Situational Scrum Mastering

Leading an Agile Team

by Mike Cohn

MOUNTAIN GOAT SOFTWARE

A few years ago, after I had hit a number of golf balls into a lake, the friend I was golfing with offered me some advice. Now my friend isn't a better a golfer than I am, but that didn't stop him from interjecting his opinion. "The problem," he said, "is that you need to hit the ball farther." He might as well have told me that the problem was that I was taking too many shots to get the ball in the cup. Of course I needed to achieve more distance. But how?

Similarly, you've probably been told that agile project management—whether performed by a Scrum Master, coach, or agile project manager—is much more about *leading* a team than *managing* a team. However, instructing a Scrum Master to "lead instead of manage" is hardly informative guidance. It's the equivalent of telling a golfer he needs to hit the ball farther. It is valid guidance, but it isn't actionable.

A better way to understand how Scrum Masters can best lead teams that are trying to become agile is to first understand what the team needs in terms of leadership, and then look at four different ways to provide that leadership.

This starts with a look at Paul Hersey's situational leadership model[1]. Hersey's model describes four levels of readiness for teams who are attempting to adopt new ways of working, as summarized in Figure 1.

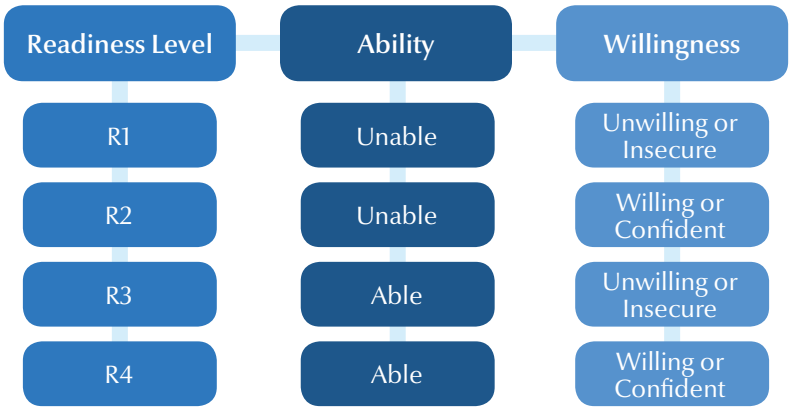

Figure 1. The four readiness levels of the situational leadership model.

1 Paul Hersey, Kenneth H. Blanchard, and Dewey E. *Johnson. Management of Organizational Behavior, 8th ed.* (Englewood Cliffs, N.J.: Prentice Hall, 2001).

According to Hersey, leaders need to adapt their behavior to fit a team's readiness level. In other words, a Scrum Master working with an able and willing team will lead her team in a different manner than someone working with an unable, unwilling team. Hersey suggests that good leaders adjust their leadership behaviors along two dimensions: task behavior and relationship behavior.

A leader's task behavior is the extent to which the leader directs the work of a team. A leader's relationship behavior is the extent to which she leads by using her relationship with the team. Together, these behaviors make up four distinct leadership styles: **Telling, Selling, Participating, and Delegating**. The "right" leadership style to use, then, is the one that best matches the team's readiness level.

Graphically, this is shown in Figure 2. In Figure 2, the leader's Task Behavior is plotted along the x-axis and her Relationship Behavior is plotted along the y-axis. Figure 2 is then divided into quadrants, with one of the four Readiness Levels plotted into each quadrant. Each quadrant contains a descriptive name for the leadership style most appropriate to teams in that quadrant.

