Module 61
A325: Determining Known Risks with Standards in Your Deployment

Ken Leonard: ITS standards can make your life easier. Your procurements will go more smoothly and you'll encourage competition, but only if you know how to write them into your specifications and test them. This module is one in a series that covers practical applications for acquiring and testing standards based ITS systems.

I'm Ken Leonard, the Director of the U.S. Department of Transportation's Intelligent Transportation Systems Joint Program Office. Welcome to our ITS standards training program. We're pleased to be working with our partner, the Institute of Transportation Engineers, to deliver this approach to training that combines web-based modules with instructor interaction to bring the latest in ITS learning to busy professionals like yourself.

This combined approach allows interested professionals to schedule training at your convenience without the need to travel. After you complete this training, we hope that you'll tell your colleagues and customers about the latest ITS standards and encourage them to take advantage of these training modules as well as archived webinars.

ITS standards training is one of the first offerings of our updated professional capacity training program. Through the PCB program, we prepare professionals to adopt proven and emerging ITS technologies that will make surface transportation safer, smarter, and greener. You can find information on additional modules and training programs on our website at www.pcb.its.dot.gov.
Please help us make even more improvements to our training modules through the evaluation process. We look forward to hearing your comments and thank you again for participating and we hope you find this module helpful.

Ken Vaughn: Hi, this is Module A325, Determining Known Risks with Standards in your Deployment. I am Ken Vaughn, the President of Trevilon LLC, I have been working for several years now with the reference architectures and in particular, with the international community in developing solutions for the different standards and interfaces in the architecture and documenting all of the issues with those standards. And we're going to explain how you can go online, find information about those issues with standards so that you can adjust those within your deployment.

Our learning objectives today include explaining a little bit about system architectures then comparing the different ITS reference architectures that are out there today and then we're going to link into those reference architecture content to the standards and understanding how all of that information is presented. Finally, we'll go into identifying known risks about your standards and providing recommended resources to learn more about the different architecture efforts.

So with that, we'll start with our first learning objective: Explaining the System Architectures. This has two major components to it; the first is kind of understanding the different levels of abstraction within ITS architectures. That includes reference architectures, regional architectures, and deployment architectures. The second aspect is understanding the purpose of those architectures and that's largely document system design defining key interfaces for integration and promoting a common marketplace.

So with that, the three levels of abstraction as I mentioned, reference architecture, the reference architecture provides overall template solutions, those template solutions can then be customized to your particular project area whether it's a regional architecture or a project or deployment architecture.

The regional architecture is kind of long-term vision of what you expect in your entire regional area and that region may vary a little bit depending on what region you live in. It may be a state-level architecture, it may be a metropolitan area architecture but it's some large area that typically involves multiple different jurisdictions and multiple agencies that all will be building equipment that need to fit together to be a holistic ITS infrastructure.

The project architecture is more detailed and it gets down to specific details of your deployment, so it's also called a deployment architecture and it provides those technical details, what standards do you want to use for a particular project or deployment. So what we're focused on here is not a long-term 20-year vision but one specific project. Both of those, the regional architecture and the project architecture, are derived from a reference architecture that provides that template solution.

So the goal of this presentation isn't so much to explain those architectures in detail but rather so that you can understand how you determine risks on your particular project and when you want to know about risks. Well the real answer is, "As soon as possible so I can start mitigating them and managing them." Well, the regional architecture, that's that long-term vision of what you're planning on doing, that's one of the very first things you should have, you should have that in your region kind of running at all times. The second aspect is the deployment architecture, which is also early in the project of any one project. Both of those, as I mentioned, are derived from that reference architecture. So that reference architecture is a great place to start documenting all of these issues so that in your deployment architecture and in your regional architecture, you can identify those issues early on and then begin to understand how we're going to handle those particular issues in our deployments.

As I mentioned, every deployment should be based on an architecture and the reference architecture just simplifies the process to provide that template solution that you can use for your project. The purpose to architecture is very similar to that for a building so if you're creating a building, you have a number of different plans and those plans, some of them might be structural—that's what people typically think of—but if you're an electrical engineer, you're interested in the electrical plans for that same exact building. Or maybe you're in heating and ventilation, then you're interested in those plans.

So all of those different plans describe the same architecture of the building but they're designed to address. The concerns of different stakeholders and the result of those different plan sheets is those different plan sheets are what we would call the view. So those views—those different views address the different stakeholder concerns of the different stakeholders. And the idea here is in software just like in buildings, it is much better to document what you're going to do on paper before you actually build it. Once you actually start the construction and building these things and deploying your equipment in the field, it gets very expensive to change so what we want to do is create an architecture that's going to describe how everything's going to work and then you move to deployment and that's the efficient way to do things. The one major concern—there are many concerns in systems just like in buildings—but one major concern almost in all projects is, "What are the known issues that I need to be managing with my project?" and that's really the focus of this presentation.

Now one of the purposes—one of the key concerns of systems architectures are the concerns of how do I integrate equipment from different vendors? I have all of this equipment in the field and they need to interoperate together in order for my system to work. Any system deployment typically involves multiple interfaces with equipment from multiple parties, so this diagram here talks about transit signal priority. So you see we have a traffic management center, a transit management center, equipment in the field, equipment on the transit vehicles and they're all exchanging information and more than likely, I'm not going to get all of this equipment from a single vendor which means now I need to integrate all of these different components.

That reference architecture starts identifying what those interfaces are so that we can produce standards so that those standards can define how these different products from different companies will interface together and build an interoperable system that doesn't favor any one party or another. So this diagram, by the way, is called the service package diagram. We'll refer to this a few times in the presentation.

