The global and independent source of data and analysis for the IT industry

Case Study

Software Cost Estimation – Redeveloping a large application

Published by the International Software Benchmarking Standards Group

09-2019

Executive Summary

A trading company would like to replace a large legacy application with an application that offers the same functionality, but in a modern technology.

Senior management understands that an accurate estimate is the basis for a successful project and wishes to carry out an independent cost estimate by a specialized third party.

In this case study, the steps that were carried out are listed and the results are given.

Table of Contents

The case

Background

A trading company in the Netherlands has a large legacy system, 20 years old and built using PowerBuilder technology. This is a RAD programming language, developed by PowerSoft, a company that was acquired by Sybase in 1995. It's an event-driven language and meant to quickly develop new applications with a database.

Now 20 years after the Go Live, the company experiences difficulties in finding people to maintain the application. The size has increased to over 2.000.000 Lines of code and it is a mission-critical system for the logistics of the organization.

One by one the employees that maintain the system retire from working and the number of FTE maintaining the application becomes critically low. In addition, there are few resources available in the market, as PowerBuilder is not seen as an interesting programming language by the younger generation and very few developers know it. IT recruitment agencies have few resources and the few available are hired out at an incredibly high hourly rate.

The Plan

The company will hire a system developer to produce the application, but after implementation, they need to have the right people in place to maintain the system.

For the company it's very important to understand when the new application will go live and the investment required for this. There are no certified software cost estimators in the company. They decide to hire a consultancy firm specializing in Software Cost Estimation, Benchmarking and Performance Measurement to carry out the independent cost estimate for rebuilding the system.

For this estimate the ISBSG repository 'Development & Enhancements (2019)' [1.] is used.

The estimate consists of several steps:

1. Determine the size of the requirements that need to be developed
2. Understand the estimation parameters and constraints
3. Select the right dataset as a reference
4. Analyze the data in the dataset
5. Create the landing zone (Low, likely, high) regarding the cost
6. Create the landing zone (Low, likely, high) regarding the schedule

These steps are further explained in this document.

Determine the size

Choosing a Sizing Method

To perform a professional independent cost estimate, it's important first to size the requirements that need to be delivered.

To use the size in LOC (lines of code) is a possibility, but there is no internationally accepted standard for LOC counting. Usually LOC is not a best practice for estimation purposes, although some organizations that consistently count the LOC in the same way can use data based on those LOC counts successfully.

For Software Cost Estimation, it's an industry best practice to use one of the 5 ISO standards for functional size measurement. Using one of these standards enables the possibility to use historical data of completed projects for purposes like software cost estimation.

The main advantage of functional size methods is the fact that they size the functionality that is requested by the user, regardless of the physical method of implementation and regardless of the development method.

These ISO/IEC standards are:

- IFPUG [2.]
- Nesma [3.]
- COSMIC [4.]
- FiSMA [5.]
- MK II [6.]

Measuring Size

The consultancy firm, in our case study, first measured the functional size using the Nesma High-level method. This is one of the above ISO/IEC standard methods for functional sizing which produces similar results to the IFPUG method, also one of the ISO standards. The advantage of this method is the fact that there is a lot of data available for projects measured in one of these methods.

The Nesma High-Level method has the advantage it can be used relatively quickly, while being almost as accurate as a detailed Nesma or IFPUG analysis.

In a study of van Heeringen, Prins and van Gorp [7.], the Nesma high-level method turned out to have an accuracy of 97.5% while 67% of the effort was necessary compared to the detailed analysis.

The certified Nesma CFPA analysts measured the functional size using various artifacts: the functional documentation which was not really maintained already for a number of years, the user documentation which was quite up-to-date, as the system is of critical importance for the company, and in some cases the physical screens and prints when the exact functionality could not be derived from the documentation.

The functional size measurement resulted in an application size of 8,500 Function Points.

Understand the estimation parameters and constraints

In the meantime, the estimation parameters and constraints needed to be defined.

The company wishes to use a modern programming language, to ensure they can use the most modern frameworks and have no problem hiring the necessary maintenance team with the knowledge of this technology.

