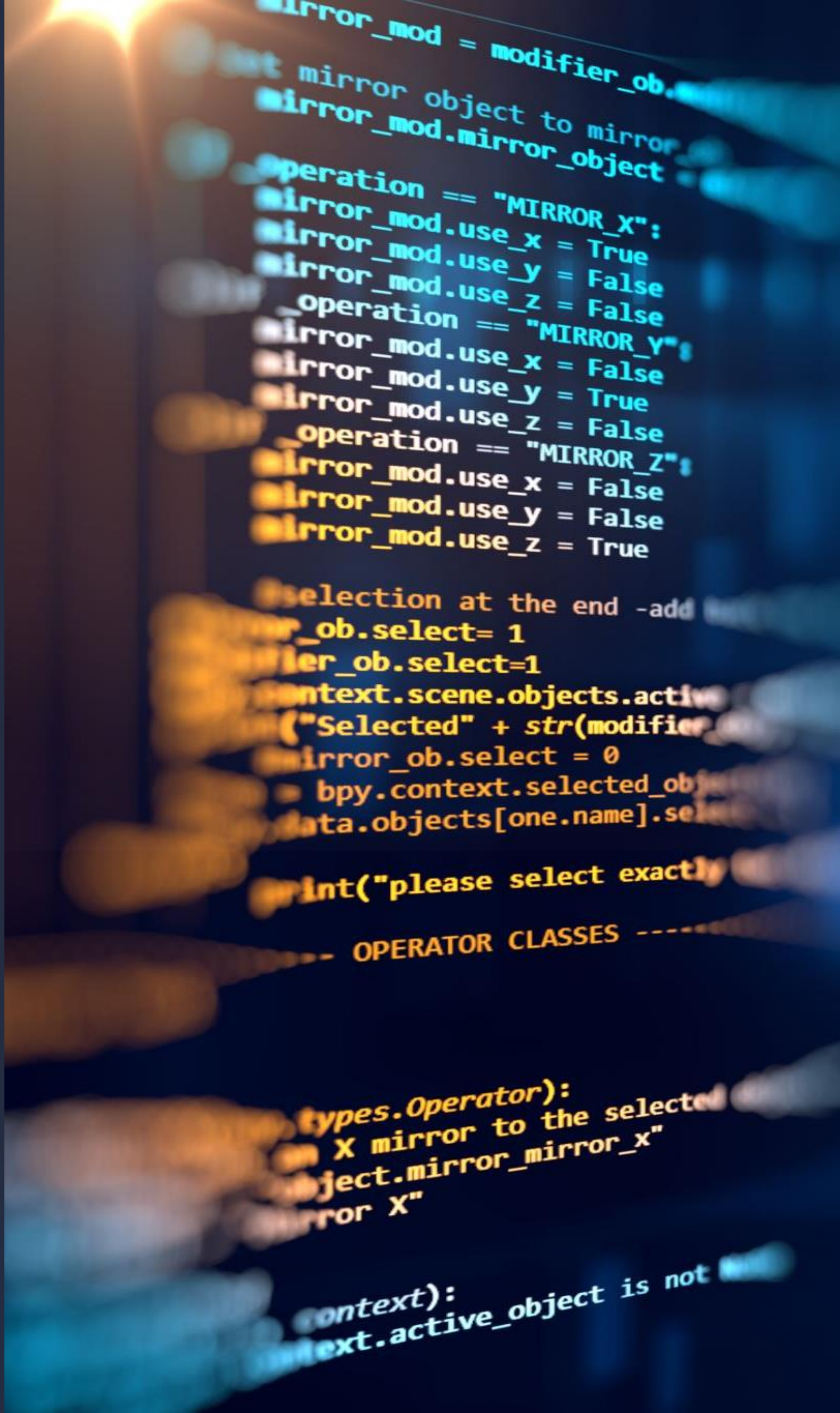
CSE216: SOFTWARE ENTERPRISE: PERSONAL PROCESSES AND QUALITY

Assist. Prof.
Dr. Noha El-Sayad



WEEK 3

- Process Measurement and Overview of Testing

Process Measurement

Measuring your process will not improve it.
You must make process changes to achieve lasting improvement.

Process Measurement: Purposes

1. understand and manage change
2. predict or plan for the future
3. compare one product, process, or organization with another
4. determine adherence to standards
5. provide a basis for control

Process Measurements: in PSP

The basic PSP data are

- program size
- time spent by phase
- defects found and injected by phase

Both actual and estimated data are gathered on every item

Measures derived from these data

- support planning
- characterize process quality

The basic PSP data are

PSP Size Measures

- define a consistent size measure
- Establish a basis for normalizing time and defect data
- Help make better size estimates

PSP Time Measures

- determine how much time you spend in each PSP phase
- help you to make better time estimates

PSP Defect Measures

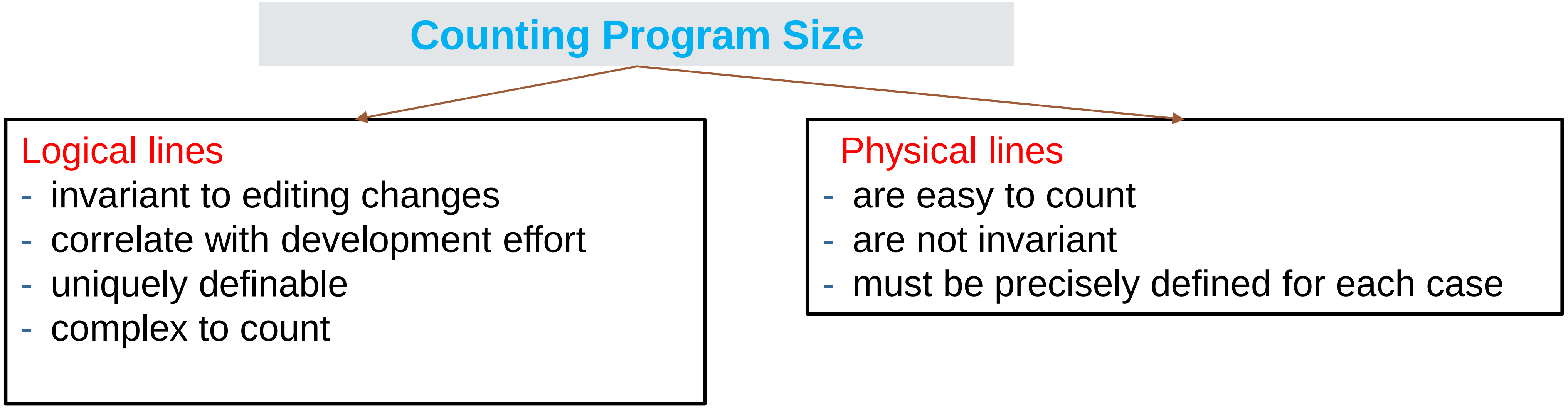
- provide a historical baseline of defect data
- understand the numbers and types of defects injected
- understand the relative costs of removing defects in each PSP phase

