Electrical Engineering and Computer Science Dept
College of Engineering

RULES FOR NEED AND REQUIREMENT STATEMENTS

Prepared by Prof. Dr. Christopher Shneider Cerqueira

SYS530-2025

Icarus, where are you? In what region shall I seek you?

V

et iam Iūnōnia laevā
parte Samos (fuerant Dēlosque Parosque relictae),
dextra Lebinthos erat fēcundaque melle Calymne,
cum puer audācī coepit gaudēre volātū
dēseruitque ducem, caelīque cupīdine tractus
altius ēgit iter. rapidī vīcīnia sōlis
mollit odōrātās, pennārum vincula, cērās.
tābuerant cērae; nūdōs quatit ille lacertōs,
rēmigiōque carēns nōn ūllās percipit aurās.
ōraque caeruleā patrium clāmantia nōmen
excipiuntur aquā, quae nōmen trāxit ab illō.
at pater īnfēlīx nec iam pater "Īcare," dīxit;
"Īcare," dīxit, "ubi es? quā tē regiōne requīram?
Īcare," dīcēbat; pennās aspexit in undīs,
dēvōvitque suās artēs corpusque sepulcrō
condidit, et tellūs ā nōmine dicta sepultī.



CONTEXT

W	Dates	Topic	Ref	Activity
35	26-Aug	Course Introduction Systems Engineering Review Requirement Engineering (Generic Process)	[1][6] [7][9]	Groups & Names
36	2-Sep	Lego exercise 01 - First assessment	[2] [10]	Exercise Report
37	9-Sep	Definition and Structure of Requirements Functional and non-functional Requirements	[8][6] [10]	Exercise List
38	16-Sep	Elicitation Techniques (Interview / Rich Picture / use Cases)		Group Project Delivery
39	23-Sep	Rules for need and requirement statements	[2][3]	Exercise List
40	30-Sep	Attributes of need and requirement statements Boilerplates/patterns	[2]	Exercise List
41	7-Oct	[Carrer Explo]		* Reading activity
42 *	14-Oct	Midterm Session		Self-paced concept questions
		[Fall break]		

W	Dates	Topic	Ref	Activity
43	21-Oct	MBSE Trends	**	Group Project Delivery
44	28-Oct	Traceability - Organization	[5][	Group Project Delivery
45	4-Nov	Metadata: Rationale + Verification and Validation (Success Criteria)	[2][4]	Group Project Delivery
46	11-Nov	[Veterans Day]		* reading activity
47	18-Nov	Lego Exercise 02	[2] [10]	Exercise Report
48	25-Nov	System Requirement Review Presentation (G1-G4)		
		[Thanksgiving]		
48	2-Dec	System Requirement Review Presentation (G5-G8)		
49	9-Dec	Finals		Self-paced concept questions and a small case

Better introduction to Use Case representations

Use Cases

- A use case describes **the goals of a system** from the perspective of the system's users.
- The objectives are described in terms of the **functionality that the system must support**.
- Typically, the use case identifies the use case purpose(s), a primary usage pattern, and several alternative uses (variants).

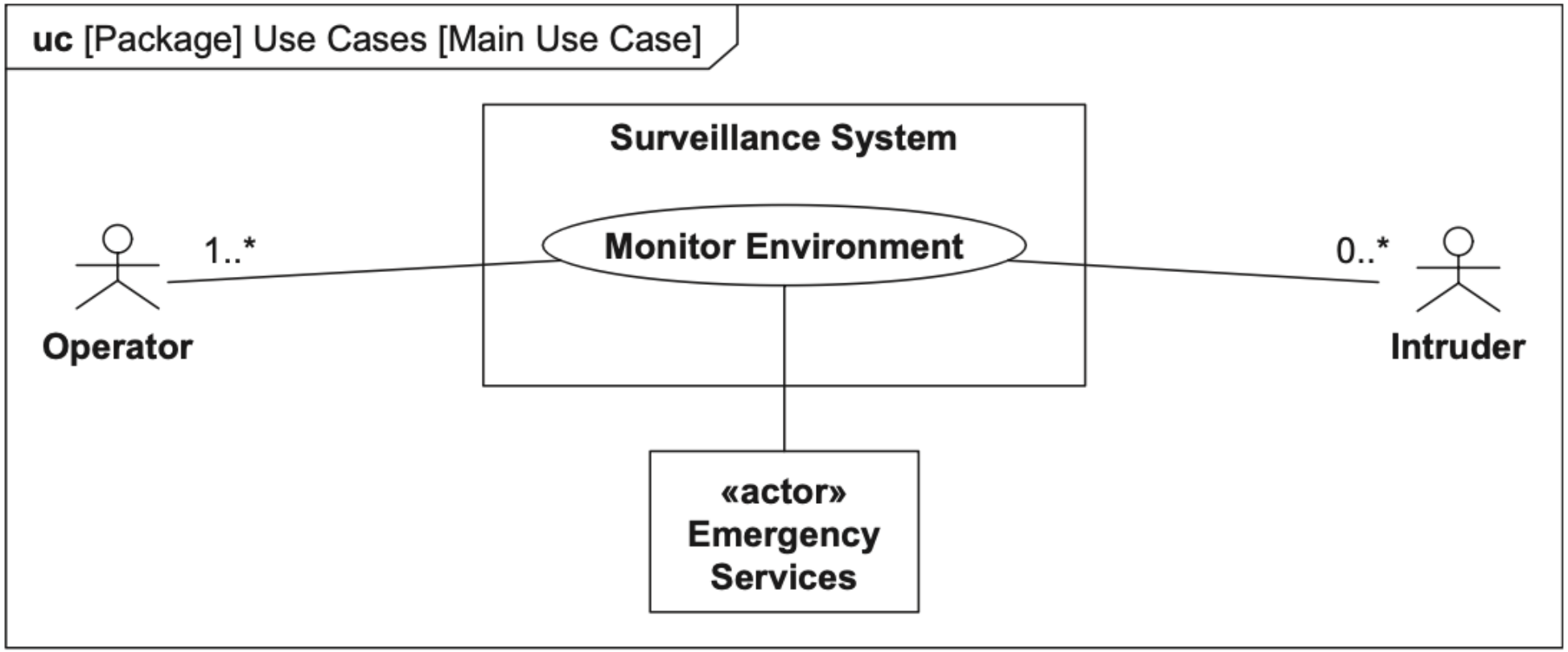


FIGURE 12.3

A use case and the actors that participate in it.

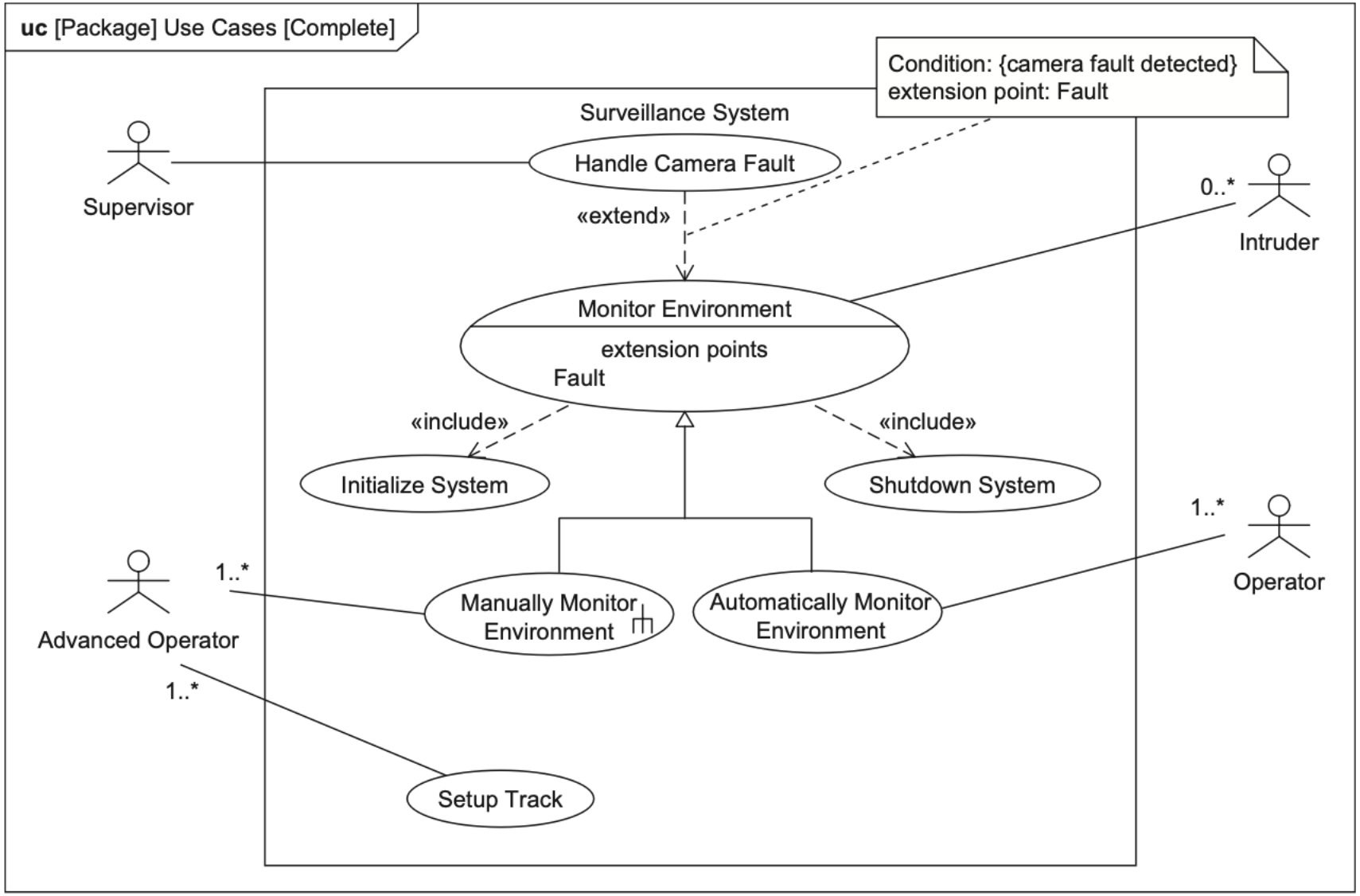


FIGURE 12.4

A set of use cases for the *Surveillance System.*

Incluside relationship

- The include relationship allows a use case, known as the base use case, to include the functionality of another use case, called the included use case.
- The included use case always runs when the base use case runs.

