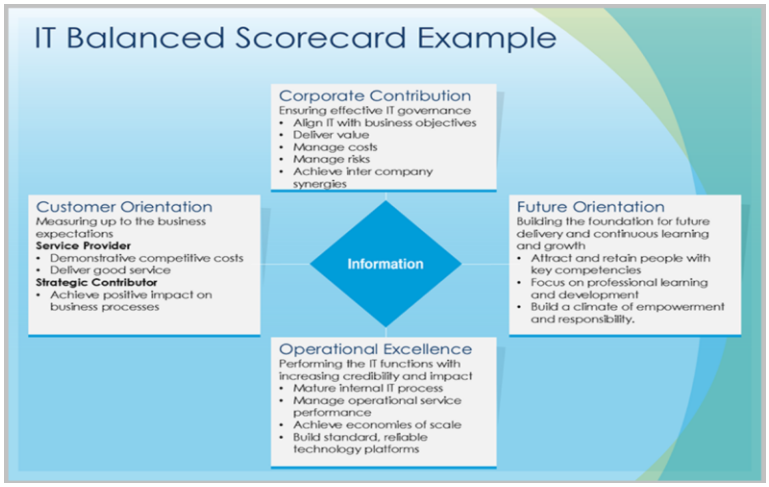
# CS3012 Software Measurement Report

Software engineering is known to be a complicated process. There are many moving parts to it and a lot of resources to manage. For software to be created and to be managed there needs to be a way for the progress on it to be measured. As once said by William Thompson, “When you can measure what you are speaking about, and express it in numbers, you know something about it, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind.” Going by that philosophy, programmers need a way of evaluating themselves in order to understand their coding performance as well as allowing the team that they are working on to assess their work.

## 1. Ways in Which Software Engineering can be Measured

Software metric is a measurement of a certain property of software or its specification. There are several software measurement methods.

### IT Balanced Scorecard




One of these methods is the IT Balanced Scorecard. This is an instrument that can be used for the measurement and management of software engineering efficiency as well as providing a link between software and the business needs of a company. The balanced scorecard was initially created by Robert S. Kaplan and David P. Norton in 1992 for a balanced and effective measurement and management of an entire organisation. It has become a popular tool, thanks to its

assumptions, propagated on a large scale by international consulting firms. In effect, as shown by the surveys of the IT Governance Institute, in 2005 there were 14 companies of which thirty percent of IT employees used the balanced scorecard. The main factors influencing the popularity of this standard are:

**Corporate Neatness in IT Management** – This is an area that is acknowledged for its importance in the development of software development management – especially for management in France and Japan. It has to be noted that the balanced scorecard allows the users to elaborate on four fields: future prospects, operational improvement as well as the contribution of stakeholders and the corporation in the development of the function of software engineering.

**Cutting costs and increase in efficiency** – The possibility of saving money while increasing efficiency is also of major importance. A well implemented balanced scorecard can monitor both the effectiveness of actions in programming as well as the effectiveness of contributions to organizational goals.

The original balanced scorecard can be adapted for software engineering purposes, the processes and projects. In order to do that, four perspectives are assigned with new goals and measurements in order to assess the current situation. These measurements would then have to be continually linked with the earlier established goals:

**Corporate Contribution** – Perspectives evaluate IT performance from the point of view of management and stakeholders. It's also a measurement of the investment input of IT in the development of the business. The main measurement challenges are therefore areas related to the strategic contribution of the software engineering business value. This perspective is characterized with the following measurements and mechanisms:

- The control of expenses on software engineering.
- The value of the company as a result of the software engineering.

**User Orientation** – An IT rating that is performed by an internal or external client. This rating is assigns the engineering process efficient from the perspective of

users. This point of view examines the client's satisfaction, efficiency of the creation of the application and the efficiency of a level of services. The measurements are:

- The preferred software supplier.
- The level of partnership with the users.
- The client's satisfaction.

