IESE Business School
University of Navarra

IES653
January 2018

Agile Project Management

An Old "New" View on Project Management

It should be clear to anyone who has worked in project management in recent years that the agile methodology is not really a new concept, but it has been seen mostly as a methodology applicable to software development and related areas. It was not until recently that the leading project management organization the Project Management Institute published a document entitled Agile Practice Guide[1] to complement its traditional PMBOK Guide,[2] already in its sixth edition.[3]

Agile methodology was seen as an alternative to what was traditionally called waterfall methodology, even though different authors refer to it as "serial," "predictive" or "plan-driven." The agile methodology in the software industry was formalized with the publication of the Manifesto for Agile Software Development,[4] which states:

> "We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
>
> - 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."

[1] Agile Practice Guide, Project Management Institute, Inc., 2017. Note that this technical note incorporates some of the ideas of this document.

[2] A Guide to the Project Management Body of Knowledge, Project Management Institute, Inc., 2017.

[3] The first edition was published in 1996.

[4] Manifesto for Agile Software Development, available at agilemanifesto.org (accessed 12/12/17).

This technical note was prepared by Professor Jaume Ribera. January 2018.
All of the material contained in this document has been developed by the author unless otherwise stated.

Last edited: 1/29/18

Moreover, it develops the concept along 12 principles:

1. The highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from every couple of weeks to every couple of months, with a preference for the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity – the art of maximizing the amount of work not done – is essential.
11. The best architectures, requirements and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

In practice, you can find these principles applied through many different practices or approaches, such as Scrum, Crystal, XP, Kanban, etc., all of them considered to be the lean thinking concepts of manufacturing adapted for project management.

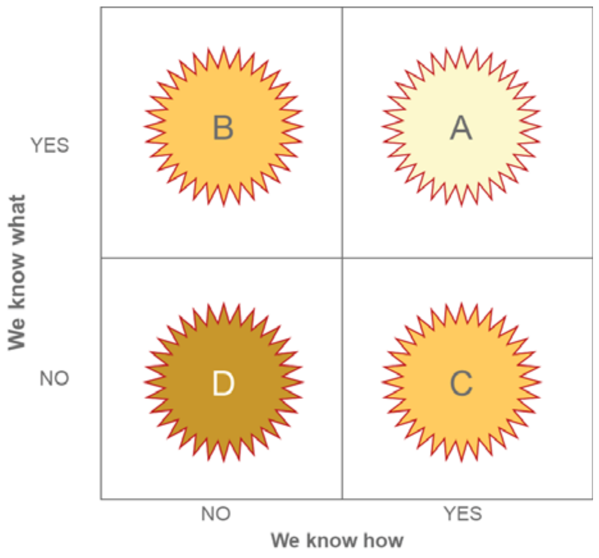
The characteristics of a project that define whether the traditional methodology or the agile one is more suitable are its level of uncertainty and complexity.

Two frameworks can help us understand these two characteristics. The first one comes from a book[5] by E. Obeng on strategy, which suggests classifying projects according to their uncertainty

[5] Obeng, E., Putting Strategy to Work, Financial Times Pitman, 1995.

on *what* they plan to achieve and on *how* they are going to do it. Using these two dimensions, we could classify projects in four quadrants:

Figure 1



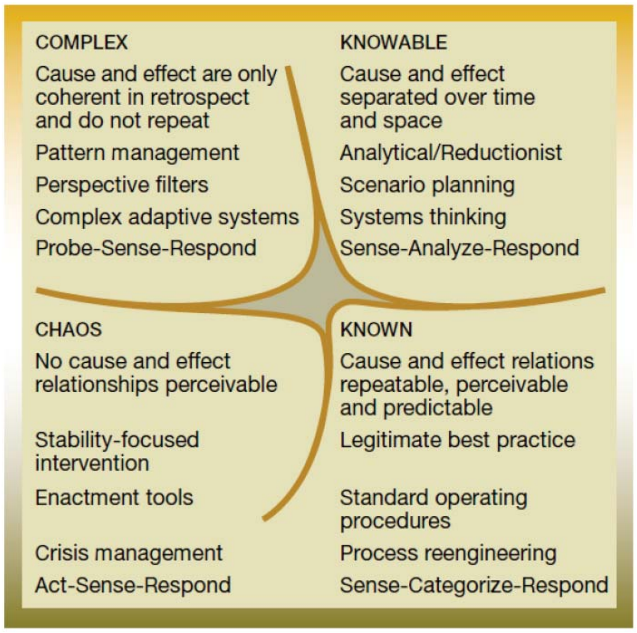
Projects in the A quadrant have a low level of uncertainty and can be planned with a lot of detail before starting their execution. This makes them suitable for a linear approach, with a lot of effort spent in the planning stage so that the execution would consist mostly of deploying the plan, not expecting major changes. Projects in the other three quadrants exhibit an important degree of uncertainty so will be much less stable and will involve a lot of learning during the project execution, which will require updating the plan quite often. For these projects, agile methodology will be more suitable.

