Software Requirements Engineering

Asst. Prof. Bilge Kağan Dedetürk

Software Requirements
Lecture Links

- bilgededeturk@erciyes.edu.tr
- https://github.com/kagandedeturk/SoftwareRequirementsEngineering
- https://avesis.erciyes.edu.tr/bilgededeturk/documents

Software Requirements
Lecture Books

- Software Engineering Body of Knowledge (SWEBOK) V3.0 IEEE
  https://www.computer.org/education/bodies-of-knowledge/software-engineering

- Software Requirements 3rd Edition – Karl Wiegers and Joy Beatty

- Software Engineering 10th Edition – Ian Sommerville

Software Requirements
Software Engineering Introduction

- “Our civilization runs on software” (Bjarne Stroustrup)

- Software controls a massive variety of devices such as cell phones, personal computers, electronic devices

Software Engineering Introduction

- Requirements
- Design
- Coding or Construction
- Testing
- Quality
- Process
- Tools and Methods
- Configuration Management
- Maintenance
- Project Management
- Economics
- Professional Practice

What We Will Learn From This Course

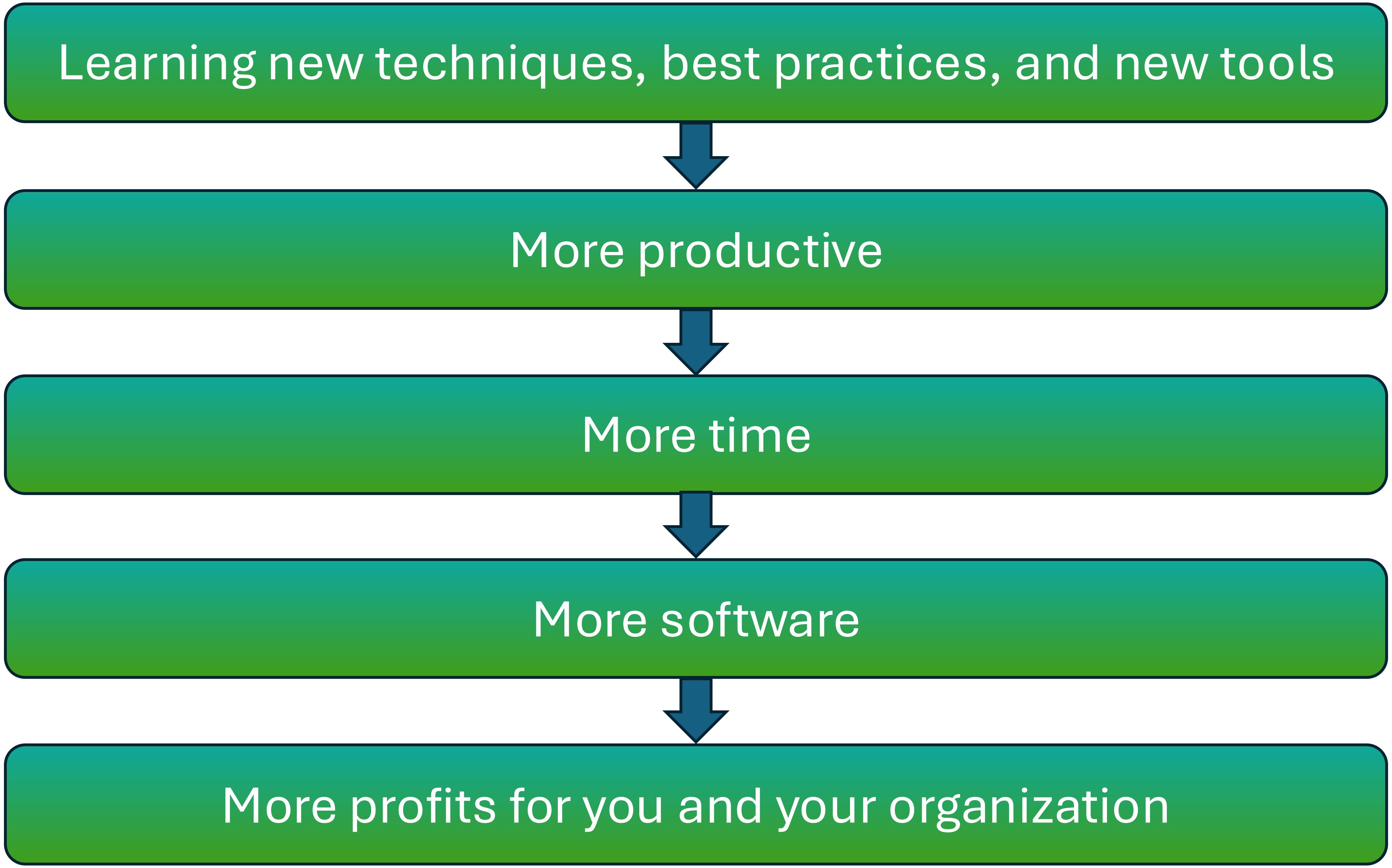
- Each software-related character’s role: developer, tester, designer, manager
- What problems they may face, how to solve them
- What should we expect from them ?
- What they’re expecting from us.
- What are their best practices, tools, and techniques?
- How do we know if they are doing a good job or not?
- Can we measure the quality of their work?
- The language they use everyday.

