

Measuring Engineering Report

The life of a software engineer is never static, as software is always changing improving and adapting to modern times. This consistent change comes with more responsibility that is placed upon a software engineer for the actions they carry out during the software engineering process. Throughout my analysis I will detail the ways in which a software engineers , engineering process can be measured and assessed in measurable data , an overview of the platforms available to so , the different algorithmic approaches available and the ethics surrounding the given analysis.

Firstly why do we measure software and the engineering process?

We do so in order to :

(i)

- Establish the quality of the current product or process.
- To predict future qualities of the product or process.
- To improve the quality of a product or process.
- To determine the state of the project in relation to budget and schedule

A software engineering process can use measurements and metrics.

A measurement is an indication of the size, quantity, amount or dimension of a particular attribute of a product or process. For example the *number of errors in a system* is a measurement.(i)

A Metric is a measurement of the degree that any attribute belongs to a system, product or process. For example the *number of errors per person hours* would be a metric.(i)

Since we're looking at the *process* of software engineering, so we will be focusing on the Metric measurement. Which in itself can be further subdivided into *product metrics* and *process metrics*.

Product metrics are used to asses the state of the product, tracking risks and discovering potential problem areas. The team's ability to control quality is assessed.

(i)

Process metrics focus on improving the long term process of the team or organisation.(i)

Once again we are only looking at the process of software engineering, so we will be focusing on the process metrics.

Such as :

- Balanced scorecard
- SLOC (Source lines of code)
- Code Coverage
- Cohesion
- Commits
- Coupling

Balanced scorecard

(ii) Balanced scorecard is an example of a closed-loop controller or cybernetic control applied to the management of the implementation of a strategy.^[3] Closed-loop or cybernetic control is where actual performance is measured, the measured value is compared to a reference value and based on the difference between the two corrective interventions are made as required. Such control requires three things to be effective:

- a choice of data to measure,
- the setting of a reference value for the data,
- the ability to make a corrective intervention.^[3]

Within the strategy management context, all three of these characteristic closed-loop control elements need to be derived from the organisation's strategy and also need to reflect the ability of the observer to both monitor performance and subsequently intervene – both of which may be constrained.

Source Lines of Code

There is also the physical counting of the lines of code in a text of the a programs source code.

Code Coverage

Code Coverage is a measurement of how many lines/blocks/arcs of your code are executed while the automated tests are running. Generally measured as a percent, a developer will look to increase this. A program with high test coverage, measured as a percentage, has had more of its source code executed during testing which suggests it has a lower chance of containing undetected software bugs compared to a program with low test coverage.

Cohesion (iii)

Cohesion refers to the *degree to which the elements inside a module belong together*. In one sense, it is a measure of the strength of relationship between the methods and data of a class and some unifying purpose or concept served by that class. In another sense, it is a measure of the strength of relationship between the class's methods and data themselves.

Cohesion is an ordinal type of measurement and is usually described as "high cohesion" or "low cohesion". Modules with high cohesion tend to be preferable, because high cohesion is associated with several desirable traits of software including robustness, reliability, reusability, and understandability. In contrast, low cohesion is associated with undesirable traits such as being difficult to maintain, test, reuse, or even understand.

Commits

This measures the amount of commits they are committing to their repository and the quantity attached.

Coupling

The measure of the degree at which different classes are related, different classes should be independent of one and other. The higher the coupling measurement, the harder it will be to change something in one class without affecting another. Developers will look to keep a low coupling value, this will make classes more independent and easier to change something in the class without causing much conflict

Platforms available

ESM (iv)

ESM is robust strategy management software which help organizations to better understand and manage strategy. ESM enables professionals in articulating organizational mission, vision, and values to clarify the foundation of strategy. ESM's PESTEL Analysis helps analyst in understanding political, economic, social, technological, environmental, and legal factors that impact the strategy .ESM's SWOT Analysis help analysts in capturing organizational strengths, weaknesses, opportunities, and threats. ESM enables decision makers in taking decisions on the basis of latest data with real time Strategy Maps, dashboards, and fully automated

Balanced Scorecards. ESM enables managers to define top-level organizational goals, KPIs, and strategic initiatives directly...

