Harold van Heeringen[1], Theo Prins[2], Edwin van Gorp[3]

Productivity Measurement of Agile Teams – Overcoming the Issues with Non-Functional Requirements

Abstract

Nowadays, as the software industry is slowly becoming more mature, software measurement and performance measurement are becoming increasingly important. Organizations need to know their productivity and competitiveness in software development projects for various reasons. In many software development contracts, targets are set for the suppliers to reach. These targets are based on software metrics like productivity, speed of delivery and software quality. In order to check if the targets are reached, it is necessary to measure the functional size of the software product that is delivered and also the functional size of the software development project that is carried out, as there is usually a difference between these two sizes. To be able to use the functional size in contracts, it must be measured in an objective, repeatable, verifiable and therefore defensible way. That being the case, the industry's best practice is to use an ISO/IEC standard for functional size measurement, e.g. Nesma, COSMIC or IFPUG function points. However, these methods only measure the functional user requirements from the total software requirements to be delivered. In activities like project estimation and productivity measurement, the influence of the non-functional requirements is expressed in the Project Delivery Rate (PDR) which is expressed in effort hours per function point. If more than the average number of non-functional requirements need to be realized in a project (or more severe non-functional requirements), the PDR used should also be higher. In the industry it is customary to set productivity targets based on an average (or calibrated) influence of non-functional requirements and this works quite fine in traditional software projects. In software development projects that are executed in an agile way, this is not always the case. When working agile, there are forces that influence the traditional way of performance measurement significantly, resulting in a number of serious issues. In this paper these issues are explained and a method to overcome these issues is proposed.

[1] METRI B.V. Schiphol-Rijk, the Netherlands, harold.van.heeringen@metrigroup.com

[2] Sizing, Estimating & Control, Sogeti Nederland B.V., Vianen, Nederland, theo.prins@sogeti.com

[3] Sizing, Estimating & Control, Sogeti Nederland B.V., Vianen, Nederland, edwin.van.gorp@sogeti.com

Keywords: agile development; story points, outsourcing, performance measurement, functional requirements, Nesma, ISBSG, non-functional requirements, Agile normalized size

1. Introduction

These days, more and more software development projects are carried out in agile oriented approaches like Scrum or DSDM. Projects following an agile software development method deliver software in an iterative way. The iterations are usually called sprints. The main idea behind agile software development is that short feedback loops enable the customer (users) to experience the product sooner, which allows the requirements and solutions to evolve during the execution of the project. As Scrum is the most dominant agile software development method nowadays, this paper is primarily focused on this method and the concepts and terminology of Scrum are used in the text.

1. The development of a specific set of requirements in a specific period of time. Although executed in an iterative way, the project has a 'goal', and at one point it is decided that the project is finished. The product owner controls the project, making sure that the necessary functionality is delivered within time and budget.

   In traditional approaches, this type of development would be called a 'new development project' or a 'release';
2. The continued development (evolving) of an application. In this situation there is no definite duration or a specific set of user requirements. A year is divided into X sprints of Y weeks and the team is continuously working on the product backlog items the product owner deems the most important until it is decided that the product does not need further maintenance (which may be so in the far future).

   In traditional approaches, this type of development would be called 'maintenance'.

Productivity measurement of the contract type listed as number 1, is not significantly different in agile projects than in traditional projects. In both approaches, the difficulty lies in measuring all functional changes properly in order to measure the actual project size.

The focus of this paper is on the contract type described under number 2. A team is working for an indefinite period of time, which is divided into sprints

of 2, 3 or 4 weeks' length each, on a product backlog. The product backlog is prioritized in such a way that the product backlog items (PBI's) with the highest importance are planned to be delivered first. The PBI's are also rated in story points, which are assigned in team effort.

2. Productivity Measurement in Traditional Software Projects

Productivity measurement in traditional software development projects is quite straightforward[4]. The customer and the supplier have to agree on a few items beforehand, the most important being:

- Which (variant of which) size measurement method to use;
- Which performance metrics to use;
- Which activities are in scope and which activities are out of scope for the performance measurement;
- Which are the benchmarks or targets to use per metric;
- Who is measuring, who is reviewing, how to deal with differences of opinion.

