Assignment 1

Software Engineering Concepts :

Section A:

Questions

Note : Explain with examples

What are the reasons of a successful and unsuccessful software project?

What Causes Your Projects to Fail?

Reason 1: Poor planning

Reason 2: Lack of leadership

Reason 3: Poor communication

Reason 4: Inadequate use of resources

Reason 5: Inability to overcome challenges

How to make your software projects succeed?

#1 Define the project’s goals

#2 Establish a project team structure

#3 Create a project plan

#4 Set realistic deadlines

#5 Communicate effectively with stakeholders

#6 Analyze the project’s challenges and opportunities

A few more tips to help your software development projects succeed:

1. Define the project scope and goals clearly from the start

2. Assemble a solid and experienced project team

3. Put together a comprehensive project plan that details all aspects of the project, from start to finish

4. Keep communication channels open throughout the project, between all team members and stakeholders

===========================================================

What types of problems may arise if a software project is developed on ad hoc

basis?

## The meaning of word Ad-hoc is something which is not in order or not organised or unstructured. In the similar note the Ad-hoc testing is nothing but a type of black box testing or behavioural testing.

## Ad-hoc testing is carried out without following any formal process like requirement documents, test plan, test cases, etc. Similarly while executing the ad-hoc testing there is NO formal process of testing which can be documented.

======================================================================

Provide three examples of software projects that would be amenable to the

waterfall model. Be specific.

1. An Operating System, as the various specific parts of the OS could be developed as the user requires them
2. A Graphical User Interface, similar to the OS, the GUI can be created according to the customer’s requirements and approval.
3. A Web Application, a a base application can be developed and delivered, followed by any number of additional plug-ins that the customer would want for additional functionality

========================================================================

Provide three examples of software projects that would be amenable to the

prototyping model. Be specific.

**Answer:**

1) fingerprint voting system

2) fingerprint. based ATM system

3) weather forecasting system

**Explanation:**

**1) Fingerprint voting system-**Fingerprint Voting System was implemented with the Arduino technology. In this System a voter can poll his vote easily. In this database server all voters' information was stored to register in this system, the voter should fill a registration form with the help of a user id and password. This information will be checked by the database server. Because all the information about the voter would be already there is anything wrong, the system will not allow the voter to poll his or her vote. This system is helpful to the voter's decreases the time of voting process **.**

**2) Fingerprint based ATM system-**Fingerprint Based ATM is a desktop application where fingerprint of the user is used as a authentication. The finger print minutiae features are different for each human being so the user can be identified uniquely. Instead of using ATM card Fingerprint based ATM is safer and secure. There is no worry of losing ATM card and no need to carry ATM card in your wallet.

**3) Weather forecasting system-** The global weather forecasting system and solutions market size was valued at USD 2.51 billion in 2016. It is expected to post a CAGR of 7.1% over the forecast period. These systems help enterprises in gaining real-time insights into atmospheric conditions, which in turn, enables enterprises to carefully plan all weather-sensitive operations to ensure security, sustainability, safety, and cost efficiency. Factors such as growing sea and air transportation, increasing stringency of norms pertaining to environmental protection, and high dependency on rainfall for water supply are among the key trends stimulating market growth.

What process adaptations are required if the prototype will evolve into a delivery

system or product?

If a prototype is evolved into a delivery system or product, it begins with communication.The software engineer and customer meet and define the overall objectives for the software, identify whatever requirements are known, and outline areas where further definition is mandatory.

=================================================================================

Provide three examples of software projects that would be amenable to the

incremental model. Be specific.

Three examples of software project that are amenable to the increment model are smart sheet, work zone, and intervals.

Explanation:

• Incremental model is one which a software is developed through steps which begins from the initiation, development, design, and implementation.

• In the incremental model, the software development process are broken down into numerous steps.

• The incremental model uses an iterative process in the development of a particular software for an industry.

===================================================================================

As you move outward along the spiral process flow, what can you say about the

software that is being developed or maintained?

