Software Engineering for Machine Learning: Characterizing and Understanding Mismatch in ML Systems

Featuring Grace Lewis and Ipek Ozkaya as Interviewed by Jonathan Spring

---

Welcome to the SEI Podcast Series, a production of the Carnegie Mellon University Software Engineering Institute. The SEI is a federally funded research and development center sponsored by the U.S. Department of Defense. A transcript of today's podcast is posted on the SEI website at sei.cmu.edu/podcasts.

Jonathan Spring: Welcome to the SEI Podcast Series. My name is Jonathan Spring, and I am a senior vulnerability researcher here at the SEI CERT Division. Today, I am joined by two illustrious colleagues, Dr. Grace Lewis, principal researcher and lead of the Tactical and AI-Enabled Systems Initiative here at the SEI, and Dr. Ipek Ozkaya, technical director of the Engineering Intelligent Software Systems Directorate also here at the SEI. Today, we are here to discuss their research in software engineering for machine learning. Welcome to you both.

Grace Lewis: Thank you.

Ipek Ozkaya: Thanks.

Jonathan: Grace, Ipek, can we start with telling our listeners a little bit about yourselves and what brought you here to the SEI?

Grace: Sure, I can start. My name is Grace Lewis. I came to the SEI, hard to believe it, but 20 years ago after completing my masters in software engineering at Carnegie Mellon University. I joined as an engineer, meaning that I was a software developer on several projects that were ongoing. Very quickly, I learned that my passion was really the research. I mean, I like engineering, but my passion was really the research side.

After that I got a PhD in computer science from the Vrije Universiteit Amsterdam. Now, with the SEI, I lead an initiative, like you said, called TAS, Tactical and AI-Enabled Systems, where we conduct applied research and development in three focus areas. One is tactical-edge systems. The second is software engineering for AI systems. And the third is the intersection of the two, meaning how do you do AI at the edge.

Jonathan: Great. Thanks. Ipek, how about yourself?

Ipek: Well, thanks for having us. Ipek Ozkaya, and I came to the SEI longer than I actually can admit to myself, around 2006. This was right after I completed my PhD at Carnegie Mellon University in a field called computational design. That was applying software engineering to design professionals and to architecture and design tasks. But I was very passionate about software and its research. Ever since I joined the SEI, my focus has been on software architecture and combining software architecture practices and applying it with organizations. My current research is focused on understanding how to manage technical debt, again which builds on architecture and software architecture principles, and also with working with Grace and our other teams on understanding what software engineering for AI-enabled systems looks like.

Jonathan: Thanks. I know building on both of these pieces of work for ML systems, you two recently wrote a blog post exploring software engineering and ML and characterizing what you call *mismatch* in ML systems. How does this problem of mismatch come up with machine learning, and can you describe the entry point for the problem or why we care about it?

Grace: Yes. Sure. So the origin of mismatch is that, when you think about how machine-learning systems are built, they are built by three different types of teams. You have data scientists or machine-learning engineers that are building machine-learning models. Then that model is typically passed on to a software engineering team for integration into a larger system. Then the system itself is passed on to an operations team that now is in charge of operating and monitoring the system as a whole. The problem is that, if you look at how these teams are composed: very different people, very different backgrounds, very different vocabulary and, in the end, also very different system concerns. So what happens is that these teams, because they also operate independently, unfortunately tend to make assumptions about what the other parts or the teams in the systems are doing, and that is what leads to mismatch. So basically, mismatch is a problem that occurs because of an incorrect assumption made in isolation by one of these teams. In addition to that, we found that you can trace the cause of the mismatch to some piece of information that should have been shared between stakeholders, and if it would have been shared, it would have avoided that problem.

Jonathan: Great. I am sure that misaligned assumptions are not totally new, but, Ipek, can you talk a little bit about how the mismatch shows up from traditional software engineering and how that gets maybe worse in machine-learning systems?

Ipek: Of course. And as you have identified, mismatch is not new to software engineering systems and actually any engineering system. It is really the assumptions that different parties make and how you communicate those assumptions. What is different is, in the software engineering systems, we have techniques that we actually anticipate and we have kind of

safeguarded with it. What has been catching a lot of the organizations who are developing now ML-enabled systems unprepared is the data scientists. The new set of technical stakeholders that are joining the development of these systems, first of all, may be making different assumptions. Their techniques are different. The way they measured success or how the elements that they develop, like the ML models are developed, behaviors are different, and these models go through so many different stages that software engineers may not always appreciate. So putting that together, there are actually new categories of mismatch or mismatch assumptions that occur and also, maybe opportunities to safeguard the systems, both with automation or other techniques. And that was the observation that we made, and that's how the research started.

