What to Do When Milestones Don't Easily "Get to Done"

Date:	Jul 5, 2013	
Title:	What to Do When Milestones Don't Easily "Get to Done"	
Code:	AR-MDN	

Abstract: *One of the most complex issues in project management to handle is when a team struggles at getting "to done" at key milestones. This article presents the problem along with suggestions on how to combat it.*

Version:	(2013-07-04)	Originated
	1.0 (2013-07-05)	First published

What If All Milestones Don't Come Together?

Often a project will achieve a milestone in different stages. For example, a database project composed of three major modules may achieve so-called "alpha" at different points in time. What date should you use to specify completion? When all three modules achieve alpha or when two of the three modules, the majority, achieve alpha?

Although the precise answer depends on the specific project (or your organization's "rule of thumb"), the team should agree that a milestone is met when certain basic requirements have been achieved. Agreement on a milestone's completion date should be designed to give the team confidence that the *next* milestone can be met. Take the traditional alpha and beta milestones used for years in traditional software development. Alpha would imply code is basically functional and, correspondingly, beta would signify functionally complete passing unit tests. (I would normally hop right into Agile instead, but alpha and beta works quite nicely as examples for this topic.)

For instance, the team may agree that alpha is met when modules 1 and 2 are complete and the user interface of module 3 is complete. The fact that all of the components of module 3 may not be at a true alpha may be considered a negative. Given "lax definitions," some product components may not make the alpha level of completeness until a week before beta!

Further, some projects may struggle at getting to alpha until some components are at near beta completion.

I can think of one example. When a desktop publishing product or word processor is being developed, the text engine that handles text in an arbitrary bounding frame must be at near-beta quality before any of the other functions can achieve alpha status.

What's worse is when the engineers (programmers) believe that a project is truly at the alpha milestone and yet QA doesn't agree resulting in a huge gap in interpretation and alpha completion attempts lasting for several more weeks (or months). The same is true with Agile projects where the goal is, the case of Scrum, some team members struggle at getting to 100% complete and product backlog items reappear for some time.

I think we’ve all encountered the infamous “almost done” milestone.

Some Ways to Clean Up Sloppy Milestones

1 Take the opportunity to teach your teams how to resolve milestone conflicts.

Although you really don’t want to embarrass a specific team member, isolate a specific, difficult schedule decision as an example case. At a critical milestone that needs to be completed, discuss the circumstances and outcomes with the team (and, quite possibly, the stakeholders). Use the situation as a lessons learned to avoid next time.

2 Use simple tools like RACI to clearly define roles.

According to Shannon Navin, a RACI matrix may be the perfect tool to remove any ambiguity of decisions, particularly when milestones need to be “negotiated.” A RACI matrix is typically represented as a two-dimensional table defining who is responsible (R), accountable (A), consulted (C), and informed (I). By assigning roles to key stakeholders, you can avoid the inevitable “you can’t make that decision!” dialog that ensues when tem members “slug it out” in deciding task and, subsequently, milestone closure.

Once you understand how RACI works, it only takes about ten minutes at the start of a project to define it with your team.

3 Don’t force fit a milestone.

Just because you want to achieve a milestone, it doesn’t automatically follow that the milestone will be successfully achieved. It is your responsibility to listen to the team’s recommendation and to declare milestone attainment (or not).

Any time a milestone has been forced, I’ve gone into schedule re-evaluation mode with the team. Forced milestones usually catch up with you jeopardizing the entire schedule and quality release.

4 Clearly define deliverables for each milestone.

Although obvious, this is something I make it a point to define at the beginning of a project. Laurence Valent states it eloquently, “milestones must be clearly delineated and identified as critical dates in the project plan, and are used for tracking progress in the weekly reporting and review process.” In addition to deliverables being product backlog (or scope) items, the degree of quality and completeness should be discussed. It isn’t unusual for specific scope to be completed only at a “stubbed out” level in order to free up other engineers to complete their work (and final feature details to take a couple of more project milestones to complete). You just need to be sure that the team understands that a “stubbed first” approach is all they’ll get for this next milestone.

And in the case of Agile projects, milestone goals should be specifically stated at a Sprint Planning meeting and re-communicated with every Daily Scrum.

5 Practice makes perfect.

Using your own project history as case studies, practice re-living milestones that were tough to negotiate and with your management team, role play better ways to handle them. Better yet, let your management team take on specific roles (QA, customer advocate, engineer, and so on). Besides being a lot of fun, you don’t need expensive consultants to facilitate this activity.

A Final Word

What happens if a team simply can't resolve whether a milestone is truly met or not? Rather than let the team members struggle among themselves, you (and your product manager/owner peer) should escalate to management. This approach of getting senior managers involved can have many benefits, especially giving them much-needed visibility as to the trials and tribulations of everyday "team warfare." This strategy also removes some of the pressure you may face with the team. I'll never forget one unpleasant situation where the development team was asked to add a feature that had been intentionally removed at t the beginning of the project.

We were two weeks from beta, and this change would adversely affect the schedule by about two weeks that we couldn't absorb and keep the same release schedule. Simply accommodating the request would be viewed by management as a schedule slip and that wouldn't have been fair to the team.

The management team met with the team and they heard two alternatives:

Stick to the original schedule and add the requested feature in a subsequent product update (the next project).

-- or --

Take a two-week schedule adjustment to accommodate the new feature.

After much discussion, the management team empathized with the team and agreed that the customer benefit was too impactful *not* to include this scope change. Everyone bought in, the schedule was adjusted (not slipped) by two weeks and they took it upon themselves to talk with customers about the schedule change and the added feature they'd receive.

This event had another benefit: it bonded the team with you (the project manager) and with the senior management team.

Bibliography

Navin, Snannon. “How a RACI Matrix Can Help Your Project Succeed?” http://www.cardinalsolutions.com/blogs/requirements/2012/06/how_a_raci_matrixca.html. (Jun 2012).

Valent, Laurence. “The Last Seven Project Fundamentals.” http://www.valantco.com/articles/The-Last-Seven-Project-Fundamentals.html. (Jan 24, 2011).

Whitaker, Ken. *Managing Software Maniacs: Finding, Managing, and Rewarding a Winning Development Team*. New York: John Wiley & Sons, 1994.

Bio

Ken Whitaker of Leading Software Maniacs® (LSM) has more than twenty-five years of software development executive leadership and training experience in a variety of technology roles and industries. He has led commercial software teams at Software Publishing (remember Harvard Graphics?), Data General, embedded systems software companies, and enterprise software suppliers. Ken is an active PMI® member, Project Management Professional (PMP)® certified, and a Certified ScrumMaster (CSM). Sources for LSM’s material come from case studies, personal leadership experience, the PMI *Project Management Book of Knowledge* (*PMBOK® Guide*), and Ken’s leadership books: *Managing Software Maniacs*, *Principles of Software Development Leadership*, and *I’m Not God, I’m Just a Project Manager*.

PM University, http://www.pmuniversity.com, is a new addition to Leading Software Maniacs online, eLearning curriculum focused on pragmatic project management and software leadership courses. Ken is also the creator of PM Chalkboard™, http://www.pmchalkboard.com—the fastest way to learn basic project management principles with entertaining, no cost tutorial videos. Ken also is a frequent guest writer for http://www.projectmanagement.com (formerly, http://www.gantthead.com).

To help project managers and anyone working digital projects and software development, Ken has developed the Spresso™ project versioning software, http://www.VersionItWithSpresso.com, (available for both Windows or OS X).

Leading Software Maniacs is proudly associated with: