LECTURE 1:
INTRODUCTION TO SOFTWARE DEVELOPMENT

What is Software Development?

- Process of designing, creating, testing, deploying, and maintaining software
- Converts user needs into working digital solutions
- Involves tools, programming languages, and structured processes

Software development is more than writing code — it includes planning, analysis, testing, deployment, documentation, and maintenance. Emphasize that developers solve real problems.

Example

- Mpesa mobile money platform development lifecycle

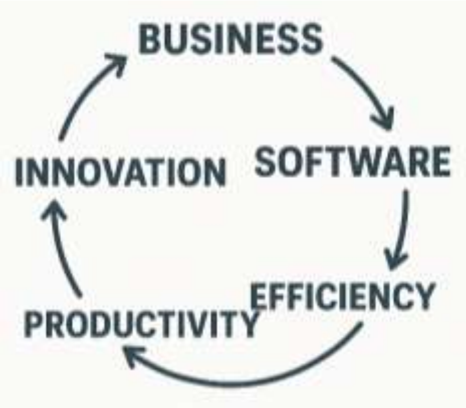
Why Software Matters Today

- Automates business processes
- Enables communication (apps, websites, messaging)
- Powers banking, healthcare, education, government services etc

Case Study

- The Kenyan eCitizen system digitizing government services

Illustration



The diagram represents a **continuous cycle** showing how different aspects of an organization interact to create growth, improvement, and sustainability. Each stage leads naturally to the next, forming a loop that strengthens business performance over time.

1. Business → Software

A business identifies needs, challenges, or opportunities.

To address these needs, the business adopts or develops **software solutions**—tools that automate tasks, improve communication, manage data, and support operations.

2. Software → Efficiency

Once implemented, software streamlines processes.

This leads to **increased efficiency**, meaning:

- Faster workflows
- Reduced errors
- Better resource utilization
- Improved coordination

3. Efficiency → Productivity

Greater efficiency results in higher **productivity**.

Employees and systems can produce more output with the same effort, time, or cost. This boosts overall performance and competitiveness.

4. Productivity → Innovation

When productivity rises, the organization gains:

- More free time
- More capacity
- More data insights
- More confidence

These enable **innovation**, encouraging the business to create new ideas, products, solutions, or strategies.

5. Innovation → Business

Innovation strengthens and transforms the **business**.

It leads to:

- New revenue streams
- Improved customer satisfaction
- Competitive advantage
- Sustainable growth

Categories of Software

- System Software (OS, drivers)
- Application Software (Office, Adobe suite, banking apps, browsers)
- Embedded Software (IoT devices, vehicle sensors)
- Middleware

System software acts as the foundation, while application software provides end-user functionality.

Software Engineering vs Software Development

Definition

Software Engineering

A disciplined, structured, and systematic approach to designing, building, maintaining, and managing software systems.

It treats software creation as an engineering process with standards, methodologies, models, and quality controls.

Software Development

The practical activity of writing, designing, deploying, and testing software applications. It is focused primarily on building functional software.

Scope

Software Engineering

- Broader scope
- Includes planning, requirements analysis, architecture, design, development, testing, deployment, maintenance, project management, quality assurance, security, and lifecycle management
- Considers scalability, reliability, performance, and long-term sustainability

Software Development

- Narrower scope
- Focuses mainly on writing code and building applications
- Involves coding, debugging, and implementing features

Examples

Software Engineering Example

- Designing a hospital management system:
- Gathering requirements
- Designing architecture
- Creating UML diagrams
- Choosing database structure
- Planning testing
- Ensuring security compliance

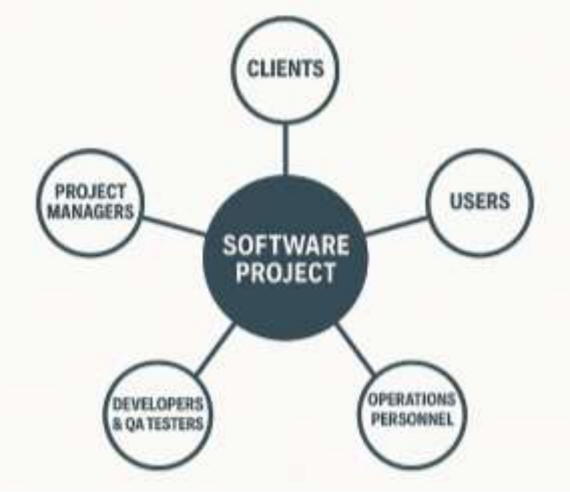
Software Development Example

- Writing the code for the login page
- Creating the dashboard
- Connecting the system to the database

