Process Activities

Requirement Gathering

WHY REQUIREMENTS GATHERING & ANALYSIS?

"Measure twice, cut once"

"Look before you leap"

- What do these sayings have in common? They're about Thinking before Acting.

Why?

- Because it Saves time, money and effort.

WHAT IS REQUIREMENTS GATHERING?

- A Process of Collecting the user needs to Solve a problem or issues and Achieve an objective.

- An important Phase in a project life cycle If the Requirements Gathering is not done properly, All below phases get incomplete.

- No matter how good the Design is Requirements are the foundation of any Project If you don’t know where you’re going, any road will take you there!

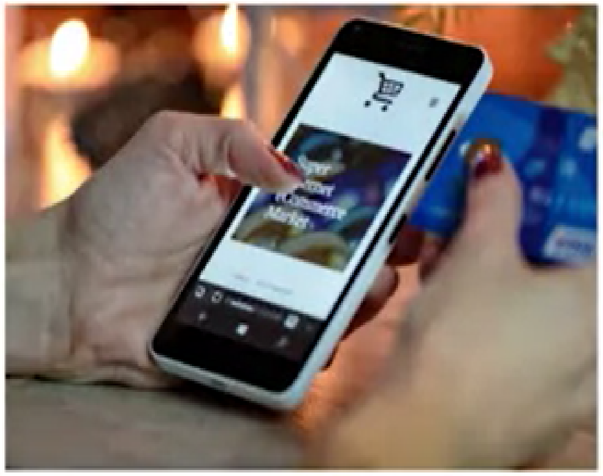
WHAT IS A REQUIREMENT?

- A requirement is a capability that a product must possess to satisfy a customer need or objective.

- They help a project team define,
  - Goals
  - Scope of the work

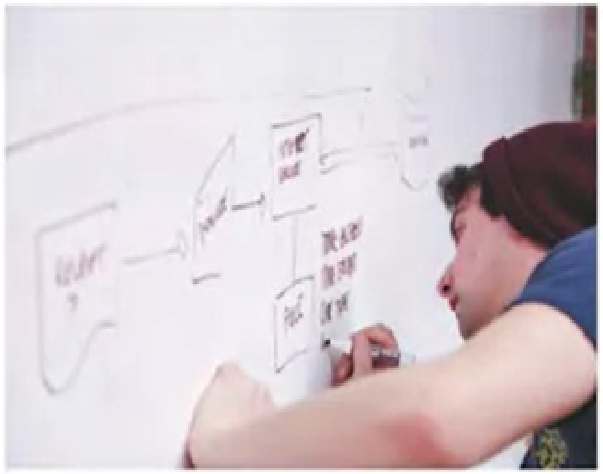
- They provide objective ways to measure the project's success

Requirement vs Specification



Users

Developers



Requirement vs Specification

- Requirements are user needs

- System spec is the usually more precise or constraining statement of how the system will meet the user requirements.

