Fallacies and biases when adding effort estimates

Magne Jørgensen
Simula Research Laboratory
Fornebu, Norway
magnej@simula.no

***Abstract*— Software professionals do not always clarify what they mean by their effort estimates. Knowing what is meant by an estimate is, however, essential when adding individual effort estimates from a work breakdown structure to find the estimated total effort. Adding the most likely instead of the mean effort of a set of cost elements may result in substantial underestimation of the total effort. In a survey of forty-four software companies we found only two companies that clarified the meaning of their estimates and had a proper method for adding these estimates. The other companies typically added single point estimates without clarifying what they added or with types of estimates likely to give too low estimates of the total effort. We examine the effect of improper addition of estimates and find, for the studied contexts, that summing the most likely effort estimates would lead to a substantial under-estimation of the most likely total effort. We also find that the use of the PERT-method, which provides a proper statistical basis for adding effort estimates and is used by many software companies, is likely to underestimate the effort in software development contexts. This is caused by, we argue and illustrate with empirical data, people's tendency towards providing too narrow minimum and maximum effort intervals. We outline a method that, we believe, better ensures that proper estimates of the total effort are produced.**

Keywords—Cost estimation; project management; adding estimates; work breakdown structures (WBS), uncertainty analysis

I. INTRODUCTION

A separation of the total software development work into cost element to produce a work breakdown structure (WBS) is a frequently used method for planning and estimation purposes in the software industry [1]. When a list of deliveries, use cases, activities, user stories or other types of cost elements has been produced, the effort needed for each element is estimated and summed to find the total work effort. Unfortunately, adding effort estimates of cost elements to find the total effort is *not* a trivial task in situations with a high degree of uncertainty in effort usage and a skewed outcome distribution, i.e., in situations typical for software development.

As an illustration, assume that your organization plans the completion of ten tasks of similar size and complexity, and that the effort you think a developer will use on each individual task is distributed as depicted in Fig. 1. The distribution in Fig. 1 reflects that you believe that a developer most likely will spend about 15 work-hours (the mode of the distribution), that it is hardly possible to spend less than 10 work-hours (the best case or minimum of the distribution), and that sometimes a developer may spend 40-50 work-hours (the worst case or maximum of the distribution). In about 50% of the cases you believe a developer will spend less than around 17 work-hours (this is the median or the p50-estimate of the distribution[1]). What is in this situation the estimated *total* effort needed for the ten tasks?

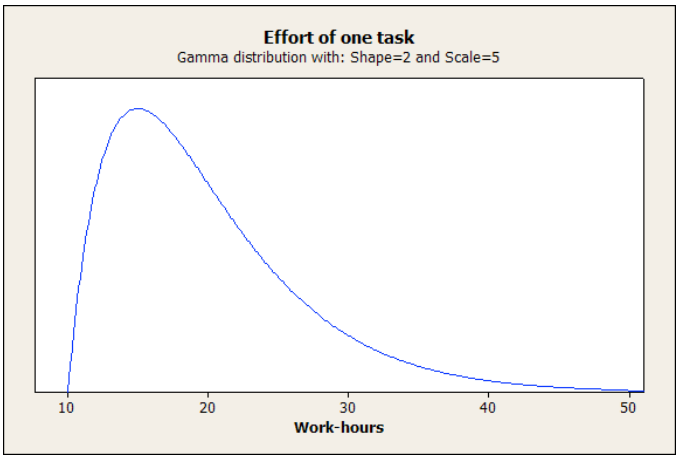

Fig. 1. The effort outcome probatility distribution of one task

An incorrect answer, reflecting some software companies' estimation practice (more on this in Section II), is that the ten tasks will require ten times the most likely effort, i.e., 10 x 15 = 150 work-hours. Another incorrect answer is that since it is 50% likely to use less than the median effort, it is 50% likely to use less than 10 x 17 = 170 work-hours on ten tasks.

