Article

Estimation Toolkit

Posted by James King on Dec 28, 2010

Community Agile Topics Agile Techniques Tags Lean , Estimating

No matter what kind of software you write, no matter what size company you work for, you probably have to provide estimates to someone. They may want to know how long the effort will take or how much it will cost. But no matter how you slice it, estimation is critical skill and essential to the success of any software development effort of any size or consequence.

There are many techniques agile teams can use to help guide their estimation efforts. The toolkit described in this article consists of a number of approaches to estimating both projects and "business as usual" work. The techniques are heavily biased toward agile projects and are designed to be used by people who should already have a basic understanding of agile estimation techniques (i.e those who have completed an agile project management course and/or been involved agile projects long enough to understand how they work).

The first section of the toolkit explains some general tools that can help the team when using story points to estimate. These consist of:

- Some tips for calculating velocity at the beginning of a project; and
- Some tools that support planning poker by improving the understanding the team have of the complexity of different stories.

However, we often need to estimate projects before we have even started the analysis on them. So in the second section of the toolkit I have included an approach to doing this called the WAG Generator. The approach does not produce a detailed estimate but it does provide an accurate enough guess to compare projects or to decide whether to start a project.

All of the techniques discussed in this article are ones that have worked well for me, but I will leave you to decide whether they are applicable or useful in your own environment.

Agile Estimation Techniques

Agile methodologies, such as Scrum, generally recommend the use of relative estimates. Typically, the team uses planning poker to allocate story points to each story and then compare these to the team's velocity in order to plan the project.

Planning poker has been discussed in other places (including the Software Education "Agile Requirements - Stories" course) so we will not discuss it in detail here. However we will discuss:

1. Estimating velocity before the project starts; and
2. Tools to help the team better understand the stories being estimated.

Three Ways to Estimate Team Velocity

The single most accurate way to estimate velocity on a project is to run several iterations and then see what the average velocity was. This approach allows the team to find its rhythm and then use its experience in the real world to predict future iterations. However, those providing the funding for the project will often require an estimate before the project team can perform multiple iterations.

If the exact same team has been together for a while (perhaps on a similar project using the same tools and technologies) then they might provide an estimate based on the previous project's average velocity.

So, if an actual observed velocity is not available to use, the next best thing is to estimate a team velocity. There are three ways that commonly work well to estimate team velocity and they are described them in the next three sections.

1) Using traditional methods after the completion of planning poker

One of the great advantages of planning poker is that it flushes out the assumptions and constraints involved in the solution. This information provides good input for team to do the estimating, so rather than trying to convert story points to time and money, the estimators simply use that information as an input into their calculation of the duration of the project (for example the team could use the WAG generator described later in this document.

This may seem like duplicated effort, but it might provide the team with confidence in their first couple of agile projects.

Once the team has an estimate for the whole project then they can decide to receive the project, gain funding and then begin the project with the same confidence that they usually have when starting. Then the team will be able to run several iterations and adjust their estimates as they gain more information. Once this has been done then the team can determine and share their actual velocity.

2) Working backwards from the iteration length

The second approach to estimating a team velocity is to determine the iteration length and team size, and then decide how many stories the team could complete in an iteration. This is done as follows:

1. Decide on the iteration length, as well as the expected team size and composition.
2. Bring the team together and select some sample stories (these need not be the stories that will be done in the early iterations).
3. Allocate stories of varying sizes to the first iteration one at a time and let the team determine how many of them they think they could get done in the time available for this iteration:
    - This can be done by simply moving stories in and out of the iteration until the team is happy they have "about one iteration's worth of work" allocated; or
    - Stories can be broken down into tasks and those tasks can be added to the iteration. This involves estimating how many hours each task will take and simply adding them together until we consume the total team hours available in the iteration.
4. Once you have worked out how many stories fit into an iteration, then the total of the story points for each story will be the estimated velocity.

