Science World Publication

Journal of Computational Science and Information Technology

journal homepage: https://www.sciworldpub.com/journal/jcsit


2019, Vol. 1, No. 1, 29-40

## Case Study

# Evidence-Based Software Cost Effort Estimation of Verification, Validation and Testing in Nepal

Gajendra Sharma

Department of Computer Science and Engineering, Kathmandu University, Dhulikhel, Kavre, Nepal

**Abstract:** Software development projects tend to be based on over-optimistic cost estimates. Better knowledge about software cost estimation is necessary to improve realism in software development project bids and budgets. A case study was conducted and collected empirical evidence from software development companies in Nepal. The minimum company size was 30 while the maximum company size was 200. The case study was performed by conducting interviews with a set of structured questionnaires. The results were compared obtained from the case study with the literature review and found that there exists practice for empirical evidence based verification, validation and testing cost/effort estimations. It was noted that test effort estimation follow the same pattern as software development project estimates. The results show that all the companies prepare separate estimates for test effort, empirical data is commonly used to estimate test effort and test effort estimation error seems to be closely correlated with development effort estimation error. A company that had estimated total of 3500 man-months had actually spent 4200 man-months implying 700 man-months of effort/cost overruns to complete the project. Another company that projected testing effort of 100 man-hour actually ended up in 120 man-hour at the end of project causing 20 man-hour effort/cost overruns.

**Key words:** Software Test Effort Estimation, Testing in Nepal, Verification and Validation, Empirical Evidence

## 1. INTRODUCTION

Better knowledge about software cost estimation is necessary to improve realism in software development project bids and budgets. The objective of this article is to systematically analyzes the vast pool of papers, innovations, and improvements in evidence based estimation of software cost effort in verification, validation and testing.

Evidence Based Software Engineering (EBSE) is scientific research conducted at real industrial setting to gather actual data to analyze the prospects of outcome as a result of the study. Random controlled experiments are also evidence based although they not necessarily depict the actual practice scenario. Software Cost/Effort Estimation is a vital activity in software development projects that allows developers, and managers to forecast, predict, and accurately quote the budget, schedule, and manpower effectively to save from overruns or underruns thereby attempting to optimize the crucial factors leading to project success. The effort and cost estimations in software development have evolved since 1950s and continually being researched as seen from papers by Jørgensen and Grimstad [20].

The objectives of the study focuses on finding out and specifying International EBSE for V&V so that the comparisons could be made with the research outcome of investigation into testing estimation practices being followed by software

**Received:** Feb-08-2020 **Accepted:** Feb-27-2020 **Published:** Mar-14-2020

**Corresponding Author:** Gajendra Sharma, Department of Computer Science and Engineering, Kathmandu University, Dhulikhel, Kavre, Nepal, Email: gajendra.sharma@ku.edu.np



**How to Cite**: Gajendra Sharma, 2020. Evidence-Based Software Cost Effort Estimation of Verification, Validation and Testing in Nepal. *Journal of Computational Science and Information Technology*,1: 29-40.

companies in Nepal. Different papers related to the title and relevant to the intended results are compared to empirical evidence based data from software industries participating in the research so that the actual facts and figures pertaining in recent testing scenarios in software industries of Nepal are analyzed versus the international research, study, and practices. The results yielded from the studies can set a new example to rest of the researchers, scholars, students and industries involved in software development.

The purpose of this study is to provide results on software test effort estimation using empirical evidence among selected software companies of Nepal, and, if they adhere to international norms and practices as seen from literature review outcomes. This research is intended to cross-study relevant papers that constitute verification, validation, and testing methodologies, estimations, models, and algorithms. The EBSE aids the study to analyze the real time software industry data compared to other simulations, lab data or randomly controlled experiments.

### 1.1. Evidence Based Software Engineering

Evidence Based Software Engineering is scientific research conducted at real industrial setting to gather actual data to analyze the prospects of outcome as a result of the study. Random controlled experiments are also evidence based although they do not necessarily depict the actual practice scenario. Synthesis and valuation of evidence is a complex task describes [34] comparing medical practices to Evidence Based Software Engineering (EBSE) pioneered by team of Dyba *et al*. [8] further illustrates types of evidences, validity and current trends. Systematic mapping studies guidelines are proposed by Petersen *et al*. [29] with visualizations, maps and characteristics using evidence based software engineering practices referencing estimation models presented by Grimstad *et al*. [13]. Only seven Systematic Literature Reviews (SLRs) is found in 2009 paper by Kitchenham *et al*. [24] that regards cost estimation out of 20 relevant studies made using EBSE approach.

### 1.2. Software Cost Effort Estimation

The effort and cost estimations in software development has evolved since 1950s as addressed by Boehm [7] taking into account different companies, models, quantitative analysis and complexity of software systems. This study claims that the ageing software development practices are too risky to repeat and exemplifies a Standish Group CHAOS Report (http://standishgroup.com ) that characterizes the fact of non-repetition of even success because this field in constantly changing. However, the Standish Group Report is strongly criticized by Jørgensen and Grimstad [19] reasoning that estimation biases due to projects sample and erroneous survey method led to huge estimation error and cost overruns that is misleading, inaccurate and incorrect.

There are reports of 30-40% effort estimation error during software development while up to 191% cost overruns occurred in Standish Group Report [18]. Summarizing that random field controlled experiments did not impact real-world estimation much illustrating regression models for accurate estimations and providing public trove of all papers in software estimation research domain http://simula.no/BESTweb that is used by 100s of researchers worldwide.

### 1.3. COCOMO / COCOMO II Cost Effort Estimation Practices

COnstructive Cost MOdel (COCOMO) was first coined in 1981 published Software Engineering Economics book by Barry Boehm that used Line of Codes (LOC) as basic parameter used in estimating software development efforts or costs. Developers' net productivity and assessment of effort has been investigated using evidence based software engineering [17] citing the need for improving average developer productivity and reducing gap between slowest and fastest programmer using COCOMO/COCOMO II effort estimation models. The COCOMO suite of models studied for last forty years [7] depicts non-linear formal model of software size and illustrated data from over 8000 software development projects in 350 companies suggesting that estimation errors can be reduced by adopting agility and improved productivity over time.

The present study has however focused on own COCOMO models only and has not referenced pioneering software estimation approaches, expert-judgment and other evidence based software engineering approaches.

### 1.4. Expert Judgment versus Formal models

Expert Judgment is a common practice seen in industries and published work by several researchers since decades that rely on expert thinking, experience, knowledge, wisdom, and analytical capabilities. Other line of thought relies on formal mathematical, experimental and simulation models not making use of expert opinion or experience. Relying on expert based opinions in selecting software engineering technologies instead of applying evidence based experimental software engineering is a common scene asserts that Action Research can improve the quality and create a win-win situation between the researcher and the organization citing refactoring scientific guidelines for developer codes [31].

There is however a debate on expert-judgment versus formal models and it is not intended to show superiority of one approach over other but to guide researchers to adopt mixed approach as much as possible to improve the accuracy, correctness and predictability of software development estimation practices. Baker and Daniel [4] discusses in depth the applicability of expert judgment over formal models when predicting estimation tasks by implementing 7 of 12 best estimation practices in software models recommended by Jørgensen *et al*. in more than 5 papers cited in the thesis and taking NASA as industry sample [20].

### 1.5. Agile Based Software Development Practices

Waterfall model of software development ruled from the beginning and are not going to vanish where iterative set of development can occur after some big finite delivery and the process is repeated after identifying issues, errors, bugs or failures from beginning of each cycle. It is useful in long term large software projects. Agile methodology adopts short term deliveries, also known as sprints which is weeklong or more cycle to make small deliveries which is constantly analyzed to uncover errors and uses burndown chart to prioritize and solve user stories derived from requirements. Agile methodology uses commonly Delphi based group participation in assigning priority to user stories or using planning poker cards to estimate cost or effort for given user stories repeatedly performing until consensus is reached in the team. Agile software development is becoming increasingly popular in past and current decade and there are 36 empirical evidence-based studies conducted to determine the limitations and benefits of agile methods as claimed by Dyba and Torgeir [9]. Tamrakar and Magne described an agile planning poker estimation improvements using Fibonacci instead of linear scale [33] conducted two empirical studies to report that Fibonacci scale based estimates impact lower than linear scale based estimates due to bias in middle of the values.

### 1.6. Verification, Validation and Testing Cost Effort Estimations

The V&V testing cost effort estimations is not new in research or industry or among academicians but its presumed as not widespread as overall software development projects estimations. Verification is software output conforming to user requirements while Validation is design, codes and implementation conforming to expected outputs commonly known as V&V which encompasses commonly used term Testing that is an essential activity for all phases and processes of software development. Therefore, V&V testing also needs to be estimated for cost and effort to save a software development project from overruns and underruns. Size and cost estimation is one of the primary tasks in software development to achieve quality assurance goals as mentioned by Petersen and Claes [30] in a study consisting of coverage of research facets considering objects of investigation during contextual empirical industrial studies but lack varieties of discussions on given topic.

The effort cost estimation models on testing using functional and non-functional requirements function size metrics based on empirical quantitative evidence related to European Aerospace Industry. The paper illustrates measurement of empirical data from 15 software projects linear regression models and using 16 functional and other non-functional requirements. Predictable fault proneness, early estimation, cost savings, and high quality achievement is contributed to best available software quality estimation model techniques derived from three NASA projects by creating a fusion metric model combined of requirement and static codes [21]. Several case studies, observations, simulations and experiments claim that 50% of the software development schedule is spent on verification, validation and testing activities ensuring quality citing empirical evidence from eleven Swedish software companies so that efficient fault detection and estimations can be achieved displaying calibration and validation.

