



34th Annual **INCOSY**
international symposium

hybrid event

Dublin, Ireland
July 2 - 6, 2024



Systematically Pulverised EARS

Improvements in requirements authoring and presentation

John Welford – 02/07/24

2 – 6 July

www.incose.org/symp2024 #INCOSY24

Contents

1. Adding syntax highlighting to EARS

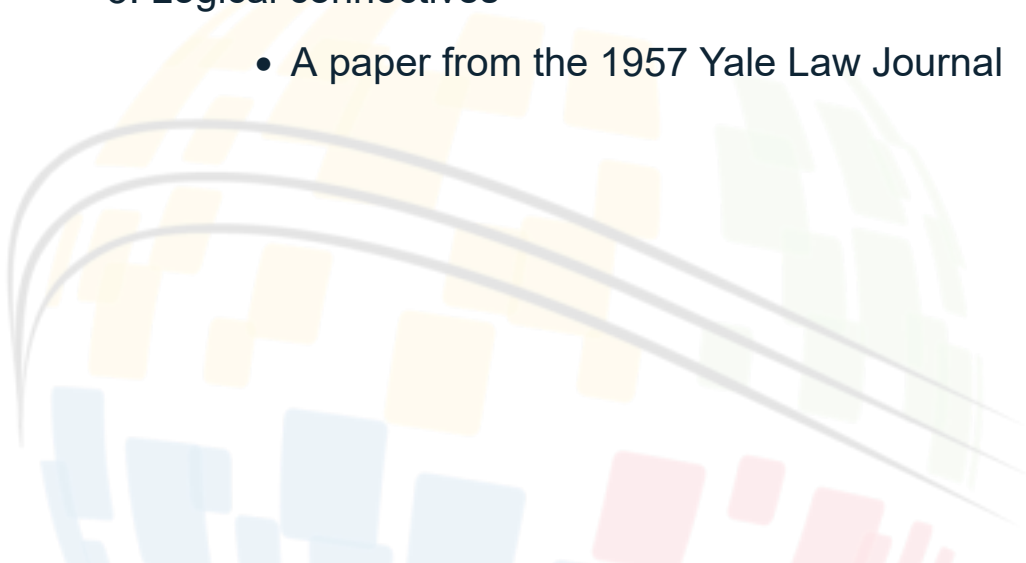
- The rules of Roadrunner and Wile E. Coyote

2. Vacuously true requirements about an empty set

- A viral puzzle from the 17th Brazilian Maths Olympiad

3. Logical connectives

- A paper from the 1957 Yale Law Journal



Requirements writing

Lots of excellent references already exist.

I agree with (almost) all of it.

What I'm suggesting here are *possible* extensions to it, mostly to presentation and syntax.

SYSTEMS ENGINEERING
A GUIDE FOR

INTERNATIONAL STANDARD ISO/IEC/IEEE 29148
Second edition 2018-11

Systems and software engineering — Life cycle processes — Requirements engineering
Ingénierie des systèmes et du logiciel — Processus du cycle de vie — Ingénierie des exigences

Enterprise Concept of Op (ConOps)
Business Management Preliminary Life Cycle Concepts (Pre-Concepts) Acquisition, Development, Support, Retirement
Business Operations Life-cycle Concepts (Acquisition, Development, Support, Retirement)
System System Life Cycle Concepts (Acquisition, Development, Support, Retirement)

Enterprise Concept of Op (ConOps)
Business Management Preliminary Life Cycle Concepts (Pre-Concepts) Acquisition, Development, Support, Retirement
Business Operations Life-cycle Concepts (Acquisition, Development, Support, Retirement)
System System Life Cycle Concepts (Acquisition, Development, Support, Retirement)

Enterprise Concept of Op (ConOps)
Business Management Preliminary Life Cycle Concepts (Pre-Concepts) Acquisition, Development, Support, Retirement
Business Operations Life-cycle Concepts (Acquisition, Development, Support, Retirement)

Guidance
Needs, Requirements, Requirements Engineering
Guide

Requirements syntax

Natural language requirements benefit from being written to follow a defined structure.

Various syntax forms have been proposed.

The Easy Approach to Requirements Syntax (EARS) is my preferred popular option.