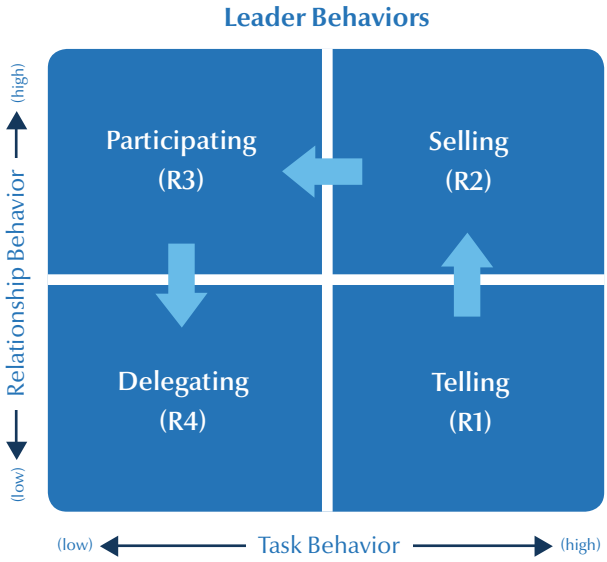

Figure 2. Four styles of leadership for four levels of team readiness.

For example, an R1 team, which is unable and insecure, will need more task guidance from their Scrum Master. The Scrum Master of this team will rely less on two-way communication (a high-relationship behavior) and instead favor one-way communication. Scrum Masters always need strong interpersonal skills but an R1 team (unable and insecure) needs a high level of task guidance so they can gain enough confidence to participate constructively in two-way communication as an R2 team. In contrast, R3 and R4 teams, with their high levels of ability, need a much lower degree of task direction from their Scrum Masters.

The rest of this paper will consider some specific things a Scrum Master should do when leading a team in each of these quadrants.

MOUNTAIN GOAT SOFTWARE

R1: The Telling Scrum Master

Before an unable and unwilling or insecure (R1) team can become truly agile, its members first need to gain confidence. To help them gain the certainty they need, a good Scrum Master will focus more on *telling* team members what to do, rather than on establishing supportive, communicative relationships.

Although this might seem at odds with Scrum's emphasis on self-organizing teams, the reality is that a team in the R1 quadrant is not yet ready to self-organize in an optimal way. Scrum Masters and coaches, however, can lay the groundwork to enable teams in this quadrant to quickly become an R2 (unable but willing or confident) team—a team that is ready to become agile.

An R1 team is not ready for a large project or aggressive goals that require its members to perform at their utmost. This kind of team needs, instead, to start with simple, small victories that will boost confidence.

The Scrum Master of an R1team will need to make some decisions for the team in order to put that team on a winning path.

The Scrum Master of an R1team will need to make some decisions for the team in order to put that team on a winning path. (Yes, I just said that.) These decisions typically involve establishing some basic processes that will help the team begin to experience some success.

First, insist on short sprints of one-to-two weeks. The shorter the sprint, the more frequent is the feedback the team members receive on how well they are executing Scrum. When introducing Scrum to a team, short sprints can be especially helpful. The newly agile team needs to gauge how it's doing. Waiting a month to see the results of a sprint is often too long.

Next, work on foundational agile skills like testing. Explain that quality is everyone's problem and everyone's job. Create a rule that programmers must unit test before they check code in. Tests, for

this type of team, do not need to be automated but they do need to be done.

At the same time, introduce a continuous integration process, or at least a nightly build. Ideally these builds should be accompanied by a simple suite of automated tests, either at the unit or functional level.

Expect that, in the beginning, there may not be many tests and what tests do exist may fail often. In most cases, however, over the course of a few weeks the number of tests will grow and the number of successful builds will increase. A few weeks of receiving the positive feedback of multiple successful builds (and the reduced rework this leads to) can work wonders for the morale—and confidence—of the team.

A few years ago, I was handed a classic R1, unable and insecure team. The team members were almost all COBOL programmers working on PCs. The company had just been acquired and the previous management had repeatedly told the team members they weren't capable of working in newer or more interesting technologies.

The team included some very smart programmers but after a few years of being treated as they had, they'd lost all their confidence and their skills had fallen out of date. After a few months of me providing very specific guidance to the team about what technologies to use, how to work, and so on, confidence was restored. The once-dysfunctional team had advanced to the R2 quadrant.

