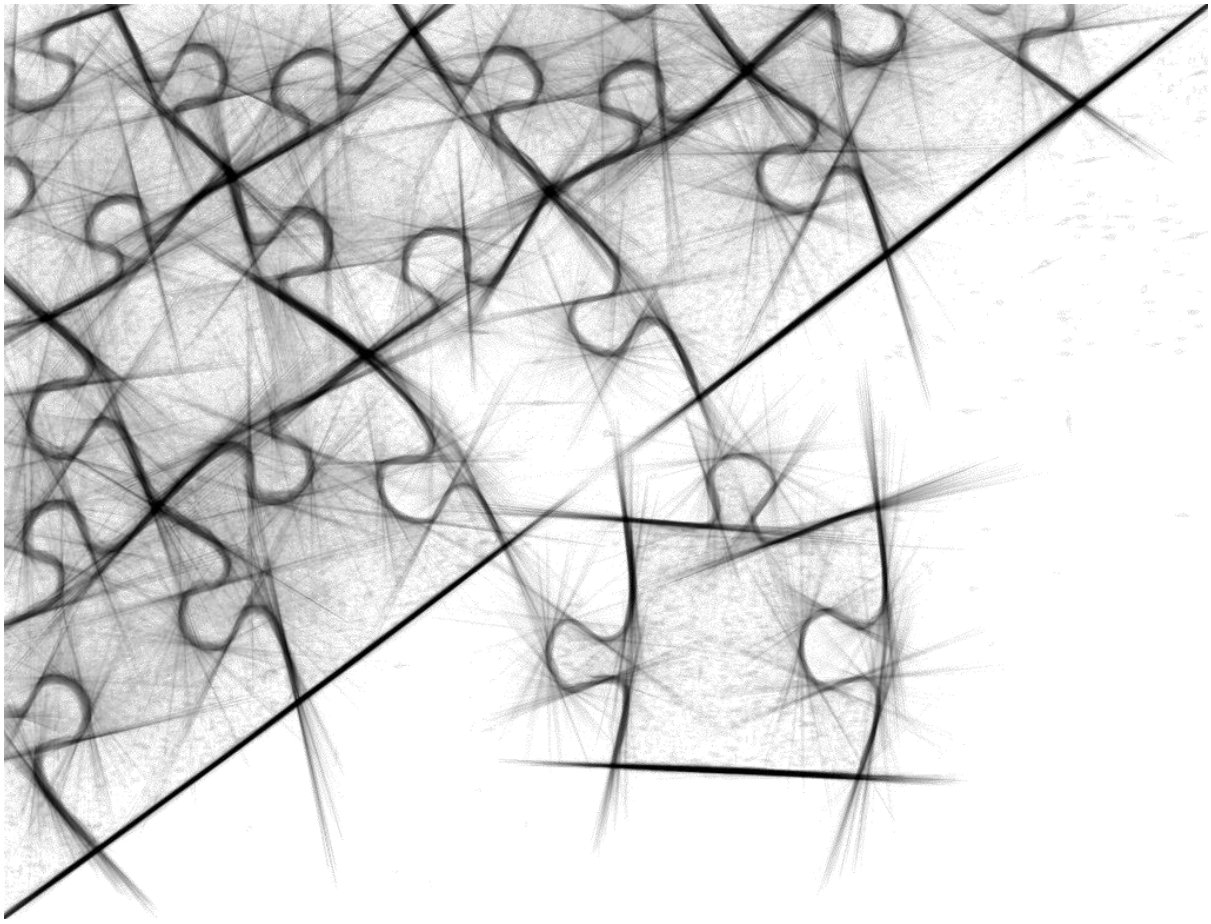

We prefer Facts to Stories

(Managing Agile activities using standardised measures)

COSMIC

IFPUG™

nesma

May 2018

Intended Readers

This paper is for anyone who cares about Agile processes for developing software, who believes that they represent a major advance in how software is delivered, and who wants to make the processes even better.

Summary

Agile processes have brought major benefits to many businesses of faster delivery of software that better meets evolving customer needs. However, the freedom given to individual teams to manage their own processes has made it difficult to manage the activities across Agile teams – what we call managing 'Agile-at-scale'.

To be specific, Agile metrics such as Story Points, may be used by individual teams to manage their own affairs but are very little help for the tasks of planning and monitoring progress across teams, for understanding performance and whether it is improving or not, and for estimating future investments.

