Course Structure

Total Credit of the Course = 96

Semester-I		
Total Credit= 24		
Paper Code	**Title**	**Credit**
MTSE 101	Advanced Software Engineering#	4
MTSE 102	Software Architecture and Design	6
MTSE 103	Agile Software Development	6
MTSE 104	Software Testing and Analysis	4
MTSE 105	• Empirical Software Engineering • Object Oriented Software Engineering	4
Semester-II		
Total Credit= 24		
MTSE 201	Software Requirement Engineering	4
MTSE 202	Secure Software Engineering#	6
MTSE 203	Component Based System Design	6
MTSE 204	Object Oriented Software Engineering	6
MTSE 205	• Genetic Algorithm • Software Quality Management	2
Semester-III		
Total Credit= 24		
MTSE 301	Software Project Management#	4
MTSE 302	Software Reliability	6
MTSE 303	Software reuse & Re-engineering	6
MTSE 304	• Software Engineering Process, Methods & Too • Formal Methods in Software Engineering	4
MTSE 305	Minor Project/Industrial Tour	4
Semester-IV		
Total Credit= 24		
MTSE 401	Dissertation & Comprehensive Viva	24

SEMESTER -1

MTSE 101: Advanced Software Engineering

Introduction: Life cycle models, Requirement Analysis and specification, Formal requirements specification.

Fundamental issues in software design: Goodness of design, cohesions, coupling. Function-oriented design: structured analysis and design.

Overview of object –oriented concepts. Unified Modeling Language (UML). Unified design process. User interface design. Coding standards and guidelines. Code walkthrough and reviews.

Unit testing. Black box and white box testing. Integration and system testing. Software quality and reliability. SEI CMM and ISO 9001.

PSP and Six Sigma. Clean room technique. Software maintenance issues and techniques. Software reuse. Client-Server software development.

Reference:

1. Ian Sommeriele, “Software Engineering”, Addison Wesley.
2. C.Easteal and G.Davis, Software Engineering Analysis and Design, Tata McGraw Hill. 3. Pressman, Software Engineering –A Practitioner’s Approach.
3. Richard Fairley, Software Engineeering Concepts, Tata Mcgraw Hill.
4. Pankaj Jalote , An Integrated Approach to Software engineering, Narosa Publication.

MTSE-102: Software Architecture and Design

Software Design: Key design principles and heuristics and trade-offs between these. “Bad smells” and refactoring.

Design Patterns: history, principles and expectations. Ways of using patterns. Detailed study of a number of representative patterns.

Software Architecture: why is architecture important? Classical architectural styles such as pipe and filter, data abstraction or OO based, event-based, etc.

Frameworks: frameworks as reusable chunks of architecture, the framework lifecycle, development using frameworks, detailed study of some well-known frameworks (e.g. HotDraw).

Major approaches to automated evaluation and analysis: dynamic analysis (e,g, testing, debugging, model inference, and visualisation) and static analysis (e.g. call and control graph extractions, metrics calculation, dataflow analysis, type systems, model checking, symbolic execution), and their application and limitations. Construction of tools to support such analysis.

Reference:

1. Head First Design Patterns. O'Reilly, Freeman and Freeman.
2. Design Patterns. Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides, . Addison-Wesley, 1995
3. An Introduction to Software Architecture. David Garlan and Mary Shaw.
4. Refactoring: Improving the Design of Existing Code. Martin Fowler. Software Design: From Programming to Architecture. Eric Braude.
5. Object-oriented Design Heuristics. Arthur Riel.
6. Software Architecture - Foundations, Theory and Practice.Taylor, Medvidovic, Dashofy. Wiley 2009.

MTSE-103: Agile Software Development

Software is new product development – Iterative development – Risk-Driven and Client-Driven iterative planning – Time boxed iterative development – During the iteration, No changes from external stakeholders – Evolutionary and adaptive development - Evolutionary requirements analysis – Early "Top Ten" high-level requirements and skilful analysis – Evolutionary and adaptive planning – Incremental delivery – Evolutionary delivery – The most common mistake – Specific iterative and Evolutionary methods.

