|  |
| --- |
| TCS |
| Computer Science & Business Systems |
| Semester 3 Curriculum |

**FORMAL LANGUAGE & AUTOMATA THEORY (PCC-CS502)**

**Introduction:** Alphabet, languages and grammars, productions and derivation, Chomsky hierarchy of languages.

**Regular languages and finite automata:** Regular expressions and languages, deterministic finite automata (DFA) and equivalence with regular expressions, nondeterministic finite automata (NFA) and equivalence with DFA, regular grammars and equivalence with finite automata, properties of regular languages, *Kleene’s theorem,* pumping lemma for regular languages, *Myhill-Nerode theorem and its uses,* minimization of finite automata.

**Context-free languages and pushdown automata:** Context-free grammars (CFG) and languages (CFL), Chomsky and Greibach normal forms, nondeterministic pushdown automata (PDA) and equivalence with CFG, parse trees, ambiguity in CFG, pumping lemma for context-free languages, deterministic pushdown automata, closure properties of CFLs.

**Context-sensitive languages:** Context-sensitive grammars (CSG) and languages, linear bounded automata and equivalence with CSG.

**Turing machines:** The basic model for Turing machines (TM), Turing recognizable (recursively enumerable) and Turing-decidable (recursive) languages and their closure properties, variants of Turing machines, nondeterministic TMs and equivalence with deterministic TMs, unrestricted grammars and equivalence with Turing machines, TMs as enumerators.

**Undecidability:** Church-Turing thesis, universal Turing machine, the universal and diagonalization languages, reduction between languages and Rice s theorem, undecidable problems about languages.

**Basic Introduction to Complexity:**Introductory ideas on Time complexity of deterministic and nondeterministic Turing machines, P and NP, NP- completeness, Cook’s Theorem, other NP -Complete problems.

**Text Books:**

1. ***Theory of Computation – Vivek Kulkarni (My personal recommendation)***
2. *Introduction to Automata Theory, Languages, and Computation* John E. Hopcroft, Rajeev Motwani and Jeffrey D. Ullman.

**Reference Books:**

1. *Elements of the Theory of Computation,* Harry R. Lewis and Christos H. Papadimitriou.
2. *Automata and Computability,* Dexter C. Kozen*.*
3. *Introduction to the Theory of Computation,* Michael Sipser*.*
4. *Introduction to Languages and the Theory of Computation,* John Martin.
5. *Computers and Intractability: A Guide to the Theory of NP Completeness,* M. R. Garey and D. S. Johnson*.*

**SOFTWARE ENGINEERING + Lab**

**Introduction:** Programming in the small vs. programming in the large; software project failures and importance of software quality and timely availability; engineering approach to software development; role of software engineering towards successful execution of large software projects; emergence of software engineering as a discipline.

**Software Project Management:** Basic concepts of life cycle models – different models and milestones; software project planning – identification of activities and resources; concepts of feasibility study; techniques for estimation of schedule and effort; software cost estimation models and concepts of software engineering economics; techniques of software project control and reporting; introduction to measurement of software size; introduction to the concepts of risk and its mitigation; configuration management.

**Software Quality and Reliability:** Internal and external qualities; process and product quality; principles to achieve software quality; introduction to different software quality models like McCall, Boehm, FURPS / FURPS+, Dromey, ISO – 9126; introduction to Capability Maturity Models (CMM and CMMI); introduction to software reliability, reliability models and estimation.

**Software Requirements Analysis, Design and Construction:** Introduction to Software Requirements Specifications (SRS) and requirement elicitation techniques; techniques for requirement modeling – decision tables, event tables, state transition tables, Petri nets; requirements documentation through use cases; introduction to UML, introduction to software metrics and metrics based control methods; measures of code and design quality.

**Object Oriented Analysis, Design and Construction:** Concepts -- the principles of abstraction, modularity, specification, encapsulation and information hiding; concepts of abstract data type; Class Responsibility Collaborator (CRC) model; quality of design; design measurements; concepts of design patterns; Refactoring; object oriented construction principles; object oriented metrics.

**Software Testing:** Introduction to faults and failures; basic testing concepts; concepts of verification and validation; black box and white box tests; white box test coverage – code coverage, condition coverage, branch coverage; basic concepts of black-box tests – equivalence classes, boundary value tests, usage of state tables; testing use cases; transaction based testing; testing for non-functional requirements – volume, performance and efficiency; concepts of inspection.

**Laboratory**

*Development of requirements specification, function oriented design using SA/SD, object-oriented design using UML, test case design, implementation using C++ and testing. Use of appropriate CASE tools and other tools such as configuration management tools, program analysis tools in the software life cycle.*

**Text Books:**

1. ***Software Engineering*, Ian Sommerville**

**Reference Books:**

1. **Software Engineering a Practitioner’s Approach – Roger Pressman, Bruce Maxim**
2. *Fundamentals of Software Engineering,* Carlo Ghezzi, Jazayeri Mehdi, Mandrioli Dino
3. *Software Requirements and Specification: A Lexicon of Practice, Principles and Prejudices*, Michael Jackson
4. ***The Unified Development Process*, Ivar Jacobson, Grady Booch, James Rumbaugh**
5. ***Design Patterns: Elements of Object-Oriented Reusable Software*, Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides**
6. ***Software Metrics: A Rigorous and Practical Approach*, Norman E Fenton, Shari Lawrence Pfleeger**
7. *Software Engineering: Theory and Practice*, Shari Lawrence Pfleeger and Joanne M. Atlee
8. *Object-Oriented Software Construction*, Bertrand Meyer
9. *Object Oriented Software Engineering: A Use Case Driven Approach* --Ivar Jacobson
10. *Touch of Class: Learning to Program Well with Objects and Contracts* --Bertrand Meyer
11. *UML Distilled: A Brief Guide to the Standard Object Modeling Language* --Martin Fowler

**INDIAN CONSTITUTION (Non Credit)**

**(To be finalised by Respective Institute)**