Can We Provide Better Protection against Budget Overruns of Software Projects?

Rainer Gerlich

BSSE System and Software Engineering

Auf dem Ruhbuehl 181
D-88090 Immenstaad, Germany

Phone +49/7545/91.12.58
Mobile: +49/171/80.20.659
Fax +49/7545/91.12.40
e-mail: Rainer.Gerlich@bsse.biz
URL: http://www.bsse.biz

Abstract: The answer to this hypothetic question is "yes", of course. The paper will approach the problem in two steps: firstly, we will discuss if and which measures exist to identify an overrun early enough, secondly, we will analyse the sources of overruns and which means may be applied not to exceed the planned budget.

1. INTRODUCTION

In the "good old days" of space engineering a reliable figure for cost estimation was the mass of the spacecraft [1]. Analyses yielded that the costs per mass unit was characteristic for a certain type of spacecraft. This shows that a rather simple parameter can well be applied to estimate costs. Also, in case of pure paperwork it was or it is usual to calculate the average costs of a page. This roughly allowed a customer to compare the efficiency of a project's output, at least for similar types of projects (and relying that the contractor did his best regarding the contents of a page).

Units like "LOC" (Lines Of Code) [2] or "FP" (Function Points) [3] are usually used by software engineers to estimate the required costs. However, the amount of LOCs and FPs is not known when the price shall be fixed. This is similar to the units "mass" and "page" which are also known only at the end (for mass this is not completely true, because an upper limit exists right from the beginning in case of a spacecraft). According to Sneed [4] the function-point method was intended to be applied after the design phase in order to estimate the costs of implementation. Such figures are usually available rather accurately at this phase of a software project . A discussion on metrics based on LOC and FP can be found in [16]. However, it also happens that cost estimations derived from the design may also be insufficient.

The principal problem of cost prediction is that knowledge exists about dependencies on cost parameters like LOC, but the information on the parameters' values is missing or not precisely known when the costs of a project need to be estimated. Therefore the early need of cost estimation in case of a firm-fixed price implies high risks.

There are two issues regarding risk reduction: (1) to identify the most suitable parameter set and parameter values as accurate as possible and, and (2) to identify uncertainties of the parameters and their impact on the predicted costs. Regarding (1) we will look for parameters better suited than LOC, regarding (2) we will look for getting (early) indicators flagging that an estimation is wrong.

2. COST DRIVING PARAMETERS

The most famous cost estimation models either take LOCs (Lines of Code) or FPs (Function Points) [see e.g. 5,6,7,8] as major parameters while only requirements (user requirments or software requirements) are available at an early project phase. Consequently, an engineer has to derive the actual values in terms of LOC from the requirements. The required transformation is - according to our knowledge - rather informal, and may lead to wrong results. Therefore we suggest to take the requirements as data on which to base cost prediction.

The classification of requirements as introduced by ESA PSS-05 [9] gives a good set of parameters: functional, interface operational, resource, reliability requirements etc. Each such category forms an own cost modelling class and the actual value of the parameter could be derived from the amount of requirements of a such a category. Unfortunately, such strict categorisation of requrements is not applied in general, and even for ESA projects it might disappear because the new ECSS E-40 [10] standards do not explicitly require a standard classification of requirements. Therefore, project managment needs to care about a standardised organisation of the specification to ensure that costs can directly be derived from requirements. We believe that a figure derived from requirements is as reliable as the mass was in the past, once a database is available for the different types of requirements. However, as this suggestion is new we cannot provide practical figures which prove the validity of our approach. This would require evaluation of the rquirements of finished projects vs. costs at completion.

The reason why we consider such an approach based on requirements as more reliable as the one based on LOC is that it makes the mapping of requirements on costs more formal. Each requirement contributes to the final costs, and the transformation constants and the dependency on the parameter set can be derived similarly as for LOCs or FPs. Then facts, i.e. the existing requirements, instead of assumptions, i.e. derived LOCs, will form the base of estimation. An engineer will do the transformation according to some "rule of thumb", only, and he may be biased because managers requiring an excellent technical product at minimum costs, making pressure on him to minimise the results of cost estimation (see also [11] and the discussion below).

