QRA

Automating the INCOSE Guide for Writing Requirements

A time-saving guide for system engineers and requirements analysts

TABLE OF CONTENTS

EXECUTIVE SUMMARY

The INCOSE *Guide for Writing Requirements* is one of the most widely used and highly respected references in requirements engineering (RE). Systems engineers, engineering managers and business analysts rely on it to help them make sure the systems their organizations produce or acquire are clearly and accurately specified.

The *Guide for Writing Requirements (GFWR)* provides perhaps the most comprehensive set of rules available for helping RE professionals write unambiguous requirements. Unfortunately, because the INCOSE rule set is so large – forty-one rules in the latest revision – accounting for all of them during authoring and manual review can prove cumbersome and tedious.
Fortunately, checking for compliance with the majority of the GFWR's rules can now be automated using new natural language processing (NLP) tools designed for requirements analysis. This process does not take the requirements engineer out of the loop, by any means. Rather, it assists the engineer by automatically analysing sets of requirements against a host of best-practice factors and by indicating which of those requirements may require further attention.

This guide will describe how compliance with much of the GFWR rule set can be automated with QVscribe, the industry's leading NLP requirements analysis tool from QRA Corp. It will show how QVscribe can streamline requirement authoring and review processes, eliminate much of the tedium from those processes, and allow domain experts more time for tasks that require their expertise.

HOW THIS GUIDE WORKS

This guide describes how the process of complying with the Rules for Requirement Statements and Sets of Requirements described in the INCOSE Guide for Writing Requirements can be automated and streamlined using QVscribe.

For readers not fully familiar with the INCOSE Guide for Writing Requirements or whose organizations have not sought compliance with its rule set, this guide first presents some background. The next two sections describe the problems associated with the writing of engineering requirements in natural language and typical methods used to try to overcome those problems.
Then, the guide introduces the QVscribe analysis tool. Readers who are familiar with the INCOSE rules but unfamiliar with QVscribe may wish to begin with the section titled A Better Way to Write and Review Requirements.

Finally, the guide describes how to automate the process of compliance with the ICOSE rule set. This automation benefits readers and their organization both by speeding up the requirements authoring and review processes and by helping requirements authors write clear and accurate requirements. Readers who are familiar with both the INCOSE rules and QVscribe may want to skip directly to Automating the INCOSE Guide for Writing Requirements with QVscribe.

WRITING CLEAR REQUIREMENTS ISN'T EASY

Most companies understand the need for specifying clear, unambiguous requirements, both for the systems they build and sell, and for the systems they commission and procure from other vendors. Numerous studies have shown that the cost of fixing engineering errors in systems and software increases exponentially over the project life cycle (Jonette[i], Boehm[ii], Rothman[iii], McGibbon[iv], Cigital[v]) and that more than half of all engineering errors originate in the requirements (Martin[vi]).

Writing clear, unambiguous requirements in natural (spoken) language on a consistent basis, however, is not easy.

"Natural language's extensive vocabulary and commonly understood syntax facilitate communication and make it an inviting choice to express requirements," says William Wilson, a former principal systems engineering consultant with the Software Assurance Technology Center (SATC). "The informality of the language also makes it relatively easy to specify high-level general requirements when precise details are not yet known.

"However, because of differences among formal, colloquial, and popular definitions of words and phrases and the effort required to produce detailed information, these same attributes also contribute to documentation problems. The use of natural language to prescribe complex, dynamic systems has at least three common and severe problems: ambiguity, inaccuracy, and inconsistency."[vii]

THE DRAWBACKS OF ABSTRACT NOTATIONS AND CHECKLISTS

These problems have long been recognized, and systems engineers and have sought solutions to them for years. Until recently, most of these solutions have fallen into two classes: (1) the use of formal and semiformal languages to eliminate natural language from specifications, and (2) the use of guides, rule sets and checklists for writing better natural language requirements. Unfortunately, each of these solution classes comes with problems of its own.

Formalized requirements languages come in a wide variety of forms, ranging from semiformal notations like Unified Modelling Language (UML), User Requirements Notation (URN) and other graphical notations, up to formal methods like Z-notation, KAOS, linear temporal logic (LTL) and proprietary model-based systems Engineering (MBSE) tools. The main problem with all formalized languages is that they cannot be used throughout the entire systems engineering processes.

In virtually all system development scenarios, formalized requirements languages are inappropriate for many stakeholders. Project stakeholders may come from a wide variety of backgrounds and disciplines and represent a broad spectrum of interests. Those in non-technical roles will likely find it unacceptable that they be forced to deal with formalized languages. As a result, some 95% of engineering specifications use some form of natural language (79% use common natural language, 16% use structured natural language) to express requirements. [viii]

To assure requirement quality in these natural language specifications, the most common tool is some sort of guide or checklist. These may be developed internally or by some expert third party. Perhaps the most well-known, respected, and widely used of these third-party requirements quality guides is the Guide for Writing Requirements published by the International Council on Systems Engineering (INCOSE).

Originally published in December 2009 and revised a dozen times since, the INCOSE *Guide for Writing Requirements* (GFWR) is comprehensive, well-structured and highly readable. It is probably the most complete set of best practices for authoring natural language requirements, and it's an excellent reference for those seeking to acquire those best practices.

Unfortunately, the same attributes that make the *GFWR* and similar rule sets good references also make them rather unwieldy in the requirements review process. The *GFWR's* forty-one rules require more than thirty pages of explanation. Just listing them takes up two pages in the *GFWR's* summary sheet. Manually reviewing every requirement in a specification – or even in a change request to an existing specification – *against forty-one different rules* is an arduous, tedious and time-consuming process.

Note: An in-depth exploration of the drawbacks of using abstract notations or best practice checklists in requirements authoring and review is beyond the scope of this solution guide. For a more detailed discussion of this topic, please see our article: https://qracorp.com/requirements-checklist-drawbacks/.

A BETTER WAY TO WRITE AND REVIEW REQUIREMENTS

Fortunately, new analysis tools are taking the tedium out of reviewing specifications against requirements engineering best practices. These tools use advanced natural language processing (NLP) technology to complete in seconds tasks that humans need hours to perform. They automatically analyse individual requirements against many established RE best-practices and point out requirements that may need attention. This allows requirements engineers and analysts to focus more time on *correcting* potential weaknesses by saving them the time they once spent *finding* those weaknesses.

In much the same way syntax checkers and debuggers help software developers refine code, these analysis tools help systems engineers and analysts refine requirements. They provide a quality assessment of each requirement analysed, cueing users to possible sources of ambiguity and allowing them to quickly correct problems *before* sending their requirements for formal review. They've been shown to reduce requirement review and correction time by 50 to 75 percent.[ix,x]

QVscribe for Word and Excel

One of these new NLP analysis tools is QVscribe, an add-in for Microsoft Word and Microsoft Excel that analyses the quality and consistency of requirements inside those applications. By integrating with popular authoring tools and presenting its results directly within them, QVscribe helps engineers and analysts reduce ambiguity and improve clarity at the earliest stages of development.

