MICHAEL VESPERMAN: OK. We're going to go ahead and get started.

So, one, I'll say, thanks everyone for joining us. If you didn't know, the closing keynote is taking place right now, and they're featuring a comedian, Rob Corddry. This class certainly won't be inspirational, as visually appealing, or nearly as funny. If you've ever seen the show, *Ballers,* or *Hot Tub Time Machine,* this is the actor from that. He's actually hilarious. I almost didn't come to this myself.

OK, so two words this class will be-- it's about process automation, OK? Which leads to making money. This is what I'm offering you. We're going to teach you guys a little bit about--

Oh, let me introduce us. So my name is Mike Vesperman. I am a Subject Matter Expert for PLM, our Fusion Lifecycle product. Been with Autodesk for two years now. Prior to Autodesk, I spent some time as a consultant, implementing enterprise-wide technologies. Been around PLM for 16 plus years, a degree in Mechanical Engineering. I have a Master's degree in Industrial Engineering.

With me today is, Tony. Oh. Go ahead, Tony. You can introduce yourself.

TONY ZOHREHVANDI: Yes, good morning. Tony Zohrehvandi. I am a PLM Technical Specialist based in Dallas, Texas. Joined Autodesk last year. I have worked for many PLM companies and [? result ?] [? confirming. ?] [INAUDIBLE] I've done many PLM implementations. And I have a Masters degree in Mechanical Engineering. And if you cannot say my last name, just say "Tony Z."

FRED SMITH: And I'm Fred Smith. I'm a relative newcomer to Autodesk. I've been around the PLM industry for a long time. I am not a mechanical engineer. Decidedly not a mechanical engineer, but I can hold my own in [INAUDIBLE] software. So I'll move over there, and I'll-- Thank you. Thank you.

MICHAEL VESPERMAN: Yeah, thanks everyone for joining. So the summary of this class is we're going to spend time understanding how we can evaluate a business process, look at opportunities to drive process automation into that process. We're going to then dig into-- in a little bit about how to design what a To-Be state would look like for this given process, and how you model it, and design it, and review it. We're going to do a technical review of the frameworks of how you add scripting

and logic into your workflows. And we're going to go into the backend. If you've been attending some of the other Fusion Lifecycle classes, this is a 200-level class, maybe even pushing 300-level class. We're going to get into some of the backend stuff. We're going to go into some workspace modeling. Talk a little bit [? about-- ?] Fred's going to do a live demonstration about how we add this decision making and record validation to our processes.

Our agenda is, I'm going to cover a use case of a real-world example of how we used decision making with scripting in a business process. We're going to go into some of like everyone kind of understand the foundation of how you add scripting to your workspaces within Fusion Lifecycle. And then, as I mentioned, Fred's going to go through a real-world, live example of how we're going to add script, how we can then see and test that script, and debug, and things like that, so--

First lesson, whenever you endeavor on any type of project like this, first and foremost, figure out who your team is. And then define roles and responsibilities. I was going to do this class on my own. Somehow I conned these two folks to join me to do this class with me. And I kind of took on a project management role. Tony took on the frameworks and architecting this class. And then Fred said, I'm going to do the heavy lifting, right? So he's our builder. And that's sort of how we model this. And that's kind of the approach that we took as we went through the implementation process of this workspace.

So a little bit about our customer use case, company by the name Danby. Danby came to us about this time last year, and they were actually endeavoring in a evaluation of different PLM systems, and I was working directly with them. And like most companies, they were coming to us for new product introduction, stage-gate execution, item and BOM management, and change process.

For those in the room, how many-- This is why you guys are looking at Fusion Lifecycle, or are using Fusion Lifecycle for. Is that pretty much what everyone's doing? OK.

So they, in this evaluation process, in the spring, like all typical PLM evaluations, these span many months, right? A lot of demos by guys like me and discussions with our salespeople and our consulting partners. Well, they had a corporate decision made. They were going to take an outsource process for reverse logistics, and they were going to bring it in-house. Reverse logistics, if you're not familiar, is where you take your product that you don't like, and you usually call, and you get issued a return merchandise authorization, and you ship the product.

And then that gets received, and then they got to do something with it.

And the goal of most reverse logistics processes are to either repair it and return it to the customer, put it into like a scratch and dent area, or scrap it, right? So you have to make these decisions. And they make a lot of different types of products. So Danby makes wine chillers, refrigerators for dormitories, if you have any children in college. I had one when I was in college. And [? Brian Shed ?] [? into ?] the back is really excited, because he found out these guys are the ones who make a kegerator, and he is a brewmaster. So these are the types products-- even air conditioners, OK?

Their challenge? 3,500 returns per month. Now, when I talked to Mark Tanner, and I told him I was going to use some of these slides, he approved them. So he did allow me to do this for this. He said, please tell everyone that we don't have a huge quality issue. We sell millions of these products, and we were getting 3,500 of these. So I wanted to make sure I'd say that for him.

The other thing is-- So the other challenges were, they needed to train a whole lot of people really quickly. And the process was really well-documented, but they had no system to manage it. So there was no way to define the business logic, and then they wanted to do an integration with their ERP system.

If you want to see one of these things live in the field, here is a Danby refrigerator. And this happens to be at my barber shop in Brooklyn. And I took this picture Saturday before I came. And I was like, oh wow, I'm doing a presentation on that company. So, this is what they look like.

