# Guideline for Sizing Agile Projects with COSMIC

*Sylvie Trudel*[1] *and Luigi Buglione*[2,3]

[1] Pyxis Technologies Inc. – Montréal (Canada), [2]École de Technologie Supérieure (ETS) - Montréal (Canada); [3] Engineering.IT – Rome (Italy)

strudel@pyxis-tech.com, luigi.buglione@eng.it

***Abstract:***

*Agile became one of the most used 'buzzwords' in ICT projects in these recent years but the growing diffusion and interest of its related methods and techniques was not accompanied by the same maturity in sizing and estimation practices. In particular the application of a functional size measurement (FSM) method for sizing requirements is not typical to "agilists", preferring to use mostly estimations based on experience and analogy. In such way, one of several drawbacks is a reduced data gathering from projects, not allowing to do (even at a basic level) statistical analysis, for better estimating the proper efforts value for next user story and – as a whole – of the project. This paper describes the reasons why a guideline for sizing Agile projects with the COSMIC method was required, along with a summary of the resulting guideline. Several agile methods are briefly described, more specifically their expected requirements format and their measurement practices.*

***Keywords***

*Functional Size Measurement, Agile Methods, Software Process Improvement, Software Project Estimation, Practical Measurement Application, COSMIC.*

## 1 Introduction to Agility

One of the most typical activities in human life is to estimate and trying to predict phenomenons of interest. And the more we know and are experienced about a certain phenomenon, the easier is the way to predict it and the lower the relative error we do and the loss of resources (time, people, materials, etc.) we have to use for that activity. Another characteristic for human beings is to be highly and quickly adaptive to any context, in order to survive.

Those two characteristics are – probably – the key concepts that can better summarize what is one of the most flavoured 'buzzwords' of last 10-15 years, that are Agile methods and techniques, properly diffused and summarized in 2001 by the 'Agile Manifesto' and its related twelve foundation principles [2]. Thus, the Agile Manifesto [1] has four main statements[1], usually called 'values':

[1] Recently several first-time agilists are stressing the need of revising the principles supporting the original 'Agile Manifesto' since almost10 years have passed since its publication in 2001. See e.g. this post (http://www.infoq.com/news/2008/08/manifesto-fifth-craftsmanship) and then directly positions

> *"**Individuals and interactions** over processes and tools*
> ***Working software** over comprehensive documentation*
> ***Customer collaboration** over contract negotiation*
> ***Responding to change** over following a plan*
> *That is, while there is value in the items on the right, we value the items on the left more."*

Since its publication, several Agile methods have been proposed to the Information and Communication Technology (ICT) audience, having in return a large acceptance and application in those contexts marked in particular by projects with a dramatic, highly dynamic change of requirements requested by the customer, the need to re-prioritize requirements and their transformation to functioning product features in the shortest timeframe as possible, and from failure of previous methods to do so, being able to deliver the highest value functionalities first. In few words, projects needing to be highly adaptive, maintaining at least the mandatory issue that is the delivery deadlines and values for the established product features. Plenty of methods and techniques were proposed [3], and in order to proper compare them, they could correctly be classified in two main categories:

- Agile Software Development (**ASD**) methods, such as Extreme Programming (XP)[7][8], Crystal Clear [15], Test Driven Development (TDD) [9][10], and Domain Driven Development (DDD)[2][22];
- Agile Project Management (**APM**) methods, such as Scrum [28], FDD (Feature Driven Development) [19] and DSDM (Dynamic Systems Development Method) [21].

## 1.1 Documented requirements in Agile projects

Agile projects will often document their requirements in the form of "User Stories" (US) [17]. A US shortly summarizes the functional requests by customers and takes the following format:

> "As a <user type>, (*part I*)
> I want to <feature or functionality>, (*part II*)
> so that <value or expected benefit>". (*part III*)

A US fits its purpose in an Agile context when it meets the three "C"s characteristics:

---

from Ron Jefferies (http://xprogramming.com/articles/beyond-agile-new-principles/) and Kent Beck (http://java-metrics.com/metrics-process/kent-beck-revises-the-agile-manifesto-for-startups).

[2] A reference website for the DDD community is: http://domaindrivendesign.org/ .

- **C**ard: the US fits on a [small] card: it is small enough to be expressed using the format described above;
- **C**onversation: the US, expressed simply, is a means to promote conversations between the customer and team members so they can reach common understanding of the feature or functionality;
- **C**onfirmation: Exactly what behaviour will be verified to confirm the scope of this US. This information is referred to the Test Plan and usually fits the back of the card.

