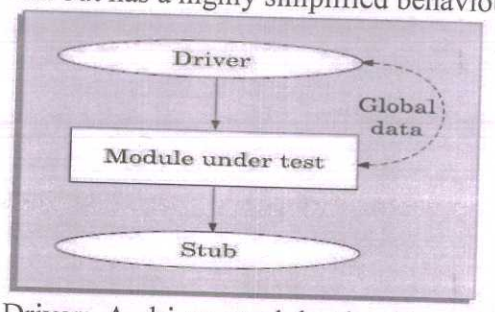
REVISION : 2015

COURSE CODE : 5132

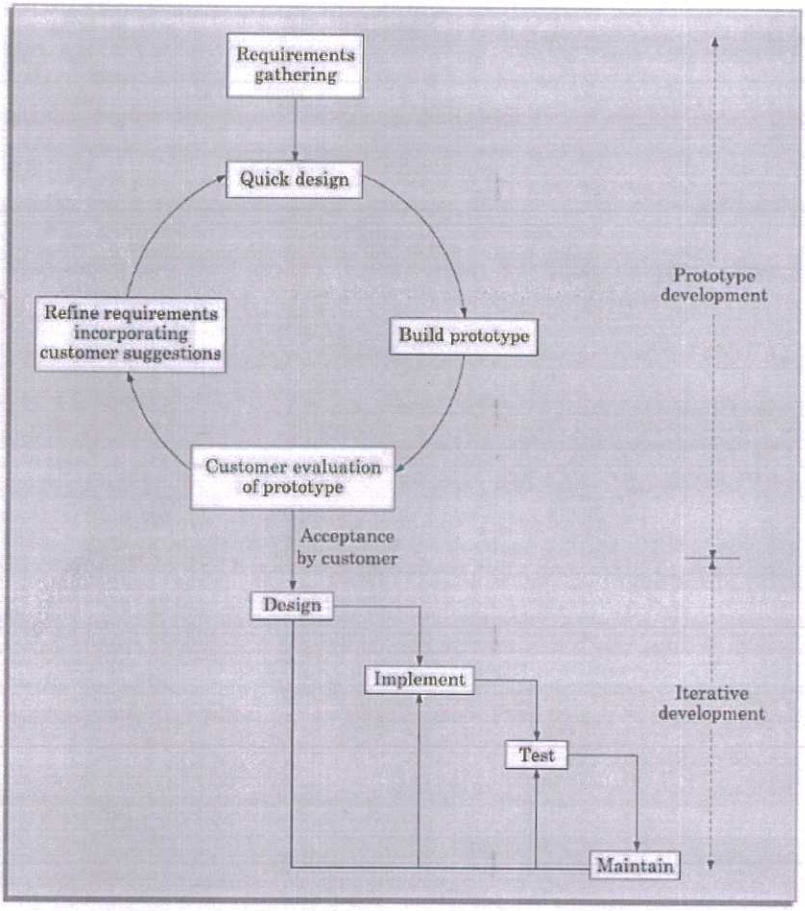
COURSE TITILE: PROJECT MANAGEMENT AND SOFTWARE ENGINEERING

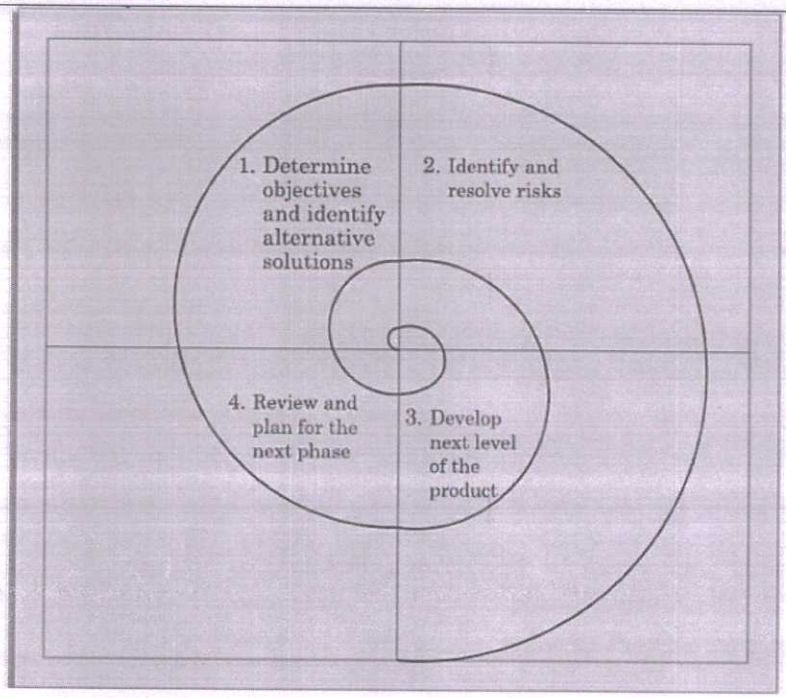
Question no	Scoring indicators	Split up score	Sub total	Total
	PART -- A			
I 1.	1. Unavailability of feedback paths to correct errors in previous stages 2. More time is required since no overlapping between phases are allowed	2	2	
2.	Structured, Completeness, Unambiguity, Consistent .	2	2	
3.	A triplet [I,S,R] I-Input to the program S-State of the program R-Result expected	2	2	10
4.	Project Risk, Technical Risk, Business Risk	2	2	
5.	Portability, Usability, Reusability, Correctness	2	2	
	PART – B			
II . 1	The agile model was primarily designed to help a project to adapt to change requests quickly.1Thus, a major aim of the agile models is to facilitate quick project completion. Agile model is being used as an umbrella term to refer to a group of development processes. These processes share certain common characteristics, but do have certain subtle differences among themselves. Afew popular agile SDLC models are the following: Crystal Scrum Extreme programming (XP) Lean development Unified process In the agile model, the requirements are decomposed into many small parts that can be incrementally developed. The agile model adopts an iterative approach. Each incremental part is developed over an iteration. Eachiteration is intended to be small and easily manageable and lasting f o r a couple of weeks only. At a time, only one increment is planned, developed, and then deployed at the customer site. No long-term plans are made.	6	6	

2.	Maintenance is required in the following three types of situations: Corrective maintenance: This type of maintenance is carried out to correct errors that were not discovered during the product development phase. Perfective maintenance: This type of maintenance is carried out to improve the performance of the system, or to enhance the functionalities of the system based on customer's requests. Adaptive maintenance: Adaptive maintenance is usually required for porting the software to work in a new environment.	3*2	6	
3.	Users, customers, and marketing personnel: These stakeholders need to refer to the SRS document to ensure that the system as described in the document will meet their needs. Software developers: The software developers refer to the SRS document to make sure that they are developing exactly what is required by the customer. Test engineers: The test engineers use the SRS document to understand the functionalities, and based on this write the test cases to validate its working. User documentation writers: The user documentation writers need to read the SRS document to ensure that they understand the features of the product well enough to be able to write the users' manuals. Project managers: The project managers refer to the SRS document to ensure that they can estimate the cost of the project easily by referring to the SRS document and that it contains all the information required to plan the project. Maintenance engineers: The SRS document helps the maintenance engineers to understand the functionalities supported by the system. A clear knowledge of the functionalities can help them to understand the design and code.	6	6	
4.	• Unlike function-oriented design methods in OOD, the basic abstraction is not the services available to the users of the system such as issuebook, display-book-details, find-issued-books, etc., but real-world entities such as member, book, book-register, etc. • In OOD, state information exists in the form of data distributed among several objects of the system. In contrast, in a procedural design, the state information is available in a centralised shared data store. • Function-oriented techniques group functions together if, as a group, they constitute a higher level function. On the other hand, object oriented techniques group functions together on the basis of the data they operate on.	6	6	30

5.	Unit testing is undertaken after a module has been coded and reviewed. Besides the module under test, the following are needed to test the module: The procedures belonging to other modules that the module under test calls. Non-local data structures that the module accesses. A procedure to call the functions of the module under test with appropriate parameters. Stub: A stub procedure is a dummy procedure that has the same I/O parameters as the function called by the unit under test but has a highly simplified behaviour.  Driver: A driver module should contain the non-local data structures accessed by the module under test. Additionally, it should also have the code to call the different functions of the unit under test with appropriate parameter values for testing.	6	6	
6.	Coding guidelines provide some general suggestions regarding the coding style to be followed but leave the actual implementation of these guidelines to the discretion of the individual developers. • Do not use a coding style that is too clever or too difficult to understand. • Avoid obscure side effects. • Do not use an identifier for multiple purposes. • Code should be well-documented. • Length of any function should not exceed 10 source lines. • Do not use GO TO statements	6	6	
7.	Configuration management is carried out through two principal activities. 1.Configuration identification : involves deciding which parts of the system should be keep track of. 2.Configuration control : ensures that changes to a system happen smoothly Configuration identification •Project manager classifies the objects associated with software development into 3 categories. Controlled, pre controlled and uncontrolled. Controlled objects are those already put under the			

