T317: Applying Your Test Plan to NTCIP 1205 Standard

Table of Contents

Module Description

This module is the last module of the NTCIP curriculum path for Closed Circuit Television (CCTV) Camera Control without following the System Engineering Process (SEP). The prerequisites of this module are A317a: Understanding User Needs for CCTV Systems Based on NTCIP 1205 Standard and A317b: Understanding Requirements for CCTV Systems Based on NTCIP 1205 Standard:

This module includes information in relation to the object definitions and conformance groups of the NTCIP 1205 standard. However, unlike other NTCIP standards such as NTCIP 1203 and 1204, the NTCIP 1205 standard was developed without following the System Engineering Process. Therefore, the standard does not contain the information that is necessary for developing test plans and test documents as required for testing CCTV camera control functions. The following information is not included in NTCIP 1205, but is necessary for developing the CCTV test plans and test documents.

- User Needs
- Requirements
- Dialogs
- Protocol Requirements List (PRL)
- Requirements Traceability Matrix (RTM)

1. Introduction/Purpose

The purpose of this module is to assist the user and agencies in developing test plans and test documents specific to their CCTV needs based on for NTCIP 1205 – Object Definitions for Closed Circuit Television (CCTV) Camera Control. Revision Amendment 1 to NTCIP 1205 (November 2004) is used to develop this module.

This module covers test materials related to elements of the NTCIP 1205 Standard required to develop test plans to verify that an agency's CCTV product or system meets design specifications and other requirements of the NTCIP 1205 CCTV Standard while following standard testing approaches and methodologies as suggested in the IEEE 829 standard.

2. Reference to Other Standards

Significant revisions were made between the 2008 version and 1998 version of IEEE 829 although the main principles are the same. These revisions help clarify the test processes that are not explicitly described in the 1998 version. The NTCIP 8007 standard refers to the 1998 version. In order to avoid the confusion between NTCIP and IEEE 829-2008, this document further explains the

difference between IEEE 829-2008, IEEE 829-1998, and NTCIP 8007, and how to use them in developing test plans and test documents.

2.1. Difference between IEEE 829-1998 and IEEE 819-2008

One of the major revisions in IEEE 829-2008 is to change from being document-focused to being process-focused. New concepts are also introduced by IEEE 829-2008 by adding the descriptions of integrity levels, master test plan, level test plan, and additional test documentation.

2.1.1. Integrity Levels

The following is excerpted from IEEE 829-2008 with regards to integrity levels.

4. Software and system integrity levels

In addition to a similar role in all other life cycle processes, a scheme of integrity levels provides the structured means for setting the breadth and depth of testing. A high integrity level requires additional test tasks than does a low integrity level. A high integrity level requires testing that is more in-depth than does a low integrity level. Without a requirement for a certain integrity level, the tester will not know which functions, requirements, or products require only a cursory effort and which require an intense effort.

This standard uses integrity levels to determine the testing tasks to be performed. Integrity levels may be applied to requirements, functions, groups of functions, components, and subsystems. Some of these may not require the assignment of an integrity level because their failure would impart no negative consequence on the intended system operation. The integrity scheme may be based on functionality, performance, security, or some other system or software characteristic.

Whether an integrity level scheme is mandatory is dependent on the needs of the stakeholders for the system. The user may follow the four-level schema provided as an example in this standard or may use a different schema. However, if a different schema is used, a mapping should be made between the user's schema and the example. Some software elements and components may not require the assignment of an integrity level (i.e., not applicable) because the failure would impart no consequences on the intended system operations. The user may want to add a Level 0 to Table 1. Level 0 would cover failures that would cause no consequences or are not applicable.

There shall be a documented definition of the integrity levels or the decision to not use an integrity level scheme. The integrity level (or the decision to not use an integrity level scheme) shall be assigned as a result of agreements among all specified parties [or their designated representative(s)], such as the acquirer, supplier, developer, and independent assurance authorities (e.g., a regulatory body or responsible agency).

4.1 Integrity levels

An integrity level is an indication of the relative importance of software (or of a software characteristic, component, or test level) to its stakeholders, as established by a set of selected attributes such as complexity, risk assessment, safety level, security level, data integrity, desired performance, reliability, quality, cost, size, and/or other unique characteristics of the software. The characteristics used to determine the integrity level vary depending on the system's intended application and use. The set of characteristics selected as important to stakeholders, together with its arrangement into discrete levels of performance or compliance, constitutes an integrity level scheme. An organization may have an integrity level scheme that is applicable for all projects; in which case, a project can simply reference the level in that scheme that is applicable to this system.

Unique identifiers (such as integer numbers) denote individual integrity levels within an integrity level scheme. Implicitly or explicitly, software with a relatively "higher" integrity level will require more testing effort, and more test documentation, than software with a relatively "lower" integrity level. The characteristics selected for inclusion in an integrity level scheme, and the number of integrity levels provided, will vary depending on the stakeholders' needs and the software's intended application and use. An assigned integrity level may change as the system evolves. Design, coding, procedural, and technology features implemented by the development organization may raise or lower the assigned integrity level.

This standard uses the concept of integrity levels to determine the recommended minimum testing tasks to be performed. The inputs and outputs of the indicated testing tasks identify the test documentation needed. High integrity software requires a larger set of test processes, a more rigorous application of testing tasks, and as a result, more test documentation. Integrity levels can be assigned to software requirements, functions, groups of functions, software components, subsystems, systems, or groups of systems (e.g., a product line). The test processes and resultant test documentation should be tailored to specific system requirements and applications through the selection of an integrity level (with its corresponding minimum testing tasks and addition of optional testing tasks). The addition of optional testing tasks allows the test effort to include application-specific characteristics of the software.

This standard provides as an example the four-level software integrity level scheme shown below. The categories of the software integrity level scheme shown below are defined based on the seriousness of the *consequence(s)* (of incorrect behavior during execution) and on the potential for mitigation (taking steps to lessen risk by lowering the probability of a risk event's occurrence or reducing its effect should it occur).

