



How to write good requirements

Session 6 of 10

Converting stakeholder needs to requirements

Version 1.2.6

How to write good requirements

0601-1

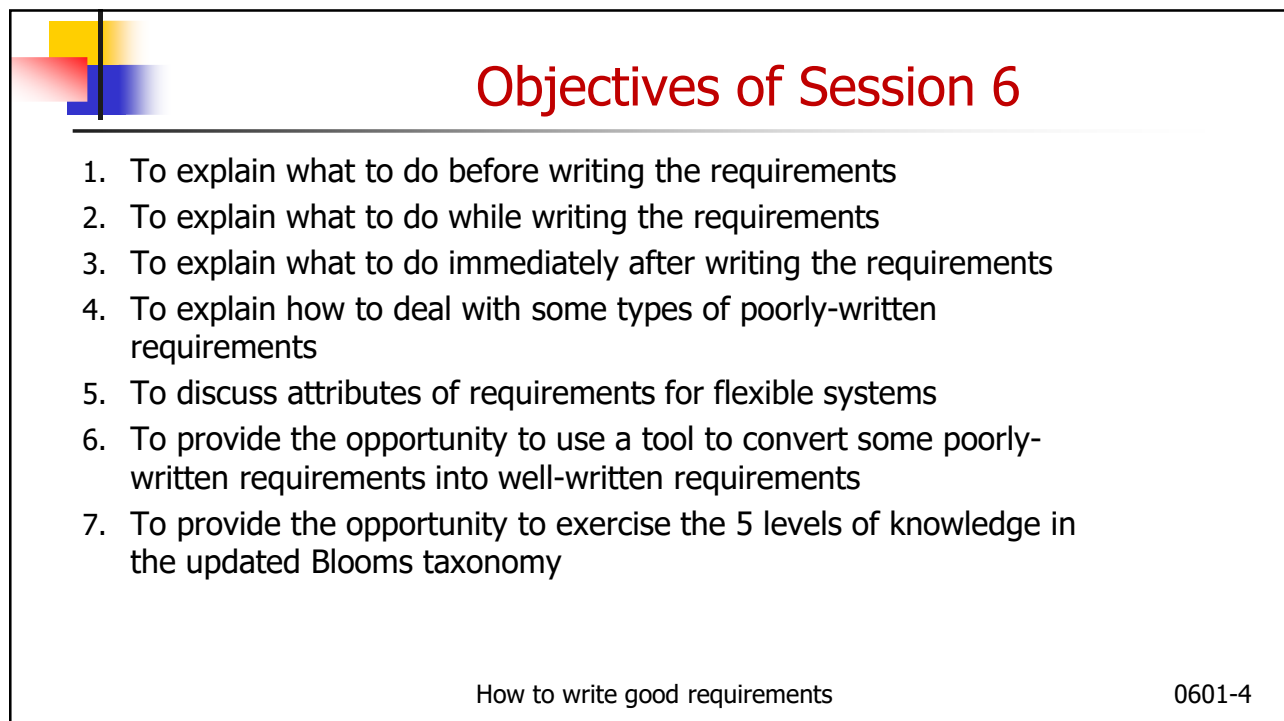
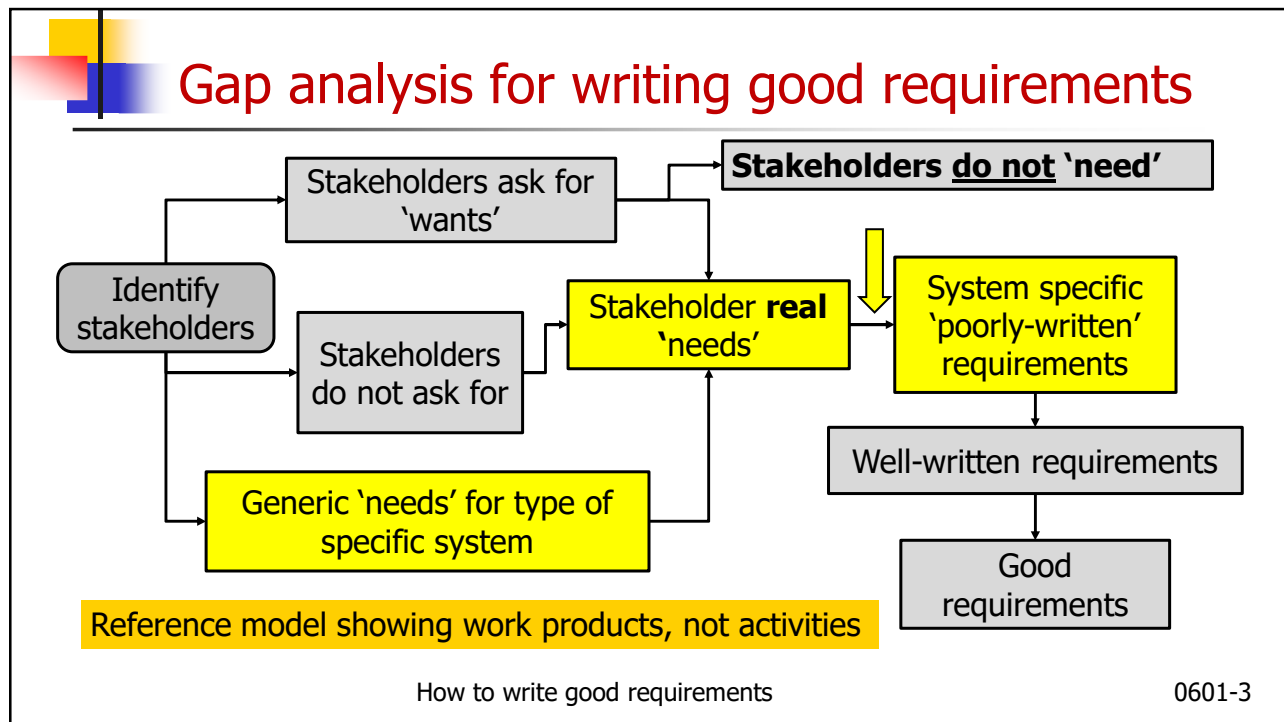