Jonathan: I think that we will want to talk to what we might recommend people can do to safeguard against different kinds of mismatch, but it is clear that we need to know what they are before we can safeguard against them. Can you tell us a little bit about how you would go about methodologically characterizing the new kinds of mismatch and maybe also interesting places where you have identified the same mismatches [but] maybe more impactful. How would we go about studying this?

Ipek: Let me get started. The approach we took was an empirical approach. There were some observations we had started to make based on the organizations we worked with, things we read, our research, and whatnot. But we stepped back and said, *OK, what does a mismatch really look like and do these mismatches appear the same way from a data-scientist perspective, from a software-engineer perspective, from an operations-individual perspective?* So we interviewed individuals across the board—that included a combination of these different expertise levels—and we collected their examples. Then we also supplemented that information with things that had been written. There are quite a number of actual papers now coming out from experiences—Microsoft, Google, and some of these organizations who are doing it—and there are examples there as well. And as a result, there were about 140 examples of mismatch that we collected, then we went about with a characterization exercise to see whether there are some categories that are emerging, and that kind of started snowballing the research.

Jonathan: Great. I think that that is a really great example of mixed methods: social science and general research, which is super important and, I think, underused to understand, at least the security side of things. I am not as expert in software engineering, but is that also something that maybe people need to do more of for software engineering?

Grace: I would say so. Like Ipek said and like you said, it is definitely a mixed-method study. One part of it was understanding, from real practitioners, what does mismatch look like? Or even, like, that was a hypothesis that we had, but is this real? Does mismatch really happen? Right? So that was the part of the interviews that we also validated with a survey. But in parallel to that, just as Ipek said, we said, *You know what? Mismatch has occurred in the past. There are*

lots of kinds of mismatch. Let's look at what the literature says and maybe are there best practices out there already for these types of systems? And so we conducted what is called a where you look not only at the white literature, which is the academic literature, but you look at the gray literature, which is basically what comes up when you Google something. So, like, more informal blog posts. And the advantage of that is that you get the academic perspective, but you also get the industry perspective because those are folks that are writing the gray literature. And so then, after that, what we did was basically a mapping. We said, *OK, we know that all these mismatches exist. Which one of these best practices can we map to those mismatches?* Then we ended up with a really, really nice matrix. Then, at the end, what happened is that all that information that should have been shared, we codified it into a set of descriptors. They're machine readable. And so these descriptors are for parts of the system. So, *This is how you describe your trained model. This is how you describe your raw data. This is how you describe your training data. This is how you describe your operational environment*. Because, that way, that information now is explicit. It is shared. The goal of making them machine readable is that our next steps are really to build tooling around this, have people build tooling around these descriptors. So yes, absolutely, totally agree.

Ipek: Part of the approach also helps us, as well as the practitioners and researchers identified. There are some categories of mismatch that will actually emerge because the field is new; you don't know some of the things you don't know. Eventually, those will fall in place. But others are actually enduring challenges that we need to do more research on, and this kind of an approach allows us to separate, *OK, some of these will eventually go away by themselves because the practices will be learned by all these individuals. There are other safeguards that are already maybe out there that people will start employing*. Others, and we have identified some of these like, for example, testing being one. That really requires more research from different individuals to actually advance the field and make these machine-readable descriptors, define them, identify more of them, developing automation and all the good stuff that comes with it.

Jonathan: Ipek, you mentioned testing there as if that's sort of a simple and one thing. But because of all of these different areas of mismatch, is there a different testing that is needed for maybe each of the pairings? I imagine that there are some mismatch between data scientists and engineers. Some mismatch between data scientists and ops. Some between ops and engineers. Are there different tests on each of those axes, and how do we go about making sure that the right people learn about the right tests?

Ipek: Absolutely. That is actually why we are going to do continuing work in this area because the work we did just scratched the surface. This happened coincidentally at the time when we were working with some organizations who happen to have challenges in, *What does it mean to test an ML model while it's being developed by data scientists? What does it mean to test that*

model when it's actually part of the system? And when it's deployed, are there techniques that are already applied by software engineers that are used? There are techniques that data scientists use, but it seems to be more of an art rather than a science in terms of how some of these organizations are going about that. So that is actually what we are going to be doing next. You are right. There are mismatches that we can safeguard. There are some, again, obvious methods that could be used. But there [are] also gaps that we need to identify and do more work on.

