KEN MARSH: My name is Ken. I'm a senior software developer at P-tech Consulting Engineers. We are a engineering service provider, primarily servicing the pre-cast industry with engineering services, shop detailing, ticket detailing, engineering, and all that sort of thing for pre-cast concrete. That's where we come from.

And what we've been doing over the past 4 and 1/2, give or take, years is working through an implementation of Revit that works for us, and what that's entailed is taking what's available in the Revit platform, manipulating that to some degree in terms of creating our own family-type contents, and then building sort of a framework on top of that. And I'll show you why that framework "on top of that" is important to facilitating a true, efficient, workflow inside of Revit.

How many people came from AutoCAD days ? Anybody from the drafting board is? Me too. We were a little behind the curve. All good times.

Welcome, come on in.

So what we've been doing is working on implementation of Revit. So Revit obviously is a great tool, in the sense that a lot of architects who are working with Revit now and they're delivering Revit to the engineering staff in terms of attempting to communicate their design intent, so to speak. What we'd like to do, of course, is to be able to help facilitate estimation workflows and shop-drawing workflows and all that sort of stuff.

From the AutoCAD days, you realize that as you put together drawings, it's kind of a tedious process, right? You have to do the same things over and over and over and over again. So what happened with Revit? Well, Revit brought us a lot of functionality in terms of allowing us to utilize a central database of information and extract information from that, so the views are live views, the schedules are live views, we'll talk about that in detail in a bit.

In order to do that, you will find out that the Revit workflow for doing this type of shop drawing creation is also a somewhat tedious process, and so in order for us to really fully implement Revit as a platform for engineering and documentation we wanted to increase the efficiency, and so we did a lot of programming on top of Revit within this framework that we've created to do that.

So what we're going to do is we're going to talk about assemblies so let me just give you a quick introduction. Ken, structural engineer in the past, software developer now, still recovering. Shannon Cooper is our lead draftsmen, been manager, and he is support for the software stuff, so whenever anyone has questions or can't remember how to do something, they call Shannon and he's pretty good at figuring out how to do that stuff. Next slide.

I assume that since you all signed up for this class. I've never had one, it's like power.

I assume you've all read that at some point or at least glossed over it enough to realize that you wanted to come and sit in this course. I'm not going to go through and read that.

What we're going to do is at the end of this class I want you to understand how assemblies work in Revit, know what their purpose is, how they're used to drive the entire shop drawing or ticketing workflow. I want you to learn how to use schedules and leverage some of the advanced features of schedules, and how we can add information to the schedules, format schedules, group schedules, and total up information that we need for shop ticketing.

And then briefly on View Templates. View templates are an incredibly powerful tool inside of Revit. How many people are using Revit actively today on a day-to-day basis? Okay, six, give or take seven. Anybody using view templates right now? Seen them? OK. Pretty familiar. So we'll kind of gloss over that a little bit, but they're pretty powerful. They allow us to do a lot of enforce consistency across our drawings and that sort of thing. And we can utilize those and leverage them in our shop ticketing phase, as well.

So, use view templates and then schedules again for the last part. A lot of things in Revit are about schedules and utilizing that database, which is a 3D database, and it's not just the geometry, it's all of the data that's associated with those pieces and parts via parameters. So you can add and remove parameters as you need, so if you want to track some additional piece of information about a part or piece, whatever you want to call it, you can do that at any time, so you can add that data, you can fill that data out, you can add that to your schedules and you can have all that information available to you for quantizing or material takeoffs or whatever you want.

So we'll dive right in with assemblies. I want to talk a little bit about assemblies and what their strengths are, what their weaknesses are in their current workflow, and then we're going to do a little demo of how to create an assembly the manual way.

Assemblies, they support fabrication documentation. And the way that they do this is by taking a small set of information, so you've got this entire Revit model, which is full of pieces and parts, and what you really want to do in a shop take is you want to focus on one particular piece, right? So an assembly is a way for us to say, these things are part of this assembly, and they are the thing I want to focus on. So an assembly in Revit is a container, it's a logical grouping of elements, it's not a thing, it's not an element, it doesn't consist of physical geometry, it's an idea that wraps up a set of elements. So we have elements that are members of an assembly or not members of an assembly. And then assembly membership defines whether they're included or not. So identify class by document, you can combine any number of model elements.

So in a pre-cast piece, what do we have? We've got the concrete, that's great, we've got rebars, we've got the wire fabric, maybe, we might have embeds of some kind, we might have lifting components and hardware. All of those different things are different categories in Revit, but they can all be added to the assembly.

So it also allows us to tag schedule and filter by these things, so we can treat them as a thing via this logical grouping, this container grouping that is assemblies and each configuration of assembly is a unique type. So in Revit, assemblies or quantified or classified by type and Revit is very specific, we'll talk about that in a second, about what things match each other as an assembly. So it facilitates us placement and logical grouping of elements. It also creates the foundation for Shop Drawings, so Shop Drawings are created in Revit from an assembly, so no assembly, no Shop Drawings You could create your Shop Drawings by creating a view that filters out everything from the view other than the things that you know are part of that assembly, you can create that view, you can put it on a ticket, you can manually do this process but the assembly allows to group these things in a logical sort of way.

