**Software Design**

**Mid Term Assignment**

****

**Submitted by**

**Rizwan Mukhtiar**

MS (Computer Science)

Registration ID: 5952

**Assigned by**

**Dr. Fazal-E-Malik**

**Department of Computer Science**

**Faculty of Engineering and Applied Sciences**

**Iqra National University, Peshawar**

**Q.1 a) Is the Software Design needed in the Software Development Projects? If yes**

**then explain how it is important?**

**b) As one of the most complex man-made artifacts (objects), computer software**

**is very difficult to design Explain what are the main challenges in software design?**

**Q.1** **a)** Yes the software design is very essential part in the software development project. Software design is the process by which a creator creates a specification of a [software](https://en.wikipedia.org/wiki/Artifact_%28software_development%29) objects, intended to accomplish [goals](https://en.wikipedia.org/wiki/Goal), using a set of primitive components and subject to [constraints](https://en.wikipedia.org/wiki/Constraint_%28mathematics%29). 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](https://en.wikipedia.org/wiki/Software_requirements) specification and before [programming](https://en.wikipedia.org/wiki/Computer_programming), as a stylized software engineering process. Software design is basically a mechanism of preparing a plan, a layout for structuring the code of your software application.

**Importance:**

Software design usually involves problem solving and planning a [software](https://en.wikipedia.org/wiki/Software) solution. This includes both a low-level component and [algorithm design](https://en.wikipedia.org/wiki/Algorithm_design) and a high-level, [architecture](https://en.wikipedia.org/wiki/Software_architecture) design. Whenever a software developer starts working on any type of new project, it is obvious that he would want to start scripting the code right away. Regardless of what his expertise is over the software are, if he builds software without laying out a plan for it, then he might be building a home without setting a strong foundation. Well, no matter how strong the building process and use of resources are, the house is going to collapse, similarly, without a foundational layout, the software is also going to crash. Nobody really wills to let the effort go in vain, just out of a silly mistake of not setting a plan for it, right? Well, that is where software design comes into play, a mechanism that allows critical software processes to be simplified in a certain manner.

Design phase is one of the most important phases in SDLC (Software Development Life Cycle). This bridges the gap between Requirements and code Implementation.

Figure 1. Software Development Life Cycle

**Requirements**

**Architectural Detailed Design**

Here the architectural and detailed design of the solution breaks the solution into different components which are allocated to one or more processing resources. Methods and fields are defined for classes. Also the algorithm for the methods are defined.

For evaluating user requirements, an SRS (Software Requirement Specification) document is created whereas for coding and implementation, there is a need of more specific and detailed requirements in software terms. Software design should be the first step in SDLC (Software Design Life Cycle), which moves the concentration from problem domain to solution domain

Following are some points that make software design so important:

**Modularity:** It means splitting the software project into small pieces, small tasks called modules for the simplicity of work. Splitting your critical software project into modules only makes it easier to work on every requirement of the project. It also gives the convenience of making changes in the future. If in case, the requirement of your client changes, then you can anytime go for restructuring a module, not the whole project

**Maintainability:** As software design is performed by creating modules, it makes the task easier to maintain. Tasks like finding bugs, debugging, restructuring, and changing the functionality of specific elements in the software application become quite easy due to software design. A good software design gives you the privilege of changing the appearance, functionality, etc, of the software by working on a specific module

**The Flow of functionality and performance:** Software design is a reflection of the performance of the software application. A good software design effectively displays the flow of functions taking place while the software is running. From input to the output, a software design shall show all the steps so that the performance of the software application can be analyzed. If in case the software project is handed over from a software developer to another, then the new developer should understand the software by just reading the software design and this is what the privilege is given by good software design

**Portability and Trackability:** When it comes to making changes in the software, the elements like portability come handy. Portability in the software design gives the convenience of transferring functions from a module to another, as it can make a whole lot of changes in the functionality of software applications. Another, important element of software design is Trackability. Trackability is as its name suggests, it is the aptitude for tracking the flow of functions taking place in the software working. Good software design provides track ability to the software application which makes it easier to use and maintain

**Q.1** **b)** As one of the most complex man-made artifacts (objects), computer software is very difficult to design. There are many factors that affect designs and many stakeholders i.e. people who participate in the design process, play various different roles in the design processes and influence the design of software.

It depends on what kind of software you want to develop. It also depends on personal skill, your attitude, and, what is impossible for a beginner, but experience matters a lot. You will gain that when you keep trying. Quickly coding something is easy. But trying to design and implement some master piece of software is generally not that easy. You need to design it so it's testable, extendable but yet simple, easy to read and not over engineered. Also it's not enough to write testable code: you also need to write the tests. A software Design is about HOW the system will perform its functions? It is just provides the overall decomposition of the system. It allows to split the work among a team of developers/designers. This also lays down the groundwork for achieving non-functional requirements: performance, maintainability, reusability, etc.

