Effort Estimation with Story Points and COSMIC Function Points - An Industry Case Study

Christophe Commeyne, Alain Abran, Rachida Djouab

Abstract

In Agile software projects developed using the Scrum process, Story Points are often used to estimate project effort. Story Points allow comparison of estimated effort with actual effort, but do not provide a size of the software developed and, therefore, do not allow determination of the productivity achieved on a project nor comparison of performance across projects and across organizations. In contrast to Story Points, international standards on functional size measurement, such as ISO 19761 (COSMIC Function Points), provide explicit measurement of software size for both estimation and benchmarking purposes. This study reports on the performance of estimation models built using Story Points and COSMIC Function Points and shows that, for the organization in which the data was collected, the use of COSMIC Function Points leads to estimation models with much smaller variances. The study also demonstrates that the use of COSMIC Function Points allows objective comparison of productivity across tasks within a Scrum environment.

Keywords: Story Points, Planning Poker, Function Points, COSMIC, ISO 19761, Effort Estimation, Scrum

1. Introduction

From their beginnings in the early 2000s, the Agile approach and Scrum process have been adopted in a number of organizations [1-3]. The Planning Poker technique is often used in Scrum projects as an effort estimation technique: it is based on practitioners' opinions, and its measurement unit, called a Story Point, allows comparison of estimated effort with actual effort deployed for a completed project.

However, Story Points do not explicitly provide a size of the software developed, and therefore cannot be used to objectively determine the productivity achieved on a project, nor to compare performance across projects and across organizations. Furthermore, since the Planning Poker technique does not provide direct information on the size of the software to be delivered nor of what was actually delivered, neither the teams nor their management have any way of knowing if their estimates are way off or if their productivity across Scrum iterations is erratic.

In addition to Story Points, there are a number of international standards that provide well-documented methods for measuring the functional size of a piece of software, independently of the technologies and development processes. These ISO standards provide a sound basis for productivity and performance comparisons across projects, technologies and development processes. Of the five functional size measurement (FSM) methods adopted by ISO, four correspond to the first generation of FSM methods, while the most recent one, COSMIC – ISO

19761 [4-5], is the only one recognized as a second-generation method that has addressed the structural weaknesses of the first-generation FSM methods.

The data analyzed in this paper come from an industry software organization where the available post-iteration information indicated that the Story Point estimates were frequently off-target and productivity across iterations could not be determined or monitored. This paper reports on a case study where Scrum iterations were also measured using COSMIC Function Points as an avenue to overcome the estimation and productivity benchmarking issues. The study reports on the performance of estimation models built using both Story Points and COSMIC Function Points and investigates whether or not the use of COSMIC Function Points leads to estimation models with smaller variances.

The paper is structured as follows: Section 2 presents an overview of the Planning Poker technique and Story Points, and of the COSMIC measurement method. Section 3 gives the background for the case study (a set of 7 Scrum iterations and 24 tasks) and analyzes the performance of Planning Poker estimates. Section 4 presents the data collection with the COSMIC measurement method of this same set of iterations and tasks, as well as the information obtained on the unit effort and productivity ratios for these tasks. Section 4 presents the derived COSMIC-based estimation model and compares it to the Story Points-based estimates. Section 5 presents the summary.

2. Overview of Planning Poker and COSMIC - ISO 19761

This section presents an overview of the Planning Poker technique and of the COSMIC – ISO 19761 Function Points measurement method.

2.1 Planning Poker technique – Overview

The Planning Poker technique was initially described by Grenning [6] and popularized by Cohn [7-8]: effort is estimated in terms of units of work referred to as 'story points' quantifying the relative complexity. Scenarios are compared and person-days estimated according to team ability, experience and knowledge of the field. The key concepts of the Planning Poker technique are based on:

- a list of features to deliver (the product backlog)
- a set of cards based on the Fibonacci sequence: 0, ½, 1, 2, 3, 5, 8, 13, 20, 40, 100, These values correspond to the expected number of days for developing a particular feature X (in a 'story point'). Cohn indicates that such estimates are intuitively transformed into intervals of hours based on the perceived 'velocity' of the team [7-8].

Planning Poker is used by a team during a planning meeting to give estimates for the development of a feature by evaluating the user scenarios of the product backlog. Each story is sized in terms of Story Points representing an estimate of the effort in days [7-8]. Designed as a consensus-based technique (i.e. a Delphi-like approach), Planning Poker calls for the participation of all developers on the team (programmers, testers, database engineers, analysts, user interaction designers, and so on.) The product owner participates but does not estimate. The technique includes the following steps, which may vary when implemented in various

organizations:

1. The product owner explains each story and its acceptance criteria in detail.
2. The team discusses the work involved and asks questions to the product owner.
3. Each team member secretly makes an estimate, in terms of Story Points.
4. Each team member selects a poker card that reflects their estimate of the effort required for that story.
5. Everyone shows their cards simultaneously.
6. The team members with the lowest and highest estimates explain the reasoning behind their estimates, and any misunderstanding of scope is resolved.
7. After discussion, the team members choose new cards to reflect their estimates based on the current discussion.
8. Agreement on the final number is taken as the team estimate.
9. The process is repeated for all the stories in the sprint backlog.