Each one of the colored boxes represents physical objects, things that might actually exist in the field as a separate distinct box. I say “might” because some of the boxes could be joined together, some of the functionality may be broken up separate boxes, but this is kind of a good reference-level architecture.

When you build your project-level architecture, you'll customize it to exactly what you'll see on your project but this is the reference architecture that we've broken everything out in this format. And then the lines between these boxes indicates the information flows that get transferred from a source to a destination. So that's called an information transfer when it's transmitted from a source to a destination, that's also called an information flow triple, which—because it's the information flow, the data content, the source, and the destination. So these three items together are sometimes called a triple, they're also called an information transfer.

One of the other key aspects in architecture is the reference architecture promotes this common marketplace. It does this by standardizing those interfaces—or actually I should say identifying those interfaces—so that later efforts can standardize the interfaces and then that promotes greater interchangeability of components when all of the vendors conform to that one standard.

It also means that there's cost sharing and debugging components. Once we have an actual standardized interface, no longer do we need custom test solutions, we can build a single test solution that makes sure all the delivered components regardless of manufacturer meets that same testing interface. It also creates a large pool of experts with a common skillset.

If we start all deploying our systems in a very similar fashion then our experts can move from one city to another and still have a very good understanding of how everything works. And if we can do this expanding, using the same technologies as the broader Internet of Things community and Smart Cities communities are doing, then we can also leverage their expertise into our industry. And then finally, with all these standardized interfaces, we'll end up with much more competitive pricing and, as you see here, a very active marketplace where everyone can exchange on equal grounds.

Well that completes our learning objective one. We'll come to our first poll question. Which type of architecture provides a solution template that can be customized for each region or project? And your answer choices are listed there: A) is deployment architecture; B) is a planning architecture; C) is a reference architecture; or D) is a regional architecture. Go ahead and make your selection and then we'll move on.
So the correct answer is: C) reference architecture. The reference architecture is a template solution that can be customized for each site, such as the Architecture Reference for Cooperative Intelligent Transportation is one example of a reference architecture, that's also known as ARC-IT, which was developed—we'll talk about later—with the U.S. Government as well as some other governments in the world.

Option A, deployment architecture, was incorrect. A deployment architecture defines the before and after details for a specific deployment project, one project you're really focused on there. B, planning architecture was also incorrect. A planning architecture defines a long-term plan for the architecture with any specific region. And then finally D, regional architecture. Really, the term regional architecture is a synonym for the planning architecture, both of which provide that long-term plan for a particular region. The regional architecture is mainly a U.S. term. Once you go international, the term region starts becoming somewhat ambiguous so the international community has adopted the planning architecture term but regional architecture tends to be used in the U.S.

That brings us to learning objective number two: Comparing ITS Reference Architectures, and then we'll talk about a few things here. One is the template solution for ITS deployments and then we'll talk about the major ITS reference architectures across the world. We'll talk about typical ITS reference architecture viewpoints and then talk about some of the support tools that help you use reference architectures.

So we've mentioned the reference architecture is a template solution. Within ITS, it's designed to allow you to customize it to meet your local needs, that the way you customize it is by identifying specific instances of each component. So if you think back to the diagram we saw of the different physical objects, those different boxes—you might combine some of those boxes together, you might separate

some of them out, you might combine some of the flows, separate them out as needed and then select the specific services to be included as well.

So that one diagram that we saw—there's actually almost 150 of those diagrams within the reference architecture, some of them you'll want to use on your project. Some of them you probably won't because that's a very long list, your project will probably be focused on a smaller subset than that. So you select which services you want on your project and then you start identifying specific instances of each component.

You can also extend the services. We have a list of 150 service packages but that doesn't limit you to that list. You can create your own ITS service packages particularly for emerging technologies or something that's very unique to your environment. You also need to refine security and other details as needed to make sure it's customized for your particular environment for your particular details. And then once you go through all of that, then you identify the interfaces and interface standards that you'll use on your project and there's toolsets to help you go through this entire process.

Now, before we start talking about any one reference architecture, this slide helps you try to understand kind of the history of architecture efforts across the world. Early on, back in the 1990s, U.S. National ITS Architecture effort began and that you see there on the lowest branch there across the screen and at the same time, roughly, the European Union also started their effort to develop an ITS reference architecture. Their reference architecture was called FRAME and you see that going that across the top segment there.

Well shortly after the EU started theirs, Australia said, "Well we like what Europeans did, we're going to customize it a little bit because we have a very different environment here in Australia than they did in Europe," so they added some additional functionality and things to that architecture and they were like an offshoot of the FRAME architecture.

And then in the 2000s, the U.S. Government decided, "Well we also want to start developing architecture for connected vehicles and eventually this will be rolled back into the main architecture but it's different enough that we're going to develop a separate architectural branch for connected vehicles." So they developed those Connected Vehicle Reference Implementation Architectural branch called CVRIA.

They're in kind of the middle and that was for only the connected vehicle aspects of the architecture. Well, around 2015 or so, there was an interest in seeing how we combine all of these different reference architectures and Japan had developed some stuff as well and we decided to develop this thing called HARTS, Harmonized Architecture Reference for Technical Standards. And what this was designed to do was strictly related to connected vehicles. We decided to look at how we take those services defined as CVRIA, combine them with what the Europeans had done in FRAME, work with the Australians to add in their additions as well, and with Japan to add their content as well, and that became HARTS.

And that was an initial effort not only to harmonize the architectures across the world, but to also begin to identify the issues and gaps of standards that need to be addressed by the international community if we're going to actually realize this technology, so that was HARTS.