To give an example, let us look at a performance measurement contract that a government agency and a supplier have used to measure the performance in a specific Java development project with an indicative size of around 400 Nesma[5] function points (FP). The items above were addressed in the following way:

- Size measurement: Estimated Nesma[6];
- Performance metrics:
  - Productivity: PDR (h/FP);
  - Functional Quality: Defects/FP Acceptance Test + 1st month production;
  - Delivery Speed: FP per calendar month.
- Activities in scope: technical design, coding, programmer test, system test, support acceptance test, project management, quality assurance;

---

4 M. Bundshuh, C. Dekkers, *The IT Measurement Compendium, Estimating and Benchmarking Success with Functional Size Measurement*, Spinger, ISBN: 978–3-540-68187–8, p. 313.

5 ISO/IEC 24570:2005, *Software Engineering – NESMA Function Size Measurement Method Version 2.1, Definitions and Counting Guidelines for the Application of Function Point Analysis*, ISBN: 978–90-76258-19–5.

6 Early Function Point Counting, http://nesma.org/themes/sizing/function-point-analysis/early-function-point-counting/

- Activities out of scope: functional design, delivery management, acceptance test, implementation;
- Benchmarks targets: A relevant subset from the ISBSG repository New Developments & Enhancements[7] (e.g. Java projects, Government, size between 300 and 500 FP)
  - PDR: ISBSG subset median PDR;
  - Defects/FP: ISBSG subset percentile 25%;
  - Speed of delivery: ISBSG subset median PDR.
- The supplier measures the functional size, the customer reviews the size measurement. The metrics can be audited by the customer periodically. Independent third parties may be used to solve differences of opinion.

Because of the specific set of requirements and the fact that the goal of the project is clear (to meet that specific set of requirements), the performance measurement is pretty straightforward. As it is a Java development project for a government agency, it is easy to select a proper subset from the ISBSG repository to set the targets and benchmarks for this project. If there are specific non-functional requirements that are expected to have a significant influence on the performance, the benchmarks and targets may need to be re-evaluated and agreed upon.

After the project has finished, the project size is determined, the performance metrics are calculated and it becomes clear if the performance targets are met or not.

3. Why Are Agile Projects Different?

Most people in the software industry know the four main principles of the agile manifesto[8]:

- Individuals and interactions over processes and tools;
- Working software over comprehensive documentation;
- Customer collaboration over contract negotiation;
- Responding to change over following a plan.

---

[7] *International Software Benchmarking Standards Group (ISBSG) New Developments & Enhancements R13*, released February 22015, www.isbsg.org

[8] K. Beck, *Manifesto for Agile Software Development*, Agile Alliance 2001, retrieved 14/06/2010.

Although the items on the right of the word 'over' are still important, the items on the left are considered more important.

In practice, an agile project can be described in the following general way:

A very important role is played by the product owner, who is a representative of the customer/user organization. This product owner needs to be knowledgeable about the functionality of the product and the business value it offers to the users. Also, the product owner needs to have a mandate from the customer organization to make decisions and he/she needs to be committed to fulfill this role. This means that he/she is able to devote the time needed to carry out the role, this means that he/she is able to meet with the team when necessary, make decisions when necessary and prioritize the backlog items by business value.

There is a product backlog available which lists the product backlog items identified by the product owner. These backlog items are either functional or non-functional by nature. The effort needed to realize the product backlog items are estimated by the team members using story points. Although many people in the agile community think the story point measure is a size measure, it is in fact an effort measure. It is a relative measure which gives an estimate of how many hours the team thinks they need to spend on a particular product backlog item, compared to a specific baseline backlog item to which a random number of story points has been assigned. There is no standard approach to assigning story points. Most teams identify the smallest product backlog item in the product backlog and assign 1, 2 or 3 story points to it. All other backlog items are then compared to this baseline backlog item and the number of story points assigned to them reflects the effort the team needs to realize them, compared to the baseline item. This method is absolutely subjective and probably not very repeatable or verifiable.

Although story points are quite useful for sprint planning, as to decide which product backlog items can be realized in a specific sprint, metrics based on story points are completely useless for performance measurement. As there is no standard for assigning story points (it is not a measurement activity), it is quite easy to manipulate the number of story points realized in a sprint in order to comply with the targets set. And since there is no reference point, it is easy to manipulate the targets as well.

