Transcript: Agile Vital Signs, presented by Damon Poole
Free Webinar on November 11, 2014

Hemant Elhence: Before I go to the topic, let me go through the quick logistics. I'll introduce a topic and then introduce our feature speaker, and then we'll have 45 minutes or so of presentation on this topic of Agile Vital Signs. We will take some questions along the way, which makes sense because of the content that has been so interesting, and the audience feel free to keep sending your questions on the GoToWebinar Q&A panel.

I am your host, Hemant Elhence from Synerzip. I'll keep track of these questions, and then we'll still keep aside another 10-15 minutes at the end for general questions on the topic of Agile Vital Signs.

With that, let me introduce the topic. The topic today is a very good one. We're calling it Agile Vital Signs, but the idea is we have covered the topic of agile adoption a couple of times if not more in these webinar series. After companies have adopted agile and they're going to some level of motions of agile practices, whether they're daily scrum, whether they're planning poker and so on, the question comes up are we really doing agile or being agile, which is, are you really just following the motions and the practices and the ceremonies that are part and parcel of agile, or are you really understanding the essence and the core spirit of agile and really being an agile software team?

I think that's the topic we'll cover today. Damon will share with us what he calls the vital signs of a healthy agile adoption. The point is the existence of a meeting with the name of a standup healthy sign is a fact that tasks are now called stories is a healthy sign, so you cover a lot of fresh topics, and really we provide some real concrete guidance on what to look for to gauge whether you have a healthy agile practice or not really. That's the topic today.

Let me introduce Damon, our feature presenter. Damon is Chief Agilest at Eliassen Group. He is a regular presenter in these monthly webinar series. He is an agile coach and trainer who specializes in helping companies find and implement their ideal way of working in order to increase profits and find and serve new markets fasters. His specialties include agile development, of course, but also software configuration management, software development, automation, and lean transformation.

I'm delighted to have Damon come with his agile topic of Agile Vital Signs. With that let me do a double check. Damon, are you on?

Damon Poole: I like how you called me agile there.

Hemant Elhence: I look at you as agile.

Damon Poole: I remember, nothing to do with me, but one time I was talking to somebody about agile, and they said, "So when did you invent that tool?" All right. I guess I'm giving this person their first introduction to agile.

Hemant Elhence: Yeah

Damon Poole: Agile Vital Signs. This is a topic that came up as a result of several engagements I was on where I realized the patient was not healthy and the folks there didn't really realize it. That's the idea of agile vital signs.

At the end of the day, it all comes back to the values and principles, so I think that's a good place to start. I'll just let people look at this for a moment and read through it just as a refresher.

We're all familiar with these things, and at the end of the day, agile is not about scrum per se, it's not about stand-ups per se, but it is about things like motivated individuals, giving them the environment support they need, trusting them to get the job done, using working software as a primary measure of progress. One could argue that that's definitely an agile vital sign.

These are the things that talk about agile. When we're looking for vital signs, we're looking to see if these things are happening or not.

Scrum, Kanban, XP, burn up charts, burn-down charts, all very useful, and actually a burn-up chart is an agile vital signs; but it's not, itself, agile. It's the principles and values that make an organization agile.

From the manifesto comes the agile community, many practitioners that have been working hard on agile for many years, even before the manifesto came out in 2001. The idea is that, if you have the principles and values of the agile manifesto, you're going to want to implement some of these tools. The tools, themselves, don't make you agile. It's your belief in agile that makes you agile. The tools are just a way to implement it.

For instance, installing voting machines does not make you democratic; being democratic is what makes you democratic. It's the voting machines that help you implement what you believe.

One of the tools that we use underlying the way that we do agile engagements is something called the Agile Maturity Matrix. There's now over 50 indicators for vital signs, if you will, in the maturity matrix, and they all go from 0-4. We'll use stand-ups which are useful. I'll let you read through this for a moment.

This is the same sort of format that's used in all 50-plus of the maturity matrix indicators. We focus on sustainability, which is not quite necessarily agile, but if you think about it there are many things to do to be agile, and you want to get some things to a level that they're sustainable so you can move on and work on other things.

For instance, with a stand-up, if it's the scrum master that … If you're the scrum master and it's because of you that you're having the meetings, that's you exerting that effort for it to happen. It's not happening in a sustainable way. You could say that's it's sustainable because you believe in it and you want it to happen, but if you go on vacation and people aren't there because they want to be there, it's not going to happen, so it's not really sustainable.

You have all kinds of things that are working against you and the team: complacency, inertia, resistance, impediments. These are the things that will take you away from sustainability. If it's just you, it's not going to work. You've got to get other people. You've got to overcome the impediments, the resistance, the inertia and complacency to get it to sustainable. Otherwise, all your effort will fall back.