C.3 Basic structure of a need or requirement pattern

Requirer

As stated
of a comp
<

The *subje*
therefore
before the
because
required

Both, sub
and, for a
Use of “s
requirem
requirem

[Condition] [Subject] [Action] [Object] [Constraint of Action]

EXAMPLE: When signal x is received [Condition], the system [Subject] shall set [Action] the signal x received bit [Object] within 2 seconds [Constraint of Action].

Or

[Condition] [Subject] [Action] [Object] [Constraint of Action]

EXAMPLE: At sea state 1 [Condition], the Radar System [Subject] shall detect [Action] targets [Object] at ranges out to 100 nautical miles [Constraint of Action].

Or

[Subject] [Action] [Constraint of Action]

EXAMPLE: The Invoice System [Subject] shall display pending customer invoices [Action] in ascending order of invoice due date [Constraint of Action].

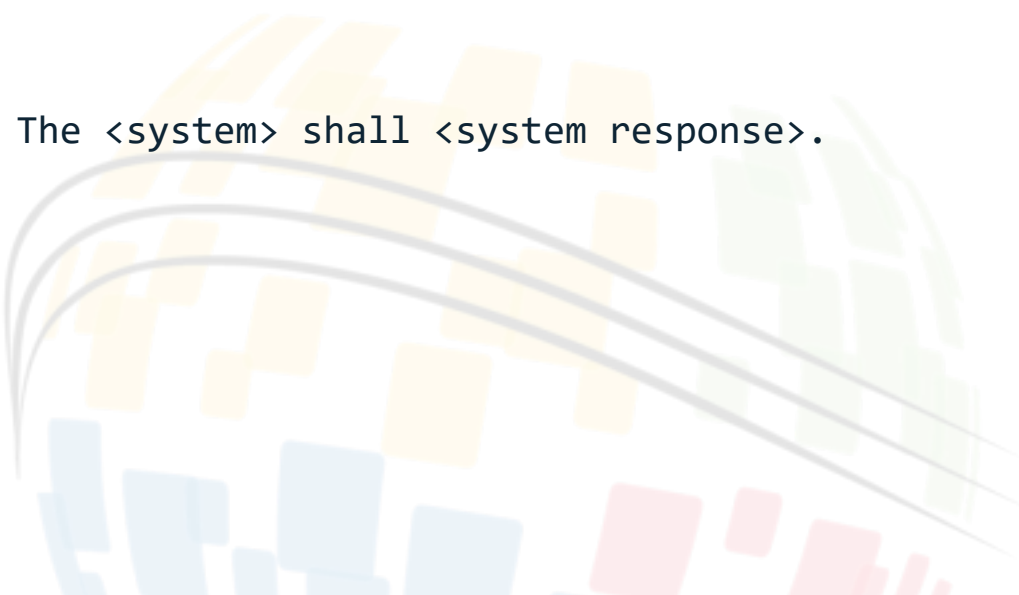
Expanding on the *<subject> <verb> <object><response>* form, the *response* may be further elaborated to address a measurable outcome:

The <entity> shall <action verb> <object> <measurable outcome>.

The Easy Approach to Requirements Syntax

(Ubiquitous requirement)

The <system> shall <system response>.

A decorative graphic in the bottom-left corner of the slide. It features a stylized globe with segments in yellow, green, blue, and red. Several grey curved lines sweep across the globe from the bottom-left towards the center.

The Easy Approach to Requirements Syntax

(Event-driven requirement)

