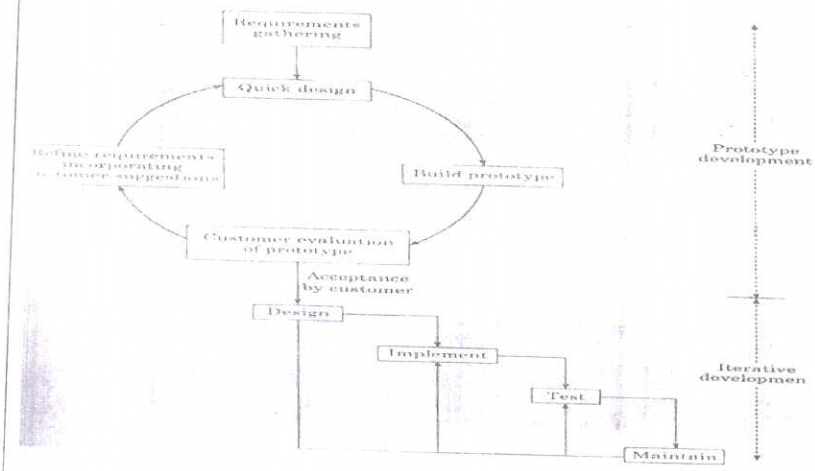
SCHEME OF VALUATION

(Scoring Indicators)

Revision : 2015		Course Code : 5132		
Course Title :	PROJECT MANAGEMENT AND SOFTWARE ENGINEERING			
Q No	Scoring Indicator	Split up score	Sub Total	Total
	PART A			
I.1	Software engineering is the application of engineering to the development of software in a systematic method.		2	
I.2	A DFD shows the flow of data through a system. It views a system as a function that transforms the inputs into desired outputs.		2	
I.3	Software testing is the systematic method to detect errors in the software. The goal is to uncover requirement, design, and coding errors in the programs.		2	
I.4	Risk Identification, Risk Analysis, Risk Planning, Risk Monitoring.	0.5 X 4	2	
I.5	Equivalence Class Partitioning, Boundary Value Analysis.		2	10
	PART B			
II. 1	The main aim of feasibility study activity is to determine whether it would be financially and technically feasible to develop the product. It involves analysis of the problem and collection of all relevant information relating to the product such as inputs, outputs, processing and constraints. These collected data are analysed to arrive at the following: 1.An abstract problem definition 2. Formulation of the different strategies for solving the problem. 3. Evaluation of the different solution strategies.		6	
II.2	The software architecture of a system is the structure of the system which comprise software elements, the externally visible properties of those elements, and the relationships among them. Some of the important uses of software architecture are: **1. Understanding and Communication-**			

	An architecture description communicates the architecture to its stakeholders, which include the users who will use the system, the clients, the developers etc. **2. Reuse** Architecture facilitates reuse of components among products that are similar and building product families such that common parts of these different but similar products can be reused. **3. Construction and evolution** The parts specified in architecture can be built independently by different teams or individuals and later combined to build the final system. **4. Analysis** From the architecture, it is possible to analyse or predict the properties of the system being built.		6	
II. 3	Pair programming is a coding process that has been proposed as a key technique in extreme programming (XP) methodology. In pair programming, code is not written by individual programmers but by a pair of programmers. In this process, one person will type the program while the other will actively participate and constantly review what is being typed. When errors are noticed, they are pointed out and corrected. The pair discusses the algorithms, data structures, or strategies to be used in the code to be written. Having two programmers result in better decisions being taken about the data structures, algorithms, interfaces, logic etc. Errors are also likely to be dealt with in a better manner.		6	
II.4	The Capability Maturity Model Integration(CMMI) is a process model that provides a clear definition of what an organization should do to promote behaviours that lead to improved performance. Levels: 1. Initial 2. Managed 3. Defined 4. Quantitatively managed 5. Optimizing	1 1 X 5	6	
II.5	Maintenance involves any one or more of the following three kinds of activities: 1. **Corrective Maintenance**-Involves correcting errors that were not discovered during the product development phase.	2		

