Business901 *Podcast Transcription*

Implementing Lean Marketing Systems

Industry Stories on Agile, Scrum and Kanban

Guest was Eric Landes of the Corporate Coder Blog

Related Podcast:

Scrum + Kanban = Agile Discussion with Landes

On the Business901 Podcast, I had the pleasure this week interviewing Eric Landes. Eric is an Agile Project Manager, who has been using Kanban for software development since 2007. He has worked with Scrum, XP and other agile methods for over the past 5 years and has been managing software projects for over 10 years. Eric has his own blog, Corporate Coder, which can be found at http://EricLandes.com. He is also a frequent contributor to http://developer.com.

Eric has been involved with ASP.NET since Beta 1. He currently is an Intranet Developer/Project Lead for a company in the auto industry in South Bend, IN. They develop using ASP.NET, Sharepoint, Project Server, Reporting Services and other Microsoft technologies. He is also the president and founder of MADNUG (the South Bend .NET User group), writes articles on Aspalliance (http://www.aspalliance.com), and is the editor of Crystal Alliance (http://www.aspalliance.com/crystal), part of Aspalliance.

Eric first caught my eye when I was reviewing the conference speakers and workshops being hosted by the Lean Software and Systems Conference held last April in Atlanta. I had noticed that he was presenting a real life scenario of how an Operations Group’s Kanban adoption failed to improve cycle times. The session ended up in a 5 Why session and a Q & A to discuss other situations that had not produced results. I thought it was an excellent way to stimulate discussion and to learn more about Kanban versus the typical “this is what we did” approach.
Our discussion started with Agile drilling down to Scrum, the last twenty minutes on Kanban and what he has learned implementing it.

Joe Dager: Thanks everyone for joining us. This is Joe Dager the host of the Business 901 Podcast. Participating in the program today is Eric Landes. Eric is an Agile project manager who has been using Kanban for software development since 2007. He has worked with Scrum, XP, and other Agile methods for over the past five years and has been managing software projects for over 10 years. Eric, I first ran across you when you were presenting at the Lean Systems and Software Conference, and I have to admit I was very impressed.

Because what you brought to the table was a discussion of a failure. And I thought that was very brave indeed to take that subject forward and talk about, I guess it was a Kanban failure?

Eric Landes: Yes. Yes, it was. Hi, Joe. Thanks for having me on. It was one of the things I had presented the previous year in Miami as well, kind of our success. And so I had been seeing a lot of chatter on some of the email lists about, as we really gain this body of knowledge about software development using Kanban it would be good to see failures and where things aren't working quite right. So, that everything isn't just roses and a rosy picture. We understand that things don't always work quite right.

And I think the idea is we want to learn from our failures, so it's OK to fail. So I'm taking that attitude when I was doing that presentation as we wanted people to kind of learn from those failures or at least that particular failure.

Joe: I give you a lot of credit for doing that. I think that's great. Now, you have a blog called "The Corporate Coder" and it can be found at ericlandes.com. Is there a certain theme for the blog or do you develop that for ... based on LEAN software development? Is that what it's all about?

Eric: For the most part it's about what I do on the business side of things, and it's definitely focused on Agile, Lean software development topics. There may be some more technical types of blog posts I might put in there. But those are becoming fewer and fewer. I come from a developer background so I do like to dabble in C# or whatever. Try and teach my son Ruby on Rails or whatever. Normally it's not going to be technical. I'm probably not as technically proficient as I used to be, and it's mainly under that Agile and Lean software and that's kind of where I'm at now.

Joe: Being involved at Agile for probably 10 years I think or somewhere close to that neighborhood. Explain for the people that maybe do not understand Agile and is Agile really Lean?

Eric: Joe, you could have a pretty good debate on that with people like David Anderson and Jeff Sutherland. But I think most people would agree there are a lot of good Lean principles in the Agile movement. The Agile software development movement really came about as a reaction to a kind of the command and control as I call it or "waterfall method' of software development, which really didn't take into account some of what I call Lean principles of valuing people, your employees, and looking for customer value. In the software development waterfall method, you have a method that really states, we are going to gather our requirements up front. We're going to give them to the IT, and we're going to spend a lot of time gathering those requirements.

