ReqInsights

SUMMARY

50 Documents Analyzed

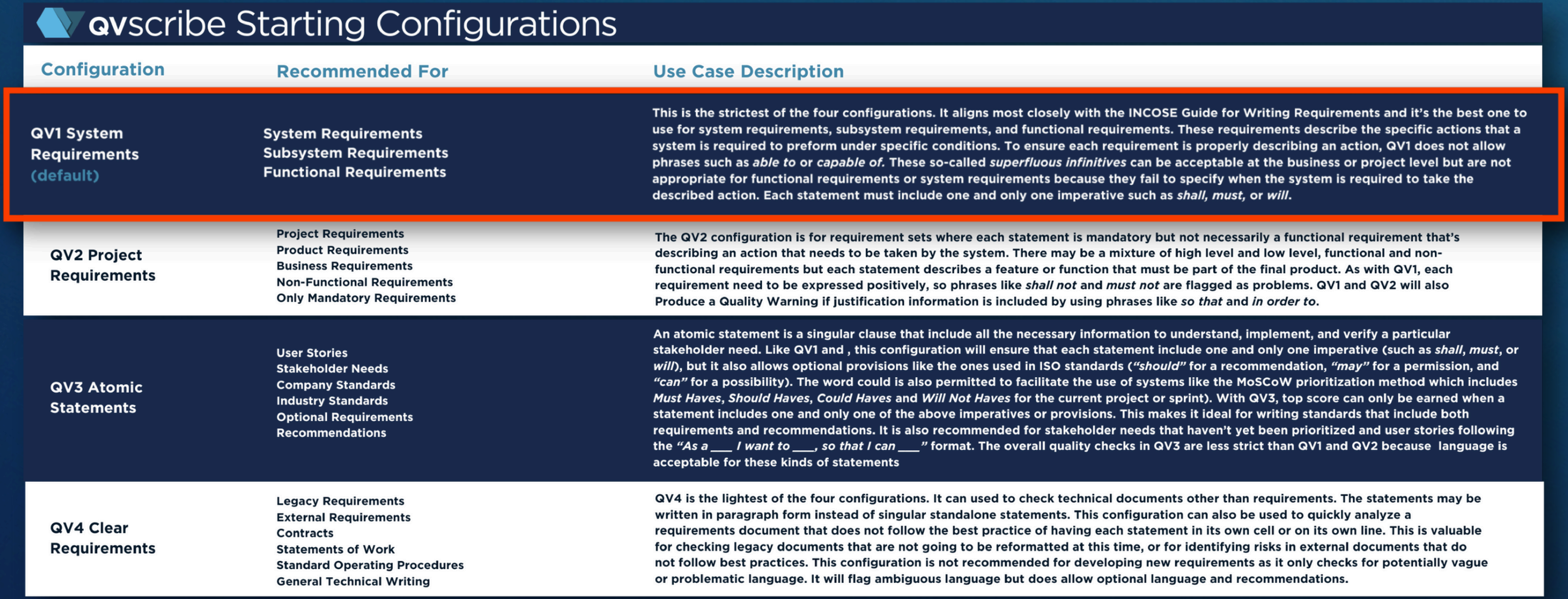
12,955 Requirements Analyzed

Across Various Industries

- Oil & Gas
- Telecommunications
- Medical Devices
- Government
- Shipbuilding
- Automotive
- Electrical Manufacturing
- Defense & Space
- Software Development
- Chemical
- Utilities

QVscribe Configurations

This analysis used the **QV1 System Requirements (default)** configuration.



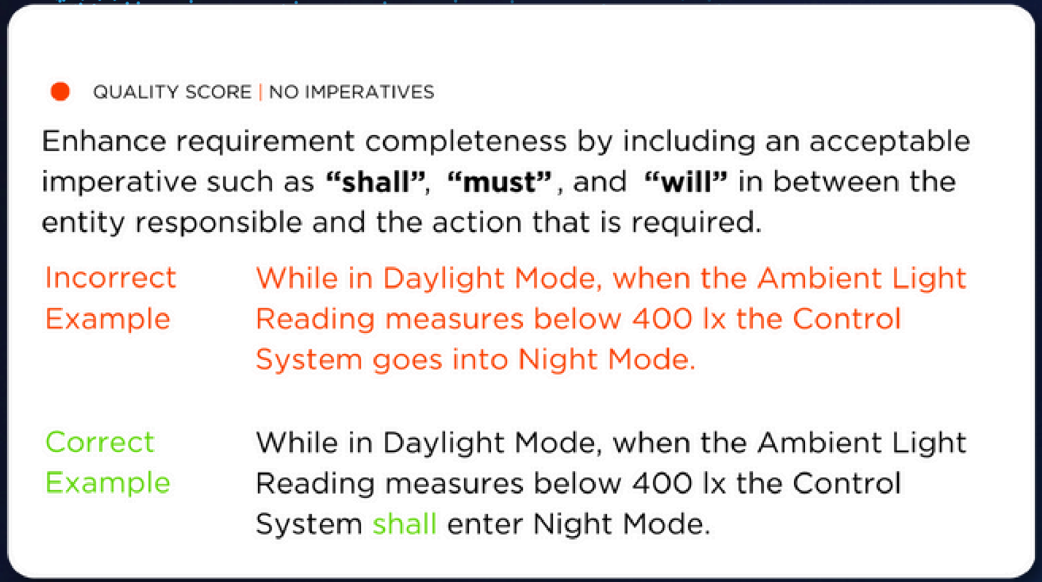
qvscribe Starting Configurations

Configuration	Recommended For	Use Case Description
QV1 System Requirements (default)	System Requirements Subsystem Requirements Functional Requirements	This is the strictest of the four configurations. It aligns most closely with the INCOSE Guide for Writing Requirements and it's the best one to use for system requirements, subsystem requirements, and functional requirements. These requirements describe the specific actions that a system is required to preform under specific conditions. To ensure each requirement is properly describing an action, QV1 does not allow phrases such as *able to* or *capable of*. These so-called *superfluous infinitives* can be acceptable at the business or project level but are not appropriate for functional requirements or system requirements because they fail to specify when the system is required to take the described action. Each statement must include one and only one imperative such as *shall, must,* or *will*.
QV2 Project Requirements	Project Requirements Product Requirements Business Requirements Non-Functional Requirements Only Mandatory Requirements	The QV2 configuration is for requirement sets where each statement is mandatory but not necessarily a functional requirement that's describing an action that needs to be taken by the system. There may be a mixture of high level and low level, functional and non-functional requirements but each statement describes a feature or function that must be part of the final product. As with QV1, each requirement need to be expressed positively, so phrases like *shall not* and *must not* are flagged as problems. QV1 and QV2 will also Produce a Quality Warning if justification information is included by using phrases like *so that* and *in order to*.
QV3 Atomic Statements	User Stories Stakeholder Needs Company Standards Industry Standards Optional Requirements Recommendations	An atomic statement is a singular clause that include all the necessary information to understand, implement, and verify a particular stakeholder need. Like QV1 and , this configuration will ensure that each statement include one and only one imperative (such as *shall*, *must*, or *will*), but it also allows optional provisions like the ones used in ISO standards (*"should"* for a recommendation, *"may"* for a permission, and *"can"* for a possibility). The word could is also permitted to facilitate the use of systems like the MoSCoW prioritization method which includes *Must Haves*, *Should Haves*, *Could Haves* and *Will Not Haves* for the current project or sprint). With QV3, top score can only be earned when a statement includes one and only one of the above imperatives or provisions. This makes it ideal for writing standards that include both requirements and recommendations. It is also recommended for stakeholder needs that haven't yet been prioritized and user stories following the *"As a ___ I want to ___, so that I can ___"* format. The overall quality checks in QV3 are less strict than QV1 and QV2 because language is acceptable for these kinds of statements
QV4 Clear Requirements	Legacy Requirements External Requirements Contracts Statements of Work Standard Operating Procedures General Technical Writing	QV4 is the lightest of the four configurations. It can used to check technical documents other than requirements. The statements may be written in paragraph form instead of singular standalone statements. This configuration can also be used to quickly analyze a requirements document that does not follow the best practice of having each statement in its own cell or on its own line. This is valuable for checking legacy documents that are not going to be reformatted at this time, or for identifying risks in external documents that do not follow best practices. This configuration is not recommended for developing new requirements as it only checks for potentially vague or problematic language. It will flag ambiguous language but does allow optional language and recommendations.

