Verifying Distributed Adaptive Real-Time Systems

featuring James Edmondson and Sagar Chaki Interviewed by Suzanne Miller

---

Suzanne Miller: Welcome to the SEI podcast series, a production of Carnegie Mellon University's Software Engineering Institute. The SEI is a federally funded research and development center, sponsored by the Department of Defense and operated by Carnegie Mellon University. Today's podcast is going to be available at the SEI website at sei.cmu.edu/podcasts.

My name is Suzanne Miller. I am a principal researcher here at the SEI. Today, I am very pleased to introduce to you two of my friends and colleagues, James Edmondson and Sagar Chaki. They are going to talk to us today about Distributed Adaptive Real-Time Systems. That is a lot of syllables, and so from now on we are going to talk about it as DART.

So Sagar, James, welcome. Thank you very much for joining us on this exciting topic. Before we get started talking about DART, why don't you each tell us a little bit about yourselves, maybe start with Sagar.

Sagar Chaki: Thanks Suzie. I joined the SEI around 2004. I was actually a grad student, a Ph.D. student, at Carnegie Mellon's School of Computer Science, and then crossed the roads and came here.

My research for my Ph.D. dissertation was on a formal verification technique called model checking. Formal verification is this approach where you have a complex system, like a piece of code, and you want to essentially prove in a mathematical way that it is correct, that it was written correctly and does what it is supposed to do.

Suzanne: And only what it is supposed to do.

Sagar: And only what it is supposed to do. That often ends up being one of the toughest challenges to express what you mean by what it is supposed to do. So anyways, I did my research there. When I joined the SEI, I started working on applying software model checking to proving correctness of code and proving correctness of complex, concurrent systems. We had a number of projects over the years [where we tried] to refine the technique and make it more applicable. We built a number of software model checking tools and put it out there.

Then, I think about four years ago we started looking at this emerging class of systems called cyber-physical systems. We were very lucky to have a group at the SEI, a group of people with the talents to tackle these very hard problems. We had experts in real-time scheduling. We had experts in distributed systems, experts in software verification, et cetera.

Over the last four years we have been slowly making our work more and more attractable, and more and more sort of applicable, trying to do more examples, trying to put our tools out there. DART, which you just referred to, is a two-year project, which started in 2014—we are just wrapping it up—that is a culmination of all this. It tries to bring together a number of different techniques, some of which James will talk about as well, to address this really hard problem of *How do we engineer high-assurance software for a class of complex distributed cyber-physical systems?*

Suzanne: Good. And James, what do you bring into this?

James Edmondson: I did not come from CMU. I came from Vanderbilt University. My Ph.D. dissertation was on distributed artificial intelligence and writing software and middleware for it, and also trying to develop emergent systems as well, like things where you could design very simple agents and make something very complex out of it, or they have, like, an overall mission, that they would not be able to do individually but they do it as a group.

I was very interested in pursuing research in capabilities like distributed artificial intelligence, especially in outdoor robotics, because that is a field that is really, really vague and ambiguous right now and really not a lot of applied science in it.

I came to SEI in, I think, 2012, somewhere around there, and started working on a project called SMASH. It was called Self-governing Mobile Adhocs with Sensors and Handhelds, SMASH was much easier. Essentially what we were trying to do there was have one person be able to control a swarm of things. We started with quadcopters, and we had them doing simple search, like for survivors in wreckage and rubble situations. I think we did a podcast on that before, so I will not talk too much about that one. Essentially, we had a lot of good success with that, and we showed 5-to-10 drones outside or UAVs, I should not say drones, outside just performing simple search.

Then, in recent years, I moved to using boats because they are just a lot easier to work with from FAA regulations and also just with getting them working properly every time. The students also can play bumper cars with them, and nothing gets broken, whereas a quadcopter generally falls from the sky forever.

Suzanne: And they are a little expensive.

James: I think actually what we have been doing has been pretty inexpensive all together. We have been trying to work with really inexpensive platforms, partly because of funding situations, but also just because we think that it has the best impact possibility because if you are doing something with a million dollar UAV or something like that, it is very rare for someone to be able to pick that up, that research, and try it themselves. But if you are using, like a Parrot AR Drone for instance, which is like $300 or something like that, then you have a baseline that is more accessible for students for research facilities and universities. And so with the boats we are also talking about relatively inexpensive things, less than $1000 for each boat. That is important for us because a lot of what I do is trying to push us not only into multi-agent systems—so having more than one self-interested agent helping out with a mission scope like extending the capabilities of one person—but also getting into kind of the swarm area, which is 50 or more things. This is how Sagar and I met and then started working together in MCDA, Model Checking for Distributed Algorithms. This was two years ago, three years ago? I guess it was three years ago.

Sagar: Yes. Before DART.

