Business901

Podcast Transcription

Implementing Lean Marketing Systems

Lean Agile Software Train

Guest was Dean Leffingwell

Related Podcast:

Lean Agile Software Train, part 1

Lean Agile Software Train, Part 2

Copyright Business901

Business901 *Podcast Transcription*

Implementing Lean Marketing Systems

Dean Leffingwell is a consultant, entrepreneur, software executive and technical author who provides product strategy, business advisory services and enterprise-level agility coaching to large software enterprises.

Mr. Leffingwell was founder and CEO of consumer marketing identity company ProQuo, Inc. Dean has also served as chief methodologist to Rally Software and as a business consultant to Ping Identity Corporation and Roving Planet, Inc. Formerly, he served as Vice President of Rational Software, now IBM's Rational Division, where he was responsible for the Rational Unified Process and promulgation of the UML. Previously, Leffingwell was co-founder and CEO of software tools company Requisite, Inc., makers of RequisitePro for requirements management, which was acquired by Rational. Mr. Leffingwell was also the founder and CEO of RELA, Inc., and publicly held Colorado Medtech.

Dean's Website: http://www.leffingwell.org
Dean's Blog: http://scalingsoftwareagility.wordpress.com/
Dean's Books (Amazon): Scaling Software Agility: Best Practices for Large Enterprises
Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)

Copyright Business901

Business901 *Podcast Transcription*

Implementing Lean Marketing Systems

Joe Dager: Welcome everyone. This is Joe Dager the host of Business901 podcast. With me today is Dean Leffingwell, a 30-year software industry veteran who has spent his career helping software teams achieve their goals. He is a renowned methodologist, author, coach, entrepreneur, and executive. He founded Requisite Inc., makers of Requisite Pro and served as its CEO. And as Vice President of Rational software -- which is now part of IBM -- he led the commercialization of the Rational unified process. As an independent consultant and adviser to Rally software, he has helped entrepreneurial teams and large distributor multinational corporations implement Agile methods at scale. He is the author of "Scaling Software Agility" and the leading author of the Managing Software Requirements.

Dean, I would like to welcome you and congratulate you on your newest book, " Agile Software Requirements." It's simply the best book I have seen on describing the structure of Agile from a team, program, and enterprise perspective. I think you did a masterful job of covering mainstream thought and how to apply it in scale. I guess I have to start out by asking what prompted the book.

Dean Leffingwell: Thanks, Joe. What prompted the book was actually a full circle in my thinking over the last eight or 10 years. I wrote about software requirements 15 years or so ago when we're all thinking more linear in our thinking, more stage gated and the plan was if you could get the requirements up front, the program would go better than if you didn't. And that was a time when software was more brittle, frankly, and it was harder to read factors and we didn't have the tools we have now. So I think the economics drove us to invest more time up front trying to get the requirements right. I evolved through the Rup, went through my time at Rational as responsible for commercialization of the Rup into

Copyright Business901

iterative, incremental development. During that time of course Agile and especially a small team Agile was just a really strong trend in the industry over the last six to 10 years or so.

So I wrote the software "Scaling Software Agility" in 2006 and published that in 2007 based on my experiences and taking some of those smaller team methods, XP and Scrum hybrids a few other feature-driven development and even DSDM techniques wrapped in. As I did that the teams got better and better, but they kept coming back to, "Well the hard part is knowing what software to write and when we've done it well." It's not just a matter of no defects; it's a matter of when our customers think this is the right stuff.

I was prompted again and again by people that I work with by saying, "OK how requirements work in this?" If you look at Agile developments from a Lean perspective -- software development from a Lean perspective -- particularly, the value stream, if you will, what we deliver are just bits and bytes, this abstractions that we call software that reflect a customer's needs.

We don't have tangible widgets to put our hand on and to measure and test and to count and to stick up a Kanban card when I need to load another resistor in the bin. A number of people prompted me to say we'd like to understand the value stream better, and that comes back to how requirements flow through the system. From maybe a high or poorly expressed customer need into a high degree of specificity and user storage and acceptance criteria and test cases.

So, a few years after writing "Scaling Software Agility" I decided to one more time take a look at perspective, look at requirements, but in this case from an entirely different, totally bottom-up Agile perspective. I also discovered, of course, in scaling Agile that we had

Copyright Business901

some wonderful new constructs in Agile that I had no part of inventing, mostly driven by XP, particularly the user story.

Card conversation and confirmation alliteration became the sweet spot, but as I was working with these larger enterprises, we discovered that by far that's not enough. We need to have the ability to describe features to our customer and large scale enterprise level epics. We need relationships amongst those things so we can track when a feature gets done. We need to talk about non-functional requirements again because if you miss a regulatory guideline or standard or some EPA regulation for emission in your engine control unit, you're going to fail just as badly as if you missed any other functional requirement.

So basically, I'm compelled to write about these things, not now. I'm certainly not writing anything right this second, and that's what brought me back to "Agile Software Requirements." Also when I wrote "Scaling Software Agility" we were relatively new in the market in terms of making that case and consequently I think over the last three or four or five years we've developed just a lot more sophistication in our practices, while still being Agile, that scale to the enterprise level that I wasn't able to express in that book.

I basically had a chance to think about Agile and scale again, and I structured the book as you know in three parts -- requirements for the team, requirements for the program and requirements for the portfolio. You can either take it a little at a time if you are an enterprise. If you're a team then only the first pieces matters or if you're working with maybe a 100 people the second piece of the program. But if you're 300 people or 3,000 people then you're going to really have to really take an enterprise view. So that's why I structured it basically in those organizational levels.

Copyright Business901

Joe: It actually is an outgrowth of the "Scaling Software Agility" book because I think so many people lead into a subject then they learn so much more but this is a deep dive into that, isn't it?