Each level may have an associated descriptive word, e.g.:

Level 4—Catastrophic

Level 3—Critical

Level 2—Marginal

Level 1—Negligible

Table 1— Consequence-based integrity level scheme

Description	Level
Software must execute correctly or grave consequences (loss of life, loss of system, environmental damage, economic or social loss) will occur. No mitigation is possible.	4
Software must execute correctly or the intended use (mission) of system/software will not be realized causing serious consequences (permanent injury, major system degradation, environmental damage, economic or social impact). Partial-to-complete mitigation is possible.	3
Software must execute correctly or an intended function will not be realized causing minor consequences. Complete mitigation possible.	2
Software must execute correctly or intended function will not be realized causing negligible consequences. Mitigation not required.	1

Note that IEEE 829-2008 does not mandate the use of the integrity level schemes. The user or agency may use the integrity level scheme provided in the standard, or develop another scheme by itself, or not use any integrity level scheme at all (choose test documentation based on another criteria). The use of an integrity level scheme is a recommended best practice that facilitates the selection of the most appropriate activities and tasks and, as a result, the needed test documentation.

The degree of intensity and rigor in performing and documenting the task are commensurate with the integrity level. As the integrity level decreases, so does the required scope, intensity, and degree of rigor associated with the testing tasks and documentation.

IEEE 829-2008 recommends using the minimum testing tasks that are to be performed for each integrity level. To identify the minimum testing tasks that apply to a different selected integrity level scheme, the users may map the integrity level scheme and associated minimum testing tasks to their selected integrity level scheme. The mapping of the integrity level scheme and the associated minimum testing tasks are documented in the Master Test Plan (MTP) or the highest Level Test Plan (LTP) if there is no MTP (or the highest level document selected if there is no LTP).

Below is an example provided in IEEE 829 on how to map test documentation to integrity levels.

Integrity level	Example test documentation
4 Catastrophic	Master Test Plan
	Level Test Plan (Component, Component Integration, System, Acceptance)
	Level Test Design (Component, Component Integration, System, Acceptance)
	Level Test Case (Component, Component Integration, System, Acceptance)
	Level Test Procedure (Component, Component Integration, System, Acceptance)
	Level Test Log (Component, Component Integration, System, Acceptance)
	Anomaly Report
	Level Interim Test Status Report (Component, Component Integration, System, Acceptance)
	Level Test Report (Component, Component Integration, System, Acceptance)
	Master Test Report
3 Critical	Master Test Plan
	Level Test Plan (Component, Component Integration, System, Acceptance)
	Level Test Design (Component, Component Integration, System, Acceptance)
	Level Test Case (Component, Component Integration, System, Acceptance)
	Level Test Procedure (Component, Component Integration, System, Acceptance)
	Level Test Log (Component, Component Integration, System, Acceptance)
	Anomaly Report
	Level Interim Test Status Report (Component, Component Integration, System, Acceptance)
	Level Test Report (Component, Component Integration, System, Acceptance)
	Master Test Report
2 Marginal	Level Test Plan (Component, Component Integration, System, Acceptance)
	Level Test Design (Component, Component Integration, System, Acceptance)
	Level Tests Case (Component, Component Integration, System, Acceptance)
	Level Test Procedure (Component, Component Integration, System, Acceptance)
	Level Test Log (Component, Component Integration, System, Acceptance)
	Anomaly Report
	Level Interim Test Status Report (Component, Component Integration, System, Acceptance)
	Level Test Report (Component, Component Integration, System, Acceptance)
1 Negligible	Level Test Plan (Component Integration, System)
	Level Test Design (Component Integration, System)
	Level Test Case (Component Integration, System)
	Level Test Procedure (Component Integration, System,)
	Level Test Log (Component Integration, System)
	Anomaly Report (Component Integration, System)
	Level Test Report (Component Integration, System)

For the purpose of ITS testing, it is recommended to agencies considering using the integrity levels to select proper test documents.

2.1.2. Test Processes

The following is excerpted from IEEE 829-2008 with regards to test processes.

5. Test processes

This standard follows the structure defined in IEEE/EIA Std 12207.0-1996 [B21] for describing processes as shown in Figure 3. Each process has one or more supporting activities that are in turn executed by one or more tasks. This is a top-down method for determining which tasks are required. Once each task has been identified, its needed inputs and resultant outputs can be identified. In the case of this standard, the inputs and outputs determine which test documentation are needed.

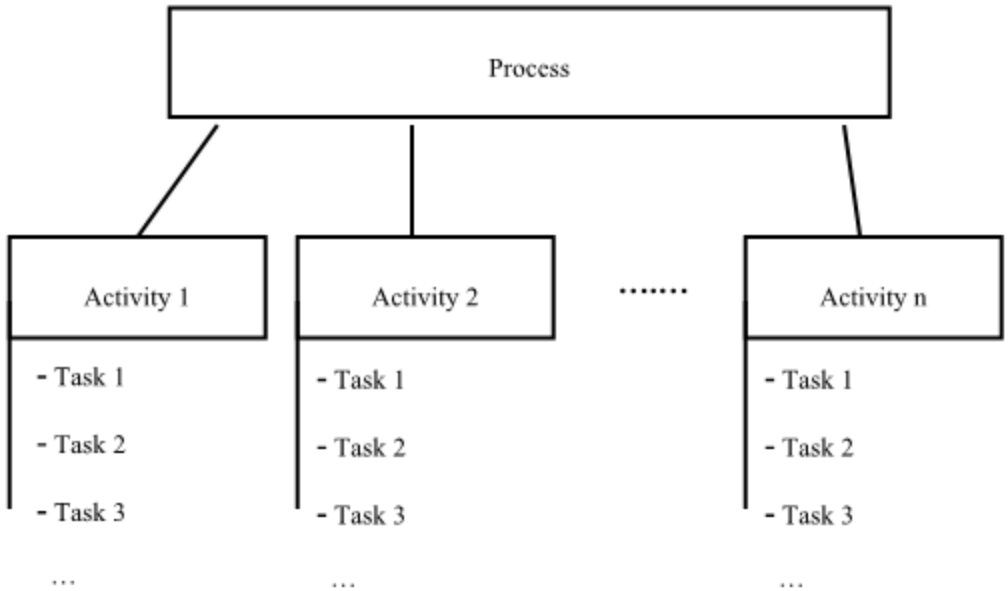


Figure 3—IEEE/EIA Std 12207.0-1996 [B21] Process Definition Infrastructure

Testing needs to support the following IEEE/EIA Std 12207.0-1996 [B21] processes:

— Management (5.1)
— Acquisition (5.2)
— Supply (5.3)
— Development (5.4)
— Operation (5.5)
— Maintenance process (5.6)

Not all software projects include each of the lifecycle processes listed above. To be in conformance with this standard, the test processes need include only those life cycle processes used by the project.

