apm Planning, Monitoring and Control
Specific Interest Group

Agile and Earned Value

A white paper

October 2013

Author – Stephen Jones, Sellafield Ltd

This document is a whitepaper produced by the APM Planning, Monitoring and Control SIG. It represents the thoughts of the author and the group and not necessarily the views of APM. It is intended to stimulate discussion on this subject and feedback on the contents of the paper should be sent to pmcsig@apm.org.uk

Summary

There are numerous articles about earned value and Agile freely available on the internet. The purpose of this paper is to synthesise this information by giving worked examples and drawing on personal experience.

Earned Value Management (EVM) is a good practice approach used for the planning, management and control of projects and programmes. It is a project management technique which measures cost and schedule against a baseline. The result is a simple set of metrics providing early warnings of performance issues, allowing for timely and appropriate adjustments. The information generated helps to keep the project team focused on making progress.

Agile software development methods have been shown to be effective in software development, giving faster results with higher quality, and meeting changing business needs. There are some misconceptions that EVM techniques are too difficult to implement effectively on an Agile project, and that EVM could not easily cope with changing requirements. Many authors have shown this not to be the case, adopting hybrid theories and giving them interesting names like AgileEVM.

This paper will discuss the EVM techniques which have been adopted to provide the benefits of traditional EVM in Agile projects.

The use of earned value management techniques with Agile software development

It has traditionally been seen as difficult to accurately forecast final project costs for any software development project. The actual performance of a project cannot be converted easily into a forecast, usually because estimates have been put together using levels of effort, rather than being deliverable based. The best option that project managers often have is to use inaccurate estimates of work done.

What is Agile and Scrum?

Agile software development methods, often referred to as SCRUM, have been developing over the last 15 years. The core principles of the Agile methods are:

- Iterative and Incremental: software is developed in a series of short (2 – 4 week) iterations known as sprints
- People centric rather than document centric: the whole team is responsible for planning and delivering each sprint. Teams are encouraged to be self-organizing and led by a SCRUM Master
- Scalable: Teams are small, 7 to 10 people and scaling happens by adding teams to the project, rather than people to a team.
- Enables change: at the end of every iteration requirements may change, enabling a better understanding of user requirements;
- Priorities focus: in every iteration, only the features that are of the highest importance are delivered;
- Regular inspection of product and process: at the end of every iteration the delivered product features are demonstrated, and the effect on the product is considered.
- Simple rules of credit: 0% or 100% if the demonstrated features are accepted by the client
- Continuous improvement: at the end of every sprint the team inspects the way they delivered the features and agrees on process improvements.

A full description of Agile Methods is beyond the scope of this paper. Suggested reading - Integrating Agile Development in the Real World (Peter Schuh) and Agile and Iterative Development: A Manager's Guide (Craig Larman).

Jargon buster

SCRUM is big on jargon, probably due to the creative nature of the people employed in the software development industry. I cannot think of any other discipline where you physically start with nothing and produce something of use, which has value and in some cases like social networking people soon become to depend on.

Some often used terms are:

- Scrum – this is the delivery team
- Scrum master – this is the person leading the team
- Product owner – The person responsible for deciding the required product features.
- Product backlog – a prioritised list of features to be delivered.
- Sprint – a two or four week period in which the delivery team develops product features.
- Feature – a product component that the product owner and customer recognize and value.

- Story – a description of a feature. A story should be **clear, feasible and testable**,
  A story is *clear* if all scrum team members have a shared understanding of what it means.
  A story is *feasible* if it can be completed in one sprint, according to the "definition of done".
  An item is *testable* if there is an effective way to determine if the functionality works as expected
- Story board – often used to describe what is to be done, what is in progress and what is done.
- Story points – each story is awarded a number of points based on the level of definition and effort required to deliver. A high number means the story is less defined and more effort is required.
- Grooming - organising the backlog, analysing the customer and user feedback, integrating the learning, deciding what to do next, splitting high value stories into smaller ones, getting the stories ready, i.e. making sure they are clear, feasible and testable
- Ceremonies - meetings which take place before, during and after each sprint

Benefits of Agile over traditional Waterfall methods

The benefits of Agile over the Waterfall method are that there is less time spent documenting what needs to be done and more time spent doing the development. This is possible because Agile enables change. With Agile you get the developers doing what they do best, cutting code, as early in the process as possible. This is a lot sooner than in the Waterfall method, and the customer gets to see some results much quicker which will increase confidence that you are actually delivering something. This is possible with Agile because you typically only need 80% of the requirements defined before you can start cutting code.