Mike Cohn, in his book, proposes the following "**INVEST**" criteria to verify the quality of a well defined US [16]:

**I**ndependent: Is this US independent of other US of a lesser priority in the product backlog? This is to avoid planning prerequisite items in early iterations instead of high priority US.

**N**egotiable: Does this US express a need instead of describing a solution? This is to foster creative ideas and exchanges between the customer and the team.

**V**alue for business: Can a final user verify the new system behaviour once the US is implemented? A US must be written in a language and form that are understandable by its targeted final users.

**E**stimable: Does this US and any linked information allow the team to estimate relative effort? This is to avoid writing US at a too high level that becomes hard to estimate and plan reasonably or with missing information at a point where the team has no idea how this US must behave or which data it must manipulate.

**S**mall: Is this US small enough to be developed in a single iteration? This is to avoid spending several iterations developing a high-priority feature without the customer being able to see that feature until it is done, eroding the team's accomplishment feeling.

**T**estable: Does this US have a defined functional test plan? This is to ensure that customer's expectations are well understood regarding the scope of this US. This functional test plan should be aligned with the defined Value for business (see above).

Using the Scrum terminology and concepts, US are gathered as backlog items into a "Product Backlog", that may also contain other items such as defects to be fixed or some product characteristics. Backlog items are prioritized by the product owner (often the customer, or a proxy).

Agile projects that define their requirements in the form of US usually understand and apply these concepts. However, in the course of our professional practice, we

encountered several teams who ignored these quality characteristics and, when using US, they would simply settle for the first two parts of the US sentence, thus not benefiting from well defined US.

But, from our industry practice observations, there are many Agile projects not using US, especially in larger organizations where adoption of Agility meant solely using an APM method such as Scrum, and for which requirement documentation takes other more traditional forms such as Software Requirements Specifications documents (SRS) or, more widely spread, Use Case (UC) documents.

## 1.2 Current measurement and estimation practices in Agile projects

Updating the analysis done in [11], the following sub-sections contain a series of typical situations related to measurement in Agile projects, stressing the flaws for determining possible improvement points.

### 1.2.1 Backlog items relative size and complexity

Looking at Agile projects from a measurement viewpoint, one common drawback is about the estimation process. When requirements are defined as US, teams roughly estimate by experience and analogy the relative "size and complexity" of each US compared to other US in the same project. The given value often fits on the Fibonacci scale (0-1-2-3-5-8-13-21- and so on) and is called "User Story Point" (USP). Looking at all US at a glance, the team decides which value is best for their "average size" US. Then, they assign a USP value to each US, reflecting its relative size and complexity to what they consider their "average" US.

The USP value cannot be considered as a measure. It is a number only meaningful to the team that have assigned it. From one team to another, this value, for the same US, is likely to vary since every team choose which value they consider as "average". Thus, USP is not a real product size unit from a measurement viewpoint and cannot be defined as a standard software sizing measure for that reason.

### 1.2.2 Velocity

Agile teams applying an APM or an ASD method generally measure their project "velocity", a trend indicator defined by the number of USP completed per iteration over time. When a team has work together for some projects, they should have accumulated data from an appreciated number of iterations. This historical data provides an average velocity for use in estimation, once the team knows the USP value for their backlog items. However, not all teams cumulate or have access to historical data, and they often rely on an estimate of their upcoming velocity to perform their preliminary project estimate. Within a project, a team

may use the latest iteration velocity value to adjust the overall estimate of the project or the next iteration estimate.

However, one team velocity cannot be used as a benchmarking value because its USP component is not a standard measure and does not comply with basic metrology concepts [4].

Velocity is an accepted term among the Agile community for productivity, which in turn is rarely accepted as a term for use, mainly for philosophic and emotive reasons.

### 1.2.3 Task effort

At the beginning of every iteration, the team performs an iteration planning activity where top priority backlog items are detailed into tasks of "manageable size", i.e. less than two days of effort, until they reach the team capacity for that sprint. They can then commit to their product owner to perform all backlog items they have planned that fit their capacity, in order to manage the customer's expectancies in terms of functionalities to be shown at iteration review at iteration end.

While performing these tasks throughout the current iteration, each team member will record the remaining effort on every task they worked on.