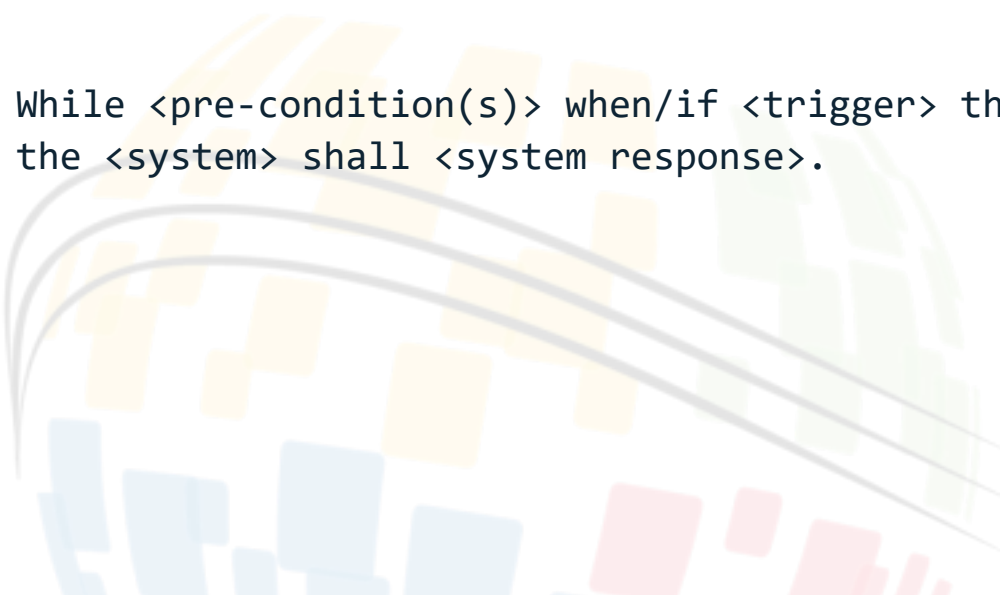
When/if <trigger> then the <system> shall <system response>.

A decorative graphic in the bottom-left corner of the slide. It features a stylized globe composed of various colored rectangular segments in shades of yellow, green, blue, and red. Overlaid on the globe are several thin, grey, curved lines that sweep across the scene from the bottom-left towards the center.

The Easy Approach to Requirements Syntax

(State- & Event-driven requirement)

While <pre-condition(s)> when/if <trigger> then
the <system> shall <system response>.



The Easy Approach to Requirements Syntax

(Optional feature & State- & Event-driven requirement)

Where <option> while <pre-condition(s)> when/if <trigger> then the <system> shall <system response>.

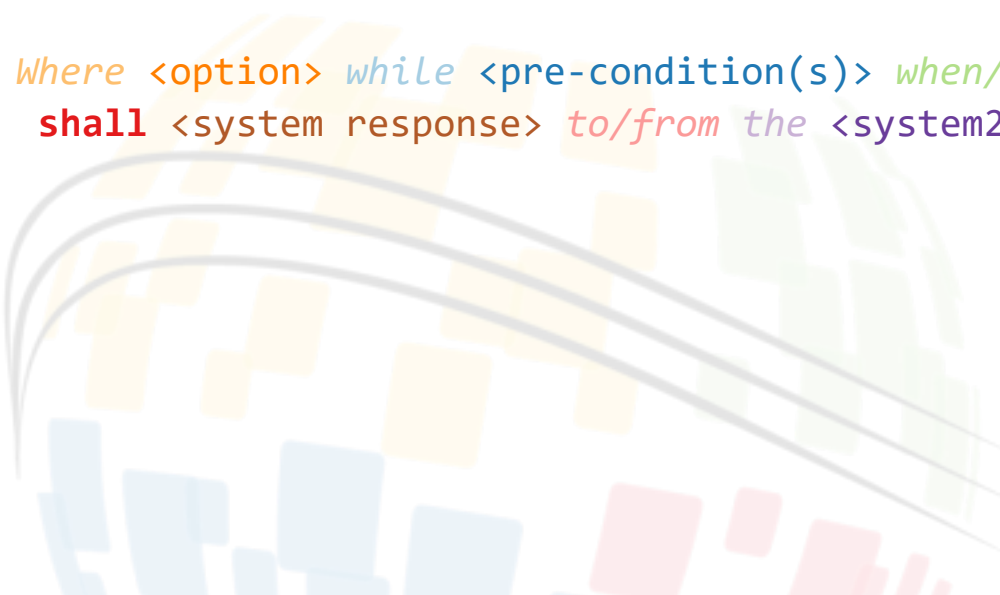


Syntax highlighting

```
PrismEars = ({
  'requirement': {
    pattern: /<sup id="fnref:\.n" role="doc-noteref">1</sup>* (shall|should|may)<sup id="fnref:." role="doc-noteref">2</sup>*\.\/i, // ignore st
    inside: {
      'option': {
        pattern: /(\bwhere\b) (.*) (?=(\bshall\b|\bshould\b|\bmay\b|\bthen\b|\bif\b|\bwhen\b|\bwhile\b))/i,
        lookbehind: true,
        inside: { // for the case where systems cannot be separated from options
          'option': /(\bthe\b).*?(=\bthe\b)/i, // multiple 'the's are still an option
          'system': {
            pattern: /(\bthe\b) (.*)/i,
            lookbehind: true,
          },
          'joining': /\b(the)\b/i,
        }
      }
    }
  }
})
```

