

|  |
| --- |
| **Course Outline**  Department of Computing Science  Faculty of Science |

**COMP 3520 – 3**

**Software engineering (3,1,0)**

**Winter, 2018**

**Instructor:** Kevin O’Neil Phone/Voice Mail: 250-371-5789 E-Mail:koneil@tru.ca

**Calendar /Course Description**

Students are introduced to the different software process models and management of modular inter-communication, software engineering tools, software testing and project management including resource estimation, team organization and review. Students learn software engineering techniques for dependable and secure systems, reliability engineering, software evolution, software maintenance, quality management, configuration management, reuse and ethical issues in software engineering.

**Course/Learning Outcomes**

Upon successful completion of the course, the student will demonstrate the ability to:

1. Explain the different practices that are key components of various process models.
2. Apply the basic principles of software project management in a team environment.
3. Understand a variety of strategies to the testing of simple programs.
4. Identify the principal issues associated with software evolution and explain their impact on the software lifecycle.
5. Identify methods that will lead to the creation of a software architecture that achieves a specified level of reliability, dependability and security.
6. Demonstrate the understanding of ethical issues in software development

**Prerequisites**

COMP 2920 Software Architecture and Design

**Recommended Texts/Materials**

Text Book: Sommerville Ian, *Software Engineering*, 10th Edition, Addison Wesley; ISBN-10: 0133943038

**Student Evaluation**

Projects and Assignments/Quiz (Including marks for participation in class) 25%

Midterm 25%

Final Exam 50%

**Syllabus**

|  |  |  |
| --- | --- | --- |
| Topic # | Chapter Title | Chapter # |
| 1 | Software processes | 2 |
| 2 | Agile software development | 3 |
| 3 | Software Testing | 8 |
| 4 | Software Evolution | 9 |
| 5 | Dependability and security | 10 |
| 6 | Security Engineering | 13 |
| 7 | Resilience engineering | 14 |
| 8 | Software Reuse | 15 |
| 9 | Project management | 22 |
| 10 | Project planning | 23 |
| 11 | Quality Management | 24 |
| 12 | Configuration management | 25 |
| 13 | Computer Reliability and Ethical Issues | Instructor notes |
| 14 | Professional Ethics and The ACM Code SE code | Instructor notes |

**Use of Technology**

BlackBoard and or Moodle may be used to assist in course delivery. Students are expected to learn to obtain instructional material and assignments electronically.

**ACM / IEEE Knowledge Area Coverage**

**IEEE Knowledge Areas that contain topics and learning outcomes covered in the course**

|  |  |
| --- | --- |
| **Knowledge Area** | **Total Hours of Coverage** |
| SE/Software Processes | 3 |
| SE/Software Project Management | 2 |
| SE/Tools and Environments | 1 |
| SE/Software Verification and Validation | 4 |
| SE/Software Evolution | 2 |
| SE/Software Reliability | 2 |
| SF/Reliability through Redundancy | 2 |
| SDF/Development Methods | 2 |
| SP/Professional Ethics | 4 |
| SP/Analytical Tools | 3 |

**IEEE Body of Knowledge coverage**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **KA** | **Knowledge Unit** | **Topics Covered** | **T1 hour** | **T2 hour** | **Elective hours** |
| SE | **SE/Software Processes** | Systems level considerations, i.e., the interaction of software with its intended environment (cross reference  IAS/Secure Software Engineering)  • Introduction to software process models (e.g., waterfall, incremental, agile)  o Activities within software lifecycles  • Programming in the large vs. individual programming  Evaluation of software process models | 2 | 1 | 0 |
| SE | **SE/Software Project Management** | • Team participation  o Team processes including responsibilities for tasks, meeting structure, and work schedule  o Roles and responsibilities in a software team  o Team conflict resolution  o Risks associated with virtual teams (communication, perception, structure)  • Effort Estimation (at the personal level)  • Risk (cross reference IAS/Secure Software Engineering)  o The role of risk in the lifecycle  o Risk categories including security, safety, market, financial, technology, people, quality, structure  and process | 0 | 2 | 0 |
| SE | **SE/Software Verification and Validation** | * Verification and validation concepts * Inspections, reviews, audits * Testing types, including human computer interface, usability, reliability, security, conformance to * specification (cross-reference IAS/Secure Software Engineering) * Testing fundamentals (cross-reference SDF/Development Methods) * Unit, integration, validation, and system testing * Test plan creation and test case generation * Black-box and white-box testing techniques * Regression testing and test automation * Defect tracking * Limitations of testing in particular domains, such as parallel or safety-critical systems | 0 | 4 | 0 |
| SE | **SE/Software Evolution** | * Software development in the context of large, pre-existing code bases * Software change * Concerns and concern location * Refactoring * Software evolution * Characteristics of maintainable software * Reengineering systems * Software reuse * Code segments * Libraries and frameworks * Components * Product lines |  | 2 |  |
| SE | **SE/Software Reliability** | * Software reliability engineering concepts * Software reliability, system reliability and failure behavior (cross-reference SF/Reliability Through   Redundancy)   * Fault lifecycle concepts and techniques * Software reliability models * Software fault tolerance techniques and models * Software reliability engineering practices * Measurement-based analysis of software reliability |  | 1 |  |
| SF | **SF/Reliability through Redundancy** | * Distinction between bugs and faults * Redundancy through check and retry * Duplication/mirroring/replicas * Other approaches to fault tolerance and availability |  | 2 |  |
| SDF | **SDF/Development Methods** | * Program comprehension * Program correctness * Types of errors (syntax, logic, run-time) * The concept of a specification * Defensive programming (e.g. secure coding, exception handling) * Code reviews * Testing fundamentals and test-case generation * The role and the use of contracts, including pre- and post-conditions * Unit testing * Simple refactoring | 2 |  |  |
| SP | **SP/Professional Ethics** | * Community values and the laws by which we live * The nature of professionalism including care, attention and discipline, fiduciary responsibility, and mentoring * Keeping up-to-date as a computing professional in terms of familiarity, tools, skills, legal and professional framework as well as the ability to self-assess and progress in the computing field * Professional certification, codes of ethics, conduct, and practice, such as the ACM/IEEE-CS, SE, AITP * Accountability, responsibility and liability (e.g. software correctness, reliability and safety, as well as ethical confidentiality of cybersecurity professionals) * Forms of professional credentialing * Acceptable use policies for computing in the workplace | 2 | 2 |  |
| SP | **SP/Analytical Tools** | * Ethical argumentation * Ethical theories and decision-making * Moral assumptions and values | 1 | 2 |  |