CHAPTER 18

Benchmarking: Comparing Apples to Apples

Frank Vogelezang, METRI, The Netherlands
Harold van Heeringen, METRI, The Netherlands

Introduction

For almost every organization, software development is becoming more and more important. The ability to develop and to release new functionality to the users and customers as fast as possible is often one of the main drivers to gain a competitive edge. However, in the software industry, there is a huge difference in productivity between the best and worst performers. Productivity can be a crucial element for many organizations (as well as cost efficiency, speed, and quality) to bring their competitiveness in line with their most relevant competitors.

Benchmarking is the process of comparing your organization's processes against industry leaders or industry best practices (outward focus) or comparing your own teams (inward focus). By understanding the way the best performers do things, it becomes possible to

- Understand the competitive position of the organization
- Understand the possibilities for process or product improvement
- Create a point of reference, a target to aim for

Benchmarking gives insight into best practices, with the aim to understand if and how one should improve to stay or become successful. Software development benchmarking can be done on any scale that is comparable: a sprint, a release, a project, or a portfolio.

C. Sadowski and T. Zimmermann (eds.), *Rethinking Productivity in Software Engineering*,
https://doi.org/10.1007/978-1-4842-4221-6_18

The Use of Standards

Benchmarking is all about comparing. A well-known phrase is "Comparing apples to apples and oranges to oranges." One of the key challenges in the software industry is to measure productivity of completed sprints, releases, projects, or portfolios in such a way that this information can be used for processes such as estimation, project control, and benchmarking. But how can we compare *apples to apples* in an industry that is immature when it comes to productivity measurement?

The economic concept of productivity is universally defined as output/input. In the context of productivity measurement in software development, input is usually measured in effort hours spent. Although it's important to define the right scope of activities when benchmarking, it's just as important to measure the output of a sprint, release, or project in a meaningful way. To be able to benchmark productivity in an "apples to apples" way, it's crucial that the output is measured in a standardized way. An important aspect of standardization is that the measurement is repeatable, so different measurers attribute the same number to the same object. In practice, many measurement methods are being used that are not standardized. Because the output is not standardized, the same number may relate to different aspects, or the same object gets different ratings. This means that the productivity information is not comparable and therefore not useful in benchmarking. Examples of these popular, but unstandardized measurement methods are lines of code (LOC) and all variants, use case points, complexity points, IBRA points, and so on. Also, the story point, which is popular in most agile development teams, is not standardized and therefore can't be used in benchmarking across teams or organizations.

At this moment, only the standards for functional size measurement (the main ones being Nesma, COSMIC, and IFPUG) comply with demands for standardized measurement procedures and intermeasurer repeatability to produce measurement results that can be compared across domains to benchmark productivity.

Functional Size Measurement

Functional size is a measure of the amount of functionality provided by the software, derived by assigning numerical values to the user practices and procedures that the software must perform to fulfill the users' needs, independent of any technical or quality considerations. The functional size is therefore a measure of what the software must do, not how it should work. This general process is described in the ISO/IEC 14143 standard.

The COSMIC method measures the occurrences of Entries, Exits, Reads, and Writes (Figure 18-1).

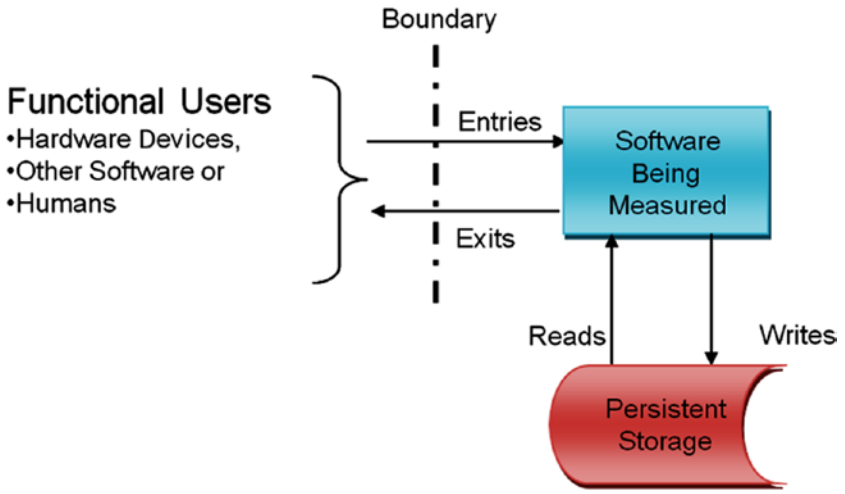

Figure 18-1. *The base functional components for the COSMIC method: Entry, Exit, Read, and Write*

