# AGILE SOFTWARE DEVELOPMENT

## COURSE OUTLINE

Responsible: Nikolaos Vidakis

| 1. GENERAL | | | |
|---|---|---|---|
| **SCHOOL** | School of Engineering | | |
| **ACADEMIC UNIT** | Department of Electrical and Computer Engineering | | |
| **LEVEL OF STUDIES** | Undergraduate | | |
| **COURSE CODE** | 0811.9.024.0 | **SEMESTER** | 1st |
| **COURSE TITLE** | Agile Software Development | | |

| **INDEPENDENT TEACHING ACTIVITIES**<br>if credits are awarded for separate components of the course | **WEEKLY TEACHING HOURS** | **CREDITS** |
|---|---|---|
| | 4 | 3 |
| | 1 | 1 |
| **Total** | **5** | **4** |

| | |
|---|---|
| **COURSE TYPE**<br>general background, special background, specialised general knowledge, skills development | Deepening / Consolidation of specialty knowledge |
| **PREREQUISITE COURSES** | Procedural Programming (1.004: 1st semester course), Object Oriented Programming (2.003: 2nd semester course) |
| **LANGUAGE OF INSTRUCTION and EXAMINATIONS** | Greek |
| **OFFERED TO ERASMUS STUDENTS** | Yes (in English) — Winter Semester |
| **COURSE WEBSITE (URL)** | https://eclass.hmu.gr/courses/ECE175/ |

## 2. LEARNING OUTCOMES

### Learning outcomes

The course deals with contemporary issues and concepts of software implementation and compares new methods and practices of development (flexible programming) with traditional (procedural programming)

The aim of the course is to acquire knowledge in modern techniques and methods of software implementation (agile programming, extreme programming, scrum development etc.). Using examples of both software implementation philosophies (flexible and pre-designed), their key features are presented in order to improve students' skills in professional software production.

Upon successful completion of the course the students will be able to:

- Know the basic concepts of Object Oriented & Procedural Software Engineering.
- Can implement Object Oriented & Procedural Software
- Know Quality Standards and Metrics of Procedural Systems
- Know Agile Programming Methods and Practices
- Compare and evaluate Planned & Agile Planning methods and practices.
- Compare and evaluate implementations with a pre-planned and agile implementation philosophy
- Adapt methods and practices to the specifics of the software

**General Competences**

- Autonomous & Independent work
- Teamwork
- Search, analysis and synthesis of data and information, using the necessary technologies
- Decision making
- Promoting liberal, creative and inductive/deductive thinking
- Work in an interdisciplinary environment Adapt to new situations
- Project Planning and Management

**3. SYLLABUS**

Theoretical Lecture Units

- Introduction and Historical Review

1. Procedural Programming
2. Object Oriented Software Implementation
3. Agile Implementation

- Principles of Analysis & Design:

1. SRP - Single Responsibility Principle
2. OCP - Open-Closed Principle (Open-Closed Design Principle)
3. LSP - Liskov Substitution Principle (Liskov Substitution Principle)
4. DIP - Dependency Inversion Principle
5. ISP - Interface Segregation Principle

- Basic Development Concepts
- Design Patterns:

1. Record
2. Standards
3. Construction standards
4. Structural standards
5. Behavioural standards

- Quality Systems of Pre-designed Systems:

1. Weighted Methods per Class (WMC)
2. Depth of Inheritance Tree (DIT)
3. Number of Children (NOC)
4. Coupling Between Object Classes (CBO)
5. Response for a Class (RFC and RFC ')
6. LCOM1, LCOM2, LCOM3

- Introduction to Flexible Programming:

1. The Manifesto for Agile Planning
2. Values and principles

- Agile Planning Perspective:

1. Comparison with traditional software implementation,

2. Adaptive vs. Predictive,
3. Iterative vs. Waterfall,
4. Code vs. Documentation

- Methods and practices used in flexible programming:

1. Methods such as ASD, RUP, AUP, DSDM etc.
2. Practices such as ATDD, IID klp.

- Customization of Methods: Customization of methods and practices to the specifics of the software.
- Overview and comparison with other methods:

1. RAD (Papid Application Development)
2. CMMI (Capability Maturity Model Integration)

- Implementation Evaluation with the philosophy of Agile Programming:

1. Implementation evaluation based on quantitative characteristics
2. Implementation evaluation based on measurable objectives

- Comparison of Planned and Flexible Programming

Laboratory Exercises

- In the laboratory part of the course students have the opportunity to practice the concepts of theory by using exercises that cover the material extensively and cultivate correct programming skills for flexible & agile software development.

| **4. TEACHING and LEARNING METHODS - EVALUATION** | |
|---|---|
| **DELIVERY**<br>Face-to-face, Distance learning, etc. | In-Class Face-to-Face |
| **USE OF INFORMATION AND COMMUNICATIONS TECHNOLOGY**<br>Use of ICT in teaching, laboratory education, communication with students | • Use of ICTs in lecturing<br>• Specialized Software for analysis, design and<br>• implementation of software.<br>• Use of ICTs for the communication with students via the e-class platform |
| **TEACHING METHODS**<br>The manner and methods of teaching are described in detail. | |

| **Activity** | **Semester workload** |
|---|---|
| Lectures | 24 |
| Tutoring | 11 |
| Small individual exercises | 20 |
| Teamwork Project with case study | 35 |
| Non-guided personal study | 30 |
| **Course total** | **120** |

| | |
|---|---|
| **STUDENT PERFORMANCE EVALUATION**<br>Description of the evaluation procedure | Assessment Language: Greek<br>All announcements for the course regulations and complementary reading material are permanently posted in the course web page. The course grade incorporates the following evaluation procedures:<br>Theory: Final written examination in the whole material (100%). The exam includes theory questions (from 3 to 5) and practice exercises (from 1 to 2).<br>Laboratory: The final grade consists of written laboratory work (10%), project preparation (50%) and final exam (40%)<br>The evaluation criteria are announced to the students at the beginning of each semester and are posted on the course website in the open e-class LMS. |

## 5. ATTACHED BIBLIOGRAPHY

Recommended Bibliography:


- Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Kern, J. (2001). Manifesto for agile software development.
- Bruce Eckel:Thinking in Java, Prentice Hall, 1999, ISBN 0-13659-723-8
- Bruegge, B., & Dutoit, A. H. (2009). Object-?Oriented Software Engineering. Using UML, Patterns, and Java. Learning, 5(6), 7.
- Cockburn, A. (2006). Agile software development: the cooperative game. Pearson Education.
- Cockburn, Alistair, Agile Software Development, Addison Wesley, 2002.
- Highsmith, J., & Cockburn, A. (2001). Agile software development: The business of innovation. Computer, 34(9), 120-127.
- Schwaber, K., & Beedle, M. (2002). Agile software development with Scrum (Vol. 1). Upper Saddle River: Prentice Hall.


Relevant Scientific Journals:

- Agile Alliance/Manifesto ? Principles of the Agile Alliance
- The New Methodology (Martin Fowler)
- The Agile Manifesto: Where It Came from and Where It May Go (Martin Fowler)
- Pairprogramming.com
- Agile Modeling
- www.AmbySoft.com/javaCodingStandards.pdf