Module 8 - A203 - Writing Requirements When ITS Standards Do Not Have SEP Content Webinar Transcript

Shelley Row
ITS Standards can make your life easier. Your procurements will go more smoothly and you'll encourage competition, but only if you know how to write them into your specifications and test for them. This module is one in a series that covers practical applications for acquiring and testing standards-based ITS systems. I am Shelley Row the director of the ITS Joint Program Office for USDOT and I want to welcome you to our newly redesigned ITS standards training program, of which this module is a part. We are pleased to be working with our partner, the Institute of Transportation Engineers, to deliver this new approach to training that combines web based modules with instructor interaction to bring the latest in ITS learning to busy professionals like yourself. This combined approach allows interested professionals to schedule training at your convenience, without the need to travel. After you complete this training, we hope that you will tell colleagues and customers about the latest ITS standards and encourage them to take advantage of the archived version of the webinars. ITS Standards training is one of the first offerings of our updated Professional Capacity Training Program. Through the PCB program we prepare professionals to adopt proven and emerging ITS technologies that will make surface transportation safer, smarter and greener which improves livability for us all. You can find information on additional modules and training programs on our web site www.pcb.its.dot.gov. Please help us make even more improvements to our training modules through the evaluation process. We look forward to hearing your comments. Thank you again for participating and we hope you find this module to be helpful.

Nicola Tavares
This module is A203, Writing Requirements When ITS Standards Do Not Have SEP Content. This course is designed for decision makers, project managers and operational stakeholders. Your instructor is Ralph Boaz. He is President of Pillar Consulting. He is an engineering management and transportation consultant with 29 years of experience. He is heavily involved in the development of standards for the transportation industry. Areas of expertise include project management, system engineering, process design and implementation, concurrent development, software development and education.

Ralph Boaz
Hello, and welcome everyone. It is great to be together today, and an opportunity for everyone to learn. I'm sure there are some of you that are here, that are getting reinforcement of things you know, or reinforcement of practices you've done, that have been successful, but actually never had somebody tell you that that was the right thing to do, and maybe this will have some new material for many of you. It's my pleasure to be your instructor. I want to emphasize that, in this course, that we're not going to be writing requirements for a standard. And what I'm trying to say is the focus is not about writing requirements for the standard, but writing requirements for the system and interface that you are trying to produce using a standard that does not contain systems engineering process information.

Here is the curriculum path that we called for the non-SEP, Systems Engineering Process courses. You'll see that by now you should have taken A202, which is, Identifying And Writing

User Needs When ITS Standards Do Not Have SEP Content, and also A103, Introduction to ITS Standards Requirements Development. I recommend all the courses in this program. They are terrific, and we have some terrific instructors. Now, as I had said, there are some standards that you've already learned, I should say, that there are some standards that have the systems engineering process information, such as the transportation sensor system standard, or the dynamic message sign standard, and other standards that we have in the program don't have this information, such as the actuated signal control standard, or the closed circuit television standard. And we're going to actually talk about those, and use examples from those today, and we're focusing on those standards that do not have the SEP information. If you'd like to see a table of ITS communication standards that do and do not have SEP information, you can see that in the supplement.

This slide is just a word version of the previous slide, and you can look at that at your leisure. Now, by now you will have probably some or all of this background. You'll have intelligent-- you'll know about intelligent transportation systems, you'll maybe have had some ITS deployment projects. Maybe you have not, and this is why you're taking the class. You may have backgrounds in the government procurement processes. Hopefully by now, you'll see the benefits of standards and you are somewhat familiar with the systems engineering process.

Now here are the learning objectives for our course. At the completion of the course, you'll be able to understand that requirements development is a process. It's not a simple thing, or limited to a single activity. We'll learn to avoid pitfalls when writing requirements, we're going to be aware of the common mistakes and when we actually write requirements. Thirdly, we'll write requirements when an ITS communication standard does not have SEP information. This part will get a bit technical, but from the looks of our class, we have an advanced class, so I'm sure we'll do well with that, and finally, using traceability matrices, as tools for requirements development, we'll actually be adding to what we learned back in Module A103.

So in the emphasis in our course here, we'll build on what we've learned about systems and user needs and requirements of the previous modules, but we'll also provide you with the additional skills to develop requirements for interoperable systems. And then we'll also teach you how to use the ITS standards that do not contain systems engineering information, and in A202, you already got a great introduction to this, and as the prerequisite, and we'll build on what you learned there. We're going to talk about the systems engineering process as applied to ITS communication standards and what's going on is, as these standards are being developed, or revised, we're adding this content to the standard, and what this does is help us create complete and correct standards and it's been advanced by the USDOT. By putting this information in the standards, it helps us ensure the standard is complete and correct.

I know that, from my own personal experience, in working in the standards community, before we started doing this, it was often difficult and always difficult, actually, to actually know, okay, when have we finished, and did we solve what we started out to do? As a group of people coming together, a group of very bright people coming together, it's easy for us to either make assumptions or also go down wrong and interesting paths. So this process has really helped us in developing quality standards. Third, the third part of this is having this information in the standards, helps you understand, users understand the process that the standards developers went

through, and the concepts we had when we put these design items of the interface items into the standard. And a great byproduct of that is that when you're writing your own standards, or I should say, your own systems, and creating your own specifications, that you have or you can often capture the information right from the standard and include it in your specification. So that's just kind of a byproduct of the work that was done. So this just kind of illustrates what we've just talked about.

