

## Agile Scrum Sprint Length: What's Right for You?

### Executive Summary

When a team begins a project using the Scrum approach, one of the first questions that often arises is, "how long should our sprints be?" After the collective team understands what a sprint is and how Scrum works, several discussions on the topic usually ensue.

In this white paper, we'll review the purpose of the Scrum timebox, the reason why two weeks has become the industry standard and some considerations for when another timebox may be appropriate for your particular situation.

### Timebox Duration Considerations

Teams new to Scrum struggle with questions of duration and expectations of what can be accomplished in a Scrum sprint. Typical questions often arise: "How long should it be? How can you expect me to get all that done in that short amount of time? How do we know we've chosen the right length?"

You will hear Scrum experts say that a sprint can vary from one to four weeks, depending upon the project and the team members involved. What's appropriate for your project?

In Scrum development, it's common for new teams to pick a timebox at the longer end of the scale because they don't understand how they could

perform design, development and testing within a shorter period and produce a quality product.

Their thinking is, if they have more time, they will be able to accomplish what they commit to. For teams new to Scrum, there might be reasons for choosing a longer duration. Perhaps they would like some padding, or they lack confidence in their ability to deliver in a shorter amount of time. Perhaps they are not sufficiently insulated from the enterprise and have many dependencies on outside teams that cannot respond quickly. ("The DBAs have a 36-hour SLA, so we'll need to wait...")

Whatever their reasoning, let's review the variables and consequences driving the decision.

### The Measuring Checkpoint

Sprints in Scrum are unchangeable timeboxes that provide the team with a "measuring checkpoint" for gathering information about progress and making adjustments to scope, staffing, forecasts, etc. With this purpose, the more frequent the checkpoint, the more often metrics are gathered and the faster the cycle time for the team to inspect and make adjustments, leading to a more efficient process more quickly (e.g., do they need a DBA to join the team?). Based on what the team has accomplished, how should the release plan (the forecast) be updated? What can management do to help the team achieve



a faster velocity? Note that this checkpoint has nothing to do with the type of project or the amount of work the team chooses to commit to. Also, for a large company's projects where a project management office must be included, this provides the PMO and leadership with business intelligence data from which they can help the team accomplish its goals.

#### The Sprint Review (a.k.a. Demo)

The sprint review meeting provides an opportunity for the team to hear feedback from product owners, peers and anyone else who attends. The more often a sprint review occurs, the more often feedback can be provided, incrementally improving the product. If this occurs on a shorter cycle time, the cumulative effort of the small improvements resulting from a series of frequently held reviews will yield a better product than that of a longer cycle time. In fact, late in a project, the team may wish to consider a move to one-week sprints to increase the opportunities for feedback as it prepares for a production release.

#### The Retrospective

The retrospective is a key aspect of the inspect-and-adapt Scrum cycle. The more often a team pauses to consider how to streamline its process, the sooner it will identify process issues, attempt to address them and reach terminal velocity. Frequent process improvement meetings also allow for a team to provide feedback to management on productivity drags, such as spending time creating status reports that no one will use or creating a business requirements document that no one will read. Not unlike the fixed political timebox of U.S. elections, the retrospective may be used as an outlet for individuals' process frustrations, not necessarily to vote anyone out of office, but to allow people to voice their opinion.

#### Sprint Planning

The perception that moving to longer duration sprints will reduce pressure and provide breathing

room without a cost is a "false truth." Indeed, it may provide breathing room at the cost of team velocity. Consider the time change during Daylight Saving Time, when people actually have one day a year with 25 hours. How do they spend their "extra" hour? Some may choose to work more, others may choose to sleep or rest more, and others may choose to do more. The point of this simple analogy is that people will adjust to fill whatever time they have been given.

We postulate that the smaller the sprint duration, the faster the engine can run. Smaller durations squeeze out the fat and can only run lean. Consider that if the team knows it has up to three weeks to perform a task, it may spend time researching a better solution than its "first thought" design. However, if the team only has one week, it must quickly implement what can be accomplished in the sprint.