ESM®

Strategy

BSC & Maps

Perspectives


Themes

Objectives

Measures

Initiatives

Action Items

 **Summary**
Scorecard: 1. OTM Corporate Scorecard

Balanced Scorecard (BSC)

Strategy Map


Governance Calendar

Scorecard Hierarchy

Scorecard Relationship

Alignment

Help





























































 Refresh

Parent Scorecard

1. OTM Corporate Scorecard

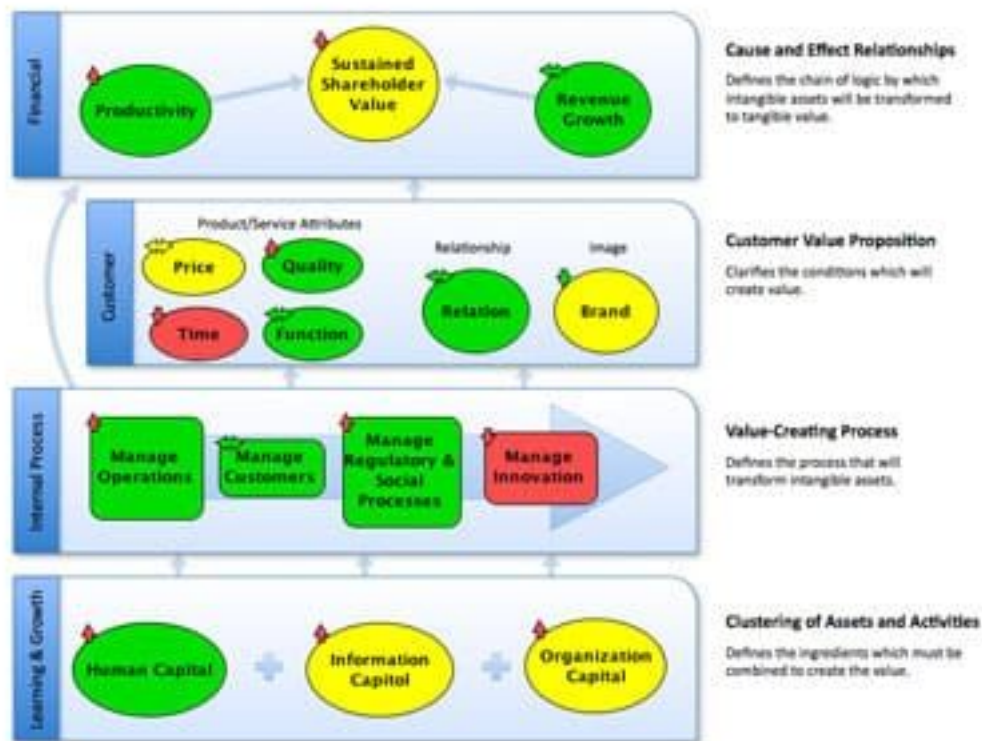
Element

Objective

1. OTM Corporate Scorecard	2. OTM Europe Manufacturing	3. OTM Americas Manufacturing	4. OTM Asia Pacific Manufacturing	5. OTM Finance Scorecard	
Perspective	Objective	Objective	Objective	Objective	
Financial Perspective	 F1. Achieve positive EBITDA contributions	 F1. Positive EBITDA Contribution  F1. Positive EBITDA Contribution	 F1. Positive EBITDA Contribution	 F1. Positive EBITDA Contribution	 F1. Positive EBITDA Contribution  I10 Optimize Cash Flows
	 F2. Maximize earnings by selling the right product mix	 F2. Maximize Earnings by Selling the Right Product Mix in Our Region	 F2. Maximize Earnings by Selling the Right Product Mix in Our Region	 F2. Maximize Earnings by Selling the Right Product Mix in Our Region	 F2. Maximize Earnings by Selling the Right Product Mix in Our Region
	 F3. Improve average net selling price	 F3. Improve Average Net Selling Price of Core Products	 F3. Improve Average Net Selling Price of Core Products	 F3. Improve Average Net Selling Price of Core Products	 F3. Improve Average Net Selling Price of Core Products
	 F4. Manage G&A costs	 F4. Reduce total cost of production			 F4. Manage G&A Costs
	 F5. Reduce total cost of production	 F5. Reduce raw material expenses	 F5. Reduce total cost of production	 F5 Reduce total cost of production  F5 Reduce total cost of production	 F5. Reduce total cost of production  F5. Reduce total cost of production
Customer Perspective	 C1. Seamless and Competitive Service and Price	 C1. Seamless and Competitive Service and Price		 C1. Seamless and Competitive Service and Price	 C1. Top level, Just in Time Internal Customer Service
	 C2. Give Me Top Quality Manufacturing Excellence	 C2. "Give Me Top Quality Manufacturing Excellence"  C3. Manufacturing solutions not just products	 C2. "Give Me Top Quality Manufacturing Excellence"	 C2. Manufacturing Excellence	 C2. Manufacturing Financial Report Accountability  C2. Manufacturing Financial Report Accountability
	 I01. Continually analyze customer profitability	 I01. Continually analyze customer profitability	 I01. Continually analyze customer profitability	 I01. Continually analyze customer profitability	 I01. Continually Analyze Customer Profitability
	 I02. Increase share with profitable customers	 I02. Increase share with profitable customers	 I02. Increase share with profitable customers	 I02. Increase share with profitable customers	 I02. I.D and Build Potential Growth Opportunities
	 I03. Continually improve customer service	 I03. Continually improve customer service	 I03. Continually improve customer service	 I03. Continually improve customer service	 I03. Continually Improve Transactional Efficiency While Maintaining Accuracy
	 I04. Reduce claims by 50%	 I04. Reduce claims	 I04. Reduce claims	 I04. Reduce claims  I04. Reduce claims	 I06. I.D and Mitigate Potential Enterprise Risk  I06. I.D and Mitigate Potential Enterprise Risk