Course session topics

1. Introduction to requirements
2. Stakeholders and their importance
3. Communicating with the stakeholders
4. Converting stakeholder wants to needs
5. Documenting and storing stakeholders' needs
- 6. Converting stakeholder needs to requirements**
7. Converting requirements to well-written requirements
8. Converting well-written requirements to good requirements
9. The use of requirements in the rest of the system development process
10. Summary and closeout

How to write good requirements

0601-2



Knowledge components

- Lecture
 - Sets the context and provides overview
- Readings
 - 0602 Kasser J.E., [Writing Requirements for Flexible Systems](#), *proceedings of the INCOSE-UK Spring Symposium* May 2001
- Exercises
 - 6-1 Tiger Pro practical
 - 6-2 Knowledge reading 0602
- Additional resources
 - Tiger Pro <https://therightrequirement.com/tiger-pro-tool-to-ingest-and-elucidate-requirements/>
 - Using Tiger Pro to fix poorly worded requirements, YouTube tutorial video <https://youtu.be/-Uegko4WQO8> (11:20)

How to write good requirements

0601-5

Session topics

- **What to do before writing the requirements**
- What to do when writing the requirements
- To show how to use smart numbering and computers to maximize completeness
- How to deal with some types of poorly-written requirements
- Writing requirements for flexible systems
- What to do after writing the requirements
- Exercises



How to write good requirements

0601-6



Reduce scenarios to atomic functions

- Verify or ensure there is a complete set of scenarios
- Scenarios may be in text or in graphic form (e.g. in a drawing)
- Separate scenario into atomic functions using Principle of Hierarchies
- Quantify the functions
- Example
- Consider the class exercises
 - The problem in the exercise can be considered **as a scenario** or a set of scenarios **to be performed**
 - The steps in the problem statement formulated per the COPS PFT can be considered as the (atomic) functions in the scenario(s) (from finish to start)
 - Adding the specific numbers of things to do in each step can be considered as an example of quantifying the functions

