Software Engineering

Week # 2

LECTURER: ABDULRAHMAN QAIM

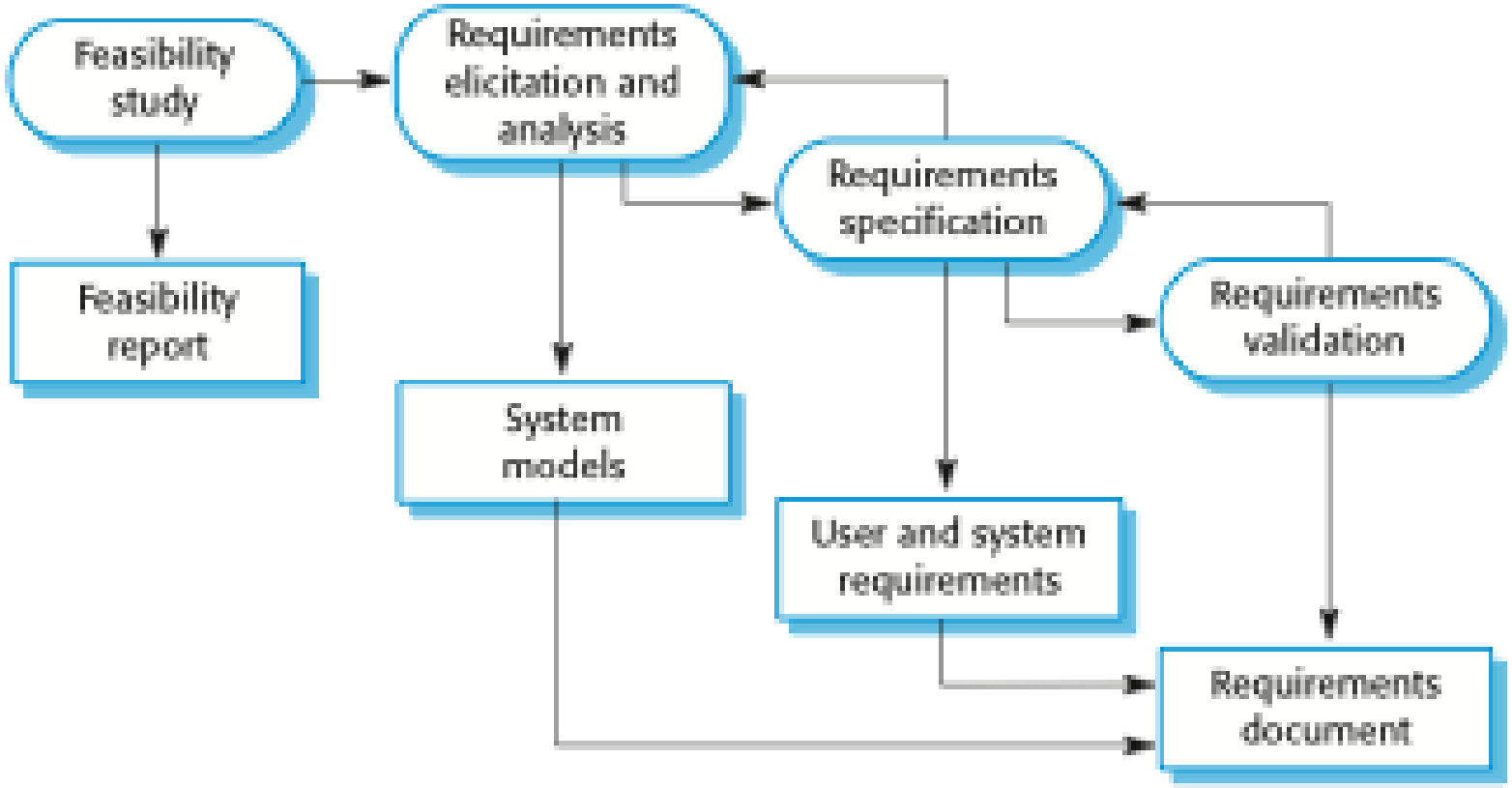
Software specification

The process of establishing what services are required and the constraints on the system's operation and development.

Requirements engineering process

- Feasibility study
  - Is it technically and financially feasible to build the system?
- Requirements elicitation and analysis
  - What do the system stakeholders require or expect from the system?
- Requirements specification
  - Defining the requirements in detail
- Requirements validation
  - Checking the validity of the requirements

The requirements engineering process



Software design and implementation

The process of converting the system specification into an executable system.