But how about collecting actual effort data? Unless the organization imposes to collect actual effort on their projects, an Agile team is only concerned with the remaining effort per task. Some teams can derive actual effort, if they are asked for it, by the number of team members times the number of work days in all their iterations, since they were – supposedly – fully dedicated to their project, and that they have made their planning in such a way that there should not be overtime work. However, some roles or individuals may be assigned part-time to the team and it becomes easier to establish the project actual effort if an actual effort recording mechanism is adopted, recording also overtime if any.

### 1.2.4 Burndown

An Agile team provides transparency of progress results by making publicly available its "burndown chart", a graphical trend of the total remaining effort per day in the current iteration, calculated as the sum of remaining effort for all tasks defined in this iteration. The burndown chart can also contain a line representing the "ideal burndown", a linear representation of total effort from day one to zero effort on the last iteration day (see Figure 1), which can ease interpreting current trend and help the team in decision-making, e.g. add a new backlog item to the current iteration if items are being developed faster than planned, or plan to work overtime or redistribute tasks or change implementation strategy if items are

being developed slower than planned, or any other decision the team finds appropriate in their specific context.

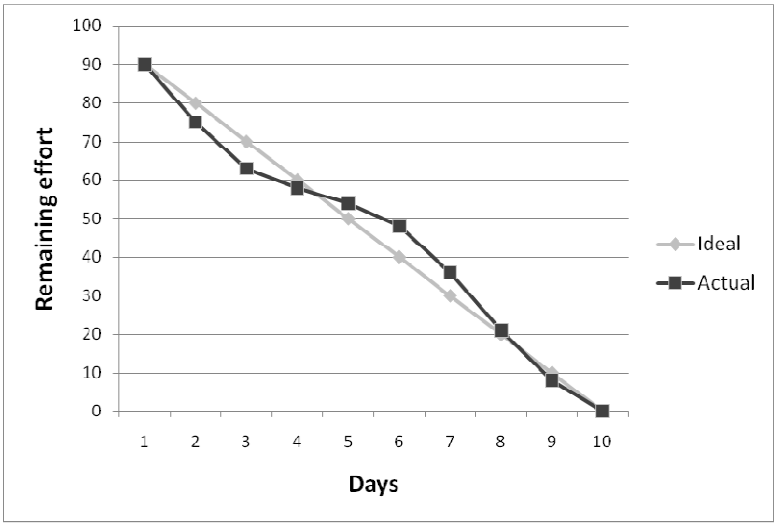


**Figure 1:** Example of a burndown chart.

### 1.2.5 Estimation

In an ideal world, Agile teams can estimate and commit to their next iteration. Period. But we don't live and work in an ideal world and organizational context often requires an up-front project estimate to release the necessary budget for performing the project.

From their known velocity and backlog items estimated USP values, an Agile team can determine which US can be included in the next iteration, and how many iterations should be necessary to complete the project. This preliminary estimate assumes that several things will not change, including:

- The current content of the product backlog;
- The team's average velocity, or factors impacting it with regards to factors used to calculate the velocity from past projects, such as:
  - Team composition;
  - Chosen technology;
  - The development process.

Provision is often added to allow for scope changes during the project, as well as any other changes the team can foreseen. Estimates are often expressed in number of iterations of a fixed number of weeks. Effort estimate is derived from the number of team members, assuming they work dedicated full-time on the project.

# 2 Measurement context of Agile projects

## 2.1 Reasons for measuring functional size

### 2.1.1 Benchmarking

Having the solely effort figures with no "size units" or not shared and common definitions about what is – for instance – a USP makes difficult to perform internal – as well as external – benchmarks. To adequately benchmark an organizational process performance or productivity, it is of primary importance to use a common standardized base measure, such as a published Functional Size Measurement (FSM) method ISO standard. The COSMIC [6] function point (cfp) size unit standardized as one data movement per data group per functional process can be used for benchmarking productivity rate (in cfp/person-month) or unit cost (in hours/cfp), regardless of the technology used to implement the software application resulting from project activities, Agile or not. An important reminder, valid for all FSM methods – including COSMIC – is that they size the software functional product size, not the whole project. Therefore a direct consequence is to carefully read productivity and cost/cfp or any other functional size unit (fsu) figures because of the presence of different elements to be discarded for 'pure' evaluations [14].

### 2.1.2 Estimation

US are typically expressing functional user requirements (FUR). The adoption of any FSM method could be very helpful for doing more reliable estimates and be less subjective in the evaluation. The other side of US is that they do not explicitly deal with Non-Functional Requirements (NFR) that could represent a significant amount of activities and related effort within the project scope. Experienced Agile teams implicitly compensate the NFR impact by assigning a higher USP value.