We're going to give them to the IT people. They're going to spend a while fixing them. We are going to get that back, and we expect it to be exactly like the requirements. The problem with that was; we weren't talking a lot to the customer to find out if their requirements changed.

We really had these death marches at the end of some of these projects to get them done, and we were really putting a lot of strain on software developers. So some people had come together they had been using some of these techniques, too, that were very successful for them.

They involved things like, instead of gathering requirements and then the IT group would go and huddle around, and do their work for six months and nobody would see them. They did a lot of feedback loops between the customer or a marketing person and themselves to get the software that they wanted correct.

So they started using tight feedback loops, they started breaking things down into smaller packages and they also tried to constrain expectations so that the developers, they would have to do the exact matches at the end of software projects where they are just working around the clock and burning out, and so I would say, when it started out, probably the relationship between the lane and agile was the idea of delivering value to the customer quickly and empowering the employees which I believe are two lean concepts.

You probably would have some people saying that agile is not necessarily lean for writing reasons but I like to believe that you can definitely move from agile into a more lean, continuous improvement kind of mindset but just because it do agile doesn't mean you are going to move into that lean concept, if that makes sense. So hopefully it was kind of a long explanation but hopefully that gives me an idea of the relationship between lean and agile.

Joe: Can you really have seven day and thirty-day iterations and get the customer feedback in short cycles like that?

Eric: Oh yeah, absolutely. We are currently using two week iterations so I use currently, I'm doing a pretty fairly pure scrum implementation which means, you have sprints and there came to decide what those sprints are, and within those sprints, your team commits to finishing x amount of stories and by finishing, we mean we can deliver that, we could actually, in this case, let's say it's a web application, we could actually deploy it to a web server and you could use those features at the end of your sprint. So we are committing to two-week sprints, and we are saying we can do, let's say for the sake of argument, five stories and we will finish those at the end of this. When we say that, that means the team, how big it is, is committed to delivering that and I have seen time and time again, teams will be able to do that. The real trick is, your stories need to be sized correctly, and your team needs to be able to have reasonable expectations of what they can deliver.

After you have done scrum a while, your team has an idea of how much they can deliver in one sprint, but I have seen sprints of one week work effectively to it. It just depends on the team and what they are comfortable with.

Joe: To build the user stories takes a little bit of experience and some trial and error because when you start out with the first, let's say scrum effort, you are going to make some mistakes?

Eric: Absolutely, yup.

Joe: You know that two-week iterations may end up being a three-week iteration and/or one fifth of the user stories not being done.

Eric: Right. So, let's say on our first... The first time I really did a scrum implementation, we just totally miss what we were doing. We totally... We over committed to our... What we could deliver. And so, when we got to the end, we had already set up expectations for the stakeholders. So, that's one of the things you really want to do with your teams is, to get an expectation that this is new. We are learning. Hopefully, you will take that into an account. If you don't have that luxury, in our case we have that luxury, if you don't have that luxury you definitely want to have some more training then probably we had. But, if you do have that luxury you can go on with that expectation, keep lower expectation with the stakeholders, with costumers. let them know, at the end, you can commit. You have better understanding of what you are doing and what you can commit to.

As long as you build that trust and start hitting your targets, you are OK. It is when you consistently don't hit targets that the costumers tend to get upset. I should also add, I was not doing this as a consultant. I was doing it as an internal group. You would probably have less forgiveness I guess if you're a consultant.

Joe: As a consultant, we could probably come up with better excuses because we've done it before, the excuses.

Eric: There you go. Make sure you have some excuses in stock, and you are good to go. But like I said after by the second sprint we did we really started hitting our targets and having better expectations of what we could deliver. So it turned out to be really worked out well.

Joe: Well a lot of my listeners are from the Lean manufacturing background so Agile may be a little foreign to them. We think of agile more what I have heard from different people that it is really from concept to consumption is a buzz word I have heard a little bit. Can you explain a feature in a story to me?

Eric: Sure. So, I am going to a couple of differences in Kanban. You can use feature and story within Scrum, as well. Most peoples do tend to use stories in Scrum. Usually, a term used... A minimal marketable feature. It's the smallest feature we can deliver on software that is going to be used by the customer. That makes a difference for the customers.