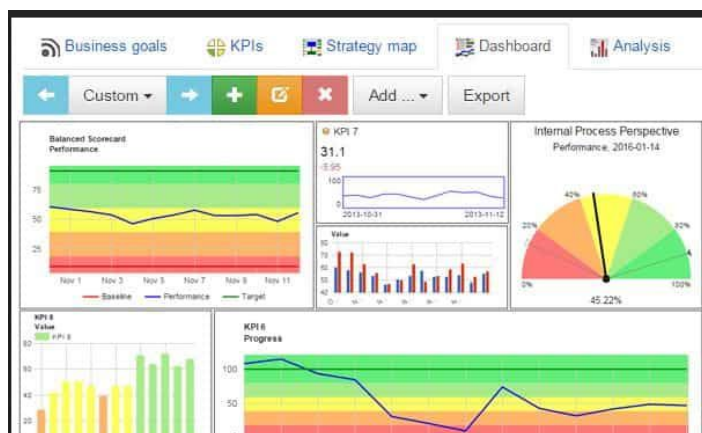
Spider Strategies (iv)

Spider Strategies is Web-based metric tracking software that powers balanced scorecards and gorgeous performance dashboards. Agile, smart, and convenient. Software developed by spider strategies enables businesses to see how well they are performing. They can see their key performance indicators and metrics. They can view their balanced score cards. Once companies view how they are performing by looking into various metrics, companies can devise strategies and alternative plans if they find out something is going wrong. The product does not require installation of special equipment or servers to run the software.



BSC Designer (iv)

BSC Designer is a balanced scorecard software available as a cloud-based service and Windows desktop application. BSC Designer is a user-friendly tool that helps business consultants in building business scorecard with strategy maps and KPIs in just few minutes. BSC designer enables business consultants to create professional strategy maps. BSC Designer is easy to install, configure and it helps a lot with Balanced Scorecard, strategy maps, and KPIs. BSC designer helps business executives to add important business goals and to specify cause-and-effect connections between them. BSC Designer enables users to create strategic themes.



USC CodeCount (v)

CodeCount automates the collection of source code sizing information. The CodeCount toolset utilizes one of two possible source lines of code (SLOC) definitions, physical or logical. UCC is a unified and enhanced version of the CodeCount toolset. It is a code counting and differencing tool that unifies the source counting capabilities of the previous CodeCount tools and source differencing capabilities of the Difftool (which is now replaced by UCC). It allows the user to count, compare, and collect logical differentials between two versions of the source code of a software product. The differencing capabilities allow users to count the number of added/new, deleted, modified, and unmodified logical SLOC of the current version in comparison with the previous version. With the counting capabilities, users can generate the physical, logical SLOC counts, and other sizing information such as comment and keyword counts of the target program.

