Software Engineering
IAT 3/Improvement Test Scheme and Solution (15 Nov 2016)

1. a) List and explain the attributes of good software. [6M]

The software should :-

1) deliver functionality as specified by customer
2) deliver expected performance to the user ;
and should be :-
3) maintainable - easy to evolve in changing requirements / environment
4) dependable - provide reliability, security, safety, etc.
5) efficient - make efficient use of available resources like memory & processor cycles. Includes responsiveness, processing time, memory utilisation, etc.
6) usable - easy to use with appropriate user interface and adequate documentation

(6M)

b) What are the differences between generic and specific software development. [4M]

Types of software Product

Generic products	Customised (or bespoke) products
→ Stand alone	→ software developed for a particular customer
→ produced by development org.	→ developed on order by customer
→ sold on open market to any customer who is able to buy them	
→ software specification is controlled by developing organization.	→ software specification is controlled by customer
→ Eg:- databases, word processors, etc.	→ Eg. Air traffic control system.

(2x1M definition)

(2x1M example)

2. Discuss the professional and ethical responsibilities of a software engineer. [10M]
(if individually explained 5x2M)
(if individually not explained 10x1M)

Ethical responsibility

- Honesty
- Integrity

Professional responsibility

- Confidentiality – Respect confidentiality of your employers, or clients irrespective of whether a formal confidentiality agreement has been signed.
- Competence – Donot ~~misisite~~ misrepresent your level of competence. Accept only that work which you can accomplish.
- Intellectual property rights – Be aware of local laws governing use of intellectual property such as patents & copyrights. Also protect intellectual property of employers & clients.
- Computer misuse – Donot ~~misuse~~ use your technical skills to misuse others's computers eg for game playing or any other serious issue like dissemination of viruses.

- Public – Act consistently with public interest.
- Client and Employer – Act in favor of client, employer and public
- Product – Ensure that products and related modifications meet the highest professional standards possible.
- Judgment – Maintain integrity and independence in professional judgment.
- Management – SE managers and leaders shall subscribe to and promote an ethical approach to management of software development and maintenance.
- Profession – Advance integrity and reputation of the profession consistent with public interest.
- Colleagues – Be fair to and supportive of your colleagues.
- Self – Participate in lifelong learning regarding the practice of your profession and promote an ethical approach to practice of the profession.

3. Distinguish between verification & validation. Discuss the steps in software testing with a neat diagram. [10M]

Validation – Are we building the right product ?
To ensure that software system meets the customer's expectations.

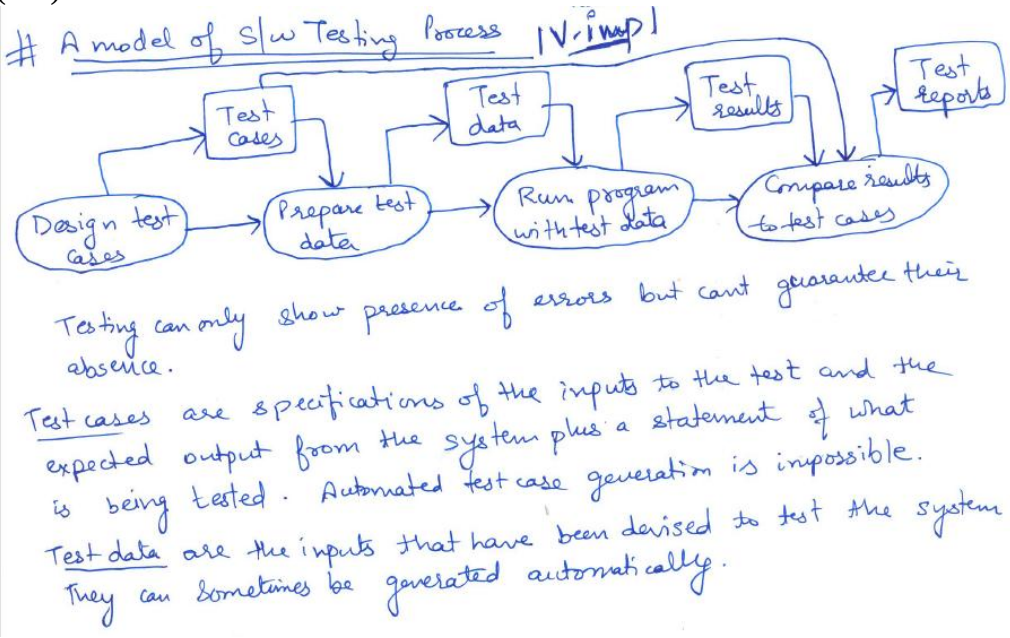
Verification – Are we building the product right ?
To ensure that software confirms to its specification.

