From *Cutter IT Journal*, June 2004

Situational Leadership for Agile Software Development

Not all teams need the same type of leadership. This article describes the need for situational leadership and presents a model for determining the appropriate type of leadership for development teams, especially those moving toward an agile process.

Mike Cohn
Mountain Goat Software

A few years ago, after these teams and managers become agile, it is useful to look at a situational leadership model proposed by Paul Hersey. In this model, leaders adapt their behavior to fit the ability and willingness of the team. Based on ability and willingness, a team is at one of four levels of readiness to adopt new ways of working, as summarized in Figure 1.

Agile project management is much more about providing leadership than providing management to a team. However, instructing a first-time agile project manager to "lead, don't manage your team" is hardly informative. It's the equivalent of telling a golfer he needs more distance. That may indeed be the problem but it doesn't say a thing about how to achieve it.

Providing leadership is about every interaction with a team: what we tell them, what we don't tell them, how we tell them, how we listen. Leading an agile team is not yelling "Damn the torpedoes!" Nor is it sitting quietly apart from the team listening to their status reports. Most teams that are learning to become agile are at the same time being managed by project managers who are learning to manage in agile ways.

To understand how these teams and managers become agile, it is useful to look at a situational leadership model proposed by Paul Hersey.[1] In this model, leaders adapt their behavior to fit the ability and willingness of the team. Based on ability and willingness, a team is at one of four levels of readiness to adopt new ways of working, as summarized in Figure 1.

Readiness Level	Ability	Willingness
R1	Unable	Unwilling or insecure
R2	Unable	Willing or confident
R3	Able	Unwilling or insecure
R4	Able	Willing or confident

Figure 1. The four Readiness Levels of the situational leadership

Clearly, an agile project manager working with an able and willing team will lead her team in a different manner than if she were working with an unable, unwilling team. Hersey suggested that good leaders will adjust their behavior to the team along two dimensions: task behavior and relationship behavior. A leader's task behavior is the degree to which the leader directs the work of a team. A leader's relationship behavior is the degree to which she leads by using her relationship with the team.

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 divided into quadrants with one of the four Readiness Levels plotted into each quadrant. Each quadrant contains a descriptive name (Telling, Selling, Participating, and Delegating) for the leadership style most appropriate to teams in that quadrant.

For example, an R1 team, which is unable and insecure, will need more task guidance from their manager. The manager of this team will rely less on two-way communication (high relationship behavior) and instead favor one-way communication. Managers 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 manager.

The rest of this article will consider some specific things an agile project manager should do when leading a team in each of these quadrants.

APM and Telling

The unable and unwilling or insecure (R1) team first needs to gain confidence. To help them gain confidence, a good manager will focus more on telling them what to do, rather than on establishing supportive, communicative relationships. Many of the agile processes rely on self-organizing teams, or at least self-directing teams. A team in the Telling quadrant cannot become truly agile. But the groundwork can be laid and a team in this quadrant can quickly become an R2 (unable but willing or confident) team.

There are a number of things an agile project manager can do to establish this groundwork and help a team gain the confidence necessary to move in an agile direction. The team needs some basic processes put in place that will help them win small victories. A team in this quadrant is not ready for a "home run project." They need, instead, to start with simple, small victories that will boost their confidence. The manager of this type of team must be forceful and lead the team by making decisions for them. The time will come to lead this team through relationships but for now they need day to day task guidance that will allow them to win small victories. These victories will then, in turn, boost their confidence.

The best way to get these small victories is by introducing short iterations to the team. I used four-week iterations for years but have recently begun to favor two-week iterations because the feedback comes even more often. When introducing a team to an agile process, I think it is especially useful to have short iterations. The

