Module 17 - T311 - Applying Your Test Plan to the NTCIP 1203 v03 DMS Standard

Webinar Transcript

Shelley Row:

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 for them. This module is one in a series that covers practical applications for acquiring and testing standards-based ITS systems. I am Shelley Row the director of the ITS Joint Program Office for USDOT and I want to welcome you to our newly redesigned ITS standards training program of which this module is a part. We are pleased to be working with our partner, the Institute of Transportation Engineers, to deliver this new 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 will tell colleagues and customers about the latest ITS standards and encourage them to take advantage of the archived version of the 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 which improves livability for us all. You can find information on additional modules and training programs on our web site 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. Thank you again for participating and we hope you find this module to be helpful.

Nicola Tavares

Today's training module is T311, Applying Your Test Plan to the NTCIP 1203 Version 03, DMS Standard. The target audience for this module, are engineering staff, operators and maintenance staff, system integrators, device manufacturers, and testing contractors. Your instructor is Patrick Chan. He has been on the NTCIP DMS working group since 2000, he has been involved with the development of several ITS standards, the last several years, including NTCIP 1203, version 3, which is the object definition for DMS, and added test procedures to the standard. Patrick has 20 years of ITS experience, and he is a senior technical staff at Consensus Systems Technologies.

Patrick Chan

Okay, hello, everybody, you should be able to see my screen now. My name is Patrick Chan, as Nicola mentioned. Just a little bit more on myself. I've been involved with the dynamic sign working group, since 2000, originally, first as a public agency, I had worked for a public agency that was interested in installing and implementing dynamic message signs, so that's how I first got involved with the working group. Further down the road, though, I left the agency and now consultant, where I work on regional ITS architectures and ITS standards, but I still remain involved with the DMS working group, and was most recently the consultant for the latest version of the standard, NTCIP 1203, version 3, and as Nicola mentioned, we added testing features to the standard. So that's just a little bit about myself.

This is a screen showing the recommended prerequisites for this module. It is assumed that everyone that's participating have already completed these modules. If you haven't, it's highly recommended that you go back and take these modules before completing this specific module, T311, because it provides, these prerequisite modules provide a little bit more background, a little bit more detail about testing in general, and a little bit more about the dynamic message sign standards, so what today we're going to do on this module, is just focus on applying things that we've learned in these previous modules, and applying a specific need to dynamic message signs.

Just to go over these modules briefly, T101 provides introduction for the need and applicability of ITS standards testing, so why do we do ITS standards testing? T201 is introduction for a need to do test plans, or developing these test plans. Why do we need to develop a test plan to test these ITS standards? T202 identifies the key elements of the test design specification, test cases and test procedures, and we'll go over the vocabulary again, later, further down the modules, but the focus, in this module is developing test design specifications, test cases and test procedures for dynamic message signs, and we'll provide a couple of examples, how to develop, use, provide a couple of examples of how these dynamic message signs can be used in these test design specs, test cases, test procedures.

A311a provides and discusses some of the user needs that are supported by the dynamic message signs standards, specifically NTCIP 1203, and A311b discusses the requirements that support it by the same standard. These latter two modules help participants understand the DMS standard a little bit more, how-- what's in it, how to use it, how to specify your specifications, your implementation when using this standard. This slide shows the context of where A311a, A311b belongs. They're both part of the curriculum path of systems engineering process. The system engineer process, the curriculum path of system engineer process involves I-101, introduction, which is introductory module of the ITS standards A101 provides some key information on procurement strategies for systems using these ITS standards, while A102 provides introduction, how to select user needs for standards.

The second line, A201 provides guidance for how to select the appropriate ITS standards for standards based systems, and as we mentioned, A311a provides information how to use NTCIP 1203, to implement a DMS system, based on the user needs. A311b, based on the user needs that we've selected already, how to identify selected requirements from NTCIP 1203, and then acquire a dynamic message signs system that performs to the standards. So all this is, this is the systems engineer process, just as a brief reminder, the reason why we do system engineering process is to-- it's a means of realization of successful systems, it's a process, so that we can properly define the customer needs and required functionality early in the development cycle, to properly document the requirements, and then, based on those requirements, how do we see what designing, implementing and testing your system that uses these ITS standards. This is the curriculum path for testing. So this provides a context for where T101, T201 and T202 belong.

So this, again, the first module, introduction to testing, the second one, how to write a test plan, the third one, the third module, T202, is just an overview of how to write-- what belongs in a test specification, test case and test procedures. And finally now we've reached T311, where now we're going to, based on this information, what we've learned so far in testing, how it applies to testing

dynamic message signs using the standard. So that's just a quick overview of some of the prerequisite modules and where they fit in, in the whole picture for dynamic message signs. For this particular module, these are the learning objectives. The first one, recognize purpose structure and content of a well-written test plan. Essentially, why do we develop test plans, what's the purpose of it, why do we want to create it, and develop a test plan for testing our systems? The second one describes, within the context of a testing lifecycle, the role of a test plan, the testing to be undertaken, essentially, we're asking the question, when do we test, what exactly are we testing when we develop a test plan? The third learning objective is, describe the application of a good test plan, to dynamic message sign system being procured, using a sample DMS test plan.

So in this module, we'll go through some examples of a test plan for dynamic message signs, so this is a high level example, but at least it will get a general idea what goes in it, using specifically dynamic message sign examples. The fourth objective, identify key elements of the NTCIP 1203 standard, relevant to what is covered in the test plan. So we're going to, using the dynamic message sign standard, NTCIP 1203, we want to identify what parts of it, what sections of the standard can be used to develop a test plan. As we'll see in this module, there are certain tables that are in the standard that are specifically put in there to help you with your testing. So you know, okay, based on my user needs, based on my requirements, how do I know how to test it, how do I know what to test for, how do I know that I'm performing to the standard? So those tables are in the standard, and we'll point them out in this module, and show you some examples of how to use those tables. And finally, the fifth learning objective, to walk through the process of adapting the test plan in the context of the needs and requirements of the dynamic message sign that have been selected by the user. So again, based on the user needs, based on requirements that you've selected for your specific implementation, we're going to show you how to develop a customized test plan for your agency.

So these are the learning objectives, for this specific module. So here's our first activity. So using the chat pod, we'd like to ask the audience to answer the question, why do users perform testing? We want to start the module by reviewing, why do we perform testing? So I want to get an idea from the audience from the participants, why do you perform testing? What is the reason that you go ahead and do the testing. So if you could use the chat pod, just please answer the question. Provide some examples why your agency performs testing.

So I'll give everyone a few seconds to go ahead and answer in the chat pod. So we've got some answers already, to-- some people to verify-- one answer is, verify that you've got what you want in a project. Another one, to make sure the system that's implemented meets your specification. Another one that proves that the contractor has satisfied your standards and your specific requirements, and they're all right, actually. So sometimes agencies put that-- do testing just to meet a payment milestone, so, by completing testing, they get paid, and to verify that the systems work, but put in technical terms, we do perform testing to verify that the system meets the requirement, the procurement specification and satisfies the requirement, meaning your specification has multiple requirements for your system, so we want the test plan to perform tests and make sure, okay, the vendor, the system that they provided you, does it satisfy all the requirements? Essentially asking the question, was the system built right? But also, another reason why we do testing is to validate that the system satisfies the user and operational needs,

meaning okay, they built the system, but they did build the right system? Does it really satisfy the user needs? The idea being that, hey, there was a reason why, there was a problem, there was a need for putting out the system to decide if it fulfills that need. If I'm trying to solve problem A, but the system doesn't solve-- help me resolve the problem, A, what good is the system? You've built the system, it might satisfy-- I mean, excuse me, you might have built the system right, meaning you satisfied the requirements, but if it doesn't help you solve problem A, you've essentially built the wrong system. And unfortunately, we, on occasion, do see that, where, you know, everyone says, oh look, and say our system does all this stuff, but guess what, it didn't solve your problem.

So you may have built the system right, but you built, unfortunately, the wrong system, because it didn't resolve the problem that you were trying to solve. Another reason why we might-- why this is important, testing is important, is to test for conformance to the standards. And this specific reason, example, NTCIP 1203. That's one of the reasons why we want to perform testing, did we conform to the standard? One of many key reasons for these ITS standards, is because we want to achieve interoperability, meaning that we want the capability to build a system, where we're not tied in to a specific vendor, whether it's their hardware or the software or communication systems. One of the purposes, one of the key purposes and goals of these ITS standards, is, we want to achieve interoperability, so that, once I have my system in place, I can use different vendors' software, I can use different vendors' hardware or whatever they're providing you. When, for example, I keep using the example, but it's an applicable example, the example Wi-Fi. Wi-Fi provides interoperability because it doesn't matter which laptop I have, who built my laptop, whether it's HP, or Dell, and it doesn't matter where I try to connect to the internet, whether it's at the airport, or one of the lounges or Starbuck's. If my computer, and if the coffee shop conforms to the Wi-Fi standard, I know I will be able to connect to the internet. I know I will be able to get my e-mail, if we all conform to the Wi-Fi standard. So that's an excellent example of interoperability, Wi-Fi, so that's what we're trying to accomplish with these ITS standards, so that agencies can procure systems that are interoperable, so that's why we perform the testing.

This testing-- or one of the reasons why we perform the testing. This testing allows us to check A, did my implementation conforms to the standards that I'm trying to be interoperable with. So this is a review of the systems engineering V diagram. Testing is actually a big part of the V diagram, it's along the right side of the V. And the right side of the V emphasizes testing throughout this part of the diagram, and includes both verification, compliance with the requirement, and validation, which is conformance to user needs. So along the right, also we'll see in the numbers, you'll see what we normally call these different types of tests. There are the protocol type tests, design approval tests, factory acceptance tests, incoming device tests, site acceptance tests, and burn in and observation tests. So just wanted to give you an idea of where your various tests, this is what agencies normally call them. I just wanted to show example where these tests fall in the V diagram.

So let's start from the bottom. So on the left, we are already in the Subsystem process. We had time to develop the concept of operation user needs, we had done requirements, we've done design, now we're implementing it. So the vendors are implementing their hardware, software solutions, so now that's when we begin our testing. The first type of testing is called unit device testing, and you can think

of it as testing the dynamic message sign by itself. We want to verify with the unit device testing, that the dynamic message sign, by itself does meet and satisfy, fulfill all the requirements that we have for it. And sometimes that might be known as protocol testing, or factory acceptance testing, or when the test arrives at your site, so might perform some testing there. The next type of testing is called subsystem verification, where we try to test the dynamic message sign in its immediate environment. So for example, we might install the dynamic message signs on the sign gantry, and when you put the sign controller, and install it in the cabinet box, so now we want to see how the dynamic message sign test works in its environment. We want to make sure it still fulfills all the requirements that we have for it, when it's connected to the cabinet, when it's connected to its modem, for example, or a power supply, so we want to verify its functionality, and as it is installed.

The next step is then, probably system verification and deployment. So now we've hooked it up to our system, the hardware system, now we may want to test it out, hook it up to the remaining systems, for example, your TMC software, so now the system verification deployment tests the whole entire dynamic message sign system. We want to verify-- we've already confirmed that the dynamic message sign works by itself, we confirmed that it works around its operating environment, in the field, for example. Now we want to see how the dynamic message sign system works overall, so we'll test it using the TMC software. While these tests are used to perform verify that the units are produced and delivered and installed, we want to verify that all the requirements are fulfilled, and usually this is also a performance payment milestone. During this point, we also might do a burn in test, so now the users will actually use the system. Oh, can I control the software, excuse me, and control my dynamic message sign, if that's one requirement. Can I upload and download messages, for example, can I see what's on the dynamic message signs? Can I perform the diagnostics that was also one of the requirements. So that's a burn in and observation test. We actually use the system to make sure that the functionality is still available, still there, and that works as expected. The last step, though, is validation, which is part of the latter part of the burn in and observation test.

At this point, you want to check, okay, we want to confirm that the system, as built, does satisfy the stakeholders' stated needs, meaning, A, did we solve the problem that we set out to solve? So again, it's not checking whether you built the system right, but that you built the right system, and the system is considered validated when it's approved by the agency and the key stakeholders, it might be your operations people for example, they have these-- they want to make sure, hey, yeah, we had a problem, wanted to be able to provide our travelers with information, can we actually put the messages up now? It might be your maintenance people, for example, we had some maintenance issues, wanted to make sure we are able to perform the diagnostics that we need, so that we maintain the system, make sure that it's operating properly. So it's also a chance, well, when we can correct any problems that might have been any anomalies that might have been detected. So now we're trying to finalize the app, accept the system, solve any problems that we may have, so that might be also part of your validation in final testing.

So now we're going to review a little, what's in a test plan, so a test plan is a high level document that answers the following questions. Now what it tells-- It tells the testers and people involved in the testing, what is being tested, how is the item expected to be tested when we go through the testing?

Who is responsible for doing the testing, what are their roles, in what detail is the item to be tested, meaning, what level of detail? What are the test deliverables, so at the end of testing, what is being submitted by the testers or the vendors, or the agencies, even? And answers the question, when is the testing going to take place. So test plans, if you want to find out more information, is defined in IEEE 829, is actually covered more in detail in the module T201, but we're now, over the next couple of slides, we're going to go to each of these questions and answer each of these questions over the next couple of slides. So the first question about test plan, is what-- it answers the question, what is the item to be tested. So it identifies the scope of the test plan. For example, is it just the sign that we're interested in testing, or is it an entire system? Will it include the interface with the existing TMC software, for example? Sometimes you may say, no, it's individual sign, I'm going to use the vendor's software so I don't need to test it and I use the TMC software, or maybe you want to, so that's what one of the questions that the test plan will answer. Are there different types of dynamic message signs? For example, there may be a blank out sign, or you might be purchasing variable message signs, possibly both, as part of the same procurement spec. So the first thing the test plan answer, is what's being tested, what's included in the testing?