Case based reasoning model Estor was found to be as accurate as an expert and better than function point or COCOMO estimation models addressing the fact that underestimation leads to inflated impressions, omissions, abortions, cost overruns, incompletion and unreliable deliveries, while, overestimation leads to cost increment, reduced developer productivity and missed opportunity by presenting empirical evidence from expert generated 15 software estimation tasks. Requirements are treated as independent variables assert remarking dependent variables as projects, products, company and society embracing the importance of requirements engineering to ensure software quality and success as a whole [11]. A review rate of 200 LOC/hour or less is best suited for effective defect removal in design and codes quantifying developer ability and process variables using personal software process approach, [22] as a result of two sets of 371 and 246 programs using regression and mixed models that contribute to higher quality of software. Verification, validation and testing activities are technically and cognitively more challenging [26] as a research outcome on self-construal cognitive biases by Jørgensen and Stein [20] affecting overall software development practices in industries.

Larson stresses on estimation of testing tasks to be unbiased when making judgments with some case studies presented from Jet Propulsion Laboratory [26]. It's a tedious task to prioritize requirements, a major contributor affecting software quality, using fuzzy logic and software agents to increase test effectiveness [28], and fault detection rate to increase probability of accuracy of quality testing activities. Cost sensitive active learning strategy is developed by Sun *et al*. [32] to prepare an accurate and reliable estimate of software testing costs, to reduce risks by prioritizing, labeling, profiling, and weighting severity levels of failures/defects, and to compare executed versus non-executed tests.

### 1.7. Improving Software Quality: Verification, Validation, and Testing Approaches

Overall software quality is measured by output evaluation, client feedback, impression and actual task fulfillment which is improved by implementing V&V Testing correctly using different approaches and practices made by industries. IEEE Fifth International Conference on Software Testing Verification and Validation 2012 focused on models, fault localizations, database/GUI testing, constraint solving, search-based testing, web-applications, test evolution, domain-specific testing, white-box techniques, state-based testing, empirical studies, failure analysis, case studies, analysis and validation, test automation and PHD Symposium, but could not see any research paper committed to software cost effort estimation on testing.

Verification x is transformational accuracy of x (creating right x), and Validation x is behavioral/representational accuracy of x (creating x right) implied by Balci [5] providing 20 golden rules in the research which is useful for high quality software development. Predicting software defects using COCOMO empirical model has been presented by Bezerra *et al*. [6] using training datasets from NASA, IV&V, MDP that predict defective modules using neural networks and shows accuracy of effort/cost predictions for development and testing. Symbolic regression in genetic programming is a viable software fault prediction and quality estimation technique using search based software engineering algorithms and for determining non-functional system properties verification, validation and testing providing insight into segmented, dynamic and overall estimation of product quality [1].

Afzal also demonstrated fault counts prediction, mean time between failure reduction, and error estimation accuracy by collecting software development data from large telecommunication company projects and other software development companies. To increase estimation effectiveness, experiments have been conducted in MATLAB so estimation and predictability of test outcomes due to fault occurrence has more reliability and proximity to correctness. The present study also include methods for finding bugs, defect, faults, and errors early to save cost and effort of development and fixing tasks which can grow exponentially if bugs are caught later that can ruin the project as well as the software company.

## 2. METHODOLOGY

It was intended to use the research methodologies during course of this study of software development market in testing arena. So, the following research methods were used to attain the research objectives to answer What, How, and Why.

Kvale is the most cited and renowned author on qualitative research, interviews, dialogues, surveys , and in page xi of his book , he has clearly demonstrated the value, significance and necessity of conducting qualitative research [25].

The survey was conducted among 5 software development companies in Nepal participated by CEOs, Directors, Managers, developers and testers. The time period was 2012/2013 when the companies were requested to help with data using phone calls, emails and even meeting personally. They were entrusted that we do not divulge their identity or produce the evidence to anyone else rather than using the data for our research. The questionnaires were prepared to interview the Companies so they can fill it easily after half an hour to one hour of discussions.

### 2.1. Interviews

The interviews were conducted by meeting personally, using emails, and phone calls. It took months to communicate (using Email, Phone calls, LinkedIn/Facebook messages) with companies, get time for case study from CEOs, their Project Managers, Test Managers and Technical leads who would analyze the questionnaire and fill it making careful calculations so accuracy and correctness is reflected in the survey data they fill. All the five selected companies are registered in Nepal and are multinational outsourcing hubs although they kept much of client side confidential, however, filled client side data too.

The different segments of clients display varying software development and estimation models. One of the companies has six branches internationally including Korea, while others have offices in USA, Denmark and Japan.

The selected companies reside within 10 km radius and

most of Nepal's software development industry is concentrated in the area. It was not possible to conduct case study of companies located outside of Kathmandu Valley assuming they do not have much quality initiative and so do not spend much efforts on testing to save costs; however, testing efforts saves cost and improve quality.

Tables 1, 2 and 3 were created after respondent's data and helped us in reaching decision that the 5 software companies of Nepal are not lagged behind in producing top quality software like international companies, and use testing estimations like project estimations giving equal importance to testing like to software development project.

## 3. RESULTS

In order to achieve answers to the 2 research questions, 7 questions were designed for company background, 9 questions for estimation /test methods, and 12 questions for last completed projects data with total of 28 questions that could provide insight into the research questions and contributing factors for the study.