5. Repeat the process for a couple more iterations if you feel this will increase the accuracy of the estimate.

3) Working upwards from the time taken for one or more stories

The third way to estimate the team velocity is to work out how long it takes to deliver one story point and then use this number to work out how many story points can be completed in an iteration. This is done as follows:

1. Choose a typical story.
2. Determine all the tasks needed to complete the story and covert these into expected person hours.
    - Add up all the hours needed by each person in the team;
    - Use ideal time (effort) rather than elapsed time;
    - For example, if a story would take 1 hour of a BA's time, 2 hours of a developer's time and 1 hour of a technical writer's time then this would translate into 4 person hours.
3. Divide the number of points in the story by the number of person hours to find out how many points can be completed in one hour.
4. Multiply the figure you have calculated in step three by the number of hours in the iteration to determine how many points could (in an ideal world) be completed in an iteration.
5. Divide the figure you have calculated in step four by two (or similar conversion factor) in order to convert from ideal time to elapsed time. This is the velocity that is predicted by the one story.
6. Repeat the process for more stories and average the results. Estimating additional stories will add to the accuracy of the estimate, but generally after 5 stories you will have found a good estimate of the velocity -particularly if you chose a range of different stories and realistically broke them into tasks.

Four tools to assist with estimating stories

1) Things That Matter Matrix

The "Things That Matter" (TTM) matrix is a simple tool for exploring the complexity of a story. A TTM matrix consists of a list of the technologies that are needed to deliver a series of stories. The technologies involved are listed on the left of the matrix and stories are listed on the top, as shown in the table below:

		Stories				
		1	2	3	4	5
Technologies involved	HTML	X				
	Silverlight	X		X	X	X
	SQL	X			X	X
	XML		X	X	X	X
	Firewall integration		X			X
	Java				X	X
	CMS			X	X	

The aim of the TTM is to allow the group to understand the complexity of story by seeing which stories involve which types of technology. The theory here is that a story crossing many technologies will be more complex and thus have a larger estimate than stories only involving one or two types of technology.

To expand on this idea a little further, the effort involved in developing some stories may depend on more than just the technologies - it may also depend on various processes as well. So rather than just listing technologies we may choose to list any significant "things that matter" such as:

- Complex testing (scalability, usability, security etc);
- User research; or
- Legal sign-off.

Rather than simply listing whether a technology or process is involved, we may list whether the impact is simple or complex. So, for example we could replace the "x" in the boxes above with the following codes to provide more detailed information to the estimator:

- H = High complexity or effort;
- M = Medium complexity or effort; and
- L = Low complexity or effort.

This would give us a slightly more complex TTM as follows:

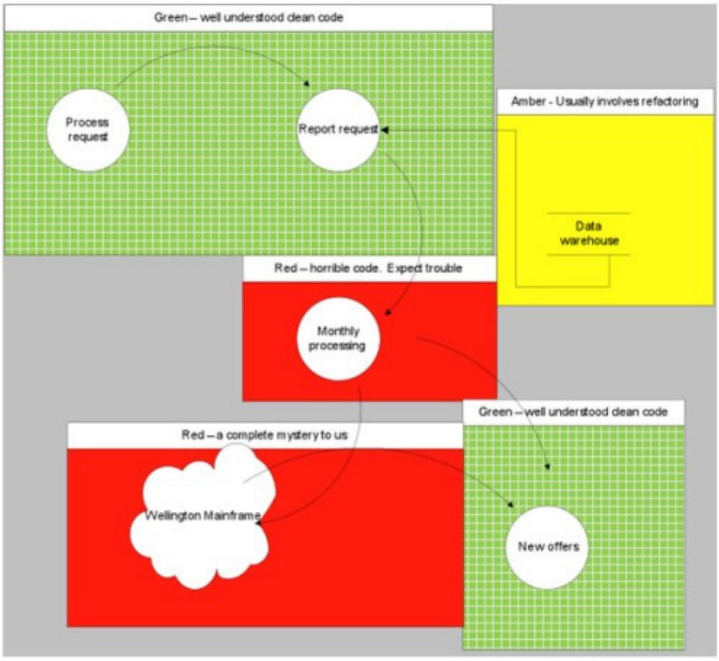
		Stories				
		1	2	3	4	5
Technologies involved	HTML	L	H		H	L
	Silverlight		H	M	M	
	XML	M		M	M	L
	Wireframes	M		M		
	Java		H			
	CMS		H			
	User research		H	M		
	Integration testing		M		M	
	Regression testing		M		H	
	SEO changes					
	Changes to contract				H	
	3rd party involvement		H		H	
	Training for sales team		H	M		