- *It is implicit in the definition of inclusion that any participant in a base use case can participate in an included use case, so an actor associated with a base use case does not need to be explicitly associated with any included use case.*

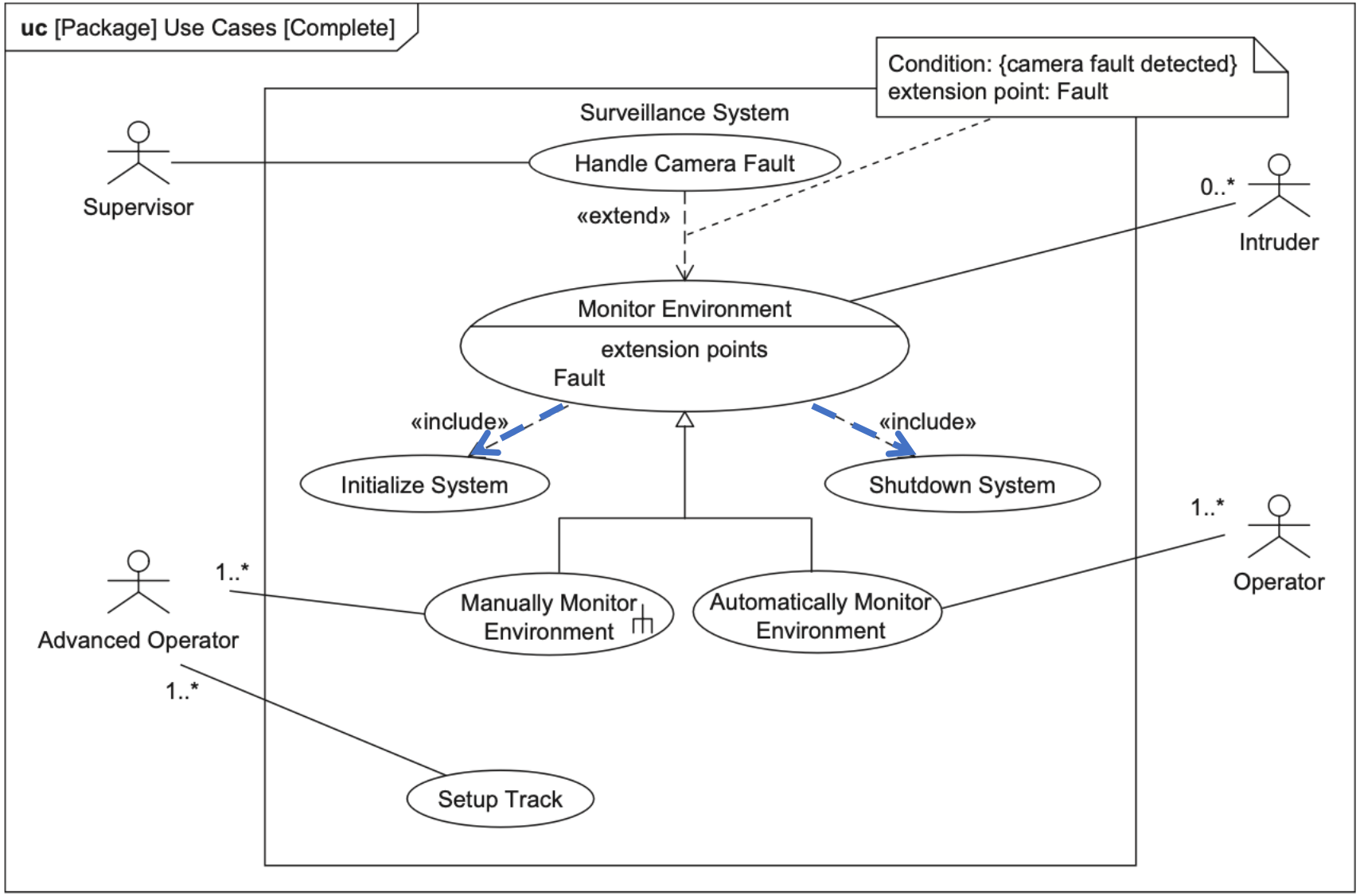


FIGURE 12.4

A set of use cases for the *Surveillance System.*

Subtle observation: it is not functional decomposition!!

- The included use cases **are not intended to represent a functional breakdown of the base use case**, but rather to describe common functionality that can be included by other use cases.

- *In a functional decomposition, the lower-level functions represent a complete decomposition of the higher-level function. On the other hand, a base use case and its included use cases often describe different aspects of the required functionality.*

Extended relationship

- A use case can also extend a base use case by using the **extension relationship.**
- The **extended use case** is a fragment of functionality that is not considered part of the functionality of the base use case.

- *It often describes some exceptional behavior in the interaction, such as error handling between the subject and the actors that do not directly contribute to the goal of the base use case*

Extension does not generate dependency

- Unlike an included use case, the base use case does not rely on an extended use case.
- However, an extended use case may depend on what's happening in your basic use case;
  - For example, the extended use case might assume that some exceptional circumstance in the base use case arose.

- There is no implication that an actor associated with the base use case participates in the extended use case, and the extended use case may in fact have entirely different participants.

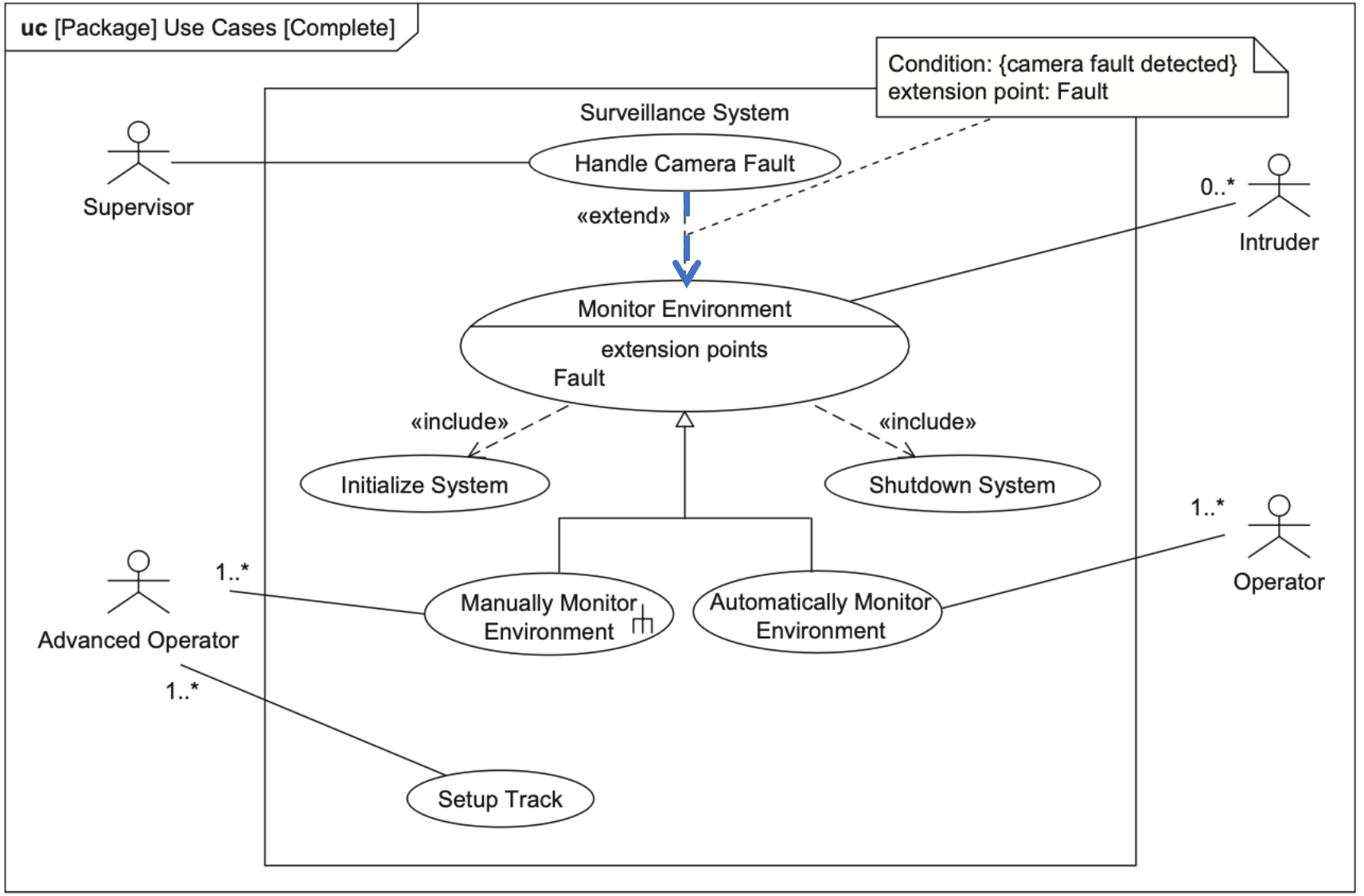


FIGURE 12.4

A set of use cases for the *Surveillance System.*

About relationship graphical representation

- Inclusion and extension are shown using **dashed lines with an open arrowhead** at the included and extended ends, respectively.
- An include line has the keyword «include» and an extension line has the keyword «extend».
- The direction of the arrows should be read as the rear end includes or extends the end of the head.

- Thus,
  - a basic use case includes an included use case, and
  - an extended use case extends a basic use case.

Classification (heritage) Relationship

- Use cases can be classified using the standard SysML generalization ratio.
- One implication, for example, is that the scenarios for the general use case are also scenarios for the specialized use case.
  - This also means that actors associated with a general use case can also participate in scenarios described by a specialized use case.
- Classification of use cases is shown using the standard SysML generalization symbol.