Languages: Ada, Assembly, C, C++, COBOL, Fortran, Java, JOVIAL, Pascal, PL1

SLOC Metrics (v)

SLOC Metrics measures the size of your source code based on the Physical Source Lines of Code metric recommended by the Software Engineering Institute at Carnegie Mellon University (CMU/SEI-92-TR-019). Specifically, the source lines that are included in the count are the lines that contain executable statements, declarations, and/or compiler directives. Comments, and blank lines are excluded from the count. When a line or statement contains more than one type, it is classified as the type with the highest precedence. The order of precedence for the types is: executable, declaration, compiler directive, comment and lastly, white space.

Languages: ASP, C, C++, C#, Java, HTML, Perl, Visual Basic

Cobertura (vi)



Based on the now-retired jCoverage, Cobertura is another popular open source code coverage tool. It lets you execute tasks via Ant, and Maven, or via the Cobertura CLI. Just like EMMA, Cobertura is also not actively maintained and doesn't support the

current Java versions. It also is embedded into multiple QA tools, and still lives on in other forms.

Programming languages: Java

Coverage.py (vi)



Coverage.py is a code coverage tool for Python. It monitors your Python programs, notes which parts of the code have been executed, and analyzes the source to identify code that could have been executed but was not.

Programming Languages: Python

Algorithmic Approaches

Weighted Micro Function Points (WMFP)

WMFP is a modern software sizing algorithm. It produces more accurate results than traditional software sizing methodologies¹, while requiring less configuration and knowledge from the end user, as most of the estimation is based on automatic measurements of an existing source code.

Unlike the previously mentioned SLOC WMFP uses a parser to understand the source code breaking it down into micro functions and derive several code complexity and volume metrics, which are then dynamically interpolated into a final effort score. It is also compatible with the waterfall software development life cycle methodology, due to its differential analysis capability made possible by its higher-precision measurement elements. **(vii)**

Calculation

"The WMFP algorithm uses a three-stage process: function analysis, APPW transform, and result translation. A dynamic algorithm balances and sums the measured elements and produces a total effort score. The basic formula is:

$$\sum (W_i M_i)^{\frac{1}{D_q}}$$

M = the source metrics value measured by the WMFP analysis stage

W = the adjusted weight assigned to metric M by the APPW model

N = the count of metric types

i = the current metric type index (iteration)

D = the cost drivers factor supplied by the user input

q = the current cost driver index (iteration)

K = the count of cost drivers

This score is then transformed into time by applying a statistical model called average programmer profile weights (APPW) which is a proprietary successor to COCOMO II 2000 and COSYSMO. The resulting time in programmer work hours is then multiplied by a user defined cost per hour of an average programmer, to produce an average project cost, translated to the user currency.” (vii)

Cyclomatic complexity (viii)

Cyclomatic complexity is used to quantitatively indicate the complexity of a program. It is “computed using the control flow graph of the program: the nodes of the graph correspond to indivisible groups of commands of a program, and a directed edge connects two nodes if the second command might be executed immediately after the first command. Cyclomatic complexity may also be applied to individual functions, modules, methods or classes within a program.” (viii)

For example, if the source code contained no control flow statements (conditionals or decision points), the complexity would be 1, since there would be only a single path through the code. However, if the code had one single-condition IF statement, there would be two paths through the code: one where the IF statement evaluates to TRUE and another one where it evaluates to FALSE, so the complexity would be 2. Two nested single-condition IFs, or one IF with two conditions, would produce a complexity of 3.

Fig 1 - A control flow graph of a simple program

Fig 2 - Similar to Fig 1 but represented using the alternative formulation, where each exit point is connected back to the entry point.

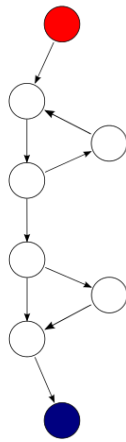


Fig 1 (viii)

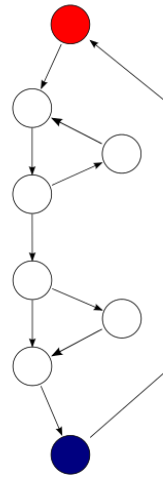


Fig 2 (viii)

Ethical much?

Software engineers in all levels and disciplines abide by some form of ethical code / code of conduct be it company mandated , personal morals or both. The thing is most analytical software , are just that software .They may not always be written with ethics in mind. This brings up questions of privacy , data management / protection and general security.

Tools such as BSC for Balance scorecards , github for commit history and algorithms like WMFP keep track and store data of engineers performances , commits, code , comments, documentation and by default name and other personal information.