The Waterfall method has attempted to achieve this by using rapid product development, and prototyping, but these still require a clear definition and understanding of the clients requirements.

Agile enables change, therefore if you show the client something they don't like, or they change their mind, or it was not what they meant, then you only lose the duration of the sprint (typically 2 weeks). Using the Waterfall method it could be several months or years before anything is demonstrated, at which point change control is required and significant sums of money have been spent.

Estimating the baseline

The baseline is estimated like any other project. It starts with a studies phase where the functional specification and business case are developed, the risks assessed, work breakdown structure (WBS) created, etc. It is important that the backlog is created early doors, however the actual definition of each story does not need to be that well defined. The level of definition will influence the number of story points in the backlog.

There are tools and techniques available for estimating the story points for each story in the backlog, e.g. Agile Poker.

What is Agile poker?

Agile is people centric, therefore the team decide on the value of each story by playing cards, each card is valued 1 to 100. The process is simple, someone reads out a story, everyone playing now scores the story based on the level of definition and complexity. 1 = well defined, easily to develop, 100 = not well defined and will take the entire sprint to develop. In addition there are two special cards "?" which means more information required and infinity which means the story cannot be estimated. Everyone plays their card, after which the persons awarding the lowest and highest values explain the reasons for their score. After hearing both explanations the team play their hand again taking into account what has been said. The average value of all the cards is now used for that story. This reaches a consensus quickly and uses the team who will deliver the story to estimate it, which gives some sort of consistency, although the actual value (£) of each point is unknown.

The cost and schedule estimate can be underpinned by the number of points in the backlog, divided by the estimated number of points which can be delivered in a sprint, and multiplied by the duration of a sprint (usually 2-4 weeks). The number of members in the scrum is known, as are the charge out rates, this will enable a baseline estimate to be produced for the development phase. Project management overheads, software and hardware purchases are then added. Finally estimating uncertainty and contingency are modelled (using Monte Carlo) and added to the baseline to give a P50 estimate, i.e. a cost and schedule with a 50% probability of being achieve, other organisations may use different values, e.g. P80.

My experience does not stretch to using pure Agile, i.e. using Agile from day 1. The Agile process was only truly adopted for detailed design and project execution. The project had to be delivered using the existing company "gated" project management system. This resulted in a hybrid system being developed to align the two.

The project concept stage focused on developing the business case, the project execution plan, commercial strategies and the identification and selection of technologies to be used. The preliminary design phase focused on producing the user requirement specifications, whilst the detail design phase developed the backlog.

This was the first major IT project undertaken using Agile by the company, therefore during detail design a series of pilot sprints were planned to verify the process and gain some experience of using Agile with the company EVM system. These pilot sprints enabled the velocity of the team to be calculated, i.e. how many points the team could deliver each sprint. The intention was to use this information to underpin the estimate.

How to calculate planned value (PV), earned value (EV) and actual cost(AC)?

From this point I assume the reader knows how to calculate schedule performance index (SPI) and cost performance index (CPI).

- Budgeted cost of work scheduled (BCWS) value is read from the schedule.
- Actual cost of work performed (ACWP) is actual spend, the value is taken from the finance system

For simplicity, the worked examples assume the estimated value of the backlog is £100,000. The backlog will not be the entire scope for the project, i.e. it does not include things like equipment purchases or project management costs. These would be estimated as per normal EVM and the earned value for none backlog items would be earned based on the normal rules of credit. The value of project management should be earned using apportioned effort, to prevent poor performance from being masked by level of effort activities.