Top Three Problem Types

Of all the requirements we analyzed for this report...

27% had *no Imperatives*

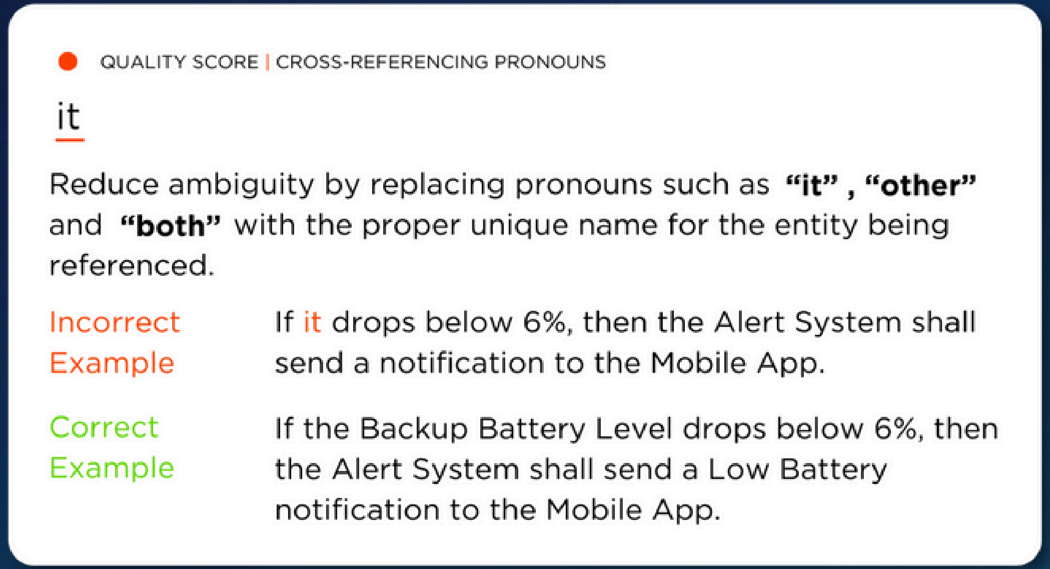


● QUALITY SCORE | NO IMPERATIVES

Enhance requirement completeness by including an acceptable imperative such as **"shall"**, **"must"**, and **"will"** in between the entity responsible and the action that is required.

Incorrect Example	While in Daylight Mode, when the Ambient Light Reading measures below 400 lx the Control System goes into Night Mode.
Correct Example	While in Daylight Mode, when the Ambient Light Reading measures below 400 lx the Control System shall enter Night Mode.

25% used *cross-referencing pronouns*



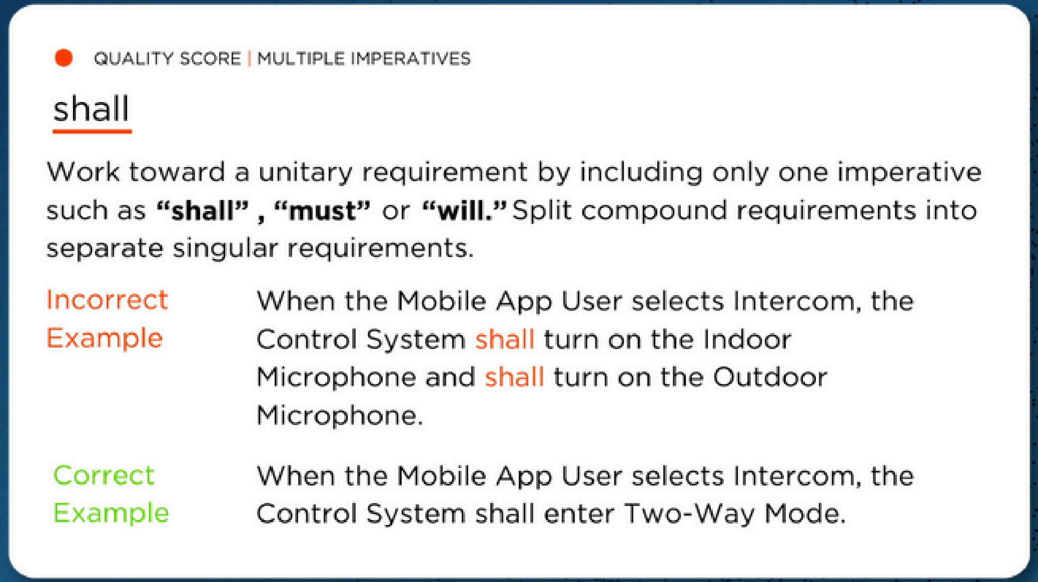
● QUALITY SCORE | CROSS-REFERENCING PRONOUNS

it

Reduce ambiguity by replacing pronouns such as **"it"**, **"other"** and **"both"** with the proper unique name for the entity being referenced.