Dean: Yes, it really is. In "Scaling Software Agility" I had a different audience, and I remember my editor actually cautioned me. I wrote that for team leads, managers and executives who needed to consider the Agile paradigm and their enterprise and he warned me, he said, "Well, number one there's not as many of them as there are practitioners, and number two..." He challenged me a little bit and said, "Are you sure they're lifelong learners?" And I said, "Well yeah, they mostly are and those that aren't need to be." I wrote the book for them, and it's effective in that market but it's not a book that was driven directly to the practitioner. It didn't provide a lot for the developer and tester. So as I learned more and as I saw teams execute Agile successfully more, I decided to go right to the sweet spot again which is, A, where there is more readers and, B, where there's more people that might get a little bit of help out of reading the book.

It is Agile at scale, and it is Agile requirements at scale and if you think about it, all we really have is requirements. We take ideas and turn it into software that you can't touch. So it's not that requirements are one of the nine random artifacts that you might talk about in Agile development. It's the one that carries all the value to the user.

Joe: I think that people get mistaken from a lot of this talk or when I talk to people about Lean-Agile software, it's becoming such a common way of businesses. The way they're looking at things with the iterations and learning cycles. It's a great learning experience.

Copyright Business901

We could drop the software off of that, and the Lean-Agile Enterprise is common place today. Maybe not commonplace but it's getting there.

Dean: It's definitely getting there. I work right now with enterprises that have quite a mix. Some of them make hardware, big iron. Others make large-scale actually sell databases. Others provide service-oriented approaches delivering various types of data and others do traditional enterprise software. They all want to be leaner. I think if you did a poll right now of basically large enterprises and large software enterprises, and you said are you doing Agile development? I believe that 80 percent of them would say yes. But when you pull the covers back a little bit and you ask some questions -- because I talk to those types of enterprises quite frequently -- and you say well what did you do last year in terms of Agile?" And they said, "Well we had 20 or 30 Agile programs, and they are quite effective." And you say, "What's the target for next year?" And they say, "Well we're going to double that. We're going to roll out Agile to twice that.

Then you ask a further question that says, "OK. How many programs do you think you're going to execute in your large enterprise next year?" And they'll say "300." So the question then becomes at that rate of adoption it's going to take another decade before you're really an Agile enterprise. So that's one caveat.

The second caveat is nobody can afford to say they're not Agile. If you call any large company today and talk to the CEO and do an analyst interview and you say, "Is your enterprise Agile?" There's only one answer to that, and the answer is yes. So there's Agile, and there's Agile. There's little Agile, yeah, we feel pretty Agile, and there's big Agile, which is the real Agile practices. I would say the enterprise adoption of big "A" Agile is still

Copyright Business901

quite limited. I think we're at maybe at five or 10 percent market adoption in terms of the number of practitioners today developing software, maybe one in 10 or one in 20 is actually exercising XP and Scrum like serious very short iterations high quality Agile.

Joe: Well iterations is a big word that people use and bantered about, and I'm not sure everybody really conforms to what an iteration is? It is a complete loop. I think sometimes people forget that.

Dean: Well, I tell you, that's a loaded word and that's one of the challenges that we have when we talk to companies that say they're doing Agile and some of this is driven by the Rup, which was iterative incremental and often misapplied. In the Rup; the inception, elaboration, construction, and transition also often got translated into business case requirements design coding and test, which was a poor interpretation of the Rup. But I see things and hear things all the time. I ran into the other day, for example, a group that was doing six-week coding iterations. A group of 25 people describing to me their Agile prowess. They were developers only, and they would do their code in large blobs six weeks at time. It wasn't tested; it wasn't validated, and it didn't have much feedback at all.

It was just the next big increment of code. Well, that's iteration but it's not an Agile iteration and there's a pretty big difference. So you run into that type of thing. I also ran into the other day a Scrum team requirements analyst.

I got to tell you I didn't know what to make of that. I know what Scrum is and I know what requirement analyst are, but you don't put six of them together and do a daily stand-up and call that Agile or Scrum, it's none of the above. I think there is just as much confusion and mislabeling as real Agile. I think that's a real danger for the industry because 24

Copyright Business901

months from now everybody is going to say they're doing Agile, but who's really going to be doing test driven developments and collective ownership and building really serious high quality software in a short time box? Or are they going to have sprints of requirements and sprints of design.

Joe: What are the keys to building a Lean-Agile software enterprise then?

Dean: Working towards my one annual presentation, I do one trade show a year. I go to the Agile conference every year. It used to be Agile XP, and that has just grown tremendously and it's a high-value conference. So I've been reframing the way I describe the way I see large enterprises being successful around five basic keys. Now I'll describe those keys here. It might be a good way to approach the things we're talking about. Key number one is, not everything is a user story. A user story is a really great invention. It came from XP, the user voice forum, which is, as a user; I can perform this activity so I can get business value. This is just a fantastic breakthrough in our thinking. But user stories are small because they fit inside iterations, and there are lots of them. So at the enterprise level they don't scale very well and that's why you see in my book "Agile Software Requirements" a three-tier, three-approach of user stories, features and epics and even a fourth year if you will which is the investment themes that drive all that.

So my first bullet thought is got to be smarter than just taking everything to a user story. The second key is at enterprise scale the way we've rolled out Agile historically which is a team entered a time or a Scrum mastered a time is simply too slow. So, I like to think in terms of Agile programs and not just Agile teams.

Copyright Business901

Business901 *Podcast Transcription*

Implementing Lean Marketing Systems

Agile programs are oriented around this construct that I've roughly described as the Agile release train, which is a synchronous set... Or a team with a synchronous set of iterations all timed together, working together, doing their planning, their commitment, or execution of their feedback together as one fractal above an Agile team, which has exactly the same pattern.

Thirdly, architecture emerges in Agile. It's part of the manifesto, but I got to tell you at enterprise-class scale it can emerge in some really bad ways. And if you're sticking together an entire portfolio or you're doing enterprise-class business applications you can't expect any one team or any half-a-dozen teams to have a real view of how that all needs to work together or what types of governance needs to be applied to those systems.