**Operational Excellence** –This perspective rates the programming efficiency from the point of view from software development department managers. The main problems are the perfecting of processes of reacting to failures or breakdowns, the management of delays as well as safety. This perspective generally focuses on all issues related to the delivery and development of services. The measurement process focuses mainly on:

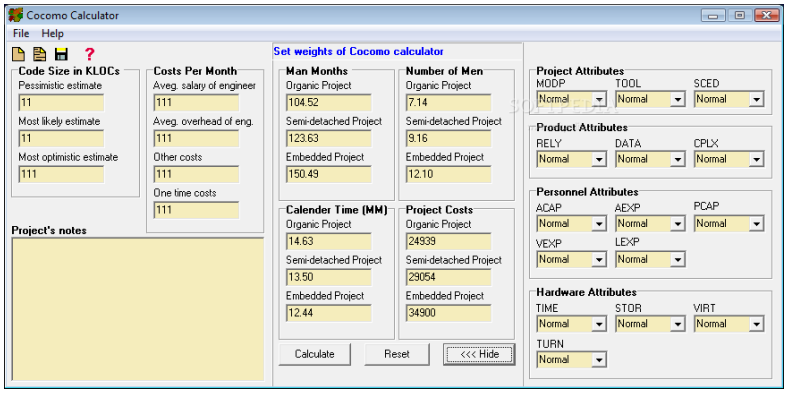
- An efficient development of software (a measurement can be the number of fixes that have to be done or an average number of delays).
- Efficient software operations.

**Future Orientation** – This perspective rates the software development efficiency from the point of view of the IT organization itself. It focuses on such things as the improvement of the capacity of service. The contributors are:

- Training of personnel.
- Expertise on the subject of software developers.
- Examination of new technologies.

## 2. Computational Platforms Available

## COCOMO (Constructive Cost Model)




This platform has been developed by Barry Boehm. It is a method of calculating the effort performed by software engineers. This method takes account of three models:

- **Application Composition Model** – This model is dedicated towards projects that use modern tools for the generation of user interfaces.
- **Early Design Model** – This model can be used in the early stage of the development of a project (at the point of the creation of an initial architecture).
- **Post-Architecture Model** – This model is the most detailed of all three. It can be used when the architecture of the system has been developed. It is also the most useful and the most used model. It is worth mentioning that the COCOMO method is based on the size of the software, which is much easier to estimate when a defined system architecture is already there. Using function points is one way of doing this.

The unit in which the effort is measured in COCOMO is PM (Person Month). One PM is a monthly time of work of one person on a given project, where days free of work are not taken into account but weekends are. The basic formula defining PM is: $\mathbf{PM_{nominal} = A * (Size)^{E}}$, where A and B are constants, size is a value given by KSLOC or thousand lines of source code, E is the factor of scale adjusting labour consumption which can be defined as:

$$E = B + 0.01 * \sum_{i=1}^{5} SF_i$$

The post-architecture model is adjusts normal estimation by using Effort Multipliers (EM):

$$PM_{adjusted} = PM_{nominal} * \prod_{i=1}^{16} EM_i$$

$$PM_{adjusted} = A * (Size)^E * \prod_{i=1}^{16} EM_i$$

The constant coefficients A and B have been assigned and calculated based on 161 projects and they are:

A = 2.94, B = 0.91

It has also been discovered that for an average project, EMs are equal to 1 and so the product of the EMs is also 1. This means that for an average project, the formula is:

$$PM_{adjusted} = 2.94 * (Size)^E$$

In the COCOMO method, 5 factors of the project scale were specified:

- **Typicality** – This determines whether the developed project is typical for the given organization and whether it contains elements of innovation.
- **Elasticity** – This determines if there exists a need for compliance with the document of requirements or with external interfaces.
- **Risk Management** – This takes into account the existence of a plan for risk management, job classification, budget determination and other risk management tools.
- **Team Cohesion –** This takes into account the difficulties of synchronization between the members of the team developing the project.
- **Process Maturity –** The maturity rating of the process based on the CMM model.

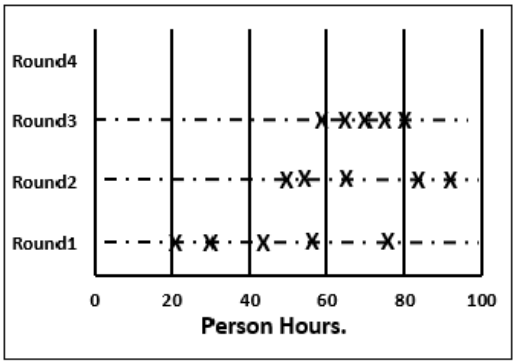
Depending on the evaluation of certain factors in the examined project, their influence is assigned on a scale from very low to extra high.

| | Very Low | Low | Nominal | High | Very High | Extra High |
|---|---|---|---|---|---|---|
| Typicality | 6.20 | 4.96 | 3.72 | 2.48 | 1.24 | 0.00 |
| Elasticity | 5.07 | 4.05 | 3.04 | 2.03 | 1.01 | 0.00 |
| Risk Management | 7.07 | 5.65 | 4.24 | 2.83 | 1.41 | 0.00 |
| Team Cohesion | 5.48 | 4.38 | 3.29 | 2.19 | 1.10 | 0.00 |
| Process Maturity | 7.80 | 6.24 | 4.68 | 3.12 | 1.56 | 0.00 |

In order for this method to be effective, the software size should be at least a dozen hundred lines of code.