Agile development – Classification of methods – The agile manifesto and principles – Agile project management – Embrace communication and feedback – Simple practices and project tools – Empirical Vs defined and prescriptive process – Principle-based versus Rule-Based – Sustainable discipline: The human touch – Team as a complex adaptive system – Agile hype – Specific agile methods. The facts of change on software projects – Key motivations for iterative development – Meeting the requirements challenge iteratively – Problems with the waterfall. Research evidence – Early historical project evidence – Standards-Body evidence – Expert and thought leader evidence – A Business case for iterative development – The historical accident of waterfall validity.

Method overview – Lifecycle – Work products, Roles and Practices values – Common mistakes and misunderstandings – Sample projects – Process mixtures – Adoption strategies – Fact versus fantasy – Strengths versus "Other" history.

Agile – Motivation – Evidence – Scrum – Extreme Programming – Unified Process – Evo – Practice Tips.

Project management – Environment – Requirements – Test – The agile alliances – The manifesto – Supporting the values – Agile testing – Nine principles and six concrete practices for testing on agile teams.

References

1. Elisabeth Hendrickson, "Agile Testing" Quality Tree Software Inc 2008.
2. Craig Larman "Agile and Iterative Development – A Manager's Guide" Pearson Education – 2004.
3. Alistair "Agile Software Development series" Cockburn - 2001.

MTSE-104: Software Testing and Analysis

Views on quality-cost of quality-quality models-Statistics and measurements-Statistics and measurements-Analysis of given source code using SQALE and Sonar models.

Quality framework characteristics – verification- Measuring test adequacy overview of black box testing techniques-decision tables-combinatorial testingclassification tree method- white box testing- Random and exploratory.

Introduction to Static analysis- Static analyzer for finding dynamic programming errors-dataflow testing – procedure to apply data flow testing- examplesperformance analysis and verification- Security analysis and verification – Software vulnerabilities and exploitation.

Applying the Design structure matrix to system decomposition and integration problems-achieving Agility through Architecture visibility-Recovering and verifying architecture through design structure matrices.

Project quality management- Essential Testing-Test driven development – guidance for software verification and validation plans-Master test planning.

References

1. Kshirasagar Naik and Priyadarshi Tripathy "Software testing and Quality Assurance: theory and practice, edited by copyright John wiley & sons Inc, 2008.
2. Daniel Galin "Software Quality Assurance from Theory to Implementation", Pearson Education Ltd., 2004.
3. Marc-Alexis Côté M. Ing , Witold Suryn Elli Georgiadou "Quality models to engineeri quality requirements" published in journal of object technology, chair of Software engineering, Vol.2 , No. 5 Sep. – October 2003. Online at http://www.jot.sm.
4. Tyson R. Browning, "Applying the design structure matrix to system decomposition and integration problems", A review and new directions IEEE transactions on Engineering management, Vol. 48, No.3, August 2001.
5. Neeraj sangal and Frank Waldman "Dependency models to manage software Architecture: journal of Defense software engineering, November 2005. Online at www.stsc.hill.af.mil.

SEMESTER -2

MTSE-201: Software Requirement Engineering

Framework for Requirements Engineering, Requirements Engineering activities – Elicitation, Analysis, Validation, Documentation, Management, Rationale for Requirements Engineering and the problems with requirements, The importance of requirements planning and estimating.

Requirements Elicitation- Knowledge types – tacit and non-tacit, Elicitation techniques, Interviews, Workshops, Observation, Formal/informal Shadowing, Focus groups, Prototyping, Scenarios, Document Analysis, Special purpose records, Questionnaires, Understanding the applicability of techniques.

Requirements Documentation- Documentation styles and levels of definition Requirements Catalogue, Identifier, Name Description, Acceptance criteria,

Requirements Analysis- Prioritizing and packaging requirements for delivery organizing requirements, Ensuring well-formed requirements, Prototyping requirements, and verifying requirements.

