Business901 *Podcast Transcription*

Implementing Lean Marketing Systems

Kanban

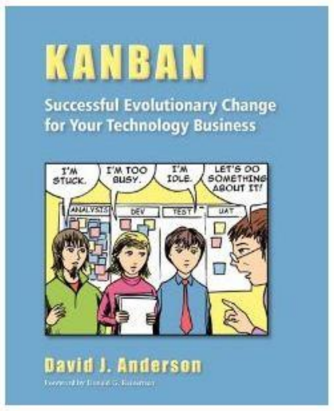
Guest was author David Anderson

Related Podcast:

[Kanban, could we call this podcast anything else?](#)

David Anderson, author of the recent book, Kanban appeared on the Business901 podcast and added 50 minutes of Kanban discussion. David covered a lot of ground in this discussion and answered a lot of questions for me that his book raised. David is a thought leader in managing highly effective software teams. He is President of David J. Anderson & Associates, based in Seattle, Washington, a management consulting firm dedicated to improving leadership in the IT and software development sectors.

David has been part of the agile and lean methodology movement since 1997 when he participated in the team that developed Feature Driven Development at United Overseas Bank in Singapore. He has 26 years' experience in the software development starting in the computer games business in the early 1980's. As a pioneer in the agile software movement David has managed teams at Sprint, Motorola and Corbis delivering superior productivity and quality. At Microsoft, in 2005, he developed the MSF for CMMI Process Improvement methodology – the first agile method to provide a comprehensive mapping to the Capability and Maturity Model Integration (CMMI) from the Software Engineering Institute (SEI).


KANBAN
Successful Evolutionary Change for Your Technology Business
I'M STUCK.
I'M TOO BUSY.
I'M IDLE.
LET'S DO SOMETHING ABOUT IT!
ANALYSIS
DEV
TEST
UAT
David J. Anderson

His first book, Agile Management for Software Engineering: Applying the Theory of Constraints for Business Results, published in 2003 by Prentice Hall, and introduced many ideas from Lean and Theory of Constraints in to software engineering. David can be found at AgileManagement.net

Joe: Thanks everyone for joining us. This is Joe Dager, the host of the Business901 podcast. Participating in the program today is David Anderson. David has been a leader in improving the performance of technology teams through Agile and Kanban methods. In fact, David was the first to implement the Kanban process for software development back in 2005. He recently authored the book "Kanban" which was just released in April of this year. David, could you start out by telling me a brief history on the why and how that you started with Kanban?

David: Well, first of all, thanks for having me, Joe. It's nice to be here on the Business901 podcast. And how did I get started?

Joe: I noticed in your first book "Agile Development," you discussed Agile and Goldratt's Drum-Buffer-Rope quite extensively. Then it seemed to evolve into Kanban. Is that how you got started?

David: Oh, definitely while I was writing the first book the seeds of what took place over the last five years and resulted in the more recent Kanban book that those things are connected. But as I was thinking about what to write in the Agile Management book and researching not just the Theory of Constraints but some related concepts, I realized that I wanted to take a different approach to introducing change and improvements. In fact, at the time while writing the first book I was experiencing for the second time things that were resisting the introduction of Agile Development methods. Not just development teams, but the organization generally was resisting the changes, and that was the second time that had happened to me over a three or four-year period when trying to introduce Agile at a reasonable scale, an organization of one to 400 people.

And it occurred to me while working that the more incremental, evolutionary, iterative approach to introducing changes was far better than coming along with a whole new methodology. That was the seeds of what ultimately became the Kanban method.

Joe: What kind of your specific problem does Kanban solve then, for software development?

David: We're not really talking about a specific problem or project management. The real problem that Kanban is solving, or at least helping with, is change management and cultural change. There's often a desire to fix things or produce better outcomes, or use different techniques that will lead to better outcomes, but the introduction of those techniques or the introduction of changes to fix problems like sales projects, those changes meet with resistance.