Another interesting framework to describe the complexity of a project comes from the work of D.J. Snowden *et al.*,[6] which developed from the areas of knowledge management, cultural change and community dynamics to evolve towards organizational strategy by questioning assumptions such as order, rational choice and intentional capability. The central element of his approach is summarized in the Cynefin framework.[7]

[6] C.F. Kurtz and D.J. Snowden, The new dynamics of strategy: Sense-making in a complex and complicated world, IBM Systems Journal, 2003 & D.J. Snowden and M.E. Boone, A leader's framework for decision making, HBR, 2007.

[7] As described in Kurtz and Snowden's paper (cited above), Cynefin is a Welsh word whose literal translation into English as "habitat" or "place" fails to do it justice; it is more properly understood as the place of our multiple affiliations. We may not know all of these affiliations, but they profoundly influence what we are.

Figure 2



In later publications, he labeled the Known quadrant as “Simple,” the Knowable as “Complicated” and labeled an area not defined (in the center of this diagram) as “Disorder.” The Known/Simple projects are suitable to be managed in a traditional, serial, waterfall approach, while the Knowable/Complicated and the Complex ones are more suitable to be managed by agile methods. The Chaos domain usually requires a rapid response to reestablish order before anything can be managed. In this situation, what is needed is for someone to take charge and act quickly. Finally, a project may fall into the Disorder domain if you cannot make it fit in any of the other domains, which means that you do not know how to make sense of your situation. A usual way out of this is to split the project into smaller subprojects and assign each one of them to the other four domains.

For the high uncertainty and complex/complicated situations, the agile approach works better as it allows project teams to select short time cycles so that projects are executed via small increments of work. After each cycle, the teams can check the work done (and possibly show it to the customer and other stakeholders) and are therefore in a better position to adapt the plans more quickly and accurately to what has been learned, before the next cycle starts. In general, the agile approach emphasizes short feedback loops, frequent adaptation, reprioritization of features, regularly updated plans and frequent delivery of (preliminary) results to be able to obtain relevant customer feedback. In this way, the team can explore uncertainty and complexity at a low cost, in a short time and therefore reduce risk and maximize business value delivery.

The Agile Practice Guide offers a more refined classification of the project life cycles, defined as follows:

- Predictive, the traditional approach where most of the planning can be done upfront.
- Iterative, an approach that allows feedback from a cycle to improve or modify the coming cycles.
- Incremental, when the project cycles provide finished deliverables that the customer can start using.
- Agile, when the approach is both iterative and incremental.

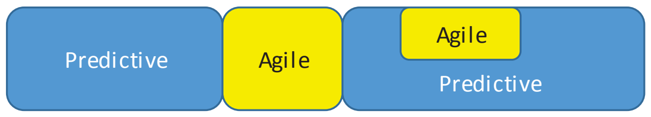
For the purposes of this note, we will not distinguish between the last three approaches, as all of them use similar concepts.

It is interesting to note that both waterfall and agile approaches require some degree of planning. The difference is that in the waterfall approach most of the planning is done upfront, the plan drives the posterior execution and most of the value is delivered when the project is completed. In the agile approach, the iterations and/or increments are also planned but the outputs of one cycle allow the replanning of subsequent cycles, in a way that value is delivered earlier and incrementally. It may even be possible that after a cycle results in a completely unexpected outcome, the team has to decide whether to redefine the project and/or cancel it.

