Texas Tech University
Department of Computer Science

Course Name: Software Engineering I **Number:** CS3365 **Semester:** Fall 2016
Class room: BIOL 102 **Class Hours:** 8.00-9:20 (Tuesday/Thursday)

Instructor: Sepideh Ghanavati **Office:** EC306A **Email:** sepideh.ghanavati@ttu.edu
Instructor Office Hours: Tuesday 10:00am – 11:00am, Thursday 12:00 – 1:00 PM or by email

TA: Moitrayee Chatterjee **TA-Office:** EC202A **TA-Email:** moitrayee.chatterjee@ttu.edu
TA-Office Hours: Monday and Thursday 10:00 am – 11:00 am or by email.

Catalogue Listing: Introduces theory and practice for software engineering. Topics include software life cycle, requirements, specification and analysis, software architecture and detailed design, and testing. (Writing intensive)

Text (required): A required reading list is provided at the end of the document.

Course objectives:

The purpose of this course is to introduce theories, methods, and tools in software engineering for developing software systems. Students who succeed in this course will:
1. Understand basic principles of Software Engineering
2. Be able to practice software engineering techniques
3. Be able to model with the Unified Modeling Language (UML)

Key Topics:

1. Software Development Lifecycle
2. Requirements Elicitation
3. Requirements Analysis
4. Architectural Design
5. Design Patterns
6. Detailed Design
7. Introduction to Verification and Validation/Testing

Writing Intensive:

CS3365 is one of the two required writing intensive courses for the Department of Computer Science. According to the University, "the fundamental objective of a writing intensive course is for students to write often and receive critical review from the course instructor. Students should be required to rewrite, based on the instructor's critique. The writing intensive course emphasizes the process as well as the products of writing. Faculty use writing to reinforce student learning. Students' writing should formulate ideas, raise questions, and express considered opinions. Students' written work should analyze, integrate, and synthesize as well as communicate."

Course Prerequisites: CS2365 (Object-Oriented Programming) or CS2413 (Data Structures), MATH3342 (Mathematical Statistics for Engineers and Scientists), or equivalent.

Expected prior knowledge and skills in: The successful student should have competent skills in procedural and object-oriented programming, knowledge of data structures and algorithm analysis, and knowledge of statistical and probabilistic mathematics.

Learning Outcomes & Assessment Methods: Students who have completed this course should have the ability to:

Objectives	ABET Outcomes	Assessment Methods
1. Ability to elicit and analyze customer requirements	b, f	P, Q, A, E
2. Ability to design software systems using modeling techniques.	c	P, Q, A, E
3. Understanding of verification and validation techniques.	c	P, Q, A, E
4. Professionalism and ethics.	d, e	P, Q, A, E
5. Understanding the use of software engineering tools, templates, and references.	h	P, Q, A, E

Grading Policy: The final grade for this course will be based on participation, projects, and exams, as described below:

- (P)roject: **40% (10% each)**
- (Q)uizzes or (A)ssignments: **20% (Quizzes and short assignments)**
  - 10 Quizzes - Each quiz has 1 or 2 marks – in total you can get 12/10 marks (2 bonus marks)
  - 6 Short assignments – Each have 2 marks – in total you can get 12/10 marks (2 bonus marks)
- (E)xams: **40% (20% each)**

Please note the following:

- The usual grading scale will be used: A (90-100), B (80-89), C (70-79), D (60-69), F (0-59). This scale may be curved to raise student grades at the instructor's discretion.
- Submitted work is due when specified. With the instructor permission, you may be able to submit 1-2 days late (with a 10-20% penalty). Turn your assignment in early if possible particularly if you know you will be absent.
- In-Class assignments are given during the class lecture period and may be discussions, assignments, presentations, quizzes, or other activities. No make-up in-class quizzes are given. However, assignments can be handed in **at the beginning** of the following session.
- The team-based project will serve to take students through the development of a software system using software engineering principles.

Ethical Conduct:
Although students are encouraged to discuss ideas and problems with the TA, instructor, and other students, academic dishonesty will not be tolerated. Unless stated otherwise by the instructor, you are not allowed to share code or answers, use or even look at code or answers obtained from online sources, friends, or classmates. **It is your responsibility to educate yourself about actions that constitute academic dishonesty**. If you are not sure whether a specific action is allowed, talk to the instructor and the TA before you indulge in it. All submitted code and assignments will be randomly checked for plagiarism. Academic dishonesty of any kind, if discovered, will result in one or more of the following sanctions: a grade of 0 for the corresponding graded item, a grade of "F" in the course, and further action according to the TTU operating procedures: http://www.depts.ttu.edu/opmanual/OP34.12.pdf.