	2. **Perfective Maintenance**-Involves improving the implementation of the system, and enhancing the functionalities of the system according to the customer's requirements. 3. **Adaptive Maintenance**-Required for porting the software to work in a new environment.	2 2	6	
II. 6	Coupling between modules is the strength of interconnections between modules or a measure of interdependence among modules. Two modules are considered independent if one can function completely without the presence of the other. If two modules are more dependent, they can't be easily solved or modified. Cohesion of a module represents how tightly bound the internal elements of the module are to one another. Cohesion of a module gives the designer an idea about whether the different elements of a module belong together in the same module. Usually, the greater the cohesion of each module in the system, the lower the coupling between the modules is.	3 3	6	
II. 7	**1. Expert Judgment Technique** In this technique, an expert makes an educated guess of the problem size after analysing the problem thoroughly. The expert estimates the cost of the different components and combines them to arrive at the overall estimate. This technique is subject to human errors and individual bias which can be eliminated by using a group of experts rather than a single one. **2. Delphi Cost Estimation** This technique is carried out by a team comprising of a group of experts and a coordinator. In this approach, the coordinator provides each estimator with a copy of SRS and a form for recording his cost estimate. The coordinator prepares a summary of the responses of all the estimators. The prepared summary is distributed to the estimators. Based on this summary, the estimators re-estimate. This process is iterated several rounds and finally the coordinator prepares the final estimate.	3 3	6	
	PART C			
III.a	In prototyping model, a working prototype of the system is built before starting the actual software development. A prototype is usually a very crude version of the actual system. The first phase of prototyping model is prototype development. This is followed by an iterative development cycle. In this model, prototyping starts with an initial requirements gathering phase. A quick design is carried out and a prototype is built. The developed prototype is submitted to the customer for his evaluation. Based on the customer feedback, the requirements are refined and the prototype is suitably modified.			

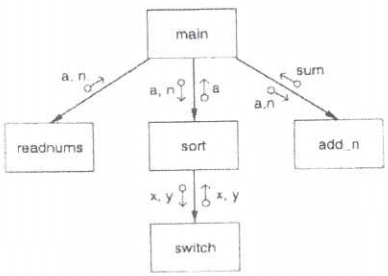
	This cycle continues until the customer approves the prototype. Once the customer approves the prototype, the actual system is developed using the iterative waterfall approach.  	5 3	 8	
III. b	The goal of the design phase is to transform the requirements specified in the SRS document into a structure that is suitable for implementation in some programming language. During the design phase, the software architecture is derived from the SRS document. There are two different design approaches **1. Traditional Design approach** This technique is based on the data flow-oriented design approach. It consists of two important activities. First a **structured analysis** of the requirements specification is carried out where the detailed structure of the problem is examined. This is followed by a **structured design** activity during which the results of structured analysis are transformed into the software design. During structured analysis, the functional requirements specified in the SRS document are decomposed into sub-functions and the data-flow among these sub-functions is analysed and represented diagrammatically in the form of DFDs. Structured design consists of two main activities: architectural design (also called high-level design) and detailed design (also called low-level design). High-level design involves decomposing the system into modules and representing the interfaces and the invocation relationships among the modules. During detailed design, internals of the individual modules are designed in greater detail. **2. Object-Oriented Design approach** In this technique, various objects that occur in the problem domain are first identified and the different relationships that exist among these objects are identified. The object structure is further refined to obtain the detailed design. The OOD approach	1 3 3	 7	

	has several benefits such as lower development time and effort, and better maintainability of the product.			
IV.a	The processes that deal with the technical and management issues of software development are collectively called the software process. The major components of a software process are: **Product engineering process** – The main objective is to produce the desired product. It consists of 1. **Development process** – specifies all the engineering activities that need to be performed. 2. **Project Management process** - specifies how to plan and control development activities so that cost, schedule, quality and other objectives are met. 3. **Software Configuration Management process** – deals with managing change, so that the integrity of the product is not violated despite changes. **Process Management process** – Deals with the understanding of the current process, analysing its properties, determining how to improve, and then affecting the improvement.	1 1 2 2 2 2	10	
IV.b	The goal of the implementation phase is to translate the design of the system into code in a given programming language. For a given design the aim is to implement the design in the best possible manner. Well written code can reduce the testing and maintenance effort. Because the testing and maintenance costs of software are much higher than the coding cost, the goal of coding should be to reduce the testing and maintenance effort. During coding the focus should be on developing programs that are easy to read and understand.	5	5	
V. a	The aim of design methodologies is to provide guidelines to aid the designer during the design process. The structured design methodology is used for developing function-oriented system designs. This methodology employs the structure chart notation for creating the design. Structure charts are used for representing designs graphically in function-oriented system design. The structure of a program is made up of the modules of that program together with the interconnections between modules. This program structure can be represented using structure charts. In a structure chart a module is represented by a box with the module name written in the box. An arrow from module A to module B represents that module A invokes module B. B is called the subordinate of A, and A is called the superordinate of B. The arrow is labelled by the parameters received by B as input and the parameters returned by B as output. The direction of flow of the input and output parameters are represented by small	6		

arrows. The parameters can be shown to be data or control.

