Whitepaper 'Estimation techniques'

Author: Brian Johnson
Version: 24 February 2017

Many good practices allude to proper estimating of time (and therefore cost), curiously not many of them point you to examples of good practice. Rather like saying the US intelligence services are not, well, intelligent and (nudge, nudge) suggesting that you have a much better source of information. But you won't say who it is…..

1. Numerous methods for estimation

As with Risk management there are numerous methods for estimation. A quick web search will result in at least some of the following being thrown up as pertinent to software engineering including:

- SEER-SEM Parametric Estimation of Effort, Schedule, Cost, Risk (based on Brooks Law). The brilliant **Fred Brooks** stated in his book 'The Mythical Man-Month'; "*Adding manpower to a late software project makes it later.*" **Fred Brooks** made this even pithier by changing it to "*The bearing of a child takes nine months, no matter how many women are assigned.*" Genius.
- Function Point Analysis (FPA)
- The Planning Game (from Extreme Programming)
- Program Evaluation and Review Technique (PERT)
- Analysis Effort method
- Parametric Estimating
- CoCoMo
- SLIM
- Hand sizing (**Hans Dithmar**, a well-respected capacity management expert used to teach that useful estimates could be made by simply looking at what is current and similar in IT Operations and comparing that to what is being proposed).

And so on *ad infinitum*.

2. Value of estimations

The value to be gained from utilizing a functional sizing technique, such as Function Points, is primarily in the capability to accurately estimate resource needs (and therefore costs) of a project early in the development process. It is also possible to state Function Point Analysis (FPA) as a Need/Value Driver. Without accurate estimates, we might be committing the enterprise to enormous effort for little returned value.

The ISO recognizes FPA as a method to measure the functional size of an information system or service. The 'functional size' reflects the degree of functionality that is relevant to and recognized by the user in the business. FPA is independent of the technology used to implement the services or systems. The unit of measurement is of course, function points. So if your improvement is 'a few FPs' most of us recognize that development is unlikely to be too fancy and expensive, on the other hand if it is three million FPs then you might need a bigger business case. And good luck applying Agile methods.

The functional size is a useful input to discussions about application development costs, maintenance, and productivity upon implementation (based on usability criteria) and is an input to derive the coding dimensions for cost estimation. In this way establishing a budget is less guesswork, more science. The cost (and size…) of the application portfolio becomes clear to the business.

In the following sections, we will give an impression of Function Point Analysis and COCOMO.

3. Function Point Analysis

Function Point Analysis is often used by application developers and application managers and it is mentioned in the Application Services Library® (ASL). The function point method was originally developed by Allen J. Albrecht. A function point is a calculated, informed estimate of a unit of delivered functionality of a software project; 'delivered functionality', or 'functional size' might be (in a Scrum development environment for example) a single business function that achieves a defined benefit. Function points measure size in terms of the overall functionality in a system; the overall functionality of say, buying a government bond, or buying a book from an on-line retailer might (in fact, will…..) have many components that can be described as beneficial.

Each useful benefit might be a single function point (or may have to be sub-divided if it is not clear). Software developers are often required to provide early and accurate software project estimates (at least they should be). BIM professionals should be aware of various techniques that can be employed by developers to derive resource (and cost) estimates and may often have to use them personally when requests for new or improved services are made. These estimating issues are not new and the issue of accurate estimating, early in the life cycle, rarely has been adequately addressed and standardized and remains a headache for many—including Agile developers.

The ability to accurately estimate the overall effort in time and cost taken for an information services programme or project to deliver, is and has been, a serious problem for software engineers, developers, application managers and BIM professionals. In the same way that ITIL established a need (and the value of) the use of repeatable, clearly defined and well understood infrastructure management processes, it has become clear that a focus on software quality requires that the software development process must gain useable historical evidence that can be used for statistical estimation.