Sustainable in the case of the standup would be 80% of the folks that show up are there … 80% of the folks on the team show up. They're there because they want to be. It's a quick meeting. It's focused on the team stories not just, "I went to a bunch of meetings," or something like that. Agile would be not only are you doing that, but let's say on the seventh or eighth day of an iteration, you can see that things aren't going well. We'll talk more about that later. The team adjusts to take that into account. "We've got to talk to the product owner. We should take on new stories," or whatever it is.

This is something that you can download from us, and I'll give instructions on that in a moment.

This just gives an idea of what it covers. It covers the organization, team dynamics, and team structure. That's on an organization-wide perspective, but also team by team, product things like invest. Are those very small? Are they thinly sliced? Are they vertically sliced? Are you doing a process

mechanics? Are you doing the engineering practices? Are you doing things like TBD?

As I said, you can download this tool. It is about 50 indicators time's five different areas, the five different levels. Its 250 different cells. Each cell has in it an in-English explanation, like with the standups, of what it means to be at that place. You can use this to figure out where you are, where you're going, and then track progress towards those goals.

This gives you … You can get a video of this later, but if you want to write it down you can go to our website and look for the "Enterprise Agility Maturity Matrix", and you can download that. It's free for you to use, and it's very helpful. It may also point you to areas where you want to learn more.

Let's start looking at some of the specifics. We looked at standups. Sometimes people put together what I like to call a franking team. "Who's available today? We've got Joe who's a tester. He's never worked in this area but he's available. We've got Sue who's a developer, and she's worked in this area a little bit. It's unfortunate she doesn't get along well with Joe, but that's who's available so we'll put this team together."

It's alive, but is it really a team? Really what you want to do is you want to make sure that you have a longstanding team, a team that is really a team.

If you think about it, if you think about a sports team, your favorite sports team, are they created and then recreated every game, or is it primarily the same people playing together well? People that play mostly together and on a regular basis for a period of time, like a couple of seasons for instance in sports, those are going to be your better teams.

It's the same in software. People, they're going to learn about each other, they're going to get along well if they spend a lot of time together.

This is a recommendation from us. There is some data behind this. I think actually the best place to look for data is Rally has a new study out that talks about keeping teams together and how allocated that they are. You can get some more data from Rally.

Our general recommendation is to try to keep allocation levels about 70%. You don't want to have folks on the team that are on two or three different teams. You want to try to make sure that they're just on one team. Looking at your allocation level is a good indicator of health.

The same thing is the length of time throughout folks are together. There's no particular time that is guaranteed to work, but you can see if a team has been together for a while, if they're still fighting with each other and not getting along, that's not a high-performing team, and that's not a good healthy sign of agility.

Agile teams work best when they're gelled, and that's one of the things you can find from the short iterations and the feedback loops is: Is this team gelling or not? Are they getting work done on a regular basis or not?

One of the best ways to see if a team has gelled is: Are they productive? If they're not productive, it's not possible that they're gelled. I suppose they could be gelled but they just don't really feel like working, and that's a different issue.

Hemant Elhence: Damon, can you take one question on agile teams?

Hemant Elhence: You made the point about 70% or higher commitment to being on the team. Is there any guidance on team interacting outside of work environment, to bring the chemistry that needs to be there for the team; or is it adequate to spend whatever time they do as part of their day-to-day work and no informal socialization, and other time chemistry dimension needed.

Damon Poole: That's a great question. That could be a whole presentation all by itself. Just quickly, I've recently been reading … It's a little dated now, but the material is still great. I've been reading *Teamwork is an Individual Skill*. One of the points that is made in there is that what really makes a team work best together is to have a common sense of purpose, and a common set of goals, and a common mission.

The team-building comes from, "Hey, we need to do this together. We need to get all this work at the end of the iteration done together." That generally works best if it's a team effort. The act of doing scrum can actually be a great team-building activity; but at the end of the day, whether it's within work or outside of work, the idea is to get to know each other, to understand how each other works.

Maybe you've got somebody on the team that always has a poker face, and you might interpret that they're unhappy or whatever; but you get to know them and you realize they just have a poker face all the time. It has nothing to do with their inner state and they're not mad at you, that's just their face.

Absolutely, that's a good point.

Hemant Elhence: Thanks.

Damon Poole: Sure. If you're a Star Trek fan you'll get the reference there. One of the things that, whether you're using the maturity matrix or some of the vital signs we talk about here today, or your own vital signs that you find useful, it's good to know what you're talking about when you're doing agile.

Sometimes, and this doesn't happen all over the place, I'm not saying that, but more frequently than I'd like I see people saying, "We're doing scrum," and if you use something like the maturity matrix to see what they're doing, they're really not doing any agile except they have a scrum meeting.

I was talking to a friend who just joined a company, and he was all excited that they were doing agile. We were talking about a month after he was there and he said, "I was wrong. They have a scrum meeting, which is a standup, and that's what they mean when they say they're doing scrum; and it's an hour long."

The translation here is, "We require developers to go to a one-hour standup meeting every day. They're not there because they want to or they get anything out of it." Maybe it's a long staff meeting every day.