The 2008 version of IEEE-829 changed its focus from being document-oriented in the 1998 version to being processed-oriented. This is a major change and the test documentation required in the 2008 version of IEEE 829 focuses on each of the processes in the system life cycle. This is important for NTCIP testing because the ITS development and implementation processes follow the system engineering process, which is the same approach as the life cycle processes in the IEEE standards.

IEEE 829-2008 specifies the minimum task activities and tasks supporting each of the system life cycle processes. The test effort will then perform the minimum tasks recommended for the assigned system integrity level.

2.1.3. Master Test Plan

The following is excerpted from IEEE 829-2008 with regards to Master Test Plan.

8. Master Test Plan (MTP)

The purpose of the Master Test Plan (MTP) is to provide an overall test planning and test management document for multiple levels of test (either within one project or across multiple projects). In view of the software requirements and the project's (umbrella) quality assurance planning, master test planning as an activity comprises selecting the constituent parts of the project's test effort; setting the objectives for each part; setting the division of labor (time, resources) and the interrelationships between the parts; identifying the risks, assumptions, and standards of workmanship to be considered and accounted for by the parts; defining the test effort's controls; and confirming the applicable objectives set by quality assurance planning. It identifies the integrity level schema and the integrity level selected, the number of levels of test, the overall tasks to be performed, and the documentation requirements.

Master Test Plan Outline (full example)

1. Introduction

1.1. Document identifier

1.2. Scope

1.3. References

1.4. System overview and key features

1.5. Test overview

1.5.1 Organization

1.5.2 Master test schedule

1.5.3 Integrity level schema

1.5.4 Resources summary

1.5.5 Responsibilities

1.5.6 Tools, techniques, methods, and metrics

2. Details of the Master Test Plan

2.1. Test processes including definition of test levels

2.1.1 Process: Management

2.1.1.1 Activity: Management of test effort

2.1.2 Process: Acquisition

2.1.2.1: Activity: Acquisition support test

2.1.3 Process: Supply

2.1.3.1 Activity: Planning test

2.1.4 Process: Development

2.1.4.1 Activity: Concept

2.1.4.2 Activity: Requirements

2.1.4.3 Activity: Design

2.1.4.4 Activity: Implementation

2.1.4.5 Activity: Test

2.1.4.6 Activity: Installation/checkout

2.1.5 Process: Operation

2.1.5.1 Activity: Operational test

2.1.6 Process: Maintenance

2.1.6.1 Activity: Maintenance test

2.2. Test documentation requirements

2.3. Test administration requirements

2.4. Test reporting requirements

3. General

3.1. Glossary

3.2. Document change procedures and history

Note that the MTP identifies test activities and tasks to be performed for each of the test processes, and provides an overview of the test activities and tasks for all system life cycle processes. The number and sequence of levels of test are also included. There may be more levels than listed in IEEE 829. Possible additional test levels may include usability, performance, stress, recovery, and regression. Small systems may have fewer levels of tests.

2.1.4. Level Test Plan

The following is excerpted from IEEE 829-2008 with regards to Level Test Plan.

9. Level Test Plan(s)

Specify for each LTP the scope, approach, resources, and schedule of the testing activities for its specified level of testing. Identify the items being tested, the features to be tested, the testing tasks to be performed, the personnel responsible for each task, and the associated risk(s). In the title of the plan, the word "Level" is replaced by the organization's name for the particular level being documented by the plan (e.g., Component Test Plan, Component Integration Test Plan, System Test Plan, and Acceptance Test Plan).

In most projects, there are different test levels requiring different resources, methods, and environments. As a result, each level is best described in a separate plan. Different Level Test Plans may require different usage of the documentation content topics listed below. Some examples of test levels for the development activity to undertake (from IEEE/EIA Std 12207.0-1996 [B21]) are as follows:

— Each software unit (IEEE Std 1008™-1987 [B9]) and database.

— Integrated units (IEEE Std 1008-1987 [B9]) and components.

— Tests for each software requirement.

— Software qualification testing for all requirements.

— Systems integration: aggregates of other software configuration items, hardware, manual operations, and other systems. It is not unusual for large systems to have multiple levels of integration testing.

— System qualification testing for system requirements.

Level Test Plan Outline (full example)