The most common application of agile project management is through Scrum. This approach is based on time-boxes (cycles) of the same duration named sprints (and based on the duration, the team selects what tasks are to be completed within the time-box). However, it is also possible to apply agile principles with varying size time-boxes, making their duration dependent on the activities that are expected to be completed. PMI calls this approach “flow-based.”

It is also quite common to encounter projects that combine both approaches; that is, where part of the project is quite predictable and therefore can be planned in detail up-front, while other parts are uncertain or complex and require agile techniques. When we face these hybrid projects, the predictive part is planned in detail from the beginning while the agile part is only planned at a low level of detail, specifying the iterations that are planned and estimating the total duration of the activities. The detailed plan of each next agile iteration (cycle) will then be made when the project reaches that stage and will be executed following an agile approach.

Figure 3
A hybrid project



The Agile Project Team

Companies that have transitioned from waterfall project management to agile project management describe the difficulties encountered. Major differences between waterfall and agile mindsets have often been discussed as the major challenges that organizations will run into when transitioning. This is true in companies where there is a strong legacy of working with waterfall methodologies, where the processes and the initial specifications are considered more important than adaptation to customer needs and discoveries. For the employees of these companies, the process of unlearning waterfall methodologies is more costly that the process of adopting an agile approach.

It helps to train the staff in the new tools (to be presented in the next section), define the new roles, eliminate multitasking, create autonomous teams when possible, automate the artifacts, foster collaboration among team members, reward flexibility and to always concentrate on the end product.

The role of the project manager is also different. Instead of a control role, ensuring that the planned activities are initiated and completed as planned, the project manager must adopt a servant attitude. This means always keeping clear the *why* above the *what* and the *how* of the project, coaching the team members and protecting them from outside interferences; in summary, facilitating teamwork and removing organizational impediments to get the job done. The emphasis is on completing cycles, not on starting activities. The following table, adapted from the PMI document[8] can be helpful in understanding and fostering the required attributes of agile teams.

Table 1

Attribute	Comment
Dedicated people	Increased focus, small teams of fewer than 10 people.
Cross-functional team members	Able to deliver value often, being able to integrate all the work activities to deliver finished work.
Collocation or ability to manage any location challenges	Improved team dynamics, better communication and knowledge sharing, ability to help each other.
Mixed teams of generalists and specialists	Specialists provide expertise and generalists provide flexibility.
Stable work environment	Trust is required as there is dependence on each other to deliver. Preservation and expansion of tacit knowledge.

[8] Agile Practice Guide, Project Management Institute, Inc., 2017, page 40.

In agile project management, there are roles that are slightly different from the roles for traditional teams. These are:

- **Cross-functional team members**, in order to ensure that within the team we have all the skills necessary to produce a working product. Remember that in some agile projects, the team must deliver a working (prototype) product at the end of each cycle. This can only be achieved if the team has all the necessary resources inside and does not depend on external resources.

- **The product owner**, who is the person responsible for guiding the direction of the project toward the desired product. To do so, the product owner must continuously coordinate with customers, stakeholders, etc., to incorporate their views and, if necessary, realign the project.

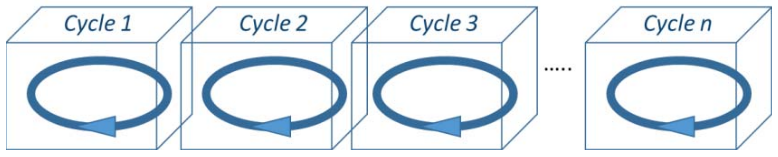
- **The team facilitator**, who is the servant leader we discussed above. We can still call this person the project manager, even though sometimes he/she is called Scrum master, project leader or team facilitator.

Agile project teams require a space in which they can work together. This is much more that the project war room that most waterfall projects have, where we maintain the documents and charts to control the progress of the project. Since the team must work together, a suitable space must be provided. Of course, when the team happens to be geographically dispersed, teams will have to combine shared virtual spaces with separate rooms and use document sharing, frequent videoconferencing and, from time to time, hold a physical meeting with all the members in the same location. This is essential at celebration time, since drinks are not transmitted well over the Internet.

