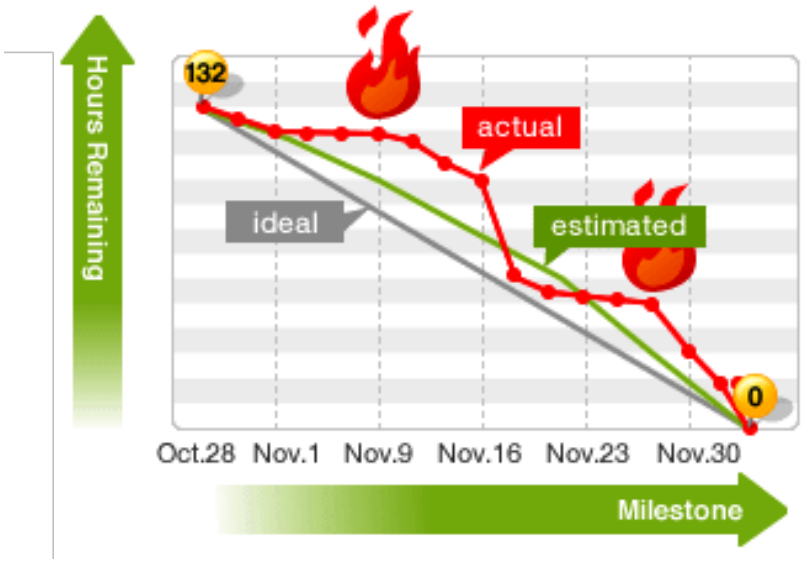
Kanban agile planning with Burn-Down chart