About the same time, that same CVRIA project was being rolled into the next version of National ITS Reference Architecture here in the U.S. which was the first time it was called ARC-IT. That became ARC-IT 8.0 and that started including that connected vehicle content. The next step, which will be happening in the summer of 2020, is HARTS will be rolled into ARC-IT 9.0 and all of the issues and gaps and everything that we originally documented within HARTS will now be shown in ARC-IT 9.0, except now we're going to be extending this to cover all of the U.S. National ITS Architecture content.

At the same time, we're also incorporating content from Canada, Australia, and Europe into that ARC-IT 9.0 content. Now the frame and Australian architectures will continue somewhat on their own path but

they're probably going to start linking into our efforts as well. So the goal is that ARC-IT 9.0 will become now, hopefully, an internationally recognized reference architecture for all of ITS showing solutions across the world. So if we don't have a solution, a set of standards to deploy something here in the U.S., we can at least see what they're doing in Europe all on this one internet site.

So with that, what exactly is a reference architecture? Well, all of the architecture content developed by the U.S. DOT has been based on the ISO 42010 standard and this standard is the international standard that defines how you define an architecture for systems. And what you see here in the middle is the architecture description. And don't get too concerned about the diagram—this is really just kind of introducing terminology to you so you understand how everything relates. That architecture description is the document you read on the website, basically. It is what is describing the architecture so it expresses documents, if you will, the architecture and then the architecture is what is exhibited by the system of interest, i.e. how it's all put together.

So it doesn't really matter if you document it or not, every system has exhibits in architecture and then the architecture description is merely an expression of what actually exists. So that description begins by identifying what is the system of interest, in our case it's ITS. Now in the reference architecture, we deal with everything in a very abstract concept so we don't identify a specific system, we identify the fact that it is an ITS system.

Your project-level architecture will be a true architecture description, would identify the specific system that's being deployed in your particular city with your particular equipment but we deal with things on a more abstract level. So the architecture description identifies the system of interest, as well as the stakeholders who have an interest in that system, and then we also identify the concerns that are held by the stakeholders.

So that gives you background. That’s kind of the problem side of the architecture. The architecture also includes the solution space. Solution space primarily considers, you see in the lower left, the architecture viewpoint. That architecture viewpoint are a set of rules that govern how I produce my architecture views, so it's just the set of rules that define how I'm going to document my architecture but both of those are part of essentially the description.

The architecture viewpoint is defined that those rules are subdivided into different types of model kinds, each with their own set of rules, and then those are realized in actual architecture models and they all create the architecture view. Now, as I mentioned, we have different views. How do those views relate to one another? Well, we have what are called “correspondence rules” that govern how I write correspondences.

So my correspondence rules are things like this information transfer in my physical view relates to a particular set of standards in my communications view and that's a rule that has to exist for every instance and then when I create those actual mappings between one information flow and one particular set of standards, that's an actual correspondence. So we'll go through this a little bit more pragmatic example-wise here in a second but that gives you a good background of why we're using the terms that we use within this presentation.

So when we talk about stakeholders, lots of different stakeholders, what are some of the major high-level concerns that they have? One concern is what relationships are needed for the entire lifecycle of a project, from planning to construction to operations to maintenance, and what are all of those relationships to make my system actually work?

The second is what functionality needs to be provided by my system, why am I building it, what's the ultimate goal? The third is what components will fulfill this functionality? So I have this functionality but how does the system actually piece together with products I can buy off the shelf. And then fourth, what interfaces do components need to support? So now that I've bought these different components, what

standards interfaces do they need to support? And then finally, what data is produced and can this data be shared? And if you look at those five major concerns, they line up really well with what we're doing within ARC-IT.

In fact, right now we have four major different views within ARC-IT. The enterprise view addresses that first concern of how do all of the interrelationships, how do they work among people to make the system work? The second view, the functional view, identifies the functionality of the system and how it all works. The physical view identifies the system components, the data that needs to be exchanged among those components. The fourth one talks about the communications view and the standards that are required to make each one of those interfaces work.

Now we're working currently on adding a fifth view. This came out of the automation concept for automated vehicles and really important that we start documenting the information that's developed, how it can be shared, and those things so that will be an information view that will be—you might start seeing some things added for it with the 9.0 or some 9.1 maybe but the actual content will probably take years to develop. But this whole thing forms the reference architecture with four distinct different views and the fifth coming in the near term.

To give you some perspective on why we ended up developing it based on ARC-IT, rather than one of the other reference architectures, is we provide this table that shows the different views provided by the different architectures that were out there and, as you can see, ARC-IT was the most all-encompassing as far as the number of views it provides. It's the easiest as used as a baseline. The good news though is when we started combining all of these architectures together, they were all very, very similar. We can map them into the components that ARC-IT had without a huge problem.

So with that, ARC-IT is the main reference term that we use. Technically, it refers to the website where you get this information and that's available at arc-it.net. It is the template solution for all ITS deployments and we say it allows customization. It allows customization by two other products—that we'll mention here in a second—but it's designed to be customized like that and that website hosts those other products—we'll talk about as well—as a series for other valuable resources.

One of the other products is called RAD-IT, this is a Regional Architecture Tool, so it helps you customize this generalized reference architecture into a tool or reference architecture for your particular region that customizes that architecture. It provides a detailed vision once you do this for your particular geographic region for perhaps 10, 20 years where you want to head with the ITS components in your region. They assist users in customizing ARC-IT and it's available off of the ARC-IT website.

