Name : Waheed Akbar **Reg no : 15426 Ms in Software Engineering**

Iqra National University Peshawar Pakistan

Department of Computer Science

Spring Semester, Final Term Exam, June 2020

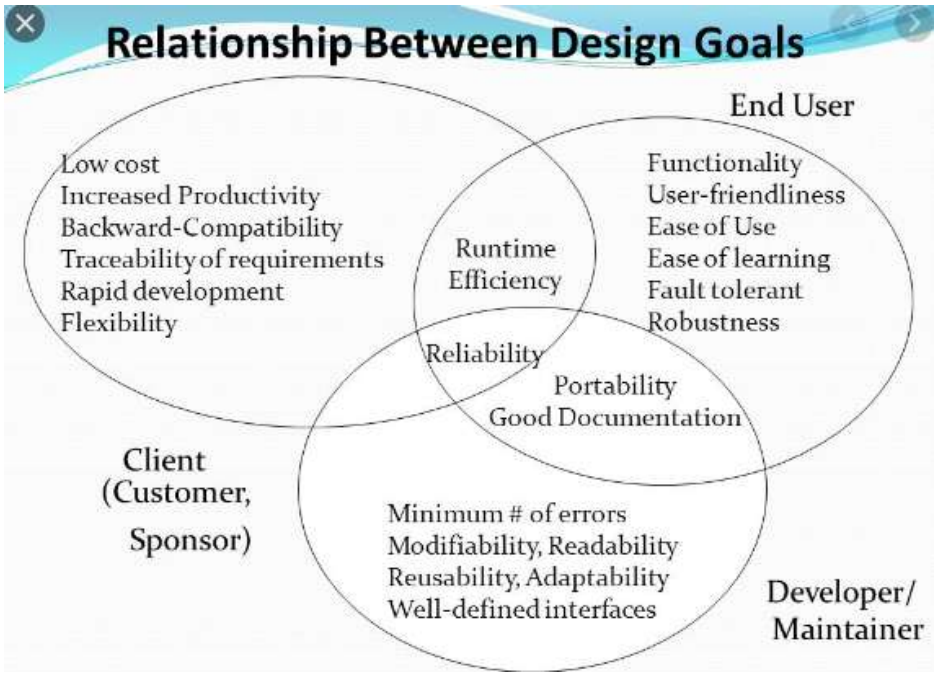
Paper :	**Software Design**	Date and Starting Time:	**24/June/2020, 9:00 am**
Program:	**MS (CS & SE)**	Uploading Date and End Time:	**24/June/2020, 3:00 pm**
Teacher Name:	**Dr. Fazal-e-Malik**	Marks	**50**

Note: Attempt all Questions. Help can be taken from net where ever is required.

Q.1 Is there any relationship among Client (Customer, sponsor), Developer and End User? If yes then explain. (10 Marks)

Ans 1 : In relationship between design goals we must have runtime efficiency and reliability .Portability and good documentation should be done so that the client and end users goals are satisfied. keeping open two separate channels of communication, one for the end-user developers and one for the entire ... sponsors, clients, project management team, stakeholders, and content developers to establish a relationship among the team.

Through diagram understanding can be easy..



A software designer is responsible for problem-solving and planning for a software solution.
Sponsor. An individual or a group that provides resources and support for the project, program, or portfolio, and is accountable for enabling success. Customer. Customer is the person(s) or organization(s) that will pay for the project's product, service, or result.

The Role of the Sponsor in Project Success
The project sponsor is an individual (often a manager or executive) with overall accountability for the project. ... Acts as the link between the project, the business community, and strategic level decision-making groups.

Run-time efficiency is a topic of great interest in computer science: A program can take seconds, hours, or even years to finish executing, depending on which algorithm it implements.

Reliability is the degree of consistency of a measure. A test will be reliable when it gives the same repeated result under the same conditions

Portability is a characteristic attributed to a computer program if it can be used in an operating systems other than the one in which it was created without requiring major rework. Porting is the task of doing any work necessary to make the computer program run in the new environment.

A good documentation is Clearly written documentation: All documents must be accurate and written in a manner that prevents errors and ensures consistency. If documents are to be used together, e.g. a SOP and a form, then each should reference the other.

The first role, that of developer, is always important and normally recedes into the ... between the supervision relationship and the supervisee's client relationships ... Being client-focused in supervision means that the supervisee's clients are ... a central place in his thoughts for the needs of those 'end users' of supervision.

