Agile Question Time
Presented by Barbara Roberts

Question Asked	Question Answered
Can you please explain the key difference between burn-up and burn-down chart in agile. There is a bit of misunderstanding in both concept	A **burndown** chart is a tool used by Agile teams to gather information about both the work they have completed and the work that is yet to be done. It displays as a graph, where the expected rate of completion is plotted, and then the current time to complete is plotted to show the difference between "expected" and "actual". Importantly "time to complete" is re-estimated at this point in time, rather than subtracting time spent from the original estimate. Typically, a project has a high-level burndown for the whole project (the big horizon), another one for the current Increment (release) (the intermediare horizon), and one for the current sprint/timebox. The bigger horizon burn-downs may use story points or time. But for the sprint/timebox (2-4 weeks, typically) we usually estimate for this using tasks and hours needed to complete a requirement (story), since at this point we know exactly what needs to be done so we can be much more detailed. If the actual burn down line is on or underneath the expected line, we are on track. As soon as it goes above the line, the team is immediately aware that there is an issue. And visually it is possible to see the trend of progress and extrapolate this to see how serious the issue might be, with the intention of taking action sooner, while there are more possible options. Within a sprint/timebox, the burndown should be updated at least once a day. The bigger horizon burn downs are usually updated on a less frequent basis. A **burn up** chart shows how much work has been completed, compared to the total amount of work expected . For both Burn-downs and Burn-Ups, it is important to choose what metric to use and when. For example burning down story points in a sprint results typically is a flat line followed by a drop - a graph that looks like a staircase. Since the team have access at this point to more details information, they should make use of it. For the bigger horizons, there will always be more uncertainty, so detailed task by task estimating on big picture items is impractical so some form a "top down" estimating is needed, such as story points, T shirts or man days - good enough at this point.

How do you develop a new sprint team, so they work smoothly together. Do you understand and document RACI and who has what skill and to what extent that skill is? That's my initial need but what do you do?	Part answered during session. Additional Info:- RACI can also be useful, one of the reasons why we included this as part of AgilePM. As we always work collaboratively in Agile, for me this could be part of a kick-off session at the start of a new piece of work, to ensure everyone is on the same page from the start. If using Scrum, for example, then I would still discuss RACI but also bring in and reference the wider stakeholders and where they fit as regards RACI. All too often, sprint teams (assuming a simple Scrum approach is being adopted) are not made aware of the wider stakeholders interests and concerns. However, if what you have is a project (rather than a team working on, for example, continuous improvement), then having a good understanding of the wider stakeholders is critical for success. This "team as an integral and collaborative part of the project" concept is embedded and emphasised as part of AgilePM. Using AgilePM as an example, part of my Project Kick off when coaching and mentoring is to facilitate the team (the whole team) filling in the names on the Roles diagram (often referred to as the "alien baby" so everyone knows who fits where - at the project level (shapers and directors for the upwards and outwards alignment, beyond the project), at the team level (the "engine room" actually building the solution), and the supporting roles (such as advisors on in-depth issues around specialist subject matters). I also often use simple techniques such as getting the team to assess themselves using Belbin, and then share the results with each other. The "soft" personality behaviours are equally important as the "who knows what, who is doing what"
How do you bridge the gap for information. Senior managers want traditional info but the team can not retrospectively produce that traditional view. Burn down charts just can not replace a good Microsoft schedule plan with milestones and completion %'s. How do I change that traditional mind set.	I completely agree that you don't just go to the exec with a burn down chart instead of what the are used to. Agile in the wider corporate context still needs to present a "big picture" plan, and this is exactly what AgilePM promotes. The exec want to know what to expect and when. However, in a corporate agile world, we present plans in terms of chunks of delivery, usually based around specific objectives, but we do not go into the lower detail in the overarching big plan. For example (based on a current agile transformation I am supporting leading), the big plan is for delivery of a 3 year strategic new product. The agile plan shows this will be achieved via 6 releases (releases will be every 6 months). Release 1 is a minimal viable product (MVP), not for public release, but to be shared with a very small group of "friendly" existing customers. Release 2 will be MVP+ (some additional features) which will be available publicly to a limited customer set. Release 3 will bring in some ground breaking new features, and will go fully live to existing and new customers. Releases 4-6 will each be adding new features, but which ones and when is still tbc. It is also possible that Release 6 may not go ahead if by that stage the value to be delivered does not justify keeping the development team assigned to the project. As you can see, this gives the exec a clear roadmap of what to expect when, and this is all presented in terms of value to be delivered, risks to be addressed / mitigated. In other words, using the language the exec expect. This plan is supported by transparency - the "go see for yourself" thinking as promoted by the Global Digital Strategy - burn downs, demos, podcasts etc. But these are all seen in the context of the overarching plan.

