This is Floyd Marinescu and Martin Fowler interviewing Dan Pritchett from EBay. Dan can you introduce yourself and tell us a little bit about what you do?

Sure. I'm Dan Pritchett at EBay. I'm a technical fellow; I have been at EBay for 5 years. I'm currently working in the corporate architecture organization; we are actaully a new organization that was formed at the beginning of 2006 to start addressing the integration issues across the EBay platforms. People think of EBay as EBay .com maybe think of PayPal.com, probably Skype given the publicity around that but the reality is that we have 19 different properties and platforms, that are all unique technology stacks, unique operational concerns and so on. The Corporate Architecture team is responsible to trying to figure out how to bring technologies to bare that allow those organizations to continue to operate independently and respond to their markets while trying to drive down the cost of doing business at EBay.

EBay doesn't use transactions? What's up with that?

So EBay .com doesn't use transactions. I was asked this question previously [concerning] PayPal, because it's moving money in and out people bank accounts. PayPal.com uses transactions. But EBay .com doesn't use transactions and a lot of it is driven by availability concerns for the majority of our use cases, there's critical data and there's non critical data, and out of necessity we have to split our databases. There's just no way to think that we would run on a single database, so it is going to be horizontally spread both within a given functional area and across functional area. So that by necessity when you start talking about transactions drives you to the 2 phase commit. And from the availability point of view, it is not pragmatic to think that we can get the number of databases that are involved in any particular use case enlisted in a 2 phase commit and both be responsive and also maintain site availability. Beyond that, there's just data that we are willing to lose. Now that doesn't mean we don't care about your data, it just means that we care about different types of your data in different ways, as I'm sure you do, right? We're pretty sure that when somebody is selling an item on EBay that they are going to be very unhappy if we told them that the item was listed and the item wasn't listed. Our accounting department would be very unhappy if the item was listed but we failed to capture the fees for the items, so we don't get paid for it. But there are other things that happen there. There's preference data, there's marketing data, these are the kind of things that we feel pretty comfortable when saying "we are going to make a best effort to make sure that we capture that data, if we miss it occasionally, which is less than at least ... we are at least three 9's in all these things, we missed it and that's unfortunate. It's potentially frustrating for the community and we understand and appreciate that, but we are willing to sacrifice that for making sure we got the other data at a much more reliable rate".

How did you find that when you first came into contact with the organization, what was your reaction to that?

Not having transactions, not letting the database do things like foreign key constraints and things that you would expect a database to do was definitely uncomfortable, put me off balance a little bit. But once you begin to understand what it is you are trying to achieve, it does make a lot of sense and the concept of the cap theorem that you have consistency, availability or partitioning, and you get 2 out of 3, is really spot on. The reality is that as you scale, as you move to high transactional volumes, you are forced in that direction, it's not like there's a set of alternatives. What I find curios and interesting is that while I can't say that I have talked to all the very high volume sites, I can say that for the ones that we have interacted with, we found the same thing. They have arrived there independently, so there comes a point where you outgrow the scaling capabilities of normal expected patterns and you have to do things differently and the way you do things differently seams to be pretty consistent, and transaction is one of the first thing to go.

Also an area that I was interested in your talk was the fact that you minimized what the database does.

The current state of that on EBay may be that we have got a little too far in the other direction and we started to back away from that a little bit. It was a reaction to the fact that there was a critical junction from 1999 and 2000 where we actually had a single database and we were using transactions and we were using foreign key constraints and all kinds of cool capabilities that Oracle

offered, and then the database crashed and it crashed a couple of times, and so the reaction was "ok what do we start off loading out, how do we pull stuff out of the code", and that drove us in this direction of minimizing what the database does. So there are certain things like referential integrity that you can't do if you are functionally decomposing data across multiple databases, because you can't have referential integrity span database instances. Some of the other things that we have stayed away from like sorting, joining, we're coming back to now, primary key joins are actually far more efficient to do in the database then they are in the application space space, some of the sorting stuff that we would like to do is reasonable to do. We do have other challenges with things like ordering, like Oracle doesn't provide good pagination, semantics for sort, so when you are sorting data and it is coming back to you and you need to show things 50 at a time and you've got 5000 items in the result set, it's difficult to rely on Oracle to do that.