A more accurate estimate of the most likely total effort is ten times the mean value, i.e., 10 x 20 = 200 work-hours. This reflects that we can only meaningfully add the individual mean effort values (and not the mode or the median values) of random values, and that the sum of the mean values of random variables approximates a normal distribution where the most likely, median and the mean values are similar (the central limit theorem). The meaningfulness of adding mean values is a consequence of the linearity of expected values of random variables. The normal distribution of the sum of the mean values is a consequence of the central limit theorem. Essential assumptions for the central limit theorem to apply fully in our estimation context include that the set of cost elements is large enough, that there is no strong dependencies between the cost elements and that there are no single or few elements that

[1] The median effort of the Gamma distribution in Fig 1 may be approximated by the formula $k\Theta(3k-0.8)/(3k+0.2)$ + a threshold value, where k (the shape) is 2, Θ (the scale) is 5 and the threshhold value is 10. The mean value of the Gamma distribution is $k\Theta$ + the threshhold value, i.e., 2x5 + 10 = 20 work-hours.

dominates with respect to size or uncertainty[2]. In our example, assuming that each cost element follows the Gamma distribution in Fig. 1, the distribution of the total effort of the ten tasks is as displayed in Fig. 2. In accordance with the central limit theorem, we see that the distribution is close to a normal distribution[3], which means that the most likely, the median and the mean effort are close to each other.

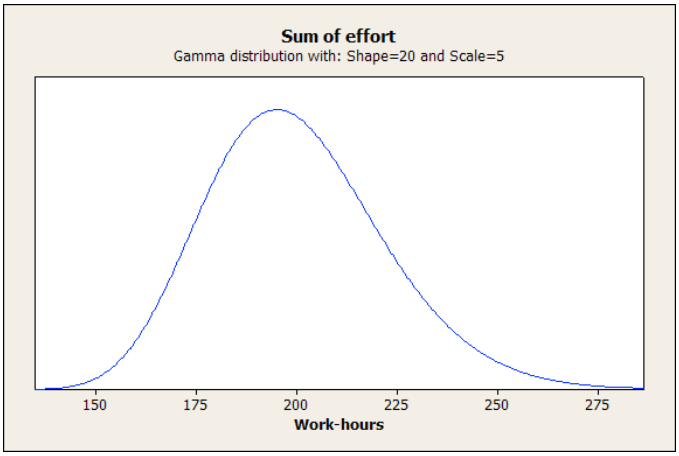

Fig. 2. The effort outcome distribution of the total effort of the ten tasks

Adding anything else than the mean effort values will give biased total estimates. It is, for example, 99% likely to exceed the sum of the most likely estimates, i.e., actual effort > 150 work-hours with 99% probability. Similarly, it is 80% likely to exceed the sum of the p50-estimates, i.e., actual effort > 180 work-hours with 80% probability.

To what extent do software companies fail to sum the mean effort estimates? Is this likely to be a serious mistake and a partial explanation of the tendency towards underestimating the effort of software projects? Does the use of the PERT (Program Evaluation and Review Technique) [3] or other three-point estimation methods that aim to provide methods for proper addition of adding estimates actually solve the problem?

In the remaining part of this paper we try to answer these questions and provide some recommendations on improved estimation practices addressing the identified problems.

There is not much previous work on challenges related to adding effort estimates in software engineering, but see for example [4-6] which addresses several of the elements discussed in this paper. We have failed to find any studies that mention this as a likely reason for cost overrun in software projects. The theory for the statistical discussions of the paper, e.g., related to properties of expected values of distributions and the central limit theorem, can be found in introductory textbooks on statistics [2].

[2] In practice, the central limit theorem often works amazingly well even when one or more of these assumptions are violated.

[3] The distribution of the sum of ten identical Gamma distribution has the same scale, but ten times higher shape (new shape = 2 x 10 = 20) and threshold (new threshhold = 10 x 10 = 100).

II. ESTIMATION SUMMATION PRACTICES

We received effort estimates together with estimation process description from forty-four small and medium large software companies located in Europe and Asia. The companies were identified through an internet search for offshoring software companies. The companies were paid for their estimation work. An examination of the estimates and estimation processes for the purpose of the analyses in this paper gave the following results:

- 73% (thirty-two) of the companies added single point estimates of individual cost elements. In all these cases, how to interpret the single point and the total effort estimates was not clarified, or the type of estimate, typically described as "most likely effort", made adding them statistically flawed in situations with skewed effort outcome distributions.
- 14% (six) of the companies used three-point estimates with varying names on the three estimation points, e.g., "minimum, average and maximum effort" or "best case, most likely case, worst case". One company estimated only the maximum and minimum effort and calculated the most likely effort as the average of these two values.
- Of the six companies using three-point estimates, only two calculated and added the mean effort values, i.e., only two of the companies demonstrated awareness of the essential difference between the mode and the mean effort when adding effort estimates. These two companies used the formula from PERT for finding the mean effort from the minimum, most likely and maximum effort (see Section IV for more about this). The other four added all the minimum effort values, all the most likely effort values, and all the maximum effort values to find the total minimum, the total most likely and the total maximum effort, i.e., they used a (statistically incorrect) strategy similar to that used for the single point estimates.
- 11% (five) of the companies applied a formal estimation model (three used Function Points, one used Use Case Points and one used COCOMO). These models are top-down (and has their own challenges [7]), which means that how to add the effort of individual cost elements is not a central issue.

