Nicola Tavares: Welcome to the ITS Standards Training

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, 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 helpful.

Nicola Tavares: Throughout the presentation this "Activity" slide will appear indicating there is a multiple choice pop quiz following this slide. The presentation lecture will pause at each quiz section to allow you to 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 on your answer choice.
Please help us make even more improvements to our training modules by completing the post-course feedback form.

This module is T321: Applying Your Test Plan to the TMDD Standard

Nicola Tavares: Your instructor, Patrick Chan is Senior Technical Staff

Consensus Systems Technologies. Patrick has been involved with the TMDD Standard since development work began on Version 3.0, since e mid-2006 and is the current consultant &developer of the ITE TMDD standard. In addition, he has been involved with the development of several other ITS standards, including NTCIP 1203, the DMS data dictionary and Patrick has 23 years of ITS experience, including 4 years with a public agency. The next voice you will hear will be of your instructor

Patrick Chan: So, thank you, and thank you for joining me for this module, "Applying Your Test Plan to TMDD Standard." So what we are going to do today is review the basics of creating a test plan for the TMDD standard. The details of your test plan may vary significantly from another agency's test plan based on the complexity or the potential risk of your system. But if you have any experience-- for example, if you have any experience using the TMDD standard or have a low-risk implementation, some of the details that will be presented may not apply to you. But for agencies implementing TMDD for the first time, most of the details presented today should be helpful. The target audience for this module includes engineering staff, which includes the specification writers who are responsible for specifying and implementing the TMDD standard for center-to-center system; operations and maintenance staff, who are the users of the system, who wish to better understand what TMDD is; systems integrators, who are responsible for implementing and testing the TMDD standard for center-to-center systems; device manufacturers who are responsible for providing the software systems that use the TMDD standard for the center-to-center interfaces; and testing contractors, who are responsible for setting up and performing the tests.

Patrick Chan: There are real recommended prerequisites for this module. T101 provides an introduction to the need for and applicability of ITS standards testing. T201 provides understanding of the need for a test plan within the context of the overall system testing process, or the lifecycle of the system. T202 provides materials on the three elements required to develop test specifications. That includes test design specifications, test case specifications, and test procedure specifications. C101 provides introduction to the NTCIP framework and offers design solutions to fulfill operational needs, including center-to-center communications. A321a provides information on how to identify the appropriate uses of TMDD standard and using the tools in the standard, such as the needs to requirements traceability matrix. And A321b provides information on how to specify requirements that are supported by the TMDD standard, and using the tools such as the requirements traceability matrix, which is in the standard.

Patrick Chan: This is the curriculum path, the recommended curriculum path leading up to this module. There are two possible paths the user can follow in implementing systems based on the ITS standards. The path to be taken depends on the standards needed to implement the system. So for TMDD, which uses the systems engineering process, the curriculum path follows the systems engineering process curriculum path. So this graphic shows the recommended sequential curriculum path leading up to this module. Many of the concepts that we'll discuss in this module have already been introduced in one of the earlier modules, one of the prerequisite modules in this curriculum path. But this module brings all these concepts together, and specific to testing and implementation using the TMDD standard.

Patrick Chan: Next we'll go over some of the learning objectives for this module. One: Describe within the context of a testing lifecycle the role of a test plan and the testing to be undertaken. So what we'll do here is identify where testing belongs in the lifecycle of the whole system. And by the system lifecycle, we mean from the idea when the system is first conceived, to when the system is retired. So for this learning objective, we want to introduce how a test plan identifies when testing is to be performed. Number two: Recognize the purpose, structure, and content of a well-written test plan for a TMDD-based system interface. So, here we'll identify what information may be included in the test plan for TMDD-based system interface. Number three: Describe the application of a good test plan to a TMDD-based system being procured using a sample TMDD test plan. So we'll identify some of the key elements of the TMDD standard which are relevant to what is being covered in the test plan. The TMDD standard includes tools in the form of traceability matrices that can be used to develop the test plan.

Patrick Chan: Number four: Identify the process to write a test plan in the context of the requirements of TMDD that have been selected by the user. So we'll be using the traceability matrices in the standard. We're going to review how we're going to use those traceability matrices and use examples-- and provide examples using the TMDD standard. Number five: Analyze how to ensure conformance with the TMDD standard. So identify some of the key elements of the conformance statement that's in the standard. And number six: Describe test documentation for TMDD-- test plan, test design specifications, test case specifications, test procedure specifications, and test reports. So we'll review some of the details of what information is included in test documentation, and we'll also take the opportunity to identify tools that can be used for testing the TMDD-based system.

Patrick Chan: So let's start with learning objective number one. So, here, describe within the context of a testing lifecycle the role of a test plan and the testing to be undertaken. So, first, explain why testing is important. So we're going to discuss why we test, why do we perform the testing. And we'll also identify how to break up, partition, the testing, and when during the system development lifecycle requirements are tested. So we're going to discuss the different types of testing that might be performed and when each type of testing would be performed.

Patrick Chan: So, why do we test? Well, we test to verify that the system works. We have procured a system; we want to make sure that, yeah, that the system works the way we expect it to. And sometimes we test to meet payment milestones. So once a contractor passes that test, it's a payment to the contractor, to the provider. But technically, we test because-- to verify-- in this case, for TMDD-- to verify that the system interface meets the procurement specification and satisfies the requirements. Meaning, we have a procurement spec. We want to make sure that the system that was provided us satisfies those requirements. It really answers the question: Was the system built right? Did it satisfy the requirements that were in the procurement specs? We also test to identify errors or bugs so that they can be corrected. By testing, we can detect if there are any problems with the system before it gets implemented, so we can correct them before the system is actually being used. And finally, we test to validate that the system interface satisfies the user and operational needs. So by this we mean: Did we build the right system? It might be nice that the system that's provided fulfills all the requirements; it does everything it's supposed to do. But if it doesn't solve the problem that we set out to fix, that the system was supposed to solve, what good is the system? It meets all the specifications, all the requirements, but if it doesn't solve the original problem it's trying to solve, what good is the system? So these are some of the reasons why we test.

Patrick Chan: This is the famous V diagram. So the reason why we show the V diagram is because it really represents the entire lifecycle of the system, starting with the idea of how the system came about, up in the upper left-hand corner. Sometimes it's from a regional ITS architecture, or sometimes it's like, "Oh, we have a problem that we're encountering. Let's create a project to resolve the problem." So looking at the left side-- we're going to follow the V diagram and look at the left side. The left side represents the development of the system, beginning with the idea of the system, determining what the user needs are, and that's documented in the concept of operations; what the requirements are for the system; and then developing the design. So that's all on the left side of the V diagram. Then we start building the system. Next, we go to the right side, and here we see the testing phase. So now that we've built the system, we want to start verifying and validating the system. Notice on the right side that we had to do testing-- that's in the testing phase that's circled-- but it also includes the operations and maintenance of the system, any changes or upgrades to the system after we start operating and maintaining it, and before the system is ultimately retired. So this presents the whole lifecycle of the system. Note that there are test activities for each component on the left side with the right side. For each step of the development of the system, there's some type of testing that can be performed so you can verify or validate the system that is being built. For example, as you define the user needs, that the system is satisfying the concept of operations, you can start thinking about, "How can I test the system and verify and validate that it satisfies my user needs?" So be thinking about testing up front at each step, you can quickly determine how to verify or validate the system, and the design the system properly.

Patrick Chan: Let's look at a different definition why do we perform testing, and we're looking now at what's in IEEE 829, which is the standard for software and system test documentation. According to its definition, the testing process provides an objective assessment of the system products through each system's lifecycle. So, some examples that it provides is at the completion of each development iteration-- we saw that earlier; at the installation and go-live; during operations and maintenance also; during system upgrades; and during system replacement. So it's not only at the development phase that we test for, but we really do perform testing throughout the entire system development cycle, all the way up to system retirement. So for example, testing may occur during the operations and maintenance phase. Even though we've approved the system, even during operations and maintenance we might come across a bug, an error, a misbehavior that's discovered during operations and maintenance. So we may at that point upgrade the system to resolve that problem, to resolve the bug, to fix the bug. So when we do that, we want to confirm that the upgrade works, that it fixes the problem, but we also want to test to make sure that it doesn't break something else. And during the system replacement, we may also do testing to verify that the new system works just as well as the old system did, or hopefully better.

Patrick Chan: Let's go a little bit more into detail. We'll talk about validation. Validation, again, answers the question: Can I operate the system and satisfy my stakeholders' user needs? Again, that is, did we build the right system? So validation confirms that the system meets the needs of the users. The system may fulfill all the requirements and specifications but, again, if it doesn't meet the needs of the user, it may not be considered successful. So when is a system considered validated? The system is validated when it's approved by the key stakeholders and agencies that are using the system; when all the project requirements have been fulfilled-- that means the specifications have been fulfilled; and corrective actions have been implemented for any anomalies that have been detected earlier in the testing.

Patrick Chan: Verification. So, what is verification? Verification is actually an ongoing process that builds quality into the system through a systematic approach of verification of the requirements. That is: Did I build the system right? Did I fulfill all the requirements that are in my project specifications? So it's performed at each stage of the V diagram or whenever a new feature is added, or when a change is made. For unit/device testing, we are verifying that the detail design, for example, by only testing the information exchanged in a controlled environment. It really answer the question: Did the data, the sequence of data, exchange correct?

Patrick Chan: For subsystem verification, we're verifying high-level design. So we're going to look at the high-level design and verify that it meets-- that the design is correct. So for this test, we'll consider-- for TMDD, we'll consider other environmental factors, such as the actual communications with the TMDD software. But first, we're going to look at the behavior of the software. So we're really just asking the question: Is the data exchanged correctly? So we're not going to look at software, per se, but we'll want to make sure, TMDD, is the data exchanged correctly? For system verification and deployment, we're verifying the system requirements. So for this test, we consider how the TMDD software uses the information that is exchanged to provide the services needed to satisfy the stated user needs. So we're answering the question: Is the data used by the software as intended?

Patrick Chan: So now we've reached our first activity.

Patrick Chan: This is a poll, just to review our understanding of the key concepts that we have learned so far in this module. So this is the first of six. So we like to ask: Which of the following is not a reason to perform testing? The possible choices are: Develop the concept of operations; verify requirements are fulfilled; validate the user needs are satisfied; or D, assess a system upgrade versus the existing system. So, please choose one of the answers. Which is the following is NOT a reason to perform testing?

Patrick Chan: So, the answers are-- let's review the answers. The correct answer actually was A, develop concept of operations. The concept of operations belongs in the definition phase of the system lifecycle. It's not one of the reasons why we perform testing. We use it as a basis for testing, but it's not part of the testing cycle-- testing phase of the system lifecycle. B, verify requirements are fulfilled-- that's one of the reasons why we do perform testing. We also perform testing to validate the user needs are satisfied. And we do perform testing to assess a system upgrade against the existing system. So we just want to confirm after a system upgrade that the system did improve the system, and during the upgrade may have features that were added-- we want to test that the features that were added or used to resolve a misbehavior and make sure that it was done correctly and we didn't cause additional problems.

Patrick Chan: So, just to summarize the learning objective number one-- and so far we've explained why testing is important within the lifecycle of a system. We test to verify that the system interface satisfies the requirements of a specification-- was the system build right?-- and to validate that the system interface satisfies the operational user needs for the system: Was the right system built? We also test to identify-- well, the test may also identify how to break up the testing, and when during the testing development lifecycle requirements are tested. So we perform different types of tests depending on what the testing is for, whether it's for verification that requirements have been fulfilled, or for validation.

Patrick Chan: Learning objective number two is to recognize the purpose, structure, and content of a well-written test plan for a TMDD-based system interface. So here we're going to identify the purpose of the test plan and to describe the components of a test plan, explain the purpose of each component. So the next several slides, we're going to introduce a test plan, what it is, what is used for. The test plan will be discussed in general terms, but with some specific TMDD examples.

Patrick Chan: So what is a test plan? Well, looking to IEEE 829, it defines the test plan as a document describing the scope, approach-- whether technical and management-- the resources, schedule of intended test activities, and deliverables. So, the test plan describes the what, why, where, when, who and how the test system will be tested. It identifies what we are testing, the test items, what features we're testing, why we are testing, where are we going to perform the testing-- is it going to be in a laboratories or at actual TMC?-- when the testing will be performed; who will perform the testing tasks or testing activities; and how the items will be tested. It also identifies what risks are associated with the testing and how those risks will be mitigated. The document may be a master test plan or a level test plan. So there are different types of test plans that we may want to consider. That may be a master test plan that describes the testing for the overall lifecycle of the system. So it is the test plan that says, "This is when we're going to provide testing throughout the whole lifecycle of the system, and this is why the types of testing." There can also be level test plans, which might be a test plan for each type of testing or for each phase of the system lifecycle, or the V diagram. So we may have a separate level test plan, for example, that validates the system. We may have a level test plan for when we verify system requirements, or a level test plan just for when we perform testing to resolve a bug that was discovered during operations and maintenance, for example. Test plans are also covered more in detail in module T201, How to Write a Test Plan. T201 discusses the role of a test plan within the testing lifecycle. It summarizes the characteristics of a good test plan, presents the outline of a test plan, and describes the relationship among the test plans and test design specifications. However, what we're going to do over the next couple of slides is just review some of the key parts of a test plan in the next couple of slides.

Patrick Chan: So, a test plan is a high-level plan that defines what items are going to be tested. Is it just a single system interface? Will this test plan considered interface with the existing TMC software, or are there different communications media, communications ports, that have to be tested? Notice that in italics we have, "What portion of the TMDD-based system interface (portion of the TMDD standard> you are going to test and in what order?" So italics represent an example using TMDD standard. What features are going to be tested? So the test plan identifies what functions the system has to perform to satisfy the user needs. The TMDD requirements to be tested will be explained a little bit more in detail later in the module.

Patrick Chan: So, what are different types of test plans? Well, again, we may have separate test plans for each type of testing. There might be unit/device test plans, subsystem verification plan, a system verification plan, a system validation plan, and maybe there's a separate test plan for periodic maintenance. The size and complexity of your system will determine if a single test plan should be created-- for example, a master test plan-- or you may need multiple test plans. As the size and the complexity or the risk of your system increases, you may want to consider having separate test plans for each phase or each type of testing. But there is a master-- you also may have a master test plan that describes how all the test plans work together to verify and validate the system.

Patrick Chan: The test plan may also-- identifies what is the overall approach to testing. So what are the means and the methods for testing? Well, the approach might permit identification of the major testing tasks and estimation of time. So the approach should be described in sufficient detail to permit identification of major testing activities and estimation of time required to perform each activity. The approach will also be determined by the complexity of the system or the risk involved. For a sample simple interface with only a few features and for an implementation that the agency or the vendor has experience with requires less detail than an interface for an agency implementing a standard for the first time, with several hundred requirements. Another example of a high-risk system is a system that supports share control of devices. They may have a different approach because control of devices brings out additional institutional issues that have to be considered. For example, liability. How do we handle the liability when we give control of our devices to other agencies? So, the approach answers really the questions: Does the test item conform to the standard? And does the system exhibit the functionality defined in the specifications? That is, how do we ensure that all the requirements are fulfilled?

Patrick Chan: Next, test plan should include what are the pass/fail criteria. So all the parties have to agree what the criteria is for passing when testing for conformance of standards. The standards are specific in most areas, but there are some things, some requirements, that are user-defined. For example, performance requirements. A software system may have a requirement of a 99 percent availability time, but that's not handled or indicated by the standard. That's not specified by the standard. What are the suspension criteria and resumption requirements? So, we need to-- the test plan may include the criteria to suspend all or part of a testing activity. Sometimes we suspend activities because it's planned, because, for example, "Hey, we reached the end of the day, so let's continue the next day." Or it might be unplanned. So we're performing testing, an issue arises, so-- and we can't continue until we resolve that issue. So how do we suspend testing until we resolve that issue? So the test plan should specify the criteria used to suspend all or a portion of the test activities and the test items. And the test plan should also indicate the resumption requirements. How does one restart the testing activities? Where they left off? Should they perform some precondition steps, perform some activities? If a test is restarted after changing something, should a regression test be required to detect any unexpected impacts resulting from the program modification? The regression test may be performed by comparing the results from the new version with the previous version. So these are all things that you want to make clear to include in your test plan so that there are no questions or no ambiguity when performing the test, or no disagreements.

Patrick Chan: The test environment-- that should be included in a test plan. How is the item going to be tested? So we want to identify what-- the environment for executing the test plans: Where is it going to be tested? What hardware is going to be used? What communications will be set up? What software will be used? And any other preconditions for testing. So the test environment describes the necessary and desired properties of the test environment.

Patrick Chan: Moving on, test plan deliverables. What are the test deliverables? So the test plan should indicate what's going to be provided, what gets delivered, from the beginning of the testing all the way to the end of the testing. It may include test design specifications, which specifies the test approach for a feature or a combination of features, and identifies the associated tasks for testing those features. Test case specifications define inputs, predicted results, and set of execution conditions for a test item. Test procedure specifications specify the sequence of actions for the execution of a test. And the test reports, which are the summaries resulting from the testing activities, which may include incidents. So when we talk about test deliverables, you want to indicate who's going to submit, provide these test deliverables; what are requirements for each deliverable; if there any incidents-- if there's an incident, what-- well, the definition of an incident, by the way, is any event that occurs, whether expected or unexpected, that occurs during the testing process.