The next question is, how is the item to be tested? So we're setting up the testing. It identifies what environment are we going to be doing the testing? It might be just, for example, be in the factory, if you've got a spare dynamic message-- portable dynamic message sign, for example, you may not want to test it in the field, and you just might want to test it in the factory. How are you going to perform the testing? The graphic on this slide shows an example of how we may want to perform the testing. You may have some test software running on a separate laptop, communicating somehow with the dynamic message sign system. You may have a data analyzer to see the information that's going back and forth between your dynamic message sign and the test software. So we're answering, we're trying to answer the question, how is it going to be tested?

For NTCIP, we usually talk-- we're really focusing on the interface, meaning, what data is going back and forth, the communications between the dynamic message sign and something else. So that's what usually the focus is on when we do this NTCIP testing. Next, we ask the question, who is going to perform the testing? So we identify the roles and responsibilities for vendors, for the agencies, for the engineers, the consultants, what their roles are, who is responsible for doing what. Who's going to design the test plan, who's going to prepare it, who's going to ask the questions, make determinations, who's going to sign off on the testing, to say, yes, the testing is being completed. So we want to resolve the roles and responsibilities of everyone involved in the testing. This is just a note, the last bullet is, just recall, there's potential conflicts of interest between roles and responsibility and the different people involved with the testing. The vendor is interested in trying to get the testing done as quickly as possible, so they can get paid, while the agency is more interested in making sure that, hey, we want to make sure that this system that we are procuring, does what we want it to do, but to make sure that it's going to work for us as long as possible. So that's why we want to clearly define in the beginning, what the roles and responsibilities are.

Next question is, in what detail is the item to be tested? And this is always a struggle with testing. What level of detail, what level of information, how detailed do we want to do the testing? And that's always a

struggle. And the answer is, it depends. But you know, we want to specify a minimum degree of comprehensive-- as much-- the minimum degree of detail as possible, so that we can make sure that we fulfill all the requirements. We also want a level of detail so that we can specify all the criteria that the requirements require, so that's the level of detail so that we can permit identification of major testing tasks and properly estimate the amount of time to perform each task, so that's probably the level of detail that we're interested in. So, next question we're trying to answer in test plan is, what are test deliverables? So you wanted the test design the test plan. Hey, at the end of the testing, what is it that everyone is supposed to submit to each other, provide to each other, what documentation, what reports are supposed to be created, what gets delivered for milestones, for payment milestones? Who submits it, and what are requirements for each deliverable?

The test plan also answers, when is the testing going to take place? It's not just always the specific dates, but when submittals are provided, when, how much time is required to perform each task, up to provide comments, to resolve issues that might appear during testing. So this is another thing that the test plan will answer. So now we get to the case study. Essentially, this is an example of a test plan, a high level test plan that you might create for a dynamic message sign. So this, again, is based on the IEEE 829 standard. The first thing you may have on your test plan is a test plan identifier. So it's identified so you can refer back to it later in the future, and so it's a unique ID, so it might be, for example, the date,-1998 the test plan number, the date that has the number, that has the date that you performed the test plan, or it might have just specification number, the procurement number. So you just want a unique ID for each, for the test plan, so that's in case you have multiple versions of the test plan. Introduction, you want to define the purpose, so it might be, verify compliance to procurement number 11-xxx and verify conformance to NTCIP 1203, version 2, so you want to identify what and why are we performing this testing. Test items, so test items, you want to say, you want to mention, and what is it that I'm exactly testing? You may have the blank out signs, for example, you may want to test a VMS sign, variable message signs that you're purchasing under this contract, so you want to identify each of them. You might have portable signs, for example, so identify that. We also want to identify what software that you might be testing, if you're including testing the traffic management center software, so it's important to note, also what build you're doing, so that way you realize, hey, you know, if we change the build, meaning the software has been updated, you want to know, oh, wait, when we tested it, we used version number 3, but now we're in version number 4, do we have to retest it? Is that a problem?

So we just want to be very clear on what you're testing, what version of the hardware, software you're testing. Next, there are features to be tested. So what is it that you're exactly testing for in your test plan? The previous two modules, we talked about protocol requirements list, that's in the standard, where, in completed protocol requirements list, essentially tells you, we recommend putting a specification, but the protocol requirements list essentially says, what are my user needs for this project, for this implementation, what are my requirements for this particular procurement, so if you created a PRL table, completed it, it could just be a copy of that, because that PRL table indicate to the testers, these are the features, these are the user needs I'm trying to satisfy, these are the requirements that I'm trying to fulfill, for my implementation, so if you had that, it could just be that. You may also want to indicate what features are not being tested. It might be because it might be in a future version, maybe

you're phasing in your design, your requirements, your system, so you may want to indicate, yeah, these are the requirements for my first phase, and these are the requirements during the second phase, but I'm not testing this right now. So you want to indicate what you are testing and what you are not. Approach, you know, discussion of how the test is organized, and how the results are logged.

For each group of major features, for example, supporting graphics, specify how you are going to approach-- what your approach is, to verify that the graphics feed requirements are being satisfied and properly tested. How are we going to test it? So we want to talk about the major activities, the techniques, how are we going to test that the graphics, our requirements have been fulfilled? You also want to indicate pass/fail criteria, so you want a clear indicator, this is what's considered passing. You may want to say, hey, you have to pass all these steps to consider it passing the test. You may also want to say, you know, when this item is-- what constitutes a failure of the test, of a test item. Suspension criteria and resumption requirements, so let's say the system only fails a portion of our testing. You want to suspend the testing, because something wasn't quite set up right. So you want to mention, okay, what are the criteria for suspending the testing, and what are the resumption requirements? So the specified criteria used to suspend all or part, a portion of the testing activities, based on what happens, so you want to say, okay, we had a problem, what vendor to go ahead and resolve it, and this is what's necessary before we go ahead and resume testing. So you want to make that clear ahead of time, so you don't-- you're not arguing with the vendor about when to restart tests from the beginning, for example, or to restart tests from where you left off. Test deliverables, again, what you want to mention what the test plan, the log reports, the summary reports, what's being delivered during the test, and after the test.