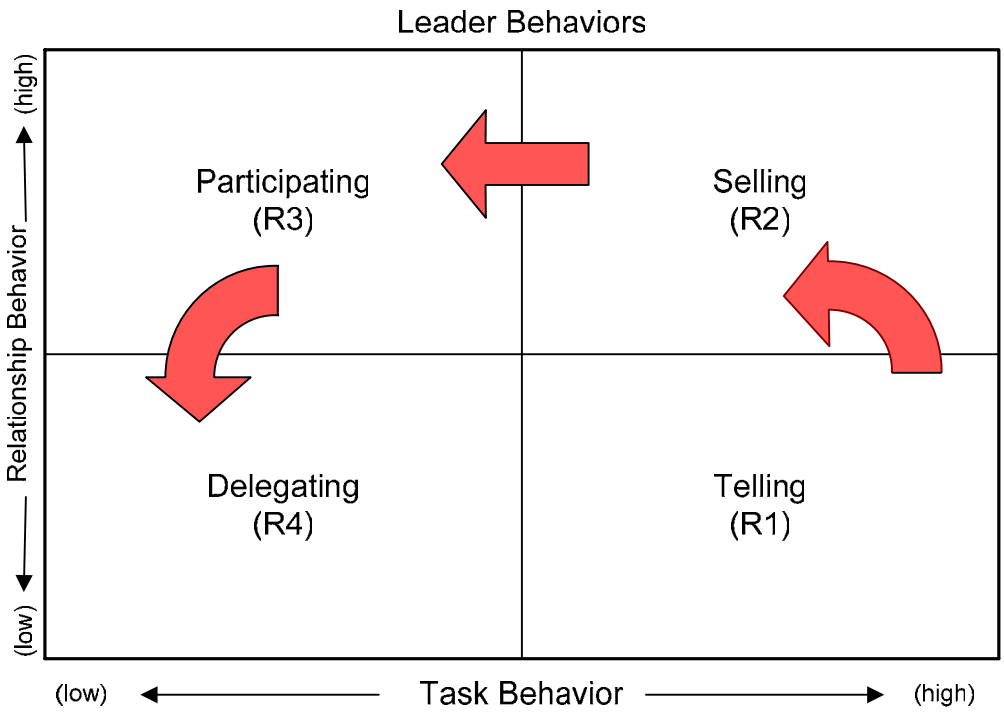

Figure 2. Four styles of leadership for four levels of team readiness.

newly agile team needs to see how they're doing. Waiting a month to see the results of an iteration is often too long.

A few years ago, I was handed a classic R1, unable and insecure team. They were almost all COBOL programmers working on PCs. The company had just been acquired and the previous management had beaten the team up by telling them 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 they'd use, how they'd work, and so on, their confidence was restored and they were an R2 team.

I like to start building the confidence of an R1 team by having them work on foundational agile skills like testing. As the team becomes agile, they will need a solid testing discipline and it works to tell an R1 team that quality is everyone's problem and everyone's job. Programmers are expected to unit test before they check. Tests, for this type of team, do not need to be automated but they need to be done.

I also like to introduce a nightly build or continuous integration process. Ideally the nightly build can be augmented with a simple suite of automated tests, either at the unit or functional level.

When the programmers arrive each morning they are greeted with an email stating whether the nightly build succeeded and the tests passed. At first there may not be many tests and they may fail often. However, over the course of a few weeks, most teams will begin adding tests and making sure the nightly build is successful. A few weeks of receiving the positive feedback of successfully tested nightly builds can work wonders for the morale of the team.

APM and Selling

In this quadrant, the agile project manager is working with a low ability team but one that is willing or confident. Her goal is to boost their abilities. Some teams arrive here after passing through the first quadrant and the Telling style of leadership. However, many teams start here and this quadrant can be thought of as the beginning of becoming agile. Here, the agile project manager is required to exhibit both highly directive task behavior and highly supportive relationship behavior.

The team has developed a level of trust and confidence in their manager. The manager must now capitalize on that to help the team make more of their own decisions. Within the Selling quadrant, the good agile project manager shifts her decision-making style. She may still make the decisions but she now does so with participation from the team.

This is also the time to help the team improve their skills. In this quadrant I again like to emphasize testing but normally shift the focus to the creation of automated tests. I'll spend as much as necessary helping the team learn tools such as JUnit, FIT, and FitNesse, which I find have a very short payback period on the effort invested. This is especially true when the tests are integrated into the nightly or continuous build system. To help knowledge and skills spread through the group, I find that this quadrant is the right time to introduce pair programming. I rarely mandate that code be written in pairs but I stress to the team that I'd like them to try it and figure out for themselves when are the right times to do it.

Most of the R2 teams I've worked with need assurance from their manager that it is acceptable to think about the quality of the code they are writing.

Most of the R2 teams I've worked with need assurance from their manager that it is acceptable to think about the quality of the code they are writing. Many have heard faster, faster, faster for so long that they are hesitant to hear the message that if we write high-quality code now, the speed will come. Personally, I never tell a team (especially an R2 team) to go faster. I often tell them they can write better, higher-quality code. By introducing pair programming, automated testing, and refactoring an R2 team can learn how to do this. A good agile project manager will build on the team's budding trust in her to deliver this quality message. If it's tempting to tell a team to go faster, imagine driving a car on the highway. If you get on the highway and floor it to 120, 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.