There are many possible measures

1. database elements
2. lines of code (LOC)
3. function points
4. pages, screens, scripts, reports

LOC Measurement

- LOC measure uses logical (versus physical) lines of code
  - A. Statement specifications
    1. Executable
    2. Nonexecutable
    3. counted statement types
  - B. Application
    1. language and code type
    2. origin and usage



Counting Program Size

Logical lines

- invariant to editing changes
- correlate with development effort
- uniquely definable
- complex to count

Physical lines

- are easy to count
- are not invariant
- must be precisely defined for each case

The PSP uses a coding standard and a physical counter for LOC size measures

- By define coding standard and physical line for each logical line
- This standard must be faithfully followed
- Then, physical line counting equals logical line counting

PSP’s LOC Counting Standard

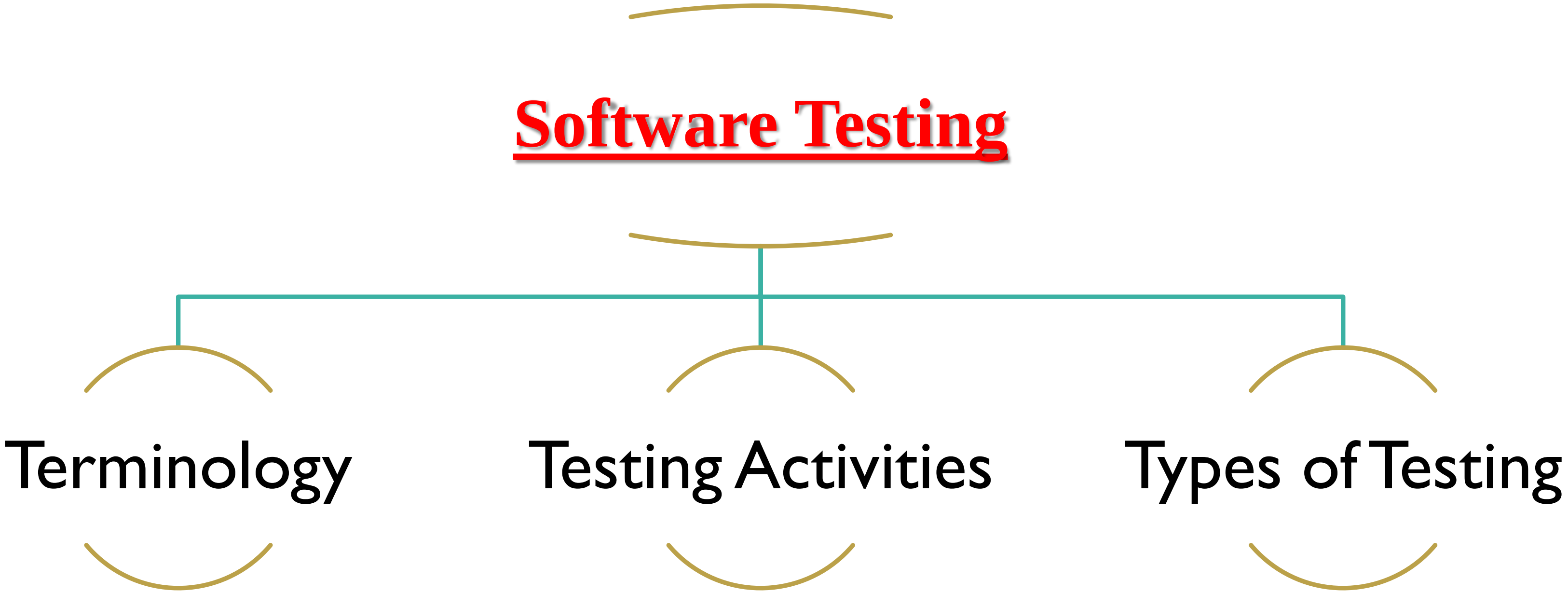
Count all statements

1. This includes begin, end, if, then, else, {, }, ;, ., declarations, directives, headers, etc.
2. Do not count blanks, comment lines, or automatically generated code
3. Count added and modified code for measuring and estimating development productivity

For small products, size tracking can be done manually, but it requires care

For larger products, size tracking requires an accounting system

Size accounting provides an orderly and precise way of tracking size changes through multiple product versions



