Business901 *Podcast Transcription*

Implementing Lean Marketing Systems

Why Architecture is needed even in Agile?

Guest was author Jim ("Cope") Coplien

Related Podcast:

Why Architecture is needed even in Agile?

Business901 *Podcast Transcription*

Implementing Lean Marketing Systems

Jim ("Cope") Coplien is an old C++ shark who now integrates the technological and human sides of the software business as an author, coach, trainer, and executive consultant. He is one of the founders of the software pattern discipline, and his organizational patterns work is one of the foundations of both Scrum and XP. He currently works for Gertrud & Cope, is based in Denmark and is a partner in the Scrum Foundation. He has authored or co-authored many books, including the recently released Wiley title, Lean Architecture for Agile Software Development. When he grows up, he wants to be an anthropologist.

Gertrud & Cope is a small family business driven by Gertrud Bjørnvig and Jim Coplien. We focus on Agile software development integrated with the power of architecture and usability. We serve a wide variety of software development cultures including RUP, CMMI, and XP practices, but our focus and specialty is Scrum. Our services include Agile Requirements, Agile Architecture, Agile Usability, Agile Test, and Agile Organizational Analysis.

Cope is a speaker and author whose works range from programming and architecture to ethnography and organizational design. Though he writes for a technical audience, his works focus on the human element of product development. His latest work, “Lean Architecture” is as much about how architecture helps make software usable, as it is about software maintainability on the technical side.

James O. Coplien Amazon Page

Implementing Lean Marketing Systems

Jim Coplien: I think, when I really practiced this the Toyota way; the Toyota production system. I go back to the roots and spend a lot of time with the Japanese folks, to go back to the foundations of it and it's often a very difficult cultural translation, particularly into North America. I think, even more so into North America than into Europe. When I go into the companies there, they'll be doing, for example, some of the practices of Lean, but they really don't get the underlying Japanese philosophy. I mean, it gives them some benefit, but it's really not what I would call Lean, in terms of the way the Toyota people envisioned it.

Joe Dager: I think, that's what we've struggled with a lot in the US, is going to that next level. It's like we scrape off the top of the tools, but we never go, really, deep into our organizations to instill it the rest of the way. The cultural side, maybe, a little bit.

Jim: Yeah, and, of course, we have some of that in Europe. It's not only with Lean. I see this all over with the IT industry. I cannot think of a single major advance in software where that didn't happen in some degree. I don't know if we can blame market opportunism or short-term market focus, or whatever, but that seems to be a universal problem that the good drives out the perfect.

Joe: Welcome, everyone. This is Joe Dager, the host of the Business901 podcast. With me today, is Jim Coplien, a software industry pioneer in object-oriented design, architectural patterns in Agile software development. He has authored several books on software design. His latest book is called "The Lean Architecture."

Joe: Start out, and just give me that elevator speech about yourself? Finish the introduction off.

Jim: Most of the time I get up in front of audiences here in Europe, which is where most of my constituency is, and the first thing I need to do is explain my accent, and that I was born in

the US. And if we fast forward, I guess, I can include that I was an electrical engineer, and have gone through a lot of different careers and jobs over the past years, including 20 years at Bell Laboratories, where I did a lot of software things. I have been a consultant, a university professor, and even some hardware work along the way.

So, I'm older than I look, and I'm older than I sound. I guess, I've been around the block and had a lot of fun on the way.

Joe: What's your new book about? What were you after with "The Lean Architecture"?

Jim: The title is "Lean Architecture." And then, the second part is "For Agile Software Development." What the book comes from is looking at the advent of Agile software development. Almost, exactly, 10 years ago -- here in about a month, we'll have the 10th anniversary of the Agile manifesto. The 17 Agile guys, who got together and did that, had a good vision and some good ideas. But, of course, like any good idea, they're not complete. They have to be mixed with some other things.

