Coding scheme requirements

Table 1 shows an overview of the coding scheme for extracting requirements (or requirements-relevant information) from issue tracking systems. The scheme differentiates two characteristics on which requirements can be grouped, namely the type of requirement and its granularity. In the scheme, we distinguish the type of requirements that can be found in projects into two main groups: functional requirements and non-functional requirements. Functional requirements have two subcategories: user-oriented functional requirements and system-oriented functional requirements. In addition to grouping the type of requirement, the scheme recognizes that requirements can be distinguished in terms of granularity level. The scheme includes three granularity levels: low-level, medium-level and high-level requirements. Besides requirements, other additional information related to requirements may be present in issue tracking systems, such as motivations for the requirements and references between requirements. In this scheme, we currently consider only the additional information related to a motivation. Each category is explained in more detail below.

Characteristic	**Category**	**Description**
Requirement type	User-oriented functional	Functionality directly experienced by the user.
	System-oriented functional	Functionality that the system will implement but that is not directly experienced by the user.
	Non-functional	Requirement that constrains or sets some quality attributes upon functional requirements [1]
Granularity level	Low	Requirement that is directly verifiable (reference: acceptance criterion).
	Medium	Requirement that refers to one specific functional or non-functional aspect of the system (reference: user story).
	High	Requirement that encompasses multiple aspects or functionalities of the systems (reference: epic or theme).

Table 1 – overview of coding scheme

Generally, requirements should be tagged at the level of entire sentences (one or more). A motivation could be tagged at the clause level (this makes the most sense, for example, in the case of user stories). The only exception is when a single sentence contains multiple requirements that can be distinguished and interpreted without the knowledge of the entire sentence.

The text should be tagged only if it clearly specifies an added functionality or non-functional aspect of the system. For example, when the text segment only states that the system needs improvement, it does not specify which functionality or quality needs improvement and what the change is. Still, only mentioning an added functionality should be tagged as a requirement, for example, "filter option by product type," indicating a functionality that allows the user to filter by product type.

To tag a text segment as a requirement, requirements must specify an added/modified functionality to the system or consider a non-functional aspect of the system; hence, process requirements should not be tagged.

1. Requirement type

Initially, it is recommended to label a requirement based on one of the options, user-oriented functional requirements, system-oriented functional requirements or non-functional requirement. Nevertheless, a requirement may *clearly* specify both a functional and a non-functional aspect. In that case, it is permissible to tag the requirement on both categories.

In some scenarios, a requirement clearly specifies user experience, which may suggest a user-oriented requirement. However, there must be functionality present to tag it as a (user- oriented) functional requirement and it must be an addition to the system that benefits the user. For example, error handling is not considered this category and should be labeled non-functional as it relates to quality attributes of the system.

Furthermore, a requirement can specify an improvement in the system. These requirements specify a necessary change in the system and must also be tagged. To determine whether it is a functional or non-functional requirement, consider whether the improvement changes a functionality (making it a functional requirement), or alters quality attributes (including appearance) of the system (indicating a non-functional requirement). If the conclusion is that it is a functional requirement, a distinction must be made between system-oriented and user-oriented (more on this in the remainder of this chapter). As indicated above, a text segment indicating only that the system needs improvement is not considered a requirement.

The remainder of this section provides more details on the different requirements types.

Functional requirements

The scheme distinguishes two different categories within functional requirements: user-oriented and system perspective.

User-oriented

User-oriented functional requirements focus on the added functionalities of the system that are directly experienced by the user. The scheme characterizes two different writing styles used for documenting user-oriented requirements:

- First person perspective: an example template for first person user-oriented requirements are User stories. However, there are also non-template versions of this writing style.
- Third person perspective: in this case, the user-oriented requirement is written from a third person perspective.

System perspective

In addition to formulating functional requirements from the user's perspective, the scheme also defines functional requirements written more from the perspective of the system. These are requirements whose effect on the user is not mentioned, is unclear or is indirect (i.e., the effect on the user is not directly caused by the addition of the functionality).

Note that specific implementation suggestions, in addition to the requirement, should be excluded.

Examples

User-oriented functional requirements:

- The home screen shows the user the most recent viewed products.
- The user is able to sort the products on alphabetical order.

System perspective functional requirements:

- An email should be sent to the third-party manufacturer when a client makes an order.
- The system must extract the data from X to feed into the model.

Non-functional requirements

In addition to functional requirements, the scheme also includes non-functional requirements. This category includes (but is not limited to) usability requirements, performance requirements and security requirements. This category excludes requirements that do not explicitly state the non-functional aspect.

Positive NFR examples

- To resolve vulnerability issues, upgrade library X to version Y.
- We need a new design for the front page to improve the ease-of-use.

Examples both FR and NFR:

- The home screen displays the user's last viewed products, so the user can easily go back if they change their mind.

Negative examples (not a requirement):

- Update library X to version y
- Team 1 needs admin rights to create feature B

2. Granularity level

In addition to the type of requirements, the scheme introduces three codes to indicate the granularity level of a requirement.

Low-level requirements

One code in the scheme refers to low-level requirements, which are generally a refinement of (non-) functional requirements. For example, this information may specify conditions that must be met or improvements that must be made to complete the related requirement. To be specific, this excludes information that indicates research activities (i.e., in this case, low-level requirements refer to questions/uncertainties that should be figured out with respect to a higher-level requirement). Low-level requirements are typically in the same issue as the requirement to which they reference. However, if this is not the case, it should be clear to which (type of) requirement or functionality it refers.

Medium-level requirements

The scheme also recognizes medium-level requirements. This code falls directly in between the other granularity levels, and a common example is a user story.

High-level requirement

The scheme also identifies backlog item segments that provide a higher-level requirement. For example, this higher-level requirements can be represented as an epic or short title (i.e., indicating only the main theme). In general, the description in this category is fairly high-level, making it challenging to directly identify individual requirements. Nevertheless, the purpose of the requirement must be clear enough to be included in this category. In addition, this category excludes an item that specifically mentions more than one functional or non-functional requirement because these will be tagged separately.

Finally, epics used only as grouping of multiple backlog items should not be tagged because they do not specify a functional or non-functional part of the system. An example of such a text segment could be: ‘Features related to search engine’.

Examples

Low-level requirements:

- When an user sorts the product list on alphabetical order, the product names are highlighted.

Medium level requirements:

- An user is able to sort the products on alphabetical order.
- An user can see an overview of all products.

High-level requirements:

- Page with all product from third party manufacturers.

3. Motivation of requirements

The scheme also recognizes information that represents the motivation or reason for a particular requirement. This category disregards information related to bugs, and includes information that specifies:

- The reason why the current state is undesirable, or
- Whether a stakeholder specifically requested the requirement and optionally who that stakeholder is.
- Other.

4. Reference

[1] Cysneiros, L.M., do Prado Leite, J.C.S., de Melo Sabat Neto, J.: A framework for integrating non-functional requirements into conceptual models. Requirements Engineering 6, 97–115 (2001)