How to write good requirements

0601-7



Quantifying the functions

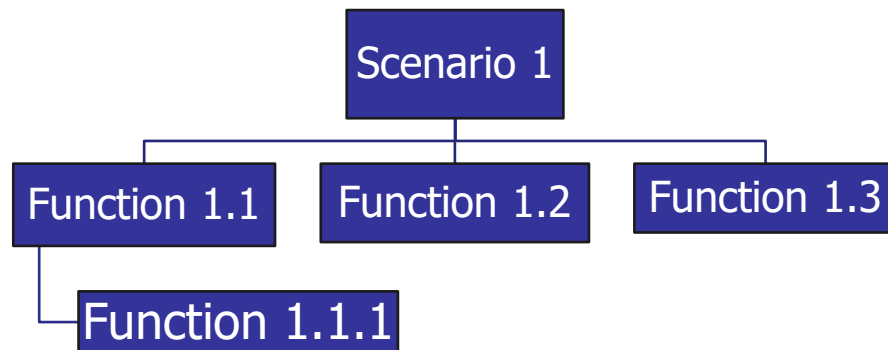
- Function "Transporting a product"
- Kipling questions (if not documented in scenario) include
 - Why is it being transported?
 - Who is transporting
 - What is being transported?
 - What care has to be taken, if any (Is it hazardous)?
 - How much is being transported
 - When is it being transported?
 - What state should it be in after transportation?
 - ... ?

How to write good requirements

0601-8

Converting scenarios to atomic functions

- Use Principle of Hierarchies



How to write good requirements

0601-9

Session topics

- What to do before writing the requirements
- **What to do when writing the requirements**
- Using smart numbering and computers to maximize completeness
- How to deal with some types of poorly-written requirements
- Writing requirements for flexible systems
- What to do after writing the requirements
- Exercises



How to write good requirements

0601-10

Convert scenarios/functions or poorly-written requirements to atomic statements

- Function or poorly-written requirement
 - 3.65 The system shall transmit all outgoing data at a rate of 1 Kbps using Reed-Solomon encoding following receipt of a transmit command
- Poorly-written because it contains **multiple items** to be verified **and** **omits** the required response time **and** does not contain a **tolerance** on 1 Kbps
- Atomic requirements
 - 3.65.1 The system shall transmit all outgoing **data at a rate of 1 Kbps** (**± 10 bps**)
 - 3.65.2 The system shall transmit all outgoing data using **Reed-Solomon encoding**
 - 3.65.3 The system shall start to transmit the outgoing data **within 10 Seconds of receipt of the command** to transmit data
- You can do this without knowing what Reed-Solomon encoding is, other than it is used in certain types of communications links

How to write good requirements

0601-11

Converting poor requirements to functions

- Poor requirement/scenario
 - 1.x. Student shall be able to quickly learn about a course (description, number of sessions, topics in each session, location and schedule) and enroll in the course using one click
- Functions (listed in sequential order)
 - 1.x.1 System provides a specific course enrollment menu containing a course description, number of sessions, topics in each session, course schedule (date and time), course location, (online, campus, building and room), instructor and an enroll and a cancel button (option)
 - 1.x.2 Student selects the desired location
 - 1.x.3 When the student selects the desired location, the system displays the name of the instructor
 - 1.x.4 Student clicks on 'enroll' button to enroll for the course
 - 1.x.5 Student clicks on 'cancel' button to cancel the enrolment request
 - 1.x.6 ...

How to write good requirements

0601-12



How to create a matched set of specifications for the whole system and its subsystems

- CONOPS scenarios uses functions/activities
- Function/performance requirements are functions with numbers
- System requirements
 - Mission and support functions (A spec)
- Subsystem requirements
 - Mission and support functions (B spec)
- Traceability of al requirements to
 - System specific requirements
 - CONOPS
 - System generic requirements
 - Domain knowledge (external)
 - Class of system
 - Laws and regulations

How to write good requirements

0601-13



Non-functional needs

- There are doing/action/functional/performance requirements and there are "being" or "non-functional requirements" (IREB's "Quality" requirements)
- Being requirements are not action requirements and just need "the shall statement" to express the ...
- Examples
 - The system shall weigh less than 1Kg
 - The system shall have **an estimated** Mean Time Between Failures (MTBF) of at least 1,000 hours

