Iqra National University Peshawar Pakistan

Department of Computer Science

Spring Semester Final Term Exam , June 2020

Subject:	**Topics In Software Engineering**	Issue Date:	**25/June/2020**
Program:	**MS (CS)**	Submission Date:	**25/June/2020**
Teacher Name:	**Dr. Fazal-e-Malik**		
Student Name	**Rooh Ullah Jan**	ID	**6611**

Q.1 **Do we need prototyping? Explain the risks in prototyping.**

Prototype:

A prototype is an early sample, model, or release of a product built to test a concept or process. It is a term used in a variety of contexts, including semantics, design, electronics, and software programming. A prototype is generally used to evaluate a new design to enhance precision by system analysts and users. In some design workflow models, creating a prototype (a process sometimes called materialization) is the step between the formalization and the evaluation of an idea. The prototype is an important draft design that precedes the development of the original design layout. It is intended not only to show the structure of the future site, but also the site map, the interrelation of its main pages. Prototyping is a process designed to significantly reduce the time to develop a site, due to the focus of the designer on the main functional and marketing factors.

Yes we need prototyping because:

- Prototyping involves building simple & quick implementations of parts of the product.
- The major reason for prototyping is to reduce some type of risk associated with the project.
- Prototyping serves to provide specifications for a real, working system rather than a theoretical one.[
- The prototype gives the customer a complete idea of how the site will look like in the final result.

- Prototyping allows you to streamline the design development process, focusing on important interface elements.
- A site's sketch is an irreplaceable thing if the client still does not fully understand what he expects from the site, what functions each page will carry. Careful planning at the prototyping stage makes it possible to avoid global changes in the finished layout.
- At the prototyping stage, it is possible to identify unnecessary elements that are best abandoned.
- This process significantly reduces the workload of the designer in developing the project, and thus saves the customer's money.
- Having a prototype in hand, the designer and customer more clearly represent the final result.
- Prototype development involves the involvement of the customer, contributes to more productive work, process consistency. In this case, the prototype is created very simply. Some species can be outlined already at the first meeting with the client, specifying certain details of the design.

Risks in Prototyping:

First risk is that users may expect to have the same functionality in the final solution as what's been indicated by the prototype. Manage the users' expectations as there are assumptions and constraints which surface between the development of the prototype and the final solution.

Another risk of the project with the completed prototyping stage are

significantly lower than the projects in which prototyping was not carried out.

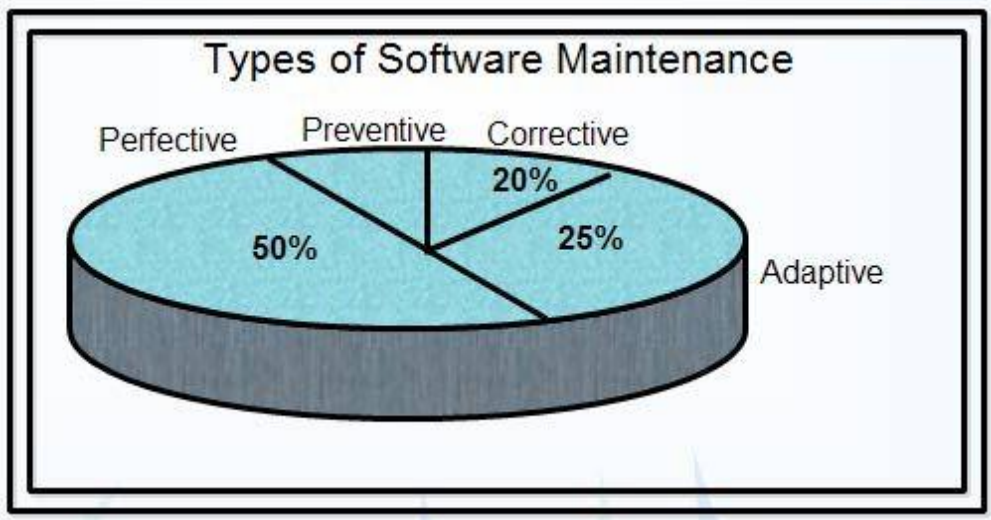
The biggest risk is that anyone who is interested in the project after facing a working prototype will decide that the final product is almost ready.
Everyone who sees a prototype must understand its purpose and the limits of its application. Web testing service comes in handy when you want to improve design and content of e-commerce site.
Do not let the fears associated with premature release of the product draw you away from creating a prototype. Explain to everyone that you are not going to release it as the final product.

