5 Years Integrated M.C.A.

DSE7 Introduction to Software Engineering
(060060509)

5th Semester

EFFECTIVE FROM JUNE - 2017

VERSION-1.0
Course Code: 060060509  
Course Title: DSE7 Introduction to Software Engineering  
Course Credits: 04  
[Lectures: 04, Tutorial: 00, Practical: 04]  
Pre-requisites:  
Object Oriented Programming and Information Systems concepts  
Objectives:  
To introduce the concept of software engineering, development models and object oriented paradigm for efficient design and development of reliable software.

1 Introduction to Software Engineering Concepts [15%]
   1.1. Types of Systems  
   1.2. Need for Systems Analysis and Design  
   1.3. Role of System Analyst: System Analyst as Consultant and Supportive Expert  
   1.4. Software complexity and characteristics of software  
   1.5. General principles of software engineering  
   1.6. Traditional vs. object oriented software engineering

2 Models for Software Development and Testing [15%]
   2.1. Meaning of process, process framework and umbrella activities  
   2.2. Conventional life cycle models: Overview  
   2.3. Iterative Enhancement model and Spiral model  
   2.4. Outline of Object oriented software life cycle models  
   2.5. Agile model: Extreme programming  
   2.6. V-model for software engineering and testing  
   2.7. Verification and validation: concepts, techniques and checklist

3 Software Requirement [20%]
   3.1. Software requirement and requirement Engineering  
   3.2. Overview of requirements elicitation techniques  
   3.3. Initial requirement document  
   3.4. Characteristics of good requirement  
   3.5. Software requirement specification (SRS) document

4 Analysis Modeling [20%]
   4.1. Structured vs. object oriented analysis  
   4.2. Data dictionary and repository: need of data dictionary, creating and using data dictionary  
   4.3. Decision analysis techniques: Structured English, decision tables and decision trees  
   4.4. Class Modeling: class diagrams: link and association concepts and generalization and inheritance concepts  
   4.5. Interaction modeling: Use case approach: Use case and actors, creating use case diagram for requirement, use case description, Activity diagram and sequence diagram

5 System Design [20%]
   5.1. Concept of Software Design  
   5.2. Cohesion and coupling with its types  
   5.3. Object oriented Design Concepts: Types of Design Classes  
   5.4. Types of Design Elements, Data Design  
   5.5. Architectural Design Styles, Component-Level Design and Basic Principles  
   5.6. Interface design and golden rules of designing
5.7. Deployment design


6.1. Emergent Requirements
6.2. Open Source Development
6.3. Importing code into project

The tool/platform/technology as an exposure to the concepts' implementation/demonstration, shall be determined by the course teacher(s) with due approval of Director and/or IQAC.

Course Outcomes: Upon completion of the course, students shall be able to

CO1: Comprehend the basic concepts and importance of traditional and object-oriented software engineering.
CO2: Compare and contrast conventional and object-oriented software process models with its applicability.
CO3: Recognize the prominence of V-model for development of quality software.
CO4: Identify needs and significance of requirements engineering using analysis modelling, data and object oriented modelling concepts.
CO5: Identify and design UML diagrams for a given system.
CO6: Classify and construct architectural, component level and user interface design for a given system.
CO7: Comprehend the latest software engineering development trends

Course Objective and Course Outcomes Mapping:

To Provide fundamentals principle of software engineering: CO1, CO2, CO3
To use data modeling tools and technique: CO4, CO5, CO6

Course Units and Course Outcomes Mapping:

<table>
<thead>
<tr>
<th>Unit No.</th>
<th>Unit</th>
<th>CO1</th>
<th>CO2</th>
<th>CO3</th>
<th>CO4</th>
<th>CO5</th>
<th>CO6</th>
<th>CO7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction to software engineering concepts</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Models for software development and testing</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Software requirement</td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Analysis modelling</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>System design</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Current trends in software</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
</tr>
</tbody>
</table>

Laboratory

- A course teacher shall prepare a fresh Practical List for each academic year with no repeated problem definitions from previous two consecutive years.
- The Practical List shall consists of "Required number of problems" for journal certification as well as "Practice problems" of varying nature from each unit as per its weightage and criticality.
- Laboratory Supervisor or Course Teacher shall sign in the journal only if he/she is
satisfied by the work of student.