On the one hand, a number of benefits are directly derived from the consensus-building process of Delphi-like approaches:

1. Brings together practitioners from all disciplines involved in a software project to do the estimating.
2. Allows everyone to express themselves freely and promotes exchanges between the product manager and the development team.
3. Encourages open discussion rather than reliance on the opinion of the most influential or vocal member of the team. This allows the team to benefit from the experience of all team members.

On the other hand, some weaknesses are mostly related to the subjective nature of the estimating technique:

1. The estimation could vary when an identical backlog is provided to another Scrum team. This may cause conflict when multiple teams are working together on the same backlog.
2. Estimates will vary with skills and experience. Those more skilled and experienced would likely provide lower values than someone new to the field, which can skew the estimates.

It should also be noted that:

1. The estimate is made by direct consensus in terms of days (or hours), without explicitly or objectively sizing the product being estimated.
2. The team's 'velocity' is typically not based on historical data collected using international standards for measuring the size of software to be delivered.
3. Previous estimates are not reused in Scrum iterations; the team starts from scratch each time.
4. There is no process for analyzing the quality of the estimates and subsequently improving their accuracy.

Furthermore, there is a lack of well-documented empirical or experimental studies comparing Story Point estimates with actual project effort or comparing Story Points with other estimation techniques.

2.2 COSMIC Measurement Method - Overview

The COSMIC Function Points method is an international standard (ISO 19761 [4-5]) for measuring the functional user requirements of the software. The result obtained is a numerical value representing the functional size of the software itself, which can be used for benchmarking and estimation purposes. As required by ISO, COSMIC functional size is designed to be independent of any implementation decisions embedded in the operational artifacts of the software. This means that the functional user requirements can be extracted not only from software already developed but also from the software requirements before the software itself is implemented.

In COSMIC, a functional process is a set of data movements representing the functional user requirements for the software being measured. According to COSMIC, software functionality is embedded within the functional flows of data groups. Data flows can be characterized by four distinct types of movement of a data group:
- Entry (E) and Exit (X) data movements of a data group between the functional user of the software and a COSMIC functional process allow data exchange with a functional user across a software boundary.
- Read (R) and Write (W) data movements of a data group between a COSMIC functional process and persistent storage allow data exchange with the persistent storage hardware.

Each data movement, of any of the four types above, is assigned a single size expressed in COSMIC Function Points (CFPs), so that one data movement of one data group = 1 CFP.

The COSMIC measurement procedure consists of three phases:
1. The measurement strategy, which specifies the purpose and scope of the measurement.
2. The mapping phase, which aligns the software to be developed with the COSMIC generic model of software functionality and assigns a COSMIC size unit to each software functional requirement.
3. The measurement phase, which aggregates the individual measurements into a consolidated size measurement. The measurement result corresponds to the functional size and is expressed in COSMIC Function Points (CFPs).

With this ISO standard, it is possible to obtain a measure of the functional size of a software application that is objective and reproducible by distinct measurers.

Once a baseline of completed projects has been measured with COSMIC, the team productivity (also referred to as velocity, in Scrum) can be derived based on historical data and estimation models can be built using various statistical techniques such as linear or non-linear regressions [9]. The COSMIC Group has also developed a Guideline document on how to apply COSMIC Function Points in the context of Agile projects [10].

3. Case study context and Story Points data

3.1 Industry context

The data were collected from an organization specializing in the development of IP solutions for security and surveillance [11]. It provides a platform that can be tailored to meet

the needs of law enforcement agencies, schools, airports, casinos, retailers, and many others, and is used all over the world.

The development team follows the Scrum methodology with iterations ranging from three to six weeks. At the beginning of each iteration, all team members meet (four to eight hours, never more) and review with the team manager (who has the role of product owner) the list of change requests for the current iteration. These discussions include technical details about the changes to be made, candidate existing extension points (without having access to the source code) and expected impacts of these changes on the rest of the application. The objective of the meeting is to ensure that everyone has an understanding of the tasks in the current iteration.

In this organization, the team usually prepares its Story Point estimates of effort directly in terms of hours, without relying on any references or historical data.

Throughout an iteration, at the 15-minute daily morning meeting, each team member briefly describes the work of the previous day and what he or she expects to accomplish during the day. Each day, team members enter the number of hours spent on tasks. The number of hours allocated to a task are adjusted upwards if need be. The progress chart is presented daily by the Scrum Master, who, along with the product owner has the responsibility to shift features that could not be developed within the current iteration to another iteration. On the last day of the iteration, the product owner is shown the changes implemented during the iteration process. Depending on acceptance criteria and what is observed, the product owner determines whether the application can be sent to the testing team for validation and further verification.

When the task is completed, the Planning Poker estimates are compared with the actual effort and recorded on the team's intranet; however, within this organization, the estimates and actuals are not reused for subsequent iterations.