The other tool that I mentioned we talk about is called SET-IT, this is the Systems Engineering Tool for the reference architecture and it's a key tool for creating the actual systems engineering content. So you can develop your con-ops from here and list requirements and all sorts of other things and really do a good job of documenting your particular project architecture and customize it to your needs.

You can add service packages as we discussed and everything else. It identifies every item—you're able to identify whether this is an existing item, a project item that will be added during the project, or if it's a future item that we have to be planning for and considering as we're building this project. It also defines the standards to be deployed for every interface that you've defined for the project, so it assists users in building their project architectures. It's also available from the ARC-IT website.

Well, that concludes learning module two and brings us to our second question. Which tool is designed to assist in developing a customized deployment architecture? The CVRIA; SET-IT; RAD-IT; or HARTS? So go ahead and make your choice and we'll review the answers.

Well, the correct answer is: SET-IT. SET-IT is the Systems Engineering Tool for assisting you in the deployment of your systems engineering details of your deployment architecture. The CVRIA was an

older U.S. reference architecture. It was limited it to connected vehicle aspects and its content has now been rolled into ARC-IT, which is always reflected—those updates are also always reflected in SET-IT and in RAD-IT.

RAD-IT is a tool that assists you in developing customized regional architectures, also known as planning architectures. And then HARTS was an international reference architecture that is being incorporated into ARC-IT 9.0, so the tool that helps you build your particular project-level architecture or deployment architecture is called SET-IT. It allows you to build those systems engineering details.

That brings us to learning objective number three: Link Reference Architecture Content to Standards. We'll be talking about a few things here: concerns addressed by the communications view, the elements of the communications view, and the solution stack. And we'll talk some about the OSI reference model, which is called the Open Systems Interconnect, OSI reference model. We'll talk about the ITS stationed architecture, which is an alternative viewpoint of that and then we'll talk about bundles, standards information, and gaps.

So a little bit about the concerns expressed by the communications view, before we talk about it at the very, very high-level, we're interested in what interface standards I need to support. Now we get a little bit more detail than our levels of concerns. So what is the purpose of this information transfer? Why is this transfer needed, where is an information transfer used, under what service packages, under what conditions do I exchange this data?

What are the characteristics of the information transfer? Does this always have to be available, what's the requirements on my systems to make this data always available, how regional of an information transfer is this, what's the latency requirements on this information transfer? What are the security requirements for the information transfer and then what protocols does my device need to support for interoperability and then finally, what risks are involved in deploying the solution? We'll talk about each of these a little bit more.

So, from an overview perspective, the communications view includes several artifacts for each information transfer, so when you go to the communications view, you'll see that at the bottom of the screen there and you'll have these different tabs across the top of the screen, so you'll have one for definition, one for correspondence links, one for—and in those correspondence links, the title is called Included in, and we'll discuss that a little bit more.

So correspondence, you might remember, was in our diagram previously of the official terminology but on the website, we just say Included in. Characteristics of the information transfer, security analysis of the information transfer, and at the very end, we'll then discuss the communication diagrams, which on the tab list is kind of in the middle there.

So we talked before about the information transfer is sometimes called a triple. This example here shows you one example of an information transfer or triple. The three components of that triple are the connected vehicle roadside equipment, the vehicle OBE, which is the on board equipment, and the intersection status, and sometimes the roadside equipment is just simply called RSE.

So if you click on that definition tab of the communications view, you'll see the definition as you would expect for that information transfer. It addresses the purpose of the transfer by identifying the information flow contained in the transfer, it identifies the physical object that is the source of the information transfer, and also the physical object that is the destination of the information transfer. And this down below provides an example of that. It’s a reasonably detailed but not exhaustive description of that information transfer but it gives you a fairly reasonable idea of what's being transmitted, what the source is, what the destination is.

Within the Included In tab, which we also said was correspondences, there's multiple sections so it identifies where the information transfer is used in service packages. So service packages were these diagrams that we talked about before that if you click—so in this particular case, this one information transfer is used in five different service packages listed in the lower left of the slide. In the lower right, you'll see if we click on one of these items, we see the diagram on the right, so on the Included In tab, you get the listing of different service packages that are all hyperlinked, so if I click on any one of those, I can see exactly where it's used.

Also on the Included In page is the set of triples where—sorry—is the set of functional objects where this particular triple is included. So I mentioned that the triple is also known as the information transfer; the two different names for it. In this case, the information transfer we're talking about is this intersection status. So in the lower right, you see that same blue highlight with the red box, that's the information transfer we're actually looking at right now and the two functional objects that are included that use this information transfer are highlighted in yellow.

So RSE intersection management and vehicle intersection warning are the two functional objects, so the hyperlink that's provided on the page will take you to a detailed description of that. So if you look at the very bottom left, that's a little bit of a snapshot of what the overview is, just provides a textual description of what that function does and then not only—so on the previous slide, you talk about what data is transferred in the definition of the information transfer. This description tells you how that information is processed within either end of the device. So now you see both the information that's transferred and then how it's used.

Finally, the Included In page also lists out the functional view data flows. So we talked about again, the correspondences between the physical view. Now we're looking back at the functional view and we're saying, "Where does the information derive from?" So we had a very high-level description of what this information flow content is. By these links you get to look back into the theory of the individual pieces of data of where it's being developed back in the functional view and that tends to be a much more detailed look at what data is contained in that information flow.

So those are the different aspects you see in the Included In view. The next tab we'll talk about is the Characteristics tab. The Characteristics talks about various types of information that's relevant to that information flow. We give it a time context, so how recent does the data need to be? Is this data that is collected over a day and then uploaded to Central? Is it archived data that's exchanged between centers or is it more real time data that's recent data that needs to be taken into consideration?

What is the spatial context? Is this data relevant to only this immediate vicinity or is this data that's shared across the entire metroplex or perhaps across the entire nation? Is there acknowledgement, when I exchange this information, do I have to have an acknowledgement that it was received? What's the cardinality? So when I send this data out, am I sending it to everyone called a broadcast or am I sending it to one particular instance of the destination and that's called a unicast? What's the initiator? So every information transfer has an arrow, source, and destination but that doesn't necessarily mean that the source initiated the transfer. It could be the destination requesting it and then the information is sent over and then authenticable and encryption are also attributes.

Which brings us into the security discussion. Those two characteristics, authenticable and encrypt, are derived from a security analysis and the architecture includes a complete description of not only what our analysis was but how we got there. Our justification for choosing the particular type of analysis and then analysis is done with what's called a CIA analysis; confidentiality, integrity ,and availability. So we look at each of those three characteristics, explain our analysis of it, and then give it a final rating. That rating then equates into whether encryption is needed or whether it needs to be authenticable.

Now we come to the Communications diagram, which is the final tab we'll talk about. The Communications Diagram in 8.3, which is what's currently up on the website today, gives our historic

view. This is the view that the architecture has provided for quite a few years now. It is based on the OSI reference model, the Open Systems Interconnect model. This is a major ISO standards used throughout information technology systems. It's also called a seven layer stack because there's seven layers that are contained within this model, from the physical layer of how I transmit bits and bytes over the airwaves or over copper wire or whatever, all the way up to the application layer which defines my messaging structure and how all of that is structured and its meaning.

However, one of the known aspects of that OSI reference model is that it really only defines the communications stack, not the things around it. So one of the big things that we've added is what we call the ITS application information layer, which is where we actually define the data that we want to put into messages that we then send down the wire and everything. The other aspect that we've had for some time is called a security plane which defines what standards we use to secure the information in that communications stack to make sure that we have a secure system. So that's what you see in ARC-IT 8.3 today.

What we're migrating to in 9.0 is based on what we had in HARTS. HARTS, you remember, was that international architecture. 9.0 will adopt that same methodology and it's based on ISO 21217, which is the ITS station communications architecture. It simplifies the OSI model by combining some of these different layers together, so the physical and data link layer are combined into what we call an access layer. The network and transport layer, which is where TCP/IP exists, is combined into a Transnet layer and then finally, we have a facility layer at top because what we discovered in practice, a lot of times the application and presentation and session layers kind of all interact together and form one unit, so we call that the facility layer.

And then just like what we had in ARC-IT 8.3, the ITS station communications architecture also adds over on the right, the security plane that we had before and on top, we see an ITS application entity. That application entity is what actually is the logic that drives everything so now it's not only the data, the information layer that we had in 8.3, but it's also the performance characteristics, the processing characteristics, and everything about that application that decides how and when I send messages and all of that is part of that ITS application entity on top.

Over on the left is the management. So this is new that's not actually in 8.3 right now but it is—will be in 9.0—is the management entity. That management entity is responsible for maintaining all of the data and basic configuration of your communication stack. So now I can configure which ports I want to use and all of those details and have that as settings within my device. So that was the one thing we've added really since 8.3, slight tweaking of some of the other terminology but everything else is essentially the same. What would that look like in 9.0? Well, we'll fill out the different standards at each of those different areas of this model and, as you'll see, that each of those standards will give the standard number plus, we'll now have that little information icon that allows you to identify more details about it. And if I click on one of those information icons, I get a screen—something like this—that will identify not only the standard number but the specific version that our analysis was based on, the full name of that standard as well as a little bit of a description, and a link to its website.

Now you'll notice in the bottom here, the access in our example we have these things called bundles. This is a wave subnet so the wave bundle that we talk about here, we can now—that's a hyperlink that I can click on and it just expands the content of that bundle. So a lot of times rather than listing out every possible standard there and confusing people, we've created a short form, called it a bundle. You can still get access to all the details if you want to but it simplifies a view of the diagram to keep things as simple as possible.

So that bundle in this case contains three different standards. You can still get information about each one of those standards individually. Those standards can be listed as being required, optional, or alternative. So the idea is that in this case, all three of these are required. Sometimes you'll have an optional standard there in the group, in which case it's just an option.

The other scenario is you'll have a list of alternatives. The idea there is you only select one of the alternatives but you have many alternatives and that's typical of most of our internet stacks, where you can use any sort of lower layer technology and it's up to you to pick which one we want there. Sometimes we've taken shortcuts and we've not tried to identify every possible technology. We just identify that alternative at the high-level. In other cases, we are very specific about which alternatives are allowed at that level.

The other key aspects of 9.0 is not only are we defining which standards are being used but we're now also identifying the issues associated with those particular standards and that includes gaps, as you see in the upper right area, and also overlaps, as you see down at the bottom in the access layer. So if you click on any one of those icons, then you get a description of what that particular issue is. The issue is either a gap or an overlap, so issue is kind of a generic term.

So in this case, if we click on the overlap icon, we see that 1609.4 is the current standard within the U.S. for 5.9 gigahertz spectrum but as of late 2019, the FCC (Federal Communications Commission) has started considering reassigning portions of the spectrum for different uses. So whether or not that will be allowed to be used a year from now, we don't know right now, we've identified that as an issue and a page that has—this will also identify LTE-V2X as an alternative solution to this and we'll talk about that in a little while.

So for all of the gaps, you can find details about what the gap is, what the issue is, and then once you know what the issues are for your particular standards, now you can start managing them and those issues equate to risks for your project.

So Communication Diagrams and ARC-IT 9.0, which was developed from HARTS, identifies all known potential solutions and we list out these solutions on the page and we order the solutions with the fewest number of issues on top. So as you go down the list, you should see more and more issues and more severe issues.

So in this particular example, we have signal control messages over WAVE. WSMP is the highest-rated U.S. based solution but because there's this overlap between WAVE and C-V2X standards, or LTE-V2X standards really, there's a little bit of a debate there and overlap between those two. They score lower than what Europe has because they've already decided to use BTP/G5 within their region. So right now, they have less issues in their standards than we do here in the U.S. You look at different transfers all over the place and you see those orders change.

So in summary, we have a communications view that addresses various concerns. The purpose of the information transfer as addressed in the Definition tab, where the information transfer is used is identified in the Included In tab, the characteristics of the information transfer are described in the Characteristics tab, the security requirements are described in the Security tab, and then finally, the protocols that you need and the risks for your deployment are both defined under the Communications Diagram and that's shown on the ITS station architectural model.

That completes learning objective number three. And our third question is: Which of the following OSI layers is not part of the facility layer? Option A is session layer; option B is application layer; option C is presentation layer; and option D is data link layer. So go ahead and make your choice and we'll review the answers.

So the correct answer is: data link layer. The OSI data link layer is contained within the access layer of the ITS station architecture. The session layer is a part of the facility layer. The session layer takes care of handling sessions among the different units. The data link layer by comparison is dealing with how I exchange data from one node to another node on a complex network. The application layer is also the facilities layer and it identifies the actual message content and the presentation Layer is part of the facility layer and it defines the encoding rules for that application layer message content.

So learning objective number four, we'll talk about identifying known risks with standards and deal a little bit more about how those issues arise on that communications view. We'll talk about the issue severities, the different types of issues you might see in the analysis, a practical example where we'll step right through a practical example so you can see how it might impact your project, and then also explain how you could provide feedback and improve the architecture content as well as explain how standards developers could use this information to improve their standards.

So with that, the issue severities are shown on this page. You saw some of these icons on the previous slides. The gap icons always say gap and the overlap icons kind of use an abbreviation there. The low severity icons are that blue circle and that basically means the technology, the standards are probably available for full-scale deployment but the deployers, the implementers should be aware of this issue and take it into consideration.

A moderate-level issue might indicate that it's sufficient for pilot deployments but not really recommended for full-scale deployments, primarily because at some point, you'll need to go back and issue some sort of correction which is going to affect all of your deployed devices. So if it's a pilot deployed small scale, no problem, but if you deployed it across the entire city, now you might actually have a lot more maintenance on your hands to try to retrofit everything.

A high-severity indicates that it fails to provide a base level of interoperability and security, as recommended for pilot deployments. So really, we're not recommending that you even do this as a pilot deployment yet due to security reasons or due to base-level of interoperability reasons. And then ultra-severity is kind of our special case of we haven't even started standardization efforts so anything you do is going to be project specific and you'll have to complete your retrofit afterwards.

Now I will say this is kind of a theoretical analysis. They're suggestive and they are merely recommendations as to how severe it is. They're used primarily as a guidance to our efforts to say how severe are the problems on this list. They're not supposed to be taken quite literally in practical scenarios, and we'll talk about some examples of some challenges you would have if you took these literally from a practical perspective. But that’s kind of our theoretical analysis of how we develop these levels.

So some examples of the types of issues it would lead to different levels of issues. So if ultra-severity issues are things like the standardization has not even been started, there are no data or messages defined for a particular area, performance and functionality standards have not been defined at all, use case is completely not considered.

High severity. You see some of the same concepts but our analysis decide if they were somewhat less severe, so the standard typically exists in the cases are under development but we still know there's major problems, such as the data or messages that are pretty critical for the flow aren't defined, the performance and functionality requirements are not defined and then you get down to the bottom, security not provided, data communications profile pairing.

So that's a case where we have a standard that defines data for this flow, we have a communications standard that says how I exchange information across this type of link but the data standard was not designed for that communications environment so there's a pairing problem that a lot of the work has been done so far but there's still customization that needs to be done. So these are high severity that ideally in the theoretical world, you'd like to see all of these problems solved before you deploy it. We'll talk about some examples coming up where, yeah, just in practical reality you have to deploy before we're completely done but we want you to be aware of these issues. That's why we leave these here.

Moderate severity deals with issues like that there are noteworthy problems that should be known and you should be concerned about but you can still go ahead with pilot deployments. So here you see we've changed the wording a little bit, the data or messages not fully defined, so there's at least some definition of the data but it may not apply to this particular use case or it may not be sufficient of this particular use

case. Performance and functionality requirements not fully defined, use cases not fully considered in their design, the security is inadequate, there's something there but not fully there, data formatting issues, it's not an open vetted standard or there's an overlap of standards.

And then finally low severity. A lot of times this is a guidance for implementations or it may be optional features, which ones you should use. Data accuracy issues, giving some advice on those, identifying specific security options, identifying the fact that there's a current effort to update the standard or it's a relatively new standard without much experience, so those are low severity type issues.

So let's look at a practical example now. How do we take all of this information that we've learned now and apply it to a real-world example? So let's look at maybe we have a project that we want to deploy transit signal priority using connected vehicle technologies. And we only want to provide priority transit vehicles that are behind schedule and we use onboard logic to determine when they request priority. So to do this, we go to our listing of service packages. One of the service packages under public transportation is transit signal priority and we would select that particular service package.