It's important to understand: what is it what you're actually doing to see how agile are you? How healthy are you? When people say, "We're doing scrum," that's almost meaningless.

A standup is a fantastic … Speaking of standups, standup is a fantastic way to see how healthy a particular team is. Let's take a look at this. Imagine now, let's imagine together, that we're going into the team room to do the morning standup, or the afternoon standup, or the evening standup, whenever you do it. If you're working with a distributed team, it might be later in the day, or very early in the day. You walk in, and this is what you see.

You're not really going to physically see, but all that you see is what you actually hear. Imagine that you hear … People walk in and you hear, "I worked on #5316 yesterday, and I'll be in meetings all day today. No impediments." You can just read through this. It's interesting, but think about, you read through this, what you actually know. What feeling do you have when you hear that, "Okay, everybody, good standup. See you tomorrow." Do you feel like you're okay? Do you feel like you're in trouble?

For me, when I see something like this, or hear something like this, I feel lost. I have no idea. It feels like, "Okay, we've gone through the motions here," but you don't actually have to be alive to go through the motions. It could be just that you're having a status meeting here and it's very interesting.

The other thing that you could do here is you could have a burn down. People talk about big visible charts. A burn down is a visible way of seeing what's going on. If I walk in and I also see this, I have some new information now. I can see we have information on day seven, so we must be on day eight, and it looks like we're trying to burn down to zero on day ten; so it looks like we have an iteration of ten days and we're on the eighth day.

We have three days to go here, and it looks like we haven't burnt anything down in a while. If we're burning down by hours that looks like people are saying that the estimated time left is the same. Why? They may have given some clues in the standup, but I don't know really which story is that attached to. I don't have that much information. This is better. Burn downs are good. This can help you see if you're healthy or not, but it's not the best.

Here's another example. Really what's going to work for you depends you your situation. The idea is: can you walk into a standup and get a good sense, even as a casual observer, as to what's going on?

I want you to think now. You're responsible for this team's delivery. Your compensation is based on it. Your quarterly bonus or whatever is based on the team doing well. You walk in. You can see from this that it looks like it's probably day four. We're at the beginning of day four, we've got seven days left here, and we've got these three points done. We've got one story all the way done. We are also in implementation and nothing in testing, nothing that's ready to accept.

What would be your thought here? Think to yourself: What would you think in here?

A common response at this point is, "Wow. Seems like there's a lot of work in progress. Couldn't you have something in testing if you took on fewer stories? Why is there nothing in testing?" It seems like there's too much work going on, but if I'm in the standup and I'm walking out, I'm feeling still pretty confident that we're going to be okay. This gives me a sense of what's going on, what might we change, and my advice walking out of here, if I'm on this team, might be, "Hey guys, let's not take anything else off the

to-do list until we've done some more things and gotten them to done. That's just not going to be a good idea."

Now it's the sixth day. We've gotten five days; we have five days left to go. Here's where we are. We've got three stories done, three stories not started, three stories in implementation and ready for testing, two stories that are doing testing.

It seems like, at the end of the standup, some of the QA folks, whoever's going to do the testing, is going to say, "Oh, I'll take on some of those stories," and life is going to be pretty good here. That seems all right.

Now imagine … This is our third example. You walk in, its 9:15, and you're responsible for this team. Here is the next slide. How do you feel? I don't know about you, but I would not feel too good.

Now I'm conflating two ideas here: how is this team doing in particular, and how is agile doing in particular? Now, because I have all of this information, I have a card wall, whether it's electronic or physical, I know how many story points I'm trying to get done, I have it in the form of stories and all that. I have pretty much the information I need to know to see how this team is doing, and having that information is a sign of health.

If I walked into a hospital and I saw a lot of beds, and I didn't see any heart rate monitors, I didn't see any kind of diagnostic equipment, I would be pretty worried to be a patient in that hospital. It's the exact same thing with agile teams. If you don't see any instrumentation like this, that's scary.

One example, I remember I was in an engagement a couple of weeks ago, and I knew that there was a team in trouble but they couldn't see it. They had their electronic card wall set up so that it was completely obfuscated what was going on. I asked them if they wouldn't mind changing it to get rid of all the swim lanes and they were on day eight; they had three days to go. They clicked … It was very easy. They were using … I forget what they were using. They clicked on something and they got rid of the swim lanes, and the whole team went, "Oh, my God." They had the most interesting standup ever.

That was interesting to me to see how just having that information allowed them to be a much more agile team. That insight that that team had spread to other teams, and they're doing much better there now.

Hemant Elhence: There was a question of this card wall topic.

Damon Poole: Sure.

Hemant Elhence: If there was a situation where there's a team doing their standup, and they all have access to the same card wall, but in their tool of choice, whatever tool they're using, whether it's Rally or Version One or Pivotal Tracker or whatever they're using, but each team member is looking at the same burn up or burn down chart, they have the card wall and all that; but when they come to the part to the standup, there's nothing pasted on the wall. They're not looking at any common picture. Is that a problem?