Stakeholders in Software Development

- Clients
- Users
- Project managers
- Developers & QA testers
- Operations personnel

Illustration



Role	Responsibilities
Clients	• Provide funding and business goals• Define requirements and expectations• Approve decisions and deliverables• Give feedback on results
Users	• Use the system or product• Provide practical feedback on usability• Identify real-world problems and improvement needs
Project Managers	• Plan and manage the project timeline and budget• Coordinate all team members and stakeholders• Track progress and manage risks• Ensure the project meets objectives
Developers	• Write, test, and debug code• Build features according to requirements• Fix issues and maintain software
QA Testers	• Test software for bugs and performance issues• Ensure reliability, security, and quality• Verify the system meets requirements before release
Operations Personnel	• Deploy and maintain the software in production• Monitor performance, uptime, and security• Manage backups, updates, and configurations• Resolve operational/system issues

The Software Crisis
• Originated in the 1960s
• Software became too complex to manage
• Projects failed or exceeded deadlines
• Introduced the need for formal engineering principles

Case Study (SAGE)
SAGE was one of the most ambitious and expensive computing/military projects ever undertaken — a massive undertaking intended to provide real-time, continent-wide air defense. Its cost and complexity far outstripped initial expectations, making it "over-budget." Moreover, its full deployment was only achieved after many years of development, by which time the nature of military threats had shifted — making some of its capabilities obsolete, which qualifies as being "late.

Modern Development Approaches (DevOps)
DevOps is a set of practices, tools, and culture that combine development (Dev) and operations (Ops) to:

- Automate and speed up delivery
- Improve reliability and deployment quality
- Break down the wall between developers and operations teams

This includes:

- Continuous Integration
- Continuous Delivery
- Infrastructure as Code
- Rapid deployment cycles

NB: DevOps improves collaboration between developers and operations. Speeds up release cycles.

Documentation Importance
• Helps new developers understand system
• Reduces errors
• Ensures maintainability

Example
• API documentation for Safaricom's Mpesa Daraja API

Skills Needed for Developers

- Problem-solving: The ability to analyze a challenge, break it into smaller parts, and create effective solutions.

- Programming skills: Writing clear and efficient code to build software that performs specific tasks.
- Debugging: Finding and fixing errors or bugs in code to ensure the software works correctly.

- Communication: Sharing ideas clearly with team members, clients, and users to ensure everyone understands requirements and solutions.
- Version control mastery: Using tools like Git to track changes, collaborate safely, and manage code updates efficiently.

NB: Technical skill alone is not enough — communication & teamwork matter.

Challenges in Software Development

• Changing requirements
• Unrealistic deadlines
• Security risks
• Technology changes quickly

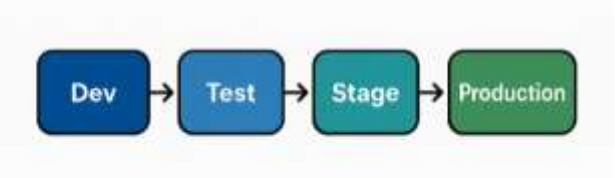
Example

• Maintaining legacy banking software

Deployment Environments

• Development environment
• Testing environment
• Staging environment
• Production environment

A pipeline showing Dev → Test → Stage → Production



Real-World Applications of Software

- Finance: mobile banking, ATMs
- Health: telemedicine, diagnostics
- Education: LMS, e-learning
- Agriculture: IoT sensors, smart irrigation

Case Study (read)

• Smart farming with IoT in Kenya

Software Development Team Roles

- Backend developer
- Frontend developer
- UI/UX designer
- QA engineer
- System architect