Foundations of Software Systems

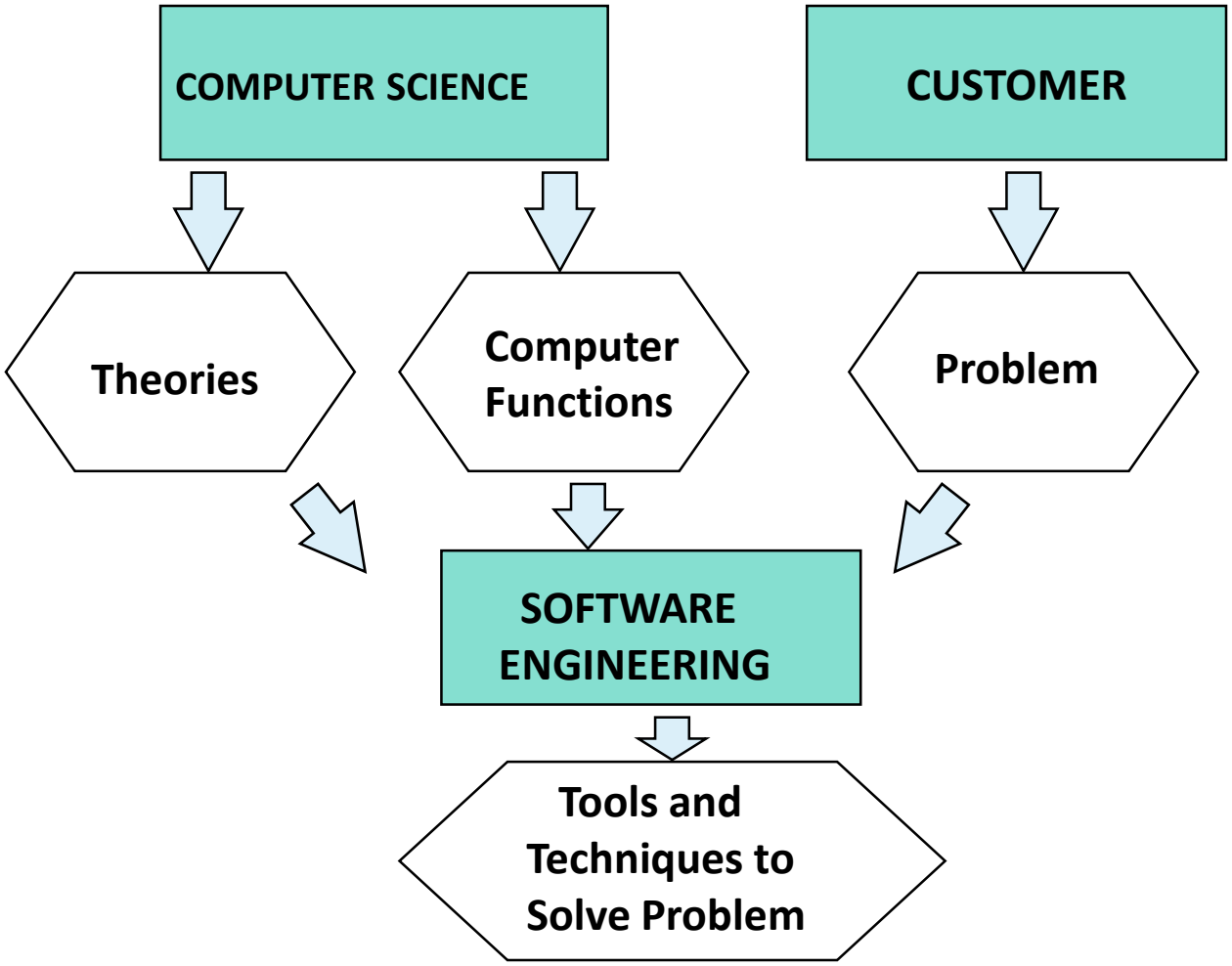
SYSTEM ANALYSIS & DESIGN

Instructor: Aphrodice Rwagaju

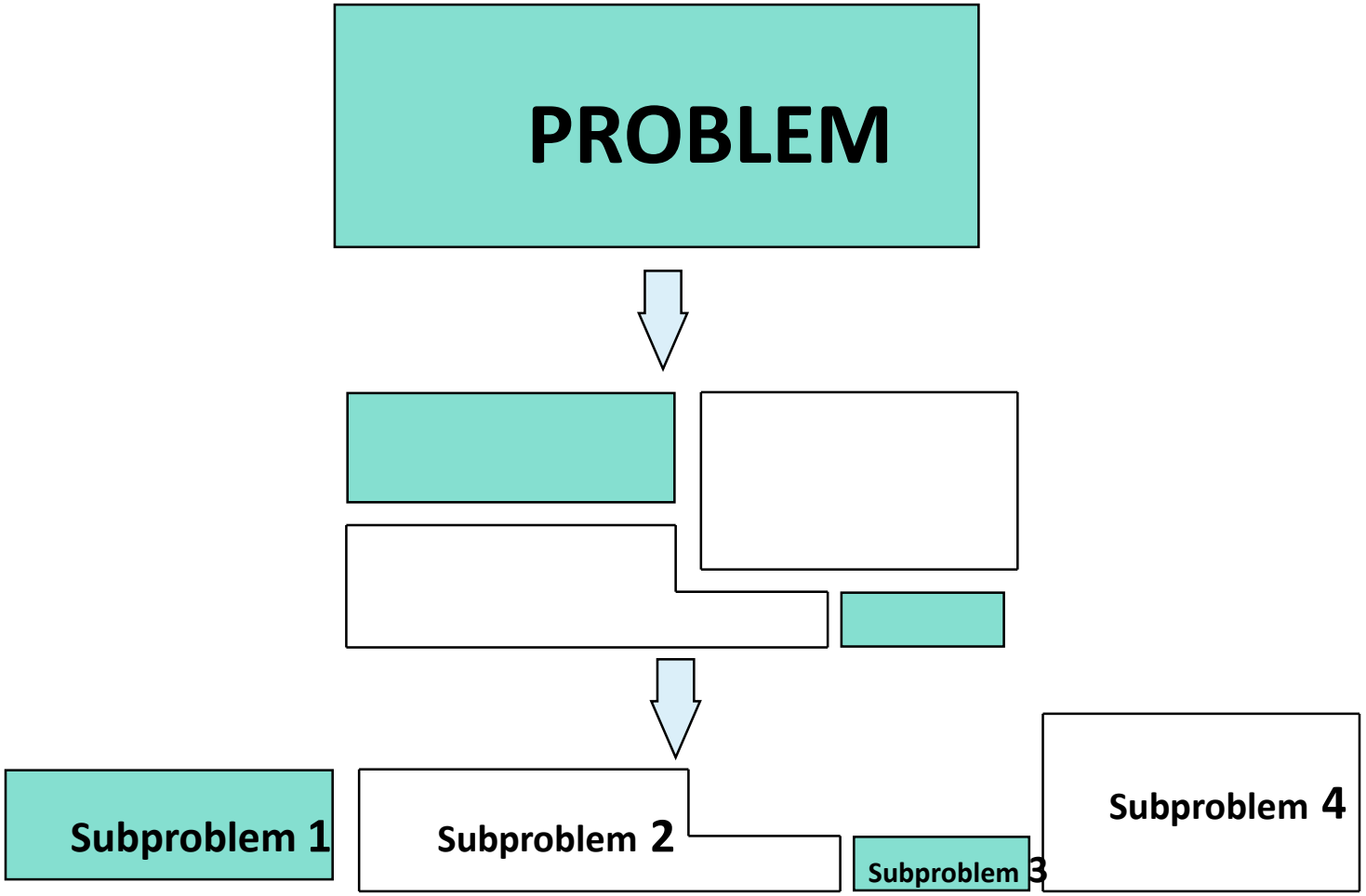
Software Systems

- Software Engineers build software systems
- Building a software system requires
  - (1) Planning,
  - (2) Understand what needed to be built,
  - (3) Designing the system,
  - (4) Writing the programs,
  - (5) Testing, and
  - (6) Maintaining the system.