Today, the software design phase is an essential phase of the development life-cycle. The increasing complexity of today’s systems has created a set of particular challenges that makes it hard for software engineers to meet the continuous customer demand for higher software quality. These challenges have prompted software engineers to pay closer attention to the design process to better understand, apply, and spread well known design principles, processes, and professional practices to overcome these challenges. A brief discussion of these challenges is presented below

**Requirements Volatility**

A major reason for the complexity of software projects is the constant change of requirements. When designed properly, software can be modified or extended easily; however, when designed poorly, modifying software can become crushing and lead to all sorts of complex problems. Unlike the development of computer hardware, bridges, houses, or mechanical parts, software's very own nature allows itself to change to provide different or new functionality to systems. This same trait that makes software so desirable is what makes it also so complex. Although much effort is put in the requirements phase to ensure that requirements are complete and consistent, rarely that is the case; leaving the software design phase as the most influential one when it comes to minimizing the effects of new or changing requirements.

**Design Process**

Software Engineering is a process oriented field. Software processes allows engineers to organize the steps required to develop software solutions with schedule and cost constraints. Therefore, at the core of every software development company, there should be a sound, well understood, and reliable process for software development. Processes can also be developed and customized for particular phases of the software engineering life-cycle. In the design phase, software processes involve a broad set of activities and tasks that bridge the gap between requirements and construction while observing to a set of project-specific or company-specific constraints. These activities include common ones, such as architectural and detailed design, as well as other supporting management activities. The design process is challenging because essential design process activities are often ignored, simply not done at all.

**The Technology**

Software is meant to be everywhere. From healthcare systems, education, defense, and everyday global devices, software is required to operate on a huge and always evolving technology landscape. Besides the operating environment, the technology for designing and implementing today's software systems continues to evolve to provide improved capabilities. Examples of these include modeling languages and tools, programming languages, development environments, design patterns, and design strategies. As new technologies emerge, software engineers are required to adjust and employ them all at the same time. Some software systems are required to interoperate with old legacy systems designed with older design methodologies. This results in software designers employing a numberless of design methodologies and technologies, all on the same software system.

**The Right and Professional Practices**

Designers create blueprints that drive the construction of the software. During this creation process, designers are required to determine how design decisions affect the environment and the people that use the software. In many cases, the software development process is traditionally carried out under tight schedule constraints. Naturally, all phases of the development life-cycle suffer from this, including the design phase. Designers are also responsible for enforcing right guidelines during the design process; evaluating the social impacts of their designs in the public domain, or in safety-critical systems; and to follow the appropriate professional practices to ensure success in the overall system. The ethical and professional practices aspect of software design are challenging because designers are constantly faced with frequent burdens from investors that impact designers decisions, most of which have consequences of social, ethical, or professional nature.

**Managing Design Effects**

Designs are shaped by many different influences from stakeholders, the development organization, and other factors. These effects can have cyclical effects between the system and its external influences, such that external factors affect the development of the system and the system affects its external factors. These influences is essential for maximizing the quality of systems and their related influence on future business opportunities. Of specific importance are design influences that come from the system stakeholders and its developing organization.

Nowadays some other design challenges also includes: Technical Challenge, Planning a project, Debugging, Integration, Security Challenges, and Operational Challenges

**Q.2 The literature on design methods began to appear in the 1950s and 60s. Since then,**

 **design methodology has become an independent discipline of scientific study**

 **What are the essential characteristics of design?**

 **What activities of Software Design Process are used by designers?**

The literature on design methods began to appear in the 1950s and 60s. Since then, design methodology has become an independent discipline of scientific study

The design must implement all of the clear requirements contained in the analysis model, and it must accommodate all of the understood requirements desired by the customer. The design must be a readable, understandable guide for Developers who generate code and for Tester who test and subsequently support the software. The design should provide a complete picture of the software, addressing the data, functional, and behavioral domains from an implementation perspective.

**Essential Characteristics of Software Design:**

The essential characteristics of software design are as follows

**Functionality**
It refers to the degree of performance of the software against its intended purpose.

Required functions are: suitability, accuracy, interoperability, compliance, and Security

**Reliability**
A set of attribute that bear on capability of software to maintain its level of performance under the given condition for a stated period of time.

Required functions are: recoverability, fault tolerance, and maturity

**Efficiency**
It refers to the ability of the software to use system resources in the most effective and efficient manner. The software should make effective use of storage space and executive command as per desired timing requirement.

Required functions are: In time, and In resource

**Usability**

It refers to the extent to which the software can be used with ease. The amount of effort or time required to learn how to use the software.

Required functions are: understandability, learnability, and operability

**Maintainability**
It refers to the ease with which the modifications can be made in a software system to extend its functionality, improve its performance, or correct errors.