James: That was the precursor to DART. I was working on getting capabilities outdoors and having these distributed agents cooperating. Sagar was working on formal verification, model checking specifically. We were not working in statistical model checking yet at that time, but we were trying to do a formally proven way of making them collaborate together and be able to check things like safety and correctness and all those kind of things before we actually deploy them.

This started another vector of my research, which was really about trying to bring artificial intelligence from simulation to reality. We are talking about artificial intelligence—not in the case that you would see in science fiction a lot of times where it is a nefarious thing—it is something that you control from the ground up, and you are really just trying to build something that thinks or acts rationally, reasonably, and can complement what you are doing.

Suzanne: Within a specific domain typically. So it is not the general *I, Robot* kind of thing. It is within a very specific domain, there is a set of boundaries the AI entity would work within.

James: Yes. You do not have to worry about Isaac Asimov's rules of governance or any of those kinds of things. This is the kind of artificial intelligence that has a very, very focused mission objective. It is usually very simple agents as well that are modeled as finite state machines, so they have certain kind of rule set that they are iterating over. Then they do that action whenever the environment is appropriate or where their collaboration status is appropriate.

Then what we wanted to do was be able to simulate that at scale. A lot of our things that we scale are between 5 and 30 agents participating. Then we wanted to see if that would transition into the real world, and we have had a lot of really good results for that.

Suzanne: Tell us about what is the scope of DART, the project and what you have accomplished over the last couple of years?

Sagar: As James was saying, we had this precursor for DART called MCDA, model-checking distributed applications, and we started exploring this idea of bringing together research and distributed systems, which is focused on capability, with research in formal verification, which is focused on getting assurance.

What we realized at that time, got really excited about, was this emergence of this class of systems that consisted of agents that are collaborating and cooperating but operating in an uncertain environment to achieve some mission and safety goals.

A classic example is what happens in an Amazon warehouse. I use this all the time because Amazon used to outsource this capability to a company called Kiva Systems. If you go in an Amazon warehouse, there are today no humans. There are robots that take packets from the trucks to the shelves and then back. What they have done is essentially engineered a lot of the hard problems away. Inside the warehouse there are no human beings so there is nothing to mess up your plan, your path.

Communication is perfect. They have WiFi going around. There is perfect localization. Every robot, every millisecond, knows exactly where it is in the warehouse. What we said is, *OK, this is great. It works today. What would it take to take this from that controlled environment to the real world?* So think of …

Suzanne: To my house.

Sagar: To your house or on the highway, right? You have cars. There are a lot of high profile stories about autonomous cars. One of the biggest challenges there is intersection, and people are working on intersection protocols. *Can we get humans out of the loop and try to make cars go through intersections without bumping into each other*?

As we started thinking about this, we came across two classes of properties. First of all, the DART term emerged Distributed Adaptive Real-Time Systems, because things have to happen in real time.

Suzanne: Right, they have to adapt.

Sagar: They have to adapt, and they have to happen in real time. And secondly, we realized there are two classes of properties that these systems must at least implement. One is what we call guaranteed properties, no collision.

The other, the flipside is the mission-critical properties. For example, for cars you need to have a certain amount of throughput, right? A perfectly safe system is where no car moves, no collisions. That is a great way to achieve your guaranteed property, but it is not ideal. It is not useful.

Suzanne: Does not achieve the goal state. I want to get here.

James: And it is not just throughput. It could be also deadline. So if you have ambulances, for instance, going through an intersection, you want to prioritize that. So a lot of intersection protocols have this criticality idea where some traffic going through is more important than others, and you have to prioritize that. That is something we have incorporated into DART as well.

Sagar: That is correct. What we realize is there are two fundamental differences between these. One is the guaranteed properties are absolute; they must always hold. The mission critical properties are probabilistic. You do not want to know, *Can I always have a million cars go through this highway every day*. The answer is *No*. *If there is a big congestion, if there is an earthquake or something happens, no.*

But the issue is, under assumptions about the uncertainty in the environment, what probabilistic guarantee can you make. That is really important from a mission-planner perspective(people who are trying to design these missions) is can I get to that…

Suzanne: *Is the probability high enough? Is it is worth actually engaging in this mission*? is the big question that is always asked.

Sagar: Exactly. In fact, going forward as more interesting work is, *If it is not high enough what can I do to make it high enough?*

Suzanne: A degraded situation.

Sagar: Exactly. These are all interesting problems. From that DART emerged. We basically said, *Let us create a process, an engineering process, by which we can build a software in the right way, starting from the right abstractions and apply the right analysis to make sure that they have all the properties. Finally, demonstrate that on some actual platform to know that what we are getting is actually correct, that you can work with…*

Suzanne: So there is a possibility that some of the capabilities that James has been able to produce would violate some of the properties of a safe algorithm. So this is really where the collaboration comes in, in terms of understanding how the capabilities need to be formulated so that they can be verified formally and proven so that we can take them out into a world that is very uncertain and very unsafe.

