

SEI Podcasts

Conversations in Artificial Intelligence,
Cybersecurity, and Software Engineering

An Introduction to Software Cost Estimation

Featuring Anandi Hira as Interviewed by Bill Nichols

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.

Bill Nichols: Welcome to the SEI Podcast Series. Welcome to the SEI Podcast Series. My name is [Bill Nichols](#), and I am the team lead for the Software Engineering Measurement and Analysis here at the SEI. Today I am joined by Dr. Anandi Hira, a data scientist on our team who is here today to talk about software cost estimation, how it works, and why we need it. Welcome. Dr. Hira.

Anandi Hira: Thank you and feel free to just call me Anandi.

Bill: Well, great. Anandi welcome back to the podcast series. Earlier this year, you and I sat down with Suzanne Miller to discuss [capability-based planning for early-stage software development](#). We will provide a link to that podcast in our transcript. For those members of the audience who didn't catch that episode, can you please tell us a little bit about yourself? What brought you to the SEI, and what do you love about [software estimation and planning](#).

Anandi: Sure. So I am Anandi. Usually I pronounce my name Anandi as well. It is a lot easier for people to remember and say, and that is perfectly fine with me. I came from USC [University of Southern California]. I did a PhD with

[Dr. Barry Boehm](#) in Software Cost Estimation. When I was in undergrad, I had the thought that I wanted to get a PhD. I didn't really know what in, but I started looking at PhD programs near where I would be comfortable going to school in in the area I wanted to be in. I saw Dr. Boehm's program. One thing I really liked about computer science was just the logic and how everything came together and worked together. For example, first you learn programming. Then you learn about how that gets converted to assembly language. Then we learned about computer systems and then [how all of the seven layers work together](#). It was just an expansion on what I enjoyed, which is looking at software kind of in the business realm. From a higher perspective, as far as how software kind of works in that business realm, doing economic analysis of whether something is feasible or has good return on investment. I thought it was really interesting. I applied into Dr. Boehm's program. Here I am years later, doing software cost estimation. I really enjoy being at the SEI. I just started here about two and a half years ago or maybe it is just exactly 2 years ago now. I get to work with amazingly smart people, you included Bill, Chris Miller as well. I get to help people with software cost estimation and planning, measurement, all of these interesting topics.

Bill: You mentioned working for Barry Bahm. I am not sure if everyone in the audience recognizes his contributions or what he did. Can you talk just a little bit about what made Barry Boehm's program unique.

Anandi: Yes, Dr. Barry Boehm had a very unique perspective as far as his background. He himself had come from working in industry primarily, and then came to USC as a professor. So while he was an academic, he was also someone who worked in industry. Something that I noticed that was really unique and impactful about his program and the research that all of his students were doing was that one, it did have academic relevance, but it also had a practical, useful impact on industry, and software cost estimation being one of them. Some of his biggest contributions were the [spiral model](#). He developed the first software cost estimation model, which was published in his [1981 software engineering economics book](#).

He then continued working on it after he started the PhD program at USC. That is [COCOMO II](#), which was published in, I think 2000 or 2001, which is usually called it the Red Book. That was, I believe, named [Software Cost Estimation with COCOMO II](#).

He has also done research in other areas, such as risk management. Some of the later work that he was doing, so some of the students that were with me were looking at the *-ilities* like maintainability, security, reliability and how

they were all interconnected. He just really got interested in I would say everything related to software engineering, and software development in that kind of business realm and addressing all of the topics that managers and organizations would have when they are developing software.

Bill: If I understand you correctly, you are talking about various work. Not necessarily addressing, *Can I build this from a technical standpoint?* but *Can I build this in the sense that do we have enough money? Do we have the right people? How long is this going to take?*

Anandi: Exactly making sure in a way, and this is how I interpret it. I'm sure other people have different perspectives of how they look at it. But looking at the entire environment of, *Do we have an environment that can support and bring success to our endeavor here?* It is really looking at the business. It is looking at the financial aspect, but it is also looking at management aspects. One of the principles in his [spiral model](#), which was then updated to the [incremental commitment spiral model, ICSM](#). If you look at his book, it is a little smaller white book. His first principle is win-win negotiation, which is a management-and-dealing-with-people kind of principle. If you look at his work, it really addresses all aspects related to management and creating an environment that can help your software be a success but also determining whether you do have what is required for success and what you can do to improve it.

