Iqra National University Peshawar Pakistan

Department of Computer Science

Spring Semester Final Term Exam , June 2020

Subject:	**Software Design**	Issue Date:	**24/June/2020**
Program:	**MS (CS)**	Submission Date:	**24/June/2020**
Teacher Name:	**Dr. Fazal-e-Malik**		
Student Name	**Rooh Ullah Jan**	ID	**6611**

Q.1 **Is there any relationship among Client (Customer, sponsor), Developer and End User? If yes then explain.**

Yes they all have strong relation with each other and sometimes known as stack holders. further explanation are given below.

A stakeholder is a person or organization that has rights, shares, claims, or interests concerning the system or its properties meeting their needs and expectations.

To put it more simply, the interests of stakeholders have some influence on the project, so their opinion should always be taken into account. If you do not do this and overlook one of the key stakeholders, you can ruin the whole project, and it will be much more expensive than just letting a development bug in the project. Stakeholders provide opportunities and limitations for the system and are the source of requirements.

The interesting point is that often, stakeholders are not defined before the decision-making stage. But as soon as the decision is designed, announced, or implemented, everyone affected by this decision will express their opinion. To save the project from potential harm, you are recommended to first answer the questions why and to whom, and only then how.

○ **Customers**. Customers are one of the key stakeholders. If you are an architect, then there is only one question. How could you forget to discuss your decision with the people who pay the money for the project development? I will answer myself. It's easy. In my practice, there was one example when a fantastic technical solution for real-time data processing and synchronization was created. This decision was one of the most advanced on the market, taking into account the latest technological trends. Furthermore, it was competently designed, correctly tested, and shown to the customer. And then it turned out that the customer wanted something different. More precisely, a completely different solution. And they did not need super synchronization at all.

○ **Developer**. Imagine that you have developed a solution that uses the .NET technology stack. But there is one problem. You have twenty available Java developers in the company and no one who knows .NET. I suppose after you remember this, there will be no need in explaining why the designed solution is terrible. And this is the simplest example. You need to know the team to understand what technologies they know well and which of them should not be used just because they are trending.

○ **End users**. So, we finally got to them. I hope they always had a key influence on the project, but in practice, this is not true.

● Those who are not involved in the project, but because of their position or activities can influence it.

Q.2 **Explain the design "Trades-Offs" between the following:**

a) Cost vs. Robustness
b) Cost vs. Reusability
c) Backward compatibility vs. Readability

a)Cost vs Robustness: Cost is the main driving factor for all projects. When it is done correctly, it helps in the successful completion of the project. In this research we have discussed various factors that affect the estimation procedure. These include team structure, team culture, managerial style, project type (Core application or integrated application), client's working environment. Accurate estimation is far difficult in developing countries where most of the organizations follow local standards. These inaccurate estimations lead to late delivery, less profit or in worst case complete failure. Software requirement gathering, development, maintenance, quality assurance and cost of poor quality are major groups responsible for overall cost in software production process. The exact proportion among them varies significantly in consecutive software releases, which is caused by many factors. The ever increasing need for the reliability of the software systems, especially mission critical applications in the public safety domain, raises the bar for the accuracy of prediction and estimation techniques.
Robustness is the ability of a computer system to cope with errors during execution and cope with erroneous input. Robustness can encompass many areas of computer science, such as robust programming, robust machine learning, and Robust Security

Network.
b)Cost vs Reusability:
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. Many reuse design principles were developed at the WISR workshops

Candidate design features for software reuse include:
•Adaptable
•Brief: small size
•Consistency
•Correctness
•Extensibility
•Fast
•Flexible
•Generic
•Localization of volatile (changeable) design assumptions
•Modularity
•Orthogonality
•Parameterization
•Simple: low complexity
•Stability under changing requirements
Cost is the main driving factor for all projects. When it is done correctly, it helps in the successful completion of the project. In this research we have discussed various factors that affect the estimation procedure. These include team structure, team culture, managerial style, project type (Core application or integrated application), client's working environment. Accurate estimation is far difficult in developing countries where most of the organizations follow local standards. These inaccurate estimations lead to late delivery, less profit or in worst case complete failure. Software requirement gathering, development, maintenance, quality assurance and cost of poor quality are major groups responsible for overall cost in software production process. The exact proportion among them varies significantly in consecutive software releases, which is caused by many factors. The ever increasing need for the

reliability of the software systems, especially mission critical applications in the public safety domain, raises the bar for the accuracy of prediction and estimation techniques.

