

4.3.2. Elective Courses

CSE4206 Formal Methods in Software Engineering

COURSE INFORMATION

Prerequisites	Academic Year & Level		Teaching Methods			Credit Hrs.
	Year	Semester	Lecture	Tutorial	Laboratory	
CS1001 & CSE2001	4	7/8	2	2	0	3

COURSE AIM

- Understand the mathematical basis of formal methods.
- Understand the role of formal methods in software engineering practice.
- Understand the risks of poorly specified software systems.
- Use formal specification languages in requirement specifications.
- Apply formal reasoning and proof techniques in software verification and validation.
- Communicate software requirements and designs using a formal notation.

COURSE WEEKLY CONTENTS

- 1 Introduction to Formal Methods
- 2 Set Theory
- 3 Introduction to Z
- 4 Elements of Z
- 5 Structure
- 6 Logic
- 7 7th Week Exam
- 8 Synthesis
- 9 State
- 10 Schemas and schema Calculus
- 11 Schema Types and Binding
- 12 Formal Reasoning
- 13 12th Week Exam
- 14 Using Formal methods with UML
- 15 Object Constraint Language (OCL)

STUDENT GRADING & ASSESSMENT

Weeks	Exams	Assign.	Quizzes	Reports	Present.	Lab.	Total
1 to 7	←		30	MARKS		→	30
8 to 12	←		20	MARKS		→	20
1 to 15	←		10	MARKS		→	10
16 or 17	40	Final					40

REFERENCES

Textbook Jonathan Jacky, The Way of Z: Practical Programming with Formal Methods, Cambridge University Press.

CSE4204 Software Engineering for Large Scale Systems

COURSE INFORMATION

Prerequisites	Academic Year & Level		Teaching Methods			Credit Hrs.
	Year	Semester	Lecture	Tutorial	Laboratory	
CSE3101 & 4203	4	7/8	2	2	0	3

COURSE AIM

- Explain the challenges of engineering complex systems, explain the industrial practice of complex systems engineering, provide examples of complex systems and explain their realization, explain concepts and processes for engineering complex systems.
- Perform domain engineering, model a system from different perspectives (e.g., using feature models, UML diagrams or architecture description languages), analyze a given industrial case study for a complex system (e.g., commonality, variability analysis, model/code analysis)
- Analyze existing systems and discuss possible improvements or re-engineering potential, also taking economic aspects into account.
- Use tools and methods appropriate for large-scale development projects.
- Analyze the utility of tools and methods for large-scale development within the context of a specific development project.
- Read and analyze scientific literature.

COURSE WEEKLY CONTENTS

1	Introduction + From Traditional Engineering to Complex systems
2	Characteristics of Ultra Large-Scale Systems
3	Challenges of engineering complex systems.
4	Industrial practice of complex systems engineering.
5	Problems Inspired by Large-Scale Computing
6	Concepts (e.g., modelling or software synthesis techniques) for engineering variant-rich systems.
7	7 th week examination
8	Processes for engineering complex systems (e.g., product-line engineering, continuous integration, and deployment)
9	Business-, Architecture-, Process-, and Organization-related aspects of engineering complex systems
10	Emergent Computing
11	Computational Engineering
12	12 th week examination
13	Adaptive System Infrastructure
14	Adaptable and Predictable System Quality
15	Presentations

STUDENT GRADING & ASSESSMENT

Weeks	Exams	Assign.	Quizzes	Reports	Present.	Lab.	Total
1 to 7	←		30	MARKS		→	30
8 to 12	←		20	MARKS		→	20
1 to 15	←		10	MARKS		→	10
16 or 17	40	Final					40

REFERENCES

Textbook KOSSIAKOFF, Alexander, et al. Systems engineering: Principles and practices. John Wiley & Sons, Inc.

CSE4205 Software Product Line Engineering

COURSE INFORMATION

Prerequisites	Academic Year & Level		Teaching Methods			Credit Hrs.
	Year	Semester	Lecture	Tutorial	Laboratory	
CSE4203	4	7/8	2	2	0	3

COURSE AIM

- Explain software product line (SPL) component development and how it differs from developing normal SW component developments.
- Assess SPL implementations using evaluation frameworks for SPL assessments, in particular maturity frameworks.
- Explain how verification and validation activities in SPL software development compared to conventional software development.
- Explain the challenges with large-scale software reuse and customization and explains how SPL tries to alleviate them.
- Recognize the situations in which SPL is an appropriate approach to software development.
- Perform scoping and domain analysis.
- Analyze a SPL architecture/design.
- Use some contemporary technologies for SPLE.
- Implement a SPL from a given software project.
- Assess the maturity and potential for a company to implement or extend an SPL way of working in their software development.
- Judges, on the basis of a software development situation, and tool limitations and capabilities, whether SPL is an appropriate approach or not.