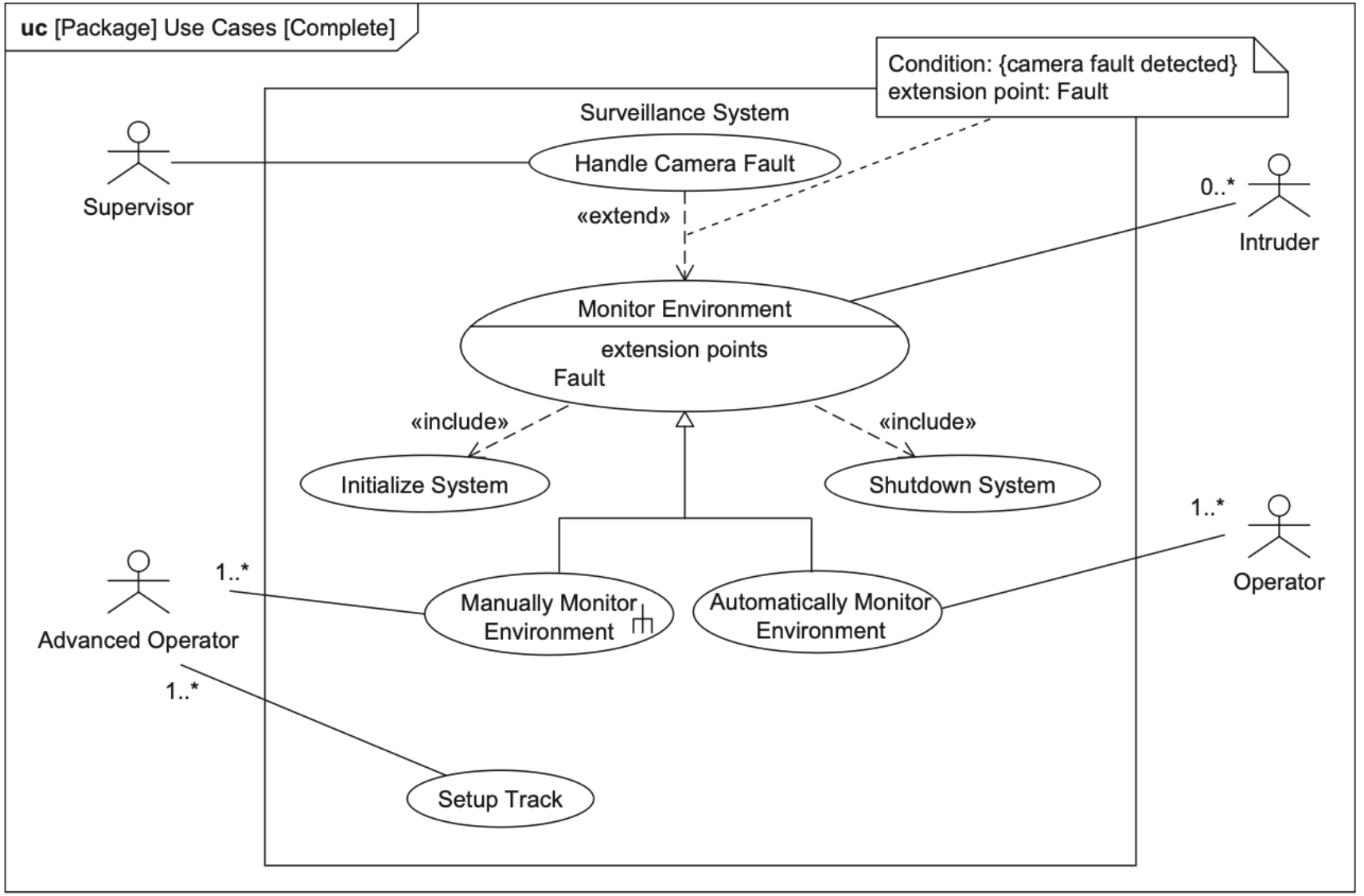
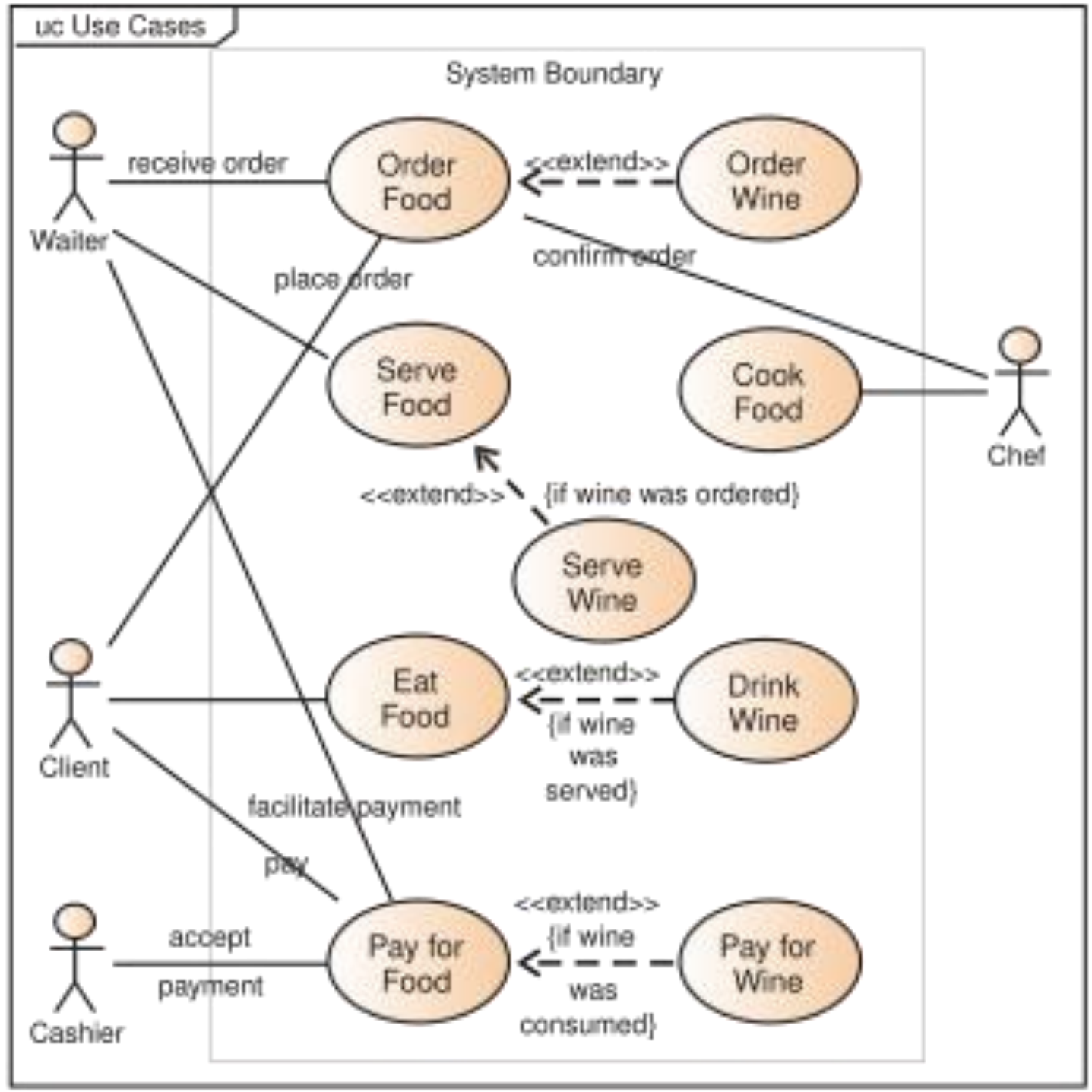


FIGURE 12.4

A set of use cases for the *Surveillance System.*



https://en.wikipedia.org/wiki/Use_case

Textual description of use cases

- A text-based use case description can be used to provide additional information to support the use case definition.
- **This description can contribute significantly to the value of the use case**.
- Description text can be captured in the template as a single or multiple comments. You can also treat each step in a use case description as a SysML requirement.

A typical use case description might include the following fields

- *Pre-conditions* - The conditions for the use case to begin.
- *Post-conditions* - the conditions after the completion of the use case.
- *Primary flow* - The most frequent scenario or scenarios of the use case.
- *Alternate and/or exception flows* - the least frequent or non-nominal scenarios. Exception flows can reference extension points and often represent flows that don't directly support the goals of the primary flow.

Here is an extract from the use case description for *Monitor Environment*:

Pre-condition

The *Surveillance System* is powered down.

Primary Flow

The *Operator* or *Operators* will use the *Surveillance System* to monitor the environment of the facility under surveillance. An *Operator* will initialize the system (see *Initialize System*) before operation and shut the system down (see *Shutdown System*). During normal operation, the system's cameras will automatically follow preset routes that have been set to optimize the likelihood of detection.

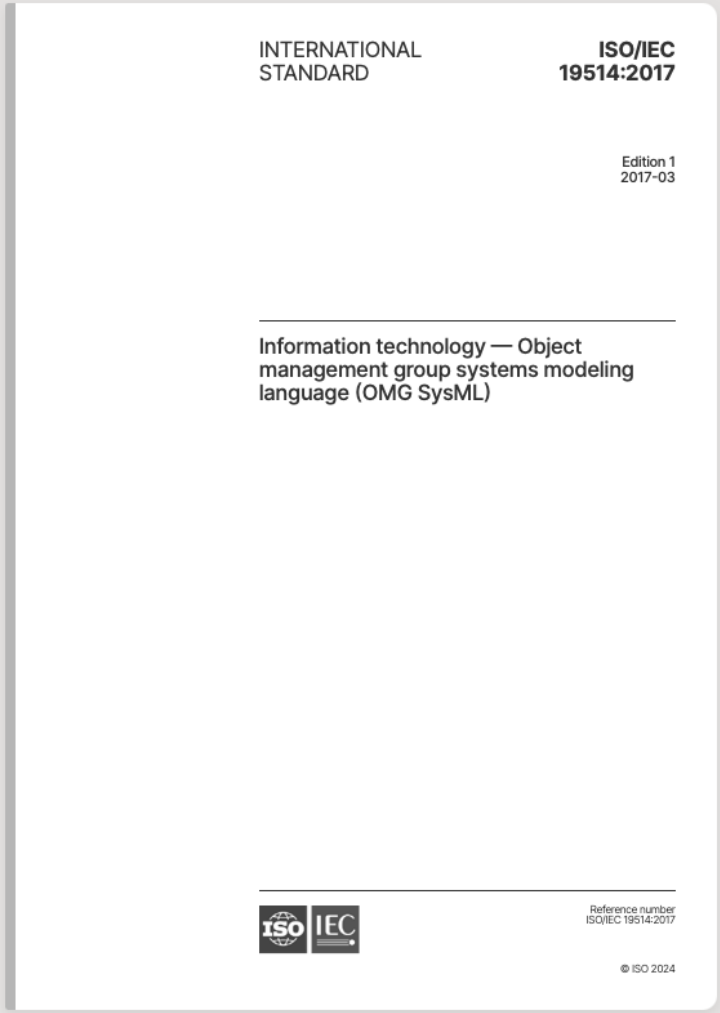
If an *Intruder* is detected, an alarm will be raised both internally and with a central monitoring station, whose responsibility it is to summon any required assistance. If available, an intelligent intruder tracking system—which will override the standard camera search paths—will be engaged at this point to track the suspected intruder. If an intelligent intruder tracking system is not available, the *Operators* are expected to maintain visual track of the suspected intruder and pass this knowledge on to the *Emergency Services* if and when they arrive.

Alternate Flow

Immediately after system initialization but before normal operation begins, it is possible that a fault will arise, in which case it can be handled (c.f. *Fault* extension point), but faults will not be handled thereafter.

Post-condition

The *Surveillance System* is powered down.



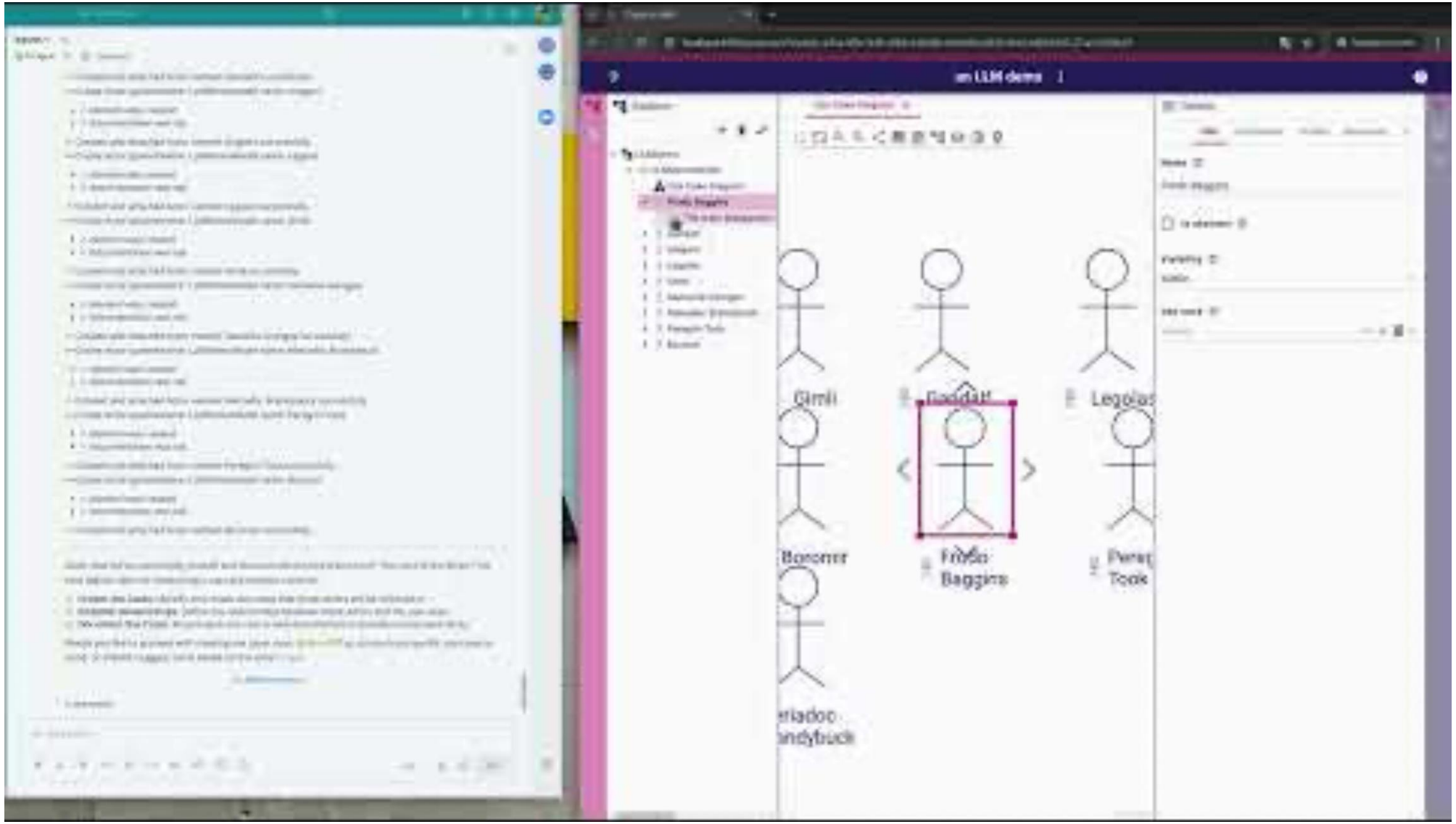
Read sample

ISO/IEC 19514:2017

Information technology — Object management group systems modeling language (OMG SysML)

