T309: Applying Your Test Plan to Ramp Meter Control Units Based on NTCIP 1207 Standard Version 2

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 am 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 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 Standard 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.

Narrator: Throughout this presentation, this activity slide will appear, indicating there is a multiple choice pop quiz following this slide. You will use your computer mouse to select your answer. There is only one correct answer. Selecting the Submit button will record your answer and the Clear button will remove your answer if you wish to select another answer. You will receive instant feedback to your answer choice. This is T309, Applying Your Test Plan to Ramp Meter Control Units Based on NTCIP 1207 Standard Version 2. Your instructor, Mr. Miller, has more than 36 years of experience in the management, design and development of critical control systems, including traffic control, bus rapid transit, and connected vehicle. Specific ITS experience includes the design and first deployments of ATC 5201 and ATC 5202, model 2070 ASCs, as well as the design and first deployments of NEMA TS2 ASCs with NTCIP.

Dave Miller: Target audience for this module is—it's primarily for traffic management and engineering staff. Other stakeholders that could benefit from this module include traffic management center operations staff, freeway and traffic signal maintenance staff, systems developers, as well as private and public sector users, including manufacturers. The prerequisites for this course are shown in this slide, and we began in this series with the basic knowledge of ITS standards and acquiring standards-based systems. Next we continued through understanding of user needs and the resulting requirements development. We got to C101. C101 provided an introduction to communications protocols, and then finally an understanding of user needs for ramp meter control units that are based on NTCIP standard version 2, and that's NTCIP 1207 corresponds to ramp metering. This is a curriculum path shown here, and again we started with I101 up in the left and we went through all of these modules, so you're probably familiar with all of these if you're taking this course. At the bottom row, those bottom three, A309 is where we got into understanding user needs for ramp meters, and that was a two-part course; and then again, 309b, the second part, was more details on understanding the requirements. So that led up to this course, where we are applying our test plan to the ramp meter control, and that goes back to A309a and b.

This particular module has five learning objectives. We'll begin by describing the role of the test plan within the context of the system lifecycle and workflow of the entire project. Next we will cover the purpose, structure, and content of test documentation based on the IEEE 829 standard. At that point, we will describe the test documents needed for NTCIP 1207, and those documents are the test plan, test design specifications, test case, test procedures, and test reports. Next we will apply good test documentation to a ramp meter unit to develop a sample requirements to test case traceability matrix. And then finally, we'll wrap up this course, we'll write an example test plan to verify that we're meeting our NTCIP 1207 requirements for a ramp meter.

So beginning with our first learning objective, Learning Objective Number 1 has four key points. We'll begin with the purpose for testing ramp meter control units, and then that will be followed by a brief review of systems lifecycle from our prior modules, and the testing to be undertaken in this module, and then finally in this learning objective we will review the ramp meter control verification methods and the RMC testing process in relation to the overall systems lifecycle.

We're going to start here by explaining the reasons for testing an RMC unit, since a number of ramp meter products have been available on the market for some time. Many states have different requirements. Since they've been on the market for quite a while, why would we at this point want to test them? Many ramp meter units were developed to the needs of requirements for a particular agency. So in Texas, we have—there's a way of doing a ramp meter; California is a little bit different. You'll find differences in the way things are done depending on the local needs, but without testing we would not know whether the RMC will work as expected by the stakeholders of the project who are paying for it. Does RMC satisfy the system requirements? Does the RMC design solve the right problem? Does the RMC design satisfy the needs and the intended use and the expectations of the stakeholders who are going to be using it at the end? So here we know that testing provides objective evidence to confirm the answers to these questions. For example, each requirement will have objective evidence that the requirement has been met by this design, such as pass/fail limits and clear test procedures, and this evidence is delivered at the end according to the IEEE 829 2008 Standard, and that standard is familiar to all stakeholders and has a common understanding of terms. We'll get into that a little bit later in later slides here.

So this is a familiar “Vee” model. The RMC system lifecycle follows the same “Vee” model. It takes the form of a “Vee” model, where the concept of operations on the left, the system requirements on the left, the high-level design, and the detailed design are decomposed on that left side of the “Vee” model, if you can see here going down. Actual hardware and software development and implementation are accomplished at the lower portion. So across the bottom, this is where we're actually doing the hardware and software design, and while testing of each unit is followed by integration testing of the entire system is shown on the right. So where we have in red here, this is the focus of this module. So this is the testing to be undertaken. And as you see for the “Vee” model, the previous slide, the testing to be undertaken consists of four level test steps of the IEEE standard. So that's over here on this side. So these are levels. So the levels include unit device testing of each hardware and software module, subsystem level testing of software that has been integrated into the hardware and the interfaces, the other subsystems. The third level shown there is system verification level testing of all subsystems connected together with their interfaces. So that would be all of the major subassemblies connected together with their cabling and interfaces. And then finally, at the top here, we have system validation level of the complete system. So that would be the same as the system verification, except this is where it's programmed in the final configuration, so we'd actually have real data and we'd be controlling real vehicles at the ramp meter.

So here we can begin to see the value of the IEEE 829 Standard. For example, each of these tests may have had several other names. In projects I've worked on in the past, they're called different things. We hear like module test, software test, hardware test, and those can mean different things to different people. So what we're doing here is using

IEEE 829, and that's really a standard—we're not going to teach that standard—but that standard has definitions of what the levels and system test levels are. It has a common understanding of terms. So beginning at the bottom right of the “Vee” and continuing upwards to each required level test in the appropriate order for the test specification, and that will eliminate confusions and questions among all the stakeholders. So we'll have more on this in later slides.

In the prior slide, we saw how the user needs are decomposed and defined on the left side of the “Vee”, while the actual design is recomposed and integrated on the right side of the “Vee”. So as you can see here, we have the levels going up on the right side, and then to the arrows to the left here is what we call traceability. Traceability is completed each step to identify problems as early as possible in the lifecycle. For example, we want to test the software in the interfaces of each module against the detailed design document for that module. All open issues for each module must be resolved before servicing later in system testing where fixing them will be much more expensive because it'll be interacting with each other in different modules.

