Danielle (00:14):

Kristopher Miller is a systems engineer and a software developer. They receive their bachelor's in aerospace engineering from Missouri University of Science and Technology. Kristopher is currently living in Huntsville, Alabama, and working for Jacob's Technology. Jacob's Technology is contracted by the NASA, Marshall Space Flight Center. And today Kristopher Miller will talk about Artemis advanced real time environment for modeling integration and simulation.

Kristopher (00:47):

Hi, thank you. So to also give a little background information about myself, like she said, I'm a software developer or I'm aerospace engineer doing systems engineering for Artemis. So I'm a part of the ES 53, which is engineering systems branch here at NASA Marshall space flight center. I've been here for just under three years and that's after graduating and getting my bachelor's degree. So in this presentation, I'm going to kind of go over generally what Artemis is, what it does and kind of how it's built. And then that'll also kind of give reference to what this is and what we use it for in the support of Artemis mission, which is the SLS rocket.

Kristopher (01:42):

So the first thing is what is Artemis? So Artemis is an acronym that stands for advanced real time environment for modeling integration simulation. So give more information about that. That's real time hardware in the loop, environment hosting simulations, and then which stimulate hardware under test. So the system and simulations in this case are aerospace vehicles or robotic systems. So for our specific case right now, this is SLS. And then we have the hardware, which would be the avionics components and the sensors, the effectors, anything of that nature on SLS. And then overall Artemis is reconfigurable for vehicle design hardware, software interfaces, hardware under test, and then laboratory computer resources.

Kristopher (02:27):

So to give a little more information about Artemis, you can kind of think of Artemis like a video game, more specifically, a flight simulator. But instead of a user or a player playing Artemis, the actual player is SLS itself. Specifically the flight software and the avionics components in SLS. So we are going to use Artemis to make SLS think that it's actually flying. So the real time aspect of this is that everything or the test that we're performing on the actual qualification and verification on the hardware is real physical time.

Kristopher (03:12):

So an example of that would be, we have a scenario that is a test of SLS over its entire launch day. So from the very, very beginning, the very first thing that they would do on launch day to the insert or orbit insertion of the crew module into orbit. So that if you know, takes roughly 35 hours. So this scenario, this test would actually take us 35 hours to run. So to continue on, we have a bunch of requirements for this just to make sure that it's as accurate as possible.

Kristopher (03:53):

So the first thing would be model vehicles with high fidelity in real time. So that's all the avionics components. That's all the subsystems that affect the vehicle in any way, that's even the environmental effects. So we opt to model the gravity, the air, the aerodynamics of the vehicle. And then we also do nonlinear six degrees of freedom vehicle equations of motion, including variable, mass staging, and flex.

Kristopher (04:21):

So variable mass, I mean your mass, if you're losing mass, as you're flying, you're also losing mass based on say the boosters falling off, or eventually you have panels that fall off or the entire core stage. Then we also have flexible body. So a rocket's really, really tall. And you also have rather large engines at the bottom of rocket. So if the rocket's moving those engines in any way, there's going to be a minor or very small degree change in the bending of the whole entire rocket. So we even have to model that.

Kristopher (04:58):

So to emphasize part of what I mentioned a little bit ago about the entire process of launch, or just the simulation itself, we have to model all phases of the mission. So in this case, prelaunch pads-op through orbit insertion. Next we have to support multiple labs and configurations, which I will talk about more in the next few slides. And then we support multiple configurations and scenarios of each lab. So we have hardware that we have to use, but before that, we have to use software that we can easily switch out. And then we have data files that can be modified with scenarios, and then those specific items to be overwritten and an initialized models for proper different or for specific tests.

Kristopher (05:44):

So to continue on with some of these requirements, we have to execute in multiple modes of operations. So the first thing is a non real time, all simulated buses. So this is something that could run on a laptop or on your desktop. The next one is an upgrade to this because we're using what is a realtime machine, which is more specifically realtime computer. So it's using a module card that is able to keep track of what and run real time simulations, which is for reference, say you have a computer or a screen that's running at 60 hertz. Our real time physics has to run at 400 hertz. So that's a very, very fast processing.

Kristopher (06:32):

So on top of that, we still have those all simulated buses. So what that means is you have a bunch of data cables. You have a bunch of bus connections between all the different components on any computer or sensors or vectors, any kind, they all have latencies and other issues or not issues, but resistances and such based on how long they can get. So we actually have to model that. We have to implement that potential change to the data, or just timing in the simulation.

Kristopher (07:06):

And then the last one is the real-time, real buses or hardware. So that's everything that we can have real is real. So then the last two things is that we have to interact with the lab configuration and control software. So this is the model configuration selection, generation of the scenarios, and then real time data viewing, fault insertion. And the last thing is the data recording and archiving, which just basically means that we have to record all of the data that we get or make.

Kristopher (07:39):

So I mentioned that we have to support a bunch of different labs and configurations. So this is specifically our customers. So before I get into these, you see on the right side of some of these bullets, I have parentheses and some numbers. These are buildings located here on Marshall. So that first one is the software development facility. So this is specifically to tests and develop the flight software of SLS.

Kristopher (08:03):

The next would be the software integration test facility. So this is specifically the core stage avionics test. And it's the formal qualification for that. SILL or the system integration lab is the core stage and booster avionics. So we have both the actual hardware for the core stage and boosters. The TDL is the SDF SITF configurations, and this is a digital version of those different labs that we use. And then RINU 6DOF test is the RINU avionics on a six degree of freedoms platform. So that's a platform that moves in all six degrees of freedom.

Kristopher (08:43):

Then we have our simulators. So the first one is the ground systems development and operations. So that's at Kennedy Space Center. So this is SHADE. So this is going to be used by the mission control people for the launch of SLS. So they actually use this to practice and prepare themselves for when SLS launches. We have a multi-purpose crew vehicle, which is at Johnson and Lockheed Martin Corporation. So this is Orion and this is everything, all of the components, all of everything that we would need for the crew model.

Kristopher (09:17):

Then we have green run. So if you've been listening to NASA news lately, we actually had the real green run test recently. So we actually modeled that as the SLS and that test stand that it would be on. And we modeled that, simulated that to show essentially just how it would run for this actual test to happen. And then we have the last thing, which is the booster hardware-in-the loop, which is just the booster avionics interfaces to the booster subsystems.

Kristopher (09:50):

So moving on. So we have the simulation design now. So this is a single copy of source code for all simulation configurations, which basically means that we're not going to have anything copied. We're going to have one version of everything. And this is also to have to input different initial conditions and configurations through an XML. The hardware is defined based on that. And then we also enable based on configuration generated by MAESTRO, I'll talk a little bit about MAESTRO in the next few slides. So separate executables for each model. So that's, again, keeping that everything is only one version of itself. So this also allows us to have a plug-in plug operation. This also allows for that faster, real time operation. And then it allows us to have better physical locations to use correct cable links for our models and simulations.

Kristopher (10:52):

So then we also have to have all models utilizing the same libraries to allow for ease incorporation. So this is your realtime controller, your inner module, communication, your hardware, and then your input parsing data recording and using utility function. So this is just a library that everything can kind of access information from, which also includes a shared memory region, which is the next point. That is a general region. That is data being stored for each frame by each model. And it's shared back and forth between whatever it needs it, and it's saved there at every single frame. And then we have the last thing which is fault insertion, which I will mention for later.

Kristopher (11:38):

So the next thing is avionics integration and testing at Marshall. So Artemis can support the entire process from concept through hardware-in-the loop qualification. So that first point is talking about digital configuration. So this is everything's in a digital form. All of our models are on different computers that we have at different labs and we're testing everything's source code. And then this is for just general development and refinement. Then we're going to introduce more hardware in the loop, distributed configurations, and then that's using models and actual buses. And then we're even using the flight like interfaces to evaluate the loading and timing requirements.

Kristopher (12:22):

So the next thing is now we're introducing the emulator configurations with some of the actual avionics. This is for local tests and specific interfaces. And then the last thing is the full blown system integration lab. So this is everything is real. Well, everything is real that we can have real.

Kristopher (12:44):

So this is kind of a diagram that's just showing you the general outlook or just overall view of what our labs would look like. So this specifically is a digital version. So over here on the left, if you see my mouse real quick, we have this red box that says MAESTRO. So MAESTRO is another set of code that is on a different team within my branch that is developing this thing. That is the test manager, test conductor. They will run and start Artemis in a certain test or configuration that we need to test. Then Artemis, which is that center column in blue, and then everything below it is your different avionics boxes for the core stage, the booster, the MPCV.

Kristopher (13:43):

But then also if you notice at the very top, we have this thing called the CoreSim. So that is your flex rigid dynamics, environments, and then aerodynamics. So that's the plant of Artemis. That's what's going to drive everything to make it think that it's running, think that it's working. So this is a digital version. So you can see that all of the dash lines on the right and all of the core stage and the booster is all kind of shaded out in gray. Because everything that is being tested is all a digital version within the Artemis.

Kristopher (14:17):

What if we go into the next lab? So the SITF lab configuration is going to change that by everything in the core stage is now using its actual hardware. And now we're using the emulator for the MPVC and LCC. So we're connecting these with the actual physical buses or data lines connected back to Artemis. But the boosters are still digital.

Kristopher (14:43):

And then the last lab, which is that full integration lab, which is SIL, is now using everything real from the core stage and the booster. But the only things that aren't different, are still digital. If you notice are the RS-25s, the main propulsion systems, and then the thrust vector controllers. Now this is the same for the boosters, because obviously we can't have an engine and a lab that's firing, nor can we have massive tanks with fuel. That would be a little dangerous.

Kristopher (15:15):

So now we're going to mention just kind of how the software is designed. So Artemis is composed of five modules. The first one being an executive, which is kind of your base, your main, this is where you input files and then your starting data recording. Next thing is synchronization. So this is where you're going to keep everything in time. And keeping things in time is very important. But then this also manages that shared memory area where that's the data that's being passed back and forth from each and every single frame of the simulation.

Kristopher (15:52):

Then we have your input and output layer. So this is what's going to be communicating to any sort of user interface or just from hardware to hardware. Then you have your models, which again, is that CoreSim, which is your dynamics, your environments, your subsystems, which is your factors, sensors. You can think of that as your pressure sensors, your thermal sensors. And then you have your components, which are the actual computers, the avionics, things that get that information from wherever on the rocket. And then we have our data recorders, which are obviously recording any and all data.

Kristopher (16:29):

So to kind of give a better visual, if you will, of how this is working. So we have a model here, which is in orange. It is split up into a system that is, or sorry, sections of code. That is the first one is on the right. It's called init or initialize. It is a section of code that is run at the initialization of the simulation. Then we have run, which is anything and everything to be run during the simulation. So this is everything that each model must do or will be doing at whatever time it needs to. Then we have shutdown. So a simulation just can't stop. It's got to shutdown, especially when you're using hardware. So the hardware can't just turn itself off. So we've got to, and we don't want, because that could damage the hardware. So we have to have a section of code that is going to shut everything down.

Kristopher (17:31):

And then, so the executive main is then going to have an instance where it's going to start everything and then initialize all of this code. And then the executive main provides standard set of rappers. That's that prototypes. And that's the specific code that is going to be run at whatever phase of the simulation. Let's keep going. So now we're going to add on the sync library. So the sync library is responsible for commanding the executive to call those model functions. But then executive also has to register itself with sync to gain any information that it has specifically from that shared memory.

Kristopher (18:19):

So the models also have to do this because it's got to get that sync timing and then any information that shared memory region has. Then we have to add the IO layer library, which provides functions to the model to read and write to the simulated area or real hardware devices. And then the IO layer library also provides functions to initialize safely shut down devices.

Kristopher (18:48):

So adding onto this some more. Now we may have multiple models within a single computer. So to do that, we still only have one version of the sync executable, which is going to be in control of the sync libraries of being each model. And then we have an IO layer version of that, the same that controls the each individual models IO layer libraries. So on each one of these multiple models may run, but there will only ever be one of those two executables, which is the sync or the IO layer.

Kristopher (19:27):

So just to recap on this, the sync, it would be that shared memory region, which is next case blackboard, is what we call it. So memory region defined by a group of structures and each executable writes about our output data to this. And then the IO layer is the shared memory region for anything, buffers populated by the model with bus data to be sent to a driver. And this is also to be received from that driver.

Kristopher (20:00):

So adding onto this even more, now we have multiple computers in this configuration. This is going to be communicating to an avionics box or giving data to a data recorder. So each machine has a separate version of that, but then only one of those that the sync is called the master. So if you remember, the sync is the timing of the simulation. So we don't want anything or everybody to have different timings. We want them all to be at lockstep. So we have to make one the master and all of the other ones follow that.

Kristopher (20:35):

So then the next thing would be the hardware avionics boxes may or may not be present, but they can communicate directly with the data bus types, which is the 1553, the ethernet and the 422. And then we have data quarters that tap off each bus, as well as off the infiniband network to capture and archive data each frame.