COSMIC is a second-generation functional size measurement method. Most first-generation methods also assign values to data structures. This limits their use in software that processes events. See also Chapter 17 for more extensive information about functional size measurement.

To benchmark productivity across projects in a comparable way, these base parameters are now available:

- *Output*: Functional size measured in a standardized way
- *Input*: Effort hours spent for agreed activities in scope

In practice, the productivity formula (output/input) usually results in numbers of *function points per effort hour* smaller than 1. Because humans are not computers and people can more easily understand and interpret numbers greater than 1, the use of the inverse is more commonly used in software benchmarking. This inverse is called the *product delivery rate* (PDR), defined as Input/Output, or *effort hours per function point* delivered. This is an outcome-oriented way of assessing productivity. See Chapter 8 for more details on assessing productivity.

When the productivity is measured in a standardized way, for benchmarking purposes it needs to be compared to relevant peer groups in the industry. The most relevant source for peer group data is the International Software Benchmarking Standards Group (ISBSG). This not-for-profit organization collects data from the industry, based on standardized measures, and provides this data in an anonymized data set in easy-to-use Excel sheets. For productivity benchmarking, this is the main resource available for practitioners in the industry. The Development & Enhancements repository currently (February 2019) contains more than 9,000 projects, releases, and sprints, most of them having a PDR in one of the functional size measurement methods mentioned earlier.

Reasons for Benchmarking

Benchmarking is often used to understand the organization's capabilities in relation to industry leaders or competitors. This most common type of benchmarking has an outward focus. The objective is usually to find ways or approaches to reach the level of productivity of the industry leaders or to improve productivity in such a way that competitors can be outperformed.

Benchmarking can also be done with an inward focus. The most common example of this type of benchmarking is the comparison of velocity in the last sprint to the velocity in previous sprints. The objective is usually to learn from earlier sprints what can be improved to reach a higher velocity. In Chapter 3, Amy J. Ko performs a thought experiment to argue that we should focus on good management rather than productivity measurement. The effects that good management will have on productivity are true for most successful organizations we have encountered. But the only way to prove that good management brings a higher productivity is…benchmarking. And benchmarking requires measuring productivity.

Another use of benchmarking is the determination of a so-called landing zone by tendering organizations. A landing zone is a range of the minimum, average, and maximum prices that can be expected for the scope offered for tender. These ranges are based on market experience. With this use of benchmarking data, bidding companies are benchmarked in advance.

Examples of a scope that is offered for tender are

- A portfolio of applications to be maintained
- A new bespoke software solution to be developed
- A number of applications to be ported to a cloud platform

We have seen tenders that exclude bids that are outside the landing zone. How the source data for such a landing zone can be obtained is described in the section "Sources of Benchmark Data." The objective is to determine where they expect the price offers of the bidding companies will fall.

A Standard Way of Benchmarking

In 2013, the ISO published an international standard describing the industry best practice to carry out IT project performance benchmarking: ISO/IEC 29155 Information technology project performance benchmarking framework. The standard consists of five parts (Figure 18-2).

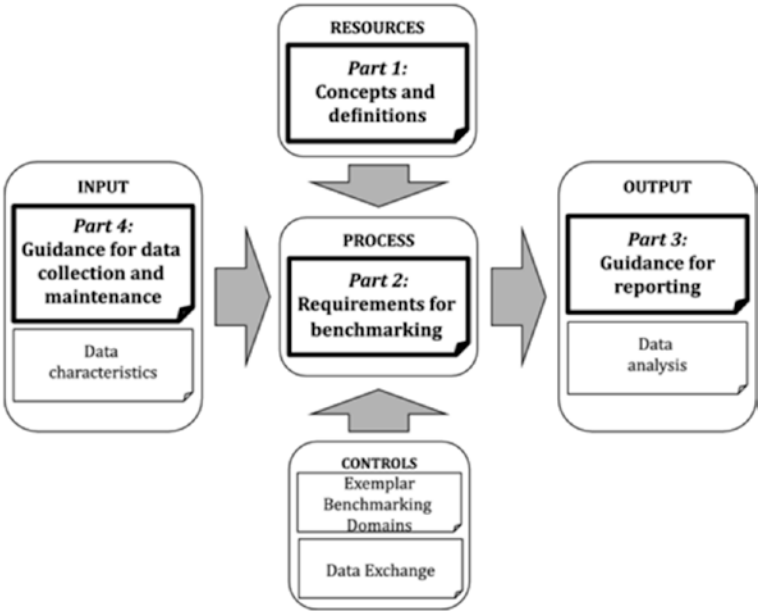

Figure 18-2. *ISO/IEC 29155 structure*

This standard can guide organizations that want to start benchmarking their IT project performance to implement an industry best practice benchmarking process in the following ways:

- By offering a standardized vocabulary of what is important in setting up a benchmark process
- By defining the requirements for a good benchmarking process

- By giving guidance on reporting, before the input part is put in place
- By giving guidance on how to collect the input data and how to maintain the benchmark process
- By defining benchmarking domains

The order of the parts of the standard is, as you can expect from an ISO-standard, deliberate. The most important aspect is that people need to know what they are talking about and need to be able to speak in the same language. The next thing is that you define *up front* what to expect from a good process. Then you need to define what you want to know. In the thought experiment by Amy J. Ko in Chapter 3, some nice examples show what can go wrong if you do not define this in the right manner. When you have done this preparation, your organization is ready to collect data and is able to make a sensible split into different domains, where apples are compared with apples and oranges with oranges.

Normalizing

Benchmarking is comparing, but more than just comparing any numbers. To really compare apples with apples, the data to be compared really needs to be comparable. In sizing, the size numbers of different software objects can be compared, either on a functional level (using standardized functional size measures, for example) or on a technical level. Different hard data about the processes to build or maintain a piece of software can be compared for measure and tracking purposes. Even soft data about the software or the process can be used for assessing the differences or resemblances between different pieces of software. This can be sufficient for estimating and planning purposes, but is insufficient for true benchmarking. Benchmarking is useful only when every aspect is the same, except for the aspect you want to benchmark. In practice, this is hardly ever the case. To have a meaningful benchmark, all aspects not under scrutiny must be made the same. This is called *normalizing*. Based on mathematical transformations or experience data, peer data can be normalized to reflect the conditions of the project that is benchmarked. Things like team size, defect density, and project duration can be made comparable. When a large data set of peer data is available, the easiest way is to select only the peer data that is intrinsically comparable and can be used without mathematical transformations. When not enough peer data is available, aspects can be normalized of which the effect is known.

For instance, the effect of team size is extensively studied. When teams of different sizes are compared, the aspects that are impacted by the team size (such as productivity, defect density, and project duration) can be normalized to reflect the size of the team that you want to benchmark.

Sources of Benchmark Data

There are multiple ways to benchmark productivity against the industry. There are several international commercial organizations worldwide that provide benchmarking services and that have collected a large amount of data through the years, examples of which are METRI, Premios, and QPMG. There are also commercial estimation models available that allow the users to benchmark their project estimates against industry knowledge bases (Galorath SEER or PRICE TruePlanning) or trendlines (QSM SLIM). Because of the confidentiality of the data, these commercial parties usually won't disclose the actual data that they use for their benchmarking services. Only the process and the results of the benchmark are usually communicated, not the actual data points used. External sources of benchmark data are particularly useful when not enough internal data is available to benchmark internal projects on an *apples to apples* basis. These external sources can be tailored to reflect the situation in the organization as well as possible.