TONY ZOHREHVANDI: And the barber shop serves alcohol?

MICHAEL VESPERMAN: The barber shop serves alcohol. It's like a--

TONY ZOHREHVANDI: It's a great barber shop.

MICHAEL VESPERMAN: It's a spa for men at my barber shop in Brooklyn. So here's their as-is process. So this is what they had for documentation of this business process. It was not modeled out in a workflow; it was just written into this quality doc. 10 pages long. There are product rules here. If you can

see-- it's tough to see, but it says pass, fail. And you're doing different types of inspections, and you need to decide, based on those answers, what to do with this thing. Ultimately, they're trying to decide whether am I going to remanufacture, more testing, scrap, scratch and dent, OK?

We spent some time, and Tony and I enlisted him to help me with this, which is par for the course for Mike. I pretty much use anyone I can to help me do things. But Tony helped me basically take that SOP document, and model it all out. This is how we designed it so that we can then take this to Danby in it, and get their confirmation that this is the process that they were looking for.

This is called a swim lane diagram. And basically-- and I'll let Tony explain it-- but swim lane diagram is just basically how you do your decisions through the workflow.

TONY ZOHREHVANDI: Yeah.

MICHAEL VESPERMAN: There's a couple of things Tony wanted to point out. So Tony-- he didn't mention this-- but prior to coming to Autodesk, Tony spent a lot of time doing consulting work for Capgemini, Accenture, and things like that. So he's really knowledgeable on how to design these things.

TONY ZOHREHVANDI: Yeah, I mean, you look at their as-is process, it's like, 10-page documents. And in order to do something, they have to follow instruction, go to page 5, go back to page 2. But that's the beauty of this, that you can capture all that process, create this to-be process.

First of all, there are many different methods capturing the process. This method, I actually had these boxes. There are three sections. The top section is the tool, the middle section is the process, and the bottom section is the actor. So you kind of capture everything in that 10 page, and put it in one flow chart. And when you do this, not only you're summarizing the process, but you'll be able to use this for hiring new people. You train them on this. You hand them this paper. This is our process.

And also, you'll be able to identify a single point of failure, or redundancy in your process. So you can identify many issues that, if I just hand you that 10-page document, you won't be able to identify. So that kind of helps us to get ready to actually implement this into the Fusion Lifecycle.

MICHAEL VESPERMAN: Yeah. It's searching for the bottleneck. That's what we're trying to do, right? That's what most business process re-engineering is all about. Find the bottleneck, and then get throughput through that bottleneck as fast as possible. You're only as fast as activity can go through your bottleneck. If you've ever read the book, *The Goal,* it's kind of the focus of that book.

OK, so here's what a workspace would look like-- and this is a screenshot from Fusion Lifecycle-- without adding scripting and logic into your workflow, OK? So here, kind of just typical item field information we enter. But down here, when you hit this area-- and this is kind of-- scratch and dent's a bad title for it. It's actually the quality engineer's second point of failure analysis test. It's gotten through the material handling. Now the quality engineer is taking a look at it.

He is going to answer some questions, OK? Cosmetic inspection, odor, safety issues. But there's all kinds of combinations of these. And so once he's answered them, he then would need to think, which of these choices do I pick? Scrap, testing, remanufacturing. In the workflow viewer within the workspace, you have these three green arrows.

Now I have to think. Likely, if you're new to this, you have to go grab your 10-page document, flip to the page and go through the rules. That's going to spend time doing it. So when you add logic to this, in this case, 75% or greater of cosmetic inspection. We have a damaged cord that's not repairable. And there's no odor. Scrap. Right?

And there's all kinds of different rules. If there's odor, scrap. They don't even mess with it if they find that they can't get the odors out. But if it's got a repairable cord-- As an example, here's a repairable cord. And then here is cosmetic damage of 0% to 50%. And it decides for me, remanufacture. OK? Making sense On how and why we do this?

OK, so in order to understand how to go through this process of adding small pieces of scripting-- I was talking to one of the product managers before this class. He said, simplicity. Write small, little scripts. If you're writing more than 20 lines or 30 lines of code, you're likely doing too much with your script, and you should be rethinking it. So the intent here is easy, not, oh my gosh, I got to program. I got to go compile code. This is all using JavaScript. And so Tony's going to take you through a little bit about the basics of this, so we can all get on the same page.

TONY ZOHREHVANDI: All right, thank you. Before I start, I just want to say this is my second year at AU, and I just love this event; it's a great event. And I love the way they put the snacks, right? They put

either a very healthy snack, or very unhealthy snack. Vegetable and donut. [LAUGHS] I think if you eat one of each, they kind of cancel each other out.

But anyway, about scripting. So there are two ways of really making things happen behind the scene in Fusion Lifecycle-- what we call internal, which is through scripting and using for workflow transition, or workspace behavior. Or you can do external, where we use for integration, or even data migration, and having things happening externally. So there are APIs available just about for every field in Fusion Lifecycle that you can access, and you can either get information or put information. But in this case, we are talking about workflow transition and workspace behaviors.