What we've discovered is that by use of the Kanban method, that resistance is greatly reduced. The core concepts of visualizing the work flow and limiting the work in progress, those two mechanisms in themselves do not meet with a lot of resistances.

It's hard to find someone who will object to the idea that visualizing the work and how it flows from one person to another, or one team to another, is a bad thing. Most people will be quite happy to admit that that's a useful addition.

And then if you ask someone if it makes sense to progress or duly recognize that they have some limited capacity and that it would be a useful thing to reflect that in the work flow, there's usually very little objection to that idea.

So, two ideas for which there's very little resistance and together those will provoke or catalyze a significant number of additional changes. And that's really the secret.

Joe: A lot of my listening audience is Lean Six Sigma type people and when they think of Kanban, they think of Kanban in the grocery store concept: using the explanation that when the boxes run low on the shelf, you replace them. There's a visual signal telling them something that needs to be replaced. What's the difference between Kanban in software development?

David: Well, that basic concept is still the same. There are a number of really significant differences in the domain problem for knowledge work generally, not just software development, in comparison to manufacturing or distribution. I think that the two key things are the differences and variability. There's a lot more variability in knowledge work problems, like software development.

Often that variability is desirable because it contains value that no two features in the software development project are ever the same. If they were the same, they would have no value. Or the second or subsequent one would have no value. So it's important to be able to implement the Kanban system design that embraces that variability and doesn't squash it too much.

Another key difference between a manufacturing or distribution type situation is the difference in the cost of delay. Generally, when you're fetching, say, gearboxes to be integrated onto a chassis or some vehicle, then the cost of delay between one gearbox and the one behind it in the line or some subsequent inventory followed by it in the value stream, it is really homogeneous. From one item to the next, there's a homogeneous cost of delay.

So, Kanban systems for manufacturing tend to be faster in first out queuing; whereas, with knowledge work problems, software development, for example, the cost of delay is completely heterogeneous and needs to be examined separately for each item. And as a result, first in, first out queuing doesn't make sense. It's not optimal economically.

So Kanban systems, while designed from the same principles that you would use for manufacturing or some other physical goods type problem, while the principles are the same, the practices and the implementation is different because it's a different domain with different considerations primarily much wider and desirable variability and heterogeneous cost of delay.

Joe: You talk about cadence in the book and why that is important. Cadence means to me more of an even flow.

David: Cadence is really the idea of getting an organization into a rhythm to do certain activities of a certain frequency. And the advantage of having a rhythm is it reduces the coordination. There's an expectation that certain things will happen at certain points in time, for example, the delivery of new requirements to a software development team. Or the delivery of working code into production back to the customers. Or the frequency of an operations review meeting, or the frequency of a project status meeting. Or the frequency of implementing process improvements and so on. So cadence has a rhythm. The rhythm has a coordination cost advantage. The question is what should that rhythm be? For example, how often should you deliver working software to the customer? Or deploy to production or whatever the delivery mechanism is.

And the answer to that question is controlled by two key variables. One is the value being delivered. And the other one is the transaction cost of making the delivery. So in the agile software community, we often hear the expression "minimal marketable feature" or MMF. And there's often debate about how minimal an MMF can be, and observations made that when people express something as minimal it can often still be half the size and it would still have value.

But the value is only one half of the problem. The other half is how much does it cost to make a delivery. It's not just the cost for the developers, but the cost for the customer. What's the training cost for the staff, the overheads for installing a new piece of software, potentially some switching costs from an older version or a competing product?

So while a minimal feature in terms of value may make sense, it doesn't make sense economically to deploy it. There might need to be several really minimal features that provide sufficient value to offset the cost of the delivery. So, the cadence needs to be worked out as a combination of those two variables.

And in general, in the abstract, if we're trying to work out the cadence for any activity, we need to take into consideration its contribution to the value being delivered and the economic costs. The transaction and coordination costs that will either be added or reduced as a result of having an activity like prioritization or retrospective, or delivery, and so on.

So, I really see cadence as a slightly separate problem from the idea of the variability and the value, or the complexity, or the engineering effort required for any given feature.