Published (Edition 1, 2017)

Some research: LLM + UseCase



https://www.youtube.com/watch?v=tNYg9kx4O7s

Functional and non-functional requirements

Functional requirements are things the product must do.

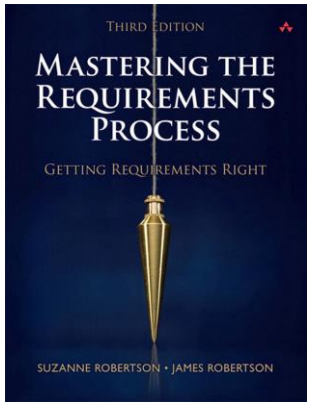
Functional Requirements

A functional requirement describes an action that the product must take if it is to be useful to its operator—they arise from the work that your stakeholders need to do. Almost any action (calculate, inspect, publish, or most other active verbs) can be a functional requirement.

Non-functional requirements are qualities the product must have.

Non-functional Requirements

Non-functional requirements are properties, or qualities, that the product must have if it is to be acceptable to its owner and operator. In some cases, non-functional requirements—these specify such properties as performance, look and feel, usability, security, and legal attributes—are critical to the product's success, as in the following case:



Characteristics of a requirement statement



C1 - Necessary	A needed transformation
C2 - Appropriate	Correct abstraction
C3 - Unambiguous	One interpretation
C4 - Complete	Sufficient description
C5 - Singular	State a single aspect
C6 - Feasible	Can be realized
C7 - Verifiable	Can be proven
C8 - Correct	Accurate transformation
C9 - Conforming	Follows style guide

Characteristics of the Set of Requirements

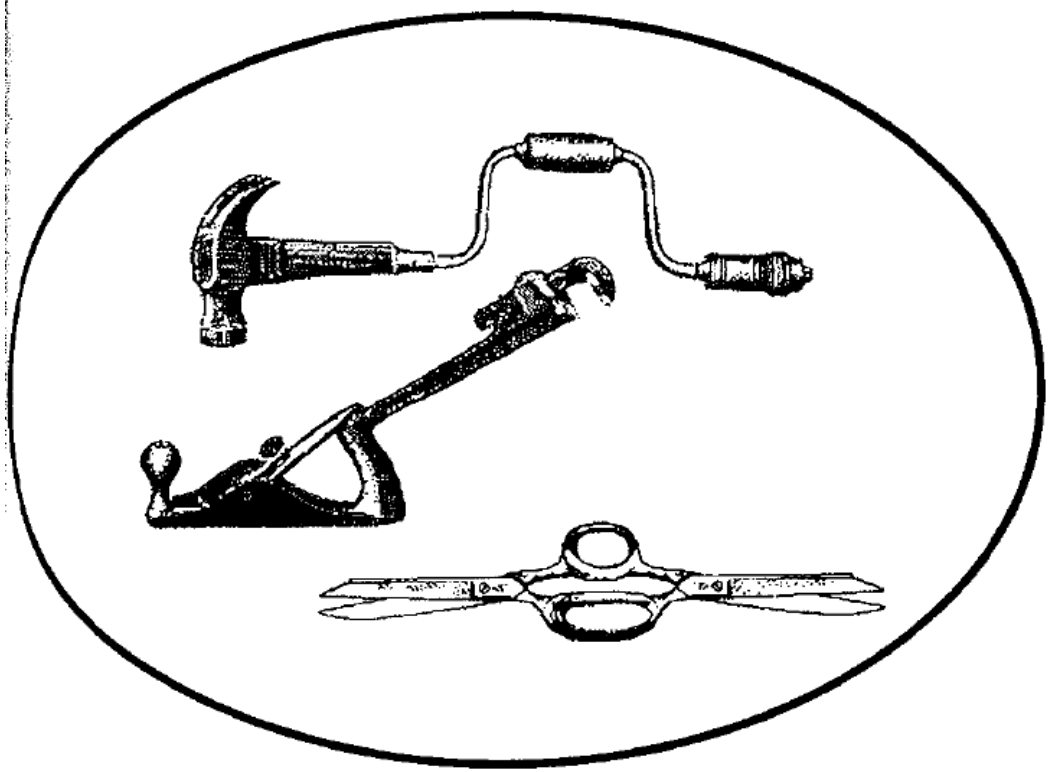


C10 - Complete	… stands alone such that it sufficiently meet the needs without requiring other sets of needs or requirements at the appropriate level of abstraction.
C11 - Consistent	.. are unique, do not conflict with or overlap with other needs in the set
C12 - Feasible	…can be realized within entity constraints (for example, cost, schedule, technical) with acceptable risk
C13 - Comprehensible	…must be written such that it is clear as to what is expected of the entity and its relation to the system of which it is a part.
C14 - Able to be validated	… must be able to be proven

Requirement writing…



A little manure on the boots may disturb city folks, but in requirements work, you learn not to mistake appearance for value.



How bad, poor and conflicting requirements create hassles:

#1) No requirements

- No requirements implies assumptions & guesses and therefore there is no confidence.
- It is very difficult to test a product/application without any baseline. And these results in more work, more bugs from client and more suffering for the project.
  - How would you report an issue about system crash when there is no definition of how the behaviour should be handled has been available?
  - How would you convey that loading time of 100 seconds for home page is unacceptable when there is no relevant requirement for performance?

#2) Poor requirements

- The quote, ***Knowing something incomplete is dangerous than not knowing it at all***, is very true when it comes to dealing with a poor requirement.
- Interpreting a poor requirement and implementing the same is a big risk.
  - How would you confirm that the pop-up showing search results is valid or not when the only requirement mentioned was – *search results should be proper*, and you are not sure which criteria should be considered while search.
  - How would you interpret this – *Forgot password should be implemented to facilitate user to regenerate/reset forgot password*. Unknown about which work flow the customer wants for forgot password, developer implements what he/she thinks is best and the conflicts start.

#3) Conflicting Requirements

- Asking someone to do two different things at same time is just getting him/her confused and system too is not an exception.
- How would you test an application with requirements mentioned are as below:
  - Application should always open on home page.
  - Users are expected to sign in to access the application.
- What would you decide the priority when the requirement document is as below:
  - The gaming application should promote user to next level if the user scores 1000.
  - User should be redirect to free subscription page once he scores 1000.

Requirement problems

REQUIREMENTS ANALYSIS ROI TO CUSTOMER

PROJECT PERFORMANCE INTERNATIONAL

Parameter	Value
Contract value	$4B
Requirements on the Ship	27,000, only fair in quality
Consequence if uncorrected	At least 20% loss of capability, costing at least $800M; or Rework costs exceeding 20%
Cost of fixing the requirements	$8M (0.2% of contract value)
Return on Investment	Approximately 100:1

P1314-005216-3 © Copyright Project Performance (Australia) Pty Ltd 1998-2018 helping projects succeed...

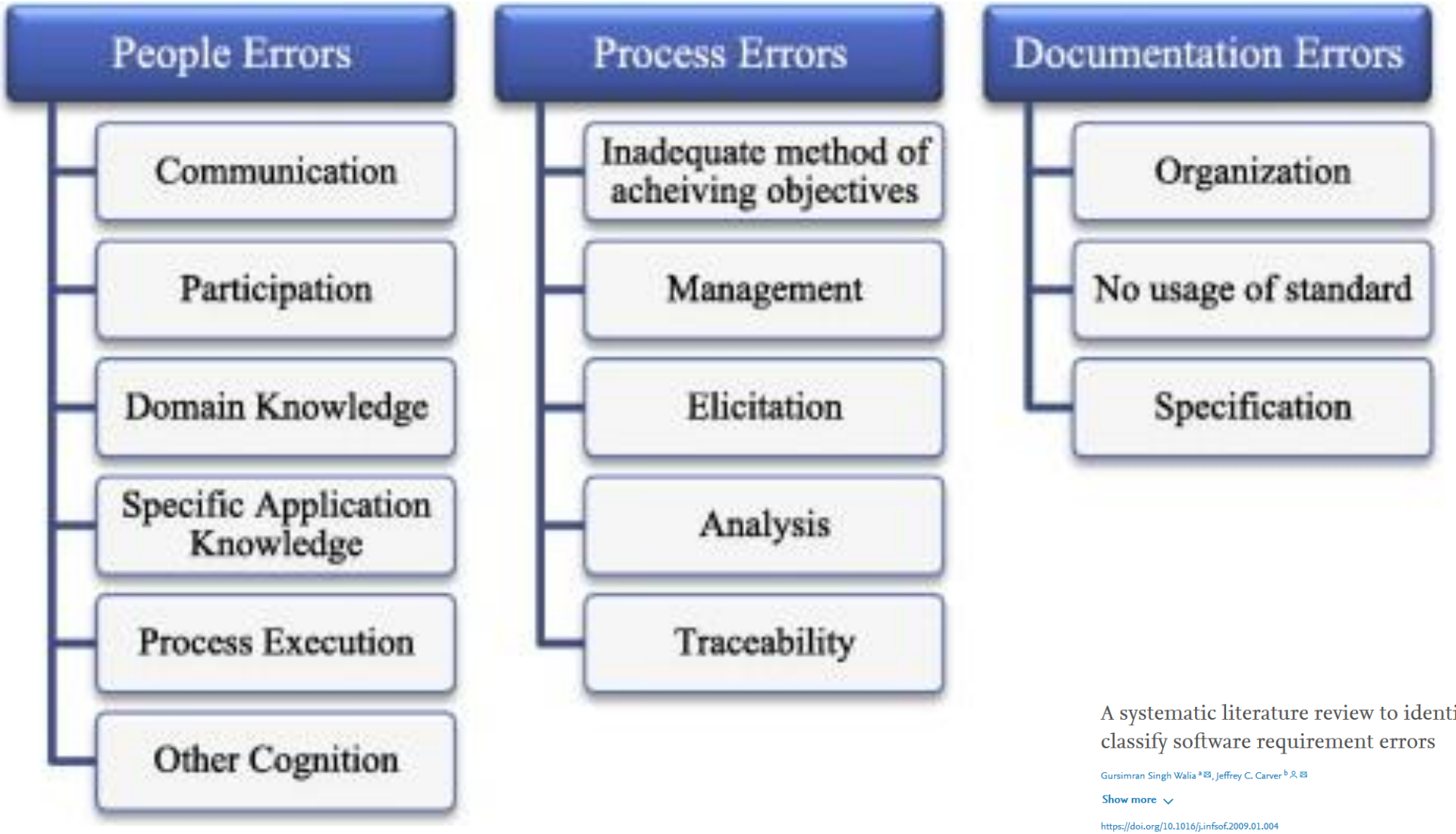
IMPACT OF REQUIREMENTS DEFECTS

PROJECT PERFORMANCE INTERNATIONAL

Organization/Project	Overruns Attributed to Requirements Problems
NASA over two decades (Werner Gruhl)	70% of overruns
U.S. Census Bureau project 2009	80% cost overrun locked in solely due to poor requirements
Marine One Helicopter Program	83% cost overrun attributed by Lockheed to requirements problems
Schwaber, 2006; Weinberg, 1997; Nelson et al, 1999	"Requirements errors are the single greatest source of defects and quality problems"
Hofmann and Lehner, 2001	"Deficient requirements are the single biggest cause of software project failure"
Standish Group, The Chaos Report on 8300 IT projects	60.9% of an average 89% cost overrun

P1314-005216-3 © Copyright Project Performance (Australia) Pty Ltd 1998-2018 helping projects succeed...