A lot of that stuff we have also moved into our voyager technology, because we have actually built voyager, one of the optimizations we made in that was to become very efficient at how you return the forth page set of 50 out of the 50000 in the result set. We have also played some tricks there that are not using some of the technologies we have just off loaded on the other platforms. But for the most part we want the database to be an essentially reliable data store and keep as much stuff as possible in logics so that we have more control over it, we can control when that logic rules, how it is implemented and all that kind of stuff as well. That's the same reason why we don't use stored procedures: anything that happens in the database is invisible to developers for all practical purposes.

Did this experience alter some of your views about the role that the database should play in this kind of application?

Absolutely. Even to the extent that I question at times, and we have been looking at this at EBay, whether you even need a database. If at the end of the day you are just storing bits of data and you are not asking for ordering, you are just doing look ups along primary key paths, you are not doing arbitrary search criteria. If I look at some of the journal file technology, why is that not appropriate at times because it gives you transactional guarantees in terms of being committed to a persistent state, and I can certainly rely on intelligent use of direct structures and file naming patterns to get very efficient look ups our primary key path. It definitely brings in the question "do I need an Oracle database in every instance? Do I need a database in every instance?" And these are things we are starting to look at, we certainly looked at on EBay , there are places where if we have this non critical data that we're willing to take some compromise on does that data need to be in Oracle at all. Are those places where we can put that in different database or not in a database at all? It definitely brings those kinds of things into question.

What other things have surprised you over the course of your last few years at EBay ?

Most of the surprises come along the lines of the things like: you have a process, you have a system, it seams to be running along in a nice state and suddenly these discontinuity events are occurring and you say: "ok, now what do we do?". Few of them blindsided us a little bit. One of them was dealing with our parallel development efforts. We suddenly reached a point where we discovered merging was very hard and trying to get code to the point where we can roll out to the site was a problem, so we had to go through some hick ups coming up with this current merge process we have that does require people to work around the clock but at least it works. Another one has taken us by surprise lately was power. This one has come out of nowhere seemingly, but in reality it was always looking under the covers, but what's most interesting about it is that it became a software problem. And I am not sure everybody is convinced of that, but I am convinced of that now, that we would reach a point where we couldn't deliver power to our datacenters because municipalities can't even deliver the power infrastructure or build the power infrastructure to do that, which then drives you into problems of "how do I drive up utilization on boxes to the point that I'm not wasting, I don't have idle processors that are consuming power but not doing work for me?". An interesting architectural deployment and software engineering problem, if you asked me 2 years ago "are going to be thinking about this problem in terms of how to maximize transactions

per watt?" I would not have predicted that it was going to be part of my life now. In terms of experiences at EBay, and it's harder for me to think of it in terms of what were the shocks from a career, its always easier for me to think in terms of what were the problems snuck up on me. It brought me to this idea that I've been trying to articulate and get my head around of "what are all the vectors of scalability?" I like to think of architecture in terms of a spider graph where all the axis on the spider graphs that you really need to be thinking about. And history teaches us that the vector that you've left off the spider graph is the one that in 2 years from now is going to be the one that has collapsed imploded and you are going to be working very hard to figure out how to incorporate it and then make the trade offs against all the other vectors to get things back into an acceptable balance. And that one is hard, guessing which vector is not on your graph is probably the hardest problem that any architect faces.

Connected with that, what are the kinds of things that surprise new people when they join EBay?