Damon Poole: Yeah, I think so. If you're not looking at a common picture, then it's hard to have a conversation about the same things. You're then having a conversation about what you remember on the tool that you were looking at or whatever.

Absolutely. Part of what I'm saying here is that, when you walk into the team room, or if you're distributed when you walk into the chat room if you will, it's important to all be looking at the same view and having the same idea of what's going on.

Imagine trying to play baseball or something and you don't have a common picture of what's going on. That's not going to work. It's the same thing with an agile team. You all need to be in alignment and sharing the same picture.

Hemant Elhence: Thanks. Just to take one quick question on the terminology you are using, I think someone is asking … You use the terminology of swim lanes. What were you really implying with swim lanes?

Damon Poole: Swim lanes. Okay, sure. A common thing with a card wall is the default. What I'm showing here is no swim lanes. You might have multiple swim lanes indicating that some stories are for different users, so you might have … Going from left to right, you might have a row that is only for Tom's stories, and another one that's only for Sue's stories. In order to get a whole picture of what's going on that's like this one, you have to scroll up and down and remember what you were seeing. "Okay, Sue had one story that was an implementation, and Tom had two, so I guess that's three."

If you remove all the swim lanes temporarily, you could see easily, "Oh, there's two stories in implementation, two stories in testing, etc."

Actually, that's a great idea. The next time I run this I'll show a jumbled mess, which will make it must clearer, as an example.

Hemant Elhence: Thanks.

Damon Poole: Interestingly, I remember before agile, when I was doing traditional development, the word trust never really came up. You might say something like, "I don't really trust this person on the team," or whatever, but it really didn't come up.

The thing is, if you're trying to get work done really in two weeks with a bunch of other people, and you don't trust your teammates, it's really hard to do.

I strongly believe that the level of trust between teammates is an agile vital signs. That may not be one of the metrics we're used to, the trust metric if you will, but I think it's very important.

There's a couple books out there, *Trust Works*, and *Book of Trust* and there's some others as well. They all overlap on these four areas. I like the ABCD that comes from one of them, but at the end of the day, when you're thinking about trust, it's really the folks that you have on the team with you, do you feel like they have the skills necessary to get the work done? If they don't, it's going to be hard to trust them.

That doesn't mean that they can't grow those skills. It may be that there's some new skill, some new technology, or whatever; but as people grow in their confidence, you're going to trust them more. Do you believe what they say? Somebody says, "Yeah, absolutely, I'll get it done by Friday." You think, "Oh, jeez, again?" Then there's not high integrity there. That could be because they always over-promise. You have to be very careful not to over-promise.

If you're in trouble and you need help, do people say, "Oh, yeah, I'll help you," or do they say, "Sorry. I have my own work I need to get done?" and then Reliability. Again, it's: Are you keeping promises? Are you acting responsibly? Something comes up unexpected. Do you talk about it or do you just keep it to yourself?

If I have a team member that keeps surprising me unpleasantly, I'm not going to trust that person. I'm going to be looking to other people. If that's happening across the whole team, that's not going to be a great team. It's not going to be a healthy agile team.

There's a lot of ways that you can take the pulse there in a lot of different ways. These are a couple of methods that I like. A friend of mine came up

with the rock-scissors-paper part of it, and Fist of Five has been out there for a while. The idea of the rock-scissors-paper is that …

First, Fist of Five is you, at the end of the standup you might as people, "How confident are you that we're going to, on a scale of 1-5, that we're actually going to get all this work done by the end of the iteration?" You might get a bunch of 5's and a 1, so it's not a person pressure thing. You ask the person, "Why not?" "Because I'm the only QA person, and we're way behind on testing, and nobody is helping me." "Oh, okay." That's a good way to get out some hidden information. If that person is the only person that has impediments you might think, "Oh, well, that's too bad;" but if it affects the whole team that's important.

The other thing you can do on teams where perhaps there's not high trust at the moment, or there's some nervousness, or maybe one team member is fairly dominant, is to do a Secret Ballot. Write down on a piece of paper one through five how confident you are. That can be done for shipping. That can be for morale. That can be done for trust. That can be done at standup. It can be done at retrospective. It's very important to do some sort of taking of the pulse on a regular basis. Otherwise, you know what? You're just going through the motions, or you may be. You don't really know.

There's a tool you can use. You can use a 3x5 card for this, to pass it out. This particular instrument can be a little awkward, so you've got to think about when to use it with a team. This is a secret ballot, anonymous thing that you do, and you have everybody rate, for yourself and others. The others is: Do you believe that 76-100% of the other folks on your team have the ability, the confidence, the skills to get the work done; or do you think it's 25%-0%? That doesn't mean that you think that they're bad people. It may be just that you need more Java skills on the team than you have.

The same thing is a reflection on yourself? What do you think others think about you? Again, this can be a little uncomfortable, but this really helps find any underlying issues that may be affecting productivity of the team. Again, without a high trust between team members, it's hard to have a healthy agile team.