Extending EARS for interfaces

Where <option> *while* <pre-condition(s)> *when/if* <trigger> *then* *the* <system>
shall <system response> *to/from* *the* <system2>.



The rules of Roadrunner and Wile E. Coyote

1. The Road Runner cannot harm the Coyote except by going “meep, meep.”
2. No outside force can harm the Coyote – only his own ineptitude or the failure of Acme products. Trains and trucks were the exception from time to time.
3. The Coyote could stop anytime – if he were not a fanatic.
4. No dialogue ever, except “meep, meep” and yowling in pain.
5. The Road Runner must stay on the road – for no other reason than that he’s a roadrunner.



The rules of Roadrunner and Wile E. Coyote

6. All action must be confined to the natural environment of the two characters – the southwest American desert.
7. All tools, weapons, or mechanical conveniences must be obtained from the Acme Corporation.
8. Whenever possible, make gravity the Coyote's greatest enemy.
9. The Coyote is always more humiliated than harmed by his failures.
10. The audience's sympathy must remain with the Coyote.
11. The Coyote is not allowed to catch or eat the Road Runner.



1. The Road Runner cannot harm the Coyote except by going “meep, meep.”

The Road Runner shall not harm the Coyote, except by going “meep, meep”.

2. No outside force can harm the Coyote – only his own ineptitude or the failure of Acme products. Trains and trucks were the exception from time to time.

The Coyote should only be harmed by his own ineptitude or the failure of Acme products, not by any outside force.

The Coyote may be harmed by trains and trucks.

3. The Coyote could stop anytime – if he were not a fanatic.

The Coyote shall be capable of stopping anytime.

4. No dialogue ever, except “meep, meep” and yowling in pain.

The Cartoon shall have no dialogue, except “meep, meep” and yowling in pain.

5. The Road Runner must stay on the road – for no other reason than that he’s a roadrunner.

The Road Runner shall stay on the road.

Rationale: He’s a roadrunner.

6. All action must be confined to the natural environment of the two characters – the southwest American desert.

The Cartoon shall be confined to the southwest American desert.

7. All tools, weapons, or mechanical conveniences must be obtained from the Acme Corporation.

Where Tools, Weapons, and Mechanical Conveniences are used, the items shall be obtained from the Acme Corporation.

8. Whenever possible, make gravity the Coyote's greatest enemy.

The Coyote should have gravity as his greatest enemy.

9. The Coyote is always more humiliated than harmed by his failures.

When he fails the Coyote shall be more humiliated than harmed.

10. The audience's sympathy must remain with the Coyote.

The Audience shall have sympathy with the Coyote.

11. The Coyote is not allowed to catch or eat the Road Runner.

The Coyote shall not catch or eat the Road Runner.

A viral puzzle

From the 17th Brazilian Maths Olympiad:

- Pinocchio always lies
- Pinocchio says “*All my hats are green*”

We can conclude which of the following?

- A) Pinocchio has at least one hat.
- B) Pinocchio has only one green hat.
- C) Pinocchio has no hats.
- D) Pinocchio has at least one green hat.
- E) Pinocchio has no green hats.



A viral puzzle

From the 17th Brazilian Maths Olympiad:

- Pinocchio always lies
- Pinocchio says “*All my hats are green*”

We can conclude which of the following?

- A) Pinocchio has at least one hat. ✓
- B) ~~Pinocchio has only one green hat.~~ (complies with the hat statement, but he isn't lying)
- C) Pinocchio has no hats. ???
- D) ~~Pinocchio has at least one green hat.~~ (he could have a blue hat and comply)
- E) ~~Pinocchio has no green hats.~~ (he could have a mix of hat colours and comply)



A viral puzzle

From the 17th Brazilian Maths Olympiad:

- Pinocchio always lies
- Pinocchio says “*All my hats are green*”

We can conclude which of the following?

