qvscribe

EARS

Easy Approach to Requirement Syntax

1 Part

Introduction

Unconstrained natural language requirements can be vague, ambiguous, verbose, and confusing. In many cases, these requirements can lead to unexpected interpretations, erroneous implementations, costly scrap and rework, and – in the worst cases – disaster.

The Easy Approach to Requirements Syntax - EARS, helps solve that problem by bringing just enough rigor to the process of writing requirements in natural language. Simplifying the requirements writing process and supporting authors in writing clear and consistent requirements.

EARS is available in QVscribe as templates, and in the following three part series, we will outline how you can optimize the EARS format to enhance and simplify your requirement writing personally, in a team and company wide.

Please note that in certain circumstances using EARS templates is not ideal. Please refer to the following blog on when not to use EARS.

Write Natural Language Requirements that are Clear, Concise, Unambiguous and Testable

EARS templates are a simple and ideal foundation for writing clear and straightforward requirements.

The EARS patterns guide us to conform to a simple, efficient format and remove extraneous words. EARS eliminates the temptation to add extra information because the patterns don't allow it. They make the resulting requirements more articulate, which saves the person reading or implementing the requirement much time and effort.

There are five fundamental EARS patterns. Four of these are used to author requirements for normal conditions, i.e., what we want the system to be and to accomplish. The fifth pattern is used for requirements to mitigate unwanted events or user behavior.

1. Ubiquitous
2. State-Driven
3. Event-Driven
4. Optional Feature
5. Unwanted Behavior

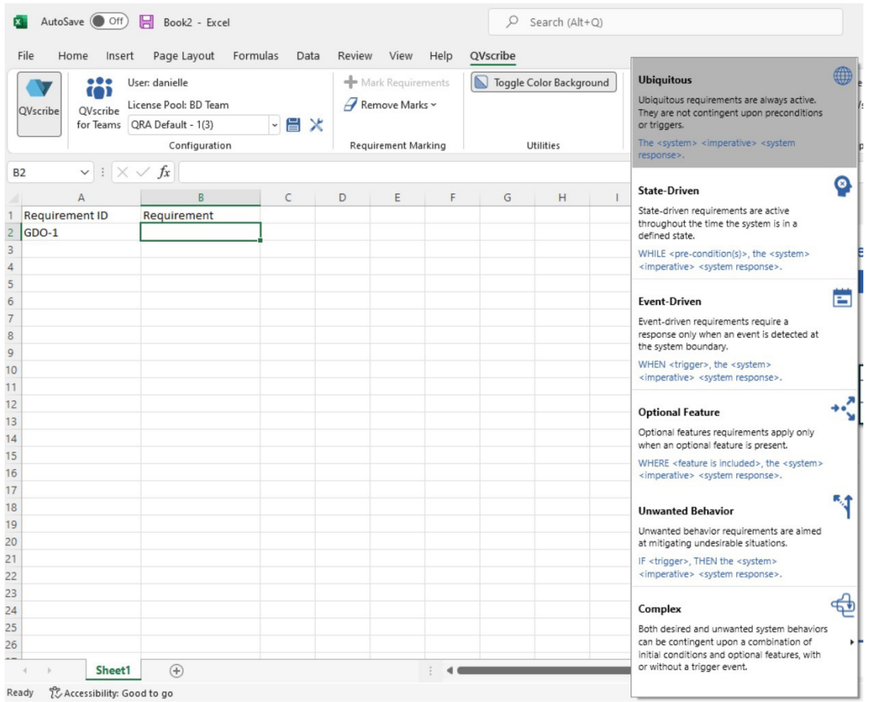
Quick Reference Guide - EARS Templates

Name	Template	Description	Example
Ubiquitous	The <system> <imperative> <system response>.	Requirements that are always active.	The Cruise Control System shall compute the Following Distance in Seconds.
State-Driven	While <pre-conditions(s)>, the <system> <imperative> <system response>.	Requirements that are active during a defined state.	While in Standby Mode, the Cruise Control System shall display the Standby Status on the dashboard.
Event-Driven	When <trigger>, the <system> <imperative> <system response>.	Requirements that require a response only during a specific event.	When the Driver presses the Set Cruise Button, the Cruise Control System shall enter Cruise Mode.
Optional Feature	Where <feature is included>, the <system> <imperative> <system response>.	Requirements that only apply when an optional feature is present.	Where the Pre-Collision System is present, the Cruise Control System shall yield to inputs from the Pre-Collision System.
Unwanted Behavior	If <trigger>, then the <system> <imperative> <system response>.	Requirements that outline situations that are undesirable.	If the Radar fails, then the Cruise Control System shall enter Standby Mode.