Product planning is a broad area, and there's a number of agile vital signs in this area. One that was a recent heads-up for me … It's interesting how you can have blind spots. I had a blind spot here that, just because you show in a schedule and in training, or even if you're working with a bunch of teams, you can lose track of the individual schedules. I wasn't noticing that sometimes people would do their iteration planning before their iteration review.

The interesting thing that would happen in iteration reviews was that the feedback was … It was too late to stick it in the iteration plan because we already planned that. You know what? The product owner is not here right now. To get accurate information, so taking the pulse if you will, or getting information metrics or whatever, is valuable; but if you're getting data without the right setup it's worthless.

It's very important that your agile feedback loop is set up so that you have iteration review first. If you find there are new backlogged items, those are taken into account during iteration planning, and then you do that planned work.

Some organizations don't like that. They call it scope creep, "Our dates are going to be off," or whatever; but at the end of the day, if you find some important information that means you should change your plan, it's not very healthy to ignore it. "Hmm. Cholesterol is getting a little high," or whatever medical problem and you ignore it, that's going to lead to problems. It's important to do these things in the right order, to get the right feedback, so you can actually apply the information.

It's one thing to take the pulse, or get the information. Then you need to apply that information.

This is another thing that came up recently. We've been talking about product owner for ages in the agile community. Unfortunately, very often it seems that it doesn't work out. What happens is that you get maybe a product manager that is designated or appointed as a product owner, and from their perspective that's all very interesting and they just keep doing their job, their regular job the regular way. It takes time for organizations, especially larger organizations, to change that behavior.

I remember I was in an iteration planning meeting that was going on and on and on, and they were talking about so many details on the how and all that. I couldn't quite figure out what was going on. Eventually I said, "Why are you spending so much time in this level of detail?" They said, "This is the only time that we have the product owner the whole two weeks." Okay!

I think an interesting concept here is the idea of a team backlog owner. Whether you have the product owner serving in this role or not, sometimes people have proxy product owners, and there's all kinds of things. I'm not saying that that's great. The preference is that the team backlog owner is the product owner.

This is a concept that helps really make it clear what's useful for a healthy team. For a team to be healthy, they need to have a single source of information on priority. They need a single person that is talking to them about what to do next. They need a single person to ask questions about …

The idea of the team backlog, responsibilities of a product owner, and whether that's the product owner or not, you need one person that is attending all iteration planning meetings for that team, that they have enough work ready. It doesn't mean that they have to get it ready, but they have to make sure that it's ready, that they have prioritized that work, the same person … not multiple people, the same person. They attend all iteration reviews, they're available to answer questions from that team about that iterations backlog immediately, most of the time, and perhaps there's a few hours' delays once in a while.

Sometimes people struggle. I had somebody ask me the other day, "Have you seen a product owner that is responsible for all these things in other companies?" I thought that was a pretty interesting question because that is the definition of a product owner.

You can use this as a way to gauge whether or now product ownership is healthy for this particular team. If there isn't a person, a single person, that is doing all these things, that is not healthy. You need to figure that out.

One coping mechanism here … I haven't given a lot of coping mechanisms, but one coping mechanism here is that you have somebody designated as the team backlog owner or whatever. I know one organization that has a team that calls that person the mom, for whatever reason. They've designated a person to do all these things and to coordinate with the official product owner so that the team's got what they need. Whenever there was a question, that person was able to answer the questions and then get to work with the product owner to make sure that that was correct.

Eventually what often happens when you do something like this is that the product owner realizes, "Wait a minute. I should be doing those things. Why is somebody else doing it." That can be the "Aha," moment.

Having something like this available for people to look at, and a checklist to go through and apply to your team is a great way to see if product ownership is healthy within your team, and also if this is not a company-wide thought across teams, and it's really …

At the end of the day, if one team is doing this or not doing this, that's interesting; but you want to make sure that the organizations understands that this is important for healthy agile teams.

Hemant Elhence: Can we take one question on product owners?

Damon Poole: Absolutely.

Hemant Elhence: Would you consider it a red flag if a product owner was supporting more than one scrum team; at least let's start with two. Is that a red flag, if they were doing work for two teams?

Damon Poole: That's a great question. Let me sidestep it for a moment and give another way to think about it. Think about that product owner and those two teams, and run this test. Is that single product owner able to attend all the iteration planning meetings for both teams? You might need to slightly reschedule so that it is physically possible. Can he attend all iteration planning meetings … or she? Do they have two iterations worth of ready stories for both teams?

I personally have done this in the past, and the way I did it was I worked with the team, and they would work on getting some stories ready. I would make sure that they were ready. I delegated it to other people that really just love to work with user stories. Does that person across both teams have the stories prioritized when they walk into the iteration planning meeting? If they want to prioritize it and use up everybody else's time while they're doing it, that's not right, so that would not count. Are they able to attend the iteration reviews for both teams? Are they able to answer questions from both teams immediately most of the time, and a few hours' delay once in a while? If so, then sure.