3.2 Data collection

For this case study, 24 tasks executed by the same team on the same application within a set of nine iterations were available for analysis, of which eight included functional requirements. Of the nine steps listed in section 2, only the following two were applied in the organization:

1. A first task was selected and its Story Points size was estimated in hours. Each team member offered a value selected from the Fibonnaci sequence: 0, ½, 1, 2, 3, 5, 8, 13, 20, 40, 100.
2. The team prepared its estimates of direct effort in terms of hours.

Table 1 presents, for each task of each iteration, the effort estimated with Story Points. The real effort, and the relative error of the Story Points estimates was calculated as follows [9]:

$$\text{Relative error (\%)} = \frac{(\text{Real effort} - \text{Estimated effort})}{\text{Real effort}}$$

For the dataset of 24 tasks, the total estimated effort corresponds to 1,329 hours, with an average of 55 hours per task (Table 1, column 1).

The total real effort by task for this dataset corresponded to 1051 hours with an average of

44 hours per task (Table 1, column 2). However, with actual effort time of individual projects varying from five to 173 hours the standard deviation was 72 hours, making it of little value for estimation purposes.

The differences between the estimated and real effort (Table 1, column 3) were considerable, ranging from -3 to -76 hours underestimation for seven tasks, and from 1 to 83 hours overestimation for 17 tasks, making for very large estimation inaccuracies at the task level. It was also noted that 18 out of 24 tasks required an effort below 50 hours.

Table 1 Story Points: Estimated and Actual Hours (N= 24 tasks)

No.	Iteration	Task	(1) Estimated hours	(2) Actual hours	(3) Under/Over estimation (hours) (3)=(2) – (1)	(4) Relative Error (4)=(2)–(1)/(2)
1	1	1.1	97	173	-76	43,9%
2	1	1.2	179	96	83	-85,8%
3	2	2.1	28	68	-40	58,8%
4	2	2.2	36	33	3	-9,1%
5	3	3.1	23	22	1	-3,0%
6	3	3.2	110	58	52	-91,3%
7	4	4.1	76	39	37	-94,2%
8	4	4.2	84	39	45	-116,1%
9	4	4.3	61	26	35	-137,3%
10	4	4.4	90	61	29	-47,6%
11	6	6.1	34	63	-29	45,6%
12	6	6.2	10	5	5	-102,0%
13	6	6.3	104	36	68	-189,7%
14	6	6.4	16	7	9	-141,8%
15	7	7.1	31	36	-5	12,7%
16	7	7.2	76	41	35	-84,7%
17	8	8.1	49	38	11	-28,7%
18	8	8.2	9	36	-27	76,0%
19	8	8.3	29	21	8	-37,7%
20	8	8.4	13	29	-16	55,2%
21	8	8.5	34	32	2	-7,2%
22	8	8.6	33	36	-3	8,7%
23	8	8.7	30	18	12	-66,7%
24	9	9.1	77	40	37	-92,9%
			Total= 1329 hrs	Total= 1051 hrs	Total error (+ & -) = 278 hrs	
			Average: 55 hrs	Average: 44 hrs	Absolute error: 669	

					hrs or 66%	

3.3 Analysis of the estimation performance of Story Points

The relative error (RE) for each task is presented in the rightmost column of Table 1 and graphically in Figure 1:

- Four projects were estimated within 10% of real hours,
- One project was estimated within 10% and 25% of real hours,
- 19 projects were estimated with an RE from 25% to 190% of real hours.

The mean magnitude of relative error (MMRE) was defined as follows [9]:

$$\text{Mean Magnitude Of Relative Error} = \frac{1}{n} \times \sum_{i=1}^{n} \left| \frac{(\text{Real effort} - \text{Estimated effort})}{\text{Real effort}} \right|$$

The mean magnitude of the relative error (MMRE) of the Planning Poker estimation process in this organization was 58%, which was large by any standard of estimation accuracy.

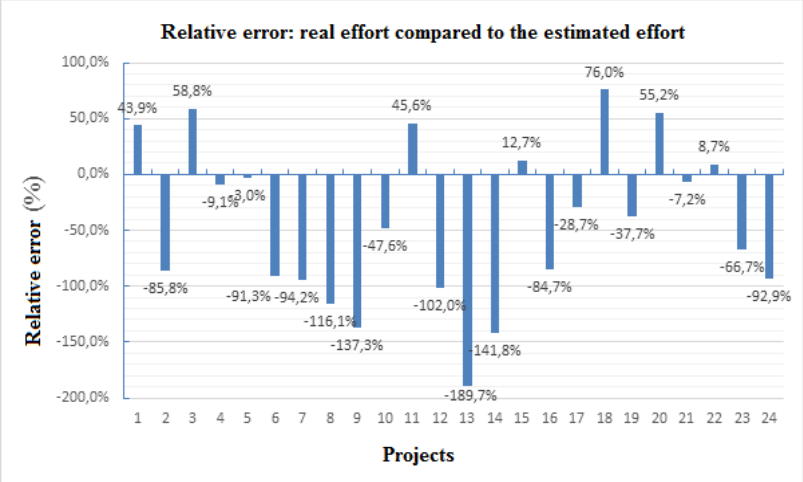

Figure 1: Story Points - Relative error of Estimated Effort (N = 24 tasks)

It can be observed from Figure 1 that of the 24 tasks, the team overestimated 17 (12 of these from 40% to 189%) and only underestimated seven. However, in absolute values, as shown in Table 1, some of the most underestimated tasks (Nos 1,3,11,18, and 20) were larger than a number of the overestimated tasks, indicating, here as well, a severe problem in underestimation.