Classification of requirements' errors



A systematic literature review to identify and classify software requirement errors

Gursimran Singh Walia [a], Jeffrey C. Carver [b]

Show more

https://doi.org/10.1016/j.infsof.2009.01.004

Get rights and content

Defining the correct heuristics to understand a requirement

Mary had a little lamb.
Its fleece was white as snow.
And everywhere that Mary went,
The lamb was sure to go.

Mary *had* a little lamb.
(She no longer has the lamb.)

Mary had *a* little lamb.
(She had only one lamb, not several.)

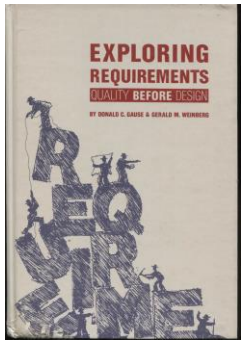
Mary had a *little* lamb.
(It really was surprisingly small.)

Mary had a little *lamb.*
(She didn't have a dog, cat, cow, goat, or parakeet.)

Mary had a little lamb.
(John still has his little lamb.)

s a tour de force, we offer all five words emphasized:

Mary had a little lamb.
(As contrasted with Pallas, who still has four large turtles.)



Rules

[INCOSE Guide for Writing Requirements – Chapter 4]



Rationale

- These rules help to formulate the requirements.
- Those rules relates to the characteristics
- 14 General Rules w/ 41 Specific Rules:
  - Accuracy (9),
  - concision (2),
  - non-ambiguity (6),
  - singularity (6),
  - completeness (2),
  - realism (1),
  - conditions (2),
  - uniqueness (2),
  - abstraction (1),
  - quantifiers (1),
  - tolerance (1),
  - quantification (2),
  - uniformity of language (4) and
  - modularity (2)

Accuracy Rules

- **R1 - /ACCURACY/SENTENCESTRUCTURE -** Use a structured sentence.
- **R2 - /ACCURACY/USEACTIVEVOICE -** Use the active voice in the main sentence structure of the need or requirement statement with the responsible entity clearly identified as the subject of the sentence.
- **R3 - /ACCURACY/SUBJECTVERB -** Ensure the subject and verb of the need or requirement statement are appropriate to the level to which the need or requirement refers.
- **R4 - /ACCURACY/USEDEFINEDTERMS -** Define terms.
- **R5 - /ACCURACY/USEDEFINITEARTICLES -** Use definite article “the” rather than the indefinite article “a.”
- **R6 - /ACCURACY/UNITS -** Use appropriate units when stating quantities.
- **R7 - /ACCURACY/AVOIDVAGUETERMS -** Avoid the use of vague terms.
- **R8 - /ACCURACY/NOESCAPECLAUSES -** Avoid escape clauses.
- **R9 - /ACCURACY/NOOPENENDED -** Avoid open-ended clauses.

R1 - /ACCURACY/SENTENCESTRUCTURE

- The need or requirements statement must be in the form of a complete sentence, the simplest form of which is:

<subject> <verb> <object>.

- Currently, no other form, other than textual "shall" requirement statements, have been shown to meet these characteristics.

When <condition clause>, the <subject clause> shall <action verb clause> <object clause> <optional qualifying clause>.

When <optional preconditions/trigger> the <system name> shall <system response>.

In the event of <specific event> the <system name> shall <system response>.

If <optional preconditions/trigger>, then the <system name> shall <system response>.

While <entering, exiting or in a specific state (or mode)> the <system name> shall <system response>.

Standard Meanings (per INCOSE / NASA / IEEE 29148)

- Shall → Mandatory requirement
  - Defines what the system is obligated to do. // “The aircraft **shall** carry 4 passengers.”
- Must → Constraint (legal, regulatory, external)
  - Not under the system designer’s control but must be complied with. // “The aircraft **must** comply with FAA Part 23 standards.”
- Will → Statement of intent / future fact
  - Describes actions by the project or environment, not requirements for the system. // “The project **will** deliver the prototype in 2026.”
- Should → Goal / Recommendation (non-mandatory)
  - Desirable, but not strictly required. // “The aircraft **should** minimize fuel consumption.”
- May → Permission / Option
  - Indicates flexibility or allowed alternatives. // “The operator **may** select manual override.”

R2 - /ACCURACY/USEACTIVEVOICE

- Use **the active voice in the main sentence** structure of the need or requirement statement with the responsible entity clearly identified as the subject of the sentence.
- Often when the phrase “shall be” is used, the statement is in the passive voice.
- *Unacceptable*: **The Identity of the Customer shall be confirmed**. *{This is unacceptable because it does not identify the entity that is responsible/accountable for confirming the identity.}*
- *Acceptable*: **The Accounting_System shall confirm the Customer_Identity**. *{Note that “Accounting_System” , “confirm”, and “Customer_Identity” must be defined in the glossary since there are a number of possible interpretations of these terms.}*

R3 - /ACCURACY/SUBJECTVERB

- Ensure the subject and verb of the need or requirement statement are **appropriate to the level** to which the need or requirement refers.
- Regardless of the level at which the requirement is *stated*, the subject of a requirement statement must be appropriate to the level to which it *refers*.

- ***System requirements have the form*** **"The <system> shall ..."*—for example,* "The Aircraft shall …"**
- ***Subsystem requirements have the form*** **"The <subsystem> shall ..." *–for example, once the subsystems are defined:* "The Engine shall …"; "The Landing_Gear shall …".**

R4 - /ACCURACY/USEDEFINEDTERMS

C3 - Unambiguous
C7 - Verifiable
C11 - Consistent
C13 - Comprehensible
C14 - Able to be Validated

- Most languages are rich with words having several synonyms, each with a subtly different meaning based on context. In needs and requirements statements, shades of meaning will most likely lead to ambiguity and to difficulty in verification and validation. **Define the term in some form of ontology, including a glossary, data dictionary, or similar artifact that allows the reader of a need or requirement to know exactly what the writer meant when the word was chosen**.

- Unacceptable: The system shall display the current time. {This is unacceptable because it is ambiguous--what is “current”, in which time zone, to what degree of accuracy, in what format?}
- Acceptable: The system shall display the Current_Time. {Note that “Current_Time” must then be defined in the glossary in terms of accuracy, format, and time zone.}

R5 - /ACCURACY/USEDEFINITEARTICLES

- **The definite article is "*the*"; the indefinite article is "*a*."** When referring to entities, use of the indefinite article can lead to ambiguity. For example, if the need or requirement refers to "a user" it is unclear whether it means any user or one of the defined users for which the system has been designed.
- *Unacceptable*: The system shall provide a time display. *{This is unacceptable for a requirement because it is ambiguous--will any time display do? Is a one-off display of the time satisfactory? The writer's intention was most likely that they wanted the system to display continuously the current time, yet if the developer provided a constant display of "10:00 am" (or even a one-off display of any time), they could argue (albeit unreasonably) that they have met the requirement; yet they would have clearly failed to meet the customer's need and intent.}*
- *Acceptable*: The system shall display the Current_Time. *{Note that "Current_Time" must be defined in the glossary since there are a number of possible meanings and formats of the more- general term "current time." There may also be other requirements addressing where time needs to be displayed, accuracy, etc. if not addressed in the glossary or data dictionary.}*

R6 - /ACCURACY/UNITS

C3 - Unambiguous
C4 - Complete
C7 - Verifiable
C8 - Correct

- **All numbers should have units of measure** explicitly stated in terms of the measurement system used or the thing the number refers. Within a project, a common measurement system must be used consistently. For example, don't mix both US and metric units of measure within any of the project's artifacts.

- *Unacceptable*: The Circuit_Board shall have a storage temperature less than 30 degrees. *{This is unacceptable because the units used are incomplete.}*
- *Acceptable*: The Circuit_Board shall have a storage temperature of less than 30 degrees Celsius.

C3 - Unambiguous
C4 - Complete
C7 - Verifiable

R7 - /ACCURACY/AVOIDVAGUETERMS

- Avoid the use of **vague terms**.
  - Vague adjectives can lead to ambiguous, unverifiable requirements that do not reflect accurately the stakeholder expectations.
  - Adverbs qualify actions in some way and are particularly troublesome. Avoid vague adverbs, such as “usually”, “approximately”, “sufficiently”, and “typically”.
  - Vague adverbs can lead to ambiguous, unverifiable requirements that do not reflect accurately the stakeholder expectations.
- Avoid words that provide vague quantification, such as “some”, “any”, “allowable”, “several”, “many”, “a lot of”, “a few”, “almost always”, “very nearly”, “nearly”, “about”, “close to”, “almost”, and “approximate”, .
- Avoid vague adjectives such as “ancillary”, "relevant”, “routine”, “common”, “generic”, “significant”, “flexible”, “expandable”, “typical”, “sufficient”, “adequate”, “appropriate”, “efficient”, “effective”, “proficient”, “reasonable” and “customary.”