COURSE WEEKLY CONTENTS

1. The problems with large-scale software reuse and customization
2. Scoping and domain analysis
3. Design of software product-line architectures
4. Development of software product-line components
5. Development of software product-line components
6. Instantiation of family members
7. 7th week examination
8. Instantiation of family members
9. Implementation of an I/O driver and multi-threaded programming using interrupts
- 9 Verification and validation in software product line architectures
- 10 Evolution of software product-line assets
11. SEIs framework for Software Product Line Practice
12. 12th week examination
13. Assessment of company's potential for working with software product line methodologies and techniques
14. Wrap up and Conclusion
15. Presentations

STUDENT GRADING & ASSESSMENT

Weeks	Exams	Assign.	Quizzes	Reports	Present.	Lab.	Total
1 to 7	←		3 0	M A R K S		→	30
8 to 12	←		2 0	M A R K S		→	20
1 to 15	←		1 0	M A R K S		→	10
16 or 17	40	Final					40

R E F E R E N C E S

Textbook	Paul Clements and Linda M. Northrop. Software Product Lines: Practices and Patterns. Addison-Wesley Professional.
----------	---

CSE4207 Software Engineering for Cloud Applications

COURSE INFORMATION

Prerequisites	Academic Year & Level		Teaching Methods			Credit Hrs.
	Year	Semester	Lecture	Tutorial	Laboratory	
CCS3203 & CSE3202	4	7/8	2	2	0	3

COURSE AIM

- Understand what cloud computing is and why it is important.
- Get a picture of the economics of cloud computing.
- Describe Big Data and the challenges of working with it.
- Learn about many fundamental technologies that enable cloud computing, such as software defined architectures, virtualization, and containers.
- Learn about many “glue” technologies that enable access to clouds, such as web middleware, JSON, REST API, RPC, etc.
- Learn about the different levels of clouds services, which include IaaS (Infrastructure as a Service), PaaS (Platform as a Service), SaaS (Software as a Service), MaaS (Metal as a Service), FaaS (Function as a Service (server-less architecture)), MBaaS (Mobile Backend as a Service (server-less architecture)), and Amazon Lambda.
- Learn about many types of cloud-based storage services, including object storage, block-level storage, archival storage, and Big Data file systems.
- Become familiar with the key concepts underlying Big Data and data streaming applications on the Cloud.
- Describe the concerns of storage, processing, parallelism, distribution, consensus, and scalability as they relate to the Cloud.
- Understand key benefits and limitations of the various technologies available in the Cloud.
- Utilize the course content to select technologies you wish to use in your work or company.

COURSE WEEKLY CONTENTS

1	Cloud Computing Foundations and cloud models: IaaS, PaaS, SaaS, BPaaS
2	Networking, IP, HTTP, REST, VPC
3	Security & Verification (Auth, JWT, OAuth 2.0)
4	Serverless Computing
5	Non-relational Databases
6	Big Data Programming: MapReduce model, Hadoop, YARN, Spark and HDFS
7	7th Week Assessment
8	Cloud Based Analytics: Data Cube, Columnar storage, Data Lake
9	Graph Processing, Graph Databases and Machine Learning in the Cloud
10	Fast Data Processing and Streaming in the Cloud
11	Virtualization, Containers and Docker
12	Container Orchestration Part 1: Docker Swarm, ECS and ACI
13	Container Orchestration Part 2: Kubernetes
14	Architecting Cloud Applications
15	Presentations

STUDENT GRADING & ASSESSMENT

Weeks	Exams	Assign.	Quizzes	Reports	Present.	Lab.	Total
1 to 7	←		30	MARKS		→	30

8 to 12	←	20	M A R K S	→	20
1 to 15	←	10	M A R K S	→	10
16 or 17	40	Final			40

R E F E R E N C E S

Textbook	Cloud Computing: Cloud Computing Fundamentals IaaS PaaS SaaS FaaS Serverless Computing Virtualization Virtual Machine Hypervisor Docker, by SK Singh, 2022
----------	--