2.1 What does a requirement cost?

For the successful completion of a project it is essential to know the costs caused by a requirement. The customer needs to know what he has to pay for, and the contractor should now which budget is required. It does not help if the customer makes pressure on the contractor to redurce the price. When the contractor goes bankrupt, the development cannot be completed or it cannot be maintained. And the contractor should not reduce the price below the real costs just to get the contract, being later faced with bankruptcy. There is an exceptional and legal, but from a principal point of view immoral situation to do so: the contractor makes a bid at a reduced price and expects that the customer has a strong interest to continue in case of a budget overrun and can be convincved to pay more.

It is common practice, that technical engineers define the requirments, while the budget is defined by managers. There is a high risk to exceed the budget, because the engineers see the technical challenge, but not the costs behind, and they may be biased regarding identification of real costs. We will discuss this sitatuation later.

Of course, it is impossible to give an exact figure on "LOC per requirement", even not for certain types of requirements, but we rely on the expressiveness or representativity of "average" figures. Probably some parts are underestimated (less effort than required), and other parts are overestimated (more effort than required). Our experience shows that the sum of all parts yields a representative figure. In one case the incomplete knowledge on a problem leads to an underestimation, in another case to an overestimation, and in total the sum of all erroneous estimations is small compared to the overall effort.

We cannot prove this hypothesis currently by real figures, but we learned by our projects that some problems will disappear, because the solution is much simpler than expected (overestimation of the problem), while other problems will come up (underestimation). We even recognised that technical solutions planned at an early stage may be replaced by better and more efficient solutions (from technical and cost perspective), when the problem has to be tackled during implementation. Therefore it is wasting of time to spend much effort in accurate cost estimation, looking on technical details, which could disappear later on. This is true at least for challenging projects which go beyond the currently applied techniques, as it happens for most of the space projects, and especially for space software.

Just recently - after the conference - we heard about a more general use of the FP-method which complies in principal with our suggestion. In case of the classical FP-approach, inputs, outputs and data are counted, and weights are assigned: 3 .. 6 to inputs, 4 .. 7 to outputs, 7 .. 15 to data, this is the IFPUG 4 standard. According to Sneed [4] one man-month is equivalent to about 20 FPs.

Such weighting can be applied to other items, too. We recognised a discussion on this subject in the IFPUG community [17] and further on-going activities on this subject [18, 19]. As discussed there, (low-level) requirements are subject of the FP formalism, i.e. weights are assigned to requirements for derivation of the required effort. Also, the estimation of test effort from requirements is under discussion.

2.2 Visible and non-visible effort

The functionality of a system is expressed by a number of requirements, the functional requirements. They can be counted and converted into an average effort. This effort is directly related to the functional requirements, and "visible" as the requirements are. However, non-functional requirements exist, like for interfaces, testing, verification, validation, reliability, safety and documentation, which apply to each functional requirement. A non-functional requirement causes additional effort. Therefore such non-functional requirements could be considered by an additional overhead to the contribution of a functional requirement. For a rough estimation a constant percentage should be sufficient. The contribution from the different categories of requirements may be weighted to achieve a higher accuracy.

There is another risk regarding effort estimation: resource constraints which may significantly increase the estimated effort. When memory or CPU budgets are exceeded, but the hardware cannot be upgraded, the effort needed to master such a problem may be a multiple of the estimated effort. If more effort is required, more man-power resources are needed. When the personnel has to work over time, additional overheads have to be paid. In consequence, costs will rise suddenly, more staff is needed and the schedule has to be extended.

It is important to consider non-visible effort such as caused by non-functional requirements. Cost risks caused by technical constraints cannot considered this way, because they are hidden. The only way to master them is to identify they early enough, and to find means to solve such problems, or to avoid the risks. This will be discussed in chapter 4.

2.3 Normalisation of estimation parameters

When deriving costs from requirements the costs per requirement need to be calibrated. Figures may be derived from completed projects and applied to predict the costs of future projects. However, we also need to care about the level of detail of requirements.

The costs may continuously be tracked from the beginning to the end. At the beginning only high level requirments may exist. Later, refined requirements will be established, so the total number of requirements will increase steadily. When the same weight is assigned to a requirement at each stage of refinment, the costs will grow as the number of requirements will. However, the costs at each stage should converge to the final and real costs.

