ROBERT ANGUS: Well, good afternoon. Thank you for coming. I hope you're having a good AU so far. My name is Robert Angus. I'm a Solution Architect. I've been with Autodesk for about 14 years now, and I work both in product from-- everything from inventor to the newest fusion product. And for the last three years I've been working in Autodesk consulting.

To begin with, I'd like to get a feel for what everyone's experience level is here with Forge Viewer. This class is mainly geared around how you can use Forge to visualize information within your enterprise. How many here have developed an application using Forge Viewer? Really, one. How many here have done any type of software development before?

AUDIENCE: [INAUDIBLE]

ROBERT ANGUS: What's that?

AUDIENCE: [INAUDIBLE] This is supposed to be Forge Viewer, virtual reality?

ROBERT ANGUS: Yeah, so it's using Forge Viewer to-- here, we'll go to the next slide. Page down. So it's using Forge Viewer to visualize data, and also how you can use virtual reality in that as well. So right now, you'll find as we get in this class that the VR capabilities of Forge Viewer are very much in their infancy but they're progressing very rapidly. And we'll talk about that at the end of the class.

How many are mostly here because they're interested in VR development? A lot of you. OK. Let's start a little bit about talking about the learning objectives. So in the class, we talked about-- our desire here is how do we combine data with engineering models to present a compelling story to our end user. And how the large model viewer, in particular, can be used and leveraged to facilitate that.

We'll look about our VR and AR options and see how-- what the large model viewer, which is really called the Forge Viewer now. They changed the name since I initially introduced the class. The Forge Viewer can support-- and talk about some of the impact it's going to have in the future, and how we can move forward on that.

So I want to start off a little bit talking about what the Forge Viewer is, and once you understand that, then we can better understand how VR can fit into that. Forge Viewer is a

WebGL-based, JavaScript library. You embed it into your web application.

It's used in conjunction with the Forge Derivative services to produce your engineering documents into a streamable format that can be downloaded to your web application. All of the Forge Viewer runs locally on your browser. There's no web services, no processing on the web that happens at all, when you're viewing an application. It's all done locally.

One of the great things about the viewer, is because it's web based it runs everywhere, most everywhere. It runs on any browser that supports WebGL. It runs best on modern browsers that support the higher JavaScript syntaxes, with more modern JavaScript syntaxes.

So in saying that, what I'm saying is Internet Explorer is a much less desirable option than Microsoft Edge. Edge runs better. It's easier to program against and probably the best support is using Chrome. So because it's browser based we can run it on most platforms, most OS'. I've listed the major ones. It should also be able to run on Linux.

One of the-- the thing that I think is pretty remarkable about it is how performant the large model viewer is with large models or the Forge Viewer, I should say. The modules load and compile very fast inside your application. It scales to very large models. And one of the reasons it can do this is because it's called a progressive renderer.

And the fact that it uses a progressive rendering format is also one of the biggest problems it has in dealing with virtual reality. And we'll talk about that in a little bit. So its ability to scale also makes it not an ideal solution for virtual reality. Also, the quality of the rendering inside the Forge Viewer is improving rapidly. If you look at the rendering quality six months ago compared to where we are now, it's pretty amazing to see the progress we've made.

As I said before, it's an embeddable HTML component. And that means it's, for those who know the HTML development, it's embedded as a component using a Shadow DOM, meaning that you don't have to worry about stomping on any of its name spaces. It's very transparent to use as a developer, you can just consume its functionality.

Because it's built upon the web platform it becomes a very collaborative environment. It's easy to establish and share sessions. It's easy to form chat or communication. Its' extensible nature, it has a built-in plug-in framework makes it easy to do things like annotate the model, and provide ways of using it as a mechanism to talk about design choices, design optimizations, or procedural information.

Forge supports both 2D and 3D formats. Both vector and raster. Well that means you can use Forge to view your PDF documents, as well as your AutoCAD drawings, as well as your Revit models, as well as your 3DMax models. It supports an amazing number of formats and that's continuing to Progress and involve. So when I say cool, what I really mean is it's useful or powerful.

Now in order to get an asset into the Forge Viewer, you have to run it through a data pipeline inside the Forge services. And we'll talk a little bit more-- we'll talk in a little more depth of how you do that so that you're familiar with that process, because the ability to create an application that consumes Forge has some very specific limitations on what data you can access and how you can access that data.

So again, summary of what we talked to before, its strengths are the fact that it's platform agnostic. It's browser based. It's [? pluggable ?] into your application. Has, I think, a relatively easy to use API that can be extended through its plugin framework. And it supports most any file format, and it does so with great performance and reasonably good quality.

So in order to do-- there's a few steps you need to do in order to use Forge to do any type of application. The first of these, is you have to register your application with Autodesk. And through that process you go onto developer.autodesk.com. You request a new application. You name your app.

And then it will give you , what used to be called the consumer secret and a consumer key, they now call it a client key and a client secret, but it's in essence, it gives your app the ability to identify itself with the Forge Services that you will need in order to get the data into a viewable format. So this is the first step that you need to do.

Now one of the things that I think it's important that you realize, is that when you're going to view Forge data, that data becomes your applications data. In other words, when you get a consumer or a client secret and a client key, what Autodesk Forge Services Is doing, is they're giving you access to your own private sandbox for your data.

And so if you have data that's currently living in BIM 360 Docs or in A360, your application cannot access that data, because as a security measure Autodesk is sandboxing one application from another. So the process you need to use in order to consume that data, is you have to download that file, and put it into your storage, your application sandbox. And then

from there then you can view it in your service.

So that's a pretty big limitation. Now they are working on some APIs that will allow your application to be able to act as a third party handler for the user's data. So in other words, your application will be able to work in proxy for your user to request that data. Those APIs are not developed yet. And so for now, you have to copy the data into your application in order to use it in your application.

So the process of doing that data copy is as follows. You're going to use the REST service API, and you're going to create a bucket. A bucket is analogise to a folder that lives inside of Autodesk's Forge's file storage system. That file storage system is called OSS.

From there, you're going to upload the file. So if the files already in Forge, in say Autodesk 360, you need to download it. Then you need to re-upload it in the Forge using your consumer key and secret, and put it into your bucket. After it's in your bucket, you can request the derivative service, which is the translation service that lives in the cloud, to translate that file into a viewable format

We'll talk a little bit more about what the viewable format is. And then the next step after you get the file so that it can be consumed, you need to create an instance of the Forge Viewer. It's super easy to do. If you look here, all you have to do is add a style sheet reference. You have to add a reference to the viewer 3D.min.

And then you need to create a place in your XML where that will live, and you do that by defining a div, which is a section in an HTML document. Once you define that div, you go through and you initialize your viewer, and then you tell it to load the model that you're interested in. So it's actually pretty remarkably easy to consume, as far as the component APIs concerned. But these are the foundational steps to being able to use LMV for a custom application to view your information.

So it just shows you-- Autodeskdeveloper.net that's or developer.Autodesk.net, this is where you register your application. This is where any documentation has, and this is where there are quite a few examples that you can reference. So one of the things that's been really interesting to me in consulting is we have gone and we've talked with all sorts of customers. And as we talked to them about their business problems, we have found that there is a remarkable resonance across a variety of different customer bases on how the ability to combine data with visualization can provide significant answers to their business problems.

It's been amazing how well that's been received by everyone we've shown it to, from the CXO level down to the manager level. It's been amazingly well received. It seems like the timing for this technology is really ripe.

So the data that we're talking about viewing can come from a variety of sources. In the case of a BIM model, or perhaps a large assembly and [? inventor. ?] That data comes from the model itself.

From the manufacturing products, such as Inventor or Forge, you have part numbers, materials, mass properties. For plant data, for visualizing large plants, you have the asset tag numbers, you've got the pipe connection codes, you've got the line numbers. All that information exists within your model.

In a BIM model, then the idea is to have extra information that describes the functionality and behavior and systems in your building through the BIM data. And that's where a lot of people are coming and finding interest in this is because of the fact that they're already trying to add metadata to their model through BIM. And because the models there, they want to find ways to consume it.

All right. Now, as part of the translation process, when you take a file and use derivative services to convert it into a viewable format, the derivative services does a few different things. In fact, I skipped down and we'll get into that in a little greater detail. It creates-- it will create thumbnails of the different views in your model. And by different views, I mean you'll have 2D representations you have 3D representations.

It will create an SVG, which is your 3D model, in a streamable format file for your model. And for certain file types, you'll have different views in your model and it will create a separate SVG for each of those views. It will create an F2D which is a 2D viewable format, for each of your sheets within your model.

And then it will take all the properties in your model and it will put them into a SQLite database, and that database has an SDB extension. It then takes all of these files, these different formats, and it stores them inside the Forge OSS file service. And we call that the bubble. If you look at the LMV, the bubble is the collection of all these artifacts that are generated through the translation process. And when you want to view an asset in LMV, you need to ask the bubble to give you the information, so you know a specific asset and you tell it to load

that's specific F2D instance, or you tell it to load that specific SVG instance.

I want to show [? you ?] a little bit about LMV, and how it can be used with the property data to view. By the property data I mean, the data is extracted and put into the local database. And how that can be used within a simple application.

So this is a Revit model. And in the Rivet model, you can see all the different views we've created. In this case, we have all the sheets. So we can look at the different-- any sheets that's been found is--

AUDIENCE: We can't see.

ROBERT ANGUS: Oh, OK. Thank you. Where is it? Hold on. I'm going to change my presentation so we share the screen. So in this file, in our bubble, then we have all these different sheets and we have our 3-D model. One of the things that we can do then, is we can query the system based on any of that data.

So if I want to say something like, I want to see roof. It queries the SQL database, and finds all of the object IDs that have anything related to roof. And so there are some interesting things here. For example, you look at this and you'll say, well, how come these items are selected when I said roof? You can look at the properties in the system, and you see that it's because the top level of that is associated with a roof.

So all we're doing in this case, is we're using the API to ask it to search all of the properties to find anything that references roof. And then we're telling it to highlight those in the model. So if we go to [INAUDIBLE] Oops. Sorry. So if we go to another example, we can see how this can be used in a little more-- in a richer way.

And what I want to show here are a couple of things. First of all, I've learned two instances of the viewer. One instance that's this pointing to a 3D model and one that's pointing to a 2D model.

This is from a Revit file. So in a Revit file, all of the entities in the model are given, when they're translated, they're given an object ID. And whether that object is visible in 2D or the 3-D, they're all referencing the same ID.

So I can tell it to select an object by its ID. For example, I can tell it to select this wall, and it knows that that's the same object in 2-D as it is in 3D. So I can then take that one step further

and I can say well, let me extract all the properties in the database. And let me try to gain some-- try to establish some context around that.

So in this case, I've taken all of the different categories of items or this applications taken all the different categories that are in our application, and we can use that to filter and view the data. So we can say, well, show me the pipes, show me the fittings, show me the curtain walls.

This has been used by some of our customers to do things, such as estimating. They can view the different options during the proposal stage of a project and they can present these to the customers. I believe Jay Dunn presented this last year. How they use this to allow their client to make decisions by isolating the information that they're trying to get them to make decisions about.

The other thing that we can do is we can use it, the rendering capabilities Invent 360 Docs to highlight information and to make it accessible. So in this case, I'm looking for the attribute level, I'm aggregating those into groups, and then I'm applying different styles to each item, each level item.

Any questions about anything we've talked about so far?

AUDIENCE: So what's it look like when [INAUDIBLE] the web page side of HTML, the JavaScript and all that, is this HTML to capture all that data [INAUDIBLE].

ROBERT ANGUS: I'm sorry. What does the HTML-- so all of the data here is in-- for this application, all the data is local in the HTML page, because we're taking the information that was in the bubble, we're pulling it into the browser, and then we're querying it. So--

AUDIENCE: [INAUDIBLE]

ROBERT ANGUS: Yeah. So there's an established API for all these calls. For example, what we're doing here, there is an API called set color. And you pass in the entity, the ID, and the red, blue, green color definition, and pass, and the object ID, and we'll set the color.

So it's a very defined well bounded API. In the case of where we are hiding objects, there is an API where you pass a collection of objects and you say, isolate. And it will only show those objects, and it will hide all the others.

And you can do the same thing with selection. You can do the same thing for making things

visible.

AUDIENCE: And [INAUDIBLE]

ROBERT ANGUS: So this code here, I can give you the URL. You can access this code. It's on GitHub, and you can download it and look at it.

AUDIENCE: OK.

ROBERT ANGUS: Yes.

AUDIENCE: I'm just curious about the data, you were talking about it's all local. Does anything go out? Because we have security requirements that we really can't [INAUDIBLE].

ROBERT ANGUS: Yeah. So as part of the translation process that I talked about, when you're using the derivative services, you take the file, it is then copied to our server and translated. And then it lives on our server. Now, there are certain cases where they're allowing customers to translate it internally, so that rather than being transferred in our server, it can be translated within your firewall.

AUDIENCE: So there is potential for that in the future?

ROBERT ANGUS: Yes. And the other questions? OK. So this is all data local to the model. So I want to do a little bit of reset here and talk about what I think is significant and what we're able to do that we weren't able to do before.

The ability to show-- that's a big model and loaded in a few seconds. It's actually pretty amazing. If you would load that in Revit, you're probably talking about downloading an 80 megabyte file that's going to take a long time to load. And that's a small Revit file.

And the things that have enabled this to happen are pretty remarkable. A few years ago-- 10 years ago when I joined Autodesk, we would have had to have a workstation to do this. We would have never been able to do this on the web, because of the bandwidth requirements of that 80 megabytes.

In addition, one of the things that is really enabling about the Forge Viewer is you're no longer bound to a software license, as far as being able to publish this and consume it. This has an amazing effect with our clients. For example, we have a client who does clash detection on the buildings.

And it used to be-- the clash workflow is very cumbersome. You would pull in your Revit model, you'd have to translate it to Navisworks. You go to Navisworks and you run a clash. Well, the Navisworks is very noisy in its clash results.

You have to define what is a clash and what isn't. And you have to define the fact that 30 or even 1,000 different clashes might actually represent the same clash. So they would take their Navisworks model, they'd pull it into AutoCAD and create spheres around their clashes. And that allowed them to visualize the spheres.

And then they would have to have a meeting where everyone could come and look at their AutoCAD model. And have a triage session where they're managing their clashes, and they're deciding who to blame, in essence, and who's responsible for the fix, or which group of people are responsible for the fix.

And all this was bound to license software that involved multiple people doing translations, and other steps. And now they're changing their business by saying, you know what, we can run this in a background process. We can pull it in the LMV and now we can remotely pull our collaborators together.

We can view the clashes that's now integrated with a system that manages who's responsible for it. So they manage the resolution of it, all on the same system. They're able to view, interact, and show that data, make decisions, and then track it through the clash resolution process.

It's pretty amazing how this has enabled them to combine a whole bunch of separate workflows into a single unified experience for their users. And they attribute this to significantly increasing their competitiveness in the marketplace.

So the fact that you're able to use software that's not bound to the OS. It's not-- another thing about AutoCAD, or Navisworks, or Revit, they're not to be trifled with. You can't give a copy of Navisworks to HVAC contractor and expect him to just know how to use it, and view it, things in it.

On the other hand, everyone knows how to use a web browser. And you can create the interface that makes it very easy for them to interact with that data. In fact, they're the ones that I first they've actually demonstrated this in AU, but they're using VR to view those clashes. So they can create a collaborative session, where everyone's looking at the clash, but yet

while they bring you to the clash, you're free in VR to look all around and examine that clash.

So now you have 10 collaborators who are each able to independently assess the impact of that clash, and come to a decision on how to resolve it. Yes.

AUDIENCE: Is there [INAUDIBLE] cost to [INAUDIBLE]?

ROBERT ANGUS: There is no software licensing cost, there are services costs, so you pay for usage. But the Forge Viewer is not licensed. The translation, which is the CPU time in the storage for that file, is what is licensed, but it's relatively trivial. I mean, the talk is for a small model, it'll cost about $0.20. For a large very complex model around $1. So but once it's translated, then everyone has access to that without cost.

AUDIENCE: So one time?

ROBERT ANGUS: Yeah and you're just paying for the processing time. Yes.

AUDIENCE: The processing time [INAUDIBLE] consumes a certain database technically [INAUDIBLE] for me if I use this for an [INAUDIBLE] multiple users, or are we talking--

ROBERT ANGUS: Well, the translation only happens once. Once is translated and you have the SVG, and the SDB, and the F2D files, there's just in the cloud. You never have to translate again. You're just downloading them. So you're not paying to bring the bits across the wire, you're paying for the compute time to do the translation. Yes. In an update cycle, you would pay for each processing that you did?

ROBERT ANGUS: Yes.

AUDIENCE: And what I was wondering is, you talked about a push up [INAUDIBLE] get a key [INAUDIBLE] initially?

ROBERT ANGUS: So yes. So right now, say your using BIM 360 Docs, and you're using that for the process of managing the change on that document. You would go through a change management process and you eventually approve that change. Once that's approved, then you would release in essence, that documentation as part of that process, then you'd retranslate that file to your application.

Now, that's only if-- so the scenario that I was talking about, is if you have a third party

application, and that third party application is doing a whole bunch of work for a whole bunch of different users. If you want to have each user authenticate as the Autodesk user, then they can directly access the file that's in BIM 360 Docs.

So you're just building functionality around the Autodesk service, but you have to have a BIM 360 doc subscription, in that case, or an Autodesk A360 team subscription. So you're still paying for those services, but you can have them log in directly, rather than your service acting in their behalf. If you're trying to write your own internal application that's independent of BIM 360 Docs, then you have to do the move the data back and forth. I hope that's clear. If they are logged in as the user, then they can access any of the server the user is subscribe-- services that the user has subscribed to.

AUDIENCE: I was more just thing about the generic public HTML that needs to be updated. Once you generate your key, if you're updating the [? bubble. ?]

ROBERT ANGUS: Yes.

AUDIENCE: You wouldn't need to regenerate your key, it would--

ROBERT ANGUS: No. The key is a one time thing. So you get the key, you're registering the application which, as I mentioned before, that does is it authorizes your application to have storage on the OSS service.

So and typically, I think for most companies, if you want to have something like this, it's more it's facing external to the engineering organization and you're trying to add value somewhere else in the supply chain or in the downstream processes. And so it makes sense to have it more of a third party application that's acting on behalf of the user.

Now that being said, I anticipate that sometime in the near future, we'll have the API's that will allow your application to act for and behalf of the other user, so you could use their storage and access that for them. The first product that will probably support that will be BIM 360 Docs.

And they will provide the API's that will allow you to do things, such as, ask the system, give me the list of files this person has. Because you're not that person. It's a tricky thing because you're giving super powers to your application to access other people's data. And so that's why they haven't got it yet. They haven't decided the rules that have come into play to allow a third party application access to other people's data. So there will be some definite rules and

security measures put in place and it's going to take some time to work all those out.

AUDIENCE: [INAUDIBLE]

ROBERT ANGUS: No. The Forge Viewer is not the viewer used in A360 Glue, and it is the viewer used in A360. Now they're in the process-- my understanding is they're in the process of moving the Glue viewer to the Forge Viewer. It just takes time. So--

AUDIENCE: So did you just say that the Forge Viewer is used-- like if I go to A360 viewer right now, is it using the Forge Viewer.

ROBERT ANGUS: Yes. So again, the Forge Viewer [? is skinnable. ?] The UI, the little toolbar buttons, those are all customizable. And so every product then creates the look and feel that they want to have. And in addition, other products are then creating wrappers around this viewer.

The BIM 360 Docs, if you want to use that for example, or is going to create what they call matrix component, which takes all the services that their viewer has and makes them as this as a higher level component. Because they've added things to the viewer, such as, redlining capabilities offline capabilities and other things. So you can choose to use that component or you can choose to use the Forge component. The caveat there is that if you use the BIM 360 component, you have to have a BIM 360 subscription or a BIM 360 doc's subscription. Yes.

AUDIENCE: [INAUDIBLE]

ROBERT ANGUS: Yeah, well, we'll talk a little bit about that. Let's move on and we'll kind of get-- so we're talking about viewing data internal to the model here. The power comes when you start expanding that beyond the model. You have all this enterprise data from all these different systems.

So let's start talking about that. And that's where I think the power of having something like an LMV really comes into play, because you already have access to the model properties within Revit. It's nothing revolutionary there.

Where you start to see the value, is when you have a work [INAUDIBLE] say an inspection process, say an estimating process, and previously they weren't able to see what they're inspecting. They weren't able to assess. In a plant scenario, you may say inspect this pump, but you don't know, do you have direct access to that pump? Is it going to require scaffolding? Do I have to move other pieces of equipment to get to it? There was no way to triage and assess your work plan because you had no way of knowing it.

In a large refinery, there's thousands of valves, and no one knows them offhand. And that's actually one of the big expenses a lot of our oil and gas customers have. There is a pump station in Alaska. It takes a helicopter to get there. And they have to send someone up there to assess the situation so that they can plan how they're going to do the maintenance of it. That's an extremely expensive operation. If they could visualize that, that would have a huge benefit downstream and upstream for their processes.

OK. One thing I wanted to point is for those who have been with Autodesk for a while, you'll remember the churn of viewers. I mean, we have never had a good viewing story and this is the first time I think we've come up with something that is a great service. All right.

One of the things that we're seeing is people's expectations are different than they were before. Especially the millennials that are coming up. They're used to having data, , all the time accessible. They're used to having apps that perform specific functions. They're very niche oriented, rather than these monolithic applications. And so we're seeing more of a desire to have to find solutions to various workflows.

So again, I mentioned before about how combining visual graphics with data has absolutely resonated with people. Everyone who sees-- whenever we show this to someone, they start coming up with ideas on how they could use this in their own organization.

So to me, it's is a very compelling thing. There are certain things that have really enabled this that didn't exist in the past. One of them is the open standards that didn't exist before. The HTTP protocol, WebGL, WebVR. All these standards-- WebVR isn't an official standard yet, but it is coming along. It's in its final stages.

But these standards allow us now to program against a standard set of rules that can be then run against anywhere. And because of this, standards based things, data vendors and information providers have been forced to comply to those standards, and they've done things like provided REST API's to access their data. And so now you have a ubiquitous way of accessing their data, that you didn't have previously. Used to be you had to have an Oracle Driver or you have to have a SQL Server driver. Now they all have REST endpoints that allow you to access that data.

The thing that's even more amazing to me is the amazing amount of community and synergy that is in the web world around tool sets, and frameworks, and helping people come up with

solutions. And that's exactly what we're trying to do with our solutions. We're trying to create a standard set of APIs and tools that are accessible to anyone that you can use as components to build solutions to real problems.

Just again, the proliferation of data, I think is very significant. And there's two types of data that I think we're trying to solve. One is big data and that is we're collecting so much information. The internet of things is going to start bombarding us with sensor data. Positional data,s we're going to start being bombarded with where our assets are.

So we have massive amounts of data, and it becomes very hard to make sense out of it. And the ability to do things to visually represent that data has significant effects on what value there is to that data. Also, right now we have tremendous amount of siloing of data. I see this all the time. They might have-- internally, your company might have a custom database for managing inspection data. It might have an SAP as an ERP system. You might have a proactive maintenance system like Meridien or Maximo for your facilities management.

All this data is really separated from one another. And then you have your model data, your BIM data. And even though we have all that data, there needs to be a way of aggregating that data so that you can define the meaningful relationships that exist between them.

Again, this is more of what I just talked about. And I mentioned before about things like clash management or in manufacturing you have a design BOM versus a manufacturing BOM. In maintenance operations you generally have tons of word documents that are procedures that don't have any context to the actual equipment in the plant. So we have all this heterogeneous data and where people are seeing the need, at least from my perspective, is how do I started aggregating that and consuming it in a way that defines a solution to a specific set of problems.

And so as we talk about all these things, what's the business benefit? Is it going to save you time? Is it going to reduce rework? Is it going to facilitate decision making? Because-- I work with clients who, because of their perceived technological leadership, anticipate that as a result of AU, they will close several large contracts.

I have another client that I've worked with, their competitive advantage is the fact that they have better control over their safety procedures and how people are adhering to them. And because of that, they have a competitive advantage because it reduces legal [? waste, ?] it reduces schedule delay, and allows them to stay on task as far as the pricing goes. There's

little things that people use to increase their competitiveness in the marketplace. And many of these are geared around their use of technology to solve a specific business objective.

And so there's either people who are collectively working to solve a problem, or there's people who are saying, what's the cost of doing nothing? And you have to ask yourself what's the cost of doing nothing in today's world, where there's so much data, so much information, and some people are choosing to take advantage of that and some aren't. Is your response time going to be as fast as your competitors? Are you going to be in control of your processes? Are you going to be able to maintain the tight schedules that people are having?

So I wanted to show a little bit of how we can start accessing external data through some-- and I like to show these because these are all examples-- oops. Page up. Where am I. These are all examples you can download on the web and look at. So I'm going to go ahead and pull up a manufactured product very similar to what you had before, but now we're going externally to the external system and we're saying, OK. Show me the price data for these things.

I'm going to go to my SAP procurement system, and I want to be able to look at all of the rubber components and see the price break down on those. Or I want to be able to look at the aluminum components.

AUDIENCE: Did you say SAP?

ROBERT ANGUS: Yes.

AUDIENCE: [INAUDIBLE]

ROBERT ANGUS: Yes. So this is not SAP, but we do have-- and I'll talk more about that in just a minute-- we do have customers who are actually queering against SAP data and pulling it back into their processes for estimating. In fact, last year, Jay Dunn presented on the main stage their system, and that's what their system does. Is they go through and they look at historical data, and they say this is how much of this type of thing costs. And then when you click on that in the model, they say based on a historical data, this is what we estimate it's going to cost to do this system. And if you change it to something else, this is how much we think it's going to cost you. So it's all mining data from their ERP system.

So very simple, but we're now starting to get information from external sources. Now, you're talking about [? SAP ?] data, there are some problems with getting information from actual

service sources that I don't know if you're aware or not, but there are major roadblocks as if you're not aware of them. So I want to talk to those real briefly, then we'll look at some more examples of external data, and then we'll get to the VR stuff that a lot of you seem to be wanting to talk about.

One of the big caveats is the same-origin policy. And what that means is the biggest, the most prevalent security vulnerability is cross-site scripting. And what that basically means is anytime you have a HTML page, you can hit F12, and you can go and you can make changes to it. You can go in, and you can click on a button, and you can override that but it's handled. You inject your code into that web page. And what cross-site scripting is people do that automatically through vulnerabilities in your code. They go in, and they inject their code, and then they try to get your information, and send it to their server.

So in other words, this is how your application expects itself to work. You're going to have your application that's going to talk to your web services. And that's the only thing the browser allows you to do. It's called the same-origin policy, meaning if you're Autodesk.com, you can't make a call to it.

Well, the problem is now you want to call your SAP server, which has some other address and origin. And you want to call that your salesforce.com service and get some information on what's coming down the pipeline. And you have your own custom homegrown database for scheduling. All these are not the same origin and as a result, if you try to call those from a web browser, you're going to get an error saying that you can't, cross-site calls are not allowed.

So the idea is to prevent the evil cross-site scripter. They're injecting code but the browser is preventing them from calling their server. So again, here's the scenario. You have many, many different sources you want to access. There's three ways to solve this problem. One is to create your own web service that does the routing independent. The other is to use JSONP, which I don't want to talk about. And the other is CORS.

Wrap it up in a web server basically you create a server that makes all those calls and your browser just calls your web server that's in your domain. That's a very common way to do those things. So it's probably the most secure way to do it, but it requires the most amount of effort.

CORS, or Cross Origin Resource Sharing is really a protocol that allows a server and a browser to negotiate, if you will, the ability to share data from a different origin. It's a little less

secure but it's way nice. And so that's typically most of your data sources, especially most of your web based data sources that allow you to access that data across site.

Another caveat is when you're building an application, this is especially true with LMV, your browser is single threaded. Everything runs through one thread of execution in your processor. And browsers also happen to make calls to resources that take a long time to get there. You're going to download that SVG file. That SVG file, that model definition streamable model file could easily be 80 megs.

That takes a long time. So what we do with a browser is we make everything nonblocking. So you call-- you make a call and you say, when you're done, I'll handle you. So you do all these non-blocking calls, but that limit your ability to handle performance.

And in a VR situation, your VR platform wants to run at at least 30 frames per second and you have one thread to do all that in. In addition to all your business logic. So we'll talk about that in a little bit.

AUDIENCE: Is the presentation going to be available later?

ROBERT ANGUS: Yeah, it will be up on the site. The other thing that's a problem when you're dealing with data from multiple sources is universal identity. Every system has its own way of identifying its entities. And how do you establish commonality? How do you--

Ron here, who works with me, he [? made a ?] terminology, he called the friendly object ID that I like and I copied. And what it's really saying is some way you've got to be able to define an ID that spans all of your systems. So that when you click on a door you know how to identify the SAP data. You know how to identify the material data sheet. You know how to identify the high definition rendering in 3D Studio Max. So you have to have a commonality of ID.

There's three different ways to do that. That I kind of use, there might be a whole lot more. I don't know. The first one is to create the friendly ID, and then the owner becomes one of your systems. Maybe SAP becomes the owner of that ID. All the maintenance to that ID happens in SAP. And every other system then has as a property, that same ID.

So in Revit then you would create a property and you would call it, if its a tag ID, say this is my tag ID. And if you change the ID in SAP, then you've got to propagate that change to all your

other systems.

The other way is to create an independent hierarchy that's in essence its own database and it establishes the mapping for you. So that if I want to change it, I change it in this independent database and I can make all the adaptations in the independent table that contains all those mappings.

And the third way is very similar is you have a separate hierarchy, but you still maintain the property and each of the individual data sources. This might be a little techy, but it's a real problem when you start to deal with data from different systems, and visualizing that data from different systems.

The other thing you have to consider is how much do you really want to maintain in your Revit model or in whatever you're trying to visualize? By very nature, model files aren't great databases. Databases tend to be transactional, accessible, lightweight access to data. And it's just the opposite with a two gigabyte Revit file. In order to read the data, you've got to load two gigabytes of data, and then you've got to access it.

And that data is really not-- that file is really not designed to handle transactional things. And do you want to go through the overhead, you mentioned before, if I want to change a property of some data, do I want to have to retranslate and go through that whole publishing pipeline of my Revit file just to change my manufacture?

So in general, I think in a large measure, especially depending on the scope of what you're trying to do with your data, it makes sense to maintain data outside of the Revit model or the Inventor model as much as much as possible. If you're only concerned with the design, then maybe that makes sense. But if the lifecycle of the object is much greater than the design process, it makes sense to maintain it externally.

OK. And any questions to this point? All right. I've seemed to have numbed you all as your waiting for virtual reality. So here's a prototype we're dealing with that has been very well-received. And it's a little different than what I showed you before.

This is a plant and in this plant there is a bunch of data associated with each of the objects in the plant. But in the process in industry, the regulated document isn't a model. The regulated document that everybody has to maintain is a PNID, which is a schematic. It's a process schematic.

And these documents really aren't related as far as how they're made. One is made in AutoCAD, the other ones made in Plant 3D. But yet they represent the same data. So in other words, if I click on this, it represents that tank in the schematic. And so now, we're starting to do things that really the people really struggle with is how do I link process's to models containing from different sources.

So what we've also done then is we said, well, there's a bunch of additional data associated with this. Let me-- Oops. Yes.

AUDIENCE: So there's a model which has [INAUDIBLE] Then you have 2D [INAUDIBLE] which isn't driven by models [INAUDIBLE] you've added some type of ID [INAUDIBLE].

ROBERT ANGUS: Yeah. So I can demonstrate this clearly in the second demo. So let me do that and we'll come back to this. This is very similar. Kind of the gen2 of what we were doing before.

So I'm going to open a PNID document. I'm going to open a model. Little simpler model. The other model, actually, that I pulled up at that plant, that's a model with about 50,000 entities. That's a huge model, and it loaded pretty quick. Pretty amazing actually.

This model has maybe eight to ten thousand entities. Much smaller. So if I look at this pump here, and I go to my properties. We can see that I have it-- a tag. Sorry, every time I highlight it, you probably can't see it very well.

P103, I have a bunch of other properties and when I click on one to the other, they highlight back and forth. But when I click on this, I come over here and I click, nothing happens.

That's because this is a piece of piping. It doesn't have a tag. It's a line number. And so there's no direct relationship. So to handle that, we say well, what do you want to search on? And we said well, let's search on line number. Whoops. We come over here and do that.

I got to search on line number so now that when I clicked the line, it will select the line for me.

So we've extracted the data, indexed it, and now we can choose how we want to process it. So we can say hey.

The other thing is that I have equipment from more of an ERP-type scenario. It lots of times doesn't know the line or the tag but it has a universal ID. So I can come here and I can say well, I want you to select using the UID so that when I come here it knows how to select it

based on the universal ID-- the universal identifier.

So that's what I'm talking about creating a mapping that maps one system to another and it's extremely powerful. But it takes a bit of work. But these are the problems that people couldn't solve before because there was no way of pulling all this data together into a single source that then can be indexed and referenced in a single application. But the open standards and the tool sets now make that very accessible.

Now when we talk about VR-- this is where you can see VR becoming a significant thing. You're responsible for maintaining that pump. You're going to want to be able to go and click on that pump and look around it, assess the clearances.

And as we started going into AR, or augmented reality, it's going to change even further, because then you'll be able to look at the pump, hold your tablet up to it, and it will identify it and then you can click the operation and maintenance man or you can click the maintenance records or whatever else you want. That's the power that's really going to be coming in the future.

AUDIENCE: Just to be clear, you're basing your [? short delay ?] CSS for screen size, and different things all in one? So you're showing your own pop-ups are going to function identically on all the machines that supported all of the browsers that you mentioned.

ROBERT ANGUS: Yes. Yeah. Well-- and in reality, right, this is prototype code. These buttons don't really apply to the specific workflow, so if I were to really design this, I would only put the buttons that I wanted my user to see.

AUDIENCE: [INAUDIBLE]

ROBERT ANGUS: Absolutely. I don't know if you've done JavaScript programming in the past, but like 6 years ago or maybe 10 years ago, it was a nightmare. I mean, you had to test on everything. I would go into cold sweats thinking about JavaScript programming and so I didn't even want-- when I got asked to do this stuff, I was dreading it and man, the world has changed. JavaScript programming, as long as you're not dealing with Internet Explorer, it's great.

The problem is every company has Internet Explorer, right? And half the people have Internet Explorer 8 still, which is a security nightmare, but it's getting better. Internet Explorer 10 is not bad and what the community has done is they've developed what they call shims.

And shim says I know Internet Explorer doesn't support all of the newer JavaScript standards, so I'll create a shim that will make it pretend like it does. In fact, in virtual reality, that's what we do.

Safari does not support Web-VR. Everybody has an iPhone and it doesn't support it. So there's a Web-VR shim that we put in that makes it think that it's a VR-enabled phone. And in actuality right now, LMV is broken-- won't show you virtual reality on an Android device, which supports Web-VR but it will on the one that doesn't through the shim, so--

Anyway, I probably overanswered the problem but you should be able to expect the same behavior on any of the supported platforms.

Let me go back real quick to the previous demo and just point out a couple of things. One of the things we try to do is try to use this to deal with procedural data. There's lots of procedures you have to go through. And in this case, we have an energy isolation on a plant, which is really defining the steps that someone has to do in order to make sure that they have a safe working environment.

And what this allows you to do is click on the different steps, and the [? PNID ?] highlights. It takes you to the place in the model so you can see what you have to do.

Currently, I mentioned this before, but this is a Word document and all it does is it has a tag number that means nothing to the person who's got to go do this job. Now, they can click on that step and it takes them right to where they need to go. They can look, do I need a ladder, do I need whatever.

And the next step would then be to integrate with something like our field application. And when they do it, they can digitally verify that it's done so then it triggers the next workflow and that would be the inspector coming and verifying it so then you're free to work on it.

So the idea is, again, you create something that solves a very specific problem but they haven't ever been able to do that before because it takes data from so many different sources and there was no way to visualize it.

AUDIENCE: [INAUDIBLE]

ROBERT ANGUS: Yeah.

AUDIENCE: Do you relate it to what the application is? [INAUDIBLE]

ROBERT ANGUS: Thank you. Yeah. Like I said, we showed this [? plant ?] to Dallas Fort Worth-- this [? plant software-- ?] which has nothing to do with an airport-- to Dallas Fort Worth International Airport. They went crazy over it, being able to use visual data to help them solve their operational problems.

OK. I need to see what time we have here.

Let's move on. I think, like I said before, I'm kind of beating you senseless with the logistics of LMV. Let's dive in real quick.

I wanted to keep revisiting why this is different than before and there's a different paradigm that existed before. And you weren't able to do these types of things before. The enabling technology is new and that's why Autodesk is jumping on it so much is because we envision a class of application and the class of accessibility of data that never existed before.

OK. As you can tell I'm not a PowerPoint jockey. You'll have to forgive me. I'm a code pig.