A US typically contains both FUR and NFR. A FUR can be sized with an FSM method such as COSMIC or IFPUG [24]. NFR can be partly sized with an FSM Method such as COSMIC [25] but not entirely (e.g. project management or QA effort, those derived from organizational/support processes – different logic and/or measures, also estimation by experience/analogy – Delphi). And $US^2$ ($2^{nd}$ generation-User Stories) [12] can help in properly define the scope of each US (in an aggregated form for the whole project) from their high-level definition in the sharing between customers and team members. The different ways to consider (or not) NFR in estimations can lead to visible negative effects in estimates because

applying "nominal" productivity[3] and cost values, strongly affecting the way the team would re-schedule and plan their project.

Next improvement issue from a measurement viewpoint against current Agile practices is the loss of historical data at the project closure, which should be done at the end of each planned iteration in the project.

### 2.1.3 Internal process improvement

An organization better understands that its process has improved over time through objective measurement which includes data of quality and sound data analysis. Many organizations measure their process unit cost (either in hours or money units – dollars, euros, etc. – per size unit) per project, per technology, or per business domain. When they do, they track this unit cost over time to report on process improvement results, typically once or twice a year.

At organizational level, as the process management activities evolve from qualitative status reporting to quantitative analysis, reporting, and decision-making, it is likely to include a shared common unit of measure for comparison over time and internal benchmarks. When transitioning to Agile methods, managers and team members would most probably want to know if the software process performance is improving, compared to their traditional process performance. This answer can only be provided if both processes are collecting and reporting with the same units of measure, e.g. functional size unit such as cfp.

### 2.1.4 Projects and applications governance

Organizations put in place mechanisms and measurement to perform governance of their project and application parameters (e.g. cost, effort, delivery date, quality, scope), not only by comparing actual data against budgeted or planned values – so they monitor the quality of their planning and estimation process – but also by looking at new indicators that include an objective size measure, such as development unit cost – so they monitor the efficiency of the development process – or maintenance relative cost (in $ or € per cfp) – so they can concentrate investigation effort on portfolio applications whose value is outside expected thresholds [32].

[3] 'Nominal' productivity is a term introduced in [14] for distinguishing the traditional way people measure productivity (in our case a fsu against the whole project effort) from what could called a strictly functional' productivity (relating the number of fsu produced to the solely effort related to FURs).

## 2.2 Applying the COSMIC FSM method in Agile projects

Sizing Agile projects requires knowledge about the COSMIC method and also guidance on the particular application type that is being developed or evolved: business application [6], data warehouse applications [31], or Service –Oriented Architecture (SOA) software [26]. The *Guideline for sizing Agile projects* [30] is meant as an extension to these guidelines. Also, improvement in US documentation is proposed to ease FSM [27] of each US, if US is the chosen requirements documentation method.

But the challenges about sizing Agile projects lies in the purpose of the measurement outcome and in the moments at which measurement should be done to fulfil this purpose, namely:

- Benchmarking;
- Upfront project estimation and budgeting;
- Iteration planning and project re-estimation;
- Process improvement monitoring.

Guidance for these specific measurement purposes is provided in the following subsections (more details can be found in the Guideline document). Planning to use historical data can be done in Agile contexts at different moments: by iteration, by observing the whole project as a sum of multiple iterations, and by sizing the resulting version of the software at any given delivery. In any case, the coherent couple of data (size and related effort) is needed.

### 2.2.1 Benchmarking

Organizations submitting or querying data for benchmarking have interest in comparing their project unit cost to similar projects in their industry, regardless of the methodology used to implement it. This also applies for internal benchmarking. For benchmarking purpose, the functional size provided is assumed to be the size of the software delivered at the end of the project, regardless if the software development or management method was Agile or not. Therefore, it would be reasonable for Agile project to provide the FSM of functionalities delivered at project end.

When Agile projects are planning release deployments of defined scope at specific dates, it is possible that a "project" relates to a given software release and that the all planned releases in time constitute a "program" – a series of projects. In that case, FSM would be done on a software release paired with actual effort data for that same release. It would be expected that release development duration be of several iterations (three and more). If there is a software release into

production after every iteration, what is considered as a "project" of a defined scope would have to be defined by the organization and measured accordingly.

### 2.2.2 Upfront project estimation and budgeting

Organizations often require allocating budget prior to begin a software project, based on its cost estimation. The first challenge of a software product owner in an Agile context is to obtain this estimation for a defined project scope. At a very early stage of an Agile project, this scope can be defined as a list of US or any other functional requirements format, with sufficient information to obtain a list of required functionalities. Also, historical data is required on the median size of a functional process (FP) and software process velocity or unit cost for a given business domain and a given technology.