Requirements Validation -Agreeing the requirements document, Types of reviews Stakeholders and their areas of concern, Requirements Management- Dealing with changing requirements, The importance of traceability, Vertical traceability (to business objectives), Horizontal traceability (from origin to deliver), Traceability and ownership, Requirements Engineering support tools

References:

1. Requirements Engineering: Fundamentals, Principles, and Techniques by Klaus Pohl
2. Requirements Engineering: A Good Practice Guide by Ian Sommerville, Pete Sawyer

MTSE-202: Secure Software Engineering

Problem, Process, and Product - Problems of software practitioners – approach through software reliability engineering- experience with SRE – SRE process – defining the product – Testing acquired software – reliability concepts- software and hardware reliability. Implementing Operational Profiles - Developing, identifying, crating, reviewing the operation – concurrence rate – occurrence probabilities- applying operation profiles

Engineering "Just Right" Reliability - Defining "failure" for the product - Choosing a common measure for all associated systems. - Setting system failure intensity objectives - Determining user needs for reliability and availability., overall reliability and availability objectives, common failure intensity objective., developed software failure intensity objectives. - Engineering software reliability strategies. Preparing for Test - Preparing test cases. - Planning number of new test cases for current release. -Allocating new test cases. - Distributing new test cases among new operations - Detailing test cases. - Preparing test procedures

Executing Test - Planning and allocating test time for the current release. - Invoking testidentifying identifying failures - Analyzing test output for deviations. – Determining which deviations are failures. Establishing when failures occurred. Guiding Test - Tracking reliability growth - Estimating failure intensity. - Using failure intensity patterns to guide test - Certifying reliability. Deploying SRE - Core material - Persuading your boss, your coworkers, and stakeholders. - Executing the deployment - Using a consultant.

Using UML for Security - UM L diagrams for security requirement -security business process physical security - security critical interaction - security state. Analyzing Model - Notation - formal semantics - security analysis - important security opportunities. Model based security engineering with UML - UML sec profile- Design principles for secure systems - Applying security patterns

Applications - Secure channel - Developing Secure Java program- more case studies. Tool support for UML Sec - Extending UML CASE TOOLS with analysis tools - Automated tools for UML SEC. Formal Foundations - UML machines - Rely guarantee specifications- reasoning about security properties.

References:

1. John Musa D, "Software Reliability Engineering", 2nd Edition, Tata McGraw-Hill, 2005 (Units I, II and III)

2. Jan Jürjens, "Secure Systems Development with UML", Springer; 2004 (Unit IV and V)

MTSE-203: Component Based System Design

INTRODUCTION - Standards – Terms and Concepts – Standardization and Normalization – Components and Interfaces – Callbacks – Contracts – Examples – Processes and multithreading.

UNIT II-JAVA BASED COMPONENT TECHNOLOGIES - Overview and history of java components – Java the language – Java Beans – Java Services – Applets, Servlets, Beans and Enterprise Beans – Advanced Java Services – Interfaces vs Classes in java - JXTA and Jini – Java and Web services.

CORBA -Object and component "wiring" standards – CORBA – The object request broker – CORBA Services – CORBA Component Model – Portable object Adapter – CCM Components, Containers – CORBA Complaint implementations – CORBA facilities– Application objects –CORBA, UML, XML and MDA-Case Study.

NET BASED COMPONENT TECHNOLOGIES- COM - Object reuse - Interfaces and Polymorphism – COM object creation and COM Library – initializing objects, persistence, structural storage, monikers – From COM to distributed COM -Meta-information and Automation – Other COM services- OLE containers and servers - Active X controls – Contextual Composition and Services - .NET framework -.NET components -Assemblies - Common language Frameworks.

COM OVERVIEW AND COMPONENT FRAMEWORKS- Component Architecture - Component Frameworks - Contribution of contextual component frameworks - Frameworks for contextual composition - Black box component framework - Black box and OLE - Portos - Component Development: Component oriented programming - Tools - Component Distribution and Acquisition - Component Assembly-Case Study.