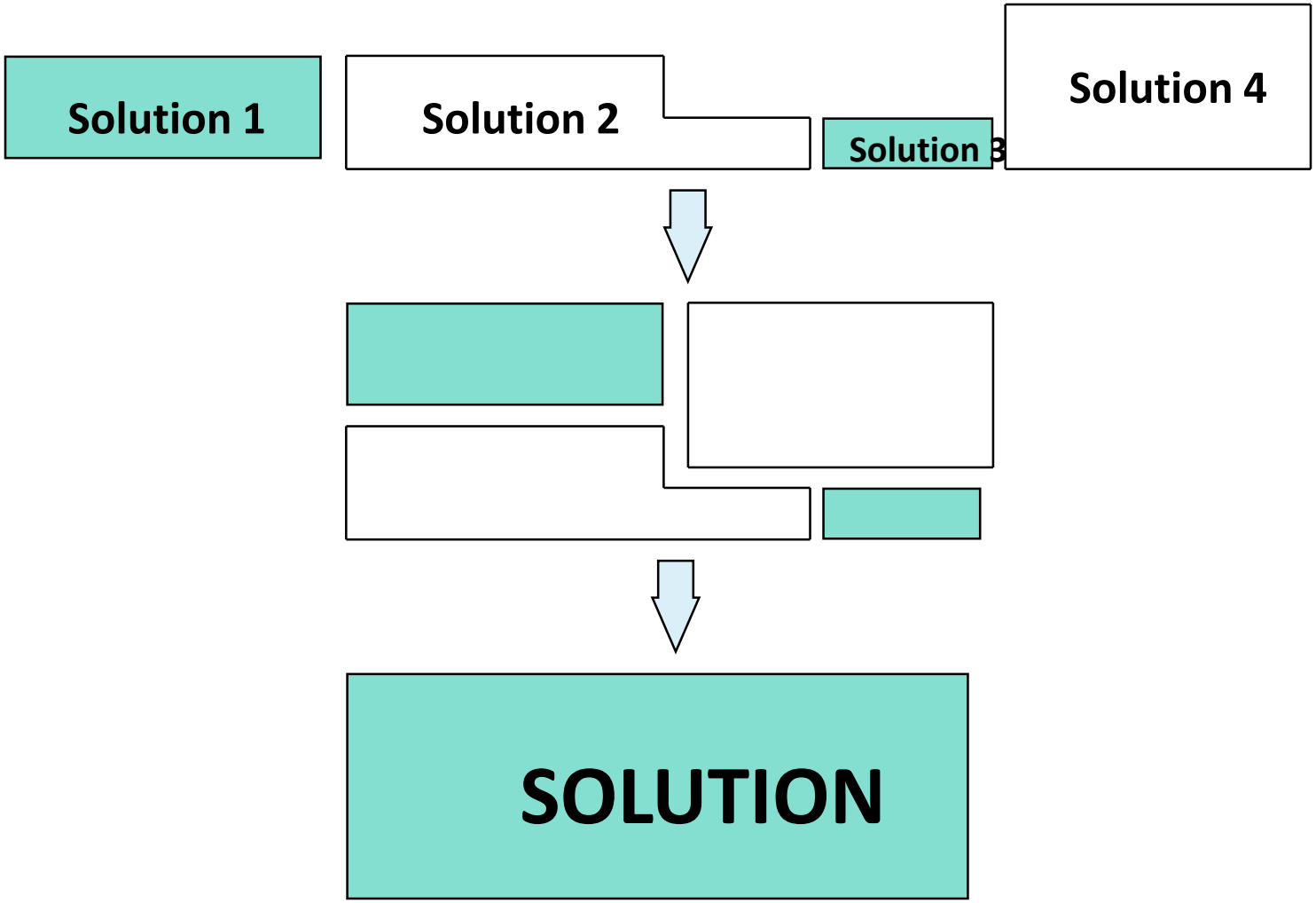
Software Engineers use knowledge of Computer Science to Solve problems



Problem Solving: Analysis



Problem Solving: Synthesis



How successful have we been?

- Writing software is an art as well as a science.
- Software engineering is about designing high-quality software.
- System Crash!!!
- Seeking wrong target!!!
- However, we have come a long way!!!
  (Can’t leave home without a computer.)

What is Good software?

