Babu Banarasi Das University, Lucknow

Department of Computer Science & Engineering

School of Engineering

Master of Technology (Software Engineering)-Regular

Evaluation Scheme

SEMESTER I									
Course Category	Course Code	Code Title	Contact Hours			Evaluation Scheme			Credits
			L	T	P	CIA	ESE	Course Total	
C	MCS4111	Advance Software Engineering Principles & Practices	4	0	0	40	60	**100**	**4**
C	MCS4112	Advance Database Management Systems	4	0	0	40	60	**100**	**4**
C	MAS4001	Probability and Statistical Analysis	4	0	0	40	60	**100**	**4**
C	MCS4103	Advanced Data Structure and Algorithms	4	0	0	40	60	**100**	**4**
GE		**Generic Elective I**	4	0	0	40	60	**100**	**4**
C	MCS4151	Advance Database Management Systems Lab	0	0	2	40	60	**100**	**1**
C	MCS4152	Advanced Data Structu and Algorithms Lab	e 0	0	2	40	60	**100**	**1**
C	MCS4153	Seminar	0	0	2	100	0	**100**	**1**
Total			**20**	**0**	**6**	**380**	**420**	**800**	**23**

SEMESTER II									
Course Category	Course Code	Code Title	Contact Hours			Evaluation Scheme			Credits
			L	T	P	CIA	ESE	Course Total	
C	MCS4211	Software Design and Testing	4	0	0	40	60	**100**	**4**
C	MCS4212	Concepts of Deep Learning	4	0	0	40	60	**100**	**4**
C	MCS4213	Software Project Management	4	0	0	40	60	**100**	**4**

GE		**Generic Elective II**	4	0	0	40	60	**100**	**4**
GE		**Generic Elective III**	4	0	0	40	60	**100**	**4**
C	MCS4251	Software Design and Testing Lab.	0	0	2	40	60	**100**	**1**
C	MCS4252	Deep Learning Lab	0	0	2	40	60	**100**	**1**
C	MCS4253	Seminar	0	0	2	100	0	**100**	**1**
Total			**20**	**0**	**6**	**380**	**420**	**800**	**23**

SEMESTER III									
Course Category	**Course Code**	**Code Title**	**Contact Hours**			**Evaluation Scheme**			**Credits**
			L	**T**	**P**	**CIA**	**ESE**	**Course Total**	
C	MCS4351	State of the art seminar[#]	-	-	-	200	-	**200**	**4**
C	MCS4352	Thesis I[*]	-	-	-	400	-	**400**	**16**
Total			**-**	**-**	**-**	**600**	**-**	**600**	**20**

SEMESTER IV									
Course Category	**Course Code**	**Code Title**	**Contact Hours**			**Evaluation Scheme**			**Credits**
			L	**T**	**P**	**CIA**	**ESE**	**Course Total**	
C	MCS4451	Thesis II[**]	-	-	-	200	800	**1000**	**28**
Total			**-**	**-**	**-**	**200**	**800**	**1000**	**28**

[#]**SOTA:** The student need to perform a literature survey, will give presentation on state of art topics and will submit a synopsis already mentioned in the problem statement. This will be evaluated internally within two months of the start of the semester and result will be intimated to the students, so to precede for Thesis I.

[*]The student will develop a workable model for the problem they have proposed in the synopsis

[**]This is in continuation with Thesis I. The required experimental/ mathematical verification of the proposed model will be done in this semester.

Legends:

L Number of Lecture Hours per week

T Number of Tutorial Hours per week
P Number of Practical Hours per week
CIA Continuous Internal Assessment
ESE End Semester Examination

Category of Courses:
F Foundation Course
C Core Course
GE Generic Elective
OE Open Elective

Course Code	Generic Elective I
GE44911	Block Chain Technology
GE44912	Software Agents
GE44913	Machine Leraning
GE44914	Real Time & Concurrent Systems
GE44915	Hardware Architecture for Artificial Intelligence

Course Code	Generic Elective II
GE44921	Software Reliability Engineering
GE44922	Software Architecture and Integration
GE44923	System Simulation and Modeling
GE44924	Agile Software Engineering

Course Code	Generic Elective III
GE44931	Embedded System Theory & Design
GE44932	Computer Vision
GE44933	Software Reusability
GE44934	Formal Software Specifications

Credit Summary Chart						
Course Category	Semester				Total Credits	%age
	I	II	III	IV		
F	0	0	0	0	0	0
C	19	15	20	28	82	87.23
GE	4	8	0	0	12	12.77
Total	**23**	**23**	**20**	**28**	**94**	**100.00**

Discipline Wise Credit Summary Chart						
Course Category	Semester				Total Credits	%age
	I	II	III	IV		
Basic Sciences	0	0	0	0	0	0
Professional Subject Core	18	14	16	28	76	80.85
Professional Subject -General Elective	4	8	0	0	12	12.77
Project Work, Seminar and / or internship in Industry or elsewhere.	1	1	4	0	6	6.38
Total	**23**	**23**	**20**	**28**	**94**	**100**

MCS4111 Advance Software Engineering Principles and Practices

Course Objective:

1. Develop technologically competent computer professionals in today's IT centric scenario by training them in the contemporary software engineering principles and paradigms.
2. Provide students a deep insight into various cutting edge technologies& tools and thereby creating diverse career opportunities.
3. Improve analytical, logical and presentation skills of the students.
4. Applying evolving technologies of software engineering in developing practical solutions to complex problems in consonance with the legal and ethical responsibilities.
5. Provide the students with project engineering and management skills catering to the changing industry needs and constraints across the advancing domains of computing.

Learning Outcome:

After completing the course, the students should be able to:

1. Apply the knowledge of software engineering principles and paradigms in the design of system components and processes that meet the specific needs of the industry.
2. Identify, analyze and formulate solutions to complex engineering problems using innovative and emerging technologies.
3. Conceptualize and solve engineering problems with feasible optimal solutions in consideration of socio-economic factors.
4. Extract information relevant to novel problems and apply appropriate research methodology to develop scientific knowledge.
5. Use the techniques, skills and CASE tools necessary for engineering practice and coordinate the construction, deployment and maintenance of software systems.

Course Contents:

Module	Course Topics	Total Hours	Credits
I	System Engineering Computer based systems, system engineering hierarchy, Information engineering, Information strategy planning, Business area analysis, Product engineering, Modeling the system architecture, System	30 Hours	1

	modeling and simulation, system specification. Computer Based System Engineering: Emergent system properties		
II	Modern Design Concept and Principles Design Concepts: Mapping of analysis model to design model, Design process, design principles, Design concepts, effective modular design, Design model, design specification. Architectural Design Process: Transform mapping and transaction mapping, Design post processing, interface design, Human computer interface design, and Interface design guidelines.	30 Hours	1
III	Real Time Software Design Real–time systems, definition, System consideration, Real time system analysis, stimulation / Response systems, Software architecture, Design Patterns, User Interface Design, Object Oriented Design with UML, Universal design applied to software engineering, Design for Reuse.	30 Hours	1
IV	Agile Programming- Introduction, Flavors of Agile Development, Agile Manifesto, Refactoring Techniques, Limitations of The Agile Process. Extreme Programming (XP)- Introduction, XP Equation, XP Values, Assuming Sufficiency- Sufficient time and resources, Constant change of cost, Developer effectiveness, Freedom to experiment.	30 Hours	1

Text/Reference Books:

1. Roger S. Pressman, "Software Engineering –A Practitioner's Approach", McGraw Hill Publications, 2009.
2. Ian Sommerville, "Software Engineering", Pearson Education Publications, 2017.
3. Shari Lawarence Pfleeger, "Software Engineering Theory and Practices", Pearson Education, 2017.
4. John W. Satzinger, Robert B Jackson, Stephen D Burd, "System Analysis and Design in Changing World", Cengage Learning, 2009.
5. Jeff Tain, Software Quality Engineering, IEEE Publication, 2004.

MCS4112 Advance Database Management Systems

Course Objective:

1. Advanced database course aims at developing computer applications with different kinds of data models.
2. A range of features and benefits of Advanced Database Management Systems.
3. Introduction and Understanding about parallel databases
4. Introduction and Understanding about object oriented databases
5. Introduction and Understanding about web databases and emerging trends in database systems.