Incorrect Example	If it drops below 6%, then the Alert System shall send a notification to the Mobile App.
Correct Example	If the Backup Battery Level drops below 6%, then the Alert System shall send a Low Battery notification to the Mobile App.

and 19% used *multiple imperatives*



● QUALITY SCORE | MULTIPLE IMPERATIVES

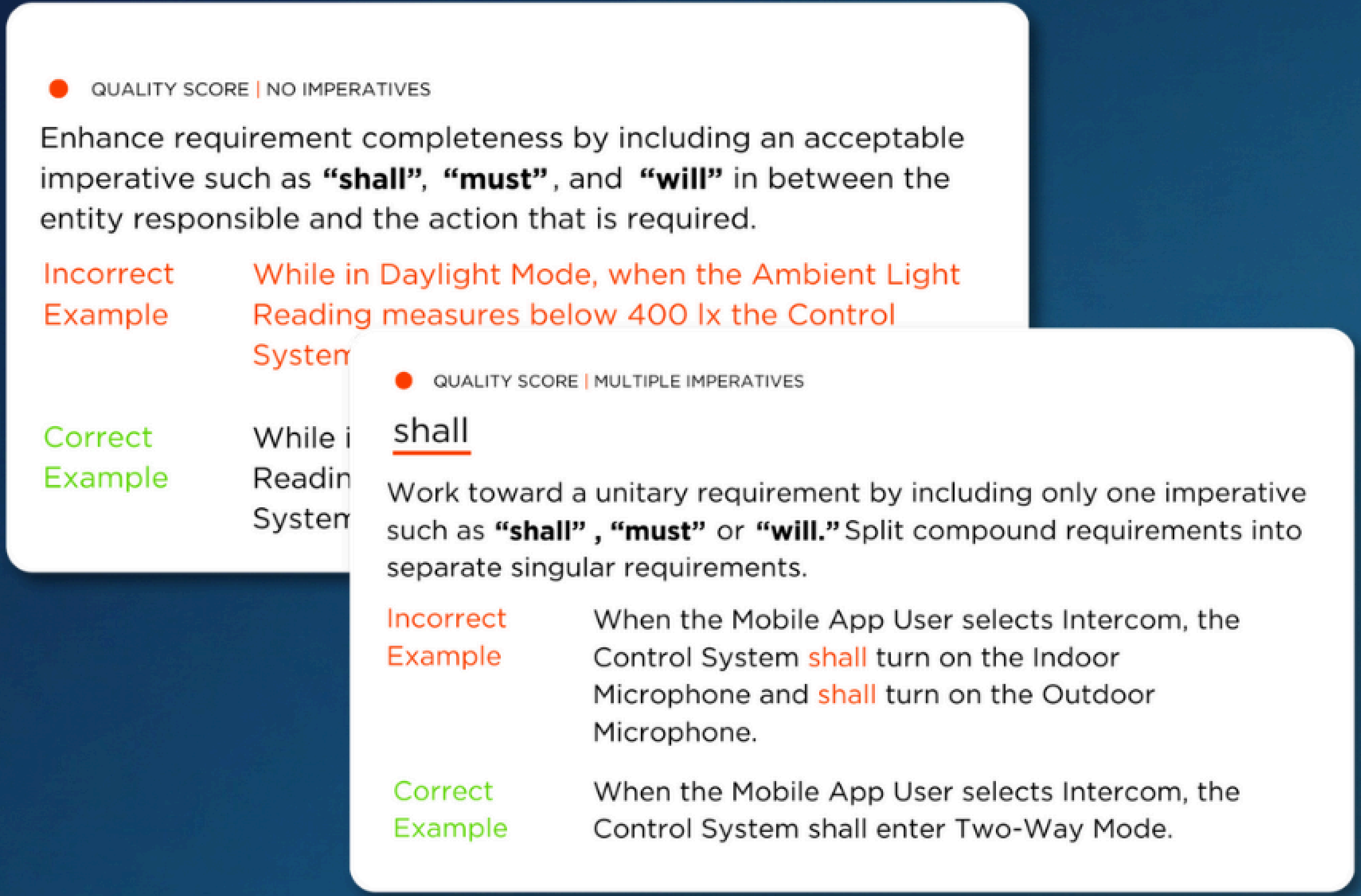
shall

Work toward a unitary requirement by including only one imperative such as **"shall"**, **"must"** or **"will."** Split compound requirements into separate singular requirements.

Incorrect Example	When the Mobile App User selects Intercom, the Control System shall turn on the Indoor Microphone and shall turn on the Outdoor Microphone.
Correct Example	When the Mobile App User selects Intercom, the Control System shall enter Two-Way Mode.

Top Three Problem Types

Missing Imperatives or Multiple Imperatives



● QUALITY SCORE | NO IMPERATIVES

Enhance requirement completeness by including an acceptable imperative such as **"shall"**, **"must"**, and **"will"** in between the entity responsible and the action that is required.

Incorrect Example	While in Daylight Mode, when the Ambient Light Reading measures below 400 lx the Control System
Correct Example	While i Readin System

● QUALITY SCORE | MULTIPLE IMPERATIVES

shall

Work toward a unitary requirement by including only one imperative such as **"shall"** , **"must"** or **"will."** Split compound requirements into separate singular requirements.

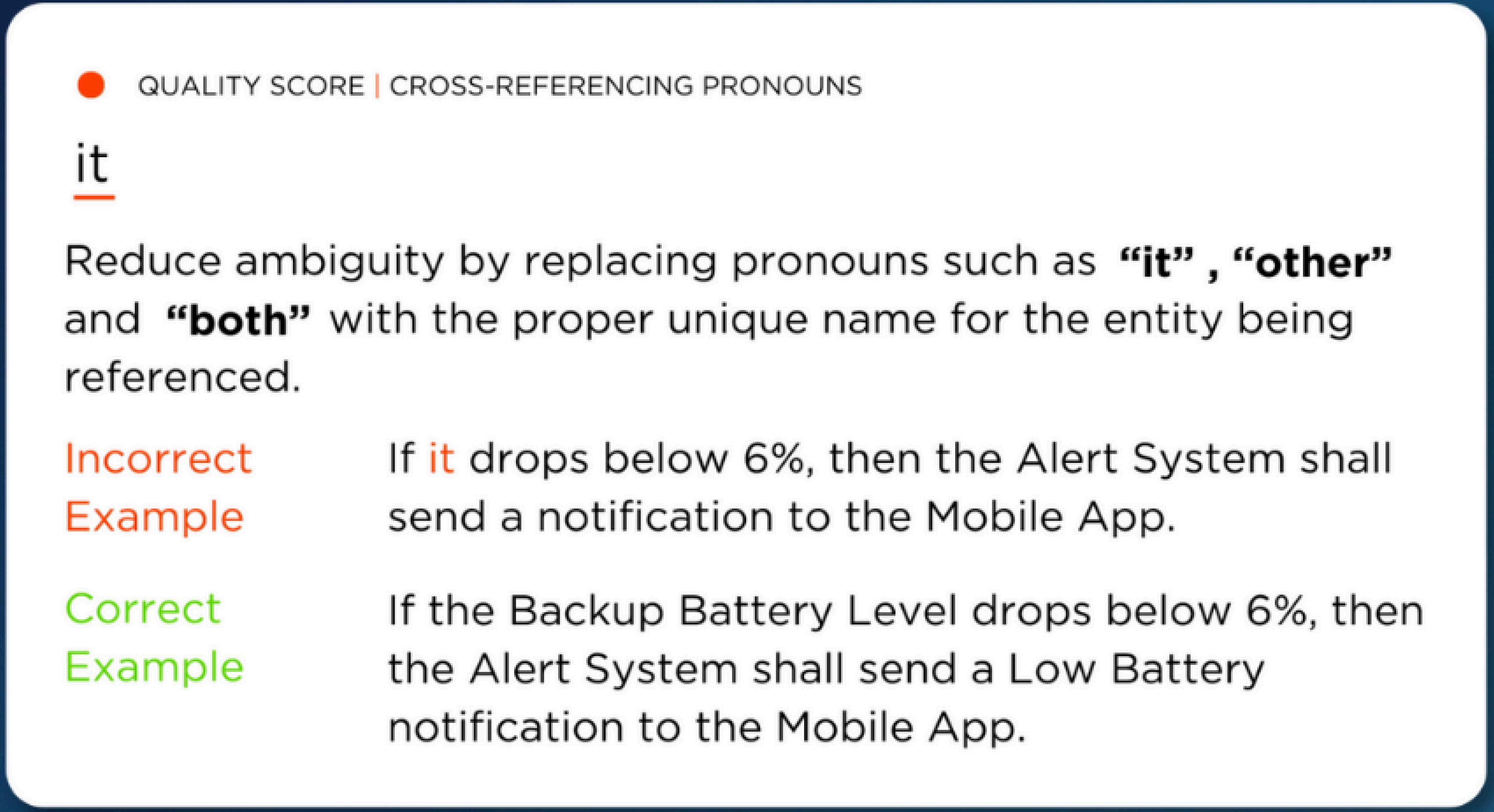
Incorrect Example	When the Mobile App User selects Intercom, the Control System shall turn on the Indoor Microphone and shall turn on the Outdoor Microphone.
Correct Example	When the Mobile App User selects Intercom, the Control System shall enter Two-Way Mode.