In the Revit workflow or in the Revit mindset, when they design assemblies, they also wanted it to be the thing that allows you to know that they're the same. So if we put the same components and pieces in, Revit can know that they're the same. There's a problem with that, though, it comes with some strings attached. So how are assembly types differentiated? Each unique assembly is a new assembly type, and we'll look at that in a minute. So they show up in the project browser at the bottom as a category of things that are organized in the project browser, so each assembly that you create gets its own place in the project browser and all the things that are associated with it, Shop Drawings and schedules and all that stuff will go together. But any edit that changes that will create necessarily a new type because it's no longer the same type that it was, it's geometrically different. And that's important, right? We don't want two things that are geometrically different to be called the same thing in our shop tickets, we don't want to wind up producing two of the same thing that aren't supposed to be the same, right?

The benefit is, if there's another one in the whole project that is geometrically identical to this one, it will match it and give it the same name, and you can't change that, that's an ingrained Revit behavior. So it's good, and it also has some catches. So what constitutes matching? Assembles can be associated with different categories, everyone that's using Revit should know that we have different categories in Revit, structural framing or specialty equipment, or rebar or whatever that category happens to be. An assembly can be any one of the categories that it contains, so if assembly contains a structural framing element and a specialty equipment element and a rebar element it could be any one of those things in terms of its categorization. So the category has to be the same, the naming category has to be the same, and it has to include the same number of elements. So if one assembly has three plates, the matching assembly needs to have the same three plates, they have to be the same family and type, they have to have the same geometric configuration, so if the plate is 4 x 6, it needs to be 4 x 6 in both places, and it has to be geometrically identical. So all those plates have to be in the same location and have the same size.

Any questions about that, right now? Pretty straightforward stuff.

There are some strings attached with the standard Revit assembly. The first one is, changes don't propagate. So if you create an assembly, think you've done a good job, pop that in all places that you think it goes, and then realize that you need to change something, change management is a problem because, unlike groups, has anyone used groups? Yes. Groups are kind of fun, right? They propagate changes, you edit them, once you change the group, all the instances of that group change. Assemblies do not work that way, if you change an instance of an assembly, the changes do not propagate, you get a new type. So now where you had DTA 02, you have something else because it's not the same, as you ought to think, so then you have to go and change all those other ones, or replace that all around the model wherever it exists. Views that we create from an assembly, so these are going to be what we're going to call the Shop Views or Ticket Views, the views that we create from that assembly are associated to an instance of that assembly, so if we have 50 instances of the assembly around, whatever one we used to create the views, those views are attached or associated with that assembly. Why is that a problem? Who cares? It's a problem because if you change that assembly for some reason, have to add an element to it, want to change the rebar size, whatever it happens to be, that forces you to delete those views or it forces you to stop in the middle of whatever you were doing, go find another instance of that thing and what they call acquire views, I thought this was insane.

So I used to work for Autodesk, I used to be a Revit QA, quality assurance analyst. So when we were developing this feature, I went "why are we doing this? This is crazy". So you'd have to find another instance of that assembly and acquire views, which would mean it would take those views from the one that you want to change and relocate them. We associate that with a different instance of that assembly. I don't have an explanation for it, I don't know why, it just is. I didn't understand it when we were designing it either, so it's a little bit unique.

Strings, right? These are challenges to utilizing assemblies for what we want to utilize them for. No tolerance. If that plate is 1/256th of an inch different than another instance of a plate, they are not the same assembly as far as Revit is concerned. It's really, really particular about this for some reason, but they wanted guaranteed geometric identicality, so no tolerance is available.

And we know that as you're modeling things, it's very difficult to make sure that things are in the same exact place and all the different places that you put it. Even if you're really, really careful you can still get these often and we don't want to deal with that, we'd like to have a little bit of tolerance.

Not all types of elements can be included in an assembly. Some of the more complicated elements, like framing systems that grow and shrink as you change the size of it, it adds more members or subtracts them, things like that, a little bit more complicated structural elements and annotations and some other thing. So here is the non-exhaustive list of all of the things that cannot be included in an assembly. By and large, this is not a deal breaker for pre-cast workflow. Probably we don't care if the stairs can't be included in an assembly, maybe we do, but they can't unless they are a particular type of stairs.

So there's some give and take here. There is little bit of gotcha in the sense that all of the things that we might want to include in an assembly, we can't necessarily include in an assembly. So that's a little bit of a gotcha, but for most of the work that we do, in terms of our standard type of building that we typically produce, pre-cast parking garages or precast buildings, concrete double tees, standard sort of stuff, this doesn't hold us back at all, but it is a gotcha.

