Timeboxing

From Wikipedia, the free encyclopedia

In time management, **timeboxing** allocates a fixed time period, called a **time box**, to each planned activity. Several project management approaches use timeboxing. It is also used for individual use to address personal tasks in a smaller time frame. It often involves having deliverables and deadlines, which will improve the productivity of the user.

Contents

> Timebox—don't scopebox.
>
> — Mary Poppendieck , *Leading Lean Software Development*[1]

In project management

Timeboxing is used as a project planning technique. The schedule is divided into a number of separate time periods (timeboxes), with each part having its own deliverables, deadline and budget.

As an alternative to fixing scope

In project management, the triple constraints are time (sometimes schedule), cost (sometimes budget), and scope (sometimes performance).[2][3][4][5][6] Quality is often added,[7][8] sometimes replacing cost.[9] Changing one constraint will probably impact the rest.[5]

Without timeboxing, projects usually work to a fixed scope,[10] such that when it is clear that some deliverables cannot be completed, either the deadline slips (to allow more time) or more people are involved (to do more in the same time). Usually both happen, delivery is late, costs go up, and often quality suffers.

With timeboxing, the deadline is fixed, but the scope may be reduced. This focuses work on the most important deliverables. For this reason, timeboxing depends on the prioritisation (with the

MoSCoW Method for example) of deliverables, to ensure that it is the project stakeholders who determine the important deliverables rather than software developers.

To manage risk

Timeboxes are used as a form of risk management, to explicitly identify uncertain task/time relationships, i.e., work that may easily extend past its deadline. Time constraints are often a primary driver in planning and should not be changed without considering project or sub-project critical paths. That is, it's usually important to meet deadlines. Risk factors for missed deadlines can include complications upstream of the project, planning errors within the project, team-related issues, or faulty execution of the plan. Upstream issues might include changes in project mission or backing/support from management. A common planning error is inadequate task breakdown, which can lead to underestimation of the time required to perform the work. Team-related issues can include trouble with inter-team communication; lack of experience or required cross-functionality; lack of commitment/drive/motivation (i.e. poor team building and management).

To stay on deadline, the following actions against the triple constraints are commonly evaluated:

- Reduce scope: drop requirements of lower impact (the ones that will not be directly missed by the user)
- Time is the fixed constraint here
- Increase cost: e.g., add overtime or resources

Adoption in software development

Many successful software development projects use timeboxing, especially smaller ones.[11] Adopting timeboxing more than tripled developer productivity at DuPont in the '80s.[12] In some cases, applications were completely delivered within the time estimated to complete just a specification.[12] However, Steve McConnell argues that not every product is suitable[12] and that timeboxing should only be used after the customer agrees to cut features, not quality.[12] There is little evidence for strong adoption amongst the largest class of projects.[11]

Timeboxing has been adopted by some notable software development methodologies:

- Dynamic systems development method (DSDM)
- In lean software development, pull scheduling with Kanban provides short term time management. When developing a large and complex system, when long term planning is required timeboxing is layered above.[13]
- Rapid application development (RAD) software development process features iterative development and software prototyping. According to Steve McConnell, timeboxing is a "Best Practice" for RAD and a typical timebox length should be 60–120 days.[12]

- Scrum was influenced by ideas of timeboxing and iterative development.[14] Regular timeboxed units known as sprints form the basic unit of development.[15] A typical length for a sprint is 30 days.[16][17] Sprint planning, sprint retrospective and sprint review meetings are timeboxed.[16]
- In Extreme programming methodologies, development planning is timeboxed into iterations typically 1, 2 or 3 weeks in length. The business revalues pending user stories before each iteration.[18]

Agile software development advocates moving from *plan driven* to *value driven* development. Quality and time are fixed but flexibility allowed in scope. Delivering the most important features first leads to an earlier return on investment than the waterfall model.[6]

A lack of detailed specifications typically is the result of a lack of time, or the lack of knowledge of the desired end result (solution). In many types of projects, and especially in software engineering, analyzing and defining *all* requirements and specifications before the start of the realization phase is impossible. Timeboxing can be a favorable type of contracting for projects in which the deadline is *the* most critical aspect and when not all requirements are completely specified up front.

There is also a better structure for allowing for new insights that are developed during the project to be reflected in the end result.

In personal time management

Individuals can use timeboxing for personal tasks, as well. This technique utilizes a reduced scale of time (e.g., thirty minutes instead of a week) and deliverables (e.g., chores instead of a component of a business project). Personal timeboxing is said to help curb perfectionist tendencies (by setting a firm time and not overcommitting to a task). Adam Pash writes that timeboxing helps overcome procrastination and that many people find that the time pressure created boosts creativity and focus.[19]

Relationship with other methods

Timeboxing acts as a building block in other personal time management methods:

- The Pomodoro Technique is based on 25 minute timeboxes of focused concentration separated by breaks allowing the mind to recover.[20]
- Andy Hunt gives timeboxing as his 'T' in SMART.[21]