So, as I wrote in "Scaling Software Agility," I'm a big fan of architecture and system architecture in the enterprise. And I call that intentional architecture, which may be flies in the face a little bit of some of our thinking. There's some of the agilest thinking about architecture, but it can't be strictly emergent in my view. So enterprise systems record potential architecture and I have a model for that. That was brought to me by others that I had the great privilege of working with a couple of enterprises that were quite Agile and also needed to re-architect their systems. I call that re-architecting the flow. So that's item number three.

And fourthly if the teams are Agile and the programs are Agile, but the enterprise isn't, if the portfolio is still being treated with fixed budgets where people must be assigned to tasks. And in month nine, in order to justify keeping them on your program or the portfolio function is driven by those who understand requirements independent of implementation

Copyright Business901

and communicate that and make commitments on behalf of the teams and programs, then you're not going to be very Agile either. And there is a set of legacy minds that are there that I think we have to cope with. So that's item number four.

Lastly, and this is where I've been spending most of my time lately, and so many things are obvious in hindsight. I think we're all brilliant in retrospect. It's just the going forward part where we question our sanity. Lastly, your enterprise is going to be no leaner than the executives thinking and no matter what you try to do at the team or program level, until your executives are totally on board.

Not with Agile so much because Agile for them is good, sounds great, but they don't do Scrum. They either think their teams are already empowered or maybe that's just not a big concern of theirs, depending upon their cultural bias and mindset and history. So, we don't have a lot of tools for them in Agile. You can't take them the manifesto. It's useful, and they're interested in that but it doesn't drive their lives.

It's tough to communicate with high-level executives about Agile practices in a way that's meaningful for them because they don't do any of it. But Lean, they can think about and understanding the value stream and understand how certain, let's just say, governance items really caused delays in the value stream and understanding that the total tack time, the touch time, we put on a system is a small percentage of the total delivery time; getting them thinking about delays in the value stream and what causes those delays and what they can do about it, is just a better approach.

I'd like to think in terms of two sub-aspects. One is Lean education and I have one go-to source now, which is Reinertsen's new book that I like to use with executives and have

Copyright Business901

them read it and sit down and discuss some of those key principles. Then lastly, you've got to show them how their enterprise is going to look like after the transformation, and I call that the Agile Enterprise Big Picture or the scaled Agile Delivery Model. If they don't know how it's going to look later they're not going to be very comfortable with a totally self-organizing complex dynamic system that's just all going to work out.

Even though that's largely the case, I think the pattern of how teams and programs get organized and how values flows; how they're involved as fiduciaries of the enterprise level; how the governance model works; how they drive the portfolio, is critical to them. Because if they can see and after they know the current picture, they know how it works now and they know there's some challenges but they wouldn't be reconsidering it -- but if they can and after it just makes their life a heck of a lot easier. I call that Lean Education in the Big Picture.

So those five keys, not everything's user story for sure, think in terms of Agile programs not just Agile teams. Admit that our large scale systems require intentional architecture. Think about some of the legacy mindsets we've had in portfolio management and how to address those. Then also understand that we're going to be no leaner than the executive's thinking. Those are the keys that I currently keep in mind whenever I approach a barbarian at the gate of the not as Agile an enterprise it wants to be.

Joe: Let me go backwards here and start with the thinking in Lean education and the big picture here with the executives. To introduce Reinertsen's book is a pretty deep dive for them.

Copyright Business901

Dean: It is. I have the good fortune to be called into circumstances where enterprises are taking this very seriously and early on in the engagement model I talk about the executives and their key role. I think one of the challenges, I don't know if you know the Scrum chicken and pig story, all right? Which it basically says is that the people that write the software are pigs, and everybody else is a chicken. Well, I tell you if you sit with a bunch of those chickens and you look at the challenge they face or you sit with them in front of their management and look at their position in the marketplace and the challenges they face in competing, they don't seem like chickens to me at all.

I approach it a little bit differently and basically challenge them up front and say that your enterprise will be no leaner than you are and you are the ultimate drivers and the ultimate limiters on how Lean and how quick your enterprise can be. I talk about the need to spend time together, and we do that often time in a workshop, quite frequently a full two-day workshop.

This is just part of the process. It's not really optional if you will, I don't present it as being optional. Reinertsen's book is short, so that's deceptively cute, right? It's Lean product development flow. It's not a translation of the Toyota way of thinking, if you will, about manufacturing to Lean.

It is a guttural view of how things like work in process limits and controlling queue lengths and decentralization and cadence and synchronization, how those tools help drive great performance and product development. So it is a deep dive for them, but so what? Agile is a deep dive. We're looking at productivity increases that are substantial, measured

Copyright Business901

productivity increases in some of the largest enterprises I've been involved with have reached 30 and 40 percent. Well, if you got a 1,000 people, that's 300 free people.

Do we think an executive who can sit there and look at Agile slides for half an hour or read the manifesto and go, "Great, we know how to execute." So you're right, it's a deep dive but we take that deep dive and we take it together. Often with me or the Agile working group -- I work with Agile working groups and there's some pretty bright people there, and they help.

And the second piece of that, of course, the scaled Agile delivery model which is showing them how their enterprise will be organized afterward. They know how to organize now. They know they've got a large test department, and they go, " Well if a test moves into development what happens to the quality? How will stuff get tested? What about all the stuff that individual teams can't test?" So the scaled Agile delivery model is the other half of that. If I can get fixed in their minds, the principles of Lean so they can operate as Lean thinking leaders every day and give them an idea of where we're headed then their comfort goes up and they can start to lead.

I think if you think about transformations in Lean versus Agile, if you were doing a factory roll out five or eight years ago or maybe even today of Lean, you don't go to the floor and just start training the people on the floor to work in cells. You take management aside for a two week boot camp and you fully indoctrinate them and you teach them Lean and you show them who their partners are going to be in that Lean roll out, and it's their job to implement Lean.

Copyright Business901

We took the opposite approach with Agile. We called them chickens and we've kept them out because we didn't whatever reason didn't think they'd led us to where we want to be and it doesn't scale to have the leaders not lead. It simply doesn't work. I just take it up front, and I say, "We're going to be no better than you guys. Let's spend whatever time we need to get together to get better principles and philosophies aligned."