Next, subsystems and modules are tested against the high-level design with each open issue resolved. After that, subsystems are integrated into a system and tested against the system requirements. And then finally at the end, the system is configured for operation and tested against the concept of operations document. So traceability of each of these steps is from right to left, and it's made up of the test plans. So these test plans are the arrows that go from right to left to verify that what was done early in the project has been realized at the end of the project.

Verification methods use a testing process to determine whether the system conforms to the requirements and intended use. In the prior slides, we saw that the testing process begins at the lowest level to ensure that issues are identified as early in the system lifecycle as possible. The validation may be determined by inspection, demonstration, analysis, or testing. And again, recall in 829, 829 provides a common understanding of terms. According to IEEE 829, the testing process provides an objective assessment of the system products throughout each project's lifecycle, and again, this testing process is focused on identifying and resolving issues early in the lifecycle. The cost to correct issues later in the lifecycle would be stepping back and retesting each step at greater expense. Beyond design and system testing, additional test steps are conducted during operation and maintenance, at system upgrades, and when the system is replaced.

T309: Applying Your Test Plan to Ramp Meter Control Units Based on NTCIP 1207 Standard Version 2

Continuing on with this idea, the testing process includes three major stages. So when we're doing a testing, we do test planning, followed by test documentation and preparation, finally test execution, reporting of the results. Each of the steps results in an output that conforms to the IEEE 829 standard framework. For example, step one results in a test plan. Step two results in a test design, test cases and test procedures, while step three results in a test report.

So putting it all together here, the test planning of step one of the prior slide is completed early in the system lifecycle, as shown. The test planning step should be developed from the concept of operations and requirements documents. So even though it's shown on the right, it's actually done pretty early in the system lifecycle. So we want to do that before we even know the design content. We can write a test plan based on what the requirements are before we even know exactly the detailed design. That way testing can be planned in the design phase very early instead of as design changes later, at greater expense. So examples of this might be system self-tests. We might write a system self-test program. So we might want to document what that would be. We would, at that point, very early, before the detailed design starts, we want to include test points, and there's a lot of other things you can do if the testing is planned in the beginning that can be put into the design as we go through it at almost no expense.

Preparation of test documentation, step two, the prior slide, is completed during high-level design and detailed design. And again, the actual test design can be included in the design instead of later at greater expense.

And finally, the test execution and reporting of step three that we had in the prior slide is completed at each level test on the right side of the “Vee”. So each one of these levels at the right side of the “Vee” is traceable back to the left side. So at this point, the test documents only need to be populated with test data and tests are executed at each level. So at this point we can have the test documents and we can have in the test documents the expected results, because we know it very early, and then later on towards the end, we've used that and we populate the test data so we can compare it. In reality, the test documentation is refined as part of executing the test, especially for the detailed test procedures of the finalized system.

So the activity here is we have a question we're going to ask, and we have a choice of answers. So one of the answers will be correct. So which of A, B, C, or D below is not a reason to test an RMC unit? Again, there's one reason that is not a reason. So let's go ahead and open the poll and just select one of the four choices there. Let's move on to the answer. So let's review the answers here. So the correct is “B”, testing is part of the NTCIP 1207 v2 standard. That's the correct answer. Testing isn't really part of the NTCIP standard. The NTCIP standard shows the conformance groups and the objects, that sort of thing, but each design really has to have its own test, because as we said in the beginning, different states and different jurisdictions and agencies have different ramp meters, so you're going to have different designs and the standard is not really going to have a full testing procedure. “A” ramp meter testing, it's tested at the system verification level. And then “C,” solve the right problem—that is correct, because testing does confirm that the right problem is solved. And again, satisfy user needs—that would be an incorrect answer here, because we saw in the “Vee” model that the systems lifecycle testing traces back. And again, what we're asking for here was the incorrect answer, so that would be “B.”

Okay, we're going to do another activity. So the question here is: “Which is not a testing process within the lifecycle?” And here again, this is a "not" question. So of the four choices—a) test planning, b) preparation of test documentation, c) test execution and reporting, and d) identification of system requirements—so which is the one that's not a process within the lifecycle. The correct answer here is d) identification of system requirements. That's not part of the testing process, and we learned that in the earlier modules. The test planning is based on system requirements, so that's not part—here. So that would make “A” an incorrect answer. Test planning is done during concept of operations and system requirements. “B,” test documents are created during high-level design and detailed testing. And again, c) execution and reporting are done at each level. So the incorrect answer here is “D.”

That kind of wraps up Learning Objective Number 1, and the summary here is—what we did is we reviewed that ramp meter control units are tested to ensure that the installed unit meets expected user needs, and again, we mentioned there's a number of different agencies and jurisdictions so they don't all work the same so there's different designs. Testing fits within the system's lifecycle on the right side of the “Vee” model, which is traceable back to the user needs. We learned that a review of the verification methods includes inspection, demonstration, analysis, and testing. And then finally, the testing process is conducted within the lifecycle in three stages, which are planning, document preparation, and then all of that is followed by test execution that results in final test reports.

T309: Applying Your Test Plan to Ramp Meter Control Units Based on NTCIP 1207 Standard Version 2

We're going to move along now to our second learning objective, and it has four key points, and we will begin by describing the purpose of test plans, and then followed by answering the question, "What is a test plan?” Next we will take a deeper look at the structure of a test plan, and then finally in this learning objective we will list the contents of a typical test plan so you'll have an example to look at.

