FARABI UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

Lecture 1. Introduction to Object-Oriented Software Development

Software, the development lifecycle and object-oriented design

Dr. Bazargul Matkerim, Professor-Researcher

Lecture goals and plan

Learning outcomes

Distinguish the main types of software.

Explain the stages of the SDLC.

Describe OOSE, OOAD and the role of modeling.

Lecture plan

1. Software and its applications
2. Software developers and teams
3. The development lifecycle
4. Object-oriented approaches
5. Example and review

Computer software

Software consists of programs and related procedures that enable a computer system to perform tasks.

System software

Manages the platform and its resources. Examples: operating systems and device drivers.

Programming tools

Support development. Examples: editors, compilers, linkers and debuggers.

Application software

Helps users perform tasks such as writing, accounting, learning and communication.

Application software: everyday and business tasks

Type	Examples from the supplied lecture
Word processing and text editing	Microsoft Word, WordPad, Notepad
Database software	Oracle, Microsoft Access
Spreadsheet software	Apple Numbers, Microsoft Excel
Multimedia software	RealPlayer, media players
Presentation software	Microsoft PowerPoint, Keynote
Enterprise software	Customer relationship management systems

Examples retain the original lecture’s categories. Some products are historical.

Application software: specialist tasks

Type	Examples from the supplied lecture
Information worker software	Documentation and resource-management tools
Educational software	Encarta, Britannica, MATLAB, Google Earth, NASA World Wind
Simulation software	Flight simulators and scientific simulators
Content access software	Web browsers and media players
Application suites	OpenOffice, Microsoft Office
Engineering and development tools	Integrated development environments (IDEs)

Examples retain the original lecture’s categories. Some products are historical.

Software development



A systematic process for creating and evolving software.

Typical work includes ideas, design, implementation, testing and defect correction.

Feedback can lead the team to revisit earlier decisions.

Software developers and teams



Developers contribute to research, design, programming and testing.

Large teams may divide responsibilities among specialists.

In smaller teams, one person may perform several roles.

Software Development Life Cycle



The SDLC organises the work of developing and reviewing software.

Its purpose is to meet user needs while managing quality, time and cost.

The stages can overlap and repeat as the team receives feedback.

Planning



Purpose: determine the problem, scope and feasibility.

Consider resources, schedule, cost and major risks.

Output: an initial plan that the team can revise as it learns.

Requirements analysis



Who will use the system, and what must they achieve?

What information is needed? Which rules and constraints apply?

Output: agreed requirements and observable acceptance conditions.

Software design



Design turns requirements into a proposed solution structure.

Define modules, responsibilities, interfaces and data relationships.

UML views can help explain selected aspects of that structure and behaviour.

Implementation



Developers translate the design into working software.

They implement modules, integrate changes and review code.

Implementation feedback may reveal changes needed in the design.

Testing



Tests compare observed behaviour with specified expectations.

The team investigates defects and checks the corrections.

Passing tests gives evidence within the tested conditions.

Deployment and maintenance



Deployment makes a release available in its intended environment.

Maintenance addresses defects, changed needs and environmental changes.

User feedback informs later releases and improvements.

Object-oriented software engineering

OOSE organises development around objects and their collaborations.

Ivar Jacobson and colleagues presented a use-case-driven approach in their 1992 book.

Use cases describe interactions through which an actor achieves a goal.

The approach connects user needs with analysis, design and implementation.

Object-oriented analysis and design

OOAD examines a problem and develops a solution using objects, responsibilities and visual models.

Analysis

Identifies domain concepts, user goals and the rules the system must respect.

Design

Assigns software responsibilities and defines interfaces and collaborations.

Implementation

Expresses the selected design in code and checks its behaviour.

Benefits and limits of object orientation

Modularity

Related state and behaviour can reside together, helping teams understand and change a system.

Reuse

Suitable classes and components can support reuse through composition or inheritance.

Maintainability

Clear interfaces and responsibilities can make testing and modification easier.

Benefits depend on design quality. Encapsulation alone does not guarantee security or correctness.

Worked example: a library loan

User goal: issue an available physical copy to a reader.



Analysis identifies Reader, BookCopy and Loan.
Design assigns the responsibility for creating and closing a loan.

Discussion: why should a book title and a physical copy be different concepts?

Summary and review

Software categories describe different purposes and users.
The SDLC connects planning, requirements, design and delivery.
OOAD uses objects and models to reason about a solution.

1. How do analysis and design differ?
2. Why does testing continue across the lifecycle?
3. Which objects would you model in a library system?

Reading and references

Washizaki, H. (Ed.). (2024).
Guide to the Software Engineering Body of Knowledge (SWEBOK), v4.0. IEEE Computer Society.

Pressman, R. S., & Maxim, B. R. (2020).
Software Engineering: A Practitioner's Approach, 9th ed. McGraw-Hill.

Sommerville, I. (2020).
Engineering Software Products: An Introduction to Modern Software Engineering, Global ed. Pearson.

Object Management Group. (2017).
OMG Unified Modeling Language (OMG UML), Version 2.5.1.