V & V process has 2 complementary approaches to system checking & analysis:-

(1) Software inspections or peer reviews – Analyze and check system representations such as requirements document, design diagrams and program source code

(2) Software testing – Involves running an implementation of the software with test data.

(3M)



A model of S/w Testing Process (V.imp)

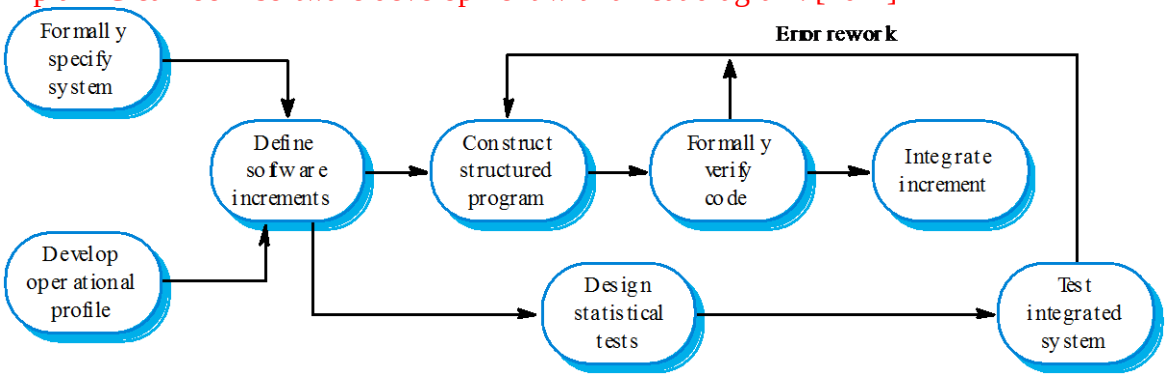
Testing can only show presence of errors but cant guarantee their absence.

Test cases are specifications of the inputs to the test and the expected output from the system plus a statement of what is being tested. Automated test case generation is impossible.

Test data are the inputs that have been devised to test the system. They can sometimes be generated automatically.

(4M diagram+3M explanation)

4. Explain Cleanroom software development with a neat diagram. [10M]



(4M diagram)

Cleanroom S/w Development v.imp

Objective of this approach is zero-defect S/w.
'Cleanroom' was derived by analogy with semiconductor fabrication units where defects are avoided by manufacturing in an ultra-clean atmosphere.

Cleanroom approach is based on 5 key strategies:-

1. Formal specification - The s/w to be developed is formally specified. State transition model is used to express specification
2. Incremental development - S/w is partioned into increments that are developed & validated separately using cleanroom process. Increments are specified using customer input at early stage
3. Structured programming - Limited number of control and data abstraction constructs are used to systematically transform specification to create program code
4. Static verification - The developed s/w is statically verified using rigorous s/w inspections. There is no unit or module testing process for code components
5. Statistical testing of system - Integrated s/w is statistically tested to determine reliability.

3 Teams are involved in cleanroom process:

1. The specification team - This group is responsible for developing and maintaining the system specification. They produce user requirements & mathematical specifications for verification
2. The development team - This team is responsible for developing and verifying the s/w. Inspections are used for verification
3. The certification team - This team is responsible for developing a set of statistical tests to exercise the s/w after it has been developed. These tests are based on formal specification. Test case development is carried out in parallel with s/w development. The test cases are used to certify the s/w reliability.

(3M strategies + 3M teams)

5. Explain the following:
   a) Unit/Component Testing [2M]

Component Testing (Unit testing)

It is the process of testing individual components in the system.
Its goal is to expose faults in these components.

Types of components:

- Individual functions / methods within an object.
- Object classes that have several attributes and methods
- Composite components made up of several different objects or functions. Composite components have a defined interface that is used to access their functionality.

