Characteristics of a Good Requirement

Bron: Requirements Management Using IBM® Rational® RequisitePro® by Peter Zielczynski, 2008.

Good requirements should have the following characteristics:

- Atomic
- Clear (concise, terse, simple, precise)
- Correct
- Implementation-free (abstract)
- Independent
- Feasible (realistic, possible)
- Necessary
- Testable (verifiable)
- Traceable
- Unambiguous
- Understandable

Besides these criteria for individual requirements, three criteria apply to the set of requirements. The set should be

- Complete
- Consistent
- Nonredundant

The sample project used in this document is an online travel agency. You're probably familiar with this type of application because variations of it can be found on several websites. The project is complex enough to show possible relationships between various requirements types, but it is small enough to be easily understood. Most of the examples in this chapter are related to this project.

Let's discuss each of the criteria of a good requirement and show some examples.

Atomic

The requirement should contain a single traceable element:

REQ1 The system shall provide the opportunity to book the flight, purchase a ticket, reserve a hotel room, reserve a car, and provide information about attractions.

This requirement combines five atomic requirements, which makes traceability very difficult. Sentences including the words "and" or "but" should be reviewed to see if they can be broken into atomic requirements.

Clear (Concise, Terse, Simple, Precise)

Requirements should not contain unnecessary verbiage or information. They should be stated clearly and simply:

REQ1 Sometimes the user will enter Airport Code, which the system will understand, but sometimes the closest city may replace it, so the user does not need to know what the airport code is, and it will still be understood by the system.

This sentence may be replaced by a simpler one:

REQ1 The system shall identify the airport based on either an Airport Code or a City Name.

Correct

If a requirement contains facts, these facts should be true:

REQ1 Car rental prices shall show all applicable taxes (including 6% state tax).

The tax depends on the state, so the provided 6% figure is incorrect.

Implementation-free (Abstract)

Requirements should not contain unnecessary design and implementation information:

REQ1 Content information shall be stored in a text file.

How the information is stored is transparent to the user and should be the designer's or architect's decision.

Independent

To understand the requirement, there should not be a need to know any other requirement:

REQ1 The list of available flights shall include flight numbers, departure time, and arrival time for every leg of a flight.

REQ2 It should be sorted by price.

The word "It" in the second sentence refers to the previous requirement. However, if the order of the requirements changes, this requirement will not be understandable.

Feasible (Realistic, Possible)

The requirement should be doable within existing constraints such as time, money, and available resources:

REQ1 The system shall have a natural language interface that will understand commands given in English language.

This requirement may be not feasible within a short span of development time.

Necessary

A requirement is unnecessary if

None of the stakeholders needs the requirement.

or

Removing the requirement will not affect the system.

An example of a requirement that is not needed by a stakeholder is a requirement that is added by developers and designers because they assume that users or customers want it. For example, the fact that a developer thinks that users would like a feature that displays a map of the airport and he knows how to implement it is not a valid reason to add this requirement.

An example of a requirement that can be removed because it does not provide any new information might look like the following:

REQ1 All requirements specified in the Vision document shall be implemented and tested.

Testable (Verifiable)

Testers should be able to verify whether the requirement is implemented correctly. The test should either pass or fail. To be testable, requirements should be clear, precise, and unambiguous. Some words can make a requirement untestable:

- Some adjectives: robust, safe, accurate, effective, efficient, expandable, flexible, maintainable, reliable, user-friendly, adequate
- Some adverbs and adverbial phrases: quickly, safely, in a timely manner
- Nonspecific words or acronyms: etc., and/or, TBD

Such a requirement might look something like this:

REQ1 The search facility should allow the user to find a reservation based on Last Name, Date, etc.

In this requirement, all search criteria should be explicitly listed. The designer and developer cannot guess what the user means by “etc.”

Other problems can be introduced by ambiguous words or phrasing:

- Modifying phrases: as appropriate, as required, if necessary, shall be considered
- Vague words: manage, handle
- Passive voice: the subject of the sentence receives the action of the verb rather than performing it

REQ1 The airport code shall be entered by the user.

REQ2 The airport code shall be entered.