One may argue that the challenge related to adding effort values to find the total effort is solved by adding a cost contingency, e.g., by adding 30% to the sum of the most likely effort values. While this may improve the situation in some contexts, we found little evidence that this would solve the problem among the surveyed companies. Only three of the companies examined added such contingencies. Even if they had added a cost contingency, it is in our opinion a poor practice to use cost contingency to repair a flawed addition of effort estimates. Not knowing the difference between the most likely and the mean effort means, for example, that you will not know how much to add as contingency.

Clearly, we cannot draw strong conclusions about the state-of-practice in the software industry based on our survey of a

few small and medium large software development companies in a few countries. The findings do nevertheless provide an indication on that there are challenges related to the interpretation and adding of effort estimates. The results are in accordance with those reported in [8].

III. How much will the total estimates be biased?

To illustrate how much adding the estimates of most likely or median effort may bias the estimated total effort we use the following measure of relative estimation error (REE):

(1) REE = Actual effort/Estimated most likely effort

Our illustration is based on the assumption that the distribution of the REE-values reflects the distribution of the effort outcome. If, for example, 25% of the cost elements have 50% or more effort overrun, i.e., have a REE-value higher than 1.5, this reflects that it is about 25% likely to use at least 50% more than the estimated most likely effort. We believe that this is a reasonable assumption, at least for a homogenous set of cost elements. It simply reflects that a meaningful estimate of the uncertainty of an estimate is the previous estimation uncertainty, as measured by the estimation error, of similar cost elements.

There are many different estimation contexts and some of them do not even show a tendency towards that using more effort than estimated is more likely than using less effort than estimated [9], i.e., the most likely and the median estimation overrun is about the same value. To reflect this diversity, at least to some extent, we analyze the following two real-world contexts:

- The context of forty-two small and medium sized software development projects, using the data collected in [10]. Project can be seen as large cost elements and our calculation reflects what will happen if a client (or software company) wants to estimate or budget the total cost of a *portfolio* of projects.
- The context of 440 smaller tasks (user stories) following an agile development process. This is a context with more feedback/learning and less estimation uncertainty compared to the previous context[4].

For each of these contexts we identified the mode values (the most likely REE) of the distributions by manually examining the empirical REE-distributions to identify the top point[5]. We use the distributions to analyze the difference between the mode and the mean and the median and the mean. The REE-distributions are displayed in Figs. 3 and 4.

Figs. 3 and 4 describe contexts where the most likely effort is the estimated effort, i.e., the mode of the REE-distribution is about 1.0 in both contexts. While the distribution in Fig. 3 is strongly right-skewed, the distribution in Fig. 4 is less so and has a larger proportion of observations to the left of the mode.

An analysis of the empirical data underlying Figs. 3 and 4 gives mean values of the REE-distributions of 1.42 and 1.16, respectively. This implies, by applying the central limit theorem and our assumptions of the relation between the REE and the effort outcome distribution, that a company using the sum of the most likely effort as their estimate of total effort should expect about 42% (the first context) and 16% (the second context) effort overrun. This level of effort overrun is only due to the incorrect addition of estimates and will be present even when the estimated most likely efforts of individual cost elements are accurate. In none of the contexts would the use of the median (p50) estimate remove the estimation addition problem. In the first context adding the median estimates would give 21% too low estimates of the total effort, while in the second this would not change anything at all, since the mode is the same as the median.