Unfortunately, what happened in the market is, first of all, people took this very one-sided view. They took some provisions of what are in the Agile manifesto, which is this document that kind of launched the Agile movement. It came out of a meeting of these 17 guys in the US, back 10 years ago.

The other thing is that, of course, a lot of people kind of interpreted it in their own way, and then, added a lot of stuff that wasn't intended. But more importantly, Agile really was a reaction against some of the methodological excesses of the 1980s and 1990s.

And too often, they ended up throwing the baby out with the bathwater. There's a lot in there about doing, and building

software, and talking with customers, and so forth. But there's very, very, very little about thinking. So, one of the casualties of Agile has been architecture. The Agile people would tell you that architecture is only emergent and that you shouldn't do any upfront thinking because its waste and you're not going to need this. I think that posture largely came out of ignorance on the part of the young, eager folks at the time to try to do something good.

So it was well meaning, but it really ended up doing a decade of a lot of damage. What I'm trying to do is get the pendulum to swing back somewhere near the middle, where we can blend some of the principles of Agile with the more deeper and older and more classic Lean principles of thinking and upfront planning.

Joe: And is that where Lean fits in then?

Jim: The Lean part, a lot of it has to do with the upfront planning. Some of these notions that we find in Lean like decision structure matrices and this whole idea that you're thinking about what you're going to be doing before you do it. Now that doesn't mean you commit to doing it. Of course, you still have some of the so-called Agile principles where you can change your mind. But, of course, those were fundamental to Lean long before Agile came along. And since you're a Lean guy and this is a Lean audience, there's just, I think, a lot of common misconceptions in the broad software world about the relationship between Lean and Agile. I think that also explains a lot of the posture of the book. A lot of people will come out and say, "Well, Agile and Lean are the same thing," and some other people will say they have nothing to do with each other, and both of those are wrong.

Most of what you find in Agile was already there in Lean, both in terms of what's in the manifesto and in terms of the culture and the practices. There are some emphases that Agile have that are good and are useful and go beyond Lean. But probably the main

distinguishing factors of Lean that come out in the Lean Architecture book that you don't find in Agile are this notion of doing some upfront planning and looking at some issues of system form, of architecture, and of some of the focus you find in Lean on the process, whereas the focus in Agile is solely on the product. I think Lean has more of a balance of these two.

Joe: That's what I found in the book when I first picked it up and looked at it, it was like a Lean book to me. It wasn't driven, let's say, by the software side of it where I would get lost. That connection between Lean and Agile, I thought, was made very well because everybody, Jim, always tries to make their own methodology, the umbrella, the thing that's bigger than life, 'cause that's my methodology. I thought you put a nice balance to that in the book.

Jim: Well, actually that's good to hear on both counts. So first of all, thanks. That's good feedback that is general. And yes, I agree that everyone has their hammer and so every problem looks like a nail. But one of the early reviews -- I think it was maybe Trygve Reenskaug's review -- said something very similar to what you just said. It said this book isn't just selling a given technique. What it's trying to do is look at the Lean principles and it's not just a methodology. It's taking people down the path of thinking, of adapting some of these principles to their own industries, their own products, using the principles as a touchstone to figure out what they're going to do.

Joe: I found it interesting that you talk about Lean and then thinking, kind of blending them two together.

Jim: Well again, as we briefly discussed maybe before you started the recording, Lean means many things. Lean was coined by these authors of this great Lean book back in 1991 in the US. It was actually about the Toyota production system, which, of course, has swept a lot of the industry in Japan. They coined the

term "Lean" for it. That helped popularize the term and some of the approaches in America. GM was doing some really good Lean things for a while that were part of Toyota's early forays into the American market through the NUMMI plant. Ford tried to get on the bandwagon as well. But if you look at broad American industry, a lot of places that use the term "Lean," it's in a different sense than the Japanese use it. So what I've been doing for years and years and years is making an investment in trying to understand what does Taiichi Ohno mean by Lean? What did Takeuchi and Nonaka mean by what they were talking about in "*This New New Product Development Game*," which is the paper that appeared back in 1984 in the Harvard Business Review that launched Scrum? Trying to go back to those fundamental roots.

