What is the main difference between Functional and Non-Functional Requirements in SE? Also draw a tree structure of Non-Functional Requirements in SE.

Functional Requirements

Functional requirements describe how a system can operate and behave upon certain inputs. These are documented in a software requirements specification (SRS) describing the expected system behavior. The lists of the services or functions that a user expects from the software are known as functional requirements. The type of software being developed, the expected users of the software, and the general approach taken by an organization when writing requirements all influence functional requirements. [1]

- The functional requirements document will include use cases which means describe how a user will interact with the system.
- User stories will be included in the functional requirements document, which will explain how a user can interact with the developed product step by step.
- The background, definitions, and system overview will all be included in the functional requirements document.
- The functional requirements document will include to describe the scope of product, the expectations, and the business rules.
- The functional requirements document will include to define the database requirements and system attributes.
- The functional requirements document will include to write down the **goal of the product** clearly.

Non-Functional Requirements:

A software requirement in software engineering that explains not what the software will do, but how the software will do it. Non-functional requirements include all the remaining requirements that are not covered by the functional requirements. [1] These are basically the quality constraints that the system must meet in order to fulfill the project contract. Non-functional requirements are difficult to test. Therefore, they are usually evaluated subjectively.

For example, the Non-Functional requirements of software performance, software external interface, software design constraints and software quality attributes.

Different Non-Functional properties and their interpretation in relation to software development is presented in Table 1.1.

Non-Functional Properties	Interpretation
Usability	It gives the ease with which the targeted users can use the system.
Safety	It is the degree to which the accidental harm is detected and prevented.
Robustness	It is the ability of the system responding effectively to different abnormal system conditions.
Integrity	The degree to which the intentional and unauthorized access to the system is restricted.
Efficiency	It refers to the effective usage of all the available resources.
Reliability	It is the extent to which the system works without failure
Maintainability	It is the ease of maintaining the systems throughout its life cycle.
Re-usability	It is the extent of reusing several components of the system.

Portability	It can be the extent to which the system can be moved in different environments or being able to run the software on different platforms.
Testability	It denotes the extent to which the system under test reacts to the testing process of creating and executing successful tests.
Interoperability	It is the degree to which the system or the individual components being interconnected for different operations.
Security	"It is the degree to which malicious harm to a valuable asset is prevented, detected and reacted to." [2]
Performance	Different timing characters of the software like Jitter, Throughput, Response Time, etc. are referred using this property.
Quality of Service *(QOS)*	It is extent to which the system offers its services with good quality being rendered to the user.
Availability	It is level to which the system is always up for the users to use.
Scalability	It is level with which the current system can be modified to a enhance the current capabilities.

Table 1.1 Non-Functional properties and their interpretation

A mind map of these properties is also available in figure 1.2 has been inspired from [3]. Chung et al. have derived different classification schemes to portray the non-functional system properties. Representing the non-functional system properties are classified considering different constraints like the Life-cycle, operational, interface, performance and economic constraints.

Figure 1.2 is the overview and many other properties can still be broken down into different properties. It cannot be said that this list of properties is not everything. As the complexity of the software systems is increasing, the non-functional properties are expected to diversify.

