1:33

Alexey

**This week, we'll talk about building scalable and reliable machine learning systems, or we can call it machine learning system design. We have a special guest today, Arseny. Arseny is a machine learning engineer who has worked on machine learning projects since 2015. He mainly focuses on deep learning and MLOps-related problems. This is not the first time Arseny has appeared on this podcast. He previously gave a talk maybe one year ago about software testing for machine learning pipelines. It was actually quite well-received. So thanks a lot, Arseny, for doing that. It's a great pleasure to have you here for a second time.**

2:09

Arseny

Likewise.

2:11

Arseny

**For those who don't know, Arseny has been working on a new book about machine learning system design for some time now, so we wanted to invite him to talk more about that. Welcome to our event.**

2:25

Arseny

Thanks, Alexey. It's a great pleasure to be here.

2:28

Alexey

**The questions for today's interview were prepared by Johanna. Thanks, Johanna, for your help.**

# **Arseny's background**

2:34

Alexey

**Before we go into our main topic of machine learning system design, Arseny, let's start with your background. Can you tell us about your career journey so far?**

2:45

Arseny

Yeah, sure. I've been doing machine learning for a while. I usually work for startups. I used to work in some big companies as well, but I realized that startups are a more preferable area for me because those are usually places where I can have a reasonable impact with my skills and my personality. For the last year, I've been working for a company named Ntropy. Ntropy has been applying machine learning and language models for financial transactions and extracting some information from it.

Before that, I've been working for Instrumental. Instrumental is a manufacturing optimization company, as I said in our pre-show discussion. Before instrumental, I've been working for an AR company that has been recently acquired by Farfetch. Even before that, I've been doing machine learning for a ride sharing company that was acquired by Lyft. So, many startups. And that's kind of the background that I wanted to express in the book.

4:10

Alexey

**This AR company that you mentioned, I think there is quite an interesting application there. It's about trying to show how shoes would look on you, right?**

4:24

Arseny

Yeah, it was a virtual try-on. We started with a very small niche of nail polish try-on. And later we started working on jewelry, like rings. At some point, we understood that well, rings are probably a very specific niche as well. What was the problem with nail polish? This is a very small market and there are not many companies who are ready to pay.

We were going to have a b2b application, selling SDK to companies who are interested in embedding it to their applications to improve conversion rates and stuff. It didn't really work out. At some point, we found a company that was trying to build a virtual try-on of shoes and their tech was actually far from perfect. We, as a company, decided “Well, we can do better,” and we did better.

5:33

Alexey

**So the idea is – you take your phone, point it to your feet and then see how your feet would look with the shoes?**

5:43

Arseny

Yeah, with various sneakers. If you consider buying some fancy new collection by Nike, but you're not sure if it fits your style or not, that's what you can do there. We had a consumer app, just for a personal trial, and we had an SDK that was integrated by big companies.

# **Working on machine learning in startups**

6:11

Alexey

**As a machine learning engineer who has worked in many different startups, I guess for you it was pretty typical that in all these companies, you don't really have anything for machine learning models, right? You have a data scientist who comes up with the model, but then what to do with this model? How do you serve this model? How do you design the whole thing? How do you actually productionize the models? Right? Was it something that you observed in all these companies?**

6:43

Arseny

Yes, that's right. Sometimes people, when working in big companies, already have some kind of template. They have a relatively small space to explore like, “Well, you just do what people do.” You don't need to make many design decisions. You don't need to choose tech for everything from scratch. When you're in a startup, you have to make all the decisions on your own or with your small team and you have to find some kind of trade-off between building reliable, scalable stuff with all the bells and whistles and from the other perspective, you need to make everything ready ASAP – yesterday. You don't have much time. Both ends are not very healthy. You need to choose something reasonably good between acceptable or unacceptable in regards to time and resources.

# **What is Machine Learning System Design?**

7:54

Alexey

**Is this what you would call machine learning system design? You have a machine learning problem or system and then you need to design it in such a way that it's both reliable, but also ready as soon as possible. So you satisfy what you just mentioned, because in a startup, you need to move fast. You don't want to spend a year building a machine learning platform, right? Or is machine learning system design about something else?**

8:21

Arseny

I would say machine learning system design is about making decisions on how to build a system given some constraints. While saying “given constraints,” they exist, but you don't know them in advance. When you are some kind of junior engineer, you usually receive your tasks in your task tracker, or directly from your lead or manager, and you are being told, “You need to implement this message. You need to write this class. You need to train this model.” Very straightforward.