So there are really four types of scripts. I'll explain that fourth one; you only see three here. But we have precondition, validation, and action. I think it's important to know where to use them and exactly, within your workflow, where to use this. So for the precondition, when you go from one state in your workflow transitioning to the next state, there [? are ?] certain condition has to be met for that transition to take place. So that's where you use the precondition.

And once those preconditions are met, then the transition is allowed to move to the next state. Validation is really after the transition is completed. Before moving to the next state, you want to validate certain things. Did this happen? Did that happen? So it kind of validates certain information about that transition.

And the last one is action. Action-- that means, OK, I made my condition. I made my validation. Now execute these actions. So that action always happens as soon as you complete the transition. The action happens couple different ways. You can actually execute the action within the workflow. Or there's another way to execute action.

So this is the second way to execute an action script. When you create a workspace, you can actually indicate that on create, execute a script. Or on edit, execute a script. So that means every time I create an item within that workspace, that script runs. Or every time I edit that item on the workspace, this script actually gets executed also.

So I put this table together for your reference, if you want to read up on it later on. But the fourth point that was talking about is the library. So there are certain scripts that you can reuse in many different locations, and you don't want to repeat them every time. So you create a library of your scripts, and just reuse them. And you can reuse those library scripts in any of

those condition, validation, action scripts. It's almost like a subroutine you create, and you just reuse it over and over.

So how do we start creating scripts? So we go into the top menu. We go into the Administration's pull down, select Setup, and select Scripts. And this is what you see when you do that. There are some scripts already available out-of-box. You click on the New Script, and that's how you create a new script.

You can edit an existing. Sometimes, I use that a lot. I just copy paste from other scripts; I don't need to recreate everything from scratch. You can delete a script, or you can run a [? Where ?] [? Used. ?] The [? Where ?] [? Used ?] tells you where this particular script, in which one of these workspaces the script's used. Kind of a useful feature.

Now, once we create the script, this is what you see. You see, you select a script type. Give it a unique name. And I think it's recommended to give it a nice description that, when we look at your script, you see the description tells you what that script is about. This is where you use the library. So this get user name is a library script that I'm putting it into the script. That way, I don't need to rewrite it again.

And also, make the scripts kind of manageable. Don't write 100 lines of scripts. 15, 20 lines-- kind of manageable that way. I think the rule of thumb that I use is if I can see it in my browser, I'm happy with it. If it is more than that, that means I'm writing too many lines. And of course, you can Save, Save and Close. There is a debugging tool; you can test your script. And if there is an issue, when you save it, there are error logs that you can review to see what are the problems with the script.

On the debugging side, there is in-editor warnings. So if I have the wrong line or something, I get these warnings when I did something wrong with my syntax. Or I can do a test. I can run the debugger. Or I use this a lot-- this print line command. So if I want to kind of test it to see if I'm getting the right value, I use the println, and I run it. And I see what item shows up there. That tells me if I'm getting the right value. So very useful, powerful debugger. You can kind of debug it line by line.

And now the use cases on the condition-- I just want to use some example as Mike mentioned. So for condition, you notice here, in the requirement definition, there are many different ways that I can go. It's almost like the decision-making in the flowchart. How do I go this way? Which

one turns green for me?

So this is where I use the condition. It hides the workflow transition, unless a certain user is logged in. For example, if I'm not the user, I'm just looking at the workflow; I don't want to be able to go and execute that action. So you can actually put that in the script, so that only this user can approve this. Or, based on certain information in the item detail of the workspace, it would take the right routing for you, just like Mike explained, based on if it's an odor problem, take this route. So you can kind of drive that using the condition script.

And then validation. I think that's-- Again, you validate certain things happen. This is a pretty useful script, like if you don't want the process to move forward until certain activities are completed. For example, I've seen a lot of examples where certain things have to happen before it moves to the next workflow state. So that's where I use the validation. For example, if there are multiple tasks within that workflow states that need to be completed, I don't want people to move this from requirement to development. I want all the tasks and the requirement to be completed before moving to the next state. So that prevents people from just moving the workflow without completing the activities within that state. So that's where you validate; make sure certain things happens.

And then you can, of course, put a nice descriptive message. You actually control these messages in the script. And you have to make these messages where the user can look at. It's like, OK, that's what I'm missing. Don't put some kind of weird message that they have no idea what's the problem. So for example, here, something needs to be added to the affected item. You need to add either object to the affected item.

And then the action script use cases-- There are many use cases for that. You can use it to create other items. For example, within a change request, an action script would be initiated and change order. Or within the problem report, an action would be create a change request. Or send an email, or set some milestones, or set some attributes. So there are many things you can do. And then this action script, like I said, you can run it from the workflow. You can run it when you create an item, or when you'd edit an item in the workspace.

Another way of doing this action script is you can actually run it ad hoc-- this icon in here. So you can actually create an action script and then run it ad hoc. For example, we used that in our Octopod integration where when you select an approved manufacturer, you actually come and run the Octopod script. And it goes to the external side and grabs information about that

particular item, and populates the workspace.

All right. Now, I'm going to hand it to Fred.

[AUDIO OUT]

FRED SMITH: This is a demonstration. I'm going to focus on the Build It part, OK? So what do we do first? We configured the workspace, and we added some attributes that we're going to use to route the workflow. We identified the decision points, and what logic was going to happen at those points. Then we wrote the code, created users' permissions, and then started testing, essentially. So it's a generic process. And, of course, when you get these slides, if you can't remember that this is a good order to work-- to find the data, then to find the process, flesh it at a high level, the diagram level, the workflow level-- the swim lane level, if you will. And then start writing the code.