CSE4208 Software Engineering for Embedded Systems

COURSE INFORMATION

Prerequisites	Academic Year & Level		Teaching Methods			Credit Hrs.
	Year	Semester	Lecture	Tutorial	Laboratory	
CSE2001 & CCS2103	4	8	2	2	0	3

COURSE AIM

- Demonstrate knowledge and understanding of Embedded software engineering concepts
 - The principles of good architecture for an embedded system
 - Design practices to help make your embedded project successful
 - Details on principles that are often a part of embedded systems, including digital signal processing, safety-critical principles, and development processes
 - Techniques for setting up a performance engineering strategy for your embedded system software
 - How to develop user interfaces for embedded systems
 - Strategies for testing and deploying your embedded system, and ensuring quality development processes
 - Practical techniques for optimizing embedded software for performance, memory, and power
 - Advanced guidelines for developing multicore software for embedded systems
- Demonstrate Hands-on practice at applying concepts
- Learn how to think about embedded systems

COURSE WEEKLY CONTENTS

1	Software Engineering for embedded and real-time systems
2	Embedded System Hardware/Software co-development
3	Software Modeling for embedded systems
4	Software Design Architecture and Patterns for embedded systems
5	Real-time Building Blocks: Events and Triggers
6	Hardware's interface to embedded software
7	7 th Week Assessment
8	Embedded Software Programming and Implementation Guidelines
9	Software Reuse by Design in Embedded Systems
10	Optimizing Embedded Software for Performance, Memory, and Power
11	Human Factors and user interface design for embedded systems
12	Embedded Software Quality, Integration and Testing Techniques
13	Software Development tools for embedded systems
14	Multicore Software Development for embedded systems
15	Presentations

STUDENT GRADING & ASSESSMENT

Weeks	Exams	Assign.	Quizzes	Reports	Present.	Lab.	Total
1 to 7	←		30	MARKS		→	30
8 to 12	←		20	MARKS		→	20

1 to 15	←	10	M A R K S	→	10
16 or 17	40	Final			40

R E F E R E N C E S

Textbook	Software Engineering for Embedded Systems: Methods, Practical Techniques, and Applications (Expert Guide) 1st Edition, by Robert Oshana
-----------------	--

CSE4209 Architecting IoT Systems

COURSE INFORMATION

Prerequisites	Academic Year & Level		Teaching Methods			Credit Hrs.
	Year	Semester	Lecture	Tutorial	Laboratory	
CCS2201 & CSE4203	4	7/8	2	2	0	3

COURSE AIM

- Understand various components comprising the IoT lifecycle from an architectural perspective.
- Define the set of fully managed and integrated services that allow us to connect, manage, and ingest IoT data from globally dispersed devices at a large scale, process and analyse/visualize that data in real time, and implement operational changes easily and securely.
- Describe various hardware and software technologies to support enterprise scale IoT solutions.
- Define the end-to-end architecture of a large-scale IoT solution.
- Understand RTOS and real-world design constraints: real-time scheduling; synchronisation and communication; multithreaded design; fault-tolerant design.
- Be able to address security within their end-to-end IoT solutions.
- Determine how analytics and machine learning services can be used to enhance IoT systems.
- Understand Data Management and Access Control and Device Management for IoT solutions

COURSE WEEKLY CONTENTS

- 1 IoT architecture and the IoT lifecycle
- 2 Utilising cloud IoT services
- 3 Analysing and documenting architectural decisions for enterprise scale IoT solutions.
- 4 IoT technologies and communication protocols.
- 5 Real-world design constraints and solution options
- 6 Defining the end-to-end IoT solution architecture.
- 7 7th week examination
- 8 Architecting and documenting the end-to-end IoT solution.
- 9 Security considerations for IoT solutions
- 10 Building intelligence into IoT solutions.
- 11 IoT Data Management and Access Control
- 12 12th week examination
- 13 Architecting and documenting the end-to-end IoT solution.
- 14 Device Management for IoT solutions
- 15 Finalising and presenting the IoT solution architecture.