Software Engineering - Introduction

- The project manager uses charter, crashing.
- The designer uses coupling, design patterns.
- The developer uses bugs, debugging, build, making, and many more

Why Learn Software Engineering?

Why Learn Software Engineering?


Learning new techniques, best practices, and new tools
More productive
More time
More software
More profits for you and your organization

Why Learn Software Engineering?

VS

Software Crisis

- It is a term that talks about the many difficulties in developing large software systems during the 60s and 70s.
- Software project failures occur more frequently than they should.
- It is still valid today.

Never completed systems

Missed deadlines

Exceeded budgets

A system that does not do all that is required of it

A system that works but is difficult to use

A system is difficult to modify

A loss of trust from users

Why Learn Software Engineering?

- Inability to scale the techniques employed when developing small software systems to handle larger, more complex systems.

- Lack of any framework for software development projects planning and organization.

Why Learn Software Engineering?

Current challenges:

- The need to develop trustworthy high-quality software
- The higher demand of quick turnaround from concept to deployment and operation
- Increased software complexity
- The diversity of software systems.
- The need for increased efficiency in component-based reuse and automatic code generation
- Handle change

Why Learn Software Engineering?

- A structured approach for building large/complex systems
- A method for decomposing the problem into manageable portions is essential.
- A shared understanding of the task and proper communication is essential.
- Building large systems involve extensive group work
- Each member of the group needs to understand their task and how it interfaces with other tasks.
- Groups and individuals need to communicate in a commonly agreed language

What is Engineering?

- **Engineering:** The use of scientific principles to design and build machines, structures, and other items to achieve a goal.

What is Software Engineering?

The Software Engineering definition from IEEE is:

- (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.
- (2) The study of approaches as in (1).

Software Engineering Benefits

- Use powerful and well-accepted techniques for completing everyday tasks among projects.
- Accurately predict cost and schedule to complete our projects.
- Build desirable characteristics in our products like maintainability, reliability, etc.

Software Requirements
Requirements Engineering - Introduction

- Requirements engineering is usually the first step in any software
- It is challenging, costly, and could be the riskiest step of them all
- The analyst job is much more challenging.

The Requirements Engineer

- The requirement engineer could be called:
  - Software analyst
  - System analyst
  - Process analyst
  - Business architect
  - Project manager
  - Process engineer
  - Product manager
  - Product owner
  - Quality assurance analyst
  - Consultant

Requirements Engineering

- Software Requirements Fundamentals
- Requirements Process model
- Requirements Elicitation
- Requirements Analysis
- Requirements Documentation
- Requirements Validation
- Practical Consideration
- Requirements Tools

What Will We Learn?