On the left side, you see a specification development, and you have a concept of operations and requirements and then the design items, and with an ITS standard with SEP content, you see that that information helps with all aspects of that specification development. Now when we go to those standards without the SEP content, we really only address-- we get right in the standards, we get right into the design details, but you really don't have that background of information in the standard to help you with your specification. Okay, here is where we're going to break the mold a bit, possibly. You picked that up in A202, in Writing User Needs, but that you'll understand that we're going to learn that requirements development is a process. It's call the systems lifecycle V diagram. By appearance of that, you might think the requirements development is a stage in a process, but here we see that requirements development is a process itself. So we'll learn that requirements development is recursive, we'll learn that it's iterative, and we'll learn that it's a process of discovery.

So if you recall, there were different types of requirements that we talked about in A103. We talked about functional requirements, and we talked about performance requirements, and we talked about nonfunctional requirements, and constraints. Well that same process could be applied at a high level, looking at a system, and as that system is broken down, we can then look at subsystems and apply the same process, and then, as we get into the subsystems, we get into interfaces, and again, we can apply that same process. So we're applying the same operation at different levels, so in this case, we call the development process recursive. Now can any of you remember from A103, there was a word that started with a D, that we called this iterative application from one level to another. There was a D word, and this is you get big bonus points for remembering that. Anybody want to take a guess? Exactly, thank you for answering. Decomposition, that is the word. And so obviously you were paying attention there. Thank you so much. Let's move on to the next slide.

Now we talked about requirements development as being iterative. Sometimes when we're creating, even user needs, we'll already be thinking about requirements that may spawn from the user needs, and maybe capture even a couple of design ideas, and we want to capture that as a flow of information. You'll see, so you might have a thread, if you will, of thought that you want to capture somewhere in your discussions. It goes all the way down. Usually, and typically, though, you will have user needs discussed and then those will be iterated with the requirements so that you think might be generated and when we get to the requirements, we'll be making sure that we are addressing the user need, and then again, you'll have this cyclical occurrence between the requirements and the design. Sometimes you'll get to some sort of design in your system, and realize that, oops, something is not going to work, and you really have to go back up and address the user need. So we call this iterative, because we may be revisiting user needs and requirements, revising, removing or adding to them.

Our third item in this section was-- is that requirements development is a process of discovery. This is what's called an inquiry circle, and all inquiry circles have periods of action, and reflection. Action to move forward, and reflection to consider whether the work is complete, going the right direction, or whether redirection should be considered. So we have, in the circle, we have discover, and here we discover our requirements and those requirements lead to system and subsystem requirements, and as we're going, we document them, and we validate them, and the process continues in this cyclical, or this I should say, in this process of discovery. So we're discovering, we document what we discover, and then we validate. And recall from A103, we validate requirements to their parents, and other requirements we've already established. And as I said previously, the standards are a source for this requirements discovery, because the standards contain information for you to use in this discovery process, and that applies to either set of standards, those with SEP content, and those without. And specifically, we'll be talking about looking at the standards that do not have SEP content, and how we might use them in our process of discovery.

When we're doing this process, we're not trying to write requirements for the standard, I said that. We're discovering the interface requirements that support the system. We are documenting as we discover the items, and we phrase, this is important, that we phrase the interface requirements in terms that are used in the standard. This is where the standard becomes very important, because we want to make sure how we phrase things, are compatible with the standard. And then we'll validate our consistency of our interface requirements with the other interface requirements, and also with the parent system or subsystem requirements. So we are doing this process from the system that we're trying describe, and it flows from the user needs, and functional requirements of the system. We reach this stage through decomposition. It's okay to add and correct what you've done. There's also an item called dialogs, a concept we call dialogs, and that's where there's a need for a series of exchanges of information and we're not really covering that in this class today, but you might think, well it may take several messages to actually do an operation out in the field, so you have to document that series of exchanges. We want to add, we want to capture performance and constraint criteria in the process, and we also want to see, one thing you can do to help you is to see the NTCIP Guide, which you all have referenced before. Okay, let's move on.

Now we're going to talk about avoiding the pitfalls when we're writing requirements. The first two bullets, talking about well-formed requirements, those are a review from A103, and then we'll talk about the process of discovery of requirements and we'll be sort of requirements explorers, and we need to think about the dangers as we are exploring, and so basically we'll be discussing those pitfalls there in our exploring of the requirements, and then we'll play a little bit of a game after that, and see how we do. Again, this slide is from a review from the A103 Module, and this talks about the structure of a well-formed requirement. We have the actor, action, target, constraint, and localization. So the actor, that's typically the system or the name of the system, so you might have the system shall or ABC will do this. That's typically the actor in our well-formed requirements. The action is the shall, the shall statement. That has for decades, that has really identified a type of kind of a contractual item that you have to fulfill. And then the target, that is something that receives the action, and then the constraint and localization, those are items that really kind of make it mean something. Constraint is the measure of success or failure, localization is under what circumstances does the requirement applies. And again, the

localization and constraint portions are important, but not all requirements will have both. So here's an example. The system, which is the actor, shall generate, that's our action, event reports, that's our target, containing the following information, that is a constraint. Now we don't list that information here, but that would be the constraint, on a scheduled interval, and that's the localization. So this is an example of the form of or the structure of a requirement when it's well-formed.