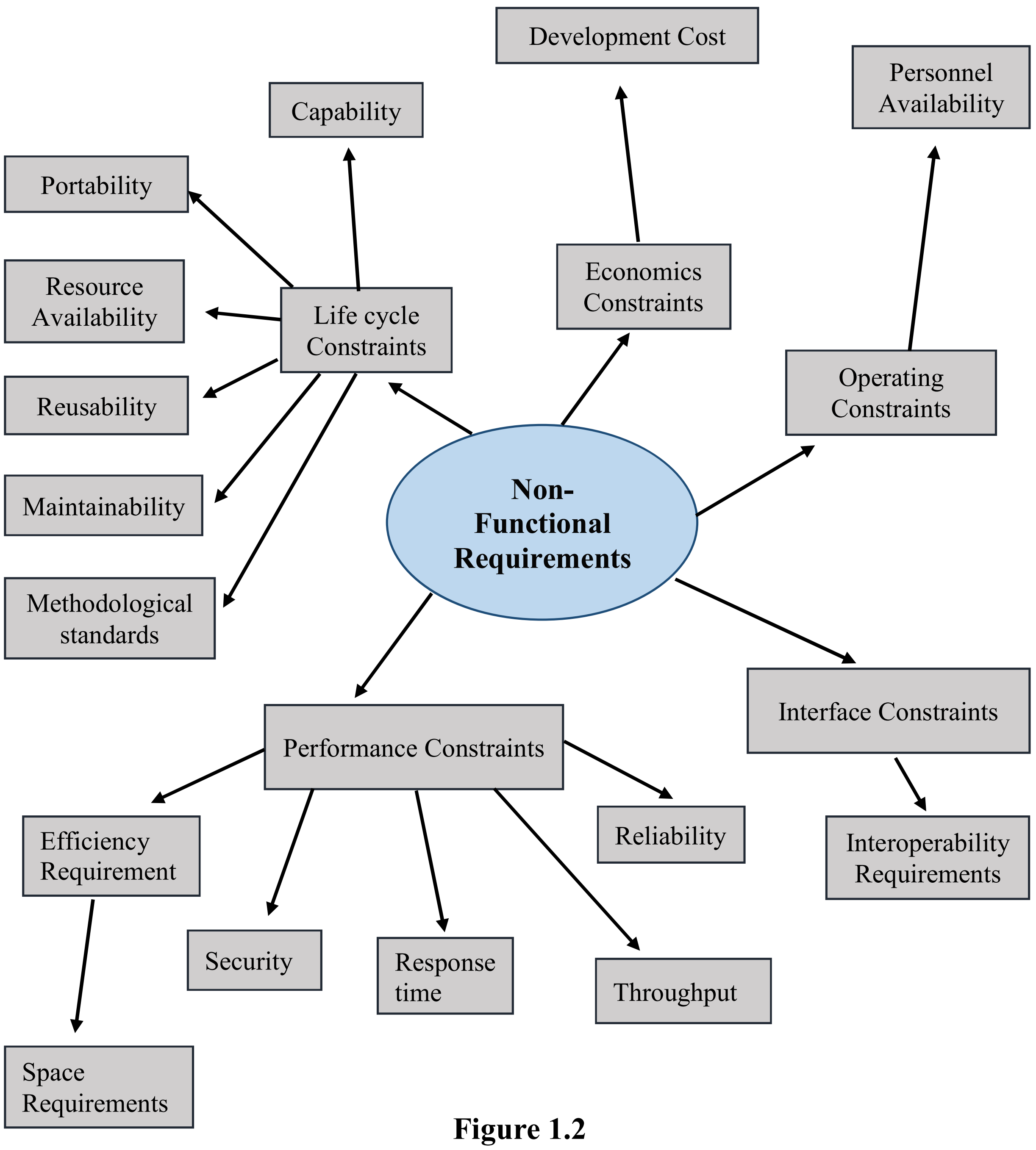

Figure 1.2

Now take a look at the table below to get a better understanding between Functional and Non-Functional Requirements

Functional Requirements	**Non-Functional Requirements**
Describes how a feature should work	Describes which properties will make the feature work
Focuses on the output of user actions	Focuses on simplifying the process and execution of the output on user actions
Defines user requirements	Defines user's expectations and experience
It has functions covered in use cases	It has constraints which will help in reducing development time and cost
Verifies the functionality of the system	Verifies the performance of the system
They define a system or its component.	They define the quality attribute of a system
It describes What the system should do?	It describes How should the system fulfill the functional requirements?
User specifies functional requirement.	Non-functional requirement is specified by technical peoples e.g. Architect, Technical leaders and software developers.
It is mandatory to meet these requirements.	It is not mandatory to meet these requirements.
It is captured in use case.	It is captured as a quality attribute.
Defined at a component level.	Applied to a whole system.
Verify the functionality of the software.	Verify the performance of the software.
System, integration, end-to-end, API, and other types of functional testing are carried out.	Performance, Stress, Usability, Security testing and other types of Non-Functional Testing are carried out.
Usually easy to define.	Usually more difficult to define

Conclusion:

I feel that functional requirements come from the clients to the developer, and non-functional requirements arrive from the developer to the client, i.e. the requirement is not granted by the client but supplied by the developer to ensure that the system runs smoothly, such as safety, security, flexibility, scalability, and availability.

References:

[1] R. H. Thayer and M. Dorfman. *System and software requirements engineering*. IEEE computer society press, 1995.

[2] D. G. Firesmith. Common concepts underlying safety, security, and survivability engineering. Technical Report CMU/SEI-2003-TN-033, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2003.

[3] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos. *Non-functional requirements in software engineering*, volume 5. Springer Science & Business Media, 2012.