Example

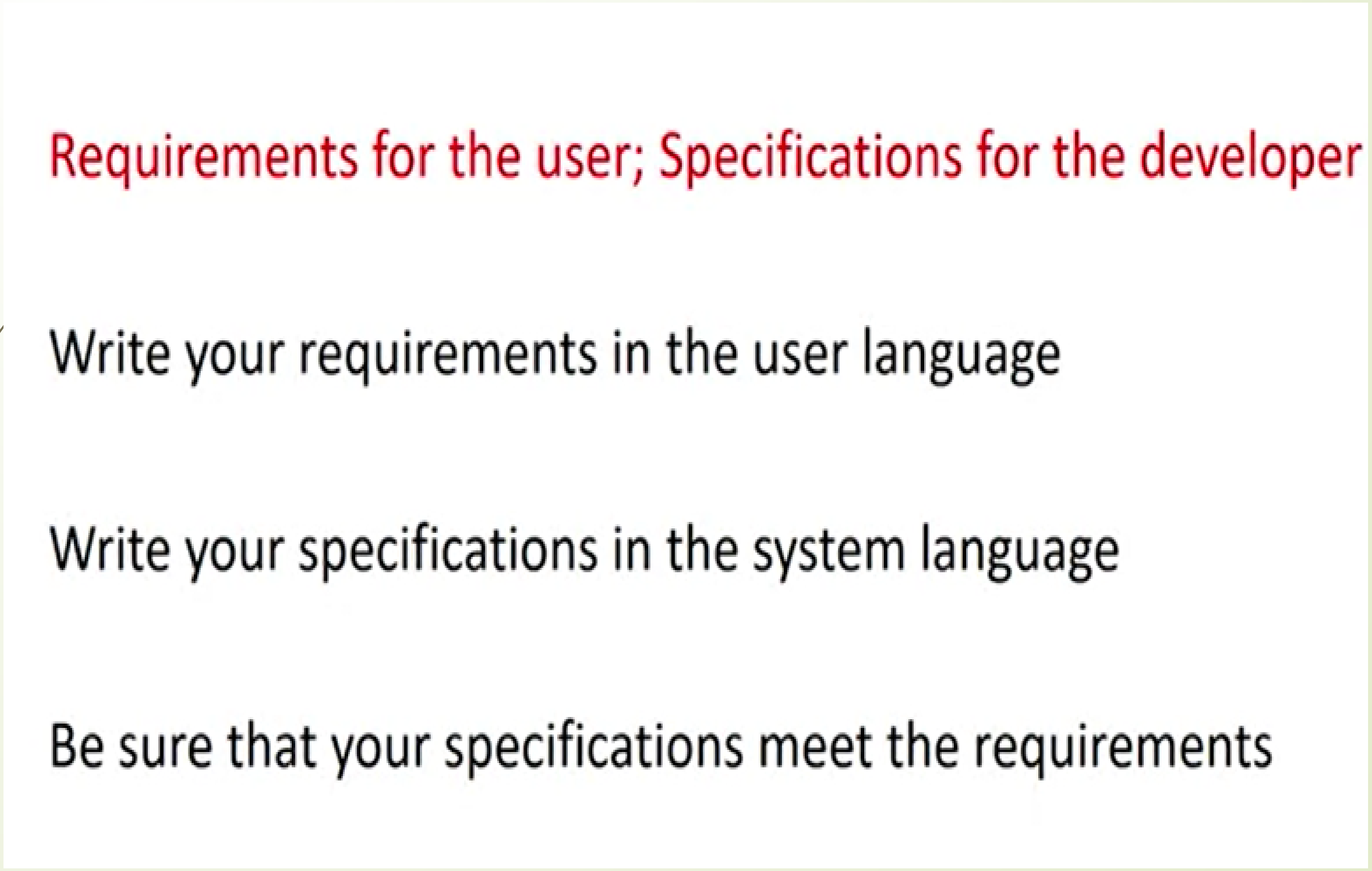
User requirements

- System must be secured.

System Specification.

- User must be registered in system before accessing the system.
- Login should be provided.
- User can’t access data of other users account.
- Etc.

