# Assignment # 4

## Introduction to Software Engineering

**BS (SE)**

**Batch 2015**

**Question # 1:**

**What is software metrics? List down the following metrics of software quality**

1. **Size Oriented Metrics**
2. **Function Oriented Metrics**
3. **Software Quality Metrics**

**ANS : Software metrics:**

Software metrics refers to a broad range of measurements for computer software. Measurement can be applied to the software process with the intent of improving it on a continuous basis. Measurement can be used throughout software project to assist in estimation, quality control, productivity assessment, and project control. Measurement can also be used by software engineers to help assess the quality of technical work products and to assist in the tactical decision making as a project proceeds.

The IEEE Standard Glossary of Software Engineering Terms defines metric as “a quantitative measure of the degree to which a system, component, or process possesses a given attribute.”

***Metrics of software quality:***

***Size Oriented Metrics:***Size-oriented software metrics are derived by normalizing quality and/or productivity measures by considering the size of the software that has been produced. If a software organization maintains simple records, a table of size-oriented measures can be created. The table lists each software development project that has been completed over the past few years and corresponding measures for that project. In the table entry for the project alpha: 12100 lines of code (LOC) were developed with 24 person-months of effort at a cost of $168,000. it should be noted that the effort and cost recorded in the table represents all software engineering activities. Further information for project alpha indicates that 365 pages of documentation were developed, 134 errors were recorded before the software was released, and 29 defects were encountered after release to the customer within the first year of operation. Three people worked on the development of the project alpha. 33 Project LOC Effort $(000) pp.doc Errors Defects People Alpha 12,100 24 168 From the rudimentary data contained in the table, a set of simple size-oriented metrics can be developed for each project: • Errors per KLOC (thousand lines of code). • Defects4 per KLOC. • $ per LOC. • Page of documentation per KLOC. In addition, other interesting metrics can be computed: • Errors per person-month. • LOC per person-month. • $ per page of documentation. Size-oriented metrics are not universally accepted as the best way to measure the process of software development. Most of the controversy swirls around the use of lines of code as a key measure.

***Function Oriented Metrics:***

Function-Oriented Metrics Function-oriented software metrics use a measure of the functionality delivered by the application as a normalization value. Since ‘functionality’ cannot be measured directly, it must be derived indirectly using other direct measures. Function-oriented metrics were first proposed by Albrecht, who suggested a measure called the function point. Function points are derived using an empirical relationship based on countable (direct) measures of software's information domain and assessments of software complexity. Function points are computed by completing the table shown below. Weighing Factor Measuring Parameter Count SimpleAverage Complex Number of user inputs ------- X 3 4 6 = ------- Number of user outputs ------- X 4 5 7 = ------- Number of user inquiries ------- X 3 4 6 = ------- Number of files ------- X 7 10 15 = ------- Number of external interfaces ------ X 5 7 10 = ------- Count = Total ------------------------------------------------------------------------------------- = ------- 34 Five information domain characteristics are determined and counts are provided in 89 the appropriate table location. Information domain values are defined in the following manner: Number of user inputs. Each distinct inquiry is counted. Number of files. Each logical master file (i.e., a logical grouping of data that may be one part of a large database or a separate file) is counted. To compute function points (FP), the following relationship is used: FP = count total X [0.65 + 0.01 X Σ(Fi)] where count total is the sum of all FP entries obtained. The Fi (i = 1 to 14) are "complexity adjustment values" based on responses to the following questions: 1. Does the system require reliable backup and recovery? 2. Are data communications required? 3. Are there distributed processing functions? 4. Is performance critical? 5. Will the system run in an existing, heavily utilized operational environment? 6. Does the system require on-line data entry? 7. Does the on-line data entry require the input transaction to be built over multiple screens or operations? 8. Are the master files updated on-line? 9. Are the inputs, outputs, files, or inquiries complex? 10. Is the internal processing complex? 11. Is the code designed to be reusable? 12. Are conversion and installation included in the design? 35 13. Is the system designed for multiple installations in different organizations? 14. Is the application designed to facilitate change and ease of use by the user? Each of these questions is answered using a scale that ranges from 0 (not important or applicable) to 5 (absolutely essential). The constant values in Equation and the weighting factors that are applied to information domain counts are determined empirically. Once function points have been calculated, they are used in a manner analogous to LOC as a way to normalize measures for software productivity, quality, and other attributes: • Errors per FP. • Defects per FP. • $ per FP. • Pages of documentation per FP. • FP per person-month.

***Software Quality Metrics:***

Metrics for Software Quality The overriding goal of software engineering is to produce a high-quality system, application, or product. To achieve this goal, software engineers must apply effective methods coupled with modern tools within the context of a mature software process. In addition, a good software engineer must measure if high quality is to be realize.The project manager must also evaluate quality as the project progresses. Metrics derived from these measures provide an indication of the effectiveness of individual and group software quality assurance and control activities. Metrics such as work product errors per function point, errors uncovered per review hour, and errors uncovered per testing hour provide insight into the efficacy of each of the activities 37 implied by the metric. Error data can also be used to compute the defect removal efficiency (DRE) for each process framework activity.