The Agile Project Common Practices[9]

In this section, we describe the most common practices associated with agile project management. Remember that agile projects are planned and executed in cycles (time-boxes), either of the same duration (as in Scrum) or of variable duration (flow-based). Therefore, many of the practices referred to here are related to these cycles.

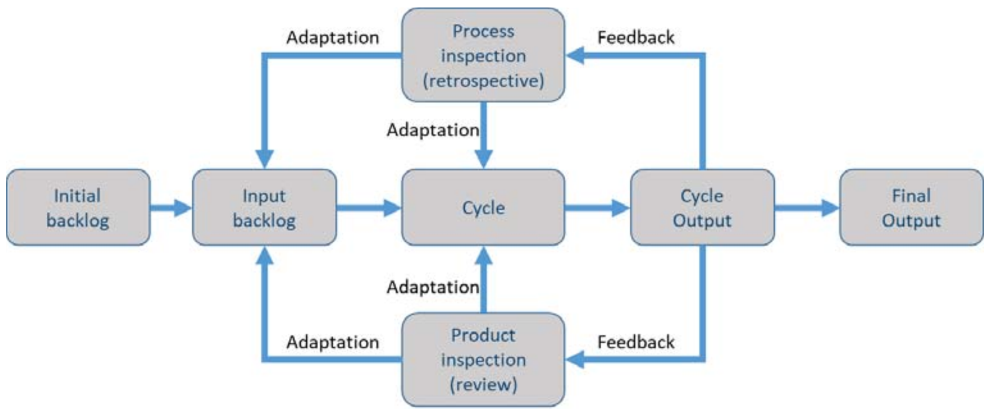
Figure 4
Agile cycles



[9] Part of this section has been inspired by the work of Kenneth S. Rubin, Essential Scrum: A practical guide to the most popular agile process, Addison Wesley, 2013.

The following diagram illustrates the agile process model, centered on a cycle:

Figure 5
The agile process model



Cycle

In the agile model, work is performed in iterations or cycles that are time-boxed with a defined beginning and end point. In Scrum these cycles are called sprints and have a duration of up to one month. One cycle follows another cycle and it is not permitted to alter the goal or the features to be completed within a cycle, even though sometimes this constraint cannot be maintained. The short duration of the cycles creates some interesting benefits to the project and the team:

- Short cycles are easier to plan and to estimate.
- They generate fast feedback, allowing the project plan to be modified and/or unnecessary or unwanted features to be pruned.
- Since most cycles are expected to already deliver working products, the return on investment is accelerated.
- Similarly, errors are also limited by what can be done in a short cycle.
- Short cycles provide frequent checkpoints, where different teams can put their parts together to test integration. You may consider the end of each cycle as a milestone, in the traditional project management terminology.
- Finally, short duration cycles also renovate excitement, similar to the energy generated at the beginning of each project.

Backlog Preparation

The product owner collaborates with the team facilitator, the team members and other stakeholders in defining the project backlog items and the project components (features, requirements, user stories,[10] etc.) that are necessary to achieve the project purpose. He/she ensures that they are placed in the correct sequence (logical dependencies, value, cost, knowledge, etc.). The backlog is in constant evolution, as items can be added, deleted or revised by the product owner as conditions change or as a better understanding of the project is developed. The effort and resources required for each item in the backlog is also estimated.

It is possible to have some backlog refinement meetings in the middle of a cycle to refine the stories or tasks in the backlog or how they are related to each other.

Cycle Planning

The backlog may represent much more work than what can be achieved in a single cycle. To determine the most important subset of the backlog that may be built on the next cycle, the product owner and the team perform the cycle planning and determine the goals to achieve in the next cycle. If a backlog item cannot be completed within one cycle, it is broken down into smaller tasks and some of the tasks are then assigned to the current cycle while the rest will be assigned to future cycles.

Daily Standup

Each day of the cycle, usually at the same time, the team members and the team facilitator hold a short meeting (less than 15 minutes) to inspect what was done and adapt if necessary. This short meeting is called a standup because, to promote brevity, everybody remains standing up during the meeting. This meeting is used to maintain the commitment to each other, to uncover problems and to ensure that progress is achieved. The following questions are answered by each team member:

- What did I accomplish since the last standup?
- What do I plan to accomplish by the next standup?
- What are the obstacles, impediments or risks that are preventing me from making more progress?