(2M)

b) Integration Testing, interface types and interface errors [8M]
(Integration testing- 3M, interface types- 3M, interface errors- 2M)

Integration testing checks that the components that are integrated actually work together, are called correctly and transfer the right data at the right time across their interfaces.

System integration

Top-down
Develop the skeleton of the system and populate it with components.
Stubs are required.

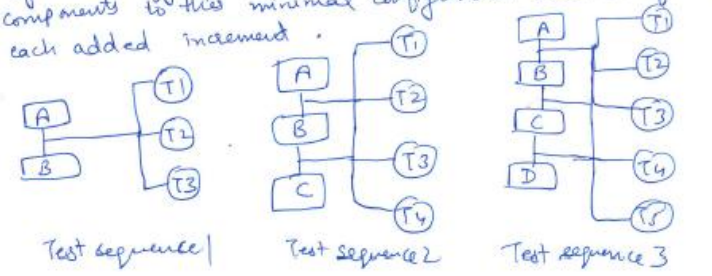
Bottom-up
Integrate infrastructure components that provide common services, like n/w and database access then add functional components.
Drivers are required.

Hybrid / Incremental
Mixture of top-down & bottom-up.
Both infrastructure components & functional components are added in increments.

Stub – dummy modules for called program
Driver – dummy module for calling program

A → B
A calls B
A is calling B is called.

To simplify error localisation, systems should be incrementally integrated. Initially, integrate a minimal system configuration and test this system. Then add components to this minimal configuration and test after each added increment.



Test sequence 1 Test sequence 2 Test sequence 3

A, B, C, D are components. T1 to T5 are related sets of tests of features incorporated in system. (2)
T1, T2, T3 are first run on a system composed of component A and component B (minimal system). If they reveal defects, they are corrected. Component C is integrated & T1, T2, T3 are repeated to ensure that there have not been unexpected interactions with A & B. If problems arise means new component is culprit. Source of problem is localised, thus simplifying defect location and repair. Test set T4 is also run. Finally D is integrated and tested using existing & new tests (T5).

Order of integration or increment may be decided by customer (based on customer priority) or by developer (like most frequently used functionality first).