Q.2 **Explain the distribution of maintenance efforts in terms of :**

a) **Fault Repair**
b) **Software Adaptation**
c) **Functionality Addition or Modification**

(a) Fault Repair:

Fault repair is also known as Corrective maintenance deals with the repair of faults or defects found in day-today system functions. A defect can result due to errors in software design, logic and coding. Design errors occur when changes made to the software are incorrect, incomplete, wrongly communicated, or the change request is misunderstood. Logical errors result from invalid tests and conclusions, incorrect implementation of design specifications, faulty logic flow, or incomplete test of data. All these errors, referred to as residual errors, prevent the software from conforming to its agreed specifications. Note that the need for corrective maintenance/Fault Repair is usually initiated by bug reports drawn by the users.
Corrective maintenance/Fault Repair is concerned with fixing errors that are observed when the software is in use.



(b)Software Adaptation:

In terms of software maintenance software Adaptation is concerned with the change in the software that takes place to make the software adaptable to new environment such as to run the software on a new operating system.

Software Adaptation is the implementation of changes in a part of the system, which has been affected by a change that occurred in some other part of the system. Adaptive maintenance consists of adapting software to changes in the environment such as the hardware or the operating system. The term environment in this context refers to the conditions and the influences which act (from outside) on the system. For example, business rules, work patterns, and government policies have a significant impact on the software system.

For instance, a government policy to use a single 'European currency' will have a significant effect on the software system. An acceptance of this change will require banks in various member countries to make significant changes in their software systems to accommodate this currency. Adaptive maintenance accounts for 25% of all the maintenance activities.

(b) Functionality Addition or Modification:

Functionality addition or modification is also known as perfective maintenance concerned with the change in the software that occurs while adding new functionalities in the software.

Functionality addition or modification mainly deals with implementing new or changed user requirements. Functionality addition or modification involves making functional enhancements to the system in addition to the activities to

increase the system's performance even when the changes have not been suggested by faults. This includes enhancing both the function and efficiency of the code and changing the functionalities of the system as per the users' changing needs.

Examples of Functionality addition or modification include modifying the payroll program to incorporate a new union settlement and adding a new report in the sales analysis system. Functionality addition or modification accounts for 50%, that is, the largest of all the maintenance activities.

Q.3 **Re-usability can be done at various levels? Explain at least Three (3) levels.**

Software Re-usability:

In computer science and software engineering, reusability is the use of existing *assets* in some form within the software product development process; these *assets* are products and by-products of the software development life cycle and include code, software components, test suites, designs and documentation. Software reusability more specifically refers to design features of a software element (or collection of software elements) that enhance its suitability for reuse.

Software reusability is an attribute in which software or its module is reused with very little or no modification. For any organization, improving the business performance means performing their software development. Software reusability offers great potential of significant gains for an organization, by reducing cost and effort, and accelerating the Time to Market of software products

Software Re-usability can be done at different level like Application level, Components level, Software product lines and Module level.

Three level of Re-usability :

1) Components level :

A component is a part of software program code, which executes an independent task in the system. It can be a small module or sub-system itself.