Another thing that I want to show you that I think is pretty cool is the internet of things. What it's really dealing is dealing with data that's coming in in massive amounts and sensor data, positional data. I have a simple demo here. Basically, what I have is I have a source app that broadcasts thousands of messages a second to a pubnub service that's probably sent somewhere in California. I don't know where the server is but it's a service that's solely independent of AutoCAD. And then it publishes that back to my viewing application. And what I wanted to show you is how performant LMV is in handling these thousands of data requests.

I'm going to start the service. Come here. Tell it to start listening on-- well, let me close these. OK, I'll do it. I'm going to tell it to start listening to the publishing of it and I'm going to go ahead and start broadcasting the data.

And what you're going to see here is now-- I'm receiving this and it's supposed to represent temperature data, the hotter it gets, the more red it gets, whatever. But I'm just trying to visualize that this is going to thousands of components and I'm sending updates three a second. And the whole time it's doing this-- it's a teeny bit jerky but I can still rotate the model. I can still interact with it. It's not slowing me down at all and I'm receiving all this data. I'm making sure I don't block as I process the data and it's just updating the model.

To me, it's pretty amazing on speaking to the scalability and performance of LMV. This is not a small model. It's not huge model, either, but it doesn't even skip a beat. The total amount of data I send is probably on the order of 300,000 or 400,000 data point updates.

One way I handle it as I collect them into batches-- and I updated about every half second so I receive the data-- but I don't update the visualization as it comes in. I group it and then dispatch it out so that it has a little more time to render. You

I should be stopping here soon. I have it on a timer here to tell me. Yeah, it's done.

So that's the final view. Pretty amazing. Again, almost zero coding. I'm able to graphically take internet of thing-type sensor data and depict it to the user.

AUDIENCE: [INAUDIBLE]

ROBERT ANGUS: What's that?

AUDIENCE: [INAUDIBLE]

ROBERT ANGUS: Yeah. OK. So let's talk a little bit about virtual reality. I wanted to distinguish the difference between virtual reality and augmented reality.

Virtual reality is putting you in a simulated environment, in essence, where you can interact with that environment and you receive sensory data about it.

An augmented reality is presenting a virtual environment into your physical environment. So Microsoft HoloLens is a good example of an augmented reality. Well, it's really about the only good example right now of an augmented reality product and it's pretty darned compelling.

Who here has used HoloLens? It's pretty amazing, isn't it? Yeah.

AUDIENCE: What's different with mixed reality-- [INAUDIBLE]

ROBERT ANGUS: I have no idea what he was showing, so--

AUDIENCE: He called it like MR., mixed reality.

ROBERT ANGUS: OK. That's what HoloLens is. It's augmented. They internally coined it as mixed reality so when you look at their brochures, they'll call it mixed reality.

AUDIENCE: But it's actually AR.

ROBERT ANGUS: Yeah, it's putting a virtual environment inside of your physical world. So they're just trying to differentiate themselves and, quite frankly, their technology is amazing.

Unfortunately, LMV right now doesn't support HoloLens, so if you think of it, it makes kind of sense because what we're doing with web VR is we're projecting everything onto a web page.

And what HoloLens is trying to do is go into the middle of your room. And so it's like we would be throwing a web page on your wall and then you'd be rotating the model. So it's going to take some significant upgrades to make it so we deal with LMV in a mixed-mode HoloLens environment.

But there are some possibilities. You could do it. It just doesn't make a lot of sense right now. So a better pathway, if you want to go more of the HoloLens route, will be rather than using LMV, you'd use-- why am I drawing a blank-- Stingray. And Stingray will have its own--

AUDIENCE: [INAUDIBLE]

ROBERT ANGUS: Yeah. Live design would take your Revit model and do a Stingray format and then Stingray will support the HoloLens platform. Which really, Stingray and HoloLens are very close, right? HoloLens right now, you do your development in Unity, which is a gaming engine and Stingray's the same basic thing. It's a gaming engine. We're taking engineering assets, pushing them into a gaming environment.

I want to show you how to do VR. VR inside of LMV is brand new and right now, it's kind of broken. So I spent the last little bit trying to get a demo that I could show you. And I have an Android phone and it just crashes. I get open GL or web GL errors. [? Ram ?] here has an iPad and a phone that we can bring it up and show it to you, if you want to come after.

This is, in essence, what it does. It just creates-- it loads the model and then it creates two rendering scenes of that model. And then to make it better, they do what they call a lens distortion. And then when you put on the goggle-- and again in my rush to get over here, I left my goggle in my room-- it then corrects that lens distortion and just makes it seem like a more real scenario.

But in a model like this, LMV is fantastic, fantastic. When you start to do something like a big building, you run into problems. And the reason why is it's what they call a progressive render.

And what a progressive rendering does is it says I'm going to sacrifice fidelity for performance. And so when you render something progressively, as you rotate that object, it's going to slice it up into like 20 or 30 frames per second and in that frame, it's going to start with the biggest faces that are most visible and it's going to render them. And then as it has more time, it will start filling in the smaller pieces. Well, as soon as it gets to the timeline to start a new frame, then it starts rerendering.

And so what you have is on a big model, as you rotate, the little pieces never get rendered, but you don't notice. I'll show you that here. So if you look here, so I rotate that-- come on-- as I rotate this around, you'll see the little green monitor come up. That's basically saying, though, that I'm completing my rendering cycle and when that's gone, the rendering cycle has been done.

So as I'm moving this around, it's not finishing this rendering cycle. When I'm done, it finally finishes. It doesn't really bother you here but when you're in a VR scenario, you have the headset on, and it's always moving the camera a teeny bit. It never stops.

And so in a big model, you'll be looking at a room and the desk will be flashing on and off and that's because it's rendering. But it's never getting to that level of detail before the next scene. And it's compounded because now I have to render two scenes, one for each eye. And so the performance really suffers.

Now the LMV team has some strategies that they want to do to help rectify that but they're not implemented yet. I believe the current plan is to do what they call a box render.

What a box render will do is you'll go into a scene and when you're there, it will paint the scene picture and buffer it. And so as long as you're looking around, it'll just show you the buffered view. It doesn't have to rerender.

When you take a step forward and translate, it's going to have to rerender the scene-- sort of rerender the box. And once you stop, then you'll be able to look around and see everything fine. So it is an optimization that will work great in a non-high-motion environment.

If there is a lot of motion, then that optimization won't help at all.

Maybe they'll create a toggle that will allow you to have-- that will preference rendering quality above performance. But this is where the single-threaded nature of a browser hurts you because you can't solve the problem by slicing the task into a whole bunch of threads and

conquering it in mass. You have to do it on a single thread.