So now we're going to flip over and Shannon, my copilot, is going to take us through creating some assembly views and I'll do my best to narrate. It's hard to keep up with him because he's so fast. I'm kind of kidding, but not really.

So what we're going to do is, we're going to start with the process of creating an assembly. What we tend to do is, we start with selecting the main thing that we'll kind of, in my opinion, anchor that assembly. So we're going to start with the structural framing on the double tee in this case, and you notice that once we get that up on the Create tab, there's a Assembly Creation button, or Create Assembly, and once you click that, you're going to get this dialogue. Remember I was talking about this? This is where you can give it a name and this is that naming category that we're talking about. OK?

So because I only picked one thing, I only got one naming category to choose from. Had I picked several things, and I could, I would have had an option to choose different categories. So our workflow is presuming that we've picked the structural framing element or the structural framing category as the name and category, and we're going to give it a name like DT, whatever, and we'll hit OK.

We also know that in order to, what I think of the assembly, as this is everything that's going into the casting, but I've got to have one half of the wing wangs, I've got to have flans connectors, or what do you want to call them, I've got to have the embeds, I've got to have the plates, I've got to have rebar, the welded wire fabric, I've got to have all that stuff. So if you create an assembly before you've picked all of the various things, you can select that assembly and then from right here you can choose Edit Assembly, and go ahead and add any of the things that you need additionally for that.

So what we do at P-tech to facilitate our modeling process is that we add these. So these plans connectors, you will notice, have two pieces, so he's tabbing in to get the nested component. So we create our connection families as one large thing, so as you get over to that the T-gutter, this is the wall panel connector, you'll see that this is one family, this is one the whole family here and it's got nested families inside of it, so we will tab in and grab the part that's embedded in the piece that we want to include in the assembly. That just allows us to be more efficient with our creating connections. So instead of putting an embed plate in the double T, putting an embed plate in the wall, trying to locate those in the same position, the same location, and then add the stitch plate in-between, we put them all in as one thing with nested families so that they can be quantified later.

So that's a little bit of efficiency enhancement on one side, but if you're going to do a manual process of creating these assemblies, you will have to tab in and get the piece that you really want. So as you can see, this is a quite manual process. You can speed this up somewhat with some crossing selections and some box selections, you can be a little bit smarter than he is about, but I really wanted to see what it's like. This is a pretty simple model, right? Our models are much more complicated than this, in general. So being able to tab in and see these things, scrunch your view down so that you can see what you're looking at and not get confused with other elements, because it can be easy to pick the wrong elements to pop in there.

So once he's got that mostly done, you can finish that. And now you see that as he hovers over that, you'll see that all of these pieces and parts are now considered part of that assembly, and the way Revit's selection works is, once you've made it an assembly, the top level thing that it wants to select when you hover over it, is the logical grouping of things that is the assembly.

And assemblies have a few properties like the assembly mark number, assembly control number, so it's got some data associated within itself, as well as the pieces that are within that logical grouping. If you wanted to get back down to your double T as you hover over it, you'll have to tab once before you click, Revit people know this intrinsically, my hand is on the Tab key the whole time I'm working, but that's the way that you get down into the individual pieces of this.

So now that we have an assembly, we have all of the things that go grouped together, now we're in a position where we're ready to start creating Shop Drawings from this, and one of the key components the Shop Drawing creation is, Revit automates the process of Shop Drawing creation, but it wants to know how you want those drawings oriented. So they provided a tool that's called the Assembly Origin Tool, it's a set of axes that you orient, and what that does is it allows you to set up the box, so imagine a conceptual box around this thing that you want to detail or document. However we orient that assembly is going to orient that box, so we could orient it skewed, orthogonal to this member, our standard of course, and I assume most everyone standards oriented orthogonally. So as he selects this assembly, and you'll see that up here when we go to Edit Assembly, there's this little thing. This is what is called the

Assembly Origin Tool, let's call it. Assembly Origins is what it really is. And when you select it, you get three axes, is the right hand rule, we have the red axis that represents X, green, which represents-- he's done it now, green, which represents Y and blue represents Z. So that's going to orient this whole box around the element.

So what we like to do is have our standard top view come out looking on a double T like a top view, so what we want to do is orient the x-axis along that top long distance, long direction of the element, so that when the shot runs automatically created for us it's already in that orientation. We could adjust that if we left it alone. We could adjust to change every one but it's nice to have it oriented in a way that we like, so to speak. So once you finish that, let's sort of re-orient that box.

And now, up on the contextual tab is Create Views so that's creating new Shop Drawing views and associating them with that instance of the assembly. OK? So in this dialog we have scale selection, it doesn't give you any help, really, you have to sort of guess. We have a lot of intuition about what to use, right? You know about how long the member is, you know about what scale you typically use, but you can adjust that or use a custom scale if you need to. So from that drop-down list you could also use a custom scale and type in your own scale factor, if you needed to. That was kind of a crazy member that wouldn't really fit nicely on our sheet, or something like that.