At this point, the project functional size is not "measured" but "estimated" using the number of FP times the median size of a FP. Provisions can be made to alleviate the risk of having an incomplete list of functionalities where the number of missing functionalities is again estimated.

This estimated functional size – that can be called "initial size" – could be compared with the actual size – or "final size" – at project end. Initial and final size results can feed the projects historical database where adjustments could be made on the average size of a FP and the software process unit cost.

### 2.2.3 Iteration planning and project re-estimation

For an Agile team to monitor its velocity in cfp per iteration, FSM has to be performed at the end of every iteration, accounting for each data movement that was added, modified, and deleted during the iteration. At iteration planning, if functional size was estimated, it is an appropriate moment to perform FSM on backlog items considered for the current iteration. With a known velocity, the team can corroborate effort estimate calculated as the size times the velocity with the sum of effort estimated per task. Team members can then look at major differences to understand what is missing before making any commitment on the iteration scope.

Also, if an Agile team has performed FSM on all items in the product backlog and the velocity assumption for early estimate turns out to be quite different from their actual velocity trend (either higher or lower), the team can re-estimate the project backlog remaining effort and work on action planning with the product owner, as appropriate.

### 2.2.4 Process improvement monitoring

Agile teams apply continuous process improvement through a practice called "retrospective" they performed at the end of each iteration[4]. The aim of a retrospective is to modify the current process – people behaviour, tools, deliverables, etc. – to increase the team's velocity at delivering functioning software. An Agile team could monitor its velocity trend in cfp delivered per iteration to better understand the efficiency of their retrospective sessions.

At the organizational level, process owners would want to monitor their process unit cost trend using the same base measures for any project, Agile or not. Since non-Agile projects are not likely applying iterative development, it becomes adequate to perform FSM based on delivered functionalities at the end of Agile projects, as it is done for non-Agile projects.

## 2.3 Potential outcomes of FSM performed at different moments

When a project measured the size of functionalities developed or modified for each iteration and also the size of functionalities delivered at project end, it should be possible to compare the two values to obtain the partial size of rework. Rework can occur when change is requested by the customer during the project or when slicing larger functionalities into smaller ones which development effort would fit in a single iteration. Whenever an Agile team modifies an existing functionality in an iteration, there is a high probability of having to modify existing data movements, which adds functional size units to the iteration but not to the project delivered software.

# 3 Frequent resistance behaviours from Agile team members to FSM

## 3.1 Current practices satisfactory for the Agile team

Agile teams that are familiar and successful with estimating and sizing their projects with USP are likely to resist from changing their sizing and estimating methodology. The fact is that they can still use USP within their projects if they feel comfortable with it. There is value in using USP that can be expressed in the richness of discussions and collaboration among team members. The planning and estimating exercise the team does must be seen as a means to reach common understanding of features and functionalities to develop and enhance.

FSM would then be required at the organization level, especially when it is a common practice within their industry sector to have their software process

[4] "Retrospective – meaning looking back at how the team performed – is defined in details in the Agile literature[20] and can be considered as an evolution of what traditional project management methods called "Post Mortem".

performance benchmarked by independent experts on a regular basis, such as in banking and insurance domains. All that is required is that someone performs FSM – that could be outside the team to avoid disturbing them too much – and obtains actual effort to contribute to the process performance database.

### 3.2 The word "productivity" is taboo

When an Agile team is being told that measuring functional size of their project serves for calculating "their" productivity, they legitimately feel judged and uncomfortable with the idea. Comparing functional size and effort to obtain a value of productivity is not a measure of individual performances but rather a measure of process performance. Nonetheless, Agile teams do measure their project velocity, but that value, often in number of USP per iteration, is only meaningful to them and cannot be used for benchmarking or estimating other projects in the organization. It is more accurate to explain to the team that measuring the functional size of their project software deliverables contributes to establish an organizational baselined unit cost that serves for benchmarking with other similar organizations, establishing estimation models for use by team members to better estimate future projects, and to objectively monitor process improvement derived from their improvement actions resulting from their retrospectives.

### 3.3 Lack of motivation ("What's in it for me?")

Any team member may ask why spending precious project effort to measure functional size if it is not improving project delivery and quality.

