

## COURSE DESCRIPTION

Dept., Number	CS487	Course Title	Software Engineering
Semester hours	3	Course Coordinator	Dr. Bogdan Korel, Associate Professor

### Current Catalog Description

Study of the principles and practices of software engineering. Topics include software quality concepts, process models, software requirements analysis, design methodologies, software testing, and software maintenance. Hands-on experience building a software system using the waterfall life cycle model. Students working in teams develop all life cycle deliverables: requirements document, specification and design documents, system code, test plan, and user manuals. Prerequisite: CS 331 or CS 401 or CS 403. (3-0-3) (T) (C)

### Textbook

R. Pressman, *Software Engineering - A Practitioner's Approach*, McGraw Hill, sixth edition, copyright 2005

### References

--

### Course Outcomes

Students should be able to:

- Understand and explain software development as a series of engineering activities, and processes.
- Demonstrate software development team-working skills.
- Analyze client/user needs.
- Select an appropriate life cycle and process model for development of a software product.
- Explain the importance of software quality evaluation activities.
- Develop a series of software life-cycle deliverables.
- Develop representations/models and descriptions of an evolving software product for inclusion in a requirements specification document.
- Build a multi-level design model and evaluate software design alternatives
- Design, execute, and log multi-level software tests.
- Describe the role that tools can play in the software life cycle.
- Communicate, verbally and in writing, the deliverables of a software development project.

## Relationship between Course Outcomes and Program Outcomes

The following Program Outcomes are supported by the above Course Outcomes:

- b. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution
- c. An ability to design, implement and evaluate a computer-based system, process, component, or program to meet desired needs
- d. An ability to function effectively on teams to accomplish a common goal
- e. An understanding of professional, ethical, legal, security, and social issues and responsibilities
- f. An ability to communicate effectively with a range of audiences
- h. Recognition of the need for, and an ability to engage in, continuing professional development
- i. An ability to use current techniques, skills, and tools necessary for computing practices.
- k. An ability to apply design and development principles in the construction of software systems of varying complexity
- l. Be prepared to enter a top-ranked graduate program in Computer Science.

## Prerequisites by Topic

Experience in developing basic programs in any computer language

Have an understanding of, and be able to apply, the essential data structures and algorithms used in computer science.

## Major Topics Covered in the Course

1. The problem statement, developer-client interactions. Overview of software engineering - life cycle models, software deliverables. 3 hours
2. Software development team concepts, team organization, team structures. Project management, the project plan. 3 hours
3. Requirements analysis, methods, models. For example, structured analysis with use of data flow diagrams, data dictionary, entity-relationship diagrams. 7 hours
4. Software specification, methods, and models. For example, structured analysis with use of process specifications, state transition diagrams. 3.5 hours
5. Preliminary design concepts, methods, and models. For example, structured analysis with use of structure charts, procedural abstractions. Concepts of top down decomposition, bottom-up composition, abstraction, coupling, cohesion, modularity, information hiding, reuse, architectural styles. 6.5 hours
6. Detailed design concepts, methods and models. For example, structured analysis with use of PDL (Program Design Language. Algorithm, and data structure design. 2.5 hours
7. Object concepts. Object-oriented analysis, nature of the approach, models. For example, 4.5 hours

Coad/Yourdon analysis model with use of class diagrams, class hierarchies, attribute, and service specifications. Role of use cases. Use of modeling languages such as UML. Object-oriented design approaches, for example Coad/Yourdon's 4-layer object-oriented design model.

8. Software implementation, transition from design to code.	1 hour
9. Software testing and evaluation. Black and white box test design strategies and related techniques, testing at multiple levels, regression test.	6.5 hours
10. Software quality, reviews, and metrics.	3 hours
11. Software maintenance and re-engineering. Types of maintenance, role of configuration management, legacy code, tool support for maintenance.	1.5 hours
12. Selected Topics	1.5 hours
Midterm Exam	1.5 hours
Final Exam	-
	45 hours

#### Assessment Plan for the Course

End of every semester Course Objective Assessments by CS department. End of semester Course Evaluations by IIT. Reviewed every Spring semester by CS Undergraduate Studies Committee for possible updates in the following Fall. Once every 4-5 years a detailed review of all materials for the course is made by the CS Undergraduate Studies Committee.

How Data in the Course is Used to Assess Program Outcomes (unless adequately covered already in the assessment discussion under Criterion 4)

See the assessment discussion under Criterion 4

#### *For a computer science program*

Estimate Curriculum Category Content (Semester hours)

Area	Core	Advanced	Area	Core	Advanced
Algorithms			Software design		3
Data structures			Concepts of programming languages		