1. Introduction

1.1. Document identifier

1.2. Scope

1.3. References

1.4. Level in the overall sequence

1.5. Test classes and overall test conditions

2. Details for this level of test plan

2.1 Test items and their identifiers

2.2 Test Traceability Matrix

2.3 Features to be tested

2.4 Features not to be tested

2.5 Approach

2.6 Item pass/fail criteria

2.7 Suspension criteria and resumption requirements

2.8 Test deliverables

3. Test management

3.1 Planned activities and tasks; test progression

3.2 Environment/infrastructure

3.3 Responsibilities and authority

3.4 Interfaces among the parties involved

3.5 Resources and their allocation

3.6 Training

3.7 Schedules, estimates, and costs

3.8 Risk(s) and contingency(s)

4. General

4.1 Quality assurance procedures

4.2 Metrics

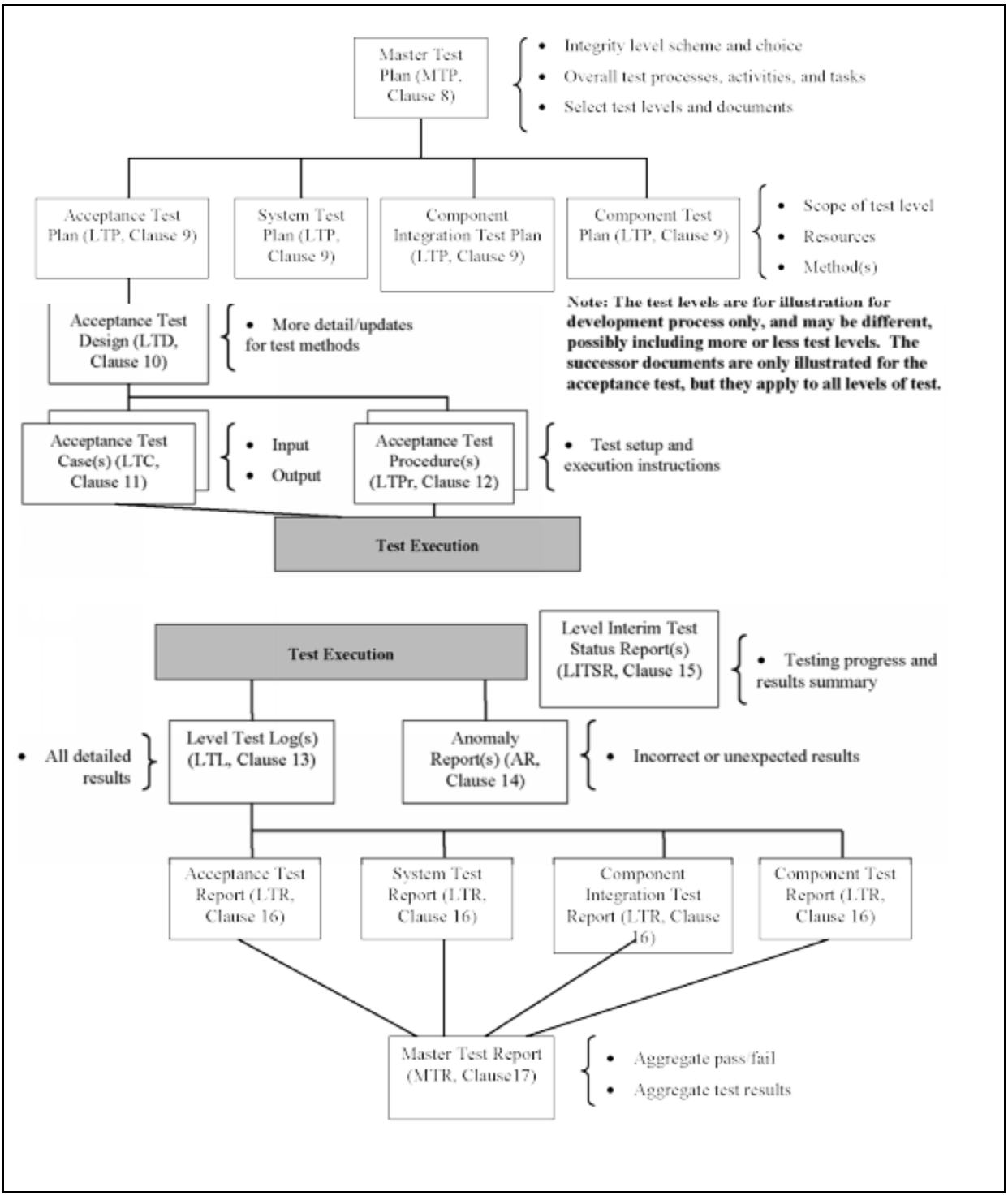
4.3 Test coverage

4.4 Glossary

4.5 Document change procedures and history

2.1.5. Test Documentation

Also called test deliverables in IEEE 829-2008. The following graphic is excerpted from IEEE 829-2008 with regards to test documentation that is to be documented prior to test execution or post-test execution.



2.2. Difference between NTCIP 8007 and IEEE 829

NTCIP 8007 is part of the NTCIP standards family and serves as a main document for NTCIP testing. It defines the testing requirements that can be used to produce test documentation based on IEEE 829, but customizes the testing principles for the NTCIP environment.

The main difference between NTCIP 8007 and IEEE 829 is twofold:

- For simple devices such as CCTV cameras, NTCIP combines test case and test procedure into one document. By doing so, one identification number is used for the test case and its associated test procedure. When another test case needs to be called out in the test procedures, one of the test steps can reference the test case number and execute that test case and its associated test procedure. For complex devices, a better option may be to separate test cases and test procedures as suggested by IEEE 829. Whether they are separate or combined, the test plan will need to clearly identify the test documentation requirements based on the complexity of the test.
- Define a set of keywords used for developing test procedures.

Excerpted below are the keywords defined in NTCIP 8007.

3.1 COMMUNITY NAME IN: The value contained in the 'community' field of the last SNMP Message received from the DUT. See RFC 1157 for additional details related to the community name.

3.2 COMMUNITY NAME OUT: The value that the test application shall use for the 'community' field of the next SNMP Message sent to the DUT. See RFC 1157 for additional details related to the community name. Unless otherwise specified, this value shall be set to the administrator community name as stored in the DUT.

3.3 CONFIGURE: The CONFIGURE keyword shall be used as a predicate to the text of a test step in order to indicate that the text identifies a configurable variable that should be defined by the user prior to performing the test.

3.4 DELAY: The test application and user shall not perform any actions for a defined period of time, which may be measured in time units or by monitoring some event that does not involve any exchange of information over the communications media (e.g., DELAY until the temperature exceeds a threshold). In the later case, the step should also define exception conditions to allow for possibility that the event never happens.

3.5 DYNAMIC OBJECT NUMBER IN: The value contained in the 'Object ID' field of the last STMP Data Packet received from the DUT. See NTCIP 1103 for additional details related to the dynamic object number.

3.6 DYNAMIC OBJECT NUMBER OUT: The value that the test application shall use for the 'Object ID' field of the next STMP Data Packet sent to the DUT. See NTCIP 1103 for additional details related to the dynamic object number.

3.7 ERROR INDEX: The value contained in the 'error-index' field of the last SNMP RESPONSE received from the DUT. See RFC 1157 for additional details related to the error index.

3.8 EXIT: This keyword indicates that the user and test application should terminate the test case without performing any more steps. The keyword by itself does not have any implications as to whether a given test passes or fails.

3.9 GET: The test application shall transmit to the DUT one SNMP Message containing a GetRequest-PDU, per the rules of NTCIP 2301. Each statement using this keyword shall unambiguously reference the value for the 'name' field(s) to be included in the request. The GetRequest-PDU shall include all of the names in its 'variable-bindings' field. See RFC 1157 for additional details related to the GetRequest-PDU.

Unless otherwise indicated, the user or test application shall VERIFY the following, in order:

a) The DUT responds with exactly one SNMP Message that contains a GetResponse-PDU, per the rules of NTCIP 2301; this is the RESPONSE. The DUT may also transmit one or more SNMP Messages that contain a Trap-PDU
b) The value contained in the 'version' field of the RESPONSE equals 0 (version-1)
c) COMMUNITY NAME IN equals COMMUNITY NAME OUT
d) REQUEST ID IN equals REQUEST ID OUT
e) RESPONSE ERROR equals 0 (noError)
f) ERROR INDEX equals 0
g) The 'variable-bindings' field contains the same number of VarBind structures as where contained in the GetRequest-PDU
h) The value of each name field in the RESPONSE equals the value of the name field in the GetRequest-PDU that is in the same ordered position.