Jonathan: Grace, I don't know if the more interesting part for our listeners is the things that we can fix easily or the things that are going to be super hard to fix and we might be stuck with for a while. But maybe do you want to talk about maybe the good news first?

Grace: Sure.

Jonathan: And then the bad news.

Grace: Right. So here is the good news. The good news is that, at a small scale, we were able to validate our findings. We were able to validate that mismatch exists. Yes, this is very important to share to avoid that mismatch. The good thing is that, for many of those mismatches, there is not really a rule that says, *This value has to equal this value*. For many mismatches, just share the information. So, for example, when you hand data to a data scientist, tell them more about it. *Where did you collect it? How did you collect it? At what time was it?* I mean, like, just basic information or, *Where did you get it from? When the model is operational, how fast is the data going to be coming into this model?* So a lot of them, I would say about 70 percent of them are really just sharing information. For others, we did have to build rules. So, for example, when a data scientist is building a model, it is a good idea if that data scientist has some information on, *What are the computing resources that are required to serve the model?* But there you can have a formula, right? We need resources in operations to be greater than resources required for running the model. So yes. So the good news is that a lot of it is really simply information sharing. For a lot of them, there are kind of simple rules.

But there are areas that are still very difficult. The number one cause of mismatch in our interviews was really testing. Because we don't really know how to test these types of systems. For example, from the data scientists' perspective, when they say, *Yes, we tested the model*, it basically means, *We evaluated the model and the model has the expected accuracy that we were targeting*. For a data scientist, that is what testing means. For a software engineer, it's like, *Have you done unit testing?* And the data scientist is like, *Oh, unit test? I'm not sure I know what that is.* So part of the work that we're going to be doing next is really building on that because that's where we saw there was, like, the biggest gap. So, for example, how can we, through tooling, how can we help data scientists build unit tests? It's nothing wrong. It's just that they weren't

trained with that. From a software-engineer perspective, what does it mean to integrate a machine-learning model and what does integration testing look like? I guess the bad news—although you could say it's good news—is there is so much work to be done.

Jonathan: Thanks, Grace. I think that the different testing stuff is really interesting. One of the things I was curious about reading your work is whether the security side of these things falls under operations. So I have been doing some thinking about how to manage vulnerabilities in ML-enabled systems, which is sort of an operations problem. But the security community has different testing like fuzzing, or symbolic execution, or whatever sort of program-verification stuff to do something similar to what unit tests are doing, but looking for a different objective.

Grace: Mm-hmm. Right.

Jonathan: How do you think we can communicate those sorts of tests? I know that this isn't exactly what you did your research on, but can we sort of integrate all of that into the testing that we need to bring together with the mismatches?

Ipek: I think there are opportunities with the mismatches and the descriptors, both from a testing perspective, but there were other categories of mismatches that we have identified. Like, for example, the mismatch between development environment and operational environment would also provide opportunities to safeguard and catch some of the issues that you are talking about. I would not, probably, lay them all on testing, but testing definitely will create opportunities to catch some of those.

Grace: Also I would say that some of the...let's call it security testing is testing that we have been doing forever and ever and ever. For example, *Validate your inputs*. That is not a common practice in the data-science community, but it is a very common practice in the security and the software engineering community. So, it is really looking at what is out there and providing those tools for people that are not aware of what you need to do to make a secure system.

Jonathan: Yes. So Ipek, can you talk a little bit more about the mismatch between the development environment and the production environment that you mentioned briefly there?

Ipek: Some of the examples that we came across were, for example, when the ML models were being developed and so happened both in the data-science aspect and the software-engineering aspect, the environment that they were collecting the data, running the model, and whatnot was different from the actual operational environment. This was a huge resource-consumption/assumptions mismatch. Sometimes it works, sometimes it doesn't, but that created a lot of failures for some of the organizations that we talked to. And this is, I think, something that could easily both be checked as well as automated because there are attributes for these that actually can be up front, or in an iterative way, identified and brought back into the environment.

Jonathan: I know it's hard to briefly summarize the whole data lifecycle, but could you give us some highlights on where some of those data-lifecycle mismatches show up?