Example: **Program**

```
main()
{
    int sum, n, N, a[MAX];
    readnums(a, &N); sort(a, N); scanf(&n);
    sum = add_n(a, n); printf(sum);
}

readnums(a, N)
int a[], *N;
{

}

sort(a, N)
int a[], N;
{
    if (a[i] > a[t]) switch(a[i], a[t]);
}

/* Add the first n numbers of a */
add_n(a, n)
int a[], n;
{

}
```



3

9

V. b The desirable characteristics of an SRS are

1. Correct – An SRS is correct if every requirement included in the SRS represents something required in the final system.

2. Complete – An SRS is complete if everything the software is supposed to do and the responses of the software to all classes of input data are specified in the SRS.

3. Unambiguous – An SRS is unambiguous if and only if every requirement stated has one and only one interpretation.

4. Verifiable – An SRS is verifiable if and only if every stated requirement is verifiable. A requirement is verifiable if there exists some cost-effective process that can check whether the final software meets that requirement.

5. Consistent – An SRS is consistent if there is no requirement that conflicts with another.

6. Ranked for importance and/or stability – An SRS is ranked for importance and/or stability if for each requirement the importance and stability of the requirement are indicated.

Any 4

1.5 X 4

6

VI. a	The basic purpose of requirement analysis is to obtain a thorough understanding of what the software needs to provide. It starts with a 'problem statement', which describes the system behaviour, constraints on the system, its inputs and outputs. During analysis, the analysts will have a series of meeting with the clients and end users. In the early meetings, the clients and end users will explain to the analyst about their work, their environment and their needs. Any documents describing the work or organisation along with outputs of the existing methods of performing the tasks will be given to the analyst. Here analyst is basically the listener absorbing the information provided. Once he understands the system to some extent, he uses the next few meeting to seek clarifications of the parts he does not understand. He may document the information or build some models. In the final few meetings, the analyst explains the client what he understands the system should do and verifies whether the proposed system meets the objectives of the client.		8	
VI. b	Planning is the most important project management activity. Planning has two basic objectives. 1. To establish reasonable cost, schedule and quality goals for the project. 2. To draw out a plan to deliver the project goals. A project succeeds if it meets its cost, schedule and quality goals. The inputs to the planning activity are the requirements specification and may be the architecture description. There are generally two main outputs of the planning activity. **1. Project Management Plan-**The overall project management plan document establishes the project goals on the cost, schedule and quality and defines the plans for managing risk, monitoring the project etc. 2. **Detailed Plan or Detailed Project Schedule-** Specifies the tasks that need to be performed to meet the goals, the resources who will perform them, and their schedule.	3 2 2	7	
VII. a	White-box testing is concerned with testing the implementation of the program. The main approach used in structural testing is control-flow based testing. **Control-flow based Criteria** This criteria makes use of the control flow graph of a program and coverage of various aspects of the graph are specified as criteria. Let the control flow graph of a program P be G. A node in this graph represents a block of statements that is always executed together. An edge (i, j) represents a possible transfer of control after executing the last statement of the block represented by node i to the first statement of the block represented by node j. A node corresponding to a block whose first	4		

statement is the start statement of P is called the start node of G, and a node corresponding to a block whose last statement is an exit statement is called an exit node. A path is a finite sequence of nodes (n1, n2, ..., nk), k > 1, such that there is an edge (ni, ni+1) for all nodes ni in the sequence (except the last node nk). A complete path is a path whose first node is the start node and the last node is an exit node .

Statement Coverage Criteria

This criteria requires that paths executed during testing include all the nodes in the graph. This criterion is not very strong and can leave errors undetected. For example, if there is an if statement in the program without having an else clause, the statement coverage criterion for this statement will be satisfied by a test case that evaluates the condition to true. No test case is needed that ensures that the condition in the if statement evaluates to false. Consider the program

```
int abs (X)
int X;
{
        if ( X > =0) X = 0-X;       // Error
        return (X);        }
```

2

Suppose we execute the function with the test case {x=0}. The statement coverage criterion will be satisfied by testing with this set, but the error will not be revealed.

Branch Coverage Criteria

This criteria each decision in the program be evaluated to true and false values at least once during testing. The 100% branch coverage criterion is also called the all-edges criterion. A set of test cases satisfying this criterion will detect the error in the above program.