MOUNTAIN GOAT SOFTWARE

R2: The Selling Scrum Master

In the R2 quadrant, the Scrum Master is working with a low ability team but one that is willing or confident. The goal at this stage of team development is to boost team members' abilities. Although some teams arrive here after passing through the first quadrant and the telling style of leadership, many teams start as R2 teams. Teams in this quadrant have the necessary foundation to become agile, but they aren't there yet. R2 teams require a *selling style* of leadership—one that combines highly directive task behavior and highly supportive relationship behavior.

The Scrum Master of an R2 team has earned some degree of trust and confidence. Now is the time to capitalize on those relationships to help increase the team's agile abilities.

The R2 quadrant is the beginning of being agile.

At this stage, involve team members more in the decision-making process. Some decisions will still need to be made for the team, but whenever possible invite the team to participate.

Focus, too, on helping members improve their skills. Continue to focus on testing but begin to challenge team members to create automated tests. Invest in training to help teams learn the appropriate automated testing tools—many tools have a very immediate payback for the effort invested.

To help knowledge and skills spread through the group, introduce pair programming. I rarely mandate that code be written in pairs but I do ask team members to try it and to figure out for themselves when are the right times to do it.

Throughout all of these initiatives, build on the team's budding trust to deliver the message that it is acceptable—and desirable—to think about the quality of the code. Many team members have heard faster, faster, faster for so long that they struggle to hear the message that if we write high-quality code, the speed will come. Remind them (and

anyone else who is tempted to focus only on speed) that if you get on the highway and floor it to 120 miles per hour, you've got a good chance of getting a ticket or being in an accident. Either way, you won't end up at your destination any faster than if you'd averaged a safe, high-quality 70 miles per hour.

At the R2 stage, involve team members in the decision-making process.

Personally, I never tell a team (especially an R2 team) to go faster. I do, however, often tell them they can learn to write better, higher-quality code through pair programming, automated testing, and refactoring.

In the selling quadrant, continue to emphasize short iterations. R2 teams still benefit from being able to assess their progress every few weeks. Many newly agile teams have trouble adapting to the idea that high-quality software can be created in each sprint. They are used to test phases that follow after coding.

Remind team members that on a Scrum project, coding and testing happen concurrently so that potentially shippable is produced in each iteration. Many teams struggle with this initially, which results in a panic-filled last week where everyone works overtime. Teams eventually figure out a way of working that lets them deliver high-quality software within each sprint, but learning how to do that often takes a handful of sprints (or more). The sprints might as well be short ones so the skill is mastered as quickly as possible.

MOUNTAIN GOAT SOFTWARE

R3: The Participating Scrum Master

As the team improves its abilities, it begins to move out of the R2 quadrant. R3 teams are capable of working in a truly agile manner. As such, in this quadrant the good Scrum Master shifts to a *participating style* of leadership, reducing task direction while increasing relationship-building behaviors.

Instead of making decisions for the team with its input (as in the R2/selling quadrant), encourage the team to make its own decisions. Do this as soon as possible—even before the team recognizes their newfound ability. In the face of these increased responsibilities, teams in the R3 quadrant typically return temporarily to feeling insecure. But, unlike an R1 team, the R3 team is highly skilled and can quickly transition to an R4 team.

With an R3 team, encourage the team to make its own decisions.

To help the team transition into making its own decisions, move completely out of the decision-making process. Be there to encourage and support the team but make it completely clear to the team members that the decision is theirs. Naturally, the team may make some mistakes. But, so what? Team members are continuing to learn and their level of performance is typically so far ahead of where it was at R1 that a few mistakes are a small price to pay.

As the team begins to self-organize and continues to develop its agile skills, the Scrum Master will naturally begin focus less narrowly on the team's performance and more broadly on big-picture factors. The team members will be maniacally focused on the work selected for the current iteration—a horizon no more than one-to-four weeks into the future. Yes, they may have a release plan covering what will likely be the next three or four months of work, but that plan is deliberately short on details. Because they are so focused on the trees of the current iteration, the team members need to trust their Scrum Master to keep an eye on the longer-term forest.

There are a number of things the Scrum Master can do in this regard.