If the cycles have different durations, we may ask ourselves as a team:

- What do we need to advance in this cycle?
- What do we need to do next as a team?
- Are there any bottlenecks or blockers in the flow of work?

[10] User stories are a convenient way to describe the desired business value for the product backlog. Usually, a user story takes the following syntax: "As a <user role> I want to <purpose> so that <the benefit>." For instance, "As a professor teaching in the MBA program, I want to have access to the contents of different courses so that I can coordinate my teaching with what other faculty members teach." This provides an input for the team to include a piece of software, a course repository, etc. in the project to satisfy my needs.

Notice that standup meetings are not like status review meetings, since you do not want to review the situation of the project; you just want to reflect on what happened the previous day, what you plan to do on the current day and what others can do to remove the impediments that may block your progress. The standup meetings are short, five to fifteen minutes, and therefore you do not want to solve problems as they become exposed, but just take note of them. If necessary, you can schedule a separate meeting to take care of the discovered issues.

Cycle Review/Demonstration

At the end of each cycle, we review what the project team has achieved and, if a product is being built, we demonstrate it to the product owner and the customer. The objective is to inspect what has been done and to adapt whatever may be necessary. An important characteristic of agile cycles is that they are expected to deliver a working product, something that the customers can evaluate and that we can get feedback from. What was learned in the cycle review is incorporated as changes or additions to the backlog.

Cycle Retrospectives

This is another activity that occurs at the end of each cycle, usually after the cycle review. While the cycle review focuses on the product (the what), the cycle retrospective provides an opportunity to reflect and adapt the process (the how), considering both objective data (measurements, cost, progress) and subjective feelings. The Product Owner, the Team Facilitator, and team members come together to discuss what is and what is not working in the cycles, and how to improve technical and behavioral practices. The emphasis should not be put on who to blame but on the shortcomings of the process, finding the root causes of the problems and developing countermeasures.

The Agile Project Estimation

Common questions project managers and team facilitators are asked are "When will this be done?" and "How much will it cost?" Agile project management approaches these questions by estimating the size of what is being done in the project and the velocity or rate at which the work can get done by the teams. The velocity is adjusted every time we obtain new information. This is a big difference with traditional project management where we try to estimate how long it will take to complete each activity at the planning stage, before beginning the execution, and the estimates are rarely updated.

To estimate the size of what the project aims to achieve, first this project is decomposed in activities, tasks, user stories, etc., and each one of these chunks that compose the project backlog is estimated separately. It is important to note some issues related to estimation before we continue in this section:

- Estimations need to be done by the members of the team that will be responsible for the execution of the tasks. That is, the product owner is not involved in the estimation nor is the project sponsor. Different team members will see the tasks differently and will probably come up with a different estimate. We will later discuss how to incorporate these different views into a consensus estimate.

- Estimates are not commitments and should not be treated as such. In fact, we can consider two estimates for each task; the average estimate and the safe estimate. The average estimate is the estimate of how much it will take to perform the task, assuming that it can be repeated several times and the unlucky runs can be compensated with the lucky ones. The safe estimate assumes that there is only one chance to perform it, and you want to ensure that the estimate is achieved with a high probability. While the average estimate is the best to use for an individual task (as it balances good and bad luck runs), the safe estimate, which tends to be much larger than the average, is the one that will be used if the estimate is to be considered a commitment.

- We can achieve better estimates with better information, but there is a point of diminishing returns, so striving for over-precise estimates is useless. It is ridiculous to claim that a task will require 128.15 hours of work. Actually, sometimes, for the purpose of planning, it may be enough to estimate the size using rough, T-shirt-style sizes: XS, S, M, L, XL, XXL. It can be useful, as well, to make relative estimations, comparing tasks; for example, concluding that that task C is four times bigger that task A rather than estimating the absolute number of hours required for each of them. Also, planning poker, a technique described in **Exhibit 1** is a common way to size backlog items with more emphasis on achieving team consensus than precision.

Two approaches are used to estimate the size or effort required to complete each item in the backlog: ideal time and story points.

Ideal time (ideal days, ideal hours, etc.) represents the effort in days or person days needed to complete a given task or story. Notice that the adjective "ideal" is used to distinguish this time from the elapsed time. For instance, an NBA basketball game consists of four 12-minute quarters, for a total of 48 minutes. This would be the ideal time.