Joe: Alan Bustamante wanted me to ask you what are the top three challenges faced in building a Lean-Agile software enterprise? Is leadership one of them? Is this what we're talking about as one of your top challenges?

Dean: Absolutely. It's the ultimate limiting factor, right? A lot of this comes from our experiences. If you train Agile teams, just maybe roll out Scrum, a very effective and very simple model. Those teams will start putting pressure on the enterprise, and they'll start asking for things, literally tearing down walls. We don't like our walls or cubicles. We'd like to use that lab over there for our daily stand-ups. We'd like to plan together. We'd like to get the folks here from India with us every three months or so, so we'd like travel budgets to do that. We need better build infrastructure. I need a team to work on continuous integration. I need the time to do some refactoring. When you start pushing on managers who aren't familiar with the model and you don't understand how those things drive performance improvements over time, or if they understand that they've already accommodated to the drone of those complaints then they're going to fall on deaf ears. You'll be ultimately blocked and you'll be more Agile than you were because you might have gotten some reorganization and maybe you got some product owners and powers to make good decisions but you're not really building an Agile enterprise, you're building teams that are a little more Agile than they were.

Copyright Business901

Business901 *Podcast Transcription*

Implementing Lean Marketing Systems

Joe: Can Agile work without the whole enterprise being Agile? Can you have a separate section? Let's say, because I know you wrote about John Deere, is just the software department, Agile or Xerox, just the software is, and the hardware guys aren't. Are they still compatible then?

Dean: Well, you have to basically start that way because the actual manifest was written for software teams, Scrum and XP, which are the primary roll out models, both very effective addressing different issues. Those get rolled out in the software teams and then they start with their cadence and synchronizations and then they want to have periodic reviews and they want to have the software run on that prototype hardware and the prototype hardware wants to run on a prototype machine or whatever. So, in those cases the teams start to drive the enterprise to be more Agile. You start putting pressure on organizations because if you run Agile releases even if the software isn't shipped you're going to have software available to be shipped earlier and more frequent. So that's going to put pressure on sales and marketing if you think about how to do that.

So the answer is yes and no. Of course, to build a true Agile enterprise, everything eventually has to be Lean and Agile, but you don't start there. These are journeys not written in six-month little case studies; these are journeys measured in years and the enterprises that I've learned the most from are the enterprises that have been added the longest. They've been headed down that path already. In their second and third and fourth year of agility and they're still starting to work with some of the issues, let's say, at the PMO level.

Copyright Business901

Business901 *Podcast Transcription*

Implementing Lean Marketing Systems

Joe: When we go into the portfolio management, that's got to be Agile. There not even a question about that to be able to make it work, is there?

Dean: Well, there's a question because it won't start that way. First, I want to give credit where credit is due and not just pick on the PMO or the other PM guys that were trained. The PMO's have built governance models around the waterfall life-cycle model because we taught them that was the way to write software. We have new ways that we're writing software now, and we have to teach them those too. But in the meantime, their governance models and all of their activities are focused around the milestones and things like design reviews and test plan complete, requirement sign off, and code freezes before defect triage, we taught them all that stuff. We shouldn't be surprised as agilists when we go to instill and install Agile and they're saying, "That's just great but you still have to meet the phase to exit gate which requires that the design is complete."

Yet, if you go to those teams and say, "When will the design be complete on this project?" They'll look you straight in the eye and say, "The day we take it out of maintenance." Not in the phase two milestone review. There are a lot of legacy mindsets that are there, but they're not there because we have people that think in legacy terms. They're there because that's what we taught them.

I think we have a responsibility to educate them with Lean thinking, and I have better luck looking at things like the way milestones... Let's just say there's a design review milestone on a program, and the program is late. I ask the PMO, I say, "Well if the program is late does the review move to the left or the right?" Meaning forward in time or later in time and

Copyright Business901

they say, "Of course the review moves later." I then comment to the fact that what you're saying then is that the later the program is the less often we look at it, right?

They go, " Well that's not very Lean thinking." Because that's the opposite of what we should be doing and since we don't know if for sure it's late or right, why don't we just do periodic review on a cadence and we'll review maybe all the programs on the same cadence and look at whatever they're doing. But if they're not creating design specifications for us because they're Agile, they'll be creating code. Why don't we assess the code? Let's ask them for a few key measures around the code. Let's ask the product managers or customers or whomever how it is they see that the see that the code is developing.

Let's look at the facts. Let's look at the quantitative objective facts of how the code is evolving. Because in Agile, there will be no excuse for not having code. While they may not have some of these other things that you're used to seeing, they'll have a test strategy. They may not have a test plan because to write down their test strategy into a plan, number one; they have a model that's integral. It's no longer separate activity. Number two, they only write down what they need to write down, and they're not going to write that down just to give to somebody else. So, let's just look at the code.

So when you approach the boundary of a PMO, the way I like to approach it is that your governance model got us here. The waterfall development got us here. We're moving forward. We're not just going to throw away governance, right? We still need oversight; we need to know how we're spending our money.

Copyright Business901

Business901 *Podcast Transcription*

Implementing Lean Marketing Systems

We still need to drive the teams to the right vision and the right programs. That hasn't changed. But the practices that we're using have changed materially. So before I just make these milestones go away, let me give you some suggestions for new ones.

How about this; how about each program has a milestone review every 60 days and every 60 days we're going to look at the current amount of working code, the current defect count, the current feedback from the customers and the current plans for distributing that piece or how it's doing if it's already distributed. Get their mind around assessing the actual use of the product or actually looking at the application rather than looking at their intermediate artifacts that we used to create, the code was all being done in parallel before we couldn't really run much of it. All we had were these artifacts.

The code was so hard to change we had to try and get the requirements right, so one of those artifacts was our software specification. Well, we don't do that anymore. We do have software requirements. They're called user storage and acceptance criteria, but we write them just in time.