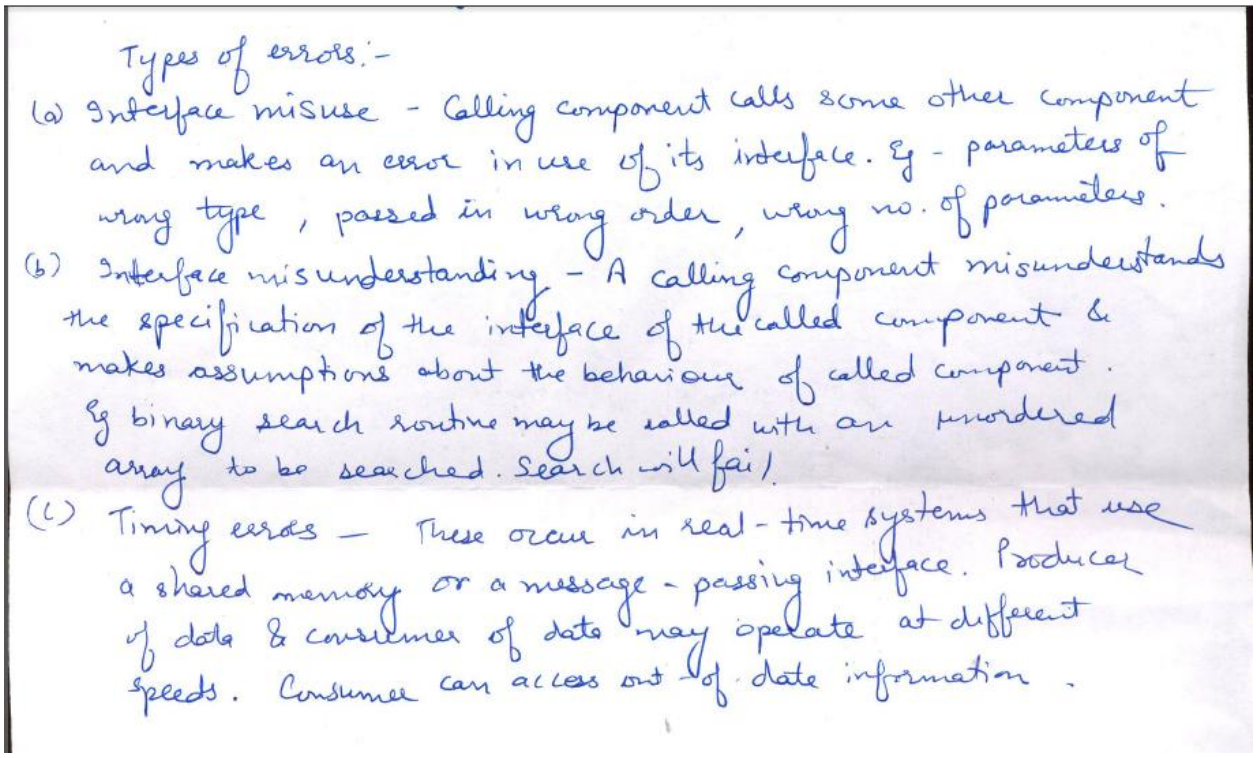
Types of interfaces :-

(a) Parameter interfaces - data or function references are passed from one component to another.

(b) Shared memory interfaces - block of memory is shared b/w components. Data is placed in memory by one sub-system and retrieved from there by other subsystems (4)

(c) Procedural interfaces - one component encapsulates a set of procedures that can be called by other components. Objects and reusable components have this form of interface

(d) Message passing interfaces - one component requests a service from another component by passing a message to it. eg client server systems.



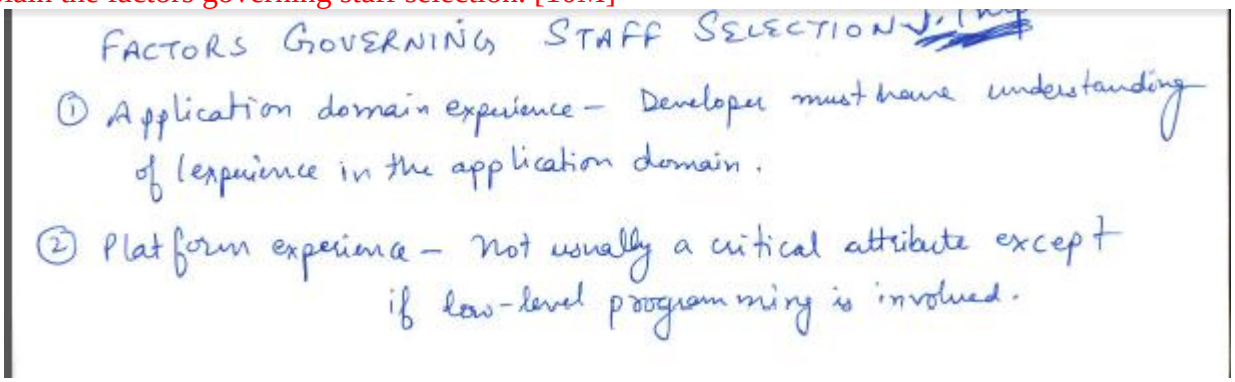
Types of errors:-

(a) Interface misuse - Calling component calls some other component and makes an error in use of its interface. Eg - parameters of wrong type, passed in wrong order, wrong no. of parameters.

(b) Interface misunderstanding - A calling component misunderstands the specification of the interface of the called component & makes assumptions about the behaviour of called component. Eg binary search routine may be called with an unordered array to be searched. Search will fail.

(c) Timing errors — These occur in real-time systems that use a shared memory or a message-passing interface. Producer of data & consumer of data may operate at different speeds. Consumer can access out of date information.

6. Explain the factors governing staff selection. [10M]



FACTORS GOVERNING STAFF SELECTION

① Application domain experience — Developer must have understanding of experience in the application domain.

② Platform experience — Not usually a critical attribute except if low-level programming is involved.

(3) Programming language experience – Significant for short duration projects as not much time is there for learning a new language

(4) Problem solving ability – Important for software engineers who constantly have to solve technical problems. Difficult to judge.

(5) Educational background – This provides an indicator of the basic fundamentals that candidate should know and of their ability to learn

(6) Communication ability – Important because project staff is required to communicate orally and in writing with other engineers, managers and customers.