If you are in a software development shop and you have software or website that you are selling something digital like software. That feature is going to be something, say like a marketing person is coming up, whereas if it's an internal software application you might actually talk directly to the customer and get that feature.

The feature verses story, normally what I would say is those features normally break out into stories from the feature. You can almost think of stories that're a smaller set of a feature if you will that probably have to be grouped together if you are actually going to sell your software.

Normally, what happens when you are moving from, say Scrum into Kanban... A lot of the Kanban teams are using their ideas of features. And then, the idea is, they want to take the feature from... As you said, from concept to cash or whatever, and bring that through their system of software development.

When I look at the differences between what we call like a Scrum or XP, and the Kanban process. The Kanban processes really are the lean process that the software development uses. We are looking at mapping your process as it stands. Seeing where the bottlenecks are, using a board to map your flow, see where the bottlenecks are, and then try and fix those bottlenecks as you'd go.

Normally, you're going to use a lot of the Agile methods for software engineering. Some people even use iterations during their Kanban. But, for the most part, you're looking in the flow of things. You aren't really doing fixed link iteration, time-box iterations. You're doing more of... I don't want to say one piece flow because that's wrong.

I mean, we would love to get to one piece flow. But I don't know anybody who would claim that they're doing one piece flow in a software Kanban. At least, I've not heard of anybody doing that.

Joe: Before I jump further into Kanban, I want to ask one question. We look at Agile as being the umbrella over all these things. Scrum, XP. Are all software companies now Agile, or the larger percentage? Where is that really in the time-frame of things? I think of a bell curve, is Agile over the bell curve? Is it mature now? Is it a mature product?

Eric: I know I've seen statistics that say... Well, I want to say 70 percent of companies are doing Agile. But that doesn't mean 70 percent of the projects are Agile, it just means they've experimented with using Agile. They may have one or two teams, out of all their software development teams, using it. I'd say it's definitely a mainstream methodology. At this point in my experience, I wouldn't say a majority of software development groups are using Agile. But it's probably getting there. The statistics are kind of hard to tell. Statistics say 70 percent of companies are using Agile, but I wouldn't say that means everybody is in those companies. So, I don't know what a good guess would be there, but it's definitely mainstream.

If you talk about Agile to management, they understand that now. So, it's also a good way to bring in some of these concepts that are very similar. Like the Lean concepts, Scrum, and XP. I remember, years ago, when I talked about XP, people thought that it was dangerous or something. So, it's a lot different now.

Joe: Is there a time that you shouldn't use Agile? Or, you shouldn't be Agile, that you should use a waterfall project?

Eric: That you should use a waterfall project. I've heard arguments... In my experience, the projects I worked on... I would say, they all should use Agile. But you have something that is fixed, and there's no variation in what you're developing... We all know, the developers know exactly what you're doing. Then you maybe want to use waterfall, or you might want to mix in some waterfall methods with Agile. If you're doing something, say like, defense contracting or something. I've heard some people talking about how they use Agile with defense contracting.

If you're writing software for the Shuttle or something like that, or lives are depending on it. I think you can use Agile in all those, but probably, your quality effort is different.

Joe: If you're comfortable with doing it. I mean, if you have a terrible golf swing, but you shoot in the 60s all the time... Keep it, right? I think that way a little bit. I mean, it depends. I mean, you may not meet the top of the circuit doing it, but whatever your expectations are, if you're meeting them, you continue on.

We mentioned Kanban, and we brought that into the conversation. This is a good lead-in because a lot of people say that Kanban is really just a waterfall project on a board.

Eric: Right. Yes. I've heard that. To be honest with you, it probably could be at first because you are mapping your existing process. So if you're existing process is waterfall you probably are mapping a waterfall method to this Kanban. The surprising thing about Kanban, if you do... And let me kind of clarify this. When I talk about doing software using a Kanban system, first I talk about mapping the current method you use your boards. For instance, we maybe have a backlog of requirements, in our case it's called stories.

Then we have a business analyst who does some of their requirements stuff gathering and some other things, then developers and then a QA group. Then we might have a user acceptance testing, that's what we always do.

And then we deploy because we are a web based App, we deploy to our web server. So if we map all that into different queues, now all of a sudden we can see what everybody is working on. One of the things that is important to do is to put limits on each queue.