3.10 GET-NEXT: The test application shall transmit to the DUT one SNMP Message containing a GetNextRequest-PDU, per the rules of NTCIP 2301. Each statement using this keyword shall unambiguously reference the value for the 'name' field(s) to be included in the request. The GetNextRequest-PDU shall include all of the names in its 'variable-bindings' field. See RFC 1157 for additional details related to the GetNextRequest-PDU.

Unless otherwise indicated, the user or test application shall VERIFY the following, in order:

a) The DUT responds with exactly one SNMP Message that contains a GetResponse-PDU, per the rules of NTCIP 2301; this is the RESPONSE. The DUT may also transmit one or more SNMP Messages that contain a Trap-PDU
b) The value contained in the 'version' field of the RESPONSE equals 0 (version-1)
c) COMMUNITY NAME IN equals COMMUNITY NAME OUT
d) REQUEST ID IN equals REQUEST ID OUT
e) RESPONSE ERROR equals 0 (noError)
f) ERROR INDEX equals 0
g) The 'variable-bindings' field contains the same number of VarBind structures as where contained in the GetNextRequest-PDU
h) The value of each name field in the RESPONSE is greater than the value of the name field in the GetNextRequest-PDU that is in the same ordered position.

3.11 NEXT: A reference to the next sequential test step.

3.12 PERFORM: The user or test application shall perform another test case as a part of this test case. Unless otherwise indicated in the "PERFORM" statement, the user (and test application) shall use the variable values defined when the other test case is performed in a stand-alone fashion.

Example 1: In order to test the illumination features of a sign, it may be necessary to display a message on the sign; however displaying a message on the sign may be a separate requirement that is addressed by a separate test case. For the illumination test, it does not matter what text is displayed, any message will do. Thus, the call to the other test case may look something like:

PERFORM the "Display a Message" test case.

Example 2: In order to test the flashing capabilities of a sign, it may be necessary to display a message with a certain set of specific characteristics. Thus, the flashing test case might have a step to CONFIGURE the flashing_message variable and a separate step elsewhere in the procedure to:

PERFORM the "Display a Message" test case where message = flashing_message.

3.13 PRE-CONDITION: The PRE-CONDITION keyword shall be used as a predicate to the text of a test step in order to indicate that the text provides a textual description of any pre-conditions for the test case.

Pre-conditions are conditions that must be met prior to running a test case. Only one pre-condition shall exist in a test case and it shall always be the first step listed, if present.

3.14 POST-CONDITION: The POST-CONDITION keyword shall be used as a predicate to the text of a test step in order to indicate that the text provides a textual description of any post-conditions for the test case. Post-conditions are conditions that exist after the successful completion of a test case. Only one post-condition shall exist in a test case and it shall always be the last step listed, if present.

3.15 RECORD: The user (or test application) shall record the information indicated by the test step as a part of the test results. This information may be referenced by a later step of the test case (or by a later step of a calling step case).

3.16 REQUEST ID OUT: The value that the test application shall use for the 'request-id' field of the next SNMP Message sent to the DUT. See RFC 1157 for additional details related to the request id. Unless otherwise specified, this value shall start at an arbitrary value and shall increment by one for each SNMP Message sent by the test application.

3.17 REQUEST ID IN: The value contained in the 'request-id' field of the last SNMP Message received from the DUT. See RFC 1157 for additional details related to the request id.

3.18 RESPONSE: The last SNMP Message containing a GetResponse-PDU received from the DUT.

NOTE—SNMP uses the same message structure to respond to a GetRequest-PDU, a GetNextRequest-PDU, and a SetRequest-PDU. SNMP calls this message structure a 'GetResponse-PDU', even though it may be a response to a SetRequest-PDU.

3.19 RESPONSE ERROR: The value contained in the 'error-status' field of the last SNMP Message received from the DUT. See RFC 1157 for additional details related to the error status.

3.20 RESPONSE OID: The value contained in the indicated 'name' field of the last SNMP Message received from the DUT. Each statement using this keyword shall unambiguously reference which name field is to be considered, if the response is expected to contain multiple name fields.

3.21 RESPONSE VALUE: The value contained in the indicated 'value' field of the last SNMP Message received from the DUT. Each statement using this keyword shall unambiguously reference which value field is to be considered, if the response is expected to contain multiple value fields.

3.22 RESTART-POINT: The RESTART-POINT keyword shall be used as a predicate to the text of a test step in order to indicate that the step is a point in the procedure that the test can be restarted if the test had to stop for any reason (e.g., due to a failure in the DUT, a failure by the test application, a break taken by the user, etc.). The test step shall identify the actions and conditions necessary to restart the procedure at the given location. When normally performing the test, the RESTART-POINT step should be ignored.

3.23 SET: The test application shall transmit to the DUT one SNMP Message containing a SetRequest-PDU, per the rules of NTCIP 2301. Each statement using this keyword shall unambiguously reference the value for the 'name' field(s) to be included in the request. The statement shall also indicate the value of the 'value' field associated with each 'name' field. Unless otherwise indicated, the value will be encoded according to the SYNTAX of the associated object. The SetRequest-PDU shall include all of the names and values, with their indicated associations in its 'variable-bindings' field. See RFC 1157 for additional details related to the SetRequest-PDU.

Unless otherwise indicated, the user or test application shall VERIFY that:

a) The DUT responds with exactly one SNMP Message that contains a GetResponse-PDU, per the rules of NTCIP 2301; this is the RESPONSE. The DUT may also respond with one or more SNMP Messages that contain a Trap-PDU
b) The value contained in the 'version' field of the RESPONSE equals 0 (version-1)
c) COMMUNITY NAME IN equals COMMUNITY NAME OUT
d) REQUEST ID IN equals REQUEST ID OUT
e) RESPONSE ERROR equals 0 (noError)
f) ERROR INDEX equals 0
g) The 'variable-bindings' field contains the same number of VarBind structures as where contained in the SetRequest-PDU
h) The value of each name field in the RESPONSE equals the value of the name field in the SetRequest-PDU that is in the same ordered position.
i) The value of each value field in the RESPONSE equals the value of the value field in the SetRequest-PDU that is in the same ordered position.

3.24 SET-UP: The SET-UP keyword shall be used as a predicate to the text of a test step in order to indicate that the test step is a preparatory step in order to set up an environment in which the actual test can take place. If the user and/or test application is unsuccessful in performing the test step, the user (and/or test application) shall EXIT the test case and the test case will neither pass nor fail. The user should then investigate the problem in performing the step and restart the test.

