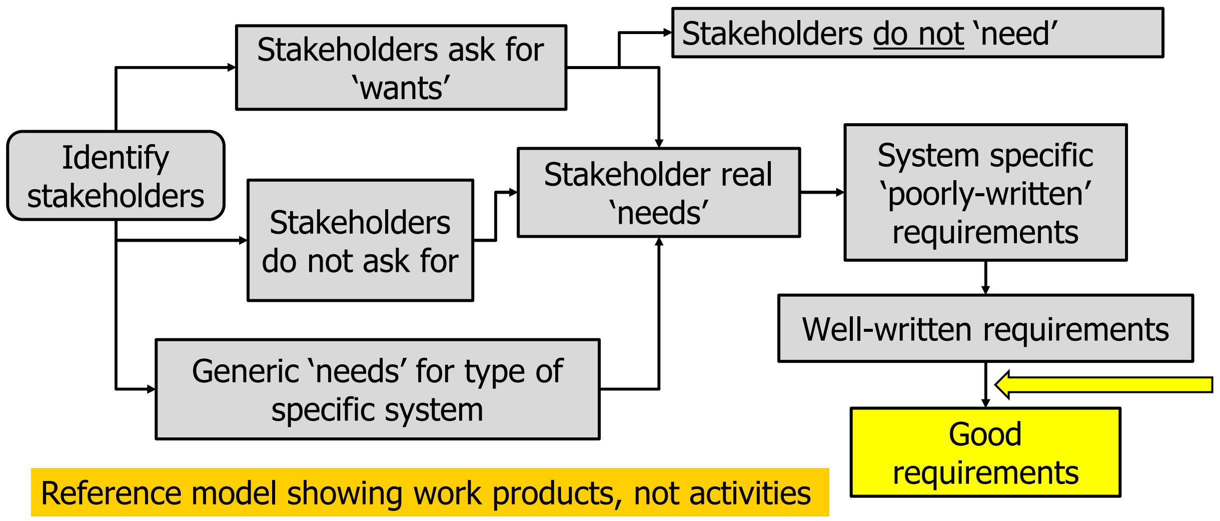
How to write good requirements Module 8 of 10 **Converting well-written requirements to good requirements**

Version 1.1.4

Course Module topics

Gap analysis for writing good requirements


Identify stakeholders
Stakeholders ask for 'wants'
Stakeholders do not ask for
Generic 'needs' for type of specific system
Stakeholders do not 'need'
Stakeholder real 'needs'
System specific 'poorly-written' requirements
Well-written requirements
Good requirements

Reference model showing work products, not activities

How to write good requirements 0301-3

Objectives of Session 8

1. To explain the additional attributes of good requirements
2. To explain the requirements for good specifications
3. To explain the systems approach to writing good requirements
4. To explain two reasons for requirements changes and how to prevent one of them
5. To practice what has been taught
6. To provide the opportunity to exercise the 5 levels of knowledge in the updated Blooms taxonomy

How to write good requirements 0801-4

Knowledge components

- Lecture
  - Sets the context and provides overview
- Readings
  - None
- Reference
  - 0850 MIL-STD 961E w/Change 1, 2 April 2008 (as a template or check list)
- Exercises
  - 8-1 Attributes of good requirements

Module topics

- **Attributes of good requirements**
- Requirements for good specifications
- The systems approach to writing good requirements
- Two reasons for requirements changes and how to prevent one of them
- Exercises

Definitions

- **Want** – something a stakeholder wants
- **Need** – something the stakeholder needs to remedy the problem
- **Requirement request** - A want/need in the form of a draft requirement that has not yet been accepted as a requirement. (*in this course, not a standard term*)
- **Stakeholder requirement** = a requirement request
- (textual) [system] **Requirement** - "A statement that identifies a product or process operational, functional, or design characteristic or constraint, which is unambiguous, testable or measurable, and necessary for product or process acceptability (by consumers or internal quality assurance guidelines)", (IEEE, 1998)
- **Well-written requirement** – a requirement that meets the grammatical and vocabulary requirements for requirement statements
- **Good requirement** – a well-written, feasible to implement, requirement for which the customer is willing to pay

13 attributes of a good requirement

1. Well-written (inherited) – the text sentence (imperative)
2. Real (inherited from need)
3. Acceptance criteria (inherited from need scenario)
4. Agreement on how compliance to the requirement will be verified
5. Traced or linked to a source (owner) via a scenario
6. Prioritized
7. Reason (inherited from need)
8. Unique
9. Consistent
10. Grouped in document (by ID numbers and keywords in database)
11. Technically feasible (inherited from scenario)
12. Affordable (estimated) (inherited from scenario)
13. Achievable by the need-by date (estimated) (inherited from scenario)