QVscribe automates the tedious steps of the requirement review process. It cues the user to potentially ambiguous requirements and indicates how each may be at odds with accepted best practices. It also provides a risk assessment of each requirement analysed to help users prioritize their review and correction tasks.

Note that QVscribe does not modify any requirements. It only reports where issues exist and why they have been pointed out. It is up to users to validate the tool's assessments and make any necessary corrections.

QVscribe comes preconfigured with default analysis settings derived from the INCOSE GFWR and NASA requirements engineering standards. The default trigger words and other settings can be assessed and modified as necessary to conform to each project's specific standards.

Coverage of the INCOSE rule set

INCOSE's forty-one Rules for Requirement Statements and Sets of Requirements are grouped into fourteen categories. This guide will cover those rules covered by QVscribe's analyses by category, in the order in which they appear within the GFWR. Rules not automated by QVscribe are not covered in this guide. Some rules—like using proper spelling, grammar and punctuation—are automated by other software tools, like Microsoft Word. Other rules must be left solely to the expertise of the RE professional.

The Rule Coverage Table (Figure 1) below shows each of the forty-one INCOSE Rules for Requirement Statements and Sets of Requirements. They are grouped according to whether they are addressed by QVscribe, automated by other tools, or left to the RE professional for manual administration.

INCOSE RULES FOR REQUIREMENT STATEMENTS

Rule Category	Rule	Code	Handled By
Accuracy	Define terms.	R4	2B - Term Consistency
Accuracy	Use appropriate units.	R6	2A - Unit Consistency
Accuracy	Avoid vague terms.	R7	1C - Vague Words
Accuracy	Avoid escape clauses.	R9	1D - Optional Escape Clauses
Accuracy	Avoid open-ended clauses	R10	1E - Optional Open-ended Clauses
Concision	Avoid unneeded infinitives.	R11	1K - Superfluous Infinitives
Non-ambiguity	Use defined conventions in expressions.	R16	1G - Continuances
Non-ambiguity	Avoid the use of 'not'.	R17	1B - Negative Imperatives
Non-ambiguity	Avoid using the 'slash' (/).	R18	1C - Vague Words
Singularity	Avoid compound/multiple sentences.	R19	1A - Imperatives
Singularity	Avoid combinators.	R20	1E - Continuances
Singularity	Separate rationale from requirement.	R22	1J - Justification
Singularity	Avoid parentheses and brackets.	R23	1C - Vague Words
Singularity	Refer to diagrams and tables.	R25	1F - Directives
Completeness	Avoid use of pronouns.	R26	1L - Cross-referencing Pronouns
Realism	Avoid using absolutes.	R28	1I - Universals Quantifiers
Uniqueness	Express each requirement only once.	R32	3 - Similarity
Quantifiers	Avoid universal qualifiers.	R34	1I - Universals Quantifiers
Quantification	Give measurable performance targets.	R36	1M - Immeasurable Quantification
Quantification	Avoid non-specific temporal words.	R37	1F - Non-specific Temporal Words
Uniformity	Use terms consistency.	R38	2B - Term Consistency
Uniformity	Define acronyms and use consistently.	R39	2B - Term Consistency
Uniformity	Avoid using abbreviations.	R40	2B - Term Consistency
Uniformity	Use a project-wide style guide.	R41	4A - Configurable Terms
Non-ambiguity	Use correct grammar.	R13	Other Tools
Non-ambiguity	Use correct spelling.	R14	
Non-ambiguity	Use correct punctuation.	R15	
Accuracy	Use the definite article.	R1	The Professional
Accuracy	Use active voice and indentify actor.	R2	
Accuracy	Ensure subject and verb are appropriate.	R3	
Concision	Use a separate clause for each condition.	R12	
Singularity	Enumerate sets of functions.	R24	
Completeness	Avoid using section headers as contect.	R27	
Conditions	State applicability conditions explicitly.	R29	
Conditions	State condition relationships explicitly.	R30	
Uniqueness	Classify requirements by type (attribute).	R31	
Abstraction	State 'what', not 'how'	R33	
Tolerance	Give range for performance quantities.	R35	
Uniformity	Use a project-wide style guide.	R41	
Modularity	Group related requirements.	R43	
Modularity	Conform to structure and patterns.	R44	

Figure 1: Rule Coverage Table for the INCOSE Guide for Writing Requirements

Accuracy Rules

Rule R4: Define terms.

This rule states that terms that have a specific meaning for the project – the names of data variables, for example – should be agreed across the project and defined in a glossary, using a standard which makes glossary terms identifiable in the requirements text. “This is essential for consistency to avoid using the word with its general meaning,” states the *GFWR*.

Compliance with this rule is largely up to the organization and individuals creating the specification. However, verification that the rule is being applied consistently – across all uses of all defined terms in any given specification – can be greatly accelerated and simplified using QVscribe.

QVscribe’s Term Consistency Analyzer detects all noun phrases present in the marked requirements. It displays those noun phrases along with similar detected terms, and shows which requirements contain them. Users can adjust the degree of similarity – using the slider indicated by the arrow in Figure 2 – to hunt for misspellings and confusion of terms.

Individual terms can be focused on by double-clicking on them in the task pane. Doing so will highlight (in the document pane) all requirements where that term appears, and show (in the task pane) a list of similar terms and the Similarity Percentage for each, as illustrated in Figure 3.

The analyser can also help users compile a glossary of terms used in the document. This practice facilitates consistency in terminology across the project, so users can feel confident everyone involved in the project is speaking the same language.

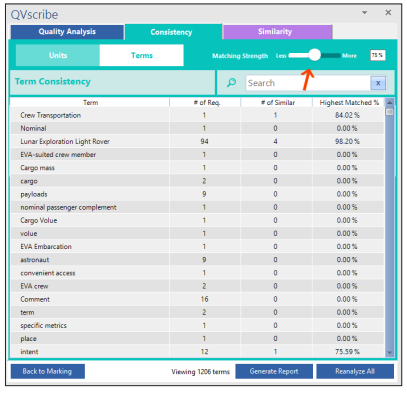


Figure 2: The Term Consistency Analyser

Accidental exchange of one domain-specific term with another, incorrect spelling of a term or acronym (especially if there are similar domain-specific terms within the document) and use of abbreviations can lead to confusion and errors during implementation. The Term Consistency Analyzer addresses this problem by helping users rapidly verify that similarly spelled terms are all valid, up-to-date, spelled correctly, and used correctly in the requirements where they reside. Consequently, it also helps users comply with Rule R38 (Use terms consistently), Rule R39 (Use acronyms consistently) and Rule R40 (Avoid abbreviations).