Patrick Chan: A test plan can consist of several test design specifications, perhaps one for each phase of the system lifecycle or one for each feature to be tested. Each test design specification may consist of several test case specifications or test procedure specifications. The test case specifications and test procedure specifications may be reused for different test design specifications. This flexibility allows the reuse of test procedures and reduces the cost of producing test procedures, which are generally the most expensive specifications to develop. For example, authentication is an optional requirement for all requests for information or control in the TMDD standard. If we had to rewrite the same procedures for each possible request, it's probably recommended that we just reuse the single test case or test procedure specification for verifying that the authentication requirement is fulfilled. So every time we send a new request, we can refer to the authentication test case specification or test procedure specification to test that, rather than rewriting it each time.

Patrick Chan: Test plan-- staff and resources. So this defines who's going to test the-- perform the testing. We want to identify the roles and responsibilities for each person involved in managing, designing, preparing, executing and resolving issues. Who's going to participate? Who's going to document the results? Who's going to determine if a test item has passed the test? What staffing and training is needed? So this identifies what staffing needs are required based on what skills are required, and identifies the training options for providing the necessary skills. Consider the participants that are involved with the testing: Do they have the skills and the knowledge to perform the responsibilities? For example, does the agency tester have the expertise to test? Does he understand the requirements that are being tested, the operational needs? Does he understand what is being tested? If the skills aren't available, then training should be provided and that should be included in the test plan, indicated in the test plan. Training might include an introduction to the system under test, because it may be the first time that agency actually sees the system, if it's a new system; how to use the test equipment, the test software; introduction to the test plan and test documentation; or introduction to the testing environment. Training may also be needed prior to the start of the test or even before the development of the test documentation.

Patrick Chan: The plan should also indicate when is testing going to take place, identifies the milestones, what the dependencies are, when the submittals are due, the time to perform each test task, and what the testing resources are. What are the risks and contingency plans? So we want to identify what the risks are and how we're going to resolve the risks. So for example, TMDD supports control of devices by other agencies. So device control requests may require testing in the evening when there's less impact, or perhaps setting a test controller for live testing. The test control might be in the laboratory, for example. So you want to define what those risks are and how you're going to handle those risks. And finally, approvals. Who needs to approve the test plans? Who is authorized to approve the test plans that will be used in the actual testing?

Patrick Chan: So now we came to our second poll, second activity.

Patrick Chan: Again, this is just to review that we understand the concepts that have been presented already. Which of the following does NOT belong in a well-written test plan? The choices are the testing environment; the testing plan staff requirements; the pass/fail criteria; or the sequence of actions to be performed. So which of the following does not belong in the test plan?

Patrick Chan: So, we're going to review the answers. The correct answer is actually D, sequence of actions to be performed. The sequence of actions to be performed during the test is defined in test procedure specifications. Is it not defined in the test plans. The test plan does define the testing environment, defines the staff requirements for the performance of the test-- so who has to be there for the testing activities-- and it does define the pass/fail criteria. So we know when test is considered passed or failed.

Patrick Chan: So, just to summarize what we've learned for learning objective number two, we've identified the purpose of a test plan. It describes the scope, the approach, the resources, schedule activities and deliverables of a test. And we've gone over some of the components of a test plan and explained the purpose of each of those components.

Patrick Chan: Learning objective number three: Describe the application of a good test plan to a TMDD-based system being procured using a sample test plan. So for the next couple of slides, we're going to review the structure of the TMDD standard, explain what is and what is not being tested when we're testing a TMDD-based system, and we're going to review a sample test plan-- we're going to provide an example test plan for a TMDD-based system.

Patrick Chan: So, what is TMDD? Well, TMDD is a communications system interface standard designed primarily for the traffic management domain. It supports the exchange of traffic information between traffic management center and other centers. It's really a data dictionary. Think of it like a vocabulary for exchanging incident information, traffic network information, and the monitoring and control of devices operated by another agency. When we talk about incident information, we mean planned events, such as construction, when construction is expected to occur, or a special event; but we're all talking about unplanned events. So, for example, an accident that occurred somewhere in the traffic network. Traffic network information must include information such as the routes, the links and nodes of someone's traffic network. And device information that we exchange includes the location of the devices, the types of devices, and the status of the devices. Control of devices is actually a request to the owner center to do something; it is not really designed for direct control or monitoring of a field device, and we show that in the graphic in the bottom right-hand corner. So we're sending a request to the owner center to say, "Hey, give me information about the device," or "Please make this change to the field device." Change the timing plan, for example. Though the focus is on traffic management centers, this data-- the data that's supported by TMDD can also be exchanged with other domains. So for example, we do support this exchange of information with other agencies such as public safety agencies, transit agencies, or planning organization.

Patrick Chan: So, what's in the TMDD? Well, since the TMDD follows the systems engineering process, it defines the user needs. So, what user needs does it support? It defines the requirements it supports based on the user needs that have been identified. And it defines a single design for each requirement supported by the standard. The design, which we also call data concepts, might be in the form of messages or data elements, and we'll talk a little bit more about that in a future slide. So, the reason why we have the single design though to fulfill each requirement is that it supports interoperability between the traffic management center and other centers. So by specifying, defining a single design, we take steps to support interoperability. And we'll discuss a little bit more about interoperability later.

Patrick Chan: So what's in the TMDD? Well, they say it contains two matrices that support the use of the standard. These matrices are key to the development of the test plan and test documentation for testing TMDD. The first matrix is something called the Needs to Requirements Traceability Matrix. It traces the user need and the requirements that satisfy that user need. The NRTM also indicates what user needs are mandatory to conform to the standard and what user needs are supported by the standard but are considered optional. It also indicates for each user need to be satisfied-- meaning, if a user need is selected for implementation, what requirements are mandatory to satisfy that user need, and what optional requirements may be selected by a project to also support satisfying that user need. The second matrix that's in the TMDD standard is something called the Requirements Traceability Matrix. It defines the design, the dialogs, the messages, data elements that must be used to fulfill a requirement. So again, this is a single design. I have a requirement-- the requirements traceability matrix indicates what dialog, what message, what data element must be used to fulfill that requirement.

Patrick Chan: What are we testing when we test the TMDD interface system? Well, we're testing one or two things. One is compliance with the project specifications, and the second is conformance to the standard. To comply, the system must fulfill all the requirements in the project specification. Conformance-- to conform to the TMDD standard, the system must fulfill all the mandatory requirements identified by the standard. So, the standard says, "Hey, all these requirements are mandatory, must be supported, must by satisfied-- excuse me, fulfilled-- but also must support the optional project requirements that have been selected for the project and use all the data concepts-- the dialogs, the messages, big data elements-- that are traced to that subject requirement and the requirements traceability matrix, and the design content, in the manner specified by the standard or the reference standard. So, based on what requirements we have selected, the TMDD standard in the requirements traceability matrix says, "Hey, you have to support these design concepts, these dialogs, these messages, these data elements." And it has to be used the way the standard specifies it to be used. So just as a note, conformance is not compliance. Conformance is conformance with the standard. Compliance is the complying to the procurement specs-- the project specifications.

Patrick Chan: In addition, TMDD does reference other normative standards. So when we test the TMDD, we also are testing that the implementation can perform to those reference standards. So for example, TMDD-- the protocols for using TMDD are defined either in NTCIP 2304 or NTCIP 2306. So when we perform testing, we also are testing conformance to one of those standards. We are also testing that the data exchanges occur as defined by the standard. The dialogs, which define the sequence of messages-- so we're testing that the sequence occurs as expected, whether it's a sequence of request response messages or the sequence of subscription of publication messages; that we correctly handle error messages; that the structure of the TMDD messages that are being exchanged is consistent, conforms to the standard as defined by the standards; and the correct data content is being exchanged.

Patrick Chan: So, we've talked about what is being tested, but let's talk about what's not being tested, or at least not being addressed by this module. How the data is used in the implementation's environment: So this module only contains examples related to the interface. The examples do not-- we do not consider how the system is implemented or actually performs its required function, only that it fulfills the interface requirements. That is, does the data that's being exchanged appear as defined by the standard and in the proper sequence? When we talk about testing in this module, we do not consider how the system-- the TMDD software, for example-- what it does with the data or how it uses the data. That's outside the scope of this module. The operations, the implementation is attempting via the interface. So, for example, this test plan does not consider how a device is monitored or controlled or how an operator views event information, or how a plan library is managed. So, a center may request a status from a device, for example, but in this module-- but we do not consider what method, means, or logic is being used by the center to monitor or determine the device status. So in summary, the TMDD testing focuses on compliance with the system interface specification requirements but not the software implementation. We're only concerned about the interface, what goes back and forth between two systems, not how it's being used.