How to write good requirements

0601-14

Converting properties and attributes to requirements

101 The system shall receive data at a rate of 1000 ± 10 bps

102 The data received by the system shall contain 8 bits per word

111 The system shall transmit data at a rate of 1000 ± 10 bps

112 The data transmitted from the system shall contain 8 bits per word

501 The **estimated** system MTBF shall be greater than 4000 hours

502 The **estimated** system MTTR shall be less than 3 hours

601 The maximum operating temperature of the system shall be greater than 35 degrees Centigrade

602 The minimum operating temperature of the system shall be less than -5 degrees Centigrade

Object inspector

Typical Component

PropertiesEvents

Attribute	Value
Data input rate	1000 +/-10
Data output rate	1000 +/-10
Bits per word	8
MTBF	4000
MTTR	3
Max operating Temp	35
Min operating Temp	-5

How to write good requirements

0601-15

Converting non-functional needs to requirements

■ Need

- The classroom shall accommodate more students than have signed up for the course.

■ Questions to stakeholders

- **What** does "accommodate" mean?
 - Desk and chair with room to spread out
- **How** many students have signed up?
 - 25
- **Why** the need for "more"?
 - Last minute additions after the classroom is allocated

■ Attributes and properties

- Classroom: size; remaining inherited from applicable occupied building and safety codes

■ Requirement?

7-3. The classroom shall accommodate 27 students

Good or bad?

How to write good requirements

0601-16

Which ~~is correct~~ are acceptable?

- 7-3.1 The classroom shall accommodate between 25 and 35 students.
- 7-3.2 The classroom shall accommodate a minimum of 25 students.
- 7-3.3 The classroom shall accommodate more than 25 students.
- 7-3.4 The classroom shall accommodate no less than 27 students.
- 7-3.5 The classroom shall accommodate up to 27-35 students.
- 7-3.6 The classroom shall accommodate up to 35 students.

- Do we want to bound the size of the room

2. Conceive
solution options

5. Select
preferred
option

How to write good requirements

0601-17

More needs to requirements

Need "A system able to transport at least 20 kilograms ..."

- **Requirement can be written in at least the following nine ways:**

1. The system shall **transport at least** 20 kilograms
2. The system shall be able to **transport at least** 20 kilograms
3. The system shall be able to support **not less than** 20 kilograms
4. The system shall support transporting **not less than** 20 kilograms
5. The system shall be able to **transport up to** 20 kilograms
6. The system shall have a **minimum carrying capacity** of 20 kilograms
7. The system shall have a **maximum carrying capacity** of at least 20 kilograms
8. The **transportation capacity of the system shall** be a minimum of 20 kilograms
9. The **transportation capacity of the system shall** be between 0 and 20 kilograms

Prime Directive: Don't use synonyms. Use "need" wording in requirement

How to write good requirements

0601-18

More needs to requirements

Need "A system able to transport at least 20 kilograms ..."

■ **Requirement can be written in at least the following nine ways:**

1. The system shall **transport at least** 20 kilograms Most concise
2. The system shall be able to **transport at least** 20 kilograms
3. The system shall be able to support **not less than** 20 kilograms
4. The system shall support transporting **not less than** 20 kilograms
5. ~~The system shall be able to transport up to~~ 20 kilograms
6. The system shall have a **minimum carrying capacity** of 20 kilograms
7. ~~The system shall have a maximum carrying capacity~~ of at least 20 kilograms
8. The **transportation capacity of the system shall** be a minimum of 20 kilograms
9. ~~The transportation capacity of the system shall~~ be between 0 and 20 kilograms

Prime Directive: Don't use synonyms. Use "need" wording in requirement

How to write good requirements

0601-19

Functional and non-functional (text) scenario

103. Except on Saturdays, the system shall transport up to 1000 men with up to 100 Kilograms of baggage each, up to 1000 miles, within 10 hours.

- Functional
 - What is being transported
- Non-functional
 - When it is being transported (Sunday to Friday)

How to write good requirements

0601-20

Atomic requirements from scenario

103. The system shall operate six days a week, Sunday to Friday^[1].
104. The system shall transport up to 1,000 men each weighing no more than w Kilograms^[2].
105. The system shall transport up to 100,000 Kilograms of baggage.
106. The system shall transport the combination of men and baggage up to 1600 Kilometers^[3].
107. The system shall complete the transport within 10 hours^[4].
108. The volume of an individual item of baggage shall not exceed h by w by d Meters.^[5]

Questions

- [1] How many hours per day?
- [2] Should we use minimum, average or maximum weights for the people?
- [3] What state should the men and baggage be in after transportation?
- [4] Is the 10 hours included in 103?
- [5] What is form factor of baggage?
- [6] ...