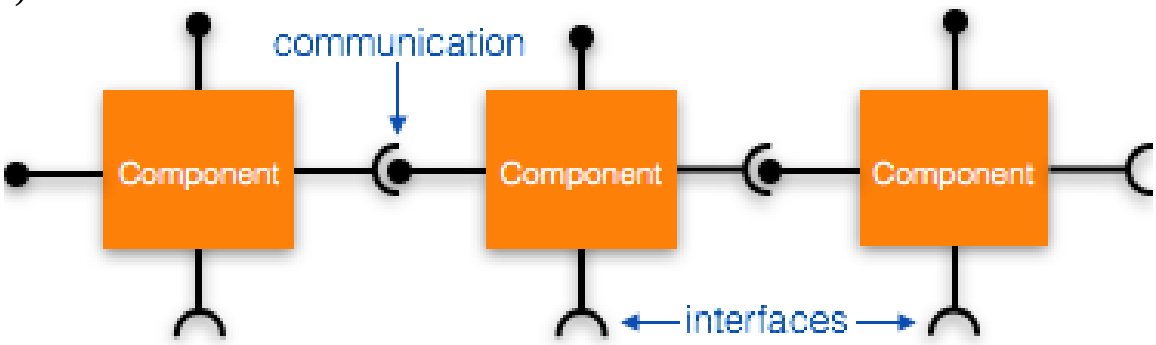
Example:

The login procedures used on the web can be considered as components, printing system in software can be seen as a component of the software.

Components have no dependency i.e. they work independently and can perform tasks without depending on other modules.

In modular programming, the modules are coded to perform specific tasks which can be used across number of other software programs.

There is a whole new vertical (involving different levels), which is based on re-use of software component, and is known as Component Based Software Engineering (CBSE).



2) Application level:

An application level re-use is that can be adapted for different customers without changing the source code of the system. Application systems have generic features and so can be used/reused in different environments. Application system products are adapted by using built-in configuration mechanisms that allow the functionality of the system to be tailored to specific customer needs.

Benefits of application system/level reuse:

- As with other types of reuse, more rapid deployment of a reliable system may be possible.
- It is possible to see what functionality is provided by the applications and so it is easier to judge whether or not they are likely to be suitable.
- Some development risks are avoided by using existing software. However, this approach has its own risks, as I discuss below.
- Businesses can focus on their core activity without having to devote a lot of resources to IT systems development.
- As operating platforms evolve, technology updates may be simplified as these are the responsibility of the COTS product vendor rather than the customer.

3) Software Product Lines:

Software product lines or application families are applications with generic functionality that can be adapted and configured for use in a specific context. A software product line is a set of applications with a common architecture and shared components, with each application specialized to reflect different requirements. Examples: a mobile operating system that works on different hardware models, a software line for a family of printers

with varying features. Adaptation of a software line may involve:

- Component and system configuration;
- Adding new components to the system;
- Selecting from a library of existing components;
- Modifying components to meet new requirements.

The base application of a software product line includes:

- Core components that provide infrastructure support. These are not usually modified when developing a new instance of the product line.
- Configurable components that may be modified and configured to specialize them to a new application. Sometimes, it is possible to reconfigure these components without changing their code by using a built-in component configuration language.
- Specialized, domain-specific components some or all of which may be replaced when a new instance of a product line is created.

Q.4 **Provision of failure-free software is the main objective of the software engineering but this failure of software is due to different reasons. Explain these different reasons of software failure.**

Software project failures are common. Even though the reasons for failures have been widely studied, the analysis of their causal relationships is lacking. This creates an illusion that the causes of project failures are unrelated.

The possibility of software projects failing can be attributed to various reasons - costs, scheduling and quality issues, and/ or achievement of objectives. This pose a serious threat to companies wishing to outsource their software development needs, as software project management failures often cause huge losses in time and money, and can prove to be detrimental to a company's growth and development.