Joe: Now, is cadence then very much like, let's say, a sprint in Scrum?

David: You could say that a sprint in Scrum has a cadence. In fact, you could say that several of the activities in that sprint have a cadence and the cadence is always the same. So, if the sprints are two weeks long, then the sprint planning cadence is every two weeks, and the retrospective cadence is every two weeks, and the delivery cadence is every two weeks.

The issue with that is that while it's very simple, it's very simple to communicate, we do everything in this two week cadence, it's probably sub-optimal economically because the economics of the individual activities are not being considered separately.

So, prioritization, for example, what's the transaction and coordination cost of that, and how does that affect value. In that particular example of prioritization, we need to consider the risk involved in any value calculation and that leads us into the consideration of different options and the timing of those options.

It's likely in most domains, that it doesn't make sense that the prioritization cadence is at the same frequency as the delivery cadence, or the retrospective cadence, or the process improvement cadence, or whatever. So, the difference is that Kanban allows people to be coupled the cadence of all these activities whereas a typical cross-generation agile method such as Scrum, couples all those cadences together. And that's sub-optimal, economically.

Joe: When you start talking about Kanban to Management, your first start, do you explain the card wall to them because the card wall isn't necessarily a "wow". I mean it's not like, "Gee, we need to dive in to do this" because we've been doing card-walls and that. How do you go about introducing Kanban to them? How do you introduce that to someone, or to an organization?

David: I go about this in a fashion. For example, why would it make sense to visualize the work flowing? What's the benefit of that? I might talk about concepts like, "How do you find a bottleneck?" Well, you can't walk around a cubicle farm and look for a big pile of inventory lying around. And you can't walk around the cubicle farm and ask people whether they're busy or not, and then try and find the one place where everyone's busy, in comparison to everywhere else in the organization that we have slack, and say, "Ah, well now we have the bottleneck."

The truth is, that people are always busy, and they're usually overloaded, and working overtime, and working from home, and working on their day-to-day basis while they're driving to and from work, and have several hundred unread emails in their inbox.

So there is really no mechanism to see the potential improvements unless you visualize the work and I think that's not a difficult argument. People get that and they also understand the work needs to be visualized in a way that it's accessible and people will use it.

And then I might talk to organizations about why it makes sense to limit work in progress. And again, this is usually not a difficult conversation. Ask someone, "How many projects you working on right now?" and they reply, "Eleven" and you say, "Well, you think that's a good idea?" and they usually say no.

Then if I ask them, "What do you think the right number is?" the right number is usually at least half of what they're currently working on and often a lot more.

With one organization I met in Denmark a couple of years ago, the Vice President of Software Development told me at a meeting in front of his colleagues that everyone in his department had at least seven things in progress on average. So I looked at the Marketing Vice President and said, "I know you're the marketing guy and software's not your specialization, but does that sound like a good idea to you?" and he replied and said no it didn't. The idea that every software developer was doing seven different things at the same time doesn't seem wise.

So I asked him, as the marketing guy what he thought the right number of things in progress would be for the developers and of course, he really didn't want to answer so I ventured an answer for him and said, "One probably doesn't feel right" and agreed with me that one thing per person wouldn't be enough. I said, "Well, how do you feel about two things per person?" He wasn't sure, but I could see his facial expressions were changing.

So I said, "Well, how about three?" And his face was relaxing even more, he was beginning to smile. And I said, "Well, OK, how about four things per person?" He immediately replied back and said, "No, four's too many."

I said to him, "So, the right answer is two or three." And he said, "Yeah, I suppose so." So I looked at the development base president and I said to him, "So, your colleague in marketing is offering me three things per person. Are you going to buy that deal?"

He looked at me wide-eyed and I said, "No, no, I'm serious here. The marketing guy who gives you the work thinks that your people shouldn't be working on more than three things per person. So are you going to go outside and shake hands on the deal with him over this?"