Sagar: That is correct.

James: I think one of the things that I mentioned might not be quite what we meant to say. This is not taking the human out of the loop. In a lot of cases it is really extending human capabilities. It is also falling into the human on the loop paradigm where the human, the person, can intervene if things go badly. In the Amazon warehouse there are people there; they are just not on the floor. They are supervising the behavior of this, they are making sure everything is done correctly, that orders are being shipped correctly, and that nothing is broken.

This is the same thing for traffic. In the United States we do not expect to have a fully autonomous driving fleet. Not everyone is going to do this. *We are Americans. This is my car, I am going to drive it*, that kind of thing.

If you are building that kind of thing, an intersection protocol, you have to take into account that humans are a part of this. There is variability. It is not going to be perfect. Everything you have planned for is going to be ruined at some point and you have to be able to adapt to that, and that is part of the adaptive part of DART is being able to adapt to the environment. So Gabe Moreno's work where it is adapting to danger, and things of that nature, comes into account.

Suzanne: So the guaranteed behaviors are the ones that are mostly about safety, security, those kinds of things, and then everything else is probabilistic. If something is happening in the environment that would cause you to violate one of the guaranteed properties, you have to adapt. That is the real-time adaptation.

James: The other thing to understand is a lot of this is a lot of the verification happens in kind of a piecemeal manner where you are in perfect conditions this is the kind of adaptation or this kind of guarantees you can give. Then you have a different scenario you test like if there is an enemy present, or, for instance for adversaries when you have an adversarial AI, like you would change your entire behavior for that.

Suzanne: Or a human that is present in the intersection.

James: Which is kind of adversarial. A person thwarting your best efforts, your optimal efforts is kind of adversarial. That is how AI kind of treats adversaries. It is not necessarily a nefarious

person, it is just a person who is kind of ruining your good day. That kind of thing is how adversaries…

Sagar: So humans actually…

Suzanne: They have a different goal.

Sagar: Humans actually play two roles. Right? Like you said they can be a supervisory role, but there are also adversary roles.

James: They can be both.

Sagar: They are both. From the DART perspective, even though we have not specifically looked at human behavior, sort of the way we would bring in the human is another uncertainty, another person that is providing input to the software essentially. But we have not actually gone that far. Right? I think it would be really interesting to see how might work out.

Part of the challenge there is [that] we do not have very good models of human beings. We cannot treat a human being as completely non-deterministic because none of us are completely random. We have rationality, yet we do make mistakes. We have to model humans in the right amount so that the analysis we produce is the right kind of results. They are not too pessimistic, not too optimistic.

James: I would say even smart people act irrationally sometimes. When I have had quadcopters flying in the hallways at SEI, I have had Ph.Ds walk up to them and put their hand on the blades. I could have never predicted that before that happened. But I mean, even smart people sometimes do dumb things.

Even when you are thinking of rational actors—and this is something that is a very core concept of AI in general—is that rationality is very relative. What is really interesting is some of the work that we have done in DART looks at parts of your algorithm or parts of your system that you would not have anticipated.

When we talk about a statistical model checking for SWARMs is another one that might have a blog sometime. One of the key things, one of the most interesting results of that work, I think, was the input attribution and the ability to essentially look at this algorithm, and just from the variables that are being implemented, the environment description, essentially, it can tell you if the mission would fail and why. For example, we had a defensive scheme, a paparazzi scheme where we were trying to block a photographer from getting a celebrity photograph, and I think drones are getting in the way.

They are just occluding the photograph. We were doing analysis on that, and we saw that it is really only two agents. We were adding more and more agents to it. We thought, *Well let's just put a swarm of drones out there in the simulation, 100 drones, and see if it makes any difference.*

There was no difference between doing 100 drones versus 200. The reason for that was the algorithm itself was setting the first agent on line of sight between the photographer and the VIP. And the next one is where it might be going. So basically the one that is currently line of sight and the one that is next to it, where it would block the next line of sight, are the most important things. So how fast they get into position and how they move, and how fast they move, is the most important thing. We saw that just from looking at the input attribution.

Susan: So one of the things these models are doing is helping you to identify non-intuitive conditions that either need to be accounted for or do not need to be accounted for. *I do not need 100 drones if I can put them in the right place.*

Sagar: Exactly.

James: Like I said, from my dissertation research I was very interested in emergent behavior. This is kind of the definition of emergent behavior, where you have—usually it is emergent misbehavior. You really do not get upset about emergent good behaviors. Right? If all of a sudden you get a million dollars for doing your job and it was not expected, that is a great thing.

What most people complain about or talk about is emergent misbehavior, where you design a system for something specific, and then out of all of that came, [something] like thrashing or something else where nothing worked over time, and you do not know why it is.

If you have a technique, which we think we have a good starting block for, that that can tell you what can cause thrashing or what can cause this kind of bad emergent behavior before you even deploy it. That can help you solve the problem before you start getting in trouble with clients and with your customers or with loss of life even in these cases. Because these are cyber physical systems working in conjunction with people, that not only brings a monetary value to it, but it brings a life value to it with automation.

Sagar: If we stay back a little bit and look at the DART process, some key ideas have emerged. There are three main aspects of DART, and DART actually brings together two paradigms that have been there for a long time in software engineering and in sort of formal analysis.

One is a model-based development. You start with a high level description that is well defined in the language, where semantics and syntax are well defined. The other is analysis, that you basically do analysis at each step of the compilation process from that high-level language down to binaries. You see this for example in a compiler. When you compile code, if you are writing in

whatever language, the language is well defined. Then the compiler actually does key analysis and gives you warmings. That rules out a lot of the errors that happen in the program. It does not rule out everything, but it rules out a lot of the errors.

In DART we have a very similar compilation process. We have a domain specific language called DMPL; it is another acronym for DART Modeling and Programing Language.

James: Recursive acronyms. There is DART, and there is DMPL.

Sagar: We use DMPL to program the DART system. We actually combined DMPL with another language called AADL. I am sure you have had other…

Suzanne: Architecture [Analysis] and Design Language.

Sagar: Exactly. They work complementarily. AADL gives you the high-level architecture of the system, DMPL gives you the low-level code. The code actually happens in the protocols in the guts of the system. Once you have that description of the DART system, you can then start doing analysis.

We do three main kinds of analysis. One is real-time scheduling. We have to make sure that all the threads have enough CPU cycles to run. Because otherwise all the other analyses are meaningless in cyber-physical systems. If your code does not run…

Suzanne: You are late. You are gone.

James: In some cases. We have some concepts of criticality, like we were saying, that essentially there are certain things you might be able to preempt that are not as important as others, but there are some that are mission critical that you cannot. It has to be schedulable. Then there are some that fall off the wayside, and that is okay. That is very common in a lot of UAV applications where you have movement, and things of that nature like flight for instance, which is something you definitely have to schedule. It has to be schedulable.

Then there is what is run on top of it, like analysis or something else, where it is taking sensor inputs, and it is trying to figure out was there a person there or something like that. That is usually important for search and rescue, but it is not as important as keeping it airborne.

A lot of this happens very naturally. If you have a UAV that has a lot of different sensors in it, for instance, a lot of video feeds even, and you are running something like open CV on it and you are trying to analyze the images in real time, then you have a really high computation overhead if you, for instance, noticed three objects. So, for example, if you pick out *there is three objects that has to run on there*. Before that it had no objects, so it is fine. The flight was working fine, the controller was working fine, I had plenty of CPU, and then it went over these

three objects and all that started kicking up, and that is when the scheduling problems started happening. So that is what we have to deal with is the mix criticality of this situation.

Sagar: And that maps very naturally to the properties that we were talking about. Remember we had the safety critical properties and the mission critical properties. In general the guaranteed properties are implemented by high critical threads, and the mission critical properties or the probabilistic properties are implemented by low critical threads.

So when push comes to shove you kind of can go degraded on the low critical properties, and you can still perform but you have to preserve the safety properties, the guaranteed properties. So that is one part, the timing.

Then comes the software model checking. This is proving that given though the threads have enough CPU cycles to run, they are doing the right thing. This is where we analyze the collision avoidance protocols. James and I did a lot of collision avoidance protocols as part of our original MCDA project. We have been just using variants of that protocol in DART, and it has worked very well.

We now have software model checking algorithms that can prove that the protocol is correct, not just for two drones or three drones, but for an arbitrary number of drones in an arbitrarily large area. So that was really exciting for us is to be able to make that contribution. We can only do that because the language, the DMPL language in which the protocol was expressed, was restricted. It was rich enough to express the protocol, but we cannot restrict it enough that it would analyze it.

As you know, in verification we often have the state space explosion problem. So if the language you are trying to analyze like C or C ++ is very complicated, it is very hard to prove anything formally. So in DMPL we kind of had this right balance between the two, the expressibility and the verifiability.

The third thing is statistical model checking. This is what James has been referring to quite a bit. This is for analyzing the probabilistic properties. So now you have this complex system, *how do you estimate the probability of finding all the objects on the ground given certain probability instruments and given some mission parameters?* That ultimately relies on lots of Monte Carlo simulations. So it is a statistical approach, so it does lots of Monte Carlo simulations. It's a statistical average, so it does lots of Monte Carlo simulations. But from those simulations using statistical hypothesis testing kind of results, it can give you an estimate of the thing you are looking for, the property you are looking for, and an error bound, so that was the important part. Because simulation is ultimately incomplete, so all of our estimates are error bounds.

But the good thing is we can do enough experiments, enough simulations to make that error bound as small as we want. So I can tell you, for example, that *Oh, the probability of completing this mission and detecting at least 80 percent of the objects is a .9 with a plus/minus .05 error bound.*

Naturally what happens is, if you want to get the error bound small, if you want a precise answer, you have to do more simulations. But we have been able to do research in techniques called importance sampling, for example, that essentially will allow us to estimate real events, low probability events, with fewer simulations.

Suzanne: You have incorporated the black swan idea, the very low probability but high impact.

Sagar: Exactly. The black swan idea. So the idea is we can use important sampling to estimate the probability of black swan kind of events but using fewer simulations. There are a lot of technical details in how this works and then, we have publications.

James: There is another pillar of this that glues everything together. So you can run all the simulations you want. You can run a billion of them if you wanted to get your error bound down low. But if that system is not implementable in the real world, if there is nothing that actually…

You have got this high level modeling language, DMPL. It can go through this process and generate, but if it does not work in the real world, then you basically have a lot of proven nothing. And so, kind of where I come in into a lot of this is with the middlewares that I have been developing. I do not know if you want me to spell out these, MADARA and GAMS, but essentially what they do is they bring in a quality of service to everything that happens within the system. So everything from threading, like the computation assigned to threading, the EPoX that are available for that, the communication itself, *How does communication work? How does messaging work? When you receive something what does it mean? Is it an all or nothing processes? Is there a partial?*

We had to have very specific semantics for every aspect of how the distributed system might work and that is the only way we can really do verification. You cannot just use an arbitrary …

Suzanne: You have got to set a boundary on the state spaces that you were talking about earlier or else you are done.

James: You also just have to have a consistent view of the world. So part of this was designing these middlewares to be more verifiable. A lot of this process was designing quality of service that could be implemented in a way that gives more confidence in the distributed system, not only from scheduling but from messaging and threading and all kinds of other things. That glues everything together so that we have this transition path from this formal language, this DMPL

language, that takes distributed processes and then guides it all the way through to the real world and real robotics.

Suzanne: So this idea of multiple layers of analysis, but analysis that focuses on different things, that is something different than what we have seen in terms of a process that goes from model all the way out to implementation, so that is very exciting. Is that what you would consider to be sort of the major achievement of DART is to be able to prove that that is a feasible thing to do?

Sagar: Yes. I think that is definitely one of the major accomplishments is *How do we bring these analyses in a semantically coherent way*? and then, *How do we actually tie them together in an implementable system*? Like James was saying, when we were designing DMPL…By the way, I cannot over emphasize the importance of the MCDA, the project, the one that preceded DART.

It turns out that DMPL language is also at play. In MCDA, we had a language called DASL. DMPL was essentially kind of a child of DASL; it was many of the concepts we got. Like, *What should be the semantics of communications between two distributed agents?* It is actually a non-trivial problem. Especially if you want to say *This has to be implementable. It cannot just be something that is nice to model, nice to analyze*. Unless we can actually implement this using our middleware, there is no point in doing this analysis. Then our analysis is all happening on abstract systems that have no real counterparts.

So that was a no, no for us. Right from the start we had this idea that our protocols and our semantics had to be implementable. That is where James and the rest of the team had a complete collaboration, because every decision we made about, *Oh, should it be shared memory? Well what does shared memory mean in a distributed system? Because you have no central place, it is not like one processor. Well, so there is consistency. Well, what kind of consistency is it?* So James would come in and say, *Ah, we have a consistency based on Lamport clocks where you have certain well defined notions that earlier values cannot be seen after later values, et cetera, et cetera. According to some logical notion of time. We do not have clock synchronization and so on and so forth.*

That was another, I think, big part of the project. So when I talk about DART, I have a pie chart kind of a slide on DART that kind of brings together all the different areas where people are to contribute. The fact that all of them play together nicely to have a system where we can have demonstrable code that goes down to binary and ZOOP! We can actually see things running, I think that is amazing. I know of no other project there that can do that level of integration of different techniques in a demonstrable way.

James: It was also kind of a nexus of just the kind of critical mass of researchers that were specialized in these fields coming together that probably are not available anywhere else in the

world. We had formal methods people. We had real time scheduling people. We had middleware and artificial intelligence people, and all of them were put together in the same place at the same time, which is kind of fortuitous, because Sagar and I are pretty smart people, but by ourselves we would not have been able to accomplish everything we did in DART.

Not only are the capabilities there, but this has been made open source. So DART is open source, MADARA, GAMS are open source. MCDA was open source as well. This is something that we have been able to do pretty consistently with all these projects. This means that people can use the research, not just read about it on paper but they can actually download it and do this.

We have actually given some tutorials recently to the Air Force Research Labs that showed them how to build a distributed system from the ground up. Some of them had never made a distributed system ever or even programmed before. Within an hour they were making a swarm of UAVs that accomplished a mission, which is pretty cool to see in action.

Suzanne: Well, and that is, I think, one of the things that draws me to what you are doing is that idea that we are going all the way to implementation. You can see on the water, in the air, the result. As somebody just out in the world building confidence into these devices is huge. To be able to get adoption of any of these human extending capabilities.

In military and certain other settings, you can mandate *You will use this*. But you cannot tell me, U.S. citizen, Suzie Miller, that I must have an autonomous vehicle right now. Now, maybe someday in the future, but I have that choice. If I am going to make that choice I have to have confidence in what I am using. These kinds of techniques are critical for us to build the confidence of the whole population that these are safe kinds of cyber-physical systems to use to enhance our lives.

James: You always see the bad headlines, you do not see anything really good about autonomy in the news. I mean you had the recent Tesla crashes when they were on autonomous mode. This became news because people are worried about the safety of these systems. Also we are trying to prove that this was possible, right, so we are at the very beginning of autonomy being pushed into the consumer space and having people able to do this. It is not just in industrial settings where these big multi-million dollar robots are doing some complex task but they are doing something that is very small.

We are having machines and autonomy being pushed into the real world and interacting with human beings in a way that is beneficial almost always but also can be dangerous. You need an approach like this that gives you safety assurances about this so that people can start to trust the autonomy and be able to use them and be able to be more effective. Because that is what the point of autonomy is. It is not replacing people or anything. It is making people hundreds of

times more effective in some cases, which we think will be beneficial for the economy, for the DoD in various ways, but really for people. I mean, for humanity.

Sagar: Often when you talk about confidence, there are two kinds of things. You see a lot of academic publication happening in every conference about verification, about cyber-physical systems. But I think to a common person who is a consumer of these technologies, that does not give confidence, because often that is esoteric academic publication.

How many people are actually going to read and follow this technology? They might hear sound bites. What gives confidence is when they actually see the thing happening in practice. So when they actually see Google driving around these cars and Google publishes some statistics or Tesla publishes some statistics that *Oh, we ran a million miles and, look, there were no accidents, or We ran a million miles and had only two accidents, et cetera, that gives us confidence*. And I think with DART, and particularly the research we are doing, we are trying to bring together both those ideas. Because we need mathematics, we need objective analysis. A million miles is nothing. A billion miles is nothing. If you calculate the number of miles driven by U.S. drivers in a year it is like trillions of miles. So the fact that you drove a million miles, that should not really give you a confidence.

What should give you confidence is that together with the mathematics, the analysis, the exhaustiveness of the nature. To me, what has been missing is something that is bringing together both. That is because the way we are set up, we have academic researchers working on the publication side, and then they kind of stop, and they expect the technology to somehow jump to the practice. But the practitioners, they do not really have time. You see this even in universities. You have groups that are heavily experiment focused, right? All they do is build things and they run but they do not really do a formal analysis. Then they have trouble in the demos and stuff. And then you have the formal people who are doing this mathematics and this and that, but they do not really talk to each other.

Suzanne: They do not get to a demo.

Sagar: They do not get to a demo. I think what we are doing and what the SEI's unique position is to be this bridge, is to be the connection between X here and Y there, and also add our own expertise. I think we are in a unique situation.

Right now, James has already started this process. His research has been heavily focused on demonstrating his techniques on real hardware platforms. We are taking that cue, and we are continuing that, so we are building a…

Suzanne: A cyber physical lab

Sagar: A CPS lab inside the SEI Library. Please come by and see what is happening there. But the idea would be to continue that thing and then we would be able to demonstrate our research on other platforms, especially things that fly around. We are trying to do this because we can do it indoors and we do not have to rely on the CPS. That is a unique role that the SEI could place to bring together ideas, transition ideas, from academia to practice.

Suzanne: That is our mission.

Sagar: That is our mission. And I think CPS is one of the best domains to be able to do that because you have software with the physical world. You have the intangible with the tangible, and you do not see that in many other domains.

James: By the way, we kind of started using some terms I do not think we defined earlier. CPS is cyber-physical systems, and cyber-physical systems are a lot of what we have been talking about. It is like the marriage of hardware and software, and the ability to control and do that. It is not like a cyber warfare thing; it is not even like this. We are talking about very specifically hardware platforms that you are controlling with software and you are making it intelligent, generally. So that is what CPS is.

Sagar: It used to be called embedded, real-time embedded systems and it got morphed into that.

Suzanne: We have to have a new term every 10 years, whether we need it or not.

James: Well, when Congress stops doing the funding, that is when they tend to come up with a new buzzword.