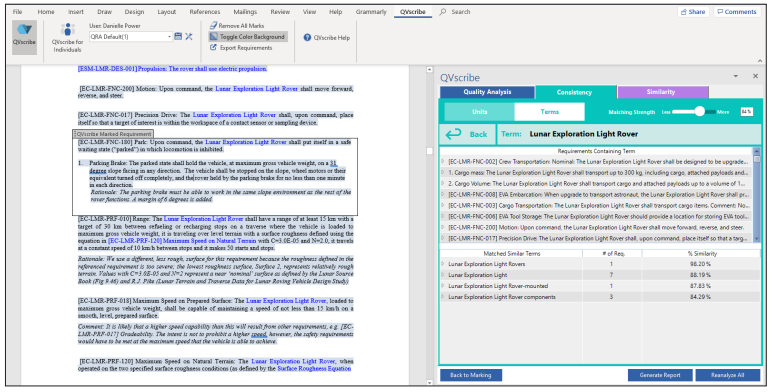


Figure 3: The Term Consistency Analyser – Inspection of individual terms

Rule R6: Use appropriate units when stating quantities.

As stated in the GFWR, this rule is quite succinct: “All numbers should have units of measure explicitly stated. Temperatures must be stated in terms of the measurement system used.”

Improper or inconsistent use of measurement units can lead to requirement incompatibility and ambiguity. This makes requirements more difficult to interpret and test and increases the chances of costly design and implementation errors.

QVscribe’s Unit Consistency Analysis displays statistics for the units of magnitude of physical quantities used in the marked requirements.

QVscribe displays all detected units in the Unit Consistency tab of the Analysis Results screen – sorted by type (length, mass, time, etc.) – along with a count of how many times each unit appears in the document

This feature helps users easily see where different units have been used within the specification, so they can quickly assess compatibility issues (metric vs. imperial, for example), and make needed corrections to ensure all requirements meet project standards.

This feature helps users easily see where different units have been used within the specification, so they can quickly assess compatibility issues (metric vs. imperial, for example), and make needed corrections to ensure all requirements meet project standards.

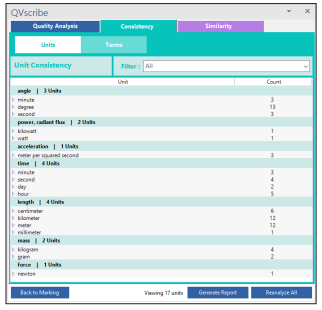


Figure 4: The Unit Consistency Scorecard

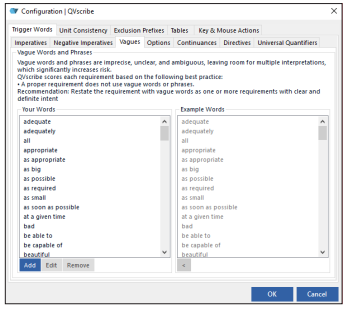


Figure 5: Vague Words & Phrases List

Rule R7: Avoid the use of vague terms.

Vague words and phrases – like *adequate, better, correctly, maximize, minimal, normal,* etc. – tend to make requirements imprecise and ambiguous. They leave room for multiple interpretations, which increases risk. QVscribe's Vague Words Test checks for an extensive list of such vague terms and flags requirements containing them. Users can add or remove terms from the Vague Words and Phrases list via the 'Vagues' pane of the QVscribe configuration window (Figure 5)

QVscribe's Vague Words Test checks for an extensive list of such vague terms and flags requirements containing them. Users can add or remove terms from the Vague Words and Phrases list via the 'Vagues' pane of the QVscribe configuration window (Figure 5)

Often, requirements authors use vague terms without realizing they have done so. Their intention is clear to themselves, so they don't recognize the possibility of misinterpretation by others unless it is brought to their attention. The QVscribe Vague Words Test automatically finds and highlights instances of vague terms, so users can efficiently eliminate any ambiguities. Clicking on the phrase 'Contains vague words' under a requirement in the QVscribe window (as shown in Figure 6, right pane) will highlight the detected vague term in the specification (left pane).

In the requirement shown in Figure 6, the vague phrase "up to" might be interpreted as a maximum storage time, allowing the designer to provide a storage capacity of less than 60 minutes. It is also unclear whether "up to 60 minutes" applies to the total run time capacity, or to individual telemetry or video clips.

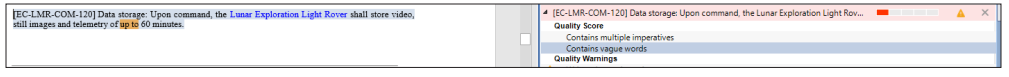


Figure 6: Highlighting of vague words within a specification

Rule R9: Avoid Escape Clauses.

Under Rule R9, the Guide for Writing Requirements states:

> *'Escape clauses give an excuse to the developer of the system at lower levels to not bother with a requirement. They 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." Escape clauses can lead to ambiguous, unverifiable requirements that are open to interpretation and that do not reflect accurately the stakeholder expectations. From a contracting standpoint, requirements with these phrases could be interpreted as being optional.'*

As in the case of vague terms, QVscribe also analyses requirements for the presence of optional escape clauses. It allows users to customize the list of optional phrases the tool checks for, and it highlights the optional words within a requirement when the requirement selected. Figure 9 shows the optional word 'should' highlighted within a requirement when the phrase 'Contains optional words' is selected under the corresponding requirement in the QVscribe analysis window.

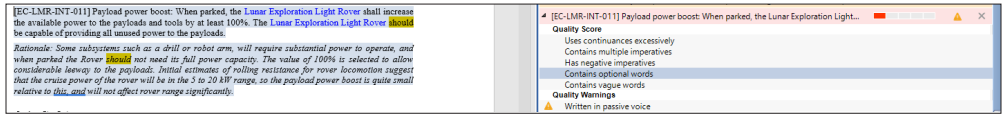


Figure 7: Highlighting of optional words within a specification

Like vague words, escape clauses can be easy to overlook in the authoring and initial review processes. Often, it's only when an implementer interprets a requirement differently than what the specifier intended that such words are noticed. QVscribe helps lower development costs by bringing escape clauses to the specifier's attention before any implementation takes place.

Rule R10: Avoid open-ended clauses.

'Open-ended clauses' are those that include phrases like 'including but not limited to', 'and so on', 'etc.' et cetera. They indicate more is required without specifying what that 'more' is. *"Depending on the contract type (fixed price vs level of effort) opened ended requirements can lead to serious interpretation problems concerning what is in or out of scope of the contract,"* states the *GFWR.*

QVscribe checks for open-ended clauses and highlights instances in the quality analysis as shown in Figure 8. Users can add to QVscribe's default list of open-ended clause keywords via the QVscribe configuration window.

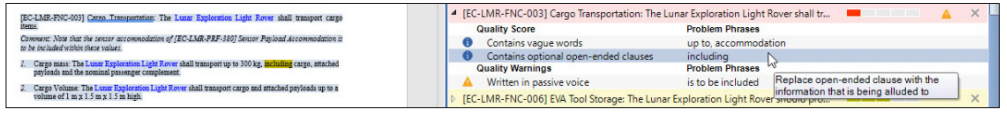


Figure 8: Highlighting of an optional open-ended clause within a specification

CONCISION

Rule R11: Avoid superfluous infinitives.