But still, organizations wish, and sometimes need, to measure the performance of their suppliers, also in agile executed projects. In many cases usual performance metrics are agreed upon, but instead of using a project or a release as a basis for measurement, the performance measurement instrument is applied to sprints. This means that for every sprint, the functional size delivered should

be measured, the effort hours in scope need to be retrieved, the defects found and resolved must be identified and the performance metrics should be calculated. Some issues, when not properly addressed, already arise here:

- Agile teams do not usually register defects. Especially not when they are easy to resolve;
- Agile teams are multidisciplinary by nature. Sometimes it is hard for the team members to register their effort on specific tasks. This is important, however, as some tasks may be out of scope for performance measurement;
- Agile teams sometimes do not change the functional documentation in the same sprint where the functionality is realized in. This makes it difficult to measure the functional size of that sprint;
- According to the theory, at the end of each sprint, working software is delivered that could be implemented in a production environment. In practice, this is not always the case and product backlog items are declared "not ready" at the end of the sprint. As only functionality that is ready is measured, this means a low size delivered in that sprint and possibly a relatively high size delivered in the next sprint.

The product owner decides which product backlog items are the most important. Here, the main issue comes to light. The product owner may decide that mainly non-functional backlog items need to be realized in a specific sprint. As functional size measurement methods only measure the functional user requirements, a size measurement of such a sprint would result in a few (or even zero) function points. In these cases, the usual performance metrics result in bad values and the targets are not met for that sprint.

So, there is a number of issues when trying to implement the traditional performance measurement instrument on the sprint level in agile contracts. Summarized, these issues are as follows:

- The functional size of the delivered product can be hard to measure, either because of underspecification, or because the documentation provided does not reflect the status of the product after the sprint;
- The effort hours are not booked consciously on the right tasks in the effort registration system, making it hard to define the number of hours spent on the activities in scope of the performance measurement;
- During the sprint, system tests and acceptance tests are carried out, but defects found and defects resolved are not logged appropriately, making it hard to use the quality metrics;

- The functional size delivered in the sprint may be very low (or zero), seriously affecting the performance metrics. Reasons for this could be (not limitative) the following:
  - The product backlog items turned out to be too big to deliver in the sprint and were not ready;
  - Illness of team members or other unplanned events that may have disturbed productivity;
  - Non-functional backlog items to realize in the sprint (e.g. improving performance, security, code quality, et cetera).

These issues result in very uneven performance metrics over time. An example of how this could look like is given below for the PDR. Let us say the following data is measured for 8 sprints (table 1).

Table 1.

Sprint	1	2	3	4	5	6	7	8
Effort	345	389	367	412	365	375	390	401
Size (FP)	15	5	16	3	25	0	36	32

Source: the authors' own study.

This would result in the following PDR per sprint (table 2).

Table 2.

Sprint	1	2	3	4	5	6	7	8
PDR (h/FP)	23.0	77.8	22.9	137.3	14.6	n/a	10.8	12.5

Source: the authors' own study.

In sprint 6 zero function points were realized, so it is impossible to determine the PDR in h/FP. Let us assume the target PDR for the specific project is set to be 25.0 hours per function point, the performance graph would look like that in figure 1.

The productivity realized shows a very uneven view. In the next paragraph a method is proposed that should give better insight into the supplier's capabilities and performance.

An option to overcome the issues mentioned would be to only use the effort hours spent on functional backlog items for the performance measurement. In practice, it is very hard to identify these effort hours. Furthermore, this approach

makes it easy to manipulate the performance measurement simply by booking more hours on non-functional backlog items (assuming that there is no control over the effort hours booked). Instead of normalizing the effort hours to match the functional size, the method proposed in this paper is to normalize the functional size to match the total effort.