Senior management is responsible for setting budgets and allocating resources optimally so as to deliver the greatest value to the organization, and for tracking progress against budgets across the organization. This cannot be done properly for a software group using only typical Agile processes where there are no common performance data across all the teams. These management tasks become even more difficult for an organization that has contracted out its software development to external suppliers that use Agile processes, but that do not use any standard performance measures.

In this paper we explain the challenges that management faces when confronted with the limitations of Agile metrics. We show how simple but effective and long-established ISO standard software measures can fit seamlessly into Agile processes to enable managers to estimate and control Agile delivery at scale. This can be achieved without needing to change any of the underlying Agile processes, and whilst continuing to obtain the benefits that Agile teams can bring in the speed and flexibility of delivering business value.

Mauricio Aguiar, **IFPUG** *(The International Function Point Users Group)*
Charles Symons, **COSMIC** *(The Common Software Measurement International Consortium)*
Eric van der Vliet, **Nesma** *(International Software Measurement and Metrics Association)*

COSMIC, IFPUG and Nesma are international organizations that maintain the ISO/IEC standards for sizing software requirements. These standards are used in industry for estimation, budgeting, contract and project management, supplier performance measurement, benchmarking and other management activities.

1. The benefits of using Agile Processes

The Agile Manifesto has caused a revolution in our understanding of how software should be delivered.

This revolution is clearly bringing great benefits to software customers in the form of:

- earlier delivery of working software that more closely meets customer needs, hence earlier delivery of business value;
- faster response to changing business needs.

There is also evidence that Agile adoption is reducing the cost and incidence of complete project failure. Agile processes help ensure that if a development is going to fail, it will 'fail-early'. This is a big step towards overcoming the endemic software industry problem of frequent and expensive project failures.

2. So what's wrong with Agile measurement methods?

Agile teams aim to plan and control their own progress using internal team metrics of which the 'Story Point' is the most common. However, there is no objective definition of one Story Point. A Story Point has been said to represent [1] 'the size or difficulty' to deliver one User Story. This mixes two separate concepts and gives no clue to Agile practitioners or managers on what is one Story Point and how to measure it.

As a consequence, each team decides for itself what it considers as one Story Point, which becomes its own unit for estimating 'how big is the task?' In practice, this gets interpreted as 'how much *effort* do we estimate will be needed to deal with the task'. One Story Point is commonly established as, for example, one estimated work-day.

Even if all teams in an organization adopt this convention, it then follows that the so-called measure of team 'velocity' (Story Points per actual work-day) is really a measure of the team's estimating accuracy. In no sense is this a measure, as its name would imply, of a team's 'speed' or 'productivity'.

Furthermore a 'velocity' measure is meaningful only within the individual team because teams inevitably differ in their real productivity. Hence measurements of team velocities cannot be relied upon to compare real productivity across teams nor to estimate effort for new software to be built by multiple teams. The meaning of 'team velocity' may even drift over time as team membership and real productivity change.

The process of arriving at a Story Point size for a given User Story – e.g. 'Planning Poker' - is valuable, primarily for helping the team understand the requirements. We have no wish in this paper to challenge this aspect of the process. And the use of Story Points for internal team purposes of estimating the next iteration may be satisfactory as far as the team is concerned. But Story Point 'sizes' are just not suitable for performance measurement and estimating tasks over and beyond individual teams, i.e. for the needs of managing 'Agile-at-scale'. Every organization has such needs to plan for the medium-term and beyond, to budget, to track organizational learning, to enter into software contracts, etc.

To manage these higher-level tasks properly, an organization needs an objective, standardized way of estimating and measuring its software output that is valid for all its types of software and that is independent of the technology being used and the efficiency of its developers.

Long-established standard software sizing methods satisfy all Agile-at-scale needs. These methods measure software sizes that:

- depend only on the software requirements, e.g. as expressed in User Stories,
- may be estimated approximately from early requirements[1] and measured precisely from delivered requirements, i.e. for code that is 'done',
- enable measurements of real productivity (i.e. size delivered /actual effort) that can be used for future effort estimation,
- and hence enable objective comparisons of productivity across different teams using different technologies, etc., monitoring of performance trends, etc.

3. The solution: use standard software functional sizing methods to manage 'Agile-at-scale'

3.1 Outline description of software functional sizing methods

The idea of measuring a size of the functional requirements of software originated over three decades ago. Nowadays, three ISO standard software sizing methods, the COSMIC [2], IFPUG [3], and Nesma [4] methods are used around the world to measure the requirements of all types of software, developed in all types of environments. Each method claims its own advantages.