And he said, "Wow that would be amazing." I said, "Yes. So you two guys can now have a very difficult discussion about which of the four things per person your developers are going to stop working on."

Just that one thing will have a massive impact on that organization, just the fact that they're now willing to have that discussion, and that they recognize that doing too many things all at the same time is bad for the business.

So that's technically two things I'll do to introduce Kanban. And at the slightly higher level, I will also ask if the organization has gone through a number of painful and failed change initiatives where they tried to introduce Agile, or CMMI, or Rational Unified Process, or something going back over the last 10 or 15 years.

And whether those initiatives met with a great deal of resistance and ultimately failed, or at least failed to institutionalize them. And generally, there's a lot of acceptance that that has happened in the organization, and that they'd like to try a different approach.

So, I sell them on Kanban as an incremental, iterative, evolutionary approach to introducing change. It's a way of producing a Lean outcome, but the Kanban method is really a way of seeding the conditions for this complex adaptive system.

That's software development and project management and the weight of business, to evolve into a Lean organization; an organization with a Lean culture that exhibits most of the properties and behaviors that you associate with a Lean company.

Joe: We talk about having too many projects on the board. But once you get into it, I think even in your book you mention co-currency and then later about swim lanes, but how do you prevent just adding things to the Kanban until it becomes just as ineffective as the system they're in now? Doesn't it just grow back into that?

David: If people are willing to break the work in progress limits, yes. Of course they'll be completely aware that they're doing it, because they'll see the effect. They'll see the effect on the board. They'll see the effect in the matrix, in the performance, in terms of the time to deliver or their due date performance against promised deliveries, or the throughput. So they'll see the creeping effects of, if you like, indiscipline. And what we've observed is that that tends to provoke corrective behavior. So things don't degenerate back to the way they were. I think that the key is Kanban reveals the mechanism.

For perhaps five years prior to that, people had been doing card walls with Agile Development, but really what the card wall was doing was helping them visualize the work. And if you notice, the one thing that I use in the Kanban book and in all the other materials, conference materials and teaching materials, it's very carefully chosen. It says, "Visualize the work flow."

That's a step above just visualizing the work. It's really saying we want to be able to see the mechanism of how different people and different teams are interacting, and the effects of their actions or inactions, or of the decisions or indecisions. I guess that should be indecision, singular.

So, I think that it creates a very self-correcting mechanism. However, it will not work without some leadership. And the teachings of Deming really stress leadership as incredibly important. And ultimately, Kanban isn't some magic pill or silver bullet. It's a mechanism for incremental effective change that will catalyze or provoke a Lean outcome. But none of that will work unless there's sufficient leadership to encourage, and make it happen on a day to day basis.

Joe: When we start out, and I've always heard that you're going to start your Kanban and you're going to draw your value stream out, to start that, and that becomes your Kanban board. Is that a correct analogy, or is that? What else do you need? How do you start with Kanban?

David: I know that you put up the getting Started dozen points on your website, so there is more to it than just mapping the value stream. It's important to understand the different types of work that flow through that value stream, and perhaps to analyze whether it's all one value stream or several. So I encourage people to understand concepts like, who gives us the work? Where does it come from? Where does it go to? What steps does it flow through, while we're handling it? And all of those contribute to identifying different types.

Typically if something come from the same place, and goes through the same place, and follows the same work flow, it may be of the same type. But as soon as those things differ, if the source differs, or the destination differs, or the work flow differs, it's usually a different type of work.

And separating work into different types is really helpful, because then we can look at the demand for those different types and the expectations. What represents customer satisfaction for delivery of those different types of work? And we can start to design the system in terms of the capacity that we allocate for different types.

The service level agreement that we might design for the handling of different types, we can design those around the demand analysis. And while we do the demand analysis we may also look at the need to offer several different classes of service.

In the book, I talk about four typical classes of service: Expedite, which means as soon as possible, delivery on a fixed date or handling, standard class delivering within some reasonable expectation. That's the standard class icons. Then I have a fourth one I call, intangible. Where there's no significant or tangible cost of delay within a reasonable delivery period.