Testing tasks, you know, identify the set of tests that are necessary to prepare for, and perform testing, and environmental needs, it's setting up the test environment. Describe the necessary desired properties of the test environment, what's required, what special equipment do we need, possibly for testing? Does it include a protocol analyzer, does it need connections, is it supposed to be performed in a lab, what software are we going to use to perform the testing, so these are examples of what a test plan for a dynamic message sign may look like. Responsibilities, schedules, so responsibilities, you might want to say, hey, the agency is responsible for providing, preparing and executing a test, a consultant is going to manage the review and witness the test, and the vendor will witness the test, and provide repairs to anomalies. So there's an example you want to make clear, of what everyone's responsibilities are during the conduct of the testing. Staffing and training needs, are there any special equipment that might-- that someone who's performing the testing may need before using that equipment, or performing the testing, so you want to make that clear, as part of your testing test plan. The schedule, what the test milestones are, when are some deliverables expected to be provided, before and after the tests? How much time is expected to perform each test, and what resources will be available? Risks and contingencies, what happens if the unit fails? Then what do we do? What do we do if there is no other testing that's available? So, for example, you might be testing a module, and you want to make sure, hey, we've got five different modules available, in case one of them fails. What are we going to do, we're facing a deadline, and a portion of testing can't be completed, so these are risks. You want to be able to mention and to write this down, so you know ahead of time, in case we run into a problem, what are we

going to do? You don't want to be arguing with the vendor in the middle of testing, when you're facing a deadline. And finally, the approvals, who's going to approve it, who has to sign off on the test plan and to sign off that the test was completed satisfactory?

The discussion, different types of test plans, there might be a test plan for each type of sign, for different hardware, there might-- for hardware testing, there might be a separate test plan just for functional or for communication, or there might be just a single test plan for the entire system. This is all more determined by how complex, how much risk there is in procuring your system. As the number of risks-- as the risk of the system increases, you may want to create separate test plans, or if there's low risk, one single test plan might be enough. What is risk? Well, what is the effect if the testing fails? So, for example, if you're procuring dynamic message sign, using NTCIP for the first time, and it's a large procurement, let's say 50 permanent dynamic message signs, that's a lot of risk. You don't have the experience, you're never tested it before, especially not with the standard, and you're purchasing 50 large multimillion dollar signs, or that's the value of the overall project. So that's a lot of risk, in which case, you may want a more comprehensive test plan, you may want to have a separate test plan for each area, one for hardware, one functional. On the other hand, if you've got plenty of experience with dynamic message signs, you've purchased these signs before, you know what you're looking for, you've got experience of it, you know how they work, you know what to look for, then that's a relatively low risk for you, for your agency, in which case, you may not need such a comprehensive test plan, and one test plan to cover all these different areas might be enough. So when you create your test plans, the first question or one of the first questions you probably should ask, what is your risk? Is it testing, is it in case the system doesn't work as expected?

Again, first time, brand new software, new hardware, first experience in dynamic message signs, that's a pretty high risk for you. But if it's stable, you've used it before, and you're using the same software you've used for years, and it works fine for you, that's a relatively low risk, so the level of detail, the number of test plans, how you perform your testing may be different. So this slide shows a well written test plan consists of a test design specification.

The test design specification is a document that shows the details of the test approach for a feature or combination of features, and identifying the associated tests. So we're going to show some examples in the next couple of modules, excuse me, next couple of slides, but an example of a test design specification might be if you have purchased-- if you're purchasing different types of signs, for example, you might have a blank out sign, you may have a permanent gantry sign, or you may have a portable sign. You may have a test design spec for each one of them, because each one of them has different features. One might be supporting multiple colors, one, because it's portable, may have additional features that a permanent sign may not have, so you would probably have a separate test design specification for each type of sign. The test design specification indicating, you know, what are we testing for each type of sign. Again, the features are going to be different, depending on the type of sign. Test case specification is a document that specifies, okay, when I'm performing a test, what are the inputs, what information am I putting in, and what am I expecting? So again, if I'm, for example, testing being able-- the ability to change the font, what are my inputs? Well, it might be the different fonts. What do I expect to happen when I try and change the fonts? And then what are conditions for testing

the ability to upload or download fonts or change the fonts? Meanwhile, the test procedure specification defines a sequence, you know, these are the actual steps that I'm performing, you know, press A to do this, press B, to do that, look at C, to make sure that the sign did what I expected it to do. So that's what the test procedure specification is. To show a more graphical view, test plan, may consist of several test design specifications, again, for example, if you have different types of signs in the same procurement, there might be a separate design of-- for each implementation, so for each type of sign, and each test design specification may consist of several test cases and test procedures. You may have a test case, for example, a separate test case to test each separate feature, one feature being able to activate a sign, another feature being able to change the graphics on the sign. So you may have separate test cases and different steps for each of those features. So we've reached our first poll, and this poll is just a quick test, and asks the question, the test case does what? So what is a test case? Does it specify the inputs, predict results and the conditions for one or more functions, specify the details of the test approach for a feature or combination of features, describes the approach, the scope, approach and resources for testing activities, or 4.

Specifies the sequence of actions for an execution of a test. So I'm going to go ahead and launch the test, and give everyone a couple-- about a half a minute to go ahead and answer the question. So the question again is, what does a test case do? Okay, closing the poll. Thank you everyone who voted, and the correct answer is actually one, and most of you did get that right. Test case specifies the inputs, you know, well what inputs are we expecting for this test, what the expected results are, I want to test the fonts, I want to be able to test the ability to change the font. My predicted result is hey, the font should have been able to change, and the conditions for the test item. Number two is actually the definition of a test design specification, it should be the ability which specifies what features I'm trying to test. Number three is actually the definition of a test plan, describing the scope, approach and resources, and number four is actually the test condition of a test procedure, which is the sequence of actions, what is it that we're doing, the exact steps that we perform to execute a test. So thank you everyone.

So the next couple of slides, we're just going to quickly go over NTCIP 1203, which is the dynamic message sign standard. The NTCIP 1203 is actually a communication interface standard, which means, this is the interface between my dynamic message sign controller and something else. It might be a maintenance laptop, it might be my TMC software, it might be my vendor's software, but the 1203 specifies what goes between my dynamic message sign and something else. It actually-- 1203 specifically, is the object definition or the vocabulary that allows us to monitor control dynamic message sign. Analogy might be the English language, for example. The English language has rules which we call grammar, which has rules for us communicating back and forth between two people, using the English language, and so-- but to use those rules, to use the grammar, we have to have vocabulary, so we have vocabulary. These are the words. So that's the analogy in this particular case, the communication really is NTCIP, that's our grammar, but the NTCIP 1203 defines the vocabulary, what words, which we call objects, but what words go back and forth, that we use that goes back and forth between my dynamic message sign and my whatever, my laptop or my TMC software. So just a quick history. NTCIP 1203, version 1, was first published in 1999, based on implementations that came out of it, we realized that there were some-- a couple of little issues with the standards, so amendment 1 was created, and was

approved in 2001. Then, based on that, we actually added a version 2, created a version 2. It adds additional functionality, for example, it now supports graphics and multiple colors, and it proves diagnostics, actually, for moderates the dynamic message signs, but more importantly, it added a systems engineering approach so prior to version 2, because we didn't have a system engineering approach, what we had was a design, some of the objects, unfortunately, were misused, or we weren't sure, or people, when they were picking up the standard, weren't sure how to use the standard. So we actually ran into interoperability issues where people used objects differently than they were intended for, so by using the system engineering process, adding that process approach, we hope that we created improved interoperability, so now people look, oh, I have this feature, I have this requirement, and the standard will now specifically point to you, okay, well we have this requirement, to fulfill this requirement, this is how the system is supposed to work, and these are the objects and this is how you're supposed to use those objects, so that was version 2.