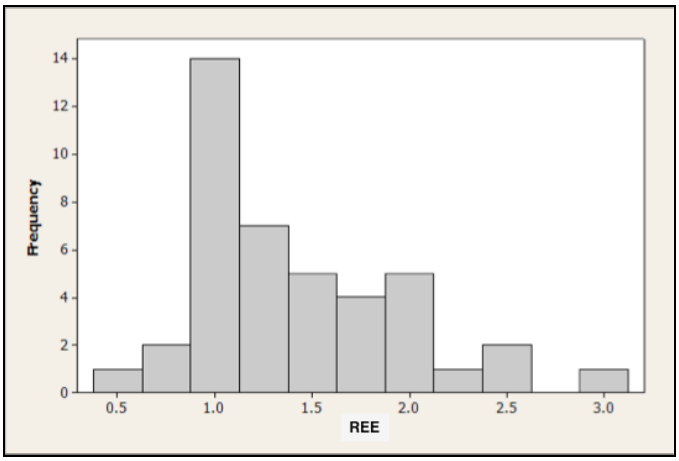

Fig. 3. The REE-distribution of the small and medium-large projects

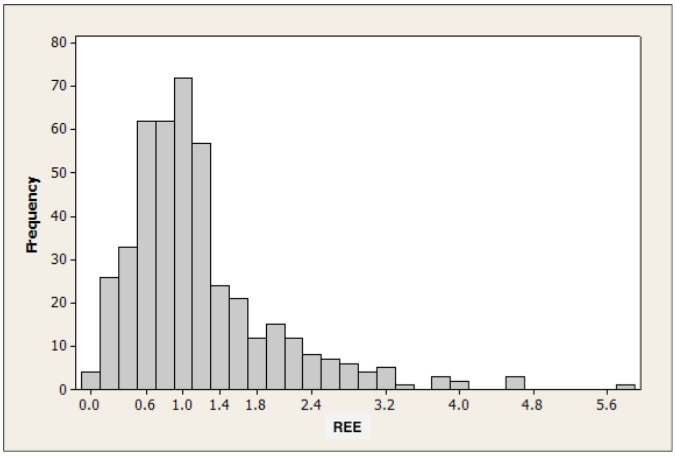

Fig. 4. The REE-distribution of agile development tasks

The above real-world scenarios have a few limitations:

[4] The data about 443 user stories were collected in a Norwegian software company with a team of seven developers completing tasks on different software systems. We removed three user stories where the effort was 2-5% of the estimated effort, suggesting that the actual effort was not correctly logged, resulting in 440 remaining user stories.

[5] The outcome of this manual process depends on the interval width of the histograms. To test the robustness of the manually found mode values, we examined the resulting mode values when fitting a Gamma distribution to the empirical data (using the statistical tool Minitab). This led to approximately the same mode values as when manually inspecting the distribution.

- We assume that the estimates leading to the REE-distribution reflect the estimated most likely effort. In the first context we explicitly asked for their estimates of most likely effort and in the second we have experienced that this is a common interpretation of estimates in agile estimation. In spite of this, we do not know how well the estimates actually reflect most likely effort and there may be a mixture of different interpretations behind the estimates and consequently the REE-values.
- The two contexts examined are selected by convenience, i.e., that we had the data available, not by any consideration about representativeness. The commonly observed tendency towards effort overruns in context similar to the first context and lack of such tendency in less competitive contexts [9], such as the second context, suggest that the contexts may be representative for a large set of contexts.

Notice that the non-symmetric distribution in Fig. 1 is consistent with, and could have been caused by, a lack of proper addition of estimates of cost elements within each project. If the projects had added the mean estimates of the cost elements of the projects, and these had been accurately estimated, the REE-distribution should have been symmetric. Lack of proper addition of estimates is, however, not the only possible explanation for the non-symmetric REE-distributions and it is highly speculative to use this as evidence for improper addition of estimates.

In spite of the above limitations, we believe that our results at least illustrates that lack of proper methods for adding estimates has the *potential* to explain parts of the effort overruns typically observed in software development contexts [11, 12] and that the awareness of how to properly add estimates is low.

IV. DO THE USE OF PERT AND SIMILAR METHODS SOLVE THE ADDITION PROBLEMS?

The PERT (Program Evaluation and Review Technique) was initially meant for estimating the duration, including the identification of the critical path, of a project [3, 13]. Many of the ideas, however, apply for effort estimation and are applied many software companies for that purpose, as well. The typical steps are as follows:

- Identify the minimum, the most likely (mode) and the maximum effort of each cost element. In practice, the interpretation of minimum and maximum is not based on the absolute minimum and maximum (which makes no meaning in software contexts), but for example a 90% minimum-maximum interval (90% likely to include the actual effort). It may also be requested on the format of p10-estimate (10% likely not to exceed) and p90-estimate (90% likely not to exceed), or more vague instructions such as: "Be almost sure (not) to include the actual effort in the minimum-maximum interval".
- Calculate the mean (expected) effort of each cost element: *(2) Mean effort = (Minimum effort + 4 x Most likely effort + Maximum effort)/6*
- Calculate the variance of each cost element: *(3) Variance of effort = (Maximum effort – minimum effort)^2/36*
- Calculate the estimated total effort (which will be close to the most likely and the median effort given that sufficiently large number of activities are included and other assumptions of the central limit theorem are met) as the sum of the mean effort.
- Calculate the total variance as the sum of the variance of each cost elements.
- Use the standard deviation (square of the total variance) and the assumption of normal distribution to find the pX-estimates, where X for example X is 50% and the p50-estimate is the number of work-hours 50% likely not to be exceeded).

