# Timeboxing

A project has the triple constraints of time, scope and cost (there are other parameters of course). If one parameter changes, something else must change to keep the three as a constant. Whilst in a perfect world, the project schedule and completion dates would be derived from the amount of work to be done and the number of resources available, this is rarely the situation.

Usually, there is an end date by which the work must be completed! The end-date may be determined by a government regulation, a contract, or a scheduled event or to coincide with another initiative. This situation is referred to as a timebox, meaning you have a fixed amount of time to get the work done and the end-date is "boxed" in[1].

Timeboxing allocates a fixed time period to the overall project, or each planned phase, iteration, sprint, or activity. The schedule is divided into a number of separate time periods (timeboxes), with each part having its own deliverables, deadline and budget.

Timeboxing makes time a fixed component and adjust the other two constraints (scope and cost) to fit. This can lead to a number of interpretations:

- We will complete as much as we can in the period available.
- We will apply as many resources as we need to complete the work in the period available.
- Or something in between.

The process works best in an agile project environment where the product is being delivered in a series of sprints or iterations and the scope of later sprints can be varied to accommodate work dropped from an earlier sprint. Timeboxing is closely associated with other concepts such as backlog management.

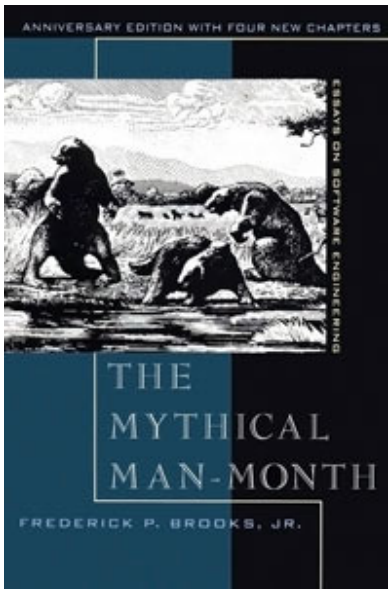
There is nothing wrong with having a fixed end-date. The problem arises when the project manager and team do not think they can get the work done by the deadline. In that case, the project manager needs to raise this as a risk. Potential risk plan actions include:

- Assigning more resources to the project. Too many resources may have diminished value (see below);
- Having the team work overtime, with the understanding that overtime itself has a diminishing return, and that long-term overtime can actually have a negative effect as people become tired and demotivated;
- Working with the stakeholders to scale back the required deliverables due by the deadline. This may include removing entire deliverables or functionality from required deliverables and is the default option for 'sizing sprints' in an agile project;
- Determining whether some required deliverables and features can actually be delivered later than the due date. In these cases, a 90% solution may be viable at the due date, with the additional work completed soon after.

Timeboxing is a very simplistic approach to management but quite common in software development. Some of the problems include:

[1] For more on ***managing agile*** see: https://www.mosaicprojects.com.au/PDF_Papers/P109_Thoughts_on_Agile.pdf

- The lack of any correlation between resource numbers and productivity. Increasing resources will definitely cost more but there is unlikely to be a commensurate increase in productivity[2]. This problem was elegantly defined by Fredrick P. Brooks in *The Mythical Man-Month*[3], summarised by the saying adding manpower to a late software project makes it later!
- De-scoping normally reduces value far quicker than any other element[4]. Timeboxing can quickly lead to a software release that has cost as much as planned and taken the time planned but delivered almost no value.




Just because time is easy to measure, it does not mean time is the most important component of a project. To be effective, time boxing requires that:

1. The features or user requirements[5] that make up the total delivery are grouped into functionally complete subsets;
2. The subsets are prioritised so it is clear which requirements should be implemented first and which ones could be eliminated if there is not enough time to complete all of them; and
3. Reasonable assurance is provided to the customer about the feasibility of a given subset within the imposed frame

Time boxes which are merely a self-imposed target, or an externally imposed target, created without any real need being defined and without agreed partial outcomes and justified estimations of what's practical to achieve are of little value or use. All these artificial targets achieve are increased costs and reduced quality and value.

The solution is to estimate the amount of Schedule Risk associated with the 'artificial deadline'. Even if the manager or sponsor has dictated a fixed end-date for the project, it is important to carefully build the schedule as if you did not have the fixed end-date first. This will give you a sense for how realistic the fixed date is, and what has to be compromised to achieve it.