2) System Heat Map

A heat map shows where to expect trouble in a system and where to expect smooth sailing.

To create a heat map, simply take any model of the system and shade or colour different areas of the model to indicate the ease or complexity of working on different each area. Things that can be taken into account include whether there are automated tests in place, areas of spaghetti code, areas that involve complex integration code, areas the developers know well or anything else the team feels is relevant to the complexity (and therefore estimated effort) involved in working on different areas of the system.

With a heat map created (as shown below), the team can adjust their estimates involving different areas of the system so that they are in line with the "heat" of the particular area.



3) Lo-Fi Prototypes

A Lo-fi prototype is simply a picture or a simple model of the system. These can be used to create new stories or to help the team provide estimates.

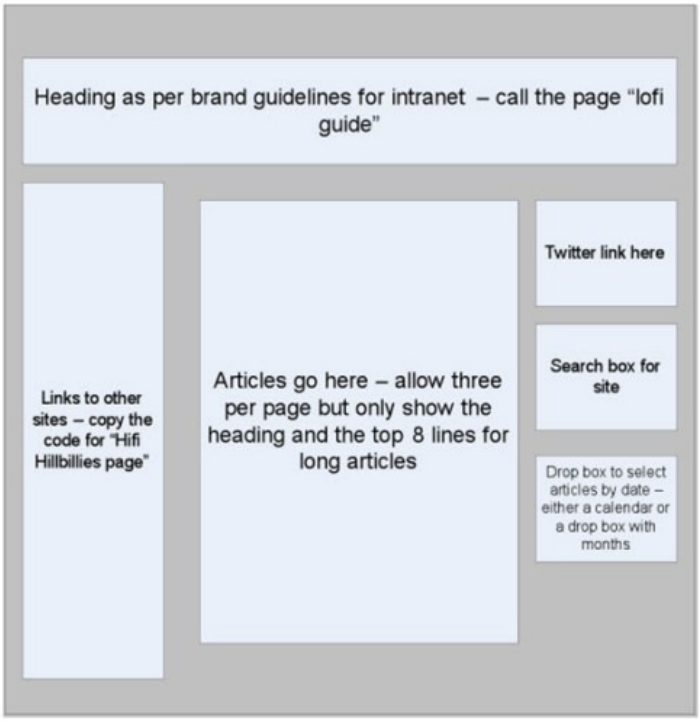
The term "Lo-fi" or "Low Fidelity" means a non-working model while a "High Fidelity" model is a working model. However, in my projects we use the term "Lo-fi" to simply mean any picture that does not comply with UML, Wireframe standards or any other formal rules.

Drawing a rough sketch of a screen or system might seem simplistic but it is amazing how many differing assumptions get unearthed when you try to draw a simple picture as a team. So whenever I am dealing with a complex estimate I always get one of the team members to draw either:

- What the screen will look like to the user; <
- How the process works from a user perspective; or
- How the system hangs together.

This can often be as simple as doing a print screen of an existing system and then adding some comments to reflect where the changes or updates will impact the screen.

However, you can also create any kind of sketch that will help people to understand the features being worked on. For example you might draw a process flow and attach the stories to the appropriate section of the process to give some understanding of how the stories all link together. Or as shown below, you might simply draw the rough boxes on a page to show where you want to display different information, include drop boxes and so forth.