However, on top of regular playing time, there are also breaks between quarters, the time in each period is also paused during certain breaks in play, like fouls, injuries or time outs. Some teams often have mascots, cheerleaders or other entertainment on the court during scheduled breaks to entertain fans. The average NBA game takes over two hours to complete. This would correspond to the elapsed time.

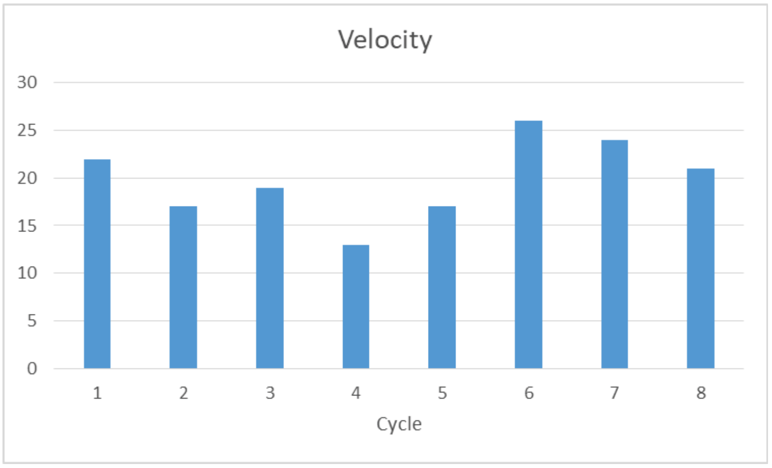
Estimating ideal time is quite straightforward, everyone understands what it means, but it is often later confused and used as calendar days, even though calendar days are elapsed time.

Story points do not directly measure the time required, but rather the magnitude of an item in the backlog. Story points combine size with complexity into one relative measure. For instance, a team may rank the items in the backlog list according to magnitude and use the easiest one as a unit of measurement (one story point), while the rest are assigned story points in comparison to the easiest one. For example, if creating the basic spreadsheet for the calculations is a 1, then finding the data to fill the cells may be a 4, implying that the second task is four times the size of the first one.

In both cases—ideal time and story points—we need a factor to transform the estimates into real days of work or, equivalently, how much work we can do in a cycle.

The concept we need is the *velocity*, defined as the amount of work that we can achieve in an agile cycle. Let us now focus on fixed time-boxes and the story points, which make the concepts easier to describe, even though a similar approach works for ideal time and for variable time-boxes. Velocity is estimated by adding up the size (story points) of the items completed during the cycle. We only include the items that have been completed; no points are considered for unfinished items. If we keep a record of the velocity achieved along several cycles, we can apply some statistical concepts to define the velocity of the team. Take, for instance, if we achieve the following velocity in the last eight cycles: 22, 17, 19, 13, 17, 26, 24, 21.

Figure 6
Calculating cycle velocity



In this case, we could use the average velocity for calculations (around 20 story points/cycle), or we could use an inter-percentile range (for example, eliminating the two highest and lowest outliers) to estimate the velocity to be between 17 and 22. In this case, if we have 240 story points to complete in the backlog, we can estimate that we can do it in between 11 and 14 cycles.

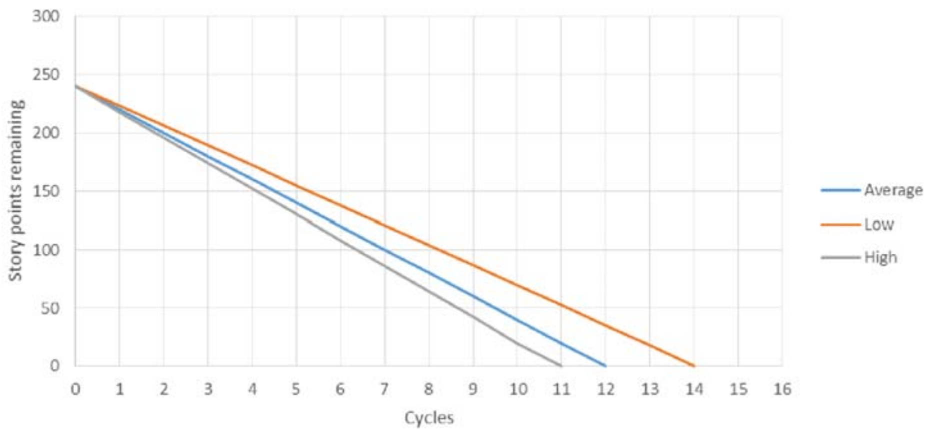
Estimates of velocity are used for several purposes. For team planning, as shown above, it allows us to estimate the number of cycles required to complete the items in the backlog, and it provides an objective by the team to do it. It is also used as a diagnostic tool to check how the agile methodology can be continuously improved to deliver customer value. By observing its velocity over time, the team can gain insight into which practices affect the delivery of customer value.