So logically what usually makes sense, if I have two QA people I'm probably not going to allow more than two things to be worked on at once in that QA queue. Those are what we call our work in process limits. We can talk about that in depth a little bit more but the important thing about those is if you get that right you can really start seeing bottlenecks and where your problems are, and that really starts driving improvement to the process.

So that's kind of the important thing. If you start with this waterfall like Kanban board, six months from now you probably aren't going to be very waterfall like. It's probably going to be driving a lot of improvements to your process.

At least it should be. You're probably going to see yourself having people help out the QA group if they're the bottleneck and trying to get them to, "Well, we're going to test now. We're normally BA, business analyst, but we're going to help you test."

Or maybe the business analyst is the bottleneck. And so now we're going to help them out and do some things there, and we're going to come up with some good ideas like running acceptance tests as developers in QA or whatever. The idea is you start improving.

You get ideas from the team, and they were going to start doing test room development which we didn't do before so that our quality is better. I think like I said, yes, you can start with waterfall. With Kanban, I think, that's probably true.

And to be honest I don't know anybody who really did that. But I think it probably could happen. I can't imagine if you're doing it right that you will be doing waterfall. Because you just can't help but improve that process. You are going to bring in new ideas and new processes in there to help make you flow faster.

Joe: If anybody is listening to this, you're probably have seen a Kanban board somewhere on my blog. But a description of it, you have four stages, for lack of a number. There is a column for each stage and its split in a doing and a queue. Getting ready to hand off to the next stage, is a pretty basic description of one. And when you do that I go back, and I see that, and then the work in process limits you're just going to... I think as David Anderson explained to me, you just circle a number. All at once it will limit your work in process, and you've got a Kanban board.

Eric: Yes, I would totally agree with that. As I said, those queues, the limits... When a queue fills up... The really interesting thing is when a queue fills up, and the team can't work anymore on their... I'm a developer, and I can't move my ticket now because here is this work in process limit. The rule is; I can't move my story to the next queue if the limit has been met. So that means I can't move on right? So now what do we do? One of the things we normally do in our Kanban projects is we would have daily stand-ups. So you would normally, daily, see what is going on. And if that would happen, hopefully, you don't wait until the daily stand-up to say that. But if it happens, at say 12, and your daily stand-up is at nine, you don't wait a half a day to tell anybody that. If you do, then we can tackle that as a team and say, "Well what is the problem here?"

If it is going to QA..."QA, what's the problem?" I really got to write a lot of tests here because we haven't done that. We didn't automate this. So we need some help. Maybe the developer can come over and help. There's some other dynamics that go along with developers helping QA but...

Joe: Really what that does is it creates more of a team effort. Is that not right?

Eric: Absolutely. That's the other thing is it introduces the idea that I'm not a developer, I am actually part of a team, and our team is trying to get this done. That's definitely something I have had issues with in some of my teams before. That idea of, "Well I'm in QA I don't do development." Or "I'm a developer, and we really don't do testing." The idea that we want to get through to them is, "No your part of a team." We want to do what it takes to get it done. If that requires you might have to test for a day, so be it. Let's do that and get it going so that we can get the team moving. The Kanban board just highlights that. You see exactly where the team stopped, and you see who's open. That's the other thing. Some people may not like the fact that is so visible, that their work is so visible and so transparent to the team.

But, for the most part, I have found that team members want to do well, and that's a good thing to see. To get that transparency and visibility into your process and they will help out when they can see that, "Yeah these guys got so much work." Or, "That girl over there really needs help." You know? We can help. That just helps tremendously.

I think that visibility just really helps the team concept going forward.

Joe: I think it is also interesting, because most of the problems that I have always seen moving from one state to the next stage, especially when you value stream map-up projects are the handoffs. If someone is coming back to help, they recognize what's needed in the handoffs, more than anybody else.

Eric: One of the first implementations we had, it turned out the bottleneck was me. I was doing builds and for some of these tests, we had to build software and then put it into a QA environment. If it didn't begin in a QA environment, it didn't have the latest build and couldn't do it. As we were looking at this, it became apparent "Well Eric, the last time you did this was three or four days ago, they needed it yesterday. So what's the problem?" That's one of the things we talked about was, "How do we know... How do we hand-off this so that Eric knows that he's signaled that, "Oh he has to do this.""