1. Well-written (inherited)

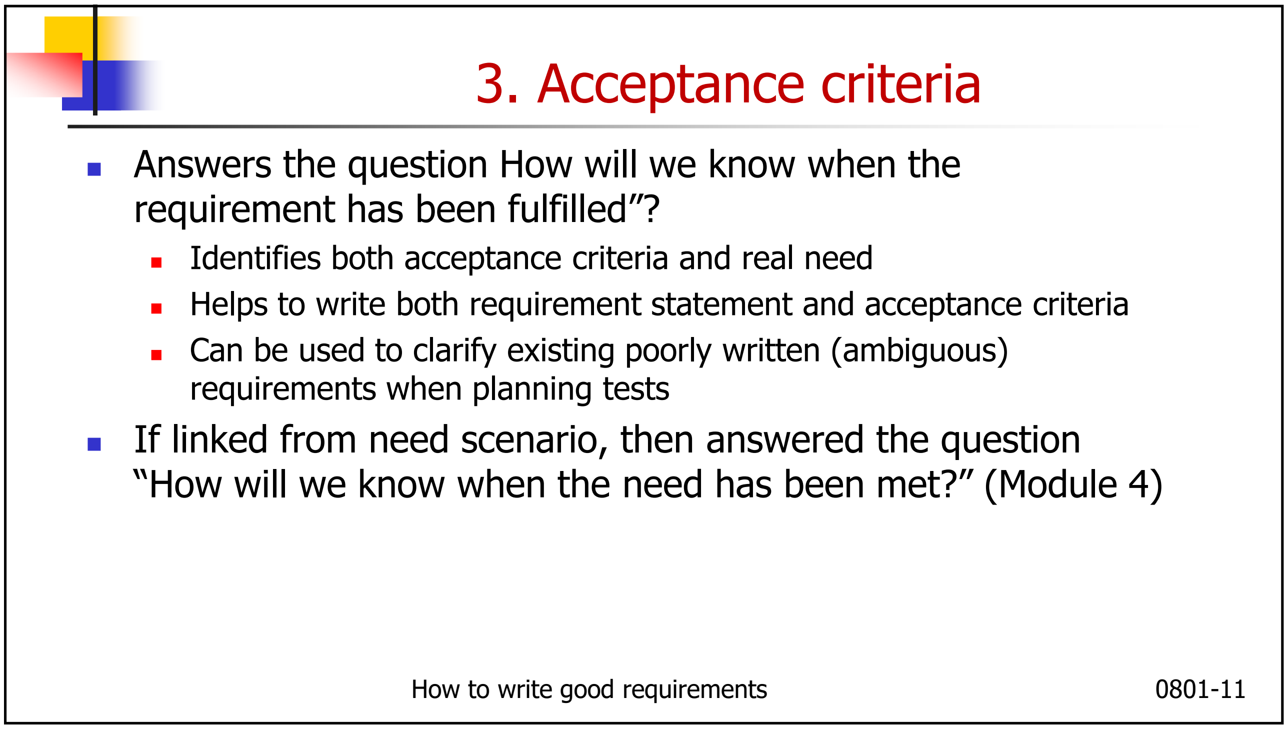
- Six attributes of a well-written requirement statement (Module 7)
  1. Atomic (singular)
  2. Unambiguous
  3. Verifiable**
  4. Understandable**
  5. Concise (Adequate)**
  6. Complete

13 + 5 = 18

2. Real (valid)

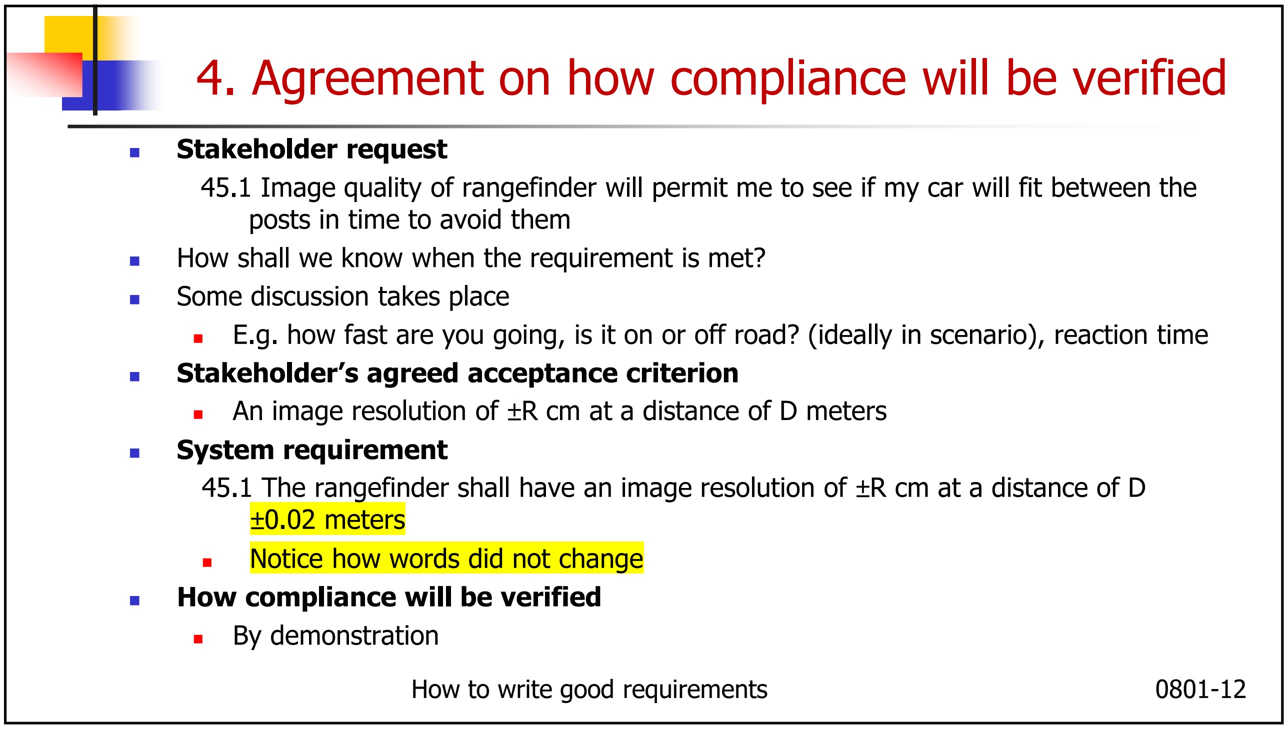
- A real need – something the stakeholder can justify
  - Specific to the situation
  - Inherited from generic needs
- "The five whys" is a useful tool to reach down to the reason during active brainstorming
  - Originally developed by Sakichi Toyoda and was used within the Toyota Motor Corporation to deal with problems*
  - Systems Thinking Toolbox Section 7.5
  - Other Kipling Questions (who, what, where, when and how) should also be used