Recommendations



Requirements for the user; Specifications for the developer

Write your requirements in the user language

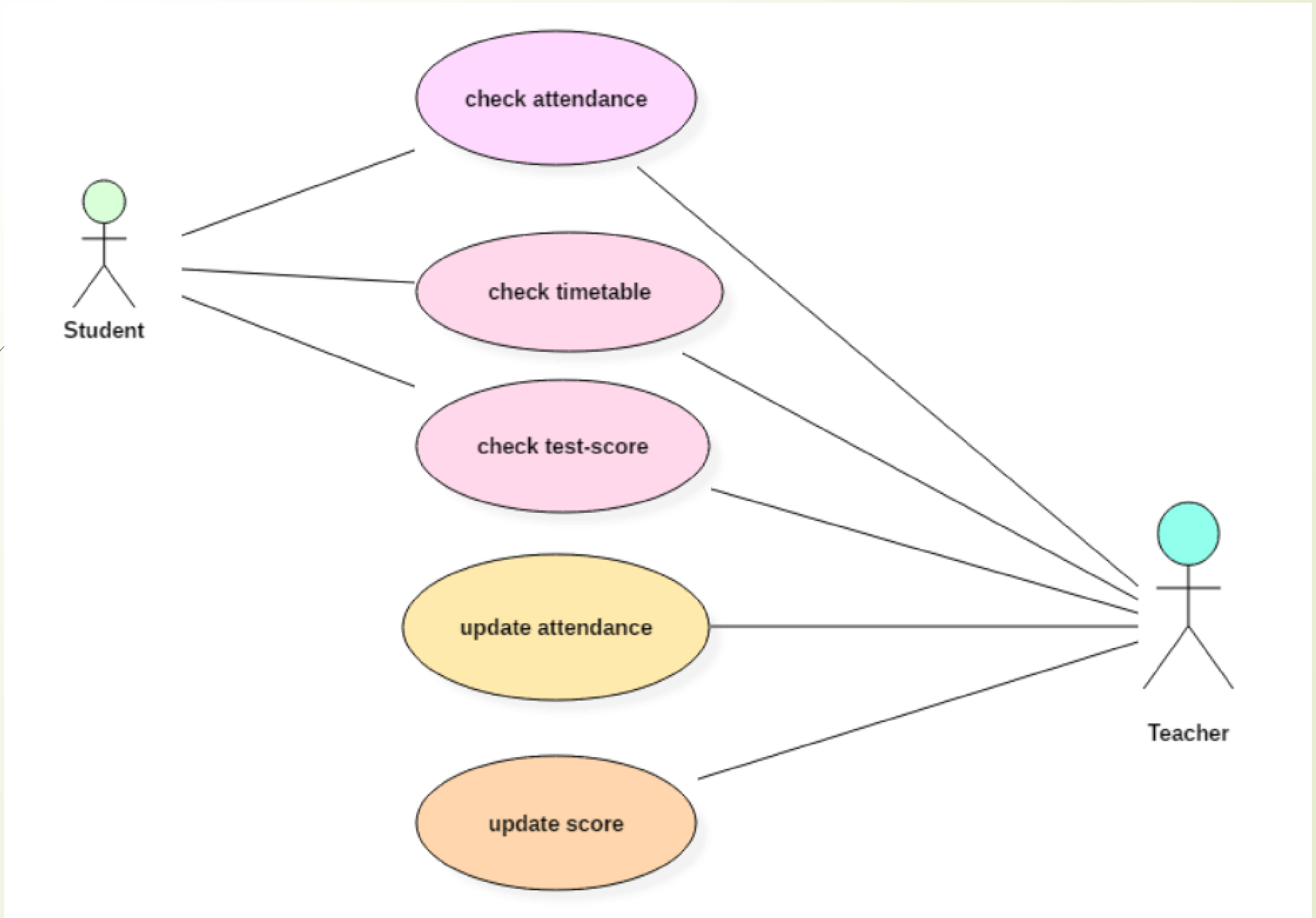
Write your specifications in the system language

Be sure that your specifications meet the requirements

System Requirements

- Whitney High School in Cerritos California wants to design a Student Management System on Initial level we have two users student and a teacher. The system represent the specific functionality of a student management system. A student can perform only these interactions with the system he can check attendance, timetable as well as test marks on the application or a system. A teacher can interact with all the functionalities of the system. Teacher can also update the attendance of a student and marks of the student. These interactions of both student and a teacher actor together sums up the entire student management application.