Most software projects fail completely or partial because they don't meet all their requirements. These requirements can be the cost, schedule, quality, or requirements objectives. According to many studies, failure rate of software projects ranges between 50% – 80%. There are a variety of causes for software failures but the most common are:

- Lack of user participation
- Changing requirements
- Unrealistic or unarticulated project goals
- Inaccurate estimates of needed resources

- Badly defined system requirements
- Poor reporting of the project's status
- Lack of resources
- Unmanaged risks
- Poor communication among customers, developers, and users
- Use of immature technology
- Inability to handle the project's complexity
- Sloppy development practices
- Poor Project Management
- Stakeholder politics
- Lack of Stakeholder involvement
- Commercial pressures
- Absence of a well formed charter
- Ineffective Integrated change control
- Inefficient or often wrong requirement definitions
- Uncontrolled change(s)
- Ineffective leadership
- Communication gap
- No Subject knowledge
- UnExperience in the field
- 33% of project failure is due to lack of involvement of senior management.
- Another 33% is due to requirement changes.
- 31% of projects fail to deliver within budget
- 31% of projects fail to deliver within budget.
- Request changes after developement in development phase.
- And lack of latest technology prop

Also

The major reason why software projects, or any other type of project fails is mostly caused by the people involved. Lack of communication, different points of view, different interests towards the project, different motivation and so on.

Q.5 **The study of software reliability can be categorized into three (3) parts: modeling, measurement and improvement. Explain them in terms of:**

a) **Software Reliability Models**
b) **Software Reliability Metrics**
c) **Software Reliability Improvement**

a)Software Reliability Models:

A software reliability model indicates the form of a random process that defines the behavior of software failures to time.

Software reliability models have appeared as people try to understand the features of how and why software fails, and attempt to quantify software reliability.

The reliability process in generic terms is a model of the reliability- oriented aspects of software development, operations, and maintenance. Quantities of interest in a project reliability profile include artifacts, errors, defects, corrections, faults, tests, failures, outages, repairs, validation, and expenditures of resources, such as CPU time, manpower effort and schedule time. The activities relating to reliability are grouped into classes:

Combination:

Integrates reusable documentation and code components with new documentation and code components

Construction:

Generates new documentation and code artifacts

Correction:

Analyzes and removes defects in documentation and code using static analysis of artifacts.

Preparation:

Generates test plans and test cases, and readies them for execution.

b) Software Reliability Metrics:

Software Reliability metrics are used to quantitatively expressed the reliability of the software product. The option of which metric is to be used depends upon the type of system to which it applies & the requirements of the application domain.

Software Reliability Measurement is not an exact science. Though frustrating, the quest of quantifying software reliability has never ceased. Until now, we still have no good way of measuring software reliability. Measuring software reliability remains a difficult problem because we don't have a good understanding of the nature of software. There is no clear definition to what aspects are related to software reliability. We cannot find a suitable way to measure software reliability, and most of the aspects related to software reliability. It is tempting to measure something related to reliability to reflect the characteristics, if we can not measure reliability directly. The current practices of software reliability measurement can be divided into four categories.

d) Software Reliability Improvement:

Software reliability improvement is necessary & hard to achieve. It can be improved by sufficient understanding of software reliability, characteristics of software & sound software design. Complete testing of the software is not possible; however sufficient testing & proper maintenance will improve software reliability to great extent.

Software Reliability is an important facet of software quality. Software reliability is the probability of the failure free operation of a computer program for a specified period of time in a specified environment. Software Reliability is dynamic and stochastic. It differs from the hardware reliability in that it reflects design perfection, rather than manufacturing perfection. In this explanation Software Reliability which can be categorized into: modeling, measurement and improvement, and then examines different modeling technique and metrics for software reliability, however, there is no single model that is universal to all the situations. The article will also

provide an overview of improving software reliability and then provides various ways to improve software reliability in the life cycle of software development.

Good engineering methods can largely improve software reliability. In real situations, it is not possible to eliminate all the bugs in the software; however, by applying sound software engineering principles software reliability can be improved to a great extent. The application of systematic, disciplined, quantifiable approach to the development operation and maintenance of software will produce economically software that is reliable and works efficiently on real machines.

****