In Rule R11, the GFWR warns authors to avoid writing requirements with *"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 …'."*

Requirements containing such superfluous verbs can cause problems in verification. It can be argued that a system meets a requirement 'to be capable of' or 'to be able to' deliver some result – *even if it fails frequently* – so long as it delivers that result some of the time.

QVscribe's Superfluous Infinitives Test (see Figure 9, below) looks for passive infinitive phrases like 'be able to', 'be capable of' and 'be designed to' and highlights them to the user for further scrutiny.

Figure 9: Highlighting of a superfluous infinitive within a specification

Non-ambiguity

Rule R16: Use a defined convention to express logical expressions.

INCOSE recommends organizations define a convention for logical expressions such as [X AND Y], [X OR Y], [X XOR Y] and NOT[X OR Y], so that the logical operators (AND, OR, NOT, etc.) of these expressions are not confused with the use of these words as combinators or continuances which, in general, are to be avoided in requirement statements. (See Rule R20: Avoid combinators.)

QVscribe does not specifically test for logical expressions, but in specifications containing such expressions, it may benefit users to include logical operators in QVscribe's list of 'Continuances'. In this way, the user can review all uses of these words to make sure all logical expressions conform to their organization's standards. Once this has been done, the user can then exclude continuances from requirements analysis, by checking the 'Exclude from analysis' box in the Continuances configuration window. Continuances will be discussed in greater detail under Rule R20.

Rule R17: Avoid the use of 'not'.

The presence of the word 'not' in a requirement, especially in combination with the imperative, can lead to problems in verification. As the Guide for Writing Requirements points out, 'not' implies 'not ever' which is impossible to verify within a finite time.

QVscribe's Negative Imperatives Test automatically checks for the presence of the word 'not' in conjunction with imperatives. QVscribe highlights all requirements containing negative imperatives, so users can restate them as one or more requirements in a verifiable form using a positive imperative. QVscribe comes with a default list of negative imperatives which the user can add to or edit in the Negative Imperatives pane of the QVscribe configuration window shown in Figure 10.

Figure 11: Highlighting of negative imperatives

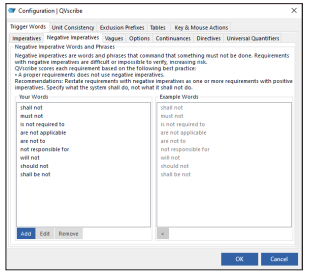


Figure 10: Negative Imperatives Menu

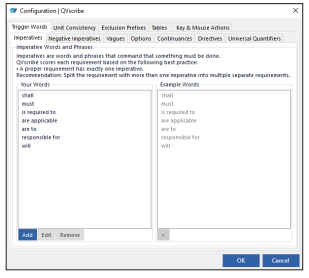


Figure 11: The Imperatives Menu

Rule R18: Avoid use of the oblique (/) symbol or 'slash'.

The oblique or 'slash' symbol ('/') is used in such a variety of ways ('and', 'or', 'and/or', etc.) that its meaning to readers is inherently vague. Except for rate units (km/h, for example), it is best to avoid using the slash in requirements.

QVscribe does not list the oblique or other symbols among the default trigger words in any of its requirement quality tests. Users may, however, add the oblique using the QVscribe Configuration Window. Because the oblique is essential to many rate units, users may find it convenient to add the symbol to the list of Continuances, as these can be excluded from analysis – by checking the 'Exclude from Analysis' box in the Continuances pane – once all instances have been checked.

Continuances will be discussed in greater detail under Rule R20.

Rule R19: Use a single sentence. Avoid compound and multiple sentences.

A requirement should be stated as a simple affirmative declarative sentence with a single subject, a single main action verb (preceded by an imperative like 'shall' or 'must') and a single object, framed and qualified by one or more sub-clauses.

Compound sentences often contain multiple requirements. These should be broken into separate sentences and numbered as separate requirements. Each separate requirement should contain one and only one imperative.

Requirements without an imperative should be corrected to include one. Otherwise, nothing is required; a requirement statement without an imperative is not a requirement.

Finally, when multiple sentences are used to express a requirement, the writer may be using the additional sentences to provide rationale for the requirement statement. In such cases, it is best to separate the rational from the requirement and label it as an attribute using a prefix, like 'Rationale:' or 'Motivation:'. This procedure will be discussed in more detail under Rule R22.

Multiple imperatives within a requirement statement normally indicate that the statement should be broken up into multiple requirements. QVscribe's Imperatives Test checks that each requirement statement contains one and only one imperative, and flags those requirements containing either multiple imperatives or no imperative, as shown in Figures 12 and 13. Notice in Figure 12 that steps 3 and 4 are distinct requirements; they should be re-stated separately from the base requirement and given their own unique identifiers.

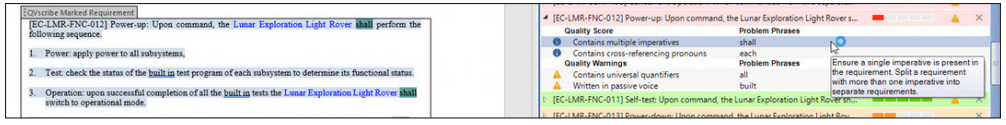
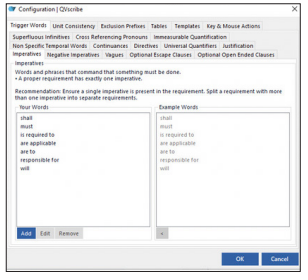


Figure 12: Detection of multiple imperatives within a requirement statement

Figure 13: Detection of no imperative within a requirement statement

QVscribe's list of imperatives analysed in the Imperatives Test can be modified to conform to an organization's standards via the Imperatives pane of the QVscribe Configuration Window (Figure 14).



Rule R20 – Avoid Combinators.

Combinators are conjunctions ('and', 'but', 'or', 'however', 'whether', etc.) and other words and phrases that combine clauses. Their presence in requirements often indicates that the statement contains more than one requirement and should be broken up (see Rule R19).

In fact, combinators are often present in requirement statements that contain multiple imperatives. However, a combinator may attach one requirement clause to another in a way that "piggybacks" the second clause on the imperative of the first clause.

Some combinators – 'AND', 'OR', 'NOT', for example – are used in logical expressions. For this reason, INCOSE recommends defining a specific notational convention for logical expressions (Rule R16) to avoid ambiguity. QVscribe refers to combinators as continuances. It's Continuances Test checks for excessive use of continuances in requirement statements and flags instances of such use. Figure 15 shows a requirement in which QVscribe has detected multiple continuances.

Figure 15: Detection of multiple continuances (combinators) within a requirement statement

Users may modify the list of checked continuances via the Continuances pane of the QVscribe Configuration Window.

Since continuances are also used in ways other than combining clauses, they do not necessarily indicate that a requirement should be broken up into multiple requirements. For that reason, QVscribe allows users to exclude continuances from analysis. This is done by checking the 'Exclude from Analysis' check box in the Continuances pane of the QVscribe configuration window (Figure 16).