STUDENT GRADING & ASSESSMENT

Weeks	Exams	Assign.	Quizzes	Reports	Present.	Lab.	Total
1 to 7	←		30	MARKS		→	30
8 to 12	←		20	MARKS		→	20
1 to 15	←		10	MARKS		→	10
16 or 17	40	Final					40

REFERENCES

Textbook LEA, Perry. *Internet of Things for Architects: Architecting IoT solutions by implementing sensors, communication infrastructure, edge computing, analytics, and security*. Packt Publishing Ltd, 2018.

CSE4210 Software Engineering for AI-enabled Systems

COURSE INFORMATION

Prerequisites	Academic Year & Level		Teaching Methods			Credit Hrs.
	Year	Semester	Lecture	Tutorial	Laboratory	
CSE2001 & CAI3101	4	8	2	2	0	3

COURSE AIM

- Analyze tradeoffs for designing production systems with AI-components, analyzing various qualities beyond accuracy such as operation cost, latency, updateability, and explainability.
- Implement production-quality systems that are robust to mistakes of AI components.
- Design fault-tolerant and scalable data infrastructure for learning models, serving models, versioning, and experimentation.
- Ensure quality of the entire machine learning pipeline with test automation and other quality assurance techniques, including automated checks for data quality, data drift, feedback loops, and model quality.
- Build systems that can be tested in production and build deployment pipelines that allow careful rollouts and canary testing.
- Consider privacy, fairness, and security when building complex AI-enabled systems.
- Communicate effectively in teams with both software engineers and data analysts.

COURSE WEEKLY CONTENTS

- 1 Introduction and Motivation
- 2 Artificial Intelligence for Software Engineers
- 3 Model Quality
- 4 From Models to AI-Enabled Systems (Systems Thinking)
- 5 Goals and Success Measures for AI-Enabled Systems
- 6 Tradeoffs among Modeling Techniques
- 7 7th week examination
- 8 Risk and Planning for Mistakes
- 9 Software Architecture of AI-Enabled Systems
- 10 Data Quality and Data Programming
- 11 Managing and Processing Large Datasets
- 12 12th week examination
- 13 Intro to Ethics + Fairness
- 14 Explainability & Interpretability
- 15 Safety (Using machine learning safely in automotive software, Military, Health)

STUDENT GRADING & ASSESSMENT

Weeks	Exams	Assign.	Quizzes	Reports	Present.	Lab.	Total
1 to 7	←		30	MARKS		→	30
8 to 12	←		20	MARKS		→	20
1 to 15	←		10	MARKS		→	10
16 or 17	40	Final					40

REFERENCES

Textbook Geoff Hulten. 2019. *Building Intelligent Systems: A Guide to Machine Learning Engineering*. Apress.

CSE4211 Software Architecture for Blockchain Applications

COURSE INFORMATION

Prerequisites	Academic Year & Level		Teaching Methods			Credit Hrs.
	Year	Semester	Lecture	Tutorial	Laboratory	
CSE4203	4	7/8	2	2	0	3

COURSE AIM

- Explain the principles of blockchain and which roles it can play in an application architecture
- Decide the suitability of blockchains and how to design applications on them
- Make functional and non-functional trade-offs for blockchain-based applications
- Build small applications on blockchain

COURSE WEEKLY CONTENTS

- 1 Introduction + Blockchain Basics + Problem Definition
- 2 Blockchain Platforms + Smart Contract Basics
- 3 Software Architecture Basics + Blockchain in Software Architecture
- 4 Blockchain Taxonomy
- 5 Blockchain Application Design Process
- 6 Blockchain Suitability Evaluation
- 7 7th Week Assessment
- 8 Blockchain Application Patterns
- 9 Performance
- 10 Security + Reliability
- 11 Smart Contract Testing
- 12 Model-driven Engineering (MDE)
- 13 Privacy + Data Governance + Data Management
- 14 Real-world Projects
- 15 Projects Presentations

STUDENT GRADING & ASSESSMENT

Weeks	Exams	Assign.	Quizzes	Reports	Present.	Lab.	Total
1 to 7	←		30	MARKS		→	30
8 to 12	←		20	MARKS		→	20
1 to 15	←		10	MARKS		→	10
16 or 17	40	Final					40

REFERENCES

Textbook Xiwei Xu, Ingo Weber, Mark Staples, *Architecture for Blockchain Applications*, Springer, 2019.