And it makes sense to have a mix of different classes of service within the combined system also. The reason being is that the lower classes of service act as a buffer to absorb the variability and impact that higher cost of service items will cause.

It's well understood in industrial engineering that expediting increases the inventory level, increases the lead time, reduces duty performance, and just generally adds additional availability into a manufacturing process. Expediting through a factory is an undesirable behavior because it impacts everything else. But, companies don't do it because expedite orders might be very valuable. They do it in order to achieve greater revenue.

It is desirable to offer an expedite process even though it is known to impact other things. If you want to minimize that impact, you want to design the Kanban system in a way that it will allow for expediting without impacting other things, which are important or critical either in revenue terms or in the trust level with the customer.

Having a mix of class of service also helps so Kanban systems for software development or technology businesses tend to be a lot more elaborate in some respects than Kanban systems in manufacturing. They tend to have multiple classes of work with multiple types of service. And that is to deal with differences in the demand, the different variability, the risks involved.

Joe: It's not just a simple card wall, with "to do", "doing", and "done" on it.

David: Well, you may start there. An organization may start with a few columns, "to do", "doing", and "done" and put a work limit on the "doing" column and really, if we are thinking in industrial engineering terms. Right at the beginning you mentioned that my thinking had evolved from Theory of Constraints to Kanban. The explanation for that is tied in with why the Kanban community website. It isn't called Kanbancommunity.org or Kanbansociety.org. It's limitedwipsociety.org. Wip being Work in Progress. And the reason is that we've come to realize that any work in progress limited tool system, like a Conwip system, or a DBR system from Theory of Constraints, or Kanban. Or it could be a number of other variants that perhaps don't have specific names.

Any of those will create the same effect. They'll provoke this incremental, iterative, evolutionary change, and provide a framework that will generate a Kaizen culture and, economically, a Lean organization as an outcome.

Joe: I think you did an excellent job with your 12 steps. Defining the entry point and the exit point are so critical in a Kanban. It sounds so simple when you do that but it's not really all that simple in practice, I don't think.

David: It does challenge to people to think hard about who gets them the work, and what do they do to it, and also where do they deliver it to. It can open their eyes; actually, the 12 points that was a relatively late introduction into the manuscript. I was probably two thirds of the way through the book and the whole concept wasn't in the table of contents. So I actually really have to thank Ron Jeffries, a well-known extreme programming advocate. He was a regular poster in the Kanban development list online, the Yahoo group that many of us use to discuss this topic.

And it was through some of his really specific, difficult and challenging questions that I realized there was something very important missing from the book. That resulted in a whole new chapter, and in the end that's chapter 15 in the finished book.

For a long time, in earlier drafts of the manuscript, it was chapter five. But I found that positioned at chapter five, it was full of a lot of forward references. For example, it would say, "Define your work item types described in chapter six." "Decide on your prioritization cadence described in chapter eight." And so on and so on.

I realized towards the end I had to move it to chapter 15 so that I could then refer back to everything people had already read. The chapter had to be changed in title. It was originally called "Getting Started with Kanban" and you couldn't have a chapter with that title, and be positioned as chapter 15.

And the really funny thing is my copy editor changed the title in the published book. Of course, it's "Starting a Kanban Change Initiative", and she thought that was too much of a mouthful so she changed it to "Getting Started with Kanban."

I had to reply back to her copy edit and say, "We can't possible have a chapter called 'Getting Started' as chapter 15." And so she reluctantly agreed with me, and the chapter is "Getting Started with a Kanban Change Initiative."

I think that really underscores what we're trying to do. Doing Kanban is a long-term commitment for an organization to change its culture and to improve its processes and its performance incrementally over a long period of time.

Joe: I think you did a great job on the book. As I've mentioned to you privately, it's a very easily read book for people that want to learn about Kanban or how to improve their culture and how to learn to visually improve their work flow. Just not a software developer needs to read it, it's really a book for practically any service industry especially, or any technology-driven industry. Even a manufacturing company, if they look at it from the fact that they don't try to make the correlation between Lean Kanban manufacturing and this Kanban.