When you are some middle-level engineer, you have more freedom, and you start making your own decisions, but your area of decisions is still relatively small. But when you are in charge of bigger systems and when working in startups, you become in charge of these systems very early. You don't have a “Senior Principal Data Architect” who will take care of everything. You just have this problem, you have a founder who says “Well, we need to build it. We have some ideas. Blah, blah, blah, but you need to build it.” This means that you have to make many decisions and you need to explore the world while trying to find all these constraints, or at least the most important constraints, and make your decisions regarding what you should do, how much effort should be dedicated to this or that stage given these constraints and requirements.

When I say “requirements,” I don't mean something very formal, because in huge enterprises, when we talk about requirements, it's a big document written by a business analyst or system analyst or product manager or whatever. There's a chance that you work at a company where there is no product manager but requirements still exist and constraints still exist, you just need to catch them out of the air.

# **Constraints and requirements**

10:34

Alexey

**I'm just wondering, what are these constraints and requirements? Do you have an example? You mentioned that you don't necessarily know them in advance. I imagine that if you work in a startup, then maybe there are some requirements coming from your product manager or CEO or whoever – things like “We need to ship this thing in two weeks.” Right? That's kind of one of the requirements, I assume. But what are the other constraints and requirements that you don't necessarily know in advance?**

11:04

Arseny

Okay. Some of them are purely technical and some are product-related. When we are talking about purely technical stuff, you can expect that your stuff is gonna be run on specific hardware. For example, we were talking about this AR company and I think I can bring many examples from this one because I was an early engineer there and many decisions were made with me and by peers at this early stage.

So, for example, we decided that it is a mobile application, and for a mobile application, to work in real time on a video stream, we definitely cannot afford to have some heavy machine learning models and we cannot afford calculating in the cloud. So it should be on edge devices. This is a significant constraint. You cannot run something like a one-gigabyte convolutional neural network on this device. But from the other perspective, we decided that we start with iOS devices. Apple infrastructure is ML-friendly and it was ML-friendly back in the days when we started this project. There was Core ML and Core ML is fairly optimized for this hardware. We were working with iPhones 10, I think – iPhone X – and it was good. Porting our software later for Android was a huge pain, but for the beginning, to build some kind of demo app, iPhone is good enough. So, given the hardware, you can understand what you can do like, “Okay, networks that are too big are a no-no, but deep learning in general is fine, because this ecosystem is deep learning-friendly.”

From the other perspective, there are some requirements about how we expect users to perceive this product. At the very beginning, we realized that if this rendered shoe that we draw on a screen is shaking, because of some instability of our model, that's not good. It annoys people and users are gonna say that that's not acceptable at all. We also realized that if our models are heavy and consume all the GPU, it also means it consumes a lot of energy (a lot of battery) and users will definitely not be happy about sucking all their energy in a minute. So that's another constraint.

This means even if the phone can run this neural network with certain FPS, it doesn't mean you should do it. Maybe you should do something like run it at 10 FPS and interpolate it with a simpler algorithm in-between frames just to save some energy and thus, the phone will not get hot. That's not a very straightforward requirement but when you start working with mobile inference, you understand it relatively quickly.

# **Known unknowns vs unknown unknowns (Design stage)**

14:49

Alexey

**I wonder, for all these things such as not being able to afford to send data over the network because you need have this in real-time, you need to know about these things in advance before you even start training your model – before you start collecting your data – because otherwise, if you don't know this, then you end up having a heavy model that doesn't really work on edge. Right? It always has to be in this “design phase” maybe, of a machine learning project, where you don't do anything except think about the problem, right? Or how is this done in practice, usually?**

15:32

Arseny

Well, as people say, “There are known unknowns and unknown unknowns.”