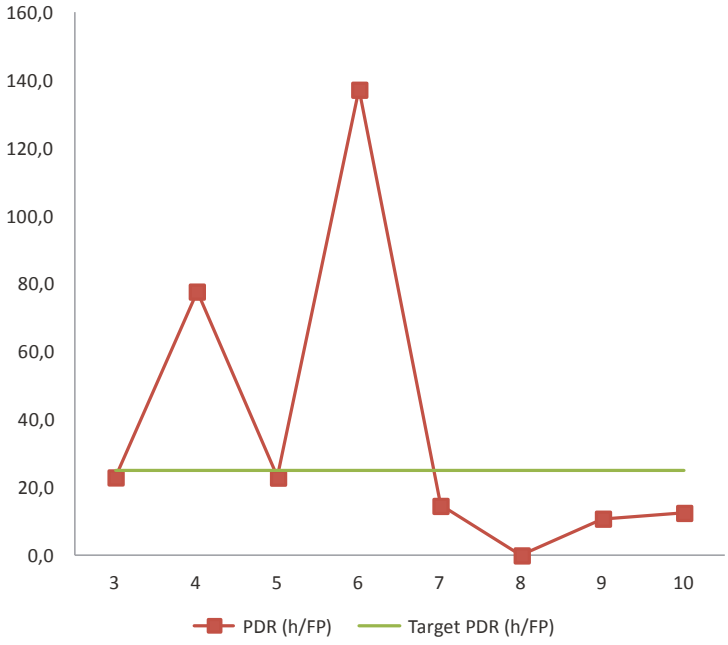

Figure 1. PDR Actual vs. Target

Source: the authors' own study.

4. The Method Proposed

The performance measurement method that is proposed in this paper is to normalize the functional size in order to determine the functional size that the team could have realized if they had not been instructed to realize any non-functional backlog items.

This method enables organizations to use the standard methods and techniques for performance measurement and benchmarking, while almost no extra data or effort is necessary. The method is not totally objective, but possibilities of manipulation (consciously or unconsciously) are very limited.

In order to carry out this normalization activity, one extra activity needs to be performed. The Scrum team assigns story points to the backlog items, so the number of story points that is realized in a specific sprint is known. Of each of the sprint backlog items, the team should decide if the nature of the item is (mainly) functional or non-functional. The ratio between the functional and the non-functional story points is used to normalize the functional size.

Examples of functional backlog items could be: add a field to the database table 'User' named 'twitter username', add a field to the 'manage user' screen in which the administrator can enter the 'twitter username' and add a twitter icon to the screen if the user has a 'twitter username', linking to the user's twitter page.

Examples of non-functional backlog items could be: improve the performance of the batch job in order to have it run in under 2 hours, improve the code quality of the application to be compliant with SQALE rating A[9] or upgrade the development environment to the next version, because the supplier stops supporting the current version soon.

The proposed method contains the following steps:

1. Measure the functional size of the functional sprint backlog items that were ready at the end of a sprint using one of the mentioned ISO/IEC standards for functional sizing;
2. Determine for all sprint backlog items that were ready at the end of the sprint if these were functional or non-functional by nature. In case of a hybrid nature, try to assess the functional/non-functional ratio;
3. Determine the number of story points that were assigned to the functional backlog items that were ready at the end of the sprint;
4. Determine the total number of story points that were assigned to all backlog items that were ready at the end of the sprint;
5. Calculate the Agile Normalized Size (ANS) using this formula:

$$ANS = (\text{functional size} / \text{functional story points}) * \text{total story points}$$

Example: In a sprint, five backlog items have been realized (see table 3). Three of them are functional by nature, the other two are non-functional. The functional size of the functional backlog items is measured using Nesma function point analysis.

[9] J.-L. Letouzey, *The SQALE Method for Evaluating Technical Debt*, 3rd International Workshop on Managing Technical Debt, Zurich, 9–12.06.2012.

Table 3.

Sprint X	Nature	Nesma FP	Story points
BLI 1	Functional	4	4
BLI 2	Non-functional	0	6
BLI 3	Non-functional	0	2
BLI 4	Functional	5	3
BLI 5	Functional	4	3
Total		13	18

Source: the authors' own study.

The ANS in this example is: $(13 / 10) * 18 = 23.4$ FP. The ANS represents the equivalent of the functional size that could have been realized if only functional backlog items had been included in the sprint.

In table 4, it is displayed that the ANS gives a more accurate view of the size in sprints in which a lot of non-functional backlog items were realized. This is especially obvious in sprint 4:

Table 4.

