AGH University of Krakow | Faculty of Mechanical Engineering and Robotics
| Department of Robotics and Mechatronics

MECHATRONIC ENGINEERING | I cycle | semester VI | 2025/2026

Object oriented programming and software engineering

Instruction III:

Software development processes

You will learn:

How software projects are organized in practice and how to start your own project in a structured way.

You will understand the difference between Waterfall and Agile development, be able to break your project into smaller, manageable parts, define a first working version of your system, divide work between team members, prepare a basic development plan for your project.

By the end of the class, your team should be ready to start implementation of the project.

Course supervisor:
dr inż. Lucjan Miękina, miekina@agh.edu.pl

Instruction author:
mgr inż. Joanna Koszyk, jkoszyk@agh.edu.pl

1. Initial information

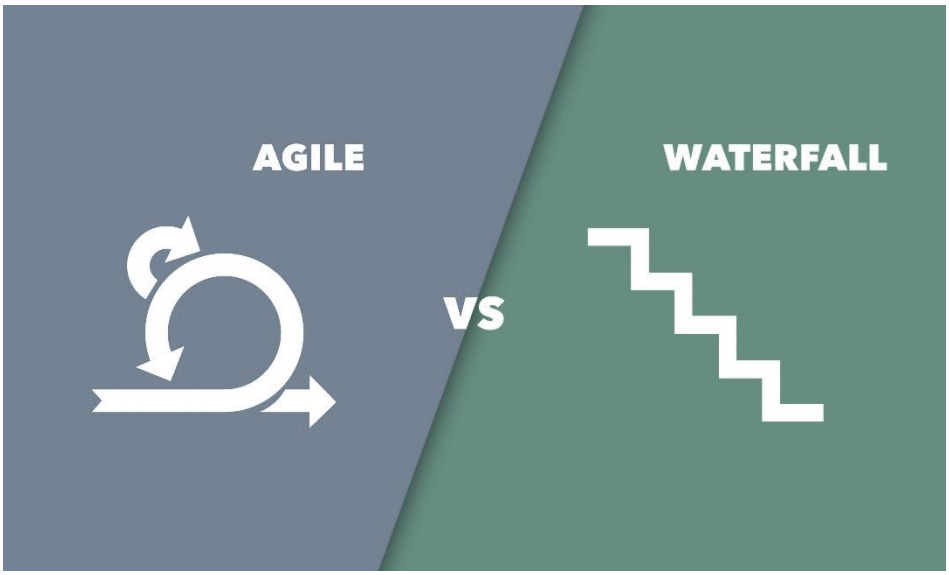
In real software development, especially in engineering systems writing code is only part of the work. Equally important is how the work is organized.

Without structure, teams can duplicate work, block each other, lose track of progress or fail to integrate their code.

To avoid this Waterfall or Agile or other development approaches can be used. They define how a system is planned, implemented, and improved over time.

In this class, you will use a simplified Agile approach to develop your project step by step, while always maintaining a working version of the system.

2. Theoretical Content



Waterfall is a sequential way of developing software. Work is divided into stages such as requirements, design, implementation, and testing. Each stage is completed before the next one begins.

This approach assumes that all requirements are known at the beginning. It works well when the system is clearly defined and unlikely to change.

Agile focuses on developing the project step by step. Work is divided into small parts, and each part is implemented in a short cycle. After each cycle, the system should work, even if it is not complete.

This approach allows incremental development, easier debugging and adaptation to new ideas.

3. Task

Work in your project teams.

Describe your project in one or two sentences. Next, list the main features of your system. Think about what the user can do and how the program should respond. Keep it simple.

Then, convert these features into tasks. Each task should describe one small piece of functionality. For example, instead of writing "build system", define tasks such as handling user input, displaying a menu, or storing data.

You should write your tasks using Microsoft Teams Tasks / Planner

Create columns such as To Do, In Progress, Done.

After defining the tasks, decide what the first working version of your program will be. This version should be very simple, but it should compile and run. It should demonstrate at least one basic functionality of your system.

Divide the tasks between team members. Each person should have clearly assigned tasks.