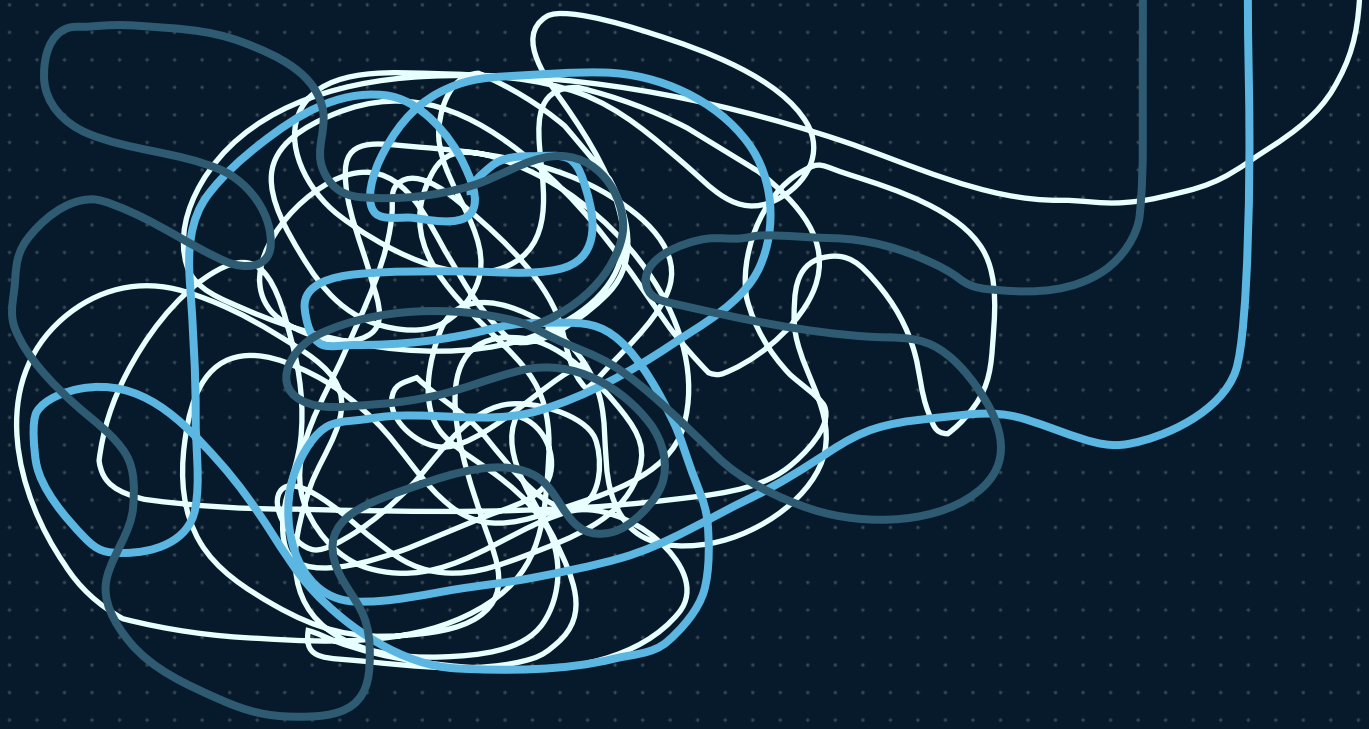
From Chaos to Clarity: Mastering Atomic Requirements in Engineering



QVscribe

Introduction.

At QRA, we champion precision in requirement writing through methods to streamline and enhance the requirements process. We are proponents of writing according to a template, a system outlined in our comprehensive EARS (The Easy Approach to Requirements Syntax) Series - see EARS Templates Parts 1, 2, and 3. We also advocate for writing atomic requirements, a concept outlined in our blog, Atomic Requirements 101: A Comprehensive Guide with Examples.

An atomic requirement is "a natural language statement that completely describes a single system function, feature, need or capability including all information, details, and characteristics."

Simply put, atomic requirements are singular statements that have all the relevant information you need in one sentence.

Incorporating atomic writing aligns with widely endorsed requirement best practices, including the INCOSE (International Council on Systems Engineering) Guide to Writing Requirements. This approach ensures that each requirement stands alone and is clear, precise, and singular. Atomic requirements translate into increased measurability, enhanced testing procedures, improved naming conventions, effortless traceability, streamlined verification processes, and potential for reusability.