Types of Function Point Counts

Function Points can be counted at all phases of a software application development project from requirements up to and including implementation. For the most part, this type of count is associated with new development work. Scope creep can be tracked and monitored by understanding the functional size at all phases of a development project; that is different to putting an end to scope creep, and different to controlling scope creep. Frequently, this type of count is called a baseline function point count.

Enhancement Project Function Point Count

It is common to enhance software after it has been placed into production. Enhance is another descriptive term for 'fix' or 'eliminate botched requirements analysis' that you can use alongside 'improve'. This type of function point count is applied in attempts to estimate the scale of improvement projects or small scale changes where there is a need for some quantitative evidence about effort. All production applications evolve and are adapted; by the time a few years have passed they are like human beings, every part has been renewed over and over again, though not necessarily improved. By tracking metrics linked to the sizing of the work and associated costs to a database, useful sizing criteria can be created.

Application Function Point Count

Application counts are undertaken upon existing production applications. This sort of metric can be used to track maintenance hours *per* function point. This 'baseline count' can be used with overall software metrics for example total maintenance hours, perfective or adaptive maintenance hours etc.

Productivity

Software productivity is defined as hours/function points or function points/hours. This is the average cost to develop software or the unit cost of software. The common definition of productivity is the output-input ratio within a time period with due consideration for quality.

Effectiveness versus Efficiency

While efficiency refers to how well something is done, effectiveness refers to how useful something is. A plane is a very effective form of transportation, able to move people across long distances, to specific places, in miserable conditions and with the added 'value' of lousy food; but it may be considered not to transport people efficiently because of how it uses fuel. You can clear your conscience of course by paying a few more dollars to the airline to show that you understand the problem of the carbon dioxide blasted into the stratosphere, in the full understanding that the money will have no impact whatsoever on global warming. But your conscience is clear so we can move on.

Effectiveness then, is about doing the right task, efficiency is about doing things in an optimal way, for example doing it in the quickest or in the least expensive way. It could be the wrong thing to do of course, but at least it was done optimally, that should make you feel better.

Productivity implies effectiveness and efficiency in individual and enterprise performance. Effectiveness then is also the achievement of objectives. Efficiency is then the achievement of the ends using the minimum amount of resources.

To complete a function point count, knowledge of function point rules and software application documentation is necessary. A software quality or an application expert can improve the accuracy of the count.

4. Other techniques, for instance COCOMO

Much more information is available on the internet and from excellent books, not just on FPA but also on all of the other techniques. COCOMO (not to be confused with the rock band Kokomo, whose personnel probably do not compose songs about software estimating) for example, created by another genius Barry Boehm of Spiral model fame, is considered particularly useful by many software engineers.

According to Wikipedia, the method was first published in Boehm's 1981 book *Software Engineering Economics* as a model for estimating effort, cost, and schedule for software projects. It drew on a study of 63 projects at TRW Aerospace where Boehm was Director of Software Research and Technology. The study examined projects ranging in size from 2,000 to 100,000 lines of code and various programming languages. These projects were based on well-known waterfall model of software development which was the prevalent software development process at the time. In 1995 *COCOMO II* was developed and finally published in 2000 in the book *Software Cost Estimation with COCOMO II.* COCOMO II is better suited for the estimation of 21st century software development projects. The need for the new model came as software development technology moved from mainframe and overnight batch processing to desktop development, code reusability, and the use of COTS (commercial-off-the-shelf software components).

4. Consult your local guru

As with all of these important and related good practices mentioned, if in doubt consult an expert. Or if you prefer, leave it up to the Scrum developers to make sure that all of this complexity is taken into account when the business critical and highly complex services you need are analyzed and built iteratively and incrementally.

Bibliography

Boehm BW, c.s., *Software Cost Estimation with COCOMO II,* New Jersey: Prentice Hall PTR, 2000

Boehm BW, c.s., S*oftware Engineering Economics,* New Jersey: Prentice Hall PTR, 1981

Johnson B C, c.s., *BiSL® Next, a framework for business information management,* Van Haren Publishing, 2017