An Architecture-Focused Measurement Framework for Managing Technical Debt featuring **Ipek Ozkaya** interviewed by Shane McGraw

---

Shane McGraw: Welcome to the SEI Podcast Series, a production of the Carnegie Mellon Software Engineering Institute. The SEI is a federally funded research and development center at Carnegie Mellon University in Pittsburgh, Pennsylvania. To learn more about the SEI, please visit our website at sei.cmu.edu. A transcript of today's podcast is posted on the SEI website. My name is Shane McGraw. Today I'm pleased to introduce you to Ipek Ozkaya, a senior researcher in the SEI's Research, Technology, and System Solutions program. Today, Ipek and I will be discussing her research, which explores the development of an architecture-focused measurement framework for managing technical debt. Ipek's work focuses on developing methods for improving software architecture practices. Primarily, she focuses on applying economic methods and theories to software architecture practices. Ipek, thank you for joining us today. To start off, I'd like you to tell us a little about what is technical debt as it applies to our discussion here today.

Ipek: Thanks for having me, Shane. The technical debt metaphor has been around for a while. In fact, we're almost celebrating its second decade. It was first used in 1992 by Ward Cunningham at an OOPSLA conference where he described the trade-off that developers make. The idea is that developers sometimes accept compromises in the system in one dimension, for example, modularity, to meet an urgent demand in some other dimension like deadlines, schedules, and performance. Obviously such compromises, if not monitored well over time, could have an effect like debt if they're not paid back and which then should be repaid or repaired so that the system continues to function with the quality that it's expected to function.

Shane: I've seen this referred to as similar to financial debt. Is that a good perspective to look at?

.

Ipek: Well that actually brings up an interesting point, Shane, because to date, the practice that software engineering consultants face and to some extent in the past couple of years, the researchers have been using as a metaphor, which is like a conceptual abstraction of it. However, I think the more we investigate it further, it's getting closer to an analogy. By that I mean the structural aspects of financial debt, which means there's a timeline of the debt and there's interest that accumulates. Well, do you pay the interest first or the principal first? So those become actual important issues when you're dealing with it in the context of software when it applies to evolution and maintainability.

Shane: So please explain how the management of architectural debt fits into your overall interest in Agile architecting?

Ipek: So what has happened in the past decade is the technical debt metaphor was actually picked up again by the Agile community. That goes back to the first principles that the Agile software development proponents follow, which is you have to push the functionality quicker to get feedback. With that what happens is some shortcuts and some trade-offs, which might be taken on as technical debt. However, most of the work focuses on code quality. What we're actually arguing, through our work, is as the scale of systems grow, when you need to deal with systems that are going to be out there for multiple decades, systems that have to reach out to many users, the architecture fundamentals and the system of system aspects of these systems become important. That's why our work focuses on how can you actually manage those shortcuts at the architecture. Can you foresee some of the detrimental effects when you're making some of the architectural decisions? So that's how we've focused our work rather than looking at the lower level code quality aspects, we are raising the abstraction a little bit and also giving information to system managers and developers earlier throughout the software development life cycle.

Shane: Now I know you've created a game called Hard Choices. Can you tell us a little bit about the game? I know it's been very well received. How does this fit into some of the work you're doing?

Ipek: Sure. This was one of the first activities that we engaged in, both our team at the SEI—myself, Robert Nord and Nanette Brown—and also Philippe Kruchten from the University of British Columbia. We brought in Chris Simms to teach us how to create our games. We wanted to understand better how to interact with an Agile teams and bring that to some of our non-traditional stakeholders within the government like the Department of Defense. We said, "Okay, let's create one for ourselves." We took the game that Elizabeth Hendrickson from Quality Tree Software has been using which was called Short Cuts, a game about speed and risk. We adapted it a little bit more to our purposes. Our goal was to bring the architecture focus a little bit more to bear with our stakeholders. What happened actually is that when we used it with our tutorials, it caught on way beyond our expectations.

Ipek: It became really popular and the most surprising feedback we got is some organizations are actually using it to bring different stakeholders from management and engineering to the same page in terms of the issue they have at hand. So that's how we've been using it.

Shane: I know that game is freely available on the SEI website. Just type in **Hard Choices** on our site. It's a free download.

Ipek: Correct. It's on the Creative Commons license. What's available on our website is both the game board and all the ways to play the game. We also prepared a facilitator guide because we do some tweaks in terms of the facilitation and people usually find that useful as well. I think through the SEI main website, if you just type **Hard Choices** you will get to that.

Shane: Excellent. Now I've heard you say in the past that your architecture focus measurement framework for managing technical debt is informed by exchanges that you have in the technical debt workshops. Can you tell us a little bit about these workshops and how the SEI became involved?

Ipek: Thanks for asking that question, Shane, because one of our goals is really to go to the research foundations of some of the theories and method that you can bring to bear for managing this kind of technical debt in the systems. For that purpose, we actually want to reach out to our research partners and that's how they actually started. We held the first one in 2010, June, at the SEI inviting some of our researcher colleagues. From that point on, it caught on and both of our collaborators in the practice and other collaborators in research asked us to bring it to other venues as well. So, for the past two years, we've been having this workshop co-located with the International Conference of Software Engineering. The reason why we do this is we have both short term and long-term goals at the metaphor itself brings to bear. We want to be able to provide methods to practitioners in the short term, but we also want to get to the root of some of the issues we have, which is a lack of repeatable measurement framework for practices like software architecture. What are those artifacts out there? Do we really have to create a big artifact to be able to really create some quantifiable measures, or are there really quick and easy ways to do that? We can only do it by learning from people who are out there and also from other people who are attacking the problem from different perspectives.