- Context helps to determine the answer
  - Games and Safety-critical systems have different standard
- Evaluate software by evaluating
  - quality of the product
  - quality of the process
  - quality in the context of the business environment

Quality of the product

- Correctness
- Reliability
- Usability
- Testability
- Maintainability

Quality of the Process

- Many (development) activities will affect the ultimate product quality, e.g.
- Talking to customers and users to understand what they want
- Code reviews
- When to conduct testing

Quality in Context of the Business Environment

- Return on investment
- Express in dollars
- Express in effort
  - schedule
  - productivity
  - customer

Who does software engineering?

CUSTOMER

Sponsors system development

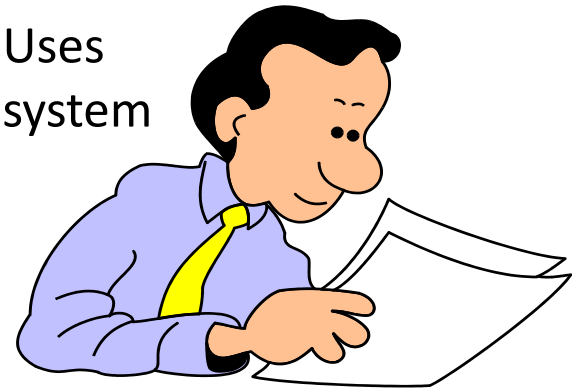
$$$, needs

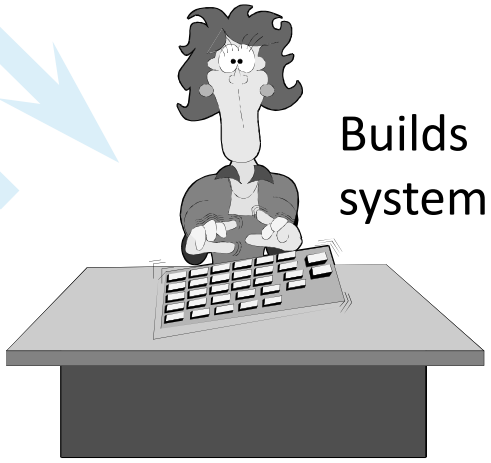
DEVELOPER

USER

Contractual obligation

Uses system





Builds system

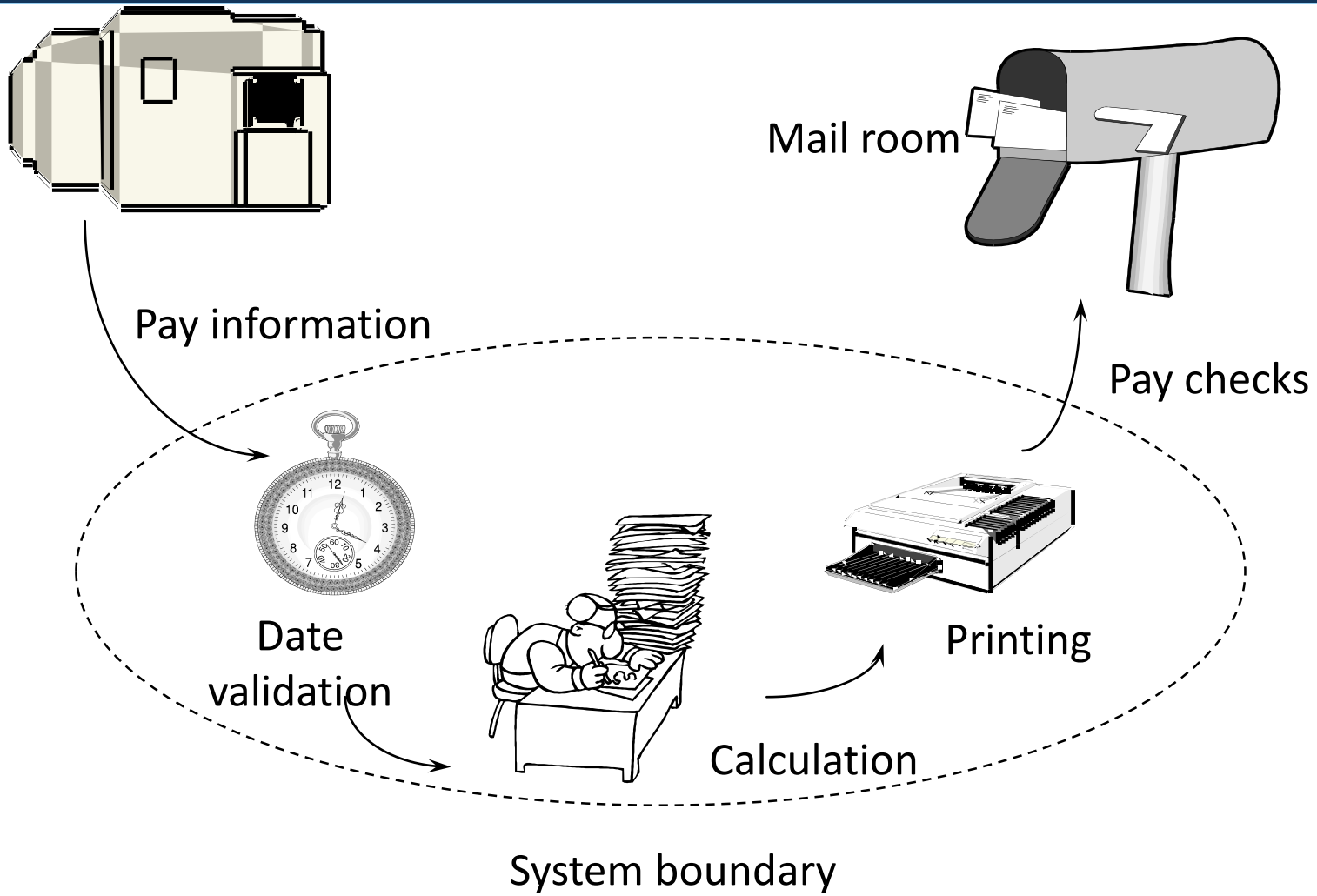
Needs

Software system

Understanding software systems: A Systems Approach

- A system is a collection of entities and activities, plus a description of the relationships that tie the entities and activities together.
  - An activity is something that happens in a system.
  - The elements involved in the activities are called entities.
  - Once entities and activities are defined, we match the entities with their activities.