Now I really like doing this. I did not always express requirements this way in my history of doing systems engineering, but I found I've become a big fan of this. Users get used to the style. Users, I should say, and even the system definers get used to the style, and then key items are easy to pick up. So I really, really recommend that you use this form. If a requirement can't be stated in this simple format, you probably need to define the functionality using multiple requirements, and I've found myself, where you end up having compound requirements, and it becomes difficult to express because you're really-- really trying to address more than one thing. Well just split them up into different requirements. Recall for me one of the three. We had characteristics of well-formed requirements, they were necessary, they had to be traceable and useful, they were concise, understandable and expressed in a declarative language, obviously shall statements, attainable, realistic, they had to be realistic and available in resources during the resources and time of your project. They should be standalone, in that the requirements should be completely stated in one place, not grouped. To be able to understand the concept it shouldn't have to be grouped in some complicated way.

Now, we do, in our requirements, often reference other documents that have a particular requirement or particular constraint that we want to abide by, so that still can be stated-- is still of the form being standalone. Needs to be consistent, can't contradict itself or other requirements, unambiguous, it should be clear and only susceptible to only one interpretation, and it should be verifiable, that we can meet it, meet this requirement through inspection, analysis, demonstration or test. Now you'll find that you're writing requirements that it's kind of an art form, because what happens as you express things, and you take a look at it a second time, you come back to it, you find that, wow, I could have done that better. Oh, there was a hole when I said this, and so you keep working on this, as we talked about being iterative, but what you do is, what we're trying to do, is get our requirements sufficient for purpose.

Okay, we talked about being explorers, so some things that we need to watch out for, in developing requirements, are listed in front of you. I'm going to read a little bit of definitions for each one of these, and it's very important that you pay attention, because we're going to use this in our game that comes in the presentation that follows. Okay, so design and implementation, there's a tendency on the part of analysts and customers, who are defining requirements, to include design and implementation decisions with requirement statements. Such information may still be important. In this case, the information should be documented and communicated in some other form of documentation in order to aid in the design and implementation. Okay, so we don't want design and implementation details in our requirements. Okay, over specified. Requirements that express an exact commercial system, or set of systems, that can be bought, rather than made, requirements that state tolerances for items deep within the conceptual system.

A lot of times you'll see, like, air requirements at very low levels. Requirements that implement solutions, requirements should be solution free. And so what you see is that we don't want things over specified and we want them all, the requirements at a given level, to be at the same level, if you will, as we go through our process. Requirements can be over constrained, they have unnecessary requirements. For instance, I thought of this example. A system, it might be stated that a system that runs on rechargeable batteries, someone might state that the system needs to be rechargeable within three hours. But it's possible that that's too restrictive, and 12 hours recharge time is sufficient. So if you overly constrain it, possible solutions are eliminated, so that's what's important about that, so we don't want to over constrain. Unbounded, requirements that make relative statements, and those statements can't be verified. For example, the requirement to minimize noise may be restated as, noise level should not exceed. Requirements that are open ended, frequently stated as including but not limited to, or lists ending, etcetera. Etcetera is not a good thing to have in our requirements. Or, requirements making subjective or vague statements, such as, user friendly, quick response time, or cost effective. Some of those terms are actually okay at a user need level, but when we get to requirements, everything needs to be testable.

Okay, another pitfall we have when we're writing requirements, this last one we'll talk about is being assumptive. These are requirements that are based on undocumented assumptions, such as, requirements that a particular standard or system that's currently undergoing development will be completed. That's the assumption and an alternative requirement should be documented. So in other words, if you have something where you're going to be kind of need to make some sort of assumption, you need to make another requirement that says that that case exists. That way you know when you're going through the process, that you can't proceed because that requirement is unfulfilled.

Okay, so we're going to go through our game now, our activity, and I will state a requirement, and we'll go through on whether-- I'll ask you, is it designed, over specified, over constrained, unbounded or assumptive. It might be design or implementation, I should say. So I'll read through it, and then give you a chance to guess. Okay, Serial Bus 2, System Messages. The command response messages shall use similar message format as serial bus number one, collecting operational status, detection speed, reports, occupancy reports, counts, etc. Which one is this? Alright, we got an answer right away. It's unbounded, yes. Right, the key word there was etcetera. We don't know where the specifier was going, so that's a great answer. Okay, let's go to the next one. Enclosure Door Frames and Door Seals. The dimension between the door edge and the enclosure external interface or surface when the door is closed and locked, shall be .156 inches, plus or minus .08 inches. What might the answer be here? Remember, this is a system requirement that we're talking about here. Yes, well, you might think, someone said over constrained, and then someone said over specified. I selected over specified at this point, but you could also say it was overly constrained if as long as a higher tolerance might be okay. But I don't think that that's really what the developer of this requirement was saying. He really wants this requirement to be there, but maybe it's just at too low a level, and it's more of what we call a design requirement. Okay, so over specified. Let's go to the next one. Power Distribution Assembly Wiring. Three 36-inch minimum length number 8 gauge wires, one black for AC plus, one white for AC minus, and one green or green-yellow, for equipment ground, shall be attached to the rear of the assembly at the AC raw power terminating block. So what might be the answer here? Yes, it looks like we've got a couple there. Design and implementation. We really, really

