ISBSG
Delivering IT Confidence

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

Short paper

Story Point metrics or Functional Size metrics?
An analysis of the main differences

Published by the International Software Benchmarking Standards Group

2023

Introduction

The International Software Benchmarking Standards Group (ISBSG) plays a pivotal role in advancing the field of software cost estimation, offering a treasure trove of data that holds immense value for organizations and professionals in the software development industry.

ISBSG data is a rich source of historical project information, encompassing a wide array of software development endeavors across diverse industries and domains. This data allows for the extraction of invaluable insights, enabling more accurate and informed software cost estimation.

In this era of increasing technological complexity and growing demand for precise project planning, the ISBSG data serves as an indispensable resource. It facilitates better decision-making, risk management, and cost optimization in the realm of software development.

The ISBSG collects industry data, where output is measured using ISO/IEC standardized and therefore objective, repeatable, auditable methods, such as Nesma, IFPUG and COSMIC function points. Typical key metrics based on function points are:

- Project Delivery Rate (PDR)[1]: Hours spent per function point
- Cost efficiency: Cost (or Price) per function point
- Quality: Defects per function point (in test and/or 1st month of production)
- Delivery Speed: Function points delivered per calendar month.

[1] The PDR is the inverse of the universal concept of Productivity (output/input) as it is easier to process for human minds, which usually struggles with metrics with many decimals.

The ISBSG 'New Developments & Enhancements' repository contains thousands of completed projects for which these metrics are calculated. This enables organizations to use this industry data for fact-based understanding and decision making.

In this short report we compare functional size metrics to the story point metrics often used in agile teams.

Story Points

Discussions do arise about the methods of measuring the output delivered by teams. Many organizations use an agile way of working nowadays, where teams use effort estimates based on story points.

Story point estimation is usually a form of relative estimation, where concrete backlog items are compared to each other, and the team estimates which one will take more time to develop. They often use a Fibonacci scale of points, which is 0, 1, 2, 3, 5, 8, 13, and then usually 20, 50 and 100.

For example, if user story XYZ is assigned 20 story points, this means: the *effort* the team thinks is needed to develop this user story is greater than a user story that is assigned 13 story points. However, XYZ requires less effort than a user story that is assigned 100 story points.

Story points are a relative method used to estimate effort for backlog items that include bug-fixing, problem analysis and other tasks. So, story points are often very useful in estimating which backlog items to develop in the next sprint. However, metrics based on story points can't be used on a higher level of aggregation for management information.

Story Points can easily be manipulated since the unit of measurement is subjective and relative. Therefore, story point metrics can never be used in a contract between a customer and a supplier. For instance, a fixed price per story point will only be beneficial for the supplier. The customer will be unable to check if they are getting the optimal value for money if they don't have objective metrics in place as well.

Functional Size

Functional Size Measurement (FSM) is a method used in software engineering to quantify the size of a software application or system based on its functional requirements.

Instead of measuring size in lines of code or other technical metrics, FSM focuses on measuring the functionality or features offered by the software. The most used

methodology for FSM are the Function Point Analysis (FPA) standards, Nesma and

IFPUG, both partners of ISBSG.

How Functional Size is Measured:

Function Point Analysis measures software size by quantifying the functionality provided to end-users based on five primary functions:

- **External Inputs**: These are user interactions that result in data being entered into the system.
- **External Outputs**: These represent the results produced by the system for the users.
- **External Inquiries**: User interactions that involve both input and output.
- **Internal Logical Files**: These are logical groups of data maintained by the system.
- **External Interface Files**: These are files shared between the system and external applications.

Each of these functions is evaluated based on complexity and then weighted to calculate a Function Point (FP) count. Function Points serve as a standardized measure of the size of the functional user requirements of a software application.

Advantages of Functional Size Measurement:

- **Focus on User Requirements**: FSM concentrates on the functional aspects of software that directly cater to user needs, providing a more user-centric view of software size.
- **Better Estimation**: Function Points can assist in estimating effort, cost, and duration for software development or maintenance projects more accurately than simple lines-of-code metrics.
- **Performance measurement and Benchmarking**: Function Points enable comparisons between projects, allowing organizations to benchmark productivity, quality, and performance across different applications or teams.
- **Vendor-Independence**: It allows for standardized measurement irrespective of the technology, programming language, or platform used to develop the software.

By providing a standardized, objective measure of software size based on functional requirements, FSM techniques like Function Point Analysis play a crucial role in project management, resource allocation, and estimating software development efforts.