……

Name : Waheed Akbar **Reg no : 15426 Ms in Software Engineering**

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

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

10 Marks)

Ans 2 : Trade-off. A trade-off (or tradeoff) is a situational decision that involves diminishing or losing one quality, quantity or property of a set or **design** in return for gains in other aspects. In simple terms, a tradeoff is where one thing increases and another must decrease

Trade offs in product design
Every time we make a **trade-off** we are consciously compromising on something. Every **product** decision you make has a **trade-off**: you achieve something at the cost of something else and you have to choose your priorities carefully

A) **Cost vs. Robustness** : Cost vs. Robustness Trade-offs create opportunity costs, one of the most important concepts in economics. Whenever you make a trade-off, the thing that you do not choose is your opportunity cost. To butcher the poet Robert Frost, opportunity cost is the path not taken (and that makes all the difference). You bought that bike.
Designing a production process normally is involved with some important constraints such as uncertainty, trade-off between production costs and quality, customer's expectations and production tolerances. In this paper, a novel multi-objective robust optimization model is introduced to investigate the best levels of design variables. Design suboptimality and robustness-fragility trade-offs ... In this case, the distribution of projected positions of a certain cell or organism for its . along the efficient frontier to the right for higher yield at the cost of robustness.

B) **Cost vs. Reusability** : Cost vs. reusability Trade-offs create opportunity costs, one of the most important concepts in economics. Whenever you make a trade-off, the thing that you do not choose is your opportunity cost. To butcher the poet Robert Frost, opportunity cost is the path not taken (and that makes all the difference). You bought that bike.
Designing a production process normally is involved with some important constraints such as uncertainty, trade-

off between production costs and quality, Is it a good rule of thumb to always write code for the intent of re-using it somewhere down the road? Or, depending on the size of the component you are writing, is it better practice to design it for re-use when it makes sense with regards to time spent on it. What is a good rule of thumb for spending extra time on analysis and design on project components that have "some probability" of being needed later down the road for other things that may or may need this part.

Backward compatibility vs. Readability : Backward compatibility vs. Readability Backward compatibility. Newer code can read data that was written by older code. Forward compatibility. Older code can read data that was written by newer code. Backward compatible refers to a hardware or software system that can use the interface of an older version of the same product. A new standard product or model is considered backward compatible when it is able to read, write or view older formats.
backward compatibility important?
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 documentscreated with a previous version of the program.

Readability in software design can be defined as the ease with which the software is read and understood. ... Programmers that are “journeyman” and move from one project to another throughout their career tend to have an easier time reading a variety of software code.

……

Name : Waheed Akbar **Reg no : 15426 Ms in Software Engineering**

Plz next page

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

A3: **Ans 3** : An ability to develop and conduct appropriate experimentation, analyze and interpret data, and use **engineering** judgment to draw conclusions. an ability to acquire and apply new knowledge as needed, using appropriate learning strategies.

Software design is the process by which an agent creates a specification of a **software** artifact, intended to accomplish goals, using a set of primitive components and subject to constraints. **Software design** usually involves problem solving and planning a **software** solution.

Software design is the process by which an agent creates a specification of a software artifact, intended to accomplish goals, using a set of primitive components and subject to constraints. Software design may refer to either "all the activity involved in conceptualizing, framing, implementing, commissioning, and ultimately modifying complex systems" or "the activity following requirements specification and before programming, as [in] a stylized software engineering process."

10 Marks)

Software design usually involves problem solving and planning a software solution. This includes both a low-level component and algorithm design and a high-level, architecture design.

There are three main patterns: Architectural - High-level pattern **type** that can be defined as the overall formation and organization of the **software** system itself. **Design** - Medium-level pattern **type** that is used by the developers to solve problems in the **design** stage of **development**.

Software design process can be perceived as series of well-defined steps. Though it varies according to **design approach** (function oriented or object oriented, yet It may have the following steps involved: A solution **design** is created from requirement or previous used system and/or system sequence diagram.

Software design is the most important phase of the software development cycle.As such, good design relies on a combination of high-level systems thinking and low-level component knowledge.

In modern software design, the best practice revolves around creating modular components that you can call and deploy as needed.

To make a software design **the following is a list of sections that you should at least consider including in your next design doc:**

- Title and People.
- Overview.
- Context.
- Goals and Non-Goals.
- Milestones.
- Existing Solution.
- Proposed Solution.
- Alternative Solutions.