For example, if you have a project that has to be completed in 6 months, and you create a "normal" schedule that shows the project will be complete in 6 ½ months, it would not be too much of a stretch to think you could accelerate the work[6] to achieve completion within the specified 6 months.

However, if the "normal" schedule shows completing the work requires a 10 months, you will understand the difficulty and the risk associated with trying to get all of the work completed in the six-month timebox. This does not mean that you will be able to change the 6 month deadline. But it does give you the information needed to have a fact-based discussion with the sponsor about the project risks and options that are available – generally de-scoping is the only option if time cannot change!

***Moscow*** is one method of prioritisation which requires the customer to rank his or her preferences into a series of categories such as 'Must have', 'Should have', 'Could have' or 'Won't have'. These categories are commonly known by the acronym 'Moscow'.

---

[2] For more on ***estimating task durations*** see: https://www.mosaicprojects.com.au/WhitePapers/WP1052_Time_Estimating.pdf

[3] First published 1975 - The Mythical Man-Month: Essays on Software Engineering, Frederick P. Brooks (ISBN 0-201-00650-2) (Cover from the 20th Anniversary edition – 1995)

[4] For more on ***benefits and value*** see: https://www.mosaicprojects.com.au/WhitePapers/WP1023_Benefits_and_Value.pdf

[5] For more on ***requirements*** see: https://www.mosaicprojects.com.au/WhitePapers/WP1071_Requirements.pdf

[6] For more on ***schedule acceleration*** see: https://www.mosaicprojects.com.au/WhitePapers/WP1059_Schedule_Compression.pdf

The definition of each category, aligned to sensible framing of a 'time box', are:

- **Must have** contains all requirements that must be satisfied in the final delivery for the solution to be considered a success. Short of a disaster, these features should be able to be delivered within the defined time box. Based on 'safe' estimates of the time and effort involved.
- **Should have** represents high-priority items that should be included in the solution if possible. These features should have a fair chance of being delivered within the defined time box if normal circumstances prevail.
- **Could have** are those requirements which are considered desirable but not necessary. They will be included if there is any time left after developing the previous two categories (ie, the work goes reasonably well).
- **Won't have** is used to designate requirements that will not be implemented in a given time box, but may be considered for the future.

Fitting of requirements into these categories is a consequence of what the development team believes can be accomplished under the specific project context and budget[7]. As a consequence, the client can be almost certain that all the requirements in the "must have" category will be completed within the time box because a requirement was only included in it if there was enough room to develop it under a worst case assumption. In all probability, a good proportion of the 'should haves' will be delivered as well with a possibility of a few 'could haves'.

## Timeboxing in Agile Projects[8]

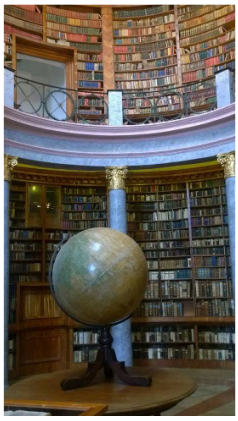
Agile projects use timeboxing quite differently. The approach used in Scrum and other Aglie approaches is to design the content of each 2 to 4 week sprint is scoped to be achievable by the team responsible based on their established productivity levels. Each sprint starts with a kick-off meeting, plans how the work will be accomplished and focuses on achieving completed and tested code by the designated completion date. This approach differs from the big bang 'timeboxing' discussed above in two key elements. First the scope of each sprint is based on realistic production levels adjusted based on experience to date. Second, the overall management of the project sits above the individual sprints and focuses on delivering value to the project's clients.

---

First published 20th Feb 2010, augmented and updated.

[7] For more ***sophisticated ranking options*** see: https://www.mosaicprojects.com.au/WhitePapers/WP1062_Ranking-Requirements.pdf

[8] For more on ***managing Agile projects*** see: https://www.mosaicprojects.com.au/PDF_Papers/P109_Thoughts_on_Agile.pdf





For more information on scheduling and planning, visit Mosaic's planning and scheduling home page at: https://mosaicprojects.com.au/PMKI-SCH.php