3.4 Estimation model built with Story Points as the independent variable

The relationship between the estimates (here, the independent variable) derived from the Story Points and actual effort (the dependent variable) can be represented by linear regression

models as in Figure 2, where the Story Points estimated hours are on the x axis, and the actual hours on the y axis. In Figure 2, the straight line represents the following equation modelling this relationship:

Actual Effort = 0.47 x Story Points Estimated Effort + 17.6 hrs
with an $R^2 = 0.33$
and MMRE = 58%

With a coefficient of determination R^2 of only 0.33, the relationship is very weak (the maximum being 1.0). This means that only 33% of the variation in increase along the y axis (the dependent variable) can be explained by an increase on the x axis (the independent variable). In other words, in this organization, the Story Points effort estimates do not provide reasonable estimates of the effort required to implement a task: It leads to a large MMRE of 58%, with both severe under-estimation and over-estimation.

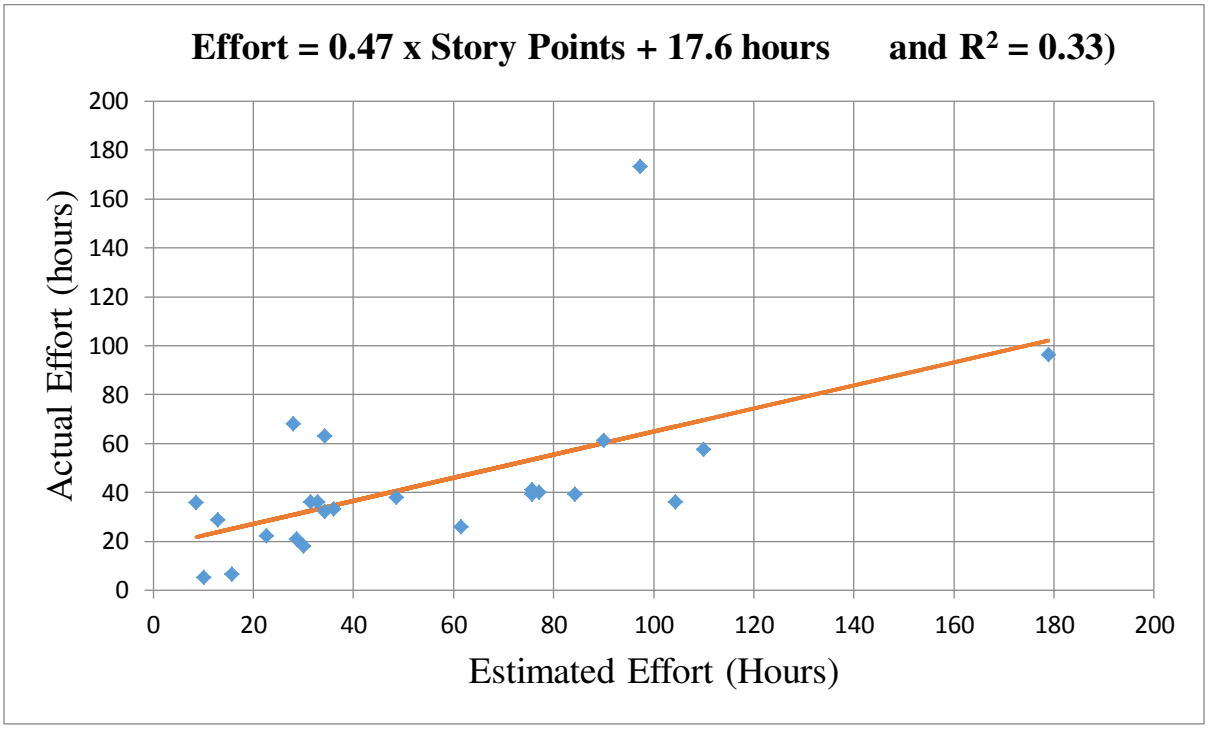

Figure 2: Story Points Estimated Effort versus Actual (N=24)

4. COSMIC data collection and analysis

4.1 COSMIC data collection

The functional size of the same set of 24 tasks was measured with the COSMIC method. The measurement was carried out either by examining the written documentation of the tasks, or by analyzing changes within the source code when the written documentation was insufficient. The measurement of the functional size of all tasks was performed by the same measurer. For each task, the components of the platform were identified, as well as the business users, the functional processes, and the objects of interest transiting across the software boundary. Table 2 presents the detailed size by data movement types (i.e., Entry, Exit,