Learning Outcome:

After completing the course, the students should be able to:

1. Study the needs of different databases.
2. Get familiarized with transaction management of the database.
3. Gain knowledge about the web and intelligent databases.
4. Provide an introductory concept about the way in which data can be stored in geographical information systems.

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**Review of Relational Data Model and Relational Database Constraints:** Relational model concepts; Relational model constraints and relational database schemas; Update operations, anomalies, dealing with constraint violations, Types and violations. Overview of Object Oriented Concepts: Objects, Basic properties. Advantages, examples, Abstract data types, Encapsulation, class hierarchies, polymorphism, examples. **PARALLEL DATABASES** Database System Architectures: Centralized and Client Server Architecture, Server System Architecture, Parallel Systems, Distributed Systems. Parallel Databases: I/O Parallelism, Inter and Intra Query Parallelism. Inter and Intra operation Parallelism, Case Studies.	30 Hours	1

II	**OBJECT ORIENTED DATABASES** Object Oriented Databases Introduction, Weakness of RDBMS. Object Oriented Concepts, Storing Objects in Relational Databases, Next Generation Database Systems. Object Oriented Data models, OODBMS Perspectives, Persistence and Issues in OODBMS. Object Oriented Database Management System Manifesto, Advantages and Disadvantages of OODBMS, Object Oriented Database Design, OODBMS Standards and System, Object Management Group, Object Database Standard ODMG, Object Relational DBMS, Postgres, Comparison of ORDBMS and OODBMS.	30 Hours	**1**
III	**WEB DATABASES** Web Technology And DBMS: Introduction, The Web, The Web as a Database Application Platform. Scripting languages, Common Gateway Interface, HTTP Cookies, Extending the Web Server, Java, Microsoft's Web Solution Platform, Oracle Internet Platform, Semi structured Data and XML: XML Related Technologies, XML Query Languages.	30 Hours	1
IV	**INTELLIGENT DATABASES** Enhanced Data Models For Advanced Applications, Active Database Concepts And Triggers, Temporal Database Concepts: Deductive databases and Knowledge Databases **CURRENT TRENDS** :Mobile Database, Geographic Information Systems, Genome Data Management, Multimedia Database, Parallel Database, Spatial Databases, Database administration, Data Warehousing and Data Mining.	30 Hours	1

Text/Reference Books:

1. Thomas M. Connolly, Carolyn E. Begg, "Database Systems - A Practical Approach to Design, Implementation, and Management", Third Edition , Pearson Education, 2003.

2. Ramez Elmasri & Shamkant B.Navathe, "Fundamentals of Database Systems", Fourth Edition , Pearson Education , 2004.

3. Tamer Ozsu M., Patrick Ualduriel, "Principles of Distributed Database Systems", Second Edition, Pearson Education, 2003.

4. Prabhu C.S.R., "Object Oriented Database Systems", PHI, 2003.

5. Peter Rob and Corlos Coronel, "Database Systems – Design, Implementation and Management", Thompson Learning, Course Technology, 5th Edition, 2003.

6. Subramanian V.S., "Principles of Multimedia Database Systems", Harcourt India Pvt Ltd., 2001.

7. Vijay Kumar, "Mobile Database Systems", John Wiley & Sons, 2006

MCS4103 Advanced Data Structure and Algorithms

Course Objective:

1. The student should be able to choose appropriate data structures, understand the ADT/libraries, and use it to design algorithms for a specific problem.
2. Students should be able to understand the necessary mathematical abstraction to solve problems.
3. To familiarize students with advanced paradigms and data structure used to solve algorithmic problems.
4. Student should be able to come up with analysis of efficiency and proofs of correctness

Learning Outcome:

After completing the course, the students should be able to:

1. implement a symbol table using hashing technique.
2. develop text processing algorithms.
3. choose suitable data structure and develop algorithms for computational geometrical problem
4. work with an advanced abstract data type (ADT) and their implementations.
5. work in the domain of text processing.

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**Hashing:** Review of Hashing, Hash Function, Collision Resolution Techniques in Hashing, Separate Chaining, Open Addressing, Linear Probing, Quadratic Probing, Double Hashing, Rehashing, Extendible Hashing, Hash tables in data-compression, LZW algorithm. **Skip Lists:** Need for Randomizing Data Structures and Algorithms, Search and Update Operations on Skip Lists, Probabilistic Analysis of Skip Lists, Deterministic Skip Lists.	**30**	**1**
II	**Trees:** Abstract data type, sequential and linked implementations, recursive and non recursive tree traversal and algorithms Binary trees and their properties, Threaded Binary Trees - differentiation, leftist trees, tournament trees, use of winner trees in mergesort as an external sorting algorithm, bin packing. **Search Trees:** Binary search trees, search efficiency, insertion and deletion operations,	**30**	**1**

	importance of balancing, AVL trees, searching, insertion and deletions in AVL trees, Tries, 2-3 Trees, B-Trees, Red Black Trees, Splay Trees.		
III	**Complexity Analysis:** Asymptotic notation, Algorithm Analysis, NP-Hard and NP-Completeness. **Text Processing:** String Operations, Brute-Force Pattern Matching, The Boyer- Moore Algorithm, The Knuth-Morris-Pratt Algorithm, Standard Tries, Compressed Tries, Suffix Tries, The Longest Common Subsequence Problem (LCS). **Parallel Algorithm:** Performance Measures of Parallel Algorithms, Parallel Merging/Sorting Algorithms on CREW/EREW, Parallel searching algorithms.	**30**	**1**
IV	**Graph Algorithms:** Elementary Graph Algorithms, Minimum Spanning Trees, Single Source Shortest Paths, All Shortest Paths, Maximum flow, Multithreaded algorithms, Matrix Operations. **Computational Geometry:** One Dimensional Range Searching, Two-Dimensional Range Searching, constructing a Priority Search Tree, Searching a Priority Search Tree, Priority Range Trees, Quadtrees, k-D Trees.	**30**	**1**

Text/Reference Books:

1. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, MIT Press, 3/e, 2009.
2. M T Goodrich, Roberto Tamassia, Algorithm Design, John Wiley, 2014.
3. S. Sahni, "Data structures, Algorithms and Applications in Java", Universities Press.2005,[ISBN:0-07-109217-x]
4. Adam Drozdek, "Data structures and Algorithms in Java", 3rd edition, Cengage Learning, 2008. [ISBN:978-9814239233]
5. R.Lafore "Data structures and Algorithms in Java", Pearson education, 2003. ISBN: 9788 131718124.
6. Mark Allen Weiss, Data Structures and Algorithm Analysis in C++, 2nd Edition, Pearson, 2004.
7. S. Sahni, Data Structures, Algorithms, and Applications in C++, Silicon Press, 2/e, 2005.
8. A. M. Tenenbaum, Y. Langsam, and M. J. Augenstein, Data Structures Using C and C++, Prentice Hall, 2/e, 1995.

MCS4151 Advance Database Management Systems Lab

Experiments

1. Implementation of Queries related to various DDL commands.

2. Implementation of Queries related to various DML commands.
a) Implement Queries using Logical Operators, SQL Operators.
b) Implement Queries using Character, Numbers, Date, Group by, Order by Clauses.
c) Implement Queries for different Join operations (Natural, Equi, Outer, Inner).

3. Implementation of various Queries to familiarize with the syntax and structure of the SQL and to test the queries for any logical errors.
a) Analyze any social networking site properly and prepare the document of all the database functionalities for the website.
b) Analyze any e-commerce site properly and prepare the document of all the database functionalities for the website.

4. Implementation of various Queries to familiarize with basic constructs involved in efficient database design. Learn to read and create database design documents.
a) Consider the case study of online shopping cart like Flipkart/ Amazon /Snapdeal and perform the following operations:
   - Write the scenario.
   - Identify the possible entity-set.
   - Construct a relationship matrix.
   - Draw a rough ER diagram.
   - List all the Assertions for all relationships.
   - Draw a detailed ER diagram after incorporating all the Assertions.
   - Design the relational database schema.

   Create the database using Oracle/SQL Server/MySql software package
b) Determine the user name along with the products in their cart and their price.
c) Determine all the users who have ordered any item.
d) Determine the details of the seller who made the maximum revenue.
e) Consider the famous website LinkedIn and analyze its functionalities. Create the task of creating a database schema for your own based on the functionalities you observed.

