2:06

Alexey

**This week, we'll talk about why machine learning design is broken. It's a very provocative topic and we have a very special guest today to talk about that, Valerii. It's not the first time we have Valerii on this podcast. Some time ago, I already had the pleasure of speaking with Valerii and it was also about ML systems. I think it was one or one and a half years ago – some time ago.**

2:30

Valerii

I think it was a year ago, something like that. Maybe you're right. Time flies.

2:37

Alexey

**Time flies. For those who don't know anything about Valerii yet, he's the VP of Data Science at Blockchain, where he is responsible for leading the company's data-driven initiatives. Before joining Blockchain, he worked at different leading tech companies like Facebook, Alibaba, and X5 Retail Group. Welcome back to our podcast.**

3:05

Valerii

Thank you very much, Alex. By the way, I am actually joining a new company pretty soon. I haven't yet disclosed the name of the company, so it remains a mystery. But it will be a significant shift in the industry. Maybe it will be a reason for you to invite me for a third time, right? Who knows?

## Valerii’s background

3:27

Alexey

**Of course. Of course. Yeah, have fun at your new place. Before we go into our main topic of ML system design and why it's broken, let's start with your background. For those who did not hear the previous interview, maybe you can briefly walk us through the career journey you had so far.**

3:49

Valerii

All right. Well, it's not an easy question, because it depends on how deep we want to dig.

3:56

Alexey

**Not too deep because we want to talk about ML design, right? [chuckles]**

4:00

Valerii

Yeah, that makes sense. Okay, let's focus on why I think I am the person who can talk about machine learning system design. As you already mentioned, I worked in many different companies – most of them are quite prominent: Alibaba, Facebook (or the company formerly known as Facebook, known as Meta at the moment) Blockchain.com, X5 Retail Group. I was leading more or less-ish 10-, 20-, 40-people teams and big-ish like 100-, 200-people teams in different countries, in different industries. But my area was almost always the same – it was data analytics and machine learning. It was also data and software engineering and computer science, but I'd say that machine learning probably is my fauter. And I've seen many different projects succeed and failed for various reasons. And that's why a friend of mine asked me...

We decided to write a book, which you already can buy, by the way – it's called Machine Learning System Design, with a ton of examples. And to be honest, I didn't know the name of the book by the very end, because the publisher was pushing to change the title – the original title was Principles of Machine Learning System Design. So I was writing the book with a title that I didn't know. This book, basically, is a Meta document. Well, the person who worked at Meta, which book could this person write? Of course, a Meta book – a Meta document. This document describes and tells you how to tackle the problem of machine learning system design from, let's say, a manual-like instruction point of view. So what are the chapters? What are the milestones? What are the cornerstones? What do you have to cover? What do you have to think about? How to create a single, coherent, and easy-to-understand document that will help you to succeed? To be honest, I was thinking, “Why do we do machine learning design documents?” And the provocative part – and you probably already noticed that I like provocative titles – would be that the goal of the machine learning system design document is for your project to fail.

7:05

Alexey

**What?**

07:06

Valerii

To fail as soon as possible – because it's much better for your project to fail if you invested two or three weeks than if you invested six or nine months. You probably do want to know before than after, right? If you think about that. Let's say doing the design doc either increases the chances of your project to succeed, or you know that it will fail six months prior to that – that makes sense, right? If you know something six months prior to that, that makes sense. You asked me for an introduction and I already started to go in the direction of design documents.

## The goal of the design document

7:44

Alexey

**[laughs] I'm already confused. [chuckles] Maybe we should take a step back. So you already said that you're writing a book and the book is about ML system design. The goal of the book, and then in general, when we create a machine learning system before we actually start actively working on this – implementing, getting data, and all that – we need to create an easy-to-understand document (a meta-document) about the system we want to create. Right? An easy-to-understand document that is coherent, that is not too big, maybe. Then you said that the goal of this document is to make the project fail as soon as possible. This is where I got confused. [chuckles] Okay, so we have this document. What's next?**

8:39

Valerii

Okay, okay. Let's clear that up. When you do a machine learning project, there is a lot of uncertainty. Well, that obviously depends on what do you, but if you think about that – software projects are not... Well, they're relatively deterministic. For example, you can estimate how long it would take to complete something, who you need for the team, etc. However, even for software engineering projects, nine out of ten times, you could find out that you actually spent 100% more time, and you needed twice as many people as you thought at the very beginning. Those are much easier to estimate projects compared to machine learning projects. Because machine learning projects, at the very beginning, are much more stochastic – you don't know what exactly to expect. I mean, of course, if it's something like a recommender system, and you've done 10 recommender systems in the same domain, you might say you have some understanding of what to expect.