This is why you need scenarios in a CONOPS

How to write good requirements

0601-21

Flow down additional attributes

- Attribute of need flows down into attribute of requirements
 - Source: need scenario (or decision based on meeting need), "owner"
 - Acceptance criteria (for each requirement; clarifies requirement and facilitates verification or test creation)
 - Priority
- If a database/tool is used, link (could be automatic), don't repeat information
 - Reason for requirement (inherited directly from source)
 - Priority (inherited directly from source)
 - Risk probability of occurrence (inherited directly from source)
 - Risk severity of consequences (inherited directly from source)
 - Scenario/requirement in which risk is mitigated (checksum)
 - Either depending on project
 - Cost estimate (normally inherited directly from source)
 - Schedule estimate (normally inherited directly from source)
 - Stability (normally inherited directly from source)

How to write good requirements

0601-22



Requirements Validation

- Concerned with checking requirements for consistency completeness and accuracy
- Involves all system stakeholders, as well as those who sourced needs/requirements
- Can be done in several ways

How to write good requirements

0601-23



Validation Methods: Requirements Reviews

- Group of people who read and analyse requirements
- Look for problems
- Discuss problems
- Agree on a set of actions to address problems
- Revise requirements (via Change Control Board)

How to write good requirements

0601-24



Requirements Validation: Prototyping

- A prototype is an interpretation of the requirements and provides a powerful tool for further requirements elicitation
- Useful
 - when system of interest hard to envision
 - for risk mitigation on challenging projects
 - And other reasons
- The prototype may need to evolve as requirements elicitation progresses

How to write good requirements

0601-25



Requirement Validation: Models

- A model:
 - May form a set of requirements
 - Can be built of the system of interest
- Models may be
 - Data-flow models
 - Object models
 - Event models
 - Entity-relationship models
 - Mathematical performance models
 - etc.

How to write good requirements

0601-26

Validating and testing requirements

- Meet acceptance criteria?
- Well-written requirements
 - Demonstration (testing)
 - Analysis
 - Simulation and modeling
 - Other
- Poorly written requirements
 - Create acceptance criteria with customer
 - Convert to set of well-written requirements (only if time and cost permits)

How to write good requirements

0601-27

Session topics

- What to do before writing the requirements
- What to do when writing the requirements
- **Using smart numbering and computers to maximize completeness**
- How to deal with some types of poorly-written requirements
- Writing requirements for flexible systems
- What to do after writing the requirements
- Exercises



How to write good requirements

0601-28

Example of smart numbering and linking

- Allows human correlation
- Shows where prevention is being performed
- Double entry (bookkeeping)
- Need columns not shown, e.g.
 - S.23 traces back to N(eed).23
 - traces back to stakeholder JK
- Expanded front end of the Requirements Traceability Matrix (RTM)
- Builds risk management into the front end of the process instead of being an add-on elsewhere
- Partial table shown
- Colours group scenarios (human factor)
- Missing link identified by colour
 - (defaults X.0.Z.0)

Scenario	Function	Risks	Risk is prevented in (->)	Prevents risk identified in (<-)
S.23	F.23.1	R.23.1.1	F.55.6	None
S.23	F.23.1	R.23.1.2	F.73.8	None
S.23	F.23.2	R.23.2.1	F.76.5	None
S.23	F.23.3	None	N/A	None
S.55	F.55.6	None	N/A	R.23.1.1
S.73	F.73.8	R.73.8.1	S.245.5	R.0.0.0
S.76	F.76.5	R.76.1.1	S.367.5	R.43.2.1

How to write good requirements0601-29

Example of smart numbering and linking in conceptual tool

- Scenario shown in centre
 - Separates input and output
 - Eyes follow flow
- Missing link identified by colour
 - (defaults X.0.Z.0)
- Not part of RTM
- Allows computer tool to check for completeness of needs, and later requirements
- Colours group scenarios (human factor)
- Partial table shown
- View parts of table (database) at a time
 - By lines, scenarios, inputs, outputs, missing links, etc.

