Selecting the Right Iteration Length

by Mike Cohn • 4 Comments • originally published in InformIT Network on 2006-03-04

The increased popularity of agile software development processes over the years since the signing of the Agile Manifesto has led to increased popularity in iterative development, whether those teams go so far as being agile or not. A key consideration in adopting an iterative process is selecting how long your iterations will be. Common recommendations vary from one-week iterations for extreme programming teams to month-long iterations for Scrum teams. Some teams use even longer iterations, but most teams have settled on an iteration length between a week and a month.

There's no magic length that's right for all teams under all circumstances. Further, the right length for a team on one project may not be the right length for that same team on a different project. Because choosing an appropriate iteration length is an important decision, this article presents the most important factors you should consider when selecting an iteration length for your team.

Overall Length of the Release

Short projects benefit from short iterations. The length of a project's iterations determines the following:

- **How often the software can be shown (in potentially shippable form) to users and customers.** Yes, of course, the software can be shown at its mid-iteration quality level to these audiences, but the software usually is of potentially shippable quality only at the end of an iteration.
- **How often progress can be measured.** It's possible to get a sense of a team's rate of progress during an iteration, but only at the end of an iteration can we accurately measure how much work has truly been completed.
- **How often the team and its customer can adjust project goals.** That is, without introducing “requirements churn” or chaos into the project by allowing major changes during iterations.

If a team is working toward a release that's perhaps only three months away, month-long iterations will give the team only two opportunities to gather end-of-iteration feedback, measure progress, and adjust priorities. In most cases, this number will be insufficient. My general rule is that any project will benefit from having at least four or five such opportunities. If the overall project duration will be four or more months, it might be worth considering monthly or four-week iterations. If the overall release schedule will be shorter, the project will benefit from proportionally shorter iterations.

Amount of Uncertainty

Uncertainty comes in multiple forms and hits a variety of areas:

- Exactly what the customer or users need
- How much the team will complete per iteration
- Technical aspects of the project

The more uncertainty of any type there is, the shorter the iterations should be. When there's a great deal of uncertainty about the work to be done or the product to be built, short iterations allow more frequent opportunities for the team to measure its rate of progress and more opportunities to get feedback from stakeholders, customers, and users

How Long Priorities Can Remain Unchanged

Once a development team commits to completing a specific set of features in an iteration, it's important that they not be redirected from that goal, so people outside the team shouldn't change the team's priorities during an iteration. Therefore, the length of time that priorities can go unchanged is a factor in selecting the iteration length.

A key consideration is how long it takes a good new idea to be turned into working software. Consider the case of a team using four-week iterations. If we assume that new ideas are equally likely to occur at any time during an iteration, then on average a new idea can be said to occur in the middle of the iteration. That new idea will be prioritized into the next iteration, which starts in two weeks. It will take another four weeks (a full iteration) before the new idea shows up as potentially shippable, working software. This means that there is six weeks from new idea to shippable software, as shown in Figure 1. The key point to remember from this example is that the time from new idea to working software will be an average of 1Â½ times the length of the team's iterations.

Changes must be prioritized before they can be worked on.

Image not found or type unknown

On average, a change is delivered 1Â½ iterations after it's identified.

The Overhead of Iterating

Costs are associated with each iteration. For example, each iteration must be fully regression tested. If this is costly (usually in terms of time), the team may prefer longer, four-week iterations. Naturally, one of the goals for your team should be to reduce—or nearly eliminate—the overhead associated with iterating. But especially during a team's early iterations, this cost can be significant and will influence the decision about the best iteration length.

How Soon a Feeling of Urgency Is Established

The intensity level varies quite dramatically on a twelve-month, non-iterative, waterfall-process

project. At the start of the project, when team members feel no stress about the deadline, they may take leisurely two-hour lunches. During the thirteenth month of the planned twelve-month project, they're already into their second or third month of working nights and weekends. As my colleague Niels Malotaux points out, “As long as the end date of a project is far in the future, we don't feel any pressure and work leisurely. When the pressure of the finish date becomes tangible, we start working harder.”