So we came up with a signaling system on our Kanban board that would tell me, "Oh, it's time to build." If we have two stories that are ready to deploy, then you got to go ahead and deploy. Until that gets filled, we don't deploy or some such rule. Bu that became kind of our signal, to show Eric that, "Hey, that's our hand-off." It's time for you to do your thing.

Nothing ever works perfectly. We all know that, and Kanban highlights it in its visual stream. But what happens when you've really got to ram something through? Do you still use the same methodology? How do you handle that?

Joe: It probably depends on who's doing the asking. But in reality, usually the really cool thing, especially about Kanban and now agile, if somebody comes to you and says we've got to get this done, you have a very good backlog or an understanding of what your team can accomplish and how long it'll take them, and you have a good system of tracking it. If they bring something in, and let's say it's a feature, especially in a Kanban board, because that's a flow kind of system. You can say, "If we know that 95 percent of the time, when we get something of this size into our Kanban board, we can deliver it in a week or three days, whatever your time is for that. You've got those statistics, and you can tell that to the business owner.

If the business owner comes back and says, "You're telling me a week. I need it in three days because we've got a customer who signed a contract. They need this feature out there," you can expedite that even more. That's something that as long as your team agrees to it, I'm comfortable with doing.

Eric: Some people might be not as flexible, but I understand when there are customer needs. But you don't want to just do that every time. You really want the business to be able to say, "Your agreement on how long it'll get done, that's comfortable to us, and we know we can count on that. We're comfortable with saying it takes a week."

That's cool, as long as you make that the next thing they're going to work on and you can tell me that they're going to get that tomorrow or whatever. That's the level of comfort I think you want to get your executives or whoever is asking those kinds of things to so that they trust your team to deliver in the time they need it.

When you have those silver bullets, you can re-prioritize or even take things off the board if you need to and put that silver bullet in. I'm throwing out that term "silver bullet." I think David Anderson needs that.

That's a case of where we can't just have this next on the priority. It's got to go now. If you have some kind of system in place, you can say, "OK, we can do that, but that means these stories, we're going to have to delay those," and the silver bullet gets done.

That's probably something I don't know that I've ever really done that. I try to discourage that. Normally, the customers that I have worked with are OK with just being next in line, with their request being next in line.

Joe: It's a poor business practice if it really happens often; you can’t really afford to let it happen that often. One of the things that I look at is that, if you have a Kanban board and you've developed a team, you know where your bottlenecks are. You know where your support's got to go. That certainly helps to get something through quickly. Your business leaders and the other people that come in can see the board. It's so visual that they know and have a certain expectation out of you, what's next, what they have to live with. Don't they? The boards can self-explain itself?

Eric: That whole visual board thing is probably the biggest selling point when you go to your executives or managers, or what-have-you. They see that value. Immediately they can see...

When you explain to them, "Well, this is what we're working on, and you can see here these are the things that we're currently working on for this project. Here's this other Kanban board. This is our support board, and this tells you what we're doing here." Wow, the lights just go on. One place where I worked we just had a tremendous amount of interest generated because where we had the Kanban boards were so visible to the rest of the company. People would just see it and go, "Wow, what is that?" And you've got a good conversation going. Next thing you know they realized they could see our work progress.

Other groups started doing similar things because they saw the value just in that visual part. Not even necessarily in the rest of it, which is even more beneficial in my opinion and at least you got them started down that path. You get a lot of good buy-in I think just because of how transparent everything is in your process.

Joe: Do you document this in software or does the visual board speak for itself?

Eric: For Kanban we had it in the software as well as on a physical board. When I first did it, we tried just doing it electronically. We ended up not continuously improving.

Joe: But you're software guys. You should be used to that, shouldn't you?

Eric: You're right. That was the argument: wow, we should do it electronically because we're software guys. But I found that having a physical board is... There's something about being able to see it. Perhaps when we all have 60-inch flat screens in our team rooms and we have the software that makes it look like that physical board. That will be a good substitute. But right now, there's nothing I know of that's really at that level or economical enough. Where you compare it to a physical board where people can see it, it's actually worth it to me for the double entry, I think. Because we were writing on a card or we printed it out and posted it up. Right now what I'm doing is actually all electronic. I'm not a big fan of that, let me tell you. I need something big and visible for the team to see. Otherwise, I think you lose a lot of that transparency.