System Specifications

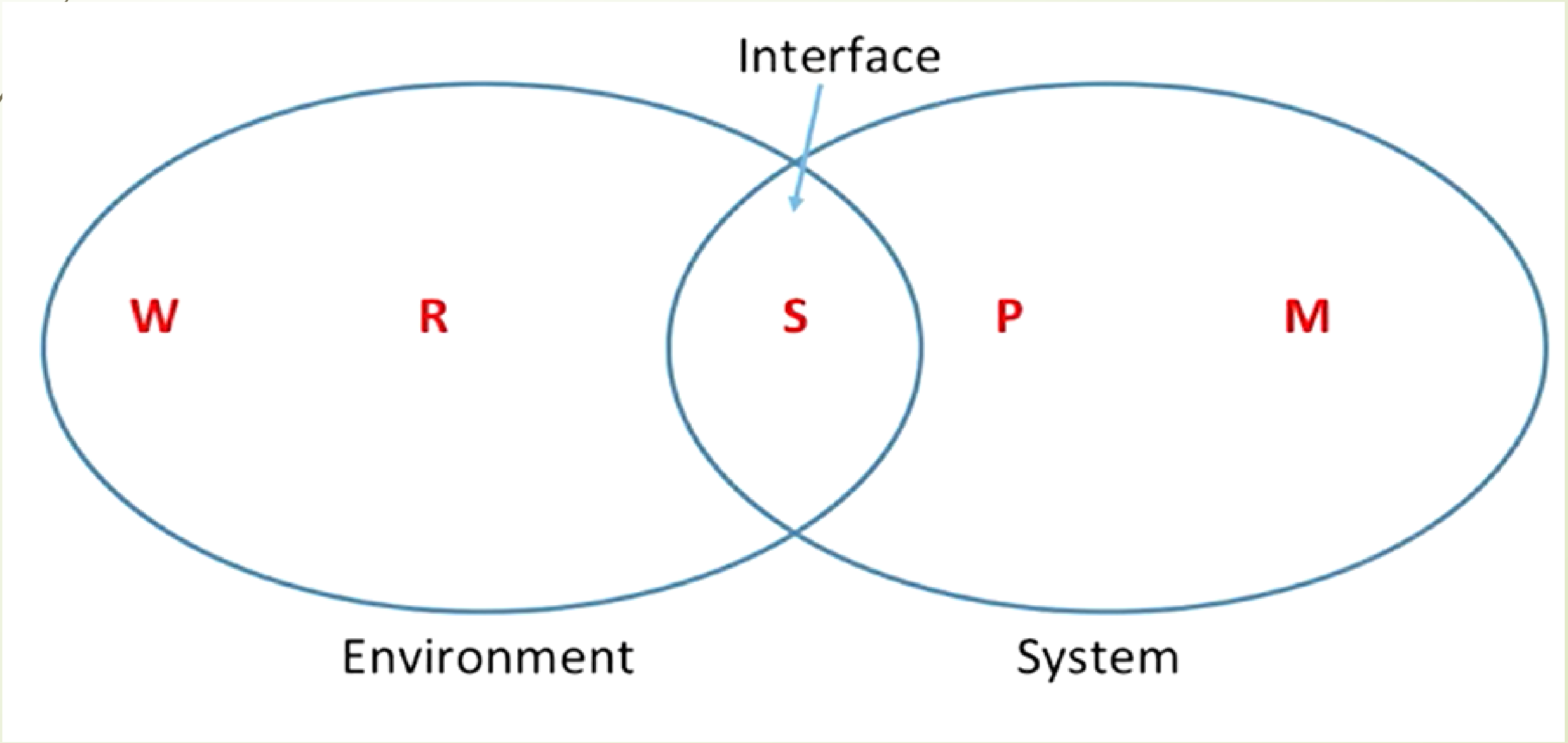


UML UseCase Diagram

WRSPM Reference model or The World Machine Model

- *Capturing the right thing* .

The **WRSPM** model helps define the problem better. Not only do we take into account the program we are building, we also consider things like the environment, and the actual hardware.