Words and phrases that command an action are missing. A proper requirement has exactly one imperative.

Recommendation: *Ensure a single imperative is present in the requirement.*

Top Three Problem Types

Cross-Referencing Pronouns



● QUALITY SCORE | CROSS-REFERENCING PRONOUNS

it

Reduce ambiguity by replacing pronouns such as **“it” , “other”** and **“both”** with the proper unique name for the entity being referenced.

Incorrect Example	If it drops below 6%, then the Alert System shall send a notification to the Mobile App.
Correct Example	If the Backup Battery Level drops below 6%, then the Alert System shall send a Low Battery notification to the Mobile App.

Words and phrases to reference a person or object without specifying who or what it is; for example, words such as “it”, “this”, “he” “she”, “they”, “them”, “other” and “both”. A proper requirement should avoid the use of pronouns or cross-referencing pronouns.

Recommendation: *Repeat nouns in full instead of using pronouns to refer to nouns in other requirements.*

Overall Results

QVscribe Quality Analysis Score Distribution

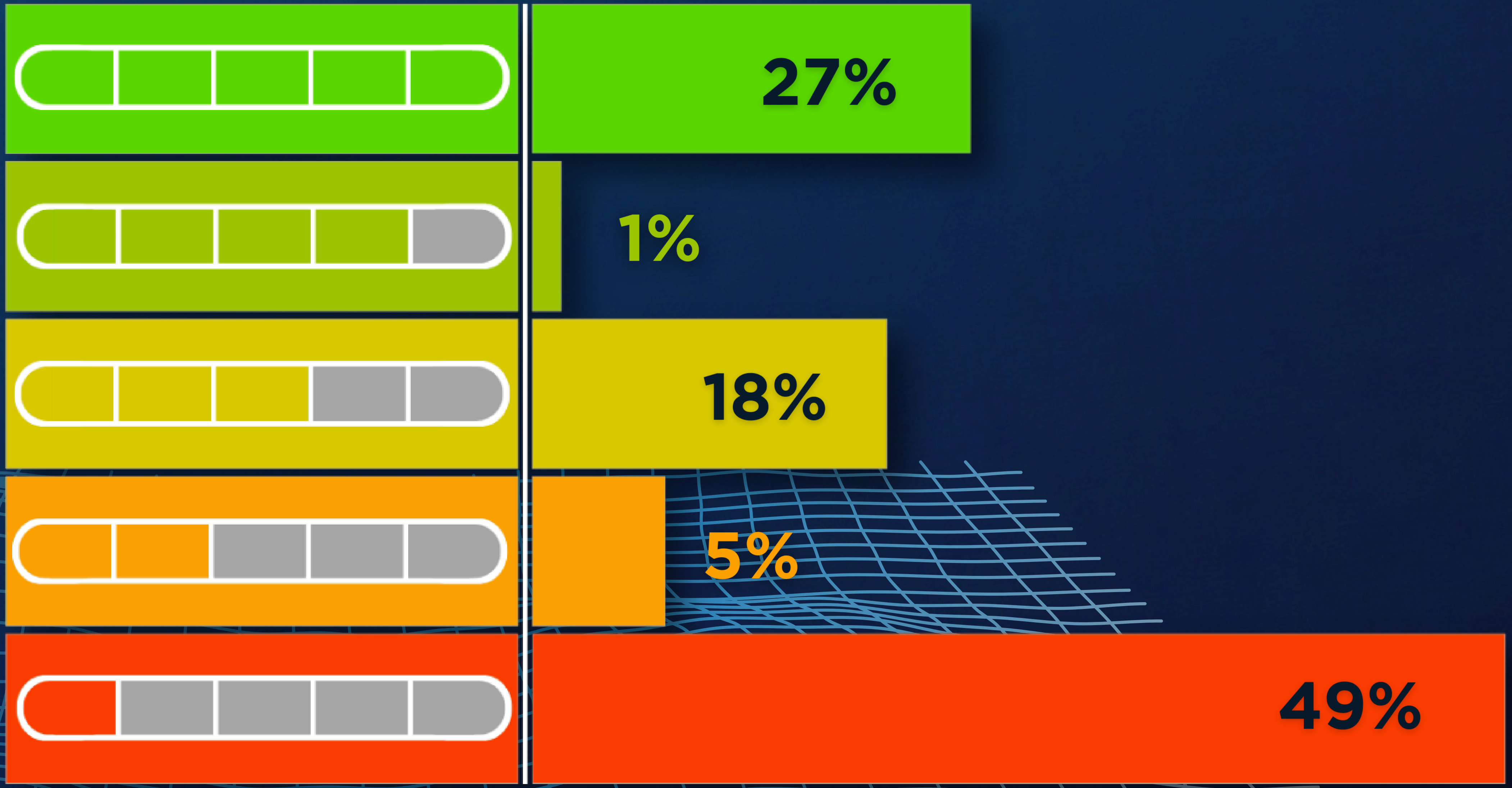
5/5, Very Low Risk: includes clear and unambiguous terminology to express the requirement.

4/5, Low Risk: may include excessive use of continuances, and or, no directives.

3/5, Medium Risk: includes a single instance of a vague, subjective, or weak term, and/or a single negative imperative.