This is the diagram that would come up if we selected that service package. And in our one example, we're going to only select one of the information transfers shown on this page. So we'll select the intersection control status as our example information transfer. So if you click on that within the ARC-IT website, you'll be taken to the communications view. We've discussed how that communications view includes several different tabs, we'll click on the Communications Diagram to identify the gaps and the issues or the gaps and overlaps for our particular communications standard.

In this case, you see on the left-hand side, we have several different options from different regions of the world. The highest-ranking U.S. option happens to be one based on the SNMPv3. So if we select that option, this is the communication stack that you would see. You'd see the NTCIP 1202 as the application entity standard and that application entity standard, if I clicked on that information icon, I would discover that this is the actuated traffic signal control standard for NTCIP.

So that makes sense. We're talking about—the information transfer we're talking about is the intersection control status, so that's the actuated traffic signal control signal. That makes sense. Moving down the stack, we see this ISO 15784-2 part two standard and if I clicked on that information icon, I would discover that this is how you use SNMPv3 within ITS which match the simple network management protocol.

Then finally in security, we have internet RFC here which happens to be how I apply the transport layer security for SNMP and then we have a standard set of field technologies I can communicate with and subnet technologies as well and the management standards. So all of the standards are there. You'll note that we have a couple of gaps associated with the solution.

So before we get into looking at those issues, let's consider the different scenarios that might occur as I click to that communications diagram. It may be, as we saw in the previous slide, it may be that I see multiple different solutions for my region and then I have to look at those different solutions and decide which solution I want to choose from. It may not necessarily be the one highest on the list and we'll talk about why here in a second. Second option is that there's only one solution for my region. The third option is that there's no solutions for my regions but there is one for some other region, and then the fourth option is that there's an interface that's labeled out of scope. So we'll discuss each one of those in order.

So the first case is multiple solutions for my region. So let's say I go to a page and I see a listing of different options and I have SNMPv3. SNMPv1, you see in the upper right-hand corner, SNMPv1 with TLS and SNMPv1 by itself. What we would do in this case is look at each one of those solutions and determine which one is most appropriate for my project deployment, recognizing that the architecture only attempts to identify the issues. It is up to the individual project to identify how those issues might impact my project.

So in this case, if we look at the issues in these three different solutions with SNMPv3, we discover that NTCIP 1202 was designed and written for SNMPv1. So if I want to use my 1202 standard with SNMPv3, I somehow have to translate my data between those two formats which can be done—there are rules for how you do it—but that's a task that has to be done. And it may mean that all the devices sold today out in the field don't support this SNMPv3. So while in theory, it's the least gappy of everything. The practical reality is you may not be able to buy a device that actually supports that interface.

The minor issue on SNMPv3 is just a note saying that we should couple SNMPv3, which TLS and that's a just a reminder when you're doing your procurement specification, that's what you would need to specify. The second option, SNMPv1 with TLS indicates that application level security is not provided. The reason that SNMPv3 is preferred is because it actually does application security, which means even if my central system gets corrupted by a virus that that virus cannot penetrate my end application; the data that I send is still secure.

In this case, if you use SNMPv1, if a virus impacts your central system, it can now actually start sending fake SNMP messages to my field devices and gain control of those field devices. So that's a security level, we call it a moderate level threat because we really want to see that application level security. Use of TLS is not vetted by industry within SNMPv1 so this is industry conventions. It's not used today or it's not recommended today by convention, i.e. the standards community has not recommended it, however the architecture team is essentially saying, "You need to do something, this is our best guess as what most people would do."

And finally, the other problem; that stop sign gap. The high-level gap is at the SNMPv1 is not allowed by the RSU specification. So if you think back to our information transfer, we're sending data from your traffic signal to your RSU but according to the RSU specification, it doesn't support SNMPv1, it only supports SNMPv3. It does that because we want to protect that connected vehicle environment from being corrupted, right? So that RSU spec was very, very strict about security. It said it only supports SNMPv3 which means it won't support SNMPv1.

So now we have a practical issue of “my field device might only support SNMPv1, my RSU might only support SNMPv3 and I have to work with the marketplace to figure out how I get two products that can speak to each other.” So the third option is two high-level gaps. SNMPv1 doesn't provide any security at all and SNMPv1 is not allowed by the RSU specification. So in essence, we're recommending you at least put in TLS. That’s fairly easy to do, it shouldn't cause a lot of problems. Ideally, you want to migrate SNMPv3, but practical reality is we don't know if there's going to be products out there to support that when you have to deploy your project. So that's the tradeoff of—the architecture can only go so far, you have to analyze how those issues impact your particular deployment.

So the best solution even still has a moderate gap and products may not exist. Some solution is needed to communicate to traffic signals. Our best solution has a moderate solution saying, "Well, okay maybe for pilot deployments," but let's face it, this is technology that you have to communicate with your signals all the time. You might have practical realities that you need to deploy this and then that's for your project to determine what to do is best. ARC-IT is intended to provide options, the agencies are responsible for deciding what should be deployed.

So with that, the other possibilities here is rather than having multiple solutions, you only have a single solution for your region. Things become a little simpler here. You pick the one for your region most typically, you might look at those that are available for other regions.

If your region only has either no solutions or none-data, then well, you may be able to look and see if there's a solution for some other region. So it may be that Europe has a standard for this particular interface, even though the U.S. doesn't. In which case, you might be able to leverage and find products that met that European standard. You then just have to integrate, "How do I actually get the source data

that I need for this interface to work?" So there might be some challenges there but it at least gives you a starting point.