The first example shows a classic example of passive voice. In active voice it would read “The user shall enter the airport code.” As the second example shows, another result of the use of passive voice is that the agent performing the action is sometimes omitted. Who should enter this code—the system or the user?

Indefinite pronouns: few, many, most, much, several, any, anybody, anything, some, somebody, someone, etc.

REQ1 The system shall resist concurrent usage by many users.

What number should be considered “many”—10, 100, 1,000?

Traceable

Each requirement should be uniquely identified and easily traceable through to lower level requirements, design, and testing.

Unambiguous

There should be only one way to interpret the requirement. Sometimes ambiguity is introduced by undefined acronyms:

REQ1 The system shall be implemented using ASP.

Does ASP mean Active Server Pages or Application Service Provider? To fix this, we can mention a full name and provide an acronym in parentheses:

REQ1 The system shall be implemented using Active Server Pages (ASP).

Here’s another example:

REQ1 The system shall not accept passwords longer than 15 characters.

It is not clear what the system is supposed to do:

- The system shall not let the user enter more than 15 characters.
- The system shall truncate the entered string to 15 characters.
- The system shall display an error message if the user enters more than 15 characters.

The corrected requirement reflects the clarification:

REQ1 The system shall not accept passwords longer than 15 characters. If the user enters more than 15 characters while choosing the password, an error message shall ask the user to correct it.

Some ambiguity may be introduced through the placement of a certain word:

REQ1 On the “Stored Flight” screen, the user can only view one record.

Does this mean that the user can “only view,” not delete or update, or does it mean that the user can view only one record, not two or three?

One way to fix the problem is to rewrite the requirement from the system’s point of view:

REQ1 On the “Stored Flight” screen, the system shall display only one flight.

Understandable

Requirements should be grammatically correct and written in a consistent style. Standard conventions should be used. The word “shall” should be used instead of “will,” “must,” or “may.”

Complete

A requirement should be specified for all conditions that can occur:

REQ1 A destination country does not need to be displayed for flights within the U.S.

REQ2 For overseas flights, the system shall display a destination country.

What about flights to Canada and Mexico? They are neither “within the U.S.” nor“overseas.”

All applicable requirements should be specified. This is the toughest condition to be checked. There is really no way to be sure that all the requirements are captured and that one week before the production date one of the stakeholders won’t say, “I forgot to mention that I need one more feature in the application.”

Consistent

There should not be any conflicts between the requirements. Conflicts may be direct or indirect. Direct conflicts occur when, in the same situation, different behavior is expected:

REQ1 Dates shall be displayed in the mm/dd/yyyy format.

REQ2 Dates shall be displayed in the dd/mm/yyyy format.

Sometimes it is possible to resolve the conflict by analyzing the conditions under which the requirement takes place. For example, if REQ1 was submitted by an American user and REQ2 by a French user, the preceding requirements may be rewritten as follows:

REQ1 For users in the U.S., dates shall be displayed in the mm/dd/yyyy format.

REQ2 For users in France, dates shall be displayed in the dd/mm/yyyy format.

This can eventually lead to the following requirement:

REQ3 Dates shall be displayed based on the format defined in the user’s web browser.

Another example of a direct conflict can be seen in these two requirements:

REQ1 Payment by PayPal shall be available.

REQ2 Only credit card payments shall be accepted.

In this case the conflict cannot be resolved by adding conditions, so one of the requirements should be changed or removed.

Indirect conflict occurs when requirements do not describe the same functionality, but it is not possible to fulfill both requirements at the same time:

REQ1 System should have a natural language interface.

REQ2 System shall be developed in three months.

Some requirements do not conflict, but they use inconsistent terminology:

REQ1 For outbound and inbound flights, the user shall be able to compare flight prices from other, nearby airports.

REQ2 The outbound and return flights shall be sorted by the smallest number of stops.

To describe the same concept, in the first requirement the term “inbound flights” is used, and in the second requirement the term “return flights” is used. The usage should be consistent.

Nonredundant

Each requirement should be expressed only once and should not overlap with another requirement:

REQ1 A calendar shall be available to help with entering the flight date.

REQ2 The system shall display a pop-up calendar when entering any date.

The first requirement (related to only the flight date) is a subset of the second one (related to any date entered by the user).