OOJ Lecture 5
Software Engineering and Software Development Process

Software Engineering Objectives

- **To achieve a level of programming literacy**
- **To expose you to software testing and debugging techniques**
- **To introduce you to the concepts of software safety and security**

Reading Material Text

- **Stephen Schach, *Classical and Object-Oriented Software Engineering*, McGraw-Hill**
- **S.L. Pfleeger, *Software Engineering, Theory and Practice*, Prentice Hall.**
- **R S Pressman, *Software Engineering: A Practitioner's Approach*, McGraw-Hill.**

Introductory Topics

- **Introduction and overview of Software Engineering**
- **Requirements and Specifications**
- **Program Design**
- **Program Coding**
- **Program Testing**
- **Program Debugging**

Objectives

- To understand the concepts behind good program design
- Understand what functional decomposition design methodology is
- Know the steps involved in functional decomposition design

Program Design and Coding

- **Problem analysis**
- **High-level design**
- **Functional decomposition**
- **Object decomposition**
- **Modularity**
- **Information hiding**
- **Cohesion & coupling**

Program Testing and Debugging

- **Unit testing**
- **Black-box testing**
- **White-box testing**
- **Boundary testing**
- **Regression testing**
- **Integration testing**
- **OO Testing**
- **Debugging**

Introduction

- **Self test questions:**
  - **What is software crisis?**
  - **What is software engineering, anyway?**
  - **How can a Java program be considered to be of good quality?**

