### SCHEME OF TEACHING AND EXAMINATION FOR M.TECH. SOFTWARE ENGINEERING

#### II Semester

<table>
<thead>
<tr>
<th>Subject Code</th>
<th>Name of the Subject</th>
<th>Hours per Week</th>
<th>Duration of Exam in Hours</th>
<th>Marks for Total Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>10SSE21</td>
<td>Topics in Software Engineering - II</td>
<td>04</td>
<td>02</td>
<td>03</td>
</tr>
<tr>
<td>10SSE22</td>
<td>Metrics and Models in Software Quality Engineering</td>
<td>04</td>
<td>02</td>
<td>03</td>
</tr>
<tr>
<td>10SSE23</td>
<td>Topics in Software Testing</td>
<td>04</td>
<td>--</td>
<td>03</td>
</tr>
<tr>
<td>10SSE24</td>
<td>Fault-Tolerant Systems</td>
<td>04</td>
<td>02</td>
<td>03</td>
</tr>
<tr>
<td>10SSE25x</td>
<td>Elective – II</td>
<td>04</td>
<td>02</td>
<td>03</td>
</tr>
<tr>
<td>10SSE26</td>
<td>*Project Phase-I(6 Week Duration)</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>10SSE27</td>
<td>Seminar</td>
<td>--</td>
<td>03</td>
<td>--</td>
</tr>
</tbody>
</table>

**Total** | 20 | 04 | 09 | 15 | 300 500 800

**Elective – II**

10SSE251 – Mobile Computing  
10SSE252 – Distributed Systems  
10SSE253 – Web Engineering

*Between the II Semester and III Semester. After availing a vacation of 2 weeks.*

Note: The Internal Assessment marks for the core subjects with 2 hours of practical will have 30 marks for theory and 20 marks for practical work.

---

### II SEMESTER

**TOPICS IN SOFTWARE ENGINEERING – II**

- **Subject Code:** 10SSE21  
  - **I.A. Marks:** 50
- **Hours/Week:** 04  
  - **Exam Hours:** 03
- **Total Hours:** 52  
  - **Exam Marks:** 100

1. **Agile development:** What is agile? Agility and cost of change; What is an agile process? Extreme programming; Other agile process models.
2. **Web Application Design:** Web application design quality; Design quality and design pyramid; Interface design; Aesthetic design; Content design; Architecture design; Navigation design; Component-level design; Object-oriented hypermedia design method.
3. **Formal Modeling and verification:** The cleanroom strategy; Functional specification; Cleanroom design; Cleanroom testing; Formal methods: Concepts; Applying mathematical notation for formal specification; Formal specification languages.
4. **Software Project Management:** The management spectrum; The management of people, product, process and project; The W5HH Principle; Critical practices.
5. **Estimation for Software Projects:** Software project estimation; Decomposition techniques, Examples; Empirical estimation models; Estimation for Object-Oriented projects; Specialized estimation techniques; The make / buy decision.
6. **Software Project Scheduling:** Basic concepts and principles of project scheduling; Defining task set and task network; Scheduling; Earned value analysis.
7. **Risk Management:** Reactive versus proactive strategies; Software risks; risk identification; Risk projection; Risk refinement; Risk mitigation, monitoring and management; The RMMM plan.
8. **Maintenance and Reengineering:** Software maintenance; Software supportability; Reengineering; Business process reengineering; Software reengineering; Reverse engineering; Restructuring; Forward engineering; The economics of reengineering.
9. **Software Process Improvement (SPI):** Approaches to SPI; Maturity models; The SPI process; The CMMI; The People CMM; Other SPI frameworks: SPICE, Bootstrap, PSP and TSP, ISO; SPI return on investment.
10. **Software Configuration Management (SCM):** Basic concepts; SCM repository; The SCM process; Configuration management for web applications; SCM standards.
11. **Product Metrics:** A framework for product metrics; Metrics for requirements model, design model, source code, testing and maintenance; Design metrics for web applications.
12. **Process and Project Metrics:** Basic concepts; Software measurement; Metrics for software quality; Integrating metrics within the software process; Metrics for small organizations; Establishing a software metrics program.