So you can choose any set of views. This is the 3D ortho view, plan views, sections, we have two sections through the thing. This is a fixed list, there's nothing else we can do about it, it is what Autodesk delivers for us to utilize. There's a cross section and a long section, so to speak. Top views, side views, front views, left views, back views. And there's also some parts list, so we've got the automated parts list the material take off, we're going to include those, and we're going to show you how to modify those to get additional information included in them, if you should want it. We can also create a sheet at the same time and choose from available sheets or title block, I shouldn't say that, you create a sheet with a title block specified.

So we've got top, and a right, and a front, and a parts list of material take off. That's kind of the stuff that we want for a double T template. I want to keep an eye on the time. I have terrible tendency to run over because I just love talking about this stuff.

So when it hits OK, Revit is going to automatically create all those drawings for us. So here's our top view, remember I said that assemblies wind up in their own category down here in the Project browser? So there's my DT 1000 I created, and all those views that I asked it to create are right here, including the sheet, parts list material take off, and my views that I want include on the sheet.

So in order to start building, you'll notice that my sheet is actually empty, so it didn't really help me that much other than the fact that it created those views, the next step in the process is to start taking these views just like other Revit views, any of your general assembly views or direction drawings, that sort of thing, and start placing around the sheet in the way that you want them. Once you've gotten those placed, one of the things that we needed to deal with is, most of the time we wouldn't want to show this in this orientation, we want to rotate it into sort of look like it's side view of the thing. One of the ways that we can do that is by the rotation on sheet parameter over here, so if you select that view, you want to not select the view but the view port, so to speak. You can use non-counterclockwise or clockwise as a rotation. That's great but here's the catch, if you do it that way and you put the title on, it rotates the title with it which is a little bit odd, maybe or maybe not the way you want it to be done.

There is another option. So another way around that is to select that and turn on the crop region. You can actually rotate the crop region of a view, you can do this with any Revit view, it gets a little wacky. But you can do that, change the rotation and notice that he put it in a clockwise 90 degree rotation and it actually rotated the other way. I still have not figured out why it does that but it does. Be aware of that little thing.

So you can do that, and then if you turn the titles on again, you'll see that they're still on the bottom where you would expect them to be. That's a little bit of a trick for dealing with rotating views on the sheet and keeping the titles if you like titles in the right place. So you notice the other thing, this isn't a very helpful view, right? This isn't like the kind of information that we really want to show in a shop ticket. We want to show the rebar in there, we want to show the embeds, we want to be able to dimension those things as well. So what we're going to use to do that is the View Templates.

If you select that view and go to-- let's show how we would do this normally. So we could select that and go to Visibility Graphics, this is where you can override any of the behaviors of that view, so we could maybe take structural framing or override that structural framing, let's give it a 12, be bold, and maybe change the color to blue or something, it doesn't really matter. Hit OK and OK, and OK. So you can see that we've changed the effect of that view the way that it's visualized currently by overriding those parameters. Now, we could take that information and we could include encode that into a view template. So we can take that view, we can save a view template of it, or we could set up a custom view template and just configure those.

The View Template tool is part of this view, it's down here it's currently set to none, but if we click that right now we get taken to the View Templates dialogue, or we can choose from our pre-configured View Templates, or we can create a new one. So we can duplicate one and change the parameters. These, we can choose what things we want to include in that view template. So for Shop Drawing views, we want to be able to configure the scale independently so we leave that unchecked so that it doesn't overwrite the scale for our views. But if you did stuff where your scales were always all the same, include it, it will just get applied automatically, it's no big deal. But ours we leave it unchecked, and here's where Visibility Graphics overrides the model, so click that button for me, this is where you can override all those settings, so you can choose what all of your view settings are for that view and apply them via View Template.

Once that's applied, it controls the view, you can't change it, which is a good thing. So we don't want the views that we want to put on sheets to be changed ad hoc or accidentally.

This is our standard double T top View Template, so to speak. Sets up some nice line waits for us and shows us the stuff that we want to see. The next step in this process, of course, is to go and dimension all those things. So we're going to pop the dimensions on, we're going to string them out, we're going to put the names of the plates that we're dimensioning or the bars or whatever it happens to be. Then we can go ahead and apply those other View Templates to the rest of those views.

And this is the setting up of View Sheets, or views on that ticket sheet. So all of these steps have to be repeated for every single assembly that you're going to document. It's a little bit intense.

Next thing I want to do is talk about schedules. So schedules in Revit are just another view. Question.

AUDIENCE: [INAUDIBLE]

KEN MARSH: Which one? Dimensional? Yes, manual process, as well.

You know, Revit dimensions are pretty good and one of the things that I think is great about Revit dimensions is, if something changes, the dimension is smart and it stays attached to the references, so if the references change, the dimension string changes. So there is that advantage of Revit in the sense of even if you're doing that work, it's not for not necessarily. We're not going to go through that process because that's literally just a manual process at this point and it's something that all of our ticket cleaners know how to do.