5. Create and Implement PL/SQL procedures and triggers.
a) Design and create a database consists of events organized in various countries of different continents. Follow all the requisite steps to design and create the database.
b) Based on the above database, create Stored procedures.
c) Create Views and understand its concepts.

6. Consider a case study of any Big Data system of your choice and design the distributed database architecture and also analyze the probable solutions available in the market.
b) Analyse various distributed database architectures and case studies.
c) Solve real world problems of Big databases using available tools.

MCS4152 Advanced Data Structure and Algorithms Lab

Implement the following using C/C++/Java/python.

1. Write a program to implement open hashing.
2. Write a program to implement close hashing.
3. Write a program to implement the LZW algorithm.
4. Write a C program that uses functions to perform the following: i) Creating a SplayTree of integers ii) Traversing the above binary tree in preorder, inorder and postorder.
5. Write a C program to perform the following: i) Creating a B-Tree of integers ii) Traversing the above binary tree in preorder, inorder and postorder.
6. Write a program to simulate various graph traversing algorithms.
7. Write a program to implement the the Boyer- Moore Algorithm
8. Write a program to implement the the Knuth-Morris-Pratt Algorithm,
9. Write a dynamic program to implement the Longest Common Subsequence Problem (LCS).
10. Write a program to implement Maximum flow.
11. Write a program to implement Multithreaded algorithms.
12. Write a program to implement Matrix Operations.

MCS4211 Software Design and Testing

Course Objective:

1. Study the basic concepts of testing.
2. Understand the static and dynamic testing.
3. Learn about software quality model.
4. Understand white box and black box testing.
5. Understand the development of test cases.

Learning Outcome:

After completing the course, the students should be able to:

1. Appreciate the fundamentals of software testing and its application through the software life cycle.
2. Develop skills in designing and executing software tests suitable for different stages in the software life cycle.
3. Understand and appreciate the role of software testing in systems development, deployment and maintenance.
4. Develop a continuing interest in software testing, and obtain satisfaction from its study and practice.
5. Appreciate the responsibilities of software testers within software projects, the profession and the wider community.

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**INTRODUCTION TO SOFTWARE TESTING** Review of software testing, Overview of software testing evolution, Myths, facts, goals, principles of testing, Models and psychology of testing, Study of bugs, Classification, priority, severity of bugs, Tracking of bugs, Software Testing, Life cycle of software testing, Types of testing, Test Plan, Test plan specification, Leveled Test Plan, Development of Test Plan, Master Test Plan, Phase Wise Test Plan, Test Management, Software Testing Guidelines, Defect Management, Analyzing and Reporting Test	30 Hours	1

II	**TESTING TECHNIQUE** Static Testing, Inspection, Structured Walkthrough, Technical reviews Black Box Testing, Boundary Value Analysis, Equivalence Class Testing, Cause Effect Graph Based Testing, White Box Testing, Basis Path Testing, Control Structure Testing, Mutation Testing, Grey Box Testing	30 Hours	1
III	**SOFTWARE TESTING STRATEGIES** Unit Testing, Integration Testing, System and Acceptance Testing, Alpha Testing and Beta Testing, Stress Testing. Incremental Design: Black box to white box in stages, Prototyping , An example- DSDM. Structured Systems Analysis and Structured Design: Origins, developments and philosophy, Representation forms for SSA/SD, The SSA/SD process, The role of heuristics in SSA/SD, Extended forms of SSA/SD, SSA/SD: an outline example.	30 Hours	1
IV	**A MODEL FOR TESTING** The Project, Overview, The Environmental, The Program, Bugs, Tests, Testing and Levels, The Role of Models Flow graphs and Path Testing: Path Testing Basics, Predicates, Path Predicates, and Achievable Paths, Path Sensitizing, Path Instrumentation, Complement and Application of Path Testing, Generalizations. Transaction- Flow Testing: Transactions Flows, Transactions – Flows Testing Techniques, Implementation Comments.	30 Hours	1

Text/Reference Books:

1. D. Budgen- Software Design, 2nd Edn Pearson Education, New Delhi, 2004.
2. B. Bezier- Software Testing 2nd Edn, Techniques, Dreamtech, New Delhi, 2004.
3. William Perry, "Effective Methods for Software Testing", John Wiley & Sons, New York, Van Nostrand Reinhold, New York, 2nd Ed., 1995.
4. CemKaner, Jack Falk, Nguyen Quoc, "Testing Computer Software", Van Nostrand Reinhold, New York, 2nd Ed., 1993.

MCS4212 Concepts of Deep Learning

Course Objective:

1. Understand the Basic Concept of Deep Learning
2. Understand complexity of Deep Learning algorithms and their limitations
3. Applying common Deep Learning algorithms in practice and implementing their own
4. Perform experiments in Deep Learning using real-world data.

Learning Outcome:

After completing the course, the students should be able to:

1. Understand the concepts of TensorFlow, its main functions, operations and the execution
2. Implement Deep learning algorithms, understand neural networks and traverse the Data Abstraction layers
3. Build deep learning models in TensorFlow and interpret the results
4. Build own Deep learning project
5. Troubleshoot and improve deep learning models

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**Introduction:** Introduction to Deep Learning, History of Deep Learning-A Probabilistic Theory of Deep Learning, Basic of Neural Network, Introduction to Perceptron Training Rule, Gradient Descent Rule, Back propagation and regularization, batch normalization, Neural Nets-Deep Vs Shallow Networks, Convolutional Networks, Activation Functions : Sigmoid, ReLU, Hyperbolic Functions, Soft max, Semi supervised Learning	30	1
II	**Deep Networks:** Introduction to Convolutional Neural Networks, Principles behind CNNs, Generative Adversarial Networks (GAN),Kernel filter, Multiple Filters, CNN applications, Introduction to Recurrent Neural Networks: Unfolded RNNs, Seq2Seq	30	1

	RNNs, Stochastic Optimization Generalization in neural networks:Spatial Transformer Networks- Recurrent networks LSTM, RNN applications, Introduction to Tensor Flow : Computational Graph, Key highlights, Creating a Graph,		
III	**Dimentionality Reduction:** Linear (PCA, LDA) and manifolds, metric learning - Auto encoders and dimensionality reduction in networks, Introduction to Convnet Architectures:– AlexNet, VGG, Inception, ResNet, Training a Convnet: weights initialization, batch normalization, hyperparameter optimization, Optimization in deep learning, Non-convex optimization for deep networks, , Deep Reinforcement Learning:- Computational & Artificial Neuroscience	30	1
IV	**Deep Learning Applications:** Image Processing, Natural Language Processing, Speech Recognition, Video Analytic	30	1

Text Books:

1. Goodfellow, I., Bengio,Y., and Courville, A., Deep Learning, MIT Press, 2016.
2. Satish Kumar, Neural Networks: A Classroom Approach, Tata McGraw-Hill Education, 2004
3. Yegnanarayana, B., Artificial Neural Networks PHI Learning Pvt. Ltd, 2009

Reference Books:

1. Michael Nielsen, Neural Networks and Deep Learning, Determination Press, 2015.
2. Deng & Yu, Deep Learning: Methods and Applications, Now Publishers, 2013
3. Yegnanarayana, B., Artificial Neural Networks PHI Learning Pvt. Ltd, 2009

MCS4213 Software Project Management

Course Objective:

1. Understand the importance of software project management.
2. To provide basic project management skills with a strong emphasis on issues and problems associated with delivering successful IT projects.
3. The module is designed to provide an understanding of the particular issues encountered in handling IT projects and to offer students methods, techniques and 'hands-on' experience in dealing with them.
4. Planning and tracking in the implementation of the software project management process.