About 100 software development companies of Nepal were approached for widespread surveys listed in everestlist.org, because the response rates were low, follow-up interviews with five companies were conducted which were willing to participate in Case Study. Nepal is a poor south asian country sandwiched between two big superpowers/ big economies of world India and China, so having big (manpower greater than 10) software companies is a dream and those having separate testing units is further an expensive desire only because of myths that testing increases cost prevails among software companies of Nepal.

It could be surmised from experience of 13 years with Nepalese software industry that most of the companies in Nepal do not have formal testing procedures so they would not be able to answer the questionnaire, and further, many companies do not prepare project and test estimates separately.

An empirical claim can be made that the case study of 5 companies with 80+30+55+31+200 totaling 396 software development team represents 20% of legal, big sample of companies involved in case study are among 100 top software companies of Nepal because there is no other empirical evidence to corroborate the fact from private or public or any government statistics.

Also, if there exist top 10 companies employing 1000 software developers matching the quality of the 5 companies selected for this research, the developer size covered by this study covers about 39.6% i.e., almost 40% of overall population working in top software companies of Nepal. This case study conducted across 5 companies shows that one of the smallest companies has employee count of at least 30 while at most 200 plus software development full time employees are seen in one of the largest software companies in Nepal.

There is however one foutth that is only 25% manpower in testing. Among 5 companies involved in case study, apart from company five other smaller companies agreed to enhance and boost testing units. The only site ranking top 100 software companies of Nepal everestlist.org claims to conduct the research among top software companies from 75 districts, however accepting that there are some other hidden software development companies not in limelight as the practice is concerned, which may have been excluded from the list.

The impression is that many software development companies fear for the data they are giving because research culture is limited in Nepal. The questions were designed for aiding empirical evidence of project/testing effort estimation models and to assess the affecting parameters to this study.

The questionnaire was prepared in English language and the companies were expected to provide effort cost estimation models data matching the understanding by their education, knowledge, experience and expertise in software effort cost estimation calculations.

Nepal is a small country and this sample size of companies with minimum approximately 30 implies big company with good revenue and experience in software development. There are only a few handful companies that have 100 plus employees. The 5 companies those agreed to participate for case studies were questioned based on their motivation to quality centric approach, software type, experience, company size, client size, project size, resources, team size, and clients they are serving.

So, using the experience, knowledge, education, and technical skills in empirical evidence, software cost effort estimation, and testing, the questionnaire was segregated into three parts, namely: Company Background, Estimation and Test Methods, and data from last completed project for the companies to be selected for case studies.

### 3.1. Case Study Results

After receiving feedback from the five selected case study respondent companies, each of the research questionnaires was analyzed as items described below:

In Table 1, the data can be seen that they cover almost entire common software development industry, maturity, experience, coverage, strength, and self-ranking and are

the empirical evidence for their quality sensitive approach.

One of the companies C is 25 years old (The CEO is also a professor of computer science) and is a remarkable existence in country like Nepal to survive that long as markets here are not strong and long-lasting local or foreign clients are rare. The company E with 200 employees is from healthcare domain in USA and boasts being one of the best job-giver, best-quality software developing company establishing own computer science college to produce required manpower, so dearth of good quality software developers in Nepal can be met easily, and competition to acquire qualified developers can be dealt with. Company A boasts existing in 6 countries and deals with varieties of clients locally and globally.

Company B claims that their application is global and so quality is imperative for their Japanese client who outsources them software projects. Company D definitely is leading error-free services to their Denmark and other European clients because of the sensitive security based software that is critical for millions of people. Company E plans to employ as much as 1000 software developers in Nepal software development center. Company B is too young they said and still perspiring to lead the industry with quality initiative with best quality outcomes.

They however have mature CEO who constantly guides them, and monitors the activities during software development and delivery. Among them, Company D pays the best salary to its software developers as investigated using inside sources and they were reluctant to reveal their revenue and pay scales. In Nepal, the culture of tax evasion by software companies is reported by experienced experts who claim that many do not participate in case studies and surveys in Nepal fear being identified or marked lest their data reaches tax office. From their own self-ranking, it can be surmised that they are over-estimating themselves as being best as there is no tangible ranking empirical evidence that can definitely point and say: this software company is best compared to other.

They might also have got hindsight of competition with other software companies. Companies were in liberty to fill in their own way so differences in filled data sets can be seen, say for company size or project size according to their own interpretation.

Total of 396 developers can be seen when summing all of them and it should represent 30% of Nepal's software development industry although there is no formal or informal empirical evidence to claim the fact, but expert-opinion could make the prediction about overall Nepal's software sizing estimation possible. The CEO of Company E has software

Table 1. Company Background

| Questions | A | B | C | D | E |
|---|---|---|---|---|---|
| When was your company established | 2006 | 2011 | 1998 | 2008 | 2008 |
| What is the size of your company | 80 | 30 | 55 | 31 | 200 |
| What kind of businesses does your company support | Mobile Technology and Mobile based value added services | Multi-channel integrations | Bank, Financial Institutions, Capital Markets, Hospitals, Cooperatives, Industries, Government, NGO | Software Security Solutions known as Security Information and Event Management (SIEM) Solutions software | US Health Care management |
| What are the average sizes of the projects your company handles | 4 man-months | 6-8 man-months | Medium-large (5 man-months) | 2 man-months | 15 man-months |
| How many types of programming languages does your company use | 4 | 3 | 5 | 6 | 2 |
| How many types of platforms does your company support | 5 | 6 | 6 | 3 | 2 |
| How do you rank your company compared with others in 1-5 scale (5 being best) | 4 | 4 | 5 | 5 | 4 |