Sprint	Func. size	Functional story points	Non-functional story points	Total story points	ANS
1	20	32	12	44	27.5
2	25	28	16	44	39.3
3	18	24	20	44	33
4	29	35	4	39	32.3
5	4	6	36	42	28
6	15	16	24	40	37.5

Source: the authors' own study.

Advantages of this Method

The main advantage of this method is that the influence of non-functional backlog items is reduced, while still an ISO/IEC standard method is used in order to determine functional size. This productivity can then be compared with the data that is measured using the same (type of) ISO/IEC standard method.

Disadvantages of this Method

The method is dependent on the story points assigned to the functional and non-functional backlog items. Assigning story points, as indicated before, is a highly subjective activity. However, as it is a necessary activity carried out before the actual start of a sprint and because of the fact that the product owner is usually present to oversee this activity, chances of the team manipulating the assigning of the story points are considered to be small. Also, for the determination of the nature of backlog items, there is no standard approach, but usually it should be easy to understand if backlog items are functional or non-functional by nature. One other disadvantage of the method is that it is impossible to measure the ANS if only non-functional backlog items are realized in a specific sprint. In section V entitled 'The Progressive Approach', this issue is solved.

Effort Hours in Scope of Performance Measurement

It is always important to carefully consider the effort hours that should be taken in scope of performance measurement. Usually, the benchmark used determines the effort hours that are in scope, as you wish to compare apples with apples. If, for instance, the benchmark used is based on a project lifecycle excluding functional design and overhead hours, the effort hours spent on these activities should be placed out of scope for the performance measurement.

Due to the fact that in the proposed method size is normalized, all activities that are carried out to realize any backlog item, functional or non-functional, have to be in scope for performance measurement. This would mean activities like (not limitative):

- Design (functional and/or technical);
- Coding and programmer test;
- System test;
- Scrum master;
- Project support (for the Scrum master);
- Creating/adjusting documentation;
- Resolving defects;
- Research (proof of concepts, et cetera);
- Meetings / sessions.

Effort hours spent on the activities that are not directly related to realizing backlog items are out of scope for performance measurement, e.g.:

- Delivery management;
- Quality management;
- Project support (for the delivery management).

After the effort hours in scope have been determined, the productivity per normalized FP can be calculated, see the example in table 5:

Table 5.

Sprint	Func. size (FP)	Agile normalized size (nFP)	Effort spent	Hours/FP	Hours/nFP
1	20	27.5	500	25	18.2
2	25	39.3	480	19.2	12.2
3	18	33	530	29.4	16.1
4	29	32.3	468	16.1	14.5
5	4	28	534	133.5	19.1
6	15	37.5	522	34.8	13.9

Source: the authors' own study.

In figure 2 it is clear that the hours/nFP metric provides a more stable view than the hours/FP metric. The fact that development of a new functionality is not the most important goal is normalized into a situation where it is so.

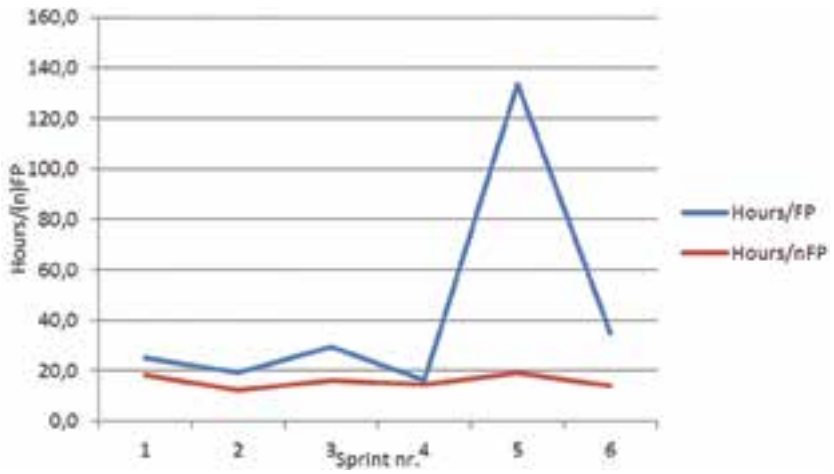

Figure 2. Hours/FP vs. Hours per nFP

Source: the authors' own study.

