University of Miskolc
Department of Information Technology

Full-time programme
2025/26. I. semester

GEIAL316-B2a
Software Technology Lab

The aim of the course is to enable students to deepen their knowledge of modern software engineering methods and tools in a practice-oriented manner. The course provides an overview of the most important software development models and methodologies (waterfall, V-model, iterative–incremental approaches, agile methods, Scrum, Kanban, DevOps), with a strong emphasis on the practical application of software testing, quality assurance, continuous integration (CI), and version control. Students become familiar with Java-based development and testing tools (JUnit, Maven, Git, Docker), as well as the principles and design patterns of object-oriented software design (SOLID, KISS, DRY). Throughout the semester, students work in teams to design, implement, and present a complex software project, fostering the collaboration, planning, and implementation skills expected in an industrial software development environment.

Course lecturer: Tamás Tompa PhD, assistant professor
Preliminary requirements: GEIALXXX-XX (software technology, object oriented programming)
Course completion: signature and practical grade, 5 credits
Contact hours per week: 1 hour lecture, 3 hours laboratory
Lecture time and place: Monday 14-15 In/15.
Laboratory time, place, instructor: Monday 15-18, In/15., Tamás Tompa, PhD

SCHEDULE

Week	Lecture	Laboratory
1.	Software technology, software development models. Software life cycle, methodologies, waterfall model, V-model, iterative–incremental methodologies, prototype model, agile software development, Scrum, Extreme Programming, comparison of methodologies, Kanban, Lean, DevOps, Chaos Engineering, software code of ethics	Discussion of software projects, teams creation, creation of GitHub repositories
2.	Basics of software testing: software quality assurance, testing principles, testing techniques, testing levels, testing activities, documentation, software relations during the life cycle	Software testing, JUnit
3.	Software testing levels: component testing, integration testing, system testing, acceptance testing. Reasons for their emergence, rules, phases, strategies	Software testing, JUnit
4.	Software testing levels: dependencies, dependency elimination, techniques, mocking	Software testing, JUnit, mocking
5.	Continuous Integration (CI) tools: definition of continuous integration, CI processes, CI life cycle	Maven basics, project creation
6.	Continuous Integration (CI) tools: CI build tools, Make, Java classpath, Apache Ant, Apache Maven, Docker: basic concepts, example, Docker workflow	Maven, multi-project setup
7.	Version control: basic concepts, file states, repository types, types of version control systems	Git, Maven
8.	Version control: Git architecture, Git hooks, communication between repositories, Gitflow, Mercurial, SVN, comparison of VCSs	Git, Maven

9.	Design patterns: OOP, OOP principles, importance of design patterns, reasons for their emergence, code decay, KISS (Keep It Simple, Stupid), Law of Demeter, SoC (Separation of Concerns), DRY (Don't Repeat Yourself)	SOLID principles
10.	Design patterns: SOLID principles, additional design patterns	SOLID principles
11.	Software project consultation	Software project consultation
12.	Software project consultation	Software project consultation
13.	Project presentations	Project presentations
14.	Project presentations retake	Project presentations retake

Course schedule and materials (slides):
www.iit.uni-miskolc.hu → Staff: Tompa Tamás → Courses → Software Technology Lab /
https://users.iit.uni-miskolc.hu/~tompa

Requirements for obtaining the signature and practical grade:

1. Completion and presentation (oral) of a complex software project in teamwork)
2. Attendance at least 70% of laboratory classes (9/13) and 60% of lectures (8/13)

Recommended literature:

1. Lecture and laboratory materials (https://users.iit.uni-miskolc.hu/~tompa)
2. Robert C. Martin - Clean Code
3. Mauro Pezzé, Michal Young: Software Testing and Analysis
4. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides - Design Patterns
5. Mark Richards, Neal Ford - Fundamentals of Software Architecture, An Engineering Approach
6. Mauro Pezzé, Michal Young - Software Testing and Analysis: Process, Principles and Techniques
7. Jez Humble, David Farley - Continuous Delivery
8. Ken Schwaber, Jeff Sutherland - The Scrum Guide
9. Martin Fowler - Refactoring

Miskolc, February 2, 2026

Tamás Tompa, PhD
Course coordinator