



Chapter 4 – Requirements Engineering-II

Requirement's Specification/Documentation

Topics covered



- ✧ Functional and non-functional requirements
- ✧ Requirements engineering processes
- ✧ Requirements elicitation
- ✧ Requirements specification
- ✧ Requirements validation
- ✧ Requirements change

Requirements engineering processes



- ✧ The processes used for RE vary widely depending on the application domain, the people involved(CEO to software engineers)_and the organisation developing the requirements.
- ✧ However, 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.

What is a Software Requirement?



It is a software capability that

- is needed by the user to solve a problem to achieve an objective, and
- must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documentation

Initial Requirements (contd)



- ✧ There are two categories of requirements
- ✧ A *functional requirement* specifies an action that the software product must be able to perform
 - Often expressed in terms of inputs and outputs
- ✧ A *nonfunctional requirement* specifies properties of the software product itself, such as
 - Platform constraints
 - Response times
 - Reliability

Project Client – Interactions/Meetings



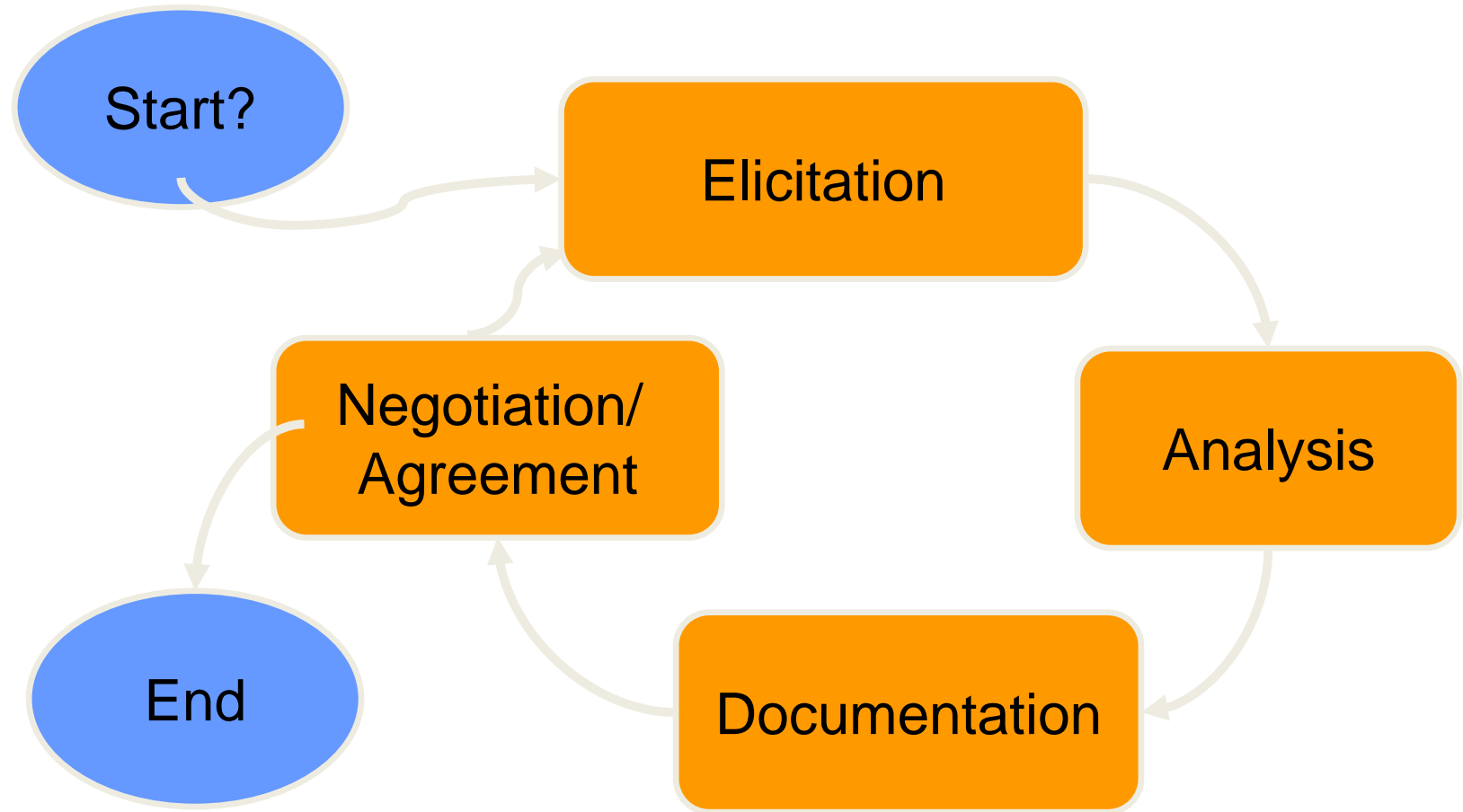
✧ What do client needs ?

- A *functional requirement* specifies an action that the software product must be able to perform
- **Programmer/developer** – Roles and Mindset

✧ How do you Achieve them ?

- A *nonfunctional requirement* specifies properties of the software product itself, such as
 - Platform constraints
 - Response times/performance
 - Reliability
- **Software Engineer** – Roles and Mindset

RM Processes



What is Requirements Management (RM)?



✧ A systematic approach for

- eliciting,
- organizing, and
- documenting

the requirements of the system, and

✧ a process that establishes and maintains agreement between the customer and the project team on the changing requirements of the system.



Requirement's elicitation

Requirements elicitation and analysis



- ✧ Sometimes called **requirements elicitation** or **requirements discovery**.
- ✧ Involves **technical staff** working with customers to find out about the **application domain**, the **services** that the system should provide and the system's **operational constraints**.
- ✧ May involve end-users, managers, engineers involved in maintenance, domain experts, trade unions, etc. These are called ***stakeholders***.

Requirements elicitation



- ✧ Software engineers work with a range of system stakeholders to find out about the **application domain**, the **services** that the system should provide, the required system performance, hardware **constraints**, other systems, etc.
- ✧ Stages include:
 - Requirements **discovery**,
 - Requirements **classification and organization**,
 - Requirements **prioritization and negotiation**,
 - Requirements **specification**.

Problems of requirements elicitation



- ✧ Stakeholders don't know what they **really want**.
- ✧ Stakeholders express requirements in **their own terms**.
- ✧ Different stakeholders may have **conflicting** requirements.
- ✧ **Organisational and political** factors may influence the system requirements.
- ✧ **The requirements change** during the analysis process.
New stakeholders may emerge and the business environment may change.

Process activities



✧ Requirements **discovery**

- Interacting with stakeholders to discover their requirements. Domain requirements are also discovered at this stage.

✧ Requirements **classification and organisation**

- Groups related requirements and organises them into coherent clusters.

✧ **Prioritisation and negotiation**

- Prioritising requirements and resolving requirements conflicts.

✧ Requirements **specification**

- Requirements are documented and input into the next round of the spiral.

Requirement's discovery



- ✧ The process of gathering information about the **required and existing systems** and distilling the **user and system** requirements from this information.
- ✧ **Interaction** is with system stakeholders from managers to external regulators.
- ✧ Systems normally have a range of stakeholders.

RE Gathering/Fact Finding Techniques



- ✧ Background Reading: about users, stakeholders, environment, company, ..etc
- ✧ Interviewing: users, customers and stakeholders
- ✧ Use cases : Model/Visual based technique
- ✧ Observation: of the existing system and users
- ✧ Document Sampling: about the old system
- ✧ Questionnaires: to the users and stakeholders
- ✧ Workshops
- ✧ Brainstorming
- ✧ **Storyboarding/Prototyping**



Requirement's **Specification/Documentation**

Stories and scenarios



- ✧ Scenarios and user stories are **real-life examples** of how a system can be used.
- ✧ Stories and scenarios are a description of how a system may be used for a particular task.
- ✧ Because they are based on a practical situation, stakeholders can relate to them and can comment on their situation with respect to the story.

Use Stories Examples



USER STORY TEMPLATE

Story Title

User Story 1

As a(stakeholder)
I want to (task),
So That (desired result)

Acceptance Criteria

Measurable results, what defines "done"?
And I know I am done when

User Story

Title:

Priority:

Estimate:

User Story:

As a [description of user],
I want [functionality]
so that [benefit].

Acceptance Criteria:

Given [how things begin]
When [action taken]
Then [outcome of taking action]

Photo sharing in the classroom (iLearn)



- ✧ Jack is a primary school teacher in Ullapool (a village in northern Scotland). He has decided that a class project should be focused around the fishing industry in the area, looking at the history, development and economic impact of fishing. As part of this, pupils are asked to gather and share reminiscences from relatives, use newspaper archives and collect old photographs related to fishing and fishing communities in the area. Pupils use an iLearn wiki to gather together fishing stories and SCRAN (a history resources site) to access newspaper archives and photographs. However, Jack also needs a photo sharing site as he wants pupils to take and comment on each others' photos and to upload scans of old photographs that they may have in their families.

Jack sends an email to a primary school teachers group, which he is a member of to see if anyone can recommend an appropriate system. Two teachers reply and both suggest that he uses KidsTakePics, a photo sharing site that allows teachers to check and moderate content. As KidsTakePics is not integrated with the iLearn authentication service, he sets up a teacher and a class account. He uses the iLearn setup service to add KidsTakePics to the services seen by the pupils in his class so that when they log in, they can immediately use the system to upload photos from their mobile devices and class computers.

Scenarios



- ✧ A structured form of user story
- ✧ Scenarios should include
 - A description of the starting situation;
 - A description of the normal flow of events;
 - A description of what can go wrong;
 - Information about other concurrent activities;
 - A description of the state when the scenario finishes.

Uploading photos iLearn)



- ✧ **Initial assumption:** A user or a group of users have one or more digital photographs to be uploaded to the picture sharing site. These are saved on either a tablet or laptop computer. They have successfully logged on to KidsTakePics.
- ✧ **Normal:** The user chooses upload photos and they are prompted to select the photos to be uploaded on their computer and to select the project name under which the photos will be stored. They should also be given the option of inputting keywords that should be associated with each uploaded photo. Uploaded photos are named by creating a conjunction of the user name with the filename of the photo on the local computer.
- ✧ On completion of the upload, the system automatically sends an email to the project moderator asking them to check new content and generates an on-screen message to the user that this has been done.

Uploading photos



- ✧ **What can go wrong:**
- ✧ No moderator is associated with the selected project. An email is automatically generated to the school administrator asking them to nominate a project moderator. Users should be informed that there could be a delay in making their photos visible.
- ✧ Photos with the same name have already been uploaded by the same user. The user should be asked if they wish to re-upload the photos with the same name, rename the photos or cancel the upload. If they chose to re-upload the photos, the originals are overwritten. If they chose to rename the photos, a new name is automatically generated by adding a number to the existing file name.
- ✧ **Other activities:** The moderator may be logged on to the system and may approve photos as they are uploaded.
- ✧ **System state on completion:** User is logged on. The selected photos have been uploaded and assigned a status 'awaiting moderation'. Photos are visible to the moderator and to the user who uploaded them.



Requirement's **Specification/Documentation**

Requirement's specification



- ✧ The process of writing the **user and system requirements** in a requirements document.
- ✧ User requirements have to be **understandable by end-users and customers** who do not have a technical background.
- ✧ System requirements are more **detailed requirements** and may include more technical information.
- ✧ The requirements may be part of a **contract** for the system development
 - It is therefore important that these are as complete as possible.

User and system requirements



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.

Ways of writing a system requirements specification



Notation	Description
Natural language	The requirements are written using numbered sentences in natural language. Each sentence should express one requirement.
Structured natural language	The requirements are written in natural language on a standard form or template. Each field provides information about an aspect of the requirement.
Design description languages	This approach uses a language like a programming language, but with more abstract features to specify the requirements by defining an operational model of the system. This approach is now rarely used although it can be useful for interface specifications.
Graphical notations	Graphical models, supplemented by text annotations, are used to define the functional requirements for the system; UML use case and sequence diagrams are commonly used.
Mathematical specifications	These notations are based on mathematical concepts such as finite-state machines or sets. Although these unambiguous specifications can reduce the ambiguity in a requirements document, most customers don't understand a formal specification. They cannot check that it represents what they want and are reluctant to accept it as a system contract

Requirements and design



- ✧ In principle, **requirements** should state **what the system should do** and the **design** **should describe how it does this**.
- ✧ In practice, requirements and design are inseparable
 - A system architecture may be designed to structure the requirements;
 - The system may inter-operate with other systems that generate design requirements;
 - The use of a specific architecture to satisfy non-functional requirements may be a domain requirement.
 - This may be the consequence of a regulatory requirement.

Natural language specification



- ✧ Requirements are written as natural language sentences supplemented by diagrams and tables.
- ✧ Used for writing requirements because it is expressive, intuitive and universal. This means that the requirements can be understood by users and customers.

Photo sharing in the classroom (iLearn)



- ✧ Jack is a primary school teacher in Ullapool (a village in northern Scotland). He has decided that a class project should be focused around the fishing industry in the area, looking at the history, development and economic impact of fishing. As part of this, pupils are asked to gather and share reminiscences from relatives, use newspaper archives and collect old photographs related to fishing and fishing communities in the area. Pupils use an iLearn wiki to gather together fishing stories and SCRAN (a history resources site) to access newspaper archives and photographs. However, Jack also needs a photo sharing site as he wants pupils to take and comment on each others' photos and to upload scans of old photographs that they may have in their families.

Jack sends an email to a primary school teachers group, which he is a member of to see if anyone can recommend an appropriate system. Two teachers reply and both suggest that he uses KidsTakePics, a photo sharing site that allows teachers to check and moderate content. As KidsTakePics is not integrated with the iLearn authentication service, he sets up a teacher and a class account. He uses the iLearn setup service to add KidsTakePics to the services seen by the pupils in his class so that when they log in, they can immediately use the system to upload photos from their mobile devices and class computers.

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 amalgamation

- Several different requirements may be expressed together.

Guidelines for writing requirements



- ✧ Invent a **standard format** and use it for all requirements.
- ✧ Use language in a consistent way. Use **shall** for mandatory requirements, **should** for desirable requirements.
- ✧ Use text highlighting to identify key parts of the requirement.
- ✧ **Avoid** the use of computer jargon.
- ✧ Include an explanation (rationale) of why a requirement is necessary.

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.)*

Structured specifications



- ✧ An approach to writing requirements where the freedom of the requirements writer is limited and requirements are written in a **standard way**.
- ✧ This works well for some types of requirements e.g. requirements for embedded control system but is sometimes too rigid for writing business system requirements.

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.
- ✧ Information about the information needed for the **computation and other entities** used.
- ✧ Description of the **action** to be taken.
- ✧ **Pre and post** conditions (if appropriate).
- ✧ The side effects (if any) of the function.

A structured specification of a requirement for an insulin pump



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.

A structured specification of a requirement for an insulin pump



Action

CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing and the rate of increase is increasing, then CompDose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result. If the result, is rounded to zero then CompDose is set to the minimum dose that can be delivered.

Requirements

Two previous readings so that the rate of change of sugar level can be computed.

Pre-condition

The insulin reservoir contains at least the maximum allowed single dose of insulin.

Post-condition r0 is replaced by r1 then r1 is replaced by r2.

Side effects None.

Tabular specification



- ✧ Used to supplement natural language.
- ✧ Particularly useful when you have to define a number of possible alternative courses of action.
- ✧ For example, the insulin pump systems bases its computations on the rate of change of blood sugar level and the tabular specification explains how to calculate the insulin requirement for different scenarios.

Tabular specification of computation for an insulin pump

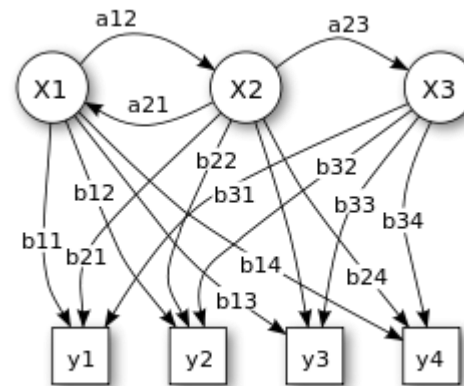
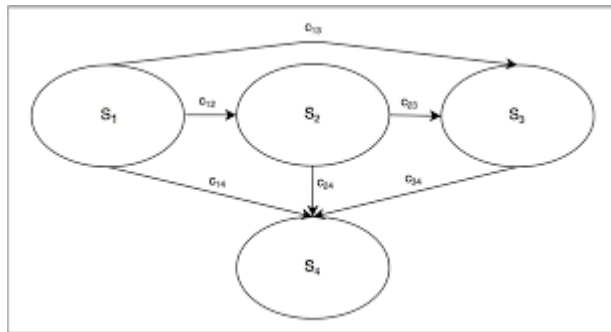


Condition	Action
Sugar level falling ($r2 < r1$)	CompDose = 0
Sugar level stable ($r2 = r1$)	CompDose = 0
Sugar level increasing and rate of increase decreasing ($(r2 - r1) < (r1 - r0)$)	CompDose = 0
Sugar level increasing and rate of increase stable or increasing ($(r2 - r1) \geq (r1 - r0)$)	CompDose = round $((r2 - r1)/4)$ If rounded result = 0 then CompDose = MinimumDose

State Transition Models



✧ To solve complex problems



Use Case(UML) Model

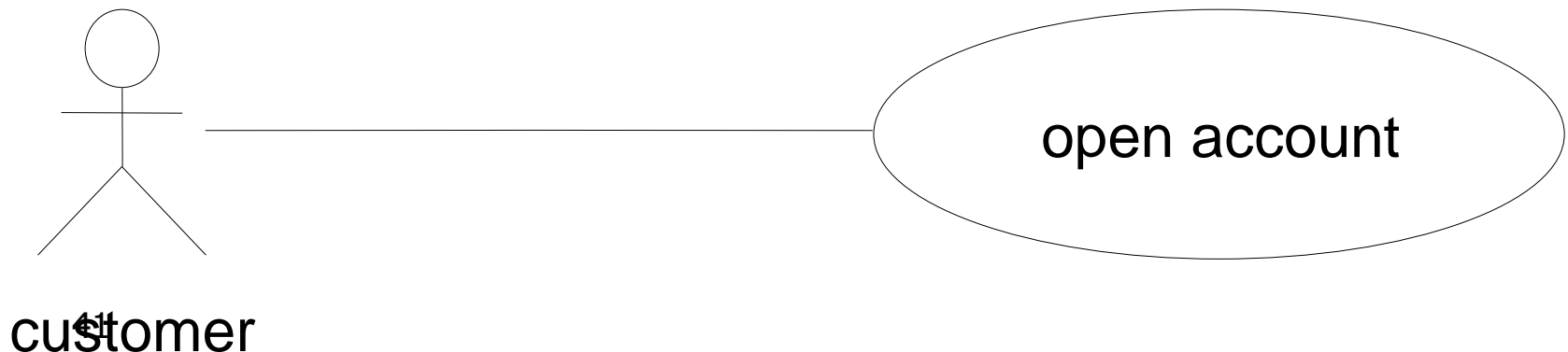


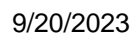
- ✧ A **use case** describes a sequence of actions the system performs that yield an observable result of value to a particular actor.
- ✧ An **actor** is someone or something that interacts with the system.
 - Users, other systems, or devices
- ✧ Documenting use cases by using
 - Use case templates: Name, descriptions, etc.
 - Use case Diagrams

UML use case diagrams



- ✧ use cases can be drawn as diagrams, with:
 - actors as stick-men, with their names below
 - use cases as ellipses with their names below or inside
 - association indicated by lines, connecting an actor to a use case in which that actor participates
 - use cases can be connected to other cases that they use / rely on(include/extend)



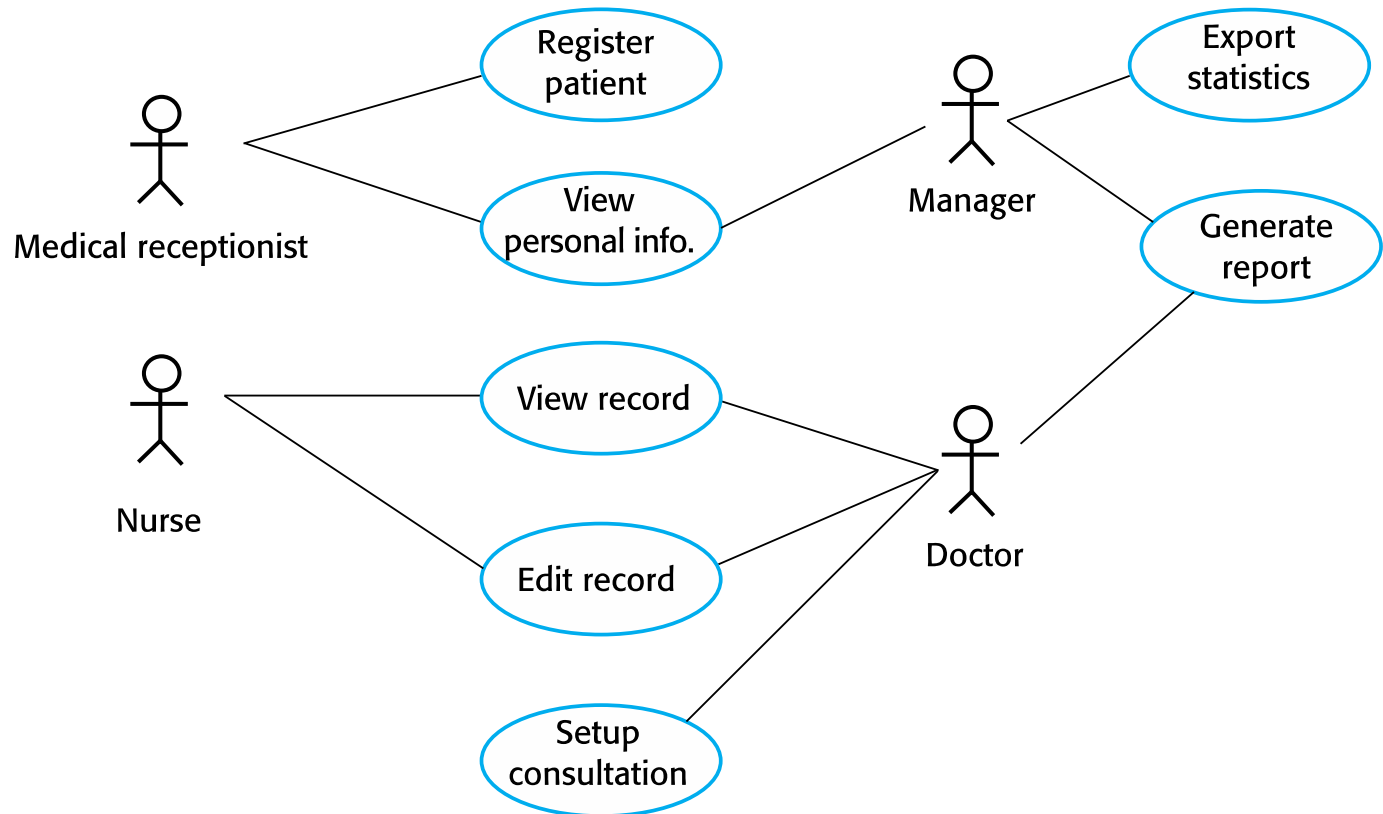


***Use cases

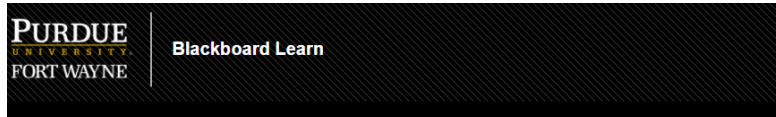


- ✧ Use-cases are a kind of scenario that are included in the UML.
- ✧ Use cases identify the actors in an interaction and which describe the interaction itself.
- ✧ A set of use cases should describe all possible interactions with the system.
- ✧ High-level graphical model supplemented by more detailed tabular description (see Chapter 5).
- ✧ UML sequence diagrams may be used to add detail to use-cases by showing the sequence of event processing in the system.

Use cases for the Mentcare system



Agile : Porotype/Story board



You've been logged out due to inactivity. Log in again to continue.

Sign-on

USERNAME

inukolly

PASSWORD

Login

Course Home Content Classlist Grades Class Progress Course Tools Course Ad

Fall 2023 ACS 56000-03 Software Engine

Announcements ▾

Test/Exams & Final Project Presentations ▾

Posted Sep 19, 2023 3:09 PM

FYI

Test -1: 25th October 2023 (In-person or online)

Test -2: 11- 14 Dec 2023. Finals week, (Online)

Final Project Demo/Presentations: 12th Dec 2023 (In-person)

Thanks

-V





Assignments

New Assignment

Edit Categories

More Actions ▾

Bulk Edit

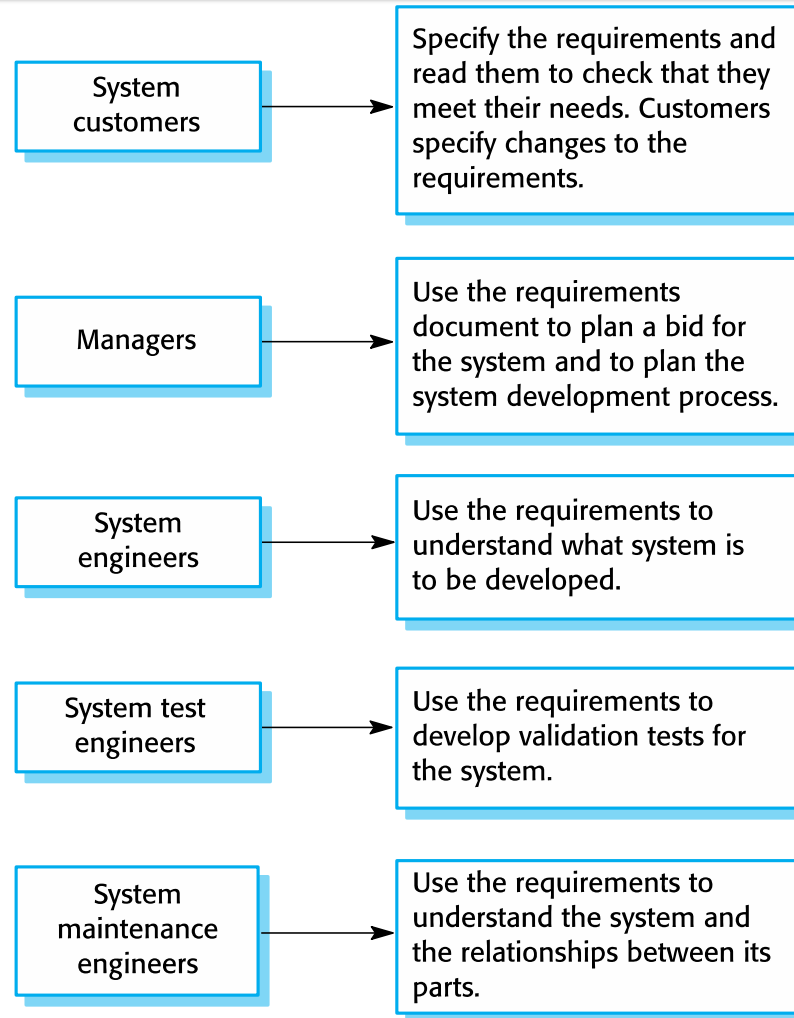
<input type="checkbox"/>	Assignment	New Submissions
	No Category	
<input type="checkbox"/>	In_class Activity: Requirements & Prototype ▾   Due on Sep 12, 2023 8:00 PM	19
<input type="checkbox"/>	In_class Activity: Requirements & Prototype 2.0 (Iteration 2.0) ▾   Due on Sep 19, 2023 8:00 PM	

The software requirements document



- ✧ The software requirements document is the official statement of what is required of the system developers.
- ✧ Should include both a definition of user requirements and a specification of the system 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.

Users of a requirements document



Requirements document variability



- ✧ Information in requirements document depends on type of system and the approach to development used.
- ✧ Systems developed incrementally will, typically, have less detail in the requirements document.
- ✧ Requirements documents standards have been designed e.g. IEEE standard. These are mostly applicable to the requirements for large systems engineering projects.

The structure of a requirements document



Chapter	Description
Preface	This should define the expected readership of the document and describe its version history, including a rationale for the creation of a new version and a summary of the changes made in each version.
Introduction	This should describe the need for the system. It should briefly describe the system's functions and explain how it will work with other systems. It should also describe how the system fits into the overall business or strategic objectives of the organization commissioning the software.
Glossary	This should define the technical terms used in the document. You should not make assumptions about the experience or expertise of the reader.
User requirements definition	Here, you describe the services provided for the user. The nonfunctional system requirements should also be described in this section. This description may use natural language, diagrams, or other notations that are understandable to customers. Product and process standards that must be followed should be specified.
System architecture	This chapter should present a high-level overview of the anticipated system architecture, showing the distribution of functions across system modules. Architectural components that are reused should be highlighted.

The structure of a requirements document



Chapter	Description
System requirements specification	This should describe the functional and nonfunctional requirements in more detail. If necessary, further detail may also be added to the nonfunctional requirements. Interfaces to other systems may be defined.
System models	This might include graphical system models showing the relationships between the system components and the system and its environment. Examples of possible models are object models, data-flow models, or semantic data models.
System evolution	This should describe the fundamental assumptions on which the system is based, and any anticipated changes due to hardware evolution, changing user needs, and so on. This section is useful for system designers as it may help them avoid design decisions that would constrain likely future changes to the system.
Appendices	These should provide detailed, specific information that is related to the application being developed; for example, hardware and database descriptions. Hardware requirements define the minimal and optimal configurations for the system. Database requirements define the logical organization of the data used by the system and the relationships between data.
Index	Several indexes to the document may be included. As well as a normal alphabetic index, there may be an index of diagrams, an index of functions, and so on.



Next Class : Requirements Validation & Intro To Design