Learning Outcome:

After completing the course, the students should be able to:

1. Participate in a software development project as a project manager.
2. Take responsibility of a project team and project organization.
3. Apply theoretical knowledge on project management and software development into practice.
4. Have good knowledge of the issues and challenges faced while doing the Software project Management.
5. Do the Project Scheduling, tracking, Risk analysis, Quality management and Project Cost estimation using different techniques

Course Contents:

Module	Course Topics	Total Hours	Credits
I	Fundamentals of Software Project Management (SPM) - Need Identification - Vision and Scope document - Project Management Cycle - SPM Objectives - Management Spectrum- SPM Framework Planning fundamentals - major issues in software project planning – planning activities - Project master schedule - software risk management – risk monitoring – risk analysis.	30 Hours	**1**

II	Software cost- major issues in estimating software cost- cost estimation method- Experience based model - parameter based model - COCOMO- versions of COCOMO – Software size estimating – function points – software project schedule – Raleigh model.	30 Hours	**1**
III	Function organization–project organization–matrix organization–staffing quality replacement–turnover management. Directing a software engineering project – issues – activities conflict management. Issues in controlling software project- controlling activities threads of control-Work breakdown structures- earned value tracking. Earned Value Indicators: Budgeted Cost for Work Scheduled (BCWS), Cost Variance (CV), Schedule Variance (SV), Cost Performance Index (CPI), Schedule Performance Index (SPI), Interpretation of Earned Value Indicators, Error Tracking, Software Reviews	30 Hours	**1**
IV	Team Software Process TSPi Overview: What is TSPi? TSPi Principles, the TSPi Design, TSPi Structure and Flow the TSPi Process, the Textbook Structure and Flow. The Logic of the Team Software Process: Why Projects Fail, Common Team Problems, What is Team? Building Effective Teams, How Teams Develop, How TSPi Builds Teams.	30 Hours	**1**

Text/Reference Books:

1. Walker Royce: —Software Project Management- Addison-Wesley, 1998.
2. Watts S. Humphrey "Introduction to the Team Software Process(sm)", Carnegie-Mellon University, Addison Wesley Professional, 2000.
3. Bob Hughes, Mike Cotterell and Rajib Mall: Software Project Management – Fifth Edition, McGraw Hill, New Delhi, 2012.
4. Robert K. Wysocki —Effective Software Project Management – Wiley Publication, 2011.
5. Gopalaswamy Ramesh, —Managing Global Software Projects – McGraw Hill Education (India), Fourteenth Reprint 2013.

MCS4251 Software Design and Testing Lab

1. Learn about different techniques of testing a software. Design unit test cases to verify the functionality and locate bugs, if any.
2. Consider the following program segment:

```
int max (int i, int j, int k)
{
int max;
if (i>j) then
if (i>k) then
max=i;
else max=k;
else if (j > k)
max=j
else max=k
return (max);
}
```

   a) Draw the control flow graph for this program segment
   b) Determine the cyclomatic complexity for this program
   c) Determine the independent paths
3. Create a Test plan Document for any application.
4. Take ATM system and study its system specifications and report various bugs.
5. A program for written in C language for Matrix Multiplication fails. Introspect the causes for its failure and write down the possible reasons for its failure.
6. Study of testing tool (e.g. winrunner)
7. Study of web testing tool (e.g. selenium)
8. Study of bug tracking tool (e.g. bugzilla)
9. Study of any test management tool (e.g. test director)
10. Study of any open source testing tool (e.g. test link)

MCS4252 Deep Learning Lab

1. To Write a program to implement Perceptron.
2. To write a program to implement AND OR gates using Perceptron.
3. To implement Crab Classification using pattern net.
4. To write a program to implement Wine Classification using Back propagation.
5. To write a MatLab Script containing four functions Addition, Subtraction, Multiply and Divide functions.
6. Write a program to implement classification of linearly separable Data with a perceptron.
7. To study Long Short Term Memory for Time Series Prediction.
8. To study Convolutional Neural Network and Recurrent Neural Network.
9. To study ImageNet, GoogleNet, ResNet convolutional Neural Networks.
10. To study the use of Long Short Term Memory / Gated Recurrent Units to predict the stock prices based on historic data.

GE44911 Block Chain Technology

Course Objective:

1. Explain designing principles of Blockchain technology.
2. Understand how Blockchain system (Bitcoin and Ethereum) work.
3. Describe the differences between the proof of work and proof of stake system.
4. Interaction between Blockchain system by sending and reading transaction.
5. Explain about all legal regulatory bodies for the dark side of Blockchain technology.

Learning Outcome:

After completing the course, the students should be able to:

1. Learn about the centralized and decentralized ledger system along with application and disadvantages.
2. Understand the Importance of cryptographies algorithms behind the Blockchain technology.
3. Evaluate security, privacy and integrity of given Blockchain system.
4. Design, build and deploy smart contracts and distributed systems.
5. Understand about the Criminal activities and usages of Legal actions.

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**Introduction:** Centralized System- Advantages and disadvantages; Decentralized: Need of Decentralized system, Advantages and Disadvantages, Issue of Decentralized system: Security, Integrity and privacy; **Blockchain Technology:** Introduction, Barriers in Blockchain, Use of Blockchain, Real-Life scenario Challenges in Blockchain, Design Issues in Blockchain, Trustlessness and Immutability of Blockchain Technology.	30	**1**

II	**Cryptographic Elements and Technology in Blockchain:** public key and private key, Public Key Cryptography, Digital signature and Hash function, Role of Encryption in Blockchain; Modification and Maintainability of transaction, Privacy Protection techniques. **Blockchain Platforms:** Classification of Blockchain Technology, Bitcoin: Basic and Mechanism; Ethereum in FinTech Ecosystem; Tokenizing Method- Tokenizing Shares and Fund Raising, Hyper Ledger.	30	**1**
III	**Blockchain Application:** Selection Criteria of Blockchain application- Key Factors, Best Fit application, Decision Making system; Blockchain in Enterprise System- Role of Permission Blockchain; proof of Work Vs. Proof of Stake. **Blockchain Use Case:** Trades Finance, Supply chain Financing, Cross border Connectivity- Trusted data transfer, Capital Markets and for General Government Services & Sustainable Livelihood.	**30**	**1**
IV	**Challenges and Opportunities of Blockchain:** Challenges in Blockchain design, Security, Privacy Risks and Limitations of Blockchain, Benefits of Blockchain in Banking and Healthcare industry ; Facebook's Libra Development in Blockchain, DLT and Cryptocurrency. **Legal Regulation for Blockchain:** Criminal activities in Blockchain, Illegal Use of Payment in Blockchain; Need of Legal Regulation, Global Digital Assets Regulatory Trends, Smart Contracts.	**30**	**1**

Text/Reference Books:

1. "The Blockchain Developer: Practical Guide for Designing, Implementiong, Testing and securing Blockchain based system." By Elad Elrom published by Apress; 1st edition (July 24, 2019)
2. 'Blockchain Quick review", by Brenn Hill, Samanyu Chopra and Paul Valencourt, Packt Publisher 1st edition (August 2018).

3. "Cryptoassets: The Innovative Investor's Guide to Bitcoin and Beyond" by Chris Burniske and jack Tatar, published by McGraw-Hill education: 1st edition (Sept 2017).
4. "Blockchain Revolution: How the Technology Behind Bitcoin and Cryptocurrency is Changing the Worlds" by Dn and Alex Tapscott, published by Portfolio: reprint edition (May 10, 2016).

GE44912 Software Agent

Course Objective:

1. Knowledge about the basic concepts of intelligent agents.
2. Understanding about Mobile agents.
3. Knowledge about Agent security.
4. Understanding of Simple construction tools.

Learning Outcome:

After completing the course, the students should be able to:

1. Analyze the various interaction protocols and communication languages which leads to develop secure agent communication.
2. Acquire Knowledge about various interaction protocols and communication languages
3. Design and develop agent-based applications

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**AGENTS - OVERVIEW** Agent Definition - Agent Programming Paradigms - Agent Vs Object - Abstract and concrete Architectures for Intelligent Agents - Mobile Agents	30 hours	**1**
II	**MULTIAGENT SYSTEMS AND SOCIETIES OF AGENTS** Introduction - Agent Communications - Agent Interaction Protocols - Societies of Agents - Learning: Introduction - Learning and Activity Coordination - Learning about and from other Agents - Learning and Communication.	30 hours	**1**
III	**AGENT COMMUNICATION LANGUAGES** Agent Knowledge representation - KQML - KIF - Agent adaptability - Belief Desire Intention – BDI Architecture.	30 hours	**1**

IV	**AGENTS AND SECURITY** Agent Security Issues - Mobile Agents Security - Protecting Agents against malicious hosts - Untrusted Agent - Black Box Security - Authentication for agents - Security issues. **AGENT CONSTRUCTION** Mobile agent with java: Agent characteristics of java - Aglet model - Aglet package - Anatomy of an Agent - Agent Design Pattern: classification - Master Slave Pattern - Itinerary pattern.	30 hours	**1**

Text/Reference Books:

1. Gerhard Weiss, "Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence", MIT Press, USA, 2012.

2. Bradshaw, "Software Agents", MIT Press, USA, 2010. LTPC 30 0 3 114

3. Mitsuru Oshima, "Programming and Deploying Java Mobile Agents with Aglets", Addison-Wesley, USA, 1998

GE44913 Machine Learning

Course Objective:

1. To introduce students to the basic concepts and techniques of Machine Learning.
2. To gain experience of doing independent study and research.
3. To develop skills of using recent machine learning software for solving practical problems.
4. To become familiar with regression methods, classification methods, clustering methods.