CSE4302 Software Quality Assurance

COURSE INFORMATION

Prerequisites	Academic Year & Level		Teaching Methods			Credit Hrs.
	Year	Semester	Lecture	Tutorial	Laboratory	
CSE2001	4	7/8	2	2	0	3

COURSE AIM

- Identify the unique characteristics and environment of SQA.
- Identify the various causes of software errors.
- Identify the elements of a quality plan.
- Identify the major software risk items.
- Explain the need for comprehensive software quality requirements documents.
- Explain the SQA architecture that contains the components of SQA system.
- Discuss the importance of carrying out a contract review.
- Compare the major review methodologies.

COURSE WEEKLY CONTENTS

- 1 Introduction to Software Quality Assurance
- 2 Software Quality Factors, Models and Standards
- 3 Inspection: Verification and Validation
- 4 Introduction to Testing
- 5 Test Assessment
- 6 Testing Techniques
- 7 7th Week Exam
- 8 Detailed Test Planning
- 9 Integration/System Test Planning
- 10 Reachability Analysis
- 11 Structural and Mutation Testing
- 12 Software Metrics
- 13 12th Week Exam
- 14 Projects Discussion
- 15 Revision

STUDENT GRADING & ASSESSMENT

Weeks	Exams	Assign.	Quizzes	Reports	Present.	Lab.	Total
1 to 7	←		30	MARKS		→	30
8 to 12	←		20	MARKS		→	20
1 to 15	←		10	MARKS		→	10
16 or 17	40	Final					40

REFERENCES

Textbook Daniel Galin, Software Quality Assurance: From Theory to Implementation, Pearson. .

CSE4403 Agile Software Development

COURSE INFORMATION

Prerequisites	Academic Year & Level		Teaching Methods			Credit Hrs.
	Year	Semester	Lecture	Tutorial	Laboratory	
CSE2001	4	7/8	2	2	0	3

COURSE AIM

- Articulate the agile principles, practices, and roles of Scrum.
- Perform Scrum Release Planning, and Scrum Sprint Planning
- Deconstruct user stories into tasks and ideal day estimates.
- End a Sprint with Sprint Reviews and Sprint Retrospectives.
- Use Scrum with multiple, or distributed, project teams..
- Easily pass any Certified Scrum Master certification class.

COURSE WEEKLY CONTENTS

1	Introduction and background
2	Agile Manifesto and Principles
3	Overview of Scrum, and Extreme Programming
4	Feature Driven development
5	Agile project management
6	Design and development practices in Agile projects
7	7 th week examination
8	The Agile lifecycle and its impact on testing, the agile alliances, Test-Driven Development (TDD), Testing user stories - acceptance tests and scenarios.
9	Planning and managing testing cycle, Test automation, Tools to support the Agile tester, Agile testing – Nine principles and six concrete practices for testing on agile teams.
10	Market scenario and adoption of Agile, Roles in an Agile project, Agile applicability, Agile in Distributed teams
11	Business benefits, Challenges in Agile, Risks and Mitigation
12	12 th week examination
13	Agile projects on Cloud
14	Balancing Agility with Discipline
15	Agile rapid development technologies

STUDENT GRADING & ASSESSMENT

Weeks	Exams	Assign.	Quizzes	Reports	Present.	Lab.	Total
1 to 7	←		30	MARKS		→	30
8 to 12	←		20	MARKS		→	20
1 to 15	←		10	MARKS		→	10
16 or 17	40	Final					40

REFERENCES

Textbook	Agile Software Development, Principles, Patterns and Practices by Robert C. Martin Publisher: Prentice Hall
----------	---

CSE4404 Software Deployment and Operations

COURSE INFORMATION

Prerequisites	Academic Year & Level		Teaching Methods			Credit Hrs.
	Year	Semester	Lecture	Tutorial	Laboratory	
CSE3202 & CCS2305	4	7/8	2	2	0	3

COURSE AIM

- Enumerate and explain the basic theory of DevOps. Appreciate the trade-offs inherent in the design of agent-based systems.
- Explain the basic classes of DevOps tools.
- Navigate the internet to learn about tool installation and use.
- Explain the basic concepts of the cloud infrastructure and infrastructure security.