We want to make sure even though it can seem fairly pointed sometimes, and sometimes the PMO has looked at the mother-ship of all impediments, I don't look at it that way. I look at it as one of the elements of the enterprise that's going to be transformed as well. There again, you've got to go in with different tools.

They can think Lean; they don't care about a daily stand up or how a Scrum team works. That's not part of what they do. So taking them, teaching them Scrum it's kind of an interesting experiment, but they don't do Scrum. You got to speak in their terms, and

Copyright Business901

that's basically the business economics, how Agile and Lean thinking drives business economics and the principles.

Again, some of the principles of product development flow that I like to lean on Reinertsen's work for that they can use to help improve the overall enterprise performance.

Joe: It's really practicing Lean or re-architecting with flow is the whole point of it?

Dean: Well at the portfolio level, I like to get them thinking in Lean terms. Architecture is a different topic entirely. I spend a lot of time with system architects and I've got to tell you I like those guys, but my first trip is usually a bit of a roast. If you sit down with 25 or 50 or 75 system architects and start talking about Agile, you're not generally in front of a friendly audience. They've read about Agile and design emergence, and they've looked at Scrum. There are no architects in Scrum and they wonder about that model and they think it's ludicrous that in an enterprise their size they're adopting this small team model and these system architectures are going to magically occur.

But after the first hour or two of roasting, if I can get them to sit still long enough to look at Slide #3 and start talking about the way architecture has to evolve, they need to be Agile too. We don't have the option for building that perfect system or we're going to stop and make a branch and in 36 months we're going to launch this new application suite. More generally, we have to refactor what we have.

Copyright Business901

Business901

Podcast Transcription

Implementing Lean Marketing Systems

At the system level, I call that "re-architecting." That requires taking the big architectural initiatives and breaking it up into smaller chunks, and as we do that, those smaller chunks have to be able to deliver immediate value.

You can't define a system that says until the tenth chunk works, none of it has value, you've got to design a system in such a way that you can roll in architecture incrementally. I had the great fortune to hook up with a company that was in their probably third or fourth year of an Agile enterprise.

They weren't huge, they were about 300 developers or so, and there were only six or eight system architects, but they were sharp, and they were trying to re-engage with their Agile enterprise.

We worked together and developed a conceptually simple Kanban system, basically just a little state management system that said "Where do our ideas come from? How do we get them and how do we rank them by economic value? How do we get the teams to estimate them without doing a work breakdown structure?" And the answer is relative estimating.

Then if we get something that makes the cut, and we go to our business owners and we say that we are going to make this investment in architecture, how do we bust it up in small enough pieces that the programs that are operating on a cadence can always have a potentially shippable increment.

That's new thinking. It's different. You may have working process limits. You may be talking to them about controlling the amount of that work that goes out onto the floor because the more of that you're doing in a time box the less potential end user value you

Copyright Business901

could be delivering in that same time box because you're playing in some new infrastructure.

So they need to be Agile too, but again, it's not Scrum for them, generally, or XP because there are certainly principles there. But continue to say that integration isn't a part of their model and while peer review is, peer programming isn't.

So again, I think a Lean approach and a Kanban-like approach, they resonate with immediately. So when I talk to architects, I talk to them about Kanban and I talk to them about a poll-based system that says "OK when are the teams ready for a new bite, and when are you guys ready for a new bite from the business backlog."

Joe: For me to understand the enterprise system is it fair to say that's where this architecture here is turning management and what they want into what to do?

Dean: Absolutely. If you look at the role of the system architects in the large enterprise, there are lots of inputs; some of these are basically big business initiatives. So let's just say an acquisition, let me just make that case. So an enterprise acquires a couple of other enterprises. That's the way that most enterprises get big, make no mistake about it. Well, all of a sudden the rules are different. Now I've got two or three business units operating largely on their own, and I've got to stitch all of that together. I need maybe a single sign on or common authentication model, or I need to not force large scale DBMS on my client when they buy my suite of products.

So the challenges that they face aren't just things that just kind of grow bottom up from what the teams are doing, they're things that come top down. There are also things that

Copyright Business901

need to happen, for example, maybe just cost. Maybe our deployed application system is costing too much, or it's too slow, so this whole big system that 300 people built is operating at a third the speed it needs to. Well, what one team can improve that? None. What one program can improve that? Well, it has to be improved in chunks. So they have that responsibility.

There's also a governance responsibility. One enterprise I know is dealing with credit card compliance. The security officers came in, and they said, "Well, we have this new compliance module for credit card processing", and it affected most of the back office systems. Well, that's not an organic thing that's a top down thing. It's foisted onto teams by external forces.

So I believe that system architects play a legitimate role not only in design of the enterprise level architecture, but in security and governance, and that got to be an empowered role in the enterprise but it still has to be Agile.

You can't do it the old way either. You can't basically shove these requirements onto the team and detailed specifications or give them all the how to do it, and say it's going to work this way and then stand back and hold the teams accountable for delivering it that way. The teams are accountable for delivering the work, so there has to be a collaboration, there has to be an agreement. In the end, it's the teams that have to be on board with those decisions because they're the people that write the code.

One of the simple principles that I like to use is the teams that code the system design the system, and I've gotten into a couple of good discussions with enterprise architects, and

Copyright Business901

they argue about that. In the end, I'll kind of look at them and say, "Well obviously you don't feel that you're part of the team that codes the system.

Why is that? "Well, it's because I'm on the third floor, whatever, and I work with system architects rather than team," I say, "Well, it seems like to really be Agile, we're going to have to get you guys working together a little tighter." So a small vignette in the middle there to kind of describe the fact that, A, those barriers tend to exist and, B, we can't leave them in place if we want to be effective as an Agile enterprise.

Joe: Well there's one interesting thing that you brought in the book and it really struck me, is that you're one of the few authors I've seen that have really combined Scrum and Kanban in the book together. Because they seem...

Dean: There's a method at work.

Joe: Are they really compatible methodologies?