Software Testing

Terminology

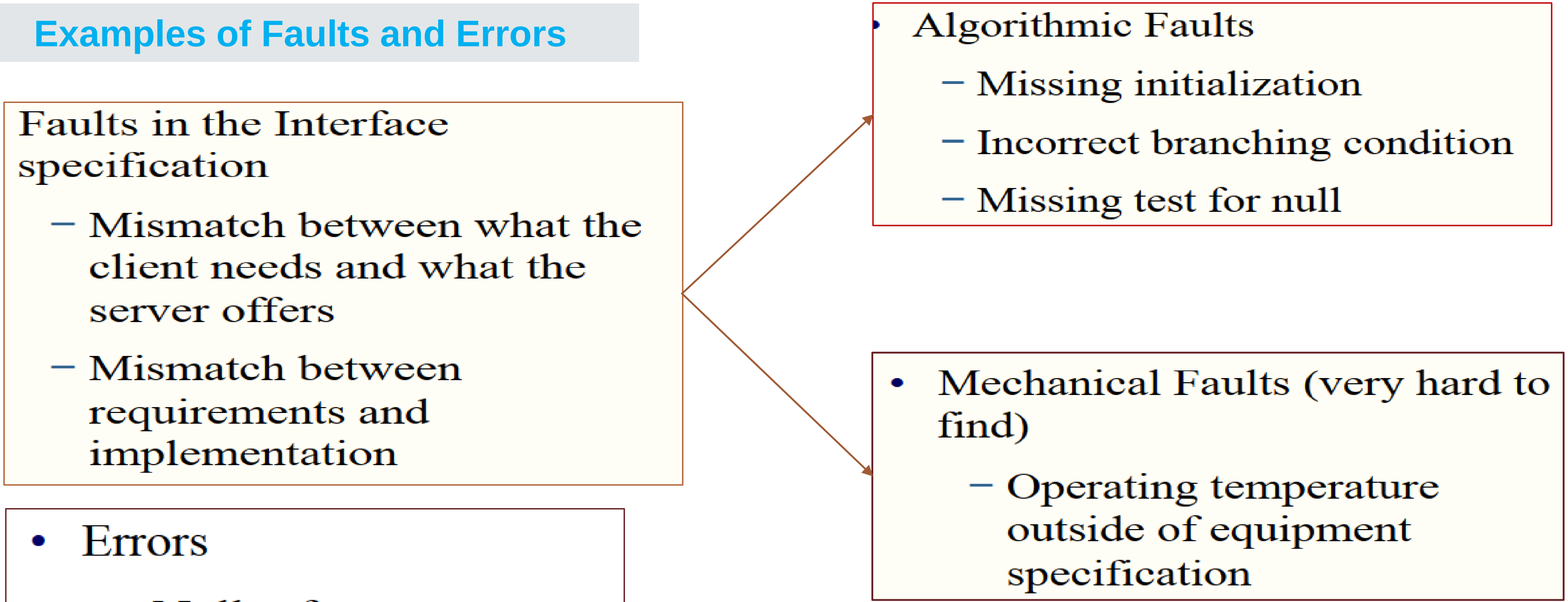
Testing Activities

Types of Testing

Terminology

Failure:	Any deviation of the observed behavior from the specified behavior
Erroneous state (“error”):	The system is in a state such that further processing by the system can lead to a failure
Fault (“bug” or “defect”) :	The mechanical or algorithmic cause of an error
Validation:	Activity of checking for deviations between the observed behavior of a system and its specification

Examples of Faults and Errors



Faults in the Interface specification

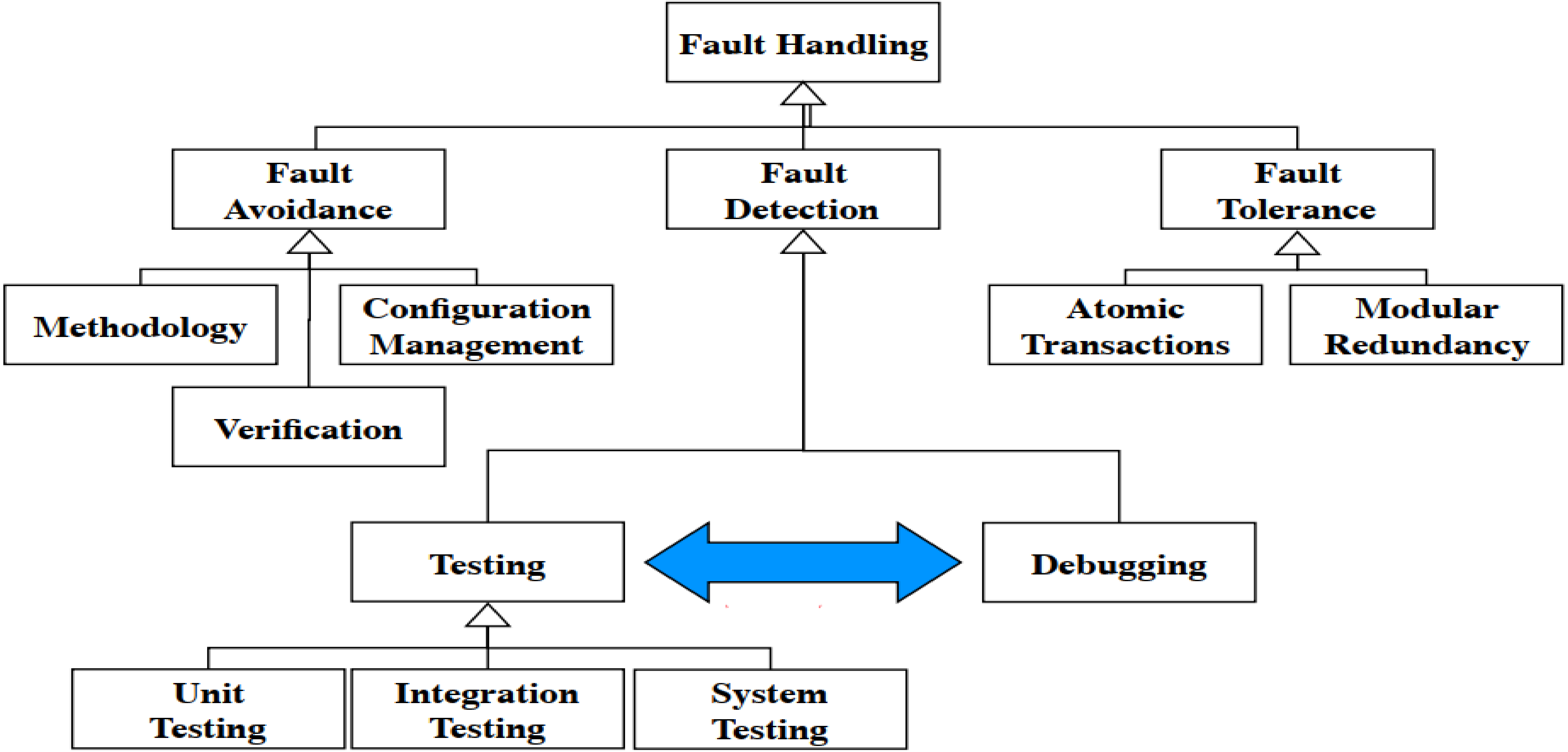
- Mismatch between what the client needs and what the server offers
- Mismatch between requirements and implementation