In the Selling quadrant, the good agile project manager continues to emphasize short iterations. As their skills improve, an R2 team benefits from being able to assess its progress every few weeks. Many newly agile teams have trouble adapting to the idea that high-quality software is available at the end of each iteration. They are used to test phases that follow after coding. On an agile project, coding and testing happen concurrently and software that could be released (but often isn't for logistical reasons) is produced in each iteration. Many teams struggle with this initially, which results in a panic-filled last week where everyone works 60 hours. Teams eventually figure out a way of working that lets them deliver high-quality software within the iteration but learning how to do that often takes three or four iterations. The iterations might as well be short ones so the skill is mastered as quickly as possible.

APM and Participating

As the team improves its abilities, they begin to move out of the R2 quadrant. In this quadrant, the team can truly say they have become agile. In this quadrant the good agile project manager shifts to a Participating style of leadership. She reduces the amount of task direction she gives the team and instead of making decisions for them with their input (as she did in the R2/Selling quadrant) she now encourages them to make their own decisions. Ideally this is done as soon as possible and often begins even before the team realizes their new abilities. The increased responsibility can make the team insecure. But, unlike an R1 team with a low level of ability, the R3 team is highly skilled.

To help the team transition into making their own decisions, the skilled agile project manager will often shift herself excessively out of the decision-making process. She'll be there to encourage and support the team but will make it completely clear to the team that the decision is theirs. Naturally they'll make a few mistakes. But, so what. They're also learning 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.

The developers on an agile project are manically focused on the work selected for the current iteration. This means their horizon is usually 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 intentionally short on details. Because they are so focused on the trees of the current iteration, the developers need to trust their manager to keep an eye on the longer-term forest.

There are a number of things the project manager can do in this regard. First, there are always some things that need to be planned further in advance than the current iteration. A good agile project manager keeps watch for these. For example, a team may benefit from a meeting with the company-wide expert on Service-Oriented Architectures. 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. Similarly, a good agile project manager may look ahead an iteration or two and realize that when a particular user story is being implemented the team will want to ask lots of questions of the VP of Sales, the VP of Marketing and the general manager of the division. Coordinating time with those three takes some advance planning and the good project manager will get it on their schedules a month in advance.

The developers on an agile project are manically focused on the work selected for the current iteration.

A good agile project manager will also make sure that the iteration-by-iteration focus on specific user stories is not drifting away from the larger themes that were selected for the release. Because agile processes encourage reassessing the priority of planned work at the start of each iteration, it is possible for the customers and developers to gradually drift away from what was intended for a release. If this drift is intentional and the result of incorporating learning from prior iterations, it can be a great thing. If, however, it is the result of the programmers and the customer losing sight of the forest by focusing too greatly on the trees, then a good agile project manager will help realign the project.

Another way in which a good agile project manager can keep an eye on the forest is to make sure the team is always working on the highest-valued work possible. The agile project manager can play a

big role in this by making sure the project's customers prioritize work on the proper basis. Typically, this means some form of ROI or net present value analysis. The way I do this is to sort the user stories that represent an agile project's requirements into various themes. A theme may include anywhere from one large user story (an "epic") to a few dozen smaller stories. A theme is a cohesive set of work that would be valued by the users or purchasers of the software. The customer team estimates the cash flows that are expected to result from the theme and the developers estimate the story point cost of the theme. Based on the number of story points the team completes per iteration, it's easy to determine a cost per story point and complete the analysis.

APM and Delegating

As the team reaches the R4 level of readiness, a good agile project manager will shift from a Participating leadership style to a Delegating style. She offers the team minimal task guidance. She is there to help but not to specify how work is accomplished. The good agile project manager does not just delegate decisions to her team, she also shows them how to defer decisions as long as possible.

When we defer making a decision we keep our options open. This allows us to incorporate new knowledge into the decision. We want developers to do this with their designs and their code. On agile projects 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'll need "someday." For example, suppose one of our partners wants to send us a weekly transaction file he'd like 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. A good agile project manager of an R4 team will delegate decisions to them but will coach them and advise them to defer decisions as long as possible.

A good agile project manager of an R4 team will delegate decisions to the team but will coach them and advise them to defer decisions as long as possible.