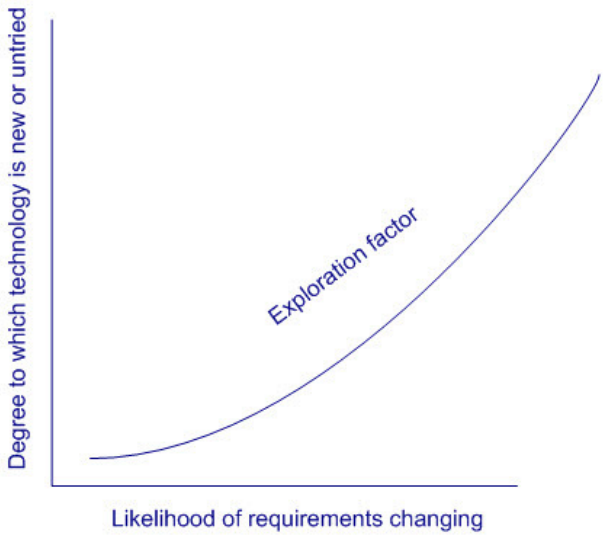


4) A Measure of Uncertainty

It is easier to estimate work that we are familiar with than things we have no experience with. While this may seem obvious, it is something that is often overlooked.

Jim Highsmith recommended looking at the "Exploration Factor" of a project in his book "Agile Project Management". Essentially he said that projects are more complicated if either we are likely to change our requirements a lot as we progress or if we are working with "bleeding edge" technology rather than well understood and tested technology.

Thus, if a project includes all bleeding edge technology and is likely to involve changing requirements as the team better understands the problem space, then the project has a high Exploration factor.



Some teams apply this approach to stories. In fact the "Heat Map" and the "Things That Matter" techniques are approaches to understanding the Exploration Factor.

Often teams take the simple approach of ranking stories from 1 to 5 to reflect increasing levels of uncertainty. Another approach is to place stories on the graph above to simply mark where they sit relative to each other.

When I have used this technique, I have often included the information in my estimates. However, other teams using this technique have recorded both the effort

estimate (in story points or hours) and the complexity estimate. Either approach seems to work well.

For more complex projects I sometimes go a little further by discussing the Exploration Factor in terms of a number of categories. Then, similar to a Heat Map, the team can assume that the stories involving high Exploration Factors in any category are larger than those stories that do not.

For example, I might use one or more of the following categories:

- Customer Exploration Factor - How new is the product to the market:
    1. This is our core market.
    2. This is a new category in our existing market or a market segment we are not strong in.
    3. This is a market we are in but not with this product or service.
    4. This is a market we have never been in.
- Product Exploration Factor:
    1. We are enhancing our existing products or services.
    2. This is similar to our existing products or services.
    3. This is a new product for us but our competitors provide it.
    4. This is a new product in our market or region.
    5. This is a product or service that does not yet exist in the world.
- Internal adoption Exploration Factor - how hard will it be to convince people to adopt the feature:
    - How well does the new feature align to the existing way the team work (1-5).
    - How easy is it for the team to customise the new feature (1-5).
    - How much say does the team have in how we design and implement the feature (1-5). <
    - Add these numbers together and rank the likelihood of adoption.

In some cases a discussion of the Exploration Factor of the project or some of its components can substantially improve the understanding of the team and therefore make estimating more accurate. It might also help the stakeholders and the team to understand the risks inherent in the project they are embarking on.

Estimating with insufficient information

The WAG generator

The WAG generator (also known as "wide angle guess" or "wild a***d guess" tool) is not in common practice. In other words, it works well for me but there is no research or alternative sources to confirm it works for others. Therefore you should consider the idea on its merits and use it if you see value in it.

The purpose of the WAG generator is to provide the estimator with a rough and ready estimate that may be accurate enough to make decisions in the absence of time consuming analysis. I have often found this to be necessary at the beginning of projects or working out which enhancements deserve further investigation and which should be abandoned straight away.

