CSc 131 – **Computer** Requirements engineering

Agenda

- Class Quiz (This week)
- Requirements Engineering

Credits

Software Engineering: A Practioner's Approach. Roger S. Pressman. McGraw Hill Text; ISBN: 9780078022128; 8th edition, 2014

Software Engineering. Ian Sommerville. Addison-Wesley Pub Co; ISBN: ISBN-13: 978-0-13-703515-1, 9th edition, 2010

Jill Seaman at Texas State University

Early course CSC 230 Spring 2015

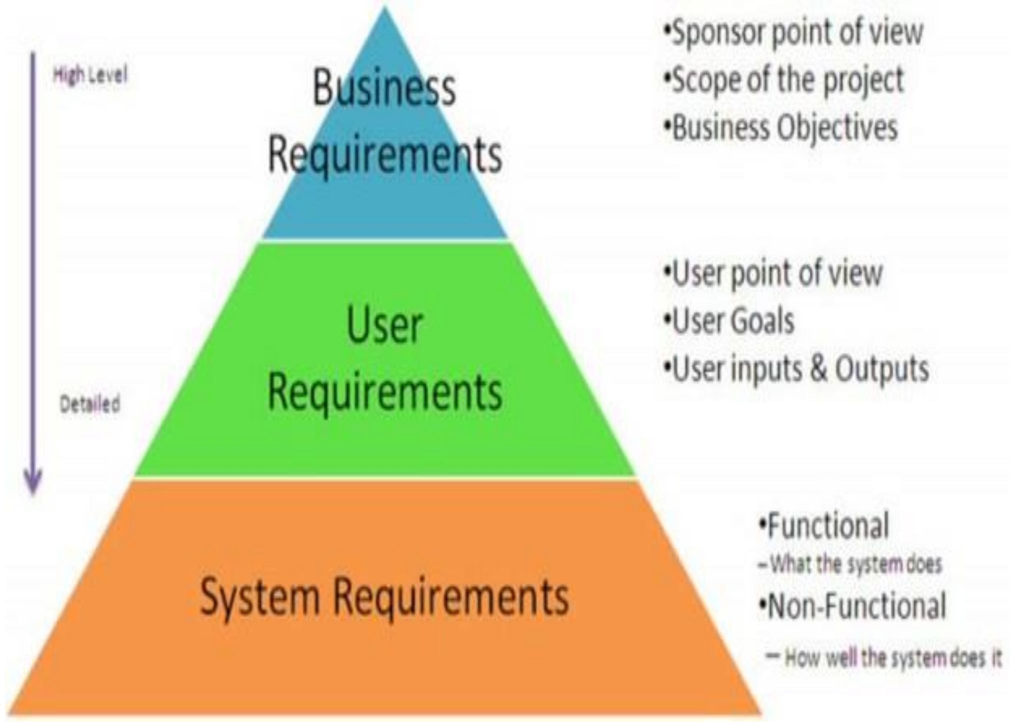
I. What are requirements?

- Sommerville:
  The descriptions of what the system should do:
  - the services that the **customer** requires
  - the constraints on its operation
- IEEE standard glossary of software engineering terminology
  - A condition or capability needed by a **user** to solve a problem or achieve an objective.
- (Wiegers, *Software Requirements 2*): A property that a system must have to provide value to a **stakeholder**

“the system” = the software system to be developed

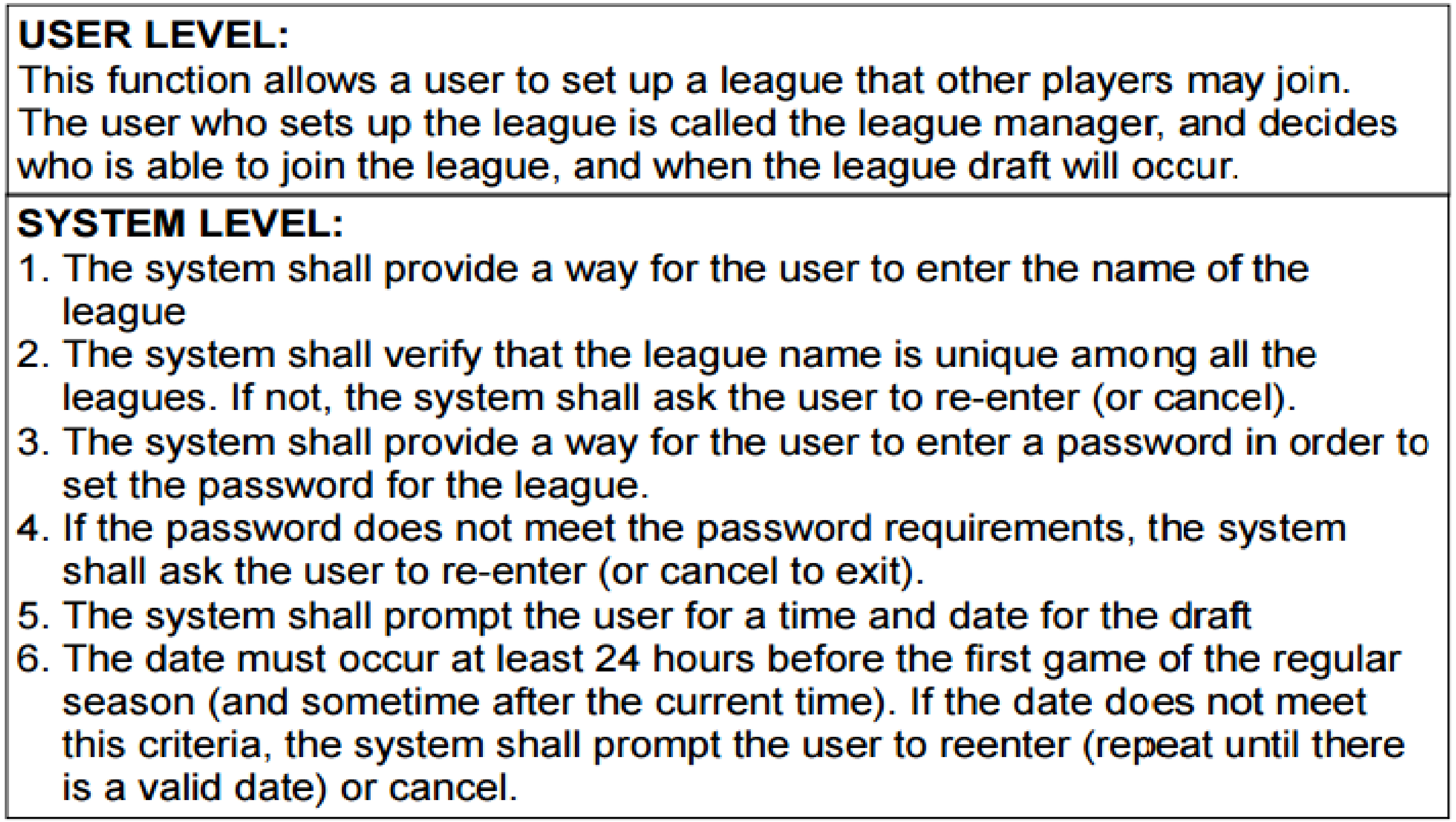
Levels of requirements specification
(in order of increasing detail)