The lead architect of the company chooses to use Microsoft .Net technology to develop the application. The architect also checks if code reuse is possible, however, the code cannot be reused. Although not maintained for a number of years, the functional design can be reused and only needs an approximate 20% update.

The company decides to have the new system in place within 1 year after starting the project. As the requirements are known, and no changes will be accepted during the project, their system integrator will use a traditional waterfall approach to develop the project.

The goal of the estimate is mainly to understand the price the selected system integrator will charge. The company is not interested in offshore or nearshore development. Onshore people that can work closely with the company staff is preferred as this will increase the chance of project success significantly.

The scope is to estimate a landing zone where the quote of the system integrators who are going to bid for the project will fall in. Internal effort for the company is excluded from the estimate.

So the estimation parameters are:

- Primary Programming Language: .Net
- Development method: Traditional (waterfall)
- Functional Size: 8,500 FP (re-use not possible)
- Functional Design: 80% reuse possible
- Other activities: Technical Design, Development + Programmer Test, Systems Test and support for the User Acceptance Test.

The company is going to implement the software into the Production environment itself, therefore implementation activities are not part of the estimate.

Select the right dataset as a reference

Selection Criteria

With these estimation parameters, the ISBSG Development & Enhancements repository [8.] is referenced. This can easily be done by applying a filter in Excel. In this case the 2019 release was used. The consultants used the following selection criteria:

<u>Data Quality Rating: ‘A’ or ‘B’</u>

All ISBSG submissions are given a Data Quality Rating by the ISBSG repository manager, based on a number of criteria and cross-checks. ‘A’ and ‘B’ are the best grades, C and D are lower quality rates.

Sometimes ‘C’ and ‘D’ data is still useful data, but when the dataset contains enough data points for statistical relevant analysis, it is recommended to use only ‘A’ and ‘B’ rated data.

<u>Year of Project > 2010</u>

The year that the project was finished is considered to have an influence on the productivity. Analysis however often shows that productivity has not improved over time. Often new frameworks or other development tools are introduced that promise to improve productivity, but in practice there is a learning curve in learning how to use these. Once the developers are skilled in a specific framework or tool, the next one is already introduced.

Over time there is not much productivity gain to be seen in the ISBSG data. Nevertheless, it is advisable not to use data that is too old.

<u>Development Type</u>

The ISBSG repository contains the following project types: New Development, Enhancement (which is a release or a set of sprints delivered to an existing system), Migration, Conversion and Re-development. At first glance it would make sense to use re-development, but this would only result in 1 datapoint, which is obviously too low. As a completely new technology is used for this redevelopment project, New Development is chosen as project type.

<u>Count Approach: ‘ÏFPUG 4+’ or ‘Nesma’</u>

In step 1, the functional size is measured in High-level function points. This method is almost as accurate as the Detailed Nesma method. The Detailed Nesma method and the detailed IFPUG4+ methods are 99% similar and projects measured in these two methods can therefore be used together in the same dataset.

As COSMIC, FiSMA and MK II are completely different methods, resulting in completely different sizes, data measured with these methods can’t be used in this particular estimate.

<u>Primary Programming Language contains .Net</u>

Unfortunately, the ISBSG data field Primary Programming Language is not a standardized field. Therefore there are sometimes some deviations. For instance there can be 'dotnet', 'C#.Net' and '.Net', which all reflect the same or very similar programming languages.

<u>Development methodologies contains 'Waterfall'</u>

The traditional Waterfall methodology is very different from the agile and iterative methodologies and therefore only waterfall projects are included in the dataset.

<u>Relative Size: > M1</u>

To make sure that very small and therefore incomparable projects are not selected, only Projects with a relative size greater than M1 are selected. This means that all projects under 300 FP are excluded from the dataset.

This selection only results in 3 data points with productivity data. This dataset is too small to use for a proper statistical analysis. Therefore some of the parameters need to be adjusted.

In a conversation with the lead architect new insights are given. It turns out that it is anticipated that a large part of the system is going to be realized in Java. This is a similar language to .Net when it comes to productivity. The consultant therefore decides to include Java in the Primary Programming Language field.