Software people will tell you, "Oh, we're computer people. We should be able to use websites, wikis, whatever. That should be how we look at things." But it's just not in your face all the time. The boards are in your face. When you get together in the team room, you see exactly where you're at. Electronically is a lot nicer for historical reporting. But it's not good at the team level when you're trying to stand up with the team and have them see where the team at. It's just a lot better to have a physical board in my opinion.

Joe: I think it was Dan Roam, who's the author of "Unfolding the Napkin." One of the things, someone, asked him one time; "What software he used to create this presentation?" He said that dawned upon him, because it was just stick men on the drawing. He said by making it humanistic it becomes human. I thought about that for a while. I think having a Kanban board, having a visual board makes it humanistic. There are mistakes; there are things up there that you need to work through. Whereas, when it's on a screen it kind of dehumanizes it and takes the issues away a little bit.

Eric: Exactly, you can't do things like. I can't move the card to the next queue. I'm just changing a bit somewhere or a flag somewhere in the software. I don't know. I also think there is something satisfying when you get done as a developer or anybody on the team, and just moving onto that next queue.

Joe: You have a 10 item 'to-do' list, and you check that thing off. You get down to the bottom and want to check it off, that is pretty good.

Eric: Right. The other thing you will find on the physical board, at least this is what we did, if we ran into issues or something, we would have a story on a card. Then we would put... Let's say something came up there, something was blocking it. We were relying on somebody else or whatever. We would take a post-it, a certain color of post-it, maybe it is red or something... Put it on there, and at some point, you might have two or three of those post-its on the card. And it kind of takes on a life of its own, its own kind of thing that people are familiar.

So, if I glance up at that board, I realize, I know what story that is. It is a lot easier to track it because know it has this characteristic that is different from the other ones, that idea of physical vs. electronic.

Joe: A couple things that during the conversation you talked about that I would enjoy you qualifying a little bit for me. One of the arguments about Agile and Scrum, I think was it became too team orientated and didn't really focus on the customer enough, that it became real internalized. Is Kanban different?

Eric: I would argue, yes. My experience is Kanban is very focused on what the value is for the customer. So, if you go from what most people are using Kanban, the main marketable feature, that should be coming from a customer, and it should be about something that is value for the company, either it is making company money because it is a new feature, or it is saving them money or something, but there is some value in it for the company. So, that is one way that Kanban helps. I think the other way is it is very focused, it doesn't just stop with the development, start and stop with the development team. You can use Kanban to really go from the marketing group that comes up with the idea, so they could have their own board that feeds into the development board, and in the end you have to interface with your infrastructure people to make sure your server is setup.

It is really focused on the whole enterprise I believe, more so than say, Agile. Agile kind of came out as a reaction to very heavy-handed techniques, but it was very soft-ward of upper centric. Whereas, I think Lean, coming from that manufacturing background, Kanban comes from that background, it is very focused on the enterprise as a whole, the entire organization and not just IT.

As you have said, you use these concepts in your marketing, and I just see there is a lot of benefit in all parts of the organization for a Kanban board or these Lean concepts, and they should just flow throughout the company. As I kind of mentioned on that Kanban board, we are looking for one-piece flow.

Well, that elusive goal, one-piece flow, it really kind of starts with where ever your ideas from your features come from, and the approval process that goes through to development, to deployment, to then the marketing of its feature.

I mean, I think it is the whole kit and caboodle, and then probably how you take in your money and your accounting processes. I think there is a lot more capability in Kanban and in Lean, ideas to bring a whole enterprise together. Whereas, Agile just started at such a developer focus, it needs to add things on. I think with Lean, those tools are already there.

So, that is really a nice part about using Kanban is we are already using Lean tools and businesses understand that. I think a lot of them do.

Joe: I also look at Kanban, and you have had a lot more experience with it, and with the teams on that, but when you talk about bottlenecks, and I mean we attribute that back to Theory of Constraints. One of their arguments has really been is that having people idle is not bad. Kanban says that to you too, doesn't it?