Dean: Well Scrum and Kanban are fundamentally different, and I try to avoid the "method" words but you can't entirely. Now after a number of years of trying, we're very effective in scaling Scrum in the larger enterprises. I know how to do that. It has a plan basis. It has a time box, which is great because time boxes help control the addition of unplanned work. Because it has a plan basis, you can extend that plan basis to the enterprise, and you can execute larger scale planning events that plan in spirits.

I believe that we have some very effective models for that. I've seen Kanban work quite well too. I've not personally seen it work at very large scale. If you see a large group of

Copyright Business901

people doing Kanban, it's typically a couple or three teams that have the ability to work together and pull from a common backlog and potentially have a daily stand-up.

I do see Kanban work well. I see it working well in IT, I see it working well in maintenance, I see it working well wherever there's a very transactional basis to the business and indeed I introduce Kanban, I introduce it in the portfolio and I introduce it at system architecture.

I don't personally introduce it at the team level, because Scrum is more mature and has some things like stronger senses of estimating, user stories, all of which can be done in Kanban, but don't necessarily come with the freight, backlog estimating, planning and time boxes, and commitments to deliverables in time boxes.

Maybe I'm an old fashioned manager, but I think it's meaningful to ask a team what they can do in a time box, not to tell them but to ask them, and to commit to that. To put the whole enterprise under the onus of saying we plan and we make small commitments, and those around us can do that and Kanban is very effective and extremely efficient, but it doesn't always bring those same attributes.

Now, that doesn't put me as not a fan of Kanban, it just says that in my view and in my world I use it with respect and to good effect in certain instances, but I've not seen it applied over a thousand developers building a common thing. There may come a time where it is applied.

I've seen some pretty silly stuff. I've heard Kabaners said the elephant in the room of Scrum is dead. And I've heard Scrum say, "We're not going to talk about Kanban because we don't think it works, and it violates some of our key principles." Well, I think we need to

Copyright Business901

talk about both of those things, but we need to keep them in perspective. I think they both provide value.

I'm also sorry. You didn't mention XP in there, I'm sorry that there is less emphasis now in XP than there used to be, because if you think about Scrum and Kanban both, none of them really put any emphasis on how the code is written.

If you want to get a team to do a better job of adhering to coding standards, or collective ownership or developing T-skills or having higher fidelity code, that map to acceptance criteria that they determine in advance, that's XP. So all of these things can work together, and if you're at the enterprise level, it's probably kind of foolish to think that any one of those things alone is your right answer because those are all really sets of Agile practices that have to be integrated.

That of course creates a challenge, and I try to address some of that challenge by bringing the various aspects of that into my book, so that people could see how XP-like practices work in the context of Scrum, how Scrum can scale, and how around in certain places in particular, architecture portfolio, maintenance, etc., Kanban may be a better choice for you.

Joe: I think you did an excellent job of that, Dean. When I read the reviews of the book, one of the most mentioned things that you introduced was the "Agile Release Train." What is that?

Dean: Let me just give you by parable, actually by story and not parable in this case. One of my contacts, clients, approached me one time, and he said, "You know, I've got 13 Agile

Copyright Business901

teams and I really like these guys. They've adopted Scrum, and a little bit of XP and their velocity is going up." He said, "I tell you; it's the 12 teams. I've got the 12 tribes of Israel here. I don't know how to bring them together into a common vision anymore." As you get into the enterprise level, you typically find that there are lots of developers, three, four, five hundred, five thousand, and they work together in large groups, and those groups have to coordinate their activities.

If you roll out just Scrum, for example, they're going tend to make a strong team, but that strong team has some borders around it, which is my backlog, my commitment, etc, etc, and your backlog and your commitment are a lot less relevant to me than perhaps even than they used to be because I've now got my own objectives.

So the "Agile Release Train" is a construct whereby we take large groups of teams that need to collaborate, typically on a single asset. Sometimes they just need to collaborate because they're all the resources.

So they may not be working on a single asset, a single program or applications, but they may be all the resources we have in a particular department. So we bring them together. We basically take a fractal. The teams individually have a known model. It's called "plan, commit, execute, demo and retrospect," in two weeks. Well, we do that, let's just say for the sake of argument, eight weeks, and we have 10 teams plan together, commit together, execute together, demo together and have a retrospective together.

So from the standpoint of the enterprise they're not thinking anymore about those 12 teams, they're thinking about a single actual program. Also, if an enterprise is just heading

Copyright Business901

down that path, it's not hard to get them all together. Get 50 or 100 people in the room; train them all at the same time in Scrum or whatever your particular variant is.

Then have a release planning session all at the same time and plan for sprints, not just one. They'll still go away and plan and execute one sprint at a time. But basically it's a fractal above the team, and it's an actual program and it has to behave just like an actual team does: plan, commit, execute, demo and retrospect. That basically gives you an organizational unit which is not necessarily a business unit or a product or product line; it's a value stream, and it gives you a governance model.

Who is going to run the train, or who is going to at least get the teams together. What metrics are we going to use to measure the train? How are we going to feed them content? That is basically a model of aligning the teams to a common mission, periodically, and putting that team or that program or that value stream in continuous product development flow.

Once you make a decision that we're in Business A for a while, I shouldn't have to do a project charter to get a new piece of content approved. I should just take that to the train and say we're going to make this investment, you guys pick which pieces of content make the most sense, and that's within your control.

It eliminates a lot of the project-like thinking and all of the starting and stopping and just says we're in this business for a while, boys and girls, so let's plan together and execute together.

Copyright Business901

So, it's a team of teams. An Agile program is a team of teams operating on a common cadence. Not necessarily a common release schedule, they may release independently, but a common planning cadence and a common demo cadence and a common retrospective cadence.

Joe: Does this really get rid of project management? Are you just having teams talking to teams, and there's not someone orchestrating it all?