- *Unacceptable*: The Flight_Information_System shall usually be online. *{This is unacceptable because "usually" is ambiguous - is availability what is meant?}*
- *Acceptable*: The Flight_Information_System shall have an Availability of greater than xx% over a period of greater than yy hours. *{Note that "Availability" must be defined in the glossary since there are a number of possible ways of calculating that measure.}*

R8 - /ACCURACY/NOESCAPECLAUSES

- **Escape clauses give an excuse to the developer** of the system at lower levels not to bother with a need or requirement.
- Such clauses provide vague conditions or possibilities, using phrases such as “so far as is possible”, “as little as possible”, “where possible”, “as much as possible”, “if it should prove necessary”, “if necessary”, “to the extent necessary”, “as appropriate”, “as required”, “to the extent practical”, and “if practicable.”
- *Unacceptable*: The GPS shall, *where there is sufficient space*, display the User_Location. *{This is unacceptable because whether there is sufficient space is vague, ambiguous, and unverifiable. The requirement is clearer without the escape clause.}*
- *Acceptable*: The GPS shall display the User_Location. *{Note that “GPS” and ”User_Location” must be defined in the glossary. Specific performance requirements need to be defined as well such as within what time, format, and accuracy.}*

R9 - /ACCURACY/NOOPENENDED

C3 - Unambiguous
C4 - Complete
C5 - Singular
C7 - Verifiable

- Open-ended clauses imply there is more required without stating exactly what. **Avoid non-specific phrases such as "including but not limited to", "etc." and "and so on."**
- Open-ended clauses can lead to ambiguous, unverifiable needs and requirements that do not reflect accurately the stakeholder's expectations and needs and can create ambiguity in the mind of the reader.
- *Unacceptable*: The ATM shall display the Customer Account_Number, Account_Balance, and so on. *{This is unacceptable because it contains an opened list of what is to be displayed.}*
- *Acceptable*: *{Split into as many requirements as necessary to be complete. Note that the types of customer information to be displayed needs to be defined in the glossary.}:*
  - The ATM shall display the Customer Account_Number.
  - The ATM shall display the Customer Account_Balance.
  - The ATM shall display the Customer Account_Type.
  - The ATM shall display the Customer Account_Overdraft_Limit.

CONSISION RULES

- **R10 - /CONCISION/SUPERFLUOUSINFINITIVES -** Avoid superfluous infinitives.
- **R11 - /CONCISION/SEPARATECLAUSES -** Use a separate clause for each condition or qualification.

R10 - /CONCISION/SUPERFLUOUS INFINITIVES

- We sometimes see a need or requirement that **contains more verbs than necessary to describe a basic action**, such as “The system shall be designed to be able to ...” or “The system shall be designed to be capable of ...” rather than simply “The system shall ...” Think ahead to verification. If the system is able to, or is capable of, doing something one time but fails 99 times, the system has not met the requirement.
- *Unacceptable*: The Weapon_Subsystem shall *be able to* store the location of each Ordnance. *{This is unacceptable because it contains the superfluous infinitive “be able to”. However, this is acceptable for the stakeholder need: “The stakeholders need the Weapon_Subsystem to be able to store the location of each Ordnance.”}*
- *Acceptable*: The Weapon_Subsystem shall store the Location of each Ordnance. *{Note that the terms “Weapon_Subsystem”, “Location”, and “Ordnance” must be defined in the glossary.}*

R11 - /CONCISION/SEPARATECLAUSES

- Each need or requirement should have a **main verb describing a basic function or need**. If appropriate, the main sentence may then be supplemented by clauses that provide conditions or qualifications (performance values or constraints). **A single, clearly identifiable clause should be used for each condition or qualification expressed**.
- *Unacceptable*: The Navigation_Beacon shall provide Augmentation_Data at an accuracy of less than or equal to 20 meters to each Maritime_User during Harbor_Harbor_Approach_Maneuvering (HHAM). {*This is unacceptable because it inserts a phrase in such a way that the object of the sentence is separated from the verb*.}
- *Acceptable*: The Navigation_Beacon shall provide Augmentation_Data to each Maritime_User engaged in Harbor_Harbor_Approach_Maneuvering (HHAM), at an accuracy of less than or equal to 20 meters. {*This places the basic function in an unbroken clause followed by the sub-clause describing performance*.}

NON-AMBIGUITY

- **R12 - /NONAMBIGUITY/CORRECTGRAMMAR -** Use correct grammar.
- **R13 - /NONAMBIGUITY/CORRECTSPELLING -** Use correct spelling.
- **R14 - /NONAMBIGUITY/CORRECTPUNCTUATION -** Use correct punctuation.
- **R15 - /NONAMBIGUITY/LOGICALCONDITION -** Use a defined convention to express logical expressions.
- **R16 - /NONAMBIGUITY/AVOIDNOT -** Avoid the use of “not.”
- **R17 - /NONAMBIGUITY/OBLIQUE -** Avoid the use of the oblique ("/") symbol.

R12 - /NONAMBIGUITY/CORRECT GRAMMAR

- We interpret language based on the rules of grammar**. Incorrect grammar leads to ambiguity and clouds understanding**. Care must be taken when translating needs and requirements from one language to another and when sentence structure differs depending on the language in which the original need or requirement statement was written. Punctuation varies from language to language and even between dialects of a given language.
- *Unacceptable*: The Weapon_Subsystem shall *storing* the location of all ordnance. *{This is unacceptable because the grammatical error leads to uncertainty about the meaning.}*
- *Acceptable*: The Weapon_Subsystem shall store the location of all Ordnance. *{Note that "Ordnance" must be defined in the glossary to be explicit about the types of weapons and ammunition.}*

R13 - /NONAMBIGUITY/CORRECTSPELLING

- **Incorrect spelling** can lead to ambiguity and confusion. Some words may sound the same but, depending on the spelling, will have entirely different meaning. For example, “red” versus “read” or “ordinance” versus “ordnance”. In these cases, a spell checker will not find the error.
- *Unacceptable*: The Weapon_Subsystem shall store the location of all *ordinance*. *{This is unacceptable because the word “ordinance” means regulation or law. It is unlikely that the Weapon_Subsystem is interested in the location of ordinance (regulations). In the context of a weapon system, what the authors meant to use is "ordnance" as in weapons and ammunition, not “ordinance”.}*
- *Acceptable*: The Weapon_Subsystem shall store the Location of all Ordnance. *{Note that “Location” and “Ordnance” must be defined in the glossary to be explicit about the types of weapons and ammunition.}*

R14 - /NONAMBIGUITY/CORRECTPUNCTUATION

- **Incorrect punctuation** can cause confusion between sub-clauses in a need or requirement statement. Note also that the more punctuation in the need or requirement statement, the greater the opportunity for ambiguity.
- *Unacceptable*: The Navigation_Beacon shall provide Augmentation_Data to each Maritime_User, engaged in Harbor_Harbor_Approach_maneuvering (HHA) at an accuracy of 20 meters or less at least 99.7% of the time. *{This is unacceptable because the incorrectly placed comma in this sentence confuses the meaning, leading the reader to believe that the accuracy is related to the maneuver rather than to the augmentation data.}*
- *Acceptable*: The Navigation_Beacon shall provide Augmentation_Data to each Maritime_User engaged in Harbor_Harbor_Approach_maneuvering (HHA), at an accuracy of less than or equal to 20 meters greater than 99.7% of the time. *{The positioning of the comma now makes it clear that the accuracy and availability relate to the data.}*

R15 - /NONAMBIGUITY/LOGICALCONDITION

- **Define a convention for logical expressions such as "[X AND Y]", "[X OR Y]", [X XOR Y]", "NOT[X OR Y]".**
- As with the other rules and characteristics, we want to keep requirement statements as one thought with singular statements. Thus, we avoid using "and" when it involves tying two thoughts together. However, it is acceptable to use "AND", "OR", "XOR", and "NOT" in a logical sense when talking about conditions to which the verb applies.
- *Unacceptable*: The Engine_Management_System shall disengage the Speed_Control_Subsystem when the Cruise_Control is engaged, and the Driver applies the Accelerator. *{This is unacceptable because of the ambiguity of "and" could be confused with combining two separate thoughts. Instead use the form of a logical expression [X AND Y].}*
- *Acceptable*: The Engine_Management_System shall disengage the Speed_Control_Subsystem, when [the Cruise_Control is engaged AND the Driver applies the Accelerator].

R16 - /NONAMBIGUITY/AVOIDNOT

- The presence of “not” in a need or requirement statement implies “not ever”, which is impossible to verify in a finite time.
- **Rewriting the need or requirement statement to avoid the use of “not” results in a need or requirement statement that is clearer and is verifiable or able to be validated.**
- *Unacceptable:* The <system> shall not fail. *{This is unacceptable because verification of the requirement would require infinite time.}*
- Acceptable: The <system> shall have an Availability of greater than or equal to 95%. Or The <system> shall have a Mean Time Between Failures (MTBF) of 6 months.
- There may be other cases such as “The <system> shall not be red in color.” Which is stating a constraint and is verifiable.

R17 - /NONAMBIGUITY/OBLIQUE

- **The oblique symbol (“/”), or “slash”, has so many possible meanings that it should be avoided**. An exception to this rule is where the oblique symbol is used in SI units (for example “km/h”).

- *Unacceptable*: The User_Management_System shall Open/Close the User_Account in less than 1 second. *{This is unacceptable because it is unclear as to what is meant: open, close, or both?}*
- *Acceptable (Split into two requirements)*:
  - The User_Management_System shall Open the User_Account in less than 1 second.
  - The User_Management_System shall Close the User_Account in less than 1 second.

SINGULARITY

- **R18 - /SINGULARITY/SINGLESENTENCE -** Write a single sentence that contains a single thought conditioned and qualified by relevant subclauses.
- **R19 - /SINGULARITY/AVOIDCOMBINATORS -** Avoid combinators.
- **R20 - /SINGULARITY/AVOIDPURPOSE -** Avoid phrases that indicate the purpose of the need or requirement statement.
- **R21 - /SINGULARITY/AVOIDPARENTHESES -** Avoid parentheses and brackets containing subordinate text.
- **R22 - /SINGULARITY/ENUMERATION -** Enumerate sets explicitly instead of using a group noun to name the set.
- **R23 - /SINGULARITY/CONTEXT -** When a need or requirement is related to complex behavior, refer to the supporting diagram or model.

R18 - /SINGULARITY/SINGLESENTENCE

C3 - Unambiguous
C4 - Complete
C5 - Singular
C7 - Verifiable
C9 - Conforming
C13 - Comprehensible

- Based on the concepts of allocation, traceability, validation, and verification, requirement statements must contain **a single thought allowing that single thought to be allocated, the resulting single thought children requirements** to trace to their allocated parent, requirements to trace to a single thought source and allowing validation and verification of the single thought.
- If multiple actions are needed for a single condition, each action should be repeated in the text of a separate need or requirement statement along with the triggering condition, rather than stating the condition and then listing the multiple actions to be taken.

- *Unacceptable*: When in the ACTIVE State, the Record_Subsystem shall display the Name of each Line_Item *and* shall record the Location of each Line_Item, without obscuring the User_ID. *{This is unacceptable because the sentence contains two requirements and the qualification only applies to the first requirement, not the second.}*
- *Acceptable (Split into two requirements)*: When in the ACTIVE State, the Record_Subsystem shall display the Name of each Line_Item, without obscuring the User_ID. When in the ACTIVE State, the Record_Subsystem shall record the Location of each Line_Item.
- *Unacceptable*: In the event of a fire:
  - • Power to electromagnetic magnetic fire door catches shall be turned off.
  - • Security entrances shall be set to Free_Access_Mode.
  - • Fire escape doors shall be unlocked.
- *{This is unacceptable because the condition "in the event of a fire" is stated following a list of actions to be taken. Also, note the actions are written in passive voice which violates R2. Each action needs to be communicated in a separate requirement statement. – for multiple conditions for a single action see R28.}*
- *Acceptable*:
  - - In the event of a fire, the Security_System shall turn off power to electromagnetic magnetic fire door catches.
  - - In the event of a fire, the Security_System shall set security entrances to the Free_Access_Mode.
  - - In the event of a fire, the Security_System shall unlock fire escape doors.