The analytical calculation steps are sometimes replaced with so-called Monte Carlo simulation, which is more robust and flexible than the above process [14].

The PERT has frequently been criticized on basis on its statistical assumptions [13], but an even more serious problem is in our opinion related to the quality of the input. If PERT or the Monte Carlo simulation-based method do not receive realistic minimum and maximum effort values as input, they will, not surprisingly, provide biased estimates of the mean effort of the cost elements and consequently give a biased estimate of the total effort.

There are good reasons to believe that the input to PERT often is of low quality. It is, for example, well documented that people tend to provide too narrow minimum-maximum intervals [15, 16]. Our study in [17], for example, reports that the minimum-maximum intervals of cost elements in projects where the software professionals were instructed to be 90% sure (or almost sure) to include the actual effort in their minimum-maximum intervals, included the actual effort, on average, only about 30% of the time.

As a demonstration of how much poor input to PERT is likely to affect an estimate of the total effort, we examined the minimum-maximum effort values the six companies in Section II that provided three-points estimates. One of the six companies with three-point estimates did this on a format difficult to use for this analysis and is not included. For the purpose of our examination we use the following measures:

(4) Relative width of the interval = RWidth = (Maximum effort – Minimum effort)/Most likely effort

(5) Ratio of minimum to most likely effort = MinToML = Minimum effort/Most likely effort

(6) Ratio of maximum to most likely effort = MaxToML = Maximum effort/Most likely effort

Table 1 displays the median values of the above measures for each of the five companies' estimates of the same project. The median values are calculated as the median of the minimum estimates, the median of the most likely estimates and the median of the maximum effort estimates of all the

identified cost elements. The companies identified between 10 and 30 cost elements each for the project.

Table I displays that the width of the effort intervals vary much among the companies. The median RWidth is 0.4. An RWidth of 0.4 reflects, for example, that a software company thinks it will maximum use 20% more and minimum 20% less than the most likely effort. Table I shows that the minimum effort values are as far (in percentage) from the most likely effort as the maximum effort values are from the most likely effort for all companies. This suggests a belief in a *symmetric* effort outcome distribution. A symmetric effort outcome distribution would mean that it is just as likely to spend more as it would be to spend less than the estimated most likely effort. This is, we believe, a highly questionable assumption in software development contexts and, as argued earlier in this paper, not consistent with the results from empirical studies on this issue, e.g., on the tendency to use more effort than estimated.

TABLE I. EFFORT INTERVAL PROPERTIES (MEDIAN VALUES)

Company	Measures		
	RWidth	***MinToML***	***MaxToML***
A	0.4	0.8	1.2
B	1.0	0.5	1.5
C	0.4	0.8	1.2
D	1.5	0.5	2.0
E	0.2	0.9	1.1
Median	0.4	0.8	1.2

Table II includes two scenarios illustrating the similarity in estimation bias between incorrect adding of estimates (as discussed in Section III) and incorrect input of minimum-maximum to PERT and similar method. For our illustrative purpose we compare the following estimates:

- TRUE-EST: The mean of the true distribution of effort values of a cost element. The true distribution is here assumed to be the REE-distribution in Fig. 3 for Scenario 1 and the REE-distribution in Fig. 4 for Scenario 2. The TRUE-EST is set to 100% for both scenarios, i.e., the other estimates (see below) are calculated as proportion of the true mean estimate. If , for example, the ML-EST is 80%, this means that ML-EST is 80% of the TRUE-EST.
- ML-EST: The estimated most likely effort of a cost element. We use the same distributions as for TRUE-EST, i.e., those in Figs. 3 and 4. Since the mean values were 142 and 116% of the most likely effort for Scenarios 1 and 2, we have that ML-EST values are the inverse, i.e., 100/142 = 70% and 100/116 = 86% of TRUE-EST.
- PERT-EST: The PERT-estimated effort uses the median minimum and maximum effort values from Table I (the values in the last row). These values reflect the narrow and symmetric minimum and maximum values typically provided by the companies applying PERT in our survey (see Section II). PERT-EST is calculated as TRUE_MEAN-EST/(MinToML + 4 x ML + MaxToML), where the denominator of the ratio is the PERT-formula for the mean estimate.