Bill: Okay, let's follow up a little bit on that. Tell us a little bit about software estimation. What are the things that you typically estimate, and how do you go about building a model?

Anandi: With software cost estimation, I would say it is much like trying to predict and estimate anything else really. Trying to build a model requires understanding how the environment that it is in works, and what are the parameters that you would use to build the model or that represents the relationships that we see playing out. For example, I think this would be a very relatable example, which is if you had any kind of remodeling work done at your home. Specifically, I am just going to go with flooring. That is basic. If you have ever asked contractors or people, *How much would it cost for me to redo my floor or put in a new floor?* The first question you are usually going to get asked is, *Well, what is the square footage of the area that you want to have your floors replaced?* That is the major cost driver or determiner of how much it is going to cost is how much space or how much flooring is needed. Then there are going to be other factors. For example, what kind of floor do you want? Is it going to be a hardwood floor? Is it going to be tile? Once you figure

out what kind of flooring then there is going to be a spectrum of more expensive options versus less expensive options. Then there are a lot of other factors. The main components of building a model for something like that is determining what is the size metric? What is the major aspect of the job that is going to determine how much this job is going to cost? Something that can, like flooring, tell us how much space do we need to cover, how much work is required. A lot of that is going to be based on how much area needs to be covered because that is going to determine the number of hours it is going to take to, for example, remove your existing floor and then put in a new flooring. Then it is going to go into, *Now what kind of flooring?* So that what kind of flooring and other aspects we would consider kind of parameters or drivers or factors as far as additional components that are needed to get a more accurate estimate. You are going to have the type of flooring, the type of labor that goes into that, the amount of scrap material you may need to be able to get the right shapes and get your flooring to exactly cover your floor. These are all additional factors that would be needed in addition to the size metric to build or develop a model.

The same thing is in software cost estimation. We need to determine what does developing software look like? How can we determine how much time we need to build software? One of the size metrics that has been used since this concept of software cost estimation was developed is [source lines of code or SLOC](#), so that is estimating the number of lines of code that you have to develop. That was the most logical output of the software development process. That is why that is used as the software metric or size metric at least until other size metrics have come up. You can come up with a different determiner for your size metric. It doesn't have to be SLOC.

Then you have factors. In the software development world, the factors are things like what is the complexity of the software that you have to develop. Is the software itself that we are developing easier or harder is going to take less or more time correspondingly. Then you have things like your development environment, so the developers that you have working on them, how experienced they are. Then you have things like the processes that you are using, so are they documenting their assumptions well. More modernly, are they applying Agile or DevOps. Are they being flexible with requirements and having code constantly built and tested, so that they have less to integrate all at once? Basically are they using good practices or best practices for development? Then you have outside impact. For example, if your team is all in the same place versus across the country or across the world that is going to impact your productivity. If you are all in the same place, you can work with each other. You can get answers to questions

quickly versus when you are across the world, you have time zone differences. A lot of these things that can impact how quickly or less quickly we can develop software would be your factors.

Bill: Now you said something that was kind of interesting. You talked about using source lines of code as a basis for predicting the total effort, which makes perfect sense except unlike square footage on your floor, you don't really know the source lines of code until you have built them. How do you come around this problem?

Anandi: Yes, that has been the biggest, I would say, challenge with software cost estimation is trying to determine the size metric for software, because software it is not physical, like flooring and a lot of the things that we do in the real world, I guess, or, hardware type things. It not physical. You don't have a good idea of how many lines of code you need to build something. As time has moved from like the early 1960s where we used to have to develop actual code, we have a repository of existing code. Now we have this culture. We have all of this software, existing software. We have services. We have COTs, which is commercial off the shelf software. You are not necessarily building all of the software that we necessarily have in our finished product. More and more estimating our effort in terms of source lines of code has become a bigger and bigger challenge. Even if we were to develop everything from scratch, it is still a big challenge, but now it just becomes a bigger and bigger challenge. It doesn't really indicate the effort that we do have to put in software development anymore.