- Journal shall be verified by the Laboratory Supervisor as well as by the Course Teacher at least thrice in a semester at appropriate interval upon the discretion of the Course Teacher.
- Journal must not be certified if required number of problems are not included and not written clearly.
- After due approval, the Practical List shall be kept by concern Course Teacher on website before the commencement of the semester.
- Problem list shall contain practical problems from each of the units as follows:

<table>
<thead>
<tr>
<th>Unit No.</th>
<th>Required no. of problems to get the journal certified</th>
<th>Covering Unit / Sub-Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1.2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2.1, 2.5, 2.6, 2.7</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3.1, 3.5</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>4.2, 4.3, 4.4, 4.5</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>5.3, 5.2</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>6.2</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>13</strong></td>
<td></td>
</tr>
</tbody>
</table>

**Hands-on Activity**

- Hands-on sessions shall be conducted on following topics:
  1. Software requirement specification

**Modes of Transaction (Delivery):**

- Lecture method shall be used for all units. For all units lecture delivery shall be supplemented with audio-visual aids.
- Self-Study of following part of the syllabus shall be done by the students:
  1.6 Traditional vs. object-oriented software engineering

**Activities/Practicum:**

The following activities shall be carried out by the students.

- Prepare SRS document for given definition by teacher

The following activities shall be carried out by the course teacher.

- Demonstrate the different use case with examples

**Text Books:**

1. Roger S. Pressman, Software Engineering, McGraw-Hill
2. Yogesh Singh and Ruchika Malhotra, Object-Oriented Software Engineering, PHI

**Reference Books:**

1. Kendall & Kendall, System Analysis and Design, PHI
2. Michael R. Blaha and James R. Rumbaugh, Object-Oriented Modeling and Design with UML, Pearson
Concept Map:

It is a hierarchical / tree based representation of all topics covered under the course. This gives direct / indirect relationship / association among topics as well as subtopics.

Unit 1: Introduction to Software Engineering Concepts

Unit 2: Models for Software Development and Testing

Unit 3: Software Requirements
Unit 4: Analysis Modelling

Analysis Modelling includes:

- Structured vs. object oriented analysis includes:
  - Data dictionary and repository
    - need of data dictionary
    - creating and using data dictionary
  - Decision analysis techniques
    - Structured English
    - decision tables
    - decision trees

- Interaction modeling includes:
  - Use case approach
  - Activity diagram
  - Sequence diagram

- Class Modeling
  - Class diagrams

Unit 5:
Unit 6: Current Trends in Software

Assessment:

- The weightage of Continuous Internal Evaluation (CIE) and University examination shall be as per the University regulations.
- The course teacher is free to decide the structure of CIE:
  i. Assessment parameters like Open Book test, Quizzes, Unit Tests, Assignments, Internal, Self-Creation etc.
  ii. Weightage and frequency of each parameter.
- After assessment parameters are approved by Director and/or IQAC, it shall be informed to the students by publishing over web before commencement of the semester.
- The assessment policy document should be uploaded on the web before the commencement of the semester.
- Syllabus for each CIE parameter shall be covered by the date of the corresponding test.
- No make-up work shall be conducted unless approval from Director.

UFM:

- Any ascertained fact of breaking institute policy shall be associated with one or all of
the following: (i) zero marks for the work; (ii) report to the programme coordinator; (iii) report to the Director.

Attendance:

- Attendance means being present for the entire class session. Those arriving significantly late or leaving significantly early without prior permission shall be counted as ABSENT for the entire class session.
- Concern teacher must clearly state his/her attendance policies at the first class meeting.