A test plan is used to plan and manage the execution of the test using a standard format and with a common understanding of the terms among all the stakeholders that are involved in the project. In addition, the test plan identifies test activities and methods that will be needed, and again, it's presented in standard format. The test plan sets objectives for each of the identified test activities and identifies the testing risks, the resources that are needed to do the test, like equipment and manpower, and the schedule. And finally, the test plan determines the requirements for the test documentation. What is a test plan? A test plan is a document that clearly describes the scope, what will be tested, and what will not be tested, and it's written in a standard format using a standard definition of terms that everyone is familiar with. The test plan also includes a description of the test approach, a high-level overview of how the test will be conducted, and in addition, the test plan lists the resources needed to conduct the test, including the required people, the skill sets of the people required, material, test equipment, samples of the units that are going to be under test, and other things. Finally, the test plan includes a high-level schedule to complete the described scope, and the high-level schedule includes a description of the approach and describes the resources needed. The test plan identifies the items to be tested, the features to be tested, a list of testing tasks that will be conducted, and also identifies the risk and the required contingency plans. So usually when you start out, you know that there are some risks and it's good in the very beginning to identify the risks. Once the project starts, we all know things can change and go wrong. Identify those up front and have a contingency plan. "If this happens, this is what we're going to do about it.” Identify all those in the beginning as much as possible. IEEE 829 identifies two types of test plans. First, a level test plan documents the test that will be conducted for one part of the complete system, while the master test plan describes how each of the level test plans fit together within the overall testing activities. Test plans are not defined in the NTCIP standards. Level tests and master tests are described in the IEEE 829 document. For example, level tests and master tests might also be known as module tests, high-level tests, and other names from other organizations, but if we use the IEEE 829 we ensure that all stakeholders use the same terminology and definitions so anyone that's in the project can read it and they know what it means by level test—this level, this level, this level.

T309: Applying Your Test Plan to Ramp Meter Control Units Based on NTCIP 1207 Standard Version 2

IEEE 829 standard structure includes a master test plan that covers the overall test processes, activities, and tasks, as well as level test plans that include unit tests, subsystem tests, and system acceptance test plans. This graphic shows the structure of a test plan. At the top, the master test plan describes the overall test processes, the activities and tasks, and identifies all the levels that are going to be tested as part of this test process. Each of the level test plans describes the scope, resources, and test methods conducted at each level. For example, each unit test plan describes the testing of each electronic design, each software module, and like mechanical packaging items. Once each unit is successfully tested, the next level test integrates the subsystems. So that would be like integrating a software unit into a hardware unit to create an electronic subassembly that executes the software. The next level test configures the subsystems into the final system acceptance test. Testing at each level intends to identify and resolve issues at the earliest stage in the project to avoid costly fixes later in the project. For example, at the system acceptance level, issues should be uncovered in the interfaces between the subsystems, but at that point we should not be finding new issues within each subsystem.

A master test plan may not always be required. For example, system tests of dialogs might be simple enough for a ramp meter control, so it may just have a level test only. So if we have a complex project, we will have a master test plan that shows how all the unit tests are conducted. If we just have a simple system, a master test plan may not be required. So basically master test plans are used when multiple level tests are used in the final acceptance.

This graphic depicts the relationship of levels tests and the workflow for testing a system and includes ASCs and RMCs. So we may have a system usually that might have actuated signal controllers connected to a system, a central system, so we probably have some ramp meter controls connected to the central system. So the overall master test plan would be first created on the left includes—it'll just state, "Here's the different levels. We have an actuated signal controller unit. We have a ramp meter controller unit. Each has software." So that might become four level tests. So the workflow, our testing workflow, would start with the master test plan, followed by execution of unit tests, followed by execution of subsystem integration tests, and then lastly the system integration test and acceptance. So you can see from this slide here that the lifecycle from the “Vee” goes up like this, but the workflow of our testing sequence goes like this. We start with our master test plan, then we test the units, then we test the subsystems, then we finally do an acceptance test.

T309: Applying Your Test Plan to Ramp Meter Control Units Based on NTCIP 1207 Standard Version 2

This slide shows a list of requirements applicable to a ramp meter project, and in this case the PRL for the NTCIP 1207 B02—this is the one that we created from training module A309b. If you took that one, this should look familiar. So this protocol requirements list identifies the requirements to be tested, so this should be familiar for those that took the prior course. So these are the different pieces of the protocol from the ramp meter that we're going to be using in our design.

The contents of the test plan—this describes a master test plan outline per IEEE 829. So this includes the master test plan—if you read IEEE 829, it's going to show a format for introduction, details, and general description. It'll have a glossary of terms and acronyms and a history of the document, when it's been changed and who changed it, and what changes were made. In this outline, the requirements for NTCIP 1207 test documents as part of the master test plan detail. So if we were doing a ramp meter control and it had a master test plan for the whole system, this is what the outline would look like.

This describes, in this slide, a level test plan outline per IEEE 829, and that includes an introduction, details and test management. The level test plan also includes general information such as metrics, how are we going to measure things, what's the data we're going to use, test coverage, a glossary of terms, document change procedures—like who has authorized the change and what's the sign-off procedure—and a document history of who changed it, and when and why. So NTCIP 1207 Test Traceability Matrix is part of the level test plan, and again, in the prior modules we developed a traceability matrix.

And again, the content of the test plan—this describes—it's just broken out here as bullets. So its plan test activities, the progression, the environment, infrastructure, responsibilities, roles and responsibilities—who does what, who's authorized to do what—the interfaces among the stakeholders, how do we communicate with each other—is there a single point of contact through a project manager, et cetera—resources and training, schedule of investment costs, and then we talked about risks and contingencies. So in the level test, each one of our levels—like if we're testing a software module—you would describe who's doing it in this format. If we're taking a piece of hardware, same thing. And again, this is continuing on, the level test plan—and again, if you look in 829, there'll just be a section for general, just general information. That might include quality assurance procedures, metrics for measures, how are we going to measure things, glossary of terms, and change procedures and history again.