2/5, High Risk: includes multiple instances of vague, subjective, or weak terms, and/or negative imperatives**.**

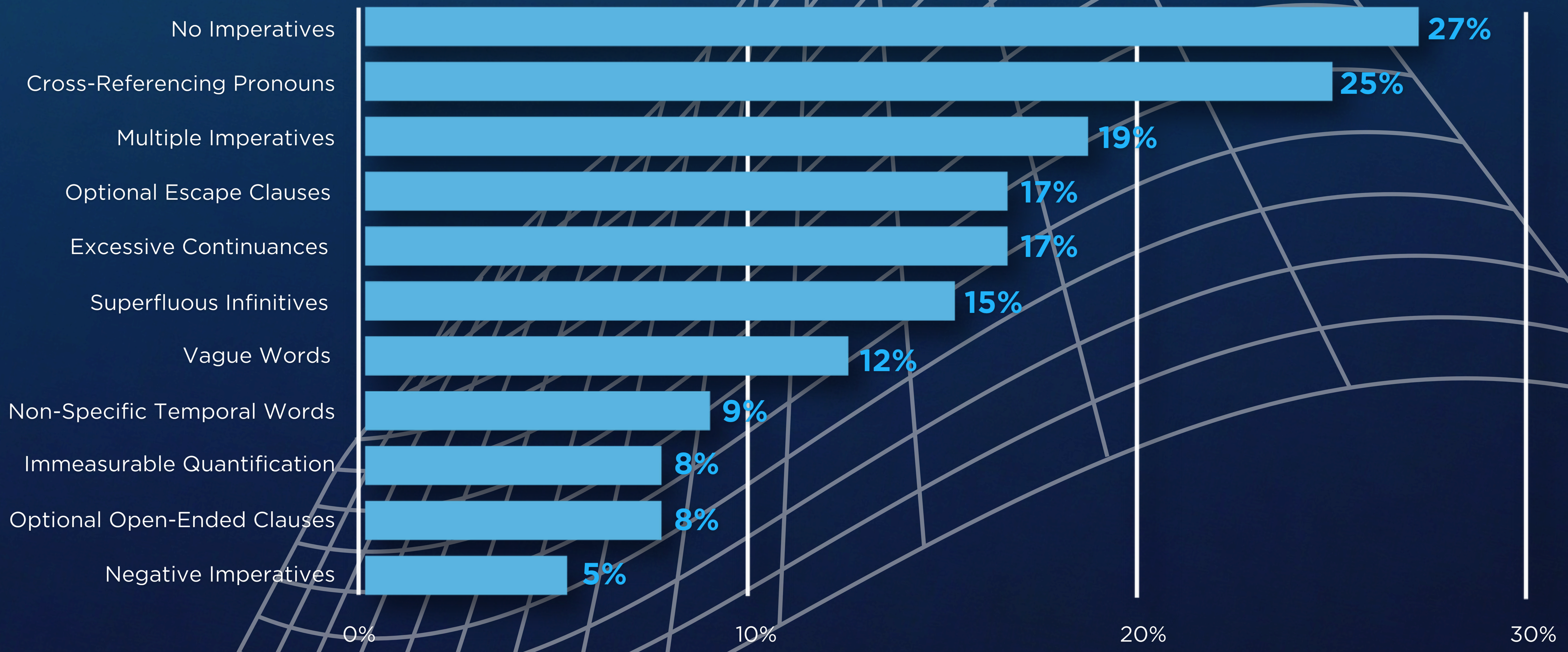
1/5, Very High Risk: includes problems imperatives or more than two instances of problematic language. It is likely that important information will be missed in the execution of the project.



Average of **2/5** QVscribe Quality Score.

Overall Results

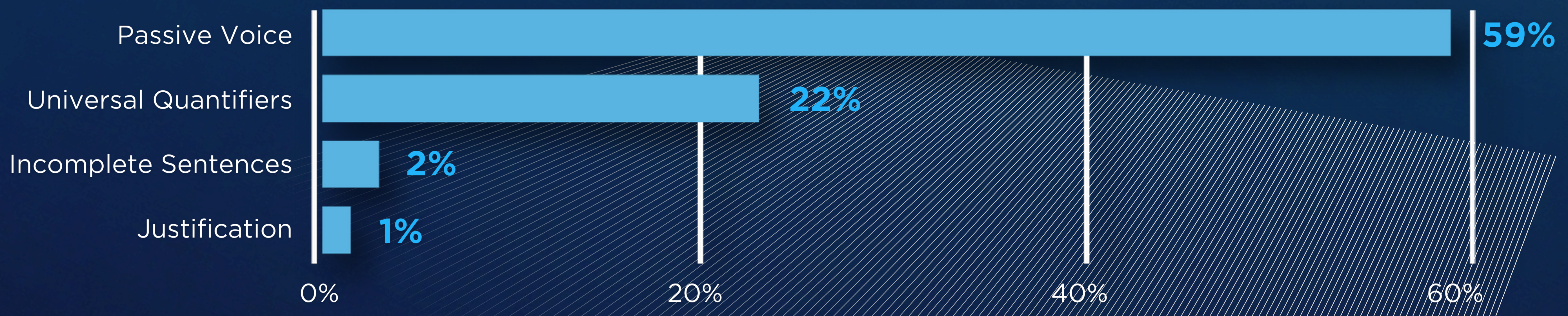
Problem Types (%)



*These issues each have a negative impact on the QVscribe Quality Score.

Overall Results

Quality Warnings (%)



*Quality Warnings do not impact the overall score but do pose potential risks depending on the context of usage.

Overall Results

EARS, Consistency and Similarity

41% average Easy Approach to Requirement Syntax (EARS) conformance.

37,336 potentially conflicting terms.

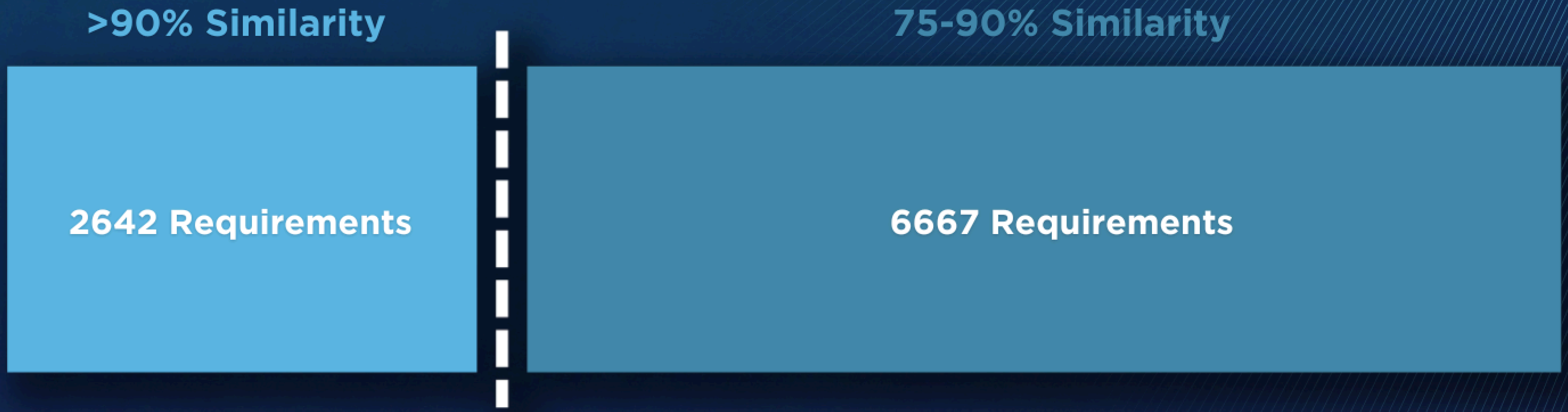
2,642 potential duplications.