As you move outward along the spiral process flow, what can you say about the software that is being developed or maintained?

ANS: As work moves outward on the spiral the product moves toward a more complete state and the level of abstraction at which work is performed is reduced (implementation specific work accelerates as we move further from the origin).

===================================================

Aim : To study Software Development Life Cycle ( SDLC) Models

1. Team Name , Number

2. Team Member’s Name

3. Model

(Introduction – Purpose of Model, Diagram, Phase Diagram , Advantages,

Disadvantages, Realtime Examples)

3.1 Waterfall Model

3.2 Rapid Application Development

3.3 Incremental Model

3.4 Spiral Model

3.5 Agile Model

3.6 ‘V’ shaped model

4. Comparison between the models

Software Engineering Concepts: Assignment 1

Aim:

To study Software Development Life Cycle ( SDLC) Models

1. Team Number and Team Name:

Team No 14. Technocrats

2. Team Members:

1. Suhas Patil

2. Sujit Ghongade

3. Sumit Vyas

4. Suniti Jha

5. Supriya Tiwari

6. Suraj Tavhare

7. Surajkumar Mahajan

8. Sushant Malgave

9. Swapnali Chavan

10. Swati Agale

3. Software Development Life Cycle ( SDLC) Models:

3.1 Waterfall Model

Winston Royce introduced the Waterfall Model in 1970.This model has five

phases: Requirements analysis and specification, design, implementation, and unit

testing, integration and system testing, and operation and maintenance. The steps

always follow in this order and do not overlap. The developer must complete every

phase before the next phase begins. This model is named "Waterfall Model", because

its diagrammatic representation resembles a cascade of waterfalls.

Fig 3.1.1: Waterfall Model

Phases of Waterfall Model

1. Requirements analysis and specification phase: The aim of this phase is to

understand the exact requirements of the customer and to document them properly.

Both the customer and the software developer work together so as to document all

the functions, performance, and interfacing requirement of the software. It describes

the "what" of the system to be produced and not "how." In this phase, a large

document called Software Requirement Specification (SRS) document is created

which contained a detailed description of what the system will do in the common

language.

2. Design Phase: This phase aims to transform the requirements gathered in the SRS

into a suitable form which permits further coding in a programming language. It

defines the overall software architecture together with high level and detailed

design. All this work is documented as a Software Design Document (SDD).

3. Implementation and unit testing: During this phase, design is implemented. If the

SDD is complete, the implementation or coding phase proceeds smoothly, because

all the information needed by software developers is contained in the SDD. During

testing, the code is thoroughly examined and modified. Small modules are tested in

isolation initially. After that these modules are tested by writing some overhead

code to check the interaction between these modules and the flow of intermediate

output.

4. Integration and System Testing: This phase is highly crucial as the quality of the

end product is determined by the effectiveness of the testing carried out. The better

output will lead to satisfied customers, lower maintenance costs, and accurate

results. Unit testing determines the efficiency of individual modules. However, in

this phase, the modules are tested for their interactions with each other and with the

system.

5. Operation and maintenance phase: Maintenance is the task performed by every

user once the software has been delivered to the customer, installed, and

operational.

Fig 3.1.2: Phase Diagram

Purpose:

Some Circumstances where the use of the Waterfall model is most suited are:

• When the requirements are constant and not changed regularly.

• A project is short

• The situation is calm

• Where the tools and technology used is consistent and is not changing

• When resources are well prepared and are available to use.

Advantages of Waterfall model:

• This model is simple to implement also the number of resources that are required for it

is minimal.

• The requirements are simple and explicitly declared; they remain unchanged during the

entire project development.

• The start and end points for each phase is fixed, which makes it easy to cover progress.

• The release date for the complete product, as well as its final cost, can be determined

before development.

• It gives easy to control and clarity for the customer due to a strict reporting system.

Disadvantages of Waterfall model:

• In this model, the risk factor is higher, so this model is not suitable for more significant

and complex projects.