Consequently, requirements need to be weighted according to the level of refinement. This could happen e.g. for the requirements available at end of Phase B or C (in ESA terminology), or at milestones during these phases.

2.3 The risk of overall underestimation

In our opinion, there is a psychological reason behind the underestimation of costs. We see two principal cases:

- requirements driven by technical issues, and
- requirements driven by political issues.

2.3.1 Requirements driven by technical issues

We presume that each engineer intends to do the job as best as possible. Consequently, the engineers at the customer's site will give as many requirements as necessary to specify the product, and to be sure to get the right product from the contractor based on the specification. Similarly, the contractor appreciates detailed requirements assuming that by more precise definitions the amount of work can be kept small, i.e. the customer does not have a chance to request more due to unclear requirements.

But the amount of work caused by such requirements may not be reflected correctly because both want to keep the price as low as possible.The customer wants to pay as little as possible, the contractor may be interested in a low price, too, because he might not get the contract if the price is too high.

We frequently observed in practice that for above reasons the management applies pressure on the technical staff to reduce the costs. The result is that the areas where too high costs have been assumed are identified and discussed until engineers are convinced that costs can be reduced. This process stops when the costs reach the envisaged envelope.

However, what is wrong with this approach is that everybody concentrates on the potential overestimations (in the sense that more costs are estimated than required). But the areas which were underestimated (less costs than required) are not identified at all, because management is not interested to increase costs. And they want the commitment from the technical staff to the lower costs, assuming that such guys always put more functionality in than required.

We have made the experience that having a first guess on the costs without looking on details, just by comparing previous with intended work, already gives a good cost figure. Of course, some assumptions may turn out as wrong later on, but on the average the first rough figure is not too bad. This means that an enginner may estimate more costs in some cases, and less costs in other cases, as already discussed above. But such wrong estimations - in both directions - lead to a good result in total, as we already know from statistical measurements e.g. in physics.

When the overestimations are removed, but not the underestimations, the result is no longer representative, it may fit the cost envelope of the management, but it is wrong. This is - in our opinion - the reason why projects usually approach the higher cost figure at the end again, which was previously derived but was lowered due to discussions on cost reduction. Unfortunately, we do not have figures which we can present, but this is what we observed in practice.

Therefore discussions on the validity of cost estimations need to consider both, over- and underestimations, otherwise a budget overrun will occur, for sure.

2.3.2 Requirements driven by political issues

In this case either the customer, the contractor or both are interested in unclear requirements, hoping that

- in case of the customer, more can be requested than paid,
- in case of the contractor, less can be delivered than paid, or a higher budget can be requested for additional requirements at a later phase of a project.

This approach bears a high conflict potential, because the requirements can be interpreted in different manner. In this case costs cannot really be derived from requirements, because they do not represent the real costs. The price has been agreed and both parties will try to optimise their position.

In such a case derivation of costs from requirements will also fail, because the amount of requirements will be reduced, and, in consequence, the costs, too. What could help - if there is an interest to identify such a problem - is to have an indicator telling both that requirements are poorly defined.

3. CROSS-CHECKING PREDICTIONS

The validity of an estimation can hardly be assessed by the originator. He always will claim, that the estimation is correct. In the technical area it is common practice to compare several (at least two) independent instances of the same item ("voting"). Differences indicate that at least one instance may be erroneous. However, the crucial point is the assumption on independence.

If two engineers make an estimation, do they apply different experience or not, do they really independently derive a result. The challenge is here: where is the common root of all conclusions? If they all relate to the same source - and this may be not visible - such a comparison may be not valid, too.

A better approach - in our opion - is, to look for inconsistencies in the assumptions the engineers or an engineer made. We learned that a conclusion is only valid, when it is reproducable under different conditions. If conclusions differ, this is an indication, that something is wrong. Usually, such conclusions are not identical, but they depend on each other and the related information needs to be transformed to become comparable. In fact, the different representations of equivalent information increases the probability that different conclusions will be drawn for the same conditions, if an engineer intends to drive conclusions into a certain direction, e.g. in order to meet some expectations.

