffing and other team resources. This often
leads to risky decisions to assign too few or too many people to complete software lifecycle activities. As a
result, projects with inaccurate resource allocation will generally experience serious schedule delay or cost
overrun. [12-16]

The Standish Group estimates that in 2013 the worldwide yearly spending for software projects was
$750 billion. The United States accounted for about 40% of this number or about $300 billion. Europe spent
about 25% or $200 billion. Asia accounted for $100 billion. The rest of the world spent the remaining $150
billion. Canceled or failed projects were 16% or $120 billion. The United States portion was a little higher and
the European portion was slightly lower. Challenged projects, those that were late, cost more and off-target,
were 48% or $360 billion. Overruns vary with many legitimate reasons, but The Standish Group estimates in
2013 the cost of unintended worldwide overruns is about $80 billion; leaving the cost of worldwide project
software failure to be about $200 billion. The 2009 Standish Report reported that out of the 9000 projects
surveyed, 32% were delivered with full capability within budget and schedule, 24% were cancelled, and 44% were either
over budget, over schedule, or under-delivered. These numbers are also consistent with the two
previous reports in 2004 and 2006 [Standish Group Report].

<table>
<thead>
<tr>
<th>Year (s)</th>
<th>Failed (%)</th>
<th>Challenged (%)</th>
<th>Succeeded (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013</td>
<td>16</td>
<td>48</td>
<td>36</td>
</tr>
<tr>
<td>2009</td>
<td>24</td>
<td>44</td>
<td>32</td>
</tr>
<tr>
<td>2006</td>
<td>19</td>
<td>46</td>
<td>35</td>
</tr>
<tr>
<td>2004</td>
<td>15</td>
<td>51</td>
<td>34</td>
</tr>
</tbody>
</table>

Successful projects did not return value to the organization or the users and executive sponsor were
unsatisfied. In addition to that many challenged projects bring great value to the organization.

![Figure 1. Standish Chaos Report on Software Projects Success, Failure and Challenged Ratio](image-url)
II. Software Product Development Effort Estimation is a Wicked Problem

Software Effort Estimation is a wicked problem as one that could be clearly defined only by solving it, or by solving part of it. This method assists software industries to estimate the required effort to be spent on various activities during requirements, architectural design, development, testing, deployment, operation and maintenance. We do not quote too less, programmers work for overnight that leads to lose the project or end doing social service, or loss. Do not quote too high that lose the project. So, be fair to ourselves and our customers. Hence, there is need to use of a repeatable, clearly defined and well understood software development process that has to be the most effective method. This paradox implies, essentially, that you have to solve the problem once in order to clearly define it and then solve it again to create a solution that works. There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies and the other is to make it so complicated that there are no obvious deficiencies. The phrase software design means the conception, invention, or contrivance of a scheme for turning a specification for a computer program into an operational program. Design is the activity that links requirements to coding and debugging. A good top-level design provides a structure that can safely contain multiple lower level designs. Good design is useful on small projects and indispensable on large projects. [17-30]

- It's easy to estimate what you know.
- It's hard to estimate what you know you don't know.
- It's very hard to estimate things that you don't know you don't know.

Figure 2. Similarities between Iceberg & Software Product

The Software Development Effort Estimation approach is just like the above one. Since it’s development is intangible unlike other product. The iceberg image to the tip represents the software size or its functionality. The real issue is not the tip, but what is under the surface of the water and cannot be seen. The same is true when you design a software application.

Figure 3. The Role Estimator: four images in one: sky, background, top iceberg, and underwater iceberg
The water was calm and the sun was almost directly overhead so that the diver was able to get into the water and take the picture. But how could anyone take such a picture? You could never see the underside of an iceberg that size in one shot - and where does all the light come from at that depth? In fact, the picture is not real. Ralph Clevenger digitally composed a nature and underwater photographer who find the stories circulating about his “impossible” picture amusing. Four separate images were used; the sky, the background, the top iceberg, and the underwater iceberg. The picture does, however, accurately represent the amount of an iceberg that is hidden underwater. It was designed to illustrate the concept of "what you see is not necessarily what you get". The project tracking mechanism allows for development teams to constantly monitor the resources required for the teams to complete their project. These estimated resources can be updated as necessary depending on the team’s productivity and capability. Software Project planning encompasses five major activities – estimation, scheduling, risk analysis, quality management planning, and change management planning. Estimation includes your attempt to determine how much money, effort, resources, and time it will take to build a specific software-based product. Software project managers using information solicited from project stakeholders and software metrics data collected from past projects.