System Boundary: Input and Output



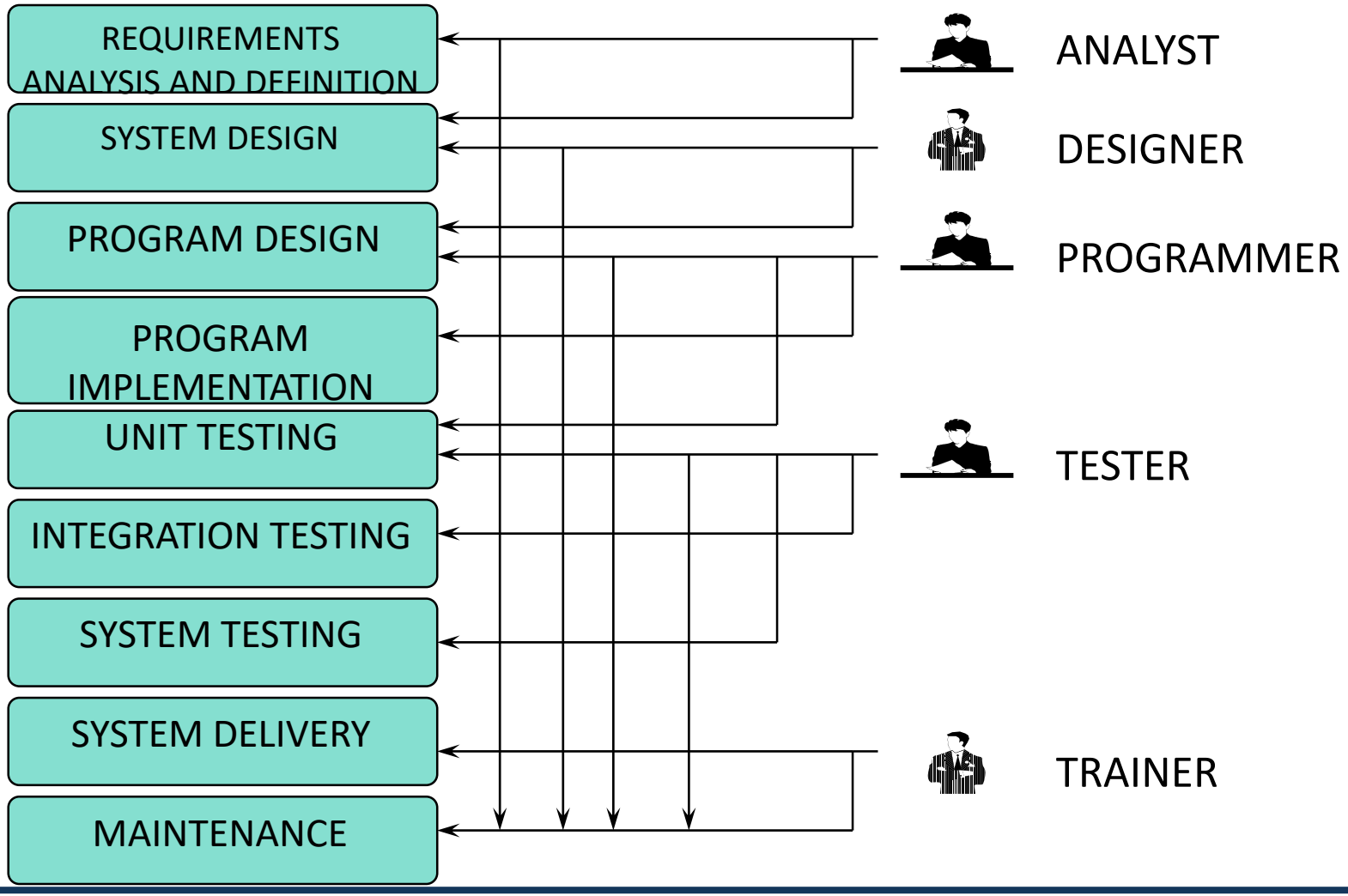
Building software systems: An Engineering Approach

- Software projects progress in a way similar to the house-building process.
  - Asking customers what house they want to build
  - Drawing floor plans (rooms) ((Model house))
  - Designing interior (e.g. where the light switch should be)
  - Testing each subsystem (e.g. testing each light switch, electrical subsystem)
  - Testing everything work together
  - Maintaining the house

Software development process

- requirements analysis and definition
- system design
- program design
- writing the programs (program implementation)
- unit testing
- integration testing
- system testing
- system delivery
- maintenance