But what I'm seeing is that in machine learning [projects], the amount of things that might severely affect the outcome is much higher than in any other [types of] projects. Now, what is the idea of a design doc? The thing about that is – it is ridiculous to imagine a crew of construction workers creating a building without a blueprint, right? Let's say, 100 construction workers gathered and said, “Okay, guys! Let's build a school!” And they just start to dig, lay out the foundational work, etc., without any blueprints. You can't imagine that, right? That's impossible. Nobody will give them any permission to do that. Nobody would give them anything to do until they have the proper blueprints. Now, you worked in many different companies, and you have plenty of experience, Alex. I have a question for you – how many times have you seen a project being started without the proper design documentation? Let's say out of 10 times, how many times would you see a project without the proper design documentation?

11:09

Alexey

**Well, if I'm being honest, like nine and a half? [chuckles]**

11:13

Valerii

Nine and a half! Crazy, right? Isn't it crazy!? Now...

11:18

Alexey

**It depends on what you mean by a “proper” document. But sometimes, projects do start without any documents whatsoever.**

11:26

Valerii

Let's say you don't have any document (documentation), surely you don't have the “proper” documentation? 100%, right?

11:33

Alexey

**[chuckles]**

Yeah.

11:34

Valerii

But you see – nine and a half out of ten. Maybe eight, whatever. But it's a majority, which is ridiculous. So when I'm saying that the goal of the design doc [is for the project] to fail, imagine that every time, there is a building to be constructed – people, engineers, architects – conduct some calculations. Is it feasible to do? Let's say, someone will come to you and say, “Hey, guys! I want a 10-kilometer-tall building.” Then they will try to outline that, they will try to do the blueprint, and it will tell you, “Sorry, we can't do that. That's impossible.” It is much better to find that out on the blueprint stage than right after you've already built like 500 meters, and then you say, “Oh, by the way, we can't do that.”

So in that case, if you have the proper documentation, and you outline most of the things (corner cases, etc.), then you already have information that will help you to understand if you can do that. In that sense, the goal is to fail because you want to understand that this is impossible before you invest too much time, effort, resources, whatever. But if it's not impossible – if, let's say, you don't need a 10-kilometer-tall building, but you need a school. You probably still would like to have a blueprint of the school, because my idea of a school and your idea of a school – it's still a school, right? But it might be very different. What does the school need? That's what the design document tries to outline – to solve. “What do we want to have at the end? What do we need to have at the end?”

You see, there is a difference between want and need. And “Can we can it or not?” That's the thing. Because the situation, both I and apparently, based on what you just told me you saw in the industry, [can see that this] is absurd. People are trying to build extremely complex solutions without laying the proper groundwork.

14:01

Alexey

**Yeah. So one of the goals of this design document is to think of all the possible ways of how it can go wrong, right? You just think, “Okay, this is what I want to do. This is what I need to do.” These two things are already quite difficult. And then you think, “Okay, in order to achieve that (in order to build that), we need to do this, this, and this. What can go wrong when we try to do this?” Maybe the data is missing, maybe a lot of traffic comes in, and then our system breaks – things like that. Right?**

14:36

Valerii

Yes, I'd say so. Not only what can go wrong, but also, our decision in the early stage definitely affects the next stage. Again, let's take the analogy of building a school. If you need a five-story school or fifty-story school (I've never seen 50-story schools, but...) you probably need very different things to be done at the very beginning. You probably need very different fundamentals for the school. The same goes for machine learning. If it's a real-time system, or if it's a batch processing system, which you might decide at the very beginning – this might really affect the outcome and what you need. So it's not only the problems, it's only the possible, let's say, split criteria. But another thing is that, as a single person, or even a small team, it's very hard to think about everything. Again, it depends on your experience and how typical the project is.