TABLE II. USE OF ML OR PERT WITH BIASED INPUT

Scenario	Estimate in percentage of mean effort		
	TRUE_EST	***ML-EST***	***PERT_ EST***
1	100%	70%	72%
2	100%	86%	88%

As can be seen in Table II, the underestimation of the total effort when applying PERT with the narrow and symmetric minimum and maximum intervals typically applied by the companies, which results in estimates of 72% and 88% of the mean-based effort, will be substantial and similar to that observed when simply adding most likely value which will give 70% and 86% of the mean-based effort.

A further illustration of the effect of too symmetric and narrow distribution is given in Figure 5, where an almost symmetric, narrow effort Gamma distribution (solid line, Shape = 5, Scale = 5, Threshold = 80) is compared with a wider and right-skewed Gamma distribution (dotted line, Shape = 2, Scale = 20, Threshold = 80). Assume that the distributions represent two possible representation of the effort outcome of the same cost element. Both distributions have the same most likely use of effort (the same mode), which in both cases equals 100. The mean values, however, differ very much (105 vs. 120).

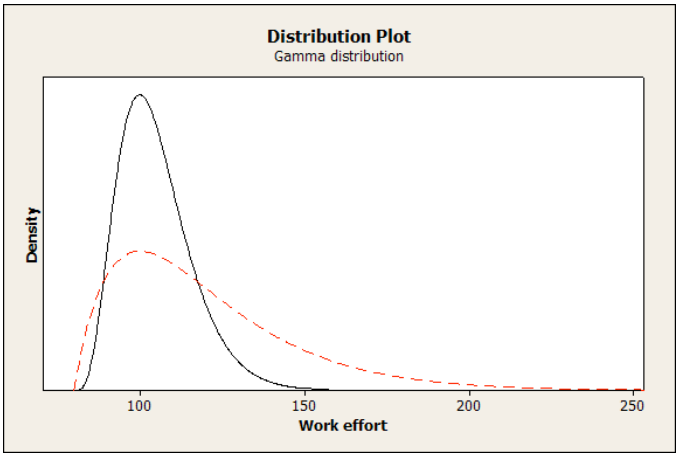

Fig. 5. Narrow, symmetric vs wide, asymmetric effort distribution for one cost element

Assume furthermore that the actual effort outcome probability distribution of the ten cost elements is the dotted line, while the provided estimates are based on the solid line. The difference between the estimated and the actual distribution of the total effort of the ten tasks will then be as displayed in Fig. 6.

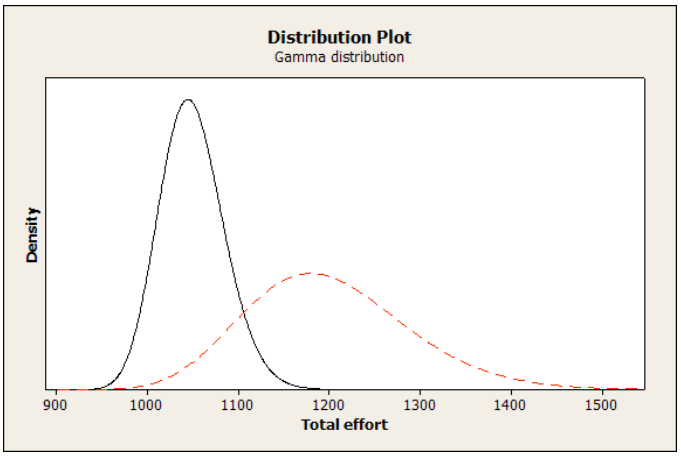

Fig. 6. Narrow, symmetric vs wide, asymmetric effort distribution for total effort of ten cost elements

When reading about PERT and similar methods in textbooks and frameworks we have failed to observe any instruction on how to provide the user with tools and techniques that ensure accurate input of minimum and maximum values. The textbooks and frameworks consequently seem to assume that people are good at knowing when they are, say 90% as compared to 60% or 95% confident in including the actual effort in their minimum-maximum interval. Our data, and many other papers, document that we are not and that we need support to find realistic minimum-maximum intervals. If we cannot manage to give PERT or similar methods proper minimum and maximum, they are not of much help and perhaps lead to as biased estimates as just adding the most likely effort values.

V. Outline of a Method for Adding Estimates

How can we improve the estimation process so that adding the effort of different cost elements does not lead to biased, in particular too low, effort estimates? Below we outline a method that addresses some of the challenges identified in the previous sections. The outlined method assumes that the known software development work has been broken down into proper cost elements and that a risk analysis has led to the identification of risk elements. It is, furthermore, based on dividing effort uncertainty into three types:

- Uncertainty connected with individual cost elements. This should be analyzed by assessing the uncertainty distributions of individual cost elements.
- Uncertainty related to general (at least two cost elements) factors (frequently termed risk factors). This should be analyzed by assessing the probability and consequences of each uncertainty factor.
- Uncertainty related to, at the time of the estimation, unknown risks, opportunities and activities.