We now have version 3, which is a recommended standard, and it's very similar to version 2, but it adds something called, Annex C, which adds the test cases and test procedures. So now we've added these test cases and test procedures, so that agencies procuring dynamic message signs can consistently test for conformance to the dynamic message sign standard. So that's the primary difference between version 2, and version 3. So when we perform interface communication testing, what exactly are we testing? Well, we're really testing two things. One is compliance with your procurement specification. Again, as we mentioned earlier, in some of the earlier modules, we hope that when agencies create their specifications, that they use the protocol requirements list, PRL table that's in there, as part of this specification. So because, based on that, the PRL essentially tells us what is it, what are my agency's user needs or my agency's requirements? So when we form the test case, we're essentially now testing, hey, this should satisfy all the requirements in my procurement spec, especially the ones dealing with the interface, the communications, the NTCIP. But we're also now testing conformance to the standard, so there's a DMS standard/ system that your procurement need to reference NTCIP standards. So the DMS system now must minimally satisfy the mandatory requirements identified as standard, that's part of it, but there are also some optional requirements which you may have, optional requirements in the standard that your procurement, your agency, may have selected for your specific project. So now we want to also test that, if you selected some optional requirements, optional in terms of the standard, now we're going to test it and make sure, hey, does it still conform to the standard? So that's what we're trying to accomplish with this interface communication, so testing compliance with your specific agency's specification, but also conformance with the NTCIP standard. Just got to note, you know, that conformance is not compliance, it is not the exact same thing. You comply with your specification, but you conform to the standard. So when we're testing for conformance with the NTCIP dynamic message sign standard, we really are testing that, A, the communication requirements are being fulfilled. Are the protocols being used, are the data exchanges, the information that gets passed and forth between dynamic message sign and TMC, are they occurring as defined by the standard?

The testing the functional requirements, so when we pass information, we expect, for example, the sign to change color, or to go blank, or to put up this message. Does it really happen, does the dynamic message sign really blank out the sign as we expected, putting up the message that we expected? We're

also testing the performance requirements. We have some performance requirements inside the DMS standard, so, for example, does it respond as quickly as we expect it to? Does it-- we might have included, oh, you shall be able to support 30 different fonts. Can it support the 30 different fonts, or the 30 different messages that we've included, that you may have in your specification. So these are the things that we're looking for when we're doing the testing. Excuse me, I missed something in the first bullet, testing the communication requirements. When we're talking about communications requirements, we are also interested in making sure the data that we get passed back and forth. So, for example, if, to fulfill a requirement, we might expect that object A and object B, gets passed back and forth between the dynamic message sign and your TMC software, so you may have a data analyzer on that communication line, and say, is that object A, and object B, actually get passed back and forth? And then when object A, passed-- object B gets passed back and forth, you know, does something happen like we expect it to? For example, if the standard said, "Hey, to go ahead and activate a message, I expect object A and object B to get passed back and forth." When I see object A and object B get passed, I expect suddenly, once it gets passed, that my message does get activated, so these are some of the things why we perform the testing, and what we're looking for when we perform this NTCIP 1203 testing. We talked about this earlier-- excuse me.

The test design specification. So the test design specification identifies features to be covered by your design and the tests. So it identifies what features am I really trying to test, and does the dynamic message sign system that I procure actually fulfill the requirement? So an example might be that the variable message sign that I'm purchasing supports 256 colors, so that's this example, so that's a feature that might have been in your procurement specifications, so now we want to test for it. Does it support the 256 colors? So in the next couple of slides, we're going to show examples of what a test design specification might look like. So features to be tested. That's one of the key things in the test design specification. Again, the completed PRL indicates what requirements, what features, you've actually selected for your implementation, so those requirements should be tested as part of your test plan. Here's an example of a PRL test that's been completed. So we have this user need, determine sign conditions, high level diagnostics, it's a mandatory user need for conformance to dynamic message sign system that conforms to the NTCIP 1203. So the first requirement that you may have is, execute lamp testing. And it's mandatory for a lamp sign or fiber sign, but let's say you were purchasing a variable message sign, so that doesn't apply to you. So that's not a requirement for your implementation. On the other hand, activate pixel testing, let's say it might be a requirement for your agency, so you would actually select yes.

And the next one is, execute climate control equipment testing. That might, again, be something that's mandatory for your agency. You may want to support that feature, so, again, you would select yes. And so this is just an example of the completed PRL. In your student supplement, under page five or seven, you'll see more detail about this requirement, the definition of this requirement for, activate pixel testing. So let's say, now we've selected, activate pixel testing, and that's a requirement for your implementation. So now what? So now we'll go through the requirement for activate pixel testing. Now we'll actually go through a table called the requirements traceability matrix. The requirement was actually pixel testing, but now the question is, what does it mean, how do I check, how do I fulfill this

particular requirement? And the RTM will actually show you the dialogs, meaning the sequence of exchanges and the objects that must go back and forth to fulfill this requirement. So essentially, the RTM and the standard, which is in Annex A, tells you how the requirement is to be fulfilled. So when we perform performance testing, we want to see that, A, you have a requirement, activate pixel testing, you want to make sure that the sequence of data exchanges or events, and the objects that are defined by standard, do indeed go back and forth, and that it results in the expectation that we expect it to do, to happen. So in this particular case, that pixel testing gets activated. So we look at the RTM, for example, you'll see that, for this requirement, activate pixel testing, it points to dialog 4.2.4.2, and that we expect an object called, pixel test activation, and you can find it in 511-243, this is how we fulfill this requirement, meaning, for me to attest that this requirement is fulfilled, the ability to activate pixel testing, we should look at dialog 4.2.4.2, and we expect the object to control test activation gets sent back and forth. So let's look at it. Let's look at the dialog. So below is the dialog, activate pixel testing. The first step of this dialog is the management station, which might be your laptop or your TMC software, shall set the object, pixel test activation, we shall set the value to test. So that's the first step.

