1.	**Name of Course/Module:** Agile Software Development
2.	**Course Code:** CAS380
3.	**Rationale for the inclusion of the course/module in the programme:** Agile software development has been gaining popularity due to the benefits brought about by this approach in software engineering. This course provides students with the traditional and latest approaches of agile design approaches. This course prepares the students to adopt agile methods in software development, as well as to keep themselves abreast with newest agile methods and their trends in the industries.
4.	**Faculty, Semester and Year offered: BICT,** Semester 5 Year 2
5.	**Credit Value: 3**
6.	**Prerequisite (if any):** None
7.	**Objectives:** The objectives of this course are to: ▪ Explain and apply OOD, UML, Design Patterns, Agile and XP methods with a detailed description of a complete software design for reusable programs in C++ and Java. ▪ Use a practical and problem-solving approach to show how to develop an object-oriented application – from the early stages of analysis, through the low-level design and into the implementation in an agile software development environment. ▪ Explain communications skills relevant to agile software development.
8.	**Learning outcomes:** Upon completion, the students should be able to: ▪ Apply the concept of agile software development in facilitating software engineering development. ▪ Identify the current tools and techniques in agile software development. ▪ Construct information system that is of high quality and consistent with business goals of an organization by applying state of the art agile software development. ▪ Complete projects that require the students to communicate effectively during the process of agile software development.
9.	**Transferable Skills:** ▪ Planning and Management – Students will be taught to plan and manage agile software development that covers testing, refactoring, and others. ▪ Applying agile-based approach – Students will be taught to adopt various agile-based design in software development such as single-responsibility principle, open-closed principle and others. ▪ Analytical – Students will be taught how to evaluate various kinds of agile-based approach. This enables the students to make the right choice of decision on adopting the kinds of agile approaches in their project. ▪ Project - Students will be given exposure working on a project which can be on an individual or team basis. ▪ Communication – Students are taught how to present their project outcomes during project presentation session. This covers how the students present their work, and response to questions.
10.	**Teaching-learning and assessment strategy:** **Teaching-learning** ▪ Lectures - lectures are conducted weekly that cover every topic stated in the course outline. Lecturers also promote interactive learning with the students where mutual participation in question and answer and short discussion are expected. ▪ Tutorials - students will be given assignment consisting short questions that require investigation and to be answered. **Assessment** ▪ Exam – it is a written form of summative assessment which is conducted in the final exam. ▪ Quiz - It is a written form of formative assessment that may be conducted during the lecture period. ▪ Assignment - It is a take home task that student has to complete outside of the classroom. It can be in the form of practical or theoretical report. ▪ Project - It is a take home task that student has to complete outside of the classroom. It may be assigned to an individual or group where marks are given based on the fulfillment to the assessment criteria of the project. ▪ Presentation - Student is assessed on the communication skills.
11.	**Synopsis:** This course covers thoughts by showing the errors, blind alleys, and creative insights that occur throughout the software design process. The syllabus also covers static and dynamics, principles of class design and complexity management. Principles of package design, analysis and design, patterns and paradigm crossings. Explanation the principles of OOD by demonstrates them with numerous examples completely worked-through designs and case studies. Also, to discuss the methods for designing and developing big software in detail and in-depth. Featured of case studies are important for building a complete system.
12.	**Mode of Delivery** Lectures and Tutorials

13. **Assessment Methods and Types**
Continuous assessment: 60%
Final exam: 40%

14. **Mapping of the course/module to the Programme Aims**
- Competent information and communication technology practitioner with solid theoretical and practical knowledge in the area of information technology.
- IT professional who is a proficient in communicating the state-of-the-art in information technology.
- IT professional who has the abilities in leading the specific task in a team with qualities in decision making and entrepreneurial attributes.

15. **Mapping of the course/module to the Programme Learning Outcomes**
- Able to apply knowledge and skills in the field of information and communication technology management.
- Able to solve real world problem using information and communication technology approach and techniques.
- Able to express ideas and opinions effectively in various communication style, tools and media.
- Able to recognize and formulate new approach or method in providing IT technical solutions.
- Able to deliver and produce ideas and solutions with entrepreneurial and technical solutions.
- Able to be participate in a project-based assignment at individual or team level.

16. **Content outline of the course/module and the SLT per topic**

1. Agile Development
   - 1.1. Agile Practices
   - 1.2. Overview of Extreme Programming
   - 1.3. Planning
   - 1.4. Testing
   - 1.5. Refactoring
   - 1.6. A Programming Episode

2. Agile Design
   - 2.1. Concept of Agile Design
   - 2.2. SRP: The Single-Responsibility Principle
   - 2.3. OCP: The Open-Closed Principle
   - 2.4. LSP: The Liskov Substitution Principle
   - 2.5. DIP: The Dependency-Inversion Principle
   - ISP: The Interface-Segregation Principle

3. The PAYROLL Case Study (Part 1)
   - 3.1. Command and Active Object
   - 3.2. Template Method & Strategy: Inheritance vs. Delegation
   - 3.3. Facade and Mediator
   - 3.4. Singleton and Monostate
   - 3.5. Null Object
   - 3.6. The Payroll Case Study: Iteration One Begins
   - The Payroll Case Study: Implementation

4. The PAYROLL Case Study (Part 2)
   - 4.1. Packaging The Payroll System
   - 4.2. Principles of Package Design
   - 4.3. Factory

5. The Weather Station Case Study
   - 5.1. Composite
   - 5.2. Observer – Backing into a Pattern
   - 5.3. Abstract Server, Adapter, and Bridge
   - 5.4. Proxy and Stairway to Heaven: Managing Third Party APIs
   - Case Study: Weather Station

6. The ETS Case Study
   - 6.1. Visitor
   - 6.2. State
   - 6.3. The ETS Framework

17. **Main references supporting the course:**

Schiel, J. (2009). *Enterprise-scale agile software development*. Taylor and Francis.

Additional references supporting the course

Stober, T., & Hansmann, U. (2009). *Agile software development: Best practices for large software development projects*. Springer.
Hazzan, O., & Dubinsky, Y. (2008). *Agile software engineering*. Springer.