A) Pinocchio has at least one hat. ✓

C) Pinocchio has no hats. ???

Why isn't option C also correct?

In formal logic Pinocchio's hat statement is *vacuously true* if he has no hats; therefore Pinocchio is not lying, making option C incorrect.

As option C is a statement about an *empty set* (something that doesn't exist) it's not immediately clear whether we can make true or false claims about its attributes.



What does this have to do with requirements?

Suppose Pinocchio is requesting that WSP populate his hat-rack for him and gives us the single requirement:

Pinocchio's hats **shall** be green.

We take this to our multi-million-dollar WSP hat design team, who come back with the following design options for us to evaluate:

A) A hat-rack with two blue hats and one green hat. ✘

B) A hat-rack with two green hats. ✔

C) A hat-rack with no hats. ???

Is option 3 an extremely cost effective solution to Pinocchio's requirement?

Or is it non-compliant?

The solution includes the *empty set* for an item mentioned in the requirement.

Confusion could have been avoided by the additional requirement:

Pinocchio's hat rack **shall** have at least one hat.

A rubbish requirement

Rubbish bins **shall** be made of non-combustible materials.

Design options

A)	A station with 5 metal (non-combustible) bins	✓
B)	A station with 5 plastic (combustible) bins	×
C)	A station without bins	???

A pair of better requirements

1. *The station shall have at least 3 rubbish bins.*
2. *The rubbish bins shall be made of non-combustible materials.*

Design options	1	2
A) A station with 5 metal (non-combustible) bins	✓	✓
B) A station with 5 plastic (combustible) bins	✓	×
C) A station without bins	×	???

A good requirement

Where rubbish bins are included in the station, the rubbish bins shall be made of non-combustible materials.

Design options

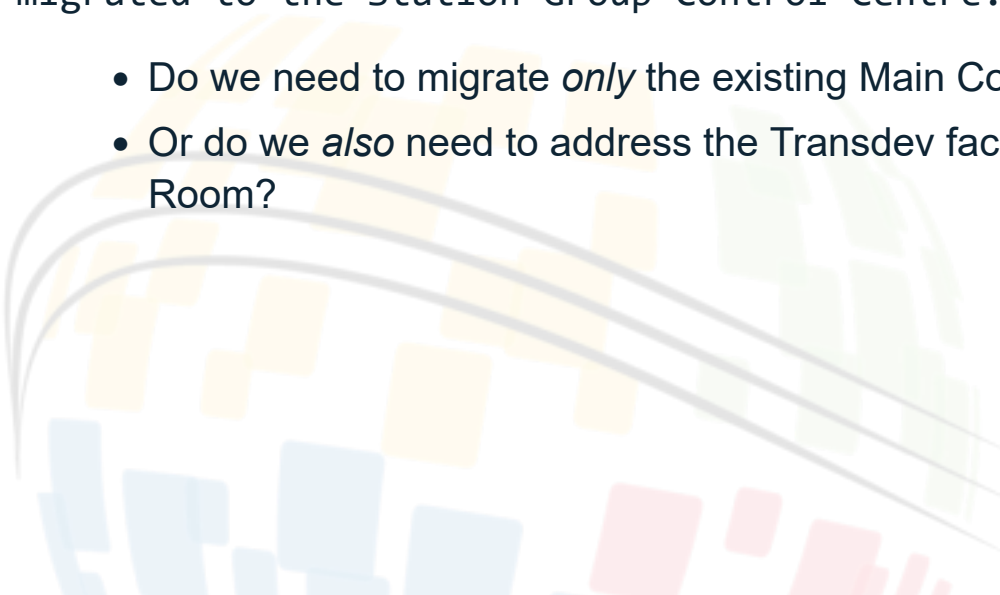
A)	A station with 5 metal (non-combustible) bins	✓
B)	A station with 5 plastic (combustible) bins	✗
C)	A station without bins	✓ (N/A)

Logical Connectives

These can make requirements difficult to parse.