In a recent paper on effort estimation uncertainty, Jorgensen [11] stated that "it matters how you ask". He pointed out that a question like "You don't believe it possible that the project will spend more than maximum 1700 work-hours, do you?" may induce an "anchoring" effect and increase the social pressure towards overoptimism. Instead, he suggests to ask "how likely is it that the project will require more than 1700 work hours?". He also refers to a paper about "Nonconscious Priming and Conformity to Social Pressure" [12]:

> When asking different questions, or requesting different inputs which rely on the same source of information like requirements, different results will be obtained, if the engineer is not really sure, or did derive the results under pressure.

Therefore identification of inconsistencies will help to protect against budget overruns. Again, there is obviously a psychological reason behind such inconsistencies. Strategies are already applied in other areas like opinion polling to identify such inconsistencies, to assess on the validity of the contents of a questionnaire and to remove or to ignore inconsistent information.

Usually, there are additional questions introduced into a questionnaire which ask for the same subject but by different phrases. This way a cross check is possible on the answers. In case of estimation of software costs such checks are also possible.

An engineer usually needs to give estimations on the technical and the cost budget. Both budgets are related to each other. In fact, this dependency is applied by the cost estimation models. When we take LOCs, we are talking about the size of the software. Hence, we are talking about the sizing budget. Now, based on figures derived from previous projects, we can compare the costs associated with the sizing budget to the result a cost estimation exercise.

In practice, the results of both estimation processes will be derived independently, more or less. At least different algorithms or a different point of view will be applied. Furthermore, an engineer will presumably try to keep the costs low, while taking more care on the sizing budget, which therefore tentatively will be higher than expected. Consequently, cross-checking of both figures will identify inconsistencies regarding cost estimation.

At the end of 1980s we identified a resource problem for a space project at the beginning of phase C/D, and initiated hardware re-design resulting in an increase of memory and CPU-power by morre than 200%. In parallel, we re-estimated the costs based on new work package descriptions, and came to the conclusion that costs would be higher by about a factor of 3, which was nearly compliant with the increase of memory and associated size of software. The project implemented the hardware re-design, but did ignore the prediction on increased costs. At the end, the timing and sizing budgets turned out as true - no problem was encountered for resources during in-orbit operations, but the new envelop of resources was really fully needed, - and the cost prediction was confirmed, too.

Another experience proving the usefullness of such cross-checks - as seen from now - was also made at end of 1980s during the phase A of a larger project. For reduction of technical risks the sizing budget was assumed to be twice of the actually estimated budget at end of phase B, while PSS-05 states that uncertainty of costs at end of Phase A should only amount to about 30%. Obviously, this was a significant contradiction indicating the high risk expected by the engineers.

By the last example one can also explain the independency of the estimations of the technical and cost budget. As hardware re-design during Phase C/D and easy increase of memory like for PCs is impossible in space area, the engineers wanted to reduce the high technical risks by taking a sufficient margin for the hardware resources. They did not recognise that this also implied doubling the costs. Nevertheless, from a global point of view the approach was not too bad, because a margin of 100% for memory doubles the costs only, if really needed, while costs would have exploded if the size of memory would have not been increased, but the need would have been arised. In fact, the big margin reduced significantly the financial risk. However, the risk in terms of cost overruns (potential doubling of costs) was not recognised at that time.

4. MEANS TO REDUCE THE BUDGET RISK

Two major sources of risks regarding cost estimation exist:

- to understimate the effort requested by requirements
- to miss technical constraints or overrun of technical budgets due to high technical challenges.

The main problem related to cost estimation is that the functionality and properties as requested by requirements are complex and not visible. Therefore an engineer may easily forget to consider a cost driver. A possibility to convert the requirements into the final system would help to get the right understanding on needed effort. Similarly, the test effort can be captured. But this requires effort for a representative implementation, and is not helpful at all.

However, in order to avoid this "hen-egg" problem, the optimum way would be to directly convert the requirements (expressed by "executable specifications") into the final product. This would reduce costs and risks. Such issues are currently addressed in the course of the ESTEC project on "Automatic Code Generation" [13] and by the methodology "Automated Software Production" (ASaP) [14].

