Agile and (financial) reporting

Author: Han Schaminée

Contributors: None yet

Version 0.1a

Introduction

The implementation of Agile practices involves much more than just the introduction of a new development process. It also impacts areas like HR and F&A. This document covers some required changes in F&A. As mentioned in other discussion papers, Agile provides a much better way to deal with the intrinsic uncertainty of product development. It therefore is also much better able to meet accounting requirements to provide a factual overview of the status of the assets under development.

Typically, the role of F&A in an organisation is to provide (financial) support for business decisions and to report on the (financial) status of the business. Both aspects will be covered in the document. It is also shown how IFRS requirements are met.

There is a lot of innovation going on in financial reporting. I recommend to read " beyond budgeting".[1] Not all managers, however, have picked up the innovation yet and maintain traditional methods for accounting and performance management.

Executive summary

I am not in favour of big dashboards, but it makes sense to track some indicators on development progress (CFD) and quality. These reporting systems can also be used for financial reporting and meet the IFRS requirements

Reporting also helps in improving estimates required for decision making. Some simple systems are proposed.

A minimal set of metrics

Metrics is good when it supports decision making. I have seen too many examples of extensive dashboards that only served to kill the hunger for data, but where the information was not used to steer the operation. Very often, the actual situation in product development cannot be captured in a few KPIs. Like Mathieu Weggeman describes in his book[2], some people manage development departments like they would manage a cookie factory. He recommends managers rather understand the work of the professionals and create conditions these professionals can be successful.

I have always preferred to steer on a very limited set of metrics. Such a minimal set is described below.

Progress on feature development

I believe Cumulative Flow Diagrams (CFD) is an excellent way to report progress. For a good description, I refer to CFD . You can read a lot from the CFD, for example

- Is the velocity enough to meet the deadlines?
- Is there specification creep?
- Do the teams manage WIP (work in progress) and cycle time?
- Are there bottlenecks?

The first derivative of the burn-up curve in the CFD is the velocity: the number of story points that meet the definition of DONE in a certain time interval. Velocities vary over the sprints due to the uncertainty in development projects. But that is not a problem, a moving average gives a good impression about the team's

[1] http://bbrt.org/what-is-beyond-budgeting/
[2] Mathieu Weggeman: Leidinggeven aan professionals, niet doen (Dutch)

or ART's velocity and can therefore be used in predictions or to define corrective actions to keep the deadlines. It can also be a good indication how productivity developers over time. But velocity should, however, never be used to compare productivity between teams, as their basis can be different, and other reasons than productivity can cause the difference.

Sometimes I ask explicitly for reporting on cycle time, as cycle time is a key factor in productivity.

Progress on quality

After productivity, the most striking advantage of the Agile way of working I have experienced is the impact on quality. The fact that each feature is immediately brought to expected quality levels boosts quality. And recent studies show that defect density is proportional to WIP (Work in Progress), which in itself is not a surprise, as we know how bad we are in multitasking.

I have also noticed that focus on quality itself also boosts productivity. So, sufficient justification to track quality.

The most widely used indicator is the (cumulative) number of all and of closed defects, often split over various defect severity categories.

The slope of the curve is the defect resolution velocity, which is useful to track as it is an indication of how many defects a team can solve in an iteration and therefore can be used in predictions. And, as explained below, the velocity can be controlled by adjusting the capacity bucket allocated to defect fixing.

At any moment in time, the difference between total number of defects and number of closed defects is the number of open defects, and this is often also shown as a separate indicator. In order to manage WIP, I often defined a ceiling for the number of open defects. If the number is too high, the team will spend more time on defect fixing and less on new feature development. This will help productivity and higher feature delivery rate on the longer term.

A last one related to defect resolution is the average resolution time. Remember the key effect time has on organisational productivity. This justifies tracking this separately, as it will immediately reduce the defect WIP.

Of course, the number of reported defects can remain very low, when testing is insufficient. Therefore, a good indicator of the quality of the test is essential. But it is not so easy to define such an indicator. I have seen people tracking the number of (automated) test cases, or the lines of code covered by tests, but they all easily lead to window dressing. In the end, I found that the best indicator for the quality of the test is the number of defects that have leaked to the customer of the organisation.

All the indicators above are lagging indicators in the sense that you measure defects that have been found. Better would be to (also) define metrics on the root cause of defects, which has more to do with code complexity and code review practices. The ISO25010 standards mentions a number of good practices and there are commercial tools available to measure against these practices. I have been impressed by the impact the use of these tools has on quality.

Other measures

Agile software development is about teams. It is well known that team happiness has a huge effect on the productivity of the team. It therefore makes sense to measure team happiness, again not to compare between teams, but primarily to track fluctuations in team happiness within a team.

User stories are derived from features and features from epics. The implementation of epics may span many iterations and are therefore difficult to track. I find it very useful to regularly report on finished features per epic. Also here, a CFD is very useful. It gives a different projection than the CFD per release train, as the work in the release train covers many epics.

Don't measure

It is also good to add what I believe you should never measure. One of these things is team predictability. When teams feel, they are measured against predictability, they will start inserting buffers to cope with the uncertainty, which immediately leads to lower productivity.

