

Scheduling Agile Delivery Dates White Paper

| | |
|-----------|---|
| Date: | Feb 28, 2010 |
| Title: | Scheduling Agile Delivery Dates |
| Code: | WSAD |
| Abstract: | Software development projects need a pragmatic method of scheduling and cost estimating even when the process framework methodology is based on agile (iterative) concepts. |
| Version: | 1.0 (2009-11-18) Original (excerpt from Leading Software Maniac's <i>Deliver Projects On Time, Every Time</i> seminar) |
| | 1.1 (2009-11-30) Updated to reflect the corresponding online whitepaper presentation |
| | 1.2 (2009-12-23) Updated to include an additional ROM diagram |
| | 2.0 (2010-02-28) Posted on LSM's web site |

Overview

Just because you've proclaimed yourself an *agile shop*, do you think management (and the Finance department) will allow your team to get away with no real estimate for completion? Scrum, as in most agile processes, takes the approach that cost (budget) and time (schedule) are *fixed* and it is the scope (features) that are *variable*. However, you'll probably hear from a team "because we now use Scrum, we'll let you know when we're done as we iterate through!" in response to the need for release dates.

So what do you tell management?

In this document, we'll examine a pragmatic way to derive a schedule (and subsequently a project cost) for a process framework that is inherently flexible, but still must plan for a release date. The approach covered in this whitepaper will hopefully satisfy both your management and your team's desire for practical schedule planning.

Background

Using Scrum as an example agile framework, a project's scope is broken down into prioritized cycles of mini-feature deliveries. This approach encourages "learning and discovery" and quick successes. *If* the development team does encounter failure, you'll know it quickly rather than taking months to eventually find out (as what might happen with a waterfall type of approach). As with most agile frameworks, each cycle is actually a fixed iteration of time which is typically 30 days each (the trend is becoming more like two weeks). In agile, this period of time is called a *timebox*.

To avoid the impression (and sometimes the desire by the development team) to keep the number of iterations a mystery ("we'll be done *when* we're done!") there needs to be a technique to estimate just how long a project should take.

How to Schedule Agile Project Release Dates

We'll take this in stages...

During the initial project plan (or Scrum Planning) step, the team identifies the work as **Product Backlog** feature items and, based on the combined effort to deliver and test these items, an estimation of a rough number of iterations is agreed upon by the team. Let's pretend the project team believes that all of the work can be accomplished in about 12 two-week **Sprints**. This is shown in Figure 1.

The Calculated Estimate is basically the summation of the estimated total effort (in hours) for all of the project backlog items to be completed.

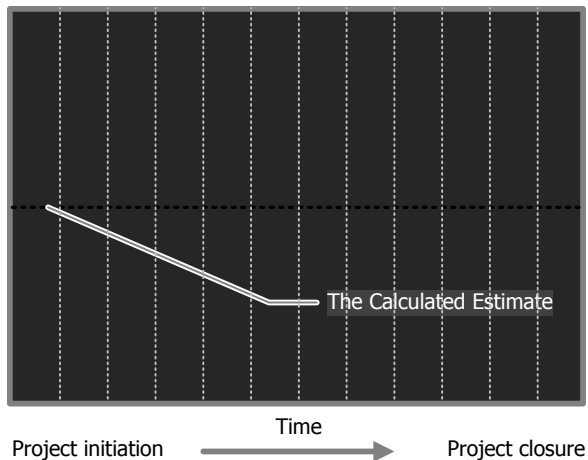


Figure 1: The estimated number of Sprints (iterations) identified during Scrum (project) Planning.

The schedule, however, comes with a fair degree of uncertainty until the team has the opportunity to examine the work details. As a result of this recognized inaccuracy, the overall schedule should be viewed as a **Rough Order of Magnitude (ROM)** estimate in the range of +50% to -50% (from the rough calculated estimate). This is shown in Figure 2:

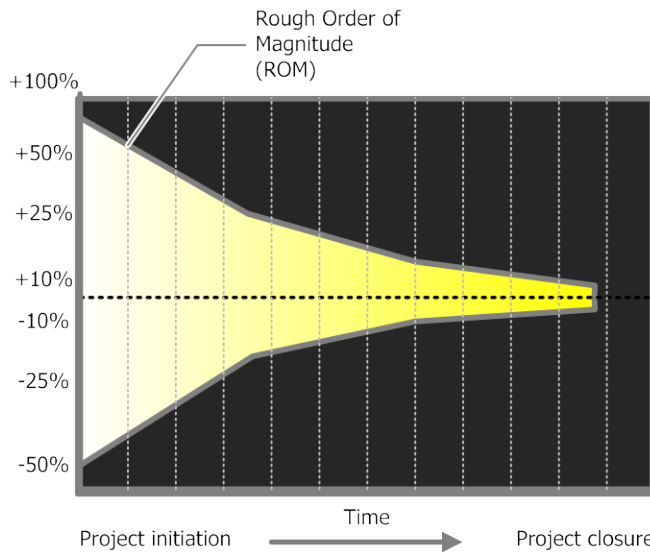


Figure 2: Using Rough Order of Magnitude estimating technique

According to the “Estimate Costs” process (Project Cost Management knowledge area of the *PMBOK® Guide*), as a project progresses so will the accuracy of the remaining work and it is hoped that this will result in a more definitive estimate in the +10% to -10% range (shown in Figure 3):

This schedule refinement technique is called the **Cone of Uncertainty** by Steve McConnell or the **Estimate-Convergence** graph (among other terms).

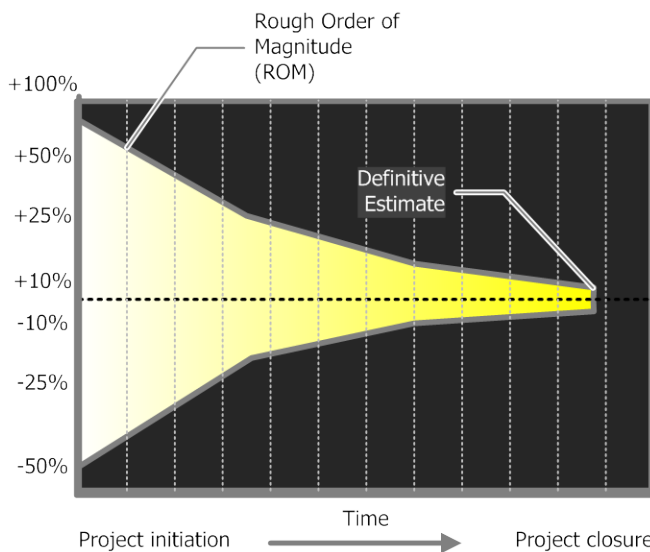


Figure 3: Fine-tuning the estimate with a more accurate, definitive schedule

In order to get the team thinking in terms of a release schedule, it is absolutely consistent with agile projects to consider a schedule as a *schedule range*. (In fact, thinking of any schedule as a worst-case and best-case range is always a good idea.) As long as the team prioritizes *Product Backlog* (feature) items based on customer/business-value first, the team may decide that there could be a release as early as Sprint 8. Thus, the team can represent the schedule as a range (Figure 4):

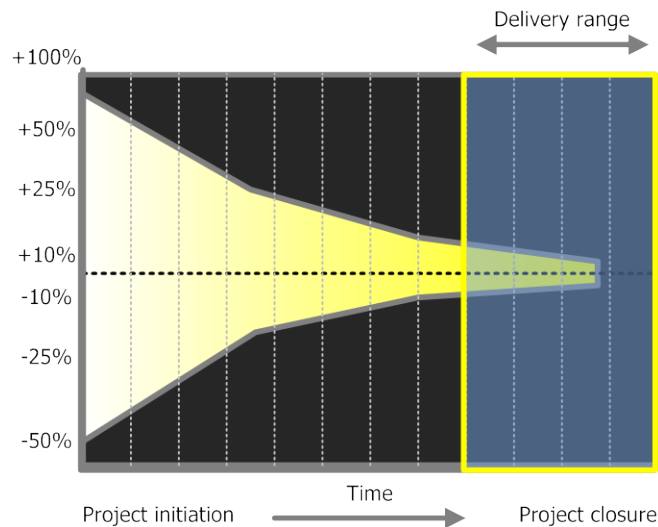


Figure 4: Identifying a range for release (product delivery)

What should executive management (and the team) plan for?

Taking the approach that a full release is estimated to take place between Sprints 8 and 12, the team should plan for the project to take up to the full 12 Sprints and budget accordingly. Finance, Sales, and executive management can plan to see the product release after Sprint 12 and the Marketing and Product Management organization can start their release planning before this delivery range (say, at Sprint 5).

Hey, wait a minute!

You *could* say that this is nothing more than a form of *sandbagging* – but, in reality, it is simply being practical. Still, the team must be focused with the intent of releasing completed customer value (in other words, finished product) as early as possible (Sprint 8). A schedule range gives stakeholders the flexibility of best-case and worst-case planning and hopefully sets the team up to win!