5. The Progressive Approach

One of the aforementioned disadvantages of the proposed method is that it is not possible to determine the normalized size for sprints in which no functional backlog items have been realized. This can be obviated by using the progressive approach.

The progressive approach divides the sum of the functional sizes of all completed sprints by the sum of the functional story points of these sprints. The result hereof is then multiplied with the sum of all story points of all completed sprints. This approach makes it possible to take sprints into account in which no functional backlog items have been realized. In table 6, the example used earlier has been elaborated upon to explain the progressive approach:

Table 6.

Sprint	Functional size (FP)	Functional story points	Non-functional story Points	Total story points	Agile normalized size (nFP)	Hours/nFP	Hours spent (cumulative)	ANS (progressive)	Hours (cum) /nFP (progressive)
1	20	32	12	44	27.5	18.2	500	27.5	18.2
2	25	28	16	44	39.3	12.2	980	66	14.8
3	18	24	20	44	33	16.1	1,510	99	15.3
4	29	35	4	39	32.3	14.5	1,978	132.2	15
5	4	6	36	42	28	19.1	2,512	163.6	15.4
6	15	16	24	40	37.5	13.9	3,034	199.2	15.2
7	0	0	41	41	n/a	n/a	3,546	231.4	15.3
8	18	24	20	44	33	15.4	4,054	264.3	15.3

Source: the authors' own study.

The productivity expressed in hours/nFP and the productivity in hours (cum)/ nFP (progressive) can be benchmarked against the traditional internal and external hours/FP benchmarks, like for instance the 'New Developments and Enhancements' Repository of the International Software Benchmarking Standards Group[10].

[10] *International Software Benchmarking Standards Group (ISBSG) New Developments & Enhancements R13*, released February 2015, www.isbsg.org

In figure 3, the figures from the example are shown. The benchmark PDR in this example is set at 20 hours/FP.

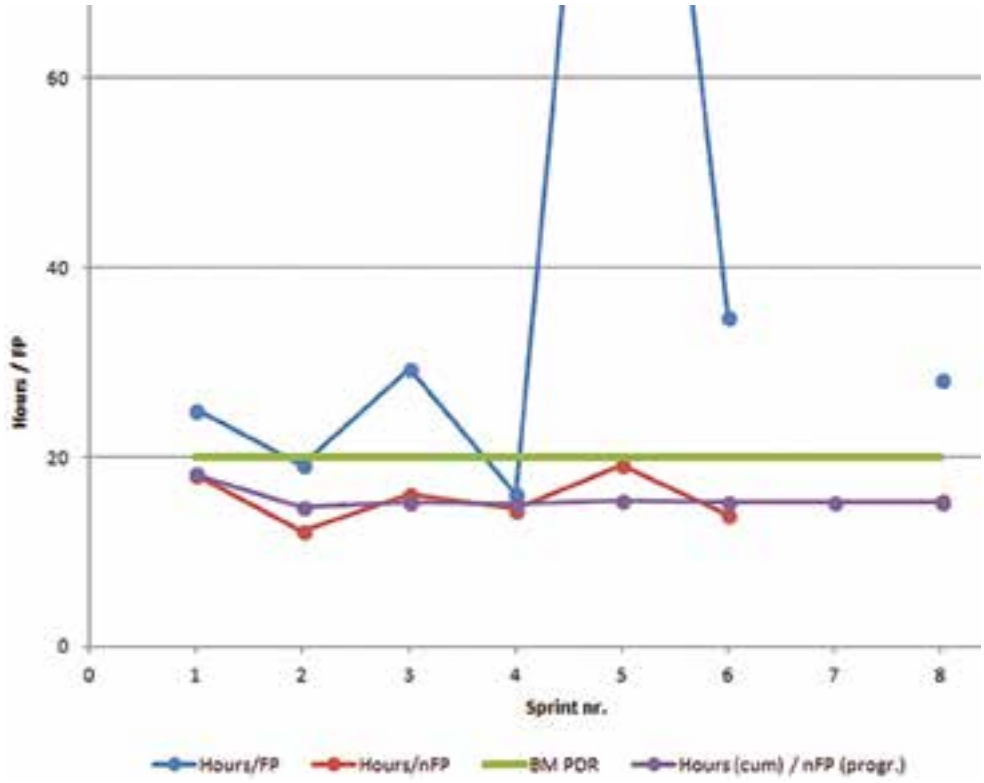

Figure 3. The Progressive Apprach Results

Source: the authors' own study.