**TEXT BOOKS:**

**REFERENCE BOOKS:**

---

**METRICS AND MODELS IN SOFTWARE QUALITY ENGINEERING**

<table>
<thead>
<tr>
<th>Subject Code: 10SSE22</th>
<th>L.A. Marks : 50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hours/Week : 04</td>
<td>Exam Hours: 03</td>
</tr>
<tr>
<td>Total Hours : 52</td>
<td>Exam Marks: 100</td>
</tr>
</tbody>
</table>

1. **Introduction:** Quality: Popular views; Quality: Professional views; Software quality; Total quality management.
2. **Overview of Software Quality Metrics:** Product quality metrics; In-process quality metrics; Metrics for software maintenance; Examples of metrics programs; Collecting software engineering data.
3. **Applying the 7 Basic Quality Tools in Software Development:** Ishikawa’s seven basic tools; Checklist; Pareto diagram; Histogram; Run charts; Scatter diagram; Control chart; Cause-and-effect diagram; Relations diagram.
4. **Defect Removal Effectiveness:** Review; A closer look at defect removal effectiveness; Defect removal effectiveness and quality planning; Cost effectiveness of phase defect removal; Defect removal effectiveness and process maturity level.
5. **The Rayleigh Model:** Reliability models; The Rayleigh model; Basic assumptions; Reliability and predictive validity.
6. **Exponential Distribution and Reliability Growth Models:** The exponential model; Reliability growth models; Model assumptions; Criteria for model evaluation; Modeling process; Test compression factor; Estimating the distribution of total defects over time.
7. **Quality Management Models:** The Rayleigh model framework; The code integration pattern; The PTR submodel; The PTR arrival / backlog projection model; Reliability growth models; Criteria for model evaluation; In-process metrics and reports; Orthogonal defect classification.
8. **In-Process Metrics for Software Testing:** In-process metrics for software testing; In-process metrics and quality management; Possible metrics for acceptance testing to evaluate vendor-developed software; When is the product good enough to ship?
9. **Metrics and Lessons Learned for Object-Oriented Projects:** Object-oriented concepts and constructs; Design and complexity metrics; Productivity metrics; Quality and quality management metrics; Lessons learned for OO projects.
10. **Availability Metrics:** Definition and measurements of system availability; Reliability, availability, and defect rate; Collecting customer outage data for quality improvement; In-process metrics for outage and availability.
11. **Measuring and Analyzing Customer Satisfaction:** Customer satisfaction surveys; Analyzing satisfaction data; Satisfaction with Company; How good is good enough?
12. **Conducting In-Process Quality Assessments:** The preparation phase; The evaluation phase; The summarization phase; Recommendations and risk mitigation.

**Laboratory Work:**
1. Design, develop and execute a program in a language of your choice to determine phase-wise effectiveness metrics from the matrix of defect data organized as Defect Origin by Where Found. Experiment with different sets of simulated data or data available from public domains and discuss the impact early defect removal efforts on software quality.
2. Design, develop and execute a program in a language of your choice to implement the Rayleigh model, plot the graph, and to estimate the latent error rate using the model. Experiment with different sets of simulated data or data available from public domains and discuss the results.

3. Design, develop and execute a program in a language of your choice to implement the Jelinski-Moranda model, plot the graph, and to estimate the software reliability using the model. Experiment with different sets of simulated data or data available from public domains and discuss the results.

4. Design, develop and execute a program in a language of your choice to implement the Musa-Okumoto model, plot the graph, and to estimate the software reliability using the model. Experiment with different sets of simulated data or data available from public domains and discuss the results.

5. Design, develop and execute a program in a language of your choice to implement the Delayed S model, plot the graph, and to estimate the software reliability using the model. Experiment with different sets of simulated data or data available from public domains and discuss the results.

TEXT BOOKS:

REFERENCE BOOKS:

TOPICS IN SOFTWARE TESTING

<table>
<thead>
<tr>
<th>Subject Code: 10SSE23</th>
<th>L.A. Marks : 50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hours/Week : 04</td>
<td>Exam Hours: 03</td>
</tr>
<tr>
<td>Total Hours : 52</td>
<td>Exam Marks: 100</td>
</tr>
</tbody>
</table>


2. **Decision Table-Based Testing**: Decision tables, Test cases for the triangle problem, Test cases for the NextDate function, Test cases for the commission problem, Guidelines and observations.

