Requirements Engineering

- The requirements for a system are simply a description of **what** the system should do.
  - A list of the functionality that it provides (functional requirements)
  - And other requirements that are not explicitly functional (non-functional requirements)
- It is an external view of the system and we are **not** concerned (yet) with **how** it works internally.
- The process of finding out, analysing and documenting these requirements is called Requirements Engineering.

Functional Requirements

- Functional requirements describe what functionality the system should have.
- For example:

The Patient Management System shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

1. On the last working day of each month, a summary of the drugs prescribed, their cost, and the prescribing clinics shall be generated.
2. The system shall automatically generate the report for printing after 17.30 on the last working day of the month.
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.
4. If drugs are available in different dose units (e.g., 10 mg, 20 mg) separate reports shall be created for each dose unit.
5. Access to all cost reports shall be restricted to authorized users listed on a management access control list.

Non-functional Requirements

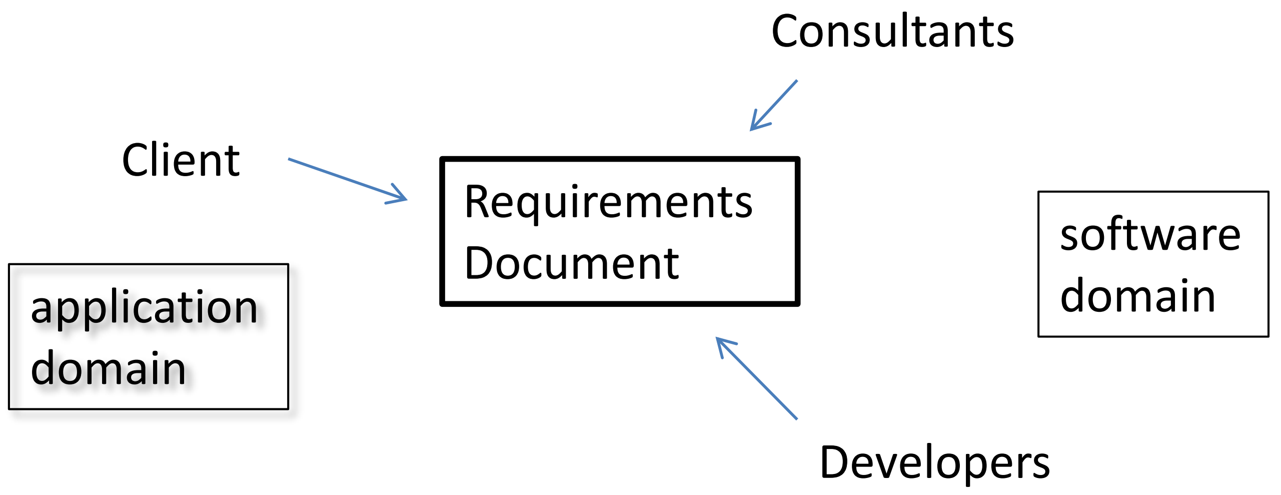
- Non-functional requirements are all other requirements that are not explicitly functional.
- For example:
  - Performance: The response time of the system should be less than 0.5 seconds.
  - Security: The system should not allow any unauthorised access.
- Non-functional requirements typically apply to the system as a whole.

Requirements Document

- The Requirements Document is the outcome of the requirements engineering process.
- It will list all of the agreed upon requirements (functional, non-functional) of the system.
- And it provides a prototype (model) of the system.
  - This details how the users will interact with the system.
  - Mock up User Interfaces and Use Cases.

Requirements Document

- The Requirements Document is used as a tool to enable clear communication between the Client, Consultants and Developers.



- So, everyone has a clear, concise understanding of what is being built.

Scope

- The Requirements Document defines the Scope of the project:
  - What functionality is included in this release of the system.
  - What is not included.
- It is very important to clearly agree on the Scope of the project with the Client.
- Do not be over ambitious.
  - Can always add more functionality in a later version of the system.

Requirements again

Functional Requirements

- Describe the functionality of the system.
- We have techniques for identifying these.

Non-functional Requirements

- Other requirements that are not explicitly functional.
- Usability
- Reliability
- Security/Privacy
- Performance
- Supportability
- Implementation requirements
- Interface requirements
- Operations requirements
- Packaging requirements
- Legal requirements

Common characteristics of Requirements

Incomplete
  - Often due to hidden assumptions on the part of the Client. They know things you don't, and don't know you don't know.

Contradictory
  - Need to decide which are more important and which can be sacrificed without too much damage.

Not well decomposed
  - One “requirement” may actually be lots of little requirements
  - These need to be fleshed out.

Unrealistic
  - Users want what isn't physically possible within the time of the project.
  - As a developer, always double your time estimate.

Characteristics of (good) Requirements

- Completeness - all necessary functionality is described by the requirements.
- Consistency - No two requirements contradict each other.
- Clarity - Requirement cannot be interpreted in two different ways.
- Correctness - The requirements describe the features of the system correctly.

Process of Requirements Elicitation

The Kick-off meeting

- Representatives from the Client (application domain)
- Representatives from your team
  - Consultants – high-level system details
  - Developers – low-level technical details
- Have a discussion on **what** it is the Client wants the system to do.
  - A whiteboard/flip chart is handy for drawing pictures.
  - Write the requirements on the whiteboard/flip chart for everyone to see and discuss.
  - A person to take detailed notes of what is said and agreed.

Process of Requirements Elicitation

- We use an example of a Patient Management System.
- Some of the high-level functional requirements are to:
  - Manage patient records.
  - Add a new patient record.
  - Be able to view and update patient records.
  - Generate reports from the patient records.

Process of Requirements Elicitation

- Identifying Actors
- Identifying Use Cases
- Identifying initial Data Objects
- Identifying Non-Functional Requirements
- Refining Requirements and Use Cases
  - through analysis and 2nd Client Meeting

Identifying Actors

- Actors represent external entities that interact with the system in some way.
  - e.g. A <u>Medical Receptionist</u> must be able to add new patient records.

Questions you can ask to identify actors:

- Which user groups use the system to perform their work?
- Which user groups execute the main functions?
- Which user groups perform secondary functions (e.g. maintenance)?
- Which pieces of external software interact with the system?

Identifying Use Cases

- The Use Cases implement the functional requirements.
- A Use Case describes how an Actor interacts with a piece of system functionality.
  - e.g. Requirement: A Medical Receptionist must be able to add new patient records.
- The Use Case has:
  - A name: Add Patient
  - A set of participating actors: Medical Receptionist
  - A flow of events (the text narrative): example of this later.

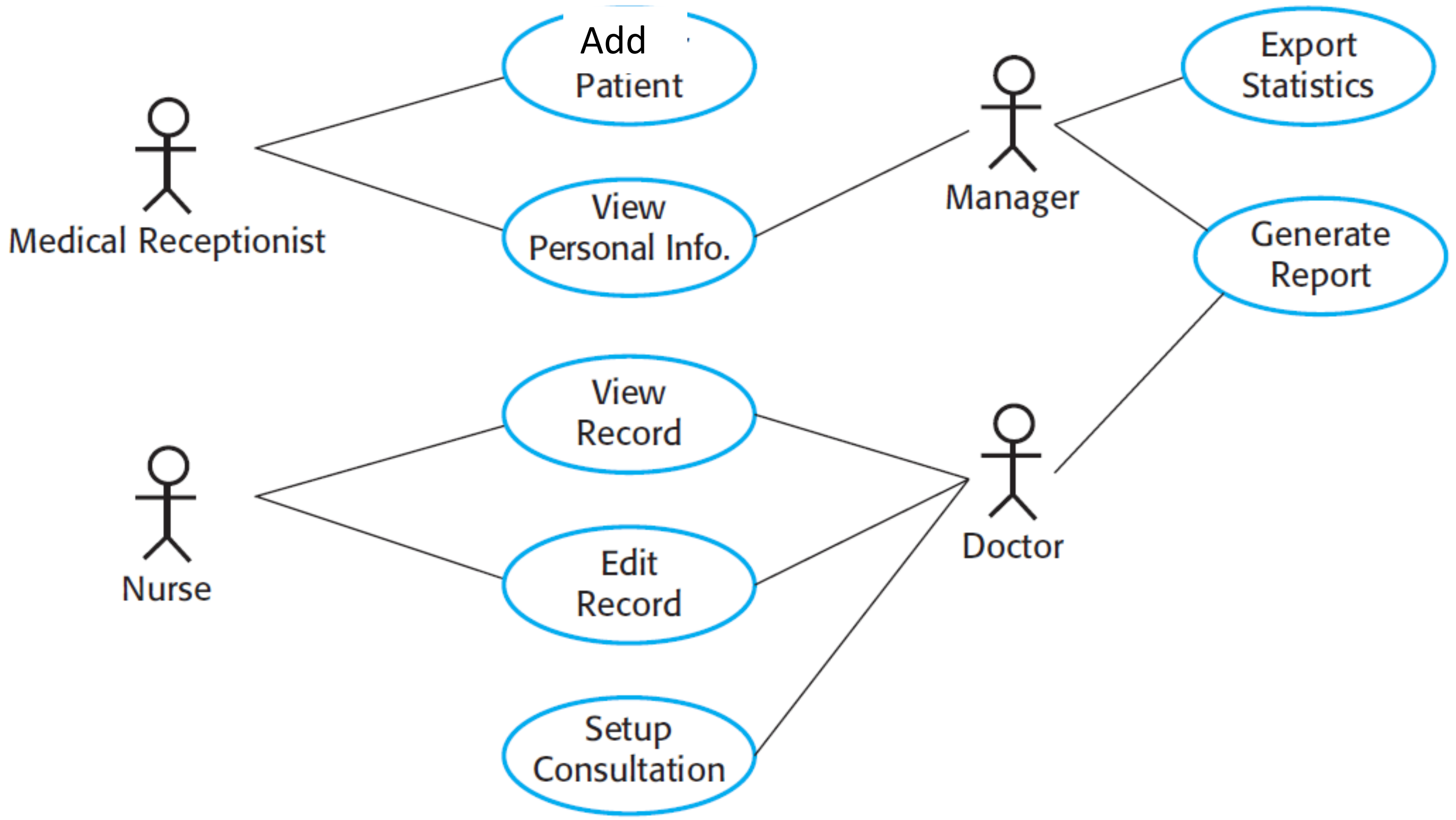
Questions to elicit Use Cases:

- What functionality does the Actor use the system for?
- What information does the Use Case need?

Scenarios

- There may be a number of scenarios for a particular use case.
- Each scenario represents a particular flow of events in the use case.
- For example, there may be several ways to add a new patient record to the system.

Example Actors and Use Cases



Gathering Data Objects

- We should also be gathering the data that relates to each of the Use Cases.
  - What are the inputs to the Use Case?
  - What are its output?
- These can be organised into Class Diagrams.
- For example, a Class Diagram defining the patient record.
- Here we are mapping out the Application Domain.

The Second Client Meeting

- Present the Requirements and the initial prototype of the system to the Client.
  - User Interface Mock Ups.
  - Uses Cases describing the system interactions.
- The run through of the system prototype will help in refining the Requirements and the system Use Cases.
- Refine the Requirements and Use Cases so they are complete, consistent, clear and correct.

Example Use Case Text Narrative: System Login

System Login	User	System
1.1	User enters Staff Id	
1.2	User enters Password and presses Login button	
1.3		System checks Staff Id and Password against authentication list.
1.4a		User is validated and system presents the Home screen dynamically tailored to their Role.
1.4b		User is not valid and system presents Invalid screen. 3 Attempts. Contact Administrator.

Questions that arise

- What Roles should the system support?
  - Doctor, Nurse, Medical Receptionist, Manager
- Can a person have more than one Role?
- How do we display the Home screen of a person with more than one Role?
  - A super set of data and functionality.
  - Or introduce a new Requirement that users can Switch Role when logged in.

Identifying non-functional requirements

A helpful checklist (though you may encounter things that don't fit this):

- Usability: How skilled are the users? What documentation will they receive? What systems are they used to?
- Reliability: How reliable does the system need to be? How much data can it lose? Uptime?
- Security/Privacy: What are the security considerations?
- Performance: How responsive need it be? What tasks (if any) are time critical? How many concurrent users do we need? How large is the typical data store? What is the worst acceptable latency?

Identifying non-functional requirements

- Supportability: Who maintains the system? What extensions do you foresee? Will it be moved to new hardware or software environments?
- Implementation: Constraints on hardware platform? Software?
- Interface: What existing systems need it interact with? How are data imported/exported? What standards need it comply with?
- Operation: Who manages the running system?
- Packaging: Who installs the system? How often?
- Legal: How should the system be licensed? Liability? Royalties?

Requirements Document

A typical outline for a Requirements Document:

- Introduction
  - Overview - Purpose of system
  - Scope
  - Objectives and success criteria
  - Definitions, abbreviations
  - References
- Current system
- Proposed System
  - Overview
  - Functional Requirements
  - Non-functional requirements
  - System prototype (models)
    - User interface mockups
    - Use cases (including text narratives)
    - Object model
    - Dynamic model