The estimate is provided in either days or dollars and is done by selecting a range from a table rather than allocating a specific value to the task.

Creating the generator

The WAG generator is simply a table containing letters that represent a range of values.

- "A" is generally taken to represent a number big enough for the customer to refuse to commit to the work without a more detailed estimate, and thus the funding of some further analysis;
- "B" is allocated to items that are about half as much as "A";
- "C", "D" and the following letters are then allocated to items that are successively smaller; and
- Once items reach a trivial level then no more letters are added.

Using the generator

Once the WAG table is created then new items are estimated using the table rather than being analysed in detail.

Based on the WAG estimate, the customer can:

- Decide not to proceed;
- Commit to proceed;
- Prioritise between multiple items without needing further analysis; or
- Commit to further analysis if it appears justified.

This allows the estimator and the customer to quickly sift through a large number of items without allocating substantial effort to any of them.

Enhancement WAG

The first version of the WAG generator is used to provide a quick and dirty estimate for small "enhancement" requests made to a "business as usual" team.

Rather than sitting down and analysing requests, the estimator allocates the item to wide range of dates (or costs). This estimate may provide enough accuracy for an initial prioritisation session, after which the team may prepare a more accurate estimate if needed.

Rating	Estimated time		Rating	Estimated time		Rating	Estimated time
A	Over 6 months		D	2 -4 weeks		G	1 -8 hours
B	3 -6 months		E	2 -10 days		H	¼ - 1 hour
C	4 -12 weeks		F	1 -2 days		I	Less than ¼ hour

Note that instead of time, it is also possible to use the approach with financial estimates. For example, by allocating the letter A to an estimate of over $1m, B to an estimate of between $500k to $1m and so forth.

Project WAG

The project WAG is similar to the enhancement WAG except that it is a little more complex and addresses an entire project. The complexity comes from the following additions to the standard WAG:

- The estimate consists of a series of smaller estimates;
- The estimate includes a range similar to the traditional "best case, worst case and most likely case" estimates. However it starts with the most pessimistic estimate and then works backward to find a rough guess; and

- The estimate explicitly references key risks.

Based on the WAG, the project manager and the customer might decide to:

- Proceed with the project;
- Reject the project; or
- Analyse and estimate only some components in more detail as these represent the biggest risk, the biggest cost or the biggest range in the estimate.

The best way to explain the use of the project WAG is through an example.

Project WAG example

A project manager needs to provide estimates for the following project:

OBJECTIVE: To deliver a data tracking system that automates reporting for sales managers.

In scope	**Out of scope**
System changes to both front end and backend systems Rollout to offices in NSW, Vic and South Australia Delivery of training material to users	Rollout to other Australian states Process change required as part of the project

Step 1 -Gather the available data

The PM must first break the project down into meaningful components. For example:

- High level tasks, milestones or deliverables;
- High level features or stories; or
- The project scope.

The project manager decides that in order to provide an estimate he will break the project into the following relevant components based on the scope and some critical tasks:

- Analysis of management reporting needs;
- Delivery of training materials;
- Front end system changes;
- Backend system changes;
- Rollout to offices in NSW, Vic and South Australia; and
- Building data interfaces to existing systems

Step 2 -Select a rating scale

The project manager creates a WAG generator. In this case our project manager has chosen to estimate in time based on the following categories:

Rating	Estimated time		Rating	Estimated time		Rating	Estimated time
A	Over 6 months		D	2 -4 weeks		G	1 -8 hours
B	3 -6 months		E	2 -10 days		H	¼ - 1 hour
C	4 -12 weeks		F	1 -2 days		I	Less than ¼ hour

Step 3 -Add the components to the table and complete the 99.5% confident column

The next step is to add the components that we are going to estimate. The table below shows the components added to the table.

Now we create the first estimates. Project managers tend to claim that they cannot provide an estimate without doing the analysis stage of the project. But this is not strictly true. What they mean is that they cannot provide an estimate that the customer would like to hear.