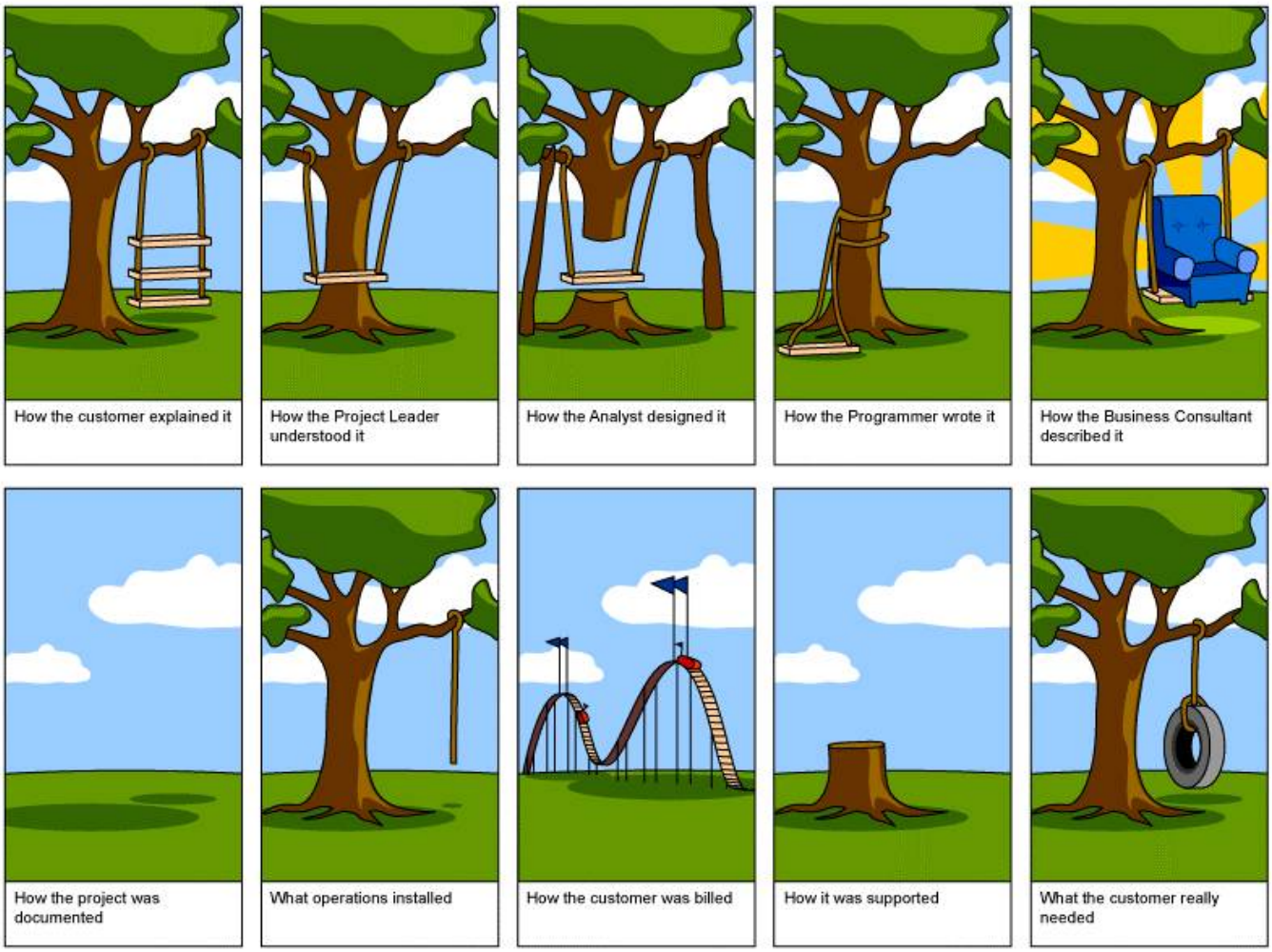


Three simple truth:

- It is impossible to gather all the requirements at the beginning of a project.
- Whatever requirements you do gather are guaranteed to change.
- There will always be more to do than time and money will allow.

The agile team should have a strategy for dealing with any changes in the process. There are several problems with a static plans:

- Our team changes – some of the lead developers can leave to another project of great strategic importance (they was used to say about our project like this). Next, we realize that, we aren't going as fast as we had thought.
- Halfway through the project, customer discovers what he really want.
- … running out of time.



In a rush to meet the new deadline, testing gets cut, the existing team need to cancel the vacations, etc., etc., it becomes another late, over-budget and failed IT project.

To deal with this realities we need a different way of planning things that will deliver great value to our customers, the plan that is visible-open-honest, lets us make promises we can keep and enables us to adopt changes when necessary.

Agile planning is measuring the speed that a team can run the customer requirements into working, production ready software and estimate when they will done.

The to-do list or customer requirements will be called user-stories. The team’s velocity is the speed of converting the user-stories into working software. We have to use team’s velocity to measure our team’s productivity to set the expectations about software delivery to the customer for the future.

Every user-story should be sized based on its difficulty and the requirement. It is good to use relative or point based systems for sizing. Let’s say small (1pts), medium is (3pts) and large is (5pts).

Let’s start creating our agile plan:

Step 1: Creating release story list. This is a logical grouping of user-stories that we can deliver to the customer after some iteration. Each logical grouping is estimated by team and can be prioritized by customer. After grouping the user-stories we can have the number of releases. Each release period can be in a range of 1 to 6 month. Each release will have several iterations which each of them can continue from 1-2 weeks.

Step 2: Sizing. Every user-story needs to be estimated and voted by the team, customers assistance would be great, because when he wants to add something in the feature, more or less he can have small imagination of how can it affect to the delivering process. Once we have our user-stories gathered and estimated we know about total points of the project.

Step 3: Prioritizing. We have to get important staff first. Together with customer and team it is good to sort the items from most important to less important. The risky ones should be managed differently.

Step 4: Estimating the team's velocity. For each iteration the team's velocity should be estimated based on current situation and previous delivered velocity. So for the first iteration the team's velocity should be guessed.

Team velocity = completed user-stories / iteration

Step 5: Delivering. We can deliver by specific iteration completed dates or by the set of the features. Every iteration the team will complete (**burns**) amount of user-stories or points and by its average we can estimate what the team can deliver in which iteration.

Burn-Down chart:

This kind of chart is extremely useful to visualize the process of how quickly the team completing the customer's user-stories, in other words burning through our customer's requirements. It will help us to tell how much work has been done, how much work remains, the team's velocity and our expected completion dates.

Let's discover most of this things in example below. Imagine that our personal work in audit contains 53 hours of work that we should release at 1st December.