	configuration tool. Eg: design documents, test cases, source code. Pre controlled objects are not yet under the configuration control but will eventually be under the control. Uncontrolled objects are not and will not be subject to configuration control. Configuration Control Configuration control is the process of managing changes to controlled objects. Configuration control allows only authorized changes and prevents unauthorized changes. In order to change a controlled object such as module a developer can get a private copy of the module by a reserve operation.Configuration management tool allow only one person to reserve a module at any time.Restoring the changed module to the system configuration requires the permission of a change control board (CCB).For every change that needs to be carried out, CCB reviews the changes made to the controlled object and certifies several things about the change as follows: 1. Change is well motivated. 2. Developer has considered and documented the effects of the change. 3. Changes interact well with changes made by other developers. 4. Appropriate people (CCB) have validated the change.	6	6	
III. (a)	PART -- C List out all the phases +Explanation of eachphase 1. Feasibility Study: determine whether it would be financially and technically feasible to develop the software. 2. Requirements collection analysis and specification: collect all relevant information regarding the software to be developed from the customer with a view to clearly understand the requirements. The goal of the requirements analysis activity is to weed out the incompleteness and inconsistencies in these gathered requirements. The identified requirements are documented. This is called a software requirements specification (SRS) document. 3. Design: The goal of the design phase is to transform the requirements specified in the SRS document into a structure that is suitable for implementation in some programming language.	1+(1.5*6)	10	15

	4. Coding and unit testing: translate a software design into source code and to ensure that individually each function is working correctly. 5. Integration and system testing: During the integration and system testing phase, the different modules are integrated in a planned manner. 6. Maintenance			
III(b)		5	5	
IV. (a)	Diagram+Explanation Each loop of the spiral is called a phase of the software process. The exact number of phases through which the product is developed can be varied by the project manager depending upon the project risks. A prominent feature of the spiral model is handling unforeseen risks that can show up much after the project has started. Explain each quadrant in the spiral	3+(1.5*4)	9	15



IV (b)

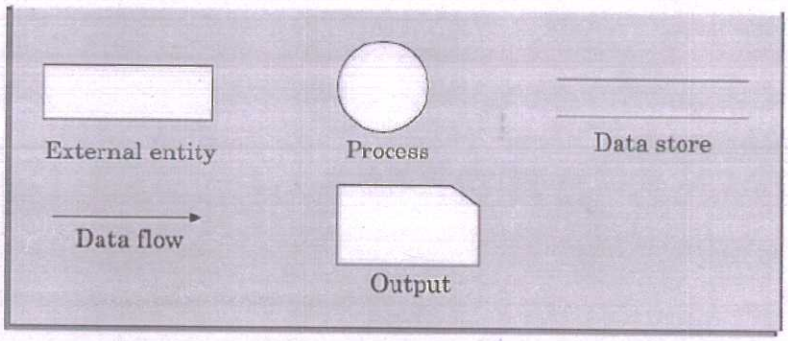
- Development of an overall understanding of the problem: For this, only the the important requirements of the customer need to be understood and the details of various requirements such as the screen layouts required in the graphical user interface (GUI), specific formulas or algorithms required for producing the required results, and the databases schema to be used are ignored.
- Formulation of the various possible strategies for solving the problem: In this activity, various possible high-level solution schemes to the problem are determined.
- Evaluation of the different solution strategies: The different identified solution schemes are analysed to evaluate their benefits and shortcomings. The different solutions are compared based on the estimations that have been worked out. Once the best solution is identified, all activities in the later phases are carried out as per this solution.

3*2 | 6

V. (a)

Expand the following structure
• Functional requirements
• Non-functional requirements
— Design and implementation constraints