- The different meanings of software requirements?
- What should the requirements engineer do?
- What skills should he/she possess?
- What tools that he/she uses?
- What techniques or best practices that we should be familiar with?
- Whom should he/she deal with?
- What should the people expect from him?
- What should he/she expect from the others?
- What terms do the requirements engineer mostluy use?
- Any special considerations we should be aware of?

Without Requirements

- Developers won’t know what is considered “Complete”.
- Testers won’t know what to test.
- Customers don’t know what to expect.
- Users can’t determine if the software will meet their needs

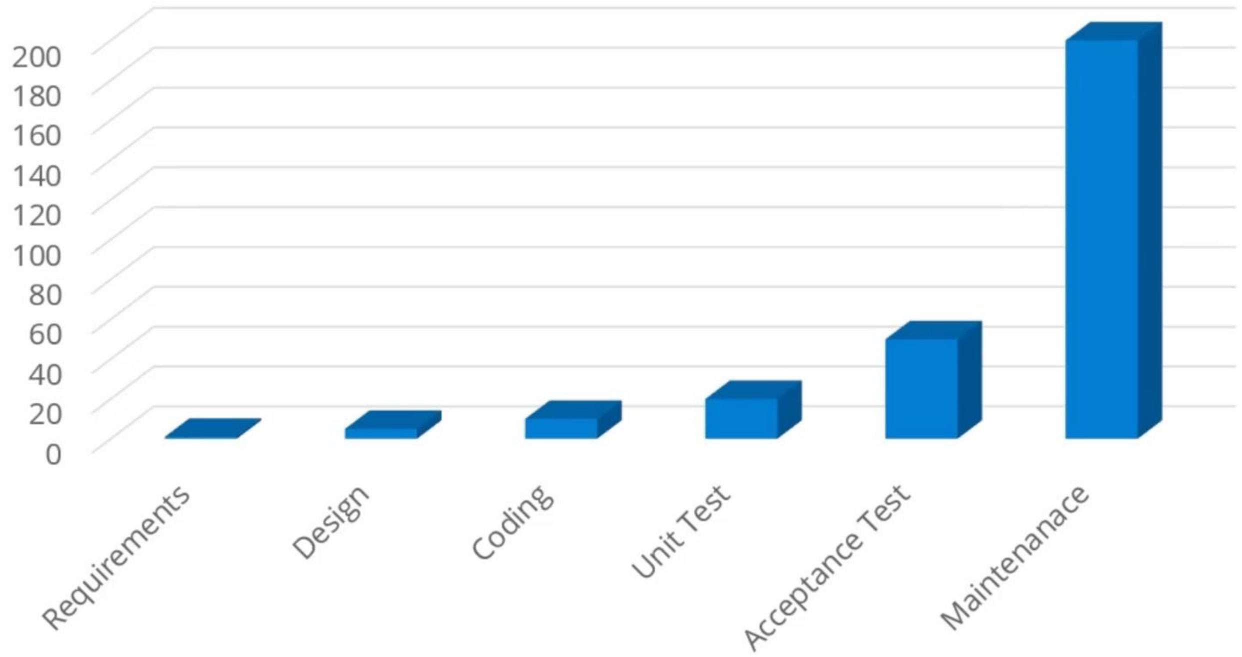
With Invalid Requirements

- Can lead to a non-useful system
- Will lead to major changes, schedule slippage, which will result in a higher cost.
- Can lead to extra steps to perform simple tasks using the software, making the system harder to use
- Can lead to an unnecessarily complicated, unusable system

Requirements Issues

- We have trouble understanding the requirements that we acquire from the customer
- We often record requirements in a disorganized manner
- We spend far too little time verifying what we do record
- We allow change to control us rather than establishing mechanisms to control change
- We fail to establish a solid foundation for the system or software that the user wants to be built

Cost of Repair



Life Cycle Phase	Relative Cost of Repair
Requirements	1
Design	5
Coding	10
Unit Test	20
Acceptance Test	50
Maintenance	200 - 1000