An Interview with Grady Booch

Featuring Grady Booch as Interviewed by Nancy Mead

Nancy Mead: 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 and operated by Carnegie Mellon University. A transcript of today's podcast is posted on the SEI website at sei.cmu.edu/podcasts.

My name is Nancy Mead. I am an SEI Fellow and principal researcher here at the Software Engineering Institute. Today, I am really pleased to be able to introduce you to Grady Booch, who is a pioneer and icon in the software engineering field. Grady is known as the developer of the Unified Modeling Language and also a pioneer in the development of the Rational Systems, for which he was Chief Scientist from its inception in 1981 until it was acquired by IBM in 2003.

Grady is now an IBM Fellow and Chief Scientist for software engineering at IBM. It is a terrific title. He has been involved in the cognitive systems strategy, including work with the Watson project that many of you are familiar with. I could talk about this for a long time, but I would really much rather spend my time talking with Grady. We are privileged to have him here today. Grady, welcome.

Grady Booch: Thank you, Nancy. It is a pleasure to be here.

Nancy: Great. Well, I have got a few questions. One of them goes back to your very early interest in software engineering.

When I was in high school, I used to do all the math problems in my textbooks just for fun, even when they weren't assigned. When I tell my grandchildren this, they look at me and say, *Grandma, how could you do that? Math?* I suspect a lot of us have similar kinds of stories. So how did you get started in the field?

Grady: I remember always being enamored by computers. I remember that I grew up in the '50s, '60s in a small town, Amarillo, Texas. I was a voracious reader. I would go to the library and read all sorts of things.

I remember one book in particular that caught my interest called *Computer Design*. I actually have a later edition sitting in my library now. I pored over it. It was largely hardware in nature. But that led me [to?] say, *I think I'll build my own computer*. So, I was mowing lawns back then and took all my allowance and bought enough stuff to buy a few random transistors, and I built a computer.

I think you would call it more so, these days, a calculator with memory. It had a little bit of if-then in it. It was close to Turing complete, but not quite (I wouldn't have known what that term was), and I was hooked.

Not long thereafter, there was an article in *Life* magazine in which Marvin Minsky was highlighted along with the work at SRI on Shakey. I thought, *This was the coolest thing*. But then I also began to realize, *Software is where it is at*. So, I literally knocked on the door of every company in Amarillo, Texas that I thought had a computer. Of course, none of them were going to hire a 12-year-old something.

They turned me away, but there was a tech guy at the local IBM office that took pity upon me and said, *Grady, come on in*. I showed him my schematics. I think he was confused and amazed, like, *Kid where did you get these kind of things?* Because it was actually pretty cool. Then he handed me a FORTRAN manual and said, *I know you wanted to learn software. Here's a manual. Go read it*. I expected he never would have imagined me to show up again, but I showed up the next Monday saying, *I have written some programs. Can you find me a place where I can run them*?

For the rest of the summer, I would go into our public utilities company, and I had full rein to use their IBM 1130. I learned to punch card myself. I learned to program. I learned to write some programs. I think the very first program I wrote—in fact I still have the deck of cards—was a simulation of two subatomic particles colliding, and I simulated the release of their energy. This is what I did for fun.

Thereafter, I was hooked. I chose to go to the [United States] Air Force Academy. I had a lot of places I could go to, but I chose to go there because it had a great undergraduate computer program. Plus it was a way I felt I could get into the space program. Indeed, I did. My first assignment was at Vandenberg [Air Force Base], where I worked on ground support for the shuttle and MX programs. As they say, the rest is history.

I started getting involved in really large software systems. I was a project engineer for a large telemetry [The Telemetry Integrated Processing System (TIPS) was designed to attend to the new forms of data and higher rates of data coming off the next generation of missiles on the range for which Space Launch Complex (SLC) 6 was being fitted] and a project manager for a

range safety system. That is when I was first exposed, in really my early 20s, to the challenges, the joys of building large, software-reliant systems, and I was hooked.

We did the Silicon Valley thing—I think it is in the water—but two of my classmates said, *Let's start a company*. I said, *OK. That is cool*. This is kind of the equivalent to the Andy Hardy movies, you know, the '50s, *Let's put on a show*. I said, *Let's do it*. Being fearless and not knowing any better, I said *I am in*. As they say, the rest is history.

Nancy: Oh, that is terrific. I know that a lot of your work has been in the architecture area. What do you think about today's understanding of software architecture and patterns by practitioners?