Patrick Chan: The next couple of slides, we're going to show some examples of what might go into a test plan for a TMDD-based system based on IEEE 829. So, for our example test plan, you may have a test plan identifier. You may have an introduction. So, "The purpose of this test plan is to verify the center-to-center interface between Agency X and Agency Y complies with the interface control document version xxx, and verify conformance with TMDD v3.03." Notice we do indicate what software version is being tested. So, we're testing the center-to-center interface as defined in TMDD and we indicate what version of the software is in each-- for each agency.

Patrick Chan: These are some of the features that might be tested within the scope of this test plan. If it's a level test plan, it's possible that only some of these features are included in one test plan, while some of the other features are addressed in other test plans. So, I may have one test plan just for testing the traffic network; I may have another test plan to support testing device control, monitoring and status.

Patrick Chan: So, as we mentioned earlier, the TMDD standard using a systems engineering process that includes a needs to requirements traceability matrix. This NRTM was specifically added to the standard to support the system engineering process and to assist the agencies in properly specifying the use of the standard for an agency-specific project and operational needs. So if you want to read more information about how to complete the NRTM for a procurement spec, please go back to module A321a, User Needs for TMDD. But here we've taken a snapshot of the NRTM. And the NRTM can be used to define the scope of the test plan. In this case, the user needs and requirements tab is circled yes, are part of the project specification. So a test plan may include the entire completed NRTM to define the features that are to be tested in the test plan that really identifies the scope. So by putting in there Completed NRTM, you've pretty much defined the scope of what belongs in your test plan.

Patrick Chan: Approach. There may be different approaches to performing a test. You may test based on user needs or a group of features, or by a phase in the system lifecycle. In this test, the tests are based on the user needs. So they've organized the testing based on-- or we're going to organize the test plan based on what features are in the completed NRTM. So we may organize by, "Oh, let's test the traffic network first," and then we may test the device control and monitoring separately. The approach here is also to try and create a modular test procedure design, so that data concepts that fulfill multiple requirements-- for example, data frame organization information-- can be used, because organization information shows up in many of the messages, many of the dialogs, with TMDD. Other examples of data concepts that fulfill multiple requirements include authentication, restrictions, data/time, device information requests. These are reused over and over again in TMDD. And then the approaches discuss-- you may want to have a paragraph discussing how the results will be logged. Items pass/fail. To pass the test, the item under test shall pass all the test procedures associated with the requirements for the test item identified in the NRTM. So here, we're identifying what the pass/fail criteria is.

Patrick Chan: Suspension criteria and resumption requirements. The test may be suspended between performance of any test procedures. So, based on this, you're not allowed to suspend testing in between test procedures. A test, if you resume, shall always resume at the start of a test procedure, and any modification to the test items may require performing a regression test. So again, a regression test is any test that might be performed to uncover any software bugs after changes have been made. We've tested them once; we want to make sure that when we make modifications that we didn't introduce something new. So you do this by performing a regression test, and it may include rerunning previously completed tests and then comparing the results of the test before and after the changes were made.

Patrick Chan: Test deliverables. So here we define what the test deliverables are. Notice that each specification is provided a unique identifier.

Patrick Chan: Testing tasks. This slide shows some of the possible tasks that might be included in a test plan-- developing the test documentation; training workshop-- so perform training for the necessary skills to review the test documentation and procedures prior to the start of the test; preparing for the test, assembling the test materials, arranging the test environments; conducting the test and generating test logs; preparing test summary report; and then transmitting the test documentation to the project manager.

Patrick Chan: Environmental needs. Test environment describes the test environment, what the test items are, what the test hardware that might be used, what the test software that's going to be used is, what documentation should be available during the performance of the test, what staffing and training needs may be required for each staff.

Patrick Chan: Responsibilities-- what the responsibilities are for each of the participants in the test plan. So, what is a project manager supposed to do? What are his supposed responsibilities? What is a test analyst supposed to do? What's the system integrator going to do?

Patrick Chan: Schedule. So this is an example schedule that might go into a test plan. In addition to this, we may also need to indicate what submittals and when the submittals are due.

Patrick Chan: Risk and contingencies-- what the risks are. One of the risks defined here is we're unable to complete all the test procedures on schedule. So the contingency may be to schedule remaining test procedures. Or another risk is the ATMS software at each TMC are unable to exchange information. So you may want to conduct a preliminary spot check to see if there's any major issues. And then finally, approvals. You want to indicate the names and titles of all persons that must approve the test plan. Not the performance of the test, but to perform the test plan.

Patrick Chan: So, activity, another poll. Which of the following is NOT in the TMDD standard v3.03? So which of the following is NOT in the standard? A needs to requirements traceability matrix; a requirements traceability matrix; requirements to test case traceability matrix; or a single design to fulfill each requirement.

Patrick Chan: So, the correct answer for this was C, requirements to test case traceability matrix. We don't-- again, we don't include test plans, test procedures that type of information currently in the standard. It does include a needs to requirements traceability matrix. It does include a requirements traceability matrix. And the standard does have a single design defined for each requirement. So for each requirement, we define how do we fulfill that specific requirement.

Patrick Chan: So, just to summarize what's in learning objective number three: Described the application of a good test plan to a TMDD-based system being procured using a sample test plan. We've gone over the TMDD standard. It uses the systems engineering process. Defining user needs, requirements to satisfy those user needs, and the design content to fulfill those requirements. We've explained what is and what is not being tested when we test a TMDD-based system. We're testing compliance for project specification, conformance to the standard. We're testing the system interface between two agencies, but we're not necessarily testing the software to see how it operates and how it performs. And finally, we reviewed an example test plan for a TMDD-based system.

Patrick Chan: Learning objective number four. So, over the next couple of slides we're going to identify the process to write a test plan in the context of the requirements of TMDD that have been selected by the user. So, based on what's been selected for the project specification. We're going to use the NRTM to identify the features to be tested. We're going to use the requirements traceability matrix to determine the standard design to verify the requirement, to verify that requirement is fulfilled; and we're going to create a requirements to test case traceability matrix. This matrix will help indicate what test cases must be passed for the requirement to be considered fulfilled, and it's also used to help verify that we've tested all the requirements of a procurement spec at least once.

Patrick Chan: So, again, the NRTM and the requirements traceability matrix in TMDD are key to writing a good test plan to test the TMDD-based system. So, the completed NRTM can be used to define the scope of a test plan. So the complete NRTM or a part of NRTM, complete NRTM, answers the question: What requirements, what features are being tested in this test plan? So this is an NRTM. Another example NRTM can be found in Table 1 of the student supplement.

Patrick Chan: The requirements traceability matrix, on the other hand, defines a single standard design in the form of dialogs, messages, data frames or data elements-- what we call design content-- that must be supported to fulfill the requirement. So go over each carefully. Dialogs are the sequence of data exchanges that are defined by the standard. So, these are the sequence of events that must occur to fulfill the requirement.

Patrick Chan: Looking at the RTM, right now we're going to go over some of the details of what's in the RTM. Again, A321b does go through this in detail, so if you want to review this, I would suggest reviewing module A321b. But just to review quickly here, the first three columns of the RTM, the requirement ID and title are in the first two columns, followed by the dialog pattern. TMDD really consists of three dialog patterns-- request response, where the request message is transmitted from external center followed by response message from the owner center-- so here's a request, and I'm expecting a response message from the owner center-- so that's one pattern. A subscription, where a subscription request is transmitted from the external center, followed by a confirmation message from the owner center that says, "Yes, I received your subscription and I'm confirming you're not subscribed for these messages." And then publication, where the partner message is transmitted from the owner center, followed by a confirmation message from the external center. So the owner center publishes a message and the external center says, "Yes, thank you very much, I received it." So that's 2.41, 2.42 and 2.43.

Patrick Chan: In the next four columns, first the data concept type is shown under the DC type. The data concept types are dialogs, messages, data frames, or data elements. So this defines-- this is the type of design that we expect to see to fulfill that requirement. Sometimes it's going to be a dialog to fulfill a requirement that says, "Hey, I expect this dialog." Sometimes it's going to be a data element that says, "Hey, to fulfill this requirement, I'm expecting this data element." The data type can be found in the definition class name, while the DCID number-- column, excuse me-- indicates the clause, in volume two of the TMDD standard, where the definition for that data type can be found. So if you're looking for more information, this is the section in volume two that defines that data design content. The data concept instance name column defines the instance name of the data concept that fulfills that specific requirement.