* Ōhno Taiichi, Toyota production system : beyond large-scale production, Productivity Press , Cambridge, MA, 1988



3. Acceptance criteria

- Answers the question How will we know when the requirement has been fulfilled"?
  - Identifies both acceptance criteria and real need
  - Helps to write both requirement statement and acceptance criteria
  - Can be used to clarify existing poorly written (ambiguous) requirements when planning tests
- If linked from need scenario, then answered the question "How will we know when the need has been met?" (Module 4)



4. Agreement on how compliance will be verified

- **Stakeholder request**

  45.1 Image quality of rangefinder will permit me to see if my car will fit between the posts in time to avoid them
- How shall we know when the requirement is met?
- Some discussion takes place
  - E.g. how fast are you going, is it on or off road? (ideally in scenario), reaction time
- **Stakeholder's agreed acceptance criterion**
  - An image resolution of ±R cm at a distance of D meters
- **System requirement**

  45.1 The rangefinder shall have an image resolution of ±R cm at a distance of D ±0.02 meters
  - Notice how words did not change
- **How compliance will be verified**
  - By demonstration

4. Agreement on how compliance will be verified

- **Stakeholder need**

  145.1 a protective fuse in the event of a short circuit
- How shall we know when the requirement is met?
- Some discussion takes place
  - Can’t test a fuse and use it
- **Stakeholder’s agreed acceptance criterion**
  1. Accept manufacturer’s ratings?
  2. Statistical random sampling from a batch of fuses to be tested?
- **System requirement**

  145.1 The fuse shall blow within T milliseconds when the current exceeds 30±0.1 Amps
- **How compliance will be verified**
  - By demonstration in 5 selected random samples from same batch

5. Traced or Linked

Product

- Backwards
  - source of requirements and reason for existence
  - owner
- Forwards towards implementation
  - Designs
  - Parts
  - minimising forgetfulness
  - change impact assessment
- Sideways
  - References (used in, used by), keywords in database

Process

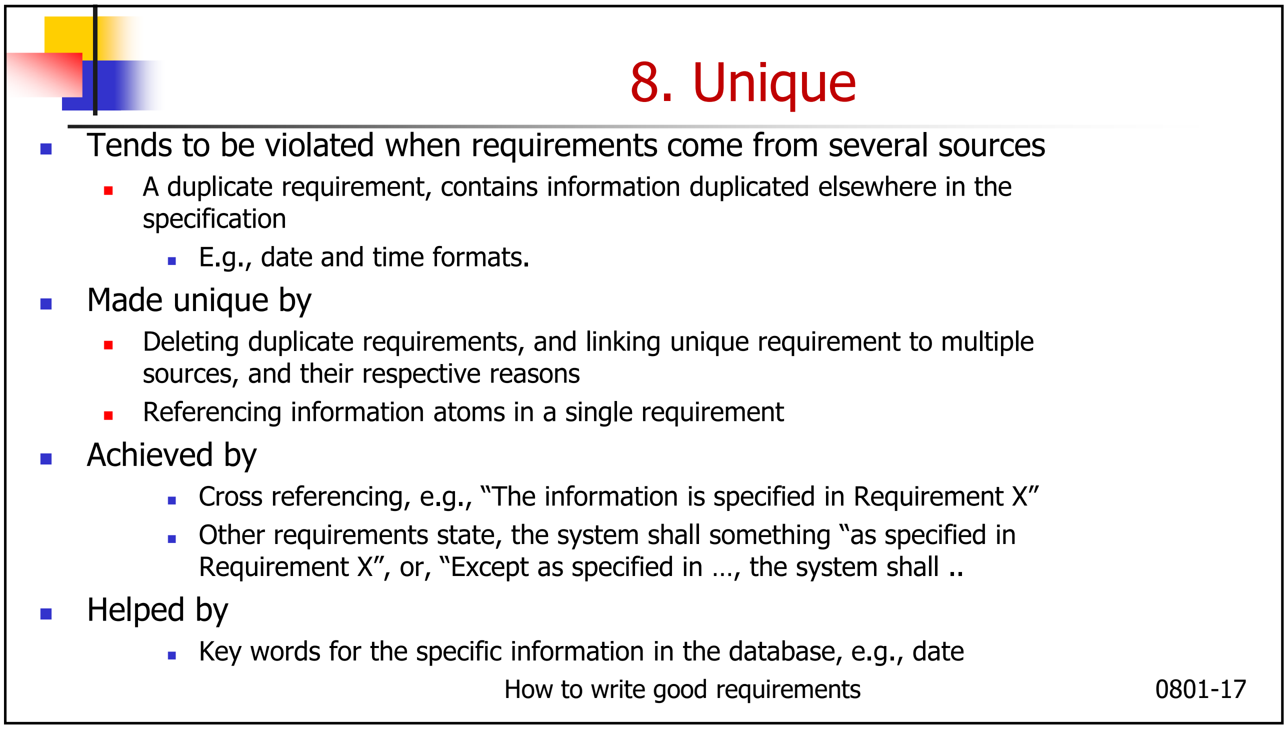
- Test path
  - Plans
  - Procedures
- Build
  - Documents

6. Prioritized

- Inherited from need
- If there are no need scenarios, then developed as outlined in Module 4
- Helps customer make choices about which stakeholder needs (requirements) to delete if
  - Budget or schedule cannot be met for the whole set of stakeholder requirement requests
  - Budgets are cut and/or need-by-date is moved up
- Which then flow down to which system requirements to "delete" (mark as not to be implemented and/or stop work, might be reinstated)
- Should be used to realize higher priority system requirements in earlier builds (in software and product versions)