- Algorithmic Faults
  - Missing initialization
  - Incorrect branching condition
  - Missing test for null

- Mechanical Faults (very hard to find)
  - Operating temperature outside of equipment specification

- Errors
  - Null reference errors
  - Concurrency errors
  - Exceptions

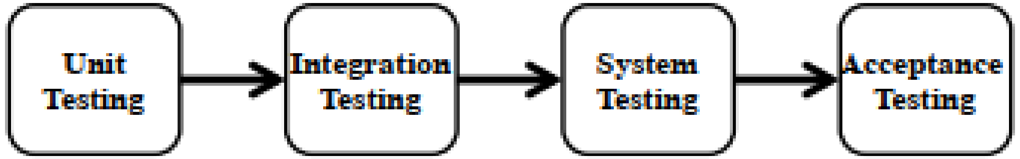
Taxonomy for Fault Handling Techniques



Testing takes creativity

1. To develop an effective test, one must have:
    - Detailed understanding of the system
    - Application and solution domain knowledge
    - Knowledge of the testing techniques
    - Skill to apply these techniques
2. Testing is done best by independent testers
    - We often develop a certain mental attitude that the program should behave in a certain way when in fact it does not
    - Programmers often stick to the data set that makes the program work
    - A program often does not work when tried by somebody else

Types of Testing



Unit Testing → Integration Testing → System Testing → Acceptance Testing

Unit Testing

- Individual component (class or subsystem)
- Carried out by developers
- Goal: Confirm that the component or subsystem is correctly coded and carries out the intended functionality

Integration Testing

- Groups of subsystems (collection of subsystems) and eventually the entire system
- Carried out by developers
- Goal: Test the interfaces among the subsystems.

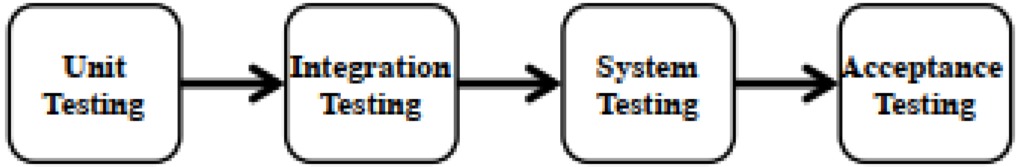
System Testing

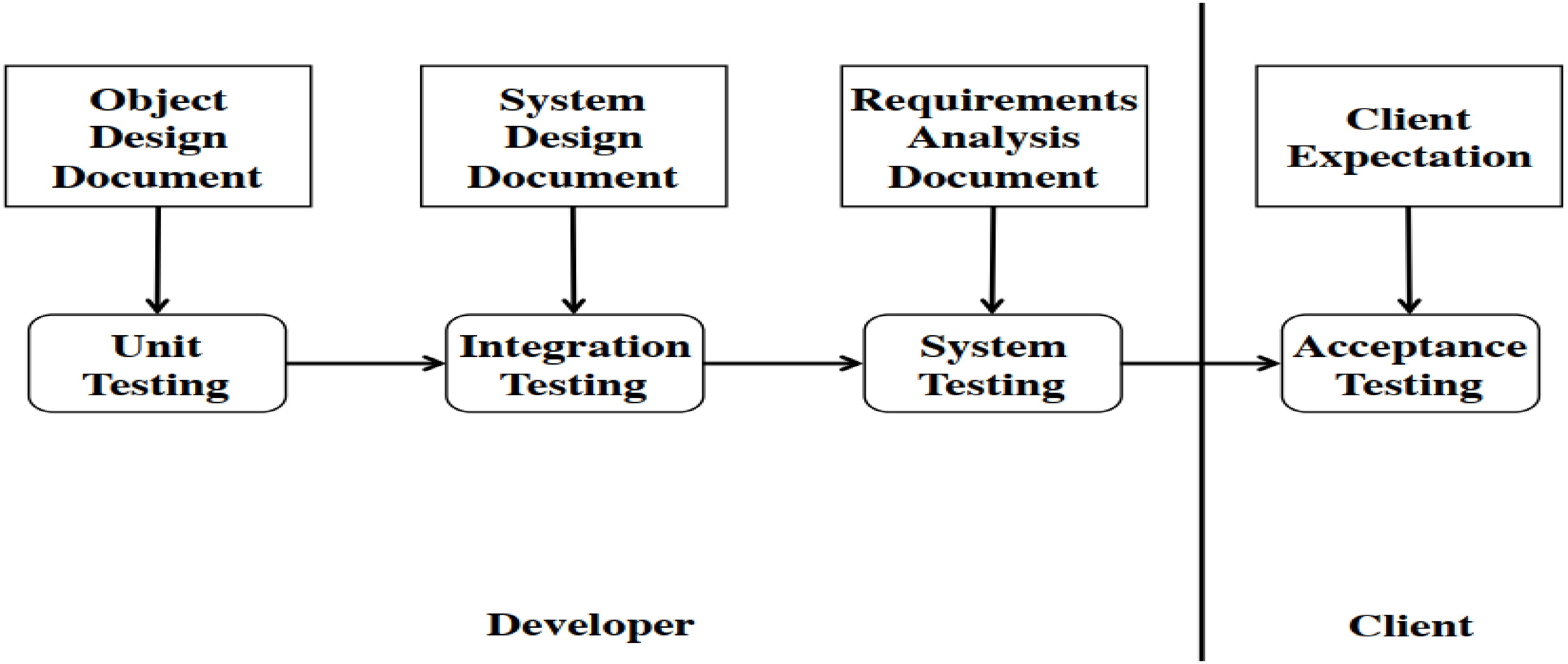
- The entire system
- Carried out by developers
- Goal: Determine if the system meets the requirements (functional and nonfunctional)

Acceptance Testing

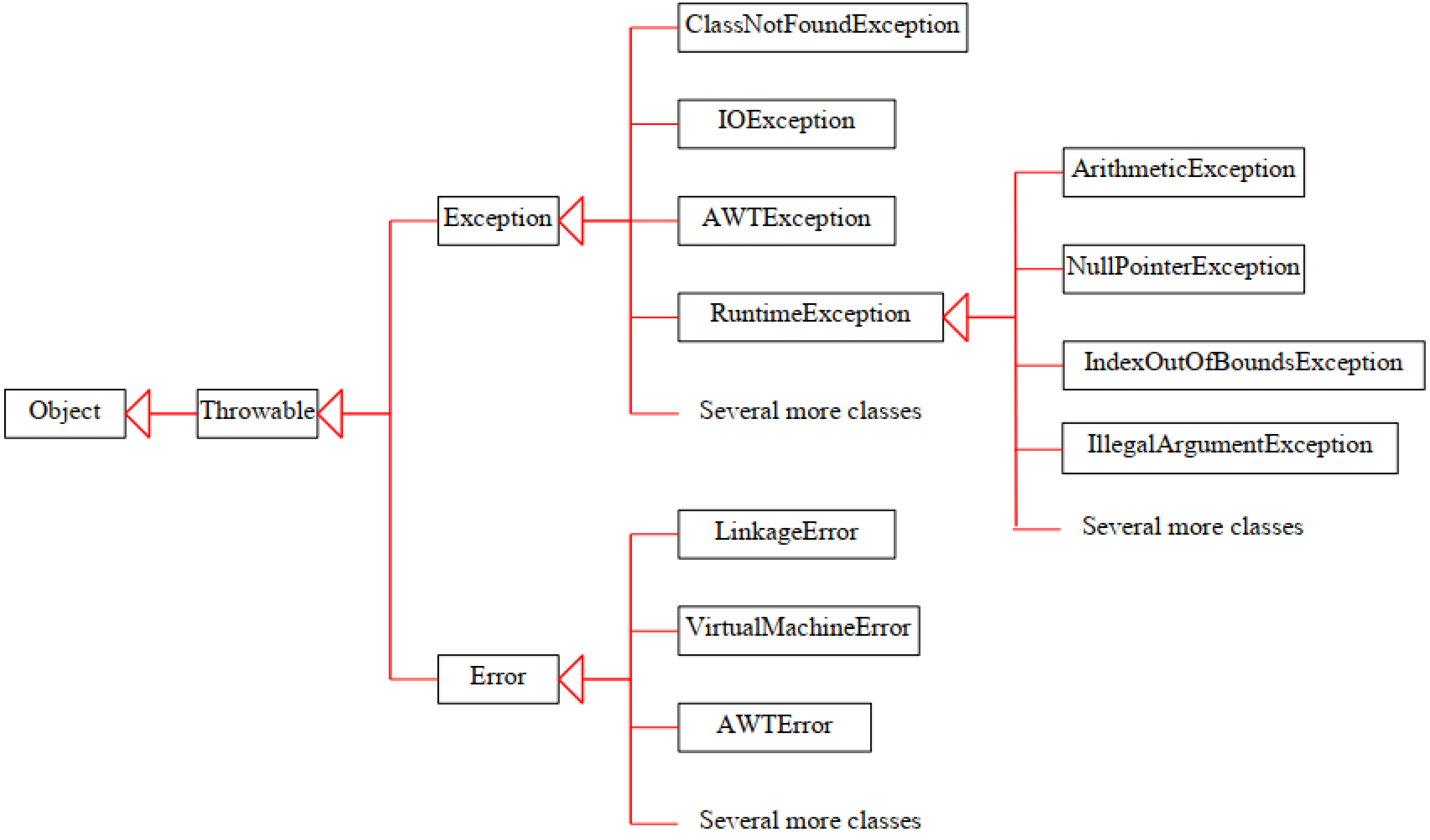
- Evaluates the system delivered by developers
- Carried out by the client. May involve executing typical transactions on site on a trial basis
- Goal: Demonstrate that the system meets the requirements and is ready to use.