3.25 STMP-GET: The test application shall transmit to the DUT one STMP Data Packet containing a STMP-GetRequest-PDU, per the rules of NTCIP 2301. Each statement using this keyword shall unambiguously reference the dynamic object number to be included in the request. The STMP-GetRequest-PDU shall include the dynamic object number in the request. See NTCIP 1103 for additional details related to the STMP-GetRequest-PDU.

Unless otherwise indicated, the user or test application shall VERIFY the following, in order:

a) The DUT responds with exactly one STMP Data Packet that contains a STMP-GetResponse-PDU, per the rules of NTCIP 2301; this is the RESPONSE. The DUT may also transmit one or more SNMP Messages that contain a Trap-PDU
b) DYNAMIC OBJECT NUMBER IN equals DYNAMIC OBJECT NUMBER OUT
c) The 'data' field can be properly parsed given the current dynamic object definition

3.26 STMP-SET: The test application shall transmit to the DUT one STMP Data Packet containing a STMP-SetRequest-PDU, per the rules of NTCIP 2301. Each statement using this keyword shall unambiguously reference the dynamic object number to be included in the request. The statement shall also indicate the value of each Referenced Object. Unless otherwise indicated, the value will be encoded according to the SYNTAX of the associated Referenced Object. The STMP-SetRequest-PDU shall include the dynamic object number and the values of the Referenced Objects. See NTCIP 1103 for additional details related to the STMP-SetRequest-PDU.

Unless otherwise indicated, the user or test application shall VERIFY that:

a) The DUT responds with exactly one STMP Data Packet that contains a STMP-SetResponse-PDU, per the rules of NTCIP 2301; this is the RESPONSE. The DUT may also respond with one or more SNMP Messages that contain a Trap-PDU
b) DYNAMIC OBJECT NUMBER IN equals DYNAMIC OBJECT NUMBER OUT
c) The PDU Information field is empty.

3.27 VERIFY: The user or test application shall evaluate the expression that follows this keyword. Each statement using this keyword shall contain an unambiguous expression that will always evaluate to either true or false without subjective or qualitative judgments by the Tester.

If the result is true:

a) The verification step shall pass, and
b) The test shall continue to the next step, unless otherwise indicated in the test case.

Otherwise, if the result is false:

a) The verification step shall fail,
b) The test case shall fail, and
c) The test case shall EXIT, unless otherwise indicated in the test case

NOTE—While criteria are often stated in exact terms (e.g., "The response shall be '3'"; or, "The sign shall display 'TEST'"; etc.), it may also be the case that criteria may be stated as ranges or thresholds (e.g., "The response shall be between '2' and '16' inclusive"; or, "The response shall be '3' or greater"; etc.). Each approach is valid and should be considered in the construction of a test case.

3. Case Studies

Included below are the steps to develop a CCTV Unit Test Plan based on the level test plan outline provided by IEEE 829-2008.

Outline provided by IEEE 829-2008		**CCTV Unit Test Plan**
1	**Introduction**	
1.1	Document identifier	Unique ID for the unit testing
1.2	Scope	Describe the scope of the CCTV Unit Test. The details for each test cameras, and their revision history both hardware and software. Describe the inclusions, exclusions, assumptions, and limitations. It is important to define clearly the limits of the test effort.
1.3	References	List all of the applicable reference documents.
1.4	Level in the overall sequence	• Show the CCTV unit testing in the overall test hierarchy such as CCTV subsystem and system acceptance testing, overall communication network testing with other ITS devices, etc. • A diagram will be helpful
1.5	Test classes and overall test conditions	• Describe the attributes of the CCTV camera unit test – Pan-Tilt-Zoom (PTZ), presets, focus, iris, alarms, zones, etc. • Detailed descriptions should be provided how these features will be tested in groups, i.e. test classes, and their associated test conditions for each group. • Test conditions o Positive testing – valid input values o Negative testing – invalid values for

		error processing o Boundary testing – input values just above, just below, and just on each limit
2	**Details for this level of test plan**	
2.1	Test items and their identifiers	• CCTV camera model, make, firmware version, etc. • Reference to the CCTV user manual, operations guide, installation guide, etc. • Transfer from other environments to the test environment. For example, from factory to test lab, or agency's facility if the test will be performed by the agency.
2.2	Test Traceability Matrix	• Provide a list of requirements and corresponding test cases or procedures - Requirements to Test Case Matrix defined in NTCIP 8007 • Or a reference to a larger Test Traceability Matrix for all levels of test, e.g. the matrix includes unit testing, subsystem testing, system testing, operation testing, etc. and traces to multiple levels of life cycle documentation products. It may include both forward and backward tracing.
2.3	Features to be tested	CCTV features based on project-specific requirements – Requirements Traceability Matrix (RTM). These camera features include NTCIP objects identified in the RTM such as Pan-Tilt-Zoom (PTZ), presets, focus, iris, alarms, zones, etc. Refer to A317b for example RTM.
2.4	Features not to be tested	Remote control functions, for example, TMC control, may not be tested during unit test (may be part of the CCTV subsystem testing)
2.5	Approach	• Overall approach for the unit testing • Commonly combined in a Test Matrix with features to be tested • Test methods – black box, white box, analysis, inspection o Black box: The test inputs can be generated and the outputs captured and completely evaluated from the outside of a test item; i.e., test cases

		are developed from the test item specification, only without looking at the code or design. o White box: Considers the internal structure of the software (e.g., attempts to reach all of the code). Commonly requires some kind of test support software. o Analysis: Just viewing the outputs cannot confirm that the test executed successfully; some kind of additional computations, simulations, studies, and so on will be required. o Inspection: This is a static test; the code or documentation is read and examined without being executed.
2.6	Item pass/fail criteria	Specify the criteria to be used to determine whether each test item has passed or failed testing. This is commonly based on the number of anomalies found in specific severity category(s).
2.7	Suspension criteria and resumption requirements	Specify the criteria used to suspend all or a portion of the testing activity on the test items associated with this plan. Specify the testing activities that must be repeated when testing is resumed.
2.8	Test deliverables	Identify all information that is to be delivered by the test activity (documents, data, etc.). The following documents may be included: – Unit Test Plan – Unit Test Design – Unit Test Cases – Unit Test Procedures – Unit Test Logs – Anomaly Reports – Unit Interim Test Status Reports – Unit Test Report – Master Test Report Test input data and test output data may be identified as deliverables. Test tools may also be included.
3	**Test management**	
3.1	Planned activities and tasks; test progression	Identify the set of tasks necessary to prepare for and perform testing. Identify all inter-task