For example:

Existing Main Control Room functions at Britomart Station comprising KiwiRail emergency signalling control facilities and the contracted Railway Operator (currently Transdev) facilities shall be migrated to the Station Group Control Centre.

- Do we need to migrate *only* the existing Main Control Room functions?
 - Or do we *also* need to address the Transdev facilities outside the Main Control Room?
- 

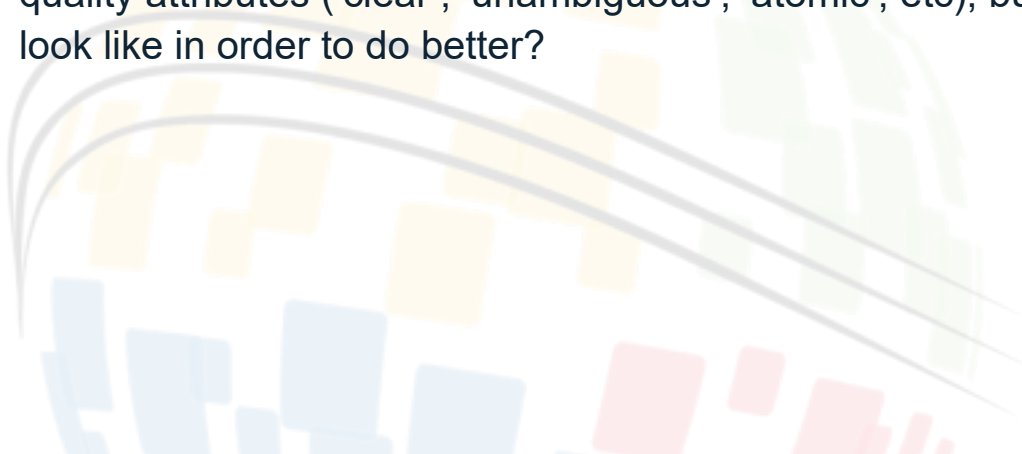
Logical Connectives

Another example:

All conductors except radio antennas shall be enclosed in their entirety in metal or non-combustible conduits or enclosed raceways if not enclosed or separated by fire-resistant construction.

- If conductors are separated by fire-resistant construction, do they need to *also* be enclosed in metal?

These are not well written requirements; they fall down on several standard requirements quality attributes ('clear', 'unambiguous', 'atomic', etc), but what should our requirement syntax look like in order to do better?



Legal Documentation

Legal documentation is an area that has been playing with the idea of formal logic in natural language statements for a long time. There can also be some overlap in application between requirements and legal documents.

There are a large number of current projects that are looking at how to state legal contracts using a more mathematically logical approach.

I'm going to reference one of the oldest papers in this area, published in the 1957 Yale Law Journal: '[Symbolic Logic: A Razor-Edged Tool for Drafting and Interpreting Legal Documents](#)' by **Layman E. Allen**. This suggests an improved technique borrowing ideas from symbolic logic to restate things in a *systematically pulverized form*.

SYMBOLIC LOGIC: A RAZOR-EDGED TOOL FOR DRAFTING AND INTERPRETING LEGAL DOCUMENTS

LAYMAN E. ALLEN†

A LARGE amount of the litigation based on written instruments—whether statute, contract, will, conveyance or regulation—can be traced to the draftsman's failure to convey his meaning clearly. Frequently, of course, certain items may purposely be left ambiguous, but often the question in issue is due to an inadvertent ambiguity that could have been avoided had the draftsman clearly expressed what he intended to say. In this Article it is suggested that a new approach to drafting, using certain elementary notions of symbolic logic, can go a long way towards eliminating such inadvertent ambiguity. This new approach makes available to draftsmen a technique that achieves some of the clarity, precision and efficiency of analysis that symbolic logic provides. In addition, it can be a valuable aid in moving towards a more comprehensive and systematic method of interpretation,¹ as well as drafting.

