SOFTWARE ENGINEERING

– Software Requirements –

Descriptions and specifications of a system

Objectives:

- To introduce the concepts of **user and system requirements**
- To describe **functional** / **non-functional requirements**
- To explain **how software requirements may be organised** in a requirements document

What is a requirement?

- It may range from a **high-level** abstract statement of a service or of a system constraint to a **detailed** mathematical functional specification

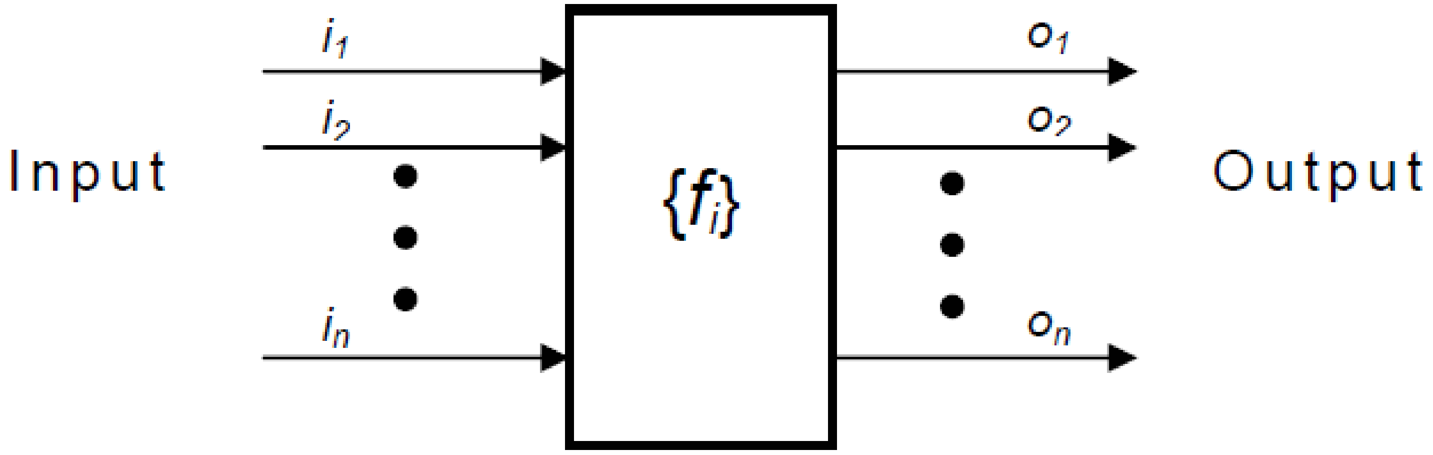
- **Functional requirements**
  - Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.
- **Non-functional requirements**
  - Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
- **Domain requirements**
  - Requirements that come from the application domain of the system and that reflect characteristics of that domain

FUNCTIONAL REQUIREMENTS

Functional requirements:-

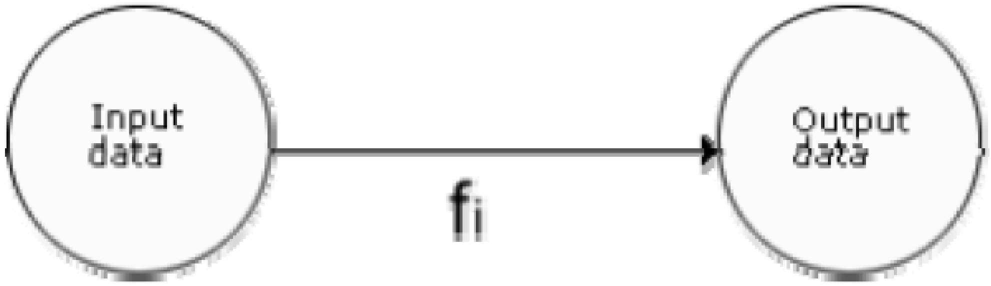
- The functional requirements part discusses the functionalities required from the system.
- The system is considered to perform a set of high-level functions {*fi*}.
- Each function fi of the system can be considered as a transformation of a set of input data to the corresponding set of output data.

FUNCTIONAL REQUIREMENTS



FUNCTIONAL REQUIREMENTS

Function fi is considered as a transformation of a set of input data to some corresponding output data.