Patrick Chan: So, let's go through each row a little bit more carefully. The first requirement is to request response dialog. Design that fulfills the requirement, "Send DMS inventory information upon request." The data type is the dlDMS inventory request, while the definition can be found in 3.1.6.1 in the standard-- volume two of the standard. So, if we looked at 3.1.6.1, it will tell you in more detail what exactly that dialog is about. The second and third requirements are a publication dialog and a subscription dialog, respectively. So the definition class name tells you what the name of the dialog is, and the DCID tells you where in volume two we can look to find out more information about that dialog.

Patrick Chan: Moving down, the fourth and fifth requirements are fulfilled by messages. Meaning those requirements-- content of the DMS inventory request and content of the DMS inventory information-- are fulfilled by messages. The request message is fulfilled by a message called Device Information Request Message, while the DMS inventory information is fulfilled by a message called DMS Inventory Message. The sixth requirement is fulfilled by a data frame called Device Inventory Header, which is always part of the message. The data frame is a bunch of data elements in a specific sequence, so that's what a data frame is. And the last two requirements are fulfilled by the data element DMS sign type and DMS sign technology, respectively. These data elements may be part of a data frame, but they're part of a message. So the RTM is important to test documentation because the RTM indicates the design that must be implemented to fulfill a requirement. The test documentation, when you're writing it, should reflect the test of the design indicated. A more complete RTM for this example can be found in Table 2 of your student supplement.

Patrick Chan: Let's take a look at what's in the design detail definition in the standard. So, below is the dialog the fulfills 3.1.6.1, which is the dialog DMS inventory request. So, moving down a little, you can see that there's a definition. The definition of this dialog is a request response dialog that allows an external center to request an owner center to provide an inventory of the owner's dynamic message sign. The descriptive name, the second line, indicates that this dialog is between an external center and an owner center. Moving further down, reference messages-- under reference messages, notice that there's an input message. The input message is the device information request message. So this is what's being transmitted from the external center. And the output message, the inventory message, is what's being transmitted by the owner center. And there's also a definition of a fault message. So if the owner center does not recognize, understand what the external center is requesting, the owner center will then transmit back an error report message. You can't see it here, but within a standard-- the PDF of the standard-- it's hyperlinked, so if I were to click on TMDD Messages 20 on the reference messages, it will jump to the definition of device information request message. TMDD Messages 20, by the way, is an object identifier, a unique ID that's been assigned to every data concept, design content in the standard.

Patrick Chan: So, looking at it graphically, this is a sequence diagram that shows the sequence of data exchanges for this dialog of dlDMS Inventory Request, the request response message. The first line on the right shows that the device information request message is being transmitted from the external center to the owner center. In return, the owner center shall transmit a DMS inventory message back to the external center. The second half of the sequence diagram, bottom half, shows the behavior in case that there's an error or something's not recognized by the owner center when it receives the device information request message. In that event, the owner center would transmit back an error report message back to the external center.

Patrick Chan: Requirements to test case traceability matrix. The next percent, the next step in creating a test plan is to create a requirements to test case traceability matrix. This matrix is not in the standard, but what it does is it traces each requirement to a test case that verifies that the implementation fulfills the requirement. It's used to verify that the test cases capture testing of all the requirements at least once. So this matrix will define, "Okay, if I want to test that this requirement is fulfilled, this matrix will indicate what test cases I have to perform and pass to confirm that the test case-- I'm sorry-- that the requirement has been fulfilled." The test matrix contains all the requirements that should be tested. By doing this, we verify that all the requirements have been tested by the test plan. So, sometimes one or more test cases may be needed to completely test the requirement. The reason why is that, one, they might be different sets of values. For example, there's more than one type of event that's supported by the TMDD standard. There’s a current event, planned events, or forecasted events. So you want to make sure you want to test all three instances, if selected, all different values of the device-- that all three types of events are tested and confirmed and are fulfilled. You may also, for example, verify the different statuses of a device. It might be online/offline. You want to confirm that the implementation supports both statuses. There also might be different conditions-- for example, that no errors are detected, or if an error was detected. We showed in the dialog before, the previous dialog, this is what happens, this is the message the owner center will transmit if there's no error, but we also showed the condition of, well, if there's an error, that the owner center should transmit an error report message. So we want to confirm that the system being implemented supports both types. You may also want to perform what's called boundary or negative testing, which means-- especially if public safety might be compromised with a bad value. So for example, the center says it supports value one, two, three, four, five, you may want-- if it's important, if it's critical, you may want to send a test value of six just to see what happens. It should fail. Nothing should happen in that particular case, but you want to confirm that. So each value, each condition, may need to be tested to fully verify a requirement.

Patrick Chan: This is an example of a requirements to test case traceability matrix. So, the requirement-- the first requirement in this matrix is 3.3.2.1, which is Send Organization Information Upon Request. And we have two test cases, TCS 1.2.1 and TCS 1.2.2. Each is important because one verifies that the system handles it properly and there’s no error; and the second handles it, verifies that the system handles it in the case that there is an error. So to fulfill this requirement, both test cases must be passed and has been completed successfully. A more complete RTCTM, requirements to test case traceability matrix, can be found in Table 3 of the student supplement.

Patrick Chan: So, reached another poll.

Patrick Chan: Which of the following is part of the Requirements to Test Case Traceability Matrix? Your possible answers are A, user needs; B, requirements; C, design; and D, test plans. Again, which of the recognize is part of the Requirements to Test Case Traceability Matrix? I'm just going to pause here for a drink.

Patrick Chan: So let's review the answers. The correct answer actually is B, requirements. Requirements are part of the requirements traceability matrix. The user needs are not. They only appear in the Needs to Requirements Traceability Matrix in the standard. Design is not. Design only appears in Requirements Traceability Matrix. And test plans are not. Test cases are part of the Requirements to Test Case Traceability Matrix. As a note though, a test plan can contain a requirements to test case traceability matrix, but.

Patrick Chan: So just to summarize learning objective number four, we've used the NRTM to identify the features to be tested in the test plan. We can also use the requirements traceability matrix to determine the standard design to verify requirements. So you say, "Hey, this is how the standard defined-- this is how we fulfill what we test to fulfill requirement." And to create a requirements to test case traceability matrix, which indicates the test cases that must be passed for the requirement to be considered fulfilled, and verifies that the test cases capture testing of all requirements at least once.

Patrick Chan: Moving to learning objective number five: Analyze to ensure conformance with the TMDD v3.03 standard. So for the next couple of slides, we're going to review what it means to conform to TMDD, specifically version 3.03, and introduce how to add extensions to support functions that are not supported by the current TMDD standard.

Patrick Chan: Interoperability is the ability for different components or, for the purpose of this module, different TMDD-based implementations from different vendors to exchange information and to use the information that's been exchanged. Interoperability is one of the key objectives for developing and for using the standards. When we achieve interoperability, we reduce cost. Reduce risk, and by extension we reduce cost for the implementers. So, what do we mean by interoperability? So, let's use an example. Along with other standards-- let's say HTTP, TCP/IP-- Wi-Fi is an example of interoperability. Together, it doesn't matter what hotspot you might be in-- you might be at Starbucks, you might be at the airport, you might be at home-- or what laptop you're using-- it might be Dell or HP-- when we conform to the group of standards, including the Wi-Fi standard, the internet standard, the HTTP and other standards, users will be able to connect to the internet and get their email. So this is the example of interoperability. The idea is similar for TMDD. If the centers-- if different centers agree to and use the same standard, including TMDD, it doesn't matter what center you're trying to share information with or whose vendor you're using; you should be able to exchange that information that the standard supports. So that's the goal of a TMDD system, for using a TMDD standard. So TMDD supports interoperability by defining a single design, a standard design-- again, in the form of dialogs, messages, data frames and data elements-- to fulfill each requirement supported by the standard. It defines the sequence of events or actions, and the data that must be exchanged. All systems shall fulfill the requirement the same way.

So that's the goal. That's what TMDD defines. And when you conform to the TMDD standard, you are able to achieve interoperability.

Patrick Chan: To claim conformance to the TMDD standard, an implementation shall satisfy all the user needs defined as mandatory in the NRTM and all user needs defined as optional in the NRTM but were selected to be supported for the implementation. So this statement is from the TMDD standard. This first paragraph defines the rules for which implementation can claim conformance to the standard. So to conform to the standard, implementation much satisfy all the mandatory user needs defined in the standard, and any user needs that have been selected for your project, for your implementation. TMDD allows implementation to claim conformance to a specific user need in a standard. So the second paragraph allows an implementation to claim conformance to specific user needs. So to conform to a specific user need, the implementation shall fulfill all requirements that trace to the user needs identified as mandatory in the NRTM and all the requirements that trace to the user needs identified as optional in the NRTM but were selected to be supported for the implementation. So you have to support all the mandatory stuff, but if you selected something that the standard defines as optional but you select it for your implementation, you have to support those requirements also in the way the standard defines it as.