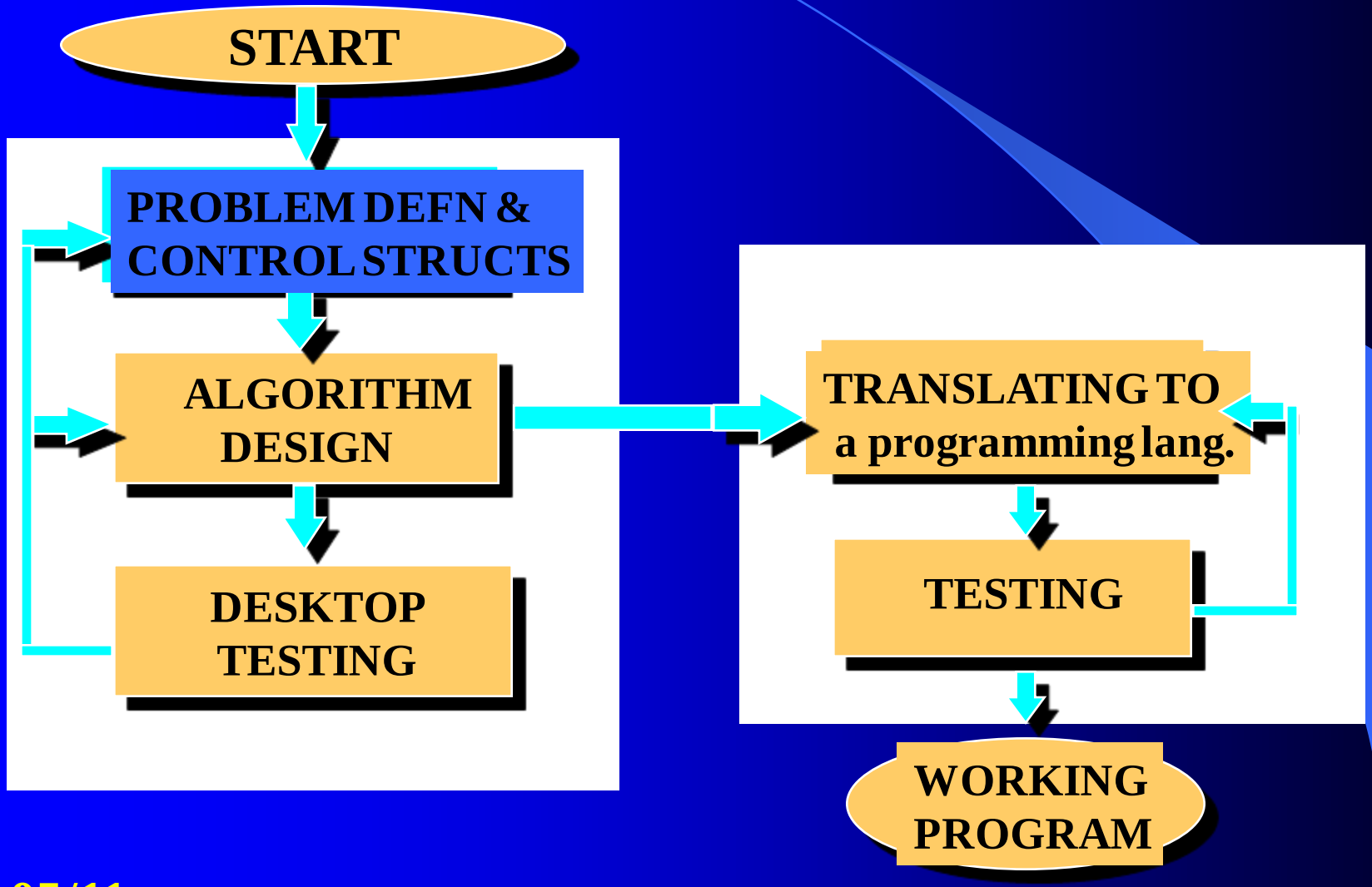
Historical Background

- **Software crisis**
  - **wrong estimates of development time, effort and costs**
  - **poor software quality**
- **SE is "the systematic approach to the development, operation, maintenance and retirement of software"-**

Software Quality

- **correctness**
- **reliability**
- **efficiency**
- **integrity**
- **usability**
- **maintainability**
- **testability**
- **flexibility**
- **portability**
- **reusability**
- **interoperability**

Program Development Process



Basic Steps in Program Development

1. **Define the problem**
2. **Outline a solution (including control structures)**
3. **Design an algorithm**
4. **Test the algorithm for correctness (desk checking/desktop testing)**
5. **Code the algorithm**
6. **Test the program**
7. **Document and maintain the program**

   (an ongoing task from Steps 1-6)

Problem Definition, Control Structures and Desk Checking

- **Self test questions:**
  - **What are the steps involved in problem definition?**
  - **What are the 3 major control structures?**
  - **What is desktop testing/ desk checking?**

What are the steps involved in problem definition?

- **Input: a list of the source data provided to the problem**
- **Output: a list of outputs required**
- **Processing: a list of actions required to produce the required outputs**

What are the 3 major control structures?

- **Sequence**
- **Selection**
- **Repetition**

What is desk checking or desktop testing?

- Walking through (each step) test data in the **algorithm** to identify any major logic errors.

Example: Yarra Valley Water Customer Records

Yarra Valley Water records its customers' water usage figures on an input file. Each record on the file contains the customer number, customer name, customer address and water usage expressed in kilo litres (kl).

The company bills its customers according to the following rate:

usage <= 100 kl @$0.65/kl

usage > 100 kl @$0.65/kl for the first 100 kl and @$0.75/kl for the remaining.

Example: Yarra Valley Water Customer Records

A program is to be written to read the input file and produce a report listing each customer's number, name, address, water usage and the amount owing.

At the end of the report, print the total number of customers and the total amount owing to the company.

Example (cont)

1. **State the input, output and processing components of problem definition**
2. **State the control structures required**
3. **Write the algorithm**
4. **Do a desktop testing on the algorithm**

Program Design

High Level Design

Object-oriented design

Functional or Object Decomposition

High Level Design

- Division into subsystems
- Division into modules
- Division into functions or objects
- Internal functional or object-oriented design

Object-Oriented Design Methodology

- Object decomposition
  - Identify first objects and classes of objects
    - **Possible inheritance**
  - Then identify the functions operating on the objects and classes
  - First we will look at functional decomposition from the procedural point of view.