Note: There is additional benefit with taking an agile approach. Agile (Scrum, specifically) has an advantage over a more traditional waterfall approach by avoiding surprises at the tail-end of a project when integration and test takes place late in the development cycle. This approach of reducing risk up front, can add credibility to the schedule which subsequently impacts cost.

Will the Finance Department Accept This Scheduling Technique?

Taking this concept a little bit further, your Finance department may need to plan when a project is to be considered *Technically Feasible*. Although this milestone on a project's life can be open for interpretation, *FASB* (Financial Accounting Standards Board) statement number 86 enables a company to spread out (capitalize) costs over the planned lifetime of a project once technical feasibility has been reached.

More information about FASB 86 can be found in Chapter 2 of *Principles of Software Development Leadership*. (You'll become your CFO's best friend if you know all about FASB 86!)

During the Scrum Planning step, you may wish to identify a bought-in schedule range that once a certain number of Sprint iterations have taken place, the risk of *not* delivering a final product will be minimal (see Figure 5).

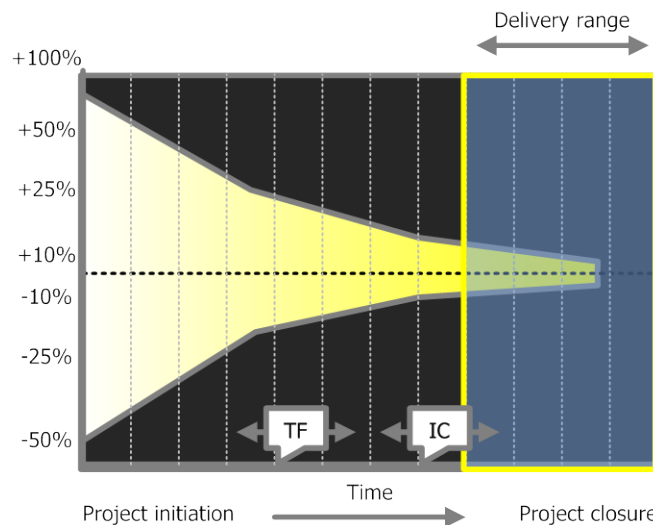


Figure 5: Estimating when technical feasibility will be reached

Note: In order to satisfy the business side of the company, you may need to plan for a range of time when Technical Feasibility (TF) will be achieved along with Implementation Complete (IC).

A Final Word

There is a benefit for *incrementally* building products and setting expectations as *schedule ranges*. Here is some good advice:

“You’ll *rarely be remembered* for missing a feature ...

But you’ll *never be forgotten* for missing a schedule.”

Since the PMBOK® Guide faithfully believes that 90% of your time should be devoted to communication, don’t forget to make sure that all stakeholders understand how you are scheduling projects and that you are consistent in your schedule planning with *all* stakeholder.

Bibliography

- Cohn, Mike. *Agile Estimating and Planning*. Upper Saddle River, NJ: Pearson Education, 2006.
- Larman, Craig. *Agile and Iterative Development: A Manager's Guide*. Boston: Pearson Education, 2004.
- McConnell, Steve. *Rapid Development: Taming Wild Software Schedules*. Redmond, WA: Microsoft Press, 1996.
- McConnell, Steve. *Software Estimation: Demystifying the Black Art*. Redmond, WA: Microsoft Press, 2006.
- Project Management Institute, Inc. *A Guide to the Project Management Body of Knowledge: PMBOK® Guide, 4th Edition*. Newton Square, PA: Project Management Institute, 2008.
- Whitaker, Ken. *Principles of Software Development Leadership: Applying Project Management Principles to Agile Software Development*. Boston: Course Technology PTR, 2009.

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, a Lewis Institute instructor, and a Certified ScrumMaster (CSM). Sources for LSM's presentations come from case studies, personal leadership experience, the PMI *Project Management Book of Knowledge (PMBOK® Guide)*, and Ken's two books: *Managing Software Maniacs* and *Principles of Software Development Leadership*.

Leading Software Maniacs is proudly associated with:



Applying Project Management Principles to Software Development Leadership, Principles of Software Development Leadership, 4Ps, Leading Software Maniacs, Soft-Audit, jus' e'nuff, Nerd Herd Game, the 4Ps logo, the Leading Software Maniacs logo, and the Nerd Herd Game logo are marks of Leading Software Maniacs, LLC.

The Lewis Institute logo is a mark of Lewis Institute, Inc.

PMI, PMP, PMBOK, the PMI logo, and the PMI Registered Education Provider logo are registered marks of the Project Management Institute, Inc.