OK, so any time I'm doing any kind of scripting, I always need at least two screens. Well, there's one screen. And on your chair-- keep it in your lap-- this is going to be the second screen. I'm going to refer back to this frequently. And what I'm going to do is I'm going to start out real easy. Over on the far left, where it says 1a, I'm going to set a value on a record, and that's going to determine which way the workflow goes. Simple, OK?

Then over on 1b, where the process says, Business Review, there's going to be some business logic that's going to determine which way it goes automatically. It's going to be an implicit routing. And what I'm talking through here are the lessons at the bottom. So those first two are really all about enabling processes and routing the process, OK?

Then lesson 3 is going to be about validating processes. Well, we wanted to keep this pretty generic, and so my policies that my company is going to implement are simply that planning must result in a plan-- Big news. And then plans must be followed. So those are my two policies that I'm going to implement. And I'm going to show an error message when you don't follow one of those policies.

And then lastly, we'll automate some actions in 3a, up here in the top. What we'll do is we'll start a process in another workspace. And in 3b, we'll just send some notifications when certain things happen. Now, so those are the lessons we're going to go through real quick.

But let me point out that there's sort of two paths through this workflow. There's this lower path, which has to do with-- and we call this workspace the business decision workspace. So

everything's a decision coming in. And at first, a decision could be, oh my gosh, it's an emergency. So let's take it. Let's go do some stuff. And then afterwards, we'll analyze what happened and how we got there, OK?

Versus, here's a business decision that's going to come in and do a business review. And then we're going to decide whether it's Opex or Capex or if we're going to cancel it. And so, this is the more contemplative kind of decision that has to be made every day, where the top route-- those are emergency cases, OK? So just so you know, that's how the workflow is logically separated. So I'll spend some time on both branches though. And that's repeated at the bottom of your map.

One thing I wanted to point out was that, don't feel like you have to capture any code in your notebooks, or anything. You can get a document where we did a step by step documentation of the workflow process here, related to this workflow. We wanted to pick a workflow that wasn't out-of-the-box. So that you could get the notion that this is all completely self-contained for this event. So everything you need to know is somewhere in that document in order to recreate this workspace.

And then over on the right side, we give you all of the code associated with it, OK? And there are comments in each one of those JavaScript files about where it goes. So you can copy and paste it in. And I think that's it.

So now, I'm going to switch over to the business decision workspace. So we started with a vanilla tenant. And the things that we've added are on this Business Operations menu. So we have a workspace for business decisions and a separate workspace for the root cause analysis. And the idea was that there might be access rights differences between those two. So there may be reasons to partition them. You can imagine your own scenarios.

But when I click on Business Decision, it brings me into the workspace. These are the business decisions that we entered just in practice and review. So you might need a better display, some signage. I'll run through that. Oh, going to plan the company Christmas party. OK, I've got to make some decisions about that. Might need a new lathe in my machine shop. So these are just arbitrary decisions; you kind of get the idea what's going on there.

But let me go through the process of creating one. OK, so I'm going to give this one a title. And the scenario I'm going to follow through is that someone ran over the sign in front of our

headquarters building, so we need some new signage. "Need new signage. Truck hit old sign."

And then I have a flag right here that's going to determine something, OK? Is this business decision-- is this an emergency? And I have a pull-down menu here, where I can obviously say either yes or no. And for some reason-- it might be because I'm a little bit too zoomed in. So for some reason, I couldn't pick-- Oh, great. Oh, there. OK.

And then I have, by the logic in this workflow, we wanted to have one person who's going to own the issue for the lifetime of the issue-- for the lifetime of the business decision. So I'm going to own this one. So I'm just going to pick myself. And then classification. This is something that I didn't build into the scripts too much, but I just wanted to throw it out there as you can embed an awful lot of business logic just by classifying objects correctly. And it gives you opportunities for more complex logic. I'll say more about that in a moment.

But let's say I'm going to change the classification of this decision. I'm going to call this a-- I'm going to increase sales by having a good sign on my front lawn. Now, what you notice is that it actually set some attributes, or it provided access to attributes differently depending on what I had picked in that previous classification. So since this is an increase sales type classification business driver, it gives me these lists. So I'm going to say, new customers. And the issue is only in America. Only at the headquarters, whatever.

I may have an expected cost. So the expected cost-- I don't know. A new sign-- I'm going to say that this decision is associated with $10,000. And obviously, I can have other attributes. Maybe my opportunities is to use new branding. And I could collect challenges, alternatives. My idea, when I was first doing this, was this would be information that would be passed down to document the decision process later. So at this point, I'm just going to save this. That's enough of my typing.

And now, I'm going to look at the workflow actions. And because I said this-- oh, I said this is an emergency. OK, so let's see what that does. So because I have a precondition script-- and let me try and give you a little bit more context here. OK, so right now, the business decision itself is in a new state. And you can see that the Mobilize Resources transition is lit up because we're going to go into an enterprise down. Well, this is just signage, but that was a mistake, right?