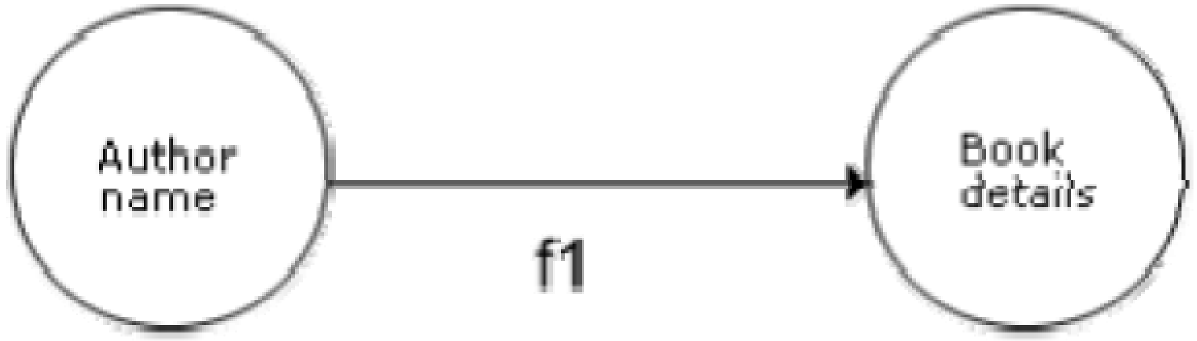
FUNCTIONAL REQUIREMENTS

Example: Consider case of library system

f1: Search Book function

Input: an author's name

Output: details of the author's books and the location of these books in the library



NONFUNCTIONAL REQUIREMENTS

Nonfunctional requirements:-

- Nonfunctional requirements deal with the characteristics of the system which can not be expressed as functions - such as the maintainability of the system, portability of the system, usability of the system, etc.
- Nonfunctional requirements may include: **#** reliability issues, **#** accuracy of results, **#** human - computer interface issues, **#** constraints on the system implementation, etc.

Non-Functional Requirements

- **Availability**
  - Availability is a measure of the planned *up time* during which the system is actually available for use and fully operational.
  - More formally, availability equals the mean time to failure (MTTF) for the system divided by the sum of the MTTF and the mean time to repair the system after a failure is encountered.
  - Scheduled maintenance periods also affect availability.
  - Some authors view availability as encompassing reliability, maintainability, and integrity.

NON-FUNCTIONAL REQUIREMENTS

- **Efficiency**
- Efficiency is a measure of how well the system utilizes processor capacity, disk space, memory, or communication bandwidth.
- Efficiency is related to performance, another class of nonfunctional requirement.
- If a system consumes too much of the available resources, users will encounter degraded performance, a visible indication of inefficiency.

NON-FUNCTIONAL REQUIREMENTS

- **Flexibility**
- Flexibility measures how easy it is to add new capabilities to the product.
- If developers anticipate making many enhancements, they can choose design approaches that maximize the software's flexibility.
- This attribute is essential for products that are developed in an incremental or iterative fashion through a series of successive releases or by evolutionary prototyping.

NON-FUNCTIONAL REQUIREMENTS

- **Interoperability**
- Interoperability indicates how easily the system can exchange data or services with other systems.

NON-FUNCTIONAL REQUIREMENTS

- **Reliability**
- The probability of the software executing without failure for a specific period of time is known as reliability.
- Ways to measure software reliability include the percentage of operations that are completed correctly and the average length of time the system runs before failing.

NON-FUNCTIONAL REQUIREMENTS

- **Robustness**
- *Robustness*, sometimes called *fault tolerance,* is the degree to which a system continues to function properly when confronted with invalid inputs, defects in connected software or hardware components, or unexpected operating conditions.
- Robust software recovers gracefully from problem situations and is forgiving of user mistakes.
- When eliciting robustness requirements, ask users about error conditions the system might encounter and how the system should react.