7. Reason

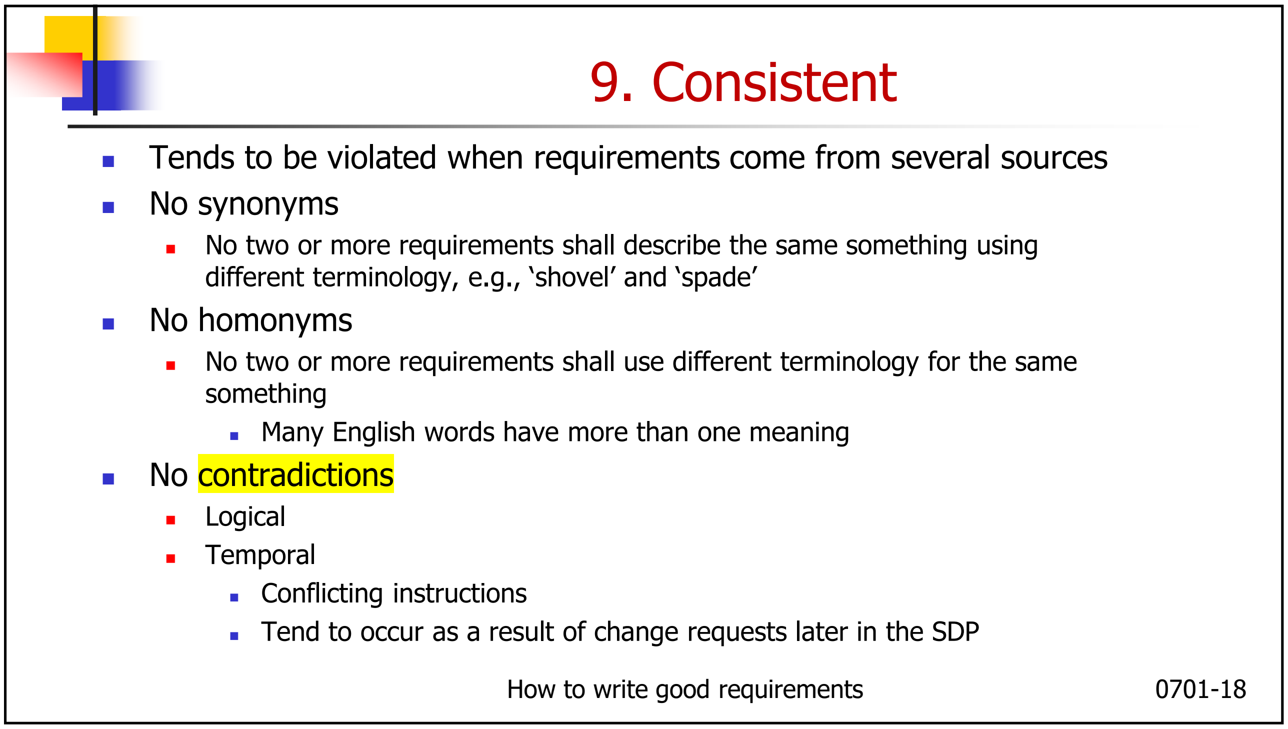
- Inherited from need
- If there are no need scenarios, then developed as described in Module 4
- Sometimes not obvious
  - E.g., to prevent known undesirable emergent properties
- Useful in influencing design decisions
- Needed when dealing with change requests once the requirements have been approved, through the operations and maintenance states
- Helps to minimize effect of loss of tacit knowledge when staff leave



8. Unique

- Tends to be violated when requirements come from several sources
  - A duplicate requirement, contains information duplicated elsewhere in the specification
    - E.g., date and time formats.
- Made unique by
  - Deleting duplicate requirements, and linking unique requirement to multiple sources, and their respective reasons
  - Referencing information atoms in a single requirement
- Achieved by
    - Cross referencing, e.g., "The information is specified in Requirement X"
    - Other requirements state, the system shall something "as specified in Requirement X", or, "Except as specified in …, the system shall ..
- Helped by
    - Key words for the specific information in the database, e.g., date