So I'm just going to go back here, edit the record, change the emergency. No, it's not really an emergency. Save it. And now refresh this. And we'll notice that that allowed a different

transition from New over to Business Review, and it took off the one up to the emergency route. OK, so that's more appropriate. I just wanted to show you that.

So the way that link behaves, that's sort of my lesson 1a, where we just have a manual flag determining which way the route's going to go. At this point, I'm going to go ahead and advance this forward into the Business Review. "Guys, have a look at this." And save it. And as you would guess, it's going to move into the Business Review state.

Now here, I have some more complex-- or implicit logic. I wanted to show a routing by implicit means on my lesson 1b. And because I had entered that the signage is expected to cost about $10,000, it's going to choose just to call it an operational expense rather than a capital expense. Now, that's just the only criteria I put on it. It could be much more complex than that-- the decision. And I'll show you where that gets coded in the scripting.

So I have a choice; I can either cancel it or move it forward. I'm going to move it forward to the operational. I'm going to confirm that, yes, it is an operational expense. Sorry, need to clear that and put that again. There we go. Opex. And go.

Now, it's going to transition into a management planning state. It can stay in planning for an arbitrary period of time, but I don't want to leave that state until I actually have a plan in place. And did this not refresh? I'm going to refresh it. There. OK, now it's in the Management Planning state. And I want to make sure that I don't move forward-- I don't advance until there's actually a plan. But let's see what happens when I try to advance it.

Here's that red message. And I'll show you in the code where that red message comes out. And it says, "there must be a business plan in order to leave the management planning lifecycle state." " Well, what do I mean by business plan? Well, there's a tab over here that says, "Project Management." So I can go into Project Management, and the intent is that this is a list of activities that I'm going to perform.

So I'm going to add one that says, "shop for sign." I'm going to start it on Monday. It's going to take me five days. And then I'm going to install the sign. And I'll start that on the 28th. That's also going to take five days. Great plan. You get the idea. Save it.

So now I have a plan in place. So what would you expect? So if I come back here, and try to mobilize this again-- I have a plan now. And save it. And it will advance into an In Process state. So now, people can be assigned to go off and do some shopping, install the sign. And

the business decision itself will stay in this In Process lifecycle state.

Now, it should stay here until the plan is complete, right? It should stay here until those two steps have been executed. So if I were, again, to try to advance it into a completed state, I'm going to get an error message which says, "the business plan is not yet 100% complete." So, big surprise. So I'll go back to the business plan.

And of course, this will happen over a period of time, but I'm just going to edit it, and I'm going to set the percent complete-- I'm going to edit it. There we go. I'm going to set the percent complete to 100% on both of these. Of course, they may spend some time in less than 100%. But until I do this-- until I set those two to 100% complete, it would not be allowed to advance in the workflow, OK?

Now my plan has been executed. And then come back here. And I can say we're complete. We're done. "Sign installed." Save it.

Now, in the interest of time, a little bit, I just want to show you what happens on the upper path. What's going to happen if we said it was an emergency and it came into this Enterprise Down state, it would stay there. And we could actually execute a script-- every time we take an action, we run a script. So there's a looping capability that I just wanted to point out.

Then once we say that we're no longer Enterprise Down, we can we can move on. But we want to do a root cause analysis before we actually call this business decision as fully completed. So what this is going to do is it's going to spawn a record in another workspace called the Root Cause Analysis workspace.

And then that goes through a separate lifecycle, independent of this workflow status, but it's a much simpler one. And it collects the information about the root cause analysis, so we can hopefully never fall into this trap again. And then once the root cause analysis is completed, it automatically transitions into a complete state. It transitions not only the Root Cause Analysis object record, but also the Business Decision record. It automatically does that transition.

MICHAEL VESPERMAN: When you do that, by the way, you can actually pass information collected on this record to that other record. So you don't have redundant entered information, right? So it's about trying to make this take place faster. A lot of times, when we're engaging some of our customers. it's about redundant entry of information. In this system, I could edit here. This is my edit here. We're trying to eliminate that. Very good.

FRED SMITH: OK, so now I want to go into the coding-- Oops, where's my coding windows? There we go. And we'll start back here. But before I even show you a line of code, I want to show you a really useful reference. And that is, if you follow the-- over here under Help, and take the Help Guide. Go to the Help Guide link. It will bring you to this page. I just leave this page up all the time while I'm scripting. And it's the Developer's Scripting Guide. Developer's Scripting Guide, and the Scripting Reference.

And what this page contains is all of the functions that you can invoke inside your scripts. In fact, they've used that println that Tony likes. And if I scroll down, I'll see for different objects, every item has a milestones tab. And you can get to the individual milestones on that tab. It's just a nice reference. Whenever I'm trying to figure out how to add a call to a certain function, I can just come here and get a line of code, read what it does, and fill it in. So very helpful.

Now, where the scripting takes place-- So this is the Workflow Editor, as I hope everybody recognizes. And this is where we're going to plug in the endpoints of the scripts. So like I said, on this first one, whenever it's in this new state-- Well, sorry, let me back up just a little bit.

So for this entire workflow, we'll see-- what are there? Nine-- nine states up on all nine gray boxes. And from those nine gray boxes, there are exiting transitions as well as entering transitions, OK? And there's a nice practice, when you're building code, especially precondition codes that Tony talked about, and even validations-- it's nice to put that all into one file, or one place-- one script. And the way you build the structure of that one script is just by looking at the boxes on this map. And that's what I wanted to point out.