If you try it again with three teams and you start to say no, then they should go back to two teams and not attempt three. That's the way that I would think about that.

Absolutely that's okay, but they need to be able to answer all of these questions yes for all of the teams that they're interacting with.

Hemant Elhence: Thanks.

Damon Poole: Sure. You might not think of a schedule as a way to check out health. This idea came to me a while back when I had the "Aha" moment that many organizations treat scrum as an overlay. I know I'm conflating scrum and agile; it's difficult to break them apart all the time. Scrum, often people

think of that as an overlay. I have people that are on five teams, so they go to five standup meetings.

That's not what's intended. People say, "Oh, agile, scrum, all these meeting I got to go to. I hate scrum." I can't tell you how many times I've heard that. That's not right.

Let's take a look at an example. This is just an example; your mileage may vary. The idea is that any one individual that's on the team, and going back to our product owner example that's a little bit of an exception, but the members directly on the team, the developers, the QA, the VA folks, whatever, their schedule should look like this as an individual. They should be on one team, going to one standup and one iteration planning meeting. Other than the 3 hours and 15 minutes a week, the rest of it should be completely uninterrupted work, no other meetings, nothing, just work on the stories for that iteration.

If it doesn't look like this … You could really think of it as clogged arteries. All the times spent on process meetings or whatever that doesn't have to do with getting the work done, you should think about as cholesterol in the arteries. You want to get rid of that.

It's not easy to do, but if you have a standup and a project status meeting, try to think if you can replace the project status meeting and instead have folks go to the standup meeting to get what they need. If you have iteration planning meetings that are going forever, maybe it's because the product owner isn't available throughout the iteration. Look into that. Anything that deviates from this isn't necessarily bad, but examine whether or not you can change it, or remove it, or do something else so that it looks more like this.

If you sketch out your schedule for the team members and it looks like this, then great. That's really healthy.

Hemant Elhence: There's a question on: where is the customer demo in this picture?

Damon Poole: The word demo is interesting. That is a word that's commonly used, a showcase or something like that; but iteration review, or in scrum the word is sprint review, that's the iteration review at the end there.

Hemant Elhence: Can you talk a little bit about backlog grooming that you have in week two, how that works?

Damon Poole: Certainly. That's a great question. Backlog grooming sometimes is thought of as a small group getting together to get ready for iteration planning, but that's not what's intended here. I see iteration planning and backlog grooming as being identical with the exception that, in iteration planning, you're saying, "This is what we're going to do," and potentially saying, "And here's who we're going to have do it." I don't mean assigning; I mean people choosing.

Really the same, the only difference is in backlog grooming you're getting stories ready and focusing on the stories for the upcoming iteration; but you're not saying for sure that's what we're going to do.

Where teams often start out is two-hour iteration planning meeting. The idea is if you do backlog grooming every other week it's going to shrink the overall meeting and also give the product owner a chance to say, "Oh, wow, that story is definitely read. That gives me a week to get it ready before the next iteration planning meeting."

You know what? I've got to say I'm only human. I'm as guilty as anybody else of this. Sometimes you forget to go do a checkup at the dentist or the doctor or whatever, but regular checkups are how you make sure that nothing is going wrong or going off the rails.

Whatever vital signs that you think are important, that you find are useful to you, and I would suggest that any of the ones that we cover today are useful or something to consider, you've got to look at them on a regular basis and when there's a problem take action.

"We didn't have a single standup the whole week." "Why is that?" "Because everybody is working on three different projects." "Oh. Why is that?" You can have those discussions about, "Why are these things happening?"

If nobody is responsible for making sure that this is happening, then you can go off the rails really quickly. I can't tell you how many organizations I've been into that say, "We've been agile for three years," and I say, "Oh, so let's talk about it." You look at the history, and the first six months when they were going agile, they were the most agile, and then it regressed from there.

One team I was talking to recently, they did the daily standup only three times a week, which is funny that you have a daily standup only three times a week, but that's what they were doing. Often it's good to have this … If you can in some way tie to goals, that there is a goal that people are doing the ceremonies on a regular basis and for a good purpose.

Here's the summary. This will be in the recording. I'm not going to go through all of these details. It's an eye chart. This is a summary of some of the things that you could consider that are very low-touch, very simple to do, won't take up a lot of time, and a good way to see … You could even use it as a checklist and go through and say, "Are we doing these things? Are we not doing these things? Why not?"

Sometimes people will say, "We can't do that here." I want to give you a thought here, which is that scrum in particular is not about, "Can we do it or not?" The idea is that, if you want to be agile, than you do it. The things that you uncover that you're tempted to change or not do, those are the things that you want to do differently in your organization.

Doing scrum does not really get your results. Doing scrum uncovers the problems in your organization that is preventing you from getting results. If you go to the doctor and they recommend changes, it's doing the changes the produces the better results. Just doing standups and retrospectives all by themselves will do absolutely nothing. You have to take a look at what it's saying and make adjustments.

