Requirements specification: a structured document that sets out the services the system is expected to provide. It should be precise so that it can act as a contract between the system procurer and software developer and thus needs to be understandable by procurers and developers. The requirements specification describes *what* the system will do but not *how* it will do it, i.e., the objectives but not how those objectives will be achieved.

Design specification: an abstract description of the software that serves as a basis for its detailed design and implementation. It describes *how* the requirements will be achieved. The primary readers of this document will be software designers and implementers rather than users or management. The goals and constraints specificed in the requirements document should be traceable to the design specification (and from there to the code.

Phases of the requirements definitions process:

(1) **Feasibility Study**: An estimate is made of whether the identified user needs can be satisfied using current software and hardware technologies, whether the proposed system will be cost-effective from a business point of view, and whether it can be developed given existing budgetary constraints. (refs: P.A. Kendall, Systems Analysis and Design, Wm. Brown, 1989 and W.S. Davis, Systems Analysis and Design, Addison-Wesley, 1983, or other books on systems analysis.

(2) **Requirements Capture and Analysis**: The process of deriving the software requirements through observation of existing systems, discussions wth potential users and procurers, task analysis, etc.

(3) **Requirements Definition**: Formulation of a system model and the goals and constraints of the software. The requirements definition may be carried out in parallel with some high-level design.

Attributes of a requirements document:

- Should be readable and understandable by customers, users, and designers.
- Should specify only external system behavior (black box).
- Should be structured to be easy to change.
- Should specify both goals and constraints.
- Should be able to serve as a reference tool for system maintainers.
- Should contain only testable requirements.
- Should be consistent, complete, unambiguous, and realistic.
- Should specify acceptable responses to undesired events.
- Should specify incremental subsets if desired or minimum and maximum functionality.
- Should specify the changes anticipated in the future, i.e, the things in the environment most likely to change that will require changes in the software.

Contents of requirements document:

- **Introduction**: Describes the need for the system and places it in context, briefly describing its functions and presenting a rationale for the software. Describes how the system fits into the overall business or strategic objectives of the organization commissioning the software.
- **System Model**: Sets out the system model showing the relationships betwen the system components and the system and its environment. An abstract data model should also be described if appropriate to the type of system.
- **System Evolution**: The fundamental assumptions on which the system is based and anticipated changes due to hardare evolution, changing user needs, etc.
- **Goals** (sometimes called **Functional Requirements**): The services provided for the user. This includes timing and accuracy requirements.
- **Constraints**: Constraints on how the goals can be achieved (restrictions on the behavior of the software and the freedom of the designer. These are restrictions on allowable designs or the ways the goals can be achieves, e.g., safety constraints, hardware, programming languages, and standards that must be followed. They also include quality requirements, such as maintainability, availability, etc.
- **Priorities**: This section basically specifies how tradeoffs will be made if all of the goals and constraints cannot be completely achieved in the design---some may be more important than others. Guides tradeoffs among design decisions.
- **Interfaces to the environment**: Input or output interfaces and relevant assumptions about environmental components with which the software will be interacting.
- **Glossary**: Definitions of technical terms used in the document.
- **Indexes**: Various types of indexes may be provided.

May be quite large and complex. One requirements document for a ballistic missile defense system contained over 8000 distinct requirements and support paragraphs and was 2500 pages long.

Requirements should be traceable to design and should be stated so that the design can be validated.

An untestable requirement: The system should be easy to use by experienced controllers and should be organized in such a way that user errrors are minimized.

- *A testable requirement*: Experienced controllers should be able to use all of the system functions after a total of two hours training. After this training, the average number of errors made by experienced users should not exceed two per day.