- Business Requirements
  - High level goals of the stakeholders for the system
  - Provides vision and scope
- User requirements
  - Tasks the **users** must be able to perform with the system.
  - Expressed in natural language and diagrams
- System requirements
  - The system's functions, services and constraints described in even more detail.
  - Exactly what the **developers** must implement
    - Must have enough detail for the developers to know how to implement them.



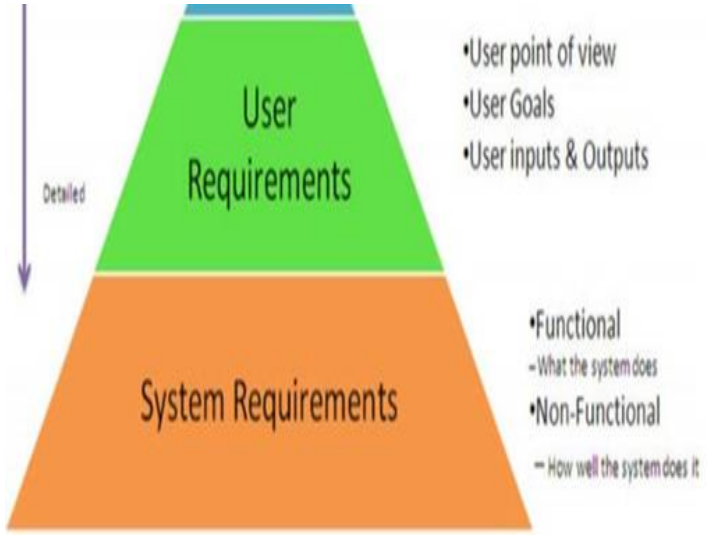
Example: User and system level requirements

Task: Set up a league for a Fantasy Football website



USER LEVEL:
This function allows a user to set up a league that other players may join. The user who sets up the league is called the league manager, and decides who is able to join the league, and when the league draft will occur.

SYSTEM LEVEL:
1. The system shall provide a way for the user to enter the name of the league
2. The system shall verify that the league name is unique among all the leagues. If not, the system shall ask the user to re-enter (or cancel).
3. The system shall provide a way for the user to enter a password in order to set the password for the league.
4. If the password does not meet the password requirements, the system shall ask the user to re-enter (or cancel to exit).
5. The system shall prompt the user for a time and date for the draft
6. The date must occur at least 24 hours before the first game of the regular season (and sometime after the current time). If the date does not meet this criteria, the system shall prompt the user to reenter (repeat until there is a valid date) or cancel.



Note: “shall” keyword usage in System Level as requirement text

Functional vs. non-functional requirements

- Functional requirements
  - Specific functions or services the system must provide.
  - How the system reacts to certain inputs (behavior)
  - Software functionality that the developers must build into the product to enable users to accomplish their tasks.
- Non-functional requirements
  - Constraints on the functions and services offered by the system.
  - These often apply to the system as a whole rather than individual features or services.
  - **How** the system must function.
  - Example: performance, security, or availability requirements.

Functional and Non-functional Requirements - Example

An example of a functional requirement would be that a system must send a an email whenever a certain condition is met (e.g. an order is placed, a customer signs up, etc).

A related non-functional requirement for the system may be that emails should be sent with a latency of no greater than 12 hours from such an activity.

Example: Functional requirements (user level)