Most times you'll only see this on a couple of cases. One either information flows that are not commonly used today or they're emerging service packages. So one of the things to consider is, “Is this flow really needed for your project or is this desirable, how much?” Recognizing that if you try to deploy this flow, your risk level has increased a fair amount, so consider what other solutions might be available.

The final area is you might see something that's called out of scope. The out of scope flows things are thing where the ITS community is not going to be the driver of what the standards are. So a classic example of this are the architecture includes interfaces to financial systems. Those financial center standards are not going to be defined by the ITS community and we're not going to be the ones driving updates to those or anything else. Those financial community will decide when they update their standards and, more than likely, what will happen is you need to talk directly to your financial institution and figure out what standards they want you to use.

So if you go through this and you find that some of these standards don't identify the gaps you think should be there or some of the gaps that were listed probably have been overcome—the standards have been updated or whatever—you can actually provide feedback to the architecture and not only assist yourself, but you'll get a response back from us but you'll also be able to help all the others across the world in ITS to make this architecture a better product.

So comments are very much welcomed and that can be a request for clarification, request for additional solutions, request for alternative architectural designs, remove issues that have been resolved, and identification of issues not previously identified, all of those we're more than happy to accept to try to improve this architecture.

The way you do that is on the ARC-IT website. There is actually a high-level menu option for connecting the architecture team. You can go to that site, fill out the information, and submit that and you will be responded to. Now with that said, don't expect a change in the architecture overnight. The architecture is rigorously version controlled. All of your requests that come in are managed. We first evaluate them, we then decide when will we address this issue—it may not even be in the next update, it may be a couple of releases from now depending on how large of a change you're requesting.

So it'll be an assigned to a release and we will respond letting you know what's happening. And then once we get to starting to update that release, we'll start implementing it and finally we'll release it. So this process is, you know, it takes months between different releases and if you're some major update, that might be some major future release, which could be years away. But most of the changes of just changing gaps, whatever, that should be reflected within a few months, if not earlier depending on when the request comes in and the cycle.

So now those issues are great for deployers. Those deployers are able to identify the risks on their project. But simultaneously one of the key reasons we started recording all of this information as well was to assist standards development process to be able to prioritize what gets addressed first. So we have a complete list of all issues with all of our standards and we now have hundreds, nearly a thousand standards in our database, and literally thousands of issues that we all need to address.

So it's very important that we try to start prioritizing these and when we start updating any one standard, make sure that we have a list of all known issues that need to be addressed with that standard. So it's a very valuable tool for the standards developers as well, which is yet another reason why if you see a problem of what's listed on our architecture site, please let us know because you're feeding that into the standards process. So even if it's just an issue of, "We saw this problem with a standard," you can let us know, eventually that will feed over to the standards process.

So issues in ARC-IT, service resource to the standards community, comments from users definitely help. It improves our list of known issues and provides that really important real-world feedback and we have a database used to prioritize all of these issues. By working with our ARC-IT team, the standards developers can ensure that information about their standards is maintained and up-to-date.

Well that concludes learning objective four and our question is: What does a moderately severe issue indicate? A) the issue is expected to be resolved within two years; B) the solution is not recommended for full scale deployments; C) users should delay their project until the issue is resolved; or D) the solution does not provide adequate security. So go ahead and make your selection and we'll discuss the answers.

So the correct answer was: B) the solution is not recommended for full scale deployments. While agencies may use their solution in their projects, full scale deployments are likely to encounter expensive upgrade efforts once the issue is resolved. Answer A, the issue is expected to be resolved within two years, is incorrect. ARC-IT does not attempt to estimate when any of these issues will be resolved, we can only gauge how big of an impact that resolution might have on your system.

Users should delay their project until your issue is resolved is also incorrect. Agencies must consider their own competing demands to determine if the risks are worth their deployment. And then finally, D was incorrect, which was the solution does not provide adequate security. While not providing adequate security is a moderate level gap, there are other types of moderate gaps as well so there's not a direct correspondence there in each direction.

And that brings us to learning objective number five: Providing Recommended Resources to Learn More. So we'll just provide some links to architectures, links to courses, and to toolsets.

So there are a variety of links to the different architectures, as we've discussed, really ARC-IT is the best resource. We're moving into that by the middle of 2020, we should have that updated to ARC-IT 9.0, which will include content from all the different regions and be a really solid product.

Architecture courses also on the website, all this information is available. There is much more detailed information available for three different topic areas: ITS architecture, software tools, and systems engineering. Those training and resources are available via web-based training if you want to, there's also onsite training that can be arranged and then finally, there are workshops that can be arranged to try to customize things to your particular region.

Also on the website under architecture resources are the tools that we talked about, the RAD-IT which helps you define your regional architecture, and the SET-IT, which is the systems engineering tool to help you define your project level architectures.

With that, our final activity. What types of training are advertised on the ARC-IT website? Your answer choices: A) systems engineering; B) software tools for architecture; C) is ITS architecture; and D) is all of the above. So go ahead and make your choice and then we'll review the answers.

So the correct answer is all of the above. Systems engineering is available for training, also the software tools for architecture as well as ITS architecture training available. So you can go online, find out the type of architecture training you're interested in and get access to that.

So we've talked about explaining system architectures, comparing the ITS reference architectures, we identified how you can link direct reference architectures to the actual standards and then identifying the known risks with those standards, and then we also provided a list of resources where you can learn more about this topic.

And that completes the curriculum for determining risks within your deployments which included I101 Using ITS Standards, an Overview, as well as this course, Determining Known Risks with Standards in

Your Deployment. So with that, thank you for completing this module and we look forward to your feedback. Thank you.