David: Well, thanks a lot. I'm actually really pleased with the book. I think it's got tremendously strong production values. I did sink quite a lot of time, energy, and money into ensuring the really high production values in the book. I also had a fantastic team of volunteers, who are all mentioned in the acknowledgments, who helped with reading and making suggestions and edits. So, altogether I think it's a great outcome.

I agree with you that the book is very accessible to a non-software development audience. In fact, last night I received a Tweet message on Twitter from a lawyer who said that she had bought the book earlier in the day and was finding it a really fantastic read.

We also know that companies that are adopting Kanban, it often starts in their software development or their IT operations group, but we see a growing number of businesses around the world are pushing out the ideas into other departments; marketing, sales, human resources, even the finance department.

We're seeing several, I have to say small to medium sized business, not larger ones, at least not yet. But we see several businesses where they're using the Kanban concept of visualizing the work flow, limiting the work in progress, and using that as a mechanism to provoke conversations about how they do things better.

We're seeing that spread, and I'm hoping that that will be a focus for us at the 2011 conference in Los Angeles. That we'll be able to attract people from other departments within companies, not just the technology people. And talk about how Kanban can help in a much wider way.

Who knows? Maybe there'll be a second edition of the book in some years to come, and it will be much less focused on software development.

Joe: I think it definitely could. There's one other item that I would like to address. It's something that I've always struggled with and I've read different things on it. It's really about when you manage the work and process, it's about flow, and flow is about having everything in place and being able to do it. Once you start it, then you can finish it.

When flow gets interrupted or gets a hiccup and it stops. It just seems like it never gets going again. But part of that, I always think about has got to do with having it ready in the queue, and having it ready to be processed. And you talked a little bit about managing the queue.

Is that similar to... to a scrum backlog?

David: Oh, for sure, yeah. There's whatever is off to the left hand side of the board that you don't see. And in general, we try and discourage putting a lot of energy onto the backlog. There's maybe a couple of things that I talked about in the book, the idea of going through the backlog and deleting items which are considered no longer relevant; we see some companies adopt policies that if something stays in the backlog for a period of time, in some cases as little as two months. If it hasn't been selected and brought onto the Kanban board within two months then it gets deleted.

Some other places have longer policies, maybe six months. But the idea is that the backlog is just a large collection of options. Sometimes there needs to be more sophistication, and my colleague Corey Ladas talked about this in his Scrumban book that was published perhaps a year or so ago and the idea that there's a certain number of things on deck, to use the aircraft carrier metaphor. Then there are some other things which are inside the hull of the ship in different states of readiness. And it's possible to put some Kanban limits on those different states of readiness.

So if you think specifically about an aircraft carrier, there may be two aircraft on deck with the fuel dock, and the pilots are sitting in the aircraft, and they are ready to take off at a moment's notice. So that's completely ready to fly, and the work in progress limit is two. That's perhaps constrained by the physical size of the deck on the carrier, but nevertheless it's a work in progress limit of two.

Inside the hull of the aircraft, they might keep a half a dozen aircraft which are sight ready and perhaps fueled up, but there are no pilots. The pilots are off in the mess room, playing cards or something. They're not sitting there in the aircraft.

There'll be some other ones that will be under repair, or that have just come back from a mission and need to be cleaned and serviced and refueled. So, we can have those same concepts in software development.

We've seen some people divide the backlog up into three different states of readiness, where they might have anything from two to 10 items which are ready to go, in terms of their analysis and the understanding and any proprietary work that may have been needed for them, research or sub-party products that need to be integrated, or environment set up for installation.

Further back in a lower state of readiness, perhaps where the requirements are not fully elaborated, there may be a larger number of those; 10, 20, 30, 40, depending. Further back there's just a list of ideas or outlines, and there may be a much larger list of those. And that stuff sometimes gets referred to as Top Stream Kanban. That may be a topic for a future book for me. Not just that one thing, but it's likely to feature as a chapter in a future book.