There have been software size metrics that have been developed over the years. For example, [function points](#) that was developed in 1979. There have been various variants on the [IFPUG \[International Function Point Users Group\]](#) function points over the years. We have also had use case points that showed up as a way to use use cases as a way to determine our software sizing. I would not call this a software sizing metric. But there is also story points which comes from the agile poker planning exercise, where they tried to use a numbering system to determine how difficult a user story is and how much time it might require to develop that particular user story. There are many other software size metrics that have been developed. I did do a high-level summary of them, and I have an image in [my PhD dissertation](#). That kind of lists, the different software size metrics and how they map to the software lifecycle development model and the things that you know, as far as the phases are. So, for example, you know requirements which of the software size metrics map well to requirements versus architecture versus the other phases. If you are interested, you could take a look that lists some

more software size metrics than what I just now listed. And if you were to look at the existing research and literature, you will find many, many more that have been developed to improve the way that we can estimate our software size and therefore effort.

Bill: That is really interesting. Now, you have talked about the problem of size as one of the more fundamental issues in software cost estimation. What are the various tools. Can you give us examples of the different tools that are used today, and what are some of the problems using them.

Anandi: Yes, so there are a few that are available for people to use either proprietary or open source and not open source, but openly available models. So Dr. Boehm's COCOMO models, that is the one I'm trying to refer to as being openly available. If you have the books, or you are able to find the equations from elsewhere, the model is completely explained in his books. Therefore, anyone can implement it and use the model. There are some other tools that exist as well, that are proprietary based. You would have to get in touch with them. They probably charge according to the type of license that you get, but some of the major ones are [SEER-SEM](#) by Galorath, and [TruePlanning by what is now Unison](#). Now all of these tools do use source lines of code as their primary software sizing method.

COCOMO, they used research that was done as far as trying to convert function points to source lines of code, and then it uses that in their model. In a way COCOMO allows you to use function points. This would be COCOMO II, the updated version, but not directly. Now, TruePlanning and SEER-SEM, both allow for multiple software size metrics, and that can be function points among some of the others, like use case points.

Bill: In [your blog post](#), you stated that you chose not to focus on the components required to support a software system, for example, components, external services supporting hardware. But these are not accounted for in software development estimates. Can you explain why you made the decision not to include them.

Anandi: Yes, well, it is not just I who made the decision. It is everyone that develops these software cost estimating tools. As you may recall, we talked about how we need a way to have a metric, a size metric, to determine the amount of work that is required. That kind of determines by a huge part how much cost and effort it is going to take to do that work. Since we are using typically source lines of code or even function points, for example, to determine the amount of work for software development, that doesn't

necessarily correlate with these other components like supporting hardware. Usually, we have to divide it up because you have different approaches to estimate anything outside of the software development effort. We have to divide it up and use other means to estimate those costs. For example, if we are using existing software services, then you would have the price it would require to get the license for your use. If you have supporting hardware. Then again, it would be the cost of how much hardware that you would need.

Bill: Anandi, what is next for you? When we bring you back here in a few months what are we going to be talking about?

Anandi: Well, I am planning to create a blog series. This [first blog that we have been referring to was just the basics of software cost estimation](#). And what we went over in today's podcast. We did touch a little bit on software size metrics. I plan to do another blog series or blog post that talks about software size metrics in particular, the pros and cons and when you would be able to use them, and things like that, and get more into detail about the issues that revolve around software cost estimation. Then [I plan to] go into some possible solutions and some new ideas on how we might be able to approach software cost estimation.

Bill: Well, Anandi, thank you for talking with us today.

Bill: For the audience. In the transcripts, we will include links to resources mentioned in this podcast including the blog post that Anandi provided earlier this year. Finally, a reminder to the audience that our podcasts are available pretty much any place you can access a podcast including SoundCloud, Spotify, and Apple Podcasts as well as [the SEI's YouTube channel](#). If you like, what you see and here today, please give us a thumbs up thanks again for joining us.

Thanks for joining us, this episode is available where you download podcasts. Including [SoundCloud](#), [TuneIn radio](#), 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 e-mail us at [info@sei.cmu.edu](#). Thank you.