Software design

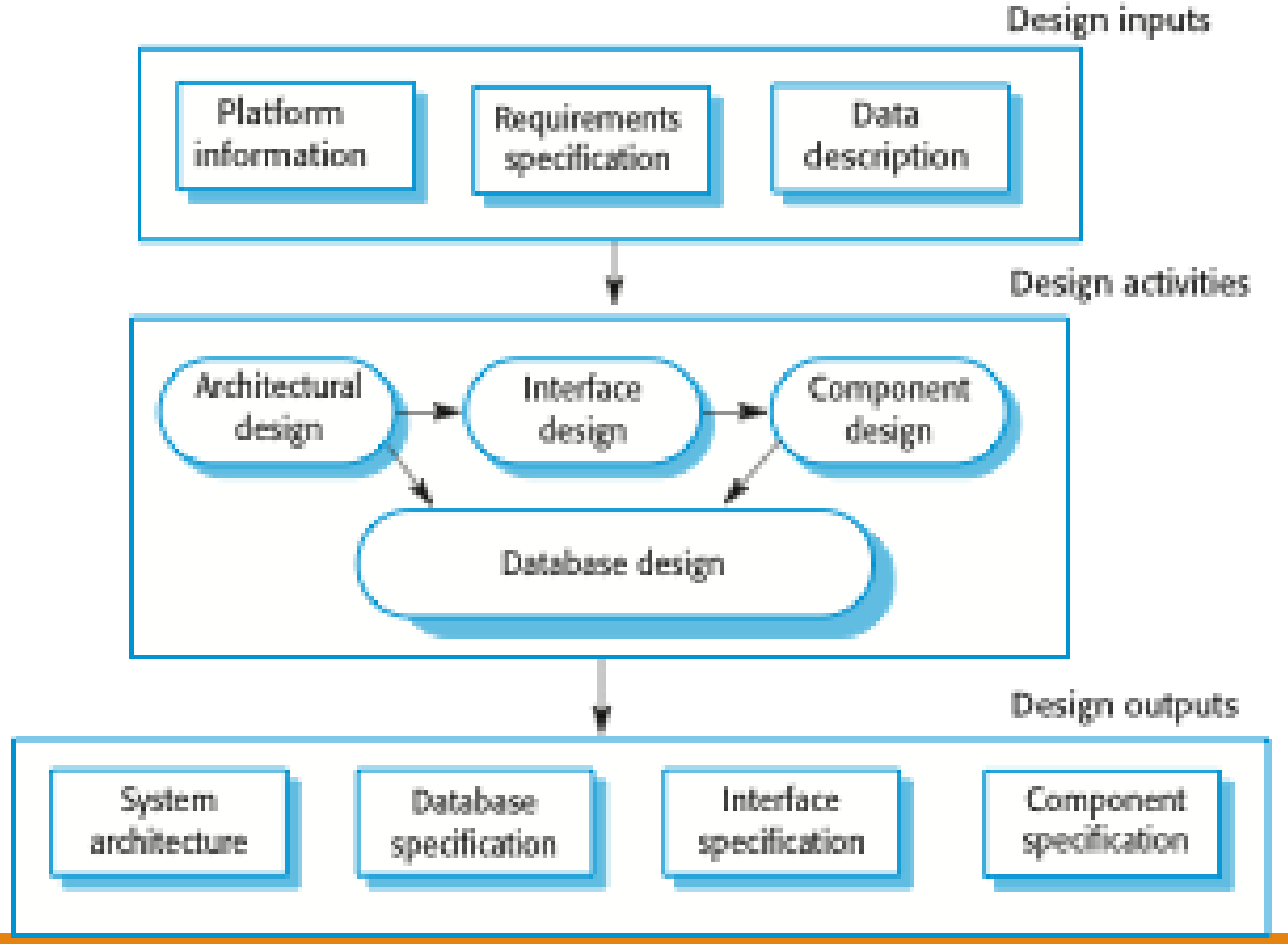
- Design a software structure that realises the specification;

Implementation

- Translate this structure into an executable program;

The activities of design and implementation are closely related and may be inter-leaved.

A general model of the design process



Design activities

Architectural design, where you identify the overall structure of the system, the principal components (sometimes called sub-systems or modules), their relationships and how they are distributed.

Interface design, where you define the interfaces between system components.

Component design, where you take each system component and design how it will operate.

Database design, where you design the system data structures and how these are to be represented in a database.

Software validation

Verification and validation (V & V) is intended to show that a system conforms to its specification and meets the requirements of the system customer.