If atomic requirements are so widely accepted as a standard in requirement writing - why isn't this practice consistently used across requirement engineering industries, and why are we here at QRA talking about it so much?

We frequently engage in discussions with requirement writers and teams who express reservations about the necessity of writing atomic requirements. It's not uncommon for organizations to be comfortable with their existing processes and view the transition to singular writing as a potential challenge to avoid or have the capacity to tackle. We understand these concerns, and as highlighted in our blog, it's true that crafting atomic requirements does take more effort to write. However, the investment is justified when you recognize the advantages, such as risk reduction and enhanced clarity, comprehension, analysis, and implementation.

Let's just be honest with each other. It's not fun to completely revamp the way you have been doing things as an individual, a team, an organization, or an industry.

In this use case, we will investigate how to write atomically, the issues with non-atomic requirements, and dive into some examples of atomic and non-atomic requirements.

Table 1: Steps for Writing Atomic Requirements

Step	Action
Identify the Function, Action, or Idea	You need to understand what you will be describing in your requirement. Confirm the specific functionality, action, or idea that you want to define in an atomic requirement.
Use Consistent and Precise Natural Language	Write your requirement using clear natural language. Avoid ambiguous, vague terms, jargon, or overly technical language which may lead to misunderstandings or confusion.
One Idea	Be sure that each requirement covers only one idea or concept. Avoid combining multiple ideas or functionalities into a single requirement. If you notice a requirement becoming complex or involving multiple ideas or actions, consider splitting them into separate requirements.
Use a Template or Start With an Action	Use a requirement template, such as EARS templates to help structure your requirement. Make sure your requirement has a strong action verb that describes the desired behavior. This makes the requirement actionable and highlights the expected action or outcome.
Make It Measurable	Provide specific details about what the system shall do or how it shall behave. Use measurable criteria and avoid vague language when defining success or completion of the requirement.
Add Necessary Context	Define the scope and context in which the functionality shall occur. This could involve describing the user, system state, triggering event, or precondition that initiates the action.
Focus On 'What' Rather Than the 'How'	Avoid going into technical details of implementation. Requirements should describe the desired outcome, not the specific technical approach to achieving it.
Use QVscribe, Review, and Validate	Use QVscribe to analyze your requirements to ensure they are clear, concise, and free of risk. Next, have your QVscribe approved atomic requirements reviewed by all stakeholders. This ensures that the requirements are comprehensive, accurate, relevant, and aligned with expectations.
Use a Requirement Management (RM) Tool: Traceability, Prioritization and Organization	We recommend using an RM tool to help maintain a traceable record of requirements. An RM tool will also help logically organize your requirements and prioritize them based on their importance and dependencies. This aids in project planning and development scheduling.

How to Write Atomically.

First things first, writing atomic requirements is an essential aspect of developing clear, precise, and manageable requirements, that turn into clear, precise, and manageable projects. For a deeper understanding of why adopting atomic writing is imperative, please refer to our blog, Atomic Requirements 101: A Comprehensive Guide with Examples. In the blog, we go into detail about the benefits you can expect in your projects from writing atomically. Our primary focus in this Use Case is to equip you with the know-how to write atomically, so you can easily implement these changes into your work.

By following these steps in (Table 1: Steps for Writing Atomic Requirements), you can be on your way to writing consistent atomic requirements from the outset. Now let's examine the problem with non-atomic requirements and why authors and teams frequently resort to using a paragraph, bullet point, or grouped requirements.

Why are Non-Atomic Requirements so Prevalent?

For the sake of convenience, it's only natural for us to gravitate towards grouping ideas, details, and information. When authoring requirements this same instinct is not the most effective approach.

In a different situation, we might find ourselves without all the necessary information to adequately write the requirements. This can occur for various reasons, such as lack of clarity, miscommunications, or incomplete information and knowledge of the project.

These two scenarios are the most frequent explanations requirements authors and teams give to why they struggle with writing atomically.

Let's dig further into the reasons behind the creation of non-atomic requirements.

- **Pressured or Short Timelines:** In fast-paced projects and environments, there is often pressure to complete your work quickly. This can force requirement authors to rush through the requirement writing process, leading to shortcuts and the inclusion of multiple ideas or actions in a single requirement.

- **Assumptions:** Requirement authors may assume that certain functionalities naturally go together, leading to the inclusion of multiple ideas or actions in a single requirement without considering the need for granularity.

- **Vague Requests:** Details or information given to the requirement authors might be vague or imprecise. The requirement author might inadvertently combine multiple ideas or actions in an attempt to clearly capture the information they were provided.