15:37

Alexey

**Right.**

15:38

Arseny

You can maybe try to mitigate the first one – you talk to people who have been doing something similar. We had buddies who have been doing very complicated deep tech, ML-based projects for the mobile platform and we talked with them a lot. They warned us about many things, both high-level and low-level things. High level meaning, as I mentioned, things like this FPS stuff and that users care about stability of the rendered object, and low level is things like “When designing your model, be careful about this particular layer, because this particular layer is not optimized for this architecture or this activation. This function is not optimized as well. But if you do this one and this one, you can merge them together and it is run by a single pass on a GPU, so it is very effective.”

These are tricks that you can learn in advance if you have people to discuss it. You can also do some early tests with a very dirty baseline, and so on and so forth. So it makes sense to make something with a simple baseline and try this simple baseline on your own, or maybe within your team, within some of your friends, in order to understand how it can be used. This reveals more and more hidden knowledge – more and more “unknown unknowns” become things that are known unknowns. Well, you're not particularly sure what's going on here, but you reveal some parts of it.

With this application, for example, we released one of our earliest versions and we have found a problem related to reflecting surfaces. We expected that users would have just their legs, but sometimes they took a photo or shot a video near the mirror and we had four legs instead of two. We did not expect this at all and the algorithm was not supporting multiple hits and there was sometimes some crazy show of a shoe jumping from the real leg to the reflection of it. It was hard to catch.

18:29

Alexey

**[chuckles] That's funny. So in this case, you said there are known unknowns and you can learn about them by talking to people with experience or doing some simple baseline and seeing how users react. But then there are unknown unknowns, where the only way to find them out is just to roll this thing out and see what happens. Right?**

18:49

Arseny

Yes. Actually, writing a design document or something is a way to transform some unknown unknowns to known unknowns. Well, you cannot predict everything, especially when you are doing something pretty new, which is a common thing for a startup. But when you try to write down what you're going to do, it often reveals what you don't know about the thing you're going to build. It reveals some lack between components, lack of understanding of the domain, and so on and so forth. So even for startups that often work in a manner “Ah, screw it. We should just build it!”

At some point, I realized, “Well, it makes sense to spend a bit of time planning. Not too much, but at least something. That's not because you have to follow this plan exactly line by line – as people say, “The plan is nothing, but planning is everything.” That's exactly what I like about the design phase. Okay, maybe you will not follow this design document exactly, but it will reveal many things, and understanding how you're going to build a thing is more important than actually following this exact paragraph.

# **Writing a design document**

20:21

Alexey

**I guess it's because it kind of forces you to learn more about the problem. It forces you to think about different situations. Otherwise, you wouldn't just think about them and just blindly think, “Let's just do it and see what happens.” Speaking about this design document, what does it look like? What kind of sections are there? Is there any structure?**

**Let's suppose I'm a machine learning engineer. I'm joining a startup and they don't have anything there before me, and I want to make things correctly. There is a new project and I want to make sure I follow the best practices. So what do I do? How do I approach this design phase? How do I structure this design document?**

21:07

Arseny

Right. It's a tricky question because in big companies there is usually something like a template, but for startups, there definitely is no such thing as a template. In this case, I prefer to say that having a template doesn't really matter. You just start with whatever. You take just a sheet of paper and start writing stuff – or you start you open a Google Doc and you start writing stuff. One heuristic that I have in my mind is that inexperienced people tend to start with the solution, but you should start with a problem.

If you would like to spend something like four hours writing a design document because you don't have more time and you would like to make a two-page document or even a one-page document, because your system is very simple – dedicate 50% to the problem and the rest for the solution. The solution is usually more straightforward when you know the problem, but it is a very easy mistake to ignore the real problem, which is complicated, and to start building a very good solution that does not address the problem itself.

# **Technical problems vs product-oriented problems**

22:48

Alexey

**Is there any framework that can help me approach... Let's take the first part. You said to spend 50% of the time (two hours) on defining the problem and the rest of the time (the other two hours) on describing the solution. So what can help us describe the problem? Because to me, it seems like it's kind of straightforward. You just go there, and spend 10 minutes describing what exactly this product that people want from us is, with a request like, “Okay, they want to build a system for automating this,” or “They want to build a system for trying sneakers on.” That's the whole problem, so why do I need to spend two hours on that? What kind of questions do I need to ask myself to help me come up with a good problem description?**

23:34

Arseny

Well, as we discussed earlier, there are some that are purely technical, and there are things that are more product-oriented. And you can split the two. So definitely some things are about appearance here and you can talk to product people like, “What is the user scenario? What do users care about and what do they not care about?” We are talking about hypothetical users, so it's all not set in stone. You should take everything here with a huge grain of salt, but still. And you can get some further insight.