So let's take a look at the names of the states. They are "New," "Enterprise Down," "Root Cause Analysis," "Complete," "Business Review," and so on. And so, I have a template of that structure that I'm going to use in several places. So for a condition script, it takes on this pattern. You declare a variable up the top. And this is a best practice, if you will, or at least a common practice. You can obviously decide on your own whether it's a best practice, but it's a common practice. Define this Boolean variable, then you're going to have a switch statement, which I'll expand and show you what it does in a moment. And then it's going to return a value. So for a condition, the important parts are that you compute a Boolean variable, and then you return a Boolean variable. That's what you have to do for a condition type script.

Now, we put these in. And let me back out now. And I know that's much harder to see, but what I want to point out is that there is a switch statement on the customTransID. That's the

name of the transition, essentially. And then the way I structured this code is there are commented blocks. And the commented blocks-- [INAUDIBLE] OK, I won't save this. But the commented blocks have the names of the lifecycle states. So for each block-- and then right below that, you have a case statement for each transition out of that state. And that's all that this script does in this-- I just have this structure script because I just wanted to show very clearly the structure of this, and how it's related back to this diagram. So I had a comment for each one of these blocks, and then I had a case statement coming out for each one of the transitions out of that box.

Now I've said that that was just an example of the structure. If I wanted to see what the real code looks like, first I want to show you where you put that. And on each of these transitions, I double-click on it. And you'll see next to the precondition statement, that it invokes BD Workflow Conditions. Business Decision Workflow Conditions. And then I use that same script on every transition. And the name of that transition-- like I said, that's real important, because it's in the case statement in the code.

On this dialog that pops up when you double-click on the transition, there is a button that says Show IDs, and what we're talking about here, the Custom ID-- this is what goes into that case statement. So you just copy this string and paste it right into the script. Edit that script and paste it right here on the case statement. So what this script is saying is that I'm going to assume that I want this transition to be enabled. That's what the first line does. Assume you want it enabled. And now, calculate whether it really is enabled or not, OK? So calculate the false statement.

So for this transition, if I'm trying to cross this [? MOBILIZE_EMERGENCY ?] transition, it's going to execute one line of code that says, look at the business decision. Look at its attribute-- and I really shouldn't change code while I'm demoing-- Look at its attribute, and then-- Sorry, I kept trying to restore this while I talk. There we go. Look at the value of that item-- its emergency field. And then if it's yes, then the enable transition is going to be true. So for this case, for [? MOBILIZE_EMERGENCY, ?] if the emergency flag is set, go that way. That's what it's saying. Or in the next one, if the emergency flag is set to no, go this way. But you'll notice that those two statements can't be both true at the same time, OK?

So that's the logic. And my whole purpose of this discussion was to show there's a structure that's independent of the logic. The logic itself is really pretty simple. And the structure itself is fairly simple. But then you just take that line of logic business code [? and ?] put it into the

structure. And that's how you can write these scripts.

MICHAEL VESPERMAN: And you'll notice it like it seems unnecessary, but if you still doubt some of those transitions that there is no logic [INAUDIBLE]. So it'll always be true. But what happens a lot, is that someone will come in from the business and say, hey, can you do the BOM?

FRED SMITH: Make a change.

MICHAEL VESPERMAN: [INAUDIBLE] still doubt you just need to add a [? logic work. ?] You don't have go [? work it. ?] So that's kind [? why you need ?] [? it. ?]

FRED SMITH: Now, the code for that lesson, 1b, in the middle of your map there-- the code for that is deciding whether or not we've got Opex or Capex. So in order to enable the operational expense transition, I'm going to invoke a library function that's coming out of this BD_Business_Logic library. And I've got one line of code that says, requiresOpex.

Now what does that function do? For me, the workflow designer, I don't really care. I can invoke it and I can say the programmer knew what he was doing. The business analyst knew what they were doing. They wrote that code. I don't care, OK? So what we're seeing here is a separation of concerns that is often a good software engineering approach of separating things out so that the person who's writing the business logic doesn't have to care where or how it's used. And the person who's filling out the workflow doesn't have to worry about testing and validating the business logic.

Now in most cases, one person is going to do both jobs in a lot of our implementations. But that doesn't have to be, and it's still a good practice-- it's still a best practice to separate out as much as you can. The only thing that's important for whatever function I put on these lines, is that they return a Boolean value, just like I said. So you can calculate whatever you want, and as long as it returns a Boolean value, it'll work here.

And so that's lesson 1a and 1b-- the codes associated with those transitions. Now let me close that window so I don't accidentally save it, and go to a validation script. And similarly, as I did the precondition scripts-- the conditions-- every one of these transitions invokes the same validation script. So whenever there's a transition across an arrow, it's going to execute this code. And that code looks like this function. BD-- or this script-- BD Workflow Validation.

And what I'll just point out real quickly is it has exactly the same structure as the preconditions, OK? The only difference is in preconditions, remember the variable was a Boolean. So I had to

return true or false. For a validation, you return an array of objects. They happen to be strings. We're going to return an array of strings. And so this first line creates an empty array. That's all it does. And then what's down in the switch statement is going to calculate that array. And then I'm going to returnValue those messages.

