CSC 701 Software Engineering **3 cr.**

Catalog description:

This course is devoted to the theory and practice of software engineering. It will explore state-of-practice and cutting-edge techniques and tools related to the specification, design, management, implementation, maintenance and evolution of software systems. Topics include: design patterns; Model Driven Architecture (MDA); test-driven development; agile development; design and implementation for reusability and maintainability; secure coding; evolution of support tools and environments. An ongoing group project will be used to gain practical experience with current software engineering practices and a variety of IDEs and CASE tools. Extensive reading and reporting on advanced topics in software engineering discipline are required. Three lecture hours per week, plus project work outside of class.

Course Prerequisite: graduate status or permission of graduate program coordinator.

Course Goals:

The purpose of this course is to develop an understanding of modern methodologies, processes and techniques encountered in the development of large-scale software systems. The goals of this course are:

- **CG01:** to give students experience with a variety of software engineering techniques and paradigms;
- **CG02:** to expand and integrate students' knowledge and skills in the areas of system analysis and software design, implementation and verification;
- **CG03:** to give students experience in making and critiquing software development presentations;
- **CG04:** to give students experience in team software development.

Course Objectives:

Upon successful completion of the course, student will have:

- **CO01:** demonstrated understanding of the software development process;
- **CO02:** demonstrated knowledge of the major techniques and models used in the implementation of each workflow of software development;
- **CO03:** gained experience with the tools and techniques of software development;
- **CO04:** demonstrated understanding of modern design paradigms;
- **CO05:** properly utilized modern CASE tool environments, specifically including UML modeling, in the design and implementation of a large-scale project;
- **CO06:** participated in the development and presentation of individual and group projects.
- **CO07:** gained knowledge by reading and synthesizing research articles from the leading journals in the software engineering discipline.

Topics Agenda:

Week01: Review of software process models, software development life cycle and its aspects and phases, software development tools and techniques; current definitions of software quality and software maintenance.
Week02: Software requirements: elicitation, analysis.
Week03: Requirements: modeling, validation.
Week04: Object-Oriented analysis: entity modeling, functional modeling, and dynamic modeling.
Week05: Object-Oriented analysis: interface class extraction, control class extraction.
Week06: Structural analysis: informal specifications, specification documents, structured system analysis.
Week07: Structural analysis: UML and ER modeling, finite state machines, formal techniques.
Week08: Midterm course review, midterm exam.
Week09: Software design: workflow, test workflow in the context of design.
Week10: Software design patterns: creational, structural and behavioral patterns.
Week11: Software implementation: workflow, languages, platforms.
Week12: Software implementation: coding standards, code reuse, maintenance-aware implementation.
Week13: Software test workflow in the context of implementation; testing techniques.
Week14: Software maintenance, Post software delivery.
Week15: Term paper or project presentations, reviews, final exam.

Testing and Grading:

The course consists of lectures, homework assignments, a midterm examination, a final examination, a term paper and a semester project. Homework assignments will be given every two to three weeks, covering different topics from the course. One or more advanced topics within the course content will be chosen for students to write term papers. Students will also be asked to review some latest research papers on some advanced topics. Writing a term paper will help students to enhance their knowledge of an advanced topic. It will help students to learn the art of reading, understanding and writing research papers. Students will complete a reasonable large software development project that requires the application of the knowledge and skills taught in the course, as well as providing an opportunity to acquire new skills and knowledge. Students will also gain extensive experience of many of the software development tools and environments used in the software development industry.

The final grade will be determined using the following approximate weights:

- Written homework assignments 15%
- Midterm examination 15%
- Final examination 20%
- Semester Project 30%
- Term paper and presentation 20%

Bibliography:

Booch, Grady; Rumbaugh, James; Jacobson, Ivar. **The Unified Modeling Language User Guide. Second Edition.** Addison-Wesley, 2005.

Brooks, Frederick. **The Mythical Man Month, 20th anniversary edition (aka 2nd ed.).** Addison-Wesley, 1995.

Bruegge, Bernd; Duto1t, Allen. **Object-Oriented Software Engineering: Using UML, Patterns and Java. Third Edition.** Prentice Hall, 2009.

Coplien, James; Harrison, Neil. **Organizational Patterns of Agile Software Development**. Prentice Hall, 2005.

Dennis, Alan; Wixom, Barabar; Tegarden, David. **Systems Analysis and Design with UML Version 2.0 : An Object-Oriented Approach . Third Edition.** John Wiley & Sons, 2007.

Dikel, David M.; Kane, David; Wilson, James R. **Software Architecture: Organizational Principles and Patterns.** Prentice Hall, 2001.

Fowler, Martin. **Analysis Patterns: Reusable Object Models.** Addison-Wesley, 1997.

Fowler, Martin; Beck, Kent. **Refactoring: Improving the Design of Existing Code. Addison-Wesley Professional, 1999**

Fowler, Martin, with Kenneth Scott. **UML Distilled: A Brief Guide to the Standard Object Modeling Language. Third Edition.** Addison-Wesley, 2004.

Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John. **Design Patterns: Elements of Reusable Object-Oriented Software**. Addison-Wesley, 1995.

Ghezzi, Carlo; Jazayeri, Mehdi; Mandrioli, Dino. **Fundamentals of Software Engineering. Second Edition**. Prentice Hall, 2002

Hoffer, et. al. **Modern Systems Analysis & Design. Sixth Edition.** Prentice Hall, 2010.

Kerievsky, Joshua. **Refactoring to Patterns .** Addison-Wesley Professional, 2004.

Larman, Craig. **Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development.** Pearson Education, 2005.

Oram & Wilson, editors. **Beautiful Code.** O'Reilly, 2007.

Pfleeger, Shari Lawrence; Atlee, Joanne. **Software Engineering: Theory and Practice. Fourth Edition.** Prentice Hall, 2009.

Pressman, Roger S. **Software Engineering: A Practitioner's Approach. Eighth Edition.** McGraw-Hill, 2014.

Sommerville, Ian. **Software Engineering. Ninth Edition.** Addison-Wesley, 2010

Schach, Stephen R. **Classical and Object-Oriented Software Engineering. Eighth Edition.** McGraw-Hill, 2010.

Shalloway, Alan; Trott, James. **Design Patterns Explained: A New Perspective on Object-Oriented Design. Second Edition.** Addison-Wesley, 2004.

Sonmez, John. **Soft Skills: The software developer's life manual.** Manning Publications, 2014.

Vliet, Hans van. **Software Engineering: Principles and Practice**. **Third Edition**. John Wiley & Sons Ltd. 2008

Zobel, Justin. **Writing for Computer Science. Second Edition**. Springer, 2004.