Software Process Requirements

Lecture 3

Requirements ..

Why are they important ?

The most common reasons for project failure are not technical.

Table 1.1 Reasons for project failure (Standish Group (1995))

**	Incomplete requirements	13.1%
**	Lack of user involvement	12.4%
	Lack of resources	10.6%
*	Unrealistic expectations	9.9%
	Lack of executive support	9.3%
**	Changing requirements/specifications	8.7%
	Lack of planning	8.1%
**	Didn't need it any longer	7.5%

Requirements ..

Why are they important ?

The most common reasons for project failure are not technical.

Table 1.2 Project success factors (Standish Group (1995))

**	User involvement	15.9%
	Management support	13.9%
**	Clear statement of requirements	13.0%
	Proper planning	9.6%
*	Realistic expectations	8.2%
	Smaller milestones	7.7%
	Competent staff	7.2%
*	Ownership	5.3%

Top Reasons for Difficulties in Software Projects as stated by a group of Project Managers based on their experience

- The REQs for the project are not explicit.
- Requirements changes are made/accepted without addressing the associated cost, schedule, and quality impacts.
- A requirements process is not used.
- No mechanism to reach agreement on the definition of the REQs and to manage the requirements through the project life cycle.
- The "real" customer needs are not defined.
- Known, familiar, proven methods, techniques, & tools are not utilized.
- The customer is not involved as a partner throughout the project life cycle.

Requirements ..

Why are they important ?

The most common reasons for project failure are not technical.

Table 1.3 Project success factors (Standish Group (2015).)

*	Executive management support	20%
**	User involvement	15%
	Optimization	15%
	Skilled resources	13%
	Project management expertise	12%
*	Agile process	10%
**	Clear business objectives	6%
*	Emotional maturity	5%
	Execution	3%
	Tools and infrastructure	1%

What is a **Requirement?**

" .. A requirement is a necessary attribute in a system, a statement that identifies a capability, characteristic, or quality factor of a system for it to have value and utility to a customer or user."

" .. A statement that identifies a product or process operational, functional, or design characteristic or constraint, which is unambiguous, testable or measurable, and necessary for product or process acceptability (by consumers or internal quality assurance guidelines)." *~IEEE-STD-1220-1998 (IEEE 1998)*

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.

This is inevitable as requirements may serve a dual function:

- ✓ May be the basis for a bid for a contract - therefore must be open to interpretation.
- ✓ May be the basis for the contract itself - therefore must be defined in detail.
- ✓ Both these statements may be called requirements.

Requirements ..

Why are they important ?

" .. Requirements are important because they provide the basis for all the development work that follows. Once the requirements are set, developers initiate the other technical work: system design, development, testing, implementation, and operation."

"Without requirements or design, programming is the art of adding bugs to an empty text file."

- Louis Srygley

Requirements Completeness, .. & .. Consistency

- Theoretically, requirements should be both **complete** & **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.*
- Actually, because of system and environmental complexity, it is impossible to produce a complete and consistent requirements document.

Other Criteria of Good **Requirements**

Table 1.1 Criteria of a Good Requirement

Each Individual Requirement Should Be
Necessary: If the system can meet prioritized real needs without the requirement, it isn't necessary.
Feasible: The requirement is doable and can be accomplished within budget and schedule.
Correct: The facts related to the requirement are accurate, and it is technically and legally possible.
Concise: The requirement is stated simply.
Unambiguous: The requirement can be interpreted in only one way.
Complete: All conditions under which the requirement applies are stated, and it expresses a whole idea or statement.
Consistent: It is not in conflict with other requirements.
Verifiable: Implementation of the requirement in the system can be proved.
Traceable: The source of the requirement can be traced, and it can be tracked throughout the system (e.g., to the design, code, test, and documentation).
Allocated: The requirement is assigned to a component of the designed system.
Design independent: It does not pose a specific implementation solution.
Nonredundant: It is not a duplicate requirement.
Written using the standard construct: The requirement is stated as an imperative using "shall."
Assigned a unique identifier: Each requirement shall have a unique identifying number.
Devoid of escape clauses: Language should not include such phrases as "if," "when," "but," "except," "unless," and "although." Language should not be speculative or general (i.e., avoid wording such as "usually," "generally," "often," "normally," and "typically").

Requirements Abstraction

" .. If a company wishes to let a contract for a large software development project, it must define its needs in a sufficiently abstract way that a solution is not pre-defined. The requirements must be written so that several contractors can bid for the contract, offering, perhaps, different ways of meeting the client organization's needs. Once a contract has been awarded, the contractor must write a system definition for the client in more detail so that the client understands and can validate what the software will do. Both documents may be called the requirements document for the system."

Views of Requirements Classifications
[i.e., Types of Requirements]

Numerous ways to organize requirements types exit.

Common Requirements Classifications include:

- **Business REQs:** The essential activities of an enterprise *(the reason for developing the software in the first place)*.
- **User REQs:** Describes for the customers/users the services the system provides and its operational constraints.
- **Functional REQs:** Describes what the software should do.
- **Non-Functional REQs:** Describes software properties.
- **Domain REQs:** Describes constraints on the system from the domain of operation.
- .. Etc.

Requirements Engineering .. *Why?*

" .. It's not just a simple matter of writing down what the customer says (s)he wants."

- A fundamental problem in business is that **requirements are inherently dynamic**; *they will change over time as our understanding of the problem we are trying to solve changes*.
- The importance of good requirements and the underlying dynamic nature of the process mean that we must be **(1) as accurate as possible**, and yet be **(2) flexible** *(i.e., we have a process for developing requirements & accommodating changed requirements as we clarify the real requirements of customers).*

Requirements Engineering .. *Why?*

" .. Too often, there is a tendency to want to start what is often referred to as "the real work" *(developing, or programming, the software)* too quickly."

- Many customers and project managers (PMs) seem to believe that actual programming work ("coding") indicates that progress is being made.
- According to industry experience, insufficient time and effort are spent on the requirements-related activities associated with system development.
- Industry experience confirms that a better approach is to invest more time in requirements gathering, analysis, and management activities.

Requirements Engineering .. *Why?*

Stated vs. Real Requirements

- **Stated Requirements** are provided by a user/customer at the beginning of a software development effort.
- **Real Requirements** reveal the verified needs for a particular system or capability *(i.e., some real requirements may be identified that the customer and users omitted in the stated requirements)*. Actually, identifying omitted requirements is a key task of the Requirements Analysts (RA).

User REQs *versus* System REQs

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.

for a **Mental Health Care Patient Management System (MHC-PMS);**
A system used to maintain records of people receiving care for mental health problems.