## 3. The Algorithmic Approaches Available

### Wideband Delphi




The easiest method of estimation is based on the knowledge and intuition of experts. One such method that uses this approach is the Wideband Delphi. Relying on knowledge is a commonly used practice. However, the negotiation process is a wider circle of experts can bring certain problems related to human psychology. The goal of the estimation relying on a larger number of experts is to minimalise an eventual subjective mistake. There are of course problems with it.

- Let's take for example that there is a hierarchical dependency between the experts, like for example the dependency between a boss and their employee. In this case, disagreeing with your boss would not be an easy task and influences on the objectivism of this expert.
- A similar problem is the difference of authority. If one person in the group is for example a respected scientist, it can influence the others by using the person's authority. Similarly to the previous point, this makes negotiations like equals harder.
- The human nature makes us unable to admit a mistake. This is also affecting the negotiation process. It can happen that a person will argue for the correctness of their argument as a result of their pride, even if they're wrong.

In order to get rid of negotiation problems, the Delphi Method has been developed at the end of the 1940s by Rand Corporation. The idea relies on the individual, independent evaluation by several experts. By using the Delphic method we are trying to reach a consensus. The algorithm reached a great popularity which inspired several modifications. One of the most well known versions is the Wideband Delphi.

The algorithm of the Wideband Delphi is composed of seven steps:

- The first stage is equipping the experts with appropriate materials that outline the project as well as a special estimation form. Each of the experts also receives an amount of time to analyse the documentation.
- The next step is the exchange of views about the project among the experts. The experts share their notes and observations taken from reading the documentation.
- After doing the individual and group analysis of the collected documentation, it is then possible to perform voting. Each expert specifies their own estimation of labour intensity. It is of utmost importance that the voting is done anonymously.
- The experts' estimations are given to a moderator that oversees the negotiation process. The moderator's job is an appropriate development

of the results of the voting process and the transfer of these results to the estimation forms of certain experts. An estimation report made by the moderator is unique for each expert. Apart from informational data such as the name of the expert, date of voting or the name of the project, the report contains the location of the expert's assessment against the background of other estimates. The expert's own rating is shown, with the ratings of the remaining experts as well as an average estimation. This allows the verification of an expert's own position and their critical evaluation. An expert has the possibility of comparing the shown results with the average as well as other experts' results. The specific information connecting the result to an expert is not provided. The analysis of the data shown in the report is meant to push the author to reflect upon their own results and thinking more about the problem.

- After reanalysing the estimation reports, the experts discuss among themselves again. It is an opportunity to reanalyse the factors that influence the labour intensity. In order to retain the anonymity it is advised not to give exact estimates, but instead to outline the problems on which the experts based their decisions.
- The entire procedure is repeated until the evaluations of the experts are similar enough.
- The final evaluation.

If a common rating cannot be reached for all the experts, the average weighted value has to be used in order to get the final rating. The average takes account of a pessimistic assessment and an optimistic assessment with a weight of 1 and an average with a weight of 4:

**Evaluation = (P + 4A + O) / 6**

**P** – pessimistic assessment, **A** – average assessment, **O** – optimistic assessment

## 4. Ethics Surrounding This Kind of Analytics

It has been noted that by several software engineers that these software measurement methods can have more harmful repercussions than good effects. Other software engineers have noticed that the software metrics have turned into an indispensible part of the software engineering process. The impact of software metrics has displayed worrying results about the psychology of a programmer with regards to their performance, mainly because of stress, nervousness and attempts to mislead the measurements. On the other hand, others find it to have a positive effect on programmers who don't feel as undervalued. Several software metrics are regarded as being too imprecise for use in software development since it is a relatively new and barely discovered field. However, software metrics are widely used by organizations such as NASA and academic institutions.

By Damian Debny

Student No.: 15315535

## References:


http://www.citeulike.org/group/3370/article/12458067

http://www.nextlearning.nl/wp-content/uploads/sites/11/2015/02/McKinsey-on-Impact-social-technologies.pdf

https://www.researchgate.net/profile/Jan_Sauermann2/publication/285356496_Network_Effects_on_Worker_Productivity/links/565d91c508ae4988a7bc7397.pdf

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.63.2683&rep=rep1&type=pdf

https://en.wikipedia.org/wiki/Software_metric