Next step in this process. Let's get into the schedules, let's pop the schedules on here, and we'll start to talk about schedules. So out-of-the-box, what Revit creates for you automatically in terms of the schedule, it's not quite well formatted, this isn't necessarily something you'd want to put right on to a shop ticket. It's enumerating every single instance of everything in the piece. If you zoom in, I think it's including the structural framing as a piece that is part of the assembly and that's kind of weird because obviously the pre-cast is the whole thing.

So what we're going to do now is we are going to edit the schedule. I'm going to show you some of the inner workings of schedules, so to speak, or new things, hopefully that you don't know about schedules or we're always curious about. This is the Edit Schedule view, and when you go over here to the side, you can access the information about that schedule. So what he's done is click Filter first, but these things just map to the things up here, I don't know why there's not just one button that says Edit, because it all is up here in the top. So what we're going to do is we're going to set the category does not equal structural framing, I don't need to see the structural framing element in my piece, but I will quantify that later in my material take off. So I'm just going to use this filter to apply a little dumb filter to take that out of the view. So, like I said, schedules are just another view of the data in the model. So if I want to change what's shown, I need to filter it in some way.

Other things that we want to put on here. What else do we do with a schedule?

AUDIENCE: Sorting and grouping.

KEN MARSH: Sorting and grouping. I'm going to sort and group on family type, and instead of itemize every instance, I am going to uncheck that and hit OK, so you can see the differences. So now I've just got all the things that I have plus their counts, which is pretty helpful, that's a much nicer schedule.

AUDIENCE: [INAUDIBLE]

KEN MARSH: No.

AUDIENCE: [INAUDIBLE]

KEN MARSH: You're jumping the gun. But yes, now that you've spilled the beans, it's a manual process and so everything that we're about to do, you would have to do it to every single schedule on every single ticket that you produce. This is a pain.

Let's talk about some other things that you can do schedules. We don't need the specialty equipment category, so we can go to Formatting, and for category we can just make that a hidden field, we don't need that in there, so we can leave that out, now that we've got some more information that we want. Let's pretend, I'm sure that nobody does this but let's pretend for the sake of argument, that you like to include a cost with all of your pieces and parts. I'm going to go down and grab the cost field out of my parameter list. So this is the list of all of the parameters available to you to schedule. You don't have to choose the parameters that rabbit gives you, you can add or remove any parameters to that schedule that you want. It's just going to show you that data. We're going to put in a cost, but I don't want to put in a cost that's my cost, so I'm going to put in a cost plus a markup. So what I'm going to do is add what we call a calculated value. We're adding another column to our schedule to display information that we want to display, and we can do that with a formula based on the cost and I'll just make it 10% more, a little extra money in my pocket, and we go to Formatting, and we don't want to see the cost field, we want to see the estimated cost, and for estimated cost we want to calculate totals. If we do those things and hit OK, now we get totaled up the total cost for each of those things, I can see what the total cost for that ticket is going to be. We could also do totals for the entire thing, so if you go back to sorting and grouping, grand totals, totals only, hit OK, and then I can get a total number for the whole thing.

So you can imagine utilizing this schedule for all of the pieces that you've modeled, adding some costing information, you don't have to do this just for a ticket, these schedules can be created for any view, any category. So anything that you want to group and calculate fields for and total things up, if you put them together and put it in is a template or add it to your Revit template, then it's there and available for you.

We've got a schedule that's a product schedule and as we're adding double T's, it's just automatically filling that up for us. Any time we want to look at it, we can look at it, see what our estimated cost is, or estimate total concrete is, or whatever those things happen to be, we're in a good position to be able to immediately extract information once we've modeled it. This is, a Revit template as opposed to a View Template.

We can take that and move it into position, he's got the columns sort of nicely lined up with stuff maybe, or he's going to have to line those columns up and adjust them in some way. Maybe move it into position, wherever that is. So kind of a manual process.

The next thing going to look at is, we like to put a total ticket weight on our tickets, so how would we do that? We could start with a material take off tool. So we've got this material take off that's been automatically generated for us, it's got the volumes of various materials. You notice that all of my rebar is 0000 cubic yards, well of course it's 00 cubic yards because it's a really small amount of rebar relative to the cubic yards. So we could change the way that this is displayed in terms of precision, so we can go to formatting, and change for material volume. We can change the field format. Instead of using the project settings, we could put in a custom format and string that out a little bit. We could change that to feet, cubic feet or cubic yards, whatever you want to do. We could change it if you want and hit OK. And now you see that we've got some amount of volume registering for that visually, OK? So you have control at the field level, how you want to display that unit information, you don't have to use the overall project settings for your unit information.