Testing Activities



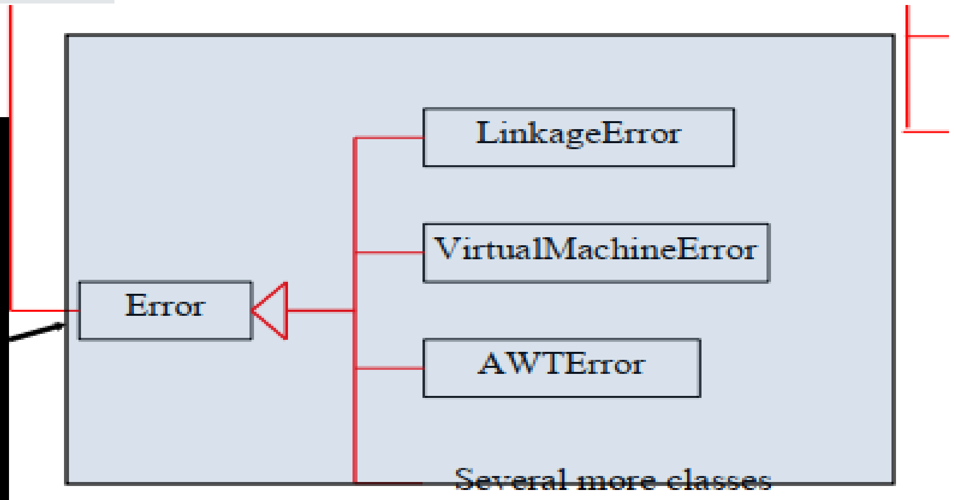


Exception Types



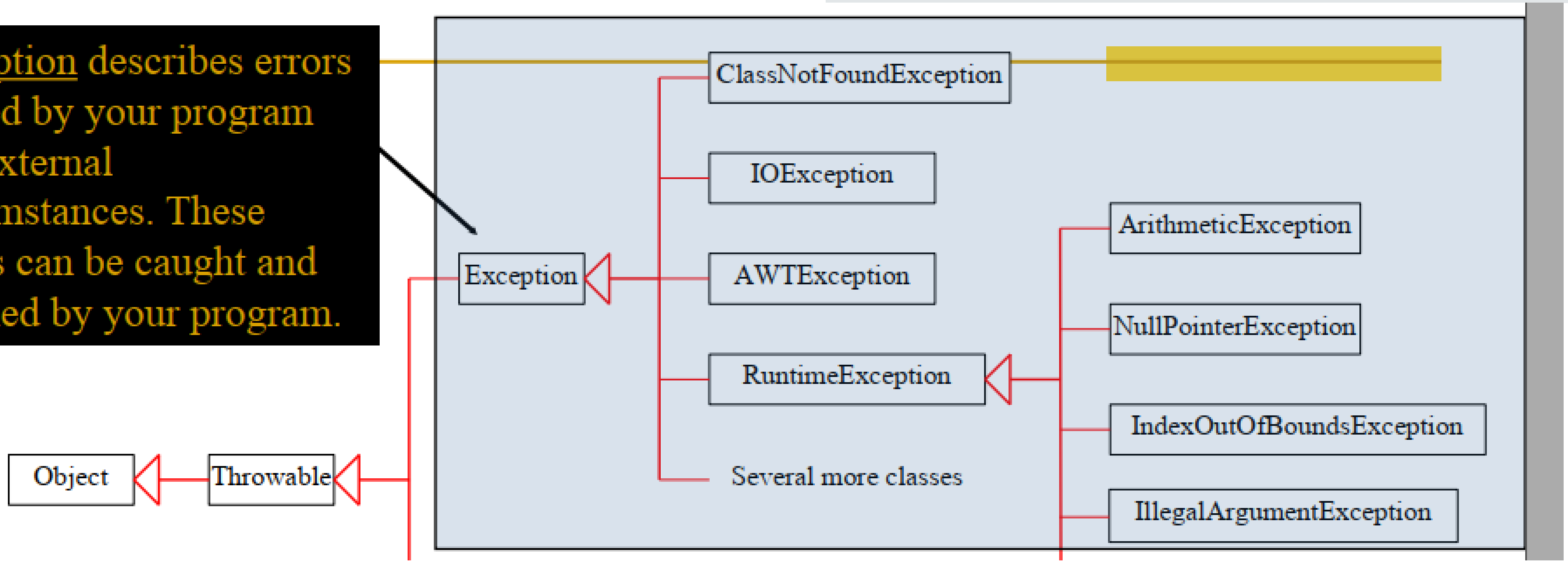
System Errors

System errors are thrown by JVM and represented in the Error class. The Error class describes internal system errors. Such errors rarely occur. If one does, there is little you can do beyond notifying the user and trying to terminate the program gracefully.

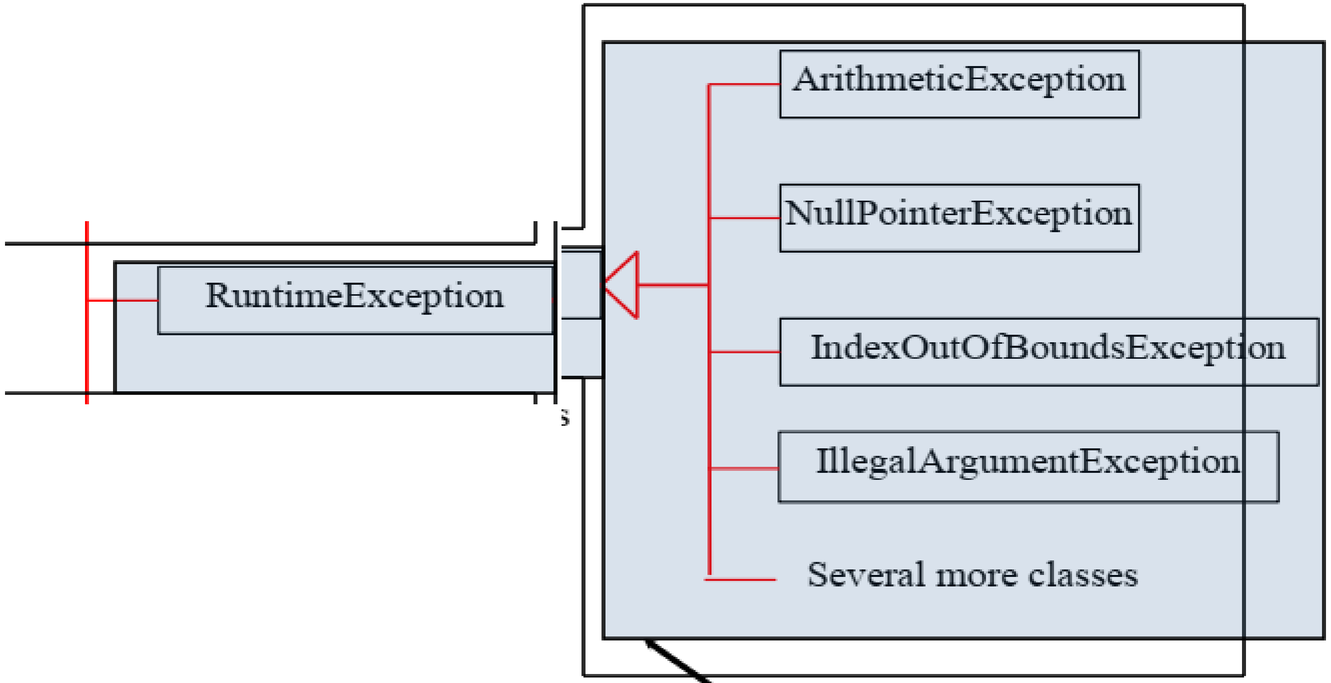


Exceptions

Exception describes errors caused by your program and external circumstances. These errors can be caught and handled by your program.