I know that Jeff Sutherland, who's the inventor of Scrum -- Jeff and I worked together -- also was very inspired by elements of Japanese culture and even as deep into issues as Buddhist meditation. Now this gets the designer and the enterprise into a thinking mode where you're not just reacting but you're thinking about how do I fit into the larger ecosystem? How do I fit into society? How am I going to make a fundamental contribution in the society in which I am indebted? What does value mean? What are my value streams?

As a consultant when I go into companies, one of the things I'll ask them is, "What are your value streams?" And they say, "Duh?" And, "Oh, what are your products?" And I'll often get an answer that "Well, you know. We have two products" or "We have 200" or "We have 40. Well, it's something in between. Well, we do stuff and we make money." And they don't really have this thought, this discipline, of what value means to them. So they're very Agile sometimes, but they're not Lean. This leads to all the more commonly known Lean consequences like waste and inconsistency and a lot of the practices and tools.

What I find in the broad industrial world, particularly in the US, is they know the tools, but they really don't know the underlying principles and structure. What I'm trying to do with this thinking, and particularly upfront thought that comes with architecture, is take people more in that direction.

Joe: So can you explain just a little bit and define architecture to me?

Jim: Oh, which one of the 5,742 definitions do you want?

Joe: Well, that's kind of where I was going with it. I was sitting here and saying what...

Jim: Well, the definition that is most commonly used and, of course, any dictionary definition is wrong, is what architecture is formed. Then for the people who want to hear more I say there's a difference between structure and form. The metaphor I use is that if you're in a manufacturing plant that makes soda pop bottles or something like that, the bottle is the actual structure -- the soda pop bottle --, but it isn't the form. The form is the essence of that bottle. So somewhere in the factory there there's a form into which they inject molten glass. After the glass cools, they open the form and the bottle drops out. Well, they could have injected molten plastic. They could have injected molten chocolate for all I care and made a bottle out of those structures, out of that material. But the form is the essence of the thing that's being built.

Now it's a little bit of a weak metaphor, but you get the idea, is that the form, the skeleton, the major thematic elements of structure are what I call the form. That's the architecture. There are a lot of things that go with that, too. Of course, an architect has to know their materials, and so there's always engineering aspects to architecture. There are aesthetic aspects to architecture, especially in software.

This is something that's I think again not appreciated by software people is some of the history of, for example, object-oriented programming, which is a lot about the human being interacting with the computer and how they're mental model is captured by the structure of the program inside the computer. So from an object-oriented programmer's point of view, what architecture means is the end user's mental model. How do I create a form of the system that echoes the forms that are latent in the end user's mind?

That takes us into the classic architectural notions of aesthetics. It's a huge, huge, huge work that covers psychology and perception and physics and engineering and just about everything you can think of, but to me the unifying concept here is form.

Joe: I did like how your book talked about the functionality and what the system does. It's one thing to sit there and says user stories, but I thought the book was very well written because I touch upon that field, but I'm not a software developer. But I got it when I read your book. Maybe not as deep as what someone else would understand it, but I certainly understood the writing and I thought it was very well written.

Jim: But I'll bet you found parallels in your experiences in other disciplines.

Joe: Well, that's exactly right. One of the questions I have for you written down here is it seems that we're finding a lot of parallels in business today, in manufacturing, and in Lean with the Agile community. It's like the software developers are leading the methodologies we're picking up behind it, even in some of the decision structures we're using. Can you say why that is or do you know?

Jim: Well, I'm not sure why but first of all, yes. I very strongly agree with you. Second of all, I'm horrified to death that it's

happening. And third, yes, I've been puzzling over the same question for about 20 years. But one of the things I've been working on a lot for 20 or -- maybe even longer -- 30 years is looking at the history of engineering and engineering designing in the computing field. And it's fascinating. In particular, there's this fascinating chapter during the 1980s. It was called the Design Movement. It's like no one ever teaches this stuff in any computer science curricula, and it's really a shame because it really has these deep insights into what design is about. But it was all these people who got together and said, "Gee. You know, we're building homes. We're building highways. We're doing civil engineering. We're building software. We all do this thing called design. Oh, gee! Design must be this thing. Well, let's get together and let's understand what design is, and let's create this notion of method." They played around with this for 10 or so years.

They found out that, because of emergent requirements and some things that physicists have known for a long time about the nature of the way the world works, then in fact what the physicists know generalizes to a lot of other human and scientific phenomena in that you really can't have a notion of method. So everyone abandoned method except for the software engineers, so they're kind of living in this la-la land fooling themselves about how scientific design can be.

Now there's an architect, I think his name was L. Bruce Archer, who made the same observation that you just did but I think he made it probably 25 years ago -- that in fact it's this man-bites-dog thing, where you find software, which is this new snot-nosed kid on the block leading the design thinking in classic fields like architecture and engineering.

Your head just has to spin and say, "Why is this so?" Maybe it's because they have nicer toys or maybe it's because of what

they're smoking. In other words, they think that design is this real thing worth studying, and so they paid more attention to it. I don't know.

Joe: The Agile side and the learning concepts and the shorter iterations and the loops that they're trying to build within to get to the customer -- even to the prototypes that they're trying to build so they can get customer feedback earlier -- and it makes sense, but why is software 10 years ahead of us?

Jim: This is another one of those arguments. We in software are kind of embarrassingly proud to point to our failures and say, "Look what a colossal monsters that we have to tilt at!" I think we sometimes around the bar we'll cry in our beer about how horrific and wondrous an industry we are in. Then someone will come up and say, "Oh, but here's a film of a bridge falling apart!" Or some engineers trying to blow up a building and the whole thing just kind of rolls over five other buildings instead of blowing up and falling on the ground.

So, we're not alone in having massive schedule overruns. People point to the Sydney Opera House, which is a story we know well here in Denmark, of course, because the architect here followed that. I don't think that software is alone in its failures and I don't think that the engineering disciplines are alone in being ahead or in being behind. I'm not convinced that there's any one of these disciplines that have the answer.

Joe: I'm not sure there is either but I do think what we have seen as the knowledge field has transpired and there's more knowledge than has expanded in every field, that we're becoming more knowledge creation in practically every field. I mean, when you think about 75 percent of everything built is any more practically has a smart side to it, an intelligence side to it. We become more of a knowledge field, software's led us into the

knowledge field and that's why we're adapting Agile. I mean, it's kind of like a short take on it?

Jim: Well, except that I don't think that Agile is about knowledge. You hit a keystone here. Knowledge really, you say field. That's an interesting word. I won't argue field but if you had said discipline, I will certainly argue that because we don't have a knowledge discipline. We do have a knowledge market. So, if I can sell services or consulting, that's certainly a growing market. But the last time we tried to make knowledge a discipline in software, was in the pattern discipline. In fact, we borrowed these ideas from an architect named Christopher Alexander who said, "We know how to build houses. A culture learns over thousands of years how to build its houses instead of reinventing things all the time or doing architecture for art's sake, which is what modern architects do. Let's build homes and follow architectural mores that's sort of comfort and quality of life." He was seen as kind of a lunatic.

Well, the software field took big inspiration from him in saying it's worthwhile building a body of literature around what we know -- that is building a culture of knowledge. And to some degree there are a few people who's been able to hold on to that and to drive some of those principals forward. But, again, it more became a market of knowledge rather than a discipline of knowledge. That, again, has something to do with 20th-century life and the way that markets work.

I don't know, I think a lot of it goes back to the 1960's when we valued progress for its own sake. I suspect that we're probably of about the same generation and that we can remember the great accolades and scientific advances back in the 1960s.