Input	Comes from	Scenario	Output	Goes to
S.23.I.1	S.12.O.4	S.23	S.23.O.1	S.25.I.2
S.23.I.2	S.10.O.1	S.23	S.23.O.2	S.28.I.1
S.23.I.3	S.12.O.4	S.23	None	N/A
S.23.I.4	S.76.O.1	S.23	None	N/A
S.55.I.1	S.43.O.1	S.55	S.55.O.1	S.73.I.1
S.73.I.1	S.55.O.1	S.73	S.73.O.1	S.0.I.0
S.76.I.1	S.65.O.1	S.76	S.76.O.1	S.80.I.1

How to write good requirements0601-30

Session topics

- What to do before writing the requirements
- What to do when writing the requirements
- Using smart numbering and computers to maximize completeness
- **How to deal with some types of poorly-written requirements**
- Writing requirements for flexible systems
- What to do after writing the requirements
- Exercises



How to write good requirements

0601-31

Dealing with poorly-written requirements

- Consider the poorly-written requirement as a poorly-written scenario
- Follow examples already shown in this session and previous sessions for converting scenarios to functions
 - Break them out into atomic functions
 - Write requirements traceable to the functions
- Try to discuss them with the stakeholder to get acceptance criteria which will further clarify the meaning
- General principle when dealing with a set of poorly-written requirements – expect missing requirements
- Use the three ways to maximize completeness (Session 4)
 1. Corresponding inputs and outputs
 2. Corresponding functions
 3. Templates

How to write good requirements

0601-32

Session topics

- What to do before writing the requirements
- What to do when writing the requirements
- Using smart numbering and computers to maximize completeness
- How to deal with some types of poorly-written requirements
- **Writing requirements for flexible systems**
- What to do after writing the requirements
- Exercises



How to write good requirements

0601-33

Reading 0602 contains approach and examples

- The Luz SEGS-1 Project
- The NASA Space Transport System
- The Division Air Defense Sergeant York gun
- The Standard Central Air Data Computer (SCADC)
- The RCA Cosmac 1802 Microprocessor
- The Joint Strike Fighter

How to write good requirements

0601-34

Session topics

- What to do before writing the requirements
- What to do when writing the requirements
- Using smart numbering and computers to maximize completeness
- How to deal with some types of poorly-written requirements
- Writing requirements for flexible systems
- **What to do after writing the requirements**
- Exercises



How to write good requirements

0601-35

What to do after writing the requirements

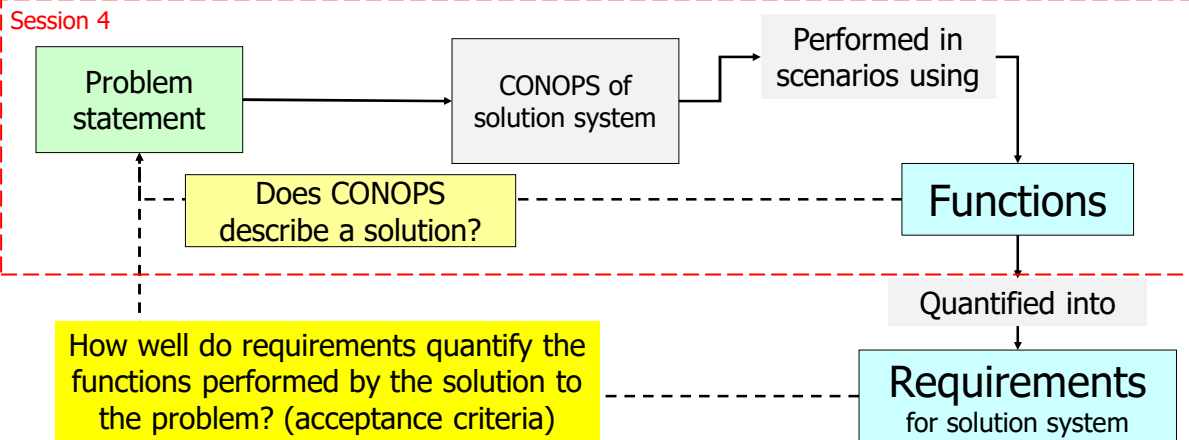
1. Remove any contradictions and duplicates from set of requirements
2. Verify how well the requirements quantify the functions performed by the system
3. Verify or ensure all requirement statements have the following additional attributes:
 1. Acceptance criteria
 2. Priority
4. If there are scenarios,
 1. Verify or ensure all remaining attributes are attached to the scenarios
5. If there are no scenarios,
 1. then try to identify remaining attributes