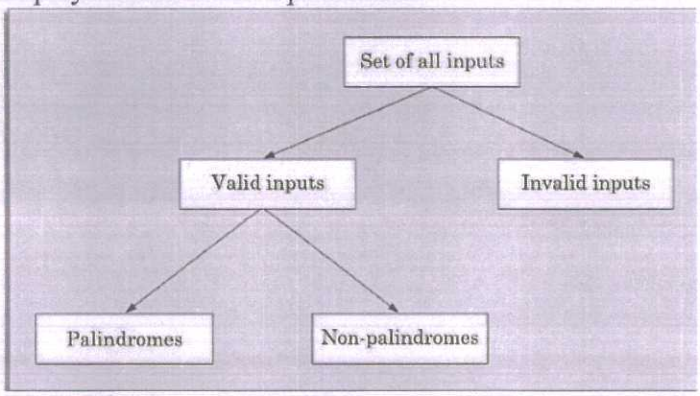
8 | 8

	— External interfaces required — Other non-functional requirements • Goals of implementation			
V (b)	The DFD (also known as the bubble chart) is a simple graphical formalism that can be used to represent a system in terms of the input data to the system, various processing carried out on those data, and the output data generated by the system. Primitivesymbols in DFD  	7	7	15
VI (a)	During detailed design the pseudo code description of the processing and the different data structures are designed for the different modules of the structure chart. These are usually described in the form of module specifications (MSPEC). MSPEC is usually written using structured English. The MSPEC for the non-leaf modules describe the different conditions under which the responsibilities are delegated to the lowerlevel modules. The MSPEC for the leaf-level modules should describe in algorithmic form how the primitive processing steps are carried out. To develop the MSPEC of a module, it is usually necessary to refer to the DFD model and the SRS document to determine the functionality of the module. o/p of Detailed Design :Module Specification(MSPEC)	7	7	
VI (b)	The main purpose of the requirements analysis activity is to analyse the gathered requirements to remove all ambiguities, incompleteness, and inconsistencies from the gathered customer requirements and to obtain a clear understanding of the software to be developed. After the analyst has understood the exact customer requirements, he proceeds to identify and resolve the various problems that he detects in the gathered requirements. During requirements analysis,the analyst needs to identify and resolve three main types of problems in the requirements: • Anomaly • Inconsistency	8	8	15

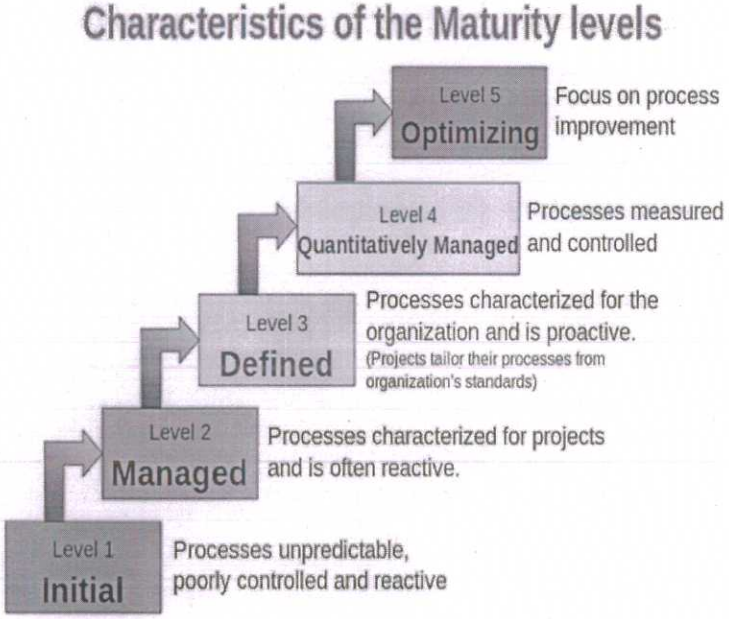
	• Incompleteness			

VII (a)	The principal idea governing the statement coverage strategy is that unless a statement is executed, there is no way to determine whether an error exists in that statement.
	
	A test suite satisfies branch coverage, if it makes each branch condition in the program to assume true and false values in turn.
	Ex:
	int computeGCD(x,y)
	int x,y;
	{
	1 while (x != y){
	2 if (x>y) then
	3 x=x-y;
	4 else y=y-x;
	5 }
	6 return x;
	}
	By choosing the test set {(x = 3, y = 3), (x = 4, y = 3), (x = 3, y = 4)}, all statements of the program would be executed at least once.
	The test suite {(x = 3, y = 3), (x = 3, y = 2), (x = 4, y = 3), (x = 3, y = 4)} achieves branch coverage.
		2*5	10	15