- **Implementation Focused:** Requirement authors may write non-atomic requirements if the focus shifts from the 'what' to the 'how'. The focus of the requirement should always describe what they want the requirement to achieve, not how they want it to achieve it.

- **Complexity:** Some functionalities inherently involve many steps, preconditions, or interactions. Requirement authors may perceive all steps, preconditions, or interactions as one idea, leading to non-atomic requirements.

- **Incomplete Analysis:** If the requirement-gathering process is rushed or incomplete, requirement authors may not fully understand the need for breaking down functionalities into atomic requirements. As a result, they might combine multiple ideas or actions into a single requirement.

- **Ease or Simplicity:** Requirement teams and authors often prefer to express information and therefore requirements with as few words or repetitions as possible. They believe that combining functionalities into one statement is simpler. However, this can lead to the grouping of requirements and eventually to ambiguity and complexity during implementation.

- **Inexperience:** Inexperienced requirement authors may struggle with breaking down complex functionalities, leading to the inclusion of multiple ideas or actions in a single requirement.

- **Lack of Awareness:** Requirement teams, organizations, or industries may not be aware of the benefits of writing atomic requirements or understand the consequences of not doing so.

- **Poor Communication:** When communication between requirements authors, teams, and stakeholders is not clear or effective, misunderstandings can result in non-atomic requirements.

- **Status Quo:** In the past, requirement teams used Word documents exclusively for authoring and shared work as hard copies or PDF files. It was best practice and widely accepted to write in paragraph form. As project and product complexity increases, organizations have adopted new technology to advance their requirement process, yet old authoring habits remain. It is hard to implement new authoring techniques when older methods have been the standard for completing projects.

If you're involved in writing requirements, at least one of the issues mentioned may impact your projects' requirements. It's crucial to raise awareness among yourself, your team, and stakeholders about the advantages of adopting atomic requirements. By conducting a comprehensive analysis, transparent communication, and implementing a consistent review process, you can mitigate and even avoid these problems. Consider running workshops, collaborating among all stakeholders, training on requirement templates, implementing an RM tool, and running all your documents through QVscribe to help create more accurate and atomic requirement statements.

What Issues can Arise from Writing Non-Atomic Requirements?

Like most change management, altering the legacy mentality within an organization presents its challenge. We like to call it 'This is the way we do it.' Teams understand the benefits and the supporting information around the impacts of writing atomic requirements but aren't always ready to implement the changes needed. Change can be hard, especially when it comes to perceived extra effort, and the need for a team or organization to adjust or re-vamp their working style.

We get it, but as a requirement-focused company, we consider it a responsibility of ours that all requirement industries understand the negative impact of non-atomic requirements on their projects.

Writing non-atomic requirements can lead to many issues with project clarity, communication, testing, implementation, and the overall success of the projects.

Here are some examples of issues that impact projects that do not write atomically:

- **High Risk of Errors:** Complex, non-atomic requirements can increase the likelihood of errors in development, testing, and deployment, which may require more time, resources and cost to rectify.

- **Difficulty in Testing:** Testing becomes complex when multiple ideas or actions are intertwined in a single requirement. Test cases may become convoluted, making it harder to ensure comprehensive coverage.

- **Scope Creep:** Non-atomic requirements can expand in scope as stakeholders add more details or functionalities within a single requirement. This can lead to project scope creep and missed deadlines.

- **Inaccurate Timelines and Planning:** Without singular and precise requirements, estimating the effort required to implement requirements becomes challenging. This can lead to inaccurate project planning and scheduling.

- **Vagueness and Ambiguity:** Non-atomic requirements may be vague or unclear, leading to misunderstandings among stakeholders. This can result in misaligned expectations and incorrect implementations.

- **Incomplete Implementations:** When a requirement combines multiple ideas or actions, developers might focus on one aspect and overlook others. This can lead to incomplete or partial implementations that don't meet the intended scope.

- **Poor Traceability:** Non-atomic requirements make it difficult to track individual functionalities and their associated changes. This impacts traceability and the ability to manage requirement modifications.

- **Impacts on Maintenance and Reusability:** Non-atomic requirements can lead to many interdependencies among systems and/or software, making maintenance, future enhancements, or reusability of projects and requirements more complex and error-prone.

- **Poor Communication:** Non-atomic requirements can result in miscommunication between stakeholders, leading to a lack of shared understanding of desired outcomes.