Joe: Sure.

Jim: Anything old was not worth valuing, it had to be moot. If you have an analog watch, you were passé, it had to be a digital watch.

Joe: Is Agile cutting edge anymore? Or Scrum, is this old hat and may be being replaced by, let's say Kanban?

Jim: What does cutting edge mean? Do you mean this marketing thing? Are you asking what has the market mindshare? Or are you asking what the advancement of social knowledge is?

Joe: I think Agile has got into the accepted practice, I mean, at the top of the bell. But is it falling off the curve here or being replaced now with the Kanban?

Jim: They're all just words and it could be a long philosophical discussion. I mean, my main model of industry is a pretty cynical model. I'm older than I look. I've been writing software since 1968 or 1969. So, I've gone through several cycles at some level of cycle. At least there are kind of two human generations and many, many technological generations. I guess as King Solomon said, "Vanity, vanity! Alas, all is vanity and the chasing of the wind! There is nothing new under the sun!" The same bad ideas keep coming up on three-year cycles and seven-year cycles. Then they get beat down and then the generation forgets -- they have a very short memory -- and the same bad ideas come up seven years later. I have seen this inside companies again and again.

One of the things I like about Scrum is that Scrum somehow has broken out of that cycle and I think the reason is that Jeff got a few more standard deviations of the mean than most people do. I can tell you something really, really crazy things about the origins of Scrum but probably can't do it here on this podcast. It gets too intricate and would take too long. But the point is we're not in Kansas anymore when you get into Scrum. Part of this again gets back to what we talked about earlier. Scrum is rooted in what we

might call more of the Japanese view of what Americans call Lean.

In this paper by Takeuchi and Nonaka, "New New Product Development Game," very high innovation, very high parallelism, working right on the edge of chaos. If I trace back Scrum and its origins, which is some of the talks that I'm giving this year and architecture conferences and elsewhere.

You can trace this literally back hundreds of years. You can find very strong parallels between some of the current dominant thinking in software architecture and some of the philosophy behind Scrum and some of the philosophy behind Toyota and some of the things in broader Lean. They all go back to the same few principals that have to do with human behavior.

Agile is a relatively young wet-behind-the-ears kind of thing, about 10 years old. It's very software-centric. I don't think people had the breadth of vision of human behavior or society when they've put this together. I know most of these guys. In fact, I was invited to that meeting but was unable to attend for health reasons. Actually they're having a reunion next month. I got invited to that, but I had a business conflict.

Agile is provocative. I think got a lot of social mindshare in a way that causes people to question a lot of the dysfunction of the '80s and '90s. In that sense, it's good. But it really was pretty shallow and certainly in its adaptation I don't think people really got down to thinking at the level of the principals of the manifesto. In terms of Kanban, Kanban is largely a marketing packaging of a slightly misunderstood interpretation of Lean.

Most of the marketing of Kanban is always around Kanban versus Scrum. And from kind of 10,000 feet, what it looks like is that the Kanban people are taking the fact that Scrum is so hard to do

because it's a discipline. Unlike Agile, which most people think is laid back California guitar playing programming.

Scrum is this discipline and people going into it thinking its Agile fail. Then the Kanban people come along and they say, "Well, you really don't have to think! You can just move around tasks at any time and we have this queue that limits work in progress." Then they make a lot of claims about it saying it's based in queuing theory, which I don't know how it can be if you're moving things in the middle of the queue in the middle of doing queuing analysis.

It's a really nice marketing packaging and it may work for simple service processes like system administration or if you're running a firehouse or something like that. But I simply can't see how it can work for building a product. We have had many and an ever-increasing number of clients who found Scrum too hard to do. They went into Kanban and they just said, "This just isn't for us. This is really not helping us in any way." So now they're going back and trying to do Scrum right, or Crystal, or something else.

Joe: What makes Scrum hard to do?