Learning Outcome:

After completing the course, the students should be able to:

1. Gain knowledge about basic concepts of Machine Learning.
2. Identify machine learning techniques suitable for a given problem.
3. Solve the problems using various machine learning techniques.
4. Apply Dimensionality reduction techniques.
5. Design application using machine learning techniques.

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**C**oncept learning, inductive learning hypothesis, inductive bias. Information theory: entropy, mutual information, KL divergence, Necessary Conditions for Optimality, Convex function, Gradient Descent, Stable learning rates, Newtons Method, Conjugate gradient method, The Liebenberg-Marquardt algorithm.	30	1
II	Decision tree representation, appropriate problems for decision tree learning, hypothesis space search in decision tree learning, inductive bias in tree learning, avoiding overfitting the data, alternative measures for selecting attribute values, ensemble methods, bagging, boosting, random forest, Computational learning theory, probably approximately correct (PAC) learning, sample complexity and VC dimension, linear SVM, soft margin SVM, kernel functions, nonlinear SVM, Multiclass classification using SVM, Support vector regression.	30	1

III	Linear classifier, Logistic Regression, Decision Boundary, Cost Function Optimization, Multi-class Classification, Bias and Variance, L1 and L2 Regularization, feature reduction, Principal Component Analysis, Singular Value Decomposition Bayesian belief networks, gradient ascent training of Bayesian networks, learning the structure of Bayesian networks	30	1
IV	The learning task, Q learning, convergence, temporal difference learning, nondeterministic rewards and actions, generalization, relationship to dynamic programming. K-nearest neighbour learning, distance weighted neighbour learning, locally weighted regression, adaptive nearest neighbour methods, The Concept of Unsupervised Learning, Competition networks, K-means clustering algorithm.	30	1

Text and Reference Books:

2. T. Mitchell, Machine Learning, McGraw Hill,1997
3. Christopher Bishop, Pattern Recognition and Machine Learning, Springer,2006
4. K. Murphy. Machine Learning: A probabilistic perspective, MIT Press,2012
5. Hastie, Tibshirani, Friedman, Elements of statistical learning, Springer 2011
6. I. Goodfellow, Y. Bengio and A. Courville. Deep Learning. MIT Press, 2016
7. Richard S. Sutton and Andrew G. Barto, Reinforcement Learning: An Introduction, MIT Press,2018

GE44914 Real Time and Concurrent Systems

Course Objective:

1. To learn real time operating system concepts, the associated issues & Techniques
2. To understand programming languages and databases
3. To analyze real time communication
4. To analyze evaluation techniques and reliability models for Hardware Redundancy
5. To understand clock synchronization

Learning Outcome:

After completing the course, the students should be able to:

1. Apply principles of real time system design techniques to develop real time applications
2. Know the various task assignment and scheduling methods.
3. Make use of architectures and behaviour of real time operating systems.
4. Know about the RT System requirement, design, and performance analysis.

Course Contents:

Module	Course Topics	Total Hours	Credits
I	Real Time Systems Introduction: Definition Issues in Real Time Computing, Structure of a Real Time System, Task Classes. Characterizing Real Time Systems and Tasks: Introduction, Performance measures for real time systems-Traditional performance measures, Performability; Cost functions and hard Deadlines.	30 Hours	**1**
II	Task Assignment and Scheduling: Introduction to Classical Uniprocessor scheduling algorithms: Rate Monotonic, EDF algorithm, Task assignment. Clock Synchronization: Clock, A Nonfault Tolerant Synchronization Algorithm, Impact of faults, Fault Tolerant Synchronization in Hardware, Fault Tolerant Synchronization in software.	30 Hours	**1**

III	Real time Databases: Basic Definition, Real time Vs General Purpose Databases, Main Memory Databases, Transaction priorities, Transaction Aborts, Concurrency control issues, Disk Scheduling Algorithms, Two phase Approach to improve Predictability, Maintaining Serialization Consistency, Databases for Hard Real Time Systems.	30 Hours	**1**
IV	Real Time Communication: Introduction: Architectural Issues; Protocols Contention based protocols, Token based protocols, Deadlines based protocols, Stop and Go Multi-hop protocol, polled bus protocol, Hierarchical round robin protocol.	30 Hours	**1**

Text/Reference Books:

1. Rajib Mall, —Real-time systems: theory and practice‖, Pearson Education, 2009
2. Software Design for Real-time Systems: J.E. Cooling, Prentice Hall, 2003
3. Stuart Bennett, —Real Time Computer Control-An Introduction, Prentice Hall ofIndia, 1998
4. C.M. Krishna, Kang G. Shin, —Real-Time Systems, McGraw-Hill International Editions, 1997
5. Allen Burns, Andy Wellings, —Real Time Systems and Programming Languages, Pearson Education, 2003

GE44915 Hardware Architecture for Artificial Intelligence

Course Objective:

1. To learn the design of hardware architectures and accelerators for deep-learning/artificial intelligence.
2. This course is at the intersection of deep-learning and computer-architecture/embedded system/VLSI.

Learning Outcome:

After completing the course, the students should be able to:

1. Understand the key design considerations for efficient DNN processing.
2. Understand tradeoffs between various hardware architectures and platforms; learn about micro-architectural knobs such as precision, data reuse, and parallelism to architect DNN accelerators given target area-power-performance metrics.
3. Evaluate the utility of various DNN dataflow techniques for efficient processing.

Course Contents:

Module	Course Topics	Total Hours	Credits
I	Background topics: Approximate computing and storage, Roofline Model, Cache tiling (blocking), GPU architecture, CUDA programming, understanding GPU memory hierarchy, FPGA architecture, Matrix multiplication using systolic array Convolutional strategies: Direct, FFT-based, Winograd-based and Matrixmultiplication based.	30	**1**
II	Deep learning on various hardware platforms: Deep learning on FPGAs and case study of Microsoft's Brainwave, Deep learning on Embedded System (especially NVIDIA's Jetson Platform), Deep learning on Edge Devices (smartphones), Deep learning on an ASIC (especially Google's Tensor Processing Unit.), Deep-learning on CPUs	30	**1**

	and manycore processor (e.g., Xeon Phi), Memristor-based processing-in-memory accelerators for deep-learning.		
III	Memory-efficiency and reliability of DNN accelerators: Model-size aware Pruning of DNNs, Hardware architecture-aware pruning of DNNs, Understanding soft-errors. Understanding reliability of deep learning algorithms and accelerators	**30**	**1**
IV	Memory-related tradeoffs in DNN accelerators: Comparison of memory technologies (SRAM, DRAM, eDRAM, STT-RAM, PCM, Flash) and their suitability for designing memory-elements in DNN accelerator, Neural branch predictors and their applications Autonomous driving and DNN training: Hardware/system-challenges in autonomous driving, Distributed training of DNNs and addressing memory challenges in DNN training	**30**	**1**

Text/Reference Books:

1. Hennessy, J. L. ,& Patterson, D. A., Computer Architecture: A quantitative approach (Sixth Edition), Elsevier https://www.google.co.in/books/edition/Computer_Architecture/cM8mDwAAQBAJ (2017)
2. Brandon Reagen, Robert Adolf, Paul Whatmough, Gu-Yeon Wei, and David Brooks Deep Learning for Computer Architects Synthesis Lectures on Computer Architecture, August 2017, Vol. 12, No. 4, Pages 1-123 (https://doi.org/10.2200/S00783ED1V01Y201706CAC041) (2017)
3. Tor M. Aamodt, Wilson Wai Lun Fung, and Timothy G. Rogers General-Purpose Graphics Processor Architectures, Synthesis Lectures on Computer Architecture, May 2018, Vol. 13, No. 2 , Pages 1-140 (https://doi.org/10.2200/S00848ED1V01Y201804CAC044) (2018)
4. Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). Deeplearning (Vol. 1, No. 2). Cambridge: MIT press.