First, effort to perform FSM is negligible once the learning curve is absorbed (for direct experience from the field, it could take the time for measuring approximately 10 functional processes) and that functionality expected (or actual) behaviour is understood. Second, measuring the functional size contributes to identify ambiguities about data groups and data movements, which is of value to any team member having to code appropriate data manipulations and movements (e.g. capture, display/print, send, store, retrieve, receive, calculate, transform, etc.). Third, FSM results can contribute to enhance a reliable and more precise estimation model – assuming that effort data is collected and used – that the team could use for upfront project estimations, an aspect of Agile methods that is known to be weak. And finally, knowing the average unit cost of their project may initiate different approaches to iteration retrospective in order to ensure an increasing effectiveness.

In summary, motivation of team members to apply FSM could be addressed with an awareness session or some training on the chosen FSM method.

# 4 Case study results

## 4.1 Incomplete documentation of requirement

A re-development project (the same functionalities to be developed in a new supported technology, with some added functionalities) had scarce requirements documentation (i.e. original documentation was lost or undocumented): a series of screen shots, no data model, and no other form of documentation. To be able to perform FSM in this context required having access to knowledge of functional behaviour and data model, such as an analyst. Measurement was done considering the new project scope and applying the measurement method with the persons having the required knowledge to describe the functional users, the list of functional processes and their triggering events, the objects of interests and their data groups, and data movements. This project had a single release at its end.

Historical data was not available in the organization for FSM results of other projects but every project effort have been carefully collected for years. Collecting the software functional size for any project release is intended to establish a unit cost baseline for estimation purpose.

## 4.2 Early estimate in telecommunication projects

Another experience was done considering enhancement projects within the telecommunication sector. In that case FUR from the customer were formulated as UML Use Cases (UC). These projects are typically quarterly time-boxed projects, delivering new functionalities and/or enhancements of existing ones, with at least two iterations (one for the setup of the environment and high-priority UC, and one with the delivery of main functionalities, to be delivered strictly within the end of the trimester).

Even if – as Cohn said – 'user stories aren't use cases' [18], the formulation of FUR with UML provided by the customer was very useful to approximate with a FSM method yet from the start the number of functional processes to be considered for enhancements, and consequently the reference data model supporting those FUR. Also, in this case, the historical data was not gathered in a regular and more detailed way. Thus the ISBSG r11 dataset for COSMIC project was used as an initial reference for determining reference values for estimating effort, considering effort and size figures for Java, that was the main programming language used.

At the same time, two improvement actions were deployed. Firstly, a refinement of the historical database, gathering data at the BFC (Base Functional Component) level as suggested in [13] and verified in different other studies using ISBSG and organization's datasets was put in action. Secondly, the estimated effort was split by Software Life Cycle (SLC) phases according to the

project team experience, trying to determine the approximated percentage of effort derived from FUR and NFR within the project scope. In order to do it, a refinement from the project task list was done, determining the nature of the process generating a certain task, according to a pre-established process model (in this case CMMI-DEV). In case of too high-described tasks (e.g. only 'analysis' instead of its split into 'functional' and 'technical analysis') were split into two or more sub-tasks, providing a more granular definition of the project tasks list. As a rule of thumb, the 'Engineering' processes leads in general to FUR-related tasks (less than architectural analysis tasks or white/gray testing tasks), while all the other process groups (Project Management; Process Management; Support) will lead to have NFR-related effort.

This cross-check between two different viewpoints: the project management one (using the task list) and the FSM one (using the FSM-related estimation figures), determined an effort range further examined and fixed into in a Delphi meeting. It helped estimators in better determining the 'functional' and 'nominal' productivities for proper scheduling of project roles and balancing the effort, being those resources not all full-time allocated on that project. Again, the determination (even if approximated) of the % of effort types helped in introducing a further discrimination element in the historical database, as a next potential filter to use for clustering projects for next estimations.

The early functional sizing with a FSM method allowed also increasing the quality and timeliness in communicating with the customer, since it clarified by numbers the amount of expected effort and therefore the need for prioritizing all the requested enhancements in different, subsequent quarters from their provider and towards their external customers.

## 5 Discussion

In an Agile context, FSM could be made at two different levels of granularity: iteration level and project level. It is highly probable that measurement made at iteration level provides a sum of functional size greater than the size of functionalities delivered at project end. FSM made at these two levels of granularity serve different purposes and interests: an Agile team would use the iteration level size while the organization would prefer the project level size.

The product backlog can be use not only to estimate the relative level of effort with USP, but also for recording the functional size of the software project. This size data can be maintained throughout the project. At the end of the project, initial and final size, along with project effort can be recorded in the organizational project and process performance database for analysis and decision-making.