One could argue that if this data is on an open source application like github then the individual engineer is aware of the open nature of the information so privacy in terms of what the software engineer in question chooses to make public they do so with the preemptive knowledge that what they chose to release it and it is free for all to see and with the way of the internet , it's there more or less forever.

Now on the flip side would be in house / less open platforms like BSC , Cobertura, Coverage.py etc the concept of ;

- privacy as in what is public information and what isn't,
- data management / protection - how an engineers data and information is managed , stored and how or if it is deleted ,
- security - who within a company gets access to said data and does the engineer know of all the potential people who can access their data . If not , why not?

Not only do the above points have to be taken into consideration but we also have to account for the fact that software development is a complex process,"with high variance on both methodologies and objectives, it is difficult to define or measure software qualities and quantities and to determine a valid and concurrent measurement metric, especially when making such a prediction prior to the detail design. Another source of difficulty and debate is in determining which metrics matter, and what they mean", (i) as more simplistic approaches like SLOC may be more harmful in some instances.

To conclude ,while you could argue that imperfect quantification measurements is better than none , in my view it is not always possible or fully accepted to measure the full extent of a software engineering process using data , tools to analysis this data or algorithms . As it may not satisfy ethical concerns (that should always be asked), and the varied amount of options available to measure and process data also incurs that the output from one may differ to that of another which may skew the overall outlook of the given engineering process.

References

(viii) CYCLOMATIC COMPLEXITY

In-text: (En.wikipedia.org, 2017)

Your Bibliography: En.wikipedia.org. (2017). *Cyclomatic complexity*. [online]

Available at: https://en.wikipedia.org/wiki/Cyclomatic_complexity [Accessed 4 Dec. 2017].

(vii) WEIGHTED MICRO FUNCTION POINTS

In-text: (En.wikipedia.org, 2017)

Your Bibliography: En.wikipedia.org. (2017). *Weighted Micro Function Points*.

[online] Available at: https://en.wikipedia.org/wiki/Weighted_Micro_Function_Points [Accessed 4 Dec. 2017].

(vi) THE ULTIMATE LIST OF CODE COVERAGE TOOLS: 25 CODE COVERAGE TOOLS FOR C, C++, JAVA, .NET, AND MORE

In-text: (Stackify, 2017)

Your Bibliography: Stackify. (2017). *The Ultimate List of Code Coverage Tools: 25 Code Coverage Tools for C, C++, Java, .NET, and More*. [online] Available at: <https://stackify.com/code-coverage-tools/> [Accessed 4 Dec. 2017].

(v) LOC METRICS - ALTERNATIVE TOOLS

In-text: (Locmetrics.com, 2017)

Your Bibliography: Locmetrics.com. (2017). *LOC Metrics - Alternative Tools*. [online] Available at: <http://www.locmetrics.com/alternatives.html> [Accessed 3 Dec. 2017].

(iv) SOFTWARE, 2.

20 Free and Top Balanced Scorecard Software in 2017

In-text: (Software, 2017)

Your Bibliography: Software, 2. (2017). *20 Free and Top Balanced Scorecard Software in 2017*. [online] Predictive Analytics Today. Available at: <https://www.predictiveanalyticstoday.com/open-source-balanced-scorecard-software/> [Accessed 3 Dec. 2017].

(iii) COHESION (COMPUTER SCIENCE)

In-text: (En.wikipedia.org, 2017)

Your Bibliography: En.wikipedia.org. (2017). *Cohesion (computer science)*. [online] Available at: [https://en.wikipedia.org/wiki/Cohesion_\(computer_science\)](https://en.wikipedia.org/wiki/Cohesion_(computer_science)) [Accessed 3 Dec. 2017].

(ii) BALANCED SCORECARD

In-text: (En.wikipedia.org, 2017)

Your Bibliography: En.wikipedia.org. (2017). *Balanced scorecard*. [online] Available at: https://en.wikipedia.org/wiki/Balanced_scorecard [Accessed 2 Dec. 2017].

(i) SOFTWARE METRICS AND MEASUREMENT - WIKIVERSITY

In-text: (En.wikiversity.org, 2017)

Your Bibliography: En.wikiversity.org. (2017). *Software metrics and measurement - Wikiversity*. [online] Available at:
https://en.wikiversity.org/wiki/Software_metrics_and_measurement [Accessed 2 Dec. 2017].