All three software sizing methods require actual functional requirements to be mapped to their respective model of software which can then be measured. Figure 1 shows the model used by the COSMIC method. Figure 2 shows the model used by the IFPUG and Nesma methods. (In both diagrams, the requirements in bold must be identified and measured to obtain the functional size.)

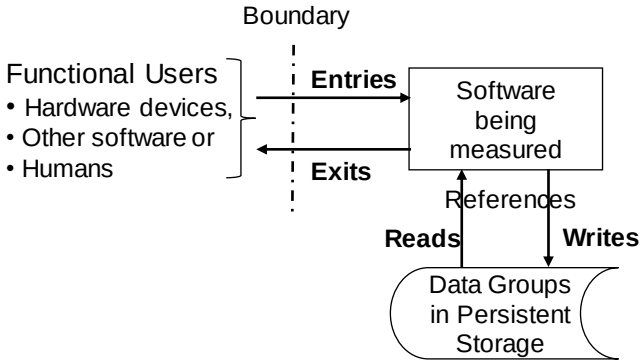

Figure 1. The COSMIC model of software functional requirements

[1] It is impossible to generalise on when requirements are known in sufficient detail that their size can be estimated approximately. See more in section 3.3 on early estimating.

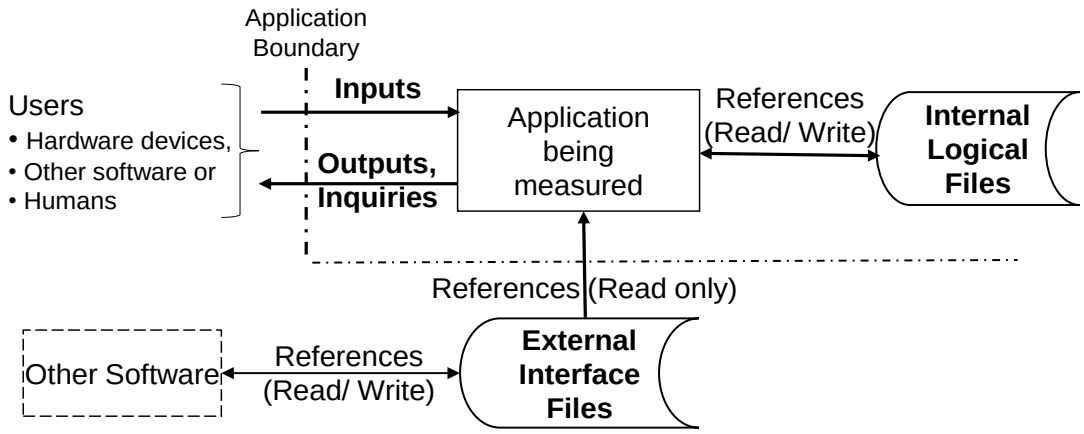

Figure 2. The IFPUG/Nesma model of software functional requirements

3.2 When to measure software sizes in Agile activities?

To discuss the question of when to measure the size of delivered software, it is helpful to distinguish three levels at which estimates and performance measurements are needed. See Figure 3, which is loosely derived from the SAFe framework [5] for managing Agile activities.

