Course Title	**Fundamental of Software Engineering**		
Operation Period	**2021 –Semester- I**	**Course Credits**	3
Class Schedule		**Code**	CSE 3205

Target Students' Major	CSE	**Target Grade**	3RD Year
Prerequisite(s) for enrollment	None	**Capacity** (Maximum Number)	30

Instructor Information	**Name**		**Office Hour**	
	Mobile		**E-Mail**	
TA	**Name**		**E-Mail**	
Course Team or SIG			**Contact person**	
		Focus area list	**Weekly programs**	

Learning outcome	Upon successful completion of the course the student will be able to: • Describe the history of the term, "software engineering," and explain its current meaning and importance. • Explain well-known software development process models. • Select, with justification, a software development process which is most appropriate for the development and maintenance of a diverse range of software products. • Use a common, semi-formal method (for example, UML diagrams) to specify the requirements of a moderately sized software product. • Conduct software design using an accepted program design methodology such as UML. • Distinguish between different types and levels of testing (for instance, unit, integration, systems, and acceptance) for medium-size software products. • Discuss various testing techniques such as white box and black box testing. • Discuss key principles and common methods for software project management such as scheduling, size estimation, cost estimation and risk analysis.

	• Get familiar with CASE tools and/or environments including UML drawing tools and IDEs. • Make presentations describing aspects of software development activities. • Discuss key concepts and common type of software project maintenance
Course Description	Software engineering is an engineered discipline in which the aim is the production of software products, delivered on time and within a set budget, that satisfies the client's needs. It covers all aspects of software production ranging from the early stage of product concept to design and implementation to post-delivery maintenance. This course introduces the major concepts and techniques of software engineering so that students can prepare for their future careers as s*oftware engineers*. Moreover, through group projects, students can obtain hands-on experiences on entire phases and workflow of the software
Related research areas	**Advanced Software Modularity:** • aspect-oriented requirements engineering; • requirements and architecture design techniques for software product line engineering; • Requirements engineering for service-oriented systems.
Major topics	**Chapter One: Introduction of software engineering** Definition of software engineering, Why software engineering?, major software engineering activities, **Chapter Two: Software processes models** Advantage ,disadvantage and when to use the process models like waterfall, spiral, Agile , unified, iterative ,etc **Chapter three: Requirements Engineering** Functional and non-functional requirements, Requirements specification, Requirements engineering processes, Requirements elicitation and analysis , Requirements validation ,Requirements management **Chapter Four: Software Design and Architecture Concepts** Object oriented design ,Software design models, design quality attribute ,Architectural design decisions, Architectural views **Chapter Five: Coding and Testing** Coding principles, standard coding practice, unit testing , integration testing , system testing , acceptance testing, testing models **Chapter Six: Software Project Management**

<table>
<tr><td></td><td></td><td colspan="3">Project planning , Project scheduling, Risk management , Managing people , Software cost estimation
Chapter Seven: Software Maintenance

Change and maintenance ,factors of maintenance cost ,Type of maintenance , maintenance model</td></tr>
<tr><td></td><td rowspan="8">Assessment</td><td>Parameter</td><td>Weight</td><td>Remark</td></tr>
<tr><td>Quiz</td><td>10%</td><td rowspan="7">Course instructors may change the weight and assessment types</td></tr>
<tr><td>Assignment / Presentation</td><td>10%</td></tr>
<tr><td>Project</td><td>15%</td></tr>
<tr><td>Mid exam</td><td>20%</td></tr>
<tr><td>Lab</td><td>10%</td></tr>
<tr><td>Final exam</td><td>35%</td></tr>
<tr><td>Total</td><td>100 %</td></tr>
<tr><td></td><td>Course Textbook</td><td colspan="3">• Software Engineering, 9th Edition, by Ian Sommerville, 2011, Addison Wesley.</td></tr>
<tr><td></td><td>Related References</td><td colspan="3">• Software Engineering, 9th Edition, by Ian Sommerville, 2011, Addison Wesley.
• Sams Teach Yourself UML in 24 Hours, 3rd edition, by Joseph Schmuller, 2009,SAMS.
• Software Modeling and Design, Hassan Gomaa, George Mason University, Fairfax, Virginia, Cambridge University Press,2011
• Fundamental Software Engineering by Rajib Mall 2nd ed, Prentice Hall, india,2004
• A. Behforooz and F. J. Hudson (1996), Software Engineering Fundamentals, Oxford UniversityPress.
• Schach, Stephen R. (2002), Classical and Object-Oriented Software Engineering, 5th ed. IRWIK
• Hoffer, Jeffrey A.; Joey F. George; and Joseph S. Valacicli (1999), Modern Systems Analysis and Design. Massachusetts: Addison-Weslev.N.E. Fenton and S.L. Pfleeger (2001), Software Metrics: a regorous & practical approach, 2nd Edition.</td></tr>
</table>