Required functions are: testability, stability, changeability, and operability

**Portability**
A set of attribute that bear on the ability of software to be transferred from one environment to another, without or minimum changes.

Required functions are: adaptability, installability, and replaceability

Some other characteristics are also considerable: that it helps in communication, it facilitates the system understanding, it reduces the software maintenance costs due to less amount of problems arises, it increases the product quality, it facilitates the implementation, solution of any problem will be easy, it might have the functionality to upgrade the product in any level.

**Activities of Software Design Process:**

The implementation phase of software development is the process of converting a system specification into an executable system through the design of system. A software design is a description of the architecture of the software to be implemented, the data which is part of the system, the interfaces between system components and, sometimes, the algorithms used. The design process activities are the followings:

**Architectural Design:** The sub-systems of system and their relationships are identified based on the main functional requirements of software.

**Abstract specification:** For each sub-system, an abstract specification of its services and the constraints under which it must operate is defined. It provides the basis for further development. The specification technique may be in a natural language or in a formal mathematical language or may be using a diagrammatic technique for specification of methodology.

**Interface design:** Interfaces allow the sub-system’ services to be used by other sub-systems. The representation of interface should be hidden. In this activity the interface is designed and documented for each sub-system. The specification of interface must be unambiguous.

**Component design:** Services are allocated to components and the interfaces of these components are designed.

**Data structure design:** The data structures used in the system implementation are designed in detail and specified.

**Algorithm design:** In this activity the algorithms used to provide services are designed in detail and specified.

**Q.3** **Study and explain Models other than McCall’ Model of attributes. Does the software Design affect these Models?**

The overall goal of any software management is Quality built-in with cost and performance as prime consideration*.* This means that the software should be built with certain quality aspects that fulfill the needs of the user. Its performance is kept on the top priority. The performance is also based on the demands of the user and the developer's perspective. The good performing software is highly in demand these days due to rising competition in the market. Successful software is developed that fulfills the user needs in consideration with the developer's point of view. McCall model is one of them, other than that different software quality models are developed for a remarkable quality attributes of the software.

**Software Quality Models**

We have a number of software quality models other than McCall’s model are developed in which some of the majorly used models are as:

1. **Boehm Model:**

Boehm software quality model was introduced in the year of 1978. The model is used to represent a hierarchical model that structures around high level characteristics, intermediate level characteristics, and primitive characteristics. All of these together results in to establishment of a high quality software model. The aim of this model is to address the contemporary shortcomings of models that automatically and quantitatively evaluate the quality of software. Therefore, Boehm model represents the characteristics of the software product hierarchically in order to get contribute in the total quality. The model defines the quality of software on the basis of a set of credentials and measurements. It is also elucidates a model of software quality characteristics. The high level of characteristics is made in such a way that answers following questions:

**As-Is Utility:** It defines the way a utility signifies the as-is utility. It creates a question of how easily, reliably and efficiently an as can be utilized.

**Maintainability:** This aspect decides how convenient it is to understand, change or re-evaluate a process.

**Portability:** This aspect helps in deciding an effective way to change an environment.

Figure 2. Boehm’s Quality Model and all of its components



1. **FURPS Quality Model:**

FURPS model was presented after the McCall’s and Boehm’s model. It was firstly presented by Robert Grady and Hewlett Packard Co. the abbreviation of word FURPS stands for Functionality, Usability, Reliability, Performance and Supportability of the product under development

Functionality: Functionality contains combination of characteristics, security, features and capabilities.

**Usability:** This section covers the factors which affect usability of the product like Human Factors, Aesthetic, material of training and documentation of the user.

**Reliability:** It ensures the reliability and integrity factors like Recovery to failures Time among failures, Frequency and severity of failures.

**Performance:** Enforces conditions on practical necessities like speed, availability, efficiency, throughput, accuracy, resource usage, response time and recovery time.

**Supportability:** This section contains the components like Extensibility, Adaptability, Maintainability, Compatibility, Configurability, Install ability, Serviceability and Localizability

Figure 3. FURPS Quality Model and all of its components



1. **IEEE Quality Model:**

Institute of Electrical and Electronics Engineers (IEEE). It is an international organization which also provided many standard models for the software product quality and maintenance. It presented, many standards of software quality assurance and verification/validation of the software product. This model illustrates several ways for the measurement of qualitative factors and reflects factors like Efficiency, Functionality, Maintainability, Portability, Reliability and Usability.

**Efficiency:** It includes the characters like time and resources.

**Reliability:** It contains the factors and the characteristics which are used to increase and maintain the maturity and reliability by fault tolerance and Recoverability.

**Functionality:** It includes characteristics like accuracy, compatibility, completeness, security and interoperability.