Read, Write) and, in the rightmost column, the total functional size of these 24 tasks in COSMIC CFP measurement units. In summary:

- The functional size of the tasks varied from two CFP to 72 CFP
- The average functional size was 20 CFP

Table 2 COSMIC – ISO 19761 size of the 24 tasks

Task	Functional Process	Entry (CFP)	Exit (CFP)	Read (CFP)	Write (CFP)	Total Size (CFP)
1	15	27	9	28	8	72
2	9	19	10	7	1	37
3	13	13	10	5	3	31
4	4	4	4	3	0	11
5	4	4	2	3	1	10
6	8	8	7	3	3	21
7	6	6	5	7	1	19
8	6	12	4	11	3	30
9	4	4	4	2	2	12
10	7	9	5	7	3	24
11	12	12	12	35	0	59
12	1	1	0	0	1	2
13	4	4	3	5	1	13
14	1	1	1	1	0	3
15	4	4	1	5	3	13
16	3	3	2	9	1	15
17	5	5	2	4	3	14
18	3	3	3	9	0	15
19	2	2	2	1	0	5
20	7	7	3	4	4	18
21	5	5	4	6	1	16
22	6	9	3	3	3	18
23	3	3	3	3	0	9
24	6	7	5	13	1	26
Total	138	172	104	174	3	493

4.2 COSMIC-based estimation model

This section presents the construction of the estimation model based on COSMIC functional size with COSMIC CFP as the independent variable and actual effort as the dependent variable. For this dataset of 24 tasks, represented graphically in Figure 2, where the COSMIC size in CFP is presented on the x axis, and the actual effort on the y axis, the linear regression model was:

$$\text{Effort} = 1.84 \times \text{COSMIC Functional Size} + 6.11 \text{ hrs}$$

with an $R^2 = 0.782$

and an MMRE = 28%

With the coefficient of determination $R^2 = 0.782$ it can be seen that the COSMIC size now explains 78% of the variation in effort, in contrast to 33% when a regression-based Story Points model was used for estimation purposes. Furthermore, the mean magnitude of the relative error (MMRE) of the COSMIC-based estimation model was 28%, significantly better than an MMRE of 58% for the Story Points estimates.

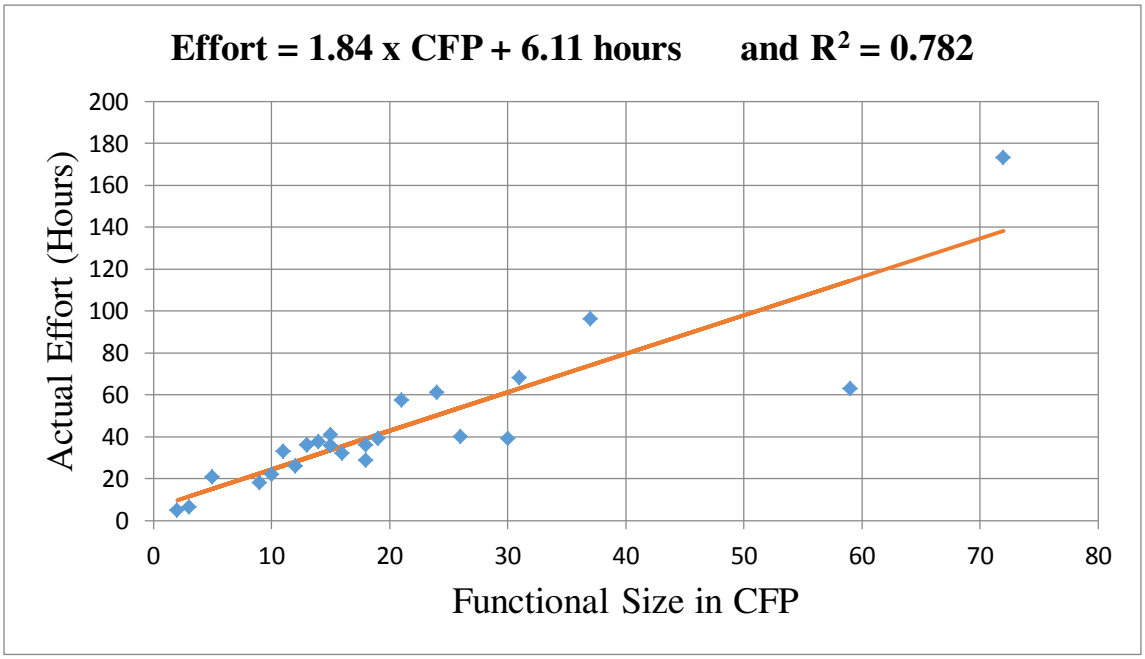

Figure 3: Estimation model with COSMIC CFP (N=24)

4.3 Project performance analysis with COSMIC

4.3.1 Unit effort comparison

In contrast to Story Points that does not size the output of a software task and cannot be used to calculate and compare project performance either in unit effort (i.e., in hours/CFP) or through a productivity ratio (i.e., in CFP/hour), the COSMIC-based measurement results allow one to calculate the performance of each task and to derive relevant management insights. For example, Figure 4 presents the performance in unit effort (i.e., hours/CFP on the y axis) for each of the 24 tasks (i.e., Tasks Id. on the x axis):

- the unit effort of 19 of the 24 tasks is within the 2 to 3 hrs/CFP range,
- four tasks (8, 11, 20 and 24) have a lower unit cost (i.e., higher productivity),
- only a single task (Id.19) has a much higher unit effort at over 4 hrs/CFP (i.e., poorer performance).