The goal of the proposed method is not to produce a single point estimate, but rather the distribution of effort outcome values, where different types of estimates can be used for different purposes, e.g. a p50-estimate for planning and a p85-estimate for budgeting. Clearly, the method needs more work to be complete and empirical studies to examine whether it actually leads to improved estimates or not. The outline is just meant to be the start on this work and to indicate that some of the identified problems need new solutions. We are in the process of supporting the steps of the method with a simple (Monte Carlo simulation-based) tool.

Phases and steps of the method:

Phase I: Estimate the distribution of total effort of the cost elements

1. Estimate the ideal effort of each cost element. The ideal effort may be understood as the effort required given that there are no disturbances and no major problems, e.g., similar to the concept "ideal hours" it in agile contexts.
2. Estimate the most likely effort by contrasting it with the ideal effort. The full motivation and argumentation for starting with ideal effort and contrast it with most likely effort is given in [18], which documents that this improves the realism of software professionals' estimates. The study in [18] also documents that software developers typically provide the same estimates regardless of whether asked to provide "most likely" or "ideal" effort, which suggest that "most likely" effort is based on too idealistic assumptions. This method has, we believe, the additional benefit of improving the consistency of what is meant by the effort estimates, which is essential for proper addition of them.
3. Select a distribution (e.g., among a few premade distributions of varying width and skewness) for different cost elements. Let these distributions be based on historical data (or experienced people's memories) on the relative estimation accuracy for similar types of tasks. More on how to do this is described in [19]. There may be a few cases where one should deviate from the premade distributions, e.g., when one knows very much about the uncertainty and how it affects the effort usage. The use of premade, empirical data-based, distributions is motivated by the need to widen and right skew the intervals in many contexts. Just asking for minimum and maximum is not likely to lead to this. In addition, we believe that premade distribution has the potential of speeding up the estimation process.
4. Use the mean effort of the distributions when adding the effort of each cost element or add the distributions by using Monte Carlo simulation.

Phase II: Add the effort distribution related to known uncertainty

5. For each of the general uncertainty factors select a distribution (among premade distribution of different types) that reflects the extra (or reduced) effort connected with it. This may very much be the same process as for the cost elements, except that the most likely impact typically will be zero work-hours. Preferably, the distributions to select between should be based on empirical data, e.g., data about how frequently a type of risk occurs and its consequences. Notice that an uncertainty factor can lead to reduction of effort. This may in particular be the case

when there is flexibility in the delivered level of non-functional requirements. The outcome of this step should be a distribution with which the distribution from part I is multiplied through Monte Carlo simulation.

Phase III: Add the effort related to unknown uncertainty

6. Add a probability distribution for unexpected activities and events. This should, as with the other elements, be based on historical data about the effort spent on non-planned activities and events. The outcome of this step is a distribution with which the effort distribution from part II is multiplied through Monte Carlo simulation.

Monte Carlo simulation may for software professionals at first sight sound complex. It is not and we have experienced that software professionals quickly understand and feel comfortable with this type of simulations. In fact, it may be easier to explain and understand Monte Carlo simulation than the weighting in, for example, PERT.

A few of the elements in the proposed method, in particular the selection of pre-made, empirically-based uncertainty distributions, have not been empirically tested, yet. A method combining several of the other elements has, however, been evaluated with good results in [20].

VI. SUMMARY

In this paper we argue that adding effort of cost elements requires a precise understanding of what an estimate is and that adding estimates to find an unbiased estimate of the total effort is not a trivial task in many software development contexts. If we add most likely or median estimates we may frequently get as result a much too low estimate of the total effort, even when the individual estimates are accurate when interpreted as the most likely use of effort.

The real-world relevance of this problem is documented by surveying a set of companies' effort estimation processes. The survey indicates a poor practice based on lack of preciseness in use of estimates and questionable practices for adding estimates. We found that only a few companies that apply three-point estimates (minimum-most likely-maximum effort) and use a proper statistical formula to calculate the mean effort and the estimated total effort. While PERT and similar methods are statistically sound, the usefulness of the methods seems to be hampered by the input of too narrow and symmetric minimum-maximum intervals. Our illustrative analysis suggests that this may lead to just as large bias towards too low estimates as when just adding the most likely effort values.