The agile challenge – measure value delivered

For traditional, waterfall, software-development projects, functional and non-

functional requirements were specified upfront. No (or very few) changes were allowed during the development phase. This led to an issue where, after particularly long delivery and testing phases, the demand for the delivered software may have waned.

Advantages of agile:

The main idea behind an agile way of software development is to deliver software quickly with a short feedback loop. Agile software development has several advantages that make it a popular and effective approach in the software industry:

- **Flexibility and Adaptability**: Agile methodologies, like Scrum or Kanban, prioritize adaptability. They allow for changes in requirements, design, and priorities even late in the development process, enabling teams to respond quickly to customer or market needs.
- **Customer-Centric**: Agile focuses on delivering value to customers through iterative development cycles. Regular feedback from customers or stakeholders ensures that the delivered product aligns closely with their needs and expectations.
- **Faster Delivery of Incremental Value**: Agile emphasizes delivering a working product in small, incremental iterations known as sprints. This means that functional parts of the software are delivered more frequently, providing tangible value early in the development process.
- **Improved Quality**: Continuous testing, frequent reviews, and iterations contribute to higher quality products. Issues are identified and resolved early, reducing the likelihood of major defects in the final product.
- **Enhanced Team Collaboration**: Agile promotes collaboration among cross-functional teams. Daily stand-up meetings, regular communication, and shared ownership of tasks foster a culture of teamwork and collective responsibility.
- **Transparency and Visibility**: Agile practices promote transparency by making project progress, challenges, and impediments visible to all team members. This transparency helps in identifying and addressing issues promptly.
- **Risk Mitigation**: Breaking down the project into smaller iterations allows for better risk management. Problems or obstacles are detected early and addressed within a shorter timeframe, reducing overall project risk.
- **Emphasis on Continuous Improvement**: Agile methodologies encourage a culture of continuous improvement. At the end of each iteration, teams reflect on what worked well and what didn't, adjusting for the next iteration.
- **Adaptive Planning**: Agile doesn't rely on rigid long-term planning. Plans are adjusted based on feedback, changing market conditions, or new insights, allowing for more effective and realistic planning.
- **Higher Customer Satisfaction**: By involving customers throughout the

development process and delivering features that meet their needs, Agile often leads to higher customer satisfaction and a better product-market fit.

These advantages contribute to Agile's popularity and success in delivering software products that are: more aligned with customer needs, of higher quality, and completed in a more efficient and adaptive manner.

Investigating the agile challenge:

There is an important challenge, however, and that is to demonstrate the value produced by team(s). In most organizations, although the budget requirement per team is quite predictable (effort hours per team member * cost per team member), the value produced is not. Teams that use story point metrics may sometimes seem very productive and predictable, but in reality, they may not produce the expected value. Figure 1 explains this challenge.

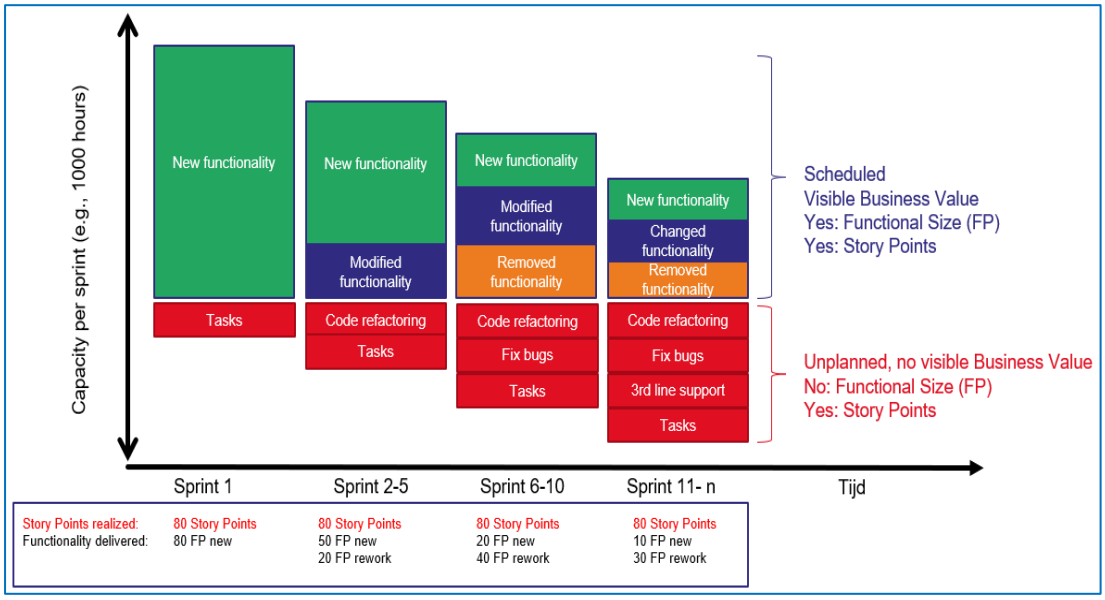

Figure 1: Value creation measured – the agile challenge[2]

In Figure 1, a hypothetical new team is starting to develop a Minimum Viable Product (MVP) and the ongoing development uses Scrum, an agile framework for project management.

