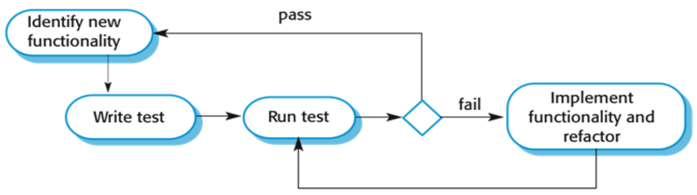
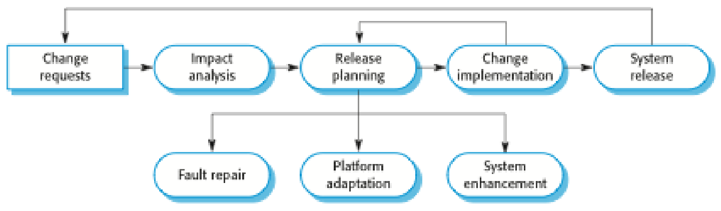
INTERNAL ASSESSMENT TEST 3 – Oct -Dec 2020
Scheme and Solution

Sub:	**SOFTWARE ENGINEERING**				Sub Code:	**18CS35**	Branch:	**ISE**	
Date:	15-12-2020	Duration:	90 min's	Max Marks: 50	Sem / Sec:	III SEM, C Sec			OBE

Answer any 5 Questions (5 X 10 = 50)

1.	a) What is Software Testing? Explain the goals of Software Testing. Ans: Software Testing shows that a program does what it is intended to do and to discover program defects before it is put into use. It is a dynamic validation technique. Explain the goals of Software Testing. 1. To demonstrate that software meets its requirements. • For custom software, at least one test for every requirement in the requirements document. • For generic software products, have tests for all of the system features AND feature combinations. 2. To discover situations in which the behavior of the software is incorrect or undesirable. • Defect testing is concerned with rooting out undesirable system behavior such as system crashes, unwanted interactions with other systems, incorrect computations and data corruption.	[4]	CO4	L2
	b) Explain Test-driven Development (TDD) with a block diagram. Mention the steps that are followed in TDD to test implementation of Calculator. Ans: Test-driven development (TDD) is an approach to program development in which you create tests before writing code. You develop code incrementally, along with a test for that increment. You don't move on to the next increment until the code that you have developed passes its test.	[6]	CO4	L2

	 Identify new functionality pass Write test Run test fail Implement functionality and refactor **TDD process activities** 1 Start by identifying the increment of functionality that is required. This should normally be small and implementable in a few lines of code. 2. Write a test for this functionality and implement this as an automated test. 3. Run the test, along with all other tests that have been implemented. Initially, you have not implemented the functionality so the new test will fail. 4. Implement the functionality and re-run the test. 5. Once all tests run successfully, you move on to implementing the next chunk of functionality.			
2.	a) Consider a scenario of an Online Exam Application; write one test case each to do Unit Testing, Component Testing and System Testing. Mention the difference among them.	[4]	CO4	L4
	b) Write the Test cases(Unit testing) for addition, substraction, multiplication and division operation and check how many test cases pass or fail. If test case fail, give solution to fix the bug. Note: 1. Write at least one test case for each function. 2. Use table with column names as, Test case no., Test Case, Expected output, Actual output, Test case pass/fail. 3. Add at least one test case that fail.	[6]	CO4	L4
3.	a) With appropriate block diagram explain the software evolution process. Ans:	[4]	CO4	L2



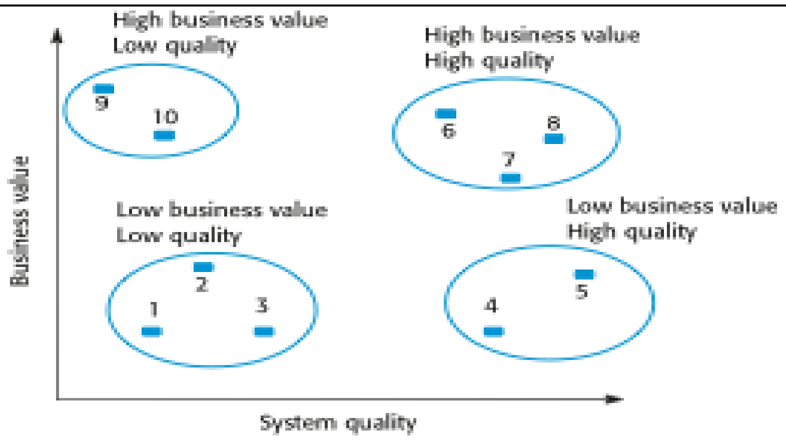
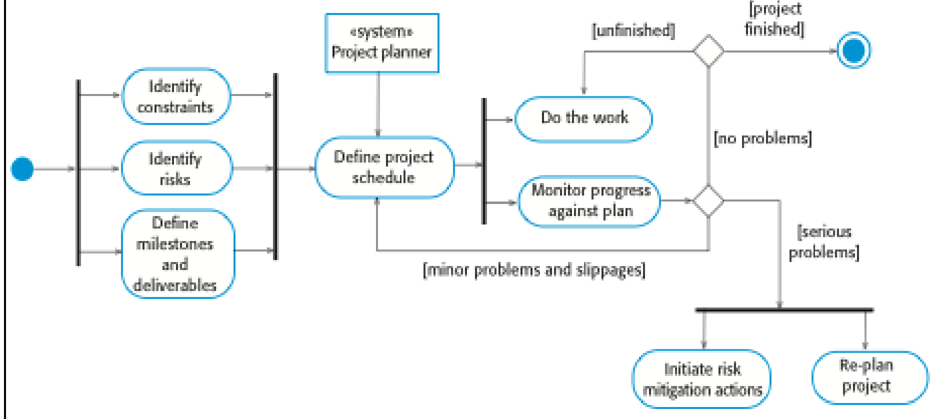
- Evolution
  - The stage in a software system's life cycle where it is in operational use and is evolving as new requirements are proposed and implemented in the system.
- Servicing
  - At this stage, the software remains useful but the only changes made are those required to keep it operational i.e. bug fixes and changes to reflect changes in the software's environment. No new functionality is added.
- Phase-out
  - The software may still be used but no further changes are made to it.

b) List the 'Lehman's Law' concern to system change. Explain any 4 'Lehman's Law'.	[2+4]	CO4	L2

Law	**Description**
Continuing change	A program that is used in a real-world environment must necessarily change, or else bec
Increasing complexity	As an evolving program changes, its structure tends to become more complex. Extra reso structure.

Large program evolution	Program evolution is a self-regulating process. System attributes such as size, tim invariant for each system release.	mber of rep
Organizational stability	Over a program's lifetime, its rate of development is approximately constant and	devoted to sy

4.	a) What is the need of Software maintenance? Differentiate Corrective, Adaptive and Perfective maintenance. • ANS: Need of maintenance modifying a program after it has been put into use. • The term is mostly used for changing custom software. Generic software products are said to evolve to create new versions. • Maintenance does not normally involve major changes to the system's architecture. • Changes are implemented by modifying existing components and adding new components to the system. Types of maintenance • Corrective maintenance: Maintenance to repair software faults • Changing a system to correct deficiencies in the way meets its requirements. • Adaptive Maintenance : to adapt software to a different operating environment • Changing a system so that it operates in a different environment (computer, OS, etc.) from its initial implementation. Perfective Maintenance : to add to or modify the system's functionality • Modifying the system to satisfy new requirements.	[5]	CO4	L3
	b) Explain Legacy system management depending on system quality and its business value strategy.	[5]	CO4	L2

	 High business value Low quality High business value High quality Low business value Low quality Low business value High quality Business value System quality 1 2 3 4 5 6 7 8 9 10 • Low quality, low business value • These systems should be scrapped. • Low-quality, high-business value • These make an important business contribution but are expensive to maintain. Should be re-engineered or replaced if a suitable system is available. • High-quality, low-business value • Replace with COTS, scrap completely or maintain. • High-quality, high business value • Continue in operation using normal system maintenance.			
5.	a) With a block diagram, explain the project planning process.  «system» Project planner Identify constraints Identify risks Define milestones and deliverables Define project schedule Do the work Monitor progress against plan [unfinished] [project finished] [no problems] [minor problems and slippages] [serious problems] Initiate risk mitigation actions Re-plan project • Project planning is an iterative process that starts when you create an initial project plan during the project startup phase. • Plan changes are inevitable.	[6]	CO5	L2

	– As more information about the system and the project team becomes available during the project, you should regularly revise the plan to reflect requirements, schedule and risk changes. – Changing business goals also leads to changes in project plans. As business goals change, this could affect all projects, which may then have to be re-planned.			
	b) Discuss an advantage and a disadvantage of Plan-driven development in Project planning. Ans: • The arguments in favor of a plan-driven approach are that early planning allows organizational issues (availability of staff, other projects, etc.) to be closely taken into account, and that potential problems and dependencies are discovered before the project starts, rather than once the project is underway. • The principal argument against plan-driven development is that many early decisions have to be revised because of changes to the environment in which the software is to be developed and used.	[4]	CO5	L3
6.	Explain the 1. Application composition • Supports prototyping projects and projects where there is extensive reuse. • Based on standard estimates of developer productivity in application (object) points/month. • Takes CASE tool use into account. • Formula is – PM = (NAP´(1 - %reuse/100)) / PROD – PM is the effort in person-months, NAP is the number of application points and PROD is the productivity. 2. Early design • Estimates can be made after the requirements have been agreed. • Based on a standard formula for algorithmic models – PM = A´Size$^{B´}$M where	[2.5X4=10]	CO5	L3

- M = PERS ´ RCPX ´ RUSE ´ PDIF ´ PREX ´ FCIL ´ SCED;
- A = 2.94 in initial calibration, Size in KLOC, B varies from 1.1 to 1.24 depending on novelty of the project, development flexibility, risk management approaches and the process maturity.

3. Reuse

- Takes into account black-box code that is reused without change and code that has to be adapted to integrate it with new code.
- There are two versions:
  - Black-box reuse where code is not modified. An effort estimate (PM) is computed.
  - White-box reuse where code is modified. A size estimate equivalent to the number of lines of new source code is computed. This then adjusts the size estimate for new code.
- For generated code:
  - PM = (ASLOC * AT/100)/ATPROD
  - ASLOC is the number of lines of generated code
  - AT is the percentage of code automatically generated.
  - ATPROD is the productivity of engineers in integrating this code.

4. Post-architecture

- Uses the same formula as the early design model but with 17 rather than 7 associated multipliers.
- The code size is estimated as:
  - Number of lines of new code to be developed;
  - Estimate of equivalent number of lines of new code computed using the reuse model;
  - An estimate of the number of lines of code that have to be modified according to requirements changes.

COCOMO cost estimation models with formula.