Grady: Great question. *Practitioners* is a big word. I would say, by and large, the concepts of architecture are understood by virtually everyone saying, *This system has an architecture*. But the intentionality of it, the semantics of it, are understood and appreciated by a small number.

I do a lot of work in the Valley [Silicon Valley]. I do a lot of work with small companies that are starting up. And there is an interesting sense of software development that you don't find in, say, a defense system or a big financial system or a medical system. There is really fundamentally different cultures going on. I am incredibly privileged by being able to walk across those cultures because I have no agenda. I just want to build great software, and it is fun to work with people across these.

In the former community of the people building startups, getting VC [venture capital] from various places, there is an understanding that architecture is important. The significant design decisions are important, but this is a domain in which time-to-market is the most important thing. So, a lot of engineering practice is thrown out the window. We hire great people. We make it happen. If we are successful, then we will worry about it.

I often get injected into situations after that fact. If they survived making it past the IPO, or if they are a large organization that is facing tremendous amounts of legacy, that is when the organizations tend to understand, *Maybe we should have engineered some things*. It is at that point in time when a project crosses a threshold for which the economic impact of a lack of architecture begins to attend to them. That is when they start applying more discipline.

Nancy: One of the things that I noticed when I was at IBM, as a software architect, and you came to visit us, which was great…

Grady: I remember that, yes.

Nancy: …was that you very quickly grasped what was going on, what was going well, and where the problems were. How do you do that so quickly when you go into work with an organization?

Grady: I think it is largely because I have seen so many organizations. I have probably worked with close to 1,000 different projects. I have had the chance to visit lots of them.

I think the other thing is I'm a curious geek in that I'm a technical guy, but I think I'm a good listener. I will listen carefully to what people say and what they don't say. From that that has armed me with the techniques where I can step in and pretty quickly assess. As I say to someone, *Sorry, I'll finish up*. Some days I'll pop in and I'll be an über geek. Some days, I'll be Dr. Phil and knock heads.

Nancy: Longer term, you mentioned the problem of people being in an organization that is dysfunctional and not knowing what to do. How do we best go about educating these folks so that, in the long-term, they can do better? Is there hope or…?

Grady: Oh, sure. There is hope. There is hope in the long-term because eventually some of those people will retire and/or die and we will get a new generation, so time is on our side. But there is also immediate hope. I mean, the first thing one has to do is, I think, to listen carefully as to where the points of pain are. When I go into an organization, I will do some quick health measures, not unlike what a doctor will do. There are some key measures I will look at that will give me a sense of the health of the organization.

Then I will turn it around and say, *What keeps you awake at night?* Engaging in a conversation like that—where I talk to the managers, I talk to the customers, I talk to developers—allows me to be able to build a mental picture to the, *If those are the symptoms, I can offer a diagnosis*. So it really depends upon the circumstances.

In some cases, it might be the easy fix to stop them from bleeding is, *We need to build a discipline of continuous integration*. And there's a path that gets one there. On the other hand, you might say, *Oh, these folks have this issue. We need to work up on the requirements process*. We might need to educate them on frameworks, so it truly does depend upon the nature of the disease going on. Again, I am graced with the ability to have seen so many. I have seen lots of symptoms. I have seen lots of great things, lots of bad things. I kind of know which drugs to give them. That sounds really strange for a software person. I am a software druggie, I guess you could say. You probably want to keep that in. Why not. Let's not edit that away.

Nancy: You get high on software.

Grady: I get high on software. Software rocks.

Actually, funny you should mention that. We are working on a documentary, my wife and I. I will look at a piece of software, and I can find awe and beauty in some software. I am taken away by some of it. I had the opportunity to interview Bill Atkinson and his colleagues, the early

days of Macintosh, and they delivered to us the source code to MacPaint. Now there is an example of a beautiful awe-inspiring piece of software. If you want to do some reading, go look at the source code.

And so, I have a visceral reaction to certain systems. There is a beauty in this stuff and that propels me. It is that kind of feeling that I want to open up to the public. We have people that have an incuriosity. They look at their cell phones, and they don't care what happens below the surface. I know that there is an enchanted world behind it, and part of my mission is to open the curtains on that to the general public.

Nancy: I can relate to what you said about elegant software. One time, I worked with Terry Baker at IBM, and he was doing some code development, not for a major project but for a smaller project. When I looked at the work that he had done, it was so elegant. It really was aesthetically pleasing. That is a commonality that I personally see between my early interest in math and what goes on at the higher levels of software engineering.