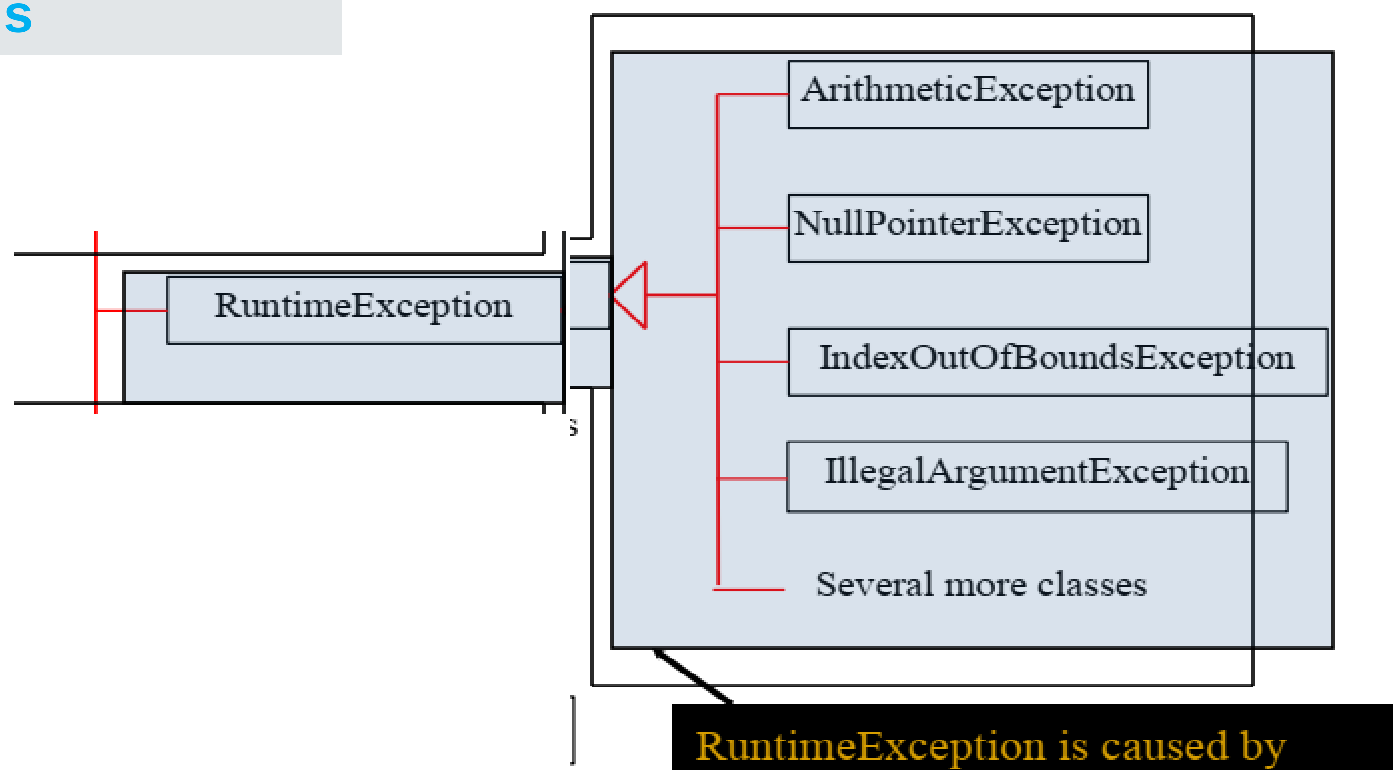


Runtime Exceptions



RuntimeException is caused by programming errors, such as bad casting, accessing an out-of-bounds array, and numeric errors.