3. **Data Flow Testing**: Definition–Use testing, Slice-based testing, Guidelines and observations.

4. **Levels of Testing**: Traditional view of testing levels, Alternative life-cycle models, The SATM system, Separating integration and system testing.

5. **Integration Testing**: A closer look at the SATM system, Decomposition-based, call graph-based, Path-based integrations, Case study.

6. **System Testing**: Threads, Basic concepts for requirements specification, Finding threads, Structural strategies and functional strategies for thread testing, SATM test threads, System testing guidelines, ASF (Atomic System Functions) testing example.


8. **Issues in Object-Oriented Testing**: Units for object-oriented testing, Implications of composition and encapsulation, inheritance, and polymorphism, Levels of object-oriented testing, GUI testing, Dataflow testing for object-oriented software, Examples.

9. **Class Testing**: Methods as units, Classes as units.

10. **Object-Oriented Integration Testing**: UML support for integration testing, MM-paths for object-oriented software, A framework for object-oriented dataflow integration testing.

11. **GUI Testing**: The currency conversion program, Unit testing, Integration Testing and System testing for the currency conversion program.

12. **Object-Oriented System Testing**: Currency converter UML description, UML-based system testing, Statechart-based system testing.

13. **Exploratory Testing**: The context-driven school, Exploring exploratory testing, Exploring a familiar example, Exploratory and context-driven testing observations.

14. **Model-Based Testing**: Testing based on models, Appropriate models, Use case-based testing, Commercial tool support for model-based testing.
14. **Test-Driven Development:** Test-then-code cycles, Automated test execution, Java and JUnit example, Remaining questions, Pros, cons, and open questions of TDD, Retrospective on MDD versus TDD.

15. **A Closer Look at All Pairs Testing:** The all-pairs technique, A closer look at NIST study, Appropriate applications for all pairs testing, Recommendations for all pairs testing.

16. **Software Testing Excellence:** Craftsmanship, Best practice of software testing, Top 10 best practices for software testing excellence, Mapping best practices to diverse projects.

**Laboratory Work:**

1. Design, develop, code and run the program in any suitable language to solve the commission problem. Analyze it from the perspective of dataflow testing, derive at least 10 different test cases, execute these test cases and discuss the test results.

2. Design, develop, code and run the program in any suitable language to solve the NextDate problem. Analyze it from the perspective of decision table-based testing, derive at least 10 different test cases, execute these test cases and discuss the test results.

3. Design, develop, code and run the program in any suitable object-oriented language to solve the calendar problem. Analyze it from the perspective of OO testing, derive test cases to test the method that increments the date and the method that increments the month., execute these test cases and discuss the test results.

4. Design, develop, code and run the program in any suitable object-oriented language to solve the currency converter problem. Analyze it from the perspective of use case-based system testing, derive appropriate system test cases., execute these test cases and discuss the test results.

**TEXT BOOKS:**


**REFERENCE BOOKS:**


**FAULT-TOLERANT SYSTEMS**

<table>
<thead>
<tr>
<th>Subject Code: 10SSE24</th>
<th>L.A. Marks : 50</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hours/Week : 04</td>
<td>Exam Hours: 03</td>
</tr>
<tr>
<td>Total Hours : 52</td>
<td>Exam Marks: 100</td>
</tr>
</tbody>
</table>

**1. Introduction:** Fault classification; Types of Redundancy; Basic measures of FaultTolerance.

**2. Hardware Fault Tolerance:** The rate of hardware failures; Failure rate, Reliability, and Mean Time To Failure; Canonical and Resilient Structures; Other Reliability Evaluation Techniques; Fault-Tolerance – Processor-Level techniques; Byzantine Failures.

**3. Information Redundancy:** Coding; Resilient Disk Systems; Data Replication; Algorithm-Based Fault Tolerance.

**4. Fault-Tolerant Networks:** Measures of Resilience; Common Network Topologies and Their Resilience; Fault-Tolerant Routing.

