Department of Computing
The University of the West Indies, Mona
Semester 1, 2016-2017

Course Title & Code:	COMP2140 - Software Engineering	**Level:** 2	**Credits:** 3
Pre-requisite:	COMP1161		
Teaching Team:			
Lecturer:	Mr. Ricardo Anderson, ricardo.anderson02@uwimona.edu.jm		
Studio Facilitators:	Dr. Claudine Allen, claudine.allen@uwimona.edu.jm		
	Mr. Eyton Ferguson, eyton.ferguson@uwimona.edu.jm		

COURSE RATIONALE

Software development, which can involve an individual developer or a team of developers, requires choosing the tools, methods, and approaches that are most applicable for a given development environment. Software engineering is the discipline that introduces students to the necessary knowledge, skills and practices for effectively and efficiently building information systems that satisfy the requirements of users and customers. Software development is not just the art of writing programs but an assorted array of concepts, principles, methods, techniques and tools that must be applied to effectively manage the analysis, design, construction, and deployment of information systems. This course covers the core body of knowledge in Software Engineering that the IEEE/ACM Computing Curricula recommends for computing graduates.

COURSE DESCRIPTION

The course will introduce students to the intricacies in planning and developing large software systems and emphasising the need for different methods. The students will be introduced to techniques and tools to facilitate large information systems development. Students will be made aware of the professional and ethical issues that arise during the development, use and evolution of software artefacts. Throughout the course emphasis will be placed on the importance of building useful software systems in a cost-effective manner.

LEARNING OUTCOMES

At the end of this course students should be able to:

- Analyse typical organisational processes and propose software solutions that will assist these processes.
- Describe different process models and recommend a process model for a given scenario.
- Implement techniques to manage the life cycle stages of the development process.
- Apply techniques for systems modelling and analysis to produce a set of software requirements for a medium-sized software system.
- Use and select a range of modelling techniques and tools in designing software systems.
- Distinguish between program validation and verification; describe the role that tools can play in the validation of software.
- Distinguish the various types and levels of testing (unit, integration, systems, and acceptance) for medium-sized software products.
- Identify the principal issues associated with software evolution and explain their impact on the software life cycle.
- Demonstrate awareness of ethical issues in software development.
- Develop software effectively in by working as part of a team.
- Demonstrate effective presentation skills.

COURSE CONTENT

- Software Design
  - Fundamental design concepts and principles
  - The role and the use of contracts
  - Structured design
  - Design qualities
    - Internal - including low coupling, high cohesion, information hiding, efficiency
    - External - including reliability, maintainability, usability, performance
- Using APIs
  - Programming using APIs

- Tools and Environments
  - Programming environments
  - Requirements analysis and design modelling tools
  - Testing tools including static and dynamic analysis tools
  - Tools for source control, and their use in particular in team-work
  - Configuration management and version control tools
  - Tool integration mechanisms
- Software Processes
  - Software life-cycle and process models
  - Software process capability maturity models
  - Approaches to process improvement
  - Process assessment models
  - Software process measurements
- Requirements Specifications
  - Systems level considerations
  - Software requirements elicitation
  - Requirements analysis modelling techniques
  - Functional and non-functional requirements
  - Acceptability of certainty / uncertainty considerations regarding software / system behaviour
  - Prototyping
- Software Verification Validation
  - Distinguishing between verification and validation
  - Static approaches and dynamic approaches
  - Validation planning; documentation for validation
  - Different kinds of testing – human computer interface, usability, reliability, security, conformant to specification
  - Testing fundamentals, including test plan creation and test case generation black-box and white-box testing techniques
  - Defect seeding
  - Unit, integration, validation, and system testing
  - Measurements: process, design, program
  - Verification and validation of non-code (documentation, help files, training materials)
  - Fault logging, fault tracking and technical support for such activities
  - Regression testing
  - Inspections, reviews, audits
- Software Evolution
  - Software maintenance
  - Characteristics of maintainable software
  - Reengineering Legacy systems
  - Refactoring
- SE/Software Project Management
  - Team management
    - Team processes
    - Team organization and decision-making
    - Roles and responsibilities in a software team
    - Role identification and assignment
    - Project tracking
    - Team problem resolution
  - Project scheduling
  - Software measurement and estimation techniques
  - Risk analysis
    - The issue of security
    - High integrity systems, safety critical systems
    - The role of risk in the life cycle
- Software quality assurance
  - The role of measurements

- Software configuration management and version control; release management
- Project management tools
- Software process models and process measurements

- 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 professional (in terms of knowledge, tools, skills, legal and
  - professional framework as well as the ability to self-assess and computer fluency)
  - Various forms of professional credentialing and the advantages and disadvantages
  - The role of the professional in public policy
  - Maintaining awareness of consequences
  - Ethical dissent and whistle-blowing
  - Codes of ethics, conduct, and practice (IEEE, ACM, SE, AITP, and so forth)
  - Dealing with harassment and discrimination
  - "Acceptable use" policies for computing in the workplace
  - Healthy computing environment (ergonomics)
- Risks
  - Historical examples of software risks (such as the Therac-25 case)
  - Implications of software complexity
  - Risk assessment and risk management; risk removal, risk reduction and risk control

METHODS OF DELIVERY
Delivery is interactive.

	Contact Hours	**Credit Hours**
Lectures	27	27
Software Development Studios* (in lieu of tutorials)	22	11

(*A SD Studio is a session whereby we focus on practical software development. Its key elements are practice, public presentation, and review by peers in a small group.)

ASSESSMENT

Final written examination (2 hours)	**40%**
Coursework	**60%**

One software development group project (Software is typically developed by teams of developers). Grades will be adjusted according to the results of peer evaluation. See COMP3900 –Group Project)

- Requirements Documentation 10%
- Design model (e.g., UML diagrams) 5%
- Presentations 15%
- Final presentation of prototype system 15%
- 1 In-Course Test 15%

Students will be required to pass both the coursework and the final examination to pass the course.

TEXTBOOKS & READING MATERIALS

Prescribed

Sommerville, I. (2010) "Software Engineering", 9/E, Addison Wesley, 792pp, paper, ISBN-13: 978- 0137035151.

Recommended

Pressman, R., S. (2010) "Software Engineering, A practitioner's approach", 7/E, McGraw-Hill, 930pp, paper, ISBN 978–0–07–337597–7

Various Online Resources (OURVLE Notes /resources)