• This model cannot accept the changes in requirements during development.

• It becomes tough to go back to the phase. For example, if the application has now

shifted to the coding phase, and there is a change in requirement, It becomes tough to

go back and change it.

• Since the testing done at a later stage, it does not allow identifying the challenges and

risks in the earlier phase, so the risk reduction strategy is difficult to prepare.

Real Time Example:

• Use to develop enterprise applications like Customer Relationship Management (CRM)

systems

• Human Resource Management Systems (HRMS)

• Supply Chain Management Systems

• Inventory Management Systems

• Point of Sales (POS) systems for Retail chains

• Development of Department Of Defence (DOD)

• military and aircraft programs followed Waterfall model in many organizations

3.2 Rapid Application Development

RAD model is Rapid Application Development model. It is a type of incremental

model. In RAD model the components or functions are developed in parallel as if they were

mini projects. The developments are time boxed, delivered and then assembled into a working

prototype.

This can quickly give the customer something to see and use and to provide feedback

regarding the delivery and their requirements.

Fig 3.2.1: RAD Model

Phases in the rapid application development (RAD) model:

1. Business modeling: The information flow is identified between various business

functions.

2. Data modeling: Information gathered from business modeling is used to define data

objects that are needed for the business.

3. Process modeling: Data objects defined in data modeling are converted to achieve the

business information flow to achieve some specific business objective. Description are

identified and created for CRUD of data objects.

4. Application generation: Automated tools are used to convert process models into code

and the actual system.

5. Testing and turnover: Test new components and all the interfaces.

Purpose;

• RAD should be used when there is a need to create a system that can be modularized in

2-3 months of time.

• It should be used if there’s high availability of designers for modeling and the budget

is high enough to afford their cost along with the cost of automated code generating

tools.

• RAD SDLC model should be chosen only if resources with high business knowledge

are available and there is a need to produce the system in a short span of time (2-3

months).

Advantages of the RAD model:

• Reduced development time.

• Increases reusability of components

• Quick initial reviews occur

• Encourages customer feedback

• Integration from very beginning solves a lot of integration issues.

Disadvantages of RAD model:

• Depends on strong team and individual performances for identifying business

requirements.

• Only system that can be modularized can be built using RAD

• Requires highly skilled developers/designers.

• High dependency on modeling skills

• Inapplicable to cheaper projects as cost of modeling and automated code generation is

very high.

Real Time Example:

• General Goals - Bonus Subsystem

• Current System - Vehicle Subsystem

• Proposed System - VIP Subsystem

➢ User Interface and Human Factors - Maintenance Subsystem

➢ Documentation - Travel Subsystem

➢ Hardware Consideration - Logbook Subsystem

➢ Performance Characteristics - Bonus Subsystem

➢ Screen Mock up - Maintenance Subsystem

➢ Navigational Path for the Web Application - Bonus Subsystem

➢ Dynamic Models - Logbook Subsystem

3.3 Incremental Model

In incremental model the whole requirement is divided into various builds. Multiple

development cycles take place here, making the life cycle a “multi-waterfall” cycle. Cycles

are divided up into smaller, more easily managed modules. Incremental model is a type of

software development model like V-model, Agile model etc.

In this model, each module passes through the requirements, design, implementation

and testing phases. A working version of software is produced during the first module, so you

have working software early on during the software life cycle. Each subsequent release of the

module adds function to the previous release. The process continues till the complete system

is achieved.

Fig 3.3.1: Incremental Model.

Purpose:

• This model can be used when the requirements of the complete system are clearly

defined and understood.

• Major requirements must be defined; however, some details can evolve with time.

• There is a need to get a product to the market early.

• A new technology is being used

• Resources with needed skill set are not available

• There are some high risk features and goals.

Advantages of Incremental model:

• Generates working software quickly and early during the software life 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.

• Lowers initial delivery cost.

• Easier to manage risk because risky pieces are identified and handled during it’d

iteration.