C) Backward compatibility vs. Readability:
Backward Compatibility: A new version of a program is said to be backward compatible if it can use files and data created with an older version of the same program. A computer is said to be backward compatible if it can run the same software as the previous model of the computer. Backward compatibility is important because it eliminates the need to start over when you upgrade to a newer product. A backward-compatible word processor, for instance, allows you to edit documents created with a previous version of the program. In general, manufacturers try to keep all their products backward compatible. Sometimes, however, it is necessary to sacrifice backward compatibility to take advantage of a new technology.
Software readability is a property that influences how easily a given piece of code can be read and understood. Since readability can affect maintainability, quality, etc., programmers are very concerned about the readability of code

Q.3 What is the outcome of the software design? Explain in detail.

There’re many benefits/outcomes of designing and developing software exactly as per your requirements.

1. Optimized business process

Each organization has its own business model and in-house processes. It is very difficult for organizations to change their processes to suit a particular software package or application, however efficient and powerful it may be. Therefore, software needs to be designed and developed in a manner such that it can align with the business model and follow the organization’s unique in-house processes. Custom software development helps to optimize your business processes rather than replacing them.

2. Invention

Since the software is totally customized, you have the option to decide what kind of software development technology to use to design your own app. You have the power to decide and opt for trend-setting disruptive technologies to design you customized app and make it work the way you want it to.

3. Emphasize your business acumen

The very fact that you are keen to develop software as per your business needs sends out a strong message that you value your in-house processes and take your work seriously. It emphasizes your commitment towards your business as you streamline your process flows and your working model so it can function smoothly.

4. Reliability

The ability to upkeep and follow your business processes over time helps you to succeed eventually. Reliability is a major factor that defines success. Proper testing of your custom software ensures you have a reliable IT tool that can grow your business.

5. Uniqueness

Each business is unique. There's no one-size-fits-all solution as far as business processes are concerned. Having a software tailor-made to suit your unique requirements can complement your working model. Custom software development also helps to support your "unique" identity in the market.

6. Adaptability

Organizational processes change with time, and as the market dynamics change, it becomes necessary to adapt to new processes and technologies to maintain your marketing stronghold. Customized software can be easily changed – New processes and technologies can be integrated into your existing software as and when marketing trends change, so you can remain abreast of your competitors.

7. Compatibility

Most organizations have an architecture model in which the outputs generated by a particular process (software app or module) function as an input for another process. A smooth flow of information is vital while streamlining your business model. Using different “packaged” software for different processes can disrupt your data flow since a hybrid software infrastructure often has to depend upon third party gadgets and apps to facilitate the communication between varied processes and systems. Building a custom software environment can resolve many types of issues concerning the flow of information between successive processes.

8. Exclusiveness

What works best for one business doesn’t necessary work for another. You may be following certain processes which others don’t. Software that is developed exclusively for you ensures that all of your activities and processes are properly addressed to and automated exactly as per your requirements.

9. Flexibility

You don’t have to mold your working to suit a particular software – Your software can be changed easily to suit your requirements as and when required provided it is custom made.

10. Security

A major concern for many B2B and B2C companies, data access and security concerns affect many end-users in the market today. People transacting online want to ensure their transactions are safe and secure at all times. Supporting expensive security protocols can make you pass on added costs to the services you offer to your customers. This can make you lose your competitive edge in the market. Moreover, the flow of data within internal processes of the organization also needs to be regulated by implementing strict security standards. With customized software development, you have the power to decide which data-security technology or protocol is ideally suited for your business and integrate that in your software.