The second step is, we shall repeatedly get pixel test activation from the dynamic message sign until the dynamic message sign returns the value of no test, or it times out. And post condition, after that, the following objects may have been updated, so please get any of these objects as appropriate. So what this dialog says, essentially, is, hey, we're going to go ahead, I'm a TMC software, I'm going to set the object, pixel test activation, I'm going to set that at value to test, and I'm going to send that value to the dynamic message sign. In turn, the dynamic message sign, when it sees that this object, pixel test activation, has been set to test, it's supposed to go ahead and start testing the pixels. So that's how we activate pixel testing. And the management station is just going to keep watching it, seeing that value, until the dynamic message sign sends it back to no test, to the value of no test, or until it times out. And if it times out, you shall do something, if I get back no test, you shall do something else.

So that's the dialog, this is how we perform pixel testing, activate pixel testing according to the standard. So that makes sense, that when we do the testing, we want to with our test case, how we perform testing, should be able to follow these same activities, follow this dialog, so now we're going to go ahead and look at version 3, of the standard. So that was the dialog, so now we're going to look at version 3 of the standard, and look at a table called, Requirements to Test Case Traceability Matrix, This is only in version 3, it's in Annex C. Version 2 did not have these test plans, but essentially they are requirements to test case traceability matrix says, based on what requirements I have, what requirements I'm trying to fulfill, what test cases must be passed, to fulfill-- to make sure that the requirement has been properly fulfilled by the standard.

So I'm going to bring up the requirements test case traceability matrix, and I'm going to look under the same requirement, activate pixel testing. And here we have it, activate pixel testing. It points to two different test cases. First test case is C.3.5.1, Pixel Test, No Errors, C.3.5.2, Pixel Test Errors, is the second one. And essentially, what we're saying, this matrix is that, hey, to confirm that the implementation has satisfied-- has fulfilled this requirement, you have to run both test cases, and you must pass both test cases. Why are there two test cases? Well, sometimes there may be different options, but sometimes also we want to make sure that there are no problems in case something else happens, which we call

negative testing. And this is probably not as critical in dynamic message signs, but maybe for other devices, let's say, traffic controllers, you want to make sure that both cases are covered, in case there is a problem. Like, if there's a safety issue involved, sometimes we do perform negative testing, which means, hey, if I expect a value to be for tracks control one or two, that's fine, we want to test that. Hey, does one come up? Does two come up? Fine, but we also, in case something gets messed up, we want to make sure that there are no safety implications, so I might, even though the value might only support one and two, I might put in a different value, let's say, three, to make sure, hey, I'm going to send a value three. I want to make sure that my controller doesn't react negatively. Like, let's say I send a value three, that suddenly, oh, the green bulb here is in multiple locations, on all approaches, for example, so that's why sometimes we perform negative testing. It's to show all the possible values, but also we want to make sure in that case I send a bad value, something negative doesn't happen. But in this case, we want to test two different test cases, one to say, hey, when the sign reports, I did my testing, and no errors happened. And then we also want to do a test case where you say, hey, I found a couple of pixels that are wrong, now what do I do? We want to make sure that the sign behaves the way we expect it, or doesn't do something that might be bad. So sometimes multiple test cases are needed to completely test a requirement. Sometimes they might have different values, for example, we choose page justification, I might want to test that the sign can properly page justify, so it may be three potential different values. I may want to top justify, center justify or bottom justify, so I want to test all three different conditions and make sure that I can do top, bottom and center justify.

Sometimes when we have different test cases to test different conditions, in this case, what we want to make sure the sign properly reports if there are no errors, and again, that it properly reports in case there is an error, a pixel error. So we want to test-- that's why we have two test cases in this particular example for activating pixel testing. So that's what the RTCTM does, the requirements test case traceability matrix does. Based on the requirements you selected from the standard, these are the test cases that we want the implementation to pass, to properly confirm that we conform to the standard. But one of the steps that the agencies may want to do is tailor their own requirements test case traceability matrix. The one that's in the standard, covers all the different requirements that are supported by the standard.

For your agency-specific implementation, you may want to create your own RTCTM table. In your tailored one, the traceability matrix would only have the test case, the requirements, that you expect to be fulfilled for your agency, so for example, if your implementation doesn't support graphics, don't include it in your tailored table. Only include those requirements that apply to your specific implementation, so this tailored RTCTM becomes part of your test design specification, and essentially identifies, hey, these are all my requirements for my specific implementation, and these are all the test cases and test procedures that I expect to pass, as part of my test plan. Again, if you have different implementations as parts specs if you have a blank out sign, and colored DMS signs as part of your same specification, you're going to have two different tailored RTCTMs, one for the blank out sign, one for the colored DMS sign. Sometimes also, one of the things we had discussed in one earlier modules, sometimes, as complete as we try to cover all the possible requirements in the standard, there unfortunately will be a couple of requirements that the standard does not support. So for those

requirements, you'll have to create your own test cases and test procedures for fulfilling those requirements. So when you create your tailored RTCTM, make sure that you include your additional requirements and test cases in your tailored RTCTM table, and your test design specification. So now we've reached another case study, so I'm going to show an example of a test design specification, for VMS sign.

Let's say, for procurement, for your implementation, for your procurement specification, you are purchasing both a blank out sign, a couple of blank out signs, and a couple of variable message signs. So in this particular case, you would probably have a different test design specification, one for your blank out sign, one for your VMS sign. For your features to be tested, you just want a copy of your completed PRL. You should probably have two PRL tables, for your specification, one for the blank out sign, one for the VMS sign, so just take the completed PRL and put in with the appropriate test design specification and that will tell you, that will tell you what features are expected to be tested for each test design spec, for each type of sign. You may want to refine the approach, you may want to, as far as test identification, you may want to create and include your tailored requirements test case traceability matrix, you may want to include what the pass/fail criteria are for your implementation. So that's the test design specification.

Now we're going to go a little bit into detail, and see a test case specification. The test case specification, again, is a document that specifies the inputs, the expected results, and the conditions, the environment for performing each test case. NTCIP 1203, version 3, does include the test cases, the test case specification, but in addition to that, we include the test procedures. If you recall, the test procedures are the step by step execution of procedures for performing the test. To make it easier, to make it less confusing for everyone, we actually include the test case specification, test procedures into a single test case, and we call it a test case. So, for example, for activate pixel testing, our example from the earlier, we have a test case called C.3.5.1, that identifies the inputs, the results, the expected-- the predictable results on the execution and conditions, but we also include the test steps, the test procedures, as part of that test case. Let's see. For activate pixel testing, only a single instance may be required to verify conformance to the standard, meaning we expect that, for a VMS sign, we may only have to run, activate pixel testing once, but sometimes for your specification, there is sometimes, there are occasions where you may have to run the test case multiple times. So for example, you may say, hey, I want to be able to store five messages for example, that are permanent.

