Timeboxing

In time management, **timeboxing** allocates a fixed time period, called a **timebox**, within which planned activity takes place. It is employed by several project management approaches and for personal time management.

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

Contents

In project management
- As an alternative to fixing scope
- To manage risk
- Adoption in software development

In personal time management

Relationship with other methods

References

External links

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. Sometimes referred to as *schedule as independent variable* (SAIV).[2]

As an alternative to fixing scope

In project management, there are generally considered to be three constraints of time (sometimes schedule), cost (sometimes budget), and scope;[3][4][5][6][7] with quality often added as a fourth constraint (represented as the middle of a triangle),[8][9][10] The assumption is that a change in one constraint will affect the others.[6]

Without timeboxing, projects usually work to a fixed scope,[11] in which case when it becomes clear that some deliverables cannot be completed within the planned timescales, either the deadline has to be extended (to allow more time to complete the fixed scope) or more people are involved (to complete the fixed scope in the same time). Often both happen, resulting in delayed delivery, increased costs, and often reduced quality (as per The Mythical Man-Month principle).

With timeboxing, the deadline is fixed, meaning that the scope would have to be reduced. As this means organizations have to focus on completing the most important deliverables first, timeboxing often goes hand-in-hand with a scheme for prioritizing of deliverables (such as with the MoSCoW method).[12]

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.[13] Adopting timeboxing more than tripled developer productivity at DuPont in the '80s.[14] In some cases, applications were completely delivered within the time estimated to complete just a specification.[14]However, Steve McConnell argues that not every product is suitable[14] and that timeboxing should only be used after the customer agrees to cut features, not quality.[14] There is little evidence for strong adoption amongst the largest class of projects.[13]

Timeboxing has been adopted by some notable software development methodologies:

- Dynamic systems development method (DSDM)[12]
- 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.[15]
- 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.[14]
- Scrum was influenced by ideas of timeboxing and iterative development.[16] Regular timeboxed units known as sprints form the basic unit of development.[17] A typical length for a sprint is less than 30 days.[18][19] Sprint planning, sprint retrospective and sprint review meetings are timeboxed.[18]
- 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.[20]

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.[7]

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. This also allows for new feedback or insights discovered during the project to be reflected in the end result.[12]

In personal time management

Timeboxing can be used for personal tasks, as well, in which case it uses a reduced scale of time (e.g., thirty minutes) and of deliverables (e.g., a household chore instead of project deliverable).

Personal timeboxing is also said to act as a life hack to help curb perfectionist tendencies (by setting a firm time and not overcommitting to a task)[21] which can also enhance creativity and focus (by creating a sense of urgency or increased pressure).[22]

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.[23]
- Andy Hunt gives timeboxing as his 'T' in SMART.[24]

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. Boehm, Barry W.; Boehm, Barry; Turner, Richard (2004). *Balancing Agility and Discipline: A Guide for the Perplexed*. Addison-Wesley Professional. ISBN 9780321186126.
3. *What are the Triple Constraints in Project Management* Archived 2006-08-20 at Archive.today, article by Rod Hutchings on Project Management Australia Archived 2009-02-16 at the Wayback Machine (22 Oct 2008)
4. Chatfield, Carl. "A short course in project management". Microsoft.
5. Dobson, Michael (2004). *The triple constraints in project management*. Vienna, Va: Management Concepts. ISBN 1-56726-152-3.
6. Kanabar, Vijay (2008). *MBA Fundamentals : Project Management*. New York: Kaplan Pub. p. 51. ISBN 978-1-4277-9744-5.
7. Leffingwell, Dean (2011). *Agile Software Requirements : Lean requirements practices for teams, programs, and the enterprise*. Upper Saddle River, NJ: Addison-Wesley. pp. 17–19. ISBN 978-0-321-63584-6.
8. Snedaker, Susan; Nels Hoenig (2005). *How to Cheat at IT Project Management*. Syngress. ISBN 1-59749-037-7.
9. Beck, Kent (2000). *Extreme programming eXplained : embrace change*. Reading, MA: Addison-Wesley. pp. 15–19. ISBN 0-201-61641-6.
10. 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*. New York: iUniverse. p. 53. ISBN 978-0-595-67081-9.

11. Godin, Seth. *Getting Real: The smarter, faster, easier way to build a successful web application*. 37signals.
12. Jennifer., Stapleton (1997). *DSDM, dynamic systems development method : the method in practice*. Harlow, England: Addison-Wesley. ISBN 0201178893. OCLC 36755892.
13. 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.
14. McConnell, Steve (1996). *Rapid Development : taming wild software schedules*. Redmond, Wash: Microsoft Press. pp. 575–583. ISBN 1-55615-900-5.
15. 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.
16. Coplien, James (2010). *Lean Architecture for Agile Software Development*. Chichester Hoboken, N.J: Wiley. p. 25. ISBN 978-0-470-68420-7.
17. 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.
18. Schwaber, Ken (2009). *Agile Project Management with Scrum*. New York: O'Reilly Media, Inc. ISBN 978-0-7356-3790-0.
19. Leffingwell, Dean (2011). *Agile Software Requirements : Lean requirements practices for teams, programs, and the enterprise*. Upper Saddle River, NJ: Addison-Wesley. p. 15. ISBN 978-0-321-63584-6.
20. Beck, Kent (2000). *Extreme programming eXplained : embrace change*. Reading, MA: Addison-Wesley. pp. 85–96. ISBN 0-201-61641-6.
21. Frankton, James (4 April 2014). "40 Ways To Stop Procrastinating". *Why Am I Lazy?*. Retrieved 22 September 2014.
22. Pash, Adam (2011). *Lifehacker the guide to working smarter, faster, and better*. Indianapolis, Ind: Wiley. Hack 29. ISBN 978-1-118-13345-3.
23. Nöteberg, Staffan. *Pomodoro Technique Illustrated*. Raleigh, N.C: Pragmatic Bookshelf. ISBN 978-1-934356-50-0.
24. Hunt, Andrew (2008). *Pragmatic thinking and learning : refactor your wetware*. Raleigh: Pragmatic. ISBN 978-1-934356-05-0.

External links

- How To Use Timeboxing To Get More Done
- Agile Requirements Change Management
- Designing To Schedule
- MSDN - How To Use Timeboxing for Getting Results
- Using timeboxing for a motivation boost
- Why time boxed sprints? What happens to releases?