AIMAN SIDDIQUA - 2K18/MC/008

SOFTWARE ENGINEERING

SURPRISE TEST-2

1. Characteristics of a good design:

(a) Correctness - The evaluators check the Software for every kind of input and action & observe the results obtained according to the proposed design. If the results are correct for every input, the design is accepted and is considered that the software produced according to the design will function correctly.

(b) Understandability - Good software design should be self explanatory. If the design is easy & self explanatory, it would be easy for developers to implement it & build the same software that is represented in the design.

(c) Efficiency - It is the ability of the software to use system resources in the most effective and efficient manner. Functions required are efficiency in time and in resources

(d) Maintainability - It refers to the ability and easy with which the modification can be made in a software design. Any change made in the software design must not affect the other available features and if the features are getting affected they must be handled properly.

Main activities in the Software design process:

(i) Architectural design
It defines the overall structure of the system, the main components, their relationships.

(ii) Interface design
It defines the interfaces b/w the components. The interface specification must be clear. Therefore a component can be used without having to know how its implemented.

(iii) Component design
Take each component and design how it will operate, with the specific design left to the programmer, or a list of changes to be made to a reusable component.

(iv) Database design
The system data structures are designed and their representation in a database is designed. This depends on whether an existing database it to be reused or a new database is to be created.

Design outputs:
(i) System architecture
(ii) Database specification
(iii) Interface specification
(iv) Component specification

(b) Levels of COCOMO model

(i) BASIC COCOMO MODEL

It takes the form

$E = a_b (KLOC)^{b_b}$

$D = c_b (E)^{d_b}$

KLOC = Kilo lines of code

where E = Effort applied in Person Months

D = Development time in months.

a_b, b_b, c_b, d_b = Constants for each group of software products.

Average staff size = $\frac{\text{Effort}}{\text{Development Time}}$

Productivity = $\frac{KLOC}{E}$

(ii) INTERMEDIATE COCOMO MODEL

The basic COCOMO model considers that the effort is only a function of the number of lines of code and some constants calculated according to the various software systems. The intermediate Cocomo model refines the initial estimates obtained through the basic Cocomo model by using a set of 15 drivers based on various attributes of Software engineering.

Cost drivers:

- Product attributes
- Hardware attributes
- Personal attributes
- Project attributes

(iii) DETAILED COCOMO MODEL.

It uses various effort multipliers for each cost driver property. The whole software is differentiated into multiple modules and then we apply COCOMO in various modules to estimate effort and then sum the effort.

$$\text{Effort} = x_1 \times (\text{KLOC})^{x_2}$$

KLOC = The estimated size of software product expressed in Kilo lines of codes

x_1 and x_2 are constants for each category of software product.

Detailed COCOMO

Phase sensitive effort multipliers

Cost Drivers → Design & test

Manpower allocation for each phase

Three level product hierarchy

Modules subsystem

System level.

2. (a) We need software metrics to determine the quality of the current product or process, improve the quality & predict the quality once the software development project is complete.

Areas of applications of software metrics

- Size & cost estimation techniques.
- The prediction of quality levels of software often in terms of reliability.
- To provide quantitative checks on software design

Advantages of using LOC as a metric

* Widely used & universally accepted
* permits comparison of size & productivity metrics b/w diverse development groups
* Directly relates to the end product.
* Measures software from the developers point of view, what he actually does.
* Continuous improvement activities exist for estimation techniques.

Disadvantages of using LOC as a metric:

* Difficult to measure LOC in the early stages of a new product.
* Software instructions vary with coding languages, design methods and with programmers ability.
* No industry standard for measuring LOC.
* LOC cannot be used for normalizing if platform and languages are different.
* The only way to predict LOC for a new app to be developed is through analogy based on similar software application.

2. (b) Risk is a problem that may cause some loss or threaten the success of the project but has not happened yet.

Various risks that can be incured in a software development project.

(i) Dependencies on outside agencies or factors
- Availability of trained, experienced persons
- Inter group dependencies.
- Customer furnished items or information
- Internal or external subcontractor relationships

(ii) Requirment issues
- Lack of clear product vision
- Unprioritized requirements
- Lack of agreement on product requirements
- New market with uncertain needs
- Rapidly changing requirements

(iii) Management Issues:
- Inadequate planning
- Inadequate visibility into actual product status
- Unclear project ownership & decision making
- Staff personality conflicts
- Poor communication

(iv) Lack of knowledge
- Inadequate training
- Poor understanding of methods, tools and techniques
- Inadequate application domain experience
- New Technologies
- Ineffective, poorly documented or neglected processes.

(v) Other risk categories
- Unavailability of adequate testing facilities
- Turnover of essential personnel.
- Unachievable performance requirements
- Technical approaches that may not work.

Risk Management Activities include:

1. Risk Assessment
   - Rist Identification
   - Risk Analysis
   - Risk Prioritization

2. Risk Control
   - Risk Management Planning
   - Risk Monitoring.
   - Risk Resolution.