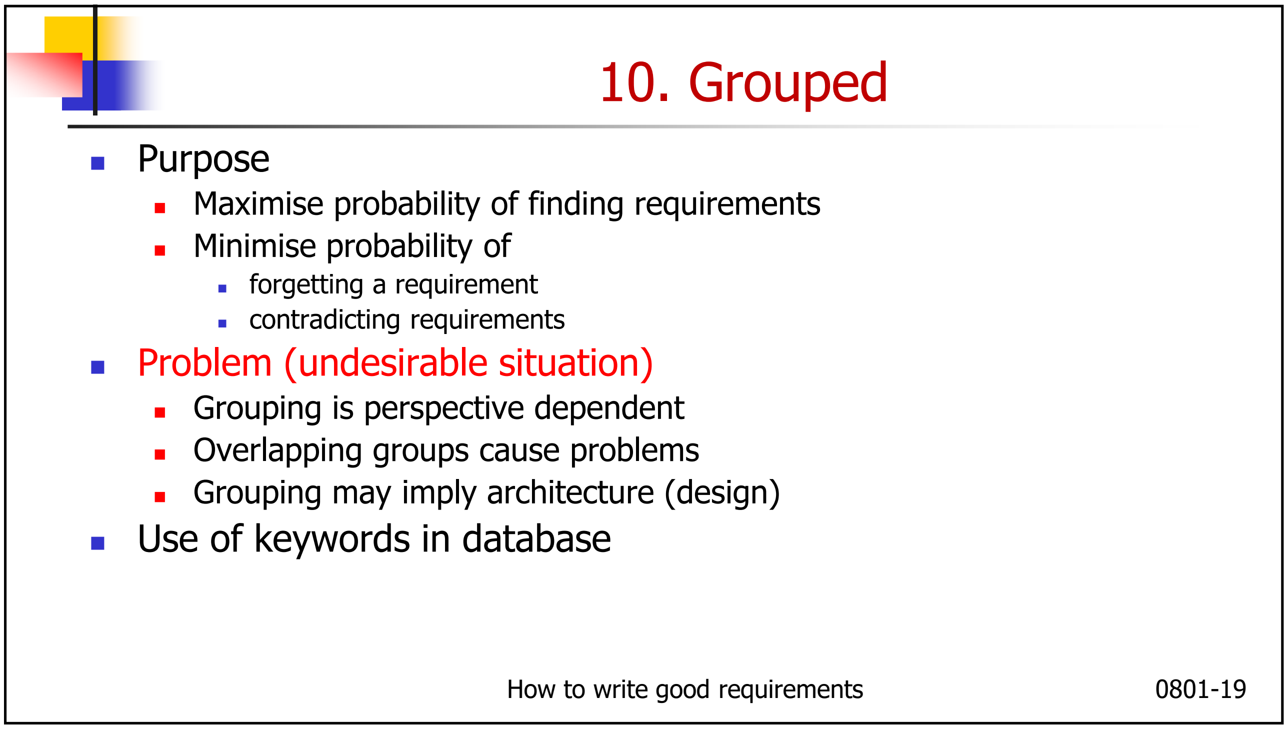
How to write good requirements 0801-17



9. Consistent

- Tends to be violated when requirements come from several sources
- No synonyms
  - No two or more requirements shall describe the same something using different terminology, e.g., 'shovel' and 'spade'
- No homonyms
  - No two or more requirements shall use different terminology for the same something
    - Many English words have more than one meaning
- No contradictions
  - Logical
  - Temporal
    - Conflicting instructions
    - Tend to occur as a result of change requests later in the SDP

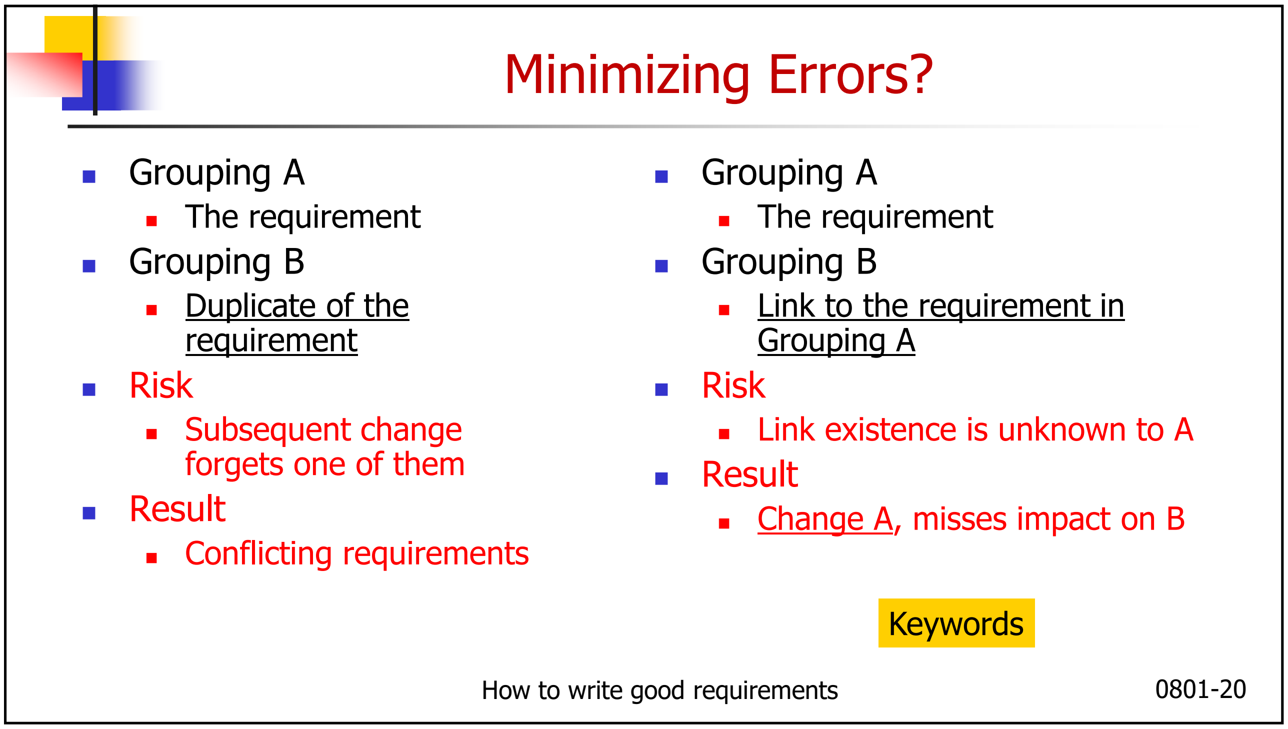
How to write good requirements 0701-18



10. Grouped

- Purpose
  - Maximise probability of finding requirements
  - Minimise probability of
    - forgetting a requirement
    - contradicting requirements
- Problem (undesirable situation)
  - Grouping is perspective dependent
  - Overlapping groups cause problems
  - Grouping may imply architecture (design)
- Use of keywords in database