User REQs *versus* System REQs

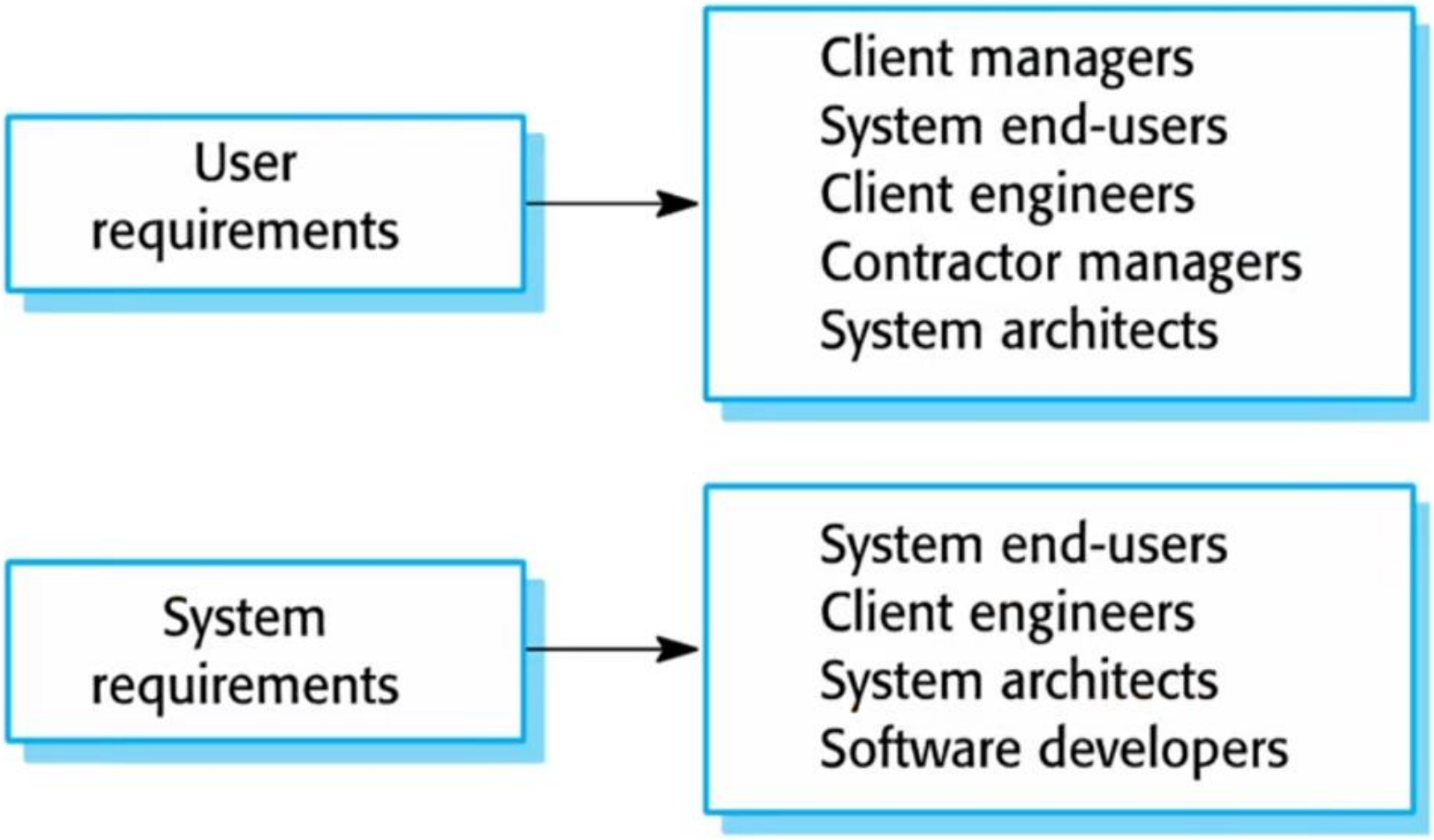
User requirements definition

1. The Mentcare system 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 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. 10mg, 20mg, etc) separate reports shall be created for each dose unit.
1.5 Access to drug cost reports shall be restricted to authorized users as listed on a management access control list.

Readers of the Different Types of Requirements Specification



Requirements Engineering Process

Generic Activities

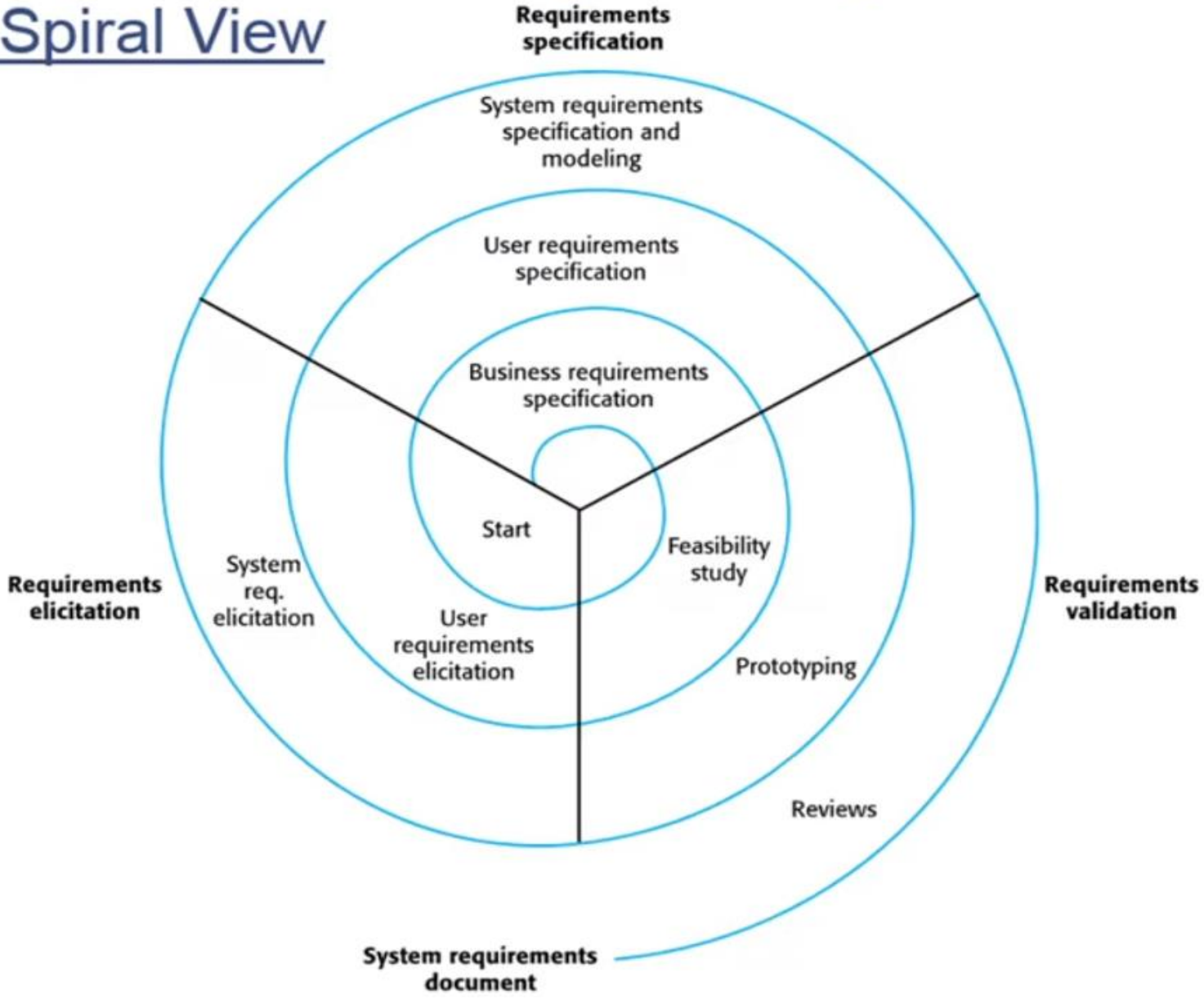
The processes used for Requirements Engineering (RE) vary widely depending on the application domain, the people involved and the organisation developing the requirements. But, there are a number of **generic activities** common to all processes:

- **Requirements elicitation;**
- **Requirements analysis;**
- **Requirements validation;**
- **Requirements management.**

In practice, RE is an iterative activity in which these processes are interleaved.

Requirements Engineering Process

A Spiral View



Requirements Engineering Processes

More detailed Requirements-Related Activities

Start by **Identifying the Stakeholders** *(i.e., anyone who has an interest in the system, or in the system's possession of particular capabilities/qualities)*. Then **Requirements Elicitation** is about gaining an understanding of the customers' & users' needs and discovering their expectations. Followed by stating the REQs in simple sentences and providing them as a set (**Identifying Requirements**), ensuring that they describe the customer's real needs and are in a form that can be understood/used by developers of the system (**Clarifying & Restating the Requirements**), ensuring that they are well defined *(in a way that means the same thing to all the stakeholders)* and that they conform to the criteria of a good REQ (**Analyzing the Requirements**).

Requirements Engineering Processes

More detailed Requirements-Related Activities

Final activities include:

Tracking Requirements by tracing where in the system each REQ is satisfied *(so that we can verify that each REQ is being addressed)*.

Testing & Verifying Requirements; checking the REQs, designs, code, test-plans, & system products to ensure that all REQs are met.

Validating Requirements *(confirming that the real REQs are implemented in the delivered system)*.