		dependencies. Identify any significant constraints such as test item availability, testing resource availability, and deadlines.
3.2	Environment/infrastructure	Specify both the necessary and the desired properties of the test environment and any relevant test data. This may include the physical characteristics of the facilities, including the hardware, the off-the shelf software, the test support tools and databases, personnel (identifying their organizations as appropriate), and anything else needed to support the test. It includes the environment for setup before the testing, execution during the testing (including data capture), and any post-testing activities (e.g., data reduction and analysis). Also specify the level of security that must be provided for, and any safety issues related to, the testing facilities, software, and any proprietary components. It may include externally provided content topics (possibly provided by third parties) including systems and/or subsystems. Identify the source(s) for all of these needs.
3.3	Responsibilities and authority	Identify the individuals or groups responsible for managing, designing, preparing, executing, witnessing, and checking results of this level of testing, and for resolving the anomalies found.
3.4	Interfaces among the parties involved	Describe the means and the contents of communication between the individuals and groups. A figure that illustrates the flow of information and data may be included.
3.5	Resources and their allocation	Delineate any additional required resources that are not already documented by other parts of the plan, which includes both internal and external resources (such as outside test resources, e.g., test labs, outsourcing, etc.).
3.6	Training	Specify test training needs by skill level. Identify training options for providing necessary skills.
3.7	Schedules, estimates, and costs	Include test milestones identified in the software or system project schedule as well as all test item transmittal events. Estimate the time required to do each testing task. Specify the schedule for each testing task and test milestone. For each testing

		resource (i.e., facilities, tools, and staff), specify its periods of use.
3.8	Risk(s) and contingency(s)	Identify the risk issues that may adversely impact successful completion of the planned level testing activities. Specify potential impact(s) of each risk, with contingency plan(s) for mitigating or avoiding the risk.
4	**General**	
4.1	Quality assurance procedures	Identify the means by which the quality of testing processes and products will be assured. Include or reference anomaly tracking and resolution procedures. The quality assurance information may be described in a Quality Assurance Plan.
4.2	Metrics	Identify the specific measures that will be collected, analyzed, and reported.
4.3	Test coverage	Specify the requirement(s) for test coverage. Test coverage is an indication of the degree to which the test item has been reached or "covered" by the test cases, including both the breadth and depth. Test coverage is often expressed in terms of percentage of code tested, and software or system validation test coverage can be a percentage of requirements tested. There is a need for specification of coverage or some other method for ensuring sufficiency of testing.
4.4	Glossary	Provide an alphabetical list of terms that may require definition with their corresponding definitions. This includes acronyms.
4.5	Document change procedures and history	Specify the means for identifying, approving, implementing, and recording changes to the test plan.

4. Samples/Examples in PowerPoint Slides

Below is the summary of the examples provided in the PowerPoint slides for the test documents including RTCTM, Test Cases, and Test procedures. This example is based on the Requirement Traceability Matrix (RTM) provided in the previous module A317b.

4.1. Requirements Test Case Traceability Matrix (RTCTM)

Requirement		Test Case	
ID	Title	ID	Title
3.3.2	**Camera Control**		
3.3.2.1	Preset Go to Position		
3.3.2.2	Move Camera to a Stored Position		
		C3.01	Preset Position
3.3.2.3	Zoom Operation		
		C3.05	Delta Zoom Motion
		C3.06	Absolute Zoom Motion
		C3.07	Continuous Zoom Motion with Timeout
		C3.08	Continuous Zoom Motion with Stop
…	…		
		…	…
3.3.2.4	Camera Position Horizontally (Pan)		
		C3.11	Delta Pan Motion
		C3.12	Absolute Pan Motion
		C3.13	Continuous Pan Motion with Timeout
		C3.14	Continuous Pan Motion with Stop

4.2. Test Case C3.01 and associated Test Procedures

Test Case: C3.01	**Title:**	Preset Position	
	Description:	This test case stores and moves the camera to preset positions	
	Variables:	Max_Preset	From Project Requirements
		Preset_Speed	From the Test Plan
		Preset_Pan_Position1	From the Test Plan
		Preset_Pan_Position2	From the Test Plan

		Preset_Tilt_Position1	From the Test Plan
		Preset_Tilt_Position2	From the Test Plan
	Pass/Fail Criteria:	The Device Under Test (DUT) shall pass every verification step included within the Test Case in order to pass the Test Case	

Step	**Test Procedure**	**Results**
1	CONFIGURE: Determine a preset position for the camera between 0 and rangeMaximumPreset.0 (per the project requirement). RECORD this information as: >>Max_Preset	
2	SET-UP: if Max_Preset is less than 2, then EXIT	
3	GET the following object: >>rangeMaximumPreset.0	Pass / Fail
4	SET-UP: VERIFY that the RESPONSE VALUE is equal to Max_Preset; otherwise, EXIT.	
5	CONFIGURE: Determine the following value from the test plan. RECORD the information as: >>Preset_Speed >>Preset_Pan_Position1 >>Preset_Pan_Position2 >>Preset_Tilt_Position1 >>Preset_Tilt_Position2	
6	SET the following objects to the values shown: >>positionPan.0 = 02 Preset_Speed Preset_Pan_Position1 >>postionTilt.0 = 02 Preset_Speed Preset_Tilt_Position1	Pass / Fail
7	VERIFY that camera is in position 1.	Pass / Fail
8	SET presetStorePosition.0 to 1	Pass / Fail
9	SET the following objects to the values shown: >>positionPan.0 = 02 Preset_Speed Preset_Pan_Position2 >>postionTilt.0 = 02 Preset_Speed Preset_Tilt_Position2	Pass / Fail
10	VERIFY that camera moved to position 2.	Pass / Fail
11	SET presetStorePosition.0 to 2	Pass / Fail
12	SET presetGotoPosition.0 to 1	Pass / Fail
13	VERIFY that camera moved in position 1.	Pass / Fail

14	GET presetPositionQuery.0	Pass / Fail
15	VERIFY that RESPONSE VALUE = 1	Pass / Fail
16	SET presetGotoPosition.0 to 2	Pass / Fail
17	VERIFY that camera moved in position 2.	Pass / Fail
18	GET presetPositionQuery.0	Pass / Fail
19	VERIFY that RESPONSE VALUE = 2	Pass / Fail
Test Case Results		
Tested By:		Date Tested: Pass / Fail
Test Case Notes:		

5. Glossary

The following acronyms are used in this module.