Table 2. Estimation and Test methods

| Questions | Choices | A | B | C | D | E |
|---|---|---|---|---|---|---|
| What kind of Model does your organization follow | A. Waterfall B. RUP (Rational Unified Process) C. Agile D. Others | B, C, D (own based on iterative incremental) | Some are water fall & some are agile | Water fall, Iterative | Agile SCRUM | Agile |
| Do you prepare a Project Plan | A. Y B. N | A | Yes | Yes | Yes | Y |
| Do you prepare an estimate for the project | A. Y B. N | A | Yes | Yes | Both Water fall and Agile SCRUM Based Estimation, Water fall: Size & Tasks, Time Boxing for Headline (aka Epic), User Stories and Tasks | Y |
| How do you prepare a project estimate | A. Expert-Judgment B. Estimation model C. Others | A | A, B & some are research based | A. Expert Judgment | Estimation Model: Tasks Based and Time Boxing for User Stories (using estimations based on history data), Expert Judgment | Expert Judgment |
| In the estimation work, did you use historical data (data from previous projects) | No choices were provided but free text was expected | To some extent based on previous projects and expert judgment | Y | Yes we do use | Yes (previous releases) | Yes |
| Which levels of testing did you use | A. Regression Testing B. Unit Testing C. Others | Functional testing, SIT, UAT | Mixed | All A Regression Testing B. Unit Testing, Functional Testing | Unit Testing (by Developer), Integration Testing (by Developer), System Testing (by QA), Regression Testing (by QA), Manual, Automation: Siesta Framework (for UI) and, Automation: Robot Framework (for Engine level acceptance tests), User Acceptance Testing (by QA and Denmark Team) | Regression, Unit, integration, system test, performance test, security test, smoke test |
| What methods did you implement to conduct testing | A Manual B Automated C Others | A | Mixed | Manual | A, B | A, B |
| Can you produce the evidence of those artifacts | A. Test Plan B. Test Case C. Test Script D. Test Data E. Test Report F. Others | A, B, D, E evidence in PM tool (Trac) | Email and Documents | Yes | Yes for Test Plan, Test Cases, Test Script, Test Data and Test Report | A, B, C, D, E |
| How Testing efforts are estimated | A. Time Boxing B. Percentage of development estimates C. Expert estimate D. No separate estimates for Testing E. Others | C and D | C | C | A, B, C (sometimes) | C |

Table 3. Data from last completed projects

| Questions | A | B | C | D | E |
|---|---|---|---|---|---|
| How many types of clients do your projects serve | 8 | 4 | Government, International, local, public, Private businesses, doctors, managers, officers etc 10+ | Finance, Healthcare, Government, Telecom & ISPs, Defense & Aerospace, Utilities and Energy (SCADA) 7+ | 22 |
| What is the average size of your clients that projects serve | Medium | 1000 | Medium-large | Medium | 5 |
| How many local clients do your projects serve | 100 | 0 | Almost all | 0 | 2 |
| How many foreign clients do your projects serve | 2 | 50 | 3 | 300 | 30 |
| Total number of Estimated Effort for your project | 500 man-hour | 12 man-months | 3500 man-months | Major Release (Enhancement): Approximately 8 - 12 months, Minor Release (Maintenance): Approximately 3 - 6 months | 160 hours |
| Total number of Estimated Effort for the Testing activities | 100 man-hour | 2 man-months | 1800 man-months | Major Release (Enhancement): Approximately 6 months, Minor Release (Maintenance): Approximately 1 month | 80 hours |
| Total number of Actual Effort for your project | 600 man-hour | 15 man-months | 4200 man-months | Major Release (Enhancement): Approximately 8 - 12 months, Minor Release (Maintenance): Approximately 3 - 6 months | 200 hours |
| Total number of Actual Effort for the Testing activities | 120 man-hour | 3 man-months | 2500 man-months | Major Release (Enhancement): Approximately 6 months, Minor Release (Maintenance): Approximately 1 month | 100 hours |
| Estimation Error for the Project | 16.66667 | 37.66234 | 16.66667 | 0 | 20 |
| Estimation Error for the Testing | 16.66667 | 33.33333 | 28 | 0 | 20 |
| Reason of Estimation Error for the project | Change in Requirement, Delayed feedback from Client | Client delays, inconsistencies in requirements | Requirement not proper, lack of system knowledge | N/A | Project complexity and unseen problems |
| Reason of Estimation Error for the Testing | Change in Requirement, Delayed feedback from client | Client delays, inconsistencies in requirements | Requirement documents not in hand before QA and not clear sometimes | N/A | Error in development estimates |

development experience since 1980s holding PHD in Economics while company D and A have younger CEOs of 2000s holding graduate level degrees.