How to write good requirements 0801-19



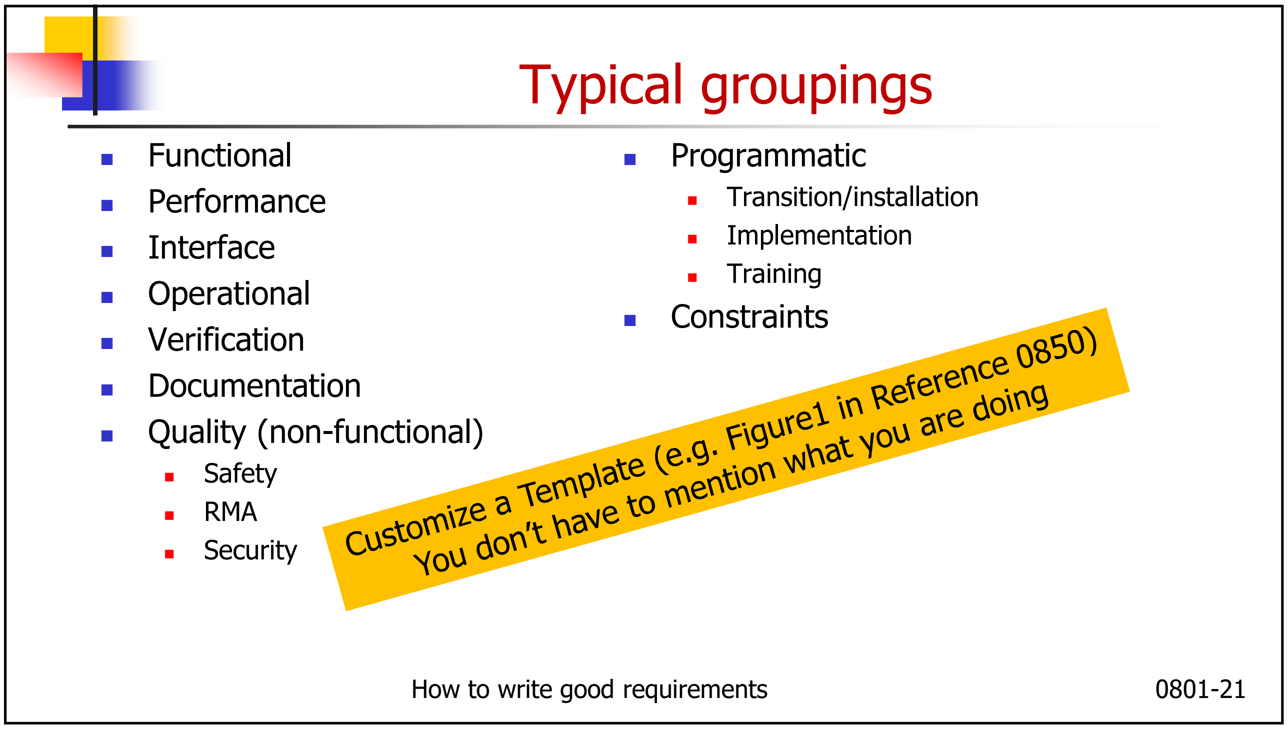
Minimizing Errors?

- Grouping A
  - The requirement
- Grouping B
  - Duplicate of the requirement
- Risk
  - Subsequent change forgets one of them
- Result
  - Conflicting requirements

- Grouping A
  - The requirement
- Grouping B
  - Link to the requirement in Grouping A
- Risk
  - Link existence is unknown to A
- Result
  - Change A, misses impact on B

Keywords

How to write good requirements 0801-20

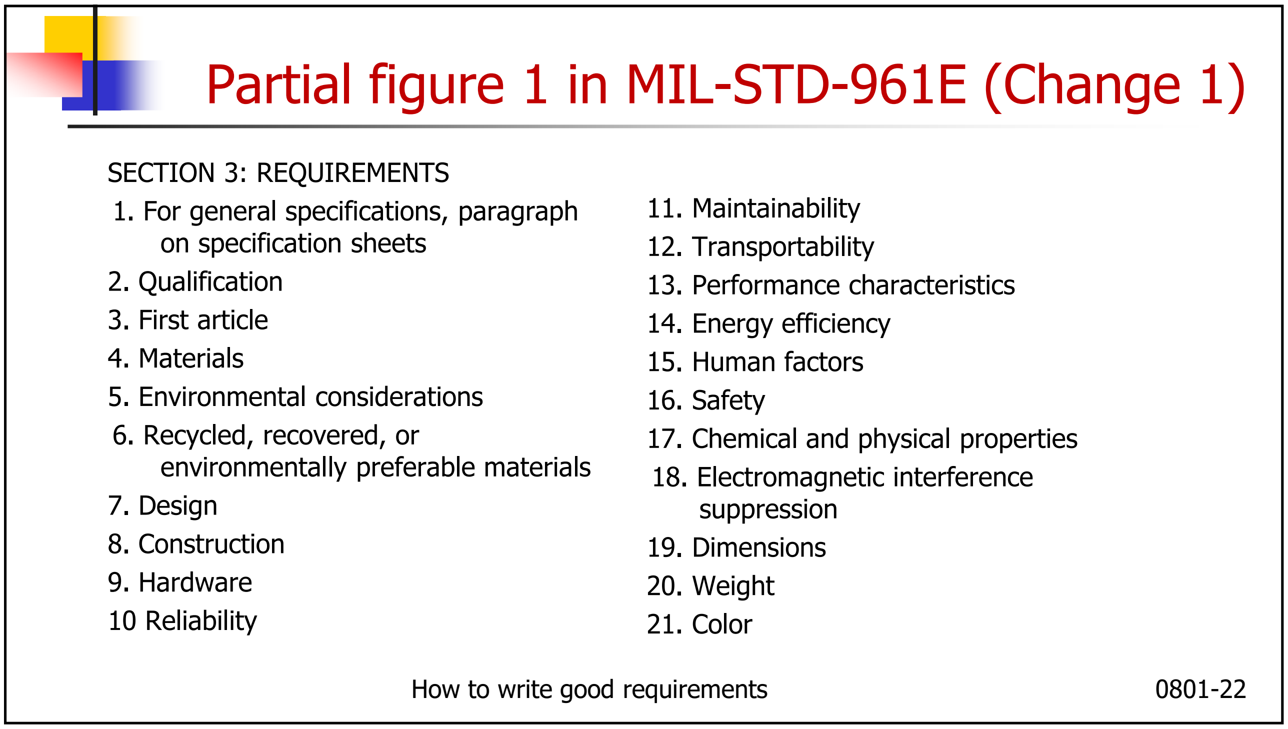


Typical groupings

- Functional
- Performance
- Interface
- Operational
- Verification
- Documentation
- Quality (non-functional)
  - Safety
  - RMA
  - Security
- Programmatic
  - Transition/installation
  - Implementation
  - Training
- Constraints