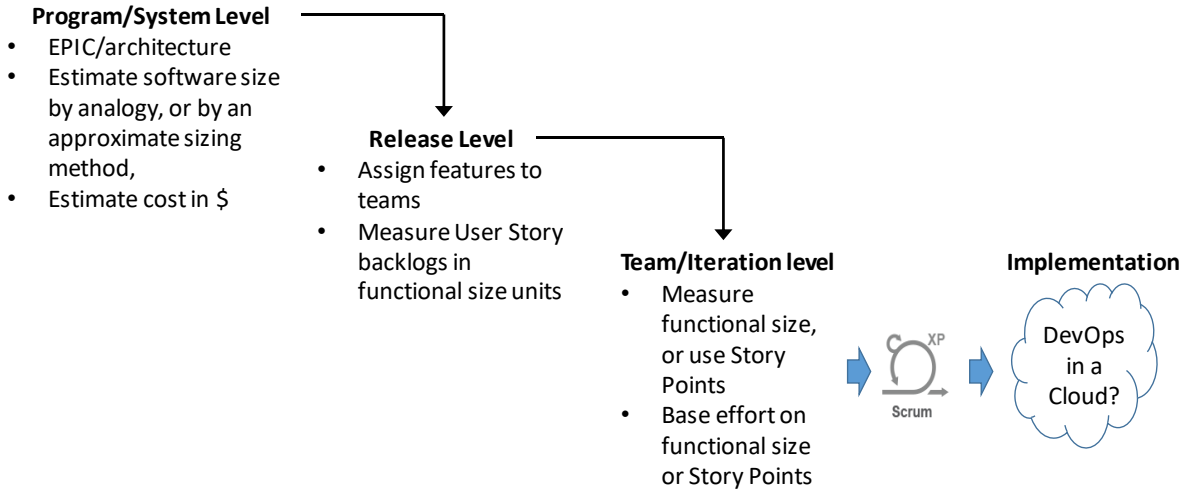

Figure 3. A framework for managing 'Agile-at-scale'

In simple terms, proper estimation, planning and control at the System and Release levels is only possible using objective measures of functional size. Early functional size estimates for a new System or Release can be combined with performance measurements on comparable completed activities to set a budget, to plan its development, allocate resources and establish a basis to control the software scope.

At the team level there is a choice between using a functional size measure or continuing to use existing Story Points. A combination of approaches seems likely to be most successful and acceptable to Agile teams. For example:

- Measure the size of System, Release and team backlogs in units of a standard software sizing method, which will assist objective project planning and resource allocation.

- Measure delivered or 'done' sizes at the Release or Iteration level during a retrospective (or maybe monthly in a DevOps environment) using a standard software sizing method for the purposes of team performance measurement and to gather data for future estimation needs.
- Allow teams, if they prefer, to make their short-term (Iteration) effort estimates using their established Agile process, e.g. 'Planning Poker' to agree on Story Points.

Note however that some organizations use a standard software sizing method to measure delivered software and for estimating effort at all levels from single User Stories all the way up to whole deliverables at the Iteration or Release levels.

Each organization should choose the approach it feels most comfortable with for estimating User Stories (Story Points or functional sizes) at the team level.

3.3 Estimating with functional sizes

Estimating the effort to deliver a new set of requirements at the System, Release or Iteration level of aggregation requires an estimate of the size of the software product to be delivered at that level, combined with 'real' productivity (not 'velocity') data from comparable previously-delivered software at the same level.

Setting up these estimating processes must be done carefully, especially for the following points.

- Establish the size/effort estimation formula by measuring size and effort for a number of completed software deliverables that all share common characteristics, e.g. common technology and other constraints. Whilst productivity data are available from external benchmarking services, we recommend that each organization collects its own data since there are so many variables in Agile activities that can influence performance.
- To estimate software size early in the life of a new system, use an approximation variant of a standard software sizing method. All these methods have their own approaches to approximate sizing.
- In the very early life of a new system when very little is known about the software requirements, estimating by analogy and expert judgement may still be the only realistic option, but the result should still be expressed in the same standard size unit.

3.4 Software size measures in contracts requiring use of Agile processes

When there is a contractual relationship between Customer and Supplier it is common for Suppliers of Agile development capacity to argue for a contract based on Time and Materials (T&M). A supplier's reasoning would be that if the Customer wishes to start an Agile development before much is known of the requirements, it is impossible to bid on any other basis than to offer day-rates for staff with various levels of promised skill.

However, pure T&M contracts are extremely unbalanced. A T&M contract is 100% safe for the Supplier – he gets paid no matter what he delivers - but leaves the Customer with only the total budget as a control on cost. There is no mechanism for the Customer to judge whether progress payments can be justified, whether the Supplier's performance is improving or deteriorating over time, and whether the Supplier is providing value-for-money for the delivered functionality.

A solution that helps balance the negotiating strengths between Customer and Supplier is to contract on the basis of 'price/unit size' where the size of delivered (or developed) functionality is measured by a standard software sizing method. The customer then takes the risk on the total size of his requirements; the Supplier takes the risk on the offered unit price. However, even for this arrangement to work, a Customer will have to define some of his requirements before a Supplier can

reasonably bid a unit price, and will remain responsible for ensuring that delivered functionality translates into business value (which is possible only when the functionality is used). And both parties will need to agree when 'done' functionality can be measured and can be invoiced.