**5. Software Fault Tolerance:** Acceptance Tests; Single-Version Fault Tolerance; N-Version Programming; Recovery Block Approach; Preconditions, Postconditions, and Assertions; Exception Handling; Software Reliability Models; Fault-Tolerant Remote Procedure Calls.

**6. Checkpointing:** What is Checkpointing? Checkpoint Level; Optimal Checkpointing – An Analytical Model; Cache-Aided Rollback Error Recovery; Checkpointing in Distributed Systems; Checkpointing in Shared Memory Systems; Checkpointing in Real-Time Systems; Other uses of Checkpointing.

**7. Defect Tolerance in VLSI Circuits:** Manufacturing Defects and Circuit Faults; Probability of Failure and Critical Areas; Basic Yield Models; Yield Enhancement through Redundancy.

**8. Fault Detection in Cryptographic Systems:** Overview of Ciphers; Security Attacks through Fault Injection;
Countermeasures.

9. Case Studies: Non-Stop Systems; Stratus Systems; Cassini Command and Data Sub-System; IBM G5; IBM Syplex; Itanium.

TEXT BOOKS:

REFERENCE BOOKS:

MOBILE COMPUTING

Subject Code: 10SSE251
L.A. Marks : 50
Hours/Week : 04
Exam Hours: 03
Total Hours : 52
Exam Marks: 100

1. Overview: Mobile communications; Mobile computing; Mobile computing architecture; Mobile devices; Mobile system networks; Data dissemination; Mobility management; Mobile phones, Digital Music Players, Handheld Pocket Computers, Handheld Devices, Operating Systems, Smart Systems, Limitations of Mobile Devices, Automotive Systems.


TEXT BOOK:

REFERENCES:
DISTRIBUTED SYSTEMS

Subject Code: 10SSE252  I.A. Marks : 50
Hours/Week : 04  Exam Hours: 03
Total Hours : 52  Exam Marks: 100

2. Networking and Internetworking: Types of Networks, Networks principles, Internet protocols, Network case studies (Ethernet, wireless LAN and ATM).
3. Interprocess Communication: Introduction, The API for the Internet protocols, External data representation and marshalling, Client-Server communication, Group communication, Case study: Interprocess Communication in UNIX.
4. Distributed Objects and Remote Invocation: Communication between distributed objects, Remote procedure call, events and notifications, JAVA RMI case study.
5. Operating System Support and Security: The Operating system layer, protection, processes and threads, communication and invocation, operating system architecture, overview of security techniques, cryptographic algorithms, digital signatures, cryptography pragmatics, case studies: Needham-Schroeder, Kerberos, SSL and Millicent.
6. Distributed File Systems: File service architecture, Sun Network file system, Andrew file system, Recent advances.
8. Distributed Transactions: Flat and nested distributed transactions, atomic commit protocols, concurrency control in distributed transactions, distributed deadlocks, transaction recovery.
9. Distributed Shared Memory: Design and Implementation issues, sequential consistency and Ivy, Release consistency and Munin, other consistency models.
10. CASE Studies: COBRA, Mach.

TEXT BOOKS:

REFERENCE BOOKS:

WEB ENGINEERING

Subject Code: 10SSE253  I.A. Marks : 50
Hours/Week : 04  Exam Hours: 03
Total Hours : 52  Exam Marks: 100

1. Introduction: Motivation, Categories of web applications, Characteristics of web applications.
5. Technology-Aware Web Application Design: Introduction, Web design from an evolutionary perspective, Presentation design, Interaction design, Functional design, Outlook.
8. **Operation and Maintenance of Web Applications:** Introduction, Challenges following the launch of a web application, Content management, Usage analysis, Outlook.

9. **Web Project Management:** From software project management to web project management, Challenges in web project management, Managing web teams, Managing the development process of a web application, Outlook.


12. **Performance of Web Applications:** Introduction, What is performance? What characterizes performance of web applications, System definition and indicators, Characterizing the work load, Analytical techniques, Representing and interpreting results, Performance optimization methods, Outlook.


**Text Book:**

**Reference Books:**