See also

- Life hacking

References

1. ^ Poppendieck, Mary (2010). *Leading Lean Software Development: Results Are Not the Point*. Upper Saddle River, NJ: Addison-Wesley. p. 139. ISBN 978-0-321-62070-5.
2. ^ *What are the Triple Constraints in Project Management* (http://www.projectmanagement.net.au/triple_constraints), article by Rod Hutchings on Project Management Australia (http://projectmanagement.net.au/) (22 Oct 2008)
3. ^ Chatfield, Carl. "A short course in project management" (http://office.microsoft.com/en-us/project/HA102354821033.aspx). Microsoft.
4. ^ Dobson, Michael (2004). *The triple constraints in project management*. Vienna, Va: Management Concepts. ISBN 1-56726-152-3.
5. ^ ***a b*** Kanabar, Vijay (2008). *MBA Fundamentals : Project Management*. New York: Kaplan Pub. p. 51. ISBN 978-1-4277-9744-5.
6. ^ ***a b*** Leffingwell, Dean (2011). *Agile Software Requirements : Lean requirements practices for teams, programs, and the enterprise* (http://books.google.co.uk/books?id=pTExbNmZwZUC). Upper Saddle River, NJ: Addison-Wesley. pp. 17–19. ISBN 978-0-321-63584-6.
7. ^ Snedaker, Susan; Nels Hoenig (2005). *How to Cheat at IT Project Management*. Syngress. ISBN 1-59749-037-7.
8. ^ Beck, Kent (2000). *Extreme programming eXplained : embrace change*. Reading, MA: Addison-Wesley. pp. 15–19. ISBN 0-201-61641-6.
9. ^ Dangelo, Mark (2005). *Innovative relevance : realigning the organization for profit : it is not a battle for the "shore lines" - it's a struggle for purpose* (http://books.google.co.uk/books?id=LQx3lab0KnIC). New York: iUniverse. p. 53. ISBN 978-0-595-67081-9.
10. ^ Godin, Seth. *Getting Real: The smarter, faster, easier way to build a successful web application*. 37signals.
11. ^ ***a b*** For all project types time boxing ranked 23 and rated "Very Good Practice"; for small (1000 function point) projects ranked 7 and rated a "Best Practice" by the survey in Jones, Capers (2010). *Software engineering best practices lessons from successful projects in the top companies*. New York: McGraw-Hill. ISBN 978-0-07-162162-5.
12. ^ ***a b c d e*** McConnell, Steve (1996). *Rapid Development : taming wild software schedules*. Redmond, Wash: Microsoft Press. pp. 575–583. ISBN 1-55615-900-5.
13. ^ Poppendieck, Mary (2010). *Leading Lean Software Development : Results are not the Point*. Upper Saddle River, NJ: Addison-Wesley. pp. 137–140. ISBN 978-0-321-62070-5.
14. ^ Coplien, James (2010). *Lean Architecture for Agile Software Development* (http://books.google.co.uk/books?id=lpvY36MPMUwC). Chichester Hoboken, N.J: Wiley. p. 25. ISBN 978-0-470-68420-7.
15. ^ Cohn, Mike (2010). *Succeeding with Agile : Software Development using Scrum*. Upper Saddle River, NJ: Addison-Wesley. pp. 257–284. ISBN 978-0-321-57936-2.

16. ^ **a b** Schwaber, Ken (2009). *Agile Project Management with Scrum* (http://books.google.co.uk/books?id=RpYX01XVMksC). New York: O'Reilly Media, Inc. ISBN 978-0-7356-3790-0.
17. ^ Leffingwell, Dean (2011). *Agile Software Requirements : Lean requirements practices for teams, programs, and the enterprise* (http://books.google.co.uk/books?id=pTExbNmZwZUC). Upper Saddle River, NJ: Addison-Wesley. p. 15. ISBN 978-0-321-63584-6.
18. ^ Beck, Kent (2000). *Extreme programming eXplained : embrace change*. Reading, MA: Addison-Wesley. pp. 85–96. ISBN 0-201-61641-6.
19. ^ Pash, Adam (2011). *Lifehacker the guide to working smarter, faster, and better* (http://books.google.co.uk/books?id=d-FYJceblAMC). Indianapolis, Ind: Wiley. Hack 29. ISBN 978-1-118-13345-3.
20. ^ Nöteberg, Staffan. *Pomodoro Technique Illustrated*. Raleigh, N.C: Pragmatic Bookshelf. ISBN 978-1-934356-50-0.
21. ^ Hunt, Andrew (2008). *Pragmatic thinking and learning : refactor your wetware*. Raleigh: Pragmatic. ISBN 978-1-934356-05-0.

External links

- Agile Requirements Change Management (http://www.agilemodeling.com/essays/changeManagement.htm)
- Designing To Schedule (http://hosteddocs.ittoolbox.com/EM032006.pdf)
- MSDN - How To Use Timeboxing for Getting Results (http://blogs.msdn.com/jmeier/archive/2007/10/21/how-to-use-time-boxing-for-getting-results.aspx)
- Using timeboxing for a motivation boost (http://motivationgrid.com/timebox-quick-motivation-boost)
- Why time boxed sprints? What happens to releases? (https://www.linkedin.com/today/post/article/20140808230234-2622543-why-time-boxed-sprints-what-happens-to-releases)