Runtime Exceptions

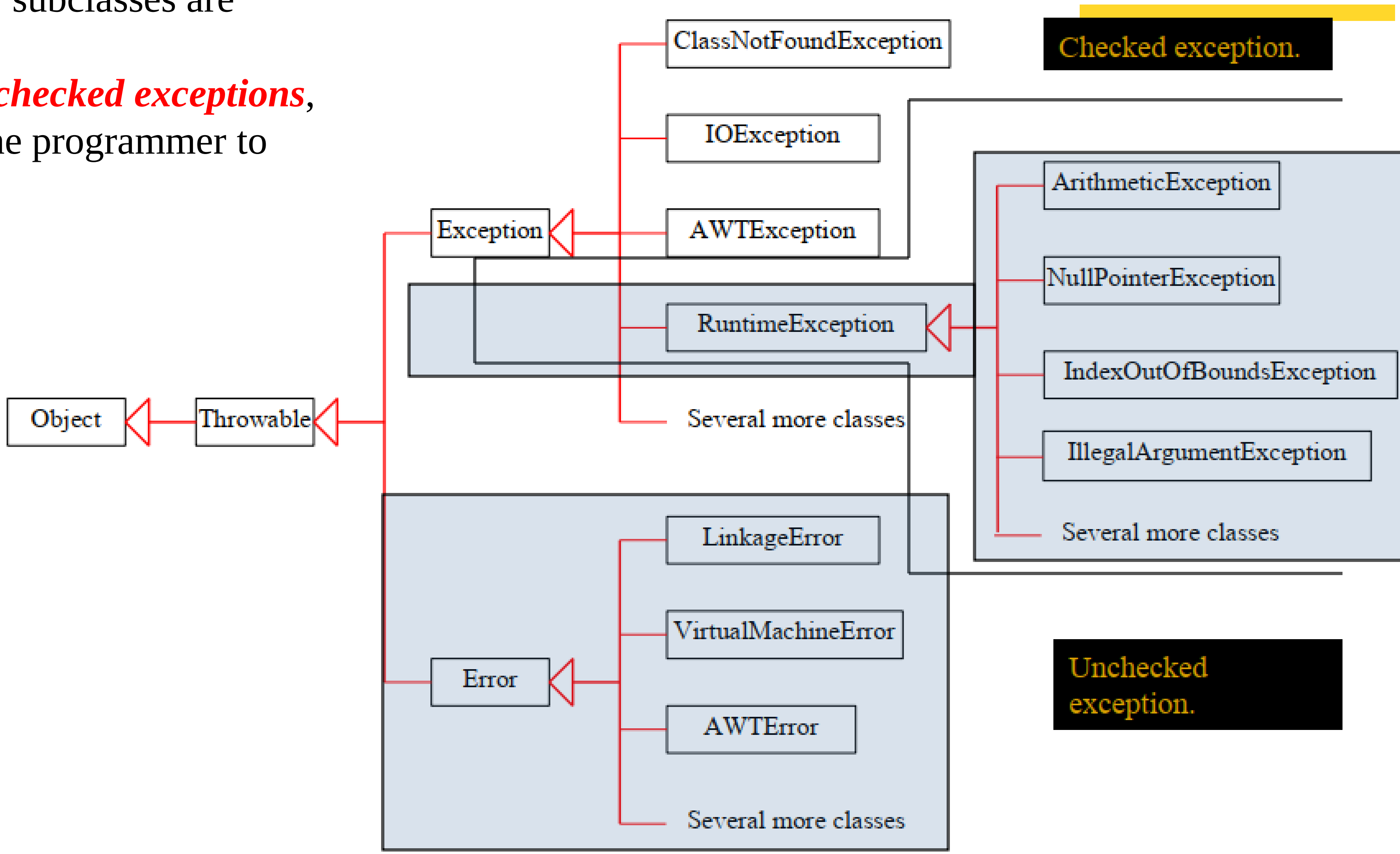


RuntimeException is caused by programming errors, such as bad casting, accessing an out-of-bounds array, and numeric errors.

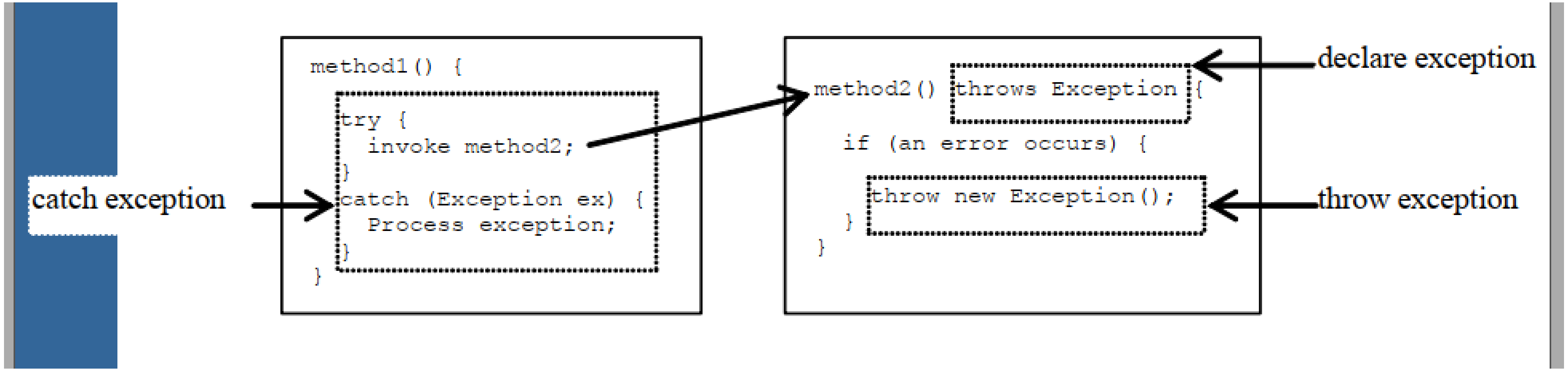
Checked Exceptions vs. Unchecked Exceptions

Runtime Exception, Error and their subclasses are known as ***unchecked exceptions***.

All other exceptions are known as ***checked exceptions***, meaning that the compiler forces the programmer to check and deal with the exceptions



Declaring, Throwing, and Catching Exceptions



catch exception

```
method1() {

  try {
    invoke method2;
  }
  catch (Exception ex) {
    Process exception;
  }
}
```

```
method2() throws Exception {

  if (an error occurs) {

    throw new Exception();
  }
}
```

declare exception

throw exception

Declaring Exceptions

Every method must state the types of checked exceptions it might throw. This is known as *declaring exceptions*.

public void myMethod()
  throws IOException

public void myMethod()
  throws IOException, OtherException

Throwing Exceptions Example

```
/** Set a new radius */
 public void setRadius(double newRadius)
      throws IllegalArgumentException
 {
   if (newRadius >= 0)
     radius =  newRadius;
   else
     throw new IllegalArgumentException(
       "Radius cannot be negative");
 }
```

Catching Exceptions

```
try {
  statements;  // Statements that may throw
  exceptions
}
catch (Exception1 exVar1) {
  handler for exception1;
}
catch (Exception2 exVar2) {
  handler for exception2;
}
...
catch (ExceptionN exVar3) {
  handler for exceptionN;
}
```