For example, in this AR problem, there is very often a trade-off: Do you want it to look realistic, or would you like it to look just “appealing”? And they're not the same. If you're going to build a virtual try-on (which is a try-on with the goal to help the user understand if this particular item fits) you should prefer a realistic system. But if you just want to make an entertainment application, (these videos will be posted to TikTok and Snapchat) then you definitely don't care about them being realistic and you want them to be fun. This is a trade-off, you don't know about, usually, and it takes some time to reveal this problem – some conversations and so on and so forth.

You ask questions, you make assumptions, you verify these assumptions, and given this pure product-related thing, you should make some connection to pure technical things. For example, they say, “We would like to have it in real-time.” What does “real-time” mean? Because, well, sometimes that is felt like real time for a user, as I said, it could be “Eyes are like 30 FPS in reality” or it could be like “10 FPS with interpolation” or “20 FPS with interpolation,” and “What is the devices that we are going to run? How many models of shoes can be tried on?” Definitely, for any complicated problem, you can always spend a couple of hours trying to understand what you need to build – what is good and what is not good.

I don't know of any specific good framework. As part of this book, we are trying to build one. It always matters what the problem is, and I don't think we can build something that is holistic, which covers everything, but we try to cover many situations and describe what you can do in different scenarios. One more heuristic that is included in our book, that I would like to mention here, given the question is that in non goals is a good technique as well. It's always tempting to say, “Well, we need to do everything perfectly.” But you cannot do everything perfectly. And you cannot say “This is important and that is important and also this one is important.” If you'd like to have a model that is fast and accurate, and it's also easy to build during two weeks – that's not very realistic.

You can choose what you don't care about and you can choose what to sacrifice, because most decisions are actually trade-offs. You need to understand what the surface of this landscape is – what are the spectrums of where you need to put this single point on this trade-off and say, “Okay, if we are going real-time, we care about performance more than just the time to development,” because you definitely you cannot build real-time too fast. And if we are going to be realistic, we should care about exact matching of this photo to another photo and later it will be reflected in our metrics and losses. But we do not really care about these images being just aesthetically appealing.

29:01

Alexey

**So, in summary, in this design document for the non-technical part (for the problem description part) we should include goals and non-goals and we should include all the assumptions we have. I guess assumptions could be how exactly users will use it or other things.**

**Then there are technical limitations and also the user experience – what part of the flow will it be used? For example, the user will take a phone, and point towards their legs and then what should happen exactly. This is all part of the problem description. Am I right?**

29:46

Arseny

Yes. I would also add that there should be some points that later will be transformed to metrics. How do we decide what's good and what's not? When you draft the solution part, you definitely would like to have some metrics regarding what is a good solution and what is not. When you have no objectives that you can measure, or at least imagine how you can measure it. You may not have an exact idea of what you will measure because, for example, the aesthetic appeal is not a thing that is easy to measure. But when you just write down “Well, we would like each to be aesthetically appealing.” That's a problem. And in the solution part, you can think about how you're going to measure.

30:37

Alexey

**It could be simple things like, “We add a feedback button to the screen. And if something goes wrong, users can give us feedback and then we see how much negative feedback we get.” Right? That could be one of the metrics?**

30:51

Arseny

Yeah, definitely. We're talking about the very, very earliest stage, where you measure this aesthetic property of your pipeline on your own. You're just looking at it and saying, “No, I don't like it.” That's my metric. Later, you introduce some human feedback. You introduce a platform for A/B testing. Instead of just you looking at these images, you ask your peers to look at them. Later you ask some external evaluators to look at them. And later than that, you receive some feedback from end users of your platform. But it's still a cascade of very similar things.