ISBSG Repository

The only open source of productivity data is the ISBSG repository, which covers more than 100 metrics on software projects. The ISBSG is an international independent and not-for-profit organization based in Melbourne, Australia. Not-for-profit members of ISBSG are software metrics organizations from all over the world. The ISBSG grows and exploits two repositories of software data: new development projects and enhancements (currently more than 9,000 projects) and maintenance and support (more than 1,100 applications). Data is submitted by consultants and practitioners in the industry. The reward for submitting data to ISBSG is a free benchmark report comparing the realized productivity, quality, and speed against a few high-level industry peer groups.

All ISBSG data is

- Validated and rated in accordance with its quality guidelines
- Current and representative of the industry
- Independent and trusted
- Captured from a range of organization sizes and industries

As the ISBSG data can be obtained in an Excel file, it is possible to analyze and to benchmark project productivity yourself. Simply select a relevant peer group and analyze the data set using the most appropriate descriptive statistics, such as shown in the example in the section "Benchmarking in Practice."

Internal Benchmark Data Repository

If the main reason for benchmarking is for internal comparison, with the objective to improve, then the best source is always to have an internal benchmark repository. In such a repository, the cultural differences that have an impact on productivity (see Chapter 3) are not present and normalizing can be done in a reliable way. When the process to build an internal repository for benchmark data is in place, ideally this process should be used to submit this data to ISBSG as well. In this way, the organization receives a free benchmark on how they stand with regard to industry peers, and the ISBSG database is strengthened with another data point.

Benchmarking in Practice

To put all the theory in practical perspective, we end this chapter with a simplified example on how a benchmark is performed in practice. This example shows how improvements can be found by comparing with others.

An insurance company has measured the productivity of ten completed Java projects. The average PDR of these ten projects was ten hours per function point. To select a relevant peer group in the ISBSG D&E repository, the following criteria could be used:

- Data quality A or B (the best two categories in data integrity and data completeness)
- Size measurement method: Nesma or IFPUG 4+ (comparable)

- Industry sector = insurance
- Primary programming language = Java

After filtering the Excel file based on these criteria, the results can be shown in a descriptive statistics table such as Table 18-1.

Table 18-1 *Example Descriptive Statistics Table*

Statistic	PDR
Number	174
Min	3,1
10% Percentile	5,3
25% Percentile	8,2
Median	11,5
75% Percentile	15,2
90% Percentile	19,7
Max	24,8

As productivity data is not normally distributed but skewed to the right (PDR cannot be lower than 0 but has no upper limit), it is customary to use the median value for the industry average instead of the average. In this case, the average productivity of the insurance company lies between the 25th percentile and the market average (median). This may seem good, but the target may be in the best 10 percent performance in the industry. In that case, there is still a lot of room for improvement. A similar analysis can be made for other relevant metrics, such as quality (defects per FP), speed of delivery (FP per month) and cost (cost per FP). From these analyses it becomes clear on which aspect improvement is required. Comparison of the underlying data with best-in-class peers or projects reveals the differences between the benchmarked project and the best in class. These differences are input for improvement efforts.

False Incentives

Benchmarking, like any type of measurement, has a certain risk. People have a natural tendency to behave toward a better outcome of the measurement. Ill-defined measures will lead to unwanted behavior, or as Amy J. Ko puts it:

> *In pursuit of productivity, however, there can be a wide range of unintended consequences from trying to measure it. Moving faster can result in defects. Measuring productivity can warp incentives. Keeping the pace of competitors can just lead to an arms race to the bottom of software quality.*

Benchmarking needs to be done on *objects* that can be normalized to be truly comparable. In software development this means a sprint, a release, a project, or a portfolio. You should not be benchmarking individuals. Why? The simple answer is that there is no way to normalize people. More arguments against measuring productivity of individual software developers can be found in Chapter 2. Although there is sufficient evidence that there is a 10:1 difference in productivity between programmers, they are also exceedingly rare. An interesting example of what happens when you try to compare individuals is in the blog "You are not a 10x software engineer." There are unmistakably software developers who are much better than others, but this difference cannot be benchmarked in a sensible way. When you compare individuals using their output per unit of time, then the junior team members who are building a lot of simple functions might appear to be better than the brightest team member who solve the three most difficult assignments while helping the juniors and reviewing the code of the other team members. This is illustrated with facts in Chapter 1.