## 6 Next steps

The *Guideline for Sizing Agile Projects with the COSMIC method* document is in draft mode and available for free at http://www.cosmicon.com . Reviewers are sought to help improve and evolve the Guideline to a final published state. Also, field trials of the Guideline are under way for feedback on its usage. It would be interesting to verify the time typically spent, the Mean Relative Error (MRE), and the Magnitude of MRE (MMRE) on a dataset of projects using this approach to enrich the current guideline on why, how, and when these data could be use in an Agile context. As adoption of FSM in Agile projects appears to struggle from human resistance, it would be interesting to verify this resistance to measurement with an agile approach while specifically applying an FSM method.

## Acknowledgement

The authors are grateful to Pyxis Technology staff that has helped in reviewing this paper, namely Line Fortier, Ida Iannizzi, Éric Laramée, and Martin Proulx.

## References

[1] ---, *Agile Manifesto*, 2001, URL: http://www.agilemanifesto.org .

[2] ---, *Principles behind the Agile Manifesto*, URL: http://agilemanifesto.org/principles.html

[3] Abrahamsson P., Salo O., Ronkainen J. & Warsta J., *Agile Software Development Methods. Review and Analysis*, VTT Publication # 478, 2002, URL: http://www.pss-europe.com/P478.pdf

[4] Abran A., *Software metrics and software metrology*, IEEE Computer Society, Wiley, 2010, ISBN 978-0-470-59720-0

[5] Abran A., Desharnais J-M., Oligny S., St-Pierre D., and Symons C., *The COSMIC Functional Size Measurement Method: Measurement manual*, v 3.0.1, The Common Software Measurement International Consortium (COSMIC), May 2009, URL: http://www.cosmicon.com

[6] Abran A. et al., *Guideline for sizing business applications software*, v 1.1, COSMIC, May 2008, URL: http://www.cosmicon.com

[7] Beck K. & Andres C., *Extreme Programming Explained. Embrace Change*, 2/e, Addison-Wesley, 2005, ISBN 0-321-27865-8

[8] Beck K. & Fowler M., *Planning Extreme Programming*, Addison-Wesley, 2000, ISBN 0201710919

[9] Beck K., *Test Driven Development by Example*, Addison-Wesley, 2002, ISBN 978-0321146533

[10] Fowler M., Beck K., Brant J., Opdyke W., Roberts D., *Refactoring – Improving the Design of Existing code*, Addison-Wesley, 1999, ISBN 978-0201485677

[11] Buglione L. & Abran A., *Improving Estimations in Agile Projects: issues and avenues*, Proceedings of the 4th Software Measurement European Forum (SMEF 2007), Rome (Italy), May 9-11 2007, ISBN 9-788870-909425, pp.265-274, URL: http://www.dpo.it/smef2007/papers/day2/212.pdf

[12] Buglione L., *Meglio Agili o Veloci? Alcune riflessioni sulle stime nei progetti XP*, XPM.it, February 2007, URL: http://www.xpm.it

[13] Buglione L. & Gencel C., Impact of Base Functional Component Types on Software Functional Size based Effort Estimation, Proceedings of PROFES 2008 Conference, Frascati, Rome (Italy), 23-25 June 2008, LNCS 5089, A.Jedlitschka & O.Salo (Eds.), pp.75-89

[14] Buglione L., *Some thoughts on Productivity in ICT projects*, v1.3, WP-2010-01, White Paper, August 2010, URL: http://www.semq.eu/pdf/fsm-prod.pdf

[15] Cockburn A., *Crystal Clear. A Human-Powered Methodology for Small Teams*, Addison-Wesley, 2005, ISBN 0-201-69947-8

[16] Cohn M., *Agile Estimating and Planning*, Prentice Hall, 2005, ISBN 0131479415

[17] Cohn M., *User Stories Applied for Agile Software Development*, Addison-Wesley, 2004, ISBN 978-0321205681

[18] Cohn M., *Advantages of User Stories for Requirements*, October 2004, URL: http://www.mountaingoatsoftware.com/articles/27

[19] De Luca J., *Feature Driven Development (FDD) processes,* v1.3, Nebulon Pty. Ltd., 2004, URL: http://www.nebulon.com/articles/fdd/download/fddprocessesA4.pdf

[20] Derby E., Larsen D. & Schwaber K., *Agile Retrospectives: Making Good Teams Great*, Pragmatic Bookshelf, 1st Ed., 2006, ISBN 978-0977616640

[21] DSDM Consortium, *DSDM Public Version 4.2*, 2007, URL: http://www.dsdm.org/version4/2/public/

[22] Evans E., *Domain-Driven Design: Tackling Complexity in the Heart of Software*, Addison-Wesley, 2003, ISBN 978-0321125217

[23] Gencel C. & Buglione L., *Do Different Functionality Types Affect the Relationship between Software Functional Size and Effort?*, Proceedings of IWSM/MENSURA 2007, Palma de Mallorca (Spain), November 5-8 2007, pp. 235-246, URL: http://www.gelog.etsmtl.ca/publications/pdf/1090.pdf

[24] IFPUG, *Function Points Counting Practices Manual – Release 4.3.1*, International Function Points Users Group (IFPUG), January 2010, URL: http://www.ifpug.org

[25] Kassab M., Ormandjieva O., Daneva M., Abran A., *Non-Functional Requirements: Size Measurement and Testing with COSMIC-FFP*. Proceedings of the International Conference on Software Process and Product

Measurement IWSM-Mensura 2007, pp. 247-259, Palma de Majorca, Spain, November 5-8, 2007, Edicions UIB. ISBN 978-84-8384-020-7, URL: http://eprints.eemcs.utwente.nl/11754/

[26] Lesterhuis A, Santillo L., & Symons C., *Guideline for sizing Service-Oriented Architecture software*, v 1.0a, COSMIC, 2010, URL: http://www.cosmicon.com

[27] Rule G., *Sizing User Stories with the COSMIC FSM method*, April 2010, URL: http://www.measuresw.com/library/Papers/Rule/20100408%20COSMICstories%20article%20v0c.pdf

[28] Schwaber K., *Agile Project Management with SCRUM*, Microsoft Press, 2003, ISBN 0-7356-1993-X

[29] Thomsett R., *Radical Project Management*, Yourdon Press, 2002, ISBN 0-13-009486-2

[30] Trudel S. & Buglione L., *Guideline for sizing Agile projects with the COSMIC method*, v1.0, COSMIC, 2010, URL: http://www.cosmicon.com

[31] Van Heeringen H. et al, *Guideline for sizing data warehouse application software*, v 1.0, COSMIC, 2009, URL: http://www.cosmicon.com

[32] Vogelezang F.W., *Portfolio Control – When numbers really count*, Proceedings of IWSM/Metrikon/MENSURA 2008, Munich (Germany), November 18-19 2008, pp. 233-244, Spinger-Verlag, ISBN 978-3-540-89402-5.

[33] Wake W., *Customer in XP*, XP123 website, June 2000, URL: http://xp123.com/xplor/xp0006c/index.shtml#PlanningGame.

## Appendix A – List of Acronyms

| | |
|---|---|
| **APM** | Agile Project Management |
| **ASD** | Agile Software Development |
| **CFP** | COSMIC Function Point |
| **CMMI** | Capability Maturity Model Integration (www.sei.cmu.edu/cmmi) |
| **CMMI-DEV** | CMMI for Development |
| **COSMIC** | Common Software Measurement International Consortium (www.cosmicon.com) |
| **DDD** | Domain Driven Development |
| **DSDM** | Dynamic Systems Development Method |
| **FDD** | Feature Driven Development |
| **FP** | Functional Process |
| **FPA** | Function Point Analysis |
| **FSM** | Functional Size Measurement |
| **FSMM** | FSM Method |
| **FSU** | Functional Size Unit |
| **FUR** | Functional User Requirement |
| **ICT** | Information & Communication Technology |
| **IEC** | International Electrotechnical Committee (www.iec.ch) |
| **IFPUG** | International Function Point User Group (www.ifpug.org) |
| **INVEST** | Independent – Negotiable – Valuable – Estimable – Small - Testable |
| **ISBSG** | International Software Benchmarking Standards Group (www.isbsg.org) |
| **ISO** | International Organization on Standardization (www.iso.org) |
| **MRE** | Mean Relative Error |
| **MMRE** | Magnitude of MRE |
| **NFR** | Non-Functional Requirement |
| **QA** | Quality Assurance |
| **SLC** | Software Life Cycle |
| **SOA** | Service-Oriented Architecture |
| **SRS** | Software Requirement Specification |
| **TDD** | Test Driven Development |
| **UC** | Use Case |
| **US** | User Story |
| **US$^2$** | 2$^{nd}$-generation User Story |
| **USP** | Use Story Point |
| **XP** | eXtreme Programming |