Yes?

AUDIENCE: Is the VR service running off the same level as the--

ROBERT ANGUS: So it's not a service. So let me show you how you turn on VR and if you have an iPhone, you can go do this-- there's a tool called-- tool. There's a website called Lmv.Ninja and Lmv.Ninja we can turn on the-- we'll step through it right now. And you go do this as soon as we're done. I don't have an iPhone and it breaks right now on my Android device so I can't do it.

OK. So you come to Lmv Ninja. Right now, the VR extension-- and it's an extension-- it was put into beta in July. It's still in beta. It was put in beta for a project that I was working on. So they hurried and rushed it through for this project but VR, it's massive and we'll talk a little bit more about it in just a minute.

So in order to view that, in Lmv Ninja, you got to go to staging which, in essence, is beta. So go to staging and then choose a model. Then say launch the viewer. OK. And when the viewer comes up, you need to go to the extensions. I need to say OK, enable Web-VR When you enable Web-VR you'll see this little dude that looks like the Google cardboard glasses. Go ahead and click that.

And it doesn't work on the desktop. It only works on your phone. It kind of messes up on the desktop but it will create that scene where you have the two screens.

Are we giving away any of the Google glasses at any of the booths out here?

AUDIENCE: Probably.

ROBERT ANGUS: I bet you can go-- there's a good chance because we did it at DevCon. We gave away a whole bunch of VR glasses.

AUDIENCE: Yeah.

ROBERT ANGUS: And you can get your iPhone right now-- you can do this.

AUDIENCE: [INAUDIBLE]

ROBERT ANGUS: This is pretty cool. It's pretty cool. The experience is great as long as your phone can handle

the frame rate.

OK. And that's why I said demo kind of-- I'm sorry I don't have a better story to show you. It works on the phone, Android. It works on iPhone but the important thing to know is VR is coming big time.

As of right now in January, Google has committed that Chrome will by default enable the Web-VR extensions. So that means every Android device that's reasonably up to date will support Web-VR.

In addition, Google is now selling their Daydream View goggle. Now what's different? Samsung has their Gear VR, but the difference here is it has the controller and that's always frustrating with an LMV-type environment. Whereas You just want to be able to take your LMV model and allow users to interact with it. And there was no great way to program motion into LMV.

The LMV team will support the motion controller and people will be able to navigate within the model using motion controller.

Yes?

AUDIENCE: You said something-- VR. [INAUDIBLE] One of my clients actually just ordered one. You can pair a Bluetooth to a [INAUDIBLE]

ROBERT ANGUS: Oh OK great. Yeah.

AUDIENCE: [INAUDIBLE]

ROBERT ANGUS: The Google Cardboard, which you can get for $10, is OK but the Gear VR experience is pretty cool. They're pretty compelling. They're like $90 nowadays. Fifty? Yeah and Daydreamer's going to sell for 79. It's finally going to make the technology accessible. You used to have to it talk about the Oculus glasses in there. Who knew how much they were going to cost?

And right now, a HoloLens runs you about $2,900 and HoloLens has-- the problem with buildings and HoloLens is after about 20,000 entities, HoloLens can't keep up and it starts to get skippy if you lose frames.

But on the other hand, the experience in HoloLens is pretty amazing. I mean, you literally want to duck around a-- you pop a building down, and you're dodging the beams that are virtual.

They're so real to you. I mean, it's a pretty cool experience.

So also Google is releasing the Daydream API, which creates a standard interface for creating applications. Microsoft is committed to supporting Web-VR, which again, that will give another platform. Apple hasn't made any noise on whether they'll support it or not but Apple processors by nature are very fast and they don't have a lot of cores, so they work really well for web.

And we get better performance using a shim-- which shims slow things down on an iPhone-- than we do on a multi-core Android device.

Where we're seeing a lot of interest is in design reviews, class management, those types of things. The experience to be able to sit around and look at a problem is amazing. I'm sorry I don't have more data, more visuals to show you. The one that I've been working on is proprietary and the customer won't let me show it to you. And then, like I said, it's broken and I can't film it. But you can access it on LmvNinja. It might work on your Android phone. Do it on Chrome. By default, the VR extensions are now enabled, at least on the Chrome development devices or Chrome development releases.

Time is running up. They know they have the optimization. There's more coming. They have some big plans for VR. Next year, the outlook on VR and LMV will be totally different than it is this year. There will be tons of success stories and tons of people will be doing it. It's just really immature right now.

So talk to a development team. Robert.Angus@autodesk.com. I'm happy to keep you in the loop with the latest developments on VR. It's looking like it's going to be a really big pervasive.

Any other questions?

AUDIENCE: [INAUDIBLE] So we have the whole coding, we have all of that in Fusion and [INAUDIBLE] in API and that [INAUDIBLE] make it to what we want it so we can get [INAUDIBLE] across the board. [INAUDIBLE] Why wouldn't you just technically make it plugin for now [INAUDIBLE] contains all your informational data to actually export and give that [INAUDIBLE] to a web page linked to your Autodesk account so you can manage it?

ROBERT ANGUS: Well, I don't see why you couldn't automate that process to do that, right? But you still have to go to an LMV format. You have to go through [? SBG. ?]

AUDIENCE: [INAUDIBLE] upload everything separate. It's one thing that just in the university and seeing all the different software. The problem that you guys are having to [INAUDIBLE] through permeability of all of the softwares talking to each other that it seems Autodesk has developed so many of that [INAUDIBLE] so much-- so I'm curious why Fusion wasn't just integrated into data sources and software we already [INAUDIBLE].

ROBERT ANGUS: Well, I think that the approach that we're taking is that the metadata you have to stay-- thanks everyone for coming. I really appreciate it. Feel free to drop off. For those who've endured it, thank you.

The model we're taking is the metadata you have to store in a fusion file the B-Reps and the level of detail is totally different than what's needed in a Revit file, which is totally different than the types of data-- you're storing features and [? depology ?] in Fusion. You're not storing any of that in [? Max ?] and so there's no common data format that we can evolve to.

So the strategy really is let's make it so the user doesn't have to care. They put it in the Cloud. They can get out whatever they need it in. So that's kind of the strategy, if that makes sense.

AUDIENCE: No, that's perfect.

ROBERT ANGUS: Thank you.