That is the presentation. I think everybody for their time and for coming and for the questions. We have some time now to go into further questions.

Hemant Elhence: Thanks, Devon. We have time for a few more questions, and I already have some in the backlog here. Let's give all in the audience a couple minutes, and in the meantime let me quickly introduce Synerzip while people are jamming in with the questions on the Q&A panel.

Devon, if you wouldn't mind advancing the pages here so I can do a quick introduce to Synerzip. We can go to the next page. For those of you in the audience today who don’t know Synerzip, we are essentially an agile product development services company. We serve as a partner to various software companies, helping them accelerate their product development velocity following agile practices. We have a development center in India and the same on the US side with a dual-shore team.

We are able to provide a very capital, efficient, and high-quality software delivery process to our clients, and we tailor our agile as appropriate in each environment.

The final point, as Demon you build out in your five bullets, is the fact that, for every client, we give them an option if they want to convert the team that is working for them as captive team. It essentially allows our client to

have a more offshore agile development team while they can convert that to their teams any way they feel like it.

That's Synerzip in a nutshell. Let's start taking questions to deliver value to our audience today.

I have a few questions here in the backlog. First one, let's go back to, Damon, the two-weekly pictures that you were showing where you had the beginning one hour of iteration planning, and then 15 minutes or every day of standup, and so on. One thing that came up is: Where is the architectural design, review for architectural design, taking place in that two-week picture.

Damon Poole: That's a great question, and that's probably a much larger topic. One of the things about doing a two-week iteration, it's difficult to do an architectural review. That doesn't mean that I don't think architectural review is important. What I would say is that doing agile and doing scrum is not about doing the things that we used to do but with different words. I'm not trying to make light of the question or anything, but agile and scrum really are profoundly different.

For the last 30-40 years, we have been doing things that require architectural reviews. Agile doesn't do it that way. That really is a very deep topic, and my suggestion would be to take a look at books like *Emergent Design* by Scott Bain. That goes into great depth in 350 pages about what I'm talking about here.

The thing is that, one of the reasons we do architectural design review is to make sure we're not making any mistakes, making sure we're doing the right thing. Agile wants to do things differently. We want to do incremental design. We want to have the team take responsibility for more of the architecture. We want to see architecture more as consultative in nature.

I remember being an architect and hearing those exact words, and I bridled at it. I thought, "Oh, wow. You guys just don't take this seriously." We do; it's just different.

It's the same high-level goals. They're just implemented differently.

Unfortunately, I can't give that the full attention that it deserves, but architectural review in agile and scrum in particular is an ongoing continuous process and not a review step. I'm sure I didn't give the question complete justice, but that's something that I would say requires a

much longer discussion. There are some good books out there like the one I mentioned to go into more detail to answer that.

Hemant Elhence: Let's take another question. I'm going to paraphrase a number of questions. The question is around distributed teams. What's your thought, again in the spirit of agile vital signs, of distributed team appearing to follow agile.

Damon Poole: Did you say pairing or appearing?

Hemant Elhence: Appearing. They may do the motions of, let's say, daily standup, but they're in different continents, different time zone. Is that a problem, or is that workable? What's the guidance to distributed teams about what works, what doesn't work, in terms of appropriate agile methodology?

Damon Poole: A couple thoughts there. If it really is a single … One of the biggest things about teams in colocation is that, if you think about it, one of the things in agile is the idea of colocation. You don't have to do colocation to be an agile team. It's just that it helps. One of the reasons colocation helps is it reinforces the idea that you're a single team working on a single project. Well, you might be working on multiple projects, but all of the work comes through a single backlog, organized and worked on by a single product owner.

If I had a number of individuals working on a bunch of different projects, and they were distributed across the world, and I could either put them all co-located, some random number of individuals co-located, or I could take the same individuals and make a real team out of them, and they weren't co-located, I would take the latter. It's much more important to have a number of individuals that are all working towards the same purpose, whether they're distributed or not.

Between … You say, "Go," at the beginning of an iteration, and you say, "Stop," at the end of the iteration. To me it doesn't really matter what you do in between or where you're located. You could be on your Xbox I or your PS4 or playing ping pong for all I care; but if you get the work done that you said you were going to get doing in the iteration planning, that's the most important thing.

The vital signs that we got into today were a little bit more detailed, but at the high level that is the most vital sign, are you getting the work done that you said you were going to get done, every iteration, iteration in and iteration out?

If you can't do that with your team, whether it's distributed or not, you've got to start looking into why. Is it because the people that I'm not seeing face to face don't take me seriously, they don't trust me? Is it … What is it? Maybe we need to think about using Skype or Google Hangouts so we can see each other's expressions. Is it that we say we're going to get stories done but then we don't? Is it we say we're going to automate the test but then we don't? What is it really that's preventing us from getting that work done?

Yes, distributed can be a problem, but really the question is: Are we having things shippable every iteration whether we ship or not, or not? If not, why not? It may be distributed. It might be something completely different.