<u>Primary Programming Language contains .Net or Java</u>

As these are languages that show similar productivity figures over time, it is possible to use these languages together in the same dataset.

Project Delivery Rate

After the selection in the repository, the analysis for cost estimation purposes usually focuses on the field 'Normalised Level 1 PDR (ufp)'.

This field in the ISBSG dataset shows the Project Delivery Rate (PDR). The PDR (hours per FP) is the inverse of productivity, which is universally defined as Output/Input, which would result in a metric FP's per hour.

In practice this would result in very small numbers, like 0.00043 or 0.0023. These small numbers are hard for the human mind to process, therefore the inverse is chosen to use in the ISBSG repository. It's called PDR but in practice often Productivity is used instead.

The PDR is normalized to reflect the full ISBSG lifecycle. This contains the 6 general phases in a project lifecycle:

Plan, Specify, Design, Build, Test and Implementation.

The reason for this is that in many projects, the submitted data is grouped into fewer phase than these 6. This is due to the fact that these phases have not been carried out as part of the data submitted. The repository manager then normalizes the data of the phases that were submitted into the effort of the full lifecycle, using the percentages of the projects per project type that the ISBSG data shows.

These percentages are also published in a special analysis report [9.]. The repository manager always checks with the submitter if the phases were not carried out, or that maybe the data was grouped together in another phase.

Level 1 means that only the effort of the development team is considered, not the effort of eventual operators or other support staff.

Analyze the data in the dataset

This results in a selection of 24 data points. However, when analyzing the Project Delivery Rate (PDR = hours per FP) data, it shows that there is 1 obvious outlier with a Productivity Rate much higher (more than twice as high) than the other PDR's. This outlier is removed from the dataset.

The removal of outliers from a dataset should be done very carefully and there are techniques and methods available to do this in a statistically sound way. The use of these techniques however is beyond the scope of this case study.

Project Delivery Rate Analysis

To understand the data, it is advisable to use basic statistics to see the data distribution in the dataset. This is shown in table 1.

Statistic	Value
N	23
Min	1.1
P10	2.3
P25	4.6
Median	8.5
P75	11.0
P90	14.3
Max	21.6
Average	8.5

Table 1: Statistical Results

The 23 data points are distributed quite widely, but the Median and the Average have the same value, indicating that the distribution is not skewed left or right.

It's usually recommended for a landing zone to look at the 25th percentile (P25) of the dataset, the Median and the 75th percentile, unless there are good reasons to use other percentiles.

Normalization and landing zone determination

Now this data shows the data for the full project lifecycle. In this case, the data needs to be normalized because part of the Functional Design and the Implementation phase are out of scope for the estimate. The exact current percentages can be found in the special report, or can be analysed in the repository using the data.

For this analysis, the PDR is normalized with 10% to account for the activities that are not in scope of the estimate. This results in the following landing zone:

P25:	4.6	* 90% =	4.14 hours/FP
Median:	8.5	* 90% =	7.65 hours/FP
P75:	11.0	* 90% =	9.90 hours/FP

The estimated effort of the suppliers should therefore be in the following landing zone:

Low:	8,500 FP	* 4.14 hours/FP	= 35,190 hours
Likely:	8,500 FP	* 7.65 hours/FP	= 65,025 hours
High:	8,500 FP	* 9.90 hours/FP	= 84,150 hours

The consultancy company also uses a large database with market hourly rates. After a careful analysis of the database and the current hourly rates in the market, they predict that the average blended rate (average hourly rate of the function/roles involved in the project, including overhead) that the system integrators are going to be using in their quote will be approximately 80 EUR.

This means that the cost estimate landing zone for the effort for this project is

Low:	35,190 hours	* 80 EUR	= 2,815,200 EUR
Likely:	65,025 hours	* 80 EUR	= 5,202,000 EUR
High:	84,150 hours	* 80 EUR	= 6,732,000 EUR

Schedule Analysis

For the schedule, the constraint is that the system must be delivered within 1 year. Hence, the team size should be carefully chosen to ensure this duration is feasible. This means that the system integrator should be able to deliver 8,500 FP / 12 = 708 FP per month.