It may sound absurd but if we allow the project manager to be as pessimistic as he or she wants to be then it is always possible to provide an estimate that the project manager could be 100% confident of meeting. For example, I would be almost 100% (say 99.9999999%) confident of:

- Launching a new company intranet in less than 10 years; or
- Creating a new blog in Wordpress in less than 1 day.

So the issue is not really that we are unsure how to estimate, it is that we need to either add ridiculous contingency to our estimates or admit that we are less than 100% confident in them.

This is the basis of our WAG. We will start by providing the smallest number that we are pretty sure (99.5%) we can meet. This means that there is a chance we would not deliver in the time frame but we are almost certain we could do it.

At this point you might choose to add risks to the estimates, but this is optional.

First cut of the WAG generator

Item	**99.5%**	**60%**	**Guess**	**Key risks**
Analysis of management needs.	A			Need to lock down the scope before committing.
Delivery of training materials.	B			Training will depend on process changes.

Front end system changes.	C			
Back end system changes.	A			We have not looked at this component -so we have no idea what is involved.
Rollout to offices in NSW, Vic and South Australia.	B			
Building data interfaces.	C			

Step 4 -Break down the items that are too big to provide a meaningful estimate

On the first pass of the estimates, the project manager identified several items as being over 6 months. This is too large a timeframe for the purpose of estimating.

So we break these items down to clarify dependencies, understand the task better or otherwise bring the item down to a point where we can provide some sort of estimate.

Second cut of the WAG generator

Item	99.5%	60%	Guess	Key risks
Workshop to scope management reporting needs.	D			
Follow up analysis on specific management reporting needs.	C			Unforseen reporting requirements lead to scope change - impacting budget.
Confirmation of reporting needs with CFO and Head of Sales.	E			
Confirm process changes with the Sales Operations Team.	E			Lack of confirmation means training material cannot be prepared -impacting schedule.
Delivery of training materials.	D			
Front end system changes.	C			
Data warehouse integration	B			Data warehouse integration not practical -impacting budget and system design.
Mainframe system changes	C			
Rollout to offices in NSW, Vic and South Australia.	B			
Building data interfaces	C			

Step 5 -Add the 60% confidence estimate

We now have a pessimistic but not impossible estimate. But we also want to know how long the project is likely to take, rather than just how bad things could be.

But rather than looking at what range we would be 80% or 90% confident of meeting, we pick one that we are only 60% confident of meeting. I am not sure if I can back this up scientifically, but I believe that:

- People tend to enter the same range for 80% confidence as they did for 100% confidence. But we actually hope to do better than our worst guess;
- If we aim for 50% confidence then people become concerned that we are not really committing to anything; and
- Picking 60% gives the ability to estimate without fear, while also committing to a number we think is more likely than not to be achievable.

You can choose a different confidence level if you believe it is useful.

In addition, at this stage we add more detail in the key risks and dependencies that the 60% confidence guess depends on. These should particularly be added to any items where there is a large difference between our 60% and 99.5% confidence estimates.

In our example the project manager has added the 60% confidence estimates in the table and also added risks. He has also added one more item (pilot rollout) to mitigate the risk he identifies in rolling out the solution to multiple states.

Third cut of the WAG generator

Item	99.5%	60%	Guess	Key risks
Workshop to scope management reporting needs.	D	E		

Follow up analysis on specific management reporting needs.	C	E		Unforseen reporting requirements lead to scope change - impacting budget.
Confirmation of reporting needs with CFO and Head of Sales.	E	F		
Confirm process changes with the Sales Operations Team.	E	E		Lack of confirmation means training material cannot be prepared -impacting schedule.
Delivery of training materials.	D	D		
Front end system changes.	C	C		
Data warehouse integration	B	C		Data warehouse integration not practical -impacting budget and system design.
Mainframe system changes	C	C		Lack of resources delays project.
Pilot rollout in NSW	E	E		
Rollout to offices in NSW, Vic and South Australia.	B	C		Miscommunication or misalignment around training, system change and process changes leads to poor experience.
Building data interfaces to existing systems.	C	C		
Overall estimate	A			