Measuring and Communicating Progress

An important aspect of any project management methodology is the chapter on communication. Sponsors, customers and other critical stakeholders want to be regularly informed of the progress of the project and to get updated estimates of when different parts of the project will be completed.

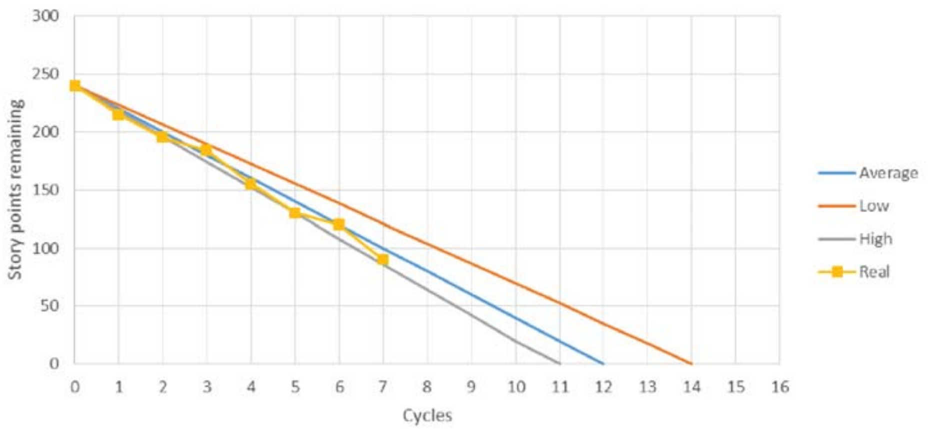
The most common tool to show progress in agile projects is the burndown chart, a diagram that shows the amount of unfinished work that is expected to remain at the end of each cycle. Using the example in the previous section, if we have 240 story points to complete, with the estimated velocity we can prepare a plan for expected point delivery as follows:

Figure 7
Burndown chart



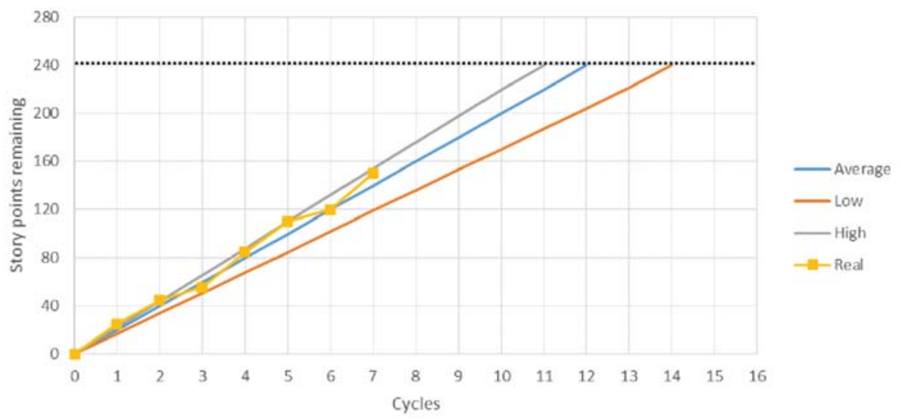
Later on, we can plot the real situation as we advance in executing the cycles:

Figure 8
Burndown chart



Some teams prefer to use a burnup chart, which is the equivalent upside-down chart showing the story points already achieved.