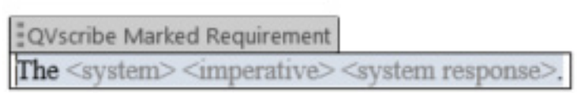
Where you can find these templates within QVscribe is outlined on the following pages.

QVscribe for Word and Excel:

Locate the ‘Place Requirement Template’ button in the QVscribe ribbon at the top of the screen.

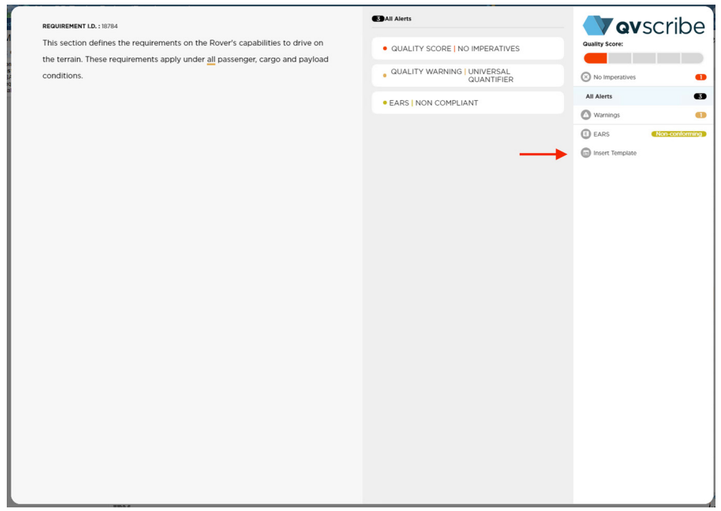


When a template is selected it will appear at the cursor location on the document. See the example below.

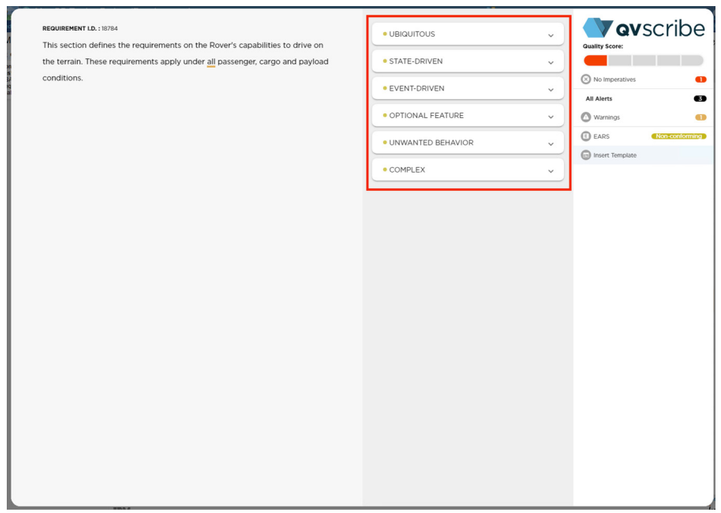


QVscribe for DOORS, Jama, and Polarion:

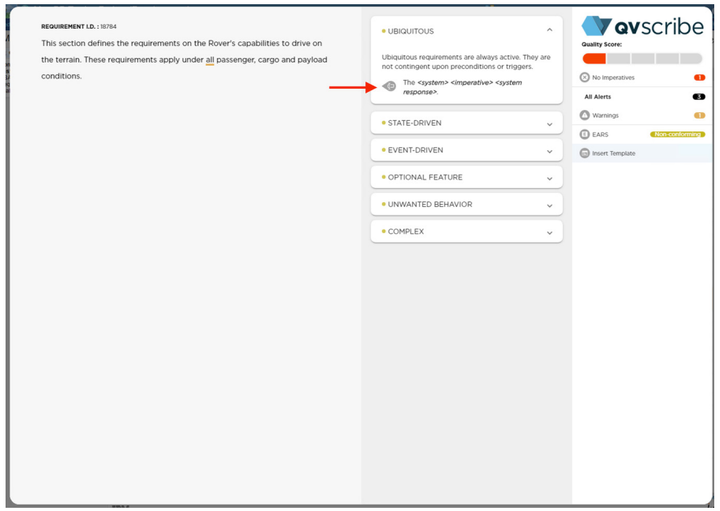
From the ‘QVscribe Authoring View’ for individual requirements, find the insert ‘Template ‘button on the far right.