I think one of the biggest surprises for new people is how much we bend the rules of architecture and in some ways this was new for me to think about as well, because we have patterns and principles and it's not that we ignore all patterns, we ignore all principles, but you have to apply them intelligently. I forgot who was giving the talk earlier and said "patterns don't mean you have to turn off your brain". But what's probably the most difficult thing for architects and developers to get their head around is how far we will deviate from a pattern, how far we will bend and break the rules, because at the end of the day the only thing that matters ultimately is that we are able to meet the availability numbers at the transaction volumes we are facing. You don't want to be careless about that, you don't want to throw all principles to the wind but you have to achieve that. EBay is an interesting place; I enjoy being there so much because very straight forward problems become extremely hard at massive scales, and our scales are so massive. One of the challenges that I often face with the product organization is that they will see a cool feature on a site that is parallel to our industry but they really want it. And it's like "they are doing this" and that's ok, but they are doing 500.000 page views a day and you want to put it on a page that's doing 300 million page views a day, and that involves 5 database round trips, in order for us to implement that feature so you are talking about adding 1.5 billion SQL executions a day to a database. A host that can do 600 million host a day, I now need 3 mid tier boxes just so that we can have this widget on this page. These are the kind of problems that you face as an architect at EBay . There is this need to be responsive to the business and to do things, but so many things are so hard. Another cool thing is that what a lot of organizations consider a scaling problem, we don't assign architects to because it's such a trivial volume of data that we expect the developers to be able to handle it. The 3 or 4 million per day per page site is just something that we will apply our pattern to. You don't need approvals, just go build it the way it is supposed to be built, we know it will run. There are people spending hours of their day trying to figure out how to get the 4 million page views a day. To us it's a cast aside problem, you are a developer at EBay you'd better know how to do a 4 million page views a day page.

You mentioned a lot of common rules that you had to bend. What are some examples of those rules?

Obviously transactions are a huge one that makes people crazy. Bringing things into the referential integrity and a lot of the database constraints and pulling those into the application tier. We've also done some interesting things in the way that we collapsed the application tier. There are places in our site where logics occur that you wouldn't necessarily want logic to occur but it occurs there because it's the most pragmatic place for it to exist, there's times when there's business logics in XSLT but that's because it has to occur there for the user experience to be correct. There's a variety of examples like that, it's hard for me to come to specific ones. It doesn't have to be ... is that most of my time at EBay has really been spent on the component level architectural considerations, some design patterns in the v3 stack but the reality is that I've done very little actual coding in that stack, I've done coding on other stacks on EBay , but my primary role has been trying to address how we do integration, how we do deployment, what do our data centers look like, and a lot of these kinds of things, and a lot less about what's the Java patterns that we are using. I did some of that early

on but we have mini architects working on that problem and I got assigned of to other challenges. Do you want to talk about what is the architecture on different layers?
From what level? Because the interesting thing is that there's always the onion. You can look at our application stack and this is a very classic core J2EE set of design patterns, without using J2EE so presentation tier: that's done via command, there's the application objects which are the equivalent of session beans, the business object which are the equivalent of entity beans, a data abstraction layer which is our O/R mapping layer, the V3 application stack is very familiar to anybody who understands the patterns. We have slightly different names for the things we have taken some interesting design decisions like the way we communicate back the page description is we built a business oriented XML structure that goes back to the presentation tear which then does a transform on it using XSLT to generate HTML. That's unique, I don't think patterns necessarily call for that, is not necessarily a deviation from the patterns, but it's a model; it's an interesting way to implement the models. But the rest of it is interesting in terms of how we break things down in the application, the functional decomposition, in the code along specific problems domains. We have some standard ones of user/registration management, buying, selling, finding, billing, I don't even remember what they all are, so that allows us to focus, in terms of code management and dependency management the; way we organize the code. The standard J2EE stuff is about "what does this stack look like?". As you start building that scale you have to go to the next level of functional decomposition saying "ok, how do I manage my dependencies between the various layers of the code, so that I don't end up with all these wads of un-maintainable logic, and then you go to the next layer, the deployment decomposition where you start working on "ok, now that I have a search domain but how do I actually deploy that into a set of operational pools, so that I can manage availability, see the dependencies across those things, how do I determine what they are dependent upon, how do I protect them from having cross dependencies that get in the deadly embrace styles of coupling and then the architectures extends out to even within that how do I actually run a search pool out of 3 data centers and make sure our search is supposed to be nearly always available, supposed to be the most available thing on the site, how do you accomplish that? What do those pools look like? What are the hardware configurations?" And one of the great enlightenments that we got to is that hardware fail over isn't completely reliable, so if you are actually looking at the way that we have configured our pools of equipment, and we have load balancers in front of them and those load balancers are paired so that we can rely on hardware fail overs but that doesn't always work so we have 2 different IP addresses into 2 different pair pools of those that we load, we let DNS load balance across so that we protect our so when you say the EBay architecture is each of these layers, and some of these I'm more intimately involved in than others, my sweet spot is between the "what is the interaction between components, what are the deployment models that we are going to use in order to achieve availability, and so on".
What kind of architecture enforcement does EBay apply across the development team and how is that managed …?
There's 1500 in house developers which is considerably more when you facture in the outsource contracting that we do, and you've got 30 architects. You can imagine that the architects are not involved in every single project. There's actually a process that goes on at EBay where at a time features are request there's always a pager written for the feature and there's a review process established that the feature sound logical; if it's in line with architectural patterns from its description and it appears that it's not going to be a large scale feature, then those are given a pass, the ones that have got potential architecture impacts or are going to be large get an architect assigned. And then enforcement is a too strong of a word for what happens at EBay . For the architects that don't get architectural assignments we have the standard patterns. The developers are supposed to pick the domains that their code impacts properly, they are supposed to follow the standard design patterns, and they do. It's more efficient if they do, so they do. The larger ones that involve architects, we do have a lot of standardized patterns and we actually have taken that beyond just the normal coding patterns you might expect. We have a whole collection of standard database patterns, based on the type of data you are trying to model, which of the read/write