Jim: Because it's a discipline. It's very simple. I mean, it says you cannot work any overtime. So, you know, management cannot come near the end of the release and say, "Well, you need to put in some extra hours here in order to make our commitments." The other thing that makes it hard is that it runs against some of the prevailing values of industry. Industry says, "We believe that we can commit an arbitrary amount of work to a fixed team and an arbitrary schedule. Well, OK, we know we can't, but we'll make it work by adding more people or by adding overtime or by trimming the fat," which means let's cut quality here and there.

Scrum is uncompromising. It says, "Well, what we deliver, we're going to deliver with the promised quality. If we can't deliver it, then we won't deliver. We're going to make it visible. The fact that we make that visible shows that, well, there's a problem in our process. We estimated wrong. We overcommitted and we need to learn to do better next time." People hate this notion of failure.

One of the key aspects of Lean that I think the Western world doesn't understand. In Lean, we keep saying Kaizen, Kaizen, Kaizen. Get better and better and better. Well, you go to the Japanese and they say, "There is no Kaizen without Hansei."

Probably the closest interpretation of the word "Hansei" in English, would be repentance. It's this deep sense of shame and apology and deep regret for not having built a process that allowed you to meet your commitment. When you fail to meet your commitment, the first step in Kaizen is Hansei, and you don't see many American managers going around doing Hansei. You certainly don't see proud nerd software engineers going around doing Hansei.

Scrum is always focused on this Kaizen mind of being able to get better and better and better, and it takes a lot of humility. I think that's what makes it hard. It takes humility, and the humility takes a high degree of trust between individuals. People have to be allowed to fail so they can learn. I haven't been in three companies in the past 10 years that had enough trust to do what the Japanese are doing in Kaizen and Lean.

Joe: Could I relate that specifically to PDCA a little bit, in the Lean terminology there? Scrum is more Lean than, let's say, other types of Agile methods.

Jim: Absolutely. Everyone thinks that Scrum came out of Agile. Now wait a minute, let's stop this for a second, because Scrum

has been around since 1993 and the Agile manifesto was 2001. How did Scrum come out of Agile? It's really the other way around. Or, even better, they both came out of Lean. Lean is a funny word that gets associated with different labels. In the software world, where people think Lean, they'll, usually, think Mary and Tom Poppendieck, who are doing a great job of getting the Lean message out in the software world. But they've got to have their own perspective. They have their own frame from which they see the world. It's a different frame from which Scrum sees the world, and it's a little bit different frame from which Toyota sees the world. But they're all very, very much in the same cauldron and the same genre of harking back to the original Toyota principles of Lean, whereas Agile is really kind of off somewhere else.

Scrum, as I said, comes from this paper by Takeuchi and Nonaka in "Harvard Business Review" called the "New New Product Development Game" where Takeuchi and Nonaka looked at practices at Honda, at Canon, at NEC, and a lot of other contemporary Japanese corporations -- this was about 1984 -- most of whom had learned their techniques by some consultants who'd come over from Toyota and taught them the Lean principles.

That's where Jeff Sutherland got the ideas for Scrum, and that was one of the main influences on Scrum. Some of my research in Bell Labs was another one of the influences on Scrum. In particular, things like stand-up meetings come out of the stuff we did at Bell Laboratories. Then Jeff added incremental development, iterative development, and time boxing. But most of it comes from Lean, absolutely. So if you look at the planning, doing, reflecting, this Kaizen notion, the cycles that we get out of Lean; this is what Scrum is about, absolutely.

Joe: Now your book is not a Scrum book by any means though, is it? Who is your book directed at?

Jim: No. I mean it depends on what glasses you look at it through, right? It can be close but no. You're not going to learn Scrum from picking up the book. But if you're a Scrum practitioner, you're going to feel at home. You're going to feel that these things fit.