Involves checking and review processes and system testing.

System testing involves executing the system with test cases that are derived from the specification of the real data to be processed by the system.

Testing is the most commonly used V & V activity.

Testing stages

Development or component testing

- Individual components are tested independently;
- Components may be functions or objects or coherent groupings of these entities.

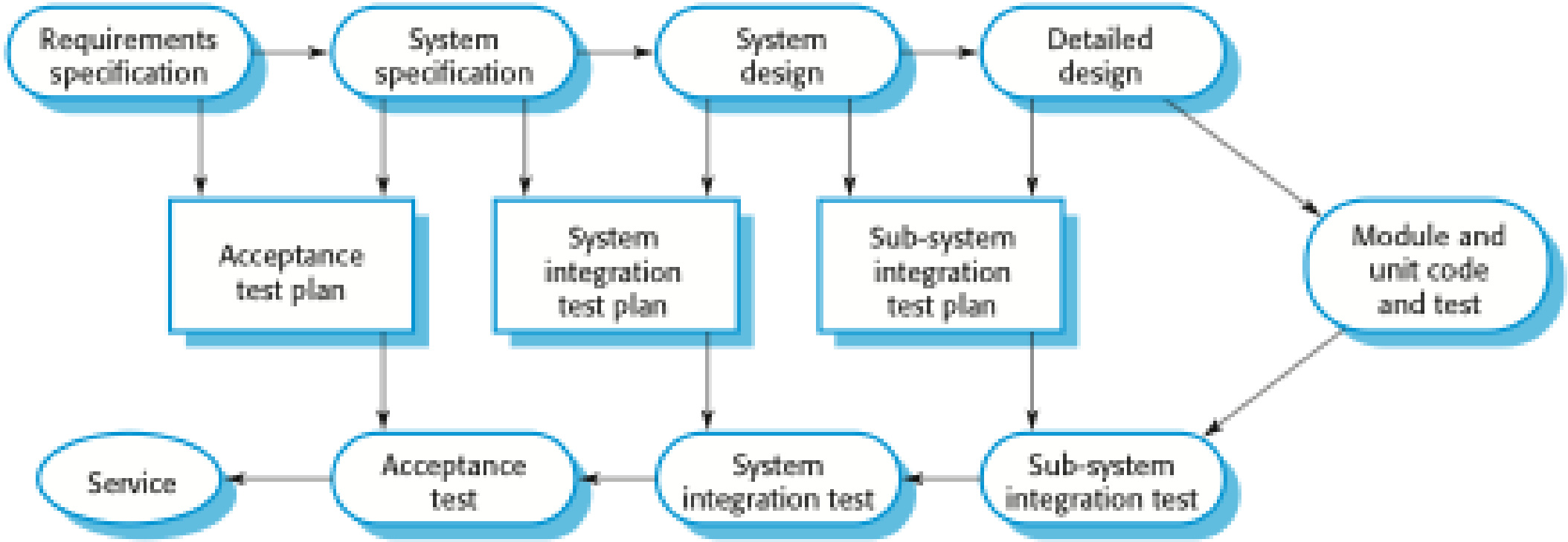
System testing

- Testing of the system as a whole. Testing of emergent properties is particularly important.

Acceptance testing

- Testing with customer data to check that the system meets the customer’s needs.

Testing phases in a plan-driven software process



Coping with change

Change is inevitable in all large software projects.

- Business changes lead to new and changed system requirements
- New technologies open up new possibilities for improving implementations
- Changing platforms require application changes

Change leads to rework so the costs of change include both rework (e.g. re-analysing requirements) as well as the costs of implementing new functionality

Reducing the costs of rework

Change avoidance, where the software process includes activities that can anticipate possible changes before significant rework is required.

- For example, a prototype system may be developed to show some key features of the system to customers.

Change tolerance, where the process is designed so that changes can be accommodated at relatively low cost.

- This normally involves some form of incremental development. Proposed changes may be implemented in increments that have not yet been developed. If this is impossible, then only a single increment (a small part of the system) may have be altered to incorporate the change.

Software prototyping

A prototype is an initial version of a system used to demonstrate concepts and try out design options.

A prototype can be used in:

- The requirements engineering process to help with requirements elicitation and validation;
- In design processes to explore options and develop a UI design;
- In the testing process to run back-to-back tests.

Benefits of prototyping

Improved system usability.