#### The Commitment

An argument for shorter duration sprints is the issue of how the team handles the absence of a key team member. Regardless of the reason for the absence, the general rule is that the team acts as a unit and must pick up the slack to meet its commitment. This places a significant burden on the remaining team members for the duration of the sprint once the absence begins. In longer duration sprints, the team has committed substantially more points than a shorter duration sprint, and if a person is out for a portion of that longer sprint, the remaining team members must either carry the load longer before they can resize velocity or de-scope items to which they've committed. We've found that, in shorter duration sprints, when a team member is absent, teams are more likely to complete the sprint with the original scope than de-scope it. The reason: Teams can "endure pain" for a few days.

#### The Unbroken Think-Stream

One consideration for longer duration sprints is that software development is, to some degree, more art than science, and it requires a level of creativity that cannot be rushed. Certainly, it seems obvious that if the team performs one-week sprints, it has only three actual work days – the first day is spent planning and designing, and the last day is spent on the review and retrospective. On longer sprint durations, say four weeks, the team has 18 unbroken days for the team think-stream. They have time to dream, be creative, identify high-risk design issues and address them,

Not unlike the fixed political timebox of U.S. elections, the retrospective may be used as an outlet for individuals' process frustrations, not necessarily to vote anyone out of office, but to allow people to voice their opinion.

Smaller durations squeeze out the fat and can only run lean.

toss out their first or even second version of an idea and restart again and again before bringing their ultimate idea to the sprint review. This degree of creativity and the pursuit of perfection are not to be found in deadline-driven software development.

### Human Nature and Reality

The boots-on-the-ground reality is that the same thing is going to happen at the end of the three-week iteration that happens at the end of a one- or two-week iteration: Time allocated for testing and preparing for the sprint review will be squeezed. What we've found is that it's more a matter of the team adjusting what it can do during the iteration duration and setting expectations appropriately among themselves than the alternative, which is trying to lengthen the iteration so the team can finish the stories they committed to. The "better" solution to this point is to reduce the amount committed to, break the stories into smaller tasks, pick up more stories opportunistically, deliver functionality earlier in the iteration and allocate time to test and prepare for the sprint review from the start.

### Procrastination

One aspect of human nature is for people to postpone and procrastinate on work that they know they need to do. While this must have a scientific name, let's affectionately call it "college term-paper" thinking. Why? The stereotypical college student who is assigned a research paper at the start of a semester will think he doesn't need to start on it right away; the end of the semester (or "term") is many weeks away. He does other things and comes back to it about three weeks before it's due, only to realize that he is going to need more hours than are left to produce a good paper. His procrastination will result in a lower quality paper, with less content or depth on the topic assigned.

The same may happen with a Scrum team with a longer duration sprint. It may not "feel" like a sprint if the duration is four weeks. Instead of diligently using the unbroken time for a think-stream and following a rapid creative-destruction cycle, the team slowly starts on the stories, building on its previous sprints' work but realizing with only a few days left that what it's built is not adequate.

### Visibility into Team Challenges

Professional developers are smart translators that like to solve puzzles. In the mindset of a stereotypical developer, when a challenge is

encountered in the course of developing a component, they put all their energy into solving it. In this scenario, one possible path has been taken many times by developers. The developer becomes consumed by the challenge and doesn't report it as a block to her Scrum master; after all, she's smart and, in the past, has solved more "hairy" challenges than this one.

The catch is that, in a short-duration sprint cycle, the challenge must become visible to the Scrum master. This is against the developer's nature – she doesn't want to bother the Scrum master with a "non-issue." So, what happens? Often, the developer works to solve the challenge right up to the deadline but fails to finish it per the acceptance criteria, and the product owner is surprised that the functionality was not completed in the sprint review. There is a danger here; if this behavior occurs in the context of a longer sprint duration, the opportunity cost will be quantified in the form of lower velocity.

### The Story Creation Debacle

Another consideration for longer duration sprints is the time it takes to flesh-out an epic into a family of fully developed stories and document the subtasks for each. The reality is that some topics require lots of time from the subject matter experts and may need several "reviews" with the product owner, his peers or other stakeholders before the acceptance criteria for a family of stories is ready for developers and testers to start translating that story into functionality.