On the vertical (i.e.Y-axis) in Figure 1, a fixed capacity per sprint is made visible as a constant capacity of 1000 effort hours per sprint. These effort hours are spent on various activities that need to be carried out. In the first development sprint, a lot of effort can be spent on new functionality (the green block). There are always some required tasks that do not directly deliver functional size, such as setting up the test

[2] Source: IDC Metri: www.idc.com/idcmetri

environment. These tasks (red blocks) are necessary but don't produce visible value for the customer or user. Therefore, the team should spent as little effort as possible on tasks that don't produce value. This enables them to focus on delivering new functionality. New functionality is seen as business value for the customer or user.

As time progresses, the product owner may wish to change already-developed functionality (blue blocks) or even remove already developed functionality (orange blocks). Although not ideal, this may be necessary to make sure the product meets the requested user requirements.

Over time new red blocks may appear. An example of this is the fixing of defects that could not be repaired in the sprint where they were introduced. Another example is the refactoring of code, needed because the team had to make short-cuts to meet completion. After going live with the MVP, user support may also be required.

Figure 1 also shows the challenge of managing the value delivered by the teams. While the tasks shown in the red blocks must be performed when they occur, it would have been better if they hadn't occured. The team could have then spent the effort and energy on delivering value to the product.

As explained before, many teams use the subjective, relative, effort estimation approach of Story Points. They assign Story Points to all the items that require effort, regardless the type of effort. So the red blocks get Story Points as well, as these require effort. In the Figure 2 we examine this more closely.

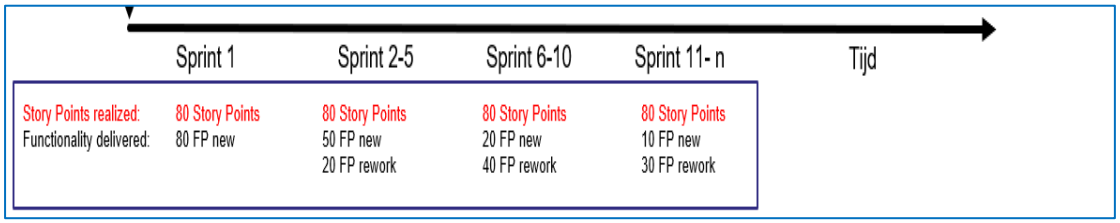

Figure 2: Story Points vs. Functional Size

When looking at the story points developed, senior management may be very happy with the team performance. The estimations were accurate as the team produced 80 story points per sprint. However, when looking at the value delivered, expressed in functionality added, modified and removed, the team performance decreased over time.

When the MVP has, for example, a functional size of 500 Nesma or IFPUG function points, story point metrics don't indicate: when the MVP will be ready, whether it is on time and aligned with the business functions such as marketing. Typical functional size metrics are useful to tackle this particular challenge.

The Nesma standard for functional size measurement (ISO/IEC 24570:2018[3] defines the concept of Project Size over a certain period of time (e.g., sprint, month, release, quarter, etc.) as follows:

Project Size = Functional Size added + Functional Size modified + Functional Size removed

Project Size is considered to be the value produced by the team, for example per sprint.

The typical metrics to understand and manage the value creation function are:

- Productivity: effort hours spent / Project Size
- Cost Efficiency: cost / Project Size
- Delivery Speed: Project Size / calendar month
- Sprint Quality: Defects / Project Size

By measuring these metrics, along with story point metrics, the team, product owner and senior management are able to understand the actual team performance and also benchmark team performance, perhaps using ISBSG data. This enables senior management to determine which teams are performing well and which teams can improve performance.