Customize a Template (e.g. Figure1 in Reference 0850)
You don't have to mention what you are doing

How to write good requirements 0801-21



Partial figure 1 in MIL-STD-961E (Change 1)

SECTION 3: REQUIREMENTS

1. For general specifications, paragraph on specification sheets
2. Qualification
3. First article
4. Materials
5. Environmental considerations
6. Recycled, recovered, or environmentally preferable materials
7. Design
8. Construction
9. Hardware

10 Reliability

11. Maintainability
12. Transportability
13. Performance characteristics
14. Energy efficiency
15. Human factors
16. Safety
17. Chemical and physical properties
18. Electromagnetic interference suppression
19. Dimensions
20. Weight
21. Color

How to write good requirements 0801-22

11. Technically feasible

- Inherited from functional scenario
- Perform feasibility study to find out if there is at least one way to manifest the function/requirement in the real world
  - Performed in parallel with cost and schedule feasibility studies
  - Don't waste time performing a technical feasibility study on something that is not affordable, or cannot be obtained or created by its need-by date

How to write good requirements 0801-23

12. Affordable (estimated)

- Estimated Cost
- Inherited from need
- If there are no need scenarios, then developed as described in Module 4
- Does not have to be identified at the atomic requirement level, can be referenced to scenario
  - If the need is removed, the entire set of requirements will be deleted

How to write good requirements 0801-24

13. Achievable by the need-by date (estimated)

- Estimated schedule
- Inherited from need
- If there are no need scenarios, then developed as described in Module 4
- Does not have to be identified at the atomic requirement level, can be referenced to scenario
  - If the need is removed, the entire set of requirements will be deleted

Module topics

- Attributes of good requirements
- **Requirements for good specifications**
- The systems approach to writing good requirements
- Two reasons for requirements changes and how to prevent one of them
- Exercises

Requirements for good specifications

1. The specification shall contain well-written good requirements
2. The specification shall conform to the applicable standard and version for documentation in the project[1]
3. The specification shall identify non-applicable sections as "non-applicable"
   - (deletions are ambiguous)
4. Each requirement in the specification shall contain the 13 attributes

Note
1 Includes grouping

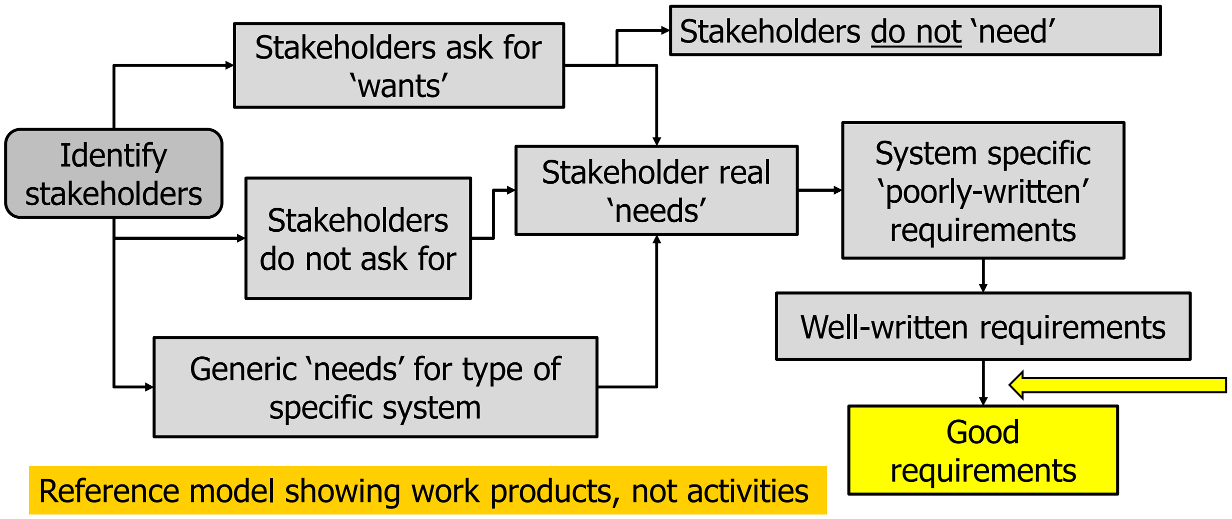
Typical Checklist Questions

1. Are specialised terms defined in a glossary?
2. Does each requirement stand on its own? (Complete)
3. Do individual requirements always use the same term for the same concept? (lack of synonyms)
4. Do any requirements use a different term for the same concept? (use of homonyms)
5. If a requirement makes reference to some other facility is this described elsewhere?
6. Are related requirements grouped together?
7. Are any requirements duplicated?
8. Is any information duplicated?

Module topics

- Attributes of good requirements
- Requirements for good specifications
- **The systems approach to writing good requirements**
- Two reasons for requirements changes and how to prevent one of them
- Exercises

Workflow analysis for good requirements


Identify stakeholders
Stakeholders ask for 'wants'
Stakeholders do not 'need'
Stakeholders do not ask for
Stakeholder real 'needs'
System specific 'poorly-written' requirements
Generic 'needs' for type of specific system
Well-written requirements
Good requirements

Reference model showing work products, not activities

The systems approach to writing good requirements

1. **Understand the current situation**
   1. Understand what is undesirable in the current situation
   2. Understand what additional changes, improvements and additions the stakeholders want/wish for
2. **The assumptions**
   1. The description is of an ideal process, the real world is different, so the process will need to be tailored
3. **The FCFDS**
   1. The customer signs off on the System Requirements Specification (SRS) after the System Requirements Review (SRR) with no changes
4. **The problem** (to develop the FCFDS working back from the SRR)
   1. At the end of the SRR, the customer signs off on the SRS with no changes
   2. Hold the SRR

The systems approach to writing good requirements

4. **The problem** (cont.)