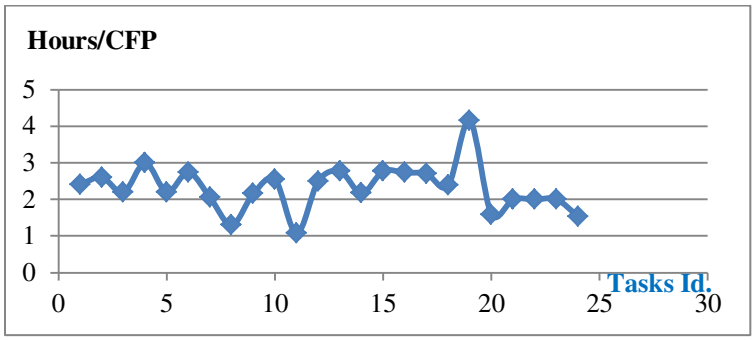

Figure 4: Unit effort in Hours/CFP of the 24 Tasks (ordered by Task Id.)

4.3.2 Analysis of productivity extremes and modified estimation model

To gain further insight into the estimation process Abran recommends [9, chapter 11] investigating the productivity extremes in order to determine what may cause such lower or higher effort. If the cause can be identified, then another independent variable can be added to the estimation model providing there are enough data points for statistical analysis. Alternatively the corresponding projects may be removed from the sample and used individually by analogy with another project with similar characteristics. If however, a reasonable cause cannot be identified, then such a data point should not be removed (i.e., being a productivity extreme alone is not a sufficient reason to remove a data point from a sample).

Within this data set the following productivity outliers can be identified from Figure 4:

- Task 19 has by far the highest unit effort at 4.15 hours/CFP, and
- Tasks 8 and 11 have by far the best performance (i.e., lowest unit effort at close to 1 hr/CFP).

A detailed analysis of the information available on task 19 could not identify any obvious cause for the poor performance of task 19, while it was observed that tasks 8 and 11 shared a common factor which positively impacted their productivity: Their requirements included significant functional reuse which led to very high code duplication which could explain such high productivity. In summary, it was deduced for this data set that the *functional reuse and related code duplication* was one reason contributing to the noise in the regression shown in Figure 2.

Therefore, if a future task includes functional reuse, as in tasks 8 and 11, then a unit effort of approximately 1 hr/CFP can be used in this organization to estimate such a specific task with this functional characteristic.

However, for estimating tasks without functional reuse, a regression model should be built excluding, in this case, these two tasks, i.e. with the remaining 22 tasks. Figure 5 presents the new linear regression model with the 22 remaining tasks. It was observed that the coefficient of determination R^2 improved from .782 to .977. The new linear regression model equation becomes:

Effort = 2.35 × Functional size – 0.08
with R^2= 0.977
MMRE = 16.5%

The new model explains more than 97% of the relationship between an increase in size and a corresponding increase in effort while the MMRE improved to 16.5%.