Typical backlog:

Story Cards	**Estimated (story points)**
Card 1 – Splash screen	10
Card 2 – Enter login name password	10
Card 3 – Verify password	10
Card 4	30
Card 5	10
Card 6	10
Card 7	20
Card 8	10
Card 9	10
Card 10	20
Card 11	10
Card 12	10
Card 13	10
Card 14	10
Card 15	20
Card 16	10
Card 17 – logout user, clean up files and shutdown application.	50
Total	**260**

In simple terms, the earned value is the total number of points delivered divided by the total points in the backlog multiplied by the value (£) of backlog.

Assume the value of points earned is as follows:
Period 1, 20 points delivered, 20/260 = 7.692% complete. Earned value = £7,692
Period 2, 40 points delivered, 60/260 = 23.076% complete. Earned value = £23,076
Period 3, 30 points delivered, 90/260 = 34.615% complete. Earned value = £34,615

Project management is calculated using apportioned effort, i.e. based on the performance of delivering the backlog. Hardware and software purchase, earned as normal. These don't appear in the backlog.

Agile enables change after each sprint – how does that work?

Yes, Agile enables change therefore the backlog is not static, unlike the example above. Taking this to an extreme, if 740 points were added to the backlog in period 2, then the actual %complete would be 60/1000, which is now 6% not 7.6% as in period 1. In a traditional EV project, this would be seen as scope change, however unless the end point as changed, Agile recognises that flexibility is required to get to the same end point.

When the backlog changes, we cannot simply un-earn the value, or reduce the % complete. The solution is similar to change control in EVM, we simply re-baseline at the end of each month (or sprint). The difference here is we are not resetting the EV and planned value (PV) to equal the AC. The previous value earned remains the same and future months are based on the total number of points now available. In reality such large adjustments to the backlog should not occur, as your backlog should be around 80% defined when initiating the project. OK, this is not pure agile but we are trying to fit this into an existing gated management system based on the waterfall method. The point here is we are using Earned Value with Agile, when many people thought we could not.

So, in order to calculate the Earned Value on a frequently changing backlog, the following is needed:

- Points in backlog
- Points added
- Points earned in month
- Earned points to date
- Available points
- Percent of backlog completed this month
- Percent remaining
- Progress this month
- Total % complete

The following calculation can be setup in a simple spread sheet.

Points in backlog is taken directly from the backlog, including completed work and remaining work.

Points added is equal to points in backlog this month less points in backlog last month. Alternatively you can track points added to the backlog each month. This is a result of grooming the backlog, i.e. the effort required to deliver each story is re-estimated based on the latest information. It should not be a result of adding features, for example the user requirements changing, the end point should still be the same.

Points earned in month. This is the total value of the story points earned this month. The rules of credit are basic 0/100, i.e. 0% if the story is not accepted, 100% if it is accepted. There can be more than one story per sprint. So some value can be earned even if all stories are not accepted.

Earned points to date. This is the total of all story points earned to date, i.e. sum all previous points earned in month, including this month (Cumulative EV)

Available points (remaining work). Points in backlog now less all previously earned points, not including this month. We don't subtract the points earned this month because they were still available at the start of the period. What we are trying to calculate is the points we started with, plus or minus any changes to the backlog, so when we calculate the % of backlog complete it includes any adjustments made to the backlog during the month. Not doing this will introduce a lag, which will cause problems.

Percent of backlog delivered this month. This is equal to points earned in month divided by available points

Percent remaining. Percent remaining last month less percent of backlog delivered this month. This is calculating how much work is remaining as a % of the total backlog.

Progress this month. Percentage of backlog completed this month multiplied by percent remaining. This is calculating the actual progress. It is weighting the calculated progress based on how much is to go. In the final period, the percent of backlog delivered this month is always 100%, because you have delivered all of what was left. Here we are working out what this month's progress means for the entire project. For example if you had delivered 75% of the project, then you complete the project, the last sprint actually represent 25% of the project, even though it delivered 100% of what was in the backlog.

Total % complete. Total % complete last month plus progress this month. This is the cumulative % complete.

Example using a stable backlog – we are assuming here the sprints are 2 weeks, and the periods are months. We want to report progress each month. If a sprint is still in progress at period then it has not delivered any points, therefore is not included.