This is where we need to recognize that it will take time to draft, socialize and refine stories as the team moves through the story creation process, and we should forecast upfront that a particular family of stories will need more than one sprint to complete. For some stories, three weeks won't be enough. In previous projects, we've found that some story families can take up to eight weeks to produce because of the subject matter experts and/or the complex nature of the topic. With that said, it is not wise to move the sprint review, retrospective and "metrics

**Another consideration for longer duration sprints is the time it takes to flesh-out an epic into a family of fully developed stories and document the subtasks for each.**

**A better approach is to extend the time to create that family of stories across sprints and provide updates at the end of each sprint as to where the group is in the process.**

gathering checkpoint” just because a “story author” team member needs more time to finish a family of stories. A better approach is to extend the time to create that family of stories across sprints and provide updates at the end of each sprint as to where the group is in the process.

Another approach is to establish a checkpoint with one or more business “peer reviewers” that would verify that a story is ready for development. This validation step could also act as a measurement checkpoint for the rate of story creation, which would provide one more data point for management and the team to understand its velocity and productivity.

### Some Metrics to Consider

Let’s assume for a moment that your Scrum team is struggling through two-week sprints but is able to produce 100 story points of velocity for each sprint. They are struggling because of pressure to complete the stories to which they committed, but they’re not working weekends to complete them. However, the testing activities are squeezed, and the team doesn’t feel it has the proper time to prepare for the sprint review. At one sprint retrospective, a member proposes that the team switch to three-week sprints to reduce the pressure, perform appropriate testing and prepare for the sprint review. The real issue is that the team is committing to deliver too many stories in one sprint, and they need to adjust the number downwards. However, let’s consider this proposal from a simple metrics perspective.

We suggest that moving to three-week sprints does not actually solve the underlying issue. Our rationale: It is false to extrapolate that the team will now average 150 points in the new three-week sprints. Typically, the team will drop back a little because it perceives that the pressure is less, say to 130 points in the three-week sprint. The team may now “feel” it has breathing room, but it was bought with a 13% reduction in throughput.

In terms of velocity capital over a six-week period, if the team could average 300 points across three two-week iterations, and if they move to two three-week iterations, the forecast is that the team will only be able to produce 260 points. Over this timeframe, the reduction in velocity capital will result in 40 points of less functionality or fewer defects fixed in the final product. For a team with an average of four story points per

story, that means it will complete 10 fewer stories over a six-week period. That represents a significant difference in functionality. A product owner would probably balk upon hearing that news.

In a baseball analogy, the difference between an average hitter and a great hitter is only one additional hit per week. If a team can do the same – that is, commit to one additional story per week – it will raise the bar and complete the project with a much better product than what an “average” team would have produced.

Should the team keep up the pressure, it may be able to produce the same as it would in the two-week sprints, but it is unlikely it will exceed that. In other words, on this point, there’s little upside to switching to three weeks but a significant potential on the downside.

### Two Weeks: The Standard?

The two-week sprint duration (10 business days) has become the de facto industry standard. Why?

The two-week sprint has the best balance of all the above factors. It allows for a team to have a small amount of creativity with eight days of unbroken think-stream, and it provides a near-term deadline that kills procrastination and forces developers’ challenges to the surface more quickly. Moreover, this approach is often enough to gather feedback from the sprint review and reflect on the process. Finally, it sets a pace for measuring checkpoints twice a month (leadership usually likes that), keeps the pressure up and “feels” like a sprint.

What’s right for your project? That’s a decision you’ll need to make. Perhaps you are in a situation where your team members lack experience or a key skill in a particular area, or your team is struggling with breaking down the stories into tasks, and you conclude that two weeks is just too short. (For a summary of various sprint lengths and their challenges, see Figure 1.)

The key with longer duration sprints is making them “feel” like a sprint by keeping up the pressure; keeping them lean; making sure the team has challenged themselves; avoiding the pitfalls of procrastination, lack of visibility into developers’ challenges and the absence of team members; and allowing story creation to cross sprint boundaries.