Classroom Civility:
All violations of classroom civility will be reported to the Student Judicial Programs. The Texas Tech University Catalog states: "Students are expected to assist in maintaining a classroom environment that is conducive to learning." In order to ensure that all students gain from time spent in class, **students are prohibited from engaging in any form of distraction**, e.g., reading newspapers (or other articles), working on other courses, and using cell-phones or laptops for calls or messages. If you indulge in any such inappropriate behavior (without explicit consent of the instructor), you will (at the very least) be asked to leave the classroom.

Student with Disabilities:
Any student who, because of a disability, may require special arrangements in order to meet the course requirements should contact the instructor as soon as possible to make any necessary arrangements. Students should present appropriate verification from Student Disability Services during the instructor's office hours. Please note instructors are not allowed to provide classroom accommodations to a student until appropriate verification from Student Disability Services has been provided. For additional information you may contact the Student Disability Services office in 335 West Hall or 806-742-2405.

Center for Campus Life:
The Center for Campus Life can assist in notifying the campus community of student illnesses, immediate family deaths and/or student death. Generally, in cases of student illness or immediate family deaths, the notification to the appropriate campus community members occur when a student is absent from class for four consecutive days with appropriate verification. It is the student's responsibility for missed class assignments and/or course work during their absence.

Class Attendance:
The student is responsible to inform the instructor, ahead of time if possible, of any absence and the reason. Make-up work due to absence(s) may be allowed on a case-by-case basis with a possible penalty only with instructor permission and with reference to TTU operating procedures. Make-up work should be submitted preferably before the next class period after the absence(s).

- Student Absence for Observance of Religious Holy Day, http://www.depts.ttu.edu/opmanual/OP34.19.pdf
- Sponsorship of Student Activities and Off-campus Trips, http://www.depts.ttu.edu/opmanual/OP34.06.pdf
- Class Attendance, http://www.depts.ttu.edu/opmanual/OP34.04.pdf

Course Schedule: The table (below) provides the initial distribution of topics discussed over the weeks in the semester. **This schedule is tentative and subject to change during the semester at the instruction discretion**. All changes will be announced in class or on the course website (Blackboard). Students are responsible for making sure they are informed about announcements.

Class (T-T)	Activity	Material
Week 1 (08/30, 09/01)	Lectures	Syllabus – Introduction to Software Engineering Software Process
Week 2 (09/06, 09/08)	Lectures	Software Process – Agile Software Development Software Requirements – Requirements Engineering
Week 3 (09/13, 09/15)	Lectures	Software Requirements Elicitation Use Case Modeling
Week 4 (09/20, 09/22)	Lectures	Use Case Modeling, Domain Modeling
Week 5 (09/27, 09/29)	**Project Presentations**	
Week 6 (10/04, 10/06)	Lectures	Domain Modeling Object Interaction Modeling
Week 7 (10/11, 10/13)	Lectures	Software Architecture – Architectural Design Design Patterns
Week 8 (10/18, 10/20)	**Project Presentations**	
Week 9 (10/25, 10/27)	Lectures **Mid-Term Exam**	Mid-Term Review **Mid-Term Exam over Software Process, Software Requirements, Use Case Modeling, Domain Modeling, Object Interaction Modeling, Software Architecture**
Week 10 (11/01, 11/03)	Lectures	Detailed Design
Week 11 (11/08, 11/10)	Lectures	Software Testing
Week 12 (11/15, 11/17)	**Project Presentations**	
Week 13 (11/22, -)	Lectures	Software Testing
Week 14 (11/29, 12/01)	**Project Presentations**	
Week 15 (12/06, -)	Lecture	Final Review
December 2016	**Final Exam**	**Final Exam over Software Process, Software Requirements, Use Case Modeling, Domain Modeling, Object Interaction Modeling, Software Architecture, Design Patterns, Detailed Design, Software Testing**

Reading Materials:

Required Textbook:

- Software Engineering – Ian Sommerville – 10th Edition, 2016
    - Chapters to read will be mentioned every week, under the mandatory part.

Optional Textbook:

- Object-oriented Software Engineering – An Agile Unified Methodology – David C. Kung, 2014
    - Chapters to read will be mentioned every week, under the optional part.