First, look for items that need to be planned further in advance than the current iteration. For example, a team may benefit from a meeting with the company-wide expert on a particular technology. Unfortunately, he works in an office 2,000 miles away. To fit the company's travel guidelines, his ticket must be purchased more than two weeks in advance. Or perhaps in a few sprints the team plans to implement a particular user story that will require a good deal of input from the VP of Sales, the VP of Marketing and the general manager of the division. Coordinating time with those three will take some advance planning.

Next, ensure that the sprint-by-sprint focus on specific user stories is not drifting away from the larger themes that were selected for the release. Because Scrum encourages reassessing the priority of planned work at the start of each sprint, it is possible for the product owner and team to gradually drift away from what was intended for a larger milestone or release.

If this drift is intentional and the result of incorporating learning from prior sprints, it can be a great thing. If, however, it is the result of the team and the product owner losing sight of the forest by focusing too greatly on the trees, then help to realign the project.

Last but not least, make sure the team is always working on the highest-valued work possible. One way to do that is to offer to work with the product owner to prioritize and adjust the product backlog as the project progresses.

R4: The Delegating Scrum Master

Once the team reaches the R4 level of readiness, a good Scrum Master will shift from a participating leadership style to a *delegating* style. At this point, the team needs minimal task guidance. Help when asked, but do not specify how work should be accomplished.

First, go beyond delegating decisions to the team by showing them how to defer decisions as long as possible. When a team defers making a decision, it keeps its options open and can incorporate new knowledge into the eventual decision. We want developers to do this with their designs and their code. On a Scrum project, we ask them to write only the code necessary to implement whatever feature they are working on. We don't ask them to speculatively build in layers of unneeded features that we might need "someday."

Once the team reaches the R4 level of readiness, a good Scrum Master will shift from a participating leadership style to a *delegating* style.

For example, suppose one of our partners wants to send us a weekly transaction file they want loaded into our system. We don't yet know if it will be a comma-delimited, fixed-length, or XML file. Rather than writing an importer that supports all three inputs, we defer the decision. That may mean we don't code any of the importer yet. Or, it may mean that we start coding with the assumption that input data will come from somewhere and we code from that part of the problem on.

Timing is everything in decision making. In the early days of Java, I was managing a project that could have been delivered as a Java client or as a native Windows client. There were pros and cons to each approach. We'd just started the project and "resolve UI platform" was on the open issues list. So, I got the lead developers together and we quickly made a decision.

Of course, we made the *wrong* decision. However, even if we'd chosen the other platform I would still say that we made the wrong decision. The right decision in this case would have been to defer the decision. In the early days of that project there was no reason whatsoever for us to make that decision. Sure, the decision needed

to be made someday, but not in the second week of the project. So let the team make decisions but coach them and advise them to defer decisions until as late as possible.

Second, work on maximizing the team's overall rate of throughput. A traditionally managed project often feels as though it is a race with a definite finish line, perhaps a 10K run or a 50-mile bike race. Scrum, on the other hand, can feel more like a race where you see how far you can run in 24 hours. There's often no finish line on a Scrum project. Instead, the clock runs out and you stop. If you need more features, you start again; otherwise, you're done. This distinction leads to differences in how traditional managers and Scrum teams approach their projects.

A Scrum Master must be more concerned with maximizing the throughput of the team.

The traditional project manager is focused entirely on meeting an end goal, releasing a specific set of functionality by a specific date. Nothing exists beyond that goal. At first this may seem a strong point in favor of traditional project management. However, consider the traditional project manager's likely response when a developer approaches her a month before the deadline with the concern that, while the code in one area currently works, it is brittle and will become a source of problems over the next few months. The traditional project manager will very likely make the short-term decision and take her chances with the brittle code.

A Scrum Master, however, must be more concerned with maximizing the throughput of the team. In any particular case, the team may also have to favor meeting the one-month deadline over refactoring the brittle code. In general, though, the right thing is to protect the overall throughput of the team and refactor the code. The Scrum Master's job is to encourage the team in that direction and protect them against outside objections.