Length was the top cause of inconsistent units.

6,667 potential contradictions.

Overall Results

Similarity Analysis



>90% Similarity

2642 Requirements

75-90% Similarity

6667 Requirements

*This includes both duplicate and contradicting requirements.

Overall Results

Duplicate Requirements

Duplicate requirements can confuse stakeholders and developers, leading to misunderstandings about the intended functionality of the system and might inadvertently inflate the scope of the project if not properly identified. This can lead to *scope creep*, where the project gradually expands beyond its original boundaries, resulting in increased costs and timelines.

Clarity is essential for ensuring everyone is on the same page regarding what needs to be built.

Overall Results

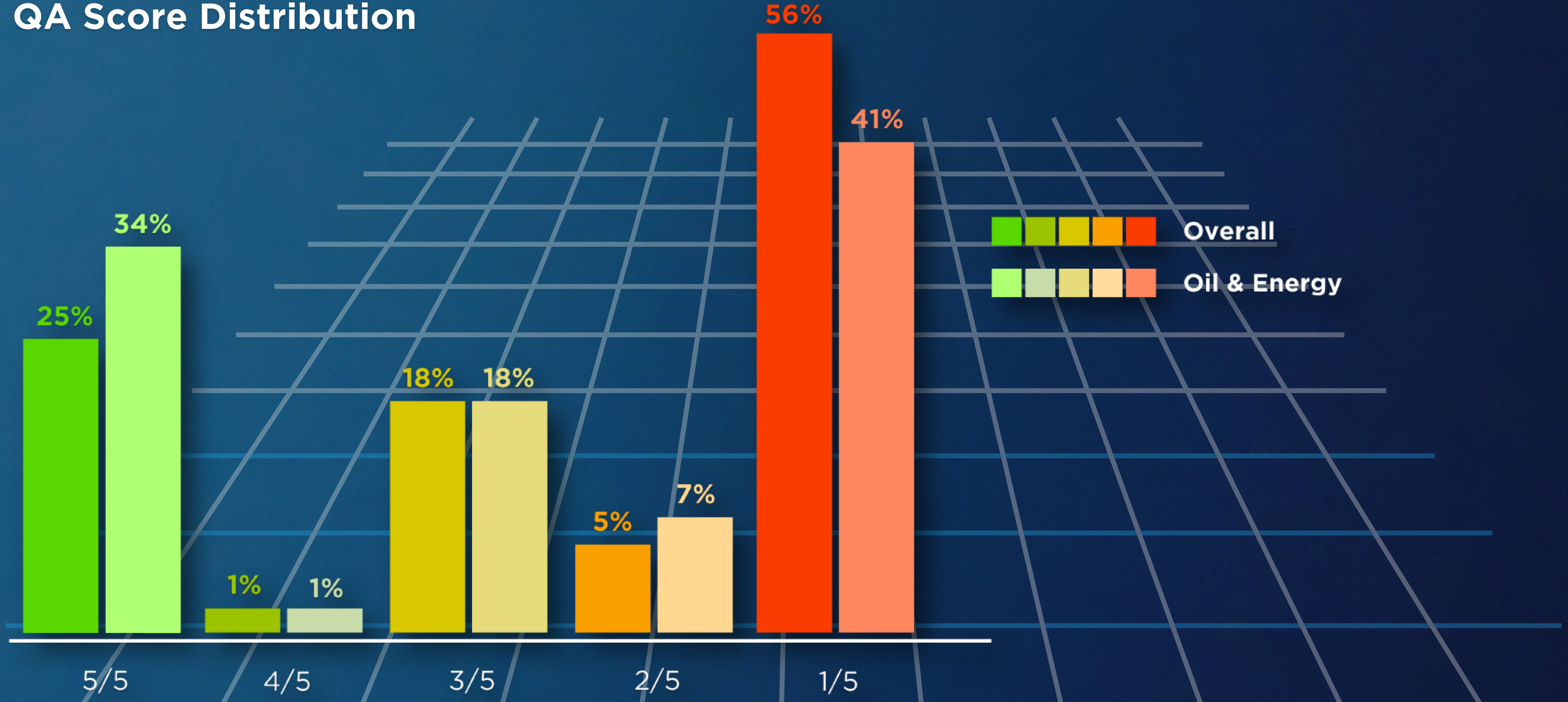
Contradicting Requirements

Contradicting Requirements can waste time and resources during the development process and lead to compromised quality in the final product. When a project has requirements that contradict, there is potential for inconsistency in the implementation of the system, potentially resulting in errors, inefficiencies, or even system failures.

Consistency ensures that the system behaves predictably and reliably across different scenarios.