Disadvantages of Incremental model:

• Needs good planning and design.

• Needs a clear and complete definition of the whole system before it can be broken down

and built incrementally.

• Total cost is higher than waterfall.

Real Time Example:

• This model can be used when the requirements of the complete system are clearly

defined and understood.

• Major requirements must be defined; however, some details can evolve with time.

• There is a need to get a product to the market early.

• A new technology is being used.

• Resources with needed skill set are not available.

• There are some high-risk features and goals.

3.4 Spiral Model

The spiral model, initially proposed by Boehm, is an evolutionary software process

model that couples the iterative feature of prototyping with the controlled and systematic

aspects of the linear sequential model. It implements the potential for rapid development of

new versions of the software. Using the spiral model, the software is developed in a series of

incremental releases. During the early iterations, the additional release may be a paper model

or prototype. During later iterations, more and more complete versions of the engineered

system are produced.

Fig 3.4.1: Spiral Model

Phases in the Spiral Model:

Each cycle in the spiral is divided into four parts:

• Objective setting: Each cycle in the spiral starts with the identification of purpose for

that cycle, the various alternatives that are possible for achieving the targets, and the

constraints that exists.

• Risk Assessment and reduction: The next phase in the cycle is to calculate these

various alternatives based on the goals and constraints. The focus of evaluation in this

stage is located on the risk perception for the project.

• Development and validation: The next phase is to develop strategies that resolve

uncertainties and risks. This process may include activities such as benchmarking,

simulation, and prototyping.

• Planning: Finally, the next step is planned. The project is reviewed, and a choice made

whether to continue with a further period of the spiral. If it is determined to keep, plans

are drawn up for the next step of the project.

The development phase depends on the remaining risks. For example, if performance or userinterface risks are treated more essential than the program development risks, the next phase

may be an evolutionary development that includes developing a more detailed prototype for

solving the risks.

The risk-driven feature of the spiral model allows it to accommodate any mixture of a

specification-oriented, prototype-oriented, simulation-oriented, or another type of approach.

An essential element of the model is that each period of the spiral is completed by a review that

includes all the products developed during that cycle, including plans for the next cycle. The

spiral model works for development as well as enhancement projects.

Purpose:

• When deliverance is required to be frequent.

• When the project is large

• When requirements are unclear and complex

• When changes may require at any time

• Large and high budget projects

Advantages of Spiral Model:

• High amount of risk analysis

• Useful for large and mission-critical projects.

Disadvantages of Spiral Model:

• Can be a costly model to use.

• Risk analysis needed highly particular expertise

• Doesn't work well for smaller projects.

Real Time Example:

• Working on the missiles or satellites is the real time example of a spiral 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.

• Gantt chart software – GanttPRO a tool for simple task handling.

• Evolution of Microsoft Windows operating system.

3.5 Agile Model

Agile model believes that every project needs to be handled differently and the existing

methods need to be tailored to best suit the project requirements. In Agile, the tasks are divided

to time boxes (small time frames) to deliver specific features for a release.

Iterative approach is taken and working software build is delivered after each iteration. Each

build is incremental in terms of features; the final build holds all the features required by the

customer.

Fig 3.4.1 Agile Model

Following are the Agile Manifesto principles −

• Individuals and interactions − In Agile development, self-organization and

motivation are important, as are interactions like co-location and pair programming.

• Working software − Demo working software is considered the best means of

communication with the customers to understand their requirements, instead of just

depending on documentation.

• Customer collaboration − As the requirements cannot be gathered completely in the

beginning of the project due to various factors, continuous customer interaction is very

important to get proper product requirements.

• Responding to change − Agile Development is focused on quick responses to change

and continuous development.

Purpose:

• Team members enjoy development work, and like to see their work used and valued.

• Scrum makes this alignment easier by providing frequent opportunities to re-prioritize

work, to ensure maximum delivery of value.

• Scrum provides high visibility into the state of a development project, on a daily basis.