Members of the development team



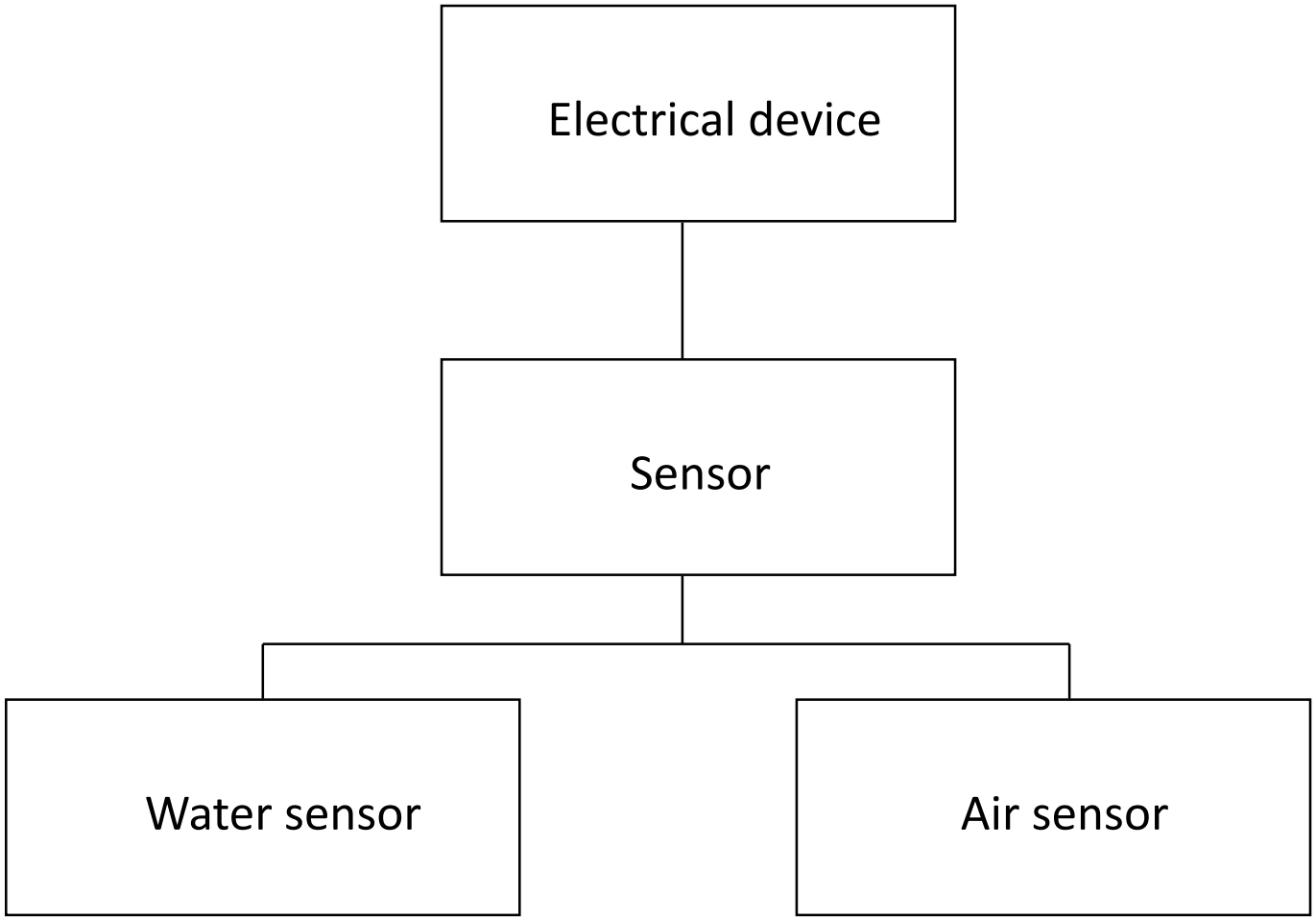
Fundamental notions in software engineering

- Abstraction
- Analysis and design methods and notations
- User interface prototyping
- Software Architecture
- Software Process
- Reuse
- Measurement
- Tools and Integrated Environments

Abstraction

- is a description of the problem at some level of **generalization** that allows us to concentrate on the key aspects of the problem without getting mired in the details.
- Identifying **classes** of **objects** that allow us to group items together
- Forming **hierarchies**

Abstraction



Analysis and Design Methods and Notations

- build models and check them for completeness and consistency
- use standard notation to help us communicate and to document decisions

User interface prototyping

- Prototyping means building a small version of a system to
  - help the user and customer identify the key requirements of a system
  - demonstrate feasibility of a design or approach
- Does the user like the “looks and feels”?

Software Architecture

- A system's architecture describes the system in terms of a set of architectural **units**, and a **map** of how the units relate to one another.

Software Architecture: Units

- Ways to partition the system into units:
  - modular decomposition: based on assigning functions to modules
  - data-oriented decomposition: based on external data structures
  - event-oriented decomposition: based on events that the system must handle
  - outside-in design: based on user inputs to the system
  - object-oriented design: based on identifying classes of objects and their interrelationships

Software Process

- Process of developing software (organization and discipline in the activities)
- contribute to the **quality** of the software and the **speed** with which it is developed

Reuse

- Take advantage of the commonalities across applications by reusing items from previous development
- Reusable components as business asset

Measurement

- By quantifying where we can and what we can, we describe our actions and their outcomes in a common mathematical language that allows us to evaluate our progress.

Tools and Integrated Environments

- Use tools to enhance software development
  - tools to help tracking the progress of the development
  - tools to help debugging programs
  - tools to help testing the programs

References

- Dr. Ben Choi, Software Design PPT presentation, Bell Labs Innovations