Table 2 shows the strength of test estimations in software development companies under case study. Definitely, as anticipated, Expert-Judgment leads across all companies although alternative estimation practices have been indicated. Similarly, Agile based software development model establishes as leader. Company D has better details on preparing estimation using own tool.

All companies claim using historical empirical evidence for estimation tasks. Company D has special automation testing levels using own framework, while Company E adds security testing as key feature compared to others. Automated testing is on rise as 3 companies claim using it. Those companies using manual testing methods claim automation is huge task in itself to implement and are considering adopting. Company B is not indicating any tool to keep test evidence but email and documents.

Only one company D has knowledge of using Time Boxing effort estimation, while others had not heard the term too, they confessed during case study interviews. Again, Expert-estimates lead across estimating tasks among companies and while averaging 25 year old expert knowledge, Company C took one week to fill the averages because they wanted to provide accurate and correct estimation data. Unit testing and regression testing are commonly followed as seen from the evidence presented by the companies.

Only one Company A knew about RUP (Rational Unified Process) software development model and rest were unaware of it. All companies prepare project plans and estimates which can be seen as a good practice.

Table 3 provides insight into last completed projects of the 5 companies who participated in case study display effort estimation by expert-judgment in development as well as testing efforts both estimation errors of approximating to 20% to complete the software development projects. There might be lack of understanding and knowledge to segregate client types in some cases that impacts their estimation errors.

All the respondents involved in case studies claim that clients definitely are major cause of project schedule and cost motivators that has been extensively researched by Grimstad *et al*. [12] based on literature review and survey of 300 software professionals. Company C and D clearly reflected the type of clients they serve by making high quality deliveries while others stated counts by keeping details confident.

Clearly, Company D develops software for critical components of society where quality is a must and cannot be compromised at any cost. In filling clients size data, the companies underestimate so write medium mostly as they must be confused on estimating how large is a big client and how small is another provided some clients may not be transparent and proper knowledge in assessing clients size is lacking that effects estimation accuracy and correctness.

It may be a need to assert rules for proper client sizing in scenarios like, developing a software module for Microsoft or Google and how to put client size then, that would definitely impact all estimates. Two companies B and D reported 0 local clients that shows their dependency on foreign outsourcing opportunities while displays their competence, quality initiative and client happiness.

Actually Nepal's software market is poorly entangled in clients not being able to pay, unmotivated for software solutions and looking for penny wise pound foolish solutions from lone consultants where quality is not guaranteed, but solves purpose of most of the market. Company E is unable to get millions from a hospital because the hospital cannot pay, although uses their software system while Company C has 100s of software in the shelf as local market is not able to grasp the trove of software. Company A faces tough competition in mobile based and other web based software share, but is leading the market since many years. In one case, even some foreign clients ran away without paying after using millions of dollars worth software. Company C has mentioned "almost all" for local clients and really they are a well-established brand since last 25 years serving almost all arenas by providing software and winning almost all bidding. Regarding serving foreign clients, Company D can be seen leading although other companies who put smaller numbers there confided that they serve more but cannot estimate properly due to complexities.

Company E could not get data from all projects due to busy schedule of project managers, product owners, so filled only 30 from average of one project only. Estimated projects/testing compared to actual values show varieties in filling the same thing: man-hour, man-months, hours, months and it was not thought to be a good idea to pressure them to fill it making suitable for same unit project estimation but converting them later into same estimation error calculations. Company D is of interest particularly because they operate differently with their clients and their estimates are time based, and there is no chance of error for criticality of their clients or, their billing process with clients depends upon month based major and minor releases, so they always meet the deadline.

It however poses risks because they must be estimating using other parameters too instead of relying on release

dates only and assuming 0% error in estimation because it is not possible to obtain 0% error as the empirical model would not support that. Another interesting study is made on companies A, C where both have same value of Project estimation errors 16.67. However, Company C has poorer estimation error 28% compared to 16.67 of Company A. Real-world evidence collected from experts in quality software development in Nepal would agree with the data filled here because of the answers filled on testing methods logic, and evidences.

Company B suffers from heavy project and testing errors 37.66% and 33.33% respectively, in estimation because of their clients' random needs, ideas and quality requirements that is constantly new and experience does not work because of clients. They confided estimation errors up to 200% when fulfilling wild dreams of clients.

Company E demonstrates 20% error in estimation which may be realistic depending upon their business success, software development growth, and huge company size, or, they might have underestimated fearing not being able to assess all the parameters properly for empirical evidence based computations for projects and testing. Inside Sources say, Company E has dedicated experienced, mature 25% of total employees in verification, validation, and testing that proves its commitment to superior quality of deliveries to clients.

Regarding cause of projects/testing estimation errors, companies A, B, C blame clients related issues while, D has nothing to say and E pointed to project complexity, unforeseen factors and error in estimation itself.

## 4. DISCUSSION

From the case studies conducted on five software companies of Nepal, it was found that software verification validation and testing cost effort estimation using empirical evidence is in practice and perceived as contributing factor in providing quality outcomes by preventing budget/schedule underruns or overruns. The results can be justified to be of international standards when comparing to works of Asztalos *et al*. on Formal Verification [2], on Empirical Research [3], on Golden rules of VV&T, [5] on Software Defect Prediction [6], on Sound Empirical Evidence in Testing [10], on Software Effort Estimation terminology [13], on specifying validating verifying requirements [15], on designing software effort estimation [16], on Software Cost Estimation [27], on Software Development Estimation Biases [20], on Predicting Top Crashes [23].