We outline a probabilistic and uncertainty analysis based method for adding effort that addresses the above problems. We plan to further elaborate on and empirically evaluate this method.

VII. REFERENCES

[1] K. Moløkken and M. Jørgensen, "A review of software surveys on software effort estimation," in *International Symposium on Empirical Software Engineering*, Rome, Italy, 2003, pp. 223-230.

[2] T. H. W. Wonnacott and R. J. Wonnacott, *Introductory Statistics*: John Wiley & Sons, 1990.

[3] J. J. Moder, C. R. Phillips, and E. W. Davis, *Project management with cpm, pert and precedence diagramming*: Van Nostradn Reinhold, 1983.

[4] M. Klaes, A. Trendowicz, A. Wickenkamp, J. Münch, N. Kickuchi, and Y. Ishigai, "The Use of Simulation Techniques for Hybrid Software Cost Estimation and Risk Analysis," *Advances in computers,* vol. 74, pp. 115-174, 2008.

[5] L. C. Briand, K. El Emam, and F. Bomarius, "COBRA: A hybrid method for software cost estimation, benchmarking, and risk assessment," in *International Conference on Software Engineering*, Kyoto, Japan, 1998, pp. 390-399.

[6] M. Klaes, A. Trendowicz, and H. Nakao, "Handling Estimation Uncertainty with Bootstrapping: Empirical Evaluation in the Context of Hybrid Prediction Methods," in *ESEM*, Banff, Canada, 2011, pp. 22-23.

[7] M. Jorgensen and B. Boehm, "viewpoints Software Development Effort Estimation: Formal Models or Expert Judgment?," *IEEE Software,* vol. 26, pp. 14-19, Mar-Apr 2009.

[8] S. Grimstad, M. Jorgensen, and K. Molokken-Ostvold, "Software effort estimation terminology: The tower of Babel," *Information and Software Technology,* vol. 48, pp. 302-310, 2006.

[9] T. Halkjelsvik and M. Jørgensen, "From origami to software development: A review of studies on judgment-based predictions of performance time," *Psychological Bulletin,* vol. 138, pp. 238-271, 2012.

[10] K. Moløkken-Østvold, M. Jørgensen, S. S. Tanilkan, H. Gallis, A. C. Lien, and S. E. Hove, "A survey on software estimation in the Norwegian industry," in *10th International Symposium on Software Metrics*, Chicago, IL, 2004, pp. 208-219.

[11] D. Yang, Q. Wang, M. S. Li, Y. Yang, K. Ye, J. Du, and Acm, "A Survey on Software Cost Estimation in the Chinese Software Industry," in *ACM/IEEE International Sympsoium on Empirical Software Engineering and Measurement*, Kaiserslautern, GERMANY, 2008, pp. 253-262.

[12] K. Moløkken, M. Jørgensen, and I. C. S. Ieee Computer Society, "A review of surveys on software effort estimation," in *International Symposium on Empirical Software Engineering*, Rome, Italy, 2003, pp. 223-230.

[13] J. E. Murray, "Consideration of PERT assumptions," *IEEE Transactions on Engineering Management,* vol. 3, pp. 94-99, 1963.

[14] Z. Laslo and G. Gregory, "PERT-type projects: time–cost tradeoffs under uncertainty," *Simulation,* vol. 89, pp. 278-293, 2013.

[15] C. R. M. McKenzie, M. Liersch, and I. Yaniv, "Overconfidence in interval estimates: What does expertise buy you?," *Organizational Behavior and Human Decision Processes,* vol. 107, pp. 179-191, 2008.

[16] T. Connolly and D. Dean, "Decomposed versus holistic estimates of effort required for software writing tasks," *Management Science,* vol. 43, pp. 1029-1045, July 1997 1997.

[17] M. Jørgensen, K. H. Teigen, and K. Moløkken, "Better sure than safe? Over-confidence in judgement based software development effort prediction intervals," *Journal of Systems and Software,* vol. 70, pp. 79-93, feb 2004 2004.

[18] M. Jørgensen, "Contrasting Ideal and Realistic Conditions as a Means to Improve Judgment-based Software Development Effort Estimation," *Information and Software Technology,* vol. 53, pp. 1382-1390, 2011.

[19] M. Jørgensen and D. I. K. Sjøberg, "An effort prediction interval approach based on the empirical distribution of previous estimation accuracy," *Information and Software Technology,* vol. 45, pp. 123-136, mar 2003 2003.

[20] M. Jørgensen and K. Moløkken-Østvold, "Eliminating over-confidence in software development effort estimates," in *PROFES*, Japan, 2004, pp. 174-184.