Eric: Yes, that whole idea of wanting work in process, if you run up against a constraint, you don't just go... I'm a developer, and I also go back to this example... But if I'm a developer, I finish my story, it's ready to go to QA, no matter what, normally. What a developer will do is, just give it to QA and go on to the next. They don't care if the QA person had the time or not, bandwidth or not, to finish that. But now, with Kanban we say, "Well, wait a minute." If they don't have the bandwidth if their queue is full, you stop, and you don't do anything until that queue is open. And that means there might be some slack time. If you didn't help them or whatever, you might have some slack time and normally, some people get a little nervous around that, managers in particular.

But the idea is use that slack time; you don't just sit around and surf the net necessarily, you want to use that time to kind of hone your skills, maybe use it go to a webinar. Do some of those things you don't normally do when you're developing to make you better and it kind of - you do get some idle timeout that. So, you're right, it's not discouraged to have idle time in Kanban, I guess. It might be the way to put that.

Joe: Now, there's been a lot of talk about cadence also and a certain rhythm to a Kanban. Have you seen that develop?

Eric: Yes, yes. That's when you're really if you're not getting to cadence; you want to see why that is because you should be delivering on a regular schedule. Normally, what we do is we would schedule meetings with stakeholders, and we would show them what was going out to release. So, in Scrum, you kind of commit to, in this fixed length iteration, we're going to deliver this, whereas in the Kanban, this is what we're delivering and we've done it in this priority. And your cadence should be so that the amount of effort of what you're delivering is very similar every time. And, if you see wide variations in that effort, then I would argue your cadence is not too good because you should be delivering very consistent chunks of effort or whatever when you deliver.

For instance, we would say every two weeks, we're going to release our features to production and that was kind of our cadence. It ended up that I think they moved that out a little bit of the team I was on. They moved that to three weeks or a month where they were delivering every month because it turned out customers were saying that was too fast. You're delivering too much, and we can't handle all the new features because every time you release something, we learn them and then it comes back.

Joe: That's a good problem to have. Is there anything you'd like to add to this conversation, Eric?

Eric: I'd just like to say, I really think, and it sounds like you're coming from a Lean background, and you would probably understand this too, but Lean really, I mean, I really believe that it really involves that whole enterprise and that whole continuous improvement. There are a lot of tools there. So, I think that's just really, really an outstanding way to go and I believe as Agile methods mature, you're going to see a lot more integration with Lean and with these Lean ideas becoming more and more commonplace in software development.

Joe: OK. I would like to thank you very much. I've had a delightful conversation. You can find out more about Eric by reading his blog, Corporate Coder at EricLandes.com. This podcast is available on the Business901 blog site and also at the Business901 iTunes store.

So, thank you, Eric.

Eric: Thank you, Joe.

Business901 *Podcast Transcription*

Implementing Lean Marketing Systems

Joseph T. Dager

Lean Six Sigma Black Belt

Ph: 260-438-0411 Fax: 260-818-2022

Email: jtdager@business901.com

Web/Blog: http://www.business901.com

Twitter: @business901

What others say: *In the past 20 years, Joe and I have collaborated on many difficult issues. Joe's ability to combine his expertise with "out of the box" thinking is unsurpassed. He has always delivered quickly, cost effectively and with ingenuity. A brilliant mind that is always a pleasure to work with." James R.*

Joe Dager is President of Business901, a progressive company providing direction in areas **such as Lean Marketing, Product Marketing, Product Launches, and Re-Launches. As a Lean** Six Sigma Black Belt, Business901 provides and implements marketing, project and performance planning methodologies in small businesses. The simplicity of a single flexible model will create clarity for your staff and, as a result, better execution. My goal is to allow you spend your time on the **need versus the plan**.

An example of how we may work: Business901 could start with a consulting style utilizing an individual from your organization or a virtual assistance that is well versed in our principles. We have **capabilities to plug virtually any marketing function** into your process immediately. As proficiencies develop, Business901 moves into a coach's role supporting the process as needed. The goal of implementing a system is that the processes will become a habit and not an event.

Business901 *Podcast Opportunity* *Expert Status*