## Sprint Length Comparison

Challenges	1 week	2 weeks	3 weeks	4 weeks
In a four-month project, "ceremonial days" for sprint planning, review and retrospective	32 days	16 days	11 days	8 days
In a four-month project, minimum number of days spent testing	~16 days	~8 days	~8 days	~12 days
Impact on other team members when a member falls absent in the middle of the sprint – length of the "endurance test"	A couple of days	Several days	More than a week	A couple of weeks
Unbroken thinking time, allowing for the team to address challenges as they arise	3 days	7 days	12 days	16 days
Visibility into team challenges	Best	Second best	Fair	Poor
Likelihood of falling into the "procrastination trap"	None	Low	Medium	High
Adaptation to uncertainty in story creation	Many stories cross sprint boundary	Some stories cross sprint boundary	Few stories cross sprint boundary	Rare for stories to cross sprint boundary
Application feedback	Best	Better	Good	Fair
Process improvement cycle	Fastest	Fast	Frequent	Slow
Adaptation to F500 enterprises that still operate in traditional models	Unsustainable without isolation strategy	Difficult due to many escalations required	Escalations required but less frequently	Best; few escalations required

Figure 1

The challenge with shorter duration sprints is breaking the stories into small tasks and reassembling them into a unit of work that will be meaningful to the product owner. Consider this: In a one-week sprint, the team will have three productive days to design, build, test and polish a component. Given the uncertainty in software development, this doesn't allow for much time for recovery when the unexpected occurs, causing velocity to appear uneven across sprints. Also if the increment produces too small a delta for the product owner to notice the update or change from the last sprint review, then your sprints may be too short.

### Sprinting Ahead

Consider carefully for what purpose you might deviate from what has become the two-week

standard. Note that there is a series of reasons why two weeks has become the default.

You should ask the question, "if we change to a different length sprint, what problem does that solve?" Are there underlying problems, like personality or behavioral issues that are manifesting themselves in other ways that would be best addressed in another way than changing the sprint duration? Generally speaking, it is better to set the cadence and have the team adjust to it rather than the other way around.

No matter what sprint duration your team chooses, make sure everyone buys in, then run it as lean as you can, and keep in mind the considerations raised in this paper.

### About the Author

*Brian Haughton is a Senior Manager in Cognizant's Advanced Solutions. He is a full-time Agile Scrum Coach and has both CSM and CSP certifications from the Scrum Alliance. His background includes many years at a Big-5 consulting firm, with a specific focus on systems integration and content management. Brian previously worked at major online media and logistics companies. He received a Masters in Mathematics from the University of Mississippi and a bachelor's degree from Mississippi College. He can be reached at Brian.Haughton@cognizant.com.*



---

## About Cognizant

Cognizant (Nasdaq: CTSH) is a leading provider of information technology, consulting, and business process outsourcing services, dedicated to helping the world's leading companies build stronger businesses. Headquartered in Teaneck, New Jersey (U.S.), Cognizant combines a passion for client satisfaction, technology innovation, deep industry and business process expertise, and a global, collaborative workforce that embodies the future of work. With over 50 delivery centers worldwide and approximately 104,000 employees as of December 31, 2010, Cognizant is a member of the NASDAQ-100, the S&P 500, the Forbes Global 2000, and the Fortune 1000 and is ranked among the top performing and fastest growing companies in the world.

Visit us online at [www.cognizant.com](http://www.cognizant.com) for more information.

---



### World Headquarters

500 Frank W. Burr Blvd.  
Teaneck, NJ 07666 USA  
Phone: +1 201 801 0233  
Fax: +1 201 801 0243  
Toll Free: +1 888 937 3277  
Email: [inquiry@cognizant.com](mailto:inquiry@cognizant.com)

### European Headquarters

Haymarket House  
28-29 Haymarket  
London SW1Y 4SP UK  
Phone: +44 (0) 20 7321 4888  
Fax: +44 (0) 20 7321 4890  
Email: [infouk@cognizant.com](mailto:infouk@cognizant.com)

### India Operations Headquarters

#5/535, Old Mahabalipuram Road  
Okkiyam Pettai, Thoraipakkam  
Chennai, 600 096 India  
Phone: +91 (0) 44 4209 6000  
Fax: +91 (0) 44 4209 6060  
Email: [inquiryindia@cognizant.com](mailto:inquiryindia@cognizant.com)

---