- **Risk to Allocation or Change Implementation:** When functionalities are combined, different parts of the system or software may not be allocated properly or accurately. This can impact implementation and can limit the flexibility to implement changes incrementally or swap out individual components.

To avoid these impactful issues, it's vital that teams learn to write atomically and practice doing so. Teams and authors can do this by breaking down ideas or actions into singular, specific requirement statements.

Examples.

Let's take a look at some examples so you can better understand how you can author atomically, or modify your non-atomic requirements. For these examples, we will reference the requirements for a Smart Doorbell.

Example: Atomic Requirement

Let's look at an example of a clear and simple atomic requirement. In this example, we want the Smart Doorbell light to dim and adjust at dusk.

While in Night Mode, the Control System shall set the Status Indicator Light Brightness to 30%.

In this example, we have clearly described one action/idea in a singular requirement statement.

Example: Non-Atomic to Atomic Requirement

Here is an example of a non-atomic or compound requirement, where multiple functionalities have been combined into a single requirement statement.

The Motion Sensor shall detect movement at distances between 5cm and 5m, and within a 150° horizontal range or a 60° vertical range.

This requirement combines three actions that the Motion Sensor must do into a single statement. This can lead to issues with implementation, testing, and allocation among other things.

Let's rewrite these three actions into 3 atomic requirement statements.

The Motion Sensor shall detect movement at distances between 5cm and 5m.

The Motion Sensor shall detect movement at a 150° horizontal sensing angle.

The Motion Sensor shall detect movement at a 60° vertical sensing angle.

Examples: Common Non-Atomic Requirements

Here are some common examples of non-atomic requirements that we see often among our customers.

Example 1:

The solution shall be able to incorporate multiple user databases & shall be able to cross reference databases by name and/or phone number (system or group administrator access only).

Example 2:

If any component of the system fails (i.e. controller, radar, etc.) then the system will disengage immediately.

These requirements demonstrate how authors often group together ideas and actions for ease and simplicity but actually end up creating requirements that are complex, hard to understand, and difficult to implement or test.

Here is how we would recommend re-writing the two examples we provided.

Example 1:

The solution shall be able to incorporate multiple user databases & shall be able to cross reference databases by name and/or phone number (system or group administrator access only).

This requirement has a fairly simple resolution. We often see authors put two or more requirements together if they are often thought of together and grouped together. This is how we naturally speak and write in day-to-day contexts, so it would make sense we would bring this to our requirement authoring as well.

These requirements should be separated into their own singular statements. If you are concerned with the requirements being separated or you want them grouped closely together, you can still achieve ease and simplicity by giving them linked requirement ID numbers.

Here is how we would recommend rewriting the non-atomic requirements.

SOL 212.1 The solution shall be able to incorporate multiple user databases.
SOL 212.2 The solution shall be able to cross reference databases by name and/or phone number (system or group administrator access only).

Example 2:

If any component of the system fails (i.e. controller, radar, etc.) then the system will disengage immediately.

There are quite a few problems with this requirement. We won't go into all the problem words, but if you would like to dive deeper into this example, please refer to our EARS Templates Part 2.

When it comes to the singularity of this requirement, the issue is that we are attempting to account for many components, we are also not being specific about how many components there are with the words **any** and **etc.** Do we really mean **any** component? How do we account for all components included in etc.? Are there 3 components, 10 components, or 100 components? Will all components be accounted for without additional expense or wasted time? Is the scope of all components clearly understood by the stakeholders?

In this case, each component should be its own singular requirement.

Here is how we would recommend rewriting the non-atomic requirements.

If the controller fails, then the Cruise Control system shall disengage within 0.2 seconds.
If the radar fails, then the Cruise Control system shall disengage within 0.2 seconds.
If the x fails, then the Cruise Control system shall disengage within 0.2 seconds.
If the xx fails, then the Cruise Control system shall disengage within 0.2 seconds.

Use these examples to help support your atomic requirement writing and to avoid the common pitfalls when it comes to capturing requirements into singular statements.

Conclusion.

While authoring your requirements, it's vital to go beyond the content, objectives, and purpose. Ensure your requirements are clear, precise, and manageable for all readers and stakeholders. We recommend using EARS templates and regularly reviewing your requirements using QVscribe. By having a clear process, you can quickly and efficiently refine your writing and correct any non-singular issues that may arise.

This guide is designed to assist you in integrating atomic requirement writing into your work, team, organization, or even industry.

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

qracorp.com