This is just one example and there could be many others. The key distinction here is that the traditional project is managed with the sole goal of meeting a deadline with a specified set of features. The agile project may, of course, have deadlines with promised functionality, but the sustained throughput of the team over the longer term is also important.

MOUNTAIN GOAT SOFTWARE

Putting Situational Scrum Mastering To Use

Leading is the sum of every interaction Scrum Masters have with their teams: what they say, what they don't say, how they say it, how they listen. Being a good leader of a Scrum team doesn't involve yelling "Damn the torpedoes!" Nor does it entail sitting quietly apart from the team and listening to the daily scrum updates. Instead, being a good agile leader—a good Scrum Master—is something that must be learned, often times alongside teams that are learning themselves how to become agile.

Developing the skills to become a Scrum Master and to help a team become agile is not difficult. It does, however, help to possess a mental framework about the readiness of the team and the leadership style one should use for a given team. The situational leadership model provides this framework. As teams gather confidence from success and grow in their abilities, they will need to be led in different ways.

As teams gather confidence from success and grow in their abilities, they will need to be led in different ways.

Teams with limited ability and limited confidence typically need a telling leadership style. When leading R1 teams, boost their confidence by introducing short sprints and setting the team up for a series of small wins. Introduce foundational agile practices such as an increased focus on testing and continuous integration. From these practices the team will gain confidence in their skills and their ability to work in a new, more agile manner.

A team with limited ability who is willing and confident benefits from a selling leadership style. When leading R2 teams, focus on the relationship with the team, rather than just directing tasks. A team of this type just needs time, motivation, and practice to improve their skills.

To create opportunities for learning, emphasize the need to deliver a high-quality product at the end of each sprint. Through working in short iterations and focusing on creating potentially releasable

software at the end of each iteration, team members will naturally look for ways to improve their skills.

Teams that have grown significantly in ability are ready for a more participating leadership style. However, when teams must make decisions for themselves, they tend to become nervous, wondering if their newly-acquired skills are sufficient. Guide R3 teams through this transition by supporting them in their decisions and by focusing your own attention on the longer-term (3-6 month) horizon, allowing the team to devote all of its attention on the current sprint.

Introducing and refining techniques at the right time for a given team is essential.

One way to do this is to ensure that sprints remain focused on the goals of the next release, even as the work of each sprint is reprioritized at its start. At the same time, make sure the company selects high-value work for the team by working with the product owner to ensure the product backlog is properly prioritized.

Some Scrum Masters are lucky enough to work with an R4 team, which is skilled and willing, ready to step up to even greater levels of responsibility. In those cases, adopt a delegating leadership style. On Scrum projects, that involves a focus on maximizing throughput over a horizon exceeding that of one project. It also means helping the team learn to defer decisions so that options remain available as long as possible.

Unlike my golf partner's advice to just "hit the ball farther," this paper has presented specific things a Scrum Master can do to be a more effective agile team leader. While any of the techniques mentioned here could be applied to a team at any level of readiness, I have described them in association with the levels of readiness where I've seen them be the most beneficial. Introducing and refining techniques at the right time for a given team is the best way for a Scrum Master to make progress toward the holy grail of an R4 team and a delegating leadership style.

MOUNTAIN GOAT
SOFTWARE

ABOUT MIKE COHN

As the founder of Mountain Goat Software, Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance development organizations.

Mike ran his first Scrum project in 1994, and has been a vocal proponent of Scrum ever since. With more than 20 years of experience, Mike was previously a technology executive in companies of various sizes, from startups to the Fortune 40.

Today, Mike is one of the industry's most sought-after Certified Scrum Trainers. He is the author of three books on Scrum and agile, and the founder of the agile online training website, FrontRowAgile.com.

He is also a founding member of the Scrum Alliance and Agile Alliance, is a frequent keynote speaker at industry conferences and corporate events, and a thought leader at his popular blog on Scrum and agile.

CONTACT US

MOUNTAIN GOAT
SOFTWARE

1140 US Hwy 287
Suite 400-108
Broomfield, CO 80020
1-888-61-AGILE (24453)

info@mountaingoatsoftware.com
www.mountaingoatsoftware.com