How to write good requirements

0601-36

Converting CONOPS to requirements

Concept map: functions and requirements



How to write good requirements

0601-37

Session topics

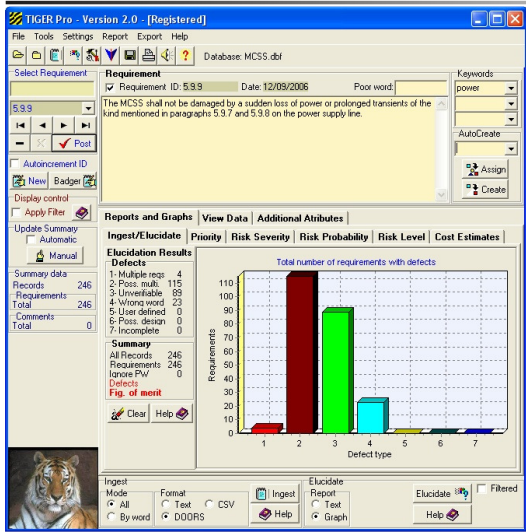
- What to do before writing the requirements
- What to do when writing the requirements
- Using smart numbering and computers to maximize completeness
- How to deal with some types of poorly-written requirements
- Writing requirements for flexible systems
- What to do after writing the requirements
- **Exercises**



How to write good requirements

0601-38

Pre Exercise 6-1 Tiger Pro demonstration



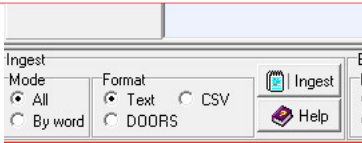
- User manual is downloadable
- See Additional resources in Knowledge components for links
- Notes
 - No duplicate links
 - Prevents errors if links change in future

How to write good requirements

0601-39


Exercise 6-1 Practical use of Tiger Pro

1. Download and install Tiger Pro
2. Run Tiger Pro
3. Verify or endure the Display Control checkbox is unchecked as shown
4. Verify or ensure ingest panel at bottom is set as shown below
 1. Mode = all
 2. Format = text



How to write good requirements

0601-40




Ingest and elucidate example file

5. Ingest and elucidate example file

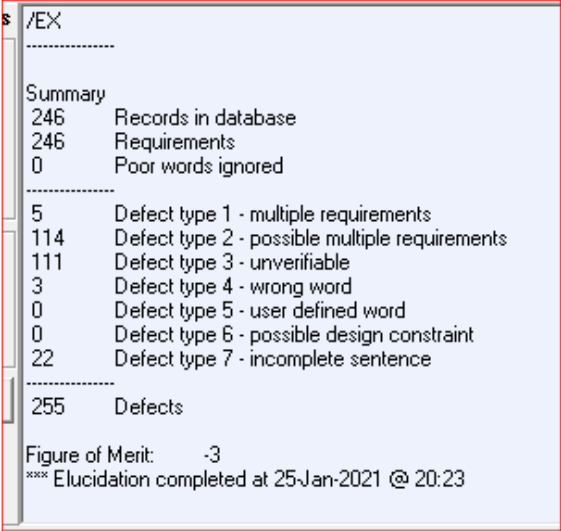
- Click on the "Ingest" button in the bottom panel.
- Scroll to the MCSS.TXT (or 0350 MCSS.TXT) on your computer
- Select it
- If you get a 'Database exists' message, then select 'Delete database' and reselect
- Tiger Pro should ingest the file
 - 246 records
- Click on the "Elucidate" button
- Tiger Pro will elucidate the file, calculate a Figure of Merit and generate a report

How to write good requirements

0601-41



Elucidation report



- Multiple Requirements in a line
 - The word "Shall" shows up more than once.
- Possible multiple requirements
 - The word "and" or "or" appears in the requirement
- Unverifiable word
 - Best practices, easy, etc and others
- Use of wrong word
 - Will, must, should