# **The solution part of the Design Document**

31:42

Alexey

**Okay. So for the design document that we talked about, there are two parts – the problem description part and the solution description part. For the problem description part, we more or less already talked about this. Here, I guess, we need to sit down with product managers and try to get as much information from them as possible. We also, perhaps, need to spend some time with our engineers – mobile engineers, for example, in this case, or some other engineers who can help us identify all these technical limitations. We write down all the assumptions and come up with one page of the document. Then there is the second page (or the second half) of the document, which is the solution part. What do we write there? How do we approach coming up with the content for that part?**

32:37

Arseny

Okay. One thing you start with are very typical primitives that are related to machine learning. For example, “What is your baseline? What are your metrics?” Those are two things that we already covered. You cannot really build something without having a solid baseline, because otherwise you don't know what to compare to. Once you have that, you can draft what your next solution is – your minimum solution that is better than the baseline and one that you're actually going to build. It's very likely that it will not be just a single model. It may be some kind of pipeline of things – interconnected blocks, components and so on, so forth.

Which of these components are your responsibility and which components are you expecting from some other people? How are they interconnected? What is their KPI? I do not say you should directly describe things like, “Your micro service is going to query that micro service with this request payload.” No. It can be written later. But that's not the goal of the design document. You need to understand, “Well, do we send data to these guys in batch or do we do it per request in real time? Do we have some kind of queue for this data or not?” So it's a very high level interaction with external components.

You definitely should mention some data-related aspects of the problem. For example, do you already have it? If you have data, you're lucky, of course. If you don't, how are you going to get this data? Getting the data itself can be the biggest problem and it can be like 90% of the solution. But if you're lucky and everything is in place, then how are you going to process the data? Is it ready to go? Does it need some preparation, filtering, some feature engineering (if you're working with classical machine learning)? Sometimes in deep learning, you can have feature engineering as well. Does it need some specific transformation? For example, “Well, we have this data, but it lies somewhere in a data lake that is barely accessible and we have many problems related to fetching this data, both technical and organizational. So we need to address them as well.

Also, how are you going to make this whole system work? “Okay, we have some baseline and we have some kind of solution. We will train model X and we will interconnect with database y and it will be connected via some protocol. We are going to evaluate it with some internal metrics and some external A/B (or something) and this is our criteria of success. If things do not go like we expected, we are going to look here, here, and here (our analysis stuff).” Again, in startups, it's very rare that you have everything in place in your first round of the design document. It happens sometimes, but that's not about a two page document that we were discussing in the very beginning.

If you have that many sections, it means the document is already like 20 pages. Plus, it usually evolves. A design document doesn't have to be a static thing that's set in stone. You start with something small, and later, you add missing parts, you collaborate with your peers, you discuss stuff, you reveal new pieces of this hidden information of unknown unknowns, and you fill the gaps.

37:15

Alexey

**So from what I understood, as a short summary, It should be as follows. First, we start with describing the baseline – what is already there or what we can build super quickly to have a number to relate to. We want to build something and we want to understand if we actually manage to improve something or build something useful. For that, we always need to have a metric and a baseline we can compare to. That's the first step. Then, you start creating a diagram. In this diagram, you outline all the things and how they are connected and how the data flows.**

**For example, there is a mobile device. From this mobile device, the data (these pictures) come to this big box that we call a model, the model does something, and then sends back something else, such as a frame with a shoe on it. Then there are some other things like extrapolation, as you mentioned. For example, maybe you don't put the shoe in every frame, but in one third of them, to the device's battery.**

**There could be some components, where one is actually putting the shoe, another component is doing extrapolation, and then maybe there is a third component that is doing something else. You put all these things in a diagram, using a Google document, or Mira (or whatever diagramming tool) and then this is where you see what kind of data you need. What kind of data flows in through your system? Also, what kind of data do you need for training the model? But maybe that's a separate diagram that describes the model. [Arseny agrees]**

**When you do this, you start to see all the dependencies. You see what your dependencies are, you see what the other dependencies are, and you see what kind of data you need. Based on that you can understand, “Do we have data? Is this data easy to access?” And then, I guess, this is a continuation – you can have a section in the solution that describes all the data issues and all the challenges that might arise. That's more or less the solution, right? Did I miss anything else?**

39:21

Arseny

Nothing significant. There are always details, and I'm pretty sure I missed some details when answering your original question. But that's why we are writing this book – to try to catch as many details as we know. [chuckles]

# **What motivated Arseny to write a book on ML System Design**

39:42

Alexey

**Yeah, so about this book – what actually motivated you to write it? What kind of problems did you see in the industry that you thought of and understood, “Okay. There is no good description of this. I need to write a book about that.” How did it happen to you? What motivated you?**

40:02

Arseny

Actually, I think there are many people in the industry who are smarter than myself – better engineers than myself – but who do not always see the forest behind the trees. You can find a person who writes better code, with fewer bugs, twice as fast compared to me, knows all the algorithms in the world, and reads fresh papers every day – a perfect candidate. But when you delegate a big system to this person, they fail for some reason. And that sucks. That's unfair. This person should build something twice better than myself – 10 times better than myself. But sometimes they do not. We would like to convert people who know a lot of stuff, but they don't have this holistic overview into people who can build nice, reliable systems.

41:03

Alexey

**So that was your main motivation? You saw that this thing is happening in practice – there are people who are very smart but don't maybe have the framework or knowledge of all the things we just discussed. For example, if I work as a data scientist or an ML engineer – as an engineer, I'm more solution-driven. Right?**

**For me it's like “Okay, there is this problem. I'll spend five minutes talking to a product manager. It seems that I understood it. It's simple. I just need to put shoes on the frame and that's it. So let's start coding.” And then two weeks later, one month later, or I don't know, half a year later, it turns out that I misunderstood some parts and that I built something completely useless.**

41:45

Arseny

It happens, it happened to me as well in the past. Again, we describe it in the book. It's very often that experienced people are available for organizations exactly because of their bad experience. They did some bad work a while ago and they will not do it again. They don't want to repeat their own mistakes. Generalizing this experience is what makes these people valuable, like, “Okay, I know, it doesn't work for me and it didn't work for my friend in some other organization.” Those people can make better decisions when building complicated systems. That's generalized, experienced – crystallized.

Actually, while writing this book, I had to generalize a lot of my experience as well. Because. Let's say that I'm writing a chapter about datasets and I definitely understand that there is always the question, “How much data do we need for this particular problem?” Usually I say, “I have some intuition about how much data I should have for this or that problem, but I don't know how it works in my head.” I need to try to generalize it, I need to try to give some wording that can be transmitted from me to another person. At this point, I realize that I don't really know.

There are some heuristics, but they're not universal. Very often I don't have significant grounds to say why we need X data for this problem. It's painful, to be honest. But it was valuable for me, and I hope it will be valuable for the readers, because I had to seek a lot about my own experience and ask everybody around me like, “Okay, when you have this problem, how do you approach it? And I realized that “Well, I had one perspective, another perspective, and I have seen some additional one, but maybe it's not everything. It still doesn't fit the full picture. I'm definitely missing something.

So I went and read things on the internet, I annoyed other experienced people, and I realized that, “Holy shit! I missed a lot. That's definitely just my very, very narrow perspective and people do it in a completely different way. I need to learn from them, discuss things with them, and later that's just like two sentences of crystallized experience.” That hurts but I believe it is useful.

# **Examples of a Design Document in the book**

44:54

Alexey

**I was going to ask you if you know any good examples of a design document, but then I opened the table of contents of your book and I saw that chapter four is called Design Document. Can I assume that there is an example in this chapter?**

45:10

Arseny

At some point, we decided to do two examples of two systems – both imaginary systems, non-existing ones – and we split them chapter-by-chapter. Our book is structured in the following way: we have 16 chapters, and the first 4 chapters are introductory (more about the problem statement and how important it is to understand the problem, and how to do preliminary research on the problem). Starting from chapter 4, we effectively introduced two systems. One is about dynamic pricing for an imaginary retail company Super Mega Retail. And another one is about an imaginary company named Photostock that needs to build a search engine for their photos.

For every chapter after that, we write a small piece of this design doc, related to this particular chapter. For example, we have a chapter on Datasets and we write a small piece (like half a page or a page) on what we should do in order to gather relevant data to build a search engine for Photostock Incorporated. The next chapter is Metrics and Losses. Here, we write out how we would approach the problem of building the metrics and losses for the search engine in the imaginary company Photostock Incorporated. And we do that for every chapter until the very end.

# **The types of readers for ML System Design**

47:09

Alexey

**In the end, I guess the main outcome for me, as a reader, will be that I will learn how to build and how to write these design docs. Right?**

47:21

Arseny

Again, that's a tricky question. From what we have seen among our earlier readers, there are two different patterns – two different modes of readers. Some people say, “Ah, screw these examples! They are boring and not relevant. I don't really care. I don't want to read it. I don't like it. But the rest of the material is nice. I enjoyed it. It is interesting. It's going to bring a lot of value to me.”

And other people say, “Well, this is boring – all this theoretical stuff and cool stories from your experience are not at all relevant. It does not help at all. But this design document for Super Mega Retail is nice. It brings a lot for me.” So I cannot say it is universal, but for some people, it works.

48:23

Alexey

**I guess it's hard to satisfy everyone. Right?**

48:25

Arseny

Oh, yeah, indeed.

# **Working with the co-author**

48:27

Alexey

**Maybe this is another challenge for you because you're writing with a co-author since you might not necessarily agree on everything, right? Maybe you also have a similar situation where you need to decide what to include and what not to include. Or do you always agree on everything?**

48:45

Arseny

Oh, definitely not. But our experience is very different. My co-author is a person with a very successful corporate career. His last position was VP of Data Science, so he's a big guy. I'm not a big guy at all.

49:06

Alexey

**In all senses. [chuckles]**

49:07

Arseny

I prefer, I prefer grunt stuff – hands-on. In this, our perspectives are very different. There are things I don't really know and here, I definitely don't want to lead. I don't want to be a driver for this chapter. Maybe I will put in my two cents here and there, but that's definitely not something where I'm an expert. For this, “It's all yours, dude.” At the same time, there are some subsets where I feel like I am an expert. I have written many training pipelines in my life and it makes sense that the chapter about training pipelines is written mostly by myself.

That's how we find the balance between our different perspectives. With these design doc examples, again, we split between us – I am writing about this Photostock example and Valerii is writing about the Super Mega Retail, because he has some experience in retail, while I decided to do something closer to my experience. I've never built a search engine, but I think I understand how it could be built even with some other experience in small companies that are related to computer vision and natural language.

50:43

Alexey

**So for you, it's also a kind of situation where you need to deal with a problem that you don't necessarily know, with all these unknown unknowns and known unknowns, but you approach it with this as an engineer who needs to solve this problem and then you show how exactly to build this design document. Right?**

51:03

Arseny

Yes, that's right. I think, first of all, it is not really possible to write about stuff that I built in my previous experience, because of NDA and stuff. And from the other perspective, I would like to write a document about how I would have written it on my own without a deep understanding of the situation. Because, well, that's what I would expect from my readers as well.

# **Reacting to constraints and feedback when writing a book**

51:39

Alexey

**I am a book author myself, and for me, the most difficult part was to decide what I should include in the book and what I should exclude from the book – what I should definitely *not* include. Also, the challenge was that this thing was changing as I was writing the book.**

**At the beginning, I thought I should definitely include this, but as I was writing, I later realized, “Okay, I actually need to split this chapter. Then I need to remove this chapter because nobody actually cares about that.” So how did this process go for you? How did you understand exactly what you wanted to put in the table of contents? How did you come up with this outline? What did you use to make these decisions?**

52:20

Arseny

Many things have changed. First, we wanted to cover everything – everything that's related. At some point, we realized that if we include everything that we planned, first of all, the book will be twice bigger than expected by the publisher and it will take four times longer than expected by the publisher. So that's constraints, again. Actually, you can treat writing the book as a project of its own, with all the stuff. The outline of the book is kind of the “designed doc” for the book.

We got some regularization (some constraints) driven by its publisher and while writing the text, we had some early feedback – some feedback from our friends, as we sometimes shared this or that chapter with a domain expert. If there's an expert in some particular area, that would be definitely useful to share the very raw draft with them to get some high-level feedback.

We also had some feedback from early reviewers. It is informative as well. Sometimes it's kind of controversial. For example, you can have one paragraph with two comments. One person said, “Wow, that's nice. It's mind-blowing. Very interesting. Very valuable.” And another “What the hell is this? That's definitely not interesting. Just a piece of shit. Why did you even write it? And...