So the important part of the logic is down in here. Ultimately, if what comes out at the bottom here is still empty, the transition will succeed. It will go over that transition. If there are any messages in there, it won't succeed, and it will give you a little pop up-- the red text you saw on the alert, OK? So what might I have done for this logic? Go back out, and I'll scroll down a little bit. Say for Opex-- or to mobilize, because I had decided-- remember, I had decided that something was Opex. And now, before I can move on with my project, I wanted to check and make sure there was a plan.

So the way I checked to see if there was a plan was just this one line of code that says, are there any items in the plan list? As long as there's more than one-- Or I'm sorry. If there's less than or equal to 0, then I'm going to display in red-- I'm going to display this message. And that's a messages.push. It's just a function that you have to learn. But it does bring up a good point. One of the reasons that we have such power here is because of JavaScript. And if I look at JavaScript-- if I look it up here real quick-- It's very easy to Google, and you'll find a couple of different sources that you like-- the W3Schools is actually-- they train people how to use the W3 languages, of which JavaScript is one.

And so here's just an example of the push function. You can see that there is an array of strings. And I'm just going to put one more string on it. That's how you use it. This is how it comes out. You can see "kiwi" has been added to the end of the list. And that's exactly all the behavior that we need in the scripting. But one of the real advantages to using JavaScript is that there's such a clear and ubiquitous sources out on the web to find those.

Let's see. I think that gets us through lesson 2a. And then on 2b-- let me go back to my script-- So 2b is when we're demobilizing after an in-process. So here, the comment says, "in process," and the case says [? "DEMOBILIZE." ?] And here's some arbitrary logic. It's not important if that looks complicated. I mean, it is, but you'll figure out the logic, ultimately. But what's important is what it's trying to do. It's just looping over all of the workflow tasks, or all of the tasks that I said needed to be complete in my plan, and it's making sure they're all 100% complete. That's what that bit of logic does. And if it finds any that aren't 100% complete,

where the progress is less than 100%, it puts out this message. So again, it's using messages.push and that will put it in that red alert that comes up.

So I know I'm flying through this pretty quickly, but I just want to show you that all the code is available to you. It's in the workflow. That last lesson was to automate actions. And so, what I wanted to show was on 3a, that transition up at the top. Let's go look where the code is for that. Do a quick time check. So the code for that is in an independent function, in a separate function called "Start Root Cause Analysis." So I'm going to go to my action scripts, and find "Start Root Cause Analysis," and let's have a look at what it does.

The main thing that I wanted it to do was I wanted it to create an object over in the Root Cause Analysis workspace. Remember that? We wanted that object to go through maybe a management review, or there might be some different people participating in that than were participating in the business decision itself. So I wanted to create that root cause analysis, and the function that does that comes from a library called Create New Item. And the function itself is called createNewItem. I think that library only has one function in it. And I tell it what workspace I want it to create a record in. And I tell it what properties I want it to contain.

And the properties-- I'm picking them up from the business decision that I'm on. Remember, I said I was accumulating some data. I had challenges and opportunities. I didn't put all of those in the code. But that was the idea when this was going is that you take some information, and you might massage it, or whatever, but then you pass it on to another process. In this case, I'm passing it to a process in another workspace. You could also think of passing it to a process-- a different, maybe an external process. But the point is that this is a process integration. So I'm creating a new object in another workspace. It's going to go through its own lifecycle state. And I'd do that with just a few lines of code.

Then the other thing that I did is the last line here is kind of important. Because what it does is it takes this new root cause analysis-- Let me see if I can get you just a little bit better. Too much. So it takes this new root cause analysis that I just created, and it assigns it back on the business decision that it's originating from. And what that allows is I have a link on each page, one to the other. So when I'm navigating in my system, I can just click and follow those links. And it's as simple as assigning the object to an object field in the item details.

And I think that's essentially it.

AUDIENCE: Did that create a new item?

[? FRED SMITH: ?] Mhm.

[? AUDIENCE: ?] [? Library ?] script should be back [INAUDIBLE] Now, [INAUDIBLE]

[? FRED SMITH: ?] Right.

AUDIENCE: Right. Most of our default tenants are coming with that, so you can just [INAUDIBLE] [? It's like ?] [? you're coding ?] [INAUDIBLE]

[? FRED SMITH: ?] Right. OK. So I think I showed you all the code associated with this workflow, and tried to describe the behavior-- that the other opportunities for scripting that we described-- one is on an escalation. And this might be an example of an escalation where I want to execute some new code-- or the same code, maybe-- but I wanted to do something whenever I'm sitting in this Enterprise Down for too long. So basically, on a time-based event, execute a script.

Or you can have a behavior, like Tony described. I'll show you that just real quick. So on my business decisions, I can set the-- I'm sorry. Let me show you on the workflow, because they are on the root cause analysis, because there actually are some behaviors here. Behaviors. And here, I said, OK, whenever you create a new item, run this script, RCA_On_Create. Whenever you edit an item, or it's modified, run RCA_On_Edit. Not a lot of creativity here.