Patrick Chan: To claim conformance to requirement in the standard-- so, TMDD also allows an implementation to claim conformance to a specific requirement. To conform to a specific requirement, the implementation shall fulfill the requirement by using all the data concepts-- whether it's a dialog, message, data frame, or data element-- traced to that requirement in the RTM in the manner specified by the standard, or the reference standard.

Patrick Chan: TMDD allows for extensions to support operational or user needs not supported by the standard. The benefit of this is it allows an implementation to add functions that are not supported by the standard. When we developed the standard, we recognized-- we supported all the user needs that we believe a lot of agencies would require needed, but we recognize there are some user needs that we couldn't support. So recognize it, or we didn't want to work on it-- define how that user need is included in the standard. So, we recognized that sometimes extensions are needed. So, we allow the implementation to add-- to support user needs, to extent the standards to support user needs that haven’t been defined by the standard. But note, by definition, implementations that add extensions to the implementation are no longer conformant to the standard. So extensions of designs are not recognized by the standard and thus are considered to be non-conformant, although the standard allows its use. The reason why is the use of standards causes interoperability problems, the very issue that standards were intended to address and avoid. With extensions, interoperability may be compromised. Other centers must support extensions in a consistent manner to support interoperability. So, test documentation needs to be expanded to support extensions. So let's say your region agrees on the extension. We have a user need. We need to-- that's not supported by a standard. You may agree on the extension, but it should be designed and documented very carefully and very consistently for procurement specification and for all subsequent procurement specifications that uses that extension. Otherwise, interoperability between the centers in your region is likely unachievable for the function that the extension was attempting to address. The problem is big within an agency when we add an extension to continue supporting interoperability, but exponentially bigger, the function, and the extensions are needed between multiple agencies. So we recognize agency extensions are needed. Interoperability may be compromised, but if you do add an extension, do it in a consistent manner and do it very carefully.

Patrick Chan: So, to assist with the extensions, TMDD, the standard, defines some rules for adding extensions so that any extension is still consistent with TMDD. Note again, consistency is not conformance. When we add extensions, you're no longer, by definition, conform-- you no longer conform to the standard. But consistency is a principle used when you extend a standard and helps you document and build consistency with the rest of the conformant features that are being implemented. To be consistent with TMDD, the following statements indicate how the implementation-- the following statements state that the implementation must adhere to the following: One, each requirement must be fulfilled by the design identified by the standard. No other standard design, such as proprietary design, is allowed to fulfill the same requirement. So for example, if the standard says that the allowable value is A, B, and C, a vendor cannot come along and change the allowable values and change it to 1, 2, and 3. So, if the functional requirement is already supported by a standard, it must be implemented as defined by the standard. Different interpretations on the meaning of a data concept, or how it is to be used, requires a new data concept. So the data concept cannot be used in any other manner than as defined by the standard. So for example, TMDD has data elements containing vehicle bins for vehicle classification by vehicle length. So you cannot reuse those vehicle lengths-- excuse me, those vehicle bins-- to support vehicle classification by, say, vehicle type. It's intended-- those vehicle classification bins are only intended for vehicle classification by vehicle length. A conformant center receiving a message must ignore any attributes or elements in the message that it does not recognize, but shall process what it does understand. So a center might receive additional information such as additional data elements in a message. This statement says, well, the receiving center shall process the portions of the message it does recognize. So it might recognize the first six data elements, for example, but if it receives a seventh data element it doesn't recognize it, it shall ignore it. Reasons why a center may not recognize that sent data element, it may not support that specific requirement in its implementation, but the sending center does support it. For example, authentication. I don't support it-- my center doesn't support authentication, and I suddenly receive it, I'm allowed to ignore it.

Or perhaps the sending center is transmitting an extension or value that the receiving center does not support. So the sending center, transmitting center, might send an extension, and I don't support that extension, but it's okay; if I see the extension, I just simply ignore it. So that way by having this rule, this statement, the-- your center can continue using TMDD, the parts that it does recognize, and ignore the parts that it does not support, or can ignore extensions that another center might have added.

Patrick Chan: Continuing, the rules for extension. The new data elements may be added but cannot reuse an existing data element name. So I have a data element called "vehicle type" and you want to create a new data element, you cannot use vehicle type. TMDD standard already uses-- you have to create something else, a different name. So it might be data-- vehicle type extension, for example. New enumerations may be added in a newly created data element but cannot reuse an existing data element name. So, for example, we have a data element that supports five values and you want to create a sixth value. You may create a new data element containing that sixth value. You cannot reuse the current data element that's defined by the standard. You have to create a new data element, and that new data element will contain the extension. Extending the range of an existing data element requires that the data element be renamed. So you may have a data element called A that supports the value from zero to ten, but you want to extend it to zero to sixteen. You have to create a new data element called B that has the extension zero to sixteen. And new messages may be added beyond those messages defined by the standard but cannot reuse an existing message name. Same thing. I have a message called DMS Inventory Message. You want to add an extension. You may create a new message, but you have to give it a different name. You cannot use DMS Inventory Message.

Patrick Chan: Next, dialogs contained in the standard may not be modified. However, new dialogs may be created to support extensions. So, using a previous example, I created a new message called DMS Inventory Message Extension. You cannot reuse the existing dialog, which was dlDMS Inventory Message-- or whatever it was. You have to create your own new dialog that uses the new message. All design extensions shall be documented in a separate XML schema. One of the changes that was made to the current version of TMDD, TMDD v3.03, was we have now frozen the XML schema, or the ASN module, ASN.1 module. You are not allowed to change the schema anymore. So if you want to add a design extension, you have to document your extensions, whether they be dialogs, messages, data frames or data elements, in a separate XML schema, or a separate ASN.1 module. And the XML schema supplied by-- that's been copyrighted by AASHTO does have a way to document your extensions in a separate schema or separate module. All extensions shall be documented in a manner consistent with the presentation in the standard, and shall include the user needs being addressed and the requirements being fulfilled and the traceability tables. So whenever you add an extension, you have to document it, just like the standard documentation. Meaning you should add the extensions and the needs to requirements traceability matrix indicating what the user needs were, what the new requirements were, documented in the RTM these other requirements that we've added, and this is the design, the messages, the dialogs that have been added to the-- that we've extended, that we've added in our implementation. So the user needs have to be spelled out, documented. Requirements have to be documented, and the traceability tables have to be updated.

Patrick Chan: So, just what I said. An RTM should be updated to include any user needs, any new requirements. RTM should be updated to include the new data concepts that fulfill each requirement. And for testing purposes, the requirements to test case traceability matrix should be updated to reflect the new requirements that have to be tested.

Patrick Chan: Next is another activity, another poll.

Patrick Chan: Which of the following is permitted by the TMDD standard? A, create a new message to fulfill a new requirement; B, change the meaning of an existing data element; C, create a new data element using an existing data element name; or D, modify an existing dialog. Again, which of the following is permitted by the TMDD standard?

Patrick Chan: So the correct answer is A, create a new message to fulfill a new requirement. So TMDD allows you to create a new message to fulfill a new requirement. It does allow you to change the meaning of an existing data element; that's not permitted. It does not allow you to reuse the name of an existing data concept for a new data concept that you may create. And it does not allow you to modify an existing dialog. It expects you to create a new dialog to support any new messages that you may wish to transmit.

Patrick Chan: So, just to summarize learning objective number five, we reviewed what the key elements are of the conformance statements that's in TMDD v3.03, and we have discussed about adding extensions. You are permitted to add extensions to the standard. They are discouraged because they break interoperability, and when you add extensions, implementations that add extensions are no longer considered conformant to that standard.

Patrick Chan: So, learning objective number six. In the next couple of slides, we're going to review a little bit more in detail some of the information and elements that comprise the test documentation that's part of a test plan, how we're going to use an RTM and the RTM to create good test documentation, and in which test documentation these matrices should appear. And we'll also introduce the reference implementation and some tools that can be used for testing.

Patrick Chan: Test documentation. So, test documentation can be written after the user needs or requirements are finalized. So for example, after the NRTM has been completed, the concept of operations, and the requirements are completed and finalized, you can start writing your test plan.

Patrick Chan: So, the next couple of slides we'll kind of talk about what is being tested in this test plan. So this table depicts the different parts of the TMDD standards and how they relate to test documentation. So for example, again, once the NRTM, which defines the user needs is finalized, you really can begin the development of the test plan. The test plan can be used to validate the system, answering if you've properly addressed multi-user needs of the system. Once the RTM, the requirements traceability matrix, is finalized, which defines the requirements of the system, the development of the test design specifications can begin. The test design specifications define the feature or combination of features to be tested, and identifies the associated tests for that feature or features. Once the design to fulfill the requirements is defined, the development of the test case specifications and test procedure specifications can begin. For the TMDD standard, the design is actually already defined by the TMDD standard. And during the implementation phase, which includes the testing phase, it's a development of the TMDD-based system, the test execution and results of the test executions in the forms of test reports will depend on the implementation.

Patrick Chan: This is a graphic that shows the relationship between the different test documents. These test documents can be integrated into sections of a few documents, depending on the complexity of the project. So, for example, the test case specification and test procedure specifications can be combined into one document. They don't have to be separate documents, depending on how complex the system is. So again, for a simple system, what belongs in a test plan and test design specification might be one document, and a test case specification and test procedure specifications might be combined into a separate document. Next, we have test execution, which is actually performing the test. And at the end, we have the test reports-- test logs, test incident reports, and summary report of the test plan, of the execution of the test plan.

Patrick Chan: Test documentation development in the initial phase, when you first create the test documentation, it might be an iterative process. The development of test documentation may occur concurrently because changes to any part of the test documentation might impact another test documentation. So, developers may have to work on several documents concurrently to maintain consistency. So, for example, if I'm updating the test case specifications and I change something, I may also have to change the test design specification.

Patrick Chan: Let's quickly review the test design specifications in a little bit more detail. The test design specifications identify the features to be covered by the design and its associated tests; identifies the test cases and test procedures required to accomplish the testing; satisfies the pass/fail criteria. You may have separate test design specifications for each phase of the lifecycle. So for example, one test design might focus on testing the requirements during the prototype test, while another test design specification may focus on testing the same requirements during the unit or device tests. You may also have different test design specifications by functional areas. So for example, you may have one test design for sharing events information or incident information, and a second test design for sharing device information.

Patrick Chan: This is an example of what a test design specification may look like. On top, we indicate the requirements that are being tested or verified as part of the test design specification. Notice on top we also have a unique identifier and a title for this test design specification. Approach refinement: This section may include notes on the approach, such as test script configuration or a variable table configuration, or what the failure criteria is. Next, moving down, the third and fourth columns here-- well, on top, in yellow, we have the requirement IDs and the title that this test design-- what features, what requirements are being tested. The third and fourth columns indicate what test cases should be performed to fulfill-- to verify that a requirement is being fulfilled. The implementation, again, must pass both test cases to completely fulfill that requirement. In the bottom, the section on pass/fail criteria, indicates the criteria for passing the test design specification. Note that in this example, the test design specification does not map to test procedure specification. That mapping can be provided in the test design specification or in the test case specification. A more complete test design specification can be found in Table 4 of the student supplement.

Patrick Chan: Test case specification. The next couple of slides provide an overview of the test case specifications using a TMDD example. So, a test case specification specifies the inputs, predicts the results, and a set of execution conditions for a test item. Recall that the requirements traceability matrix defines the dialogs and data concepts that must be supported to fulfill a requirement. So, the test case specification answers the question: How is that requirement fulfilled? Also recall that the requirements to test case traceability matrix defines the test cases that must be passed to verify a requirement is fulfilled. So the test case, or test cases, should confirm that the system performs the same sequence of data exchanges or events and uses the same data concepts to fulfill the requirement that's being verified.

Patrick Chan: So, here's an example of what a test case specification may look like. Again, on top, we have the unique identifier and title of the test case specification. Next is the objective, or the purpose of this test case: What is that this test case is testing, or verifying? Next we have the test items, which specify the items or features being tested by this test case. In this case, it's the requirement IDs and the requirement titles. It specifies the source the inputs required to execute a test case. For example, where am I getting the test data to be used for the request messages? It may be a random generator, random value generator. It might be written down or, in this case, we created a test case input table to indicate what inputs are going to be used. Test output specification-- that's where we document what the expected test outputs are from performing the test case. The actual output should be compared to those expected outputs, with allowable tolerances as appropriate. Next, we have environmental needs, which that specifies the characteristics and configurations of the hardware, the system, and the application software, or any unique facility needs that might be required to execute this test case. Special procedures might specify any special constraints or dependencies that must be considered or executed prior to or during the test case, such as any special setups that may have to be performed. Like, for example, let's set up the test controller to be in this mode, or any operator interventions. For example, execute several steps to set up the system and data to a known operating condition prior to or after the performance of the test case. Indicate dependencies; identify any test cases that must be executed prior to this test case. For example, in this example, we want to make sure we set up the subscription for the DMS inventory prior to running this test case specification. Another example of a test case specification can be found in Table 5 of the student supplement.

Patrick Chan: We mentioned that sometimes the inputs for a test case might be in a separate specification, separate document. So this is an example of one that has the inputs for the test cases. On top is the unique identifier and title of the input specification. Note that IEEE 829 does not define what goes into an input specification. In this example, the requirements that are addressed by this test case specification are listed and are as sorted in the requirements traceability matrix. The first column is the requirement ID by the data concept type. The next two columns are the instance names for that data concept, the section in the TMDD standard where the design concept is defined. So in case we have any questions, we can look it up; we know where to look it up in the standard. Followed by the if applicable, the variable, or the value that we use during the execution of the test case. So for example, in the second to last row shown on the table, it's the device information type, which is a numeric value. And for this case, because we're testing DMS, the enumeration for this test case shall be device inventory, or number one. And above that, it's device type and, again, it's an enumeration, and the device type should be three, which represents a dynamic message sign.

Patrick Chan: This is an output specification. It's very similar to the input specification, but it indicates what the expected outputs and results are. Again, we have the ID and title of the output specification on top. We have the requirements ID, the data design concept type that fulfills that requirement in the second column; the instance name; the ID with more information about that design concept can be found; and the last column, the value domain, we show what the expected value is going to be. So, for example, in the last row, one of the messages-- when we send an inventory request, we get back a DMS inventory message, and one of those values that comes back from the owner center is the DMS sign type, and we may be expecting the sign type of 5 vmsLine, which is the enumerated value of five. So this is a line matrix VMS that we're expecting back.

Patrick Chan: Test procedure specification. So, the next couple of slides provide an overview of the test procedure specification. The test procedure specification indicates the sequence of actions for the execution of a test. It is important not to skip any steps in the test procedures to ensure proper conformance testing. We try and set up the test procedures so that they can be reused. So for example, again, the authentication data frame appears in every request message, TMDD v3.03. Rather than retype the test procedures for authentication for every request message or dialog, test procedures or test case, to just call the test case for authentication test procedure. Other examples of data frames or data concepts that appear in every request message include error reports, the organization of information, and device information request message. So we try and reuse these test procedure specifications wherever we can to save on cost.

Patrick Chan: Looking at an example of test procedure specification, on top we have the unique ID and title of the text procedure. Next is the scope or the purpose of this test procedure: What is it that this test procedure is verifying? Special requirements: Are there any prerequisite procedures or special environments requirement? For example, if we're testing a device queue, what's in the device queue, we want to make sure that there are no other existing queued requests, or nothing else is in the queue that we are not aware of before beginning the test. And also to identify any preconditions. Next are the steps to be executed. Notice that we number each step with a unique ID. The steps may include actions for preparing for the execution of the test, actions for the execution such as preceding something, logging, measuring, or stopping the execution. The results column allows test personnel to indicate if during the execution of the test if the procedure step was passed or failed successfully or not. And the reference column-- so in case we wanted to verify what the steps are or why we're performing that step, why we're performing a specific step, we can refer to a standard or to another test specification, another test document, to find out more information about that step. More complete test procedure specification can be found in Table 6 of the student supplement.

Patrick Chan: So, we reached our last poll. In this example, we're going to look at the definition for a dialog. In this case, the dialog is "Dialog Video Switch Status Update." Notice that this dialog has a precondition on top-- that's what's circled in red. "An owner center shall provide updates to an external center upon acceptance of a DL Device Information Subscription Dialog." So a precondition to this dialog is that the owner center has approved an external center's request to subscribe to the information. So the subscription has been verified. Notice that at the bottom, the input message is the video switch status message for this dialog, meaning that the owner center will transmit the video switch status message to the external center, and in return the external center will send back a C2C Message Received back to the owner center. In case there's an error, the external center is supposed to send back an error report message.

Patrick Chan: So, based on that particular description, and this should also be in your student supplement, in the back, let's answer the question: Which of the following is not an appropriate test step for the previous dialog? The owner center sends the video switch status message to the external center; precondition to execute the dialog device information subscription dialog; the external center sends a center-to-center message received message to the owner center; or the external center sends a device information request message to the owner center. Again, which of the following is not an appropriate test step for the device dialog? And again, the definition of this dialog can also be found in your student supplement.