……

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

Ans 4 : Architecture Description Language (ADL) is defined as "a language (graphical, textual, or both) for describing a software system in terms of its architectural elements and the relationship among them". ... UML includes a set of graphical notation techniques to create abstract models of specific systems.

Architecture description languages are used in several disciplines: system engineering, software engineering, and enterprise modelling and engineering. The system engineering community uses an architecture description language as a language and/or a conceptual model to describe and represent system architectures. 10 Marks)

Characteristics

There is a large variety in ADLs developed by either academic or industrial groups. Many languages were not intended to be an ADL, but they turn out to be suitable for representing and analyzing an architecture. In principle ADLs differ from requirements languages, because ADLs are rooted in the solution space, whereas requirements describe problem spaces. They differ from programming languages, because ADLs do not bind architectural abstractions to specific point solutions. Modeling languages represent behaviors, where ADLs focus on representation of components. However, there are domain specific

modeling languages (DSMLs) that focus on representation of components.

Minimal requirements

The language must:

- Be suitable for communicating an architecture to all interested parties
- Support the tasks of architecture creation, refinement and validation
- Provide a basis for further implementation, so it must be able to add information to the ADL specification to enable the final system specification to be derived from the ADL
- Provide the ability to represent most of the common architectural styles
- Support analytical capabilities or provide quick generating prototype implementations

Positive and negative elements of ADL :

Positive elements of ADL

- ADLs are a formal way of representing architecture
- ADLs are intended to be both human and machine readable
- ADLs support describing a system at a higher level than previously possible
- ADLs permit analysis and assessment of architectures, for completeness, consistency, ambiguity, and performance
- ADLs can support automatic generation of software systems

Negative elements of ADL

- There is no universal agreement on what ADLs should represent, particularly as regards the behavior of the architecture
- Representations currently in use are relatively difficult to parse and are not supported by commercial tools
- Most ADLs tend to be very vertically optimized toward a particular kind of analysis

ADLs have in common:

- **Graphical syntax with often a textual form and a formally defined syntax and semantics**
- **Features for modeling distributed systems**

- **Little support for capturing design information, except through general purpose annotation mechanisms**
- **Ability to represent hierarchical levels of detail including the creation of substructures by instantiating templates**

ADLs differ in their ability to:

- **Handle real-time constructs, such as deadlines and task priorities, at the architectural level**
- **Support the specification of different architectural styles. Few handle object oriented class inheritance or dynamic architectures**
- **Support the analysis of the architecture**
- **Handle different instantiations of the same architecture, in relation to product line architectures**

Type of ADL

ADLs have been classified into three broad categories: box-and-line informal drawings, formal **architecture description language**, and UML (Unified Modeling **Language**)-based notations. Box-and-line have been for a long time the most predominant means for describing SAs

UML The Unified Modeling Language is a general-purpose, developmental, modeling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system.

UML, short for Unified Modeling Language, is a standardized modeling language consisting of an integrated set of diagrams, developed to help system and software developers for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling

Mainly, UML has been used as a general-purpose modeling language in the field of software engineering. However, it has now found its way into the documentation of several business processes or workflows. For example, activity diagrams, a type of UML diagram, can be used as a replacement for flowcharts.

Why is UML (Unified Modelling Language) called unified? Because, at its origin in the mid 1990's, it was a merge/unification of three approaches: Grady Booch's eponimous Booch Method

List of UML Diagram Types
Structure Diagrams. Class Diagram. Component Diagram. Deployment Diagram. Object Diagram. Package Diagram. Profile Diagram. Composite Structure Diagram.

Behavioral Diagrams. Use Case Diagram. Activity Diagram. State Machine Diagram. Sequence Diagram. Communication Diagram. Interaction Overview Diagram.

UML stands for Unified Modeling Language. It's a rich language to model software solutions, application structures, system behavior and business processes. There are 14 UML diagram types to help you model these behaviors.

UML is a combination of several object-oriented notations: Object-Oriented Design, Object Modeling Technique, and Object-Oriented Software Engineering. UML uses the strengths of these three approaches to present a more consistent methodology that's easier to use.

The UML notation is a notation conceived for modeling object of applications and continue and extend, in particular,
the notations of OMT (Object Modeling Technique) and Booch methods. More precisely, here we describe the principles of the use-case diagrams, classes, objects and sequence diagrams.