So you may want to run that test case five times, one for each message, to make sure that, oh, that message is in there as I expected, I'm able to activate it. So that might be an instance where you may want to run a test case multiple times, where you have, let's say, three fonts and you want to verify that you can control and change all three fonts that are included in your sign. So you might run those test cases three times, one for each font, but pixel testing is one of those examples where you probably only have to run it once. So that might be part of your overall test design specification. Here's an example. So this is what I was saying, if your specification requires that DMS comes with three fonts, you may want to run this particular requirement, run this particular test case three times, once for each font. Test procedures, this is the sequence of events, a sequence of steps for executing a test. So the standards for test procedures are related in a specific manner in the standard, so that we try to stay neutral, meaning,

we don't tell you the test case and test procedures don't exactly say, oh, we expect-- let's say you want to put up a message, we don't say, oh, use this message. Like, oh we kept it open deliberately, wrote it deliberately open, so you can come up with whatever message that you want, so it's kind of generic in a way, and for example, if you wanted to test multiple values of a number, let's say there's a feature where I can show what temperature it is right now, and the test procedures are written in a way, so that I don't care where you come up with the value, you could include it in your test plan, test case, that says, hey, use this value, 67 degrees, or maybe it's a random number generator, or you may pick it out in there. So when we wrote these standard test procedures, we kept it very generic, so we don't specify, oh you have to- or- how you create the values or what values you use. It's up to you as an agency, to select the values, how you want to select the values. Again, you might or may want to simply define it ahead of time, or you may select the values while you're actually performing the testing. So per the IEEE 829 definition, the test procedure only defines the steps necessary to perform the function. So let's just talk a little bit about the heading.

This part is the test case, so we have a heading, this is the test case for pixel test, no errors. We have to find out a description, which kind of talks about what this test case is trying to accomplish, what requirements are being fulfilled, inputs specifications, where are the test values coming from? Are they outputs from a different test, are they randomly generated, might be predetermined test value, so that's what the variables are. It's up to you. Sometimes it's up to you, sometimes we'll say, okay, these values, these variables, might be manufacturer specific, so in this particular case, the pixel test time might come from the manufacturer, because the manufacturer it's the manufacturer's sign. They might say, it takes me three minutes to complete an active pixel test, so you may want to put the value at three minutes, for example. Then, below that, we have the pass/fail criteria, so the DUT, is Device Under Test, shall pass every verification test, included within the test case, to pass the test case, so we clearly define, for the test case to pass-- to pass the test case, you have to pass every step. And below it, we have, actually what are the test procedures, and you notice that every test procedure, every step is uniquely numbered, and under the test procedure, we define the steps that are going to be performed.

So the first step, you know, it's configuring, we're setting up the test case. Second one, same thing, we're just recording the values, how long we expect the test case to be, the test time to be performed. Three is setup, we want to make sure that all the pixels are working prior to this test, and four is actually, the first step. If you recall, back in the dialog, from one of the earlier screens-- earlier slides, the first step of the test case, of the dialog, was to set pixel test activation to the value of test. And so that's what this step says, set it now to the value of test. And if you look a little further to the right, we have some references, under results, additional references. The result is to continue to pass this step. It's under results, if the area where we indicate whether we're able to properly pass this step. In this case, were we able to properly set the value of pixel test activation to test, yes or no. If the answer is yes, we passed it, if not, we failed it. And below it we have the reference, section 3.5.3.1.1.2, which is the requirement that we fulfill, when we pass this test, and then we also have to verify the dialog, section 4.2.4.2, step A, that was the step that we're trying to pass. That was the step that we're trying to pass, that we're looking at with the dialog. So on this page, we show everything that we need to know, to begin the tests. We have the test steps, we have the variable values, what the variables are, we learn

how to set it up, we indicate, you know, what we're trying to test, what requirement we're trying to fulfill, and we provided references, in case we need to look for further information about what we're trying to test. At the bottom of the test procedures, that's the output specifications. So we indicate, along the test cases and test procedure, what are the expected values. So for example, in step 17, verify that the response value for a different pixel has bit 5 cleared? So that's actually one of the steps that we have to check. So this is the expected results, this is what we expect to happen, so that's parts of the test procedures.

At the bottom, we also have, where we complete it, we complete the test case information, who tested it, what day was tested, any notes that we may want to include, and whether we passed the test case or not. What else belongs in the test procedure? Environmental needs, test set-up, so one of the things we want to do, we include in the procedures to make sure that we've properly set up the test properly, and that's usually indicated by the word, for example, Set-Up. You'll notice that we have keywords to most, if not all the test procedures, and these will be in all caps. So environmental needs, that's how we set it up, so those are usually indicated by the words, SET-UP in the test procedure. Sometimes we have special requirements for setting up, special constraints for setting up the test case, and those are usually indicated by the word, NOTE, in caps, or CONFIGURE in caps. On occasion, test procedures are dependent on other cases, test cases, meaning, we need something to happen before we perform this test case, we need the test unit to be in a known condition. So we know, you know, what values are there before we start the test. Otherwise, if we have the wrong values, or wrong information, or it's not set up properly, it's going to fail, so sometimes it might be dependent on other test cases. One of the things that we've done specifically, in the dynamic message signs test case, is, at the end of a lot of test cases, we perform a procedure to blank the sign. So that's quite often the last step of a lot of our test cases, to blank the sign, so we PERFORM those-- that's in caps, the test case label, blank the sign. That way, whenever we end a test, any of the test cases, the sign is in a known condition. That makes it easiest for us to go ahead and continue performing other test cases.

Procedure steps, each step usually has a unique number. So it's unique within the scope of that specific test case, and the numbering usually shows the normal sequential order of execution, so you can tell what order we expect to perform these tests. This is an example that you see, it's an example of how, sometimes, we do loops, so may perform multiple steps, multiple times, maybe one for each character, one for each font, one for each graphic, so sometimes we perform multiple steps, the same steps multiple times, so we'll go through a loop so you may see a GOTO keyword, for example. So that was a lot that we covered, a lot, so far. We've essentially gone through the test cases, and the test procedures that are indicated, that would be provided in the standard. So just want to just do a little quick test. Below is a dialog, in NTCIP 1203, and just wanted to focus in on the steps you see in this dialog. The management station shall GET, the DMS message beacon, X.Y. So which test that properly reflects step C of the previous dialog. So this actually would be a test step in one of our test cases. What's the-- we're asking the question, what's the equivalent of that, when we create the test case? So is it with step C, get the following object, was it to get dmsmessagemultiStringx.y, dmsmessagestatusx.y, dmsmessagebeaconx.y, or dmsmessagepixelServicex.y? And, if you need to find it again, it should be in

your supplement also. So the dialog was, the management station shall get dmsmessagebeaconx.y. So let me go back.

So the answer was just-- So we just want-- this is this example to show how we would go ahead and based on each step of the dialog, we would create and upgrading a test step in the test cases, in the test procedure for that, so step C was, the management station shall get DMS message beacon X,Y. So the correct answer actually was three. This is how the test step would look like in the test case. We would have a test step that says get the following objects, DMSmessagebeaconx.y. This review of the learning objectives for this module.