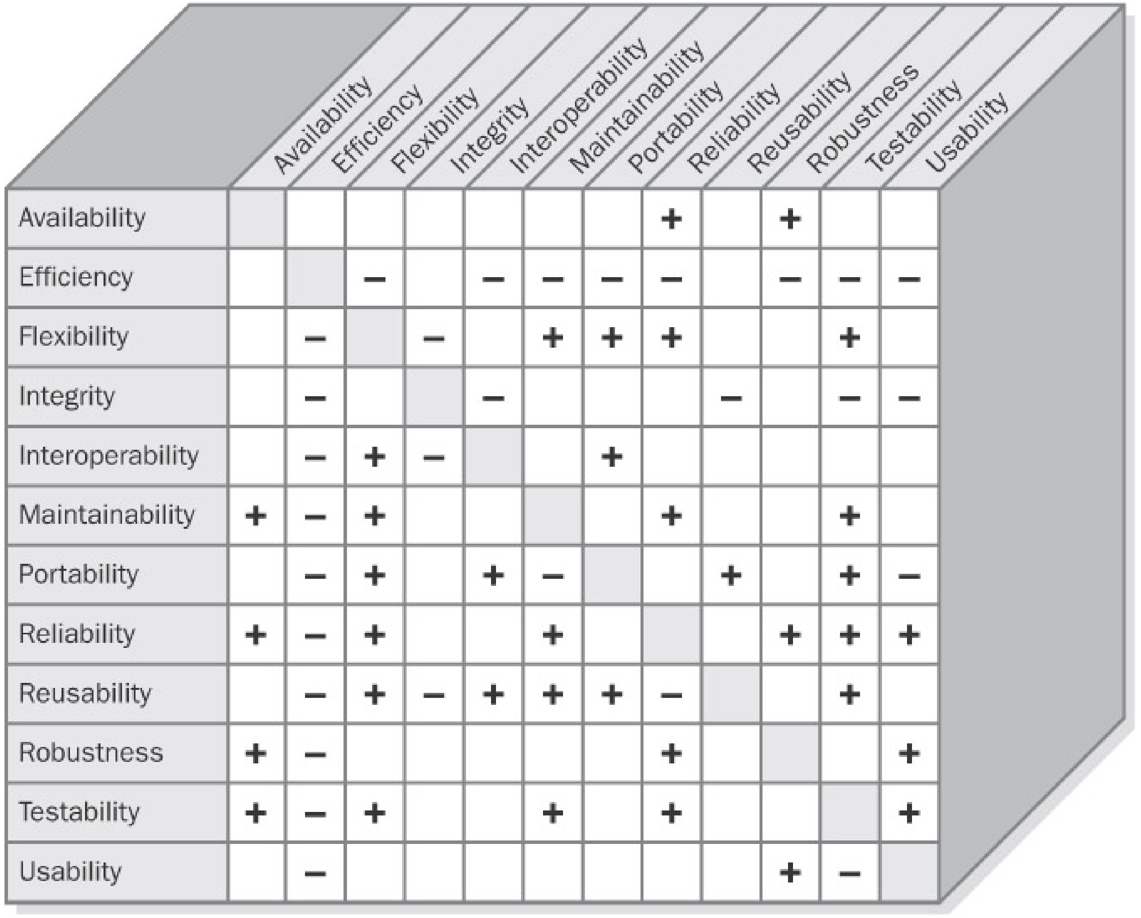
NON-FUNCTIONAL REQUIREMENTS

- **Usability**
- Also referred to as *ease of use* and *human engineering*, usability addresses the myriad factors that constitute what users often describe as *user-friendliness*.
- Software designed for effective and unobtrusive usage.
- Usability measures the effort required to prepare input for, operate, and interpret the output of the product.

NON-FUNCTIONAL REQUIREMENTS

- Integrity
  - No Change in data

- **Maintainability**
- Maintainability indicates how easy it is to correct a defect or modify the software.
- Maintainability depends on how easily the software can be understood, changed, and tested.



	Availability	Efficiency	Flexibility	Integrity	Interoperability	Maintainability	Portability	Reliability	Reusability	Robustness	Testability	Usability
Availability								+		+		
Efficiency			−		−	−	−	−		−	−	−
Flexibility		−		−		+	+	+			+	
Integrity		−			−				−		−	−
Interoperability		−	+	−			+					
Maintainability	+	−	+					+			+	
Portability		−	+		+	−			+		+	−
Reliability	+	−	+			+				+	+	+
Reusability		−	+	−	+	+	+	−			+	
Robustness	+	−						+				+
Testability	+	−	+			+		+				+
Usability		−								+	−	

REQUIREMENTS INTERACTION

Conflicts between different non-functional requirements are common in complex systems

Spacecraft system

- To minimise weight, the number of separate chips in the system should be minimised
- To minimise power consumption, lower power chips should be used
- However, using low power chips may mean that more chips have to be used.
- **Which is the most critical requirement?**

Domain requirements

- **Derived from the application domain** and **describe system characteristics and features** that reflect the domain
- May be new functional requirements, constraints on existing requirements or define specific computations
- If domain requirements are not satisfied, the system may be unworkable

STRUCTURED LANGUAGE SPECIFICATIONS

A limited form of natural language may be used to express requirements

This removes some of the problems resulting from ambiguity and flexibility and imposes a degree of uniformity on a specification

Special-purpose forms where designed to describe the input, output and functions of a software system

Leads to Software Requirements Specifications (SRS) document.