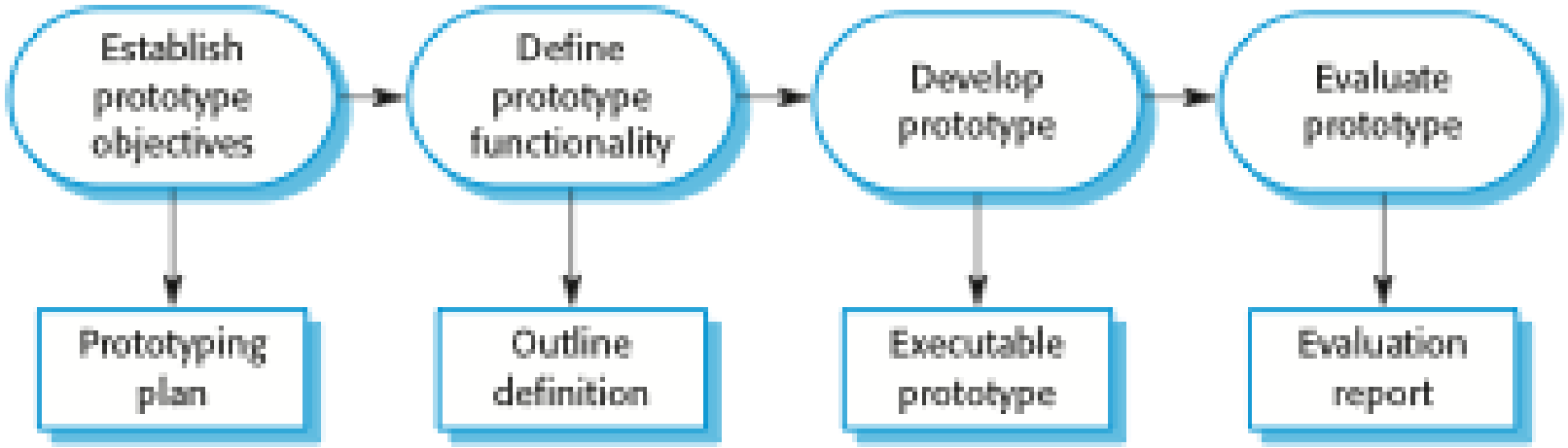
A closer match to users' real needs.

Improved design quality.

Improved maintainability.

Reduced development effort.

The process of prototype development



Incremental delivery

Rather than deliver the system as a single delivery, the development and delivery is broken down into increments with each increment delivering part of the required functionality.

User requirements are prioritised and the highest priority requirements are included in early increments.

Once the development of an increment is started, the requirements are frozen though requirements for later increments can continue to evolve.

Incremental delivery advantages

Customer value can be delivered with each increment so system functionality is available earlier.

Early increments act as a prototype to help elicit requirements for later increments.

Lower risk of overall project failure.

The highest priority system services tend to receive the most testing.

Incremental delivery problems

Most systems require a set of basic facilities that are used by different parts of the system.

- As requirements are not defined in detail until an increment is to be implemented, it can be hard to identify common facilities that are needed by all increments.

The essence of iterative processes is that the specification is developed in conjunction with the software.

- However, this conflicts with the procurement model of many organizations, where the complete system specification is part of the system development contract.

Types of requirement

User requirements

- Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.

System requirements

- A structured document setting out detailed descriptions of the system's functions, services and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor.

User and system requirements

User Requirement Definition

1. The MHC-PMS shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

System Requirements Specification

1.1 On the last working day of each month, a summary of the drugs prescribed, their cost, and the prescribing clinics shall be generated.

1.2 The system shall automatically generate the report for printing after 17.30 on the last working day of the month.

1.3 A report shall be created for each clinic and shall list the individual drug names, the total number of prescriptions, the number of doses prescribed, and the total cost of the prescribed drugs.

1.4 If drugs are available in different dose units (e.g., 10 mg, 20 mg) separate reports shall be created for each dose unit.