R19 - /SINGULARITY/AVOIDCOMBINATORS

- Combinators are words that join clauses, such as “and”, “or”, “then”, “unless”, “but”, “as well as”, “but also”, “however”, “whether”, “meanwhile”, “whereas”, “on the other hand”, and “otherwise.” Their presence in a requirement usually indicates that multiple requirements should be written.
- *Unacceptable: The “command the sidelights” function shall* ensure the regulatory consistency of lights illumination *by ensuring* the illumination of sidelights is maintained during the illumination of the side lamps, or the head lights, or the fog lights or the rear fog lights or a combination of these lights. *{Again, this is unacceptable because it is non-singular, uses dubious grammar, contains elements of purpose, and is generally confusing.}*
- *Acceptable:* The Control_Sidelights function shall illuminate the Sidelights while any combination of the following conditions are true: the Side_Lamps are illuminated, the Head_Lamps are illuminated, the Front_Fog_Lamps are illuminated, the Rear_Fog_Lamps are illuminated. *{Rules 16 and 17 have also been applied to remove combinators and clarify the exact conditions. Note that the use of “any” in this example is not ambiguous and thus is an exception to R24 and R32.}*

R20 - /SINGULARITY/AVOIDPURPOSE

- The text of a need or requirement statement **does not have to carry around extra baggage such as the purpose for its existence**. Expressions of purpose are often indicated by the presence of phrases such as "……to", "in order to", "so that", and "thus allowing." This extra information should be included in the rationale attribute

- *Unacceptable*: The Operation_Logger shall record each Warning_Message produced by the system. *{This is unacceptable because of the superfluous words "produced by the system".}*
- *Acceptable*: The Operation_Logger shall record each Warning_Message.

R21 - /SINGULARITY/AVOIDPARENTHESES

- If the text of a need or requirement statement contains **parentheses or brackets, it usually indicates the presence of superfluous information that can simply be removed or communicated in the rationale**. Other times, brackets introduce ambiguity. Conventions for the use of parentheses or brackets may be agreed upon for specific purposes, but such conventions should be documented at the start of the needs or requirements document.
- *Unacceptable*: The Control_Unit shall disconnect power from the Boiler when the water temperature exceeds 85 °C (usually towards the end of the boiling cycle.) *{This is unacceptable because of the parenthetical phrase as well as the ambiguous word "usually".}*
- *Acceptable*: The Control_Unit shall disconnect power from the Boiler when the water temperature is greater than 85 °C.

C2 – Appropriate
C3 – Unambiguous
C5 – Singular

R22 - /SINGULARITY/ENUMERATION

- If a number of functions are implied, a need or requirement statement should be written for each.
- **The use of a group noun to combine functions or entities is often ambiguous because it leaves membership of that group in doubt.**
- *Unacceptable:* The thermal control system shall manage temperature-related functions. *{This is unacceptable because there is ambiguity regarding which functions are to be managed. The specific functions should be enumerated explicitly in separate requirement statements.}*
- *Acceptable (three separate requirements):*
  - The Thermal_Control_System shall update the display of Current_System_Temperature every 10 +/- 1 seconds.
  - The Thermal_Control_System shall maintain the Current_System_Temperature between 95°C and 98°C.
  - The Thermal_Control_System shall store the history of Current_System_Temperature.

R23 - /SINGULARITY/CONTEXT

- Sometimes it can be very difficult to express a complicated need or requirement in words, and it is better simply to **refer to a diagram or model**. An example is a requirement on voltage at turn on which involves a magnitude, rise time, over shoot, and dampening time along with tolerances. Stating all these values in the same requirement could be seen to violate our combiner and multiple thought rules.
- *Unacceptable*: The control system shall close Valves A and B within 5 seconds of the temperature exceeding 95 °C and within 2 seconds of each other. *{This is unacceptable because of the confusing set of conditions. In this case what is meant will be clear if expressed in the form of a diagram.}*
- *Acceptable*: When the Product temperature exceeds 95 °C, the Control_System shall close Input_Valves as specified in timing diagram 6. *{Note that this assumes, of course, that the timing diagram itself is not ambiguous. If it is not possible to remove the ambiguity using a diagram, it may be better to split the requirement into two.}*

COMPLETNESS

- **R24 - /COMPLETENESS/AVOIDPRONOUNS -** Avoid the use of pronouns and indefinite pronouns.
- **R25 - /COMPLETENESS/USEOFHEADINGS** - Avoid relying on headings to support explanation or understanding of the requirement.

C3 - Unambiguous
C4 - Complete
C7 - Verifiable

R24 - /COMPLETENESS/AVOIDPRONOUNS

- Repeat nouns in full instead of using pronouns to refer to nouns in other need or requirement statements. Pronouns are words such as “it”, “this”, “that”, “he”, “she”, “they”, and “them.”
- Indefinite pronouns are words such as “all”, “another”, “any”, “anybody”, “anything”, “both”, “each”, “either”, “every”, “everybody”, “everyone”, “everything”, “few“none”, “one”, “several”, “some”, “somebody”, “someone”, “something”, and ”, “many”, “most”, “much”, “neither”, “no one”, “nobody”, “such.” **Indefinite pronouns stand in for unnamed people or things**, which makes their meaning subject to interpretation, ambiguous, and unverifiable.

- When originally written, the noun that defines the pronoun may have preceded the pronoun in the previous need or requirement statement, however, as the set of requirements mature, individual requirements may be deleted, reordered, or regrouped, and the defining requirement is no longer nearby.

- *Unacceptable*: The controller shall send the driver his itinerary for the day. It shall be delivered at least 8 hours prior to his Shift. *{This is unacceptable because the requirement is expressed as two sentences (R19), and the second sentence uses the pronouns "it" and "his."}*

- *Acceptable*: The Controller shall send the Driver_Itinerary for the day to the Driver less than 8 hours prior to the Driver_Shift. *{Note use of the glossary to define terms and to be explicit about the relationship between the driver, shift, and the itinerary for that particular driver.}*

R25 - /COMPLETENESS/USEOFHEADINGS

- It is a common mistake in document-centric documentation of needs and requirements to **use the heading for a specific topic or subject under which the requirement applies to contribute to the explanation of the need or requirement statement**. The need or requirement statement should be complete in and of itself and not require the heading for the intent of the need or requirement to be clearly understood.
- *Example heading:* Alert Buzzer Requirements
- *Unacceptable:* The system shall sound it for greater than 20 minutes. *{This is unacceptable because the requirement uses the pronoun "it" (R24), which requires the heading to understand what "it" means.}*
- *Acceptable:* The system shall sound the Alert_Buzzer for greater than 20 minutes.

REALISM

- **R26 - /REALISM/AVOIDABSOLUTES -** Avoid using unachievable absolutes.

R26 - /REALISM/AVOIDABSOLUTES

C6 – Feasible (Individual requirement)
C7 - Verifiable
C12 – Feasible (Set of requirements)

- An absolute, such as **“100% availability”, is not achievable**. Think ahead to verification and validation: how would you prove 100% availability? Even if you could build such a system, could you afford to verify or validate it?
- Other examples to avoid are “all”, “always”, and “never” since such absolutes are **impossible to verify** without an infinite number of verification or validation activities.
- *Unacceptable*: The system shall have 100% availability. *{This is unacceptable because 100% is an absolute that is impossible to achieve and verify. Also, available over what time period? }*
- *Acceptable*: The system shall have an Availability of greater than or equal to 98% during operating hours.

CONDITIONS

- **R27 - /CONDITIONS/EXPLICIT -** State applicability conditions explicitly.

C4 - Complete
C7 - Verifiable

R27 - /CONDITIONS/EXPLICIT

- State applicability **conditions explicitly instead of leaving applicability to be inferred from the context**. Sometimes need or requirement statements are only applicable under certain conditions. If so, the condition should be repeated in the text of each need or requirement statement, rather than stating the condition and then listing the actions to be taken.
- *Unacceptable*: In the event of a fire:
  - Power to electromagnetic magnetic fire door catches shall be turned off.
  - Security entrances shall be set to Free_Access_Mode.
  - Fire escape doors shall be unlocked.
  - *{This is unacceptable because the condition "in the event of a fire" is stated following a list of actions to be taken. Also, note the actions are written in passive voice which violates R2.}*
- *Acceptable*:
  - In the event of a fire, the Security_System shall turn off power to electromagnetic magnetic fire door catches.
  - In the event of a fire, the Security_System shall set security entrances to the Free_Access_Mode.
  - In the event of a fire, the Security_System shall unlock fire escape doors.

UNIQUENESS

- **R29 - /UNIQUENESS/CLASSIFY -** Classify the need or requirement according to the aspects of the problem or system it addresses.
- **R30 - /UNIQUENESS/EXPRESSONCE -** Express each need and requirement once and only once.

R29 - /UNIQUENESS/CLASSIFY

- By classifying needs or requirements by type or category, it is possible to **regroup or sort requirements to help identify potential duplications and conflicts within a given classification**. The ability to view specific groups of needs or requirements can also assist in identifying what needs or requirements may be missing helping to address the characteristic of completeness as associated with sets of needs or requirements.
- Examples of organizing needs include form, fit, function, quality, and compliance.
- How an organization organizes needs and requirements should be clearly defined in the organization's process documents and associated templates for organizing sets of needs and requirements.

C1 - Necessary
C9 - Conforming
C11 - Consistent
C12 - Feasible

R30 - /UNIQUENESS/EXPRESSONCE

- Avoid including the same or equivalent need and requirement more than once, either as a duplicate or in similar form. Exact duplicates are relatively straightforward to identify; finding similar need or requirement statements with slightly different wording is much more difficult but is aided by the consistent use of defined terms and by classification.
- For example: **"The system shall generate a report of financial transactions containing the information defined in <some standard or contract deliverable listing>"** and **"The system shall generate a financial report"** are overlapping in that the first is a subset of the second.

ABSTRACTION

- **R31 - /ABSTRACTION/SOLUTIONFREE -** Design inputs avoid stating a solution unless there is rationale for constraining the design.

R31 - /ABSTRACTION/SOLUTIONFREE

- Design inputs focus on the problem “what” rather than the solution “how”.
- **As design inputs, every system endeavor should have a level of needs and requirements that captures the problem to be solved (design inputs) without including solutions (design outputs).**
- System level needs and requirements should provide a system-level requirement for the overall problem to be addressed by design. The first level of architecture may be laid out, but subsystems are considered as black-boxes to be elaborated at the next level down.
- When reviewing a requirement that states a solution, again ask “for what purpose? The answer will reveal the real requirement. (Notice that this question is subtly different from simply asking "Why?” and encourages a teleological rather than causal response.) Understanding the concepts of design inputs versus design outputs allows you to ask the question: “Is this requirement addressing what the system needs to do versus how the system needs to do it”?

- *Unacceptable:* Traffic lights shall be used to control traffic at the junction. *{This is unacceptable because "Traffic lights" are a solution (design output). This requirement is also written in passive voice – see R2}*
- *Acceptable (several requirements):*
  - The Traffic_Control_System shall signal "Walk" when the Pedestrian is permitted to cross the road at the junction. *{This is the motivating purpose.}*
  - The Traffic_Control_System shall limit the Wait_Time of Vehicles traversing the Junction to less than 2 minutes during normal daytime traffic conditions.
- *Unacceptable*: By pressing a button on the traffic-light pillar, the Pedestrian signals his presence, and the light turns red for the traffic to stop. *{This is unacceptable because this requirement contains solution-biased detail – design output. In addition, the requirement is unacceptable because the subject is the user over which we have no control—that is, we cannot tell the user what to do in a requirement statement.}*
- *Acceptable*: The Traffic_Control_System shall allow the Pedestrian to signal intent to cross the road. *{This requirement allows freedom in determining the best solution (design input), which may be a means of automatic detection rather than button pushing.}*

QUANTIFIERS

- **R32 - /QUANTIFIERS/UNIVERSALS -** Use “each” instead of “all”, “any”, or “both” when universal quantification is intended.