Patrick Chan: So let's review the answer. The correct answer was the external center sends a device information request message to the owner center. That is not what the external center is supposed to send, according to the definition of that dialog. The external center is supposed to send a C2C message received message to the owner center. If you look at the definition of dialog, the video switch status message is the input message. There is a precondition to execute the dialog device information subscription dialog. And the external center is supposed to send a C2C message received message back to the owner center. So the correct answer was D.

Patrick Chan: Continuing, test log. Up to this point, we've been talking about the documentation prior to and during the execution of the test documentation. Now we're going to go over what is the documentation that's going to be produced when we perform-- when we execute the test. So the first test documentation is the test log, which provides a chronological record of the relevant details about the execution of the test. It should identify the testers, identify what items are being tested, including version levels, what version of the software am I using, for example, and the test environment. Might include or identify the test procedures being executed; records the results; indicates if the test has been successfully passed or not. May include a copy of the test procedure specifications for approval signature, and it might also include printer output, screen snapshots, etcetera-- anything to verify the test results, whether it was a good test result, expected test result, or not.

Patrick Chan: Another output might be a test incident report. It documents any event that occurs during testing that may require additional investigation. So it might include a summary of the incident, what the inputs were, what the expected results were, what the actual results were, what anomalies were observed, when the anomaly was observed, and during what test step being executed did we observe the anomaly and who the observers were. The incident report may also indicate the impact that the incident will have on the test plan, other test documentation.

Patrick Chan: Test summary report. So, this summarizes the results of all the testing activities and provides an evaluation based on the test items and based on the results of the test items. It notes any variance of test items from the test design specifications or the standards, the test plan, test designs, or test procedures. And it indicates all the persons who have approved this report, this test summary report.

Patrick Chan: What we're going to do now is discuss one of the tools that will be available for testing a TMDD-based interface. As you may be aware, the TMDD consists of a whole bunch of user needs and over one thousand requirements. So as you can see, development test plans, test cases, test procedures to test implementation of TMDD is an enormous effort. Recognizing that the USDOT has developed something called a reference implementation tool to aid in performing the testing on center-to-center interfaces. The reference implementation allows users to create a test configuration based on selected user needs and requirements. So, this test configuration is based on your needs, your requirements that have been selected. It will execute those tests. It will create test reports.

Patrick Chan: The reference implementation follows the system engineering process in the IEEE 829 standard, so it includes the key elements of the TMDD standard relevant to the testing, including the needs to requirements traceability matrix, the requirements traceability matrix, and helps create a requirements test case traceability matrix. It also includes test documentation such as test design specifications, test case specifications, and test procedure specifications, and outputs test reports and test logs. There is a catch with the reference implementation; it only supports the XML implementation of TMDD standard. It was not designed for ASN.1 implementation.

Patrick Chan: The reference implementation also allows a user to create custom test configurations to define the requirements for the center-to-center interface. So, the reference implementation allows you to specify, "Hey, which information layer standard do you want to use, do you want to test for? What application layer standard?" and indicate if the reference implementation should act as the owner center or as an external center. It allows the user to identify a system under test, the IP addresses, location of the web services, username and password. So it allows you to configure the reference implementation based on what your particular needs, what your particular implementation uses.

Patrick Chan: Configuring the information layer parameters, which really is TMDD. Select the information layer user needs to be tested. So right now, it will be only version 3.03 of the TMDD, but it may be used for further future versions of the TMDD standard. Based on the user need highlighted, select the requirements to be tested. So you can select which requirements you want to test for your particular test plan. It does show the NRTM that's in the standard, indicating any predicates and if the requirement was mandatory or optional. All mandatory user needs and requirements are preselected. So that's automatically part-- mandatory user needs and requirements are automatically selected by the reference implementation. Any additional specification in the NRTM can be filled in. So if you look at the NRTM, there is a column called Additional Specifications where you can enter values. Usually they're related to performance requirements, but the reference implementation will allow you to fill in those same-- fill in the blanks, so to speak, for those additional specifications.

Patrick Chan: The reference implementation provides a list of test cases that are applicable based on the user needs and requirements that have been selected in your test configuration files. So you can select which test cases to be executed, including selection of user-defined test cases. So it will allow you to create your own test cases if you desire or in case, for example, you have an extension. The reference implementation will then execute those tests and provide the results of each test case and each test procedure step on the screen. It will also provide the test reports, including conformance compliance reports for each user need and requirement selected. It will highlight any errors that were encountered during the execution of the test.

Patrick Chan: An official test suite will be provided. It includes TMDD standard WSDL schema definitions; the NRTM as defined in TMDD v3.03; requirements to test case traceability matrix; test case definition files identifying the data parameters and expected results of each test case; and test procedures scripts. It allows you to create your own user-defined test suite to support user-defined user needs and requirements, and supports additional test cases and test procedures if you so desire.

Patrick Chan: So, in summary, the reference implementation tool verifies compliance with specification and conformance with the standards, specifically TMDD v3.03 and

NTCIP 2306 v1.69. It will not verify how the data is used in the implementation's environment. So all it does, again, is verify the data that's being exchanged and the sequence of those data exchanges. What does your software-- what your implementation software does with that data is outside the scope of the reference implementation. And it does not verify the operations the implementation is attempting to support via the interface. It cannot validate the system. So again, it doesn't answer the question: Did you build the right system? Did you satisfy the user needs?

Patrick Chan: Other test tools that might be considered is XML Schema Validator. So, it can validate your XML document, especially for the extensions against the TMDD schema-- excuse me, I didn't mean to say extensions before-- so it can validate your XML document against the scheme, which verifies that the XML structure is well-formed, verifies structure of the data is correct, and verifies that the data content is correct.

Patrick Chan: So in summary, we've gone through learning objective number six. We reviewed the elements that comprise these types of test documentation as part of the test plan. And we've also introduced reference implementation to you and what the reference implementation can do.

Patrick Chan: So now we're at the end, so let's just quickly review what we've learned in this module. We tested TMDD-based interface to verify it fulfills the system requirements and to validate if it satisfies the user needs of the system. The test plan for a TMDD-based system interface defines what portion of the system interface is to be tested. A completed NRTM and RTM are key elements of the TMDD standard that should be used to develop the test plan. And a requirements to test case traceability matrix should be created to ensure all requirements are tested at least once.

Patrick Chan: The TMDD standard supports interoperability by defining a single design to fulfill each requirement. Extensions are allowed by the TMDD standard but are discouraged because they can impact interoperability. And finally, the reference implementation is a tool to aid in performing testing of the center-to-center interfaces.

Patrick Chan: What we have listed here are some resources, for any resources for learning about testing TMDD. There is the student supplement that is part of this module. There's also some websites that might provide you more information. The most important one is probably the second one, www.ite.org /standards/tmdd. That's where participants can find the most updated, recent information about the standard and its status. So we recommend definitely looking at that. There's also a TMDD guide which kind of describes a little bit more information-- provides a little bit more information about the TMDD standard. The NTCIP guide, which provides a little bit more information about the NTCIP framework that the TMDD standard uses. And NTCIP 8007, which talks a little bit more about testing. It's geared more for center-to-field testing, but there are some concepts there that are applicable to center-to-center testing.

Patrick Chan: So, any questions? Some of the most frequently asked questions are: Where can I find out more information about the center-to-center reference implementation? At this time, when this recording was done, there was no website or contact information to find out more information about the reference implementation. However, this is a list server on the ITE website-- again, that's ite.org/standards/tmdd-- where participants and interested parties may ask and find out information about the TMDD standard. Any information about the reference implementation, such as when the reference implementation will be available, and the website address will be announced on the list server, but the reference implementation is right now going through-- it's going through acceptance testing soon, so we expect it to be available in early 2014. To join the list server, send an email to the ITE standards coordinator. At this time, it's snarla@ite.org. Another common question is: I am using an older version of TMDD. How do I upgrade? Older versions of TMDD, meaning prior to v3.0, did not have formal dialogs. Formal dialogs-- again, the sequence of events, sequence of actions or messages-- first showed up in v3. To upgrade to TMDD v3.03, it is recommended that the project develop a specification based on the TMDD v3 NRTM and RTM to identify the features to be supported in the upgrade and what the end product is. So for example, by performing this step, you'll be able to identify which dialogs you need to support and to implement. It can also lay out what messages and other data concepts they need to support, and allows the implementer to compare the old messages side by side with the new one and map out where the old data will have to end up in the new TMDD schema. These steps will help the implementer to determine what changes need to be made to the system software. And then reference implementation would be the tool for testing the upgrade. Another common question is: Where do I find support for testing my TMDD implementation? Again, the list server for the TMDD community is a good starting point. The developers, the maintainers, the TMDD steering committee and other interested parties are all on the list server and can help to answer many of your questions. So thank you very much for joining. Hope this has been helpful. And this concludes the module. Thank you.

End of 2013_12_09_13.16_T321_ Final_Recording.wmv