model versus the horizontal split model, what will your key management be, all those kind of things. I have got some standard patterns, we do it also for asyncronization standards, and we've got 3 or 4 groupings of these that we do. We use a rules base system; we have one for rules and one for services. That gives the architects some standard frameworks that they can apply which is as in all patterns, it's helpful for them because it reduce their workload, but it gives a lever to use when working with development and operations and that kind of stuff. If I have already got a standard pattern and we pick pattern C for database use, this has already been approved by the architectural review board, there's an architectural review board called ARB, so it's already been approved by them, we don't have to go ask for their approval. Now if you want to do something different, that's going to get through a whole different set of review processes. Standardizing these patterns and having approvals help them considerably with that.

A spin on the previous question; you said you focused more on the deployment side of things. How do you interact with developers?

My previous role at EBay was, when I was part of the EBay.com architecture team, I was senior architect, I tended to interact with developers less because I was working on the meta-architectural issues for the architects. So I drove a lot of the patterns getting them pre-approved and so on, I had less day by day interaction with the developers. The new role that I'm in where we have actually brought the development team together, in that case I talk to the developers daily. I actually am a big proponent and I try very hard even with my responsibilities to write code, I think architects have to write code for 2 reasons: one - they need to stay in touch with what the real problems are that you are trying to solve, if you don't understand a problem it's simple, if you actually spend time understanding the problem, then you gain in appreciation for the complexity of it. So that's one reason, the other reason is that I think that in order to be highly effective with the development team you have to have a certain amount of credibility with them, and if I go in and I actually make contributions to the effort that has being done, I deliver code, they see that I know how to do this, this means I am going to be more likely to propose reasonable solutions, and they are going to be more likely to believe me, because they have seen me do the work and I have credibility as a developer as well as an architect. That's the way I interact with the current team and I actually like that interaction the best, because I believe design is a spectrum and where architecture ends and where design begins is very fuzzy. And there's also a loop back that's involved because there are going to have problems that run headlong into design that are going to have architecture impacts, there are limitations of what you can accomplish with code. You've always got constraints of what can you do with current programming languages, what have you got in terms of the current hardware that will be able to help you and so on. Architecturally you can make some decisions, but when you run to the walls you can try to fix that up by design changes or you can go back and revise if you are on the right architectural path and whether it is feasible this time, then continue to pursue that. I prefer that more continuous feedback loops all the way up and down the process chain. There's a lot of talk about Agile, and Agile work but it tends to be towards designing code, architecture needs to be somewhat Agile as well, and much more so than it has been. You have to bend a little bit, if your architecture is waffling all over the place it will make everybody crazy but you have to be open minded to the fact that you're going to make adjustments in the architecture.

Are you conscious of deployment issues being more important earlier on for the application developers?