Even with four-week iterations, the end date is never very far in the future. But it's sufficiently far away that many teams will feel tangibly less stress during their first week than during the fourth and final week of an iteration. The goal is to select an iteration length that evens out the pressure the team feels. The point is not to put the team under more pressure. (“You will deliver today!”) Rather, the point is to take the total amount of stress the team would normally feel and distribute it more evenly across a suitably long iteration.

Project Example

The seven-person Napa team was working on a client/server desktop application. The application would be used by 600 employees of the company; it wouldn't be sold or used outside the company. Users were located in three cities, one of which was also home to the entire development team. The idea for the product began as a technology update of an existing system that had become expensive to maintain. However, due to changes in the company's core business, a great deal of new functionality was planned in the project as well. The project had been estimated to take thirteen months, but the company's rapid growth was creating pressure for an earlier release, even if it included only a subset of the desired functionality.

For the Napa project, the team chose four-week iterations. We knew that the project would last at least six months, so even four-week iterations gave us plenty of opportunities to bring the software to a potentially releasable state so that we could put it in the hands of real users. The project had a fair but not excessive amount of requirements and technology uncertainty. The developers were all experienced in the technologies being used (C++ and Oracle). And even though the new application would have features taking it well beyond what the current application did, we had the current application as a basic model of what was needed.

The project team was collocated with many of the intended users of the system. Most of the

users were eager to participate in discussions about what would become their new system. However, the company was growing so quickly that access to these users was partially restricted. We had to be careful not to use too much of their time. Four-week iterations worked well in this situation. Showing them new versions every two weeks would have been too much in this environment. By making a new version available every four weeks, we were able to get more of the users to experiment with the software in a sandbox we had set up for that purpose.

As this was an entirely new application, there was very little overhead of iterating, so that wasn't a factor in the decision. This project was critical to the continued success of the company and had extremely high visibility from the CEO down. For the most part, we were able to establish and maintain priorities for four weeks. Even with four weeks, however, a sense of urgency was maintained because the team remained aware of the need to get an initial release into users' hands as quickly as possible.

Making a Decision

One of the main goals in selecting an iteration length is finding one that encourages everyone to work at a consistent pace throughout the iteration. If the iteration is too long, there's a natural tendency to relax a bit at the start, which leads to panic and longer hours at the end of the iteration. Strive to find an iteration duration that smoothes out these variations.
“One of the main goals in selecting an iteration length is finding one that encourages everyone to work at a consistent pace throughout the iteration.”

Having experimented with a variety of iteration lengths, my general preference is two weeks. One-week iterations (or anything shorter) can be very hectic and stressful. The next deadline is never more than four days away. Extremely short iterations leave no time for recovery if a team member is out sick or if anything goes wrong. Unless a project already has fully automated tests for all parts of the system, I don't often recommend starting with one-week iterations.

A four-week iteration, on the other hand, begins to feel like an eternity after having worked in one- and two-week iterations. With four-week iterations, I find that the team often has time to investigate and pursue more creative solutions than they may have time for with shorter iterations. An experienced agile team working on a highly exploratory phase of a project may benefit from a four-week iteration. However, four-week iterations have a feeling of very distinct

beginnings, middles, and ends. I don't like how different the relaxed beginning feels from the more frantic end.

I find two-week iterations to be ideal. The overhead for planning and testing is much more manageable when amortized across two weeks. The first week of a two-week iteration may feel different from the second week, but the difference is not as dramatic as on a four-week iteration. Additionally, with sufficient practice most organizations can learn to set and not change priorities for two weeks, whereas doing so for four weeks can be very difficult.

As a final bit of advice, once you determine the appropriate iteration length, stick with it. Teams benefit greatly from having a rhythm to their projects. Any regular iteration length can provide this rhythm. This doesn't mean that you can't experiment with a different length, but avoid bouncing among different lengths without good reason.

Posted: March 4, 2006

Tagged: teams, programming, testing, customers, transitioning to agile, agile software development

---