Grace: Absolutely. Absolutely. Kind of like from a data lifecycle, so you start out with raw data. The raw data gets processed, and you come up with training data and evaluation data. Then eventually a model gets put into production. Now you are dealing with operational data or production data. There are lots of opportunities for mismatch, not only just related to the data, but also, again, related to the different categories. For example, a data scientist gets raw data. We heard it over and over again, the data scientist saying that somebody just tosses them a piece of data and says go do some data science to this. Just go and do something. Tell me something about this data. And it's, like, *OK. What problem are you trying to solve? How will you know that the problem is being solved?* From that perspective, there are mismatches between the expectations from a product owner of what a model could do and what you actually can do with that data, if that makes any sense. So there are definitely mismatches there. Mismatches between, let's say, raw data and training data usually occur because there is not enough information about the raw data. Sure, you give me the raw data and, it's field 52, field 53, field 73. And I'm the data scientist. I'm looking at this and I'm, like, *I have no idea what field 53 is*. *It might be that I may be able to tell you something interesting about field 53, but unless I have more context, I really can't give you a good, good, good solution*.

And then, further on, okay, you have a trained model. Everything is great. But what happens? Usually, when a data-science team trains a model, you are given the data in some, let's call it static format. It's in a database. It's in a data file. It's in some data source. When that model is put into production, the data is not coming from a static data source, it's coming from a data stream. It's coming from user input. It's coming from another thing. And, again, if the data-science team doesn't know, like, in operations, what are the data rates? It's not only the data-science team, but the software-engineering team is not going to be able to say, *Yes, this model is performing well.* But not so much from an accuracy perspective but from a performance perspective like running, producing inferences in an appropriate amount of time. So I hope that answers your question, but those are some of the mismatches related to data that we found.

Jonathan: Yes. Thanks, Grace. I think that that is a good summary of the whole lifecycle from data collection through forming hypotheses in a model and moving that into production and the stuff that can happen. Some of this is essentially good scientific practice generally: have a question you're trying to answer. [Do you] have a success criteria where you know whether you have answered it or not, like understand how it's going to change your model of the world or what you want to learn from it and why you care about it. I know that, Ipek, you've been

involved in that more generally in software engineering through the journal that you are chief editor of [*IEEE Software*] and all of this. You have a lot of different perspectives on this.

Ipek: So it boils down to codification of best practices. There is a very important aspect of the ML mismatch work that is codification of the best practices. Some of these are things experienced engineers, experienced data scientists already do know. But it's really based on their experience. Once you codify, especially when you can codify and represent them as a descriptor and provide some form of ability to check, regardless of who is doing it, that is really when you baseline some of these avoidable problems. That is really what it boils down to, and that is probably what is transferring from software engineering and software engineering for machine learning. That is really how software engineering has improved over the decades. I think that information is transferring, and that is the approach we brought to the project as well.

Jonathan: Ipek, that is a really good understanding of codification. What are some of the remaining challenges in deploying, operating, and sustaining ML-enabled systems?

Ipek: There are large industry organizations that actually have been using ML models and AI techniques as part of their software. They do it because they have resources. Some of the software that they produce are not as mission-critical. One of the things, I think, especially for our stakeholders within the DoD and more mission-critical, safety-critical domains, sustainment and maintaining these systems will actually probably pose new challenges. We don't know what we don't know there yet. We are mostly focused on the best use of algorithms, developing algorithms, making sure that we improve the precision, their decision-making capability, ethics, bias, privacy. All these good things that are actually involved in developing ML-enabled systems. But once, I think, these systems are out there in the field, we will start seeing some of their sustainment and deployment challenges. We will see how we will actually be able to upgrade. Some of the work we are doing in ML mismatch will definitely contribute to that. But there is more work to be done in terms of how we monitor them, how we scale them, how we architect for them. Those are all, I think, going to be things that we will learn more as we have more experience coming from the field as well.

Jonathan: That is great. Grace, can you talk a little bit about how those challenges that Ipek just mentioned might inform your future work or the future work of our colleagues here at the SEI?

Grace: Yes. Absolutely. First of all, we have upcoming work in addition to building tooling around mismatch detection, like we said before, we are really focusing on testing because that is a big issue, on providing support for testing not only for data scientists but also for software engineers and even operations folks. In addition to that, as people that have followed the SEI for many years, especially our software-architecture work, we have a big emphasis on quality attributes, understanding a system through its quality attributes. What does security mean for a

system? What does scalability mean for a system? And all the *-ilities* you can think of. One big quality attribute that is not new but I really think it's going to be key for these types of systems is monitorability. If you think of the model, I mean, the model only knows what it knew at training. If it starts seeing new data, it is not going to perform well. It might be that the environment itself changes and then all of its predictions are somewhat different. So being able to develop systems that have monitorability built in so that you can tell that something is different in the system. I think that is going to be a big challenge, like Ipek said, as systems are deployed. And it is something that we are definitely very interested in, given, like I said, our quality-attribute focus. I just think that is one of those quality attributes that is going to emerge.