References

1. Clemens Szyperski. Dominik Gruntz and Stephan Murer. "Component Software: Beyond Object-Oriented Programming", Addison-Wesley Professional, 2nd edition, 2011.
2. Ed Roman, "Mastering Enterprise Java Beans", 3rd edition, John Wiley Publications, 2005.
3. Mowbray, "Inside CORBA", Pearson Education, 2006.
4. Hortsamann, Cornell, "CORE JAVA Vol- II", 8th edition, 2008.
5. Sudha Sadasivam, "Component Based Technology", John Wiley and Sons, 2008.
6. Freeze, "Visual Basic Development Guide for COM & COM+", BPB Publication, 2001.

MTSE-204: Object Oriented Software Engineering

INTRODUCTION : System Concepts – Software Engineering Concepts – Development Activities – Managing Software Development – Unified Modeling Language – Overview – modeling concepts – deeper view into UML - Project Organization – Communication

ANALYSIS : Requirements Elicitation – Concepts – Activities – Management – Arena Case Study - Analysis Object Model – Analysis – Concepts – activities - Managing analysis - Case Study

SYSTEM DESIGN: Decomposing the system – Overview of System Design – System Design Concepts – System Design Activities – Addressing Design Goals – Managing System Design – Case Study

OBJECT DESIGN AND IMPLEMENTATION ISSUES : Reusing Pattern Solutions – Concepts – Activities – Managing Reuse – Case Study - Specifying Interfaces – Concepts – Activities – Management – Case Study - Mapping Models to Code – Concepts – Activities – Management – Case Study – Testing – Concepts – Activities – Management

MANAGING CHANGE: Rationale Management – Concepts – Activities – Management - Configuration Management – Concepts – Activities – Management - Project Management - Concepts – Activities – Management – Software Life Cycle

References:

1. Bernd Bruegge and Alan H Dutoit, “Object-Oriented Software Engineering”, 2nd edition, Pearson Education, 2010.
2. Timothy Lethbridge and Robert Laganiere, “Object-oriented Software Engineering: Practical Software Development using UML and Java”, Mc Graw Hill Publication, 2010.

MTSE-205: Genetic Algorithm

Introduction

A brief history of evolutionary computation, Elements of Genetic Algorithms, A simple genetic algorithm, Applications of genetic algorithms

Genetic Algorithms in Scientific models

Evolving computer programs, data analysis & prediction, evolving neural networks, Modeling interaction between learning & evolution, modeling sexual selection, measuring evolutionary activity.

Theoretical Foundation of genetic algorithm

Schemas & Two-Armed and k-armed problem, royal roads, exact mathematical models of simple genetic algorithms, Statistical- Mechanics Approaches.

Computer Implementation of Genetic Algorithm

Data structures, Reproduction, crossover & mutation, mapping objective functions to fitness form, fitness scaling, coding, a multiparameter, mapped, fixed point coding, discretization and constraints.

Some applications of genetic algorithms

The risk of genetic algorithms, De Jong & function optimization, Improvement in basic techniques, current application of genetic algorithms

Advanced operators & techniques in genetic search

Dominance, duplicity, & abeyance, inversion & other reordering operators. Other micro operators, Niche & speciation, multiobjective optimization, knowledge based techniques, genetic algorithms & parallel processors.

Text:

1. David E. Goldberg, "Genetic algorithms in search, optimization & Machine Learning" Addison Wesley, 1989

References:

1. Melanle Mitchell, "An introduction to genetic algorithms" MIT press, 2000.
2. Masatoshi Sakawa, "Genetic Algorithms & Fuzzy Multi objective Optimization", Kluwer Academic Publisher, 2001
3. D. Quagliarella, J Periaux, C Poloni & G Winter, "Genetic Algorithms in Engineering & Computer science", John Wiley & Sons, First edition, 1997

SEMESTER -3

MTSE-301: Software Project Management

PROJECT CONCEPTS AND ITS MANAGEMENT -Project life cycle models-ISO 9001 model-Capability Maturity Model-Project Planning-Project tracking-Project closure. Evolution of Software Economics – Software Management Process Framework: Phases, Artifacts, Workflows, Checkpoints – Software Management Disciplines: Planning / Project Organization and Responsibilities / Automation / Project Control – Modern Project Profiles

COST ESTIMATION -Problems in Software Estimation – Algorithmic Cost Estimation Process, Function Points, SLIM (Software Life cycle Management), COCOMO II (Constructive Cost Model) – Estimating Web Application Development – Concepts of Finance, Activity Based Costing and Economic Value Added (EVA) – Balanced Score Card.

OFTWARE QUALITY MANAGEMENT- Software Quality Factors – Software Quality Components – Software Quality Plan – Software Quality Metrics – Software Quality Costs – Software Quality Assurance Standard – Certification – Assessment.

SOFTWARE MANAGEMENT AND METRICS - Software Configuration Management – Risk Management: Risk Assessment: Identification / Analysis / Prioritization – Risk Control: Planning / Resolution / Monitoring – Failure Mode and Effects Analysis (FMEA) – Defect Management – Cost Management.Software Metrics – Classification of Software Metrics: Product Metrics: Size Metrics, Complexity Metrics, Halstead's Product Metrics, Quality Metrics, and Process metrics.

PROJECT EVALUATION AND EMERGING TRENDS - Strategic Assessment–Technical Assessment–Cost Benefit Analysis–Cash Flow Forecasting–Cost Benefit Evaluation Technique–Risk Evaluation–Software Effort Estimation. Emerging Trends: Import of the internet on project Management – people Focused Process Models.

References

1. Ramesh Gopalaswamy , "Managing and global Software Projects", Tata McGraw Hill Tenth Reprint, 2011.
2. Roger S.Pressman, "Software Engineering- A Practitioner's Approach", 7th Edition ,McGraw Hill, 2010.
3. Daniel Galin, "Software Quality Assurance: from Theory to Implementation", Addison-Wesley, 2003.
4. Bob hughes and Mike Cotterell, "Software Project Management" second edition,1999. 5. Royce, W. "Software Project Management: A Unified Framework", AddisonWesley, 1998.

MTSE-302 Software Reliability

INTRODUCTION- Need and Concepts of Software Reliability, Failure and Faults – Prevention, Removal, Tolerance, Forecast, Dependability Concept – Failure Behavior, Characteristics, Maintenance Policy, Reliability and Availability Modeling, Reliability Evaluation

SOFTWARE RELIABILITY MODELS- Introduction - Historical Perspective and Implementation, classification, limitations and issues, Exponential Failure Models – Jelinski-moranda model, Poisson, Musa, Exponential models, Weibull Model, Musa-okumoto Model, Bayseian Model – Littlewood verral Model, Phase Based Model

PREDICTION ANALYSIS- Model Disagreement and Inaccuracy – Short & Long Term Prediction, Model Accuracy, Analyzing Predictive Accuracy – Outcomes, PLR, U & Y Plot, Errors and Inaccuracy, Recalibration – Detecting Bias, Techniques, Power of Recalibration, Limitations in Present Techniques, Improvements.

THE OPERATIONAL PROFILE- Concepts and Development Procedures – Customer Type, User Type, System Mode, Functional and Operational Profile, Test Selection - Selecting Operations, Regression Test, Special Issues – Indirect Input Variables, Updating, Distributed system, CASE STUDY - Application of DEFINITY & FASTAR, Power Quality Resource System

UNIT V-TESTING FOR RELIABILITY MEASUREMENT - Software Testing – Types, White and Black Box, Operational Profiles – Difficulties, Eatimating Reliability, Time/Structure based software reliability – Assumptions, Testing methods, Limits, Starvation , Coverage, Filtering, Microscopic Model of Software Risk.

References

1. 1. Patric D. T.O connor, “Practical Reliability Engineering”, 4th Edition, John
2. Wesley & sons, 2003.
3. 2. John D. Musa, “Software Reliability Engineering”, Tata McGraw Hill, 1999.
4. 3. Michael Lyu, “Handbook of Software Reliability Engineering”, IEEE Computer
5. Society Press, ISBN: 0-07-039400-8, 1996.

DEPARTMENT OF INFORMATION TECHNOLOGY

MTSE-303 SOFTWARE REUSE & RE- ENGINEERING

UNIT-I

Software reuse definitions, Reuse Artefacts, Artefact Characteristics, Reuse in Software Life-Cycle. Assessing the R Process and its Goals, software reuse principles software reuse scope, reuse potential. restricting factors, diversity, solt reuse methods. generative reuse methods, compositional reuse methods, reuse within object oriented methodol application frameworks

UNIT-II

Design Patterns- Introduction. Creational Patterns - Factory Pattern, Factory Method. Abstract Factory Pattern, Singh Pattern, Builder Pattern.

Structural Patterns -- Adapter Pattern, Bridge Pattern, Composite Pattern, Decorator Pattern, Façade Pattern, Flies Pattern. Proxy Pattern Behavioral Patterns - Chain of responsibility Pattern, Command Pattern, Interpreter Pattern.

UNIT-III

Object Oriented Business Engineering -Business Process Reengineering, Software Engineering Process in reuse business Component System Engineering - building flexible components systems, requirement analysis, robustness analysis, desi implementation and testing the component system.

UNIT-IV

Introduction Re-engineering. Restructuring and Reverse Engineering, Reverse engineering, Static analysis. Dynamic analysis. Scope of reverse engineering. Codes, specifications. and documentations. Domain analysis. Design recovery

UNIT-V

Re- engineering existing systems, refactoring, Data Re-engineering and migration. Software Reuse and Reengineering Design for reuse. Object-orientation and reuse, reuse metrics. Reengineering Metrics. Reengineering and Maintenance in software cycle.

Suggested Reading:

1.Ivar Jaco Martin Griss, Patrick Johnsson, “Software Reuse Architecture, Process and Business Success” Pearson Education, 2003.

2 James W Cooper, “Java Design Pattes a tutorial”, Pearson Education, 2003.
3.Frank Buschmann, et al., "Pattern Oriented Software Architecture – Volume I” John Wiley & Sons; 1996.

Textbook:

1.Taking A.A, and Grubb, P.A (1996) **Software Maintenance: Concepts and Practice**. Boston International Thomson Computer Press.
2 Leac’h, R.J. (1997). **Software Reuse: Methods, Models, and Costs**. New York: McGraw-Hill.

MTSE-304: Software Engg. Process, Methods & Tool

Unit 1: Introduction to Software Engineering Process. The Software Development Life Cycle(SDLC). Importance of SDLC, Requirement Gathering and Analysis, Designing. Implementation or Coding. Integration and Testing, Deployment, Operations and Maintenance.

Unit 2: Software Engineering Models: Waterfall Model, Iterative Model, Rapid Application Development (RAD Model). V&V Model, Spiral Model. Prototype Model, Comparison among Development Models, Introduction to Rational Unified Process, Phases of Rational Unified Process.

Unit 3: Software Engineering Methods, Extreme Programming, Feature Driven Development methodology. Joint Application Development Methodology, Scrum Development Methodology, Dynamic Systems Development Method, Agile Software Development Methodology.

Unit 4: The Software Requirements Document, Software Analysis and Design Tools: E-R Diagram, Data Flow Diagram, Structural Chart, Decision Table, Data Directory, Project Scheduling and Tracking Tools: PERT, CPM and Gantt chart.

Unit 5: Software Engineering Tools, Computer Aided Software Engineering (CASE) Tool, Database Tools for Software, Programming Language Tools, Web Application Tools, SCM Tools, Design and Analysis Tools, Testing Tools

References

I. R. S. Pressman, Software Engineering: A Practitioners Approach, McGraw Hill.
2. Rajib Mall. Fundamentals of Software Engineering, PHI Publication.
3. Ian Sommerville, "Software Engineering", Pearson Education Publications.
4. Kent Beck, "Extreme Programming Explained: Embrace Changes" Addison Wesley. 1999
5. Ken Schwaber and Mike Beedle, "Agile Software Development with SCRUM" Prentice Hall, 2002 ISBN 0-13-067634-9.