SOLUTION-2015

Q1.) a) What is the principle aim of software engineering?

(MUSKAN MOHANTY 1729139)

The principal aim of software engineering is to develop methods and procedures for software development that can scale up for large systems and that can be used consistently to produce high-quality software at low cost and with a small cycle of time. Drawing on computing as one of its foundations, software engineering seeks to develop and use systematic models and reliable techniques to produce high-quality software. These concerns extend from theory and principles to the development practices that are most visible to those outside the discipline.

(b) What are the objective of feasibility study?

The purpose behind a project feasibility study is to know the different variables involved with your business venture and how it will be accepted on the open market along with who will be the target audience. A hospital, for example, aiming to expand, i.e., add an extension to the building, may perform a feasibility study. The study will determine whether the project should go ahead. The people carrying out the study will take into account labour and material costs.

(c)Give two examples of non-functional requirements?

Non-functional requirements specify the system's 'quality characteristics' or 'quality attributes'. Examples are as follows:

- Performance – for example Response Time, Throughput, Utilization, Static Volumetric
- Scalability
- Capacity
- Availability

(d)How do you understand by the term software crisis and what are its causes?

Software crisis is a term used in the early days of computing science for the difficulty of writing useful and efficient computer programs in the required time. The software crisis was due to the rapid increases in computer power and the complexity of the problems that could not be tackled.

(e) How does feature point metric differ from function point metric?

A **function point** is a "unit of measurement" to express the amount of business functionality an information system (as a product) provides to a user. Function points are used to compute a functional size measurement (FSM) of software .

A function point extension called **feature point** is a superset of the function point measure that can be applied to systems and Engineering software applications. The feature point measures and accommodate applications in which Algorithm Complexity is high.

Q.2 (a)Discuss the circumstances under which the following software development models would be appropriate i)Waterfall model ii)Prototyping model iii)Incremental model iv)Spiral model

There are specific circumstances where S/W life cycle models are appropriate:

- The **Waterfall model** is a breakdown of project activities into linear sequential phases, where each phase depends on the deliverables of the previous one and corresponds to a specialisation of tasks. So, waterfall model can be suited to projects where requirements and scope are fixed, the product itself is firm and stable, and the technology is clearly understood.

- Prototyping is defined as the process of developing a working replication of a product or system that has to be engineered. The **Prototyping Model** is one of the most popularly used Software Development Life Cycle Models (SDLC models).This model is used when the customers do not know the exact project requirements beforehand.

- **Incremental Model** is a process of software development where requirements divided into multiple standalone modules of the software development cycle. This model is more flexible – less costly to change scope and requirements. It is easier to test and debug during a smaller iteration. In this model customer can respond to each built. So it suits accordingly to provided circumstances.

- The **spiral model** is a risk-driven software development process model. Based on the unique risk patterns of a given project, the spiral model guides a team to adopt elements of one or more process models, such as incremental, waterfall, or evolutionary prototyping. Thus, it suits the circumstances where:
  - Customer is not sure of their requirements**,** which is usually the case.
  - Requirements are complex and need evaluation to get clarity.
  - New product line which should be released in phases to get enough customer feedback.
  - Significant changes are expected in the product during the development cycle.

b)Suggest a suitable life cycle model for a software project where several kinds of risks are there.Justify your answer and explain all phases of the proposed model with schematic diagram.

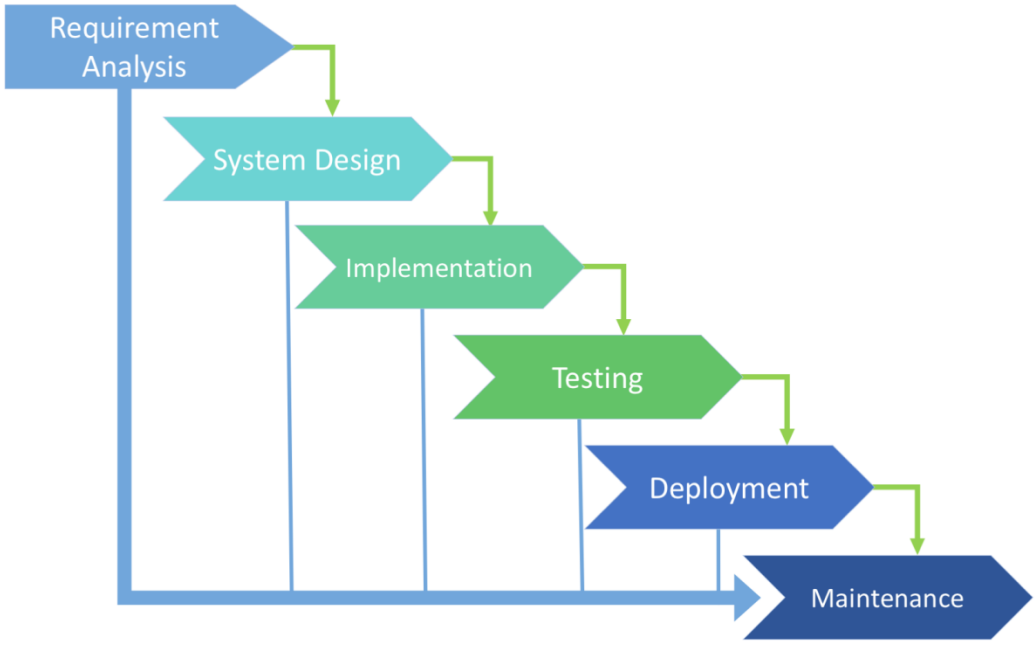
Waterfall model:

Waterfall model – is a cascade SDLC model, in which development process looks like the flow, moving step by step through the phases of analysis, projecting, realization, testing, implementation, and support. This SDLC model includes gradual execution of every stage completely. This process is strictly documented and predefined with features expected to every phase of this software development life cycle model.

Since this model does not allow much reflection or revision, it leads to lots of risks

involved in this model. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

The sequential phases in Waterfall model are –



- **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

- **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

Q.5 KIIT University is planning to develop a payroll software with the following activities.A list of employees with their basic pay is sent to the Accounts section where an Assistant calculates gross pay including standard allowances which are known for each pay slab.Deduction statements for loan repayment, Income tax etc. are sent to another Assistant to match these deduction-slips with the slips bf gross pay and calculate 119t payment.The net-payment slip is used by the third Assistant to write out pay checks for each employee and send to the respective employees.The total pay bill paid is also computed.Write the functional and non-functional re^{s} uirements for the above software product.

(NIBEDITA MISRA 1729141)

NON FUNCTIONAL REQUIREMENTS-

1.Costs

The costs of creating and maintaining a new system are common nonfunctional issues that have an impact on the creation and development of a payroll program.

2.Network Bandwidth

You can have the best payroll program, but unless you have proper bandwidth, the system will not work well.

3.Flexibility

A payroll system must be nimble enough to allow for easy changes. Flexibility is an important nonfunctional requirement.

4.Reliability

Users expect the payroll system to be reliable, a common nonfunctional requirement for most systems.

FUNCTIONAL REQUIREMENTS-

The functional requirements of a payroll system describe what the payroll system is the salary computed for each employee automatically.

Q6) a) Briefly describe the emergence of Software Engineering over the years.

(SWADESH NAYAK 1729085)

Engineering approach to develop software. Building Construction Analogy.
Systematic collection of past experience:

- Techniques,
- Methodologies,
- Guidelines.

The early programmers used an exploratory (also called build and fix) style.
In the build and fix (exploratory) style, normally a `dirty' program is quickly developed. The different imperfections that are subsequently noticed are fixed.

To Fix it, we can imply heavy use of past experience:
1) Past experience is systematically arranged.
2) Theoretical basis and quantitative techniques provided.
3) Trade-off between alternatives.
4) Pragmatic approach to cost-effectiveness.
e.g..
A small program having just a few variables, Is within the easy grasp of an individual. As the number of independent variables in the program increases. It quickly exceeds the grasping power of an individual:
Requires an unduly large effort to master the problem.
Instead of a human, if a machine could be writing (generating) a program, the effort-size curve becomes almost linear when software engineering principles are deployed. Software engineering principles extensively use techniques specifically to overcome the human cognitive limitations.
Mainly two important principles are deployed:

Abstraction: Simplify a problem by omitting unnecessary details. Focus attention on only one aspect of the problem and ignore irrelevant details.

Decomposition: Decompose a problem into many small independent parts. The small parts are then taken up one by one and solved separately.

a. The idea is that each small part would be easy to grasp and can be easily solved.
b. The full problem is solved when all the parts are solved.
c. As an outcome of the increased use of Software Engineering skills we have:
d. Exponential growth in complexity and difficulty level with size.
e. The ad hoc approach breaks down when size of software increases
f. Ability to solve complex programming problems
g. How to break large projects into smaller and manageable parts.
h. Also learn techniques of: Specification, design, user interface development, testing, project management, etc.
The Overall outcome will be

i. To acquire skills to be a better programmer
ii. Higher Productivity
iii. Better Quality Programs

b)Discuss the res onsibilities of a Software Project Manager.
8 key roles and responsibilities Of PM:
1. Activity and resource planning:

Planning is instrumental for meeting project deadlines, and many projects fail due to poor planning.
First and foremost, good project managers define the project’s scope and determine available resources.
Good project managers know how to realistically set time estimates and evaluate the team or teams’ capabilities. They then create a clear and concise plan to both execute the project and monitor its progress. Projects are naturally unpredictable, so

project managers know how to make adjustments along the way as needed before the project reaches its final stages.