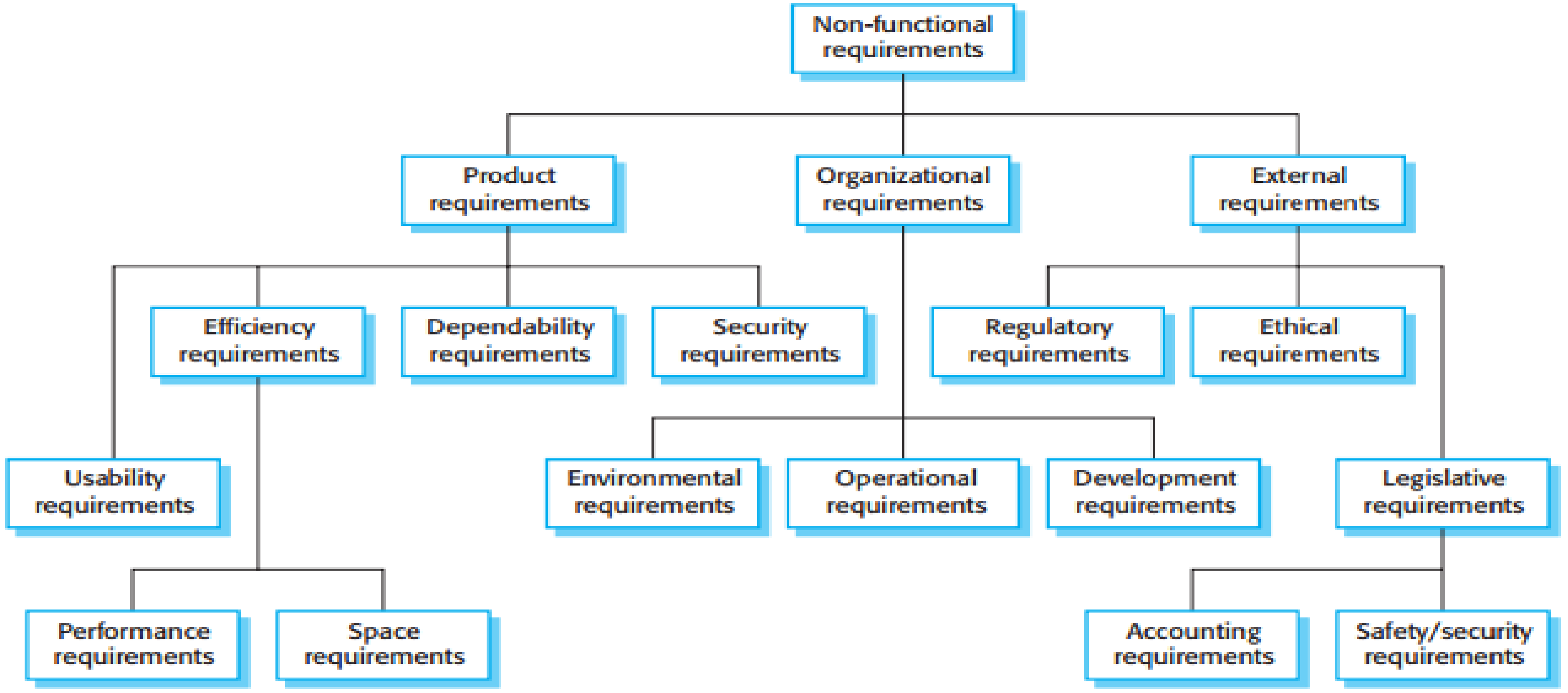
for a system used to maintain information about patients receiving treatment for mental health problems.

1. The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.
2. A user shall be able to search the appointments lists for all clinics.
3. Each staff member using the system shall be uniquely identified by his or her 8-digit employee number.

Different types of Non-functional requirements

- Performance and space goals
- Descriptions of quality attributes, including:
  - reliability
  - security
  - usability
- Design and implementation constraints:
  - must run on certain platform or operating system
  - must be written in a certain programming language

Sources of non-functional requirements



Examples of nonfunctional requirements

Product requirement
The system 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 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.

Consequences of Non-functional requirements

- Non-functional requirements may be more critical to the success of the project than functional requirements.
- A single non-functional requirement, such as a security requirement, may generate a number of related functional requirements that define required system services.
- A non-functional requirement may impose restrictions on some functional requirements.
- Implementation of a non-functional requirement may influence choices of software architecture.

Characteristics of excellent requirements

- **Correct**
  - Each requirement must reflect actual needs of the stakeholders (no gold plating).
- **Unambiguous**
  - Each requirement must have only one possible interpretation.
- **Complete**
  - All services required by the users must be defined (no requirements are missing)
  - Each requirement must fully describe the functionality to be delivered.
    - It must contain all the information necessary for the developer to design and implement that bit of functionality.
- **Consistent**
  - There should be no conflicts or contradictions among the requirements
- **Verifiable**
  - Each requirement must be written in a way so that the completed system could be tested against it.

Ambiguous requirements

- Capable of being interpreted in different ways by different people (it has more than one interpretation).
- Consider this functional requirement from a previous example:

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

  - User intention – given a name as input, search across all appointments **in all clinics**;
  - Developer interpretation – given a name and a clinic as inputs, search in **the individual clinic only**.

Verifiable requirements

- [non-verifiable] The system should be easy to use by medical staff and should be organized in such a way that user errors are minimized.
  - problems: “easy to use” cannot be measured or tested, and how many errors are acceptable?

- [verifiable] Medical staff shall be able to use all the system functions after four hours of training. After this training, the average number of errors made by experienced users shall not exceed two per hour of system use.