(7) Adaptability – may be judged by looking at different types of experience that candidates have had. It indicates ability to learn

(8) Attitude – Should have positive attitude to their work and should be willing to learn new skills. Important but difficult to assess.

(9) Personality – Important but difficult to assess. Candidates must be reasonably compatible with other team members.

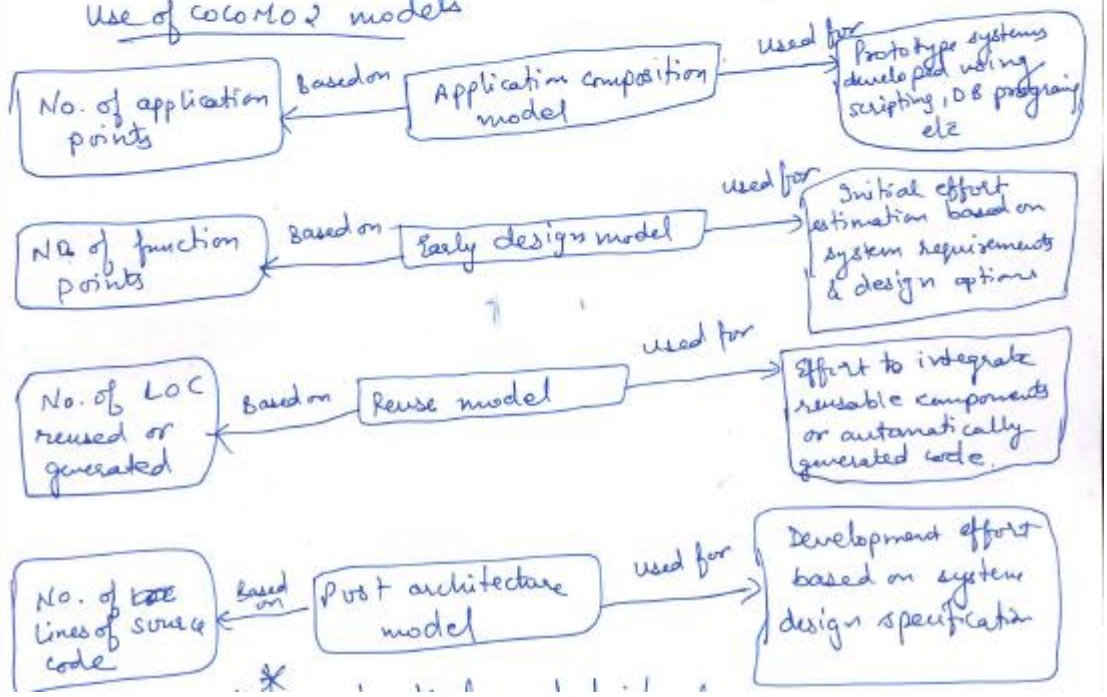
Some companies may take programming attribute, psychometric tests.

(10M)

7. Describe with neat diagram COCOMO II model [10M]
(Diagram- 6M, Explanation- 4M)

- COCOMO 2 incorporates a range of sub-models that produce increasingly detailed S/W estimates.
- The sub-models in COCOMO 2 are:
  1) Application Composition Model - used when S/W is composed from existing parts
  2) Early design model - used when requirements are available but design not yet started.
  3) Reuse model - Used to compute the effort of integrating reusable components
  4) Post architectural model - Used once the system architecture has been designed and more info. about system is available

Use of COCOMO 2 models



8. Define PCMM (People Capability Maturity Model). Draw a block diagram to explain P-CMM levels. [10M]
   (Definition- 1M, Objectives- 2M, Levels- 2M, Diagram- 5M)

The People Capability Maturity Model.

Intended as a framework for managing the development of people involved in s/w development.

P-CMM Objectives:

- To improve organisational capability by improving workforce capability.
- To ensure that s/w development capability is not reliant on a small number of individuals.
- To align motivation of individuals with that of organisation.
- To help retain people with critical knowledge and skills.

Levels:

- Initial – Adhoc people management
- Repeatable – Policies developed for capability improvement
- Defined – Standardised people management across organisation
- Managed – Quantitative goals for people management in place.
- Optimizing – Continuous focus on improving individual competence and workforce motivation.

People-Capability Maturity Model (PC-MM)