Absolutely. I think that one of the things that we have done in our team that has been highly successful was that we own all the way from requirements definitions to deployment, everybody is involved in that as part of the same team, we sit on the same floor, we have joined meetings, when we are working on architecture issues around the things that we want to do, there's a requirements person in the room, there's an architect in the room, there's a developer in the room, there's an operation person in the room. So we get all the concerns out on the table and get all the constraints and be able to focus on them to make it happen. You have to think about building this stuff from the very beginning that it has to be run, so you have to involve the entire stack. And that goes back

to my earlier point of envisioning scalability and I cheat. Scalability to me includes things like manageability, because that's how well you can scale your operational team, I got called on this, but we can do it however. But it you think about it as a spider graph, if you leave somebody out of the room that owns a vector on the spider graph, you are going to get wrong. It really is a fixed sized string if you pull on one of them something else is going to give in order to make it happen. You don't have infinite elasticity of where that thing can go. You have to make the right set of compromise.

What are the interactions styles you defined? What is your services strategy?

There's an outward facing services strategy and there's the inside of EBay services strategy, as I said earlier with 19 distinct platforms, we have a reasonably large integration challenge even if we don't look at what our external community needs in terms of interfaces. It's about trying to define services, trying to find the best way to expose services, make them known through whatever method, if it's WSDL or something more pragmatic like a Wiki but just to make sure that platforms know what other platforms have to offer and how we let them interact. The other area that it's proving to be interesting is that services are interesting but they are asynchronous couplings. It's actually a scary proposition at EBay not only because of the availability considerations, but there's also a matter of scale. If one of our platforms, like Rent, has something that they have to offer on the EBay site, EBay gets a little nervous about the coupling between their availability, but Rent is actually is scared to death about the EBay transaction volume getting pointed at one of their service integrations. We are spending a lot more time trying to figure out what's the appropriate way to do asynchronous integration, what's the right technology stack for doing that, what are the right interaction styles, because we definitely want to be able to decouple volumes and availability and all these kinds of things, between the platforms, let them continue to grow and Agility is important. One of the things that has been very important is not just the things you would expect, in terms of coupling, but also not couple those businesses ability to respond to their markets, because each one of them has got very different market pressures that they are trying to respond to. This became very apparent to me when I was looking at the auction site itself. We have EBay.com which services 33 countries but we have also got Korea which it's own auction platform. We have been very careful with how we help Korea, the architectural "help" idea, but they have a competitor there. EBay is in our space of online auctions, our competitive pressures are different, we are getting a lot of non auction competition, that we have try to figure how to respond to, but that's a different problem. In Korea where they actually have a neck to neck auction competitor, and they are in the lead right now but the slightest stumble, a poor community experience, any of these things, it is close enough that it could flip it and knock them out from being a leader and then they have to try to play catch up, which is not a position you want to be in. It's hard enough to stay in front of your competitor it's even worse to try and catch them. At an architectural level we're talking about interaction styles. But we have to be very cognizant of anything that we do to try and integrate with them and help them does not detract from their ability to be responsive to their business and maintain that position that is a very big challenge for them to hold on to right now. And then back to my earlier point that architecting at EBay is a challenge, yet another piece of this problem that you bring in when you're trying to architect solutions and so on, you have to understand what you are facing you are trying to find your way to the proper set of compromises.

What are your 2 favorite computer books?

That's an interesting question. Actually there's one, and I'm not going to be able to remember the name of this one, but it's a book on Java axioms that I really like because it's a very well written book in the sense that it's broken down a little one page set of axioms that you should follow, and it's a quick read, it can be read quickly but also can be used as a reference when you get into certain situations. The problem I have with answering this question is that so much of my reading of late has moved into the online world, the blog world, trying to track what the opinions are of the people going on and get the pulse of where things are and that may just be the nature of the world now, the interesting dialogs and content that is being generated at such a quick pace that it's difficult to think about going in a computer store and buying a book. I have a lot of blogs that I

follow I also tend to follow tags, a lot of delicious tags, and the delicios tags. I've got 4 or 5 RSS feeds on architecture issues. The bulk of my reading computer industry trends and so on is really coming out of there and looking at what other people are thinking is worth reading.