R32 - /QUANTIFIERS/UNIVERSALS

- The use of “all”, “both”, or “any” is confusing because it is hard to distinguish whether the action happens to the whole set or to each element of the set. “All” can also be hard to verify unless “all” can be clearly defined as a closed set. In many cases, the word “all” is unnecessary and can be removed, resulting in a less ambiguous need or requirement statement.
- *Unacceptable*: The Operation_Logger shall record any *(or all)* warning messages. *{This is unacceptable because of the use of the word “any”, which is then made worse by the addition of “(or all)”.}*
- *Acceptable*: The Operation_Logger shall record each Warning_Message. *{Note that Warning_Message must be defined so that it is clear that the system only will record each defined Warning Message.}*
- *Unacceptable*: The Record_Subsystem shall display the Names of both of the Line_Items. *{This is unacceptable because of the use of the word “both”.}*
- *Acceptable*: The Record_Subsystem shall display the Name of each Line_Item.
- *Acceptable*: The Record_Subsystem shall display each Line_Item_Name.

TOLERANCE

- **R33 - /TOLERANCE/VALUERANGE -** Define quantities with a range of values appropriate to the level stated.

R33 - /TOLERANCE/VALUERANGE

C3 - Unambiguous
C4 - Complete
C6 - Feasible
C7 - Verifiable
C8 - Correct
C12 - Feasible

- When it comes to defining performance, single-point values are seldom sufficient and are difficult to test. Ask yourself the question: **“if the performance was a little less (or more) than this, would I still buy it?”** If the answer is yes, then change the need or requirement statement to reflect this.
- Care should be taken to avoid unspecified value ranges and ensure the quantities are expressed with tolerances or limits. *There are two reasons for this:*
  - *1) Several requirements may have to be traded against each other, and providing tolerances or limits is a way of describing the trade-space. Seldom is a quantity absolute. A range of values is usually acceptable, providing different performance levels.*
  - *2) Verification against a single absolute value is usually not possible or at best very expensive and time consuming, whereas verification against a defined range of values with upper and lower limits makes verification more manageable.*
- Care should also be taken to ensure the tolerances are no tighter than needed. Tighter tolerances can drive costs both in system design and manufacturing as well as the costs in verifying the system can perform within the tighter tolerances.

- *Unacceptable:* The Pumping_Station shall maintain the flow of water at 120 liters per second for 30 minutes. *{This is unacceptable because we don't know whether a solution that carries more or less than the specified quantities is acceptable.}*
- *Acceptable:* The Pumping_Station shall maintain a flow of water at 120 ±10 liters per second for greater than 30 minutes. *{Now we know the range of acceptable flow performance, and that the 30 minutes is a minimum acceptable performance.}*

- *Unacceptable:* The System shall limit arsenic contamination in the drinking water to 1 part per trillion.
  - Rationale: Arsenic contamination in drinking water can cause health problems.
  - *{This is unacceptable because the EPA contamination limit in drinking water is 10 parts per billion. Requiring a tighter limit may be beyond the ability of current technology to measure or if measuring concentrations of 1 part per trillion are possible, the cost to do so may be unacceptably high. Also, no range is specified. Using less than is probably the real intent.}*
- *Acceptable*: The System shall limit arsenic contamination in the drinking water to less than 10 parts per billion.
  - *Rationale*: EPA set the arsenic standard for drinking water at 10 ppb (or 0.010 parts per million). The EPA has determined that concentrations of this level or less will protect consumers from the effects of long-term, chronic exposure to arsenic.

QUANTIFICATION

- **R34 - /QUANTIFICATION/MEASURABLE -** Provide specific measurable performance targets appropriate to the level at which the need or requirement is stated.
- **R35 - /QUANTIFICATION/TEMPORALINDEFINITE -** Define temporal dependencies explicitly instead of using indefinite temporal keywords.

R34 - /QUANTIFICATION/MEASURABLE

C3 - Unambiguous
C4 - Complete
C7 - Verifiable
C12 - Feasible

- Some words signal unmeasured quantification, such as ”prompt”, ”fast”, ”routine”, ”maximum”, ”minimum”, ”optimum”, ”nominal”, ”easy to use”, ”close quickly”, ”high speed”, ”medium-sized”, ”best practices”, and ”user-friendly.” These are ambiguous and need to be replaced by specific quantities that can be measured.
- *Unacceptable:* The system shall use minimum power. *{This is unacceptable because “minimum” is ambiguous.}*
- *Acceptable:* The system shall use less than or equal to 50W of main power. *{This both takes into account the underlying goal - to minimize power consumption - and provides a measurable target.}*
- *Unacceptable:* The engine shall achieve an emissions level that is at least 5% less than the competition’s emission levels 2 years from now. *{This is an actual requirement from marketing to an engineering department. The statement sets a completely unmeasurable end state.}*
- *Acceptable:* The Engine shall achieve an emissions level that is at least xxx. *{where xxx represents the required threshold value.}*

R35 - /QUANTIFICATION/ TEMPORALINDEFINITE

C3 - Unambiguous
C4 - Complete
C7 - Verifiable

- Some words and phrases signal non-specific timing, such as “eventually”, “until”, “before”, “when”, “after”, “as”, “once”, “earliest”, “latest”, “instantaneous”, “simultaneous”, “while”, and “at last”, are ambiguous and not verifiable. Indefinite temporal keywords can cause confusion or unintended meaning. These words should be replaced by specific timing constraints.
- *Unacceptable*: Continual operation of the pump shall eventually result in the tank being empty. *{This is unacceptable because “eventually” is ambiguous. Also, this statement is not written in the proper form. It is written on “operation” rather than on a requirement on the pump which violates R3.}*
- *Acceptable*: The Pump shall remove greater than 99% of the Fluid from the Tank in less than 3 days of continuous operation.

UNIFORMITY OF LANGUAGE

- **R36 - /UNIFORMLANGUAGE/USECONSISTENTTERMS -** Use each term and units of measure consistently throughout need and requirement sets.
- **R37 - /UNIFORMLANGUAGE/DEFINEACRONYMS -** If acronyms are used in need and requirement statements, use a consistent set of acronyms.
- **R38 - /UNIFORMLANGUAGE/AVOIDABBREVIATIONS -** Avoid the use of abbreviations in needs and requirement statements.
- **R39 - /UNIFORMLANGUAGE/STYLEGUIDE -** Use a project-wide style guide for individual need and requirement statements.

R36 - /UNIFORMLANGUAGE/ USECONSISTENTTERMS

C3 - Unambiguous
C8 - Correct
C9 - Conforming
C11 - Consistent
C13 - Comprehensible
C14 - Able to be Validated

- A common ontology therefore needs to be defined for each project defining terms and units of measure across all artifacts, including the sets of needs and requirements as well as all design output artifacts. Synonyms are not acceptable. A glossary is very useful to define words precisely. Glossary terms are capitalized to signify that the word or term being used has a specific meaning in the context of the set of statements.
- Ideally, the glossary used for the sets of needs and requirements is a subset of the project glossary.
- *Unacceptable*: When the Input_Valve is connected to the Water_Source, the Control_Subsystem shall open the Inlet_Valve. *{Two terms are used for the same thing "Input_Valve" and "Inlet_Valve".}*
- *Acceptable:* When the Inlet_Valve is connected to the Water_Source, the Control_Subsystem shall open the Inlet_Valve.
- *Unacceptable: It would not be acceptable for one requirement to use one unit of measure (e.g. US - feet) and another to use another unit of measure (e.g. Metric - meter).*

R37 - /UNIFORMLANGUAGE/ DEFINEACRONYMS

C3 - Unambiguous
C9 - Conforming
C11 - Consistent
C13 - Comprehensible
C14 - Able to be Validated

- In paper-based specifications, a common rule is to use the full term and the abbreviation or acronym (in brackets) the first time and then use just the abbreviation or acronym from then on. In requirement sets that are generated from databases, there is no guarantee that the set will be extracted in any particular order, so the old practice is not useful. Acronyms must be used consistently throughout not only the sets of needs and requirements, but all artifacts developed across all life cycle stages. To address this issue, make sure acronyms are defined in an acronym list or glossary.
- *Unacceptable: It would not be acceptable for one requirement to refer to the "Global Positioning System" and the remaining requirements refer just to "GPS."*
- *Acceptable: Use the full term every time or, perhaps more usefully, define the acronym in the acronym list or glossary and use it every time.*

R38 - /UNIFORMLANGUAGE/ AVOIDABBREVIATIONS

C9 - Conforming
C11 - Consistent
C13 - Comprehensible
C14 - Able to be Validated

- The use of abbreviations adds ambiguity and is to be avoided.
- *Unacceptable: It would not be acceptable for one requirement to refer to the abbreviation "op" meaning operation and another to refer to the "op" meaning the operator.*
- *Acceptable: Avoid the abbreviation and use the full term every time.*

R39 - /UNIFORMLANGUAGE/ STYLEGUIDE

C5 - Singular
C9 - Conforming
C11 - Consistent
C13 - Comprehensible
C14 - Able to be Validated

- The style guide provides a template and patterns for developing requirements, defines the attributes that the organization chooses to document with each requirement statement, selects the requirement patterns to be used for specific types of requirements, and defines what other information needs to be included (such as the glossary). **The patterns can come from an international standard or an organizational standard set of patterns based on type of stakeholder need or requirement.**
- **Each organization should have a style guide or schema to address such issues as the selection and definition of what need and requirement patterns** should be used for writing needs and requirements of a particular type, the selection and use of attributes, standard abbreviations and acronyms, layout and use of figures and tables, layout of needs and requirements documents or databases and, need and requirement statement numbering.

MODULARITY

- **R40 - /MODULARITY/RELATEDREQUIREMENTS**
- Group related requirements together.
- **R41 - /MODULARITY/STRUCTURED -** Conform to a defined structure or template for sets of needs and requirements.

R40 - /MODULARITY/ RELATEDREQUIREMENTS

C9 - Conforming
C11 - Consistent
C13 - Comprehensible

- Need and requirements **statements that belong together should be grouped together**. This is a good principle for structuring a requirements document or organizing requirements with a set documented within an RMT or another SE tool**. One approach is to put requirements into groups based on their classification.**
- In functional decomposition, functional requirements are grouped by the function that originates them. Performance requirements are grouped with the functions they apply to.
- *Requirements may be related by:*
  - *type (for example, safety requirements);*
  - *scenario (for example, requirements arising from a single scenario).*

R41 - /MODULARITY/ STRUCTURED

C10 - Complete
C11 - Consistent
C13 - Comprehensible
C14 - Able to be Validated

- A well-structured set of needs and requirements allows **an understanding of the whole set to be assimilated without undue cognitive loading on the reader**.
- Despite what has been stated about the completeness of individual need and requirement statements, it is often easier to understand when it is placed in context with other related requirements.
- *For sets of needs or requirements, an outline can be defined that organizes them in categories or types such as: functional/performance, operational, quality (-ilities), design and construction standards, and physical characteristics. The outlines can come from international standards or an organizational standard tailored to the organization's domain and different types of products within that domain. (The organization for system level needs and requirements may be different than the organization of a set of software needs and requirements.)*