**Supportability:** It helps in maintaining the testability, extendibility and correctability.

**Portability:** It makes the software portable by ensuring characteristics like hardware independency, software independency, adaptability, install-ability and reusability.

**Usability:** It ensures the comprehensibility, communicativeness and ease of learning.

Figure 4. IEEE Quality Model and all of its components



1. **ISO 9126 Quality Model:**

ISO 9126 is presented by the international standard organization. It is one of the most implemented and used quality model for maintaining the quality of the software product. This modern model is based at the previous models like McCall, Boehm, FURPS and many other old versions of the ISO like ISO 9000 and etc. This version of ISO also takes account of functionality as parameter and include identification of both internal quality characteristics and external quality characteristics of the products. The characteristics new version of this model contains are Efficiency, Maintainability, Functionality, Reliability, Portability and Usability.

**Efficiency:** It contains the characters which affects the throughput in a given time and resources used.

**Reliability:** It includes the Maturity, Fault Tolerance and Recoverability.

**Functionality:** It contains many essential characteristics which are necessary for the proper functionality of the product like suitability, accurateness, interoperability, compliance and security.

**Maintainability:** It helps in maintaining the changeability, stability and testability by using analyzability.

**Portability:** It makes the software more dynamic and portable by ensuring characteristics like adaptability, install-ability, conformance and replaceability.

**Usability:** It makes sour that the learnability and operability of the software product should be easy and simple.

Figure 5. ISO 9126 Quality Model and all of its components



**5. Capability Maturity Model (CMM):**

The Capability Maturity Model (CMM) is a methodology used to develop and refine an organization's software development process. The model describes a five-level evolutionary path of increasingly organized and systematically more mature processes. CMM was developed and is promoted by the [Software Engineering Institute (SEI)](https://searchsoftwarequality.techtarget.com/definition/Software-Engineering-Institute-SEI), a research and development center sponsored by the U.S. Department of Defense (DoD). The CMM is similar to ISO 9001, one of the [ISO 9000](https://searchdatacenter.techtarget.com/definition/ISO-9000) series of standards specified by the International Organization for Standardization ([ISO](https://searchdatacenter.techtarget.com/definition/ISO)). The ISO 9000 standards specify an effective quality system for manufacturing and service industries; ISO 9001 deals specifically with [software development](https://whatis.techtarget.com/reference/Learn-IT-Software-development) and maintenance. CMM specifies an increasing series of levels of a software development organization. The higher the level, the better the software development process, hence reaching each level is an expensive and time-consuming process.

**Levels of CMM**

**Level One, Initial:** The software process is characterized as inconsistent, and occasionally even chaotic. Defined processes and standard practices that exist are abandoned during a crisis. Success of the organization majorly depends on an individual effort, talent, and heroics. The heroes eventually move on to other organizations taking their wealth of knowledge or lessons learnt with them.

**Level Two, Repeatable:** This level of Software Development Organization has a basic and consistent project management processes to track cost, schedule, and functionality. The process is in place to repeat the earlier successes on projects with similar applications. Program management is a key characteristic of a level two organization.

**Level Three, Defined:** The software process for both management and engineering activities are documented, standardized, and integrated into a standard software process for the entire organization and all projects across the organization use an approved, tailored version of the organization's standard software process for developing, testing and maintaining the application.

**Level Four, Managed:** Management can effectively control the software development effort using precise measurements. At this level, organization set a quantitative quality goal for both software process and software maintenance. At this maturity level, the performance of processes is controlled using statistical and other quantitative techniques, and is quantitatively predictable.

**Level Five, Optimizing:** The Key characteristic of this level is focusing on continually improving process performance through both incremental and innovative technological improvements. At this level, changes to the process are to improve the process performance and at the same time maintaining statistical probability to achieve the established quantitative process-improvement objectives.

Figure 6. Capability Maturity Model (CMM)



**Software Design effects on Quality Models:**

Yes the software design effects on all the software quality models. When any level of stage software performs irregular responses. So there quality must be maintained during the whole process of software development. There will be two major reasons of effecting the design quality i.e. design quality requirements e.g. costly errors impossible to be corrected and design decisions that is effected the final product

Further design tasks divided into interrelated subtasks:

* 1. Architectural design
	2. Interface design
	3. Detail design including algorithm and data structure design

The following discusses how design (architecture design, interface design and detailed design) affect the key quality attributes of software.

1. Efficiency: number of processes processed in some interval of time.
2. Correctness: specification of user requirements.
3. Reliability: user requirements at a specified environment in given time slot.
4. Portability: transportation from one software platform to another.
5. Maintainability: repair action to machine or software system to its normal state.
6. Reusability: software components must be reusable to another development of software.
7. Interoperability: the ability of computer systems or software to exchange and make use of information, depends on the interface between a software system and its environment.