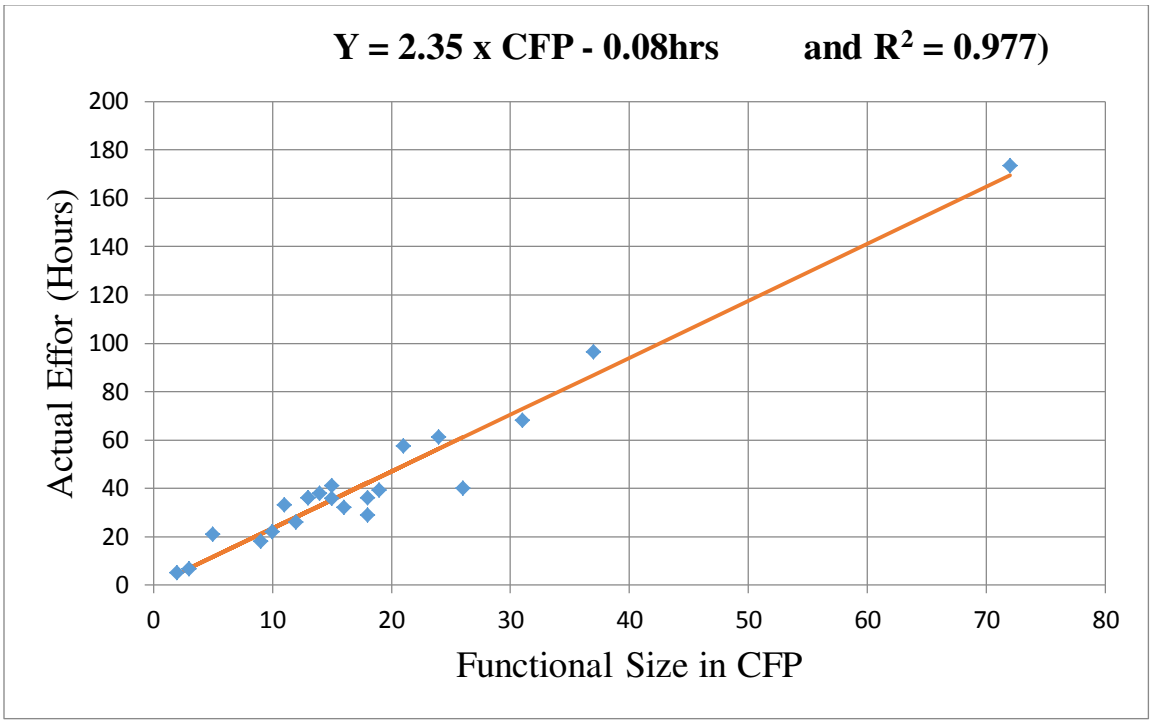

Figure 5: COSMIC-based Estimation Model (N = 22)

4.4 Using the COSMIC-based model for estimating an additional task

The COSMIC-based estimation model built from the set of 22 tasks was used with an additional project from this organization, which project was not included in the initial set of tasks. The new task corresponded to an improvement to the existing software application. The COSMIC measurement was made at the design phase of the iteration by the same measurer. The model was used as a method for a priori estimating, which means measuring a task before its implementation and providing an estimate of the development effort based on functional size measurement then using this size as the input to the COSMIC-based estimation model derived from the previous projects.

The size of the new task was 76 CFP close to task no. 1 with a size of 72 CFP. When 76 CFP was used as the independent variable in the estimation model of section 4.3.2, the model provided an estimated effort of 178 hours with an expected MMRE variation of ± 16.5%, i.e., an effort estimated to be between 149 and 207 hours. In practice, for estimation purpose for

this specific software project, the MMRE lower limit was selected for the estimate for this specific additional task.

Once the task was completed, the time recording system indicated an (actual) development effort of 131 hours: this represents a unit effort of 1.72 hours per CFP. Considering the actual effort of 131 hours, the MMRE lower limit selected for estimation represented an overestimation of only 14%.

4.5 Updating the COSMIC-based model with the additional task.

The organization can of course update its COSMIC-based estimation model every time data from a completed task becomes available. This is illustrated here with the addition of the project from the previous section as the 23rd data point. This updated estimation model is presented graphically in Figure 6, which now has 23 data points. The new linear regression model becomes:

Task Effort = 2.01 × COSMIC Functional Size + 4.93 hours
with an $R^2 = 0.977$
and an MMRE = 16.6%

The coefficient of determination R^2 has been impacted downwards slightly from 0.977 to 0.962. Its MMRE has increased slightly from 16.5% to 16.6%.

Estimation of the next task can then be based on the updated model and therefore, iteration after iteration, the model can be adjusted to take account of additional information as it becomes available.