VII (b)	The principal aim of code inspection is to check for the presence of some common types of errors that usually creep into code due to programmer mistakes and oversights and to check whether coding standards have been adhered to. The programmer usually receives feedback on programming style, choice of algorithm, and programming techniques. The other participants gain by being exposed to another programmer's errors. Good software development companies collect statistics regarding different types of errors that are commonly committed by their engineers and identify the types of errors most frequently committed. Such a list of commonly committed errors can be used as a checklist during code inspection to look out for possible errors.	5	5	
VIII(a)	(i)The main idea behind defining equivalence classes of input data is that testing the code with any one value belonging to an equivalence class is as good as testing the code with any other value belonging to the same equivalence class. Ex: Design equivalence class partitioning test suite for a function that reads a character string of size less than five characters and	2*5	10	

	displays whether it is a palindrome.  Selecting one representative value from each equivalence class, we have the required test suite: {abc,aba,abcdef}. (ii) Boundary value analysis-based test suite design involves designing test cases using the values at the boundaries of different equivalence classes. Ex: For a function that computes the square root of the integer values in the range of 0 and 5000, determine the boundary value test suit. There are three equivalence classes—The set of negative integers,the set of integers in the range of 0 and 5000, and the set of integers larger than 5000. The boundary value-based test suite is: {0,-1,5000,5001}.			15
VIII(b)	An *error* is the result of a mistake committed by a developer in any of the development activities. A *failure* of a program essentially denotes an incorrect behaviour exhibited by the program during its execution. An incorrect behaviour is observed either as an incorrect result produced or as an inappropriate activity carried out by the program. A *test case* is a triplet [I , S, R], where I is the data input to the program under test, S is the state of the program at which the data isto be input, and R is the result expected to be produced by the program. A *test suite* is the set of all test that have been designed by a tester to test a given program.	2.5*2	5	
IX(a)	Basic COCOMO MODEL Gives an approximate estimate of project parameters $\text{Effort} = a_1 \times (KLOC)^{a_2}$ PM $\text{Tdev} = b_1 \times (Effort)^{b_2}$ Months			

	Where (a) KLOC is the estimated size of the software product expressed in Kilo Lines of Code, (b) a_1, a_2, b_1, b_2 are constants for each category of software products, (c) Tdev is the estimated time to develop the software, expressed in months, (d) Effort is the total effort required to develop the software product, expressed in person months (PMs). **Estimation of development effort** For the three classes of software products, the formulas for estimating the effort based on the code size are shown below: Organic : Effort = $2.4(KLOC)^{1.05}$ PM Semidetached : Effort = $3.0(KLOC)^{1.12}$ PM Embedded : Effort = $3.6(KLOC)^{1.20}$ PM **Estimation of development time** For the three classes of software products, the formulas for estimating the development time based on the effort are given below: Organic : Tdev = $2.5(Effort)^{0.38}$ Months Semidetached : Tdev = $2.5(Effort)^{0.35}$ Months Embedded : Tdev = $2.5(Effort)^{0.32}$ Months	2*4	8	
IX (b)	Risk management consists of 3 essential activities : risk identification, risk assessment and risk containment. Risk Identification Project manager needs to predict the risks in project as early as possible so that impact of the risk can be minimized. The risks can be categorized into 3: project risk, technical risk, business risk. Project Risks Project risks concern various forms budgetary, schedule, personnel, resource and customer related problems. An important project risk is schedule slippage. Technical Risks Technical risks concern potential design, implementation, interfacing, testing and maintenance problems. Business Risks Risks include risks of building an excellent product that no one wants, losing budgetary or personnel commitments.			15

		Risk Assessment The objective of risk assessment is to rank the risk in terms of their damage causing potential.Risks rated in 2 ways: The possibility of a risk coming true (r) The consequence of the problem associated with that risk (s) Based on these two factors, priority is calculated as P=r*s Risk Containment After all the identified risks of project are assessed, plans must be made to first contain the most damaging and most possible risks. There are three main strategies to plan for risk containment: Avoid the risk Transfer the risk Risk reduction	7	7	
X	(a)	 Explain each level in detail	10	10	15
X	(b)	Change management procedures •Change management procedures are concerned with analyzing the costs and benefits of proposed changes, approving those changes that are worthwhile and tracking which components of the system have been changed. •The change management process should come into effect when the software or associated documentation is base lined by the configuration management team. •The first stage in the change management process is to complete a change request form(CRF) describing the change required to			

	the system. •As well as recording the change required, the CRF records the recommendations regarding the change, the estimated cost of the change and the dates when the change was requested, approved, implemented and validated. •The CRF may also include a section where as analyst outlines how the change is to be implemented. •The CRF is usually defined during the CM planning process. •The engineer making the changes decides how to implement that change in these situation. •Once a change request form has been submitted, it should be registered in the configuration database. •The change request is then analysed to check that the change requested is necessary. •If the analysis discovers that a change requests is invalid, duplicated or has already been considered, the change is rejected. •For valid changes, the next stage of the process is change assessment and costing. •The impact of the change on the rest of the system must be checked. •This involves identifying all of the components affected by the change using information from the configuration database and the source code of the software.	5	5	