Jim: I think the primary audience, thinking people who have looked at Agile and they come away a little bit skeptical because they see this thinking part or the architecture part or the planning part or the domain analysis part missing. They're thinking, "Where does this fit in? How can I take my view of Agile and reconcile it with what I've known works all these years in terms of architecture?" So the main target is the people who already know that architecture has value or suspect that architecture has value. They want to reap some of the benefits of Scrum and of Lean and of Agile, and they're trying to figure out how to do that.

Maybe I'm going to have a few converts along the way where I'll take the people who only believe in emergent architecture, and I'll try to give them a perspective that some upfront planning has value. I don't know. I'm a little less hopeful about those. I'd like to be able to convert some of those people.

But in fact I'm guessing that most of the people who are going to be able to pick up the book and, without struggling too much with their value system take it forward, are those who already have a penchant for thinking and know the value of planning. I need to emphasize here. I want to quote from Kevlin Henney because people confuse two things. It's not about plans. It's about planning.

And Kevlin Henney quotes Dwight Eisenhower who said that, "I find when going into battle that plans are useless. Planning,

however, I find to be indispensable." So Kevlin says, "It's the -ing! It's the -ing! It's planning, not a plan."

So it's this whole notion, this Lean notion, of a cross-functional team, of the entire team, where everybody's doing everything from the beginning that's important to this planning perspective of authentic interconnectivity between people instead of having stovepipes, of having the Toyota notion of this table where everyone works instead of an assembly line. So people who can see that vision and see the value of that vision will be able to fall into the book very naturally.

Joe: I enjoyed the book tremendously. I struggled with the last half of it a little bit because I'm not a software guy.

Jim: The last half is interesting for the nerds because this comes out of the vision of--well, kind of the original vision of--what object orientation was supposed to be which was this Nordic vision from people like Ole-Johan Dahl and Kristen Nygaard. Someone who was hanging around with them at the time was a friend of mine named Trygve Reenskaug, who just turned 80 years old this last year. He's the inventor of something that's very well known in software called "Model-View-Controller," which is how most modern interactive applications are structured. What people don't understand is that the whole purpose of object orientation was to engage the end user and to capture the end user's mental model. Well, Trygve knew that they got it half right back in the 1970s. But there was this other part missing, and he never really had time to chase it down.

Then he retired in 1997 and said, "OK, now it's time to start doing the real work." I hooked up with him a few years after that, and we've been working on the other half of this, which is the real value proposition for software. It's not so much its structure of what the system is but the form of its function, which is what we deliver. That's what gives software value, is what it does.

Now that has form as well. So function has form. In the last part of the book are software techniques to take the market understanding of form or of requirements and get it more directly into the code than the object-oriented people ever allowed. I think this is a major advance in computing thinking, and most of the credit goes to Trygve in terms of the ideals and the models.

Of course, we'll have to see. We can look for the next 10 or 20 years and see how well it's accepted and see what its benefit is. But I have very, very high hopes for those last two chapters, the so-called Data Context and Interaction, or DCI, Architecture.

Joe: Why do you think that Lean is maybe understood differently in Europe than it is in the U. S.?

Jim: I don't know if the gradient between Europe and the U. S. is that great. I have a really hard time judging firsthand because I haven't been in many software clients inside the US in the past 10 years. Most of my software clients have been in Europe. So as Lean has unfolded as a buzzword in software, it's been difficult for me to compare the American and European markets. So I have to do this vicariously through the literature. It's even worse because what I'm doing is I'm comparing what I read about how the auto industry is starting embrace Lean in the US verses what I'm experiencing in software here in Europe and then try to calibrate this with what I talk about with the Japanese.

So I'm not really in a good position to give a blow-by-blow comparison. I could get some wild guesses, but I'm guessing my wild guesses or yours or your listeners would be just as good -- is that Americans are a little bit more into doing and a little less into thinking. I mean there's this expansionist pioneering spirit in America that's been so good for innovation where America has often outstripped Europe.