54:04

Alexey

**Was that a quote?**

54:10

Arseny

Almost. [Alexey chuckles] We had one reviewer who shitted all over the book. For chapter number 3, they wrote something like, “Well, after these three chapters, I tried to do my best to read through this 'writing' but I cannot do it anymore. It's too boring and useless.”

54:32

Alexey

**I've seen you just made air quotes around “writing”. For those who just hear this as an audio-only version, “writing” was in air quotes. [chuckles] That must be tough. How do you deal with this sort of feedback? It's hard not to take this personally, right? Do you start questioning yourself or not really?**

54:58

Arseny

Actually, I started to. I'm not a very confident person, especially when we are talking about this kind of work. But after some discussions with my co-author and with our publisher, I realized that it is actually good. If we have something like more than 10 early reviewers for the stage and there is one person who is not happy and the rest of the people are happy and most of the criticism is not related to our writing, but more related to how stuff is formatted or how illustrations are made – well, this is good feedback. I am not the dollar bill to be loved by everyone. Yeah.

# **Arseny's favorite chapter of the book**

55:48

Alexey

**Nice. [chuckles] That's a good way of approaching any criticism, I guess. Just think about other things. We still have time – about four minutes. In these four minutes, I wanted to ask you to pick any chapter that you liked working on most, and tell us about this chapter.**

**Secondly, so we're mindful of time, maybe you can give us some extra recommendations, apart from your book, that people can read or listen to or watch, in order to improve their system design knowledge. Let's start with the first one. Tell us about the chapter you liked working on.**

56:31

Arseny

I think, so far, I enjoyed writing chapter number 3 about Preliminary Research. That's not something that is widely covered by existing literature, I believe, because most textbooks that are related to machine learning are about how to build models, how to run models, how to embed them into your inference pipeline, whatever. There are some materials about how to frame the initial problem, how to scope the project – all this project and product manager stuff – but not many materials on how to start working with it. Where should you find your inspiration?

You definitely don't want to build everything from complete scratch. You need to understand what you can reuse, what you will build from scratch, what you can buy as an existing solution (maybe SaaS), what you can buy as a dataset, where you can see how people approach this problem, etc. This was a chapter where I had to think a lot, because, well, I definitely have been doing this stuff for previous projects in my career, but I never generalized them.

A while ago, in this podcast, I've been whining about how hard it is to generalize things, and I think this chapter was the most complicated – trying to understand how to do this. Well, I know what I would do for any specific project. How would I approach this project? I don't know. So it's kind of meta.

# **Other resources where you can learn about ML System Design**

58:28

Alexey

**Yeah. And the other thing – can you give us any recommendations or any resource that could also be useful if you want to learn more about machine learning system design? Do you know of anything?**

58:43

Arseny

Well, there are some things that are transferable from regular software – system design to machine learning system design. It's less about technical stuff, but more about the mindset – how you should frame the problem and how you should approach the problem. I totally love the guide by the Interviewing,io guys. I think I can post the link later.

59:11

Alexey

**Yeah, please send it.**

59:13

Arseny

Yeah. This guide is about regular system design but, first of all, you cannot build machine learning system design if you are not able to build, at some point, a software system. They are interconnected. In our book, we focus more on the machine learning aspect of things, but a good machine learning engineer is definitely a good software engineer as well. There's at least some surface-level skill in building a regular software system that is essential. We do not focus on it in the book, because we are definitely not experts there. We have some knowledge, but it's not the level of knowledge where we can teach other people. That's why I would recommend this guide.

# **Twitter Giveaway**

60:00

Alexey

**Thank you. Before we wrap up, I want to mention a few things. We did run a giveaway on Twitter. For those who took part, I will announce the winners right now, after we finish this conversation – I will click the Tweet button. There are three winners who will get a free copy of the book. Also, for those who are interested in the book, I put the link in the description – Manning was kind enough to give us (DataTalks.Club podcast) a discount of 35% off. There is a coupon code there for the discount. So if you're interested, check it out.**

60:37

Alexey

**With that, I think that's all we have time for today. Arseny, thanks a lot for joining us today, for sharing your knowledge, for talking about the book. Thanks, everyone, for joining us today, for listening in. I guess that's all for today.**

60:54

Arseny

Thanks! It was a really fun podcast.

60:58

Alexey

**[chuckles] I'm happy to hear that.**