Traditional contracts tend to use a waterfall manna, and the data that contract refers to for acceptance is waterfall. How do I get Agile data shopped to such contracts for payment milestones?	Partly answered during Q&A session, but this is a huge topic. Additional information - Agile Procurement is a major issue, as the bulk of all contracts are defined in such a way as to make taking an agile approach harder than it needs to be. When the starting point for procurement is "We need to be clear when we sue the supplier" (and I quote exactly from a procurement group I was working with), then trying to "sell" collaboration, shared ownership and responsibility etc is hard. But it can be done, and the Agile Business Consortium has done quite a bit of work, working with solicitors, on creating a base for an agile contract. Mirko Kleiner is also doing some excellent work in this area. But in essence, where there is the will to make Agile work under formal contracts, it can be done (and I have successfully delivered a number of agile projects under formal fixed price contracts, with the understanding that the customer has the right to cancel at the end of each delivery stage. (And they never did!) It is also "interesting" to remember that the traditional contractual model has been shown to be ineffective in many circumstances, where apportioning blame is seen as more important that getting to a satisfactory solution
How would you set up funding gates? I really like that idea. This is great if we an a-line it to the contract/ sla.	Using AgilePM lifecycle as a base, I have a Go/No-go gate at the end of Feasibility and at the end of Foundations, where the team get the funding for Feasibility and on completion, have to justify why the project should be funded for the next step (Foundations) and at the end of Foundations, justify why they get funding for the 1st increment (Release), with the same process after each subsequent Increment (release). There is always more work that there are people to do it, and strategically it may make more sense to stop a project that is delivering minimal value, or which is less important than a critical piece of work that has just surfaced. We really need to think about use of resources and delivery of value differently, so we no longer view stopping a project early or putting it on hold for now as some sort of failure. It is actually good financial and corporate management IMHO (See Q6 for more information on the topic as regards contracts
In Agile, how the sponsor ensures whether the Business case is efficiently delivered. Because, basically with Agile, we manage (take out or keep it back) from the backlog (issue logs) rather than delivering the entire scopes into the given time?	In Agile PM we have 2 roles who are on top of this: the Business Sponsor "owns" the business case, in that they want to ensure the spend is justified in terms of delivery of value (which is not always £££). We also have the Business Visionary who is looking at the future position to ensure the day to day work is aligned to this objective. Part of that role will be making sure the work being prioritised (which can mean including or excluding things) aligns with the business case. In Agile the expectation is that you are highly unlikely to get 100% of features, because estimates are never 100% accurate. (If you consistent do get 100%, then it is usually because people are creating "comfortable" estimates!!). Setting realistic expectations is important, so for example (using AgilePM) we promise / guarantee delivery of the Must Haves. We expect to deliver the Musts and the Shoulds. ANd you may get some of the Could Haves. But this also relies on MoSCoW being balanced correctly (No more than 60% for Must Haves, at least 20% effort for Could Have (where Musts + Shoulds + Coulds = 100% effort)

Is it fair to say, agile is more effective for small projects?	No, not really. It is probably easier / more straightforward, especially when you are in the early stages of learning how to work and think in an agile way. But if you are doing bigger projects, then this is where you may find that for example Scrum alone is not enough, because it has no inbuilt concept of "Project - temporary, with a beginning, middle and end" . This is where AgilePM comes into its own, as it is designed to be scaleable to deliver large or very large or complex projects (as well as being capable of being scaled down to do small and simple as well) By comparison, Scrum is designed as a (brilliant but simple) team process, and it lacks the integration into the "bigger picture".
a) What could be the success criteria and b) Success Rate?	One of my success criteria for all my Agile pilots (and all subsequent agile projects) is that there will be zero functional errors delivered when the solution goes live. Provided we have engagement from our Business Visionary, Business Ambassador (this role equates to Scrum's Product Owner), and our Business Advisors (our SMEs), then these people ensure that what we build and deliver is the right thing and will enable people t do their job, whatever that job is). Another one I emphasis is that Deadlines are never slipped (because Functionality is our "negotiable factor" in the normal application of agile