With this feature, users can check for continuances and correct requirements that need to be broken up on an initial pass, then exclude continuances from subsequent analyses. By excluding continuances users can remove flags from requirements that use continuances correctly and thus declutter their requirements scorecard once ill-used continuances have been eliminated.

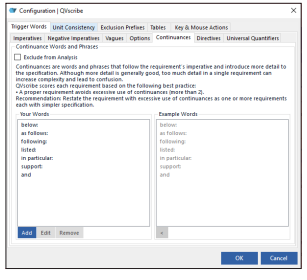


Figure 16: The Continuances Menu

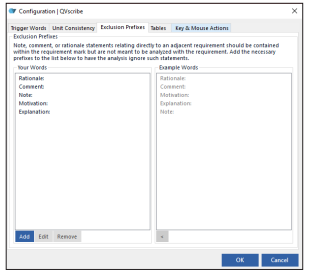


Figure 17: The Exclusion Prefixes Menu

Rule R22: Avoid phrases that indicate the purpose of the requirement.

"The text of a requirement does not have to carry around extra baggage such as the purpose for its existence," states Rule R22 of the *GFWR*. "This extra information should be included in the requirement attribute A1- Rationale."

As was mentioned under Rule R19, it is always best to separate rationale from requirement. This can be done by tagging each supporting with a prefix, like 'Rationale:' or 'Motivation:'. in much the same way as requirements statements are tagged with unique identifiers in best-practice RE procedures.

QVscribe provides the capability to exclude rationale, comments, notes and other supporting attributes from its requirements analysis, so long as paragraphs containing that information are preceded by an attribute prefix. This can be done by including the attribute prefix in the list of Exclusion Prefixes in the QVscribe Configuration Window (Figure 17). By excluding this supporting information from analysis, users get a more accurate indication of where work is needed within the actual requirement statements themselves.

Phrases like 'in order to', 'so that', and 'thus allowing' often indicate the presence of rationale within the requirement statement. QVscribe checks for these and similar phrases, and indicates there presence with a quality warning, as shown in Figure 18, below. Note in Figure 18 the paragraph following the requirement labeled 'Rationale:' This and similarly labeled paragraphs can be excluded from analysis by including 'Rationale:' in the Exclusion Prefixes list.

Figure 18: Detection of justification phrases in a requirement statement

Rule R23: Avoid parentheses and brackets containing subordinate text.

Parentheses or brackets within requirements statements usually indicate the presence of explanatory information that can either be communicated as rationale (see Rule R22) or simply removed. Sometimes, the use of parentheses or brackets can introduce ambiguity.

There are some situations, however, where it might be useful to use brackets or parentheses to make requirements clearer, as in the case of logical expressions (see Rule R16). Such situations should be agreed upon within the organization.

Like the oblique (see Rule R18), parentheses, brackets and other symbols are not included in any of QVscribe's default trigger word lists. Thanks to QVscribe's Configuration Window, however, it is easy to add such symbols to QVscribe's analysis. They can be removed just as easily.

It can be useful, therefore, to include parentheses, brackets and other symbols to QVscribe's Vagues list (see Rule R7, Figure 7), for example, on a temporary basis. After running an analysis and checking for those symbols in the analysis results, the user can make any needed corrections and then remove the symbols from the list. Alternatively, symbols can be added to the Continuances list and, again after running an analysis and checking for those symbols in the analysis results, excluding continuances from later analyses.

Rule R25: Refer to diagrams and tables when expressing complex requirements.

Often, it can be very difficult to express a complicated requirement in words. In such cases, it is usually better to refer to a diagram, table or model. Such references are made with directives.

Directives are terms pointing to additional information – examples, tables, figures, etc. – which should clarify and strengthen the requirement.

The use of directives needs to be considered carefully, however. Inattentive use of directives increases the potential for confusion and misinterpretation. Observe the following:

> *"The left wing inner flap ECU shall allow for a signal delay of 5 milliseconds. For examples, see diagram 3.4.2.1, and table 54.32.2."*

The directives in this case would likely lead to assessment of a diagram and a table, so it's important to ensure both these items do indeed increase the clarity of the requirement and NOT the difficulty of interpreting and verifying it. If referenced items contradict the text of the requirement, or each other, implementation and verification will likely be delayed, and errors may be introduced.

QVscribe's Directives Test checks for the presence of keywords that are often used as directives. Users can modify the list of directives to conform to organizational procedures, by adding or removing directive terms via the Directives pane of the QVscribe Configuration Window (Figure 18).

As with continuances (see Rule R20), QVscribe flags requirements containing directives, but also allows the user to exclude directives from analysis. This allows the analyst to quickly check that directives are used properly, then declutter the analysis – through directives exclusion – on a second pass.

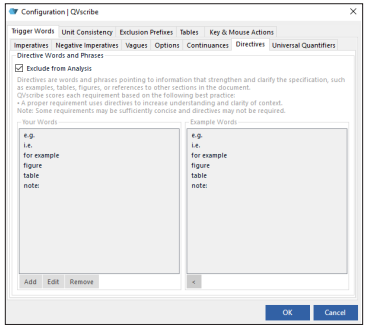


Figure 19: The Directives Menu

Completeness

Rule R26: Avoid the use of pronouns.

Regarding pronouns, the INCOSE *Guide for Writing Requirements* says,

> *"When writing stories, pronouns are a useful device for avoiding the repetition of words; but when writing requirements, pronouns are effectively cross references to nouns in other requirements and, as such, are ambiguous and should be avoided."*

The *GFWR* advises requirements authors to, "Repeat nouns in full instead of using pronouns to refer to nouns in other requirement statements." For while pronouns can be used to avoid repetitive use of nouns within a sentence, it must be absolutely clear to which noun a pronoun refers.

QVscribe checks for the presence of pronouns with its Cross-Referencing Pronouns Test, and flags requirements containing pronouns in the Quality Analysis window (Figure 20).

Figure 20: Detection of cross-referencing pronouns within a requirement statement

Realism

Rule R28: Avoid using absolutes.

The GFWR advises RE professionals to "avoid using unachievable absolutes."

QVscribe refers to these 'absolutes' as *universal qualifiers.*

Universal qualifiers are words and phrases like *all, always, every, never, often, usually, etc.*, which generalize quantities or sets of conditions related to a subject. Their presence can make requirements difficult or impossible to verify without a thorough understanding of their context.

Introducing a universal qualifier often opens the door to interpretation and can lead to unnecessary and costly system functionality. Consider the following requirement:

> *"The light lunar rover shall be able to navigate all types of terrain."*

Here, 'all types of terrain' can be interpreted to include water. The design of the lunar rover might then include some sort of flotation device and propeller... despite the lack of water on the moon!

However, there are situations where a universal quantifier is called for, as in this example:

> *"The left wing inner flap ECU shall disregard all automatic control signals when manual override has been switched to the ON position."*

As with directives and continuances, universal quantifiers need to be used with care. The goal of QVscribe's quality analysis for these indicators is to bring them to the user's attention. The user can then apply correction, if needed. For this reason, universal quantifiers do not factor in QVscribe's requirements scoring. Instead, QVscribe gives a warning when a universal quantifier is encountered in a requirement, as illustrated in Figure 21.

Figure 21: Detection of universal quantifiers within a requirement statement

Uniqueness

Rule R32: Express each requirement once and only once.

In assembling a large requirements document for a complex project – especially when multiple authors are involved – there is a distinct possibility of introducing requirements which are similar. These similar requirements may be redundant or may even contradict one another. Such occurrences can lead to confusion, redundancies, incompatibilities, delays and added cost in both the implementation and verification phases of the project.

Finding exact duplicates is relatively straightforward using string matching. Unfortunately, finding similar requirements with slightly different wording is much more difficult.

QVscribe's Requirement Similarity Analyzer (Figure 22) gauges the similarity between all requirements confirmed for analysis. Users may adjust the degree of similarity ("matching strength") required for display using a slider (indicated by the arrow in Fig. 22), which allows them to find and filter requirements with similar wording.

To help users compare similar requirements, QVscribe adds a Similarity Pane at the bottom of the screen when a matched requirement is selected in the analysis window, as shown in Figure 22. QVscribe displays the two requirements side by side, highlighting similarities in red and differences in blue.

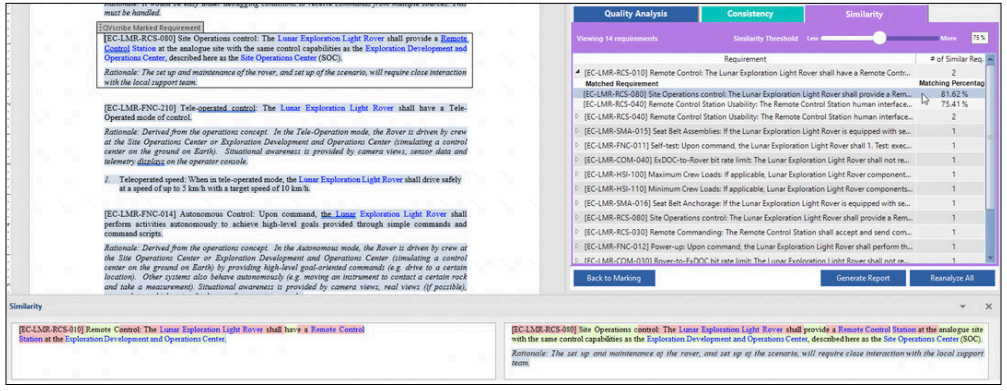


Figure 22: The QVscribe Requirement Similarity Analyzer

Setting the Matching Strength in the **95% to 100%** range makes the analyser display only **identical** requirements. This makes for easy detection and elimination of duplicate requirements. Eliminating duplicate requirements reduces verification effort and cost.

Setting the Matching Strength in the **75% to 95%** range, as shown in Figure 20, causes the analyser to also display requirements which are similar in structure and content. This facilitates quick comparisons of similar requirements to make sure they don't conflict with or duplicate one another. Elimination of requirements conflicts during the requirements definition phase reduces project delay and correction costs later in the project lifecycle and thus lowers overall lifecycle cost.

Quantifiers

Rule R34: Avoid universal qualifiers

The INCOSE *Guide for Writing Requirements* recommends against using universal quantifiers like 'all', 'any' and 'both', because "it is hard to distinguish whether the action happens to the whole set or to each element of the set." The word 'all' can prove especially problematic in verification and can often be simply removed.

QVscribe checks for universal qualifiers in its Universal Qualifiers Test, which was described earlier under Rule R28 (Avoid using absolutes).

Quantification

Rule R36: Provide specific measurable performance targets

Many words – such as 'prompt', 'fast', 'minimum', 'maximum', 'optimum', and similar – indicate quantities that cannot be objectively measured. Such terms are inherently ambiguous and must be replaced by specific, measurable quantities.

QVscribe looks for such quantifications in its Immeasurable Quantification test. Requirements containing immeasurable quantifications are flagged in the in the QVscribe Quality Analysis window, as shown in Figure 23. Users can modify the list of words and phrases QVscribe checks for in the test via the Immeasurable Quantification pane of the QVscribe Configuration Window.

Figure 23: Detection of immeasurable quantification within a requirement statement

Rule R37: Avoid using non-specific temporal words

Some words and phrases signal non-specific timing. Indefinite temporal words like 'eventually', 'earliest', 'latest', 'instantaneous', 'simultaneous', and similar should be replaced by specific timing constraints.

QVscribe checks for many non-specific temporal words in its Non-specific Temporal Words Test and highlights those found in the Quality Analysis window (Figure 24). Other such terms can be added to the test via the QVscribe Configuration Window.

Figure 24: Detection of non-specific temporal words within a requirement statement

Uniformity of Language

Rule R38: Use terms consistently throughout requirement sets

Besides defining terms in a glossary (Rule R4), organizations need to ensure those defined terms are used consistently throughout the specification to ensure that only one term is applied to identify a specific concept. QVscribe's Term Consistency Analyzer can be used to quickly identify terms that are similar, but not exactly alike, so that those that differ from the agreed term can be quickly corrected. The Term Consistency Analyzer was described in detail under Rule R4.

Rule R39: Define acronyms and use them consistently

Like other terms, acronyms must be agreed upon, defined in a glossary, and used consistently throughout a specification. As for other natural language terms, QVscribe's Term Consistency Analyzer can be used to ensure consistent use of acronyms throughout the specification. See Rule R4 for a through description of the Term Consistency Analyzer.

Rule R40: Avoid using abbreviations in requirement statements

INCOSE advises against using abbreviations in requirement statements, because related terms with similar spellings may have the same abbreviation. The abbreviation 'op', for example, could mean 'operation' in one requirement, 'operator' in another, and thus be misinterpreted in either. In cases where abbreviations are beneficial – as in the case of units of measurement – they should be standardized and included in the project's glossary of terms (see Rule R4).

QVscribe's Term Consistency Analyzer (see Rule R4 for explanation) checks for words with similar spellings and thus can be used to find requirements in which an abbreviation is used. Abbreviations used frequently within the context of a given project can also be added to QVscribe's Vagues list (described in Rule R7), so they will be flagged by the tool's Vague Words Test for modification.

Rule R41: Use a project-wide style guide

Using a project-wide style guide is a best practice which will make both your individual requirements and your specifications easier to use and understand. Among other things, your style guide should define:

- Templates to be used for various project specifications
- Which rules (based on the INCOSE Guide or others) your organization wants to use
- Which imperatives are acceptable for use in the project's requirements, and which are not
- What prefixes may be used to identify paragraphs of rationale or comment
- What attributes (unique identifier, etc.) should be attached to each requirement
- How each attribute should be formatted
- A glossary of standard terms and acronyms

You may also want to specify patterns that are to be used for specific types of requirements.

While it cannot define a style guide for your organization, QVscribe can help team members adhere to the style guide you define. As mentioned earlier, QVscribe's Configuration Window allows you to define which imperatives and units of measurement maybe used within requirements, and which will be flagged for correction. You may also define which prefixes may be used to separate rationale and other commentary from requirement statements, so that QVscribe will recognize such paragraphs and exclude them from its requirement analysis.