The trouble with branch coverage comes if a decision has many conditions in it. For example, consider the following function that checks the validity of a data item. The data item is valid if it lies between 0 and 100.

2

```
int check (x)
int x;
{
if ( ( x >=0) && (x <=200))
```

	check=True;
	else
	check=False:
	}
	The module is incorrect, as it is checking for x <=200 instead of 100. Suppose the module is tested with the following set of test cases: { x = 5, x = -5 }. The branch coverage criterion will be satisfied for this module by this set. However, the error will not be revealed.
	Path Coverage Criteria
	This criteria requires that all possible paths in the control flow graph be executed during testing. The difficulty with this criterion is that programs that contain loops can have an infinite number of possible paths. Even then, path coverage criteria is the strongest among the three.
		2	10	

VII b	The goal of structured programming is to ensure that the static structure and the dynamic structures are the same. That is, the objective of structured programming is to write programs so that the sequence of statements executed during the execution of a program is the same as the sequence of statements in the text of that program. In structured programming, a statement is not a simple assignment statement, it is a structured statement. The key property of a structured statement is that it has a single-entry and a single-exit. That is, during execution, the execution of the statement starts from one defined point and the execution terminates at one defined point. The most commonly used single-entry and single-exit statements are ***Selection:*** if B then S1 else S2 if B then S1 ***Iteration:*** While B do S repeat S until B ***Sequencing:*** S1; S2; S3;.....		5	
VIII a	Code inspection can be viewed as "static testing" in which defects are detected in the code not by executing the code but through a manual process. Code inspection is a review of code by a group of peers following a clearly defined process. Inspections are performed by a team of reviewers including the author, with one of them being the moderator. The moderator has the overall responsibility to ensure that the review is done in a proper manner and all steps in the review process are followed. The different stages in this process are: planning, self-review, group review meeting, and	2		

rework and follow-up. These stages are generally executed in a linear order. 2

1. Planning

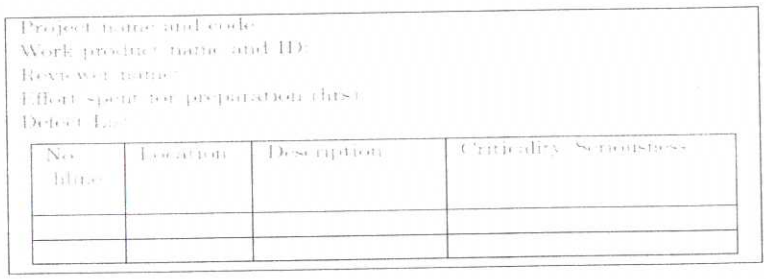
The objective of the planning phase is to prepare for inspection. An inspection team is formed, which should include the programmer whose code is to be reviewed. The team should consist of at least three people. A moderator is appointed.

The author of the code ensures that the code is ready for inspection and the entry criteria are satisfied. The moderator checks that the entry criteria are satisfied by the code. A package is prepared and distributed to the inspection team. The package typically consists of the code to be inspected, the specifications for which the code was developed, and the checklist that should be used for inspection.

The package for review is given to the reviewers. The moderator may arrange an opening meeting, if needed, in which the author may provide a brief overview of the product and any special areas that need to be looked at carefully. 2

2. Self-Review

In this phase, each reviewer does a self-review of the code. During the self-review, a reviewer goes through the entire code and logs all the potential defects he or she finds in the self-preparation log. An example form for self-preparation log is as shown.


Project name and code:
Work product name and ID:
Reviewer name:
Effort spent for preparation (hrs):
Defect List:

No.	Location	Description	Criticality/ Seriousness

2

3. Group Review Meeting

The basic purpose of the group review meeting is to come up with the final defect list, based on the initial list of defects reported by the reviewers and the new ones found during the discussion in the meeting. The main outputs of this phase are the defect log and the defect summary report.