Functional Decomposition

- The division of a problem into separate tasks or functions as the first step towards designing the solution algorithm
- Each of these tasks may be looked at to identify further subtasks, and so on
- Each function will be dedicated to performing a single specific task
- A top-down design, stepwise refinement

Main Steps in Functional Decomposition

1. Group the activities (in the processing) into subtasks or functions
2. Construct a hierarchy chart
3. Write the algorithm for the top level
4. Develop the algorithm for each successive refinement

Yarra Valley Water Customer Records (revisited)

- Using functional decomposition design methodology
- Group the activities (in the processing) into subtasks or functions
- Construct a hierarchy chart

1(a) i. State Input

- customer_number
- customer_name
- customer_address
- water_usage

1(a) ii. State Output

- customer_number
- customer_name
- customer_address
- water_usage
- amount_owing
- total_customers
- total_amount_owing

1(a) iii. State Processing

- Read customer records
- Calculate amount owing
- Print customer details
- Increment total customers
- Update total amount owing
- Print totals

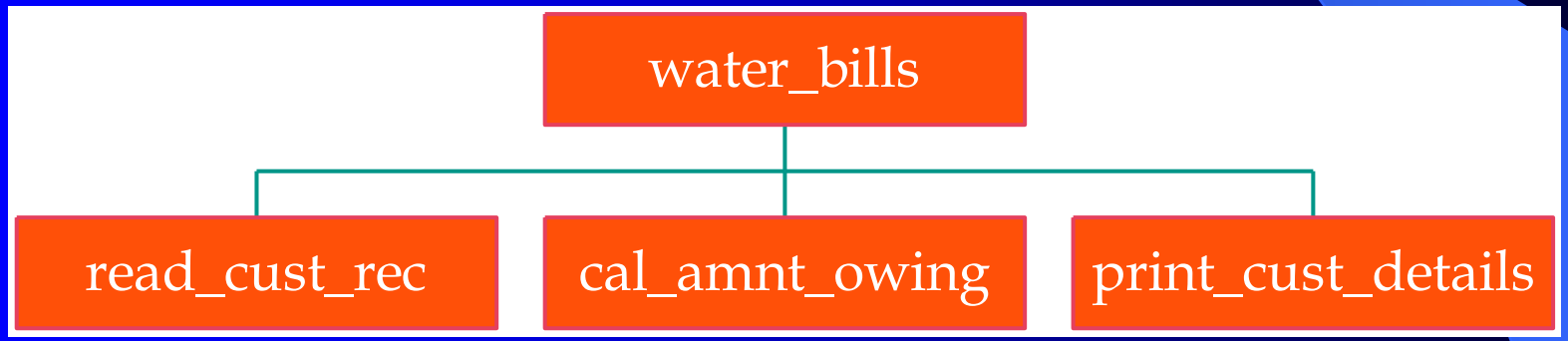
1(b) State the control structures

- Repetition: a while loop for reading customer records (into an *object*), calculating amount owing, printing customer details, increment the counters (total customers & total amount owing)
- Selection: if statements to calculate amount owing
- Sequence: Print totals

1(c) Group activities into functions

1. Read customer records
2. Calculate amount owing
3. Print customer details

2. Construct a hierarchy chart



3. Algorithm for the top level

- **Write the algorithm for the top-level "main" function**

program water_bills

Initialise total_customers, total_amount_owing

while there are still records to be read

read_a_cust_record(input_file, cust_record)

cust_record.amount_owing =

calculate_amount_owing(cust_record.water_usage)

print_customer_details(cust_record)

Update total_amount_owing

Increment total_customers

Print total_customers

Print total_amount_owing

4. Develop the algorithm for each successive refinement

method read_a_cust_record(in/out: input_file,
in/out: cust_record)

Read from file cust_record.number

Read from file cust_record.name

Read from file cust_record.address

Read from file cust_record.water_usage

4. Develop the algorithm for each successive refinement

```
function calculate_amount_owing( in: usage )
    if usage <= 100 then
            amnt_owing = usage * 0.65
    else
       amnt_owing = (100 * 0.65) + (usage - 100) * 0.70
   return amnt_owing
```

4. Develop the algorithm for each successive refinement

```
function print_customer_details( in: cust_record )
        Print cust_record.number
        Print cust_record.name
        Print cust_record.address
        Print cust_record.water_usage
        Print cust_record.amount_owing
```