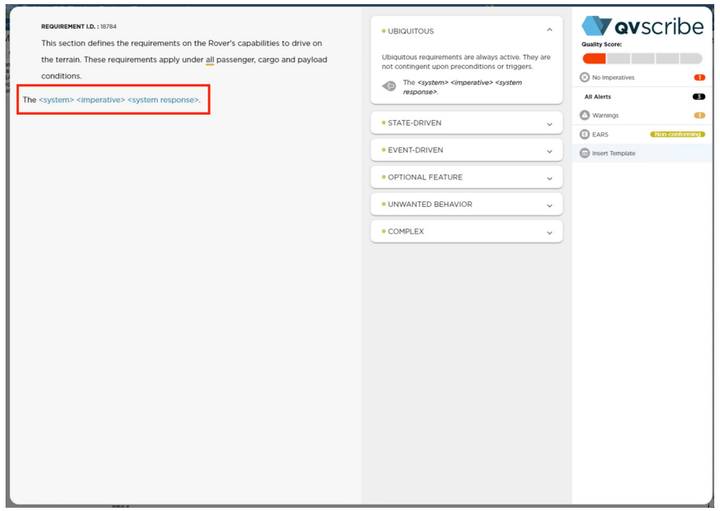
Select the appropriate template from the center column:



Press the gray insert ‘Template’ icon to add the template to your requirement text:



The template appears at the cursor location in the text. Replace the template text to create your EARS-compliant requirement:



Part 1: EARS Templates Series

In Part 1 of the EARS Templates Series, we will focus on the five fundamental EARS patterns. Let’s walk through each template.
For more information, Click here to refer to our EARS training video.

Let’s set the stage.

You are writing a new requirement document for a Garage Door Opener. You will be using EARS templates to write your requirements.

Part 1

Template 1: Ubiquitous

A Ubiquitous requirement is always active and is not contingent upon events, triggers, or preconditions.

The best format for our scenario that provides guidance in creating an always active requirement is the Ubiquitous Requirement Template.

The EARS template for Ubiquitous requirements is:

> *The <system> <imperative> <system response>.*

Ubiquitous Requirements Tip:
Always question Ubiquitous requirements when writing or reviewing them. What at first seems Ubiquitous may be State-Driven. Check carefully that the requirement is indeed true in all states in which the system must operate.

Using this template, let's practice with the information you know and how you can fit it into a requirement.

You want to ensure the Garage Door Opener always displays/shows the status of the Garage Door Opener (active and on, inactive and off).

1. What is the <system>?
The system being specified is what must provide the required response —once any required preconditions and/or trigger events have been detected. The system name appears in the requirement statement immediately before the word shall (or another imperative verb, like must) but after any preconditions and trigger.

In this case, the system is the *Garage Door Opener*.

> *The Garage Door Opener <imperative> <system response>.*

2. What is the <imperative>?
The author uses an imperative verb to connect the subject of the requirement to the desired response. A requirement must contain an imperative to assert that it is mandatory for the subject to take this action. Without an imperative, it can be difficult to discern whether what is being described is actually required.

In this case, the imperative you will use is *shall*.

> *The Garage Door Opener shall <system response>.*

3. What is the <system response>?
The system response – being the desired result that the subject must produce – logically comes at the end of the requirement statement. A requirement may include multiple response elements if they are all caused by the same set of features, preconditions, and triggers.

In this case, you want the system to show the status of the system when it is active or inactive. In other words, you want the system to display the Power Status of the Garage Door Opener.

> *The Garage Door Opener shall display the Power Status.*

You now have a ubiquitous requirement written for your document.

Template 2: State-Driven

A State-Driven requirement is always active throughout the time the system is in a defined state.

The best format for our scenario that provides guidance in creating a requirement that depends on a specific state is the State-Driven Requirement Template.