**Question # 2:**

**What is Management spectrum and 4p’s in software Engineering.\**

**ANS:*The Management Spectrum:***

**Effective software project management focuses on these items (in this order)**

**The people**

**Deals with the cultivation of motivated, highly skilled people**

**Consists of the stakeholders, the team leaders, and the software team**

**The product**

**Product objectives and scope should be established before a project can be planned**

**The process**

**The software process provides the framework from which a comprehensive plan for software development can be established**

**The project**

**Planning and controlling a software project is done for one primary reason…it is the only known way to manage complexity**

**In a 1998 survey, 26% of software projects failed outright, 46% experienced cost and schedule overruns**

***The People: The Stakeholders***

**Five categories of stakeholders**

**Senior managers – define business issues that often have significant influence on the project**

**Project (technical) managers – plan, motivate, organize, and control the practitioners who do the work**

**Practitioners – deliver the technical skills that are necessary to engineer a product or application**

**Customers – specify the requirements for the software to be engineered and other stakeholders who have a peripheral interest in the outcome**

**End users – interact with the software once it is released for production use**

***The Product***

**he scope of the software development must be established and bounded**

**Context – How does the software to be built fit into a larger system, product, or business context, and what constraints are imposed as a result of the context?**

**Information objectives – What customer-visible data objects are produced as output from the software? What data objects are required for input?**

**Function and performance – What functions does the software perform to transform input data into output? Are there any special performance characteristics to be addressed?**

**Software project scope must be unambiguous and understandable at both the managerial and technical levels**

**Problem decomposition**

**Also referred to as partitioning or problem elaboration**

**Sits at the core of software requirements analysis**

**Two major areas of problem decomposition**

**The functionality that must be delivered**

**The process that will be used to deliver it**

***The Process***

**Getting Started**

**The project manager must decide which process model is most appropriate based on**

**The customers who have requested the product and the people who will do the work**

**The characteristics of the product itself**

**The project environment in which the software team works**

**Once a process model is selected, a preliminary project plan is established based on the process framework activities**

**Process decomposition then begins**

**The result is a complete plan reflecting the work tasks required to populate the framework activities**

**Project planning begins as a melding of the product and the process based on the various framework activities**

***The Project: A Common Sense Approach***

**Start on the right foot**

**Understand the problem; set realistic objectives and expectations; form a good team**

**Maintain momentum**

**Provide incentives to reduce turnover of people; emphasize quality in every task; have senior management stay out of the team’s way**

**Track progress**

**Track the completion of work products; collect software process and project measures; assess progress against expected averages**

**Make smart decisions**

**Keep it simple; use COTS or existing software before writing new code; follow standard approaches; identify and avoid risks; always allocate more time than you think you need to do complex or risky tasks**

**Conduct a post mortem analysis**

**Track lessons learned for each project; compare planned and actual schedules; collect and analyze software project metrics; get feedback from teams members and customers; record findings in written foRm**

**Question # 3:**

**What is transaction and transform mapping briefly explain with example.**

**ANSWER:**

|  |  |
| --- | --- |
| **Transform Mapping:**  Transform mapping is a set of design steps that allows a DFD with transform flow characteristics to be mapped into a specific architectural style. In this section transform mapping is described by applying design steps to an example system—a portion of the SafeHome security software.  **Example:** The SaeHome security system is representative of many computer-based products and systems in use today. The product monitors the real world and reacts to changes that it encounters. It also interacts with a user through a series of typed inputs and alphanumeric displays. The level 0 data flow diagram for SafeHome, is shown in figure  [Figure 1](http://3.bp.blogspot.com/-xtmgyAVqXGM/T7z5wCSdR_I/AAAAAAAAARk/IlMnNNQIlvI/s1600/Capture.PNG)  During requirements analysis, more detailed flow models would be created for SafeHome. In addition, control and process specifications, a data dictionary, and various behavioral models would also be created. | **Transaction Mapping:**  In many software applications, a single data item triggers one or a number of information flows that effect a function implied by the triggering data item. The data item, called a transaction, and its corresponding flow characteristics . In this section we consider design steps used to treat transaction flow.  **Example:**Transaction mapping will be illustrated by considering the user interaction subsystem of the SafeHome software.  [IMG_256](http://1.bp.blogspot.com/--DVdhoQxy2w/T7z-40JZJwI/AAAAAAAAAS8/1UxENLxvE6o/s1600/Capture.PNG)  As shown in the figure, user commands flows into the system and results in additional information flow along one of three action paths. A single data item, command type, causes the data flow to fan outward from a hub. Therefore, the overall data flow characteristic is transaction oriented.  It should be noted that information flow along two of the three action paths accommodate additional incoming flow (e.g., system parameters and data are input on the "configure" action path). Each action path flows into a single transform, display messages and status. |

***Question # 4:***

***What is design Pattern list down types of Design pattern?***

Design patterns represent the best practices used by experienced object-oriented software developers. Design patterns are solutions to general problems that software developers faced during software development. These solutions were obtained by trial and error by numerous software developers over quite a substantial period of time.

## Types of Design Pattern

As per the design pattern reference book **Design Patterns - Elements of Reusable Object-Oriented Software** , there are 23 design patterns. These patterns can be classified in three categories: Creational, Structural and behavioral patterns. We'll also discuss another category of design patterns: J2EE design patterns.

|  |  |
| --- | --- |
| **S.N.** | **Pattern & Description** |
| 1 | **Creational Patterns** These design patterns provides way to create objects while hiding the creation logic, rather than instantiating objects directly using new operator. This gives program more flexibility in deciding which objects need to be created for a given use case. |
| 2 | **Structural Patterns** These design patterns concern class and object composition. Concept of inheritance is used to compose interfaces and define ways to compose objects to obtain new functionalities. |
| 3 | **Behavioral Patterns** These design patterns are specifically concerned with communication between objects. |
| 4 | **J2EE Patterns** These design patterns are specifically concerned with the presentation tier. These patterns are identified by Sun Java Center. |