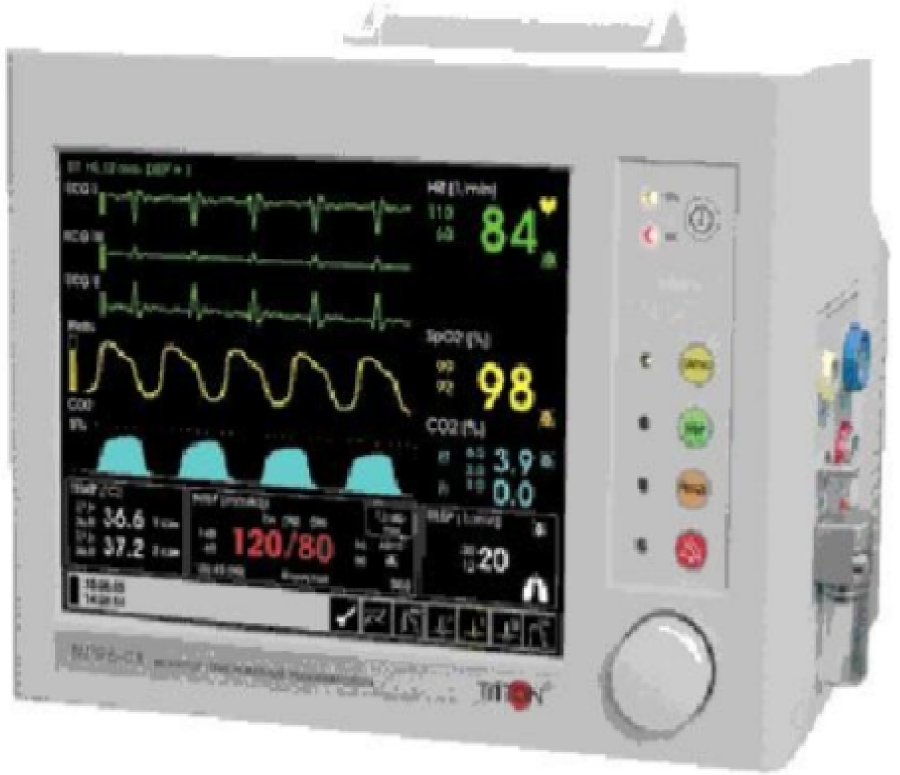


WRSPM Reference model or The World Machine Model

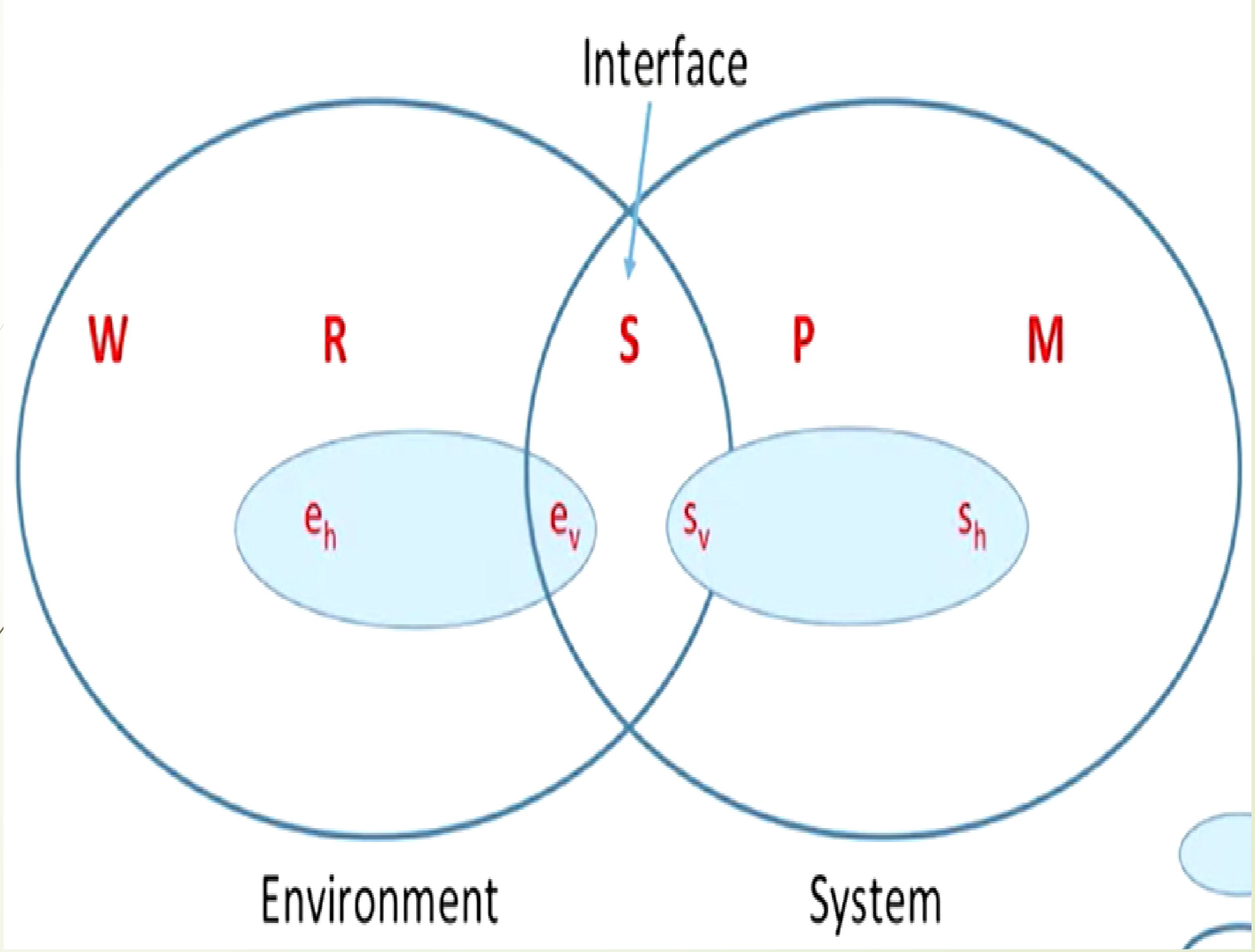
- **World** - The world assumptions which are used to develop the system. (Will there be internet? Electricity? What is the speed of the internet?)
- **Requirements-** Defining the problem at hand and how a solution to that problem should operate.
- **Specifications** - Defining the technical requirements of the system. Here we are linking together the idea of the solution, to the system itself.
- **Program -** The program and code itself.
- **Machine -** The physical hardware needed for the software. Servers, RAM, CPU, etc