COURSE WEEKLY CONTENTS

1	Introducing DevOps Concepts
2	Working with DevOps Mindset
3	Keeping Work Organized with DevOps
4	Using DevOps to Speed Up Time to Market
5	Infrastructure as code
6	Configuration Management
7	7 th week examination
8	Virtual Machines, Containers
9	Networking
10	The Cloud
11	Container Management
12	12 th week examination
13	Deployment Pipeline
14	Microservices Architecture, Service Mesh
15	Postproduction, Disaster Recovery

STUDENT GRADING & ASSESSMENT

Weeks	Exams	Assign.	Quizzes	Reports	Present.	Lab.	Total
1 to 7	←		30	MARKS		→	30
8 to 12	←		20	MARKS		→	20
1 to 15	←		10	MARKS		→	10
16 or 17	40	Final					40

REFERENCES

Textbook Len Bass and John Klein. *Deployment and Operations for Software Engineers*.

CSE4405 Software Configuration Management

COURSE INFORMATION

Prerequisites	Academic Year & Level		Teaching Methods			Credit Hrs.
	Year	Semester	Lecture	Tutorial	Laboratory	
CSE4203	4	7/8	2	2	0	3

COURSE AIM

- Effectively Plan and Run an SCM Process considering specific organizational aspects in terms of people, product, project, cross-organizational, process, and tools.
- Understand the basic concepts of advanced machine learning algorithms.
- Effectively identify and organize configuration items to be controlled with SCM, including labeling and version control.
- Perform effective surveillance of SCM activities including software configuration auditing to evaluate the conformance of software products and processes to applicable regulations, standards, guidelines, plans, and procedures.
- Manage software baselines.
- Effectively manage changes during the software life cycle including deviations and waivers in software configuration control.
- Record and report information needed for effective management of the software configuration.
- Effectively manage software releases and deliveries including versioning control.
- Select and apply SCM Tools.

COURSE WEEKLY CONTENTS

1	Foundations of Software Configuration Management (SCM)
2	Foundations of Software Configuration Management (SCM)
3	Principles of Software Configuration Identification
4	Principles of Configuration Change Management
5	Principles of Configuration Change Management
6	Principles of Configuration Status Accounting (CSA)
7	7th week examination
8	Principles of Configuration Audits
9	Developing an Effective SCM Plan (SCMP)
10	Principles for Effective Measurements and Metrics
11	Using Effective SCM to Improve Software Development
12	12th week examination
13	The Bottom Line
14	Presentations
15	Revision

STUDENT GRADING & ASSESSMENT

Weeks	Exams	Assign.	Quizzes	Reports	Present.	Lab.	Total
1 to 7	←		30	MARKS		→	30
8 to 12	←		20	MARKS		→	20
1 to 15	←		10	MARKS		→	10
16 or 17	40	Final					40

REFERENCES

Textbook	BERCZUK, Stephen P.; BERCZUK, Steve; APPLETON, Brad. <i>Software configuration management patterns: effective teamwork, practical integration</i> . Addison-Wesley Professional, 2003.
----------	--

CCS4406 Software Maintenance and Evolution

COURSE INFORMATION

Prerequisites	Academic Year & Level		Teaching Methods			Credit Hrs.
	Year	Semester	Lecture	Tutorial	Laboratory	
CSE3202 & CSE4301	4	7/8	2	2	0	3

COURSE AIM

- Understand the difficulties of developing code in a change context as opposed to 'green field' development.
- Assess the impact of a change request to an existing product of medium size.
- Describe techniques, coding idioms and other mechanisms for implementing designs that are more maintainable.
- Understand how design patterns can improve the design of a software system.
- Refactor an existing software implementation to improve some aspect of its design.
- Identify the principal issues associated with software evolution and explain their impact on the software lifecycle.'
- Discuss the advantages and disadvantages of different types of software reuse.

COURSE WEEKLY CONTENTS

- 1 Software comprehension and concern location
- 2 Change impact analysis
- 3 Reverse engineering
- 4 Best programming practices
- 5 Coding standards
- 6 Design principles and heuristics
- 7 7th week examination
- 8 Design patterns
- 9 Refactoring
- 10 Reengineering
- 11 The laws of software evolution
- 12 12th week examination
- 13 Reuse techniques and design for reuse
- 14 Libraries vs. application frameworks
- 15 Software product lines

STUDENT GRADING & ASSESSMENT

Weeks	Exams	Assign.	Quizzes	Reports	Present.	Lab.	Total
1 to 7	←		30	MARKS		→	30
8 to 12	←		20	MARKS		→	20
1 to 15	←		10	MARKS		→	10
16 or 17	40	Final					40

REFERENCES

Textbook	Priyadarshi Tripathy and Kshirasagar Naik. <i>Software Evolution and Maintenance: A Practitioner's Approach</i> . Wiley.
----------	--

CSE4407 Software Risk Management

COURSE INFORMATION

Prerequisites	Academic Year & Level		Teaching Methods			Credit Hrs.
	Year	Semester	Lecture	Tutorial	Laboratory	
CSE2001	4	7/8	2	2	0	3

COURSE AIM

- Become familiar with the fundamentals of Risk Management in IT area, IT specifics and modern trends.
- Become familiar with some key standards.
- To be able to identify risks for the given situation with different methods.
- To perform qualitative/ quantitative risk analysis
- Understand the key outsourcing area in IT projects and key risks in each area.
- To understand the specifics of information security, key IT threats and vulnerabilities
- To know the roles and responsibilities of Risk Manager and Risk owner/Risk Responsible
- To be able to create risk reporting documentation with accordance to team needs.
- To understand the purpose and goals of IT audit
- To get an overview of available CASE tools and applications for risk management

COURSE WEEKLY CONTENTS

- 1 Risk management fundamentals. Specifics of IT industry and projects
- 2 Overview of the main standards and methodologies on RM: MSF, RUP, XP, PMBoK
- 3 Key success factors and problem areas in IT
- 4 Key risk factors: quality improvement and requirements tracking.
- 5 Risk Taxonomy/ classification. Risk factors
- 6 Risk management process: Identification
- 7 7th week examination
- 8 Qualitative assessment
- 9 Qualitative/mathematical assessment and modeling
- 10 Risk management process: risk response and control
- 11 Risks in IT outsourcing
- 12 12th week examination
- 13 Risks and information security
- 14 Organization of risk management. Team management
- 15 IT Risk management Audit

STUDENT GRADING & ASSESSMENT

Weeks	Exams	Assign.	Quizzes	Reports	Present.	Lab.	Total
1 to 7	←		30	MARKS		→	30
8 to 12	←		20	MARKS		→	20
1 to 15	←		10	MARKS		→	10
16 or 17	40	Final					40

REFERENCES

Textbook Carl L. Pritchard. Risk Management: Concepts and Guidance. Auerbach Publications .

CSE4408 Security in Software Engineering

COURSE INFORMATION

Prerequisites	Academic Year & Level		Teaching Methods			Credit Hrs.
	Year	Semester	Lecture	Tutorial	Laboratory	
CSE2001 & CCY2001	4	7/8	2	2	0	3

COURSE AIM

- Apply contemporary formal mathematical modeling techniques to model and analyze the security of a software system.
- Identify project security risks & selecting risk management strategies.
- Use statistical methods to collect and analyze metrics for assessing and improving the security of a product, process, and project objectives.
- Describe and discuss security concerns designs at multiple levels of abstraction.
- Comply with data privacy and security requirements when designing a software system.
- Design a software solution for secure access and protection of data.
- Use quality assurance activities and strategies that support early vulnerability detection and contribute to improving the development process.

COURSE WEEKLY CONTENTS

1	Course Outline and Secure SDLC
2	Security Requirements
3	Requirements: Misuse Cases
4	Planning: Risk Assessment
5	Design: Secure Design patterns
6	Design: Architectural Risk Analysis
7	7th Week Exam
8	Design: Threat Modelling
9	Implementation: Defensive Coding Practices
10	Security Testing: Penetration Testing, Risk-based Security Testing
11	Assessment: CVSS
12	12th Week Exam
13	Assessment: CWSS
14	Assessment: CVSS
15	Projects Discussions

STUDENT GRADING & ASSESSMENT

Weeks	Exams	Assign.	Quizzes	Reports	Present.	Lab.	Total
1 to 7	←		30	MARKS		→	30
8 to 12	←		20	MARKS		→	20
1 to 15	←		10	MARKS		→	10
16 or 17	40	Final					40

REFERENCES

Textbook	Micheal Howard, David Leblanc, John Viega, <i>24 Deadly Sins of Software Security: Programming Flaws and How to Fix Them</i> , McGraw-Hill Education
----------	--