Recommended Readings:

- UML Distilled: A Brief Guide to the Standard Object Modeling Language – M. Fowler – 3rd Edition
- Design patterns: Elements of reusable object-oriented software – E. Gamma, R. Helm, R. Johnson, J. Vlissides, 1994

Additional Readings:

- Object Oriented Software Engineering: Practical Software Development Using UML and Java – T. C. Lethbridge and R. Laganière – 2nd Edition, 2004
- Object-Oriented Design with UML and Java – Kenneth Barclay & John Savage– 1st Edition, 2004
- Object-Oriented Analysis and Design for Information Systems: Modeling with UML, OCL, and IFML – Raul Sidnei Wazlawick, 2014
- An Integrated Approach to Software Engineering – P. Jalote – 3rd Edition, 2006
- A Concise Introduction to Software Engineering – P. Jalote, 2008
- Software Engineering: A Lifecycle Approach – Pratap K.J. Mohapatra, 2010
- Software Engineering: A Hands-On Approach – Roger Y. Lee, 2013
- Software Engineering: A Methodical Approach – E. Foster, 2014
- An Introduction to Requirements Engineering – I.K. Bray, 2002
- Model-Driven Software Engineering in Practice – M. Brambilla, J. Cabot, M. Wimmer, 2013
- Continuous Integration: Improving Software Quality and Reducing Risk – P.M. Duvall, S. Matyas, A. Glover, 2007
- Software Testing – R. Patton – 2nd Edition, 2005
- Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation – J. Humble, D. Farley, 2010
- Code Complete: A Practical Handbook of Software Construction – S. McConnell – 2nd Edition, Microsoft Press, 2004
- Clean Code: A Handbook of Agile Software Craftsmanship – Robert C. Martin – 1st Edition, 2008

Web Resources (in alphabetical order):

- Android Studio IDE: https://developer.android.com/sdk/index.html
- Eclipse IDE: http://www.eclipse.org/
- Git: http://www.git-scm.com/
- GitHub: https://github.com/
- Umple: http://cruise.eecs.uottawa.ca/umple/

Code Analysis and Debugging Tools:

- Android Lint – helps identifying and correcting problems with structural quality of code: https://developer.android.com/studio/write/lint.html or http://tools.android.com/tips/lint
- Find Bugs – uses static analysis to find bugs in Java code: http://findbugs.sourceforge.net/
- SonarQube – manages code quality: http://www.sonarqube.org/
- CheckStyle – help checking Java code to adhere to coding standards: http://checkstyle.sourceforge.net/
- PMD – source code analyzer: http://pmd.github.io/

Software Development Resources:

- Java and Android Development
  - Effective Java – J. Bloch – 2nd Edition, 2008
  - Android Programming: The Big Nerd Ranch Guide – B. Phillips, Ch. Stewart, B. Hardy, K. Marsicano – 2nd Edition, 2015
  - Building Your First App: https://developer.android.com/training/basics/firstapp/index.html
  - Android application development: a beginner's tutorial – Budi Kurniawan, 2015
  - Android application development – Budi Kurniawan, 2015
  - Java for Android – Budi Kurniawan, 2014
  - Android Programming for Beginners – John Horton, 2015
  - Learning Java by Building Android Games – John Horton, 2015
  - Learn Java for Android Development – Jeff Friesen, 2014
  - Learning Android Intents – M. Usama bin Aftab, 2014
  - Android Design Patterns: Interaction Design Solutions for Developers – Greg Nudelman, 2013
  - Android Studio Application Development – Belen Cruz Zapata, 2013
  - Head First Android Development – Dawn Griffiths and David Griffiths O'Reilly Media, 2015
  - Mastering Android Application Development – Antonio Pachon Ruiz, 2015
  - Android Studio Essentials – Belen Cruz Zapata, 2015
  - Beginning Android 3D Game Development – Robert Chin, 2014
  - Pro Android UI – Wallace Jackson, 2014
  - Android Fragments – Dave MacLean, Satya Komatineni, 2014

- Git/Github:
  - Git: Distributed Version Control—Fundamentals and Workflows – Rene PreiBel & Bjorn Stachmann, 2014
  - Git: Version Control for Everyone – Ravishankar Somasundaram, 2013
  - Jump Start Git – Shaumik Daityari, 2015
  - Git for Teams – Emma Jane Hogbin Westby, 2015
  - Pro Git – Scott Chacon, Ben Straub, 2014
  - Git Recipes – Wlodzimierz Gajda, 2013

- PHP, MySQL, Web Services
  - Web Programming with PHP and MySQL – Max Bramer, 2015
  - PHP Web Services – Lorna Jane Mitchell – 2nd edition, 2016
  - PHP for Absolute Beginners – Thomas Blom Hansen, Jason Lengstorf, 2014
  - PHP Solutions – David Powers, 2014.
  - Learn PHP 7 – Steve Prettyman, 2016.