Dean: No, it doesn't get rid of project management. On the contrary to some beliefs, some of my best friends in implementing Agile at scale are project and program managers. Their roles change, for sure, especially at the program level. They're no longer just kind of program portfolio managers where they're moving content around and giving teams tasks and assignments, they become large-scale program facilitators, but running a release train is essentially a program management function. There's a tremendous amount of logistics involved in getting the train together. The train does not run itself.

The train consists of people from lots of different teams and departments. There will be product managers from the product management group, marketing. There will be program managers or sorry operations people, developers, and testers. Those may come from a lot of different department. You think about it It's a large continuous cross-department operation.

Send me a good software project manager, program manager and we can make that work. Without it, you've got a couple of VPs meeting and going, where's the room? Who do we invite? How do we drive content because that comes from marketing? They are largely self-organizing, but they're not self-led. Self-organization at the enterprise level only goes

Copyright Business901

so far, and as executives, we need to have some, I guess, what the new product development game we call has control over that.

At the team level you may not have a project manager per team, but four or five teams might have a hard time getting the equipment they need or support from the field or feedback from customers or whatever. If project managers get Agile, they become the glue chips that hold it all together.

There are also, of course, old-fashioned project managers and program managers that are heretical and dictatorial. They don't work in the model, but I think if you eliminate project management approach, program management is a function in a large scale enterprise. You're going to end up putting a lot of that back later, but kind of like architects you need it, but you don't need like you had it before.

That's just another kind of domain of people that we need to have on board. Many will make the cut. Many will get it. Many will resist because that was their control, and that was their value added was to tell teams what to do. We don't do that anymore. We put teams in a situation where they can do the right thing, and the content authorities, whomever it is that determines what the right product compliment is, which typically comes from marketing or product management. They give the teams the content of what needs to be delivered. The teams are organized in an Agile way, but nobody tells the teams what to do. They figure that out for themselves. That's what makes an Agile team an Agile team.

Joe: So, they're really supplying the what or what to do or what for.

Copyright Business901

Dean: Right. They're facilitating a process of getting the what which comes typically from product management or the business owners or whatever in front of the teams that do the how, right? There's a little bit of organizational work on the what because getting the portfolio working and getting content fed to the teams at release planning boundaries or PSI planning boundaries, potentially shipping increment boundaries, that's real work, too. That program manager doesn't determine the what. That has to come from the product group or the business owners. The program manager facilitates the process, so they create a process whereby the what is put in front of the teams who do the how, and they do that forever. So, there's always a new what, and the teams do the how. That's where the teams get empowered on both sides.

When you make the translation that says, the program managers also do the what, and because they manage programs, they tell the teams the how. Well, you do have a point of control, but you also have a point of micromanagement, and you have people telling people to do the what that aren't actually developers and aren't involved in the what. So, that part of the model is not effective.

Having said that, leave me some project and program managers if you want to execute Agile on an enterprise scale because you're really going to be hurting without them. In many of the Agile working groups I'm working with, the strongest people I have, the people that are really driving the organization are project or program managers that might even work for the PMO who totally get it. They're the ones that are driving the hardest to put this new model into place.

Copyright Business901

I have a lot of empathy for those who understand the model. I have less empathy for those whose needs are more for control. As one of my mentors once told me on one project that was struggling, he said, you know, I think the executives' need for control there is greater than their need for results. That's tough. Give me people with the need for results, and we'll create a model where the teams are in control of their own destiny.

Joe: I think that's one of the problems of extending Agile, let's say, out of software because you hear all of these terms, like self-managing, self-directing, and it's foreign to the product managers and people at the enterprise level. It's real foreign to them, and it causes issues with them. It's not like they want to jump into that fray.

Dean: Think about it. Put yourself in there. Let's just say you're a VP of a company, a VP of development, and you control maybe three or 400 people in an enterprise that has 1,000, and we present them with a model that says this is all going to self-organize and self-manage. It seems ludicrous to them, right?

How is that going to work? And it rarely does work. It has to be led, and it's a facilitated process, right? It's a led process where the teams largely do self-organize, and the programs organize themselves, but they don't do it totally independently. The release trains don't just form themselves. You have to say no, here's a place where this is a great place to bring these people together. The people in release trains don't even have the authority to fly themselves to meet, and who's going to put the agenda together? Who's going to create a schedule? We need those leaders, and that's why you started with number five. We're going to be no leader than the executive, so we need the people that have brought our enterprise to this level, but just like we need the development teams, we

Copyright Business901

need them different than we used to have them. We don't need to hand them SRSs. We need to give them vision and feature sets and now create the SRSs.

We don't need to tell every person what to do or every team what feature they're working on. They can do some self-selecting. They can do some volunteering. They can draft features, but we are responsible for creating an enterprise organization and process model that allows that to happen, and that's still the executives' responsibility.

As leaders and managers, it's great to go Agile, but we're not abrogating our responsibility as fiduciaries of the enterprise. We've got to know what they're doing, and we need to know how they're doing it, and we need to know how well they're doing at it. But that's different from saying we're personally making all that work happen.

Joe: When we get back to the very basics where we started from, Agile requirements, forming them and creating them is really what makes it scalable.

Dean: The content flows through requirements, and as I've described in the book there's different levels of content. If you go to the portfolio level, it should go back to those executives. They're talking about six or eight really large-scale business initiatives. Well, those aren't user stories. Those are investment teams. That's the way they want to spend their money. And then, they say, OK, one of the things we need to do is we need to get single sign-on across all these things. Well, that's an architectural epic. So, it's true that I can express it as a user story, but it's a much bigger thing than that. So, I think of it as either bottom up or top down. Bottom up, teams work with stories. Programs work with features because features deliver real value, features, and benefits. Every product manager's been taught that from day one.

Copyright Business901

Above that, we have to think about that at the architecture and portfolio level, the epics. And above that, we have to make conscious decisions about how we're going to spend our money, and those are the investment teams. So, they're all requirements. They're all forms of requirement, right? It's a requirement that we spend this much money in this domain. We've made a requirement that says we're going to implement single sign-on across these five teams.

