

IQRA NATIONAL UNIVERSITY

Final-Term Examination Spring 2020

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Course Name** | **Max. Marks** | **Max. Time** | **Date** | **Instructor** |
| Software Requirement Specification | 50 | 6 hrs.  9-3 PM | 24th June 2020 | Aasma Khan |

* **Attempt all questions.**
* **Marks will be given as per the DEPTH of the answer, not LENGTH.**
* **Name : Shoaib Rehman**
* **Id : 14293**
* **Dept : BS(SE)**

**Question No: 01** (10)

Define requirements and define what the system (take example of any system) is required to do and what are the features and constraints under which it operates.

**Answer:**

**Requirement:**

Requirements are descriptions of the services that a software system must provide and the constraints under which it must operate.

**System:**

**System** is a set of rules, an arrangement of things, or a group of related things that work toward a common goal.

**Examples**

File Transfer Protocol (FTP): It can help users to download different types of files from internet.

The features are that any client can get info easily no need for payment.

**Constraints:**Students cannot access teachers or admin panel.

Teacher can upload only lectures and take online exam.

Admin can control teachers and students dashboard.

**Question No: 02** (10)

Explain software requirements types.

#### Different Types of Software Requirements

The most common types of software requirements are:

1. **Business Requirements (BR)**
   * These are high-level business goals of the organization building the product, or the customer who commissioned the project.
   * These are usually provided as a single page of high-level bullets.
2. **Market Requirements (MR)**
   * These drill down into BRs, but still are high-level. In addition to business goals, they also outline market needs.
   * These are usually provided as a prioritized bulleted list or table, and are usually less than 5 pages long.
3. **Functional Requirements (FR) – Use Cases**
   * These cover the functionality of the product in detail. [Use cases](http://www.accompa.com/product-management-blog/2009/09/19/use-cases-definition-requirements-management-basics/) are one of the [best ways of documenting functional requirements](http://www.accompa.com/product-management-blog/2009/09/22/use-cases-top-10-reasons-for-using-them-to-document-your-requirements/).
   * Depending on the product being built, FRs can run several hundred pages.
4. **Non-Functional Requirements (NFR)**
   * These are not related to the “functionality” of the product – but cover goals such as Reliability, Scalability, Security, Integration, etc.
   * Many projects make the mistake of not specifying these explicitly.
5. **UI Requirements (UIR)**
   * User interface specs are not considered “requirements” in traditional requirements management theory.
   * Phooey! In my opinion, UI specs are indeed requirements ([what else are they?](http://www.accompa.com/product-management-blog/2009/11/20/should-user-interface-ui-be-a-part-of-requirements/)) – and in fact should be considered an integral part of requirements for any software that has a UI.

**Question No: 03** (10)

State difference between system requirement engineering and software requirement engineering.

**Answer:**

[**System requirements**](https://www.sebokwiki.org/wiki/System_Requirement_(glossary)):

 are all of the [**requirements**](https://www.sebokwiki.org/wiki/Requirement_(glossary)) at the *system level* that describe the functions which the system as a whole should fulfill to satisfy the [**stakeholder needs and requirements**](https://www.sebokwiki.org/wiki/Stakeholder_Requirement_(glossary)), and are expressed in an appropriate combination of textual statements, views, and non-functional requirements; the latter expressing the levels of safety, security, reliability, etc., that will be necessary.

System requirements play major roles in systems engineering, as they:

* Form the basis of system [**architecture**](https://www.sebokwiki.org/wiki/Architecture_(glossary)) and [**design**](https://www.sebokwiki.org/wiki/Design_(glossary)) activities.
* Form the basis of system [**integration**](https://www.sebokwiki.org/wiki/Integration_(glossary)) and [**verification**](https://www.sebokwiki.org/wiki/Verification_(glossary)) activities.
* Act as reference for [**validation**](https://www.sebokwiki.org/wiki/Validation_(glossary)) and stakeholder acceptance.
* Provide a means of communication between the various technical staff that interact throughout the project.

**Software Requirement Engineering:**

### *Product and Process Requirements*

A product requirement is a need or constraint on the software to be developed (for example, “The software shall verify that a student meets all prerequisites before he or she registers for a course”).

A process requirement is essentially a constraint on the development of the software (for example, “The software shall be developed using a RUP process”).

Some software requirements generate implicit process requirements. The choice of verification technique is one example. Another might be the use of particularly rigorous analysis techniques (such as formal specification methods) to reduce faults that can lead to inadequate reliability. Process requirements may also be imposed directly by the development organization, their customer, or a third party such as a safety regulator.

### *Functional and Nonfunctional Requirements*

*Functional* requirements describe the functions that the software is to execute; for example, formatting some text or modulating a signal. They are sometimes known as capabilities or features. A functional requirement can also be described as one for which a finite set of test steps can be written to validate its behavior.

*Nonfunctional* requirements are the ones that act to constrain the solution. Nonfunctional requirements are sometimes known as constraints or quality requirements. They can be further classified according to whether they are performance requirements, maintainability requirements, safety requirements, reliability requirements, security requirements, interoperability requirements or one of many other types of software requirements (see Models and Quality Characteristics in the Software Quality KA).

### *Emergent Properties*

Some requirements represent emergent properties of software—that is, requirements that cannot be addressed by a single component but that depend on how all the software components interoperate. The throughput requirement for a call center would, for example, depend on how the telephone system, information system, and the operators all interacted under actual operating conditions. Emergent properties are crucially dependent on the system architecture.

### *Quantifiable Requirements*

Software requirements should be stated as clearly and as unambiguously as possible, and, where appropriate, quantitatively. It is important to avoid vague and unverifiable requirements that depend for their interpretation on subjective judgment (“the software shall be reliable”; “the software shall be user-friendly”). This is particularly important for nonfunctional requirements. Two examples of quantified requirements are the following: a call center’s software must increase the center’s throughput by 20%; and a system shall have a probability of generating a fatal error during any hour of operation of less than 1 \* 10 −8 . The throughput requirement is at a very high level and will need to be used to derive a number of detailed requirements. The reliability requirement will tightly constrain the system architecture.

**Question No 04:** (10)

Give five reasons why requirements negotiation is needed in software engineering.

**Negotiation**

is a decision-making process

involves interaction and interdependency

implies two parties with conflicts

Requirements negotiation is an iterative process through which stakeholders make tradeoffs between .

requested system functions

the capabilities of existing or envisioned technology

the delivery schedule ν the cost

**Conflicts are inevitable since stakeholders have mismatching goals**

User: many features, high level of service, early availability

Customer: budget/schedule constraints, cost effectiveness, compliance with standards

Developer: flexible contracts, stable requirements

The objectives of customers, users, or developers have to be reconciled to develop mutually acceptable agreements

Stakeholders are not forced to agree

The result of negotiation is also to understand why stakeholders disagree

Identifying opposed interests is crucial for project success

Identified disagreements represent risks that need to be addressed by managers Introduction Requirements Negotiation

Requirements negotiation should be used early on and repeated in later stages

Establishing a requirements negotiation process is not easy

How can conflicts be identified?

How can the identified conflicts be resolved?

How can feasible alternatives be found?

Who is in charge of the negotiation, the stakeholders themselves or a facilitator?

How can the negotiation be supported with tools or other means?

Requirements negotiation is an iterative process through which stakeholders make tradeoffs between

requested system functions

the capabilities of existing or envisioned technology

the delivery schedule ν the cost

**Question No 05:** (10)

Identify the **actors** and the **objects** in the following scenario to register a patient in a hospital management system and draw a **use case diagram**:

The administrator enters the patient’s name, address, date of birth and emergency contact details into the system. If the patient has only public health insurance, the administrator enters the patient’s Medicare number, and the system verifies this with government health database. If the patient also has private health insurance, then the administrator enters also the patient’s private health insurance details, and the system verifies these details with the private health insurance system. When these details are verified as correct, the system saves the patient's details and confirms the registration.

**Answer:**

**Identification of the actors and objects in the above scenario are given below**

* **Actors**

1. Administrator
2. Government Health Database.
3. Private health insurance system.

* **Objects**

1. Patient
2. Address
3. Emergency Contact
4. Public health insurance
5. Private health insurance
6. Registration

