# A203 Writing Requirements When ITS Standards Do Not Have SEP Content

STANDARDS ITS TRAINING

STUDENT SUPPLEMENT

RITA Intelligent Transportation Systems Joint Program Office

# A203: Writing Requirements When ITS Standards Do Not Have SEP Content

## Table of Contents

## PURPOSE

This is the supplemental information for the Professional Capacity Building (PCB) Module A203. In some cases, additional information is included within the supplement and, references are provided for more in-depth study.

Module A203 trains how to write requirements that support a functional decomposition process and how to verify said requirements at the system, subsystem, and communications standard level. In A203, participants will be able to:

1) Understand that requirements development is a process
2) Avoid pitfalls when writing requirements
3) Write requirements when an ITS communication standard does not have SEP information
4) Use traceability matrices as tools for requirements development

The remainder of this supplement is organized based upon these learning objectives and concludes with a section on references.

## REQUIREMENTS DEVELOPMENT IS A PROCESS

Requirements development is *recursive*. Recall from the previous course how the system could be decomposed into subsystems. The process of defining Functional Requirements, Performance Requirements, Non-Functional Requirements, and Constraints can be applied recursively over the requirements expressed at higher *levels*. See Figure 1.

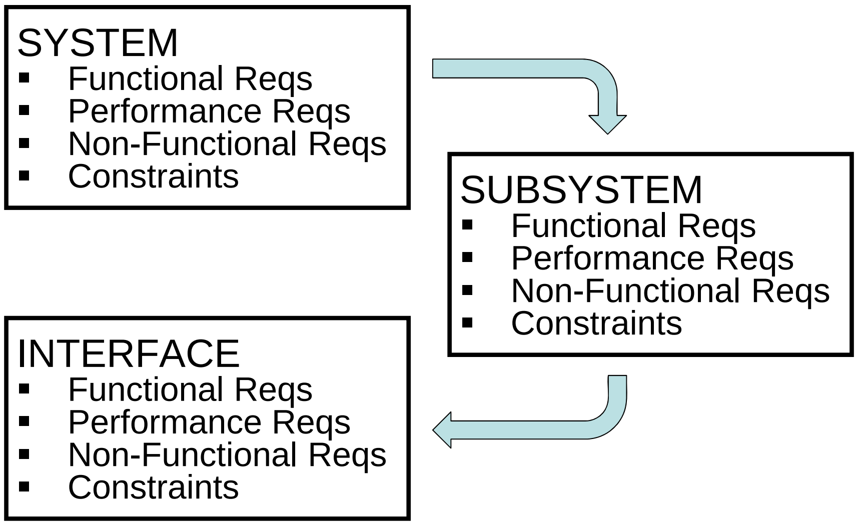


**Figure 1: Developing requirements recursively over previous levels of requirements.**

Requirements development is *iterative*. It is often the case that needs and requirements established at one stage of development must be revised or rephrased after later stages of development. Sometimes a project team will know that a requirement will need to be rephrased later. This is a case of "sufficient for purpose" to allow a project to advance to the next stage of development. See Figure 2.

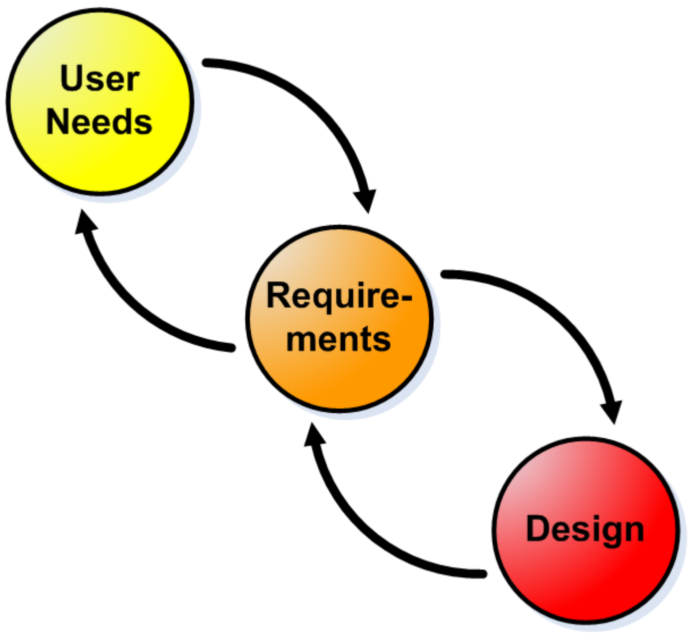


**Figure 2: Developing requirements iteratively by revising and rephrasing after later stages of development.**

Requirements development is a *process of discovery.* All inquiry circles have periods of action and reflection: action to move forward and reflection to consider whether the work is complete, going in the right direction, or consider a redirection. See Figure 3.