T309: Applying Your Test Plan to Ramp Meter Control Units Based on NTCIP 1207 Standard Version 2

At this point, we're going to go ahead and do another activity. So the question in this activity is: “Which is a reason not to use IEEE 829 Standard?” And again in these activities, there's one correct answer, but again this is a "not.” So of A, B, C, and D, which is not the correct answer? So is the answer 829 is part of NTCIP 1207 B02? Would the reason not to use it is to provide familiar documents? C) standard definition of terms, and d) reuse in later projects. So let's go ahead and make a selection. After closing the poll, the review of answers here is—the correct answer is “A.” NTCIP 1207, if you read that, there is no reference to IEEE 829, and again, that's because 1207 describes ramp meters, to use that standard to design the ramp meter system. You use the IEEE 829 document to do your test in familiar formats and definition of terms. So that makes the incorrect answers as “B,” 829 does provide familiar documents and steps; “C”, 829 does provide standard definitions; and “D,” using standard steps and definitions from 829 results in a document that can be easily reused in later projects for when the existing systems are expanded at a later date. So if you used 829 and everything's documented correctly, we did everything correctly, we want to expand the system later, so we can just start going to the master test plan and say, "Oh, we haven't changed any of the modules. We're just adding one more ramp meter, but we have to go through the final acceptance test again.” So you can just pull out that level, read the master test plan. It'll tell you which level test is the one that you need because we're not redesigning the system from scratch. Saves lots of time.

Again here, we're going to do another activity. “Which of the answers here is not part of a level test plan?” So your choices are a) introduction, b) test details, c) planning for multiple levels of tests, or d) test management. So one of these answers is incorrect and the others are correct. The correct answer here is c) planning for multiple levels of testing. Master test plan is the document that coordinators the multiple level test plans. So you will not have a level test that describes how all the level test plans fit together. That's done at the master test plan level. So that makes “A” incorrect. Each level test plan includes an introduction. “B,” each level test plan includes test details. And again, “C,” each level test plan includes test management. That wraps up Learning Objective 2.

So here we began by discussing the purpose of the test plan, which is an overall document for planning and managing ramp meter test activities, and we answered the question, "What is a test plan?” And we answer that as a test plan is documents to describe the testing scope, the testing approach, the resources required and the schedule, plus some other items, general. We learned that test plans are structured as level test plans for all projects, plus master test plans for large, complex projects. And finally in this learning objective we identified that the test plan content includes introduction, test details, test management, and other general information.

We're going to move on to our third learning objective, which has three key points. In this learning objective, we will begin with an overview of test documentation. Next we will develop an understanding of the differences between test plans and test documentation, and that will be followed by an overview of a test design and the relationship between test plans, test design, test cases, and test procedures.

And we mentioned this earlier—ITS standards do not include the standards for test documentation. And again, we used IEEE 829 as a framework for test documentation to provide standard definitions and formats for everyone working on the project. And again, as we learned in the previous slide, a master test plan may be used for a complex system with multiple subassemblies. A master test plan is not required for simple systems that only have a few levels of testing.

According to IEEE 829, the master test plan if used defines test documentation requirements, and IEEE 829 also includes a detailed list of deliverables. Some of the deliverables are developed prior to test execution, such as test plans, test design, test cases, and procedures. Deliverables created during an after-test execution include test logs, anomaly reports, and test reports, and if we have a master test report it will also be included.

This graphic depicts the deliverables expected to be produced, and these deliverables are identified and planned before the tests are executed. So everything you see on this slide really should be done at the very beginning of the project, during the first 15 or 20 percent of the project. All of these should be laid out, they should be documented, and again, at the bottom, we're starting with the level tests and we move up to unit test design, subsystem, and master tests. So the bottom three rows are level tests—those are three levels—and at the top is master test plan that shows how all of these go together. Now this graphic depicts the deliverables expected to be produced, but here notice that anomaly reports are created at each level of testing. So at every one of these levels, we're going to do a testing of—say a level is the hardware for the ramp meter. We're going to test the hardware. We're going to test the ramp meter software. That's another level, et cetera. So we've put all these levels together like we talked about, and each one of these we've had a test design that we've done ahead of time, so now each of these level tests is going to have a test specification, it's going to have a test procedure, and the test procedure is going to include the expected results. Meaning, from the beginning we planned that when we get to the end and we're testing it and we turn on red phase number one, we expect that to work. So the expected results are included in the beginning. So as the test is conducted at the end, we're simply filling in the actual results. We run the test, we look at the expected result at each one of these levels, and we could get an anomaly report. We'll talk about that later.

And again, ITS standards do not include standards for test documentation, and we mentioned this before that 829, we use that as a framework, and the framework for test documentation includes all the information that is to be delivered at the end. So in IEEE 829 format, we do our test documentation at the beginning with all the information, including the expected results. So test documentation includes test cases, test procedures, test reports, and other documents. The documentation also includes the test inputs, the test output data, the expected output data, and the test tools that were needed to conduct the test procedures. Test plans on the other hand define the required test documents and therefore developed earlier than the test documents. So the plans are actually done before the documents. You plan the test, and from the plan you create the documents, including the expected results. That's all done up front, in that order.

IEEE 829 defines a test design as a test document that details the test approach and identifies the features to be tested. So the features are derived from the requirements test case traceability matrix, and we'll go into one here in a little bit, an example later on here. The test design identifies the associated tests. It usually organizes the test into test groups of test cases and test procedures. The design says, "These are the cases we're going to test, and these are the procedures we're going to use in the test cases.”

So now we're going to get into a little bit more detail. This graphic—this shows an example, RCM requirements test case traceability matrix. So this traces each test case back to the associated requirements. So again, as you recall from the “Vee” model where it showed the red arrows to the left, we're trying to make sure that our test case is going to verify that every requirement was met. So as you can see here, each requirement lists the title and identification number. So we have a requirement, and each one has an identification number here in the left column, and then each major requirement identification is broken down into a minor requirement identification with the test case. So here we say we're going to test meter lane main configuration. So to do that, we have to know the maximum number of metered lanes, the number of metered lanes we're actually using, and the configuration. So these are major, and then the majors break down into minors. So they're numbered in this indented format here. And again, if we're following IEEE 829 and NTCIP 1207, this is what the results—this is what you're going to be making. So you would do something like this at the beginning of your ramp meter project so when we get to the end we know how to make sure we have a test case that goes back for each requirement.

The workflow of the test case sequence is shown. Only one test design is associated with each test plan, but one test design may be associated with multiple test cases, but only one test case is associated with only one test design. That might seem confusing, but again, you might be wanting to test—might be testing a red phase and you're testing a green phase, for example—those are two different requirements and there's two different procedures. We might have—those would be two different test cases, Test Case Red, Test Case Green, but we might have a procedure that would test both of them in one procedure. So that's really what we're saying here. And if we have very simple devices, NTCIP, you can combine the test case and the test procedure into one if you're just doing something very simple. But again, we might want to break those out because we might have one procedure that tests multiple cases.

This graphic depicts the relationship between the test plan, test design, test cases, and procedure elements, the test sequence, and beginning from the test, the unit test plan includes the scope, resources, and methods. So we can see that at the top. The unit test plan flows into the unit test design that provides detailed test methods for each feature to be tested. The unit test design flows into one or more test cases that describes the test inputs and outputs, and then finally the unit test cases flow into the test procedures to describe the test set of instructions to execute each test. And again here, as we described before, you can see that in this case the test procedures can handle one or more test cases here at the bottom. So unit test procedure one would do Test Case One and Test Case Two into one procedure, so we can verify two different things in one procedure.

We're going to do another activity at this point. The question here is: “When is the test documentation completed?” And the possible answers are “before the test is executed,” or “do we do the test documentation only during the test execution?” “Do we do it after the test is executed,” or is the “test documentation completed during and after the test execution?” So I'll go ahead and open up the poll here. The correct answer is “D.” Test data and test summaries are both part of the documentation and taken both during and after the test execution. So that would make “A” incorrect. The test plan is developed before test execution, but it's not filled in with the test documentation at that point. “B” is also incorrect. The test data is recorded during the test execution but the documentation also requires summaries of what happened after the execution. And “C” is also incorrect. Test summaries are documented after the test execution but it also includes the test data during the test execution. So the correct answer here is “D.” That's the only one that's done both before and after.

So wrapping up Learning Objective 3, what we learned is that the test deliverables according to IEEE 829 include a detailed list for LTPs. The test plans are developed and delivered prior to the test execution, not during. Test documentation are delivered during and after the test execution, as we just talked about in the prior activity. We learned that IEEE 829 defines the test design as a test document that describes the details of the test approach, features and groups of tests to be performed. We learned that only one test design per plan, but the test design may be associated with multiple test cases. And then finally, a test case may be associated with multiple procedures, and vice versa.

Moving on to our fourth learning objective, our fourth learning objective has three key points. First, we will begin by identifying the key elements within NTCIP 1207 standard that are relevant to what is covered in the test plan. Next, we will develop a requirements test case traceability matrix for ramp meter control unit. And finally, we will review the key elements in the conformance statement.

So recall when you took module A309, a ramp meter control unit is a system in which the entry of vehicles onto a freeway from an on ramp is controlled by a traffic signal that allows a fixed number of vehicles to enter from each metered lane of an on ramp during each ramp meter cycle. So what we're trying to do is—instead of a platoon of vehicles coming from—say a frontage road has a traffic signal. The traffic signal turns green on the frontage road. A big platoon of vehicles comes along, they all merge at once, and you can slow down or stop the freeway. So what we're trying to do is to smooth out the vehicles entering the ramp so that we're staying within the capacity of the freeway, not having peaks and valleys there. So the ramp meter control consists of a field controller, a suite of sensors, and warning signs and signals.

So before we go into the test documentation, we'll kind of start—refresh here. We'll start with what a typical ramp meter control system configuration consists of, and it consists of field hardware located at the roadside and the ramp meter control software. So this particular example—there's lots of them as we talked about early on. Ramp meters have been around for a long time. Here we're talking about one that conforms to NTCIP 1207 Standard. So the field hardware here that we're showing in this graphic includes advanced traffic controller, and these field hardware here, this could be one that's shown. It's an ATC that would conform to the ATC 5201 standard. It's a Linux-based controller that's made by pretty much all the manufacturers now. Or it may be an ATC 5202 standard. That would be like the 2070e, would fall under that standard. So here we're showing one. Legacy ramp meter control units exist, but again, for the purpose of this training we'll exclude the legacy proprietary controllers that a lot of the manufacturers made, and a lot of those are nearing the end of service and they'll likely be replaced with ones that have an ATC 5102 or 5101 and will be NTCIP-compliant. So we're kind of just bypassing the legacy and just move on to what is going to be coming up in future deployments.

RMC field hardware also includes a communications device that connects to a central traffic management center, like Ethernet or fiber converter usually. RMC systems typically include vehicle sensors. You want to know how many vehicles are approaching so the control algorithm can do it. Those are typically inductive loops, video detection, magnetic. They could be radar. There's other vehicle detectors. And then ATC outputs drive load switches that control sometimes two--head or sometimes three--head traffic signals based on your requirements and your user needs. And then beyond the signals, the other field hardware is housed in a roadside electrical cabinet. There's standards that do that, of course.

The RMC software app provides a means to configure the RMC parameters at installation and commissioning. So there's going to be a piece of software that runs in the ATC controller, and there's ways to configure it and then there's going to be a control algorithm. So part of the control algorithm sets the outputs to the lamp switches according to the inputs that are read from the vehicle detectors. And then the software app, the third aspect of the software app, communicates to and from the central system. So we showed that here in our graphic.

So if we're doing a ramp meter control test environment for unit testing—so if we wanted to break this down into units and then each of the units would be tested as different levels. So again, we're doing the level testing as we talked about previously. So we would take this graphic—so the test environment, if we were going to test this in a lab or on the street, the test environment would consist of the software—and the ramp meter software is installed in an advanced transportation controller—and then that is connected to a vehicle detector and the traffic signal. So that would be our test environment shown in this graphic. The test software is installed in the computer, and that's done to simulate the traffic management center using NTCIP objects and dialogs over a communication network. So in the test environment we're showing here, we're not necessarily connecting it. This might be done in a lab, or maybe in a parking lot. I mean, we've done it both ways, whichever way you want to do it. But what we would do to test software would be—it's installed in probably a laptop or a computer in the lab. It would simulate the dialogs here of an NTCIP 1202 protocol, simulating what the central would be, out to the advanced transportation controller running our ramp meter algorithm. Ramp meter algorithm sensing the detectors, and then that runs this algorithm, puts out signals, and then communicates back to the central office. So you have your “gets” and your “sets” going back and forth here, per 1207.

NTCIP 1207 Standard includes RMC objects shown and conformance groups shown. So if I went into the 1207 Standard, we would see these conformance groups. And again, a conformance group is a set of managed objects. So we'll have these conformance groups, and you can look up in those clauses in the standard, and these are listed as mandatory or optional, and we'll get to that later.

This is a list of required information used to develop test documentation that is outside the NTCIP standards. So if we go back to—recall from the list from our A309a and A309b module—this is what we got from prior modules. If you went back at this point, you should know about user needs, requirements, dialogs, the PRL traceability matrix, test case traceability matrix, and then we're going to do test cases and test procedures. So from taking 309a and b, this should all be familiar, so we're just flowing through all this traceability at this point.

So what's showing in this graphic is a list of required information used in developing test documentation, and again, this is an example test case matrix. So here we're correlating each requirement to a test case. We want to make sure that when we had our user need from A309 and we went to requirements, at this point we want to verify each requirement. So each test case has to be correlated to a requirement. So that's what we do in this matrix here. So we have a requirements identification—like we had before when we developed our requirements, and now we have, each one of those has a test case identification. So as you can see here, Requirement Identification 1.1 is to be verified by Test Case 1.1.

We're going to talk a little bit about a conformance statement, and a conformance statement is required by any supplier or manufacturer that says they are claiming to be conformant to NTCIP. If you have a manufacturer, a supplier, or an entity that is telling you that they are conformant to NTCIP 1207, they must provide a conformance statement. So the conformance statement is provided by the manufacturer to the user or to the person writing the procurement contract. Separate conformance statements can be provided for specific implementation. So RMC may have a conformance statement, an actuated signal controller may have a separate conformance statement, and others. Dynamic message signing may have a different conformance statement. A conformance statement includes the PRL that indicates the features, which are the objects, that are supported in the implementation that they are claiming to be NTCIP-compliant. So if you have an actuated signal controller or a ramp meter controller, it should come with a conformance statement. If not, you can ask for one, and it'll have a protocol requirements list that shows that this supports these objects. There's lots of 1207 objects—some are mandatory, some are optional—but it'll tell you which ones are included.

At this point, we're going to do another activity. The question we are asking here is: “What is the primary purpose of the RTCTM?” Requirements Test Case Traceability Matrix. What is its primary purpose? So we'll go ahead and open the poll. The correct answer is “D.” RTCTM depicts the test cases that will be used to verify each requirement. That's its primary purpose. “A” isn't correct. Testing workflow is not part of the level test plans. “B” is also incorrect. User needs to requirements are part of the protocol requirements list, not the traceability matrix. And finally, “C” is also not correct. Optional and mandatory objects are provided by the manufacturer and those are part of the conformance statement; they're not part of the traceability matrix. The correct answer is “D.”

We're going to summarize Learning Objective 4. A conformance statement is provided by any supplier claiming to be NTCIP-compliant, in this case 1207. We learned that separate conformance statements can be provided by specific implementations. So if I'm a manufacturer and I make ramp meter controls, actuated signal controls, and message signs, I would have three different conformance statements. And in fact, if I had two different or three different actuated signal controller pieces of software, I might have different conformance statements for each one, because it tells you which of the objects are implemented in that controller algorithm.

So we're going to move on to our final learning objective, number 5. It has three key points. We will look at a process to create test documentation based on our test specifications. So that's where we're going to start here with. Next, we will address the consequences of test boundary and error conditions, and in the end we will wrap up by describing some of the test tools and equipment that are available. So in this graphic—this is starting to look a little bit more busy—but this is what we're going to end up with in the end if we did everything correctly. We've gone through all of the A309 courses when we complete this course. We're going to have user needs, requirements. We're going to have protocols and we're going to tie them all together clear out into the test case documents.

So the requirements test case traceability matrix—and again, that is used to identify the features to be tested at each level. So we had an example of that in a previous slide. The features to be tested are based on the RTM. So here we would provide a test case, an identifier, a test case title, a test case description, the variables—like the inputs, such as the settings—and then the pass/fail criteria. So this ties everything together. We're going to have a test case identifier, a title, a description of what the test case is, what are the variables, what are the input stimuluses, what are the expected outputs, and what is the pass and fail criteria. So that's all wrapped up here in this matrix. So the test procedure is shown here in the lower table. So this is where we had our test case. This is our title, and the title is “Test the Boundaries” this is how we test the boundaries. So this is our procedure. We have multiple steps down here. The test procedure lists each step and the test procedure for each step. So this is step one, this is the procedure, set the maximum number of lanes. Step two, set the number of metered lanes. Step three, set the number of metered lanes and record the response, et cetera. So this test procedure is completed before the test is executed; therefore the results will initially be blank. So we do this, again, very early in the project. So once we know what our user needs are and we know what our requirements are, we can finish the rest of this before we do the design. So it would be very nice if I were a person designing the control algorithm or I'm designing any piece of that ramp meter hardware to have all this up front, so as I'm designing my software and I'm testing my software, I can test it to the final results for that level. That's very important to do this up front. Procedures completed before the tests are executed. So these are the expected results, and the actual results will be blank. That would be something out here, as we'll show later. But this is the expected results, and when we're actually executing it; the person who’s sitting there actually executing the test plan—we'll add another column with the actual results and we'll see if they're expected or unexpected.