The next thing we want to do is, we need to add in our unit weight field, so we need to know how much the stuff weighs per unit, since we have the units now, so we've got a weight per unit parameter. That's a parameter that we created and added to Revit and we associated it with all the elements in the project. So any element that I want to have a unit weight for I can pop that parameter and we filled it out ahead of time so that it's got the right things in it.

Let's get to weight per unit. We're going to add a calculated value, and the calculated value is going to be weight and we're going have a formula for calculating that weight. So we're just going to use the volume of material, we're going to multiply that by our weight per unit that we've already encoded in there, and let's see if you can do it. I had them put on another sheet so you could paste it in. We've taken the thing, we've done a little bit of conversion because our weight per unit is in cubic feet whereas our volume is coming in cubic yards, so we just did a little calculation there. We hit OK, and then hit OK again before we do any more formatting. So you see I've got my weight per unit, so I've got 490 in there for steel and 150 in there for concrete, so we're calculating a total weight for that piece and a total weight for the rebar. So we could total up those, gives us some more valuable information.

If I want to put the weight on the ticket, unfortunately it's sort of a braking process, so Revit is great, it keeps all the things together, it keeps all the data consistent, but if I want to put that information onto my title block, I've got to copy it manually into there and put it there. It breaks the Revit workflow a little bit because if I change this piece, and this will change automatically, but this will not. So we've done some things to help automate that process to some degree, but just note that copying that information, instead of listing it here, listing it in the title block in particular, it puts a gap in between those pieces of data, they're no longer linked together.

So that's a little bit about how we can utilize schedules, utilizing hidden fields, calculated fields to create new values or fields or columns of data that we want to show based potentially on calculations that we've done on other fields in that schedule. Hiding fields, grouping and totalling, and what else did we cover? I should ask you. Pop quiz. Any questions about schedules right now? Anyone bored to tears? That's good.

What we're going to move to next is creating an assembly. We talked about that, we've got the sheet, we've got the schedules on the sheet. We're 40 minutes in, I do tend to go over.

What you've seen is the out-of-the-box, give or take, Revit workflow. These are the set of tools that Revit gives you to create assemblies and create shop tickets and all that stuff. So remember, in order to create shop tickets we have to select every piece that goes into the assembly, we have to then create the assembly views, manually place them onto the sheet, manually reconfigure the schedules, we're probably not going to put calculated values into every single shop ticket scheduled that we do, but if you wanted to, you would have to manually do that for every single one of them. You would also have to manually adjust the column widths, because the column widths come in the same way every single time, so you have to manually adjust those for every single ticket. Pretty tedious process, so we've just spent 40 minutes working on this, maybe 10 minutes talking. The rest of it is just manual processes of adjusting these things and getting them set up. So the question that we had at P-tech is, this is great, we want to be able to use Revit, we want to be able to take information in from the architect, how do we get efficient with this? That was the question that we asked ourselves.

So tools that we have available, nowadays we have Dynamo but back in the day we didn't have Dynamo, there was no Dynamo, so we started developing tools on top of the Revit API to help us facilitate these workflows. So that's why I'm a software developer instead of a structural engineer. So most of my job is to help make our team as productive as possible and then we wanted to share that with the rest of the community. One of my passions, in particular, having come from a structural engineering background, is helping our entire industry utilize and take advantage of these technologies that we now have via Revit and Autodesk and all the integration that they have. So that's sort of a little bit more insight on my background.

Obviously, grabbing and selecting all these little individual things is kind of a pain in the butt. So one of the first tools that we we designed or coded was, let's take the fact that these things are embedded in that concrete piece and leverage that fact to identify the things that should be included in the assembly. And so instead of going and picking, we have a tool that's called Assembly Creation, you just click that tool by selecting the structural framing element and that's how fast it is, just like that. So no more clicking and picking and tabbing, and all that business. We just utilize the fact that all those things create a geometric interference with that structural framing element. So, in modern day, you could utilize Dynamo for the same thing. So you can leverage a Dynamo script to identify those interfering pieces and collect them up into part of that assembly. So we've just automated that process to make it really quick, click the piece, click the tool, done.

The next thing that we want to do is orient the assembly origin. We've seen how to do that manually by creating the thing, editing the thing, orienting little axes around, we created a tool to make it easy because we've got some people that don't know how to do it, and so we created a tool for it. So we click that tool, we isolate the assembly. This makes it a lot easier to pick the things that we need to pick so we don't have all that extra congestion of the other model pieces around. So we just click the top as cast face, the mark-in and the right end, and that just helps us orient that assembly. It's pretty straightforward. And we get a little graphic, that was my idea, to put the little graphic on there so you can see what it was. And if you highlight that and open it and select it, you will see that the assembly origin and has been moved to that same place. Is that a real efficiency answer? No. If you know how to manipulate this assembly origin, you can be pretty quick with it, but if you can do it really quick with graphical editing, that's kind of easy and fun and you get a cool little graphic.