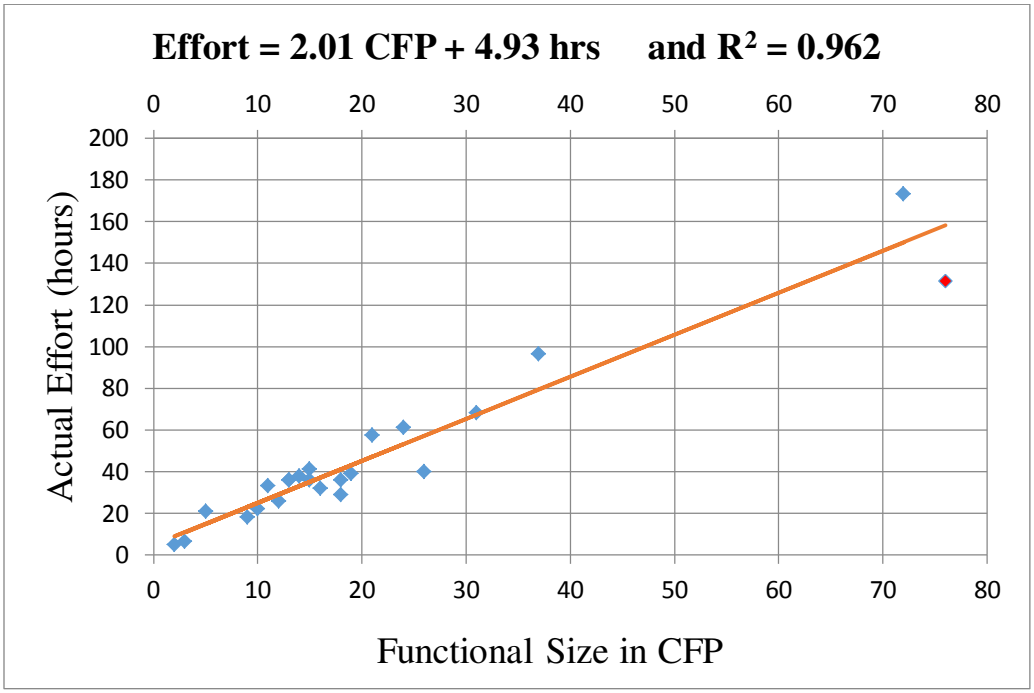

Figure 6: Estimation model updated with information from an additional task completed

(N = 23)

5. Summary and Future Work

This study has examined the performance of the Planning Poker / Story Points as an estimation technique within an organization where software is being delivered using the Scrum methodology.

When compared to actual effort for 24 tasks completed at the industry site in the case study, estimates with Story Points in this organization were shown to have led to large under- and over-estimates, with an MMRE of 58% - see Figure 2. It was also observed that since Story Points do not size the output of a software task, it could not be used to calculate and compare tasks performance.

When the functional size of the corresponding completed projects was measured using the COSMIC Function Points method, productivity could be calculated and compared using either unit effort (hours/CFP) or a productivity ratio (CFP/hour). This study has allowed us to emonstrate that in this organization the development team implemented the tasks at a sustained pace within a range of 2 to 3 hrs/CFP for 20 or the 24 tasks, iteration after iteration, throughout the period for which the data was collected – Figure 4.

An estimation model built with better estimation performance could then be compared to the Planning Poker estimates. The COSMIC based estimation model built with the initial 24 tasks in this organization had a much smaller estimation variance (i.e., an MMRE of 28%) – Figure 3.

The analysis of productivity extremes within this data set allowed identifying a functional reuse context within this organization that had led to major productivity gains for 2 tasks with such high functional reuse. For the purpose of estimating new tasks without high functional reuse, a new estimation model was built excluding these two tasks with high reuse: the reduced estimation model based on the subset of 22 tasks led to an improved estimation model with a much smaller MMRE of 16.5% - Figure 5. This paper illustrated as well how this industry site used this estimation model equation and expected MMRE variance to estimate an additional task, and compared its actual effort with the estimation range derived from the model. It also illustrated how to improve the estimation model by adding into it this additional completed task – Figure 6.

Finally, it can be noticed that measuring tasks with the COSMIC ISO standard and using their functional size in CFP unit to build an estimation model is an objective process without subjective judgments. Therefore, using a COSMIC-based estimation model means that, from one task to another, the effort for a software task of a given COSMIC size can be estimated within an expected smaller relative error range, regardless of which team members participating in the estimate, their experience and their knowledge of the software in question.

In summary, although the Planning Poker / Story Points are widely recognized and used in the Agile community, the COSMIC measurement method provides objective evidence of the team performance as well as better estimates.

References

1. Moniruzzaman, ABM; Akhter Hossain, S., "Comparative Study on Agile software development methodologies", Global Journal of Computer Science and Technology, Volume 13, Issue 7, 2013.
2. Schwaber, Ken, "Agile Project Management with Scrum", Redmond, WA: Microsoft Press, 2004.
3. Schwaber, K.; Beedle, M., "Agile Software Development with Scrum", Prentice Hall, Upper Saddle River, N.J., 2002.
4. ISO, "ISO 1971: 2011 - Software engineering - COSMIC: A Functional Size Measurement Method", International Organization for Standardization, Geneva, Switzerland, 2011.
5. COSMIC Group, "The COSMIC Functional Size Measurement Method – Version 4.0.1: Measurement Manual – The COSMIC Implementation Guide for ISO/IEC 19761: 2011", http://cosmic-sizing.org/publications/measurement-manual-401 [Accessed February 2, 2016].
6. Grenning, J., "Planning Poker", Renaissance Software Consulting, 2002.
7. Cohn, M., "Agile Estimating and Planning", Prentice Hall, 2005.
8. Cohn M., "User Stories Applied for Agile Software Development, Addison-Wesley, 2004.
9. Abran, Alain, "Software Project Estimation – The Fundamantal for Providing High Quality Information to Decision Makers", John Wiley & Sons, Hoberken, New Jersey, 2015, pp. 261.
10. COSMIC Group, "The COSMIC Functional Size Measurement Method - Version 3.0.1 - Guideline for Sizing Agile Projects", 2011. http://cosmic-sizing.org/publications/guideline-for-the-use-of-cosmic-fsm-to-manage-agile-projects/ accessed on February 21, 2016.
11. Commeyne, Christophe, "Établissement d'un modèle d'estimation *a posteriori* de projets de maintenance de logiciels", Master's thesis in Software Engineering, École de Technologie Supérieure (ÉTS) – University of Québec, Montreal, Canada, 2014.