How to write good requirements

0601-42

Ignore headings (not requirements)

6. Clear headings

1. Scroll back to the start of the file using the 'start of file' button

2. It is a heading so verify of ensure the requirement ID box is unchecked

3. Select the next requirement using the forward button

4. Repeat process for remaining requirement to uncheck all headings

Start of file

Forward

Select Requirement

Requirement

5.1

MCSS SYSTEM DESCRIPTION

Requirement ID

0601-43

Thinking about requirements

6. Adjust requirement example

1. Select Requirement 5.11.7 via the Select Requirement panel

1. Select the number from the drop down list

note it is currently showing 5.10.11

2. Scroll the report window up to (near top)

5.11.7 Postponed courses shall be rescheduled with the mutual agreement of the Government and the MCSS supplier.

and defect type 2

3. Insert the word "and" in the poor word box

Poorword: |

4. Click on the report window

Select Requirement

5.10.11

Post

Autoincrement ID


New Badger

Display control

Apply Filter

How to write good requirements

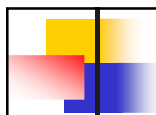
0601-44



Exercise 6-1 Practical use of Tiger Pro (cont.)

7. Scroll up to the start of the report
8. Spend at least 30 minutes examining the remaining defective requirements to determine if they are truly defective.
 1. If not, enter the appropriate poor word in the poor word box.
9. Elucidate the file after the changes
10. Ingest and elucidate **your** Exercise 4.11 requirements
 1. Save the requirements you wrote in Exercise 4.11 as a text file (*.txt)
 2. Ingest the requirements
 3. Elucidate the requirements


How to write good requirements 0601-45



Exercise 6-1 Practical use of Tiger Pro (cont.)

11. Ingest and elucidate **your** Exercise 5.21 requirements
 1. Save the requirements you wrote in Exercise 5.21 as a text file (*.txt)
 2. Ingest the requirements
 3. Elucidate the requirements
12. Prepare a <5 minute presentation containing
 1. The three elucidation reports
 2. The exercise problem formulated per COPS problem formulation template
 3. A compliance matrix for the exercise
 4. Lessons learned from exercise
13. Save as a PowerPoint file in format Exercise6.1-abcd.pptx
14. Post/email presentation as, where and where instructed

How to write good requirements 0601-46




Exercise 6-2 knowledge reading

1. Prepare a brief on two main points in reading 0602 (< 5min)
2. Presentation to contain
 1. Formulated problem per COPS problem formulation template
 2. A summary of the content of the reading (<1 minute)
 3. The compliance matrix
 4. This slide and version number of session
 5. The main points
 6. The two briefings
 7. Reflections and comments on reading (<2 minute)
 8. Comparisons of content with other readings and external knowledge
 9. Why you think the reading was assigned to the module
 10. Lessons learned from module and source of learning e.g. readings, exercise, experience, etc. (<2 minutes)
3. Save as a PowerPoint file as Exercise6.2-abcd.pptx
4. Post/email presentation as, when and where instructed
5. Brief on one main point

How to write good requirements

0601-47



Meeting the objectives

#	Objectives	Met
1	To explain what to do before writing the requirements	18-20
2	To explain what to do while writing the requirements	22-32
3	To explain what to do immediately after writing the requirements	41-42
4	To explain how to deal with some types of poorly-written requirements	37
5	To discuss attributes of requirements for flexible systems	39
6	To provide the opportunity to use a tool to convert some poorly-written requirements into well-written requirements	44-51
7	To provide the opportunity to exercise the 5 levels of knowledge in the updated Blooms taxonomy	45-52

How to write good requirements

0601-48

Objectives of Session 7

1. To explain the requirements for well-written requirement
2. To explain the structure of a well-written requirement
3. To explain the use of spelling, grammar and vocabulary
4. To explain how to test the text statement for conformance to the specification for a well-written requirement
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

0601-49

Any questions ?

1. Best
2. Worst
3. Missing



Email: beyondsystemsthinking@yahoo.com.

Subject: <class title> BWM Module #

How to write good requirements

0601-50