The first learning objective was just to recognize the purpose and structure and content of a well written test plan. So test planning then identifies the scope of the test, what is it just what we're trying to test, what items are we trying to test. Is it just the sign, or are we including the software as part of this testing. What the test environment is, what software we're trying to use. Are we going to test in the lab, or maybe we're going to test it in the field, for example. The roles and responsibilities of all the individuals involved in the testing, who is supposed to provide what information, who is responsible for setting it up or creating the test plan, and who's responsible for the deliverables, and finally, what do you expect to be delivered upon the completion, or during, or before the creation of the test plan to your own testing activities.

The second learning objective was to describe, within the context of a testing lifecycle, the role of a test plan, the test plan to be undertaken, so testing usually occurs after the requirements development, and after the system has been implemented, and the reason why we perform testing, is to verify that the system fulfills all the requirements, so it meets your project specifications, satisfies your project specification, and validates that the system satisfies all the stated user needs. So the first part, again, did you build the right system-- I'm sorry, was the system built right, and the second part, again, did you build the right system, did it solve your problem, the system that you've implemented. The third learning objective, describe the application of a good test plan, to DMS being procured, using a sample DMS test plan. So throughout the module, we showed examples using dynamic message sign examples of what a test plan for a dynamic message sign might look like. What type design specification for dynamic message sign might it contain, and we showed some examples of test cases, using dynamic message sign examples, mostly test case specifications and test procedures, again, are we in 1203 version 3, but you may want to tailor it for your specific agency, if you know how, for example, what values you want to use, update it. Update the test case, test procedures for your examples, for your values that you plan to use. Identify key elements of the NTCIP 1203 standard, relevant to the test plan. So there are tables again, inside NTCIP 1203, that help you, that provide you with information you need to perform testing. The PRL identifies the requirements, for satisfying the user need, but this is, importantly, if you complete the PRL, it identifies what your agency's user needs are, what your requirements are for the project implementation, so that's why we encourage that everyone creates and completes a PRL table. There's also requirements traceability matrix, where, based on the requirements that you selected, you can see what the standard-- what the design-- defines as a design. Excuse me, what the standard defines as a design to fulfill that requirement. What information do we expect to get past and forth, what's the sequence of that information, what objects do we want to go

back and forth, what's supposed to be happening at the sign, or at the demand specifications? That's all part of the standards. By defining that design, that's how we accomplish interoperability, so that all the different vendors, whether it's software to hardware, whether it's at the TMC, or at the sign itself in the field, so that we know what the behavior is going to be, so that we get the expected results when something happens. Whenever we get these data objects, whenever the sequence of dialog occurs, so we know what's going to happen. That's how we establish interoperability.

And finally, there is a requirement to test case traceability matrix, based on the requirements that are supported by Dynamic Message Sign. The standard indicates these are the test cases that we expect to happen, to occur, to be performed so that we can verify conformance to the standards. And finally, the last learning objective was to walk through the process of adapting a test plan in the context of the needs and requirements of dynamic message signs that have been selected by the user.

So again, taking a completed PRL table, which has the end, has your user needs, and your requirement, and for your implementation, and based on those requirements that you selected, how do we find, how do we use the RTCTM table to determine what test cases do you want to perform for your implementation. We hope that you would tailor these test cases, these test procedures for your own implementation, so you only select and test, use the test cases associated with your project requirements that are in the PRL table. Here are some resources that are available in case you want to find out more information about the NTCIP standard, but also a little bit more about testing. All these are NTCIP documents, and you can find these documents on www.ntcip.org. When you get to that site, on the top you'll find a link to the library. If you click on library, you'll find the option for document links. So just select the appropriate document, whether it's NTCIP 1203, NTCIP 9001, 8007 or 9012. 1203, is the dynamic message signs standard. The published version right now is version 2. You can get it downloaded free as a PDF file from that site. And then there's also version 3, which is not a published standard yet, but it is a recommended standard, so you can use that. It's probably-- and that contains the test cases, the test procedures that I've used examples of in this module. So I probably would recommend version-- downloading version 3. NTCIP 9001 is the NTCIP Guide, and that provides a larger overview about the NTCIP standards in general. NTCIP 8007, is a document on how to create test cases and test procedures for the NTCIP standards, and NTCIP 9012 gives you a little bit more information about where NTCIP testing belongs to overall context of center to field communications, in the overall context of ITS standards.

Where do NTCIP testing, communication testing belong in the overall context of testing. So that's it for this module. I'm going to open it up for questions right now, in case anyone may still have questions about this module. I thank everyone for joining us today. And hopefully this has been helpful. Again, this module is really just an overview of how to use the NTCIP standards to create your test plans to perform testing. So again, so we do have version 3, and I suspect we'll continue creating testing cases for some of the other NTCIP standards. The test cases, if you look at the documents, that would be 300 pages for the standard, almost another 300 pages for the testing, for the test cases, especially just for the standard. The tests, the Annex C, the testing stuff, material in the standard is pretty detailed, We try to keep it, again, we try to keep it generic as possible, so that you're not stuck using a particular way of testing. We try to keep it as generic and flexible as possible, but we also want it to be as complete as possible, so

that you can properly test and make sure, if you pass all your testing, that you can be pretty sure that what you purchased, what you procured, whether software or hardware, is complete and conforms to the standard.

I have a question. How much time and budget should a contractor identify for testing DMS in a project, any guidelines? That's an excellent question. I'm not going to put a numeric value on it, but, well, I guess I am. I take it back. I've heard values of between, like, maybe five to ten percent. Again, I think it's more dependent on how much risk the system is to your particular agency. Again, if it's a high risk, meaning you don't have much experience of it, and some multimillion dollar procurement for you, you've got little experience with standards, you've got little experience with testing, but it's a high priority, a high profile project for you in this big project, let's say, 50 signs, then yeah, you may probably venture closer to the ten percent. But realistically, a lot of, at least a lot of the larger agencies, let's say state agencies, this is not new to them. They've procured dynamic message signs in the past before, they've got their TMC software that they've used, and it's pretty stable. They're comfortable with testing, or they're comfortable using dynamic message signs, in which case, their testing may not be as attentive, as intensive or extensive. The vendors are pretty, you know, the vendors have done this for a while now, so a lot of larger vendors and older vendors are familiar with this, so there's not much of a risk for them. Although they say they've seen everything, but they pretty much know what to expect. So, again, it's more of the level of risk for your specific implementation. Five percent might be realistic if it's a low risk for your agency. Ten percent might be more realistic if it's a high risk for your agency.

Okay, I don't think there are any more questions. So I want to thank everyone for participating and joining us today, and this will then close-- end this particular module. Thank you again, everyone.