Suzanne: That has been known to happen. So what is next? You finished this phase of the project. I am sure you have ideas on what you want to do next. What do you have in your plans for the near term?

Sagar: One aspect of this is another project that we are just starting. It is called Certifiable Distributed Runtime Assurance, and it is not as cool as DART. It is CDRA. But the idea is actually really cool. The idea is you are going to have complex systems that are going to be impossible to formally verify. You have a machine-learning component. I think of it as almost a car. It is going to have components, and they are going to do extremely advanced computation, and they are extremely capable but fundamentally unpredictable.

Most of the time, it will give you tremendous capability, like your Google Maps, right? Almost all the time it gives you this nice way to get you home, but then, in the back of your mind, you have this nagging suspicion. *Oh, what if it is taking me through an area I do not know*? That kind of thing.

We have to live with that power, with the unpredictability. The way we can do that is to put a monitor, a guardian, as a trusted part in software and in the system that is going to watch over this powerful but unpredictable rest of the system and stop it from doing really bad things.

This idea actually emerged at the SEI. It was called Simplex. Lui Sha, did this and a bunch of other researchers did it, but in the control domain. So you had a powerful controller, and you had a trusted controller. If the pendulum was falling down, then the inverted pendulum was falling down, but the monitor would switch it back to the trusted controller. There is a lot of analysis about when to switch, how often to monitor, et cetera.

Then what happened was recently the Air Force Research Lab, AFRL, picked up this idea and they called it runtime assurance. They enlarged this idea to not just control, but to any kind of system. You basically have to put the monitor in the right place and to make sure that it stops the software from sending bad commands across what I call the cyber-physical boundary.

The software can do whatever it wants to do, but at some point, it is going to actually actuate. It is going to send the command and say *Now, affect the physical world*. *They move this arm. Open this door. Drop something*.

What the monitor does is make sure that those actions do not happen under very specific circumstances. Our research here is actually *How do we build a trusted monitor? How do we verify that monitor?* The good news is the monitor can be simple. The rest of the system can be very complex, will be very complex, and will be unverifiable, but the monitor can be engineered to be really simple. We still do all the verification, and it has to be performed. It cannot monitor too many things. If you put too many checks, and the rest of the system comes to a halt. So there is this balance between verifiability and how much you interfere.

Finally, we have to put the monitor from a security perspective to make it uncircumventable. So [if] somebody hacks into your system and tries to disable the monitor, that should be impossible or as close to impossible as possible. We have a security angle in this research where we have an attacker model, what can attacker do, and then under the attacker model, we should be able to prove that monitor cannot be taken over, or if it is taken over, before it gets taken over, it will do some default action like bring the system down to a safe state, et cetera. So this is really exciting.

Suzanne: You have got your work cut out for you on that.

Sagar: It is part of the reason we want to demonstrate. So actually one of the examples we have for this is what we call a virtual tether. Remember in the beginning of the program, we talked about how these drones, they will often jump to the ceiling. Well, one of our colleagues, Scott Hissam, said, *Well, I am going to bring a fish line. I am going to tether that thing so it does not jump up.*

Then we thought, *Well, what if we can use a monitor as a virtual tether? Suppose we can actually have good localization, and we can put a monitor and say no matter what happens. So if I give you the RC controller, and you try to push the module out through the ceiling, the system, the monitor is going to watch and above a certain height, will cut it off.*

One of our goals actually is to do a demonstration of this and to say *Here in the library, here is sort of the virtual area within which it cannot escape this area. Try your best to do it.* We are going to give you the RC controller and try to do it.

Suzanne: Maybe we'll have to come and film that when you get a little farther along.

Sagar: Absolutely.

James: What he is talking about, the monitor, is a little more I guess invasive than you might think. It is not just intercepting human commands to tell whether it goes left or right, up, down, whatever like that. It is actually at the flight controller level. I think before, there was no taping going on, when there was no taping going on, we were talking about the quads flying into the ceiling or flying anywhere they felt like going. But essentially, when I was doing quad copter research, I did actually when I was outdoors have some situations where I would tether them with fishing line to stop them from going wherever. And that was to keep us safe and also keep the drone safe, UAV safe. To actually stop the flight controller—because it was not a person causing it to fly out into nowhere— the flight controller itself sometimes is just wrong. It is implemented. It has bugs in it, various other things, and so a part of what we have to do in that project is to invasively put monitors inside the flight controller to say *The flight controller might be pushing this out, not just the user*.

Suzanne: And when and where and why and justifying that is going to be one of the research questions.

Sagar: That is correct.

James: I think from my interest, from this one especially—I do not think we are addressing this this year or this project—but essentially, there are a couple of things about DART that are limiting as far as what kind of systems you can build.