2. Organizing and motivating a project team

They develop clear, straightforward plans that stimulate their teams to reach their full potential.
They cut down on bureaucracy and steer their teams down a clear path to the final goal.

3. Controlling time management

Clients usually judge a project's success or failure on whether it has been delivered on time.
Therefore, meeting deadlines is non-negotiable. Project managers know how to set realistic deadlines, and
how to communicate them consistently to their teams.
They know how to effectively do the following:

- Define activity
- Sequence activity
- Estimate the duration of activity
- Develop a schedule
- Maintain a schedule

4. Cost estimating and developing the budget

Project managers know how to keep a project within its set budget. Even if a project meets a client's expectations and is delivered on time, it will still be a failure if it goes wildly over-budget.Project managers frequently review the budget and plan ahead to avoid massive budget overruns.

5. Ensuring customer satisfaction

In the end, a project is only a success if the customer is happy. One of the key responsibilities of every project manager is to minimize uncertainty, avoid any unwanted surprises and involve their clients in the project as much as is reasonably possible.

6. Analysing and managing project risk

The bigger the project is, the more likely there are to be hurdles and pitfalls that weren't part of the initial plan. Hiccups are inevitable, but Project managers know how meticulously and almost intuitively, identify and evaluate potential risks before the project begins. They know how to then avoid risks or at least minimize their impact.

7. Monitoring progress

During the initial stages, project managers and their teams have a clear vision and high hopes of producing the desired result.
However, the path to the finish line is never without some bumps along the way. When things don't go according to a plan, a project manager needs to monitor and analyse both expenditures and team performance and to always efficiently take corrective measures.

8. Managing reports and necessary documentation

Finally, experienced project managers know how essential final reports and proper documentation are.
Project managers can present comprehensive reports documenting that all project requirements were fulfilled, as well as the projects' history, including what was done, who was involved, and what could be done better in the future.
Project managers are integral parts of almost every kind of organization—from small agencies with only one project manager guiding a handful of projects to multinational IT companies that employ highly specialized project managers placed in charge of ambitious projects.

Q.7 Write short notes on any two:
A) Agile methodologies
B) Gantt chart and Activity networks
C) Delphi technique
D) Requirement engineering tasks
E) Disadvantages of estimation of cost of software based on LOC

(NILANJAN ROY 1729143)

Requirement engineering consists of seven different tasks as follow:

1. Inception

- Inception is a task where the requirement engineering asks a set of questions to establish a software process.
- In this task, it understands the problem and evaluates with the proper solution.
- It collaborates with the relationship between the customer and the developer.
- The developer and customer decide the overall scope and the nature of the question.

2. Elicitation
Elicitation means to find the requirements from anybody.
The requirements are difficult because the **following problems occur in elicitation**.

Problem of scope: The customer give the unnecessary technical detail rather than clarity of the overall system objective.

Problem of understanding: Poor understanding between the customer and the developer regarding various aspect of the project like capability, limitation of the computing environment.

Problem of volatility: In this problem, the requirements change from time to time and it is difficult while developing the project.

3. Elaboration

- In this task, the information taken from user during inception and elaboration and are expanded and refined in elaboration.

- Its main task is developing pure model of software using functions, feature and constraints of a software.

4. Negotiation

- In negotiation task, a software engineer decides the how will the project be achieved with limited business resources.
- To create rough guesses of development and access the impact of the requirement on the project cost and delivery time.

5. Specification

- In this task, the requirement engineer constructs a final work product.
- The work product is in the form of software requirement specification.
- In this task, formalize the requirement of the proposed software such as informative, functional and behavioral.
- The requirement are formalize in both graphical and textual formats.

6. Validation

- The work product is built as an output of the requirement engineering and that is accessed for the quality through a validation step.
- The formal technical reviews from the software engineer, customer and other stakeholders helps for the primary requirements validation mechanism.

7. Requirement management

- It is a set of activities that help the project team to identify, control and track the requirements and changes can be made to the requirements at any time of the ongoing project.
- These tasks start with the identification and assign a unique identifier to each of the requirement.
- After finalizing the requirement traceability table is developed.
- The examples of traceability table are the features, sources, dependencies, subsystems and interface of the requirement.

E)DISADVANTAGES

- Different programming languages contains different number of lines.
- No proper industry standard exist for this technique.
- It is difficult to estimate the size using this technique in early stages of project.