**Non-Functional requirements :**

Non-functional requirements tend to be stated in terms of **constraints** on the results of tasks which are given as functional requirements (e.g., constraints on the speed or efficiency of a given task), a task-based functional requirements statement is a useful skeleton upon which to construct a complete requirements statement. That is the approach taken in this work. It can be helpful to think of non-functional requirements as adverbially related to tasks or functional requirements: how fast, how efficiently, how safely, etc., is a particular task carried out by a particular system?

Non-functional requirements are often called **qualities** of a system. Other terms for non-functional requirements are "constraints", "quality attributes", "quality goals", "quality of service requirements" and "non-behavioral requirements". Qualities, that is non-functional requirements, can be divided into two main categories:

1. Execution qualities, such as security and usability, which are observable at run time.
2. Evolution qualities, such as [testability](http://en.wikipedia.org/wiki/Software_testability), maintainability, extensibility and scalability, which are embodied in the static structure of the software system

The non-functional requirements are

1. Accessbility

2. Availabilty

3. Scalabilty

4. Portability

5. Robustness

**Accessibility** is a general term used to describe the degree to which a product, device, service, or environment is available to as many people as possible. Accessibility can be viewed as the "ability to access" and benefit from some system or entity. Accessibility is often used to focus on people with disabilities or special needs and their right of access to entities, often through use of [assistive technology](http://en.wikipedia.org/wiki/Assistive_technology).

Accessibility is not to be confused with [usability](http://en.wikipedia.org/wiki/Usability) which is used to describe the extent to which a product (e.g., device, service, environment) can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.

Availabilty

* The degree to which a [system](http://en.wikipedia.org/wiki/System), [subsystem](http://en.wikipedia.org/wiki/Subsystem), or equipment is in a specified operable and committable state at the start of a mission, when the mission is called for at an unknown, *i.e.,* a random, time. Simply put, availability is the proportion of time a system is in a functioning condition. This is often described as a **mission capable rate**. Mathematically, this is expressed as 1 minus [unavailability](http://en.wikipedia.org/wiki/Unavailability).
* The ratio of (a) the total time a [functional unit](http://en.wikipedia.org/wiki/Functional_unit) is capable of being used during a given interval to (b) the length of the interval.

For example, a unit that is capable of being used 100 hours per week (168 hours) would have an availability of 100/168. However, typical availability values are specified in [decimal](http://en.wikipedia.org/wiki/Decimal) (such as 0.9998). In [high availability](http://en.wikipedia.org/wiki/High_availability) applications, a metric known as [nines](http://en.wikipedia.org/wiki/Nines_%28engineering%29), corresponding to the number of nines following the decimal point, is used. In this system, "five nines" equals 0.99999 (or 99.999%) availability.

**scalability** is the ability of a system, network, or process, to handle growing amount of work in a capable manner or its ability to be enlarged to accommodate that growth.[[1]](http://en.wikipedia.org/wiki/Scalability#cite_note-0) For example, it can refer to the capability of a system to increase total throughput under an increased load when resources (typically hardware) are added. An analogous meaning is implied when the word is used in a [commercial](http://en.wikipedia.org/wiki/Commerce) context, where scalability of a company implies that the underlying [business model](http://en.wikipedia.org/wiki/Business_model) offers the potential for [economic growth](http://en.wikipedia.org/wiki/Economic_growth) within the company.

Scalability, as a property of systems, is generally difficult to define[[2]](http://en.wikipedia.org/wiki/Scalability#cite_note-1) and in any particular case it is necessary to define the specific requirements for scalability on those dimensions that are deemed important. It is a highly significant issue in electronics systems, databases, routers, and networking. A system whose performance improves after adding hardware, proportionally to the capacity added, is said to be a **scalable system**. An [algorithm](http://en.wikipedia.org/wiki/Algorithm), design, [networking protocol](http://en.wikipedia.org/wiki/Protocol_%28computing%29), [program](http://en.wikipedia.org/wiki/Computer_program), or other system is said to **scale**, if it is suitably [efficient](http://en.wikipedia.org/wiki/Algorithmic_efficiency) and practical when applied to large situations (e.g. a large input data set or a large number of participating nodes in the case of a distributed system). If the design fails when the quantity increases, it **does not scale**.

The concept of scalability is desirable in technology as well as [business](http://en.wikipedia.org/wiki/Business) settings. The base concept is consistent - the ability for a business or technology to accept increased volume without impacting the [contribution margin](http://en.wikipedia.org/wiki/Contribution_margin) (= [revenue](http://en.wikipedia.org/wiki/Revenue) - [variable costs](http://en.wikipedia.org/wiki/Variable_costs)). For example, a given piece of equipment may have capacity from 1-1000 users, and beyond 1000 users, additional equipment is needed or performance will decline (variable costs will increase and reduce contribution margin).

**Portability** in [high-level computer programming](http://en.wikipedia.org/wiki/High-level_programming_language) is the usability of the same [software](http://en.wikipedia.org/wiki/Software) in different environments. The prerequirement for portability is the generalized [abstraction](http://en.wikipedia.org/wiki/Abstraction_%28computer_science%29) between the application logic and [system interfaces](http://en.wikipedia.org/wiki/Interface_%28computer_science%29). When software with the same functionality is produced for several [computing platforms](http://en.wikipedia.org/wiki/Computing_platform), portability is the key issue for development cost reduction.

This article is about portability in itself. The work required to make software portable is described in the article on [porting](http://en.wikipedia.org/wiki/Porting).

In [computer science](http://en.wikipedia.org/wiki/Computer_science), **robustness** is the ability of a computer system to cope with errors during execution or the ability of an algorithm to continue to operate despite abnormalities in input, calculations, etc. Formal techniques, such as [fuzz testing](http://en.wikipedia.org/wiki/Fuzz_testing), are essential to showing robustness since this type of testing involves invalid or unexpected inputs. Various commercial products perform robustness testing of software systems. Robustness is a consideration in [failure assessment](http://en.wikipedia.org/wiki/Failure_assessment) analysis.