# Measuring Productivity of a Software Team

*by Carlos Eduardo Vazquez, CFPS, Fatto Software Consulting*

As a software organization aims to reach maximum team productivity levels, there are key questions that must be addressed. How to ensure that a software team is productive? How to improve team performance to achieve higher levels of performance? Those are some of the questions addressed in this article.

## What is a productive software development team?

People ask me: "What is a productive software development team?" The answer is too simple: It is a team which produces outcomes! This answer seems as simplistic as the classic "42" answer from Deep Thought in the *The Hitchhiker's Guide to the Galaxy* book; so, it's worth exploring the question a little more—perhaps analyzing the possible outcomes - before coming to a conclusion.

The term software may refer to a variety of different product types such as computer programs, configuration scripts, user interface specifications, requirements documents, architectural design plans, test cases and other software representations. Therefore, when performing productivity tests, it is necessary to establish the appropriate perspective (or perspectives in some cases) in order to assess how productive a team is when compared to a referenced scale.

The referenced scale is necessary, in that often the interest is not whether a team is productive or not, but:

1. How productive it is?
2. How productive does it need to be?
3. How does its productivity performance compare against other teams in the market?

If management chooses a business perspective instead of a technical one (that requires far greater technical details, issues resolved, and detailed understanding), then an appropriate choice of measure to use is the functionality delivered or impacted by the project. For software maintenance activities, a compatible product representation can be the functionality added, deleted and changed.

## Maintenance activities and productivity

There are different activity types related to software maintenance, such as bug fixes, functional enhancements, technology upgrades, etc. It is possible to measure productivity in all of them. However, it is first necessary to determine what the desired deliverables – or outcomes - are for a given perspective for each type of maintenance.

If the outcome is related to service levels of availability:

What are the outcomes in maintenance types such as bug fixes (not covered by the warranty), level III help desk support, and other application support activities? It is the service availability instead of a series of functionalities delivered. There are times when no support activity is required and the team spends no direct effort to achieve the desired outcome; but when they do, it must adhere to certain service levels, such as the time to start addressing the issue and the time limit to solve it. What matters most to manage productivity for this maintenance type is the resource usage levels and not the outcome already defined in service level agreements (SLA). It may be measured in terms of degrees of adherence of the actual performance against the targets defined within the SLA during a tracking window, such as a month, a quarter or a fiscal year.

If the outcome addresses non-functional requirements and some infrastructure functionality:

Imagine a scenario where there is no change whatsoever to the functional dimension of an application software, so that there is no change in:

- Business rules;
- Information presented to the user by the application;
- Data input by the user into the application;
- Data retrieved by the application;
- Data stored by the application.

However, there is the introduction of a new framework providing a series of shared services to the application, which requires a high-impact intervention spanning the application as a whole. In a scenario like this, the delivered outcomes of such maintenance might be the:

- New framework delivered;
- Existing functionality working properly in the new context defined by that framework.

Therefore, two measures for two different outcomes may be used. One measurement would be to size the development of the new framework and another to size the scope of the application impacted by the introduction of the new framework. Both measures can assist to plan and measure productivity using the appropriate reference scale.

Productivity as a process attribute

The concept of measuring a teams' productivity is, in fact, a simplification. Actually, management wants to assess the performance of a (software) production process. It is paramount you do not mix up data from a project like the previous example, which introduces the new framework, with data from developing a new system from scratch. Each software production process has its own unique probability distribution of its productivity and information. For example, average productivity rates from

one process are meaningless to plan and monitor the performance of another.

Let me explain a bit further, what I meant when I stated:

"…another to size the scope of the application impacted by the introduction of the new framework."

Even though it is definitely not an enhancement project, there is nothing stopping me from using the functionality impacted as a reference for the product outcome for this engagement. I just cannot compare the measurement of this outcome with other enhancement measurements.

However, if I use a factor in order to make the two processes in review (a functional development and a technological improvement) compatible, then there should be no problem in comparing the two. A factor (let us call it an Average Functional Technical Equivalent Factor - AFTEF) plays this role and the comparison of data from both processes produces this factor value.

The concept is not new. NESMA's Function Point Analysis for Software Enhancement does something similar by determining a percentage of change as an impact factor, and Q/P Management Group made available its Impact Points Counting Guidelines.

If the Outcome is Software Performance Improvement:

*Some software processing takes up to 48 hours to complete: the software team should restructure its architecture and the programs implementing it, so that after its completion, it will take less than 24 hours to process.*

It is not an enhancement and the team must update the software configuration only regarding configuration items from architecture and implementation disciplines. After all, this engagement has not altered the application baseline functionality at the requirements level. However, it is still possible to track which functionality in this business perspective the engagement impacts.

The best practice is to compare different processes, such as the one about the new framework and the other about the performance enhancement, in order to determine objective criteria to classify those maintenance types and to establish values corresponding to the AFTEF.

For instance, analysis of 12 engagements for performance enhancements, such as the one described in this text, indicates that while the average productivity for a change in an enhancement project functionality is 05 Staff-Hours / FP, the performance improvement engagements have a poorer average delivery rate of 10 Staff-Hours / FP.

The AFTEF from Performance Improvement (as described in the example) is set as a factor of 2.00. All other priorities aside and, depending on the relationship between the productivity of a change involving enhancement project functionality and new development functionality, it is more productive to throw away the software and develop a new one in this scenario.

## Advantages of using FPA in productivity measurement

It is important to state that FPA, which produces units for functional measurement, is a method with a higher level of maturity, professional support and organizational experience than any other in the marketplace. There is also the IFPUG – International Function Points Users Group – responsible for its maintenance and evolution since 1986.

The use of functional measurement takes as an input the functional requirements related to the tasks and services as defined within business functional organization structure and business process models. It does not consider late design or implementation decisions. Therefore, it is a measurement directly related to user knowledge, skills, vocabulary and understanding.

If some other internal or technical measurement method were to be used, it would not be possible for the client to audit the results presented by the development team. That by itself is a big enough reason to discourage its use for productivity assessments purposes.

To use a functional metric balances the trends in action when assessing productivity. There is a push from the team towards inflation of resource usage, while the client has the power to impose a different push to increase the production. In some circumstances, we witness some very passionate debates over this dynamic. If the team increases its resource usage regardless of a proportional increase in production, then it creates a drive to decrease its productivity. If the user increases the scope, then he understands there will be a proportional counterpart in resource usage. A dynamic like this is only possible when there is some reliable functional metric supporting the measurement of production, like IFPUG function points.

## A non-IT professional can use FPA?

There are those who think that if a person is not from the IT area, then that person cannot use FPA. That belief is wrong. I have trained thousands of people in FPA for over 20 years and I can safely say that about half of the time spent was dedicated to supporting IT professionals to unlearn (and sometimes to accept the dual role of) a software technical perspective. This definitely makes FPA more difficult to use. I have had, during those years, the opportunity to watch business people use FPA and take control of processes where they, at one point in time in the past, played the role of observers.