So once we've done that, we've got to create the assembly view. So what we're going to do now is I'm going have Shannon go through and create a shop ticket. We're going to lay it out, I'm just going to let him kind of churn and burn on that, because he's really good at it. One of the few things. I'm going to let him do that. He's going to apply the view templates and then, when he's done with that, he's going to add some, I think we have a note that will place just to sort illustrate this fact, any legend views that we have, so we have sort of standard details or standard notes, or standard anything we want to include with this ticket, we're to lay that out on top of that ticket as well because what we're going to do next is we're going to create a template from that. Once we've made that lay out, why can't we just take that information that we have, in terms of the number of views, the types of views, the orientations of those views, why can't we just remember that and then use that to make the next one? So that was the thought behind this whole thing.

So I'm just going to add those, you've got to have a note, created the templates, applied the view templates already. One thing that we then did, this problem of the schedules, you'll see that, highlight those things again. The problem with the schedule thing is that all the adjusting of the widths of the columns and adding fields, and calculated fields, and adjusting the formatting, we thought there was a colossal pain in the butt, however, there is no way currently to create a templating thing for that schedule view to have it automatically placed in the way that we want it. So we had to sort of rewind a little bit.

Was anybody here yesterday for the Manufacturers Fabricators Forum? One. Did you see Marius' talk? It was kind of crazy. So his whole point is kind of like there's a mindset to sort of venturing into this 3-D world, which is, there is a way to do it if you have the will to do it, and so this was kind of a big stumbling block for us, because it's a colossal amount of time wasted if you don't do something about this. So what we chose to do is create some "schedule templates" and what I mean by that is just something we call a template, it's just a schedule, it has information and then we're going to do some stuff behind the scenes. Remember I was talking about this framework that we kind of put around stuff? One of those components of that framework is what we call a Construction Product Host, we automatically write it. So when we created this assembly, So this thing, wing wang, is there a real name for that? I did structural steel, I didn't do precast when I was a structural engineer. I called it wing wang. So in that wing wang we've got a construction product host, you see that's DT04 so that's its parent. So in that same process that we were locating things within that precast member, we also took time to pop that DT4 in there, we can leverage that in the schedules.

So what this next tool does is, we go there, start and duplicate schedules, so it's going to go and grab the schedules that are appropriate for this and he's just going to be able to automatically place them. So see, this is the DT04 parts list, it's just a copy of that schedule, and it's already formatted the way we want it, it's the columns the right width and everything. And if you go into the schedule and go to the filter, you'll notice that right here we have this construction product host at the DT04. That's the magic. There's no magic, really. So we've just done this behind the scenes. We've automatically set the construction product host, then the tool duplicates the schedule and it adds this DT04 filter to that schedule. That's how I get it looking and containing only the information that's part of that assembly. So it's just a little bit of sleight of hand, right? Again, in a modern day, it's nothing that you couldn't necessarily do with this fancy Dynamo script if you had time to write it, but that makes us a lot more effective and efficient, so being able to click that tool and duplicate the schedules and set that filter automatically makes it so much easier, makes it a lot faster. So that's one of the reasons why we developed that tool.

Now I have a model ticket. I've got some extra information, views that I need, rotations that I need, and the schedule that I need. The next thing that we did was create something where we can just record that. So lets record it by going to the admin and we will create a template out of that. Create a template, we give it a name, something, we pick an alignment for the building materials, were going to line that for the bottom because we want it located from the bottom to the top and we're just in a stack those. So if we had multiple building materials, maybe we have a little bit of wire fabric in there, or c-grade schedule in there, or whatever it happens to be, we could add those schedules in as well automatically. So he's just going to create that template, just going to save them into a template file, pretty straightforward stuff. The magic is what happens now, because now that we're in a situation where we've sort of pre-configured everything in terms of utilizing this ticket template saving, having our tools that do all of the schedule manipulation, and that sort of thing, all we really need to do is click the thing, create the assembly, set the assembly origin, and then all we need to do is select that and go to ticket tools and take the populator. And so that's the final solution on that. That's the template we just created, you can adjust the scale if you need to, but that comes from the template originally and click populate. So that's just the tool that we made that makes us really efficient, a whole lot faster, right?

The crowd went wild. Did it do it? We did a rehearsal earlier and we were pretty excited.

So you can see how efficient this can be on top of Revit, but in contrast to what the out-of-the-box tools are, so if this is the way with a lot of Autodesk tool-sets, so Autodesk will deliver some new functionality, they usually get about halfway through it and then they'll get sort of distracted by some bright shiny object, you noticed that, right? Eventually they'll get back around to it, potentially, and I think a lot of times they see the kinds of things that we do to make ourselves more efficient, we're in pretty close contact with them. You were here for the manufacturers forum and you're already using Edge. OK. People are already in the hook line and sinker. But you can obviously see that if you want to implement Revit for a full-shop drawing workflow, it's a pretty tedious task out of the box. Efficiency add-ons can help that, there are several different add-ons available, so AGA CAD is doing stuff around this.