got down to the details in this one, didn't we? Okay, we have one more. I think it's actually, we have a couple more. Okay, Public Safety. The system shall use the low-voltage output options available to the ITS Cabinet Standard Version 2, currently in development. Do you remember what this one was? Very good, I got an answer right back, assumptive. Remember, I saw an answer for unbounded. You could sort of think of it as being unbounded, but the word here, a better-- I think the best answer here would be that it's assumptive, that we're assuming that the standard is going to be in place. Okay, let's go to the next one.

Startup Considerations. The Engine Board low-level hardware and O/S initialization shall be completed, and application software shall be capable of exercising control of all ATC unit hardware within a maximum of 4.5 seconds. Now this one is a little harder. Yes, someone already answered, over constrained. And why this was not fair, is that you needed to know a little bit of history. What happened is, there was this specification for how fast this engine board, as part of the ATC program, needed to reboot, and someone arbitrarily picked 4.5 seconds, but in actuality, it was taking 6 or 7 or 8 seconds, so the spec was changed from 4 seconds to 10 seconds, because there was really no need. For instance, this would be used in the case where, if a signal or an intersection was in a flash condition, the people sitting there in their cars need to know it's taking 4 seconds or 6 seconds before the intersection turned green again. So that was basically the reason for this over constrained statement on this pitfall. Okay, some words of wisdom. Knowledgeable stakeholders must be part of the project team, and this applies especially to people who know the devices and systems you're interfacing to. Many times, if we do projects and we're dealing with the same people we deal every day, let's say one department gets funding for a project, and you kind of work-- there's a tendency to work in your own world, and you really don't know what all the impact it's going to be from the systems and devices around you, and it's easy to make assumptions. So it's really important that a knowledgeable stakeholder is part of the team. And that, for whatever device that you're interfacing to, or system you're interfacing to, you should have the experience in house, or if you don't, hire a consultant that has that experience. It's very important. But if you're like me, I find that, in all these conditions, it still helps to research it yourself. You don't want to-- you need to be able to make good decisions, and have enough information to make good decisions.

Now someone asks the question is, do I need a subject matter expert, or an (SE) or a systems engineering expert? Well, I would say that you might use a systems engineering expert, that's kind of above this whole level of class, it's really, that spans all the classes you've been getting, but hopefully you're being trained in this area as well, and that it's possible for you to tackle these kinds of items yourself. Now what I was really referring to here though, was the subject matter expert itself. Okay, I had another question, is, why don't you want the design as part of the requirements? A lot of times, if you,- people start writing requirements for something they've already had built. And what you want to be able to do is create that-- is to lay the requirements out, to know what needs to be satisfied, and that gives you the most flexibility in developing your design. Many times when we are expressing requirements, there are some levels of architecture and that's applied-- that's specified in the concept of operation that's assumed. For instance, if you're creating a signal system, you're going to be assuming that there are still devices out in the street, and a central system located someplace else. So that's a kind of a high level architecture, and not talking about design details. So that's why we don't want to put the

design into the requirements, because we want to give the most opportunity to find the best solutions. Very good question. Any other questions from what we've learned so far?

Okay, let's go on. We're going to go to our third objective, writing requirements when an ITS communication standard does not have SEP information. We're going to talk about an example signal system, and using the National Transportation Communication for ITS protocol standards, the actuated signal control and the closed circuit television, and then we're going to apply, we're going to talk about the system a little bit, and then we're going to apply the discover-document-validate process. Okay, here we have a traffic management center, and we have our downtown area depicted as a grid, as they usually are. And we have, out on the street is, we have traffic signal devices, field devices, called-- well these would be cabinets they go in, but we're actually going to be talking to the computers that control the traffic, which we call a controller, and then we're also going to be talking to surveillance cameras that are in some of the intersections, and we want to be able to move them, and observe what's going on in our and on our streets. So here's a context diagram, for high level architecture diagram, of our system, so we're trying to decide,-- we're trying to defining this traffic management system in the middle, and as we develop our requirements, we're eventually going to get down to the interfaces, and that's where these ITS communication standards apply, and 1202 is the ASC standard, so that will help us talk to traffic controllers, and then NTCIP standard 1205 helps us talk to the CCTV cameras.

Okay, just again, to review what an ASC device is, it's an environmentally hardened computer device. It runs applications that control the vehicle and pedestrian right of way, at roadway intersections. It communicates with the sensors that are in the street, or sometimes they're overhead, or there's various technologies to determine when there's vehicles approaching, and then it communicates with what we call field displays. You might think of that as traffic lights, because that's exactly what we're talking about there, to control the right of way. We're talking about cameras, we're talking about providing composite video over a closed system to a traffic management center, so it's a restricted access that we're talking about here. And just-- want to point out that the standard covers the camera control, including the camera, the lens, and pan-tilt functions. The actual video, how you get the video back, that technology was chosen not to be in the standard, because that tends to change quickly, and advance quickly, so how you get it back, the form you get it back, whether it's closed circuit, whether it's cable or Ethernet or some other form, that is up to you. Each of the NTCIP, what we call device standards, it's-- they need to be able to do the following. They need to be able to configure a device, control the device, monitor the device, and read historical information from the device. Those tend to be what is in every device standard for NTCIP. Again, these are all communications, they say device standards, but they are all really communications to devices.

So again, in review, in stating what we've said earlier in the presentation, the organization of the NTCIP standards without SEP content, have some general information, and then it goes into the design or what we call the object definitions, these are the interface data items, and we call that a management information base, or MIB. And then there's a discussion, if, in some of the cases where they have block objects, and block objects, that really defines a group of data elements as a single larger data element that could be transferred more efficiently, so block objects are only in some of the standards. And then there's annexes that contain things like conformance groups and other items and basically a conformance group is a unit of conformance that's used discussed

by a collection of related items. So there might be, for instance-- well we'll get into that in a moment.

So, we'll talk about the management information base, it defines data elements, of the device, that are covered by the standard, it's written in abstract syntax notation, ASN.1, and I know that this was touched on in A202. It's a reference or resource for those familiar with the device. Now, it's easy for an expert to adapt to the standard and a view, and know what's going on when he sees a data item. For instance, a software programmer, I had this, because I have a software development background, when they start out, they usually think of themselves-- they usually learn one, maybe one language, they learn it pretty well, then they learn a second language and they kind of learn the second language in context of the first, but then when they start learning third or fourth languages, pretty soon, they know how to adapt and know what things to look for in any language, so that's kind of similar thing that goes on with these devices. If you have somebody that's an expert, they'll be able to pick up what's going on in the standard pretty readily.

Now, it's ineffective for a novice to learn the device by reading a MIB. For instance, if you're not familiar with the traffic control device, and what it does, it's not going to be very effective for you to try and become an expert just by reading the MIB. That's kind of like learning English by reading the dictionary. Here's what an object looks like, that's expressed in a MIB. This is the ASN.1 notation. So here we have the object name, we call it, phaseWalk. Anyway, just for background, a phase, in traffic control terms, is like a turning movement or an action and here, this is talking about a pedestrian one, we call phaseWalk. That's the name of it. We have a range of values specified. We say that its access is read-write. That means that the central system can tell the device what the value should be, and it can also read it, of course. So that's why it's called a read/write. Sometimes they will also be read-only. In this case, the status is optional. The options for this are or the possible items that could be in here are mandatory, optional, deprecated or obsolete. And when we say it's optional, it means that the traffic control device does not necessarily need to support this particular object and it would still be considered conformant to the standard. So this is an optional object. This is the definition of this object in human terms, in English. It says, Phase Walk Parameter in seconds. This shall control the amount of time the walk indication shall be displayed. And then we have a reference, and this tells the reader where this object came from, or the idea of this object came from. In this case, it's referred to NEMA TS 2 standard and it gets the clauses where phase walk is described in that standard. And finally, this last entry is-- this says, phaseEntry 2. Basically, it says the phase walk parameter is part of this other parameter called phase entry, and it's the second item in that. So it's the location within the MIB, or the location within another object that's specified at the bottom here.

At this point, if you want to get into more details about the MIB, and reading MIBs, I suggest you go to our Participant Supplement, and there are some good references about learning MIBs and reading MIBs, and such, but it's beyond our class today. Now we talked about conformance groups. And that's kind of a groups, objects together that perform kind of a like function of the device. Now these are-- this is for the actuated system signal control, and there are actually 33 conformance groups in the standard, and these are just six of them. Again, the conformance groups cover a level of capability, the block objects I mentioned earlier aren't conformance

groups, that's grouping objects for purposes of efficiency. And again, these groups, these conformance groups may also be mandatory or optional, and they're formed-- they're in a table here, so that if someone was making a specification, and you wanted to use this conformance table, and this teaches you this in the standard, you could make your specifications to your vendors, based on the conformance group. I want to say one point here, is that I would say that we don't prefer you do that, we prefer you go through the systems engineering process, but at least this kind of gets you in the idea of whether you want a certain group of features or not.

Okay, here's a sample of a phase conformance group, or of the phase conformance group, and it's in the form of a table. And what you see is the clause, this is called phase conformance group, it says its status is mandatory, because of that, all the objects must be supported here. Max phases, we say that you can see this goes across as the-- that's the object type, S stands for, it's a status object. Down below, you'll see P, and that's a parameter object, and there are a couple of others, and you can see the standards for those. You see the range of allowed values, and this is what a conformance group looks like, and I know that in A202, you went through this also. But it's really-- what you did learn in A202, was that these conformance groups gave you clues to user needs in a very good way, because they're packaged that way by the developers of the standard. So they give you a great start to our inquiry circle, and then from there, they feed into-- can feed us into the thought process going behind requirements.

Okay, now we're going to apply the discover-document process. We're going to assume that, in our system, the decision has been made to use the NTCIP standards. And then we're going to write a user need, and then the requirements that are consistent with the ASC standard, and also then we'll have another little example, where we will test our skill for the CCTV standard. Okay, again, okay, so this is an example user need, that we would have come up with by looking at the conformance groups, or it would be just about anybody who's working on a signal system, they would come-- any group of stakeholders would come up with a user need like this. The user needs the system to display current phase outputs information, and we describe that as, red, yellow, green, for each signalized intersection. And then we have the rationale, which is that TMC operators must be able to identify the current right- of- way for each signalized intersection on the system. And you could sort of think of a guy,- an operator sitting at the Traffic Management Center, wanting to see what's going on in the street. And remember that we want our user need to be uniquely identifiable, and that, we see a reference there, it's a major desired capability, displaying intersection right of way. It's solution free, we don't state how that's to be done, and it captures the rationale, why we want to do it. So again, this goes back to our A202 module. Now in doing this, we need to make sure that key items in this user need are addressed by the requirements. For instance, what do we mean by-- what do we want to do by display? What does it mean by current? We want to make sure that we're displaying the phase outputs information. It's kind of defined for you here, but we need to make sure there's a requirement that speaks to that. And we want to be able to identify or select or something, a signalized intersection, or have all of them displayed. Somehow our requirements need to address each piece here.

So based on this user need, the user needs a system to display current phase outputs information for each signalized intersection, you might-- probably the first requirement that you might come up with would be this one. The system shall allow the operator to display the phase output

information for a selected intersection. So we're assuming here that they want, the user wants to display this information for a particular intersection at a time. That's a little bit of assumption here, but you can't really display all of that information for all intersections in a city, because there's just not enough room on your screen, anyway. So this is likely our first requirement that we come up with. And just for reiterating, and bringing our skill set home here, I just want to go back through that we want to show that we do have our actor, and that's the system. We have our target, sorry, we have our action, which was a display, we have our target, which is the phase output information, and then we have our localization. It's for a selected intersection. So again, we're writing these requirements in a well formed structure. And that's really the best way to verify that your requirement is correct. Now whether it's consistent and complete, and everything that we're going to go into that detail, as we move along in the presentation. Oh, this was a great idea. I could have written the requirement for a selected group of intersections. I suppose that could be another requirement. So you could have a selected intersection or a selected group of intersections. Yeah, that might be another one that's added here, but I don't have that down in our presentation, so we won't go there today.

Okay, from that previous slide, from this requirement, we have-- now we have derive requirements, something that was really not-- something we really expressed explicitly in the user need, but that comes because we need something to support what we just-- support the requirement we just wrote. So this is-- this might be called a derived requirement. The system shall display geolocated user selectable intersection icons for each managed signalized intersection on the road map. Basically, the idea here is that the display is going to have some sort of map in which a user can select an icon. And again, there is even in my description of this, you're already-- we're already deriving other requirements, such as these two. The system shall provide a road map of the area to be managed by the system, and the system shall provide a pointing device which allows the operator to select icons on the road map. Now you might say that maybe this is getting silly, we're getting into too much detail here. Again, the purpose is to document your system in a fashion that's sufficient for purpose. And usually that's lower than what you think. And especially when it comes to interfaces, that's where the end user is especially sensitive to-- so you'll find that interfaces in most systems, become the largest part of the system specification. If they have the opportunity, if they're not buying something off the shelf, they're wanting to specify how a system should work. This gets very detailed.

Now, after we got through those other requirements, and you'll note that it took four requirements just to kind of cover display and each signalized intersection and phase output information. So those were those key items from that user need that we discussed at the beginning, and it took four requirements just to cover those. We still need at least one more, and we need-- and that's this one, retrieve phase outputs information. The system shall retrieve phase outputs information at a rate of once very second-- acceptable range .6 to 1 second, from every intersection managed by the system. So you'll note that in this requirement, I needed to put a range here, because systems really don't work that accurately to the point of being able to say it's every-- precisely every second. So you have to have a range of acceptable values that make this testable, and that makes it achievable.

Okay. One of the comment came in about, that instead of using the words, shall allow, you could term it, the system-- instead of the system shall allow the operator, he preferred, the system shall,

and then do whatever it does, and that's acceptable also. The question of allow, might come in, is if this is something that has to happen every time, or it's something that may occur at different intervals, but I think that in this case, that that would have worked just fine. Good comment. Okay, now we're going to go through this validation portion, or verification and validation portion, and I'm not going to go through the details of this, but just a reminder is we need to look at it and see if the requirements, are the requirements we just wrote, are they well formed? And then we need to say, are the requirements logically consistent with the parent requirements and user needs? Are the requirements consistent with sibling requirements? Is there traceability between the needs and the requirements? And then, are all key items of the user need addressed by the requirements, and we're going to go into that in the next slide. If you recall from A103, that we did go through this process before, but that's what we're talking about when we get into the validate portion of our inquiry circle. So here is the user need that we stated at the beginning of this section, the display-- the intersection needs the system to display current phase output information for each signalized intersection, and then it goes on to the rationale. So here are the requirements we just wrote, and what we see is, when we were talking about road map-- when we were talking about, sorry, signalized intersection, that those were really addressed, the road map, talking about geolocating, and intersection icon, talking about appointing a device, this is all in the idea of selecting an intersection, getting to an individual intersection. Then we also talked about the displaying of the phase output information, and again, we spoke to the requirement of phase outputs, and we also talked to the issue of displaying. And then, finally, we talked about retrieving phase output information, and that gave us-- we spoke about phase outputs information, we talked about each signal intersection in that requirement, but we also addressed the idea of what it meant to be current. Remember that range of values, so now what we've done is, we have now made sure, by going through our requirements and seeing that we address each of the aspects of the user need, that we have satisfied this user need in our requirement specification.

Okay, before I go on we're going to have an activity in a minute, but there was another question that came up. Would you use requirements to buy something off the shelf? Yes, you can do that. I wonder then if that would be-- well then, what might happen, for instance, there are companies there that sell systems, that you may have researched and like, etcetera, but then you still, whenever a system is being put in, even if there is a kind of an existing base product, you still need to tailor that for your organization. So you'd still want to go through this process. And maybe, when you're buying something specific off the shelf, like, when they're calling out a certain part or something, if you're at that level, and you're talking about a device, then you're probably at a too low level anyway. That is a solution. Buying a system, a blanket system like that, and not going into the details about it, is more of a solution than it is a requirement. Okay, so we just went through an ASC example, actuated signal control example. Now we're going to have a little activity here, and we'll give you the opportunity to select the best answer, and this particular example will come from the closed circuit television standard. Here's an example user need. It might come from reviewing the standard, or from a group of stakeholders that are putting in a CCTV device. It says, the TMC operator needs to remotely control the position of the CCTV device. And the rationale is, the TMC operators must be able to control the on-street cameras to be able to determine roadway congestion, assess environmental conditions, monitor incidents, and verify proper signal operation. Again, this would be by just about any stakeholder in this area would come up with something like that. Okay, now I'm going to go through, I'm going to

show you a couple of requirements and you're going to have an opportunity to pick from the two, and I'll use the facilities of this seminar to do that. You'll have an opportunity to pick the best answer, the best requirement, that fits with this user need. Okay, the first one is, get CCTV position. The system shall be able to retrieve the pan/tilt/zoom of a remotely located CCTV device. The second one is, set CCTV position. The system shall be able to set the position of a CCTV device located within a 12 mile radius of the TMC. Now we'll start the poll. Select the better requirement. And let's look at our results. And we see here that we had 60 percent of us thought it was A, and 40 percent of us thought it was B, and that just shows you that this is somewhat subjective, and that we just need to, again, strive to do our best in this process. Now let's take a look at this. The answer that I came up with was this one. And the reason, although we didn't define, remotely, it's not defined here. This requirement does define the position of the camera, it does speak to that requirement. The second one, the idea of position can be confused with location. You see that? With the format of this requirement. If you wrote it the first time, you might think that's okay, but then when you come back and read it again, you say, well is the position a geographic location, or is it the position of the camera. So in this case, I pick A, but it was, you know, you had to really look at it to get there.

Now let's go on to the next one. Get CCTV feature status. The system shall be able to retrieve the camera feature status from the CCTV device. The second one is, set CCTV tilt position. The system shall be able to set the tilt position of the CCTV device. Which is the better requirement in this case? Again, remember what our user need was, which was, the TMC operator needs to remotely control the position of the CCTV device. Okay, now we'll put up a poll for us again. Okay. We'll close the poll. Let's take a look at our answers. And we're exactly 50-50 on this one. Well that's kind of understandable, if we go back and look at this now. Both of these are actually very good requirements. They are well formed, precise, things like that. But I selected this one, because our user need was talking about the position of the camera. The user need wasn't talking about the feature status, which might be some other aspect of the camera, which wasn't defined here, but we could assume that it was defined someplace else. So that's why this one was second, it wasn't actually in the form of the requirement, but was in what it covered, and so if you didn't remember your user need very well, you didn't get this one right. Okay, you all have been doing outstanding. This has been a great class. Obviously an advanced group here, and we've talked about understanding that requirements development as a process, we talked about avoiding pitfalls when writing requirements, we've written requirements when an ITS communication standard does not have SEP information, and now we're going to use traceability matrices as tools for requirements development. So we're going to talk about building traceability matrices, we'll talk about, again, needs-to-requirements traceability, and requirements to interface item traceability.

Now it's important to remember, before I actually get into this detail, I want see if there's any other questions. And I do have this comment. So the idea is, we're looking at-- we're taking our user needs, as the basis, we're developing requirements, but then we're turning to the standard and writing the requirements in the language that's used within the standard. And how do we know that language? Well we have to go in and take a look at the management information bits, and who's going to help us understand what that says? Is it because we have an in-house expert or a subject matter expert that we hire. That kind of sums up the last section. Okay, let's move on. So we want to do this is a little refresher from before. We want to trace user needs to

requirements, we want to trace requirements to design items, and we have-- and we're going to talk about tools that are used to help us verify completeness and correctness, and we're going to suggest capturing this traceability information through the discover-document-validate process. So in other words, when we are thinking of our user needs or our requirements, and sometimes we think of design items, we might want to take a scratch paper somewhere, or virtual one, or document, and write these things down, so that we don't forget them. But what a traceability matrix does for us, is we put them right in front of us, and in an area where we can track this and make updates. This is a needs-to-requirements traceability matrix, or we're going to talk about needs-to-requirements traceability. We call this NRTM. In that, every need must be addressed by at least one requirement. Every requirement must trace to at least one need, and any need that's not addressed by at least one requirement, means a requirement was missed, or the user need must be reevaluated. If we could not take that need and express it in a shall statement, at least one shall statement, then we need to really take a look at how we wrote that need, or if that need was really valid. Conversely, every requirement that does not address at least one need, means that the requirement must be reevaluated or a user need was missed. So in other words, we wrote our user needs, and we came up with a bunch of other requirements, but we kind of, in that process, need to go back and say, are we writing a requirement for something we didn't want to do, or did we really want to do it, and we should be adding that to our user needs. That's essentially what that's saying. So every aspect of each user need should be addressed in the requirements, and that's the part that we stepped through previously, we're talking about our key items.

Again, I talked about the needs. I was really describing a needs-to-requirements traceability matrix. Here you see a user need, it was called-- this is from our example. Provide intersection right of way information. We have, in the table, then, the ID for that user need, and then again, we go over to the requirements, to the right of that, where we specify the requirement IDs, and the name of those requirements, and we see that our requirements, the five of them that we wrote there, showing that they support this user need. The next one down was just to show you there was a lot of-- in one of these kinds of NRTMs, there's many, many, many entries for each user need, and again, more so for each requirement.

Okay, now we're going to talk about requirements-to-design traceability. In that, every requirement should trace to at least one interface item. Every interface item must trace to at least one requirement. Any requirement, that's not addressed by at least one interface item, means an interface item was missed, or the requirement must be reevaluated. We used to say that there should be no orphan interface items. Every interface item that does not address at least one requirement means, the interface item must be reevaluated, or a requirement was missed. I guess that speaks more to my orphan interface item. So again, just like we did between the user needs and requirements, we have here with our requirements to design, and by design, we're talking about the items in our interface specification. Again, every aspect of each requirement should be addressed with at least one design item. Here we have the requirements traceability matrix that's used to capture what we just discussed. Across the top you see the requirements ID, the requirement, the dialog identifier, the dialog name, the object ID, which is the object in the standard, and the object name. This is how we trace requirements to the design items in our interfaces.

So now let me go back again. In the standards, when we want to capture a series of exchanges of information, a series of messages between two items, then we capture that in what's called a dialog. And I have examples of dialogs in the Participant Supplement. And what I would like to say is, in this process, what happened was, we looked at, okay, we need to retrieve phase outputs information, how did I know to call it phase outputs information? Some expert may call it something different. Well, the expert would go to the standard, and he would go look, and he'd say, okay, this is what I'm looking for, this is called phase outputs, and I see the information that I want in there, and it's all described in details, how that phase output information is broken up, and there's much more information included in this, that's not listed, but just was using this for an example. Great, okay.

Now we're having our final activity, and that final activity is to see what we learned today. So get your chat box ready. So what did we learn today? To understand that requirements development is a what? Any answers? It's a process, thank you. It's a process of recursion, a process of iteration and a process of discovery. Second one, we learned to avoid what when writing requirements? What do we need to avoid? Pitfalls. Excellent. Process and pitfalls here. Need to write well-formed requirements, and watch out for design and implementation, over specification, overly constrained, unbounded or assumptive requirements. Excellent, pitfalls. Third, we learned how to write requirements when an ITS communication standard does not have-- thank you, SEP, systems engineering process information. Excellent. And finally, we learned to use what as-- what matrices as tools for requirements development? Someone brave? We just talked about it. Traceability, thank you. So again, these are tools, the traceabilities are tools to do what we want to do and that's to make sure that our requirements track to user needs, and our design tracks to requirements. Very good, excellent class. These are some sources for more information and I think there's others listed in the Participant Supplement.

I want to encourage you to keep your eyes and ears open, and we have a new set of classes that will be coming out next year, that follow from this one, where we will be talking about particular modules, the ASC and the CCTV. We'll talk about those. We'll center those classes on those standards explicitly, and they are very helpful, because again, they don't have the SEP information. If there are no further questions, then, this concludes module A203, Writing Requirements When ITS Standards Do Not Have SEP Content.