Example Patient Monitoring

- **Desire**
  - A warning system that notifies a nurse if the patients heart stops
- **“Real” Requirement**
  - When the patient’s heart stops, a nurse shall be notified
- **“System” Requirement**
  - When the sound from the sensor (microphone taped over the heart) falls below a certain threshold, the alarm shall be actuated

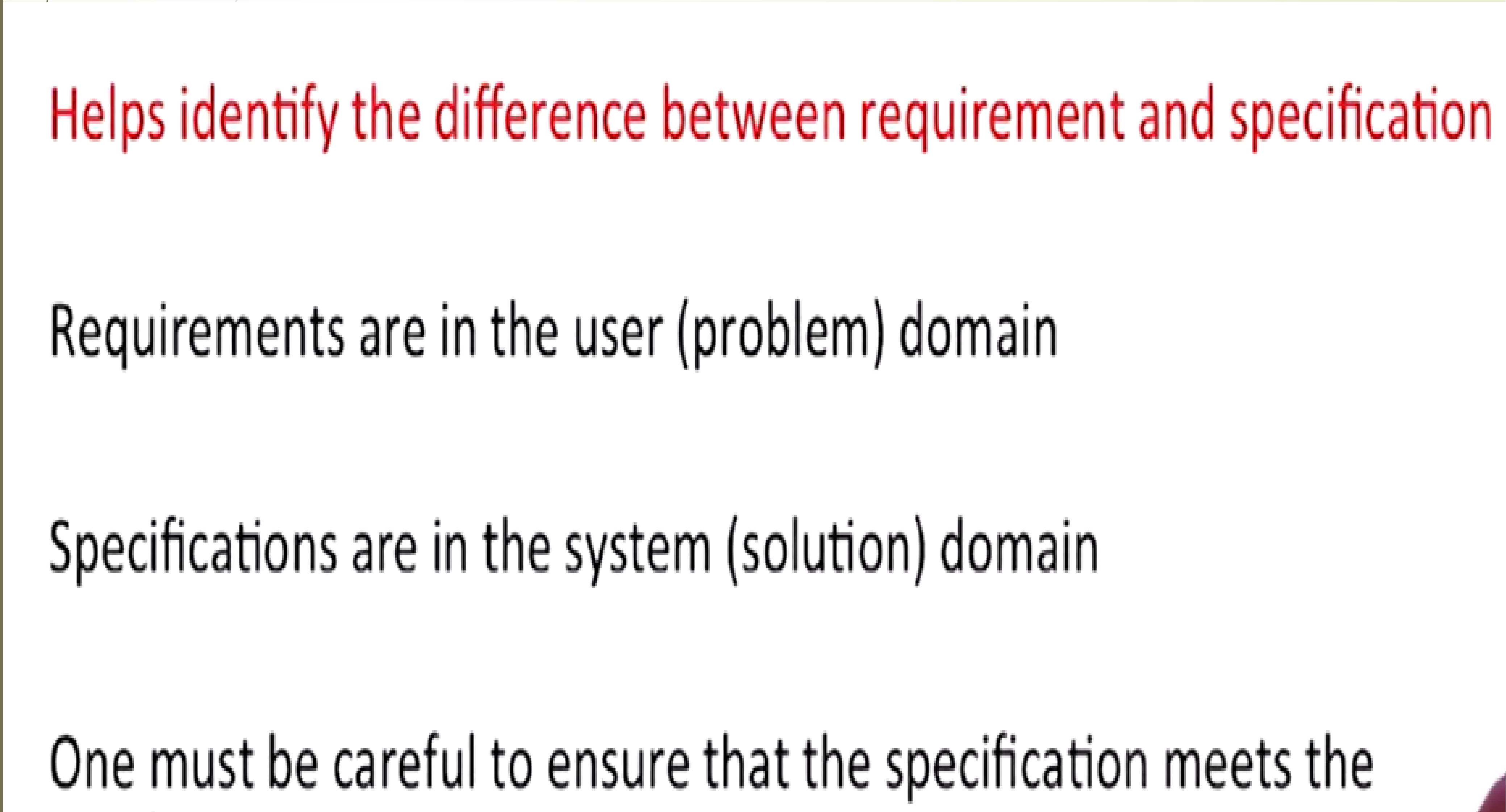


For Illustration Only



e_h: the nurse and the heartbeat of the patient.
e_v: sounds from the patient's chest.
s_v: the buzzer at the nurse's station.
s_h: internal representation of data from the sensor.

WRSPM Model



Helps identify the difference between requirement and specification

Requirements are in the user (problem) domain

Specifications are in the system (solution) domain

One must be careful to ensure that the specification meets the

Design

Introduction

- Software design is a process to transform user requirements into some suitable form, which helps the programmer in software coding and implementation.
- For assessing user requirements, an SRS (Software Requirement Specification) document is created whereas for coding and implementation, there is a need of more specific and detailed requirements in software terms.
- The output of this process can directly be used into implementation in programming languages.

Modularization

- Modularization is a technique to divide a software system into multiple discrete and independent modules, which are expected to be capable of carrying out task(s) independently.
- These modules may work as basic constructs for the entire software. Designers tend to design modules such that they can be executed and/or compiled separately and independently.
- Modular design unintentionally follows the rules of ‘divide and conquer’ problem-solving strategy this is because there are many other benefits attached with the modular design of a software.

Principles of Modularity

- Complex problem must be broken down in small parts.
- Three primary goals,
  - Decomposability
  - Composability
  - Ease of understanding

Aspects of Modularity

- Coupling
- Cohesion
- Information Hiding and Data Encapsulation

Coupling

- Coupling is a measure that defines the level of inter-dependability among modules of a program.

- The lower the coupling, the better the program.

Tight Coupling

- **Content Coupling** : Suppose module A directly rely on local data members of module B.