But the good thing is that if you have some kind of ideal blueprint in your head, it is very hard to share it with other people. You can even come to other people, and look into their eyes, and they still can't download it from your head. It doesn't work that way. Unfortunately, maybe Elon Musk and his Neuralink technology will solve that, but you can't do that. That's why we had to upload this information on paper, or digital paper, like a Google Doc. Then you can share it with many different people. Let's even forget about that – the vision you have on day 1, and the vision you might have on day 100, you might think that they're the same, but you might have forgotten some stuff and... and imagine you now have three people on the team. Are you sure you have the same vision if it's all in your head? How synchronized are you? I don't think you're very much synchronized. So having that on paper helps you to A) Remember B) Outline C) Share with other people who will provide feedback. There are many different possible ways to get feedback, from, “Actually guys, you don't need an ML system to do that,” which is the best [feedback] because, as you know the best code is code that is never written – or written with as least code is possible.

The same applies to systems. If you can do something simple, do something very simple. An if statement is always better than linear regression, and linear regression is always better than a non-linear model, a non-linear model (like boosting) is usually better than a neural network, etc. So if it is simple, it's good. But then, the last thing is, you can receive all this feedback from the people, and they can highlight the things you could miss. And also, if you think about that, if you've done that, and you have this culture in the company, that means you could also read other people's docs, and enhance and widen your experience by at least reading it.

## The challenges of machine learning system design

18:20

Alexey

**Yeah, that's really great. One of the things that you mentioned was that we want to have a shareable document, where we write what we need and what we want (and that these are different things). And I think this is one of the common challenges that we have when we actually design a system, right? In advance, we don't always know what we need – we only know what we want to have, and these things could be different. I wanted to talk a bit more about these differences, and in general, the common challenges that we have in addition to that when building systems.**

19:01

Valerii

Well, you have a wealth of experience, and you know that one of the hardest things is to maintain. Let's say that. You could say, “Oh, I need to do things right. I will create a proper document.” And let's say you've done that. On the next day, the document you created is obsolete – it's outdated. Because you might say, “Oh, actually, we need to change something,” or, “Actually, that's not what we needed. Actually, we had a new input.” Then, a week later, a month later, or six months later – you would see that the document and the reality diverge. And so, it is very hard to maintain the document. Because [things happen] like, “Oh, yeah. I will do that. It's just a small thing. I won't put that into the document.” What you said is 100% right. Life will change. Software is never done. That's why you still have tens of thousands of people working at Facebook, at Google, etc. You might say, “Yeah, but Facebook already exists as an app. What do these people do?” Or Google, “It already exists as a search.” But no, you always need to have people, right? And obviously, if you think about a product like Google Search – it probably had a design doc. How many times had it been updated? Many. [chuckles] And it's probably still being updated. [Alexey agrees]

So that's why you have to keep your design doc... a design doc is a living thing. Actually, if we come back to the original title Why ML Design is Broken – because people think about the design doc as just some artifact, “Okay. It's done. Let's mark the box and go to the next stage.” It is never done. It's never done because the system is constantly changing. One thing that is broken is that even if people have prepared the documentation – which, as we've already discussed, is a very rare thing – it becomes outdated pretty soon, if it's not a living thing. I remember when I was working at Yandex as a Yandex advisor, the one thing I was astonished is the quality of documentation. The documentation there was extremely good and well. Instead of taking other people's...

Well, technically, you can use the word molesting, but now molesting has a different context. You still can use it if you're noisy or taking someone's time. Technically, if you take a look in the dictionary, you can still use molesting, like, “Oh, I molested my co-workers. I took a lot of their time and I was noisy.” So yeah, we need to count this. But I'm not a native speaker. Anyway, so you're taking the time of your co-workers, you are spending a lot of your time, you're spending a lot of time doing improper things, instead of which, you can just go read the docs. And that's perfect. That works. That's actually why Stack Overflow is such a great thing. Basically, to some extent, it's like a living doc, right? You can go there, you can find the answer to your questions, and you leave it immediately – everything is good. I just checked the dictionary – you can use a “molest” if you're just noisy, or you're trying to get information from someone, and they're like, “Stop doing that.”

23:20

Alexey

**[chuckles] You're very good at multitasking.**

23:24

Valerii

I have to be. But anyway, that's a problem that we don't do that. And that's why it's broken. We don't maintain. We think, “Oh, it's done.” However, to some extent, it's a design document, it's like code. Code is never done – code is never finished. The same applies to design documents. It's never finished. In this sense, it is very different from blueprints. Right? You probably won't expect the blueprints of the building to change over time. I mean, you might have some additions but, usually, they remain the same.

24:06

Alexey

**Hopefully so. If it changes midway through the building – it's already half done – and then [somebody says], “Oh, by the way, we decided to add 30 more stories. Surprise!”**

24:19

Valerii

“Yeah, let's do that. Actually, now it's not a school, it's not a helicopter park.” “What is a helicopter park?” “It doesn't matter. Now, it's a helicopter park.”