This approach is a compromise between expression in ordinary prose and

CONNECTIVE	SYMBOL	EXAMPLES	
		ORDINARY VERBAL FORM	SYSTEMATICALLY-PULVERIZED FORM
Conjunction	1. &2.	The six logical connectives dealt with here are conjunction, exclusive disjunction, inclusive disjunction, negation, implication and coimplication.	The six logical connectives dealt with here are 1. conjunction &2. exclusive disjunction &3. inclusive disjunction &4. negation &5. implication &6. coimplication.
Exclusive disjunction	1- OR 2-	A person either understands them or he does not.	A person either 1- does OR 2- does NOT understand them
Inclusive disjunction	1) 2) &OR	Exclusive disjunction and/or inclusive disjunction may prove tricky for a while, but one soon learns to distinguish them.	1) Exclusive disjunction 2) &OR inclusive disjunction may prove tricky for a while, but one soon learns to distinguish them.
Negation	NOT	The explanation here should not be hard to understand.	The explanation here should NOT be hard to understand.
Implication	1. _____	If a person can read, then he should be able to understand it very easily.	1. A person can read _____
	2.		2. HE SHOULD BE ABLE TO UNDERSTAND IT VERY EASILY.
Coimplication	1. =====	If, and only if, a person can read, he should be able to understand it very easily.	1. A person can read =====
	2.		2. HE SHOULD BE ABLE TO UNDERSTAND IT VERY EASILY
1. Antecedent _____			
2. CONSEQUENT			

Negations

It is considered bad practice to use negation in the `<system response>` part of a requirement (i.e. to state what a system ought not do).

Sometimes this is difficult or it's necessary as part of the `<option(s)>`, `<pre-condition(s)>`, or `<trigger(s)>`.

In these cases it needs to be clear what part of the requirement the negation applies to, especially where it is combined with a conjunction or disjunction.



Conjunction

These are 'AND' type statements.

The presence of an 'AND' statement within the `<system response>` part of a requirement suggests that your requirement is not atomic and could be broken into two separate requirements; for example:

The track shall provide a corridor for passenger trains and maintenance trains to travel on.

All statements in a requirement set should be met at the same time, so there is an implied 'AND' between all of them.

The above would be better written as:

The track shall provide a corridor for passenger trains to travel on.

The track shall provide a corridor for maintenance trains to travel on.

Conjunctions may be necessary in the statement of requirement <option(s)>, <pre-condition(s)>, or <trigger(s)> (the *antecedent* in logical connective sense – see later).

In complex conditions it can help to structure the statement using brackets.

The existing Main Control Room functions at Britomart Station (comprising [KiwiRail emergency signalling control facilities AND the contracted Railway Operator (currently Transdev) facilities]) shall be migrated to the Station Group Control Centre.

Or as an alternative interpretation as two requirements:

The existing Main Control Room functions at Britomart Station (comprising KiwiRail emergency signalling control facilities) shall be migrated to the Station Group Control Centre.

The contracted Railway Operator facilities (currently Transdev) shall be migrated to the Station Group Control Centre.

These resolve our previous issue of whether Transdev facilities outside of the Main Control Room need to be transferred – no in the first interpretation and yes in the second.

Disjunction

These are 'OR' type statements.

'OR' statements might be necessary in the `<option(s)>`, `<pre-condition(s)>`, `<trigger(s)>`, or `<system response>` parts of a requirement.

They can get complicated if there are multiple.



Our example from earlier could mean:

Where NOT ((enclosed) OR (separated)) by fire-resistant construction conductors (except radio antennas) **shall** be enclosed in their entirety in ((metal OR non-combustible) conduits) OR (enclosed raceways).

or it could mean the combination of:

Conductors (except radio antennas) **shall** be enclosed in their entirety in ((metal OR non-combustible) conduits) OR (enclosed raceways).

Where NOT ((enclosed) OR (separated)) by fire-resistant construction conductors (except radio antennas) **shall** be enclosed in raceways.

These make it clearer whether conductors that are separated by fire-resistant construction need to be enclosed; in the first case no (as they are not explicitly requested), in the second case yes.

However, there is still some potential ambiguity; in the first case is it allowable for conductors to be enclosed in a conduit, which is then inside a raceway? This highlights the difference between exclusive or inclusive disjunction...

Inclusive Disjunction

In logic gate terms this is simply 'OR', in natural language terms we might say 'AND/OR'.

NOTE: The INCOSE Guide for Writing Requirements suggests that 'and/or' is ambiguous, and suggests "at least one of ...". This can be cumbersome in use in certain circumstances.