But it may come at the expense of a discipline that works better in an arena with scarce resources, and I think Europe has slightly scarcer resources than America traditionally has had. Now, with looking at dwindling oil resources over the next century and increasing population and increasing demands on energy, that may start to change even the United States. And it's hard to separate in current, say, public policy what actually is part of a long-term trend and to separate that from current political fads.

But I see some good signs in the United States in that it's starting to take more of this thinking posture rather than just a reactionary posture. On the other hand, the Europeans are starting to learn to innovate as well a lot more, and we're seeing some of the academic strings loosening up and a lot more creativity here. It's just, all in all, becoming a smaller world and the world community where it's going to make less and less sense to talk about the cultural gradients between continents than it has in the past.

Joe: What have you learned that you were like maybe you were a little bit different that you would like to put in your book that you didn't?

Jim: Oh, it's too soon yet. I was just looking this morning, and it looks like if you look at my books they tend to come out on just about a six-year cycle. I think some of the aha's, and the gotchas and the oh, darns really only hit me two or three years after the book comes out. It's only been a few months in the US. And, by the way, after you get a book out, you don't want to think about it for about a year because all you've been doing is thinking about it for six years.

So I haven't been thinking too much about that. But I had dinner with a colleague last night and we were talking about programming languages. Programming language technology in fact right now is going through some exciting changes. Maybe if I

were to make one change, if I had to wind the clock back about four years, I might have chosen a programming language other than C++ as the language I use to give the examples in software. It probably would have been probably some of the .NET framework view of things, the common language runtime from Microsoft and probably using C# instead of C++ as the example language. So it doesn't really change anything at the level of the principles. I really doubt that I'm going to seriously change my mind about any of the principles or foundational issues that are in the book.

One of the other things I realized when I woke up yesterday morning is, in reflecting back on my books, my first book, which is the C++ book will be 20 years old this year. I thought, "My gosh! That's impossible. The thing is still in print and people are still buying it."

The reason isn't because of the C++ insights. It's because of the system design insights, and I tried to write on the basis of timeless principles of design to the best that I can appreciate them. I'm not a person of fashion. I'm not a person of fad. So once I make up my mind about something, I have to have a lot of evidence to change my mind.

So I don't think I'm going to be waking up over the next couple of years and saying, "Oh, darn! I didn't really tailor the book well enough to this fad." The closest it comes is this programming language consideration for maybe reaching a broader audience more effectively, but that's kind of a shallow issue.

Joe: How could someone get a hold of you or how would you like someone to learn more about the book? What's out there for them? It's on Amazon, I assume?

Jim: Oh, yeah. It's on Amazon and Barnes and Noble and Borders and all the great book sites. We have a website for our

company. I have a company together with my wife Gertrud, and it's called Gertrud and Cope. So we have a website. That's probably a good way to reach us. If you Google me, you'll probably find a lot of pointers that will allow you to get to me via email. And I'm on Facebook and LinkedIn. Those are other good places to get hold of me. Yeah, the book you can get in most bookstores here and there. I show up at conferences now and then. On my web page, I have my calendar about what speaking engagements I have coming up. I'm also a certified Scrum trainer for the Scrum Alliance, and so you can contact me through the Scrum Alliance.

I mean not necessarily for engagement. I love talking with people and email. I love people who can challenge me and I can challenge them back. That's how I learn. So I'd encourage people to get hold of me with questions about this podcast or to relate stories about what they're doing or with inquiries about what they're doing.

And I'd be more than happy to interact with them over email and just chat it up and give them all the benefit of whatever I've seen along the way that might be able to help them out. On the other hand, I will ask them some hard questions, and I'm going to ask them to think.

Joe: Well, I think that's the fun part of it, so I'd like to thank you very much, Jim. I appreciated it. This podcast, of course, will be available on the Business901 blog site and the Business901 iTunes store. So thanks again, Jim.

Jim: Well, thank you, too, Joe. I think you're providing a good service in doing these things. I'm really proud to have been chosen to work with you on this, so thanks a lot.

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*