Step 6 - Complete the WAG generator

The final step is to complete the generator by adding information to the remaining fields in the table:

- Add a row for each of the following and include them in the estimate:
  - Business case, project approval and project establishment;
  - Project management effort of 10% if you want to run streams concurrently; and
  - Project deployment, handover, operationalisation and closure
- Add up the worst case values for the 99.5% column and then adjust for the fact that many tasks can be completed in parallel. Take a guess as to which letter best represents the total project duration/cost and add this to the bottom of the table;
- For each item, guess how long the item will take to deliver in days or weeks (or how much it will cost in dollars). This should be a figure from the range shown in the 60% confident column;
- Add the numbers in the guess column and adjust for items that can occur in parallel or a little contingency and enter the number at the bottom of the table in the square provided; and
- Provide an estimate along the following lines:
  - An optimistic guess is that it will take/cost (the total from the guess column) but it could be as much as (the total for the 99.5% column);
  - Subject to the following risks and assumptions (show the risks from the risk column); and
  - If you want to know the critical areas that are impacting the estimate then they are (list the items that have the biggest impact or carry the largest cost).

Third cut of the WAG generator

Item	99.5%	60%	Guess	Key risks
Workshop to scope management reporting needs.	D	E	4 days	
Follow up analysis on specific management reporting needs.	C	E	5 days	Unforseen reporting requirements lead to scope change - impacting budget.
Confirmation of reporting needs with CFO and Head of Sales.	E	F	2 days	
Confirm process changes with the Sales Operations Team.	E	E	6 days	Lack of confirmation means training material cannot be prepared -impacting schedule.
Delivery of training materials.	D	D	15 days	
Front end system changes.	C	C	30 days	

Data warehouse integration	B	C	45 days	Data warehouse integration not practical -impacting budget and system design.
Mainframe system changes	C	C	30 days	Lack of resources delays project.
Pilot rollout in NSW	E	E	5 days	
Rollout to offices in main states	B	C	40 days	Miscommunication or misalignment around training, system change and process changes leads to poor experience.
Building data interfaces to existing systems.	C	C	35 days	
Project management			25 days	
Business case and setup	C	D	10 days	
Handover and closure	C	C	25 days	
Overall estimate	A		277 days	Recommend kill the project.

How accurate is the project WAG?

This estimate is not very precise. But it is often accurate enough to compare projects, highlight where further analysis will be the most beneficial, understand the riskiness of the project and make decisions about projects.

How does the project WAG compare to story points or "T-Shirting"?

A common approach to estimating is to start with "T-shirting" and then move onto story points. In this respect the WAG approach is similar to T-Shirting.

T-Shirting involves the team sitting down to compare their stories to t-shirt sizes. In other words they will break their stories into:

- Extra small;
- Small;
- Medium;
- Large; and
- Extra large.

This allows the team to discuss which stories are the most valuable and compare this to which stories are the most costly to implement. They might then simplify the project before doing too much analysis on features that will not add sufficient value.

This is a popular technique and works well for many teams. But I don't use it myself because it doesn't give a feel for how long the project will take and it also lacks the benefits of story points without being (in my opinion) much easier.

So why not jump straight to story points for large sections of the project. The short answer is that you can and it can work well. All you need to do is to apply the same approach to project phases or major deliverables that teams usually apply to stories.

I tend to use the WAG approach to projects where I don't yet have an estimate of the velocity and I am only after a quick and dirty estimate. After I have done that estimate then (assuming that we go forward) I will break the project into smaller components and apply story points, which then allow better release planning and change management down the track in addition to providing an initial estimate.

Conclusions

The more experience I gain the more I come back to the realisation that the best way to estimate work is to involve the team in sharing their understanding of the problem, their expectations about what a solution will involve and their belief as to how hard the problem will be to solve.