GE44921 Software Reliability Engineering

Course Objective:

1. To learn about the engineering techniques for developing and maintaining reliable software systems.
2. To measure the reliability of software systems.
3. To understand about fault prevention, removal and tolerance in software systems.
4. To learn about failure forecasting in software systems.
5. To learn different software reliability models and design reliability models for software system.

Learning Outcome:

1. After completing the course, the students should be able to:
2. Develop reliable software systems.
3. Understand the fault handling techniques in software system.
4. Apply the failure forecasting techniques in software systems.
5. Understand time dependent and independent reliability models.
6. Design reliability models for software systems.

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**Introduction:** Software reliability- Need and importance; Hardware Vs. Software reliability; Reliabilities metrics; Software reliability and testing; Software verification and validation. **Reliability Engineering Measure:** Fault and failure, mean time to failure, failure rate function, and reliability function for common distribution; Data collection and analysis.	30	**1**
II	**Software Reliability Modelling:** Introduction, McCabe's Cyclomatic Complexity metric, Error Seeding models, failure rate Models, Curve Fitting models,	30	**1**

	Reliability Growth Model; Bayesian Model: Little wood verral Model; NHPP reliability Model: Estimation Parameter, Application, mean time between failures for NHPP.		
III	**Prediction Analysis:** Model disagreement and Inaccuracy: short and Long term prediction, Model accuracy; Analyzing predictive accuracy: Outcomes, PLR, U & T Plot, Error and Inaccuracy. **Reliability Evaluation:** Evaluation of Network Reliability- Series systems Vs. Parallel System, Paths based and Cut based Approach, Complete event tree and Reduced event tree method with example.	**30**	**1**
IV	**Software Reliability Models With Environment Factors:** Environmental Factors, Environmental Factors Analysis, A Generalized Model With Environmental Factors, Enhanced Proportional Hazard Jelinski-Moranda, and Applications. **Fault Tolerance Software:** Basic fault-Tolerance techniques, Self-Checking Duplex scheme, Reduction of common- Cause Failures.	**30**	**1**

Text/Reference Books:

1. System Reliability Concepts by V. Sankar, Himalaya Publishing House, 2015.
2. J.D. Musa et. al- Software Reliability Engineering, TMH, New Delhi 2005.
3. H. Pham, Software Reliability, Springer Verlag, New York , 2000.

GE44922 Software Architecture and Integration

Course Objective:

1. In depth study of current software architecture research topics and middleware technologies.
2. To test software performance under given conditions without any type of corrective measure.
3. To find the number of failures occurring in a specified amount of time.
4. Students will apply their knowledge of data structures and programming to the architecture, design, and development of a class or team-sized project.
5. To find the mean life of the software.

Learning Outcome:

After completing the course, the students should be able to:

1. Students will learn middleware architecture design principles.
2. Students will understand requirements traceability and how to insure the system meets cross-cutting end-to-end software architectural properties.
3. Be familiar with a variety of architectural styles and how they may be combined in a single system.
4. Have a working knowledge of software architecture design for a non-trivial system.
5. Understand how software architecture aids different stages of the software lifecycle.

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**Software Architecture terms** Component, Relationship View, Architectural Styles, Frameworks, Patterns, Methodologies, Processes. Functional and Non-Functional Properties of Software architectures, Goals of architecture, Role of Software architect, Architectural structure and view. Many contexts of software architecture: Architecture in a technical context, Architecture in a project life cycle context.	30 Hours	1

II	**Enabling Techniques for Software Architecture** Abstraction, Encapsulation, Information Hiding, Modularization Separation of Concerns, Coupling and Cohesion, Sufficiency, Completeness and Primitiveness Separation of Policy and Implementation, Separation of Interface and Implementation, Decomposing and modularizing. Typical Architectural Design , Data Flow, Independent Components, Call and Return, Using Styles in Design, choices of styles, Architectural design space, Theory of Design Spaces , Design space of Architectural Elements, Design space of Architectural styles.	30 Hours	**1**
III	**Architectural Styles** Pipes and Filters, Data Abstraction and Object-Orientation, Event-Based, Implicit Invocation, Layered Systems, Repositories, Interpreters, Process Control, Heterogeneous Architectures. Implementing and Testing: Characteristics of architectures, fundamentals, languages, classes, coding style efficiency. Testing: Objectives, black box and white box testing, various testing strategies, Art of debugging.	30 Hours	**1**
IV	**Software Implementation-development environment facilities** Code Generation, Reverse Engineering, Profiling, Software Libraries, Testing and debugging. Software Quality**:** Changeability, Efficiency, Interoperability, Reliability, Testability, Reusability, Fault tolerant software. Software quality: SEI CMM and ISO-9001, Software reliability and fault-tolerance, software project planning, monitoring, and control. Computer-aided software engineering (CASE), Component model of software development Software reuse.	30 Hours	**1**

Text/Reference Books:

1. M. Shaw: Software Architecture Perspectives on an Emerging Discipline, Prentice-Hall, 1996.
2. "Software Architecture: Foundations, Theory and Practice" by Richard N.Taylor, Nenad Medvidovic, Eric Dashofy, ISBN:978-0-470-16774-8, 2009.
3. Len Bass, Paul Clements, Rick Kazman: Software Architecture in Practice, Pearson, 2015.

GE44923 System Simulation and Modeling

Course Objective:

1. The basic system concept and definitions of system.
2. Techniques to model and to simulate various systems the ability to analyze a system and to make use of the information to improve the performance.
3. To understand General Principles of Discrete-Event Simulation.
4. To learn how the Random Number Generate.
5. Demonstrate understanding of various Random Processes, Queuing Models and Network Simulation.

Learning Outcome:

After completing the course, the students should be able to:

1. Define basic concepts in modeling and simulation (M&S).
2. Classify various simulation models and give practical examples for each category.
3. Construct a model for a given set of data and motivate its validity.
4. Generate and test random number varieties and apply them to develop simulation models.
5. Analyze output data produced by a model and test validity of the model.

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**Introduction To System & Simulation Simulation:** Simulation as a tool, Advantages and Disadvantages of Simulation, Areas of Application systems and System Environment. Components of a System. Discrete and Continuous Systems. Model of a System, Types of Models, deterministic and stochastic systems, static and dynamic systems, continuous simulation, Monte Carlo simulation, Steps of Simulation Study.	30 Hours	1
II	**General Principles of Discrete-Event Simulation and Random Number Generation Concepts in Discrete-Event Simulation:** The Event- Scheduling / Time-Advance: Algorithm, World Views. Manual simulation, Using Event Scheduling. **Random Number Generation:** Properties of Random Numbers, Techniques for Generating Random Numbers, Generation of Pseudo-Random Numbers, Congruence generators, long period generators, statistical quality measures of generators, uniformity and independence testing, chi-square and other hypotheses testing, Tests for Random Numbers.	30 Hours	1

III	**Random Variable Generation** **Random variable**: Probability density and distribution functions, Location, scale and shape parameters, discrete and continuous probability distributions; Inversetransform method, composition and acceptance-rejection methods, efficiency and quality measures of generators; Input Modeling, selection of distribution for a random source, fitting distributions to data, constructing empirical distributions from data.	30 Hours	1
IV	**Random Processes, Queuing Models and Network Simulation** **Random process:** Discrete/continuous time processes, Markovian property, Markov chain, state transition diagrams, birth-death process, Little's theorem, steady state analysis of M/M/1 model; multi-server models, M/G/1 and other queuing models, Burke's theorem, network of queues, Jackson theorem. **Network Simulation:** SimEvent tool box in MATLAB, general features of network simulation packages, case study of OMNET++/ns2/ns3/NetSim.	30 Hours	1

Text/Reference Books:

1. Jerry Banks, John S. Carson, Barry L. Nelson, David M. Nicol, "Discrete- Event System Simulation", Fourth Edition, Prentice-Hall India, 2007.
2. Averill M. Law, W. David Kelton, "Simulation Modeling and Analysis" Third Edition, McGraw Hill, 2000.
3. Geoffrey Gordon, "System Simulation", Second Edition, Prentice-Hall India, 1979.
4. Law, A.M. and Kelton, W.D., "Simulation, Modeling and Analysis", 3rd Ed., Tata McGraw-Hill, 2003.