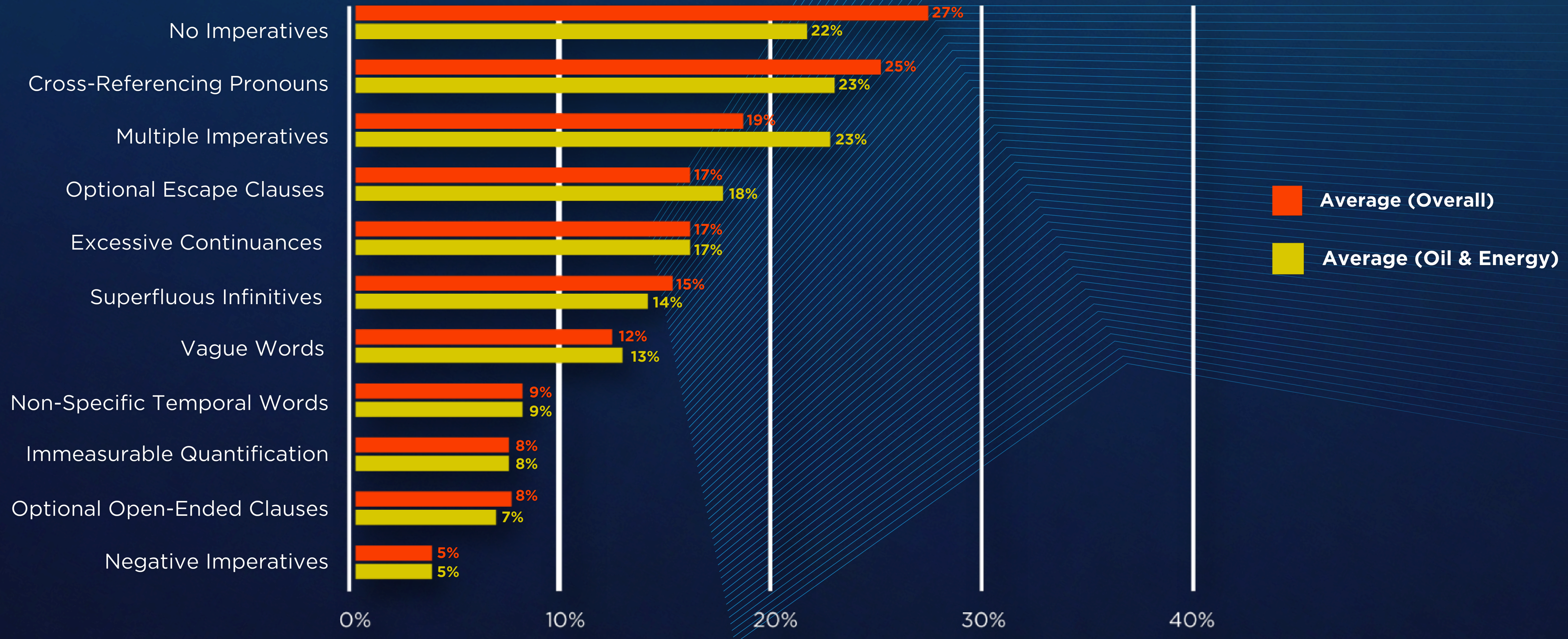
Industry Comparison

QA Score Distribution



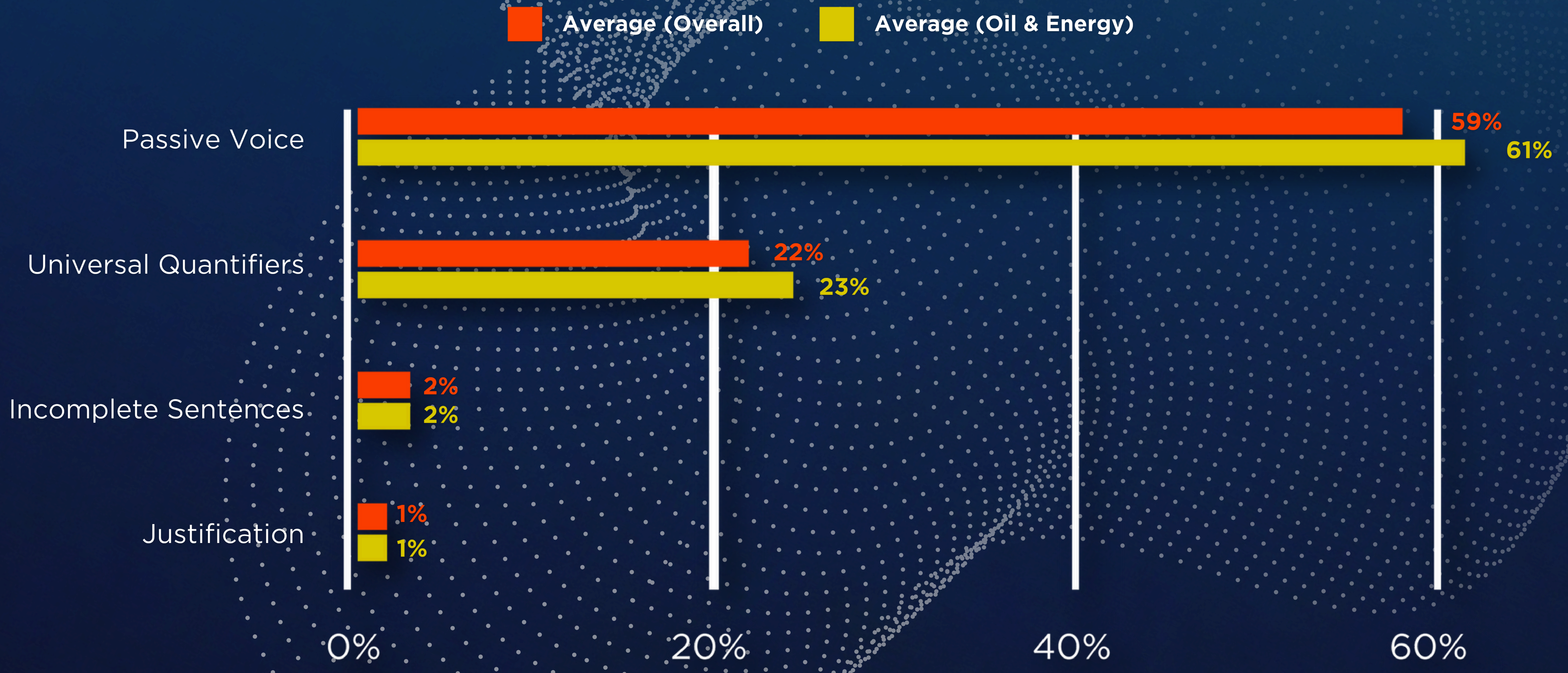
Industry Comparison

Problem Types (%)



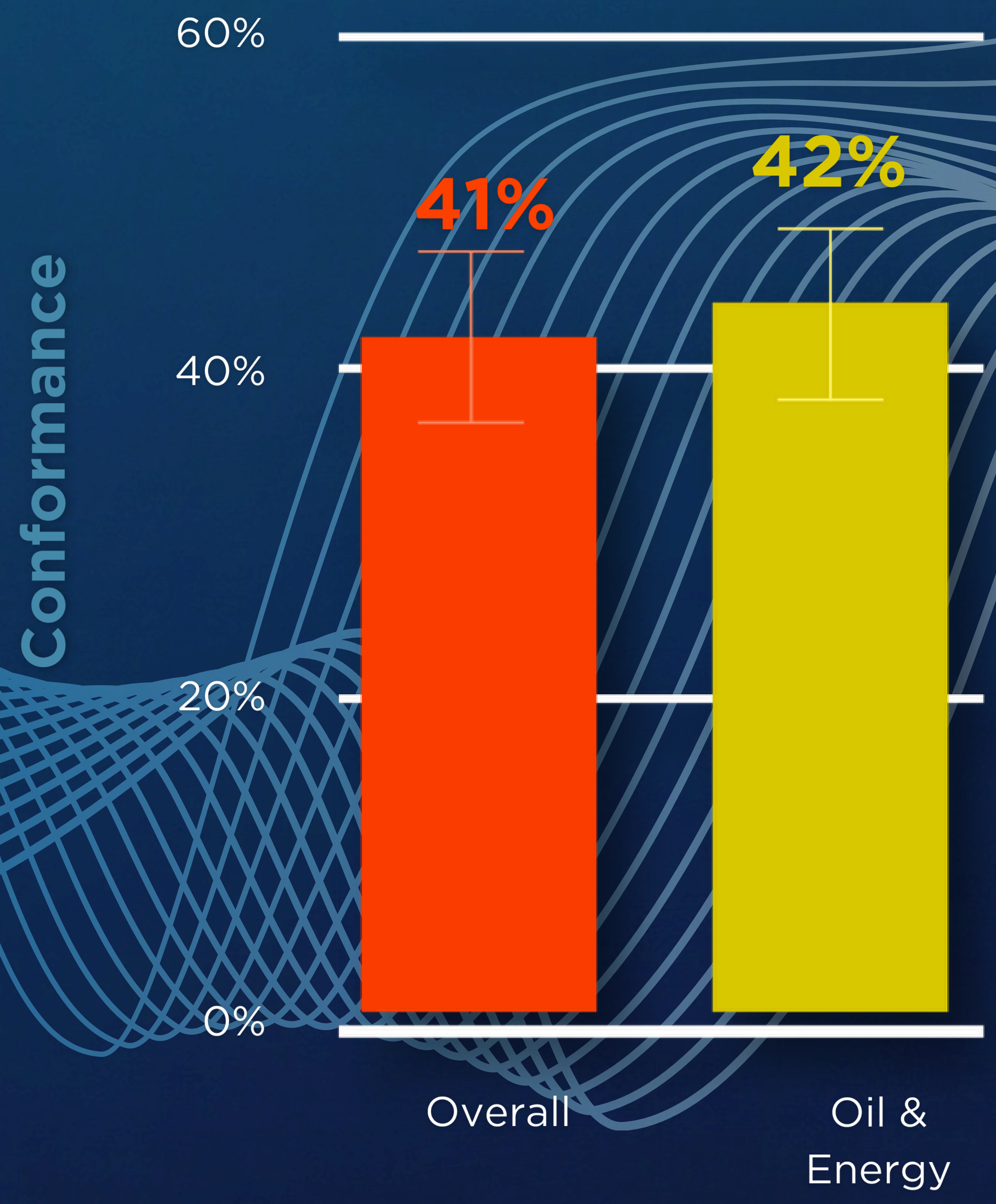
Industry Comparison

Quality Analysis (%)



Industry Comparison

EARS Conformance (%)



The **EARS** patterns force an author to conform to a simple, efficient format. The patterns tear away extraneous words; virtually eliminating the temptation to add extra information, because the patterns do not allow for it. This makes the resulting requirements much clearer and easier to understand, which consequently saves time when reading or implementing said requirement.

Industry Comparison

Problem Words

55 “Problem Words” were identified based on the top three Problem Types for each document. These words were repeated throughout thousands of requirements.

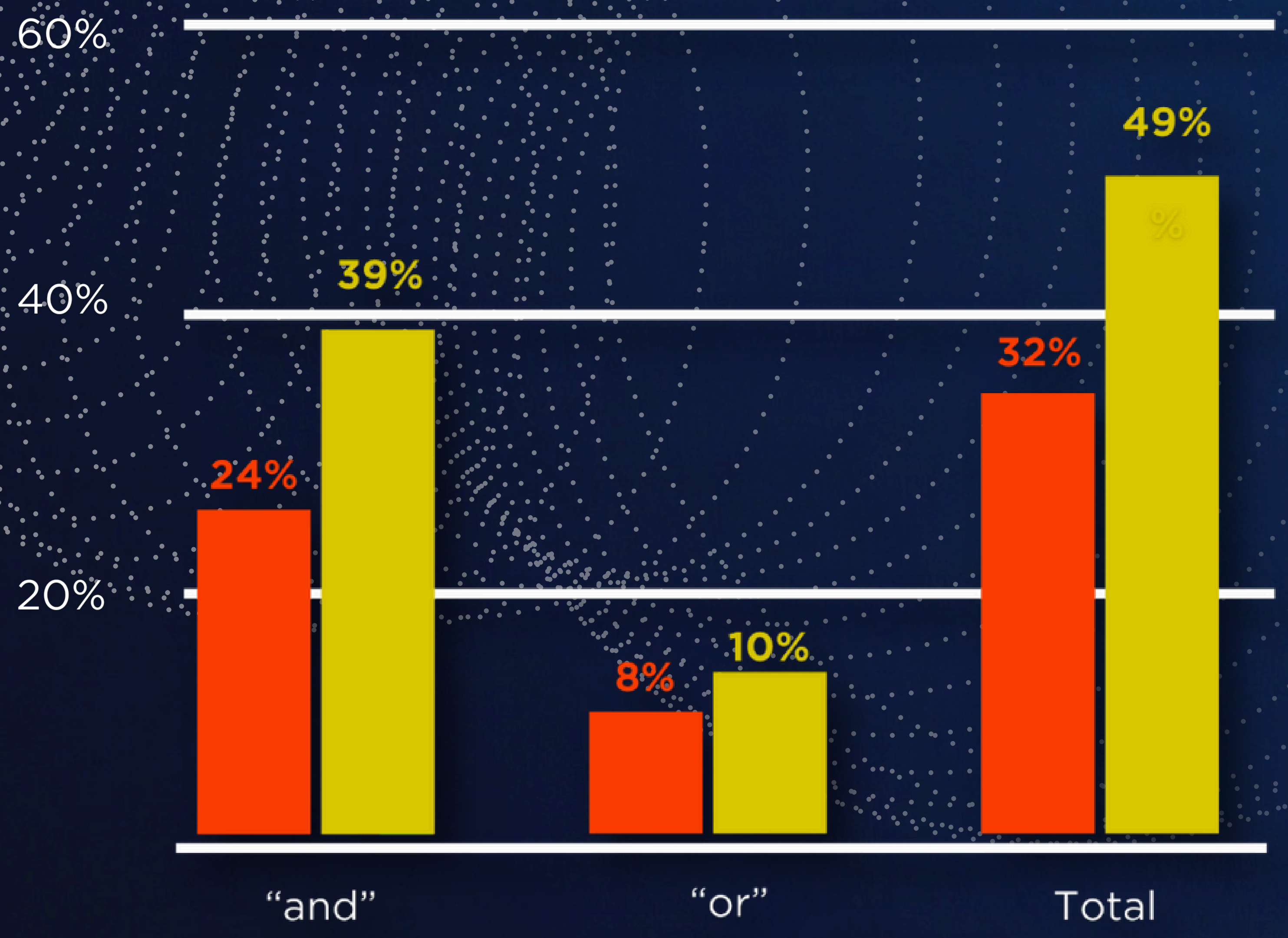
Total number of problem words counted

- Overall: 8033
- Oil and Energy: 2309

Industry Comparison

Excessive Continuances

Excessive Continuances are words or phrases that follow the requirement's imperative and introduce more detail to the specification. A proper requirement avoids excessive use of continuances and combinators (generally not more than two).



This was not included in the results, as "or" and "and" are not problem words in themselves (e.g. they are not BAD), it is the excessive use within a single requirement that is an issue.

**Only instances where excessive continuances was triggered in an individual requirement were counted.*

For a comprehensive understanding of the results, check out our webinar, *Mastering Requirements: A Deep Dive into Automation and Accuracy.*