Joe: You put a task up there in the ideas and they have to get broken down into user stories somewhat, taking a thing from Agile, into something manageable. Do you worry about breaking them down into different sized user stories when you go to Kanban or not?

David: So, I think there are a couple of questions hiding in there. First of all, the idea of breaking something down and having a two tier hierarchy of coarse-grained features, maybe like in Epic extreme programming and then breaking that down into smaller, finer-grained items, like a user story. That's very common, and I do address that a little in the book in the chapter on scaling. And we have plenty examples either in the book or the teaching conference presentation materials, pictures of forms. At least one of the electronic tools supports hierarchical decomposition.

So that's fine. There is a question about how fine grains do you get, and my preference is to stick at the user story level and not to break things out into tasks. But I do in terms of teams including some of my clients really do break things down into tasks. My feeling is that when you have a team or project level boards and you're breaking things out into tasks that the boards get very busy with lots and lots of tickets.

My recommendation is that when you get to the task level, you do that as personal Kanban in your cubicle. You have your own little mini white board there, with your own tasks breakdown. But then you also asked about the different sizes, and it does make sense to partition a Kanban system by size or order of magnitude of the size of items.

And we see teams that often adopt three different swim lanes for small, medium, and large where those terms refer to really different orders of magnitude, small being perhaps one or two days, medium being one to two weeks, and large being one to two months.

Our very famous example would be Phidelis in Brazil, Alisson Vale's company. Alisson was one of the winners of the new Brickell Key Award that was given out at the Lean Software and Systems Conference, and is really one of the leading components of Kanban in the world and has really built a whole software company around the concept.

At Phidelis, they have four work item types, and they have three different sizes: small, medium, and large. So altogether they have 12 different types that they're tracking. They track different matrix in terms of lead time and due date performance, and variability in the lead time across those 12 different types, the four types of work and three different sizes.

I think that that's a highly recommendable practice, because for any one combination of a type and a size, the variability in the lead time will be much lower. And therefore you can set much better customer expectation and you can exhibit a much higher level of predictability.

Joe: I guess I'd like to finish up here with a couple things, and I appreciate all your time. I could go on and on for hours just because of my interest in this subject! But David, what do you do? I know you're a management consultant but how do you find your clients? Are you doing some different, you mentioned a conference next year in LA. You just got over one in Atlanta. But could you tell me what's going on in your life a little bit for our listeners, if they'd like to get a hold of you?

David: The way to find me is to look at the channelkanban.com website or the djandersonassociates.com website. And follow me on Twitter @agilemanager. I travel around a lot, and I do a combination of speaking engagements, conferences, training classes and coaching workshops, both public classes, and also private for specific corporations.

I do that pretty much all over the world. In the first half of 2010, I'll have been on every continent with the exception of Australasia. So I've been in Brazil, Argentina, South Africa, Israel, I'm going to India next month, all over Europe and quite a bit of travel within the United States. I have one or two associates who work for my firm and do some similar things. We also do some coaching engagements for private clients, helping them to introduce Kanban ideas in their organizations.

I personally do less of the day to day coaching, that's more a role for my associates. I also do management training and more strategic corporation work with the senior leadership of organizations. Helping them to diagnose problems and to come up with a strategy or direction on how they might improve.

Joe: So, the best site to get a hold of you, the most comprehensive site, is the David J. Anderson site?

David: Not at the moment. The existing agilemanagement.net site has all the content. But we're in the process of doing some changes there, which will be public very soon. The best URL is to use the channelkanban.com. At the moment that redirects to a page on agilemanagement.net, but that'll be changing very soon.

Joe: I'd like to thank you very much. This podcast will be available, a portion on the Business 901 blog site, and also the iTunes store under the same name, Business 901. So again, thank you very much, David. I appreciate it.

David: Well, thank you, Joe. Thanks for having me. I appreciate it, too.

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*