Grady: Right. The mathematical world really celebrates the notion of elegance and beauty in proofs, in mathematical structures. I think the same kind of thing happens within our software. There is a distinction between elegance and cleverness. I see a lot of developers who do clever things. The problem is there is a technical debt associated with it because your cleverness may not be evident to the person that has to debug your code later. I go for elegance, not cleverness.

Nancy: Very good. I know the work that you did on Rational was very innovative. How did that influence your later thinking, perhaps, when you joined IBM?

Grady: Great question. When I first arrived at IBM, I continued in the software engineering space with Rational. I think that whole experience of not just developing Rational's products, but the many customers we worked with, gave me a foundation for understanding the importance of engineering, the different ways in which software could be architected, the processes there of, the human element of it, and the social element. I think that has been growing and festering for some time.

As I moved into IBM, it was clear that my time horizon of worrying about five to ten years out was a little orthogonal to what the product units wanted. So, that is when I moved over to research, and I continued in the software engineering space, but I found myself drawn into cognitive systems. When Watson was released, I was brought in to say, *Grady, we built this thing. Help us document what we did*. So, I went in and did an architectural dig and documented the architecture of Watson. Then I helped shape our strategy for what we do beyond Watson. That is kind of what I'm doing right now.

In many ways, that experience prepared me for what I'm doing today in that I have a foundation of understanding of processes and development, and the engineering of software, and always an interest in artificial intelligence systems. Those are coming together now in the current work that I am doing.

The exciting thing is there are lots of unknowns. We don't know the right lifecycles for systems that learn. We don't know the right development tools for massive neural networks, networks that are 10 to the 16th, 10 to the 15th neurons. There are still many edges of the engineering of software that remain to be explored. We have so many problems to solve, and it is an exciting time to be doing what we are doing.

Grady: I wish I was 20. I feel like I'm 20.

Nancy: As they say, you wish you were 20, but having had the experience that you have now. That combination.

Changing the topic very slightly, I know that you and your wife, Jan, are working on a project called the story of computing.

Grady: Computing**:** The Human Experience is the title.

Nancy: Tell me about it.

Grady: I am on the board of trustees for the Computer History Museum and John Hollar--who had just been appointed CEO, he had come from PBS and worked at Penguin before—he was responsible for PBS Kids, I think. I had a great conversation with him, and he had just landed a considerable sum from the Gates Foundation. So, laughingly, I said to him, *John, what are you going to do for me next? When are you going to produce a series like Sagan's Cosmos?* He paused and said, *Grady, why don't you be our Carl? And I paused myself and said, I'm no Sagan, but that's an intriguing idea*.

A lot of what I have done in the past, I think, has prepared me for this moment. I am a reasonably articulate geek. I recognize that, to paraphrase Carl Sagan, *We live in a world that is exquisitely dependent upon computing, and yet, most of the world doesn't understand computing*. I feel that the computing industry has given so much to me. I want to give back.

It is our intent to develop a long-form documentary series for public television that explores the intersection of computing and humanity. A few years ago—we have been working on this for about five to six years now—we connected with KQED, the PBS station in the Bay Area.

They supported us to—they actually funded us to develop a teaser, a treatment, and we schlepped this to corporate PBS, briefed it to Beth Hoppe. Her reaction was, *This is exactly the*

kind of series to which PBS aspires. So, for the last year and a half, we have been on a journey to find the production team to make it happen.

If any of the listeners out there have a couple million—actually we need about $6 to $10 million—I would be more than happy to put it to a good cause. Our intent is to deliver a series very much in the spirit of Carl Sagan's Cosmos, exploring the wonder, the beauty, the foundation, the history of computing, and what it means, and basically invite people on a journey to explore that, but also a call to action that we are slowly, irreversibly, inevitably surrendering ourselves to computing. Therefore, we, as individuals, must be intentional about it. We have an opportunity to shape the future and every one of us can participate in it. That is my call to action.

Nancy: Thank you very much, Grady. It has been great to talk to you.

Nancy: Thank you. It is a pleasure. Thanks very much.

Nancy: Thanks again to Grady, our viewers, and our listeners for joining us today. Whenever possible, we will try to include links to the resources that we have referred to in the podcast.

This podcast is available on the SEI website at sei.cmu.edu/podcasts and on Carnegie Mellon University's iTunes U site. If you have any questions, please don't hesitate to email us at info@sei.cmu.edu. Thank you for joining us.