Such automated approaches will also help to master risks related to missed technical constraints. They provide - if properly etablished - an emergency exit and allow to rapidly change a technical concept in a safe manner.

In case of the "Material Science Laboratory" (MSL) [15] a problem came up when the on-board database was brought into operation. The task of the MSL database is to manage data acquisition, calibration, limit monitoring, processing and telemetry handling for about 600 data items on an SPLC (ERC32) at 14 MHz. This caused immediately a CPU-overload, which however was expected, and therefore provisions had been foreseen to master such a situation.

The problem was caused by the given ordering of the data regarding calibration and post-processing. As the database was automatically generated, only the inputs to the generator were changed, and the completely restructured database software was available after about one day without any need for re-testing, then meeting the performance constraints of the CPU.

5. CONCLUSIONS

We have outlined why the current parameters LOCs or FPs currently used for cost prediction are not the best ones at an early stage of a project - in our opinion. Instead, we propose to take the requirements as parameters which will allow a formal derivation of costs. Also, we have pointed out that inconsistencies between the technical and the cost budgets can identify an invalid estimation.

Finally, we have shown how risks can be reduced and the impact by technical constraints and challenges can be mastered by a concept allowing to react on unpredictable events while keeping the impact on the cost budget neglegible.

REFERENCES

[1] internal, confidential communication, end of 1980's

[2] B.W.Boehm: Software Engineering Economics, Prentice Hall, 1981 (COCOMO-Model)

[3] A.J.Albrecht: Measuring Applications Development Productivity, Proceedings of IBM Applic. Dev. Joint SHARE/GUIDE Symposium, Monterey CA, 1979, pp.83-92

[4] H.Sneed: Function-Points aus historischer Perspektive, SEM, Heft 58, Februar 2002

[5] A survey on cost estimation measures can be found at http://irb.cs.tu-berlin.de/~zuse/metrics/History_05.html

[6] K.Maxwell, T.Eisele: ESA/INSEAD Data Analysis Report No. 5, Productivity of Core Database, May 1997

[7] K.Maxwell: ESA/INSEAD Data Analysis Report No. 6, General Effort Estimation Models based on Significant Productivity Figures, May 1997

[8] K.Maxwell, L.van Wassenhove, S.Dutta: Benchmarking: The Data Contribution Dilemma, INSEAD,,April 1997

[9] ESA PSS-05, ESA Software Engineering Standards

[10] ECSS European Cooperation on Space Standards, E-40,

[11] M.Jorgensen: Realism in Assessment of Effort Estimation Uncertainty: It Matters How You Ask, IEEE Transactions on Software Engineering, Vol. 30, No. 4, pp. 209-217, April 2004

[12] N.Epley, T.Gilovich: Just Going Along: Nonconscious Priming and onformity to Social Pressure, J. Experimental Social Psychology, Vol. 35, pp. 578-589, 1999

[13] ACG - Automatic Code Generation, ESTEC contract. no. , 2004, Noordwijk, The Netherlands

[14] ASaP - Automated Software Production, Dr. Rainer Gerlich BSSE System and Software Engineering, Auf dem Ruhbuehl 181, 88090 Immenstaad, germany, http://www.bsse.biz

[15] Eurospace Symposium DASIA2000 "Data Systems in Aerospace", May 22-26, 2000, Montreal, Canada
M.Birk, U.Brammer, K.Lattner, M.Ziegler, R.Gerlich:
Software Development for the Material Science Laboratory on ISS by Automated Generation of Real-time Software from Datasheet-based Inputs

[16] Software Sizing Measure,
http://www.testablerequirements.com/testablerequirements/soft_size_meas.htm Mosaic Inc.

[17] IFPUG News Group, Function Points vs. Testable Requirements,
http://www.ifpug.dom/discus/messages/1780/5234.html, dated Nov./Dec. 2003

[18] Testable Requirements as a Sizing Measure,
http://www.testablerequirements.com/testablerequirements/tr_as_size_meas.htm, Mosaic Inc.

[19] A New Paradigm From Function Points or Lines of Code For Sizing Software Systems,
http://www.testablerequirements.com/testablerequirements/new_paradigm_size_soft .htm, Mosaic Inc.