But of course, it makes sense to have a feedback loop on the T-shirt size estimates from the actual user story estimates. It will help to continuously calibrate how many story points in average for example a LARGE feature is.

Also, don't measure hours spent on the development of a certain feature. Nobody is interested in sunk costs, except for improving predictions. And measuring velocity is a better way to improve your future estimates.

At last, don't measure hours spent. I have seen too many examples where time sheets were filled randomly. I have also seen examples where people reported the project should be half way, because half of the hours planned were spent. It is better to measure progress in in terms if features completed (weighted for the size of the feature, we call it story points).

Financial reporting

These metrics, combined with some standard F&A reporting is enough to meet the requirements for financial reporting.

The personnel costs per team are fixed

The Agile way of working assumes stable teams. So, it is clear for all the teams in the various release teams, who will be in the team and working on the user stories assigned to the team. Monthly costs per team can easily be calculated. As other F&A systems often keep track of holiday and training, even the net days spent on product development can be determined and translated into costs.

All other personnel costs are allocated to the teams as overhead

Teams are at the core of Agile development. All other people (for instance the SAFe program and portfolio level) are seen as overhead and are only there to create conditions for success for the teams. As overhead typically is less than 15%, no activity based costing is required to allocate these costs to the teams.

Each team defines per iteration buckets for feature development and defect fixing

Teams work on feature development (user stories) and other stuff like defect fixing, sales support etc. At the beginning of a release iteration, buckets can be defined for feature development, defect fixing and other activities (for instance, 30% defect fixing). These buckets are derived from the historical velocity curves for user story development and defect fixing and the number of must have user stories and must fix defects.

Monthly costs per user story can be determined

From the actual realised velocity (story points delivered), the bucket for feature development and the costs of the team, the costs per story point can be calculated for each team in the past month. This can be used to calculate the costs for all user stories delivered.

Monthly costs per feature can be determined

With the costs per delivered user story, we can calculate costs spent on features. Note that user stories aim to be finished within one sprint and therefore do no span several months, where features can.

Feature capitalisation can be determined

Per feature it is known if the feature can be capitalised and when amortization will start. So, total capitalisation and amortization in the month can be determined.

Other financial reporting

The ultimate judgement of the productivity of an (R&D) organisation is if they create sufficient return on the R&D investment. A good measure is EBIT/R&D. It is proven that in many different industries, best in class companies realise EBIT/R&D of around 2 to 2.5 .

Decision support

During product development, many decision have to be taken. We discuss two that are relevant for reporting.

Feature priority

When features are developed incrementally and the WIP for features under development has to be kept at a minimum, features have to be prioritised. There may be many reasons to pull features forward including:

- Highest return on investment
- Lowest costs of delay
- First needed in customer application
- Be first in the market with a feature
- Highest risk first
- Etcetera …

It is worthwhile to regularly revisit the priority allocation as insights may change continuously. The Agile way of working and its incremental development allows for this regular re-visit and in this way, can avoid a lot of waste. And avoiding waste leads to higher productivity.

In many cases an estimate is required what it takes to develop the feature. As also the estimates of the benefits are often not very accurate, there is no need to have a highly accurate estimate of the effort.

We make T-shirt size estimates of the features to be developed. These estimates are made by experienced people, because we don't want to waste too much time of the team on the estimate. We assume that the large number of features will make the overall estimate more accurate (economy of scale). T-shirt size estimates are translated into story point using standard mapping parameters per release train (eg. Large corresponds to 21 story points).

During execution, we continuously monitor these mapping parameters. As the remaining work is more comparable to the work already done with incremental than with traditional development models, we may assume we can have more reliable estimates for the remaining work. When features are translated into user stories and user stories have been estimated, there is a feedback loop on the T-shirt size estimates. This will also help to improve the quality of the key parameters used to translate T-short sizes into story points.

Sometimes, there is too much uncertainty to provide an estimate with a reasonable accuracy. This need to be recognized by the organisation and the teams should not be pushed to make an estimate. It is better to take a bite, just do some user stories to get some feel for the total work involved (highest risk first).

Capacity allocation

One may say, the organisation operates optimally when all features are developed aligned with their priority and that you don't have to worry about the outcome, as it is optimal anyway. It leads to one of the misconceptions of Agile, that you will see the features when they are ready. But in many cases, organisations have to deal with commitments. Commitments may be made for good reasons (synchronisation of activities) or bad reasons (predictability is valued more than productivity), but commitments still exist.

The advantage of incremental development is that the organisation has early indications whether the velocity is sufficient to meet the commitment. But is assumes we have a good feel for the remaining work. As said, incremental development will help to create much better estimates of the remaining work by using a feedback from the work done, where there is more similarity between the work done and the remaining work than with traditional development models.

When the velocity is not enough to meet the commitments, capacity may be freed up by de-prioritizing non-committed features. General purpose, cross functional teams help, as user stories may be allocated to any team, and the organisation faces less the problem that one team is the bottleneck for many features. An organisational set-up with feature teams rather than component teams also helps here. Adding capacity and build new teams is also possible, but has a much longer lead time before it will show effect. When all the teams are fully loaded with committed features, the organisation has no room for corrective measures. These organisations will show anyway problems of congestion resulting in lower productivity.