Expert opinion is regarded as the major method for estimating software cost effort apart from the empirical evidence present across projects claimed by all 5 companies that participated in research [14,34].

**4.1. RQ1: Do Companies collect and use historical empirical data for the purpose of estimation of V&V effort?** Yes, all the five companies claim to collect and use historical empirical data for estimating V&V effort although they admitted expert opinion as the key factor.

**4.2. RQ2: What are the current practices in effort and cost estimation?**

The companies involved in the case study use agile and waterfall models both depending on the nature of projects while all claimed to use agile only one company A claimed to use Rational Unified Process (RUP).

All companies claimed preparing project plans and estimates. All companies prepare project estimates primarily using expert-judgment and Company B claimed using other research estimation models also not directly specifying which one. Only Company D claimed using time boxing model. All companies claimed using historical empirical evidence not disclosing how they used it.

All five companies claimed they use regression testing unit testing and others. Companies A and C use manual testing method while companies B, D, and E claimed using manual as well as automated testing. V&V testing artifacts like Test Plans, Test Scripts, Test Data, Test Reports are claimed to be prepared by all companies as evidence while Company A claims using Project Management tool and Company B claims keeping evidence in email/documents specifically.

All companies replied that they need to invest more on research in improving current V&V effort estimation trends continually and by studying constantly the research papers to increasing the awareness, and knowledge among team members.

The result motivates fellow researchers, academicians, and industries to follow the practice to improve their estimation accuracy and predictability in testing. So, the society can benefit with better quality software produced by better estimation practices and accuracy. The management on the other hand can benefit by avoiding software cost effort overruns and underruns meeting deadline of client with lowered resources and better quality delivery. Research Scholars can widen the study to get more evidence based testing estimation practices or use artificial intelligence to analyze the data to improve the testing accuracy and predictions.

The research was conducted within narrow geographic location in a small country involving a small number of developers compared to big countries with thousands of developers working for single software product. The variety of projects was not specifically mentioned by companies involved in the study except some top level domain. The research does not have weak and ailing companies participating which could be vital in collecting evidence for failure trends due to lack of quality centric standards. The companies may have their biases while answering the questions.

## 5. CONCLUSION

It can be concluded that the reviewed 50 papers out of 150 selected papers spanning 30 year time period fail to provide conclusive evidence of research in evidence based software cost effort estimation for V&V Testing that indicates more effort on the education and knowledge is expected to spread the research arena.

The papers that were studied focused on quality of software development but attributed less to cost effort estimation of V&V testing using empirical evidence. Therefore, the case study conducted on 5 companies in Nepal helped assess the industrial scenario and found that practically the companies are estimating V&V Testing cost efforts using empirical evidence and expert-judgment based models. The main findings are summarized on the case study by following observations:

- Test effort is similarly calculated as Total Project effort estimation using expert-judgment
- The estimation error of Testing effort seems to correlate closely to the estimation error of Total Project

However, the companies can still improve the cost effort estimation errors of V&V Testing and projects by detailed analysis of their process by taking help from consultants. More research is requested in diverse demographics, culture, client base, application type and clients.

## 6. REFERENCES

[1] Afzal, W., 2009. Search-Based approaches to software fault prediction and software testing. Blekinge Institute of Technology, Karlskrona, Sweden.

[2] Asztalos, M., L. L´aszl´o and L. Tiham´er, 2010. Towards automated, formal verification of model transformations. 2010 Third International Conference on Software Testing, Verification and Validation, 6-10, April, 2010 IEEE, 15-24.

[3] Bai, X., Z. He and H. LiGuo, 2011. Empirical research in software process Modeling: A systematic literature review. Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on. Banff, AB, IEEE, 339-342.

[4] Baker and R. Daniel, 2007. A Hybrid Approach to Expert and Model Based Effort Estimation. West Virginia University, Morgantown, West Virginia.

[5] Balci, O., 2010. Golden Rules of Verification, Validation, Testing, and Certification of Modeling and Simulation Applications. Department of Computer Science Virginia Polytechnic Institute and State University (Virginia Tech) Blacksburg, Virginia 24061, U.S.A.

[6] Bezerra, M.E.R., L.I.O. Adriano and J.L.A. Paulo, 2011. Predicting software Defects: A cost-sensitive approach. 2011 IEEE International Conference on Systems, Man, and Cybernetics, 9-12, Oct, 2011, IEEE, 2515-2522.

[7] Boehm, B.W. and V. Ricardo, 2008. Achievements and challenges in cocomo-based software resource estimation. IEEE Software, 2: 74-83.

[8] Dyba, T., B.A. Kitchenham and J. Magne, 2005. Evidence-Based software engineering for practitioners. IEEE Software, 22: 58-65.