	Backlog						
Period 1	20						240
Period 2	20	40					200
Period 3	20	40	30				170
Period 4	20	40	30	10			160
Period 5	20	40	30	10	40		120
Period 6	20	40	30	10	40	40	80
Period 7	20	40	30	10	40	40	80

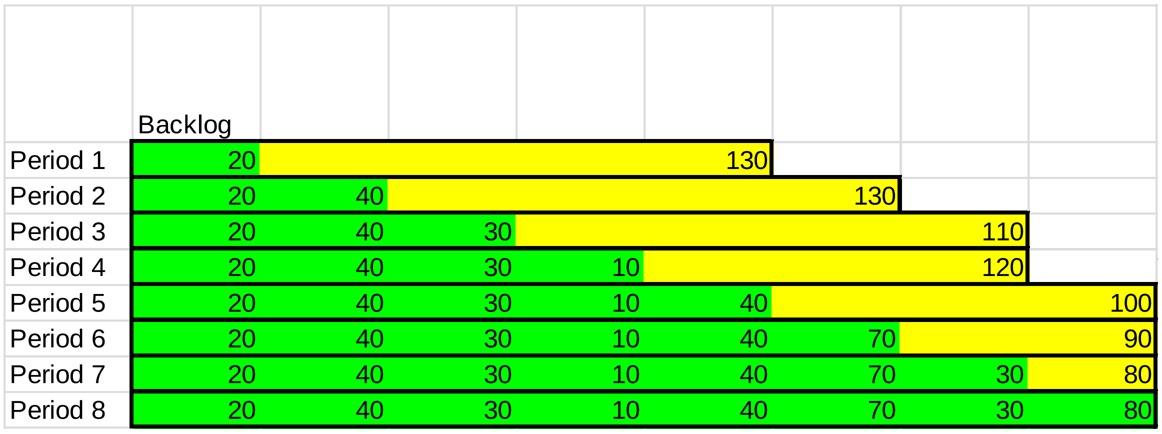
Calculations of total % complete

Points in backlog	Added	Points earned in month	Earned points to date	Available points	% of backlog complete this month	Percent remaining	Progress this month (% complete)	Total % complete
260	0	20	20	260	8%	100%	8%	8%
260	0	40	60	240	17%	92%	15%	23%
260	0	30	90	200	15%	77%	12%	35%
260	0	10	100	170	6%	65%	4%	38%
260	0	40	140	160	25%	62%	15%	54%
260	0	40	180	120	33%	46%	15%	69%
260	0	80	260	80	100%	31%	31%	100%

To get earned value for the backlog, multiply by total BCWS for Backlog (£100,000).

Note: in Period 1, the available points is 260, this is because we have not calculated the %complete yet, after we have used this to calculate the % of backlog complete, we can reduce the available points by the points earned to date. It is actually the points available at the start of the period plus or minus any adjustment to backlog.

Example of backlog Increasing



	Backlog							
Period 1	20				130			
Period 2	20	40				130		
Period 3	20	40	30				110	
Period 4	20	40	30	10			120	
Period 5	20	40	30	10	40			100
Period 6	20	40	30	10	40	70		90
Period 7	20	40	30	10	40	70	30	80
Period 8	20	40	30	10	40	70	30	80

Calculations of total % complete

Points in backlog	Added	Points earned in month	Earned points to date	Available points	% of backlog complete this month	Percent remaining	Progress this month (% complete)	Total % complete
150	0	20	20	150	13%	100%	13%	13%
190	40	40	60	170	24%	87%	20%	34%
200	10	30	90	140	21%	66%	14%	48%
220	20	10	100	130	8%	52%	4%	52%
240	20	40	140	140	29%	48%	14%	66%
300	60	70	210	160	44%	34%	15%	81%
320	20	30	240	110	27%	19%	5%	86%
320	0	80	320	80	100%	14%	14%	100%

To get earned value for the backlog, multiply by Total BCWS for backlog.