If we intended to allow the case where conductors could be enclosed by both a conduit and raceway simultaneously, then our previous first option could be rewritten:

Where NOT (enclosed AND/OR separated) by fire-resistant construction conductors (except radio antennas) shall be enclosed in their entirety in ((metal AND/OR non-combustible) conduits) AND/OR (enclosed raceways).

You might also notice that I've also interpreted the disjunction in the `<option>` here as inclusive. If this is correct, then it can be separated into multiple requirements:

Where NOT enclosed by fire-resistant construction conductors (except radio antennas) shall be enclosed in their entirety in ((metal AND/OR non-combustible) conduits) AND/OR (enclosed raceways).

Where NOT separated by fire-resistant construction conductors (except radio antennas) shall be enclosed in their entirety in ((metal AND/OR non-combustible) conduits) AND/OR (enclosed raceways).



Exclusive Disjunction

In logic gate terms this would be an 'EXCLUSIVE OR' (XOR), i.e. one or other option is allowable, but not both.

NOTE: The INCOSE Guide for Writing Requirements suggests using “either ... OR ... but NOT both” for EXCLUSIVE OR.

Exclusive disjunction is less commonly intended in requirement statements; however, there can be cases where it is specifically what we want, for example where achieving both would be overly costly:

Where running from main switchboards to distribution-boards submain cabling shall be secured with either (steel cable cleats OR stainless cable ties) but NOT both.

Implication

Implication is of the form 'IF ... THEN ...'.

This is the separation between the <option(s)> <pre-condition(s)> <trigger(s)> parts of EARS and the <system> **shall** <system response> part.

These are referred to as the *antecedent* and the *consequent* in an implication.

Layman suggests separating the presentation as two parts:

Where running from main switchboards to distribution-boards...

...submain cabling **shall** be secured with steel cable cleats OR stainless cable ties.

Coimplication

Coimplication to state 'IF, AND ONLY IF, ... THEN ...'. I don't think this is well addressed in the current EARS syntax but is easily added.

Consider the following requirement:

The platform-to-track intruder detection system shall clear the train driver notice only after manual confirmation.

This has a lot contained within it, which can be made more explicit by rewriting using the EARS syntax as:

When manual confirmation is provided *then* the platform-to-track intruder detection system **shall** clear the train driver notice.

If manual confirmation is NOT provided *then* the platform-to-track intruder detection system **shall** NOT clear the train driver notice.

The inclusion of the double negatives are necessary here to denote the 'only' part of the requirement.

Using Layman's *systematically pulverized form* we could potentially drop these requirements and rewrite as:

ONLY *when* manual confirmation is provided *then* ...

... *the* platform-to-track intruder detection system **shall** clear the train driver notice.



Summary

Syntax highlighted EARS:

Where <option> *while* <pre-condition(s)> *when/if* <trigger> *then the* <system>
shall <system response> *to/from the* <system2>.

If the <system> discussed in a requirement is absolutely required in the solution then ensure that there is another explicit requirement requesting it.

If it is allowable that a <system> addressed by a requirement could be not present in the solution (i.e. an *empty set*) then write requirements as:

Where <system> *is included,* *the* <system> **shall** <system response>.

Summary

		<i>antecedent</i>	<i>consequent</i>
Implication	IF ... THEN ...	<i>Where</i> <option> <i>while</i> <pre-condition(s)> <i>when/if</i> <trigger> <i>then</i>	<i>the</i> <system> shall <system response>
Negation	NOT	Make scope clear using brackets	Avoid by reframing requirement
Conjunction	AND	Make scope clear using brackets	Avoid by splitting requirement
Inclusive Disjunction	AND/ OR	Avoid by splitting requirement	Make scope clear using brackets
Exclusive Disjunction	OR	Avoid by splitting requirement	Make scope clear using brackets State 'either ... but NOT both'
Coimplication	IF, AND ONLY IF ...	ONLY <i>Where</i> <option> ONLY <i>while</i> <pre-condition(s)> ONLY <i>when/if</i> <trigger> <i>then</i>	<i>the</i> <system> shall <system response>



34th Annual **INCOSE**
international symposium

hybrid event

Dublin, Ireland
July 2 - 6, 2024

www.incose.org/symp2024

#INCOSEIS

john.welford@wsp.com