The EARS template for a State-Driven requirement is:

> *While <pre-conditions(s)>, the <system> <imperative> <system response>.*

Using this template, let's practice the information you know, and apply it to a requirement.

You want to ensure the Garage Door Opener's light is always on while the Garage Door is open.

1. What is/are the<pre-condition(s)>?
The preconditions for a State-Driven requirement are the prerequisites for activating a specific system response. The system needs to be in this state for the system response to occur. These preconditions must exist for the requirement to be applied before anything can happen. This is the reason they appear in the requirement statement before the system response.

In this case, you need the Garage Door to be open in order for the light to turn on.

> *While the Garage Door is open, the <system> <imperative> <system response>.*

2. What is the <system>?
The system being specified is what must provide the required response (once any required preconditions and/or trigger events have been detected). The system name appears in the requirement statement immediately before the word shall (or another imperative verb, like must) but after any preconditions and trigger.

In this case, the system is the *Garage Door Opener*.

> *While the Garage Door is open, the Garage Door Opener <imperative> <system response>.*

3. What is the <imperative>?
The author uses an imperative verb to connect the subject of the requirement to the desired response. A requirement must contain an imperative to assert that it is mandatory for the subject to take this action. Without an imperative, it can be difficult to discern whether what is being described is actually required.

In this case, the imperative you will use is *shall*.

> *While the Garage Door is open, the Garage Door Opener shall <system response>.*

4. What is the <system response>?
The system response – being the desired result that the subject must produce – logically comes at the end of the requirement statement. A requirement may include multiple response elements if they are all caused by the same set of features, preconditions, and triggers.

In this case, you want the system to turn the light on when the Garage Door is open.

> *While the Garage Door is open, the Garage Door Opener shall turn the Light on.*

You now have a State-Driven requirement written for your document.

Template 3: Event-Driven

An Event-Driven requirement requires a response only when an event is detected at the system boundary.

The best format for our scenario that provides guidance in creating a requirement that happens when a specific event takes place is the Event-Driven Requirement Template.

The EARS template for an Event-Driven requirement is:

> *When <trigger>, the <system> <imperative> <system response>.*

Using this template, let's practice the information you know and how you can fit it into a requirement.

You want the Garage Door to open when the button to open is pressed.

1. What is the<trigger>?
A trigger causes something to happen. The trigger specifies events that are desired. A trigger can also specify an undesirable situation or event, we will outline this scenario in the Unwanted Behavior section.

In this case, you need the button to open the Garage opener to be pressed. The button being pressed is the trigger.

> *When the Open Button is pressed, the <system> <imperative> <system response>.*

2. What is the <system>?
The system being specified is what must provide the required response (once any required preconditions and/or trigger events have been detected). The system name appears in the requirement statement immediately before the word shall (or another imperative verb, like must) but after any preconditions and trigger.

In this case, the system is the *Garage Door Opener*.

> *When the Open Button is pressed, the Garage Door Opener <imperative> <system response>.*

3. What is the <imperative>?
The author uses an imperative verb to connect the subject of the requirement to the desired response. A requirement must contain an imperative to assert that it is mandatory for the subject to take this action. Without an imperative, it can be difficult to discern whether what is being described is actually required.

In this case, the imperative you will use is *shall*.

> *When the Open Button is pressed, the Garage Door Opener shall <system response>.*

4. What is the <system response>?
The system response – being the desired result that the subject must produce – logically comes at the end of the requirement statement. A requirement may include multiple response elements if they are all caused by the same set of features, preconditions, and triggers.

In this case, you want the system to open the Garage Door when the open button is pressed.

> *When the Open Button is pressed, then the Garage Door Opener shall open The Garage Door.*

You now have an Event-Driven requirement written for your document.

Template 4: Optional Feature

Optional Feature requirements apply only when an optional feature is present.

The best format for our scenario that provides guidance in creating a requirement that outlines a system response when a specific feature is present is the Optional Feature Requirement Template.

The EARS template for an Optional Feature requirement is:

> *Where <feature is included>, the <system> <imperative> <system response>.*

Using this template, let's practice the information you know and how you can fit it into a requirement.

You want the Garage Door to require a passcode to open if a keypad is installed.

1. What is the <feature is included>?
If a requirement only applies when a specific optional feature is present, that should be made clear at the outset, so the 'Where' phrase always comes first in such requirements.

In this case, the Optional feature is the *keypad*.

> *Where the Keypad is installed, the <system> <imperative> <system response>.*

2. What is the <system>?
The system being specified is what must provide the required response (once any required preconditions and/or trigger events have been detected). The system name appears in the requirement statement immediately before the word shall (or another imperative verb, like must) but after any preconditions and trigger.

In this case, the system is the *Garage Door Opener*.

> *Where the Keypad is installed, the Garage Door Opener <imperative> <system response>.*

3. What is the <imperative>?
The author uses an imperative verb to connect the subject of the requirement to the desired response. A requirement must contain an imperative to assert that it is mandatory for the subject to take this action. Without an imperative, it can be difficult to discern whether what is being described is actually required.

In this case, the imperative you will use is *shall*.

> *Where the Keypad is installed, the Garage Door Opener shall <system response>*

4. What is the <system response>?
The system response – being the desired result that the subject must produce – logically comes at the end of the requirement statement. A requirement may include multiple response elements if they are all caused by the same set of features, preconditions, and triggers.

In this case, you want the system to require a passcode to open the Garage code when the Optional feature is present - the keypad is installed.

> *Where the Keypad is installed, the Garage Door Opener shall require a passcode to open the Garage Door.*

You now have an Optional Feature requirement written for your document.

Template 5: Unwanted Behavior

Unwanted Behavior requirements cover all situations that are undesirable and are often imposed when the system must respond to a trigger under less than optimum conditions.

The best format for our scenario that provides guidance in creating a requirement that outlines a system response to an undesirable situation or event present is the Unwanted Behavior Requirement Template.

Unwanted Behaviour Requirement Tip:
Unwanted Behavior is a major source of omissions in requirements, necessitating costly rework. It is a recommended practice to write unwanted Behavior requirements during your second pass of authoring and after you've written all your normal conditions requirements. On the second pass, examine the 'normal operation' requirements you've written to see if any Unwanted conditions or inputs need to be mitigated. The If/Then keywords provide a useful cue that tells readers this is a requirement for the system to mitigate an unwanted event.

The EARS template for an Unwanted Behavior requirement is:

> *If <trigger>, then the <system> <imperative> <system response>.*

Using this template, let's practice the information you know and how you can fit it into a requirement.

You want the Garage Door to enter Emergency Mode if there is an obstruction blocking the Garage Door.

1. What is the<trigger>?
A trigger causes something to happen. In this case, if we are discussing an Unwanted Behavior, we will use *If/Then* to discuss the trigger, instead of 'When' which describes the desired event.

In this case, the blocker or obstruction to the system is detected would be the trigger.

> *If the Door Sensor detects an obstruction, then the <system> <imperative> <system response>.*

2. What is the <system>?
The system being specified is what must provide the required response (once any required preconditions and/or trigger events have been detected). The system name appears in the requirement statement immediately before the word shall (or another imperative verb, like must) but after any preconditions and trigger.

In this case, the system is the *Garage Door Opener.*

> *If the Door Sensor detects an obstruction, then the Garage Door Opener <imperative> <system response>.*

3. What is the <imperative>?
The author uses an imperative verb to connect the subject of the requirement to the desired response. A requirement must contain an imperative to assert that it is mandatory for the subject to take this action. Without an imperative, it can be difficult to discern whether what is being described is actually required.

In this case, the imperative you will use is *shall*.

> *If the Door Sensor detects an obstruction, then the Garage Door Opener shall <system response>.*

4. What is the <system response>?
The system response – being the desired result that the subject must produce – logically comes at the end of the requirement statement. A requirement may include multiple response elements if they are all caused by the same set of features, preconditions, and triggers.

In this case, you want the system to *enter Emergency Mode* when an obstruction is found.

> *If the Door Sensor detects an obstruction, then the Garage Door Opener shall enter Emergency Mode.*

You now have an Unwanted Behavior requirement written for your document.

This ends Part 1 of our 3 part series on EARS Templates. In the next part, we will outline how you can optimize the EARS format to help with rewriting high risk requirements, and writing new requirements using the Complex templates.

EARS Use Case Series:

Part 1: Write natural language requirements that are clear, concise, unambiguous and testable.

Part 2: Rewriting high risk requirements and using complex EARS templates

Part 3: Learning requirement structure and standardizing requirement writing processes for speed and consistency

QRA

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

qracorp.com