[9] Dyba°, T. and D. Torgeir, 2008. Empirical studies of agile software development: A systematic review. Inf. Software Technol., 50: 833-859.

[10] Fraser, G. and A. Andrea, 2011. Sound empirical evidence in software testing. 2012 34th International Conference on Software Engineering, 2-9, June, 2012, IEEE,

[11] Gorschek, T. and M.D. Alan, 2008. Requirements engineering: In search of the dependent variables. Inf. Software Technol., 50: 67-75.

[12] Grimstad, S., J. Magne and M. Kjetil, 2005. The clients' impact on effort estimation accuracy in software development projects. 11th IEEE International Symposium, 19-22, Sept, 2005, IEEE, -

[13] Grimstad, S., J. Magne and M. Kjetil, 2006. Software effort estimation terminology: The tower of Babel. Inf. Software Technol., 48: 302-310.

[14] Grimstad, S. and M. Jørgensen, 2009. Preliminary study of sequence effects in judgment-based software development work-effort estimation. IET Software, 3: 435-441.

[15] Heitmeyer, C.L., 2007. Formal methods for specifying, validating, and verifying requirements. J. Universal Comput. Sci., 13: 607-618.

[16] Idri, A., Z. Abdelali and Z. Azeddine, 2010. Design of radial basis function neural networks for software effort

estimation. Int. J. Comput. Sci. Issues, 7: 11-17.

[17] Johnson, P.M., 2005. A continuous, evidence-based approach to discovery and assessment of software engineering best practices. Technical Report CSDL-05-05, Department of Information and Computer Sciences, University of Hawaii, U.S.

[18] Jørgensen, M., P. Sørgaard and S. Grimstad, 2004. Project management of public software Projects: Avoiding effort overruns. Department of Informatics Faculty of Mathematics and Natural Sciences University of Oslo, Norway.

[19] Jørgensen, M. and S. Grimstad, 2009. The impact of irrelevant and misleading information on software development effort Estimates: A randomized controlled field experiment. IEEE Trans. Software Eng., 37: 695-707.

[20] Jørgensen, M. and G. Stein, 2012. Software development estimation Biases: The role of interdependence. IEEE Trans. Software Eng., 38: 677-693.

[21] Kaur, A. and G. Sunil, 2011. A framework for analyzing software quality using hierarchical clustering. Int. J. Comput. Sci. Eng., 3: 854-861.

[22] Kemerer, C.F. and C.P. Mark, 2009. The impact of design and code reviews on software QUALITY: An empirical study based on PSP data. IEEE Trans. Software Eng., 35: 534-550.

[23] Kim, D., W. Xinming, K. Sunghun, Z. Andreas, S.C. Cheung and P. Sooyong, 2011. Which crashes should i fix first?: Predicting top crashes at an early stage to prioritize debugging efforts. IEEE Trans. Software Eng., 37: 430-447.

[24] Kitchenham, B., O.P. Brereton, B. David, T. Mark, B. John and L. Stephen, 2009. Systematic literature reviews in software engineering - A systematic literature review. Inf. Software Technol., 51: 7-15.

[25] Kvale, S., 1996. Interviews: An introduction to qualitative research interviewing. Sage Publications, Thousand Oaks, California.

[26] Larson, S., 2012. Cognitive Bias in the Verification and Validation of Space Flight Systems. 3-10, March, 2012 IEEE, -

[27] Leung, H. and F. Zhang, 2012. Software Cost Estimation. Hong Kong: Department of Computing The Hong Kong Polytechnic University, Hong Kong.

[28] Malz, C., J. Nasser and G. Peter, 2012. Prioritization of Test Cases using Software Agents and Fuzzy Logic. 2012 IEEE Fifth International Conference on Software Testing, Verification and Validation, 17-21, April, 2012 IEEE, 483-486.

[29] Petersen, K., F. Robert, M. Shahid and M. Michael, 2008. Systematic Mapping Studies in Software Engineering. 12th International Conference on Evaluation and Assessment in Software Engineering, 26-27, June, 2008.

[30] Petersen, K. and W. Claes, 2009. Context in Industrial Software Engineering Research. 3rd International Symposium on Empirical Software Engineering and Measurement. 15-16, Oct., 2009, IEEE, -

[31] Santos, P.S.M.d. and H.T. Guilherme, 2013. Action research can swing the balance in experimental software engineering. Adv. Comput., 83: 205-276.

[32] Sun, B., S. Gang, P. Andy and R. Soumya 2012. CARIAL: Cost-Aware Software Reliability Improvement with Active Learning. IEEE Fifth International Conference on Software Testing, Verification and Validation, 17-21, April, 2012, IEEE, 360-369.

[33] Tamrakar, R. and J. Magne, 2012. Does the Use of Fibonacci Numbers in Planning Poker Affect Effort Estimates?. Oslo: Simula Research Laboratory, 14-15, May, 2012, IET, 228-232.

[34] Wohlin, C., 2013. An Evidence Profile for Software Engineering Research and Practice. In: Perspectives on the Future of Software Engineering - Essays in Honor of Dieter Rombach. Münch, J. and K. Schmid (Eds.). Springer Berlin Heidelberg, Germany, pp: 145-158.