GE44924 Agile Software Engineering

Course Objective:

1. Understand the principles of agile development.
2. Understand a range of practices that agile software development teams can apply.
3. Understand the principles of component-based development.
4. Develop a large software artifact using .NET technologies as a member of a team using both agile project management and agile software engineering practices.
5. Demonstrate proficiency in using a range of contemporary software development tools and techniques.

Learning Outcome:

After completing the course, the students should be able to:

1. Understand and explain the range of factors that can influence the success of an agile development project.

2. Develop a large software artifact using Extreme Programming as part of a team which demonstrates good software design skills, code refactoring skills, comprehensive software testing skills and good coding standards and documentation skills.

3. Complete a project which demonstrates strong time management and agile project management skills.

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**Agile methodology and agile processes** Theories for Agile Management – Agile Software Development – Traditional Model vs. Agile Model – Classification of Agile Methods – Agile Manifesto and Principles – Agile Project Management – Agile Team Interactions – Ethics in Agile Teams – Agility in Design, Testing – Agile Documentations – Agile Drivers, Capabilities and Values Lean Production – SCRUM, Crystal, Feature Driven Development- Adaptive Software Development – Extreme Programming: Method Overview – Lifecycle – Work Products, Roles and Practices.	30 Hours	1

II	**Extreme Programming** Extreme Programming Outlined, Some Guiding Principles, The Five Values, Communication Feedback, Simplicity Courage, The 12 Basic Practices of XP 25, Test First Programming, Pair Programming, On-Site Customer, The Planning Game, System Metaphor, The Evidence for XP, Evidence for Test First, Evidence for Pair Programming	30 Hours	1
III	**Foundations: People and Teams Working Together** Observations of Team Behavior in XP Projects, Setting Up a Team, Developing Team Skills, Training Together, Finding and Keeping a Client for a University-Based Project or a Small Business Start- Up, The Organizational Framework, Planning PERT (Program Evaluation and Review Technique), Gantt Chart	30 Hours	**1**
IV	**XP Project** Starting an XP Project, The First Meetings with the Client, Business Analysis and Problem Discovery, The Initial Stages of Building a Requirements Document, Techniques for Requirements Elicitation, XP Architectures and Patterns, Finite State Machines, Extreme Modeling (XM), Multiple Stories and XXMs, Building the Architecture to Suit the Application: A Dynamic System Metaphor, Another Look at Estimation; Testing Internet Applications and Web Sites, Units and Their Tests, Writing Unit Tests in JUniti.	30 Hours	**1**

Text/Reference Books:

1. Kent Beck, "Extreme Programming Explained: Embrace Changes", Addison Wesley, 1999.

2. Jim Highsmith, "Agile Software Development Ecosystems", Addison Wesley, 2002

3. Alistair Cockburn, "Agile Software Development", Addison Wesley, 2002

GE44931 Embedded System Theory & Design

Course Objective:

1. To impart fundamental concepts in the area of Embedded Systems.
2. To impart the design an embedded system.
3. To impart the knowledge about how to partition a system into hardware and software parts efficiently.
4. To impart the Hardware/software Co-design concepts.
5. Explain the theoretical background and practical experience in the design and development of sophisticated embedded system

Learning Outcome:

After completing the course, the students should be able to:

1. Design embedded system architectures for various applications.
2. Identify, formulate, and solve engineering problems
3. Learn Function on multidisciplinary teams.
4. Apply knowledge of mathematics, science and engineering.
5. Explain applications, benefits, and limitations of networked embedded systems for environmental science, health, and safety, industrial, and consumer usage objectives.
6. Develop standard project plans for a software development team including interface definition

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**Introduction:** Introduction to embedded systems, Application Areas, Categories of embedded systems, Overview of embedded system architecture, Specialties of embedded systems, recent trends in embedded systems, Architecture of embedded systems, Hardware architecture, Software architecture, Application Software, Communication Software, Development and debugging Tools. Embedded System Design and Issues**,** Design Cycle**,** Use of Software Tools for Development of Embedded System**,** Middleware Design and Implementation for Networked Embedded Systems, Challenges and issues in embedded software development.	30 Hours	1

II	**RTOS, Microcontroller and ARM:** Data Operating System Services**:** Message Queues**,** Timer Functions**,** Events**,** Memory management**,** Interrupts Routines in an RTOS**,** Microprocessor Vs. microcontroller**,** 8051 microcontroller	30 Hours	1
III	**Embedded System Development:** Embedded System Evolution Trends**,** Round Robin**,** Robin with Interrupts**,** Function One Scheduling Architecture, Assembler, Compiler, Cross Compiler and IDE**,** Object Oriented Interfacing**,** Recursion**,** Debugging Strategies, Simulators, Wireless Sensor Networks: Introduction**,** Architectures for Wireless Sensor Networks **,** Time Synchronization Issues in Sensor Networks, Resource Aware Localization in WSN**,** Power Efficient Routing in WSN**,** Energy Efficient MAC Protocols for WSN 3.10.6 Energy Efficient MAC Protocols for WSN.	30 Hours	1
IV	**Safety and Reliability** Techniques, Proactive Approach, Software Solution, Approaches, Hardware Solutions, Approaches, Steps to a Safe Design, Extreme Reliability, Long Life Applications, Critical Components, Dealing with Failure, Specification	30 Hours	1

Text/Reference Books:

1. "Embedded Systems", Raj Kamal, TMH, 2017
2. "The 8051 Microcontroller", K. J. Ayala, Penram International, 1991
3. "Design with PIC controller", J. b. Peatman, Printice Hall, 2002
4. "Real Time Systems", H.Kopetz, Kluwer, 1997.
5. "Co-synthesis of hardware and software for embedded systems", R. Gupta, Kluwer, 1996.

GE44932 Computer Vision

Course Objective:

After learning the course, the students should be able to:

1. To implement fundamental image processing techniques required for computer vision
2. Understand Image formation process
3. To perform shape analysis
4. Extract features form Images and do analysis of Images
5. Generate 3D model from images
6. To develop applications using computer vision techniques
7. Understand video processing, motion computation and 3D vision and geometry

Learning outcome:

1. In this course students will learn basic principles of image formation
2. image processing algorithms and different algorithms for 3D reconstruction and recognition from single or multiple images (video).
3. This course emphasizes the core vision tasks of scene understanding and recognition.
4. Applications to 3D modelling, video analysis, video surveillance, object recognition and vision-based control will be discussed.

Course Contents

Module	Course Topic	otal Hrs	Credits
I	**Introduction** : Image Processing, Computer Vision and Computer Graphics , What is Computer Vision - Low-level, Mid-level, High-level , Overview of Diverse Computer Vision Applications: Document Image Analysis, Biometrics, Object Recognition, Tracking, Medical Image Analysis, Content-Based Image Retrieval, Video Data Processing, Multimedia, Virtual Reality and Augmented Reality	**30**	1
II	**Image Formation Models** : Monocular imaging system , Radiosity: The 'Physics' of Image Formation, Radiance, Irradiance, BRDF, color etc, Orthographic & Perspective Projection,• Camera model and Camera calibration, Binocular imaging systems, Multiple views geometry, Structure determination, shape from shading , Photometric Stereo, Depth from Defocus , Construction of 3D model from images **Image Processing and Feature Extraction**: Image preprocessing, Image representations (continuous and discrete) , Edge detection	**30**	1

	Motion Estimation : Regularization theory , Optical computation , Stereo Vision , Motion estimation , Structure from motion		
III	**Shape Representation and Segmentation :** Contour based representation, Region based representation, Deformable curves and surfaces , Snakes and active contours, Level set representations , Fourier and wavelet descriptors , Medial representations , Multiresolution analysis **Object recognition** : Hough transforms and other simple object recognition methods, Shape correspondence and shape matching , Principal component analysis , Shape priors for recognition **Image Understanding** : Pattern recognition methods, HMM, GMM and EM	**30**	**1**
IV	**Applications**: Photo album – Face detection – Face recognition – Eigen faces – Active appearance and 3D shape models of faces Application: Surveillance – foreground-background separation – particle filters – Chamfer matching, tracking, and occlusion – combining views from multiple cameras – human gait analysis Application: In-vehicle vision system: locating roadway – road markings – identifying road signs – locating pedestrians	**30**	**1**

Reference Books:

1. Computer Vision - A modern approach, by D. Forsyth and . Ponce, Prentice Hall Robot Vision, by B. K. P. Horn, McGraw-Hill.
2. Introductory Techniques for 3D Computer Vision, by E. Trucco and A. Verri, Publisher: Prentice Hall.
3. R. C. Gonzalez, R. E. Woods. Digital Image Processing. Addison Wesley Longman, Inc., 1992.
4. D. H. Ballard, C. M. Brown. Computer Vision. Prentice-Hall, Englewood Cliffs, 1982.
5. Richard Szeliski, Computer Vision: Algorithms and Applications (CVAA). Springer, 2010
6. Image Processing, Analysis, and Machine Vision. Sonka, Hlavac, and Boyle. Thomson.

GE44933 Software Reusability

Course Objective:

1. This course is to equip students with knowledge about software engineering advanced concepts such as software reuse, service oriented architectures,
2. Aspect oriented software engineering, etc., as well as the skills required to develop a software systems.
3. To provide students with a theoretical understanding of current best practices in software engineering
4. To provide students with practical experience to produce high-quality software with an emphasis on design quality and technical evaluation

Learning Outcome:

After completing the course, the students should be able to:

1. To explain the advantages and disadvantages of reusing software components.
2. 2. To describe the processes involved in software development with reuse and software development for reuse.
3. 3. To discuss the characteristics of reusable software components and to provide an example of a generic reusable component
4. 4. To describe methods of developing application systems so that they can be reused across a range of different computers and operating systems.
5. To introduce the notion of legacy systems and to explain why these systems are critical to some businesses.
6. To describe different approaches to software re-engineering where legacy systems are restructured into a more maintainable form.
7. To explain why data re-engineering is a particularly complex and expensive process.
8. To discuss the process of reverse engineering and explain the differences between reverse engineering and re-engineering.

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**WHAT IS SOFTWARE REUSE?** Origins of Software Reuse, Reuse and the Software Life Cycle, Software Reuse, Rapid Prototyping, and Evolving Systems, Typical Duties of Members of a Reuse Team, Reengineering and Reuse, Library Issues, Potential Disadvantages of Reuse, Legal and Contractual Issues with Software Reuse, The Current Status of Software Reuse Summary **TECHNIQUES** Domain Analysis, An Example - Domain Analysis of	**30**	**1**

	the Linux Operating System, Domain Analysis Revisited, Object-Oriented Approaches, Standard Interfaces, Designing for Reuse, Using Reuse to Drive Requirements Analysis, Metrics for Reuse **Reusability libraries** General Reuse Library Issues, Organizational Issues for Reuse Libraries, Managerial Issues for Reuse Libraries, Research Issues in Reuse Library Organization, Reuse Libraries for Ada Software, The Public Ada Library, The Ada Software Repository, The STARS and CARDS Programs, The Asset Program, Other Sources, Reuse Libraries for C++ Language Software, Reuse Libraries for C Language Software, Reuse Libraries for Higher Level Language Software		
II	**CERTIFICATION OF REUSABLE SOFTWARE COMPONENTS** The Need for Certification, The Difference Between Certification and Testing, Suggested Standards and Practices for Certification of Software Artifacts, Code Certification, Requirements Certification, Design Certification, Test Plan and Test Results Certification, Documentation Certification, System Certification, The Role of Metrics, An Overview of Software Reliability, Certification, Testing, and Reliability Modelling, Certification of Potentially Reusable Software Components is not Always Necessary_	**30**	**1**
III	**THE ECONOMICS OF SOFTWARE REUSE** Life Cycle Leverage, Cost Models for Reuse Using the Classic Waterfall Life Cycle, Reuse in the Requirements or Specification Phase, Reuse in the Design phase, Reuse in the Coding phase, Reuse in the Testing and Integration phase, Reuse in the Maintenance phase, A Cost Model for Reuse Using the Rapid Prototyping Model, A Cost Model for Reuse for a System Developed Using the Spiral Model, A Cost Model for Reuse for a System Using Only COTS, Other Reuse-Based Cost Estimation Models, Estimation of Other Resources in Reuse-based Environments, The Economic Reuse Quantity	**30**	**1**

IV	**REENGINEERING, APPLICATIONS** Program Translation, An example of semantic reasoning in reengineering, Transitioning to an Object-Oriented System, Specifications for a File System Simulation, Procedurally-based System Design, Implementation Details for a Procedurally-based Disk Simulation. Source Code for Procedural System (Optional), Reengineering a Procedurally-based System into an Object- Oriented One, An Object-Oriented Disk Simulation Program, Source Code for an Object-Oriented Solution, Comparison of Object-Oriented and Procedural Solutions **CASE STUDIES (APPLICATIONS)** Some Reuse Activities at NASA, Introduction, Methodology Used for the Collection of Metrics Data, Results, Recommendations Some Reuse Activities at AT&T, Some Reuse Activities at Battelle Laboratory, Some Reuse Activities at Hewlett-Packed, A Hypothetical Failed Software Reuse Program **TOOLS FOR SOFTWARE REUSE** The InQuisiX System of Reuse Tools, A Simple Text-Based System, The AT&T BaseWorX Application Platform, A Knowledge-Based Tool for Reuse, Issues with Network-based Tools for Software Reuse	**30**	**1**

Reference book:

1. Ian Sommerville. Software Engineering (9th Edition). Addison Wesley, 2011, ISBN-10: 0-13-703546-0. B-

Text Books:

1. Roger S. Pressman. Software Engineering A practitioner's Approach, Sixth Edition. McGraw-Hill, 2004.
2. Shari Lawrence Pfleeger, Software Engineering: Theory and Practice, 2nd Ed., Prentice-Hall, 2001.
3. J. Rambaugh, I. Jacobson, and G. Booch. The Unified Modeling Language Reference Manual. Addison-Wesley, Longman, Mass, USA, 1999.

GE44934 Formal Software Specifications

Course Objective:

1. To learn the basics of Formal Software Specifications.
2. To learn the basics of Specification in VDM-SL.
3. To learn the basics of Sets, Sequences and Stack.
4. To learn the basics of Composite Objects & Maps.
5. To learn the Implementation of Formal Software Specifications.

Learning Outcome:

After completing the course, the students should be able to:

1. An ability to understand the basics of Formal Software Specifications.
2. To analyze and understand the Implementation of Formal Software Specifications.
3. An ability to understand Composite Objects & Maps.
4. To analyze and understand the UML specifications

Course Contents:

Module	Course Topics	Total Hours	Credits
I	**Introduction to Formal Methods** Formal methods, Importance and applicability of formal methods in the development of software, Life cycle models and software development processes by incorporating formal methods. Specification, refinement, implementation, verification and validation, Classification of formal methods, Explicit vs. implicit models, executable vs. non-executable, Formal verification techniques **Propositional and Predicate Logic** Introduction to Propositions & Predicate Logic; An Introduction to Specification in VDM-SL; UML Specifications: Introduction to UML, Specifying the Operations, Specifying the State Specifying Functions, A Standard Template for VDM-SL Specifications.	30 Hours	1
II	**Formal Events** Proof of program correction, Application of Hoare logic to proof of algorithm correction, A correct programming approach by construction, Program refinement, techniques, Dafny tool.	30 Hours	1

III	**Sets, Sequences and Stack** Sets: Introduction, Sets for System Modeling, Declaring Sets in VDM-SL, Defining Sets in VDMSL; Implementing Sets: Introduction, The Collection Classes of Java, Using a Vector to Implement a Set; Sequences: Introduction, Notation, Sequence Operators, Defining a Sequence by Comprehension, Using the Sequence Type in VDM-SL. Stack: Specifying a Stack, Specifying the State of the Stack, Specifying the Operations on the Stack**.**	30 Hours	1
IV	**Composite Objects & Maps** Composite Objects: Defining Composite Object Types ,Composite Object operators, Implementing Composite Objects; Maps :Introduction and Implementation, Using the Map Type in VDM-SL.	30 Hours	1

Text/Reference Books:

1. Quentin Charatan, Aaron Kans, "Formal Software Development", Palgrave Macmillan, 2003
2. J. B. Wordsworth, "Software Development with Z", Addison Wesley, 1992)