Shane: Excellent. So speaking of research, your particular research investigates which measures a software development team can apply to effectively monitor challenging qualities of software such as degrading modifiability and extensibility of the system at each iteration or an iteration in incremental lifecycles like Agile. Please tell us about some of your findings and were there unexpected results that you found?

Ipek: So we went into the work thinking, if we want to really be able to do something that is longstanding with it, we need to be able to provide some quantifiable aspects of it. In essence,

technical debt is about rework. Can you estimate rework? Can you anticipate rework, and at what stage of the software development lifecycle do you do that? What is out there actually is—there are lots of tools and there are more tools emerging that do kinds of things like static analysis looking at the code, whether there are cycles in the code, whether there are unexpected function calls to particular classes that might actually have a burden on the system maintainability. That's all very well, and we're actually learning a lot from that work.

Our goal is to be able to do that earlier, whether when you have some architectural decisions, can you anticipate those kinds of rework that might emerge, or can you look at an architectural artifact along with the code artifacts and understand whether it will be harder down the line to add functionality. Or, if you need to upgrade technology, can you anticipate the amount of rework that you are bringing to bear? We started with modifiability and extensibility for obvious reasons because it's easier to provide some measurements on. We can learn from some of the lessons that static analysis artifacts have done. Of course, it's also harder because there's a lot of skepticism around those as well, and we actually ourselves are skeptics in that area.

If you're asking me what the most unexpected finding in that, it's our own skepticism going into it. We all thought we would come to a solution a lot earlier. We challenged ourselves a lot more than we thought we would initially. What we're doing now is we're trying to introduce other quality attributes in addition to some of the things like modifiability and extensibility. We're looking at availability and performance-related aspects like, for example, can you anticipate rework when you are creating the different caches or distributed systems? So that's where we're actually headed, and it's exciting from one perspective because you're trying to understand some of the questions that have been asked for a long time. Sometimes it's also tough because you know there's actually a lot of work to do and that's why a community movement of different researchers and folks who are using this in the trenches is very essential for our work.

Shane: Excellent. So what areas do you plan on exploring in the upcoming year?

Ipek: So there are three areas that are in our focus for the upcoming year. One of them is a way to come up with an initial algorithm in terms of measuring potential impact of rework, looking at the static structure of the architecture. So we are creating some simulations in terms of looking at the boundaries where that breaks. So that's a focus and our colleagues in the University of British Columbia with Philippe Kruchten are actually working with us on that as well.

Shane: Now are these large-scale systems, all of them? Or, does it just run the gamut of the size?

Ipek: So the initial systems are of course smaller in size, but they're real systems. They come from real systems. One of the systems, for example, is a building information automation control system that we had some artifacts for from Siemens.

Ipek Cont.: That's actually created a very useful playground for us to test our ideas. We're using some open-source systems to test the scalability of it, and then we will be putting it into a larger system. The whole idea is large-scale systems because what happens in these, especially when you bring a measurement focus, is the number of elements that you look at very quickly get into the hundreds and thousands. Our goal is to decrease the number of elements to look at while not compromising from the predictability of the analysis. That's actually the bigger question in terms of whether these techniques will ever really scale up.

Of course the other big question is, do we really need to go into a quantifiable aspect of debt, or is it good enough to just state in the metaphor realm? I tend to believe that if we can't measure it, we can't control it. So that's why I and the rest of our team actually continue in this area.

Shane: Excellent. And were there two others?

Ipek: There were two others. The second one is looking at some harder to measure quality attributes with availability and we'll be looking at some runtime patterns with that and apply the same type of analysis. The third is looking at some architectural decisions earlier in the context of system of systems. Our DoD stakeholders actually are very much interested in being able to bring some rigor to a context where system of systems architectural decisions are made and the risks that they embody in terms of rework going down the line. That's an area that we will start looking more closely into.

Shane: We will look for that work. Ipek, thank you very much for joining us today. Can you tell us whether there are any publications where our listeners can go for a deeper dive in this research?

Ipek: Sure. It depends on what they are interested in. If they are looking for some high-level introductory kinds of materials that whets their appetite and gives references and also if they're coming from some of our government stakeholders' perspective, we published quite a bit in *Crosstalk*. We have some publications in the May issue in 2012 and also in 2011 as well. So that's one venue that we publish in. We also publish in some research venues like WICSA, and also our Technical Debt Workshop has not only publications by us but other colleagues who are doing work in the same area. One other upcoming resource for people interested in this area is that Robert Nord, Philippe Kruchten and myself are editing a special issue of *IEEE Software* on technical debt, which will be out later in 2012.

Shane: What about SATURN or are any of the proceedings from the conference available to outside people?

Ipek: Oh, that's actually a good question. The presentations of SATURN are on the website and this year almost 50 percent of the presentations did mention their struggle with technical debt and providing an in-house practice for technical debt. One of the biggest issues that everybody

tries to push forward on it is when you know it, make it known so that you can actually take precautions against it. So there's some presentations from other folks.

Shane: Thank you. This podcast and the SEI Podcast Series is also available on CMU's iTunes U website. As always, if you have any questions, please don't hesitate to email us at info@sei.cmu.edu. Thank you for joining us today.