As an example, a handful of years ago I was managing a project that could have been delivered through a web browser or through 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 the second week of the project.

The traditional project manager is focused entirely on meeting an end goal.

A second way in which a good agile project manager can help her team in this quadrant is by maximizing their 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. An agile project, on the other hand, feels more like a race where you see how far you can run in 24 hours. There's often no finish line on an agile project. Instead, the clock runs out and you stop. If you need more features, you start again; otherwise, you're done. This difference leads to differences in how traditional and agile project managers approach their projects.

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.

An agile project manager, on the other hand, must be more concerned with maximizing the throughput of her team. In any particular case she may also have to favor meeting the one month deadline over refactoring the brittle code. However, the agile project manager will know that—in general—the right thing is to protect the overall throughput of the team and refactor the code. If there's time to do that she will make that decision. If there isn't time to refactor, the agile project manager will raise the issue to the attention of her customers and involve them in the decision.

An agile project manager must be more concerned with maximizing the throughput of her team.

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 is, of course, managed with deadlines and functionality in mind but it is also managed with the sustained throughput of the team in mind.

Putting This Into Use

Developing agile project management skills and helping a team become agile are not difficult. A project manager who has mastered the traditional project management skills is certainly up to the task. However, it helps to possess a mental framework for how we think about the readiness of the team and the leadership style of the manager. The situational leadership model described in this article provides this framework. As teams gain ability and from that gather confidence, it makes sense that they will need to be managed in different ways.

In this article we've seen that teams with limited ability and limited confidence need a Telling leadership style. The good agile project manager of this team boosts their confidence by introducing short iterations and setting the team up for a series of small wins. Foundational agile practices such as an increased focus on testing and a nightly build are introduced. From these practices the team gains confidence in their skills and their ability to work in a new, agile manner.

A team with limited ability that is willing and confident benefits from a Selling leadership style. With this type of team, the good agile project manager begins to manage through her relationship with the team, rather than just by directing their tasks. These are the first steps toward becoming an agile team. A team of this type needs to improve their skills. They are confident they can do it, now they need time, motivation, and practice. The best thing the manager can do is become strict about the need to deliver a high-quality product at the end of each iteration. By keeping iteration lengths short and by stressing the need for potentially releasable software at the end of each iteration, the manager gives the team the opportunity they need to improve their skills.

As the agile manager shifts into a Participating leadership style, she stops making decisions for the team. This makes many teams nervous as they wonder if their newly-acquired skills are sufficient. However, a good agile project manager guides them through this transition by supporting them in their decisions and by focusing some of her attention on the longer-term (3-6 month) horizon, allowing the team to devote all of its attention on the current iteration. She does this by making sure that iterations remain focused on the goals of the next release, even as the work of each iteration is reprioritized at the start of the iteration. She also makes sure that the company selects high-value work for the team by looking at the ROI of each theme of work.

A good agile project manager of an R4 team will delegate decisions to the team but will coach the and advise them to defer decisions as long as possible.

When lucky enough to work with an R4 team, which is skilled and willing, the project manager uses a Delegating leadership style. She has accustomed her team to accepting great responsibility and the team steps up to that responsibility. With a team at this level, the agile project manager focuses on maximizing their throughput over a horizon exceeding that of one project. She also helps the team by helping them learn to defer decisions so that options remain available as long as possible.

Unlike my golf partner's advice that I needed more distance, this article has presented some very specific things an agile project manager can do. 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 an agile project manager to make progress toward the holy grail of an R4 team and a Delegating leadership style.

References

1. Hersey, Paul, Kenneth H. Blanchard, and Dewey E. Johnson. Management of Organizational Behavior. 8th ed. Prentice Hall, 2001.

Mike Cohn is the founder of Mountain Goat Software, which specializes in training and mentoring companies in agile software development. Mike has over 20 years of experience in various facets of software development, particularly C++ and Java programming and project management. Mike is the author of *User Stories Applied for Agile Software Development* as well as books on Java and C++ programming. He has also written articles for *IEEE Computer, Cutter IT Journal, Software Test and Quality Engineering*, *Better Software*, *Agile Times*, and *C++ Users' Journal*. Mike is a founding member of the Agile Alliance and serves on its Board of Directors. Mike is a Certified ScrumMaster and a member of the IEEE Computer Society and the ACM. He can be reached at mike@mountaingoatsoftware.com.