• Project Manager tremendous awareness about the state of the project at all times.

Advantages of Agile Model:

• Is a very realistic approach to software development.

• Promotes teamwork and cross training.

• Functionality can be developed rapidly and demonstrated.

• Resource requirements are minimum.

• Suitable for fixed or changing requirements

• Good model for environments that change steadily.

• Minimal rules, documentation easily employed.

• Little or no planning required.

• Easy to manage.

• Gives flexibility to developers.

Disadvantages of Agile Model:

• Not suitable for handling complex dependencies.

• More risk of sustainability, maintainability and extensibility.

• An overall plan, an agile leader and agile PM practice is a must without which it will

not work.

Real Time Example:

• Restaurant orders:

➢ Preparation of some of the food before opening the shop (sprint planning)

➢ continuous delivery of orders (adhoc stories)

➢ number of successful orders (velocity)

• Cricket:

➢ over (sprint length)

➢ team (scrum team self-sufficient)

➢ Run rate (velocity)

➢ Captain/ coach (scrum master)

3.6 ‘V’ Model:

V- model means Verification and Validation model. Just like the waterfall model, the

V-Shaped life cycle is a sequential path of execution of processes. Each phase must be

completed before the next phase begins.

Fig 3.6.1 ‘V’ Model

Advantages of V-model:

• Simple and easy to use.

• Testing activities like planning, test designing happens well before coding. This saves

a lot of time. Hence higher chance of success over the waterfall model.

• Proactive defect tracking – that is defects are found at early stage.

• Avoids the downward flow of the defects.

• Works well for small projects where requirements are easily understood.

Disadvantages of V-model:

• Very rigid and least flexible.

• Software is developed during the implementation phase, so no early prototypes of the

software are produced.

• If any changes happen in midway, then the test documents along with requirement

documents has to be updated.

Purpose:

• The V-shaped model should be used for small to medium sized projects where

requirements are clearly defined and fixed.

• The V-Shaped model should be chosen when ample technical resources are available

with needed technical expertise.

Real Time Example:

• IT projects by federal agencies

• public-sector software projects

• In electronic and mechanical system in research and science

• software for agencies and ministries

4. Comparison between the Models

Properties of

Model

Water-Fall

Model

Incremental

Model

Spiral

Model

RAD

Model

Agile Model V shaped

Model

Planning in early

stage

Yes Yes Yes No No Yes

Returning to an

earlier phase No Yes Yes Yes

Yes No

Handle LargeProject

Not

Appropriate Not Appropriate Appropriate Not

Appropriate

Yes Not

appropriate

Detailed

Documentation Necessary Yes, but not

much Yes Limited

Yes Yes

Cost Low Low Expensive Low Low Expensive

Requirement

Specifications Beginning Beginning Beginning Time boxed

release

Time boxed

release

Beginning

Flexibility to

change Difficult Easy Easy Easy Easy Difficult

User

Involvement

Only at

beginning Intermediate High Only at the

beginning

High

Only at the

beginning

Maintenance Least Promotes

Maintainability Typical Easily

Maintained

Easily

Maintained

Easily

maintained

Duration Long Very long Long Short Depends on

project

Long

Risk

Involvement High Low Medium to

high risk Low Low Moderate to

high

Framework Type Linear Linear +

Iterative

Linear +

Iterative Linear Iterative and

incremental

Sequential

Testing

After

completion of

coding phase

After every

iteration

At the end

of the

engineering

phase

After

completion

of coding

After every

iteration

After every

iteration

Overlapping

Phases No

Yes (As parallel

development is

there)

No Yes

No

No

Re-usability Least possible To some extent

To some

extent

Yes

Yes No

Time-Frame Very Long Long Long Short Long Ideal time

Working

software

availability

At the end of

the life-cycle

At the end of

every iteration

At the end

of every

iteration

At the end

of the life

cycle

At the end of

every iteration

At the end of

every iteration

Team size Large Team Not Large Team Large Team Small Team Large team Large team