When we get to the subsystem and interface levels, we find that the standards can be a source for discovery.

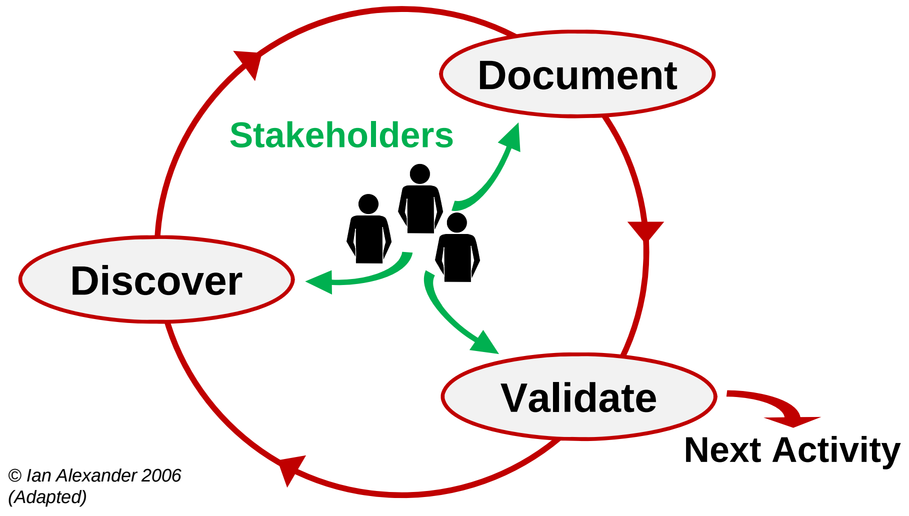


**Figure 3: Developing requirements is a process of discovery.**

## AVOIDING PITFALLS WHEN WRITING REQUIREMENTS

**Definition of a Well Formed Requirement**
A statement of system functionality (a capability) that can be validated, and that must be met or possessed by a system to solve a customer problem or to achieve a customer objective, and is qualified by measurable conditions and bounded by constraints. (IEEE Std 1233, 1998 IEEE Guide for Developing System Requirements)

STRUCTURE OF A WELL FORMED REQUIREMENT

The Form: [Actor] [Action] [Target] [Constraint] [Localization]

Where:

| | |
|---|---|
| Actor | Identifies who or what that does the action |
| Action | Identifies what is to happen |
| Target | Identifies who or what receives the action |
| Constraint | Identifies how to measure success or failure of the requirement |
| Localization | Identifies the circumstances under which the requirement applies |

*Localization and constraint portions are important but not all requirements will have both*

Example: The system [Actor] shall generate [Action] event reports [Target] containing the following information [Constraint] on a scheduled interval [localization]

*If a requirement can't be stated in this simple format, you probably need to define the functionality using multiple requirements*

**Characteristics of Well Formed Requirements**

- Necessary
  - Must be useful and traceable to needs.
- Concise
  - Minimal, understandable and expressed in a declarative language (e.g. "shall statements").
- Attainable
  - Realistic to achieve within available resources and time.
- Standalone
  - The requirement is stated completely in one place. Not grouped.
- Consistent
  - Does not contradict itself, or any other stated requirement.
- Unambiguous
  - Susceptible to only one interpretation.
- Verifiable
  - Must be able to determine that the requirement has been met through one of four possible methods: inspection, analysis, demonstration, or test.

Some pitfalls to avoid when building well-formed requirements are as follows:

**Pitfall #1** - *Design and implementation.* There is a tendency on the part of analysts and customers who are defining requirements to include design and implementation decisions along with the requirements 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 design and implementation.

**Pitfall #2** - *Overspecified.* Requirements that express an exact commercial system set or a system that can be bought rather than made (these are not an expression of what the system should do). Requirements that state tolerances for items deep within the conceptual system (frequently stated as error requirements at very low levels); Requirements that implement solutions (requirements state "what" is needed).

**Pitfall #3** - *Overconstrained.* Requirements with unnecessary constraints. (For example, if a system must be able to run on rechargeable batteries, a derived requirement might be that the time to recharge should be less than 3 hours. If this time is too restrictive and a 12 hour recharge time is sufficient, potential solutions are eliminated.)

**Pitfall #4** - *Unbounded.* Requirements making relative statements. (These requirements cannot be verified and may only need to be restated. For example, the requirement to "minimize noise" may be restated as "noise levels should not exceed..."). Requirements that are open-ended (frequently stated as "including, but not limited to..." or lists ending in "etc."). Requirements making subjective or vague statements (frequently contain terms such as "user friendly," "quick response time," or "cost effective").