11. Cost effectiveness

With customized development, you can plan and phase the development process. You're not required to invest a huge sum of money first on to reap the benefits of automation. Based upon you budget and funds availability, you can start automating individual process flows in an organized and timed manner over time to make development affordable through affordable software development services.

The takeaway for organizations and businesses is even though you're required to spend some time to define your exact automation needs and wait while your software is developed, it's worthwhile to opt for customized software development since you can benefit from an automation process that is tailor made to suit your unique needs and business-centric requirements.

Q.4 What is ADL (Architectural Descriptive Language)? How many ADLs are there? Explain one of them.

Architecture description languages (ADLs) are formal languages that can be used to represent the architecture of a software-intensive system. As architecture becomes a dominating theme in large system development, methods for unambiguously specifying architecture will become indispensable**.**

The Architecture Analysis & Design Language (AADL) is an architecture description language standardized by SAE. AADL was first developed in the field of avionics, and was known formerly as the Avionics Architecture Description Language.

The Architecture Analysis & Design Language is derived from MetaH, an architecture description language made by the Advanced Technology Center of Honeywell. AADL is used to model the software and hardware architecture of an embedded, real-time system. Due to its emphasis on the embedded domain, AADL contains constructs for modeling both software and hardware components (with the hardware components named "execution platform" components within the standard). This architecture model can then be used either as a design documentation, for analyses (such as schedulability and flow control) or for code generation (of the software portion), like UML.

Types Of ADLs:

There are many types Architecture Description Language. Which are given below.

ACME

Rapide

Wright

Unicon

Aesop

MetaH

Lileanna

ACME

- ACME was developed jointly by Monroe, Garlan (CMU) and Wile (USC)
- ACME is a general purpose ADL originally designed to be a lowest common denominator interchange language
- ACME as a language is extremely simple (befitting its origin as an interchange language)
- ACME has no native behavioral specification facility so only syntactic linguistic analysis is possible

  - there are currently efforts under consideration to define a behavioral semantics for ACME, possibly along the Wright/CSP line

- ACME has no native generation capability
- ACME has seen some native tool development, and there are indications of more, as well as use of other language tools via interchange

ACME Overview

• Provides constructs for describing *systems as graphs of* components interacting via connectors, *a* representation mechanism *for* hierarchical decomposition of components and connectors into subsystems.

- Does not provide a specific model for describing system behaviour, instead elements may be annotated with properties that represent this Information.
- Rather than providing a fixed set of models for formulating solutions, Acme provides general, domain-neutral foundation for developing new tools and notations.
- Structures provided by Acme are based on the informal box and line diagrams traditionally used to depict the architecture of the system. Acme helps a designer document design decisions and reason about the implications of those decisions.

Q.5 What is the Architectural Style? Explain components of a style?

An architecture style (also known as an “architecture pattern”) abstracts the common properties of a family of similar designs.

In software engineering, an Architectural Pattern is a general and reusable solution to an occurring problem in a particular context. It is a recurring solution to a recurring problem.

The purpose of Architectural Patterns is to understand how the major parts of the system fit together and how messages and data flow through the system.

An architectural style defines a family of systems in terms of a pattern of structural organization; a vocabulary of components and connectors, with constraints on how they can be combined.

Architectural styles are reusable 'packages' of design decisions and constraints that are applied to an architecture *to induce chosen desirable qualities.*
A software designer or architect may identify a design problem which has been visited and perhaps even solved by others in the past. A template or pattern describing a solution to a common problem is known as a design pattern. The reuse of such patterns can help speed up the software development process

Each style will describe a system category that consists of :

- A set of components (eg: a database, computational modules) that will perform a function required by the system.
- The set of connectors will help in coordination, communication, and cooperation between the components.
- Conditions that how components can be integrated to form the system.
- Semantic models that help the designer to understand the overall properties of the system.

The use of architectural styles is to establish a structure for all the components of the system.

The key components of an architecture style are:

- Elements/components
  - that perform functions required by a system
- connectors
  - that enable communication, coordination, and cooperation among elements
- constraints
  - that define how elements can be integrated to form the system
- attributes
  - that describe the advantages and disadvantages of the chosen structure.

****