24:29

Alexey

**Yeah. It's good that we work in the software industry, right? Maybe.**

24:36

Valerii

## Yeah, that's reasonable.

## The importance of updating the design document and assigning responsibility

24:37

Alexey

**If I summarize, one of the reasons why the machine learning design document might fail, or why a project might fail, is that we assume that the design document is a static thing. We finish it, and [believe] it's not going to change. But it's actually a dynamic thing. It changes. Like code, it's never done. It's never finished. It requires maintenance. Because requirements change, life changes, and things will be different tomorrow, and we need to account for these things. We need to go back to our document and update some things. What are the consequences of that? Let's say, we're working on a project. We have a design document. We already start implementing this. Then when a requirement comes in and we need to update the document. What happens then?**

25:30

Valerii

So here's the problem with that. It's very hard to imagine one person working throughout the whole system, especially if it's a big project. If it's a one-person project, you probably don't even need... Well, I'd say you still need a document but it's a different thing. However, if it says a team project, that means you probably have different people responsible for different things. Again, I will appeal to your experience. Have you ever noticed that, let's say, if you have a 10-person team, every person in this team might have a slightly different (or very different) understanding of who is responsible for what?

26:26

Alexey

**Quite often. Most of the time, yeah. Unless it's explicitly written somewhere.**

26:33

Valerii

Right! That's exactly the case – unless it's explicitly written. So the first point is, it has to be written somewhere, like, “Person A (Alice) is responsible for this. Person B (Bob) is responsible for that, etc.,” You probably would like to have some overlap, because you don't want only one person... you don't want to have a boss factor being equal to one. But why not have a design doc as a place of accountability and areas of responsibility? Now, as soon as you know who is accountable for what, and who owns what, those people are now responsible for updating their parts of the document. Which makes sense, right? Because they know their part of the system the best. It's probable that nobody else knows it better than they do. And as soon as there is something new, which might pop up quite often, they go to the doc, and they update it. Sometimes it's also nice to notify other people, “Oh, by the way, there is a new change coming.”

Now it's much more feasible than having one person running around the 10 different team members (or sometimes 10 different teams because it depends on how big is the project – sometimes the project might be really big). You might have 10 people, you might have 10 teams, you might have 10 teams with 100 people each, with 10 subteams each – you probably would like to have this accountability and ownership. Then those people can maintain the document distributed way. Because otherwise, it's very hard – it's almost impossible – to have a keeper. Maybe you can hire a full-time job person whose goal will be to keep the document updated and a living thing. We usually call these people “project managers,” right? [chuckles]

28:41

Alexey

**Or... Yeah, product managers sometimes.**

28:45

Valerii

Well, I'd say that a product [manager] is for a product. [cross-talk]

28:50

Alexey

**Sometimes it's their responsibility. When the team does not have a project manager, then somebody needs to write it. Then it's like an elite project manager or an engineering manager. Somebody has to do this, right? To your point, there is not always a person whose full-time job is only that.**

29:11

Valerii

There is never a person whose full-time [job is only that]. I have never seen that. No.

29:16

Alexey

**I think I remember when I worked at one of the companies – it was quite some time ago – there was actually a person responsible for only the documentation. But it was customer-facing documentation.**

29:29

Valerii

Okay, yeah. That makes sense. But that's a bit different, right?

29:33

Alexey

**Yeah. That's different. That's very, very different.**

29:38

Valerii

It's very hard for me to imagine... Well, I mean, it does make some sense because it's always... I've been thinking about that for a while. There is always an equilibrium – a balance – between efficiency and redundancy. Obviously, having a person whose full-time job is only maintaining and keeping the documentation seems like a thing that not every company can afford. It doesn't seem like a very efficient way of spending your money. On the other hand, if you need redundancy – as you know, there is always redundancy or efficiency – if you build a system that has to be reliable, you probably think about a lot of redundancy because if one machine goes down, you don't want the whole system to go down. That's why you'd like something to be double-checked and be in abundance. The question always is, “What is more important – efficiency or redundancy?”

30:45

Alexey

**Okay. I guess that's another reason why, or another challenge or flow, that we have in the machine learning system design practice – we don't have accountability included in the design doc. We don't know who is doing what. If we don't have that, then who is going to maintain the different parts of the system or different parts of the design document? That's another reason why things might go wrong, because there's a change, maybe, and nobody documents this change. People just talk about this at a meeting (on a daily) saying, “Hey, there is a piece of feedback coming from customers, and we need to account for that. We need to do something.” And then somebody says, “Okay, I'll just go and create a fix or a workaround. I'll fix that thing.” But it doesn't get updated in the documentation, and then we have a problem. A solution to that would be that the person who's responsible for that area makes sure that the code and the documentation are kept in sync. Right?**

31:59

Valerii

You probably have heard about this horror story from Twitter. They recently were bought by the extravaganza billionaire, who decided that the best idea is to lay off most of your engineers. Which is okay, because Twitter was very well known (was notorious for) being the place where you can chill out and not work at all. However, the thing is that, two weeks later, they had to rehire hundreds of those people because it turned out that some of these people were irreplaceable. Well, it's not like people are replaceable – it depends on how many people you have with a given set of knowledge, which is basically a bus factor. How many people have to be in the bus or accident for your project to close? If you have this accountability written in the document, it's very easy for you to understand what your choking points are. What are the bottlenecks? What are the risk areas? Let's say, you find out that there is only one person who knows about A, B, and C in your system. That's probably not the thing you'd like to have. Right?

33:30

Alexey

**How often does it actually happen in practice? Sorry for interrupting. I think what you're saying makes total sense, but I'm just wondering how often it actually gets implemented. How often can you open a document and see, “Okay, we have these risk areas. This part of the project depends only on this person. I can't fire that person”**

33:51

Valerii

Okay, I can tell you that every proper engineering manager implements that, at least in their heads.

34:00

Alexey

**Okay.**

34:01

Valerii

Because they want to understand... they need to appreciate the possibility of some people leaving or not performing – how fragile their project is. Now, I have an idea for a startup. I'll tell you [about it]. This startup is like no layoffs.com or.ai – I already proposed a name. This is how it works. You create the design doc, and in this design doc, you'd like to have a list of people accountable for specific errors. And you know that you have to have at least one person (or probably two). Now, you try to make a combination to have a maximum amount of people being only one or two in these areas, so you can't lay them off. You understand, right? Because, let's say you have three people and you have six areas... Or let's say, three areas, and three people, and each person knows every area. So that means that your boss factor is three. You basically can fire one person, and you still have two people knowing each area. But instead of that, now imagine that one person knows only one area. Suddenly, your boss factor is one, and you can't find anyone. This startup will provide you with a design doc that allocates people to specific areas, so you can't fire them. That's an idea that I'm just giving to the world.

35:34

Alexey

**Do you actually need a startup for that? You can just assign a single person to a topic.**

35:40

Valerii

Yeah, if they can do that. You'd be surprised how bad people are with combinatorics.

35:47

Alexey

**[chuckles] Oh, okay.**

35:49

Valerii

You know the traveling salesman problem, right? It is a very well-studied problem in discrete math. There are some people who just created libraries, and they sell them for money to companies that need this kind of optimization. Because they can just do that better. So how much are you willing to pay for being sure that you won't be laid off? This is a very good startup idea. Think about it.

36:19

Alexey

**[chuckles] Yeah. I guess the main idea here is that you want to be redundant – you want to have multiple people knowing at least multiple areas. Right? But then... [cross-talk]**

36:30

Valerii

Yeah, it's like a toolbar. What do you want – efficiency or redundancy? If it's a critical component, you'd like to have more redundancy.

36:47

Alexey

**Okay, interesting.**

36:48

Valerii

It's always a trade-off.

## The importance of modularity for the design document

36:50

Alexey

**Okay. I think one of the things we talked about quite extensively here is that things change – life will change, and we need to account for that. Our design document has to be non-static – it has to be a dynamic thing. So how do we build, or how do we design a machine learning system in such a way that it's lean, it's maintainable, it's extensible? When a new piece of requirement comes, we can actually go there, know who owns it, and this person can just go and update it. How do we do this?**

37:29

Valerii

So the answer is very simple, right? You go and you buy our book. [Alexey chuckles] You have a design doc outline as an example for two different things. Probably later, we will create a course that people can buy and learn how to do that. However, the answer to that... Well, obviously, you'd like to incorporate meaningful things into your design, right? You might want to have chapters. You'd like to have a modular structure – module number one, module number two, module number three. Therefore, if something is added, it's not a monolith – it's just another module that you can add. Then you know, again, who's responsible for what. Nobody's preventing you from even writing a specific test to see what the last time a specific chapter was updated and have some kind of alarms [to tell you that]. Again, it depends on how far you'd like to go. You know that you can link tickets in Jira, for example, and any other task tracker, and your GitHub or whatever you have as a code version control system. They say it might be done because as with any mete document, you can do whatever you want. You can reuse a piece of code.

Let's say, if you know that the last time the code for this chapter was updated was a week ago, but the last time the chapter was updated was a year ago – something is wrong. Again, it always depends on how important that is to you. Basically, what I prefer is just to have a clear structure – a clear outline of the design doc. The design has been split into chapters. Each chapter or subchapter has its owner (or maintainer, or person accountable for that) and then you have a list of people working on the project. What areas do they cover? And what is the coverage factor for each area – how many people can cover the specific area? One of the most important things is that, in some companies, for engineers, they review how many divs (we call them divs in Meta) or pull requests, as you probably call them, usually – how many code changes the engineer has made. Sometimes that's a metric.

I think that another important metric is how many times the design doc has been updated. If this metric will be incorporated... Basically, if you have engineering excellency as your performance review X, it makes sense to incorporate the design doc there as well, because what is usually incorporated [there] is code divs, and code reviews. If you think about that, design doc reviews are even... Well, it's hard to say more important, because it's very hard to assess what is more important. But they're also very important.

40:55

Alexey

**[inaudible]**

40:56

Valerii

Well, hard to say. [cross-talk]

40:57

Alexey

**Because it impacts our entire system?**

41:01

Valerii

Yeah, it impacts the full system. But again, you can't say, “Oh, I updated three lines. It is ten times more impactful than the actual code, which is doing that.” Do you see? As soon as a metric becomes a goal, it ceases to be a good metric, but we need to incorporate something like that in the performance review. Because as soon as it is incorporated, people will start chasing it. But they have to have some incentive to do that.

41:34

Alexey

**So what you're saying is that in order to be able to maintain our documents, [keep them] in good shape, and in order to make them extensible, it's not enough to just have a clear structure – we also need to give a bit of an extra incentive for people to do that. For example, we can make sure that their performance review depends on how good they do this.**

42:00

Valerii

This is good, because if you only have the code, that's only a part of the job, right? You know that some companies, they don't even do proper code reviews, which is bad. People are struggling, saying, “Oh, how can we incorporate that? How can we work on that?” So code reviews make sense. Nobody asks, “Oh, why do we spend time and extra effort on code review?” You do that to check if the code is correct, and to increase your bus factor – because you want to have more people that have at least some knowledge of the code base. The same applies to design. It actually doesn't matter if it's machine learning system design or just system design. You'd like as many people being there as you could, knowing and working on that. It also gives them an extra, let's say, opportunity for growth.

## Is there a universal template of a machine learning system design document?

42:55

Alexey

**We have a question that is quite on-topic to what we discussed. That question is, “Is there a universal template of a machine learning system design document?” I think it's related because you mentioned that a design document should be modular – there should be chapters, and it should follow a clear structure. So maybe you could give us an example of what such a design document can look like. What are the things there? Maybe you could use some examples, without making it too abstract.**

43:31

Valerii

I think that it's very hard to say if there's one template that's like “the best”. Well, apparently, these templates exist in our book. Right. It has the best in the world – you can't find... There are two design doc examples, but they follow the same template.

43:51

Alexey

**Uh-huh. Okay.**

43:53

Valerii

So if you don't want to spend like 30 bucks on our book (which actually brings me a lot of sorrow to find that out) you can just... if you can share the link to our book, what people can do is just read the outline. Our book structure is built as a design doc itself. Every chapter basically describes a chapter you would expect to have in the real design doc. We discussed why it's important, what has to be there, what can't be there, and at the end, we provide two examples. But basically, you take the outline, it's already kind of an example of the design doc itself. Again, it depends. For example, we were thinking about having 16 chapters, and then an appendix. But then it turns out that the appendix would be as big as the book itself, so we decided to keep it for a second book. But basically, you can go there, and you can find out the outline, like, if there is a problem, preliminary research, design document loss function and metrics, gathering datasets, validation schema, baseline solution, error analysis, training pipeline, features and feature engineering, reporting, integration, reliability and monitoring, serving and inference optimization, ownership and maintenance. So you see, some of the things we already discussed, like even ownership and maintenance are parts of this design document. Integration.