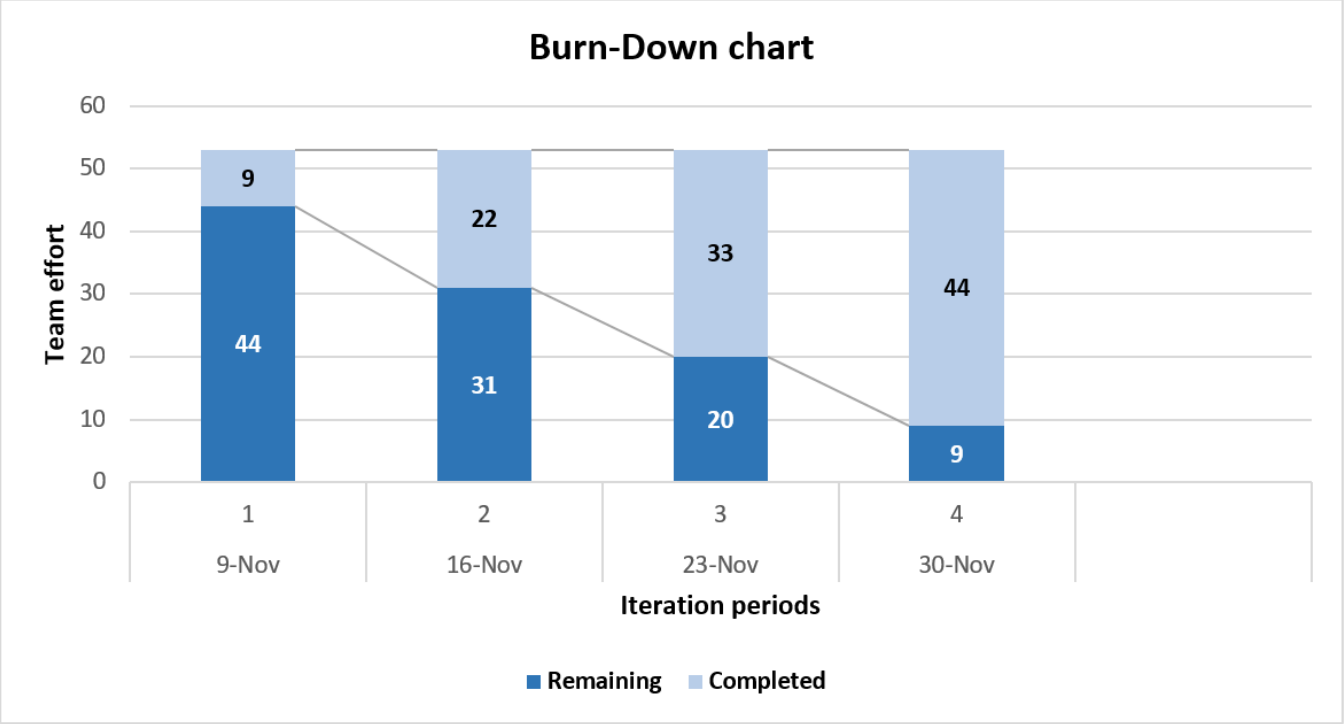
ITERATIONS	9-NOV 1	16-NOV 2	23-NOV 3	30-NOV 4
TOTAL EFFORT	53	53	53	53
VELOCITY	9	13	11	11
REMAINING	44	31	20	9
COMPLETED	9	22	33	44
% COMPLETE	17%	42%	62%	83%

Flow status within iterations

RELEASE OF PERSONAL WORK ☺, DECEMBER 1

TOTAL ESTIMATION:	53
COMPLETED:	44
REMAINING:	9
% COMPLETE:	83%
AVERAGE VELOCITY	11
ESTIMATED # ITERATIONS REMAINING	1

Flow status on release date



Burn-down chart in release date

From this chart we can see that the release was planning to happen on 1 of December but it didn't happened. That can happen because of user-story estimations, team's velocity or other reasons. The table with flow status on release date shows how many more iterations need to make the release happen.

Burn-down chart makes all the events in our project visible. If the customer decide to add something more, we can instantly see the impact of it to the delivery dates. Project's burn-down chart shows the flow like it is. This is highly visible part of the planning.