In agile teams this is generally reflected in the use of planning poker and a focus on estimating by feature rather than task.

This article is a collection of some of the techniques I have used with teams to:

- Help clarify the assumptions involved in the work they are estimating; and
- Provide a framework for the discussions the team have in coming up with better estimates.

The WAG generator provides a simple line in the sand to help the team and its sponsor decide whether to expend further effort on a proposed initiative. The other techniques help the team to explore the problems they are solving more fully. For example, the heat map and exploration factor techniques are both ways to highlight where danger or likely variance is hidden, while the rest of the techniques help the team get started with the task of determining an estimate.

The techniques discussed here are certainly not a complete list of the best practices available. Rather they are a collection of techniques that have worked well for me in the past. I am still learning new techniques and I would be more than happy to hear about techniques that have worked for you and your team.

About The Author

James King is a consultant who works with teams to improve their IT processes. He also coaches teams and leaders in improved communication, risk management and knowledge management approaches.

James also works closely with Software Education who provide training in Agile techniques, testing and business analysis.

3 comments **Watch Thread** **Reply**

Using FPA instead of "Agile" stuff by Adam Nemeth Posted Dec 28, 2010 10:55 AM

Re: Using FPA instead of by Dandik M Posted Jan 7, 2011 6:04 AM
Potent, Focused Approach by Benjamin Lunzer Posted Jan 5, 2011 11:39 AM

Sort by date descending

Using FPA instead of "Agile" stuff

Dec 28, 2010 10:55 AM by **Adam Nemeth**

I don't really use new stuff which has an "agile" vignette on them even in an agile environment.

In the most agile team I had, a real agile one (own company, only developers, own product, full commitment, cycles and so on) - and even afterwards, I use a modified version of the Function Point Analysis.

What does it say?
- Every user story will sure have some datasources to get data from. Say these are Input Forms (IFs). Let's say creating such a form takes 5 points.

- Every user story will surely have some outputs. Let's call them Output Forms. Let's say it takes 4 point to create them

- Every user story will have to deal with some internal datastructures - tables, files, so on. Let's see how many distinct classes (tables, filetypes) seem to be involved; let's say developing handling of such a type is about 9 points, and call them Internal Logical Files (ILFs)

- Some user stories will have to deal with systems not within reach; these could be unknown libraries, web service APIs, maybe even the mailsystem. Let's say handling such stuff is 12 points.

Then, for each of the stories, we have a formula, like:
Creating a login screen = 1 IF, 1 OF, 2 ILF, 0 EX

Add them together multiplied:
1 x 4 + 1 x 5 + 2 x 9 + 0 x 12 = 18 FP (function point)

We could have also additional measures, but this will make them ok.

Now we have to determine how much time does it take to create a single function point. It takes a lot of efforts; my agile team had about 0.5 hours, the enterprise team was sometimes as slow as 2 hours. So, creating a login form with all bells and whistles takes about a day, or a week, depending wether you are in a small, enthusiastic company, or in an enterprise cubicle. Makes a lot of sense, even if you don't like the results.

Also the factors could be changed.

It's not agile, but this method was able to provide +–10% efficiency; so, after the second iteration, it's expected that a two week sprint's products cannot be late more than a day.

Reply

Potent, Focused Approach

Jan 5, 2011 11:39 AM by **Benjamin Lunzer**

Excellent and pragmatic. Thank you.

Ben Lunzer
Senior Architect
GXS, Inc.
Gaithersburg, MD

Reply

Re: Using FPA instead of

Jan 7, 2011 6:04 AM by **Dandik M**

Can you point any differences of FPA compared to story points? They seam to be exactly the same to me. Both measure Task "weight" and then are multiplied by team velocity to answer "how long will it take?" question. The way you get FPA/Story points depends on the team and how you are used to do it, what works for you. Otherwise I see no difference.

Reply