We'll talk a little bit about boundary conditions. As we can see, a complete ramp metering system, especially when it's combined into larger systems—so we have a central system, central office system, traffic management is running and connected to the signal controllers, ramp meters, signs—whole bunch of things—that can be very complex. So it's not possible to test every combination of everything. We know that. So what we try to do is do positive and negative testing and boundary conditions, since we can't do—there are billions and billions of different combinations. So when we're doing a positive test, the positive test subject, the device under test—and we'll use the term DUT; you'll hear that a lot—Device Under Test. That's what we're testing. That's the ramp meter that we're testing. So the positive test would subject the DUT to valid inputs, described in the test procedure. We'll give it inputs that we know that are going to happen on the street. So valid inputs, we would send the unit under test, valid, correct NTCIP objects. We'd look at the conformance statement for that device under test and we'd send it valid objects, the ones that are actually part of the conformance statement, and with the values that are within the specified range, using dialogs that are correct with the sequence and timing as described in the test procedure.

So we're going to give it things we want it to do and expect it to do it correctly. So a pass means that the DUT responded with the expected outputs as described in the test procedure. Fail means that it didn't respond or it responded with unexpected outputs. So that's a positive test. A negative test, on the other hand, subjects the device under test to invalid inputs. So described in our test procedure, we're going to give it some things that we know it should not like, it should not accept them. So invalid inputs include combinations of incorrect NTCIP objects, values that might be outside the specified ranges, like 17 metered lanes. That's not going to happen. Or maybe dialogs with invalid sequences or timing—things came in the wrong order. So negative testing assures that the device under test will respond to invalid inputs as expected by the test procedure. So the test procedure is going to say, "Here are these invalid objects that I know are wrong, and you should handle them thusly." So the invalid inputs should not be processed, but the device under test should continue to run normally and control the ramp normally and should respond with an error message and type as described in the test procedure. So we give it invalid results. A pass would be, it says, "Oh, those aren't right." It throws an error message back as an object so that you know that it happened, and the ramp runs normally without being affected. That would be a negative test procedure.

T309: Applying Your Test Plan to Ramp Meter Control Units Based on NTCIP 1207 Standard Version 2

Test procedures also include tests of boundary conditions, and again, this is because we can't test every single combination of timing and everything that can be done in a complex system. So we will look at the critical boundaries. So in boundaries testing, the test procedure will describe at least three sets for each boundary condition, maybe more. So at a boundary, we will test just below the limit, just above the limit, or exactly on the limit, and the test procedure will also describe the expected results, whether below the limit is valid or invalid, whether above the limit is valid or invalid, and whether on the limit is handled as above or below. So if you're on the limit, maybe it might be handled as above the limit or below the limit. So it's very critical to test the boundaries, and we'll show an example of that later.

If the boundary is handled as expected by the test procedure, the device under test should process the object and dialog successfully with the expected response, and if error conditions occur, the test procedure expects that the device under test will respond with an error message and remain in normal operation without loss of communication. So again, it's sort of like we talked about before. We're just testing just below, just above, and right at the boundary, and we describe exactly what should happen, what's expected at each one of those in our test procedure.

And again, NTCIP testing is very complex, as you can see here, and it requires advanced test planning and preparation of documents. And again, the test must be executed per the test procedure, and the test results are reported in the test documents. Every NTCIP object should be tested. Because of the large number of valid inputs, 100 percent coverage of each possible combination of valid objects, again, is usually not practical. And again, what we're going to do is—the test procedure will typically test the boundaries and then a sample of valid inputs over the range. So every boundary condition is always tested, and again, because of the large number of invalid inputs, error conditions are selectively tested in order of criticality. So knowing that we don't have enough resources to test every combination, we put them in order of criticality. Mission critical. So mission-critical errors that could compromise the safety or result in a significant loss of performance are always selected for test every time. If it would make the ramp fail or cause an unsafe condition or even somebody would have to go out and run it manually or reset it—always test those. So mission-critical is always tested. And then others are selected for test in descending order of criticality as your resources allow. So progression testing is conducted per the test procedure. So the concept of progression testing is we're progressing along the test design or test cases, so every new thing that we added is progression-tested, of new and corrected features.

T309: Applying Your Test Plan to Ramp Meter Control Units Based on NTCIP 1207 Standard Version 2

Regression testing is a concept where we test on features that were not changed to ensure that we didn't break anything. So we add a new feature, you do a progression test of the new feature. If we get through that okay, we do a regression test where we may go back and sample in reverse order, beginning with mission-critical, of the things we didn’t change to make sure we didn’t affect anything else. And again, it's often impractical to test everything over and over again like that, but regression tests are conducted selectively and they're focused on parts of the system that might be affected by a test failure. If you did a progression test with a new feature, you pretty much know that the parts that it could affect and the parts that it couldn't affect. So again, put those in order. Test the most important things, because we don't have enough resources to retest everything every time.

As we know from previous slides, NTCIP testing is a complex process, and you can see that this got to be pretty complicated, but fortunately the good news, NTCIP test tools are available to simplify and automate your test execution. The tools are off-the-shelf and are capable of performing specific NTCIP levels of the standard. The tools support communication testing, such as SNMP that we learned about in prior modules, using simple configuration screens. So you can buy an off-the-shelf piece of test software, put it on your laptop computer and do SNMP testing and do object testing. You can actually, in these test tools, you can automate the testing. So you can write scripts. So you can manually run through a test. You can do a “get” and a “set” and say, "Oh, I know I'm going to be doing this over and over again," so you can write a script that says, "Do a “get,” get a “set,” report something.” So you can write test scripts. These test scripts can be developed beforehand, or they can even be reused from a prior project. So once you get using these tools, you get very familiar with them, and you can start reusing large portions of prior projects, some of the same test cases, might be able to use those scripts and just modify the script a little bit, turn it on, and it runs a complete test procedure for you. The tools support various protocols that are commonly used in transportation such as PPP, PMPP, TCIP/IP and others, and again, I think we talked about that in the prior modules, all these different protocols. The tools also support each of the communications media typically used in the transportation system, which is usually Ethernet or asynchronous serial ports.

So we kind of break this test tool down into active and passive tools. Passive NTCIP test tools provide a capture and display of objects and dialogs without affecting the device under test. So this might sometimes they're called “sniffers.” So you'd put them on the communications line. You'd be running the test tool. It has no effect on the test you're running; it's just automatically recording and logging clicks going over the lines. It's automatically collecting data for you and putting it into a file. Then you can reproduce the file, print it out. or look at it on the screen, know what happened. So again, passive means you're not really affecting the test. It's not really conducting any part of the test; it's just watching the test and recording for you. So passive NTCIP test tools are used sort of like a data analyzer that monitors the data exchange and has a display and captures the records for further analysis. And again, they don't provide any stimulus to the device under test and do not respond to stimulus. There's many examples of these that are available and you can find them online.

Active NTCIP tools differ from passive in that the active tools actually send a message to the device under test as a stimulus and then timestamps or records data response. So active tools are used as the main NTCIP test tool that can generate valid and invalid messages. They're very powerful and they're very useful. So you can use to create valid messages and invalid messages automatically per the test procedure. It will timestamp them, display in logs the device under test response, and you can look at the responses for pass/fail analysis per the test procedure. If you buy active off-the-shelf NTCIP test tools, they typically support all the mandatory and optional standardized objects. So you get your conformance statement from your manufacturer, you look at their claiming compliance, it supports all the mandatory ramp meter objects. It will list the optional ones that they support in this control algorithm, and if you buy one of these off-the-shelf active test tools for NTCIP 1207 compliance, you should be able to use that directly. All this should be able to be supported by the test. But be aware that the off-the-shelf tools do not necessarily support block objects or manufacturer-specific objects. These vary among manufacturers. So if you have no block objects and no manufacturer-specific objects, you most likely can use the active test tool directly, but if you have block objects or MSOs that differ among manufacturers, you're kind of on the hook. You can use the tool but you're going to have to develop your own objects and dialogs. You might have to develop your own special purpose software or in one of these active off-the-shelf tools you will have to go ahead and get from the manufacturer what the content and the responses of those objects are, and you're on your own as a test case developer and a test procedure developer to write those yourself. But, once developed, the special software can be used by the active tool for execution and automatic recording. So even though you have to develop it up front, if you're using some special objects, it will still—once you've gone through that and got it to work, you can use it over and over again, because the active test tool, they'll put out those and get the responses. And then those scripts can be reused. And there are some—I've got some listed there. I probably missed some, and if I missed somebody's, I apologize, but they're commercially available active test tool:DeviceTester, NTCIP Exerciser, NTester, and there's others listed there. So you can go ahead and look for those online on your own.

Okay, we're going to wrap up Learning Objective 5. We developed example test documents, including test cases, procedure, and the documents for the test results. In this learning objective, we also learned the difference between positive and negative testing and the importance of each. We learned how each boundary condition is tested. We learned the complexity of NTCIP testing and the number of input combinations usually preclude 100-percent test coverage of every possible object and dialog combination, due to resource limitations. So a good test strategy would include 100-percent testing of every boundary condition followed by sample testing across the range of input values, as your resources permit. Each mission-critical function, whether it's safety related or performance related, should be tested every time, including negative tests to ensure that it's reporting errors correctly, and alarms are created correctly. Progression tests of each new feature is followed by a regression test to ensure unaffected features were not broken. Although NTCIP testing is complex, the good news is that test tools are available and they're available as both passive tools for monitoring the display of communications dialogs, as well as active tools that simulate the device under test and record the responses in real-time. And also we learned that you should be aware that block objects and manufacturer-specific objects vary among suppliers and require special software development for software automation. But once you've developed those, with the automated tools, you can use that and run those scripts over and over again.

To wrap up the module, what we've learned in this module as a summary, we learned that ramp meter control units are tested to ensure that RMC units meet the expected user needs and associated requirements when installed. In addition to testing, other verification methods include, inspection, demonstration, and analysis. We can verify by any one of those three methods. Test plan is used to plan and manage ramp meter tests, including the scope, the approach, the resources, and the schedule of the test flow. According to IEEE 829 standard, the test documentation includes test data and test summaries that are delivered before, during, and after the test's execution by the person running the test, the test operator. We learned that key elements of a ramp meter that are relevant to the test plan include a configuration, detector inputs, and signal outputs. These key elements are included in a requirements test case traceability matrix that is based on test cases. We learned that manufacturers claiming conformance to NTCIP 1207 must always provide a conformance statement that includes the requirements list and the features supported in the RMC implementation. For boundary conditions and

T309: Applying Your Test Plan to Ramp Meter Control Units Based on NTCIP 1207 Standard Version 2

error conditions, a sample of valid inputs of the most critical functions is tested at each boundary condition. Passive tools monitor and record, while active tools simulate to create responses. The active tools actually provide the stimulation to create real responses. Special tests must be developed for block objects and manufacturer-specific objects. If you're not using standardized objects that are shown in the 1207 Standard, you have to create those themselves, but you can put them in the test tools and use them over and over.

That wraps up our module here. These are the resources. Again, I always encourage everyone on that first bullet to read the *System Engineering for Intelligent Transportation Systems*, at least the first 15 or 20 pages. There's a lot of good information in there. And then again, we have a reference here to the standard for 1207 for ramp meters and we're referencing the earlier parts of this. If we have things in here that you missed, you can go back to some of these. And again, this is the sequence of the courses for the testing. They are from T202 to T204. And again, we did have a reference to ATC 5201, so that's the Linux-based Advanced Transportation Controller the manufacturers are all making now, and then there's also a reference to—if you go on the ITE website, there's ATC 5202, which is a standard for the model 2070 e-controller. That was done by the joint committee, so if you're using 2070s you'll find that one to be very familiar. Then finally we have the IEEE Standard. That's the 829 that does the formats and definition terms.

End of T309_Final.mp4