Another type of quality attribute is that we talk about ethics a lot. AI and ethics. I know that is a big theme, but what does ethics mean from a quality-attribute perspective? How do I build a system that shows that it is ethical? How do I say, *This design is more ethical than this other design?* So it is understanding those new requirements in quality attributes that come from this field. I think that is also a very interesting area of research that we would like to explore.

Ipek: We have efforts going on in all those areas, in addition to those that Grace mentioned. Understanding how to move these into operations, there is all the new work coming in MLOps, but there are missing parts of the lifecycle, and that actually meets well with the monitorability, how you actually provide some of the information to the operations as well as data scientists throughout the lifecycles.

Jonathan: Thanks, Grace. Thanks, Ipek. I have a number of topics I know that we could spin off to on those two things that you just said. *What is ethics from a quality-attributes point of view?* is an amazing question that I think we could probably talk about for another hour.

Ipek: Maybe next podcast.

Jonathan: Before we go too far off onto something like that, I think it is important to understand what we can transfer to practice right now out of what we already know. For our listeners that are software engineers or machine-learning engineers or operations people who are getting engaged with an ML system, what resources are available to those listeners to do a little bit better right now?

Grace: OK. So part one of this research project, which was under characterizing and understanding mismatch, was presented in an SEI webinar. So I would definitely start there. It's available on the SEI website. Just go to the SEI website, search for Software Engineering for Machine Learning under the Webinar Series. Now the paper that describes the methodology more in detail was recently delivered and published at a brand new ICSE [International Conference on Software Engineering] 2021 workshop on AI engineering called WAIN...W-A-I-

N...2021. And a preprint of that paper is available on arXiv. It's called "Characterizing and Detecting Mismatch in ML-Enabled Systems. So obviously, you can look on arXiv for that. And finally, the replication package for Part 1 of the study—if you are interested in seeing the actual data and what the mismatches were—it is available on GitHub on github.com/galewis/ml-mismatch. And the plan is to very soon—Ipek and I are working on that—is to upload the replication package for Part 2, which is also going to include the resulting descriptors, which I think are the main outcome of this project. And it is our belief that it is going to bring a lot of benefit, especially to practitioners in the field.

Jonathan: That's great. And we'll definitely have links to all of those resources in the show notes so our listeners don't have to rewind and scribble that down. So everyone can go to the show notes and you'll find links to all of those wonderful resources that Grace just mentioned. Ipek, is there anything else that you think our listeners should know right now about what they can maybe do a little bit better with what we have available right now?

Ipek: I think the SEI puts out some of the results of the early work out there. I would encourage listeners to visit our website and blog regularly. There are also other things that have been published. You mentioned *IEEE Software*. There was a specialty issue focusing on AI effect and there were some interesting papers that relate to what we have done. We [SEI] published 11 Practices for AI Engineering. I think those resources could also be starting points for practitioners or researchers who are starting to think about applying software engineering to ML engineering, ML-enabled systems, as well as thinking about AI engineering.

Jonathan: Yes. That 11 good-practices document is definitely a good place to start.

Ipek: And, in addition, recently SEI has published the Three Pillars of AI Engineering, which include Robust and Secure AI, Scalable AI, and Human-Centered AI. I think those are also available, and they're good resources to get started.

Jonathan: I could go off and talk about any of those other tangents, but I can wrap it up if you are done. So Grace, Ipek, thank you so much for taking your time to talk with us today. And thank you to all the listeners for joining us. We'll have links to all of the various things that we've mentioned during the podcast in the show notes. Grace, Ipek, thank you very much.

Thanks for joining us. This episode is available where you download podcasts, including SoundCloud, Stitcher, TuneIn Radio, Google Podcasts, and Apple Podcasts. It is also available on the SEI website at sei.cmu.edu/podcasts and the SEI's YouTube channel. This copyrighted work is made available through the Software Engineering Institute, a federally funded research and development center sponsored by the U.S. Department of Defense. For more information

about the SEI and this work, please visit www.sei.cmu.edu. As always, if you have any questions, please don't hesitate to email us at info@sei.cmu.edu. Thank you.