**Pitfall #5** - *Assumptive.* Requirements based on undocumented assumptions. The assumption should be documented as well as the requirement. Requirements based on the assumption that a particular standard or system undergoing development will reach completion. The assumption and an alternative requirement should be documented.

## WRITING REQUIREMENTS WHEN AN ITS COMMUNICATION STANDARD DOES NOT HAVE SEP INFORMATION

It is important to understand how the design is specified in the standard. For device standards this is generally expressed as Objects and Dialogs. Objects are organized in a Management Information Base (MIB):

- Defines the data elements (objects) of the ASC device that are covered by the standard
- Written in Abstract Syntax Notation (ASN.1) notation
- Resource/reference for those familiar with the device
- Ineffective for a novice to learn a device by reading a MIB

Example Object Definition (Data Element Definition) in ASN.1 Notation

```
phaseWalk OBJECT-TYPE
  SYNTAX INTEGER (0..255)
  ACCESS read-write
  STATUS optional
  DESCRIPTION
    "<Definition> Phase Walk Parameter in
     seconds. This shall control the amount
     of time the Walk indication shall be
     displayed."
  REFERENCE
    "NEMA TS 2 Clause 3.5.3.1 & 3.5.3.2.2.a"
::= { phaseEntry 2 }
```

Dialogs – A series of communication exchanges to execute some feature or task. A dialog may be expressed in words or as a sequence diagram.

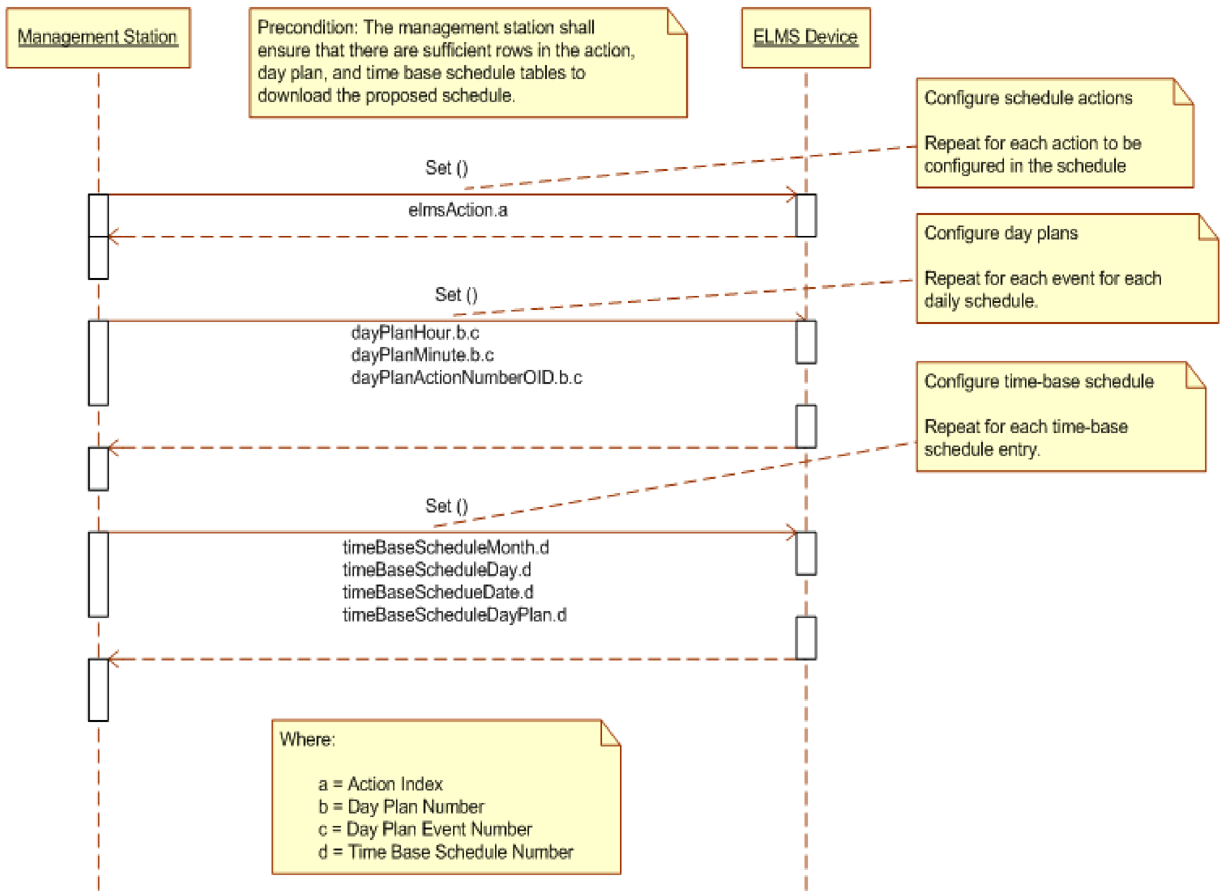
**Example #1: A Dialog Expressed in Words**