The moderator first checks to see if all the reviewers are fully prepared. If everyone is ready, the group review meeting is held. A team member (called reader) goes over the code line by line, and interprets each line to the team. At any line, if any reviewer finds any issue, he raises it. There could be a discussion on the issue raised. The author accepts the issue as a defect or clarifies why it is not a defect. After discussion an agreement is reached and one member of the review team (called the scribe) records the identified defects in the defect log. At the end of the meeting, the scribe reads out the defects

	recorded in the defect log for a final review by the team members. The final defect log is the official record of the defects identified in the inspection and may also be used to track the defects to closure.		8	
VIII b	Testing is a quality control activity which focuses on identifying defects. Testing process consists of three high-level tasks. **1.Test Plan--** In a project testing commences with a test plan and terminates with successful execution of acceptance testing. A test plan is a general document for the entire project that defines the scope, approach to be taken, the schedule of testing as well as identifies the test items for testing and the personnel responsible for the different activities of testing. Inputs for test plan are 1.Project Plan 2.Requirements Document 3.Architecture or Design Document A Test Plan should contain the following 1.Test unit specification 2.Features to be tested 3.Approach for testing 4.Test deliverables 5.Schedule and task allocation **2.Test Case Design** Test case design has to be done separately for each unit. Based on the approach in the test plan the test cases are designed and specified for testing the unit. Test case specification gives, for each unit to be tested, all test cases, inputs to be used in the test cases, conditions being tested by the test case, and outputs expected for those test cases. **3.Test Case Execution** Executing the test cases may require construction of driver modules or stubs. It	1 2 2 2		

Requirement Number	Condition to be tested	Test data and settings	Expected output

	may also require modules to set up the environment as stated in the test plan and test case specifications. If test frameworks are being used, then the setting of the environment as well as inputs for a test case is already done in the test scripts, and execution is straightforward. During test case execution, defects are found. These defects are then fixed and testing is done again to verify the fix. To facilitate reporting and tracking of defects found during testing defects found are often logged.		7	
IX. a	A software project management framework is a combination of of processes, tasks, and tools used to transition a project from start to finish. The main parts of project management framework are lifecycle, control cycle and tools and templates. **Lifecycle** The lifecycle of the framework explains the stages involved in the project and what needs to happen at each stage. It allows the management team to make adjustments and customize the stages based on the size and scope of the project. The main stages of project management lifecycle are: **1. Initiation**: - Identify the objective of the project, determine whether the project is feasible, and identify the major deliverables for the project. **2. Planning**: - Break down the larger project into smaller tasks, build the project team and prepare a schedule for the completion of assignments. **3. Execution, Monitoring and Control**: - Keep work on track, organize team members, manage timelines and make sure that the work is done according to the original plan. **4. Closure**: - provide final deliverables, release project resources and determine the success of the project.	3 6	9	
IX. b	**Activities of Software Quality Management:** **Quality Planning** – Select applicable procedures and standards for a particular project and modify as required to develop a quality plan. It involves the preparation of a quality management plan that describes the processes and metrics that will be used. **Quality Assurance** – QA aims at developing Organizational procedures and standards for quality at Organizational level. Quality assurance is the process or set of processes used to measure and assure the quality of a product. **Quality Control** - Ensure that best practices and standards are followed by the software development team to produce quality products. Quality control is the process of ensuring products and services to meet consumer expectations. **Continual improvement process**: It is also often called a continuous improvement	1.5 X 4	6	

	process (abbreviated as CIP or CI), is an ongoing effort to improve products, services, or processes. These efforts can seek "incremental" improvement over time or "breakthrough" improvement all at once.			
X. a	Resources constitute all elements used to develop a software product. They include human resources, productive tools, and software libraries. The main feature of resources is that they are limited in quantity and allocating extra resources increases development cost. **Resource management activities** **1. Planning activities** -Estimate requirement of resource -Request for requisite resources -chedule resource utilization -Resource levelling **2. Utilization activities** -Allocation of resources to various activities -Ensure the activities are performed **3. Deallocation and release activities** - Performance appraisals for human resources - Reconciliation for monetary resources – planned vs. actual utilization - Document lessons learned in planning and utilization of resources - Release resources	3 6	9	
X b	 The change management process is initiated when a customer completes and submits a change request describing the change required to the system. This could be a bug report, where the symptoms of the bug are described, or a request for additional functionality to be added to the system. Change requests may be submitted using a change request form (CRF). As the change request is processed, information is added to the CRF to record decisions made at each stage of the process. At any time, it therefore represents a snapshot of the state of the change request. As well as recording the change required, the CRF	2 4		

	records the recommendations regarding the change; the estimated costs of the change; and the dates when the change was requested, approved, implemented, and validated. The CRF may also include a section where a developer outlines how the change may be implemented. **Factors considered while approving/denying a change request** 1. What will happen if the change is not implemented? 2. The benefits of the change. 3. The number of users affected by the change. 4. The costs of making the change - If making the change affects many system components. 5. The product release cycle- If a new version of the software has just been released to customers, it may make sense to delay the implementation of the change until the next planned release.		6	