……

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

A **ANS 5** : An architectural pattern is a general, reusable solution to a commonly occurring problem in software architecture within a given context. Architectural patterns are often documented as software design patterns

An architectural style is characterized by the features that make a building or other structure notable or historically identifiable. ... A style may include such elements as form, method of construction, building materials, and regional character.

10 Marks)

here are many recognized architectural patterns and styles, among them:

- Blackboard.
- Client-server (2-tier, 3-tier, n-tier, cloud computing exhibit this style)
- Component-based.
- Data-centric.
- Event-driven (or implicit invocation)
- Layered (or multilayered architecture)
- Microservices architecture.

- Monolithic application.

This style is divided into various horizontal layers and each layer has some specific function. Generally, this architectural style has four layers namely presentation, business, persistence, and database, where each layer has a different function.

Three types of architectural styles have been described in this lesson including layered, object-oriented, and data-centric.

The architectural styles that are used while designing the software as follows:

Data-centered architecture. The data store in the file or database is occupying at the center of the architecture. ...

Data-flow architecture. ...

Call and return architectures. ...

Object-oriented architectures. ...

Layered architectures.

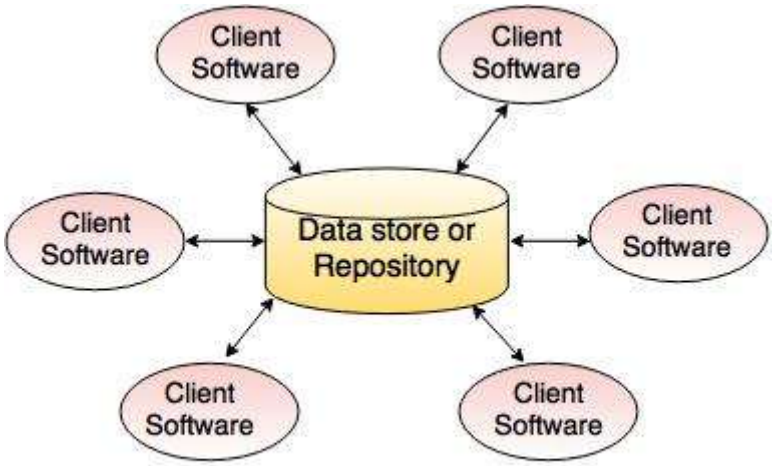


Fig.- Data centered architecture

Model–view–controller

(usually known as MVC) is a software design pattern commonly used for developing user interfaces that divides the related program logic into three interconnected elements. This is done to separate internal representations of information from the ways information is presented to and accepted from the user. This kind of pattern is used for designing the layout of the page.

Traditionally used for desktop graphical user interfaces (GUIs), this pattern has become popular for designing web

applications. Popular programming languages like JavaScript, Python, Ruby, PHP, Java, C#, and Swift have MVC frameworks that are used for web or mobile application development straight out of the box.

Components

Model

The central component of the pattern. It is the application's dynamic data structure, independent of the user interface. It directly manages the data, logic and rules of the application.

View

Any representation of information such as a chart, diagram or table. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants.

Controller

Accepts input and converts it to commands for the model or view.

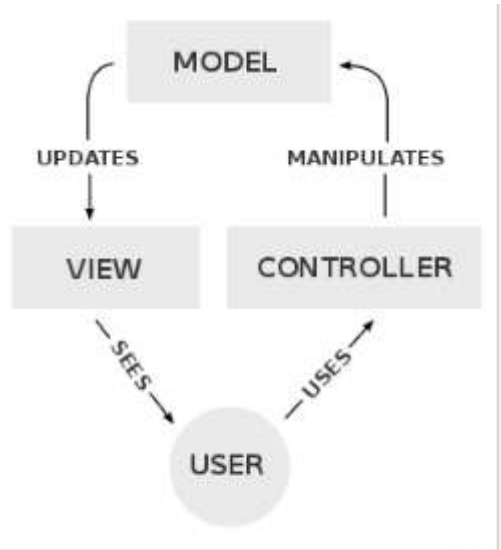


Diagram of interactions within the MVC pattern

……

☺ **********☺

Name : Waheed Akbar **Reg no : 15426 Ms in Software Engineering**

Note: Please write your Name and ID on top of your answer Paper otherwise you will get zero marks.