4.3.3.5 Retrieve Sensor Zone Class Labels
The standardized dialog for a management station to retrieve the class labels for a sensor zone shall be as follows:

1) (Precondition) The management station shall be aware that the sensorZoneNumber must be less than or equal to the maxSensorZones. The TSS must support sampling features
2) The management station shall GET zoneSequenceEntry:numSensorZoneClass.x
3) sampleZoneClass = zoneSequenceEntry:numSensorZoneClass.x from Step 2
4) The management station shall GET zoneClassLabel.y.x
5) If zoneClassEntry is greater than 0, then zoneClassEntry = zoneClassEntry - 1 and go to Step 4
6) Retrieval of class labels for this sensor zone is complete

Where:
x = zoneClassEntry
y = sampleZoneClass

## Example #2: A Dialog Expressed as a Sequence Diagram


Management Station
Precondition: The management station shall ensure that there are sufficient rows in the action, day plan, and time base schedule tables to download the proposed schedule.
ELMS Device
Configure schedule actions
Repeat for each action to be configured in the schedule
Set ()
elmsAction.a
Configure day plans
Repeat for each event for each daily schedule.
Set ()
dayPlanHour.b.c
dayPlanMinute.b.c
dayPlanActionNumberOID.b.c
Configure time-base schedule
Repeat for each time-base schedule entry.
Set ()
timeBaseScheduleMonth.d
timeBaseScheduleDay.d
timeBaseSchedueDate.d
timeBaseScheduleDayPlan.d
Where:
a = Action Index
b = Day Plan Number
c = Day Plan Event Number
d = Time Base Schedule Number

### Applying Discover-Document-Validate to the ASC Standard

- Not trying to write requirements for a standard
- “Discovering” interface requirements that support system requirements
- “Documenting” what we find
- “Validating” using the techniques we have learned
- Get to this stage through decomposition of the architecture and the requirements
- Need to document requirements that require a series of exchanges (dialogs)
- Capturing performance and constraint criteria
- See also the NTCIP Guide

## TRACEABILITY MATRICES AS TOOLS FOR REQUIREMENTS DEVELOPMENT

Traceability for Verifying Requirements

- A tool used to help verify completeness and correctness
- Every need must be addressed by at least one requirement
- Every requirement must trace to at least one need
- Any need that is not addressed by at least one requirement means:
  - A requirement was missed or
  - The user need must be reevaluated
- Every requirement that does not address at least one need means:
  - The requirement must be reevaluated or
  - A user need was missed
- Every aspect of each user need should be addressed in requirements

There are numerous types of traceability matrices used in ITS Standards from the simple to the complex. Below are some examples.

**Example #1:** Needs-To-Requirements Traceability Matrix (NRTM). User needs are traced to requirements.

| User Need ID | User Need | Req ID | Requirement |
|---|---|---|---|
| 2.5.2.6 | Manage Real-Time Clock | 3.4.1.4.1 | Get Date and Time |
| | | 3.4.1.4.2 | Get Daylight Saving Time Mode |
| | | 3.4.1.4.3 | Set Date and Time |
| | | 3.4.1.4.4 | Set Daylight Saving Time Mode |

**Example #2**: Requirements Traceability Matrix (RTM) traces requirements to the design elements of the standard. In this case, the design elements include the dialog (sequence of data exchanges) and objects (data elements) that are used to fulfill the requirement.

| Requirement ID | Require-ment | Dialog ID | Dialog | Object ID | Object |
|---|---|---|---|---|---|
| 3.4.1.3.8 | Execute Pending Configuration | | | | |
| | | 4.3.1.3 | Execute Pending Configuration Change | | |
| | | | | 5.2.1 | sensorSystemReset |
| | | | | 5.2.2 | sensorSystemStatus |
| 3.4.1.3.9 | Abort Pending Configuration | | | | |
| | | 4.3.1.4 | Abort Pending Configuration | | |
| | | | | 5.2.1 | sensorSystemReset |
| | | | | 5.2.2 | sensorSystemStatus |
| 3.4.1.3.10 | Validate Pending Configuration | | | | |
| | | 4.3.1.5 | Validate a Pending Configuration | | |
| | | | | 5.2.1 | sensorSystemReset |
| | | | | 5.2.2 | sensorSystemStatus |