3. Circulate the draft SRS and summaries of other documentation to the stakeholders for agreement prior to the SRR
4. Prepare other appropriate documentation for the SRR (e.g., updated plans)
5. Create the draft SRS
6. Verify or ensure the well-written requirements are also good requirements
7. Verify or ensure the requirements are also well-written requirements
8. Verify or ensure the non-functional requirements are written
9. Verify or ensure the other appropriate generic requirements not derived from scenarios are written
10. Write the requirements to perform the quantified functions in the CONOPS scenarios
11. Tag the quantified functions with the properties and attributes of good requirements
12. Expand the scenarios in the CONOPS into atomic quantified functions
13. The customer signs off on the CONOPS at OCR and agrees to proceed with the project

The systems approach to writing good requirements

4. The problem (cont.)

14. Hold the Operations Concept Review (OCR) or equivalent milestone review
15. Adjust draft CONOPS until customer agrees it is feasible, affordable and can be delivered by the need-by date
16. Verify or ensure the appropriate non-functional attributes are added to the functional and performance scenarios in the draft CONOPS
17. Transform the conceptual reference representation into a draft CONOPS
18. Create a conceptual reference representation of a solution system that will remedy the problem (meet the stakeholder's need)
    1. Without the undesirability in the current situation
    2. With the additional changes, improvements and additions the stakeholders want/wish for
    3. The transition process

5. The solution is how you implement each step

Continuum HTP Numerology 18 = חי (Chai) = Life (English)

How to write good requirements 0801-33

The systems approach to writing good requirements

4. The solution

- Notes
- Requirements writing steps 4.6 to 4.10 do not have to be performed in a sequential manner
  - You should be able to write a set of well-written requirements directly from the appropriate sources
- If you are working in an environment where you are eliciting and elucidating requirements without a CONOPS (the B paradigm), then
  1. Create a rough but complete draft CONOPS but don't mention/share it unless you get a request to do so
     1. Don't try to change the system in which you are working
  2. Treat poorly-written requirements as poorly-written scenarios

How to write good requirements 0801-34

Module topics

- Attributes of good requirements
- Requirements for good specifications
- The systems approach to writing good requirements
- **Two reasons for requirements changes and how to prevent one of them**
- Exercises

Two reasons for requirements changes

1. The stakeholders' , "I forgot to mention"
    1. The [baseline] needs at the start of the project
        1. "get all the [baseline] requirements up front"
2. The need has changed
    1. Since the project (system development process) began
        1. The need has changed
        2. A better understanding of what the need really is
    2. Since the system was placed into service
        1. Latent defects
        2. The need has changed
        3. A better understanding of what the need really is

Preventing the stakeholders' , "I forgot to mention"

- Systemically and systematically identifying all relevant stakeholders
  - Tools include the Nine System Model
- Developing an understanding of the stakeholders' real needs
- Developing an understanding of the stakeholders' domains
  - (1) Problem, (2) implementation and (3) solution domains
  - Needed for identifying generic requirements
  - Needed for identifying 'requests' ('wants') that are not real 'needs'
  - Needed for identifying needs that were not requested
- Holding a dialogue
- Converting assumptions into facts (true or false (reason for false))

How to write good requirements 0801-37

Preventing the stakeholders' , "I forgot to mention"

- Creating the draft generic CONOPS reference model of needed solution system
- Converting the draft CONOPS reference model into the specific CONOPS
- Tools include
  - Short restricted attendance meetings (less than 1 hour)
  - Meeting agendas (provided ahead of time)
  - Active brainstorming
  - Active listening
  - The five why's
  - KISS
  - The Principle of Hierarchies
  - Concept maps, Checkland's rich pictures and other sketches
  - Modelling tools

How to write good requirements 0801-38

Module topics

- Attributes of good requirements
- Requirements for good specifications
- The systems approach to writing good requirements
- Two reasons for requirements changes and how to prevent one of them
- **Exercises**

Exercise 8-1

- In exercise 5-2 you wrote a set of well-written requirements, in exercise 6-1 you ingested them into Tiger Pro, in exercise 7-1 you rewrote them and added acceptance criteria to the requirements

1. Verify or ensure the acceptance criteria are in the Tiger Pro database
2. Add the priority and risk (both probability and severity) attributes to each of the 7-1 requirements (make some guesses, or select random numbers)
3. Display and save the attribute profile graphs, for priority, risk probability and severity
4. Create a traditional requirement Risk Matrix for the 7-1 requirements
5. Prepare a <5 minute presentation containing
    1. The three attribute profiles for the requirements
    2. The requirement risk matrix
    3. The exercise problem formulated per COPS problem formulation template
    4. A compliance matrix for the exercise
    5. Lessons learned from exercise (<3 min)
    6. Why you think this exercise was included in the lesson
    7. This slide and the lesson version number
6. Save as a PowerPoint file in format Exercise8.1-abcd.pptx
7. Post/email presentation as and where instructed

Meeting the objectives

#	Objectives	Met
1	To explain the additional attributes of good requirements	14-32
2	To explain the requirements for good specifications	34-35
3	To explain the systems approach to writing good requirements	37-41
4	To explain two reasons for requirements changes and prevent one of them	43-45
5	To practice what has been taught	47
6	To provide the opportunity to exercise the 5 levels of knowledge in the updated Blooms taxonomy	47

How to write good requirements 0801-41

Any questions ?

1. Best
2. Worst
3. Missing

Email: beyondsystemsthinking@yahoo.com
Subject: <class title> BWM Module #

How to write good requirements 0801-42