The style guide itself, however, must be defined by your organization. Thus, the bulk of the responsibility for complying with Rule R41 lies with the professionals within your organization. It is for this reason that Rule R41 appears twice in our INCOSE Rule Coverage Table (Figure 1).

Some points regarding the tasks left to the professional

If both QVscribe and a document processing program like Microsoft Word – which checks for spelling, grammar and punctuation errors – are used for analysing and specifying requirements, only fourteen of the forty-one GFWR Rules are left primarily in the hands of the RE professional. All fourteen are best practices skills every RE practitioner should acquire.

Some of the rules – R1, R2, R3 and R27, for example – quickly become second nature to aware RE professionals. Others, like R12, R29, R30, R31 and R33 require a bit more care in application but are easily acquired over time. Writing implementation-agnostic requirements (Rule R33), for example, takes some practice.
The last three rules in INCOSE's list – R41, R43 and R44 – help make requirements and specifications easier to read and understand. Compliance with these rules can be facilitated and simplified, if not automated, using other proven tools.

Compliance with Rule R44 (conform to a defined structure or template for sets of requirements and patterns for individual requirement statements), for example, can be facilitated with document templates and requirement statement patterns that conform to RE best practices. If your organization needs a requirements document template, you can download one here.

Stating requirements according to precisely defined patterns can make them much easier to read and understand. One such set of patterns that is simple and intuitive, yet highly effective, is the Easy Approach to Requirements Syntax (EARS). QVscribe for Teams provides an EARS Conformance Check that assesses how closely individual requirements conform to the prescribed EARS patterns. To learn more about EARS, you can download one here.

How NLP Analysis Improves the Overall specification

By helping requirements engineers and business analysts follow the INCOSE rules for writing requirements, NLP analysis tools like QVscribe also help ensure that individual requirements and sets of requirements possess their desired characteristics.

In Section 2 of the *Guide for Writing Requirements,* INCOSE lists and describes nine characteristics, or properties, that every requirement statement should possess. These nine essential characteristics of requirement statements are:

- Necessary
- Appropriate
- Unambiguous
- Complete
- Singular
- Feasible
- Verifiable
- Correct
- Conforming

In Section 3 of the *Guide for Writing Requirements*, INCOSE lists and describes nine characteristics that a set of requirements should possess. These five essential characteristics of requirements statements are:

- Complete
- Consistent
- Feasible
- Comprehensible
- Able to be validated

Figure 25, below, shows how each of the INCOSE Rules for Requirement Statements and Sets of Requirements contributes to achieving the characteristics listed above.

As can be seen, NLP Analysis can provide the lion's share of the coverage in achieving these desired characteristics.

While QVscribe doesn't make corrections to requirements, it does point out weaknesses that could lead to misinterpretation and to subsequent design errors. And it finds all those weaknesses – throughout the entire specification – in seconds.

Effective NLP analysis helps RE professionals correct requirement errors quickly, saving them hours of work. Plus, it saves their organizations the exponentially greater costs of correcting those errors later, in subsequent phases of the project lifecycle.

In short, NLP analysis helps assure overall specification quality by (1) automating majority of requirements quality assurance tasks, (2) reducing requirement review and correction time by as much as 50 to 75 percent, and (3) allowing engineers and analysts more time to focus on a much smaller set of tasks that truly require their expertise.

INCOSE RULES FOR REQUIREMENT STATEMENTS

			Characteristics of Requirement Statements									Characteristics of Sets of Requirements				
			Necessary	Appropriate	Unambiguous	Complete	Singular	Feasible	Verifiable	Correct	Conforming	Complete	Consistent	Feasible	Comprehensible	Able to be Validated
Rule	Code	Handled By	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14
Define terms.	R4	2B - Term Consistency			✓				✓						✓	✓
Use appropriate units.	R6	2A - Unit Consistency			✓	✓			✓	✓			✓			
Avoid vague terms.	R7	1C - Vague Words			✓	✓			✓							
Avoid escape clauses.	R9	1D - Optional Escape Clauses			✓	✓			✓							
Avoid open-ended clauses	R10	1E - Optional Open-ended Clauses			✓	✓	✓		✓							
Avoid unneeded infinitives.	R11	1K - Superfluous Infinitives			✓				✓							
Use defined conventions in expressions.	R16	1G - Continuances			✓											
Avoid the use of 'not'.	R17	1B - Negative Imperatives			✓				✓							
Avoid using the 'slash' (/).	R18	1C - Vague Words			✓				✓							
Avoid compound/multiple sentences.	R19	1A - Imperatives			✓		✓				✓				✓	
Avoid combinators.	R20	1E - Continuances			✓		✓									
Separate rationale from requirement.	R22	1J - Justification					✓									
Avoid parentheses and brackets.	R23	1C - Vague Words					✓									
Refer to diagrams and tables.	R25	1F - Directives			✓		✓									
Avoid use of pronouns.	R26	1L - Cross-referencing Pronouns			✓	✓			✓							
Avoid using absolutes.	R28	1I - Universals Quantifiers						✓	✓					✓		
Express each requirement only once.	R32	3 - Similarity	✓								✓		✓	✓		
Avoid universal qualifiers.	R34	1I - Universals Quantifiers			✓				✓	✓						
Give measurable performance targets.	R36	1M - Immeasurable Quantification			✓	✓			✓					✓		
Avoid non-specific temporal words.	R37	1F - Non-specific Temporal Words			✓	✓			✓							
Use terms consistency.	R38	2B - Term Consistency			✓					✓	✓		✓		✓	✓
Define acronyms and use consistently.	R39	2B - Term Consistency			✓						✓		✓		✓	✓
Avoid using abbreviations.	R40	2B - Term Consistency									✓		✓		✓	✓
Use a project-wide style guide.	R41	4A - Configurable Terms									✓		✓		✓	✓
Use correct grammar.	R13	Other Tools			✓						✓					
Use correct spelling.	R14	Other Tools			✓											
Use correct punctuation.	R15	Other Tools			✓											
Use the definite article.	R1	The Professional			✓				✓							
Use active voice and indentify actor.	R2	The Professional			✓				✓							
Ensure subject and verb are appropriate.	R3	The Professional		✓	✓				✓			✓				✓
Use a separate clause for each condition.	R12	The Professional			✓											
Enumerate sets of functions.	R24	The Professional			✓		✓									
Avoid using section headers as contect.	R27	The Professional				✓										
State applicability conditions explicitly.	R29	The Professional				✓			✓							
State condition relationships explicitly.	R30	The Professional			✓				✓							
Classify requirements by type (attribute).	R31	The Professional										✓	✓	✓		
State 'what', not 'how'	R33	The Professional		✓												
Give range for performance quantities.	R35	The Professional			✓	✓		✓	✓	✓				✓		
Use a project-wide style guide.	R41	The Professional									✓		✓		✓	✓
Group related requirements.	R43	The Professional					✓				✓		✓		✓	
Conform to structure and patterns.	R44	The Professional										✓	✓		✓	✓

Figure 25: Coverage of Rules and Characteristics for Requirements and Sets of Requirements

CONCLUSIONS

Advanced NLP analysis tools, like QRA Corp's QVscribe, make conformance to RE best practices simpler, easier and faster. Their configurability also makes it easy to tailor them to internal processes and procedures. NLP analysis tools free RE professionals from tedious tasks by automating those tasks, and by providing immediate indication of where (and what) attention may be needed to eliminate possible ambiguities. Professionals then have more time to focus their attention where it's really needed.