Summary

Benchmarking is the process of comparing your organization's processes against industry leaders or industry best practices (outward focus) or comparing your own teams (inward focus). By understanding the way the best performers do things, it becomes possible to improve. One of the key challenges in the software industry is to measure productivity of completed sprints, releases, projects, or portfolios in an *apples to apples* way so that this information can be used for processes such as estimation, project control, and benchmarking. At this moment, only the standards for functional size measurement comply with demands for standardized measurement procedures

and intermeasurer repeatability to produce measurement results that can be compared across domains to benchmark productivity. Benchmarking is useful only when every aspect is the same, except for the aspect you want to benchmark. In practice, this is hardly ever the case. To have a meaningful benchmark, all aspects not under scrutiny must be made the same. This is called *normalizing*. Based on mathematical transformations or experience data, peer data can be normalized to reflect the conditions of the project that is benchmarked. There are multiple ways to benchmark productivity. The best source is always to have an internal benchmark repository. In such a repository, normalizing can be done in a reliable way. External sources of benchmark data are particularly useful when not enough internal data is available to benchmark internal projects on an apples-to-apples basis. These external sources can be tailored to reflect the situation in the organization as well as possible. Benchmarking, like any type of measurement, has a certain risk. People have a natural tendency to behave toward a better outcome of the measurement. Benchmarking needs to be done on objects that can be normalized to be truly comparable. In software development, this means a sprint, a release, a project, or a portfolio. You should not be benchmarking individuals.

Key Ideas

The following are the key ideas from this chapter:

- Benchmarking is necessary to compare productivity across teams and organizations.
- Productivity can be compared across products, but you have to compare the right thing.
- Comparison across organization makes sense only if you do it in a standardized way.

Further Reading

- Wikipedia, on: Cyclomatic complexity, http://en.wikipedia.org/wiki/Cyclomatic_complexity, Lines of Code (LoC), http://en.wikipedia.org/wiki/Source_lines_of_code, Productivity, http://en.wikipedia.org/wiki/Productivity, Use Case Points, http://en.wikipedia.org/wiki/Use_Case_Points.
- Nesma, on IBRA points, http://nesma.org/themes/productivity/challenges-productivity-measurement.
- Scrum alliance, on Story points, http://scrumalliance.org/community/articles/2017/January/story-point-estimations-in-sprints.
- ISO, on: Information Technology project Performance Benchmarking (ISO/IEC 29155), http://iso.org/standard/74062.html, Functional Size Measurement (ISO/IEC 14143), http://iso.org/standard/38931.html.
- ISBSG, on the source of benchmark data, http://isbsg.org/project-data.
- Amy J. Ko, on the downside of benchmarking, Chapter 3 in Caitlin Sadowski, Thomas Zimmermann: Rethinking Productivity in Software Engineering, Apress Open, 2019.
- Ciera Jaspan and Caitlin Sadowski, on the arguments against a single metric for measuring productivity of software developers, Chapter 2 in Caitlin Sadowski, Thomas Zimmermann: Rethinking Productivity in Software Engineering, Apress Open, 2019.
- Steve McConnell, on the underlying research of the 10x Software Engineer, http://construx.com/10x_Software_Development/Origins_of_10X_-_How_Valid_is_the_Underlying_Research_/.

- Sean Cassidy, on the fact that you are most likely NOT a 10x Software Engineer, http://seancassidy.me/you-are-not-a-10x-developer.html.
- Yevgeniy Brikman, on the rarity of 10x Software Engineers, http://ybrikman.com/writing/2013/09/29/the-10x-developer-is-not-myth/.
- Lutz Prechelt, on why looking for the mythical 10x programmer is about asking the wrong question, Chapter 1 in Caitlin Sadowski, Thomas Zimmermann: Rethinking Productivity in Software Engineering, Apress Open, 2019.