Basically, there are 16 chapters, and every four chapters are grouped into aggregated parts. Part one – preparations, part two – early stage, part three – intermediate steps, part four – integration growth. But obviously, you can add other chapters. For example, you could say, “Oh, we need to add a module on fairness.” It's actually something we were hoping to cover in the appendix, but it turned out that we already wrote too many words. Because the book has to be between 100-220,000 words, and it's already too big for the appendix. But maybe the book will be a successful one. Then we'll see if the second will come.

## The importance of monitoring and fallback solutions

46:50

Alexey

**Okay. So keeping my fingers crossed. So my question was also a bit of... what I was struggling with sometimes is maybe having some examples. I guess right now, if somebody wants to have an example, you mentioned that in the book, there are two examples. I think we also talked about that with Arseny, who is the co-author, like a month ago, maybe more. So maybe we can talk about an example but also think about how things could go wrong? Do you have, maybe in your practice, things that went wrong? I assume that you do. Maybe, you could tell us what the reasons were?**

47:46

Valerii

100%. In our book, we covered that in the chapter on reliability and monitoring, and the fallbacks. Basically, it doesn't matter how good your system is, it will go down sooner or later. Google went down, Facebook went down, AWS went down, and Twitter is going down. [chuckles] Twitter went down. You know, just this weekend, Twitter has a lot of issues with rate limiting and apparently DDoSed itself. Let's say that there are many things that might go wrong in an ML system. There are two things you can do. You can monitor what's going on. Is it a data drift? Is it an error in the data? Is it a concept drift? Is the prediction drift? Or whatever. And the second question is how you could react to that. You might have a fallback. Let's say your model went crazy, but you have some other solution that is worse than the model, but better than nothing. So if you have that, you can fall back to that solution, but you can only do that if you know that your model went crazy in the first place.

If you don't know that something's wrong, you can't react. Basically, we spent 30 pages writing about what might go wrong, how to detect that, and how to react to that. Another thing I can suggest to people who are interested in this topic – there is a website called Evidently AI. Basically, it's a startup (a company) that has a library (a framework) for monitoring. Monitoring, meaning model quality, data drift, target drift, data quality, etc. They have a very comprehensive set of blog posts and articles which you can read. They can tell you what might go wrong, how to detect that, and some basic stuff you can do to react to that. But detection is the first step. Because if you can't detect that, that's it – finish.

51:01

Alexey

**This reminds me that we had Helena, one of the co-founders from Evidently, like two years ago. The title of your talk was, “Why Your Machine Project Will Fail,” where she outlines (describes) pretty much what you said. So monitoring is important, and most of us have heard that, probably. I'm wondering – okay, we know that we need to monitor. We need to have a fallback solution, and we need to know that if our machine learning model goes crazy, we can actually fall back to our...**

51:36

Valerii

Let's say that it might be a model problem, it might be... because we mentioned four things – data quality, model quality, concept drift, and prediction drift – there are many different things.

51:47

Alexey

**So how do we account for that in the design phase? What place should these things have in our design document? And how do we actually structure them?**

51:59

Valerii

Well, we think that accounting for that is a part of... Let's say, it will split the design doc into four parts, like preparation, early stage, intermediate steps, and integration growth – we think that reliability monitoring and fallback is the integration and growth part. It's definitely not preparation. It's definitely not early stages. It's even not intermediate steps. It's something where you already have a system, it works, it has pipelines. You'd like that to be reliable, and you'd like to have a fallback. Basically, your whole focus at the end is on reliability. Remember – efficiency, redundancy. Reliability, to some extent, is redundancy. You might have three different fallbacks depending on what is broken. Maybe some data is damaged, but you have a set of three critical features, and they are not affected.

You still might have a model, which is much less sophisticated but still better than nothing. Let's say, it's two if statements. If the data is completely affected or corrupted, let's use three or two if statements. If even that doesn't work, let's use a constant placeholder. As you see, you have four solutions to use only once – a lot of redundancy, but a lot of stability and reliability. But yes, that's something that has to be addressed. If you think about that – if you take a look at least in our part four outline – reliability and monitoring, and ownership and maintenance, are all about these fallbacks concerning what might go wrong. It might go wrong from the technical side, it might go wrong from the people side – how do you take these into account?