In summary, the major benefits of using NLP requirement analysis tools are:

- Significantly fewer errors made and left uncorrected during requirements definition
- Much less time spent reviewing and correcting requirements
- Errors are corrected before they propagate into design and become costlier to repair
- Much higher specification quality and usability
- Reduced development costs

ABOUT QRA

QRA Corp's mission is to accelerate the design process across industries who are tackling the most complex systems by empowering them to build tomorrow's safe, secure, and incredibly powerful products. QRA's technology, patented toolsets and capabilities have been used to avoid stressful reworks, enable confident engineering, and find previously undetected catastrophic flaws.

QRA's requirements analysis tool, QVscribe, harnesses Natural Language Processing to automatically apply the best requirements analysis tactics by leading industry experts. Automated requirements analysis empowers engineering teams to build faster by identifying errors where they matter most - in the requirements.

To learn more about QVscribe and find additional helpful resources for improving your requirements and your RE processes, visit qracorp.com/qvscribe.

To discover how QVscribe can help your organization improve and accelerate its requirements definition and analysis processes, click here to schedule an online demonstration.

QRA Corp
101-6080 Young Street
Halifax, Nova Scotia B3K 5L2
Canada
Email: sales@qracorp.com
Tel: 1.902.422.0212

REFERENCES

[i] Jonette M.,et al, *Error Cost Escalation Through the Project Life Cycle*, NASA Johnson Space Center and INCOSE, June 2004.

[ii] Boehm, B. W., *Software Engineering Economics*, Prentice-Hall, 1981.

[iii] Rothman, J., *What does it cost to fix a defect?*, Stickyminds.com, August 2002.

[iv] McGibbon, T, *Return on Investment from Software Process Improvement*, The DoD Software Tech News, Volume 5, Number 4, November 2002.

[v] Cigital, *Case study: Finding defects earlier yields enormous savings*, Cigital, 2003.

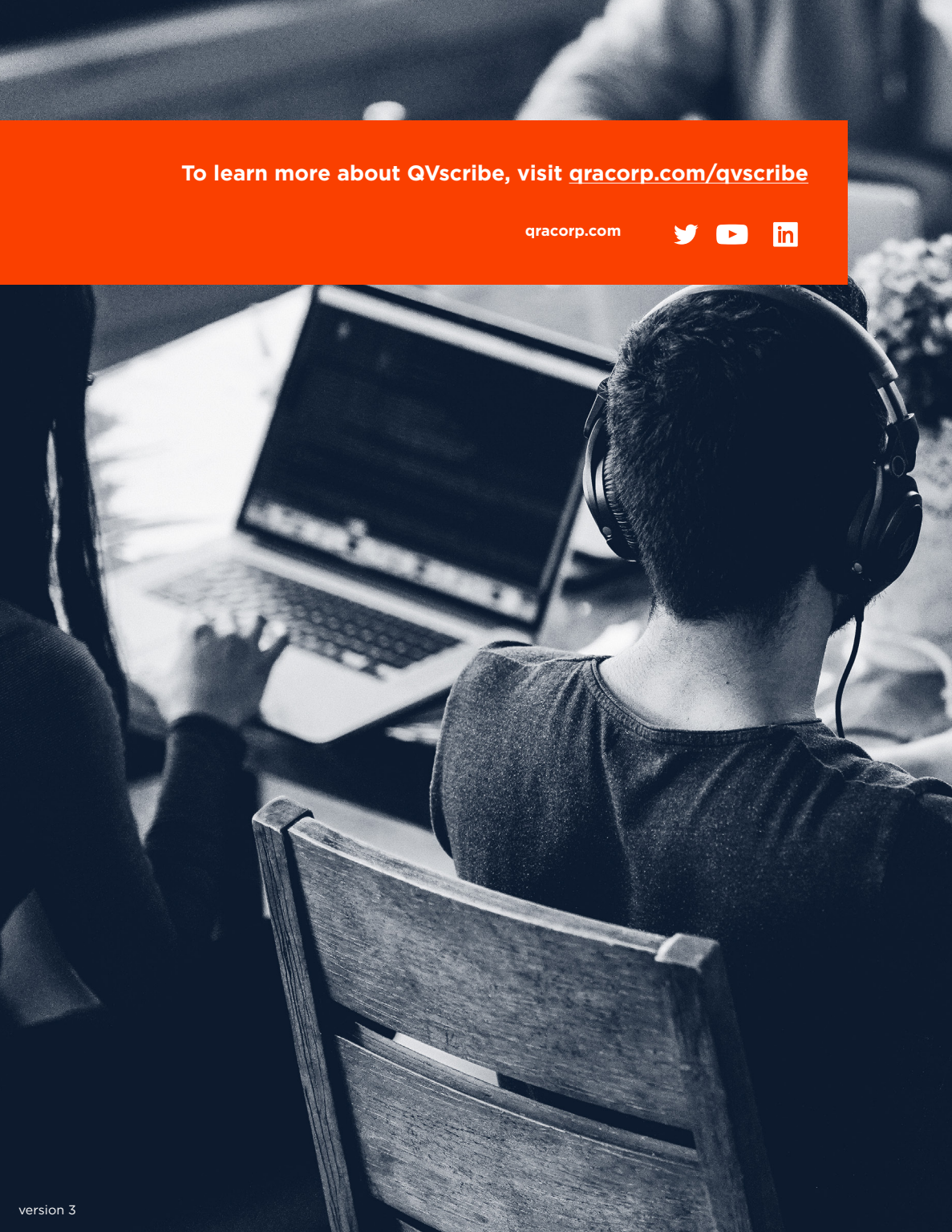
[vi] Martin, James, *An Information Systems Manifesto*, Prentice Hall, January 1984.

[vii] Wilson, William, *Writing Effective Natural Language Requirements Specifications*, Crosstalk, February 1999.

[viii] Mich, L., Franch, M., Novi, I.P.: *Market research for requirements analysis using linguistic tools*. Requirements Engineering, Vol. 9, pp. 4056, 2004.

[ix] *How the Royal Canadian Air Force is Reducing Requirements Review Time by Over 50% with QVscribe*, QRA Corp, January 2019.

[x] *How Ultra Electronics Maritime Systems Reduced the Time and Cost to Revise Requirements by 75% or More*, QRA Corp, January 2019.



To learn more about QVscribe, visit qracorp.com/qvscribe

qracorp.com

version 3