One of them is asynchrony. We tend to force a synchronous model of computation on the distributed agents in DART. What that means is everything has to barrier together when they are doing a mission. So if they are moving together, they are kind of waiting for each other to come to a certain mission state, and then they move to the next mission state together. Or, they essentially can move asynchronously to the next mission state, but there is still that barrier on the mission state, on where they are going. This is very limiting for a lot of real time applications,

especially outdoor robotics, because things lose communication for a long period of time in some cases. We have to figure out how to make things more asynchronous and along the gradient from synchronous to quasi-asynchronous to fully asynchronous.

That is still an open problem and something that I do not know if we even have something slated this year to look into that. But it is something that we knew in DART that was going to be an issue with making all systems verifiable. And it is one of the reasons we started looking into statistical model, checking other things like that, so that we could say, *Look. It is going to be deviating from this perfect mission state. We just need to handle it in some other alternative way.*

Suzanne: It is good to know that you have work ahead of you. You are not going to be finished any time soon.

James: The other thing is, so when we are talking about this monitor, he is talking about a simple monitor and he is talking about making it not too complex that it would interfere with his cell phone. Also, there is this Turing level problem, which is essentially if you make these simple monitors and you make five of them, let us say, on a UAV, how do you compose them together to have this kind of verifiable monitor between five different things? That is like Ed Clark and Joseph Sifakis plus plus. This is, like, higher level verification that has not been done before.

Sagar: So that is on our trajectory, probably is like you said not going to be until second year of the research.

Suzanne: One step at time.

Sagar: One step at a time. I think there is a big role for sort of middleware and the AI technology to play there because there is no reason that we cannot engineer the middleware to actually help make the monitoring simpler or make the monitoring more tractable. That to me is one of the nice things, technically, when it is balanced between verifiability and capability. And that is, I think…

Suzanne: That is the crux. So I know that there are some folks out there that are probably going where do I get my hands on all this stuff? You have got—you said you have open source, yay. But what are some key resources for software architects, cyber-physical system developers that they should look at in getting educated about DART and what its boundaries are as well as what is been accomplished?

Sagar: The DART system is open source. It is publicly available on GitHub. I think we can make a link to that available.

Suzanne: We will make that available.

Sagar: You have the full source code. You have tutorials. You have instructions to download and compile. You have various videos showing what kind of simulations you can run. We have actually tested it extensively within the group and across some known friends, and it has worked very well.

It has largely supported the Linux platform, but it has worked very well on Linux. DART actually relies on the MADARA and GAMS packages that James develops. I think James, you are probably the better person to talk about how those are available.

James: We will obviously provide the URLs for the MADARA project and the GAMS project, and how they interact is described in a lot of different Wiki pages and PowerPoints on there. I will try to make sure you guys have a link to the tutorial series we gave at AFRL because I have all of the slide decks we used for giving that tutorial on the GAMS site. That walks people through the installation of it, which we have tried to make into just a one-line command thing that you can do on Windows or Linux.

MADARA and GAMS should work on most platforms: Apple iOS, Android. We have had Android running on the boats before and basically every operating system I can think of. Essentially, if ACE, which is the Adaptive Communication Environment from Doug Schmidt over at Vanderbilt, which has been around for 25, 30 years, if it can run with that, so even on old Citrix servers, basically, you can run that if you wanted to, Solaris, whatever you want to do. But essentially, it runs on most platforms, hundreds or maybe even thousands of platforms, ARM and Intel, things of that nature.

All of this is described on the Wiki pages. We will provide some of the links for you. And essentially, if they just follow the tutorial, the first slide will show them how to install and then they can go to the specific tutorial they want to build algorithms. For instance, if you want to learn how to do distributed algorithms, you can do that. We also have some papers. I do not know if could provide papers.

Suzanne: All those kind of resources, we will give you links to the digital library and all that.

Sagar: The DART website, I think, actually has also a publication space where we talk about some of the papers.

Suzanne: Excellent. Sagar, James, as always, it is fascinating to talk with you. I love being able to catch up and sort of find out where you have gone ever since you have both come here and talked about some of your initial research. I am really thrilled to see you being able to work together and bring together some of these great ideas. So thank you so much for sharing this.

We have a blog post on this topic also. That will be available at insights.sei.cmu.edu. That is where we have all our blog posts. James, I think you are the lead author on that. So James Edmondson is the name to look for. We will provide resource links throughout the transcript to the things that you have talked about.

The podcast itself is available on the SEI website, as I said at the beginning, at sei.cmu.edu/podcasts. It is also available on the Carnegie Mellon University's iTunes U site and on the SEI's YouTube channel. Once again, thank you for viewing and thank you, again, for sharing all of this wonderful information.

As always, if you have any questions, please don't hesitate to email us at info@sei.cmu.edu. Thank you.

Suzanne: Thanks for having us.

Suzanne: Thanks.