54:07

Alexey

**Okay. I want to summarize everything we talked about so far. We started by discussing a design document – a machine learning design document. We talked about what we needed to document there – what we need, what we want. Then we started talking about why some things might go wrong. One of the things we need to keep in mind is that the design doc has to be a dynamic thing – a dynamic document. It's not a static thing. Then another thing we mentioned was that we should know who is accountable for what. If there is no accountability mentioned in the design document, then nobody knows what they're responsible for and then nobody will keep the document updated.**

55:01

Valerii

Worse than that. It's not that nobody knows – people think wrongly. People think, “Oh, somebody's responsible for that,” and this person thinks something else. So this is even worse than that.

55:13

Alexey

**Right. Okay. So we need to think about that and explicitly document who is accountable for what. Then another thing that we need to keep in mind is that the document should have some sort of modular structure – it should have chapters – and we should know the last time each chapter (when each model) was updated. Thus, we know how outdated the document is in general, and how much effort we put into maintaining it. So that's another thing.**

**Towards the end, we discussed that we sometimes forget to think about monitoring and there are many things that might happen, so we need to account for that too, but maybe at a later stage – the growth stage (I think, I don't remember). Or the one before that – the integration stage. Basically, at some point, we will need to think about that, and we need to keep that in mind. Did we not talk about something else – something that is equally important to these four things?**

56:24

Valerii

Look, everything is important, right? Because every project is quite unique. For example, one common pitfall, which probably both you and I have seen, is when people try to do something complex and sophisticated, instead of working with a simple baseline solution that can help them to test the hypothesis, to iterate very quickly, and to see what an extra mile can bring. Not that many people do that and that's extremely important – to build the baseline solution. But if you don't build a baseline solution, it doesn't mean that your project will fail, but you might spend an extra three months of work, which is probably not great. Let's say that. We decided to focus on these 16 chapters because we think they are important. We obviously haven't gone through all of them because it'll take longer than an hour. But what I suggest for you guys to do is to just take a look at the outline – it's free to look at. You can then further dive into any specific topic you'd like. We've already spoken about Evidently AI, where you can dive into the specific topic of reliability and monitoring. As soon as you have this idea of these chapters being important, then you can start digging further into them, and discuss them in your own design documents.

58:15

Alexey

**By the way, while we were speaking, I went to our previous conversation with Arseny, who is the co-author of the book, and I found out that we have a discount code for the book. I included the discount code in the description of this episode. I think it's like 35%. Feel free to use it if you want to buy the book.**

## Valerii’s resource recommendations

58:37

Alexey

**I want to ask you one last thing. We already talked about resources. One resource, if somebody wants to learn more about ML systems design, is obviously your book. The other resource is Evidently's website. Is there anything else you would recommend for the listeners to check out if they want to learn more about this topic?**

58:59

Valerii

Look, I know that there is a Telegram channel, but it's in Russian. It's called Reliable ML. Those people are also working on a machine learning system design course, they have a template – I don't remember if it's in Russian or in English. Because if it's in Russian, then people who speak Russian already know about them. It doesn't make sense to tell other people about this because it's in Russian, right? If it's in English then... Let me check which language they use because I just don't remember. Ah.

59:53

Alexey

**Is it Reliable and Reproducible ML? Something like this?**

59:58

Valerii

Well, don't ask too much. They just call themselves “reliable” and then they don't call themselves reproducible.

60:06

Alexey

**Maybe I'm confusing them with somebody else.**

60:08

Valerii

Okay, there is a template – it's in GitHub, and it's in Russian. So, no. By the way, they have some materials at the bottom, and they have some links to an ML design template from an ML Engineering interview. An article design document for ML models on Medium in English. Okay. You can add some of these links. I can share it with you.

## How to find Valerii online

60:44

Alexey

**Please send the link and we will include them in the description. If somebody wants to get in touch with you and ask something, what's the best way of doing that?**

60:54

Valerii

I think that the best way is to go to LinkedIn and ping me there. I'm always happy to chat. Well, depending on what you want to ask me, but in general, I'm very open to chatting.

61:17

**Alexey  
Okay. Thanks, Valerii, for joining us for the second time. It's always a pleasure speaking with you. Maybe, as you mentioned at the beginning, as you join a new company and you encounter and solve new challenges, it will be pretty interesting to discuss these challenges with you at some point in the future. You're always welcome.**

61:38

Valerii

Hopefully. Hopefully. I hope so. Thank you, Alexey. Thank you so much for the invitation. Have a nice day.

61:44

Alexey

**Yeah. And thanks, everyone, for joining us today too. Have a nice day and the rest of the week. See you soon!**