Figure 9
Burndown chart



For companies who are familiar with the earned value methodology, it is also possible to define the schedule performance index (SPI) as the ratio of completed story points over planned story points (using the average velocity). In this case, in cycle 7, the SPI will be approximately 150/140 = 107%. For the cost performance index, it would be necessary to keep track of the value of the stories already completed (earned value) and divide it by the cumulative actual costs. However, since most agile projects have dedicated teams and their salaries are the most important part of the costs, the cost variation would be very similar to the progress variation as described with the SPI.

The charts above are used to display the evolution of the agile project, considering the backlog list that supposedly includes all the items that have to be completed. As mentioned before, this list is very dynamic, with items being added, modified or deleted as the team and the product owner adjust the project. In this case, the top line in the previous burnup chart will not be a straight line, but rather it will adapt to reflect the latest estimates.

Similar charts can be used to track the progress within a cycle, but this is seldom used as cycles tend to be short enough not to require these tools to be monitored. In this situation we only check whether or not the team has accomplished all the items planned during the cycle preparation, and we record the team velocity in the cycle, for tracking and diagnostic purposes.

Exhibit 1
Planning Poker

Planning poker is a consensus-based technique for estimating effort. It is based on the assumption that the experts (the team members) will be able to achieve a consensus estimate after being exposed to the item characteristics and the assumptions, having had intense discussions with the product owner and the other team members to acquire a sharing understanding of the tasks at hand.

Because the goal is to be accurate but not overly precise,[11] the estimation scale used in planning poker is either the Fibonacci scale (0, 1, 2, 3, 5, 8, 13, 21, 34, ...), a modified scale, proposed by M. Cohn (1, 2, 3, 5, 8, 13, 20, 40, 100), or a scale based on powers of two (1, 2, 4, 8, 16, 32, 64, ...). Usually, this scale is transferred onto a deck of cards, similar to poker cards, hence the name of this estimation technique.

We can also include cards with the symbols 0, ½, ∞, ?, π or ☕. A zero indicates that the task being estimated is already completed, or unnecessary, and it does not make sense to estimate it; ½ indicates a very tiny item; ∞ indicates that the task is so large that it cannot be estimated, it needs to be broken down in smaller parts; ? indicates that the team member does not understand the item well and requires further details; finally, π or a coffee cup indicate that the team member is tired and needs a piece of pie (π) or a coffee break.

The rules for the planning poker estimation are as follows (whether they estimate ideal hours/days or story points):

1. The product owner selects an item from the backlog that needs to be estimated. He or she describes the item to the team.

2. The team members discuss the item and ask the product owner further clarifying questions until they understand the item well enough.

3. Each team member selects the card from the deck that best represents his/her estimates. This is done privately, without showing the chosen card to the rest of the team.

4. Once everybody is ready, all members simultaneously expose their chosen card to the rest of the team.

5. If everyone has selected the same card, a consensus is achieved and the consensus becomes the estimate.

[11] The accuracy is the degree of closeness of estimates or measurements of a quantity to that quantity's true value. The precision is related to reproducibility and repeatability; that is, the degree to which repeated measurements under unchanged conditions show the same results.

6. If the estimates are not the same, the team members engage in further discussion, usually by asking the high and low estimators to explain their rationale. After the discussion, they return to step 3 and repeat until a consensus is obtained. Either a good enough consensus is reached or a large majority of the members agree on a close range (with the exception of a few stubborn outliers who are then neglected).

It is important for the team members not to show their estimates or make any comment on which estimates they are favoring before step 4 to avoid the team anchoring effect or the Abilene paradox.[12]

[12] Anchoring is a cognitive bias that describes the common human tendency to rely too heavily on the first piece of information offered (the anchor) when making decisions. During decision-making, anchoring occurs when individuals use an initial piece of information to make subsequent judgments. Once an anchor is set, other judgments are made by adjusting away from that anchor, and there is a bias toward interpreting other information around the anchor. For example, the initial price offered for a used car sets the standard for the rest of the negotiations, so that prices lower than the initial price seem more reasonable even if they are still higher than what the car is really worth.
In the Abilene paradox, a group of people collectively decide on a course of action that is counter to the preferences of many or all of the individuals in the group. It involves a common breakdown of group communication in which each member mistakenly believes that their own preferences are counter to the group's and, therefore, does not raise objections.
(Wikipedia, accessed 13/12/2017.)