Characteristics of good metrics and KPIs

Good metrics share several characteristics that make them effective in measuring performance or progress in a meaningful way:

1. **Relevance**: Metrics should directly align with your objectives or goals. They must measure something that has a tangible impact on the desired outcome.
2. **Clarity**: They should be easy to understand and interpret. Ambiguous or complex metrics can lead to confusion and misinterpretation.
3. **Measurable**: Metrics need to be quantifiable or at least have a clear methodology for measurement. This allows for consistent tracking and comparison over time.
4. **Actionable**: Good metrics should provide insights that prompt action. They should indicate areas for improvement or highlight successful strategies.
5. **Timeliness**: They should provide information in a timely manner. Real-time or regularly updated metrics are often more valuable as they allow for quick adjustments and decision-making.
6. **Consistency**: Metrics should be consistent and comparable across different time frames, teams, or departments. Consistency ensures reliability in

[3] https://www.iso.org/standard/72505.html /

evaluating progress.

7. **Cost-Effective**: Gathering and analyzing metrics shouldn't be overly resource intensive. The cost of obtaining and utilizing the metrics should be justified by the value they provide.
8. **Strategic Alignment**: Metrics should tie back to the broader strategic goals of the organization. They help in measuring progress towards these overarching objectives.
9. **Contextual**: Understanding the context surrounding the metrics is crucial. Comparing metrics without considering the context can lead to misinterpretation.
10. **Feedback Loop**: Good metrics facilitate a feedback loop where the results obtained lead to actions, which in turn impact future metrics.

The choice of metrics can significantly influence decision-making and behavior within an organization. It's crucial to regularly reassess metrics to ensure they remain relevant and continue to drive progress towards desired outcomes.

In Table 1, characteristics are rated for Story point and Functional Size metrics, using. This is done at an organizational level (including stakeholders such as the product owner, senior IT managers and C-level management) rather than at the team-level.

The ranking system used in Table 1 is: **+++** – extremely relevant, **++** – mostly relevant, **+** – somewhat relevant, **-** – little relevance, **---** – not relevant.

Characteristics	Story Point metrics	Functional Size metrics
Relevance	---	+++
Clarity	+	+
Measurable	+++	+++
Actionable	-	++
Timeliness	+++	+++
Consistency	---	+++
Cost-Effective	+++	-
Strategic Alignment	---	+++
Contextual	---	+++
Feedback loop	+++	+++

Table 1: Good Metric characteristics applied to Story Point and Functional Size metrics from an organization perspective.

Table 1 shows that functional size metrics give much greater insights from an organization perspective. The ISBSG repository Developments & Enhancements (2023 release) contains over 1500 data points of agile projects, releases or sprints measured in one of the ISO standards for functional measurement. If you wish to obtain the ISBSG D&E repository: https://www.isbsg.org/subscriptions/

Why then is functional size measurement used in so few organizations? Function Points suffer from a bad reputation, formed during the traditional software development era. During this time, many projects failed and function points were often blamed, although this was unfair. Another argument opposing the use of function points is that it was expensive since certified experts were required to measure functional size.

However, nowadays, there is technology that can automatically measure functional size from user stories. Also, as there is usually limited functionality developed in a sprint (typically not more than 100 function points per sprint), it takes very short time to manually measure. For example, using the global method of functional size measurement[4] , it should be possible to measure the Project Size of a Sprint in 2 to 4 hours. Scrum Masters or Product Owners should be able to learn the method quite easily. Organizations should invest in building this capability, so they get better management information to manage the value creation function.

Conclusion

In this short paper, the main differences between the often-used story point metrics and functional-size metrics are discussed. The conclusion is the Story Point metrics are useful for planning on team-level. However, they can't be used on an aggregated level, for instance to compare teams to each other or to industry data, such as ISBSG data.

Functional Size metrics, such as Productivity, Cost Efficiency, Delivery Speed and Sprint Quality, are objective and repeatable. They do not rely on non-functional and/or project requirements.

Organizations should invest in building the functional size measurement capability into their teams and/or use technology that measure functional size automatically. This will help the management and improve the value delivery function of their organization.

If you wish to do your own analysis, or if you are interested to use the ISBSG data for cost estimation, benchmarking, performance measurement, procurement, etc., please subscribe to the data here: https://www.isbsg.org/project-data/

[4] See for instance: https://nesma.org/2015/06/the-accuracy-of-high-level-fpa/

The International Software Benchmarking Standards Group (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 to help all organizations (commercial and government, suppliers and customers) in the software industry to understand and to improve their performance and decision making. ISBSG sets the standards of software data collection, software data analysis and software project benchmarking processes and is the international thought leader in these practices.

The ISBSG mission is to support commercial and public organizations to improve the estimation, planning, control and management of 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, we issue 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/

A specific Agile/Scrum data collections questionnaire can be downloaded here: https://cutt.ly/4vnuXVT

Partners

This page will help you to find an ISBSG partner in your country: https://www.isbsg.org/board/