Metrics for specifying nonfunctional requirements

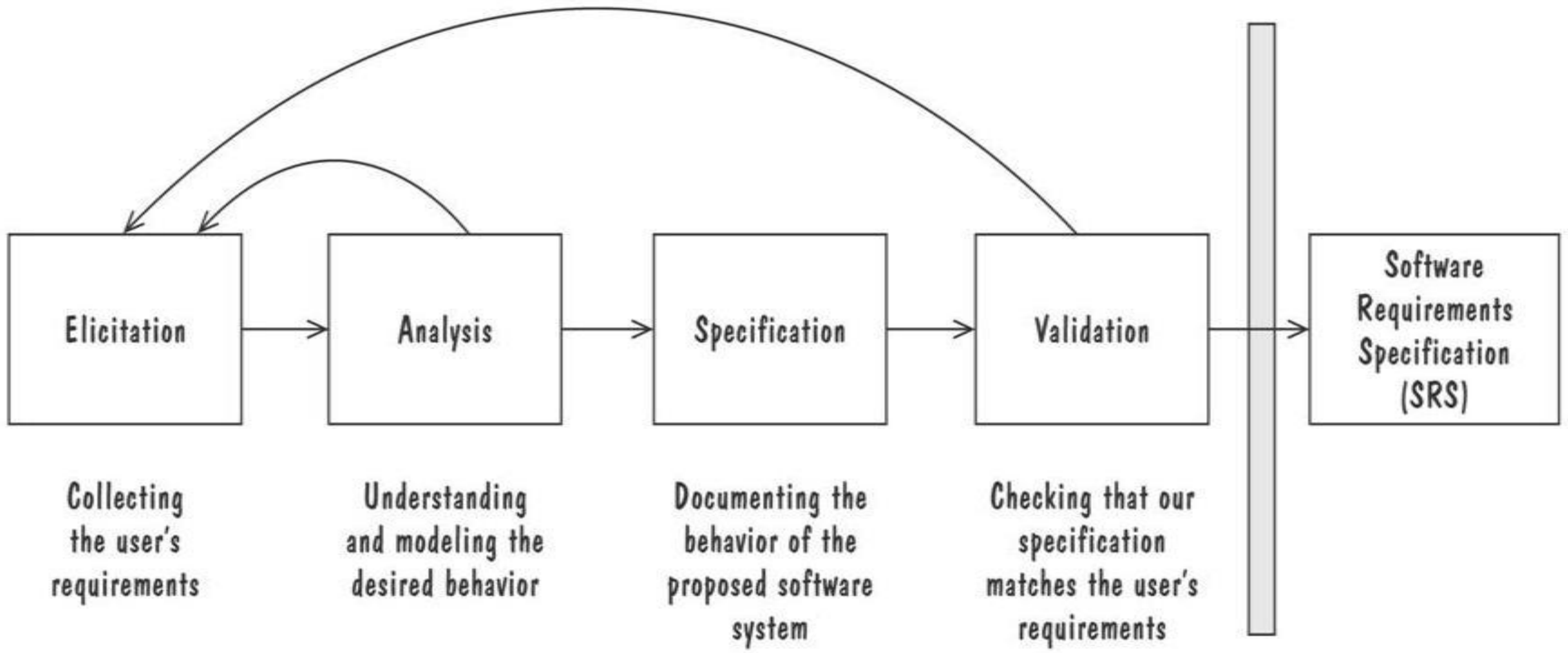
Property	**Measure**
Speed	Processed transactions/second User/event response time Screen refresh time
Size	Mbytes Number of ROM chips
Ease of use	Training time Number of help frames
Reliability	**Mean time to failure (MTTF)** Probability of unavailability Rate of failure occurrence Availability
Robustness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

The Requirements Process

Process for Capturing Requirements

Performed by the req. analyst or system analyst

The final outcome is a Software Requirements Specification (SRS) document


Elicitation
Analysis
Specification
Validation
Software Requirements Specification (SRS)
Collecting the user's requirements
Understanding and modeling the desired behavior
Documenting the behavior of the proposed software system
Checking that our specification matches the user's requirements

II. The software requirements document

- Software Requirements Specification (SRS)
  - Official statement of
  - What will be implemented
- Should include:
  - User requirements
  - Detailed system requirements
  - Including functional and non-functional requirements
- It is NOT a design document.
  - Should not indicate HOW the features will be implemented

Software Requirements Doc Users/Uses

- Set of users (readers):
  - Customers/Users
  - Project managers
  - Developers
  - Test engineers
  - Technical writers
  - Maintenance engineers
- Uses:
  - Understand scope of system
  - Project planning
  - Design and implementation
  - System testing
  - User documentation
  - Bug fixing (correctness), adding new features (consistency), etc.

Requirements document variability

- Level of detail, length, and format depends on:
  - Type of system being developed (interactive, embedded, etc)
  - Size of system
  - Development process used (waterfall vs agile)
  - Presence of safety-critical features (=> formal notation)
- Incremental development => Incremental SRS.
  - Revise the SRS at beginning of each iteration
  - Once it is reviewed and approved it becomes the Baseline SRS
  - Must have a Baseline SRS for each iteration/cycle
  - Changes to the Baseline during development must have special approval

IEEE Standard: SRS template

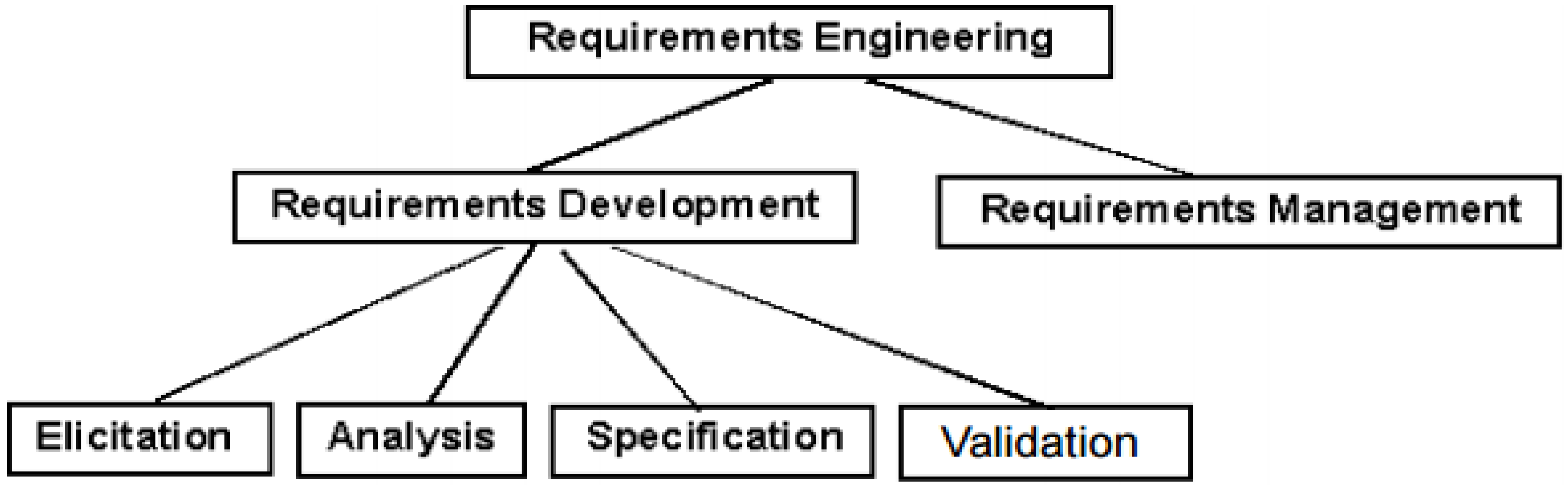
Table of Contents

IEEE Std 830-1998 (on TRACS)

SRS writing: good practices

- Label sections, subsections, requirements consistently
  - Don’t ever renumber/relabel requirements
  - Use sequential numbers OR
  - Hierarchical numbers or labels (1.1.2.3 or ship.table.col.sort)
- Use “TBD” as a placeholder for missing info
  - Resolve these before accepting as baseline SRS
- Cross reference other documents (avoid duplication)
- User interface elements
  - Don’t include screenshots in SRS (they are design specifications)

III. Requirements engineering processes



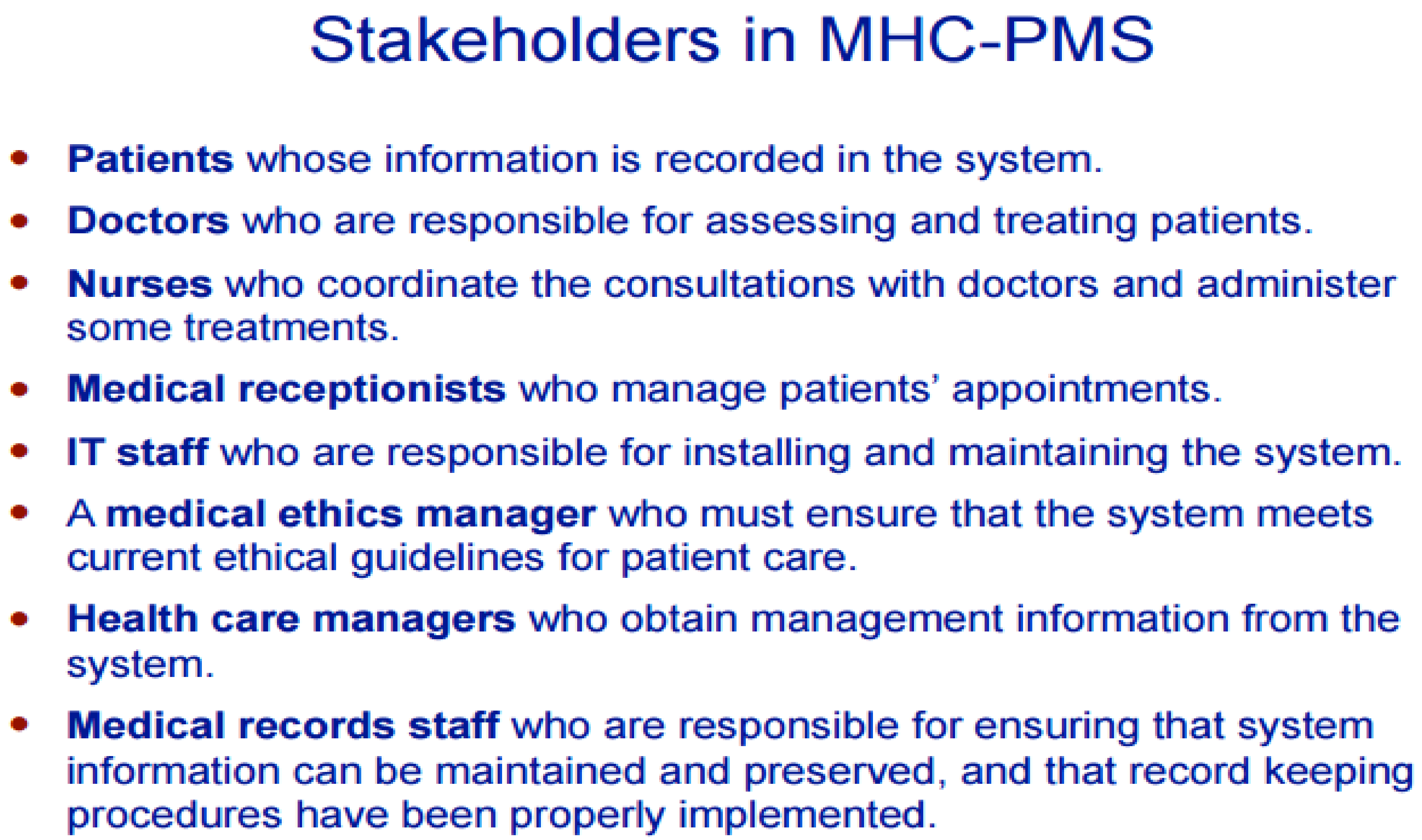
What is requirements engineering?

- The process of
  - finding out (elicitation)
  - analyzing (analysis)
  - documenting (specification)
  - and checking (validation)

  the required services and constraints of the system to be developed
- Result: software requirements specification document.
- This is the traditional approach to handling requirements.

Actors in Requirements Development

- Requirements Analyst (or Requirements Engineer)
  - Work with customers to gather, analyze, and document requirements
  - Developer may work in this role
- Stakeholders
  - customers, end users, legal staff
  - may include members of developer organization



Stakeholders in MHC-PMS

- **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.

MHC-PMS - A patient information system for mental health care

1 Requirements Elicitation

- What is the goal of this discipline?

  Identify needs and constraints of stakeholders

- What methods are used to carry it out?
  - Interviews: analysts meet with stakeholders one on one
  - Elicitation workshops: panel or forum of stakeholders
  - Ethnography: observation/immersion in user environment
- What are some tools that the requirements analyst can use?
  - Scenarios: **specific** stories that describe specific interactions
  - Use Cases: **generalized** descriptions of interactions

Scenarios and Use cases

- What is a scenario?
  - Description of one (or more) <u>specific</u> interaction(s) between a specific user and the system
  - A narrative, a story, in language the user understands
- What is a use case?
  - Description of a single <u>general</u> interaction of an actor (or role) with the system.
  - Includes all alternatives that may occur during the interaction
- What is a use case diagram?
  - Consists of an actor (stick figure) and the name of the use case in an oval
  - Description of each use case is documented elsewhere

Scenario for “collecting medical history” in MHC-PMS

John, the patient, meets with Rita the receptionist.
Rita searches for John's record in the system by family name ("Doe"). There's already a Doe in the system so she uses his first name and birthdate to find his record.
Rita chooses to add medical history to John's record.
Rita follows the prompts from the system and enters information about John's previous consultation with a psychiatrist that he had because he had been feeling suicidal. She also enters information about his existing medical conditions (he has diabetes) and the medication he is currently taking (Metformin). He has no allergies, so she leaves that blank. She fills in information about his home life (he is single, recently divorced, living alone).
John's record is then entered into the database, and this fact is recorded into the system log showing the start and end time of the session and Rita's name.

Use case for “collecting medical history” in MHC-PMS, page 1

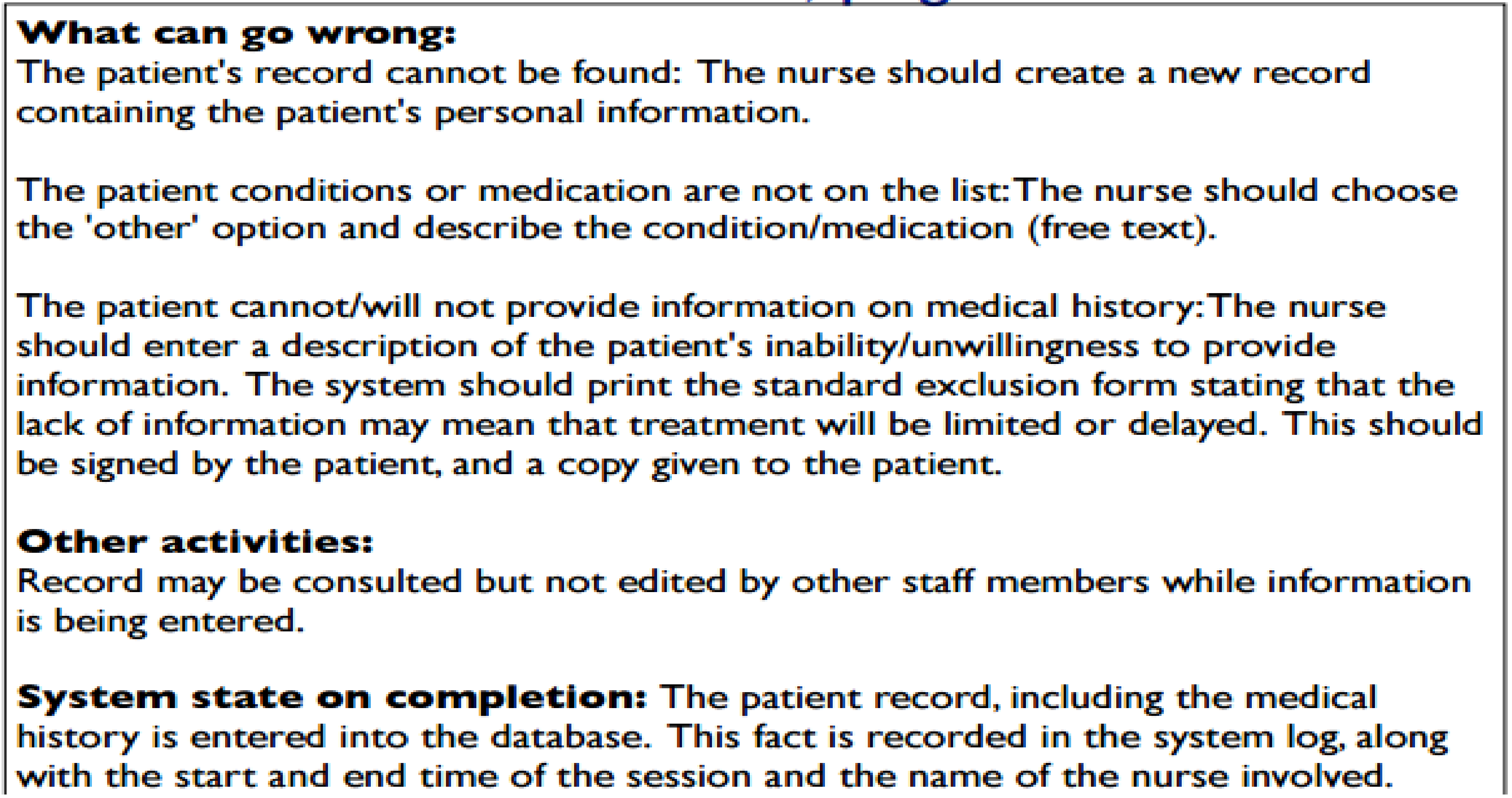
Initial assumption: The patient has seen a medical receptionist who has already created a record in the system containing his/her personal information (name, address, age, etc.). A nurse is logged into the system.

Normal Interaction:
The nurse searches for the patient by family name. If there is more than one patient with the same surname, the first name and date of birth are used to identify the patient.

The nurse chooses an option to add medical history.

The nurse then follows a series of prompts from the system to enter information about consultations elsewhere on mental health problems (free text input), existing medical conditions (selected from a list), medication currently taking (selected from a list), allergies (free text) and home life (filling out a form).

Use case for "collecting medical history" in MHC-PMS, page 2



What can go wrong:
The patient's record cannot be found: The nurse should create a new record containing the patient's personal information.

The patient conditions or medication are not on the list:The nurse should choose the 'other' option and describe the condition/medication (free text).

The patient cannot/will not provide information on medical history:The nurse should enter a description of the patient's inability/unwillingness to provide information. The system should print the standard exclusion form stating that the lack of information may mean that treatment will be limited or delayed. This should be signed by the patient, and a copy given to the patient.

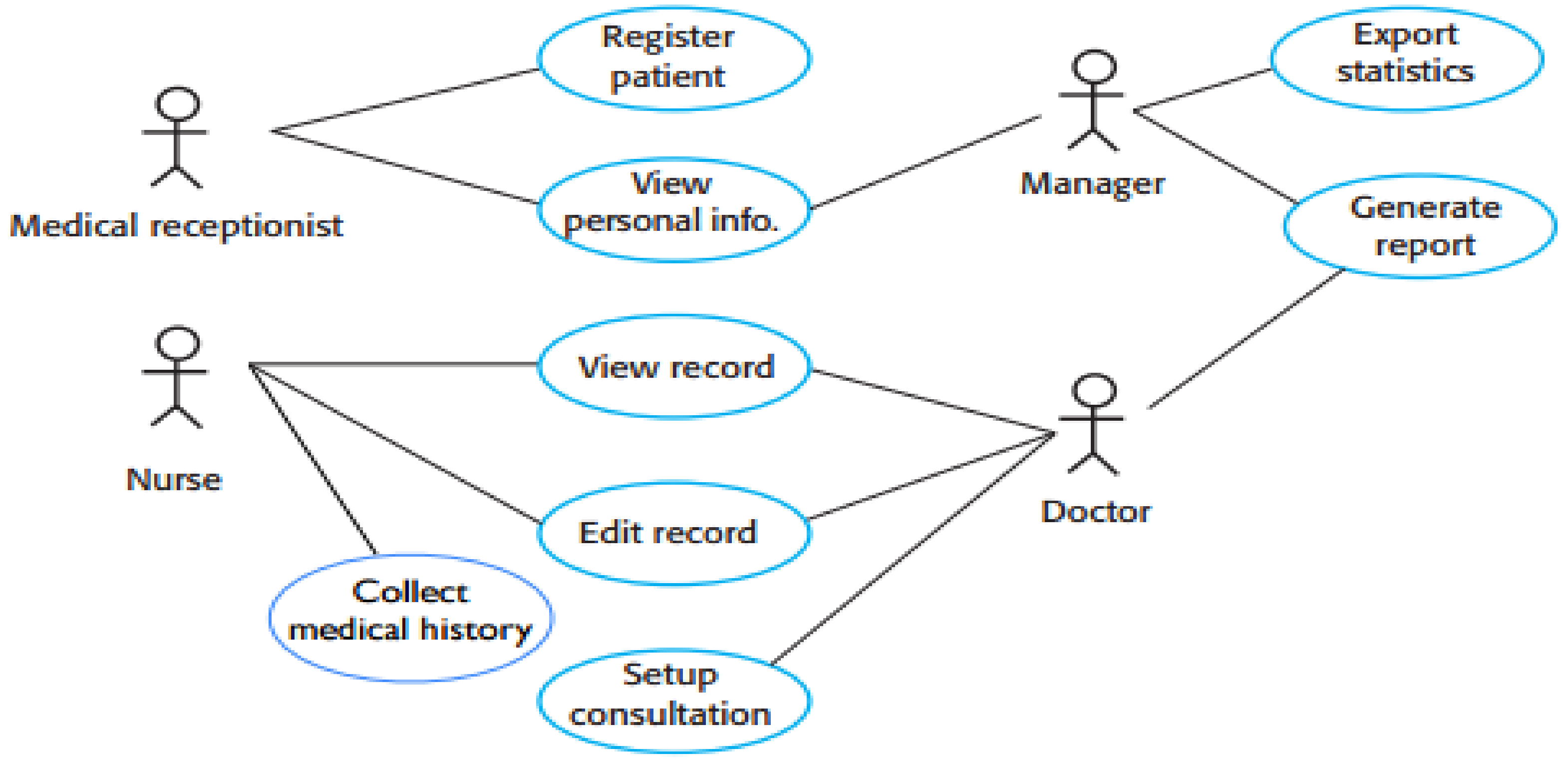
Other activities:
Record may be consulted but not edited by other staff members while information is being entered.

System state on completion: The patient record, including the medical history is entered into the database. This fact is recorded in the system log, along with the start and end time of the session and the name of the nurse involved.

Use case diagrams

- Use Case: overview of one user/system interaction
  - Focused on **one goal** of an actor
- Use Case Diagram components:
  - **stick figure:** actor (user or external system)
  - **ellipse:** named interaction (verb-noun phrase)
  - **line:** indicates involvement in interaction
- Diagram is supplemented with further details describing the use case (see previous 2 slides).
- Composite use case diagram:
  - all interactions involving a given actor or
  - all interactions of the whole system

Use case diagram for the MHC-PMS



Each oval represents a single use case (goal).
Each one should also have a textual description.

2 Requirements Analysis

- What is the goal of this discipline?

  Develop good quality, detailed requirements (by refining and organizing them)

- What methods are used to carry it out?
  - <u>Organizing</u> requirements into groups
  - <u>Modeling</u>: represent requirements in a model, refine models that were developed in elicitation
    - use case diagram: to track completeness
  - <u>Prototypes</u>: use to clarify and explore requirements with users

3 Requirements Specification

- What is the goal of this discipline?

  Document the collected user needs and constraints in a consistent and accessible format

  - Carefully record requirements in a repository (document)

- What format can the specifications take?
  - Natural language sentences
  - Structured specifications (forms/templates)
  - Graphical notations (UML diagrams, etc.)
  - Mathematical specifications: clear, but not universal

Natural language specification

- Natural language is expressive, intuitive and universal.
  - The requirements can be understood by users and customers.
- Natural language is vague and ambiguous.
  - Assumed meaning often depends on the background of the reader.
- What are good guidelines for writing specifications in natural language?
  - Use a standard format (see next page)
  - Use active voice (the system shall ...)

Example requirements for the insulin pump software system

3.2 The system shall measure the blood sugar and deliver insulin, if required, every 10 minutes. *(Changes in blood sugar are relatively slow so more frequent measurement is unnecessary; less frequent measurement could lead to unnecessarily high sugar levels.)*

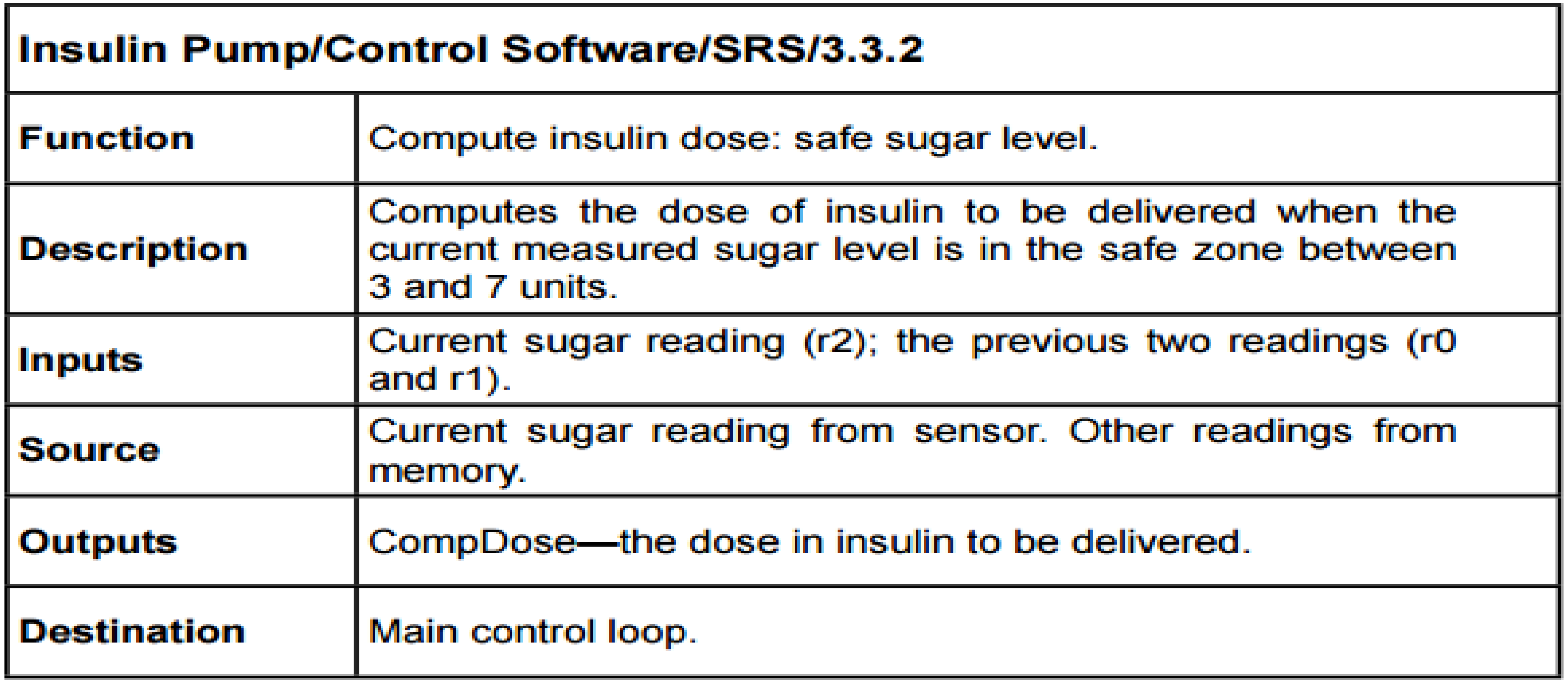
3.6 The system shall run a self-test routine every minute with the conditions to be tested and the associated actions defined in Table 1. *(A self-test routine can discover hardware and software problems and alert the user to the fact the normal operation may be impossible.)*

- These are user requirements
- Each is one sentence, along with a “rationale” statement

Structured specifications

- An approach to writing requirements using templates to write them in a standard way.
- For example, have entries for:
  - Name and description of the function or entity.
  - Description of inputs and where they come from.
  - Description of outputs and where they go to.
  - Description of the action to be taken.
  - Information about the information needed for the computation and other entities used.
  - Pre and post conditions (if appropriate).
  - The side effects (if any) of the function.
- Works well for some types of requirements but is often too rigid for writing business system requirements.

A structured specification of a requirement for an insulin pump (p1)



Insulin Pump/Control Software/SRS/3.3.2	
Function	Compute insulin dose: safe sugar level.
Description	Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units.
Inputs	Current sugar reading (r2); the previous two readings (r0 and r1).
Source	Current sugar reading from sensor. Other readings from memory.
Outputs	CompDose—the dose in insulin to be delivered.
Destination	Main control loop.

4 Requirements Validation

- What is the goal of this discipline?

  Ensure requirements demonstrate desired quality characteristics

- What are the desired characteristics?
  - See slide 11, or IEEE STD 830
- What methods are used to carry it out?
  - Requirements reviews (inspections): stakeholders and developers formally analyze requirements
  - Test case generation: can reveal problems in requirements: ambiguity, vagueness, omissions
- How successful is this process?
  - Somewhat, it's a very difficult problem.

5 Requirements Management

- Problem: the requirements specification document will need to change after development begins.
  - Errors may be found in the requirements
  - Users' needs change
  - Business needs change
- What are the effects of changing the set of requirements during development?
  - Rewrite part of the requirements specification doc
  - Rework: re-do design and implementation, if already started.

Requirements Management

- Who should decide what changes should be accepted?
  - Developers?
  - Customers/Users?
  - Project managers?
  - Requirements Analyst?
- Change Control board
  - Made up of representatives from groups listed above.
- How do they decide?
  - change is proposed, validated against existing requirements
  - proposal is evaluated for impact and cost
  - if approved, requirements doc, design and implementation are updated