Note: in Period 2 you can see the available points includes any in-period backlog adjustments, so there was 150 in period 1, then 40 were added giving 190, but we had delivered 20 of these to date leaving 170 remaining. We have earned 60 of the 170 points, i.e. we have completed 24% of the remaining backlog this month. As we only have 87% of the project remaining having delivered 13% last month, the 24% actually represents 20% of the project. This gives a total of 34% complete. We multiply this by the baseline estimate of the backlog (£100,000) to give our actual earned value.

Example of a decreasing backlog

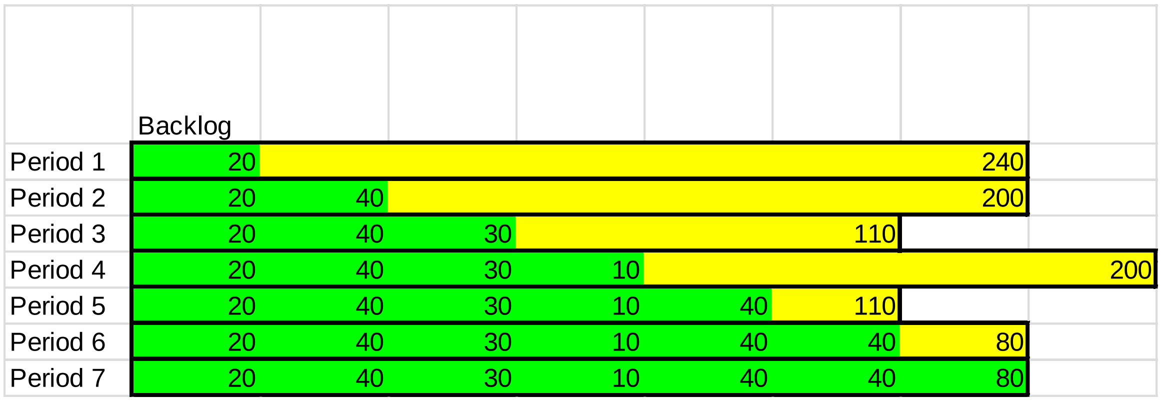
	Backlog							
Period 1	20							300
Period 2	20	40					190	
Period 3	20	40	30			140		
Period 4	20	40	30	10		120		
Period 5	20	40	30	10	40	70		
Period 6	20	40	30	10	40	70		

Calculations of total % complete

Points in backlog	Added	Points earned in month	Earned points to date	Available points	% of backlog complete this month	Percent remaining	Progress this month (% complete)	Total % complete
320	0	20	20	320	6%	100%	6%	6%
250	-70	40	60	230	17%	94%	16%	23%
230	-20	30	90	170	18%	77%	14%	36%
220	-10	10	100	130	8%	64%	5%	41%
210	-10	40	140	110	36%	59%	21%	63%
210	0	70	210	70	100%	37%	37%	100%

To get earned value for the backlog, multiply by Total BCWS for backlog.

Example of when backlog varies



	Backlog							
Period 1	20						240	
Period 2	20	40					200	
Period 3	20	40	30			110		
Period 4	20	40	30	10				200
Period 5	20	40	30	10	40	110		
Period 6	20	40	30	10	40	40	80	
Period 7	20	40	30	10	40	40	80	

Calculations of total % complete

Points in backlog	Added	Points earned in month	Earned points to date	Available points	% of backlog complete this month	Percent remaining	Progress this month (% complete)	Total % complete
260	0	20	20	260	8%	100%	8%	8%
260	0	40	60	240	17%	92%	15%	23%
200	-60	30	90	140	21%	77%	16%	40%
300	100	10	100	210	5%	60%	3%	42%
250	-50	40	140	150	27%	58%	15%	58%
260	10	40	180	120	33%	42%	14%	72%
260	0	80	260	80	100%	28%	28%	100%

To get earned value for the backlog, multiply by Total BCWS for backlog.

In all these examples it can be noted that the number of points delivered each sprint can vary quite noticeably, this may not be entirely representative of a real life situation, but could happen.

Forecast to completion.

In Agile we can use the velocity to forecast to completion.