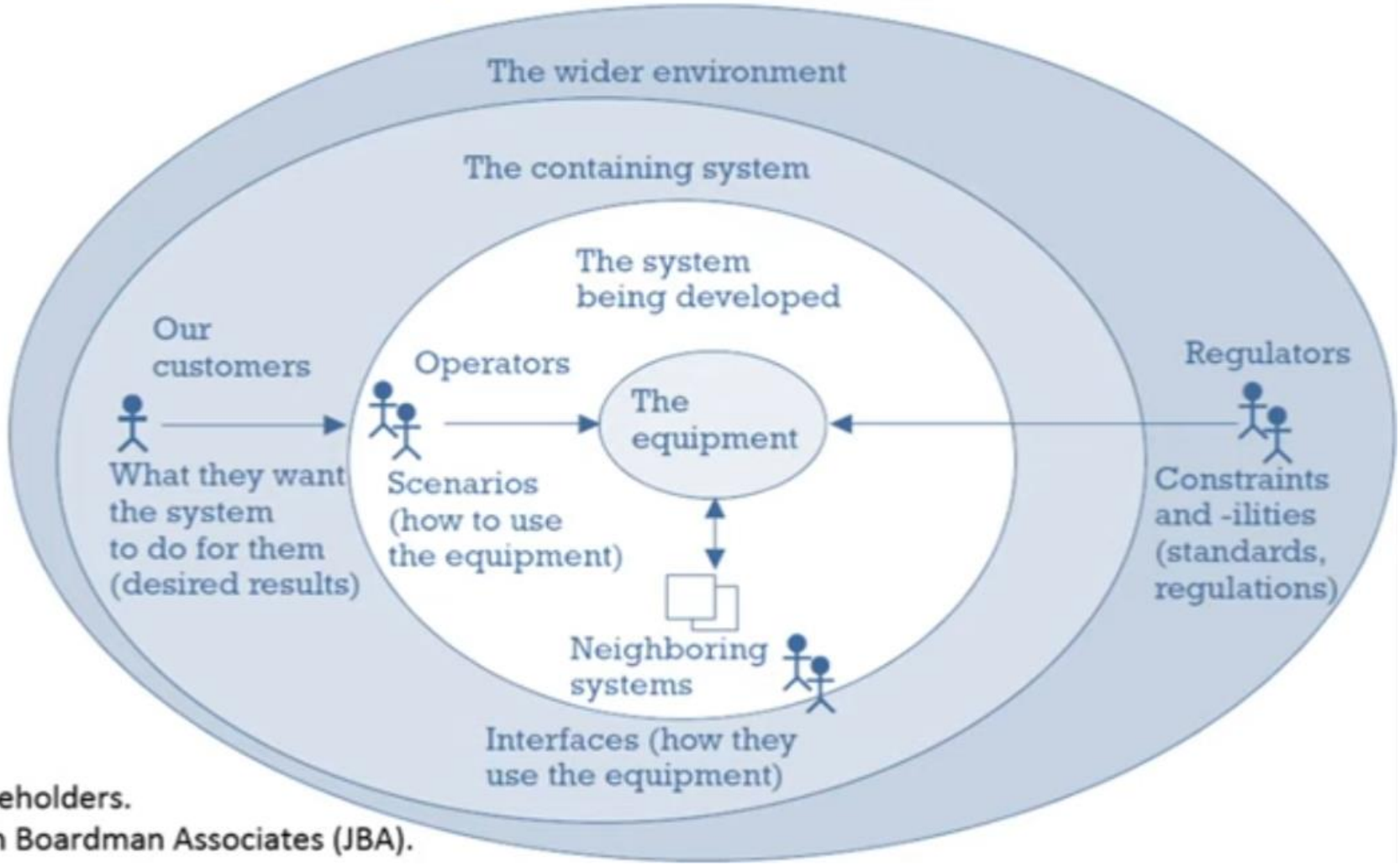
Managing Requirements *(adding, deleting, and modifying REQs during all the phases of system design, development, integration, testing, deployment, and operation)*.

System Stakeholders

A stakeholder is anyone *(person or organization)* who has a legitimate interest in the project and anyone who will be affected or influenced by the proposed system in some way.

- Think of **customers / system-owners / investors** *(those who are paying for the work)*, **end-users / system-users** *(people who will use the system)*, **government & public opinion**, **advisors / regulatory-authorities** *(such as legal experts or regulators who have relevant information about the requirements)*, **system managers**, **maintenance & service staff**, project groups that are involved in developing the system *(e.g., **systems engineers**, **software engineers**, **training personnel**, **testers**, etc.)*.

System Stakeholders
How to identify them ..?



Roles of stakeholders.
(*Source:* John Boardman Associates (JBA).

for a **Mental Health Care Patient Management System (MHC-PMS);**
A system used to maintain records of people receiving care for mental health problems.

System Stakeholders

How to identify them ..? An Example

- **Patients** whose information is recorded in the system.
- **Doctors** who are responsible for assessing and treating patients.
- **Nurses** who coordinate the consultations with doctors and administer some treatments.
- **Medical Receptionists** who manage patients' appointments.
- **IT Staff** who are responsible for installing and maintaining the system.
- A **Medical Ethics Manager** who must ensure that the system meets current ethical guidelines for patient care.
- **Health Care Managers** who obtain management information from the system.
- **Medical Records Staff** who are responsible for ensuring that system information can be maintained and preserved, and that record keeping procedures have been properly implemented.