III. Organizational Strategies & Social Interaction in Decision Making

Before taking the final decision data analytical techniques have been played a vital role. Since beginning no organization started their business only with service motto. No one will do the business with loss. So they need profit. Decision-making is choosing between alternatives while having incomplete / unreliable information about the scenario at hand and with uncertain and unpredictable outcomes of the available alternatives, mainly for the sake of expediency. Decisions are made in organizations to tide over the present situation / difficulty. Therefore, sometimes decisions may render injustice. One misunderstanding that is prevalent is that, the decisions are judgments – it is far from true. We can classify decision in to the following classes for our better understanding. [46-51]

Classification of Decisions
1. Strategic & Periodic Decisions
   a. Selection Decisions
      i. Products / Services  
      ii. Process  
      iii. Locations  
      iv. Layout  
      v. Equipment  
      vi. Workforce  
   b. Design Decisions
      i. Product design  
      ii. Service Design  
      iii. Job Design  
      iv. Process Design  
      v. Control System Design  
      vi. Capacity Design

2. Recurring Decisions
   a. Target Setting
   b. Scheduling
   c. Sequencing
   d. Inventory Control
   e. Cost Control
   f. Maintenance

3. Planning Decisions
   a. Planning the system
   b. Planning the usage of the system

4. Organizing Decisions
   a. Organization Structure
   b. Organizing the jobs
   c. Staffing
   d. Work and Workstation Design
   e. Standards of Performance
   f. Compensation Systems

5. Controlling Decisions
a. Quality  
b. Quantity  
c. Schedule  
d. Inventories  
e. Costs  
f. Maintenance

It is not necessary that all decision makers make all the above-mentioned decisions. All of us make some of those decisions. It is perhaps, very few people – especially entrepreneurs – make all the above-mentioned decisions. Besides the following things are to be considered for social interaction:

Relationship with Customers
- Satisfaction
- On time Delivery
- Accountability
- Top Priority
- Maintenance

Benefits from other Sectors
- Govt. / R & D support
- Financial
- Tax
- Business Expansion
- Recognition

Relationship with Employees
- Training / Job Satisfaction
- Partnership / Organization Hierarchy
- Policies / Timings / Importance
- Job Recognition / Security

Therefore before finalize the price of the product they need consider the above in different perceptive. At the same time they need to satisfy the stakeholder in order to promote the product. [52-62]

**IV. Methodology For Estimation With Machine Learning**

This article describes two methods of machine learning, which we use to build estimators of software development effort from historical data with Neuro-Fuzzy Expert System. In order to optimize the results data analytical techniques have been used [63-71]. First we need to find the size of the product then it is easy to find the required effort. Also capabilities of the team members in terms of project complexity and risk factors to be measured. After that we can focus on estimating the total cost of the product and as a result we can schedule the product [2, 4-7].

The process encompasses certain key ingredients that include
- Developing a good project execution plan
Organizational Strategies & Social Interaction Influence in Software Development Effort Estimation

- Converting the goal into a well-structured requirement specification
- Dividing the project into well-defined phases
- Allocating the right resources to the right job
- Working at the cost of project execution
- Delivering the project within the overall timeframe

Formal estimation models not tailored to a particular organization’s own context, may be very inaccurate. Use of own historical data is consequently crucial if one cannot be sure that the estimation model’s core relationships are based on similar project contexts [72-88].

Formulating the problem and fitting into Machine learning. Domain knowledge is more required in identifying the features in the form of requirements that can represent the data set. All the problems may not be easily expressed. Without proper understanding of distribution of the data, formulation of the problem, preprocessing, assumptions and presumptions of machine learning algorithms implementation is not that easy task. [89-101]

- In order to solve the estimation problem the following key points to be addressed:
  - Unavailability of data in a suitable format
  - Incomplete, noisy, and inconsistent data
  - Assessing the expected error of a learning algorithm on a problem
  - Data normalization and standardization
  - Relationships and correlations can be hidden within large amounts of data
  - Human expertise does not exist for all kinds of problems
  - Human designers often produce machines that do not work as desired in the environments in which they are not used.
  - The amount of knowledge available about certain tasks might be too large for explicit encoding by humans
  - Environments change over time.
  - New knowledge about tasks is constantly being discovered by humans. It may be difficult to continuously re-design systems “by hand”.
  - What algorithms are available for learning a concept? How well do they perform?
  - How much training data is sufficient to learn a concept with high confidence?
  - When is it useful to use prior knowledge?
  - What are best tasks for a system to learn?
  - What is the best way for a system to represent its knowledge?
  - How can we optimize the accuracy on future data points?
  - How can we formulate application problems as machine learning paradigms?

Figure 5. Model of Neuro-Fuzzy Expert Estimation System with Data Analytics tools
In Software Industry size estimation is a precursor to effort, schedule and cost estimation, the accuracy of these estimates depend on the accuracy of the size estimates.

- **Unfamiliar project**
  For the rest of this report, the definition of unfamiliar projects, or teams, means they either have little knowledge or little experience with the type of project being developed.
  This includes the following:
  - Inexperienced in general
  - Experienced, but in a new domain
  - Experienced, but using new technologies

- **Continuous assessment**
  Continuous assessment is a type of assessment methodology that takes place over a period of time. In the software development context, the process is done parallel to the development, instead of being done at major milestones or reviews.

- **Traditional development project**
  A type of project in which, the product must be developed from scratch.
  The development team must write the majority of the source code to implement the end user functionalities.
  A type of project that aims at integrating and/or tailoring either one, or a set of, non-developmental items or commercial off-the-shelf products.

![Figure 6. Optimized Estimation Method](image)

To use this method, one needs to be aware of the techniques but need not be an expert. What is needed is the knowledge of the software project or which they are making an estimate. This method facilitates making a new estimate thru copying an existing one and modifying it. This also facilitates making a new estimate from an existing estimate - thus improving estimation productivity greatly. Thus is a time and effort saving mechanism resulting in decreasing pressure on the valuable time of Project Managers / Leaders and leads to better productivity in general. It is important to point out that though producing high quality software requires more effort, this additional effort is more than recovered by lower maintenance effort and trouble [101].

The estimation process, model and technique that we decide to use should take into account all the below parameters appropriately:

- Availability and stability of the developer environment
- Team capability, skill and experience
- Team stability / manpower turnover
- Maturity of the processes
- Reusable software available to the project
- Reusable software to be build by the project
- Extent of communication possible with the user / customers
- Extent of automated tools used for software development and maintenance
- Extent and degree of detail of the required user documentation
- Cohesion of stakeholders and teams
V. Expected Outcome With Data Analytics

Sometimes there is a dominant factor that influences the decision-making. For example – for a mining company there is no alternative but to open it near the mine. A maritime ship liner needs to be near the seacoast. Location of market is another dominant factor. Other cases may include emotional factors of the entrepreneur – like his native place when it comes to opening his company or the expertise of the entrepreneur when it comes to selecting the product and so on. In day-to-day affairs, customer preference becomes a dominant factor, around which we have to manage. In some cases like Y2K, the time becomes the dominant factor. In some cases, the statutory obligations become the dominant factor. When a dominant factor is present in a decision scenario – the decision is made for us.

The software size is the most important factor that affects the software cost. Here the system will consider data analytical techniques to analyze the appropriate decision. The lines of code and function point are the most popular software size metrics used in practice. The cost to fix a defect rises dramatically as the time from when it’s introduced. These remains true whether the project is highly sequential or highly iterative. The main reason is here doing around five to ten percent of requirements gathering and design based on that [2, 4, 5].

With high objective the organizations may establish the service level agreements for resolving issues. The targets pertain to factors that will measure the effectiveness of the system [6].

Analytical Decision Making – this style implies that a thorough analysis is carried out in which all possible alternatives are considered along with their costs and possible results are analyzed and the optimal decision is selected. This is used by knowledgeable people and somewhat less experienced in their field. The scenarios that come to mind where this style is appropriate are –

a) Strategic decisions which have long term impact – especially selection and design decisions
b) There is time available for making the decision
Standard method for measuring software development from the customer’s point of view is always different. Quantifies functionality provided to the user based primarily on logical design. Measure software development and maintenance independently of technology used for implementation [6].

In this paper the author’s approach is not to criticize the existing popular estimation models. Most of the models are useful. Different models cater different needs. Existing models focused either on size or line of code. Actual required effort estimated by the existing models. Besides, organizational strategies and social interaction among the group members are the key attributes in software estimation. The accurate estimation immediately affects the success of project. Data analytical techniques will help the estimators to track the project status and take a correct measure in order to improve the quality of product. With the help of Machine Learning and Big Data Analytics tools & techniques optimization is possible. Further research will be continued with sufficient amount of industrial data with other parameters.

VI. Conclusion

References

Organizational Strategies & Social Interaction Influence in Software Development Effort Estimation


www.iJosjournals.org 38 | Page
Organizational Strategies & Social Interaction Influence in Software Development Effort Estimation


Organizational Strategies & Social Interaction Influence in Software Development Effort Estimation