- **Common Coupling**: Suppose module A and B both rely on some global data or variable.
- **External Coupling**: Suppose A and B both rely on some protocol or standard.

Medium Coupling

- **Control Coupling:** Suppose module A decides the function of the other module B or changes its logical flow of execution.
- **Data Structure Coupling** : Suppose module A and B rely on same composite data structure and change of data structure in one module affect other module.

Loose Coupling

- **Data Coupling**: Only parameters are shared between 2 modules.
- **Message Coupling**: No centralize communication between modules. Only message passing can be possible between modules.
- **No Coupling**: No interaction among modules.

Cohesion

- Cohesion is a measure that defines the degree of intra-dependability within elements of a module.
- The greater the cohesion, the better is the program design.
- Implements a single logical entity or function.

Weak Cohesion

- **Co-incidental cohesion:** It is unplanned and random cohesion, which might be the result of breaking the program into smaller modules for the sake of modularization.
- **Temporal Cohesion:** When elements of module are organized such that they are processed at a similar point in time, it is called temporal cohesion.
- **Procedural cohesion -** When elements of module are grouped together, which are executed sequentially in order to perform a task, it is called procedural cohesion.
- **Logical cohesion -** When logically categorized elements are put together into a module, it is called logical cohesion.

Medium Cohesion

- **Communicational cohesion -** When elements of module are grouped together, which are executed sequentially and work on same data (information), it is called communicational cohesion.

- **Sequential cohesion -** When elements of module are grouped because the output of one element serves as input to another and so on, it is called sequential cohesion.

Strong Cohesion

- **Functional Cohesion**: It is considered to be the highest degree of cohesion, and it is highly expected. Elements of module in functional cohesion are grouped because they all contribute to a single well-defined function. It can also be reused.

Information Hiding

- Hide the complexity in “black box”.
- How to use system is your concern not how the system provide this functionality.
- E.g.: If you want a system which can sort an array, You are not concerned the type of sorting algorithm used in the system.