4. Introducing a standard software size measurement method

There is evidently a 'clash of cultures' between the values of individual self-organizing Agile teams and the values and justified control needs of higher management. So if management wishes to introduce a standard software sizing method into an existing Agile development group, the Agile culture must be carefully taken into account so as not to disrupt teams and risk losing the benefits of Agile processes.

The most important questions are when to do the measurements (see 3.2 above) and who will do them. The answer to the 'who' question depends on the scale of the organization and the relationship (contractual or informal) between the software customer and the Agile development teams.

Ideally, teams should do their own measurements and see software size measurement as an integral aid to Agile processes, not as an overhead. (Experienced users of standard software sizing methods find that measuring can also help control the quality and completeness of requirements.)

However, at the level of planning releases or whole deliveries in a large organization, and/or in a customer-supplier contract situation, and/or in the early stages of introducing a software sizing method, it will probably be beneficial to use a group of measurement experts, either:

- internally, e.g. in a Project Management Office,
- or from an external specialist supplier of functional size measurement services.

The advantages of using an expert group of measurement specialists will be their objectivity, speed and accuracy of the measurements.

Whichever organizational solution is adopted, it is vital that the measurement data is collected centrally for shared use and as a basis for organizational learning.

As to how to introduce a software sizing method to an organization that lacks experience in these methods, it is definitely preferable to proceed in a 'bottom-up' manner. In other words, start by introducing the software sizing method at the team level in a pilot project. Roll out sideways and then upwards (to the Release and System levels) as satisfactory results are obtained and the organization gains confidence.

5. Summary: Functional size measures versus Story Points

Standard Software Size Measurement Methods	Story Points
ISO standard methods for objectively measuring a size of functional requirements, independent of technology, process, etc.	A way of assessing requirements for estimating Agile activities, where the meaning of 'one Story Point' is different for each individual team.
The process of measuring a standard software size requires a good understanding of the requirements and may help discover any defects. The resulting size measure is objective and verifiable post-implementation.	A process such as 'Planning Poker' is valuable for understanding the requirements and discovering any defects; estimating Story Points then depends on expert judgement. ('Planning poker' may still be used when measuring a standard software size.)

Can be used to compare performance across teams, enabling internal and external benchmarking, long-term tracking of performance and hence organizational learning.	Aims to be usable to track performance within a continuing team at the level of User Stories and Iterations but in reality only tracks the team's estimating accuracy.
Measurements of delivered sizes and project effort may be used to build estimation models. These, combined with using an approximate sizing variant can enable early project-life effort estimation.	Story Point sizes are so ill-defined as to be very little use for early estimation tasks above the level of individual teams.
Each software sizing method advises on how to deal with Non-Functional Requirements.	Provide no specific help for estimating the effort to implement Non-Functional Requirements.
The only way to control price/performance of external software suppliers.	Impossible to use reliably to control the price/ performance of external suppliers.
Their use may be resisted due to fear of loss of team freedom and due to resistance to control.	Familiarity means that Story Points 'feel good' – so maybe they should continue to be used at the team level.

6. Conclusions

Setting targets and budgets, and measuring performance against these goals is absolutely standard practice in every type of business, private or public, large or small. There is no reason why Agile software development activities should be excepted from these standard business practices.

Story Points are very little help for controlling or estimating Agile activities at levels above individual teams. Measures of functional sizes can be used for these purposes and can be easily substituted for the use of Story Points at these higher levels, without any negative effects on the value of Agile processes. At the level of individual User Stories and Iterations, each organization may decide for itself whether it is preferable to use a standard software sizing method or to continue to use Story Points for short-term estimating.

Introducing a standard software sizing method into an established Agile development group must be handled carefully, as a typical change management project. The aim must be to convince the organization of the collective benefits of using objective control and estimating methods without upsetting the Agile culture and losing the benefits of the speed and flexibility of Agile processes in delivering software.

References

[1] 'Agile Estimating and Planning', Mike Cohn, Robert C. Martin Series,

[2] 'Introduction to the COSMIC method of measuring software, v4.0.1', January 2016, http://cosmic-sizing.org/publications/introduction-to-the-cosmic-method-of-measuring-software-2/

[3] See www.ifpug.org In particular, see 'Function Point Counting Practices Manual', Release 4.3.1', 2010.

[4] See www.nesma.org, In particular, see 'Nesma FPA standard', Release 2.3, 2018.

[5] 'The Scaled Agile Framework', www.scaledagileframework.com