CSE 2410 Introduction to Software Engineering

Primary instructor: Keith Gallagher **Supporting faculty:** Cem Kaner

Textbooks and references:

Vaclav Rajlich, Software Engineering: The Current Practice. Chapman and Hall/CRC, first edition, 2011. ISBN-13: 978-1439841228. (T)

Course information:

2014–2015 Catalog description: CSE 2410 Introduction to Software Engineering (3 credits). Presents a basis for the integration of engineering rigor and software development. Students are shown a practical yet rigorous method of going from a problem concept to a software solution. Includes requirements specification, functional specification and coding techniques using information hiding and stepwise refinement. Prerequisites: CSE 2010 or ECE 2552.

Prerequisites by topic: Fluency in a higher level programming language, knowledge of data structures and algorithms

Place in program: Required. Prerequisite for: CSE 3421 Software Design, CSE 4415 Software Testing 2, CSE 4610 Requirements Engineering, CSE 4621 Software Metrics and Modelling.

Course outcomes & related student outcomes: The student will be able to

1. Understand process improvement and learning from previous mistakes. (2: Scientific, computing, and engineering problem solving)
2. Identify appropriate life-cycle models for software engineering as well as their pros and cons. (1: Fundamental knowledge & 2: Scientific, computing, and engineering problem solving)
3. Understand project planning and management including effort estimation, risk analysis and personnel. (1: Fundamental knowledge & 2: Scientific, computing, and engineering problem solving)
4. Use design notations to analyze and design systems. (1: Fundamental knowledge)
5. Describe design architectures and the necessity for information hiding. (1: Fundamental knowledge)
6. Employ different strategies for testing software. (2: Scientific, computing, and engineering problem solving)
7. Practice software engineering with a small group project. (8: Effective teamwork)

Topics covered:

1. The engineering process (2 hours)

2. Feasibility, requirements, life-cycles (2 hours)
3. Estimation, project management, and risk analysis (3 hours)
4. Design I – abstraction, information hiding, coupling, and cohesion (3 hours)
5. Design II – formal methods and UML (3 hours)
6. Implementation coding standards, stepwise refinement, reviews (3 hours)
7. Testing I – testing methods (2 hours)
8. Testing II – integration, metrics (2 hours)
9. Quality – CCM, ISO9001, Agile (2 hours)
10. Maintenance – problem reporting, debugging, and fault analysis (2 hours)
11. Management – problem solving, teams, staffing, and motivation (2 hours)
12. Software tools – the future of software engineering and of being a good developer (2 hours)
13. Project reviews (6 hours)
14. Revisions (3 hours)

Approved by: Keith Gallagher, Associate Professor, Director of Software Engineering Programs & Cem Kaner, Professor

Signature: Keith Gallagher **Date:** 2 FEB 15

Signature: Cem Kaner **Date:** 2/2/15