Kristopher (20:59):

So the next thing I want to talk about is fault insertion. So there's two types of faults. First one is overwrite simulation variables on Blackboard. It's the least expensive. It's relatively the easiest. It's simple. It's just going to be a basic line by line type code input that we can change rather easily. But the thing is, it doesn't cover everything that we need to have it do. And then the next thing would be the execute an embedded fault in the simulation. So this is going to take a lot more time, take a little bit more effort, but then we have a set flag or tripping device, if you will. That's now set in the code that, if we want this set, it will trip this fault flag to happen.

Kristopher (21:50):

So how we would do this is then we would have a peek and poke via MAESTRO. So if you remember again, MAESTRO is that test conductor. So they set up the test that you want to run. So that means they can also do these types of faults where they're going to introduce a potential issue or problem in the simulation. So the next thing would be a separate fault insertion function. So this would be, I could write a set of code that would be having a certain criteria or certain timing or something that has to be met that then would set a specific set of code that would set that fault to occur or that issue or whatever to happen.

Kristopher (22:38):

So to give a little bit more information about what this is and what that's talking about is a fault insertion would be a problem, or it could be a problem happening during the simulation. So one example would be say, an engine were to go out during any time in the flight. So if we could set that one of the four engines to go out at the beginning before it even launches. Now the flight software has to determine whether or not it can actually launch. So if it doesn't, it actually shuts down, which is what it needs to, it lost an engine.

Kristopher (23:12):

Or the next thing is that maybe that engine goes out later. So it will try to determine whether or not it can still continue with the simulation or continue with flight in this case. So if it can't, it'll launch the SLS or the launch board system and get the crew module away from the core stage and the boosters, or does this happen later in the simulation where the engine suddenly turns off and it's almost there to orbit insertion. So maybe it's not going to be the correct orbit, but maybe it's close enough that it can at least still get in, and then we can do whatever other maneuvers or corrections to get in that correct orbit.

Kristopher (23:55):

So the last thing I'm going to show you guys is a photo of what is the core stage lab that we have. So the main importance to understand here is that everything that you see on this giant half circle rack is all of the components, the avionics, the cables of SLS. So this is basically the brain of the rocket. So this is only half of it, but it is actually the full thing. So if you were to cut this horizontally down the middle and take that, flip it back on its side, then you would suddenly have the full ring, inner diameter of SLS, and you would have every single avionics box and computer within the exact location that it will be during launch. And then this is where again, we're going to be testing all of those scenarios, all of those different tests and all of those different issues potentially to make sure that the flight software is ready for launch. And with that, I think I'm done and ready for any questions you guys may or may not have.

Danielle (25:07):

Thank you so much, Kristopher Miller. If anyone has any questions, you should be able to unmute yourself now.

Joshua Rovey (25:13):

Hi Christopher. This is Joshua Rovey, University of Illinois. Thank you very much for presentation this afternoon. Could you say a little bit about, or and maybe I missed this. I apologize if I missed this, but could you talk a little bit about redundancy and what happens if there's some failure in one of the components?

Kristopher (25:35):

Yeah, so a lot of the redundancies are built in already to the simulations or say into the flight software or the specific component per se. So in that case, some of the redundancies are dependent on what you're doing or what is taking place. So in that case, if the engine goes out, it's not necessarily the end of the world, but depending on if it's early in the flight or later in the flight, the flight computers will have to then do a vote, because there's three different flight computers. So they will have to do a vote together to determine whether or not they are going to continue flight or not. And then some other redundancies are just built in already hardware wise or in general... I guess you could say that's also what we're testing is those redundancies, if that makes sense.

Joshua Rovey (26:37):

Yeah. Great. Thank you.

Heidi (26:39):

Hi Christopher. This is Heidi Bjerke from the Illinois Space Grant. I wondered if you would share how you ended up in this part of the program for Artemis.

Kristopher (26:50):

So part of this was based on some classwork at my school at Missouri University of Science and Technology. I was doing a lot of modeling within just physics of just propulsion systems and then just in aerodynamic vehicles. And I was reached out to by one of my coworkers on the team. And then I was actually contact with the lead of this project and we got to talking and essentially, it was a field that I wanted to get into more of and wanted to understand a little bit better for myself, just because it was much more involved in the general process of everything, knowing everything about SLS. So I would say I got a little more fortunate due to the fact that they were able to reach out to me, but there are still ways to, I'm sure there's many, many other companies out there that are doing simulation work that you can probably look up and find.

Heidi (28:08):

Thank you.

Kristopher (28:08):

Mm-hmm.