The main benefits of the proposed method are clearly visible in sprints 5 (in which mainly non-functional backlog items were realized) and 7 (in which only non-functional backlog items were realized).

Starting Points

In order to use the proposed method in the proper way, a number of starting points have to be taken into account.

Effort Administration

- The effort hours need to be booked in the effort administration in such a way that it is possible to clearly identify the effort hours in scope and out of scope of the performance measurement;
- All effort hours spent on a project must be booked, also overtime;
- Hours spent on other projects should not be booked on the project that is being measured;
- The team members need to understand the importance of an accurate and correct time administration.

Documentation

- After each sprint the functional documentation should be made up-to-date and it must be clear:
  - which functionality was added in the sprint;
  - which functionality was changed in the sprint and in which way;
  - which functionality was deleted in the sprint.

Size Measurement

- The effort hours need to be booked in the effort administration in such a way that it is possible to clearly identify the effort hours in scope and out of scope of the performance measurement.

Product Backlog and Sprint Backlog

- The product backlog consists of functional and non-functional backlog items. For each item it is known which type it concerns;
- All items have been assigned story points to in a realistic way;
- The product owner decides which product backlog items will be placed on the sprint backlog of the sprints;
- After each sprint it is decided which backlog items are ready (meet the definition of done[11]) and which items are not (or not completely);

[11] The definition of done is determined before the start of an agile project. Only if all characteristics of that definition have been met, the item is considered 'ready'.

Data Collection

- After each sprint, a data collection form (DCF) must be filled in by the Scrum master. In this DCF at least the following information must be filled in:
  - the effort hours per activity;
  - the backlog items declared ready and their associated type;
  - the story points assigned to these backlog items.

Performance Measurement

- The productivity is determined using the method described above;
- The functional size, the data and the results of the productivity measurement is stored in a central location;
- Standard reports are created in which the productivity measurement is shown and compared to predefined internal and external benchmarks. Also trends in productivity within the projects are reported.

6. Conclusions

More and more organizations adopt agile software development methodologies to deliver new functionalities in a faster way to their customer. In the meantime, in the slowly maturing software industry, the need for productivity measurement is becoming more and more important, especially in outsourcing contracts. As long as the most important objective of agile projects remains delivering a new or changed functionality, the traditional productivity measurement instruments can still be used in an effective way. In projects or contracts that are mainly about evolving a certain application, without a specific end date or a specific scope, productivity is sometimes hard to measure in ISO/IEC standards for functional size measurements. The influence of non-functional requirements can really impact productivity and the product owner (customer) is the only one who can control this. Some attempts have been made in the recent time to normalize the effort hours in order to capture only effort hours that were spent on functional user requirements. However, in practice it seems impossible to do so. Instead, this paper proposes normalizing functional size in such a way that functional size is calculated that could have been realized in a sprint if the product owner had only put functional backlog items on the sprint backlog. Whether

this method really is going to solve all the issues in productivity measurement of agile projects still has to be proven.

References

Beck K., *Manifesto for Agile Software Development*, Agile Alliance 2001, retrieved 14/06/2010.

Bundshuh M., Dekkers C., *The IT Measurement Compendium, Estimating and Benchmarking Success with Functional Size Measurement*, Spinger, ISBN: 978–3-540-68187–8, p. 313.

Early Function Point Counting, http://nesma.org/themes/sizing/function-point-analysis/early-function-point-counting/

International Software Benchmarking Standards Group (ISBSG) New Developments & Enhancements R13, released February 22015, www.isbsg.org

ISO/IEC 24570:2005, *Software Engineering – NESMA Function Size Measurement Method Version 2.1, Definitions and Counting Guidelines for the Application of Function Point Analysis*, ISBN: 978–90-76258-19–5.

Letouzey J.-L., *The SQALE Method for Evaluating Technical Debt*, 3rd International Workshop on Managing Technical Debt, Zurich, 9–12.06.2012.