ISBSG
Delivering IT Confidence

The global and independent source of data and analysis for the IT industry

Tutorial

How to estimate the required team size of agile teams?

Published by the International Software Benchmarking Standards Group

2017

Executive Summary

Agile software development and scrum teams have brought many benefits to organizations lately. In many cases the close alignment between small IT- and business teams accelerated the development and delivery of software and increased the quality and added value. For many managers who manage large software development projects this development method was certainly a leap of faith. In a traditional waterfall set-up, it was customary to give much more guidance by obtaining estimates of the programming work that could be done in a release cycle. Based on these estimates the programmers are put to work and then project managers constantly check the progress against the plans.

In the agile/scrum method, project managers don't exist any longer. Substantial control of large software implementation projects is handed over to self-managing teams. Product owners enforce the scrum teams, allowing the scrum teams to work independently in confidence that this will work out fine. This often feels scary because stakeholders and financial controllers often put final responsibility for actual deadlines and targets on their heads head. Productivity and quality remain essential and necessary functionality needs to be finished in time.

Most Agile teams have adopted some variant of Planning Poker, which uses the Story Point measurement method to estimate the number of hours necessary to implement a certain product backlog item. Product owners use this number to determine the product backlog items that can be delivered in the next few sprints. For the teams, this usually works fine. However, Story points should not be mistaken for a **size measure**. It's **an arbitrary measure of effort**, that can't be compared between teams. Therefore, story point metrics are not useful for management decision making on a higher level than the team, even though many organizations don't realize this. For processes like long-term estimation, budgeting, team size estimation, productivity measurement and benchmarking, **a standardized metric of size** is still necessary to estimate the output to be delivered and to measure the output delivered by the teams in a standardized, objective, repeatable and defensible way. In case of outsourcing contracts, this notion becomes even more important, even crucial.

For agile teams, it's usually quite easy to estimate the cost. This is roughly the team size * the duration * the blended rate of the team * the average hours per team member per month.

The important management question however is: **'How many people should be in the team in order to have an X amount of functionality ready on moment Y in time, and how can I measure and keep a grip on progress against during the course of time'.**

In this tutorial, it is explained how this question can be answered using international standards, proven cost engineering techniques and ISBSG data [1].

Table of Contents

Agile Team Estimation

For Management in organizations that have adopted the Agile way of working, longer term estimation is a real challenge. On the team level, the non-standardized Story Points are used for sprint planning and for the estimation of the effort necessary to implement certain product backlog items. Story points is an arbitrary method to assign points to product backlog items in a 'wide-band Delphi' [2] kind of way. So, the team at some points agrees to the amount of story points assigned and the product owner can use this information in prioritizing and planning the sprints.

Story Points can be used perfectly to estimate the effort needed to implement a well described and product backlog item. However, in most cases the product backlog will also feature backlog items that have not been described properly and need to be refined later into detail. It's not possible for the team to make a realistic assessment of the number of story points these features should be granted. Also, it's a fact of life that changes will occur and that rework after sprints will be necessary from time to time. But how much?

While very useful, the story point metric can't be used to estimate for the longer term. For instance, an actual management question could be '**How many people do I need in the team in order to have all the necessary functionality ready 6 months from now when a certain law takes effect? And what will be the cost?'**. The number of people and the timeframe, together with the internal employee costs per team will determine the costs, which are rather easy to calculate for agile teams. However, the main question is about the output that will be ready 6 months from now at which number of people in the team. Which portion of the product will be ready when put 5,6,7,8,9 or even more people in the team? Here, the concept of **productivity** becomes evident. It all depends on two questions:

1. Which functionality needs to be ready at moment X?
2. What is the likely productivity of the team per team size?

These are the typical questions that can be answered by applying software cost engineering techniques, which have proven accurate in the past.

To answer question 1, it is necessary to understand that there is a standardized way to measure the functional size of software. There is an ISO/IEC standard for functional size measurement methods. The methods Nesma [3], COSMIC [4] and IFPUG [5] all comply to this standard and are standards of their own. The Nesma and COSMIC methods have well documented and easy to use high-level variants that can be used to fairly accurately determine the functional size when not all requirements are described yet, or when described but high-level, not in a detailed way. These methods are easy to learn (1-day training) and prove a lot of insight into the functional size that needs to be delivered.

To answer question 2, analysis of historical data of similar projects or teams will provide a lot of help. The ISBSG repository Development & Enhancements [1] provides an Excel sheet with over 8000 projects, releases and (sets of) sprints. Simply selecting the most relevant similar projects and analyse the reported productivity regarding statistics like the minimum,

percentile 25%, average, median, percentile 75% and maximum productivity figures. This can then be used to create the long-term estimation or for other activities, like benchmarking a commercial offer.

A 'simple' Example

Software Cost Engineering is more complicated than depicted in this example, but the example should give a good idea how the concept works.

The example: An organization needs to deliver a software product in 6 months and a list of high-level user stories that need to be realized ('Must haves') is provided. Using the high-level Nesma FPA method, the Scrum master is able to determine the functional size of the functional user requirements: 1300 Nesma function points. The reported accuracy of this method -8% to +15%.

So the functional size of the product that needs to be realized by the team is:

- Min: 1200 FP
- Likely: 1300 FP
- Max: 1500 FP

Then selecting a relevant subset from the ISBSG repository gives the following productivity statistics.

Statistic	Hours/FP
Min	4,5
P25	5,8
Median	7,5
P75	8,3
Max	11,2
Average	7,8

Table 1 – Productivity dataset statistics

In this case, the organization understands that they are most likely not in the upper half of the table, as they are not on a high level of maturity yet regarding software development and agile adoption. The realistic range is determined to be:

- Low: 7,5 hours/FP
- Likely: 8,3 hours/FP
- Max: 11,2 hours/FP

So, the overall estimation can be made:

- Min: 1200 FP * 7,5 hours/FP = 9000 hours
- Likely: 1300 FP * 8,3 hours/FP = 10790 hours
- Max: 1500 FP * 11,2 hours/FP = 16800 hours

Now, software cost engineering techniques would be required to factor the effect of possible rework after sprints and the changes in already existing functionality during the time period. From experience, a 10% uplift should be realistic. Therefore, the new (rounded) range is:

- Min: 10000
- Likely: 12000
- Max: 18500

Assuming a person works on average 140 hours per month (corrected with vacation and sick leave), the estimate becomes clear:

Min: 10000 hours / 6 months / 140 = 11,9 people

Likely: 12000 hours / 6 months / 140 = 14,3 people

Max: 18500 hours / 6 months / 140 = 22,0 people

The organization decides to put 15 people on the project and split it up into 2 teams. The fact that there are now 2 teams influences the productivity, but for the simplicity of this example this effect is ignored.

Against a blended rate of $ 80 per hour, the total cost is calculated: 15 people * 140 hours * 6 months = $ 1.008.000.

Conclusion

Although simplified, this example shows how management can easily determine a long-term estimate of the cost, the necessary team size and the number of agile teams that are required to implement a set of user stories on a specific moment in time. This estimate can be carried out using proven software cost engineering techniques, an international ISO/IEC standard to determine the functional size of the software to be implemented and relevant historical data of similar projects and teams.

The Cost estimate can be baselined in a Basis of Estimate document [6]. After each sprint, the number of function points realized can then be measured and the actual amount of sprint rework can be measured. The actual productivity can therefore be monitored, which might lead to a re-baseline of the estimate and/or an adjustment to the team size.

This method enables product owners to estimate more accurately and to monitor progress more accurately, as the estimate is made and the progress is tracked based on an international size standard instead of an arbitrary method to estimate effort.

References

[1] ISBSG D&E repository 2017 R1, http://isbsg.org/project-data/

[2] Wide-band Delphi technique: https://en.wikipedia.org/wiki/Wideband_delphi

[3] Nesma: www.nesma.org

[4] COSMIC: www.cosmic-sizing.org

[5] IFPUG: www.ifpug.org

[6] Basis of Estmate for Software Services: http://nesma.org/themes/estimating/basis-estimate/

Appendix A: Agile ISBSG data

The data used for this tutorial is the 2017 Release of the Development and Enhancement repository of ISBSG, containing over 8.000 development projects, releases and sprints [1]. The data is collected in a wide range of industry types and concerns a wide range of application type. Of the 8.000 projects, 169 have indicated the development type 'Agile' or 'DevOps'. It is possible that there are more agile and/or DevOps projects in the data, but that the development method was left blank or was otherwise not submitted to the repository manager of ISBSG.

The programming languages used in the 169 projects are listed in the next table.

Dataset Characteristics	
Programming Languages	
Primary Programming Language	**N**
.Net	28
ABAP	1
ASP.Net	1
C#	39
C++	2
Groovy	4
Java	29
JavaScript	2
Mendix	1
Oracle	49
PL/I	1
Proprietary Agile Platform	12
Total	169

The data is from a wide variety of industries, predominantly government and banking.

Dataset Characteristics	
Industry Sector	
Industry Sector	**N**
Banking	28
Communication	1
Education	14
Electronics & Computers	3
Financial	1
Government	86
Insurance	1
Logistics	2
Manufacturing	3
Medical & Health Care	3
Other	1
Service Industry	14
Utilities	5
Wholesale & Retail	1
Unknown	6
Grand Total	**169**

Appendix B: The International Software Benchmarking Standards Group (ISBSG)

The ISBSG is a not-for-profit organization founded in 1997 by a group of national software metrics associations. Their aim was to promote the use of IT industry data to improve software processes and products.

ISBSG is an independent international organization that collects and provides industry data of software development projects and maintenance & support activities in order to help all organizations (commercial and government, suppliers and customers) in the software industry to understand and to improve their performance. ISBSG sets the standards of software data collection, software data analysis and software project benchmarking processes and is considered to be the international thought leader in these practices.

The ISBSG mission is to help YOU and your organization improve the estimation, planning, control and management of your IT software projects and/or maintenance and support contracts.

To achieve this:

ISBSG maintains and grows 2 repositories of IT software development/maintenance & support data. This data originates from trusted, international IT organizations and can be obtained for a modest fee from the website www.isbsg.org/project-data/

Please help us to collect data ISBSG is always looking for new data. In return for your data submission, you receive a free benchmark report that shows the performance in your project or contract against relevant industry peers.

Please submit your data through one of the forms listed on http://isbsg.org/submit-data/

Partners This page will help you to find an ISBSG partner in your country:

http://isbsg.org/meet-isbsg-partners/