After a number of sprints it will be possible to calculate the teams velocity, i.e. how many points per sprint do they normally deliver. It is a real measure of performance, and as it is the teams job to estimate the points value of each story, it allows for their ability to estimate.

Velocity = Average number of point earned each sprint.

If you know how many points are remaining in the backlog you can calculate the number of sprints required. Each sprint is a fixed duration of time, therefore you can estimate the remaining time, and cost. This is of course assuming the backlog remains stable from this point on, and the performance is constant throughout. This method, like traditional EV, it is based on the best available information.

For example, if there are 400 points in the backlog and the team has a velocity of 50 points per sprint, then you can calculate there are 8 sprints to go. If a sprint is two weeks, then you can estimate the completion date as being 4 months from now, as long as the backlog does not change and the velocity of the team remains constant.

Calculating the estimate to complete (ETC) and estimate at completion (EAC).

In the worked example above, the method described to forecast the completion uses an average velocity function. This is a reasonable method for determining the estimated to completion however from the worked examples and empirical evidence this method appears to give a pessimistic result depending on the method used to calculate the velocity, i.e. average points per sprint.

In practice early sprints usually deliver a lower number of points than later sprints as the project team move up the learning curve, plus the Truckman effect of forming, storming, etc. Without correction these early sprints will influence the calculated velocity for the duration of the project, plus any short dips in output have a similar longevity on the velocity. The table below is taken from the stable backlog example above.

Period	Points delivered per period	Average points per period	ETC	Forecast completion Period
1	20	20.0	12.0	13.0
2	40	30.0	6.7	8.5
3	30	30.0	5.7	8.7
4	10	25.0	6.4	10.4
5	40	28.0	4.3	9.3
6	40	30.0	2.7	8.7
7	80	37.1	0.0	7.0

It would not be a surprise to anyone practising EVM that period 1 is very pessimistic, and not withstanding the exceptional final period, the forecasts in period 2 and 3 are reasonable, however the effect of the apparently poor performance in period 4 takes time to recover.

This simple example raises a few questions.

- Using the remaining points to predict a completion time (and cost) may give an overly pessimistic prediction of completion.
- The rate of change in velocity (acceleration) does not appear to be taken into consideration in the forecasting process. This conundrum must be similar to predicting the lap time of a racing car travelling around a track, as the velocity varies throughout the circuit.
- Further more, velocity is not a performance measure, unlike EVM's scheduled performance index (SPI), it does not take into account the planned values, it only looks at the output achieve irrespective of what was planned, therefore is assuming a linear schedule, rather than the typical S curve.

This phenomenon may not come as a surprise to the Agile practitioner.

An alternative to the velocity method could be to use a simplified version of earned schedule. Agile due to its very nature does not have a planned baseline however in practice sprints are prepared ("groomed") therefore some forward planning is required to ready the team. This would enable a measure of performance against a plan.

Identifying failure early is a success.

Earned Value and Agile gives a more accurate measure of project performance allowing informed decisions to be made, and better forecasting of outturn costs and durations. The Waterfall method does not give the same level of detail, often based on level of effort, and with no certainty that the end product will meet the customers' expectations until the development is complete.

Predicting failure early is a success. In the case of the worked example above, if the project started to fail, this would become evident after only a few months, i.e. the ETC would increase. This would be a trigger for project management intervention and an investigation into the reasons for the variance, enabling the project to be put back on track. If a more traditional approach were used, then these failings could have been masked. Level of effort would have been earned regardless of the actual usable output from the team, and this would have resulted in the project failing after a significant amount of time and money had been expended, without delivering a working solution.

If you are interested in finding out more about Agile, Earned Value, Planning or Monitoring and Control, then the APM Planning Monitoring and Control Specific Interest Group is willing to help. The SIG can be contacted by emailing pmcsig@apm.org.uk .

Acknowledgements

Thanks to Alex Davis (PMC SIG) for his help in putting this article together.

Bibliography

Tamara Sulaiman, http://www.methodsandtools.com/archive/archive.php?id=6
Peter Schuh, Integrating Agile Development in the Real World
Craig Larman, Agile and Iterative Development: A Manager's Guide