Or you can run a script on demand. And the idea of a behavior is it's code that executes outside of any particular workflow process. Because after all, when you create a new object, which workload is it participating in? You don't know. So these are implicit behaviors that have to do with the nature of the object itself, rather than any particular workflow process that it's going to be participating in. So that's the reason why you would have behaviors, as opposed to running script within a process. I mean, you can ask yourself if the question is, do I know what the process is? If so, you can go put the code in the workflow process. If at any given time, you say, I wanted to do something, but don't know what process is running, then consider a behavior.

And I think that's it. I'll go back to the slides. There might have been a wrap-up slide.

MICHAEL VESPERMAN: Yep.

AUDIENCE: [INAUDIBLE] Do you-- Just one question. You said that you have the action when it says [?

belong in ?] that state. Is that for the escalation?

MICHAEL VESPERMAN: Yeah.

AUDIENCE: There's an equal in my page that I have to select language description [? to execute in ?] [INAUDIBLE]

MICHAEL VESPERMAN: Yes, so escalations are actually applied in the workflow editor, when you're actually on a state. So you actually define it on a state, and then in there, there's actually the ability to call a script with an escalation. Have you ever-- Do you want to do that real quick? Just do it really quick.

[? FRED SMITH: ?] Sure. Where did you want to go?

MICHAEL VESPERMAN: Go to the workflow editor real quick.

[? FRED SMITH: ?] OK, oh no, it's on the backend. Let me find that right here. Workflow editor. OK.

MICHAEL VESPERMAN: And then just click on-- I don't know. Yeah, you can [? double-click ?] on that. And then--

[? FRED SMITH: ?] Escalation.

MICHAEL VESPERMAN: There you go.

[? FRED SMITH: ?] So this is an example. I'm running this script every day. And that's escalation within a process, obviously. So it's not what I was calling a behavior; it's really a workflow process.

AUDIENCE: OK, so it's actually executed server-side and then do something?

[? FRED SMITH: ?] Mhm. They're executed server-side in order to calculate the user interface. And that's an important thing. A lot of subtlety in that statement. But that's the important thing to understand. Yeah, they're executed on the server. And so one thing to understand about that is that if you went out to Google and search JavaScript-- because you know it's got this one function in here-- but if that function doesn't apply to the server-based object-- If it generally applies to the client-side document, then it's not going to be enabled. It's not going to work, because all of this code is actually executing on a server, like you said.

MICHAEL VESPERMAN: And you could do things like change the status of the record if you have some status drop-down, or something. You can add a number of days in escalation. So you just look at the date, or look at the number of days; add 1 to it. All kinds of different things. Send an email, obviously, is another one, right? So those are the opportunities you have.

[? FRED SMITH: ?] Yeah. I did include a library in the code that provides four really convenient email functions to help you either email to an arbitrary address. So it could be someone even that's not a user on the system, or email to someone who is a user on the system, or email to a group of users on the system, or-- I don't know, there was a fourth variant of that. But it's a convenient email library. So you can look at that code that use it sometime. [INAUDIBLE]

MICHAEL VESPERMAN: So, one, you've probably seen this on every single one of your classes you've been to-- please fill out the survey. By all means, we're just looking to provide really good classes at AU and understanding. Teaching scripting in a class on the last day of AU can be-- and I can't even believe you guys are all sitting here and doing this, but hopefully we didn't bore you. Part of this is get you familiar with this. And that handout and that source code that we gave you could really help you, right? Hopefully you didn't find this too scary.

I mean, a lot of times, this is kind of a sensitive place. I'm kind of in a presale side of things. And so we don't like to talk about all this programming you have to do in Fusion Lifecycle, and, oh my gosh, you won't be able to use it. The purpose of this is really about how we can really drive some serious power into these workflows. And that's the goal, right? Not to terrify you. Hopefully you guys didn't find this too scary. Did this seem complex, or did it seem pretty easy?

AUDIENCE: Pretty easy.

MICHAEL VESPERMAN: Pretty easy, right? And, again, it's just about little bits of code. But we're out of time. This is our first time teaching. So we didn't really think about how much time we should allot. We certainly didn't think about what should we not try to be lined up as the same time. Keynotes are a bad thing to actually schedule your meeting for.

But the three of us are going to go to the Answer Bar. So it's lunchtime now; obviously go get some chow. But we'll be there for the next hour and a half. If you guys have any questions, or you want bounce some thoughts or ideas off us, by all means you've got access to us for the next hour and a half, and--

[? FRED SMITH: ?] We'll probably sit here, unless we're kicked out. But then we'll go down to the Answer Bar, so--

MICHAEL VESPERMAN: You'll find us down there, too. Yes, [INAUDIBLE]

AUDIENCE: [INAUDIBLE]

MICHAEL VESPERMAN: Yeah. And we have a couple of thoughts. You know, I've been playing around with ideas about a lot of times when I'm engaging with a new prospect, and soon-to-be customer, I've been thinking about how can I empower some person who's willing to pick up some JavaScript and use this, right? So this course-- I'd love to just give. There's no reason why you couldn't share this with someone who has just graduated from university and is willing to give this a go. because you really would pick this up quickly. I was talking to Michelle about maybe trying to make this an app in the App Store and maybe be like a little training app that you could add to your tenant in your sandbox, or something. So there's some thoughts going around on how to take this to the next level.

All right. Thanks everyone for joining us.

[? FRED SMITH: ?] Very good.

MICHAEL VESPERMAN: You guys have a good trip home. Thank you.

[APPLAUSE]