In this case, the same dataset is analyzed to understand the typical Speed (FP/Month) in these type of projects. This results in the following table:

Statistic	Value
N	34
Min	27.8
P10	30.6
P25	42.6
Median	71.5
P75	111.6
P90	171.9
Max	248.9
Average	88.0

The number of data points is higher for this metric, as there are a number of projects that have data of functional size and duration, but no effort.

It becomes clear in one moment that of the 34 projects in the dataset, the maximum speed delivered was 248.9 FP per month. This is much lower than the speed the customer expects from the system integrator. Therefore, the schedule constraint is not very realistic and suppliers should not adhere to this schedule as the project will probably result in a complete failure.

Since the schedule constraint is important for the customer, the consultancy company advises to prioritize the functionality using the MoSCoW method. This way the focus is to develop the 'Must Haves' first and develop the 'Should Haves' and other functionality in later releases.

Assuming that a professional service integrator is capable of delivering the P90 Speed, in this case 171.9 FP per month, in one year it should be able to deliver 12 * 171.9 FP = 2060 FP.

Using the MoSCoW technique, the customer prioritized the functionality approximately the following way:

• Must Haves:	2,000 FP
• Should Haves:	3,000 FP
• Could Haves:	2,000 FP
• Would Haves:	1,500 FP

In this case, only the Must Have functionality can be delivered in one year time, but only if the selected supplier will be able to deliver against the P90 Speed. In that case, the estimated effort of the suppliers should therefore be in the following landing zone:

Low:	2,000 FP	* 4.14 hours/FP	= 8,280 hours
Likely:	2,000 FP	* 7.65 hours/FP	= 15,300 hours
High:	2,000 FP	* 9.86 hours/FP	= 19,720 hours

This means that the cost estimate landing zone for the effort for this project is

Low:	8,280 hours	* 80 EUR	= 662,400 EUR
Likely:	15,300 hours	* 80 EUR	= 1,224,000 EUR
High:	19,720 hours	* 80 EUR	= 1,577,600 EUR

The consultancy firm advises the customer to use this landing zone to select the right supplier. Supplier should demonstrate the ability to deliver P90 Speed in the chosen technology and commit to deliver all the Must Have functionality within one year. Supplier offers that are lower than the Low estimate should not be chosen. The reason for this is given in Appendix A. The Customer may chose the cheapest offer in the landing zone, but should always ask for proof.

Appendix A: The Importance of a Realistic Estimate

A realistic estimate is one of the most important conditions for a successful project.

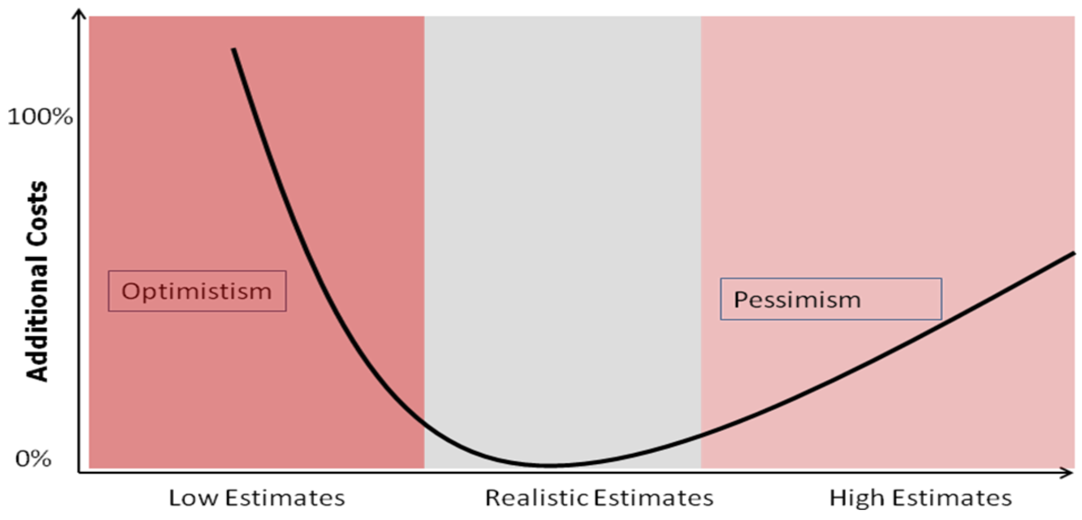
The estimate is the basis for:

- Business case
- Planning
- Proposal (outsourcing: fixed price / date)
- Financial result of the project… and the organization
- Claiming and releasing of resources
- Alignment between IT and business / customer
- Progress reports / dashboards
- The feeling of the team and the stakeholders

Without a realistic estimate, **the project is likely to fail!**

In the next figure, derived from Steve McConnell's book 'Software Estimation, Demystifying the black art' [10.], the effect of an optimistic and a pessimistic estimate are shown.

An optimistic (low) estimate can easily result in non-linear extra costs. At some point, it becomes clear that the project is in danger, and this usually results in extra management attention, more stress in the team, ineffective measures (e.g. adding people to a project makes it more expensive, not faster). It can also result in more defects and lower maintainability of the code, resulting in higher maintenance cost.



Pessimistic (high) estimates usually result in linear extra cost due to Parkinson's law [11.]. This means that if there are extra hours available, the people in the team will find a way to spend them somehow.

An example:

A realistic estimate for a given project would be 1000 effort hours.

If the project is estimated to cost 800 hours, then it may turn out to be not 1000 hours, but 1500 hours or worse in the end. The effect of underestimation and non-linear extra costs. In this case the project is unsuccessful regarding cost.

If the project is estimated for 1200 hours, it will probably also cost 1200 hours. In this case the project was successful regarding cost.

This example illustrates the importance of a realistic estimate and the determination of the right landing zone.

Appendix B: About the ISBSG

The ISBSG is a not-for-profit organization founded in 1997 by a group of national software metrics associations. Their aim was to promote the use of IT industry data to improve software processes and products.

ISBSG is an independent international organization that collects and provides industry data of software development projects and maintenance & support activities in order to help all organizations (commercial and government, suppliers and customers) in the software industry to understand and to improve their performance. ISBSG sets the standards of software data collection, software data analysis and software project benchmarking processes and is considered to be the international thought leader in these practices.

The ISBSG mission is to help YOU and your organization improve the estimation, planning, control and management of your IT software projects and/or maintenance and support contracts.

To achieve this:

ISBSG maintains and grows 2 repositories of IT software development/maintenance & support data. This data originates from trusted, international IT organizations and can be obtained for a modest fee from the website www.isbsg.org/project-data/

Help us to collect data

ISBSG is always looking for new data. In return for your data submission, you receive a free benchmark report that shows the performance in your project or contract against relevant industry peers.

Please submit your data through one of the forms listed on http://isbsg.org/submit-data/

Partners

This page will help you to find an ISBSG partner in your country http://isbsg.org/meet-isbsg-partners/

References

[1.] ISBSG, The International Software Benchmarking Standards Group, www.isbsg.org

[2.] IFPUG, the International Function Point User Group, www.ifpug.org

[3.] Nesma, the International organization for Benchmarking, Outsourcing, Productivity Measurement, Project Control, Estimation and Sizing, www.nesma.org

[4.] COSMIC, The Common Software Measurement International Consortium: www.cosmic-sizing.org

[5.] FiSMA, The Finnish Software Metrics Association, https://www.fisma.fi/in-english/introduction/

[6.] MK II, https://en.wikipedia.org/wiki/MK_II_FPA

[7.] Van Heeringen H.S., van Gorp, E.W.M. & Prins T.H., Functional size measurement – accuracy versus costs – is it really worth it?, Proceedings of the Software Measurement European Forum (SMEF), Rome, Italy, 2009.

[8.] ISBSG Development & Enhancements (D&E) repository – www.isbsg.org/project-data

[9.] ISBSG Special report - Planning projects - Phase Effort Ratios - https://www.isbsg.org/reports/

[10.] McConnell, S., Software Estimation, Demystifying the black art (Microsoft, 2006)

[11.] Parkinson's law: https://en.wikipedia.org/wiki/Parkinson%27s_law