1.5 Access to all cost reports shall be restricted to authorized users listed on a management access control list.

Figure 4.1 User and system requirements

Readers of different types of requirements specification

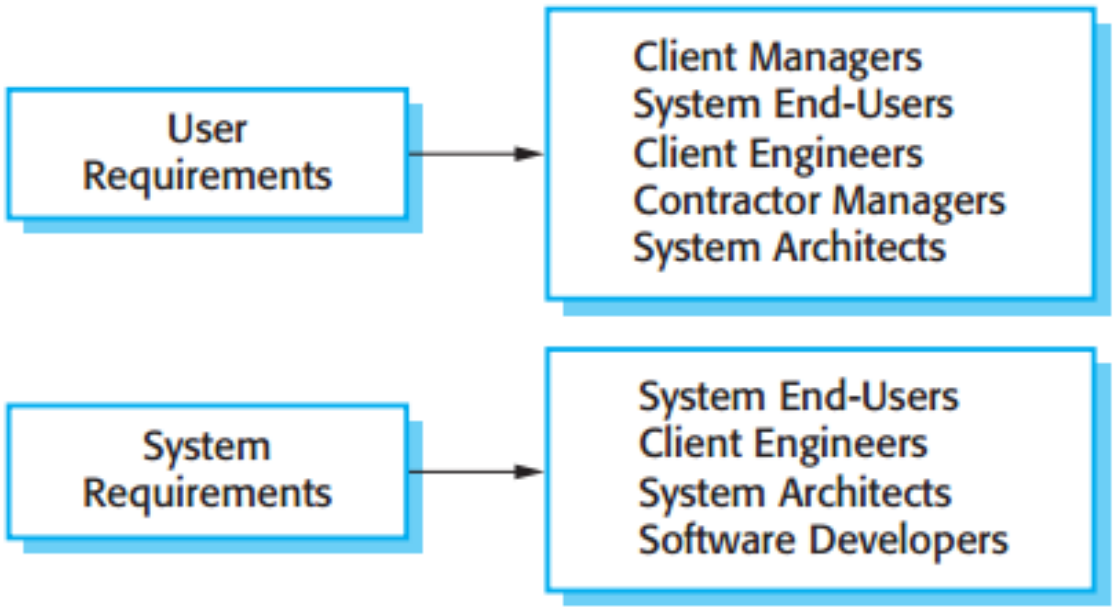


Figure 4.2 Readers of different types of requirements specification

Functional and non-functional requirements

Functional requirements

- Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.
- May state what the system should not do.

Non-functional requirements

- Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
- Often apply to the system as a whole rather than individual features or services.

Functional requirements for the MHC-PMS

A user shall be able to search the appointments lists for all clinics.

The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.

Each staff member using the system shall be uniquely identified by his or her 8-digit employee number.

Requirements imprecision

Problems arise when requirements are not precisely stated.

Ambiguous requirements may be interpreted in different ways by developers and users.

Consider the term ‘search’ in requirement 1

- User intention – search for a patient name across all appointments in all clinics;
- Developer interpretation – search for a patient name in an individual clinic. User chooses clinic then search.

Requirements completeness and consistency

In principle, requirements should be both complete and consistent.

Complete

- They should include descriptions of all facilities required.

Consistent

- There should be no conflicts or contradictions in the descriptions of the system facilities.

In practice, it is impossible to produce a complete and consistent requirements document.

Non-functional requirements

These define system properties and constraints e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc.

Process requirements may also be specified mandating a particular IDE, programming language or development method.

Non-functional requirements may be more critical than functional requirements. If these are not met, the system may be useless.

Non-functional requirements implementation

Non-functional requirements may affect the overall architecture of a system rather than the individual components.

- For example, to ensure that performance requirements are met, you may have to organize the system to minimize communications between components.

A single non-functional requirement, such as a security requirement, may generate a number of related functional requirements that define system services that are required.

- It may also generate requirements that restrict existing requirements.

Non-functional classifications

Product requirements

- Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.

Organisational requirements

- Requirements which are a consequence of organisational policies and procedures e.g. process standards used, implementation requirements, etc.

External requirements

- Requirements which arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements, etc.

Examples of nonfunctional requirements in the MHC-PMS

PRODUCT REQUIREMENT
The MHC-PMS shall be available to all clinics during normal working hours (Mon–Fri, 08.30–17.30). Downtime within normal working hours shall not exceed five seconds in any one day.

ORGANIZATIONAL REQUIREMENT
Users of the MHC-PMS system shall authenticate themselves using their health authority identity card.

EXTERNAL REQUIREMENT
The system shall implement patient privacy provisions as set out in HStan-03-2006-priv.

Goals and requirements

Non-functional requirements may be very difficult to state precisely and imprecise requirements may be difficult to verify.

Goal

- A general intention of the user such as ease of use.

Verifiable non-functional requirement

- A statement using some measure that can be objectively tested.

Goals are helpful to developers as they convey the intentions of the system users.

Metrics for specifying nonfunctional requirements

Property	Measure
Speed	Processed transactions/second User/event response time Screen refresh time
Size	Mbytes Number of ROM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems