SOLUTION-2018

Q-1

a) What are functional and non-functional requirements?
b) Why spiral model is known as meta model?
c) Identify the different types of problems an analyst may come acrossduring requirement analysis.
d) What are the reasons for software crisis? Explain.

e) What are the roles of a software project manager ?

(NIBEDITA MISRA 1729141)

a) **Functional Requirements** - A functional requirement, in software and systems engineering, is a declaration of the intended function of a system and its components. Based on functional requirements, an engineer determines the behavior (output) that a device or software is expected to exhibit in the case of a certain input. A system design is an early form of a functional requirement.

Non-Functional Requirements - Nonfunctional Requirements (NFRs) define system attributes such as security, reliability, performance, maintainability, scalability, and usability. They serve as constraints or restrictions on the design of the system across the different backlogs.

b) The Spiral model is called as a Meta Model because it subsumes all the other SDLC models. For example, a single loop spiral actually represents the Iterative Waterfall Model.The spiral model uses the approach of Prototyping Model by building a prototype at the start of each phase as a risk handling technique.

Spiral model is one of the most important Software Development Life Cycle models, which provides support for Risk Handling. In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project. Each loop of the spiral is called a Phase of the software development process. The exact number of phases needed to develop the product can be varied by the project manager depending upon the project risks. As the project manager dynamically determines the number of phases, so the project manager has an important role to develop a product using spiral model.

c) The different types of problems an analyst may come across during requirement analysis :-

- Customers don’t really know what they want.
- Requirements change during the course of the project.
- Customers have unreasonable timelines
- Communication gaps exist between customers, engineers and project managers.
- The development team doesnt understand the politics of the customers organisation.

d) The software crisis is characterized by an inability develop software on time, within budget, and within requirements. The following are the main reasons for software crisis:

- Lack of communication between software developers and users.
- Increase in size of software.
- Increase in cost of developing a software.
- Increased complexity of the problem area.
- Project management problem.
- Lack of understanding of the problem and its environment.
- Duplication of efforts due to absence of automation in most of the software development activities.
- High optimistic estimates regarding software development time and cost.

e) The roles of a software project manager are:-
- Activity and resource planning.
- Organizing and motivating a project team,
- Controlling time management.
- Cost estimating and developing the budget.
- Ensuring customer satisfaction.
- Analyzing and managing project risk.
- Monitoring progress.
- Managing reports and necessary documentation.

Q.2(a) Suppose you are the project manager in a software organization. During requirement gathering you found that the end users are not clear about their requirements and even after several iterations of discussion the specification document is not finalized. This problem will motivate you to follow which software life cycle models? Justify and discuss about the life cycle model. Rapid Application Development (RAD) Model

(TANISHA BHARDWAJ 1729089)

RAD is a type of increment model. Sometimes it is also referred as rapid prototyping model.

Major aims:

- To reduce the communication gap between the clients and the developers.
- To reduce the time and the cost incurred to develop the software.
- To facilitate the accommodation of change requests as early as possible. It can be done by doing minimum planning and make heavy reuse of existing codes.
- The development team usually consists of about five to six members including the customer representative.

Methodology:

- Here, the development takes place in short cycles or iterations.
- Plans are made for one increment at a time. The time planned for each iteration is called “Time box”.
- Each increment enhances the implemented functionality of the application a little as compared to the previous increment.
- During each iteration:
  - A quick and dirty prototype style software is developed. This prototype is of limited functionality. Once developed, it is evaluated and approved by the customer with some modifications, if any.
  - This prototype is refined into successive iterations and at the end it becomes deliverable.

Advantages of RAD Model:

- Fast application development and delivery.
- Least testing activity required.
- Visualization of progress.
- Less resources required.
- Review by the client from the very beginning of development so very less chance to miss the requirements.
- Very flexible if any changes required.
- Cost effective.
- Good for small projects.

Disadvantages of RAD Model:

- High skilled resources required.
- On each development phase client's feedback required.
- Automated code generation is very costly.
- Difficult to manage.
- Not a good process for long term and big projects.
- Proper modularization of project required.
- Applications for which RAD is suitable:

b)Write down the functional requirements for an automated library management system which supports the functions like add new member, renew membership and cancel membership. Please also create the suitable decision tree.

A Library Management System is a software built to handle the primary housekeeping functions of a library. Libraries rely on library management systems to manage asset collections as well as relationships with their members. Library management systems help libraries keep track of the books and their checkouts, as well as members' subscriptions and profiles.

Library management systems also involve maintaining the database for entering new books and recording books that have been borrowed with their respective due dates.

System Requirements

We will focus on the following set of requirements while designing the Library Management System:

1. Any library member should be able to search books by their title, author, subject category as well by the publication date.

2. Each book will have a unique identification number and other details including a rack number which will help to physically locate the book.

3. There could be more than one copy of a book, and library members should be able to check-out and reserve any copy. We will call each copy of a book, a book item.

4. The system should be able to retrieve information like who took a particular book or what are the books checked-out by a specific library member.

5. There should be a maximum limit (5) on how many books a member can check-out.

6. There should be a maximum limit (10) on how many days a member can keep a book.

7. The system should be able to collect fines for books returned after the due date.

8. Members should be able to reserve books that are not currently available.

9. The system should be able to send notifications whenever the reserved books become available, as well as when the book is not returned within the due date.

10. Each book and member card will have a unique barcode. The system will be able to read barcodes from books and members' library cards.

Use case diagram

We have three main actors in our system:

- **Librarian:** Mainly responsible for adding and modifying books, book items, and users. The Librarian can also issue, reserve, and return book items.
- **Member:** All members can search the catalog, as well as check-out, reserve, renew, and return a book.
- **System:** Mainly responsible for sending notifications for overdue books, canceled reservations, etc.

Here are the top use cases of the Library Management System:

- **Add/Remove/Edit book:** To add, remove or modify a book or book item.
- **Search catalog:** To search books by title, author, subject or publication date.
- **Register new account/cancel membership:** To add a new member or cancel the membership of an existing member.
- **Check-out book:** To borrow a book from the library.
- **Reserve book:** To reserve a book which is not currently available.
- **Renew a book:** To reborrow an already checked-out book.

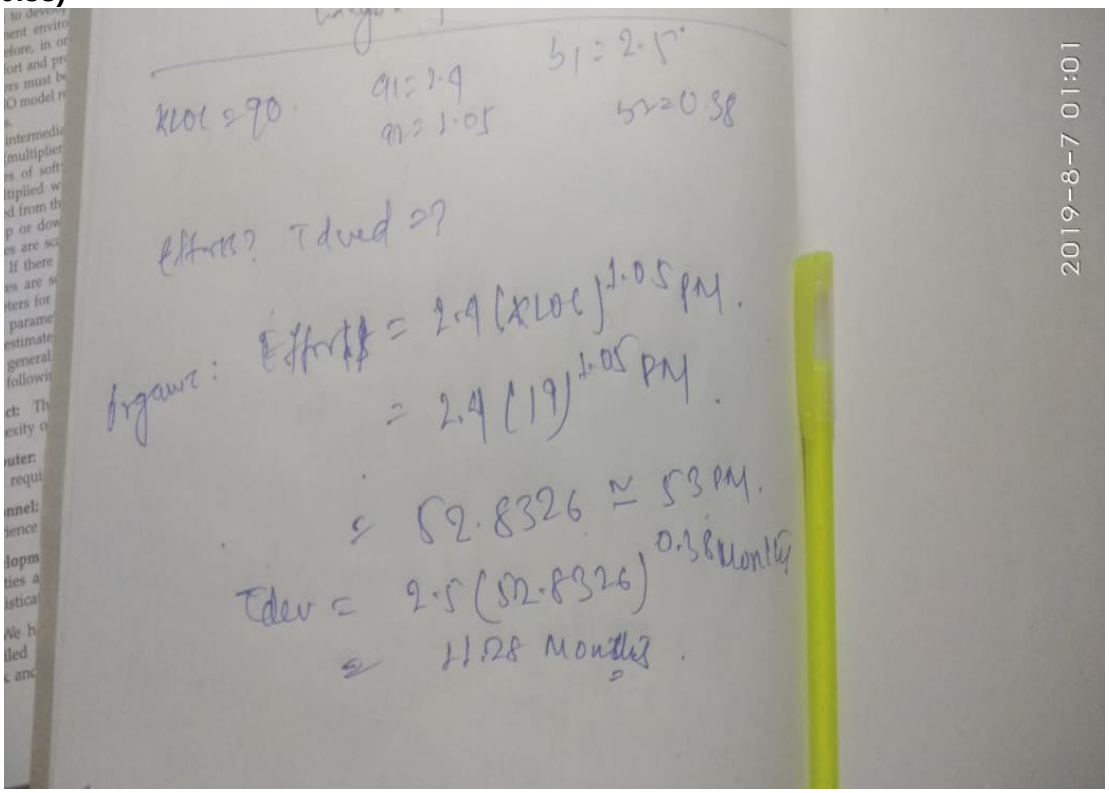
Return a book: To return a book to the library which was issued to a member.

Q4 a)Differentiate between basic COCOMO and intermediate COCOMO. (MUSKAN MOHANTY 1729139)

- Basic COCOMO model: Basic COCOMO model estimates the software development effort using only a **single predictor variable** (size in DSI) and three software development modes. Where as in the Intermediate Model estimates the software development effort by using **fifteen cost driver variables** besides the size variable used in Basic COCOMO

- Basic COCOMO is good for **quick, early, rough order of magnitude** estimates of software costs where as the Intermediate Model can be applied **across the entire software product** for easily and rough cost estimation during the early stage or it can be applied at the **software product component level** for more accurate cost estimation in more detailed stages

(b) Assume that the size of an organic type software product has been estimated to be 90,0001ines of source code. Assume that the average salary of software engineers be Rs. 20,000/- per month. Determine the effort required to develop the software product, the nominal development time and productivity using COCOMO. (the constants are a or al=2.4, b or a2=1.05, c or bl=2.5, d or b2=0.38)



Q.5 Write short notes (any two)

I. Agile development process
II. Critical path and slack time.
III. Software Project Management Plan (SPMP) document.
IV. Criteria of a good SRS document.

(SAGNIK SARBADHIKARI 1729151)

a)

AGILE DEVELOPMENT PROCESS:

- The agile software development model was proposed in the mid-1990s to overcome the shortcomings of the waterfall model of development.
- The agile model help a project to adapt to change requests quickly.Thus ,a major aim of the agile model is to facilitate quick project completion.
- Agility is achieved by fitting the process to the project.That is,it gives the required flexibility so that the activities that may not be neccesary for a specific project could be easily removed.
- In an agile model the requirements are decomposed into many small parts that can be incrementally developed.The agile models adopt an incremental and iterative approach.
- Each incremental part is developed over an iteration. Each iteration is intended to be small and easily manageable and lasts for a couple of weeks only.
- At a time,only one increment is planned,developed,and then deployed at the customer site.No long-term plans are made.The time to complete an iteration is

called a time box.The implication of the term time box is that the end date for an iteration does not change.

- A central principle of agile model is the delivery of an increment to the customer after each timebox.

b)

CRITERIA OF A GOOD SRS DOCUMENT:

Following are the characteristics of a good SRS document:

1)Correctness:

User review is used to ensure the correctness of requirements stated in the SRS. SRS is said to be correct if it covers all the requirements that are actually expected from the system.

2)Completeness:

Completeness of SRS indicates every sense of completion including the numbering of all the pages, resolving the to be determined parts to as much extent as possible as well as covering all the functional and non-functional requirements properly.

3)Consistency:

Requirements in SRS are said to be consistent if there are no conflicts between any set of requirements. Examples of conflict include differences in terminologies used at separate places, logical conflicts like time period of report generation, etc.

4)Unambiguousness:

An SRS is said to be unambiguous if all the requirements stated have only 1 interpretation. Some of the ways to prevent unambiguousness include the use of modelling techniques like ER diagrams, proper reviews and buddy checks, etc.

5)Ranking for importance and stability:

There should be a criterion to classify the requirements ,as less or more important or more specifically as desirable or essential. An identifier mark can be used with every requirement to indicate its rank or stability.

6)Modifiability:

SRS should be made as modifiable as possible and should be capable of easily accepting changes to the system to some extent. Modifications should be properly indexed and cross-referenced.

7)Understandable by the customer:

An end user maybe an expert in his/her specific domain but might not be an expert in computer science. Hence, the use of formal notations and symbols should be avoided to as much extent as possible. The language should be kept easy and clear.