SRS Document

- Parts of SRS document are:

  - Functional requirements of the system
  - Non-functional requirements of the system, and
  - Domain Requirements

SRS Document

- The requirements document is the official statement of what is required of the system developers
- Should include both a definition and a specification of requirements
- It is NOT a design document. As far as possible, it should set of WHAT the system should do rather than HOW it should do it

PROBLEMS WITH NATURAL LANGUAGE

- **Lack of clarity**
  - Precision is difficult without making the document difficult to read
- **Requirements confusion**
  - Functional and non-functional requirements tend to be mixed-up
- **Requirements combination**
  - Several different requirements may be expressed together

Methods for Making SRS Document

- Form-based specifications
- PDL-based requirements definition
- Planguage

Form-based specifications

- Definition of the function or entity
- Description of inputs and where they come from
- Description of outputs and where they go to
- Indication of other entities required
- Pre and post conditions (if appropriate)
- The side effects (if any)

PDL-BASED REQUIREMENTS DEFINITION

- Requirements may be defined operationally using a language like a programming language but with more flexibility of expression

- **Most appropriate in two situations**
  - Where an operation is specified as a sequence of actions and the order is important
  - When hardware and software interfaces have to be specified

PART OF AN ATM SPECIFICATION

```
class ATM {
  // declarations here
  public static void main (String args[]) throws InvalidCard {
        try {
              thisCard.read () ;  // may throw InvalidCard exception
              pin = KeyPad.readPin () ; attempts = 1 ;
              while ( !thisCard.pin.equals (pin) & attempts < 4 )
                    {     pin = KeyPad.readPin () ;  attempts = attempts + 1 ;
                    }
                    if (!thisCard.pin.equals (pin))
                          throw new InvalidCard ("Bad PIN");
              thisBalance = thisCard.getBalance () ;
              do {  Screen.prompt (" Please select a service ") ;
                    service = Screen.touchKey () ;
                    switch (service) {
                          case Services.withdrawalWithReceipt:
                                      receiptRequired = true ;
```

PDL DISADVANTAGES

- **PDL may not be sufficiently expressive** to express the system functionality in an understandable way
- **Notation is only understandable to people with programming language knowledge**
- **The requirement may be taken as a design specification** rather than a model to help understand the system

Defining Non-Functional Requirements with Planguage

- To address the problem of ambiguous and incomplete nonfunctional requirements, consultant Tom Gilb (1988; 1997) has developed *Planguage*, a planning language with a rich set of keywords that permits precise statements of quality attributes and other project goals.

Defining Non-Functional Requirements with Planguage

- Each requirement receives a unique *tag*, or label.
- The *ambition* states the purpose or objective of the system that leads to this requirement.
- *Scale* defines the units of measurement
- *Meter* describes precisely how to make the measurements.

DEFINING NON-FUNCTIONAL REQUIREMENTS WITH PLANGUAGE

- The *must* criterion is the minimum acceptable achievement level. The requirement isn't satisfied unless every *must* condition is completely satisfied, so the *must* condition should be justifiable in business terms.
- The *plan* value is the nominal target
- The *wish* value represents the ideal outcome.

Non-Functional Requirements with Planguage

- **TAG** Performance.QueryResponseTime
- **AMBITION** Fast response time to database queries.
- **SCALE** Seconds of elapsed time between pressing the Enter key or clicking OK to submit a query and the beginning of the display of query results.
- **METER** Stopwatch testing performed on 250 test queries that represent a defined usage operational profile.
- **MUST** No more than 10 seconds.
- **PLAN** No more than three seconds.
- **WISH** No more than two seconds for all queries.

SUMMARY

- user and system requirements
- functional / non-functional requirements
- Software requirements may be organised

End of Lecture