Acronym	Definition
AR	Anomaly Report
CCTV	Close Circuit Television
DUT	Device Under Test
LITSR	Level Interim Test Status Report
LTC	Level Test Case
LTD	Level Test Design
LTL	Level Test Log
LTP	Level Test Plan
LTPr	Level Test Procedure
LTR	Level Test Report
MTP	Master Test Plan
MTR	Master Test Report
PPP	Point-to-Point Protocol
PMPP	Point-to-Multi-Point Protocol
PRL	Protocol Requirement List
PTZ	Pan-Tilt-Zoom
RTCTM	Requirements Test Cast Traceability Matrix
RTM	Requirements Traceability Matrix
SEP	System Engineering Process
SNMP	Simple Network Management Protocol

Acronym	Definition
TCP/IP	Transmission Control Protocol/Internet Protocol

The following terms and definitions are used in this module. All definitions are excerpted from IEEE 829-2008.

Term	Definition
integrity level	(A) The degree to which software complies or must comply with a set of stakeholder-selected software and/or software-based system characteristics (e.g., software complexity, risk assessment, safety level, security level, desired performance, reliability, or cost), defined to reflect the importance of the software to its stakeholders. (B) A symbolic value representing this degree of compliance within an integrity level scheme.
integrity level scheme	A set of system characteristics (such as complexity, risk, safety level, security level, desired performance, reliability, and/or cost) selected as important to stakeholders, and arranged into discrete levels of performance or compliance (integrity levels), to help define the level of quality control to be applied in developing and/or delivering the software.
life cycle processes	A set of interrelated activities that result in the development or assessment of software products. Each activity consists of tasks. The life cycle processes may overlap one another.
minimum tasks	Those tasks required for the integrity level assigned to the software to be tested.
test approach	A particular method that will be employed to pick the particular test case values. This may vary in specificity from very general (e.g., black box or white box) to very specific (e.g., minimum and maximum boundary values).
test case	(A) A set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement. (B) Documentation specifying inputs, predicted results, and a set of execution conditions for a test item. (adopted from IEEE Std 610.12-1990 [B2])
test class	A designated grouping of test cases.
test design	Documentation specifying the details of the test approach for a software feature or combination of software features and identifying the associated tests (commonly including the organization of the tests into groups). (adopted from IEEE Std 610.12-1990 [B2])
test effort	The activity of performing one or more testing tasks.

Term	Definition
test level	A separate test effort that has its own documentation and resources (e.g., component, component integration, system, and acceptance).
testing	(A) An activity in which a system or component is executed under specified conditions, the results are observed or recorded, and an evaluation is made of some aspect of the system or component. (B) To conduct an activity as in (A).
test item	A software or system item that is an object of testing.
test plan	(A) A document describing the scope, approach, resources, and schedule of intended test activities. It identifies test items, the features to be tested, the testing tasks, who will do each task, and any risks requiring contingency planning. (B) A document that describes the technical and management approach to be followed for testing a system or component. Typical contents identify the items to be tested, tasks to be performed, responsibilities, schedules, and required resources for the testing activity. (adopted from IEEE Std 610.12-1990 [B2]) The document may be a Master Test Plan or a Level Test Plan.
test procedure	(A) Detailed instructions for the setup, execution, and evaluation of results for a given test case. (B) A document containing a set of associated instructions as in (A). (C) Documentation that specifies a sequence of actions for the execution of a test. (adopted from IEEE Std 982.1TM-2005 [B7])
validation	(A) The process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements. (adopted from IEEE Std 610.12-1990 [B3]) (B) The process of providing evidence that the software and its associated products satisfy system requirements allocated to software at the end of each life cycle activity, solve the right problem (e.g., correctly model physical laws, implement business rules, or use the proper system assumptions), and satisfy intended use and user needs.
verification	(A) The process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase. (adopted from IEEE Std 610.12-1990 [B3]) (B) The process of providing objective evidence that the software and its associated products comply with requirements (e.g., for correctness, completeness, consistency, and accuracy) for all life cycle activities during each life cycle process (acquisition, supply, development, operation, and maintenance), satisfy standards, practices, and conventions during life cycle processes, and successfully complete each life cycle activity and satisfy all the criteria for initiating succeeding life cycle activities (e.g., building the software correctly).

6. References

- IEEE 829, IEEE Standard for Software Test Documentation, IEEE, 1998 or 2008 version.
- NTCIP 8007:2008, National Transportation Communications for ITS Protocol: Testing and Conformity Assessment Documentation within NTCIP Standards Publications, v01, AASHTO/ITE/NEMA, May 2008.
- NTCIP 9001 Version v04, National Transportation Communications for ITS Protocol, The NTCIP Guide, AASHTO/ITE/NEMA, July 2009.
- NTCIP 1205 v01.08, National Transportation Communications for ITS Protocol: Object Definition for Closed Circuit Television (CCTV) Camera Control, AASHTO/ITE/NEMA, December 2001, or Revision Amendment A1 v08a, November 2004.
- Systems Engineering Guidebook for Intelligent Transportation Systems Version 3.0, United States Department of Transportation, November 2009.

7. Study Questions

These are questions from the presentation.

1) **Which of the following Statements is <u>not</u> correct?**

a) Requirements can be verified by inspection, demonstration, analysis and testing of the system products.
b) The testing process provides an objective assessment of system products throughout the system life cycle.
c) Test documentation needs to be prepared only at the completion of system development
d) Development of test plans can begin as soon as the system ConOps is being developed.

2) **Which of the following is included in LTP but not in MTP?**

a) Test scope
b) Test processes
c) Test resources and responsibilities
d) Test Traceability Matrix

3) **Which of the following is part of test documentation?**

a) Test Data
b) Level Test Plans
c) Requirement Test Case Traceability Matrix
d) All of the above

4) **Which of the following test documents is included in NTCIP 1205?**

a) Protocol Requirements List (PRL)
b) Requirements Traceability Matrix (RTM)
c) Requirements Test Cast Traceability Matrix (RTCTM)
d) None of the above

5) **Which of the following statements is correct?**

a) Data analyzer is an active test tool and can be used to respond to the DUT's request
b) All possible permutations and combinations of valid input values need to be tested
c) Performing boundary analysis is not necessary during NTCIP testing
d) None of the above