Edge for Revit is our sub-company, so to speak, it's our software development arm, if you will, because we found that, while these things make us more efficient, I don't think we have a mission statement or anything, but I feel like, and my sense is that, we want to share these tools with the world. And so if we can increase the efficiency and increase the ability to utilize and leverage Revit, I think we all see that it can be a much broader, much more applicable platform, especially for the precast industry.

We haven't seen a lot of adoption until we started creating some of these tools on top of Revit and people started to get the sense that Revit can be used for pre-cast with a little bit of help from some options. So these are just some examples of our shop tickets. This is the stack of building materials, here's a little additional schedule, the template that gets created with it, there's some standard details or mark numbers, all the dimensions still are manual, that's something we're working on, how do we take that to the next step, so we can automate the dimensioning on these things? And then it'll just make it a breeze, make it almost painless to create an assembly. So you can take advantage of all the sort of tools.

That's my story for today, that's what I want to communicate. Does anyone have any questions at this point?

AUDIENCE: [INAUDIBLE]

KEN MARSH: It will turn into an assembly.

AUDIENCE: [INAUDIBLE]

KEN MARSH: This is a key.

AUDIENCE: [INAUDIBLE]

KEN MARSH: Yes. Because we short-circuited it, and the way we did that was by only having one assembly as the recipe. Our whole workflow hinges on having one assembly that represents all of the places it's going to be. So what we do is, we use that structural framing element to show where that's going to go, we associate them by having the same control mark, so it's all keyed on that control mark. We pick one of those to create an assembly from, and that does those two things for us. The first thing is, I only put rebar lifting and handling components in that assembly, that's my recipe that describes what we're actually going to pour in the bed. In doing that, all the rest are just placeholders, and then I've got a bunch of magic behind the scenes that tallies up my quantities for me.

Let's go back to the shop ticket, because I'll address your other question too in a second. So you notice that this title block is also automatically filled out with all that information? There's the yards and the weight and I had to manually copy that before? I just automated all that same thing. So we've also got, we only have one.

AUDIENCE: [INAUDIBLE] I might have to recreate this.

KEN MARSH: No, you don't need to recreate it. No. But you do have to re-run the ticket populator. You go off-script and then it's bound to go wrong, but it all worked. Behind the scenes we figured out that there's another one structural framing element that has the same control mark, but we only have one recipe, that's our secret sauce, so to speak. So we're not modeling rebar at every single place that it exists, what we're doing behind the scenes, there's another trick behinds the scenes that we're doing, is the following. Can you go and select one of those rebars? In the assembly. This isn't going to work.

Everything is not completely automated yet, we just to show some manual steps in the whole process. You listen to the count multiplier. So basically, what that says is, I know that I'm only modeling my rebar in one place, but I know that it needs to exist in all the places that I have these placeholder elements. So the count multiplier is the thing that I use behind the scenes to stick that information in, so when I run what we call the BOM product hosting updater, it fills out that quantity count multiplier automatically for me and that allows me to drive my schedules with actual quantities of the rebar that I need. Has anyone ever given a presentation where you went off script and nothing worked? That's my life. But luckily it's all working, so count multiplier, three, so I know that I've got three of those same assemblies, I know that I need three of those same rebars. One of the things it does for us is it allows us to keep the model from getting too heavy.

Did you see Project BRAN yesterday? From Autodesk, their walls and thing dividers. So Autodesk is putting together some automation tools around what they're calling European Residential Construction, which IS walls and flats laps. There's no corbels, there's no columns, there's no anything like that, but they've automated a lot of those tools. They can take a Revit wall and break it up automatically and create assemblies from parts, so parts is something we haven't talked about because it's not part of our workflow at P-tech, but it is a viable or it's a potential workflow for also creating shop drawings.

I guess you could look at IDAT I guess they don;t even have any information anymore. They probably have some information. There were some pre-cast fabrication tools that they bought from a company called IDAT in Germany, and IDAT's whole thing was, let's take a Revit wall, which is a totally different element and break it up into parts, and then use those parts as our assembly creation. So we're basing our assemblies on loadable families, whereas Autodesk is basing their assemblies on parts.

Parts is a whole other story, we can probably talk for another hour about parts in general, but also powerful functionality for fabrication documentation.

It allows us to keep our model light, and it allows us to do this. I don't have to worry about the assembly changing because I only have one of them.

It's a little bit of how do we achieve what we want to achieve and work around some of these strings that come attached with assemblies? We also have tools, we've just started developing mark verification, so what we've wanted to do is implement a tolerance in terms of identifying things that are the same, so we've recently implemented that behind the scenes. We hope to be delivering that soon. But things like that will help us speed up our workflow as well by identifying things that are identical or identical enough for us.

Other questions about the shop ticketing workflow, the Revit workflow, assemblies, schedules? Want to play stump the chump?

All right, you are free to go. Thank you very much.

AUDIENCE: Thank you.