Now, that becomes a requirement for number A to implement single sign on the same way all the other teams did it because we don't want to support ten different models, and below that some number of teams are going to end up with a bunch of stories that say, OK, as a user I can log into this and access this other application, so they're going to get user stories out of that.

It's a hierarchy that works at different levels of abstraction so that each organization gets to think about the business the way they naturally think about the business. Portfolio managers don't think in terms of user stories, and teams don't think in terms of investment teams. That's fine. They're paid for different things.

This slight separation of concerns actually works for us. We just have to give them a language to talk about it, and Agile with just stories or maybe just stories and epics and no non-functional requirements or features or investment teams isn't quite rich enough at the enterprise level. Now, at a small team level, a small program level it absolutely is, but that's not where I work. I work where 300 is a small number not a big number.

Joe: Would you like to add anything that maybe I didn't ask about, the keys to building this enterprise?

Copyright Business901

Business901

Podcast Transcription

Implementing Lean Marketing Systems

Dean: If you understand Agile, and you understand the enterprise and you think about these five things, you'll probably get it. They're not separate initiatives although there are places where starting a release train is one type of activity and getting the teams trained in Agile another, it's all part of the gestalt that it takes to really lean this thing out. You don't have to do them all at the same time. You can start bottom up because you can't have a Lean enterprise if the teams can't build working software in a time box. I don't think there's anything wrong with a bottom-up approach, getting as many teams Agile as we can, as quickly as we can, but as soon as you have five or 10 or 20 of them starting to push back on the system, you've got to have a broader view. You've got to abstract up a notch. Look at the enterprise. Look at the needs of all the key stakeholders in the enterprise and have Lean and Agile answers for all of them.

Joe: Could you sum up the top three challenges that you're going to face going about this process?

Dean: Well, I think in terms of lessons learned, there's three or four kind of weak spots in there, if you will, that we think about, and one is that the Scrum brings this product owner role which people go, OK, this is a product owner and they manage the backlog. That's a really big deal because that also is a person that makes decisions on behalf of the enterprise. They're typically engineering. They're often in engineering rather than product management, so that's a bit of a power shift or power struggle. That is one case absolutely where lesson learned is it's pretty easy to underestimate the impact and importance of that role and the training that's required. Another lesson learned is this isn't just Scrum or Scrum and Kanban or Scrum and Kanban and Agile portfolio management. The teams as part of Agile need to learn to write better code. The cleaner their code is, the faster they

Copyright Business901

go. If the system is Lean, the higher the quality, the faster they go. If the system isn't Lean, in order to get high quality, I have to test more at the end, and that slows things down.

So, I'd say bullet number one is Agile is basically the role of the product owner is key. Bullet number two is these technical practices actually matter; therefore, if you're rolling out just Scrum, you've got to be thinking about TDD and collective ownership and pair-programming or, at least, peer review. A lot of people only rolled out Scrum because pair-programming was too difficult a cultural thing to implement, and in year three and four they start selectively pairing because they're diving into areas of code that they're uncomfortable with or they went to increase the code quality. That's the second element.

The third element is probably what I mentioned before. It's inadequate to hope that the executives simply understand what we're doing, right? It's inadequate to hope that they support that we're doing. It's adequate and only adequate if they lead and drive what we're doing. That's where I like to spend my time, is helping those executives regain the control.

To a certain extent, they have to be in control of their destiny even though this is a largely self-organizing process, but get them in a position where they're driving the behaviors that drive the Lean-Agile enterprise. Even though it's not like what it was before, it's not through their spreadsheet of allocations, and it's not through their individual productivity measures, and it's not through micro-managing the teams. It's through driving vision to large programs. They're still in control of their destiny, and I think they need to be.

Copyright Business901

Business901 *Podcast Transcription*

Implementing Lean Marketing Systems

Joe: I think it's interesting the way you put that because it is the kind of change that we're seeing created in a lot of organizations. From a marketing standpoint, I talk about co-creation and collaboration with the customers and everything in a service dominant type society. You're driving the value in use, and that's what you're trying to do as a manager. The value is not in the resource allocation; it's in use in the development side of it.

That is happening, and that's the same thing that's happening in the exterior of the company. Yesterday I looked to buy some printer cartridges, and I could buy a printer cheaper than I could buy the cartridges for my old printer. I really debated about it. I was giving up a few things. It goes back to that value in use, and you charge for the value in use. You charge for the printer cartridges. I think the organization -- that may not be a fair analogy --, but it's what's happening. The value is in the development. It's not in the organization anymore. It's what's coming out of it.

Dean: Exactly. Our customers don't buy code from us, and they don't buy tests from us, and they don't buy architectural designs from us. They buy value so that manager's role is solely to facilitate value to delivery, and our model to facilitating value delivery is different now. It used to be, understand it, write it down, and instruct teams to do it. Well, that worked at one level of complexity. Now, it's here's a vision. Value is very much in the details, and value is going to be determined by our customer in using this stuff in real time, over time at a high level of specificity. So, what's the best way to organize for continuous value delivery, and I know Lean and Agile give us better answers than we had before. They won't be the end answer any more than any of our other answers over the last few decades were the end answer, but they're where we are today. It's the tip

Copyright Business901

provision of our best thinking. It's the best we have, and it works better than anything else we've been doing, and that's why it's getting such great adoption is because it gets results.

Joe: I've enjoyed the conversation tremendously. I think there are a lot of great things you do. I recommend the "Agile Software Requirements" book for a lot of people to take a look at and share it even outside just the software community. I think there's a lot to be said for that to take a deeper dive in the way things are going.

How can someone get a hold of you?

Dean: Deanleffingwell@Gmail.com or just go to scalingsoftwareagility.wordPress.com. My blog is there. You can contact me through there. I'm pretty easy. Google me. You can't miss me.

Joe: Well, I want to thank you very much, and this podcast will be available in the Business901 website and also the Business901 iTunes store. So, thanks again, Dean.

Copyright Business901

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*

Lean Agile Software Train, part 1 Lean Agile Software Train, Part 2

Copyright Business901