Hemant Elhence: Let's take another question, Damon. You mentioned through the presentation a number of times that iteration planning should take no more than two hours, preferably closer to one hour. If a team is consistently appearing to always need half a day on iteration planning, or even backlog grooming activity, what does this say? What might be going on? What could they do to …?

Damon Poole: Great question. There are a number of answers to that. I'll give a couple. It may be that they're just getting started and there's a lot of discovering, "What are we doing here?" That makes perfect sense.

More commonly I find that there's a couple of things going on, that team members believe that they need to discuss multiple approaches and arm wrestle it out which approach is going to be done. That is absolutely not the point of iteration planning. The point of iteration planning is only to make sure that the stories are ready, that they're small enough and that somebody's going to do them, and we have a belief as to what the approximate size is. That's it. There's no how. There's no, "How many if statements is it going to be." Nothing. All of that is doing by whoever is going to implement it later.

You may have some questions about how, but the questions should be entirely focused on what. What is desired? Is that an idea, are we supposed to have that on every screen in our product? Is that just in one screen? "It's just in the one screen." "Oh, great. That's a smaller story, then." "It's across our entire product." "Oh, my God. That's huge. Can we break that down?"

So often, because of the way we think from traditional development, which is perfectly natural and understandable, we really want to dive down into the details; and agile says, "Nope. That's not the idea here."

It may seem alien and wrong and counterintuitive, but we really have been doing things the same way for 30 or 40 years. Agile is absolutely different. When agile is done right, in the beginning it doesn't feel right because it really is different.

Hemant Elhence: Great. Thanks. Let's take another question, which I'm not sure I fully understood, but I'll read it out to you. It might make more sense to you. "Is there integration testing scrum required?"

Damon Poole: Integration testing. Scrum officially … This is just focused on scrum, not all of agile. Scrum, the official definition of scrum is that all of the work items are done to the definition of done at the end of the iteration. If your definition of done, which is supposed to be shippable, is shippable, which is what it's supposed to be, then if it's not integrated I don't know how you ship it. Absolutely you just do integration testing within the iteration.

I can see people rolling their eyes or saying, "Oh, my God. I could never do that." Absolutely understood. Scrum is not easy, and in the beginning you can't do that. Most teams don't do that immediately. It might take a year of more, but that is the goal. That's the aspiration, and it can be done. Teams do it. It's not easy. Agile is not easy, but the payoff is really big.

Hemant Elhence: Let's take … I think we have time for two more questions. Let's take the first one which is that you talked a lot in the beginning about trust. Is there any … I think you did allude to it, but do you have any guidance of Fist of Five, and any other way to measure trust?

Damon Poole: There was that card that I gave with different columns and rows and everything. That's a tool you can use.

It doesn't need to be that difficult. I think if you have a team that won't look each other in the eye and they're not laughing during the standups and stuff, it's probably pretty clear that there's some problem. The "Fist of Five" or really, if you think there's a trust problem, probably "Secret Ballot" is better. That can uncover that there is a problem, not necessarily what the problem is.

That's a whole different area. That's coaching. You really need somebody that is good at human dynamics. That could be a manger or that could be a coach, it could be a scrum master who helps the team figure out where the root problem is and to deal with it.

Hemant Elhence: Let's take one final question on metrics. At a very high level, if a team is running and been from all observations following good agile healthy

practices, what are the key metrics an executive or a manager should be following on a regular basis that make sense from their perspective to keep track of how we are doing?

Damon Poole: That's the Holy Grail, I think, that we've been looking for many years. Honestly I don't think I have a good answer there, but I can tell you what my answer is, which is: How did you ever figure that out in the past? Whatever you did in the past, you're welcome to do it now.

I think really the answer is: when you walk into the team room, or when you go into a standup, the people seem happy. When you talk to the business stakeholders and ask them about the team, do they roll their eyes or do they say, "Oh, yeah, that's a great team. They get stuff done, and they know exactly what I need. They're very responsive."

If you look at the business output, are the revenues up? Are the salespeople happy? If it's an internal thing or if it's over the web, are your statistics going up as far as product usage? Are you saving the money that you were hoping to save? At the end of the day, the team is there for a purpose that has to do with business, whether it's saving money or making money or some other value. Really you've got to look at the desired results.

If you don't know what the desired results are, I don't know how to help you. Productivity is interesting, but my belief is that people show up at work with a good work ethic, and it's not about measuring them. They either have a good work ethic or they don't, and if you provide an environment where they can live up to their potential, then whatever they're doing is what they're going to do. To me that's really a very judgment call.

Hemant Elhence: Thank you so much Damon for this topic. It really had a lot of value to our audience, and thank you all for attending it. Stay tuned for another webinar next month.

Damon Poole: My pleasure. Thank you for proving the forum to have this presentation.

Hemant Elhence: Absolutely. Thank you all. Bye-bye.

Damon Poole: Bye-bye.