**Example #3:** Protocol Requirements List (PRL) Traces User Needs to Requirements but with additional information. It indicates whether the requirement is mandatory or optional within the standard or if there is some conditional conformance. It then provides a checklist on whether users want to include the requirement in their project. The PRL also provides for other information to be added for further specification or if instructive information is necessary.

| User Need Section Number | User Need | FR Section Number | Functional Requirement | Conformance | Support / Project Requirement | Additional Specifications |
|---|---|---|---|---|---|---|
| 2.5.2.1 | Reset the TSS | | | | | |
| | | 3.4.1.3.1 | Restart the TSS | M | Yes | |
| | | 3.4.1.3.2 | Reinitialize User Settings | M | Yes | |
| | | 3.4.1.3.3 | Restore Factory Defaults | M | Yes | |
| | | 3.4.1.3.4. | Retune | M | Yes | |
| | | 3.4.1.3.8 | Execute Pending Configuration | O.1 | Yes/No | |
| | | 3.4.1.3.9 | Abort Pending Configuration | O.1 | Yes/No | |
| | | 3.4.1.3.10 | Validate Pending Configuration | O.1 | Yes/No | |
| 2.5.2.2 | Initiate Sensor Diagnostics | | | | | |
| | | 3.4.1.3.6 | Short Diagnostics | M | Yes | |
| | | 3.4.1.3.7 | Full Diagnostics | M | Yes | |

## NTCIP Device Standards With Systems Engineering Content

| Doc # | NTCIP Device Standards With Systems Engineering Content |
|---|---|
| 1203 | NTCIP Object Definitions for Dynamic Message Signs (DMS) |
| 1204 | NTCIP Environmental Sensor Station Interface Standard (ESS) |
| 1209 | NTCIP Data Element Definitions for Transportation Sensor Systems (TSS) |
| 1210 | NTCIP Field Management Stations – Part 1: Object Definitions for Signal System Masters (FMS) |
| 1211 | NTCIP Object Definitions for Signal Control and Prioritization (SCP) |
| 1213 | NTCIP Object Definitions for Electrical and Lighting Management Systems (ELMS) |

## NTCIP Device Standards Without Systems Engineering Content

| Doc # | NTCIP Device Standards Without Systems Engineering Content |
|---|---|
| 1202 | NTCIP Object Definitions for Actuated Traffic Signal Controller Units (ASC) |
| 1205 | NTCIP Object Definitions for Closed Circuit Television Camera Control (CCTV) |
| 1206 | NTCIP Object Definitions for Data Collection and Monitoring Devices (DCM) |
| 1207 | NTCIP Object Definitions for Ramp Meter Control Units (RMC) |
| 1208 | NTCIP Object Definitions for Closed Circuit Television Switching (CCTV) |

## REFERENCES

**Web sites for ITS Standards**

- http://www.standards.its.dot.gov/
- www.fhwa.dot.gov
- www.its.dot.gov/standards/
- www.ntcip.org
- www.ite.org
- www.ieee.org

**Web sites for Systems Engineering Development**

- www.incose.org
- www.fhwa.org
- www.ieee.org

**Guides that Use the Systems Engineering Process**

- NTCIP Guide
- TMDD Guide
- IEEE 1512 Guide

**Specific Texts, Documents and Standards**

- International Council on Systems Engineering. Systems Engineering Handbook Version 3.2. January 2010.
- Alexander, Ian, and Ljerka Beus-Dukic. Discovering Requirements. Wiley, 2009.
- United States Department of Transportation. Systems Engineering Guidebook for Intelligent Transportation Systems Version 3.0. November 2009.
- Berenbach, Brian, Daniel Paulish, Juergen Kazmeier, and Arnold Rudorfer. Software and Systems Requirements Engineering: In Practice. McGraw Hill, 2009.
- IEEE 1233-1998 IEEE Guide for Developing System Requirements Specifications.
- IEEE 830-1998 Recommended Practice for Software Requirements Specifications.
- IEEE Std 1362-1998 IEEE Guide for Information Technology – System Definition – Concept of Operations (ConOps) Document.
- ANSI/AIAA G-043-1992 Guide for the Preparation of Operational Concept Documents.

**Finding the Relevant Standards**

- USDOT Research and Innovative Technology Administration (RITA) ITS Program Website http://www.standards.its.dot.gov
- Standards Search Feature http://www.standards.its.dot.gov/StdsSearch.asp
- Links to Standard Development Organizations (SDOs) and Standards Working Groups